

# E-GOLDradio

PMB 7870

GSM/GPRS Single Chip Solution

CONFIDENTIAL  
[Distribution with NDA by Marketing only]

Wireless Communications



Never stop thinking.

**Edition 2005**

**Published by Infineon Technologies AG,  
St.-Martin-Strasse 53,  
D-81541 München, Germany**

**© Infineon Technologies AG 2005.  
All Rights Reserved.**

For questions on technology, delivery and prices please contact the Infineon Technologies Offices in Germany or the Infineon Technologies Companies and Representatives worldwide: see our web page at <http://www.infineon.com>.

#### **Attention Please!**

The information herein is given to describe certain components and shall not be considered as warranted characteristics.

Terms of delivery and rights to technical change reserved.

We hereby disclaim any and all warranties, including but not limited to warranties of non-infringement, regarding circuits, descriptions and charts stated herein.

Infineon Technologies is an approved CECC manufacturer.

#### **Warnings**

Due to technical requirements components may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies Office.

Infineon Technologies Components may only be used in life-support devices or systems with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system, or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body, or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

#### **Trademarks and Registered Terms**

Bluemoon<sup>®</sup>, SMARTi<sup>®</sup>, and C166<sup>®</sup> are registered trademarks of Infineon Technologies AG.  
TEAKlite is a registered trademark of ParthusCeva, Inc.

# E-GOLDradio

PMB 7870

GSM/GPRS Single Chip Solution

CONFIDENTIAL  
[Distribution with NDA by Marketing only]

Wireless Communications



Never stop thinking.

**CONFIDENTIAL**

**We Listen to Your Comments**

Do you think there is any information in this document that is wrong, unclear, or missing? Please let us know, your feedback will help us in our continuous effort to improve the quality of this document.

Please send your proposal (including a reference to this document) to:  
[smsdocu.comments@infineon.com](mailto:smsdocu.comments@infineon.com)



CONFIDENTIAL



**CONFIDENTIAL**

<b>1</b>	<b>Introduction</b>	<b>35</b>
<b>1.1</b>	<b>Overview of E-GOLDRadio</b>	<b>37</b>
<b>1.2</b>	<b>GSM/GPRS System Description</b>	<b>38</b>
<b>1.3</b>	<b>Package</b>	<b>40</b>
<b>1.4</b>	<b>Version Naming</b>	<b>41</b>
<b>1.5</b>	<b>Bus Concept</b>	<b>43</b>
1.5.1	C166S Buses	43
1.5.2	TEAKLite Bus	43
1.5.3	Bus Interconnections	43
<b>1.6</b>	<b>Clock Concept</b>	<b>43</b>
<b>1.7</b>	<b>Interrupt Concept</b>	<b>43</b>
<b>1.8</b>	<b>Debug Concept</b>	<b>43</b>
1.8.1	C166 Debug Concept	43
1.8.2	TEAKLite Debug Concept	44
<b>1.9</b>	<b>Power Management</b>	<b>44</b>
<b>1.10</b>	<b>Security Concept</b>	<b>44</b>
<b>1.11</b>	<b>Asynchronous Operation Mode Concept</b>	<b>44</b>
<b>2</b>	<b>Pin Diagram</b>	<b>45</b>
<b>2.1</b>	<b>E-GOLDRadio Ballout</b>	<b>46</b>
<b>3</b>	<b>Pin Descriptions</b>	<b>49</b>
<b>4</b>	<b>Block Diagram</b>	<b>65</b>
<b>5</b>	<b>TEAKlite DSP</b>	<b>69</b>
<b>5.1</b>	<b>Introduction</b>	<b>69</b>
<b>5.2</b>	<b>TEAKLite Memory Organization</b>	<b>70</b>
5.2.1	DSP Page Select Register	74
<b>5.3</b>	<b>Hardware Version</b>	<b>75</b>
5.3.1	DSP Identification Register	75
<b>5.4</b>	<b>Boot Address</b>	<b>76</b>
<b>5.5</b>	<b>DSP Power Saving Modes</b>	<b>76</b>
5.5.1	DSP in IDLE Mode	76
5.5.2	Clock Off	77
5.5.3	Power-Down	78
5.5.4	Power-Up	78
<b>5.6</b>	<b>Hardware Reset</b>	<b>78</b>
<b>5.7</b>	<b>DSP Debug Register</b>	<b>78</b>
<b>5.8</b>	<b>Pad Access Register</b>	<b>81</b>
<b>6</b>	<b>C166S MCU</b>	<b>83</b>
<b>6.1</b>	<b>Introduction</b>	<b>84</b>
6.1.1	C166S - A Member of 16-bit Microcontroller Family	84
6.1.2	Summary of Basic Features	84

**CONFIDENTIAL**

<b>6.2</b>	<b>System Overview</b>	<b>86</b>
6.2.1	Basic MCU Concepts	87
6.2.2	E-GOLDRadio System Resources	92
<b>6.3</b>	<b>Central Processing Unit, PEC, and Interrupt Controller</b>	<b>97</b>
6.3.1	MCU Special Function Registers	98
6.3.2	Instruction Fetch and Program Flow Control	99
6.3.3	Interrupt and Exception Execution	111
6.3.4	Using General-Purpose Registers	148
6.3.5	Data Addressing	153
6.3.6	Data Processing	167
6.3.7	Instruction Pipeline	180
6.3.8	Dedicated CSFRs	185
6.3.9	Summary of MCU Registers	186
<b>6.4</b>	<b>Address Mapping and Memory Use</b>	<b>191</b>
6.4.1	Data Organization in Memory	191
6.4.2	Crossing Memory Boundaries	192
6.4.3	Memory Organization	192
<b>6.5</b>	<b>X-Bus System Architecture</b>	<b>205</b>
6.5.1	External Bus Unit	205
6.5.2	The Internal X-Bus Interface	205
6.5.3	BUS Arbitration	209
<b>6.6</b>	<b>Instruction Set</b>	<b>210</b>
6.6.1	Instruction Summary	210
6.6.2	Opcode Tables	210
6.6.3	Instruction Set Summary	212
6.6.4	Instruction Opcodes	224
<b>6.7</b>	<b>SCU</b>	<b>231</b>
6.7.1	SCU Registers Overview	231
6.7.2	Reset Control	231
6.7.3	System Configuration Block	237
6.7.4	Watchdog Timer Submodule	242
6.7.5	System Control Block	247
6.7.6	Extended Power Management Module	259
6.7.7	Peripheral Management Module	264
6.7.8	Identification Register Block	266
<b>6.8</b>	<b>OCDS</b>	<b>272</b>
6.8.1	Introduction	272
6.8.2	Enabling and Disabling the OCDS	273
6.8.3	Reset to Halt Mode	274
6.8.4	Debug Event Sources	274
6.8.5	Debug Event Actions	277
6.8.6	Debug Registers	280
6.8.7	Reset Behavior	291

**CONFIDENTIAL**

<b>6.9</b>	<b>Cerberus</b>	291
6.9.1	Operational Overview	291
6.9.2	Registers	298
6.9.3	Operation Modes	304
6.9.4	Error Handling	310
6.9.5	System Security	310
6.9.6	Power Saving	311
6.9.7	Reset Behavior	311
<b>6.10</b>	<b>Configuring External Bus and MCU Signals</b>	313
6.10.1	General	313
6.10.2	Data Bus Functions Pins D(7:0) and D(15:8)	313
6.10.3	Address Bus Functions	315
6.10.4	Port D Functions	316
6.10.5	Chip Select and HOLD Functions	316
6.10.6	RD, WR, and READY Pins	316
6.10.7	Byte High Enable (BHE)	317
6.10.8	RSTOUT and CLKOUT	317
6.10.9	Visible Mode	318
<b>7</b>	<b>Functional Description of RF Part</b>	321
<b>7.1</b>	<b>Pin Definition and Function</b>	323
<b>7.2</b>	<b>Functional Block Description</b>	325
7.2.1	Receiver	325
7.2.2	Transmitter	328
7.2.3	RF-Synthesizer	328
7.2.4	Power Up Sequencer	328
7.2.5	3-Wire Serial Bus	332
7.2.6	DCXO	332
7.2.7	RF Supply Voltage Domains	334
<b>8</b>	<b>TEAKlite Bus</b>	337
<b>8.1</b>	<b>TEAKLite Interrupt Unit</b>	337
8.1.1	Functional Overview	337
8.1.2	Register Descriptions	341
<b>8.2</b>	<b>Timer</b>	353
8.2.1	Functional Overview	353
8.2.2	DSP Timer 1	353
8.2.3	DSP Timer 2	357
<b>8.3</b>	<b>Viterbi Coprocessor, Channel Decoder</b>	361
8.3.1	Interface	361
<b>8.4</b>	<b>Bi-directional Serial Audio Interface I2S1 (4-pins)</b>	375
8.4.1	Functional Overview	376
8.4.2	Structural Overview	377
8.4.3	Operating Mode Generalities	378

**CONFIDENTIAL**

8.4.4	Serial Audio Data Transmission in Normal Mode	379
8.4.5	PCM Mode Operation of I2S interface	380
8.4.6	DAI Mode Operation of I2S Interface	383
8.4.7	Clock Configuration	385
8.4.8	Interface to TEAKLite Bus	399
8.4.9	I2S Control and Status Registers	403
8.4.10	I2S Receiver Registers	411
8.4.11	I2S Transmitter Registers	413
<b>8.5</b>	<b>Bi-directional Serial Audio Interface I2S2 (6-pins)</b>	<b>417</b>
<b>8.6</b>	<b>Unidirectional Serial Audio Interface I2S3</b>	<b>419</b>
8.6.1	Functional Overview	419
8.6.2	Structural Overview	420
8.6.3	Operating Mode Generalities	421
8.6.4	Serial Audio Data Transmission in Normal Mode	422
8.6.5	PCM Mode Operation of I2S interface	422
8.6.6	Clock Configuration	425
8.6.7	Interface to TEAKLite Bus	436
8.6.8	I2S Control and Status Registers	439
8.6.9	I2S Transmitter Registers	443
<b>8.7</b>	<b>Synchronous Serial Controller on DSP</b>	<b>447</b>
8.7.1	Functional Overview	447
8.7.2	Register Overview	448
8.7.3	Structural Overview	449
8.7.4	Operating Modes	456
8.7.5	General Operation	462
8.7.6	Error Detection Mechanisms	475
<b>8.8</b>	<b>GMSK Modulator</b>	<b>479</b>
8.8.1	Functional Overview	479
8.8.2	Register Overview	479
8.8.3	Modulator Block Overview	481
8.8.4	Clocking Issues of the Modulator	481
8.8.5	Programming Description	482
8.8.6	Register Description of the Modulator	484
8.8.7	Analog Domain of the Modulator	490
8.8.8	Baseband Spectrum Requirements	492
<b>8.9</b>	<b>Baseband Receive Unit</b>	<b>497</b>
8.9.1	General Description	498
8.9.2	Register Overview	499
8.9.3	CORDIC and DC-Offset	501
8.9.4	Adaptive Switchable FIR Filter	501
8.9.5	Interface to Dual-Port RAM	505
8.9.6	Register Description	507
8.9.7	Analog Pre-Filters	515

**CONFIDENTIAL**

8.9.8	The SD Analog-to-Digital Converter for Baseband Signals	515
8.9.9	Filter Coefficients	518
<b>8.10</b>	<b>Audio Front-End</b>	<b>521</b>
8.10.1	Functional Overview	523
8.10.2	Digital Part	525
8.10.3	Interface to DSP Bus and C166	525
8.10.4	Analog Part	529
<b>8.11</b>	<b>GSM Cipher Unit</b>	<b>545</b>
8.11.1	Introduction	545
8.11.2	A51 and A52	546
8.11.3	A53	552
8.11.4	Ciphering Hardware Block Selection	564
<b>8.12</b>	<b>Viterbi Coprocessor, Equalizer</b>	<b>567</b>
8.12.1	Interface	567
8.12.2	Flow of Equalization Process	585
8.12.3	Architecture	591
<b>8.13</b>	<b>Shared Memory</b>	<b>593</b>
<b>8.14</b>	<b>Multicore Synchronization</b>	<b>593</b>
<b>8.15</b>	<b>OCEM/SEIB</b>	<b>595</b>
8.15.1	Functional Overview	595
8.15.2	Program Address Break Point Operation	595
8.15.3	Pipe Break Point Operation	596
8.15.4	Data Address Break Point Operation	596
8.15.5	Data Value Break Point Operation	597
8.15.6	External Registers Break Point	597
8.15.7	Clarifications for the Data Value/Address Break Point	598
8.15.8	Illegal Break Point	598
8.15.9	DSP Monitor Program	598
8.15.10	OCEM Physical Interface	599
8.15.11	OCEM Register Interface	601
8.15.12	Enabling DSP Debugging	613
8.15.13	SEIB Block	613
<b>9</b>	<b>LM-Bus</b>	<b>615</b>
<b>9.1</b>	<b>RAM</b>	<b>615</b>
<b>9.2</b>	<b>ROM</b>	<b>616</b>
<b>10</b>	<b>X-Bus</b>	<b>617</b>
<b>10.1</b>	<b>X-Bus Description</b>	<b>617</b>
10.1.1	X-Bus Interface	617
10.1.2	8- and 16-bit Access to X-Bus Peripherals	618
10.1.3	Reset Behavior of X-Bus	618
<b>10.2</b>	<b>Shared Memory</b>	<b>621</b>
10.2.1	Size	621

**CONFIDENTIAL**

10.2.2	Access	623
<b>10.3</b>	<b>Multicore Synchronization</b>	<b>625</b>
10.3.1	General	625
10.3.2	Cross Software interrupts	627
10.3.3	Communication Flags	631
10.3.4	Semaphore Flags	637
10.3.5	XBIU	643
<b>10.4</b>	<b>PLL - CGU - SCCU</b>	<b>651</b>
10.4.1	Clock Generation Unit	653
10.4.2	Standby Clock Control Unit (SCCU)	685
<b>10.5</b>	<b>Measurement Interface</b>	<b>715</b>
10.5.1	Functional Overview	716
10.5.2	Register Overview	717
10.5.3	Measurement Circuit and Mode Setting	718
10.5.4	Differential General Purpose Measurements (MxMy)	722
10.5.5	Single-Ended General Purpose Measurements (Mx)	726
10.5.6	On-Chip Temperature Measurement	728
10.5.7	Modulator Unit Offset Measurement TXOFI and TXOFQ	729
10.5.8	Offset Calibration of Measurement Interface	731
10.5.9	BPI Bus Register Descriptions	732
10.5.10	Special Operating Modes and Examples	744
10.5.11	System Block Diagram of Measurement Interface	747
10.5.12	Clock Control of Measurement Interface	748
<b>10.6</b>	<b>Analog Control Registers</b>	<b>750</b>
10.6.1	Functional Overview	750
10.6.2	Analog Control Register 1	751
10.6.3	Analog Control Register 2	753
<b>10.7</b>	<b>Keypad</b>	<b>755</b>
10.7.1	Description	755
10.7.2	Keypad Registers	757
10.7.3	Keypad Port	758
<b>10.8</b>	<b>SIM Interface</b>	<b>761</b>
10.8.1	Functional Overview	761
10.8.2	SIM Register Overview	764
10.8.3	Register Description	764
10.8.4	SIM Card Interface	776
10.8.5	Clock Control	776
10.8.6	SIM Activation and Deactivation	778
10.8.7	Initialization Sequence Overview	779
10.8.8	Automatic Power Down	779
10.8.9	SIM Character Mode	781
10.8.10	SIM T=0 Protocol Mode	782
10.8.11	Connection of 5 V, 3 V, and 1.8 V SIM Cards	785

**CONFIDENTIAL**

10.8.12	<b>SIM Card Pads</b>	785
<b>10.9</b>	<b>AFC</b>	787
10.9.1	Introduction	787
10.9.2	Use of AFCVAL Register for Standard Applications	789
10.9.3	Use of AFCVAL Register for Special Applications	789
10.9.4	AFC Identification Register	791
<b>10.10</b>	<b>RF Power Ramping</b>	793
10.10.1	Introduction	793
10.10.2	Application Notes	800
10.10.3	System Register	800
<b>10.11</b>	<b>GSM Timer Unit</b>	803
10.11.1	Overview	803
10.11.2	GSM Clock Control Unit	828
10.11.3	GSM Timer Decoder	833
<b>10.12</b>	<b>RF Control</b>	835
10.12.1	Introduction	835
10.12.2	RF RAM	836
10.12.3	Synchronous Serial Interface (3-wire RF interface)	842
10.12.4	Control Block	847
<b>10.13</b>	<b>External Bus Unit</b>	851
10.13.1	Introduction	852
10.13.2	Single-Chip Mode	853
10.13.3	External Bus Modes	854
10.13.4	Programmable Bus Characteristics	864
10.13.5	Controlling the External Bus Controller	869
10.13.6	EBC Idle State	876
10.13.7	Page Mode Flash Control Unit	877
10.13.8	EBU Application Notes	893
<b>10.14</b>	<b>Logic Arranger</b>	915
10.14.1	Introduction	915
10.14.2	Functional Description	916
10.14.3	LPSA Registers	918
<b>10.15</b>	<b>GPRS Cipher Unit</b>	921
10.15.1	GPRS_GEA3 Overview	922
10.15.2	Functional Overview	922
10.15.3	Physical Interface	933
10.15.4	Interrupts	933
10.15.5	MCU Register Interface for GPRS_GEA1/2	934
10.15.6	MCU Register Interface for GPRS_GEA3	939
<b>11</b>	<b>PD-Bus</b>	947
<b>11.1</b>	<b>I2C Bus Interface</b>	948
11.1.1	Introduction	948

**CONFIDENTIAL**

11.1.2	Operational Overview	949
11.1.3	Functional Overview	952
11.1.4	Registers	954
11.1.5	Reset Behavior	972
11.1.6	Interrupts	973
11.1.7	Synchronization	974
11.1.8	Programming	974
<b>11.2</b>	<b>Synchronous Serial Interface on PD Bus</b>	<b>977</b>
11.2.1	Introduction	977
11.2.2	General Operation	979
11.2.3	SSC Kernel Registers	995
11.2.4	Interrupts	1008
<b>11.3</b>	<b>ASC0</b>	<b>1011</b>
11.3.1	ASC0 Description	1012
11.3.2	Features of the ASC0 Controller	1012
11.3.3	Functional Description of the ASC0	1013
11.3.4	Operational Overview	1014
11.3.5	General Operation	1014
11.3.6	Registers	1047
<b>11.4</b>	<b>ASC1</b>	<b>1067</b>
11.4.1	Features of the ASC1 Controller	1068
11.4.2	Functional Description of the ASC1	1069
11.4.3	Registers	1070
<b>11.5</b>	<b>CAPCOM 1 and 2</b>	<b>1083</b>
11.5.1	Introduction	1084
11.5.2	Operational Overview	1084
11.5.3	Functional Overview	1085
11.5.4	Registers	1100
11.5.5	Interrupts	1118
<b>11.6</b>	<b>RTC</b>	<b>1121</b>
11.6.1	Introduction	1122
11.6.2	RTC Register Overview	1123
11.6.3	RTC Shell	1123
11.6.4	32k Oscillator	1128
11.6.5	RTC Macro	1130
11.6.6	RTC Power Supply Concept	1144
<b>11.7</b>	<b>GPT 1 and 2</b>	<b>1145</b>
11.7.1	Introduction	1145
11.7.2	Overview	1146
11.7.3	Kernel Description	1147
11.7.4	Interrupts	1189
<b>11.8</b>	<b>Port Control Logic</b>	<b>1191</b>
11.8.1	Functional Overview	1192



**CONFIDENTIAL**

11.8.2	Register Description PCL_<pad> .....	1195
11.8.3	IDMX .....	1198
11.8.4	OMX .....	1198
11.8.5	DIRCTL .....	1199
11.8.6	General Purpose I/O .....	1200
11.8.7	Control Registers .....	1202
11.8.8	Flash Overwrite Protection .....	1203
11.8.9	Fuse Boxes and Corresponding Registers .....	1205
11.8.10	Internal Signal Monitoring .....	1209
<b>11.9</b>	<b>TAP Controller and Break Switch .....</b>	<b>1243</b>
11.9.1	Introduction .....	1243
11.9.2	Break Switch .....	1244
11.9.3	JTAG .....	1248
<b>12</b>	<b>Register Lists and Mapping .....</b>	<b>1265</b>
<b>12.1</b>	<b>TEAKLite Bus Register Addresses .....</b>	<b>1265</b>
12.1.1	DSP Memory Maps .....	1265
12.1.2	Registers in the DSP Memory Space .....	1266
<b>12.2</b>	<b>PD-Bus Register Addresses .....</b>	<b>1273</b>
12.2.1	Register Addresses .....	1273
<b>12.3</b>	<b>X-Bus Register Addresses .....</b>	<b>1285</b>
<b>13</b>	<b>Electrical and Temperature Characteristics .....</b>	<b>1297</b>
<b>13.1</b>	<b>Maximum Values (Destruction limits) .....</b>	<b>1300</b>
13.1.1	Maximum ESD .....	1300
13.1.2	Maximum Temperature .....	1300
13.1.3	Maximum Voltages .....	1301
13.1.4	Maximum Current .....	1301
13.1.5	Absolute Maximum Ratings .....	1302
<b>13.2</b>	<b>Normal Operation Values .....</b>	<b>1303</b>
13.2.1	Baseband Electrical Data .....	1303
13.2.2	Mixed Signals and Other Values .....	1362
13.2.3	RF Electrical Data .....	1383
<b>14</b>	<b>System Reset .....</b>	<b>1401</b>
<b>14.1</b>	<b>Introduction .....</b>	<b>1401</b>
14.1.1	Software Reset .....	1401
14.1.2	Watchdog Timer Reset .....	1401
14.1.3	Reset of RTC, SIM Cards, DSP, and Analog .....	1402
14.1.4	CGU Reset Block .....	1402
<b>14.2</b>	<b>Reset .....</b>	<b>1404</b>
14.2.1	Reset Timing Diagrams .....	1405
14.2.2	Power On/Off Sequences .....	1407
14.2.3	CPU Boot Configuration .....	1411

**CONFIDENTIAL**

<b>15</b>	<b>Debug</b>	<b>1413</b>
<b>16</b>	<b>Software Boot Code</b>	<b>1415</b>
<b>16.1</b>	<b>Functionality</b>	<b>1415</b>
16.1.1	Features	1415
16.1.2	Utility Routines Interface	1416
<b>17</b>	<b>Application</b>	<b>1421</b>
<b>17.1</b>	<b>Circuits</b>	<b>1422</b>
<b>17.2</b>	<b>System Block Diagram</b>	<b>1423</b>
<b>17.3</b>	<b>External 26 MHz Clock Application Circuit</b>	<b>1424</b>
<b>18</b>	<b>Package Outline</b>	<b>1425</b>
<b>19</b>	<b>Glossary</b>	<b>1429</b>

## CONFIDENTIAL

Figure 1-1	Typical E-GOLDradio Application Circuit . . . . .	38
Figure 1-2	Package Label (Character height = 1.0 mm) . . . . .	40
Figure 2-1	Legend for E-GOLDradio Ballout . . . . .	46
Figure 4-1	E-GOLDradio Block Diagram . . . . .	66
Figure 4-2	Diagram of E-GOLDradio Buses . . . . .	67
Figure 5-1	TEAKLite Core Block Diagram . . . . .	69
Figure 5-2	Address Space Partitioning on TEAKLite . . . . .	71
Figure 6-1	Basic Block Diagram of a C166S System . . . . .	87
Figure 6-2	MCU Architecture . . . . .	97
Figure 6-3	Addressing via the Code Segment and Instruction Pointers . . . . .	106
Figure 6-4	Interrupt Arbitration . . . . .	113
Figure 6-5	Task Status Saved on System Stack . . . . .	118
Figure 6-6	PEC Pointer Address Handling . . . . .	127
Figure 6-7	Implicit CP Use by logical Short GPR Addressing Modes . . . . .	148
Figure 6-8	Interpretation of a 16-bit Long Address . . . . .	155
Figure 6-9	Addressing via Data Page Pointer . . . . .	156
Figure 6-10	Overriding the DPP Mechanism . . . . .	158
Figure 6-11	Physical Stack Address Generation . . . . .	166
Figure 6-12	Storage of Words, Byte and Bits in a Byte Organized Memory . . . . .	191
Figure 6-13	C166S Address Space Overview . . . . .	193
Figure 6-14	DPRAM and SFR Areas within Segment 0 (Simplified) . . . . .	195
Figure 6-15	Detailed Memory Map for Segments 0-4 . . . . .	200
Figure 6-16	Even and Odd Signals Waveforms Example . . . . .	204
Figure 6-17	Register Overview . . . . .	231
Figure 6-18	A Simplified Reset State Diagram of the Core . . . . .	235
Figure 6-19	A Simplified Reset Sequence Timing Diagram . . . . .	236
Figure 6-20	Reset Configuration Source Selection . . . . .	240
Figure 6-21	WDT Block Diagram . . . . .	242
Figure 6-22	State Machine for Security Level Switching . . . . .	250
Figure 6-23	Clock Output Signal Generation . . . . .	255
Figure 6-24	Signal Waveforms . . . . .	256
Figure 6-25	Connection to Port Logic (Functional Approach) . . . . .	257
Figure 6-26	Simplified Idle/Sleep/Power Down State Diagram . . . . .	262
Figure 6-27	Sleep + Idle Wakeup via RxD0 . . . . .	264
Figure 6-28	Clock Control on PD Bus . . . . .	265
Figure 6-29	OCDS Module Block Diagram . . . . .	272
Figure 6-30	Hardware Trigger Generation Unit . . . . .	275
Figure 6-31	Simple and Advanced Debug Model . . . . .	278
Figure 6-32	Cerberus Block Diagram . . . . .	291
Figure 6-33	Serial Bit Stream Syntax for TDI and TDO in Shift_DR State . . . . .	293
Figure 6-34	Shift Register Behavior in the Shift_DR State . . . . .	296
Figure 6-35	Example: IO_CONFIG . . . . .	296
Figure 6-36	Example: IO_SET_ADDRESS . . . . .	296

**CONFIDENTIAL**

Figure 6-37	Example: IO_WRITE_WORD .....	297
Figure 6-38	Example: IO_READ_WORD .....	297
Figure 7-1	Functional Block Diagram of RF Part. ....	322
Figure 7-2	PMB7870 Single Step, Constant Gain Switching Scheme .....	327
Figure 7-3	PMB7870 Multiple Step, Constant Gain Switching Scheme .....	327
Figure 7-4	PMB7870 Transceiver State Machine Modes .....	329
Figure 7-5	RX Sequencer Timing .....	330
Figure 7-6	TX Sequencer Timing .....	331
Figure 7-7	DCXO Sub-range Selection .....	332
Figure 7-8	Crystal Oscillator Functional Overview .....	333
Figure 7-9	Power Up and Down Sequence: Scenario 1 .....	335
Figure 7-10	Power Up and Down Sequence: Scenario 2 .....	336
Figure 8-1	Enabling of Interrupts in TEAKLite Interrupt Unit .....	339
Figure 8-2	Reset and Set INT_FINTy of TEAKLite Interrupt Unit .....	340
Figure 8-3	Simplified Functional Block Diagram of Timer 1 .....	354
Figure 8-4	Simplified Functional Block Diagram of Timer 2 .....	358
Figure 8-5	TEAKLite Interface and HW Internal Memory .....	369
Figure 8-6	Packing Modes .....	369
Figure 8-7	Clocking Scheme .....	371
Figure 8-8	RAM2 - Memory Partitioning .....	372
Figure 8-9	RAMW1/RAMW2 - Memory Partitioning .....	373
Figure 8-10	Block Diagram of Bi-Directional I2Sx 4-Pin Interface .....	377
Figure 8-11	Principal Timing of I2S Interface Signals (Normal Mode) .....	379
Figure 8-12	Timing of I2S Interface Signals in PCM Mode .....	381
Figure 8-13	Timing of I2S Interface Signals in PCM Mode with Burst Transmission .. 382	382
Figure 8-14	Timing of I2S Interface Signals in PCM Mode with Burst Reception .	382
Figure 8-15	Signal Timing and System Setup for DAI Mode .....	384
Figure 8-16	Block Diagram of Clock Source Selection and Distribution for I2S 1.	385
Figure 8-17	Functional Block Diagram of the I2Sx Transmit and Receive Buffers	400
Figure 8-18	Mapping of Data Words into the Buffer .....	401
Figure 8-19	Block Diagram of Bi-Directional I2S 2 6-Pin Interface .....	418
Figure 8-20	Block Diagram of Unidirectional I2S Interface Hardware .....	420
Figure 8-21	Principal Timing of I2S Interface Signals (Normal Mode) .....	422
Figure 8-22	Timing of I2S Interface Signals in PCM Mode .....	423
Figure 8-23	Timing of I2S Interface Signals in PCM Mode with Burst Transmission .. 424	424
Figure 8-24	Block Diagram of Clock Source Selection and Distribution for I2S3 .	425
Figure 8-25	Functional Block Diagram of the I2S3 Transmit Buffer .....	437
Figure 8-26	Mapping of Data Words into the Buffer .....	437
Figure 8-27	Synchronous Serial Interface Controller Block Diagram .....	449
Figure 8-28	Serial Clock Phase and Polarity Options .....	458
Figure 8-29	SSC Duplex Configuration .....	459

## CONFIDENTIAL

Figure 8-30	SSC Half-Duplex Configuration . . . . .	462
Figure 8-31	Transmit FIFO Operation Example . . . . .	463
Figure 8-32	Receive FIFO Operation Example . . . . .	465
Figure 8-33	Transparent Mode Receive FIFO Operation . . . . .	467
Figure 8-34	SSC Baud Rate Generator . . . . .	473
Figure 8-35	SSC Error Interrupt Control . . . . .	476
Figure 8-36	Mono Slot Application . . . . .	482
Figure 8-37	Multi Slot Application . . . . .	483
Figure 8-38	Modulator Mixed Signal Parts . . . . .	490
Figure 8-39	GSM Spectrum Mask for the Baseband Chip (RBW = 30 kHz) . . . . .	494
Figure 8-40	Spurious Emissions Mask (dBc/Hz) . . . . .	495
Figure 8-41	Interface of BB Receive Unit . . . . .	498
Figure 8-42	Simplified Functional Block Diagram of Baseband Filter . . . . .	499
Figure 8-43	CORDIC-Based Demodulator Block Diagram . . . . .	501
Figure 8-44	Block Diagram of Adaptive FIR Filter . . . . .	503
Figure 8-45	Coefficient Storage Format (Coefficient are Preliminary) . . . . .	504
Figure 8-46	Amplitude Functions of the Digital Baseband Filter . . . . .	516
Figure 8-47	Signal to Noise + Distortion Ratio for BB-RX Path . . . . .	517
Figure 8-48	Signal to Noise + Distortion Ratio for BB-RX with 100 mVpp Wanted Signal 518	
Figure 8-49	Overview of Clocking and Interfaces of Audio Front End . . . . .	524
Figure 8-50	Functional Block Diagram of the Audio Front End DSP Interface . . . . .	528
Figure 8-51	Output Selection: Block Diagram and Potential External Connections . . . . .	532
Figure 8-52	Block Diagram of Analog Part of PMB7870 Audio Front End . . . . .	533
Figure 8-53	Microphone Supply Generation Without Low-Pass Filtering . . . . .	535
Figure 8-54	Microphone Supply Generation with Low-Pass Filtering . . . . .	536
Figure 8-55	Principle of Ringing via the CAPCOM Units . . . . .	543
Figure 8-56	Block Diagram of the Ciphering Unit . . . . .	546
Figure 8-57	Data and Control Flow . . . . .	553
Figure 8-58	Timing . . . . .	556
Figure 8-59	Block Selection . . . . .	565
Figure 8-60	TEAKLite interface and Hardware Internal Memories . . . . .	578
Figure 8-61	Packing Modes . . . . .	579
Figure 8-62	Clocking Scheme . . . . .	581
Figure 8-63	RAM 1 Memory Partitioning . . . . .	582
Figure 8-64	RAM2 Memory Partitioning . . . . .	583
Figure 8-65	RAMW1/RAMW2 Memory Partitioning . . . . .	584
Figure 8-66	Initialization of Equalization . . . . .	588
Figure 8-67	Flow of Equalization . . . . .	589
Figure 8-68	Global Architecture . . . . .	592
Figure 8-69	OCEM Program Address Break Point Block Diagram . . . . .	596
Figure 8-70	OCEM Data Address/Data Value Operation Block Diagram . . . . .	597

**CONFIDENTIAL**

Figure 8-71	OCEM Enabling Block . . . . .	613
Figure 8-72	PMB7870 Debug PC Communication . . . . .	614
Figure 10-1	General Block Diagram of X-Bus Peripheral . . . . .	618
Figure 10-2	MCU Configuration during Reset via X-Bus. . . . .	619
Figure 10-3	E-GOLDRadio Duel-Port RAM Shared Memory . . . . .	622
Figure 10-4	Multicore Synchronization . . . . .	626
Figure 10-5	Block Diagram of Communication Flag (Only One Slice Shown) . . . . .	632
Figure 10-6	Block Diagram of Semaphore Flag (only one slice shown) . . . . .	638
Figure 10-7	Bus Interface Unit between X-Bus and TEAKLite . . . . .	645
Figure 10-8	Clock Generation Unit Scheme . . . . .	655
Figure 10-9	Overview of E-GOLDRadio CPU Clock Generation System. . . . .	656
Figure 10-10	Clock Enable . . . . .	662
Figure 10-11	Clock and Enable Generation for MCU Sub-System . . . . .	668
Figure 10-12	Generation of Clocks for the DSP Section. . . . .	670
Figure 10-13	Overview of Interface Control Functions . . . . .	689
Figure 10-14	SCCU Main State Machine . . . . .	690
Figure 10-15	Function of Signals from GSM Sleep Timer. . . . .	691
Figure 10-16	Timing Diagram for Transition $\mu c\_off \rightarrow tcxo\_off$ . . . . .	692
Figure 10-17	Timing Diagrams for Transitions $tcxo\_off \rightarrow tcxo\_on \rightarrow shap\_on$ . . . . .	695
Figure 10-18	Timing of Wake-Up Phase . . . . .	696
Figure 10-19	Sleep Function Timing Diagram. . . . .	699
Figure 10-20	Early Termination of GSM Timer Sleep Phase . . . . .	700
Figure 10-21	Stand-By Clock Calibration . . . . .	701
Figure 10-22	Measurement Circuit and Mode Setting. . . . .	719
Figure 10-23	Equivalent Network for Differential Measurement Circuit. . . . .	723
Figure 10-24	Equivalent Network for Reference Voltage Generation . . . . .	724
Figure 10-25	Equivalent Network for Measurement Signal Input . . . . .	725
Figure 10-26	Equivalent Network for Single-Ended Measurement Mx . . . . .	726
Figure 10-27	Equivalent Network for Measurement Signal Input . . . . .	727
Figure 10-28	PAOUT Offset Measurement. . . . .	731
Figure 10-29	Equivalent Network for Offset Calculation . . . . .	731
Figure 10-30	Logic Block Diagram of Measurement Interface . . . . .	748
Figure 10-31	Clock Control for Measurement Interface . . . . .	749
Figure 10-32	Block Symbol of the Keypad Interface . . . . .	755
Figure 10-33	Block Diagram of the Internal Keypad Port Circuitry . . . . .	758
Figure 10-34	Block Symbol of the SIM Card Interface . . . . .	762
Figure 10-35	SIM Card Interface Block Diagram . . . . .	763
Figure 10-36	Timing of Automatic Power Down Sequence. . . . .	780
Figure 10-37	SIM Character Data Structure . . . . .	781
Figure 10-38	Power Amplifier Control Hardware System Overview . . . . .	794
Figure 10-39	PA Control Hardware . . . . .	795
Figure 10-40	Linear Power Ramping during Active Part of the Burst . . . . .	796
Figure 10-41	Operating Range of PAOUT1 Output after Offset Calibration . . . . .	799

**CONFIDENTIAL**

Figure 10-42	External Interfaces of the GSM Timer Unit	804
Figure 10-43	Basic Operation of the GSM Timer Unit.	807
Figure 10-44	TDMA Counter Unit	812
Figure 10-45	Overflow Behavior of RTDMA Counter	813
Figure 10-46	GSM Clock Control Unit.	828
Figure 10-47	RF Control Unit	836
Figure 10-48	Principle Timings of 8 Bit Telegrams	848
Figure 10-49	SFRs and Port Pins Associated with the External Bus Interface	853
Figure 10-50	Demultiplexed Bus, Write Access	856
Figure 10-51	Demultiplexed Bus, Read Access	857
Figure 10-52	Optional BSWC Wait between BUSCON Windows	859
Figure 10-53	Latched and Early Chip Select.	864
Figure 10-54	Programmable External Bus Cycle	865
Figure 10-55	READY Controlled Bus Cycles	868
Figure 10-56	Address Window Arbitration Example	875
Figure 10-57	Page Mode Control Unit (PMCU) Block Diagram	878
Figure 10-58	De-assertion of CSx_n_i before Timeout.	881
Figure 10-59	Timeout Occurs before CSx_n_i Is De-asserted	882
Figure 10-60	At the Next Falling Edge of CSx_n_i, When the Timeout Has Occurred	883
Figure 10-61	Demultiplexed Read Access Timing Diagram	886
Figure 10-63	Example Read Access at 78MHz	891
Figure 10-64	Normal Connection of Flash to EGR	896
Figure 10-65	Connection of Multiple Devices Using CS2	897
Figure 10-66	Propagation Delay and Setup Time.	899
Figure 10-67	Timing Diagram	900
Figure 10-68	Random Read Access for External Flash with Normal ALE	901
Figure 10-69	Random Read Access for External Flash with Extended ALE	902
Figure 10-70	Page Mode Enable - OFF page Read Access.	903
Figure 10-71	Page Mode Enable - ON page Read Access.	904
Figure 10-73	Number of WS for INTEL Combo	906
Figure 10-74	Number of WS for INTEL Flash	906
Figure 10-75	Number of WS for AMD Flash	907
Figure 10-76	Number of WS for Infineon CRAM.	907
Figure 10-77	Wait State Computation and Performance	908
Figure 10-78	Address Areas Mapping.	909
Figure 10-79	Block Diagram of Port Signal Logic Arranger	916
Figure 10-80	GPRS_GEA3 Unit Block Diagram	922
Figure 10-81	Example GEA1/GEA2 or GEA3 Block Diagram.	923
Figure 10-82	Structure of a LLC frame (Simplified Example)	931
Figure 10-83	Splitting of the LLC Frame Into Words.	932
Figure 10-84	MSB/LSB Processing Order within the GPRS Block	932
Figure 10-85	FCS Ciphering	933

**CONFIDENTIAL**

Figure 11-1	Bus Conditions . . . . .	950
Figure 11-2	IIC Bus Line Connections . . . . .	951
Figure 11-3	SSC Interface Diagram . . . . .	979
Figure 11-4	Synchronous Serial Channel SSC Block Diagram . . . . .	980
Figure 11-5	Serial Clock Phase and Polarity Options . . . . .	982
Figure 11-6	SSC Duplex Configuration . . . . .	983
Figure 11-7	SSC Half-Duplex Configuration . . . . .	985
Figure 11-8	Transmit FIFO Operation Example . . . . .	986
Figure 11-9	Receive FIFO Operation Example . . . . .	988
Figure 11-10	Transparent Mode Receive FIFO Operation . . . . .	989
Figure 11-11	SSC Baudrate Generator . . . . .	991
Figure 11-12	SSC Error Interrupt Control . . . . .	993
Figure 11-13	ASC0 Interface Diagram . . . . .	1013
Figure 11-14	Block Diagram of the ASC0 . . . . .	1014
Figure 11-15	Asynchronous Mode of Serial Channel ASC0 . . . . .	1016
Figure 11-16	Asynchronous 8-Bit Frames . . . . .	1017
Figure 11-17	Asynchronous 9-Bit Frames . . . . .	1018
Figure 11-18	IrDA Frame Encoding/Decoding . . . . .	1019
Figure 11-19	Transmit FIFO Operation Example . . . . .	1020
Figure 11-20	Receive FIFO Operation Example . . . . .	1023
Figure 11-21	Transparent Mode Receive FIFO Operation . . . . .	1024
Figure 11-22	Fixed IrDA Pulse Generation . . . . .	1025
Figure 11-23	RXD/TXD Data Path in Asynchronous Modes . . . . .	1027
Figure 11-24	Synchronous Mode of Serial Channel ASC0 . . . . .	1028
Figure 11-25	ASC0 Synchronous Mode Waveforms . . . . .	1030
Figure 11-26	ASC0 Baudrate Generator Circuitry in Asynchronous Modes . . . . .	1032
Figure 11-27	ASC0 Baudrate Generator Circuitry in Synchronous Mode . . . . .	1036
Figure 11-28	Asynchronous Mode Block Diagram . . . . .	1037
Figure 11-29	Two-Byte Serial Frames with ASCII 'at' . . . . .	1038
Figure 11-30	Two-Byte Serial Frames with ASCII 'AT' . . . . .	1039
Figure 11-31	ASC0 Interrupt Generation . . . . .	1046
Figure 11-32	ASC1 Interface Diagram . . . . .	1069
Figure 11-33	CAPCOM 1 Interface Diagram . . . . .	1085
Figure 11-34	CAPCOM Unit 1 Block Diagram . . . . .	1086
Figure 11-35	Block Diagram of Timers . . . . .	1087
Figure 11-36	Capture Mode Block Diagram . . . . .	1091
Figure 11-37	Compare Mode 0 and 1 Block Diagram . . . . .	1092
Figure 11-38	Timing Example for Compare Modes 0 and 1 . . . . .	1093
Figure 11-39	Compare Mode 2 and 3 Block Diagram . . . . .	1094
Figure 11-40	Timing Example for Compare Modes 2 and 3 . . . . .	1095
Figure 11-41	Double-Register Compare Mode Block Diagram . . . . .	1097
Figure 11-42	Timing Example for Double-Register Compare Mode . . . . .	1098
Figure 11-43	Interface Diagram for Alternative Implementation . . . . .	1099



**CONFIDENTIAL**

Figure 11-44	Port Input Selection for Capture Channels	1110
Figure 11-45	RTC Interrupt Generation	1127
Figure 11-46	MS Clock Generation with On-Chip Oscillators	1129
Figure 11-47	Circuitry to Use the Internal 32 kHz Oscillator	1129
Figure 11-48	Circuitry to Use the External 32 kHz Oscillator	1130
Figure 11-49	RTC Kernel Block Diagram	1132
Figure 11-50	ACCPOS Bit Generation	1135
Figure 11-51	Differentiating Circuits for RTC interrupt	1141
Figure 11-52	GPT12 Interface Diagram	1146
Figure 11-53	Structure of Timer Block 1	1148
Figure 11-54	Block Diagram of Core Timer T3 in Timer Mode	1151
Figure 11-55	Block Diagram of Core Timer T3 in Gated Timer Mode	1151
Figure 11-56	Block Diagram of Core Timer T3 in Counter Mode	1152
Figure 11-57	Block Diagram of Core Timer T3 in Incremental Interface Mode	1153
Figure 11-58	Interfacing the Encoder to the Microcontroller	1154
Figure 11-59	Evaluation of the Incremental Encoder Signals	1155
Figure 11-60	Evaluation of the Incremental Encoder Signals	1156
Figure 11-61	Block Diagram of an Auxiliary Timer in Counter Mode	1158
Figure 11-62	Concatenation of Core Timer T3 and an Auxiliary Timer	1159
Figure 11-63	GPT1 Auxiliary Timer in Reload Mode	1160
Figure 11-64	GPT1 Timer Reload Configuration for PWM Generation	1161
Figure 11-65	Auxiliary Timer of Timer Block 1 in Capture Mode	1162
Figure 11-66	Structure of Timer Block 2	1164
Figure 11-67	Block Diagram of Core Timer T6 in Timer Mode	1166
Figure 11-68	Block Diagram of Core Timer T6 in Gated Timer Mode	1167
Figure 11-69	Block Diagram of Core Timer T6 in Counter Mode	1168
Figure 11-70	Block Diagram of Auxiliary Timer T5 in Counter Mode	1170
Figure 11-71	Concatenation of Core Timer T6 and Auxiliary Timer T5	1171
Figure 11-72	Timer Block 2 Register CAPREL in Capture Mode	1172
Figure 11-73	Timer Block 2 Register CAPREL in Reload Mode	1173
Figure 11-74	Timer Block 2 Register CAPREL in Capture-And-Reload Mode	1174
Figure 11-75	Architecture of Pad Control and Port Logic	1193
Figure 11-76	Example for Connecting PCL Blocks and Chip Internal Blocks	1194
Figure 11-77	Flash Write Protection Functional Diagram	1203
Figure 11-78	Monitoring of Internal Signals at the Monitor Pads	1209
Figure 11-79	Break Switch Interface	1244
Figure 11-80	Break Bus Switch	1245
Figure 11-81	JTAG Module Block Diagram	1249
Figure 11-82	TAP Controller State Diagram	1250
Figure 11-83	IOPATH Register	1256
Figure 11-84	JTAG IO Mode Application Example	1259
Figure 11-85	IO Mode Basic Architecture	1260
Figure 11-86	C166 CERBERUS IO-Client Selection	1261

**CONFIDENTIAL**

Figure 13-1	External Filter at PLL Supply . . . . .	1308
Figure 13-2	Input Output Waveform for AC Test . . . . .	1314
Figure 13-3	Float Waveforms . . . . .	1314
Figure 13-4	Overview of EBU Parameter Characterizations . . . . .	1325
Figure 13-5	Timing Diagram of Parameters . . . . .	1326
Figure 13-6	Timing Diagram for Parameters . . . . .	1327
Figure 13-7	Timing Diagram for Parameters . . . . .	1328
Figure 13-8	Timing Diagram for Parameters . . . . .	1329
Figure 13-9	Overview of PMCU Parameter Characterizations . . . . .	1331
Figure 13-10	First OFF-PAGE Access after Flash CS\ Validation 1 (PMCx.OFFPWS > 00 <sub>H</sub> ) 1332	
Figure 13-11	First OFF-PAGE Access after Flash CS\ Validation 2 (PMCx.OFFPWS = 00 <sub>H</sub> ) 1333	
Figure 13-12	OFF-PAGE Access Number N after Flash CS\ Validation (with N > 1) . . . 1334	
Figure 13-13	Switch of CS\ from Valid to Invalid . . . . .	1335
Figure 13-14	First OFF-PAGE Access after Flash CS\ Validation 1 (PMC0/1.OFFPWS > 00 <sub>H</sub> ) 1337	
Figure 13-15	First OFF-PAGE Access after Flash CS\ Validation 2 (PMCx.OFFPWS = 00 <sub>H</sub> ) 1338	
Figure 13-16	First OFF-PAGE Access after the Flash CS\ Validation 3 (PMCx.OFFPWS > 00) 1339	
Figure 13-17	First OFF-PAGE Access after the Flash CS\ Validation 4 (PMCx.OFFPWS = 00) 1340	
Figure 13-18	First OFF-PAGE Access after the Flash CS\ Validation 5 (PMCx.OFFPWS > 00 <sub>H</sub> ) 1341	
Figure 13-19	First OFF-PAGE Access after the Flash CS\ Validation 6 (PMCx.OFFPWS = 00 <sub>H</sub> ) 1342	
Figure 13-20	SIM Timing . . . . .	1344
Figure 13-21	Definition of Timing for F/S-Mode Devices on I <sup>2</sup> C-Bus . . . . .	1345
Figure 13-22	Test circuitry of I2C-bus . . . . .	1346
Figure 13-23	Timing Diagram of I <sup>2</sup> S Signals in Normal Mode - Master Mode . . .	1347
Figure 13-24	Timing Diagram of I <sup>2</sup> S Signals in Normal Mode - Slave Mode . . .	1348
Figure 13-25	Timing Diagram of I <sup>2</sup> S Signals in Burst Mode - Master Mode . . . .	1350
Figure 13-26	Timing Diagram of I <sup>2</sup> S Interface Signals in Burst Mode - Slave Mode . . . 1352	
Figure 13-27	Timing Diagram of I <sup>2</sup> S Interface Signals in DAI Mode . . . . .	1354
Figure 13-28	Timing of ASC in Synchronous Mode . . . . .	1355
Figure 13-29	SSC Interface Timing in Master Mode . . . . .	1356
Figure 13-30	SSC Interface Timing in Slave Mode . . . . .	1358
Figure 13-31	Output Clocks: CLKOUT Timing . . . . .	1362
Figure 13-32	Timing Diagram for Analog Ringer Mode . . . . .	1366
Figure 13-33	Definition of the DCXO Tuning Nonlinearity . . . . .	1396

**CONFIDENTIAL**

Figure 13-34	TX Power and Spectrum Measurement Setup . . . . .	1397
Figure 13-35	RX Measurement Setup. . . . .	1397
Figure 13-36	Ripple Mask for 1.5 V Supply. . . . .	1398
Figure 14-1	Reset Timing Diagram . . . . .	1405
Figure 14-2	Reset Interaction Timing Diagram Example . . . . .	1406
Figure 14-3	Power-Off. . . . .	1408
Figure 14-4	Power-Up Sequence . . . . .	1409
Figure 14-5	Boot Configuration Latch Register . . . . .	1411
Figure 15-1	Break Switch Interface . . . . .	1413
Figure 15-2	JTAG IO Mode Application Example . . . . .	1414
Figure 17-1	Application Circuit . . . . .	1422
Figure 17-2	System Block Diagram. . . . .	1423
Figure 17-3	External 26 MHz Module and FSYS Outputs. . . . .	1424
Figure 18-1	E-GOLDradio: LF2BGA-233 Package Details and Ball Labelling . .	1426

**CONFIDENTIAL**

CONFIDENTIAL

**CONFIDENTIAL**

Table 1-1	E-GOLDRadio Version Naming . . . . .	41
Table 3-1	Pin List . . . . .	51
Table 5-1	Partitioning of Data and Address Space on TEAKLite and Data Page Mode 72	
Table 5-2	Boot Address Selection . . . . .	76
Table 6-1	Branch Target Addressing Modes . . . . .	100
Table 6-2	Sequential Instruction Execution (Local Memory, 0/1 Wait States) . .	101
Table 6-3	Unconditional Branches (LM-Bus, 0/1 Wait States). . . . .	102
Table 6-4	Conditional branches (LM-Bus, 0/1 Wait States). . . . .	103
Table 6-5	Unconditional cached branches (LM-Bus, 0/1 Wait States). . . . .	104
Table 6-6	Conditional cached branches (LM-Bus, 0/1 waitstate). . . . .	105
Table 6-7	Hardware Traps . . . . .	122
Table 6-8	DPRAM Addresses of PEC Source and Destination Pointer . . . . .	129
Table 6-9	PEC Channels That Can Be Linked Together . . . . .	136
Table 6-10	PEC Interrupt Level Control with PLEV Bits in PECCx Registers . .	137
Table 6-11	Software Controlled Interrupt Classes (Example) . . . . .	141
Table 6-12	E-GOLDRadio Interrupt List . . . . .	142
Table 6-13	Interrupt Types . . . . .	147
Table 6-14	Addressing Modes to Access Word-GPRs . . . . .	150
Table 6-15	Addressing Modes to Access Byte-GPRs . . . . .	151
Table 6-16	Short addressing modes . . . . .	153
Table 6-17	Long Addressing Mode . . . . .	159
Table 6-18	Indirect Addressing Modes . . . . .	160
Table 6-19	Circular Stack Address Transformation . . . . .	165
Table 6-20	MCU Data Formats . . . . .	168
Table 6-21	ANSI C Data Types . . . . .	168
Table 6-22	Constant Formats . . . . .	169
Table 6-23	Shift Right Rounding Error Evaluation . . . . .	178
Table 6-24	Minimum Execution Times . . . . .	184
Table 6-25	Addressing Modes to Access Word-GPRs . . . . .	187
Table 6-26	Addressing Modes to Access Byte-GPRs . . . . .	188
Table 6-27	Core Special Function Registers Ordered by Address . . . . .	189
Table 6-28	Register Overview Interrupt and Peripheral Event Controller . . . . .	190
Table 6-29	Mapping of General-Purpose Registers to DPRAM Addresses. . . . .	197
Table 6-30	System Stack Size . . . . .	198
Table 6-31	Address Range and Address Range Start Definition of XADRx Registers 207	
Table 6-32	Instructions Referenced by Opcode (Part 1) . . . . .	210
Table 6-33	Instructions Referenced by Opcode (Part 2) . . . . .	211
Table 6-34	Instruction Set . . . . .	214
Table 6-35	Instruction Set Summary . . . . .	215
Table 6-36	Mnemonics, Operands (Part 1) . . . . .	225
Table 6-37	Mnemonics, Operands (Part 2) . . . . .	226

**CONFIDENTIAL**

Table 6-38	Mnemonics, Operands (Part 3) . . . . .	228
Table 6-39	Mnemonics, Operands (Part 4) . . . . .	229
Table 6-40	Reset Types. . . . .	232
Table 6-41	Values of Reset Signals in Reset-All-Phase . . . . .	233
Table 6-42	Values of Reset Signals in Reset-Sequence-Phase . . . . .	233
Table 6-43	Values of Reset Signals in Init-Phase . . . . .	234
Table 6-44	System Startup Configuration Latching . . . . .	240
Table 6-45	Code Definitions of Startup Configurations (conf_rst_cfg). . . . .	241
Table 6-46	WDT Prescaler Values. . . . .	244
Table 6-47	<b>WDTCN</b> Reset Sources . . . . .	245
Table 6-48	Reset Values of Bits 0 to 7, P = Previous Value . . . . .	245
Table 6-49	State Machine Commands. . . . .	250
Table 6-50	Alternate Sources for External Interrupts. . . . .	253
Table 6-51	Clock Output Frequencies Control Register . . . . .	257
Table 6-52	OCDS Register Overview . . . . .	273
Table 6-53	Hardware Triggers . . . . .	274
Table 6-54	Debug Event Priority . . . . .	276
Table 6-55	Debug Event Actions . . . . .	277
Table 6-56	OCDS Register Summary . . . . .	280
Table 6-57	DTREVT.COM_R Field . . . . .	283
Table 6-58	SELECT_E Field . . . . .	283
Table 6-59	Cerberus I/O Instructions. . . . .	294
Table 6-60	Cerberus Register Summary . . . . .	298
Table 6-61	<b>IOSR</b> .CRSYNC Bit. . . . .	306
Table 6-62	<b>IOSR</b> .CWSYNC Bit . . . . .	306
Table 6-63	dirty_bit Read Tag . . . . .	307
Table 6-64	Register Reset Behavior . . . . .	312
Table 7-1	Pin Definition and Function . . . . .	323
Table 8-1	Interrupt Unit Registers . . . . .	338
Table 8-2	Interface Description . . . . .	362
Table 8-3	Register Description (Configuration and Status Registers) . . . . .	362
Table 8-4	I2Sx Signals. . . . .	377
Table 8-5	Selection of I2Sx_CLK0 for I2Sx . . . . .	386
Table 8-6	Selection of I2Sx_CLK1 for I2Sx . . . . .	386
Table 8-7	Selection of Reference Clock. . . . .	388
Table 8-8	Bit Clock Rates with 89.1 MHz Reference and Integer Ratio of Fractional Divider 389	
Table 8-9	Bit Clock Rates with 89.1 MHz Reference and Non-Integer Ratio of Fractional Divider 391	
Table 8-10	Bit Clock Rates with 104 MHz Reference and Integer Ratio of Fractional Divider 394	
Table 8-11	Bit Clock Rates with 104 MHz Reference and Non-Integer Ratio of Fractional Divider 396	

**CONFIDENTIAL**

Table 8-12	Selection of clk_TX and clk_RX and WA signals . . . . .	398
Table 8-13	Jitter of CLKx, WAx and TX Outputs in Master Mode . . . . .	399
Table 8-14	Delay Of Output Signals In Master Mode . . . . .	399
Table 8-15	I2Sx Register List. . . . .	403
Table 8-16	I2Sx Signals. . . . .	418
Table 8-17	I <sup>2</sup> S 3 Signals . . . . .	421
Table 8-18	Selection of I2S3_CLK for I2S3 . . . . .	425
Table 8-19	Selection of Reference Clock. . . . .	426
Table 8-20	Bit Clock Rates with 89.1 MHz Reference and Integer Ratio of Fractional Divider (Minimum Jitter) 427	
Table 8-21	Bit Clock Rates with 89.1 MHz Reference and Non-Integer Ratio of Fractional Divider (Minimum Bit Clock Deviation) 429	
Table 8-22	Bit Clock Rates with 104 MHz Reference and Integer Ratio of Fractional Divider (Minimum Jitter) 431	
Table 8-23	Bit Clock Rates with 104 MHz Reference and Non-Integer Ratio of Fractional Divider (Minimum Bit Clock Deviation) 433	
Table 8-24	Selection of clk_TX . . . . .	435
Table 8-25	Jitter of CLK3, WAx and TX Outputs in Master Mode . . . . .	436
Table 8-26	Delay Of Output Signals In Master Mode . . . . .	436
Table 8-27	I2S3 Register List. . . . .	439
Table 8-28	SSC Register Overview . . . . .	448
Table 8-29	Modulator Register List . . . . .	480
Table 8-30	GSM Requirements: TX Spectrum Due to Modulation . . . . .	492
Table 8-31	GSM Requirements: Spurious Emissions . . . . .	492
Table 8-32	Correction Values for Bandwidth . . . . .	493
Table 8-33	Definition of the GSM and Baseband Templates (BB). . . . .	493
Table 8-34	Definition of the GSM and BASEBAND Templates (BB) . . . . .	493
Table 8-35	Modulator Register List . . . . .	499
Table 8-36	External Voice Signals . . . . .	523
Table 8-37	Requirements for Digital Interpolation Filters. . . . .	529
Table 8-38	Overall Analog Gain for Output Buffers . . . . .	531
Table 8-39	Settings of <b>AFE_VRXCTRL1</b> Control Bits in for Earpiece Amplifiers 531	
Table 8-40	Overview of Audio Front-End Registers. . . . .	536
Table 8-41	Control Bits for Ringing via Line-Out Buffers (X = 'don't care', that is, RINGSELSP = 1 542	
Table 8-42	Control bits for ringing via internal power buffers (X = 'don't care), that is, RINGSELPA = 1 542	
Table 8-43	Cipher Ram Output Encryption/Decryption Sequences . . . . .	552
Table 8-44	KGCORE Parameters and A53 Registers . . . . .	557
Table 8-45	Cipher RAM Format . . . . .	564
Table 8-46	Registers . . . . .	568
Table 8-47	Register-Description (Configuration and Status Registers) . . . . .	568
Table 8-48	Clock, Reset. . . . .	599

**CONFIDENTIAL**

Table 8-49	Core-External Interface . . . . .	599
Table 8-50	Core Bus Interface . . . . .	599
Table 8-51	OCEM Registers . . . . .	601
Table 8-52	Clock, Reset. . . . .	614
Table 8-53	I/O Interface . . . . .	614
Table 10-1	Signals to DPEC Block of MCU . . . . .	646
Table 10-2	Target Operating Frequencies for E-GOLDRadio . . . . .	658
Table 10-3	MCU Clock Master. . . . .	663
Table 10-4	MCU Sub-System Clock . . . . .	664
Table 10-5	MCU Sub-System Bus Clocks . . . . .	665
Table 10-6	MCU Sub-System Peripheral Clocks (ASC0, ASC1, SCC, PCL, IIC) . . . . .	666
Table 10-7	MCU Sub-System Peripheral Clocks (CAPCOM, GPT). . . . .	667
Table 10-8	DSP Clock Sources . . . . .	669
Table 10-9	Analog Clock Generation . . . . .	671
Table 10-10	The Analog Clocks . . . . .	671
Table 10-11	AFC Clock Generation . . . . .	672
Table 10-12	Output Clock . . . . .	672
Table 10-13	Phase Shifter Frequencies. . . . .	679
Table 10-14	Phase Shifter Frequencies. . . . .	680
Table 10-15	Setup of register <b>SCCUSPCR</b> . . . . .	687
Table 10-16	SCCU Signal Descriptions . . . . .	697
Table 10-17	SCCU Register List 1 . . . . .	704
Table 10-18	Measurement Interface Register List . . . . .	717
Table 10-19	Measurement Switch Settings . . . . .	720
Table 10-20	General Pre-Amplifier Settings . . . . .	721
Table 10-21	Resistors for Measurement Circuits. . . . .	722
Table 10-22	Measurement BPI Interrupt Requests (Service Requests) . . . . .	743
Table 10-23	<b>MEAS_CTRL1</b> = 0009 . . . . .	745
Table 10-24	<b>MEAS_CTRL2</b> = 382E . . . . .	745
Table 10-25	<b>MEAS_CTRL1</b> = 080B . . . . .	746
Table 10-26	<b>MEAS_CTRL2</b> = 3008 . . . . .	746
Table 10-27	<b>MEAS_CTRL1</b> = 3001 . . . . .	747
Table 10-28	<b>MEAS_CTRL2</b> = A000 . . . . .	747
Table 10-29	K and L Values for Various Bus Clock Frequencies . . . . .	749
Table 10-30	GSM Timer Unit Signals . . . . .	805
Table 10-31	Structure of Timing Events. . . . .	809
Table 10-32	Structure of Timing Advance . . . . .	810
Table 10-33	K and L Values for Various Kernel Clock Frequency Errors . . . . .	833
Table 10-34	RF RAM Partitioning of the RF Control Unit (Type 1) . . . . .	837
Table 10-35	RF RAM Partitioning of the RF Control Unit (Type 2) . . . . .	839
Table 10-36	Structure of Telegrams . . . . .	840
Table 10-37	Summary of External Bus Modes . . . . .	854
Table 10-38	Use of WRL#, WRH#, A0, BHE for WRITE . . . . .	861



**CONFIDENTIAL**

Table 10-39	Use of WRL#, WRH#, A0, BHE for READ .....	861
Table 10-40	Bus Mode Versus Performance .....	862
Table 10-41	Chip-Select Generation Modes .....	863
Table 10-42	Address Window Definition .....	874
Table 10-43	Result of Dhrystone 2.1 for E-GOLDradio V1.0 with program running in internal RAM with 0 WS 911	
Table 10-44	Result of Dhrystone 2.1 for E-GOLDradio V1.0 with program running in internal RAM with 1 WS 911	
Table 10-45	Result of Dhrystone 2.1 for E-GOLDradio V1.0 with program running in external RAM or in external flash without the Page Mode with 0 WS in internal RAM 911	
Table 10-46	Result of Dhrystone 2.1 for E-GOLDradio V1.0 with program running in external RAM or in external flash without the Page Mode with 1 WS in internal RAM 912	
Table 10-47	Result of Dhrystone 2.1 for E-GOLDradio V1.0 with program running in external flash with the Page Mode enable with 0 WS in internal RAM 912	
Table 10-48	Result of Dhrystone 2.1 for E-GOLDradio V1.0 with program running in external flash with the Page Mode enable with 1 WS in internal RAM 913	
Table 10-49	LPSA RAM Words 0 and 1 .....	917
Table 10-50	LPSA RAM Example .....	917
Table 10-51	Uplink Example .....	925
Table 10-52	Downlink Example .....	929
Table 10-53	Clock, Reset. ....	933
Table 10-54	Data Interface .....	933
Table 11-1	I2C Register List .....	954
Table 11-2	IIC-Bus Baud Rate Selection for BRPMOD = 0 .....	971
Table 11-3	IIC-Bus Baud Rate Selection for BRPMOD = 1 .....	972
Table 11-4	Interrupt Sources .....	973
Table 11-5	Typical Baudrates of the SSC (fhw_clk = 52 MHz) .....	991
Table 11-6	Typical Baudrates of the SSC (fhw_clk = 78 MHz) .....	992
Table 11-7	SSC Register Summary .....	995
Table 11-8	SSC Interrupt Sources .....	1008
Table 11-9	Formulas for IrDA Pulse Width Calculation .....	1026
Table 11-10	IrDA Pulse Width Adaption to 1.627 ms. ....	1026
Table 11-11	Asynchronous Baudrate Formulas using Fixed Input Clock Dividers	1033
Table 11-12	Typical Asynchronous Baudrates Using Fixed Input Clock Dividers	1033
Table 11-13	Asynchronous Baudrate Formulas using the Fractional Input Clock Divider	1035
Table 11-14	Typical Asynchronous Baudrates using the Fractional Input Clock Divider	1035
Table 11-15	Synchronous Baudrate Formulas .....	1036
Table 11-16	Autobaud Detection using Standard Baudrates (fDIV = 11.0592 MHz) ..	1040

**CONFIDENTIAL**

Table 11-17	Standard Baudrates - Deviations and Errors for Autobaud Detection . . . .	1041
Table 11-18	Autobaud Detection using Non-Standard Baudrates (fDIV = 9.6 MHz) . . .	1042
Table 11-19	Autobaud Detection Overwrite Values for the CON Register . . . . .	1043
Table 11-20	ASC0 Register Summary . . . . .	1047
Table 11-21	ASC1 Registers Summary . . . . .	1070
Table 11-22	Timing Examples . . . . .	1088
Table 11-23	Register Pairs for Double-Register Compare Mode . . . . .	1096
Table 11-24	CAPCOM Register Summary . . . . .	1100
Table 11-25	Selection of Capture Modes and Compare Modes . . . . .	1105
Table 11-26	CAPCOM 1 Input Signal Selection . . . . .	1111
Table 11-27	CAPCOM 2 Input Signal Selection . . . . .	1112
Table 11-28	CAPCOM Interrupt Sources . . . . .	1118
Table 11-29	RTC Register List. . . . .	1123
Table 11-30	Register Assignment to Clock and Reset Domains . . . . .	1142
Table 11-31	Impact on Counting Accuracy . . . . .	1144
Table 11-32	Core Timer T3 Count Direction Control . . . . .	1149
Table 11-33	Timer 3 Input Parameter Selection: Timer Mode and Gated Timer Mode .	1150
Table 11-34	Example for Timer 3 Frequencies and Resolutions . . . . .	1150
Table 11-35	Core Timer T3 (Counter Mode) Input Edge Selection . . . . .	1152
Table 11-36	Core Timer T3 (Incremental Interface Mode) Input Edge Selection. . . . .	1154
Table 11-37	Core Timer T3 (Incremental Interface Mode) Count Direction . . . . .	1155
Table 11-38	Timer 2,4 Input Parameter Selection: Timer Mode and Gated Timer Mode	1157
Table 11-39	Auxiliary Timer (Counter Mode) Input Edge Selection . . . . .	1158
Table 11-40	Timer x Input Parameter Selection for Incremental Interface Mode. . . . .	1163
Table 11-41	Core Timer T6 Count Direction Control (Tx = T5 or T6) . . . . .	1165
Table 11-42	Timer 6 Input Parameter Selection: Timer Mode and Gated Timer Mode .	1166
Table 11-43	Core Timer T6 (Counter Mode) Input Edge Selection . . . . .	1168
Table 11-44	Timer 5 Input Parameter Selection: Timer Mode and Gated Timer Mode .	1169
Table 11-45	Auxiliary Timer (Counter Mode) Input Edge Selection . . . . .	1170
Table 11-46	GPT12 Register Summary . . . . .	1176
Table 11-47	Timer 3 Input Parameter Selection for Timer Mode and Gated Timer Mode	1180
Table 11-48	Timer 3 Input Parameter Selection for Counter Mode . . . . .	1180
Table 11-49	Timer 3 Input Parameter Selection for Incremental Interface Mode . . . . .	1180
Table 11-50	Timer 2,4 Input Parameter Selection for Timer Mode and Gated Timer	Mode 1183
Table 11-51	Timer 2,4 Input Parameter Selection for Counter Mode . . . . .	1183

**CONFIDENTIAL**

Table 11-52	Timer 2,4 Input Parameter Selection for Incremental Interface Mode . . . .	1183
Table 11-53	Timer 6 Input Parameter Selection for Timer Mode and Gated Timer Mode	1185
Table 11-54	Timer 6 Input Parameter Selection for Counter Mode . . . . .	1186
Table 11-55	Timer 5 Input Parameter Selection for Timer Mode and Gated Timer Mode	1188
Table 11-56	Timer 5 Input Parameter Selection for Counter Mode . . . . .	1188
Table 11-57	GPT1_2 Interrupt Sources . . . . .	1189
Table 11-58	GPIO Pad Reset Values . . . . .	1197
Table 11-59	Input Programming . . . . .	1198
Table 11-60	Output Programming . . . . .	1198
Table 11-61	Pad Function Depending on Programming of PCL_<pad> . . . . .	1199
Table 11-62	Wafer Character Coding (on 6 bits) . . . . .	1208
Table 11-63	Block List . . . . .	1211
Table 11-64	Signals from the DSP (Mainly Interrupts) . . . . .	1212
Table 11-65	Signals from DSP . . . . .	1214
Table 11-66	Signals from A53 Peripherals . . . . .	1219
Table 11-67	Signals from the CGU . . . . .	1224
Table 11-68	Pad Signals . . . . .	1227
Table 11-69	INTR_EXT Signals . . . . .	1231
Table 11-70	PCL_PER Signals . . . . .	1234
Table 11-71	GPRS_GEA3 Signals . . . . .	1234
Table 11-72	MCU_PER Signals . . . . .	1239
Table 11-73	JTAG Instructions . . . . .	1251
Table 11-74	JTAG Module Port Pins . . . . .	1253
Table 11-75	JTAG Module Register Overview . . . . .	1254
Table 11-76	IOPATH Register . . . . .	1256
Table 12-1	DSP Register Addresses . . . . .	1266
Table 12-2	Bit and Non-Bit Addressable SFR Areas . . . . .	1273
Table 12-3	Bit and Non-Bit Addressable ESFR Areas . . . . .	1279
Table 12-4	Address Mapping of X-Bus Peripherals . . . . .	1285
Table 12-5	Register Overview for XADRS1 . . . . .	1287
Table 12-6	Register Overview for XADRS2 . . . . .	1291
Table 12-7	Register Overview for XADRS4 . . . . .	1292
Table 12-8	CHANNEL1 Register . . . . .	1295
Table 12-9	Dummy Table . . . . .	1299
Table 12-10	Maximum ESD . . . . .	1300
Table 12-11	Maximum Temperature . . . . .	1300
Table 12-12	Maximum Power Supply Voltages . . . . .	1301
Table 12-13	Absolute Maximum Ratings, TAMB = -30°C .. + 85°C . . . . .	1302
Table 12-14	Operating Temperature . . . . .	1303
Table 12-15	Normal Power Supply Voltages . . . . .	1303

**CONFIDENTIAL**

Table 12-16	Digital Power Supply Currents . . . . .	1304
Table 12-17	Analog Power Supply Currents (IDD measured with VDD = 2.75 V) . . . . .	1305
Table 12-18	Input Capacitances and Resistors . . . . .	1307
Table 12-19	External Load Capacitances and Resistors . . . . .	1307
Table 12-20	Pad Output Current . . . . .	1309
Table 12-21	Pull-Up/Pull-Down Currents . . . . .	1311
Table 12-22	Pad Resistances (1,75 V to 2,75 V) . . . . .	1312
Table 12-23	EBU and PMCU Measured Parameters . . . . .	1315
Table 12-24	EBU and PMCU Derived Parameters . . . . .	1322
Table 12-25	SIM Timing . . . . .	1343
Table 12-26	Timing Characteristic of I <sup>2</sup> C-Bus Interface . . . . .	1345
Table 12-27	Timing Characteristic of I <sup>2</sup> S in Normal Mode - Master Mode . . . . .	1348
Table 12-28	Timing Characteristic of I <sup>2</sup> S in normal mode - slave mode . . . . .	1349
Table 12-29	Timing Characteristics of I2S in Burst Mode - Master Mode . . . . .	1351
Table 12-30	Timing Characteristics of I2S in Burst Mode - Slave Mode . . . . .	1353
Table 12-31	Timing Characteristics of I2S in DAI Mode . . . . .	1354
Table 12-32	Timing Characteristic of ASC with D Drivers in Synchronous Mode . . . . .	1355
Table 12-33	Timing Characteristic of SSC in Master Mode . . . . .	1357
Table 12-34	Timing Characteristic of SSC in Slave Mode . . . . .	1359
Table 12-35	Standby Clock Timings . . . . .	1360
Table 12-36	Pad Tristate Control Timing . . . . .	1361
Table 12-37	Output Clock CLKOUT . . . . .	1362
Table 12-38	General Electrical Characteristics of Audio Receive Path . . . . .	1362
Table 12-39	Characteristics of Audio Rx Path for Power Buffer EPp1 and EPp2 . . . . .	1364
Table 12-40	Characteristics of Audio Rx Path for Differential Line-Out Drivers EPp1 and EPn1 . . . . .	1365
Table 12-41	Electrical characteristics of Ringer Support . . . . .	1365
Table 12-42	Timing Parameters of Ringer Output . . . . .	1366
Table 12-43	Electrical Characteristics of Audio Transmit Path . . . . .	1367
Table 12-44	Electrical Characteristics of Microphone Supply . . . . .	1368
Table 12-45	Baseband Transmit Path (Signal Outputs I/IX, Q/QX) for TXON = 1 . . . . .	1369
Table 12-46	Baseband Transmit Path (Signal Outputs I/IX, Q/QX) for TXON=0. . . . .	1370
Table 12-47	Baseband Receiver Path (Signal Inputs IR/IRX, QR/QRX) . . . . .	1371
Table 12-48	ADC Characteristics . . . . .	1373
Table 12-49	Specification of pins M0 to M2 and M7 to M10 . . . . .	1374
Table 12-50	Specification of BB_I, BB_Ix and BB_Q, BB_Qx . . . . .	1376
Table 12-51	Specification of PAOUTOF1 . . . . .	1377
Table 12-52	On Chip Temperature Measurement TIC . . . . .	1377
Table 12-53	Reference Voltage Generation (Band-Gap) . . . . .	1379
Table 12-54	Specification of Pin PAOUT1 (TXON = 1) . . . . .	1379
Table 12-55	Specification of Pins PAOUT1 (TXON = 0) . . . . .	1381
Table 12-56	Operating Range, T <sub>AMB</sub> = -30°C .. + 85°C . . . . .	1383

**CONFIDENTIAL**

Table 12-57	AC/DC Characteristics . . . . .	1385
Table 13-1	Block Reset Actions . . . . .	1404
Table 13-2	Power-Off Output Tristate Control . . . . .	1407
Table 15-1	Addresses of the Routines (in hex) . . . . .	1418
Table A-1	Glossary of Terms . . . . .	1429

CONFIDENTIAL

**CONFIDENTIAL**

CONFIDENTIAL

**CONFIDENTIAL**

# 1 Introduction

<b>History</b>	
Design Specification	Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.02	
<b>Page 41</b>	Updated <b>Table 1-1 E-GOLDradio Version Naming</b> WS00007493
Changes for Rev. 1.04	
<b>Page 41</b> <b>Page 41</b> <b>Page 41</b> <b>Page 41</b> <b>Page 41</b>	Update of <b>Table 1-1 E-GOLDradio Version Naming</b> : <ul style="list-style-type: none"> <li>• <b>PMB 7870 V1.0C</b> and</li> <li>• <b>PMB 7870 V1.1A</b> WS00007926</li> <li>• <b>PMB 7870 V2.0D</b> WS00007928</li> <li>• <b>PMB 7870 V2.0E</b> WS00007929</li> <li>• <b>PMB 7870 V1.0</b> (Development Code M2899-A001-12) WS00007931</li> </ul>
<b>Page 41</b> <b>Page 41</b>	Update of <b>Table 1-1 E-GOLDradio Version Naming</b> : <ul style="list-style-type: none"> <li>• <b>PMB 7870 V2.0D</b>,</li> <li>• <b>PMB 7870 V2.0E</b></li> </ul> WS00008243
<b>Page 41</b> <b>Page 41</b> <b>Page 41</b> <b>Page 41</b> <b>Page 41</b> <b>Page 42</b>	Update of <b>Table 1-1 E-GOLDradio Version Naming</b> : <ul style="list-style-type: none"> <li>• <b>PMB 7870 V2.1F</b>,</li> <li>• <b>PMB 7870 V2.2F</b>,</li> <li>• <b>PMB 7870 V2.1G</b>,</li> <li>• <b>PMB 7870 V2.2G</b>,</li> <li>• <b>PMB 7870 V2.1H</b>, and</li> <li>• <b>PMB 7870 V2.2H</b> WS00008535</li> </ul>
Changes for Rev. 1.05	
<b>Page 42</b> <b>Page 42</b> <b>Page 42</b> <b>Page 42</b> <b>Page 42</b>	Update of <b>Table 1-1 E-GOLDradio Version Naming</b> : <ul style="list-style-type: none"> <li>• <b>PMB 7870 V2.2I</b>,</li> <li>• <b>PMB 7870 V2.2J</b>, and</li> <li>• <b>PMB 7870 V2.2K</b> WS00009000</li> <li>• <b>PMB 7870 V2.2L</b> and</li> <li>• <b>PMB 7870 V2.2M</b> WS00009024</li> </ul>
Changes for Rev. 1.06	
<b>Page 40</b>	Updated <b>Figure 1-2 Package Label (Character height = 1.0 mm)</b> WS00009111
<b>Page 42</b>	Under Mask Change: LB -> RB in <b>PMB 7870 V2.2L</b> and <b>PMB 7870 V2.2M</b>
<b>Page 42</b>	Removed "Min power" from <b>PMB 7870 V2.1F</b> & <b>PMB 7870 V2.2F</b> and added to <b>PMB 7870 V2.1O</b> WS00008535 & WS00009052

**CONFIDENTIAL**

CONFIDENTIAL



## 1.1 Overview of E-GOLDRadio

Refer to the *E-GOLDRadio Product Specification* for a quick overview of the E-GOLDRadio.

The E-GOLDRadio is a GSM/GPRS baseband modem including RFquad-band transceiver for GSM850/GSM900/GSM1800/GSM1900 functionality designed for voice and data applications.

The E-GOLDRadio is designed as a single chip solution that integrates the digital, mixed signal portions of the baseband and RF portions. It uses a leading 0.13  $\mu\text{m}$  technology with a core voltage of 1.5 V to produce a high performance mobile station with a large set of features at very low cost.

The transceiver consists of a constant gain direct conversion receiver with an analog I/Q baseband interface, a fully integrated SD-synthesizer with HSCSD and GPRS capability, a fully integrated quad-band RF oscillator, a quad-band digital GMSK modulator with an analog TX I/Q interface, and a digitally controlled crystal oscillator with three outputs.

The E-GOLDRadio has a flexible set of interfaces that allows a wide choice of communication interfaces and supports a high multimedia data rate.

The E-GOLDRadio supports a wide variety of Power architectures. In combination with either the:

- Infineon Technologies SM-POWERlite (PMB 6811)
- Infineon Technologies E-POWER Lite (PMB 6814).

only two chips are needed for a complete GSM/GPRS system. This results in an extremely compact integration with high performance, low power consumption, and low cost.

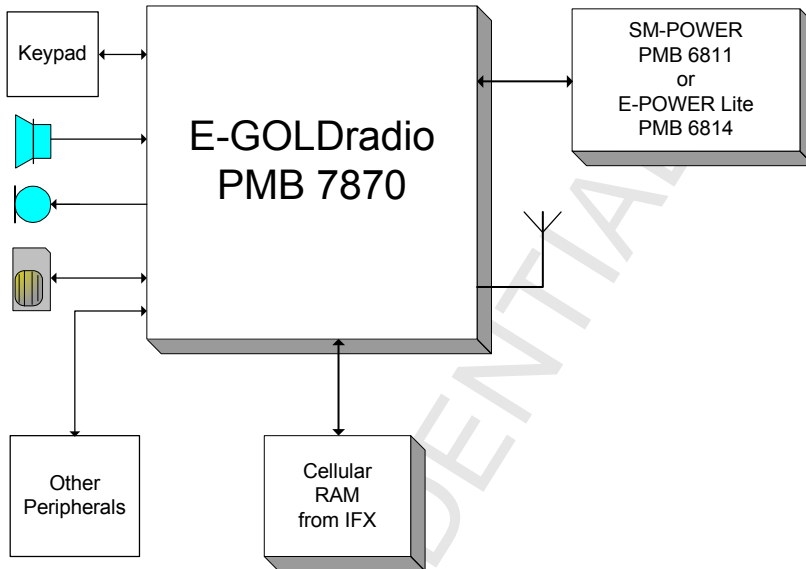
The E-GOLDRadio is powered by the C166<sup>®</sup>S CPU and TEAKLite<sup>®</sup> DSP cores.

**Figure 4-2 Diagram of E-GOLDRadio Buses (on Page 67)** is an overview of the E-GOLDRadio functional blocks.

## 1.2 GSM/GPRS System Description

**Figure 1-1** shows a typical application circuit for the E-GOLDradio.

**Figure 1-1 Typical E-GOLDradio Application Circuit**



The E-GOLDradio is suited for mobile stations operating in the GSM850/900/1800/1900 bands.

In the receiver path the antenna input signal is converted to the base band, filtered, and amplified to target level by the RF transceiver chipset. The resulting differential I and Q baseband signals are fed into the E-GOLDradio. The A-to-D converter generates two 6.5 Mbit/s data streams. The decimation and narrowband channel filtering is done by a digital baseband filter in each path. The DSP performs:

1. The GMSK equalization of the received baseband signal
2. Viterbi channel decoding, which is supported by an hardware accelerator.

The recovered digital speech data is fed into the speech decoder. The E-GOLDradio supports fullrate, halfrate, enhanced fullrate and adaptive multirate speech CODEC algorithms.

The generated voice signal passes through a digital voiceband filter. The resulting 4 Mbit/s data stream is D-to-A converted by a multi-bit-oversampling converter, postfiltered, and amplified by a programmable gain stage.

The output buffer can drive a handset ear-piece or an external audio amplifier.

**CONFIDENTIAL****GSM/GPRS System Description**

In the transmit direction the microphone signal is fed into a programmable gain amplifier. The prefiltered and A-to-D converted voice signal forms a 2 Mbit/s data stream. The oversampled voice signal passes a digital decimation filter.

Speech and channel encoding (including voice activity detection (VAD) and discontinuous transmission (DTX)) as well as digital GMSK modulation is carried out by the E-GOLDRadio.

The digital I and Q baseband components of the GMSK modulated signals (48-times oversampled with 13 MSamples/s) are D-to-A converted. The analog differential baseband signals are fed into the RF part of the E-GOLDRadio.

In the RF transceiver part, the baseband signal modulates the RF carrier at the desired frequency in the 850 MHz, 900 MHz, 1.8 GHz, and 1.9 GHz bands using an I/Q modulator. The E-GOLDRadio supports quad band applications.

Finally, an RF power module amplifies the RF transmit signal to the required power level. Using software, the E-GOLDRadio controls the gain of the power amplifier by predefined ramping curves (16 words, 11 bits).

The E-GOLDRadio also includes battery charger support (various sensor connections for temperature, battery technology, voltage, etc.) and a ringer buffer.

For base band operation, the E-GOLDRadio supports:

- High Speed Circuit Switched Data (HSCSD) class 4
- Packet-oriented data (GPRS) class 4 with a coding scheme from 1 to 4. It provides fixed, dynamic, and extended dynamic modes.

If the E-GOLDRadio is only used as a modem, then it supports:

- High Speed Circuit Switched Data (HSCSD) class 10
- Packet-oriented data (GPRS) class 10 with a coding scheme from 1 to 4. It provides fixed, dynamic, and extended dynamic modes.

The E-GOLDRadio can support Class B operation. The mobile phone can be attached to both GPRS and GSM services, using one service at a time. During a GPRS connection Class B enables either:

- Making or receiving a voice call
- Sending or receiving an SMS.

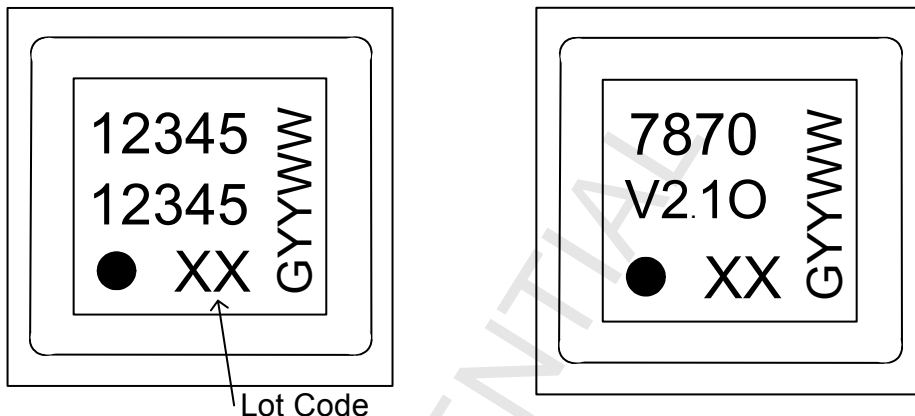
During voice calls or SMS, GPRS services are suspended and then resumed automatically after the call or SMS session has ended.

**CONFIDENTIAL**

### 1.3 Package

The package is a SG-LF2BGA233-1 (a “flipchip” with 233 pins), see [Figure 1-2](#).

**Figure 1-2 Package Label (Character height = 1.0 mm)**



Legend for <a href="#">Figure 1-2</a>		An example for E-GOLDradio V2.10
1234(5)	PMB number (position 5 not used)	7870
1234(5)	Product version (position 5 not used)	
	1 = Prefix	V
	2 = Major full mask redesign	2
	3 = Firmware mask	.1
	4 = Minor redesign (metal bug fix)	O
GYYWW	G is for the green package	G
	YY for year number modulus 100 For the engineering samples the first Y is replaced by the letter "E".	E4
	WW for week number in the year	50

**CONFIDENTIAL**
**Version Naming**
**1.4 Version Naming**
**Table 1-1 E-GOLDRadio Version Naming**

Product Code and Version	Development Code	Firmware Mask Code	Mask Change	Package Label	Description
<b>PMB 7870 V1.0</b>	M2899-A001-11	G14		7870 V1.1B •01 GE450	First Version (Rocket Samples)
<b>PMB 7870 V1.0</b>	M2899-A001-11	G14		7870 V2.0 •2A GE452	First Version
<b>PMB 7870 V1.0</b>	M2899-A001-12	G14	RB	7870 2.0	Yield improvement: DRC violations minimized
<b>PMB 7870 V1.0C</b>	M2899-A001-15	G14	CA, CL, BQ, RV, RB, RT Boot Code V2.0	7870 V2.1A •2C GE516	Performance improvement: Reduction of RF spurs
<b>PMB 7870 V1.1A</b>	M2899-A002-15	G15	CA, CL, BQ, RV, RB, RT Boot Code V2.0	7870 V2.1B •2B GE516	As A001-15
<b>PMB 7870 V2.0D</b>	M2899-A001-21	G14	Full Mask Redesign Boot Code V2.2	7870 2.0D GE520 •	New SD2 VCO New LNA layout Supply blocking to reduce spurs Improved RX-symmetry to improve AM suppression
<b>PMB 7870 V2.0E</b>	M2899-A001-31	G14	Full Mask Redesign Boot Code V2.2	7870 2.0E GE510 •	As A001-21 plus SD3 VCO
<b>PMB 7870 V2.1F</b>	M2899-A002-41	G15	Full Mask Redesign Boot Code V2.2	7870 2.1F	Bug fixes for: RTC EBU PDC
<b>PMB 7870 V2.1G</b>	M2899-A002-42	G15	Full Mask Redesign Boot Code V2.2	7870 2.1G	Same as <a href="#">PMB 7870 V2.1F</a> + A variant of the RF Macro
<b>PMB 7870 V2.1H</b>	M2899-A002-43	G15	Full Mask Redesign Boot Code V2.2	7870 2.1H	Same as <a href="#">PMB 7870 V2.1F</a> + A variant of the RF Macro
<b>PMB 7870 V2.2F</b>	M2899-A003-41	G16	Full Mask Redesign Boot Code V2.2	7870 2.2F	Bug fixes for: RTC EBU PDC
<b>PMB 7870 V2.2G</b>	M2899-A003-42	G16	Full Mask Redesign Boot Code V2.2	7870 2.2G	Same as <a href="#">PMB 7870 V2.2F</a> + A variant of the RF Macro

**CONFIDENTIAL**

**Version Naming**

**Table 1-1 E-GOLDradio Version Naming**

Product Code and Version	Development Code	Firmware Mask Code	Mask Change	Package Label	Description
<b>PMB 7870 V2.2H</b>	M2899-A003-43	G16	Full Mask Redesign Boot Code V2.2	7870 2.2H	Same as <b>PMB 7870 V2.2F</b> + A variant of the RF Macro
<b>PMB 7870 V2.2I</b>	M2899-A003-44	G16	BQ, RV, RB and RT	7870 2.2I	Same as <b>PMB 7870 V2.2F</b> with fix DCXO startup
<b>PMB 7870 V2.2J</b>	M2899-A003-45	G16	BQ, RV, RB and RT	7870 2.2J	Same as <b>PMB 7870 V2.2G</b> with fix DCXO startup
<b>PMB 7870 V2.2K</b>	M2899-A003-46	G16	BQ, RV, RB and RT	7870 2.2K	Same as <b>PMB 7870 V2.2H</b> with fix DCXO startup
<b>PMB 7870 V2.2L</b>	M2899-A003-47	G16	RB	7870 2.2L	Based on <b>PMB 7870 V2.2I</b> with fix to re-center VCO.
<b>PMB 7870 V2.2M</b>	M2899-A003-48	G16	RB	7870 2.2M	Based on <b>PMB 7870 V2.2J</b> with fix to re-center VCO.
<b>PMB 7870 V2.1O</b>	M2899-A002-50	G15	M1 to M4, V1,V2,V3 RV, CL, RB, BQ, RT with HVT (PR, NR)	7870 2.1O	Based on <b>PMB 7870 V2.1F</b> with: <ul style="list-style-type: none"> <li>Fix for Min Power</li> <li>Fix for VDD_MAIN consumption in standby mode when DSP not in reset</li> <li>Fix for Latchup issue: Weakness on TRSTN, changed the polarity of a MOS capacitor</li> <li>Fix for the problem of increased current due to ESD diode</li> <li>Fix for ESD failures: new package with VSS_PLL connected to VSS_MAIN</li> <li>Fix for RF Digital timing problems: only digital TX path is supported</li> <li>IDA disconnected from I/Q interface</li> <li>FSYS2 and FSYS3 outputs are inverted</li> <li>Increase boost time for DCXO startup to 3 ms</li> </ul>
<b>PMB 7870 V2.2O</b>	M2899-A003-50	G16	RB	7870 2.2O	Same as <b>PMB 7870 V2.1O</b> but with G16 DSP ROM

## **1.5 Bus Concept**

The E-GOLDRadio has two cores (a microcontroller and a DSP); each with its own bus. They are described in [Section 1.5.1 \(on page 43\)](#) and [Section 1.5.2 \(on page 43\)](#).

There is an interconnections between the TEAKlite bus and the C166S X-Bus as explained in [Section 1.5.3 \(on page 43\)](#).

### **1.5.1 C166S Buses**

The C166S is connected to three buses:

1. Local Memory (LM) bus
2. X-Bus
3. PD-Bus.

### **1.5.2 TEAKlite Bus**

The TEAKlite is connected to the TEAKlite bus.

### **1.5.3 Bus Interconnections**

The interconnection between the X-Bus and the TEAKlite Bus uses:

- Multicore Synchronization
- Shared Memory.

## **1.6 Clock Concept**

The E-GOLDRadio has flexible clock control.

## **1.7 Interrupt Concept**

The E-GOLDRadio interrupt system is performed by the C166 MCU.

## **1.8 Debug Concept**

The E-GOLDRadio includes a multi-core debug. The C166 and TEAKlite cores can be debugged in parallel with:

- A single JTAG port (that is, on a single host)
- Mutual breakpoint control.

### **1.8.1 C166 Debug Concept**

The debugging of the C166 used the OCDS and the Cerberus.

### **1.8.2 TEAKLite Debug Concept**

TEAKLite debugging uses the OCEM and the SEIB.

## **1.9 Power Management**

The flexible clock switching options allow minimizing the power consumption during the operation phases of the PMB7870.

The main task of the power management is to reduce current consumption during the standby mode by reducing the clock to 32 kHz and switching it off for most of the device. In addition, the power supply for the TEAKLite ROM is switched off and the controller RAM is switched to a power saving mode.

Optionally, the power supply for the TEAKLite subsystem may be switched off by external power management.

### **1.10 Security Concept**

The E-GOLDradio security features are not publicly documented.

### **1.11 Asynchronous Operation Mode Concept**

The E-GOLDradio can operate in either:

- The traditional synchronous mode where the 26 MHz system clock is synchronized on the base station
- A special asynchronous mode (XO concept).  
In the asynchronous mode the 26 MHz clock input is not synchronized with the base station; the residual frequency offset is compensated in the digital signal processing domain. This processing includes frequency and timing compensation of the baseband and voiceband signals.



**CONFIDENTIAL**













## 2 Pin Diagram

History	
Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.01	
<b>Page 54</b>	Pin D8 name updated in <b>Table 2-2 E-GOLDradio Ballout</b> also Pin A4 name updated WS00006752
<b>Page 54</b>	Updated in <b>Table 2-2 E-GOLDradio Ballout</b> WS00006922
Changes for Rev. 1.02	
<b>Page 54</b>	EFUSE removed from <b>Table 2-2 E-GOLDradio Ballout</b> WS00007257
Changes for Rev. 1.04	
<b>Page 47</b>	<b>Table 2-2</b> replaced by spreadsheet in <b>Figure 2-2 E-GOLDradio Ballout</b> WS00008438
<b>Page 46</b>	Updated <b>Figure 2-1 Legend for E-GOLDradio Ballout</b> WS00008455
Changes for Rev. 1.06	

## 2.1 E-GOLDradio Ballout

**Figure 2-2** shows the location of the balls on the E-GOLDradio package and their associated signals and voltages (refer to **Figure 2-2**). The ballout view in **Figure 2-2** is from above the chip, that is, the ball connections on the mobile motherboard<sup>1)</sup>.

**Figure 2-1 Legend for E-GOLDradio Ballout**

	Not connected
	RF Power domain balls
	SIM Power domain balls
	VDD_MAIN balls
	VDD_DIGA power domain balls (Test and low activity)
	EBU power domain balls
	RTC power domain balls
	VDD_DIGB power domain balls (low activity)
	Mixed Signal power domain balls
	Mixed signal audio power domain balls
	VSS Balls
	VDD_PLL power domain balls

<sup>1)</sup> The ballout view in **Figure 18-1 E-GOLDradio: LF2BGA-233 Package Details and Ball Labelling (on Page 1426)** is from the bottom of the chip.

Figure 2-2 E-GOLDradio Ballout

	A	B	C	D	E	F	G	H	J	K	L	M	N	P	R	T	U																									
17	VSS_RF	RX1X	RX1	RX2X	RX2	RX3X	RX3	RX4X	RX4	VREFP	MICP1	MICN1	VSS_MS	VDDOBR	VSS_MS	EPPA2	EPPA2																									
16	TX2	TRIG_OUT	VSS_RF	VSS_RF	VSS_RF	VSS_RF	VSS_RF	VDDBG	PAOUT1	VSS_MS	MICP2	MICN2	VSS_MS	EPPA1	VDDOBR	M0	M1																									
15	TX1	TRIG_IN	VSS_RF	VSS_RF	VSS_RF	VSS_RF	VSS_RF	REXT	VMCN	IREF	VREFN	VDDOBT	EPN1	EPPA1	M10	M8	M2																									
14	TMS	MON2	MON1	VDD_TRX	VDD_LO	VSS_RF	VSS_RF	VDD_RX	VMCP	AGND VSS_MS	BB_I	BB_IX	VDDO	EPP1	M9	M7	VDDM																									
13	TDI	TCK	TRST_n	VDD_VCO	<table><tr><td>VDD_DIG</td><td>PAOUT1s</td><td>BB_QX</td><td>BB_Q</td><td>EPP1s</td></tr><tr><td>VSS_RF</td><td>VDD_DSP</td><td>VDDP_PLL</td><td>VSS_PLL</td><td>VSS</td></tr><tr><td>VDD_DSP</td><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td></tr><tr><td>VDD_DSP</td><td>VSS</td><td>VSS</td><td>VDD_MAIN</td><td>VSS</td></tr><tr><td>VDD_DSP</td><td>VSS</td><td>VDD_DSP</td><td>VDD_MAIN</td><td>VDD_MAIN</td></tr></table>									VDD_DIG	PAOUT1s	BB_QX	BB_Q	EPP1s	VSS_RF	VDD_DSP	VDDP_PLL	VSS_PLL	VSS	VDD_DSP	VSS	VSS	VSS	VSS	VDD_DSP	VSS	VSS	VDD_MAIN	VSS	VDD_DSP	VSS	VDD_DSP	VDD_MAIN	VDD_MAIN	VDDOBB	VSS_MS	RFSTR3	RFSTR2
VDD_DIG	PAOUT1s	BB_QX	BB_Q	EPP1s																																						
VSS_RF	VDD_DSP	VDDP_PLL	VSS_PLL	VSS																																						
VDD_DSP	VSS	VSS	VSS	VSS																																						
VDD_DSP	VSS	VSS	VDD_MAIN	VSS																																						
VDD_DSP	VSS	VDD_DSP	VDD_MAIN	VDD_MAIN																																						
12	XOX	FSYS3	TDO	VDD_XQ	VSS_MS	VSS_MS	RFSTR1	RFSTR0																																		
11	XO	FSYS2	KP0	VSS_RF	VSS_MS	EPN1s	RFDATA	RFCLK																																		
10	KP2	KP4	KP3	KP7	VSS	SSC1_MTSR	SSC1_MRST	I2S2_RX																																		
9	KP8	KP5	KP6	VSS	CCIN	I2S3_WA	SSC1_CLK	I2S3_WA0																																		
8	ASC_CTS_n	KP9	RXD	VDDP_DIGA	CCIO	I2S3_CLK	I2S2_CLK0	I2S2_TX																																		
7	ASC_RTS_n	TXD	SSC0_CLK	SSC0_MRST	CCLK	I2S3_TX	VDDP_DIGB	I2S1_RX																																		
6	A0	SSC0_MTSR	SCL	SDA	CCVZ_n	T_OUT10	T_OUT0	I2S1_CLK0																																		
5	A2	A1	A3	AFC	CCRST	T_OUT1	T_OUT11	I2S1_WA0																																		
4	A5	A4	A6	A8	VDDP_EBU	A17	A20	A22	VSS	D7	VSS	F32K	OSC32K	VDDP_SIM	T_OUT3	T_OUT4	I2S1_TX																									
3	A7	A10	A9	A14	A15	A16	CS1_n	A23	D4	WR_n	RTCOUT	VDD_RTC	PM_INT	RESET_IN_n	LPAOUT0	VCXO_EN	T_OUT8																									
2	A12	A11	CS0_n	A13	RD_n	BHE_n	A19	D0	D3	D9	ADV_n	D11	D14	T_OUT7	T_OUT2	T_OUT5	CLKOUT																									
1	NC	CS3_n	OE_n	A18	A21	D1	D2	D6	D5	D8	D10	D13	D12	D15	RSTOUT_N	CC00IO	EFUSE																									

CONFIDENTIAL

**CONFIDENTIAL**

### 3 Pin Descriptions

<b>History</b>	
Current version: Rev. 1.06, 2005-11-04	
Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial version based <i>E-GOLDradio Design Specification</i> , Rev. 2.02
Changes for 0.01	
	Add A23 on the EBU, move alternative function from A22 to A23 with the goal of having A22 with the same behavior as A0-A21
Changes for 0.02	
<b>Page 57</b>	Added BB_I, BB_IX, BB_Q, BB_QX
Changes for Rev. 1.01	
<b>Page 51</b>	Updated <b>Table 3-1 Pin List</b> WS00006922
Changes for Rev. 1.04	
<b>Page 58</b> <b>Page 61</b>	Corrections to <b>Table 3-1 Pin List</b> WS00008539 WS00006914
Changes for Rev. 1.06	
<b>Page 50</b>	Updated Legend for <b>Table 1-1</b>

CONFIDENTIAL

Table 3-1 describes the E-GOLDradio ballout.

Legend for Table 3-1:

Name	Definitions
ST:	Schmitt Trigger (Input pads)
BU:	Buffer (Output pads)
BU_I2C:	I <sup>2</sup> C buffer
PU+PD only:	Only Pull-Up and Pull-Down
Fixed PD:	Fixed at Pull-Down
Fixed PU:	Fixed at Pull-Up
Programmed?:	The pin be programmed (yes/no)
T:	Tristate (output driver disabled)

Pin reset default mode

CONFIDENTIAL

Table 3-1 Pin List

Ball	Ball Name	Bump Name Alt 0	Dir	Bump Name Alt 1	Dir	Bump Name Alt 2	Dir	GPIO	Reset Value <sup>1)</sup>	Supply Domain	ST or BU	Program?	Driver Class	Pull-up/down Pull-down Class
#Keypad														
C11	KP0	KP0	O	T2EUD	I	EX0IN	I	GPIO_00	T	VDDP_DIGA	ST	yes	F	B
A10	KP2	KP2	O	CC20IOb	I/O	A/D_0	O	GPIO_01	T	VDDP_DIGA	ST	yes	F	B
C10	KP3	KP3	O	CC16IOb	I/O	A/D_1	O	GPIO_02	T	VDDP_DIGA	ST	yes	F	B
B10	KP4	KP4	O	CC01IOa	I/O	A/D_2	O	GPIO_03	T	VDDP_DIGA	ST	yes	F	B
B9	KP5	KP5	O	CC02IO	I/O	A/D_3	O	GPIO_04	T	VDDP_DIGA	ST	yes	F	B
C9	KP6	KP6	I	EX5IN	I	A/D_4	O	GPIO_05	T	VDDP_DIGA	ST	yes	F	B
D10	KP7	KP7	I	T7IN	I	A/D_5	O	GPIO_06	T	VDDP_DIGA	ST	yes	F	B
A9	KP8	KP8	I	CC22IOa	I/O	A/D_6	O	GPIO_07	T	VDDP_DIGA	ST	yes	F	B
B8	KP9	KP9	I	CC18IOa	I/O	A/D_7	O	GPIO_08	T	VDDP_DIGA	ST	yes	F	B
#ASC														
C8	RXD	RXD	I/O	EX1IN	I	T5EUD	I	GPIO_09	T	VDDP_DIGA	ST	yes	E	C
B7	TXD	TXD	O	CC04IO	I/O	NA	NA	GPIO_10	T	VDDP_DIGA	ST	yes	E	C
A8	ASC_RTS_n	ASC_RTS_n	O	CC18IOb	I/O	DSP_INT3	I	GPIO_11	T	VDDP_DIGA	ST	yes	E	C
A7	ASC_CTS_n	ASC_CTS_n	I	CAPIN	I	CC00IOb	I/O	GPIO_12	T/PU	VDDP_DIGA	ST	yes	F	C
#SSC0														
C7	SSC0_CLK	SSC0_CLK	I/O	CC00IO	I/O	T2IN	I	GPIO_13	T	VDDP_DIGA	ST	yes	D(E)	B
D7	SSC0_MRST	SSC0_MRST	I/O	LPAOUT1	O	DSP_INT4	I	GPIO_14	T	VDDP_DIGA	ST	yes	D(E)	B
B6	SSC0_MTSR	SSC0_MTSR	I/O	CC17IO	I/O	T6IN	I	GPIO_15	T	VDDP_DIGA	ST	yes	D(E)	B
#I2C														
C6	SCL	B13	I/O	NA	NA	NA	NA	GPIO_16	T/OD	VDDP_DIGA	BU_I2C	Fixed OD		External Pull-up
D6	SDA	B14	I/O	NA	NA	NA	NA	GPIO_17	T/OD	VDDP_DIGA	BU_I2C	Fixed OD		External Pull-up

**CONFIDENTIAL**
**Table 3-1 Pin List**

Ball	Ball Name	Bump Name Alt 0	Dir	Bump Name Alt 1	Dir	Bump Name Alt 2	Dir	GPIO	Reset Value <sup>1)</sup>	Supply Domain	ST or BU	Program?	Driver Class	Pull-up/down	Pull-down Class
#Sim Card I/F															
P8	CCIO	CCIO	I/O	NA	NA	NA	NA	NA	OD/L	VDDP_SIM	BU	Fixed OD	E		B
P6	CCVZ_n	CCVZ_n	O	CC16IOa	I/O	TTRACE	O	GPIO_18	H	VDDP_SIM	BU	yes	E		B
P7	CCLK	CCLK	O	NA	NA	NA	NA	NA	L	VDDP_SIM	BU	NA	E		B
P5	CCRST	CCRST	O	NA	NA	NA	NA	NA	L	VDDP_SIM	BU	NA	E		B
P9	CCIN	CCIN	I	T6OUT	O	CCIOSW	O	GPIO_19	T/OD	VDDP_SIM	ST	yes	E		B
#I2S1: DAI-PCM															
U6	I2S1_CLK0	I2S1_CLK0	I/O	CC05IO	I/O	TXD1	O	GPIO_20	T/OD	VDDP_DIGB	ST	yes	E		B
U7	I2S1_RX	I2S1_RX	I	CC06IOa	I/O	DSPIN1	I	GPIO_21	T/OD	VDDP_DIGB	ST	yes	E		B
U4	I2S1_TX	I2S1_TX	O	CC06IOb	I/O	A23	O	GPIO_22	T/OD	VDDP_DIGB	ST	yes	E		B
U5	I2S1_WA0	I2S1_WA0	I/O	DSPOUT2	O	RXD1	I/O	GPIO_23	T/OD	VDDP_DIGB	ST	yes	E		B
#I2S2: Digital Audio I/F															
T8	I2S2_CLK0	I2S2_CLK0	I/O	EX6IN	I	SSC1_CLK	I/O	GPIO_24	T/OD	VDDP_DIGB	ST	yes	E		B
U10	I2S2_RX	I2S2_RX	I	CC07IO	I/O	SSC1_MTSR	I/O	GPIO_25	T/OD	VDDP_DIGB	ST	yes	E		B
U8	I2S2_TX	I2S2_TX	O	EX4BIN	I	SSC1_MRST	I/O	GPIO_26	T/OD	VDDP_DIGB	ST	yes	E		B
U9	I2S2_WA0	I2S2_WA0	I/O	T3OUT	O	T0IN	I	GPIO_27	T/OD	VDDP_DIGB	ST	yes	E		B
#I2S3: Digital BB I/F															
R8	I2S3_CLK	I2S3_CLK	I/O	CC05IO	I/O	I2S2_CLK1	I/O	GPIO_28	T/OD	VDDP_DIGB	ST	yes	E		B
R7	I2S3_TX	I2S3_TX	O	CC06IOa	I/O	NA	NA	GPIO_29	T/OD	VDDP_DIGB	ST	yes	E		B
R9	I2S3_WA	I2S3_WA	I/O	CC06IOb	I/O	I2S2_WA1	I/O	GPIO_30	T/OD	VDDP_DIGB	ST	yes	E		B
#Jtag/Debug															
C12	TDO	TDO	O	NA	NA	NA	NA	NA	T	VDDP_DIGA	ST	NA	D		C
A13	TDI	TDI	I	NA	NA	NA	NA	NA	PU	VDDP_DIGA	ST	fixed PU	F		C



**CONFIDENTIAL**
**Table 3-1 Pin List**

Ball	Ball Name	Bump Name Alt 0	Dir	Bump Name Alt 1	Dir	Bump Name Alt 2	Dir	GPIO	Reset Value <sup>1)</sup>	Supply Domain	ST or BU	Program?	Driver Class	Pull-up/down	Class
A14	TMS	TMS	I	NA	NA	NA	NA	NA	PU	VDDP_DIGA	ST	fixed PU	F	C	
B13	TCK	TCK	I	NA	NA	NA	NA	NA	PD	VDDP_DIGA	ST	fixed PD	F	C	
C13	TRST_n	TRST_n	I	NA	NA	NA	NA	NA	PD	VDDP_DIGA	ST	fixed PD	F	C	
B16	TRIG_OUT	TRIG_OUT	O	NA	NA	NA	NA	NA	T	VDDP_DIGA	ST	NA	D	C	
B15	TRIG_IN	TRIG_IN	I	NA	NA	NA	NA	NA	PD	VDDP_DIGA	ST	fixed PD	F	C	
C14	MON1	MON1	I/O	NA	NA	NA	NA	NA	T/PD	VDDP_DIGA	BU	PU+ PD only	C	B	
B14	MON2	MON2	I/O	NA	NA	NA	NA	NA	T/PD	VDDP_DIGA	BU	PU+ PD only	C	B	
#External Bus I/F															
H2	D0	D0	I/O	NA	NA	NA	NA	NA	T	VDDP_EBU	ST	NA	C	C	
F1	D1	D1	I/O	NA	NA	NA	NA	NA	T	VDDP_EBU	ST	NA	C	C	
G1	D2	D2	I/O	NA	NA	NA	NA	NA	T	VDDP_EBU	ST	NA	C	C	
J2	D3	D3	I/O	NA	NA	NA	NA	NA	T	VDDP_EBU	ST	NA	C	C	
J3	D4	D4	I/O	NA	NA	NA	NA	NA	T	VDDP_EBU	ST	NA	C	C	
J1	D5	D5	I/O	NA	NA	NA	NA	NA	T	VDDP_EBU	ST	NA	C	C	
H1	D6	D6	I/O	NA	NA	NA	NA	NA	T	VDDP_EBU	ST	NA	C	C	
K4	D7	D7	I/O	NA	NA	NA	NA	NA	T	VDDP_EBU	ST	NA	C	C	
K1	D8	D8	I/O	NA	NA	NA	NA	NA	T	VDDP_EBU	ST	NA	C	C	
K2	D9	D9	I/O	NA	NA	NA	NA	NA	T	VDDP_EBU	ST	NA	C	C	
L1	D10	D10	I/O	NA	NA	NA	NA	NA	T	VDDP_EBU	ST	NA	C	C	

**Table 3-1 Pin List**

<b>Ball</b>	<b>Ball Name</b>	<b>Bump Name Alt 0</b>	<b>Dir</b>	<b>Bump Name Alt 1</b>	<b>Dir</b>	<b>Bump Name Alt 2</b>	<b>Dir</b>	<b>GPIO</b>	<b>Reset Value<sup>1)</sup></b>	<b>Supply Domain</b>	<b>ST or BU</b>	<b>Program?</b>	<b>Driver Class</b>	<b>Pull-up/down Pull-down Class</b>
M2	D11	D11	I/O	NA	NA	NA	NA	NA	T	VDDP_EBU	ST	NA	C	C
N1	D12	D12	I/O	NA	NA	NA	NA	NA	T	VDDP_EBU	ST	NA	C	C
M1	D13	D13	I/O	NA	NA	NA	NA	NA	T	VDDP_EBU	ST	NA	C	C
N2	D14	D14	I/O	NA	NA	NA	NA	NA	T	VDDP_EBU	ST	NA	C	C
P1	D15	D15	I/O	NA	NA	NA	NA	NA	T	VDDP_EBU	ST	NA	C	C
A6	A0	A0	O	NA	NA	NA	NA	NA	T/PD	VDDP_EBU	ST	NA	C	B
B5	A1	A1	O	NA	NA	NA	NA	NA	T/PD	VDDP_EBU	ST	NA	C	B
A5	A2	A2	O	NA	NA	NA	NA	NA	T/PD	VDDP_EBU	ST	NA	C	B
C5	A3	A3	O	NA	NA	NA	NA	NA	T/PD	VDDP_EBU	ST	NA	C	B
B4	A4	A4	O	NA	NA	NA	NA	NA	T/PD	VDDP_EBU	ST	NA	C	B
A4	A5	A5	O	NA	NA	NA	NA	NA	T/PD	VDDP_EBU	ST	NA	C	B
C4	A6	A6	O	NA	NA	NA	NA	NA	T/PD	VDDP_EBU	ST	NA	C	B
A3	A7	A7	O	NA	NA	NA	NA	NA	T/PD	VDDP_EBU	ST	NA	C	B
D4	A8	A8	O	NA	NA	NA	NA	NA	T/PD	VDDP_EBU	ST	NA	C	B
C3	A9	A9	O	NA	NA	NA	NA	NA	T/PD	VDDP_EBU	ST	NA	C	B
B3	A10	A10	O	NA	NA	NA	NA	NA	T/PD	VDDP_EBU	ST	NA	C	B
B2	A11	A11	O	NA	NA	NA	NA	NA	T/PD	VDDP_EBU	ST	NA	C	B
A2	A12	A12	O	NA	NA	NA	NA	NA	T/PD	VDDP_EBU	ST	NA	C	B
D2	A13	A13	O	NA	NA	NA	NA	NA	T/PD	VDDP_EBU	ST	NA	C	B
D3	A14	A14	O	NA	NA	NA	NA	NA	T/PD	VDDP_EBU	ST	NA	C	B
E3	A15	A15	O	NA	NA	NA	NA	NA	T/PD	VDDP_EBU	ST	NA	C	B
F3	A16	A16	O	NA	NA	NA	NA	NA	T/PD	VDDP_EBU	ST	NA	C	B
F4	A17	A17	O	NA	NA	NA	NA	NA	T/PD	VDDP_EBU	ST	NA	C	B
D1	A18	A18	O	NA	NA	NA	NA	NA	T/PD	VDDP_EBU	ST	NA	C	B

CONFIDENTIAL



**CONFIDENTIAL**
**Table 3-1 Pin List**

Ball	Ball Name	Bump Name Alt 0	Dir	Bump Name Alt 1	Dir	Bump Name Alt 2	Dir	GPIO	Reset Value <sup>1)</sup>	Supply Domain	ST or BU	Program?	Driver Class	Pull-up / Pull-down	Class
G2	A19	A19	O	NA	NA	NA	NA	NA	T/PD	VDDP_EBU	ST	NA	C	B	
G4	A20	A20	O	NA	NA	NA	NA	NA	T/PD	VDDP_EBU	ST	NA	C	B	
E1	A21	A21	O	NA	NA	NA	NA	NA	T/PD	VDDP_EBU	ST	NA	C	B	
H4	A22	A22	O	NA	NA	NA	NA	NA	T/PD	VDDP_EBU	ST	NA	C	B	
H3	A23	A23	O	DSPOUT0	O	NA	NA	GPIO_31	T/PD	VDDP_EBU	ST	yes	C	B	
C2	CS0_n	CS0_n	O	NA	NA	NA	NA	NA	T/PD	VDDP_EBU	ST	NA	C	B	
G3	CS1_n	CS1_n	O	EX4IN	I	NA	NA	GPIO_32	T/PD	VDDP_EBU	ST	yes	C	B	
B1	CS3_n	CS3_n	O	NMI_n	I	HLDA_n	O	GPIO_33	T/PD	VDDP_EBU	ST	yes	C	B	
E2	RD_n	RD_n	O	NA	NA	NA	NA	NA	T/PD	VDDP_EBU	ST	NA	C	B	
K3	WR_n	WR_n	O	NA	NA	NA	NA	NA	T/PD	VDDP_EBU	ST	NA	C	B	
F2	BHE_n	BHE_n	O	NA	NA	NA	NA	GPIO_34	T/PD	VDDP_EBU	ST	yes	C	B	
L2	ADV_n	ADV_n	O	CC20IOa	I/O	HOLD_n	I	GPIO_35	T/PD	VDDP_EBU	ST	yes	C	B	
C1	OE_n	OE_n	O	CC21IO	I/O	CS2_n	O	GPIO_36	T/PD	VDDP_EBU	ST	yes	C	B	
#RF Control Unit															
U12	RFSTR0	RFSTR0	O	NA	NA	NA	NA	NA	L	VDDP_DIGB	ST	NA	E	C	
T12	RFSTR1	RFSTR1	O	T3EUD	I	CLK32K	O	GPIO_37	T/PD	VDDP_DIGB	ST	yes	E	B	
U13	RFSTR2	RFSTR2	O	T5IN	I	READY_n	I	GPIO_38	T/PD	VDDP_DIGB	ST	yes	E	A	
T13	RFSTR3	RFSTR3	O	DSPOUT1	O	CC02IOb	I/O	GPIO_39	T/PD	VDDP_DIGB	ST	yes	E	A	
T11	RFDATA	RFDATA	O	NA	NA	NA	NA	NA	L	VDDP_DIGB	ST	NA	E	C	
U11	RFCLK	RFCLK	O	NA	NA	NA	NA	NA	L	VDDP_DIGB	ST	NA	E	C	
#GSM TDMA Timer															
T6	T_OUT0	T_OUT0	O	NA	NA	NA	NA	NA	T/PD	VDDP_DIGB	ST	NA	E	C	
R5	T_OUT1	T_OUT1	O	A22	O	NA	NA	GPIO_40	T/PD	VDDP_DIGB	ST	yes	E	A	
R2	T_OUT2	T_OUT2	O	CC07IO	I/O	NA	NA	GPIO_41	T/PD	VDDP_DIGB	ST	yes	B slow	B	

**Table 3-1 Pin List**

Ball	Ball Name	Bump Name Alt 0	Dir	Bump Name Alt 1	Dir	Bump Name Alt 2	Dir	GPIO	Reset Value <sup>1)</sup>	Supply Domain	ST or BU	Program?	Driver Class	Pull-up/down	Pull-down Class
R4	T_OUT3	T_OUT3	O	NA	NA	NA	NA	GPIO_42	T/PD	VDDP_DIGB	ST	yes	B slow		A
T4	T_OUT4	T_OUT4	O	LPAOUT3	O	NA	NA	GPIO_43	T/PD	VDDP_DIGB	ST	yes	B slow		A
T2	T_OUT5	T_OUT5	O	EX7IN	I	NA	NA	GPIO_44	T/PD	VDDP_DIGB	ST	yes	B slow		B
P2	T_OUT7	T_OUT7	O	LPAOUT2	O	NA	NA	GPIO_45	T/PD	VDDP_DIGB	ST	yes	E		B
U3	T_OUT8	T_OUT8	O	DSPIN0	I	T4EUD	I	GPIO_46	T/PD	VDDP_DIGB	ST	yes	E		B
R6	T_OUT10	T_OUT10	O	CC19IO	I/O	T4IN	I	GPIO_47	T/PD	VDDP_DIGB	ST	yes	E		A
T5	T_OUT11	T_OUT11	O	T6EUD	I	CS4_n	O	GPIO_48	T/PD	VDDP_DIGB	ST	yes	E		A
#Other Functional Pins: Clocks and Control															
D5	AFC	AFC	O	NA	NA	NA	NA	NA	T	VDDP_DIGA	ST	NA	D		C
U2	CLKOUT	CLKOUT	O	T_OUT6	O	CLK26M	O	GPIO_49	T/PD	VDDP_DIGB	ST	yes	D		B
M4	F32K	F32K	I/O analog	NA	NA	NA	NA	NA		VDD_RTC	NA	NA	#		#
N4	OSC32K	OSC32K	I analog	NA	NA	NA	NA	NA		VDD_RTC	NA	NA	#		#
L3	RTCOUT	RTCOUT	O	NA	NA	NA	NA	NA	#	VDD_RTC	ST	NA	F		C
P3	RESET_IN_n	RESET_IN_n	I	NA	NA	NA	NA	NA	#	VDD_RTC	BU	NA	#		B
R1	RSTOUT_n	RSTOUT_n	O	KP1	O	EX2IN	I	GPIO_50	T/PD	VDDP_DIGB	ST	yes	F		A
T3	VCXO_EN	VCXO_EN	O	T3IN	I	NA	NA	GPIO_51	H	VDDP_DIGB	ST	yes	F		B
N3	PM_INT	PM_INT	I	NA	NA	NA	NA	NA	T/PD	VDD_RTC	ST	NA	F		B
T9	SSC1_CLK	SSC1_CLK	I/O	T_OUT9	O	CC23IO	I/O	GPIO_53	T/PD	VDDP_DIGB	BU	yes	E		B
R10	SSC1_MTSR	SSC1_MTSR	I/O	CLK32K	O	CC03IO	I/O	GPIO_54	T/PD	VDDP_DIGB	BU	yes	D		B
T10	SSC1_MRST	SSC1_MRST	I/O	CC22IOb	I/O	EX3IN	I	GPIO_55	T/PD	VDDP_DIGB	BU	yes	D		B
T1	CC00IO	CC00IO	I/O	EX5BIN	I	I2S2_WA1	I/O	GPIO_56	T/PD	VDDP_DIGB	ST	yes	E		B
R3	LPAOUT0	LPAOUT0	O	CC02IO	I/O	I2S2_CLK1	I/O	GPIO_57	T	VDDP_DIGB	ST	yes	D		B
#Voiceband: Analog I/F															
N15	EPn1	EPn1	O	NA	NA	NA	NA	NA		VDDA_VBR	NA	NA			

CONFIDENTIAL



**CONFIDENTIAL**
**Table 3-1 Pin List**

Ball	Ball Name	Bump Name Alt 0	Dir	Bump Name Alt 1	Dir	Bump Name Alt 2	Dir	GPIO	Reset Value <sup>1)</sup>	Supply Domain	ST or BU	Program?	Driver Class	/dn- Pull-down	Class
R11	EPn1s	EPn1s	O	NA	NA	NA	NA	NA		VDDA_VBR	NA	NA			
P14	EPp1	EPp1	O	NA	NA	NA	NA	NA		VDDA_VBR	NA	NA			
L11	EPp1s	EPp1s	O	NA	NA	NA	NA	NA		VDDA_VBR	NA	NA			
P15	EPpa1	EPpa1_1	O	NA	NA	NA	NA	NA		VDDA_VBR					
P16	EPpa1	EPpa1_2	O	NA	NA	NA	NA	NA		VDDA_VBR					
T17	EPpa2	EPpa2	O	NA	NA	NA	NA	NA		VDDA_VBR					
U17	EPpa2	EPpa2_2	O	NA	NA	NA	NA	NA		VDDA_VBR					
M17	MICN1	MICN1	I	NA	NA	NA	NA	NA		VDDA_VBT					
L17	MICP1	MICP1	I	NA	NA	NA	NA	NA		VDDA_VBT					
M16	MICN2	MICN2	I	NA	NA	NA	NA	NA		VDDA_VBT					
L16	MICP2	MICP2	I	NA	NA	NA	NA	NA		VDDA_VBT					
J14	VMICP	VMICP	NA	NA	NA	NA	NA	NA		VDDA_BG					
J15	VMICN	VMICN	NA	NA	NA	NA	NA	NA		VDDA_BG					
#I/Q-Signal: Analog I/F Baseband															
J16	PAOUT1	PAOUT1	O	NA	NA	NA	NA	NA		VDDA_BB					
H11	PAOUT1s	PAOUT1s	O	NA	NA	NA	NA	NA		VDDA_BB					
L14	BB_I	BB_I	I/O	NA	NA	NA	NA	NA		VDDA_BB					
M14	BB_IX	BB_IX	I/O	NA	NA	NA	NA	NA		VDDA_BB					
K11	BB_Q	BB_Q	I/O	NA	NA	NA	NA	NA		VDDA_BB					
J11	BB_QX	BB_QX	I/O	NA	NA	NA	NA	NA		VDDA_BB					
#Measurement															
T16	M0	M0	I	NA	NA	NA	NA	NA		VDDA_M					
U16	M1	M1	I	NA	NA	NA	NA	NA		VDDA_M					
U15	M2	M2	I	NA	NA	NA	NA	NA		VDDA_M					

CONFIDENTIAL

Table 3-1 Pin List

Ball	Ball Name	Bump Name Alt 0	Dir	Bump Name Alt 1	Dir	Bump Name Alt 2	Dir	GPIO	Reset Value <sup>1)</sup>	Supply Domain	ST or BU	Program?	Driver Class	Pull-down Pull-up Class
T14	M7	M7	I	NA	NA	NA	NA	NA		VDDA_M				
T15	M8	M8	I	NA	NA	NA	NA	NA		VDDA_M				
R14	M9	M9	I	NA	NA	NA	NA	NA		VDDA_M				
R15	M10	M10	I	NA	NA	NA	NA	NA		VDDA_M				
#Bandgap ref: Analog I/F														
K17	VREFp	VREFp	NA	NA	NA	NA	NA	NA		VDDA_BG				
K15	IREF	IREF	NA	NA	NA	NA	NA	NA		VDDA_BG				
K14	AGND	AGND	NA	NA	NA	NA	NA	NA		VDD_BG	NA	NA		
L15	VREFn	VREFn	NA	NA	NA	NA	NA	NA		VDD_BG	NA	NA		

CONFIDENTIAL

Table 3-1 Pin List

Ball	Ball Name	Bump Name Alt 0	Dir	Bump Name Alt 1	Dir	Bump Name Alt 2	Dir	GPIO	Reset Value <sup>1)</sup>	Supply Domain	ST or BU	Program?	Driver Class	Pull-down Pull-up Class
#RF Macro Antenna														
C17	RX1	RX1	NA	NA	NA	NA	NA	NA						
B17	RX1X	RX1X	NA	NA	NA	NA	NA	NA						
E17	RX2	RX2	NA	NA	NA	NA	NA	NA						
D17	RX2X	RX2X	NA	NA	NA	NA	NA	NA						
G17	RX3	RX3	NA	NA	NA	NA	NA	NA						
F17	RX3X	RX3X	NA	NA	NA	NA	NA	NA						
J17	RX4	RX4	NA	NA	NA	NA	NA	NA						
H17	RX4X	RX4X	NA	NA	NA	NA	NA	NA						
#RF Macro Power Amplifier														
A15	TX1	TX1	NA	NA	NA	NA	NA	NA						
A16	TX2	TX2	NA	NA	NA	NA	NA	NA						
#RF Macro Crystal Oscillator														
A11	XO	XO	NA	NA	NA	NA	NA	NA						
A12	XOX	XOX	NA	NA	NA	NA	NA	NA						
#RF Macro Sys Clock or GPO														
B11	FSYS2	FSYS2	NA	NA	NA	NA	NA	NA						
B12	FSYS3	FSYS3	NA	NA	NA	NA	NA	NA						
#RF Macro External reference														
H15	REXT	REXT	NA	NA	NA	NA	NA	NA						

CONFIDENTIAL

**Table 3-1 Pin List**

Ball	Ball Name	Bump Name Alt 0	Dir	Bump Name Alt 1	Dir	Bump Name Alt 2	Dir	GPIO	Reset Value <sup>1)</sup>	Supply Domain	ST or BU	Program?	Driver Class	Pull-down Pull-up Class
#Digital Power Supply														
K8	VDD_MAIN	VDD_MAIN1	NA	NA	NA	NA	NA	NA		NA	NA	NA		
K7	VDD_MAIN	VDD_MAIN2	NA	NA	NA	NA	NA	NA		NA	NA	NA		
L7	VDD_MAIN	VDD_MAIN3	NA	NA	NA	NA	NA	NA		NA	NA	NA		
G9	VDD_DSP	VDD_DSP1	NA	NA	NA	NA	NA	NA		NA	NA	NA		
G8	VDD_DSP	VDD_DSP2	NA	NA	NA	NA	NA	NA		NA	NA	NA		
G7	VDD_DSP	VDD_DSP3	NA	NA	NA	NA	NA	NA		NA	NA	NA		
H10	VDD_DSP	VDD_DSP3	NA	NA	NA	NA	NA	NA		NA	NA	NA		
J7	VDD_DSP	VDD_DSP4	NA	NA	NA	NA	NA	NA		NA	NA	NA		
H9	VSS	VSS_MAIN1	NA	NA	NA	NA	NA	NA		NA	NA	NA		
H8	VSS	VSS_MAIN2	NA	NA	NA	NA	NA	NA		NA	NA	NA		
H7	VSS	VSS_MAIN3	NA	NA	NA	NA	NA	NA		NA	NA	NA		
J9	VSS	VSS_MAIN4	NA	NA	NA	NA	NA	NA		NA	NA	NA		
J8	VSS	VSS_DSP1	NA	NA	NA	NA	NA	NA		NA	NA	NA		
K9	VSS	VSS_DSP2	NA	NA	NA	NA	NA	NA		NA	NA	NA		
L10	VSS	VSS_DSP3	NA	NA	NA	NA	NA	NA		NA	NA	NA		
L9	VSS	VSS_DSP3	NA	NA	NA	NA	NA	NA		NA	NA	NA		
L8	VSS		NA	NA	NA	NA	NA	NA		NA	NA	NA		
M3	VDD_RTC	VDD_RTC	NA	NA	NA	NA	NA	NA		NA	NA	NA		
L4	VSS	VSS_RTC	NA	NA	NA	NA	NA	NA		NA	NA	NA		
P4	VDDP_SIM	VDDP_SIM	NA	NA	NA	NA	NA	NA		NA	NA	NA		
J10	VDDP_PLL	VDDP_PLL	NA	NA	NA	NA	NA	NA		NA	NA	NA		
K10	VSS_PLL	VSSP_PLL	NA	NA	NA	NA	NA	NA		NA	NA	NA		



**Table 3-1 Pin List**

Ball	Ball Name	Bump Name Alt 0		Bump Name Alt 1		Bump Name Alt 2		GPIO	Reset Value <sup>1)</sup>	Supply Domain	ST or BU	Program?	Driver Class	Pull-up/ Pull-down	Class
E4	VDDP_EBU	VDDP_EBU1	NA	NA	NA	NA	NA	NA		NA	NA	NA			
		VDDP_EBU2	NA	NA	NA	NA	NA	NA		NA	NA	NA			
		VDDP_EBU3	NA	NA	NA	NA	NA	NA		NA	NA	NA			
D8	VDDP_DIGA	VDDP_DIGA_1	NA	NA	NA	NA	NA	NA		NA	NA	NA			
		VDDP_DIGA_2	NA	NA	NA	NA	NA	NA		NA	NA	NA			
T7	VDDP_DIGB	VDDP_DIGB_1	NA	NA	NA	NA	NA	NA		NA	NA	NA			
		VDDP_DIGB_2	NA	NA	NA	NA	NA	NA		NA	NA	NA			
J4	VSS	VSSP_EBU_1	NA	NA	NA	NA	NA	NA		NA	NA	NA			
		VSSP_EBU_2	NA	NA	NA	NA	NA	NA		NA	NA	NA			
		VSSP_EBU_3	NA	NA	NA	NA	NA	NA		NA	NA	NA			
D9	VSS	VSSP_DIGA_1	NA	NA	NA	NA	NA	NA		NA	NA	NA			
		VSSP_DIGA_2	NA	NA	NA	NA	NA	NA		NA	NA	NA			
P10	VSS	VSSP_DIGB_1	NA	NA	NA	NA	NA	NA		NA	NA	NA			
		VSSP_DIGB_2	NA	NA	NA	NA	NA	NA		NA	NA	NA			
		VSSP_SIM	NA	NA	NA	NA	NA	NA		NA	NA	NA			
#Analog Power Supply															
P13	VDDBB	VDDbb_1	NA	NA	NA	NA	NA	NA		NA	NA	NA			
		VDDbb_2	NA	NA	NA	NA	NA	NA		NA	NA	NA			
R12	VSS_MS	VSSbb_1	NA	NA	NA	NA	NA	NA		NA	NA	NA			
		VSSbb_2	NA	NA	NA	NA	NA	NA		NA	NA	NA			
P17	VDDVBR	VDDVbr_1	NA	NA	NA	NA	NA	NA		NA	NA	NA			
		VDDVbr_2	NA	NA	NA	NA	NA	NA		NA	NA	NA			
R16	VDDVBR	VDDVbr_1	NA	NA	NA	NA	NA	NA		NA	NA	NA			
		VDDVbr_2	NA	NA	NA	NA	NA	NA		NA	NA	NA			

CONFIDENTIAL



**CONFIDENTIAL**
**Table 3-1 Pin List**

Ball	Ball Name	Bump Name Alt 0	Dir	Bump Name Alt 1	Dir	Bump Name Alt 2	Dir	GPIO	Reset Value <sup>1)</sup>	Supply Domain	ST or BU	Program?	Driver Class	Pull-up/ Pull-down Class
N16	VSS_MS	VSSVbr_1	NA	NA	NA	NA	NA	NA		NA	NA	NA		
		VSSVbr_2	NA	NA	NA	NA	NA	NA		NA	NA	NA		
R17	VSS_MS	VSSVbr_1	NA	NA	NA	NA	NA	NA		NA	NA	NA		
		VSSVbr_2	NA	NA	NA	NA	NA	NA		NA	NA	NA		
M15	VDDVBT	VDDVbt	NA	NA	NA	NA	NA	NA		NA	NA	NA		
N17	VSS_MS	VSSVbt	NA	NA	NA	NA	NA	NA		NA	NA	NA		
N14	VDDD	VDDd	NA	NA	NA	NA	NA	NA		NA	NA	NA		
P12	VSS_MS	VSSd	NA	NA	NA	NA	NA	NA		NA	NA	NA		
U14	VDDM	VDDm	NA	NA	NA	NA	NA	NA		NA	NA	NA		
R13	VSS_MS	VSSm	NA	NA	NA	NA	NA	NA		NA	NA	NA		
H16	VDDBG	VDDbg	NA	NA	NA	NA	NA	NA		NA	NA	NA		
K16	VSS_MS	VSSbg	NA	NA	NA	NA	NA	NA		NA	NA	NA		
P11	VSS_MS	GUARDA	NA	NA	NA	NA	NA	NA		NA	NA	NA		
		GUARDD	NA	NA	NA	NA	NA	NA		NA	NA	NA		
#RF Power Supply														
H14	VDD_RX		NA	NA	NA	NA	NA	NA		NA	NA	NA		
			NA	NA	NA	NA	NA	NA		NA	NA	NA		
D12	VDD_XO		NA	NA	NA	NA	NA	NA		NA	NA	NA		
			NA	NA	NA	NA	NA	NA		NA	NA	NA		
D11	VSS_RF		NA	NA	NA	NA	NA	NA		NA	NA	NA		
			NA	NA	NA	NA	NA	NA		NA	NA	NA		
D13	VDD_VCO		NA	NA	NA	NA	NA	NA		NA	NA	NA		
			NA	NA	NA	NA	NA	NA		NA	NA	NA		

CONFIDENTIAL

**Table 3-1 Pin List**

Ball	Ball Name	Bump Name Alt 0	Dir	Bump Name Alt 1	Dir	Bump Name Alt 2	Dir	GPIO	Reset Value <sup>1)</sup>	Supply Domain	ST or BU	Program?	Driver Class	Pull-down Pull-up Class
G14	VSS_RF		NA	NA	NA	NA	NA	NA		NA	NA	NA		
			NA	NA	NA	NA	NA	NA		NA	NA	NA		
			NA	NA	NA	NA	NA	NA		NA	NA	NA		
G15	VSS_RF		NA	NA	NA	NA	NA	NA		NA	NA	NA		
			NA	NA	NA	NA	NA	NA		NA	NA	NA		
			NA	NA	NA	NA	NA	NA		NA	NA	NA		
G11	VDD_DIG		NA	NA	NA	NA	NA	NA		NA	NA	NA		
G10	VSS_RF		NA	NA	NA	NA	NA	NA		NA	NA	NA		
E14	VDD_LO		NA	NA	NA	NA	NA	NA		NA	NA	NA		
D14	VDD_TRX		NA	NA	NA	NA	NA	NA		NA	NA	NA		
A17	VSS_RF		NA	NA	NA	NA	NA	NA		NA	NA	NA		
			NA	NA	NA	NA	NA	NA		NA	NA	NA		
C16	VSS_RF		NA	NA	NA	NA	NA	NA		NA	NA	NA		
			NA	NA	NA	NA	NA	NA		NA	NA	NA		
C15	VSS_RF		NA	NA	NA	NA	NA	NA		NA	NA	NA		
			NA	NA	NA	NA	NA	NA		NA	NA	NA		
D16	VSS_RF		NA	NA	NA	NA	NA	NA		NA	NA	NA		
			NA	NA	NA	NA	NA	NA		NA	NA	NA		
D15	VSS_RF		NA	NA	NA	NA	NA	NA		NA	NA	NA		
			NA	NA	NA	NA	NA	NA		NA	NA	NA		
E16	VSS_RF		NA	NA	NA	NA	NA	NA		NA	NA	NA		
			NA	NA	NA	NA	NA	NA		NA	NA	NA		
E15	VSS_RF		NA	NA	NA	NA	NA	NA		NA	NA	NA		
			NA	NA	NA	NA	NA	NA		NA	NA	NA		

CONFIDENTIAL

Table 3-1 Pin List

Ball	Ball Name	Bump Name Alt 0	Dir	Bump Name Alt 1	Dir	Bump Name Alt 2	Dir	GPIO	Reset Value <sup>1)</sup>	Supply Domain	ST or BU	Program?	Driver Class	Pull-down Pull-up Class
F16	VSS_RF		NA	NA	NA	NA	NA	NA		NA	NA	NA		
			NA	NA	NA	NA	NA	NA		NA	NA	NA		
F15	VSS_RF		NA	NA	NA	NA	NA	NA		NA	NA	NA		
			NA	NA	NA	NA	NA	NA		NA	NA	NA		
F14	VSS_RF		NA	NA	NA	NA	NA	NA		NA	NA	NA		
			NA	NA	NA	NA	NA	NA		NA	NA	NA		
G16	VSS_RF		NA	NA	NA	NA	NA	NA		NA	NA	NA		
			NA	NA	NA	NA	NA	NA		NA	NA	NA		

1) These reset values are valid when the PCL is reset at system startup:

T= Tristate (output driver disabled)

PU= Pull Up

PD= Pull Down

OD= Open Drain

L= Low level

H= High level

CONFIDENTIAL

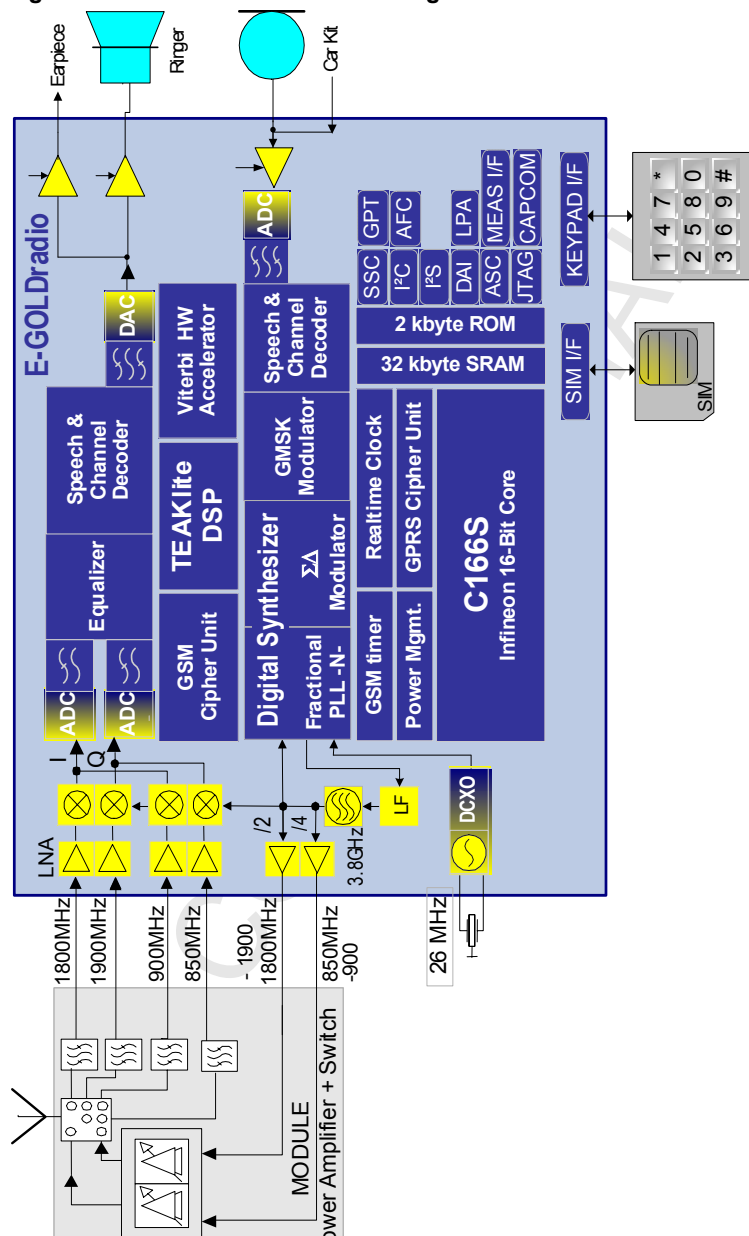
## 4 Block Diagram

History	
Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.01	
<b>Page 67</b>	Updated <b>Figure 4-2 Diagram of E-GOLDradio Buses</b> WS00006968
Changes for Rev. 1.04	
<b>Page 66</b>	Updated <b>Figure 4-1 E-GOLDradio Block Diagram</b> WS00008670
Changes for Rev. 1.05	

**Figure 4-1** is an overview of the E-GOLDradio functional blocks and **Figure 4-2** shows its buses.

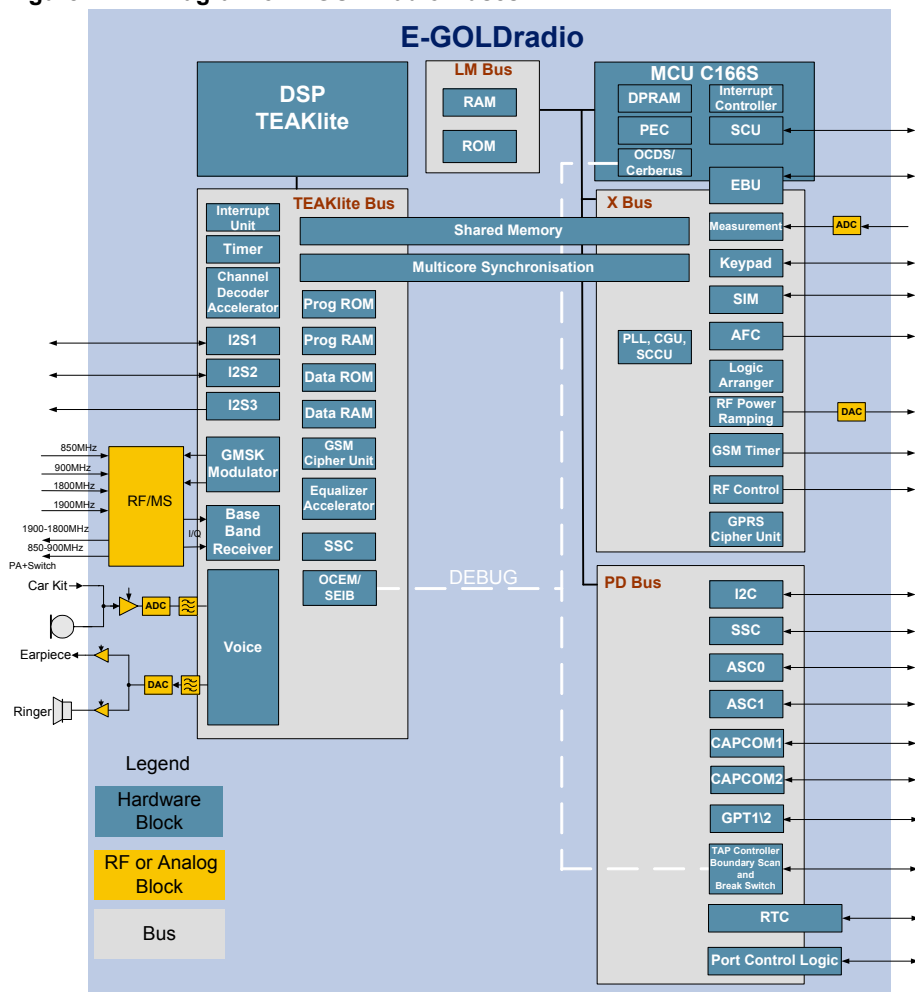
CONFIDENTIAL

Figure 4-1 E-GOLDradio Block Diagram



CONFIDENTIAL

Figure 4-2 Diagram of E-GOLDradio Buses



**CONFIDENTIAL**

CONFIDENTIAL



## 5 TEAKLite DSP

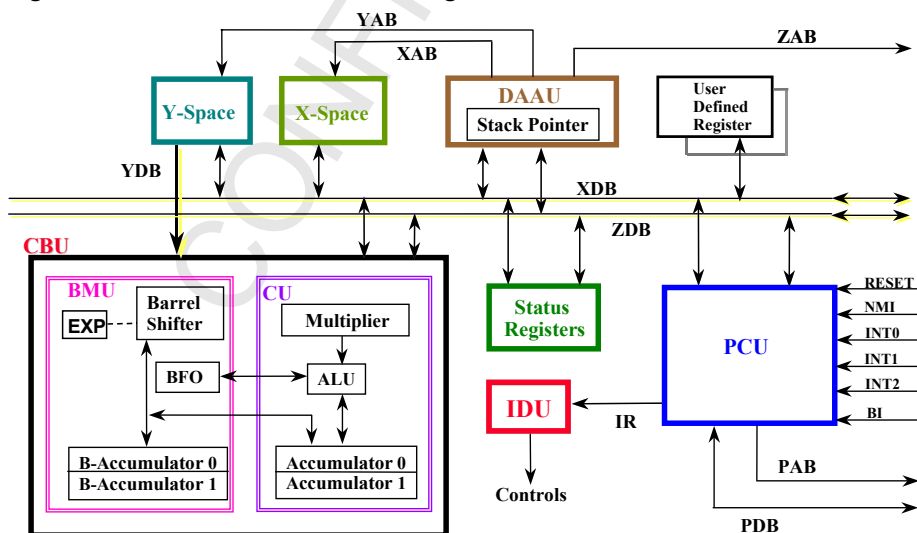
History	
Current version:	Rev. 1.06, 2005-11-04
Previous version:	Rev. 1.05, 2005-08-02
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.06	

### 5.1 Introduction

A block diagram of the TEAKLite core is shown in **Figure 5-1**. It consists of five units:

- Computation Unit (CU)
- Bit Manipulation Unit (BMU)
- Data Address Arithmetic Unit (DAAU)
- Program Control Unit (PCU)
- Instruction Decoder Unit (IDU).

**Figure 5-1 TEAKLite Core Block Diagram**



CONFIDENTIAL

TEAKLite Memory Organization

The Computation Unit consists of a 16 by 16 to 32 bit *multiplier* and a 36-bit *ALU* with two *accumulator* registers A0 and A1. The Bit Manipulation Unit consists of a full 36-bit *barrel shifter*, an *exponent* unit (EXP), a *bit-field operation* unit, two additional 36-bit *accumulator registers* B0 and B1 and a *shift value* register (SV).

The Data Address Arithmetic Unit performs all address storage and address calculation necessary to access data and program memories. Moreover it supports software stack pointer, loop counter and min/max operations.

The Program Control Unit performs instruction fetch and decoding, exception handling and hardware loop control. This unit is provided with three independent interrupt input lines INT0, INT1 and INT2, which are connected to the DSP interrupt unit. High levels on these lines are interpreted as active interrupts. INT0 has the highest priority, INT2 the lowest.

## 5.2 TEAKLite Memory Organization

### Memory Map

The E-GOLDradio TEAKLite contains on-chip memory as follows:

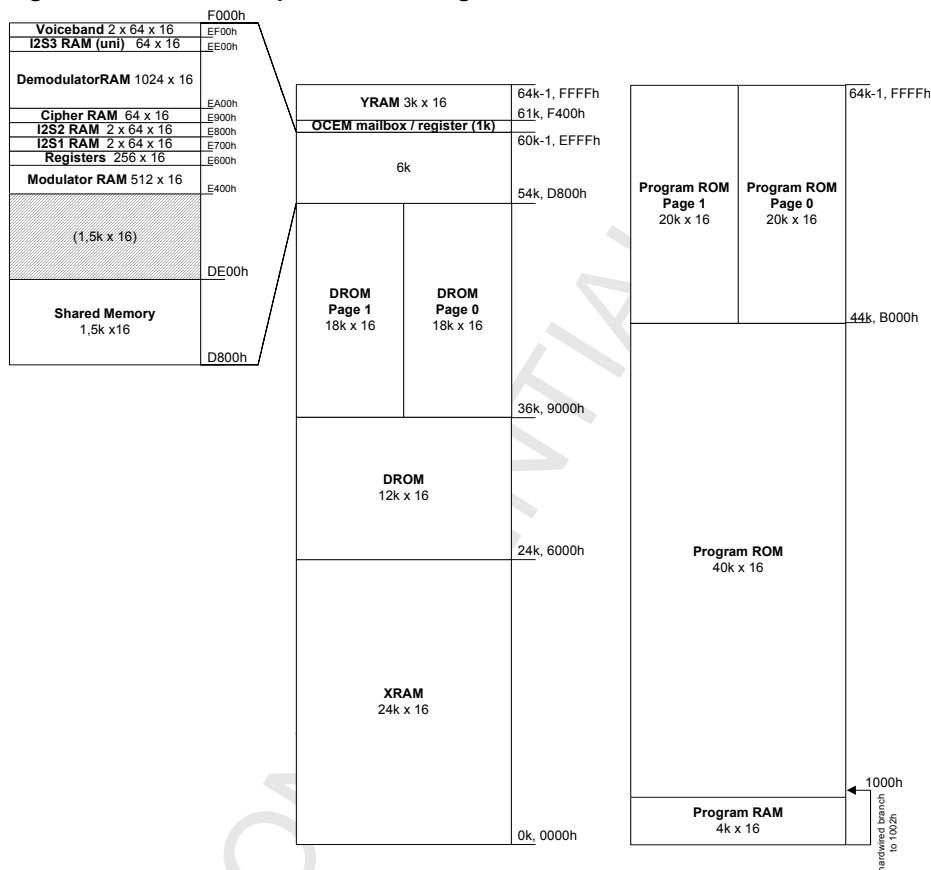
- Data memory (XRAM): 24k x 16 bit
- Data memory (DROM): 48k x 16 bit
- Data memory (YRAM): 3k x 16 bit
- Program memory (PRAM, PROM):
  - 4k x 16 bit program RAM
  - 80k x 16 bit program ROM.

**Figure 5-2** gives an overview of the TEAKLite memory and **Table 5-1** shows the address space partitioning. The number of wait states that are written in **Table 5-1** are each valid for the whole data segment.

**CONFIDENTIAL**

**TEAKLite Memory Organization**

**Figure 5-2 Address Space Partitioning on TEAKLite**



**CONFIDENTIAL**

**TEAKLite Memory Organization**

**Table 5-1 Partitioning of Data and Address Space on TEAKLite and Data Page Mode**

Data Address Range (Hex)		Size	Wait state	Function
0000-5FFF		24k x 16	0	XRAM
6000-8FFF		12k x 16	0	DROM
data Page0	9000-D7FF	18k x 16	0	DROM
data Page1	9000-D7FF	18k x 16	0	DROM
D800-DDFF		1.5k x 16	2	Shared Memory (dual port RAM)
DE00-E3FF		(1.5k)	-	Not available
E400-E5FF		512 x 16	2	Modulator RAM (dual port RAM)
E600-E6FF		256 x 16	2	Registers
		1 x 16	4	AFE_RWADDR
		1 x 16	4	BB_WR_POINTER
		1 x 16	4	I2S1_RWADDR
		1 x 16	4	I2S2_RWADDR
		1 x 16	4	I2S3_RADDE
E700-E77F		2*64 x 16	2	I2S1RAM
E780-E7FF		(128 x 16)	-	Not available
E800-E87F		2*64 x 16	2	I2S2RAM
E880-E8FF		(128 x 16)	-	Not available
E900-E93F		64 x 16	2	Cipher RAM
E940-E9FF		(192)	-	Not available
EA00-EDFF		1k x 16	2	Demodulator RAM (dual port RAM)
EE00-EE3F		64 x 16	2	I2S3 (uni) RAM
EE40-EEFF		(192 x 16)	-	Not available
EF00-EF7F		2*64 x 16	2	Voiceband, Audio Front End
EF80-EFFF		(128 x 16)	-	Not available
F000-F3DF		992	≥ 2	Reserved for OCEM mailbox (virtual dual port RAM)
F3E0-F3EF		16	-	Not available

**CONFIDENTIAL**

**TEAKLite Memory Organization**

**Table 5-1 Partitioning of Data and Address Space on TEAKLite and Data Page Mode**

F3F0-F3FF		16	2	Reserved for OCEM registers
F400-FFFF		3k x 16	0	YRAM
Program Address Range		Size		Function
0000-0001		4k x 16	0	Hardwired branch to normal mode (1002 <sub>H</sub> ) or test mode according to JTAG settings
0002-0FFF			0	Program RAM
1000-AFFF		40k x 16	0	Program ROM
program Page 0	B000-FFFF	20k x 16	0	Program ROM
program Page 1	B000-FFFF	20k x 16	0	Program ROM

A page mechanism has been implemented for extending program and data memory space. The **DSP\_PAGE (on Page 74)** register determines the accessible page by the TEAKLite core.

**CONFIDENTIAL**

**TEAKLite Memory Organization**

### 5.2.1 DSP Page Select Register

**DSP\_PAGE**

**DSP Page Select Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														PRO G P A G E	DAT A P A G E

Field	Bits	Type	Description
<b>DATA_PAGE</b>	0	rw	<b>Data ROM Page Select @ [9000<sub>H</sub>...D7FF<sub>H</sub>]</b> 0 Select page 0 1 Select page 1
<b>PROG_PAGE</b>	1	rw	<b>Program ROM Page Select @ [B000<sub>H</sub>...FFFF<sub>H</sub>]</b> 0 Select page 0 1 Select page 1
<b>RESERVED</b>	15:2	r	Reserved, these bits must be left at their reset values.

After writing to **DSP\_PAGE** the selected data page is available within 4 DSP cycles. That means whenever the TEAKLite changes the value of the register there are 4 NOP commands required before the first access can be applied to the selected area. ROM memory which is not part of the data pages can be accessed immediately.

*Note: Modifying of bit **DSP\_PAGE.PROG\_PAGE** is not allowed if the program is inside any page in the address range B000<sub>H</sub> to FFFF<sub>H</sub>. That means the program page must only be selected in a DSP program which is located in the address range 0000<sub>H</sub> to AFFF<sub>H</sub>. The firmware has to take care of that.*

*Note: The firmware must be take care that the monitor program of the OCEM is not located in the page area.*

### 5.3 Hardware Version

In **DSP\_ID** the version number of the TEAKLite hardware blocks can be read. The reset value is incremented for every change of the TEAKLite hardware blocks.

#### 5.3.1 DSP Identification Register

##### DSP\_ID

##### DSP Identification Register

**Reset value: E001<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DSPID															

Field	Bits	Type	Description
<b>DSPID</b>	15:0	r	Contains the revision number of the hardware blocks.

*Note: After RESET during the TEAKLite boot phase the DSP always writes the Firmware and the Hardware version to the shared memory location. For a detailed description refer to the E-GOLDradio Firmware User Manual.*

## 5.4 Boot Address

The program address 0000<sub>H</sub> of the DSP contains a hardwired branch to the Firmware entry address at 1002<sub>H</sub>. That means whenever the TEAKLite is started from RESET it jumps to this address. The address 1002<sub>H</sub> and 1003<sub>H</sub> contains a branch to the boot function of the firmware normal mode.

Additionally, it is possible to start the DSP in different test modes. These special test modes are only accessible through the JTAG interface. That means the user has to preselect the test mode via JTAG. After resetting the DSP it directly jumps to a special address.

An overview of the TEAKLite boot address setting is given in [Table 5-2](#).

**Table 5-2 Boot Address Selection**

JTAG setting		PROM Entry	
		Branch to Address	Description
000	Normal Mode	1002 <sub>H</sub>	Firmware entry
001	Test Mode	1004 <sub>H</sub>	Fast boot for test
010		1006 <sub>H</sub>	Self test
011		1008 <sub>H</sub>	Not used
100		100A <sub>H</sub>	Not used
101		100C <sub>H</sub>	Not used
110		100E <sub>H</sub>	Not used
111		0020 <sub>H</sub>	Jump directly into firmware startup code (program RAM)

## 5.5 DSP Power Saving Modes

Three different power saving modes are supported by the E-GOLDRadio DSP subsystem:

- Idle
- Clock off
- Power-Down.

### 5.5.1 DSP in IDLE Mode

The DSP can force itself into the IDLE mode by setting bit **DSP\_CTRL.DSPDIS**. In this mode the internal TEAKLite clock is switched off and TEAKLite stops working. During IDLE mode all peripherals can be used normally. The DSP is started again by any



**CONFIDENTIAL**

**DSP Power Saving Modes**

interrupt on lines INT0 or INT1 or INT2. The TEAKLite itself is responsible that peripherals not used during IDLE are switched off.

*Note: At least one interrupt (for example, **INT\_FINTA0 (on Page 341).MCU0**) must be enabled by the DSP in order to have the possibility to restart the DSP.*

### 5.5.1.1 DSP Control Register

#### DSP\_CTRL

#### DSP Control Register

Reset value: 000?H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															DSP DIS

Field	Bits	Type	Description
<b>DSPDIS</b>	0	w	Set by TEAKLite by writing any value to <b>DSP_CTRL</b> , reset by hardware if high level on INT0, INT1 or INT2. 0 The clock for TEAKLite core is switched on. 1 The clock for TEAKLite core is switched off.
<b>RESERVED</b>	15:1	r	Reserved, these bits must be left at their reset values.

*Note: After writing to **DSP\_CTRL**, the bit **DSP\_CTRL.DSPDIS** is set to 1 within 3DSP cycles. After an interrupt Int0, Int1, or Int2, the value of bit **DSP\_CTRL.DSPDIS** is set to 0 within 3DSP cycles.*

### 5.5.2 Clock Off

In this case the CGU switches off the clock signal *gclk\_dsp* in the whole DSP subsystem. The MCU has to go through the following sequence when applying this mode:

- First of all the MCU has to switch off voiceband and I2S peripheral by giving the appropriate TEAKLite command (see “E-GOLDRadio Firmware User Manual”).
- Then the MCU has to take care that the DSP is in IDLE state and its internal clock is switched off by itself (refer to **RST\_CTRL\_STA.DSP\_STATUS**)
- The System Interface must be configured in a way that the baseband and modulator DSP peripherals are not started via signals *rxon* or *txon*. The MCU has to take care of that.

After these steps the MCU may switch off the clock of the DSP subsystem in the CGU.

*Note: Leaving this clock off mode can be done by switching on the clock *gclk\_dsp* in the CGU. There is no DSP reset necessary.*

## CONFIDENTIAL

## Hardware Reset

### 5.5.3 Power-Down

Power down means the clock *gc/k\_dsp* is switched off, furthermore, the power of the whole DSP subsystem is switched off by the MCU.

Before the MCU switches off the power of the DSP subsystem the following steps have to be executed by the MCU:

1. The MCU must switch off voice band and I2S peripheral by giving the appropriate DSP command (refer to the *E-GOLDRadio Firmware User Manual*).
2. The MCU has to make sure that the DSP is in IDLE state and its internal clock is switched off by itself (the DSP clock status is given in [RST\\_CTRL\\_STA.DSP\\_STATUS](#)).
3. The MCU has to make sure that the System Interface is configured so that the baseband and modulator DSP peripherals are not started via signals *rxon* or *txon*.
4. The MCU switches off the clock of the DSP subsystem.

Now the DSP subsystem is ready for switching off the power (refer to [Section 14.2.2 Power On/Off Sequences \(on Page 1407\)](#)).

### 5.5.4 Power-Up

Before the MCU initiates the power-up procedure it has to activate *cgu\_reset\_dsp* of the SCU. This resets the whole DSP subsystem. After power-up the MCU may release *cgu\_reset\_dsp* and the DSP starts from address 0000<sub>H</sub>.

## 5.6 Hardware Reset

During reset TEAKLite, memories, and interrupt unit are clocked with the frequency coming from the external oscillator, which is either 13 MHz or 26 MHz:

- Different reset sources are possible but the DSP subsystem gets only one signal via line *cgu\_reset\_dsp* from the SCU. The different conditions on which signal *cgu\_reset\_dsp* is given are described in [Section 14.2 Reset \(on Page 1404\)](#).
- All registers are reset asynchronously during reset. The clock of each peripheral is set according to the initial value of its control register.
- The clock of the TEAKLite itself and the DSP interrupt unit is always switched on after reset.

## 5.7 DSP Debug Register

This register reflects the clock status of TEAKLite OCEM peripheral.

**CONFIDENTIAL**

**DSP Debug Register**

**DSP\_DEBUG**  
**DSP Debug Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								RES ERV ED	RES ERV ED	RES ERV ED	RES ERV ED	RES ERV ED	RES ERV ED	OCE M	RES ERV ED

Field	Bits	Typ	Description
<b>OCEM</b>	1	r	Clock status of TEAKLite OCEM peripheral 0 OCEM clock is switched off 1 OCEM clock is switched on
<b>RESERVED</b>	15:2, 0	r	Reserved; these bits must be left at their reset values.

CONFIDENTIAL

**CONFIDENTIAL**

**Pad Access Register**

## 5.8 Pad Access Register

Via the pad access register **DSP\_DSPOUT** the DSP core has direct read or write access to certain pads. The external input/output signals that can be read/written are *dspout* (bits 2:0), *dspin* (bits 1:0), and *mon* (bits 2:1). The mapping of these signals to pins is defined by programming the port control registers.

### DSP\_DSPOUT

#### DSP Pad Access Register

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED							MON IN4	MON IN3	MON IN2	MON IN1	DSP IN1	DSP IN0	DSP OUT 2	DSP OUT 1	DSP OUT 0
r							r	r	r	r	r	r	rw	rw	rw

Field	Bits	Type	Description
<b>DSPOUT0</b>	0	rw	<b>Controls Output Pin DSPOUT0</b> Toggled by firmware, if PMB7870 received a data word by the serial communication interface.
<b>DSPOUT1</b>	1	rw	<b>Controls Output Pin DSPOUT1</b> <b>DSPOUT1</b> can be used as general purpose output if the signal is multiplexed to a pin.
<b>DSPOUT2</b>	2	rw	<b>Controls Output Pin DSPOUT2</b> <b>DSPOUT2</b> can be used as general purpose output if the signal is multiplexed to a pin.
<b>DSPIN0</b>	3	r	<b>Latches Input Pin DSPIN0</b> <b>DSPIN0</b> can be used as general purpose input, if the signal is multiplexed to a pin.
<b>DSPIN1</b>	4	r	<b>Latches Input Pin DSPIN1</b> <b>DSPIN1</b> can be used as general purpose input, if the signal is multiplexed to a pin.
<b>MONIN1</b>	5	r	<b>Latches Input Pin MON1</b>
<b>MONIN2</b>	6	r	<b>Latches Input Pin MON2</b>
<b>MONIN3</b>	7	r	<b>Latches Input Pin DSP_INT3</b>
<b>MONIN4</b>	8	r	<b>Latches Input Pin DSP_INT4</b>
<b>RESERVED</b>	15:9	r	Reserved; these bits must be left at their reset values.

CONFIDENTIAL

**CONFIDENTIAL**

## 6 C166S MCU

History	
	Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.01	
<b>Page 315</b>	Reset Note updated in <b>Section 6.10.3 Address Bus Functions</b> WS00006730
<b>Page 258</b>	Updated register <b>SYSCON1</b> WS00005667
<b>Page 206</b>	Updated <b>XADRSx</b> registers WS00007127
Changes for Rev. 1.02	
<b>Page 265</b>	Updated <b>Figure 6-28 Clock Control on PD Bus</b> WS00007397
<b>Page 265</b>	Removed note about ECO block above <b>Figure 6-28 Clock Control on PD Bus</b> WS00007397
Changes for Rev. 1.03	
<b>Page 316</b>	Updated <b>Section 6.10.5 Chip Select and HOLD Functions</b>
Changes for Rev. 1.04	
<b>Page 87</b>	Note added for <b>Figure 6-1 Basic Block Diagram of a C166S System</b> WS00008670
<b>Page 84</b>	Note moved to footnote in <b>Section 6.1.2 Summary of Basic Features</b> WS00008670
<b>Page 111</b>	Added 4th bullet for PEC interrupts
Changes for Rev. 1.05	
<b>Page 318</b>	A(22:0) -> A(23:0) WS00008970
<b>Page 315</b>	Updated definition of address line A23 WS00008970
<b>Page 315</b>	Updated modes for the address bus pins in <b>Section 6.10.3 Address Bus Functions</b> WS00008970
Changes for Rev. 1.06	
<b>Page 258</b>	Updated reset value of <b>SYSCON1</b> WS00009047
<b>Page 245</b>	Updated <b>Section 6.7.4.1.1 Watchdog Timer Control Register</b> WS00009048
<b>Page 263</b>	Added <b>Section 6.7.6.7 Wakeup Trigger for Idle or Sleep Mode with Standby Clock Mode</b> WS00009096
<b>Page 110</b>	Updated bit <b>SYSCON.BYTDIS</b> WS00005895
<b>Page 315</b> <b>Page 318</b>	Updated "Immediately after RESET" in <b>Section 6.10.3 Address Bus Functions</b> , "Data Bus" and "Address Bus" in <b>Section 6.10.9 Visible Mode</b> WS00008970

## **6.1 Introduction**

The rapidly growing area of embedded control applications represents one of the most time-critical operating environments for today's microcontrollers. Complex control algorithms have to be processed based on a large number of digital and analog input signals, and the appropriate output signals must be generated within a defined maximum response time. Embedded control applications, such as E-GOLDradio, are sensitive to board space, power consumption, and overall system cost.

Embedded control applications, therefore, require microcontrollers, which:

- Offer a high level of system integration
- Eliminate the need for additional peripheral devices and the associated software overhead
- Provide system security and fail-safe mechanisms
- Provide effective means to control (and reduce) the device's power consumption.

### **6.1.1 C166S - A Member of 16-bit Microcontroller Family**

The microcontroller-subsystem of the Infineon 16-bit family has been designed to meet the high performance requirements of real-time embedded control applications. The architecture of this family has been optimized for high instruction throughput and minimum response time to external stimuli (interrupts). Intelligent peripheral subsystems have been integrated to reduce the need for MCU (Central Processing Unit) intervention. This also minimizes the need for communication via the external bus interface.

The core of the 16-bit family has been developed with a modular family concept in mind. All family members execute an efficient control-optimized instruction set.

The Internal Bus Interface (IBI) concept and the internal X-Bus opens a straight forward path for the integration of application specific peripheral modules in addition to the standard on-chip peripherals on the PD-Bus.

### **6.1.2 Summary of Basic Features**

The C166S is a full featured 16-bit single-chip CMOS (Complementary Metal Oxide Silicon) microcontroller. It combines high MCU performance with high peripheral functionality.

Several key features described below contribute to the high performance of the C166S based subsystem.



### **High Performance 16-Bit MCU With Four-Stage Pipeline**

- 38 ns minimum instruction cycle time<sup>1)</sup> with most instructions executed in 1 cycle (2 clocks)
- 192 ns multiplication (16-bit x 16-bit), 384 ns division (32-bit/16-bit)
- Parallel use of multiple high bandwidth internal data buses
- Register based design with multiple variable register banks
- Single cycle context switching support
- 16 MBytes linear address space for code and data (von Neumann architecture)
- System stack cache support with automatic stack overflow/underflow detection.

### **Control Oriented Instruction Set with High Efficiency**

- Bit, byte, and word data types
- Flexible and efficient addressing modes for high code density
- Enhanced boolean bit manipulation with 6-kbits direct addressing of for peripheral control and user defined flags
- Hardware traps to identify exception conditions during runtime
- HLL support for semaphore operations and efficient data access.

### **External Bus Interface**

- Demultiplexed bus configurations
- Segmentation capability and chip select signal generation
- 8-bit or 16-bit data bus
- Bus cycle characteristics selectable for five programmable address areas.

### **16-Priority-Level Interrupt System**

- Up to 112 interrupt nodes with separate interrupt vectors
- 16 priority levels and 8 group levels.

<sup>1)</sup> These timings refer to a 52 MHz MCU clock.

## **16-Channel Peripheral Event Controller (PEC)**

- Interrupt driven single cycle data transfer
- Transfer count option (standard MCU interrupt after a programmable number of PEC transfers)
- Long Transfer Counter
- Channel Linking
- Elimination of overhead for saving and restoring system state for interrupt requests

## **6.2 System Overview**

The architecture of the C166S combines the advantages of both RISC (Reduced Instruction Set Computing) and CISC (Complex Instruction Set Computing) processors in a well-balanced way. C166S based derivatives not only integrate a powerful MCU (Central Processing Unit) core and a set of peripheral units into one chip, but also connects the units in a very efficient way. One of the four buses used concurrently on the C166S is the Internal Bus Interface, an internal representation of the external bus interface. This bus provides a standardized method of integrating application-specific peripherals to produce derivatives of the standard C166S.

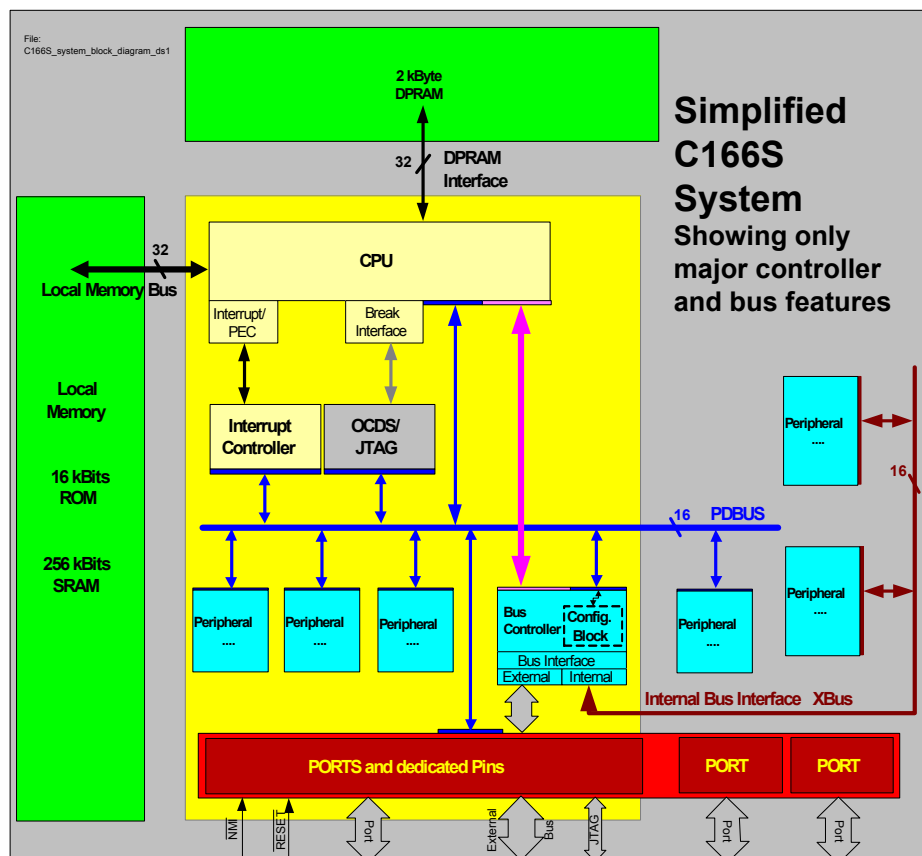
### **The Principle Elements of a C166S Based System**

- MCU block including the configurable Interrupt/PEC controller and debug and break logic
- Configurable dual port RAM
- Configurable Interrupt/PEC controller
- All interfaces for system (on chip) integration, including X-Bus, PD peripheral bus, Local Memory bus (for ROM or SRAM).

The C166 architecture (see [Figure 6-1](#)) allows instruction execution and data access from all memory locations. This includes X-Bus, local memory bus, dual port and external memories.

All four bus Interfaces of the MCU (X-Bus, LM Bus, RAM Bus and PD Bus) are operated on at the same time by the MCU.

Figure 6-1 Basic Block Diagram of a C166S System



Note: The Bus Controller in **Figure 6-1** is the bridge between the C166S CPU and the External and Internal bus interfaces.

## 6.2.1 Basic MCU Concepts

The main core of the MCU consists of a 4-stage instruction pipeline, a 16-bit arithmetic and logic unit (ALU) and dedicated Special Function Registers (SFRs). Additional hardware is provided for a separate multiply and divide unit, a bit-mask generator and a barrel shifter.

To meet the demand for greater performance and flexibility, a number of areas has been optimized in the processor core. Functional blocks in the MCU core are controlled by

**CONFIDENTIAL****System Overview**

signals from the instruction decode logic. These are summarized below and described in detail in the following sections:

- 1) High Instruction Bandwidth/Fast Execution
- 2) 8-bit and 16-bit Arithmetic and Logic Unit
- 3) Extended Bit Processing and Peripheral Control
- 4) High Performance Branch-, Call-, and Loop-Processing
- 5) Consistent and Optimized Instruction Formats
- 6) Programmable Multiple Priority Interrupt Structure.

**6.2.1.1 High Instruction Bandwidth/Fast Execution**

Based on the hardware provisions, most C166S instructions can be executed in just one machine cycle, which requires 2 MCU clock cycles T1 and T2

( $2 * 1/f_{CPU} = 4$  TCLs). For example, shift and rotate instructions are always processed within one machine cycle, independent of the number of bits to be shifted.

Branch, multiply and divide instructions normally take more than one machine cycle. These instructions, however, have also been optimized.

A 32-bit /16-bit division takes 20 MCU clock cycles, a 16-bit \* 16-bit multiplication takes 10 MCU clock cycles.

The instruction cycle time has been reduced through the use of instruction pipelining. This technique allows the core MCU to process portions of multiple sequential instruction stages in parallel. The four stage pipeline provides the optimum balancing for the MCU core (refer to [Section 6.3.7 Instruction Pipeline \(on Page 180\)](#)):

1. FETCH
2. DECODE
3. EXECUTE
4. WRITE BACK.

If this technique were not used, each instruction would require four machine cycles.

**Instruction Decoder**

Instruction decoding is primarily generated from PLA (Programmable Logic Array) outputs based on the selected opcode. No microcode is used and each pipeline stage receives control signals staged in control registers from the decode stage PLAs. Pipeline holds are primarily caused by waitstates for external memory accesses and cause the holding of signals in the control registers. Multiple-cycle instructions are performed through instruction injection and simple internal state machines which modify the required control signals.

### **6.2.1.2 8-bit and 16-bit Arithmetic and Logic Unit**

All standard arithmetic and logical operations are performed in a 16-bit ALU. In addition, for byte operations, signals are provided from bits six and seven of the ALU result to correctly set the condition flags. Multiple precision arithmetic is provided through a 'CARRY-IN' signal to the ALU from previously calculated portions of the desired operation.

Most internal execution blocks have been optimized to perform operations on either 8-bit or 16-bit quantities. Once the pipeline has been filled, one instruction is completed per machine cycle, except for multiply and divide. An advanced Booth algorithm has been incorporated to allow four bits to be multiplied and two bits to be divided per machine cycle. Thus, these operations use two coupled 16-bit registers, MDL (Multiply Divide Low Word) and MDH (Multiply Divide High Word), and require four and nine machine cycles, respectively, to perform a 16-bit by 16-bit (or 32-bit by 16-bit) calculation plus one machine cycle to setup and adjust the operands and the result. Even these longer multiply and divide instructions can be interrupted during their execution to allow for very fast interrupt response. Instructions have also been provided to allow byte packing in memory while providing sign extension of bytes for word-wide arithmetic operations. The internal bus structure also allows transfers of bytes or words to or from peripherals based on the peripheral requirements.

A set of consistent flags is automatically updated in the PSW (Program Status Word) after each arithmetic, logical, shift, or movement operation. These flags allow branching on specific conditions. Support for both signed and unsigned arithmetic is provided through user-specifiable branch tests. These flags are also preserved automatically by the MCU upon entry into an interrupt or trap routine. All targets for branch calculations are also computed in the central ALU.

A 16-bit barrel shifter provides multiple bit shifts in a single cycle. Rotates and arithmetic shifts are also supported.

### **6.2.1.3 Extended Bit Processing and Peripheral Control**

A large number of instructions has been dedicated to bit processing. These instructions provide efficient control and testing of peripherals while enhancing data manipulation. Unlike other microcontrollers, these instructions provide direct access to two operands in the bit-addressable space without requiring to move them into temporary flags.

The same logical instructions available for words and bytes are also supported for bits. This allows the user to compare and modify a control bit for a peripheral in one instruction. Multiple-bit shift instructions have been included to avoid long instruction streams of single-bit shift operations. These are also performed in a single machine cycle.

In addition, bit field instructions have been provided, which allow the modification of multiple bits from one operand in a single instruction.

### **6.2.1.4 High Performance Branch, Call, and Loop Processing**

Due to the high percentage of branching in controller applications, branch instructions have been optimized to require one extra machine cycle only when a branch is taken. This is implemented by precalculating the target address while decoding the instruction. To decrease loop execution overhead, three enhancements have been provided:

1. Two cycle branch execution after the first iteration of a loop. Thus, only one additional machine cycle is lost during the execution of the entire loop. In loops which fall through upon completion, no additional machine cycles is lost when exiting the loop. No special instructions are required to perform loops, and loops are automatically detected during execution of branch instructions.
2. The detection of the end of a table thus avoiding the use of two compare instructions embedded in loops. One simply places the lowest negative number at the end of the specific table, and specifies branching if neither this value nor the compared value have been found. Otherwise the loop is terminated if either condition has been met. The terminating condition can then be tested.
3. A more flexible solution than the Decrement and Skip on Zero instruction which is found in other microcontrollers. Through the use of Compare and Increment or Decrement instructions, the user can make comparisons to any value. This allows loop counters to cover any range. This is particularly advantageous in table searching.

Saving of system state is automatically performed on the internal system stack avoiding the use of instructions to preserve state upon entry and exit of interrupt or trap routines. Call instructions push the value of the **IP** on the system stack and require the same execution time as branch instructions.

Instructions have also been provided to support indirect branch and call instructions. This supports implementation of multiple CASE statement branching in assembler macros and high level languages.

### **6.2.1.5 Consistent and Optimized Instruction Formats**

To obtain optimum performance in a pipelined design, an instruction set has been designed which incorporates RISC concepts. These concepts primarily allow fast decoding of the instructions and operands while reducing pipeline holds. These concepts, however, do not preclude the use of complex instructions, which are required by microcontroller users. The following goals were used to design the instruction set:

1. Provide powerful instructions to perform operations which currently require sequences of instructions and are frequently used. Avoid transfer into and out of temporary registers such as accumulators and carry bits. Perform tasks in parallel such as saving state upon entry into interrupt routines or subroutines.
2. Avoid complex encoding schemes by placing operands in consistent fields for each instruction. Also avoid complex addressing modes which are not frequently used. This decreases the instruction decode time while also simplifying the development of compilers and assemblers.
3. Provide the most frequently used instructions with one-word instruction formats. All other instructions are placed into two-word formats. This allows all instructions to be placed on word boundaries, which alleviates the need for complex alignment hardware. It also has the benefit of increasing the range for relative branching instructions.

The high performance offered by the hardware implementation of the MCU can efficiently be utilized by a programmer via the highly functional C166S instruction set which includes the following instruction classes:

- Arithmetic Instructions
- Logical Instructions
- Boolean Bit Manipulation Instructions
- Compare and Loop Control Instructions
- Shift and Rotate Instructions
- Prioritize Instruction
- Data Movement Instructions
- System Stack Instructions
- Jump and Call Instructions
- Return Instructions
- System Control Instructions
- Miscellaneous Instructions.

Possible operand types are bits, bytes and words. Specific instruction support the conversion (extension) of bytes to words. A variety of direct, indirect or immediate addressing modes are provided to specify the required operands.

### **6.2.1.6 Programmable Multiple Interrupt Priorities and PEC System**

The following enhancements allow processing of a large number of interrupt sources:

**1. Peripheral Event Controller (PEC):**

This processor is used to off-load many interrupt requests from the MCU. It avoids the overhead of entering and exiting interrupt or trap routines by performing single-cycle interrupt-driven byte or word data transfers between any two locations with an optional increment of either the PEC source or the destination pointer. Just one cycle is 'stolen' from the current MCU activity to perform a PEC service.

**2. Multiple Priority Interrupt Controller (ITC):**

This controller allows all interrupts to be placed at any specified priorities. Interrupts may also be grouped, which provides the user with the ability to prevent similar priority tasks from interrupting each other. For each of the possible interrupt sources there is a separate control register, which contains an interrupt request flag, an interrupt enable flag, and an interrupt priority bitfield. Once accepted by the MCU, an interrupt service can only be interrupted by a higher priority service request. For standard interrupt processing, each of the possible interrupt sources has a dedicated vector location.

**3. Multiple Register Banks:**

This feature allows the user to specify up to sixteen general purpose registers located anywhere in the internal DPRAM (Dual Port RAM). A single one-machine-cycle instruction allows to switch register banks from one task to another.

**4. Interruptible Multiple Cycle Instructions:**

Reduced interrupt latency is provided by allowing multiple-cycle instructions (multiply, divide) to be interruptible.

**5. Hardware Traps:**

The C166S provides a mechanism to identify and to process exceptions or error conditions that arise during run-time, so called 'Hardware Traps'. Hardware traps cause an immediate non-maskable system reaction which is similar to a standard interrupt service (branching to a dedicated vector table location). The occurrence of a hardware trap is additionally signified by an individual bit in the Trap Flag Register (TFR). Except if a higher priority trap service is in progress, a hardware trap will interrupt any current program execution. However, hardware trap services cannot normally be interrupted by standard or PEC interrupts.

**6. Software Traps:**

Software interrupts are supported by means of the 'TRAP' instruction in combination with an individual trap (interrupt) number.

### **6.2.2 E-GOLDradio System Resources**

A C166S based subsystem provides a number of powerful system resources designed around the MCU. The combination of MCU and these resources results in the high performance of the members of this controller family.



### 6.2.2.1 Memory Areas

The memory space of the C166S is configured in a Von Neumann architecture which means that code memory, data memory, registers and I/O ports are organized within the same linear address space which covers up to 16 MBytes. The entire memory space can be accessed byte-wise or word-wise. Particular portions of the on-chip memory are directly bit addressable.

#### 2kByte 32-Bit Program/16-Bit DPRAM

A 2kByte 32-Bit Program/16-bit data wide internal Dual Port RAM (DPRAM) provides fast access to General Purpose Registers (GPRs), user data (variables) and the system stack. The DPRAM may also be used for code. A unique decoding scheme provides flexible user register banks in the internal memory while optimizing the remaining RAM for user data.

The MCU has an actual register context consisting of up to 16 word-wide and/or byte-wide GPRs at its disposal, which are physically located within the on-chip RAM area. A Context Pointer (**CP**) register determines the base address of the active register bank to be accessed by the MCU at a time. The number of register banks is only restricted by the available DPRAM space. For easy parameter passing, one register bank may overlap others.

A system stack is provided for storage of temporary data. The system stack is also located within the on-chip RAM area, and it is accessed by the MCU via the stack pointer (**SP (on Page 161)**) register. Two separate SFRs, STack OVerflow (**STKOV (on Page 162)**) and STack UNderflow (**STKUN (on Page 163)**), are implicitly compared against the stack pointer (**SP**) value on each stack access to detect a stack overflow or underflow.

Hardware detection of the selected memory space is placed at the internal memory decoders and allows the user to specify any address directly or indirectly and obtain the desired data without using temporary registers or special instructions.

**For Special Function Registers** 1024 Bytes of the address space are reserved. The standard Special Function Register area (SFR) uses 512 bytes, while the Extended Special Function Register area (ESFR) uses the other 512 bytes. (E)SFRs are word-wide registers, which are used for controlling and monitoring functions of the different on-chip units.

**A Local Memory** is provided for both code and data storage. This memory area is connected to the MCU via a 32-bit-wide local memory bus. Program execution from Local Memory is the fastest of all possible alternatives.

## **6.2.2.2 Internal Bus Systems**

### **Local Memory Bus**

The LM Bus (16-bit data read-write/32-bit program read) allows 32 bits to be fetched in parallel from the parts of the internal local memory (2kbyte ROM, 32kbyte SRAM) which are used for a program. For the RAM memory, the LM Bus also provides write operations. The LM access is 0 ws@26 MHz CPUCLK resp (ws = wait state).

0 ws@52 MHz. At 52 MHz the hardware will NOT automatically insert a wait state.

*Note: For LM-Bus ROM and RAM wait state information, refer to [Section 9.1 RAM \(on Page 615\)](#).*

### **Peripheral Bus**

The PD-Bus allows MCU and PEC (Peripheral Event Controller) access to core-related and generic (standard) peripherals every half machine cycle. The bus cycles are fully synchronized without wait states. The bus also supports read-modify-write cycles without performance losses. Bit modification is provided, and bit protection for simultaneous writes is also supported. However, only 512 16-bit registers can be addressed via the demultiplexed address bus.

### **RAM Bus**

The RAM bus interface provides two 16-bit buses between the dual-port RAM and the MCU. Its bus cycles are fully synchronized, with a cycle time of half a machine cycle; thus, four accesses can be made every machine cycle. This performance allows multiple GPR (General Purpose Register) accesses to occur without processor stalls. The RAM bus is an internal bus and therefore not visible on the core boundary.

### **X-Bus**

The X-Bus is the internal part of the external bus. In general, the X-Bus is placed between the bus controller and the external bus ports. The X-Bus provides a full 24-bit address bus and a 16-bit data bus between the MCU and its X-Bus peripherals (the address width of external bus depends on the selected port configuration). On the X-Bus, one MCU controlled access can be executed every machine cycle. MCU accesses to X-Bus peripherals are (but do not have to be) synchronous accesses, which may be delayed by programmable wait states or with a READY acknowledgement.

The differentiation between accesses to external bus and internal X-Bus is performed within the bus controller by address range detection. For accesses to internal X-Bus peripherals, six different address ranges can be selected, each with its own bus type definition, by programming the necessary [XADRSx \(on Page 206\)](#) and [XBCONx \(on Page 208\)](#) registers. To each address range belongs a chip select signal which also may be shared between more than one X-Bus peripheral. For accesses to external

peripherals up to 5 additional address ranges may be selected using the **ADDRSELx** (on Page 873) and **BUSCONx** (on Page 870) registers with optional chip select outputs.

### **6.2.2.3 External Bus Interface**

To meet the needs of designs where more memory is required than is provided on chip, up to 16 MBytes of external RAM and/or ROM can be connected to the microcontroller via its external bus interface. The integrated Bus Controller (BC) allows the access of external memory and/or peripheral resources in a very flexible way.

The BC can be programmed either to Single Chip Mode when no external memory is required, or to one of two different external memory access modes, which are as follows:

- 16-/18-/20-/24-bit Addresses 16-bit Data Demultiplexed
- 16-/18-/20-/24-bit Addresses 8-bit Data Demultiplexed

Important timing characteristics of the external bus interface (Memory Cycle Time, Memory Tri-State Time, Length of ALE (ALE is an internal signal), Read Write Delay  $\overline{CS}$  and  $\overline{WR}$ ) have been made programmable to allow the use of a wide range of different types of memories. Different address ranges may be accessed with different bus characteristics. Up to 5 external  $\overline{CS}$  signals can be generated to save external glue logic. Access to very slow memories is supported via a special 'Ready' function.

### **6.2.2.4 The On-Chip Peripheral Blocks**

The C166 Family clearly separates peripherals from the core. This structure permits the maximum number of operations to be performed in parallel and allows peripherals to be added or deleted from family members without modifications to the core. These built in peripherals either allow the MCU to interface with the external world or provide functions on-chip that otherwise would have to be added externally in the respective system. Each functional block processes data independently and communicates information over common buses. Individually selected clock signals are generated for each peripheral.

### **Peripheral Interfaces**

The on-chip peripherals generally have two different types of interfaces:

1. To the MCU
2. To external hardware.

Communication between MCU and peripherals is performed through Special Function Registers (SFRs) and interrupts.

**CONFIDENTIAL****System Overview**

Each peripheral contains a set of SFRs, which control the functionality of the peripheral and temporarily store intermediate data results. These SFRs are located either:

- In the standard SFR area (00 FE00<sub>H</sub> - 00 FFFF<sub>H</sub>)
- In the extended ESFR area (00 F000<sub>H</sub> - 00 F1FF<sub>H</sub>).

Each peripheral has an associated set of status flags.

Interrupt requests are generated by the peripherals based on specific events which occur during their operation (for example, operation complete, error, etc.).

For interfacing with external hardware, specific pins of the parallel ports can be used as alternative functions for the on-chip peripherals.

**Peripheral Timing**

Internal operation of the MCU and peripherals is based on the MCU clock ( $f_{CPU}$ ). The clock signal ( $f_{PDBus}$ ) which is gated to the peripherals is independent from the clock signal which feeds the MCU. During the Idle mode the MCU clock is stopped while the peripherals continue their operation. Peripheral SFRs may be accessed by the MCU once per state. When an SFR is written to by software in the same state where it is also to be modified by the peripheral, the software write operation has priority.

For more detailed information on the clocking system refer to **Section 10.4 PLL - CGU - SCCU (on Page 651)**.

**6.2.2.5 OCDS and JTAG**

The On-Chip Debug Support (OCDS) provides facilities to the debugger to emulate resources and assists in application program debugging. The main features are:

- Real time emulation
- Extended trigger capability including: instruction pointer events, data events on address and/or value, external inputs, counters, chaining of events, timers, etc....
- Software break support
- Break and "break before make" (on **IP** events only)
- Simple monitor mode or JTAG based debugging through instruction injection.

The C166S OCDS is controlled by the debugger<sup>1)</sup> through a set of registers accessible from the JTAG interface. The OCDS also receives informations (such as **IP**, data, status) from the core for monitoring the activity and generating triggers. Finally, the OCDS interacts with the core through a break interface to suspend program execution and an injection interface to allow execution of OCDS generated instructions.

<sup>1)</sup> The debugger refers to the tool connected to the emulator, more specifically to the OCDS via the JTAG, which manages the emulation/debugging task.

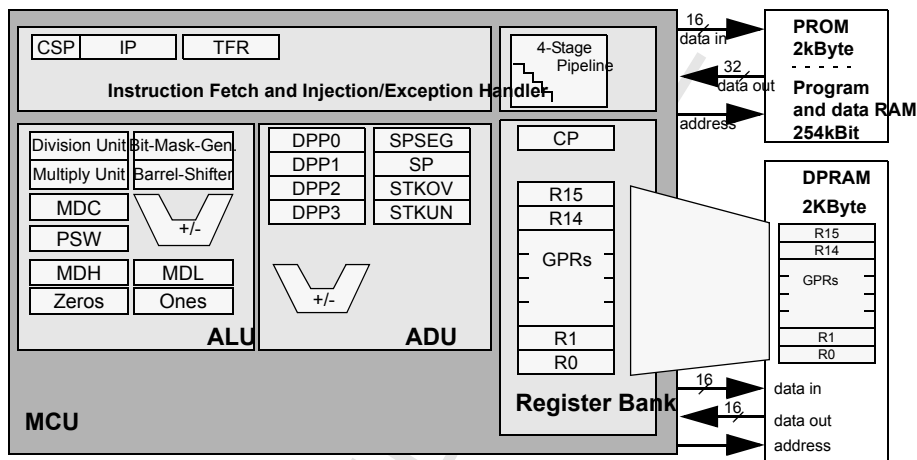
**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

## 6.3 Central Processing Unit, PEC, and Interrupt Controller

The C166S Central Processing Unit (MCU) is the synthesized generation of the C166 core family. It has many powerful enhancements while remaining compatible with the C166 family. The new architecture offers a high-performance MCU, fast and efficient access to different kinds of memories, and efficient integration with peripheral units.

**Figure 6-2 MCU Architecture**



The new core architecture of the C166S results in higher MCU clock frequencies compared to the C166 full custom cores.

C166S has 5 main units. All these units have been optimized to achieve maximum performance and flexibility. The units are:

1. High Performance Instruction Fetch Unit (IFU)
  - High bandwidth fetch interface
  - Instruction FIFO
  - High-performance branch-, call-, and loop-processing with instruction flow prediction
2. Injection/Exception Handler
  - Handling of interrupt requests
  - Handling of hardware failures
3. Instruction PIPEline (IPIP)
  - 4-stage execution pipeline
4. Address and Data Unit (ADU)
  - 16-bit arithmetic unit for address generation

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

**5. Arithmetic and Logic Unit (ALU)**

- 8-bit and 16-bit arithmetic unit
- 16-bit barrel shifter
- Multiplication and division unit
- 8-bit and 16-bit logic unit
- Bit-manipulation unit

**6.3.1 MCU Special Function Registers**

The core MCU requires a set of MCU Special Function Registers (CSFRs) to maintain the system state information, to control system and bus configuration, and to manage code memory segmentation and data memory paging. The MCU also uses the CSFRs to access the General-Purpose Registers (GPRs) and the System Stack, to supply the ALU with register-addressable constants, and to support multiply and divide ALU operations.

The access mechanism for these CSFRs in the MCU core is identical to the access mechanism for any other SFR. Since all SFRs can be controlled by any instruction that is capable of addressing the SFR/CSFR memory space, there is no need for special system control instructions.

However, to ensure proper processor operations, certain restrictions on the user access to some CSFRs must be applied. For example, the Instruction Pointer (**IP**) and Code Segment Pointer (**CSP**) registers cannot be accessed directly, they can only be changed indirectly via branch instructions.

The Program Status Word (PSW), Stack Pointer (**SP**), and Multiply/Divide Control Register (MDC) registers can be modified explicitly by the programmer and implicitly by the MCU during normal instruction processing.

*Note: Any explicit write request (via software) to a (C)SFR supersedes a simultaneous modification by hardware of the same register.*

*Note: All (C)SFRs may be accessed word-wise or byte-wise (some bitwise). Reading bytes from word (C)SFRs is a non-critical operation. Any write operation to a single byte of an (C)SFR clears the non-addressed complementary byte within the specified (C)SFR.*

*Non-implemented (reserved) (C)SFR-Bits cannot be modified, and are always read as a 0. A non-implemented (C)SFR always return a read of FFFF<sub>H</sub>.*

**CONFIDENTIAL****Central Processing Unit, PEC, and Interrupt****Programming Hints****Access to SFRs**

All SFRs reside in a dedicated page of the memory space. The following addressing mechanisms allow to access the (C)SFRs:

- Indirect or direct addressing with **16-bit (mem) addresses** must guarantee that the used data page pointer (DPP0...DPP3) selects data page 3.
- Accesses via the Peripheral Event Controller (PEC) use the **SRCPx** and **DSTPx** pointers instead of the data page pointers.
- **Short 8-bit (reg) addresses** to the standard SFR area do not use the data page pointers but directly access the registers within this 512 Byte area.
- **Short 8-bit (reg) addresses** to the extended **ESFR** area require switching to the 512 Byte extended SFR area. This is done via the EXTension instructions EXTR, EXTP(R), EXTS(R).

**Byte write operations** to word wide SFRs via indirect or direct 16-bit (mem) addressing or byte transfers via the PEC force zeros in the non-addressed byte. Byte write operations via short 8-bit (reg) addressing can only access the low byte of an SFR and force zeros in the high byte. It is therefore recommended to use the bit field instructions (BFLDL and BFLDH) to write to any number of bits in either byte of an SFR without disturbing the non-addressed byte and the un-selected bits.

**Reserved Bits**

Some of the bits which are contained in the C166S's SFRs are marked as Reserved. User software should never write a 1 to the reserved bits. These bits are currently not implemented and may be used in future products to invoke new functions. In this case, the active state for these functions will be 1, and the inactive state will be 0. Therefore writing only 0 to the reserved locations provides portability of the current software to future devices. After read accesses, reserved bits should be ignored or masked out.

**6.3.2 Instruction Fetch and Program Flow Control**

The C166S can fetch on average one 32-bit or two 16-bit instructions via the 32-bit wide Local Memory Bus (LM-Bus) every machine cycle (which equals two clock cycles T1 and T2) to provide a continuous instruction flow. The instructions can be fetched via this new internal LM-Bus from the internal local memories (ROM and SRAM) every clock cycle. A wait state mechanism allows the MCU to adapt to different kind of memories. For example, this mechanism can be used to:

- Access slower memories
- Generate a power ramp up phase for flash modules
- Stall a DRAM access during the refresh cycles.

*Note: Additionally, the LM-Bus provides 16-bit read and write data accesses with and without wait states to the internal local memory. Furthermore, read protection is*

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

*provided by the MCU to protect the internal local memories against illegal data accesses.*

### 6.3.2.1 Branch Target Addressing Modes

The target address and the segment of jump or call instructions can be specified by several addressing modes. The **IP** (instruction pointer) register (see **IP (on Page 107)**) may be updated using relative, absolute, or indirect modes. The **CSP** register can be updated only by using an absolute value. A special mode is provided to address the interrupt and trap jump vector table, which resides in the lowest portion of the code segment 0.

**Table 6-1 Branch Target Addressing Modes**

Mnemonic	Target Address	Target Segment	Valid Address Range
<b>caddr</b>	<b>(IP)</b> = caddr	-	caddr = 0000 <sub>H</sub> ...FFFE <sub>H</sub>
<b>rel</b>	<b>(IP)</b> = <b>(IP)</b> + 2*rel <b>(IP)</b> = <b>(IP)</b> + 2*(rel+1)	- -	rel = 00 <sub>H</sub> ...7F <sub>H</sub> rel = 80 <sub>H</sub> ...FF <sub>H</sub>
<b>[Rw]</b>	<b>(IP)</b> = (Rw)	-	Rw w = 0...15
<b>seg</b>	-	<b>(CSP)</b> = seg	seg = 0...255(3)
<b>#trap7</b>	<b>(IP)</b> = 0000 <sub>H</sub> + 4*trap7	<b>(CSP)</b> = 0000 <sub>H</sub>	trap7 = 00 <sub>H</sub> ...7F <sub>H</sub>

**caddr**: Specifies an absolute 16-bit code address within the current segment. Branches **must not** be taken to odd code addresses. Therefore, the least significant bit of caddr must always contain a 0 or a hardware trap will occur.

**rel**: This mnemonic represents an 8-bit signed word offset address relative to the current **IP** contents, which point to the instruction after the branch instruction. Depending on the offset address range, both forward (rel = 00<sub>H</sub> to 7F<sub>H</sub>) and backward (rel = 80<sub>H</sub> to FF<sub>H</sub>) branches are possible. The branch instruction itself is repeatedly executed, when rel = -1 (FF<sub>H</sub>) for a word-sized branch instruction, or rel = -2 (FE<sub>H</sub>) for a double-word-sized branch instruction.

**[Rw]**: In this case, the 16-bit branch target instruction address is determined indirectly by the contents of a word GPR. In contrast to indirect data addresses, indirectly-specified code addresses are **not** calculated via additional pointer registers (for example, DPP registers). Branches **must not** be taken to odd code addresses. Therefore, the least significant bit of the address pointer GPR must always contain a 0 or a hardware trap would occur.

**seg**: Specifies an absolute code segment number. The C166S supports 256 different code segments, so only the 8 lower bits (respectively) of the 'seg' operand value are used to update the **CSP** register.

**#trap7**: Specifies a particular interrupt or trap number for branching to the corresponding interrupt or trap service routine via a jump vector table. Trap numbers from 00<sub>H</sub> to 7F<sub>H</sub> can be specified to access any double-word code location within the address



**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

range 00 0000<sub>H</sub>-00 01FC<sub>H</sub> in code segment 0 (that is, the interrupt jump vector table).

For the association of trap numbers with the corresponding interrupt or trap sources, please refer to [Section 6.3.3 Interrupt and Exception Execution \(on Page 111\)](#).

### 6.3.2.2 Sequential and Non-Sequential Instruction Flow

Since passing through one pipeline stage takes at least one machine cycle (which equals two clock cycles T<sub>1</sub> and T<sub>2</sub>), any isolated instruction takes at least four machine cycles to be completed. Pipelining, however, allows parallel (that is, simultaneous) processing of up to four instructions. Therefore, most instructions will seem to be processed during one machine cycle as soon as the pipeline has been filled once after reset.

Pipelining increases the average instruction throughput. In this manual, any execution time specification always refers to the average instruction execution time due to pipelined parallel processing.

The execution time of a sequential and non-sequential instruction flow is mainly given by the instruction fetch from different kind of memories (number of wait states).

The following pipeline diagram ([Table 6-2](#)) shows the continuous execution of instruction under the assumption of a fast Local Memory (0/1 wait states).

**Table 6-2 Sequential Instruction Execution (Local Memory, 0/1 Wait States)**

Clock Cycle	T <sub>1</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>2</sub>
<b>FETCH</b>	I <sub>n</sub>		I <sub>n+1</sub>		I <sub>n+2</sub>		I <sub>n+3</sub>		I <sub>n+4</sub>		I <sub>n+6</sub>	
<b>DECODE</b>	I <sub>n-1</sub>		I <sub>n</sub>		I <sub>n+1</sub>		I <sub>n+2</sub>		I <sub>n+3</sub>		I <sub>n+4</sub>	
<b>EXECUTE</b>	I <sub>n-2</sub>		I <sub>n-1</sub>		I <sub>n</sub>		I <sub>n+1</sub>		I <sub>n+2</sub>		I <sub>n+3</sub>	
<b>WRITE BACK</b>	I <sub>n-3</sub>		I <sub>n-2</sub>		I <sub>n-1</sub>		I <sub>n</sub>		I <sub>n+1</sub>		I <sub>n+2</sub>	
<b>Machine Cycle</b>	T <sub>m</sub>		T <sub>m+1</sub>		T <sub>m+2</sub>		T <sub>m+3</sub>		T <sub>m+4</sub>		T <sub>m+5</sub>	

The fetch stage fetches instructions from the Local Memory (LM) via the 32-bit LM-Bus. If 16-bit instructions are fetched from the LM-Bus, instructions can be buffered in the 3-word FIFO. The fetch stage always prefetches instructions. If the buffer is filled with instructions, LM-Bus accesses are stopped until the fetched instructions can be loaded into the buffer again.

[Table 6-3](#) shows the standard unconditional branch (branch taken) instruction pipeline, assuming a fast local memory (0/1 wait states).

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

**Table 6-3      Unconditional Branches (LM-Bus, 0/1 Wait States)**

Clock Cycle	T <sub>1</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>2</sub>
<b>Address</b>						I <sub>a_t</sub>						
<b>Data 32bit</b>						I <sub>d_t</sub>	I <sub>d_t</sub>					
<b>FETCH</b>	I <sub>n</sub>		I <sub>n+1</sub> == branch					I <sub>t</sub>		I <sub>t+1</sub>		I <sub>t+2</sub>
<b>DECODE</b>	I <sub>n-1</sub>		I <sub>n</sub>		I <sub>n+1</sub> == branch		-	I <sub>t</sub>		I <sub>t+1</sub>		
<b>EXECUTE</b>	I <sub>n-2</sub>		I <sub>n-1</sub>		I <sub>n</sub>		I <sub>n+1</sub> == branch		-	I <sub>t</sub>		
<b>WRITE BACK</b>	I <sub>n-3</sub>		I <sub>n-2</sub>		I <sub>n-1</sub>		I <sub>n</sub>		I <sub>n+1</sub> == branch		-	
<b>Machine Cycle</b>	T <sub>m</sub>		T <sub>m+1</sub>		T <sub>m+2</sub>		T <sub>m+3</sub>		T <sub>m+4</sub>		T <sub>m+5</sub>	

In case of a branch to a 32-bit target instruction, which is not aligned to a 32-bit address, one additional machine cycle (T1, T2) is required.

**Table 6-4** shows a standard conditional branch (branch taken) instruction pipeline, assuming a fast local memory (0/1 wait states).

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

**Table 6-4      Conditional branches (LM-Bus, 0/1 Wait States)**

Clock Cycle	T <sub>1</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>2</sub>
Address								I <sub>a_t</sub>				
Data 32bit								I <sub>d_t</sub>	I <sub>d_t</sub>			
FETCH		I <sub>n</sub>		I <sub>n+1</sub> == branch		(I <sub>n+2</sub> )				I <sub>t</sub>		I <sub>t+1</sub>
DECODE	I <sub>n-1</sub>		I <sub>n</sub>		I <sub>n+1</sub> == branch		I <sub>n+1</sub> == branch		-		I <sub>t</sub>	
EXECUTE	I <sub>n-2</sub>		I <sub>n-1</sub>		I <sub>n</sub>		I <sub>n+1</sub> == branch		I <sub>n+1</sub> == branch		-	
WRITE BACK	I <sub>n-3</sub>		I <sub>n-2</sub>		I <sub>n-1</sub>		I <sub>n</sub>		I <sub>n+1</sub> == branch		I <sub>n+1</sub> == branch	
Machine Cycle	T <sub>m</sub>		T <sub>m+1</sub>		T <sub>m+2</sub>		T <sub>m+3</sub>		T <sub>m+4</sub>		T <sub>m+5</sub>	

### Cache Jump Instruction Processing

The C166S incorporates a jump cache to optimize conditional jumps, which are processed repeatedly within a loop. Whenever a jump on cache is taken, the extra time to fetch the branch target instruction can be saved and thus the corresponding cache jump instruction in most cases takes only one (unconditional branch) or two (conditional branch) machine cycles.

This performance is achieved by the following mechanism: Whenever a cache jump instruction passes through the decode stage of the pipeline for the first time (and provided that the jump condition is met), the jump target instruction is fetched as usual, causing a time delay of one machine cycle. In contrast to standard branch instructions, however, the target instruction of a cache jump instruction (JMPA, JMPR, JB, JBC, JNB, JNBS) is additionally stored in the cache after having been fetched.

After each subsequent execution of the same cache jump instruction, the jump target instruction is not fetched from program memory but taken from the cache and immediately injected into the fetch/decode stage of the pipeline (see table below [Table 6-5](#)).

A time-saving jump on cache is always taken after the second and any subsequent occurrences of the same cache jump instruction, unless an instruction that has the fundamental capability of changing the **CSP** register contents (JMPS, CALLS, RETS,

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

TRAP, RETI), or any standard interrupt has been processed during the period of time between two following occurrences of the same cache jump instruction.

**Table 6-5** shows a standard unconditional branch (branch taken and target cached) instruction pipeline, assuming a fast local memory (0/1 wait states).

**Table 6-5      Unconditional cached branches (LM-Bus, 0/1 Wait States)**

Clock Cycle	T <sub>1</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>2</sub>
Address						I <sub>a_t+1</sub>						
Data 32bit						I <sub>d_t+1</sub>	I <sub>d_t+1</sub>					
FETCH	I <sub>n</sub>		I <sub>n+1==</sub> branch			I <sub>t</sub>		I <sub>t+1</sub>		I <sub>t+2</sub>		I <sub>t+3</sub>
DECODE	I <sub>n-1</sub>		I <sub>n</sub>		I <sub>n+1==</sub> branch	I <sub>t</sub>		I <sub>t+1</sub>		I <sub>t+2</sub>		
EXECUTE	I <sub>n-2</sub>		I <sub>n-1</sub>		I <sub>n</sub>	I <sub>n+1==</sub> branch		I <sub>t</sub>		I <sub>t+1</sub>		
WRITE BACK	I <sub>n-3</sub>		I <sub>n-2</sub>		I <sub>n-1</sub>	I <sub>n</sub>		I <sub>n+1==</sub> branch		I <sub>t</sub>		
Machine Cycle	T <sub>m</sub>		T <sub>m+1</sub>		T <sub>m+2</sub>		T <sub>m+3</sub>		T <sub>m+4</sub>		T <sub>m+5</sub>	

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

**Table 6-6** shows a standard conditional branch (branch taken and target cached) instruction pipeline, assuming a fast local memory (0/1 waitstates).

**Table 6-6 Conditional cached branches (LM-Bus, 0/1 waitstate)**

Clock Cycle	T <sub>1</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>2</sub>
Address								I <sub>a_t+</sub> 1				
Data 32bit								I <sub>d_t+</sub> 1	I <sub>d_t+</sub> 1			
FETCH		I <sub>n</sub>		I <sub>n+1</sub> == bran ch		(I <sub>n+2</sub> )		I <sub>t</sub>		I <sub>t+1</sub>		I <sub>t+2</sub>
DECODE	I <sub>n-1</sub>		I <sub>n</sub>		I <sub>n+1</sub> == bran ch		I <sub>n+1</sub> == bran ch		I <sub>t</sub>		I <sub>t+1</sub>	
EXECUTE	I <sub>n-2</sub>		I <sub>n-1</sub>		I <sub>n</sub>		I <sub>n+1</sub> == bran ch		I <sub>n+1</sub> == bran ch		I <sub>t</sub>	
WRITE BACK	I <sub>n-3</sub>		I <sub>n-2</sub>		I <sub>n-1</sub>		I <sub>n</sub>		I <sub>n+1</sub> == bran ch		I <sub>n+1</sub> == branch	
Machine Cycle	T <sub>m</sub>		T <sub>m+1</sub>		T <sub>m+2</sub>		T <sub>m+3</sub>		T <sub>m+4</sub>		T <sub>m+5</sub>	

### 6.3.2.3 ATOMIC and EXTENDED Instructions

ATOMIC and EXTENDED instructions (ATOMIC, EXTR, EXTP, EXTS, EXTPR, EXTSR) disable the standard and PEC interrupts and class A traps until the completion of the next sequence of instructions. The number of instructions in the sequence may vary from 1 to 4. The instruction number is coded in the 2-bit constant field #irang2 and takes values from 0 to 3. The EXTENDED instructions additionally change the addressing mechanism during this sequence (see instruction description).

ATOMIC and EXTENDED instructions become active immediately, so no additional NOP instructions are required. All instructions requiring multiple cycles or hold states for execution are considered as one instruction. ATOMIC and EXTENDED instructions can be used with any instruction type.

CONFIDENTIAL

Central Processing Unit, PEC, and Interrupt

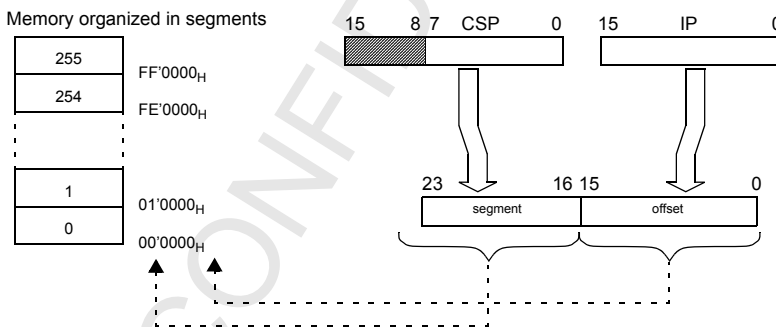
*Note: If a class B trap interrupt occurs during an ATOMIC or EXTENDED sequence, then the sequence is terminated, an interrupt lock is removed, and the standard condition is restored before the trap routine is executed. The remaining instructions of the terminated sequence that are executed after returning from the trap routine will run under standard conditions.*

*Note: Certain precautions are required when using nested ATOMIC and EXTENDED instructions. There is only one counter to control the length of the sequence, i.e., issuing an ATOMIC or EXTENDED instruction within a sequence will reload the counter with the value of the new instruction.*

### 6.3.2.4 Code Addressing via Code Segment and Instruction Pointer

The C166S provides a total addressable memory space of 16 MBytes. This address space is arranged as 256 segments of 64kBytes each. A dedicated 24-bit code address pointer is used to access the memories for instruction fetches. This pointer has two parts: An 8-bit Code Segment Pointer (**CSP**), and a 16-bit offset Instruction Pointer (**IP**). The concatenation of the **CSP** and **IP** results directly in a correct 24-bit physical memory address.

Figure 6-3 Addressing via the Code Segment and Instruction Pointers



#### 6.3.2.4.1 The Instruction Pointer IP

This register determines the 16-bit intrasegment address of the currently fetched instruction within the code segment selected by the **CSP** register. The **IP** register is not mapped into the C166S address space, and thus it is not directly accessible by the programmer. It can be read by a special debug register.

The **IP** can be modified indirectly by return instructions via the stack. The **IP** register is updated implicitly by the C166S for branch instructions and after instruction fetch operations.

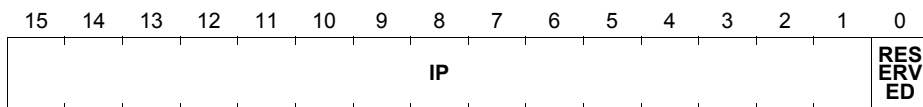
**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

**IP**

**Instruction Pointer**

**Reset value: 0000<sub>H</sub>**



Field	Bits	Type	Description
IP	15:1	rwh	Specifies the intrasegment offset from which the current instruction is to be fetched; <b>IP</b> refers to the current segment <SEGNR>.  <i>Note: <b>IP</b> is always word-aligned.</i>
RESERVED	0	r	Reserved; these bits must be left at their reset values.

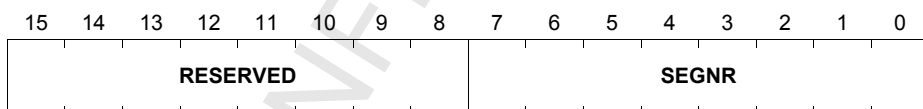
#### 6.3.2.4.2 The Code Segment Pointer CSP

This non-bit-addressable register selects the code segment being used at run-time to access instructions. The lower 8 bits of the register **CSP** select one of up 256 segments of 64kBytes each, while the higher 8 bits are reserved for future use.

**CSP**

**Code Segment Pointer**

**Reset value: 0000<sup>1)</sup><sub>H</sub>**



<sup>1)</sup> The reset value of the bitfield segnr[1:0] is product-specific. With an alternate boot mode feature, the code execution can be started at different segments after reset.

Field	Bits	Type	Description
SEGNR	7:0	rwh	Specifies the code segment where the current instruction is to be fetched
RESERVED	15:8	r	Reserved; these bits must be left at their reset values.

The actual code memory address is generated by direct extension of the 16-bit contents of the **IP** register by the lower byte of the **CSP** register, as shown in **Figure 6-3 (on page 106)**.

There are two modes: Segmented and non-segmented. The mode is selected with **SYSCON.SGTDIS**. After reset, the segmented mode is selected.

### 6.3.2.5 System Configuration Register SYSCON

This register is used to configure the C166S. It is bit-addressable and provides general system configuration and control functions. The reset value of register **SYSCON** depends on the state of the configuration inputs during reset.

#### Segmented Mode

The **CSP** register can be only read and may not be written by data operations. The **CSP** is modified either directly by the JMPS and CALLS instructions, or indirectly via the stack by the RETS and RETI instructions. Upon the acceptance of an interrupt or the execution of a software TRAP instruction, the **CSP** register is automatically loaded with the segment address of the vector location.

#### Non-Segmented Mode

In the non-segmented mode, the **CSP** is fixed to segment 0. It is no longer possible to modify the **CSP** either directly by the JMPS or CALLS instructions, or indirectly via the stack by the RETS (RETI) instruction. For non-segmented memory mode, the contents of this register are not significant, because all code accesses are restricted automatically to segment 0.



**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

## SYSCON

### System Control Register

Reset value: 0XX0<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STKSZ			ROM S1	SGT DIS	ROM EN	BYT DIS	RES ERV ED	WR CFG	CS CFG	RESERVED			XPE N	VISI- BLE	RES ERV ED

Field	Bits	Type	Description
<b>VISIBLE</b>	1	rw	<b>Visible Mode Control</b> 0: Accesses to X-Bus peripherals are done internally 1: X-Bus peripheral accesses are made visible on the external pins Reset value is 0.
<b>XPEN</b>	2	rw	<b>X-Bus Peripheral Enable</b> 0: Disables accesses to the on-chip X-Peripherals and their functions 1: Enables the on-chip X-Peripherals, which can be accessed Reset value is 0.
<b>CSCFG</b>	6	rw	<b>Chip Select Configuration Control</b> 0: Latched $\overline{CS}$ mode (normal $\overline{CS}$ ). The $\overline{CS}$ signals are latched internally and driven to the (enabled) port pins synchronously. 1: Unlatched $\overline{CS}$ mode (early $\overline{CS}$ ). The $\overline{CS}$ signals are directly derived from the address and driven to the (enabled) port pins. Reset value is 0.
<b>WRCFG</b>	7	rwh	<b>Write Configuration Control</b> 0: Pins $\overline{WR}$ and $\overline{BHE}$ retain their normal function 1: Pin $\overline{WR}$ acts as $\overline{WRL}$ , pin $\overline{BHE}$ acts as $\overline{WRH}$ <b>Note: This bit is hard-wired to 1.</b>

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

Field	Bits	Type	Description
<b>BYTDIS</b>	9	rwh	<b>Disable/Enable Control for Pin BHE</b> (Set according to data bus width.) 0: Pin <u>BHE</u> enabled 1: Pin <u>BHE</u> disabled, pin may be used for general purpose I/O  The reset value is set according to the value on pin MON1 if and only if pin MON2 = 1 (refer to <a href="#">Section 6.7.3.2 Internal or External Boot Configuration (on Page 240)</a> ): MON1 MON2 -> <b>BYTDIS</b> Reset Value 0 0 -> 1 0 1 -> MON1 1 0 -> 1 1 1 -> MON1
<b>ROMEN</b>	10	rwh	<b>MCU Configuration: Internal ROM Enable</b> 0 Internal Local Memory disabled: accesses to the Local Memory area use the external bus 1 Internal Local Memory enabled After reset the bit is set to the <b>inverted</b> value of the pin MON2.
<b>SGTDIS</b>	11	rw	<b>MCU Configuration: Segmentation Disable/Enable Control</b> 0 Segmentation enabled ( <b>CSP</b> is saved/restored during interrupt entry/exit) 1 Segmentation disabled (Only <b>IP</b> is saved/restored) Reset value is 0.
<b>ROMS1</b>	12	rw	<b>MCU Configuration: Internal Local Memory Mapping</b> 0 Internal Local Memory area mapped to segment 0 (00 0000 <sub>H</sub> ...00 7FFF <sub>H</sub> ) 1 Internal Local Memory area mapped to segment 1 (01 0000 <sub>H</sub> ...01 7FFF <sub>H</sub> )
<b>STKSZ</b>	15:13	rw	<b>MCU Configuration: System Stack Size</b> Selects the size of the system stack (in the internal DPRAM) from 32 to 1024 words.
<b>RESERVED</b>	0, 5:3, 8	r	Reserved, these bits must be left at their reset values.

*Note: Register **SYSCON** cannot be changed after execution of the EINIT instruction.*

**CONFIDENTIAL****Central Processing Unit, PEC, and Interrupt**

### **6.3.3 Interrupt and Exception Execution**

*Note: Refer to the [Figure 6.3.3.11 Interrupt List \(on Page 142\)](#) for additional information.*

An Interrupt and Exception Handler is responsible for managing all system and core exceptions. There are four different kinds of exceptions that are executed in a similar way:

- Interrupts generated by the InTerrupt Controller (ITC)
- Software traps caused by the TRAP instruction
- Hardware traps issued by faults or specific system states
- Interrupts generated by the Peripheral Event Controller (PEC).

#### **Normal Interrupt Processing**

The MCU temporarily suspends the current program execution and branches to an Interrupt Service Routine (ISR) to service an interrupt-requesting device. The current program status [Instruction Pointer (**IP**), Processor Status Word (PSW), and in segmentation mode, the Code Segment Pointer (**CSP**)] is saved on the internal system stack. A prioritization scheme with 16 priority levels and with 8 sub-levels (8 group levels) specifies the order of multiple interrupt-request handling. The maximum number of interrupt requests supported by the core architecture is 112 (configured in steps of 16). To find the number of interrupts used in the PMB7870, refer to [Table 6.3.3.11 Interrupt List \(on Page 142\)](#). The lowest priority level is reserved for the MCU and cannot be used for interrupts.

#### **Software and Hardware Traps**

Trap functions are activated in response to special conditions that occur during the execution of instructions. A trap can also be caused externally by the Non-Maskable Interrupt (**NMI**) pin. Several hardware trap functions are provided for handling erroneous conditions and exceptions that arise during the program execution. Hardware traps always have highest priority and cause immediate system reaction. The software trap function is invoked by the TRAP instruction, which generates a software interrupt for a specified interrupt vector. For all types of traps, the current program status is saved in the system stack.

#### **Interrupt Processing via the Peripheral Event Controller**

A faster alternative to normal interrupt processing is servicing an interrupt requesting device by the C166S integrated PEC. Triggered by an interrupt request, the PEC performs a single-word or byte data transfer between any two memory locations through one of 16 programmable PEC service channels. During a PEC transfer, the normal program execution of the MCU is halted. No internal program status information needs

CONFIDENTIAL

Central Processing Unit, PEC, and Interrupt

to be saved. The same prioritization scheme is used for PEC service as for normal interrupt processing.

### 6.3.3.1 Interrupt System Structure

The C166S architecture provides up to 112 separate interrupt nodes that may be assigned to 128 arbitration priority levels with 16 interrupt priority groups and 8 priorities inside each group. To find the number of interrupts used in the PMB7870, refer to [Table 6.3.3.11 Interrupt List](#). To support modular and consistent software design techniques, most sources of an interrupt or PEC request are supplied with a separate interrupt control register and interrupt vector. The control register contains an interrupt request flag, Interrupt Enable (IE) bit, and interrupt priority of the associated source. Each source request is activated by one specific event, depending on the selected operating mode of the respective device. In some cases, the multi-source interrupt nodes are incorporated for efficient use of system resources. These nodes can be activated by several source requests.

The C166S provides a vectored interrupt system. This system reserves specific vector locations in the memory space for the reset, trap, and interrupt service functions. Whenever a request occurs, the MCU branches to the location that is associated with the respective interrupt source. The reserved vector locations build a jump table in the C166S's address space.

The arbitration winner is sent to the MCU together with its priority level and action request. The MCU triggers the corresponding action, which depends on the required functionality (normal interrupt, PEC etc.) of the arbitration winner.

An action request will be accepted by the MCU if the requesting source has a higher priority than the current MCU priority level, and if interrupts are globally enabled. If the requesting source has a lower (or equal) interrupt level priority, then the requested interrupt stays pending.

### 6.3.3.2 Interrupt Arbitration

The C166S interrupt arbitration system can handle interrupt requests from up to 112 sources. To find the number of interrupts used in the PMB7870, refer to [Table 6.3.3.11 Interrupt List](#). Interrupt requests may be triggered either by the C166S peripherals or by external inputs. The "End of PEC" interrupt for supporting enhanced PEC functionality is connected internally to one of the interrupt request lines, refer to [Section 6.3.3.6.7 Programmable End of PEC Interrupt Level \(on Page 137\)](#).

The arbitration process starts by an enabled interrupt request and stays active for as long as interrupt request is pending. If nothing is pending then the arbitration logic switches to the idle state to save power.

Each interrupt request line is controlled by its interrupt control register **xxIC**. ('xx' stands for the mnemonic of the respective interrupt source). An interrupt request event sets the

CONFIDENTIAL

Central Processing Unit, PEC, and Interrupt

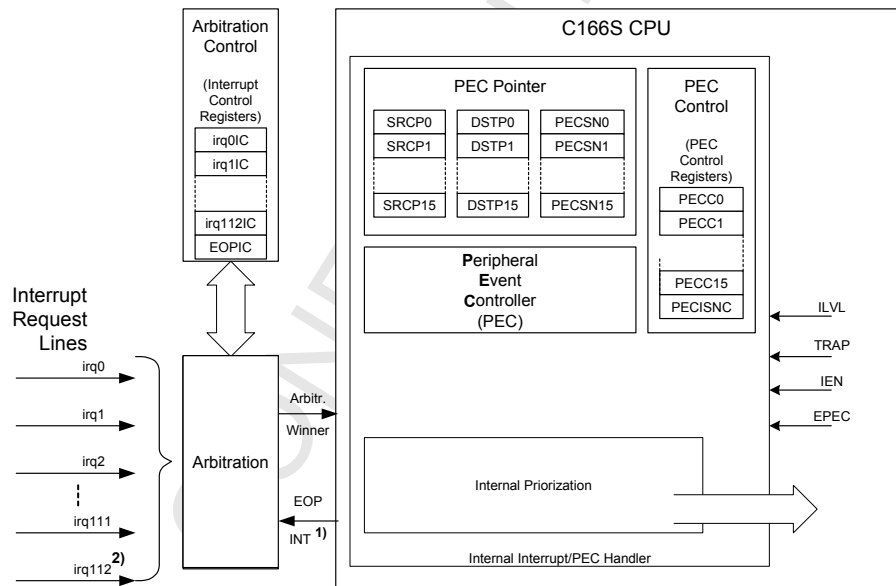
interrupt request flag to 1 in the corresponding interrupt control register (bit **xxIC.xxIR**). The interrupt request can also be triggered by the software if the program sets the respective interrupt request bit. This feature is used by operating systems.

If the request bit has been set and this interrupt request is enabled by the IE bit of the same control register (bit **xxIC.xxIE**), then an arbitration cycle starts on the next clock cycle. However, if an arbitration cycle is currently in progress, the new interrupt request will be delayed till the next arbitration cycle. If an interrupt request (or PEC request) is accepted by the core, the respective interrupt request flag is cleared automatically.

All interrupt requests that are pending at the beginning of a new arbitration cycle are considered simultaneously. Within the arbitration cycle, the arbitration is independent of the actual request time.

C166S uses a two-stage interrupt prioritization scheme for interrupt arbitration, as shown in **Figure 6-4**.

**Figure 6-4 Interrupt Arbitration**



<sup>1)</sup> End of PEC Interrupt (EOPINT) is connected to one of the interrupt request lines. Therefore, only up to 111 interrupt lines are available for peripheral request handling.

<sup>2)</sup> The number of interrupt lines used can be seen in the chapter "Microcontroller Interrupt Vector List"

The first arbitration stage compares up to 128 priority levels of interrupt request lines. The priority level of each request consists of Interrupt priority LeVeL (**xxIC.ILVL**) and Group priority LeVeL (**xxIC.GLVL**). An interrupt priority level is programmed for each

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

interrupt request line by the respective 4-bit bitfield **xxIC.ILVL**. The group priority level is programmed for each interrupt request line by the 2-bit bitfield **xxIC.GLVL** and the group extension bit **xxIC.xxGP**.

*Note: All interrupt request sources that are enabled and programmed to the same **xxIC.ILVL** must have different group priority levels. Otherwise, an incorrect interrupt vector may be generated.*

In the second arbitration stage, the priority level of the first-stage winner is compared with the priority of the current MCU task. An action request will be accepted by the MCU if the requesting source has a higher priority level than the current MCU priority level (bits ILVL of the PSW register), and if interrupts are enabled globally by the global IEN flag in PSW. The MCU denies all requests in case of a cleared IEN flag. If the requester has a lower or equal priority level than current MCU task, the request stays pending.

*Note: Priority level 0000<sub>B</sub> is the default level of the MCU. Therefore, a request on ILVL 0000<sub>B</sub> will be arbitrated, but the MCU will never accept an action request on this level. However, every enabled interrupt request (including all denied interrupt requests - also priority level 0000<sub>B</sub> requests) triggers a MCU wake-up from idle state independent of the setting of the global interrupt enable bit **PSW.IEN**.*

*Note: The first 16 trap numbers are reserved for the MCU traps. The first usable interrupt trap number starts with 10<sub>H</sub>. Therefore, the maximum possible number of interrupt nodes is limited to 112. For the number of interrupts used in the PMB7870, refer to [Table 6.3.3.11 Interrupt List](#).*

### **6.3.3.3 Interrupt Control Registers**

*Note: Refer to [Table 6.3.3.11 Interrupt List](#)*

All interrupt control registers are organized identically. The lower 9 bits of an interrupt control register contain the complete interrupt status information of the associated source, which is required during one round of prioritization, the upper 7 bits of the respective register are reserved. All interrupt control registers are bit-addressable and all bits can be read or written via software. This allows each interrupt source to be programmed or modified with just one instruction. When accessing interrupt control registers through instructions which operate on word data types, their upper 8 bits (15...8) will return zeros, when read, and will discard written data. Zeros should always be written to these bit positions.

The layout of the Interrupt Control registers shown below applies to each xxIC register, where xx stands for the mnemonic for the respective source

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

**xxIC**

**Interrupt Control Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED							xxG P	xxIR	xxIE	ILVL			GLVL		

Field	Bits	Type	Description
GLVL	1:0	rw	<b>Group Priority Level</b> Defines the internal order for simultaneous requests of the same priority. 3 <sub>H</sub> Highest priority level ... 0 <sub>H</sub> Lowest priority level
ILVL	5:2	rw	<b>Interrupt Priority Level</b> F <sub>H</sub> Highest priority level ... 0 <sub>H</sub> Lowest priority level
xxIE	6	rw	<b>Interrupt Enable Control Bit</b> (individually enables/disables a specific source) 0 Interrupt request is disabled 1 Interrupt request is enabled
xxIR <sup>1)</sup>	7	rwh	<b>Interrupt Request Flag</b> 0 No request pending 1 This source has raised an interrupt request.
xxGP	8	rw	<b>Group Priority Extension</b> Defines the value of high-order group level bit
RESERVED	15:9	r	Reserved; these bits must be left at their reset value.

<sup>1)</sup> Bit xxIR supports bit-protection

The arbitration scheme allows nesting of up to 15 ISRs of different priority levels (level 0 cannot be used; see note above).

*Note: When no interrupt request is active, arbitration is stopped to reduce power consumption.*

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

**Interrupt Node Sharing**

The interrupt controller of the E-GOLDRadio can be configured to directly control up to 80 different sources. The number of sources directly controlled can be seen in

**Table 6.3.3.11 Interrupt List**. Where there is a need for a greater number of interrupt sources to be managed, or in some cases to ensure compatibility to earlier designs, a group of interrupt requests may share the same interrupt node. In this case, all the sources on the same node share the priority level defined by the corresponding Interrupt Control register xxIC and may be globally enabled/disabled by the IE bit of this register.

Arbitration between sources connected to the same node must be performed by the interrupt handler associated to this node. For low rate requests, the software overhead is not critical. To allow arbitration in such cases, a interrupt subnode control register is provided outside the main interrupt controller with at least a request flag and an enable control bit for each source. An example of such a register is given by the PECISNC Channel Link Interrupt SubNode register (**PECISNC (on Page 139)**).

**6.3.3.4 Interrupt Control Functions in the Processor Status Word**

**PSW** is divided functionally into 2 parts. The lower byte of the **PSW** represents the arithmetic status of the MCU; the upper byte of the **PSW** controls the interrupt system of the C166S.

**PSW**

**Processor Status Word**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ILVL				IEN	S1	RESERVED			USR 0	MUL IP	E	Z	V	C	N

Field	Bits	Type	Description
ILVL	15:12	rwh	<b>MCU Priority Level</b> 0 <sub>H</sub> Lowest Priority ... F <sub>H</sub> Highest Priority
IEN	11	rw	<b>Interrupt/PEC Enable Flag (global)</b> 0 Interrupt/PEC requests are disabled 1 Interrupt/PEC requests are enabled

*Note: For a summary of the other parts of the **PSW** register, refer to **Section 6.3.6.6 Processor Status Word Register (PSW) (on Page 176)**.*

- **MCU Priority Level (ILVL)** defines the current level for the MCU operation, this bit field gives the priority level of the routine currently being executed. When the MCU



**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

enters an ISR, this bitfield is set to the priority level of the request that is being serviced. The previous PSW is saved in the system stack before entering the ISR. To be serviced, any interrupt request must have a higher priority level than the current MCU priority level. Any request of the same or a lower level will not be acknowledged. The current MCU priority level may be adjusted via software to select interrupt request sources that can be serviced.

PEC transfers do not really interrupt the MCU, but rather “steal” some MCU cycles, so PEC services do not influence the **PSW.ILVL** field.

Hardware traps set the MCU level to the maximum priority (15). Therefore, no interrupt or PEC requests will be acknowledged while an exception trap service routine is executed.

The TRAP instruction does not change the MCU level, so software trap service routines may be interrupted by higher-level requests.

- **Interrupt Enable Flag (IEN)** globally enables or disables interrupts and PEC operations. When **PSW.IEN** is cleared, no new interrupt requests are accepted by the MCU after **PSW.IEN** was set to 0. However, the requests that have already entered the pipeline will be completed. If **PSW.IEN** is set to 1, all interrupt sources are globally enabled.

*Note: To generate requests, interrupt sources must be also enabled by the **IEN** bits in their associated control registers.*

*Note: Traps are non-maskable and, therefore, they are not controlled by the **PSW.IEN** bit.*

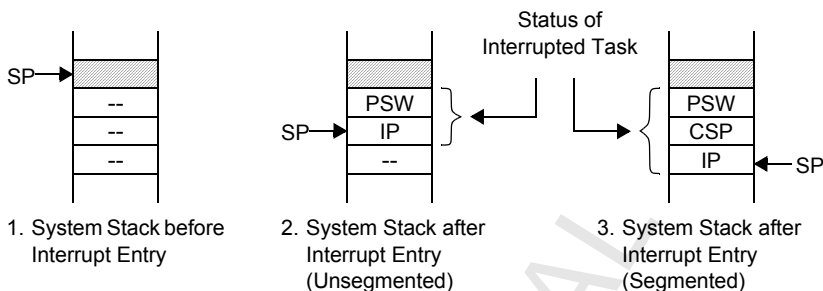
#### **6.3.3.4.1 Saving the Status during Interrupt Service**

Before an operating system can actually service a task switch request or interrupt, the MCU must save the current task status. The C166S saves the MCU status (**PSW**, the Processor Status Word) along with the return address in the system stack. The return address defines the point where the execution of the interrupted task is to be resumed after returning from the service routine. This return address is specified by the **IP** (Instruction Pointer) and, in the case of a segmented memory mode, also by the **CSP** (Code Segment Pointer). **SYSCON.SGTDIS** defines which mode is used and, therefore, controls how the return address is stored.

In the case of non-segmented mode, the system stack stores the **PSW** first and then the **IP**. In the segmented mode, **PSW** is followed by **CSP** and the **IP**. This order optimizes the use of the system stack if segmentation is disabled.

The MCU priority field (**PSW.ILVL**) is updated with the priority of the interrupt request that is to be serviced, so the MCU now executes on the new level.

Figure 6-5 Task Status Saved on System Stack.



After accepting an interrupt request, the C166S sends an acknowledgement to the ITC (InTerrupt Controller) that the requested interrupt is being serviced. The vector associated with the requesting source is loaded into the **IP** and **CSP**, and the first instruction of the service routine is fetched. All other MCU resources such as data page pointers and the context pointer are not affected.

When the MCU returns from the ISR [RETurn from Interrupt (RETI) is executed], the status information is "popped" from the system stack in reverse order. The status information contents depend on the **SYSCON.SGTDIS** value.

**CONFIDENTIAL****Central Processing Unit, PEC, and Interrupt****6.3.3.4.2 Context Switching**

An ISR usually saves all the registers it uses on the stack, and restores them before returning. The more registers a routine uses, the more time is wasted by saving and restoring.

The C166S makes it possible to switch the complete register bank of MCU registers (GPRs) with a single instruction, so the service routine executes within its own separate context. The instruction “SCXT CP, #New\_Bank” pushes the contents of the Context Pointer (**CP**) into the system stack and loads the **CP** with the immediate value “New\_Bank”. The new **CP** value sets a new register bank. The service routine may now use its own registers. This register bank is preserved when the service routine is terminated, that is, its contents are available for the next call. Before returning (RETI), the previous **CP** is simply popped from the system stack, which returns the registers to the original register bank.

*Note: Resources that are used by the interrupting program must eventually be saved. Important are often the Data Page Pointers (DPP) and the registers of the multiply and divide unit.*

*Note: The first instruction following the SCXT CP,... Instruction must not use a General Purpose register (GPR).*

**6.3.3.5 Traps****6.3.3.5.1 Software Traps**

The TRAP instruction is used to cause a software call to an ISR. The trap number that is specified in the operand field of the trap instruction determines which vector location of the vector table will be used.

The TRAP instruction's effect is similar to that of an interrupt request that uses the same vector. **PSW**, **CSP** (in segmentation mode), and **IP** are pushed into the system stack and then a jump is taken to the specified vector location. When a software trap is executed, the **CSP** for the trap service routine is loaded with segment address 0. No Interrupt Request flags are affected by the TRAP instruction. The ISR called by a TRAP instruction must be terminated with a RETI instruction to ensure correct operation.

*Note: The MCU priority level is not modified by the TRAP instruction, so the service routine is executed with the same priority level as the interrupt task. Therefore, the service routine entered by the TRAP instruction can be interrupted by other traps or by higher priority interrupts, other than when triggered by a real hardware event.*

**CONFIDENTIAL****Central Processing Unit, PEC, and Interrupt****6.3.3.5.2 Hardware Traps**

Hardware traps are issued by faults or specific system states that occur during runtime (not identified at assembly time). The C166S distinguishes eight different hardware trap functions. When a hardware trap condition has been detected, the MCU branches to the trap vector location for the respective trap condition. The instruction that caused the trap event is either completed or canceled before the trap-handling routine is entered.

Hardware traps are not-maskable and always have a higher priority than any other MCU task. If several hardware trap conditions are detected within the same machine cycle, the highest-priority trap is serviced. In the case of a hardware trap, the injection unit injects a TRAP instruction into the pipeline. The TRAP instruction performs the following actions:

- Push **PSW**, **CSP** (in segmented mode) and **IP** onto the system stack
- Set MCU level in the **PSW** register to the highest possible priority level, which disables all interrupts
- Branch to the trap vector location specified by the trap number of the trap condition

The eight hardware functions of the C166S are divided in two Classes.

Class A traps are:

- External NMIs
- Stack overflow
- Stack underflow
- Software Break.

These traps share the same trap priority, but have an individual vector address.

Class B traps are:

- Undefined opcode
- Protection fault
- Illegal word operand access
- Illegal instruction access
- Illegal external bus access.

The Class B traps share the same interrupt node and interrupt vector. The bit-addressable Trap Flag Register (**TFR**) allows a trap service routine to identify the trap that caused the exception.

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

## Trap Flag Register TFR

Each trap function is indicated by a separate request flag. When a hardware trap occurs, the corresponding request flag in register TFR is set to 1.

### TFR

#### Trap Flag Register

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NMI	STK OF	STK UF	DEB TRAP	RESERVED			UND OPC	RESERVED			PRT FLT	ILL OPA	ILL INA	ILL BUS	

Field	Bits	Type	Description
ILLBUS	0	rwh	<b>ILLegal External BUS Access</b> 0 No illegal external bus access detected 1 An external access has been attempted with no bus defined.
ILLINA	1	rwh	<b>ILLegal INstruction Access</b> 0 No illegal instruction access detected 1 A branch to an odd address has been attempt.
ILLOPA	2	rwh	<b>ILLegal word OPerand Access</b> 0 No illegal word operand access event detected 1 Illegal word operand access event detected
PRTFLT	3	rwh	<b>PRoTection FauLT</b> 0 No protection fault event detected 1 Protection fault event detected
UNDOPC	7	rwh	<b>UNDeFined OPCode</b> 0 No undefined opcode event detected 1 Undefined opcode event detected
DEBTRAP	12	rwh	<b>DEBug TRAP Flag</b> 0 No debug trap event detected 1 Debug trap event detected
STKUF	13	rwh	<b>STAcK UnderFlow Flag</b> 0 No stack underflow event detected 1 Stack underflow event detected
STKOF	14	rwh	<b>STAcK OverFlow Flag</b> 0 No stack overflow event detected 1 Stack overflow event detected

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

Field	Bits	Type	Description
<b>NMI</b>	15	rwh	<b>Non-Maskable Interrupt Flag</b> 0 No non-maskable interrupt detected 1 Non-maskable interrupt detected
<b>RESERVED</b>	11:8, 6:4	r	Reserved, these bits must be left at their reset values.

*Note: These bits support bit protection.*

*Note: The trap service routine must clear the respective trap flag. Otherwise, a new trap is requested after exiting the service routine. Setting a trap request flag by software causes the same effects as if it had been set by hardware.*

The reset functions (hardware, software, watchdog) may be also regarded as a type of trap. Reset functions have the highest trap priority (trap priority IV). The Debug trap has the second-highest trap priority (trap priority III), followed by the third-highest trap priority traps, Class A traps (trap priority II), and then by Class B traps (trap priority I). So the Debug trap can interrupt a Class A and B trap and a Class A trap can interrupt a Class B trap. The Debug trap is a special kind of interrupt-service channel for debug purposes whose priority lies between the Class A trap and the reset function. This allows the debugger to interrupt hardware traps and hardware interrupts

**Table 6-7 Hardware Traps**

Exception Condition	Trap Flag	Trap Vector	Trap Number	Trap Priority
Reset Functions: Hardware Reset Software Reset Watchdog Timer Overflow		RESET RESET RESET	00 <sub>H</sub> 00 <sub>H</sub> 00 <sub>H</sub>	IV IV IV
Debug Trap	DEBUG	DEBTRAP	08 <sub>H</sub>	III
Class A Hardware Traps: Non-Maskable Interrupt STack OverFlow STack UnderFlow	NMI STKOF STKUF	NMITRAP STOTRAP STUTRAP	02 <sub>H</sub> 04 <sub>H</sub> 06 <sub>H</sub>	II.3 II.2 II.1
Class B Hardware Traps: UNDefined OPCode PRoTectiOn FauLT ILLegal word Operand Access ILLegal INstruction Access ILLegal external BUS access	UNDOPC PRTFLT ILLOPA ILLINA ILLBUS	BTRAP BTRAP BTRAP BTRAP BTRAP	0A <sub>H</sub> 0A <sub>H</sub> 0A <sub>H</sub> 0A <sub>H</sub> 0A <sub>H</sub>	I I I I I

**CONFIDENTIAL****Central Processing Unit, PEC, and Interrupt****Class A Trap**

Class A traps are generated by the high-priority system NMI or by special MCU events such as a software break, or a stack overflow or underflow event. Class A traps are not used to indicate hardware failures. After a Class A event, a dedicated service routine is called to react to the events. Each Class A trap has its own vector location in the vector table. After finishing the service routine, the remainder of the instruction flow must be executed correctly. This explains why Class A traps cannot interrupt atomic/extend sequences.

In case of an atomic/extend sequence, the execution continues until sequence completion. Upon completion, the **IP** of the instruction following the last executed one is pushed onto the stack.

If more than one Class A trap occurs at a same time, they are prioritized internally. The NMI trap has the highest priority, and the stack underflow trap has the lowest.

*Note: When two different Class A traps occur simultaneously, both trap flags are set. The trap with the higher priority is executed. After return from the service routine, the **IP** is popped from the stack and immediately pushed again because of the other pending Class A trap (unless the second trap flag in **TFR** has been cleared by the first trap service routine).*

**External NMI Trap (NMI)**

Whenever a high-to-low transition on the dedicated  $\overline{\text{NMI}}$  is detected, the NMI flag in register TFR is set, and the MCU will enter the NMI trap routine. The **IP** value pushed on the system stack is the address of the instruction following the one after which normal processing was interrupted by the NMI trap.

*Note: The  $\overline{\text{NMI}}$  is sampled with every MCU clock cycle to detect transitions.*

**STack OverFlow Trap (STKOF)**

Whenever the stack pointer (**SP**) is decremented to a value less than the value in the stack overflow register **STKOV**, the STKOF flag in register TFR is set and the MCU will enter the stack overflow trap routine. Which **IP** value will be pushed onto the system stack depends on which operation caused the decrement of the **SP**. When an implicit decrement of the **SP** is made through a push or all instruction, or upon interrupt or trap entry, the **IP** value pushed is the address of the following instruction. When the **SP** is decremented by a subtract instruction, the **IP** value pushed represents the address of the instruction after the instruction following the subtract instruction.

For recovery from stack overflow, there must be enough excess space on the stack for saving the current system state (**PSW**; **IP**; and, in segmented mode, the **CSP**) twice. Otherwise, a system reset should be generated.

CONFIDENTIAL

Central Processing Unit, PEC, and Interrupt

### STack UnderFlow Trap (STKUF)

Whenever the stack pointer (**SP**) is incremented to a value greater than the value in the stack underflow register **STKUN**, the **TFR.STKUF** flag is set and the MCU enters the stack underflow trap routine. Again, which **IP** value is pushed onto the system stack depends on which operation caused the increment of the **SP**. When an implicit increment of the **SP** is made through a POP or return instruction, the **IP** value pushed is the address of the following instruction. When the **SP** is incremented by an add instruction, the pushed **IP** value represents the address of the instruction after the instruction following the add instruction.

### Class B Trap

Class B traps are generated by unrecoverable hardware failures. In case of hardware failure, the MCU must immediately start a failure service routine. Class B traps can interrupt an atomic/extend sequence. **After finishing a Class B service routine, the interrupted instruction flow cannot be restored.**

*Note: If a Class A trap and a Class B occur simultaneously, both trap flags are set. If this occurs during execution of an atomic/extend sequence, then the presence of the Class B trap breaks the protection of atomic/extend operations, and the Class A trap will be executed immediately without waiting for the sequence completion. After return from the service routine, the **IP** is popped from the system stack and immediately pushed again because of the other pending Class B trap. In this situation, the interrupted instruction flow cannot be restored.*

All Class B traps have the same trap priority (trap priority I). When several Class B traps are active at the same time, the corresponding flags in the **TFR** are set, and the trap service routine is entered. Since all Class B traps have the same vector, the priority of service of Class B that occur simultaneously is determined by the software in the trap service routine.

During the execution of a Class A trap service routine, any Class B trap will not be serviced until the Class A trap service routine is exited with a RETI instruction. In this case, the Class B trap condition is stored in the **TFR** but the **IP** value of the instruction that caused this trap is lost.

### UNDefined OPCode Trap (UNDOPC)

When the instruction currently decoded by the MCU does not contain a valid C166S opcode, the **TFR.UNDOPC** flag is set, and the MCU enters the undefined opcode trap routine. The **IP** value pushed onto the system stack is the address of the instruction that caused the trap.

This can be used to emulate un-implemented instructions. The trap service routine can examine the faulting instruction to decode operands for un-implemented opcodes based on the stacked **IP**. To resume processing, the stacked **IP** value must be incremented by



**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

the size of the undefined instruction, which is determined by the user, before a RETI instruction is executed.

**PRotection FauLT Trap (PRTFLT)**

DISWDT, EINIT, IDLE, PWRDN, SRST, and SRVWDT are protected instructions. Whenever one protected instruction is executed and the protection is broken, the **TFR.PRTFLT** flag is set and the MCU enters the protection fault trap routine. The **IP** value pushed onto the system stack for the protection fault trap is the address of the instruction that caused the trap.

**ILLegal word OPerand Access Trap (ILLOPA)**

Whenever a word operand read or write access is attempted to an odd byte address, the **TFR.ILLOPA** flag is set, and the MCU enters the illegal word operand access trap routine. The **IP** value pushed onto the system stack is the address of the instruction following the one that caused the trap.

**ILLegal INstruction Access Trap (ILLINA)**

Whenever a branch is made to an odd byte address, the **TFR.ILLINA** flag is set and the MCU enters the illegal instruction access trap routine. The **IP** value pushed onto the system stack is the illegal odd target address of the branch instruction.

**ILLegal external BUS access Trap (ILLBUS)**

Whenever the MCU requests an external instruction fetch or a data read or a data write, and no external bus configuration has been specified, the **TFR.ILLBUS** flag R is set and the MCU enters the illegal bus access trap routine. The **IP** value pushed onto the system stack is the address of the instruction following the one that caused the trap.

**6.3.3.6 Peripheral Event Controller**

The Peripheral Event Controller (PEC) “decides” which MCU action is required to manage an interrupt request. It may be either normal interrupt service, or fast data transfer between two memory locations. The C166S PEC controls sixteen fast data transfer channels.

If a normal interrupt is requested, the MCU temporarily suspends the current program execution and branches to an Interrupt Service Routine (ISR). The current program status and context must be preserved.

If a PEC channel is selected for servicing an interrupt request, a single word or byte data transfer between any two memory locations is to be performed. During a PEC transfer, the normal program execution of the MCU is halted for just 1 machine cycle. No internal program status information needs to be saved. The PEC transfer is the fastest possible

**CONFIDENTIAL**

## **Central Processing Unit, PEC, and Interrupt**

interrupt response. In many cases, a PEC transfer is sufficient to service the peripheral request (serial channels, for example).

The PEC channels can perform the following actions:

- Byte or word transfer
- Continuous data transfer
- PEC channel-specific interrupt request upon data transfer completion; or for all channels a common End of PEC (EOP) interrupt for enhanced handling
- Automatic increment of source or destination pointers
- Channel linking of two PEC channels

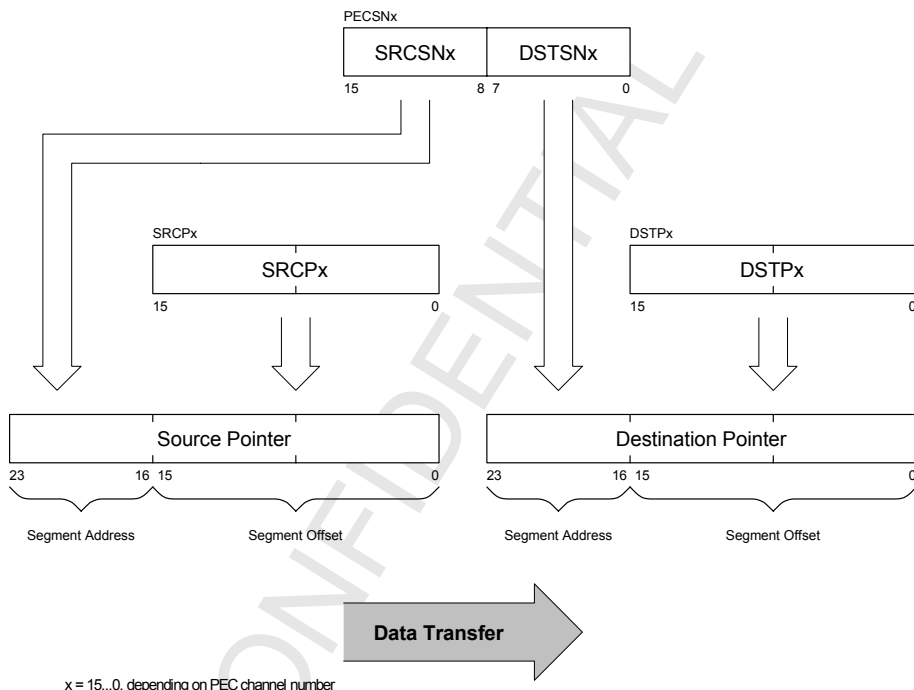
*Note: PEC transfer is executed if its priority level is higher than current MCU priority level.*

CONFIDENTIAL

### 6.3.3.6.1 The PEC Source and Destination Pointers

The PEC channels' source and destination pointers specify the locations between which the data is to be moved. All pointers are 24 bits wide. The 24-bit source address is stored in the internal DPRAM location **SRCPx** (lower 16 bits of address) and in the low byte of register **PECSNx** (highest 8 address bits)

**Figure 6-6 PEC Pointer Address Handling**



The 24-bit destination address is stored in the DPRAM<sup>1)</sup> location **DSTPx** (lower 16 bits of address) and in the high byte of register **PECSNx** (highest 8 address bits). Only the lower 16 bits of the PEC address pointers (segment offset) can be modified (incremented) by the PEC transfer mechanism. The highest 8 bits, which represent the segment number, are not modified by hardware. Therefore, the PEC pointers may be incremented within the address space of one segment and may not cross the segment border. If the offset address pointer has a value of  $FFFF_H$  in the case of byte transfers (BWT = 1) or  $FFFE_H$  in the case of word transfers (BWT = 0), the next increment will lead

<sup>1)</sup> Refer to [Section 6.4.3.1 DPRAM, \(E\)SFR and PEC Pointer Memory Areas \(on Page 194\)](#).

**CONFIDENTIAL**

## Central Processing Unit, PEC, and Interrupt

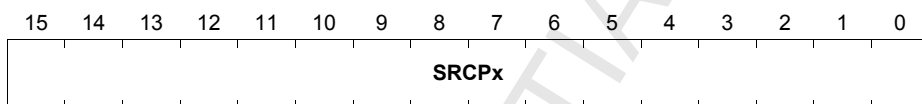
to an overflow. No explicit error event is generated by the system in case of address pointer(s) overflow; therefore, the user must prevent this condition from occurring.

*Note: If a word data transfer is selected for a specific PEC channel (that is, BWT = 0), the respective source and destination pointers must both contain a valid word address that points to an even byte boundary. Otherwise, the Illegal Word Access trap will be invoked when this channel is used.*

### SRCPx

#### PEC Source Pointer

Reset value: UUUU<sub>H</sub>

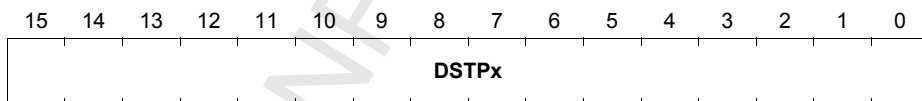


Field	Bits	Type	Description
SRCPx	15:0	rwh	<b>Source Pointer Address of Channel x</b> Source Address bits 15-0

### DSTPx

#### PEC Destination Pointer

Reset value: UUUU<sub>H</sub>



Field	Bits	Type	Description
DSTPx	15:0	rwh	<b>Destination Pointer Address of Channel x</b> Destination Address bits 15-0

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

**Table 6-8      DPRAM Addresses of PEC Source and Destination Pointer**

<b>Pointer</b>	<b>Address (Hex)</b>	<b>Pointer</b>	<b>Address (Hex)</b>	<b>Pointer</b>	<b>Address (Hex)</b>	<b>Pointer</b>	<b>Address (Hex)</b>
<b>DSTP7</b>	00 FCFE	<b>SRCP7</b>	00 FCFC	<b>DSTP11</b>	00 FCDE	<b>SRCP11</b>	00 FCDC
<b>DSTP6</b>	00 FCFA	<b>SRCP6</b>	00 FCF8	<b>DSTP10</b>	00 FCDA	<b>SRCP10</b>	00 FCD8
<b>DSTP5</b>	00 FCF6	<b>SRCP5</b>	00 FCF4	<b>DSTP9</b>	00 FCD6	<b>SRCP9</b>	00 FCD4
<b>DSTP4</b>	00 FCF2	<b>SRCP4</b>	00 FCF0	<b>DSTP8</b>	00 FCD2	<b>SRCP8</b>	00 FCD0
<b>DSTP3</b>	00 FCEE	<b>SRCP3</b>	00 FCEC	<b>DSTP15</b>	00 FCCE	<b>SRCP15</b>	00 FCCC
<b>DSTP2</b>	00 FCEA	<b>SRCP2</b>	00 FCE8	<b>DSTP14</b>	00 FCCA	<b>SRCP14</b>	00 FCC8
<b>DSTP1</b>	00 FCE6	<b>SRCP1</b>	00 FCE4	<b>DSTP13</b>	00 FCC6	<b>SRCP13</b>	00 FCC4
<b>DSTP0</b>	00 FCE2	<b>SRCP0</b>	00 FCE0	<b>DSTP12</b>	00 FCC2	<b>SRCP12</b>	00 FCC0

**PECSN<sub>x</sub>**

**PECSN<sub>0</sub>**

**PECSN<sub>1</sub>**

**PECSN<sub>2</sub>**

**PECSN<sub>3</sub>**

**PECSN<sub>4</sub>**

**PECSN<sub>5</sub>**

**PECSN<sub>6</sub>**

**PECSN<sub>7</sub>**

**PECSN<sub>8</sub>**

**PECSN<sub>9</sub>**

**PECSN<sub>10</sub>**

**PECSN<sub>11</sub>**

**PECSN<sub>12</sub>**

**PECSN<sub>13</sub>**

**PECSN<sub>14</sub>**

**PECSN<sub>15</sub>**

**PEC Segment Pointer**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>DSTSN<sub>x</sub></b>								<b>SRCSN<sub>x</sub></b>							

<b>Field</b>	<b>Bits</b>	<b>Type</b>	<b>Description</b>
<b>SRCSN<sub>x</sub></b>	7:0	rw	<b>Source Pointer Segment Address of Channel x</b> Source Address bits 23-16
<b>DSTSN<sub>x</sub></b>	15:8	rw	<b>Destination Pointer Segment Address of Channel x</b> Destination Address bits 23-16

CONFIDENTIAL

Central Processing Unit, PEC, and Interrupt

### 6.3.3.6.2 PEC Control Registers

Each PEC channel is controlled by the respective PEC Channel Control register (**PECCx**) and a set of source and destination pointers (**SRCPx**, **DSTPx**, and **PECSNx**), where “x” stands for the PEC channel number. The **PECCx** registers control the arbitration priority level assigned to the PEC channels and specifies the action to be performed.

PECCx  
PECC0  
PECC1  
PECC2  
PECC3  
PECC4  
PECC5  
PECC6  
PECC7  
PECC8  
PECC9  
PECC10  
PECC11  
PECC12  
PECC13  
PECC14  
PECC15

PEC Channel Control Register

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PT	EOP INT	PLEV	CL	INC	BWT	COUNT									

Field	Bits	Type	Description
COUNT	7:0	rwh	<b>PEC Transfer Count<sup>1)</sup></b> Counts PEC transfers and influences the channel's action
BWT	8	rw	<b>Byte/Word Transfer Selection</b> 0 Transfer a word 1 Transfer a byte

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

Field	Bits	Type	Description
<b>INC</b>	10:9	rw	<b>Increment Control</b> (Modification of source and destination pointer after PEC transfer) 00 No modification 01 Increment of destination pointer <b>DSTPx</b> by 1 ( <b>BWT</b> = 1) or by 2 ( <b>BWT</b> = 0) 10 Increment of source pointer <b>SRCPx</b> by 1 ( <b>BWT</b> = 1) or by 2 ( <b>BWT</b> = 0) 11 Reserved
<b>CL</b>	11	rw	<b>Channel Link Control</b> 0 PEC channels work independently 1 Pairs of channels are linked together
<b>PLEV</b>	13:12	rw	<b>PEC Level Selection<sup>2)</sup></b> This bitfield controls the PEC channel assignment to an arbitration priority level. (see section below)
<b>EOPINT</b>	14	rw	<b>End of PEC Interrupt Selection<sup>3)</sup></b> 0 EOP interrupt with the same level as the PEC transfer is triggered <sup>4)</sup> 1 EOP interrupt is serviced by a separate interrupt node with programmable interrupt level (EOPIC) and interrupt sharing control register (PECISNC)
<b>PT</b>	15	rw	<b>Transfer Mode<sup>5)</sup></b> 0 Short Transfer Mode 1 Long Transfer Mode

<sup>1)</sup> Refer to [Section 6.3.3.6.3 Short Transfer Mode \(on Page 133\)](#). The COUNT bitfield controls the PEC transfer only in case PT is set to 0.

<sup>2)</sup> Refer to [Section 6.3.3.6.6 PEC Channels Assignment and Arbitration \(on Page 136\)](#)

<sup>3)</sup> Refer to [Section 6.3.3.6.7 Programmable End of PEC Interrupt Level \(on Page 137\)](#)

<sup>4)</sup> The EOPIC interrupt node is not used in this case and has no function for this PEC channel.

<sup>5)</sup> The long transfer mode is only supported for PEC channels 0, 2, 4, and 6. The other channels do not support the long transfer mode, the PT-bit is hardwired to zero. Refer to [Section 6.3.3.6.3 \(on page 133\)](#) and [Section 6.3.3.6.4 Long Transfer Mode \(on Page 134\)](#).

- **Byte/Word Transfer bit (BWT)** of the **PECCx** register selects whether a byte or a word is to be moved during a PEC service cycle, and defines an increment step size for the pointer(s) to be modified.



## CONFIDENTIAL

## Central Processing Unit, PEC, and Interrupt

- **Increment Control Field (INC)** of the **PECCx** register defines when which one of the PEC pointers have to be incremented after the PEC transfer. If the pointers are not to be modified (INC = 00<sub>B</sub>), the respective channel will always move data from the same source to the same destination.

### 6.3.3.6.3 Short Transfer Mode

If the short transfer mode is enabled by the **PECCx.PT** flag (PT = 0) in the PEC control register, the PEC Transfer Count Field (**PECCx.COUNT**) controls directly the action of the respective PEC channel. The contents of the bitfield **PECCx.COUNT** can specify a certain number of PEC transfers, unlimited transfers, or no PEC service at all.

- a) If the **PECCx.COUNT** value is set to 00<sub>H</sub>, the normal interrupt requests are processed instead of PEC data transfers, and the corresponding PEC channel remains idle.
- b) Continuous data transfers are selected by setting the **PECCx.COUNT** to FF<sub>H</sub>. In this case, **COUNT** is not decremented by the transfers, and the respective PEC channel can serve unlimited number of PEC requests until it is modified by the program.
- c) If the **PECCx.COUNT** is set to service a specified number of requests by the respective PEC channel, it is decremented with each PEC transfer, and the request flag is cleared to indicate that the request has been serviced. When **COUNT** reaches 00<sub>H</sub> it activates the ISR that has the same priority level (**PECCx.EOPINT** = 0), or triggers the **PECCx.EOPIR** with a different priority level (**PECCx.EOPINT** = 1). When **COUNT** is decremented from 01<sub>H</sub> to 00<sub>H</sub> after a data transfer, the request flag is cleared if EOP INT is set to 1. If **PECCx.EOPINT** is 0, the request flag is be cleared and another interrupt request is generated on the same priority level. The respective PEC channel remains idle, and the associated ISR is activated instead of PEC transfer, because **COUNT** contains the 00<sub>H</sub> value.

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

### 6.3.3.6.4 Long Transfer Mode

If the long transfer mode<sup>1)</sup> is enabled by the PT flag (**PECCx.PT** = 1), the PEC Transfer Count Field (**PECXCx.COUNT2**) directly controls the action of the respective PEC channel.

**PECXCx**

**PECXC0**

**PECXC2**

**PECXC4**

**PECXC6**

**PECXC8**

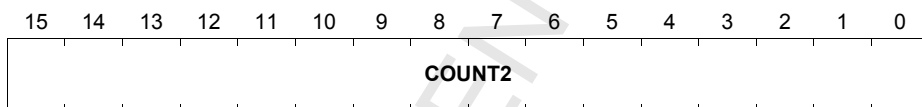
**PECXC10**

**PECXC12**

**PECXC14**

**PEC Extended Count Register**

(Reset value: 0000<sub>H</sub>)



Field	Bits	Type	Description
<b>COUNT2</b>	15:0	rwh	<b>PEC Extended (Long) Transfer Count</b> PEC transfer count extension

The long transfer mode is only available for PEC channels 0, 2, 4, and 6.

*Note: The channel link mode is independent of the long transfer mode. Both modes can be combined.*

*Note: The PEC Transfer Count Field (**PECCx.COUNT**) must be set to 0.*

*Note: Crossing of segment boundaries is not checked during data transfers with long transfer count, and is not supported. A wrap around occurs when reaching the segment boundary .*

The contents of the bitfield **PECXCx.COUNT2** may specify a certain number of PEC transfers or no PEC service at all. The 16-bit transfer counter permits servicing up to 65535 byte or word transfers.

- a) If the PEC transfer counter **COUNT2** value is set to 0000<sub>H</sub> , the normal interrupt requests are processed instead of PEC data transfers, and the corresponding PEC channel remains idle.

<sup>1)</sup> The long transfer mode is only supported for PEC channels 0, 2, 4 and 6.

CONFIDENTIAL

Central Processing Unit, PEC, and Interrupt

- b) If the bitfield **COUNT2** is set to service a specified number of requests by the respective PEC channel, it is decremented with each PEC transfer and the request flag is cleared to indicate that the request has been serviced. When **COUNT2** reaches 0000<sub>H</sub>, it activates the ISR that has the same priority level (**PECCx.EOPINT** = 0), or triggers the EOP ISR with a different priority level (**PECCx.EOPINT** = 1). When **COUNT2** is decremented from 0001<sub>H</sub> to 0000<sub>H</sub> after a data transfer, the request flag will be cleared if **PECCx.EOPINT** is set to 1. If **PECCx.EOPINT** is 0, the request flag is be cleared and another interrupt request is generated on the same priority level. The respective PEC channel remains idle and the associated interrupt service routine is activated instead of PEC transfer, because **COUNT2** contains the 0000<sub>H</sub> value. **COUNT2** is decremented only if it is set to FFFE<sub>H</sub> or to a value less than FFFE<sub>H</sub>.

### 6.3.3.6.5 Channel Link Mode for Data Chaining

Channel linking, if enabled, links two channels together to serve the data transfer requests of one peripheral. The whole data transfer (for example a message) is divided into separately-controlled block transfers. The two PEC channels that are linked together handle chained block transfers alternately with one another. At the end of a data block transfer controlled by one PEC channel, the other (linked) PEC channel is started automatically to continue the transfer with the next data block. Channel linking and data (block) chaining are supported within pairs of PEC channels (channels 0&1, 2&3, 4&5 etc.). Each data block is controlled by one PEC channel of the channel pair.

Channel linking is enabled if the Channel Link (CL) control bits of **both** PEC channels are set to 1 in their **PECCx** registers. The data transfer of linked channels must always be started always with the **even** numbered channel of the channel pair.

If in channel link mode the channel's data block is completely transferred, the PEC service request processing is automatically switched to the other PEC channel of the pair. CL of the previously active PEC channel is then reset.

Every channel toggle is indicated to MCU by means of an EOP interrupt. This makes it possible to set up multiple buffers for PEC transfers by changing pointer and count values of one channel while the other channel is active. Inside the EOP interrupt, the Channel Link Control bit CL must be set again before the channel is reactivated or the channel link mode is finished. This EOP interrupt is requested, indicated, and enabled in the respective PEC Interrupt Subnode Control Register (**PECISNC** or **PECCISNC**).

Thus, all EOP interrupts are controlled with the one EOP interrupt control register **EOPIC** and therefore with the same interrupt priority level. This service request node requests the MCU in case of one or more pending EOP interrupt requests if the respective enable control bit(s) are set in the according subnode control register and in the interrupt control register **EOPIC**.

If CL of the previous PEC channel is set to zero and the count field (**PECCx.COUNT** = 0 or **PECXCx.COUNT2** = 0, dependent on the mode) of the active

**CONFIDENTIAL**

## Central Processing Unit, PEC, and Interrupt

channel is zero as well, the whole data transfer is finished and the channel link interrupt represents a termination interrupt, the End of PEC interrupt.

The channel link feature is supported for all PEC channels, including the new PEC channels 8-15. The following table shows the channels that can be linked together and the channel numbers required to start transfers via linked channels.

**Table 6-9      PEC Channels That Can Be Linked Together<sup>1)</sup>**

Linked PEC Channels		Linked PEC Start Channel	Linked PEC Channels	
PEC Channel A	PEC Channel B		PEC Channel A	PEC Channel B
channel 0	channel 1	channel 0	channel 8	channel 9
channel 2	channel 3	channel 2	channel 10	channel 11
channel 4	channel 5	channel 4	channel 12	channel 13
channel 6	channel 7	channel 6	channel 14	channel 15

<sup>1)</sup> Refer to [Table 6-8 DPRAM Addresses of PEC Source and Destination Pointer \(on Page 129\)](#).

The two PEC control registers of a pair are linked to one interrupt control register, whereby in this IC register only the even-numbered PEC channel is indicated with the priority/group bits.

### 6.3.3.6.6 PEC Channels Assignment and Arbitration

The PEC channels can be assigned to arbitration priority levels. All requests with interrupt priority levels 8 to 15 can be associated with the PEC functionality (up to a total of sixteen PEC channels). The following formula shows how to program the bitfield **PECCx.PLEV** to set up a link to a certain interrupt priority level and a group priority level.

PEC channel:  $x = (x.3, x.2, x.1, x.0)$

linked to

Interrupt priority level:  $(1, \sim\text{PLEV}.1, \sim\text{PLEV}.0, x.2)$

Group priority level:  $(x.3, x.1, x.0)$

[0.1]

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

**Table 6-10** lists all possible combinations.

**Table 6-10 PEC Interrupt Level Control with PLEV Bits in PECCx Registers**

Priority Level		PEC Channel Selection (x)			
Interrupt Level ILVL3-0	Group Level xxGP, GLVL1,0	PLEV[1,0] = 00	PLEV[1,0] = 01	PLEV[1,0] = 10	PLEV[1,0] = 11
15	7-4	15-12 <sup>1)</sup>	-	-	-
15	3-0	7-4 <sup>2)</sup>	-	-	-
14	7-4	11-8 <sup>1)</sup>	-	-	-
14	3-0	3-0 <sup>2)</sup>	-	-	-
13	7-4	-	15-12 <sup>1)</sup>	-	-
13	3-0	-	7-4 <sup>2)</sup>	-	-
12	7-4	-	11-8 <sup>1)</sup>	-	-
12	3-0	-	3-0 <sup>2)</sup>	-	-
11	7-4	-	-	15-12 <sup>1)</sup>	-
11	3-0	-	-	7-4 <sup>2)</sup>	-
10	7-4	-	-	11-8 <sup>1)</sup>	-
10	3-0	-	-	3-0 <sup>2)</sup>	-
9	7-4	-	-	-	15-12 <sup>1)</sup>
9	3-0	-	-	-	7-4 <sup>2)</sup>
8	7-4	-	-	-	11-8 <sup>1)</sup>
8	3-0	-	-	-	3-0 <sup>2)</sup>

<sup>1)</sup> PEC channels 12/8 are assigned to Group Level 4, PEC channels 13/9 to Group Level 5, PEC channels 14/10 to Group Level 6 and PEC channels 15/11 to Group Level 7.

<sup>2)</sup> PEC channels 4/0 are assigned to Group Level 0, PEC channels 5/1 to Group Level 1, PEC channels 6/2 to Group Level 2 and PEC channels 7/3 to Group Level 3.

All interrupt requests that are not assigned to a PEC channel go directly to the interrupt handler.

### 6.3.3.6.7 Programmable End of PEC Interrupt Level

The programmable EOP interrupt supports PEC transfers, which need a high priority level for the transfer request, and which do not need the same priority level for the termination interrupt. One dedicated service request node with a programmable interrupt level is shared among all PEC channels. This service request node is controlled by the **EOPIC** interrupt control and the **PECISNC** and **PECXISNC** subnode control register.

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

**EOPIC**

**Interrupt Control Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED							xxGP	EOPIR	EOPIE	ILVL			GLVL		

Field	Bits	Type	Description
<b>GLVL</b>	1:0	rw	<b>Group Priority Level</b> 3 <sub>H</sub> Highest priority level ... 0 <sub>H</sub> Lowest priority level
<b>ILVL</b>	5:2	rw	<b>Interrupt Priority Level</b> F <sub>H</sub> Highest priority level ... 0 <sub>H</sub> Lowest priority level
<b>EOPIE</b>	6	rw	<b>Interrupt Enable Control Bit</b> 0 Interrupt request is disabled 1 Interrupt request is enabled
<b>EOPIR<sup>1)</sup></b>	7	rwh	<b>Interrupt Request Flag</b> 0 No request pending 1 This source has raised an interrupt request
<b>xxGP</b>	8	rw	<b>Group Priority Extension</b> Defines the value of high order group level bit
<b>RESERVED</b>	15:9	r	Reserved, these bits must be left at their reset values.

<sup>1)</sup> Bit EOPIR supports bit-protection.

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

The Register **PECISNC** and **PECXISNC** contain flags of the EOP interrupt node. This node is used when the enhanced End of PEC interrupt feature is invoked and control bit **PECCx.EOPINT** is set to 1 in the corresponding **PECCx**.

**PECISNC**

**PEC Interrupt Sub Node Control 1**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C7IR	C7IE	C6IR	C6IE	C5IR	C5IE	C4IR	C4IE	C3IR	C3IE	C2IR	C2IE	C1IR	C1IE	C0IR	C0IE

**PECXISNC**

**PEC Interrupt Sub Node Control 2**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C15IR	C15IE	C14IR	C14IE	C13IR	C13IE	C12IR	C12IE	C11IR	C11IE	C10IR	C10IE	C9IR	C9IE	C8IR	C8IE

Field	Bits	Type	Description
<b>CxIE</b>	14, 12, 10, 8, 6, 4, 2, 0	rw	<b>Interrupt Sub Node Enable Control Bit of PEC Channels 0 to 15<sup>1)</sup></b> (individually enables/disables a specific source) 0 EOP interrupt request of PEC channel x is disabled 1 EOP interrupt request of PEC channel x is enabled
<b>CxIR</b>	15, 13, 11, 9, 7, 5, 3, 1	rwh	<b>Interrupt Sub Node Request Flags of PEC Channels 0 to 15<sup>2)</sup></b> 0 No special EOP interrupt request is pending for PEC channel x 1 PEC channel x has raised an EOP interrupt request

- <sup>1)</sup> It is recommended that an interrupt request flag (CxIR) be cleared before setting the respective enable flag (CxIE). Otherwise, pending former requests immediately trigger an interrupt request after setting the enable bit.
- <sup>2)</sup> The EOP sub-node interrupt request flags are not cleared by hardware when entering the ISR (interrupt has been accepted by the MCU), unlike the interrupt request flags of the interrupt nodes (request flags **xxIC.xxIR**). The ISR has to check the request flags and to clear them before executing the RETI instruction.

## CONFIDENTIAL

## Central Processing Unit, PEC, and Interrupt

### 6.3.3.7 Prioritization of Interrupt and PEC Service Requests

Interrupt and PEC service requests from all sources can be enabled, so they are arbitrated and serviced (if they win), or they may be disabled, so their requests are disregarded and not serviced.

### 6.3.3.8 Enabling/Disabling Interrupt Requests

There are three mechanisms for enabling/disabling interrupt requests.

- **Control Bits** allow to switch each individual source “ON” or “OFF”, so it may generate a request or not. The control bits (xxIE) are located in the respective interrupt control registers. All interrupt requests may be enabled or disabled generally via bit **PSW.IEN**. This control bit is the “main switch” that selects, if requests from any source are accepted or not.  
For a specific request to be arbitrated the respective source’s enable bit and the global enable bit must both be set.
- **The Priority Level** automatically selects a certain group of interrupt requests that will be acknowledged, disclosing all other requests. The priority level of the source that won the arbitration is compared against the MCU current level and the source is only serviced, if its level is higher than the current MCU level. Changing the MCU level to a specific value via software blocks all requests on the same or a lower level. An interrupt source that is assigned to level 0 will be disabled and never be serviced.
- **The ATOMIC and EXTend instructions** automatically disable all interrupt requests for the duration of the following 1...4 instructions. This is useful, for example, for semaphore handling and does not require to re-enable the interrupt system after the un-separable instruction sequence.

### Interrupt Class Management

An interrupt class covers a set of interrupt sources with the same importance, that is, the same priority from the system viewpoint. Interrupts of the same class must not interrupt each other. The PMB7870 supports this function with two features:

- Classes with up to 8 members can be established by using the same interrupt priority (ILVL) and assigning a dedicated group level (GLVL) to each member. This functionality is built-in and handled automatically by the interrupt controller.
- Classes with more than 4 members can be established by using a number of adjacent interrupt priorities (ILVL) and the respective group levels (4 per ILVL). Each interrupt service routine within this class sets the MCU level to the highest interrupt priority within the class. All requests from the same or any lower level are blocked now, that is, no request of this class is accepted.

The example below establishes 3 interrupt classes which cover 2 or 3 interrupt priorities, depending on the number of members in a class. A level 6 interrupt disables all other



**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

sources in class 2 by changing the current MCU level to 8, which is the highest priority (ILVL) in class 2. Class 1 requests or PEC requests are still serviced in this case.

The 24 interrupt sources (excluding PEC requests) are so assigned to 3 classes of priority rather than to 7 different levels, as the hardware support would do.

**Table 6-11 Software Controlled Interrupt Classes (Example)**

ILVL (Priority)	GLVL								Interpretation
	7	6	5	4	3	2	1	0	
15									PEC service on up to 16 channels
14									
13									
12				X	X	X	X	X	Interrupt Class 1 5 sources on 2 levels
11									
10									
9									Interrupt Class 2 9 sources on 3 levels
8	X	X	X	X	X	X	X	X	
7	X								
6									
5			X	X	X	X	X	X	Interrupt Class 3 5 sources on 2 levels
4									
3									
2									
1									
0									
									No service!

### 6.3.3.9 Saving the Status during Interrupt Service

Before an interrupt request that has been arbitrated is actually serviced, the status of the current task is automatically saved on the system stack. The MCU status (**PSW**) is saved along with the location, where the execution of the interrupted task is to be resumed after returning from the service routine. This return location is specified through the Instruction Pointer (**IP**) and, in case of a segmented memory model, the Code Segment Pointer (**CSP**). **SYSCON.SGTDIS** controls, how the return location is stored.

For details refer to [Section 6.3.3.4.1 Saving the Status during Interrupt Service \(on Page 117\)](#).

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

### 6.3.3.10 Fast Interrupts

Refer to [Section 6.3.3 Interrupt and Exception Execution \(on Page 111\)](#).

The interrupt inputs) are sampled every 8 states (16 TCL), that is, external events are scanned and detected in timeframes of 16 TCL. E-GOLDradio provides an addition 8 interrupt inputs (*fex\_int* bus) that are sampled every 2 TCL, so external events are captured faster than with standard interrupt inputs.

Refer to [Section 6.7.5.2 Fast External Interrupt and Interrupt Source Control \(on Page 251\)](#).

### 6.3.3.11 Interrupt List

**Table 6-12 E-GOLDradio Interrupt List**

INT Number	INT Type	Source of Interrupt or PEC Service Request	INT Name	Vector Location (Hex)	Trap Number (Hex)	Trap Number (Decimal)	SFR Address (Hex)
0	0	CAPCOM-Register-0	CC0_INT	00 0040	10	16	FF78
1	0	CAPCOM-Register-1	CC1_INT	00 0044	11	17	FF7A
2	0	CAPCOM-Register-2	CC2_INT	00 0048	12	18	FF7C
3	0	CAPCOM-Register-3	CC3_INT	00 004C	13	19	FF7E
4	0	CAPCOM-Register-4	CC4_INT	00 0050	14	20	FF80
5	0	CAPCOM-Register-5	CC5_INT	00 0054	15	21	FF82
6	0	CAPCOM-Register-6	CC6_INT	00 0058	16	22	FF84
7	0	CAPCOM-Register-7	CC7_INT	00 005C	17	23	FF86
8	0	CAPCOM-register-16	CC16_INT	00 0060	18	24	FF88
9	0	CAPCOM-register-17	CC17_INT	00 0064	19	25	FF8A
10	0	CAPCOM-register-18	CC18_INT	00 0068	1A	26	FF8C
11	0	CAPCOM-register-19	CC19_INT	00 006C	1B	27	FF8E

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

**Table 6-12 E-GOLDRadio Interrupt List**

INT Number	INT Type	Source of Interrupt or PEC Service Request	INT Name	Vector Location (Hex)	Trap Number (Hex)	Trap Number (Decimal)	SFR Address (Hex)
12	0	CAPCOM-register-20	CC20_INT	00 0070	1C	28	FF90
13	0	CAPCOM-register-21	CC21_INT	00 0074	1D	29	FF92
14	0	CAPCOM-register-22	CC22_INT	00 0078	1E	30	FF94
15	0	CAPCOM-register-23	CC23_INT	00 007C	1F	31	FF96
16	0	CAPCOM-Timer-0	T0_INT	00 0080	20	32	FF9C
17	0	CAPCOM-Timer-1	T1_INT	00 0084	21	33	FF9E
18	0	GPT1-Timer-2	T2_INT	00 0088	22	34	FF60
19	0	GPT1-Timer-3	T3_INT	00 008C	23	35	FF62
20	0	GPT1-Timer-4	T4_INT	00 0090	24	36	FF64
21	0	GPT2-Timer-5	T5_INT	00 0094	25	37	FF66
22	0	GPT2-Timer-6	T6_INT	00 0098	26	38	FF68
23	0	CAPCOM-Timer-7	T7_INT	00 009C	27	39	FF6A
24	0	CAPCOM-Timer-8	T8_INT	00 00A0	28	40	FF98
25	0	GPT2-CAPREL-Register	CR_INT	00 00A4	29	41	FF9A
26	0	RTC (RTC_INT from RTC)	RTC_INT	00 00A8	2A	42	FF6C
27	0	RTC_T14 (RTC_T14_INT from RTC)	RTC_T14_INT	00 00AC	2B	43	FF6E
28	0	ASC0-Transmit	S0T_INT	00 00B0	2C	44	FF70
29	0	ASC0-Receive	S0R_INT	00 00B4	2D	45	FF72
30	0	ASC0-Error	S0E_INT	00 00B8	2E	46	FF74
31	0	ASC0-Transmit-Buffer	S0TB_INT	00 00BC	2F	47	FF76

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

**Table 6-12 E-GOLDRadio Interrupt List**

INT Number	INT Type	Source of Interrupt or PEC Service Request	INT Name	Vector Location (Hex)	Trap Number (Hex)	Trap Number (Decimal)	SFR Address (Hex)
32	0	ASC0-Autobaud-Start	S0ABST_INT	00 00C0	30	48	F160
33	0	ASC0-Autobaud-Detect	S0ABDET_INT	00 00C4	31	49	F162
34	0	ASC0-Cts	S0CTS_INT	00 00C8	32	50	F164
35	0	ASC0-Tm0	S0TM0_INT	00 00CC	33	51	F166
36	0	ASC1-Transmit	S1T_INT	00 00D0	34	52	F168
37	0	ASC1-Receive	S1R_INT	00 00D4	35	53	F16A
38	0	ASC1-Error	S1E_INT	00 00D8	36	54	F16C
39	0	ASC1-Transmit-Buffer	S1TB_INT	00 00DC	37	55	F16E
40	0	SSC-Transmit	SSC0T_INT	00 00E0	38	56	F170
41	0	SSC-Receive	SSC0R_INT	00 00E4	39	57	F172
42	0	SSC-Error	SSC0E_INT	00 00E8	3A	58	F174
43	0	Reserved MMCI	RES1_INT	00 00EC	3B	59	F176
44	0	Reserved	RES2_INT	00 00F0	3C	60	F178
45	2	Reserved	RES3_INT	00 00F4	3D	61	F17A
46	2	Keypad-interrupt-KPD_INT	KPD_INT	00 00F8	3E	62	F17C
47	4	Power-Management	ECO_INT	00 00FC	3F	63	F17E
48	3	uC-interrupt-from-autostart	S0AS_INT	00 0100	40	64	F186
49	2	T_INT1-of-GSM-Timer	T_INT1	00 0104	41	65	F18E
50	2	T_INT2-of-GSM-Timer	T_INT2	00 0108	42	66	F196
51	2	INT_GP(0)-of-GSM-Timer	TIM0_INT	00 010C	43	67	F19E
52	2	INT_GP(1)-of-GSM-Timer	TIM1_INT	00 0110	44	68	F184

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

**Table 6-12 E-GOLDRadio Interrupt List**

INT Number	INT Type	Source of Interrupt or PEC Service Request	INT Name	Vector Location (Hex)	Trap Number (Hex)	Trap Number (Decimal)	SFR Address (Hex)
53	2	INT_GP(2)-of-GSM-Timer	TIM2_INT	00 0114	45	69	F18C
54	2	INT_GP(3)-of-GSM-Timer	TIM3_INT	00 0118	46	70	F194
55	2	INT_GP(4)-of-GSM-Timer	TIM4_INT	00 011C	47	71	F19C
56	??	Measurement interrupt	MEAS0_INT	00 0120	48	72	F182
57	??	Measurement interrupt	MEAS1_INT	00 0124	49	73	F18A
58	??	Measurement interrupt toggled	MEAS0_TOGGLE_INT	00 0128	4A	74	F192
59	??	Measurement interrupt toggled	MEAS1_TOGGLE_INT	00 012C	4B	75	F19A
60	2	RFSSSTINT-of-GSM-Timer	RFSSOT_INT	00 0130	4C	76	F180
61	0 ??	I2C data	IIC_D_INT	00 0134	4D	77	F190
62	0 ??	I2C protocol	IIC_P_INT	00 0138	4E	78	F198
63	0 ??	I2C error	IIC_E_INT	00 013C	4F	79	F188
64	2	SIMCard-Interface-OK	SIMOK_INT	00 0140	50	80	F140
65	2	SIMCard-Interface-BAD	SIMERR_INT	00 0144	51	81	F142
66	2	SIMCard-in-(pad CCIN => from SIMCARD)	SIMIN_INT	00 0148	52	82	F144
67	0a	External-Interrupt-0/and-or s0rint	FEX0_INT	00 014C	53	83	F146
68	0a	External-Interrupt-1/and-or siminint	FEX1_INT	00 0150	54	84	F148

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

**Table 6-12 E-GOLDRadio Interrupt List**

INT Number	INT Type	Source of Interrupt or PEC Service Request	INT Name	Vector Location (Hex)	Trap Number (Hex)	Trap Number (Decimal)	SFR Address (Hex)
69	0a	External-Interrupt-2/and-or kpdint	FEX2_INT	00 0154	55	85	F14A
70	0a	External-Interrupt-3/and-or rtc_int	FEX3_INT	00 0158	56	86	F14C
71	0a	External-Interrupt-4 and/or ex4bint	FEX4_INT	00 015C	57	87	F14E
72	0a	External-Interrupt-5 and/or ex5bint	FEX5_INT	00 0160	58	88	F150
73	0a	External-Interrupt-6 and/or tim2int	FEX6_INT	00 0164	59	89	F152
74	0a	External-Interrupt-7 and/or t3int	FEX7_INT	00 0168	5A	90	F154
75	3	DSP-Interrupt-0	FROM_DSP_TO_MCU0_INT	00 016C	5B	91	F156
76	3	DSP-Interrupt-1	FROM_DSP_TO_MCU1_INT	00 0170	5C	92	F158
77	3	DSP-Interrupt-2	FROM_DSP_TO_MCU2_INT	00 0174	5D	93	F15A
78	3	DSP-Interrupt-3	FROM_DSP_TO_MCU3_INT	00 0178	5E	94	F15C
79	0	PEC-link	PECLIN_INT	00 017C	5F	95	F15E
80	0	GEA3	GEA3_int	00 0180	60	96	F112

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

**Table 6-13 Interrupt Types**

Type	Description	Duration
<b>0</b>	Exact sync High level interrupt, synchronous to CPU clock	2 CPU clk
<b>0a</b>	Fast external interrupt to SCU	$\geq 2$ CPU clk
<b>1</b>	Short High level interrupt, asynchronous to CPU clock	DSP clk $\leq$ CPU clk
<b>2</b>	Variable Toggle interrupt, asynchronous to CPU clock	DSP clk
<b>3</b>	Variable High level/Rising edge interrupt	DSP clk
<b>4</b>	Variable High level/Rising edge interrupt	4.6 ms

### 6.3.4 Using General-Purpose Registers

The C166S uses several banks of 16 dedicated General Purpose Registers (GPRs) **R0, R1, R2... R15** (refer to [Section 6.3.9.1 General Purpose Registers \(on Page 187\)](#)) that can be accessed in one MCU cycle. The GPRs are the working registers of the Arithmetic and Logic Units (ALU) and may also serve as address pointers in indirect addressing modes.

Several banks of GPRs are memory-mapped. The banks of these GPRs are located in the DPRAM. One bank uses a block of 16 consecutive words. A Context Pointer (**CP**) register determines the base address of the currently selected bank.

The C166S can switch the complete GPR bank with a single instruction for time-critical tasks. After switching, the new task is executed within its own separate context.

There are 3 different ways to access the GPRs:

1. **Short 4-bit GPR addresses** (mnemonic: Rw or Rb) specify an address relative to the memory location pointed to by the contents of the **CP** register, that is, the base of contents of the current register bank. Both byte-wise and word-wise GPR accesses are possible. The short 4-bit GPR address is logically added to the contents of register **CP** if a byte (Rb) GPR address is specified, or multiplied by two and then added to **CP** if a word (Rw) GPR address is specified (see [Figure 6-7](#)).

*Note: If GPRs are used as indirect address pointers, they are always accessed word-wise.*

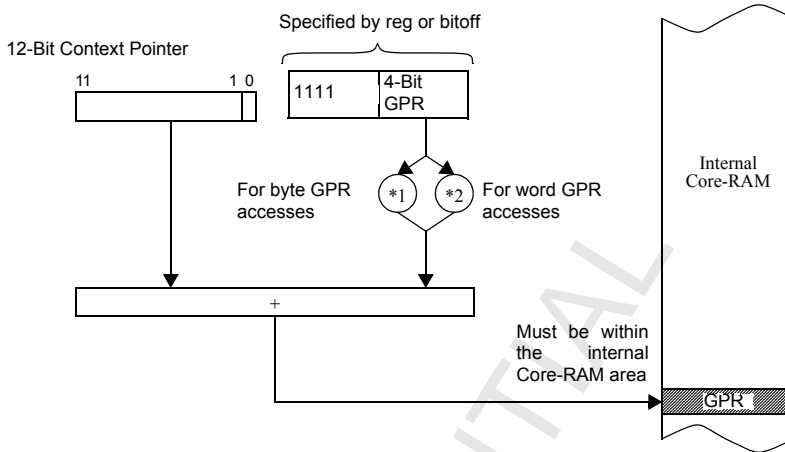
For some instructions, only the first 4 GPRs (R0, R1, R2 and R3) can be used as indirect address pointers. These GPRs are specified via short 2-bit GPR addresses. The physical address calculation is identical to the one for the short 4-bit GPR addresses.

2. **Short 8-bit register addresses** (mnemonic: reg or bitoff) within a range from  $F0_H$  to  $FF_H$  interpret the four least-significant bits as a short 4-bit GPR address, while the four most significant bits are ignored. The physical GPR address is calculated in a similar fashion as the short 4-bit GPR addresses. For single-bit GPR accesses, the GPR's word address is calculated in the same way. The accessed bit position within the word is specified by a separate additional 4-bit value.

#### Figure 6-7 Implicit CP Use by logical Short GPR Addressing Modes

3. **24-Bit memory addresses** within a range from (**CP**)+0 to (**CP**)+30 can be used to access GPRs directly. Both byte and word GPR accesses are possible. The 24-bit memory address is generated according to the rules for long- and indirect-addressing mode, refer to [Section 6.3.5.2 Long and Indirect Addressing Modes \(on Page 155\)](#).





**Table 6-14 Addressing Modes to Access Word-GPRs**

Name	Physical Address	8-Bit Address	4-Bit Address	Description	Reset Value
R0	(CP)+0	F0 <sub>H</sub>	0 <sub>H</sub>	General-Purpose word Register R0	UUUU <sub>H</sub>
R1	(CP)+2	F1 <sub>H</sub>	1 <sub>H</sub>	General-Purpose word Register R1	UUUU <sub>H</sub>
R2	(CP)+4	F2 <sub>H</sub>	2 <sub>H</sub>	General-Purpose word Register R2	UUUU <sub>H</sub>
R3	(CP)+6	F3 <sub>H</sub>	3 <sub>H</sub>	General-Purpose word Register R3	UUUU <sub>H</sub>
R4	(CP)+8	F4 <sub>H</sub>	4 <sub>H</sub>	General-Purpose word Register R4	UUUU <sub>H</sub>
R5	(CP)+10	F5 <sub>H</sub>	5 <sub>H</sub>	General-Purpose word Register R5	UUUU <sub>H</sub>
R6	(CP)+12	F6 <sub>H</sub>	6 <sub>H</sub>	General-Purpose word Register R6	UUUU <sub>H</sub>
R7	(CP)+14	F7 <sub>H</sub>	7 <sub>H</sub>	General-Purpose word Register R7	UUUU <sub>H</sub>
R8	(CP)+16	F8 <sub>H</sub>	8 <sub>H</sub>	General-Purpose word Register R8	UUUU <sub>H</sub>
R9	(CP)+18	F9 <sub>H</sub>	9 <sub>H</sub>	General-Purpose word Register R9	UUUU <sub>H</sub>
R10	(CP)+20	FA <sub>H</sub>	A <sub>H</sub>	General-Purpose word Register R10	UUUU <sub>H</sub>
R11	(CP)+22	FB <sub>H</sub>	B <sub>H</sub>	General-Purpose word Register R11	UUUU <sub>H</sub>
R12	(CP)+24	FC <sub>H</sub>	C <sub>H</sub>	General-Purpose word Register R12	UUUU <sub>H</sub>
R13	(CP)+26	FD <sub>H</sub>	D <sub>H</sub>	General-Purpose word Register R13	UUUU <sub>H</sub>
R14	(CP)+28	FE <sub>H</sub>	E <sub>H</sub>	General-Purpose word Register R14	UUUU <sub>H</sub>
R15	(CP)+30	FF <sub>H</sub>	F <sub>H</sub>	General-Purpose word Register R15	UUUU <sub>H</sub>

*Note: U = undefined*

*Note: The first 8 GPRs (R7...R0) may also be accessed byte-wise.*

*Note: Writing to a GPR byte does not affect the other byte of the same GPR.*

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

Each half of the byte-wise accessible registers has a special name (refer to [Table 6-15](#)).

**Table 6-15 Addressing Modes to Access Byte-GPRs**

Name	Physical Address	8-Bit Address	4-Bit Address	Description	Reset Value
RL0	( <a href="#">CP</a> )+0	F0 <sub>H</sub>	0 <sub>H</sub>	General-Purpose byte Register RL0	UU <sub>H</sub>
RH0	( <a href="#">CP</a> )+1	F1 <sub>H</sub>	1 <sub>H</sub>	General-Purpose byte Register RL1	UU <sub>H</sub>
RL1	( <a href="#">CP</a> )+2	F2 <sub>H</sub>	2 <sub>H</sub>	General-Purpose byte Register RL2	UU <sub>H</sub>
RH1	( <a href="#">CP</a> )+3	F3 <sub>H</sub>	3 <sub>H</sub>	General-Purpose byte Register RL3	UU <sub>H</sub>
RL2	( <a href="#">CP</a> )+4	F4 <sub>H</sub>	4 <sub>H</sub>	General-Purpose byte Register RL4	UU <sub>H</sub>
RH2	( <a href="#">CP</a> )+5	F5 <sub>H</sub>	5 <sub>H</sub>	General-Purpose byte Register RL5	UU <sub>H</sub>
RL3	( <a href="#">CP</a> )+6	F6 <sub>H</sub>	6 <sub>H</sub>	General-Purpose byte Register RL6	UU <sub>H</sub>
RH3	( <a href="#">CP</a> )+7	F7 <sub>H</sub>	7 <sub>H</sub>	General-Purpose byte Register RL7	UU <sub>H</sub>
RL4	( <a href="#">CP</a> )+8	F8 <sub>H</sub>	8 <sub>H</sub>	General-Purpose byte Register RL8	UU <sub>H</sub>
RH4	( <a href="#">CP</a> )+9	F9 <sub>H</sub>	9 <sub>H</sub>	General-Purpose byte Register RL9	UU <sub>H</sub>
RL5	( <a href="#">CP</a> )+10	FA <sub>H</sub>	A <sub>H</sub>	General-Purpose byte Register RL10	UU <sub>H</sub>
RH5	( <a href="#">CP</a> )+11	FB <sub>H</sub>	B <sub>H</sub>	General-Purpose byte Register RL11	UU <sub>H</sub>
RL6	( <a href="#">CP</a> )+12	FC <sub>H</sub>	C <sub>H</sub>	General-Purpose byte Register RL12	UU <sub>H</sub>
RH6	( <a href="#">CP</a> )+13	FD <sub>H</sub>	D <sub>H</sub>	General-Purpose byte Register RL13	UU <sub>H</sub>
RL7	( <a href="#">CP</a> )+14	FE <sub>H</sub>	E <sub>H</sub>	General-Purpose byte Register RL14	UU <sub>H</sub>
RH7	( <a href="#">CP</a> )+15	FF <sub>H</sub>	F <sub>H</sub>	General-Purpose byte Register RL15	UU <sub>H</sub>

#### 6.3.4.1 Context Switching

An Interrupt Service Routine (ISR) or a task scheduler of an operating system usually saves the contents of all used registers into the stack, and restores them before returning. The more registers a routine uses, the more time is wasted by saving and restoring.

The contents of the register bank are switched by changing the base address of the memory-mapped GPR bank. The base address is given by the contents of the Context Pointer ([CP](#)) register.

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

**The Context Pointer**

The **CP** register is not bit-addressable. It can be updated via any instruction capable of modifying SFRs.

**CP**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	<b>CP</b>											0

Field	Bits	Type	Description
<b>0</b>	0	r	This bit is fixed to 0. ( <b>CP</b> is always word-aligned.)
<b>CP</b>	11:1	rw	<b>Modifiable Portion of CP Register</b> Specifies the (word) base address of the current memory-mapped register bank.  <i>Note: When writing a value to register <b>CP</b> with bits <b>CP</b>[11:9] = 000, bits <b>CP</b>[11:10] are set to 11 by hardware.</i>
<b>1</b>	15:12	r	This bit is fixed to 1. ( <b>CP</b> always points in the DPRAM.)

*Note: It is the user responsibility to ensure that the physical GPR address specified via **CP** register plus short GPR address must always be an RAM location. If this condition is not met, unexpected results may occur. Do not set **CP** below the DPRAM start address.*

*Note: Due to the internal instruction pipeline, a new **CP** value cannot be used for GPR address calculations for the instruction immediately following the instruction updating the **CP** register.*

The C166S switches the complete memory-mapped GPR bank with a single instruction. After switching, the service routine executes within its own separate context.

The instruction SCXT CP, #New\_Bank pushes the value of the current context pointer (**CP**) into the system stack and loads **CP** with the immediate value New\_Bank, which selects a new register bank. The service routine may now use its own registers. This memory register bank is preserved when the service routine terminates, that is, its contents are available on the next call. Before returning from the service routine (RETI), the previous **CP** is simply popped from the system stack, which returns the registers to the original bank.

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

### 6.3.5 Data Addressing

The C166S provides a lot of powerful addressing modes for word-wise, byte-wise and bitwise data accesses (short, long, indirect). The different addressing modes use different formats and have different scopes.

The following major tasks are performed:

- Address generation using short-, long- and indirect-addressing modes
- Data paging or overwriting mechanism
- System stack handling.

#### 6.3.5.1 Short Addressing Modes

All of these addressing modes use an implicit base offset address to specify a 24-bit physical address. Short addressing modes allow access to the GPRs, SFRs, or bit-addressable memory space:

$$\text{Physical Address} = \text{Base Address} + \Delta * \text{Short Address}$$

*Note:  $\Delta$  is 1 for byte-wise GPRs,  $\Delta$  is 2 for word-wise GPRs.*

**Table 6-16 Short addressing modes**

Mnemonic	Physical Address	Short Address Range	Scope of Access
<b>Rw</b>	( <b>CP</b> ) + 2*Rw or local	Rw = 0...15	GPRs (Word)
<b>Rb</b>	( <b>CP</b> ) + 1*Rb or local	Rb = 0...15	GPRs (Byte)
<b>reg</b>	00 FE00 <sub>H</sub> + 2*reg 00 F000 <sub>H</sub> + 2*reg ( <b>CP</b> ) + 2*(reg^0F <sub>H</sub> ) ( <b>CP</b> ) + 1*(reg^0F <sub>H</sub> )	reg = 00 <sub>H</sub> ...EF <sub>H</sub> reg = 00 <sub>H</sub> ...EF <sub>H</sub> reg = F0 <sub>H</sub> ...FF <sub>H</sub> reg = F0 <sub>H</sub> ...FF <sub>H</sub>	SFRs (Word, Low byte) ESFRs (Word, Low byte) GPRs (Word) GPRs (Bytes)
<b>bitoff</b>	00 FD00 <sub>H</sub> + 2*bitoff 00 FF00 <sub>H</sub> + 2*(bitoff^7F <sub>H</sub> ) 00 F100 <sub>H</sub> + 2*(bitoff^7F <sub>H</sub> ) ( <b>CP</b> ) + 2*(bitoff^0F <sub>H</sub> )	bitoff = 00 <sub>H</sub> ...7F <sub>H</sub> bitoff = 80 <sub>H</sub> ...EF <sub>H</sub> bitoff = 80 <sub>H</sub> ...EF <sub>H</sub> bitoff = F0 <sub>H</sub> ...FF <sub>H</sub>	DPRAM Bit word offset SFR Bit word offset ESFR Bit word offset GPR Bit word offset
<b>bitaddr</b>	Word offset as with bitoff. Immediate bit position.	bitoff = 00 <sub>H</sub> ...FF <sub>H</sub> bitpos = 0...15	Any single bit

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

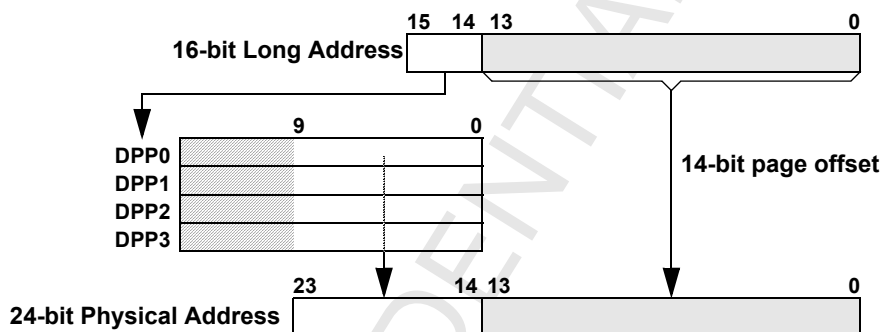
- **Rw,**
- **Rb:**  
They specify direct access to any GPR in the currently active context. Both Rw and Rb require 4 bits in the instruction format. The base address of the global register bank is determined by the contents of register **CP**. Rw specifies a 4-bit word GPR address relative to the base address (**CP**), while Rb specifies a 4-bit byte GPR address relative to the base address (**CP**).
- **reg:**  
Specifies direct access to any (E)SFR or GPR in the currently active context. The reg value requires 8 bits in the instruction format. Short reg addresses in the range from 00<sub>H</sub> to EF<sub>H</sub> always specify (E)SFRs. In that case, the factor  $\Delta$  equals 2, and the base address is 00 FE00<sub>H</sub> for the standard SFR area or 00 F000<sub>H</sub> for the extended ESFR area. The reg accesses to the ESFR area require a preceding EXTR instruction to switch the base address. Depending on the opcode, either the total word (for word operations) or the low byte (for byte operations) of an SFR can be addressed via reg. The high byte of an SFR cannot be accessed via the reg addressing mode. Short reg addresses in the range from F0<sub>H</sub> to FF<sub>H</sub> always specify GPRs. In that case, only the lower 4 bits of reg are significant for physical address generation and, therefore, the address calculation is identical to the address generation process described for the Rb and Rw addressing modes.
- **bitoff:**  
Specifies direct access to any word in the bit-addressable memory space. The bitoff value requires 8 bits in the instruction format. Depending on the specified bitoff range, different base addresses are used to generate physical addresses: Short bitoff addresses in the range from 00<sub>H</sub> to 7F<sub>H</sub> use 00 FD00<sub>H</sub> as a base address to specify the 128 highest DPRAM word locations in the range from 00 FD00<sub>H</sub> to 00 FDFE<sub>H</sub>. Short bitoff addresses in the range from 80<sub>H</sub> to EF<sub>H</sub> use base address 00 FF00<sub>H</sub> to specify the internal SFR word locations in the range from 00 FF00<sub>H</sub> to 00 FFDE<sub>H</sub> or base address 00 F100<sub>H</sub> to specify the internal ESFR word locations in the range from 00 F100<sub>H</sub> to 00 F1DE<sub>H</sub>. The bitoff accesses to the ESFR area require a preceding EXTR instruction to switch the base address. For short bitoff addresses from F0<sub>H</sub> to FF<sub>H</sub>, only the lowest four bits are used to generate the address of the selected word GPR.
- **bitaddr:**  
Any bit address is specified by a word address within the bit-addressable memory space (see bitoff), and by a bit position (bitpos) within that word. Therefore, bitaddr requires 12 bits in the instruction format.

### 6.3.5.2 Long and Indirect Addressing Modes

These addressing modes use one of the 4 DPP registers to specify a 24-bit address. Any word or byte data within the entire address space can be accessed with these modes. Any long or indirect 16-bit address contains two parts that have different meanings. Bits 13-0 specify a 14-bit data page offset, while bits 15-14 specify the **Data Page Pointer (DPP)** (1 of 4) register, which is used to generate the full 24-bit address (see [Figure 6-8](#)).

The C166S also supports an override mechanism for the DPP addressing scheme [EXTP(R) and EXTS(R) instructions].

**Figure 6-8 Interpretation of a 16-bit Long Address**



*Note: Word accesses on odd byte addresses are not executed. A hardware trap is triggered.*

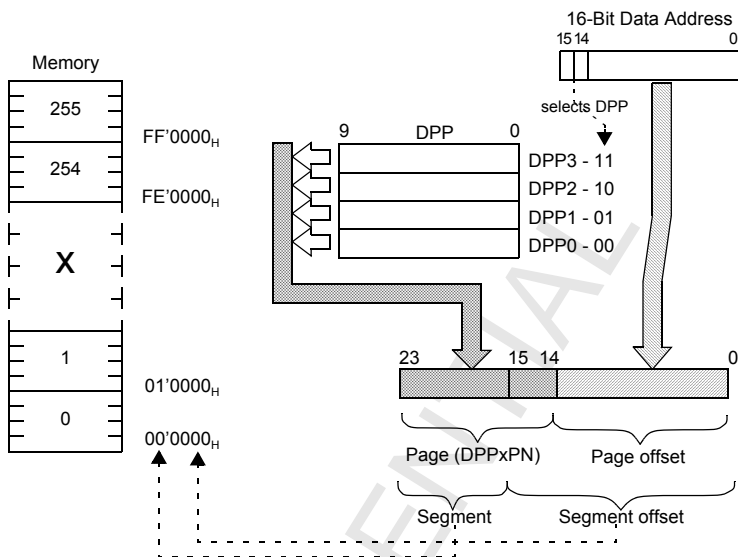
#### 6.3.5.2.1 Addressing via Data Page Pointer

The 4 non-bit-addressable DPP registers select up to 4 different data pages. The lower 10 bits of each DPP register select one of the 1024 possible 16kByte data pages, while the upper 6 bits are reserved for the future use. The DPP registers provide access to the entire memory space in 16kByte pages.

The DPP registers are used implicitly whenever data accesses to any memory location are made via indirect or direct long 16-bit addressing modes (except for override accesses via EXTENDED instructions and PEC data transfers).

Data paging is performed by concatenating the lower 14 bits of an indirect or direct long 16-bit address with the contents of the DPP register selected by the upper 2 bits of the 16-bit address. The contents of the selected DPP register specify one of the 1024 possible data pages. This data page base address together with the 14-bit page offset forms the physical 24-bit address, see [Figure 6-9](#).

Figure 6-9 Addressing via Data Page Pointer





**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

After reset, the DPP registers select data pages 3-0 within segment 0. If the user does not want to use any data paging, no further action is required.

**DPPx**

**DPP0**

**Data Page Pointer 0**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						DPP0PN									
r						rw									

**DPP1**

**Data Page Pointer 1**

**Reset value: 0001<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						DPP1PN									
r						rw									

**DPP2**

**Data Page Pointer 2**

**Reset value: 0002<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						DPP2PN									
r						rw									

**DPP3**

**Data Page Pointer 3**

**Reset value: 0003<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						DPP3PN									
r						rw									

*Note: In a non-segmented memory mode, the whole DPP register is still used for the calculation of the physical 24-bit address.*

A DPP register can be updated via any instruction that is capable of modifying an SFR.

*Note: Due to the internal instruction pipeline, a new DPP value is not usable for the operand address calculation of the instruction immediately following the instruction updating the **DPPx** register.*

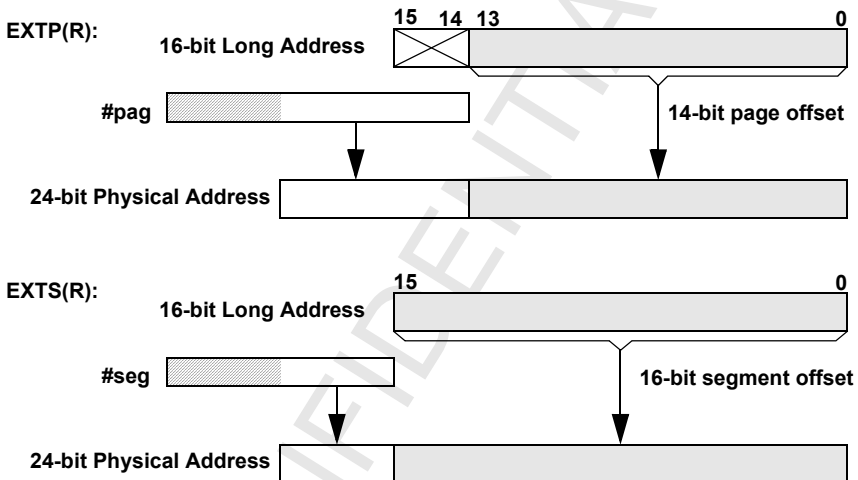
The Reserved bits must be left at their reset value.

### 6.3.5.2.2 DPP Override Mechanism in the C166S

The C166S provides an override mechanism to temporarily bypass the DPP addressing scheme.

The EXTP(R) and EXT(S) instructions override this addressing mechanism. Instruction EXTP(R) replaces the contents of the DPP register, while instruction EXT(S) concatenates the complete 16-bit long address with the specified segment base address. The overriding page or segment may be specified directly as a constant (#pag, #seg) or via a word GPR (Rw).

**Figure 6-10 Overriding the DPP Mechanism**



**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

### 6.3.5.2.3 Long Addressing Mode

The long addressing mode uses a 16-bit constant value encoded in the instruction format which specifies the data page offset and the DPP.

The long addressing mode is referred to by the mnemonic **mem**, refer to [Figure 6-17](#).

**Table 6-17 Long Addressing Mode**

Mnemonic	Physical Address	Scope of Access
<b>mem</b>	(DPP0)    mem^3FFF <sub>H</sub> (DPP1)    mem^3FFF <sub>H</sub> (DPP2)    mem^3FFF <sub>H</sub> (DPP3)    mem^3FFF <sub>H</sub>	any word or byte
<b>mem</b>	pag    mem^3FFF <sub>H</sub>	any word or byte
<b>mem</b>	seg    mem	any word or byte

*Note: The long addressing mode may be used with the DPP overriding mechanism (EXTP(R) and EXTS(R)).*

### 6.3.5.2.4 Indirect Addressing Modes

These addressing modes can be considered as a combination of short and long addressing modes. This means that a long 16-bit address is provided indirectly by the contents of a word GPR that is specified directly by a short 4-bit address (Rw = 0 to 15). There are indirect addressing modes which add a constant value to the GPR contents before the long 16-bit address is calculated. Other indirect addressing modes can decrement or increment the indirect address pointers (GPR contents) by 2 or 1 (referring to words or bytes).

In each case, one of the four DPP registers is used to specify physical 24-bit addresses. Any word or byte data within the entire memory space can be addressed indirectly.

*Note: Indirect addressing may be used with the DPP overriding mechanism (EXTP(R) and EXTS(R)).*

Some instructions use only the lowest 4-word GPRs (R3-R0) as indirect address pointers, which are then specified via short 2-bit addresses.

Physical addresses are generated from indirect address pointers using the following algorithm:

1. Calculate the physical address of the word GPR, which is used as indirect address pointer, using the specified short address (Rw) and

$$\text{GPR Address} = (\text{CP}) + 2 * \text{Short Address}$$

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

2. If required, pre-decrement indirect address pointer (-Rw) by the data-type-dependent value ( $\Delta = 1$  for byte operations,  $\Delta = 2$  for word operations) before the long 16-bit address is generated:

$$(\text{GPR Address}) = (\text{GPR Address}) - \Delta; [\text{optional step}]$$

3. Calculate the long 16-bit address by adding a constant value (Rw+const16 if selected) to the contents of the indirect address pointer:

$$\text{Long Address} = (\text{GPR Pointer}) + \text{Constant}; [\text{the Constant is optional}]$$

4. Calculate the physical 24-bit address using the resulting long address and the corresponding DPP register contents (see long memory addressing modes).

$$\text{Physical Address} = (\text{DPPx}) + \text{Page offset}$$

5. If required, post-in/decrement indirect address pointers (Rw $\pm$ ) by the data-type-dependent value ( $\Delta = 1$  for byte operations,  $\Delta = 2$  for word operations).

$$(\text{GPR Pointer}) = (\text{GPR Pointer}) \pm \Delta; [\text{optional step}]$$

**Table 6-18** provides the indirect addressing modes.

**Table 6-18 Indirect Addressing Modes**

Mnemonic	Comments
[Rw]	Most instructions accept any GPR (R15...R0) as indirect address pointer. Some instructions accept only the lower four GPRs (R3...R0).
[Rw+]	The specified indirect address pointer is automatically post-incremented by 2 or 1 (for word or byte data operations) after the access.
[-Rw]	The specified indirect address pointer is automatically pre-decremented by 2 or 1 (for word or byte data operations) before the access.
[Rw+#data16]	The specified 16-bit constant is added to the indirect address pointer before the long address is calculated.

### 6.3.5.3 The System Stack

A system stack is provided to store return vectors, segment pointers, and processor status for procedures and interrupt routines.

The internal system stack can also be used to store data temporarily, or pass it between subroutines or tasks. Instructions are provided to push or pop registers on/from the system stack. However, in most cases, the register banking scheme provides the best performance for passing data between multiple tasks.

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

*Note: The system stack allows the storage of words only. Bytes must either be converted to words or the “unwanted” other byte must be disregarded.*

*Register **SP** can be loaded only with even byte addresses (The LSB of **SP** is always 0).*

The **Stack Pointer (SP)** addresses the stack within the DPRAM area.

**The Stack Pointer Register**

The non-bit-addressable Stack Pointer (**SP**) register is used to point to the Top Of the System (TOS) stack. The **SP** register is pre-decremented whenever data is to be pushed onto the stack, and it is post-incremented whenever data is to be popped from the stack. Therefore, the system stack grows from higher toward lower memory locations.

Since the Least Significant Bit (LSB) of register **SP** is tied to 0, and bits 15-12 are tied to 1 by hardware, the **SP** register can contain values only from F000<sub>H</sub> to FFFE<sub>H</sub>. This allows access to a physical stack within the DPRAM of the C166S. A virtual stack (usually bigger) can be implemented via software. This mechanism is supported by registers **STKOV** and **STKUN** (refer to [Section 6.3.5.3.1 Stack Overflow and Underflow \(on Page 162\)](#)).

The **SP** register can be updated via any instruction that is capable of modifying a 16-bit SFR.

*Note: Due to the internal instruction pipeline, a POP or RETurn instruction must not immediately follow an instruction updating the **SP** register.*

**SP**

**Stack Pointer**

**Reset value: FC00<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1111				SP											0

Field	Bits	Type	Description
<b>0</b>	0	r	This bit is fixed at 0.
<b>SP</b>	11:1	rwh	Modifiable portion of register <b>SP</b> Specifies the top of the system stack.
<b>1111</b>	15:12	r	These bits are fixed at 1111.

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

### 6.3.5.3.1 Stack Overflow and Underflow

Detection of stack overflow/underflow is supported by two registers, **STKOV** (STack OVerflow pointer) and **STKUN** (STack UNderflow pointer). Specific system traps (Stack Overflow trap, Stack Underflow trap) will be entered whenever the **SP** reaches either boundary specified in these registers.

In many cases, the user will place a Software ReSeT instruction (SRST) into the stack underflow and overflow trap service routines. This is an easy approach that does not require special programming. However, this approach assumes that the defined internal stack is sufficient for the current software, and that exceeding its upper or lower boundary represents a fatal error (refer to [Section 6.3.5.3.2 Linear Stack \(on Page 164\)](#)).

It is also possible to use the stack underflow and stack overflow traps to cache portions of a larger external stack. Only the portion of the system stack currently being used is placed into the internal memory, thus allowing a greater portion of the internal RAM to be used for program, data, or register banking. This approach assumes no error but requires a set of control routines (refer to [Section 6.3.5.3.3 Circular \(Virtual\) Stack \(on Page 164\)](#)).

#### The STack Overflow Pointer Register

This non-bit-addressable **STKOV** pointer register is compared to the **SP** register after each operation that pushes data onto the system stack (for example, PUSH and CALL instructions or interrupts), and after each subtraction from the **SP** register. If the contents of the **SP** register is less than the contents of the **STKOV** pointer register, a stack overflow trap occurs.

#### STKOV

##### STack Overflow Pointer

**Reset value: FA00<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1111				STKOV											
															0

Field	Bits	Type	Description
<b>0</b>	0	r	This bit is fixed at 0.
<b>STKOV</b>	11:1	rw	Modifiable portion of register <b>STKOV</b> Specifies the segment offset address of the lower limit of the system stack.
<b>1111</b>	15:12	r	These bits are fixed at 1111.

**STKOV** can be updated via any instruction that is capable of modifying an SFR.

**CONFIDENTIAL**

## Central Processing Unit, PEC, and Interrupt

*Note: When a value is MOVED into the stack pointer (**SP**), NO check against the overflow/registers is performed.*

- **Fatal error indication** treats the stack overflow as a system error and executes associated trap service routine. Under these circumstances, data in the bottom of the stack may have been overwritten by the status information stacked upon servicing the stack overflow trap.
- **Automatic system stack flushing** allows the system stack to be used as a “Stack Cache” for a bigger external user stack. In this case, **STKOV** should be initialized to a value that represents the desired lowest Top Of Stack (TOS) address plus an offset based on the selected maximum stack size. This offset considers the worst case that will occur when a stack overflow condition is detected just during entry into an ISR, or during an ATOMIC/EXTend sequence. Under these conditions, additional stack word locations are required to push **IP**, **PSW**, and **CSP** for both the ISR and the hardware trap service routine.

### The STaCK Underflow Pointer Register

**STKUN** is a non-bit-addressable register that is compared to the **SP** register after each operation that pops data from the system stack (for example, POP and RET instructions), and after each addition to the **SP** register. If the content of the **SP** register is greater than the content of **STKUN**, a stack underflow hardware trap will occur.

Since the LSB of **STKUN** is tied to 0 and bits 15 through 12 are tied to 1 by hardware, **STKUN** register can only contain values from F000<sub>H</sub> to FFFE<sub>H</sub>.

#### STKUN

#### STaCK Underflow Pointer

**Reset value: FC00<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1111				STKUN											0

Field	Bits	Type	Description
<b>0</b>	0	r	This bit is fixed at 0.
<b>STKUN</b>	11:1	rw	Modifiable portion of register <b>STKUN</b> Specifies the segment offset address of the upper limit of the system stack.
<b>1111</b>	15:12	r	These bits are fixed at 1111.

**STKUN** can be updated via any instruction capable of modifying an SFR.

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

*Note: When a value is MOVED into the stack pointer, NO check against the overflow/registers is performed.*

- **Fatal error indication** treats the stack underflow as a system error and executes associated trap service routine.
- **Automatic system stack refilling** allows the system stack to be used as a “Stack Cache” for a bigger external user stack. In this case, **STKUN** should be initialized to a value that represents the desired highest Bottom of Stack address.

### **Scope of Stack Limit Control**

The stack limit control by the register pair **STKOV** and **STKUN** detects cases where **SP** is moved outside the defined stack area either by ADD or SUB instructions, or by PUSH or POP operations (explicit or implicit, for example, CALL or RET instructions).

This control mechanism is not triggered and no stack trap is generated when either:

- The stack pointer **SP** is directly updated via the MOV instructions
- The limits of the stack area (**STKOV**, **STKUN**) are changed so that **SP** is outside the new limits.

#### **6.3.5.3.2 Linear Stack**

The C166S offers a linear stack option ( $STKSZ = 111_B$ ) in which the system stack may use the complete DPRAM area. This provides a large system stack without requiring procedures to handle data transfers for a circular stack. However, this method also leaves less RAM space for variables or code. The DPRAM area that may be consumed by the system stack is defined via the **STKUN** and **STKOV** pointers. The underflow and overflow traps in this case serve for fatal error detection only.

For the linear stack option, all modifiable bits of register **SP** are used to access the physical stack. Although the stack pointer (**SP**) may cover addresses from  $00\ F000_H$  up to  $00\ FFFE_H$ , the (physical) system stack must be located within the DPRAM and therefore may only use the address range  $00\ F600_H$  to  $00\ FDFE_H$ . It is the responsibility of the user to restrict the system stack to the DPRAM range.

*Note: Stack accesses below the DPRAM area (ESFR space and reserved area) and within address range  $00\ FE00_H$  and  $00\ FFFE_H$  (SFR space) will have unpredictable results.*

#### **6.3.5.3.3 Circular (Virtual) Stack**

This basic technique allows pushing until the overflow boundary of the internal stack is reached. At this point, a portion of the stacked data must be saved into external memory to create space for further stack pushes. This is called “stack flushing”. When executing a number of return or pop instructions, the upper boundary (since the stack empties upward to higher memory locations) is reached. The entries that have been previously



**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

saved in external memory must now be restored. This is called “stack filling”. Because procedure call instructions do not continue to nest infinitely and call and return instructions alternate, flushing and filling normally occurs very infrequently. If this is not true for a given program environment, this technique should not be used because of the overhead of flushing and filling.

**The basic mechanism** is the transformation of the addresses of a virtual stack area controlled via **SP**, **STKOV**, and **STKUN** to a defined physical stack area within the DPRAM via hardware. This virtual stack area covers all possible locations that **SP** can point to, that is, 00 F000<sub>H</sub> through 00 FFE<sub>H</sub>. **STKOV** and **STKUN** accept the same 4kByte address range. The size of the physical stack area within the DPRAM that is used for standard stack operations is defined via bitfield **SYSCON.STKSZ** (refer to [Table 6-19](#)).

**Table 6-19 Circular Stack Address Transformation**

<b>STKSZ</b>	<b>Stack Size (Words)</b>	<b>DPRAM Addresses (Words) of Physical Stack</b>	<b>Significant Bits of <b>SP</b></b>
000 <sub>B</sub>	256	00 FBFE <sub>H</sub> -00 FA00 <sub>H</sub> (Default after Reset)	SP.8-SP.0
001 <sub>B</sub>	128	00 FBFE <sub>H</sub> -00 FB00 <sub>H</sub>	SP.7-SP.0
010 <sub>B</sub>	64	00 FBFE <sub>H</sub> -00 FB80 <sub>H</sub>	SP.6-SP.0
011 <sub>B</sub>	32	00 FBFE <sub>H</sub> -00 FBC0 <sub>H</sub>	SP.5-SP.0
100 <sub>B</sub>	512	00 FBFE <sub>H</sub> -00 F800 <sub>H</sub> (not for 1kByte DPRAM)	SP.9-SP.0
101 <sub>B</sub>	---	Reserved. Do not use this combination.	---
110 <sub>B</sub>	---	Reserved. Do not use this combination.	---
111 <sub>B</sub>	1024	00 FDFE <sub>H</sub> -00 FX00 <sub>H</sub> <i>Note: No circular stack</i> 00 FX00 <sub>H</sub> represents the lower DPRAM limit, that is, 1kB: 00 FA00 <sub>H</sub> , 2kB: 00 F600 <sub>H</sub> ,	SP.11...SP.0

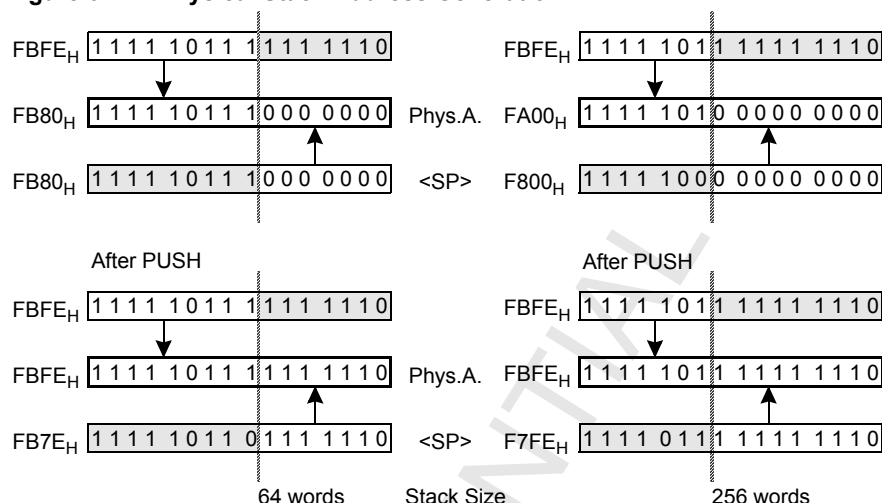
The virtual stack addresses are transformed to physical stack addresses by concatenating the significant bits of **SP** (refer to [Table 6-19](#)) with the complementary most significant bits of the upper limit of the physical stack area (00 FBFE<sub>H</sub>). This transformation is done via hardware (see [Figure 6-11](#)).

The reset values (**STKOV** = FA00<sub>H</sub>, **STKUN** = FC00<sub>H</sub>, **SP** = FC00<sub>H</sub>, **STKSZ** = 000<sub>B</sub>) map the virtual stack area directly to the physical stack area, and allow using the internal system stack without any changes, provided that the 256-word area is not exceeded.

CONFIDENTIAL

Central Processing Unit, PEC, and Interrupt

Figure 6-11 Physical Stack Address Generation



The following example demonstrates the circular stack mechanism that is also an effect of this virtual stack mapping: First, register R1 is pushed onto the lowest physical stack location according to the selected maximum stack size. The next instruction pushes register R2 onto the highest physical stack location, although the **SP** is decremented by 2 as for the previous push operation.

```
MOV      SP, #0F802H ;Set SP before last entry of physical stack of
                        ;256 words
...
PUSH     R1           ;(SP) = F802H: Physical stack address = FA02H
PUSH     R2           ;(SP) = F800H: Physical stack address = FA00H
                        ;(SP) = F7FEH: Physical stack address = FBFEH
```

The effect of the address transformation is that the physical stack addresses wrap around from the end of the defined area to its beginning. When flushing and filling the internal stack, this circular stack mechanism only requires moving that portion of stack data that is to be re-used (that is, the upper part of the defined stack area) instead of the whole stack area. Stack data that remain in the lower part of the internal stack need not be moved by the distance of the space being flushed or filled, as the stack pointer (**SP**) automatically wraps around to the beginning of the freed part of the stack area.

*Note: This circular stack technique is applicable for stack sizes of 32 to 512 words ( $STKSZ = 000_B$  to  $100_B$ ). It does not work with option  $STKSZ = 111_B$ , which uses the complete DPRAM for system stack; in this case, the address transformation mechanism is deactivated.*

When a boundary is reached, the stack underflow or overflow trap is entered. Inside the trap handler a predetermined portion of the internal stack is moved to or from the external

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

stack. The amount of data transferred is determined by the average stack space required by routines and the frequency of calls, traps, interrupts, and returns. In most cases, this will be approximately 1/4 to 1/10 the size of the internal stack. Once the transfer is complete, the boundary pointers are updated to reflect the newly allocated space on the internal stack. Thus, the user is free to write code without concern for the internal stack limits. Only the execution time required by the trap routines affects user programs.

The following procedure initializes the controller for usage of the circular stack mechanism:

1. Specify the size of the physical system stack area in the DPRAM (bitfield **SYSCON.STKSZ**).
2. Define two pointers that specify the upper and lower boundary of the external stack. These values are then tested in the stack underflow and overflow trap routines when moving data.
3. Set **STKOV** to the limit of the defined internal stack area plus six words (for the reserved space to store two interrupt entries).

The internal stack will now fill until the overflow pointer is reached. After entry into the overflow trap procedure, the top of the stack will be copied to the external memory. The internal pointers will then be modified to reflect the newly-allocated space. After exiting the trap procedure, the internal stack will wrap around to the top of the internal stack, and continue to grow until the new value of the stack overflow pointer is reached.

When the underflow pointer is reached, while the stack is emptied, the bottom of stack is reloaded from the external memory, and the internal pointers are adjusted accordingly.

### **6.3.6 Data Processing**

All standard arithmetic, shift, and logical operations are performed in the 16-bit Arithmetic and Logic Unit (ALU). In addition to the standard ALU, the ALU of the C166S includes bit manipulation, and a multiply-and-divide unit. Most internal execution blocks have been optimized to perform operations on either 8-bit or 16-bit numbers. Once the pipeline has been filled, most instructions are completed in one machine cycle.

The status flags are automatically updated in the PSW register (refer to **PSW (on Page 176)**) after each ALU operation. These flags allow branching upon specific conditions. Support of both signed and unsigned arithmetic is provided by the user-selectable branch test. The status flags are also preserved automatically by the MCU upon entry into an interrupt or trap routine.

#### **6.3.6.1 Data Types**

The C166S supports operations on boolean/bit, bit string, character, and integer data types. Most instructions operate with specific data types, while others are useful for manipulating several data types.

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

The C166S data formats support all ANSI C data types. In addition, some C compilers support new types that allow the efficient use of the bit-manipulation instructions in embedded control applications.

The C166S directly supports the following data formats in [Table 6-20](#) and [Table 6-21](#).

**Table 6-20 MCU Data Formats**

MCU data format	Size	Range
BIT	1 bit	0 or 1
BYTE	1 byte	0 to 255U or -128 to +127
WORD	2 bytes	0 to 65535U or -32768 to +32767

**Table 6-21 ANSI C Data Types**

ANSI C Data Types	Size	Range	MCU Data Format
bit	1 bit	0 or 1	BIT
sfrbit	1 bit	0 or 1	BIT
esfrbit	1 bit	0 or 1	BIT
signed char	1 byte	-128 to +127	BYTE
unsigned char	1 byte	0 to 255U	BYTE
sfr	1 byte	0 to 65535U	WORD
esfr	1 byte	0 to 65535U	WORD
signed short	2 bytes	-32768 to +32767	WORD
unsigned short	2 bytes	0 to 65535U	WORD
bitword	2 bytes if word	0 to 65535U	WORD or BIT
signed int	2 bytes	-32768 to +32767	WORD
unsigned int	2 bytes	0 to 65535U	WORD
signed long	4 bytes	-2147483648 to +2147483647	Not directly supported
unsigned long	4 bytes	0 to 4294967295UL	Not directly supported
float	4 bytes	+/-1,176E-38 to +/-3,402E+38	Compiler dependent
double	8 bytes	+/- 2,225E-308 to +/- 1,797E+308	Compiler dependent

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

**Table 6-21 ANSI C Data Types**

ANSI C Data Types	Size	Range	MCU Data Format
long double	8 bytes	+/- 2,225E-308 to +/- 1,797E+308	Compiler dependent
near pointer	2 bytes	16/14bits depending on memory model	WORD
far pointer	4 bytes	14bits (16k) in any page	Compiler dependent
huge pointer	4 bytes	24bits (16M)	Compiler dependent
shuge pointer	4 bytes	24bits (16M), but arithmetic is done 16-bit wide	Compiler dependent

### 6.3.6.2 Constants

In addition to the addressing modes the C166S instruction set also supports word-wide or byte-wide immediate constants. For an optimum utilization of the available code storage, these constants are represented in the instruction formats by either 3, 4, 8, or 16 bits. The short constants are always zero-extended, while the long constants are truncated if necessary to match the data format required for the particular operation (refer to [Table 6-22](#)).

**Table 6-22 Constant Formats**

Mnemonic	Word Operation	Byte Operation
#data3	$0000_H + \text{data3}$	$00_H + \text{data3}$
#data4	$0000_H + \text{data4}$	$00_H + \text{data4}$
#data8	$0000_H + \text{data8}$	data8
#data16	data16	$\text{data16} \wedge FF_H$
#mask	$0000_H + \text{mask}$	mask

*Note: Immediate constants are always signified by a leading #.*

### 6.3.6.3 The 16-bit Adder/Subtractor, Barrel Shifter, and 16-bit Logic Unit

All standard arithmetic and logical operations are performed by a 16-bit ALU. In case of byte operations signals from bits six and seven of the ALU result are used to control the condition flags. Multiple precision arithmetic is supported by a CARRY-IN signal to the ALU from previously calculated portions of the desired operation.

CONFIDENTIAL

Central Processing Unit, PEC, and Interrupt

A 16-bit barrel shifter provides multiple bit shifts in a single machine cycle. Rotations and arithmetic shifts are also supported.

#### 6.3.6.4 Bit-manipulation Unit

The C166S offers a large number of instructions for bit processing. The special bit-manipulation unit was implemented for this purpose. The bit-manipulation instructions are for efficient control and testing of peripherals. Unlike other microcontrollers, the C166S has instructions that provide direct access to two operands in the bit-addressable space without requiring them to be moved into temporary locations.

The same logical instructions that are available for words and bytes can also be used for bits. The user can compare and modify a control bit for a peripheral in one instruction. Multiple-bit shift instructions have been included to avoid long instruction streams of single bit shift operations. These instructions require a single machine cycle. In addition, bit field instructions are able to modify multiple bits of one operand in a single instruction.

All instructions that manipulate single bits or bit groups internally use a read-modify-write sequence that accesses the whole word, which contains the specified bit(s).

This method has several consequences:

- Bits can only be modified within the internal address areas, that is, DPRAM and SFRs. External locations cannot be used with bit instructions. The upper 256 bytes of the SFR area, the ESFR area, and the DPRAM are bit-addressable (refer to [Section 6.4.3 Memory Organization \(on Page 192\)](#)), that is, those register bits located within the respective sections can be manipulated directly using bit instructions. The other SFRs must be accessed byte- or word-wise.

*Note: All GPRs are bit-addressable independent of the allocation of the register bank via the context pointer CP. Even GPRs which are allocated to not bit-addressable RAM locations provide this feature.*

- The read-modify-write approach may be critical with hardware-effected bits. In these cases, the hardware may change specific bits while the read-modify-write operation is in progress, where the write back would overwrite the new bit value generated by the hardware. The solution is either to use the implemented hardware protection (below) or through special programming (refer to [Section 6.3.7 Instruction Pipeline \(on Page 180\)](#)).

**Protected bits** are not changed during the read-modify-write sequence, i.e., when hardware sets an interrupt request flag between the read and the write of the read-modify-write sequence, for example. The hardware protection logic guarantees that only the intended bit(s) is/are affected by the write-back operation.

*Note: If a conflict occurs between a bit manipulation generated by hardware and an intended software access, the software access has priority and determines the final value of the bit.*

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

### 6.3.6.5 Multiply and Divide Unit

The C166S has a separated multiply-and-divide unit. The multiplication is executed within five machine cycles, while a division takes 20 machine cycles. The multiply-and-divide process is interruptible by an interrupt that has a higher priority level than the current MCU level.

#### The Multiply/Divide High Word Register

The non-bit-addressable Multiply/Divide High Word register (**MDH**) that contains the upper bits of the 32-bit Multiply/Divide (MD) factor, which is used by the MCU when it performs a multiplication or a division using implicit addressing (DIV, DIVL, DIVLU, DIVU, MUL, MULU). After an implicitly-addressed multiplication, this register represents the high-order 16 bits of the 32-bit result. For long divisions, **MDH** must be loaded with the high-order 16 bits of the 32-bit dividend before the division has started. After any division, **MDH** represents the 16-bit remainder.

#### MDH

##### Multiply/Divide High Word

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MDH															

Field	Bits	Type	Description
<b>MDH</b>	15:0	rwh	<b>High Bits of the MD</b> The high order 16 bits of the 32-bit multiply and divide register MD.

Whenever this register is updated via software, the Multiply/Divide Register In Use (**MDC.MDRIU**) flag in the Multiply/Divide Control (**MDC**) register is set to 1.

When a multiplication or division is interrupted before its completion and when a new multiply or divide operation is to be performed within the Interrupt Service Routine (ISR), the contents of **MDH** must be saved along with the contents of registers **MDL** and **MDC** to avoid erroneous results.

#### Multiply/Divide Low Word Register

The non-bit-addressable Multiply/Divide Low Word register (**MDL**) contains the low word of the 32-bit multiply/divide (MD) factor which is used by the MCU when it performs a multiplication or a division using implicit addressing (DIV, DIVL, DIVLU, DIVU, MUL, MULU). After a multiplication, this register represents the low-order 16 bits of the 32-bit result. For long divisions, the **MDL** register must be loaded with the low-order 16 bits of

**CONFIDENTIAL**

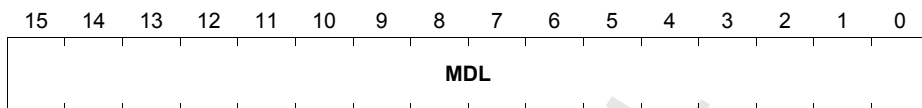
## Central Processing Unit, PEC, and Interrupt

the 32-bit dividend before the division has started. After any division, **MDL** represents the 16-bit quotient.

### **MDL**

#### **Multiply/Divide Low Word**

**Reset value: 0000<sub>H</sub>**



Field	Bits	Type	Description
<b>MDL</b>	15:0	rwh	<b>Low Part of the MD</b> The low order 16 bits of the 32-bit multiply and divide register MD.

Whenever this register is updated via software, the **MDC.MDRIU** flag is set to 1. The **MDC.MDRIU** flag is cleared whenever the **MDL** register is read via software.

When a multiplication or division is interrupted before its completion and when a new multiply or divide operation is to be performed within the ISR, the contents of **MDL** must be saved along with the contents of registers **MDH** and **MDC** to avoid erroneous results.



**CONFIDENTIAL**

## Central Processing Unit, PEC, and Interrupt

### Multiply/Divide Control Register

The bit-addressable 16-bit Multiply/Divide Control (**MDC**) register is implicitly used by the MCU, when it performs a multiplication or a division. It is used to store the required control information for the corresponding multiply or divide operation. **MDC** is updated by hardware during each single cycle of a multiply or divide instruction.

#### MDC

#### Multiply/Divide Control

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								IMS2		MDRIU	IMS1				

Field	Bits	Type	Description
IMS1	3:0	rwh	<b>Internal Machine Status 1</b> The multiply/divide unit uses these bits to control internal operations. Never modify these bits without saving and restoring register <b>MDC</b>
MDRIU	4	rwh	<b>Multiply/Divide Register In Use</b> 0: Cleared when register <b>MDL</b> is read via software. 1: Set when register <b>MDL</b> or <b>MDH</b> is written via software, or when a multiply or divide instruction is executed.
IMS2	7:5	rwh	<b>Internal Machine Status 2</b> The multiply/divide unit uses these bits to control internal operations. Never modify these bits without saving and restoring register <b>MDC</b>
RESERVED	15:8	r	Reserved, these bits must be left at their reset values.

When a division or multiplication was interrupted before its completion and the multiply/divide unit is required, the **MDC** register must first be saved along with registers **MDH** and **MDL** (to be able to restart the interrupted operation later), and then it must be cleared for the new calculation. After completion of the new division or multiplication, the state of the interrupted multiply or divide operation must be restored.

The **MDC.MDRIU** flag is the only portion of the **MDC** register that might be of interest for the user. The remaining portions of the **MDC** register are reserved for dedicated use by the hardware, and should never be modified by the user other than described above. Otherwise, correct continuation of an interrupted multiply or divide operation cannot be guaranteed.

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

Multiplication or division is performed simply by specifying the correct (signed or unsigned) version of the multiply or divide instruction. The result is then stored in register MD. The overflow flag (V) is set if the result of a multiply or divide instruction is greater than 16 bits. This flag can be used to determine whether both word halves must be transferred from register MD. The high portion of MD (**MDH**) must be moved into the register file or memory first. To ensure that the **MDC.MDRIU** flag reflects the correct state.

The following instruction sequence performs an unsigned 16-by-16-bit multiplication:

```
SAVE:
JNB      MDRIU, START    ;Test if MD was in use.
SCXT     MDC, #0010H     ;Save and clear control register, leaving MDRIU
                                ; set (only required for interrupted
                                ;multiply/divide instructions)
BSET     SAVED           ;Indicate the save operation
PUSH     MDH             ;Save previous MD contents...
PUSH     MDL             ;...on system stack
START:
MULU     R1, R2           ;Multiply 16-16 unsigned, Sets MDRIU
JMPR     cc_NV, COPYL    ;Test for only 16-bit result
MOV      R3, MDH         ;Move high portion of MD
COPYL:
MOV      R4, MDL         ;Move low portion of MD, Clears MDRIU
RESTORE:
JNB      SAVED, DONE     ;Test if MD registers were saved
POP      MDL             ;Restore registers
POP      MDH
POP      MDC
BCLR     SAVED           ;Multiplication is completed,
                                ;program continues
DONE:    ...
```

The above save sequence and the restore sequence after **COPYL**: are required only if the current routine could have interrupted a previous routine that contained a **MUL** or **DIV** instruction. Register **MDC** is also saved because it is possible that a previous routine Multiply or Divide instruction was interrupted while in progress. In this case, the information about how to restart the instruction is contained in this register. Register **MDC** must be cleared to be initialized correctly for a subsequent multiplication or division. The old **MDC** contents must be popped from the stack before the **RETI** instruction is executed.

For a division, the user must first move the dividend into the MD register. If a 16/16-bit division is specified, only the low portion of MD must be loaded. The result is also stored in MD. The low portion (**MDL**) contains the integer result of the division, while the high portion (**MDH**) contains the remainder.

**CONFIDENTIAL****Central Processing Unit, PEC, and Interrupt**

The following instruction sequence performs a 32-by-16-bit division:

```
MOV      MDH, R1      ;Move dividend to MD register. Sets MDRIU
MOV      MDL, R2      ;Move low portion to MD
DIV      R3           ;Divide 32/16 signed, R3 holds divisor
JMPR     cc_V, ERROR  ;Test for divide overflow
MOV      R3, MDH      ;Move remainder to R3
MOV      R4, MDL      ;Move integer result to R4. Clears MDRIU
```

Whenever a multiply or divide instruction is interrupted while in progress, the address of the interrupted instruction is pushed onto the stack and the **PSW.MULIP** flag of the interrupting routine is set. When the interrupt routine is exited with the RETI instruction, this bit is tested implicitly before the old **PSW** is popped from the stack. If **PSW.MULIP** = 1, the multiply/divide instruction is re-read from the location popped from the stack (return address) and will be completed after the RETI instruction has been executed.

*Note: The **PSW.MULIP** flag is part of the context of the interrupted task. When the interrupting routine does not return to the interrupted task (for example, scheduler switches to another task), **PSW.MULIP** must be set or cleared according to the context of the task that is switched to.*

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

### 6.3.6.6 Processor Status Word Register (PSW)

The bit-addressable Processor Status Word register reflects the current status of the microcontroller. Two groups of bits represent the current ALU status and the current MCU interrupt status. One separate bit (USR0) within **PSW** is provided as a general-purpose flag.

#### PSW

#### Processor Status Word

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ILVL				IEN	S1	RESERVED			USR 0	MUL IP	E	Z	V	C	N

Field	Bits	Type	Description
<b>N</b>	0	rwh	<b>Negative Result</b> 0 ALU result is not negative 1 ALU result is negative
<b>C</b>	1	rwh	<b>Carry Flag</b> 0 No carry/borrow bit produced 1 Carry/borrow bit produced
<b>V</b>	2	rwh	<b>oVerflow Flag</b> 0 No overflow produced 0 Overflow produced
<b>Z</b>	3	rwh	<b>Zero Flag</b> 0 ALU result is not zero 1 ALU result is zero
<b>E</b>	4	rwh	<b>End of Table Flag</b> 0 Source operand is neither 8000 <sub>h</sub> nor 80 <sub>h</sub> 1 Source operand is 8000 <sub>h</sub> or 80 <sub>h</sub>
<b>MULIP</b>	5	rh	<b>MULTiPlication/division In Progress</b> 0 No multiplication/division in process 1 Multiplication/division has been interrupted
<b>USR0</b>	6	rw	<b>General Purpose Flag</b> May be used by an application.
<b>S1</b>	10	rw	<b>Reserved for System</b>
<b>IEN</b>	11	rw	<b>Interrupt/PEC ENable Bit (global)</b> 0 Interrupt/PEC requests are disabled 1 Interrupt/PEC requests are enabled

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

Field	Bits	Type	Description
ILVL	15:12	rwh	<b>MCU Priority LeVeL</b> 0 <sub>H</sub> Lowest priority ... F <sub>H</sub> Highest priority
RESERVED	9:7	r	Reserved, these bits must be left at their reset values.

### ALU Status Flags (N, C, V, Z, E, MULIP)

The condition flags (**N, C, V, Z, E**) in **PSW** indicate the ALU status resulting from the ALU operation last performed. They are set by the majority of instructions according to specific rules depending on the ALU operation or data movement.

After execution of an instruction that explicitly updates **PSW**, the condition flags may no longer represent an actual MCU status. An explicit write operation to **PSW** supersedes the condition flag values that are implicitly generated by the MCU. An explicit read access to **PSW** returns the value of **PSW** after execution of the previous instruction.

*Note: After reset, all of the ALU status bits are cleared.*

- **N Flag:**

For the majority of ALU operations, **PSW.N** is set to 1 if the most significant bit of the result contains a 1. Otherwise, it is cleared. For integer operations, **PSW.N** can be interpreted as the sign bit of the result (negative: **PSW.N** = 1, positive: **PSW.N** = 0). Negative numbers are always represented as the 2's complement of the corresponding positive number. The range of signed numbers extends from  $-8000_H$  to  $+7FFF_H$  for the word data type, or from  $-80_H$  to  $+7F_H$  for the byte data type. For Boolean bit operations with only one operand, **PSW.N** represents the previous state of the specified bit. For Boolean bit operations with two operands, the **PSW.N** represents the logical XORing of the two specified bits.

- **C Flag:**

After an addition, **PSW.C** indicates that a “carry” from the most significant bit of the specified word or byte data type has been generated. After a subtraction or a comparison, **PSW.C** indicates a “borrow,” which represents the logical negation of a carry for the addition. This means that **PSW.C** is set to 1 if **no** carry from the Most Significant Bit (MSB) of the specified word or byte data type has been generated during a subtraction. Subtraction is performed by the ALU as a 2's-complement addition. The **PSW.C** is cleared when this complement addition caused a carry. **PSW.C** is always cleared for logical, multiply and divide ALU operations, because these operations can not cause a carry flag to be set.

For shift and rotate operations, **PSW.C** represents the value of the bit shifted out last. If a shift count of zero is specified, **PSW.C** is cleared. **PSW.C** is also cleared for a prioritize operation because a 1 is never shifted out of the MSB during the normalization of an operand.

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

For Boolean bit operations with only one operand, **PSW.C** is always cleared. For Boolean bit operations with two operands, **PSW.C** represents the logical ANDing of the two specified bits.

- **V Flag:**

The addition, subtraction, and 2's complement operations set **PSW.V** to 1 if the result exceeds the range of 16-bit signed numbers for word operations ( $-8000_H$  to  $+7FFF_H$ ), or 8-bit signed numbers for byte operations ( $-80_H$  to  $+7F_H$ ). Otherwise, **PSW.V** is cleared. If **PSW.V** indicates an arithmetic overflow, the result of an integer addition, integer subtraction, or 2's complement operation is not valid.

For multiplication and division, if **PSW.V** indicates an arithmetic overflow is set to 1 if the result can not be represented in a word data type; otherwise, it is cleared. A division by zero will always cause an overflow. Unlike the division result, the result of multiplication is valid regardless of the **PSW.V** value.

Since logical ALU operations cannot produce an invalid result, **PSW.V** is cleared by these operations.

**PSW.V** is also used as a "sticky bit" for rotate-right and shift-right operations. By only using **PSW.C**, a rounding error caused by a shift-right operation can be estimated as up to one half of the LSB of the result. In conjunction with **PSW.V**, **PSW.C** allows the rounding error to be evaluated with a finer resolution (see table below).

For Boolean bit operations with only one operand, the V flag is always cleared. For Boolean bit operations with two operands, the V flag represents the logical ORing of the two specified bits.

**Table 6-23 Shift Right Rounding Error Evaluation**

<b>C flag</b>	<b>V flag</b>	<b>Rounding Error Quantity</b>
0	0	No rounding error
0	1	$0 < \text{Rounding error} < \frac{1}{2} \text{ LSB}$
1	0	Rounding error = $\frac{1}{2} \text{ LSB}$
1	1	Rounding error = $\frac{1}{2} \text{ LSB}$

- **Z flag:**

The Z flag is normally set to 1 if the result of an ALU operation equals zero; otherwise it is cleared.

For addition and subtraction with carry, the Z flag is set to 1 only if the Z flag already contains a 1 as a result of a previous operation, and if the result of the current ALU operation equals zero. This mechanism supports multiple precision calculations.

For Boolean bit operations with only one operand, the Z flag represents the logical negation of the previous state of the specified bit. For Boolean bit operations with two operands, the Z flag represents the logical NORing of the two specified bits. For the prioritize operation, the Z flag indicates whether or not the second operand was zero.

**CONFIDENTIAL****Central Processing Unit, PEC, and Interrupt**

- **E flag:**

End of table flag. The E flag can be altered by instructions that perform ALU or data movement operations. The E flag is cleared by those instructions that can not be reasonably used for table search operations. In all other cases, the E flag value depends on the value of the source operand to signify whether or not the end of a search table is reached. If the value of the source operand of an instruction equals the lowest negative number that depends on the data format of the corresponding instruction (8000<sub>H</sub> for the word data type, or 80<sub>H</sub> for the byte data type), the E flag is set to 1; otherwise it is cleared.

- **MULIP flag:**

The MULIP flag will be set to 1 by hardware upon the entrance into an ISR when a multiply or divide ALU operation was interrupted before completion. Depending on the state of the MULIP bit, the hardware decides whether a multiplication or division must be continued or not after the end of an interrupt service. The MULIP bit is overwritten with the contents of the stacked MULIP flag when RETurn-from-Interrupt-instruction (RETI) is executed. This normally means that the MULIP flag is cleared again after that.

*Note: The MULIP flag is a part of the task environment. When the ISR does not return to the interrupted multiply/divide instruction (for example, in case of a task scheduler that switches between independent tasks), the MULIP flag must be saved as part of the task environment and must be updated accordingly for the new task before this task is entered.*

**MCU Interrupt Status (IEN, ILVL)**

The Interrupt ENable (IEN) bit makes it possible to enable (IEN = 1) or disable (IEN = 0) interrupts globally. The four-bit LeVeL field (ILVL) specifies the priority of the current MCU activity. The priority level is updated by hardware upon entry into an ISR, but it can also be modified via software to prevent other interrupts from being acknowledged. If an priority level 15 has been assigned to the MCU, it has the highest possible priority, and thus the current MCU operation cannot be interrupted except by hardware traps or external non-maskable interrupts. For details, please refer to [Section 6.3.3 Interrupt and Exception Execution \(on Page 111\)](#).

After reset, all interrupts are disabled globally, and the lowest priority (ILVL = 0) is assigned to the initial MCU activity.

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

### **6.3.7 Instruction Pipeline**

The instruction pipeline of the C166S partitions instruction processing into four stages. Each of these has a separate task:

**1. FETCH:**

In this stage, the instruction selected by the Instruction Pointer (**IP**) and the Code Segment Pointer (**CSP**) is fetched from either the internal local memory, DPRAM, or external memory.

**2. DECODE:**

In this stage, the instructions are decoded and, if required, the operand addresses are calculated and the respective operands are fetched. For all instructions that implicitly access the system stack, the **SP** register is either decremented or incremented, as specified. For branch instructions, the **IP** and the **CSP** are updated with the desired branch target address (provided that the branch is taken).

**3. EXECUTE:**

In this stage, an operation is performed on the previously fetched operands in the ALU. Additionally, the condition flags in the PSW register are updated as specified by the instruction. All explicit writes to the SFR memory space, and all auto-increment or auto-decrement writes to GPRs used as indirect address pointers, are performed during the execute stage of the instruction.

**4. WRITE BACK:**

In this stage, all external operands and the remaining operands within the DPRAM space are written back.

The C166S has “injected instructions.” These injected instructions are generated internally by the machine to provide the time needed to process instructions that cannot be processed within one machine cycle. They are automatically injected into the decode stage of the pipeline, and then they pass through the remaining stages like every standard instruction. Program interrupts are also performed by means of injected instructions.

Although these internally injected instructions are not, in reality, noticeable, they are introduced here to ease the explanation of the pipeline.

#### **6.3.7.1 Particular Pipeline Effects**

Since up to 4 different instructions are processed simultaneously, additional hardware has been included in the C166S to prevent a loss of performance when dealing with all causal dependencies on instructions in different pipeline stages. This extra hardware (for example, for “forwarding” operand read and write values) resolves most of the possible conflicts (for example, multiple usage of buses) in a time-optimized way, preventing the pipeline dependencies from becoming noticeable to the user in most cases. However, in some very rare cases, attention from the programmer is required specifically because the C166S is a pipelined machine. In these cases, the delays caused by pipeline conflicts can be used for other instructions to optimize performance.



**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

### Address Pointer Updating

Indirect addressing modes use a GPR value to generate the address of the source and/or destination operand. If this GPR is updated explicitly by the preceding instruction, one NOP instruction is automatically inserted.

```
In      :ADD R0,#0002h      ;increment address pointer GPR 0
Iinject :NOP                ;automatically injected NOP
In+1    :MOV R2,[R0]        ;use GPR 0 for indirect addressing
```

To improve performance, an instruction not using this new GPR as a destination operand can be inserted between an explicit GPR-changing and a subsequent instruction using an indirect addressing mode.

```
In      :ADD R0,#0002h      ;increment address pointer GPR 0
In+1    :ADD R2,R0          ;must not be an instruction updating GPR 0
In+2    :MOV R2,[R0]        ;use GPR 0 for indirect addressing
```

### Context Pointer Updating

An instruction that calculates a physical GPR operand address via the **CP** register is incapable of using a new **CP** value that is to be updated by the preceding instruction. Thus, to make sure that the new **CP** value is used, at least one instruction must be inserted between an instruction that changes the **CP** and a subsequent instruction that uses the GPR, as shown in the following example:

```
In      :SCXT CP,#0FC00h    ;select a new context
In+1    :....              ;must not be an instruction using a GPR
In+2    :MOV R0,#dataX      ;write to GPR 0 in the new context
```

### Data Page Pointer Updating

An instruction that calculates a physical operand address via a particular **DPP<sub>x</sub>** (x = 0 to 3) register is not capable of using a new **DPP<sub>x</sub>** register value that is to be updated by the preceding instruction. Thus, to make sure that the new **DPP<sub>x</sub>** register value is used, at least one instruction must be inserted between a **DPP<sub>x</sub>**-changing instruction and a subsequent instruction that implicitly uses **DPP<sub>x</sub>** via a long or indirect addressing mode, as shown in the following example:

```
In      :MOV DPP0,#4        ;select data page 4 via DPP0
In+1    :....              ;must not be an instruction using DPP0
In+2    :MOV DPP0:0000H,R1  ;move contents of R1 to address location
                                ;01 0000H(in data page 4) supposed segment
                                ; is enabled.
```

CONFIDENTIAL

Central Processing Unit, PEC, and Interrupt

### Explicit Stack Pointer Updating

None of the RET, RETI, RETS, RETP or POP instructions is capable of correctly using a new **SP** register value, which is to be updated by an immediately preceding instruction. Thus, to use the new **SP** register value without erroneously performed stack accesses, at least one instruction must be inserted between an instruction that explicitly writes to **SP**, and any of the afore mentioned subsequent instructions that implicitly use the **SP**, as shown in the following example:

```
In      :MOV SP,#0FA40H    ;select a new top of stack
In+1    :....             ;must not be an instruction popping operands
                               ;from the system stack
In+2    :POP R0           ;pop word value from new top of stack into R0
```

*Note: Conflicts with instructions writing to the stack (PUSH, CALL, SCXT) are solved internally by the MCU logic.*

### Controlling Interrupts

Software modifications (implicit or explicit) of the **PSW** are done in the execute phase of instructions. To maintain fast interrupt responses, however, the current interrupt prioritization round does not consider these changes, that is, an interrupt request may be acknowledged after the instruction that disables interrupts via IEN or ILVL, or after the following instructions. Therefore, time-critical instruction sequences should not begin directly after the instruction disabling interrupts, as shown in the following examples:

```
INTERRUPTS_OFF:
BCLR    IEN          ;globally disable interrupts
<Instr non-crit>    ;non-critical instruction
<Instr 1st-crit>    ;begin of uninterruptable critical sequence
. . .
<Instr last-crit>   ;end of uninterruptable critical sequence
INTERRUPTS_ON:
BSET    IEN          ;globally re-enable interrupts

CRITICAL_SEQUENCE:
ATOMIC #3            ;immediately block interrupts
BCLR    IEN          ;globally disable interrupts
. . .               ;here is the uninterruptable sequence
BSET    IEN          ;globally re-enable interrupts
```

*Note: The described delay of 1 instruction also applies for enabling the interrupt system; that is, no interrupt requests are acknowledged until the instruction following the enabling instruction.*

### External Memory Access Sequences

The effect described here will only become noticeable when watching the external memory access sequences on the external bus (for example, by means of a logic

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

analyzer). Different pipeline stages can simultaneously put a request on the External Bus Controller (EBC). The sequence of instructions processed by the MCU may diverge from the sequence of the corresponding external memory accesses performed by the EBC, due to the predefined priority of external memory accesses:

1. Write data
2. Fetch code
3. Read data.

### Initialization of Port Pins

Modifications of the direction of port pins (input or output) become effective only after the instruction following the modifying instruction. As bit instructions (BSET, BCLR) use internal read-modify-write sequences accessing the whole port, instructions modifying the port direction should be followed by an instruction that does not access the same port (see example below).

PORT\_INIT\_WRONG:

```
BSET    DPD.13      ;change direction of PD.13 to output
BSET    PD.9         ;PD.13 is still input, rd-mod-wr reads pin PD.13
```

PORT\_INIT\_RIGHT:

```
BSET    DPD.13      ;change direction of PD.13 to output
NOP                                     ;any instruction not accessing port D
BSET    PD.9         ;PD.13 is now output, rd-mod-wr reads PD.13's output
                                     ;latch
```

### Changing the System Configuration

The instruction following an instruction that changes the system configuration via register **SYSCON** (for example, the mapping of the internal local memory, segmentation, stack size) cannot use the new resources (for example, local memory or stack). In these cases, an instruction that does not access these resources should be inserted. Code accesses to the new local memory area are only possible after an absolute branch to this area.

*Note: As a rule, instructions that change local memory mapping should be executed from DPRAM or external memory.*

### BUSCON/ADDRSEL

The instruction following an instruction that changes the properties of an external address area cannot access operands within the new area. In these cases, an instruction that does not access this address area should be inserted. Code accesses to the new address area should be made after an absolute branch to this area.

*Note: As a rule, instructions that change external bus properties should not be executed from the external memory area.*

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

**Timing**

Instruction pipelining usually reduces the average instruction processing time (typically within a range of 1 to 4 machine cycles). However, there are some rare cases, where a particular pipeline situation causes the processing time for a single instruction to be extended either by a half or by one machine cycle. Although this additional time represents only a tiny part of the total program execution time, it might be of interest to avoid these pipeline-caused time delays in time-critical program modules.

Besides a general description of execution time, the following section provides some hints on how to optimize time-critical program parts with regard to such pipeline-caused timing particularities.

**6.3.7.2 Instruction State Times**

The time to execute an instruction depends on where the instruction is fetched from, and where possible operands are read from or written to. The fastest processing mode of the C166S is to execute a program fetched from the internal code memory. In that case, most of the instructions can be processed within just one machine cycle, which is also the general minimum execution time.

All external memory accesses are performed by the EBC, which works in parallel with the MCU. This section summarizes execution times.

The table below shows the minimum execution times required to process an instruction fetched from the internal local memory, the DPRAM, or external memory. These execution times apply to most instructions except some of the branches, the multiplication, the division, and a special move instruction. In case of internal local memory program execution, the execution time does not depend on the instruction length except for some special branch situations. The numbers in [Table 6-24](#) are in units of MCU clock cycles and assume no waitstates.

**Table 6-24 Minimum Execution Times**

Memory Area	Instruction Fetch		Word Operand Access	
	Word Instruction	Double-Word Instruction	Read from	Write to
Internal local memory	1 <sup>1)</sup>	1 <sup>1)</sup>	1 <sup>1)</sup>	1 <sup>1)</sup>
DPRAM	1 <sup>1)</sup>	2	1 <sup>1)</sup>	1 <sup>1)</sup>
16-bit demux bus	2	4	2	2
8-bit demux bus	4	8	4	4

<sup>1)</sup> Minimum execution time for instruction fetch and operand accesses. Nevertheless, the minimum execution time of an instruction remains 2 clock cycles (one machine cycle).

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

Execution from the DPRAM provides flexibility in terms of loadable and modifiable code. The execution time from external memory depends strongly on the selected bus mode and the programming of the bus cycles (wait states).

The operand and instruction accesses listed below can extend the execution time of an instruction:

- Internal Local Memory operand accesses (same for byte and word operand accesses)
- DPRAM operand reads via indirect addressing modes
- Internal SFR operand reads immediately after writing
- External operand reads
- External operand writes
- Jumps to non-aligned double word instructions in the internal local memory space
- Testing branch conditions immediately after PSW writes.

### 6.3.8 Dedicated CSFRs

#### The Constant Zeros Register

All bits of this bit-addressable register are fixed at 0 by hardware. This register is read-only. **ZEROS** can be used as a register-addressable constant of all zeros for bit manipulation or mask generation. It can be accessed via any instruction capable of accessing an SFR.

#### ZEROS

##### Constant Zeros Register

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Field	Bits	Type	Description
0	15:0	r	All bits fixed at zero.

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

### The Constant Ones Register

All bits of this bit-addressable register are fixed at 1 by hardware. This register is read-only. **ONES** can be used as a register-addressable constant of all ones for bit manipulation or mask generation. It can be accessed via any instruction which is capable of accessing an SFR.

#### ONES

##### Constant Ones Register

**Reset value: FFFF<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Field	Bits	Type	Description
1	15:0	r	All bits fixed to one.

### MCU Identification Register

**CPUID** contains the module and revision number of the implemented C166S module.

#### CPUID

##### MCU Identification Register

**Reset value: 0403<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPUMODNO								CPUREVNO							

Field	Bits	Type	Description
CPUMODNO	7:0	r	<b>Version Number</b> Version Number, starts with 01 <sub>H</sub> and is incremented with every new core version.
CPUREVNO	15:8	r	<b>Module Number</b> 04 <sub>H</sub> , C166S core module number.

### 6.3.9 Summary of MCU Registers

This section summarizes all registers in the C166S. There are two kind of registers, the General Purpose Registers (GPR) and the MCU-Special Function Registers (CSFR). The GPRs are the working registers of the arithmetic and logic operations and may be also used as address pointers indirect addressing modes. The CSFRs are the control registers of the C166S.

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

For easy reference, the CSFRs are listed in this section by address (to identify a register at a given address) and by name (to find an address of a specific register).

### 6.3.9.1 General Purpose Registers

The GPRs are the working registers of the C166S. All GPRs are bit addressable.

**Table 6-25 Addressing Modes to Access Word-GPRs**

Name	Physical Address	8-Bit Address	4-Bit Address	Description	Reset Value
R0	(CP)+0	F0 <sub>H</sub>	0 <sub>H</sub>	General-Purpose Word Register R0	UUUU <sub>H</sub>
R1	(CP)+2	F1 <sub>H</sub>	1 <sub>H</sub>	General-Purpose Word Register R1	UUUU <sub>H</sub>
R2	(CP)+4	F2 <sub>H</sub>	2 <sub>H</sub>	General-Purpose Word Register R2	UUUU <sub>H</sub>
R3	(CP)+6	F3 <sub>H</sub>	3 <sub>H</sub>	General-Purpose Word Register R3	UUUU <sub>H</sub>
R4	(CP)+8	F4 <sub>H</sub>	4 <sub>H</sub>	General-Purpose Word Register R4	UUUU <sub>H</sub>
R5	(CP)+10	F5 <sub>H</sub>	5 <sub>H</sub>	General-Purpose Word Register R5	UUUU <sub>H</sub>
R6	(CP)+12	F6 <sub>H</sub>	6 <sub>H</sub>	General-Purpose Word Register R6	UUUU <sub>H</sub>
R7	(CP)+14	F7 <sub>H</sub>	7 <sub>H</sub>	General-Purpose Word Register R7	UUUU <sub>H</sub>
R8	(CP)+16	F8 <sub>H</sub>	8 <sub>H</sub>	General-Purpose Word Register R8	UUUU <sub>H</sub>
R9	(CP)+18	F9 <sub>H</sub>	9 <sub>H</sub>	General-Purpose Word Register R9	UUUU <sub>H</sub>
R10	(CP)+20	FA <sub>H</sub>	A <sub>H</sub>	General-Purpose Word Register R10	UUUU <sub>H</sub>
R11	(CP)+22	FB <sub>H</sub>	B <sub>H</sub>	General-Purpose Word Register R11	UUUU <sub>H</sub>
R12	(CP)+24	FC <sub>H</sub>	C <sub>H</sub>	General-Purpose Word Register R12	UUUU <sub>H</sub>
R13	(CP)+26	FD <sub>H</sub>	D <sub>H</sub>	General-Purpose Word Register R13	UUUU <sub>H</sub>
R14	(CP)+28	FE <sub>H</sub>	E <sub>H</sub>	General-Purpose Word Register R14	UUUU <sub>H</sub>

The first 8 GPRs (R7...R0) may be also accessed byte-wise. Unlike SFRs, writing to a GPR byte does not affect another byte of the GPR.

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

The byte-wise accessible registers in [Table 6-26](#) have special names.

**Table 6-26 Addressing Modes to Access Byte-GPRs**

Name	Physical Address	8-Bit Address	4-Bit Address	Description	Reset Value
RL0	(CP)+0	F0 <sub>H</sub>	0 <sub>H</sub>	General-Purpose Byte Register RL0	UU <sub>H</sub>
RH0	(CP)+1	F1 <sub>H</sub>	1 <sub>H</sub>	General-Purpose Byte Register RL1	UU <sub>H</sub>
RL1	(CP)+2	F2 <sub>H</sub>	2 <sub>H</sub>	General-Purpose Byte Register RL2	UU <sub>H</sub>
RH1	(CP)+3	F3 <sub>H</sub>	3 <sub>H</sub>	General-Purpose Byte Register RL3	UU <sub>H</sub>
RL2	(CP)+4	F4 <sub>H</sub>	4 <sub>H</sub>	General-Purpose Byte Register RL4	UU <sub>H</sub>
RH2	(CP)+5	F5 <sub>H</sub>	5 <sub>H</sub>	General-Purpose Byte Register RL5	UU <sub>H</sub>
RL3	(CP)+6	F6 <sub>H</sub>	6 <sub>H</sub>	General-Purpose Byte Register RL6	UU <sub>H</sub>
RH3	(CP)+7	F7 <sub>H</sub>	7 <sub>H</sub>	General-Purpose Byte Register RL7	UU <sub>H</sub>
RL4	(CP)+8	F8 <sub>H</sub>	8 <sub>H</sub>	General-Purpose Byte Register RL8	UU <sub>H</sub>
RH4	(CP)+9	F9 <sub>H</sub>	9 <sub>H</sub>	General-Purpose Byte Register RL9	UU <sub>H</sub>
RL5	(CP)+10	FA <sub>H</sub>	A <sub>H</sub>	General-Purpose Byte Register RL10	UU <sub>H</sub>
RH5	(CP)+11	FB <sub>H</sub>	B <sub>H</sub>	General-Purpose Byte Register RL11	UU <sub>H</sub>
RL6	(CP)+12	FC <sub>H</sub>	C <sub>H</sub>	General-Purpose Byte Register RL12	UU <sub>H</sub>
RH6	(CP)+13	FD <sub>H</sub>	D <sub>H</sub>	General-Purpose Byte Register RL13	UU <sub>H</sub>
RL7	(CP)+14	FE <sub>H</sub>	E <sub>H</sub>	General-Purpose Byte Register RL14	UU <sub>H</sub>
RH7	(CP)+15	FF <sub>H</sub>	F <sub>H</sub>	General-Purpose Byte Register RL15	UU <sub>H</sub>



**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

### 6.3.9.2 Core Special Function Registers Ordered by Address

**Table 6-27** lists all CSFRs which are implemented in the C166S ordered by their physical address. **Bit-addressable** CSFRs are marked with the letter “b” in column “Name”.

CSFRs within the **Extended SFR-Space** (ESFRs) are marked with the letter “E” in column “8-Bit Address”.

**Table 6-27 Core Special Function Registers Ordered by Address**

Name	Physical Address	8-Bit Address	Description	Reset Value
<b>CPUID</b>	F00C <sub>H</sub>	E-06 <sub>H</sub>	MCU Identification Register	04YY <sub>H</sub> <sup>1)</sup>
<b>DPP0</b>	FE00 <sub>H</sub>	00 <sub>H</sub>	Data Page Pointer 0 (10 bits)	0000 <sub>H</sub>
<b>DPP1</b>	FE02 <sub>H</sub>	01 <sub>H</sub>	Data Page Pointer 1 (10 bits)	0001 <sub>H</sub>
<b>DPP2</b>	FE04 <sub>H</sub>	02 <sub>H</sub>	Data Page Pointer 2 (10 bits)	0002 <sub>H</sub>
<b>DPP3</b>	FE06 <sub>H</sub>	03 <sub>H</sub>	Data Page Pointer 3 (10 bits)	0003 <sub>H</sub>
<b>CSP</b>	FE08 <sub>H</sub>	04 <sub>H</sub>	Code Segment Pointer (8 bits, not directly writable)	0000 <sub>H</sub>
<b>MDH</b>	FE0C <sub>H</sub>	06 <sub>H</sub>	Multiply Divide High Word	0000 <sub>H</sub>
<b>MDL</b>	FE0E <sub>H</sub>	07 <sub>H</sub>	Multiply Divide Low Word	0000 <sub>H</sub>
<b>CP</b>	FE10 <sub>H</sub>	08 <sub>H</sub>	Context Pointer	FC00 <sub>H</sub>
<b>SP</b>	FE12 <sub>H</sub>	09 <sub>H</sub>	Stack Pointer	FC00 <sub>H</sub>
<b>STKOV</b>	FE14 <sub>H</sub>	0A <sub>H</sub>	Stack Overflow Register	FA00 <sub>H</sub>
<b>STKUN</b>	FE16 <sub>H</sub>	0B <sub>H</sub>	Stack Underflow Register	FC00 <sub>H</sub>
<b>MDC</b>	FF0E <sub>H</sub>	87 <sub>H</sub>	Multiply Divide Control Register	0000 <sub>H</sub>
<b>PSW</b>	FF10 <sub>H</sub>	88 <sub>H</sub>	Program Status Word	0000 <sub>H</sub>
<b>SYSCON</b>	FF12 <sub>H</sub>	89 <sub>H</sub>	System/MCU Control Register	YYYY <sub>H</sub> <sup>2)</sup>
<b>ZEROS</b>	FF1C <sub>H</sub>	8E <sub>H</sub>	Constant Value 0's Register (read only)	0000 <sub>H</sub>
<b>ONES</b>	FF1E <sub>H</sub>	8F <sub>H</sub>	Constant Value 1's Register (read only)	FFFF <sub>H</sub>
<b>TFR</b>	FFAC <sub>H</sub>	D6 <sub>H</sub>	Trap Flag Register	0000 <sub>H</sub>

<sup>1)</sup> YY: defined by implemented MCU version

<sup>2)</sup> YYYY: defined by reset and system configuration

**CONFIDENTIAL**

**Central Processing Unit, PEC, and Interrupt**

### 6.3.9.3 Register Overview Interrupt and Peripheral Event Controller

The following table lists all xSFRs which are implemented in the C166S Interrupt and Peripheral Event Controller.

**Table 6-28 Register Overview Interrupt and Peripheral Event Controller**

Register Name	Register Description	Module Block
<b>PECSNx</b>		
PECSN0	PEC Pointer 0 Segment Address Reg.	PEC Pointer
PECSN1	PEC Pointer 1 Segment Address Reg.	PEC Pointer
PECSN...	PEC Pointer ... Segment Address Reg.	PEC Pointer
PECSN7	PEC Pointer 7 Segment Address Reg.	PEC Pointer
PECSN8	PEC Pointer 8 Segment Address Reg.	PEC Pointer
PECSN...	PEC Pointer ... Segment Address Reg.	PEC Pointer
PECSN15	PEC Pointer 15 Segment Address Reg.	PEC Pointer
<b>PECCx</b>		
PECC0	PEC Channel 0 Control Register	PEC Control
PECC...	PEC Channel ... Control Register	PEC Control
PECC7	PEC Channel 7 Control Register	PEC Control
PECC8	PEC Channel .8 Control Register	PEC Control
PECC...	PEC Channel ... Control Register	PEC Control
PECC15	PEC Channel 15 Control Register	PEC Control
<b>IRQxIC</b>		
IRQ0IC	Interrupt 0 Control Register	Arbitration Control
...	...	...
IRQ63IC	Interrupt 63 Control Register	Arbitration Control
IRQ64IC	Interrupt 64 Control Register	Arbitration Control
IRQ65IC	Interrupt 65 Control Register	Arbitration Control
...	...	...
IRQ111IC	Interrupt 111 Control Register	Arbitration Control

CONFIDENTIAL

Address Mapping and Memory Use

## 6.4 Address Mapping and Memory Use

### 6.4.1 Data Organization in Memory

Bytes are stored at even or odd byte addresses. Words are stored in ascending memory locations with the low byte at an even byte address, followed by the high byte at the next odd byte address. Instruction double-words are stored in ascending memory locations as two subsequent words, without any restrictions (non-aligned). Single bits are always stored in the specified bit position at a word address. The memory and registers store data and instructions in little-endian byte order (the least significant bytes are at lower addresses). The byte ordering is illustrated in [Figure 6-12](#). Bit position 0 is the least significant bit of the byte at an even byte address, and bit position 15 is the most significant bit of the byte at the next odd byte address. Bit addressing is supported for a part of the SFRs, a part of the DPRAM and for the General-Purpose Registers (GPRs)

**Figure 6-12 Storage of Words, Byte and Bits in a Byte Organized Memory**

°				xxxx xxxA <sub>H</sub>
15	14	... Bits ...	8	xxxx xxx9 <sub>H</sub>
7	6	... Bits ...	0	xxxx xxx8 <sub>H</sub>
Byte				xxxx xxx7 <sub>H</sub>
Byte				xxxx xxx6 <sub>H</sub>
Word (High Byte)				xxxx xxx5 <sub>H</sub>
Word (Low Byte)				xxxx xxx4 <sub>H</sub>
Double Word (High Byte)				xxxx xxx3 <sub>H</sub>
Double Word (Third Byte)				xxxx xxx2 <sub>H</sub>
Double Word (Second Byte)				xxxx xxx1 <sub>H</sub>
Double Word (Low Byte)				xxxx xxx0 <sub>H</sub>
°				xxxx xxxF <sub>H</sub>

Byte units forming a single word must always be stored within the same physical (internal, external, ROM, RAM) and organizational (page, segment) memory area.

### 6.4.2 Crossing Memory Boundaries

The address space of the C166S CPU is divided implicitly into equally-sized blocks of different granularity and into logical memory areas. Crossing the boundaries between these blocks (code or data) or areas requires special attention to ensure that the controller executes the desired operations.

**Memory Areas** are partitions of the address space that represent different kinds of memory (if provided at all). These memory areas are the DPRAM, the internal IO, the internal LM (if available), and the external memory.

Accessing subsequent data locations that belong to different memory areas is not fully supported, and may therefore lead to erroneous results. There is no problem if the memory boundaries are word-aligned. However, when executing code, the different memory areas (internal memory areas and external memory) must be switched explicitly via branch instructions. Sequential boundary crossing is not supported and may lead to erroneous results.

**Segments** are contiguous 64kByte blocks. They are referenced via the Code Segment Pointer (**CSP**) for code fetches, and via an explicit segment number for data accesses overriding the standard DPP scheme.

During code fetching, segments are not changed automatically, but rather must be switched explicitly. The instructions JMPS, CALLS, and RETS do this. Larger sequential programs make sure that the highest used code location of a segment contains an unconditional branch instruction to the next following segment, to prevent the prefetcher from trying to leave the current segment.

**Data Pages** are contiguous 16kByte blocks. They are referenced via the Data Page Pointers DPP3...0 and via an explicit data page number for data accesses overriding the standard DPP scheme. Each DPP register can select one of the possible 1024 data pages. The DPP register that is used for the current access is selected via the two upper bits of the 16-bit data address. Subsequent 16-bit data addresses that cross the 16kByte data page boundaries will use different data page pointers, while the physical locations need not be contiguous within memory.

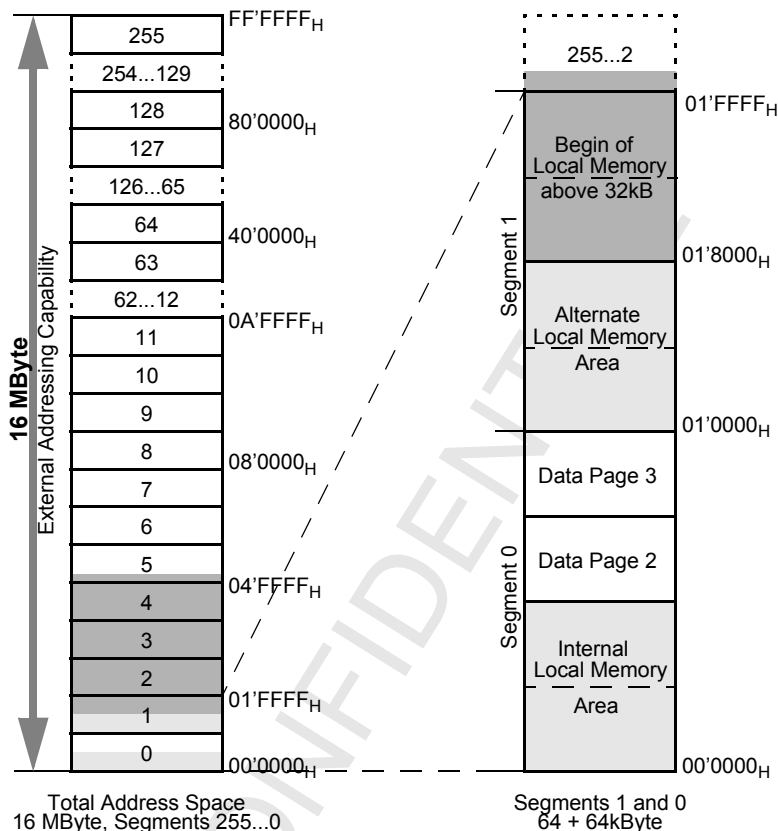
### 6.4.3 Memory Organization

The memory space of the C166S has a “Von Neumann” architecture. This means that code and data are accessed within the same linear address space. All of the physically separated memory areas, including internal ROM/FLASH/DRAM (where integrated), DPRAM, the internal Special Function Register (SFR) and Extended Special Function Register (ESFR) areas, and external memory are mapped into one common address space.

CONFIDENTIAL

Address Mapping and Memory Use

Figure 6-13 C166S Address Space Overview



The C166S has a total addressable memory space of 16 MBytes. This address space is arranged as 256 segments of 64kBytes each, and each segment is again subdivided into four data pages of 16kBytes each (see [Figure 6-13](#)).

Most internal memory areas are mirrored into segment 0, the system segment. The upper 4kBytes of segment 0 (00 F000<sub>H</sub>-00 FFFF<sub>H</sub>) are the SFRs and ESFRs and the DPRAM areas. The lower 32kByte of segment 0 (00 0000<sub>H</sub>-00 7FFF<sub>H</sub>) may be occupied by a part of the on-chip program memory and is called the internal Local Memory (LM) area. This LM area can be remapped to segment 1 (01 0000<sub>H</sub>-01 7FFF<sub>H</sub>) to enable external memory access in the lower half of segment 0, or the internal LM may be disabled completely.

**CONFIDENTIAL**

**Address Mapping and Memory Use**

Code and data may be stored in any part of the internal memory areas except for the SFR blocks, which may be used for control/data, but not for instructions.

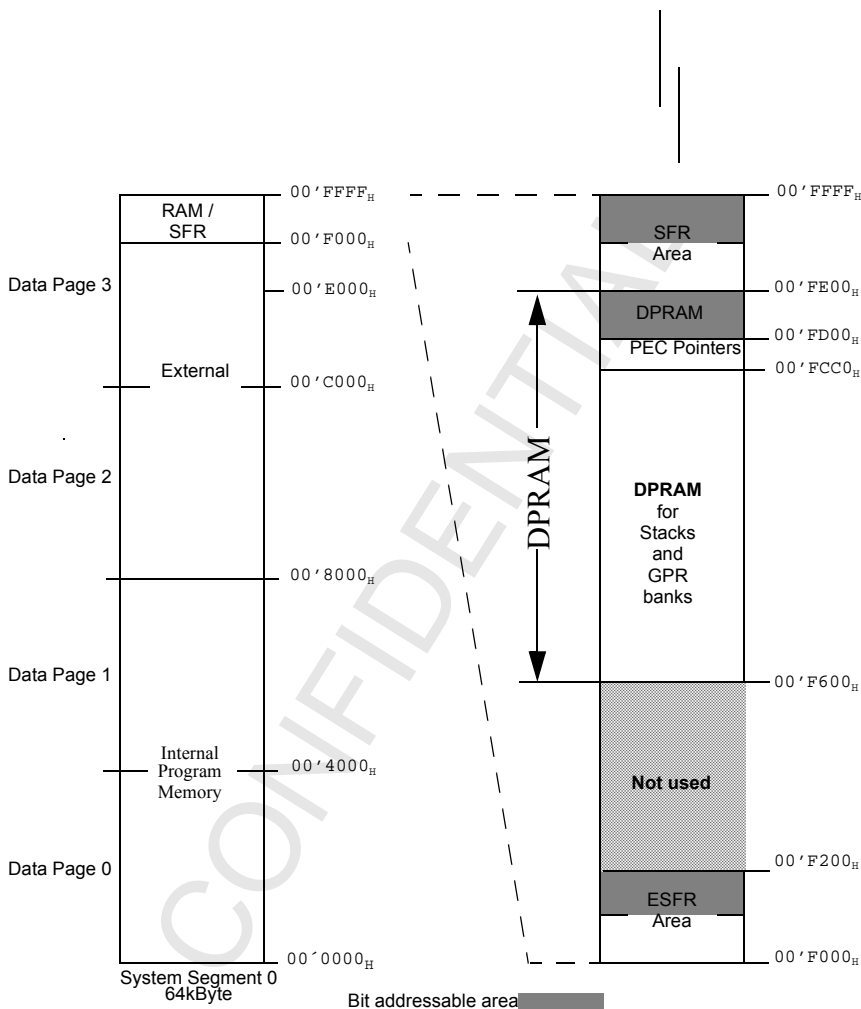
*Note: Accesses to the internal LM area on devices without LM will produce unpredictable results.*

### **6.4.3.1 DPRAM, (E)SFR and PEC Pointer Memory Areas**

The C166S differentiates between the internal dual port data memory (DPRAM), and the normal (SFR) and extended (ESFR) special function register addresses used for the internal peripherals. The DPRAM, the SFR and the ESFR areas are located within data page 3, and provide fast accesses using one dedicated Data Page Pointer (DPP) (see [Figure 6-10](#)).

*Note: Code accesses are not possible from the SFR areas.*

Figure 6-14 DPRAM and SFR Areas within Segment 0 (Simplified)



*Note: High-byte accesses of SFRs using the 8-bit offset addressing mode are not possible.*

*Note: Writing to any byte of an SFR causes the non-addressed complementary byte to be cleared.*

CONFIDENTIAL

Address Mapping and Memory Use

*Note: GPRs can be accessed using the 8-bit offset addressing mode, but the GPRs are not mapped into the SFR and ESR memory area. Using the corresponding long address instead of a GPR access executes an internal peripheral bus access.*

### 6.4.3.1.1 Dual Port Data Memory

The DPRAM is a fast dual port SRAM available mainly for data storage. It has two ports both with address and data lines to allow parallel access. All accesses take only one clock cycle. It is used for:

- GPR banks
- Variable and other data storage
- System and user stacks
- PEC source and destination pointers.

A 2kByte memory area (00 F600<sub>H</sub>-00 FE00<sub>H</sub>) is reserved for the DPRAM. The upper 256 Bytes of the DPRAM (00 FD00<sub>H</sub>-00 FDFF<sub>H</sub>) and the GPRs of the current bank are provided for single-bit storage, and thus they are bit addressable (see shaded blocks in [Figure 6-14](#)). Any word and byte data in the DPRAM can be accessed via indirect or long 16-bit addressing modes if the selected DPP register points to data page 3. Any word data access is made on an even byte address. The highest possible word data storage location in the DPRAM is 00 FDFE<sub>H</sub>.

The highest possible code storage location in the DPRAM is either 00 FDFE<sub>H</sub> for single-word instructions, or 00 FDFC<sub>H</sub> for double-word instructions (but this is the bit-addressable area, which should not be used for code). The respective location must contain a branch instruction (unconditional), because sequential boundary crossing from DPRAM to the SFR area is not supported and causes erroneous results.

### 6.4.3.1.2 Dual Port RAM/General Purpose Registers

#### Data Organization

The memory-mapped GPRs use a block of 16 consecutive words within the DPRAM Segment 0. The Context Pointer ([CP](#)) register determines the base address of the currently active register bank. This register bank may consist of up to 16 word GPRs (R0, R1, ..., R15), and/or up to 16 byte GPRs (RL0, RH0, ..., RL7, RH7). The 16-byte GPRs are mapped onto the first 8 word GPRs (refer to [Table 6-29](#)).

In contrast to the system stack, a register bank grows from lower towards higher address locations and occupies a maximum space of 32 bytes. The GPRs are accessed via short 2-, 4-, or 8-bit addressing modes using the [CP](#) register as base address (independent of the current DPP register contents). Additionally, each bit in the currently active register bank can be accessed individually.



**CONFIDENTIAL**

**Address Mapping and Memory Use**

The C166S supports register bank (context) switching. Multiple memory-mapped register banks can physically exist within the DPRAM at the same time. Only the register bank selected by the **CP** register is active at a given time, however. Selecting a new active register bank is simply done by updating the **CP** register.

**Table 6-29 Mapping of General-Purpose Registers to DPRAM Addresses**

DPRAM Address	Byte Registers	Word Register
<CP> + 1E <sub>H</sub>	---	R15
<CP> + 1C <sub>H</sub>	---	R14
<CP> + 1A <sub>H</sub>	---	R13
<CP> + 18 <sub>H</sub>	---	R12
<CP> + 16 <sub>H</sub>	---	R11
<CP> + 14 <sub>H</sub>	---	R10
<CP> + 12 <sub>H</sub>	---	R9
<CP> + 10 <sub>H</sub>	---	R8
<CP> + 0E <sub>H</sub>	RH7 RL7	R7
<CP> + 0C <sub>H</sub>	RH6 RL6	R6
<CP> + 0A <sub>H</sub>	RH5 RL5	R5
<CP> + 08 <sub>H</sub>	RH4 RL4	R4
<CP> + 06 <sub>H</sub>	RH3 RL3	R3
<CP> + 04 <sub>H</sub>	RH2 RL2	R2
<CP> + 02 <sub>H</sub>	RH1 RL1	R1
<CP> + 00 <sub>H</sub>	RH0 RL0	R0

A particular Switch Context (SCXT) instruction performs register bank switching and automatic saving of the previous context. The number of implemented register banks (arbitrary sizes) is limited only by the size of the available DPRAM.

### SFR/ESFR Table

A list of all SFRs/ESFRs which are implemented in the E-GOLDradio ordered by their physical address is contained in [Section 12.2 PD-Bus Register Addresses \(on Page 1273\)](#).

#### 6.4.3.1.3 Dual Port RAM /System Stack

The system stack must be defined within the DPRAM. The size of the system stack is controlled by bitfield **SYSCON.STKSZ** (refer to [Table 6-30](#)).

**Table 6-30 System Stack Size**

<STKSZ>	Stack Size (Words)	DPRAM Addresses (Words)
000 <sub>B</sub>	256	00 FBFE <sub>H</sub> -00 FA00 <sub>H</sub> (Default after Reset)
001 <sub>B</sub>	128	00 FBFE <sub>H</sub> -00 FB00 <sub>H</sub>
010 <sub>B</sub>	64	00 FBFE <sub>H</sub> -00 FB80 <sub>H</sub>
011 <sub>B</sub>	32	00 FBFE <sub>H</sub> -00 FBC0 <sub>H</sub>
100 <sub>B</sub>	512	00 FBFE <sub>H</sub> -00 F800 <sub>H</sub>
101 <sub>B</sub>	---	Reserved. Do not use this combination.
110 <sub>B</sub>	---	Reserved. Do not use this combination.
111 <sub>B</sub>	1024	00 FDFE <sub>H</sub> -00 F200 <sub>H</sub> <i>Note: The stack is not circular.</i>

For all system stack operations, the stack memory is accessed via the Stack Pointer (**SP**). The system stack implementation in the C166S is from high to low memory. The system stack grows downward as it is filled. The **SP** register is decremented first each time data is pushed on the system stack, and incremented after each time the data is pulled from the system stack. Only word accesses are supported to the system stack. The **SP** points to the address of the latest system stack entry, rather than to the next available system stack address.

A STack OVerflow (**STKOV**) register and a STack UNderflow (**STKUN** (on Page 163)) register are provided to control the lower and upper limits of the selected stack area. These two stack boundary registers can be used not only for protection against data destruction, but also to implement a circular stack with hardware-supported system stack flushing and filling (except for option **SYSCON.STKSZ** = 111).

#### 6.4.3.1.4 Addresses for PEC Source and Destination Pointers

The 32 word locations for the 16 PEC channels in the DPRAM from 00 FCC0<sub>H</sub> to 00 FCFE<sub>H</sub> (just below the bit-addressable section) are provided as source and destination address pointers for data transfers on the PEC channels. Each channel uses a pair of pointers stored in two subsequent word locations with the SouRCe Pointer (**SRCPx**) on the lower and the DeSTination Pointer (**DSTPx**) on the higher word address.

Whenever a PEC data transfer is performed, the pair of **SRCPx** and **DSTPx** selected by the specified PEC channel number is accessed independent of the current DPP register contents; also, the locations referred to by these pointers are accessed independently of the current DPP register contents. If a PEC channel is not used, the corresponding pointer location area is available and can be used for word or byte data storage or for instructions.

For more details about use of **SRCPx** and **DSTPx** for PEC data transfer, see [Section 6.3.3 Interrupt and Exception Execution \(on Page 111\)](#).

#### 6.4.3.1.5 Special Function Register Areas

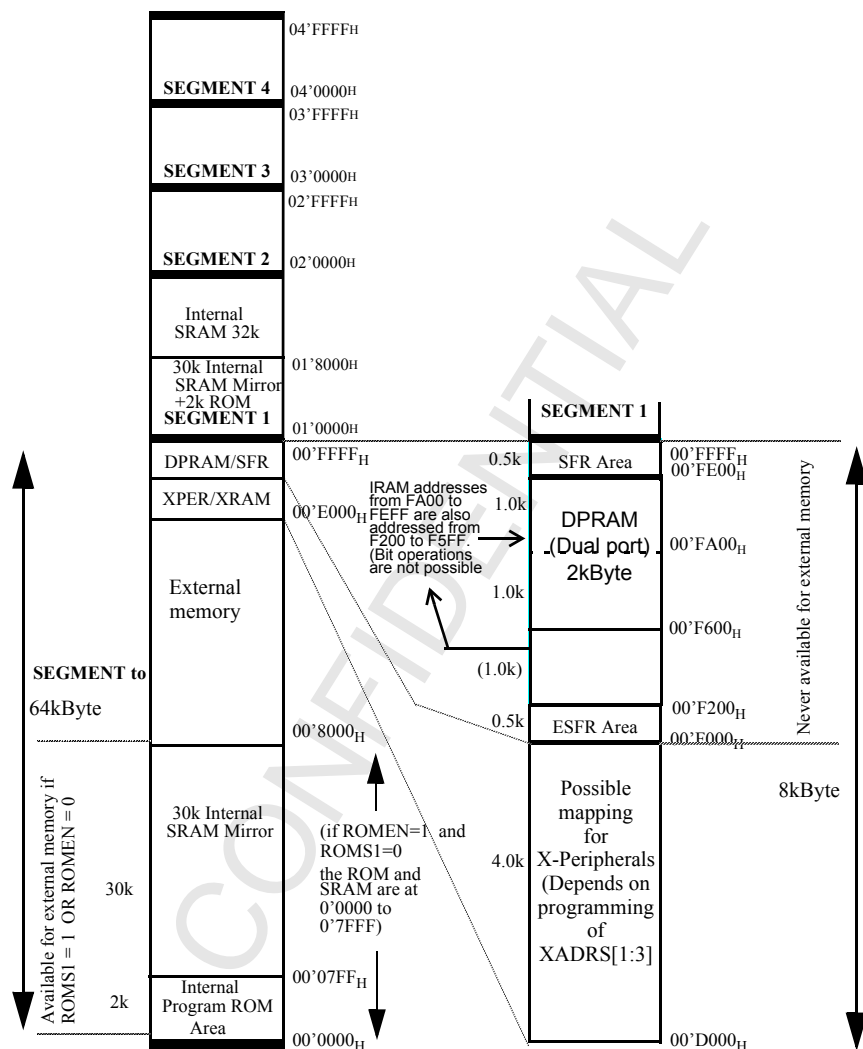
The functions of the MCU, the bus interface, the I/O ports, and the on-chip peripherals of the C166S are controlled via a number of SFRs. These SFRs are arranged within two 512-Byte areas. The first register block, the SFR area, is located in the 512 Bytes above the DPRAM (00 FFFF<sub>H</sub>-00 FE00<sub>H</sub>). The second register block, the ESFR area, is located in the 512 Bytes below the DPRAM (00 F1FF<sub>H</sub>-00 F000<sub>H</sub>).

SFRs can be addressed via indirect and long 16-bit addressing modes. Using an 8-bit offset together with an implicit base address makes it possible to address word SFRs and their respective low bytes. However, this does not work for the respective high bytes.

CONFIDENTIAL

Address Mapping and Memory Use

Figure 6-15 Detailed Memory Map for Segments 0-4



The upper half of each register block is bit-addressable, so the respective control/status bits can be modified directly, or checked using bit addressing.

When accessing registers in the ESFR area using 8-bit addresses or direct bit addressing, the EXTend Register (EXTR) instruction is required before accessing

**CONFIDENTIAL**

**Address Mapping and Memory Use**

registers in the ESFR area, to switch the short-addressing mechanism from the standard SFR area to the ESFR area. This is not required for 16-bit and indirect addresses. The GPRs R15...R0 are duplicated, that is, they are accessible within both register blocks via short 2-, 4- or 8-bit addresses without switching.

Example 1:

```
EXTR    #4                ;Switch to ESFR area for the next 4
                        ; instructions
MOV     ODP2, #data16     ;ODP2 (ESFR register) uses 8-bit register
                        ;addressing
BFLDL   DP6, #mask, #data8 ;DP6 (ESFR register) bit addressing for bit
                        ;fields
BSET    DP6.7             ;DP6 (ESFR register) bit addressing for
single                                     ;bits
MOV     T8REL, R1         ;T8REL uses 16-bit address, R1 is duplicated...
                        ;...and also accessible via the ESFR mode
                        ;(EXTR is not required for this access)
                        ;----- ;-----;The scope of the EXTR #4
                        ;instruction ends here
MOV     T8REL, R1         ;T8REL uses 16-bit address, R1 is duplicated...
                        ;...and does not require switching
```

To minimize the switching of SFR banks, the ESFR area holds registers that are mainly required for initialization and mode selection. Registers that need to be accessed frequently are allocated to the standard SFR area, wherever possible.

*Note: The tools are equipped to monitor accesses to the ESFR area and will automatically insert EXTR instructions, switch the SFR bank address, or issue a warning in case of missing or excessive EXTR instructions.*

### 6.4.3.2 External Memory Space

The C166S CPU can use an address space of up to 16 MBytes. Only parts of this address space are occupied by the internal LM, DPRAM and the IO/SFR area. All addresses not used for this kind of on-chip memory or for registers may reference external memory locations. This external memory space is accessed via the Bus Controller (BC).

The BC is the bus-bridge between the C166S CPU and the external/internal bus interfaces. The external bus interface allows access to external (off-chip) peripherals and additional off-chip volatile and non-volatile memories. The external bus interface may further limit the amount of addressable off-chip memory. The internal bus interface provides an internal system bus that allows the on-chip integration of customer-specific peripherals, volatile and non-volatile memories. The availability of the internal and external bus interfaces depends on the functionality of the integrated BC.

## CONFIDENTIAL

## Address Mapping and Memory Use

#### 6.4.3.2.1 External Data Accesses

External word and byte data can be accessed only via indirect or long 16-bit addressing modes using one of the four DPP registers. There is no short-addressing mode for external operands. Any word data access is made to an even byte address, and double-word accesses to modulo-4 byte addresses (even word addressing).

External memory is not provided for single-bit storage and therefore it is not bit-addressable.

#### 6.4.3.3 SFR/ESFR Table

A list of all SFRs/ESFRs which are implemented in the E-GOLDradio ordered by their physical address is contained in [Section 12.2 PD-Bus Register Addresses \(on Page 1273\)](#).

#### 6.4.3.4 Internal ROM and Local Memory RAM

*Note: Refer to [Table 6-15 Detailed Memory Map for Segments 0-4 \(on Page 200\)](#).*

*For LM-Bus ROM and RAM wait state information, refer to [Section 9.1 RAM \(on Page 615\)](#).*

After reset in internal ROM mode the address range  $0000_H \dots 7FFF_H$  (32kByte) is not available for external memory access but is allocated to internal ROM and program RAM.

As in previous versions of EGOLD 2kByte of internal ROM are available which are located in the address range  $0000_H \dots 07FF_H$ , that is, the lower end of the masked 32kByte address region.

In addition to the dual port RAM available already on previous versions 32kBytes internal SRAM are available in this version of E-GOLDradio. The SRAM is usable both as data and program RAM. The RAM permits 32 bit wide code fetches from the controller as in the program ROM and 16 bit wide data access. It can be used to boost controller performance for frequently used critical program sections. It also reduces controller power consumption since few external memory accesses are required.

The program memory can be used for both code (instructions) and data (constants, tables, etc.) storage.

Code fetches are always made on even byte addresses. The highest possible code storage location in the program memory is either  $xx\ xxFE_H$  for single word instructions, or  $xx\ xxFC_H$  for double word instructions. The respective location must contain a branch instruction (unconditional), because sequential boundary crossing from program memory to X-Bus/external memory is not supported and causes erroneous results.

Any word and byte data read accesses may use the indirect or long 16-bit addressing modes. There is no short addressing mode for program memory operands. Any word data access is made to an even byte address. The highest possible word data storage

**CONFIDENTIAL**

**Address Mapping and Memory Use**

location in the program memory is  $xx\ xxFE_H$ . For PEC data transfers the program memory can be accessed independent of the contents of the DPP registers via the PEC source and destination pointers.

By setting bit **SYSCON.ROMS1** to 1 both internal ROM and program RAM can be mapped to segment 1. By setting bit **SYSCON.ROMS1** to 0 both internal ROM and program RAM can be mapped to segment 0.

By setting bit **SYSCON.ROMEN** to 0 both internal ROM and internal program RAM will be switched off. The 32kByte address range will be usable again for accesses to external memory.

The segment 1 is split into 2 parts. The first part is from  $1\ 0000_H$  to  $1\ 7FFF_H$ , and the second part is from  $1\ 8000_H$  to  $1\ FFFF_H$ .

The first part contains the following:

- The complete internal ROM (2kbytes at the address from  $1\ 0000_H$  to  $1\ 07FF_H$ ).
- A part of the internal SRAM (30kbytes at the address  $1\ 0800_H$  to  $1\ 7FFF_H$  mapped to the higher part of the 32kbytes internal SRAM).

The second part contains the complete internal SRAM (at the address from  $1\ 8000_H$  to  $1\ FFFF_H$ ).

The segment 0 is explained in the [Table 6-15 Detailed Memory Map for Segments 0-4 \(on Page 200\)](#).

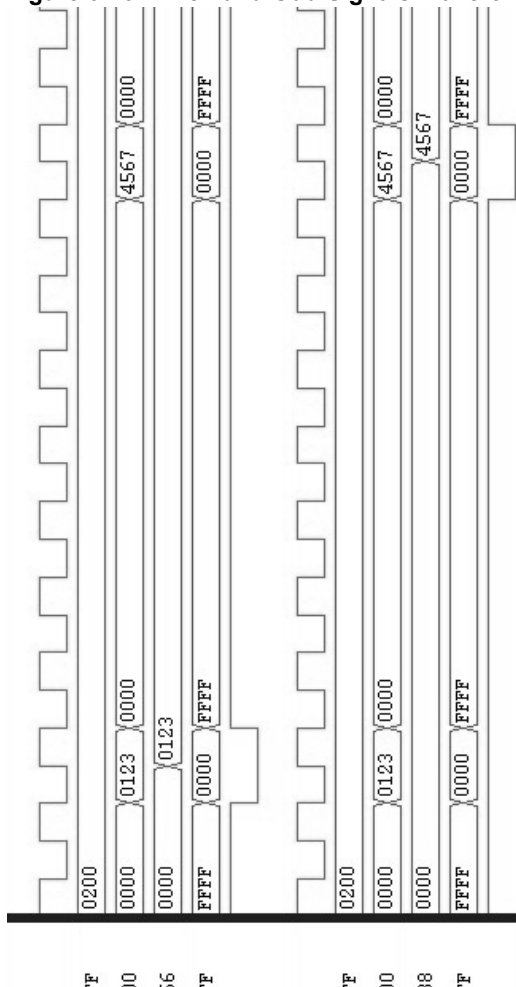
#### **6.4.3.4.1 Even and Odd Memories Mechanism**

The c166s\_lm\_ramrom module is build based on 2 parallel architectures, an even memory block and an odd memory block. The 14-bit address bus (15:2) coming into the Local Memory is aligned by removing the 2 low bits. The C166S decodes this address by selecting either the even memory or the odd memory. Even memory is selected when bit 1 is zero, odd memory is selected when bit 1 is one.

The following example illustrates this mechanism (see [Figure 6-16 on Page 204](#)):

1. A 16-bit data  $0123_H$  is written to address  $10800_H$  (segment 1).
2. A 16-bit data  $4567_H$  is written to address  $10802_H$ .
3. The MCU:
  - a) Decodes address  $10800_H$
  - b) Sends  $0200_H$  to the LM bus
  - c) Selects lm\_sel\_even\_n\_i, which is connected to the mem\_CEN of lm\_even module.
4. The MCU:
  - a) Decodes address  $10802_H$
  - b) Sends  $0200_H$  to the LM bus
  - c) Selects lm\_sel\_odd\_n\_i, which is connected to the mem\_CEN of lm\_odd module.

Figure 6-16 Even and Odd Signals Waveforms Example





CONFIDENTIAL

X-Bus System Architecture

**a67 assembler code:**

```
mov    dpp2, #4           ;this offset 800H comes from the ROM which covers the
                           ;lower 2kByte of the LM Memory address space overlapping
                           ;with a small part of the RAM
mov    r2, #00123h
mov    dpp2:0800h, r2
mov    r2, #04567h
mov    dpp2:0802h, r2
```

## 6.5 X-Bus System Architecture

### 6.5.1 External Bus Unit

Refer to [Section 10.13 External Bus Unit \(on Page 851\)](#).

### 6.5.2 The Internal X-Bus Interface

The C166S provides an on-chip interface (the X-Bus interface) by which integrated application-specific peripherals can be connected to the standard controller core. The X-Bus is an internal representation of the external bus interface, that is, it is operated in the same way.

For each peripheral on the X-Bus (an X-Peripheral) there is a separate address window controlled by a register pair **XBCONx/XADRSx** (similar to registers **BUSCONx (on Page 870)** and **ADDRSELx (on Page 873)**). Because an interface to a peripheral is represented in many cases by just a few registers, most of the **ADDRSELx** registers select smaller address windows than the standard **BUSCONx** registers. As the register pairs control integrated peripherals rather than externally connected ones, most of the registers are fixed by mask programming rather than being user-programmable.

The X-Bus provides byte-wide or word-wide X-Peripheral accesses. Because the on-chip connection can be very efficient, and for performance reasons, X-Peripherals are implemented only with a separate address bus, that is, in demultiplexed bus mode.

### Enabling X-Bus Peripherals

After reset, all on-chip X-Bus peripherals are disabled. An X-Bus peripheral cannot be used unless it has been enabled via the global enable bit **SYSCON.XPEN**. Before using any X-Bus peripheral its **XBCONx (on Page 208)** register has to be programmed to a recommended value (its **XBCONx.BUSACT** bit set to 1).

CONFIDENTIAL

X-Bus System Architecture

### 6.5.2.1 X-Bus Access Control

In E-GOLDradio up to six (configurable) address ranges with according bus definitions can be programmed for X-Bus peripherals (including memories).

Address ranges and thus address mapping of memories or peripherals on X-Bus are controlled with the address selection registers **XADRSx**. The respective bus type definitions are controlled with registers **XBCONx**.

#### 6.5.2.1.1 X-Bus Address Selection Registers

**XADRSx**

**XADRS1**

**XADRS2**

**XADRS3**

**XADRS4**

**XADRS5**

**XADRS6**

**X-Bus Address Selection Registers**

(Reset value: XXXX<sub>H</sub>)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RGSADx												RGSZx			

Field	Bits	Type	Description
<b>RGSZx</b> (x = 1 to 6)	3:0	rw	<b>Address Range Start Address Selection</b>
<b>RGSADx</b> (x = 1 to 6)	15:4	rw	<b>Address Range Size Selection</b>

The respective SFR addresses of **XADRSx** registers can be found in list of SFRs, refer to **Section 12.2 PD-Bus Register Addresses (on Page 1273)**.

Due to the different range size options, address mapping of XPERs is possible only within the first MByte of the total address range if **XADRS1** to **XADRS4** is used. The upper four address lines (A23:A20) are set to zero.

*Note: The range start address can be located only on boundaries specified by the selected range size.*

**Table 6-31** shows the definitions of range size selections and range start addresses for the address selection registers **XADRSx**.

The address range and address range start definition of **XADRS5** and **XADRS6** registers is identical to the address selection definition for external devices (refer to

**CONFIDENTIAL**

**X-Bus System Architecture**

**External Memory Space (on Page 201).** It is thus possible to use the whole address range also for internal memories or peripherals.

**Table 6-31 Address Range and Address Range Start Definition of XADRx Registers**

Range Size RGSZ	Selected Address Range	Relevant(R) Bits of RGSAD	Selected Range Start Address (Relevant(R) Bits of RGSAD)
0000	256 Byte	RRRR RRRR RRRR	0000 RRRR RRRR RRRR 0000 0000
0001	512 Bytes	RRRR RRRR RRR0	0000 RRRR RRRR RRR0 0000 0000
0010	1kBytes	RRRR RRRR RR00	0000 RRRR RRRR RR00 0000 0000
0011	2kBytes	RRRR RRRR R000	0000 RRRR RRRR R000 0000 0000
0100	4kBytes	RRRR RRRR 0000	0000 RRRR RRRR 0000 0000 0000
0101	8kBytes	RRRR RRR0 0000	0000 RRRR RRR0 0000 0000 0000
0110	16kBytes	RRRR RR00 0000	0000 RRRR RR00 0000 0000 0000
0111	32kBytes	RRRR R000 0000	0000 RRRR R000 0000 0000 0000
1000	64kBytes	RRRR 0000 0000	0000 RRRR 0000 0000 0000 0000
1001	128kBytes	RRR0 0000 0000	0000 RRR0 0000 0000 0000 0000
1010	256kBytes	RR00 0000 0000	0000 RR00 0000 0000 0000 0000
1011	512kBytes	R000 0000 0000	0000 R000 0000 0000 0000 0000
11xx	- Reserved		

**CONFIDENTIAL**

**X-Bus System Architecture**

### 6.5.2.1.2 X-Bus Control Registers

All **XBCONx** registers are located in the bitaddressable ESFR memory space. The respective SFR addresses of **XBCONx** registers can be found in list of SFRs.

**XBCONx**

**XBCON1**

**XBCON2**

**XBCON3**

**XBCON4**

**XBCON5**

**XBCON6**

**X-Bus Control Register**

(Reset value: XXXX<sub>H</sub>)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			RDY EN	RE- SER VED	BUS ACT	RESERV ED		B TYP	RES ERV ED	1	1	MCTC <sub>x</sub>			

Field	Bits	Type	Description
<b>MCTC</b>	3:0	rw	<b>Memory Cycle Time Control</b> <sup>1)</sup> (Number of memory cycle time waitstates) 0000: 15 waitstates ... (Number = 15 - <MCTC>) 1111: No waitstates <b>Recommended value for E-GOLDradio is 1111.</b>
<b>1</b>	5:4	r	These bits are fixed to 1.
<b>BTYP</b>	7	rw	<b>X-Bus Type Definition</b> 0 8-bit Demultiplexed Bus 1 16-bit Demultiplexed Bus <b>Recommended value for E-GOLDradio is 1.</b>
<b>BUSACT</b>	10	rw	<b>X-Bus Active Control</b> 0 X-Bus (peripheral) disabled 1 X-Bus (peripheral) enabled Enables the X-Bus and the according chip select <u>XCS<sub>x</sub></u> for the respective address window (respective X-Bus peripheral), selected with <b>XADRS<sub>x</sub></b> window. <i>Note: Enable/Disable also is controlled with the SYSCON registers.</i>

**CONFIDENTIAL**

**X-Bus System Architecture**

Field	Bits	Type	Description
<b>RDYEN</b>	12	rw	<b>READY Enable</b> 0 The bus cycle length is controlled by bit field MCTC only 1 The bus cycle length is controlled by the peripheral using <u>READY</u> signal <b>Recommended value for E-GOLDradio is 1.</b>
<b>RESERVED</b>	15:13, 11, 9:8, 6	r	Reserved; these bits must be left at their reset values.

<sup>1)</sup> When the READY function is selected (**RDYEN** = 1), only the lower 3 bits of the respective MCTC bit field define the number of inserted wait states (0-7), while the MSB of bit field MCTC is unused

## **6.5.3 BUS Arbitration**

### **6.5.3.1 X-Bus**

The internal X-Bus does not use or support arbitration. The X-Bus is however blocked when a second external master has used arbitration to access the external bus.

### **6.5.3.2 External Bus**

Integrated systems with a second external bus master are supported by using the standard arbitration signals.

- HOLD input allows the waiting bus master to request the X-Bus from the active master.
- HLDA output (master) or input (slave) indicating that the bus is released by the master.
- BREQ bus request output.

**CONFIDENTIAL**

**Instruction Set**

## 6.6 Instruction Set

### 6.6.1 Instruction Summary

In the following sections the instructions are compiled according to different criteria to provide different levels of precision:

- The **Opcode Tables** ([Table 6-32](#) and [Table 6-33](#)) reference the instructions by their hexadecimal opcode with the respective mnemonic.
- The **Instruction Set Summary** ([Table 6-34](#) and [Table 6-35](#)) groups the individual instructions into functional groups. These tables list the instructions by their mnemonic and identifies the addressing modes that may be used with a specific instruction and the instruction length, which depends on the selected addressing mode. These tables help to optimize instruction sequences in terms of code size and/or execution time.
- The **Cross Reference Tables** ([Table 6-36](#), [Table 6-37](#), [Table 6-38](#), and [Table 6-39](#)) summarize all instructions in condensed tables. These tables quickly identify a specific instruction and provide basic information about it.

### 6.6.2 Opcode Tables

**Table 6-32 Instructions Referenced by Opcode (Part 1)**

•	0x	1x	2x	3x	4x	5x	6x	7x
x0	ADD	ADDC	SUB	SUBC	CMP	XOR	AND	OR
x1	ADDB	ADDCB	SUBB	SUBCB	CMPB	XORB	ANDB	ORB
x2	ADD	ADDC	SUB	SUBC	CMP	XOR	AND	OR
x3	ADDB	ADDCB	SUBB	SUBCB	CMPB	XORB	ANDB	ORB
x4	ADD	ADDC	SUB	SUBC	-	XOR	AND	OR
x5	ADDB	ADDCB	SUBB	SUBCB	-	XORB	ANDB	ORB
x6	ADD	ADDC	SUB	SUBC	CMP	XOR	AND	OR
x7	ADDB	ADDCB	SUBB	SUBCB	CMPB	XORB	ANDB	ORB
x8	ADD	ADDC	SUB	SUBC	CMP	XOR	AND	OR
x9	ADDB	ADDCB	SUBB	SUBCB	CMPB	XORB	ANDB	ORB
xA	BFLDL	BFLDH	BCMP	BMOVN	BMOV	BOR	BAND	BXOR
xB	MUL	MULU	PRIOR	-	DIV	DIVU	DIVL	DIVLU
xC	ROL	ROL	ROR	ROR	SHL	SHL	SHR	SHR
xD	JMPR	JMPR	JMPR	JMPR	JMPR	JMPR	JMPR	JMPR
xE	BCLR	BCLR	BCLR	BCLR	BCLR	BCLR	BCLR	BCLR
xF	BSET	BSET	BSET	BSET	BSET	BSET	BSET	BSET

**Table 6-33 Instructions Referenced by Opcode (Part 2)**

	<b>8x</b>	<b>9x</b>	<b>Ax</b>	<b>Bx</b>	<b>Cx</b>	<b>Dx</b>	<b>Ex</b>	<b>Fx</b>
x0	CMPI1	CMPI2	CMPD1	CMPD2	MOVBZ	MOVBS	MOV	MOV
x1	NEG	CPL	NEGB	CPLB	-	AT/DEB/ EXTR	MOVB	MOVB
x2	CMPI1	CMPI2	CMPD1	CMPD2	MOVBZ	MOVBS	PCALL	MOV
x3	-	-	-	-	-	-	-	MOVB
x4	MOV	MOV	MOVB	MOVB	MOV	MOV	MOVB	MOVB
x5	-	-	DIS WDT	EINIT	MOVBZ	MOVBS	-	-
x6	CMPI1	CMPI2	CMPD1	CMPD2	SCXT	SCXT	MOV	MOV
x7	IDLE	PWRDN	SRV WDT	SRST	-	EXTP/S/ R	MOVB	MOVB
x8	MOV	MOV	MOV	MOV	MOV	MOV	MOV	-
x9	MOVB	MOVB	MOVB	MOVB	MOVB	MOVB	MOVB	-
xA	JB	JNB	JBC	JNBS	CALLA	CALLS	JMPA	JMPS
xB	-	TRAP	CALLI	CALLR	RET	RETS	RETP	RETI
xC	-	JMPI	ASHR	ASHR	NOP	EXTP/S/ R	PUSH	POP
xD	JMPR	JMPR	JMPR	JMPR	JMPR	JMPR	JMPR	JMPR
xE	BCLR	BCLR	BCLR	BCLR	BCLR	BCLR	BCLR	BCLR
xF	BSET	BSET	BSET	BSET	BSET	BSET	BSET	BSET

### 6.6.3 Instruction Set Summary

This chapter summarizes the instructions by listing them according to their functional class. This allows to identify the right instruction(s) for a specific required function.

The following apply to this summary:

#### Data Addressing Modes

- Rw: – Word GPR (R0, R1, ... , R15)
- Rb: – Byte GPR (RL0, RH0, ..., RL7, RH7)
- reg: – SFR or GPR  
(in case of a byte operation on an SFR, only the low byte can be accessed via 'reg')
- mem: – Direct word or byte memory location
- [...]: – Indirect word or byte memory location  
(Any word GPR can be used as indirect address pointer, except for the arithmetic, logical and compare instructions, where only R0 to R3 are allowed)
- bitaddr: – Direct bit in the bit-addressable memory area
- bitoff: – Direct word in the bit-addressable memory area
- #data: – Immediate constant  
(The number of significant bits which can be specified by the user is represented by the respective appendix 'x')
- #mask8: – Immediate 8-bit mask used for bit-field modifications

#### Multiply and Divide Operations

The **MDL** and **MDH** registers are implicit source and/or destination operands of the multiply and divide instructions.

#### Branch Target Addressing Modes

- caddr: – Direct 16-bit jump target address (Updates the Instruction Pointer)
- seg: – Direct 2-bit segment address  
(Updates the Code Segment Pointer)
- rel: – Signed 8-bit jump target word offset address relative to the Instruction Pointer of the following instruction
- #trap7: – Immediate 7-bit trap or interrupt number.



**CONFIDENTIAL**

**Instruction Set**

**Extension Operations**

The EXT\* instructions override the standard DPP addressing scheme:

#pag10: – Immediate 10-bit page address.

#seg8: – Immediate 8-bit segment address.

**Branch Condition Codes**

cc:      Symbolically specifiable condition codes

cc_UC	–Unconditional
cc_Z	–Zero
cc_NZ	–Not Zero
cc_V	–Overflow
cc_NV	–No Overflow
cc_N	–Negative
cc_NN	–Not Negative
cc_C	–Carry
cc_NC	–No Carry
cc_EQ	–Equal
cc_NE	–Not Equal
cc_ULT	–Unsigned Less Than
cc_ULE	–Unsigned Less Than or Equal
cc_UGE	–Unsigned Greater Than or Equal
cc_UGT	–Unsigned Greater Than
cc_SLE	–Signed Less Than or Equal
cc_SGE	–Signed Greater Than or Equal
cc_SGT	–Signed Greater Than
cc_NET	–Not Equal and Not End-of-Table

**CONFIDENTIAL**

**Instruction Set**

**Table 6-34 Instruction Set**

Mnemonic	Addressing Modes Bytes			Mnemonic	Addressing Modes Bytes		
ADD[B]	Rwn	Rwm <sup>1)</sup>	2	CPL[B]	Rwn	<sup>1)</sup>	2
ADDC[B]	Rwn	[Rwi] <sup>1)</sup>	2	NEG[B]			
AND[B]	Rwn	[Rwi+] <sup>1)</sup>	2				
OR[B]	Rwn	#data3 <sup>1)</sup>	2	DIV	Rwn		2
SUB[B]				DIVL			
SUBC[B]	reg	#data16	4	DIVLU			
XOR[B]	reg	mem	4	DIVU			
	mem	reg	4				
ASHR	Rwn	Rwm	2	MUL	Rwn	Rwm	2
ROL / ROR	Rwn	#data4	2	MULU			
SHL / SHR				CMPD1/2	Rwn	#data4	2
				CMP11/2	Rwn	#data16	4
BAND	bitaddrZ.z	bitaddrQ.q	4		Rwn	mem	4
BCMP				CMP[B]	Rwn	Rwm <sup>1)</sup>	2
BMOV					Rwn	[Rwi] <sup>1)</sup>	2
BMOVN					Rwn	[Rwi+] <sup>1)</sup>	2
BOR / BXOR					Rwn	#data3 <sup>1)</sup>	2
					reg	#data16 <sup>2)</sup>	2
					reg	mem	4
							4
BCLR	bitaddrQ.q		2	CALLA	cc	caddr	4
BSET				JMPA			
BFLDH	bitoffQ	#mask8	4	CALLI	cc	[Rwn]	2
BFLDL	#data8			JMPI			
MOV[B]	Rwn	Rwm <sup>1)</sup>	2	CALLS	seg	caddr	4
	Rwn	#data4 <sup>1)</sup>	2	JMPS			
	Rwn	[Rwm] <sup>1)</sup>	2				
	Rwn	[Rwm+] <sup>1)</sup>	2	CALLR	rel		2
	[Rwm]	Rwn <sup>1)</sup>	2				
	[-Rwm]	Rwn <sup>1)</sup>	2	JMPR	cc	rel	2
	[Rwn]	[Rwm]	2				
	[Rwn+]	[Rwm]	2	JB	bitaddrQ.q	rel	4
	[Rwn]	[Rwm+]	2	JBC			
				JNB			
	reg	#data16 <sup>2)</sup>	4	JNBS			
	Rwn	[Rwm+#d16] <sup>1)</sup>	4	PCALL	reg	caddr	4
	[Rwm+#d16]	Rwn <sup>1)</sup>	4				
	[Rwn]	mem	4	POP	reg		2
	mem	[Rwn]	4	PUSH			
	reg	mem	4	RETP			
	mem	reg	4				
				SCXT	reg	#data16	4
					reg	mem	4
MOVBS	Rwn	Rbm	2	PRIOR	Rwn	Rwm	2
MOVBZ	reg	mem	4	TRAP	#trap7		2
	mem	reg	4	ATOMIC	#irang2		2
				EXTR			
EXTS	Rwm	#irang2	2	EXTP	Rwm	#irang2	2
EXTSR	#seg	#irang2	4	EXTPR	#pag	#irang2	4
NOP	-		2	SRST/IDLE	-		4
DEBUG				PWRDN			
RET				SRVWDT			
RETI				DISWDT			
RETS				EINIT			

**CONFIDENTIAL**

**Instruction Set**

<sup>1)</sup> Byte oriented instructions (suffix 'B') use Rb instead of Rw (not with [Rwn]!).

<sup>2)</sup> Byte oriented instructions (suffix 'B') use #data8 instead of #data16.

**Table 6-35 Instruction Set Summary**

Mnemonic		Description	Bytes
<b>Arithmetic Operations</b>			
ADD	Rw, Rw	Add direct word GPR to direct GPR	2
ADD	Rw, [Rw]	Add indirect word memory to direct GPR	2
ADD	Rw, [Rw +]	Add indirect word memory to direct GPR and post-increment source pointer by 2	2
ADD	Rw, #data3	Add immediate word data to direct GPR	2
ADD	reg, #data16	Add immediate word data to direct register	4
ADD	reg, mem	Add direct word memory to direct register	4
ADD	mem, reg	Add direct word register to direct memory	4
ADDB	Rb, Rb	Add direct byte GPR to direct GPR	2
ADDB	Rb, [Rw]	Add indirect byte memory to direct GPR	2
ADDB	Rb, [Rw +]	Add indirect byte memory to direct GPR and post-increment source pointer by 1	2
ADDB	Rb, #data3	Add immediate byte data to direct GPR	2
ADDB	reg, #data8	Add immediate byte data to direct register	4
ADDB	reg, mem	Add direct byte memory to direct register	4
ADDB	mem, reg	Add direct byte register to direct memory	4
ADDC	Rw, Rw	Add direct word GPR to direct GPR with Carry	2
ADDC	Rw, [Rw]	Add indirect word memory to direct GPR with Carry	2
ADDC	Rw, [Rw +]	Add indirect word memory to direct GPR with Carry and post-increment source pointer by 2	2
ADDC	Rw, #data3	Add immediate word data to direct GPR with Carry	2
ADDC	reg, #data16	Add immediate word data to direct register with Carry	4
ADDC	reg, mem	Add direct word memory to direct register with Carry	4
ADDC	mem, reg	Add direct word register to direct memory with Carry	4
ADDCB	Rb, Rb	Add direct byte GPR to direct GPR with Carry	2
ADDCB	Rb, [Rw]	Add indirect byte memory to direct GPR with Carry	2
ADDCB	Rb, [Rw +]	Add indirect byte memory to direct GPR with Carry and post-increment source pointer by 1	2
ADDCB	Rb, #data3	Add immediate byte data to direct GPR with Carry	2
ADDCB	reg, #data8	Add immediate byte data to direct register with Carry	4

**CONFIDENTIAL**

**Instruction Set**

**Table 6-35 Instruction Set Summary (cont'd)**

<b>Mnemonic</b>		<b>Description</b>	<b>Bytes</b>
ADDCB	reg, mem	Add direct byte memory to direct register with Carry	4
ADDCB	mem, reg	Add direct byte register to direct memory with Carry	4
SUB	Rw, Rw	Subtract direct word GPR from direct GPR	2
SUB	Rw, [Rw]	Subtract indirect word memory from direct GPR	2
SUB	Rw, [Rw +]	Subtract indirect word memory from direct GPR and post-increment source pointer by 2	2
SUB	Rw, #data3	Subtract immediate word data from direct GPR	2
SUB	reg, #data16	Subtract immediate word data from direct register	4
SUB	reg, mem	Subtract direct word memory from direct register	4
SUB	mem, reg	Subtract direct word register from direct memory	4
SUBB	Rb, Rb	Subtract direct byte GPR from direct GPR	2
SUBB	Rb, [Rw]	Subtract indirect byte memory from direct GPR	2
SUBB	Rb, [Rw +]	Subtract indirect byte memory from direct GPR and post-increment source pointer by 1	2
SUBB	Rb, #data3	Subtract immediate byte data from direct GPR	2
SUBB	reg, #data8	Subtract immediate byte data from direct register	4
SUBB	reg, mem	Subtract direct byte memory from direct register	4
SUBB	mem, reg	Subtract direct byte register from direct memory	4
SUBC	Rw, Rw	Subtract direct word GPR from direct GPR with Carry	2
SUBC	Rw, [Rw]	Subtract indirect word memory from direct GPR with Carry	2
SUBC	Rw, [Rw +]	Subtract indirect word memory from direct GPR with Carry and post-increment source pointer by 2	2
SUBC	Rw, #data3	Subtract immediate word data from direct GPR with Carry	2
SUBC	reg, #data16	Subtract immediate word data from direct register with Carry	4
SUBC	reg, mem	Subtract direct word memory from direct register with Carry	4
SUBC	mem, reg	Subtract direct word register from direct memory with Carry	4
SUBCB	Rb, Rb	Subtract direct byte GPR from direct GPR with Carry	2
SUBCB	Rb, [Rw]	Subtract indirect byte memory from direct GPR with Carry	2
SUBCB	Rb, [Rw +]	Subtract indirect byte memory from direct GPR with Carry and post-increment source pointer by 1	2
SUBCB	Rb, #data3	Subtract immediate byte data from direct GPR with Carry	2

**CONFIDENTIAL**

**Instruction Set**

**Table 6-35 Instruction Set Summary (cont'd)**

<b>Mnemonic</b>		<b>Description</b>	<b>Bytes</b>
SUBCB	reg, #data8	Subtract immediate byte data from direct register with Carry	4
SUBCB	reg, mem	Subtract direct byte memory from direct register with Carry	4
SUBCB	mem, reg	Subtract direct byte register from direct memory with Carry	4
MUL	Rw, Rw	Signed multiply direct GPR by direct GPR (16-/16-bit)	2
MULU	Rw, Rw	Unsigned multiply direct GPR by direct GPR (16-/16-bit)	2
DIV	Rw	Signed divide register MDL by direct GPR (16-/16-bit)	2
DIVL	Rw	Signed long divide register MD by direct GPR (32-/16-bit)	2
DIVLU	Rw	Unsigned long divide register MD by direct GPR (32-/16-bit)	2
DIVU	Rw	Unsigned divide register MDL by direct GPR (16-/16-bit)	2
CPL	Rw	Complement direct word GPR	2
CPLB	Rb	Complement direct byte GPR	2
NEG	Rw	Negate direct word GPR	2
NEGB	Rb	Negate direct byte GPR	2
<b>Logical Instructions</b>			
AND	Rw, Rw	Bitwise AND direct word GPR with direct GPR	2
AND	Rw, [Rw]	Bitwise AND indirect word memory with direct GPR	2
AND	Rw, [Rw +]	Bitwise AND indirect word memory with direct GPR and post-increment source pointer by 2	2
AND	Rw, #data3	Bitwise AND immediate word data with direct GPR	2
AND	reg, #data16	Bitwise AND immediate word data with direct register	4
AND	reg, mem	Bitwise AND direct word memory with direct register	4
AND	mem, reg	Bitwise AND direct word register with direct memory	4
ANDB	Rb, Rb	Bitwise AND direct byte GPR with direct GPR	2
ANDB	Rb, [Rw]	Bitwise AND indirect byte memory with direct GPR	2
ANDB	Rb, [Rw +]	Bitwise AND indirect byte memory with direct GPR and post-increment source pointer by 1	2
ANDB	Rb, #data3	Bitwise AND immediate byte data with direct GPR	2
ANDB	reg, #data8	Bitwise AND immediate byte data with direct register	4
ANDB	reg, mem	Bitwise AND direct byte memory with direct register	4
ANDB	mem, reg	Bitwise AND direct byte register with direct memory	4
OR	Rw, Rw	Bitwise OR direct word GPR with direct GPR	2
OR	Rw, [Rw]	Bitwise OR indirect word memory with direct GPR	2

**CONFIDENTIAL**

**Instruction Set**

**Table 6-35 Instruction Set Summary (cont'd)**

<b>Mnemonic</b>		<b>Description</b>	<b>Bytes</b>
OR	Rw, [Rw +]	Bitwise OR indirect word memory with direct GPR and post-increment source pointer by 2	2
OR	Rw, #data3	Bitwise OR immediate word data with direct GPR	2
OR	reg, #data16	Bitwise OR immediate word data with direct register	4
OR	reg, mem	Bitwise OR direct word memory with direct register	4
OR	mem, reg	Bitwise OR direct word register with direct memory	4
ORB	Rb, Rb	Bitwise OR direct byte GPR with direct GPR	2
ORB	Rb, [Rw]	Bitwise OR indirect byte memory with direct GPR	2
ORB	Rb, [Rw +]	Bitwise OR indirect byte memory with direct GPR and post-increment source pointer by 1	2
ORB	Rb, #data3	Bitwise OR immediate byte data with direct GPR	2
ORB	reg, #data8	Bitwise OR immediate byte data with direct register	4
ORB	reg, mem	Bitwise OR direct byte memory with direct register	4
ORB	mem, reg	Bitwise OR direct byte register with direct memory	4
XOR	Rw, Rw	Bitwise XOR direct word GPR with direct GPR	2
XOR	Rw, [Rw]	Bitwise XOR indirect word memory with direct GPR	2
XOR	Rw, [Rw +]	Bitwise XOR indirect word memory with direct GPR and post-increment source pointer by 2	2
XOR	Rw, #data3	Bitwise XOR immediate word data with direct GPR	2
XOR	reg, #data16	Bitwise XOR immediate word data with direct register	4
XOR	reg, mem	Bitwise XOR direct word memory with direct register	4
XOR	mem, reg	Bitwise XOR direct word register with direct memory	4
XORB	Rb, Rb	Bitwise XOR direct byte GPR with direct GPR	2
XORB	Rb, [Rw]	Bitwise XOR indirect byte memory with direct GPR	2
XORB	Rb, [Rw +]	Bitwise XOR indirect byte memory with direct GPR and post-increment source pointer by 1	2
XORB	Rb, #data3	Bitwise XOR immediate byte data with direct GPR	2
XORB	reg, #data8	Bitwise XOR immediate byte data with direct register	4
XORB	reg, mem	Bitwise XOR direct byte memory with direct register	4
XORB	mem, reg	Bitwise XOR direct byte register with direct memory	4

**Boolean Bit Manipulation Operations**

BCLR	bitaddr	Clear direct bit	2
BSET	bitaddr	Set direct bit	2
BMOV	bitaddr, bitaddr	Move direct bit to direct bit	4

**CONFIDENTIAL**

**Instruction Set**

**Table 6-35 Instruction Set Summary (cont'd)**

<b>Mnemonic</b>		<b>Description</b>	<b>Bytes</b>
BMOVN	bitaddr, bitaddr	Move negated direct bit to direct bit	4
BAND	bitaddr, bitaddr	AND direct bit with direct bit	4
BOR	bitaddr, bitaddr	OR direct bit with direct bit	4
BXOR	bitaddr, bitaddr	XOR direct bit with direct bit	4
BCMP	bitaddr, bitaddr	Compare direct bit to direct bit	4
BFLDH	bitoff, #mask8, #data8	Bitwise modify masked high byte of bit-addressable direct word memory with immediate data	4
BFLDL	bitoff, #mask8, #data8	Bitwise modify masked low byte of bit-addressable direct word memory with immediate data	4
CMP	Rw, Rw	Compare direct word GPR to direct GPR	2
CMP	Rw, [Rw]	Compare indirect word memory to direct GPR	2
CMP	Rw, [Rw +]	Compare indirect word memory to direct GPR and post-increment source pointer by 2	2
CMP	Rw, #data3	Compare immediate word data to direct GPR	2
CMP	reg, #data16	Compare immediate word data to direct register	4
CMP	reg, mem	Compare direct word memory to direct register	4
CMPB	Rb, Rb	Compare direct byte GPR to direct GPR	2
CMPB	Rb, [Rw]	Compare indirect byte memory to direct GPR	2
CMPB	Rb, [Rw +]	Compare indirect byte memory to direct GPR and post-increment source pointer by 1	2
CMPB	Rb, #data3	Compare immediate byte data to direct GPR	2
CMPB	reg, #data8	Compare immediate byte data to direct register	4
CMPB	reg, mem	Compare direct byte memory to direct register	4
<b>Compare and Loop Control Instructions</b>			
CPMD1	Rw, #data4	Compare immediate word data to direct GPR and decrement GPR by 1	2
CPMD1	Rw, #data16	Compare immediate word data to direct GPR and decrement GPR by 1	4
CPMD1	Rw, mem	Compare direct word memory to direct GPR and decrement GPR by 1	4
CPMD2	Rw, #data4	Compare immediate word data to direct GPR and decrement GPR by 2	2
CPMD2	Rw, #data16	Compare immediate word data to direct GPR and decrement GPR by 2	4

**CONFIDENTIAL**

**Instruction Set**

**Table 6-35 Instruction Set Summary (cont'd)**

<b>Mnemonic</b>		<b>Description</b>	<b>Bytes</b>
COMPD2	Rw, mem	Compare direct word memory to direct GPR and decrement GPR by 2	4
CMPI1	Rw, #data4	Compare immediate word data to direct GPR and increment GPR by 1	2
CMPI1	Rw, #data16	Compare immediate word data to direct GPR and increment GPR by 1	4
CMPI1	Rw, mem	Compare direct word memory to direct GPR and increment GPR by 1	4
CMPI2	Rw, #data4	Compare immediate word data to direct GPR and increment GPR by 2	2
CMPI2	Rw, #data16	Compare immediate word data to direct GPR and increment GPR by 2	4
CMPI2	Rw, mem	Compare direct word memory to direct GPR and increment GPR by 2	4
<b>Prioritize Instruction</b>			
PRIOR	Rw, Rw	Determine number of shift cycles to normalize direct word GPR and store result in direct word GPR	2
<b>Shift and Rotate Instructions</b>			
SHL	Rw, Rw	Shift left direct word GPR; number of shift cycles specified by direct GPR	2
SHL	Rw, #data4	Shift left direct word GPR; number of shift cycles specified by immediate data	2
SHR	Rw, Rw	Shift right direct word GPR; number of shift cycles specified by direct GPR	2
SHR	Rw, #data4	Shift right direct word GPR; number of shift cycles specified by immediate data	2
ROL	Rw, Rw	Rotate left direct word GPR; number of shift cycles specified by direct GPR	2
ROL	Rw, #data4	Rotate left direct word GPR; number of shift cycles specified by immediate data	2
ROR	Rw, Rw	Rotate right direct word GPR; number of shift cycles specified by direct GPR	2
ROR	Rw, #data4	Rotate right direct word GPR; number of shift cycles specified by immediate data	2
ASHR	Rw, Rw	Arithmetic (sign bit) shift right direct word GPR; number of shift cycles specified by direct GPR	2



**CONFIDENTIAL**

**Instruction Set**

**Table 6-35 Instruction Set Summary (cont'd)**

<b>Mnemonic</b>		<b>Description</b>	<b>Bytes</b>
ASHR	Rw, #data4	Arithmetic (sign bit) shift right direct word GPR; number of shift cycles specified by immediate data	2
<b>Data Movement</b>			
MOV	Rw, Rw	Move direct word GPR to direct GPR	2
MOV	Rw, #data4	Move immediate word data to direct GPR	2
MOV	reg, #data16	Move immediate word data to direct register	4
MOV	Rw, [Rw]	Move indirect word memory to direct GPR	2
MOV	Rw, [Rw +]	Move indirect word memory to direct GPR and post-increment source pointer by 2	2
MOV	[Rw], Rw	Move direct word GPR to indirect memory	2
MOV	[-Rw], Rw	Pre-decrement destination pointer by 2 and move direct word GPR to indirect memory	2
MOV	[Rw], [Rw]	Move indirect word memory to indirect memory	2
MOV	[Rw +], [Rw]	Move indirect word memory to indirect memory and post-increment destination pointer by 2	2
MOV	[Rw], [Rw +]	Move indirect word memory to indirect memory and post-increment source pointer by 2	2
MOV	Rw, [Rw + #data16]	Move indirect word memory by base plus constant to direct GPR	4
MOV	[Rw + #data16], Rw	Move direct word GPR to indirect memory by base plus constant	4
MOV	[Rw], mem	Move direct word memory to indirect memory	4
MOV	mem, [Rw]	Move indirect word memory to direct memory	4
MOV	reg, mem	Move direct word memory to direct register	4
MOV	mem, reg	Move direct word register to direct memory	4
MOVB	Rb, Rb	Move direct byte GPR to direct GPR	2
MOVB	Rb, #data4	Move immediate byte data to direct GPR	2
MOVB	reg, #data8	Move immediate byte data to direct register	4
MOVB	Rb, [Rw]	Move indirect byte memory to direct GPR	2
MOVB	Rb, [Rw +]	Move indirect byte memory to direct GPR and post-increment source pointer by 1	2
MOVB	[Rw], Rb	Move direct byte GPR to indirect memory	2
MOVB	[-Rw], Rb	Pre-decrement destination pointer by 1 and move direct byte GPR to indirect memory	2
MOVB	[Rw], [Rw]	Move indirect byte memory to indirect memory	2

**CONFIDENTIAL**

**Instruction Set**

**Table 6-35 Instruction Set Summary (cont'd)**

<b>Mnemonic</b>		<b>Description</b>	<b>Bytes</b>
MOVB	[Rw +], [Rw]	Move indirect byte memory to indirect memory and post-increment destination pointer by 1	2
MOVB	[Rw], [Rw +]	Move indirect byte memory to indirect memory and post-increment source pointer by 1	2
MOVB	Rb, [Rw + #data16]	Move indirect byte memory by base plus constant to direct GPR	4
MOVB	[Rw + #data16], Rb	Move direct byte GPR to indirect memory by base plus constant	4
MOVB	[Rw], mem	Move direct byte memory to indirect memory	4
MOVB	mem, [Rw]	Move indirect byte memory to direct memory	4
MOVB	reg, mem	Move direct byte memory to direct register	4
MOVB	mem, reg	Move direct byte register to direct memory	4
MOVBS	Rw, Rb	Move direct byte GPR with sign extension to direct word GPR	2
MOVBS	reg, mem	Move direct byte memory with sign extension to direct word register	4
MOVBS	mem, reg	Move direct byte register with sign extension to direct word memory	4
MOVBS	Rw, Rb	Move direct byte GPR with zero extension to direct word GPR	2
MOVBS	reg, mem	Move direct byte memory with zero extension to direct word register	4
MOVBS	mem, reg	Move direct byte register with zero extension to direct word memory	4

**Jump and Call Operations**

JMPA	cc, caddr	Jump absolute if condition is met	4
JMPI	cc, [Rw]	Jump indirect if condition is met	2
JMPR	cc, rel	Jump relative if condition is met	2
JMPS	seg, caddr	Jump absolute to a code segment	4
JB	bitaddr, rel	Jump relative if direct bit is set	4
JBC	bitaddr, rel	Jump relative and clear bit if direct bit is set	4
JNB	bitaddr, rel	Jump relative if direct bit is not set	4
JNBS	bitaddr, rel	Jump relative and set bit if direct bit is not set	4
CALLA	cc, caddr	Call absolute subroutine if condition is met	4
CALLI	cc, [Rw]	Call indirect subroutine if condition is met	2

**CONFIDENTIAL**

**Instruction Set**

**Table 6-35 Instruction Set Summary (cont'd)**

<b>Mnemonic</b>	<b>Description</b>	<b>Bytes</b>
CALLR    rel	Call relative subroutine	2
CALLS    seg, caddr	Call absolute subroutine in any code segment	4
PCALL    reg, caddr	Push direct word register onto system stack and call absolute subroutine	4
TRAP    #trap7	Call interrupt service routine via immediate trap number	2
<b>System Stack Operations</b>		
POP    reg	Pop direct word register from system stack	2
PUSH    reg	Push direct word register onto system stack	2
SCXT    reg, #data16	Push direct word register onto system stack and update register with immediate data	4
SCXT    reg, mem	Push direct word register onto system stack and update register with direct memory	4
<b>Return Operations</b>		
RET	Return from intra-segment subroutine	2
RETS	Return from inter-segment subroutine	2
RETP    reg	Return from intra-segment subroutine and pop direct word register from system stack	2
RETI	Return from interrupt service subroutine	2
<b>System Control</b>		
SRST	Software Reset	4
IDLE	Enter Idle Mode	4
PWRDN	Enter Power Down Mode (supposes NMI-pin being low)	4
SRVWDT	Service Watchdog Timer	4
DISWDT	Disable Watchdog Timer	4
EINIT	Signify End-of-Initialization on RSTOUT-pin	4
ATOMIC    #irang2	Begin ATOMIC sequence <sup>*)</sup>	2
EXTR    #irang2	Begin EXTended Register sequence <sup>*)</sup>	2
EXTP    Rw, #irang2	Begin EXTended Page sequence <sup>*)</sup>	2
EXTP    #pag10, #irang2	Begin EXTended Page sequence <sup>*)</sup>	4
EXTPR    Rw, #irang2	Begin EXTended Page and Register sequence <sup>*)</sup>	2
EXTPR    #pag10, #irang2	Begin EXTended Page and Register sequence <sup>*)</sup>	4
EXTS    Rw, #irang2	Begin EXTended Segment sequence <sup>*)</sup>	2
EXTS    #seg8, #irang2	Begin EXTended Segment sequence <sup>*)</sup>	4

**CONFIDENTIAL**

**Instruction Set**

**Table 6-35 Instruction Set Summary (cont'd)**

Mnemonic	Description	Bytes
EXTSR   Rw, #irang2	Begin EXTended Segment and Register sequence <sup>*)</sup>	2
EXTSR   #seg8, #irang2	Begin EXTended Segment and Register sequence <sup>*)</sup>	4
<b>Miscellaneous Operations</b>		
NOP	Null operation	2
DEBUG	Generates debug trigger event	2

### 6.6.4 Instruction Opcodes

The following pages list the instructions of the C166S ordered by their hexadecimal opcodes. This helps to identify specific instructions when reading executable code, that is, during the debugging phase.

#### For Opcode Lists

- These instructions are encoded by means of additional bits in the operand field of the instruction

$x0_H - x7_H$ :Rw, #data3or   Rb, #data3  
 $x8_H - xB_H$ :   Rw, [Rw]   or   Rb, [Rw]  
 $xC_H - xF_H$ :   Rw, [Rw +]   or   Rb, [Rw +]

For these instructions only the lowest four GPRs, R0 to R3, can be used as indirect address pointers.

- These instructions are encoded by means of additional bits in the operand field of the instruction:

00   xx.xxxx<sub>B</sub>:   EXTS   or   ATOMIC  
 01   xx.xxxx<sub>B</sub>:   EXTP   or   DEBUG  
 10   xx.xxxx<sub>B</sub>:   EXTSR   or   EXTR  
 11   xx.xxxx<sub>B</sub>:   EXTPR

#### For JMPR Instructions

The condition code to be tested for the JMPR instructions is specified by the opcode. Two mnemonic representation alternatives exist for some of the condition codes.

#### For BCLR and BSET Instructions

The position of the bit to be set or to be cleared is specified by the opcode. The operand 'bitoff.n' (n = 0 to 15) refers to a particular bit within a bit-addressable word.

#### For Undefined Opcodes

A hardware trap occurs when one of the undefined opcodes signified by '----' is decoded by the MCU.

**Table 6-36 Mnemonics, Operands (Part 1)**

Hex-code	Num-ber of Bytes	Mnemonic	Operands	Hex-code	Num-ber of Bytes	Mnemonic	Operands
00	2	ADD	Rw, Rw	20	2	SUB	Rw, Rw
01	2	ADDB	Rb, Rb	21	2	SUBB	Rb, Rb
02	4	ADD	reg, mem	22	4	SUB	reg, mem
03	4	ADDB	reg, mem	23	4	SUBB	reg, mem
04	4	ADD	mem, reg	24	4	SUB	mem, reg
05	4	ADDB	mem, reg	25	4	SUBB	mem, reg
06	4	ADD	reg, #data16	26	4	SUB	reg, #data16
07	4	ADDB	reg, #data8	27	4	SUBB	reg, #data8
08	2	ADD	Rw, [Rw +] or Rw, [Rw] or Rw, #data3	28	2	SUB	Rw, [Rw +] or Rw, [Rw] or Rw, #data3
09	2	ADDB	Rb, [Rw +] or Rb, [Rw] or Rb, #data3	29	2	SUBB	Rb, [Rw +] or Rb, [Rw] or Rb, #data3
0A	4	BFLDL	bitoff, #mask8, #data8	2A	4	BCMP	bitaddr, bitaddr
0B	2	MUL	Rw, Rw	2B	2	PRIOR	Rw, Rw
0C	2	ROL	Rw, Rw	2C	2	ROR	Rw, Rw
0D	2	JMPR	cc_UC, rel	2D	2	JMPR	cc_EQ, rel or cc_Z, rel
0E	2	BCLR	bitoff.0	2E	2	BCLR	bitoff.2
0F	2	BSET	bitoff.0	2F	2	BSET	bitoff.2
10	2	ADDC	Rw, Rw	30	2	SUBC	Rw, Rw
11	2	ADDCB	Rb, Rb	31	2	SUBCB	Rb, Rb
12	4	ADDC	reg, mem	32	4	SUBC	reg, mem
13	4	ADDCB	reg, mem	33	4	SUBCB	reg, mem
14	4	ADDC	mem, reg	34	4	SUBC	mem, reg
15	4	ADDCB	mem, reg	35	4	SUBCB	mem, reg
16	4	ADDC	reg, #data16	36	4	SUBC	reg, #data16
17	4	ADDCB	reg, #data8	37	4	SUBCB	reg, #data8

**CONFIDENTIAL**

**Instruction Set**

**Table 6-36 Mnemonics, Operands (Part 1)**

Hex-code	Num-ber of Bytes	Mnemonic	Operands	Hex-code	Num-ber of Bytes	Mnemonic	Operands
18	2	ADDC	Rw, [Rw +] or Rw, [Rw] or Rw, #data3	38	2	SUBC	Rw, [Rw +] or Rw, [Rw] or Rw, #data3
19	2	ADDCB	Rb, [Rw +] or Rb, [Rw] or Rb, #data3	39	2	SUBCB	Rb, [Rw +] or Rb, [Rw] or Rb, #data3
1A	4	BFLDH	bitoff, #mask8, #data8	3A	4	BMOVN	bitaddr, bitaddr
1B	2	MULU	Rw, Rw	3B	-	-	-
1C	2	ROL	Rw, #data4	3C	2	ROR	Rw, #data4
1D	2	JMPR	cc_NET, rel	3D	2	JMPR	cc_NE, rel or cc_NZ, rel
1E	2	BCLR	bitoff.1	3E	2	BCLR	bitoff.3
1F	2	BSET	bitoff.1	3F	2	BSET	bitoff.3

**Table 6-37 Mnemonics, Operands (Part 2)**

Hex-code	Num-ber of Bytes	Mnemonic	Operands	Hex-code	Num-ber of Bytes	Mnemonic	Operands
40	2	CMP	Rw, Rw	60	2	AND	Rw, Rw
41	2	CMPB	Rb, Rb	61	2	ANDB	Rb, Rb
42	4	CMP	reg, mem	62	4	AND	reg, mem
43	4	CMPB	reg, mem	63	4	ANDB	reg, mem
44	-	-	-	64	4	AND	mem, reg
45	-	-	-	65	4	ANDB	mem, reg
46	4	CMP	reg, #data16	66	4	AND	reg, #data16
47	4	CMPB	reg, #data8	67	4	ANDB	reg, #data8
48	2	CMP	Rw, [Rw +] or Rw, [Rw] or Rw, #data3	68	2	AND	Rw, [Rw +] or Rw, [Rw] or Rw, #data3

**CONFIDENTIAL**

**Instruction Set**

**Table 6-37 Mnemonics, Operands (Part 2) (cont'd)**

Hex-code	Num-ber of Bytes	Mnemonic	Operands	Hex-code	Num-ber of Bytes	Mnemonic	Operands
49	2	CMPB	Rb, [Rw +] or Rb, [Rw] or Rb, #data3	69	2	ANDB	Rb, [Rw +] or Rb, [Rw] or Rb, #data3
4A	4	BMOV	bitaddr, bitaddr	6A	4	BAND	bitaddr, bitaddr
4B	2	DIV	Rw	6B	2	DIVL	Rw
4C	2	SHL	Rw, Rw	6C	2	SHR	Rw, Rw
4D	2	JMPR	cc_V, rel	6D	2	JMPR	cc_N, rel
4E	2	BCLR	bitoff.4	6E	2	BCLR	bitoff.6
4F	2	BSET	bitoff.4	6F	2	BSET	bitoff.6
50	2	XOR	Rw, Rw	70	2	OR	Rw, Rw
51	2	XORB	Rb, Rb	71	2	ORB	Rb, Rb
52	4	XOR	reg, mem	72	4	OR	reg, mem
53	4	XORB	reg, mem	73	4	ORB	reg, mem
54	4	XOR	mem, reg	74	4	OR	mem, reg
55	4	XORB	mem, reg	75	4	ORB	mem, reg
56	4	XOR	reg, #data16	76	4	OR	reg, #data16
57	4	XORB	reg, #data8	77	4	ORB	reg, #data8
58	2	XOR	Rw, [Rw +] or Rw, [Rw] or Rw, #data3	78	2	OR	Rw, [Rw +] or Rw, [Rw] or Rw, #data3 <sup>1)</sup>
59	2	XORB	Rb, [Rw +] or Rb, [Rw] or Rb, #data3	79	2	ORB	Rb, [Rw +] or Rb, [Rw] or Rb, #data3
5A	4	BOR	bitaddr, bitaddr	7A	4	BXOR	bitaddr, bitaddr
5B	2	DIVU	Rw	7B	2	DIVLU	Rw
5C	2	SHL	Rw, #data4	7C	2	SHR	Rw, #data4
5D	2	JMPR	cc_NV, rel	7D	2	JMPR	cc_NN, rel
5E	2	BCLR	bitoff.5	7E	2	BCLR	bitoff.7
5F	2	BSET	bitoff.5	7F	2	BSET	bitoff.7

**Table 6-38 Mnemonics, Operands (Part 3)**

Hex-code	Num-ber of Bytes	Mnemonic	Operands	Hex-code	Num-ber of Bytes	Mnemonic	Operands
80	2	CMPI1	Rw, #data4	A0	2	CMPD1	Rw, #data4
81	2	NEG	Rw	A1	2	NEGB	Rb
82	4	CMPI1	Rw, mem	A2	4	CMPD1	Rw, mem
83	4	CoXXX	xx	A3	4	CoXXX	xx
84	4	MOV	[Rw], mem	A4	4	MOVB	[Rw], mem
85	-	-	-	A5	4	DISWDT	
86	4	CMPI1	Rw, #data16	A6	4	CMPD1	Rw, #data16
87	4	IDLE		A7	4	SRVWDT	
88	2	MOV	[-Rw], Rw	A8	2	MOV	Rw, [Rw]
89	2	MOVB	[-Rw], Rb	A9	2	MOVB	Rb, [Rw]
8A	4	JB	bitaddr, rel	AA	4	JBC	bitaddr, rel
8B	-	-	-	AB	2	CALLI	cc, [Rw]
8C	-	-	-	AC	2	ASHR	Rw, Rw
8D	2	JMPR	cc_C, rel or cc_ULT, rel	AD	2	JMPR	cc_SGT, rel
8E	2	BCLR	bitoff.8	AE	2	BCLR	bitoff.10
8F	2	BSET	bitoff.8	AF	2	BSET	bitoff.10
90	2	CMPI2	Rw, #data4	B0	2	CMPD2	Rw, #data4
91	2	CPL	Rw	B1	2	CPLB	Rb
92	4	CMPI2	Rw, mem	B2	4	CMPD2	Rw, mem
93	4	-	-	B3	4	-	-
94	4	MOV	mem, [Rw]	B4	4	MOVB	mem, [Rw]
95	-	-	-	B5	4	EINIT	
96	4	CMPI2	Rw, #data16	B6	4	CMPD2	Rw, #data16
97	4	PWRDN		B7	4	SRST	
98	2	MOV	Rw, [Rw+]	B8	2	MOV	[Rw], Rw
99	2	MOVB	Rb, [Rw+]	B9	2	MOVB	[Rw], Rb
9A	4	JNB	bitaddr, rel	BA	4	JNBS	bitaddr, rel
9B	2	TRAP	#trap7	BB	2	CALLR	rel



**CONFIDENTIAL**

**Instruction Set**

**Table 6-38 Mnemonics, Operands (Part 3) (cont'd)**

Hex-code	Num-ber of Bytes	Mnemonic	Operands	Hex-code	Num-ber of Bytes	Mnemonic	Operands
9C	2	JMPI	cc, [Rw]	BC	2	ASHR	Rw, #data4
9D	2	JMPR	cc_NC, rel or cc_UGE, rel	BD	2	JMPR	cc_SLE, rel
9E	2	BCLR	bitoff.9	BE	2	BCLR	bitoff.11
9F	2	BSET	bitoff.9	BF	2	BSET	bitoff.11

**Table 6-39 Mnemonics, Operands (Part 4)**

Hex-code	Num-ber of Bytes	Mnemonic	Operands	Hex-code	Num-ber of Bytes	Mnemonic	Operands
C0	2	MOV BZ	Rw, Rb	E0	2	MOV	Rw, #data4
C1	-	-	-	E1	2	MOV B	Rb, #data4
C2	4	MOV BZ	reg, mem	E2	4	PCALL	reg, caddr
C3	4	-	-	E3	-	-	-
C4	4	MOV	[Rw+#data16], Rw	E4	4	MOV B	[Rw+#data16], Rb
C5	4	MOV BZ	mem, reg	E5	-	-	-
C6	4	SCXT	reg, #data16	E6	4	MOV	reg, #data16
C7	-	-	-	E7	4	MOV B	reg, #data8
C8	2	MOV	[Rw], [Rw]	E8	2	MOV	[Rw], [Rw+]
C9	2	MOV B	[Rw], [Rw]	E9	2	MOV B	[Rw], [Rw+]
CA	4	CALLA	cc, addr	EA	4	JMPA	cc, caddr
CB	2	RET		EB	2	RETP	reg
CC	2	NOP		EC	2	PUSH	reg
CD	2	JMPR	cc_SLT, rel	ED	2	JMPR	cc_UGT, rel
CE	2	BCLR	bitoff.12	EE	2	BCLR	bitoff.14
CF	2	BSET	bitoff.12	EF	2	BSET	bitoff.14
D0	2	MOVBS	Rw, Rb	F0	2	MOV	Rw, Rw

**CONFIDENTIAL**

**Instruction Set**

**Table 6-39 Mnemonics, Operands (Part 4) (cont'd)**

Hex-code	Num-ber of Bytes	Mnemonic	Operands	Hex-code	Num-ber of Bytes	Mnemonic	Operands
D1	2	ATOMIC/ DEBUG/ EXTR	#irang2	F1	2	MOVB	Rb, Rb
D2	4	MOVBS	reg, mem	F2	4	MOV	reg, mem
D3	4	-	-	F3	4	MOVB	reg, mem
D4	4	MOV	Rw, [Rw + #data16]	F4	4	MOVB	Rb, [Rw + #data16]
D5	4	MOVBS	mem, reg	F5	-	-	-
D6	4	SCXT	reg, mem	F6	4	MOV	mem, reg
D7	4	EXTP(R), EXTS(R)	#pag10, #irang 2 #seg8, #irang2	F7	4	MOVB	mem, reg
D8	2	MOV	[Rw+], [Rw]	F8	-	-	-
D9	2	MOVB	[Rw+], [Rw]	F9	-	-	-
DA	4	CALLS	seg, caddr	FA	4	JMPS	seg, caddr
DB	2	RETS		FB	2	RETI	
DC	2	EXTP(R), EXTS(R)	Rw, #irang2	FC	2	POP	reg
DD	2	JMPR	cc_SGE, rel	FD	2	JMPR	cc_ULE, rel
DE	2	BCLR	bitoff.13	FE	2	BCLR	bitoff.15
DF	2	BSET	bitoff.13	FF	2	BSET	bitoff.15

CONFIDENTIAL

SCU

## 6.7 SCU

### 6.7.1 SCU Registers Overview

Figure 6-17 Register Overview

Extended Power Management Registers	WDT Registers	ID Control Registers	External Interrupt Control Registers	Central System Control Registers
SYSCON1	WDTCON WDT	IDMANUF IDCHIP IDMEM IDPROG IDMEM2 CPUID SCUID	EXISEL EXICON	RP0H RP0L RSTCON  FOCON SCUSLC SCUSLS
SYSCON1	Bus Clock Control Register	EXISEL	External Interrupt Source Selection Register	
WDTCON	Watchdog Timer Control Register	EXICON	External Interrupt Control Register	
WDT	Watchdog Timer	RP0H	Port0 Start-Up Register	
IDMANUF	Manufacturer ID Register	RP0L	Port0 Start-Up Register	
IDCHIP	Chip Identification Register	RSTCON	Reset Control Register	
IDMEM	Memory Identification Register	FOCON	Frequency Output Control Register	
IDPROG	Programming ID Register	SCUSLC	Security Level Control Register of SCU	
IDMEM2	Memory Identification Register 2			
SCUID	ID Register of SCU	SCUSLS	Security Level Status Register of SCU	
CPUID	Cell based core ID			

### 6.7.2 Reset Control

The internal system reset function provides initialization of the C166S into a defined default state and is invoked either by asserting a hardware reset signal on pin `RESET_IN` (Hardware Reset Input), upon the execution of the `SRST` instruction (Software Reset) or by an overflow of the watchdog timer.

Whenever one of these conditions occurs, the microcontroller is reset into its predefined default state through an internal boot sequence. An external bus cycle is aborted, except for a watchdog reset ([Section 6.7.2.1.3 WDT Reset \(on Page 232\)](#)).

### 6.7.2.1 Reset Sources

For compatibility of system reset to Full-Custom C16x designs, the SCU differs and indicates (in WDT block) the reset types in [Table 6-40](#).

**Table 6-40 Reset Types**

Reset Type	Shortcut	Condition (Reset-Trigger)
Power-on Reset	PONR	RESET_IN, (asynchronous)
Watchdog Timer Reset	WDTR	WDT overflow (synchronous)
Software Reset	SWR	SRST command (synchronous)

#### 6.7.2.1.1 Power-on Reset

A power-on reset requires a low level on the RESET\_IN input. When RESET\_IN becomes inactive, the internal reset sequence is started. For SW and WDT reset, the sequence duration can be adjusted to 1024, 2048, or 4096 clk by programming the [RSTCON](#) register.

*Note: Every hardware reset will be detected as an PONR. The reset sequence starts only with an inactive external reset.*

#### 6.7.2.1.2 Software Reset

This reset type is generated and controlled in the core. The core provides a control signal `srst_dec` (8TCL) for the SCU to start the reset sequence and to indicate this reset type in the [WDTCON](#) register.

#### 6.7.2.1.3 WDT Reset

When the watchdog timer is not disabled during the initialization or serviced regularly during program execution, it will overflow and trigger the watchdog timer reset (signal to the core).

### 6.7.2.2 Reset Phases

#### 6.7.2.2.1 Pre-Reset-Phase

The pre-reset phase is started by one of the Reset Sources mentioned above. Depending on the reset source, some actions must be provided before starting the general system reset. This pre-reset phase is specially for a watchdog timer reset, important where all bus accesses must be finished (indicated by `eof_buscycle`) before

CONFIDENTIAL

SCU

resetting the system. The pre-reset phase is relevant only for the core and is controlled by the core itself.

#### 6.7.2.2.2 Reset-All-Phase

During the general system reset all system modules are reset. The general system reset phase stays for at least 5 clock cycles. Refer to [Table 6-41](#).

**Table 6-41 Values of Reset Signals in Reset-All-Phase**

Reset Signals	Value
ex_rstout_n	0
reset_n	0
wdt_reset_n	0: WDTR 1: other resets

#### 6.7.2.2.3 Reset-Sequence-Phase

The minimum reset duration of 1024 clk is determined by safe latching of startup/reset configuration from MON1 and MON2 pins. For SW and WDT resets it can be lengthen to 4096 clocks by [RSTCON](#) register.

The reset sequence is started with every reset type. As long as the reset sequence is running, short reset types (WDTR and SWR) are lengthened to the minimum reset duration. Refer to [Table 6-42](#).

**Table 6-42 Values of Reset Signals in Reset-Sequence-Phase**

Reset Signals	Value
ex_rstout_n	0
reset_n	0
wdt_reset_n	0: WDTR 1: other resets

#### 6.7.2.2.4 Reset-Wait-Phase

Not used in PMB7870.

#### **6.7.2.2.5 Init-Phase**

After hardware ramp-up the system is initialized by the boot program. Subsequently the first part of the user application software is executed. The End of Init-Phase is initiated by the command EINIT, refer to [Table 6-43](#).

**Table 6-43 Values of Reset Signals in Init-Phase**

<b>Reset Signals</b>	<b>Value</b>
ex_rstout_n	0
All other reset outputs	1

#### **6.7.2.2.6 User-Phase**

In User-Phase all Reset Signals are deactivated.

### 6.7.2.2.7 State Diagram

Figure 6-18 A Simplified Reset State Diagram of the Core

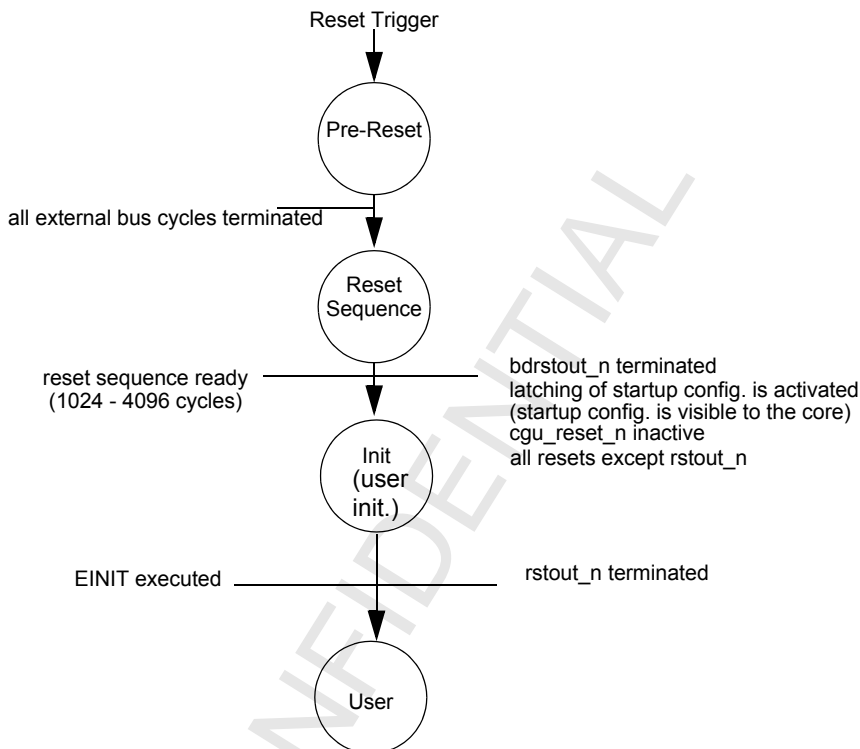
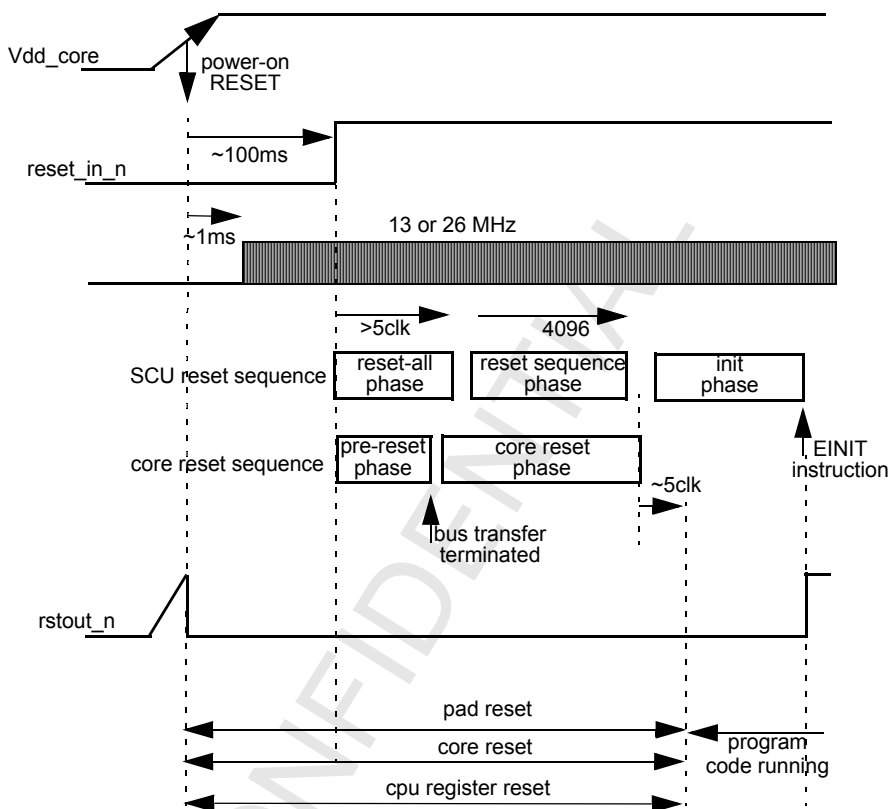


Figure 6-19 A Simplified Reset Sequence Timing Diagram





CONFIDENTIAL

SCU

### 6.7.2.3 Reset Sequence Control Register

**RSTCON**

**Reset Sequence Control**

**Power-On-Reset value: 0002<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														RSTLEN	

Field	Bits	Type	Description
<b>RSTLEN</b>	1:0	rw	<b>Reset Length Control</b> (duration of reset sequence) 00 1024 TCL 01 2048 TCL (doubled) 10 4096 TCL (standard reset sequence duration) 11 Reserved
<b>RESERVED</b>	15:2	r	Reserved; these bits must be left at their reset values.

*Note: **RSTCON** is only cleared with Long HW reset (including Power-on reset)*

*Note: **RSTCON** is a security register. The security level is automatically set to write protection after execution of EINIT.*

***RSTCON** can only be accessed via its long (mem) address.*

### 6.7.3 System Configuration Block

#### 6.7.3.1 System Startup Configuration Control

The system startup configuration is sampled and latched with a specific reset signal provided by the Reset Control Block of SCU. In general, for latching of the high byte of port P0 configuration the register **RP0H** is used, which controls the configuration. In E-GOLDradio, the reset value of **RP0H** cannot be overwritten, that is, the contents of **RP0H** is fixed by hardware.

**CONFIDENTIAL**

**SCU**

### 6.7.3.1.1 High Reset P0 Port Register

**RP0H**

**High Reset P0 Port**

**Reset value: 0017<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								RESERVED		SALSEL		CSSEL		WRC	

Field	Bits	Type	Description
<b>WRC</b>	0	r	<b>Write Configuration</b> (controlled and evaluated in <b>SYSCON</b> register) 0 No action 1 Standard function of pins WR and BHE
<b>CSSEL</b>	2:1	r	<b>Chip Select Line Selection</b> (Number of active $\overline{CS}$ outputs) 00 Not used 01 Not used 10 Not used 11 All $\overline{CS}$ lines: $\overline{CS4} \dots \overline{CS0}$
<b>SALSEL</b>	4:3	r	<b>Segment Address Line Selection</b> (Number of active segment address outputs) 00 Not used 01 Not used 10 Full segment address: A23 ... A16 11 Not used
<b>RESERVED</b>	7:5, 15:7	r	Reserved; these bits must be left at their reset values.

CONFIDENTIAL

SCU

### 6.7.3.1.2 Low Reset P0 Port Register

RP0L

Low Reset P0 Port

Reset value: 00000000X0111111<sub>B</sub><sup>1)</sup>  
Latching of Startup Configuration

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								BUSTYP		RESERVED					

<sup>1)</sup> These bits will be only reset with the start of every reset sequence. **RP0L.7** depends on the level of the MON1 pin during PO, SW, and WDT reset.

Field	Bits	Type	Description
<b>BUSTYP</b>	7:6	r	<b>External Data Bus Width</b> Description of possible selections: refer to <a href="#">Section 6.7.3.3 Code Definitions of Startup Configuration (on Page 241)</a>
<b>RESERVED</b>	0:5, 15:8	r	Reserved; these bits must be left at their reset values.

In PMB7870, the only variable for **RP0L** is the external bus width of 8 or 16 bit (RP0L.7). All the other RP0L bits will never change their reset values. The bus width is set via MON1 pin during reset (refer to [Chapter 14 System Reset](#)).

The latched value of **RP0L[7]** corresponds to the inverse level of MON1.

For latching of the P0 startup configuration (from the X-Bus) the reset types (refer to [Section 6.7.2.1 Reset Sources \(on Page 232\)](#)) are grouped as follows:

- Power-on reset (PONR)
- SW reset (SWR) or watchdog timer reset (WDTR)

**Table 6-44** shows how and in which register the system startup configuration is latched depending on the reset type and the bidirectional reset function.

**Table 6-44 System Startup Configuration Latching**

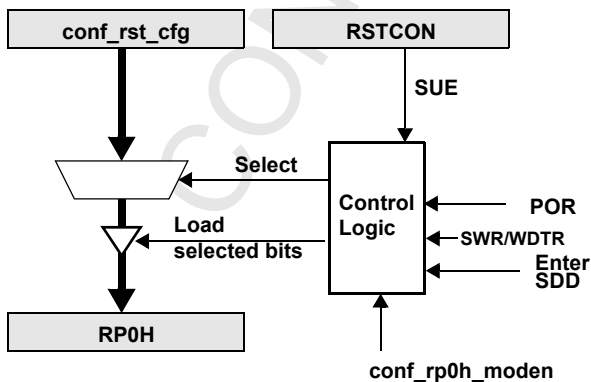
X:  Pin is sampled  -: Pin is not transparent and not sampled  Sample event	conf_rst_cfg															
	Clock options			Segm. Addr. Lines		Chip Selects		WR Config.	Bus Type							
	P0H.7	P0H.6	P0H.5	P0H.4	P0H.3	P0H.2	P0H.1	P0H.0	P0L.7	P0L.6	P0L.5	P0L.4	P0L.3	P0L.2	P0L.1	P0L.0
PONR	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
WDTR/SWR	-	-	-	X	X	X	X	X	X	X	X	-	-	-	-	-
	Programmable in <b>RSTCON</b>							SY SC ON	BUS CON0							
	Latched into <b>RP0H</b>							Latched into <b>RP0L</b>								

*Note: The configuration bits P0H[15:8] be are **not** transparent during the reset sequence phase. After the reset sequence phase, control of transparency and latching depends on the reset type.*

### 6.7.3.2 Internal or External Boot Configuration

The switching between internal and external boot mode is done by MON2 input pin.

**Figure 6-20 Reset Configuration Source Selection**



The PMB7870 latches the reset configuration from the conf\_rst\_cfg port. Signals conf\_rst\_cfg are hard-wired on module boundary by connection to V<sub>DD</sub> or V<sub>SS</sub>, except for bit[7], which switches the external bus width between 8 and 16 bits.

### 6.7.3.3 Code Definitions of Startup Configuration

The low order 6 bits of startup configuration define the system mode of operation. These bits are only **decoded in the core or in the product** for selection and control of the core related modes of operation, including the normal start after reset with instruction fetch from address 00 0000<sub>H</sub>. Also the configuration of bus type is sampled and latched only in the core (the bus controller belongs to core).

**Table 6-45** shows all possible selections for startup configuration and those used in E-GOLDradio. In this table, the note 1 is assigned to core related system modes of operation, which are controlled in the core and not in the SCU.

H.7	H.6	H.5	H.4	H.3	H.2	H.1	H.0	L.7	L.6	L.5	L.4	L.3	L.2	L.1	L.0
CLKCFG		SALSEL		CSSEL		WRC		BUSTYP		PROD (not used)					
0	0	0	1	0	1	1	1	MON 1	0	1	1	1	1	1	1

**Table 6-45 Code Definitions of Startup Configurations (conf\_rst\_cfg)**

Pin	Mode	Comment
BUSTYP (P0L.7:6)	External Data Bus Width <sup>1)</sup>	External Address Bus Mode
00	8-bit Data <sup>1)</sup>	Demultiplexed Addresses
01		Not Used
10	16-bit Data <sup>1)</sup>	Demultiplexed Addresses
11		Not Used
WRC	Write Configuration	Is always 1
CSSEL (P0H.1:2)	Chip Select Lines	
11	Max: CSx...CS0	Default without pull-downs
10		Not Used
01		Not Used
00		Not Used

**Table 6-45 Code Definitions of Startup Configurations (conf\_rst\_cfg)**

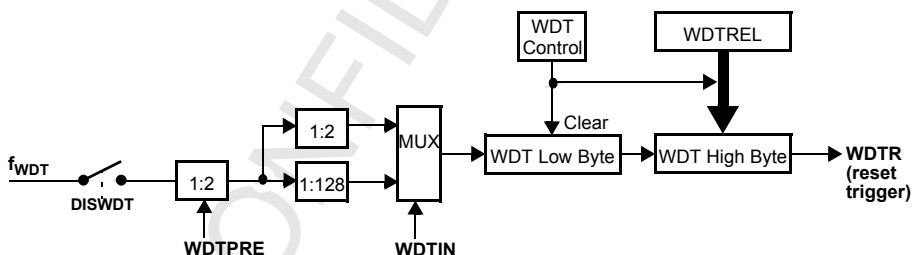
Pin	Mode	Comment
SALSEL (P0H.4:3)	Segment Address Lines	Directly accessible Address Space
11		Not Used
10	Axx...A16	(Maximum)
01		Not Used
00		Not Used

1) This operation mode is controlled by MON1 CB-core.

## 6.7.4 Watchdog Timer Submodule

The watchdog timer is a 16-bit up counter which can be clocked with the MCU clock ( $f_{CPU}$ ) divided by 2, 4, 128, or 256. This 16-bit timer is realized as two concatenated 8-bit timers (see [Figure 6-21](#)). The upper 8 bits of the watchdog timer can be preset to a user-programmable value via a watchdog service access to vary the watchdog expire time. The lower 8 bits are reset on each service access.

**Figure 6-21 WDT Block Diagram**



The current count value of the Watchdog Timer is contained in the Watchdog Timer Register **WDT**, which is a non-bitaddressable read-only register. The operation of the Watchdog Timer is controlled by its bitaddressable Watchdog Timer Control Register **WDTCON**. This register specifies the reload value for the high byte of the timer and selects the input clock prescaling factor.

After any software reset, external hardware reset (see next note), or watchdog timer reset, the watchdog timer is enabled and starts counting up from  $0000_H$  with the frequency  $f_{CPU}/2$ . The input frequency may be switched to  $f_{CPU}/128$  by setting bit **WDTCON.WDTIN**. The watchdog timer can be disabled via the instruction `DISWDT` (Disable Watchdog Timer). Instruction `DISWDT` is a protected 32-bit instruction which

CONFIDENTIAL

SCU

will ONLY be executed during the time between a reset and execution of either the EINIT (End of Initialization) or the SRVWDT (Service Watchdog Timer) instruction. Either one of these instructions disables the execution of DISWDT.

When the watchdog timer is not disabled via instruction DISWDT it will continue counting up, even during Idle Mode. If it is not serviced via the instruction SRVWDT by the time the count reaches  $FFFF_H$  the watchdog timer will overflow and cause an internal reset. This reset will pull the external reset indication pin `ex_rstout_n` to 0. The Watchdog Timer Reset Indication Flag **WDTCON.WDTR** is set in this case.

*Note: In Boot Mode the watchdog timer can be disabled by combining `diswdt_dec` and `bootmode` with an OR for compatibility to FC cores. Hence, no internal reset will be initiated.*

To prevent the watchdog timer from overflowing, it must be serviced periodically by the user software. The watchdog timer is serviced with the instruction SRVWDT, which is a protected 32-bit instruction. Servicing the watchdog timer clears the low byte and reloads the high byte of the watchdog time register **WDT** with the preset value in bit field **WDTCON.WDTREL**. Servicing the watchdog timer also reset bit **WDTCON.WDTR**. After being serviced the watchdog timer continues counting up from the value  $(\langle \text{WDTCON.WDTREL} \rangle * 28)$ . Instruction SRVWDT has been encoded in such a way that the chance of unintentionally servicing the watchdog timer (for example, by fetching and executing a bit pattern from a wrong location) is minimized. When instruction RVWDT does not match the format for protected instructions, the Protection Fault Trap is entered, instead of the instruction be executed.

The time period for an overflow of the watchdog timer is programmable in two ways:

1. **Input frequency** to the watchdog timer can be selected via bit **WDTCON.WDTIN** to be either  $f_{CPU}/2$  or  $f_{CPU}/128$ .
2. **RELOAD value** **WDTCON.WDTREL** can be programmed for the high byte of **WDT**.

The period  $P_{WDT}$  between servicing the watchdog timer and the next overflow can therefore be determined by the following formula:

$$P_{WDT} = \frac{2^{(1 + \langle \text{WDTIN} \rangle * 6)} * (2^{16} - \langle \text{WDTREL} \rangle * 2^8)}{f_{CPU}} \quad [0.2]$$

*Note: For safety reasons, the user is advised to rewrite **WDTCON** each time before the watchdog timer is serviced.*

As in C16x standard, each one of the different reset sources is indicated in the **WDTCON** register. The indicated reset bits are cleared with the EINIT instruction. It is thus possible to identify the reset reason (refer to [Table 6-46](#)) during the initialisation phase.

A new function is the indication of power-on reset. An external (to SCU) power-on detection is used to identify a HW reset as a power-on reset in the **WDTCON** register. Additionally, a fail safe feature is combined with the power-on detection: to protect the user's application in the case that a power-on reset is missing when the supply voltage

**CONFIDENTIAL**

**SCU**

is ramped up, the watch dog is enabled when power-on is detected in the dedicated external hardware. Then the watchdog timer will initiate a WDT reset after time out.

As in power down state, the Watchdog Timer is disabled during sleep mode.

A programmable prescaler for the Watchdog Timer is provided in **WDTCON** register to support high frequencies of system operation. For most applications the absolute running time of the WDT is more important than the counter value and, therefore, the timer resolution. With bit **WDTCON.WDTPRE** an additional prescaler can be selected. The combination of bits **WDTCON.WDTIN** and **WDTCON.WDTPRE** four prescaling factors are selectable (refer to **Table 6-46**).

**Table 6-46 WDT Prescaler Values**

<b>WDTIN</b>	<b>WDTPRE</b>	<b>Prescaler Factor</b>
0	0	2
0	1	4
1	0	128
1	1	256



CONFIDENTIAL

SCU

## 6.7.4.1 Watchdog Timer Registers

### 6.7.4.1.1 Watchdog Timer Control Register

The current value of bits 1, 2, and 5 depends on the reset source (refer to [Table 6-47](#)).

**Table 6-47 WDTCON Reset Sources**

Reset Source	Detection Bits That Are Set		
	PONR	SWR	WDTR
Power-On Reset	X	X	
Software Reset		X	
Watchdog Timer Reset		X	X

After an external PONR reset, the **WDTIN** and **WDTPRE** bits are reset to 0.

After a SRST or WDTR reset, the **WDTIN** and **WDTPRE** bits are not reset to 0 and retain their previous pre-reset values.

**Table 6-48 Reset Values of Bits 0 to 7, P = Previous Value**

Reset Type	Reset Bit Values
Power-On Reset	00xx xxx0
Software Reset	0Pxx xxxP
Watchdog Timer Reset	0Pxx xxxP

## WDTCON

### Watchdog Timer Control Register

Reset value: 00XX<sub>H</sub>(Refer to [Table 6-48](#))

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDTREL								WDT PRE	RES ERV ED	PON R	RES ERV ED	RES ERV ED	SWR	WDT R	WDT IN

Field	Bit	Type	Description
WDTIN	0	r	<b>Watchdog Timer Input Frequency Selection 1</b> This bit is used in conjunction with <b>WDTPRE</b> to select the Watchdog Timer Prescaler factor, refer to <a href="#">Table 6-46</a> .
WDTR <sup>1)</sup>	1	r	<b>Watchdog Timer Reset Indication Flag</b> Set by the watchdog timer on an overflow. Cleared by the SRVWDT instruction.

**CONFIDENTIAL**

**SCU**

<b>SWR<sup>1)</sup></b>	2	r	<b>Software Reset</b> Set by the command SRST <i>Note: Remains set because all types of resets set this bit.</i>
<b>Reserved</b>	3	r	<b>Not used</b> (Must not be evaluated by software)
<b>Reserved</b>	4	r	<b>Not used</b> (Must not be evaluated by software)
<b>PONR<sup>1)</sup></b>	5	r	<b>Power-On Reset</b> This equals HW Reset via reset_in_n. Set by Power-On Reset detection. <i>Note: Remains set until next reset (of any type).</i>
<b>WDTPRE</b>	7	rw	<b>Watchdog Timer Input Frequency Selection 2</b> This bit is used in conjunction with <b>WDTIN</b> to select the Watchdog Timer Prescaler factor, refer to <a href="#">Table 6-46</a> .
<b>WDTREL</b>	15:8	rw	<b>Watchdog Timer Reload Value</b> (For the high byte of WDT)
<b>RESERVED</b>	6	r	Reserved; these bits must be left at their reset values.

<sup>1)</sup> More than one reset source may be indicated (refer to [Table 6-47](#)). After EINIT all reset bits are cleared.

CONFIDENTIAL

SCU

### 6.7.4.1.2 Watchdog Timer Register

This register is a read-only register. Only during the test mode can a write access be performed on this register.

WDT

Watchdog Timer Register

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDT															

Field	Bit	Type	Description
WDT	15:0	r	<b>Watchdog Timer Value</b> This contains the current count value of the Watchdog Timer.

Note:

## 6.7.5 System Control Block

### 6.7.5.1 Register Write Protection

The System Control Unit CB-Core provides two different protection types of registers:

1. Unprotected Registers
2. Protected Registers.

The unprotected registers allow reading and writing (if not read-only) of register values without any restrictions. However, the write access of the protected registers can be programmed for three different security level modes (the read access is always unprotected):

1. Write Protected Mode  
The registers can not be accessed by a write command.
2. Low Protected Mode  
The registers can be written with a special command sequence (see description below).
3. Unprotected Mode.  
All write accesses are possible.

Some register controlled functions and modes that are critical for the controller operation are locked after the execution of EINIT, so that these vital system functions cannot be changed inadvertently, for example, by software errors. However, as these security

CONFIDENTIAL

SCU

registers also control the power management, they need to be accessed during operation to select the appropriate mode.

The switching between the different security levels is controlled by a state machine. Via a password and a command sequence the security levels can be changed. After reset the unprotected mode is automatically selected. The EINIT command switches the security level automatically to protected mode.

The low protected mode is important for an application standby state. This mode allows fast accesses within two commands to the protected registers without completely removing the protection.

### Security Level Switching

Two registers are provided for switching the security level, the security level command register **SCUSLC** and the security level status register SCUSLS. **SCUSLC** is used to control the state machine for switching the security level. **SCUSLC** is loaded with the different commands of the command sequence necessary to control a change of the security level. It is also used for the unlock command, which is necessary in the low protected mode to access one protected register. The commands of the unlock command sequence are characterized by certain pattern of words (such as AAAA<sub>H</sub>) or by patterns combined with an 8-bit password.

The password is defined with command 3 and stored in **SCUSLS.PASSWORD**.

*Note: The control of register write protection for the protected registers with security levels is **not** compatible with the release command sequence of the C16x specification. The new architecture has been selected for CB-Core because of better adjustment to the requirements of a modularized system design with platform modules and with the C166S VHDL core.*

#### 6.7.5.1.1 Security Level Command Register

##### SCUSLC

##### Security Level Command Register

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMMAND															

The command definitions are described in [Figure 6-22](#), the state diagram and [Table 6-49](#).

**CONFIDENTIAL**

**SCU**

### 6.7.5.1.2 Security Level Status Register

The security level status register **SCUSLS** is a read-only register, which shows the current password, current security level, and state of the switching state machine.

#### SCUSLS

#### Security Level Status Register

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>STATE</b>				<b>SL</b>		<b>RESERVED</b>			<b>PASSWORD</b>						

Field	Bit	Type	Description
<b>PASSWORD</b>	7:0	r	<b>Current Password</b>
<b>SL</b>	12:11	r	<b>Security Level</b> 00 Unprotected write mode 01 Low protected mode 10 Reserved 11 Write protected mode
<b>STATE</b>	15:13	r	<b>Current State Machine Position</b> 000 Wait for first command (command 0) 001 Wait for command 1 010 Wait for command 2 011 Wait for new security level and for new password (command 3) 100 Security registers are unlocked; access to one register is possible (only in low protected mode) 101 Reserved 110 Reserved 111 Reserved.
<b>RESERVED</b>	6	r	Reserved; these bits must be left at their reset values.

The following registers are defined as protected (security) registers:

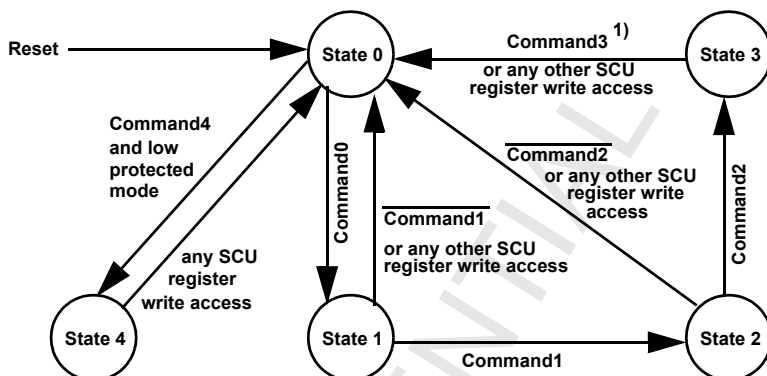
- **SYSCON1**
- **RSTCON**

*Note: A register on product level can also be protected with this mechanism. The signal supervisor\_mode\_o indicates that a write access is allowed either in state 0 or state 4.*

Figure 6-22 shows the state machine for:

- Security level switching
- Unlock command execution in the low protected mode.

**Figure 6-22 State Machine for Security Level Switching**



<sup>1)</sup> Only if the security level command register is accessed, the new security level and the new password is valid.

**Table 6-49 State Machine Commands**

Command Number	Command
0	AAAA <sub>H</sub>
1	5554 <sub>H</sub>
2	96 <sub>H</sub> + inverse (old) password
3	000 <sub>B</sub> + new level + 000 <sub>B</sub> + new password
4	8E <sub>H</sub> + inverse (new) password: Unlocks security register in low protected mode

### Write Access in Low Protected Mode

The write access in the low protected mode is also done via the command sequence:

1. Command 4 with the current password is written to register **SCUSLC**.
2. All security registers are unlocked until the next write access to any Einit protected register.

Read access is always possible for all SCU registers and does not influence the command sequences. In the register **SCUSLS** the current status of the command state machine can always be read.

*Note: It is recommended to use an atomic sequence for all command sequences.*

### 6.7.5.2 Fast External Interrupt and Interrupt Source Control

Up to 8 fast external interrupts can be selected and their trigger transitions be controlled in the SCU with the External Interrupt Control register **EXICON**, refer to [Section 6.3.3.10 Fast Interrupts \(on Page 142\)](#).

A full description of this function can be found in the C166S manual.

The fast interrupts can be used to wake the controller from sleep mode, refer to [Section 6.7.6 Extended Power Management Module \(on Page 259\)](#).

CONFIDENTIAL

SCU

### 6.7.5.2.1 External Interrupt Control Register

EXICON

External Interrupt Control Register

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXI7ES	EXI6ES	EXI5ES	EXI4ES	EXI3ES	EXI2ES	EXI1ES	EXI0ES								

Field	Bits	Type	Description
EXI0ES EXI1ES EXI2ES EXI3ES	7:0	rw	<b>External Interrupts 0, 1, 2, 3 Edge Selection</b> 00 Fast external interrupts disabled: standard mode 01 Interrupt on positive edge (rising) 10 Reserved 11 Reserved
EXI4ES EXI5ES	11:8	rw	<b>External Interrupts 4, 5 Edge Selection</b> 00 Fast external interrupts disabled: standard mode 01 Interrupt on positive edge (rising) 10 Interrupt on negative edge (falling) 11 Interrupt on any edge (rising or falling)
EXI6ES EXI7ES	15:12	rw	<b>External Interrupts 6, 7 Edge Selection</b> Refer to <b>EXI0ES</b> description.

The fast external interrupts are sampled in the SCU every 2 TCL. The interrupt request arbitration and processing, however, is executed in the core every 8 TCL.

*Note: In the Sleep mode, no clock is available; therefore sampling **must** be performed with asynchronous structures.*

*Note: In the Sleep mode fast external interrupt inputs as well as the NMI input are controlled for spike suppression in the System Control Block. Input signals shorter than 10ns are suppressed and lowest level detection is guaranteed for 150ns input signals.*



**CONFIDENTIAL**

**SCU**

### 6.7.5.2.2 External Interrupt Selection Register

Fast external interrupts may also have interrupt sources selected from other peripherals. This function is very advantageous in the Slow Down mode or in the Sleep mode. The register **EXISEL** is used to switch the receive inputs of the serial interfaces in SCU to the fast external interrupts to detect incoming messages in case of disabled serial interface modules.

#### **EXISEL**

#### **External Interrupt Selection Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>EXI7SS</b>	<b>EXI6SS</b>	<b>EXI5SS</b>	<b>EXI4SS</b>	<b>EXI3SS</b>	<b>EXI2SS</b>	<b>EXI1SS</b>	<b>EXI0SS</b>								

**Table 6-50 Alternate Sources for External Interrupts**

<b>EXIxSS</b>	<b>Alternate Source (input fex_int_b)</b>
<b>EXI0SS</b>	RxD0 (ASC0)
<b>EXI1SS</b>	sim_inout (SIM card interface)
<b>EXI2SS</b>	keypad_int (keypad interface)
<b>EXI3SS</b>	rtc_int (RTC block)
<b>EXI4SS</b>	Pin EX4BIN
<b>EXI5SS</b>	Pin EX5BIN
<b>EXI6SS</b>	INT_GP(2) of GSM timer
<b>EXI7SS</b>	GPT1 timer 3 interrupt

<b>Field</b>	<b>Bits</b>	<b>Type</b>	<b>Description</b>
<b>EXIxSS</b> (x = 0 to 7)	15:0	rw	<b>External Interrupt x Source Selection</b> 00 Input from default pin (fex_int_a) 01 Input from "alternate source" (fex_int_b) 10 Input from default pin ORed with "alternate source" 11 Input from default pin ANDed with "alternate source"

*Note: The SCU uses per fast external interrupt two control signals for source selection control.*

*Note: The internally processed fast external interrupt requests from SCU are routed to the interrupt controller of the core for execution.*

### **6.7.5.3 Clock Output Frequency Control**

A clock output signal with programmable frequency ( $f_{OUT}$ ) can be output via pin fout. This clock signal is generated via a reload counter, so the output frequency can be selected in small steps. An optional toggle latch provides a clock signal with a 50% duty cycle.

The fout signal is available via an alternate pin function. It is used for pin timing testing/verification.

Signal fout always provides complete output periods (see [Figure 6-24 \(on page 256\)](#)):

- When fout is started (FOEN--> 1) FOCNT is loaded from FORV
- When fout should be stopped (FOEN--> 0) FOCNT is stopped when fout has reached (or is) 0.

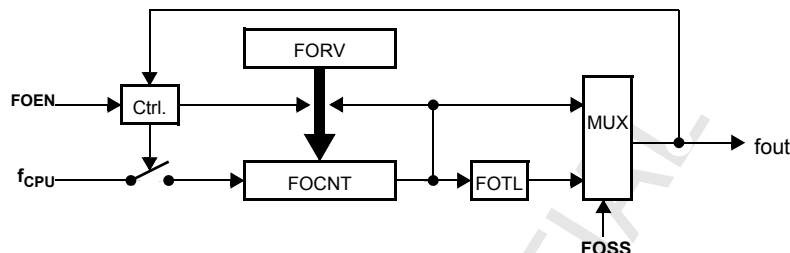
CONFIDENTIAL

SCU

### 6.7.5.3.1 Frequency Output Control Register

Register **FOCON** provides control over the output signal generation, refer to [Figure 6-23](#).

**Figure 6-23 Clock Output Signal Generation**



#### FOCON

#### Frequency Output Control Register

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FOEN	FOS	FORV						RESERVED	FOTL	FOCNT					

Field	Bits	Type	Description
<b>FOCNT</b>	5:0	rw	<b>Frequency Output Counter</b>
<b>FOTL</b>	6	rw	<b>Frequency Output Toggle Latch</b> Toggled on each underflow of <b>FOCNT</b> .
<b>FORV</b>	13:8	rw	<b>Frequency Output Reload Value</b> Copied to <b>FOCNT</b> on each underflow of <b>FOCNT</b> .
<b>FOSS</b>	14	rw	<b>Frequency Output Signal Select</b> 0 Output of the toggle latch: DC = 50%. 1 Output of the reload counter: DC depends on <b>FORV</b> .
<b>FOEN</b>	15	rw	<b>Frequency Output Enable</b> 0 Frequency output generation stops when signal <b>fout</b> is/goes low. 1 <b>FOCNT</b> is running, <b>fout</b> is gated to pin. 1st reload after 0-1 transition.
<b>RESERVED</b>	7	r	Reserved; these bits must be left at their reset values.

CONFIDENTIAL

SCU

## Output Frequency Calculation

The output frequency can be calculated as:

$$f_{OUT} = f_{CPU} / ((FORV+1) * 2^{(1-FOSS)}) \quad [0.3]$$

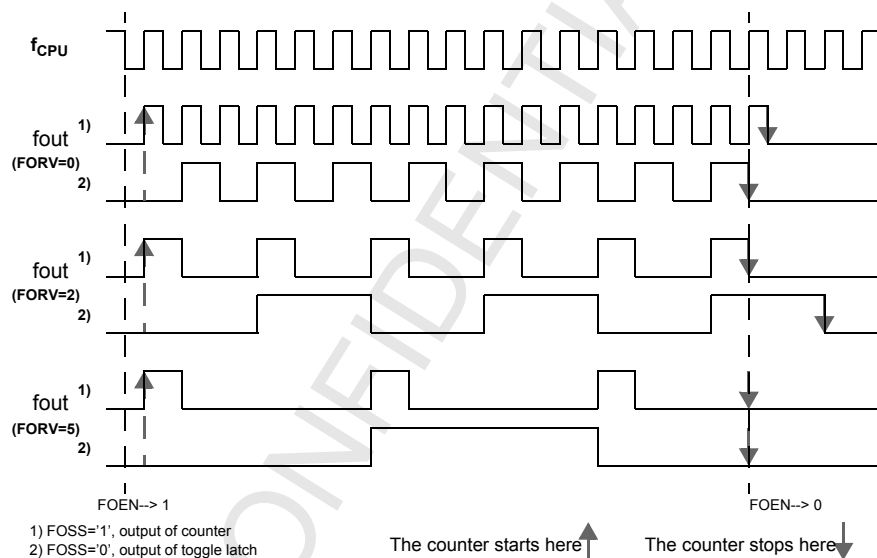
so

$$f_{OUTmin} = f_{CPU} / 128 \quad (FORV = 3F_H, FOSS = 0) \quad [0.4]$$

and

$$f_{OUTmax} = f_{CPU} / 1 \quad (FORV = 00_H, FOSS = 1). \quad [0.5]$$

**Figure 6-24 Signal Waveforms**



**Note:** The output signal for FOSS = 1 is high for the duration of 1  $f_{CPU}$  cycle for all reload values  $FORV > 0$ . For  $FORV = 0$  the output signal corresponds to  $f_{CPU}$ .

## Connection to Output

Signal f<sub>out</sub> is connected to pad CLKOUT. As described in [Figure 6-25](#), when FOEN is:

- 0 PAD CLKOUT is driven by signal clkout
- 1 PAD CLKOUT is driven by signal f<sub>out</sub>.

Figure 6-25 Connection to Port Logic (Functional Approach)

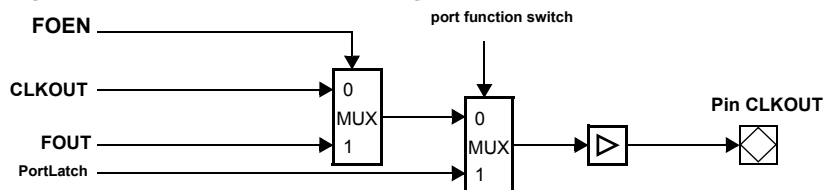


Table 6-51 Clock Output Frequencies Control Register

$f_{CPU}$	$f_{OUT}$ in kHz for FORV = xx, FOSS = 1/0					FORV for $f_{OUT} = 1$ MHz	
	xx= 00 <sub>H</sub>	01 <sub>H</sub>	02 <sub>H</sub>	3E <sub>H</sub> (62 <sub>D</sub> )	3F <sub>H</sub> (64 <sub>D</sub> )	FOSS = 0	FOSS = 1
12 MHz	12000 6000	6000 3000	4000 2000	190.476 95.238	187.5 93.75	05 <sub>H</sub>	0B <sub>H</sub>
16 MHz	16000 8000	8000 4000	5333.33 2666.667	253.968 126.984	250 125	07 <sub>H</sub>	0F <sub>H</sub>
20 MHz	20000 10000	10000 5000	6666.667 3333.33	317.46 158.73	312.5 156.25	09 <sub>H</sub>	13 <sub>H</sub>
25 MHz	25000 12500	12500 6250	8333.33 4166.667	396.825 198.4126	390.625 195.3125	0C <sub>H</sub> (1.04167)	17 <sub>H</sub>
40 MHz	40000 20000	20000 10000	13333,33 6666,66	645,16 322,58	625 312,5	19 <sub>D</sub>	39 <sub>D</sub>
80 MHz	80000 40000	40000 20000	26666,66 13333,33	1290,32 645,16	1250 625	39 <sub>D</sub>	----

#### 6.7.5.4 Frequency Control and Operation Control on Extbus/X-Bus

There are two dedicated clocking domains inside the C166S: pd\_clk and cpu\_clk. Both clocks are derived from the clk\_master, which is generated inside the CGU. There is a clock divider with a programmable dividing factor for pd\_clk (PDCLKDIV) and cpu\_clk (CPUPRE). For the CPUPRE description, refer to [Section 10.4.1 Clock Generation Unit \(on Page 653\)](#).

CONFIDENTIAL

SCU

### 6.7.5.4.1 System Control Register 1

**SYSCON1**

**System Control Register 1**

**Reset value: 0100<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
RESERVED					PDCLKDIV			RESERVED					ALE DIS	RES ERV ED	SLEEP CON	

Field	Bits	Type	Description
<b>SLEEPCON</b>	1:0	rw	<b>SLEEP Mode Configuration</b> (refer to <a href="#">Section 6.7.6.2</a> ) 00 IDLE mode 01 SLEEP mode 10 Reserved 11 Reserved
<b>ALEDIS</b>	3	rw	<b>External ALE Signal Selection</b> 0 Enables generation of ALE in bus controller 1 Disables generation of ALE in bus controller
<b>PDCLKDIV</b>	10:8	rw	<b>Bus Clock Configuration for PD Bus and CLKOUT</b> 000 Normal mode; pd_clk and CLKOUT (if enabled) identical to clk_master 001 Dividing factor is 2; pd_clk and CLKOUT half as fast as clk_master (default value and it is changed by Boot Code) 010 Dividing factor for pd_clk is 4 011 Dividing factor for pd_clk is 8 100 Dividing factor for pd_clk is 16 11x Reserved
<b>RESERVED</b>	2, 7:4, 15:11	r	Reserved; these bits must be left at their reset values.

*Note: **SYSCON1** is a security register. The security level is automatically set to write protection after execution of EINIT.*

The synchronous timings with setup and hold times related to the bus clock as defined in the C166S spec for the External Bus and for the X-Bus are independently of the bus clock frequency and remain unchanged if a dividing factor is enabled. Only the cycle time and the length of wait states is delayed to support an optimized adaption to the systems configuration and bus load.

CONFIDENTIAL

SCU

### 6.7.6 Extended Power Management Module

The Extended Power Management Module controls the Idle/Sleep mode. Power Down is not supported because similar functions are available in the SCCU block.

*Note: The known basic power reduction mode (Idle) is controlled as described in every C16x manual, but with one exception: the external bus and the X-Bus are released for other bus masters during Idle state (refer to [Section 6.7.6.2 Description of the Sleep Mode \(on Page 259\)](#)).*

#### 6.7.6.1 Idle Mode

Before entering the MCU idle mode by executing IDLE command, the external bus must be released to save power consumption (VEBU). To release it, set [EBU\\_PDC \(on Page 1201\)](#).PDW.

*Note: The IDLE command must be executed internally.*

After MCU leaves the idle mode, to retrieve the external bus, reset PDW.

#### 6.7.6.2 Description of the Sleep Mode

The Sleep mode is a new power management function which represents and is equal to a Power Down mode but with exit/wakeup handling as in the Idle mode. Wakeup from Sleep state is possible with all external interrupts (including alternate sources), with nmi\_n and with the RTC interrupts rtc\_int and t14\_int. As in Idle mode PEC requests are executed in Sleep mode resulting in an interruption and resumption of Sleep mode. The watchdog timer is stopped in Sleep mode. The contents of internal RAM and registers are preserved through the voltage supplied via the VDD pins.

Generally, the external bus and the X-Bus are released during Sleep mode if enabled by the last Hold Enable bit [PSW.S1](#). If enabled, the signal hlda is active as long as the Sleep mode (or the Idle mode) is active. Only when the clock is available again after wakeup, the HOLD request signal is sampled and the hlda state is continued until HOLD is deactivated.

As in Power Down mode, the Sleep mode may also be combined with a running real time clock RTC.

The Sleep mode is entered after the standard IDLE instruction (a protected 32 bit instruction) has been executed and the instruction before the IDLE instruction has been completed. Selection between standard Idle mode and Sleep mode is controlled with the register [SYSCON1](#).

#### 6.7.6.3 Sleep Mode Control

The Sleep mode is controlled by the bitfield [SYSCON1.SLEEPCON](#).

CONFIDENTIAL

SCU

Before entering Sleep mode with the IDLE instruction, the continuation of instruction processing **after** termination of Sleep mode must be prepared as in the standard Idle mode. For wakeup using a interrupt, four general possibilities of continuation can be selected, which are controlled (prepared) as follows:

- **Continuation with first instruction after the IDLE instruction** is enabled if either:
  - Interrupts are globally disabled with the Interrupt Enable bit in **PSW**
  - The interrupt is enabled by global (**PSW**) and by individual (interrupt control register) enable bit, but the current MCU priority level (in **PSW**) of IDLE instruction is higher than the interrupt level.
- **Continuation with the first instruction of dedicated interrupt service routine** will be selected if the interrupt is enabled by global (in **PSW**) and by individual (interrupt control register) enable bit, and the MCU priority level of IDLE instruction is lower than the interrupt level, thus the enabled interrupt has highest priority. Additionally, PEC Transfer for this interrupt is not enabled. The continuation with the dedicated service routine is always performed in case of NMI hardware traps, independently of any enable bit or MCU priority level.
- **Execution of one PEC Transfer and resumption of Sleep mode** will be selected if the interrupt is enabled by global (in **PSW**) and by individual (interrupt control register) enable bit, and the MCU priority level of IDLE instruction is lower than the PEC level, thus the enabled interrupt has highest priority. Additionally, PEC Transfer for this interrupt must be enabled.



#### 6.7.6.4 Wakeup Triggers for Idle Mode

Wakeup from Idle mode is performed with any enabled interrupt request. Idle mode is terminated and the before selected (and above described) continuation of processing is executed, if one of the following interrupts occur:

- Any enabled Interrupt<sup>1)</sup> processed by the Interrupt Controller
- Any JTAG request
- RTC interrupt(s)
- Non-Maskable Interrupt NMI (A high-to-low transition on nmi\_n pin always terminates the Sleep mode).

#### 6.7.6.5 Wakeup Triggers for Sleep Mode

As wakeup from Idle mode, wakeup from the Sleep mode is performed with any enabled interrupt request. The Sleep mode is terminated and the selected (and above described) continuation of processing is executed (refer to [Sleep Mode Control \(on Page 259\)](#)) if one of the following interrupts occur:

- **Fast External Interrupts (EXxINT) (default or alternate sources as selected by EXISEL).** All fast external interrupts can be selected for wakeup from Sleep mode by defining the related trigger transitions (edges) in the [EXICON](#) register. All transition types are allowed also in the Sleep mode.
- **RTC interrupt(s).** These are indicated by high level on rtc\_int signal. For wakeup, RTC operation during the Sleep mode must be selected in [SYSCON1](#).
- **Non-Maskable Interrupt NMI.** A high-to-low transition on nmi\_n pin always terminates the Sleep mode.

*Note: All the signals described above have to be processed without any clock in the sleep mode.*

#### 6.7.6.6 Setup Lengthening Control (Start Delay)

Contrary to Idle mode, after wakeup from Sleep mode at first the ramp-up of clock system (oscillator and PLL) has to be controlled before any MCU operation can be started. Only when the clock system is locked on configured frequency, the following processing is identical to wakeup from Idle mode. This includes delay of first instruction execution in case of voltage ramp-up of program Flash module.

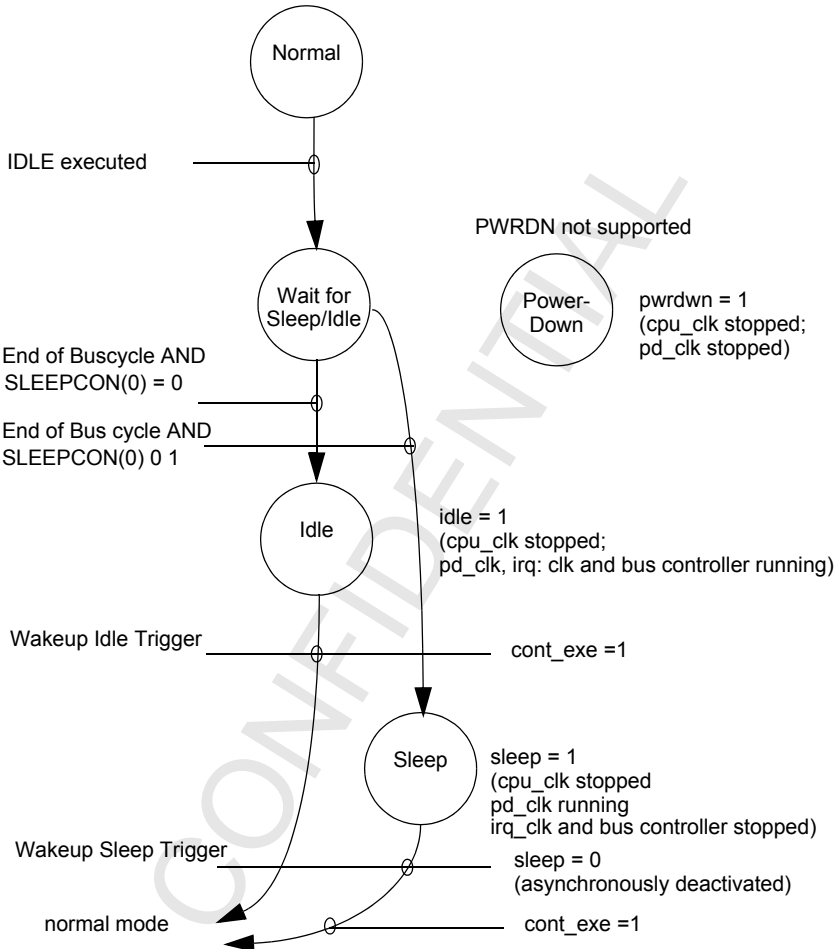
*Note: This will be controlled by the CGU and is not visible to the C166S: the clocks pd\_clk and cpu\_clk become active only with stable clocking system (clk\_lock = 1).*

*Note: [SYSCON1](#). 8 => syscon1\_sleepcon*

<sup>1)</sup> The fast external interrupts are processed by the interrupt controller in the Idle mode.

### 6.7.6.6.1 State Diagram

Figure 6-26 Simplified Idle/Sleep/Power Down State Diagram



### 6.7.6.7 Wakeup Trigger for Idle or Sleep Mode with Standby Clock Mode

1. To prepare for the C166S Idle or Sleep mode and for the Standby Clock Mode (refer to:

**Section 6.7.6.4 Wakeup Triggers for Idle Mode** or  
**Section 6.7.6.5 Wakeup Triggers for Sleep Mode**) and  
**Section 10.4.2 Standby Clock Control Unit (SCCU) (on Page 685))**:

- Enable Wakeup from an external pin: **SCCUHWWAKEUPL.EXT\_EN** = 1
- Select interrupt of positive edge: **EXICON.EXIOES** = 01<sub>B</sub>
- Select the Alternate Source: **EXISEL.EXIOSS** = 01<sub>B</sub>
- Select either:
  - The Idle Mode: **SYSCON1.SLEEPCON** = 00<sub>B</sub>.
  - The Sleep Mode: **SYSCON1.SLEEPCON** = 01<sub>B</sub>

2. Configure the Standby Clock Mode by starting the SCCU block state machine by setting the following bits:

- **SCCUSCTRL.UCSLP** = 1
- **SCCUSLCTRL.SLPEN** = 1.

3. Verify that: **SCCUSMSTA.S3** = 1, then call the IDLE instruction to enter either the Idle Mode or the Sleep Mode (depending on **SYSCON1.SLEEPCON** setting).

At this point the SCCU state machine is in the **tcxo\_off (S3)** state, which means that:

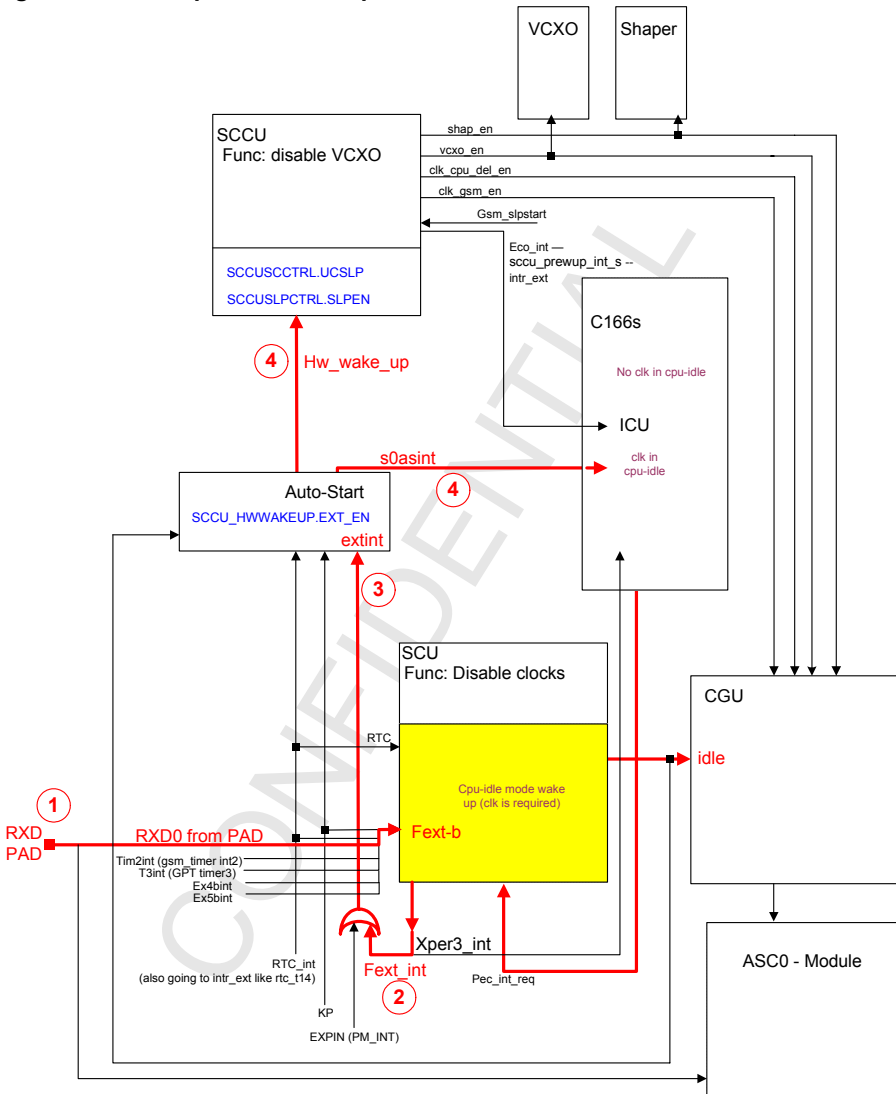
- The Stand-by clock (32 kHz) is used.
- The VCXO is stopped.

Also, no SW fetching is possible by the C166S.

There are several methods to wake-up, the following scenario uses the RDX PAD (see **Figure 6-27**):

1. The Wakeup sequence is triggered by detection of a Low to High transition on RxD0.
2. Fext\_int is triggered.
3. After the OR, extint is generated, which wakes up the Auto-Start block.
4. From the Auto-Start two things happen at the same time:
  - d) **S0asint** generated. This wakes up the C166S and SW fetching is authorized again.
  - a) **Hw\_wake\_up** generated (because **SCCUHWWAKEUPL.EXT\_EN** = 1) and the SCCU state machine executes state transitions (refer to **Figure 10-14 SCCU Main State Machine**): S3 (tcxo\_off) -> S4 (tcxo\_on) -> S5 (shaper\_on) -> S1 (uc\_on).

Figure 6-27 Sleep + Idle Wakeup via RxD0



## 6.7.7 Peripheral Management Module

Flexible peripheral management is performed in the Clock Management Module. An introduction is presented in the overview section.

This module especially serves for power management support, controlling dynamically the operation and thus the power consumption of the different peripherals on PD Bus and X-Bus. In each situation (for example, several system operating modes, standby, etc.) only those peripherals may be kept running which are required for the respective functionality. All others can be switched off. It also allows the operation control of whole groups of peripherals.

Peripheral's operation is disabled or enabled by controlling the specific clock input. This function also is supported in idle and/or slow down mode.

While a peripheral is disabled its output pins remain in the state they had at the time of disabling.

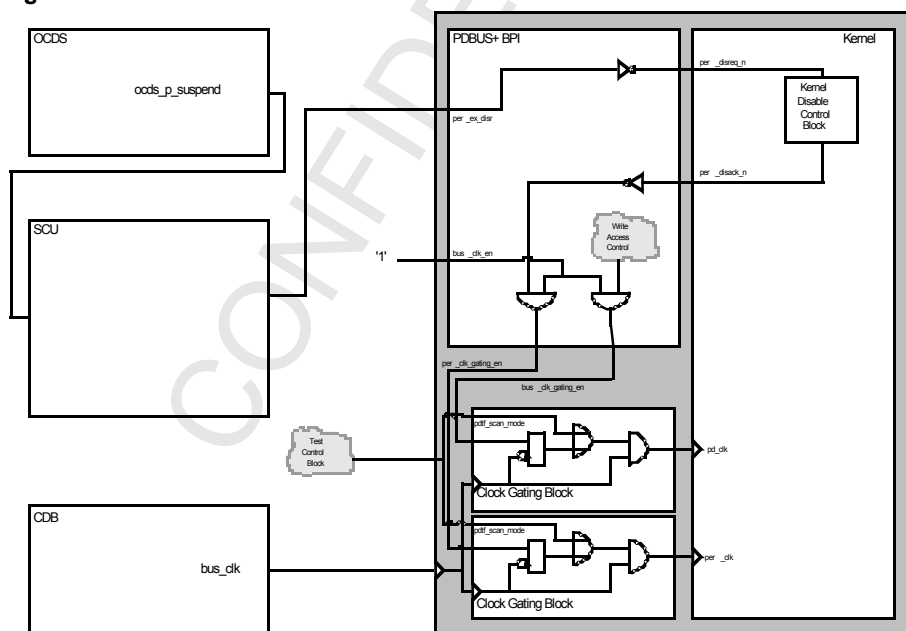
In contrast to the peripheral management of the 16x family the registers of a **disabled** module are not accessible. Only the clock control register of the platform peripheral is accessible.

The Clock Generation Unit can not be disabled.

*Note: The RTC is enabled or disabled via **RTC CTRL (on Page 1124)** register.*

The output signals per ex disc are derived as indicated in **Figure 6-28**.

**Figure 6-28 Clock Control on PD Bus**



CONFIDENTIAL

SCU

## 6.7.8 Identification Register Block

The derivatives of 16-bit microcontrollers with VHDL CoreCB-Core provide a set of 8 identification registers that offer information on the chip, as manufacturer, chip type and its memory (EEPROM, OTP, DRAM, or Flash memory) properties, and information on the controller-core (type of module, redesign state).

The ID registers are read only registers.

*Note: The **SCUID** register does not identify the whole chip, it identifies the module itself and its redesign state.*

The ID registers are placed in the extended SFR area.

### 6.7.8.1 Identification of Manufacturer

IDMANUF

Identification of Manufacturer

Reset value: 1823<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MANUF												DEPT			

Field	Bits	Type	Description
DEPT	3:0	r	<b>Department</b> Indicates the department within Infineon. 3 <sub>H</sub> : WS BB CR
MANUF	15:4	r	<b>Manufacturer</b> This is the JEDEC normalized manufacturer code. 182 <sub>H</sub> : Infineon

CONFIDENTIAL

SCU

## 6.7.8.2 Identification of Device Register

IDCHIP

Identification of Device Register

Reset value: 0700<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHIPID								REVISION							

Field	Bits	Type	Description
REVISION	7:0	r	<b>Device Revision Code</b> Identifies the device step.
CHIPID	15:8	r	<b>Device Identification</b> Identifies the device name. 7 <sub>H</sub> is used for the E-GOLDradio.

CONFIDENTIAL

SCU

### 6.7.8.3 Identification of Memory Register 1

#### IDMEM

##### Identification of Memory Register 1

Reset value: 1000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type				Size											

Field	Bits	Type	Description
Size	11:0	r	<b>Size of the On-Chip Program Memory</b> The size of the implemented program memory in terms of 4k blocks, for example, Memory-size = <Size>*4kByte.
Type	15:12	r	<b>Type of On-Chip Program Memory</b> Identifies the memory type on this silicon: 000 ROMless 001 Mask ROM 010 EPROM 011 Flash 100 OTP 101 EEPROM 110 DRAM/SRAM Other Reserved

### 6.7.8.4 Identification of Programming Voltages

*The E-GOLDradio does not have programmable on-chip memory.*

#### IDPROG

##### Identification of Programming Voltages

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															

Field	Bits	Type	Description
RESERVED	15:0	r	Reserved; these bits must be left at their reset values.



CONFIDENTIAL

SCU

### 6.7.8.5 Identification of Memory Register 2

**IDMEM2** describes the second block of (program) memory. For example, a device may contain Flash and EEPROM or DRAM sections. Static RAM modules are not described with ID registers.

If the second memory block in a device is not existent, **IDMEM2** reads all zeros.

#### IDMEM2

##### Identification of Memory Register 2

Reset value: 6008<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type				Size											

Field	Bits	Type	Description
<b>Size</b>	11:0	r	<b>Size of the On-Chip Program Memory</b> The size of the implemented program memory in terms of 4k blocks, for example, Memory-size = <Size>*4kByte. <i>Note: If the size is equal to 0 with an type not equal to 0, this indicates the size is kept confidential.</i>
<b>Type</b>	15:12	r	<b>Type of On-Chip Program Memory</b> Identifies the memory type on this silicon: 000 ROMless 001 Mask ROM 010 EPROM 011 Flash 100 OTP 101 EEPROM 110 DRAM/SRAM Other Reserved

## CPUID

### MCU Identification Register

Reset value: 0403<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ModuleNum								RevisionNum							

Field	Bits	Type	Description
RevisionNum	7:0	r	<b>Module Number of MCU</b> The current revision number is 03 <sub>H</sub> .
ModuleNum	15:8	r	<b>Module Number of MCU</b> The current module number is 04 <sub>H</sub> .

### 6.7.8.6 Redesign Tracing Register

#### IDRT

### Redesign Tracing Register

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															

Field	Bits	Type	Description
RESERVED	15:1	r	Reserved; these bits must be left at their reset values.

CONFIDENTIAL

SCU

### 6.7.8.7 SCU Identification Register

SCUID

SCU Identification Register

Reset value: 6101<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCUModuleNum								SCURevisionNum							

Field	Bits	Type	Description
SCURevisionNum	17:0	r	<b>SCU Revision Number</b> The current revision number is 01 <sub>H</sub> .
SCUModuleNum	15:8	r	<b>SCU Module Number</b> The current module number is 61 <sub>H</sub> .

*Note: The above described identification registers identify the whole chip and the SCU.  
For identification of the VHDL Core, a separate MCU identification register **CPUID** is provided in the core.*

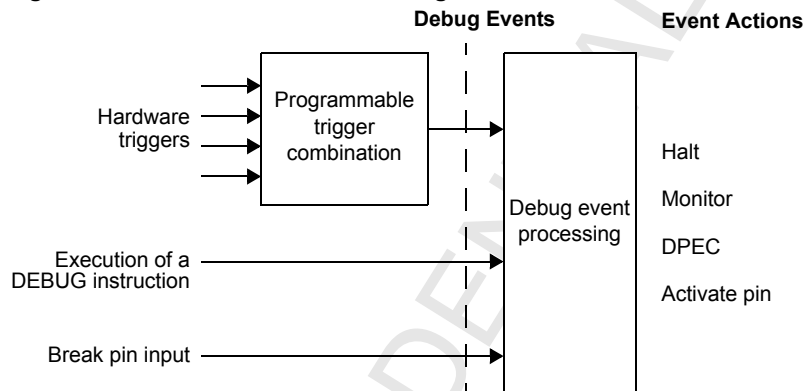
## 6.8 OCDS

### 6.8.1 Introduction

#### Basic Concept

The debug concept includes both the generation of debug events and the definition of event actions taken when a debug event is generated.

**Figure 6-29 OCDS Module Block Diagram**



#### Debug Events

- Hardware trigger combination
- Execution of a DEBUG instruction
- Break pin input.

#### Debug Event Actions

- Halt the MCU
- Call a monitor
- Trigger a data transfer (DPEC) executed by Cerberus
- Activate external pin.

CONFIDENTIAL

OCDS

## Register Overview

Table 6-52 OCDS Register Overview

Register	Description
<b>DBGSR</b>	Debug status register
<b>DEXEVT</b>	Specifies action if external break pin asserted
<b>DSWEVT</b>	Specifies action if a DEBUG instruction is executed
<b>DTREVT</b>	Combination criteria for hardware triggers and resulting action
<b>DCMPDP</b>	Data programming register for the compare registers DCMPx
<b>DCMPSP</b>	Select and programming register for the compare registers DCMPx
<b>DCMPx<sup>1)</sup></b>	Hardware event equal comparison register 0
<b>DCMP1</b>	Hardware event equal comparison register 1
<b>DCMP2</b>	Hardware event equal comparison register 2
<b>DCMPG</b>	Hardware event range comparison register (greater)
<b>DCMPL</b>	Hardware event range comparison register (less)
<b>DIP</b>	Instruction pointer register
<b>DIPX</b>	Instruction pointer register extension
<b>DTIDR</b>	Task ID register

<sup>1)</sup> Accessed with DCMPSP and DCMPDP.

### 6.8.2 Enabling and Disabling the OCDS

By default, the OCDS is disabled to protect the system during normal execution. Events can be generated only when the OCDS is enabled. The OCDS Module has an enable signal that is normally connected to the chip internal JTAG reset. This means that the OCDS is enabled when the JTAG Module is not in reset state. This is always the case when the external debugger uses Cerberus.

*Note: Depending on the system architecture, the enable signal may be controlled by another source.*

The OCDS Module can also be optionally enabled by software. To avoid an unintentional enabling by an incorrect user program, the following conditions must be true:

1. OCDS is disabled.
2. **DTREVT.MUX\_E** = 10<sub>B</sub>.
3. **DTREVT.SELECT\_E** != 00<sub>B</sub> (enables the equal comparators).
4. **DCMPx** comparison matches (independent of **SELECT\_E**).
5. Currently written **DBGSR.DEBUG\_ENABLED** = 1<sub>B</sub>.

Thus, a monitor must do the following:

1. Write  $F0FC_H$  to **DCMPx** (address of **DBGSR**).
2. Write  $2200_H$  to **DTREVT**.
3. Write  $0001_H$  to **DBGSR**.

If the OCDS was enabled by software, it can only be disabled by a reset.

*Note: This feature (OCDS enabling by software) might be disabled depending on the system architecture.*

### 6.8.3 Reset to Halt Mode

The MCU can be forced directly to Halt Mode ([Section 6.8.5.3 Halt Mode \(on Page 278\)](#)) after reset. This is controlled by the **CCONF.RST\_HLT** bit in the JTAG Module. The reset to Halt Mode requires three steps:

1. Set **CCONF.RST\_HLT** before the MCU reset goes inactive.
2. Set **DBGSR.DEBUG\_ENABLED** to Halt Mode after the reset is released.
3. Clear **CCONF.RST\_HLT**.

To remove the MCU from Halt Mode, **DBGSR.DEBUG\_STATE** must be set to User Mode.

*Note: This feature (Reset to Halt Mode) might be disabled or controlled by another mechanism depending on the system architecture, since the JTAG Module is not a part of the MCU macro and can be modified.*

### 6.8.4 Debug Event Sources

#### 6.8.4.1 Hardware Trigger Combinations

[Table 6-53](#) lists the possible hardware trigger sources.

**Table 6-53 Hardware Triggers**

Trigger source	Size	Description
<b>TASKID</b>	16 bits	This is the contents of the <b>DTIDR.TASKID</b> . It is used by advanced real time operating systems to store the Task ID of the active task.
<b>IP</b>	24 bits	Instruction Pointer
<b>R_ADR</b>	24 bits	Data address of reads
<b>W_ADR</b>	24 bits	Data address of writes
<b>DA</b>	16 bits	Data value (reads or writes)

**TASKID** is the contents of the **DTIDR** register. It is used by advanced real time operating systems to store the Task ID of the active task.

CONFIDENTIAL

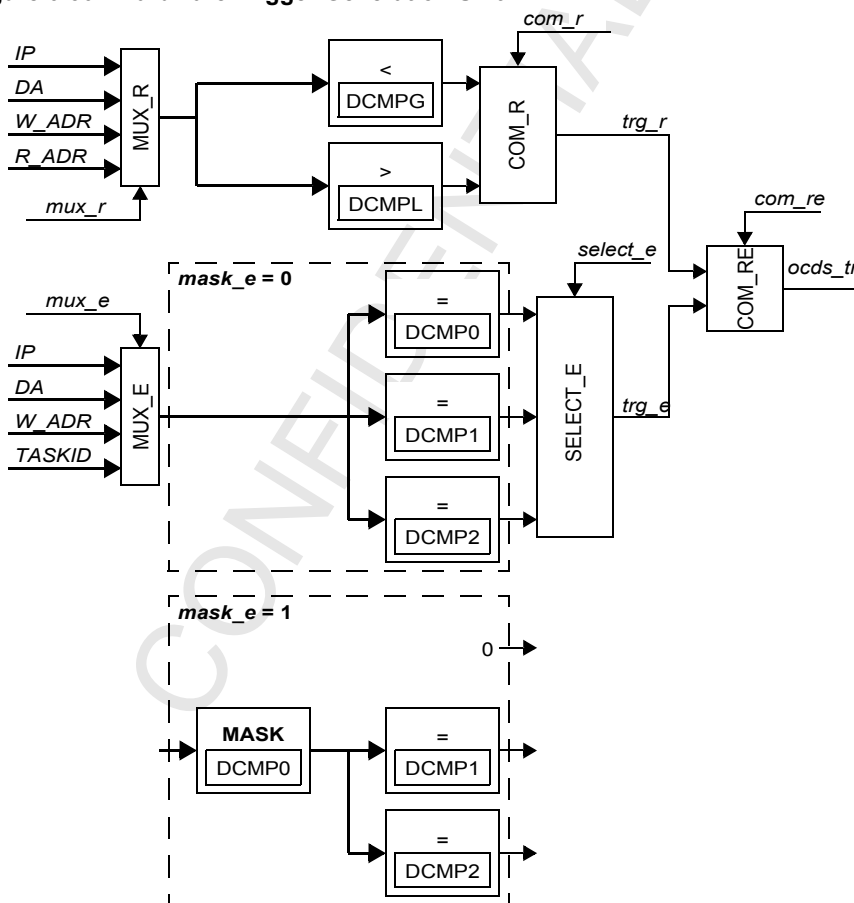
OCDS

The trigger sources are compared and combined in the hardware trigger generation unit (**Figure 6-30**). The hardware trigger generation unit is programmable with the **DTREVT** debug event control register. It consists of two paths:

- The upper path is for one range comparisons
- The lower path for three equal comparisons.

The equal path can be optionally configured for two masked equal comparisons. The configuration options are described in detail in **Chapter 6.8.6.1 Debug Event Control Registers**.

**Figure 6-30 Hardware Trigger Generation Unit**



### 6.8.4.2 Execution of a DEBUG Instruction

There is a mechanism through which software can explicitly generate a debug event. This can be used, for instance, by a debugger to patch code held in RAM to implement breakpoints. A special DEBUG (opcode D140<sub>H</sub>) instruction is defined that is a User Mode instruction and its operation is dependent on whether OCDS is enabled.

If OCDS is enabled, the DEBUG instruction causes a debug event to be raised and the action specified in the **DSWEVT** debug control register is taken. If OCDS is not enabled, the DEBUG instruction is treated as a NOP.

### 6.8.4.3 Break Pin Input

An external debug break pin is provided to allow the debugger to asynchronously interrupt the processor. The action taken when this signal is asserted is specified in the **DSWEVT** debug control registers. This input is triggered on a negative clock edge must be followed by at least two MCU clock cycles with the break signal (TRIG\_IN) is 0.

### 6.8.4.4 Event Prioritizing

It is possible that more than one event may be raised in a single cycle. In this case, the priority of events to be handled is based on the sequence in which the events appear in the event sources list; those listed first are handled before those listed later.

**Table 6-54 Debug Event Priority**

Event	Debug Event Control Register	Priority
Break pin input	<b>DEXEVT</b>	1 (highest)
Execution of a DEBUG instruction	<b>DSWEVT</b>	2
Hardware trigger combination	<b>DTREVT</b>	3



**CONFIDENTIAL**

**OCDS**

### 6.8.5 Debug Event Actions

When the OCDS is enabled and a debug event is generated, one of the actions listed in [Table 6-55](#) is taken. These actions are explained in detail in the following sections.

**Table 6-55 Debug Event Actions**

Debug Event Action	User Resources	Interruptible?	Break before make?
Activate external pin	-	-	-
Trigger data transfer (DPEC)	Cycle stealing for DPEC	-	Only for program address triggers.
Call a monitor	Stack User address space (Interrupt address)	Yes (after entry).	
Halt	-	No	

#### 6.8.5.1 Trigger Data Transfer (DPEC)

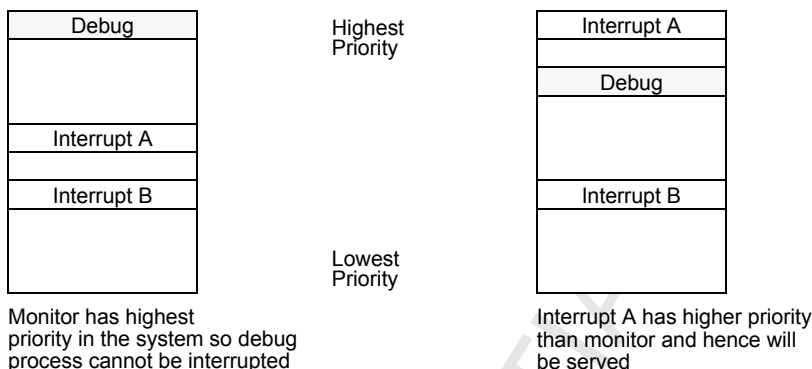
Triggering Cerberus to execute a pending transfer ([Section 6.9.3.3 Triggered Transfers \(DPEC\) \(on Page 307\)](#)) is one of the actions that can be specified to occur when a debug event is raised. This can be used in critical routines where the system cannot be interrupted to transfer a memory location to the **RWDATA** register and read it (trace) through the Cerberus debug port.

#### 6.8.5.2 Call a Monitor

Calling a monitor with a special debug hardware trap (trap number 8, vector location 20<sub>H</sub>) is one of the possible actions to be taken when a debug event is raised. This trap has the highest priority, but the monitor routine can reduce its own priority level by resetting the debug flag bit **TFR.DEBTRAP** and writing the priority to the **PSW.ILVL** field.

This short entry to an interruptible monitor allows a flexible debug environment to be defined that is capable of satisfying many of the requirements for efficient debugging of a real time system. For example, safety critical code can be served while the debugger is active. The monitor is ended with a regular RETI instruction. The debug flag bit **DEBTRAP** has to be cleared on exiting the TRAP routine, otherwise it will be called again.

Figure 6-31 Simple and Advanced Debug Model



### Structure of a Non-Interruptible Monitor Routine

1. Do processing (non interruptible).
2. Set **DBGSR** = 0000<sub>H</sub>.
3. Clear the **TFR.DEBTRAP**.
4. Return to user program with RETI instruction.

### Structure of an Interruptible Monitor Routine

1. Set **DBGSR.DEBUG\_STATE** == 00<sub>B</sub> (User Mode).
2. Clear the **TFR.DEBTRAP**.
3. Reduce the interrupt level ILVL in W.
4. Do Processing.
5. Set **DBGSR** = 0000<sub>H</sub>.
6. Return to user program with RETI instruction.

*Note: The reduction of the interrupt priority of the monitor can cause stack overflows. If the task that causes the debug event has a higher priority than the monitor, the monitor will be pushed onto the stack again and again.*

*Note: Be sure that the monitor does not cause an event itself. Otherwise, it will be started again and again and cause a stack overflow.*

#### 6.8.5.3 Halt Mode

The system suspends execution by halting the instruction flow and will not respond to any interrupts. It then relies on the external debug system to interrogate the target entirely by reading and updating through the Cerberus debug port. The MCU resumes in User Mode when the external debug hardware resets **DBGSR.DEBUG\_STATE** to

CONFIDENTIAL

OCDS

User Mode. It should also reset **DBGSR.OCDS\_P\_SUSPEND** and **DBGSR.EVENT\_SOURCE**.

#### 6.8.5.4 Activate External Pin

An external pin assertion can be specified as a debug event action. This is to be used in critical routines where the system cannot be interrupted to signal to the external world that a particular event has happened. This feature could also be useful to synchronize the internal and external debug hardware or to do profiling. In most cases the break out pin is active 0 for as long as the trigger condition is met.

#### 6.8.5.5 Single Stepping

Single stepping can be done in Halt Mode or with a debug monitor.

##### Single Stepping in Halt Mode

For this behavior, the trigger condition is set to be always true (example: trigger on **IP** range with **DCMPL** = 000000<sub>H</sub>, **DCMPG** = 000001<sub>H</sub>, **COM\_R** = 11<sub>B</sub> and **COM\_RE** = 0<sub>B</sub>) and the **DTREVT.BREAK\_AFTER\_MAKE** bit is set. After every restart, the MCU will be halted again when the next instruction has been executed.

##### Single Stepping with a Debug Monitor

The advantage of this type of single stepping is that the system can serve high priority interrupt requests (**Section 6.8.5.2 Call a Monitor (on Page 277)**). The basic approach is similar to the single stepping in Halt Mode with two differences:

- The event action is set to Call a monitor
- The code of the interrupt service routines and of the debug monitor may not be part of the **IP** address trigger range.

It is recommended to adjust the **IP** address trigger range to the current (C-) function of the user code. This results in a step-over behavior if sub-functions are called within this function. If a step-in behavior is required, an additional single **IP** address trigger can be set to the entry of the sub-function and when it is entered, the **IP** address trigger range is changed to cover the sub-function.

**CONFIDENTIAL**

**OCDS**

## 6.8.6 Debug Registers

**Table 6-56 OCDS Register Summary**

Name	ESFR	Type	Description
<b>DBGSR</b>	F0FC <sub>H</sub>	h	Debug status register
<b>DIPX</b>	F0FA <sub>H</sub>	h	Instruction pointer register extension
<b>DIP</b>	F0F8 <sub>H</sub>	h	Instruction pointer register
<b>DSWEVT</b>	F0F4 <sub>H</sub>		Specifies action if DEBUG instruction is executed
<b>DEXEVT</b>	F0F2 <sub>H</sub>		Specifies action if external break pin is asserted
<b>DTREVT</b>	F0F0 <sub>H</sub>		Specifies hardware triggers and action
<b>DCMPDP</b>	F0EE <sub>H</sub>		Data programming register for DCMPx
<b>DCMPSP</b>	F0EC <sub>H</sub>		Select and programming register for DCMPx
<b>DTIDR</b>	F0D8 <sub>H</sub>	a	Task ID register
<b>DCMPx</b>	_ <sup>1)</sup>		Hardware event equal comparison register 0
<b>DCMP1</b>	_ <sup>1)</sup>		Hardware event equal comparison register 1
<b>DCMP2</b>	_ <sup>1)</sup>		Hardware event equal comparison register 2
<b>DCMPG</b>	_ <sup>1)</sup>		Hardware event range comparison register (greater)
<b>DCMPL</b>	_ <sup>1)</sup>		Hardware event range comparison register (less)

<sup>1)</sup> Accessed with DCMPSP and DCMPDP.

### 6.8.6.1 Debug Event Control Registers

Each possible source of a debug event has an associated register that defines which action should be taken when that debug event is raised. The debug event control registers have the same structure for all currently defined sources.

CONFIDENTIAL

OCDS

### 6.8.6.1.1 Break Pin and Software Debug Event Control Registers

DEXEVT

DSWEVT

Break Pin and Software Debug Event Control Registers

Reset value 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED										ACTI VAT E PIN	RES ERV ED	PERI PHE RAL S STOP	EVENT_ACTION		

Field	Bits	Type	Description
EVENT_ACTION	2:0	rw	Defines action taken on debug event: 000 None 001 Software Debug Mode 010 Halt Debug Mode 011 Reserved 100 Reserved 101 Execute DPEC 110 Reserved 111 Set event source bit in <b>DBGSR</b> only
PERIPHERALS_STOP	3	rw	0 Peripherals are not affected by this event 1 Sensitive peripherals suspend operation if event occurs
ACTIVATE_PIN	5	rw	0 External pin always inactive 1 External pin is active during debug event
RESERVED	4, 15:6	r	Reserved; these bits must be left at their reset values.

**EVENT\_ACTION** specifies what happens when the associated debug event is raised. The event specifier can have one of the indicated values. For Software and Halt Mode, the lower two bits of the **EVENT\_ACTION** set the **DBGSR.DEBUG\_STATE** field.

The **PERIPHERALS\_STOP** bit controls the operation mode of the peripherals when the associated debug event is raised. If this bit is set, the **DBGSR.OCDS\_P\_SUSPEND** bit is set; this causes sensitive peripherals to suspend.

*Note: Presently, **DBGSR.OCDS\_P\_SUSPEND** is set only when the associated **EVENT\_ACTION** is either Halt Mode or Software Debug Mode. This may change in future versions of OCDS.*

CONFIDENTIAL

OCDS

### 6.8.6.1.2 Hardware Trigger Combination Debug Event Control

#### DTREVT

#### Hardware Trigger Combination Debug Event Control

Reset value 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COM_RE	SELECT_E	MASK_E	COM_R	MUX_E	MUX_R	ACTIVATE_PIN	BREAK_AFTER_MAKE	PERIPHERALS_STOP	EVENT_ACTION						

Field	Bits	Type	Description
EVENT_ACTION	2:0	rw	Identical to <a href="#">DEXEVT.EVENT_ACTION</a>
PERIPHERALS_STOP	3	rw	Identical to <a href="#">DEXEVT.PERIPHERALS_STOP</a>
BREAK_AFTER_MAKE	4	rw	0 Break before make ( <a href="#">IP</a> only) 1 Break after make ( <a href="#">IP</a> only)
ACTIVATE_PIN	5	rw	Identical to <a href="#">DEXEVT.ACTIVATE_PIN</a>
MUX_R	7:6	rw	<b>Range Comparison Input Mux</b> ( <a href="#">Figure 6-30</a> ): 00 Instruction pointer ( <a href="#">IP</a> ) 01 Data value (DA) 10 Write address (W_ADR) 11 Read address (R_ADR)
MUX_E	9:8	rw	<b>Equal Comparison Input Mux</b> ( <a href="#">Figure 6-30</a> ) control: 00 Instruction pointer ( <a href="#">IP</a> ) 01 Data value (DA) 10 Write address (W_ADR) 11 Task ID ( <a href="#">TASKID</a> )
COM_R	11:10	rw	<b>Select range comparison</b> ( <a href="#">Table 6-57</a> )
MASK_E	12	rw	0 Unmasked equal comparison ( <a href="#">Figure 6-30</a> ) 1 Masked equal comparison
SELECT_E	14:13	rw	<b>Select equal comparison</b> ( <a href="#">Table 6-58</a> )
COM_RE	15	rw	<b>Equal and range comparison combination:</b> 0 <sup>1)</sup> ocds_trgevt signal is trg_r OR trg_e 1 ocds_trgevt signal is trg_r AND trg_e

- <sup>1)</sup> The first option (OR) is intended to be used for the case `mux_r == mux_e` only and in particular to have four **IP** triggers. In case of different comparison sources this option results in a complex behavior, because the triggers of for instance **IP** and `W_ADR` are created in different pipeline stages of the MCU.

The **COM\_R** field enables the range comparison to be included in the `ocds_trgevt` generation ([Figure 6-30 Hardware Trigger Generation Unit \(on Page 275\)](#)). For in-range comparisons, **DCMPG** is used as the upper boundary and **DCMPL** is the lower boundary. For out-of-range comparisons, it is the other way round. [Table 6-57](#) lists the options:.

**Table 6-57 DTREVT.COM\_R Field**

Value	<i>trg_r</i> signal
00	0 (not enabled)
01	In range: 1 if <b>DCMPG</b> > input > <b>DCMPL</b> , otherwise 0
10	Reserved
11	Out of range: 1 if ( <b>DCMPG</b> > input) or (input > <b>DCMPL</b> ), otherwise 0

The **MASK\_E** bit distinguishes between masked or unmasked input for the equal comparison ([Figure 6-30](#)). In the masked case, **DCMPx** controls the relevant bits for the comparison. All bits of the input signal where the associated **DCMPx** bit is 0 are also set to 0 prior to the comparison. Note that the comparison values in **DCMP1** and **DCMP2** must also be 0 where the **DCMPx** mask is 0. Otherwise, the comparison will not match.

The **SELECT\_E** field enables the equal comparisons to be included in the `ocds_trgevt` generation and selects which is used ([Figure 6-30 Hardware Trigger Generation Unit \(on Page 275\)](#)). Note that for masked comparisons, the **SELECT\_E** field must be set to `10B` or `11B`. [Table 6-58](#) lists the options.

**Table 6-58 SELECT\_E Field**

Value	mask_e	<i>trg_e</i> signal
00	0	0 (not enabled)
01		1 if <b>DCMPx</b> matches, otherwise 0
10		1 if <b>DCMPx</b> or <b>DCMP1</b> match, otherwise 0
11		1 if <b>DCMPx</b> or <b>DCMP1</b> or <b>DCMP2</b> match, otherwise 0
00	1	0 (not enabled)
01		0 (always)
10		1 if <b>DCMP1</b> matches, otherwise 0
11		1 if <b>DCMP1</b> or <b>DCMP2</b> match, otherwise 0

CONFIDENTIAL

OCDS

### 6.8.6.2 Debug Status Register DBGSR

Debug status register contains information about the current status of the OCDS, including:

- If the debug support is enabled
- The source of the last debug event
- The system debug state.

#### DBGSR

#### Debug Status Register

Reset value 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EVENT SOURCE			RESERVED		TRG EVT E CMP 2	TRG EVT E CMP 1	TRG EVT E CMP 0	TRG EVT R CMP	RESERVED		OCDS_P_SUSPEND	DEBUG STATE		RESERVED	DEBUG_ENABLED

Field	Bits	Type	Description
DEBUG_ENABLED	0	rwh	0 OCDS is disabled 1 OCDS is enabled
DEBUG_STATE	3:2	rwh	<b>Current Debug State</b> 00 User Mode 01 Software Debug Mode 10 Halt Debug Mode 11 Reserved
OCDS_P_SUSPEND	4	rwh	0 No effect 1 Sensitive peripherals suspend their operation
TRGEVT_R_CMP	7	rwh	0 Range comparison did not match 1 Comparison matched for the current event
TRGEVT_E_CMP0	8	rwh	0 Equal comparison 0 did not match 1 Comparison matched for the current event
TRGEVT_E_CMP1	9	rwh	0 Equal comparison 1 did not match 1 Comparison matched for the current event
TRGEVT_E_CMP2	10	rwh	0 Equal comparison 2 did not match 1 Comparison matched for the current event



**CONFIDENTIAL**

**OCDS**

Field	Bits	Type	Description
<b>EVENT_SOURCE</b>	15:13	rwh	<b>Source of Last Debug Event</b> 000 No event xx1 External break pin ( <b>DEXEVT</b> ) x1x DEBUG instruction executed ( <b>DSWEVT</b> ) 1xx Hardware trigger combination ( <b>DTREVT</b> )
<b>RESERVED</b>	1, 6:5, 12:11	r	Reserved; these bits must be left at their reset values.

The **OCDS\_P\_SUSPEND** bit controls the peripheral suspend signal. If set to 1, all sensitive peripherals will suspend. This bit is set by a debug event according to the associated **PERIPHERALS\_STOP** bit in the active debug event control register. This bit must be reset by the debugger.

*Note: If a debug monitor is interrupted by a user task with higher priority, the **DEBUG\_STATE** and **OCDS\_P\_SUSPEND** bits and the peripheral suspend signal are not changed.*

The **EVENT\_SOURCE** bits are set independently from the **EVENT\_ACTION** field in the associated debug event control register, with the exception 000 (None). These bits must be reset by the debugger.

*Note: For the equal comparisons, the **TRGEVT\_E\_CMPx** bits are set only when the associated comparison was enabled (**DTREVT.SELECT\_E**). These bits must be reset by the debugger.*

CONFIDENTIAL

OCDS

### 6.8.6.3 Task ID Register

DTIDR

Task ID Register

Reset value 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TASKID															

Field	Bits	Type	Description
TASKID	15:0	rw	Input to the hardware trigger event generation unit ( <a href="#">Section 6.8.4.1 Hardware Trigger Combinations (on Page 274)</a> ). Intended to be used by advanced real time operating systems to store the task ID of the active task.

### 6.8.6.4 Instruction Pointer Registers

These registers are provided to make the instruction pointer (PC) visible when the MCU is in Halt Mode ([Section 6.8.5.3 Halt Mode \(on Page 278\)](#)).

DIP

Instruction Pointer Register

Reset value 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IP															

Field	Bits	Type	Description
IP	15:0	rh	Bits [15:0] of the current instruction pointer in Halt Mode. <i>Note: IP is valid in Halt Mode only.</i>

## DIPX

### Instruction Pointer Register Extension

Reset value 3000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VERSION				RESERVED				IPX							

Field	Bits	Type	Description
IPX	7:0	rh	Bits [23:16] of the current instruction pointer in Halt Mode. Extends <a href="#">IP</a> .
VERSION	15:12	r	Hardcoded as 3 <sub>H</sub>
RESERVED	11:8	r	Reserved; these bits must be left at their reset values.

The [VERSION](#) field is used by debuggers to adapt to the specific version of the OCDS.

### 6.8.6.5 Hardware Trigger Comparison Registers

The [DCMPx](#) registers are used in the hardware trigger event generation unit ([Section 6.8.4.1 Hardware Trigger Combinations \(on Page 274\)](#)) as reference values for the comparisons. They can be programmed with the two SFR registers [DCMPSP](#) and [DCMPDP](#). [DCMPSP.SELECT\\_DCMP](#) selects the comparison register and writes its highest byte. The lower 16 bits can then be written by an access to register [DCMPDP](#).

DCMPx

DCMP0

DCMP1

DCMP2

DCMPG

DCMPL

### Hardware Trigger Comparison Registers

Reset value 0000<sub>H</sub>

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP_VALUE																							

Field	Bits	Type	Description
CMP_VALUE	23:0	rw	Comparison value for the hardware trigger event generation unit. Can be written only indirectly with <a href="#">DCMPSP</a> and <a href="#">DCMPDP</a> .

CONFIDENTIAL

OCDS

### 6.8.6.6 Select and Programming Register for DCMPx

#### DCMPSP

#### Select and Programming Register for DCMPx

Reset value 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				SELECT_DCMP				DCMP_DATA_X							

Field	Bits	Type	Description
DCMP_DATA	7:0	w	Sets bits [23:16] of selected <b>DCMPx</b> register (via <b>SELECT_DCMP</b> ).
SELECT_DCMP	11:8	w	<b>Selection of Comparison Register</b> 0000 Select DCMP0 0001 Select DCMP1 0010 Select DCMP2 0011 Reserved 0100 Select DCMP3 0101 Select DCMP4 OthersReserved
RESERVED	15:12	r	Reserved; these bits must be left at their reset values.

### 6.8.6.7 Data Programming Register for DCMPx

#### DCMPDP

#### Data Programming Register for DCMPx

Reset value 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCMP_DATA															

Field	Bits	Type	Description
DCMP_DATA	15:0	w	Sets bits [15:0] of selected ( <b>SELECT_DCMP</b> ) <b>DCMPx</b> register (via <b>DCMPSP.SELECT_DCMP</b> )

### **6.8.6.8 Common Considerations on Accessing OCDS Registers**

The functions of OCDS are generally controlled by writing to the Debug Status Register (**DBGSR**). To be executed correctly, any debug step needs the respective bitfields to be properly and at a time set. As **DBGSR** is accessed over the PD+ bus, time needed to have new values effective depends on the speed of that bus. This becomes as more important, as the bus-speed becomes lower compared to the core speed, that is, to the speed of executing instructions. Other important thing is the instance of C166S as a pipelined machine, with the different operations (read/write) executed at different pipeline-stages.

The basic potential problem to be kept in mind is: the new **DBGSR** value (the same is true for any SFR) can not be as a rule effective for the instruction immediately following its modification. The delay - in terms of core instructions executed with the old **DBGSR** value - has two parts:

- Fixed part (in most cases one instruction)
- Variable part (depending on the PD-Bus bus speed).

The most critical points for possible conflicts are:

- Setting-up and enabling OCDS

For proper operation, **DBGSR** must be set after the **DTREVT** register already holds the new value programmed.

- Exiting the monitor

All updates to **DBGSR** must be effective before returning to the user program.

Otherwise a possibility exists, that a breakpoint in code will be reached before **DBGSR** holds the proper settings. This can cause a variety of problems, like calling the monitor after executing the breakpoint or immediate stepping over the breakpoint, instead of breaking before the breakpoint.

### 6.8.6.9 General Work Around to Avoid Software Problems with OCDS

The principal solution to avoid problems on accessing OCDS registers is to assure that after an instruction writing to a register, the instruction which uses the new value will be executed only when new settings are really effective.

- Use non-critical instructions
  - After writing to an OCDS control register (that is, **DBGSR**), instructions which execution does not depend on the new settings can follow:
    - $I_n$  :Writing to DBGSR
    - $I_{n+1}$  :Non-critical instruction(s) ;DBGSR still holds the old value
    - .....
    - $I_{n+d}$  :Any instruction ;new DBGSR is already effective
  - The simplest way to assure some time-to-set is to insert enough no-operations before the next critical instruction:
 

```
extr  #1                      ; EFSR area
mov   DTREVT,#02200h          ; select_e=01, mux_e=10 @repeat(10)
nop                                ; dummy-loop of 10 NOPs
@endr
extr  #1                      ; new DTREVT is already effective:
mov   DBGSR,#00001h           ; enable OCDS
```
- The difficulty here is to estimate what is the enough time to have write completed and effective, more at different PD+ bus-speeds programmed. The above example is true for  $f(PD)/f(MCU) = 1/8$ , for lower ratio even more NOPs will be needed.
- Read-after-write operation
 

Immediately after writing to the OCDS register, an (dummy) read operation from the same address can follow:

```
extr  #2                      ;EFSR area
mov   DBGSR,#00803h           ;enable trigger, SW debug mode,
                                ; execute 1 instruction after RETI
mov   R7,DBGSR                ; new DBGSR is already effective !
reti                                ; exit monitor
```

This assures that the new **DBGSR** value is already effective before to continue with the next instruction. More, this is independent of the PD+ bus-speed, because the MCU takes care write operation to be completed, before to continue with the following read from the same location. So, in fact this is the easiest and most reliable decision to assure proper OCDS operation.

### 6.8.7 Reset Behavior

If OCDS is disabled (usually-when JTAG Module is in reset state), OCDS Module and all its registers are reset with every MCU reset; otherwise, it is never reset. This behavior allows a defined reset in the cases when no debugger is connected or the debugger controls the OCDS indirectly with a monitor.

When the debugger controls the OCDS directly, the OCDS registers are not affected by user program or system environment resets.

## 6.9 Cerberus

Cerberus is a versatile and high performance access port using the JTAG pins.

The connection between the Cerberus and the external JTAG port is done via the **JTAG (on Page 1248)**.

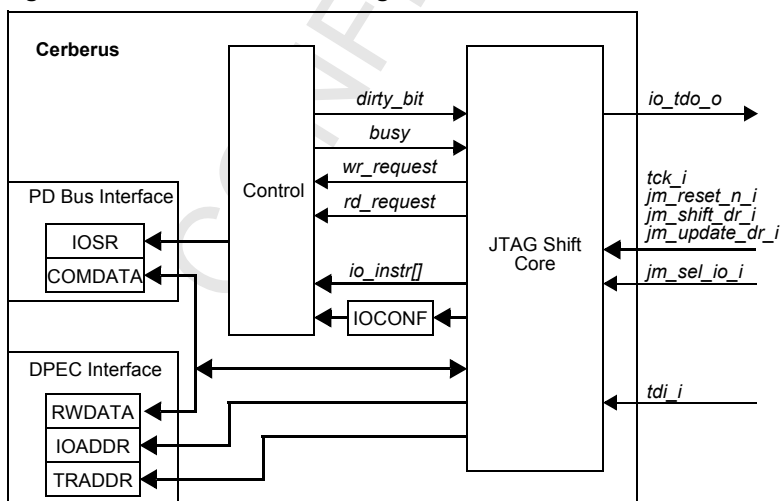
### 6.9.1 Operational Overview

Cerberus is operated by the external debugger across the JTAG Module.

#### Block Diagram

The Cerberus Core contains the JTAG Shift Core as a sub-block, shown in **Figure 6-32**. The JTAG Shift Core is controlled by the JTAG signals and therefore is asynchronous to the other parts of the Cerberus Core.

**Figure 6-32 Cerberus Block Diagram**



### **6.9.1.1 Definitions**

#### **RW Mode and Communication Mode**

Cerberus can be used for two different purposes. The first is to read and write memory locations (I/O Mode) and the second is to exchange data with a program (monitor) running on the MCU (Communication Mode).

#### **Error Protection**

The JTAG Standard does not include any error protection for serial transmission (TDI and TDO pins) and control (TMS pin). However, there are some ways to include error protection without extending too much beyond the JTAG framework.

Error protection for input data (TDI) is achieved by making it directly observable on the output pin (TDO) with one clock cycle delay. Output data can be shifted out twice (multiple) and then compared for maximum error protection.

#### **Busy**

Cerberus is considered busy when the requested read or write operation has not yet been finalized.

#### **LSB First**

All data and addresses are shifted in and out with LSB first.

#### **External Host is Master**

The external host is master of all transactions, initiating the transfers for both directions.

### **6.9.1.2 Serial Bit Stream Syntax (TDI, TDO)**

When Cerberus is selected, it is controlled with the TDI bit stream with the JTAG sequence Capture\_DR, multiple Shift\_DRs, and Update\_DR. The first 4 bits shifted in are the I/O instruction ([Figure 6-33](#)). The next bits (busy bits) are ignored, until a start bit occurs on TDO. Busy bits can occur for all I/O instructions except IO\_CONFIG, when the previous operation has not yet finished.

If the instruction is a write type instruction ([Table 6-59](#)), the TDI bit, in parallel to the start bit, is used as the first data bit, followed by the rest of the data and ending with a “don’t care” bit. If more data bits are shifted in than required, the first (superfluous) data bits are ignored and the last are used for the update.

If the instruction is a read type instruction ([Table 6-59](#)), all TDI bits after the instruction are ignored. After the start bit on TDO, the read data is shifted out.

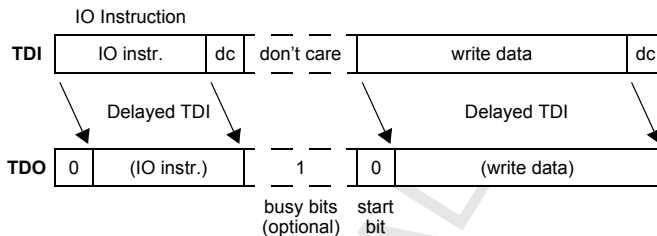
If the instruction is undefined or not implemented, the client responds with an indefinite number of busy bits.



Figure 6-33 Serial Bit Stream Syntax for TDI and TDO in Shift\_DR State

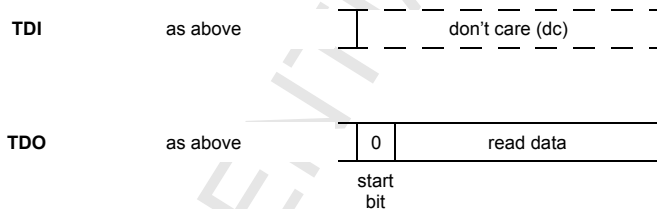
**IO Instructions  
of Write Type**

IO\_CONFIG  
IO\_SET\_ADDRESS  
IO\_WRITE\_BLOCK  
IO\_WRITE\_WORD  
IO\_WRITE\_BYTE  
IO\_SET\_TRADDR



**IO Instructions  
of Read Type**

IO\_READ\_BLOCK  
IO\_READ\_WORD  
IO\_CLIENT\_ID  
IO\_SUPERVISOR



**CONFIDENTIAL**

**Cerberus**

### 6.9.1.3 I/O Instructions

**Table 6-59** lists the I/O instructions. Unlike the JTAG instructions of the JTAG Module (**Section 11.9.3 JTAG (on Page 1248)**), they are not transferred to the JTAG instruction register with an IR Scan, but are the first four bits of a DR Scan to the shift register of Cerberus.

**Table 6-59 Cerberus I/O Instructions**

Instructions	Code	Type	Description
IO_CONFIG	0 <sub>H</sub>	W	Set the configuration register <b>IOCONF</b> .
IO_SET_ADDRESS	1 <sub>H</sub>	W	Set the address register <b>IOADDR</b> .
IO_WRITE_BLOCK	2 <sub>H</sub>	W	Write data block starting with the address in <b>IOADDR</b> .
IO_READ_BLOCK	3 <sub>H</sub>	R	Read data block starting with the address in <b>IOADDR</b> .
IO_WRITE_WORD	4 <sub>H</sub>	W	RW mode: Write word Communication mode: Send word
IO_READ_WORD	5 <sub>H</sub>	R	RW mode: Read word Communication mode: Request word
Reserved	7 <sub>H</sub> -6 <sub>H</sub>		
IO_WRITE_BYTE	8 <sub>H</sub>	W	RW mode: Write byte Communication. mode: Reserved
Reserved	9 <sub>H</sub>		
IO_SET_TRADDR	A <sub>H</sub>	W	Set the <b>TRADDR</b> register.
IO_SUPERVISOR	B <sub>H</sub>	R	Acknowledge resets and analyze bus locking situations.
Reserved	E <sub>H</sub> -C <sub>H</sub>		
IO_CLIENT_ID	F <sub>H</sub>	R	Read the Client ID.

**IO\_CONFIG** is used to abort RW Mode write operations and to configure Cerberus with the IOCONF register. The IO\_CONFIG instruction never produces any busy bits. Note that when the IO\_CONFIG instruction becomes active, the last RW mode write operation is aborted (soft reset).

**IO\_SET\_ADDRESS** sets the address IOADDR for the next RW Mode access.

**IO\_READ\_WORD** is used to read data in RW Mode or to receive data in Communication Mode. **IO\_READ\_BLOCK** is for RW Mode only. The only difference from IO\_READ\_WORD is that the address for IO\_READ\_BLOCK is post-incremented by a word address. Read instructions can be aborted when the external host sets the Update\_DR state. For IO\_READ\_WORD in Communication Mode, at least 4 shift cycles

**CONFIDENTIAL****Cerberus**

must occur after the output of the start bit to acknowledge the read. This prevents the loss of read data words.

**IO\_WRITE\_WORD** is used to write data in RW Mode or to send data in Communication Mode. **IO\_WRITE\_BLOCK** is used in RW Mode only. The only difference from **IO\_WRITE\_WORD** is that the address for **IO\_WRITE\_BLOCK** is post-incremented by a word address. For all write instructions (also for **IO\_WRITE\_BYTE**), at least 4 shift cycles must occur after the output of the start bit for the write that is actually requested in the Update\_DR state. This allows the success of the last write (start bit) to be checked without initiating a new write.

The **IO\_WRITE\_BYTE** instruction is a special case of **IO\_WRITE\_WORD** for writing bytes. For **IO\_WRITE\_BYTE**, it is required that a complete 16 bit word must be shifted in from which the lower byte is always written (for even and uneven addresses).

The **IO\_SET\_TRADDR** instruction sets the **TRADDR** register which is used for tracing with external bus address ([Section 6.9.3.4 Tracing with External Bus Address \(on Page 308\)](#)).

The **IO\_SUPERVISOR** instruction is used to release RW Mode and Communication Mode from the Error state ([Section 6.9.4 Error Handling \(on Page 310\)](#)). This instruction also outputs the **IOINFO** register after a start bit.

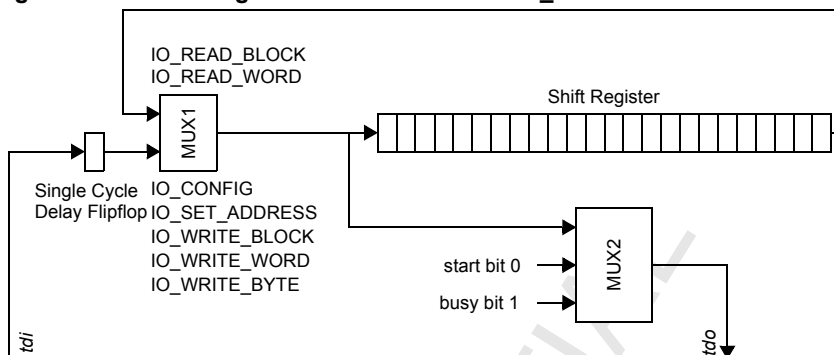
**IO\_CLIENT\_ID** returns a client-specific ID code from register **CLIENT\_ID**.

#### **6.9.1.4 Shift Register Behavior**

[Figure 6-34](#) shows the relationships among TDI, TDO, and the Shift Register content of Cerberus after the client instruction has been shifted in. MUX1 is controlled by the active instruction, MUX2 is controlled by the status of the client (busy or operation finished).

In the case of I/O write type instructions, after the TDO start bit occurs, the delayed data is shifted into the shift register and in parallel is output on TDO. In the case of I/O read type instructions, the captured data is shifted out via MUX1 and MUX2. The shift register forms a circular buffer that can be used for double shift out for error protection.

**Figure 6-34 Shift Register Behavior in the Shift\_DR State**



### 6.9.1.5 Data Transfer Examples

**Figure 6-35** shows the behavior of the JTAG I/O Interface for the IO\_CONFIG instruction. In this figure, the TDI/TDO output is shown only for the Shift\_DR state.

**Figure 6-35 Example: IO\_CONFIG**

```
IO_CONFIG
IOCONF 01h RWDATA 0000h IOADDR 000000h
tdi: 000001000000000
tdo: 000000100000000
```

**Figure 6-36** shows the behavior of the JTAG I/O Interface for the IO\_SET\_ADDRESS instruction for two cases. In the first case, there are no busy bits and the first address bit is shifted in parallel with the start bit. In the second case, there are four busy bits and the external host starts to shift in the address one cycle after the start bit. The result of both cases is exactly the same

**Figure 6-36 Example: IO\_SET\_ADDRESS**

```
1 IO_SET_ADDRESS
  IOCONF 01h RWDATA 0000h IOADDR 000033h
  tdi: 10001110011000000000000000000000
  tdo: 01000011001100000000000000000000

2 IO_SET_ADDRESS
  IOCONF 01h RWDATA 0000h IOADDR 000033h
  tdi: 1000111111110011000000000000000000
  tdo: 0100011110110011000000000000000000
```

**Figure 6-37** shows the behavior of the JTAG I/O Interface for the IO\_WRITE\_WORD instruction. There is 1 busy bit and the first data bit is shifted in parallel with the start bit.



**CONFIDENTIAL**

**Cerberus**

## 6.9.2 Registers

**Table 6-60 Cerberus Register Summary**

Register	Width	Address	Description
<b>DSWEVT</b>	16	- <Helveti ca.7pt>	Client ID
<b>IOADDR</b>	24	- <Helveti ca.7pt>	Address for next RW mode accesses
<b>IOCONF</b>	8	- <sup>1)</sup>	Configuration register
<b>IOINFO</b>	16	- <Helveti ca.7pt>	Chip state analysis register
<b>TRADDR</b>	4	- <Helveti ca.7pt>	External bus trace mode address
<b>COMDATA</b>	16	F068 <sub>H</sub>	Communication Mode data register
<b>RWDATA</b>	16	F06A <sub>H</sub>	RW mode data register
<b>IOSR</b>	16	F06C <sub>H</sub>	Status register

<sup>1)</sup> Only accessible from the JTAG port.

### 6.9.2.1 CLIENT\_ID Register

The **CLIENT\_ID** register allows that the external debugger checks the hardware in an auto-configuration mode.

#### CLIENT\_ID

**Client Type Identification Register**

**Reset value: 01UU<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPE								VERSION				REVISION			

Field	Bits	Type	Description
<b>REVISION</b>	3:0	r	Silicon revision
<b>VERSION</b>	7:4	r	Cerberus Version
<b>TYPE</b>	15:8	r	Client type (01 <sub>H</sub> )

In current implementation the overall register value is hardcoded as **0121<sub>H</sub>**.

### **6.9.2.2 IOADDR Register**

#### **IOADDR**

The **IOADDR** register holds the 24 bit address for the next Cerberus access. **IOADDR** is updated in the Update\_DR state with the shift register contents when the IO\_SET\_ADDRESS instruction is active or incremented by two (16 bit word) if an IO\_READ\_BLOCK or IO\_WRITE\_BLOCK instruction has been executed.

CONFIDENTIAL

**CONFIDENTIAL**

**Cerberus**

### 6.9.2.3 IOCONF Register

The **IOCONF** register is used to configure Cerberus. The **IOCONF** register is write only for the host and is not accessible from the MCU side.

#### IOCONF

#### Configuration Register

Reset value: 00<sub>H</sub>

7	6	5	4	3	2	1	0
RESERVED			EX BUS TRACE	TRIGGER ENABLE	COM SYNC	COM MODE RST	MODE

Field	Bits	Type	Description
<b>MODE</b>	0	w	0 Communication Mode (refer to <a href="#">Section 6.9.3.2 Communication Mode (on Page 305)</a> )
			1 RW Mode (refer to <a href="#">Section 6.9.3.1 RW Mode (on Page 304)</a> )
<b>COM_MODE_RST</b>	1	w	0 No action
			1 In the Communication Mode, <b>IOSR.CRSYNC</b> and <b>IOSR.CWSYNC</b> are reset (not static; it is only done once when <b>IOCONF</b> is updated).
<b>COM_SYNC</b>	2	w	0 No action
			1 Sets <b>IOSR.COM_SYNC</b>
<b>TRIGGER_ENABLE</b>	3	w	0 No action
			1 While in the RW Mode, the next transfers must be triggered by the DPEC event action (refer to <a href="#">Section 6.8.5 Debug Event Actions (on Page 277)</a> and <a href="#">Section 6.9.3.3 Triggered Transfers (DPEC) (on Page 307)</a> ) provided by the OCDS Module.
<b>EX_BUS_TRACE</b>	4	w	0 No action
			1 Enables triggered transfers to an external bus address (refer to <a href="#">Section 6.9.3.4 Tracing with External Bus Address (on Page 308)</a> ).
<b>RESERVED</b>	7:5	r	Reserved; these bits must be left at their reset values.



CONFIDENTIAL

Cerberus

#### 6.9.2.4 IOINFO Register

The **IOINFO** register is provided to analyze bus locking situations or certain other chip internal error states. It is not a physical register, but represents certain chip state information. After an IO\_SUPERVISOR instruction, this information is shifted out.

*Note: The captured signals are usually static only during these locking and error situations. This means that **IOINFO** should not be used during normal operation, and if used in error situations (no start bit for RW mode operation), it should be read out several times to ensure that the sampled values are static.*

#### IOINFO

##### State Information for Error Analysis

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											P BUS HLD	LM BUS HLD	EXT BUS HLD	PWR DWN	IDLE

Field	Bits	Type	Description
IDLE	0	r	0 Chip is not in idle state
			1 Chip is in idle state
POWER_DOWN	1	r	0 Chip is not in power down state
			1 Chip is in power down state
EXTBUS_HOLD	2	r	0 External or X-Bus is not busy
			1 External or X-Bus is busy
LMBUS_HOLD	3	r	0 LM-Bus is not busy
			1 LM-Bus is busy
PBUS_HOLD	4	r	0 Peripheral Bus is not busy
			1 Peripheral Bus is busy
RESERVED	15:8	r	Reserved; these bits must be left at their reset values.

#### 6.9.2.5 TRADDR Register

##### TRADDR

The 4 bit wide **TRADDR** register is used for tracing with external bus address ([Section 6.9.3.4 Tracing with External Bus Address \(on Page 308\)](#)). It defines the uppermost 4 bits of the external bus address. It is set with the IO\_SET\_TRADDR instruction by the external host.

CONFIDENTIAL

Cerberus

## 6.9.2.6 RWDATA and COMDATA Registers

### RWDATA COMDATA

The **RWDATA** register is used as the data register for both read and write transfers in RW Mode.

**COMDATA** is the equivalent register for Communication Mode.

## 6.9.2.7 IOSR Register

The **IOSR** register is used in Communication Mode ([Section 6.9.3.2 \(on page 305\)](#)), to disable Cerberus from the MCU side for security reasons ([Section 6.9.5 System Security \(on Page 310\)](#)) and to do monitor controlled tracing ([Section 6.9.3.5 Monitor Controlled Tracing \(on Page 308\)](#)). The **IOSR** register is only accessible from the MCU side.

### IOSR

#### Status and Control Register

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MTR CTL P	MTR CTL	RESERVED				CLT ON	DBG ON	COM SYNC	CW ACK	CW SYNC	CR SYNC	RW EN P	RW ENA BLD	RW DIS P	RW DISA BLE

Field	Bits	Type	Description
RW_DISABLE	0	r(w <sup>1</sup> )	<b>RW Mode Protection</b> 0 RW mode is enabled 1 RW mode is disabled
RW_DIS_P	1	w	<b>RW_DISABLE Bit Protection</b> 0 Unchanged 1 <b>RW_DISABLE</b> will be changed
RW_ENABLED	2	rw(h)	<b>Used by user program for security</b> Reset by a JTAG reset (h) only and not by a MCU reset.
RW_EN_P	3	w	<b>RW_ENABLED Bit Protection</b> 0 Unchanged 1 <b>RW_ENABLED</b> will be changed
CRSYNC	4	rh	<b>Read Sync for Communication Mode</b> 0 No receive request pending 1 External debugger requests value ( <b>COMDATA</b> )

**CONFIDENTIAL**

**Cerberus**

Field	Bits	Type	Description
<b>CWSYNC</b>	5	rh	<b>Write Sync for Communication Mode</b> 0 No send request pending 1 External debugger offers value ( <b>COMDATA</b> )
<b>CW_ACK</b>	6	w	<b>Write Request Acknowledge in Communication Mode</b> 0 No action 1 Acknowledge that sent value was read from <b>COMDATA</b> by the monitor
<b>COM_SYNC</b>	7	rh	<b>High Level Sync for Communication Mode</b> 0 <b>IOCONF.COM_SYNC</b> is 0 1 <b>IOCONF.COM_SYNC</b> is 1
<b>DBG_ON</b>	8	rh	<b>External Debugger</b> 0 Not present 1 Present
<b>CLNT_ON</b>	9	rh	<b>Client Select</b> 0 Not selected 1 Selected
<b>MTR_CTRL</b>	14	rw	<b>Monitor Controlled Tracing</b> 0 Disabled 1 Enabled
<b>MTR_CTRL_P</b>	15	w	<b>MTR_CTRL Bit Protection</b> 0 Unchanged 1 <b>MTR_CTRL</b> will be changed
<b>RESERVED</b>	11:13	r	Reserved; these bits must be left at their reset values.

<sup>1)</sup> Can be written in Communication Mode only.

The **RW\_DISABLE** bit is used to prevent Cerberus from entering RW Mode. It can only be set by the MCU in Communication Mode. If Cerberus has already entered RW Mode, all attempts by the MCU to set this bit are ignored. The application of **RW\_DISABLE** is described in [Section 6.9.5 System Security \(on Page 310\)](#).

The **RW\_ENABLED** bit has no effect on Cerberus behavior. It is provided to the user program to store whether RW Mode is enabled already or not, because it is not affected by a chip reset. The application of **RW\_ENABLED** is described in [Section 6.9.5](#).

**CRSYNC**, **CWSYNC**, **CW\_ACK** and **COM\_SYNC** bits are used in Communication Mode ([Section 6.9.3.2 Communication Mode \(on Page 305\)](#)).

CONFIDENTIAL

Cerberus

The **DBG\_ON** bit indicates whether an external debugger is present. It is directly controlled by the internal JTAG reset signal. The application of **DBG\_ON** is described in [Section 6.9.5](#).

The **CLNT\_ON** bit indicates whether this Cerberus is currently selected by the external debugger. It is directly controlled by the Cerberus select signal that is set with the [IOPATH Register \(on Page 1255\)](#) in the JTAG Module. The application of **CLNT\_ON** is described in [Section 6.9.3.5 Monitor Controlled Tracing \(on Page 308\)](#).

The **MTR\_CTRL** field can be used by a monitor to control the tracing of memory locations ([Section 6.9.3.5](#)).

*Note: This feature may be used only if no external debugger controls Cerberus across the JTAG Interface.*

## 6.9.3 Operation Modes

### 6.9.3.1 RW Mode

The RW Mode is used by the external host to read or write memory locations. In the RW Mode, the instructions **IO\_READ\_WORD**, **IO\_WRITE\_WORD**, **IO\_READ\_BLOCK**, **IO\_WRITE\_BLOCK** and **IO\_WRITE\_BYTE** are used in their generic meaning. The data address is in [IOADDR](#) and is set with **IO\_SET\_ADDRESS**. The RW Mode needs the DPEC Interface to actively request data reads or writes.

#### Entering RW Mode

The RW Mode is entered when the [IOSR.RW\\_ENABLED](#) bit is 0 and the external host writes a 1 to the [IOCONF.MODE](#) bit.

#### Data Type Support

The default data type is a 16-bit word and it is used for single word transfers and block transfers. If the external host wants to read a single byte, it must read the associated word (**IO\_READ\_WORD**) and extract the needed byte by itself.

Writes to bytes are supported with the **IO\_WRITE\_BYTE** instruction. Also, for this instruction, the external host must shift in the full 16-bit word, but only the selected byte is actually written. Its position is defined by the lowest address bit in [IOADDR](#).

#### DPEC Interface

The DPEC Interface does the actual read or write of memory locations. It is configured with the [IOCONF](#) register and the transactions are requested by the JTAG Shift Core ([Figure 6-32](#)). The data is transferred to/from the [RWDATA](#) register. DPECs always have the highest MCU priority, but they can not interrupt ATOMIC/EXTx sequences.

CONFIDENTIAL

Cerberus

### 6.9.3.2 Communication Mode

The Communication Mode is a mode of Cerberus for communication between an external host (debugger) and a program (monitor) running on the MCU. Also, in this mode, the external host is master of all transactions. The external host requests the monitor to write or read a value to/from **COMDATA**. The difference from RW mode is that in the Communication Mode, the read or write request is not actively executed by Cerberus, but it sets request bits in a MCU accessible register to signal the monitor that the host wants to send (IO\_WRITE\_WORD) or receive (IO\_READ\_WORD) a value. The monitor must poll this I/O status register **IOSR**. The **IOADDR** register is not used.

Host and monitor exchange data directly with the **COMDATA** register. For the synchronization of host and monitor accesses, there are four associated control bits **CRSYNC**, **CWSYNC**, **CW\_ACK**, and **COM\_SYNC** in the Cerberus status register **IOSR**. **CRSYNC**, **CWSYNC**, and **COM\_SYNC** are set and cleared by hardware, but can be read by the monitor (MCU). On the JTAG side, they affect the start bit on TDO. **CW\_ACK** is set by the monitor and acknowledges that the sent value was read from **COMDATA**.

The Communication Mode assures that all send and receive transactions are served under all conditions in the correct sequence, even if Cerberus changes to the RW Mode in the meantime.

For bidirectional communication, the host simply switches between the IO\_READ\_WORD and IO\_WRITE\_WORD instructions.

#### Entering Communication Mode

The Communication Mode is the default mode after reset. If Cerberus is in the RW Mode, the Communication Mode is entered when the external host writes a 0 to the **IOCONF.MODE** bit.

#### Communication Mode Instructions

Communication Mode uses only the IO\_WRITE\_WORD and IO\_READ\_WORD instructions. An IO\_SET\_ADDRESS instruction sets **IOADDR** just as in the RW Mode, but there is no effect in the Communication Mode.

#### Monitor to Host Data Transfer (Receive)

The **IOSR.CRSYNC** bit signals the monitor (MCU) that the external host wants to receive a new **COMDATA** value. It is set in the Communication Mode with the active *rd\_request* signal for the IO\_READ\_WORD instruction. The **CRSYNC** bit is automatically cleared when the monitor (MCU) writes to **COMDATA** independent of the mode (Communication Mode or RW Mode). The host can request data (**CRSYNC** is not reset during Update\_DR), do something in the RW Mode, and then fetch the requested data with the next receive cycle, refer to [Table 6-61](#).

**Table 6-61 IOSR.CRSYNC Bit**

CRSYNC	Description
0	No read requests pending.
1	Host requests monitor to write a value to <b>COMDATA</b> .

### Host to Monitor Data Transfer (Send)

The **IOSR.CWSYNC** bit signals the monitor (MCU) that the external host has written a new value to the **COMDATA** register. It is set in the Communication Mode with the **IO\_WRITE\_WORD** instruction. The **CWSYNC** bit is cleared when the monitor (MCU) sets the **IOSR.CW\_ACK** acknowledge bit independent of the mode (Communication Mode or RW Mode). This allows sending data in the Communication Mode, switching to the RW Mode, and, then, doing some other operations without having to wait until the monitor has read **COMDATA**. The next time that the Communication Mode is entered, busy bits are output when **COMDATA** was not already read by the monitor, refer to **Table 6-62**.

*Note: In case of a send (IO\_WRITE\_WORD) followed by receive (IO\_READ\_WORD), both bits **CWSYNC** and **CRSYNC** are set and must be served by the monitor in this sequence.*

*Note: A previous receive request blocks the send. This means that a requested value must be fetched by the host before it issues a new send command.*

**Table 6-62 IOSR.CWSYNC Bit**

CWSYNC	Description
1	Host requests monitor to read a value from <b>COMDATA</b> .
0	<b>COMDATA</b> not valid or <b>COMDATA</b> read by the monitor (MCU)

### Aborting Requests

If the monitor (MCU) does not serve the request (read or write **COMDATA**), the **CWSYNC** and **CRSYNC** bits can be reset with the **IOCONF.COM\_MODE\_RST** bit.

### High Level Synchronization

To improve the robustness of the communication channel, it is very helpful to distinguish between commands from the debugger and regular data exchange. For example, if the debugger aborts its request just when the monitor responds, the high level synchronization between host and monitor would be lost.

To prevent this, a **IOCONF.COM\_SYNC** bit is provided to synchronize the communication channel between debugger and monitor on a higher level. It can be read in **IOSR** by the debugger. The debugger/monitor can simply use this bit to reset the communication channel or for a more advanced use, can use this bit to tag data from the debugger to the monitor as instructions.

### 6.9.3.3 Triggered Transfers (DPEC)

Triggered transfers are an OCDS specific feature of Cerberus. They can be used to read or write a certain memory location when an OCDS trigger becomes active.

Triggered transfers are executed when Cerberus is in the RW Mode, the **IOCONF.TRIGGER\_ENABLE** bit is 1, the JTAG Shift Core has requested a transaction, and an OCDS DPEC event action ([Section 6.8.5 Debug Event Actions \(on Page 277\)](#)) occurs.

Triggered transfers behave like normal transfers, except that there must also be a transfer trigger after the JTAG Shift Core requests the transfer.

### Tracing of Memory Locations

The main application for triggered transfers is to trace a certain memory location. This can be done when the OCDS core activates the DPEC event action if this memory location is written by the user program. Cerberus is configured to read the location on this trigger. The maximum transfer rate that can be reached is defined by:

$$N_{instr} = \frac{30}{T_{instr} \cdot f_{JTAG}} \quad [0.6]$$

$N_{instr}$  is the number of instruction cycles that need to be between two MCU accesses to the memory location.  $T_{instr}$  is the instruction cycle time of the MCU and  $f_{JTAG}$  is the clock rate of the JTAG Interface (TCK). For instance, if  $T_{instr} = 100$  ns and  $f_{JTAG} = 10$  MHz accesses in every 30th instruction cycle can be fully traced. In many cases, this will be sufficient to trace something, for instance, the task ID register. The factor 30 is the sum of 16 bits for the data, 10 bits for the JTAG state machine, I/O instruction and start bit, and 4 bits for the synchronization between the transfer trigger and the shift out.

If the trigger rate is higher, some accesses are lost. To notify the external debugger about these missed events, the **dirty\_bit** read tag is set. This bit is appended to the read data when it is shifted out, refer to [Table 6-63](#).

**Table 6-63** *dirty\_bit* Read Tag

Value	Description
0	Not the case above
1	At least one missed transfer trigger event between the last triggered read and the current.

### 6.9.3.4 Tracing with External Bus Address

This is a special operating mode of the DPEC Interface for faster tracing. In this mode, the data is not written to **RWDATA** and shifted out via the JTAG port, but is directly written to an external bus address. The data is then captured from the external bus by the debugger ("trace box"). This kind of tracing can be enabled in the Communication Mode only and can be used in parallel to it.

The condition for transfers is that **IOCONF.MODE** = 0, **IOCONF.TRIGGER\_ENABLE** = 1, **IOCONF.EX\_BUS\_TRACE** = 1 and a transfer trigger exists. The external bus address is defined by:

23			0
TRADDR	0 <sub>H</sub>	F0 <sub>H</sub>	6A <sub>H</sub>

The **TRADDR** register sets the most significant bits, the rest is hardwired to 0 F06A<sub>H</sub>.

### 6.9.3.5 Monitor Controlled Tracing

Monitor controlled tracing is provided for tracing in the end product when it is (mechanically) inconvenient to make the JTAG Interface accessible.

*Note: It is very important that the monitor uses this feature only when no external debugger is connected to Cerberus across JTAG. Otherwise, errors will occur because this feature shares resources (**COMDATA**, **RWDATA**) with the normal modes used by the external debugger.*

Monitor controlled tracing is not a security risk. Even if it is unintentionally enabled by a user program, a transfer occurs only when the OCDS triggers it. The enabling of the OCDS is very well protected (**Section 6.8.2 Enabling and Disabling the OCDS (on Page 273)**).



CONFIDENTIAL

Cerberus

### 6.9.3.5.1 COMDATA Usage in Monitor Controlled Tracing Mode Register

#### COMDATA

COMDATA Usage in Monitor Controlled Tracing Mode

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MTR_ADDR															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
MTR_ADDR	15:0	w	Sets bits 15:0 of selected (RWDATA.MTR_SELECT_ADDR) address register.

### 6.9.3.5.2 RWDATA Usage in Monitor Controlled Tracing Mode

#### RWDATA

RWDATA Usage in Monitor Controlled Tracing Mode

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						MTR_SELECT_ADDR		MTR_ADDR_X							

Field	Bits	Type	Description
MTR_ADDR_X	7:0	w	Sets bits 23:16 of selected (MTR_SELECT_ADDR) address register.
MTR_SELECT_ADDR	9:8	w	00 No selection 01 Select MTR source address 10 Select MTR target address 11 Reserved
RESERVED	15:10	r	Reserved; these bits must be left at their reset values.

Monitor controlled tracing is equivalent to triggered transfers ([Section 6.9.3.3 Triggered Transfers \(DPEC\) \(on Page 307\)](#)) but is controlled by a monitor running on the MCU. It can be used to move an arbitrary memory location on an OCDs core DPEC event action ([Section 6.8.5.1 Trigger Data Transfer \(DPEC\) \(on Page 277\)](#)). The transfer is executed when Cerberus is not selected (CLNT\_ON = 0), IOSR.MTR\_CTRL is set, and there is a transfer trigger.

CONFIDENTIAL

Cerberus

Source and target addresses are programmed with **RWDATA.MTR\_SELECT\_ADDR**, **RWDATA.MTR\_ADDR\_X** and **COMDATA.MTR\_ADDR**. With a write to **RWDATA**, the address (source or target) is selected (**MTR\_SELECT\_ADDR**) and the highest byte of the address is written. The lower 16 bits can then be programmed with **COMDATA**.

The following C-source code example shows how a monitor enables the trace:

```
// Trace source address: 0x543210
unsigned trace_source_ptr_ext = 0x54;
unsigned trace_source_ptr = 0x3210;

// Trace target address: 0xABCDEF
unsigned trace_target_ptr_ext = 0xAB;
unsigned trace_target_ptr = 0xCDEF;

// Setting the trace source address
RWDATA = 0x0100 | trace_source_ptr_ext;
COMDATA = trace_source_ptr;

// Setting the trace target address
RWDATA = 0x0200 | trace_target_ptr_ext;
COMDATA = trace_target_ptr;

// Starting the monitor controlled trace with MTR_CTRL
IOSR = 0xC000

// Programming the OCDS to create DPEC triggers:
...
```

#### 6.9.4 Error Handling

Cerberus enters the error state on all chip internal resets (except JTAG reset). It can be left with the **IO\_SUPERVISOR** instruction. While in error state, every instruction except **IO\_SUPERVISOR** responds with an indefinite number of busy bits.

Another error state is when the chip internal bus is blocked for DPEC transfers. If this condition occurs, the **IO\_SUPERVISOR** instruction can be used to read the **IOINFO** register which provides analysis information.

#### 6.9.5 System Security

After reset, Cerberus is in the Communication Mode and needs at least 30 TCK clock cycles to be brought into the RW Mode (10 cycles to acknowledge the reset with **IO\_SUPERVISOR** and 20 cycles to set **IOCONF**). If the user program running in the MCU sets **CCONF (on Page 1257).RST\_HLT** immediately after reset, there is no way from the outside to get Cerberus into the RW Mode via the JTAG Interface.

CONFIDENTIAL

Cerberus

To have a protected system in the field that can be accessed by authorized users, the following solution can be used:

1. If **IOSR.RW\_ENABLED** is 0, the first instruction of the user program after reset disables the RW Mode by setting **RST\_HLT** to 1.
2. The user program checks **IOSR.DBG\_ON** to determine if an external debugger is present. If not, it just continues with the regular code.
3. External debugger sends key numbers ( $n \times 16$  bits) in the Communication Mode.
4. User program starts to accept and compare these number some time  $t_d$  after reset. This time must be long enough (about 100 ms) to allow even a slow (5 kHz) JTAG driver to shift in the send request. Additionally, it is recommended to poll **CRSYNC** in reasonable distances to allow a hot attach of the external debugger.
5. If all numbers are correct, the user program resets **RST\_HLT** and sets **IOSR.RW\_ENABLED**.
6. Now, the user program knows (**IOSR.RW\_ENABLED**) that Cerberus has been enabled once and thus does not prevent the enabling after the next resets.

*Note: Average time to crack the system for:  $n = 2$  and  $t_d := 1$  s:  $2^{32} * 1$  s / 2 = 1634 years.*

### 6.9.6 Power Saving

Cerberus is in Power Saving Mode when it is not selected from the JTAG side. The only register that is always accessible and working is **IOSR**.

If the monitor controlled tracing mode ([Section 6.9.3.5 Monitor Controlled Tracing \(on Page 308\)](#)) is enabled, the required resources are functional.

### 6.9.7 Reset Behavior

#### Reset from the JTAG Side

If the internal JTAG reset becomes active, all RW Mode and Communication Mode requests are aborted and also the **CRSYNC** and **CWSYNC** bits are reset. The behavior of the registers is specified in [Table 6-64 Register Reset Behavior \(on Page 312\)](#).

#### Reset from the Chip/MCU Side

In this case, all I/O instructions except **IO\_CONFIG** are responded to with an indefinite number of busy bits (Error state). The external host must acknowledge this state with the **IO\_SUPERVISOR** instruction as described in [Section 6.9.4 Error Handling \(on Page 310\)](#). This is done to notify the external host that something possibly unexpected has happened and that it must check such things as the communication channel to the monitor.

**Table 6-64 Register Reset Behavior**

Register	JTAG Reset	Chip/MCU Reset
<b>CLIENT_ID</b>	Hardwired	Hardwired
<b>COMDATA</b>	Unchanged	0000 <sub>H</sub>
<b>IOADDR</b>	000000 <sub>H</sub>	Unchanged
<b>IOCONF</b>	00 <sub>H</sub>	Unchanged
<b>IOINFO</b>	Chip specific	Chip specific
<b>IOSR</b>	UUUU UUUU UUUU U0UU <sub>B</sub> SW <sup>1)</sup> : UUUU UU00 UUUU U0UU <sub>B</sub>	0000 0000 0000 0U00 <sub>B</sub> SW<Helvetica.7pt>: 0000 00UU 0000 0U00 <sub>B</sub>
<b>RWDATA</b>	Unchanged	0000 <sub>H</sub>
<b>TRADDR</b>	0 <sub>H</sub>	Unchanged

<sup>1)</sup> From the software point of view, bits [9:8] have this behavior because their origin is in the JTAG reset controlled domain. Only their synchronization in flip-flops is connected to the chip/MCU reset.

*Note: A JTAG reset always requires a following MCU reset to ensure that the JTAG Shift Core and the control part of Cerberus are in a defined state under all conditions.*

## CONFIDENTIAL

## Configuring External Bus and MCU Signals

## 6.10 Configuring External Bus and MCU Signals

### 6.10.1 General

The following section describes the previous and new modes and properties used in the port logic to support the external BUS. Also described is the configuration of the signals RSTOUT and CLKOUT (the signal ALE is not available in E-GOLDradio). The MCU controls the signals for the external bus automatically but the application software may have, after RESET, to modify the pin logic to connect/disconnect some of these signals to/from device pins.

In general, the bus functions of each of these ports have been implemented. The alternative and port I/O functions for the external bus signals have been implemented differently from those of the standard C16x controller family.

Some optional bus functions can be replaced at the pins by a different port or alternative functions.

All ports and pins assigned to bus functions are always in the same bus mode (HOLD, or Normal External Access).

The modes XPER-SHARE and EMULATOR are not supported in E-GOLDradio.

The Input threshold control register, PICON is not used. No ports use TTL levels. The input signal definitions are in the [Chapter 13 Electrical and Temperature Characteristics \(on Page 1297\)](#). For a description of the control registers for the port and pin logic, refer to [Section 10.13 External Bus Unit \(on Page 851\)](#). For information on pin and alternative functions refer to [Chapter 3 Pin Descriptions \(on Page 49\)](#).

### 6.10.2 Data Bus Functions Pins D(7:0) and D(15:8)

In the E-GOLDradio the data bus functions for D(7:0) have no alternative port functions, (D15:8) have alternative port functions in port A, PA(15:8), refer to [Chapter 3](#).

*Note: In standard C166 controllers the data bus uses port Zero.*

The data bus can be used for 8 or 16-bit data, but the multiplexed bus mode is not supported.

The Data bus pins perform the following functions controlled directly by the bus controller (except RESET):

**RESET:**

Port tri-stated, **NO** pins used as inputs. Any external pull-ups/pull downs used on this bus during this mode have no influence on the bus or controller starting configuration. (The start configuration is determined by internal preprogrammed registers and the state of the pins MON1 and MON2 during RESET, refer to [Chapter 14 System Reset \(on Page 1401\)](#)).

**CONFIDENTIAL**

**Configuring External Bus and MCU Signals**

If the 8-bit bus start configuration is selected the pins PA(15:8) remain in tri-state during and after reset till they are configured before by software as address lines or used for some other purpose.

**EXTERNAL BUS:** The active data pins output during write operations. Data pins are in input mode during read operations. Data is transferred to the X-Bus data bus only when the internal bus controller control signal is active.

**VISIBLE:** Refer to [Section 6.10.9 Visible Mode \(on Page 318\)](#).

**HOLD:** Data pins tri-stated, and internal XBUS-DATA is not driven from port. The MCU can not access XPERs or external memory but can access internal program memory and internal dual port memory, but no external access to XPERs is permitted. VISIBLE accesses cause internal wait states until HOLD mode is exited.

**Input/ Output:** Port I/O not supported for D(7:0). If only a 8-bit bus is required, the pins otherwise used for the upper 8-bits, D(15:8) of the data port can be reprogrammed as PA(15:8).

**Idle Mode:** Tri-state pull down if used as data bus.

**Power Down Mode:** Tri-state pull down if used as data bus.

CONFIDENTIAL

Configuring External Bus and MCU Signals

### 6.10.3 Address Bus Functions

In the E-GOLDradio the address lines A(22:0) have no alternative functions and no GPIO functions. Pins A(22:0) generate the lower 23-bits of an address.

The address line A23 has alternate functions and a GPIO function, and its default function is the 24th bit of address after RESET. These pins remain in a fixed state during and after the end of RESET until the EINIT instruction unless they are configured before by software as address lines or used for some other purpose.

Below are the possible modes for the address bus pins:

**RESET:** Pins A(23:0) are tri-state, pull down.

**Immediately after RESET:**

The pins A(0:23) are driven and latched for addresses.

**EXTERNAL BUS:** Pins A(23:0) are driven and latched for addresses during transfers.

**VISIBLE:** Refer to [Section 6.10.9 Visible Mode \(on Page 318\)](#).

**HOLD:** **Port is tri-stated, pull-down, and internal X-Bus address lines are not driven from ports.**

**Input/Output:** General purpose I/O and alternative functions are not supported for A(22:0).  
A23 has alternate functions and a GPIO function controlled by PCL\_31.

**Idle Mode:** Tri-state pull down if used as address bus.

**Power Down Mode:** Tri-state pull down if used as address bus.

*Note: The number of pins allocated to segment addresses cannot be configured during Reset.*

**CONFIDENTIAL**

**Configuring External Bus and MCU Signals**

### 6.10.4 Port D Functions

Refer to [Section 6.10.3 Address Bus Functions \(on Page 315\)](#).

### 6.10.5 Chip Select and HOLD Functions

Refer to [CS Signal Generation \(on Page 862\)](#).

If the HOLD mode is required the pin logic has to be configured to connect the functions to external pins. If the pins are configured the  $\overline{\text{HOLD}}$  pin becomes active after the HOLD mode has been enabled in the Processor Status Word register, [PSW](#).

The  $\overline{\text{CSx}}$  signals identify access to different address ranges with the C16x memory map. The number of pins assigned to chip selects can be changed by configuring the pin logic by the application SW after RESET. The possible modes for this port are:

**RESET:** The signals  $\overline{\text{CS0}}$  and  $\overline{\text{CS1}}$  are always configured during Reset and may be used immediately following RESET if an external bus mode has been selected (refer to [Chapter 14 System Reset \(on Page 1401\)](#)).

The internal HLDA signal to the controller core is not connected to a pin after Reset.

**EXTERNAL BUS:** The associated chip select for the external address range used will become active.

**VISIBLE:** Refer to [Section 6.10.9 Visible Mode \(on Page 318\)](#).

**HOLD:** Chip select signals are driven high in the clock cycle where the HLDA is driven active low. Afterwards they are tristate pull up. (In the hold state, internal X-Bus signals are not driven from external port pins.)

### 6.10.6 $\overline{\text{RD}}$ , $\overline{\text{WR}}$ , and $\overline{\text{READY}}$ Pins

The  $\overline{\text{RD}}$ ,  $\overline{\text{WR}}$ , and  $\overline{\text{READY}}$  functions are dedicated to bus operations. The  $\overline{\text{RD}}$  and  $\overline{\text{WR}}$  are dedicated output pins.  $\overline{\text{READY}}$  is one of two possible inputs for Port C.1. Below are the possible modes for these pins:

**RESET:** The  $\overline{\text{RD}}$  and  $\overline{\text{WR}}$  signals are pulled to high using a weak pull-up device. The  $\overline{\text{READY}}$  pin is configured as a general purpose Port C.1 which is floating. The external value is not seen by the core because the reset value of the [BUSCONx \(on Page 870\)](#) register causes the ready input to be ignored.

**EXTERNAL BUS:** The  $\overline{\text{WR}}$ , and  $\overline{\text{RD}}$  signals are generated by the bus controller and driven on their associated pins. The  $\overline{\text{READY}}$  pin is used as an input only if it has been correctly configured, and used by the bus controller if the function is enabled in the associated [BUSCONx](#).

**VISIBLE:** Refer to [Section 6.10.9 Visible Mode \(on Page 318\)](#).



**CONFIDENTIAL**

**Configuring External Bus and MCU Signals**

**HOLD:** The external values of the pins  $\overline{RD}$  and  $\overline{WR}$ , return to their Reset values, high using a weak pull-up, and can be overdriven by a second master. X-peripheral accesses are not possible.

**Idle Mode:**  $\overline{RD}$  and  $\overline{WR}$  are high.

**Power Down mode:**  $\overline{RD}$  and  $\overline{WR}$  are high.

### 6.10.7 Byte High Enable ( $\overline{BHE}$ )

Port F.15 provides the  $\overline{BHE}$  bus control function as alternative function which is not selected automatically on RESET. The Port is reset to a state (pull up) which permits 16-bit access to any 16-bit memory. It has to be reconfigured to the alternative function  $\overline{BHE}$  and the **SYSCON (on Page 109)** register set correctly before byte accesses to 16-bit memories are used.  $\overline{BHE}$  is only used when enabled in the **SYSCON** register. The following table should be ignored when  $\overline{BHE}$  is disabled.<sup>1)</sup>

**RESET:** The Port F.15 function tri-state pull up is selected. The port has to be reconfigured before EINIT (otherwise, it will float) and before the first external byte write access.

**EXTERNAL BUS:** If configured and in 16-bit mode, the  $\overline{BHE}$  pin is driven from the bus controller for any external access. The exact function is depends on the entry in the **SYSCON** register.

**HOLD:** If configured, the  $\overline{BHE}$  pin enters tri-state. Internal XPER accesses are not possible.

**Idle Mode:** If configured,  $\overline{BHE}$  retains its last value.

**Power down mode:** If configured,  $\overline{BHE}$  retains its last value.

Refer to **Summary of Use of WRL, WRH, A0, and  $\overline{BHE}$  (on Page 860)** for information about on external bus accesses.

### 6.10.8 $\overline{RSTOUT}$ and $\overline{CLKOUT}$

**RESET:**  $\overline{RSTOUT}$  pin is Pull down but configured as port I/O. If the  $\overline{RSTOUT}$  is required the alternate function must be selected before EINIT.

$\overline{CLKOUT}$  pin is reset to a tristate port input function. The alternative  $\overline{CLKOUT}$  function has to be selected by software.

**EINIT:**  $\overline{RSTOUT}$  goes high if the pin is still configured as  $\overline{RSTOUT}$  when the EINIT instruction is executed.

**HOLD:** No influence on  $\overline{RSTOUT}$  and  $\overline{CLKOUT}$ .

<sup>1)</sup> When  $\overline{BHE}$  is disabled, this pin can be used for any port function. Note that this function is enabled during RESET when a 16-bit wide bus is used until disabled by software.

**CONFIDENTIAL**

**Configuring External Bus and MCU Signals**

**Idle Mode:** RSTOUT is HIGH if EINIT was executed before entering IDLE mode,  
LOW otherwise.  
CLKOUT remains active if selected.

### 6.10.9 Visible Mode

The VISIBLE mode accesses of XPERs use the programmed XPER bus configuration externally. Thus, 16-bit accesses to the X-Bus are truncated externally to 8-bit if only eight external data bits are used.

The visible mode which is controlled by the **SYSCON.VISIBLE** bit in the core has a reduced functionality compared to the standard controller specification.

**Data Bus:** If in the VISIBLE mode, XPER accesses always drive read or write data values to external pins.  
If not in the VISIBLE mode, XPER accesses **do not** drive read or write data values to external pins.

**Note: The external bus address and data signals are not driven during not-visible internal accesses.**

If there is an extended, period without an external access these signals may float away from the last value driven and could cause excessive power consumption in the inputs of external memories.

**Address Bus:** If in the VISIBLE mode, XPER accesses drive address values to pins A(23:0).  
If not in the VISIBLE mode, XPER accesses **do not** drive address values to pins A(23:0).

**Note: Caution, the external bus address and data signals are not driven during not-visible internal accesses. If there is an extended period without an external access these signals may float away from the last value driven and could cause excessive power consumption in the inputs of external memories.**

**Chip Selects:** If in the VISIBLE mode, XPER accesses have no effect on external chip selects. They must remain high during "VISIBLE accesses"  
If not in the VISIBLE mode, XPER accesses have no effect on external chip selects.

**HOLD:** This input is not effected by the visible mode.

**HLDA:** If in the VISIBLE mode, XPER accesses have no effect on the hold acknowledge output.  
If not in the VISIBLE mode, XPER accesses have no effect on the HLDA pin.

**CONFIDENTIAL**

**Configuring External Bus and MCU Signals**

**$\overline{RD}$ ,  $\overline{WR}$ ,  $\overline{READY}$ :** The  $\overline{WR}$ , and  $\overline{RD}$  pins always directly follow their internal X-Bus equivalents.  $\overline{READY}$  is not driven externally for VISIBLE accesses whether it is enabled or disabled in the XPER **XBCONx** register, but the XPER must activate this signal internally if  $\overline{READY}$  function is enabled.

**$\overline{BHE}$ :** Not generated for X-Bus accesses.

**$\overline{RSTOUT}$ :** Not dependant on visible mode

**$\overline{CLKOUT}$ :** Not dependant on visible mode

**VISIBLE in HOLD MODE:**

If the processor is currently in the HOLD mode, the MCU waits until the external bus becomes available before accessing any XPER.

VISIBLE mode accesses of XPERs use the programmed XPER bus configuration externally. Thus, users who wish to perform visible demultiplexed XPER accesses must specify at least one **BUSCONx** as demultiplexed to allocate Port 1 to bus functions. 16-bit accesses to the X-Bus are truncated externally to 8-bit if only 8 external data bits are used.

CONFIDENTIAL

**CONFIDENTIAL**

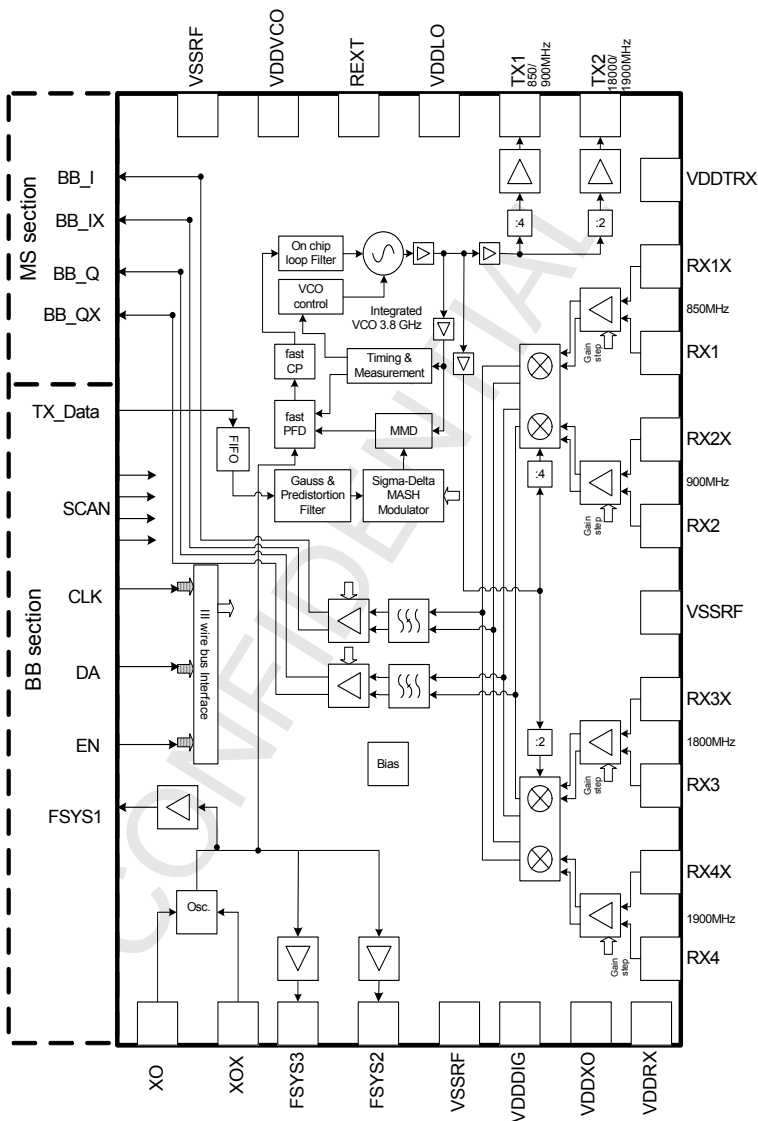
## 7 Functional Description of RF Part

<b>History</b>	
Design	Current version: Rev. 1.06, 2005-11-04
Specification	Previous version: Rev. 1.05, 2005-08-02
Page	Subjects (major changes since last revision)
	Initial revision based on SMARTi-SD2 PMB6271 Target Specification, Rev 2.0
Changes for Rev. 1.00	
Changes for Rev. 1.06	

CONFIDENTIAL

**CONFIDENTIAL**

**Figure 7-1 Functional Block Diagram of RF Part**

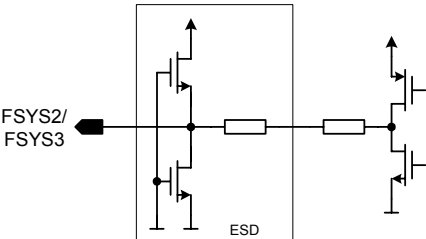
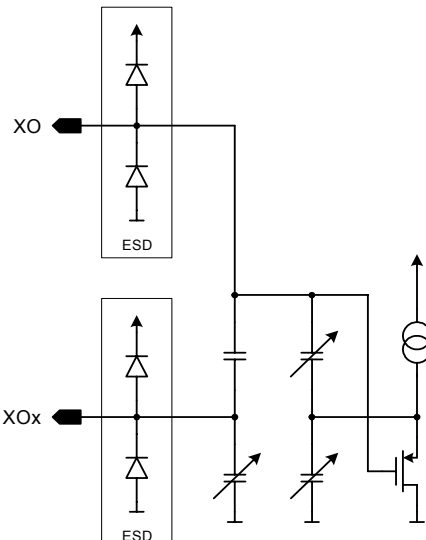


**CONFIDENTIAL**

**Pin Definition and Function**

**7.1 Pin Definition and Function**

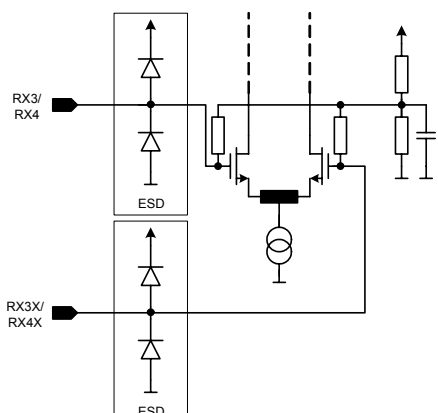
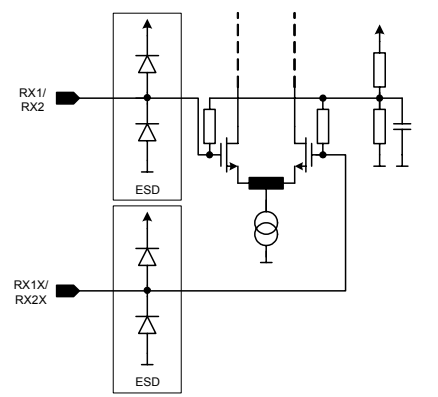
**Table 7-1 Pin Definition and Function**

Symbol	Equivalent I/O-Schematic	Function
VDDRX		VDD2V5
FSYS3		26 MHz reference clock output 3
FSYS2		26 MHz reference clock output 2
VDDXO		VDD2V5
XO		26 MHz crystal
XOx		26 MHz crystal
VDDDIG		VDD1V5

**CONFIDENTIAL**

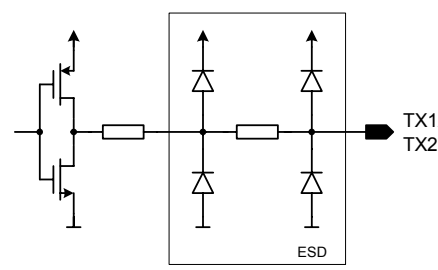
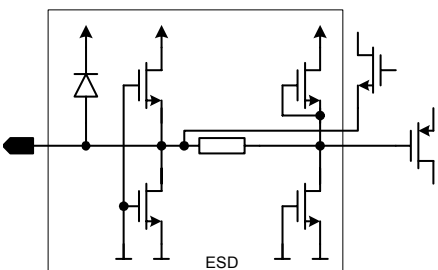
**Pin Definition and Function**

**Table 7-1 Pin Definition and Function**

Symbol	Equivalent I/O-Schematic	Function
RX4		GSM 1900 Rx input
RX4X		GSM 1900 Rx input
RX3		GSM 1800 Rx input
RX3X		GSM 1800 Rx input
VSSRF		
RX2		GSM 900 Rx input
RX2X		GSM 900 Rx input
RX1		GSM 850 Rx input
RX1X		GSM 850 Rx input
VDDTRX		VDD1V5



**Table 7-1 Pin Definition and Function**

Symbol	Equivalent I/O-Schematic	Function
TX2		GSM 1800/1900 Tx output
TX1		GSM 850/900 Tx output
VDDLO		VDD1V5
REXT		External reference resistor
VDDVCO		VDD2V5

## 7.2 Functional Block Description

### 7.2.1 Receiver

The PMB7870 features a fully integrated quad-band constant gain direct conversion receiver, that is, there is no interstage filter needed and the baseband level at the analogue IQ-interface follows directly the RF input level. Dependent on the baseband ADC dynamic range single or multiple step gain switching schemes are possible. An integrated self-aligning low-pass filter ensures the receivers functionality under blocking and reference interference conditions as well as avoids aliasing by baseband sampling. Furthermore an automatic DC offset compensation is implemented and can be switched dependent on the gain setting.

**CONFIDENTIAL****Functional Block Description****7.2.1.1 RF Front-End and Demodulator**

The PMB7870 RF front-end contains four integrated LNAs for GSM850/900/1800/1900 with balanced inputs. The amplified RF-signal is direct converted by a quadrature demodulator to the final output signals at the baseband frequency. The orthogonal LO signals are internally generated by a divider by four for GSM850/900 band and by two for the GSM1800/1900 band.

**7.2.1.2 Baseband Stage**

The resulting in-phase and quadrature signals are fed into the baseband stage, that comprises low pass filtering, programmable gain steps (refer to [Section 7.2.1.3 Gain Concept \(on Page 326\)](#)) and an automatic DC offset compensation circuitry. The fully integrated baseband filter provides sufficient suppression of blocking signals and adjacent channel interferers to match optimally with baseband ADCs providing 72dB dynamic range at full scales from  $1 V_{pp}$  up to  $4 V_{pp}$ . The ADCs anti-aliasing requirements are fulfilled for sampling rates from 6.5 MHz on. The low pass filter is self-aligning with a residual 3dB roll off frequency tolerance of  $\pm 7\%$ .

**7.2.1.3 Gain Concept**

The PMB7870 receiver gain concept is optimized for internal baseband ADCs providing a noise floor 72dB below full scales from  $1 V_{pp}$  up to  $4 V_{pp}$  within 0..100 kHz ideally. A signal-to-distortion ratio of SDR > 50 dBFS for frequency offsets up to 600 kHz is recommended.

With the PMB7870 constant gain receiver architecture no complex PGA algorithm is needed. The PMB7870 provides three reference gains high/medium/low programmed by RXGAIN[1:0] and four relative gain steps -3/-6/+6/+12 dB controlled by RXGS[3:0] to combine with the absolute gains. The four gain steps are implemented in the output buffer and do therefore not affect the receiver noise figure.

**Single Step Constant Gain Switching Scheme**

For internal baseband ADCs providing a noise floor >82 dB below full scale a straight forward single step scheme proposal as presented in [Figure 7-2](#) can be applied. This is well proven by the combination of the predecessor PMB6270 SMART<sup>®</sup> SD with the baseband processors E-GOLDlite and S-GOLDlite (ADC enhanced mode).

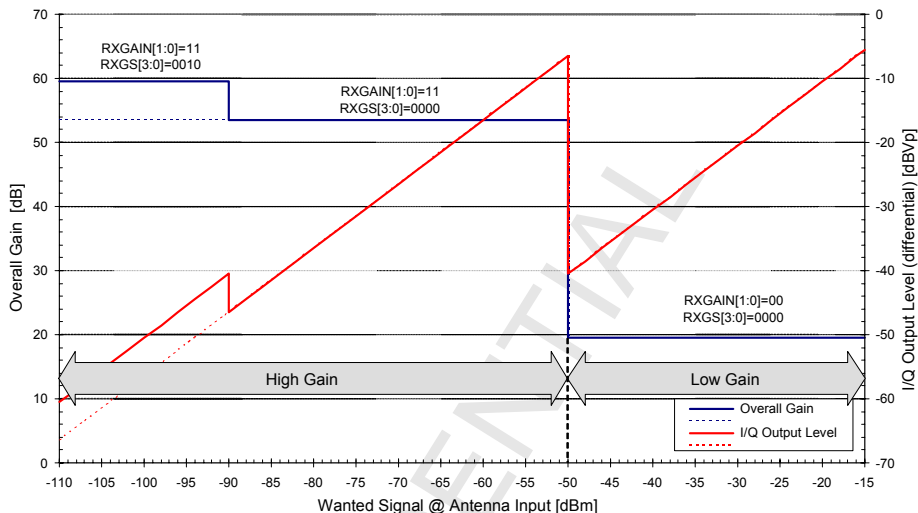
High gain is recommended for wanted signal levels up to  $P_W = -53..-50$  dBm referred to the antenna input, respectively if the signal swing at the PMB7870 I/Q-output reaches the specified full scale of the baseband ADC including sufficient margin for faded signals. To reduce the ADC noise contribution to negligibility a +6 dB gain step (RXGS1) is applied for wanted signal levels lower than  $P_W = -90$  dBm (reference sensitivity levels).

**CONFIDENTIAL**

**Functional Block Description**

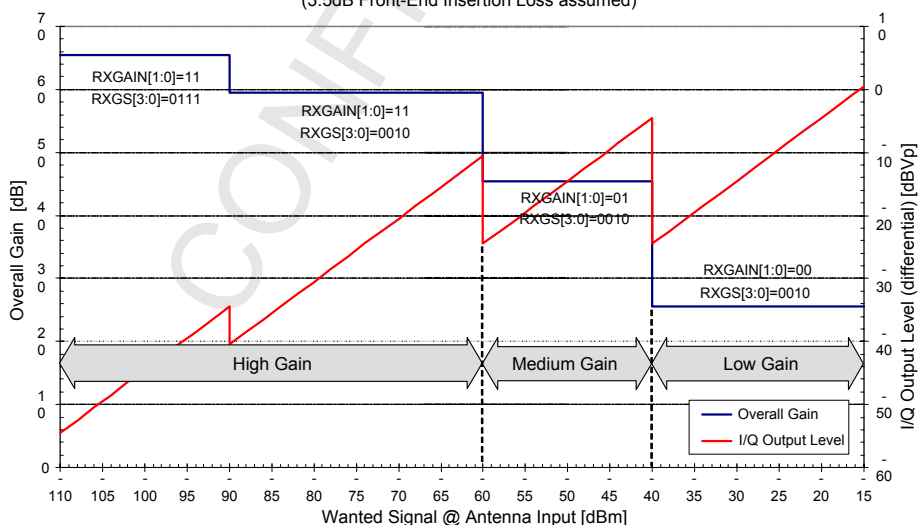
**Figure 7-2 PMB7870 Single Step, Constant Gain Switching Scheme**

PMB7870 Gain Switching Scheme for 82dB SNR ADC 2Vpp Full Scale  
(3.5dB Front-End Insertion Loss assumed)



**Figure 7-3 PMB7870 Multiple Step, Constant Gain Switching Scheme**

PMB7870 Gain Switching Scheme for 72dB SNR ADC 2Vpp Full Scale  
(3.5dB Front-End Insertion Loss assumed)



**CONFIDENTIAL****Functional Block Description****7.2.2 Transmitter**

The digital transmitter architecture is based on a fractional-N sigma-delta synthesizer for constant envelope GMSK modulation. This configuration allows a very low power design with a reduced count of external components.

The modulation is transferred between baseband- and RF-part of the PMB7870 via a digital interface signal into the digital modulator.

The following Gaussian filter shapes the digital data stream for the GMSK modulation. Additionally a pre-distortion filter compensates the attenuation of the PLL transfer function resulting in a very low distortion at the transmit output.

The filtered digital data stream is scaled appropriately and added to the channel word. This sum is fed into the MASH modulator. The output of the MASH modulator is a sequence of integer divider values representing the high resolution fractional input signal. This sequence controls the MMD (multi modulus divider) at a sample rate of 26 MHz. Thus a tightly controlled frequency modulation of the VCO is achieved.

The output signal of the VCO is divided by four for GSM 850/900 or by two for GSM 1800/1900 respectively. Finally the divided signal is amplified by a single ended output driver with 50 Ohm output impedance to allow for a direct connection to the PA.

The transmitter achieves a very low out-of-band noise, typically -163.5 dBc/Hz @ 20 MHz offset, and a very low rms phase error of 1.2 degree typically.

**7.2.3 RF-Synthesizer**

The PMB7870 contains a fractional-N sigma-delta synthesizer for the frequency synthesis in the RX and TX operation mode. The 26 MHz reference signal is provided by the internal crystal oscillator. This frequency serves as comparison frequency of the phase detector and as clock frequency of all digital circuitry.

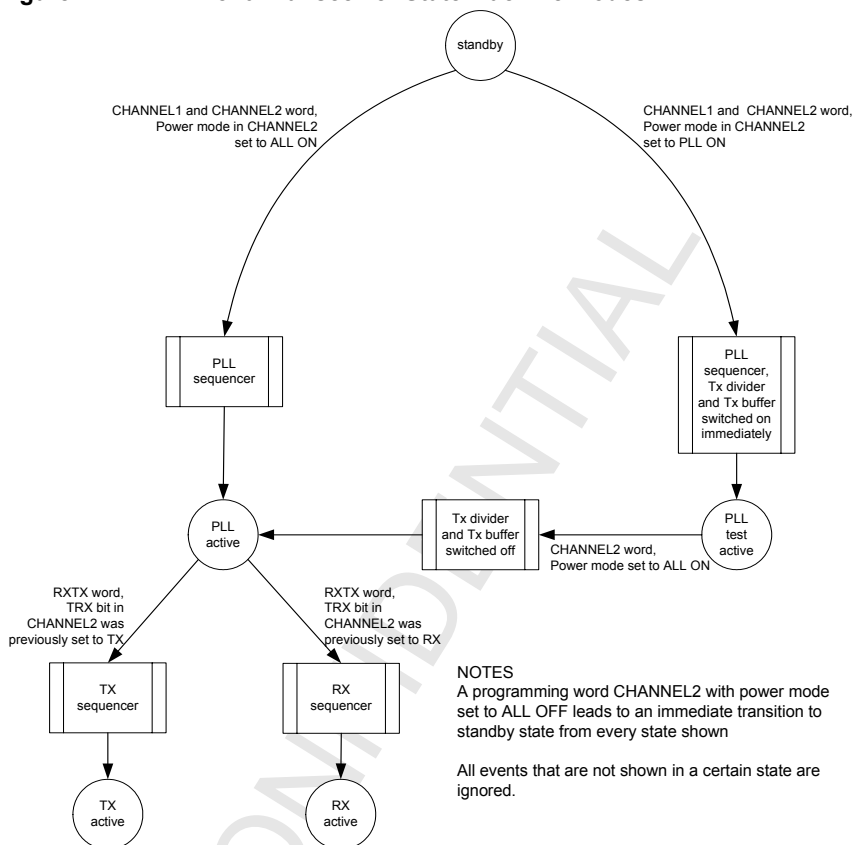
The N-counter of the synthesizer is carried out as a multi-modulus divider (MMD). The loop filter is fully integrated and the loop bandwidth is about 100 kHz to allow the transfer of the phase modulation.

The fully integrated quad-band VCO is designed for the four GSM bands (850, 900, 1800, 1900 MHz) and operates at the double or four times transmit or receive frequency. To cover the wide frequency range the VCO is automatically aligned by a binary automatic band selection (BABS) before each synthesizer startup.

**7.2.4 Power Up Sequencer**

The different transceiver states are defined by a central state machine. The mode transitions are achieved by sending one of the 3-wire-bus telegrams CHANNEL2 or RXTX. As prerequisite the 26 MHz crystal oscillator has to be active.

**Figure 7-4 PMB7870 Transceiver State Machine Modes**



Most of the mode transitions cause the start of a power up sequencer. There are three sequencers implemented on the chip:

- PLL sequencer
- RX sequencer
- TX sequencer

The PLL sequencer controls the internal alignments and startup of the sigma delta synthesizer. There is no effect on any signal outputs.

The RX and TX sequencer timings are shown in [Figure 7-5](#) and [Figure 7-6](#). According to the state diagram the PLL must be started to enable an activation of the RX or TX sequencer.

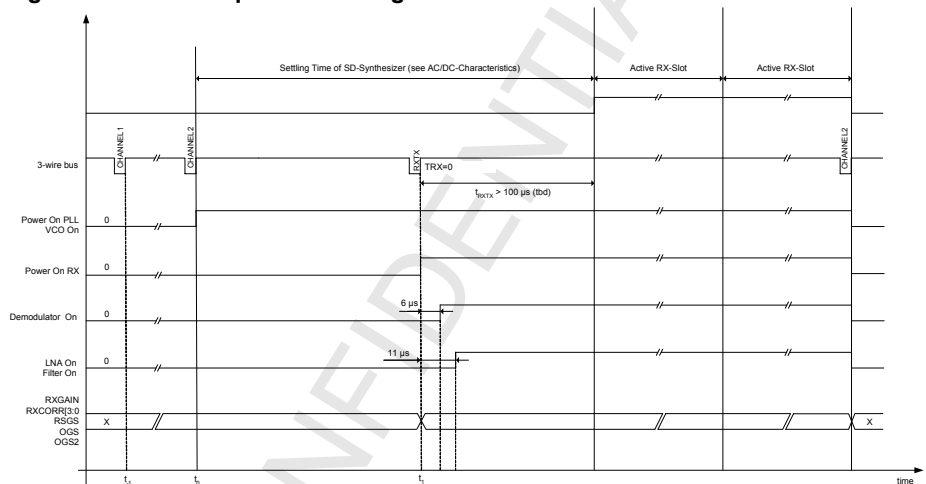
**CONFIDENTIAL**

**Functional Block Description**

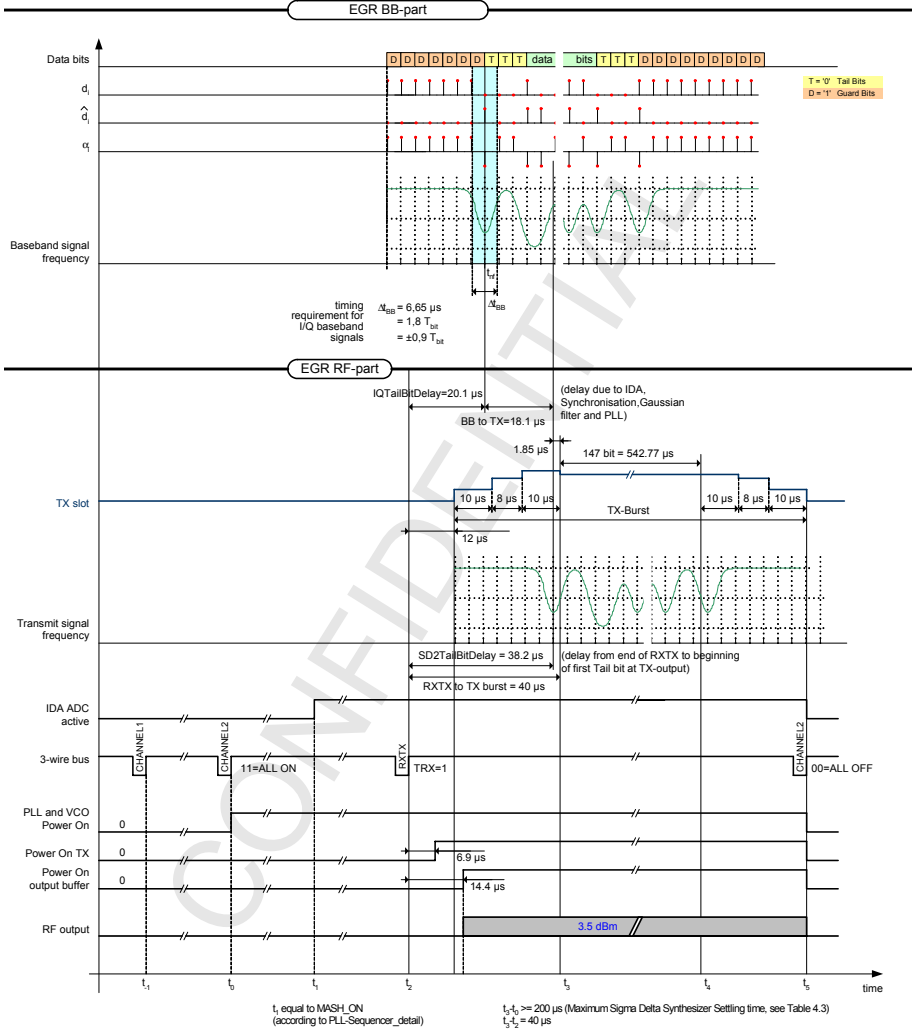
The TX sequencer also controls the timing of the transmit output buffer. 14.4 $\mu$ s after the end of the RXTX word the transmit output buffer is activated. The so caused rise of the output level should be time aligned with the 1st step of the power time template. This reduces the isolation requirements to the PA.

The “PLL test active” mode allows to monitor the PLL lock-in transition at the transmitter output since the transmit divider and buffer are activated immediately after the CHANNEL2 word. The lock-in behavior can deviate from the normal behavior (using ALL ON and a following RXTX word) since high startup currents and settling of the internal bias in the transmit chain may cause interference to the PLL startup steps.

**Figure 7-5 RX Sequencer Timing**



**Figure 7-6 TX Sequencer Timing**



### 7.2.5 3-Wire Serial Bus

The operational functionality as well as the power-on timing of the PMB7870 is controlled by software programming via a internal the 3-wire bus (CLK, DA, EN). This interface is also accessible externally when in a certain test mode.

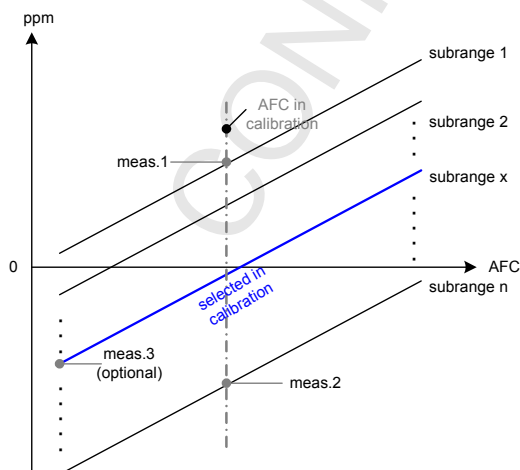
The required programming data is split into several 24 bit long registers. For operation three 24 bit telegrams before the active slot and 1 telegram for power down after the active slot are needed. Before programming the 3-wire bus VDDXO (2.5V) and VDDDIG (1.5V) have to be applied. Three initiation telegrams are needed after the activation of the crystal oscillator to adjust frequency and operating mode.

### 7.2.6 DCXO

The PMB7870 contains a fully integrated 26 MHz digitally controlled crystal oscillator (DCXO).

The overall pulling range of the DCXO consists of 8 subranges. The sub-range closest to the 0ppm at the middle AFC-value is selected during the calibration process in production. The spacing between the sub-ranges is constant. Therefore, for a proper sub-range selection only 2 measurements are sufficient. Afterwards, the selected sub-range number will be stored in the software and used for the mobile's lifetime. An additional measurement in calibration can be used to determine the AFC value, which corresponds to the 0 ppm along the selected sub-range. The pulling along the selected sub-range is used for correction of all kinds of frequency drift. The figure below shows the sub-range selection.

**Figure 7-7 DCXO Sub-range Selection**





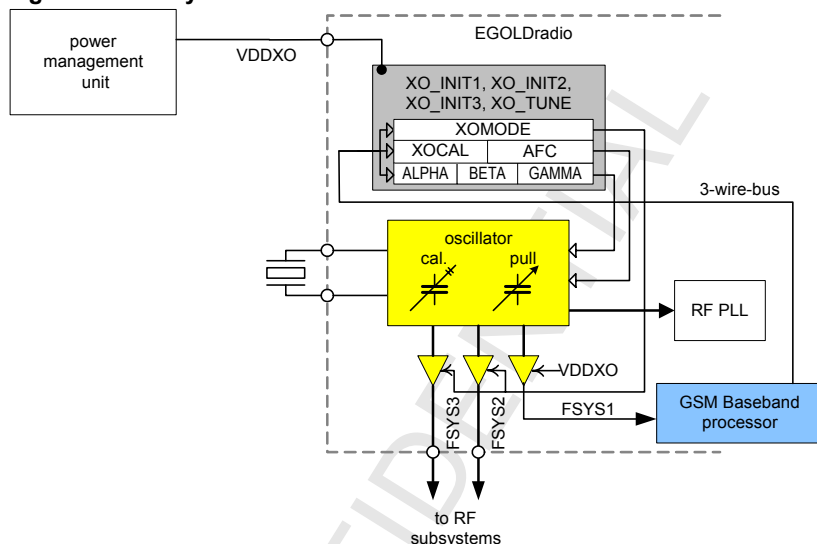
**CONFIDENTIAL**

**Functional Block Description**

**Functional Overview**

The PMB7870 is able to provide 26 MHz frequency reference signal for up to 2 additional RF subsystems (such as Bluetooth or GPS) via two output buffers (FSYS2 and FSYS3). This signal is derived from the internal 26MHz reference. Please refer to [Figure 7-8](#).

**Figure 7-8 Crystal Oscillator Functional Overview**



The outputs FSYS2 (Bluetooth) and FSYS3 can be activated through the bits XOMODE1 and XOMODE3 in the XO\_INIT1 register. For the first power-on procedure the output buffers FSYS2 and FSYS3 are deactivated by the power-on reset.

**The XO\_INIT1, XO\_INIT2, XO\_INIT3 and XO\_TUNE Registers**

The XO\_TUNE register holds the digital correction value for the crystal oscillator frequency. The XOMODE bits of XO\_INIT1 register contain setup informations for the crystal oscillator (for example, current programming, etc.).

The registers XO\_INIT2 and XO\_INIT3 contain the coefficients information for the linearization unit of crystal oscillator (LUXO). This linearization unit computes the required digital control word out of the programmed AFC bits in order to have a linear pulling curve ppm vs. AFC word. The resulting digital control word DIG is filtered by a digital lowpass filter, which can be scaled or deactivated using the bits DIGFILT0 and DIGFILT1 of the XO\_INIT3 register.

**CONFIDENTIAL****Functional Block Description**

The frequency correction splits into 2 parts:

- 1) The XOCAL bits in the XO\_INIT1 register are used for the coarse frequency adjustment and are set once for a mobile lifetime (during production test)
- 2) The XO\_TUNE register contains the information for frequency correction when the mobile is used (correction of temperature drift, crystal aging).

### **7.2.7 RF Supply Voltage Domains**

The PMB7870 has implemented 3 different RF power supply domains:

- 2.5 V domain for the DCXO (VDDXO)
- 2.5 V domain for the transceiver
- 1.5 V domain for the transceiver and for all programming registers.

The 2.5 V DCXO supply domain is required for the oscillator circuit to start the oscillation. The 1.5 V supply is required for programming the crystal oscillator settings (for example, sub-range, AFC setting, DCXO setup information).

The DCXO immediately starts oscillating after VDDXO is applied and the output buffer outputs the system clock to the baseband regardless of any register settings. The DCXO registers XO\_MODE and XO\_INIT1..3 can be programmed by the 3-wire-bus, which is located in the 1.5 V domain.

For starting up the voltage regulators there are requirements to the startup and power down sequence.

### **Power Up Procedure**

Please see **Figure 7-9**.

The PMB7870 supports 2 power-up scenarios:

1. In the first scenario the 1.5 V domain is powered up first. When the 1.5 V supply is settled, the power-on reset signal POR1 is generated which ties all registers to the power-on default settings. Immediately after the power-on reset the 2.5 V DCXO domain can be powered up. This causes the DCXO to start the oscillation and the 26 MHz clock signal will drive the internally connected GSM baseband part of PMB7870 a buffer. Now the DCXO settings (that is, calibration and setup of DCXO) have to be programmed by the GSM baseband processor through the 3-wire-bus into the DCXO registers.
2. In the second scenario the 2.5 V (VDDXO) is powered up first. The 1.5 V supply is not available yet. In this case the power-on reset signal POR2 is generated when the VDD2 supply reaches a certain level and forces only the DCXO registers to the power-on default settings. The DCXO is using the default sub-range. To change the register settings by programming the 1.5 V supply must be powered up first.

## Power Down Procedure

When the transceiver is powered down it is recommended to keep the 1.5 V supply on in order to keep the setting informations in the DCXO registers. Otherwise the DCXO information has to be resent each time after the power up procedure and the DCXO setup and calibration data (XO\_INIT1, XO\_INIT2, XO\_INIT3 registers) have to be reloaded before an other RF subsystem (for example, Bluetooth) starts using the 26 MHz clock.

When the 1.5 V supply has to be powered down the 2.5 V supply must be switched off before or at the same time. This will cause the DCXO to stop the oscillation.

### Figure 7-9 Power Up and Down Sequence: Scenario 1

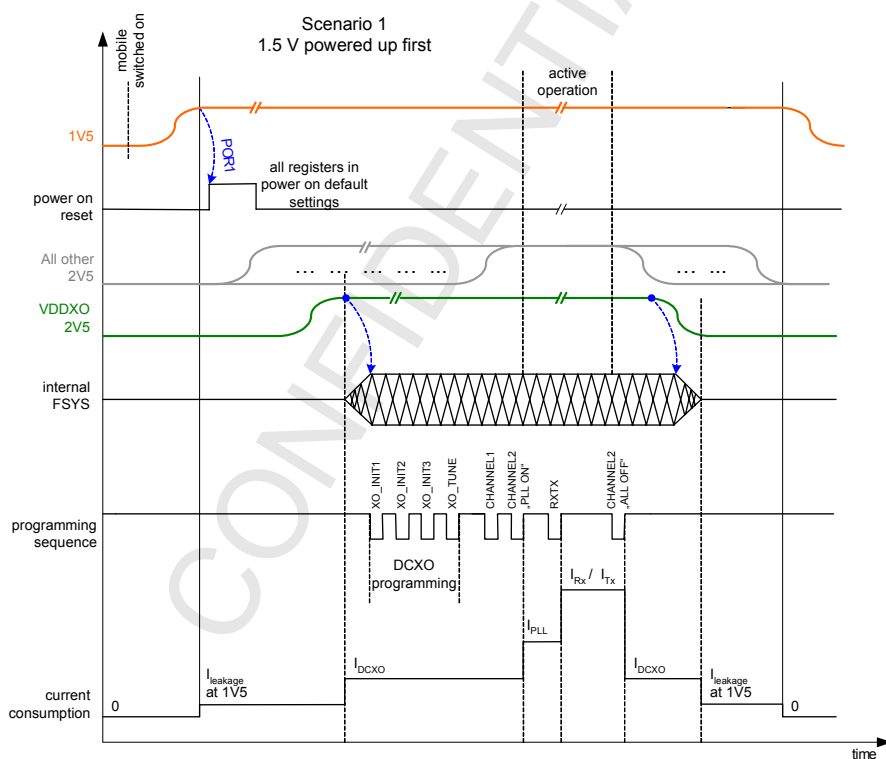
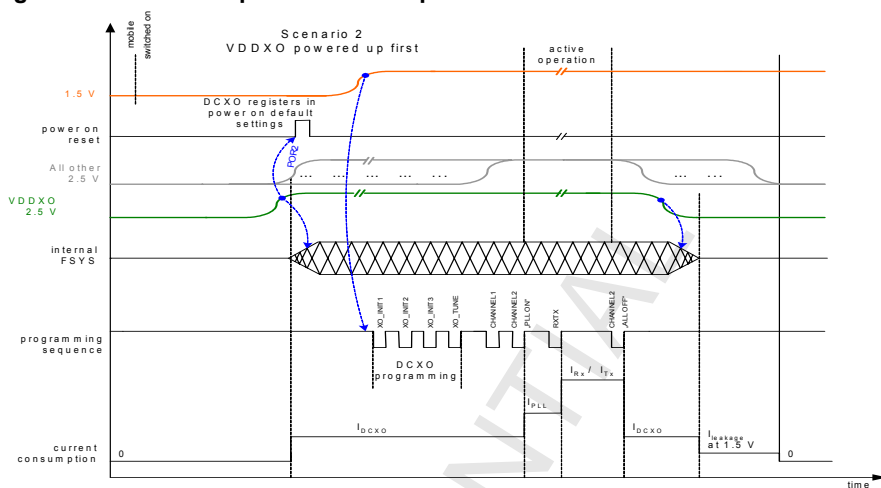


Figure 7-10 Power Up and Down Sequence: Scenario 2



## 8 TEAKLite Bus

### 8.1 TEAKLite Interrupt Unit

History	
Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.04	
<b>Page 338</b>	Updated <b>Table 8-1 Interrupt Unit Registers</b> WS00008083
Changes for Rev. 1.06	

#### System Integration on TEAKLite

- Supply domain: VDD\_DSP
- Chip internal interfaces:
  - Clock domain: gclk\_dsp\_per
  - Bus domain: TEAKLite Z-Bus
  - Interrupt sources: none
- Chip external signals related to this block: MON1, MON2.
- Monitor Pins: INT0, INT1, and INT2

#### 8.1.1 Functional Overview

The TEAKLite core supports three high-active interrupt input lines INT0, INT1, and INT2. They are controlled via TEAKLite internal registers. The TEAKLite can separately enable or disable each of them. INT0 has the highest priority. The order how the interrupts are executed is determined by the TEAKLite itself according to priority list [INT0, INT1, INT2].

To extend the possible number of interrupts, sources from different peripherals have been multiplexed to interrupt lines INT0 and INT1. No peripheral signals are connected to INT2, because it is used by firmware only. That means firmware is able to generate interrupts independently of peripheral signals, for example, for firmware task switching based on an operating system.

There are various interrupt sources which are connected to the interrupt lines: TEAKLite peripherals, signals from the system interface, and signals which are generated by the

**CONFIDENTIAL**

**TEAKLite Interrupt Unit**

MCU. Furthermore, the TEAKLite has the possibility to generate MCU interrupts. In the following chapters suffix “x” stands for the different interrupt lines 0, 1 or 2, while suffix “y” denotes the interrupt register number A0, B0, 1 or 2.

The TEAKLite interrupt unit operates at the TEAKLite peripheral clock. During reset the clock is enabled. The interrupt unit processes a total number of 56 interrupts. 24 of those interrupts are assigned to INT0, while each of the interrupts INT1 and INT2 is connected to 16 different interrupt sources. **Table 8-1** gives an overview on the registers inside the interrupt unit which are provided to control the different sources.

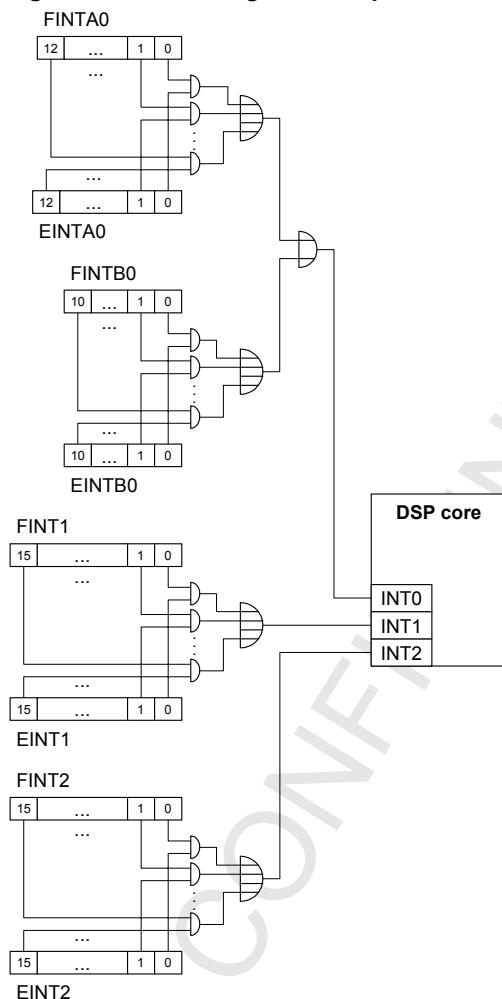
**Table 8-1      Interrupt Unit Registers**

Interrupt Line	Flag Register(S)	Enable Register	Reset Register	Set Register
INT0	<b>INT_FINTA0</b> (13 bits)	<b>INT_EINTA0</b> (13 bits)	<b>INT_RINTA0</b> (13 bits)	<b>INT_SINTA0</b> (13 bits)
	<b>INT_FINTB0</b> (11 bits)	<b>INT_EINTB0</b> (11 bits)	<b>INT_RINTB0</b> (11 bits)	<b>INT_SINTB0</b> (11 bits)
INT1	<b>INT_FINT1</b> (16 bits)	<b>INT_EINT1</b> (16 bits)	<b>INT_RINT1</b> (16 bits)	<b>INT_SINT1</b> (16 bits)
INT2	<b>INT_FINT2</b> (16 bits)	<b>INT_EINT2</b> (16 bits)	<b>INT_RINT2</b> (16 bits)	<b>INT_SINT2</b> (16 bits)

*Note: Because INT0 has more than 16 possible sources the control and status bits have been split on two register sets.*

INT\_FINTy is the Flag INTerrupt register containing one bit for every interrupt source being able to generate an interrupt on the corresponding TEAKLite input INTx. An interrupt is only generated if the corresponding bit in the Enable INTerrupt register INT\_EINTy is set. The value of the bits of the register INT\_EINTy does not affect the setting or resetting of the corresponding flags in INT\_FINTy. As shown in **Figure 8-1**, an interrupt signal INTx is pending on the TEAKLite core as long as at least one bit in INT\_FINTy and the corresponding bit in INT\_EINTy is set.

**Figure 8-1 Enabling of Interrupts in TEAKLite Interrupt Unit**



All interrupt sources on interrupt line INTx have the same priority. Nevertheless the TEAKLite can decide which bits are checked first after an interrupt INTx occurs.

It is also possible to generate interrupts by firmware writing a 1 to the corresponding bit in the Set INTerrupt register INT\_SINTy. Furthermore, the TEAKLite can reset an interrupt by writing a 1 to the corresponding bit in the Reset INTerrupt register

CONFIDENTIAL

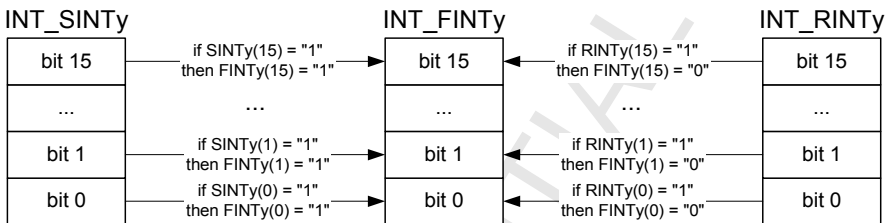
TEAKLite Interrupt Unit

INT\_RINTy. **Figure 8-2** shows the connection between FINTy, INT\_EINTy, and INT\_RINTy.

The priority of setting and resetting the bits of register INT\_FINTy is defined as follows:

1. Set by hardware
2. Set by firmware
3. Reset by firmware.

**Figure 8-2 Reset and Set INT\_FINTy of TEAKLite Interrupt Unit**



The three registers INT\_FINTy, INT\_RINTy, and INT\_SINTy are implemented as only one register, but they have different addresses. All interrupt unit registers are cleared by software and hardware reset. A detailed description of all interrupt registers is given in [Section 8.1.2 Register Descriptions \(on Page 341\)](#).

*Note: After detection of an interrupt (rising and/or falling edge) the core interrupt INTx is activated within 3 TEAKLite cycles if interrupt is enabled.*

*Note: To safely detect a toggle of an interrupt signal the new level must be valid for at least one TEAKLite cycle.*

*Note: If the TEAKLite is in IDLE-Mode (bit [DSP\\_CTRL \(on Page 77\)](#).DSPDIS = 1) and an interrupt on the interrupt lines INT0, INT1 or INT2 is generated, the bit **DSPDIS** is cleared, the TEAKLite core clock is switched on and the TEAKLite is in operational mode again and can process the interrupt.*

*Note: The non-maskable-interrupt input of the TEAKLite core is not connected.*



## 8.1.2 Register Descriptions

To obtain the addresses of these registers refer to [Section 12.1.2 Registers in the DSP Memory Space \(on Page 1266\)](#).

### 8.1.2.1 A0 Interrupt Registers

#### INT\_xINTA0

(x = F, E, R, or S)

#### INT\_FINTA0

Flag Register A to INT0 Line of TEAKLite

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			BB FUL	BBL O	BBHI	EQ	CHA DEC	MOD U	COD ONL O	COD ONH I	FRA ME	MCU 3	MCU 2	MCU 1	MCU 0

Field	Bits	Typ	Description
MCU0	0	r	0 Signal not detected 1 $\uparrow$ <i>todsp_0</i> detected -> an interrupt was activated by the MCU
MCU1	1	r	0 Signal not detected 1 $\uparrow$ <i>todsp_1</i> detected -> an interrupt was activated by the MCU.
MCU2	2	r	0 Signal not detected 1 $\uparrow$ <i>todsp_2</i> detected -> an interrupt was activated by the MCU.
MCU3	3	r	0 Signal not detected 1 $\uparrow$ <i>todsp_3</i> detected -> an interrupt was activated by the MCU.
FRAME	4	r	0 Signal not detected 1: $\uparrow$ <i>frame_int</i> or $\downarrow$ <i>frame_int</i> (INT_GP[5]) from the GSM System Interface detected.
CODONHI	5	r	0 Signal not detected 1 $\uparrow$ <i>codon</i> from the GSM timer detected.
CODONLO	6	r	0 Signal not detected 1 $\downarrow$ <i>codon</i> from the GSM timer detected.
MODU	7	r	0 Signal not detected 1 $\uparrow$ <i>mod_int</i> detected -> the specified (MINT_ADDR) modulator RAM address was reached.

**CONFIDENTIAL**

**TEAKLite Interrupt Unit**

Field	Bits	Typ	Description
<b>CHADEC</b>	8	r	0 Signal not detected 1 $\uparrow chadec\_int$ detected -> a new block in the channel decoder hardware accelerator has been calculated and is ready to be read by the TEAKLite.
<b>EQ</b>	9	r	0 Signal not detected 1 $\uparrow eq\_int$ detected -> a new block in the equalizer hardware accelerator has been calculated and is ready to be read by the TEAKLite.
<b>BBHI</b>	10	r	0 Signal not detected 1 One of $\uparrow eqon$ , $\uparrow fcon$ , $\uparrow monon$ , or $\uparrow scon$ from the GSM timer detected.
<b>BBLO</b>	11	r	0 Signal not detected 1 One of the signals $\downarrow eqon$ , $\downarrow fcon$ , $\downarrow monon$ , or $\downarrow scon$ from the GSM timer detected.
<b>BB_FULL</b>	12	r	0 Signal not detected 1 $\uparrow bbfull\_int$ detected -> the specified (INT_POINTER) baseband buffer RAM address reached.
<b>RESERVED</b>	15:13	r	Reserved; these bits must be left at their reset values.

*Note: If bits (minimum of one bit) in the register **INT\_FINTAO** are set and the corresponding interrupt enable bits in the **INT\_EINTAO** register are set, the INTO interrupt line of the TEAKLite core is activated.*

## INT\_EINTA0

Enable Register A to INT0 Line of TEAKLite

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			EBB FOL L	EBB LO	EBB HI	EEQ	ECH A DEC	EMO DU	ECO DON LO	ECO DON HI	EFR AME	EMC U3	EMC U2	EMC U1	EMC U0

Field	Bits	Typ	Description
<b>Exx</b> (xx = 0 to 12)	12:0	rw	Interrupt Line INT1 Enable for Bitxx of <b>INT_FINTA0</b> 0 Disables INT0 for the bit xx 1 Enables INT0 for the bit xx
<b>RESERVED</b>	15:13	r	Reserved; these bits must be left at their reset values.

*Note: The values of the bits in register **INT\_EINTA0** do not affect the set or reset of the corresponding flags in **INT\_FINTA0**.*

## INT\_RINTA0

Reset Register A to INT0 Line of TEAKLite

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			RBB FOL L	RBB LO	RBB HI	REQ	RCH A DEC	RMO DU	RCO DON LO	RCO DON HI	RFR AME	RMC U3	RMC U2	RMC U1	RMC U0

Field	Bits	Typ	Description
<b>Rxx</b> (xx = 0 to 12)	12:0	w	0 No action (do not write 0) 1 Resets bit xx of <b>INT_FINTA0</b> .
<b>RESERVED</b>	15:13	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**TEAKLite Interrupt Unit**

### INT\_SINTA0

**Set Register A to INT0 Line of TEAKLite**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			SBB FOL L	SBB LO	SBB HI	SEQ	SCH A DEC	SMO DU	SCO DON LO	SCO DON HI	SFR AME	SMC U3	SMC U2	SMC U1	SMC U0

Field	Bits	Typ	Description
<b>Sxx</b> (xx = 0 to 12)	12:0	w	0 No action (do not write 0) 1 Sets bit xx of <b>INT_FINTA0</b> .
<b>RESERVED</b>	15:13	r	Reserved; these bits must be left at their reset values.

### 8.1.2.2 B0 Interrupt Registers

#### INT\_xINTB0

(x = F, E, R, or S)

#### INT\_FINTB0

**Flag Register B to INT0 Line of TEAKLite**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED					SYS MCU	SSC 1 ERR	SSC 1 TX	SSC 1 RX	VB TX	VB RX	I2S3 TX	I2S2 RX	I2S2 TX	I2S1 RX	I2S1 TX

Field	Bits	Typ	Description
<b>I2S1TX</b>	0	r	0 Signal not detected 1 $\uparrow$ i2s1_tx_int detected -> the specified (TXINTADDR) RAM address in the TX buffer reached.
<b>I2S1RX</b>	1	r	0 Signal not detected 1 $\uparrow$ i2s1_rx_int detected -> the specified (RXINTADDR) RAM address in the RX buffer reached.
<b>I2S2TX</b>	2	r	0 Signal not detected 1 $\uparrow$ i2s2_tx_int detected -> the specified (TXINTADDR) RAM address in the TX buffer reached.

**CONFIDENTIAL**

**TEAKLite Interrupt Unit**

Field	Bits	Typ	Description
<b>I2S2RX</b>	3	r	0 Signal not detected 1 $\uparrow i2s2\_rx\_int$ detected -> the specified (RXINTADDR) RAM address in the RX buffer reached.
<b>I2S3TX</b>	4	r	0 Signal not detected 1 $\uparrow i2s3\_tx\_int$ detected -> the specified (TXINTADDR) RAM address in the TX buffer reached.
<b>VBRX</b>	5	r	0 Signal not detected 1 $\uparrow vbrx\_int$ detected -> the specified (RXINTADDR) RAM address in the RX buffer reached.
<b>VBTX</b>	6	r	0 Signal not detected 1 $\uparrow vbtx\_int$ detected -> the specified (TXINTADDR) RAM address in the TX buffer reached.
<b>SSC1RX</b>	7	r	0 Signal not detected 1 $\uparrow ssc1\_rx\_int$ detected -> a receive interrupt request from the SSC detected.
<b>SSC1TX</b>	8	r	0 Signal not detected 1 $\uparrow ssc1\_tx\_int$ detected -> a transmit interrupt request from the SSC detected.
<b>SSC1ERR</b>	9	r	0 Signal not detected 1 $\uparrow ssc1\_err\_int$ detected -> an error interrupt request occurred.
<b>SYSMCU</b>	10	r	0 Signal not detected 1 $\uparrow sys\_mcu\_int$ or $\downarrow sys\_mcu\_int$ (INT_GP[6]) from the GSM System Interface detected. Toggle interrupt.
<b>RESERVED</b>	15:11	r	Reserved; these bits must be left at their reset values.

*Note: If bits (minimum of one bit) in the register **INT\_FINTB0** are set and the corresponding interrupt enable bits in the **INT\_EINTB0** register are set, the INT0 interrupt line of the TEAKLite core is activated.*

### INT\_EINTB0

**Enable Register B to INT0 Line of TEAKLite**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED					ESY S MCU	ESS C1E RR	ESS C1T X	ESS C1R X	EVB TX	EVB RX	EI2S 3 TX	EI2S 2 RX	EI2S 2 TX	EI2S 1 RX	EI2S 1 TX

Field	Bits	Typ	Description
<b>Exx</b> (xx = 0 to 15)	10:0	rw	Interrupt Line INT0 Enable for Bitxx of <b>INT_FINTB0</b> 0 Disables INT0 for the bit xx 1 Enables INT0 for the bit xx
<b>RESERVED</b>	15:11	r	Reserved; these bits must be left at their reset values.

*Note: The values of the bits in register **INT\_EINTB0** do not affect the set or reset of the corresponding flags in **INT\_FINTB0**.*

### INT\_RINTB0

**Reset Register B to INT0 Line of TEAKLite**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED					RSY S MCU	RSS C1E RR	RSS C1T X	RSS C1R X	RVB TX	RVB RX	RI2S 3 TX	RI2S 2 RX	RI2S 2 TX	RI2S 1 RX	RI2S 1 TX

Field	Bits	Typ	Description
<b>Rxx</b> (xx = 0 to 10)	10:0	w	0 No action (do not write 0) 1 Resets bit xx of <b>INT_FINTB0</b> .
<b>RESERVED</b>	15:11	r	Reserved; these bits must be left at their reset values.

CONFIDENTIAL

TEAKLite Interrupt Unit

## INT\_SINTB0

Set Register B to INT0 Line of TEAKLite

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED					SSY S MCU	SSS C1E RR	SSS C1T X	SSS C1R X	SVB TX	SVB RX	SI2S 3 TX	SI2S 2 RX	SI2S 2 TX	SI2S 1 RX	SI2S 1 TX

Field	Bits	Typ	Description
<b>Sxx</b> (xx = 0 to 10)	10:0	w	0 No action (do not write 0) 1 Sets bit xx of <a href="#">INT_FINTB0</a> .
<b>RESERVED</b>	15:11	r	Reserved; these bits must be left at their reset values.

### 8.1.2.3 INT1 Interrupt Registers

#### INT\_xINT1

(x = F, E, R, or S)

#### INT\_FINT1

Flag Register to INT1 Line of TEAKLite

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MON IN4L O	MON IN4H I	MON IN3L O	MON IN3H I	MON IN2L O	MON IN2H I	MON IN1L O	MON IN1H I	DSP IN1L O	DSP IN1H I	DSP IN0L O	DSP IN0H I	TMR 2	TMR 11	TMR 10	CIPH

Field	Bits	Typ	Description
<b>CIPH</b>	0	r	0 Signal not detected 1 $\uparrow$ <i>ciph_int</i> detected -> the end of the cipher algorithm.
<b>TMR10</b>	1	r	0 Signal not detected 1 $\uparrow$ <i>dtmr10_int</i> of DSP timer 1 detected -> the counter of timer 1 reached the value of <a href="#">TMR1_INT0 (on Page 356)</a> register.
<b>TMR11</b>	2	r	0 Signal not detected 1 $\uparrow$ <i>dtmr11_int</i> of DSP timer 1 detected -> the counter of timer 1 reached the value of <a href="#">TMR1_INT1 (on Page 357)</a> register.

**CONFIDENTIAL**

**TEAKLite Interrupt Unit**

Field	Bits	Typ	Description
<b>TMR2</b>	3	r	0 Signal not detected
			1 $\uparrow dtmr2\_int$ of DSP timer 2 detected, -> the counter of timer 2 reached the value of <b>TMR2_MAX</b> (on <a href="#">Page 359</a> ) register.
<b>DSPIN0HI</b>	4	r	0 Signal not detected
			1 $\uparrow dspin0$ detected -> an interrupt generated by an external device (available via an Alternate Pin function).
<b>DSPIN0LO</b>	5	r	0 Signal not detected
			1 $\downarrow dspin0$ detected -> an interrupt generated by an external device (available via an Alternate Pin function).
<b>DSPIN1HI</b>	6	r	0 Signal not detected
			1 $\uparrow dspin1$ detected -> an interrupt generated by an external device (available via an Alternate Pin function).
<b>DSPIN1LO</b>	7	r	0 Signal not detected
			1 $\downarrow dspin1$ was detected -> an interrupt generated by an external device (available via an Alternate Pin function).
<b>MONIN1HI</b>	8	r	0 Signal not detected
			1 $\uparrow monin1$ detected -> an interrupt generated by the user for test purposes.
<b>MONIN1LO</b>	9	r	0 Signal not detected
			1: $\downarrow monin1$ detected -> an interrupt generated by the user for test purposes.
<b>MONIN2HI</b>	10	r	0 Signal not detected
			1: $\uparrow monin2$ detected -> an interrupt generated by the user for test purposes.
<b>MONIN2LO</b>	11	r	0 Signal not detected
			1: $\downarrow monin2$ detected -> an interrupt generated by the user for test purposes.
<b>MONIN3HI</b>	12	r	0 Signal not detected
			1: $\uparrow dsp\_int3$ detected -> an interrupt generated by the user for test purposes.
<b>MONIN3LO</b>	13	r	0 Signal not detected
			1 $\downarrow dsp\_int3$ detected -> an interrupt generated by the user for test purposes.



CONFIDENTIAL

TEAKLite Interrupt Unit

Field	Bits	Typ	Description
MONIN4HI	14	r	0 Signal not detected
			1 $\uparrow dsp\_int4$ detected -> an interrupt generated by the user for test purposes.
MONIN4LO	15	r	0 Signal not detected
			1 $\downarrow dsp\_int4$ detected -> an interrupt generated by the user for test purposes.

Note: If bits (minimum of one bit) in the register **INT\_FINT1** are set and the corresponding interrupt enable bits in the **INT\_EINT1** register are set, the INT0 interrupt line of the TEAKLite core is activated.

### INT\_EINT1

Enable Register to INT1 Line of TEAKLite

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EMONIN4LO	EMONIN4HI	EMONIN3LO	EMONIN3HI	EMONIN2LO	EMONIN2HI	EMONIN1LO	EMONIN1HI	EDSPIN1LO	EDSPIN1HI	EDSPIN0LO	EDSPIN0HI	ETMR2	ETMR11	ETMR10	ECIPH

Field	Bits	Typ	Description
Exx (xx = 0 to 15)	15:0	rw	Interrupt line INT1 enable for bit xx of <b>INT_FINT1</b>
			0 Disables INT1 for the bit xx
			1 Enables INT1 for the bit xx

Note: The values of the bits in the register **INT\_EINT1** do not affect the set or reset of the corresponding flags in **INT\_FINT1**.

### INT\_RINT1

Reset Register to INT1 Line of TEAKLite

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RMONIN4LO	RMONIN4HI	RMONIN3LO	RMONIN3HI	RMONIN2LO	RMONIN2HI	RMONIN1LO	RMONIN1HI	RDSPIN1LO	RDSPIN1HI	RDSPIN0LO	RDSPIN0HI	RTMR2	RTMR11	RTMR10	RCIPH

Field	Bits	Typ	Description
Rxx (xx = 0 to 15)	15:0	w	0 No action (do not write 0)
			1 Resets bit xx of <b>INT_FINT1</b> .

## INT\_SINT1

Set Register to INT1 Line of TEAKLite

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMO N IN4L O	SMO N IN4H I	SMO N IN3L O	SMO N IN3H I	SMO N IN2L O	SMO N IN2H I	SMO N IN1L O	SMO N IN1H I	SDS P IN1L O	SDS P IN1H I	SDS P IN0L O	SDS P IN0H I	STM R2	STM R11	STM R10	SCIP H

Field	Bits	Typ	Description
<b>Sxx</b> (xx = 0 to 15)	15:0	w	0 No action (do not write 0) 1 Sets bit xx of <b>INT_FINT1</b> .

## 8.1.2.4 INT2 Interrupt Registers

### INT\_xINT2

(x = F, E, R, or S)

### INT\_FINT2

Flag Register to INT2 Line of TEAKLite

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FW 15	FW 14	FW 13	FW 12	FW 11	FW 10	FW9	FW8	FW7	FW6	FW5	FW4	FW3	FW2	FW1	FW0

Field	Bits	Typ	Description
<b>FWxx</b> (xx = 0 to 15)	15:0	r	0 An interrupt for Flag xx has not been generated 1 An interrupt for Flag xx has been generated by the firmware.

*Note: If bits (minimum of one bit) in the register **INT\_FINT2** are set and the corresponding interrupt enable bits in the **INT\_EINT2** register are set, the INT0 interrupt line of the TEAKLite core is activated.*

## INT\_EINT2

Enable Register to INT2 Line of TEAKLite

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EFW 15	EFW 14	EFW 13	EFW 12	EFW 11	EFW 10	EFW 9	EFW 8	EFW 7	EFW 6	EFW 5	EFW 4	EFW 3	EFW 2	EFW 1	EFW 0

Field	Bits	Typ	Description
<b>EFWxx</b> (xx = 0 to 15)	15:0	rw	Interrupt line INT2 enable for the bit xx of <b>INT_FINT2</b> 0 Disables INT2 for the bit xx 1 Enables INT2 for the bit xx

*Note: The values of the bits of the register **INT\_EINT2** do not affect the set or reset of the corresponding flags in **INT\_FINT2**.*

## INT\_RINT2

Reset Register to INT2 Line of TEAKLite

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RFW 15	RFW 14	RFW 13	RFW 12	RFW 11	RFW 10	RFW 9	RFW 8	RFW 7	RFW 6	RFW 5	RFW 4	RFW 3	RFW 2	RFW 1	RFW 0

Field	Bits	Typ	Description
<b>RFWxx</b> (xx = 0 to 15)	15:0	w	0 No action (do not write 0) 1 Resets bit xx of <b>INT_FINT2</b>

## INT\_SINT2

Set Register to INT2 Line of TEAKLite

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SFW 15	SFW 14	SFW 13	SFW 12	SFW 11	SFW 10	SFW 9	SFW 8	SFW 7	SFW 6	SFW 5	SFW 4	SFW 3	SFW 2	SFW 1	SFW 0

Field	Bits	Typ	Description
<b>SFWxx</b> (xx = 0 to 15)	15:0	w	0 No action (do not write 0) 1 Sets bit xx of <b>INT_FINT2</b>

**CONFIDENTIAL**

**TEAKLite Interrupt Unit**

### TEAKLite Interrupts to MCU

The TEAKLite can generate 4 different interrupts at the MCU; the flag interrupt register **INT\_TOMCU** with 4 bits TOMCU(3:0) is provided.

Writing 1 in **INT\_TOMCU.TOMCUx** generates an interrupt on the corresponding interrupt line TOMCU\_INTx to the MCU. The interrupt must not be cleared by the TEAKLite; the hardware clears the bit by giving an acknowledge signal. The TEAKLite has to be careful because the MCU needs time to process the interrupt request. This has to be taken into consideration when the data flow between the TEAKLite and the MCU is specified.

#### INT\_TOMCU

##### Flag Register for Interrupts to MCU

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												TO MCU 3	TO MCU 2	TO MCU 1	TO MCU 0

Field	Bits	Typ	Description
<b>TOMCUx</b> (X = 0 TO 3)	3:0	w	0 No effect. 1 Generates an interrupt to the MCU on the TOMCU_INTx interrupt line.
<b>RESERVED</b>	15:4	r	Reserved; these bits must be left at their reset values.

*Note: After writing a 1 to TOMCU1/2/3/4, the MCU-Interrupt hardware needs 241 cycles to recognize the interrupt.*

**CONFIDENTIAL**

**Timer**

## 8.2 Timer

History	
Current version: Rev. 1.06, 2005-11-04	
Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.06	

### System Integration on TEAKLite:

- Supply domain: VDD\_DSP
- Chip internal interfaces:
  - Clock domain: gclk\_dsp\_per
  - Bus domain: TEAKLite Z-Bus
  - Interrupt sources: TMR10, TMR11, TMR2
- Monitor Pins: DTMR10\_INT, DTMR11\_INT, and DTMR2

### 8.2.1 Functional Overview

Two general purpose timers are implemented in this hardware peripheral. Each Timer is clocked by the TEAKLite clock and divided internally by a fixed factor. Both timers can be used independently by the TEAKLite.

The first timer is a 12-bit counter and has an internal clock divider by 384. Thus, for an TEAKLite clock of 104 MHz Timer 1 is clocked with 270.83 kHz, which gives a counting length of about 15.12 ms. Timer 1 includes two comparators comparing the counter value in register TMR1\_CNT with the values in the two 12-bit registers [TMR1\\_INT0 \(on Page 356\)](#) and [TMR1\\_INT1 \(on Page 357\)](#). If the counter and the value in TMR1\_INTx are equal, the interrupt flag bit [INT\\_FINT1 \(on Page 347\)](#).TMR1x.

The second timer is a 16-bit wrap around counter and has a clock divider by 96. This means if the TEAKLite is running at 104 MHz Timer 2 is clocked with 1083.33 kHz and has a maximum period of about 60.49 ms. Timer 2 includes one comparator comparing the counter value in register [TMR2\\_CNT \(on Page 359\)](#) with the value in the 16-bit register [TMR2\\_MAX \(on Page 359\)](#). If both values are equal the interrupt flag bit [INT\\_FINT1](#).TMR2 is set.

#### 8.2.2 DSP Timer 1

Timer 1 is externally clocked with gclk\_dsp\_per, which is divided by 384 internally. The compare values can be written to the registers [TMR1\\_INT0 \(on Page 356\)](#) and

CONFIDENTIAL

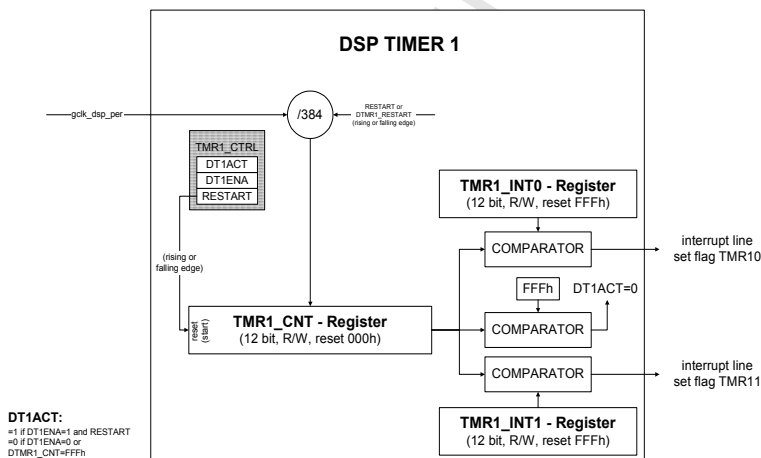
Timer

**TMR1\_INT1 (on Page 357)** independent of the value of bit **TMR1\_CTRL (on Page 355).DT1ENA**. Timer 1 is enabled by setting bit **TMR1\_CTRL.DT1ENA**. The `gclk_dsp_per` clock is enabled automatically. After setting **TMR1\_CTRL.DT1ENA** the timer is in the standby mode. When the timer has been enabled a 1 can be written to bit **TMR1\_CTRL.RESTART**. This restarts the timer from value  $000_H$  and sets bit **TMR1\_CTRL.DT1ACT**. Timer 1 can be stopped by the TEAKLite by resetting the bit **TMR1\_CTRL.DT1ENA**. Then the clock `gclk_dsp_per` is switched off automatically. If the timer reaches the value  $FFF_H$  the hardware clears the bit **TMR1\_CTRL.DT1ACT** and the counter is stopped. The external clock `gclk_dsp_per` is not switched off and the timer goes into the stand-by mode and waits for the next write of a 1 to **TMR1\_CTRL.RESTART**.

*Note: TEAKLite Timer 1 starts with a synchronization delay of one (`gclk_dsp_per/8`)-cycle and an additional delay of two (`gclk_dsp_per/8`)-cycles caused by the edge detection.*

*Note: A timer restart also resets the clock divider and the counter.*

**Figure 8-3 Simplified Functional Block Diagram of Timer 1**



**CONFIDENTIAL**

**Timer**

### 8.2.2.1 DSP Timer 1 Registers

To obtain the addresses of these registers refer to [Section 12.1.2 Registers in the DSP Memory Space \(on Page 1266\)](#).

#### TMR1\_CTRL

##### DSP Timer 1 Control Register

[Reset value: 0000<sub>H</sub>]

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED													DT1 ACT	RE STA RT	DT1 ENA

Field	Bits	Type	Description
<b>DT1ENA</b>	0	rw	Set and reset by TEAKLite 0 The clock is switched off. 1 The clock is switched on. Timer is in standby mode until TEAKLite writes a 1 to bit <b>RESTART</b> .
<b>RESTART</b>	1	rw	0 No effect 1 Toggles the value of the bit and starts Timer 1 if <b>DT1ENA</b> is set. <i>Note: This bit is always read as 0.</i>
<b>DT1ACT</b>	2	rh	Set and reset by hardware. 0 Timer 1 is stopped. 1 Timer 1 is running. Set by hardware if <b>DT1ENA</b> was set and firmware writes a 1 to bit <b>RESTART</b> . Reset by hardware if <b>TMR1_CNT</b> reaches FFF <sub>H</sub> or <b>DT1ENA</b> is reset. Writing by the TEAKLite has no effect.
<b>RESERVED</b>	15:3	r	Reserved; these bits must be left at their reset values.

*Note: Writing to this register is possible if timer clock is switched off.*

**CONFIDENTIAL**

**Timer**

### TMR1\_CNT

#### DSP Timer 1 Counter

[Reset value: 0000<sub>H</sub>]

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				T1CNT											

Field	Bits	Type	Description
T1CNT	11:0	r	<b>Current Counter Value</b> If the counter is stopped, it is set automatically to 000 <sub>H</sub> by starting Timer 1.
RESERVED	15:12	r	Reserved; these bits must be left at their reset values.

### TMR1\_INT0

#### DSP Timer 1 Interrupt Value 0

[Reset value: 0FFF<sub>H</sub>]

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				T1INT0											

Field	Bits	Type	Description
T1INT0	11:0	rw	If the counter value reaches this value the interrupt flag bit <b>INT_FINT1.TMR10</b> is set and counting is continued.
RESERVED	15:12	r	Reserved; these bits must be left at their reset values.

*Note: Read/write access is only allowed if DSP Timer 1 is not active  
(**TMR1\_CTRL.DT1ACT** has to be 0).*



CONFIDENTIAL

Timer

## TMR1\_INT1

### DSP Timer 1 Interrupt Value 1

[Reset value: 0FFF<sub>H</sub>]

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				T1INT1											

Field	Bits	Type	Description
T1INT1	11:0	rw	If the counter value reaches this value the interrupt flag bit <b>INT_FINT1</b> (on Page 347).TMR11 is set and counting is continued.
RESERVED	15:12	r	Reserved; these bits must be left at their reset values.

Note: Read/write access is only allowed if DSP Timer 1 is not active  
(**TMR1\_CTRL.DT1ACT** must be 0)

## 8.2.3 DSP Timer 2

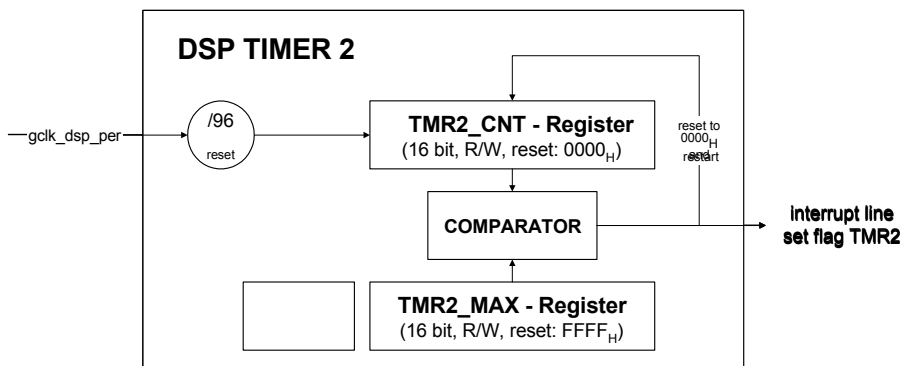
Timer 2 is externally clocked with `gclk_dsp_per` which is divided by 96 internally. Before the Timer is activated the start value must be written to register **TMR2\_CNT** and the interrupt (= wrap around) value must be set in register **TMR2\_MAX**. By setting bit **TMR2\_CTRL.DT2ACT**, the 96-divider is reset, Timer 2 is enabled and started. The read access to both registers is possible all the time. The timer is started at the value of register **TMR2\_CNT**. If the counter reaches the value of register **TMR2\_MAX** the interrupt flag **INT\_FINT1.TMR2** is set, the counter is reset (**TMR2\_CNT** = 0000<sub>H</sub>) in the next (`gclk_dsp_per/96`)-cycle and started again. Timer 2 can be stopped by the TEAKLite by clearing bit **TMR2\_CTRL.DT2ACT**. Then the clock of the timer is switched off.

Note: Timer 2 starts with a synchronization delay of one (`gclk_dsp_per/96`)-cycle.

Note: Example: **TMR2\_CNT** = 0010<sub>H</sub>, **TMR2\_MAX** = 00FF<sub>H</sub>. The counter is started with the value 0010<sub>H</sub> by setting bit **TMR2\_CTRL.DT2ACT**. The counter reaches 00FF<sub>H</sub>. The interrupt flag bit **TMR2** is set. During the next clock cycle the counter contains the value 0000<sub>H</sub> and after one more clock cycle 0001<sub>H</sub>.

Note: If the value **TMR2\_CTRL** is greater than **TMR2\_MAX** when starting the counter value wraps around from FFFF<sub>H</sub> to 0000<sub>H</sub> and no interrupt flag is set. The counter continues and the interrupt flag is set if the value reaches **TMR2\_MAX** for the first time.

Figure 8-4 Simplified Functional Block Diagram of Timer 2



### 8.2.3.1 DSP Timer 2 Registers

To obtain the addresses of these registers refer to [Section 12.1.2 Registers in the DSP Memory Space \(on Page 1266\)](#).

#### TMR2\_CTRL

DSP Timer 2 Control Register

[Reset value: 0000<sub>H</sub>]

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															DT2ACT

Field	Bits	Type	Description
DT2ACT	0	rw	1 Timer 2 is running, the clock is switched on. 0 Timer 2 is stopped, the clock is switched off.
RESERVED	15:1	r	Reserved; these bits must be left at their reset values.

*Note: Writing to this register must be possible also if timer clock is switched off*

**CONFIDENTIAL**

**Timer**

## **TMR2\_CNT**

**DSP Timer 2 Counter**

**[Reset value: 0000<sub>H</sub>]**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T2CNT															

Field	Bits	Type	Description
<b>T2CNT</b>	15:0	rw	Can be used to set the start value for Timer 2. If <b>TMR2_CTRL.DT2ACT</b> = 1 writing is not allowed. During <b>TMR2_CTRL.DT2ACT</b> = 1 the counter contains the current counter value. After stopping the DSP Timer 2 by <b>TMR2_CTRL.DT2ACT</b> = 0, the <b>TMR2_CNT</b> holds the last value of the counter.

## **TMR2\_MAX**

**DSP Timer 2 Interrupt value**

**[Reset value: FFFF<sub>H</sub>]**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T2MAX															

Field	Bits	Type	Description
<b>T2MAX</b>	15:0	rw	<b>TMR2_MAX</b> contains the interrupt and wrap around value. If the counter reaches this value the interrupt flag bit <b>INT_FINT1.TMR2</b> is set and the counter continues at 0000 <sub>H</sub> . Can only be written if <b>TMR2_CTRL.DT2ACT</b> = 0. If <b>TMR2_CTRL.DT2ACT</b> = 1, writing is not allowed.

CONFIDENTIAL

**CONFIDENTIAL**

**Viterbi Coprocessor, Channel Decoder**

## 8.3 Viterbi Coprocessor, Channel Decoder

History	
Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.04	
<b>Page 361</b>	Heading "Channel Decoder Accelerator" changed to <b>Viterbi Coprocessor, Channel Decoder</b> WS00007944
Changes for Rev. 1.06	

### System Integration on TEAKLite:

- Supply domain: VDD\_DSP
- Chip internal interfaces:
  - Clock domain: gclk\_dsp\_per
  - Bus domain: TEAKLite Z-Bus
- Interrupt sources: CHADEC
- Monitor pins:

### 8.3.1 Interface

#### 8.3.1.1 TEAKLite Interface

The interface to the TEAKLite consists of a register bank with status and configuration registers and of one "user defined core register" (rd/wr, ext0). By means of this register, the TEAKLite can access the internal RAM2 of the hardware-peripheral and can initialize the two working RAMs: RAMW1 and RAMW2.

The entries of the configuration register **VH\_CONF1\_DEC (on Page 364)** determines which RAM area of RAM2 and of RAMW1, RAMW2 is addressed if ext0 is accessed. The access to the internal RAMs is performed in a FIFO-like manner starting with a base address and incrementing the current address every time a read or write access to ext0 is detected. By this means the TEAKLite can write the RAM areas dedicated for the soft-input values for the decoder and so on. Bits in the configuration register **VH\_CONF1\_DEC** are provided to reset the current address of the special RAM area to the base address. After finishing writing the dedicated RAM area, the internal data paths have to be enabled and get access to these areas for internal arithmetical calculations.

**CONFIDENTIAL**

**Viterbi Coprocessor, Channel Decoder**

**Figure 8-5** shows the TEAKLite interface and the HW-internal registers and memories, **Figure 8-8** the partitioning of the HW-internal memory RAM2 and **Figure 8-9** the partitioning of the HW-internal working memories RAMW1 and RAMW2.

**Table 8-2 Interface Description**

Register or Memory	Address	Description
Configuration and status registers	Situated within the data RAM address area.	The entries of the configuration registers select the RAM area which shall be overwritten or read. The decoding process can be enabled. The response of the status register confirms the end of the chosen process. All registers are accessible with 2 wait states.
RAM2 RAMW1, RAMW2	Access via ext0 (user defined core register)	RAM2, RAMW1, and RAMW2 access: the addresses within RAM2, RAMW1, and RAMW2 are determined by the entries of the configuration register <b>VH_CONF1_DEC</b> . The ext0 register can be accessed without wait states and is accessed only within the decoding procedure.

### 8.3.1.2 Register Description

**Table 8-3 Register Description (Configuration and Status Registers)**

Register Name	Access	Comment
<b>VH_CONF1_DEC</b>	WR	Configuration Register 1 for Decoder.
<b>VH_CONF2_DEC</b>	RD/WR	Configuration Register 2 for Decoder.
<b>VH_STATUS_DEC</b>	RD	Status register.
<b>VH_CONF_CNT_D</b>	RD/WR	Count configuration, Number of Bits to be Decoded.
<b>VH_STAT_CNT_D</b>	RD	Count status, Number of Bits Decoded.
<b>VH_REF_BR_BFLYx</b>	RD/WR	Reference table: relation between branch metric and butterfly (Channel Decoding)

#### Example of Access to Configuration Registers and Internal RAMs

Before the RAM2, RAMW1 or RAMW2 can be accessed the base address of the targeted RAM area must be defined in the configuration register **VH\_CONF1\_DEC** (on [Page 364](#)).

The access to the RAM2, RAMW1, and RAMW2 is performed by reading from or writing to the TEAKLite address ext0.

CONFIDENTIAL

Viterbi Coprocessor, Channel Decoder

**Example:** For writing input values for the channel decoder to the RAM2 area the following procedure has to be performed:

- **VH\_CONF2\_DEC (on Page 365)** is set to 0001<sub>H</sub> (bit **HW\_ENA\_DEC** = 1) and **VH\_CONF1\_DEC** is set to 0002<sub>H</sub> (bit **RES\_SIN01\_BASE** = 1, reset RAM2 write pointer).
- Every input value written to ext0 is stored within the selected region. The address starts at the base address and is successively incremented after each write access.

*Note: Only one of the bits RES\_...\_BASE may be set. If more than 1 of the bits is set the resulting procedure will be erroneous. The firmware has to take care of this. After resetting the pointers by hardware the bits of **VH\_CONF1\_DEC** are reset automatically by hardware after 4 cycles.*

*Note: The access to ext0 and, therefore, to the RAMs RAM2, RAMW1, and RAMW2 is performed without wait states.*

### TEAKLite Restriction

The following TEAKLite instructions access ext0:

```
mov acc,ext0 and mov (r0),
ext0 (mov ext0,acc and mov ext0,(r0))
```

where r0 is a pointer to the on-core RAM of the TEAKLite.

**Note: If one specific RAM is reset to a chosen base address for a read operation the firmware must wait 4 cycles after writing the base address before accessing ext0. If the read access from the TEAKLite to ext0 directly follows the RAM reset this read value can be delivered without a wait state to the TEAKLite. For writing no wait cycles are required.**

**CONFIDENTIAL**

**Viterbi Coprocessor, Channel Decoder**

### 8.3.1.2.1 Viterbi Decoder Configuration Registers

#### VH\_CONF1\_DEC

**Configuration Register 1 for Viterbi Decoder**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED							RES_DM BASE	RESERVED			RES TR BASE	RES SIN2 BASE	RES SIN0 1 BASE	RES RW1 RW2	

Field	Bits	Type	Description
RES_RW1_RW2	0	w	Select between equivalent memory areas. 0    RAMW1 1    RAMW2
RES_SIN01_BASE	1	w	Reset RAM2 pointer to SIN01_BASE, reset RAM32_RD_CNT_DEC and RAM32_WR_CNT_DEC flags.
RES_SIN2_BASE	2	w	Reset RAM2 pointer to SIN2_BASE, reset RAM32_RD_CNT_DEC and RAM32_WR_CNT_DEC flags.
RES_TR_BASE	3	w	Reset RAM2 pointer to TR_BASE, reset RAM32_RD_CNT_DEC and RAM32_WR_CNT_DEC flags.
RES_DM_BASE	7	w	Reset RAMW1/W2 pointers to DM_BASE, reset RAM32_RD_CNT_DEC and RAM32_WR_CNT_DEC flags.
RESERVED	6:4, 15:8	r	Reserved; these bits must be left at their reset values.



**CONFIDENTIAL**

**Viterbi Coprocessor, Channel Decoder**

## VH\_CONF2\_DEC

**Configuration Register 2 for Viterbi Decoder**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEC 64	OFL OW PRO T	PC DEC 1	PC DEC 0	DEC FLA G RD	RESERV ED	RES ACL	RESERVED						DEC ON	RES DEC	HW ENA DEC

Field	Bits	Type	Description
<b>HW_ENA_DEC</b>	0	rw	<b>Enable Clock for Hardware Decoding</b> 0 Switch off 1 Switch on <i>Note: Only if <b>HW_ENA_DEC</b> is set the decoder related registers can be written.</i>
<b>RES_DEC</b>	1	rw	<b>Decoder Reset</b> 0 No effect 1 Reset Viterbi decoding Reset current timestamp and butterfly counter to zero, reset internal decoding flags to default value, reset RAM32_WR_CNT_DEC and RAM32_RD_CNT_DEC flags, reset <b>VH_STATUS_DEC.DEC_BUSY</b> , <b>DEC_64</b> bit, <b>DEC_ON</b> bit, <b>PC_DEC_1</b> bit, <b>PC_DEC_0</b> bit, <b>DEC_FLAG_RD</b> bit, <b>OFLOW_PROT</b> bit, <b>VH_STAT_CNT_D</b> , <b>VH_CONF_CNT_D</b> . <i>Note: <b>RES_DEC</b> does not reset <b>VH_REF_BR_BFLYx</b>.</i>
<b>DEC_ON</b>	2	rwh	<b>Start Viterbi Decoding</b> 0 No effect 1 Viterbi-Decoding on This bit is reset by hardware after <b>VH_STATUS_DEC.DEC_BUSY</b> is set in status register.
<b>RES_ALL</b>	8	rw	Reset all internal registers. Bit is reset within 4 cycles.
<b>DEC_FLAG_RD</b>	11	rw	<b>Direction Flag for Access to ext0</b> 0 Write access to ext0 1 Read access to ext0

**CONFIDENTIAL**

**Viterbi Coprocessor, Channel Decoder**

Field	Bits	Type	Description
<b>PC_DEC_0</b>	12	rw	<b>Packing Mode 0, for RAMW1 and RAMW2</b>
			<p>0 <b>Write access:</b> Incoming 16-bit words are stored at incrementing RAM addresses, the 16 MSB of each RAM cell are set to zero. <b>Read access:</b> The transfer refers to adjacent RAM-addresses.</p> <p>1 <b>Write access:</b> 2 incoming 16-bit words are packed into a 32 bit word. <b>Read access:</b> The two 16-bit words of the 32-bit word are transferred subsequently.</p>
<b>PC_DEC_1</b>	13	rw	<b>Packing Mode 1, for RAM2</b>
			<p>0 <b>Write access:</b> Incoming 16-bit words are stored at incrementing RAM addresses. <b>Read access:</b> The transfer refers to adjacent RAM addresses.</p> <p>1 <b>Write access:</b> 2 incoming 8-bit words are packed into a 16-bit word. <b>Read access:</b> The two 8-bit words of the 16-bit word are transferred subsequently, the 8 MSB of ext0 are sign-extended by bit 7 of the 8-bit word to be transferred.</p>
<b>OFLOW_PROT</b>	14	rw	<p><b>Overflow Protection for Metric Values</b></p> <p>0 Disabled</p> <p>1 Enabled. If one state metric exceeds the value of 16384 a value of 4096 will be subtracted from each state metric during the second following timestamp.</p>
<b>DEC_64</b>	15	rw	<p><b>Select Viterbi-Decoding</b></p> <p>0 16 states</p> <p>1 64 states</p>
<b>RESERVED</b>	7:3, 10:9	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**Viterbi Coprocessor, Channel Decoder**

### 8.3.1.2.2 Viterbi Hardware Decoding Status

**VH\_STATUS\_DEC**

**Viterbi Hardware Decoding Status Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RES ERV ED</b>	<b>DEC BUSY</b>	<b>RESERVED</b>													

Field	Bits	Type	Description
<b>DEC_BUSY</b>	14	rh	<b>Viterbi Decoding in Progress.</b> Set by the hardware peripheral, reset when the number of timestamps set in <b>VH_CONF_CNT_D</b> has been processed. The falling edge of <b>DEC_BUSY</b> creates an interrupt at the TEAKLite by setting bit <b>INT_FINTA0 (on Page 341)</b> . <b>DEC_BUSY</b> can also be reset by <b>VH_CONF2_DEC.RES_DEC</b> or <b>VH_CONF2_DEC.RES_ALL</b> .
<b>RESERVED</b>	13:0, 15	r	Reserved; these bits must be left at their reset values.

### 8.3.1.2.3 Decoding Bit Count Configuration

**VH\_CONF\_CNT\_D**

**Decoding Bit Count Configuration Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>								<b>C_DEC</b>							

Field	Bits	Type	Description
<b>C_DEC</b>	7:0	rw	Number of bits to be decoded.
<b>RESERVED</b>	15:8	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**Viterbi Coprocessor, Channel Decoder**

### 8.3.1.2.4 Decoding Bit Count Status

**VH\_STAT\_CNT\_D**

**Decoding Bit Count Status Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								S_DEC							

Field	Bits	Type	Description
<b>S_DEC</b>	7:0	rh	Number of bits already decoded. Set by hardware, reset by <a href="#">VH_CONF2_DEC.RES_DEC</a> or <a href="#">VH_CONF2_DEC.RES_ALL</a> .
<b>RESERVED</b>	15:8	r	Reserved; these bits must be left at their reset values.

### 8.3.1.2.5 Branch Metric Address for Butterfly x

**VH\_REF\_BR\_BFLYx**

x = 0...7

**Branch Metric Address for Butterfly x**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REF_BR_BFLYx_4_y				REF_BR_BFLYx_3_y				REF_BR_BFLYx_2_y				REF_BR_BFLYx_1_y			

Field	Bits	Type	Description
<b>REF_BR_BFLYx_1_y</b>	3:0	rw	The data path of the HW accelerator addresses the separate 4 bit combinations to access the calculated branch metrics.
<b>REF_BR_BFLYx_2_y</b>	7:4	rw	
<b>REF_BR_BFLYx_3_y</b>	11:8	rw	
<b>REF_BR_BFLYx_4_y</b>	15:12	rw	

CONFIDENTIAL

Viterbi Coprocessor, Channel Decoder

Figure 8-5 TEAKLite Interface and HW Internal Memory

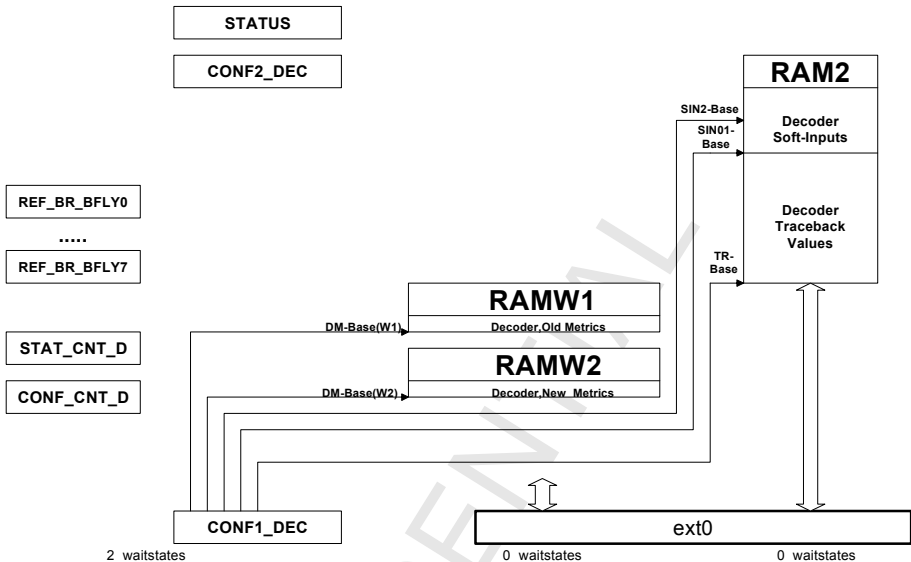
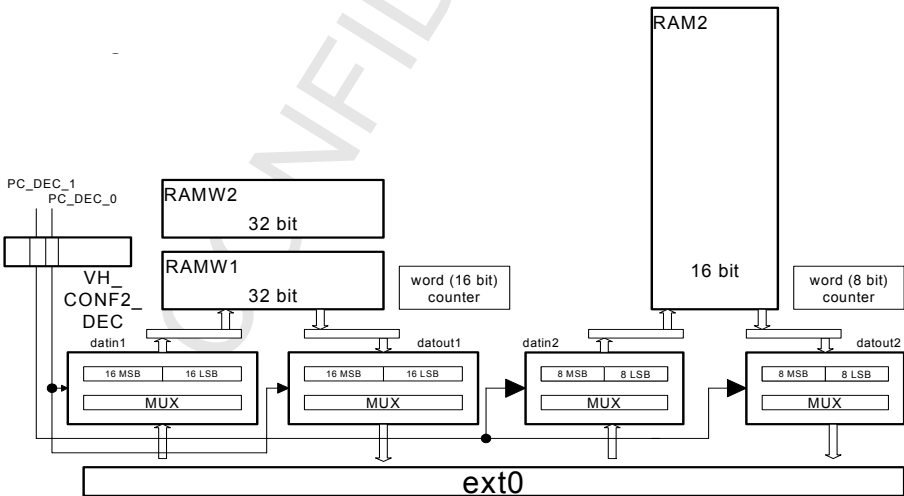


Figure 8-6 Packing Modes



RAMW1 and RAMW2 have a word width of 32 bits, while RAM2 is only 16 bits wide.

The **VH\_CONF2\_DEC.PC\_DEC\_0** and **VH\_CONF2\_DEC.PC\_DEC\_1** bits control the access mode to RAMW1 (**PC\_DEC\_0**), RAMW2 (**PC\_DEC\_0**), and RAM2

**CONFIDENTIAL**

**Viterbi Coprocessor, Channel Decoder**

(**PC\_DEC\_1**). The different modes are required to fit different word formats for input and output values to the available RAM format (see [Figure 8-6](#)).

All formats referring to RAMW1 and RAMW2 are organized in 16 bits, while formats for RAM2 refer to 16 bits and 8 bits (soft-inputs). 8-bit soft-inputs have to be adapted to the 16 bits of RAM2.

The circuitry for setting the different packing modes has to be implemented in the “user defined core register” ext0.

#### **RAMW1, RAMW2: [VH\\_CONF2\\_DEC.PC\\_DEC\\_0](#)**

- **PC\_DEC\_0 = 1:**

As the TEAKLite transfers 16-bit words the adaptation to 32-bit word width has to be performed within the HW peripheral.

The incoming data from TEAKLite are stored temporarily and is combined with the following 16-bit word before the entire 32-bit word is written to RAMW1 or RAMW2. The TEAKLite transfers the least significant 16-bit word first.

In the opposite direction from hardware to TEAKLite the least significant 16-bit word is transferred first, followed by the most significant 16-bit word. The second 16-bit word is temporarily stored within the hardware peripheral until the read access from the TEAKLite is performed.

For the transfer of both 16-bit words from one 32-bit word in RAMW1 and RAMW2 the internal flags RAM32\_WR\_CNT\_DEC and RAM32\_RD\_CNT\_DEC are reset if the bit [VH\\_CONF1\\_DEC.RES\\_DM\\_BASE](#) is set to ensure proper initial conditions.

- **PC\_DEC\_0 = 0:**

In this mode 16-bit words are transferred directly to or from subsequent RAM addresses without introducing any packing mechanism.

**CONFIDENTIAL**

**Viterbi Coprocessor, Channel Decoder**

## RAM2: **VH\_CONF2\_DEC.PC\_DEC\_1**

The values that have to be transferred between TEAKLite and the HW peripheral are organized in 16- or 8-bit words.

- **PC\_DEC\_1 = 1:**

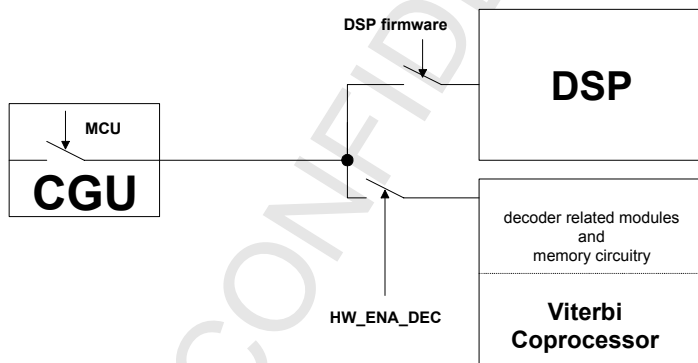
From each incoming 16-bit word the 8 LSBs are combined in a 16-bit word before storing the value to RAM2. At a read access a 16-bit word is decomposed into two words of 8 bits and transferred to the TEAKLite at two successive ext0 accesses. In both cases the least significant 8-bit word is transferred first, followed by the most significant 8-bit word.

As for RAMW1 and RAMW2, there are also two internal flags (RAM16\_RD\_FLAG\_DEC, RAM16\_WR\_FLAG\_DEC) for RAM2, which select the 8-bit sub-words of a 16-bit word currently being transferred. Both flags are reset if one of the bits **VH\_CONF1\_DEC.RES\_TR\_BASE**, **VH\_CONF1\_DEC.RES\_SIN01\_BASE** or **VH\_CONF1\_DEC.RES\_SIN2\_BASE** is set.

- **PC\_DEC\_1 = 0:**

16-bit words are transferred between TEAKLite and hardware without any packing procedure. 16-bit values are written to or read from successive addresses.

**Figure 8-7 Clocking Scheme**

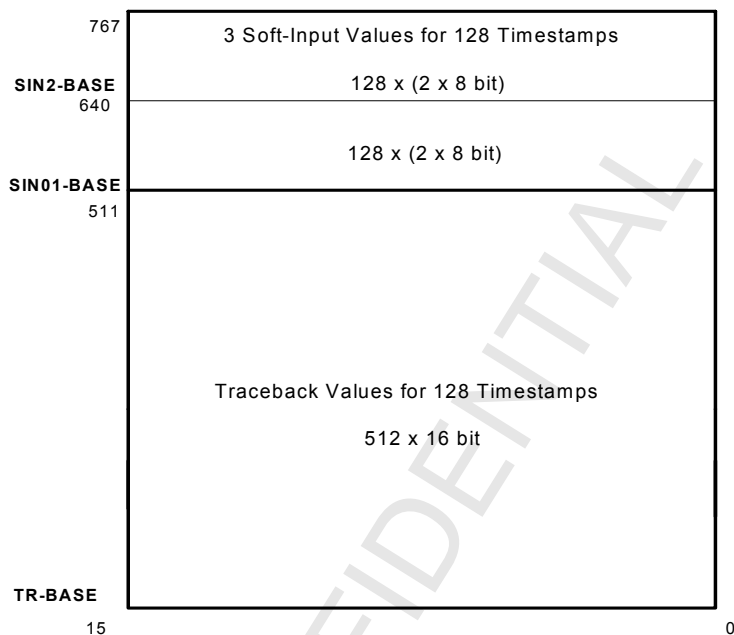


The register interface to the DSP is supplied by the DSP clock, HW\_ENA\_DEC, switches on the internal HW and memory circuitry. The MCU switches on all the HW circuitry. A status bit informs the MCU about the Hardware Accelerator clock.

**CONFIDENTIAL**

**Viterbi Coprocessor, Channel Decoder**

**Figure 8-8 RAM2 - Memory Partitioning**  
**GSM Channel Decoding**





**CONFIDENTIAL**

**Viterbi Coprocessor, Channel Decoder**

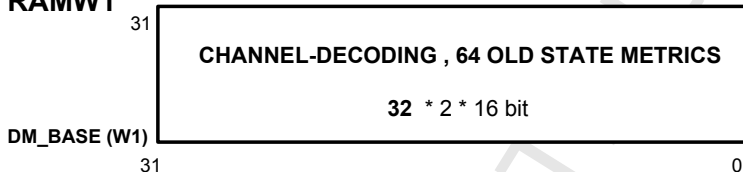
**Figure 8-9 RAMW1/RAMW2 - Memory Partitioning**

## Memory Mapping: RAMW1 / RAMW2

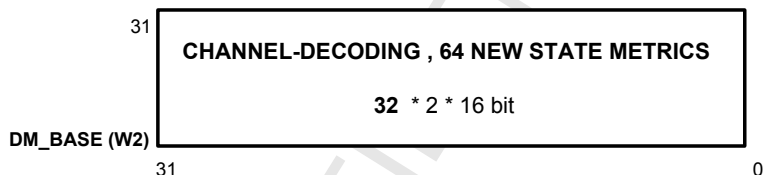
### GSM Channel Decoding

#### WORKING RAMs

#### RAMW1



#### RAMW2



CONFIDENTIAL

**CONFIDENTIAL**

**Bi-directional Serial Audio Interface I2S1 (4-pins)**

## 8.4 Bi-directional Serial Audio Interface I2S1 (4-pins)

History	
Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Change for Rev 0.02	
Page 392	Update <a href="#">Section 8.4.1 Functional Overview (on Page 376)</a> WS00005907
Changes for Rev. 1.02	
<a href="#">Page 378</a>	Updated <a href="#">Section 8.4.3 Operating Mode Generalities</a> WS00005886
Changes for Rev. 1.04	
<a href="#">Page 375</a>	Heading "I2S 1 (DAI)" changed to <a href="#">Bi-directional Serial Audio Interface I2S1 (4-pins)</a> WS00007943
Changes for Rev. 1.06	
<a href="#">Page 405</a>	Table cross-reference corrected in <a href="#">I2Sx_CSEL</a>

### System Integration on TEAKLite:

- Supply domain: VDD\_DSP
- Chip internal interfaces:
  - Clock domain:
    - Peripheral kernel: gclk\_dsp\_per
    - Clocks for baud rate generation: gclk\_pll, gclk\_dsp\_per
  - Bus domain: TEAKLite Z-Bus
  - Interrupt sources:
    - I2S1\_TX\_INT, I2S1\_RX\_INT
- Chip external signals related to this block (refer to [Chapter 3 Pin Descriptions \(on Page 49\)](#) for pin configuration options):
  - I2S1\_CLK0, I2S1\_WA0, I2S1\_TX, I2S1\_RX
- Monitor Pins:

**CONFIDENTIAL****Bi-directional Serial Audio Interface I2S1 (4-pins)****8.4.1 Functional Overview**

The I<sup>2</sup>S 1 (DAI) and I<sup>2</sup>S 2 (Audio), I2Sx, x = 1 or 2, interfaces are digital audio interfaces for external high-resolution digital-to-analog converters or external audio sources that provide digital data output.

**General Features of the I2Sx Interfaces**

- 3 Supported modes: Normal mode, PCM mode and DAI mode
- 32 bit data path
- Interface can operate in master and slave mode
- Two independent 64 word data buffers for receiver and transmitter
- Bi-directional Transmission with independent receiver and transmitter
- Two independent fractional dividers for bit clock generation for reception and transmission in master mode
- Adjustable audio sampling rate (for example, 48 kHz, 44.1 kHz, 32 kHz, 24 kHz, 22.05 kHz, 16 kHz, 12 kHz, 11.025 kHz and 8 kHz).

**Mode Dependant Features of the I2Sx Interfaces**

- Normal mode:
  - Supports delayed and non-delayed WA mode
  - Configurable to different word lengths (16, 18, 20, 24 and 32 bit) and different frame lengths (64, 48 and 32 bit)
  - Receiver auto frame length detection in master mode
  - Switch of left/right order in frame and left/right alignment of data in frame
  - Data valid on rising or falling edge of bit clock
- PCM mode:
  - Supports 16 bit mode with short frame synchronization
  - Supports both burst mode and continuous transmission
  - Length of WA pulse selectable between one and two clock cycles
  - CLK line is configurable to be either low, high or to remain running when no transmission takes place
- DAI mode:
  - 13 bit linear PCM data format with 8000 samples per second (104 kbit/s)

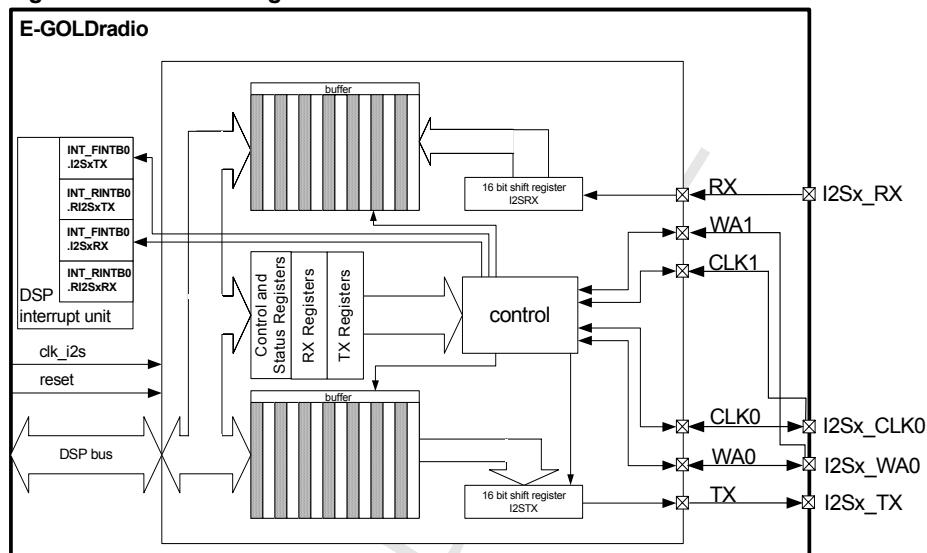
**CONFIDENTIAL**

**Bi-directional Serial Audio Interface I2S1 (4-pins)**

## 8.4.2 Structural Overview

A block diagram of the bi-directional I2Sx 4-pin interface is shown in **Figure 8-10**.

**Figure 8-10 Block Diagram of Bi-Directional I2Sx 4-Pin Interface**



The bi-directional I2Sx modules have the following signals:

**Table 8-4 I2Sx Signals**

	Signal Name	Function
<b>TX I2S</b>	i2sx_clk0	Transmit clock
	i2sx_wa0	Transmit word select
	i2sx_tx	Serial output audio data
<b>RX I2S</b>	i2sx_clk1 <sup>1)</sup>	Receive clock
	i2sx_wa1 <sup>1)</sup>	Receive word select
	i2sx_rx	Serial input audio data

<sup>1)</sup> This signal is not available outside the chip when in the 4-pin configuration.

In the E-GOLDradio the number of required interface wires is reduced because a common clock source for transmit and receive is used (either internal or external). The word alignment and clock signals always have the same direction. Therefore, the word alignment signals i2sx\_wa0 and i2sx\_wa1 are combined and the clock signals are

**CONFIDENTIAL****Bi-directional Serial Audio Interface I2S1 (4-pins)**

combined. As a result, the number of required signals (and also the number of required pins) is reduced from six to four.

*Note: For the I2S2 interface the signals I2S2\_CLK1 and I2S2\_WA1 are not available outside the chip. For this reason I2S 2 can only be operated with combined clock and word align signals.*

### 8.4.3 Operating Mode Generalities

Regardless of the chosen operating mode the following considerations apply to the I2S interface.

To set up a data transfer using the serial interface, first set **I2Sx\_CTRL.I2SON**. This will reset the internal state of the interface. In particular the read and write pointer, internal state machines and counters are reset. The control registers are not affected when **I2SON** is set.

After this all necessary configurations like mode setting, clock source selection, etc. have to be made. If a transmission is intended the first set of audio samples must then be written to the interface buffer. Finally, by setting the start bit **I2Sx\_CTRL.I2SxTXSTART** or bit **I2Sx\_CTRL.I2SxRXSTART** the data transfer (receive or transmit) can be started.

The transfer can stop at two different conditions:

- Automatic stop when interrupt address is reached (PCM mode)
- Start bit **I2SxTXSTART** or **I2SxRXSTART**, respectively, is reset by software.

For the receive path, the automatic stop in the PCM mode becomes effective when the interrupt address defined in **I2Sx\_RXINTADDR** is reached. At the same time an I2SxRX interrupt is generated. This means that the last sample received is stored at the address before the receive interrupt address.

For the transmit path, the I2SxTX interrupt is also generated when the interrupt address defined in **I2Sx\_TXINTADDR** is reached. The last sample transferred is stored at the address just before the transmit interrupt address. An I2SxTX interrupt is generated when the transfer of the last sample begins.

For stopping the interface by software the bit **I2SxTXSTART** or **I2SxRXSTART** can be reset. This will end transmission or reception as soon as the transfer of the current frame has been finished.

In order to change the configuration settings the following sequence is required:

- Stop interface either automatically or by resetting the start bit
- Switch bit **I2SON** to 0 to reset internal states
- Set bit **I2SON** to 1 again
- Change configuration
- Restart data transfer by setting the bit **I2SxTXSTART** or bit **I2SxRXSTART**.

CONFIDENTIAL

Bi-directional Serial Audio Interface I2S1 (4-pins)

After restarting data transmission by setting bit **I2SxTXSTART**, the **I2Sx\_RWADDR.RDADDR** pointer is incremented. This means that the next audio sample to be sent is taken from the buffer position following the one of the last transmitted audio sample. The first audio sample transmitted belongs to the left channel.

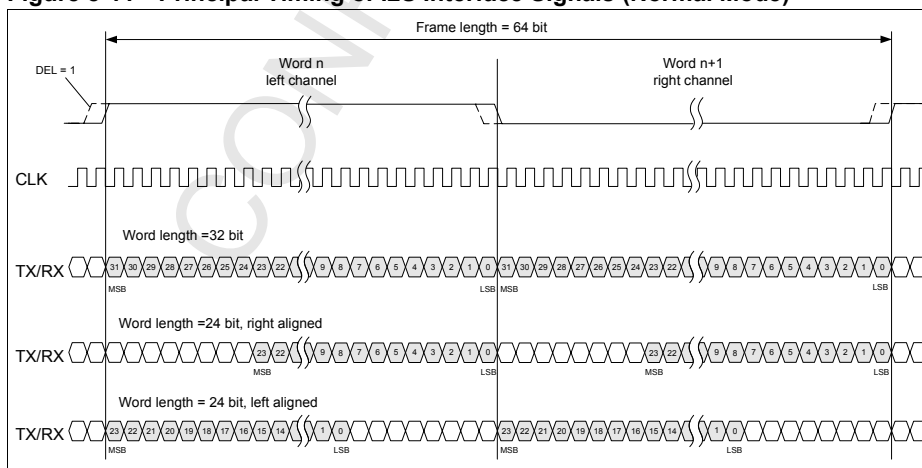
After restarting data reception by setting bit **I2SxRXSTART**, the **I2Sx\_RWADDR.WRADDR** pointer is incremented. This means that the next audio sample is stored at the buffer position following the one of the last received audio sample.

The I2S interface operating as master supports all data rates which can be derived by dividing the reference frequency by a fractional divider (for example, 48 kHz, 44.1 kHz, 32 kHz, 24 kHz, 22.05 kHz, 16 kHz, 12 kHz, 11.025 kHz, and 8 kHz are possible when a 104 MHz reference frequency is supplied. Refer to [Table 8-8](#) and the following tables for an overview).

### 8.4.4 Serial Audio Data Transmission in Normal Mode

**Figure 8-11** gives the basic timing of the I2S signals in normal mode. It shows combinations of a 64-bit frame with 32-bit words, 24-bit words right aligned and 24-bit words left aligned as examples. Transmission in normal mode is supported in master as well as in slave operation. Serial audio data is transmitted with the most significant bit first. In this example data is launched at the transmitting side with the falling edge of **i2sx\_clk0** and latched at the receiving side with the rising edge of **i2sx\_clk0**. Signal **i2sx\_wa0** is used to select between left and right channel and to determine the start of word.

**Figure 8-11 Principal Timing of I2S Interface Signals (Normal Mode)**



**CONFIDENTIAL**

**Bi-directional Serial Audio Interface I2S1 (4-pins)**

For use with low-cost digital-to-analog converters that have a simplified I2S interface, **I2Sx\_TXCONF.DEL** and **I2Sx\_RXCONF.DEL** can be set. This results in a delay of the signal bits by one bit clock cycle with respect to **i2sx\_wa0** (see **Figure 8-11**).

### 8.4.5 PCM Mode Operation of I2S interface

The I2S interface can also be set up in a mode that allows PCM data transfers from and to other devices (for example, PMB 6752 Bluemoon® I). In this mode short frame synchronization with a 16-bit frame length is supported. Both burst and continuous transfers are possible. PCM mode supports master and slave operation. The PCM mode transmission or reception is selected by setting the bit **I2Sx\_CTRL.TXPCM** or **I2Sx\_CTRL.RXPCM**. The main difference between this mode and the normal I2S mode is that the signal **i2sx\_way** now serves as a frame synchronization signal and that the data transfer does not need to be continuous. A timing diagram which is valid for both data transmission and data reception is shown in **Figure 8-12**. The rising edge of **i2sx\_clk** is always used to shift out the next transmit data bit and to shift in a captured receive data bit. The receive data bit itself is captured with the falling edge of **i2sx\_clk**.

The data transmit operation works as follows: At each falling edge of **i2sx\_clk** the value on the **I2Sx\_WA0** line is checked. After a high level on **i2sx\_wa0** has been detected the MSB of the transmit data is put on the TX line with the next rising edge of **i2sx\_clk**. Now the interface waits until a low level on **i2sx\_wa0** is detected during the falling edge of **i2sx\_clk**. If this condition has been found the interface shifts out the remaining data bits, one bit at each rising edge of **i2sx\_clk**. As soon as all 16 bits have been transmitted a new data sample is transferred immediately from the TX buffer to the shift register and **I2Sx\_RWADDR.RDADDR** is incremented by one. Thus, a continuous data transfer can be achieved.

The data receive operation works in nearly the same way. After a high level on the **i2sx\_wa1** line was detected at the falling edge of **i2sx\_clk** the interface begins to capture the value on the RX line at each falling edge of **i2sx\_clk** and waits until the **i2sx\_wa1** line goes low. When a low level on **i2sx\_wa1** is detected during the falling edge of **i2sx\_clk** the interface shifts in the captured data with the next rising edge of **i2sx\_clk**. The next RX bit is captured with the following falling edge of **I2Sx\_CLK**. As soon as all 16 bits have been received, the content of the receive shift register is transferred to the RX buffer and **I2Sx\_RWADDR.WRADDR** is incremented by one. Thus, a continuous data reception can be achieved.

The length of the **i2sx\_way** pulse can be configured to be either one or two cycles of **i2sx\_clk**. Furthermore it is possible to configure whether **i2sx\_clk** should be running or switched either to low or high when no transmission takes place.

The I2S interface in PCM mode operating as master supports bit clock rates, which can be divided down from the reference frequency by a fractional divider. For details on reference clock calculation see section on Clock Configuration.



CONFIDENTIAL

Bi-directional Serial Audio Interface I2S1 (4-pins)

Figure 8-12 Timing of I2S Interface Signals in PCM Mode

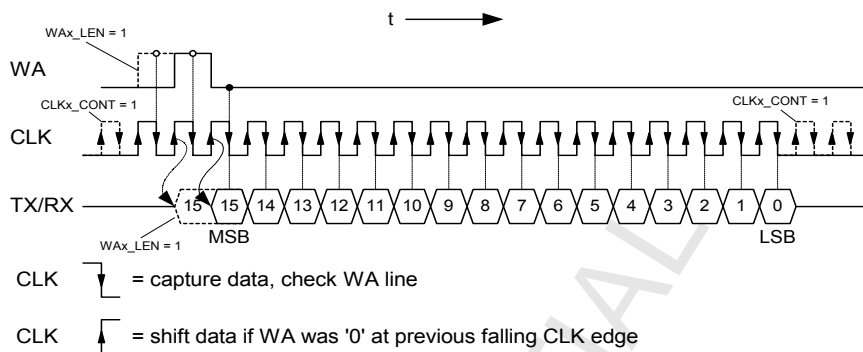


Figure 8-13 and Figure 8-14 show the timing of two PCM burst transfers, starting with the initialization of the interface.

After **I2Sx\_CTRL.I2SON** has been set the control bits **I2Sx\_CSEL.CLKx\_CONT** and **I2Sx\_RXCONF.CLKx\_OUT** can be configured. These bits define the behavior of the **i2sx\_clk** line when no transmission takes place.

**CLKx\_CONT** determines whether the clock should be switched off (**CLKx\_CONT** = 0) or remain running (**CLKx\_CONT** = 1) between two consecutive bursts. When the interface is initialized after **I2SON** has been set, the **i2sx\_clk** line is always idle until **I2Sx\_CTRL.TXSTART** or **I2Sx\_CTRL.RXSTART** is set. This behavior cannot be changed by the **CLKx\_CONT** bit.

**CONFIDENTIAL**

**Bi-directional Serial Audio Interface I2S1 (4-pins)**

The bit **CLKx\_OUT** determines the level of **i2sx\_clky** if the serial clock is idle. This clock is idle in two cases:

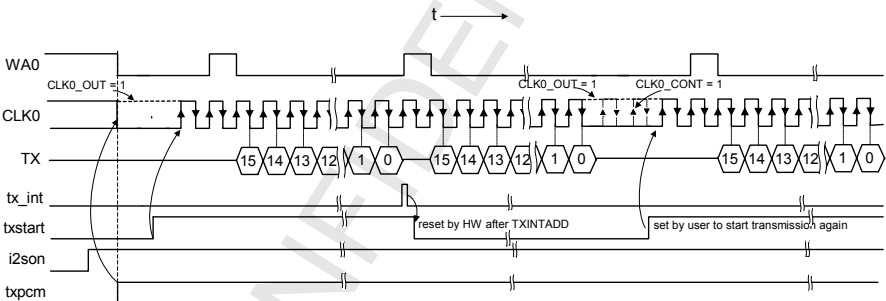
- After **I2SON** has been set, the **i2sx\_clky** line is idle until **RXSTART** or **TXSTART** has been set.
- Between two consecutive transmissions **i2sx\_clky** is idle if **CLKx\_CONT** = 0.

The configuration of **CLKx\_OUT** takes effect as soon as **I2Sx\_CTRL.TXPCM** or **I2Sx\_CTRL.RXPCM** is set regardless of the setting of **I2SON**.

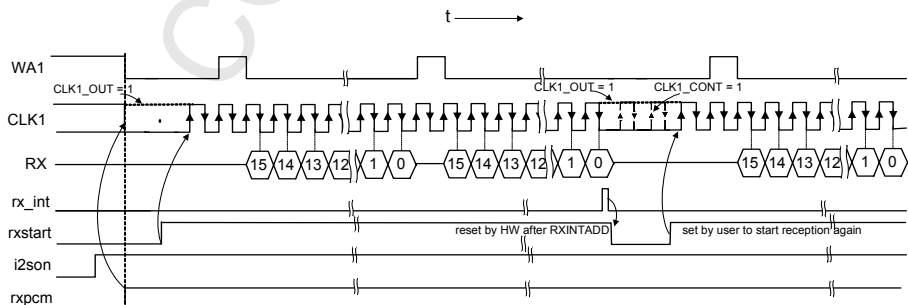
In the given example the transfer is stopped based on a given interrupt level. For the receive path this means that after the **RXINT** occurred **RXSTART** is reset and the reception is stopped immediately. In transmit direction **TXSTART** is reset as soon as **TXINT** occurs, but the frame referenced by the interrupt address is sent before the transmission finally stops.

After reading the received data from receive buffer or storing new data in the transmit buffer the transfer can be continued by setting **TXSTART** or **RXSTART** again.

**Figure 8-13 Timing of I2S Interface Signals in PCM Mode with Burst Transmission**



**Figure 8-14 Timing of I2S Interface Signals in PCM Mode with Burst Reception (from PCM\_frame)**



**CONFIDENTIAL****Bi-directional Serial Audio Interface I2S1 (4-pins)**

### **8.4.6 DAI Mode Operation of I2S Interface**

For an easy connection of PMB7870 to a GSM system simulator (complying with GSM specification 11.10-1), the I2S interface can be set into a dedicated DAI-mode. The DAI mode is used during system test for bit-exact verification of speech coder/decoder and for performance test of the analogue and acoustic devices in the handset. The particular test modes are set up through specific test signals which are either submitted from the system simulator through a layer 3 message (that is, over the air interface) or through dedicated control lines.

The data is transferred between the system simulator and the handset at a rate of 8000 samples per second in a two's complement 13-bit linear PCM data format. The transfer is performed in a duplex way using two data lines which are connected to the I2Sx\_TX and I2Sx\_RX pins of I2S. In DAI mode a synchronous connection is used, too, i.e. data are clocked over the interface in MSB first orientation at 104 kbit/s. Prior to the start of a DAI data exchange the system simulator transmits a reset signal which will reset the speech transcoder and the audio front-end (D-to-A and A-to-D paths). In DAI mode, the handset is the clock master for the system simulator and, hence, controls the clock line between system simulator and handset.

As the transfer between handset and system simulator is entirely synchronous and built upon the 13 bit data format, it is possible to use the four pins of a minimum I2S configuration for the physical DAI interface. The I2Sx\_WA0 pin is used as the reset input, the I2Sx\_TX, I2Sx\_RX and I2Sx\_CLK0 pins have their respective meanings. Data read at the I2Sx\_RX pin is clocked at the rising edge. Data is written to the I2Sx\_TX pin with the falling edge of the clock (in compliance with GSM 11.10).

The DAI mode operation can be described as follows. When the handset receives a request for DAI mode via a layer 3 message or via the DSPIN signals, the TEAKLite sets the I2S interface into DAI mode and waits for a reset signal from the system simulator. When the reset signal is received, an interrupt (I2Sx\_TX\_INT) to the TEAKLite is generated which is forwarded to the controller, and the clock output is set to HIGH. After the reset pulse is released (which results in an additional interrupt to the TEAKLite) the TEAKLite configures the interface for DAI transfers. The TEAKLite then initiates transfer of data (sets the respective start bits). Clock and data start simultaneously, that is, the first falling edge of the clock corresponds to the first TX-sample's MSB. The interface buffer behaves just as in normal I2S mode.

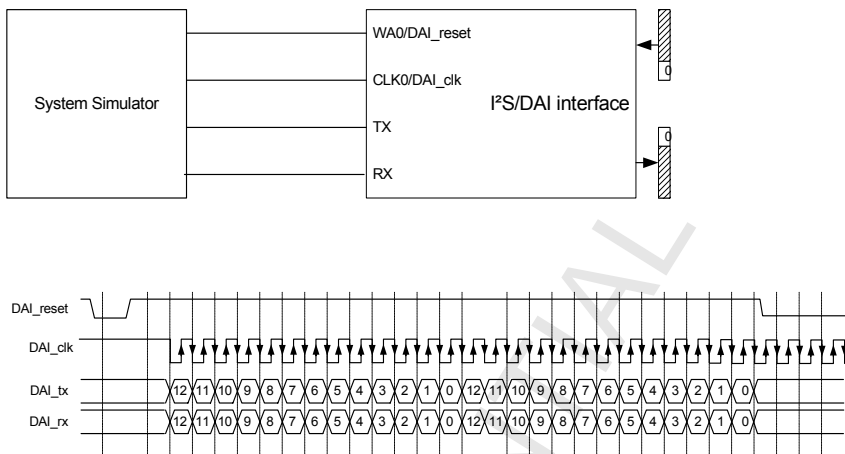
The received data is serial-to-parallel converted and written into the interface buffer in a way that the LSB of the received sample is written to the LSB of the buffer word. The upper 3 bits (15:13) are filled with zeros.

In transmit direction the samples are aligned to the MSB, that is, the most significant bit of the sample to be transmitted is located in the MSB position of the transmit buffer data word. The least significant 3 bits are ignored by the HW as they are not transmitted. A summary of the DAI mode of the I2S interface is given in **Figure 8-15**.

CONFIDENTIAL

## Bi-directional Serial Audio Interface I2S1 (4-pins)

**Figure 8-15 Signal Timing and System Setup for DAI Mode**



CONFIDENTIAL

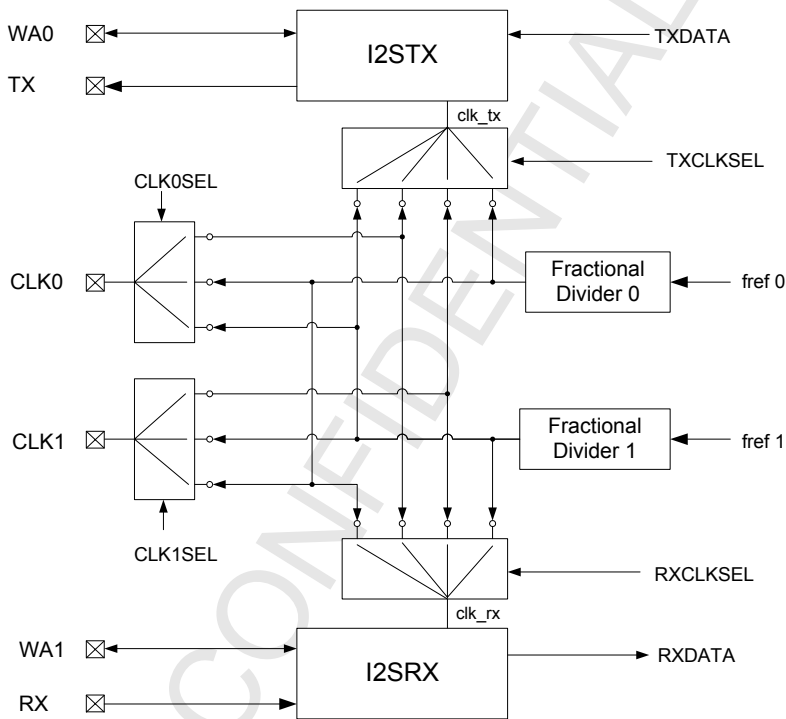
Bi-directional Serial Audio Interface I2S1 (4-pins)

## 8.4.7 Clock Configuration

### 8.4.7.1 Overview

Master or slave mode operation of the I2S interface as well as the combinations of transmission and reception can be configured by setting the clock source of the I2S interface in the register **I2Sx\_CSEL**. **Figure 8-16** describes the clock source selection and distribution.

**Figure 8-16 Block Diagram of Clock Source Selection and Distribution for I2S 1**



**Note:** To see the 6-pin-to-4-pin connections between the six TX, RX, WA, and CLK I2S block-pins and the E-GOLDradio chip pads, see **Figure 8-10**.

The direction of the word align signal I2Sx\_WAy is defined by the direction of the clock signal i2sx\_clky: The signals i2sx\_way and i2sx\_clky always have the same direction, which is defined by the fields **I2Sx\_CSEL.CLK0SEL** and **I2Sx\_CSEL.CLK1SEL** according to **Table 8-5** and **Table 8-6**.

**CONFIDENTIAL**

## Bi-directional Serial Audio Interface I2S1 (4-pins)

*Note: Special care has to be taken when only one transfer direction is used in normal mode with the I2S configured as master. If both the transmit and the receive path use the same clock source in master mode, that is, **CLK0SEL = CLK1SEL**, the WA and CLK outputs of the inactive direction will toggle anyway, even when **I2Sx\_CTRL.TxSTART** or **I2Sx\_CTRL.RxSTART** is set to zero.*

**Table 8-5 Selection of I2Sx\_CLK0 for I2Sx**

CLK0SEL	I2Sx_CLK0 Source	Direction of I2Sx_CLK0	Direction of I2Sx_WA0	Operation Mode
00	Fractional Divider 0	output	output	master
01	Fractional Divider 1	output	output	master
10	external	input	input	slave
11	-	-	-	reserved

**Table 8-6 Selection of I2Sx\_CLK1 for I2Sx**

CLK1SEL	I2Sx_CLK1 Source	Direction of I2Sx_CLK1	Direction of I2Sx_WA1	Operation Mode
00	Fractional Divider 0	output	output	master
01	Fractional Divider 1	output	output	master
10	external	input	input	slave
11	-	-	-	reserved

### 8.4.7.2 Support of 4 Pin Interface

The same clock and word alignments are used for both receive and transmit direction. The I2S is configured to operate only with 4 pins. In this configuration only pins RX, TX, CLK0, and WA0 are used externally.

### 8.4.7.3 Slave Mode

If the I2S is to be operated in slave mode, the clock source during this mode is external to PMB7870.

*Note: In slave operation the maximal external bit clock rate is 1/4 of the internal module clock.*

CONFIDENTIAL

Bi-directional Serial Audio Interface I2S1 (4-pins)

#### 4 Pin Configuration

To reduce the number of interface wires in slave mode both the transmit and the receive path must be connected to the same external WA and CLK signal.

For this purpose **I2Sx\_CSEL.CLK0SEL** and **I2Sx\_CSEL.CLK1SEL** need to be configured to select the external clock source. This also configures the WAx lines as input. Additionally, **I2Sx\_CSEL.TXCLKSEL** and **I2Sx\_CSEL.RXCLKSEL** must select the same external clock, CLK0.

In contrast to the internal combination of CLK signals in slave mode the WA lines cannot be connected internally in any case.

For the I2S1 interface it is not possible to combine the WA0 and WA1 signals internally. Therefore this interface can only be reduced to 5 pins. WA0 and WA1 must be connected externally.

The I2S1 interface in PMB7870 is limited to 4 external pins. The CLK signals coming from outside the chip are distributed to both CLK0 and CLK1 while the external WA signal drives WA0 and WA1.

#### 8.4.7.4 Master Mode

The I2Sx interface configured as master supports independent bit clock rates for reception and transmission and therefore provides two fractional dividers. To achieve the desired audio sampling rate the fractional divider must be programmed to generate the appropriate bit clock rate. This bit clock rate can be calculated by multiplying the sampling rate with the frame length.

In normal mode the frame length can be either 32, 48 or 64 bits. In contrast with this, the frame length in PCM mode does not correspond with the number of bits transmitted. Instead, one or two additional clock cycles are required for the WA signal. This leads to a frame length 17 or 18 bits in PCM mode, depending on the chosen length for the WA signal.

The resulting bit clock frequency of the fractional divider can be calculated according to the following equation:

$$f_{CLKy} = \frac{f_{ref} \cdot \text{NUMERATOR}}{4 \cdot \text{DENOMINATOR}} \quad [0.7]$$

The NUMERATOR values can be set in the registers **I2Sx\_NUM0** and **I2Sx\_NUM1** for the Fractional Divider 0 and the Fractional Divider 1, respectively. The DENOMINATOR values can be set in the registers **I2Sx\_DEN0** and **I2Sx\_DEN1**.

*Note: The maximum possible frequency of the bit clock in is 1/8 of the reference frequency.*

**CONFIDENTIAL**

**Bi-directional Serial Audio Interface I2S1 (4-pins)**

The clock signal is derived from the reference clock, which is selected by the bit fields FREF of the registers **I2Sx\_NUM0** and **I2Sx\_NUM1** according to [Table 8-7](#).

**Table 8-7 Selection of Reference Clock**

FREF[1:0]	Reference clock (fref 0/1)
00	module clock (clk_dsp)
01	104 MHz clock (clk_ms)
10	Reserved
11	Reserved

*Note: The module clock is running at the same frequency as the DSP and is therefore not fixed. If the module clock is used as reference the fractional divider settings need to be adapted when the DSP clock frequency changes.*

#### 4 Pin Configuration

For 4 pin configuration only one fractional divider is used. Therefore, **I2Sx\_CSEL.TXCLKSEL** and **I2Sx\_CSEL.RXCLKSEL** must be set accordingly to select the same fractional divider as source for the transmit and receive clock. This clock must also be connected to the CLK0 or CLK1 output, respectively, by setting **I2Sx\_CSEL.CLK0SEL** or **I2Sx\_CSEL.CLK1SEL** to the correct value.

In addition to the clock selection, configure the receive and transmit path to support the 4 pin interface mode. In particular, the registers **I2Sx\_RXCONF** and **I2Sx\_TXCONF** need to be set as follows:

- For normal mode data transfers the bits **POL**, **DEL**, **PERIOD**, and **WIDTH** must have the same value in **I2Sx\_RXCONF** and **I2Sx\_TXCONF**, while the **EDGE** bits need to be set to opposite values.
- For data transfers in PCM mode **WAx\_LEN**, **CLKx\_CONT**, and **CLKx\_OUT** must be set to the same value in registers **I2Sx\_RXCONF** and **I2Sx\_TXCONF**.

To start the interface configured for 4 pin operation the bits **I2Sx\_CTRL.I2SRXSTART** and **I2Sx\_CTRL.I2STXSTART** must be set simultaneously during the same register access to achieve synchronous operation of the state machines. Furthermore, the addresses for receive and transmit interrupt in registers **I2Sx\_RXINTADDR** and **I2Sx\_TXINTADDR** should be set to the same value. This ensures that both shifting clocks stop at the same time.

In case that only the RX direction is needed, it is still necessary to start the transmission because CLK0 and WA0 are needed for the TX state machine. The transmit interrupts should be ignored for this configuration and the transmit buffer should be filled with 0s to avoid toggling on the TX pin.



**CONFIDENTIAL**

**Bi-directional Serial Audio Interface I2S1 (4-pins)**

### Example Clock Configurations

Examples for usual sampling rates in master mode are 48 kHz, 44.1 kHz, 32 kHz, 24 kHz, 22.05 kHz, 16 kHz, 12 kHz, 11.025 kHz and 8 kHz. The following tables show the numerator/denominator values required to select those frequencies in PCM and normal mode.

In PCM mode each frame contains 16 bits of data. Taking into account one or two cycles for the pulse on the WA line, this results in a frame length of 17 or 18 shift clock cycles as given in the tables below. Furthermore, the tables contain the settings for 32, 48 and 64 cycle frame lengths, which can be selected in normal mode.

**Table 8-8** shows the corresponding settings of the numerator and the denominator values for a 89.1 MHz reference clock. In the given configuration the (4\*denominator)/numerator ratio always results in an integer. This way the fractional divider will not produce additional jitter, but at the cost of higher frequency deviations.

**Table 8-8 Bit Clock Rates with 89.1 MHz Reference and Integer Ratio of Fractional Divider**

(with Minimum Jitter)

Nominal Audio Sampling Rates	Frame Length <sup>1)</sup>	Nominal Bit Clock	NUM	DEN	Real Bit Clock	Bit Clock Deviation
8.000 kHz	17	136.00 kHz	4	655	136.03 kHz	+ 0.02%
	18	144.00 kHz	4	619	143.94 kHz	-0.04%
	32	256.00 kHz	4	348	256.03 kHz	+ 0.01%
	48	384.00 kHz	4	232	384.05 kHz	+ 0.01%
	64	512.00 kHz	4	174	512.07 kHz	+ 0.01%
11.025 kHz	17	187.43 kHz	4	475	187.58 kHz	+ 0.08%
	18	198.45 kHz	4	449	198.44 kHz	- 0.00%
	32	352.80 kHz	4	253	352.17 kHz	- 0.18%
	48	529.20 kHz	4	168	530.36 kHz	+ 0.22%
	64	705.60 kHz	4	126	707.14 kHz	+ 0.22%
12.000 kHz	17	204.00 kHz	4	437	203.89 kHz	- 0.05%
	18	216.00 kHz	4	413	215.74 kHz	- 0.12%
	32	384.00 kHz	4	232	384.05 kHz	+ 0.01%
	48	576.00 kHz	4	155	574.84 kHz	- 0.20%
	64	768.00 kHz	4	116	768.10 kHz	+ 0.01%

**CONFIDENTIAL**

**Bi-directional Serial Audio Interface I2S1 (4-pins)**

**Table 8-8 Bit Clock Rates with 89.1 MHz Reference and Integer Ratio of Fractional Divider**

(with Minimum Jitter)

Nominal Audio Sampling Rates	Frame Length <sup>1)</sup>	Nominal Bit Clock	NUM	DEN	Real Bit Clock	Bit Clock Deviation
16.000 kHz	17	272.00 kHz	4	328	271.65 kHz	- 0.13%
	18	288.00 kHz	4	309	288.35 kHz	+ 0.12%
	32	512.00 kHz	4	174	512.07 kHz	+ 0.01%
	48	768.00 kHz	4	116	768.10 kHz	+ 0.01%
	64	1024.00 kHz	4	87	1024.14 kHz	+ 0.01%
22.050 kHz	17	374.85 kHz	4	238	374.37 kHz	- 0.13%
	18	396.90 kHz	4	224	397.77 kHz	+ 0.22%
	32	705.60 kHz	4	126	707.14 kHz	+ 0.22%
	48	1058.40 kHz	4	84	1060.71 kHz	+ 0.22%
	64	1411.20 kHz	4	63	1414.29 kHz	+ 0.22%
24.000 kHz	17	408.00 kHz	4	218	408.72 kHz	+ 0.18%
	18	432.00 kHz	4	206	432.52 kHz	+ 0.12%
	32	768.00 kHz	4	116	768.10 kHz	+ 0.01%
	48	1152.00 kHz	4	77	1157.14 kHz	+ 0.45%
	64	1536.00 kHz	4	58	1536.21 kHz	+ 0.01%
32.000 kHz	17	544.00 kHz	4	164	543.29 kHz	- 0.13%
	18	576.00 kHz	4	155	574.84 kHz	- 0.20%
	32	1024.00 kHz	4	87	1024.14 kHz	+ 0.01%
	48	1536.00 kHz	4	58	1536.21 kHz	+ 0.01%
	64	2048.00 kHz	4	44	2025 kHz	- 1.12%
44.100 kHz	17	749.70 kHz	4	119	748.74 kHz	- 0.13%
	18	793.80 kHz	4	112	795.54 kHz	+ 0.22%
	32	1411.20 kHz	4	63	1414.29 kHz	+ 0.22%
	48	2116.80 kHz	4	42	2121.43 kHz	+ 0.22%
	64	2822.40 kHz	4	32	2784.38 kHz	- 1.35%

**CONFIDENTIAL**

**Bi-directional Serial Audio Interface I2S1 (4-pins)**

**Table 8-8 Bit Clock Rates with 89.1 MHz Reference and Integer Ratio of Fractional Divider**

(with Minimum Jitter)

Nominal Audio Sampling Rates	Frame Length <sup>1)</sup>	Nominal Bit Clock	NUM	DEN	Real Bit Clock	Bit Clock Deviation
48.000 kHz	17	816.00 kHz	4	109	817.43 kHz	+ 0.18%
	18	864.00 kHz	4	103	865.05 kHz	+ 0.12%
	32	1536.00 kHz	4	58	1536.21 kHz	+ 0.01%
	48	2304.00 kHz	4	39	2284.62 kHz	- 0.84%
	64	3072.00 kHz	4	29	3072.41 kHz	+ 0.01%

<sup>1)</sup> Number of shift clock cycles

**Table 8-9** shows the settings of the numerator and the denominator values for a 89.1 MHz reference clock. In the given configuration the (4\*denominator)/numerator ratio does not need to result in an integer. This way the fractional divider will produce additional jitter, but the given values result in the minimum possible bit clock deviation.

**Table 8-9 Bit Clock Rates with 89.1 MHz Reference and Non-Integer Ratio of Fractional Divider**

(with Minimum Bit Clock Deviation)

Nominal Audio Sampling Rates	Frame Length <sup>1)</sup>	Nominal Bit Clock	NUM	DEN	Real Bit Clock	Bit Clock Deviation
8.000 kHz	17	136.00 kHz	136	22275	136.00 kHz	+0.00%
	18	144.00 kHz	16	2475	144.00 kHz	+0.00%
	32	256.00 kHz	256	22275	256.00 kHz	+0.00%
	48	384.00 kHz	128	7425	384.00 kHz	+0.00%
	64	512.00 kHz	512	22275	512.00 kHz	+0.00%
11.025 kHz	17	187.43 kHz	302	35891	187.43 kHz	+0.00%
	18	198.45 kHz	49	5500	198.45 kHz	-0.00%
	32	352.80 kHz	196	12375	352.80 kHz	+0.00%
	48	529.20 kHz	98	4125	529.20 kHz	+0.00%
	64	705.60 kHz	392	12375	705.60 kHz	+0.00%

**CONFIDENTIAL**

**Bi-directional Serial Audio Interface I2S1 (4-pins)**

**Table 8-9 Bit Clock Rates with 89.1 MHz Reference and Non-Integer Ratio of Fractional Divider**

(with Minimum Bit Clock Deviation)

Nominal Audio Sampling Rates	Frame Length <sup>1)</sup>	Nominal Bit Clock	NUM	DEN	Real Bit Clock	Bit Clock Deviation
12.000 kHz	17	204.00 kHz	68	7425	204.00 kHz	+0.00%
	18	216.00 kHz	8	825	216.00 kHz	+0.00%
	32	384.00 kHz	128	7425	384.00 kHz	+0.00%
	48	576.00 kHz	64	2475	576.00 kHz	+0.00%
	64	768.00 kHz	256	7425	768.00 kHz	+0.00%
16.000 kHz	17	272.00 kHz	272	22275	272.00 kHz	+0.00%
	18	288.00 kHz	32	2475	288.00 kHz	+0.00%
	32	512.00 kHz	512	22275	512.00 kHz	+0.00%
	48	768.00 kHz	256	7425	768.00 kHz	+0.00%
	64	1024.00 kHz	1024	22275	1024.00 kHz	+0.00%
22.050 kHz	17	374.85 kHz	833	49500	374.85 kHz	+0.00%
	18	396.90 kHz	49	2750	396.90 kHz	-0.00%
	32	705.60 kHz	392	12375	705.60 kHz	+0.00%
	48	1058.40 kHz	196	4125	1058.40 kHz	+0.00%
	64	1411.20 kHz	784	12375	1411.20 kHz	+0.00%
24.000 kHz	17	408.00 kHz	136	7425	408.00 kHz	+0.00%
	18	432.00 kHz	16	825	432.00 kHz	+0.00%
	32	768.00 kHz	256	7425	768.00 kHz	+0.00%
	48	1152.00 kHz	128	2475	1152.00 kHz	+0.00%
	64	1536.00 kHz	512	7425	1536.00 kHz	+0.00%
32.000 kHz	17	544.00 kHz	544	22275	544.00 kHz	+0.00%
	18	576.00 kHz	64	2475	576.00 kHz	+0.00%
	32	1024.00 kHz	939	20426	1024.00 kHz	+0.00%
	48	1536.00 kHz	512	7425	1536.00 kHz	+0.00%
	64	2048.00 kHz	1109	12062	2048.00 kHz	+0.00%

**CONFIDENTIAL**

**Bi-directional Serial Audio Interface I2S1 (4-pins)**

**Table 8-9      Bit Clock Rates with 89.1 MHz Reference and Non-Integer Ratio of Fractional Divider**

(with Minimum Bit Clock Deviation)

Nominal Audio Sampling Rates	Frame Length <sup>1)</sup>	Nominal Bit Clock	NUM	DEN	Real Bit Clock	Bit Clock Deviation
44.100 kHz	17	749.70 kHz	833	24750	749.70 kHz	+0.00%
	18	793.80 kHz	49	1375	793.80 kHz	+0.00%
	32	1411.20 kHz	1024	22275	1411.20 kHz	+0.00%
	48	2116.80 kHz	392	4125	2116.80 kHz	+0.00%
	64	2822.40 kHz	1568	12375	2822.40 kHz	-0.00%
48.000 kHz	17	816.00 kHz	272	7425	816.00 kHz	+0.00%
	18	864.00 kHz	32	825	864.00 kHz	+0.00%
	32	1536.00 kHz	512	7425	1536.00 kHz	+0.00%
	48	2304.00 kHz	256	2475	2304.00 kHz	+0.00%
	64	3072.00 kHz	1024	7425	3072.00 kHz	+0.00%

<sup>1)</sup> Number of shift clock cycles

**CONFIDENTIAL**

**Bi-directional Serial Audio Interface I2S1 (4-pins)**

**Table 8-10** shows the settings of the numerator and the denominator values for a 104 MHz reference clock. In the given configuration the (4\*denominator)/numerator ratio always results in an integer. This way the fractional divider will not produce additional jitter, but at the cost of higher frequency deviations.

**Table 8-10 Bit Clock Rates with 104 MHz Reference and Integer Ratio of Fractional Divider**

(with Minimum Jitter)

Nominal Audio Sampling Rates	Frame Length <sup>1)</sup>	Nominal Bit Clock	NUM	DEN	Real Bit Clock	Bit Clock Deviation
8.000 kHz	17	136.00 kHz	4	762	135.95 kHz	- 0.04%
	18	144.00 kHz	4	722	144.04 kHz	+ 0.03%
	32	256.00 kHz	4	406	255.16 kHz	+ 0.06%
	48	384.00 kHz	4	271	383.76 kHz	- 0.06%
	64	512.00 kHz	4	203	512.32 kHz	+ 0.06%
11.025 kHz	17	187.43 kHz	4	555	187.39 kHz	- 0.02%
	18	198.45 kHz	4	524	198.47 kHz	+ 0.01%
	32	352.80 kHz	4	295	352.54 kHz	- 0.07%
	48	529.20 kHz	4	197	527.92 kHz	- 0.24%
	64	705.60 kHz	4	147	707.48 kHz	+ 0.27%
12.000 kHz	17	204.00 kHz	4	510	203.92 kHz	+ 0.04%
	18	216.00 kHz	4	481	216.22 kHz	+ 0.10%
	32	384.00 kHz	4	271	383.76 kHz	- 0.06%
	48	576.00 kHz	4	181	576.58 kHz	- 0.25%
	64	768.00 kHz	4	135	770.37 kHz	+ 0.31%
16.000 kHz	17	272.00 kHz	4	382	272.25 kHz	+ 0.09%
	18	288.00 kHz	4	361	288.09 kHz	+ 0.03%
	32	512.00 kHz	4	203	512.32 kHz	+ 0.06
	48	768.00 kHz	4	135	770.37 kHz	+ 0.31
	64	1024.00 kHz	4	102	1019.61 kHz	- 0.43

**CONFIDENTIAL**

**Bi-directional Serial Audio Interface I2S1 (4-pins)**

**Table 8-10 Bit Clock Rates with 104 MHz Reference and Integer Ratio of Fractional Divider**

(with Minimum Jitter)

Nominal Audio Sampling Rates	Frame Length <sup>1)</sup>	Nominal Bit Clock	NUM	DEN	Real Bit Clock	Bit Clock Deviation
22.050 kHz	17	374.85 kHz	4	277	375.45 kHz	+ 0.16%
	18	396.90 kHz	4	262	396.96 kHz	+ 0.01%
	32	705.60 kHz	4	147	707.48 kHz	+ 0.27%
	48	1058.40 kHz	4	98	1061.22 kHz	+ 0.27%
	64	1411.20 kHz	4	74	1405.41 kHz	- 0.41%
24.000 kHz	17	408.00 kHz	4	255	407.84 kHz	- 0.04%
	18	432.00 kHz	4	241	431.54 kHz	- 0.11%
	32	768.00 kHz	4	135	770.37 kHz	+ 0.31%
	48	1152.00 kHz	4	90	1155.56 kHz	+ 0.31%
	64	1536.00 kHz	4	68	1529.41 kHz	- 0.43%
32.000 kHz	17	544.00 kHz	4	191	544.50 kHz	+ 0.09%
	18	576.00 kHz	4	181	574.59 kHz	- 0.25%
	32	1024.00 kHz	4	102	1019.61 kHz	- 0.43%
	48	1536.00 kHz	4	68	1529.41 kHz	- 0.43%
	64	2048.00 kHz	4	51	2039.22 kHz	- 0.43%
44.100 kHz	17	749.70 kHz	4	139	748.20 kHz	- 0.20%
	18	793.80 kHz	4	131	793.89 kHz	+ 0.01%
	32	1411.20 kHz	4	74	1405.41 kHz	- 0.41%
	48	2116.80 kHz	4	49	2122.45 kHz	+ 0.27%
	64	2822.40 kHz	4	37	2810.81 kHz	- 0.41%
48.000 kHz	17	816.00 kHz	4	127	818.90 kHz	+ 0.36%
	18	864.00 kHz	4	120	866.67 kHz	+ 0.31%
	32	1536.00 kHz	4	68	1529.41 kHz	- 0.43%
	48	2304.00 kHz	4	45	2311.11 kHz	+ 0.31%
	64	3072.00 kHz	4	34	3058.82 kHz	- 0.43%

<sup>1)</sup> Number of shift clock cycles

**CONFIDENTIAL**

**Bi-directional Serial Audio Interface I2S1 (4-pins)**

**Table 8-11** shows the settings of the numerator and the denominator values for a 104 MHz reference clock. In the given configuration the (4\*denominator)/numerator ratio does not need to result in an integer. This way the fractional divider will produce additional jitter, but the given values result in the minimum possible bit clock deviation.

**Table 8-11 Bit Clock Rates with 104 MHz Reference and Non-Integer Ratio of Fractional Divider**

(with Minimum Bit Clock Deviation)

Nominal Audio Sampling Rates	Frame Length <sup>1)</sup>	Nominal Bit Clock	NUM	DEN	Real Bit Clock	Bit Clock Deviation
8.000 kHz	17	136.00 kHz	17	3250	136.00 kHz	+0.00%
	18	144.00 kHz	9	1625	144.00 kHz	+0.00%
	32	256.00 kHz	16	1625	256.00 kHz	+0.00%
	48	384.00 kHz	24	1625	384.00 kHz	+0.00%
	64	512.00 kHz	32	1625	512.00 kHz	+0.00%
11.025 kHz	17	187.43 kHz	103	14288	187.43 kHz	+0.00%
	18	198.45 kHz	65	8516	198.45 kHz	-0.00%
	32	352.80 kHz	441	32500	352.80 kHz	+0.00%
	48	529.20 kHz	1323	65000	529.20 kHz	+0.00%
	64	705.60 kHz	441	16250	705.60 kHz	+0.00%
12.000 kHz	17	204.00 kHz	51	6500	204.00 kHz	+0.00%
	18	216.00 kHz	27	3250	216.00 kHz	+0.00%
	32	384.00 kHz	24	1625	384.00 kHz	+0.00%
	48	576.00 kHz	36	1625	576.00 kHz	+0.00%
	64	768.00 kHz	48	1625	768.00 kHz	+0.00%
16.000 kHz	17	272.00 kHz	17	1625	272.00 kHz	+0.00%
	18	288.00 kHz	18	1625	288.00 kHz	+0.00%
	32	512.00 kHz	32	1625	512.00 kHz	+0.00%
	48	768.00 kHz	48	1625	768.00 kHz	+0.00%
	64	1024.00 kHz	64	1625	1024.00 kHz	+0.00%



**CONFIDENTIAL**

**Bi-directional Serial Audio Interface I2S1 (4-pins)**

**Table 8-11 Bit Clock Rates with 104 MHz Reference and Non-Integer Ratio of Fractional Divider**

(with Minimum Bit Clock Deviation)

Nominal Audio Sampling Rates	Frame Length <sup>1)</sup>	Nominal Bit Clock	NUM	DEN	Real Bit Clock	Bit Clock Deviation
22.050 kHz	17	374.85 kHz	817	56668	374.85 kHz	+0.00%
	18	396.90 kHz	65	4258	396.90 kHz	+0.00%
	32	705.60 kHz	441	16250	705.60 kHz	+0.00%
	48	1058.40 kHz	1323	32500	1058.40 kHz	+0.00%
	64	1411.20 kHz	441	8125	1411.20 kHz	+0.00%
24.000 kHz	17	408.00 kHz	51	3250	408.00 kHz	+0.00%
	18	432.00 kHz	27	1625	432.00 kHz	+0.00%
	32	768.00 kHz	48	1625	768.00 kHz	+0.00%
	48	1152.00 kHz	72	1625	1152.00 kHz	+0.00%
	64	1536.00 kHz	96	1625	1536.00 kHz	+0.00%
32.000 kHz	17	544.00 kHz	34	1625	544.00 kHz	+0.00%
	18	576.00 kHz	36	1625	576.00 kHz	+0.00%
	32	1024.00 kHz	64	1625	1024.00 kHz	+0.00%
	48	1536.00 kHz	96	1625	1536.00 kHz	+0.00%
	64	2048.00 kHz	128	1625	2048.00 kHz	+0.00%
44.100 kHz	17	749.70 kHz	817	28334	749.70 kHz	+0.00%
	18	793.80 kHz	65	2129	793.80 kHz	+0.00%
	32	1411.20 kHz	441	8125	1411.20 kHz	+0.00%
	48	2116.80 kHz	1323	16250	2116.80 kHz	+0.00%
	64	2822.40 kHz	882	8125	2822.40 kHz	+0.00%
48.000 kHz	17	816.00 kHz	51	1625	816.00 kHz	+0.00%
	18	864.00 kHz	54	1625	864.00 kHz	+0.00%
	32	1536.00 kHz	96	1625	1536.00 kHz	+0.00%
	48	2304.00 kHz	144	1625	2304.00 kHz	+0.00%
	64	3072.00 kHz	192	1625	3072.00 kHz	+0.00%

<sup>1)</sup> Number of shift clock cycles

**CONFIDENTIAL**

## Bi-directional Serial Audio Interface I2S1 (4-pins)

For the parallel-to-serial conversion the clock signal clk\_tx and for the serial-to-parallel conversion the clock signal clk\_rx are used (see [Figure 8-16](#)). With the bit fields [I2Sx\\_CSEL.TXCLKSEL](#) and [I2Sx\\_CSEL.RXCLKSEL](#) the sources of these clocks can be set according to [Table 8-12](#).

**Table 8-12 Selection of clk\_TX and clk\_RX and WA signals**

TXCLKSEL	clk_TX	WA (TX)	RXCLKSEL	clk_RX	WA (RX)
00	Fractional Divider 0	from I2STX	00	Fractional Divider 1	from I2SRX
01	i2sx_clk1	WA0	01	i2sx_clk0	WA1
10	i2sx_clk0	WA0	10	i2sx_clk1	WA1
11	Fractional Divider 1	from I2STX	11	Fractional Divider 0	from I2SRX

### 8.4.7.5 Jitter Of Output Signals

There are three different sources which can cause the output signals to jitter:

- Jitter from PLL, phase shifter and pad non-linearizations
- Jitter of reference clock from fractional divider
- Jitter due to synchronization of reference clock into peripheral clock domain

The individual contributions of these three jitter types sum up to the total jitter. While the jitter caused by the first source is inevitable the occurrence of the other two jitters depends on the selected clock configuration and the operation mode.

The fractional divider causes a jitter if the  $(4 \cdot \text{denominator}) / \text{numerator}$  ratio does not result in an integer. As the fractional divider is not used in slave mode this kind of jitter can only occur in master mode.

In addition to this a synchronization jitter can occur if the reference clock is different from the peripheral clock, which is running at the clk\_dsp frequency. In slave mode the reference clock comes from outside the chip. Therefore, the synchronization jitter is always present in slave mode.

[Table 8-13](#) gives an overview on all possible combinations in master mode. If clk\_dsp and clk\_ms are running at the same frequency no synchronization jitter will occur.

**CONFIDENTIAL**

## Bi-directional Serial Audio Interface I2S1 (4-pins)

In slave mode the jitter only consists of the synchronization jitter and the fixed jitter, which always leads to a total jitter of 2 ns + 1 clk\_dsp cycle.

**Table 8-13 Jitter of CLKx, WAX and TX Outputs in Master Mode**

Ref clock	clk_dsp	clk_dsp	clk_ms	clk_ms
<b>Sync jitter<sup>1)</sup></b>	0	0	1 clk_dsp cycle	1 clk_dsp cycle
<b>Ratio (4*denominator) /numerator</b>	integer	non-integer	integer	non-integer
<b>FDIV jitter</b>	0	1 clk_dsp cycle	0	1 clk_ms cycle
<b>Fixed jitter</b>	2 ns	2 ns	2 ns	2 ns
<b>Total jitter</b>	2 ns	1 clk_dsp cycle +2 ns	1 clk_dsp cycle +2 ns	1 clk_dsp cycle +1 clk_ms cycle +2 ns

<sup>1)</sup> Does not occur if clk\_dsp = clk\_ms

### 8.4.7.6 Delay Of Output Signals

In master mode the output signals show a variable delay with respect to the output clock. This delay depends on the frequency of clk\_dsp. [Table 8-14](#) gives an overview. This delay is increased by the fixed path delay from the I2S block to the pads at the chip, refer to [Section 13.2.1.2.6 I2S \(on Page 1346\)](#).

**Table 8-14 Delay Of Output Signals In Master Mode**

Signals	Delay
CLKx to WAX	2 clk_dsp cycles
CLKx to TX	1 clk_dsp cycle

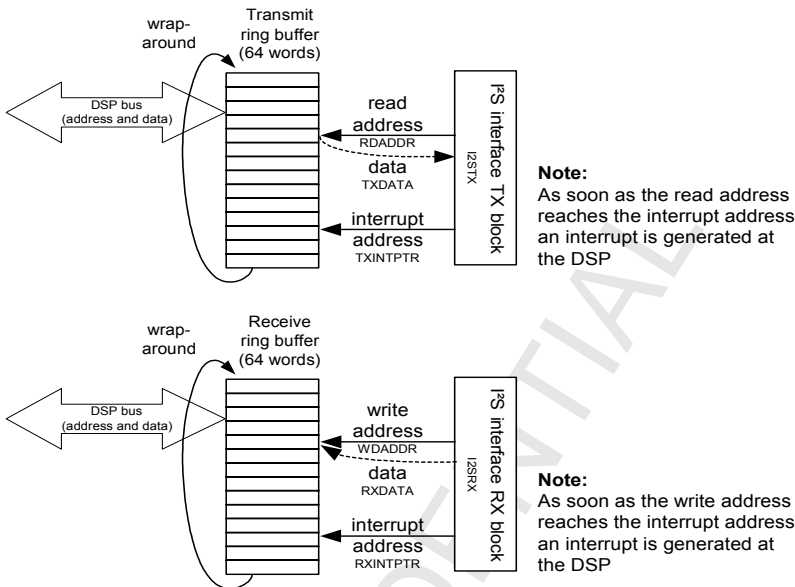
### 8.4.8 Interface to TEAKLite Bus

The data transfer between TEAKLite and I2S Interface is performed via dual port data ring buffers with a size of 64 words of 16 bits each. The communication is controlled by the TEAKLite interrupt unit. The ring buffers are implemented as dual port memory, which can be accessed by the TEAKLite with 2 wait states. A simplified block diagram of the interface for the I2Sx is shown in [Figure 8-17](#).

CONFIDENTIAL

Bi-directional Serial Audio Interface I2S1 (4-pins)

Figure 8-17 Functional Block Diagram of the I2Sx Transmit and Receive Buffers



The data buffers and registers of the I2S interfaces are directly accessible from the TEAKLite via the TEAKLite bus. The dual port data buffers between the TEAKLite and the I2S interface are accessible by the TEAKLite as well as the I2S interfaces. In order to transmit audio data (normal mode) the TEAKLite has to write pairs of stereo audio samples to consecutive addresses in the ring buffers. These audio samples will be read by the I2S interfaces at a later time.

Moreover, the TEAKLite may access each 16-bit word of the ring buffers independently, this means that all 64 words of each buffer are mapped to the data address space of the TEAKLite. In detail the transmit ring buffer is mapped to addresses  $XXXX_H$  up to  $XXXX_H + 3F_H$  and the receive ring buffer is mapped to addresses  $XXXX_H + 40_H$  up to  $XXXX_H + 40_H + 3F_H$ . In this way the TEAKLite has full control what memory location it wants to write to and from which it wants to read. It is also possible to operate the interface buffers in a way best suiting the needs of the software as various logical data types (for example, a FIFO queue) may be implemented on these buffers. The receive buffer of the I2S interface is readable and writable from the TEAKLite while the transmit buffer can only be written by the DSP.

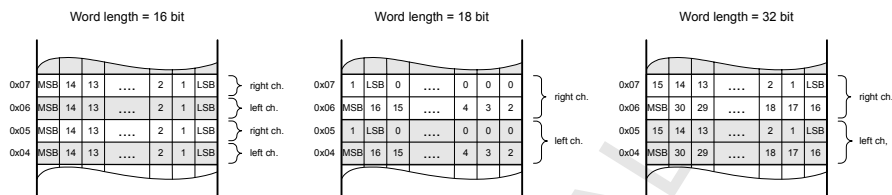
In the normal mode the buffers hold left channel data and right channel data in an alternating order. The mapping starts always at the buffer address  $00_H$  with the left channel. Thereby the mapping of the left and right channels into the buffer is fixed.

CONFIDENTIAL

## Bi-directional Serial Audio Interface I2S1 (4-pins)

**Figure 8-18** shows how the data words for different word lengths are mapped into the ring buffer.

**Figure 8-18 Mapping of Data Words into the Buffer**



For flow control an interrupt mechanism is implemented which causes an interrupt (I2Sx\_TX\_INT / I2Sx\_RX\_INT) at the TEAKLite as soon as the I2S interface reads from / writes to a specific buffer location. The address where the interrupt is generated at the TEAKLite is stored in the register **I2Sx\_TXINTADDR** and **I2Sx\_RXINTADDR**. This registers can be written to by the TEAKLite even during operation of the peripheral.

### 8.4.8.1 Data Transmission

The read address where the I2S interface currently reads from is stored in **I2Sx\_RWADDR.RDADDR**. It is readable by the TEAKLite. The address **RDADDR** is auto incremented with each read access by the I2S interface to the ring buffer and is wrapped around from the end address to the start address. The write position of the TEAKLite to the transmit buffer is controlled by the TEAKLite and hence does not need to be stored in a separate register in the I2S interface.

As soon as the read address **RDADDR** equals the interrupt address **I2Sx\_RXINTADDR.TXINTPTR**, an interrupt is generated at the TEAKLite. This means that **INT\_FINTB0 (on Page 344)**. **I2SxTX** is set by the interrupt line I2Sx\_TX\_INT. When the TEAKLite enters the interrupt service routine for this interrupt it has to reset the interrupt source (**INT\_FINTB0.RI2SxTX**) and to set a new value for the next interrupt position.

The I2S interface is activated by setting **I2Sx\_CTRL.I2SON** to 1. This powers up the interface but does not yet activate the output clock until **I2Sx\_CTRL.I2STXSTART** is set to 1. Hence, it is possible to write to the data buffer without sending dummy values to the external device. After writing data in the transmit buffer, the PCM or DAI mode can be selected.

When the bit **I2STXSTART** is set to 1 the I2S hardware starts to read from the transmit buffer. In normal mode the first element to be transmitted belongs to the left channel (start address). The transmission can be stopped by setting **I2STXSTART** to 0. After setting **I2STXSTART** back to 1 again the pair of samples (in normal mode) / the sample (in PCM mode) is read that immediately follows the one of the last read access before

## CONFIDENTIAL

## Bi-directional Serial Audio Interface I2S1 (4-pins)

the data transfer had been stopped. After reset or after **I2SON** has been toggled both the interrupt pointer and the read pointer are reset to zero.

#### 8.4.8.2 Data Reception

The write address where the I2S interface currently writes to is stored in **I2Sx\_RWADDR.WRADDR**. This register is readable from the TEAKLite. The address **WRADDR** is auto incremented at each write access by the I2S interface to the ring buffer and is wrapped around from the end address to the start address. The read position of the TEAKLite from the receive buffer is controlled by the TEAKLite and hence does not need to be stored in a separate register in the I2S interface.

As soon as the write address **WRADDR** equals the interrupt address **I2Sx\_RXINTADDR.RXINTPTR**, an interrupt is generated at the TEAKLite. This means that **INT\_FINTB0.I2SxRX** is set by the interrupt line **I2Sx\_RX\_INT**. Within the interrupt service routine the TEAKLite must reset the interrupt source (**INT\_FINTB0.RI2SxRX**) and set a new value for the next interrupt position.

The I2S interface is activated by setting **I2Sx\_CTRL.I2SON** to 1. This powers up the interface, but does not yet activate the output clock until **I2Sx\_CTRL.I2SRXSTART** is set to 1. After activating and starting the I2S interface in normal mode, the first element being received belongs to the left channel (start address).

When the bit **I2SRXSTART** is set to 1 the I2S hardware starts writing to the receive buffer. The reception can be stopped by setting **I2SRXSTART** to 0. After setting **I2SRXSTART** back to 1 again the pair of samples (in normal mode) / the sample (in PCM mode) is written to the location that immediately follows the one of the last write access before the data reception had been stopped. After reset or after **I2SON** has been toggled both the interrupt pointer and the write pointer are reset to zero.



**CONFIDENTIAL**

**Bi-directional Serial Audio Interface I2S1 (4-pins)**

Field	Bits	Type	Description
<b>I2SRXSTART</b>	2	rwh	<b>Start operation of I2S data reception</b> 0: Data reception halted. 1: Start data reception.
<b>TXPCM</b>	5	rw	<b>PCM mode select for transmit direction</b> 0: Normal operation. 1: PCM mode operation.
<b>RXPCM</b>	6	rw	<b>PCM mode select for receive direction</b> 0: Normal operation. 1: PCM mode operation.
<b>DAI_EN</b>	7	rw	<b>DAI mode enable</b> 0: Normal operation (16 bit data format). 1: DAI mode switched on (13 bit data format). <i>Note: In DAI mode all settings from the <b>I2Sx_TXCONF</b> and <b>I2Sx_RXCONF</b> registers are ignored.</i> <i>Note: In the DAI mode <b>I2Sx_CSEL.TXCLKSEL</b> must be set to 00<sub>H</sub>.</i>
<b>RESERVED</b>	15:8, 4:3	r	Reserved; these bits must be left at their reset values.

*Note: **DAI\_EN** must not be set if **TXPCM** or **RXPCM** is set because this will lead to undefined behavior.*



**CONFIDENTIAL**

**Bi-directional Serial Audio Interface I2S1 (4-pins)**

### 8.4.9.2 Clock Select Register

#### I2Sx\_CSEL

Master or slave mode operation of the I2S interface as well as the combinations of transmission and reception can be selected by setting the clock source of the I2S interface in the register [I2Sx\\_CSEL](#).

#### I2S1\_CSEL

#### I2S2\_CSEL

#### Clock Select Register

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								CLK1 SEL	RXCLK SEL		CLK0 SEL		TXCLK SEL		

Field	Bits	Type	Description
TXCLKSEL	1:0	rw	Select clock source for signal clk_tx (refer to <a href="#">Table 8-12</a> )
CLK0SEL	3:2	rw	Select source for signal I2Sx_CLK0 (refer to <a href="#">Table 8-5</a> )
RXCLKSEL	5:4	rw	Select clock source for signal clk_rx (refer to <a href="#">Table 8-12</a> )
CLK1SEL	7:6	rw	Select source for signal I2Sx_CLK1 (refer to <a href="#">Table 8-6</a> )
RESERVED	15:8	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**Bi-directional Serial Audio Interface I2S1 (4-pins)**

### 8.4.9.3 Read/Write Address Register

**I2Sx\_RWADDR**

**I2S1\_RWADDR**

**I2S2\_RWADDR**

**Read/Write Address Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		WRADDR						RESERVED		RDADDR					

Field	Bits	Type	Description
<b>RDADDR</b>	5:0	rh	Address of transmit interface ring buffer read position
<b>WRADDR</b>	13:8	rh	Address of receive interface ring buffer write position
<b>RESERVED</b>	17:6, 15:14	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**Bi-directional Serial Audio Interface I2S1 (4-pins)**

### 8.4.9.4 Divider 0 Numerator Register

#### I2Sx\_NUM0

The firmware uses **I2Sx\_NUM0** to set the numerator of the fractional divider 0 and select the reference frequency of the transmit baud rate generator.

#### I2S1\_NUM0

#### I2S2\_NUM0

#### Divider 0 Numerator Register

Reset value: 0001<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		FREF		RESERVED	NUMERATOR										

Field	Bits	Type	Description
<b>NUMERATOR</b>	10:0	rw	<b>Numerator</b> Determines the numerator value of the fractional divider 0 described in <a href="#">Section 8.4.7 Clock Configuration (on Page 385)</a> . <i>Note: The value 0 is not permitted and leads to undefined results.</i>
<b>FREF</b>	13:12	rw	<b>FREF selection</b> 00: Reference input 0 (Module clock). 01: Reference input 1 (104 MHz). 10: Reserved. 11: Reserved.
<b>RESERVED</b>	11, 15:14	r	Reserved; these bits must be left at their reset values.

CONFIDENTIAL

Bi-directional Serial Audio Interface I2S1 (4-pins)

### 8.4.9.5 Divider 0 Denominator Register

I2Sx\_DEN0

I2Sx\_DEN0 sets the denominator of the Fractional Divider 0.

I2S1\_DEN0

I2S2\_DEN0

Divider 0 Denominator Register

Reset value: 0002<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DENOMINATOR															

Field	Bits	Type	Description
DENOMINATOR	15:0	rw	<b>Denominator value</b> Determines the denominator value of the fractional divider 0 described in <a href="#">Section 8.4.7 Clock Configuration (on Page 385)</a> .  <i>Note: The values 0 and 1 are not permitted and lead to undefined results.</i>

*Note: Please keep in mind that for all  $(4 \times \text{denominator})/\text{numerator}$  ratios which do not result in an integer the bit clock will have a jitter of one clock period of the reference clock. Ratios of numerator/denominator which are larger or equal to 0.5 are not allowed.*

**CONFIDENTIAL**

**Bi-directional Serial Audio Interface I2S1 (4-pins)**

### 8.4.9.6 Divider 1 Numerator Register

#### I2Sx\_NUM1

The firmware uses **I2Sx\_NUM1** to set the numerator of the fractional divider 1 and select the reference frequency of the transmit baud rate generator.

#### I2S1\_NUM1

#### I2S2\_NUM1

#### Divider 1 Numerator Register

Reset value: 0001<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		FREF		RESERVED	NUMERATOR										

Field	Bits	Type	Description
<b>NUMERATOR</b>	10:0	rw	<b>Numerator</b> Determines the numerator value of the fractional divider described in <a href="#">Section 8.4.7 Clock Configuration (on Page 385)</a> . <i>Note: The value 0 is not permitted and leads to undefined results.</i>
<b>FREF</b>	13:12	rw	<b>FREF selection</b> 00: Reference input 0 (Module clock). 01: Reference input 1 (104 MHz). 10: Reserved. 11: Reserved.
<b>RESERVED</b>	11, 15:14	r	Reserved; these bits must be left at their reset values.

CONFIDENTIAL

Bi-directional Serial Audio Interface I2S1 (4-pins)

### 8.4.9.7 Divider 1 Denominator Register

I2Sx\_DEN1

I2Sx\_DEN1 sets the denominator of the fractional divider 1.

I2S1\_DEN1

I2S2\_DEN1

Divider 1 Denominator Register

Reset value: 0002<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DENOMINATOR															

Field	Bits	Type	Description
DENOMINATOR	15:0	rw	<b>Denominator value</b> Determines the denominator value of the fractional divider 1 described in <a href="#">Section 8.4.7</a> . <i>Note: The values 0 and 1 are not permitted and lead to undefined results.</i>

*Note: Please keep in mind that for all  $(4 \times \text{denominator})/\text{numerator}$  ratios which do not result in an integer the bit clock will have a jitter of one clock period of the reference clock. Ratios of numerator/denominator which are larger or equal to 0.5 are not allowed.*

**CONFIDENTIAL**

**Bi-directional Serial Audio Interface I2S1 (4-pins)**

## 8.4.10 I2S Receiver Registers

### 8.4.10.1 Receiver Configuration Register

**I2Sx\_RXCONF**

**I2S1\_RXCONF**

**I2S2\_RXCONF**

**Receiver Configuration Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>WA1 _LE _N</b>	<b>CLK 1_C ONT</b>	<b>CLK 1_O UT</b>	<b>RESERVED</b>				<b>ALIG N</b>		<b>WIDTH</b>		<b>PERIOD</b>	<b>POL</b>	<b>DEL</b>	<b>EDG E</b>	

Field	Bits	Type	Description
<b>EDGE</b>	0	rw	<b>Edge for sampling of received data in normal mode</b> 0: Falling edge 1: Rising edge  <i>Note: To ensure proper operation in normal mode the sampling edge at the receiver should be set complementary to the output edge at the transmitter.</i>
<b>DEL</b>	1	rw	<b>Delay of first bit in normal mode</b> 0: New word starts at edge of WA. 1: New word starts one clock after edge of WA.
<b>POL</b>	2	rw	<b>Polarity of WA signal in normal mode</b> 0: WA = 1 indicates right channel; WA = 0 indicates left channel. 1: WA = 1 indicates left channel; WA = 0 indicates right channel.
<b>PERIOD</b>	4:3	rw	<b>Frame period in normal mode</b> 00: 64 clocks 01: 48 clocks 10: 32 clocks 11: Reserved

**CONFIDENTIAL**

**Bi-directional Serial Audio Interface I2S1 (4-pins)**

Field	Bits	Type	Description
<b>WIDTH</b>	7:5	rw	<b>Sample width in normal mode</b> 000: 16 bits 001: Reserved 010: Reserved 011: Reserved 100: 18 bits 101: 20 bits 110: 24 bits 111: 32 bits
<b>ALIGN</b>	8	rw	<b>Data alignment in normal mode</b> 0: Left aligned 1: Right aligned
<b>CLK1_OUT</b>	13	rw	<b>CLK1 burst output level in PCM mode</b> This bit is only used when <b>CLKx_COUNT</b> is zero 0: i2sx_clk1 is low 1: i2sx_clk1 is high The setting of <b>CLK1_OUT</b> takes effect as soon as bit <b>i2sx_CTRL.RXPCM</b> is set.
<b>CLK1_COUNT</b>	14	rw	<b>CLK1 burst behavior in PCM mode</b> This bit determines the behavior of the CLK1 line when the reception is stopped (before first burst and between consecutive bursts). 0: i2sx_clk1 is stopped and switched to a fixed value determined by bit <b>CLK1_OUT</b> . 1: i2sx_clk1 remains running
<b>WA1_LEN</b>	15	rw	<b>WA1 pulse length in PCM mode</b> 0: i2sx_wa1 is high for 1 cycle of i2sx_clk1 1: i2sx_wa1 is high for 2 cycles of i2sx_clk1
<b>RESERVED</b>	12:9	r	Reserved; these bits must be left at their reset values.



**CONFIDENTIAL**

**Bi-directional Serial Audio Interface I2S1 (4-pins)**

### 8.4.10.2 Receiver Interrupt Address Register

**I2Sx\_RXINTADDR**

**I2S1\_RXINTADDR**

**I2S2\_RXINTADDR**

**Receiver Interrupt Address Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED										RXINTPTR					

Field	Bits	Type	Description
<b>RXINTPTR</b>	5:0	rw	<b>Receiver Interrupt Address</b>
<b>RESERVED</b>	15:6	r	Reserved; these bits must be left at their reset values.

### 8.4.11 I2S Transmitter Registers

#### 8.4.11.1 Transmitter Configuration Register

**I2Sx\_TXCONF**

**I2S1\_TXCONF**

**I2S2\_TXCONF**

**Transmitter Configuration Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WA0 _LE N	CLK 0_C ONT	CLK 0_O UT	MUT E_R	MUT E_L	MONO	ALIGN	WIDTH	PERIOD	POL	DEL	EDGE				

Field	Bits	Type	Description
<b>EDGE</b>	0	rw	<b>Edge for output of transmit data in normal mode</b> 0: Falling edge 1: Rising edge  <i>Note: To ensure proper operation in normal mode the sampling edge at the receiver should be set complementary to the output edge at the transmitter.</i>

**CONFIDENTIAL**

**Bi-directional Serial Audio Interface I2S1 (4-pins)**

Field	Bits	Type	Description
<b>DEL</b>	1	rw	<b>Delay of first bit in normal mode</b> 0: New word starts at edge of WA. 1: New word starts one clock after edge of WA.
<b>POL</b>	2	rw	<b>Polarity of WA signal in normal mode</b> 0: WA = 1 indicates right channel. 1: WA = 1 indicates left channel.
<b>PERIOD</b>	4:3	rw	<b>Frame period in normal mode</b> 00: 64 clocks 01: 48 clocks 10: 32 clocks 11: Reserved  <i>Note: This value is only used in master mode. In slave mode the receiver detects the frame length and adjusts its input accordingly after 1-2 frames.</i>
<b>WIDTH</b>	7:5	rw	<b>Sample width in normal mode</b> 000: 16 bits 001: Reserved 010: Reserved 011: Reserved 100: 18 bits 101: 20 bits 110: 24 bits 111: 32 bits
<b>ALIGN</b>	8	rw	<b>Data alignment in normal mode</b> 0: Left aligned 1: Right aligned  In right aligned mode, unused bits in front of the transmitted data are set to the value of the MSB to be transmitted (two's complement sign extension).
<b>MONO</b>	10:9	rw	<b>Transfer mode in normal mode</b> 00: Stereo mode. 01: Reserved. 10: Right data transmitted on both channels. 11: Left data transmitted on both channels.
<b>MUTE_L</b>	11	rw	<b>Mute left channel in normal mode</b> 0: Channel active. 1: Left channel muted.

**CONFIDENTIAL**

**Bi-directional Serial Audio Interface I2S1 (4-pins)**

Field	Bits	Type	Description
<b>MUTE_R</b>	12	rw	<b>Mute right channel in normal mode</b> 0: Channel active. 1: Right channel muted.
<b>CLK0_OUT</b>	13	rw	<b>CLK0 burst output level in PCM mode</b> This bit is only used when <b>CLK0_CONT</b> is zero 0: i2sx_clk0 is low 1: i2sx_clk0 is high The setting of <b>CLK0_OUT</b> takes effect as soon as bit <b>I2Sx_CTRL.TXPCM</b> is set.
<b>CLK0_CONT</b>	14	rw	<b>CLK0 burst behavior in PCM mode</b> This bit determines the behavior of the CLK0 line when the transmission is stopped (before first burst and between consecutive bursts). 0: i2sx_clk0 is stopped and switched to a fixed value determined by bit <b>CLK0_OUT</b> . 1: i2sx_clk0 remains running
<b>WA0_LEN</b>	15	rw	<b>WA0 pulse length in PCM mode</b> 0: i2sx_wa0 is high for 1 cycle of i2sx_clk0 1: i2sx_wa0 is high for 2 cycles of i2sx_clk0

#### 8.4.11.2 Transmitter Interrupt Address Register

**I2Sx\_TXINTADDR**

**I2S1\_TXINTADDR**

**I2S2\_TXINTADDR**

**Transmitter Interrupt Address Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>										<b>TXINTPTR</b>					

Field	Bits	Type	Description
<b>TXINTPTR</b>	5:0	rw	<b>Transmitter Interrupt Address</b>
<b>RESERVED</b>	15:6	r	Reserved; these bits must be left at their reset values.

CONFIDENTIAL

**CONFIDENTIAL**

**Bi-directional Serial Audio Interface I2S2 (6-pins)**

## 8.5 Bi-directional Serial Audio Interface I2S2 (6-pins)

History	
Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.04	
<b>Page 417</b>	Heading "I2S 2 (Audio)" changed to <b>Bi-directional Serial Audio Interface I2S2 (6-pins)</b> WS00007943
Changes for Rev. 1.06	

### System Integration on TEAKLite

- Supply domain: VDD\_DSP
- Chip internal interfaces:
  - Clock domain:  
Peripheral kernel: gclk\_dsp\_per  
Clocks for baud rate generation: gclk\_pll, gclk\_dsp\_per
  - Bus domain: TEAKLite Z-Bus
  - Interrupt sources:  
I2S2\_TX\_INT, I2S2\_RX\_INT
- Chip external signals related to this block (refer to [Chapter 3 Pin Descriptions \(on Page 49\)](#) for pin configuration options):  
I2S2\_CLK0, I2S2\_WA0, I2S2\_TX,  
I2S3\_CLK, I2S3\_WA, I2S2\_RX
- Monitor pins: Refer to [Section 11.8.10 Internal Signal Monitoring \(on Page 1209\)](#)

Refer to [Section 8.4 Bi-directional Serial Audio Interface I2S1 \(4-pins\) \(on Page 375\)](#) for information about the I2S 2 in the 4-pin configuration.

### 6-Pin Configuration

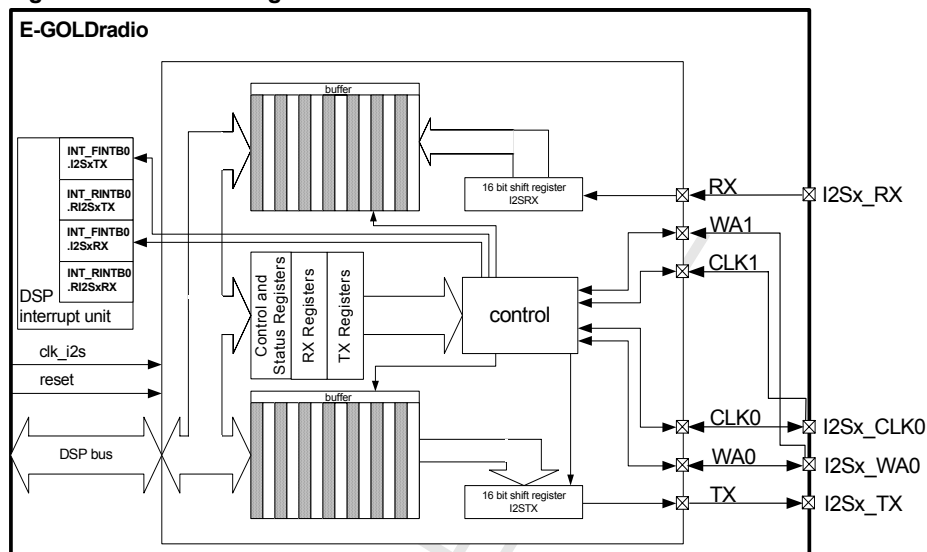
The I2S 2 interface can be configured with 6 pins allowing full control of the RX and TX paths. The I2S3\_WA and I2S3\_clk chip pads are connected to the I2S 2 block WA1 and CLK1 pins using the register **PCL\_<pad>** (**PCL\_30** and **PCL\_28**, refer to [Table 3-1 Pin List \(on Page 51\)](#)).

A block diagram of the bi-directional I2S 2 6-pin interface is shown in [Figure 8-19](#).

CONFIDENTIAL

Bi-directional Serial Audio Interface I2S2 (6-pins)

Figure 8-19 Block Diagram of Bi-Directional I2S 2 6-Pin Interface



When in the 6-pin configuration, the bi-directional I<sup>2</sup>S 2 module has the signals in [Table 8-16](#).

Table 8-16 I2Sx Signals

	Signal Name	Function
TX I2S	i2sx_clk0	Transmit clock
	i2sx_wa0	Transmit word select
	i2sx_tx	Serial output audio data
RX I2S	i2sx_clk1 <sup>1)</sup>	Receive clock
	i2sx_wa1 <sup>1)</sup>	Receive word select
	i2sx_rx	Serial input audio data

<sup>1)</sup> This signal is available outside the chip when in the 6-pin configuration.

**Note:** When the I<sup>2</sup>S 2 in the 6-pin configuration, the I<sup>2</sup>S 3 cannot be used because two of its pads are used by the I<sup>2</sup>S 2.

**CONFIDENTIAL**

**Unidirectional Serial Audio Interface I2S3**

## 8.6 Unidirectional Serial Audio Interface I2S3

History	
Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.02	
<b>Page 421</b>	Updated <b>Section 8.6.3 Operating Mode Generalities</b> WS00005886
Changes for Rev. 1.04	
<b>Page 419</b>	Heading "I2S 3 (Unidirectional)" changed to <b>Unidirectional Serial Audio Interface I2S3</b> WS00007944
Changes for Rev. 1.06	

### System Integration on TEAKLite:

- Supply domain: VDD\_DSP
- Chip internal interfaces:
  - Clock domain:  
Peripheral kernel: gclk\_dsp\_per  
Clock for baud rate generation: gclk\_pll, gclk\_dsp\_per
  - Bus domain: TEAKLite Z-Bus
  - Interrupt source: I2S3\_TX\_INT
- Chip external signals related to this block (refer to **Chapter 3 Pin Descriptions (on Page 49)** for pin configuration options):  
I2S3\_CLK, I2S3\_WA, I2S3\_TX
- Monitor pins: Refer to **Section 11.8.10 Internal Signal Monitoring (on Page 1209)**

### 8.6.1 Functional Overview

The unidirectional I<sup>2</sup>S-Interface is implemented to transmit data to an external modulator.

#### General features of the I2S3 Interface

- 2 Supported modes: Normal mode and PCM mode
- 32 bit data path
- Interface can operate in master and slave mode
- 64 word data buffer transmitter
- Fractional divider for bit clock generation in master mode

**CONFIDENTIAL**

**Unidirectional Serial Audio Interface I2S3**

- Adjustable audio sampling rate (for example, 48 kHz, 44.1 kHz, 32 kHz, 24 kHz, 22.05 kHz, 16 kHz, 12 kHz, 11.025 kHz and 8 kHz).

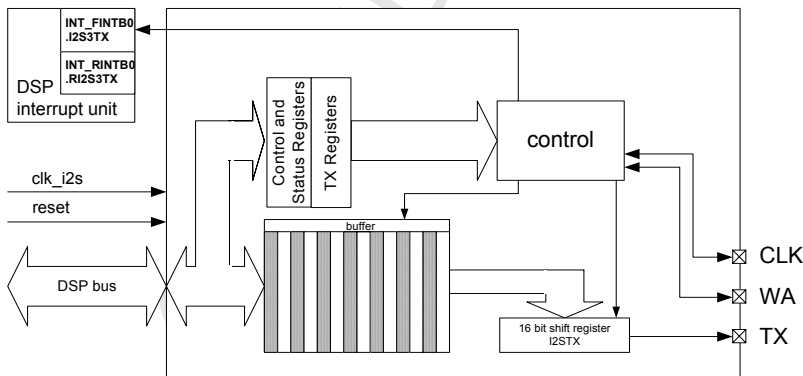
**Mode dependant features of the I2S3 Interface**

- Normal mode:
  - Supports delayed and non delayed WA mode
  - Configurable to different word lengths (16, 18, 20, 24 and 32 bit) and different frame lengths (64, 48 and 32 bit)
  - Switch of left/right order in frame and left/right alignment of data in frame
  - Data valid on rising or falling edge of bit clock
- PCM mode:
  - Supports 16 bit mode with short frame synchronization
  - Supports both burst mode and continuous transmission
  - Length of WA pulse selectable between one and two clock cycles
  - CLK line is configurable to be either low, high or to remain running when no transmission takes place

**8.6.2 Structural Overview**

A block diagram of the unidirectional I2S interface hardware is shown in **Figure 8-20**.

**Figure 8-20 Block Diagram of Unidirectional I2S Interface Hardware**





**CONFIDENTIAL**

## Unidirectional Serial Audio Interface I2S3

The unidirectional I2S module is a 3-wire master interface with the following signals:

**Table 8-17 I<sup>2</sup>S 3 Signals**

	Signal Name	Function
I <sup>2</sup> S output	i2s3_clk	I <sup>2</sup> S transmit clock
	i2s3_wa	Transmit word select
	i2s3_tx	Serial output audio data

### 8.6.3 Operating Mode Generalities

Regardless of the chosen operating mode, the following considerations apply to the I2S interface.

To set up a data transmission using the serial interface, first set **I2S3\_CTRL.I2SON**. This resets the internal state of the interface. In particular the read pointer, internal state machines and counters are reset. The control registers are not affected when **I2SON** is set.

After this all necessary configurations like mode setting, clock source selection, etc. have to be made. Then the first set of audio samples must then be written to the interface buffer. Finally, by setting the start bit **I2S3\_CTRL.I2SxTXSTART** the data transmission can be started.

The transfer can stop at two different conditions:

- Automatic stop when interrupt address is reached (PCM mode only)
- Start bit **I2STXSTART** is reset by software.

For the automatic stop in PCM mode, the interface stops as soon as the interrupt address defined in **I2S3\_TXINTADDR** is reached. The last sample transferred is stored at the address just before the transmit interrupt address. An I2SxTX interrupt is generated when the transfer of the last sample begins.

For stopping the interface by software the bit **I2S3TXSTART** can be reset. This will end transmission or reception as soon as the transfer of the current frame has been finished.

In order to change the configuration settings the following sequence is required:

- Stop interface either automatically or by resetting the start bit
- Switch bit **I2SON** to 0 to reset internal states
- Set bit **I2SON** to 1 again
- Change configuration
- Restart data transfer by setting the bit **I2STXSTART**.

After restarting data transmission by setting bit **I2STXSTART**, the **I2S3\_RADDR.RDADDR** pointer is incremented. This means that the next audio sample to be sent is taken from the buffer position following the one of the last transmitted audio sample. The first audio sample transmitted belongs to the left channel.

CONFIDENTIAL

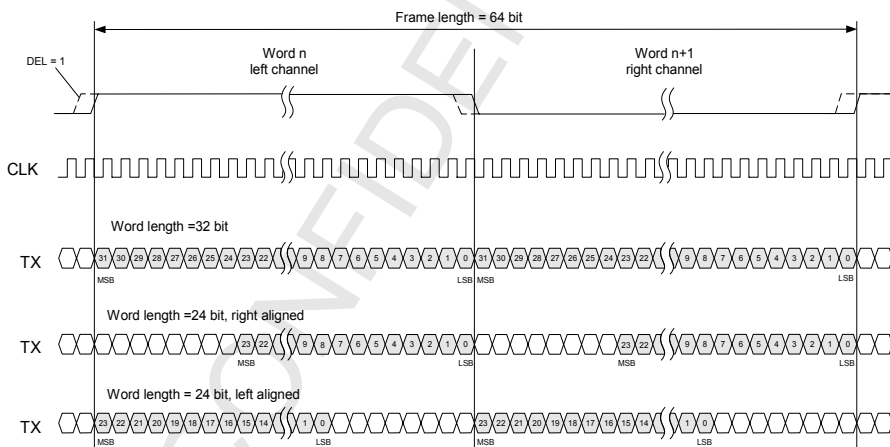
Unidirectional Serial Audio Interface I2S3

The I2S interface operating as master in normal mode supports all data rates which can be derived by dividing the reference frequency by a fractional divider (e.g. 48 kHz, 44.1 kHz, 32 kHz, 24 kHz, 22.05 kHz, 16 kHz, 12 kHz, 11.025 kHz, and 8 kHz are possible when a 104 MHz reference frequency is supplied. Refer to [Table 8-22](#) and the following tables for an overview.

### 8.6.4 Serial Audio Data Transmission in Normal Mode

[Figure 8-21](#) gives the basic timing of the I2S signals in normal mode. It shows combinations of a 64-bit frame with 32-bit words, 24-bit words right aligned and 24-bit words left aligned as examples. Transmission in normal mode is supported in master as well as in slave operation. Serial audio data is transmitted with the most significant bit first. In this example data is launched at the transmitting side with the falling edge of I2s3\_clk and latched at the receiving side with the rising edge of I2s3\_clk. Signal i2s3\_wa is used to select between left and right channel and to determine the start of word.

**Figure 8-21 Principal Timing of I2S Interface Signals (Normal Mode)**



For use with low-cost digital-to-analog converters which have a simplified I2S interface the bit DEL in receive and transmit configuration register can be set. This results in a delay of one bit clock for the signal i2s3\_wa with respect to the signal bits, thus providing the edge of i2s3\_wa with the MSB of each sample (see [Figure 8-21](#)).

### 8.6.5 PCM Mode Operation of I2S interface

The I2S interface can also be set up in a mode that allows PCM data transfers to other devices (e.g. PMB 6752 Bluemoon I). In this mode short frame synchronization with 16 bit frame length is supported. Both burst and continuous transfers are possible. PCM

CONFIDENTIAL

Unidirectional Serial Audio Interface I2S3

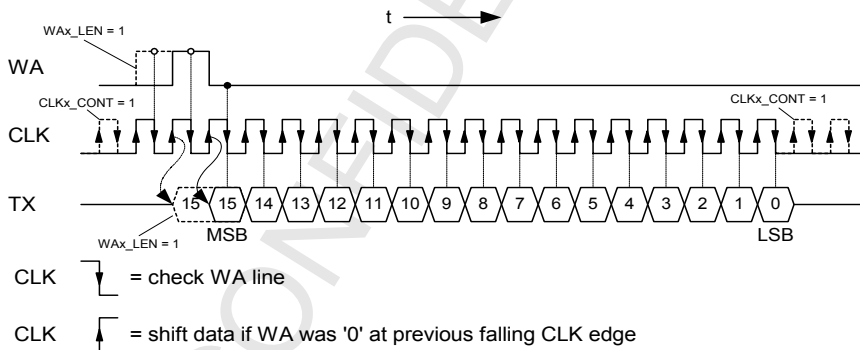
mode supports master and slave operation. The PCM mode transmission is selected by setting the bit **I2S3\_CTRL.TXPCM** in the control register. The main difference between this mode and the normal I2S mode is that the signal `i2s3_wa` now serves as a frame synchronization signal and that the data transfer does not need to be continuous. A timing diagram shown in **Figure 8-22**.

The data transmit operation works as follows: with the falling edge of signal `i2s3_wa` the transmission of a data sample starts with the most significant bit. With each rising edge of `i2s3_clk` a new data bit starts. As soon as all 16 bits have been transmitted a new data sample is transferred immediately from the TX buffer to the shift register and the **I2S3\_RADDR.RDADDR** is incremented by one. Thus, a continuous data transfer can be achieved.

The length of the `i2s3_wa` pulse can be configured to be either one or two cycles of `i2s3_clk`. It is possible to configure whether I2S3\_CLK should be running or switched either to low or high when no transmission takes place.

The I2S interface in PCM mode operating as master supports bit clock rates, which can be divided down from the reference frequency by a fractional divider. For details on reference clock calculation see section on Clock Configuration.

**Figure 8-22 Timing of I2S Interface Signals in PCM Mode**



**Figure 8-23** shows the timing of two PCM burst transfers, starting with the initialization of the interface.

After **I2S3\_CTRL.I2SON** has been set the control bits **I2S3\_CSEL.CLK\_CONT** and **I2S3\_CSEL.CLK\_OUT** can be configured. These bits define the behavior of the I2S3\_CLK line when no transmission takes place.

**CLK\_CONT** determines whether the clock should be switched off (**CLK\_CONT** = 0) or remain running (**CLK\_CONT** = 1) between two consecutive bursts. When the interface is initialized after I2SON has been set the `i2s3_clk` line is always idle until **I2S3\_CTRL.TXSTART** is set. This behavior cannot be changed by the **CLK\_CONT** bit.

**CONFIDENTIAL**

**Unidirectional Serial Audio Interface I2S3**

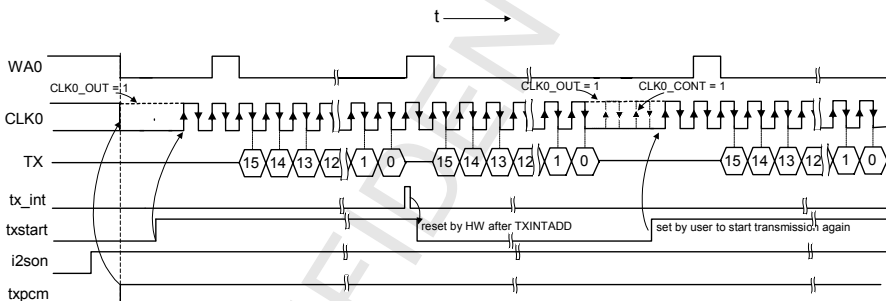
The bit **CLK\_OUT** selects the level of i2s3\_clk if the serial clock is idle. The clock can be idle in two cases:

- After **I2SON** is set the clock is always inactive. In this state **CLK\_OUT** can be configured. It takes effect as soon as **I2S3\_CTRL.TXPCM** is set. i2s3\_clk remains in this state until **TXSTART** is set and the transmission of the first frame starts.
- If the transmission is stopped at the end of a burst the clock line will be idle when **CLK\_CONT** has been set to zero during initialization.

In the given example the transfer is stopped based on a given interrupt level. **TXSTART** is reset as soon as TXINT occurs, but the frame referenced by the interrupt address is sent before the transmission finally stops.

After storing new data in the transmit buffer the transfer can be continued by setting **TXSTART** again.

**Figure 8-23 Timing of I2S Interface Signals in PCM Mode with Burst Transmission**



**CONFIDENTIAL**

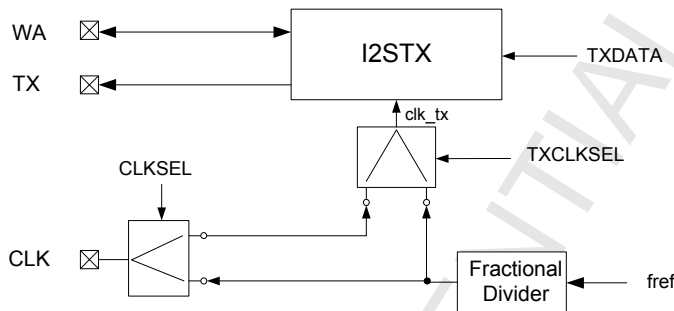
**Unidirectional Serial Audio Interface I2S3**

## 8.6.6 Clock Configuration

### 8.6.6.1 Overview

Master or slave mode operation of the I2S interface can be selected by setting the clock source of the I2S interface in the register **I2S3\_CSEL**. **Figure 8-24** shows the clock source selection and distribution.

**Figure 8-24 Block Diagram of Clock Source Selection and Distribution for I2S3**



The direction of the word align signal **i2s3\_wa** is defined by the direction of the clock signal **i2s3\_clk**. The signals **i2s3\_wa** and **i2s3\_clk** always have the same direction, which is defined by the field **I2S3\_CSEL.CLKSEL** according to **Table 8-18**.

**Table 8-18 Selection of I2S3\_CLK for I2S3**

CLKSEL	i2s3_clk Source	Direction of i2s3_clk	Direction of i2s3_wa	Operation Mode
0	Fractional Divider 0	output	output	master
1	external	input	input	slave

### 8.6.6.2 Slave Mode

If the I2S3 is to be operated in slave mode, the clock source during this mode is external to PMB7870.

*Note: In slave operation the maximum external bit clock rate is 1/4 of the internal module clock.*

### 8.6.6.3 Master Mode

The I2S3 interface configured as master supports independent bit clock rates for reception and transmission and therefore provides two fractional dividers. To achieve the desired audio sampling rate the fractional divider must be programmed to generate the

**CONFIDENTIAL**

**Unidirectional Serial Audio Interface I2S3**

appropriate bit clock rate. This bit clock rate can be calculated by multiplying the sampling rate with the frame length.

In normal mode the frame length can be either 32, 48 or 64 bits. In contrast with this the frame length in PCM mode does not correspond with the number of bits transmitted. Instead, one or two additional clock cycles are required for the WA signal. This leads to a frame length 17 or 18 bits in PCM mode, depending on the chosen length for the WA signal.

The resulting frequency of the bit clock can be calculated according to the following equation.

$$f_{CLK} = \frac{f_{ref} \cdot \text{NUMERATOR}}{4 \cdot \text{DENOMINATOR}} \quad [0.8]$$

The NUMERATOR value can be set in the registers **I2S3\_NUM**. The DENOMINATOR value can be set in the registers **I2S3\_DEN**.

*Note: The maximum possible frequency of the bit clock in is 1/8 of the reference frequency.*

The clock signal is derived from the reference clock, which is selected by the bit field **I2S3\_NUM.FREF** according to **Table 8-19**.

**Table 8-19 Selection of Reference Clock**

<b>FREF[1:0]</b>	<b>Reference clock (fref 0/1)</b>
00	Module clock (clk_dsp)
01	104 MHz clock (clk_ms)
10	Reserved
11	Reserved

*Note: The module clock is running at the same frequency as the DSP and is therefore not fixed. If the module clock is used as reference the fractional divider settings need to be adapted each time the DSP clock frequency changes.*

### Example Clock Configurations

Examples for usual sampling rates in master mode are 48 kHz, 44.1 kHz, 32 kHz, 24 kHz, 22.05 kHz, 16 kHz, 12 kHz, 11.025 kHz, and 8 kHz. The following tables show the numerator/denominator values required to select those frequencies in PCM and normal mode.

In the PCM mode each frame contains 16 bits of data. Taking into account one or two cycles for the pulse on the WA line this results in a frame length of 17 or 18 shift clock cycles as given in the tables below. Furthermore, the tables contain the settings for 32, 48 and 64 cycle frame lengths, which can be selected in normal mode.

**CONFIDENTIAL**

## Unidirectional Serial Audio Interface I2S3

**Table 8-20** shows the corresponding settings of the numerator and the denominator values for a 89.1 MHz reference clock. In the given configuration the (4\*denominator)/numerator ratio always results in an integer. This way the fractional divider will not produce additional jitter, but at the cost of higher frequency deviations.

**Table 8-20 Bit Clock Rates with 89.1 MHz Reference and Integer Ratio of Fractional Divider (Minimum Jitter)**

Nominal Audio Sampling Rates	Frame Length <sup>1)</sup>	Nominal Bit Clock	NUM	DEN	Real Bit Clock	Bit Clock Deviation
8.000 kHz	17	136.00 kHz	4	655	136.03 kHz	+ 0.02%
	18	144.00 kHz	4	619	143.94 kHz	-0.04%
	32	256.00 kHz	4	348	256.03 kHz	+ 0.01%
	48	384.00 kHz	4	232	384.05 kHz	+ 0.01%
	64	512.00 kHz	4	174	512.07 kHz	+ 0.01%
11.025 kHz	17	187.43 kHz	4	475	187.58 kHz	+ 0.08%
	18	198.45 kHz	4	449	198.44 kHz	- 0.00%
	32	352.80 kHz	4	253	352.17 kHz	- 0.18%
	48	529.20 kHz	4	168	530.36 kHz	+ 0.22%
	64	705.60 kHz	4	126	707.14 kHz	+ 0.22%
12.000 kHz	17	204.00 kHz	4	437	203.89 kHz	- 0.05%
	18	216.00 kHz	4	413	215.74 kHz	- 0.12%
	32	384.00 kHz	4	232	384.05 kHz	+ 0.01%
	48	576.00 kHz	4	155	574.84 kHz	- 0.20%
	64	768.00 kHz	4	116	768.10 kHz	+ 0.01%
16.000 kHz	17	272.00 kHz	4	328	271.65 kHz	- 0.13%
	18	288.00 kHz	4	309	288.35 kHz	+ 0.12%
	32	512.00 kHz	4	174	512.07 kHz	+ 0.01%
	48	768.00 kHz	4	116	768.10 kHz	+ 0.01%
	64	1024.00 kHz	4	87	1024.14 kHz	+ 0.01%

**CONFIDENTIAL**

**Unidirectional Serial Audio Interface I2S3**

**Table 8-20 Bit Clock Rates with 89.1 MHz Reference and Integer Ratio of Fractional Divider (Minimum Jitter)**

Nominal Audio Sampling Rates	Frame Length <sup>1)</sup>	Nominal Bit Clock	NUM	DEN	Real Bit Clock	Bit Clock Deviation
22.050 kHz	17	374.85 kHz	4	238	374.37 kHz	- 0.13%
	18	396.90 kHz	4	224	397.77 kHz	+ 0.22%
	32	705.60 kHz	4	126	707.14 kHz	+ 0.22%
	48	1058.40 kHz	4	84	1060.71 kHz	+ 0.22%
	64	1411.20 kHz	4	63	1414.29 kHz	+ 0.22%
24.000 kHz	17	408.00 kHz	4	218	408.72 kHz	+ 0.18%
	18	432.00 kHz	4	206	432.52 kHz	+ 0.12%
	32	768.00 kHz	4	116	768.10 kHz	+ 0.01%
	48	1152.00 kHz	4	77	1157.14 kHz	+ 0.45%
	64	1536.00 kHz	4	58	1536.21 kHz	+ 0.01%
32.000 kHz	17	544.00 kHz	4	164	543.29 kHz	- 0.13%
	18	576.00 kHz	4	155	574.84 kHz	- 0.20%
	32	1024.00 kHz	4	87	1024.14 kHz	+ 0.01%
	48	1536.00 kHz	4	58	1536.21 kHz	+ 0.01%
	64	2048.00 kHz	4	44	2025 kHz	- 1.12%
44.100 kHz	17	749.70 kHz	4	119	748.74 kHz	- 0.13%
	18	793.80 kHz	4	112	795.54 kHz	+ 0.22%
	32	1411.20 kHz	4	63	1414.29 kHz	+ 0.22%
	48	2116.80 kHz	4	42	2121.43 kHz	+ 0.22%
	64	2822.40 kHz	4	32	2784.38 kHz	- 1.35%
48.000 kHz	17	816.00 kHz	4	109	817.43 kHz	+ 0.18%
	18	864.00 kHz	4	103	865.05 kHz	+ 0.12%
	32	1536.00 kHz	4	58	1536.21 kHz	+ 0.01%
	48	2304.00 kHz	4	39	2284.62 kHz	- 0.84%
	64	3072.00 kHz	4	29	3072.41 kHz	+ 0.01%

<sup>1)</sup> Number of shift clock cycles



**CONFIDENTIAL**

## Unidirectional Serial Audio Interface I2S3

**Table 8-21** shows the settings of the numerator and the denominator values for a 89.1 MHz reference clock. In the given configuration the  $(4 \times \text{denominator}) / \text{numerator}$  ratio does not need to result in an integer. This way the fractional divider will produce additional jitter, but the given values result in the minimum possible bit clock deviation.

**Table 8-21 Bit Clock Rates with 89.1 MHz Reference and Non-Integer Ratio of Fractional Divider (Minimum Bit Clock Deviation)**

Nominal Audio Sampling Rates	Frame Length <sup>1)</sup>	Nominal Bit Clock	NUM	DEN	Real Bit Clock	Bit Clock Deviation
8.000 kHz	17	136.00 kHz	136	22275	136.00 kHz	+0.00%
	18	144.00 kHz	16	2475	144.00 kHz	+0.00%
	32	256.00 kHz	256	22275	256.00 kHz	+0.00%
	48	384.00 kHz	128	7425	384.00 kHz	+0.00%
	64	512.00 kHz	512	22275	512.00 kHz	+0.00%
11.025 kHz	17	187.43 kHz	302	35891	187.43 kHz	+0.00%
	18	198.45 kHz	49	5500	198.45 kHz	-0.00%
	32	352.80 kHz	196	12375	352.80 kHz	+0.00%
	48	529.20 kHz	98	4125	529.20 kHz	+0.00%
	64	705.60 kHz	392	12375	705.60 kHz	+0.00%
12.000 kHz	17	204.00 kHz	68	7425	204.00 kHz	+0.00%
	18	216.00 kHz	8	825	216.00 kHz	+0.00%
	32	384.00 kHz	128	7425	384.00 kHz	+0.00%
	48	576.00 kHz	64	2475	576.00 kHz	+0.00%
	64	768.00 kHz	256	7425	768.00 kHz	+0.00%
16.000 kHz	17	272.00 kHz	272	22275	272.00 kHz	+0.00%
	18	288.00 kHz	32	2475	288.00 kHz	+0.00%
	32	512.00 kHz	512	22275	512.00 kHz	+0.00%
	48	768.00 kHz	256	7425	768.00 kHz	+0.00%
	64	1024.00 kHz	1024	22275	1024.00 kHz	+0.00%

**CONFIDENTIAL**

**Unidirectional Serial Audio Interface I2S3**

**Table 8-21 Bit Clock Rates with 89.1 MHz Reference and Non-Integer Ratio of Fractional Divider (Minimum Bit Clock Deviation)**

Nominal Audio Sampling Rates	Frame Length <sup>1)</sup>	Nominal Bit Clock	NUM	DEN	Real Bit Clock	Bit Clock Deviation
22.050 kHz	17	374.85 kHz	833	49500	374.85 kHz	+0.00%
	18	396.90 kHz	49	2750	396.90 kHz	-0.00%
	32	705.60 kHz	392	12375	705.60 kHz	+0.00%
	48	1058.40 kHz	196	4125	1058.40 kHz	+0.00%
	64	1411.20 kHz	784	12375	1411.20 kHz	+0.00%
24.000 kHz	17	408.00 kHz	136	7425	408.00 kHz	+0.00%
	18	432.00 kHz	16	825	432.00 kHz	+0.00%
	32	768.00 kHz	256	7425	768.00 kHz	+0.00%
	48	1152.00 kHz	128	2475	1152.00 kHz	+0.00%
	64	1536.00 kHz	512	7425	1536.00 kHz	+0.00%
32.000 kHz	17	544.00 kHz	544	22275	544.00 kHz	+0.00%
	18	576.00 kHz	64	2475	576.00 kHz	+0.00%
	32	1024.00 kHz	939	20426	1024.00 kHz	+0.00%
	48	1536.00 kHz	512	7425	1536.00 kHz	+0.00%
	64	2048.00 kHz	1109	12062	2048.00 kHz	+0.00%
44.100 kHz	17	749.70 kHz	833	24750	749.70 kHz	+0.00%
	18	793.80 kHz	49	1375	793.80 kHz	+0.00%
	32	1411.20 kHz	1024	22275	1411.20 kHz	+0.00%
	48	2116.80 kHz	392	4125	2116.80 kHz	+0.00%
	64	2822.40 kHz	1568	12375	2822.40 kHz	-0.00%
48.000 kHz	17	816.00 kHz	272	7425	816.00 kHz	+0.00%
	18	864.00 kHz	32	825	864.00 kHz	+0.00%
	32	1536.00 kHz	512	7425	1536.00 kHz	+0.00%
	48	2304.00 kHz	256	2475	2304.00 kHz	+0.00%
	64	3072.00 kHz	1024	7425	3072.00 kHz	+0.00%

<sup>1)</sup> Number of shift clock cycles

**CONFIDENTIAL**

## Unidirectional Serial Audio Interface I2S3

**Table 8-22** shows the settings of the numerator and the denominator values for a 104 MHz reference clock. In the given configuration the (4\*denominator)/numerator ratio always results in an integer. This way the fractional divider will not produce additional jitter, but at the cost of higher frequency deviations.

**Table 8-22 Bit Clock Rates with 104 MHz Reference and Integer Ratio of Fractional Divider (Minimum Jitter)**

Nominal Audio Sampling Rates	Frame Length <sup>1)</sup>	Nominal Bit Clock	NUM	DEN	Real Bit Clock	Bit Clock Deviation
8.000 kHz	17	136.00 kHz	4	762	135.95 kHz	- 0.04%
	18	144.00 kHz	4	722	144.04 kHz	+ 0.03%
	32	256.00 kHz	4	406	255.16 kHz	+ 0.06%
	48	384.00 kHz	4	271	383.76 kHz	- 0.06%
	64	512.00 kHz	4	203	512.32 kHz	+ 0.06%
11.025 kHz	17	187.43 kHz	4	555	187.39 kHz	- 0.02%
	18	198.45 kHz	4	524	198.47 kHz	+ 0.01%
	32	352.80 kHz	4	295	352.54 kHz	- 0.07%
	48	529.20 kHz	4	197	527.92 kHz	- 0.24%
	64	705.60 kHz	4	147	707.48 kHz	+ 0.27%
12.000 kHz	17	204.00 kHz	4	510	203.92 kHz	+ 0.04%
	18	216.00 kHz	4	481	216.22 kHz	+ 0.10%
	32	384.00 kHz	4	271	383.76 kHz	- 0.06%
	48	576.00 kHz	4	181	576.58 kHz	- 0.25%
	64	768.00 kHz	4	135	770.37 kHz	+ 0.31%
16.000 kHz	17	272.00 kHz	4	382	272.25 kHz	+ 0.09%
	18	288.00 kHz	4	361	288.09 kHz	+ 0.03%
	32	512.00 kHz	4	203	512.32 kHz	+ 0.06%
	48	768.00 kHz	4	135	770.37 kHz	+ 0.31%
	64	1024.00 kHz	4	102	1019.61 kHz	- 0.43%

**CONFIDENTIAL**

**Unidirectional Serial Audio Interface I2S3**

**Table 8-22 Bit Clock Rates with 104 MHz Reference and Integer Ratio of Fractional Divider (Minimum Jitter)**

Nominal Audio Sampling Rates	Frame Length <sup>1)</sup>	Nominal Bit Clock	NUM	DEN	Real Bit Clock	Bit Clock Deviation
22.050 kHz	17	374.85 kHz	4	277	375.45 kHz	+ 0.16%
	18	396.90 kHz	4	262	396.96 kHz	+ 0.01%
	32	705.60 kHz	4	147	707.48 kHz	+ 0.27%
	48	1058.40 kHz	4	98	1061.22 kHz	+ 0.27%
	64	1411.20 kHz	4	74	1405.41 kHz	- 0.41%
24.000 kHz	17	408.00 kHz	4	255	407.84 kHz	- 0.04%
	18	432.00 kHz	4	241	431.54 kHz	- 0.11%
	32	768.00 kHz	4	135	770.37 kHz	+ 0.31%
	48	1152.00 kHz	4	90	1155.56 kHz	+ 0.31%
	64	1536.00 kHz	4	68	1529.41 kHz	- 0.43%
32.000 kHz	17	544.00 kHz	4	191	544.50 kHz	+ 0.09%
	18	576.00 kHz	4	181	574.59 kHz	- 0.25%
	32	1024.00 kHz	4	102	1019.61 kHz	- 0.43%
	48	1536.00 kHz	4	68	1529.41 kHz	- 0.43%
	64	2048.00 kHz	4	51	2039.22 kHz	- 0.43%
44.100 kHz	17	749.70 kHz	4	139	748.20 kHz	- 0.20%
	18	793.80 kHz	4	131	793.89 kHz	+ 0.01%
	32	1411.20 kHz	4	74	1405.41 kHz	- 0.41%
	48	2116.80 kHz	4	49	2122.45 kHz	+ 0.27%
	64	2822.40 kHz	4	37	2810.81 kHz	- 0.41%
48.000 kHz	17	816.00 kHz	4	127	818.90 kHz	+ 0.36%
	18	864.00 kHz	4	120	866.67 kHz	+ 0.31%
	32	1536.00 kHz	4	68	1529.41 kHz	- 0.43%
	48	2304.00 kHz	4	45	2311.11 kHz	+ 0.31%
	64	3072.00 kHz	4	34	3058.82 kHz	- 0.43%

<sup>1)</sup> Number of shift clock cycles

**CONFIDENTIAL**

**Unidirectional Serial Audio Interface I2S3**

**Table 8-23** shows the settings of the numerator and the denominator values for a 104 MHz reference clock. In the given configuration the (4\*denominator)/numerator ratio does not need to result in an integer. This way the fractional divider will produce additional jitter, but the given values result in the minimum possible bit clock deviation.

**Table 8-23 Bit Clock Rates with 104 MHz Reference and Non-Integer Ratio of Fractional Divider (Minimum Bit Clock Deviation)**

Nominal Audio Sampling Rates	Frame Length <sup>1)</sup>	Nominal Bit Clock	NUM	DEN	Real Bit Clock	Bit Clock Deviation
8.000 kHz	17	136.00 kHz	17	3250	136.00 kHz	+0.00%
	18	144.00 kHz	9	1625	144.00 kHz	+0.00%
	32	256.00 kHz	16	1625	256.00 kHz	+0.00%
	48	384.00 kHz	24	1625	384.00 kHz	+0.00%
	64	512.00 kHz	32	1625	512.00 kHz	+0.00%
11.025 kHz	17	187.43 kHz	103	14288	187.43 kHz	+0.00%
	18	198.45 kHz	65	8516	198.45 kHz	-0.00%
	32	352.80 kHz	441	32500	352.80 kHz	+0.00%
	48	529.20 kHz	1323	65000	529.20 kHz	+0.00%
	64	705.60 kHz	441	16250	705.60 kHz	+0.00%
12.000 kHz	17	204.00 kHz	51	6500	204.00 kHz	+0.00%
	18	216.00 kHz	27	3250	216.00 kHz	+0.00%
	32	384.00 kHz	24	1625	384.00 kHz	+0.00%
	48	576.00 kHz	36	1625	576.00 kHz	+0.00%
	64	768.00 kHz	48	1625	768.00 kHz	+0.00%
16.000 kHz	17	272.00 kHz	17	1625	272.00 kHz	+0.00%
	18	288.00 kHz	18	1625	288.00 kHz	+0.00%
	32	512.00 kHz	32	1625	512.00 kHz	+0.00%
	48	768.00 kHz	48	1625	768.00 kHz	+0.00%
	64	1024.00 kHz	64	1625	1024.00 kHz	+0.00%

**CONFIDENTIAL**

**Unidirectional Serial Audio Interface I2S3**

**Table 8-23 Bit Clock Rates with 104 MHz Reference and Non-Integer Ratio of Fractional Divider (Minimum Bit Clock Deviation)**

Nominal Audio Sampling Rates	Frame Length <sup>1)</sup>	Nominal Bit Clock	NUM	DEN	Real Bit Clock	Bit Clock Deviation
22.050 kHz	17	374.85 kHz	817	56668	374.85 kHz	+0.00%
	18	396.90 kHz	65	4258	396.90 kHz	+0.00%
	32	705.60 kHz	441	16250	705.60 kHz	+0.00%
	48	1058.40 kHz	1323	32500	1058.40 kHz	+0.00%
	64	1411.20 kHz	441	8125	1411.20 kHz	+0.00%
24.000 kHz	17	408.00 kHz	51	3250	408.00 kHz	+0.00%
	18	432.00 kHz	27	1625	432.00 kHz	+0.00%
	32	768.00 kHz	48	1625	768.00 kHz	+0.00%
	48	1152.00 kHz	72	1625	1152.00 kHz	+0.00%
	64	1536.00 kHz	96	1625	1536.00 kHz	+0.00%
32.000 kHz	17	544.00 kHz	34	1625	544.00 kHz	+0.00%
	18	576.00 kHz	36	1625	576.00 kHz	+0.00%
	32	1024.00 kHz	64	1625	1024.00 kHz	+0.00%
	48	1536.00 kHz	96	1625	1536.00 kHz	+0.00%
	64	2048.00 kHz	128	1625	2048.00 kHz	+0.00%
44.100 kHz	17	749.70 kHz	817	28334	749.70 kHz	+0.00%
	18	793.80 kHz	65	2129	793.80 kHz	+0.00%
	32	1411.20 kHz	441	8125	1411.20 kHz	+0.00%
	48	2116.80 kHz	1323	16250	2116.80 kHz	+0.00%
	64	2822.40 kHz	882	8125	2822.40 kHz	+0.00%
48.000 kHz	17	816.00 kHz	51	1625	816.00 kHz	+0.00%
	18	864.00 kHz	54	1625	864.00 kHz	+0.00%
	32	1536.00 kHz	96	1625	1536.00 kHz	+0.00%
	48	2304.00 kHz	144	1625	2304.00 kHz	+0.00%
	64	3072.00 kHz	192	1625	3072.00 kHz	+0.00%

<sup>1)</sup> Number of shift clock cycles

**CONFIDENTIAL**

## Unidirectional Serial Audio Interface I2S3

For the parallel-to-serial conversion the clock signal `clk_tx` is used (see [Figure 8-24](#)). With the bit field **I2S3\_CSEL.TXCLKSEL** the source of this clock can be selected according to [Table 8-24](#).

**Table 8-24 Selection of `clk_TX`**

TXCLKSEL	clk_TX
0	Fractional Divider
1	i2s_clk

### 8.6.6.4 Jitter Of Output Signals

There are three different sources which can cause the output signals to jitter:

- Jitter from PLL, phase shifter and pad non-linearities
- Jitter of reference clock from fractional divider
- Jitter due to synchronization of reference clock into peripheral clock domain.

The individual contributions of these three jitter types sum up to the total jitter. While the jitter caused by the first source is inevitable the occurrence of the other two jitters depends on the selected clock configuration and the operation mode.

The fractional divider causes a jitter if the  $(4 \cdot \text{denominator}) / \text{numerator}$  ratio does not result in an integer. As the fractional divider is not used in slave mode this kind of jitter can only occur in master mode.

In addition to this a synchronization jitter can occur if the reference clock is different from the peripheral clock, which is running at the `clk_dsp` frequency. In slave mode the reference clock comes from outside the chip. Therefore, the synchronization jitter is always present in slave mode.

**CONFIDENTIAL**

**Unidirectional Serial Audio Interface I2S3**

**Table 8-25** gives an overview on all possible combinations in master mode. If clk\_dsp and clk\_ms are running at the same frequency no synchronization jitter will occur.

In slave mode the jitter only consists of the synchronization jitter and the fixed jitter, which always leads to a total jitter of 2 ns + 1 clk\_dsp cycle.

**Table 8-25 Jitter of CLK3, WAX and TX Outputs in Master Mode**

Ref clock	clk_dsp	clk_dsp	clk_ms	clk_ms
<b>Sync jitter<sup>1)</sup></b>	0	0	1 clk_dsp cycle	1 clk_dsp cycle
<b>Ratio (4*denominator) /numerator</b>	integer	non-integer	integer	non-integer
<b>FDIV jitter</b>	0	1 clk_dsp cycle	0	1 clk_ms cycle
<b>Fixed jitter</b>	2 ns	2 ns	2 ns	2 ns
<b>Total jitter</b>	2 ns	1 clk_dsp cycle +2 ns	1 clk_dsp cycle +2 ns	1 clk_dsp cycle +1 clk_ms cycle +2 ns

<sup>1)</sup> Does not occur if clk\_dsp = clk\_ms

### 8.6.6.5 Delay Of Output Signals

In master mode the output signals show a variable delay with respect to the output clock. This delay depends on the frequency of clk\_dsp. **Table 8-26** gives an overview. This delay is increased by the fixed path delay from the I2S block to the pads at the chip, refer to **Section 13.2.1.2.6 I2S (on Page 1346)**.

**Table 8-26 Delay Of Output Signals In Master Mode**

Signals	Delay
CLK to WA	2 clk_dsp cycles
CLK to TX	1 clk_dsp cycle

### 8.6.7 Interface to TEAKLite Bus

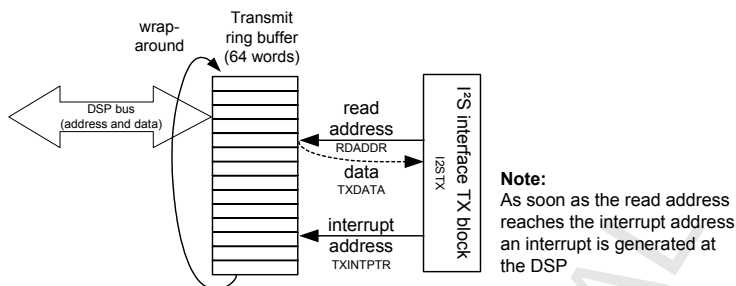
The data communication between the TEAKLite and the I2S Interface is performed using a dual port data ring buffer with a size of 64 words and 16-bit word width. The data transfer is controlled by the TEAKLite interrupt unit. The TEAKLite accesses the buffer with 2 wait states. A simplified block diagram of the interface for the I2S3 is shown in **Figure 8-25**.



CONFIDENTIAL

Unidirectional Serial Audio Interface I2S3

Figure 8-25 Functional Block Diagram of the I2S3 Transmit Buffer

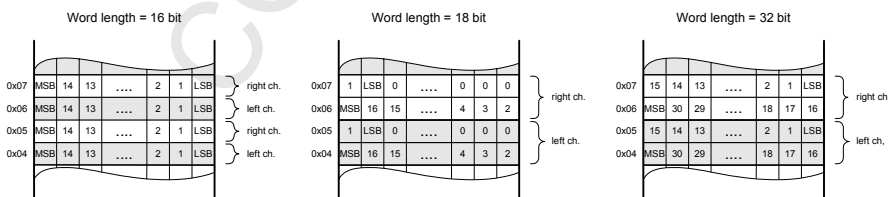


The data buffers and the registers of the I2S interfaces are directly accessible from the TEAKLite via the Z-Bus. The dual port data buffer between the TEAKLite and the I2S interface is accessible by the TEAKLite as well as from the I2S interface. In order to transmit audio data (normal mode) the TEAKLite has to write pairs of stereo audio samples to consecutive addresses in the ring buffer. These audio samples will be read by the I2S interface at a time.

Moreover, the TEAKLite may access each 16-bit word of the ring buffer independently. This means that all 64 words of the buffer are mapped to the data address space of the TEAKLite. In detail the transmit ring buffer is mapped to addresses  $XXXX_H$  up to  $XXXX_H + 3F_H$ . In this way the TEAKLite has full control what memory location it wants to write to. It is also possible to operate the interface buffer in a way best suiting the needs of the software as various logical data types (for example, a FIFO queue) may be implemented on this buffer.

In the normal mode the buffer holds left channel data and right channel data in an alternating order. The mapping starts always at the buffer address  $00_H$  with the left channel. The mapping of the left and right channels into the buffer is fixed. **Figure 8-26** shows for different word lengths, how the data words are mapped into the ring buffer.

Figure 8-26 Mapping of Data Words into the Buffer



For flow control an interrupt mechanism is implemented which causes an interrupt (I2S3\_TX\_INT) at the TEAKLite as soon as the interface reads from specific buffer locations. The address where the interrupt is generated at the TEAKLite is stored in the

**CONFIDENTIAL****Unidirectional Serial Audio Interface I2S3**

register **I2S3\_TXINTADDR**. This register can be written by the TEAKLite even during operation of the peripheral.

**8.6.7.1 Data Transmission**

The read address the I2S interface currently reads from is stored in **I2S3\_RADDR.RDADDR**. This register is readable from the TEAKLite. The address **RDADDR** is automatically incremented at each read access by the I2S interface to the ring buffer and is wrapped around from the end address to the start address. The write position of the TEAKLite to the transmit buffer is controlled by the TEAKLite and hence does not need to be stored in a separate register within the I2S interface.

As soon as the read address **RDADDR** equals the interrupt address **I2S3\_TXINTADDR.TXINTPTR**, an interrupt is generated at the TEAKLite setting bit **INT\_FINTB0.I2S3TX**. Within the interrupt service routine the TEAKLite has to reset the interrupt source (**INT\_FINTB0.RI2S3TX**) and to set a new value for the next interrupt position.

The I2S interface is activated by setting the bit **I2S3\_CTRL.I2SON** to 1. This enables the interface, but does not yet activate the output clock until bit **I2S3\_CTRL.I2STXSTART** is set to 1. Hence, it is possible to write to the data buffer without sending dummy values to the external device. After writing data in the transmit buffer, the burst or DAI mode can be selected.

When the bit **I2STXSTART** is set to 1, the I2S hardware starts to read from the transmit buffer. In normal mode the first element to be transmitted belongs to the left channel (start address). The transmission can be stopped by setting **I2STXSTART** to 0. After setting **I2STXSTART** back to 1 again, the transmission is continued at the next data set in memory, which follows the data sent before the transfer had been stopped. After reset or after **I2SON** has been toggled both the interrupt pointer and the read pointer are reset to zero.

**CONFIDENTIAL**

## Unidirectional Serial Audio Interface I2S3

### 8.6.8 I2S Control and Status Registers

**Table 8-27 I2S3 Register List**

Register Group	Register Name	Register Symbol
Control and Status Registers	Control Register	<b>I2S3_CTRL</b>
	Clock Select Register	<b>I2S3_CSEL</b>
	Write Address Register	<b>I2S3_RADDR</b>
	Divider Numerator Register	<b>I2S3_NUM</b>
	Divider Denominator Register	<b>I2S3_DEN</b>
Transmitter Registers	Transmitter Configuration Register	<b>I2S3_TXCONF</b>
	Transmitter Interrupt Address Register	<b>I2S3_TXINTADDR</b>

#### 8.6.8.1 Control Register

## I2S3 CTRL

## Control Register

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED										TX BUR ST	RESERV ED	RES ERV ED	I2ST XST ART	I2S ON	

Field	Bits	Type	Description
I2SON	0	w	<b>Module Clock On</b> 0: i2s_clk is switched off. 1: i2s_clk is switched on. <i>Note: Before changing the interface configuration this bit must be switched off and on again.</i>
I2STXSTART	1	rwh	<b>Start operation of I2S data transmission</b> 0: Data transmission halted. 1: Start data transmission.
TXPCM	5	rw	<b>PCM mode select for transmit direction</b> 0: Normal operation. 1: PCM mode operation. <i>Note: In PCM mode the settings from the <b>I2S3_TXCONF</b> register bits 12..0 are ignored.</i>
RESERVED	2, 3, 4, 15:6	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**Unidirectional Serial Audio Interface I2S3**

### 8.6.8.2 Clock Select Register

Master or slave mode operation of the I2S interface as well as the combinations of transmission and reception can be selected by setting the clock source of the I2S interface in register **I2S3\_CSEL**.

#### **I2S3\_CSEL**

**Clock Select Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>												<b>CLK SEL</b>	<b>RES ERV ED</b>	<b>TX CLK SEL</b>	<b>RES ERV ED</b>

Field	Bits	Type	Description
<b>TXCLKSEL</b>	1	rw	Select clock source for signal clk_tx (see <a href="#">Table 8-24 Selection of clk_TX (on Page 435)</a> )
<b>CLKSEL</b>	3	rw	Select source for I2S3_CLK (see <a href="#">Table 8-18 Selection of I2S3_CLK for I2S3 (on Page 425)</a> )
<b>RESERVED</b>	0, 2, 15:4	r	Reserved; these bits must be left at their reset values.

### 8.6.8.3 Read Address Register

#### **I2S3\_RADDR**

**Read Address for Interface Buffer Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>										<b>RDADDR</b>					

Field	Bits	Type	Description
<b>RDADDR</b>	5:0	rh	Address of transmit interface ring buffer read position
<b>RESERVED</b>	15:6,	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**Unidirectional Serial Audio Interface I2S3**

### 8.6.8.4 Divider Numerator Register

With this register the SW can set the numerator of the Fractional Divider and select the reference frequency of the transmit baud rate generator.

#### I2S3\_NUM

#### Divider Numerator Register

**Reset value: 0001<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		FREF		RESERVED	NUMERATOR										

Field	Bits	Type	Description
<b>NUMERATOR</b>	10:0	rw	<b>Numerator</b> Determines the numerator value of the fractional divider described in <a href="#">Section 8.6.6</a> . <i>Note: The value 0 is not permitted and will lead to undefined results.</i>
<b>FREF</b>	13:12	rw	<b>FREF selection</b> 00: Reference input 0 (Module clock). 01: Reference input 1 (104 MHz). 10: Reserved. 11: Reserved.
<b>RESERVED</b>	15:14, 11	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**Unidirectional Serial Audio Interface I2S3**

### 8.6.8.5 Divider Denominator Register

This register sets the denominator of the Fractional Divider.

**I2S3\_DEN**

**Divider Denominator Register**

**Reset value: 0002<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>DENOMINATOR</b>															

Field	Bits	Type	Description
<b>DENOMINATOR</b>	15:0	rw	<b>Denominator value</b> Determines the denominator value of the fractional divider 1 described in <a href="#">Section 8.6.6</a> .  <i>Note: The values 0 and 1 are not permitted and lead to undefined results.</i>

*Note: Please keep in mind that for all  $(4 \times \text{denominator})/\text{numerator}$  ratios which do not result in an integer the bit clock will have a jitter of one clock period of the reference clock. Ratios of numerator/denominator which are larger or equal to 0.5 are not allowed.*

**CONFIDENTIAL**

**Unidirectional Serial Audio Interface I2S3**

## 8.6.9 I2S Transmitter Registers

### 8.6.9.1 Transmitter Configuration Register

**I2S3\_TXCONF**

**Transmitter Configuration Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WA LEN	CLK CO NT	CLK OU T	MUT E_R	MUT E_L	MONO		ALIG N		WIDTH		PERIOD	POL	DEL	EDGE	

Field	Bits	Type	Description
<b>EDGE</b>	0	rw	<b>Edge for output of transmit data in normal mode</b> 0: Falling edge 1: Rising edge  <i>Note: To ensure proper operation in normal mode the sampling edge at the receiver should be set complementary to the output edge at the transmitter.</i>
<b>DEL</b>	1	rw	<b>Delay of first bit in normal mode</b> 0: New word starts at edge of WA. 1: New word starts one clock after edge of WA.
<b>POL</b>	2	rw	<b>Polarity of WA signal in normal mode</b> 0: WA = 1 indicates right channel; WA = 0 indicates left channel. 1: WA = 1 indicates left channel; WA = 0 indicates right channel.
<b>PERIOD</b>	4:3	rw	<b>Frame period in normal mode</b> 00: 64 clocks 01: 48 clocks 10: 32 clocks 11: Reserved  <i>Note: This value is only used in master mode and as an initial value for slave mode. In slave mode the transmitter detects the frame length and adjusts its output accordingly after 1-2 frames.</i>

**CONFIDENTIAL**

## Unidirectional Serial Audio Interface I2S3

Field	Bits	Type	Description
<b>WIDTH</b>	7:5	rw	<b>Sample width in normal mode</b> 000: 16 bits 001: Reserved 010: Reserved 011: Reserved 100: 18 bits 101: 20 bits 110: 24 bits 111: 32 bits
<b>ALIGN</b>	8	rw	<b>Data alignment in normal mode</b> 0: Left aligned 1: Right aligned In right aligned mode, unused bits in front of the transmitted data are set to the value of the MSB to be transmitted (two's complement sign extension).
<b>MONO</b>	10:9	rw	<b>Transfer mode in normal mode</b> 00: Stereo mode. 01: Reserved. 10: Right data transmitted on both channels. 11: Left data transmitted on both channels.
<b>MUTE_L</b>	11	rw	<b>Mute left channel in normal mode</b> 0: Channel active. 1: Left channel muted.
<b>MUTE_R</b>	12	rw	<b>Mute right channel in normal mode</b> 0: Channel active. 1: Right channel muted.
<b>CLK_OUT</b>	13	rw	<b>CLK burst output level in PCM mode</b> This bit is only used when CLK_CONT is zero 0: i2s3_clk is low 1: i2s3_clk is high
<b>CLK_CONT</b>	14	rw	<b>CLK burst behavior in PCM mode</b> This bit determines the behavior of the CLK line when the transmission is stopped (before first burst and between consecutive bursts). 0: i2s3_clk is stopped and switched to a fixed value determined by bit <b>CLK_OUT</b> . 1: i2s3_clk remains running



**CONFIDENTIAL**

**Unidirectional Serial Audio Interface I2S3**

Field	Bits	Type	Description
<b>WA_LEN</b>	15	rw	<b>WA pulse length in PCM mode</b> 0: i2s3_wa is high for 1 cycle of i2s3_clk 1: i2s3_wa is high for 2 cycles of i2s3_clk

### 8.6.9.2 Transmitter Interrupt Address Register

#### **I2S3\_TXINTADDR**

**Transmitter Interrupt Address Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>										<b>TXINTPTR0</b>					

Field	Bits	Type	Description
<b>TXINTPTR</b>	5:0	rw	<b>Transmitter Interrupt Address</b> of transmit interface ring buffer
<b>RESERVED</b>	15:6	r	Reserved; these bits must be left at their reset values.

CONFIDENTIAL

**CONFIDENTIAL**

**Synchronous Serial Controller on DSP**

## 8.7 Synchronous Serial Controller on DSP

History	
Design Spec.	Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.01	
<b>Page 464</b>	Removed note from <b>Section 8.7.5.1.1 Transmit FIFO</b> WS00005897
Changes for Rev. 1.02	
<b>Page 473</b>	Added Note to <b>Section 8.7.5.3 Baud Rate Generation</b> WS00005890
Changes for Rev. 1.03	
<b>Page 464</b>	Added Note to <b>Transparent Mode</b> WS00005897
Changes for Rev. 1.06	

### System Integration on TEAKLite:

- Supply domain: VDD\_DSP
- Chip internal interfaces:
  - Clock domain: gclk\_dsp\_per drives the internal module clock ssc1\_clk
  - Bus domain: TEAKLite Z-Bus
  - Interrupt sources: SSC\_TX\_INT, SSC\_RX\_INT, SSC\_ERR\_INT
- Chip external signals related to this block (refer to **Chapter 3 Pin Descriptions (on Page 49)** for pin configuration options): SSC1\_SCLK, SSC1\_MTSR, SSC1\_MRST
- Monitor pins: Refer to **Section 11.8.10 Internal Signal Monitoring (on Page 1209)**

### 8.7.1 Functional Overview

The Synchronous Serial Interface Controller (SSC) allows serial communication to other microcontrollers, microprocessors or external peripherals. The PMB7870 implements one SSC interface in the TEAKLite subsystem.

#### SSC Features

- Master and slave mode operation
- Full-duplex or half-duplex transfers
  - Flexible data format:
  - Programmable number of data bits: 2 to 16 bit
  - Programmable shift direction: LSB or MSB shift first

**CONFIDENTIAL**

**Synchronous Serial Controller on DSP**

- Programmable clock polarity: idle low or high state for the shift clock
- Programmable clock/data phase: data shift with leading or trailing edge of shift clock
- Baud rate generation up to 26 MBaud
- Interrupt generation
  - On a transmitter empty condition
  - On a receiver full condition
  - On an error condition (receive, phase, baud rate, transmit error)
- FIFO:
  - 32-stage receive FIFO (RXFIFO)
  - 32-stage transmit FIFO (TXFIFO)
  - Independent control of RXFIFO and TXFIFO
  - 16-Bit FIFO data width
  - Programmable Receive/Transmit Interrupt Trigger Level
  - Receive and transmit FIFO filling level indication
  - Overrun error generation
  - Underflow error generation
- Three pin interface:
  - Flexible SSC pin configuration

### 8.7.2 Register Overview

To obtain the addresses of these registers refer to [Section 12.1.2 Registers in the DSP Memory Space \(on Page 1266\)](#).

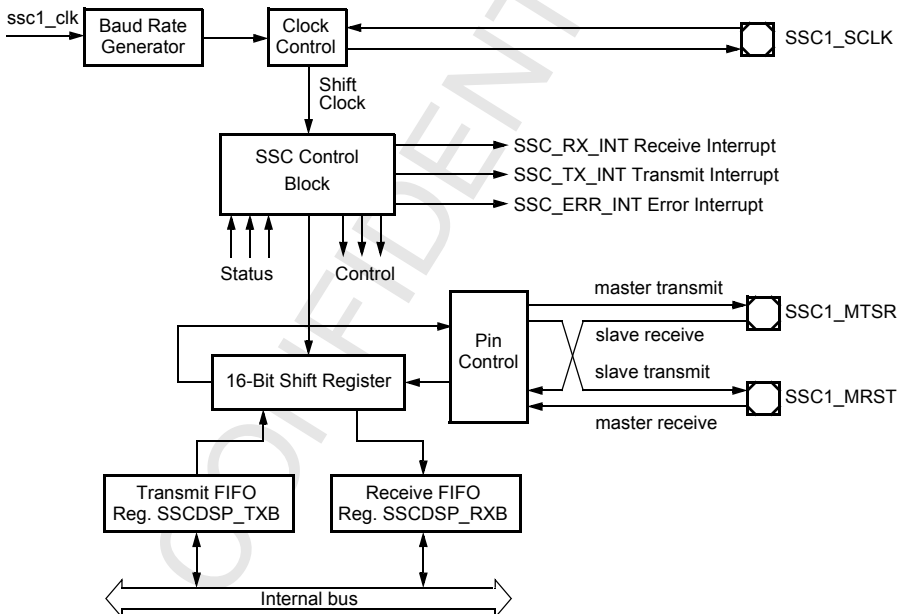
**Table 8-28 SSC Register Overview**

Register Group	Register Name	Register Symbol
Control	Control Register Write HW Modified Bits Control Register	<a href="#">SSCDSP_CON (on Page 450)</a> <a href="#">SSCDSP_WHBCON (on Page 453)</a>
Data	Transmit Buffer Register Receive Buffer Register	<a href="#">SSCDSP_TXB (on Page 455)</a> <a href="#">SSCDSP_RXB (on Page 456)</a>
FIFO	Receive FIFO Control Register Transmit FIFO Control Register FIFO Status Register	<a href="#">SSCDSP_RXFCON (on Page 468)</a> <a href="#">SSCDSP_TXFCON (on Page 470)</a> <a href="#">SSCDSP_FSTAT (on Page 472)</a>
Baud Rate Generation	Baud Rate Timer Reload Register Fractional Divider Register	<a href="#">SSCDSP_BR (on Page 474)</a> <a href="#">SSCDSP_FDV (on Page 475)</a>

### 8.7.3 Structural Overview

The SSC supports full-duplex and half-duplex synchronous communication at a rate of up to 26 Mbaud . The serial clock signal can be generated by the SSC itself (master mode) or be received from an external master (slave mode). Data width, shift direction, clock polarity and phase are programmable, so it can be used with other synchronous serial interfaces, serve for master/slave or multimaster interconnections or operate compatible to the popular SPI interface. Possible applications are: IO expansion via with shift registers, connection to peripherals (for example, EEPROMs, etc.) or communication with other controllers (networking). Data is transmitted or received on pad SSC1\_MTSR (Master Transmit or Slave Receive) and SSC1\_MRST (Master Receive/Slave Transmit). The clock signal is output or input on pad SSC1\_SCLK (Shift Clock). These pads are alternate functions of port pins.

**Figure 8-27 Synchronous Serial Interface Controller Block Diagram**



#### 8.7.3.1 SSC Control Registers

##### 8.7.3.1.1 Control Register

The operating mode of the SSC is set by the control register **SSCDSP\_CON**. This register contains control bits for mode, baud rate and error check selection, and status

**CONFIDENTIAL**

**Synchronous Serial Controller on DSP**

flags for error identification. Depending on bit **SSCDSP\_CON.EN**, either control functions (programming mode) or status flags and master/slave control (operating mode) is enabled.

The setting and clearing of the bits **SSCDSP\_CON.TE**, **SSCDSP\_CON.RE**, **SSCDSP\_CON.PE** and **SSCDSP\_CON.BE** can be done via the special register **SSCDSP\_WHBCON**.

**1. SSCDSP\_CON.EN = 0: Programming Mode**

**SSCDSP\_CON**  
**Control Register**

**Reset Value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN	MS	CLK ON	ARE N	BEN	PEN	REN	TEN	LB	PO	PH	HB	BM			

Field	Bits	Type	Description
<b>BM</b>	3:0	rw	<b>Data Width Selection</b> 0000 Reserved. Do not use this combination. 0001 to 1111 Transfer Data Width is 2...16 bit (<BM>+1)
<b>HB</b>	4	rw	<b>Heading Control</b> 0 Transmit/Receive LSB First. 1 Transmit/Receive MSB First.
<b>PH</b>	5	rw	<b>Clock Phase Control</b> 0 Shift transmit data on the leading clock edge, latch on trailing edge. 1 Latch receive data on leading clock edge, shift on trailing edge.
<b>PO</b>	6	rw	<b>Clock Polarity Control</b> 0 Idle clock line is low, leading clock edge is low-to-high transition. 1 Idle clock line is high, leading clock edge is high-to-low transition.
<b>LB</b>	7	rw	<b>Loop Back Control</b> 0 Normal output. 1 Receive input is connected to transmit output (half-duplex mode).
<b>TEN</b>	8	rw	<b>Transmit Error Enable</b> 0 Ignore transmit errors. 1 Check transmit errors.

**CONFIDENTIAL**

**Synchronous Serial Controller on DSP**

Field	Bits	Type	Description
<b>REN</b>	9	rw	<b>Receive Error Enable</b> 0 Ignore receive errors. 1 Check receive errors.
<b>PEN</b>	10	rw	<b>Phase Error Enable</b> 0 Ignore phase errors. 1 Check phase errors.
<b>BEN</b>	11	rw	<b>Baud Rate Error Enable</b> 0 Ignore baud rate errors 1 Check baud rate errors
<b>AREN</b>	12	rw	<b>Automatic Reset Enable</b> 0 No additional action upon a baud rate error 1 The SSC is automatically reset upon a baud rate error
<b>CLKON</b>	13	rw	<b>Module Clock On</b> 0 ssc1_clk is switched off 1 ssc1_clk is switched on
<b>MS</b>	14	rw	<b>Master Select</b> 0 Slave Mode. Operate on shift clock received via ssc1_clk 1 Master Mode. Generate shift clock and output it via ssc1_clk
<b>EN</b>	15	rw	<b>Enable Bit = 0</b> 0 Transmission and reception disabled. Access to control bits. 1 Transmission and reception enabled. Access to status flags and M/S control.

*Note: The module clock ssc1\_clk is controlled by bit **SSCDSP\_CON.CLKON**. This bit must be set before the other bits of the register **SSCDSP\_CON** can be written to.*

**CONFIDENTIAL**

**Synchronous Serial Controller on DSP**

## 2. **SSCDSP\_CON.EN = 1: Operating Mode**

**SSCDSP\_CON**

**Control Register**

**Reset Value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN	MS	CLK ON	BSY	BE	PE	RE	TE	RESERVED					BC		

Field	Bits	Type	Description
<b>BC</b>	3:0	rh	Bit Count Field 0001 - 1111 Shift counter is updated with every shifted bit.
<b>TE</b>	8	rh	Transmit Error Flag 0 No error. 1 Transfer starts with the slave's transmit buffer not being updated. <i>Note: Bit cannot be written. To reset the flag use register <b>SSCDSP_WHBCON</b> instead.</i>
<b>RE</b>	9	rh	Receive Error Flag 0 No error. 1 Reception completed before the receive buffer was read. <i>Note: Bit cannot be written. To reset the flag use register <b>SSCDSP_WHBCON</b> instead.</i>
<b>PE</b>	10	rh	Phase Error Flag 0 No error. 1 Received data changes around sampling clock edge. <i>Note: Bit cannot be written. To reset the flag use register <b>SSCDSP_WHBCON</b> instead.</i>
<b>BE</b>	11	rh	Baud Rate Error Flag 0 No error. 1 More than factor 2 or 0.5 between slave's actual and expected baud rate. <i>Note: Bit cannot be written. To reset the flag use register <b>SSCDSP_WHBCON</b> instead.</i>
<b>BSY</b>	12	rh	Busy Flag Set while a transfer is in progress.



**CONFIDENTIAL**

**Synchronous Serial Controller on DSP**

Field	Bits	Type	Description
<b>CLKON</b>	13	rw	Module Clock On 0 ssc1_clk is switched off. 1 ssc1_clk is switched on.
<b>MS</b>	14	rw	Master Select Bit 0 Slave Mode. Operate on shift clock received via SSC1_SCLK. 1 Master Mode. Generate shift clock and output it via SSC1_SCLK.
<b>EN</b>	15	rw	<b>Enable Bit = 1</b> 0 Transmission and reception disabled. Access to control bits. 1 Transmission and reception enabled. Access to status flags and M/S control.
<b>RESERVED</b>	7:4	r	Reserved; these bits must be left at their reset values.

*Note: The target (control bits or flags) of an access to **SSCDSP\_CON** is determined by the state of bit **SSCDSP\_CON.EN** prior to the access, for example, writing C057<sub>H</sub> to **SSCDSP\_CON** in programming mode (**SSCDSP\_CON.EN** = 0) initializes SSC (**SSCDSP\_CON.EN** was 0) and then turn it on (**SSCDSP\_CON.EN** = 1).*

*Note: The bits which can be modified by HW have shadow bits, so that the contents do not change during a two wait states read access. Thus new bit values can be read with a delay of one clock cycle.*

### 8.7.3.1.2 Write Hardware Modified Control Register

When the SSCx is in the operating mode (**SSCDSP\_CON.EN** = 1), the bits **SSCDSP\_CON.BE**, **SSCDSP\_CON.PE**, **SSCDSP\_CON.RE** and **SSCDSP\_CON.TE** are modified using the register **SSCDSP\_WHBCON**.

#### **SSCDSP\_WHBCON**

**Write Hardware Modified Control Register**

**Reset Value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>SET BE</b>	<b>SET PE</b>	<b>SET RE</b>	<b>SET TE</b>	<b>CLR BE</b>	<b>CLR PE</b>	<b>CLR RE</b>	<b>CLR TE</b>	<b>RESERVED</b>							

*Note: The module clock ssc\_clk is controlled by bit **SSCDSP\_CON.CLKON**. This bit must be set before the other bits of the register **SSCDSP\_WHBCON** can be written to.*

**CONFIDENTIAL**

**Synchronous Serial Controller on DSP**

Field	Bits	Type	Description
<b>BC</b>	3:0	rh	<b>Bit Count Field</b> 0001 to 1111 Shift counter is updated with every shifted bit.
<b>CLRTE</b>	8	w	<b>Clear Transmit Error Flag Bit</b> 0 No action 1 Clears bit <b>SSCDSP_CON.TE</b> <i>Note: Reading always returns 0.</i>
<b>CLRRE</b>	9	w	<b>Clear Receive Error Flag Bit</b> 0 No action 1 Clears bit <b>SSCDSP_CON.RE</b> <i>Note: Reading always returns 0.</i>
<b>CLRPE</b>	10	w	<b>Clear Phase Error Flag Bit</b> 0 No action 1 Clears bit <b>SSCDSP_CON.PE</b> <i>Note: Reading always returns 0.</i>
<b>CLRBE</b>	11	w	<b>Clear Baud Rate Error Flag Bit</b> 0 No action 1 Clears bit <b>SSCDSP_CON.BE</b> <i>Note: Reading always returns 0.</i>
<b>SETTE</b>	12	w	<b>Set Transmit Error Flag Bit</b> 0 No action 1 Sets bit <b>SSCDSP_CON.TE</b> <i>Note: Reading always returns 0.</i>
<b>SETRE</b>	13	w	<b>Set Receive Error Flag Bit</b> 0 No action 1 Sets bit <b>SSCDSP_CON.RE</b> <i>Note: Reading always returns 0.</i>
<b>SETPE</b>	14	w	<b>Set Phase Error Flag Bit</b> 0 No action 1 Sets bit <b>SSCDSP_CON.PE</b> 0 has no effect. <i>Note: Reading always returns 0.</i>

**CONFIDENTIAL**

**Synchronous Serial Controller on DSP**

Field	Bits	Type	Description
<b>SETBE</b>	15	w	<b>Set Baud Rate Error Flag Bit</b> 0 No action 1 Sets bit <b>SSCDSP_CON.BE</b> <i>Note: Writing 0 has no effect. Reading always returns 0.</i>
<b>RESERVED</b>	7:0	r	Reserved; these bits must be left at their reset values.

### 8.7.3.2 SSC Data Registers

#### 8.7.3.2.1 Transmitter Buffer Register

The SSC Transmitter Buffer Register contains the transmit data value.

##### **SSCDSP\_TXB**

**Transmitter Buffer Register**

**Reset Value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>TXB_VALUE</b>															

Field	Bits	Type	Description
<b>TXB_VALUE</b>	15:0	rw	<b>Transmit Buffer Value</b> <b>TXB_VALUE</b> is the data value to be transmitted. Unselected bits of TXB are ignored during transmission.

*Note: The register **SSCDSP\_TXB** is driven by a clock independent from the module clock **ssc1\_clk**. Thus this register can be written to, even if the bit **SSCDSP\_CON.CLKON** is not set.*

**CONFIDENTIAL**

**Synchronous Serial Controller on DSP**

### 8.7.3.2.2 Receiver Buffer Register

The SSC Receive Buffer Register contains the received data value.

#### **SSCDSP\_RXB**

#### **Receiver Buffer Register**

**Reset Value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RXB_VALUE</b>															

Field	Bits	Type	Description
<b>RXB_VALUE</b>	15:0	rh	<b>Receive Buffer Value</b> <b>RXB_VALUE</b> is the received data value. Un-selected bits of RXB must be ignored by SW.

*Note: This register has a shadow register, so that the content does not change during a two wait states read access. Thus a new received data value can be read with a delay of one clock cycle.*

### 8.7.4 Operating Modes

The operating mode of the SSC is controlled by its bit-addressable control register **SSCDSP\_CON**. This register serves for two purposes:

- During programming (SSC disabled by **SSCDSP\_CON.EN** = 0) it provides access to a set of control bits.
- During operation (SSC enabled by **SSCDSP\_CON.EN** = 1) it provides access to a set of status flags.

The shift register of SSC is connected to both the transmit and the receive pin via the pin control logic (see block diagram in **Figure 8-27**). Transmission and reception of serial data are synchronized and take place at the same time, that is, the receive bits are shifted in while the transmit bits are shifted out. Transmit data is written into the Transmit Buffer **SSCDSP\_TXB** (on Page 455) and is moved to the shift register as soon as the register is empty. If the SSC is configured as master (**SSCDSP\_CON.MS** = 1) it begins transmitting immediately. In slave mode (**SSCDSP\_CON.MS** = 0) the SSC will wait for an active shift clock. When the transfer starts, the busy flag **SSCDSP\_CON.BSY** is set and the Transmit Interrupt Request Line **SSC1\_TX\_INT** is activated to indicate that **SSCDSP\_TXB** can be reload again. When the programmed number of bits (2...16) has been transferred, the content of the shift register is moved to the Receive Buffer **SSCDSP\_RXB** and a Receive Interrupt Request Line **SSC\_RX1\_INT** is activated. If **SSCDSP\_RXB** is empty **SSCDSP\_CON.BSY** will be cleared at the same time and the SSC stops. Software should not modify **SSCDSP\_CON.BSY**, as this flag is controlled by

**CONFIDENTIAL**

**Synchronous Serial Controller on DSP**

hardware. All interrupt flags set for the TEAKLite are located in register **INT\_FINTB0** (on [Page 344](#)).

*Note: Only one peer in a SSC communication can be master at any given time.*

Several parameters of the serial data transfer can be configured:

- The data width can be chosen from 2 to 16 bits.
- A transfer may start with the LSB or the MSB.
- The shift clock may be low or high when idle.
- The data bits may be shifted with the leading or trailing edge of the clock signal.
- The baud rate may be set from 1.55 Baud up to 26 MBaud.
- The shift clock can be generated (master) or received (slave).

These features allows the adaptation of the SSC to a wide range of applications, where a serial data transfer is required.

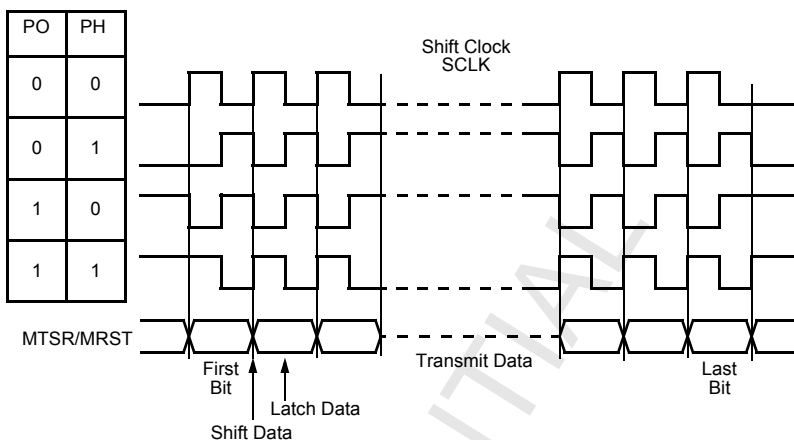
### **Data Width Selection**

The data width selection supports the transfer of frames with any word length from 2 up to 16 bits. Starting with the LSB (**SSCDSP\_CON.HB** = 0) allows communication, for example, with an SSC device in synchronous mode (C166 family) or 8051 like serial interfaces. Starting with the MSB (**SSCDSP\_CON.HB** = 1) allows operation compatible with the SPI interface. Regardless of the selected data width and independent from the bit ordering (MSB or LSB first) the transfer data is always right aligned in registers **SSCDSP\_TXB** (on [Page 455](#)) and **SSCDSP\_RXB** (on [Page 456](#)) with the LSB stored at bit 0. The data bits are rearranged for transfer by the internal shift register logic. The unselected bits of **SSCDSP\_TXB** are ignored, the unselected bits of **SSCDSP\_RXB** are not valid and should be ignored by the receiver service routine.

### **Clock Control**

The Clock Control allows the adaptation of the SSC transmit and receive behavior to a variety of serial interfaces. One clock edge (rising or falling) is used to shift out transmit data, while the other clock edge is used to latch in received data. Bit **SSCDSP\_CON.PH** selects the leading edge or the trailing edge for each function. Bit **SSCDSP\_CON.PO** selects the level of the clock line in the idle state.

Figure 8-28 Serial Clock Phase and Polarity Options



## Continuous Transfers

When the transmit interrupt request flag is set the Transmit Buffer **SSCDSP\_TXB** is empty and ready to be loaded with the next transmit data. If **SSCDSP\_TXB** has been reloaded by before the current transmission is finished, the data is immediately transferred to the shift register and the next transmission will start without any additional delay. On the data line there is no gap between the two successive frames. As a result two byte transfers will look the same as one word transfer. This feature can be used to interface with devices which can operate with or require more than 16 data bits per transfer. It is just a matter of software, how long a total data frame length can be. This option can also be used to interface to byte-wide and word-wide devices on the same serial bus.

*Note: This method is limited to multiples of the selected basic data width, since it would require disabling/enabling of the SSC to reprogram the basic data width on-the-fly.*

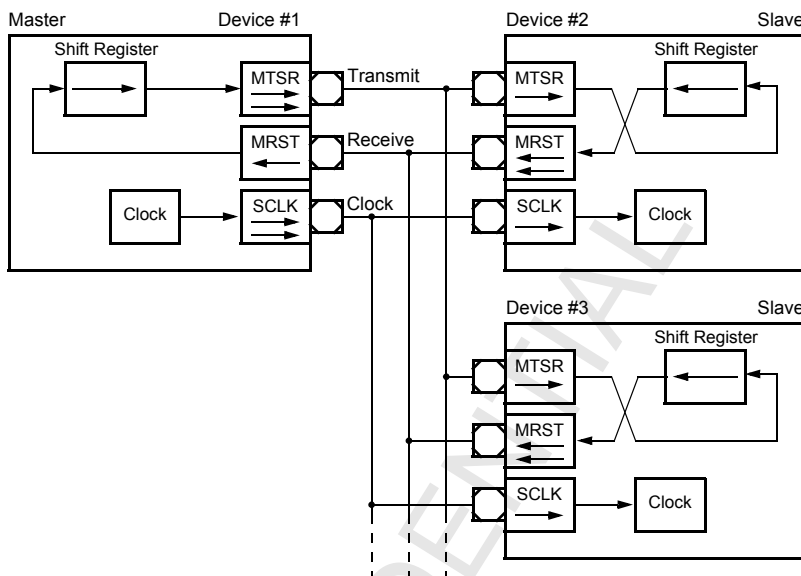
### 8.7.4.1 Duplex Operation

The different devices are connected through three lines as shown in **Figure 8-29**. The definition of these lines is always determined by the master: The line connected to the master's data output pad **SSC1\_MTSR** is the transmit line, the receive line is connected to its data input line **SSC1\_MRST**, and the clock line is connected to pin **SSC1\_SCLK**.

CONFIDENTIAL

Synchronous Serial Controller on DSP

Figure 8-29 SSC Duplex Configuration



Only the device selected for master operation generates and outputs the serial clock on pin SSC1\_SCLK. All slaves are driven by this clock, so the pin SSC1\_SCLK must be switched to input mode at each slave. The output of the master's shift register is connected to the external transmit line, which in turn is connected to the slave's shift register input. The output of the slave's shift register is connected to the external receive line in order to enable the master to receive the data shifted out of the slave. The external connections are hard-wired, the function and direction of these pins is determined by the master or slave operation of the individual device.

*Note: The shift direction shown in **Figure 8-29** applies to MSB-first transmission as well as to LSB-first transmission.*

When initializing the devices in this configuration, one device must be selected for master operation while all other devices must be set into slave mode. Initialization includes the operating mode of the SSC and also the function of the respective port lines.

The data output pins SSC1\_MRST of all slave devices are connected to the same receive line in this configuration. During a transfer each slave shifts out data from its shift register. There are two ways to avoid collisions on the receive line due to different slave data:

- **Only one slave drives the line** and enables the driver at its SSC1\_MRST pin. All other slaves have to configure their SSC1\_MRST pins as input. This way only one slave can put its data on the master's receive line. All other slaves can only receive

**CONFIDENTIAL**

**Synchronous Serial Controller on DSP**

data from the master. The master can select the slave device from which it expects data either by separate select lines, or by sending a special command to this slave. The selected slave then switches its SSC1\_MRST line to output, until it gets a de-selection signal or command.

- **The slaves use open drain output on SSC1\_MRST.** This results in a wired-AND connection of all slave outputs. On the receive line an external pull up is required in this case. Corruption of the data sent by the selected slave is avoided when all slaves which are not selected for transmission only send ones ('1'). Since this high level is only held through the pull up device, the selected slave can pull this line actively to a low level when transmitting a zero bit. The master selects the slave device from which it expects data either by separate select lines, or by sending a special command to this slave.

After performing all necessary initialization of the SSC, the serial interfaces can be enabled. For a master device, the alternate clock line will now switch to its programmed polarity. The alternate data line will either go to '0' or '1' until the first transfer starts.

When the serial interfaces are enabled, the master device can initiate the first data transfer by writing the transmit data into register **SSCDSP\_TXB**. This value is copied to the shift register and the selected first bit of the transmit data will be shifted to the SSC1\_MTSR line on the next clock from the baud rate generator (transmission only starts, if **SSCDSP\_CON.EN** = 1). Depending on the selected clock phase a clock pulse will also be generated on the SSC1\_SCLK line. At the opposite clock edge the master latches and shifts in the value detected at the input line SSC1\_MRST. Since the clock line is connected to all slaves, their shift registers will be shifted synchronously with the master's shift register, shifting out the data contained in the registers, and shifting in the data detected at the input line. After the pre-programmed number of clock pulses (determined by the data width selection) all slave shift registers contain the data transmitted by the master while the master's shift register holds the data of the selected slave. The content of all shift registers is copied into the receive buffers **SSCDSP\_RXB** (on Page 456) and the Receive Interrupt Line SSC1\_RX\_INT is activated.

A slave device will immediately output the selected first bit (MSB or LSB of the transfer data) at pin SSC1\_MRST when the content of the transmit buffer is copied into the slave's shift register. It will not wait for the next clock from the baud rate generator, as the master does.

*Note: As the serial clock is used for both transmit and receive path data transmission and reception run concurrently, that is, for each bit shifted out to the transmit line a single bit is shifted in from the receive line.*

Special care has to be taken at the initialization of the SSC1\_SCLK pin at the master to avoid undesired clock transitions which might disturb the other receivers. The state of the internal alternate output lines is '1' as long as the SSC is disabled. This alternate output signal is multiplexed with other alternate outputs in the port logic. Enabling the SSC with an idle-low clock (**SSCDSP\_CON.PO** = 0) drives the alternate data output and



CONFIDENTIAL

Synchronous Serial Controller on DSP

the port pin SSC1\_SCLK low as soon as the port logic is programmed accordingly. To avoid this, use the following sequence:

1. Select the clock idle level (**SSCDSP\_CON.PO**).
2. Load the port output latch with the desired clock idle level.
3. Switch the pin to output.
4. Enable the SSC (**SSCDSP\_CON.EN** = 1).
5. If **SSCDSP\_CON.PO** = 0: enable alternate data output.

The same mechanism as for selecting a slave for transmission (separate select lines or special commands) may also be used to select another device in the network as a master. In this case the previous master and the future master (previous slave) will have to change their operating mode (**SSCDSP\_CON.MS**) and the direction of their port pins.

### 8.7.4.2 Half-Duplex Operation

In a half-duplex configuration only one data line is required for both reception **and** transmission of data. The data exchange line is connected to both pins SSC1\_MTSR and SSC1\_MRST of each device, the clock line is connected to the SSC1\_SCLK pin.

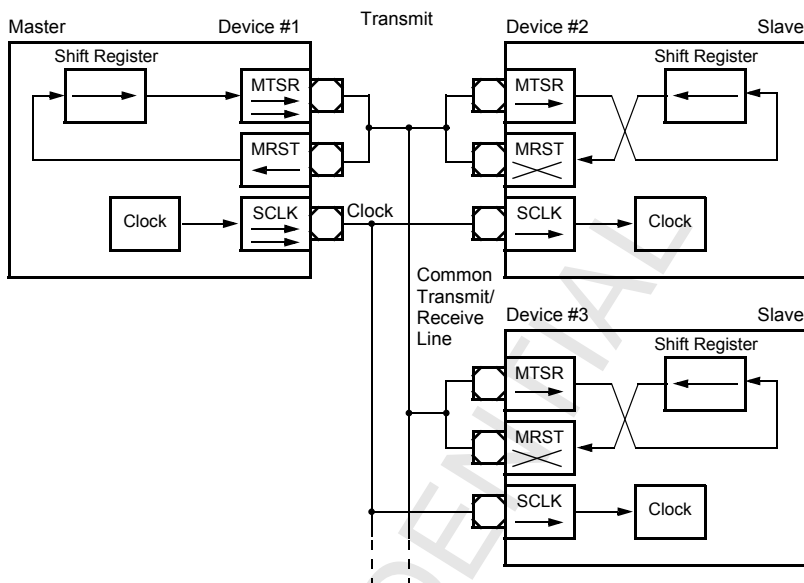
The master device controls the data transfer by generating the shift clock, while the slave devices is controlled by this clock. Due to the fact that all transmit and receive pins are connected to the same data line, serial data may be moved between arbitrary stations.

Similar to the duplex mode, there are **two ways to avoid collisions** on the data line:

- Only the transmitting device may enable its transmit pin driver.
- The non-transmitting devices use open drain output and only send ones.

Since the data inputs and outputs are connected, a transmitting device will shift in its own data at the input pin (SSC1\_MRST for a master device, SSC1\_MTSR for a slave). This way any corruptions on the common data line can be detected.

Figure 8-30 SSC Half-Duplex Configuration



## 8.7.5 General Operation

### 8.7.5.1 FIFO Operation

#### 8.7.5.1.1 Transmit FIFO

##### Standard Mode

The TXFIFO provides the following functionality:

- Enable/disable control.
- Programmable fill level for transmit interrupt generation.
- Fill level indication.
- FIFO clear (flush) operation.
- FIFO overflow error generation.

The 32-stage 16-bit transmit FIFO is controlled by the TXFIFO Control Register **SSCDSP\_TXFCON** (on Page 470). When bit **SSCDSP\_TXFCON.EN** is set, the TXFIFO is enabled. The interrupt trigger level set by **SSCDSP\_TXFCON.ITL** defines the fill level of the TXFIFO at which a Transmit Interrupt **SSC1\_TX\_INT** is generated. This

CONFIDENTIAL

Synchronous Serial Controller on DSP

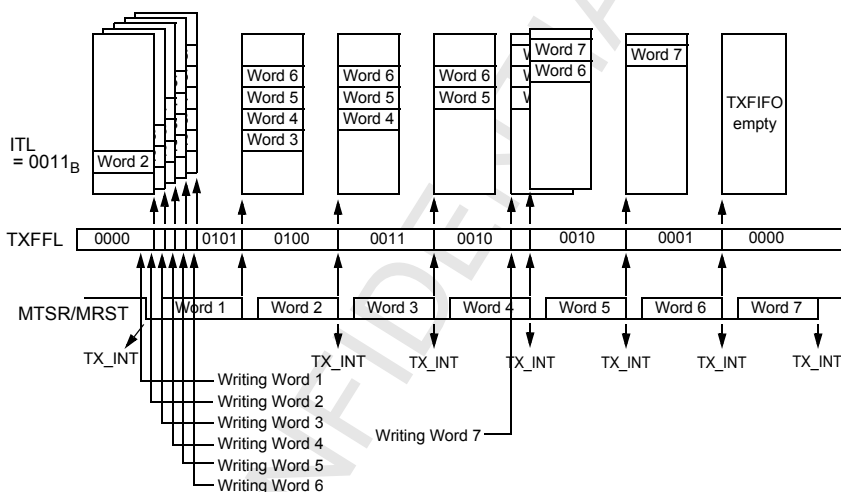
interrupt is always generated when the fill level of the transmit FIFO is equal to or less than the value stored in **SSCDSP\_TXFCON.ITL**.

Bit field **SSCDSP\_FSTAT (on Page 472).TXFFL** in the FIFO Status Register indicates the number of entries that are currently in the TXFIFO. The software can, for instance in the interrupt service routine, check how many bytes can still be written into the transmit FIFO via register **SSCDSP\_TXB (on Page 455)** without getting an overrun error.

The transmit FIFO cannot be accessed directly. All data write operations into the TXFIFO are executed by writing into the **SSCDSP\_TXB** register.

The example in **Figure 8-31** shows a typical transmit FIFO operation.

**Figure 8-31 Transmit FIFO Operation Example**



In this example seven words are transmitted via the SSC1\_MSTR pin or the SSC1\_MRST pin. The transmit FIFO interrupt trigger level **SSCDSP\_TXFCON.ITL** is set to 000011<sub>B</sub>. The first word written into the empty TXFIFO via **SSCDSP\_TXB** is directly transferred to the transmit shift register. Subsequently the following words (numbers 2 to 6) are written into the transmit FIFO.

After the transfer of word 3 from the TXFIFO to the transmit shift register of SSC, 3 words remain in the TXFIFO. The value of **SSCDSP\_TXFCON.ITL** is reached and a Transmit Interrupt SSC1\_TX\_INT will be generated at the end of the word 3 serial transmission. During the serial transmission of word 4 another word (word 7) is written into the TXFIFO (TXB write operation). Finally, after the start of the serial transmission of word 7, the TXFIFO is empty again.

If the TXFIFO is full and additional words are written into **SSCDSP\_TXB** and if bit **SSCDSP\_CON.TEN** is set, the Error Interrupt SSC\_ERR\_INT will be generated and bit

CONFIDENTIAL

Synchronous Serial Controller on DSP

**SSCDSP\_CON.TE** is set,. In this case the last data word written into the transmit FIFO is overwritten and the transmit FIFO fill level **SSCDSP\_TXFCON.TXFLL** is set to its maximum value (10 0000<sub>B</sub>).

The TXFIFO can be flushed or cleared by setting bit **SSCDSP\_TXFCON.FLU**. After this TXFIFO flush operation, the TXFIFO is empty and the transmit FIFO fill level

**SSCDSP\_TXFCON.TXFLL** is set to 00 0000<sub>B</sub>. A serial transmission in progress is not aborted by a receive FIFO flush operation.

*Note: The TXFIFO is flushed automatically at a reset operation of the SSC module and if the TXFIFO is disabled (resetting bit **SSCDSP\_TXFCON.EN**).*

### Transparent Mode

#### For E-GOLDradio V1.0:

*In the Transparent Mode interrupts are generated as in the standard mode. This means that there are no transmit interrupts generated when writing in the **SSCDSP\_TXB** register in the Transparent Mode.*

The Transparent Mode for the TXFIFO is enabled when bits **SSCDSP\_TXFCON.TMEN** and **SSCDSP\_TXFCON.EN** are set.

In the Transparent Mode, a specific interrupt generation mechanism is used for the Transmit Interrupt Request SSC1\_TX\_INT. The relevant conditions for interrupt generation in Transparent Mode are:

- FIFO fill levels
- Write operations on the data register **SSCDSP\_TXB**.

A Transmit Interrupt SSC\_TX\_INT is always generated after a word has been written into register **SSCDSP\_TXB**, if the TXFIFO is not full (**SSCDSP\_FSTAT.TXFLL** not equal to 100000<sub>B</sub>).

SSC\_TX\_INT is also generated after a TXFIFO flush operation or when the TXFIFO is enabled (**SSCDSP\_TXFCON.TMEN** and **SSCDSP\_TXFCON.EN** set). In these cases, the TXFIFO is empty and ready to be filled with data.

If the TXFIFO is full and an additional word is written into **SSCDSP\_TXB**, a SSC\_TX\_INT is not generated after the **SSCDSP\_TXB** write operation. In this case, the data byte last written into the transmit FIFO is overwritten and an Error Interrupt SSC\_ERR\_INT is generated, which sets bit **SSCDSP\_CON.TE**.

*Note: The Transmit FIFO Interrupt Trigger Level bit field **SSCDSP\_TXFCON.ITL** is not used in the Transparent Mode.*

### 8.7.5.1.2 Receive FIFO Operation

#### Standard Mode

The RXFIFO provides the following functionality:

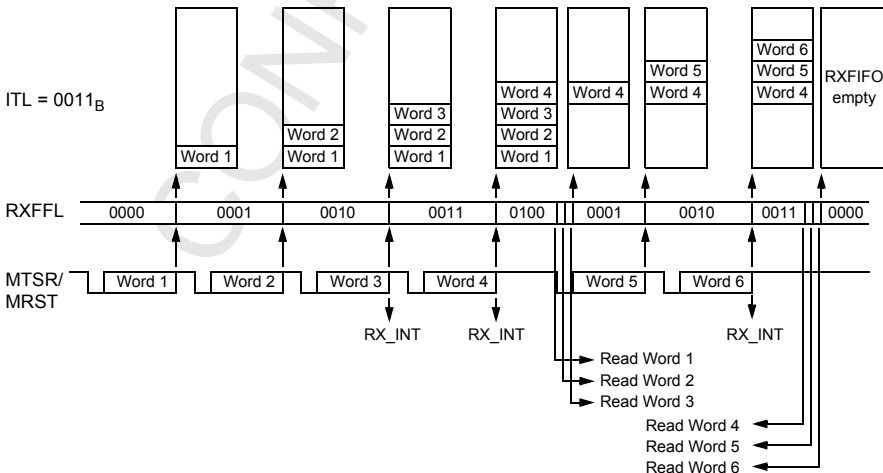
- Enable/disable control.
- Programmable fill level for receive interrupt generation.
- Fill level indication.
- FIFO clear (flush) operation.
- FIFO overflow error generation.

The receive FIFO comprises 32 stages of 16 bit each and is controlled by the RXFIFO Control Register **SSCDSP\_RXFCON** (on Page 468). When bit **SSCDSP\_RXFCON.EN** is set, the RXFIFO is enabled. The interrupt trigger level defined by **SSCDSP\_RXFCON.ITL** defines the fill level of RXFIFO at which a Receive Interrupt **SSC\_RX\_INT** is generated. **SSC\_RX\_INT** is always generated when the fill level of the receive FIFO is equal to or greater than the value stored in **SSCDSP\_RXFCON.ITL**.

Bit field **SSCDSP\_FSTAT.RXFFL** in the FIFO status register indicates the number of words that are currently contained in the FIFO and can be read out of the FIFO by a user program. The receive FIFO cannot be accessed directly. All data read operations from the RXFIFO are executed by reading the register **SSCDSP\_RXB** (on Page 456).

The example in Figure 8-32 shows a typical RXFIFO operation.

Figure 8-32 Receive FIFO Operation Example



**CONFIDENTIAL**

**Synchronous Serial Controller on DSP**

In this example, six words are received via the SSC1\_MTSR pin or the SSC1\_MRST pin. The receive FIFO interrupt trigger level **SSCDSP\_RXFCON.ITL** is set to 000011<sub>B</sub>. Therefore, the first Receive Interrupt **SSC\_RX\_INT** is generated after the reception of word 3 (RXFIFO is filled with three words).

After the reception of word 4, three words are read out of the receive FIFO. After this read operation, the RXFIFO still contains one word. **SSC\_RX\_INT** becomes active again after two more words (word 5 and 6) have been received (RXFIFO filled again with 3 words). Finally, the FIFO is empty again after three more read operations.

If the RXFIFO is full and additional bytes are received, the Receive Interrupt **SSC\_RX\_INT** and the Error Interrupt **SSC\_ERR\_INT** is generated with bit **SSCDSP\_CON.RE** set. In this case, the last data byte written to the receive FIFO is overwritten. At the overrun condition, the receive FIFO fill level **SSCDSP\_FSTAT.RXFFL** is set to 100000<sub>B</sub>. If a **SSCDSP\_RXB** read operation is executed with the RXFIFO enabled but empty, an Error Interrupt **SSC\_ERR\_INT** is generated with bit **SSCDSP\_CON.RE** set. In this case, the receive FIFO fill level **SSCDSP\_FSTAT.RXFFL** is set to 000000<sub>B</sub>.

If the RXFIFO is available but disabled (**SSCDSP\_RXFCON.EN** = 0) the receive operation is functionally equivalent to the receive operation of the SSC module without FIFO.

The RXFIFO can be flushed or cleared setting bit **SSCDSP\_RXFCON.FLU**. After this operation, the RXFIFO is empty and the receive FIFO fill level **SSCDSP\_FSTAT.RXFFL** is set to 000000<sub>B</sub>.

The RXFIFO is flushed automatically at a reset operation of the SSC module and if the RXFIFO is disabled (resetting bit **SSCDSP\_RXFCON.EN**).

### Transparent Mode

In Transparent Mode, a specific interrupt generation mechanism is used for Receive Interrupts **SSC\_RX\_INT**. The relevant conditions for interrupt generation in Transparent Mode are:

- FIFO fill levels.
- Read operations on the data register **SSCDSP\_RXB**.

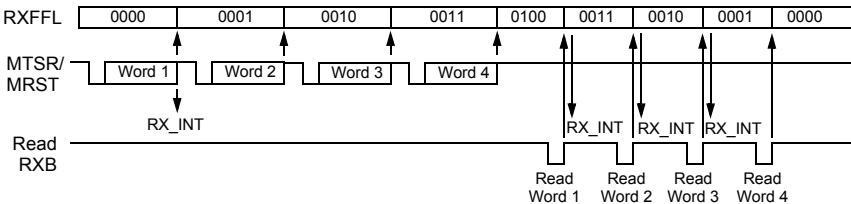
Transparent Mode for the RXFIFO is enabled when bits **SSCDSP\_RXFCON.TMEN** and **SSCDSP\_RXFCON.EN** are set.

If the RXFIFO is empty, a Receive Interrupt **SSC\_RX\_INT** is generated when the first byte is written into an empty RXFIFO (see [Figure 8-33](#)).

CONFIDENTIAL

Synchronous Serial Controller on DSP

Figure 8-33 Transparent Mode Receive FIFO Operation



If the RXFIFO is filled with at least one word, the generation of further Receive Interrupts depends on the read operations of register **SSCDSP\_RXB**. The Receive Interrupt **SSC\_RX\_INT** is always activated after an RXB read operation, if the RXFIFO still contains data (**SSCDSP\_FSTAT.RXFLL** is not equal to 000000<sub>B</sub>). If the RXFIFO is empty after a RXB read operation, no further receive interrupt will be generated.

If the RXFIFO is full (**SSCDSP\_FSTAT.RXFLL** = 100000<sub>B</sub>) and additional words are received, an Error Interrupt **SSC\_ERR\_INT** will be generated with bit **SSCDSP\_CON.RE** set. In this case, the last data word written to the receive FIFO is overwritten. If a RXB read operation is executed with the RXFIFO enabled but empty (underflow condition), an Error Interrupt **SSC\_ERR\_INT** will be generated as well, with bit **SSCDSP\_CON.RE** set.

If the RXFIFO is flushed in Transparent Mode, the software must make sure that a possibly pending receive interrupt is ignored.

*Note: The Receive FIFO Interrupt Trigger Level bit field **SSCDSP\_RXFCON.ITL** will be ignored in Transparent Mode.*

**CONFIDENTIAL**

**Synchronous Serial Controller on DSP**

### 8.7.5.1.3 Receive FIFO Control Register

**SSCDSP\_RXFCON**

**Receive FIFO Control Register**

**Reset value: 0100<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
RESERVED		ITL						RESERVED						TME N	FLU	EN

Field	Bits	Type	Description
<b>EN</b>	0	rw	<b>Receive FIFO Enable</b> 0 Receive FIFO is disabled. 1 Receive FIFO is enabled <i>Note: Resetting <b>EN</b> automatically flushes the receive FIFO.</i>
<b>FLU</b>	1	rw	<b>Receive FIFO Flush</b> 0 No operation. 1 Receive FIFO is flushed. <i>Note: Setting <b>FLU</b> clears bit field <b>SSCDSP_FSTAT.RXFFL</b>. <b>FLU</b> is always read as 0.</i>
<b>TMEN</b>	2	rw	<b>Receive FIFO Transparent Mode Enable</b> 0 Receive FIFO Transparent Mode is disabled. 1 Receive FIFO Transparent Mode is enabled. <i>Note: This bit will be ignored if the receive FIFO is disabled (<b>EN</b> = 0).</i>



**CONFIDENTIAL**

**Synchronous Serial Controller on DSP**

Field	Bits	Type	Description
ITL	13:8	rw	<p><b>Receive FIFO Interrupt Trigger Level</b></p> <p>A receive interrupt request (SSC_RX_INT) is generated after the reception of a byte when the fill level of the receive FIFO is equal to or greater than ITL.</p> <p>000000Reserved. Do not use this combination.</p> <p>000001Interrupt trigger level is set to 1.</p> <p>000010Interrupt trigger level is set to 2.</p> <p>...</p> <p>111110Interrupt trigger level is set to 62.</p> <p>111111Interrupt trigger level is set to 63.</p> <p><i>Note: In Transparent Mode this bit field is ignored.</i></p> <p><i>Note: Combinations defining an interrupt trigger level greater than the configured FIFO size must not be used.</i></p>
<b>RESERVED</b>	7:3, 15:14	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**Synchronous Serial Controller on DSP**

### 8.7.5.1.4 Transmit FIFO Control Register

**SSCDSP\_TXFCON**

**Transmit FIFO Control Register**

**Reset value: 0100<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
RESERVED		ITL						RESERVED						TME N	FLU	EN

Field	Bits	Type	Description
<b>EN</b>	0	rw	<b>Transmit FIFO Enable</b> 0 Transmit FIFO is disabled. 1 Transmit FIFO is enabled. <i>Note: Resetting <b>EN</b> automatically flushes the transmit FIFO.</i>
<b>FLU</b>	1	rw	<b>Transmit FIFO Flush</b> 0 No operation. 1 Transmit FIFO is flashed. <i>Note: Setting <b>FLU</b> clears bit field <b>SSCDSP_FSTAT.TXFFL</b>. <b>FLU</b> is always read as 0.</i>
<b>TMEN</b>	2	rw	<b>Transmit FIFO Transparent Mode Enable</b> 0 Transmit FIFO Transparent Mode is disabled. 1 Transmit FIFO Transparent Mode is enabled. <i>Note: This bit will be ignored if the receive FIFO is disabled (<b>EN</b> = 0).</i>

**CONFIDENTIAL**

**Synchronous Serial Controller on DSP**

Field	Bits	Type	Description
ITL	13:8	rw	<p><b>Transmit FIFO Interrupt Trigger Level</b></p> <p>A transmit interrupt request (SSC_TX_INT) is generated after the transfer of a byte when the fill level of the transmit FIFO is equal to or less than ITL.</p> <p>000000Reserved. Do not use this combination.</p> <p>000001Interrupt trigger level is set to 1.</p> <p>000010Interrupt trigger level is set to 2.</p> <p>...</p> <p>111110Interrupt trigger level is set to 62.</p> <p>111111Interrupt trigger level is set to 63.</p> <p><i>Note: In the Transparent Mode this bit field will be ignored.</i></p> <p><i>Note: Combinations defining a interrupt trigger level greater than the configured FIFO size must not be used.</i></p>
RESERVED	7:3, 15:14	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**Synchronous Serial Controller on DSP**

### 8.7.5.1.5 FIFO Status Register

**SSCDSP\_FSTAT**

**FIFO Status Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		TXFFL						RESERVED		RXFFL					

Field	Bits	Type	Description
<b>RXFFL</b>	5:0	rh	<b>Receive FIFO Filling Level</b> 000000Receive FIFO is filled with 0 words. 000001Receive FIFO is filled with 1 word. ... 111110Receive FIFO is filled with 62 words. 111111Receive FIFO is filled with 63 words. <i>Note: <b>RXFFL</b> is cleared after a receive FIFO flush operation.</i>
<b>TXFFL</b>	13:8	rh	<b>Transmit FIFO Filling Level</b> 000000Transmit FIFO is filled with 0 words. 000001Transmit FIFO is filled with 1 word. ... 111110Transmit FIFO is filled with 62 words. 111111Transmit FIFO is filled with 63 words. <i>Note: <b>TXFFL</b> is cleared after a receive FIFO flush operation.</i>
<b>RESERVED</b>	7:6, 15:14	r	Reserved; these bits must be left at their reset values.

*Note: To this register has a shadow register exists. Thus the content does not change during a two wait states read access. New register values can be read with a delay of one clock cycle.*

CONFIDENTIAL

Synchronous Serial Controller on DSP

### 8.7.5.2 Port Control

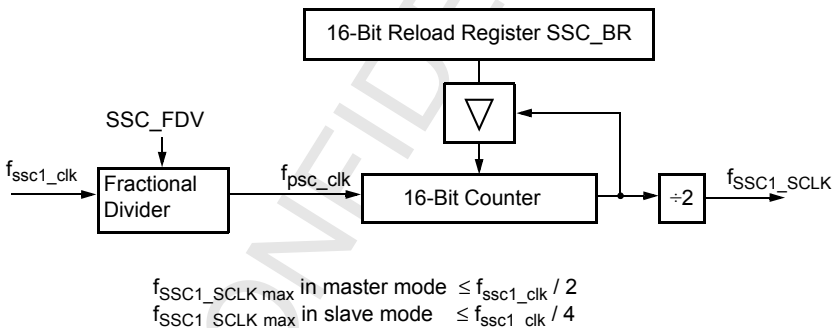
The SSC interface uses a total of three pins. Pin SSC1\_SCLK serves as the clock line, while pins SSC1\_MRST (Master Receive/Slave Transmit) and SSC1\_MTSR (Master Transmit/Slave Receive) represent the serial data input/output lines.

The direction of the port lines depends on the operating mode. The SSC will automatically use the correct alternate input or output line of the ports when switching modes. However, the direction of the port pins must be programmed by the user. Using the open drain output feature of the port lines helps to avoid conflicting levels on the bus line and reduces the need for hardwired hand-shaking or slave select lines. In this case it is not always necessary to switch the direction of a port pin.

### 8.7.5.3 Baud Rate Generation

The Synchronous Serial Interface Controller SSC has its own dedicated baud rate generator, consisting of a 16-bit timer with 16-bit reload capability and a prescaler. This baud rate generator is independent from other timers. In addition to [Figure 8-27 \(on page 449\)](#), [Figure 8-34](#) shows the baud rate generator of the SSC in more detail.

Figure 8-34 SSC Baud Rate Generator



The prescaler derives its output clock  $\text{psc\_clk}$  from the module clock  $\text{ssc1\_clk}$ . The fractional divider divides  $\text{ssc1\_clk}$  by a fraction of  $n/512$  for any value of  $n$  from 0 to 511. If  $n = 0$ , the divider ratio is 1, which means that  $f_{\text{psc\_clk}} = f_{\text{ssc1\_clk}}$ . The value of  $n$  is programmable via the register [SSCDSP\\_FDV \(on Page 475\)](#).

*Note: Never write to [SSCDSP\\_FDV](#) during SSC is enabled ([SSCDSP\\_CON.EN](#) = 1).*

The shift clock  $\text{SSC1\_SCLK}$  is generated with a duty cycle of 50% by the 16-bit timer clocked with the pre-scaled clock  $f_{\text{psc\_clk}}$ . The timer is counting down. Register [SSCDSP\\_BR \(on Page 474\)](#) is the dual-function Baud Rate Generator/Reload register. Reading [SSCDSP\\_BR](#), while the SSC is enabled, returns the content of the timer.

**CONFIDENTIAL**

**Synchronous Serial Controller on DSP**

Reading **SSCDSP\_BR**, while the SSC is disabled, returns the programmed reload value. In this mode the desired reload value can be written to **SSCDSP\_BR**.

*Note: Never write to **SSCDSP\_BR**, while the SSC is active.*

The baud rate of the SSC is determined by the frequency of the shift clock SSC1\_SCLK. Thus, by setting the **SSCDSP\_BR** and **SSCDSP\_FDV** appropriately, the required baud rate can be derived from the given module clock frequency with very little deviation.

*Note: The resulting minimum period of SSC1\_SCLK may be smaller than 1/f<sub>SSC1\_SCLK</sub> if the fractional divider is used. The minimum length is the largest integer multiple of the f<sub>ssc\_clk</sub> period which is less or equal 1/f<sub>SSC1\_SCLK</sub>. The baudrate is limited by the minimum period and the timing constraints given for the SSC block in [Section 13.2.1.2.8 SSC \(on Page 1356\)](#).*

The formulas below describe the resulting baud rates for the different settings of the baud rate generator:

$$\text{Baud Rate}_{\text{SSC}} = \frac{f_{\text{ssc1\_clk}}}{2 \cdot (\text{<BR>} + 1)} \quad \text{for } \text{<FDV>} = 0$$

$$\text{Baud Rate}_{\text{SSC}} = \frac{\text{<FDV>}}{512} \cdot \frac{f_{\text{ssc1\_clk}}}{2 \cdot (\text{<BR>} + 1)} \quad \text{for } \text{<FDV>} > 0$$

<BR> and <FDV> represents the content of the reload register **SSCDSP\_BR** and **SSCDSP\_FDV**.

## 8.7.5.4 Baud Rate Generation Registers

### 8.7.5.4.1 Baud Rate Timer Reload Register

The SSC baud rate timer reload register **SSCDSP\_BR** contains the 16-bit reload value for the baud rate timer.

#### **SSCDSP\_BR**

#### **Baud Rate Timer Reload Register**

**Reset Value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR_VALUE															

Field	Bits	Type	Description
<b>BR_VALUE</b>	15:0	rw	<b>Baud Rate Timer/Reload Register Value</b> Reading BR returns the 16-bit content of the baud rate timer. Writing BR loads the baud rate timer reload register with BR_VALUE.

**CONFIDENTIAL**

**Synchronous Serial Controller on DSP**

*Note: The register **SSCDSP\_BR** is connected to a clock independent from the module clock **ssc1\_clk**. Thus this register can be written, even if the bit **SSCDSP\_CON.CLKON** is not set.*

### 8.7.5.4.2 Fractional Divider Register

The fractional divider register **SSCDSP\_FDV** contains the 9-bit value for the prescaler.

#### SSCDSP\_FDV

#### Fractional Divider Register

**Reset Value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								FD_VALUE							

Field	Bits	Type	Description
<b>FD_VALUE</b>	8:0	rw	<b>Fractional Divider</b> 0: Fractional Divider Off. 1..511: Fractional Divider On, Factor = FDV/512.
<b>RESERVED</b>	15:9	r	Reserved; these bits must be left at their reset values.

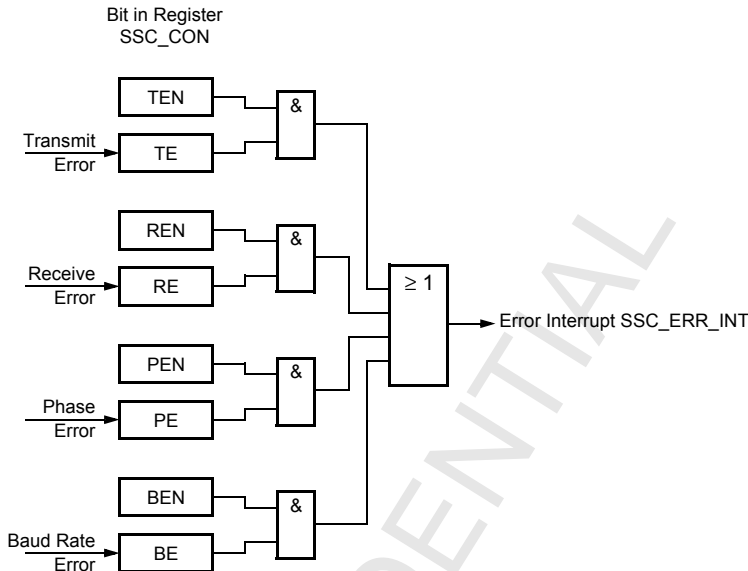
*Note: The register **SSCDSP\_FDV** is driven by a clock independent from the module clock **ssc1\_clk**. Thus this register can be written to, even if the bit **SSCDSP\_CON.CLKON** is not set.*

### 8.7.6 Error Detection Mechanisms

The SSC is able to detect four different error conditions. Receive Error and Phase Error are detected in all modes, Transmit Error and Baud Rate Error only apply to slave mode. When an error is detected, the respective error flag is set and an error interrupt request will be generated by activating the **SSC\_ERR\_INT** line (see [Figure 8-35](#)). The error interrupt handler may then check the error flags to determine the cause of the error. The error flags are not reset automatically but must be cleared by software. This allows servicing of some error conditions via interrupt, while the others may be polled by software.

*Note: The error interrupt handler must clear the associated (enabled) error flag(s) to prevent repeated interrupt requests.*

Figure 8-35 SSC Error Interrupt Control



A **Receive Error** (Master or Slave mode) is detected, when a new data frame is completely received, but the previous data was not fetched from the receive buffer SSCDSP\_RXB. This condition sets the error flag **SSCDSP\_CON.RE**. When the receive error interrupt is enabled via **SSCDSP\_CON.REN** the Error Interrupt Request Line SSC\_ERR\_INT will also be activated. The old data in the receive buffer **SSCDSP\_RXB** will be overwritten with the new value and is irretrievably lost.

A **Phase Error** (Master or Slave mode) is detected, when the incoming data at pin SSC1\_MRST (master mode) or SSC1\_MTSR (slave mode), sampled with the same frequency as the module clock, changes between one cycle before and two cycles after the latching edge of the shift clock signal SSC1\_SCLK. This condition sets the error flag **SSCDSP\_CON.PE**. When the phase error interrupt is enabled via **SSCDSP\_CON.PEN** the error interrupt request line SSC\_ERR\_INT will also be activated.

A **Baud Rate Error** (Slave mode only) is detected, when the incoming clock signal deviates from the programmed baud rate by more than a factor of two, that is, it is either more than double or less than half the expected baud rate. This condition sets the error flag **SSCDSP\_CON.BE**. When the baud rate error interrupt is enabled via **SSCDSP\_CON.BEN** the Error Interrupt Request Line to SSC\_ERR\_INT will also be set. Using this error detection capability requires that the baud rate generator at the slave is set to the same baud rate as the master device. This feature detects additional or missing pulses on the clock line (within a certain frame).



## CONFIDENTIAL

## Synchronous Serial Controller on DSP

*Note: If this error condition occurs and bit **SSCDSP\_CON.REN** = 1 an automatic reset of the SSC will be performed. This is done to re-initialize the SSC if too few or too many clock pulses have been detected.*

A **Transmit Error** (Slave mode only) is detected when a transfer was initiated by the master (shift clock starts running), but the transmit buffer **SSCDSP\_TXB** of the slave was not updated since the last transfer. This condition sets the error flag **SSCDSP\_CON.TE**. When the transmit error interrupt is enabled via **SSCDSP\_CON.TEN** the Error Interrupt Request Line SSC\_ERR\_INT will also be set. After each transmission the buffer contains the data received during the last transfer. This data will be sent if a transmission is started before the transmit buffer has been updated. As a result data on the transmit/receive line might be corrupted in half-duplex mode (open drain configuration). This can happen if the slave, at which the transmit error occurs, is not selected for transmission. As already mentioned half-duplex mode requires that slaves not selected for transmission only shift out ones, that is, their transmit buffers must be loaded with FFFF<sub>H</sub> prior to each transfer.

*Note: A slave with push/pull output drivers, which is not selected for transmission, normally has its output drivers switched off. However, in order to avoid possible conflicts or misinterpretations, it is recommended to load the slave transmit buffer before each transfer.*

The cause of an error interrupt request (receive, phase, baud rate, transmit error) can be identified by the error status flags in control register **SSCDSP\_CON**.

*Note: The error status flags **SSCDSP\_CON.TE**, **SSCDSP\_CON.RE**, **SSCDSP\_CON.PE**, and **SSCDSP\_CON.BE** are not reset automatically by hardware and must be cleared by software.*

CONFIDENTIAL

**CONFIDENTIAL**

**GMSK Modulator**

## 8.8 GMSK Modulator

History	
Design Spec. Current version:	Rev. 1.06, 2005-11-04
	Previous version: Rev. 1.05, 2005-08-02
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.06	
all	Remove Analog path WS00009088

### System Integration:

- Supply domains:
  - Digital parts of the modulator unit: VDD\_DSP
  - Chip internal interfaces:
- Clock domain:
  - gclk\_dsp\_per for digital part
- Bus domain: TEAKLite Z-Bus interface
- Interrupt sources: One interrupt from modulator HW to TEAKLite: MODU
- Other Interfaces: TXON, CODON
- Chip external interfaces (see [Chapter 3 Pin Descriptions \(on Page 49\)](#) for pin configuration options):
  - BB\_I, BB\_IX, BB\_Q, BB\_QX.

### 8.8.1 Functional Overview

The GMSK modulator is suitable for systems which meet the ETSI GSM recommendations.

The DSP writes the bits which are to be transmitted into the dual port RAM. The modulator reads the data bits out of this dual port RAM, performs modulating and filtering and corrects the center frequency, the amplitude and the offset of the complex I/Q-signal.

### 8.8.2 Register Overview

CONFIDENTIAL

GMSK Modulator

**Table 8-29 Modulator Register List**

Register Group	Register Name	Register Symbol
Control and Status Registers	Control Register	<a href="#">MOD_CTRL (on Page 484)</a>
	Status Register	<a href="#">MOD_STAT (on Page 485)</a>
	Interrupt Address Register	<a href="#">MOD_INT_ADDR (on Page 485)</a>
	Offset Correction Register I-Component	<a href="#">MOD_OCI (on Page 485)</a>
	Offset Correction Register Q-Component	<a href="#">MOD_OCQ (on Page 487)</a>
	Amplitude Correction Register I-Component	<a href="#">MOD_ACI (on Page 488)</a>
	Amplitude Correction Register Q-Component	<a href="#">MOD_ACQ (on Page 489)</a>
	Frequency Correction Register	<a href="#">MOD_FC (on Page 490)</a>

CONFIDENTIAL

GMSK Modulator

### 8.8.3 Modulator Block Overview

The complete modulator consists of the following parts:

- Dual port RAM
- GMSK modulator
- I/Q-amplitude-correction + I/Q-ramping
- I/Q-offset-correction
- Interpolation & noise shaping
- The analog blocks as described in [Section 8.8.7 Analog Domain of the Modulator \(on Page 490\)](#).

#### 8.8.3.1 GMSK Modulator

The input values of the modulation unit are bits (0 or 1) which are read from modulation RAM with a rate of 270.83 kHz. Differential encoding is performed first, for example, a bit change is encoded as -1 and no bit change as +1.

The resulting data stream is filtered by a gaussian filter with a BxT product of 0.3. The output signal of the filter is integrated and frequency corrected with the correction value of the MOD\_FC-register. Finally, the phase information is converted to a complex baseband signal with an in-phase (I) and a quadrature (Q) component by applying a cosine and a sine function.

#### 8.8.3.2 IQ Amplitude and Offset Correction

The IQ amplitude and offset correction is necessary to compensate possible impairments of the analog modulator parts (DACs, post filters) and can be used for the correction of the RF-impairments as well. The amplitude correction is performed using a multiplier in the I- and Q-path, while the offset correction is done by an adder in each path.

The point of measurement decides which impairments can be corrected. If the measurements are taken at the antenna output the impairments of the complete chain including RF chip and PA are corrected. If the measurements are done at the I/Q-pins at the baseband chip only the impairments of the analog parts in the baseband chip are corrected.

*Note: If the offset correction is used the amplitude correction must be set according to the equation described in the [MOD\\_OCI \(on Page 485\)](#) and [MOD\\_OCQ \(on Page 487\)](#) register description to prevent an overflow of the digital words.*

### 8.8.4 Clocking Issues of the Modulator

The clock (mod\_clk) for the digital modulator blocks is active, when CODON is switched on. For debug and test purposes the clock can also be switched on by the TEAKLite

**CONFIDENTIAL**

**GMSK Modulator**

setting the bit **MOD\_CTRL (on Page 484).MSWACT** to 1. After CODON returns to 0 the clock for the modulation unit is switched off after 1 or 2 clock cycles (clock frequency = 26 MHz) by hardware or by resetting the bit **MSWACT**.

### 8.8.5 Programming Description

The modulator is activated at the rising edge of the timer signal CODON (from the GSM timer unit) or by the **MSWACT** bit. Once the modulator clock is activated the HW reads the modulating bits from dual port RAM starting at address 0<sub>H</sub>. The address counter is incremented after each read access. The modulator RAM acts as a circular buffer. This means that the address pointer wraps around to 0 after reading from the highest address. If the address is equal to the one specified in the register MOD\_INT\_ADDR, the hardware sets the interrupt bit **INT\_FINTA0.MODU** in register and generates an interrupt (if enabled within the interrupt controller). The modulator performs the modulation of the bits as long as signal CODON remains high (or **MSWACT** = 1).

The data is written to the modulator RAM by the TEAKLite. Burst building is done by firmware. The firmware writes the complete burst including the dummy bursts. For test cases it is also possible to write the data from the MCU to the modulator using the shared memory protocol between MCU and TEAKLite.

#### Modulator Dual Port RAM Access and Timing Control

When the modulator starts with reading the data out of the dual port RAM it must be ensured by firmware, that valid data is already written into the RAM. Multi-slot transmission is also possible due to the Dual Port RAM interface. The **Figure 8-36** and **Figure 8-37** give a timing overview for mono and multi slot applications.

**Figure 8-36 Mono Slot Application**

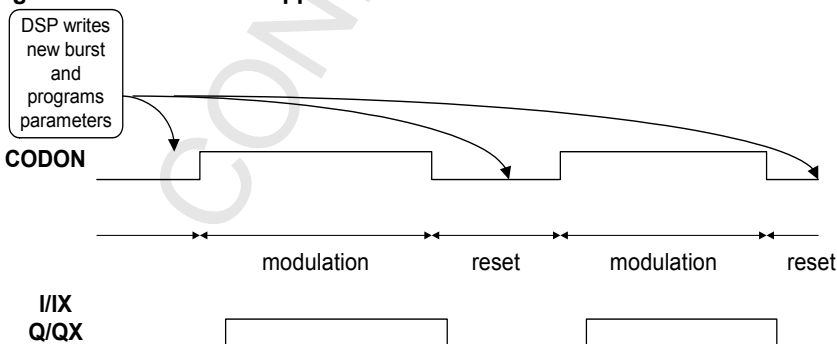
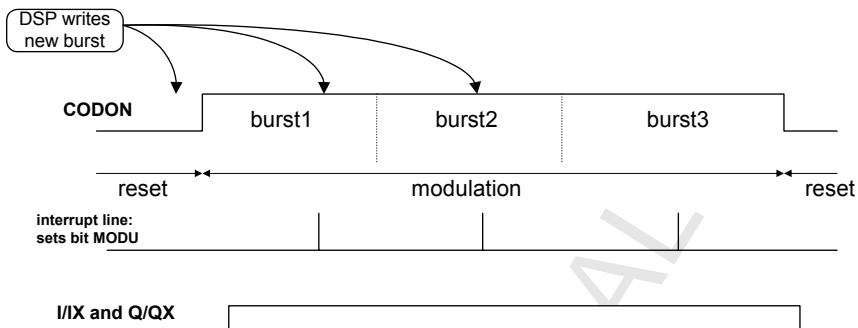


Figure 8-37 Multi Slot Application



The amplitude and offset can be corrected for the I- and Q-component separately according to the register values of **MOD\_OCI** (on Page 485), **MOD\_OCQ** (on Page 487), **MOD\_ACI** (on Page 488), and **MOD\_ACQ** (on Page 489).

*Note: PA ramp up and PA ramp down of the RF transmit power is initiated by the GSM timer unit where the start and stop is set by separate timing triggers.*

### Programming Sequence for TX Modulator

The following list describes the sequence which must be performed by the MCU to get the TX path working after a boot up of E-GOLDradio:

1. Enable the signal CODON by programming the GSM timer unit (**Section 10.11.3 GSM Timer Decoder** (on Page 833)). This signal enables the digital modulator parts and the DSP-firmware.
2. Enable the signal TXON by programming the GSM timer unit (**Section 10.11.3**) to enable the analog modulator parts.
3. Refer to the E-GOLDradio DSP firmware description about the command structure from MCU to TEAKLite and the burst building of the data bits.
4. The burst building timing must be controlled by correct setting of the CODON and TXON signals.
5. To disable the modulator in a correct way, first the TXON and then the CODON have to be switched off.

*Note: The signal CODON enables the digital and TXON enables the analog blocks of the PA power ramping HW as well.*

**CONFIDENTIAL**

**GMSK Modulator**

### 8.8.6 Register Description of the Modulator

All registers are read/write with two wait states also if the modulator is inactive or the clock is switched off.

#### **MOD\_CTRL**

**Modulator Control Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED							MSW ACT	RESERVED							IQS WAP

Field	Bits	Type	Description
<b>IQSWAP</b>	0	rw	0 No action 1 The I- and Q-output of the GMSK-modulator are swapped. This feature is used for dual band applications in which one RF band is generated by means of high side injection and the other band by means of low side injection.  <i>Note: This bit must not be changed while the modulator is active.</i>
<b>MSWACT</b>	8	rw	<b>Modulator Clock Enable</b> 0 Modulator clock is switched off. 1 Modulator clock is switched on.  <i>Note: This bit activates the modulator clock independently from the CODON-signal ("OR") and is used for test purposes. If MSWACT is reset (and CODON is inactive) the clock is stopped.</i>  <i>Note: For normal operation this bit must be set to zero. Otherwise the modulator RAM pointer will not be reset correctly.</i>
<b>RESERVED</b>	1:7, 9:15	r	Reserved; these bits must be left at their reset values.



**CONFIDENTIAL**

**GMSK Modulator**

## MOD\_STAT

**Modulator Status Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														MSTAT	

Field	Bits	Type	Description
MSTAT	0	rh	Modulator Status (set and reset by modulator hardware) 0 Modulator is inactive. 1 Modulator is active.
RESERVED	15:1	r	Reserved; these bits must be left at their reset values.

## MOD\_INT\_ADDR

**Modulator Interrupt Address Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED							MINT_ADDR								

Field	Bits	Type	Description
MINT_ADDR	8:0	rw	Interrupt address for the dual port RAM in the range of 0 to 511. If the dual port RAM address matches the value in <b>MINT_ADDR</b> and the <b>INT_FINTA0.EMODU</b> bit is set, the modulator generates a TEAKLite interrupt.
RESERVED	15:9	r	Reserved; these bits must be left at their reset values.

## MOD\_OCI

**Modulator Offset Correction Register I-Component**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				OCIV											

**CONFIDENTIAL**

**GMSK Modulator**

Field	Bits	Type	Description
<b>OCIV</b>	11:0	rw	<p>Offset correction value (I component) in the range -2047to 2047:</p> $I = I + OCIV$ <p>This register can always be read. Writing to this register is only allowed if the modulator is inactive.</p> <p><i>Note: The following sum must not exceed the overall range of 0 to 2047:</i></p> $2047 \times (ACIV \div 256) +  OCIV $
<b>RESERVED</b>	15:12	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**GMSK Modulator**

## MOD\_OCQ

**Modulator Offset Correction Register Q-Component**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				OCQV											

Field	Bits	Type	Description
<b>OCQV</b>	11:0	rw	Offset correction value (Q component) in the range -2047to 2047: $Q = Q + OCQV$ <p>This register can always be read. Writing to this register is only allowed if the modulator is inactive.</p> <p><i>Note: The following sum must not exceed the overall range of 0 to 2047:</i></p> $2047 \times (ACQV \div 256) +  OCQV $
<b>RESERVED</b>	15:12	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**GMSK Modulator**

**MOD\_ACI**

**Modulator Amplitude Correction Register I-Component**

**Reset value: 00FF<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								ACIV							

Field	Bits	Type	Description
<b>ACIV</b>	7:0	rw	<p>Amplitude correction value (I component) in the range of 0 to 255:</p> $I = I \times (ACIV \div 256)$ <p>This register can always be read. Writing to this register is only allowed if the modulator is inactive.</p> <p><i>Note: The following sum must not exceed the overall range of 0 to 2047:</i></p> $2047 \times (ACIV \div 256) +  OCIV $
<b>RESERVED</b>	15:8	r	Reserved; these bits must be left at their reset values.

**MOD\_ACQ**

**Modulator Amplitude Correction Register Q-Component**

**Reset value: 00FF<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								ACQV							

Field	Bits	Type	Description
<b>ACQV</b>	7:0	rw	<p>Amplitude correction value (Q component) in the range of 0 to 255:</p> $Q = Q \times (ACQV \div 256)$ <p>This register can always be read. Writing to this register is only allowed if the modulator is inactive.</p> <p><i>Note: The following sum must not exceed the overall range of 0 to 2047:</i></p> $2047 \times (ACQV \div 256) +  OCQV $
<b>RESERVED</b>	15:8	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**GMSK Modulator**

## MOD\_FC

**Modulator Frequency Correction Register For GMSK**

**Reset value: 0000<sub>H</sub>**

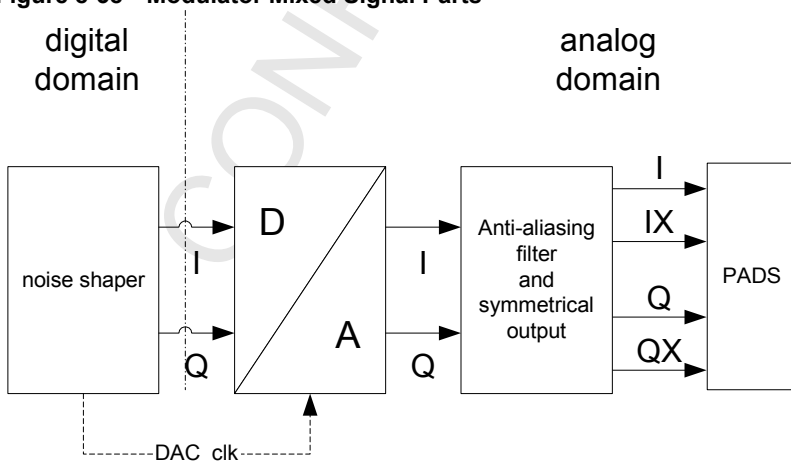
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>				<b>FCV</b>											

Field	Bits	Type	Description
<b>FCV</b>	11:0	rw	Frequency correction value in the range -2047 to 2047 for GMSK-modulation: frequency = frequency + FCV x 16.530355 Hz Writing to this register is only allowed if the modulator is inactive. <i>Note: The maximum frequency correction value is ± 33.8376 kHz</i>
<b>RESERVED</b>	15:12	r	Reserved; these bits must be left at their reset values.

### 8.8.7 Analog Domain of the Modulator

#### 8.8.7.1 Mixed Signal Parts of the Modulator

**Figure 8-38 Modulator Mixed Signal Parts**



The blocks in the analog domain of the E-GOLDradio modulator are for I and Q:

CONFIDENTIAL

GMSK Modulator

1. Digital to analog conversion-
2. Anti aliasing filter with an analog filter-
3. The filtered data is provided at the symmetrical output pins I, IX and Q, QX

The output signals after the analog post filters form double balanced analog quadrature components (I/IX and Q/QX) of a baseband signal with GMSK modulation. The common mode voltage of the single ended output signals I, IX, Q and QX can be selected by **ANA\_CTRL2.TREF** for maximum flexibility.

*Note: The signal TXON enables the power supply of the analog macro of the modulator. If TXON is disabled and output of the transmit path is set to high impedance.*

*Note: The signals I/IX and Q/QX of the transmit path are multiplexed together with the according signals of the receive path to the same pins, I/IX and Q/QX. The selection of the TX or RX path is done by the GSM timer signals TXON and RXON which must not be activated at the same time.*

### 8.8.7.2 Spectrum Requirements According to GSM Standard

The requirements of the spectrum mask are defined in the GSM recommendations. The table below shows the definition of the spectrum mask for the overall GSM requirements for GMSK. The data is valid for the following mobile power classes:

GSM400 & GSM900 & GSM850: power class = 4 (2 W, 33 dBm)

DGS1800 & PCS1900: power class = 1 (1 W, 30 dBm)

**CONFIDENTIAL**

**GMSK Modulator**

**Table 8-30 GSM Requirements: TX Spectrum Due to Modulation**

Frequency/kHz	RBW/kHz	GSM 900/dB	GSM 1800/dB
100.0	30.0	0.5	0.5
200.0	30.0	-30.0	-30.0
250.0	30.0	-33.0	-33.0
400.0	30.0	-60.0	-60.0
600–1800	30.0	-60.0	-60.0
1800–3000	100.0	-63.0	-65.0
3000–6000	100.0	-65.0	-65.0
> 6000	100.0	-71.0	-73.0

**GSM Requirements: Spurious Emission for GSM900 and GSM1800**

(GSM recommendations 05.05, chapter 4.3.3)

- 925..935 MHz: spurious emission  $\leq -67$  dBm (RBW: 100 kHz)
- 935..960 MHz: spurious emission  $\leq -79$  dBm (RBW: 100 kHz)
- 1805..1880 MHz: spurious emission  $\leq -71$  dBm (RBW: 100 kHz)

The highest TX transmit frequency in GSM900 is 915.0 MHz and for DCS1800-1785.0 MHz. This leads to the following requirements for a mobile with power class 4- (33dBm) for GSM900 and with a power class 1 (30 dBm) for DCS1800:–

**Table 8-34 GSM Requirements: Spurious Emissions**

Frequency/MHz	RBW/kHz	GSM 900/dBc	GSM 1800/dBc
10.0	100	-100.0	not defined
20.0	100	-112.0	-101.0

### 8.8.8 Baseband Spectrum Requirements

The requirements for the output spectrum of the baseband chip are calculated using the worst case data in the chapter above and an additional margin for the RF impairments.

*Note: The values for the baseband are calculated under the assumption, that a one pole filter with a cutoff frequency of 1 MHz attenuates the noise spectrum for higher frequencies.*

The BW correction for 1 Hz, 30 kHz and 100 kHz in dB for noise power:



**CONFIDENTIAL**

**GMSK Modulator**

**Table 8-32 Correction Values for Bandwidth**

<b>RBW</b>	<b>1 Hz</b>	<b>30 kHz</b>	<b>100 kHz</b>
<b>1 Hz</b>	0 dB	44.77 dB	50.0 dB
<b>30 kHz</b>	-44.77 dB	0 dB	5.23 dB
<b>100 kHz</b>	-50.0 dB	-5.23 dB	0 dB

**Table 8-33 Definition of the GSM and Baseband Templates (BB)**

<b>Freq / Hz</b>	<b>GSM / dB RBW 1Hz</b>	<b>BB / dB RBW 1Hz</b>	<b>BB / dB RBW 30kHz</b>	<b>BB / dB RBW 100kHz</b>
100k	-44.3	-44.3	0.5	not defined
200k	-74.8	-76.8	-32.0	not defined
250k	-77.8	-79.8	-35.0	not defined
400k - 600k	-104.8	-111.2	-66.4	-61.2
600k - 1800k	-104.8	-111.2	-66.4	-61.2
1800k - 3M	-113.0	-113.8	-69.0	-63.8
3M - 6M	-115.0	-115.8	-71.0	-65.8
6M - 10M	-121.0	-115.8	-71.0	-65.8

*Note: The values in [Table 8-33](#):*

- *Are minimum values*
- *Are defined relative to a measurement in 30 kHz on the carrier*
- *The values above 1.8 MHz for BB are proved by design and not tested in production.*

**Table 8-34 Definition of the GSM and BASEBAND Templates (BB)**

<b>Freq / Hz</b>	<b>GSM / dBc RBW 1Hz</b>	<b>BB / dBc RBW 1Hz</b>	<b>BB / dBc RBW 30kHz</b>	<b>BB / dBc RBW 100kHz</b>
10M - 20M	-150.0	-145.0	-100.2	-95.0
20M	-162.0	-152.0	-107.2	-102.0
80M	-162.0	-146.0	-94.2	-89.0

CONFIDENTIAL

GMSK Modulator

Note: The values in [Table 8-34](#):

- Are minimum values
- Are measured relative to the entire carrier power
- The values for BB are proved by design and not tested in production.

**Figure 8-39 GSM Spectrum Mask for the Baseband Chip (RBW = 30 kHz)**

GSM/BB spectrum mask (RBW 30kHz)

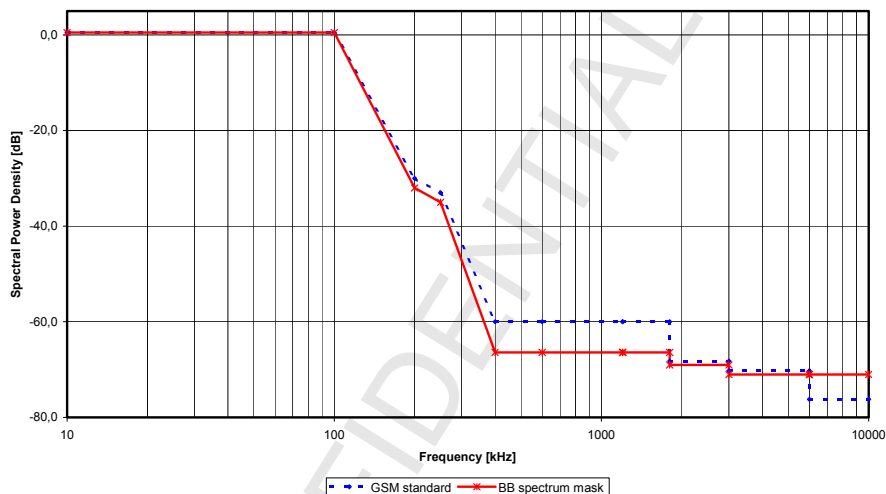
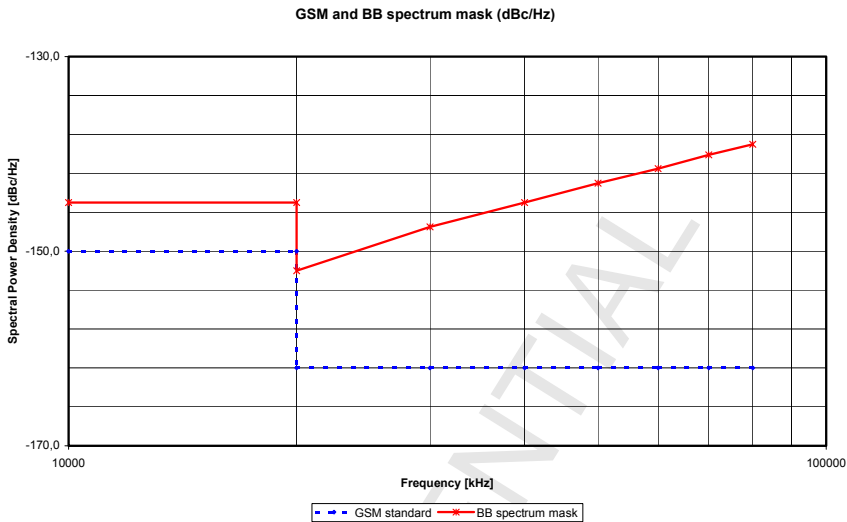


Figure 8-40 Spurious Emissions Mask (dBc/Hz)



### 8.8.8.1 Differential Offset Compensation

To reduce the differential offset voltage at the baseband output an offset measurement support is implemented in E-GOLDradio. To compensate the differential offset a combined software and firmware approach must be implemented. The measurement for compensation can be enabled when the modulation HW and the measurement interface is not used, e.g. after power on of the controller.

To determine the differential offset the measurement interface allows to connect the signals of the modulator TX path to the respective inputs of the measurement ADC. For accuracy enhancement of the TX offset measurement a compensation of the offsets within the measurement interface itself can be carried out (refer to [Section 10.5.7 Modulator Unit Offset Measurement TXOFI and TXOFQ \(on Page 729\)](#)). For each path of I and Q two values are supplied, one inverted and one not inverted:

- TXOFI (INV = 0) and TXOFI inverted (INV = 1) for the I path.
- TXOFQ (INV = 0) and TXOFQ inverted (INV = 1) for the Q path.

**CONFIDENTIAL**

**GMSK Modulator**

The controller must use all four values to calculate the offset of the I- and Q-path. For each value the measurement interface must be triggered with the GSM-timer signal ADCTRIG. The following steps are required to compensate the offset of the transmit I- and Q-path:

1. GSM timer: Enable CODON and TXON signals, refer to [Section 10.11.3 GSM Timer Decoder \(on Page 833\)](#).
2. TEAKLite: Set **MOD\_ACI** = 0 and **MOD\_ACQ** = 0.
3. TEAKLite: Program digits\_out\_I and digits\_out\_Q to the corresponding offset compensation registers **MOD\_OCI** and **MOD\_OCQ**.

## 8.9 Baseband Receive Unit

History	
Design Spec.	Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03,
Changes for Rev. 1.04	
<b>Page 516</b>	Updated <b>Section 8.9.8 The SD Analog-to-Digital Converter for Baseband Signals</b> WS00007299
Changes for Rev. 1.06	

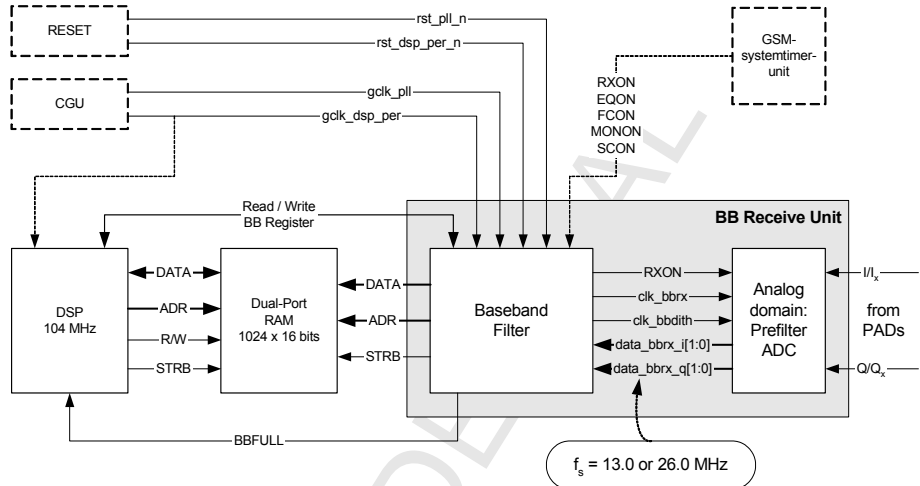
### System Integration

- Supply domains:
  - VDD\_DSP for digital parts of the baseband receive unit.
  - VDD\_BB for analog parts of the baseband receive unit.
- Clock domain:
  - gclk\_pll for analog part
  - gclk\_dsp\_per for digital part
- Bus domain / bus interface: TEAKLite data bus interfaces.
- Number of interrupt sources:
  - 1 TEAKLite interrupt: BB filter to DSP (bbfull).

## 8.9.1 General Description

### 8.9.1.1 Functional Overview

**Figure 8-41 Interface of BB Receive Unit**



The ADC converters deliver the I and Q samples to the baseband filter block. The samples are processed by a multi-stage low-pass filter for channel filtering. The complete receive chain consists of the  $\Sigma\Delta$  ADC converters, the low pass filters, a DC compensation stage and the CORDIC processor.

The last filter stage is an adaptive switchable linear-phase FIR filter. Depending on the level of adjacent channel interference it selects a filter with appropriate frequency transfer characteristic. The decision is made burst by burst. An optimized set of coefficients is provided on firmware mask, however filter coefficients can also be provided by an external memory via firmware command. The complete filter chain does not only perform lowpass filtering for the decimation but also the channel filtering.

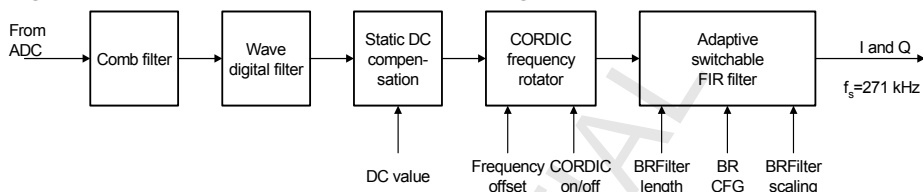
The ADC offers two operating modes, a standard mode and an enhanced mode. If the RF transceiver chip provides channelization filtering the standard mode is suitable, while the enhanced mode is for "lite" RF transceiver chips which provide anti-aliasing filtering only. The ADC mode is controlled by the bit **BB\_CTRL.BB\_ADCMODE**. In standard mode a 3rd-order  $\Sigma\Delta$ -modulator performs the A/D conversion at a sampling rate of 13 MHz with a very low power consumption. The digital filters perform a decimation by a ratio of 48 to lower the sampling rate to GSM symbol rate. In enhanced mode a 4th-order  $\Sigma\Delta$ -modulator that performs the A/D conversion at a sampling rate of 26 MHz offers the required 14-bit resolution. A decimation by a ratio of 96 is performed by the digital filters.

CONFIDENTIAL

Baseband Receive Unit

**Figure 8-42** shows a simplified functional block diagram of the digital baseband filter. The blocks of the digital baseband filter are described in the following chapters. For information on the analog domain, please refer to chapters **Section 8.9.7 Analog Pre-Filters (on Page 515)** and **Section 8.9.8 The SD Analog-to-Digital Converter for Baseband Signals (on Page 515)**.

**Figure 8-42 Simplified Functional Block Diagram of Baseband Filter**



The complete path is enabled via RXON signal.

*Note: The signals I/IX and Q/QX of the receive path are multiplexed together with the according signals of the transmit path to the same balls, I/IX and Q/QX. The selection of the RX- or TX-path must be done by the GSM timer signals RXON and TXON which must not be activated at the same time.*

*Note: The RXON signal must be inactive if there is no active clock, in other case it would cause a very high power consumption.*

## 8.9.2 Register Overview

**Table 8-35 Modulator Register List**

Register Group	Register Name	Register Symbol
Control and Status Registers	Control Register	<b>BB_CTRL</b> <sup>1)</sup>
	Interrupt Pointer Register	<b>BB_INT_POINTER</b> <sup>1)</sup>
	Write Pointer Register	<b>BB_WR_POINTER</b> <sup>2)</sup>
	IQ Imbalance Register	<b>BB_IQ_IMBALANCE</b> <sup>3)</sup>
	Status Register	<b>BB_STATUS</b> <sup>4)</sup>
	DC Offset Register for I-Component	<b>BB_DCOFFSET_I</b> <sup>3)</sup>
	DC Offset Register for Q-Component	<b>BB_DCOFFSET_Q</b> <sup>3)</sup>
	Frequency Shift Register	<b>BB_FSHIFT</b> <sup>3)</sup>
	Broad Filter Control Register	<b>BB_BRFILTER_CTRL</b> <sup>3)</sup>
	MSBs of the estimated power of wanted signal	<b>BB_Pbase_MSB</b> <sup>5)</sup>
	LSBs of the estimated power of wanted signal	<b>BB_Pbase_LSB</b> <sup>5)</sup>
	MSBs of the estimated power of adjacent channel interferer	<b>BB_Padj_MSB</b> <sup>5)</sup>
	LSBs of the estimated power of adjacent channel interferer	<b>BB_Padj_LSB</b> <sup>5)</sup>

- 1) clock domain: clk\_pll. Register can be read/written any time. Write access only by DSP
- 2) clock domain: clk\_pll. Write access by BB\_RX.
- 3) clock domain: clk\_dsp. Write access is only allowed during the inactive phase of BB\_RX (bb\_rxon = 0).
- 4) clock domain: clk\_pll.
- 5) clock domain: clk\_pll. DSP read access only allowed after the new power estimate values are written into the registers.

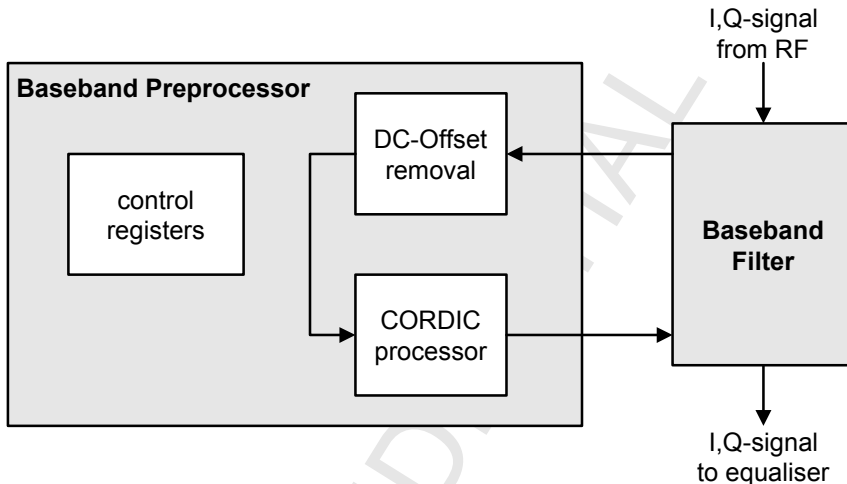
CONFIDENTIAL



### 8.9.3 CORDIC and DC-Offset

The CORDIC-based baseband preprocessor (see [Figure 8-43](#)) can be used during asynchronous operation mode of the MS. The baseband preprocessor corrects the frequency offset in the baseband signal prior to demodulation in the channel equalizer.

**Figure 8-43 CORDIC-Based Demodulator Block Diagram**



The baseband signals usually have a DC offset superposed to the baseband signal. The DC offset removal block first subtracts the DC value from the baseband signal. The DC offset value is estimated in monitoring mode. For a heterodyne receiver (IF-receiver) architecture the DC offset value typically has a very slow drift. If the DC offset value is unstable as for instance in a homodyne receiver (direct conversion receiver), the CORDIC algorithm is not advantageous.

The DC-offset-free baseband signal is further processed in the CORDIC processor. The CORDIC processor can shift the baseband signal in the frequency domain. The frequency shift value is estimated by the FCB search algorithm and the channel equalizer.

### 8.9.4 Adaptive Switchable FIR Filter

As in many communication systems also in a GSM system the reception of the wanted signal is distorted by some other signals like noise, interferences etc. The wanted signal typically is a narrow-band signal. In contrast with this noise is wideband. Interferences often have a spectrum which is similar to that of the wanted signal. But they differ from the wanted signal in other properties like time, frequency, code, direction, and power.

**CONFIDENTIAL****Baseband Receive Unit**

While a co-channel interference lies in the same frequency range as the wanted signal, an adjacent channel interference occupies the left or right neighboring frequency channel. In the GSM system the channel spacing is 200 kHz. In other words, the adjacent channels are centered around  $\pm 200 \cdot n$  kHz in the baseband while the wanted signal has a center frequency of 0 Hz. Co-channel interferences having the same center frequency as the wanted signal arise from neighboring cells.

Both noise and interferences degrade the received signal. One solution against these signal distortions is to employ a channelization filter. It is a low pass filter with the center frequency of 0 Hz. It is designed to let the wanted signal pass the filter without distortion while the noise and adjacent channel interference are suppressed as much as possible.

This is easy to accomplish if there is no spectrum overlapping between the wanted signal and the adjacent channel interference. For highly spectral efficient communication systems like GSM, however, there is an intended overlapping between the wanted signal and the adjacent channel interference in the frequency domain. In this case, the adjacent channel interference cannot be suppressed completely without cutting into the spectrum of the wanted signal. Therefore, the design of the low pass channelization filter becomes a matter of compromise:

**Narrow channelization filter**

If the adjacent channel interference is very strong (even stronger than the wanted signal), the low pass channelization filter should be as narrow as possible to sufficiently suppress the interference. A slight cut into the spectrum of the wanted signal is tolerated in this case.

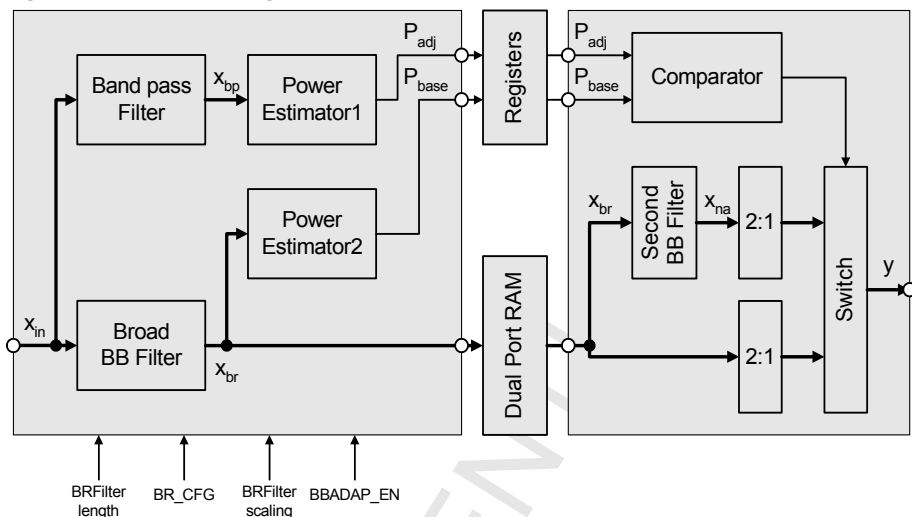
**Broad channelization filter**

If there is no adjacent channel interference and the noise level is quite low, the channelization filter should be broad and not cut into the spectrum of the wanted signal.

**Medium channelization filter**

If the noise is dominating, a channelization filter with a medium cut-off frequency is preferable. It reduces the noise bandwidth without severe distortion of the wanted signal. Clearly, an optimum channelization filter strongly depends on the level of the adjacent channel interference. In E-GOLDradio a two-stage adaptive filter for the channelization filtering is used. The term "adaptive" does not mean a continuous sample-by-sample adjustment of the filter coefficients, it means a burst-by-burst decision.

**Figure 8-44 Block Diagram of Adaptive FIR Filter**



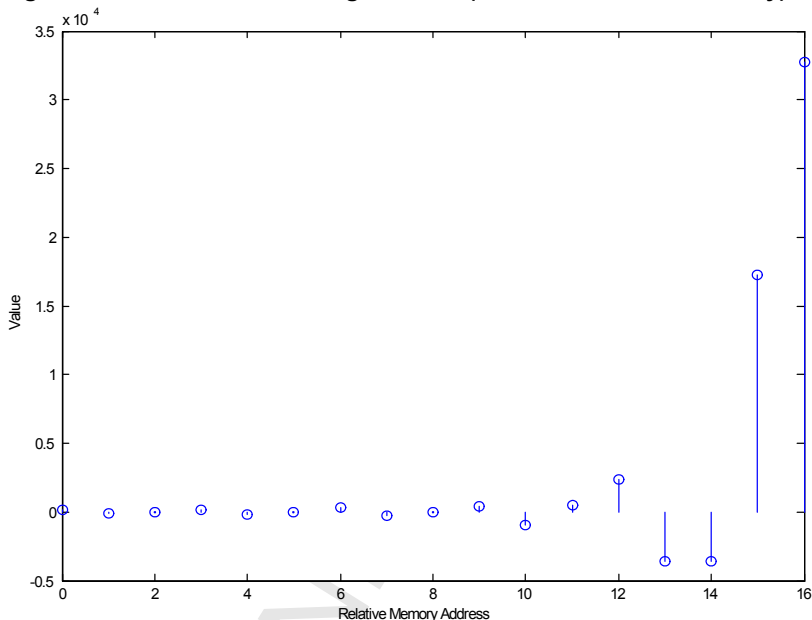
**Figure 8-44** shows a block diagram of the adaptive channelization filter. It consists of a broad filter, a second filter, a band pass filter and two power estimators. The broad filter has a slightly high cut-off frequency. It is intended to pass the wanted signal without any distortion when there is no adjacent channel interference. The second filter has the function of narrowing the amplitude response of the broad filter. In other words, the cascade of the broad filter and the second filter behaves like a narrow filter having a smaller cut-off frequency. It should reject strong adjacent channel interferences. However, it also cuts into the spectrum of the wanted signal causing an intentional signal distortion. This signal distortion is much smaller than what the adjacent channel interference would cause if it is not rejected by a narrow low pass filter. The band pass filter extracts that portion of signal power which is important for the control device to make a decision. Both broad filter output and band pass filter output are fed to a power estimator. Because the result of the power estimation for wanted signal can be highly affected by DC offset, the first stage of the power estimator 2 in **Figure 8-44** contains a notch filter which removes the DC offset from the broad BB filter output signal. Then the estimated adjacent channel signal power  $P_{adj}$  is multiplied with a user defined threshold  $t$  and the product  $P_{adj} * t$  is compared with the second signal power estimation  $P_{base}$ . The result of this comparison is used to control the switch which selects the appropriate filter output.

*Note: The threshold  $t$  also depends on the gain of the implemented broad filter and band pass filter. Any changes of the filter gain (for example change of BRFilterScaling value) must also be considered in the threshold determination.*

The adaptive filter can be disabled by a firmware command. In this case the second filter can always be set on or off by TEAKLite command, and the band pass filter and the power estimators can be disabled for power saving.

### 8.9.4.1 Programming of FIR Filter

**Figure 8-45 Coefficient Storage Format (Coefficient are Preliminary)**



The broad filter can be configured either as a FIR filter without decimation or as a two-fold decimation filter. The maximum filter order is limited to 32 or 64, respectively. If the broad filter is configured as decimation filter the adaptive filter must be switched off and the second filter must be disabled. In this case the last filter stage realizes a compromise channelization filter.

The linear phase broad filter is fully programmable. The filter coefficients have to be stored in a dual port memory which is read over the CoeffIn port while the IQwrite flag is high. Only half of the symmetrical real coefficient set with an odd number of elements is stored in the memory starting at the address 0. The coefficient in the middle is stored at the highest address which corresponds to the filter order divided by two. **Figure 8-45** shows an example with a filter order of 32. To achieve an optimum SNR the coefficients have to be scaled in a way that the biggest coefficient uses the full dynamic range of the available 16 bits. To save power the filter order has to be programmed via the 8 bit

BRFilterLength port. Please note that the filter order divided by two has to be applied to the BRFilterLength port.

The output scaling of the broad filter can be programmed via the BRFilterScaling port in order to get the best dynamic range. The maximum output value of the FIR filter is  $\sum |Coeff(n)|$  times the maximum input value in case of L1 scaling. If you want to use the L1 scaling you have to program the BRFilterScaling port in the following way:

$$BRFilterScaling = \left\lceil 2 + \text{ld}\left(\sum_{n=1}^{2BRFilterLength+1} |Coeff(n)|\right) \right\rceil \quad [0.9]$$

*Note: For the following calculations the fractional representation of the coefficients Coeff(n) has to be used*

The minimum scaling of two is necessary because of internal rear bits used in the FIR filter implementation. The DC gain caused by the broad filter can be calculated in the following way:

$$Gain_{BR} = 2^{2-BRFilterScaling} \left( \sum_{n=1}^{2BRFilterLength+1} Coeff(n) \right) \quad [0.10]$$

*Note: Any different gain of the FIR has to be considered in the overall gain calculations.*

The second filter has a maximum order of 12 and is initialized with a predefined filter coefficient set. The DSP can program both filter order and filter coefficients. The filter coefficients must be quantized such that  $\sum Coeff(n) = 65536$ . The DC gain caused by the second filter is then 1.

## **8.9.5 Interface to Dual-Port RAM**

The filtered 16-bit samples have a resolution of approximately 12 bit or 14 bit, depending on the selected ADC mode. These samples are stored in the baseband filter RAM (BB\_RAM) in an alternating order. I-channel samples are written to even addresses, while Q samples are store at odd addresses. The signal IQFlag marks I (IQFlag = High) and Q (IQFlag = Low) Samples. The RAM itself is a 1024 word dual-port memory.

Each new job of the BB-Filter starts at a rising edge of one of the signals EQON, FCON, MONON or SCON (rising edge of SAMPON). The address of the write counter is automatically incremented.

The TEAKLite has read access to the write pointer (reads the current write address of the IQ input samples). After the falling edge of EQON, FCON, MONON, or SCON, the **BB\_WR\_POINTER** keeps the value of the last value stored. The **BB\_WR\_POINTER** is reset at the beginning of the next job at the rising edge of the signals listed above. All other registers are not influenced by the signals EQON, FCON, MONON or SCON.

The TEAKLite can program an interrupt value using the **BB\_INT\_POINTER**. The usual length of the job is 640 samples (320 I/Q pairs) if the broad filter is not configured as decimation filter, and 320 samples if the broad filter is configured as decimation filter. When the **BB\_WR\_POINTER** reaches **BB\_INT\_POINTER** an interrupt BBFULL in **INT\_FINTAO** is generated. Now the TEAKLite can pick up the samples from the RAM

starting from Address  $0_D$ . An automatic wrap-around occurs when reaching the last address available for BB-Filter ( $959_D$ ).

The address area between  $960_D$  and  $1023_D$  is used for BB filter coefficient set. This set is programmed by the TEAKLite. To avoid data mismatch, the coefficient set should be written while the BB filter is not active.

After RESET =1 all registers and the BB RAM are reset.

The TEAKLite sets the register **BB\_INT\_POINTER**. The internal clock of the BB filter and analog block is switched on by signal RXON from the GSM system interface.

CONFIDENTIAL

**CONFIDENTIAL**

**Baseband Receive Unit**

## 8.9.6 Register Description

### 8.9.6.1 Baseband Filter Control Register

**BB\_CTRL**

**Baseband Filter Control Register**

**Reset value: 0110<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED							RESERVED	RESERVED	RESERVED	BBADAP_EN	BBADC_MODE	BB_ON	CORDICON	BB_STOP	

Field	Bits	Type	Description
<b>BB_STOP</b>	0	rw	<b>Override Function of Hardware Control of BB-Filter</b> 0 BB filter controlled by RXON signal from GSM Timer. 1 BB filter stopped by TEAKLite (RXON has no influence). <b>BB_STOP</b> is reset by RXON = 0.
<b>CORDICON</b>	1	rw	<b>CORDIC Block Control</b> 0 Disabled 1 Enabled <i>Note: Gain is also switched.</i>
<b>BB_ON</b>	2	rw	<b>Baseband Clock Switch</b> 0 Switched off 1 Switched on <i>Note: Signal is or combined with RXON from system interface.</i>
<b>BB_ADCMODE</b>	3	rw	<b>ADC Mode Switch</b> 0 Standard mode 1 Enhanced mode If ADC is set to enhanced mode, the sampling frequency is doubled
<b>BBADAP_EN</b>	4	rw	<b>Filter Mode Switch</b> 0 Non-adaptive filter mode 1 Adaptive filter mode

Field	Bits	Type	Description
<b>RESERVED</b>	5	rw	Reserved; these bits must be left at their reset values. <i>Note: In the standard ADC mode <b>BB_LOWIF</b> must be set to zero.</i>
<b>RESERVED</b>	6	rw	Reserved; these bits must be left at their reset values.
<b>RESERVED</b>	8:7	rw	Reserved; these bits must be left at their reset values.
<b>RESERVED</b>	15:9	r	Reserved; these bits must be left at their reset values.

### 8.9.6.2 Baseband Filter Interrupt Pointer Register

#### BB\_INT\_POINTER

#### Baseband Filter Interrupt Pointer Register

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>						<b>INT_POINTER</b>									

Field	Bits	Type	Description
<b>INT_POINTER</b>	9:0	rw	<b>Address of the Interrupt Pointer</b> When the <b>BB_WR_POINTER.WR_POINTER</b> reaches <b>INT_POINTER</b> , an interrupt BBFULL in register <b>INT_FINTAO</b> is generated.
<b>RESERVED</b>	15:10	r	Reserved; these bits must be left at their reset values.

### 8.9.6.3 Baseband Filter Write Pointer Register

#### BB\_WR\_POINTER

#### Baseband Filter Write Pointer Register

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>						<b>WR_POINTER</b>									



Field	Bits	Type	Description
<b>WR_POINTER</b>	9:0	r	<b>Address of the Write Pointer</b> The <b>WR_POINTER</b> is reset at the beginning of the next job at the rising edge of EQON, MONON, SCON or FCON.
<b>RESERVED</b>	15:10	r	Reserved; these bits must be left at their reset values.

CONFIDENTIAL

**CONFIDENTIAL**

**Baseband Receive Unit**

#### 8.9.6.4 Baseband Filter IQ Imbalance Register

##### BB\_IQ\_IMBALANCE

##### Baseband Filter IQ Imbalance Register

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				RESERVED				RESERVED				RESERVED			

Field	Bits	Type	Description
RESERVED	4:0	rw	Reserved; these bits must be left at their reset values.
RESERVED	12:8	rw	Reserved; these bits must be left at their reset values.
RESERVED	7:5, 15:13	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**Baseband Receive Unit**

### 8.9.6.5 Baseband Filter Status Register

**BB\_STATUS**

**Baseband Filter Status Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											<b>RX ON</b>	<b>SC ON</b>	<b>MON ON</b>	<b>FC ON</b>	<b>EQ ON</b>

Field	Bits	Type	Description
<b>EQON</b>	0	r	The register carries the current status of the BB-Filter relevant signals from the GSM system interface.
<b>FCON</b>	1	r	
<b>MONON</b>	2	r	
<b>SCON</b>	3	r	
<b>RXON</b>	4	r	
<b>RESERVED</b>	15:5	r	Reserved; these bits must be left at their reset values.

### 8.9.6.6 Baseband Filter DC Offset Register for I-Component

**BB\_DCOFFSET\_I**

**Baseband Filter DC Offset Register for I-Component**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Field	Bits	Type	Description
<b>DCOFFSETI</b>	15:0	rw	1 LSB corresponds to either: <ul style="list-style-type: none"> <li>• 0.72 LSBs in the filter RAM (adaptive channelization filter)</li> <li>• 1.03 LSBs in the filter RAM (compromise channelization filter).</li> </ul> <i>Note: Only applies when CORDIC_ON = 1.</i>

**CONFIDENTIAL**

**Baseband Receive Unit**

### 8.9.6.7 Baseband Filter DC Offset Register for Q-Component

**BB\_DCOFFSET\_Q**

**Baseband Filter DC Offset Register for Q-Component**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>DCOFFSETQ</b>															

Field	Bits	Type	Description
<b>DCOFFSETQ</b>	15:0	rw	1 LSB corresponds to either: <ul style="list-style-type: none"> <li>• 0.72 LSBs in the filter RAM (adaptive channelization filter)</li> <li>• 1.03 LSBs in the filter RAM (compromise channelization filter).</li> </ul> <i>Note: Only applies when CORDIC_ON = 1.</i>

### 8.9.6.8 Baseband Filter Frequency Shift Register

**BB\_FSHIFT**

**Baseband Filter Frequency Shift Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>FSHIFT</b>															

Field	Bits	Type	Description
<b>FSHIFT</b>	15:0	rw	$fshift = \Delta f[Hz] * 2^{16*24/(13E6[Hz])}$ .

**CONFIDENTIAL**

**Baseband Receive Unit**

### 8.9.6.9 Broad Filter Control Register

**BB\_BRFILTER\_CTRL**

**Broad Filter Control Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			BR CFG	BRFilterScaling				BRFilterLength							

Field	Bits	Type	Description
BRFilterLength	7:0	rw	Filter order divided by two.
BRFilterScaling	11:8	rw	$\text{Gain}_{\text{Scaling}} = 2^{2 - \text{BRFilterScaling}}$
BRCFG	12	rw	0 FIR filter configured as FIR without decimation. 1 FIR filter configured as decimation filter.
RESERVED	15:13	r	Reserved; these bits must be left at their reset values.

### 8.9.6.10 MSB of Estimated Power of Wanted Signal

**BB\_Pbase\_MSB**

**MSBs of Estimated Power of wanted Signal**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								Pbase[23:16]							

Field	Bits	Type	Description
Pbase[23:16]	7:0	r	MSBs of the estimated power of wanted signal. The estimation is made burst-by-burst.
RESERVED	15:8	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**Baseband Receive Unit**

### 8.9.6.11 LSBs of Estimated Power of Wanted Signal

**BB\_Phase\_LSB**

**LSBs of Estimated Power of wanted Signal**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>Pbase[15:0]</b>															

Field	Bits	Type	Description
<b>Pbase[15:0]</b>	15:0	r	LSBs of the estimated power of wanted signal. The estimation is made burst-by-burst.

### 8.9.6.12 MSBs of Estimated Power of Adjacent Channel Interference

**BB\_Padj\_MSB**

**MSBs of the estimated power of adjacent channel interference**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>								<b>Padj[23:16]</b>							

Field	Bits	Type	Description
<b>Padj[23:16]</b>	7:0	r	MSBs of the estimated power of adjacent channel interference. The estimation is made burst-by-burst.
<b>RESERVED</b>	15:8	r	Reserved; these bits must be left at their reset values.

### 8.9.6.13 LSBs of Estimated Power of Adjacent Channel Interference

**BB\_Padj\_LSB**

**LSBs of Estimated Power of Adjacent Channel Interference**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>Padj[15:0]</b>															

Field	Bits	Type	Description
<b>Padj[15:0]</b>	15:0	r	LSBs of the estimated power of adjacent channel interference. The estimation is made burst-by-burst.

### 8.9.7 Analog Pre-Filters

Both signals, for example, the in-phase and quadrature components of the baseband signal from the RF demodulator circuit, have to be delivered to E-GOLDradio as differential analog signals (IR/IRX and QR/QRX). Each of them will pass a 1st order RC low pass filter which acts as an *anti-aliasing pre-filter* with a typical 3 dB attenuation @400 kHz (Standard mode) or @ 6 MHz (Enhanced mode).

### 8.9.8 The $\Sigma\Delta$ Analog-to-Digital Converter for Baseband Signals

The 2-channel analog-to-digital converter path is part of the mixed signal block. The complete path can be put into power down mode by the signal RXON.

The analog-to-digital conversion of the pre-filtered baseband signal is performed using for each component I and Q a suitable combination of a *Sigma-Delta* ( $\Sigma\Delta$ ) modulator of 3rd order (standard ADC mode) or 4th order (enhanced ADC mode) and a subsequent digital decimating low pass filter

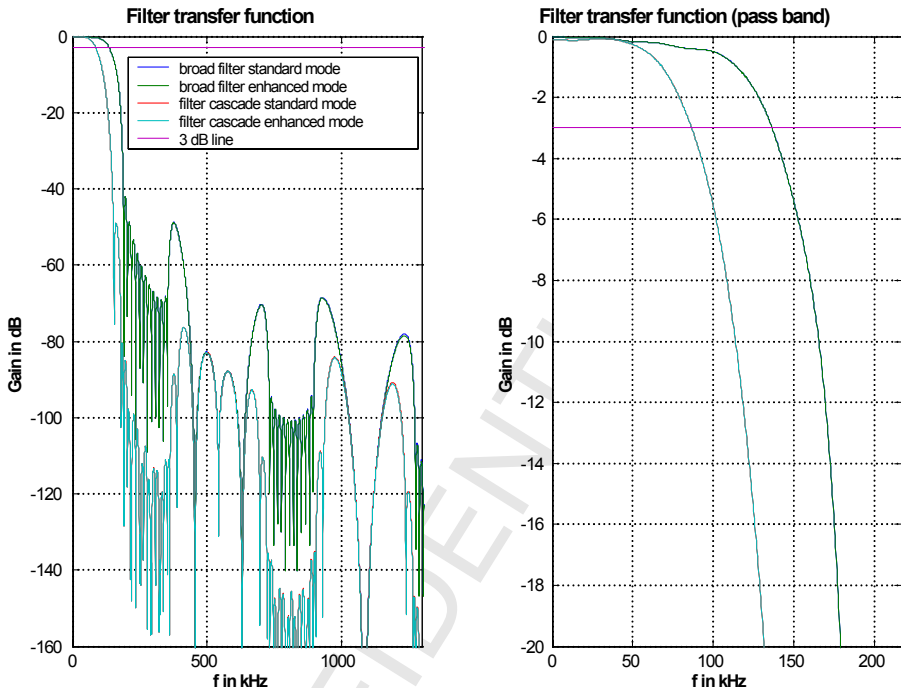
The modulator shapes the noise spectrum such that the main noise energy is shifted to higher frequencies (maximum at half of the sampling frequency).

The tasks of the digital low pass filters are:

- Suppression of the high frequency quantization noise introduced by the  $\Sigma\Delta$  modulator,
- Reconstruction of the original I or Q component of the baseband signal,
- Decimation of the sampling rate,
- Suppression of adjacent channel interferences to improve the overall selectivity.

The baseband filter is a digital multi-rate decimating low pass filter. The -3 dB corner frequency of this filter depends on the FIR filter chosen. **Figure 8-46** shows the amplitude function of the baseband filter in different modes.

**Figure 8-46 Amplitude Functions of the Digital Baseband Filter**



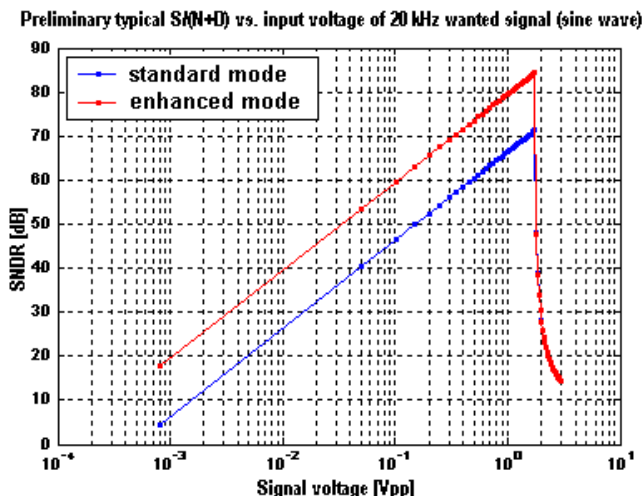
After setting RXON, the IQ data path needs a transient response time of 75  $\mu$ s. Transients in the power estimation path are significantly longer than 75  $\mu$ s and must be compensated by the DSP firmware. For a successful compensation, a longer delay between RXON and EQON/SCON may be necessary. The minimum delay depends on the RF transceiver used.

**Notes:**

- Group delay and settling time assumes BB broad filter order of 32
- The complete A-to-D converter consists of the  $\Sigma\Delta$ -modulator plus the baseband filter
- The baseband filter is reset after every falling edge of RXON.



Figure 8-47 Signal to Noise + Distortion Ratio for BB-RX Path

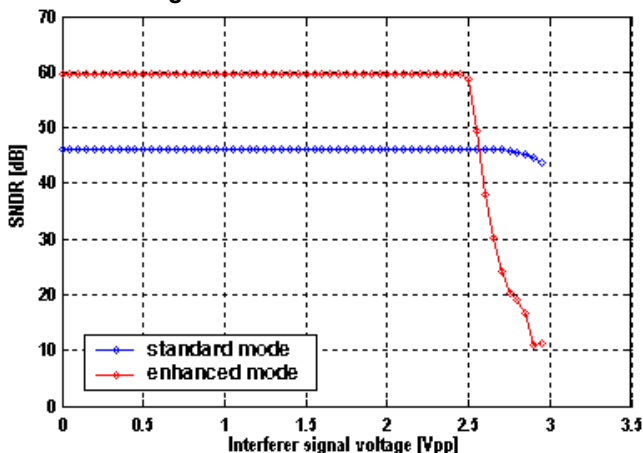


An important performance measurement of an A-to-D converter is the  $S/(N+D)$  which can be achieved. In [Figure 8-47](#) the  $S/(N+D)$  is depicted versus of the signal amplitude. A 20 kHz sine-wave signal was used as input. This measurement evaluates the complete A-to-D converter path consisting of the  $\Sigma\Delta$ -converter and the decimating filters.

A more powerful measurement for GSM systems is the achievable  $S/(N+D)$  in the presence of strong interference from the adjacent channels. Measurements with a small wanted signal and a strong interfering signal show what amplitude of the interference will cause the A-to-D converter to go into saturation.

In [Figure 8-48](#) the wanted signal has an amplitude of 100 mVpp @ 20 kHz. Depicted on the x-axis is the signal amplitude of the interfering signal in Vpp. The interfering signal is transmitted at a frequency of 400 kHz.

**Figure 8-48 Signal to Noise + Distortion Ratio for BB-RX with 100 mVpp Wanted Signal**



*Note: The depicted signal to noise ratios in [Figure 8-47](#) and [Figure 8-48](#) are simulation results based on typical  $\Sigma\Delta$ -modulator performance (including the baseband filter). This refers to the typical value of  $S/(N+D)$ ; the worst case value is lower.*

## 8.9.9 Filter Coefficients

### 8.9.9.1 Adaptive Channelization Filter

The following broad filter coefficients used in this section:

116,-64,-42,156,-176,-45,307,-256,0,372,-944,457,2327,-3570,-3581,17297,32767,  
17297,-3581, -3570,2327,457,-944,372,0,-256,307,-45,-176,156,-42,-64,116

BRFilterLength = 16 (Number of Filter Coefficients = 2\*BRFilterLength+1)

BRFilterScaling = 3

BRCFG = 0

*Note: Prior to the gain calculation and scaling, all filter coefficients have to be divided by 32768.*

Gain\_BR =  $\text{sum}(\text{Coeff}) / 32768 / 2^{\text{BRFilterScaling} - 2} =$   
 $\text{sum}(\text{Coeff}) / 2^{\text{BRFilterScaling} + 13} = 0.877$

[0.11]

The following second filter coefficients are used in this section:

570,-315,-2750,-2237,6080,18839,25162,18839,6080,-2237,-2750,-315,570

FilterLength = 6 (Number of Filter Coefficients = 2\*FilterLength+1)

*Note: Prior to the gain calculation and scaling, all filter coefficients have to be divided by 65536.*

$$\text{Gain\_CAS} = \text{sum}(\text{Coeff})/65536 = 1 \quad [0.12]$$

### 8.9.9.2 Compromise Channelization Filter

The following broad filter coefficients are used in this section:

22,57,-17,-63,18,80,-11,-104,-12,113,34,-117,-38,148,58,-208,-140,232,243,-206,  
-284,275,367,-582,-936,815,2481,147,-4910,-4533,7260,24477,32767,24477,7260,  
-4533,-4910,147,2481,815,-936,-582,367,275, -284,-206,243,232,-140,-208,58,148,  
-38,-117,34,113,-12,-104,-11,80,18,-63,-17,57,22

BRFilterLength = 32 (Number of Filter Coefficients = 2\*BRFilterLength+1)

BRFilterScaling = 3

BRCFG = 1

*Note: For gain calculation and scaling all filter coefficients have to be divided by 32767 prior*

$$\text{Gain\_BR} = \text{sum}(\text{Coeff}) / 32768 / 2^{\text{BRFilterScaling} - 2} = \text{sum}(\text{Coeff}) / 2^{\text{BRFilterScaling} + 13} = 1.2527 \quad [0.13]$$

CONFIDENTIAL

**CONFIDENTIAL**

**Audio Front-End**

## 8.10 Audio Front-End

History	
Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.04	
<b>Page 521</b>	Heading "Voice" changed to <b>Audio Front-End</b> WS00007944
<b>Page 532</b>	Updated <b>Figure 8-51 Output Selection: Block Diagram and Potential External Connections</b> WS00008013
Changes for Rev. 1.05	
<b>Page 527</b> <b>Page 541</b>	Updated bit types in registers <b>AFE_RWADDR</b> & <b>AFE_VTXCTRL</b> WS00008923
<b>Page 540</b>	Added "Reserved" bits to register <b>AFE_VRXCTRL2</b> WS00008923
Changes for Rev. 1.06	

### System Integration:

- Supply domain:
  - Digital part of the audio interface: VDD\_DSP
  - Analog part of the audio front end: VDDvbt, VDDvbr
- Chip internal interfaces:
  - Clock domain:
    - clk\_pll drives the internal module clock
    - clk\_teak drives the interface clock to TEAK bus
  - Bus domain: TEAK Data Bus
  - Interrupt sources: Rx/Tx IRQ
- Chip external signals:
  - AFE\_OUT0: Line-out driver EPp1
  - AFE\_OUT1: Line-out driver EPn1
  - AFE\_OUT2: Output buffer EPpa1
  - AFE\_OUT3: Output buffer EPpa2
  - MICP1: First differential microphone input
  - MICN1: First differential microphone input
  - MICP2: Second differential microphone input
  - MICN2: Second differential microphone input
  - VMICP: Microphone power supply voltage

**CONFIDENTIAL**

**Audio Front-End**

- VMICN: Microphone power supply voltage
- Monitor pins: Refer to [Section 11.8.10 Internal Signal Monitoring \(on Page 1209\)](#)

CONFIDENTIAL

**Table 8-36 External Voice Signals**

<b>Signal</b>	<b>E-GOLDradio Ball Name</b>
AFE_OUT0	EPp1
AFE_OUT1	EPn1
AFE_OUT2	EPpa1
AFE_OUT3	EPpa2

### 8.10.1 Functional Overview

The audio front-end of PMB7870 offers the digital and analog circuit blocks for both receive and transmit audio operation and ringing. It features a high-quality, digital-to-analog path with amplifying stages for connecting acoustic transducers to the PMB7870. In the transmit direction the supply voltage generation for microphones, low-noise amplifier and analog to digital conversion are integrated on the PMB7870. A more detailed functional description is given in the following sections.

The audio front-end itself can be considered to be organized in three sub-blocks:

- Interface to processor cores (TEAKlite and - indirectly - C166S)
- Digital filters
- Analog part.

The interface to the processor cores consists of a direct physical connection to the TEAKlite DSP bus and a set of firmware commands to handle communication between the C166S and the audio front-end which serves as the interface peripheral for audio algorithms running on the DSP or the controller. The audio front-end generates interrupts on certain occasions, for example, when exchange of data is requested. The core interface part of the audio front-end also contains the control and status registers which are used to set up certain operation modes of the peripheral.

The section next to the core interface contains the digital filters for interpolation and decimation of the audio signals being received and transmitted. The data path for the receive direction can be set up to process sampling rates between 8 kHz and 48 kHz. The interpolation filters for the respective sampling rates are implemented in a dedicated hardware block and are automatically selected to suite the chosen sampling rate. Low-pass interpolation filtering, which produces an unsigned 16-bit data stream with a sampling rate of 4 MHz, is performed digitally. D-to-A conversion, postfiltering, and final amplification are performed on the analog part. The amplifier buffer for voiceband receive does also support ringer functionality. The ringer functionality is activated by setting bits RINGSELPN or RINGSELPa in the voiceband part of the analog control register.

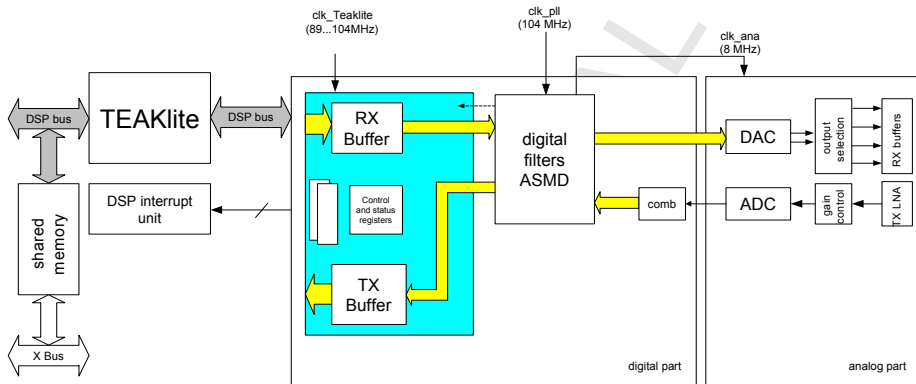
**CONFIDENTIAL**

**Audio Front-End**

In transmit direction, amplification, prefiltering and A-to-D conversion (analog  $\Sigma\Delta$  modulation) are performed on the analog part. The resulting 2-Mbit/s data stream is filtered by a digital low-pass decimation filter for further processing by DSP firmware. Two sampling rates, 8 kHz and 16 kHz, are supported.

The analog section contains all the necessary analog functional blocks including microphone supply generation, output and input amplifiers and analog filtering.

**Figure 8-49 Overview of Clocking and Interfaces of Audio Front End**



The audio front-end of PMB7870 has the following major operation modes:

- Power-down mode: All analog parts are in power down and all clocks of the digital part are switched off.
- Audio mode: Digital decimation and interpolation filters are connected to the interface buffers and the analog part is enabled.

These major modes can be modified by control register settings.

**Note:**

- Due to the gain settings in the TX path, the maximum input voltage is limited to 1.03 V<sub>pp</sub>.
- In the TX path, the range for voice samples is confined to 60%, that is, to [-19661, 19660] or [B333<sub>H</sub>, 4CCC<sub>H</sub>] in the PMB7870.
- In the RX path, the range for voice samples is confined to 97.5%, that is, to [-31948, 31947] or [8334<sub>H</sub>, 7CCB<sub>H</sub>] in the PMB7870.
- The highpass functions of the voiceband filters have to be implemented in firmware on DSP2.



**CONFIDENTIAL**

**Audio Front-End**

## **8.10.2 Digital Part**

The digital part of the PMB7870 audio front-end comprises an interface to the TEAKlite DSP bus, interfaces to the interrupt units of TEAKlite, digital interpolation filters for oversampling, digital-to-analog conversion, digital decimation filters for analog-to-digital conversion and an interface to the analog part of the audio front-end.

## **8.10.3 Interface to DSP Bus and C166**

The audio front-end is accessible from both the C166S MCU and the TEAKlite DSP. Access from the DSP is performed directly over the DSP bus. Access from C166S is performed via the shared memory.

The interface between the DSP and the audio front-end interface is two 64-word, dual port ring buffers (one for RX and one for TX direction).

The DSP may access each 16-bit word of these ring buffers independently, this means all 64 words of each of the buffers are mapped to the data address space of the DSP (the transmit ring buffer is mapped to addresses  $XXXX_H$  up to  $XXXX_H+3F_H$  and the receive ring buffer is mapped to addresses  $XXXX_H+40_H$  up to  $XXXX_H+40_H+3F_H$ ). This way the DSP has full control of which memory location it writes to and from which it reads from. It is also possible to operate the interface buffers according to the needs of the software because various logical data types (for example, a FIFO queue) may be implemented on these buffers.

To play back audio data, the DSP has to write data samples to consecutive addresses in the ring buffer. These audio samples are read by the audio front-end at a later time. For flow control an interrupt mechanism causes an interrupt on DSP side as soon as the interface reads from a specific memory location. Using this feature the user may adjust the time between the occurrence of the interrupt signal and the buffer being empty.

The voice data can only be written in the RX buffers and can only be read from the TX buffer.

*Note: The downlink data from the antenna goes via the AFE RX buffer to the speaker.  
The uplink data from the microphone goes via the AFE TX buffer to the antenna.*

For this purpose a number of access routines is provided in the E-GOLDradio firmware (refer to the *E-GOLDradio Firmware Manual*).

The addresses where the interrupt is generated in the DSP are stored in the register **AFE\_INTPTR**. This register can be written to by the DSP.

CONFIDENTIAL

Audio Front-End

## AFE\_INTPTR

Interrupt Address Register for Interface Buffers

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								RESERVED							

Field	Bits	Type	Description
RXINTPTR	5:0	rw	Interrupt pointer for receive interface ring buffer
TXINTPTR	13:8	rw	Interrupt pointer for transmit interface ring buffer
RESERVED	7:6, 15:14	r	Reserved; these bits must be left at their reset values.

The read address, where the audio front-end currently reads from, is stored in [AFE\\_RWADDR.RDADDR](#). This register is read-only for the DSP. It is auto incremented with each read access by the audio front-end to the ring buffer and is wrapped around from value  $XXXX_H+40_H$  up to  $XXXX_H+40_H+3F_H$ .

Similarly, the address of the transmit buffer, where the audio front-end is currently writing to is stored in [AFE\\_RWADDR.WRADDR](#).

The read position of the transmit buffer and the write position of the receive buffer are controlled by the DSP and hence do not need to be stored in a separate register in the audio front-end. The register is reset with each restart of the audio front-end (refer to [AFE\\_BCON \(on Page 537\)](#)).

## AFE\_RWADDR

Read/Write Address Register for Interface Buffers

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		WRADDR						RESERVED		RDADDR					

Field	Bits	Type	Description
RDADDR	5:0	rh	Address of receive interface ring buffer read position
WRADDR	13:8	rh	Address of transmit interface ring buffer write position
RESERVED	7:6, 15:14	r	Reserved; these bits must be left at their reset values.

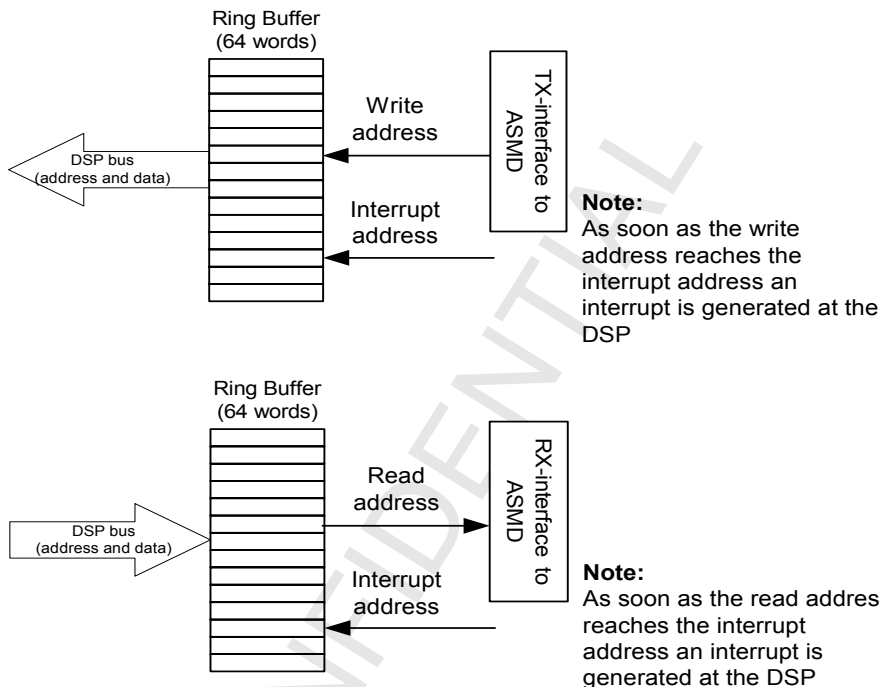
As soon as the read address **RDADDR** equals the interrupt address (stored in register **AFE\_INTPTR.RXINTPTR**) an interrupt is generated in the DSP and the bit **INT\_FINTB0.VBTX** is set. When the DSP enters the interrupt-service-routine for this interrupt it has to reset the interrupt source and to set a new value for the next interrupt position.

Similarly, when the write address **WRADDR** equals the receive interrupt address (stored in register **AFE\_INTPTR.TXINTPTR**) an interrupt is generated in the DSP and **INT\_FINTB0.VBRX** is set.

Figure 8-50 is a simplified block diagram of the interface.

**Figure 8-50 Functional Block Diagram of the Audio Front End DSP Interface**

(RX and TX Buffers)



When normal operation of either path (RX or TX) of the audio front-end is to be started the following steps have to be followed:

- The respective **AFE\_BCON (on Page 537).xxSTART** bits are set to zero. This resets the respective **AFE\_RWADDR.xxADDR** pointer.
- All control settings, in particular the sampling rate setting, the initial setup of the buffer (including the **AFE\_INTPTR.xxINTPTR** setting) are performed.
- The respective **AFE\_BCON.xxSTART** bit is set to one.

*Note: A change of sampling rate can only be performed when the respective **AFE\_BCON.xxSTART** bit is set to zero. Changes of gain settings or other analogue parameters can, of course, be made during the operation of the audio front-end.*

**CONFIDENTIAL**

**Audio Front-End**

### 8.10.3.1 Interpolation Filter

The interpolation path of the PMB7870 audio front-end increases the sampling rate of the audio samples to the rate of the digital-to-analog converter. Because the input sampling rates can vary between 8 kHz and 48 kHz, the filter characteristic and oversampling ratio can be adjusted to the respective sampling rate. The requirements for the interpolation filters are dependent on the sampling rate, because a sufficient out-of-band discrimination in the audio frequency band (20 Hz,...,20 kHz) has to be ensured. The filter requirements are summarized in [Table 8-37](#).

**Table 8-37 Requirements for Digital Interpolation Filters**

Parameter	Unit	8 kHz	16 kHz	31.746 kHz	44.444 kHz	47.619 kHz
S/(N+D)	dB	90	90	96	108	108
Passband	kHz	3.4	6.8	14.4	19.8	21.6
Passband ripple	dB	1	1	1	0.5	0.5
Stopband	kHz	4.6	9.2	17.6	24.26	26.4
Stopband attenuation	dB	80	80	60	60	60

The different interpolation filters are implemented using a dedicated processing core, ASMD (Application Specific Multi-rate DSP), which has an optimized data path to perform filter operations, data memory and a state machine which controls the data path such that the respective filter is implemented. The filter program executed by the state machine can be selected through an entry in register [AFE\\_BCON \(on Page 537\)](#).RXRATE.

### 8.10.3.2 Decimation Filter

The digital decimation filter on PMB7870 has two operation modes: 8 kHz output sampling rate and 16 kHz output sampling rate. The filter decimates the data sampling rate from down to 8 kHz or 16 kHz, respectively, generating one 16-bit voice sample every 125 µs or 62.5 µs, respectively. Each sample is stored in the audio transmit buffer. The attenuation values of the digital filters in 16 kHz mode can be obtained from those of the 8 kHz mode by doubling the values on the frequency axis.

## 8.10.4 Analog Part

The analog part of the PMB7870 audio front-end in the receive direction consists of a digital to analog converter (multi-bit oversampling converter) which transforms the output of the digital interpolation filter into analog signals. It is followed by the gain control/ amplifier section. The DAC outputs can be switched to several output buffers.

In the transmit section there is an input multiplexer which selects either one of two differential microphone inputs to be connected to the low-noise amplifier and analog pre-

**CONFIDENTIAL****Audio Front-End**

filter. The signals from the analog pre-filter are input to a second-order sigma-delta analog-to-digital converter. An overview of the analog part of the PMB7870 audio front-end is given in [Figure 8-52 \(on page 533\)](#).

#### **8.10.4.1 Audio Receive Part and Ringer Support**

The analog audio receive part consists of a multi-bit digital-to-analogue converters (DAC) and an output stage.

The signal sources are switched to the output drivers in the output stage. The output drivers consist of one fully differential driver for connecting to an external amplifier or internal speaker and one fully differential ended drivers for directly connecting headsets, earpiece devices, or the ringer.

##### **8.10.4.1.1 Digital-to-Analog Converter**

The multi-bit oversampling DAC of the PMB7870 audio front-end converts the 16-bit data words coming from the digital interpolation filters into analogue signals.

##### **8.10.4.1.2 Output Selection**

The output selection of PMB7870 audio receive part enables the individual and independent operation of up to two audio devices. The receive section has two fully differential output amplifiers; they can be selected separately using [AFE\\_VRXCTRL2.VEPPA](#) and [AFE\\_VRXCTRL2.VEPSP](#).

Gains of the individual outputs can also be adjusted separately so that a maximal degree of flexibility is achieved. The structure is illustrated in [Figure 8-51 \(on page 532\)](#).

##### **8.10.4.1.3 Output Amplifiers**

There are two fully differential output buffers on PMB7870. These are driven by the outputs of the selection block. The differential buffer (speaker driver) may be used to drive an external amplifier or a 16  $\Omega$  earpiece only, at -12 dBFS gain. The fully differential power buffer is able to drive a 16  $\Omega$  load in differential or single-ended operation mode. Furthermore the buffers can individually be put into tristate mode, power down, by setting the appropriate power-up bits, [AFE\\_VRXCTRL2.VEPSP](#) and [AFE\\_VRXCTRL2.VEPPA](#). Over all analog gains of each output channel (refer to [Table 8-38](#)) can be adjusted in eight analog stages. Intermediate gain settings can be achieved by digital gain control. The gain is normalized to 0 dBFS which is 3.7 Vpp differential and 1.85 Vpp single-ended.

**Table 8-38 Overall Analog Gain for Output Buffers**

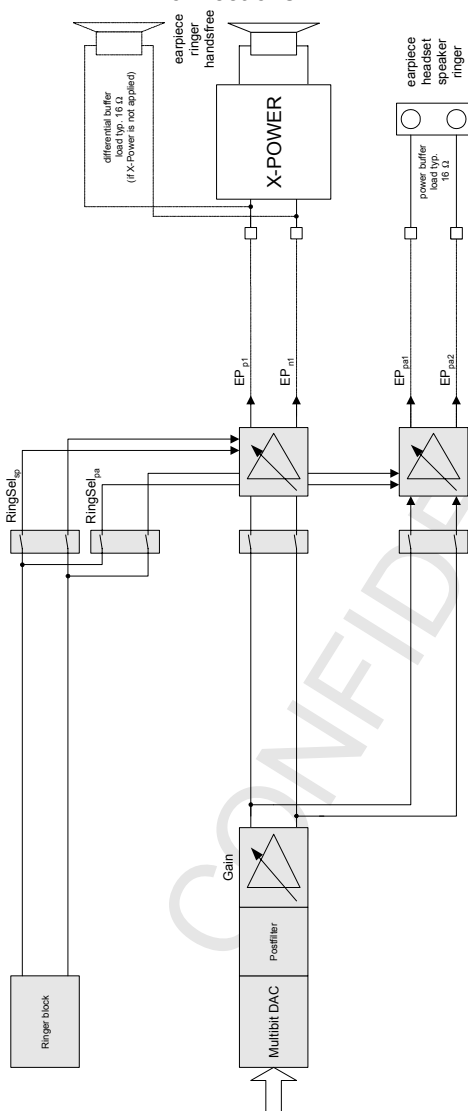
VGRXx	Gain Stage [dB]
000	0
001	-3
010	-6
011	-9
100	-12
101	-15
110	-18

To adjust the driver capabilities of the earpiece amplifiers for different load capacitances it is recommended to use the control bit settings of [AFE\\_VRXCTRL1.EPSAV](#) given in [Table 8-39](#).

**Table 8-39 Settings of [AFE\\_VRXCTRL1](#) Control Bits in for Earpiece Amplifiers**

Control bit	Load capacitance CL		
	< 2.5 nF	2.5 nF * CL * 5nF	5 nF < CL * 10 nF
EPSAV	1	1	0

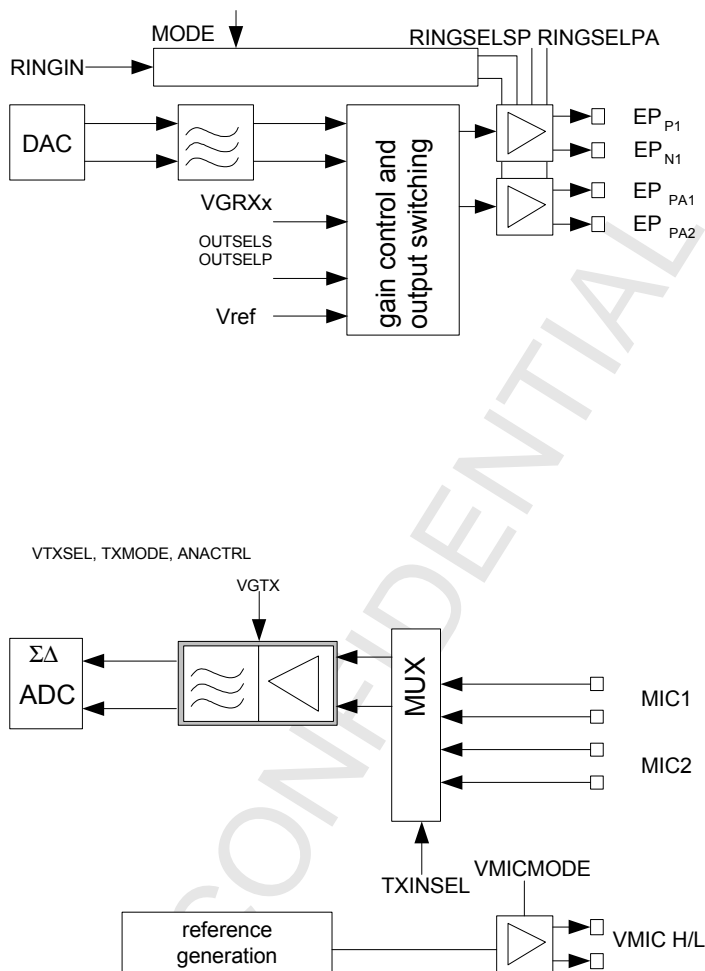
**Figure 8-51 Output Selection: Block Diagram and Potential External Connections**



*Note: This is example application scenario.*



**Figure 8-52 Block Diagram of Analog Part of PMB7870 Audio Front End**



#### 8.10.4.1.4 Ringer Mode

The PMB7870 audio part offers support for ringing over an earpiece device. Due to the sound pressure requirements for this function, it is advisable to use the earpiece device connected to the buffer on S/M-Power. There are two ways in which the ringer tones can be generated:

1. Digital tone sources can be transmitted over the normal digital to analog converter path and can be used to drive the acoustic transducer.
2. The PMB7870 supports driving a ringer with a digital PWM signal RINGIN.

Either the line-out driver EPp1 and EPn1 or the power buffer EPpa1 and EPpa2 can be switched into ringer mode by setting bits [AFE\\_VRXCTRL2 \(on Page 539\)](#).RINGSELSP and [AFE\\_VRXCTRL2.RINGSELPA](#). The input to RINGIN can be selected by setting bits [AFE\\_RINGCTRL \(on Page 541\)](#).RINS0 and [AFE\\_RINGCTRL.RINS1](#). The ringer is usually a PWM signal which can be programmed conveniently in the CAPCOM units. [Figure 8-55 \(on page 543\)](#) shows the internal connection of CAPCOM unit 0 (signal CC0CC6IO) and CAPCOM unit 1 (signal CC1CC6IO) to the ringer buffer. There are two ringer modes available:

- **Analog Ringer Mode:** the output of the CAPCOM unit 0 or 1 is connected to a specific driver circuit which guarantees a definite slew rate of the ringer output signal. This feature ensures that excessive load of power supply circuits and overshoots at the ringer outputs due to inductive load are avoided. The analog ringer mode (**RING** mode) is characterized by rise time for the first slope, rise and fall times for slopes during ringer operation and fall time for the last slope (transition to **RINGHOLD** mode) given in [Table 12-43 Electrical Characteristics of Audio Transmit Path \(on Page 1367\)](#). **RINGHOLD** mode ensures transition from **RING** mode to any other mode with a definite slew rate. The user should take care when leaving the **RING** mode to switch first to **RINGHOLD** mode to avoid overshoots at the ringer outputs (**RING** then **RINGHOLD** then any other mode). For the ringer wave form see [Figure 13-25 Timing Diagram of I2S Signals in Burst Mode - Master Mode \(on Page 1350\)](#).
- **Digital Ringer Mode:** The digital ringer mode does not provide any slew rate control. It is kept for compatibility with previous versions and uses the same control settings as in the previous versions.

#### 8.10.4.2 Audio Transmit Path

The audio transmit path of PMB7870 supports the selection of two differential microphone input sources, MIC1and MIC2:

- The inputs for microphone MIC1 are MIC1P and MIC1N.
- The inputs for microphone MIC2 are MIC2P and MIC2N.

**CONFIDENTIAL**

**Audio Front-End**

The transmit path consists of an input selector, a low noise amplifier with gain control, a second order  $\Sigma\Delta$ -converter, and a digital decimation filter. It supports both standard GSM (bandwidth 3.5 kHz) and wideband (bandwidth 7 kHz) speech bands.

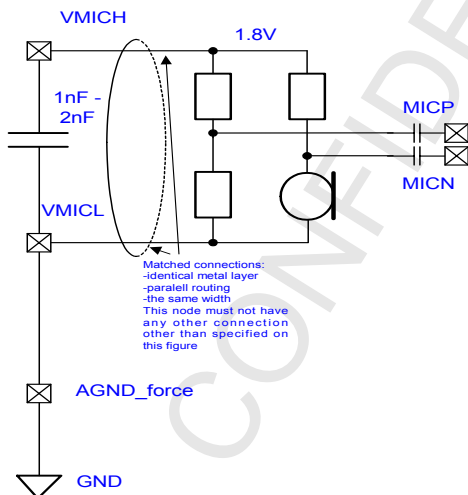
The differential input signal from the microphone first passes a low noise amplifier with gain settings ranging from 0dB to +42 dB and an anti-aliasing pre-filtering stage. This signal is then modulated by a second order  $\Sigma\Delta$ -converter which is clocked with the same clock rate as the digital to analog converters. The  $\Sigma\Delta$ -converter delivers a 1-bit pulse density modulated data stream at a rate of 2MHz to the digital decimation filter which reduce the rate to 8 kHz or 16 kHz depending on the current mode.

### 8.10.4.3 Microphone Supply

PMB7870 has a power-supply concept for electret microphones which allows two connection modes:

- Without low-pass filtering: A possible application example is given in [Figure 8-53](#). The microphone can be connected to the PMB7870 as shown in [Figure 8-54](#), although without low-pass filtering.

**Figure 8-53 Microphone Supply Generation Without Low-Pass Filtering**



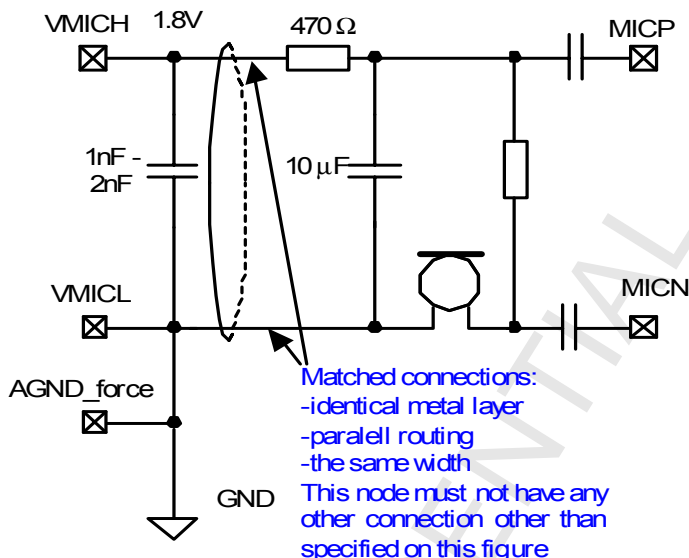
- With low-pass filtering: A possible application is given in [Figure 8-54](#). The microphone can be connected to the PMB7870 as shown in [Figure 8-53](#) using low-pass filtering.

For both modes a minimal load capacitance of 1nF is necessary to guarantee stable operation of the buffer. The maximal load capacitance must not exceed 2nF.

CONFIDENTIAL

Audio Front-End

Figure 8-54 Microphone Supply Generation with Low-Pass Filtering



#### 8.10.4.4 Audio Front-End Control Registers

Table 8-40 Overview of Audio Front-End Registers

Register name	Description
<a href="#">AFE_INTPTR</a>	Interrupt pointers for audio front end (see <a href="#">Section 8.10.3</a> )
<a href="#">AFE_RWADDR</a>	Addresses for read and write pointers of interface buffers (see <a href="#">Section 8.10.3</a> )
<a href="#">AFE_BCON</a>	Control register for basic digital configurations
<a href="#">AFE_VRXCTRL1</a>	Output buffer and DAC settings for receive path
<a href="#">AFE_VRXCTRL2</a>	Auxiliary settings for audio front end
<a href="#">AFE_VTXCTRL</a>	Control settings for audio transmit path
<a href="#">AFE_RINGCTRL</a>	Ringer settings for receive path

**CONFIDENTIAL**

**Audio Front-End**

## AFE\_BCON

### Basic Control of Audio Front-End

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED									TX RATE	TX START	RXRATE			RX START	MOD E

Field	Bits	Type	Description
<b>MODE</b>	0	rw	<b>Reset of Digital Filters</b> 0 Power down 1 Audio front-end operation (using analogue functions)
<b>RXSTART</b>	1	rw	<b>Start Operation of Audio Receive Path</b> 0 Disabled 1 Enabled
<b>RXRATE</b>	4:2	rw	<b>Rate Selection for Interpolation Path</b> 000: 8 kHz (Default) 001: 16 kHz 010: 31.746 kHz 011: 44.444 kHz 100: 47.619 kHz
<b>TXSTART</b>	5	rw	<b>Start Operation of Audio Transmit Path</b> 0 Disabled 1 Enabled
<b>TXRATE</b>	6	rw	<b>Select Output Rate of Transmit Path</b> 0: 8 kHz 1: 16 kHz
<b>RESERVED</b>	15:6	r	Reserved; these bits must be left at their reset values.

CONFIDENTIAL

Audio Front-End

## AFE\_VRXCTRL1

### Receive Control Register

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDITH	RESERVED	RXGAINPA				RXGAINSP			VCMEPB			RESERVED	EPS AV	RESERVED	RESERVED

Field	Bits	Type	Description
EPSAV	2	rw	<b>Enable Power Save Mode for Earpieces</b> 0 Disabled (Reset Value) 1 Enabled
VCMEPB	6:4	rw	<b>Common Mode Voltage for Earpiece Buffers</b> 000 VDD/2 001 1.10 V (typical) 010 1.15 V (typical) 011 1.20 V (typical) 100 1.25 V (typical) 101 1.30 V (typical) 110 1.35 V (typical) 111 1.40 V (typical)
RXGAINSP	9:7	rw	<b>Gain Setting for Line-Out Audio Buffer EPp1 and EPn1</b> (refer to <a href="#">Table 8-38</a> )
RXGAINPA	12:10	rw	<b>Gain Setting for Power Audio Buffer EPpa1 and EPpa2</b> (refer to <a href="#">Table 8-38</a> )
RXDITH	15:14	rw	<b>Dither Control for RX DAC</b> 00 Reserved 01 -60 Dbfs Dither Amplitude 10 Reserved 11 Reserved
RESERVED	13, 3, 1:0	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**Audio Front-End**

## AFE\_VRXCTRL2

### Receive Mode Register

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	MODE	RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	ZERODETECT	VEPSP	RESERVED	VEPPA	RINGSELP	RINGSELS	RESERVED	RXDAC	

Field	Bits	Type	Description
<b>RXDAC</b>	0	rw	<b>Enable RX Channel DAC</b> 0 Disabled (power down) 1 Enabled
<b>RINGSELSP</b>	2	rw	<b>Setup Ringer Mode (CAPCOM Input) for Line-Out Buffers EPp1 and EPn1</b> 0 Disabled 1 Enabled <i>Note: Selecting ringer mode overrides the settings for outputs EP<sub>p1</sub> and EP<sub>n1</sub>.</i>
<b>RINGSELPA</b>	3	rw	<b>Setup Ringer Mode (CAPCOM input) for Power Audio Buffers EPpa1 and EPpa2</b> 0 Disabled 1 Enabled <i>Note: Selecting ringer mode overrides the switch matrix settings for outputs EP<sub>pa1</sub> and EP<sub>pa2</sub>.</i>
<b>VEPPA</b>	4	rw	<b>Earpiece Amplifier EPpa1 and EPpa2</b> 0 Power down 1 on
<b>VEPSP</b>	6	rw	<b>Earpiece Amplifier EPn1 and EPp1</b> 0 Power down 1 on
<b>ZERODETECT</b>	7	rw	<b>Enable Zero Detect Feature</b> 0 Disabled 1 Enabled <i>Note:</i>

**CONFIDENTIAL**

**Audio Front-End**

Field	Bits	Type	Description
<b>MODE</b>	14	rw	<b>Operation Modes of Ringer and Earpieces Used In Conjunction with VEPPA and VEPSP Bits</b> Refer to <a href="#">Table 8-41</a> and <a href="#">Table 8-42</a> for the conditions necessary for each mode.
<b>RESERVED</b>	15, 11:10, 5, 1	r	Reserved; these bits must be left at their reset values.

### AFE\_VTXCTRL

#### Receive Mode Register

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				VMIC		TXINSEL		RESERVED	TXMODE	TXGAIN3	TXGAIN2	TXGAIN1	TXGAIN0	TXDITH	

Field	Bits	Type	Description
<b>TXDITH</b>	0	rw	<b>Enable Dither for TX ADC</b> 0 Disabled (reset value) 1 Enabled
<b>TXGAIN0<sup>1)</sup></b>	1	rw	<b>Switch on 3dB Gain Stage</b> 0: 0dB gain 1: +3dB gain setting
<b>TXGAIN11)</b>	2	rw	<b>Switch on 6dB Gain Stage</b> 0: 0dB gain 1: +6dB gain setting
<b>TXGAIN21)</b>	3	rw	<b>Switch on 12dB Gain Stage</b> 0: 0dB gain 1: +12dB gain setting
<b>TXGAIN31)</b>	4	rw	<b>Switch on 24dB Gain Stage</b> 0: 0dB gain 1: +24dB gain setting
<b>TXMODE</b>	6:5	rw	<b>Analog Operation Mode Setting for Transmit Path</b> 00 TX power down 01 Normal talk mode of transmit path 10 Reserved 11 Reserved



**CONFIDENTIAL**

**Audio Front-End**

Field	Bits	Type	Description
<b>TXINSEL</b>	10:8	rw	<b>Select Sources for Transmit Path</b> 100 MIC1 inputs on 010 MIC2 inputs on All other combinations: Reserved
<b>VMIC</b>	12:11	rw	<b>Microphone Supply Voltage</b> 00 Power down 01 1.8 V (typical) for Vdd = 2.25 V ... 2.75 V 10 2.0 V (typical) for Vdd = 2.4 V ... 2.75 V 11 2.2 V (typical) for Vdd = 2.5 V ... 2.75 V
<b>RESERVED</b>	15:13	r	Reserved; these bits must be left at their reset values.

1) Total TX gain is 42 dB maximum.

#### AFE\_RINGCTRL

**Ringer Control of Audio Front-End**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED													RINS	ALT RIN	

Field	Bits	Type	Description
<b>ALTRIN</b>	0	rw	<b>Alternative Ringer Output Selection</b> 0 Disabled 1 Enabled <i>Note: The alternative ringer output can be connected as an alternative source for the ringer output.</i>
<b>RINS</b>	2:1	rw	<b>Ringer Input Selection</b> 00 From CAPCOM unit 0 (CC0CC6 O) 01 From CAPCOM unit 1 (CC1CC6 O) 10 Voiceband control register bit <b>ALTRIN</b> 11 Reserved
<b>RESERVED</b>	15:3	r	Reserved; these bits must be left at their reset values.

**Table 8-41 Control Bits for Ringing via Line-Out Buffers (X = 'don't care', that is, RINGSELSP = 1<sup>1)</sup>)**

MODE	VEPSP	Operation Mode of Ringer/Earpieces
0	1	Ringer hold mode ( <b>RINGHOLD</b> )
0	0	Digital ringer mode (slew-rate circuit not used)
1	1	Analog ringer mode ( <b>RING</b> ) (slew-rate circuit used)
1	0	reserved

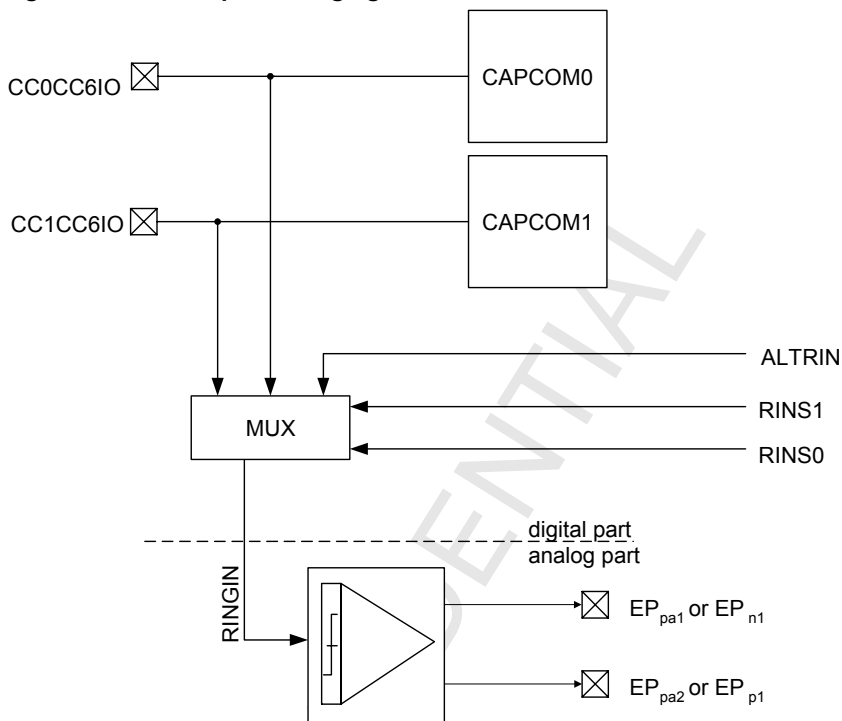
<sup>1)</sup> All other combinations are reserved.

**Table 8-42 Control bits for ringing via internal power buffers (X = 'don't care'), that is, RINGSELPA = 1<sup>1)</sup>)**

MODE	VEPPA	Operation Mode of Ringer/Earpieces
0	1	Ringer hold mode ( <b>RINGHOLD</b> )
0	0	Digital ringer mode (slew-rate circuit not used)
1	1	Analog ringer mode ( <b>RING</b> ) (slew-rate circuit used)
1	0	reserved

<sup>1)</sup> All other combinations are reserved.

Figure 8-55 Principle of Ringing via the CAPCOM Units



CONFIDENTIAL

**CONFIDENTIAL**

**GSM Cipher Unit**

## 8.11 GSM Cipher Unit

History	
Current version: Rev. 1.06, 2005-11-04	
Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.06	

### System Integration on TEAKLite:

- Supply domain: VDD\_DSP
- Chip internal interfaces:
  - Clock domain: gclk\_dsp\_per
  - Bus domain: TEAKLite Z-Bus
  - Interrupt sources: none
- Chip external signals related to this block: none
- Monitor Pins: CIPH\_INT

#### 8.11.1 Introduction

This block calculates the GSM/EDGE encryption keystream and the GSM/EDGE decryption keystream.

It implements the following algorithms: A5/1, A5/2, and A5/3.

The cipher unit is split into 2 distinct hardware blocks:

1. The A5/1 and A5/2 hardware sub-block, refer to [Section 8.11.2 A51 and A52 \(on Page 546\)](#).
2. The A5/3 hardware sub-block, refer to [Section 8.11.3 A53 \(on Page 552\)](#).

The selection between these two blocks is explained in [Section 8.11.4 Ciphering Hardware Block Selection \(on Page 564\)](#).

## 8.11.2 A51 and A52

### 8.11.2.1 Functional Overview

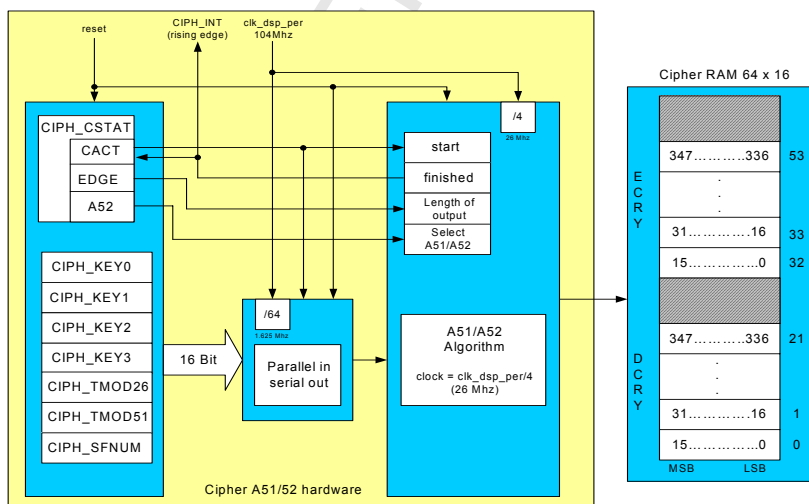
On E-GOLDradio the ciphering algorithms A51 and A52 are both implemented in hardware. The ciphering unit simultaneously computes one encryption bit stream for an uplink burst and one decryption bit stream for a downlink burst. For this calculation the ciphering unit only requires the input keys and the TDMA frame counters. The output encryption/decryption bit streams are XOR combined with the uplink/downlink data bits by the DSP.

The unit can also be used for circuit switched services in both the GSM and EDGE modes. For packet switched service GPRS and EGPRS ciphering is done in the upper layer (LLC layer 2) where GEA and GEA2 are used. This ciphering unit fulfils the requirements for the GSM layer 1 encryption and decryption procedure and cannot be used for packet switched services.

The ciphering unit consists of a hardware block which contains two ciphering algorithms (A51/A52) and a 64x16-bit RAM to store the output encryption/decryption bit streams. The ciphering hardware includes 4 registers for the input keys and 3 registers for the input TDMA frame counters. **Figure 8-56** is a block diagram of the ciphering unit.

The signal `gclk_dsp_per` from the CGU and a clock divider by 4 which is located in the Cipher Unit peripheral are used for the Cipher Unit clock. Thus the clock is equal to the TEAKLite peripheral clock divided by 4 (signal: `gclk_dsp_per/4`).

**Figure 8-56 Block Diagram of the Ciphering Unit**



### 8.11.2.2 Generating Encryption and Decryption Sequences

Based on a typical sequence, this section describes how the ciphering unit can be used. The output format of the encryption/decryption bit stream is described in [Section 8.11.2.4 Cipher RAM \(on Page 552\)](#); the use of the input key registers and the status register is shown in [Section 8.11.2.3 A51 and A52 Register Description \(on Page 548\)](#). The procedure follows:

1. First the TEAKLite must write the 64-bit cipher key KC(63:0) to the four 16-bit registers **CIPH\_KEY0**,
2. , **CIPH\_KEY2** and **CIPH\_KEY3** and the current 22-bit TDMA frame number to the registers **CIPH\_TMOD26** (5 bit)s, **CIPH\_TMOD51** (6 bits) and **CIPH\_SFNUM** (11 bits). The contents of the registers **CIPH\_SFNUM**, **CIPH\_TMOD26** and **CIPH\_TMOD51** correspond to the frame number parameters T1, T2, and T3 given in the GSM Technical Specifications.
3. Bit **CIPH\_CSTAT** (on Page 548).**A52** selects either the A51 and A52 algorithm.
4. The cipher unit either operates in GSM or EDGE mode. The mode is configured by the bit **CIPH\_CSTAT.EDGE**. In GSM mode the cipher unit generates two 114 bit output sequences whereas in EDGE mode two 348 bit output sequences are obtained. The two output sequences (encryption/decryption) are written in the cipher RAM.
5. The ciphering sequence generation starts as soon as the bit **CIPH\_CSTAT.CACT** has been set. Then the clock is switched on automatically. The resulting 114-bit, resp. 348, sequences for decryption and encryption are stored in the Cipher RAM (see [Figure 8-56 \(on page 546\)](#)).
6. As soon as the last bit is stored, **CIPH\_CSTAT.CACT** is reset, the clock is switched off automatically, and the interrupt bit **INT\_FINT1.CIPH** is set.

*Note: The time between ciphering is started (**CIPH\_CSTAT.CACT** is set) and finished (**CIPH\_CSTAT.CACT** is reset) depends on the TEAKLite clock and is about 46  $\mu$ s (GSM) or about 120  $\mu$ s (EDGE) for both A51 and A52 if the TEAKLite is clocked with 104 Mhz. After this time the encryption and decryption data are available in the Cipher RAM.*

**CONFIDENTIAL**

**GSM Cipher Unit**

### 8.11.2.3 A51 and A52 Register Description

#### CIPH\_CSTAT

**Ciphering Unit Status**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED													EDGE	A52	CACT

Field	Bits	Type	Description
<b>CACT</b>	0	rw	0 Cipher Unit is not active 1 Cipher Unit is active Can always be read, writing a 1 switches on the clock and starts ciphering. After ciphering is completed <b>CACT</b> is automatically reset by hardware, the ciphering clock is switched off and the ciphering interrupt bit <b>INT_FINT1.CIPH</b> is set. The ciphering algorithm is latched in at the moment the bit <b>CACT</b> is set.
<b>A52</b>	1	rw	0 A51 ciphering hardware is selected 1 A52 ciphering hardware is selected
<b>EDGE</b>	2	rw	0 GSM mode is selected 1 EDGE mode is selected
<b>RESERVED</b>	15:2	r	Reserved; these bits must be left at their reset values.

*Notes:*

- Read and write to **CSTAT** is also possible if ciphering clock is switched off
- Writing to **CSTAT** while **CACT** = 1 is not allowed.
- While **CACT** = 1, it is not possible to read or write the Cipher RAM.
- While **CACT** = 1, the cipher registers must not be read or written.



**CONFIDENTIAL**

**GSM Cipher Unit**

### **CIPH\_KEY0**

**Ciphering Key Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KC(15:0)															

Field	Bits	Type	Description
<b>KC(15:0)</b>	15:0	rw	Bits 15:0 of the 64-bit Cipher Key for the A51 or A52 ciphering algorithm.

### **CIPH\_KEY1**

**Ciphering Key Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KC(31:16)															

Field	Bits	Type	Description
<b>KC(31:16)</b>	15:0	rw	Bits 31:16 of the 64-bit Cipher Key for the A51 or A52 ciphering algorithm.

### **CIPH\_KEY2**

**Ciphering Key Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KC(47:32)															

Field	Bits	Type	Description
<b>KC(47:32)</b>	15:0	rw	Bits 47:32 of the 64-bit Cipher Key for the A51 or A52 ciphering algorithm.

### **CIPH\_KEY3**

#### **Ciphering Key Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>KC(63:48)</b>															

<b>Field</b>	<b>Bits</b>	<b>Type</b>	<b>Description</b>
<b>KC(63:48)</b>	15:0	rw	Bits 63:48 of the 64-bit Cipher Key for the A51 or A52 ciphering algorithm.

### **CIPH\_TMOD26**

#### **TDMA Frame Number Mod-26 Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>											<b>T26N</b>				

<b>Field</b>	<b>Bits</b>	<b>Type</b>	<b>Description</b>
<b>T26N</b>	4:0	rw	Lower five bits of the TDMA frame number in modulo-26 form. The contents of TMOD26 correspond to T2 parameter in the GSM Technical Specifications.
<b>RESERVED</b>	15:6	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**GSM Cipher Unit**

**CIPH\_TMOD51**

**TDMA Frame Number Mod-51 Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>										<b>T51N</b>					

Field	Bits	Type	Description
<b>T51N</b>	5:0	rw	Lower six bits of the TDMA frame number in modulo-51 form. The contents of TMOD51 correspond to the parameter T3 in the GSM Technical Specifications.
<b>RESERVED</b>	15:6	r	Reserved; these bits must be left at their reset values.

**CIPH\_SFNUM**

**Superframe Number Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>						<b>SFN</b>									

Field	Bits	Type	Description
<b>SFN</b>	10:0	rw	11-bit number counting the superframes within a hyperframe. The contents of SFNUM correspond to the T1 parameter in the GSM Technical Specifications.
<b>RESERVED</b>	15:6	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**GSM Cipher Unit**

### 8.11.2.4 Cipher RAM

The Cipher RAM is a synchronous RAM logically organized by 64 x 16 bit. The Cipher RAM must not be read or written while the cipher algorithm is running (**CIPH\_CSTAT.CACT** = 1), but read and write access from the TEAKLite is possible even if the local clock is switched off.

In **Figure 8-43** the format of the output encryption/decryption sequences within cipher RAM is shown.

**Table 8-43 Cipher Ram Output Encryption/Decryption Sequences**

Address	Word	GSM	EDGE
<b>Downlink/Decryption</b>			
offset+0 <sub>H</sub>	0	DCRY[15:0]	DCRY[15:0]
offset+0 <sub>H</sub>	1	DCRY[31:16]	DCRY[31:16]
...			
offset+06 <sub>H</sub>	6	DCRY[111:96]	DCRY[111:96]
offset+07 <sub>H</sub>	7	DCRY[113:112]	DCRY[127:112]
...			
offset+14 <sub>H</sub>	20	not used	DCRY[335:320]
offset+15 <sub>H</sub>	21	not used	DCRY[347:336]
...			
<b>Uplink/Encryption</b>			
offset+20 <sub>H</sub>	32	ECRY[15:0]	ECRY[15:0]
offset+21 <sub>H</sub>	33	ECRY[31:16]	ECRY[31:16]
...			
offset+26 <sub>H</sub>	38	ECRY[111:96]	ECRY[111:96]
offset+27 <sub>H</sub>	39	ECRY[113:112]	ECRY[127:112]
...			
offset+34 <sub>H</sub>	52	not used	ECRY[335:320]
offset+35 <sub>H</sub>	53	not used	ECRY[347:336]
...			

### 8.11.3 A53

**Note: The A53 implementation is based on ETSI May 2002 release.**

### 8.11.3.1 Functional Overview

This block computes the GSM/EDGE encryption keystream for an uplink burst and the decryption keystream for a downlink burst. It performs the ciphering algorithms A53 used for GSM/EDGE.

The inputs are:

- The cipher key  $K_C$ .
- The T1, T2 and T3 TDMA frame counters.
- The hardware block control (activation).

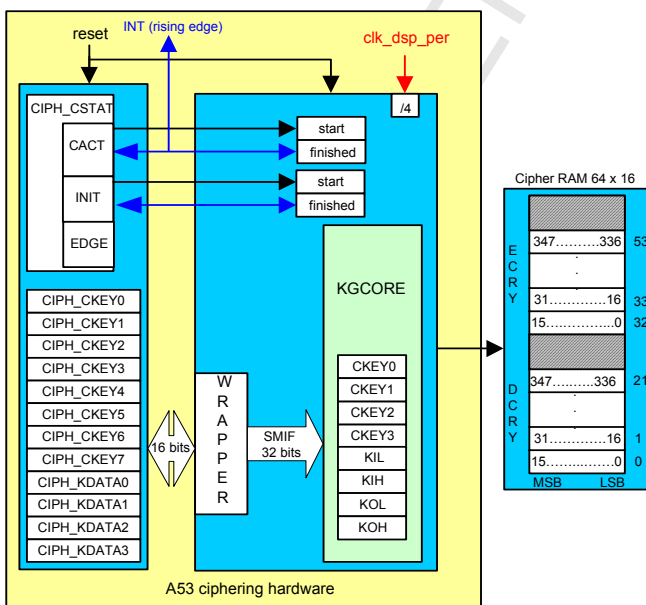
The outputs are:

- BLOCK1 and BLOCK2.

The encryption keystream is named BLOCK1 and the decryption keystream is named BLOCK2.

The output encryption and decryption bit streams are **not** XOR combined with respectively the uplink and downlink data bits by the hardware (but by the TEAKLite).

**Figure 8-57 Data and Control Flow**



The  $K_C$ , T1, T2, T3, and the Mode need to be provided to the hardware block before the activation. Once the activation is done BLOCK1 and BLOCK2 are processed and stored in the Cipher RAM.

**CONFIDENTIAL**

**GSM Cipher Unit**

The decrypted data are obtained with the XOR operation with the BLOCK1 like follows:

Decrypted data bit 0 = (Encrypted data Bit0) XOR (BLOCK1 Bit 0) [0.14]

Decrypted data bit N = (Encrypted data BitN) XOR (BLOCK1 Bit N) [0.15]

*Note: N equals to 114 for the GSM mode and 348 for the EDGE mode.*

The encrypted data are obtained with the XOR operation with the BLOCK2 (: Open in the ETSI spec) like follows:

Encrypted data bit 0 = (Non encrypted data Bit0) XOR (BLOCK2 Bit 0) [0.16]

Encrypted data bit N = (Non encrypted data BitN) XOR (BLOCK2 Bit N) [0.17]

*Note: N equals to 114 for the GSM mode and 348 for the EDGE mode.*

CONFIDENTIAL

### 8.11.3.2 Programming Procedure

The following procedure has to be used to control the A53 block. The steps are ordered in chronological order:

1. To select the A53 ciphering hardware set the **CIPH\_CSTAT.A53** bit to 1.

*Note: **CIPH\_CSTAT.A53** is implemented at the top level of the cipher, for the selection between the two cores (cipher A51/52 or cipher A53).*

2. To activate the cipher unit set the **CIPH\_CSTAT.CACT** bit to 1 to switch on the clock.

*Note: **CIPH\_CSTAT.CACT** is active during the init and ciphering phases.*

3. The cipher unit either operates in GSM or EDGE mode. The mode is configured by the bit **CIPH\_CSTAT.EDGE** and **CIPH\_KDATA1.CA** (see [Section 8.11.3.4 Algorithm \(on Page 556\)](#)). In GSM mode the cipher unit generates two 114 bit output sequences whereas in EDGE mode two 348 bit output sequences are obtained. The two output sequences (encryption/decryption) are written in the cipher RAM.

4. To initialize the keystream generator with the input variables, the TEAKlite must:

- a) Write into the registers **CIPH\_KDATAx.CE**, **CIPH\_KDATAx.CB**, **CIPH\_KDATAx.CD**, **CIPH\_KDATAx.CC** and **CIPH\_KDATAx.CC** (refer to [Section 8.11.3.5.2 CIPH\\_KDATA Input Registers \(on Page 562\)](#) and [Section 8.11.3.4 Algorithm \(on Page 556\)](#)).

- b) Write the cipher key  $K_C$  into the [Ciphering Key Registers \(on Page 559\)](#).

*Note: If the  $K_C$  length is < 128 bits then*

***CIPH\_CKEYx** bit 0 to ( $K_C$  length - 1) must be set to the  $K_C$  bit 0 to ( $K_C$  length - 1).*

***CIPH\_CKEYx** bit  $K_C$  length to 127 must be set to  $K_C$  bit 0 to ( $127 - K_C$  length).*

5. To start the init phase set the **CIPH\_CSTAT.INIT** bit to 1. After a while the ciphering sequence starts. The resulting 114-bit GSM (348 bit EDGE) sequence for decryption and encryption are stored in the cipher RAM. When the last bit is stored, the bit **CIPH\_CSTAT.CACT** is reset, the clock is switched off automatically, and the interrupt bit **INT\_FINT1.CIPH** is set.

Now the TEAKlite can read the data from the Cipher RAM.

### 8.11.3.3 Timing

[Figure 8-58](#) shows the timing needed.

In the GSM mode, the A53 algorithm produces two blocks of 114 bits each 4.615 ms. (BLOCK1 for enciphered, BLOCK2 for deciphered).

In EDGE, the block size is greater than 114 bits. The algorithm produces both blocks which contains 348 bits during a TDMA frame duration in 4.615 ms.

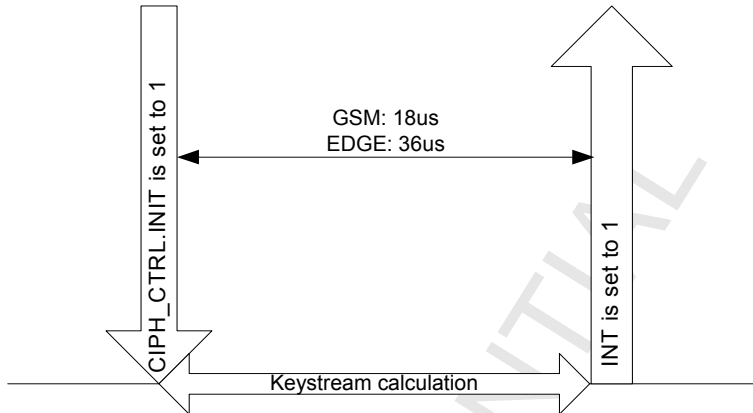
The signal **gclk\_dsp\_per** from the CGU and a clock divider per 4 which is located in the Cipher Unit peripheral are used for the Cipher Unit clock. Thus the clock is equal to the

**CONFIDENTIAL**

**GSM Cipher Unit**

TEAKLite peripheral clock divided per 4 (signal: gclk\_dsp\_per/4), for example, if the TEAKLite clock is 104 Mhz the Cipher Unit clock is 26 Mhz.

**Figure 8-58 Timing**



### 8.11.3.4 Algorithm

The A53 algorithm is a stream cipher that is used to encrypt/decrypt blocks of data under a confidentiality key Kc.

Each of these algorithms are based on the 3 GPP Kasumi algorithm. Kasumi is a block cipher that produces a 64-bit output from a 64-bit input under control of a 128-bit key.

The individual encryption for GSM is defined using a keystream generation function KGCORE.

The KGCORE contains 8 times 32-bit registers (CKEY0, CKEY1, CKEY2, CKEY3, KIL, KIH, KOL, and KOH) linked with the A53 registers.

As these specifications are confidential the algorithm is not explained in this section, only the link between the parameters from the KGCORE and the register list is described (refer to [Table 8-44](#)). The functionality is confidential.



**CONFIDENTIAL**

**GSM Cipher Unit**

**Table 8-44 KGCORE Parameters and A53 Registers**

<b>KGCORE Parameters</b>	<b>A53 Register Name or Value</b>
CA	<b>CIPH_KDATA2.CA</b> In mode GSM CA = 00001111 In mode EDGE CA = 11110000
CB	<b>CIPH_KDATA2.CB</b> CB = 00000
CC	CC Bit 0 to 4 = (T2), <b>CIPH_KDATA3.T26N</b> CC Bit 5 to 10 = (T3), <b>CIPH_KDATA3.T51N</b> CC Bit 11 to 21 = (T1), <b>CIPH_KDATA3.SFN(4:0)</b> and <b>CIPH_KDATA4.SFN(10:5)</b>
CD	<b>CIPH_KDATA2.CD</b> CD = 0
CE	<b>CIPH_KDATA1.CE</b> CE = 0000000000000000
CK	CK bit 0 to 127 = <b>CIPH_KEY0</b> , <b>CIPH_KEY1</b> , <b>CIPH_KEY2</b> , <b>CIPH_KEY3</b> , <b>CIPH_KEY4</b> , <b>CIPH_KEY5</b> , <b>CIPH_KEY6</b> and <b>CIPH_KEY7</b> .  <i>Note: If the Kc length is &lt; 128 bits then CKEYx bit 0 to (Kc length - 1) must be set to the Kc bit 0 to (Kc length - 1) CIPHx bit Kc length to 127 must be set to Kc bit 0 to (127 - Kc length)</i>

**CONFIDENTIAL**

**GSM Cipher Unit**

### 8.11.3.5 A53 Register Descriptions

**Note:** The A53 registers cannot be read or written while A53 calculates the keystream (**CIPH\_CSTAT.CACT** equals 1).

#### **CIPH\_CSTAT**

**Ciphering Unit Status**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>											<b>A53</b>	<b>INIT</b>	<b>EDGE</b>	<b>RESERVED</b>	<b>CACT</b>

Field	Bits	Type	Description
<b>CACT</b>	0	rw	0 A53 block does not calculate the keystream 1 A53 block calculates the keystream After ciphering is completed <b>CACT</b> is reset by hardware, and the interrupt INT is generated. <i>Note: The A53 registers and Cipher RAM cannot be read or written while A53 calculates the keystream (<b>CACT</b> equals 1).</i>
<b>EDGE</b>	2	rw	0 GSM mode is selected 1 EDGE mode is selected
<b>INIT</b>	3	rw	0 Initialization phase of A5/3 algorithm is finished 1 Initialization phase of A5/3 algorithm is running <i>Note: After initialization phase is completed the INIT is automatically reset by hardware.</i>
<b>A53</b>	4	rw	Enable A53 Ciphering Block 0 Turns off the A53 clock (disables the A53 block) 1 Turns on the A53 clock (enables the A53 block)
<b>RESERVED</b>	1,15:5	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

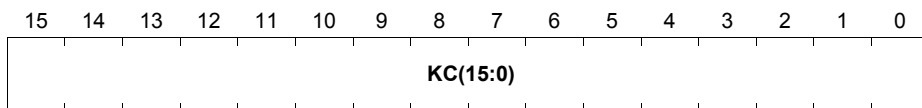
**GSM Cipher Unit**

### 8.11.3.5.1 Ciphering Key Registers

#### CIPH\_KEY0

**Ciphering Key Register**

**Reset value: 0000<sub>H</sub>**

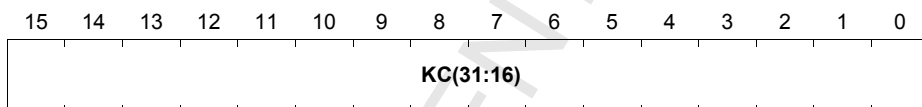


Field	Bits	Type	Description
KC(15:0)	15:0	rw	Bits 15:0 of the 128-bit Cipher Key

#### CIPH\_KEY1

**Ciphering Key Register**

**Reset value: 0000<sub>H</sub>**

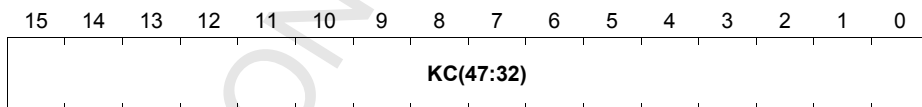


Field	Bits	Type	Description
KC(31:16)	15:0	rw	Bits 31:16 of the 128-bit Cipher Key

#### CIPH\_KEY2

**Ciphering Key Register**

**Reset value: 0000<sub>H</sub>**



Field	Bits	Type	Description
KC(47:32)	15:0	rw	Bits 47:32 of the 128-bit Cipher Key

**CONFIDENTIAL**

**GSM Cipher Unit**

**CIPH\_KEY3**

**Ciphering Key Register 3**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KC(63:48)															

Field	Bits	Type	Description
<b>KC(63:48)</b>	15:0	rw	Bits 63:48 of the 128-bit Cipher Key

**CIPH\_KEY4**

**Ciphering Key Register 1**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KC(79:64)															

Field	Bits	Type	Description
<b>KC(79:64)</b>	15:0	rw	Bits 79:64 of the 128-bit Cipher Key

**CIPH\_KEY5**

**Ciphering Key Register 1**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KC(95:80)															

Field	Bits	Type	Description
<b>KC(95:80)</b>	15:0	rw	Bits 95:80 of the 128-bit Cipher Key

**CONFIDENTIAL**

**GSM Cipher Unit**

### **CIPH\_KEY6**

**Ciphering Key Register 1**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KC(111:96)															

Field	Bits	Type	Description
<b>KC(111:96)</b>	15:0	rw	Bits 111:96 of the 128-bit Cipher Key

### **CIPH\_KEY7**

**Ciphering Key Register 7**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KC(127:112)															

Field	Bits	Type	Description
<b>KC(127:112)</b>	15:0	rw	Bits 127:112 of the 128-bit Cipher Key

**CONFIDENTIAL**

**GSM Cipher Unit**

### 8.11.3.5.2 CIPH\_KDATA Input Registers

See [Figure 8-59 Block Selection \(on Page 565\)](#).

#### CIPH\_KDATA1

##### CIPH\_KDATA1 Input Register

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CE(15:0)															

Field	Bits	Type	Description
CE	15:0	rw	CE = 0000000000000000

#### CIPH\_KDATA2

##### CIPH\_KDATA2 Input Register

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CB(4:0)					CD	RESERVED	CA(7:0)								

Field	Bits	Type	Description
CA	7:0	rw	GSM mode: 00001111 EDGE mode: 11110000
CD	10	rw	CD = 0
CB	15:11	rw	CB = 00000
RESERVED	9:8	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**GSM Cipher Unit**

**CIPH\_KDATA3**

**CIPH\_KDATA3 Input Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>SFN(4:0)</b>					<b>T51N(5:0)</b>					<b>T26N(4:0)</b>					

Field	Bits	Type	Description
<b>T26N(4:0)</b>	4:0	rw	Lower five bits of the TDMA frame number in modulo-26 form. The contents of T26N correspond to T2 parameter in the GSM technical Specifications.
<b>T51N(5:0)</b>	10:5	rw	Lower six bits of the TDMA frame number in modulo-51 form. The contents of T51N correspond to T3 parameter in the GSM technical Specifications.
<b>SFN(4:0)</b>	15:11	rw	Bits 4:0 of SFNUM counter. SFNUM counts the superframes within a hyperframe. Its contents correspond to the T1 parameter in the GSM Technical Specification.

**CIPH\_KDATA4**

**CIPH\_KDATA4 Input Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>Reserved</b>										<b>SFN(10:5)</b>					

Field	Bits	Type	Description
<b>SFN</b>	5:0	rw	Bits 10:5 of SFNUM counter. SFNUM counts the superframes within a hyperframe. Its contents correspond to the T1 parameter in the GSM Technical Specification
<b>Reserved</b>	15:6	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**GSM Cipher Unit**

### 8.11.3.6 Cipher RAM

The cipher RAM is a 64 x 16-bit synchronous RAM.

*Note: The Cipher RAM cannot be read or written to while the A53 calculates the keystream (CIPH\_CSTAT.CACT = 1).*

A read or write access from the Cipher RAM to the TEAKLITE is possible even if local clock is switched off.

**Table 8-45** shows the format of the output encryption/decryption sequences.

**Table 8-45 Cipher RAM Format**

Address	16-Bit Word Number	GSM Keystream	EDGE Keystream (Optional Feature)
Offset + 0 <sub>H</sub>	0	BLOCK1[15:0]	BLOCK1[15:0]
..	..	..	..
Offset + 7 <sub>H</sub>	7	BLOCK1[113:112]	BLOCK1[127:112]
Offset + 8 <sub>H</sub>	8	Not used	BLOCK1[143:128]
..	..	..	..
Offset + 15 <sub>H</sub>	21	Not used	BLOCK1[347:336]
Offset + 20 <sub>H</sub>	32	BLOCK2[15:0]	BLOCK2[15:0]
..	..	..	..
Offset + 27 <sub>H</sub>	39	BLOCK2[113:112]	BLOCK2[127:112]
Offset + 28 <sub>H</sub>	40	Not used	BLOCK2[143:128]
...	...	...	...
Offset + 35 <sub>H</sub>	53	Not used	BLOCK2[347:336]

### 8.11.4 Ciphering Hardware Block Selection

The A51/A52 block is selected when the bit **CIPH\_CSTAT.A53** is set to 0.

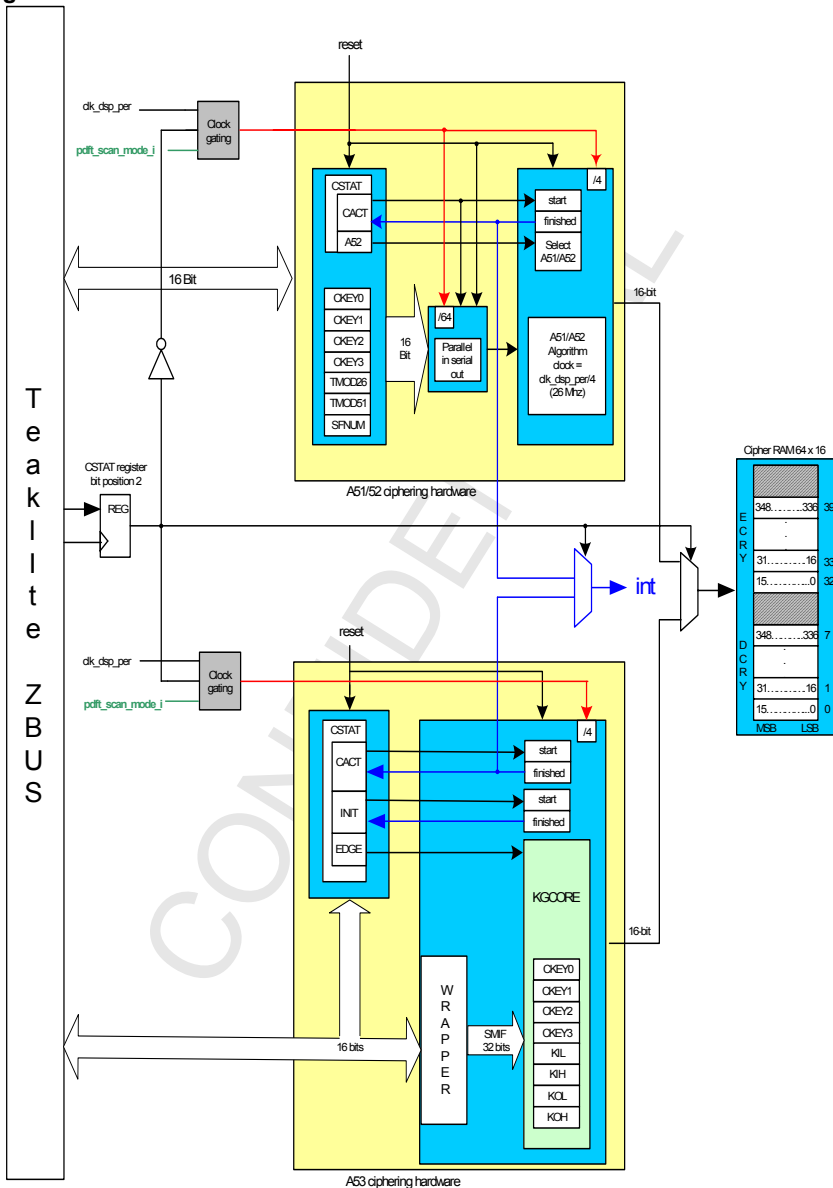
The A53 block is selected when the bit **CIPH\_CSTAT.A53** is set to 1.

When a block is selected its master clock is enabled, its interrupt output and the keystream output flow are selected (see **Figure 8-59**).

*Note: After the chip reset the A51/52 ciphering hardware is selected.*



**Figure 8-59 Block Selection**



CONFIDENTIAL

**CONFIDENTIAL**

**Viterbi Coprocessor, Equalizer**

## 8.12 Viterbi Coprocessor, Equalizer

History	
Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.04	
<b>Page 567</b>	Heading "Equalizer Accelerator" changed to <b>Viterbi Coprocessor, Equalizer</b> WS00007944
Changes for Rev. 1.06	

### System Integration on TEAKLite:

- Supply domain: VDD\_DSP
- Chip internal interfaces:
  - Clock domain: gclk\_dsp\_per
  - Bus domain: TEAKLite Z-Bus
- Interrupts: EQ
- Monitor Pins: Refer to **Section 11.8.10 Internal Signal Monitoring (on Page 1209)**

### 8.12.1 Interface

#### 8.12.1.1 TEAKLite Interface

The interface to the TEAKLite consists of a register bank of status and configuration registers and of one user defined core register (rd/wr, ext1). By means of this user defined core register the TEAKLite can access the two internal RAMs: RAM1 and RAM2 of the hardware peripheral and can initialize the two working RAMs: RAMW1 and RAMW2.

The entries of the configuration register **VH\_CONF1\_EQ** determine which RAM area of RAM1, RAM2 and of RAMW1, RAMW2 is addressed if address ext1 is accessed. The access to the internal RAMs is performed in a FIFO-like way starting at a base address and incrementing the current address each time a read or write access to ext1 is performed. By this means the TEAKLite can write the RAM areas for the input values of the equalizer, the partial branch metrics for the equalizer and so on. Bits within the configuration register **VH\_CONF1\_EQ** are provided to reset the current address of the special RAM area to the base address. After finishing writing the dedicated RAM area

**CONFIDENTIAL**

**Viterbi Coprocessor, Equalizer**

the internal data paths have to be enabled and will get access to these areas for internal arithmetical calculations.

**Figure 8-60** shows the TEAKLite interface and the HW internal registers and memories, **Figure 8-63** and **Figure 8-64** describe the partitioning of the HW internal memories RAM1/RAM2 and **Figure 8-65** gives the partitioning of the HW internal working memories RAMW1/RAMW2.

**Table 8-46 Registers**

Register or Memory	Address	Description
Configuration and status registers	Situated within the register address area.	The entries of the configuration registers define the RAM area which shall be overwritten or read. The equalization process can be enabled. The response of the status register confirms the end of the chosen process. All registers are accessible with 2 wait states.
RAM1, RAM2, RAMW1, RAMW2	Access via ext1 (user defined core register)	RAM1, RAM2, RAMW1, RAMW2 access: The addresses within the RAM1, RAM2, RAMW1 and RAMW2 are determined by the entries of the configuration register <b>VH_CONF1_EQ</b> . The ext1 register is accessible without wait states and is accessed only within the equalization procedure.

**Table 8-47 Register-Description (Configuration and Status Registers)**

Register Name	Comment
<b>VH_CONF1_EQ</b>	Configuration register 1 for equalizer
<b>VH_CONF2_EQ</b>	Configuration register 2 for equalizer
<b>VH_STATUS_EQ</b>	Status register
<b>VH_CONF_CNT_E</b>	Count configuration, number of symbols to be equalized
<b>VH_STAT_CNT_E</b>	Count status, number of symbols equalized
<b>VH_SC_SOUT</b>	Scaling value for soft outputs
<b>VH_SQUAL</b>	Output register for RX_QUAL histogram

**Example of access of the configuration registers and the internal RAMs:**

Before the RAM1, RAM2, RAMW1 or RAMW2 can be accessed the base address of the targeted RAM area has to be set in the configuration register **VH\_CONF1\_EQ**.

**CONFIDENTIAL****Viterbi Coprocessor, Equalizer**

The access to RAM1, RAM2, RAMW1 and RAMW2 is performed by reading from or writing to the TEAKLite address ext1.

**Example:** For writing input values for the equalizer to the RAM1 area the following procedure has to be performed:

- **VH\_CONF2\_EQ** is set to 0001<sub>H</sub> (**HW\_ENA\_EQ** = 1) and **VH\_CONF1\_EQ** is set to 0004<sub>H</sub> (set **RES\_RX\_BASE** = 1, reset RAM1 pointer to **RX\_BASE** and switches on clock).
- Every input value written to ext1 is stored within the selected region. The address starts at the base address and is successively incremented after each write access.

*Note: Only one of the bits RES\_...\_BASE may be set at a certain time. If more than 1 of the bits is set the resulting procedure will be erroneous. The firmware has to take care of this. After resetting the pointers by hardware the bits of **VH\_CONF1\_EQ** are reset automatically by hardware after 4 cycles.*

*Note: The access to ext1 and therefore to the RAMs RAM1, RAM2, RAMW1 and RAMW2 is performed without waitstates.*

TEAKLite restriction: The following TEAKLite instructions can access ext1:

- mov acc, ext1
- mov (r0), ext1
- mov ext1, acc
- mov ext1, (r0)

where r0 is a pointer to the data RAM of the TEAKLite.

**Note: If one specific RAM is reset to a chosen base address for a read operation the firmware must wait 4 cycles after writing the base address before accessing ext1. For writing no wait cycles are required.**

**CONFIDENTIAL**

**Viterbi Coprocessor, Equalizer**

### 8.12.1.1.1 Configuration Register 1 for Channel Equalizer

**VH\_CONF1\_EQ**

**Configuration Register 1 for Channel Equalizer**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES EB BASE	RES EPL BASE	RES EPR BASE	RES EML BASE	RES EMR BASE	RESERVED			RES ELAT BASE	RES ERV ED	RES HOUT BASE	RES SOUT BASE	RES BPAR BASE	RES RX BASE	RES ERV ED	RES RW1 RW2

Field	Bits	Type	Description
RES_RW1_RW2	0	w	Select between equivalent memory areas. 0: RAMW1 1: RAMW2
RES_RX_BASE	2	w	Reset RAM1 pointer to RX_BASE, reset RAM32_RD_FLAG_EQ and RAM32_WR_FLAG_EQ.
RES_BPAR_BASE	3	w	Reset RAM1 pointer to BPAR_BASE, reset RAM32_RD_FLAG_EQ and RAM32_WR_FLAG_EQ.
RES_SOUT_BASE	4	w	Reset RAM2 pointer to SOUT_BASE, reset RAM16_RD_FLAG_EQ and RAM16_WR_FLAG_EQ.
RES_HOUT_BASE	5	w	Reset RAM2 pointer to HOUT_BASE, reset RAM16_RD_FLAG_EQ and RAM16_WR_FLAG_EQ.
RES_ELAT_BASE	7	w	Reset RAM2 pointer to ELAT_BASE, reset RAM16_RD_FLAG_EQ and RAM16_WR_FLAG_EQ.
RES_EMR_BASE	11	w	Reset RAMW1/W2 pointer to EM_BASE_R, reset RAM32_RD_FLAG_EQ and RAM32_WR_FLAG_EQ.
RES_EML_BASE	12	w	Reset RAMW1/W2 pointer to EM_BASE_L, reset RAM32_RD_FLAG_EQ and RAM32_WR_FLAG_EQ.
RES_EPR_BASE	13	w	Reset RAMW1/W2 pointer to EP_BASE_R, reset RAM32_RD_FLAG_EQ and RAM32_WR_FLAG_EQ.

**CONFIDENTIAL**

**Viterbi Coprocessor, Equalizer**

Field	Bits	Type	Description
<b>RES_EPL_BASE</b>	14	w	Reset RAMW1/W2 pointer to EP_BASE_L, reset RAM32_RD_FLAG_EQ and RAM32_WR_FLAG_EQ.
<b>RES_EB_BASE</b>	15	w	Reset RAMW1/W2 pointer to EB_BASE, reset RAM32_RD_FLAG_EQ and RAM32_WR_FLAG_EQ.
<b>RESERVED</b>	1, 6, 10:8	r	<b>Reserved</b> ; reading returns 0; writing to these bit positions has no effect.

CONFIDENTIAL

Viterbi Coprocessor, Equalizer

### 8.12.1.1.2 Configuration Register 2 for Channel Equalizer

VH\_CONF2\_EQ

Configuration Register 2 for Channel Equalizer

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES ERV ED	EQ RIG HT	PC EQ_1	PC EQ_0	EQ FLA G RD	S COM B	S SEG	RES ALL	RESERVED					EQ ON	RES EQ	HW ENA EQ

Field	Bits	Type	Description
HW_ENA_EQ	0	rw	<b>Enables Clock Hardware</b> 0 Switch off clock for hardware block. 1 Switch on clock for hardware block. <i>Note: Only when <b>HW_ENA_EQ</b> is set can the equalizer related registers be written.</i>
RES_EQ	1	rw	<b>Equalizer Reset</b> 0 No effect. 1 Resets equalization. Reset current timestamp to zero, reset state and transition counter to zero, reset internal equalization flags to default value, reset <b>VH_STATUS_EQ.EQ_BUSY</b> , <b>EQ_RIGHT</b> , <b>EQ_ON</b> , <b>PC_EQ_1</b> , <b>PC_EQ_0</b> , <b>EQ_FLAG_RD</b> , <b>S_COMB</b> , <b>S_SEG</b> , <b>VH_STAT_CNT_E</b> , and <b>VH_CONF_CNT_E</b> . <i>Note: <b>RES_EQ</b> does not reset <b>VH_SC_SOUT</b>.</i>
EQ_ON	2		<b>Starts Viterbi Equalization</b> 0 No effect. 1 Equalization on. This bit is reset by hardware after <b>VH_STATUS_EQ.EQ_BUSY</b> is set in status register.
RES_ALL	8	rw	<b>Resets All Internal Registers</b> Bit is reset within 4 cycles. <i>Note: Register <b>VH_CONF2_EQ</b> will also be reset.</i>
S_SEG	9	rw	<b>Packing Mode for Combined Soft Outputs</b> 1 Packing mode for segments adjacent to the training sequence (except of synch burst). 0 Packing mode for all other segments.



**CONFIDENTIAL**

**Viterbi Coprocessor, Equalizer**

Field	Bits	Type	Description
<b>S_COMB</b>	10	rw	<b>Combining of Soft-Output and Hard Output Values</b> 0 Combining and packing off. 1 Combining and packing on.
<b>EQ_FLAG_RD</b>	11	rw	<b>Direction Flag for Access to ext1</b> 0 Write access to ext1. 1 Read access to ext1.
<b>PC_EQ_0</b>	12	rw	<b>Packing mode 0 for RAM1, RAMW1, RAMW2</b> 0 <b>Write access:</b> Incoming 16 bit words are stored at incremented RAM addresses. <b>Read access:</b> The transfer refers to adjacent RAM addresses. 1 <b>Write access:</b> 2 incoming 16 bit words are packed into one 32 bit word. <b>Read access:</b> The two half-words of the 32 bit word are transferred successively.
<b>PC_EQ_1</b>	13	rw	<b>Packing Mode 1 for RAM2</b> 1 <b>Write access:</b> 2 incoming 8 bit words are packed into a 16 bit word. <b>Read access:</b> The two 8 bit words of the 16 bit word are transferred successively, the 8 MSB of ext1 are sign-extended by bit 7 of the 8 bit word to be transferred. 0 <b>Write access:</b> Incoming 16 bit words are stored at incremented RAM addresses. <b>Read access:</b> The transfer refers to adjacent RAM addresses.
<b>EQ_RIGHT</b>	14	rw	<b>Equalization Select</b> 0: Left halfslot. 1: Right halfslot.
<b>RESERVED</b>	3:7, 15	r	<b>Reserved;</b> reading returns 0; writing to these bit positions has no effect.

**CONFIDENTIAL**

**Viterbi Coprocessor, Equalizer**

### 8.12.1.1.3 Status

**VH\_STATUS\_EQ**

**Status register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>EQ BUSY</b>	<b>RESERVED</b>														

Field	Bits	Type	Description
<b>EQ_BUSY</b>	15	r	<b>Equalization Is Running</b> Set by the hardware peripheral, reset by hardware when the number of timestamps set in <b>VH_CONF_CNT_E</b> has been processed. The falling edge of <b>EQ_BUSY</b> creates an interrupt at the TEAKLite setting bit <b>INT_FINTA0.EQ</b> . <b>EQ_BUSY</b> can also be reset by <b>VH_CONF2_EQ.RES_EQ</b> or <b>VH_CONF2_EQ.RES_ALL</b> .
<b>RESERVED</b>	14:0	r	<b>Reserved</b> ; reading returns 0; writing to these bit positions has no effect.

### 8.12.1.1.4 Equalization Bit Count Configuration

**VH\_CONF\_CNT\_E**

**Equalization Bit Count Configuration**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>										<b>C_EQ</b>					

Field	Bits	Type	Description
<b>C_EQ</b>	5:0	rw	<b>Number of Symbols to be Equalized</b> <i>Note:</i>
<b>RESERVED</b>	15:6	r	<b>Reserved</b> ; reading returns 0; writing to these bit positions has no effect.

CONFIDENTIAL

Viterbi Coprocessor, Equalizer

### 8.12.1.1.5 Equalization Bit Count Status

VH\_STAT\_CNT\_E

Equalization Bit Count Status

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED										S_EQ					

Field	Bits	Type	Description
S_EQ	5:0	r	<b>Number of Symbols Already Equalized</b> Set by hardware, reset by <a href="#">VH_CONF2_EQ.RES_EQ</a> or <a href="#">VH_CONF2_EQ.RES_ALL</a> .
RESERVED	15:6	r	<b>Reserved</b> ; reading returns 0; writing to these bit positions has no effect.

**CONFIDENTIAL**

**Viterbi Coprocessor, Equalizer**

### 8.12.1.1.6 Scaling Value for Soft-Outputs

**VH\_SC\_SOUT**

**Scaling Value for Soft-Outputs**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>SC 15</b>	<b>SC 14</b>	<b>SC 13</b>	<b>SC12_9</b>				<b>SC8</b>	<b>SC7_0</b>							

Field	Bits	Type	Description
<b>SC7_0</b>	7:0	rw	<b>Shift Value for Scaling of Soft Outputs</b> <ul style="list-style-type: none"> <li>• If all bits are set, a left shift of the soft outputs is performed.</li> <li>• If more than one bit (except the upper case) is set, the hardware refers to the MSB and performs a right shift.</li> <li>• If the bits are set to zero, no shift is performed.</li> </ul>
<b>SC8</b>	8	rw	<b>Enable for Second Shifting Stage</b>
<b>SC12_9</b>	12:9	rw	<b>Saturation Selection</b> SC_12 = 1: Saturation of soft outputs to [-127,+127]. SC_11 = 1: Saturation of soft outputs to [-63,+63]. SC_10 = 1: Saturation of soft outputs to [-31,+31]. SC_9 = 1: Saturation of soft outputs to [-15,+15]. If more than one bit is set a saturation to [-127,+127] is performed. If the bits are set to zero a saturation to [-127,+127] is performed.
<b>SC13</b>	13	rw	Control bits to increase scaling accuracy. Enables adding stages for adding fractions of the original value: approximation of the scaling division operation by shift & add stages.
<b>SC14</b>	14		
<b>SC15</b>	15		

**CONFIDENTIAL**

**Viterbi Coprocessor, Equalizer**

## VH\_SQUAL

**Output Register for RX\_QUAL Histogram**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ															

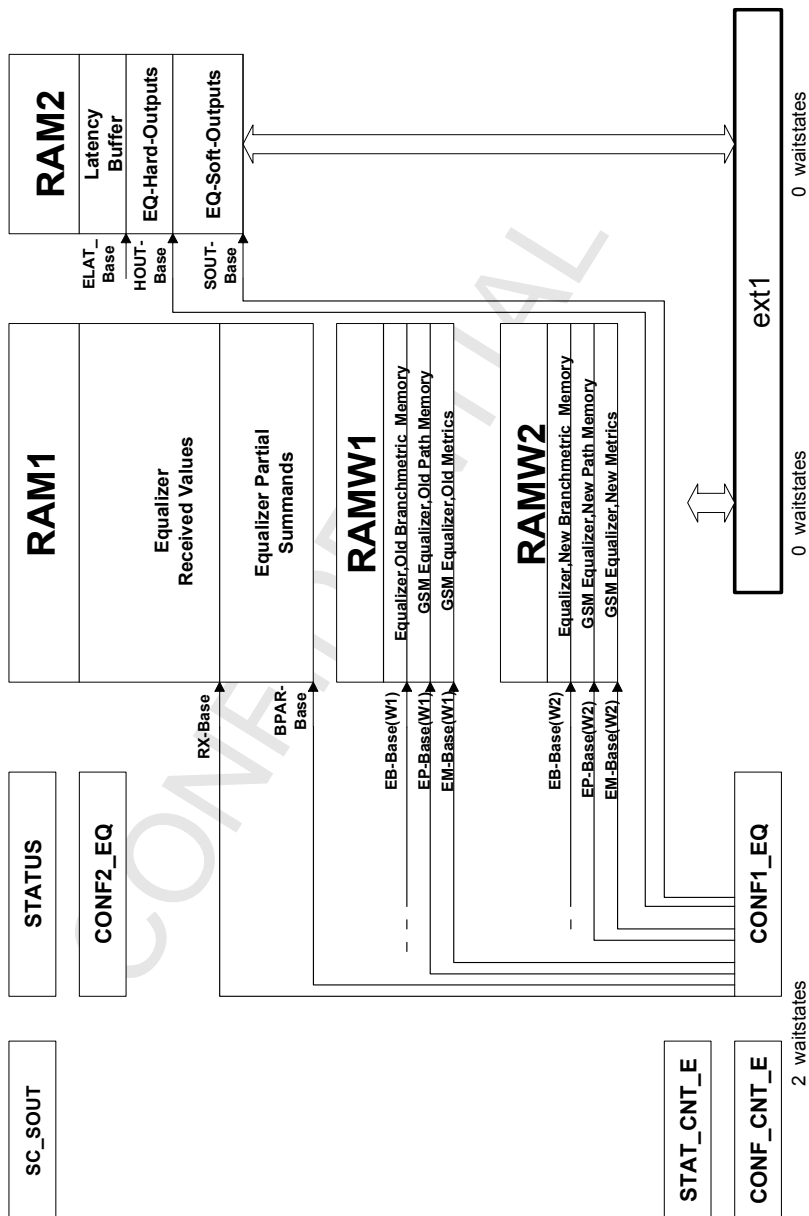
Field	Bits	Type	Description
<b>SQ</b>	15:0	r	<b>Output from the 16 Registers of the Histogram FIFO.</b> The Fifo pointer is initialized to histogram value number 15 and is decremented at each read access. After 16 read commands the FIFO pointer returns to Hist15.  <i>Note: Between successive read accesses one wait state is required.</i>

CONFIDENTIAL

Viterbi Coprocessor, Equalizer

Figure 8-60 TEAKLite interface and Hardware Internal Memories

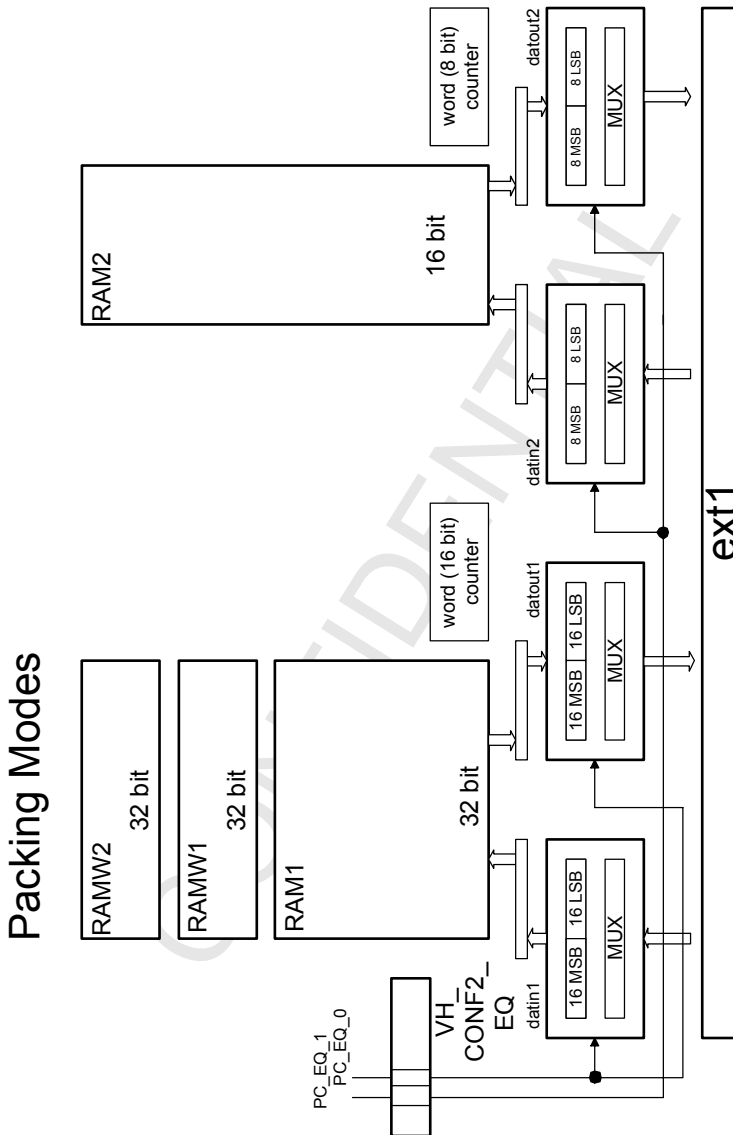
# DSP Interface and Internal Memory



CONFIDENTIAL

Viterbi Coprocessor, Equalizer

Figure 8-61 Packing Modes



All RAMs are organized in 32 bit word width, except from RAM2, which only supports 16 bits.

**CONFIDENTIAL**

**Viterbi Coprocessor, Equalizer**

**VH\_CONF2\_EQ.PC\_EQ\_0**, **VH\_CONF2\_EQ.PC\_EQ\_1** are defined which control the access to RAM1, RAMW1, RAMW2 (**PC\_EQ\_0**) and RAM\_2 (**PC\_EQ\_1**). The different modes are required to fit different word formats for input and output values to the available RAM format (**Figure 8-61**).

All formats referring to RAM1, RAMW1 and RAMW2 are organized in 16 bits, while formats referring to RAM2 refer to 16 bits as well as to 8 bits (soft inputs, soft outputs) which have to be adapted to the 16 bits of RAM2.

#### **RAM1, RAMW1, RAMW2: Register VH\_CONF2\_EQ bit PC\_EQ\_0**

- **PC\_EQ\_0 = 1:**

As the TEAKLite transfers 16-bit words the adaptation to 32-bit word width has to be performed within the HW peripheral.

The incoming data from TEAKLite will be stored temporarily and is combined with the following 16-bit word before the entire 32-bit word is written to the RAM1, RAMW1 or RAMW2. The TEAKLite transfers the least significant 16-bit word first.

In the opposite direction from hardware to TEAKLite first the least significant 16-bit word is transferred first, followed by the most significant 16-bit word. The second 16-bit word is temporarily stored within the hardware peripheral until the read access of the TEAKLite is performed.

In case of complex values to be transferred the least significant 16-bit word is the real component (I) and the most significant 16-bit word form the imaginary component (Q). For switching between the least and most significant 16-bit word of one 32-bit word in RAM1, RAMW1 or RAMW2 two internal signals, RAM32\_WR\_FLAG\_EQ and RAM32\_RD\_FLAG\_EQ, have been introduced. These signals will be reset if one of the bits **RES\_x\_BAS** (x = RX, BPAR, EMR, EML, EPR, EPL, EB) is set to ensure proper initial conditions.

- **PC\_EQ\_0 = 0:**

In this mode 16-bit words are transferred directly to or from successive RAM addresses without introducing any packing mechanism.

#### **RAM2: Register VH\_CONF2\_EQ bit PC\_EQ\_1**

The values that have to be transferred between TEAKLite and the HW peripheral are organized in 16-bit or 8-bit words.

- **PC\_EQ\_1 = 1:**

From each incoming 16-bit word the 8 LSB are combined in a 16-bit word (2x8 bit) before storing the value to the RAM2. At a read access a 16-bit word is decomposed into 2 words of 8-bit and transferred to the TEAKLite at two successive ext1 accesses. In both cases for write and read the least significant 8-bit word is transferred first, followed by the most significant 8-bit word.

As for RAM1, RAMW1 and RAMW2 there are also two internal flags (RAM16\_RD\_FLAG\_EQ, RAM16\_WR\_FLAG\_EQ) for RAM2, which select the 8-bit sub-words of a 16-bit word currently being transferred. Both flags are reset if one of



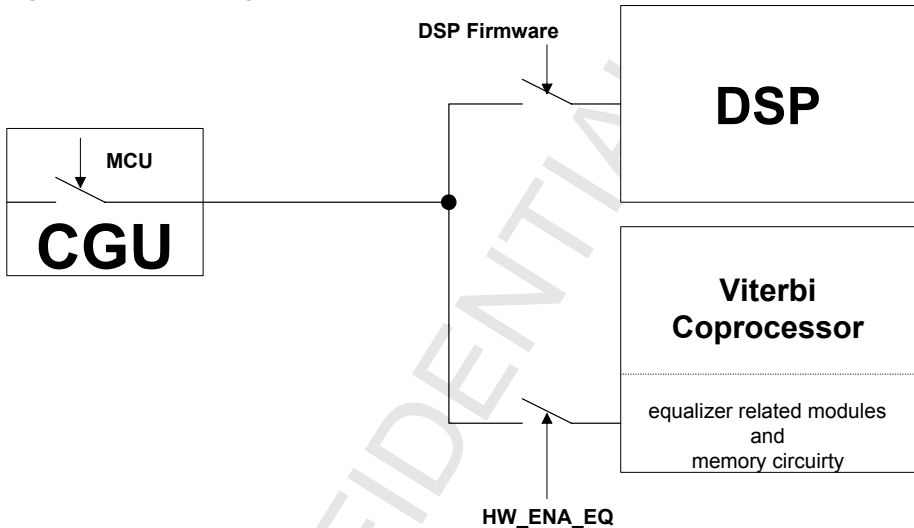
CONFIDENTIAL

Viterbi Coprocessor, Equalizer

the bits **VH\_CONF1\_EQ.RES\_HOUT\_BASE**, **VH\_CONF1\_EQ.RES\_SOUT\_BASE** or **VH\_CONF1\_EQ.RES\_ELAT\_BASE** is set.

- **PC\_EQ\_1 = 0:**  
16-bit words are transferred between TEAKLite and hardware without introducing any packing procedure. 16-bit values are written to or read from successive addresses.

Figure 8-62 Clocking Scheme



The register interface to the DSP is supplied by the DSP clock,

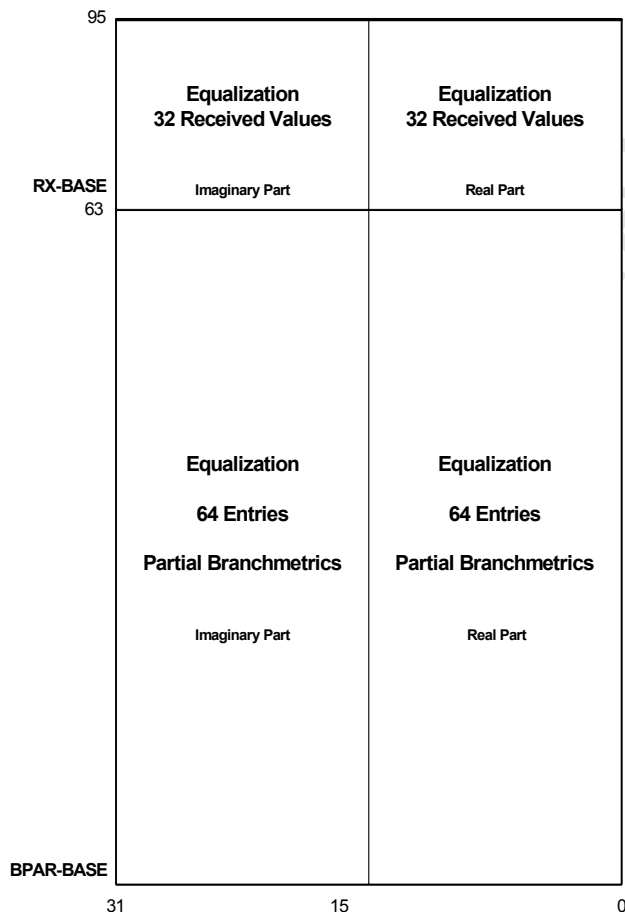
**HW\_ENA\_EQ** switch on the hardware internal memory circuitry

CGU switches on the total hardware circuitry,  
a status bit informs the CGU about the clock in the hardware accelerator

**CONFIDENTIAL**

**Viterbi Coprocessor, Equalizer**

**Figure 8-63 RAM 1 Memory Partitioning**  
**Memory Mapping: RAM1**  
**Equalization**

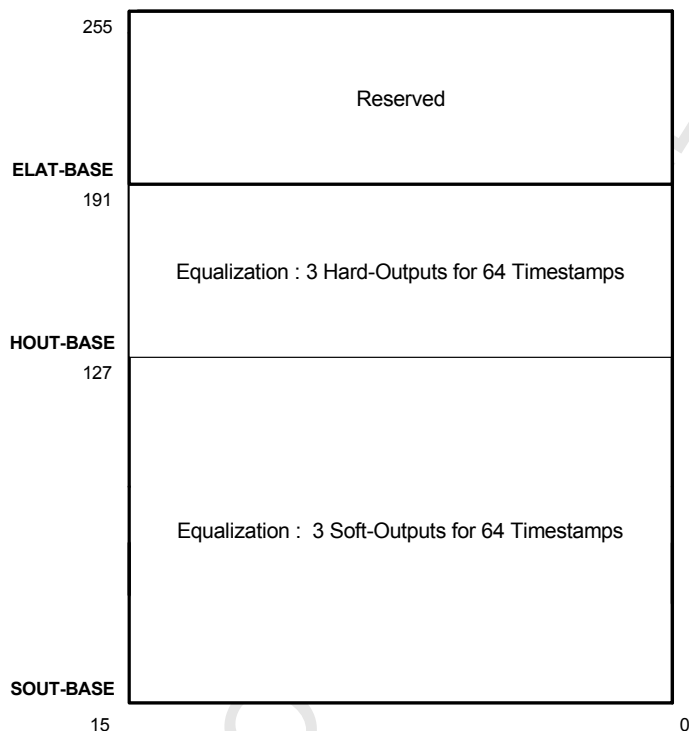


**CONFIDENTIAL**

**Viterbi Coprocessor, Equalizer**

**Figure 8-64 RAM2 Memory Partitioning**  
**Memory Mapping: RAM 2**

**Equalization**



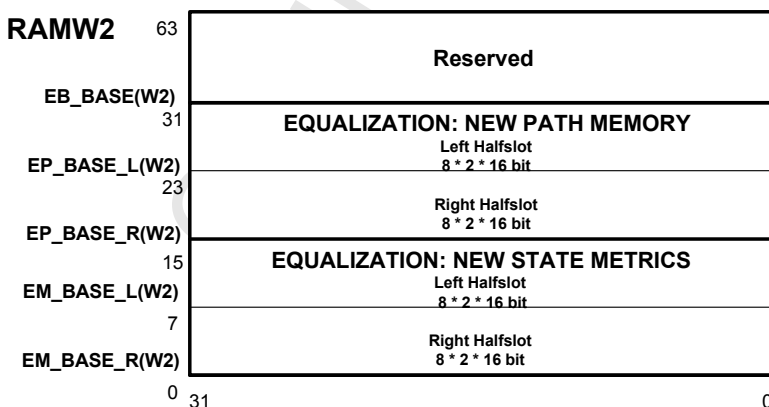
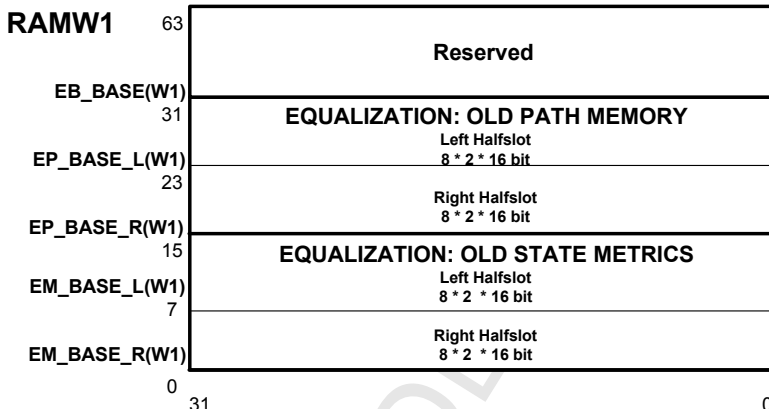
CONFIDENTIAL

Viterbi Coprocessor, Equalizer

Figure 8-65 RAMW1/RAMW2 Memory Partitioning  
Memory Mapping: RAMW1/RAMW2

## Equalization

### WORKING RAMs



## 8.12.2 Flow of Equalization Process

**Figure 8-66** shows the initialization of the hardware accelerator for equalization. The required information are the initial state vectors depending on the training sequence. The start values for the state vectors are initialized in RAMW1. Additionally the start values for the metric values have to be initialized in the working RAMW1 for equalization.

### Equalization

**Figure 8-67** shows the flow of an equalization task. Especially the required access to the HW internal registers and memory blocks is explained. To perform an equalization the following steps are required:

1. Enable the hardware accelerator clock by setting bit **VH\_CONF2\_EQ.HW\_ENA\_EQ**. Before equalization can be started the bit **VH\_CONF2\_EQ.RES\_EQ** must be set. This causes a reset of all internal registers and counters.
2. **VH\_CONF2\_EQ.EQ\_RIGHT** selects between equalization of the left or the right half slot.
3. The register **VH\_CONF\_CNT\_E** determines the number of symbols to be equalized and must also be set. During the equalization the number of already equalized symbols is updated in register **VH\_STAT\_CNT\_E**.
4. The transfer of received values is prepared by setting bit **VH\_CONF1\_EQ.RES\_RX\_BASE**. This will reset the RAM1 pointer to its base value **RX\_BASE** and therefore enable the access to the targeted RAM1 area
5. The dedicated RAM1 area is filled with the received values using successive write operations to ext1 (RAM1). The RAM1 pointer will be post-incremented after each write operation.
6. To transfer the partial summands the previous step has to be performed after setting bit **VH\_CONF1\_EQ.RES\_BPAR\_BASE**. The successive write access to ext1 (RAM1) will then fill the RAM1 area dedicated to the partial sums.
7. The equalization procedure is started as soon as bit **VH\_CONF2\_EQ.EQ\_ON** has been set. The equalization task takes approximately  $(n+1) \cdot 26 \cdot 8$  cycles for an equalization partition of  $n$  symbols (interaction and data exchange with firmware not regarded). After this time the bit **VH\_STATUS\_EQ.EQ\_BUSY** is reset. This causes an interrupt at the TEAKLite by setting bit EQ in **INT\_FINTA0**. Now the TEAKLite can transfer the equalizer hard and soft outputs from the reserved RAM2 area to its internal memory.
8. To prepare the transfers to TEAKLite the following bits must be set: **VH\_CONF1\_EQ.RES\_HOUT\_BASE** or **VH\_CONF1\_EQ.RES\_SOUT\_BASE** and **VH\_CONF2\_EQ.EQ\_FLAG\_RD**.
9. Again, the transfer is performed by successive reads from ext1.

After this procedure the HW accelerator can be disabled by **VH\_CONF2\_EQ.HW\_ENA\_EQ** = 0, which switches off the clock for power saving reasons. Alternatively, the equalization task can be continued for the following partitions.

**CONFIDENTIAL**

**Viterbi Coprocessor, Equalizer**

*Note: Make sure that only the required bit entries in **VH\_CONF1\_EQ** and **VH\_CONF2\_EQ** are set and that all unused entries are cleared.*

**EQ Summary**

Use the following procedure for equalization:

1. Set **VH\_CONF2\_EQ.HW\_ENA\_EQ** = 1 (Must be set before other bits can be set).
2. Set **VH\_CONF2\_EQ.RES\_EQ** = 1 (Only to be set at slot begin, not during equalization of successive segments).
3. Set **VH\_CONF2\_EQ.HW\_ENA\_EQ** = 1, **VH\_CONF2\_EQ.PC\_EQ\_0** = 0 or 1.

*Note: The **VH\_CONF2\_EQ** register must be written before the **VH\_CONF1\_EQ** register is set. Set and reset commands of single bits are not allowed to follow each other directly.*

**Initialization Procedure**

1. Set **VH\_CONF1\_EQ.RES\_EP(L/R)\_BASE**, **VH\_CONF1\_EQ.RES\_RW1\_RW2**.
2. Transfer of start state vectors to RAMW1: **VH\_CONF2\_EQ.PC\_EQ\_0** = 0.
3. Set **VH\_CONF1\_EQ.RES\_EM(L/R)\_BASE**, **VH\_CONF1\_EQ.RES\_RW1\_RW2**.
4. Transfer of start metric values to RAMW2: **VH\_CONF2\_EQ.PC\_EQ\_0** = 1.

**Processing Procedure**

1. Set **VH\_CONF2\_EQ.HW\_ENA\_EQ** = 1, **VH\_CONF2\_EQ.PC\_EQ\_0** = 1.

*Note: The **VH\_CONF2\_EQ** register must be written before the **VH\_CONF1\_EQ** register is set. Set and reset commands of single bits are not allowed to follow each other directly.*

2. Load **VH\_CONF\_CNT\_E**
3. Set **VH\_CONF1\_EQ.RES\_BPAR\_BASE**.
4. Carry out transfer of partial sums to HW peripheral.
5. Set **VH\_CONF1\_EQ.RES\_RX\_BASE**.
6. Carry out transfer of received values to HW peripheral.
7. Set **VH\_CONF2\_EQ.HW\_ENA\_EQ** = 1, **VH\_CONF2\_EQ.EQ\_RIGHT** = 1 or 0, **VH\_CONF2\_EQ.EQ\_EDGE** = 1, **VH\_CONF2\_EQ.EQ\_ON** = 1.

Hardware equalization processing is now active.

CONFIDENTIAL

Viterbi Coprocessor, Equalizer

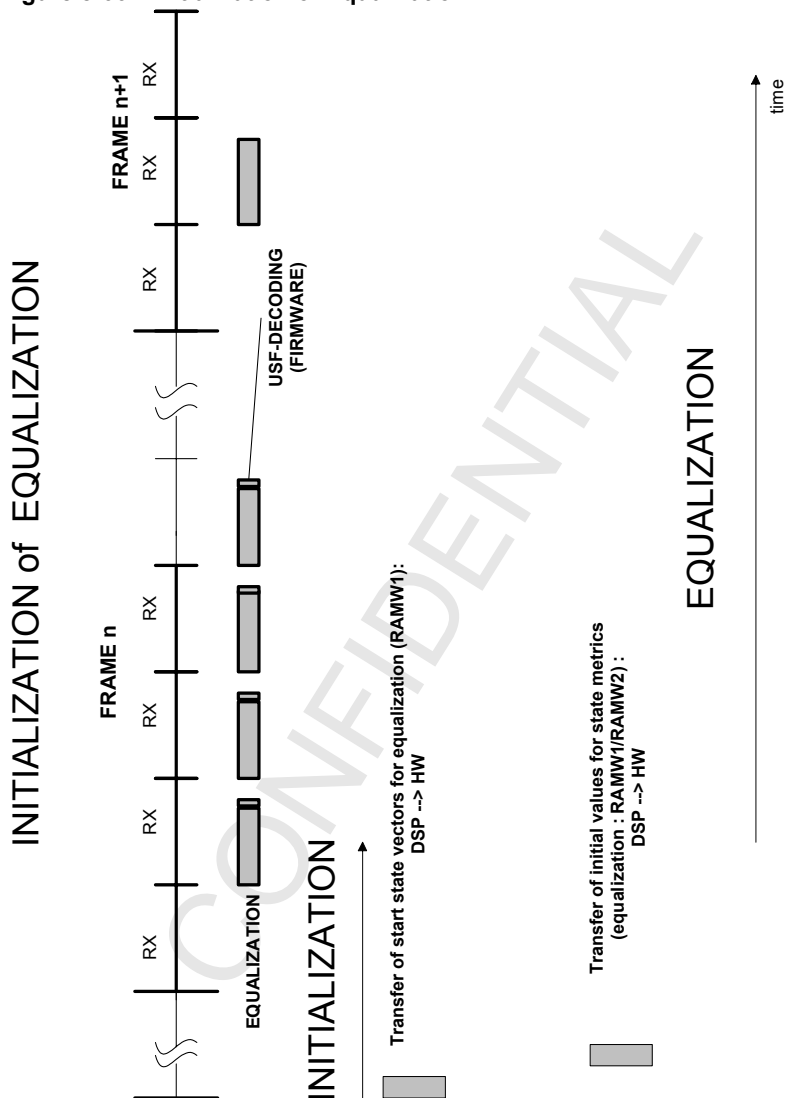
At an EQ interrupt proceed as follows:

1. Set **VH\_CONF2\_EQ.PC\_EQ\_1** = 0, **VH\_STATUS\_EQ.EQ\_FLAG\_RD** = 1, **VH\_CONF2\_EQ.HW\_ENA\_EQ** = 1.
2. Set **VH\_CONF1\_EQ.RES\_HOUT\_BASE** and transfer hard decision values to TEAKLite.
3. Set **VH\_CONF2\_EQ.PC\_EQ\_1** = 1, **VH\_STATUS\_EQ.EQ\_FLAG\_RD** = 1, **VH\_CONF2\_EQ.HW\_ENA\_EQ** = 1
4. Set **VH\_CONF1\_EQ.RES\_SOUT\_BASE** and transfer soft decision values to TEAKLite.

*Note: If **VH\_CONF\_CNT\_E** is set to zero and **VH\_STATUS\_EQ.EQ\_ON** is set, the equalizer will always process one symbol.*

CONFIDENTIAL

Figure 8-66 Initialization of Equalization



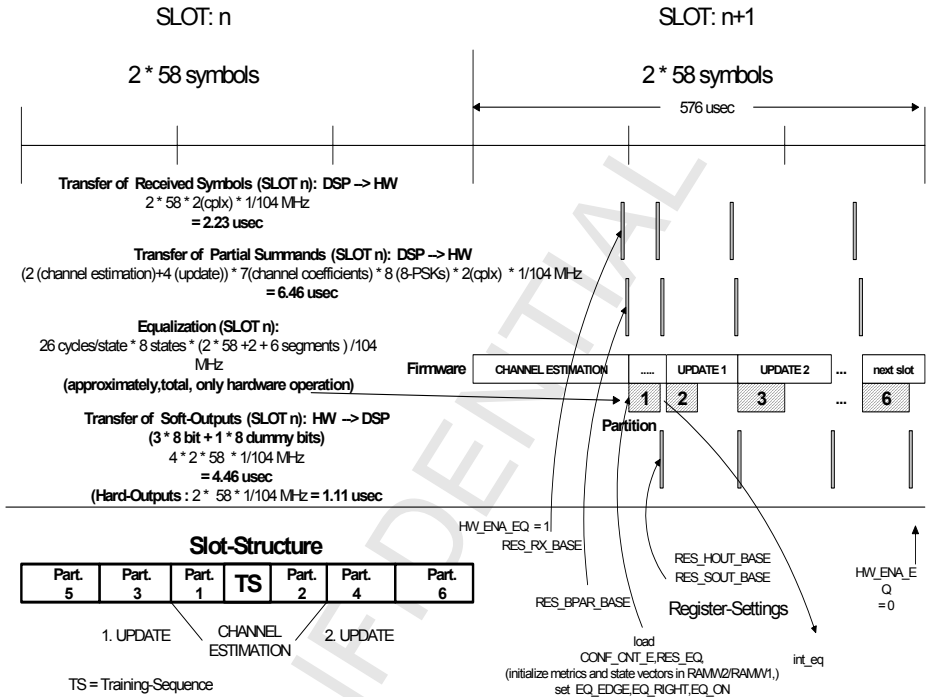


**CONFIDENTIAL**

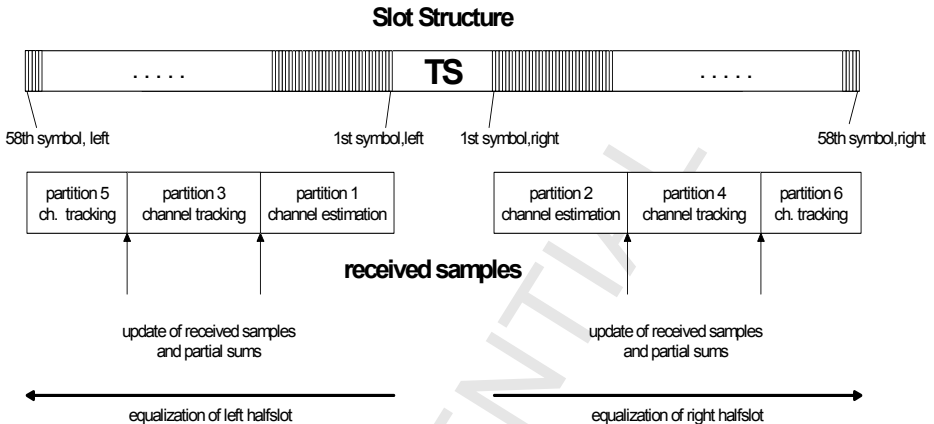
**Viterbi Coprocessor, Equalizer**

**Figure 8-67 Flow of Equalization**

## FLOW of EDGE EQUALIZATION



# EQUALIZATION FLOW



During the time interval the hardware accelerator is active (**VH\_STATUS\_EQ.EQ\_ON**, **VH\_STATUS\_EQ.EQ\_BUSY**) the TEAKLite has to be used to perform the updates of the channel estimation (channel tracking). In case the channel estimation update at the TEAKLite takes more cycles than the equalization task of the HW accelerator the equalizer can be disabled and started again when the channel estimation update has been finished.

In **Figure 8-67** the process of equalization is separated into 2\*3 intervals (The figure only provides an example. At least two segments for each halfslot are mandatory). The reason for this approach can be understood when regarding to the slot structure. With the channel estimation based on the training sequence the two data partitions 1 and 2 can be equalized. At the end of the equalization procedure of partition 1 the resulting hard and soft outputs are used for the first update of the channel estimation. For this reason the hard and soft output values have already to be transferred to the TEAKLite. During the calculation of the first update within the TEAKLite the equalization of the second partition takes place in hardware. The results of this equalization procedure are used to enable the second update of the channel estimation. With the transfer of the partial summands based on the first update of the channel estimation and the pre-filtered received samples (RAM1) the third data partition can be equalized. Meanwhile, the calculation of the second update of the channel estimation takes place within the TEAKLite. The fourth data partition can be equalized as soon as the second update of the channel estimation and the transfer of partial summands and pre-filtered received values have been finished. The other two segments are treated in the same manner.

The approach of simultaneously processing the channel estimation update within the TEAKLite and the equalization within the hardware on the one hand and the necessity

**CONFIDENTIAL**

**Viterbi Coprocessor, Equalizer**

of different updates based on already equalized symbols on the other hand has the consequence that between switching from one data partition to another data partition different memory areas for the current state vectors and for the current metric values for the left and right half slot have to be provided.

For the data partitions 1 and 2 the initial state vectors are based on the training sequence. The last state vector of the training sequence is the starting state vector of the equalization task. This state is preferred compared to all other states. Therefore the initial state metric of this state is set to a dominating value. All other state metrics are set to a default value of minor significance.

For the data partition 3 (4) the initial state vectors are the ending state vectors of data partition 1 (2) and the initial metrics are the final metrics of data partition 1 (2).

Internal flags support the continuous flow of equalization of successive segments of the right or left halfburst. When stepping from a segment of one halfburst to a segment of the other halfburst these flags are preserved and guarantee a continuous processing when switching back to the next segment of the former halfburst. These flags can be reset by **VH\_STATUS\_EQ.RES\_ALL** and **VH\_STATUS\_EQ.RES\_EQ**.

The state metrics in combination with the state vectors are kept within the working RAMs RAMW1/RAMW2. As these RAMs are directly accessible by the TEAKLite an additional debugging feature is provided.

The equalization process of one halfburst within the hardware accelerator delivers a sequence of hard and soft output values which have to be compared in the firmware or in a special hardware block. If the sign of the hard output value and the related soft output value differ the soft output value adopts the sign of the hard output value but with a magnitude value of one. As the hard output values are delivered delayed in time by the hardware accelerator the pairs of hard and soft output values to be combined have to exhibit this delay.

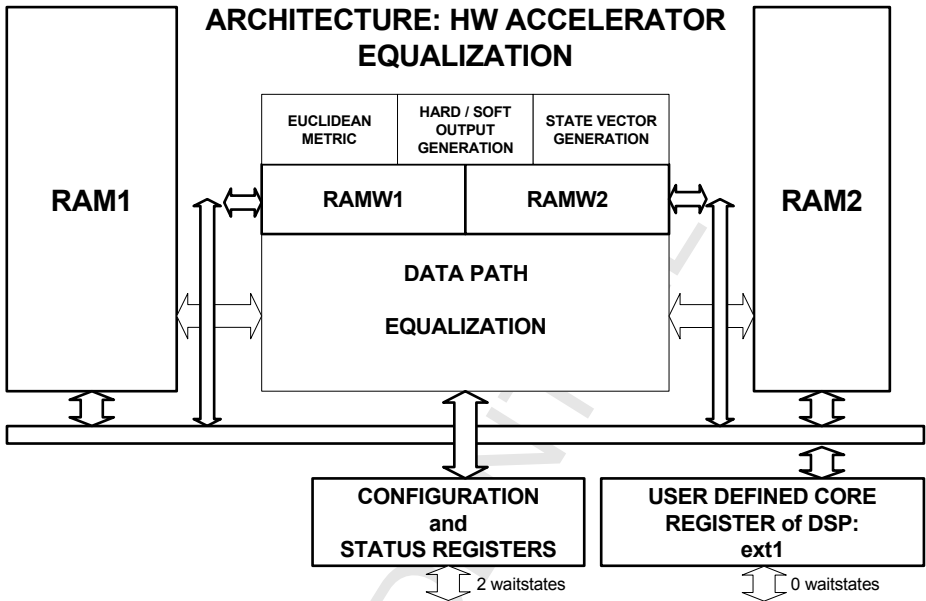
### **8.12.3 Architecture**

An overview of the equalizer architecture is shown in **Figure 8-68**.

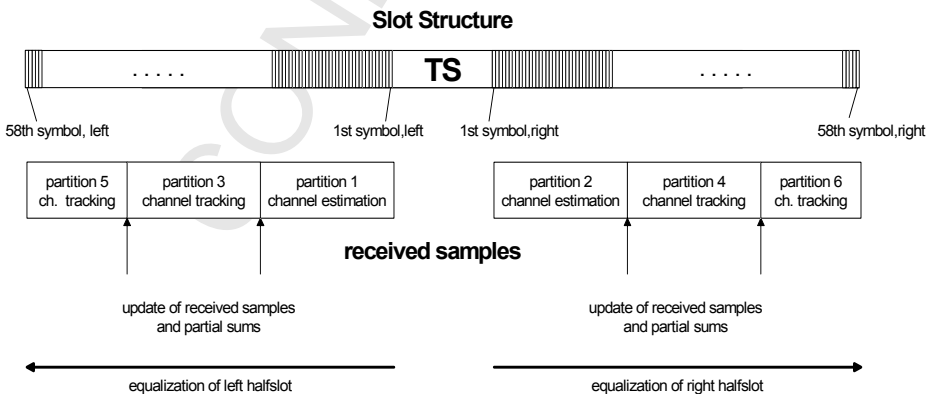
CONFIDENTIAL

Viterbi Coprocessor, Equalizer

Figure 8-68 Global Architecture



## EQUALIZATION FLOW



**CONFIDENTIAL**

**Shared Memory**

### **8.13 Shared Memory**

Refer to the X-Bus [Shared Memory \(on Page 621\)](#).

### **8.14 Multicore Synchronization**

Refer to the X-Bus [Multicore Synchronization \(on Page 625\)](#).

CONFIDENTIAL

CONFIDENTIAL

**CONFIDENTIAL**

**OCEM/SEIB**

## 8.15 OCEM/SEIB

History	
Design Spec.	Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.06	

### System Integration

- Supply domain: VDD\_DSP
- Chip internal interfaces:
  - Clock domain: gclk\_dsp\_per (SEIB), gclk\_dsp (OCEM)
  - Bus domain: TEAKLite Z-Bus, Debug Interface to TEAKLite (OCEM)
- Interrupt sources: none
- Monitor Pins: Refer to [Section 11.8.10 Internal Signal Monitoring \(on Page 1209\)](#)

#### 8.15.1 Functional Overview

The DSP Debug concept is based on the On-chip Emulation Module (OCEM).

The OCEM uses the DSP core TRAP mechanism to implement its emulation functions. The emulation functions include two main tasks:

- Break point generation
- Program flow tracing.

A break point can be generated at predefined conditions that are programmed into the OCEM registers. Once a condition is met the OCEM activates the TRAP mechanism causing the core to suspend any action and jump to the service routine.

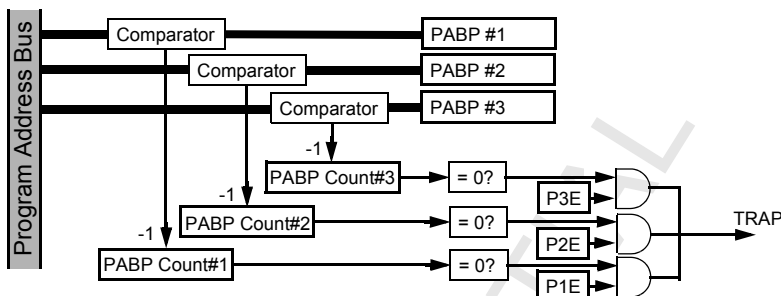
Program flow tracing includes dynamic recording of program addresses. Those addresses provide a full program flow graph of instruction being executed.

#### 8.15.2 Program Address Break Point Operation

When the OCEM senses a match between the program address from the core and one of the addresses written in the PABP registers and a fetch is performed, the counter corresponding to that address is decreased by 1, provided that it was not a dummy fetch. If the counter value is 0, then a TRAP request is issued. If the user wishes to have the break point only in the  $n^{\text{th}}$  occurrence the counter has to be set to  $n$  (see [Figure 8-69](#)).

The program address break point can be disabled by writing 0 to the **MODE0.P1E** to **MODE0.P3E** bits. The program address break point source is indicated in bits **STATUS0.P1** to **STATUS0.P3**.

**Figure 8-69 OCEM Program Address Break Point Block Diagram**



### 8.15.3 Pipe Break Point Operation

The OCEM allows the user to insert break points upon pipe breaking events such as branch, call, interrupt or looping to the beginning of block repeat. The user can select which event of the following list will cause a break point:

- BR, BRR, CALL, CALLR, CALLA, RET, RETI, RETD, RETID, RETS, Mov to PC
- Interrupt service start
- Transition from last to first address in BKREP loop.

*Note: The first time the first instruction of the block is fetched there is no break point since this is a sequential execution.*

### 8.15.4 Data Address Break Point Operation

The data address break point is initiated upon a match between the OCEM data address register (**DABP**) and the DSP core data address bus.

The data address mask register allows expansion of the data address break point to an address space instead of a one single address. The address space length is always a power of two and it is specified by writing a 1 to the mask register bits. The comparison is performed bit-wise between the current data address and the data address register. If the corresponding bit in the mask register is set to zero the comparison will not be carried out for this bit.

The data address break point can be enabled for read or write transactions. It is done by setting the bits **MODE0.DARE** (for breaking on read transactions) or **MODE0.DAWE** (for breaking on write transactions). Detection of data address break point is indicated in **STATUS0.DABP** bit.



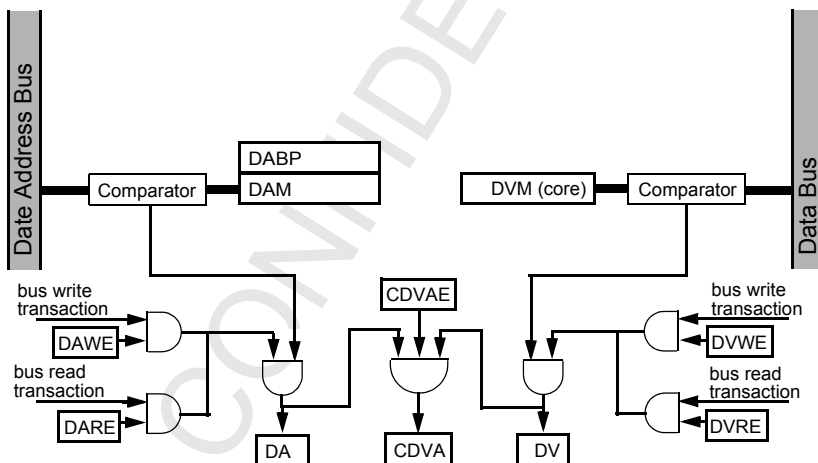
### 8.15.5 Data Value Break Point Operation

Data value break point is initiated upon detection of a data match between the DSP core internal data bus and the content of the internal DSP DVM (Data Value Match) register. Note that the data value breakpoint is initiated only at data matches during MEMORY transactions. There is an option to enable the data value break point for read or write transactions through the bits **MODE0.DVRE** (for read) and **MODE0.DVWE** (for write transactions) within the OCEM mode register. Detection of a data value break point is indicated in the **STATUS0.DV** bit (see [Figure 8-70](#)).

There is an option to detect a combined data address and value condition. It is enabled by the bit **MODE0.CDVAE**. This break point type is enabled by the **STATUS0.CDVA** bit. Note that in order to break on the combined condition, the **CDVAE** bit must be set together with **MODE0.DARE** and **MODE0.DVRE** bits for read transaction or **MODE0.DAWE** and **MODE0.DVWE** for write transactions.

*Note: The data address as well as data value break point are being initiated AFTER the transaction is completed (as opposed to program address break point where the break point occurs PRIOR to the instruction executed).*

**Figure 8-70 OCEM Data Address/Data Value Operation Block Diagram**



### 8.15.6 External Registers Break Point

An external register break point can be enabled for both read (**MODE0.EXTRE**) or write (**MODE0.EXTWE**). An indication for this type of break point is in **STATUS0.EREG** bit.

*Note: A break point on external registers transaction is activated AFTER the transaction is completed (similar to data address and data value case).*

### **8.15.7 Clarifications for the Data Value/Address Break Point**

Since the data value is detected at the operand fetch state, two more instructions are already in the pipe and issued for decode and executed before the interrupt is served. This means that for data value break points there is a latency of two instructions and the user will see the status of the DSP only two instructions after the event has happened.

For data address break point the latency could have been reduced to one instruction. However, to enable combined **DABP (on Page 606)** and DVBP (Data Value Break Point) the TRAP is delayed and the latency is therefore two instructions.

### **8.15.8 Illegal Break Point**

The OCEM block contains a capability to protect the reserved mail box area and the OCEM registers from illegal access. An access of the mailbox space is only allowed through the break point handler (that is, the routine that is being executed as a result of TRAP interrupt). Any attempt to access the mailbox space as well as the OCEM space from outside the break point routine causes a break point and sets the **STATUS0.ILL**. The illegal break point can be disabled through the bit **MODE0.ILLE** bit.

Any attempt to write to the **MODE0.ILLE** bit and into **STATUS0.ILL** bit is automatically disabled if it is not done through the break point handler.

The illegal break point mechanism is also used to protect the break point handler routine from illegal access (for example, when the user routine from some erroneous state jumps into break point handler). If the handler tries to access the mail box area in such a case it will activate the illegal break point.

### **8.15.9 DSP Monitor Program**

The DSP monitor program (TEAKliteMON) is a DSP program routine which automatically starts working upon receipt of a hardware or software TRAP interrupt. The monitor program:

- Dumps all internal registers into the mailbox memory buffer for the debugger to upload it into the PC memory.
- Dumps all OCEM registers into the mailbox memory buffer for the debugger to upload it into the PC memory.
- Checks for not getting a false trap request as a result of a code break point right after a repeated instruction; this is done with special flag.
- Enters a waiting loop by writing an IDLE flag into the mailbox memory buffer. The loop consists of waiting for a debugger command that must be deposited in the mailbox memory. The program knows when a new command is given by polling another memory location in the mailbox which should contain a continue flag.
- Executes the debugger commands.

**CONFIDENTIAL**

**OCEM/SEIB**

- Reloads the OCEM and internal registers from the mailbox memory buffer and returns to the main program by executing a RETI instruction.

The monitor program is stored in the program ROM. The start of the monitor program requires a DSP reset in order to activate the debug boot mode. The debug boot mode is selected when the bit **STATUS0.DBG** is was selected.

### 8.15.10 OCEM Physical Interface

**Table 8-48 Clock, Reset**

Signal Name	Direction	Source/ Destination	Activity	Description
LRSTP	Input	CLK	High	Reset
LERSTP	Input	CLK	High	Early reset
LCLKP	Input	CLK	High	Clock

**Table 8-49 Core-External Interface**

Signal Name	Direction	Source/ Destination	Activity	Description
BBOOTP	Input	MCD Block	High	Boot request.
BDEBUGP	Input	MCD Block	High	Debug request.
BABORTP	Input	MCD Block	High	Abort request.

**Table 8-50 Core Bus Interface**

Signal Name	Direction	Source/ Destination	Activity	Description
TDZAP [15:0]	Input	Core	High	DSP Z address bus
TDRZRMP	Input	Core	High	Z memory read strobe
TDWZRMP	Input	Core	High	Z memory read strobe
TDYAP [15:0]	Input	Core	High	DSP Y address bus
TDRYRMP	Input	Core	High	Y memory read strobe
TDXAP [15:0]	Input	Core	High	DSP X address bus
TDRXRMP	Input	Core	High	X-memory read strobe
TPPAP[15:0]	Input	Core	High	Core program address bus
TPPRP	Input	Core	High	Core program read

**CONFIDENTIAL**

**OCEM/SEIB**

**Table 8-50 Core Bus Interface**

Signal Name	Direction	Source/ Destination	Activity	Description
TPPWP	Input	Core	High	Core program write
PRPAGEP [3:0]	Input	Core	High	Program page number
PAGE_UPDATE	Input	Core	High	Indication for update of program page number
BSEAP [3:0]	Input	Core	High	OCEM registers address bus
BSBP [15:0]	Input	Core	high	DSP Z-Bus input
SBEDP [15:0]	Output	Core	high	DSP Z-Bus output
BSERDP	Input	Core	high	OCEM registers read strobe
BSEWRP	Input	Core	high	OCEM registers write strobe
BMBOXTRANSP	Input	BIU	high	Mailbox address space indication
BURSTP	Input	I/O	high	User reset indication
TPDUMMYP	Input	Core	high	Indicator that the instruction which should be at that time in operand-fetch phase is aborted and the instruction fetch was dummy
TODTVMP	Input	Core	high	Data value match indicator
TIXRDP	Input	Core	high	External registers read strobe
TIXWRP	Input	Core	high	External registers write strobe
TPTRAPAP	Input	Core	high	Trap routine execution indication
TPSFTP	Input	Core	high	Software trap indication
BABORTP	Input	I/O	high	Abort request
BDEBUGP	Input	I/O	high	Debug mode request
BBOOTP	Input	I/O	high	Boot request
ETRAPREQP	Output	Core	high	Trap request
TPIACKN	Input	Core	low	Core interrupt acknowledge

**CONFIDENTIAL**

**OCEM/SEIB**

**Table 8-50 Core Bus Interface**

Signal Name	Direction	Source/ Destination	Activity	Description
TPSTATUSP[3:0]	Input	Core	high	Core status pin for pipeline break point logic
TDWWBP	Input	Core	high	Indication for write to buffer or to Z space

### 8.15.11 OCEM Register Interface

The OCEM register addresses are in

**Table 8-51 OCEM Registers**

Register	Reset value	Description
<b>PFT</b>	XXXX <sub>H</sub>	Program flow trace register
<b>PABP1</b>	XXXX <sub>H</sub>	Program address break point 1
<b>PABP2</b>	XXXX <sub>H</sub>	Program address break point 2
<b>PABP3</b>	XXXX <sub>H</sub>	Program address break point 3
Reserved	XXXX <sub>H</sub>	Reserved
<b>SIGNATURE</b>	3000 <sub>H</sub>	OCEM programming model signature
<b>PFTPG</b>	XXXX <sub>H</sub>	Trace page register
<b>PACNT1</b>	XXXX <sub>H</sub>	Program address break point counter 1
<b>PACNT2</b>	XXXX <sub>H</sub>	Program address break point counter 2
<b>PACNT3</b>	XXXX <sub>H</sub>	Program address break point counter 3
<b>DAMASK</b>	XXXX <sub>H</sub>	Data address mask register
<b>DABP</b>	XXXX <sub>H</sub>	Data address break point
Reserved	XXXX <sub>H</sub>	Reserved
<b>MODE0</b>	0000 <sub>H</sub>	OCEM mode register 0
<b>STATUS1</b>	1000 0000 0000 000X <sub>B</sub>	OCEM status register 1
<b>STATUS0</b>	0000 <sub>H</sub>	OCEM status register 0

## PFT

### Program Flow Trace Register

Reset value: XXXX<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PFT]															

Field	Bits	Type	Description
PFT	15:0	rw	Reflects the last stage of the trace buffer FIFO. Refer to register <a href="#">PFTPG</a> .

## PABP1

### Program Address Break Point 1

Reset value: XXXX<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PABP1															

Field	Bits	Type	Description
PABP1	15:0	rw	Address for generating a program address break.

## PABP2

### Program Address Break Point 2

Reset value: XXXX<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PABP2															

Field	Bits	Type	Description
PABP2	15:0	rw	Address for generating a program address break.

**CONFIDENTIAL**

**OCEM/SEIB**

### PABP3

**Program Address Break Point 3**

**Reset value: XXXX<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>PABP3</b>															

Field	Bits	Type	Description
<b>PABP3</b>	15:0	rw	Address for generating a program address break.

### SIGNATURE

**OCEM Programming Model Signature**

**Reset value: 3000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>SIGNATURE</b>								<b>RESERVED</b>							

Field	Bits	Type	Description
<b>SIGNATURE</b>	15:8	r	Signature to identify the OCEM programming model version. Set to 30 <sub>H</sub> (version 3.0) for the current implementation.
<b>RESERVED</b>	7:0	r	Reserved; these bits must be left at their reset values.

### PFTPG

**Trace Page Register**

**Reset value: XXXX<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>PFTPG</b>				<b>RESERVED</b>											

Field	Bits	Type	Description
<b>PFTPG</b>	15:12	rw	Reflects the program page of the last stage of the trace buffer FIFO. Refer to register <a href="#">PFT</a> .
<b>RESERVED</b>	11:0	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**OCEM/SEIB**

### **PACNT1**

#### **Program Address Break Point Counter 1**

**Reset value: XXXX<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>PAPG1</b>				<b>RESERVED</b>				<b>PACNT1</b>							

<b>Field</b>	<b>Bits</b>	<b>Type</b>	<b>Description</b>
<b>PACNT1</b>	7:0	rw	Program address break point counter. The counter is decremented on a matching break condition. The counter can be set to the following values: 0 or 1 A TRAP is generated at each matching condition. n A TRAP is only generated at the n <sup>th</sup> occurrence of a matching condition.
<b>PAPG1</b>	15:12	rw	Program page used for generation of program address break point 1.
<b>RESERVED</b>	11:8	r	Reserved; these bits must be left at their reset values.

### **PACNT2**

#### **Program Address Break Point Counter 2**

**Reset value: XXXX<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>PAPG2</b>				<b>RESERVED</b>				<b>PACNT2</b>							

<b>Field</b>	<b>Bits</b>	<b>Type</b>	<b>Description</b>
<b>PACNT2</b>	7:0	rw	Program address break point counter. The counter is decremented on a matching break condition. The counter can be set to the following values: 0 or 1 A TRAP is generated at each matching condition. n A TRAP is only generated at the n <sup>th</sup> occurrence of a matching condition.
<b>PAPG2</b>	15:12	rw	Program page used for generation of program address break point 2.
<b>RESERVED</b>	11:8	r	Reserved; these bits must be left at their reset values.



CONFIDENTIAL

OCEM/SEIB

### PACNT3

Program Address Break Point Counter 3

Reset value: XXXX<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PAPG3				RESERVED				PACNT3							

Field	Bits	Type	Description
PACNT3	7:0	rw	Program address break point counter. The counter is decremented on a matching break condition. The counter can be set to the following values: 0 or 1 A TRAP is generated at each matching condition. n A TRAP is only generated at the n <sup>th</sup> occurrence of a matching condition.
PAPG3	15:12	rw	Program page used for generation of program address break point 3.
RESERVED	11:8	r	Reserved; these bits must be left at their reset values.

### DAMASK

Data Address Mask Register

Reset value: XXXX<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAMASK															

Field	Bits	Type	Description
DAMASK	15:0	rw	<b>Mask for Data Address Comparison.</b> 1 The corresponding bits in register <b>DABP</b> and of the current data address are compared. 0 The corresponding bit of the current data address is not taken into consideration for comparison.

**CONFIDENTIAL**

**OCEM/SEIB**

## DABP

**Data Address Break Point**

**[Reset value: XXXX<sub>H</sub>]**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>DABP</b>															

Field	Bits	Type	Description
<b>DABP</b>	15:0	rw	Address to be used for data address break point generation.

## MODE0

**OCEM Mode Register 0**

**[Reset value: 0000<sub>H</sub>]**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>SSE</b>	<b>ILLE</b>	<b>BKR<sub>E</sub></b>	<b>TBF<sub>E</sub></b>	<b>INTE</b>	<b>BRE</b>	<b>P3E</b>	<b>P2E</b>	<b>P1E</b>	<b>EXT<sub>RE</sub></b>	<b>EXT<sub>WE</sub></b>	<b>CDV<sub>A</sub></b>	<b>DAR<sub>E</sub></b>	<b>DAW<sub>E</sub></b>	<b>DVR<sub>E</sub></b>	<b>DVW<sub>E</sub></b>

Field	Bits	Type	Description
<b>DVWE</b>	0	rw	<b>Data Value Write Enable</b> Enables data value break point on data write transaction upon match with the value at data value break point internal register (DVM register within the DSP core). 0 Disables 1 Enables
<b>DVRE</b>	1	rw	<b>Data Value Read Enable</b> Enables data value break point on data read transaction upon match with the value at data value break point internal register (DVM register within the DSP core). 0 Disables 1 Enables

Field	Bits	Type	Description
<b>DAWE</b>	2	rw	<b>Data Address Write Enable</b> Enables data address break point as a result on data write transaction where the address is matched with the address at data address break point register <b>DABP</b> . 0 Disables 1 Enables
<b>DARE</b>	3	rw	<b>Data Address Read Enable</b> Enables data address break point as a result on data read transaction where the address is matched with the address at data address break point register <b>DABP</b> . 0 Disables 1 Enables
<b>CDVA</b>	4	rw	<b>Combined Data Value and Address Enable</b> Enables break point as a result of simultaneous data address and data value match. 0 Disables 1 Enables
<b>EXTWE</b>	5	rw	<b>External Register Write Enable</b> Enables break point as a result of external register write transaction. 0 Disables 1 Enables
<b>EXTRE</b>	6	rw	<b>External Register Read Enable</b> Enables break point as a result of external register read transaction. 0 Disables 1 Enables
<b>P1E</b>	7	rw	<b>P1 Enable</b> Enables program break point 1. Break point is activated upon match on the address specified at register <b>PABP1</b> . 0 Disables 1 Enables

**CONFIDENTIAL**

**OCEM/SEIB**

Field	Bits	Type	Description
<b>P2E</b>	8	rw	<b>P2 Enable</b> Enables program break point 2. Break point is activated upon match on the address specified at register <b>PABP2</b> . 0     Disables 1     Enables
<b>P3E</b>	9	rw	<b>P3 Enable</b> Enables program break point 3. Break point is activated upon match on the address specified at register <b>PABP3</b> . 0     Disables 1     Enables
<b>BRE</b>	10	rw	<b>Break Enable</b> Enables break point every time the program jumps instead of executing the next sequential address. <i>Note: The BR is not activated as a result of interrupts, or REP instructions.</i> 0     Disables 1     Enables
<b>INTE</b>	11	rw	<b>Interrupt Enable</b> Enables break point upon detection of interrupt service routine execution. 0     Disables 1     Enables
<b>TBFE</b>	12	rw	<b>Program Flow Trace Buffer Full Enable</b> Enables break point as a result of program flow trace buffer full. 0     Disables 1     Enables
<b>BKRE</b>	13	rw	<b>Block Repeat Enable</b> Enables break point when returning to the beginning of block repeat loop. <i>Note: The indication for block repeat point is shown in BR bit within <b>STATUS0</b> register.</i> 0     Disables 1     Enables

**CONFIDENTIAL**

**OCEM/SEIB**

Field	Bits	Type	Description
<b>ILLE</b>	14	rw	<b>Illegal Access Enable</b> Enables break point on illegal condition (that is, trying to access the mail box space not through the break point handler). <b>ILLE</b> can be set/reset by the DSP core only while it is within a break point routine. 0     Disables 1     Enables
<b>SSE</b>	15	rw	<b>Single Step Enable</b> Enables single step operation. 0     Disables 1     Enables

**CONFIDENTIAL**

**OCEM/SEIB**

**STATUS1**

**OCEM Status Register 1**

**[Reset value: 1000 0000 0000 000X<sub>B</sub>]**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBG	BOOT	ERR	MVD	RESERVED											TREI

Field	Bits	Type	Description
<b>TREI</b>	0	rw	<b>Program Flow Trace Entry Indication</b> Acts as a tag bit per each program flow TRACE entry. 0 The current TRACE entry (the one that is at the trace top entry) has to be combined with the next TRACE entry for proper expansion of user program. 1 The current TRACE entry is not combined with the next TRACE entry.
<b>MVD</b>	12	rw	<b>Detection of MOVD Instruction</b> 0 No Instruction detected 1 MOVD Instruction detected
<b>ERR</b>	13	rw	<b>Detection of User Reset</b> 0 No user reset detection 1 User reset activated during execution of break point service routine; the DSP debugger might not follow the user commands.  <i>Note: User reset is the reset being activated from the user hardware and not through the debugger RESET command.</i>
<b>BOOT</b>	14	rw	<b>Boot</b> Indicates the debug mode. Activated as a result of indication on CA1 pin at RSTN rising edge.
<b>DBG</b>	15	rw	<b>Debug</b> Indicates the debug mode. Activated as a result of indication on CA0 pin at RSTN rising edge.
<b>RESERVED</b>	11:1	r	Reserved; these bits must be left at their reset values.

CONFIDENTIAL

OCEM/SEIB

## STATUS0

### OCEM Status Register 0

[Reset value: 0000<sub>H</sub>]

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SFT	ILL	TBF	INT	BR	RESERVED			PA3	PA2	PA1	ABORT	EREG	CDVA	DA	DV

Field	Bits	Type	Description
DV	0	rw	<b>Detection of Break Point Due to Matched Data Value</b> 0 No break point detected 1 Break point detected
DA	1	rw	<b>Detection of Break Point Due to Matched Data Address</b> 0 No break point detected 1 Break point detected
CDVA	2	rw	<b>Detection of Break Point Due to Matched Data Value and Matched Data Address</b> 0 No break point detected 1 Break point detected
EREG	3	rw	<b>Detection of Break Point Due to a Transaction Involving a User Defined Register</b> 0 No break point detected 1 Break point detected
ABORT	4	rw	<b>Detection of Break Point Due to an External Event</b> 0 No break point detected 1 Break point detected
PA1	5	rw	<b>Program Address 1</b> Indicates the detection of program address break point set in register <b>PABP1</b> . 0 No break point detected 1 Break point detected
PA2	6	rw	<b>Program Address 2</b> Indicates the detection of program address break point set in register <b>PABP2</b> . 0 No break point detected 1 Break point detected

**CONFIDENTIAL**

**OCEM/SEIB**

Field	Bits	Type	Description
<b>PA3</b>	7	rw	<b>Program Address 3</b> Indicates the detection of program address break point set in register <b>PABP3</b> . 0 No break point detected 1 Break point detected
<b>BR</b>	11	rw	<b>Branch Break Point</b> Indicates the detection of branch or block repeat. 0 No branch or block repeat point 1 branch or block repeat point detected
<b>INT</b>	12	rw	<b>Interrupt Break Point</b> Indicates the detection of break point due to acceptance of interrupt by the DSP core. 0 No break point detected 1 Break point detected
<b>TBF</b>	13	rw	<b>Program Flow Trace Buffer Full</b> Indicates the detection of break point caused by the trace buffer being full. 0 No break point detected 1 Break point detected
<b>ILL</b>	14	rw	<b>Illegal</b> Indicates the detection of illegal break point. Activated once the user program tries to access the mail box space or the OCEM registers not through the break point handler. 0 No illegal break point detected 1 Illegal break point detected
<b>SFT</b>	15	rw	<b>Software Trap</b> Indicates the detection of software trap. This means that the current break point was caused by a TRAP. 0 No break point detected 1 TRAP break point detected
<b>RESERVED</b>	10:8	r	Reserved for future use; these bits must be left at their reset values.

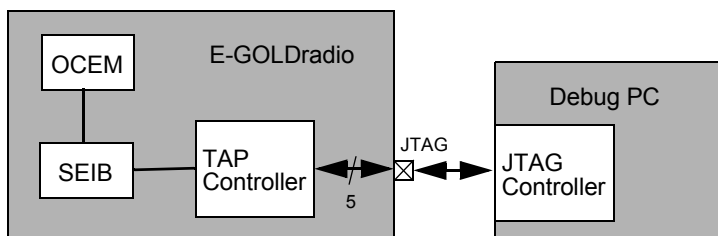




**CONFIDENTIAL**

**OCEM/SEIB**

**Figure 8-72 PMB7870 Debug PC Communication**



### 8.15.13.1 SEIB Physical Interface

**Table 8-52 Clock, Reset**

Signal Name	Direction	Source/ Destination	Activity	Description
LRSTP	Input	CLK	High	Reset
DSP_CLK	Input	CLK	High	DSP Clock

**Table 8-53 I/O Interface**

Signal Name	Direction	Source/ Destination	Activity	Description
GEXDBP [15:0]	In/Out	Core	High	DSP Data Bus
DXAP [9:0]	Input	Core	High	DSP Address Bus
I_DATA_P[15:0]	Output	Core	High	SEIB Data Out Bus
I_SEIB_RD_P	Input		High	SEIB Memory Read
I_SEIB_WR_P	Input		High	SEIB Memory Write
I_TDI_P	Input	TAP Controller	High	Serial Input Port
I_JTCLK_P	Input	TAP Controller	High	Serial JTAG Clock
O_TDO_P	Output	TAP Controller	High	Serial Output Port
O_ABORT_OUT_N	Output	OCEM	Low	Abort output to OCEM
O_READY_N	Output		Low	Force wait states when SEIB active

CONFIDENTIAL

RAM

## 9 LM-Bus

History	
Design Spec.	Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.00	
<b>Page 615</b>	Updated <b>Section 9.1 RAM</b> WS00005898
Changes for Rev. 1.04	
<b>Page 615</b>	Updated <b>Section 9.1 RAM</b> WS00008455
Changes for Rev. 1.06	
<b>Page 615</b> <b>Page 616</b>	Updated <b>System Intergration</b> for RAM and ROM WS00009050

The Local Memory Bus (LM-Bus) is a 16-bit data read-write/32-bit program read bus, that allows 32 bits to be fetched in parallel from the parts of the internal local memory that are used for the program code.

### 9.1 RAM

#### System Integration:

- Supply domain: VDD\_MAIN
- Chip internal interfaces:
- Clock domain: cgu\_clk\_mem\_o
- Interrupt sources: n/a
- Chip external signals related to this block: n/a
- Monitor Pins:

CONFIDENTIAL

ROM

The RAM is a 256 Kbit SRAM that contains the data of the MCU.

The LM-Bus access is programmable.

When **RST\_CTRL\_STA.LMRAMROMCL** equals:

- 0: 0 wait states for data access. For code fetches there is one wait state. This is independent of the MCU clock (52 MHz or 78 MHz).
- 1: One wait state is automatically inserted on all local memory bus accesses.

*Note: This is not true for code fetching because bit **LMRAMROMCL** does not effect the code fetching speed.*

By default at reset, **RST\_CTRL\_STA.LMRAMROMCL** is set to 1. For any other configuration of bits **LMRAMROMCL**, **CPUH**, and **CPUPRE** there is no wait state on the C166s\_lmbus data accesses.

*Note: This is also not true for code fetching because bit **LMRAMROMCL** does not effect the code fetching speed.*

## 9.2 ROM

### System Integration:

- Supply domain: VDD\_MAIN
- Chip internal interfaces:
- Clock domain: cgu\_clk\_mem\_o
- Interrupt sources: n/a
- Chip external signals related to this block: n/a
- Monitor Pins:

The ROM contains the needed code to boot the MCU.

The ROM size is 1K x 16 bits.

**CONFIDENTIAL**

**X-Bus Description**

## 10 X-Bus

History	
Design Spec.	Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.06	

### 10.1 X-Bus Description

The X-Bus access to the X-peripherals is determined by the **XBCONx** and **XADRSx** registers. The C166S core supports only synchronous-ready operations on the X-Bus.

The description of the XBCON registers is in [Section 6.5.2.1.2 X-Bus Control Registers \(on Page 208\)](#).

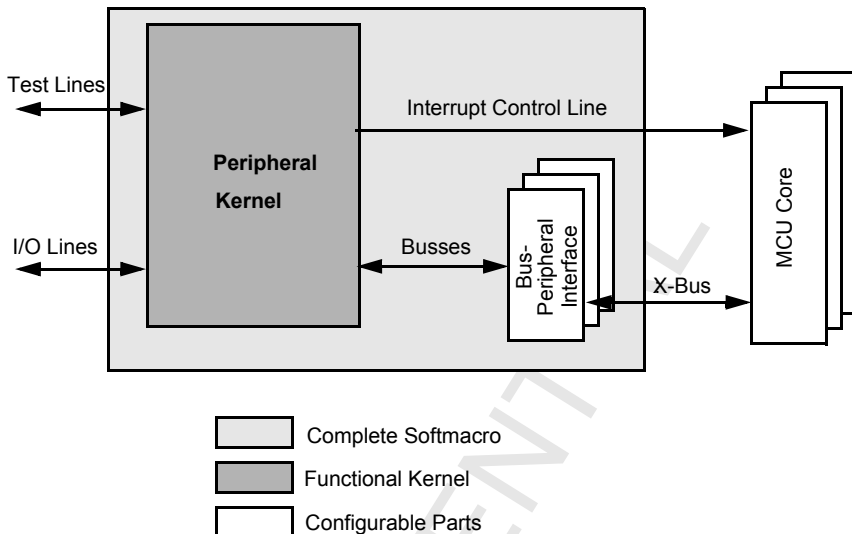
The description of the XADRS registers is in [Section 6.5.2.1.1 X-Bus Address Selection Registers \(on Page 206\)](#).

The Address Mapping of the X-Bus Peripherals is given in [Section 12.3 X-Bus Register Addresses \(on Page 1285\)](#).

#### 10.1.1 X-Bus Interface

All X-Bus peripherals are split into a peripheral kernel and a bus peripheral interface (BPI), see [Figure 10-1](#). The BPI interface performs the adaptation to the X-Bus.

**Figure 10-1 General Block Diagram of X-Bus Peripheral**



As the registers of the peripherals have different addresses in different MCUs, also the addresses are decoded in the BPI block.

X-Bus Peripherals which include only a very low number of registers (<256byte) should be mapped within one address window of 256 bytes and use only one **XADRSx/XBCONx** register pair. Therefore all these peripherals must use the same bus cycles (MUX/DEMUX, wait states, etc.) and allow a flexible mapping of their addresses into the 256byte address space of one **XADRSx/XBCONx** register pair. These should be ensured by the X-Bus-BPI. Which 256 byte address segment of the whole address space is used is determined by the **XADRSx** register and so by the XCS signal of the peripheral.

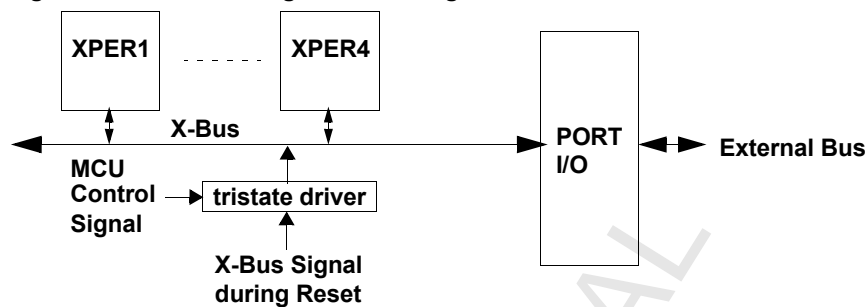
### 10.1.2 8- and 16-bit Access to X-Bus Peripherals

All registers in X-Bus peripherals should support byte accesses to the upper and lower byte of 16-bit words. RAM accesses (GSM Timer Unit, Shared memory) can only be performed 16 bit wise.

### 10.1.3 Reset Behavior of X-Bus

Refer to [Section 6.7.2 Reset Control \(on Page 231\)](#) and [Figure 10-2](#).

**Figure 10-2 MCU Configuration during Reset via X-Bus**



CONFIDENTIAL



## 10.2 Shared Memory

History	
Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.02	
<b>Page 622</b>	Updated <b>Figure 10-3 E-GOLDradio Duel-Port RAM Shared Memory</b> WS00007509
Changes for Rev. 1.06	

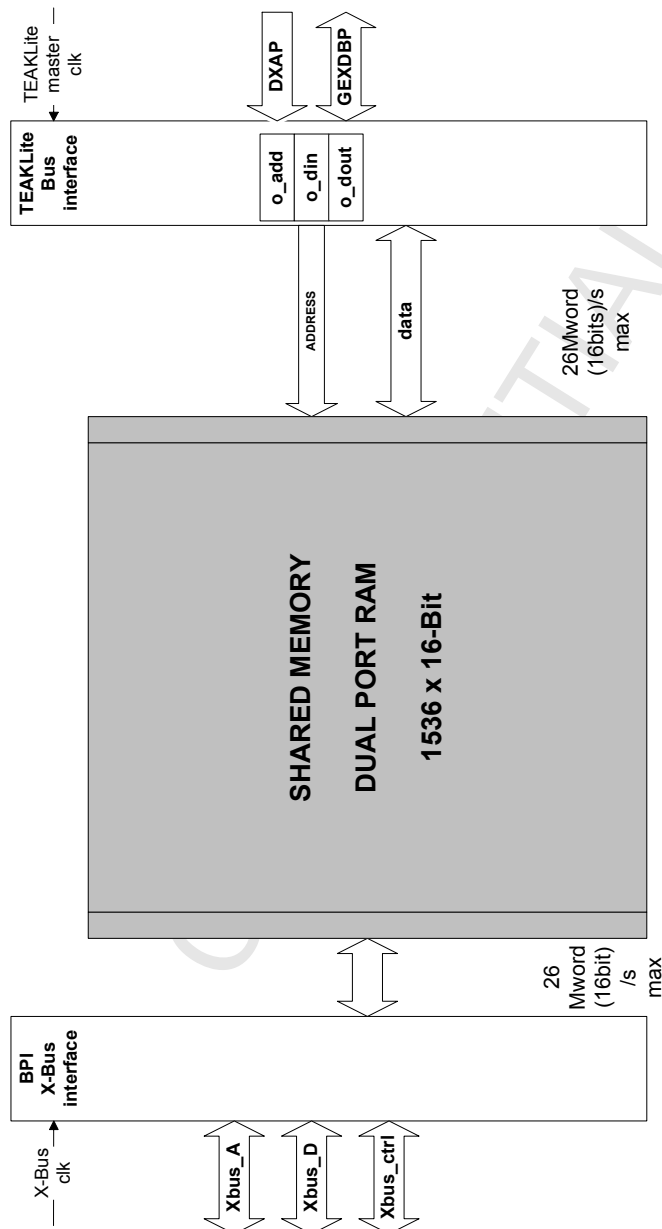
### System Integration:

- Supply domain: VDD\_Main
- Chip internal interfaces: TEAKlite Z Bus - X-Bus
- Interrupt sources: NA
- Monitor Pins: Refer to **Section 11.8.10 Internal Signal Monitoring (on Page 1209)**

### 10.2.1 Size

The size of the Shared Memory (SM) is 1536 x 16-bits (see **Figure 10-3**).

Figure 10-3 E-GOLDradio Duel-Port RAM Shared Memory



**CONFIDENTIAL**

**Shared Memory**

## **10.2.2 Access**

### **10.2.2.1 DSP-Side Addresses**

From the TEAKlite DSP the SM is 16-bit word aligned and:

- Begins at  $D800_H$
- Ends at  $DDFE_H$  (the last word).

### **10.2.2.2 MCU-Side Addresses**

From the MCU the SM is byte aligned and:

- Begins at  $D000_H$
- Ends at  $DBFF_H$  (the last byte).

The MCU must generate a 16-bit word-aligned address to:

- Read a word from the DSP
- Write a word to the DSP.

CONFIDENTIAL

**CONFIDENTIAL**

**Multicore Synchronization**

## 10.3 Multicore Synchronization

History	
Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.3,
Changes for Rev. 1.02	
<b>Page 626</b>	Updated <b>Figure 10-4 Multicore Synchronization</b> WS00007509
Changes for Rev. 1.06	

### System Integration of TEAKLite:

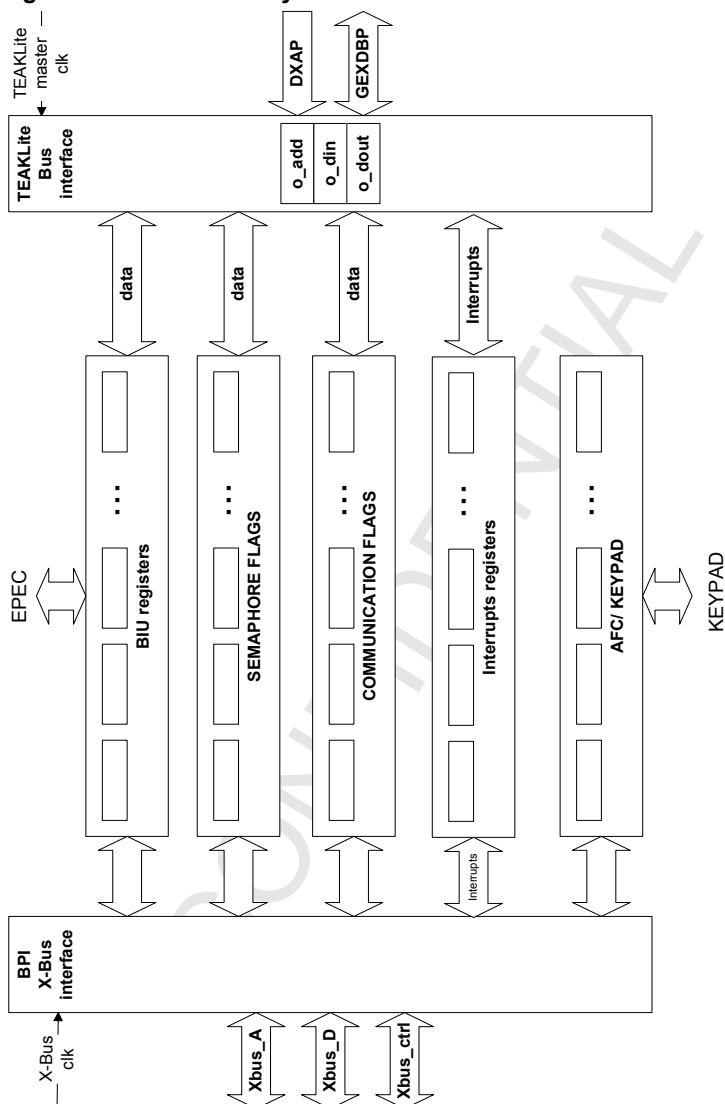
- Supply domain: VDD\_MAIN
- Chip internal interfaces:
  - Bus domain: X-Bus and TEAKLite Z-Bus
- Interrupts: none
- Monitor Pins: Refer to **Section 11.8.10 Internal Signal Monitoring (on Page 1209)**

### 10.3.1 General

The Multicore Synchronization mechanism contains five separate functional blocks (see **Figure 10-4**):

- The Shared Memory itself: Dual Port RAM of 1.5 kWord, organized in 1536x16 bits, refer to **Section 10.2 Shared Memory (on Page 621)**.
- A set of semaphore flags to control the use of resources shared between TEAKLite and Controller as for example the Dual Port RAM, refer to **Section 10.3.4 Semaphore Flags (on Page 637)**.
- A set of communication flags for the exchange of status information between TEAKLite and Controller, refer to **Section 10.3.3 Communication Flags (on Page 631)**.
- A set of BIU registers which allow to TEAKLite to program MCU to do an access on all registers in the MCU address field, refer to **Section 10.3.5 XBIU (on Page 643)**.
- Four software interrupts from the C166S to the TEAKlite core and four software interrupts from the TEAKlite to the C166S core.

Figure 10-4 Multicore Synchronization



### 10.3.2 Cross Software interrupts

The C166S can generate four software interrupts to the TEAKlite, the description is in the chapter [MCU to DSP interrupts \(on Page 627\)](#).

The TEAKlite can generate four software interrupts to the C166S, the description is in the chapter [DSP to MCU interrupts \(on Page 628\)](#).

#### 10.3.2.1 MCU to DSP interrupts

##### Interrupt generation

The bits 0 to 3 of the register [MCU\\_DSP\\_INT\\_ACK](#) are used by the MCU to generate the interrupt(s) to the DSP.

Setting one of these bits to 1 will activate the corresponding interrupt(s) in [Section 8.1 TEAKLite Interrupt Unit \(on Page 337\)](#):

- MCU0
- MCU1
- MCU2
- MCU3.

Refer to [INT\\_FINTA0 \(on Page 341\)](#).

##### Interrupt acknowledge

The interrupt acknowledge is done by the DSP via the register [INT\\_RINTA0 \(on Page 343\)](#).

The bits 8 to 11 of the register [MCU\\_DSP\\_INT\\_ACK](#) indicates the acknowledge of the previously generated interrupt. Their values are the image of the bit 0 to 3 from the [INT\\_RINTA0](#) register.

### 10.3.2.2 DSP to MCU interrupts

#### Interrupt generation

The bits 0 to 4 of the register **INT\_TOMCU** (on Page 352) are used by the DSP to generate the interrupt(s) to the MCU.

Setting one of these bits to 1 will activate the respective interrupt(s) in the C166S interrupt controller:

- FROM\_DSP\_TO\_MCU0\_INT
- FROM\_DSP\_TO\_MCU1\_INT
- FROM\_DSP\_TO\_MCU2\_INT
- FROM\_DSP\_TO\_MCU3\_INT

Refer to **Section 6.3.3 Interrupt and Exception Execution** (on Page 111).

#### Interrupt acknowledge

The bits 4 to 7 of the register **MCU\_DSP\_INT\_ACK** will generate a pulse to acknowledge the previously generated interrupt. This pulse will clear the **INT\_TOMCU** interrupt in respect to the MCU interrupt behavior.

Refer to **Section 6.3.3 Interrupt and Exception Execution**.



**CONFIDENTIAL**

**Multicore Synchronization**

### 10.3.2.3 MCU/DSP Interrupt and Acknowledge Register

**MCU\_DSP\_INT\_ACK**

**MCU/DSP Interrupt and Acknowledge Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				DSP_TO_MCU_ACK				MCU_TO_DSP_ACK				MCU_TO_DSP_IRQ			

Field	Bits	Type	Description
<b>MCU_TO_DSP_IRQ</b>	0:3	rw from MCU	<b>Interrupt (from MCU to DSP)</b> 0 Not used 1 The interrupt is set to high-level. The bit 0 is mapped to the first MCU to DSP interrupt, and so on.
<b>MCU_TO_DSP_ACK</b>	7:4	w from MCU	<b>Acknowledge from MCU to DSP</b> 0 Not used 1 A pulse (2 clock cycles) is generated. The bit 4 is the acknowledge indication of the first MCU to DSP interrupt, and so on.
<b>DSP_TO_MCU_ACK</b>	11:8	r from MCU	<b>Acknowledge indication (from DSP to MCU)</b> The bit 8 is the acknowledge indication of the first MCU to DSP interrupt, and so on.
<b>Reserved</b>	15:12	r	Reserved; these bits must be left at their reset values.

CONFIDENTIAL

**CONFIDENTIAL**

### 10.3.3 Communication Flags

History	
Design Spec.	Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.3,
Changes for Rev. 1.01	
<b>Page 631</b>	Updated <b>System Integration</b> : WS00006682
Changes for Rev. 1.06	

#### System Integration:

- Supply domain: VDD\_MAIN
- Chip internal interfaces:
  - Clock domains: Refer to **Section 10.4.1.3 Sub-System Clocks and Enables (on Page 661)** and see **Figure 10-11 Clock and Enable Generation for MCU Sub-System (on Page 668)**.
  - Bus domain: X-Bus and TEAKLite Z-Bus
- Interrupts: none
- Monitor Pins: Refer to **Section 11.8.10 Internal Signal Monitoring (on Page 1209)**

#### 10.3.3.1 Access of Communication Flags

##### MCU Access

The access of the communication flags is performed using the BPI-X-Bus-Interface.

##### TEAKLite Access

The TEAKLite access is done via the TEAKLite peripheral clock. The clock gating is done via the register select lines. With an active select signal the TEAKLite master clock is fed through for supplying the communication flags.

#### 10.3.3.2 Description

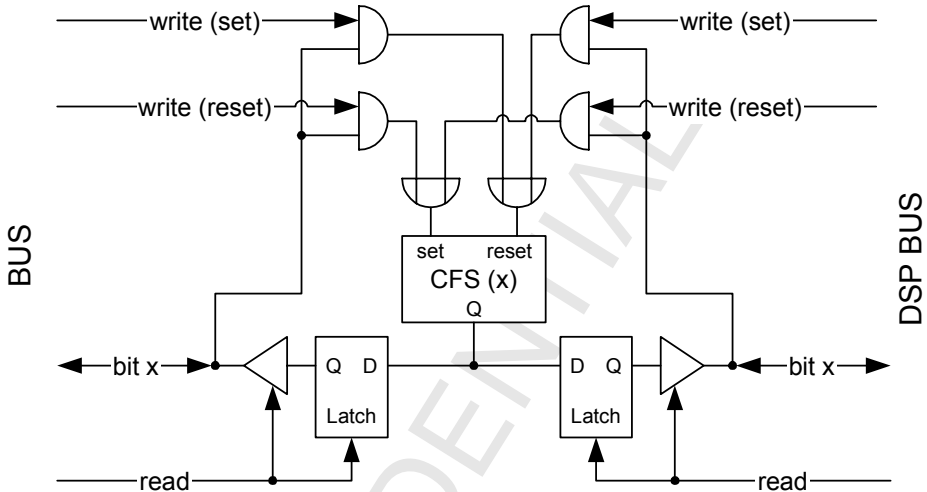
A set of 16 communication flags is provided to exchange the status of the communication protocol between MCU and TEAKLite. The communication flag can be set and reset by MCU and TEAKLite. By means of the communication protocol it has to be guaranteed that such a write operation is never performed by both processors at the same time. The

**CONFIDENTIAL**

status of the communication flag can always be read by both devices, no matter whether a write operation is currently performed.

A communication flag cell is shown in **Figure 10-5**.

**Figure 10-5 Block Diagram of Communication Flag (Only One Slice Shown)**



In the following Communication Flag registers they can be set, reset, and read. For the register addresses refer to [Section 12.3 X-Bus Register Addresses \(on Page 1285\)](#).

**CONFIDENTIAL**

### 10.3.3.2.1 MCU Communication Flag Identification Register

**MCU\_CFID**

**Peripheral Identification Register**

**Reset value: AA01<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>MOD_NUMBER</b>								<b>MOD_REV</b>							

Field	Bits	Type	Description
<b>MOD_REV</b>	7:0	r	<b>Module Revision Number</b> Value = 01 <sub>H</sub>
<b>MOD_NUMBER</b>	15:8	r	<b>Module Number.</b> For ASC1, it is AA <sub>H</sub>

### 10.3.3.2.2 MCU Communication Flag Set Register

**MCU\_CFS**

**MCU Communication Flag Set Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>MCUCFS<sub>x</sub></b>															

Field	Bits	Type	Description
<b>MCUCFS<sub>x</sub></b> (x = 0 to 15)	15:0	w by MCU	<b>Communication Flag Set Bits</b> 0 No change of Communication Flag 1 MCU sets Communication Flag SM_CF(x)

**CONFIDENTIAL**

### 10.3.3.2.3 MCU Communication Flag Reset Register

**MCU\_CFR**

**MCU Communication Flag Reset Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>MUCCFRx</b>															

Field	Bits	Type	Description
<b>MUCCFRx</b> (x = 0 to 15)	15:0	w by MCU	<b>Communication Flag Reset Bits</b> 0 No change of Communication Flag 1 MCU resets Communication Flag SM_CF(x)

### 10.3.3.2.4 MCU Communication Flag Status Register

**MCU\_CFSTA**

**MCU Communication Flag Status Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>MCUCFSTA<sub>x</sub></b>															

Field	Bits	Type	Description
<b>MCUCFSTA<sub>x</sub></b> (x = 0 to 15)	15:0	w by MCU	<b>Communication Flag Bits</b> This is the value of Communication Flag SM_CF(i)

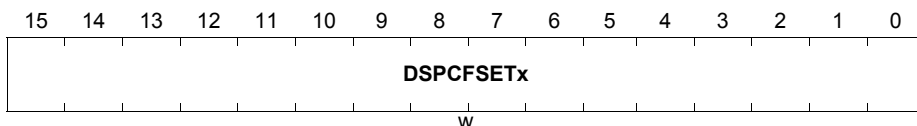
**CONFIDENTIAL**

### 10.3.3.2.5 DSP Communication Flag Set Register

**DSP\_CFSET**

**DSP Communication Flag Set Register**

**Reset value: 0000<sub>H</sub>**



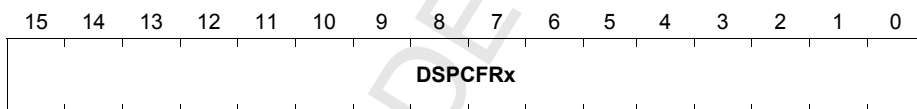
Field	Bits	Type	Description
<b>DSPCFSETx</b> (x = 0 to 15)	15:0	w by DSP	<b>Communication Set Flag Bits</b> 0 No change of Communication Flag x 1 DSP sets Communication Flag x

### 10.3.3.2.6 DSP Communication Flag Reset Register

**DSP\_CFR**

**DSP Communication Flag Reset Register**

**Reset value: 0000<sub>H</sub>**



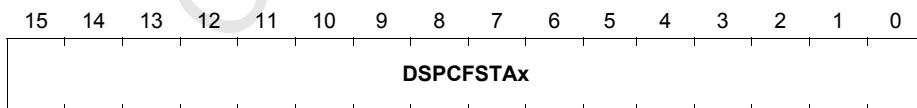
Field	Bits	Type	Description
<b>DSPCFRx</b> (x = 0 to 15)	15:0	w by DSP	<b>Communication Flag Reset Bits</b> 0 No change of Communication Flag x 1 DSP resets Communication Flag x

### 10.3.3.2.7 DSP Communication Flag Status Register

**DSP\_CFSTA**

**DSP Communication Flag Status Register**

**Reset value: 0000<sub>H</sub>**



Field	Bits	Type	Description
<b>DSPCFSTAx</b> (x = 0 to 15)	15:0	r by DSP	<b>Communication Flag Bits</b> This is the value of Communication Flag SM_CF(i).

**CONFIDENTIAL**

### 10.3.3.3 Communication Flags Usage

The shared memory comprises N ( $N \leq 8$ ) fixed address locations for MCU commands which have to be performed by the TEAKLite. These address locations are called instruction banks abbreviated with `instr_b(N-1:0)`. For each instruction bank two communication flags are provided.

1. The `INSTR(i)` flag (for example, `MCU_CFSTA[7:0]`) is high when the `instr_b(i)` contains a instruction of the MCU which has not been read by the TEAKLite. When the `INSTR(i)` flag is low there is no valid instruction in `instr_b(i)`.
2. The `OP(i)` flag (for example, `MCU_CFSTA[15:8]`) is high when the TEAKLite has performed the instruction of `instr_b(i)`.

If the MCU issues a command to the TEAKLite, the following protocol must be used by MCU and TEAKLite:

#### MCU Tasks

1. IF(`INSTR(i) = 0`) THEN  
    MCU writes instruction into `instr_b(i)`  
    END IF
2. MCU sets `INSTR(i)` flag (must be set before the MCU rises TEAKLite interrupt)  
    MCU resets `OP(i)` flag (must be reset before MCU starts polling)
3. MCU performs interrupt of TEAKLite
4. MCU starts polling of `INSTR(i)` and `OP(i)` flag, in order to know when the instruction was read by the TEAKLite respectively the corresponding operation has been performed by the TEAKLite.

#### TEAKLite Tasks

(They have to be performed when a MCU interrupt has been detected by TEAKLite.)

1. TEAKLite reads `INSTR(7:0)`
2. For all instruction banks with `INSTR(i) = 1`:
  - a) TEAKLite reads corresponding `instr_b(i)`
  - b) TEAKLite resets corresponding `INSTR(i)` flag
  - c) TEAKLite performs operations of corresponding `instr_b(i)`  
    When operation is executed, TEAKLite sets corresponding `OP(i)` flag

Using the above described software protocol it can be guaranteed that the instruction banks `instr_b(i)` are never accessed concurrently by TEAKLite and MCU. The `INSTR(i)` and `OP(i)` flags can be implemented by means of the Communication Flags defined in this chapter, since they are never written by the TEAKLite and MCU at the same time, but can always be read (for example, by using `MCU_CFSTA[7:0]` as `INSTR[7:0]` and `MCU_CFSTA[15:8]` as `OP[7:0]`).



**CONFIDENTIAL**

### 10.3.4 Semaphore Flags

History	
Current version: Rev. 1.06, 2005-11-04	
Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.01	
<b>Page 637</b>	Updated <b>System Integration</b> : WS00006682
Changes for Rev. 1.06	

#### System Integration:

- Supply domain: VDD\_MAIN
- Chip internal interfaces:
  - Clock domains: Refer to **Section 10.4.1.3 Sub-System Clocks and Enables (on Page 661)** and see **Figure 10-11 Clock and Enable Generation for MCU Sub-System (on Page 668)**.
  - Bus domain: X-Bus and TEAKLite Z-Bus
- Interrupts: none.
- Monitor Pins: Refer to **Section 11.8.10 Internal Signal Monitoring (on Page 1209)**

#### 10.3.4.1 Semaphore Flag Access

##### MCU Access

The access of the semaphore flags is performed using the BPI-X-Bus-Interface.

##### TEAKLite Access

The TEAKLite access is done via the TEAKLite peripheral clock. The clock gating is done via the register select lines. With an active select signal the TEAKLite master clock is fed through for supplying the semaphore flags.

#### 10.3.4.2 Description

Semaphore signalling is a method of allocating mutually exclusive accesses to blocks of memory that are shared among several processors. In PMB7870 a set of 16 semaphores is provided to control the use of resources shared between MCU and TEAKLite. The particular use of the semaphores will be defined by software and

## CONFIDENTIAL

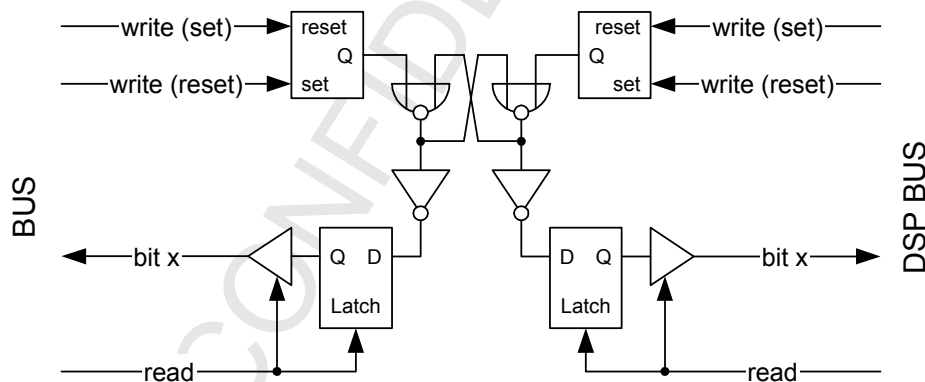
firmware. Examples of resources which can be controlled are sections of the shared memory, a serial port or off chip memory.

Semaphore control requests are handled using a standard write to the semaphore latch followed by a read instruction. There is no requirement to lockout the other processor access to the semaphore between the write and read. If a semaphore is requested and afterwards a 0 is read, the semaphore has been free and is now owned by the requesting processor. If a 1 is read the semaphore was already locked by the other processor and the blocked processor has to continue its read operation until a 0 can be read which shows that the semaphore is freed by the other processor and now owned by the formerly blocked processor. The semaphore is freed by the processor by releasing the semaphore.

The semaphore latch cell can only have three states (see [Figure 10-6](#))

1. Nodes A and B both high: The Semaphore is free. Either side can write a 0 to the cell and when one side can read the 0 then it has gained control of the semaphore..
2. Node A low, node B high: the semaphore is owned by the X-Bus bus. Only an X-Bus bus write of 1 can free the semaphore.
3. Node A high, node B low: the semaphore is owned by the TEAKLite-BUS. Only a TEAKLite-BUS write of 1 can free the semaphore.

**Figure 10-6 Block Diagram of Semaphore Flag (only one slice shown)**



**CONFIDENTIAL**

### 10.3.4.2.1 MCU Semaphore Flag Identification Register

**MCU\_SEMID**

**Peripheral Identification Register**

**Reset value: AA01<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>MOD_NUMBER</b>								<b>MOD_REV</b>							

Field	Bits	Type	Description
<b>MOD_REV</b>	7:0	r	<b>Module Revision Number</b> Value = 01 <sub>H</sub>
<b>MOD_NUMBER</b>	15:8	r	<b>Module Number.</b> Value = AA <sub>H</sub>

### 10.3.4.2.2 MCU Semaphore Set Register

**MCU\_SEMS**

**MCU Semaphore Set Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>MCUSEMSx</b>															

Field	Bits	Type	Description
<b>MCUSEMSx</b> (x = 0 to 15)	15:0	w by MCU	<b>Semaphore Flag Set Bits</b> 0 No semaphore request 1 X-Bus bus requests semaphore x

**CONFIDENTIAL**

### 10.3.4.2.3 MCU Semaphore Reset Register

**MCU\_SEMR**

**MCU Semaphore Reset Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>MCUSEMRx</b>															

Field	Bits	Type	Description
<b>MCUSEMRx</b> (x = 0 to 15)	15:0	w by MCU	<b>Semaphore Flag Set Bits</b> 0 No action 1 X-Bus bus releases semaphore x

### 10.3.4.2.4 MCU Semaphore Register

**MCU\_SEM**

**MCU Semaphore Register**

**Reset value: FFFF<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>MCUSEMx</b>															

Field	Bits	Type	Description
<b>MCUSEMx</b> (x = 0 to 15)	15:0	w by MCU	<b>Semaphore Flags</b> 0 When semaphore i was requested by the MCU ( <b>MCUSEMS(x) = 1</b> ) and a following read operation shows <b>MCUSEM(x) = 0</b> , the semaphore x is now owned by the MCU. 1 When semaphore i was requested by the MCU ( <b>MCUSEMS(x) = 1</b> ) and a following read operation shows <b>MCUSEM(x) = 1</b> , the semaphore i is already owned by the TEAKLite bus.

**CONFIDENTIAL**

### 10.3.4.2.5 DSP Semaphore Set Register

**DSP\_SEMS**

**DSP Semaphore Set Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>DSPSEMSx</b>															

Field	Bits	Type	Description
<b>DSPSEMSx</b> (x = 0 to 15)	15:0	w by DSP	<b>Semaphore Set Bits</b> 0 No action 1 DSP bus requests semaphore x

### 10.3.4.2.6 DSP Semaphore Reset Register

**DSP\_SEMR**

**DSP Semaphore Reset Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>DSPSEMRx</b>															

Field	Bits	Type	Description
<b>DSPSEMRx</b> (x = 0 to 15)	15:0	w by DSP	<b>DSP Semaphore Reset Bits</b> 0 No action 1 DSP bus releases semaphore x

CONFIDENTIAL

### 10.3.4.2.7 DSP Semaphore Flags Register

DSP\_SEM

DSP Semaphore Flags Register

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DSPSEM <sub>x</sub>															

Field	Bits	Type	Description
<b>DSPSEM<sub>x</sub></b> (x = 0 to 15)	15:0	w by DSP	<b>Semaphore Flags</b> 0 When semaphore i was requested by the DSP ( <b>DSPSEMS</b> (x) = 1) and a following read operation shows <b>DSPSEM</b> (x) = 0, the semaphore x is now owned by the DSP. 1 When semaphore i was requested by the DSP ( <b>DSPSEMS</b> (x) = 1) and a following read operation shows <b>UCSEM</b> (x) = 1, the semaphore x is already owned by the X-Bus bus.

**CONFIDENTIAL**

### 10.3.5 XBIU

History	
Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.01	
<b>Page 643</b>	Updated <b>System Integration</b> : WS00006682
Changes for Rev. 1.06	

#### System Integration:

- Supply domain: VDD\_MAIN
- Chip internal interfaces:
  - Clock domains: Refer to **Section 10.4.1.3 Sub-System Clocks and Enables (on Page 661)** and see **Figure 10-11 Clock and Enable Generation for MCU Sub-System (on Page 668)**.
  - Bus domain: X-Bus and TEAKLite Z-Bus
- Interrupts: none
- Monitor Pins: Refer to **Section 11.8.10 Internal Signal Monitoring (on Page 1209)**

#### 10.3.5.1 Overview

The Bus Interface Unit between the X-Bus and TEAKLite-bus (XBIU) enables the TEAKLite to access all registers located in the controller memory space, for example, the X-Bus registers, PD-Bus registers, and internal registers. Only the Program Bus Memory can not be accessed via the XBIU. A function which allows the MCU to read or write registers on the TEAKLite-bus is not available, the TEAKLite must request, via the XBIU, the MCU to perform a single byte or word transfer from any memory location addressable by the MCU to a fixed data register in the XBIU, which is then accessible by the TEAKLite.

The XBIU is used, for example, for the battery and temperature measurements.

#### 10.3.5.2 Description

##### 10.3.5.2.1 Access to BIU Registers

The Bus Interface Unit between the X-Bus and TEAKLite-bus (BIU) enables the TEAKLite to access all registers located in the controller memory space, for example, X-

## CONFIDENTIAL

Bus registers, PD-Bus registers, internal registers. Only the Program Bus Memory can not be accessed via the XBIU. A function which allows the MCU to read or write registers on the TEAKLite-bus is NOT foreseen. For this purpose the TEAKLite requests, via the XBIU, the MCU to perform a single byte or word transfer from any memory location addressable by the MCU to a fixed data register in the XBIU which is also accessible by the TEAKLite.

The BIU is used for example for the battery and temperature measurement path.

### 10.3.5.2.2 EPEC Access

TEAKLite writes to BIU write-only registers on TEAKLite data bus to program a EPEC channel, which EPEC perform a transfer through the BIU data registers with the parameters given by TEAKLite.

### 10.3.5.2.3 Read Operation

The TEAKLite writes the address of the MCU register from which a READ operation should be transferred into **SD\_ADDRL/SD\_ADDRH** registers in the XBIU. When the TEAKLite sets the **BIU\_CTRL (on Page 648).RQACK** (Request/Acknowledge) bit in the XBIU, the MCU performs a PEC transfer from the specified address to the **BIU\_DATA\_R (on Page 649)** in the XBIU. When the MCU has performed this operation, the **BIU\_CTRL.RQACK** bit is reset by the XBIU and the TEAKLite can access the **BIU\_DATA\_R** register.

### 10.3.5.2.4 Write Operation

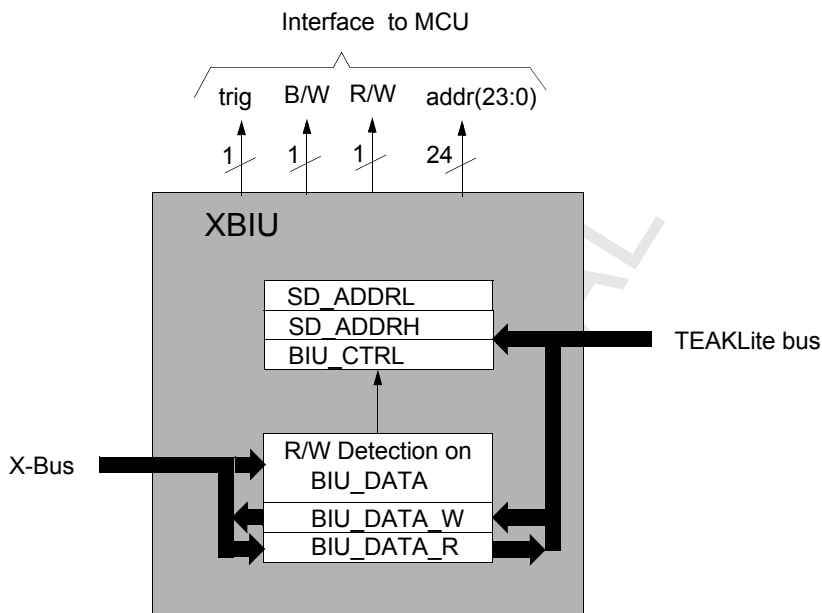
The TEAKLite writes the data to be transferred into the **BIU\_DATA\_W (on Page 649)** register of the XBIU and the MCU register address to which the data should be transferred, into the **SD\_ADDRL/SD\_ADDRH** registers. When the TEAKLite sets the **BIU\_CTRL.RQACK** bit in the XBIU, the MCU performs a PEC transfer from the **BIU\_DATA\_W** register to the specified register in the controller memory space. When the MCU has performed this operation, the **BIU\_CTRL.RQACK** bit is reset by the XBIU.

The MCU has for this function a PEC channel in addition to the 8 standard PEC channels. A principle block diagram of the XBIU can be seen in **Figure 10-7**.



CONFIDENTIAL

Figure 10-7 Bus Interface Unit between X-Bus and TEAKLite



**Note:** If the Bus Interface Unit between X-Bus and TEAKLite is used by the TEAKLite to access registers located in the controller memory space, **XADRS1.RGSAD** must be  $0EF_H$ , **XADRS1.RGSZ**:  $0000_H$  (256 bytes). Any XBIU operation is controlled by the TEAKLite only. . If the Bus Interface Unit between X-Bus and TEAKLite is NOT used by the TEAKLite, since the TEAKLite does not access registers located in the controller memory space, then **XADRS1.RGSAD** is freely programmable.

**CONFIDENTIAL**

### 10.3.5.3 I/O Signals

**Table 10-1 Signals to DPEC Block of MCU**

Signals	Function
trig	Signal indicates DPEC block of MCU that TEAKLite requests data transfer (toggle line).
B/W	Signal indicates whether a byte or word transfer has to be performed (derived from <b>BIU_CTRL.INSTR</b> bits).
R/W	Signal indicates whether MCU has to perform a read access to <b>BIU_DATA_W</b> or write access to <b>BIU_DATA_R</b> register (derived from <b>BIU_CTRL.INSTR</b> bits).
addr(23:0)	Source or destination address for data transfer, identical to registers <b>SD_ADDRL/SD_ADDRH</b> bits.

### 10.3.5.4 XBIU Register Descriptions

For the address mapping of these registers refer to [Section 12.3 X-Bus Register Addresses \(on Page 1285\)](#).

#### BIUID

#### X-Bus Interface Unit Identification Register

Reset value: **A901<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>MOD</b>								<b>REV</b>							

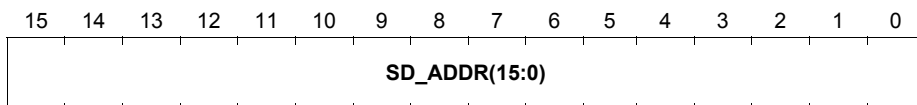
Field	Bits	Type	Description
<b>REV</b>	7:0	r	<b>XBIU Revision Number</b> The REV bits are used for revision numbering. They are 01 <sub>H</sub> for this module.
<b>Mod</b>	15:8	r	<b>XBIU Module Identification Number</b> The MOD bits are used for module identification. They are A9 <sub>H</sub> for this module.

**CONFIDENTIAL**

### SD\_ADDRL

**Low Source/Destination Address Register**

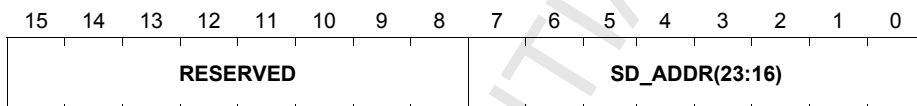
**Reset value: 0000<sub>H</sub>**  
**Read/Write by TEAKLite**



### SD\_ADDRH

**High Source/Destination Address Register**

**Reset value: 0000<sub>H</sub>**  
**Read/Write by TEAKLite**



Field	Bits	Type	Description
<b>SD_ADDR (15:0)</b> <b>SD_ADDR (23:16)</b>	15:0, 7:0	rw	Source Address for Read, Destination Address for Write Operation of TEAKLite to MCU memory space (24 bits). Can only be accessed by TEAKLite.
<b>RESERVED</b>	15:8	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**BIU\_CTRL**  
**XBIU Control Register**

Reset value: 0000<sub>H</sub>  
Read/Write by TEAKLite

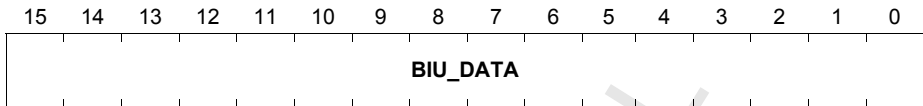
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED													INSTR	RQ ACK	

Field	Bits	Type	Description
<b>RQACK</b>	0	rw	<b>Request/Acknowledge Bit</b> 0: Cleared by XBIU itself, when XBIU detects a X-Bus access to the BIU_DATA register 1: Set by TEAKLite when a data transfer should be performed
<b>INSTR</b>	1:2	rw	<b>Instruction Bits</b> 00: Reserved 01: TEAKLite requests byte transfer from <a href="#">BIU_DATA_W</a> register to <a href="#">SD_ADDRL/SD_ADDRH</a> 10: TEAKLite requests word transfer from <a href="#">BIU_DATA_W</a> register to <a href="#">SD_ADDRL/SD_ADDRH</a> 11: TEAKLite requests word transfer from <a href="#">SD_ADDRL/SD_ADDRH</a> to <a href="#">BIU_DATA_R</a> register'
<b>RESERVED</b>	15:3	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**BIU\_DATA\_W**  
**XBIU Data Register**

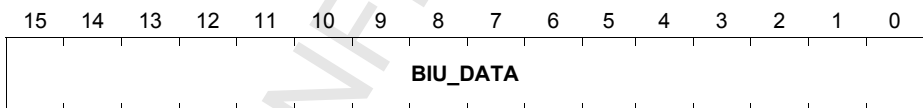
**Reset value: 0000<sub>H</sub>**  
**Write by TEAKLite**  
**Read by MCU**



Field	Bits	Type	Description
BIU_DATA	15:0	w by DSP by MCU	<b>DATA Register</b> If <b>BIU_CTRL.INSTR</b> = 01: MCU transfers data (bytes) from <b>BIU_DATA</b> to <b>SD_ADDRL/SD_ADDRH</b> 10: MCU transfers data (words) from <b>BIU_DATA</b> to <b>SD_ADDRL/SD_ADDRH</b>

**BIU\_DATA\_R**  
**XBIU Read Data register**

**Reset value: 0000<sub>H</sub>**  
**Read by TEAKLite**  
**Write by MCU**



Field	Bits	Type	Description
BIU_DATA	15:0	r by DSP w by MCU	<b>DATA Register</b> If <b>BIU_CTRL.INSTR</b> = 11: MCU writes data word into <b>BIU_DATA</b> bits, which can be read by TEAKLite, when <b>BIU_CTRL.RQACK</b> = 0

CONFIDENTIAL

### 10.3.5.5 Procedure for TEAKLite Access to X-Bus

#### TEAKLite Read Operation

1. Program the read address into **SD\_ADDRL/SD\_ADDRH** registers
2. Program **BIU\_CTRL** register (**RQACK** = 1, **INSTR** = 11<sub>B</sub>)
3. Poll **BIU\_CTRL.RQACK** bit  
If **RQACK** = 0, TEAKLite reads the **BIU\_DATA\_R** register.

#### TEAKLite Write Operation

1. Program the write address into **SD\_ADDRL/SD\_ADDRH** registers and the data into **BIU\_DATA\_W** register.
2. Program **BIU\_CTRL** register (**RQACK** = 1, **INSTR** = 10<sub>B</sub> or 01<sub>B</sub>)
3. Poll **BIU\_CTRL.RQACK** bit  
If **RQACK** = 0, the operation is finished.

### 10.3.5.6 TEAKLite Read/Write Request Response Time

The TEAKLite requests the MCU to perform an PEC transfer. This PEC transfer of the MCU is not maskable and always has a priority over interrupt requests on any priority level. This PEC transfer has even a higher priority than trap interrupts besides the trap interrupt of the OCDS debug block.

It takes 1 to 2 MCU clock cycles to synchronize the TEAKLite request to the MCU clock.

In addition, the normal PEC response time has to be added. In best case this delay is 5 MCU clock cycles, in worst case this delay takes 9-word bus accesses to the external memory.

If the MCU clock frequency is 6.5 MHz and if the external memories are 16-bit wide and can be accessed without any WAIT state, the total response time to a read/write request of the TEAKLite takes between 1050 ns and 2700 ns.

**CONFIDENTIAL**

**PLL - CGU - SCCU**

## 10.4 PLL - CGU - SCCU

History	
Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 0.02	
<b>Page 674</b>	Updated <b>CGUID</b> register
<b>Page 705</b>	Updated <b>SCCUID</b> register
<b>Page 657</b> <b>Page 659</b> <b>Page 660</b>	Updated <b>Section 10.4.1.1 GSM/GPRS Baseband System Clock Concept Description, Section 10.4.1.2.1 Startup on External Reset and Section 10.4.1.2.3 Changing Secondary Phase Shifter Frequency (MCU Phase Shifter)</b>
<b>Page 663</b> to <b>Page 667</b>	Updated <b>Table 10-3 MCU Clock Master</b> and Note, <b>Table 10-4 MCU Sub-System Clock,</b> <b>Table 10-5 MCU Sub-System Bus Clocks,</b> <b>Table 10-6 MCU Sub-System Peripheral Clocks (ASC0, ASC1, SCC, PCL, IIC),</b> and <b>Table 10-7 MCU Sub-System Peripheral Clocks (CAPCOM, GPT)</b>
<b>Page 675</b>	Updated <b>MST_CLK_CTRL</b> register
<b>Page 680</b>	Added <b>PHX2_CTRL</b> register
<b>Page 681</b>	Updated <b>SIFCLKS</b> register
<b>Page 683</b>	Updated <b>RST_CTRL_STA</b> register
<b>Page 680</b>	Updated <b>Table 10-14 Phase Shifter Frequencies</b>
Changes for Rev. 1.00	
<b>Page 656</b>	Updated <b>Figure 10-9 Overview of E-GOLDradio CPU Clock Generation System</b>
Changes for Rev. 1.01	
<b>Page 658</b>	Updated <b>Table 10-2 Target Operating Frequencies for E-GOLDradio (on Page 658)</b> WS00006912
Changes for Rev. 1.02	
<b>Page 662</b>	Added <b>Figure 10-10 Clock Enable</b> WS00006643
<b>Page 664</b> to <b>Page 667</b>	Added a Note to <b>Table 10-4 MCU Sub-System Clock,</b> <b>Table 10-5 MCU Sub-System Bus Clocks,</b> <b>Table 10-6 MCU Sub-System Peripheral Clocks (ASC0, ASC1, SCC, PCL, IIC),</b> and <b>Table 10-7 MCU Sub-System Peripheral Clocks (CAPCOM, GPT)</b>
<b>Page 668</b>	Updated <b>Figure 10-11 Clock and Enable Generation for MCU Sub-System</b>
Page 871	Updated <b>Figure 10-19 Output Clock Enables</b> WS00006993

<b>History</b>	
<b>Page 659</b>	Updated <b>Section 10.4.1.2 Operation Description</b> WS00007126
<b>Page 653</b>	Updated <b>Section 10.4.1 Clock Generation Unit</b> WS00007126
<b>Page 669</b>	Removed the Figure <b>DSP Clock Multiplexer</b> WS00007317
<b>Page 671</b>	Added <b>Table 10-9 Analog Clock Generation</b> replacing figure with the same name WS00007317
<b>Page 672</b>	Added <b>Table 10-11 AFC Clock Generation</b> replacing figure with the same name WS00007317
<b>Page 672</b>	Added <b>Table 10-12 Output Clock</b> replacing Figure <b>Output Clock Enables</b> WS00007317
<b>Page 655</b>	Added <b>Figure 10-8 Clock Generation Unit Scheme</b> WS00007317
<b>Page 668</b>	Updated <b>Figure 10-11 Clock and Enable Generation for MCU Sub-System</b> WS00007317
Changes for Rev. 1.03	
<b>Page 703</b> <b>Page 683</b>	Updated <b>Section 10.4.2.4 Setup</b> and register <b>RST_CTRL_STA</b> WS00007593
<b>Page 709</b> <b>Page 712</b>	Updated <b>Section 10.4.2.5.7 Standby Clock Reference Input Register</b> and <b>Section 10.4.2.5.11 High Pre-Wakeup and Wait Registers</b> WS00007742
Changes for Rev. 1.04	
<b>Page 714</b>	Registers <b>SCCUCLKSTA</b> and <b>SCCUSMSTA</b> removed from Internal Text WS00008015
<b>Page 696</b>	Updated <b>Figure 10-18 Timing of Wake-Up Phase</b> WS00007724
<b>Page 675</b>	Removed reference to ECO block in <b>MST_CLK_CTRL.AFC32KEN</b> register WS00008455
<b>Page 653</b> <b>Page 655</b> <b>Page 656</b>	Updated incorrect speed, <b>Figure 10-8 Clock Generation Unit Scheme</b> , and <b>Figure 10-9 Overview of E-GOLDradio CPU Clock Generation System</b> in <b>Section 10.4.1 Clock Generation Unit</b> WS00008670
<b>Page 657</b>	Updated <b>Section 10.4.1.1 GSM/GPRS Baseband System Clock Concept Description</b> WS00008670
<b>Page 661</b>	Note added to <b>Section 10.4.1.2.4 Changing the MCU Sub-System Frequencies During Operation</b> WS00008670
<b>Page 685</b>	Updated <b>Section 10.4.2.1.2 System Clock Control</b> WS00008011
Changes for Rev. 1.05	
<b>Page 659</b>	Removed Step 7 and Note in <b>Section 10.4.1.2.1 Startup on External Reset</b> WS00008973
<b>Page 673</b>	Removed <b>Section 10.4.1.5 Locked Interrupt</b>



History	
Changes for Rev. 1.06	
<b>Page 706</b>	Updated <b>SCCUSPCR</b> WS00009046

### System Integration:

- Supply domain:
  - VDD\_PLL for the PLL block
  - VDD\_MAIN for the CGU and SCCU blocks
- Chip internal interfaces:
  - Clock domain: Refer to **Section 10.4.1.1 GSM/GPRS Baseband System Clock Concept Description (on Page 657)**
- Interrupt sources:
- Monitor Pins: Refer to **Section 11.8.10 Internal Signal Monitoring (on Page 1209)**

### 10.4.1 Clock Generation Unit

The E-GOLDRadio integrates a power full clocking scheme which facilitates flexibility during normal operation mode combined with minimized power dissipation during standby and sleep mode. Two separate clock inputs are provided, a 26 MHz low swing input and a 32.768 kHz square wave input. The 32.768 kHz clock can also be generated by an one chip oscillator.

The low swing input F26M (XO) passes a clock shaper first. This shaped clock or the 32.768 kHz may be used directly for clocking E-GOLDRadio in power saving mode. During normal operation a PLL with 1 phase shifter generate different clocks from the shaped 26 MHz clock. The principle relations between the frequencies of the input clock, the PLL clock and the phase shifter clock are:

$$F_{pll} = finputx \times \frac{m}{n} \quad [0.18]$$

$$F_{phaseshifter} = \frac{F_{pll} \times 12}{k} \quad [0.19]$$

Where m,n,K are programmable integer. For the E-GOLDRadio application,  $F_{pll}$  must always be set to 208 MHz (that is,  $clk\_pll = 104$  MHz).

The PLL (phased locked loop) macro includes 1 phase shifter. The PLL or phase shifter clocks as well as the input clock and the 32.768 kHz clock are distributed via synchronous multiplexers to the master clock independent for each master clock.

From the master clocks additional clocks for different peripherals and buses are derived. In addition, the peripherals allow frequency reduction and clock disabling.

The principal structure of the generation logic is shown in [Figure 10-8](#).

Most, but not all, of the logic shown is implemented in the clock generation unit CGU. An exception is the 32 kHz oscillator, which is implemented together with the RTC in the RTC voltage domain. Also the DSP peripherals clocks are generated locally in the DSP sub-system.

CONFIDENTIAL

**Figure 10-8 Clock Generation Unit Scheme**

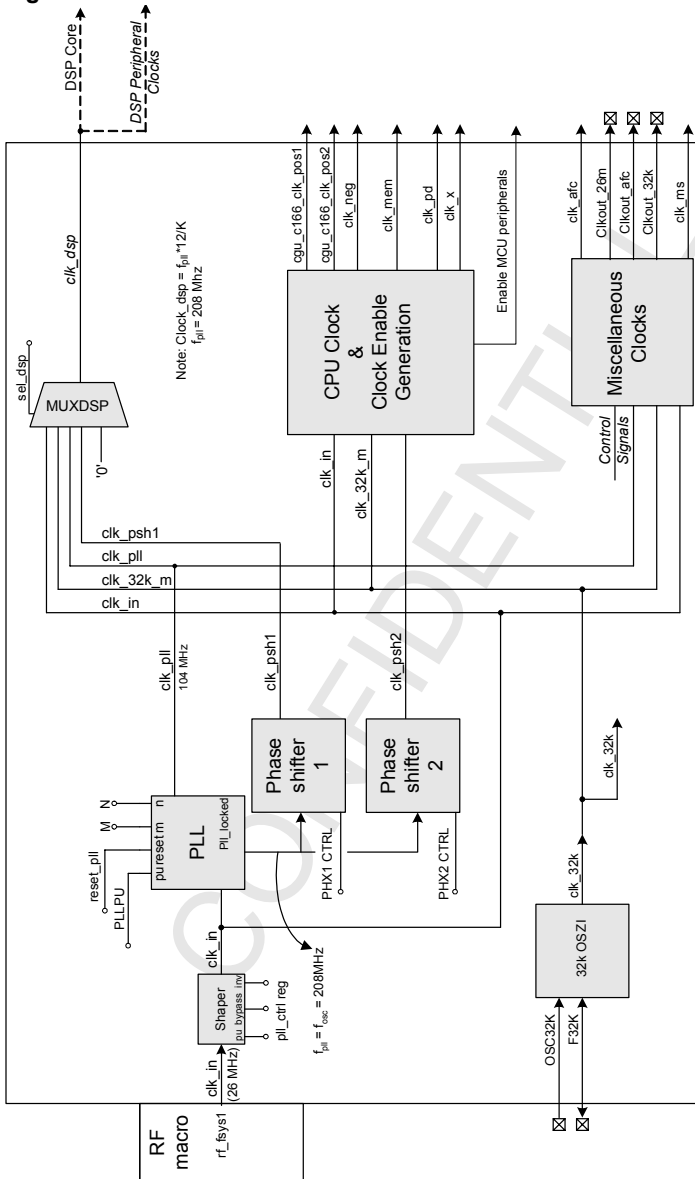
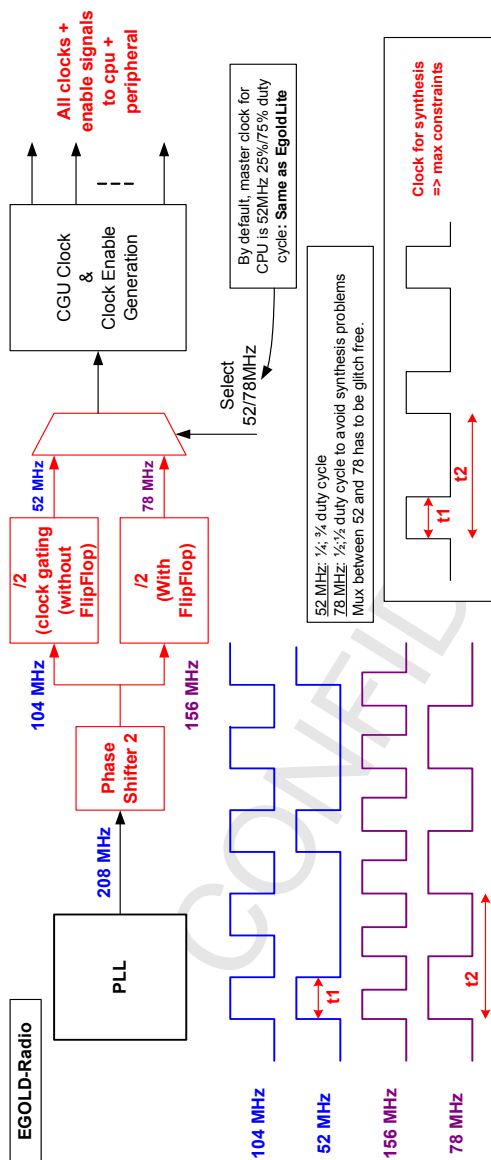


Figure 10-9 Overview of E-GOLDradio CPU Clock Generation System



#### 10.4.1.1 GSM/GPRS Baseband System Clock Concept Description

The GSM/GPRS Baseband System divides into several clock subsystems which can operate asynchronously to each other. Each clock sub-system is driven by a master clock. DSP clock sub-system may operate at a lower frequencies. In this case, the lower frequency clock is derived from the respective master clock by an integer divider or a fractional divider and is synchronous with the master clock.

The clock sub-systems are:

- PLL frequency clock system  
is driven by the PLL output `clk_pll`. The `clk_pll` runs at 104 MHz.
- The real time clock RTC  
the RTC clock `clk_32K` runs at 32 kHz.
- The controller system  
includes the C166s with its peripherals. The master `clk_cpu_master` runs at 52 MHz (or 78 MHz, depends on `phx2` configuration) in the normal mode (it could be 26 MHz or 32 kHz). It is derived from the `clk_phx2` (104 MHz or 156 MHz in normal mode) because of the division by 2. No other secondary phase shifter frequency is allowed. The clocks and enables are generated internally in the CGU.  
To save power, use the 26 MHz clock from the shaper or the 32 kHz clock from the RTC.
- The DSP sub-system  
includes the DSP core and its peripherals. The master `clk_dsp` is coming from a Phase shifter. The target frequency is 104 MHz. Lower frequency are configurable. However, due to GSM system requirements, system functions are not guaranteed at lower frequencies.
- The Analog macro  
receives its clocks from some peripherals, such as the DSP sub-system, the GSM interface, or the measurement unit.
- The AFC block  
can be turned on or off by enabling/disabling the AFC clock with **`MST_CLK_CTRL.AFCEN`**. A 26 MHz clock is required for regular AFC operation. The AFC block can alternatively run with 32 kHz clock during the sleep mode period when setting the **`MST_CLK_CTRL.AFC32KEN`** bit.

**CONFIDENTIAL**

**PLL - CGU - SCCU**

- The interfaces between the subsystems support asynchronous clocks. These interfaces are:
  - Controller sub-system – DSP sub-system  
One asynchronous shared memory interfaces between the X-Bus and the data buses of the DSP  
Interrupt lines between the controller system and the DSP sub-system
  - Controller system – RTC, AFC, Keypad  
Asynchronous interface within the peripherals
  - DSP System – Analog part of GSM  
Asynchronous interfaces between the mixed signals blocks and the DSP buses

The following parts of E-GOLDRadio normally run at different frequencies:

- MCU sub system
- DSP sub system
- RTC
- TCU
- Analog macro.

**Table 10-2** shows the target operating frequencies for the different parts of E-GOLDRadio.

**Table 10-2 Target Operating Frequencies for E-GOLDRadio**

<b>BLOCK</b>	<b>TARGET FREQUENCY</b>
MCU Sub-System	78 MHz
DSP	104 MHz
Shared Ram	104 Mhz
RTC	78 MHz
TCU	10 MHz
ANALOG	104 MHz
PLL MACRO	104 MHz

### 10.4.1.2 Operation Description

The internal RF oscillator provides the major 26 MHz reference clock for E-GOLDRadio via the XO pad.

An external VCXO may be connected to the E-GOLDRadio via pins XO and XOX.

The input clock first passes through a shaper, which keeps a low swing input level on XO. The shaper output is `clk_in`. The shaper can be powered up with a delay related to the VCXO power up. This might slightly reduce the overall power consumption.

The VCXO can operate in an AFC loop in a synchronous mode. The AFC block can be used to generate a control voltage for the external VCXO.

The following subsection describes different scenarios for starting or re-configuration of the clock system.

#### 10.4.1.2.1 Startup on External Reset

Upon reset of the MCU, all internal clocks are supplied with the input clock `clk_in`, then:

1. The PLL is powered up automatically and starts generating `clk_pll` with 4 times the frequency of `clk_in`. That is, `clk_pll` runs at 104 MHz because `clk_in` is running at 26 MHz.
2. The phase shifter is switched off.
3. The SW must configure the PLL (if needed) and the phase shifter as desired for the application.
4. The `clk_pll` must always be configured for 104 MHz.
5. After setting the PLL parameters, the primary phase shifter and the secondary phase shifter will reach their target frequencies within a maximum of 100  $\mu$ s. The two frequency targets for the secondary phase shifter are:
  - 104 MHz for a 52 MHz master clock
  - 156 MHz for a 78 MHz master clock.
6. When PLL has reached its target frequency, it activates an internal 'lock', which is latched afterwards as a register bit.

Thus, the SW has two options to find a suitable point of time for switching from `clk_in` to the PLL and phase shifter clocks. Perform either:

1. Use a 100  $\mu$ s timer
2. Poll the bit `PLL_CTRL.PLL_locked`.

CONFIDENTIAL

PLL - CGU - SCCU

When the PLL is locked, the `clk_pll` and phase shifter clocks may be used as master clocks for the other clock subsystems of E-GOLDRadio by:

1. Deactivating the external INIT reset.
2. Setting M and N PLL factors via `PLL_CTRL.Pll_M` and `PLL_CTRL.Pll_N`.  
PLL\_powerup is already activated.
3. If the phase shifter is used, set phase shifter factors X1 and Y1 via `PHX_CTRL.Phs_X` and `PHX_CTRL.Phs_Y`.
4. Setting the phase shifter power up, if needed, via `PLL_CTRL.Pll_powerup`.
5. Waiting for the PLL to lock-on.
6. When locked-on, the SW can set `PLL_CTRL.Pll_bypass_n`.
7. the SW can set `PHX2_CTRL.phs2_bypass_n` to use the normal mode (master clock to 52 MHz or 78 MHz)
8. If the phase shifter is used, the SW can set `PHX_CTRL.Phs1_bypass_n`.
9. Setting the select and divider field as desired.

The clock multiplexers deliver the new clock 4 cycles in the old clock domain plus 4 cycles in the new clock domain after changing the select field.

#### 10.4.1.2.2 Changing the Phase Shifter Frequency

Before changing the phase shifter frequency, it is mandatory to disconnect all clocking systems connected to that source via `MST_CLK_CTRL.DSP_SEL = 100B`. The frequency of the phase shifter can only be changed only while the clock is not used by the systems.

To change the phase shifter frequency:

1. Set `PHX_CTRL.Phs1_powerup` and `PHX_CTRL.Phs1_bypass_n` to 0
2. Set `PHX_CTRL.Phs_X` and `PHX_CTRL.Phs_Y` to the new values
3. Set `PHX_CTRL.Phs1_powerup` and `PHX_CTRL.Phs1_bypass_n` to 1
4. Wait for 1  $\mu$ s before using the clock in the system.

#### 10.4.1.2.3 Changing Secondary Phase Shifter Frequency (MCU Phase Shifter)

Before changing the secondary phase shifter frequency, it is mandatory to change the master clock to 26 MHz `MST_CLK_CTRL.CPUH = 0`:

1. Reset `MST_CLK_CTRL.CPUH`
2. Set `PHX_CTRL.Phs2_X` and `PHX_CTRL.Phs2_Y` to the new value
3. Set `MST_CLK_CTRL.CPUH = 1`.

After these operations, the master clock is 52 MHz or 78 MHz.



#### 10.4.1.2.4 Changing the MCU Sub-System Frequencies During Operation

The controller clock source can always be changed during operation.

*Note: The duty cycle of the CCLK (SIM card clock) signal is impacted when the following bits are updated:*

- [MST\\_CLK\\_CTRL.CPUH](#)
- [MST\\_CLK\\_CTRL.CPUPRE](#).

#### 10.4.1.2.5 Using the 26 MHz Input Clock

In some situations it may be sufficient to clock the device with the 26 MHz input clock `clk_in` and to switch off the PLL to save power. The SW has to connect each master clock to `clk_in` by appropriate programming of the selection fields in [MST\\_CLK\\_CTRL](#). Peripherals which use the bus clock may have to be reprogrammed to compensate for the frequency change.

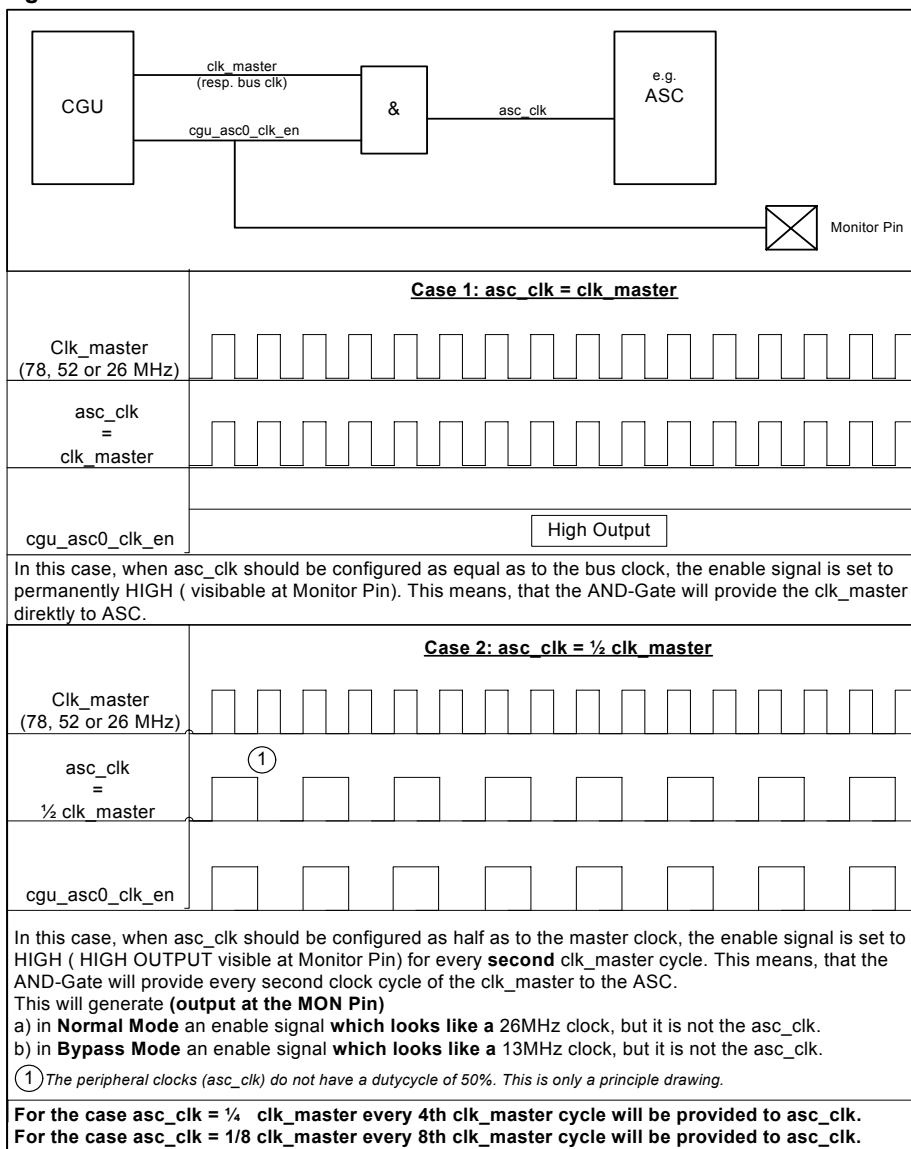
Two blocks, the GSM System Interface and the System Timer, have an automatic clock frequency compensation controlled by MCU signal

#### 10.4.1.3 Sub-System Clocks and Enables

A sub-system includes a bus and its peripherals. The bus interface is always clocked with its respective bus clock. For the peripherals, it depends on which sub-system they are integrated. For the DSP sub-system the clock generation is made in the DSP part (refer to [Chapter 5 TEAKlite DSP \(on Page 69\)](#)). For the MCU sub-system the clocks and enables are generated internally of the clock generation unit CGU with the corresponding enable via the `clock_cpu_master`.

[Figure 10-10](#) shows an example how the clocks enable works.

Figure 10-10 Clock Enable



CONFIDENTIAL

PLL - CGU - SCCU

#### 10.4.1.3.1 MCU System Clocks

**Table 10-3** shows configurations available for the MCU system.

**Table 10-3 MCU Clock Master**

Mode	CLK_PHX2_CTRL	MST_CLK_CTRL.CPUH	Clk_master
Reset	X	X	26 MHz
Power saving	X	X	32 KHz
PHX2 bypass	X	0	26 MHz
Normal 52MHz	0206 <sub>H</sub>	1	52 MHz
Normal 78MHz	2106 <sub>H</sub>	1	78 MHz

Notes:

- The 26 MHz clock source is taken after the shaper from the PLL Macro.
- The 32 KHz clock source comes from the Pad oscillator.
- The 52 MHz or 78 MHz clock source comes from the secondary phase shifter (104 MHz or 156 MHz) with a division by 2.

**Table 10-4 MCU Sub-System Clock**

Mode	MST_CLK_CTRL. CPUH	Clk_master	MST_CLK_CTRL. CPUPRE	Cpu_clk_pos Coefficient <sup>1)</sup>	Cpu_clk_neg Coefficient <sup>1)</sup>
Reset	X	x	X	0	0
Power saving	X	32 KHz	X	1	0
Normal 52 MHz	1	52 MHz	000	1	1
			001	1/2	1/2
			010	1/4	1/4
			011	1/8	1/8
			100	1/16	1/16
			101	1/32	1/32
			110	1/64	1/64
			111	1/128	1/128
Normal 78 MHz	1	78 MHz	000	1	1
			001	1/2	1/2
			010	1/4	1/4
			011	1/8	1/8
			100	1/16	1/16
			101	1/32	1/32
			110	1/64	1/64
			111	1/128	1/128
Bypass	0	26 MHz	000	1	1
			001	1/2	1/2
			010	1/4	1/4
			011	1/8	1/8
			100	1/16	1/16
			101	1/32	1/32
			110	1/64	1/64
			111	1/128	1/128

<sup>1)</sup> The real clock is calculated as follows: Clk\_master x Clock Coefficient.

*Note: The fraction value indicate the division frequency applied on the clock master.  
Normal mode "cpu\_pre = 001" => cpu\_clk\_pos frequency (52 MHz \* 1/2) = 26 MHz*

**Table 10-5 MCU Sub-System Bus Clocks**

Mode	MST_CLK_CTRL .Cpuh	Clk_master	SYSCON1. PDCLKDIV	PD-Bus clock Coefficient <sup>1)</sup>	Xper clock Coefficient <sup>1)</sup>
Reset	x	x	X	0	0
Power saving	x	32 KHz	X	1	1
Normal 52 MHz	1	52 MHz	000	1	1/4
			001	1/2	1/4
			010	1/4	1/4
			011	1/8	1/4
			100	1/16	1/4
Normal 78 MHz	1	78 MHz	000	1	1/6
			001	1/2	1/6
			010	1/4	1/6
			011	1/8	1/6
			100	1/16	1/6
PLL bypass	0	26 MHz	000	1	1/2
			001	1/2	1/2
			010	1/4	1/2
			011	1/8	1/2
			100	1/16	1/2

<sup>1)</sup> The real clock is calculated as follows: Clk\_master x Clock Coefficient.

**Table 10-6 MCU Sub-System Peripheral Clocks (ASC0, ASC1, SCC, PCL, IIC)**

Mode	MST_CLK_CTRL.CPUH	Clk_master	SIFCLKS.(perif)CL	Peripheral clock coefficient <sup>1)</sup>
Reset	x	x	X	0
Power saving	x	32 KHz	X	1
Normal 52MHz	1	52 MHz	11	1
			10	1/2
			01	1/4
			00	1/8
Normal 78MHz	1	78MHz	11	1
			10	1/2
			01	1/4
			00	1/8
PLL bypass	0	26 MHz	11	1
			10	1
			01	1/2
			00	1/4

<sup>1)</sup> The real clock is calculated as follows: Clk\_master x Clock Coefficient.

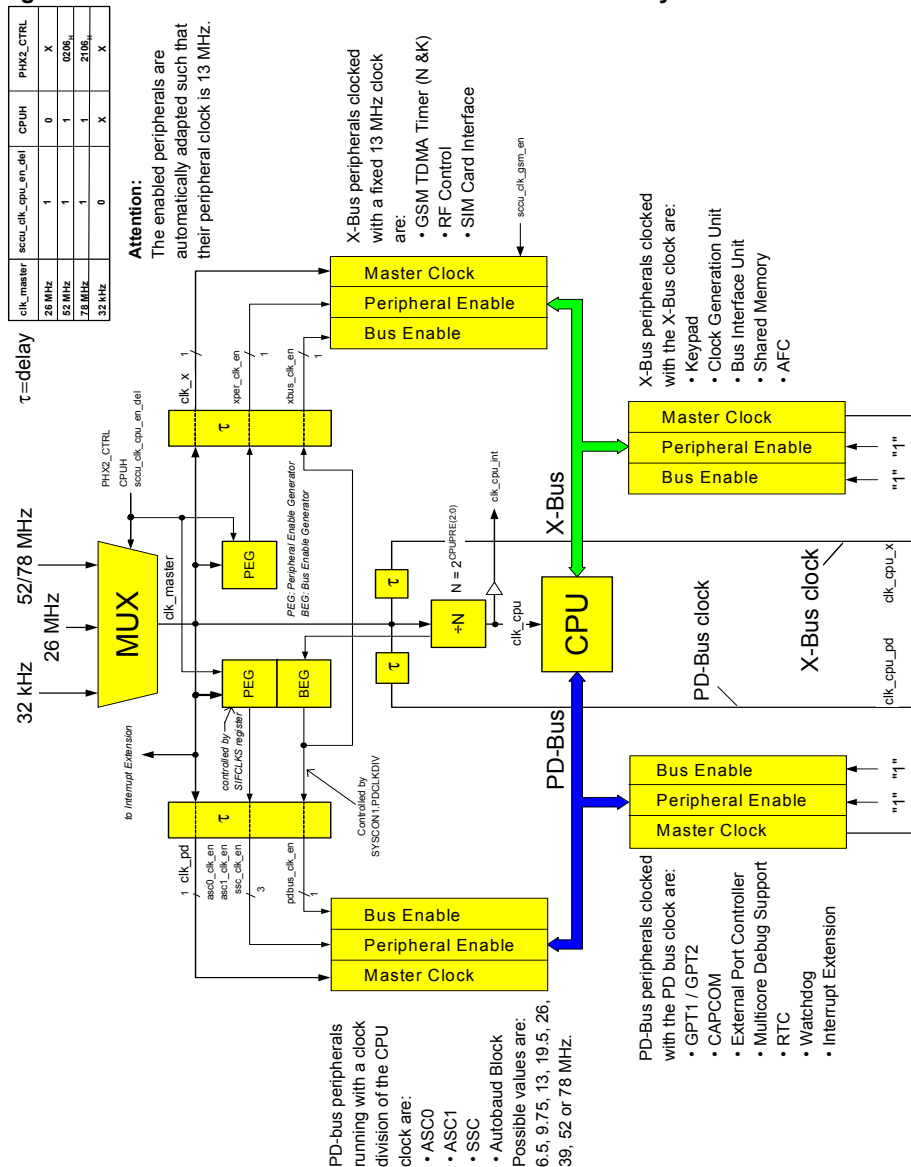
**Table 10-7 MCU Sub-System Peripheral Clocks (CAPCOM, GPT)**

Mode	MST_CLK_CTRL.CPUH	Clk_master	SIFCLKS.(perif)CL	Peripheral clock Coefficient <sup>1)</sup>
Reset	x	x	X	0
Power saving	x	32 KHz	x	1
Normal 52MHz	1	52 MHz	11	1/2
			10	1/4
			01	1/8
			00	1/16
Normal 78MHz	1	78 MHz	11	1/2
			10	1/4
			01	1/8
			00	1/16
PLL bypass	0	26 MHz	11	1
			10	1/2
			01	1/4
			00	1/8

<sup>1)</sup> The real clock is calculated as follows: Clk\_master x Clock Coefficient.

### 10.4.1.3.2 MCU System Clock

**Figure 10-11 Clock and Enable Generation for MCU Sub-System**





### 10.4.1.3.3 DSP Clock System

The DSP and its peripherals are clocked with the DSP master clock `clk_dsp`. The related mixed signals blocks are clocked with the `clk_pll`. All clock enabling and dividing is done locally in the peripherals. Details are given in the respective peripheral sections (see [Figure 10-12 Generation of Clocks for the DSP Section \(on Page 670\)](#)).

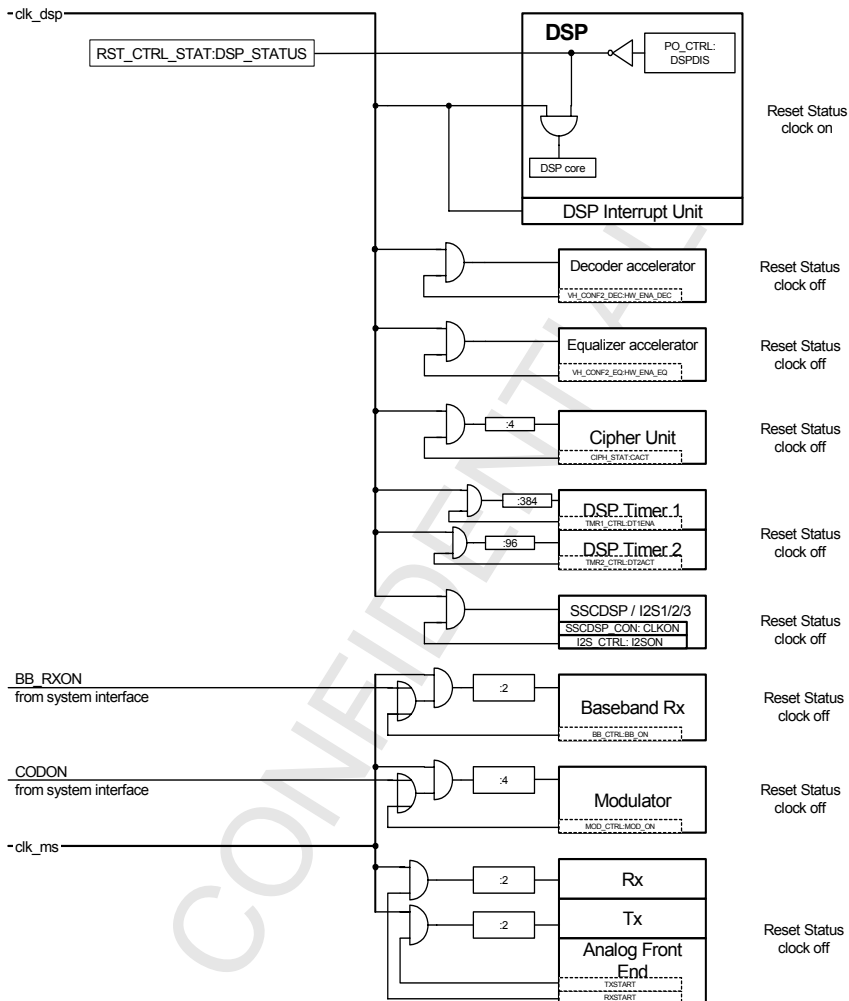
The controller may get information on the peripherals clock status via the shared memory and a dedicated command structure. The CGU provides different clocks source to the DSP via the `clk_dsp` signal. According the value of `MST_CLK_CTRL.DSP_SEL`, the `CLOCK_DSP` signal receives one of the clock sources shown in [Table 10-8](#).

**Table 10-8 DSP Clock Sources**

DSP_SEL	CLK_DSP	Frequency
000	<code>clk_in</code>	26 MHz
001	<code>clk_phs1</code>	104 MHz
010	<code>clk_pll</code>	Programmable (default: 104 MHz)
011	<code>clk_32K</code>	32 kHz
100	0	0
Other	<code>clk_dsp</code> clock fixed to 0 Hz	0

The test clock is provided when the device is in the test mode.

**Figure 10-12 Generation of Clocks for the DSP Section**



#### 10.4.1.3.4 Analog Clock

The master analog clock is derived from the `clk_pll_o` which runs at 104 MHz. It is called `clk_ms`, for mixed signal. The Analog master clock can be turned on or off by enabling/disabling the `clk_ms` with **MSB\_CLK\_CTRL.MSEN** bits. `Clk_ms` can also receive the clock after the shaper called `CLK_IN`.

**CONFIDENTIAL**

**PLL - CGU - SCCU**

Clk\_ms does not go directly to the Analog macro, it goes through several peripherals to generate several sub-clocks.

**Table 10-9 Analog Clock Generation**

<b>MST_CLK_CTRL.MSEN</b>	<b>SCCU signal <i>sccu_clk_cpu_en_del</i> (see <a href="#">Page 697</a>)</b>	<b>Clk_ms switched to:</b>
	0	Clock not enabled
00	1	Clk_in (26 MHz)
01	1	Clock not enabled
10	1	Clk_pll (104 MHz or 26 MHz, according the value of <a href="#">PLL_CTRL.PLL_bypass_n</a> )
11	1	Clock not enabled

There are several analog clocks. They come from different peripherals as given in [Table 10-10](#).

**Table 10-10 The Analog Clocks**

<b>Analog Clock</b>	<b>Frequency</b>
clk_vbrx	4 MHz (has to have a low jitter)
clk_vbtx	2 MHz
clk_vbtx_dith	2 MHz/10
clk_kernel	8 MHz
clk_meas	1 MHz
clk_par	6,5 MHz
clk_pa2	6,5 MHz
clk_pa1	6,5 MHz
clk_bbtX	13 MHz
clk_bbrx	Now 26 MHz
clk_bbrx_dith	13 MHz/12 (1.08 MHz: 50% duty cycle)

#### **10.4.1.3.5 AFC Clock**

The AFC block can be turned on or off by enabling the AFC clock with [MST\\_CLK\\_CTRL.AFCEN](#). A 26 MHz clock is required for regular AFC operation. The AFC block can alternatively run with a 32 kHz clock during the sleep mode period by

**CONFIDENTIAL**

**PLL - CGU - SCCU**

setting **MST\_CLK\_CTRL.AFC32KEN** bit. **Table 10-11** shows how to control the AFC clock, **clk\_afc\_o**.

**Table 10-11 AFC Clock Generation**

Mode	2 SCCU Signals <sup>1)</sup> : sccu_clk_gsm_en sccu_clk_cpu_en	AFCEN	AFC32KEN	clk_afc_o
<b>Normal</b>	01	0	Don't care	0
	10	1	Don't care	26 MHz
	11			
<b>Standby</b>	00	1	0	0
		1	1	32 kHz
		0	Don't care	0

<sup>1)</sup> Refer to **Table 10-16 SCCU Signal Descriptions (on Page 697)**.

#### 10.4.1.3.6 Output Clock

The E-GOLDRadio provides three different clocks for external devices (see **Table 10-12**). These three independent outputs allow external access to:

- clk\_in -> CLKOUT\_26
- clk\_32k -> CLKOUT\_32K
- clk\_out -> CLKOUT.

**Table 10-12 Output Clock**

<b>MST_CLK_CTRL.CLK_OUTL_EN</b> (Output Clock Enables)	Output Clocks
Bit 8	0 Disables 1 Enables CLKOUT_26
Bit 9	0 Disables 1 Enables CLKOUT_32K
Bits 11:10	00 No clocks applied on output CLK_OUT 01 Enables clk_pdbus 10 Enables clk_xbus 11 Enables clk_cpu_master

#### 10.4.1.4 Standby Mode Support

The standby mode is controlled by the SCCU block. In the standby mode the shaper is switched off and some clocks are switched to clk\_32k, controlled by signals from the SCCU. The other clocks have to be adjusted by SW before entering and after leaving the standby mode.

#### **10.4.1.5 Clock Control Register Descriptions**

The CGU is located on the X-bus. For the mapping of these registers refer to [Section 12.3 X-Bus Register Addresses \(on Page 1285\)](#).

CONFIDENTIAL

**CONFIDENTIAL**

**PLL - CGU - SCCU**

### 10.4.1.5.1 Clock Control Identification Register Real Time Clock

**CGUID**

**Clock Control Identification Register**

**Reset values: 4003<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>Module_ID</b>								<b>Revision_Number</b>							

Field	Bits	Type	Description
<b>Revision_Number</b>	7:0	r	<b>CGU Revision Number</b> These hard-wired bits are used for the CGU revision numbering.
<b>Module_ID</b>	15:8	r	<b>CGU Identification Number</b> These hard-wired bits are used for CGU identification numbering.

### Interface Enable

**RTCIF**

**Real Time Clock Interface Enable**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>								<b>RTCIFEN</b>							

Field	Bits	Type	Description
<b>RTCIFEN</b>	7:0	rw	<b>RTC Interface Enable Field</b> 10101010: Enables the RTC register interface access via the level shifter and clock enabling (required for register R/W operation)  Any other: Disables the RTC register interface access (greatly reduces vdd_RTC current consumption during CPU_not_in_idle)
<b>Reserved</b>	15:8	r	Reserved, these bits must be left at their reset values.

**CONFIDENTIAL**

**PLL - CGU - SCCU**

### 10.4.1.5.2 Master Clock Control Register

**MST\_CLK\_CTRL**

**Master Clock Control Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>DSP_SEL</b>			<b>CBUSH</b>	<b>CLK_OUTL_EN</b>			<b>AFC32KEN</b>	<b>AFCEN</b>	<b>MSEN</b>		<b>CPUPRE</b>			<b>CPUH</b>	

Field	Bits	Type	Description
<b>CPUH</b>	0	rw	<b>Enables 52 MHz or 78 Mhz Operation of the MCU and Serial Interfaces</b> The maximum clock for the MCU and serial interfaces is: 0    26 MHz 1    Normal mode: 52 MHz or 78 MHz (depends on PHX2_CTRL control)
<b>CPUPRE</b>	3:1	rw	<b>MCU Clock Prescaler Factor</b> Sets the divider factor of the MCU prescaler, <b>CPUPRE</b> = 2 <sup>n</sup> (n = 0..7)
<b>MSEN</b>	5:4	rw	<b>Mixed Signal Clock Enable</b> Clk_ms is switched to: 00   Clk_in (26 MHz) 01   Clock not enabled 10   Clk_pll (104 MHz or 26 MHz, according PLL bypass value) 11   Clock not enabled
<b>AFCEN</b>	6	rw	<b>AFC Enable</b> 26 MHz clock to AFC is switched: 0:   Off 1:   On
<b>AFC32KEN</b>	7	rw	<b>AFC Clock Enable during Stand-By Mode</b> 0:   Disables 1:   Enables

**CONFIDENTIAL**

**PLL - CGU - SCCU**

Field	Bits	Type	Description
<b>CLK_OUTL_EN</b>	11:8	rw	<b>Output Clocks Enables</b> Bit 8 0 Disables 1 Enables CLKOUT_26 Bit 9 0 Disables 1 Enables CLKOUT_32K Bits 11:10 00 No clocks applied on output CLK_OUT 01 Enables clk_pdbus 10 Enables clk_xbus 11 Enables clk_cpu_master
<b>CBUSH</b>	12	rw	<b>Update GSM Timer</b> 0 No update 1 Update value
<b>DSP_SEL</b>	15:13	rw	<b>DSP Mode Select</b> Selects the clock source for clk_dsp: 000 Clk_in clock selected (26 MHz) 001 Clk_phs1 clock selected (refer to <a href="#">Table 10-13 Phase Shifter Frequencies (on Page 679)</a> ) 010 Clk_pll clock selected (104 MHz) 011 clk_32k clock selected (32 kHz) 100 clk_dsp clock signal fixed to 0 others clk_dsp clock fixed to 0 Hz.  <i>Note: When in the Power Down Mode, clk_dsp is no longer fixed at 32 kHz, but is controlled by this bit field.</i>



**CONFIDENTIAL**

**PLL - CGU - SCCU**

### 10.4.1.5.3 PLL Control Register

**PLL\_CTRL**

**PLL Control Register**

**Reset value: 0711<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>PII_locked</b>	<b>PII_sh_invel</b>	<b>PLL_N</b>					<b>PLL_M</b>					<b>PII_sh_bypass_n</b>	<b>PII_sh_powerup</b>	<b>PII_bypass_n</b>	<b>PII_powerup</b>

Field	Bits	Type	Description	Reset
<b>PII_powerup</b>	0	rw	<b>Powers Up the Analog Part of the PLL</b> 0: Disables 1: Enables	<b>1</b>
<b>PII_bypass_n</b>	1	rw	<b>PLL Bypass Control</b> 0: PLL bypassed 1: PLL not bypassed  <i>Note: The PLL bypass control is effective immediately and does not depend on bit <b>PII_locked</b>.</i>	<b>0</b>
<b>PII_sh_powerup</b>	2	rw	<b>Enables Shaper Power Up Model</b> 0: Powers down the Shaper (pll_sh_clk_o is 1 or 0 depending on the value of <b>PII_sh_invel</b> ) 1: Powers up the Shaper	<b>0</b>
<b>PII_sh_bypass_n</b>	3	rw	<b>Enables Bypass Mode of the Shaper</b> 0: Enable $PLL\_sh\_clk\_0 = PLL\_clkref\_i$ Remark : no dependent of <b>PII_sh_invel</b> 1: Disable	<b>0</b>
<b>PLL_M</b>	7:4	rw	<b>Input Divider Factor M in the PLL</b>	<b>1</b>
<b>PLL_N</b>	13:8	rw	<b>Feedback Divider Factor N in the PLL</b> PLL acts as a frequency synthesizer following the equation: $F_{pll\_clkout} = \frac{(N + 1) \times F_{pllclkref}}{(M + 1)}$	<b>7</b>
<b>PII_sh_invel</b>	14	rw	<b>Inversion of the Shaper Output</b> 0: Disables 1: Enables	<b>0</b>

**CONFIDENTIAL**

**PLL - CGU - SCCU**

Field	Bits	Type	Description	Reset
<b>PII_locked</b>	15	r	<b>PLL Status</b> 0: PLL not locked 1: PLL locked	<b>0</b>

#### 10.4.1.5.4 Phase Shifter Control Register

**PHX\_CTRL**

**Phase Shifter Control Register**

**Reset value: 2200<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	Phs_X			Phs_Y			RESERVED			RESERVED		Phs1_bypass_n	Phs1_powerup	RESERVED	

Field	Bits	Type	Description	Reset
<b>Phs1_powerup</b>	1	rw	<b>Phase Shifter 1 Power Up Enable</b> 0: Disables PII_phs1_clk_o 1: Enables PII_phs1_clk_o	<b>0</b>
<b>Phs1_bypass_n</b>	2	rw	<b>Phase Shifter 1 Bypass Control</b> 0: PII_phs1_clk_o bypassed 1: PII_phs1_clk_o not bypassed <i>Note: The PII_phs1_clk_o control is effective immediately and does not depend on bit <a href="#">PLL_CTRL.PII_locked</a>.</i>	<b>0</b>
<b>Phs_Y</b>	11:8	rw	<b>Y Factor of Phase Shifter 1</b> Refer to <a href="#">Table 10-13</a> .	<b>2</b>
<b>Phs_X</b>	14:12	rw	<b>X Factor of Phase Shifter 1</b> Refer to <a href="#">Table 10-13</a> .	<b>2</b>
<b>Reserved</b>	0,3:4 7:5, 15	r	Reserved, these bits must be left at their reset values.	

**Table 10-13 Phase Shifter Frequencies**

<b>Frequency</b>	<b>Phs_X</b>	<b>Phs_Y</b>
48 MHz	2	4
52 MHz	0	4
62.4 MHz	2	3
69.3 MHz	0	3
78 MHz	4	2
89.1 MHz	2	2
96 MHz	1	2
113.5 MHz	5	1
124.8 MHz	4	1
138.7 MHz	3	1
156 MHz	2	1

CONFIDENTIAL

PLL - CGU - SCCU

#### 10.4.1.5.5 MCU phase Shifter 2 Control Register

PHX2\_CTRL

MCU Phase Shifter Control Register

Reset value: 0202<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	Phs2_X			Phs2_Y			RESERVED			RESERVED	Phs2_bypass_n	Phs2_powerup	RESERVED		

Field	Bits	Type	Description	Reset
Phs2_powerup	1	rw	<b>Phase Shifter 2 Power Up Enable</b> 0: Disables Pll_phs2_clk_o 1: Enables Pll_phs2_clk_o	1
Phs2_bypass_n	2	rw	<b>Phase Shifter 2 Bypass Control</b> 0: Pll_phs2_clk_o bypassed 1: Pll_phs2_clk_o not bypassed <i>Note: The Pll_phs2_clk_o control is effective immediately and does not depend on bit <a href="#">PLL_CTRL.Pll_locked</a>.</i>	0
Phs2_Y	11:8	rw	<b>Y Factor of Phase Shifter 2</b> Refer to <a href="#">Table 10-14</a> .	2
Phs2_X	14:12	rw	<b>X Factor of Phase Shifter 2</b> Refer to <a href="#">Table 10-14</a> .	0
Reserved	0, 3:4, 7:5, 15	r	Reserved, these bits must be left at their reset values.	

Table 10-14 Phase Shifter Frequencies

Frequency	Phs2_X	Phs2_Y
104 MHz	0	2
156 MHz	2	1
others	Do not use	Do not use

**CONFIDENTIAL**

**PLL - CGU - SCCU**

### 10.4.1.5.6 Serial Interface Clock Select Register

**SIFCLKS**

**Serial Interface Clock Select Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>PCL</b>		<b>GPTCL</b>		<b>CAPCOM CL2</b>		<b>CAPCOM CL1</b>		<b>SSCCL</b>		<b>IICCL</b>		<b>ASC1CL</b>		<b>ASC0CL</b>	

Field	Bits	Type	Description For CPUH = 1 clk_master = 52 MHz/ 78 MHz	For CPUH = 0
<b>ASC0CL</b>	1:0	rw	<b>ASC0 Clock Select</b> 00: 6.5 MHz / 9.75 MHz 01: 13 MHz / 19.5 MHz 10: 26 MHz / 39 MHz 11: 52 MHz / 78 MHz	6.5 MHz 13 MHz 26 MHz 26 MHz
<b>ASC1CL</b>	3:2	rw	<b>ASC1 Clock Select</b> 00: 6.5 MHz / 9.75 MHz 01: 13 MHz / 19.5 MHz 10: 26 MHz / 39 MHz 11: 52 MHz / 78 MHz	6.5 MHz 13 MHz 26 MHz 26 MHz
<b>IICCL</b>	5:4	rw	<b>IIC Clock Select</b> 00: 6.5 MHz / 9.75 MHz 01: 13 MHz / 19.5 MHz 10: 26 MHz / 39 MHz 11: 52 MHz / 78 MHz	6.5 MHz 13 MHz 26 MHz 26 MHz
<b>SSCCL</b>	7:6	rw	<b>SSC Clock Select</b> 00: 6.5 MHz / 9.75 MHz 01: 13 MHz / 19.5 MHz 10: 26 MHz / 39 MHz 11: 52 MHz / 78MHz	6.5 MHz 13 MHz 26 MHz 26 MHz
<b>CAPCOMCL1</b>	9:8	rw	<b>CAPCOM1 Clock Select</b> 00: 3.25 MHz / 4.875 MHz 01: 6.5 MHz / 9.75 MHz 10: 13 MHz / 19.5 MHz 11: 26 MHz / 39 MHz <i>Note: Maximum CAPCOM1 clock is clk_master/4.</i>	3.25 MHz 6.5 MHz 13 MHz 26 MHz

**CONFIDENTIAL**

**PLL - CGU - SCCU**

Field	Bits	Type	Description For CPUH = 1 clk_master = 52 MHz/ 78 MHz For CPUH = 0
<b>CAPCOMCL2</b>	11:10	rw	<b>CAPCOM2 Clock Select</b> 00: 3.25 MHz / 4.875 MHz      3.25 MHz 01: 6.5 MHz / 9.75 MHz      6.5 MHz 10: 13 MHz / 19.5 MHz      13 MHz 11: 26 MHz / 39 MHz      26 MHz <i>Note: Maximum CAPCOM2 clock is clk_master/4.</i>
<b>GPTCL</b>	13:12	rw	<b>GPT Clock Select</b> 00: 3.25 MHz / 4.875 MHz      3.25 MHz 01: 6.5 MHz / 9.75 MHz      6.5 MHz 10: 13 MHz / 19.5 MHz      13 MHz 11: 26 MHz / 39 MHz      26 MHz <i>Note: Maximum GPT clock value is clk_master/4.</i>
<b>PCL</b>	15:14	rw	<b>PCL Clock Select</b> 00: 6.5 MHz / 9.75 MHz      6.5 MHz 01: 13 MHz / 19.5 MHz      13 MHz 10: 26 MHz / 39 MHz      26 MHz 11: 52 MHz / 78 MHz      26 MHz

**CONFIDENTIAL**

**PLL - CGU - SCCU**

### 10.4.1.5.7 Reset Control and Status Register

**RST\_CTRL\_STA**

**Reset Control And Status Register**

**Reset value: 0008<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DSP STA	pll_p hs1_ corr_ res_ ync	pll_c lkref_ fail_ res_ ync	pll_p hs2_ corr_ res_ ync	RESERVED				SCCUCL		RTCCL		IMR AMR OMC L	DSP RE SET	SIM RE SET	RTC RE SET

Field	Bits	Type	Description
RTC_RESET	0	rw	<b>RTC Software Reset</b> 0: No action 1: Reset applied
SIM_RESET	1	rw	<b>SIM Software Reset</b> 0: No action 1: Reset applied
DSP_RESET	2	rw	<b>DSP Software Reset</b> 0: No action 1: Reset applied
LMRAMROMCL	3	rw	<b>LM RAM/ROM Wait State Control</b> If MCU clock = 52MHz ( <b>MST_CLK_CTRL.CPUH</b> = 1 and <b>MST_CLK_CTRL.CPUPRE</b> = 000), then on the LM RAM and ROM bus there is: 0: 0 wait states for data accesses <i>Note: This is not true for code fetching because bit <b>LMRAMROMCL</b> does not effect the code fetching speed.</i> 1: 1 wait state
RTCCL	5:4	rw	<b>RTC Clock Select</b> <b>clk_master = 52MHz/ 78MHz</b> <b>For CPUH = 1</b> 00: 6.5 MHz / 9.75 MHz 01: 13 MHz / 19.5 MHz 10: 26 MHz / 39 MHz 11: 52 MHz / 78 MHz <b>For CPUH = 0</b> 6.5 MHz 13 MHz 26 MHz 26 MHz





## **10.4.2 Standby Clock Control Unit (SCCU)**

### **10.4.2.1 Functional Overview**

The SCCU provides all control signals for the transitions from the normal mode to the standby clock mode and back.

#### **10.4.2.1.1 Clock Control Function**

The SCCU controls the timing of the transitions from normal clock mode to standby clock mode and back to normal clock mode. For timing critical parts of the device like the GSM timer unit the SCCU guarantees an accurate time synchronization after return from standby clock mode.

In standby clock mode basic wake-up functions are still active permitting fast return to active mode whenever required, for example, when a key on the keypad has been pressed. Output signal VCXO\_EN may be used to control the VCXO. It will be de-asserted whenever the VCXO is not needed.

The clock control functionality splits up into two parts, the GSM timer clock control and the system clock control.

#### **10.4.2.1.2 System Clock Control**

The system clock control function permits switching-over to the slow standby-clock for the major part of the system whenever no activity is required.

The system clock control function also controls the switching off of the external VCXO, which may be switched off if neither the system nor the GSM timer requires the VCXO any more.

Prior to return from the standby clock mode to normal clock operation the SCCU generates an external signal enabling the VCXO again. The pre-wakeup time interval between turn-on of the VCXO and return to normal clock operation is selectable in the range from 0 up to 252 RTC clock cycles. This corresponds to a maximum pre-wakeup time of 7.7 ms.

#### **10.4.2.1.3 GSM Timer Clock Control**

The counter of the GSM Timer Unit (refer to [Section 10.11 GSM Timer Unit \(on Page 803\)](#)) runs off a 2.166 MHz clock derived from the VCXO reference clock. Since this clock is not available in standby clock mode the GSM Timer has to be stopped during standby clock mode. To permit accurate re-synchronization with the GSM network after the end of the standby clock phase the GSM timer clock is stopped for an accurate integer multiple of the length of a GSM TDMA frame. Therefore, the GSM timer remains in synchronization with the network except for the TDMA frame number which has to be adjusted by software.

The duration of the GSM timer stop phase may be defined in two ways. It may be predefined by software. This is done by writing the number of TDMA frames to a control register. Alternatively it may be terminated by a hardware-controlled wake-up function triggered e.g. by a key being pressed on the keypad. Thereby generating an early termination of the sleep phase can be generated. Also with the early termination of the GSM timer stop phase the length of the stop phase is kept to an accurate multiple of a GSM TDMA frame.

To permit return from standby clock mode after an accurate integer multiple of TDMA frames the SCCU maintains an internal time scale based upon the RTC clock. Since the frequency tolerance of the RTC clock is much higher than that of the VCXO calibration of the RTC clock is required to achieve the targeted accuracy. This calibration has to be done before entering standby clock mode for the first time. It has to be repeated in regular intervals to account for a potential temperature drift of frequency between the RTC clock frequency.

#### **10.4.2.1.4 Wake-Up Functions**

Return from standby clock mode to normal operation (wake-up) may be triggered either by C166 software or by hardware functions.

##### **Software Triggered Wakeup**

By setting a control bit in the SCCU C166 may trigger a return from standby clock mode to normal operation (wake-up) if C166 is running off the standby clock. In this case C166 can trigger a return to normal clock mode whenever an interrupt occurs.

##### **Hardware Triggered Wakeup**

Besides that the SCCU permits hardware triggered wakeup from standby clock mode by some predefined sources. Wakeup may be triggered by one of the following interrupt sources:

1. Keypad event
2. SIM card interrupt
3. RTC interrupt
4. Interrupt on one of the external interrupts or one of the CAPCOM interrupt signals
5. Event on the interrupt lines (IRQ or FIR) from the interrupt control unit (ICU) to C166.

Each of these sources can be enabled individually by setting a bit in the control registers **SCCUHWWAKEUPH** and **SCCUHWWAKEUPL**. The interrupt sources from the peripherals are not affected by the peripheral internal enabling mechanism.

Enabling the direct hardware wake-up functions 1 to 4 permit the fastest wake-up. Using the alternate functions an external or CAPCOM interrupt may be used to trigger a wake-up on a rising or falling edge on most pins. This also applies to the serial interfaces. Using the interrupt from the ICU causes a delay of several clock cycles due to the internal

processing of the ICU. It permits wakeup from all interrupt sources which have been enabled.

Wakeup may also be triggered by the interrupt handling routine by C166. This introduces even more clock cycles of delay depending upon software.

#### 10.4.2.1.5 Other Functions

Besides the clock control functions the SCCU also controls functions to reduce the leakage current of the device.

#### 10.4.2.1.6 Power Control

The voltage regulators on the power management IC can be turned off during standby clock mode. For this purpose output signal VCXO\_EN is available on the device which is controlled by the internal signal *vcxo\_en* from the SCCU. This signal will be de-asserted in standby mode. The power management IC should be programmable to turn off the DSP supply voltage, the analog supply voltage and the VCXO supply voltage when output signal VCXO\_EN is de-asserted.

#### Reset Control

If VDD\_DSP and VDD\_ANA is switched off in standby power-down these blocks have to be reset upon power-up. For this purpose the SCCU has to be set up to generate the reset signals *SCCU\_resd* for the VDD\_DSP supply domain and *SCCU\_resa* for the analog supply domain. These signals are already asserted when the system is ready for standby power-down and are kept asserted until power (and clock) has been applied again. The corresponding bits **SCCUSPCR.DPDN** and **SCCUSPCR.APDN** have to be set (refer to [Table 10-15](#)) for that purpose as shown in [Figure 10-13](#). The controller will have to set these bits to their appropriate values prior to first entering standby power-down.

**Table 10-15 Setup of register **SCCUSPCR****

VDD_DSP	VDD_ANA	Register SCCUSPCR	
		DPDN	APDN
on	on	0	0
off	on	1	0
off	off	1	1
on	off	not allowed	

The reset signals are routed to the controlled blocks via the **SCU (on Page 231)**.

## DSP and Analog Supply Domain Interface Control

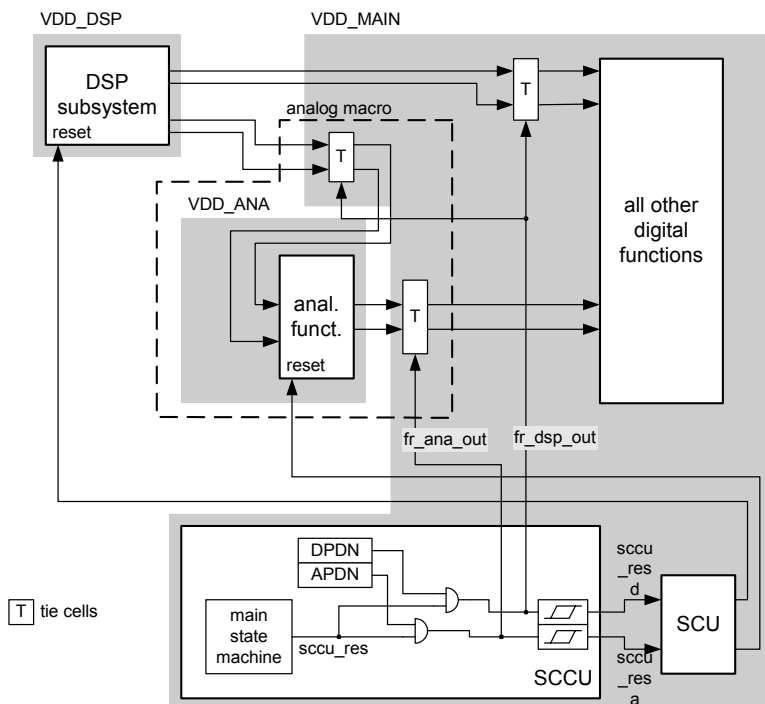
If the DSP subsystem and/or the analog supply voltage are switched off in standby clock mode all output signals from these domains to the other supply domains are tied to their reset values. Else floating levels on these outputs might cause short-circuit currents in the other parts of the circuit.

This is avoided by means of tie cells (T) as shown in [Figure 10-13](#). When enabled, the tie cells tie their output signal to a predefined value, else they are transparent to the signal. The tie cells are controlled by signals *fr\_dsp\_out* and *fr\_ana\_out*. These signals are activated one clock cycle before the reset is applied to the blocks to be powered down. The sequence is triggered by signal *SCCU\_res* from the SCCU main state machine.

With a delay of one real-time clock cycle after the tie signals have been asserted the reset signals for the digital part *SCCU\_resd* and of the analog part *SCCU\_resa* are asserted.

During standby power-down the VDD\_DSP supply domain is held in reset state. This is controlled by signal *SCCU\_resd* sent to the SCU. Signal *SCCU\_resd* is asserted one X-Bus bus clock cycle after signal *fr\_dsp\_out* to prevent generation of spikes at the inputs of the VDD\_MAIN domain. It is de-asserted simultaneously with signal *fr\_dsp\_out*.

Figure 10-13 Overview of Interface Control Functions



If the analog supply domain is powered down during standby power-down (bit APDN set) the analog block has to be reset upon power-up. This is done by signal *SCCU\_resa* sent to the SCU. Signal *SCCU\_resa* is asserted one X-Bus bus clock cycle after signal *fr\_ana\_out*. It is de-asserted simultaneously with signal *fr\_ana\_out*.

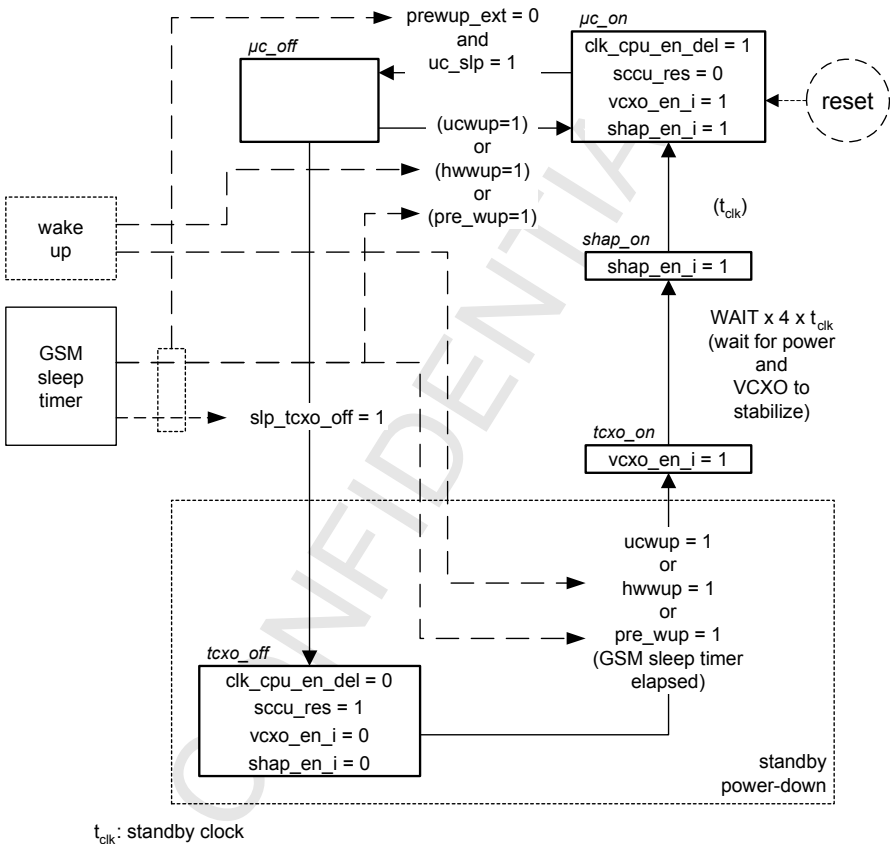
Before the analog supply domain is powered down all analog input pins have to be at low level. They have to be kept at low level during standby power-down.

If the DSP subsystem is selected to be switched off during the standby clock mode a reset signal is generated during this phase. After termination of the standby clock mode phase the DSP subsystem will then leave reset and re-boot.

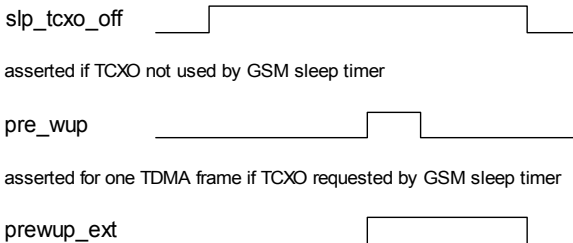
### 10.4.2.2 System Clock Control

The hardware controlled sequence of standby power-down states is shown in [Figure 10-14](#). The state machine is clocked with the standby clock from the RTC block. The function of the control signals from the GSM sleep timer is shown in [Figure 10-15](#).

**Figure 10-14 SCCU Main State Machine**



**Figure 10-15 Function of Signals from GSM Sleep Timer**



### Entering Standby Power-Down ( $\mu c\_on \rightarrow \mu c\_off$ )

After reset the main state will be in state  $\mu c\_on$ .

To enter standby power-down the controller will have to set bit **SCCUSCTRL.UCSLP**. The SCCU main state machine enters state  $\mu c\_off$  which signals that the reference clock is no longer required as system clock.

If the GSM sleep timer is already in the wake-up phase and a signal *prewup\_ext* is asserted, the transition is inhibited.

### Transition from state $\mu c\_off$ to state $\mu c\_on$

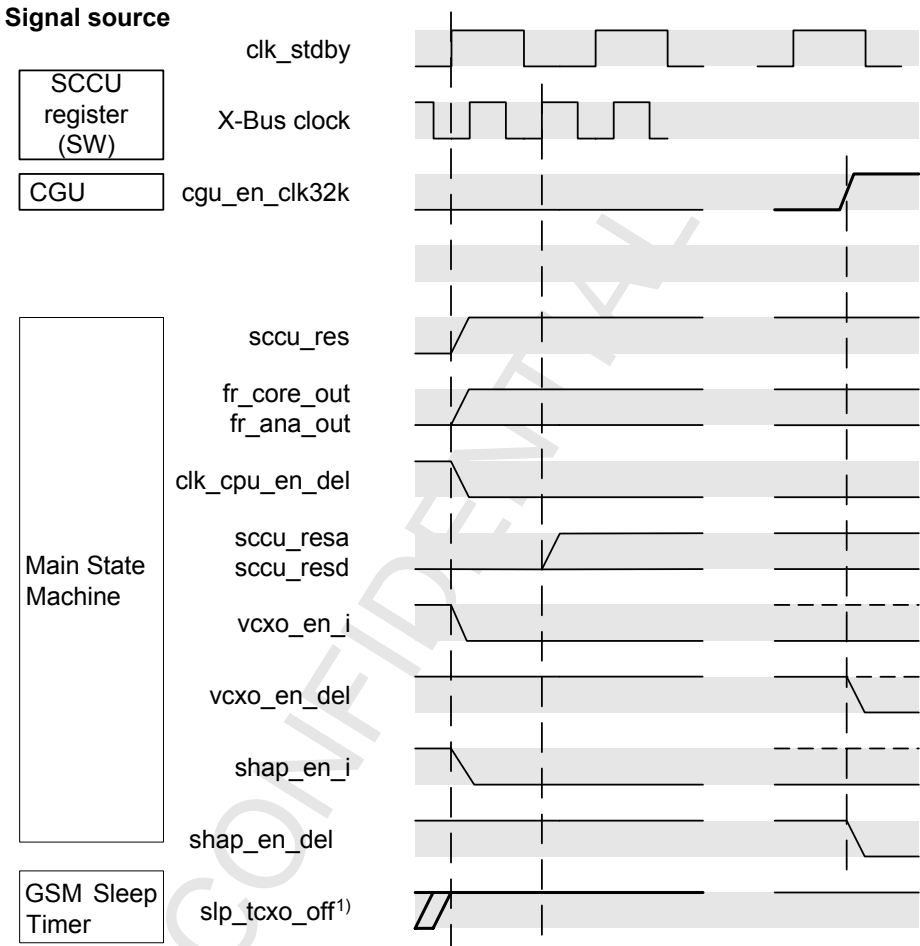
As long as the SCCU main state machine is in state  $\mu c\_off$ , the controller can abort the transition to standby power-down by setting bit **SCCUSCTRL.UCWUP**, which triggers the transition to state  $\mu c\_on$ .

The transition to state  $\mu c\_on$  is also triggered if the pre-wakeup phase is started by the GSM sleep timer (signalled by assertion of signal *pre\_wup*). This transition has priority over the transition to state *tcxo\_off*.

### Transition from state $\mu c\_off$ to state *tcxo\_off*

This transition is enabled only by an output signal from the GSM sleep timer. When the GSM sleep timer has frozen the GSM timer and runs off the standby clock, it asserts signal *slp\_tcxo\_off*. This triggers the transition to state *tcxo\_off* which triggers the transition to standby power-down mode as shown in **Figure 10-16**.

Figure 10-16 Timing Diagram for Transition  $\mu\text{c\_off} \rightarrow \text{tcxo\_off}$



1) Synchronized version of signal, it is delayed by one standby clock cycle in relation to original signal.

In state  $\text{tcxo\_off}$  the switch-over of the system clock ( $\text{clk\_tc\_m}$  from the reference clock from VCXO) to the standby clock (from the RTC) is started by de-asserting signal  $\text{clk\_cpu\_en\_del}$  to the clock generation unit (CGU). Also signals  $\text{vcxo\_en\_i}$  and  $\text{shap\_en\_i}$  are set to 0 thereby enabling switching off of the external VCXO and the shaper a few clock cycles later.



To prepare for a standby power-down (see [Figure 10-13 Overview of Interface Control Functions \(on Page 689\)](#)):

- Set bit **SCCUSPCR.DPDN** (signal *fr\_dsp\_out* is activated)
- Set bit **SCCUSPCR.APDN** (signal *fr\_ana\_out* is activated).

One X-Bus bus clock cycle later signals *SCCU\_resa* and *SCCU\_resd* are asserted.

The switch-over from the reference clock to the standby clock requires approximately 6 standby clock cycles. The completion of the switch-over is signaled by signal *cgu\_en\_clk32k* from the CGU being asserted. Only after the switch-over has been completed are signals *vcxo\_en* and *shap\_en\_del* (controlling the external power-management IC and the shaper in the CGU) set to 0. Depending upon the setup of the power management IC the power supply of the VCXO, the baseband DSP subsystem supply and the analog supply may be switched off by the de-assertion of output signal *vcxo\_en*.

### Transition from State *tcxo\_off* to State *tcxo\_on*

If the SCCU is in state *tcxo\_off*, wakeup can be triggered by several sources: the GSM sleep timer, the hardware wake-up functions or the controller (see [Figure 10-17](#)). If the GSM sleep timer of the SCCU elapses, signal *pre\_wup* is asserted. If the hardware wake-up function detects an event signal, *hwwup* is asserted if the corresponding bit in **SCCUHWWAKEUPH** or **SCCUHWWAKEUPL** is set. For example, for *hwwup* to be asserted both:

- A SIM interrupt must occur
- **SCCUHWWAKEUPL.SIM\_EN** must be set.

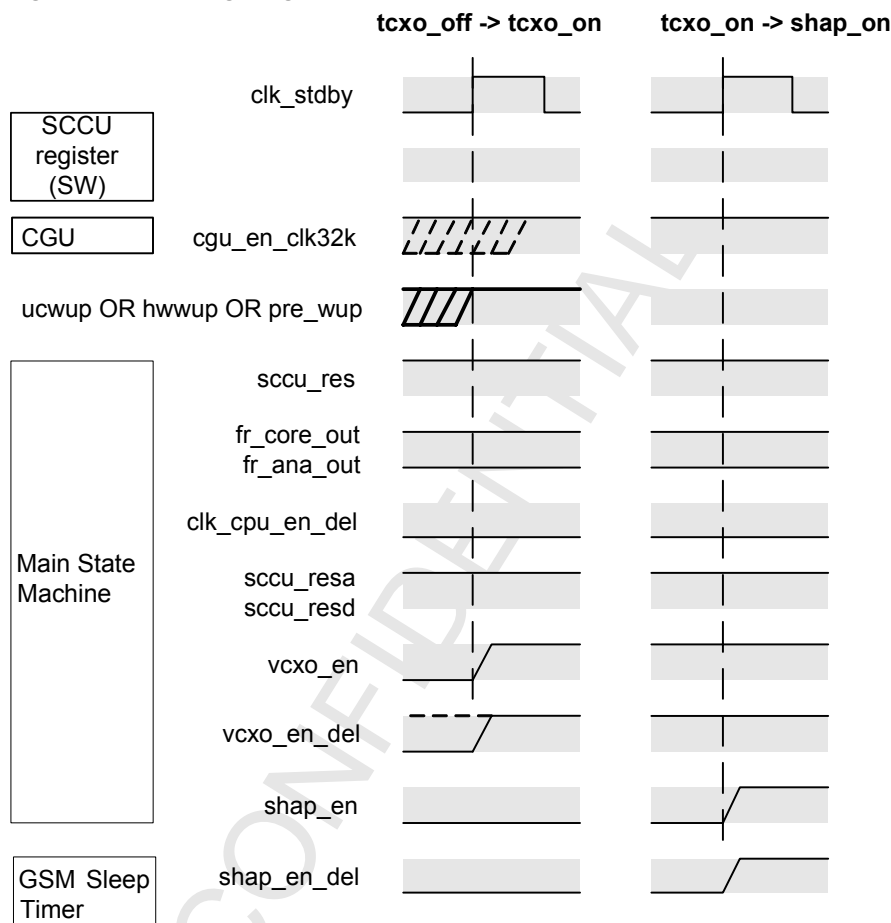
Any interrupt can trigger the controller to set bit UCWUP thereby asserting signal *ucwup*.

CONFIDENTIAL

CONFIDENTIAL

PLL - CGU - SCCU

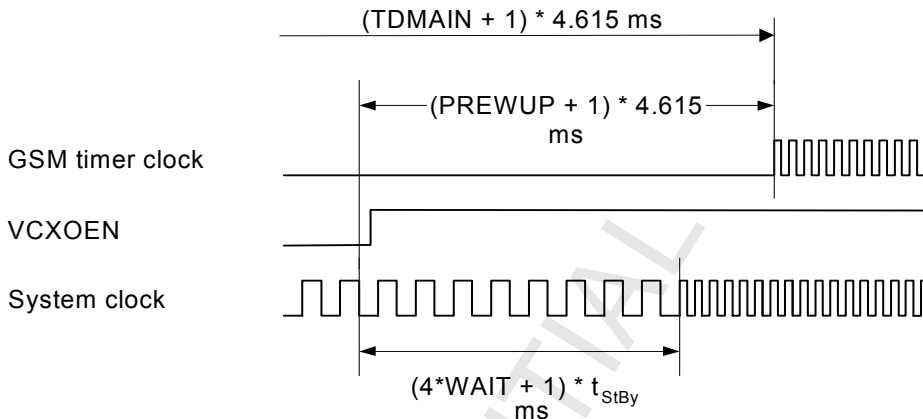
Figure 10-17 Timing Diagrams for Transitions  $tcxo\_off \rightarrow tcxo\_on \rightarrow shap\_on$



### Transition from state $tcxo\_on$ to state $\mu c\_on$

The transition from state  $tcxo\_on$  to state  $\mu c\_on$  occurs after a predefined time interval set by value in **SCCUWAITH.WAIT**. After  $4 * WAIT + 1$  cycles of the standby clock the system clock switches back to the VCXO clock as shown in **Figure 10-18**.

Figure 10-18 Timing of Wake-Up Phase



*Note: Always make sure that  $(PREWUP + 1) * t_{Frame}^{1)} > (4 * WAIT + 1) * t_{StBy}^{2)}$  because if, for example,  $PREWUP = 0$  and  $WAIT$  is at its maximum (63), the Pre-wakeup time in this case (4.615 ms) is not longer than the system clock wakeup time (7.7 ms), refer to the registers in [Section 10.4.2.5.11 High Pre-Wakeup and Wait Registers \(on Page 712\)](#).*

<sup>1)</sup>  $t_{Frame} = 4.615 \text{ ms}$

<sup>2)</sup>  $t_{StBy} = \left( \frac{1}{32768 \text{ Hz}} \right) \text{ sec.}$

**CONFIDENTIAL**

**PLL - CGU - SCCU**

**Table 10-16 SCCU Signal Descriptions**

Signal Name	I/O	From/to Block	Value	Function	
sccu_slp_tcxo_off	I	GSM sleep timer	1	VCXO clock no more required for GSM sleep timer - GSM timer frozen	
			0	VCXO clock still required by GSM sleep timer	
sccu_pre_wup	I		1	VCXO clock requested for preparation of end of GSM timer sleep period	
			0	VCXO clock not requested	
sccu_hwwup	I	Hardware		Wakeup signalling service request from hardware wakeup sources	
sccu_cgu_en_clk32k	I	CGU		Enables power-down of VCXO	
sccu_shap_en_del	O	Shaper		Enables shaper	
sccu_clk_cpu_en_del	O		1	Normal	Selects system clock clk_tc_m (except GSM timer)
			0	32 kHz standby	
sccu_clk_gsm_en	O		1	Clock supplied to GSM timer	
			0	No clock supplied to GSM timer	
sccu_vcxo_en_del	O	Power Mgmt. IC		Main power control signal also connected to pin VCXO_EN for control of power management IC	
sccu_fr_dsp_out	O	Tie cells		Ties the output level of all signals from the DSP supply domain	
sccu_fr_ana_out	O	Tie cells		Ties the output level of all signals from the analog supply domain	
sccu_resd	O			Resets DSP subsystem supply domain during standby power-down	
sccu_resa				Resets analog supply domain	

### 10.4.2.3 GSM Timer Control

#### 10.4.2.3.1 Synchronous Wake-Up Concept

To permit synchronous wake-up of the mobile after the end of the standby power-down phase, the GSM timer is frozen for a fixed time representing an accurate multiple  $n_{\text{TDMA}}$  of the TDMA frame duration  $t_{\text{TDMA}}$ . The duration of the freeze period is controlled by the GSM sleep timer of the SCCU. Since only the inaccurate standby clock is available during standby power-down this clock will have to be calibrated against the highly accurate clock of the 26 MHz system reference oscillator (VCXO) prior to entering standby power-down. A detailed description of the calibration can be found in [Section 10.4.2.1.3 GSM Timer Clock Control \(on Page 685\)](#). After this calibration the GSM timer freeze time interval can be accurately expressed as sum of an integer multiple  $n_{\text{stdby}}$  of the standby clock period  $t_{\text{stdby}}$  and an integer multiple  $n_{\text{ref}}$  of the reference clock period  $t_{\text{ref}}$ :

$$n_{\text{TDMA}} \cdot t_{\text{TDMA}} = n_{\text{stdby}} \cdot t_{\text{stdby}} + n_{\text{ref}} \cdot t_{\text{ref}} \quad [0.20]$$

Therefore, an accurate wake-up after the freeze time interval is guaranteed, it is only limited by the stability of the standby clock frequency over the freeze interval.

#### 10.4.2.3.2 GSM Sleep Timer

The GSM state machine freezes the GSM timer for a defined number of TDMA frames (5 to 8192 frames, that is, 0.023 to 37 s). After the end of the sleep period no correction to the GSM timer is required. Only the TDMA frame counters have to be updated.

To start the sleep function, the controller writes the number of TDMA frames diminished by 1 to register [SCCUTDMINL](#).

The GSM timer sleep phase is activated by setting bit [SCCUSLPCTRL.SLPEN](#).

Depending upon the value of bit [SCCUSLPCTRL.HWACTDI](#) there are two modes:

- 1 The sleep phase starts when bit **SLPEN** is set. Thereby the start of the sleep phase is purely software controlled.
- 0 The start of the sleep phase may be controlled by the rising edge of the GSM timer signal SLPSTART.
  - If signal SLPSTART is already high when bit **SLPEN** is set, the sleep phase starts immediately.
  - If signal SLPSTART is low when bit **SLPEN** is set, the sleep phase starts with the next rising edge of signal SLPSTART (hardware controlled).

After the end of the sleep phase the stand-by clock control block resets bit **SLPEN** to 0. To monitor the duration of the sleep phase, poll **SLPEN**.

Before the end of the sleep period the GSM state machine requests the VCXO reference clock by asserting signal `pre_wakeup` (see [Figure 10-19](#)). If the VCXO has been

CONFIDENTIAL

PLL - CGU - SCCU

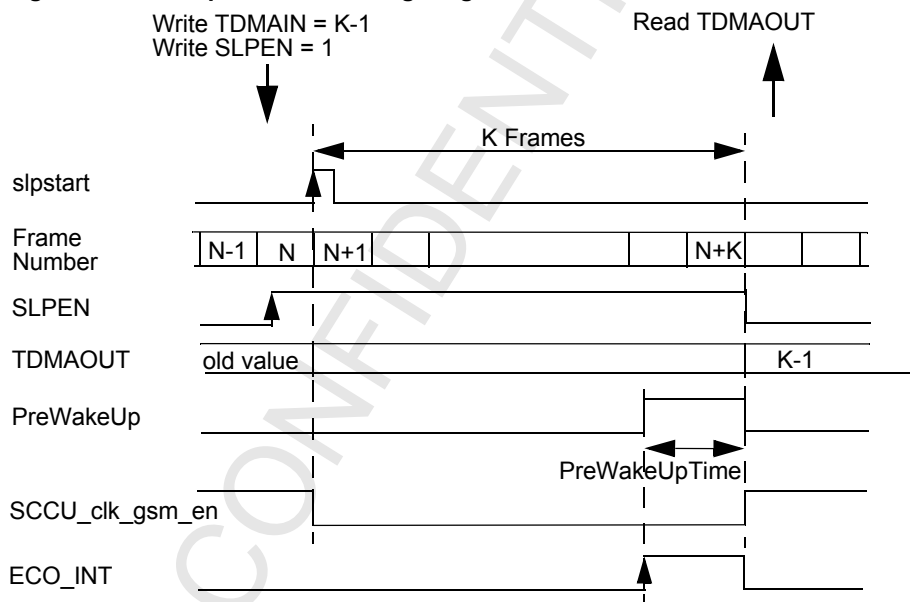
disabled, the number of TDMA frames before the end of the sleep period is defined by the value in **SCCUWAITH.PREWUP**.

The SCCU can signal to the controller the end of the sleep phase by asserting interrupt signal ECO\_INT(see **Figure 10-19**).

*Note: The length of the TDMA frame used for the GSM sleep phase is always based upon the nominal TDMA frame length of 60 000 13 MHz clock cycles corresponding to a TDMA timer overflow value of 10 000. Any differing values stored in registers elsewhere (for example, in the GSM Timer Unit)are not taken into account.*

*Note: During the sleep phase the number of TDMA frames slept is shown in **SCCUTDMOUTL.TDMAOUT** and may be read by the controller.*

**Figure 10-19 Sleep Function Timing Diagram**

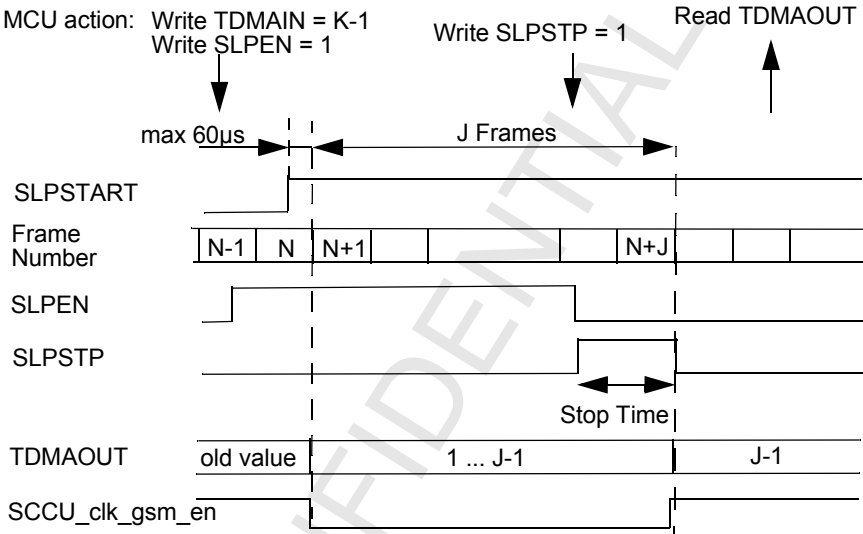


### Early Termination of GSM Timer Sleep Phase

If the GSM timer functionality is required, the GSM timer sleep phase can be aborted by setting bit **SCCUSLPCTRL.SLPSTP** (see [Figure 10-20](#)).

The duration of the sleep period can be calculated from **SCCUTDMOUTL.TDMAOUT** after the sleep phase has been terminated. The number of TDMA frames equals **TDMAOUT** incremented by 1.

**Figure 10-20 Early Termination of GSM Timer Sleep Phase**



#### 10.4.2.3.3 Stand-By Clock Calibration

##### Principle of Calibration

To be able to sleep for a GSM sleep timer period with a duration of an exact multiple of a TDMA frame, the duration of a TDMA frame  $t_{\text{TDMA}}$  has to be expressed as sum of a multiple NQTZ of a standby clock period  $t_{\text{stdby}}$  and a multiple  $n_{\text{ref}}$  of a reference clock period  $t_{\text{ref}}$  as shown in the equation below:

$$t_{\text{TDMA}} = \text{NQTZ} \cdot t_{\text{stdby}} + n_{\text{ref}} \cdot t_{\text{ref}} \quad [0.21]$$

During the calibration value  $n_{\text{ref}}$  has to be determined. For this purpose the required multiple of the standby clock period  $\text{NQTZ} \cdot t_{\text{stdby}}$  has to be measured accurately in terms of reference clock periods  $t_{\text{ref}}$ .

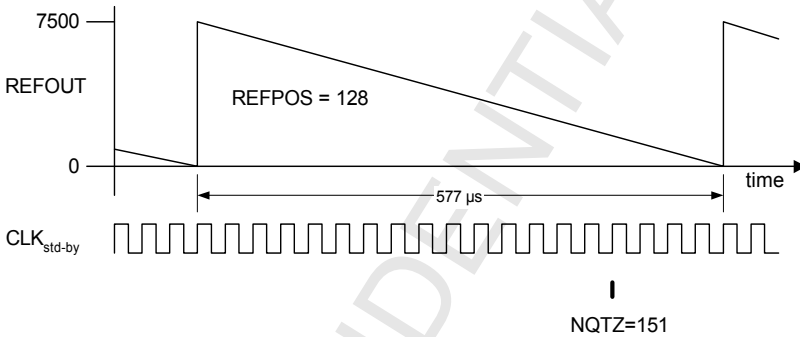


To increase the accuracy of this calibration 16 TDMA frames are chosen for this calibration procedure instead of one given in [Equation \[0.21\]](#):

$$16 \cdot t_{\text{TDMA}} = 16 \cdot \text{NQTZ} \cdot t_{\text{stdby}} + 16 \cdot n_{\text{ref}} \cdot t_{\text{ref}} \quad [0.22]$$

To measure value  $16 \cdot n_{\text{ref}}$  each TDMA frame is divided into 8 interval of equal length (corresponding in length to the TDMA time slots). In each of these intervals the REFOUT counter is running with the reference clock being decremented from 7500 to 0. In the last (8th) interval of the 16th TDMA frame the counter is stopped when the predefined number ( $16 \cdot \text{NQTZ}$ ) of standby clock cycles has elapsed as shown in [Figure 10-21](#). The value of the REFOUT counter then shows exactly the 16-fold value of  $n_{\text{ref}}$ .

**Figure 10-21 Stand-By Clock Calibration**



The number of standby clock cycles per TDMA frame NQTZ has to be selected so that it is slightly below the TDMA frame duration.

For a nominal standby clock frequency of 32 768 Hz the ration of the TDMA frame length to the standby clock period is 151.2:

$$1 \text{ TDMA frame} = 1 \cdot T_{\text{slot}} \cdot 8 = \frac{75}{130} \cdot 1 \times 10^{-3} \cdot 8 \quad [0.23]$$

Choosing NQTZ = 151 gives a value of  $n_{\text{ref}}$  of 94. The nominal value of  $16 \cdot n_{\text{ref}}$  for the 16 TDMA frame calibration period is 1504. This is the value of counter REFOUT for a standby clock frequency of exactly 32 768 Hz.

In general, the value of REFOUT for a stand-by clock frequency  $f_{\text{stand-by}}$  can be calculated by

$$\text{REFOUT} = 128 \cdot 7500 - \text{NQTZ} \cdot 16 \cdot f_{\text{scuu}} / f_{\text{stby}} \quad [0.24]$$

where  $f_{\text{scuu}} = 13 \text{ MHz}$  (refer to [RST\\_CTRL\\_STA.SCCUCL](#))

The optimum achievable resolution of the calibration can be calculated from the calibration time used. Since 16 TDMA frames are used for the calibration the total number of reference clock cycles used is  $16 \cdot 4.615 \text{ ms} \cdot 13 \text{ MHz} = 960\,000$ . The optimum achievable resolution, therefore, is about 1 ppm/LSB.

**CONFIDENTIAL**

**PLL - CGU - SCCU**

For a valid measurement the value of counter REFOUT must not be below 16. This limits the frequency tolerance to about  $\pm 1500$  ppm% which is far above the tolerance of standard crystals available on the market.

If a stand-by clock frequency other than 32 768 Hz is used, a similar calculation has to be done. The number of stand-by clock cycles NQTZ has to be adjusted so that values **SCCUREFH.REFPOS** and **SCCUREFH.REFOUT** meet the above criteria. The value NQTZ has to be chosen so that:

$$4.579 / \text{kHz} * f_{\text{stand-by}} < \text{NQTZ} < 4.615 / \text{kHz} * f_{\text{stand-by}} \quad [0.25]$$

### Operational Description

The calibration function is activated by setting bit **SCCUSLPCTRL.REFEN** to 1. As soon as the calibration has been completed the hardware resets bit **REFEN** to 0. By polling the value of this bit the end of the calibration can be determined. As soon as bit **REFEN** has been set, the values in registers **SCCUREFH** and **SCCUREFL** become invalid. When bit **REFEN** has been reset by the hardware and the calibration has been completed successfully, the new values in registers **SCCUREFH** and **SCCUREFL** are valid again.

*Note: The calibration is done completely independent from the system interface unit. All GSM timer functions, including modifications of registers **TCOR** or **TOVF**, can still be used without affecting the calibration process.*

A valid calibration has been performed if **SCCUREFH.REFPOS** equals 128 and value **SCCUREFH.REFOUT** is larger than 16. For a frequency of 32 768 Hz **REFOUT** is set to 1504.

An invalid calibration where the criteria mentioned above have been violated is signalled by bit **SCCUSLPCTRL.REFERR** being set to 1.

The result of the calibration contained in register **SCCUREFH.REFOUT** can be modified by the controller by writing a value to register **SCCUREFINL**. This new value will be copied to register **SCCUREFH** and will be used for the next sleep phase timing. Therefor, the controller can average the results over several calibration results.

CONFIDENTIAL

PLL - CGU - SCCU

#### 10.4.2.4 Setup

All SCCU functions rely on the SCCU clock which is generated by frequency division in the BPI from the X-Bus clock. For proper operation of the SCCU the frequency of this clock has to be 13 MHz. Do not select other SCCU reference clock settings if the Stand-By Mode is entered (even though other values are available via

**RST\_CTRL\_STA.SCCUCL.**

**Note: For E-GOLDradio < V3.0:**

*If the Master clock is 52 MHz, the SCCU frequency must be 13MHz.*

**But if the Master clock is 78 MHz, SCCUCL cannot be set to 13 MHz; at this frequency the Stand-By Mode cannot be entered.**

Before switching to standby-clock mode the PLL in the CGU should be by-passed and turned off to save energy. The X-Bus clock will then have to be set up in the CGU (refer to **Section 10.4.1 Clock Generation Unit (on Page 653)**) using the PLL by-pass.

After switch-back from standby-clock mode the SCCU clock must remain at 13 MHz until the switch-back is completed, that is, until the GSM timer clock has been switched back to the reference oscillator (see **Figure 10-18 Timing of Wake-Up Phase**). The state of the GSM sleep timer can be detected by evaluating bit **SCCUSLPCTRL.SLPEN**.

**CONFIDENTIAL**

**PLL - CGU - SCCU**

## 10.4.2.5 SCCU Registers

### 10.4.2.5.1 Overview

**Table 10-17 SCCU Register List 1**

Register Group	Register Name	Register Symbol
Identification	SCCU Identification	<b>SCCUID</b>
GSM sleep timer	Sleep Duration Value	<b>SCCUTDMINL</b>
	Sleep Duration Status	<b>SCCUTDMOUTL</b>
	Sleep and Calibration Control Register <sup>1)</sup>	<b>SCCUSLPCTRL</b>
	Pre-Wakeup and Wait Register <sup>2)</sup>	<b>SCCUWAITH</b>
Standby clock calibration	Reference Calibration Output Value High & Low	<b>SCCUREFH &amp; SCCUREFL</b>
	Quartz Number Register	<b>SCCUNQTZ</b>
	Sleep and Calibration Control Register <sup>3)</sup>	<b>SCCUSLPCTRL</b>
	Reference Input for standby clock	<b>SCCUREFINL</b>
System control	Switch Control Register	<b>SCCUSCCTRL</b>
	Pre-Wakeup Register Wait Register <sup>4)</sup>	<b>SCCUWAITH</b> <b>SCCUWAITL</b>
Other registers	Standby Power Down Control	<b>SCCUSPCR</b>
	Hardware Wakeup Control Register High	<b>SCCUHWWAKEUPH</b>
	Hardware Wakeup Control Register Low	<b>SCCUHWWAKEUPL</b>
	Clock Status Register	<b>SCCUCLKSTA</b>
	State Machine Status Register	<b>SCCUSMSTA</b>

<sup>1)</sup> Except bits REFEN and REFERR

<sup>2)</sup> Pre-Wakup bits only belong to GSM sleep timer

<sup>3)</sup> Bits REFEN and REFERR only

<sup>4)</sup> Wait bits are only for system control.

**CONFIDENTIAL**

**PLL - CGU - SCCU**

#### 10.4.2.5.2 SCCU Identification Register

**SCCUID**

**SCCU Identification Register**

**Reset values:**

**For E-GOLDradio Versions before V3.0: 4002<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>Module_ID</b>								<b>Revision_Number</b>							

Field	Bits	Type	Description
<b>Revision_Number</b>	7:0	r	<b>CGU Revision Number</b> These hard-wired bits are used for the CGU revision numbering.
<b>Module_ID</b>	15:8	r	<b>CGU Identification Number</b> These hard-wired bits are used for CGU identification numbering.

CONFIDENTIAL

PLL - CGU - SCCU

### 10.4.2.5.3 Standby Power Control Register

SCCUSPCR

Standby Power Control Register

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED									DR EN	DR OFF	RESERVED			A PDN	D PDN

Field	Bits	Type	Description
DPPN	0	rw	0 No action
			1 VDD_DSP supply domain reset in Standby Mode (refer to <a href="#">Figure 10-13 Overview of Interface Control Functions (on Page 689)</a> )
APDN	1	rw	0 No action
			1 Analog supply domain reset in Standby Mode (refer to <a href="#">Figure 10-13</a> )
DROFF	5	rw	0 DSP ROM is dependant on DREN
			1 DSP ROM is off immediately
DREN	6	rw	0 DSP ROM is never off
			1 DSP ROM is off when VCXO is off ( <a href="#">SCCUSMSTA.tcxo_off (S3)</a> = 1) and SHAPER POWER is down ( <a href="#">PLL_CTRL.Pll_sh_powerup</a> = 0)
Reserved	2, 3, 4,15:7	r	Reserved; these bits must be left at their reset values.

CONFIDENTIAL

PLL - CGU - SCCU

#### 10.4.2.5.4 Sleep Duration Value Register

SCCUTDMINL

Sleep Duration Value

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			TDMAIN												

Field	Bits	Type	Description
TDMAIN	12:0	rw	Duration (-1) of the Sleep time (in TDMA frames) $T_{\text{sleep}} = (\text{TDMAIN} + 1) * 4.615 \text{ ms}$
Reserved	15:13	r	Reserved, these bits must be left at their reset values.

#### 10.4.2.5.5 Sleep Duration Status Register

SCCUTDMOUTL

Sleep Duration Status

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			TDMAOUT												

Field	Bits	Type	Description
TDMAOUT	12:0	rh	Number of frames -1 during which the GSM timer has remained frozen
Reserved	15:13	r	Reserved, these bits must be left at their reset values.

**CONFIDENTIAL**

**PLL - CGU - SCCU**

#### 10.4.2.5.6 Sleep Control Register

**SCCUSLPCTRL**

**Sleep Control Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										<b>HW ACT DI</b>	<b>REF ERR</b>	<b>SLP STP</b>	<b>SLP RST</b>	<b>SLP EN</b>	<b>REF EN</b>

Field	Bits	Type	Description
<b>REFEN<sup>1)</sup></b>	0	rwh	<b>Reference Enable</b> If set by software, calibration of standby clock is started. Reset by hardware after the end of calibration of standby clock. Read values: 0 Calibration inactive 1 Calibration active
<b>SLPEN<sup>1)</sup></b>	1	rwh	<b>Sleep Enable</b> If set by software, activates the GSM sleep timer (sleep mode is started either immediately or after next rising edge of signal slpstart from GSM timer depending upon bit HWACTDI). Reset by hardware after GSM sleep timer has terminated. Read values: 0 GSM sleep timer inactive 1 GSM sleep timer running or activated
<b>SLPRST<sup>2)</sup></b>	2	rw	<b>Reset Sleep Counter</b> 0 Normal operation 1 Resets register <b>SCCUTDMOUTL</b>
<b>SLPSTP<sup>1)</sup></b>	3	rwh	<b>Sleep Stop</b> If set by software, synchronously stops the sleep counter. Reset by hardware when the sleep counter actually stops. Read values: 0 No sleep stop action pending (see <b>Figure 10-20 Early Termination of GSM Timer Sleep Phase (on Page 700)</b> ) 1 Sleep stop action pending
<b>REFERR</b>	4	rh	<b>Reference Error Flag</b> Set by HW during calibration. Reset by HW at the beginning of the calibration. 0 Calibration within range 1 Calibration out of range



**CONFIDENTIAL**

**PLL - CGU - SCCU**

Field	Bits	Type	Description
<b>HWACTDI</b>	5	rw	<b>slpstart Activation Disable</b>
			0 Sleep phase starts if SLPEN is set with rising edge of signal slpstart from GSM timer
			1 Sleep phase starts when SLPEN is set
<b>Reserved</b>	15:6	r	Reserved, these bits must be left at their reset values.

- 1) Writing 0 to this bit has no effect on hardware and does not reset this bit, it is only reset by hardware; when writing to this register always write 0 to this bit if the function is not activated.
- 2) Must not be released as long as **SCCUTDMOUTL.TDMAOUT** != 0.

#### 10.4.2.5.7 Standby Clock Reference Input Register

**SCCUREFINL**

**Standby Clock Reference Input Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>				<b>REFIN</b>											

Field	Bits	Type	Description
<b>REFIN<sup>1)</sup></b>	12:0	rw	Value of the reference controlled by the MCU (1 LSB = 1.043 ppm) optionally used for overriding value in register <b>SCCUREFL.REFOUT</b>
<b>Reserved</b>	15:13	r	Reserved, these bits must be left at their reset values.

- 1) **REFIN** is only taken into account after:
- SCCUREFL.REFOUT** has been modified by a calibration operation (**SCCUSLPCTRL.REFEN** = 1)  
=> **SCCUREFL.REFOUT** = **REFOUT<sub>calibrated</sub>**
  - REFIN** is written to  
=> **SCCUREFL.REFOUT** is not overwritten, **REFOUT** != **REFIN**
  - Do a Sleep Enable (**SCCUSLPCTRL.SLPEN** = 1)
  - Wakeup the GSM (**SCCUSLPCTRL.SLPEN** = 0)  
=> Now **SCCUREFL.REFOUT** = **REFIN**.

**CONFIDENTIAL**

**PLL - CGU - SCCU**

### 10.4.2.5.8 Reference Calibration Output Values

#### SCCUREFH

**High Reference Calibration Output Value**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			REFPOS												

Field	Bits	Type	Description
REFPOS	12:0	rh	Coarse measurement of the slow Xtal frequency (1 LSB = 1/8-th of TDMA frame). Nominal value is 128 (16 TDMA frames).
Reserved	15:13	r	Reserved, these bits must be left at their reset values.

#### SCCUREFL

**Low Reference Calibration Output Value**

**Reset value: 05E0<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			REFOUT												

Field	Bits	Type	Description
REFOUT	12:0	rh	Fine measurement (1 LSB = 1.043 ppm)
Reserved	15:13	r	Reserved, these bits must be left at their reset values.

CONFIDENTIAL

PLL - CGU - SCCU

### 10.4.2.5.9 Oscillator Crystal Periods per Frame

#### SCCUNQZT

Xtal Oscillator Number Register

Reset value: 0097<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								NQZT							
r								rw							

Field	Bits	Type	Description
NQZT	7:0	rw	Number of slow Xtal oscillator periods in one TDMA frame (default 151)

### 10.4.2.5.10 Switch Control Register

#### SCCUSCTRL

Switch Control Register

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												SSC RST	UC WUP	UC SLP	
r												wh	wh	wh	

Field	Bits	Type	Description
UCSLP <sup>1)</sup>	0	wh	<b>Sleep Command</b> Set by software to start system sleep phase, then reset by hardware after 3 standby clock periods.
UCWUP <sup>1)</sup>	1	wh	<b>Wakeup Command</b> Set by software to start wakeup of system, then reset by hardware after 3 standby clock periods.
SSCRST <sup>1)</sup>	2	wh	<b>Reset Command</b> Can optionally be set by software to reset state machine. It is reset by hardware after 3 standby clock periods.
Reserved	15:3	r	Reserved for future use; these bits must be left at their reset values.

<sup>1)</sup> Bit can also be reset by software, but not required; resetting bit too early (before next rising edge of standby clock) may cancel function; bit will in any case be reset by hardware after 3 standby clock periods  
example : set bit ucslp : write 0001<sub>H</sub>, set bit ucwup : write 0002<sub>H</sub> - potentially cancels function triggered by bit ucslp since 0 is written to bit ucslp; bit ucwup will in any case be reset after 3 standby clock periods and system will return to  $\mu c_{on}$  state

**CONFIDENTIAL**

**PLL - CGU - SCCU**

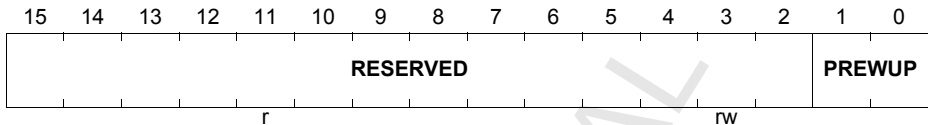
### 10.4.2.5.11 High Pre-Wakeup and Wait Registers

*Note: Refer to the note after [Figure 10-18 Timing of Wake-Up Phase \(on Page 696\)](#).*

#### SCCUWAITH

##### High Pre-Wakeup Register

**Reset value: 0003<sub>H</sub>**

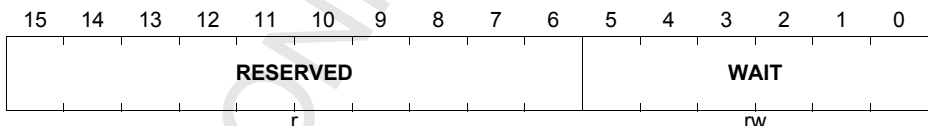


Field	Bits	Type	Description
PREWUP	1:0	rw	<b>Pre-Wakeup Time</b> Determines the number of TDMA frames signal <i>pre_wup</i> is asserted before the end of the sleep phase. The number of TDMA frames is given by <b>PREWUP</b> + 1, that is, it is selectable between 1 and 4 frames. Reset value is 4 frames.
Reserved	15:2	r	Reserved for future use; these bits must be left at their reset values.

#### SCCUWAITL

##### Wait Register

**Reset value: 003F<sub>H</sub>**



Field	Bits	Type	Description
WAIT	5:0	rw	<b>Used to Calculate the VCXO Wait Loop Duration</b> Range: 0 to 63  Resolution = $\left(\frac{4}{32768 \text{ Hz}}\right)$ sec.  The reset value (63) corresponds to the maximum loop value of 7.7 ms.
Reserved	15:6	r	Reserved for future use; these bits must be left at their reset values.

**CONFIDENTIAL**

**PLL - CGU - SCCU**

### 10.4.2.5.12 Hardware Wakeup Control Registers

#### SCCUHWWAKEUPH

##### High Hardware Wakeup Control Register

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															ICU_EN
	r		rw	r	rw	rw	rw					r			

Field	Bits	Type	Description
ICU_EN	0	rw	Enables wake-up of SCCU by interrupt from ICU to MCU
Reserved	15:1	r	Reserved for future use; these bits must be left at their reset values.

#### SCCUHWWAKEUPL

##### Low Hardware Wakeup Control Register

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			EXT_EN	RESERVED	SIM_EN	KPD_EN	RTC_EN	RESERVED							
		r	rw	r	rw	rw	rw					r			

Field	Bits	Type	Description
RTC_EN	8	rw	Enables the SCCU sleep mode termination by RTC block
KPD_EN	9	rw	Enables the SCCU sleep mode termination by pressing a keypad key
SIM_EN	10	rw	Enables the SCCU sleep mode termination by SIM card insertion/removal
EXT_EN	12	rw	Enables wake-up from one of the CAPCOM or external interrupt pins
Reserved	15:13, 11, 7:0	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**PLL - CGU - SCCU**

### SCCUCLKSTA

**Clock Status Register**

**Reset value: 0003<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED													GSM CLK	CPU CLK	

Field	Bits	Type	Description
CPUCLK	0	r	<b>Status of the MCU Clock</b> 0: Slow MCU clock 0 or 32 kHz 1: Fast MCU clock
GSMCLK	1	r	<b>Status of the System Interface Clock</b> 0: 0 Hz 1: 13 MHz
Reserved	15:2	r	Reserved; these bits must be left at their reset values.

### SCCUSMSTA

**State Machine Status Register**

**Reset value: 0001<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											shap on (S5)	tcxo on (S4)	tcxo off (S3)	µc off (S2)	µc on (S1)

Field	Bits	Type	Description
µc_on (S1)	0	r	0: SCCU state machine is not in indicated state 1: SCCU state machine is in indicated state Refer to <a href="#">Section 10.4.2.2 System Clock Control (on Page 690)</a> .
µc_off (S2)	1	r	
tcxo_off (S3)	2	r	
tcxo_on (S4)	3	r	
shap_on (S5)	4	r	
Reserved	15:5	r	Reserved; these bits must be left at their reset values.

## 10.5 Measurement Interface

History	
Current version: Rev. 1.06, 2005-11-04	
Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 0.02	
<b>Page 744</b>	Updated register <b>MEAS_ID</b>
Changes for Rev. 1.01	
<b>Page 716</b>	Updated <b>System Integration</b> , WS00006682
Changes for Rev. 1.03	
<b>Page 749</b>	Update parameters for 78 MHz in <b>Table 10-29 K and L Values for Various Bus Clock Frequencies</b> WS00007771
Changes for Rev. 1.04	
<b>Page 728</b>	Uncovered the end of <b>Equation [22]</b> WS00008365
<b>Page 728</b>	Corrections to <b>Equation [20]</b> and <b>Equation [21]</b> WS00008366
<b>Page 744</b>	Updated reset value of register <b>MEAS_ID</b> WS0007724
Changes for Rev. 1.06	

**CONFIDENTIAL**

**Measurement Interface**

**System Integration**

- Supply domains
  - VDD\_MAIN for the digital part of the measurement interface
  - VDDm/VSSm, AGND for the analog part of the measurement interface (analog domain)
- Chip internal interfaces:
  - Clock domain: Refer to [Section 10.4.1.3 Sub-System Clocks and Enables \(on Page 661\)](#) and see [Figure 10-11 Clock and Enable Generation for MCU Sub-System \(on Page 668\)](#).
  - Bus domain: X-Bus
- Interrupt sources 1 regular interrupt
- Other interfaces ADCTRIG
- Chip external signals
  - M0, M1, M2, M7, M8, M9, M10 measurement inputs
  - VDDm, VSSm supply

**10.5.1 Functional Overview**

This peripheral is connected to the 16-bit X-Bus.

**10.5.1.1 Features**

Seven general purpose measurement inputs are provided for the measurement of:

- Battery voltage (VBAT)
- Battery technology (BTEC)
- Battery temperature (TBAT)
- Environmental temperature (TENV)
- Oscillator temperature (TVCO)
- Power of power amplifier (PWPA)
- Brightness sensor
- Accessory detector
- etc.

There are various functions for on-chip measurements:

- On chip temperature (TIC)
- Static offset of baseband transmit I and Q path (TXOF)
- Static offset of power ramp output (PAOUTOF1)



**CONFIDENTIAL**

**Measurement Interface**

- Measurement offset self-calibration (OFS)

One separate analog-to-digital converter (ADC) is used.

The general purpose measurement inputs can be configured for voltage measurements with the following characteristics:

- Single-ended, voltage mode, referred to internal AGND
- Single-ended, current mode, referred to internal AGND
- Single-ended, voltage mode, referred to external voltage
- Single-ended, current mode, referred to external voltage drop (resistor)
- Fully differential, voltage mode.

Various measurements with different characteristics can be combined. The measurement modes are adjusted via analog switches as shown in the following sections. The switches are controlled by the Measurement Control Registers, **MEAS\_CTRL1/MEAS\_CTRL2**. The Measurement Status Register, **MEAS\_STAT**, provides information about the status of the measurement interface. Data is transferred via the Measurement Data Registers, **MEAS\_DATAx**. All registers are part of the measurement interface X-Bus interface (refer to [Section 10.5.9 \(on page 732\)](#)).

## 10.5.2 Register Overview

All registers are accessed as 32 bit registers. Unused or reserved bits have to be set to zero. The register addresses are defined in [Section 12.3 X-Bus Register Addresses \(on Page 1285\)](#).

**Table 10-18 Measurement Interface Register List**

Register	Register Name
<b>MEAS_CTRL1</b>	Low Measurement Control Register
<b>MEAS_CTRL2</b>	High Measurement Control Register
<b>MEAS_STAT</b>	Measurement Status Register
<b>MEAS_DATA0</b>	Measurement Data Register 0
<b>MEAS_DATA1</b>	Measurement Data Register 1
<b>MEAS_DATA2</b>	Measurement Data Register 2
<b>MEAS_DATA3</b>	Measurement Data Register 3
<b>MEAS_DATA4</b>	Measurement Data Register 4
<b>MEAS_DATA5</b>	Measurement Data Register 5
<b>MEAS_DATA6</b>	Measurement Data Register 6
<b>MEAS_DATA7</b>	Measurement Data Register 7

**Table 10-18 Measurement Interface Register List (cont'd)**

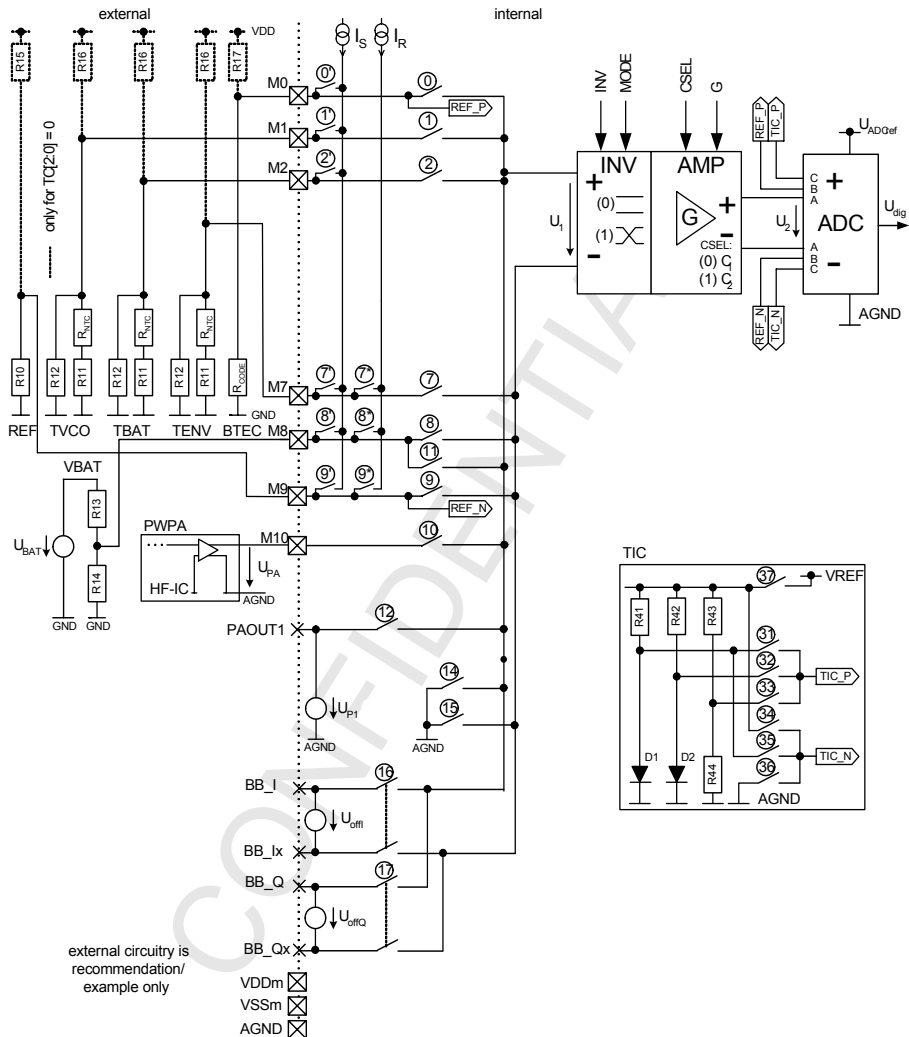
Register	Register Name
<b>MEAS_CLK</b>	Measurement Peripheral Clock Control Register
<b>MEAS_ID</b>	Module Identification Register

### 10.5.3 Measurement Circuit and Mode Setting

For the conversion of all measurement signals like battery voltage, battery technology code, temperature, etc. identical measurement inputs M0 to M2 and M7 to M10 can be used. For self-calibration purposes several additional chip internal inputs are available.

**Figure 10-22** shows the principle of the measurement circuits consisting of external sensing components (recommendation/example) and integrated analog multiplexers and switches. **Table 10-19 Measurement Switch Settings (on Page 720)** shows the corresponding switch settings.

**Figure 10-22 Measurement Circuit and Mode Setting**



The measurement modes and switch settings defined in [Table 10-19](#) are controlled via the Measurement Control Registers ([MEAS\\_CTRL1](#) and [MEAS\\_CTRL2](#)). The dashed lines in [Figure 10-22](#) indicate the additional external components that have to be used if the internal current source is not used. On chip temperature measurement TIC is described later (refer to [Section 10.5.6 \(on page 728\)](#)).

**Table 10-19 Measurement Switch Settings**

Mode	MEAS_CTRL1.MX [5:0] <sup>1)</sup>	Switches Closed <sup>1)2)</sup>	Gain <sup>1)</sup>	ADC Inputs <sup>1)</sup>	Description
M0A	01 <sub>H</sub>	0, 0', 15	0.5	A	GP <sup>3)</sup> referred to internal AGND
M1A	02 <sub>H</sub>	1, 1', 15	0.5	A	
M2A	03 <sub>H</sub>	2, 2', 15	0.5	A	
M7B	08 <sub>H</sub>	7, 7', 14	1.0	A	
M8B	09 <sub>H</sub>	8, 8', 14	1.0	A	
M9B	0A <sub>H</sub>	9, 9', 14	1.0	A	
M10	0B <sub>H</sub>	10, 15	0.4	A	GP/PWPA referred to internal AGND
M0M9A	0C <sub>H</sub>	0, 0', 9, 9*	0.5	A	GP referred to external voltage or GP differential
M1M9A	0D <sub>H</sub>	1, 1', 9, 9*	0.5	A	
M2M9A	0E <sub>H</sub>	2, 2', 9, 9*	0.5	A	
M7M8A	13 <sub>H</sub>	7, 7', 11, 8*	0.5	A	
M1M8A	14 <sub>H</sub>	1, 1', 8, 8*	0.5	A	
M2M7A	15 <sub>H</sub>	2, 2', 7, 7*	0.5	A	
M0M9B	18 <sub>H</sub>	0, 0', 9, 9*	1.0	A	
M1M9B	19 <sub>H</sub>	1, 1', 9, 9*	1.0	A	
M2M9B	1A <sub>H</sub>	2, 2', 9, 9*	1.0	A	
M1M8B	1D <sub>H</sub>	1, 1', 8, 8*	1.0	A	
M7M8B	20 <sub>H</sub>	7, 7', 11, 8*	1.0	A	
M2M7B	21 <sub>H</sub>	2, 2', 7, 7*	1.0	A	
PAOUTOF1	24 <sub>H</sub>	12, 15	1.0	A	Offset of power ramping output PAOUT1
PAOUTOF2	25 <sub>H</sub>	13, 15	1.0	A	Offset of power ramping output PAOUT2 (Not used)
TXOFI	26 <sub>H</sub>	16	4.0	A	Offset TX path I component
TXOFQ	27 <sub>H</sub>	17	4.0	A	Offset TX path Q component

**CONFIDENTIAL**

**Measurement Interface**

**Table 10-19 Measurement Switch Settings (cont'd)**

Mode	MEAS_CTRL1.MX [5:0] <sup>1)</sup>	Switches Closed <sup>1)2)</sup>	Gain <sup>1)</sup>	ADC Inputs <sup>1)</sup>	Description
ADCCAL	2A <sub>H</sub>	-	-	B	ADC gain calibration mode
TICD1	2B <sub>H</sub>	31, 36, 37	-	C	On chip temperature: U <sub>BE,D1</sub>
TICD2	2C <sub>H</sub>	32, 36, 37	-	C	On chip temperature: U <sub>BE,D2</sub>
TICR1	2D <sub>H</sub>	31, 34, 37	-	C	On chip temperature: -U <sub>R41</sub>
TICR2	2E <sub>H</sub>	32, 34, 37	-	C	On chip temperature: -U <sub>R42</sub>
TICDIFF	2F <sub>H</sub>	32, 35, 37	-	C	On chip temperature: U <sub>BE,D2</sub> - U <sub>BE,D1</sub>
TICREF	30 <sub>H</sub>	33, 36, 37	-	C	On chip temperature calibration: VREF/2
OFF	00 <sub>H</sub>	-	-	-	Power save

<sup>1)</sup> A given value for the MX bit field sets the mode. This closes the switches and sets the gain and ADC inputs to the values listed on the same line as the mode.

<sup>2)</sup> Switches not mentioned are open. In the power save mode (OFF) all switches are open.

<sup>3)</sup> General purpose

**Table 10-20 General Pre-Amplifier Settings**

Bit Name	Value	Description
MEAS_CTRL2.CSEL	0	Slow settling of pre-amplifier (C1) <sup>1)</sup>
	1	Fast settling of pre-amplifier (C2)
MEAS_CTRL2.INV	0	No inversion of pre-amp. input voltage
	1	Inversion of pre-amplifier input voltage <sup>2)</sup>

<sup>1)</sup> It is recommended to set **CSEL** = 0 for all measurement modes, except PWPA, to ensure 12-bit resolution of the ADC.

<sup>2)</sup> Voltages measured with **INV** = 1 are marked with an asterisk (\*) in this document section.

**Table 10-21 Resistors for Measurement Circuits**

<b>Resistor/Capacitor</b>	<b>Value<sup>1)</sup> (typical)</b>
R10	user defined
R11	user defined
R12	user defined
R13	user defined
R14	user defined
R15	user defined
R16	user defined
R17	user defined
R <sub>NTC</sub>	user defined
R <sub>CODE</sub>	user defined
R41	500k
R42	50k
R43	50k
R44	50k

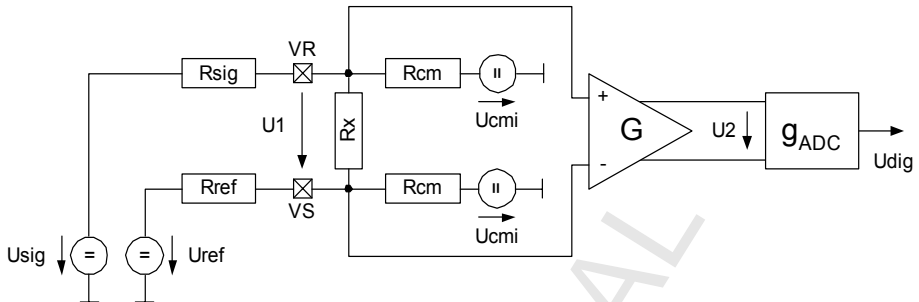
<sup>1)</sup> Resistor values R41-R44 are typical values. These values may differ by  $R = R_{typ} \cdot (1 \pm 25\% \text{ process tolerances} \pm 10\% \text{ temperature coefficient})$ .

## **10.5.4 Differential General Purpose Measurements (MxMy)**

### **10.5.4.1 Equivalent Network for Differential Measurement Circuit**

The electrical behavior of the differential measurement circuit for the modes MxMyA and MxMyB (that is, VR-VS) can be modelled by the equivalent network according to **Figure 10-23** containing an ideal amplifier with gain  $G$ . The parameters  $R_{cm}$ ,  $R_x$ ,  $U_{cmi}$  and  $G$  are individual for each measurement mode but do not depend on the external circuitry. The user dependent external circuitry is also replaced by equivalent voltage sources  $U_{ref}$  and  $U_{sig}$  with internal resistances  $R_{ref}$  and  $R_{sig}$  respectively. Refer to **Section 10.5.4.2 Equivalent Network for Reference Voltage Generation (on Page 724)** and **Section 10.5.4.3 Equivalent Network for Signal Input (on Page 725)**.

Figure 10-23 Equivalent Network for Differential Measurement Circuit



The input voltage  $U_2$  of the analog-to-digital converter can be calculated according to

$$U_2 = G \cdot \frac{U_{cmi} \cdot \left( \frac{1}{1 + R_{cm}/R_{sig}} - \frac{1}{1 + R_{cm}/R_{ref}} \right) + \frac{U_{sig}}{1 + R_{sig}/R_{cm}} - \frac{U_{ref}}{1 + R_{ref}/R_{cm}}}{1 + \frac{1}{R_x} \cdot \left( \frac{1}{1/R_{cm} + 1/R_{sig}} + \frac{1}{1/R_{cm} + 1/R_{ref}} \right)} \quad [1]$$

and the output of the ADC results in

$$U_{dig} = U_2 \cdot g_{ADC} + U_{dig,0} \quad [2]$$

$$\text{with } g_{ADC} = 2048 \text{ LSB} \cdot \frac{1}{0.8333 \cdot V_{REF}} = 2048 \text{ LSB/V typ.} \quad [3]$$

$$U_{dig,0} = 2048 \text{ LSB typ.} \quad [4]$$

$$\text{and } V_{REF} = 1.2 \text{ V typ.} \quad [5]$$

**Note:** The reference voltage of the ADC ( $0.8333 \cdot V_{REF}$ ) shows the same tolerance as  $V_{REF}$ .

The measurement modes M0M9A/B, M1M8A/B and M2M7A/B are especially suited for measurements of true differential voltages. If the internal resistance of the differential source is negligible the input voltage of the ADC in these modes is given by

$$U_2 = G \cdot (U_{Mx} - U_{My}) \quad [6]$$

**Note:** In addition to the condition for the input voltage, the conditions for differential and common mode voltage have to be satisfied.

*Note: For offset elimination of offsets caused by the measurement path itself the measurement can be repeated with **MEAS\_CTRL1.INV** = 1 and the evaluation method as described in [Section 10.5.8 \(on page 731\)](#) can be used.*

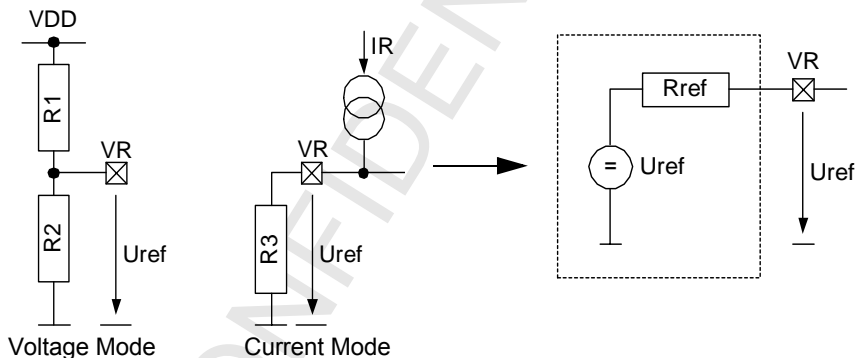
#### 10.5.4.2 Equivalent Network for Reference Voltage Generation

According to the example for the external circuitry (REF) in [Figure 10-22 \(on page 719\)](#) the external reference voltage generation can be replaced by a voltage source as shown in [Figure 10-24](#).

For the pins Mx and My an alternative measurement mode exists which can be enabled via the control bits **MEAS\_CTRL1.TC**. In this case the internal current sources are switched on and the difference of the voltage which drops over the external resistors is measured at the differential input. With **TC** the current source  $I_R = 60 \mu A$  is switched on and for IS different currents can be selected.

*Note:  $I_R$  and  $I_S$  are proportional to  $V_{REF}$  ( $I_R = G_{IR} V_{REF}$ ,  $I_S = G_{IS} V_{REF}$ ).*

**Figure 10-24 Equivalent Network for Reference Voltage Generation**



The equivalent parameters in voltage mode are

$$U_{ref} = VDD \cdot \frac{R_2}{R_1 + R_2} \quad [7]$$

and 
$$R_{ref} = \frac{R_1 \cdot R_2}{R_1 + R_2} \quad [8]$$



**CONFIDENTIAL**

**Measurement Interface**

and in current mode they are

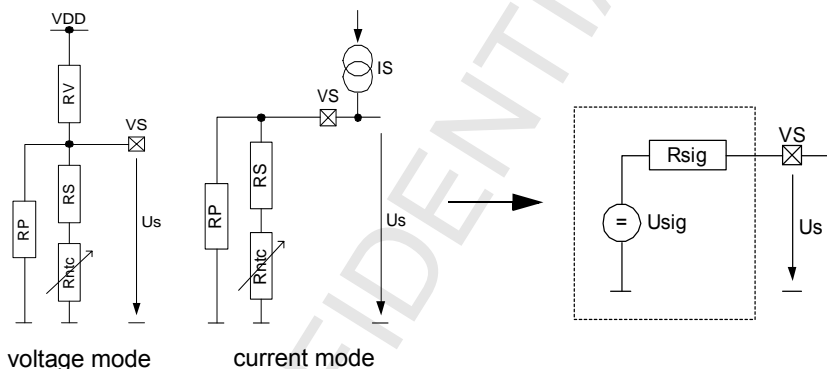
$$U_{\text{ref}} = IR \cdot R_3 \text{ with } IR = 60 \mu\text{A typ.} \quad [9]$$

$$\text{and } R_{\text{ref}} = R_3 . \quad [10]$$

### 10.5.4.3 Equivalent Network for Signal Input

According to the example for the external circuitry (TVCO, TENV, TBAT) in [Figure 10-22 \(on page 719\)](#) the signal input can also be replaced by a voltage source as shown in [Figure 10-25](#).

**Figure 10-25 Equivalent Network for Measurement Signal Input**



The equivalent parameters in voltage mode are

$$U_{\text{sig}} = \frac{VDD}{1 + R_V \cdot \left( \frac{1}{R_P} + \frac{1}{R_S + R_{ntc}} \right)} \quad [11]$$

$$\text{and } R_{\text{sig}} = \left( \frac{1}{R_V} + \frac{1}{R_P} + \frac{1}{R_S + R_{ntc}} \right)^{-1} \quad [12]$$

CONFIDENTIAL

Measurement Interface

In current mode the following equations apply:

$$U_{\text{sig}} = \frac{IS}{\frac{1}{R_p} + \frac{1}{R_s + R_{\text{ntc}}}} \quad \text{with } IS = I_{\text{TC}[2:0]} \text{ typ.} \quad [13]$$

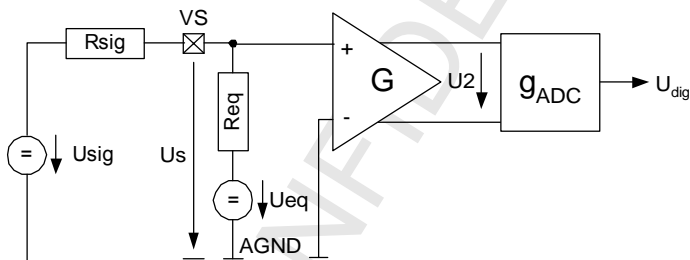
$$\text{and } R_{\text{sig}} = \left( \frac{1}{R_p} + \frac{1}{R_s + R_{\text{ntc}}} \right)^{-1} \quad [14]$$

## 10.5.5 Single-Ended General Purpose Measurements (Mx)

### 10.5.5.1 Equivalent Network for Measurement Circuit

The electrical behavior of the measurement circuit for the input modes Mx (i.e. VS) in voltage mode can be modelled by the equivalent network according to [Figure 10-26](#) containing a resistor, a voltage source and an ideal amplifier with gain  $G$ . The parameters  $R_{\text{eq}}$ ,  $U_{\text{eq}}$  and  $G$  are individual for each measurement mode but not dependent on the external circuitry.

**Figure 10-26 Equivalent Network for Single-Ended Measurement Mx**



The input voltage  $U_2$  of the analog-to-digital converter can be calculated according

$$U_2 = G \cdot \frac{U_{\text{sig}}/R_{\text{sig}} + U_{\text{eq}}/R_{\text{eq}}}{1/R_{\text{sig}} + 1/R_{\text{eq}}} \quad [15]$$

and the output of the ADC according [Equation \[2\] \(on Page 723\)](#).

### 10.5.5.2 Equivalent Network for Signal Input

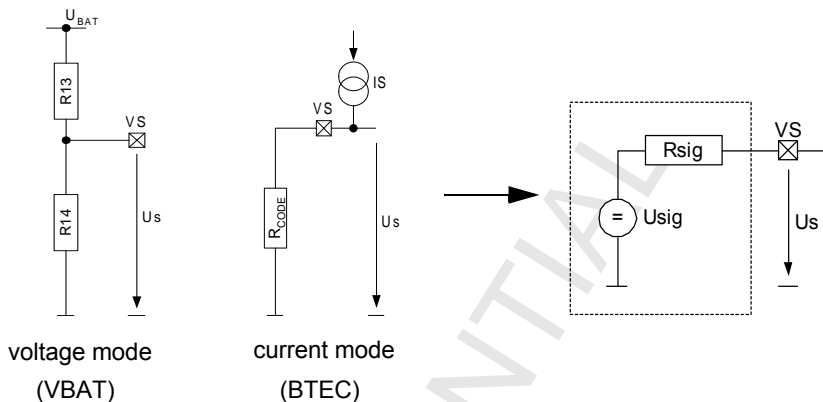
According to the example for the external circuitry (VBAT, BTEC, PWPA) in [Figure 10-22 \(on page 719\)](#) the signal input can also be replaced by a voltage source as shown in [Figure 10-27](#). For the single-ended measurements an alternative measurement mode exists which can be also enabled via the control bits [MEAS\\_CTRL1.TC](#). In this case an

CONFIDENTIAL

Measurement Interface

internal current source is switched on and the voltage drop over an external resistor is measured at the input Mx. With TC different IS currents can be selected.

**Figure 10-27 Equivalent Network for Measurement Signal Input**



The equivalent parameters in voltage (for example, VBAT) mode are:

$$U_{sig} = \frac{U_{BAT}}{1 + R_{13}/R_{14}} \quad [16]$$

$$\text{and } R_{sig} = \left( \frac{1}{R_{13}} + \frac{1}{R_{14}} \right)^{-1} \quad [17]$$

In current mode (for example, BTEC) the following equations apply:

$$U_{sig} = IS \cdot R_{CODE} \text{ with } IS = I_{TC[2:0]} \text{ typ.} \quad [18]$$

$$\text{and } R_{sig} = R_{CODE} \quad [19]$$

*Note: IS is proportional to VREF (IS = G<sub>IS</sub> VREF).*

### 10.5.6 On-Chip Temperature Measurement

The temperature sensor for measurement of the on chip temperature (TIC) is placed in the mixed signal section of the chip in the neighborhood of the controller block. The temperature characteristic of a well-defined semiconductor junction at a defined current level is used for determining the on chip temperature.

For the measurement modes TICDx and TICRx the output of the ADC is given by

$$U_{\text{dig, TICDx}} = g_{\text{ADC}} \cdot \frac{R_{\text{Dx}}(I_{\text{Dx}})}{R_{\text{Dx}}(I_{\text{Dx}}) + R_{4x}} \cdot V_{\text{REF}} + U_{\text{dig,0}} \quad [20]$$

$$U_{\text{dig, TICRx}} = -g_{\text{ADC}} \cdot \frac{R_{3x}}{R_{\text{Dx}}(I_{\text{Dx}}) + R_{4x}} \cdot V_{\text{REF}} + U_{\text{dig,0}} \quad [21]$$

where  $R_{\text{Dx}}(I_{\text{Dx}})$  is the actual temperature sensitive resistance of the diode Dx with current  $I_{\text{Dx}}$  and  $g_{\text{ADC}}$  is the gain of the ADC according to [Equation \[36\] \(on Page 732\)](#).

To avoid expensive temperature calibration processes a four-step measurement method can be carried out. For the first two measurements the modes TICD1 and TICD2 are used and to determine the current ratio of the two diodes D1 and D2. The absolute temperature in degrees Kelvin can be calculated by

$$^{\circ}\text{K} = \frac{q}{k} \cdot \left[ \ln \left[ 14 \cdot \frac{R_{41}}{R_{42}} \cdot \frac{U_{\text{dig, TICR2}} - U_{\text{dig,0}}}{U_{\text{dig, TICR1}} - U_{\text{dig,0}}} \right] \right]^{-1} \cdot \frac{U_{\text{dig, TICD2}} - U_{\text{dig, TICD1}}}{g_{\text{ADC}}} \quad [22]$$

$$\text{where } q = 1.602177 \cdot 10^{-19} \text{ As}, k = 1.380658 \cdot 10^{-23} \text{ J/K}. \quad [23]$$

*Note: The area ratio of the diodes is  $A_{\text{D1}}/A_{\text{D2}} = 14$ .*

To achieve the specified accuracy in [Table 12-52 On Chip Temperature Measurement TIC \(on Page 1377\)](#), a factory calibration of the ADC gain  $g_{\text{ADC}}$  has to be carried out before temperature measurements using the measurement mode ADCCAL. Refer to [Section 10.5.8.1 Gain Calibration of Measurement ADC \(on Page 732\)](#) for further details.

### 10.5.7 Modulator Unit Offset Measurement TXOFI and TXOFQ

The input voltage of the ADC in mode TXOFI and TXOFQ is given by

$$U_2 = G \cdot U_{\text{off}/Q} \quad [24]$$

If the bit INV is set HIGH (this is indicated by an asterisk) then the input voltage of the ADC in mode TXOFI\* and TXOFQ\* is given by

$$U_2^* = -G \cdot U_{\text{off}/Q} \quad [25]$$

To eliminate effects caused by offsets in the measurement interface the offsets in the transmit paths can be calculated using

$$U_{\text{off}/Q} = \frac{1}{G} \cdot \frac{1}{2} (U_2 - U_2^*) \quad [26]$$

and

$$U_{\text{off}/Q} = \frac{1}{G} \cdot \frac{U_{\text{dig,TXOFI}/Q} - U_{\text{dig,TXOFI}/Q}^*}{2 \cdot g_{\text{ADC}}} \quad [27]$$

*Note: This offset elimination method can be used for all differential measurements.*

#### 10.5.7.1 PA Control Hardware Offset Measurement PAOUTOF1

For PAOUT1 outputs of the PA Control Hardware the offset can be determined. The output of the measurement ADC is

$$U_{\text{dig,PAOUTOF}} = G \cdot g_{\text{ADC}} \cdot U_p + U_{\text{dig},0} \quad [28]$$

and the PAOUT voltage is

$$U_p = -\frac{U_{\text{dig,PAOUTOF}} - U_{\text{dig},0}}{g_{\text{ADC}} \cdot G} \quad [29]$$

*Note: For measurement accuracy enhancement, a calibration of  $g_{\text{ADC}}$  using the mode ADCCAL according to [Equation \[36\] \(on Page 732\)](#) and an offset correction according [Section 10.5.7 Modulator Unit Offset Measurement TXOFI and TXOFQ \(on Page 729\)](#) is recommended.*

To determine the offset of the PAOUT signal two measurements at different PAOUT voltages  $U_{p_a}$  and  $U_{p_b}$  should be carried out according [Figure 10-28 \(on page 731\)](#). Then, the offset voltage

$$U_{\text{offP}} = U_{p_a} - \frac{U_{p_b} - U_{p_a}}{\text{PARx}(16)_b - \text{PARx}(16)_a} \cdot \text{PARx}(16)_a \quad [30]$$

CONFIDENTIAL

Measurement Interface

can be calculated by solving a linear equation.

*Note: The PAOUT voltage  $U_p$  can be adjusted in the GSM System Interface by programming a specific power ramp (ramp up sequence) with the desired final output voltage at the PAOUT pin defined by PARx(16) (refer to [Section 10.12.2 RF RAM \(on Page 836\)](#)). Set PAINCx to 000<sub>H</sub>. Additionally, the fields **ANA\_CTRL2.PA\_CAL1** have to be set to 00<sub>H</sub> prior to any PAOUTOF measurement.*

For determining  $U_{\text{offp}}$  set the **ANA\_CTRL1.PA\_OFF1** flag to 0. If the result shows  $U_{\text{offp}} > U_{\text{sat-}}$  then determine  $U_{\text{offp}}$  a second time with **PA\_OFF1** = 1. In this case it should be  $U_{\text{offp}} \leq U_{\text{sat-}}$ . For usual PAOUT operation use the **PA\_OFF1** value where  $U_{\text{offp}} \leq U_{\text{sat-}}$ .

Finally, the calibration value is given by

$$\text{PA\_CAL} = \text{PARx}(16)_a - \frac{\text{PARx}(16)_b - \text{PARx}(16)_a}{(U_{\text{pb}}/U_{\text{pa}}) - 1} + \frac{U_{\text{drop-}}}{1.25 \text{ mV/LSB}} \quad [31]$$

The calibration field **ANA\_CTRL2.PA\_CAL1** can now be programmed with the value according to [Equation \[31\]](#) to avoid saturation of the output amplifier for the pin PAOUT and to avoid the need of device specific power ramp sequences (only concerning any offset effects).

Due to this offset calibration the output voltage at the pin PAOUT for PARx(16) = 00<sub>H</sub> is defined and is  $U_{\text{PAOUT}} = U_{\text{drop-}}$  which is within the linear operating range of the output buffer. This is the precondition for transient-free power ramps.

*Note: During final offset measurement the **ANA\_CTRL1.PA\_OFF1** flag has to be set in the same way as it is set during usual PAOUT operation to get correct measurement results and a valid **PA\_CAL1** value.*

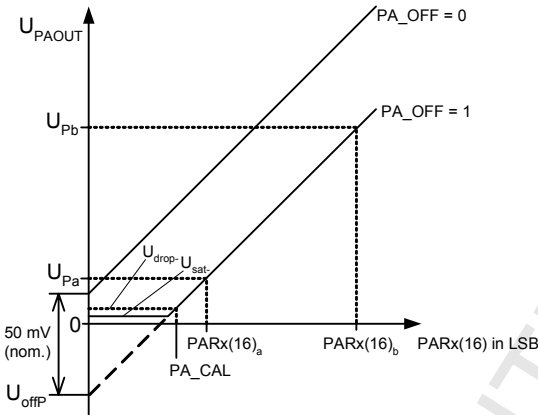
*Note: Setting **PA\_OFF1** = 1 during normal operation without any digital offset correction causes saturation effects.*

*Note: Due to quantization errors the accuracy of the **PA\_CAL1** value depends on the proper selection of PARx(16)<sub>1</sub> and PARx(16)<sub>2</sub>.*

*Note: It has to be ensured that the power amplifier does not output any power for approximately  $U_{\text{PAOUT}} \leq U_{\text{drop-}} + 10 \text{ mV}$ . For the design of the power ramp sequences the defined minimum output voltage of  $U_{\text{drop-}}$  has to be considered (refer to [Table 12-51 Specification of PAOUTOF1 \(on Page 1377\)](#)).*

**CONFIDENTIAL**

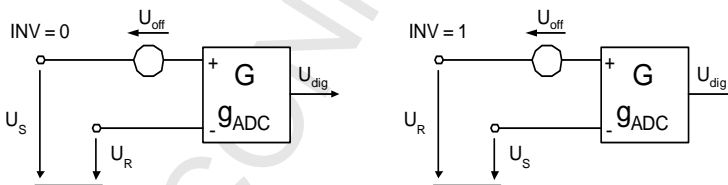
**Figure 10-28 PAOUT Offset Measurement**



### 10.5.8 Offset Calibration of Measurement Interface

For each measurement path the offset introduced by the pre-amplifier and the ADC itself can be determined and eliminated if the measurement is carried out twice, the first time with **MEAS\_CTRL1.INV** = 0 and the second time with **INV** = 1. In this case a simple offset model with one offset source  $U_{off}$  comprising the offsets of the pre-amplifier and the ADC is assumed as shown in **Figure 10-29**. Measurements with **INV** = 1 are indicated by an asterisk (\*).

**Figure 10-29 Equivalent Network for Offset Calculation**



The output voltage of the ADC with **INV** = 0 is given by

$$U_{dig} = g_{ADC} \cdot G \cdot (U_S - U_R + U_{off}) \quad [32]$$

and with **INV** = 1 by

$$U_{dig}^* = -g_{ADC} \cdot G \cdot (U_R - U_S + U_{off}) \quad [33]$$

CONFIDENTIAL

Measurement Interface

$U_{\text{off}}$  can easily be eliminated by calculating

$$U_{\text{dig}}' = (U_{\text{dig}} - U_{\text{dig}}^*)/2 \quad [34]$$

and the digital correction value  $U_{\text{dig,off}}$  which has to be subtracted from the measurement result can be calculated by

$$U_{\text{dig,off}} = (U_{\text{dig}} + U_{\text{dig}}^*)/2. \quad [35]$$

### 10.5.8.1 Gain Calibration of Measurement ADC

In gain self-calibration mode ADCCAL a known voltage is applied to the pins M0 and M9 and the ADC gain can be calculated by

$$g_{\text{ADC}} = \frac{U_{\text{dig,ADCCAL}} - U_{\text{dig,0}}}{U_{\text{M0}} - U_{\text{M9}}}. \quad [36]$$

If the pins M0 and M9 are shortened the actual offset  $U_{\text{dig,0}}$  (refer to [Equation \[2\] \(on Page 723\)](#)) of the ADC can be determined, too.

*Note: During measurement mode ADCCAL the electrical characteristics of the pins M0 and M1 are different to the characteristics during measurement mode M0 and M1!*

*Note: This calibration method can be used to enhance measurement accuracy of other measurement modes. However, with ADCCAL only the gain of the ADC can be calibrated and not the gain of the pre-amplifier.*

### 10.5.9 BPI Bus Register Descriptions

All clocks necessary for the measurement interface are derived from the master clock  $\text{clk}_{\text{bus}}$ .

The register addresses are in [Table 12-4 Address Mapping of X-Bus Peripherals \(on Page 1285\)](#).



**CONFIDENTIAL**

**Measurement Interface**

### 10.5.9.1 Measurement Control Registers

These registers control the measurement operating modes and the data transfer.

#### MEAS\_CTRL1

#### Measurement Control Register 1

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES ERV ED		TC		C SEL		RESERVED			INV				MX		

Field	Bits	Type	Description
<b>MX</b>	5:0	rw	<p>Measurement mode 00<sub>H</sub> Off (Default) Other values Refer to <a href="#">Table 10-19 Measurement Switch Settings (on Page 720)</a>, do not use any combination not specified this table.</p> <p><i>Note: Due to the pre-amplifier and multiplexer settling time, the measurement mode has to be set prior to setting the start flag if this settling time is longer than the acquisition delay (refer to <a href="#">Table 12-48 ADC Characteristics (on Page 1373)</a>).</i></p>
<b>INV</b>	6	rw	<p><b>Inversion of Pre-Amplifier Input</b> The input voltage of the pre-amplifier is: 0 Not inverted 1 Inverted</p> <p>If two measurements with and without inversion are carried out, the offset of the pre-amplifier and ADC can be determined and eliminated for a dedicated measurement mode. This is valid for all modes which use the ADC input A. <b>INV</b> only has an effect if the ADC input A is used.</p>
<b>CSEL</b>	11	rw	<p><b>Fast Settling of Pre-Amplifier</b> 0 Fast settling disabled, capacitor C1 of the pre-amplifier is used 1 Fast settling enabled, capacitor C2 with a lower capacitance is used. For the PWPA Measurement Mode <b>CSEL</b> must = 1.</p>

Field	Bits	Type	Description
<b>TC</b>	14:12	rw	<b>Current Source for Current Measurement Mode</b> 000 $I_S = I_R = 0$ : Voltage mode (Default) 001 $I_S = 30 \mu A, I_R = 60 \mu A$ : Current mode 010 $I_S = 60 \mu A, I_R = 60 \mu A$ : Current mode 011 $I_S = 90 \mu A, I_R = 60 \mu A$ : Current mode 100 $I_S = 120 \mu A, I_R = 60 \mu A$ : Current mode 101 $I_S = 150 \mu A, I_R = 60 \mu A$ : Current mode 110 $I_S = 180 \mu A, I_R = 60 \mu A$ : Current mode 111 $I_S = 210 \mu A, I_R = 60 \mu A$ : Current mode Only use the Current Measurement Mode for the measurements defined in <a href="#">Table 10-19 Measurement Switch Settings (on Page 720)</a> . Current sources are switched on as soon as <b>TC</b> is set to any value greater than $0_H$ . <b>TC</b> has to be set to $0_H$ to turn current sources off.
<b>RESERVED</b>	15, 10:7	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**Measurement Interface**

## MEAS\_CTRL2

### Measurement Control Register 2

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
START	RESERVED	ADCON	ENTRIG	ENSTOP	RESERVED	MXREF				BUFSIZE			FREQ		

Field	Bits	Type	Description
FREQ	2:0	rw	<p><b>ADC Sampling Rate Control</b></p> <p>000 Single shot mode (Default)</p> <p>001 Repetitive mode with <math>f_M \cdot S / (32 \cdot 640)</math> (typical 0.39 kHz)</p> <p>010 Repetitive mode with <math>f_M \cdot S / (32 \cdot 320)</math> (typical 0.78 kHz)</p> <p>011 Repetitive mode with <math>f_M \cdot S / (32 \cdot 160)</math> (typical 1.56 kHz)</p> <p>100 Repetitive mode with <math>f_M \cdot S / (32 \cdot 80)</math> (typical 3.13 kHz)</p> <p>101 Repetitive mode with <math>f_M \cdot S / (32 \cdot 40)</math> (typical 6.25 kHz)</p> <p>110 repetitive mode with <math>f_M \cdot S / (32 \cdot 20)</math> (typical 12.50 kHz)</p> <p>111 Repetitive mode with <math>f_M \cdot S / (32 \cdot 15)</math> (typical 16.67 kHz)</p> <p><math>f_M</math> is the peripheral master clock frequency <math>clk\_bus</math>.  <math>f_M = 52</math> MHz (typical) and <math>S = K/L</math>, <math>K = 02_H</math>, and <math>L = 0D_H</math> (typical). Refer to <a href="#">Section 10.5.12 (on page 748)</a>.</p> <p>If <b>ENSTOP</b> = 1, successive conversion is stopped after the number of measurements defined in <b>BUFSIZE</b> until it is started again by <b>START</b> or the ADCTRIG signal.</p> <p>The ADC performs measurements adjusted in <a href="#">MEAS_CTRL1.MX</a> with frequency defined in <b>FREQ</b>.</p>

Field	Bits	Type	Description
<b>BUFSIZE</b>	5:3	rw	<b>Size of Output Buffer</b> 000 <b>MEAS_DATA0</b> (Default) 001 <b>MEAS_DATA0..MEAS_DATA1</b> 010 <b>MEAS_DATA0..MEAS_DATA2</b> 011 <b>MEAS_DATA0..MEAS_DATA3</b> 100 <b>MEAS_DATA0..MEAS_DATA4</b> 101 <b>MEAS_DATA0..MEAS_DATA5</b> 110 <b>MEAS_DATA0..MEAS_DATA6</b> 111 <b>MEAS_DATA0..MEAS_DATA7</b>
<b>MXREF</b>	9:6	rw	<b>Reference Measurement Mode</b> 00 <sub>H</sub> No second measurement(Default) Other values Refer to <b>Table 10-19 Measurement Switch Settings (on Page 720)</b> , do not use any combination not specified this table. For <b>MXREF</b> > 0 <sub>H</sub> : 1. A measurement with the mode defined in <b>MEAS_CTRL1.MX</b> is carried out 2. A measurement defined by <b>MXREF</b> is carried out. The <b>MX</b> measurement result is always stored first, then the <b>MXREF</b> measurement result is stored in the <b>MEAS_DATAx</b> with the next higher index. The <b>MEAS_STAT.READY</b> flag is set to 1 and a RDYIRQ interrupt is generated depending on the value of <b>BUFSIZE</b> . <b>MEAS_STAT.WPTR</b> indicates where the last value was written to. For <b>MXREF</b> = 0 <sub>H</sub> no second measurement is carried out. For <b>MXREF</b> > 0 <sub>H</sub> it is recommend to set <b>BUFSIZE</b> to 1, 3, 5 or 7. If <b>MEAS_CTRL1.MX</b> = 0 <sub>H</sub> , <b>MXREF</b> is ignored.

Field	Bits	Type	Description
<b>ENSTOP</b>	11	rw	<p><b>Circular Output Buffer Disable</b></p> <p>0 The output buffer operates as a circular buffer with size defined in <b>BUFSIZE</b>.</p> <p><i>Note: The results of a series of conversions are stored in increasing order in <b>MEAS_DATAx</b> beginning at <b>MEAS_DATA0</b> and ending at <b>MEAS_DATAx[BUFSIZE]</b>. The following result is stored in <b>MEAS_DATA0</b> again and circular operation is going on.</i></p> <p><i>After a change in any of the <b>MEAS_CTRL1</b> bit fields (except <b>START</b> and <b>ENSTOP</b>), the next measurement result is always written to register <b>MEAS_DATA0</b>.</i></p> <p><i>The <b>MEAS_STAT.WPTR</b> field indicates the index of the <b>MEAS_DATAx</b> where the latest value was written to.</i></p> <p><i>Whenever a new value is ready for reading in the with the index defined in <b>BUFSIZE</b>, the <b>MEAS_STAT.READY</b> flag is set to 1 and a <b>RDYIRQ</b> interrupt is generated.</i></p>
			<p>1 In the Repetitive Mode:</p> <p>The output buffer operates as a linear buffer with size defined in <b>BUFSIZE</b>. The results of a series of conversions are stored in increasing order in <b>MEAS_DATAx</b> beginning at <b>MEAS_DATA0</b> after successive conversion is started by setting <b>MEAS_CTRL1.START</b> or via the signal <b>ADCTRIG</b>. The last result is written to <b>MEAS_DATAx[BUFSIZE]</b> and successive conversion is stopped until it is started again.</p> <p><i>Note: The <b>MEAS_STAT.WPTR</b> field, <b>MEAS_STAT.READY</b> flag, and <b>RDYIRQ</b> interrupt show the same behavior as they do when <b>ENTRIG</b> = 0.</i></p>
			<p>1 In the Single Shot Mode:</p> <p>Only <b>MEAS_DATA0</b> is used and is independent from the buffer size defined in <b>BUFSIZE</b>. If <b>BUFSIZE</b> is greater than 0, no <b>RDYIRQ</b> interrupt is generated.</p>

Field	Bits	Type	Description
<b>ENTRIG</b>	12	rw	<p><b>Triggered Mode Enable</b></p> <p>0 Triggered Mode Disabled</p> <p>1 Conversion is started when a rising edge occurs on the trigger signal ADCTRIG provided by the GSM timer unit.</p> <p><i>Note: Setting <b>MEAS_CTRL1.START</b> has no effect when <b>ENTRIG</b> = 1.</i></p> <p>If in the <i>Single Shot Mode</i>, a single conversion is carried out whenever a rising edge of ADCTRIG occurs.</p> <p>If in the <i>Repetitive Mode</i>, successive conversions starts with a rising edge of ADCTRIG.</p>
<b>ADCON</b>	13	rw	<p><b>ADC Power Save Control</b></p> <p>0 The ADC and pre-amplifier are in the Power Save Mode and no conversion is performed.</p> <p>1 The ADC and pre-amplifier are powered up and is ready for start of conversion.</p> <p><i>Note: <b>ADCON</b> has to be set to 1 in advance to any measurement due to the wake-up time according to <a href="#">Table 12-48 ADC Characteristics (on Page 1373)</a>.</i></p>
<b>START</b>	15	rw	<p>0 No action. <b>START</b> is always cleared when read.</p> <p>1 Initiates conversion:</p> <ul style="list-style-type: none"> <li>• In the single shot mode a single conversion is performed when <b>START</b> is set to 1.</li> <li>• In repetitive mode successive conversion is started when <b>START</b> is set to 1.</li> </ul> <p><i>Note: Setting <b>START</b> has no effect if <b>ENTRIG</b> is set to 1. If <b>START</b> is set to 1 during a conversion, it is ignored.</i></p> <p><i>The measurement mode (<b>MEAS_CTRL1.MX</b>, <b>MEAS_CTRL1.INV</b>, <b>MEAS_CTRL1.CSEL</b>, <b>MEAS_CTRL1.TC</b>, <b>MEAS_CTRL2.FREQ</b>, <b>MEAS_CTRL2.BUFSIZE</b>, etc.) must be set properly in advance to setting <b>START</b> to 1 to ensure settling and wake-up times.</i></p>

**CONFIDENTIAL**

**Measurement Interface**

Field	Bits	Type	Description
RESERVED	14, 10	r	Reserved; these bits must be left at their reset values.

CONFIDENTIAL

**CONFIDENTIAL**

**Measurement Interface**

### 10.5.9.2 Measurement Status Register

This register contains the READY flag indicating if a new value can be read and a BUSY flag indicating if a conversion is in progress.

#### MEAS\_STAT

#### Measurement Status Register

**Reset value: 4000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
READY	BUSY	RESERVED	RESERVED										WPTR		

Field	Bits	Type	Description
WPTR	2:0	rh	<b>Write Pointer</b> Indicates the <b>MEAS_DATAx</b> index where the latest valid result was written to. 000 <b>MEAS_DATA0</b> contains valid data 001 <b>MEAS_DATA0</b> and <b>MEAS_DATA1</b> contain data ... 111 <b>MEAS_DATA0</b> to <b>MEAS_DATA7</b> contain data <b>WPTR</b> is updated whenever valid data is written to <b>MEAS_DATA0</b> to <b>MEAS_DATA7</b> .
BUSY	14	rh	<b>Busy Flag</b> 0 ADC is free 1 An ADC operation is in progress or the ADC is in the Power Save Mode. <i>Note: No new ADC conversion should be started while <b>BUSY</b> is set.</i>
READY	15	rh	<b>Data Ready</b> 0 No data is available for reading. <b>READY</b> is cleared by hardware after any of the <b>MEAS_DATAx</b> registers are read. 1 Data is available for reading. <b>READY</b> flag is set and a RDYIRQ interrupt is generated whenever a new value is ready for reading in <b>MEAS_DATAx</b> . The index is defined in <b>MEAS_CTRL2.BUFSIZE</b> .
RESERVED	12:3	r	Reserved; these bits must be left at their reset values.



CONFIDENTIAL

Measurement Interface

### 10.5.9.3 Measurement Data Registers 0-7

Data from the ADC is transferred via the eight Measurement Data Registers to the X-Bus bus.

MEAS\_DATAx

MEAS\_DATA0

MEAS\_DATA1

MEAS\_DATA2

MEAS\_DATA3

MEAS\_DATA4

MEAS\_DATA5

MEAS\_DATA6

MEAS\_DATA7

Measurement Data Registers

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				DATAx											

Field	Bits	Type	Description
<b>DATAx</b> (x = 0 to 7)	11:0	rh	<b>ADC Data 0..7</b> The number of registers used depends on <a href="#">MEAS_CTRL2.BUFSIZE</a> .
<b>RESERVED</b>	15:12	r	Reserved; these bits must be left at their reset values.

### 10.5.9.4 Measurement Peripheral Clock Control Register

The internal peripheral clock clk\_kernel\_2 for the digital part of the measurement interface and of the reference voltage generation can be adjusted via a fractional clock divider with the rational factor K/L. The numerator K and the denominator L can be programmed in the [MEAS\\_CLK](#) register. This register is also part of the measurement X-Bus interface. In this way the clock clk\_kernel\_2 can be a constant, independent from the variable peripheral master clock clk\_bus.

MEAS\_CLK

Measurement Peripheral Clock Control Register

Reset value: 0100<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	L					RESERVED					K				

**CONFIDENTIAL**

**Measurement Interface**

Field	Bits	Type	Description
<b>K</b>	0:4	rw	<b>Numerator of Fractional Divider</b> $K = 0_D \dots 31_D$ Setting K depends on the frequency of clk_bus according <a href="#">Table 10-29 K and L Values for Various Bus Clock Frequencies (on Page 749)</a> . K must always be $< L$ . $K = L$ is not allowed, except when $K = L = 0$ . If $K = L = 0$ , clk_kernel_2 is equal to clk_bus. <i>Note: K has to be programmed prior to any action of the ADC or band-gap.</i>
<b>L</b>	14:8	rw	<b>Denominator of Fractional Divider</b> $L = 0_D \dots 127_D$ L is set in dependence of the frequency of clk_bus according <a href="#">Table 10-29</a> . <i>Note: L has to be programmed prior to any action of the ADC or band-gap.</i>
<b>RESERVED</b>	7:5, 15	r	Reserved; these bits must be left at their reset values.

### 10.5.9.5 Interrupts

The following interrupt requests (service requests) are generated by the measurement BPI. Refer to [Section 8.1 TEAKLite Interrupt Unit \(on Page 337\)](#).

**Table 10-22 Measurement BPI Interrupt Requests (Service Requests)**

Interrupt Request	Description
RDYIRQ	<b>Ready Interrupt Request</b> Indicates that data can be read from the <a href="#">MEAS_DATAx</a> registers. When a new data value is ready for reading in the <a href="#">MEAS_DATAx[MEAS_CTRL2.BUFSIZE]</a> registers. The <a href="#">MEAS_STAT.READY</a> flag is set to 1 and a RDYIRQ interrupt is generated

### 10.5.9.6 System Registers

The measurement interface and the analog control block (refer to [ANA\\_CTRLx \(on Page 751\)](#)) share one BPI and they have the following system registers:

- Service Request Control
- Peripheral Clock Control  
[MEAS\\_CLK](#)  
Measurement Interface Clock Control Register (refer to [Page 741](#)).
- Module Identification  
[MEAS\\_ID](#)  
Measurement Interface Identification Register (refer to [Page 744](#)).

**CONFIDENTIAL**

**Measurement Interface**

### 10.5.9.6.1 Measurement & Analog Identification Register

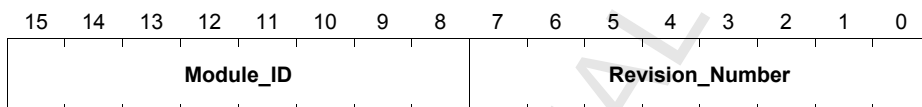
This is the ID for both the measurement interface and the analog control part.

**MEAS\_ID**

**Measurement Identification Register**

**Reset value:**

**For E-GOLDRadio Versions before V3.0: 2412<sub>H</sub>**



Field	Bits	Type	Description
Revision_Number	0:7	r	<b>Measurement Revision Number</b> These hard-wired bits are used for module revision numbering.
Module_ID	8:15	r	<b>Measurement Identification Number</b> These hard-wired bits are used for module identification numbering.

## 10.5.10 Special Operating Modes and Examples

### 10.5.10.1 Triggered and Repetitive Measurements

To measure at definite time, for example, during a burst, an A-to-D conversion can be started by the trigger signal ADCTRIG, which is generated by the programmable GSM Timer Unit. To enable this mode the bit **MEAS\_CTRL2.ENTRIG** must be equal to 1.

To get a series of measurement values in a dedicated measurement mode, repetitive conversion can be enabled by setting the field **MEAS\_CTRL2.FREQ** to one of its possible sampling rates. Repetitive conversion can be started either by:

- If **ENTRIG** = 0, setting the **MEAS\_CTRL2.START** flag
- If **ENTRIG** = 1, the signal ADCTRIG.

**MEAS\_CTRL2.BUFSIZE** controls whether the **MEAS\_STAT.READY** flag is set after each conversion or after a series of up to 8 conversions (that are stored in **MEAS\_DATAx**).

The bit **MEAS\_CTRL2.ENSTOP** controls whether the output buffer operates as linear or circular buffer.

In the triggered mode (**ENTRIG** = 1) measurement is started when a rising edge on signal ADCTRIG occurs. Setting **START** has no effect while **ENTRIG** = 1. The user has

CONFIDENTIAL

Measurement Interface

to be sure that the GSM Timer Unit is programmed to set ADCTRIG at the desired points in time, for example, at the start of a burst taking in account any delays occurring in the system.

In the repetitive mode successive measurements with frequency defined in **FREQ** are carried out until **FREQ** is set to 0<sub>H</sub> or **ENSTOP** is set to 1.

### 10.5.10.2 Measurement of VBAT During Transmission Bursts

To measure the battery voltage at several instances in time during a transmission burst on pin M8 consider the following example:

The battery voltage should be automatically measured at 6 equidistant instances in time during a dedicated burst. The six results should be available for reading after completion of measurement operation has been indicated via an interrupt request.

Use the configurations in [Table 10-23](#) and [Table 10-24](#).

**Table 10-23** **MEAS\_CTRL1 = 0009**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES ERV ED	TC			C SEL	RESERVED				INV	MX					
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1

**Table 10-24** **MEAS\_CTRL2 = 382E**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STA RT	RES ERV ED	ADC ON	EN TRIG	EN STOP	RES ERV ED	MXREF				BUFSIZE			FREQ		
0	0	1	1	1	0	0	0	0	0	1	0	1	1	1	0

The GSM Timer Unit has to be programmed so that the ADCTRIG signal provides a rising edge at the beginning of the burst. The falling edge of ADCTRIG is not evaluated.

When the RDYIRQ interrupt occurs, the results can be read from **MEAS\_DATA0** ... **MEAS\_DATA5**.

### 10.5.10.3 Measurement of PWPA During Tail Symbols of a Burst

To measure the output transmission power during the tail symbols at the beginning and end of one dedicated burst in every frame, use the configurations in [Table 10-25](#) and [Table 10-26](#).

**Table 10-25** [MEAS\\_CTRL1](#) = 080B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES ERV ED	TC			C SEL	RESERVED				INV	MX					
0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	1

**Table 10-26** [MEAS\\_CTRL2](#) = 3008

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STA RT	RES ERV ED	ADC ON	EN TRIG	EN STOP	RES ERV ED	MXREF				BUFSIZE			FREQ		
0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0

The GSM Timer Unit has to be programmed that ADCTRIG signal provides rising edges at the tail symbols at the beginning and the end of the burst in every frame considering all relevant delays. The falling edge of ADCTRIG is not evaluated.

When the RDYIRQ interrupt occurs, the results can be read from [MEAS\\_DATA0](#) ... [MEAS\\_DATA1](#). They will be overwritten in the following frame.

To stop measurements, set [MEAS\\_CTRL2.ENTRIG](#) to 0 or change configuration of the GSM Timer Unit.

#### 10.5.10.4 Measurement of Battery Technology

To carry out a single measurement of the battery technology in the current mode on pin M0, use the configurations in [Table 10-27](#) and [Table 10-28](#).

**Table 10-27** [MEAS\\_CTRL1 = 3001](#)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES ERV ED	TC			C SEL	RESERVED				INV	MX					
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1

**Table 10-28** [MEAS\\_CTRL2 = A000](#)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STA RT	RES ERV ED	ADC ON	EN TRIG	EN STOP	RES ERV ED	MXREF				BUFSIZE			FREQ		
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

If [MEAS\\_STAT.READY](#) is set to 1 or a RDYIRQ interrupt has occurred, the result can be read from [MEAS\\_DATA0](#).

#### 10.5.10.5 Power Save Features

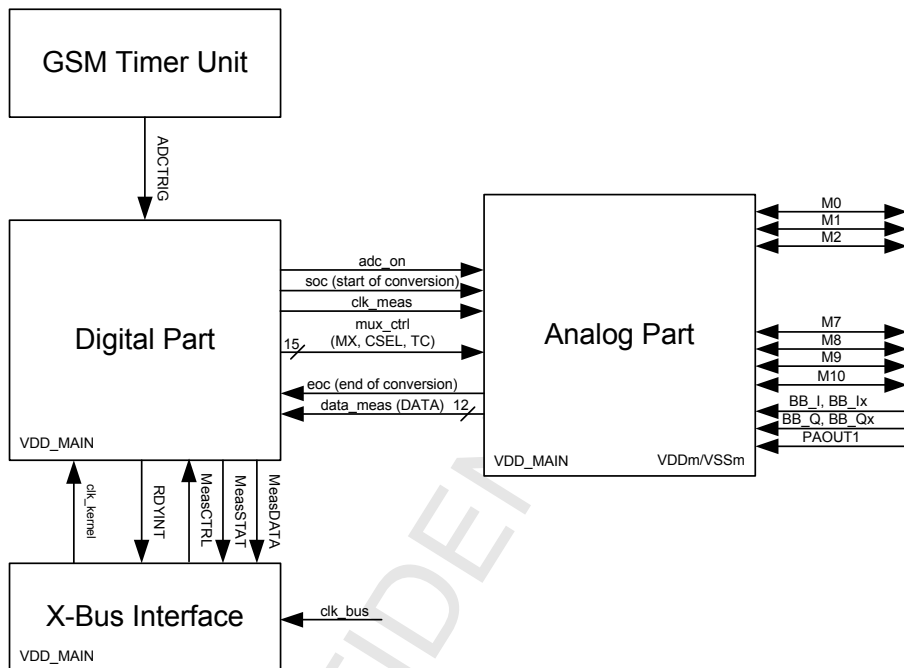
The digital part of the measurement interface is switched off by disabling the clock of the bus interface ([MEAS\\_CLK.K](#) = 0).

Power control for the complete analog block of the measurement interface is managed via the bit [MEAS\\_CTRL2.ADCON](#), which can switch the ADC and pre-amplifier to the power save mode. Additional power down features for the mixed signal section are explained in [Section 10.4.2 Standby Clock Control Unit \(SCCU\) \(on Page 685\)](#).

#### 10.5.11 System Block Diagram of Measurement Interface

The measurement interface consists of two parts, the analog part and the digital part. Both parts communicate via the signals shown in [Figure 10-30](#). The measurement Interface is connected to the X-Bus via an X-Bus interface. The signal ADCTRIG is provided by the GSM Timer Unit, which is not part of the measurement interface. Usually, ADCTRIG is used to control measurement operation for battery voltage or transmission power (PWPA).

Figure 10-30 Logic Block Diagram of Measurement Interface



### 10.5.12 Clock Control of Measurement Interface

All clocks necessary for the measurement interface (and the reference voltage generation) are derived from the peripheral master clock  $clk\_bus$  using clock dividers as it is shown in [Figure 10-31](#).

This  $clk\_bus$  is divided by a fractional divider with factor  $K/L$  giving  $clk\_kernel\_2$ .

This  $clk\_kernel\_2$  is divided by  $M = 8$  in the digital part and the output is  $clk\_meas$ .  $clk\_kernel\_2$  determines most of the timing characteristics of the ADC (refer to [Table 12-48 ADC Characteristics \(on Page 1373\)](#)). The programming of the fractional divider for different frequencies of the clock  $clk\_bus$  are listed in [Table 10-29 K and L Values for Various Bus Clock Frequencies \(on Page 749\)](#).  $K$  and  $L$  are in the **MEAS\_CLK** register.

*Note: For normal operation, that is,  $clk\_bus = 52\text{ MHz}$ , the fractional divider has to be programmed with  $K = 02_H$ ,  $L = 0D_H$  to have a  $clk\_kernel\_2$  of 8 MHz and a  $clk\_meas$  of 1 MHz which is the usual operating condition for the ADC.*



CONFIDENTIAL

Measurement Interface

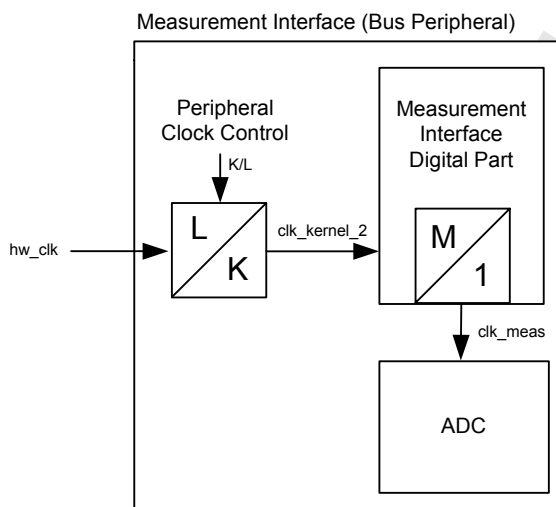
The programming of the measurement interface has to be done in the following order:

1. K and L in the **MEAS\_CLK** register are programmed.
2. **MEAS\_CTRL1/MEAS\_CTRL1** are programmed
3. An A-to-D conversion can be started.

**Note: Never start a A-to-D conversion before programming the appropriate frequency of `clk_kernel_2`.**

*Note: The `band_gap` needs the `clk_kernel_2` before it is powered up.*

**Figure 10-31 Clock Control for Measurement Interface**



Refer to the Clock Domain in [System Integration \(on Page 716\)](#).

**Table 10-29 K and L Values for Various Bus Clock Frequencies**

clk_bus in MHz	K	L
78.0	4 <sub>D</sub>	39 <sub>D</sub>
52.0	2 <sub>D</sub>	13 <sub>D</sub>
26.0	4 <sub>D</sub>	13 <sub>D</sub>
13.0	8 <sub>D</sub>	13 <sub>D</sub>
X	X	X

## **10.6 Analog Control Registers**

### **10.6.1 Functional Overview**

The analog part contains several dedicated registers which are provided for specific control purposes of the analog blocks.

- Control of reference voltage generation
- Control of nominal common mode voltage of GSM TX path
- Control of offset voltage for power amplifier control
- Identification.

CONFIDENTIAL

**CONFIDENTIAL**

**Analog Control Registers**

**ANA\_CTRLx**

### 10.6.2 Analog Control Register 1

**ANA\_CTRL1**

**Analog Control Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES ERV ED	PA OPM 1	RES ERV ED	RESERVED				TX REF _PU	RES ERV ED	RES ERV ED	RES ERV ED	TX DIS	RES ERV ED	PA OFF 1	BG PW UP	RX REF _PU

Field	Bits	Type	Description
<b>RXREF_PU</b>	0	rw	<b>RX Local Reference Buffer Switch</b> 0 Local reference buffer in baseband receive path is off when signal RXON = 0. 1 Local reference buffer in baseband receive path is on when RXON = 0.
<b>BG_PWUP</b>	1	rw	<b>Band-Gap Power Control</b> 0 Power down of reference generation 1 Normal operation of reference generation. clk_kernel_2 must be running before the band-gap can be enabled via <b>BG_PWUP</b> .  <i>Note: Settling time (wake-up time) of reference voltage has to be considered according <a href="#">Table 12-53 Reference Voltage Generation (Band-Gap)</a> (on Page 1379).</i>
<b>PA_OFF1</b>	2	rw	<b>PAOUT1 Analog Offset Correction</b> 0 No Analog offset correction 1 Analog offset correction with nominal -50 mV
<b>TX_DIS</b>	4	rw	<b>Disable Analog Part of Modulator Unit</b> 0 Analog part is switched on and off depending on the signal TXON 1 Analog part is always off  <i>Note: TX_DIS is set to 1 in those cases, where the analog part of the on chip modulator isn't used, for example, if an external modulator is connected to the E-GOLDradio.</i>  <i>Note:</i>

**CONFIDENTIAL**

**Analog Control Registers**

Field	Bits	Type	Description
<b>TXREF_PU</b>	8	rw	<b>TX Local Reference Buffer Switch</b> 0 Local reference buffer in baseband transmit path is off when signal TXON = 0. 1 Local reference buffer in baseband transmit path is on when TXON = 0.
<b>PAOPM1</b>	14	rw	<b>PAOUT1 Output Buffer Operation Mode</b> 0 Output buffer in standard mode 1 Output buffer in enhanced transient performance mode
<b>RESERVED</b>	15, 13:9, 7:5, 3	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**Analog Control Registers**

### 10.6.3 Analog Control Register 2

**ANA\_CTRL2**

**Analog Control Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>TREF</b>				<b>RESERVED</b>						<b>PA_CAL1</b>					

Field	Bits	Type	Description
<b>PA_CAL1</b>	5:0	rw	<b>PAOUT1 Digital Offset Correction</b> 00 <sub>H</sub> No digital offset correction every DAC input value is 01 <sub>H</sub> Increased by 1 LSB 02 <sub>H</sub> Increased by 2 LSB ... 3F <sub>H</sub> Increased by 63 LSB <i>Note: Refer to <a href="#">Section 10.10.1.4 D-to-A Conversion and Post Filtering (on Page 798)</a> and <a href="#">Section 10.5.7.1 PA Control Hardware Offset Measurement PAOUTOF1 (on Page 729)</a>.</i>
<b>TREF</b>	15:12	rw	<b>Common Mode Voltage for Baseband TX Path<sup>1)</sup></b> 1111 0.900 V (typical) 0000 0.955 V (typical) (default) 0001 1.010 V (typical) 0010 1.065 V (typical) 0011 1.120 V (typical) 0100 1.175 V (typical) 0101 1.230 V (typical) 0110 1.285 V (typical) 0111 1.340 V (typical)
<b>RESERVED</b>	11:6	r	Reserved; these bits must be left at their reset values.

<sup>1)</sup> (BB\_I + BB\_Ix)/2 and (BB\_Q + BB\_Qx)/2 in TX case

CONFIDENTIAL

**CONFIDENTIAL**

**Keypad**

## 10.7 Keypad

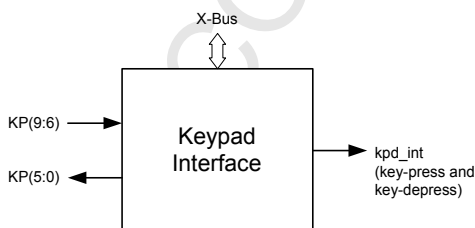
History	
Current version: Rev. 1.06, 2005-11-04	
Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.01	
<b>Page 755</b>	Updated <b>System Integration</b> : WS00006682
Changes for Rev. 1.04	
<b>Page 759</b>	Removed reference to ECO block in <b>Section 10.7.3 Keypad Port</b> WS00008455
Changes for Rev. 1.06	

### System Integration:

- Supply domain: VDD\_MAIN
- Chip internal interfaces:
  - Clock domain: Refer to **Section 10.4.1.3 Sub-System Clocks and Enables (on Page 661)** and see **Figure 10-11 Clock and Enable Generation for MCU Sub-System (on Page 668)**.
  - Bus domain: X-Bus
- Interrupt sources:
- Monitor Pins: Refer to **Section 11.8.10 Internal Signal Monitoring (on Page 1209)**

### 10.7.1 Description

**Figure 10-32 Block Symbol of the Keypad Interface**



The keypad interface is connected to the X-Bus together with the AFC Unit, XBIU, and the Shared Memory Register using a single Bus Interface.

**CONFIDENTIAL**

**Keypad**

The keypad interface is a 10-bit port with four input, six output pins, and two addressable 16 bit interface registers.

CONFIDENTIAL



**CONFIDENTIAL**

**Keypad**

## 10.7.2 Keypad Registers

### 10.7.2.1 Keypad Identification Register

**KBID**

**Keypad Identification Register**

**Reset values: A802<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>Module_ID</b>								<b>Revision_Number</b>							

Field	Bits	Type	Description
<b>Revision_Number</b>	0:7	r	<b>Keypad Revision Number</b> These hard-wired bits are used for the module revision numbering.
<b>Module_ID</b>	8:15	r	<b>Keypad Identification Number</b> These hard-wired bits are used for module identification numbering.

### 10.7.2.2 Keypad Input Register

**KBDINP**

**Keypad Input Register**

**Reset value: 000X<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>												<b>KYIN<sub>3</sub></b>	<b>KYIN<sub>2</sub></b>	<b>KYIN<sub>1</sub></b>	<b>KYIN<sub>0</sub></b>

Field	Bits	Type	Description
<b>KYIN<sub>x</sub></b> (x = 0 to 3)	0:3	r	<b>Keypad Input x</b> The reset value for this register depends on the value of <b>KYIN<sub>x</sub></b> .
<b>RESERVED</b>	15:4	r	Reserved for future use; these bits must be left at their reset values.

CONFIDENTIAL

Keypad

### 10.7.2.3 Keypad Output Register

KBDOUT

Keypad Output Register

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED										KY OUT 5	KY OUT 4	KY OUT 3	KY OUT 2	KY OUT 1	KY OUT 0

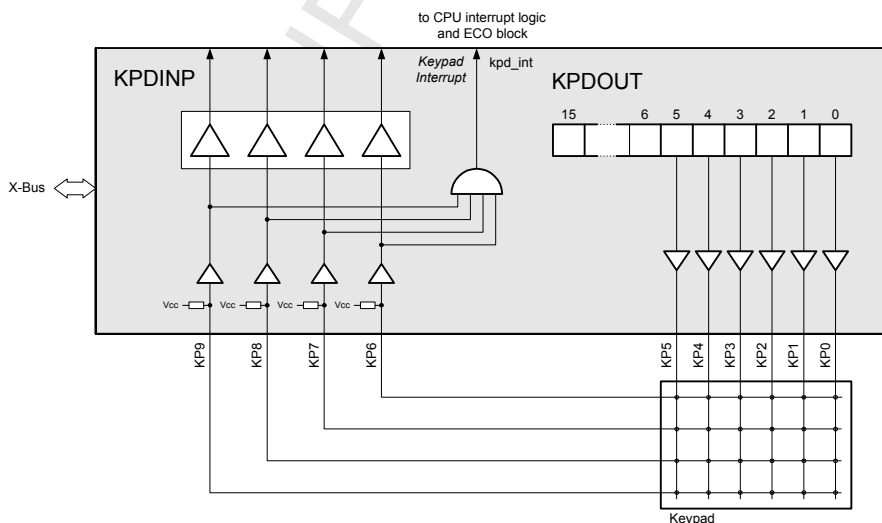
Field	Bits	Type	Description
KYOUTx (x = 0 to 5)	0:5	w	Keypad Output x Output pin values
RESERVED	15:6	r	Reserved for future use; these bits must be left at their reset values.

### 10.7.3 Keypad Port

The levels at the input pins KP(9:6) will be loaded into **KBDINP** with a read access. The levels which should show at the output pins KP(5:0) have to be loaded into **KBDOUT** with a write access. Pressing a key directly generates an interrupt to the MCU. Releasing this key generates an additional interrupt. By those means a length of a key press can be simply examined when measuring the time between this two interrupts.

**Figure 10-33** shows a block diagram of the keypad interface.

**Figure 10-33 Block Diagram of the Internal Keypad Port Circuitry**



**CONFIDENTIAL****Keypad**

The keypad port can be operated as follows:

As long as no input port pin (KP9...6) is connected to an output port pin (KP5...0) the input pins are pulled high internally if the pad is programmed with a pull up in the port control logic. In this case a high level appears on the keypad interrupt input of the controller.

If the controller wants to check the key status, it should load **KBDOUT** with 00<sub>H</sub> to start the keypad scan. If a key is pressed, both the corresponding keypad row and column will be connected to each other and the corresponding **KBDINP** register bit is set to LOW. Due to the LOW level the kpd\_int interrupt (active high, duration 2 X-Bus clock cycles) will be generated to the controller and to the SCCU block, which is then able to initiate an early wake-up procedure in case of sleep mode. Knowing the right row number the controller scans the six bits of **KBDOUT** by switching them from 1 to 0 step by step getting in this way also the right column number.

During normal operation the clock frequency of the keypad interface is 13 MHz, during sleep mode the 32 kHz sleep clock is provided.

Pins not to be used for the keypad can be configured to show either GPIO port or alternate functionality as described in **Chapter 3 Pin Descriptions**.

After a key-press IRQ is generated and the control logic scans the keypad to detect the key release. When the release of the same key is detected, a second interrupt is generated.

- There must be an instruction or NOP between a write to **KBDOUT** and a read from **KBDINP** to avoid pipeline effects.
- The interrupt to the CPU is generated even when the X-Bus clock is switched off.
- If the keypad port is not used the inputs are pulled high internally.  
**KBDINP** is not a real register, it is a set of four parallel drivers (with enable) which connect the input pins with the internal bus. A read access will open these drivers.
- Pressing two or more keys at the same time leads to a short circuit of two output pins and with this to a short circuit current. Reducing the short circuit current requires either external series resistors or a limitation of the short circuit time.
- The junction resistance of the keypad should be < 2 kΩ and the max. leakage current 5 μA.

CONFIDENTIAL

**CONFIDENTIAL**

**SIM Interface**

## 10.8 SIM Interface

History	
	Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.01	
<b>Page 761</b>	Updated <b>System Integration</b> :, WS00006682
all	“USIM” changed to “SIM” except in signal and directory names WS00007145
Changes for Rev. 1.03	
<b>Page 770</b>	Bit <b>ENT1END</b> removed in register <b>SIMIRQEN</b> WS00007674
Changes for Rev. 1.04	
<b>Page 768</b>	Correction to <b>SIMSTATUS.SIMDET</b> bit WS00008180
Changes for Rev. 1.06	

### System Integration:

- Supply domain: VDD\_MAIN
- Chip internal interfaces:
  - Clock domain: Refer to **Section 10.4.1.3 Sub-System Clocks and Enables (on Page 661)** and see **Figure 10-11 Clock and Enable Generation for MCU Sub-System (on Page 668)**.
  - Bus domain: X-Bus bus
- Interrupt sources: 3 regular controller interrupts:  
USIM\_ERR\_INT, USIM\_IN\_INT, USIM\_OK\_INT.
- Chip external signals related to this block (refer to **Chapter 3 Pin Descriptions** for pin configuration options): CCIO, CC\_VZ\_n, CC\_CLK, and CC\_RST.
- Monitor pins:

#### 10.8.1 Functional Overview

The SIM interface is compatible with the ISO 7816-3 IC Card standard on the issues required by the GSM 11.11 Phase 2+ standard. Features included in the interface are:

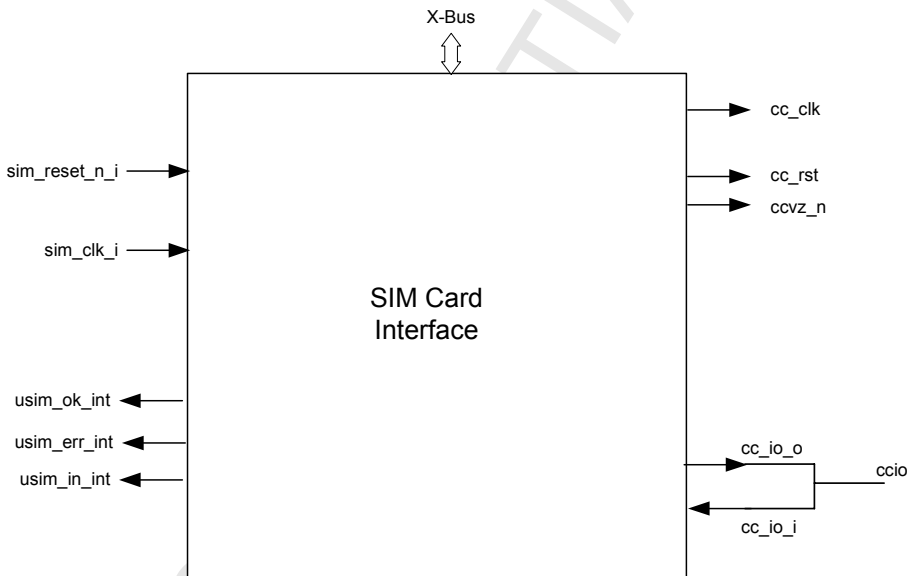
- SIM Card is provided with a 3.25 MHz clock (derived from 13 MHz clock).
- Automatic parity error detection and error signalling in RX mode.
- Automatic character repetition on parity errors in TX mode.

**CONFIDENTIAL**

**SIM Interface**

- Automatic switching between RX and TX mode.
- Automatic work waiting timer supervision for use with T=0 protocol.
- Supports enhanced speed SIM's as specified in GSM 11.11 - Phase 2+ using the SIM Baud Rate Factor settings.
- Supports both SW and HW controlled T=0 protocol.
- Work Waiting for T=0 mode.
- Supports 1 MHz SIM clock in low power mode.
- Supports GSM Phase 2 clock stop modes.
- Automatic power down for immediate SIM deactivation.

**Figure 10-34 Block Symbol of the SIM Card Interface<sup>1)</sup>**



Refer to **Figure 10-35**, the block diagram.

<sup>1)</sup> Refer to Clock Domain in **System Integration**: on **page 761**.

## CONFIDENTIAL

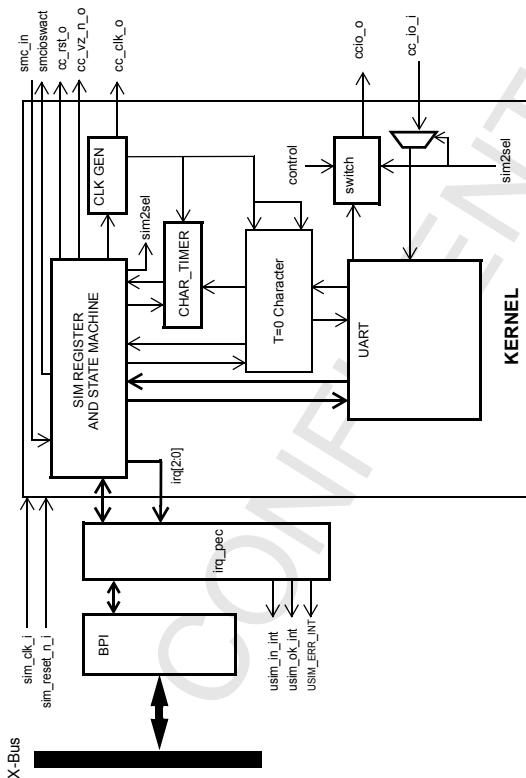
## SIM Interface

The SIM interface consists of 7 main modules:

- SIM UART
- T=0 Controller
- Clock generation unit.
- Register and state machine unit
- Character timer unit (Work Waiting Timer for T=0)

For an overview, see [Figure 10-35](#).

**Figure 10-35 SIM Card Interface Block Diagram<sup>1)</sup>**



<sup>1)</sup> Refer to Clock Domain in [System Integration](#): on [page 761](#).

**CONFIDENTIAL**

**SIM Interface**

## 10.8.2 SIM Register Overview

Register Group	Register Name	Register Symbol
System Register	SIM Identification Register	<b>SIMID</b>
Control and Status Registers	SIM Control Register	<b>SIMCTRL</b>
	SIM Baud Rate Factor Register	<b>SIMBRF</b>
	SIM Status Register	<b>SIMSTATUS</b>
	SIM Interrupt Enable Register	<b>SIMIRQEN</b>
	SIM RX Spacing Register	<b>SIMRXSPC</b>
	SIM TX Spacing Register	<b>SIMTXSPC</b>
	SIM Character Timer Registers	<b>SIMCHTIMERx</b>
Data Registers	SIM transmit data register	<b>SIMTX</b>
	SIM receive data register	<b>SIMRX</b>
	SIM instruction class register	<b>SIMINS</b>
	SIM parameter 3 register	<b>SIMP3</b>
	SIM Status Word 1 Register	<b>SIMSW1</b>
	SIM Status Word 2 Register	<b>SIMSW2</b>

## 10.8.3 Register Description

### 10.8.3.1 SIM Interface Identification Register

**SIMID**

**SIM Interface Identification Register**

**Reset values: 0012<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>Module_ID</b>								<b>Revision_Number</b>							

Field	Bits	Type	Description
<b>Revision_Number</b>	0:7	r	<b>SIM Interface Revision Number</b> These hard-wired bits are used for the CGU revision numbering.
<b>Module_ID</b>	8:15	r	<b>SIM Interface Identification Number</b> These hard-wired bits are used for CGU identification numbering.



**CONFIDENTIAL**

**SIM Interface**

### 10.8.3.2 SIM Control Register

The **SIMCTRL** register contains information for controlling the operation mode of the SIM interface except the clock signals used.

#### SIMCTRL

#### SIM Control Register

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	SMCIOSWACT	UARTON	CLKHIGH	CLKSEL	SIMON	SIMPDOWN	APDWN	RPTOFF	ERROFF	SIMRST	SIMVCC	SIMEN	SIMIOL	SIMT0	INCON

Field	Bits	Type	Description
<b>INCON</b>	1	rw	0 Direct convention. LSB transmitted/received first, logic 1 equals high level. 1 Inverse convention. MSB transmitted/received first, logic 1 equals low level.
<b>SIMT0</b>	2	rw	0 SIM runs in normal character based mode. This mode is used during ATR (Answer to Reset). 1 SIM runs in instruction mode. The T=0 protocol is handled in HW.
<b>SIMIOL</b>	2	rw	0 Sets SIM I/O to low level as long as this bit is reset. 1 SIM I/O line enabled.
<b>SIMEN</b>	3	rw	0 SIM interface disabled. 1 SIM interface enabled.
<b>SIMVCC</b>	4	rw	0 SIM supply voltage is removed ( $\overline{\text{CCVZ}}$ at high level). 1 SIM supply voltage is applied ( $\overline{\text{CCVZ}}$ at low level).
<b>SIMRST</b>	5	rw	0 SIMRST signal at low level. 1 SIMRST signal at high level.
<b>ERROFF</b>	6	rw	0 Enables error signalling during transmission and reception 1 Disables error signalling during transmission and reception.
<b>RPTOFF</b>	7	rw	0 Character retransmission is enabled. 1 Character retransmission is switched off.

**CONFIDENTIAL**

**SIM Interface**

Field	Bits	Type	Description
<b>APDWN</b>	8	rw	0 Automatic power down is disabled. 1 Automatic power down of the SIM occurs on a falling edge of CCIN.
<b>SIMPDWN</b>	9	rw	0 The SIM signals can be controlled manually. 1 When set, the SIM is powered down and the SIM signals stay disabled until SIMPDWN is reset. This also causes SIMEN to be disabled after a delay of 5 of the 13 MHz clock cycles.
<b>SIMON</b>	10	rw	0 The 3.25 MHz SIM clock is switched off. Can be used if the SIM supports clock stop mode. 1 The 3.25 MHz SIM clock is switched on.
<b>CLKSEL</b>	11	rw	0 SIM clock is 3.25 MHz. 1 SIM clock is 1.08 MHz. Must only be used between commands.
<b>CLKHIGH</b>	12	rw	0 SIM clock stops at a low level. 1 SIM clock stops at a high level.
<b>UARTON</b>	13	rw	0 UART, T=0 clocks are off. Use this mode between instructions to save power. 1 UART and T=0 clock is on.
<b>SMCSWACT</b>	14	rw	0 Smart card interface data output is connected to the SMC_IO pin. Output SMC_IOSW is high (if pin is not configured as GPIO) 1 Inverted Smart card interface data output is connected to SMC_IOSW. Pin SMC_IO is a tri-state input.
<b>RESERVED</b>	15	r	Reserved; these bits must be left at their reset values.

### 10.8.3.3 SIM Baud Rate Factor Register

This factor defines the baud rate used when communicating with the SIM card. On Reset the register is loaded with the value 93 (5D<sub>H</sub>) corresponding with the default values for D and F (D = 1 and F = 372). The BRF is calculated using the following equation:

$$BRF = \frac{F}{4D} \quad [36.1]$$

*Note: The SW must ensure that the BRF is an integer value and appropriate to the ISO 7816 and GSM specification, that is, BRF = 16 (10<sub>H</sub>) for F = 512, D = 8 and BRF 8 (08<sub>H</sub>) for F = 512, D = 16.*

**SIMBRF**

**SIM Baud Rate Factor Register**

**Reset value: 005D<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED									BRF						

Field	Bits	Type	Description
<b>BRF</b>	6:0	rw	<b>SIM Baud-Rate Factor</b> Legal values are 2 to 127.
<b>RESERVED</b>	15:7	r	Reserved; these bits must be left at their reset values.

CONFIDENTIAL

SIM Interface

### 10.8.3.4 SIM Status Register

**SIMSTATUS** shows which event caused the interrupt and the status of pin CCIN.

*Note: The reset value of **SIMSTATUS** is 0000<sub>H</sub> if the SIM card or Smart card is not connected and 0010<sub>H</sub> if the SIM card or Smart card is connected.*

#### SIMSTATUS

#### SIM Status Register

Reset value: See Note

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED							RESERVED	RESERVED	RESERVED	CHTMEOUT	SIMDET	T0END	OVRUN	PARINT	UARTOK

Field	Bits	Type	Description
UARTOK	0	rw	<b>UART Ready Interrupt</b> 0 No interrupt 1 One byte was received or transmitted without errors
PARINT	1	rw	<b>UART Parity Error Interrupt</b> 0 No error 1 Parity error occurred in receive or transmit direction
OVRUN	2	rw	<b>UART Overrun Error</b> 0 No error 1 New byte written to <b>SIMTX</b> before previous byte has been transmitted or a new byte was received before <b>SIMRX</b> was read
T0END	3	rw	<b>T=0 Instruction Ended</b> 0 Command not executed 1 Command executed. Refer to <b>SIMSW1</b> and <b>SIMSW2</b> for status.
SIMDET	4	rw	<b>SIM Present Indication</b> This bit can always be read. 0 SIM card is not present 1 SIM card is present (pin SMC_IN = 1)

**CONFIDENTIAL**

**SIM Interface**

Field	Bits	Type	Description
<b>CHTIMEOUT</b>	5	rw	<b>Character Timer Has Timed Out</b> 0 Has not timed out 1 The character time out as defined in the <b>SIMCHTIMERx</b> registers has expired. In T=0 mode this indicates that a WWT time out has occurred.
<b>RESERVED</b>	15:6	r	Reserved; these bits must be left at their reset values.

*Note: The reset value of **SIMSTATUS** is 0000<sub>H</sub> if the SIM card or Smart card is not connected and 0010<sub>H</sub> if the SIM card or Smart card is connected.*

*Note: The values of **SIMSTATUS** can only be written by software with the value zero.*

*Not all bits are writable by software:*

*To reset the bits **PARINT**, **OVRRUN**, **T0END**, and **CHTIMEOUT** the value 0000<sub>H</sub> must be written to the register. This does not effect **SIMDET** that reflects the level of the pin CCIN even if the interface has been disabled and does not effect the value of **UARTOK**.*

*If **UARTOK** is set (due to successful data transmission) **PARTINT**, **OVRRUN**, and **CHTIMEOUT** are cleared.*

**CONFIDENTIAL**

**SIM Interface**

### 10.8.3.5 SIM Interrupt Enable Register

These bits enable the interrupts shown in **SIMSTATUS**.

#### **SIMIRQEN**

#### **SIM Interrupt Enable Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>										<b>EN BWT TIME R</b>	<b>EN CHT IME R</b>	<b>EN T0E ND</b>	<b>EN OVR</b>	<b>EN PAR</b>	<b>EN OKI NT</b>

Field	Bits	Type	Description
<b>ENOKINT</b>	0	rw	<b>UARTOK Interrupt Enable</b> 0 Disabled 1 Enabled
<b>ENPAR</b>	1	rw	<b>PARINT Interrupt Enable</b> 0 Disabled 1 Enabled
<b>ENOVR</b>	2	rw	<b>OVRRUN Interrupt Enable</b> 0 Disabled 1 Enabled
<b>ENT0END</b>	3	rw	<b>T0END Interrupt Enable</b> 0 Disabled 1 Enabled
<b>ENCHTIMER</b>	4	rw	<b>Character Timer Interrupt Enable</b> 0 Disabled 1 Enabled
<b>ENBWTTIMER</b>	5	rw	<b>BWT Timer Interrupt Enable</b> 0 Disabled 1 Enabled
<b>RESERVED</b>	15:6	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**SIM Interface**

### 10.8.3.6 SIM Transmit Register

**SIMTX**

**SIM Transmit Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>								<b>SIMTX</b>							

Field	Bits	Type	Description
<b>SIMTX</b>	7:0	rw	Data byte to be transmitted to the SIM card.
<b>RESERVED</b>	15:8	r	Reserved; these bits must be left at their reset values.

### 10.8.3.7 SIM Receive Register

**SIMRX**

**SIM Receive Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>								<b>SIMRX</b>							

Field	Bits	Type	Description
<b>SIMRX</b>	7:0	rw	Data byte received from the SIM card.
<b>RESERVED</b>	15:8	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**SIM Interface**

### 10.8.3.8 SIM Instruction Class Register

This register contains the instruction code in the instruction class as described in the Protocol type T=0.

When this word is written the T=0 controller executes the instruction. Hence, this word must be written after **SIMP3**.

#### SIMINS

##### SIM Instruction Class Register

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED							INS DIR	INS							

Field	Bits	Type	Description
INS	7:0	rw	Data byte received from the SIM card.
INSDIR	8	rw	<b>Instruction Direction</b> 0 The T=0 controller sends data to the SIM. 1 The T=0 controller receives data from the SIM.
RESERVED	15:9	r	Reserved; these bits must be left at their reset values.

### 10.8.3.9 SIM Parameter 3 Register

#### SIMP3

##### SIM Parameter 3 Register

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED							P3								

Field	Bits	Type	Description
P3	7:0	rw	This value determines the number of bytes to receive or transmit depending on the bit <b>SIMINS.INSDIR</b> .
RESERVED	15:8	r	Reserved; these bits must be left at their reset values.



**CONFIDENTIAL**

**SIM Interface**

### 10.8.3.10 SIM Status Words

These registers contain the instruction status words SW1 and SW2 as described in the Protocol type T=0. These register can be read when a T0END interrupt has been received.

**SIMSW1** always holds the last procedure byte sent by the SIM. This can be used for debugging purposes if the SW implements a SIM timeout.

These registers are reset when a new instruction starts.

#### **SIMSW1**

##### **SIM Status Word 1 Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								SW1							

Field	Bits	Type	Description
<b>SW1</b>	7:0	r	<b>Status Word 1</b>
<b>RESERVED</b>	15:8	r	Reserved; these bits must be left at their reset values.

#### **SIMSW2**

##### **SIM Status Word 2 Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								SW2							

Field	Bits	Type	Description
<b>SW2</b>	7:0	r	<b>Status Word 2</b>
<b>RESERVED</b>	15:8	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**SIM Interface**

### 10.8.3.11 SIM Receive Spacing Register

**SIMRXSPC**

**SIM Receive Spacing Register**

**Reset values: T=0: 0028<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>								<b>RXSPC</b>							

Field	Bits	Type	Description
<b>RXSPC</b>	7:0	rw	This defines the spacing between a received command response character and the next transmitted character.
<b>RESERVED</b>	15:8	r	Reserved; these bits must be left at their reset values.

#### For T=0 Mode

The spacing is measured in 1/16 ETU, which means that the reset value of 0028<sub>H</sub> is 2.5 ETU.

### 10.8.3.12 SIM Transmit Spacing Register

**SIMTXSPC**

**SIM Transmit Spacing Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>								<b>TXSPC</b>							

Field	Bits	Type	Description
<b>TXSPC</b>	7:0	rw	This defines the extra spacing between successive transmitted characters for the character, T=0mode.
<b>RESERVED</b>	15:8	r	Reserved; these bits must be left at their reset values.

The spacing is given in ETU.

**CONFIDENTIAL**

**SIM Interface**

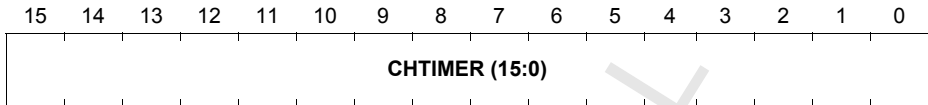
### 10.8.3.13 SIM Character Timer Registers

#### SIMCHTIMERx

##### SIMCHTIMER1

##### SIM Character Timer Register 1

**Reset value: 2580<sub>H</sub>**

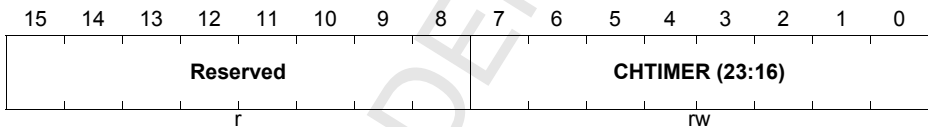


Field	Bits	Type	Description
<b>CHTIMER (15:0)</b>	15:0	rw	<b>SIM Character Timer</b>

##### SIMCHTIMER2

##### SIM Character Timer Register 2

**Reset value: 0000<sub>H</sub>**



Field	Bits	Type	Description
<b>CHTIMER (23:16)</b>	7:0	rw	<b>SIM Character Timer</b>
<b>RESERVED</b>	15:8	r	Reserved; these bits must be left at their reset values.

The spacing is given in ETU. The reset value = 9600 ETU.

**CONFIDENTIAL**

**SIM Interface**

### 10.8.3.14 SIM Block Waiting Timer Register

**SIMBWT2**

**SIM Block Waiting Timer Register 2**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								BWT (23:16)							

Field	Bits	Type	Description
<b>BWT (23:16)</b>	7:0	rw	<b>SIM Block Waiting Timer</b>
<b>RESERVED</b>	15:8	r	Reserved; these bits must be left at their reset values.

The spacing is given in ETU. The reset value = 9600 ETU.

### 10.8.4 SIM Card Interface

The SIM interface can issue three interrupts:

1. SIMOKINT for the normal character interrupts
2. SIMERRINT for the parity, overrun, T=0 complete, and work waiting timer interrupts
3. SIMININT for the SIM presence interrupt (auto power down).

### 10.8.5 Clock Control

All SIM related clock signals<sup>1)</sup> are controlled via the **SIMCTRL** register. After system reset all clocks are off and must be enabled by the MCU during initialization of the SIM interface.

The SIM UART is initialized by enabling the SIM UART clock by setting **SIMCTRL.UARTON** and enabling the SIM card interface by setting **SIMCTRL.SIMEN**. By doing this, the UART enters RX mode and is ready for Answer To Reset (ATR). Initializing the UART with **SIMEN** does not affect the SIM clock, SIM V<sub>CC</sub>, SIM RST or SIM I/O. Before entering low power mode:

1. Disable the SIM card (**SIMEN** = 0)
2. Turn off the UART clock .

The bit **SIMCTRL.UARTON** and the register **SIMBRF** control the clock used internally by the SIM UART. **UARTON** enables the clock controlling the UART and the T=0 controller and can be set to zero between commands to save power.

The value of **SIMBRF** (SIM Bit Rate Factor) directly controls the baud rate used by the UART. This factor can be used if the SIM during ATR lets the ME (mobile equipment)

<sup>1)</sup> Refer to **System Integration**: Clock domain on **Page 761**.

## CONFIDENTIAL

## SIM Interface

know that it supports enhanced speed mode – D = 8, F = 512. By changing the BRF, virtually any other combination of D and F can be used. After a successful Protocol and Parameter Selection the ME can switch to the enhanced speed and use this speed during the rest of the session.

The BRF is calculated by using the following equation:

$$\text{BRF} = \frac{F}{4D} \quad [36.2]$$

During ATR the SIM card uses the default values D = 1 and F = 372 which result in BRF = 93. This value is automatically loaded into **SIMBRF** when the peripheral is reset (refer to **RST\_CTRL\_STA (on Page 683)** register), but **not** when the SIM interface is disabled by resetting **SIMCTRL.SIMEN**.

The value of the BRF must be an integer and appropriate to the ISO 7816 and GSM specification, that is, BRF = 16 (10<sub>h</sub>) for F = 512, D = 8 and BRF 8 (08<sub>h</sub>) for F = 512, D = 16.

The clock signal feeding the SIM card can be switched on and off by the bit **SIMCTRL.SIMON**. This is used when starting an ATR sequence and between commands if the SIM supports the clock stop mode. Some Phase 2 SIMs may require the clock to stop at a high state. If this is the case, the bit **SIMCTRL.CLKHIGH** must be set prior to a clock stop.

*Note: If card deactivation is done in SW, **CLKHIGH** must be reset before stopping the clock. If card deactivation is done automatically, the setting of this bit is ignored.*

Because PMB7870 supports low power mode in which the 13 MHz reference oscillator can be switched off, the 3.25 MHz SIM clock may not be available.

For SIM cards, which do NOT support clock stop operation, the reference oscillator and the clock operating the SIM interface must stay switched on. In this case an 1.08 MHz clock generated from the 13 MHz clock can be used instead of the 3.25 MHz clock. This 'sleep' clock is selected by setting bit **SIMCTRL.CLKSEL**.

*Note: The 'sleep' clock must not be used during communication with the SIM card.*

## CONFIDENTIAL

## SIM Interface

### 10.8.6 SIM Activation and Deactivation

Prior to an ATR:

- The SIM interface must be enabled
- The normal speed must be selected.

Next, the UART ready interrupt and UART parity interrupt must be enabled by setting bit **SIMIRQEN.ENOKINT** and **SIMIRQEN.ENPAR**. UART overrun interrupts can be enabled when debugging the ATR state machine. During ATR, the HW controlled T=0 protocol must be disabled (default after reset).

The SIM interface is now ready to receive the initial character TS given the card has successfully been reset. The card is reset by doing the following steps:

1. Switch on SIM  $V_{CC}$  by setting bit **SIMCTRL.SIMVCC**. SIMVCC is inverted on the pin for control of an external PNP transistor. If the ME supports both 5 V and 3 V Smart cards, the proper voltage must be selected and controlled via one of the I/O ports.
2. Enable the SIM I/O line by setting **SIMCTRL.SIMIOL**. The I/O line is now tristated.
3. Switch on the SIM clock by setting **SIMCTRL.SIMON**. An internal reset card now answers within 40 000 clock cycles.

If the card is with active low reset, it does not answer until the RST signal goes high. This is controlled by setting bit **SIMCTRL.SIMRST**.

*Note: According to ISO/IEC 7816-3 - Amendment 2 (1997), internal reset cards are no longer produced. However, old versions may still be in use.*

By default, the interface expects the SIM to use direct convention (**SIMCTRL.INCON** = 0). If this is the case, the TS character causes an interrupt on SIMOKINT, and **UARTOK** is the only bit set in **SIMSTATUS**. If the SIM uses inverse convention instead, the TS character causes an interrupt on SIMERRINT and **SIMSTATUS.PARINT** is set.

When parity error detection signalling on received characters is disabled (**SIMCTRL.ERROFF** = 1), no error signal is generated on the I/O-line upon parity error detection. The received TS character can be read from **SIMRX** and should be used by MCU to determine the character coding convention used by SIM.

Enabling of parity error signalling on received characters (**SIMCTRL.ERROFF** = 0) and, if needed, selection of inverse convention (**SIMCTRL.INCON** = 1) must be done within two ETU after SIMOKINT or SIMERRINT interrupt.

If the UART overrun interrupt is enabled the MCU must always read the **SIMRX** register even if the data received is not needed. If not read, an overrun interrupt is generated instead of a UART ready interrupt.

SIM deactivation is done by executing the following steps:

1. If the clock is stopped at high level switch it back on (**SIMCTRL.SIMON** = 1) and set clock stop to low level (**SIMCTRL.CLKHIGH** = 0).
2. Set RST to low level (**SIMCTRL.SIMRST** = 0).

CONFIDENTIAL

SIM Interface

3. Stop the SIM clock (**SIMCTRL.SIMON** = 0).
4. Set SIM I/O to low level (**SIMCTRL.SIMIOL** = 0).
5. Remove SIM V<sub>CC</sub> (**SIMCTRL.SIMVCC** = 0).

The interface can now be disabled.

*Note: Forcing the I/O line to LOW while the SIM is transmitting may damage the SIM. It should always be done as above.*

### 10.8.7 Initialization Sequence Overview

The following gives the programming sequence for initializing the SIM interface. This should be read in conjunction with the **SIMCTRL** register description.

1. Make sure that bit **SIMCTRL.SIMRST** is not set.
2. Configure CCIOSW corresponding to hardware (only required for 5V Smart Card) by programming bit **SIMCTRL.SMCIOSWACT**, and programming the port logic of SMCIOSW.
3. Enable SIM interface (**SIMCTRL.SIMEN** = 1).
4. Enable SIM UART (**SIMCTRL.UARTON** = 1).
5. Configure SIM interrupts for ATR procedure.
6. Start driving the external interface by programming the following:
  - a) Voltage on (**SIMCTRL.SIMVCC** = 1)
  - b) I/O line open drain (**SIMCTRL.SIMIOL** = 1)
  - c) Enable SIM clock (**SIMCTRL.SIMCLK** = 1).
7. Wait 108 ETUs (~12,4ms).
8. Set CC\_RST line high (**SIMCTRL.SIMRST** = 1).
9. Wait for ATR - copied by interrupt handler into some internal buffer; change convention if first character generates a parity error (**SIMCTRL.INCON** = 1)
10. Evaluate ATR and make a PTS procedure if high-speed mode is supported and desired: PTS is done sending in character mode (refer to **Section 10.8.9 SIM Character Mode (on Page 781)**) a PTS command and receiving the confirmation/rejection from the card. The usage of the T=0 controller is not possible for the PTS command. The bitrate factor may be changed in **SIMBRF**.

### 10.8.8 Automatic Power Down

**Note: The Automatic Power Down of the SIM card performed by the HW is not functional in ES1 and ES2. A software work around has to be done. When the SIM card is removed, SW has to perform the power down.**

The SIM interface has a feature that can be used in conjunction with mechanical detection of SIM presence. A mechanical switch can be connected to the SMC\_IN pin. The SMC\_IN pin can be used either by the Smart card interface (with some limitations) or the SIM interface.

**CONFIDENTIAL**

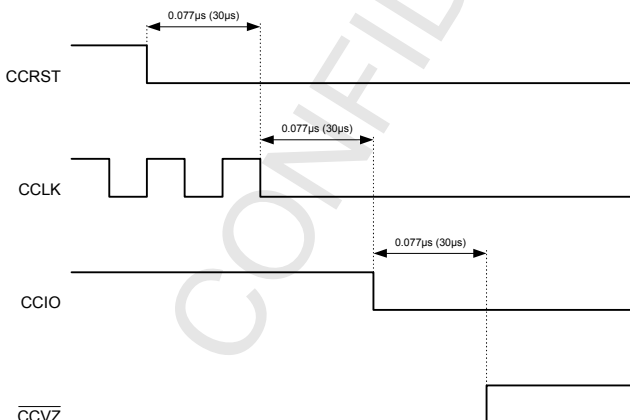
**SIM Interface**

SMC\_IN can be used in several ways:

1. The status of pin SMC\_IN can always be read on bit **SIMSTATUS.SIMDET**. Reading this bit does not affect any of the other bits that may have been set by UART and T=0 controller interrupts. This functions when connected to the Smart Card or the SIM card.
2. The SIM\_IN\_INT interrupt is enabled by setting bit **SIMIRQEN.EN**. The interrupt is issued each time the level of SMC\_IN changes. The SIM card can now be disabled in three ways:
  - a) The MCU can manually disable the SIM as described above. This method functions in conjunction with either the SIM card of the Smart card.
  - b) The MCU can force an automatic power down by setting bit **SIMCTRL.SIMPDWN**. This also cause the **SIMCTRL.SIMEN** bit to be reset after an internal delay. This method only functions when one SIM card is being supported by the SIM card interface.
  - c) If bit **SIMCTRL.APDWN** is set, any transition from HIGH to LOW on the pin SMC\_IN causes an automatic power down. This method also only functions when one SIM card is being supported by the SIM card interface.

The automatic power down works even if PMB7870 is running on 32 kHz - although it will run considerably slower. **Figure 10-36** shows the power down sequence and timing (the values in brackets are for sleep mode, 32 kHz operation).

**Figure 10-36 Timing of Automatic Power Down Sequence**



Once triggered the SIM signals are held to LOW level until the MCU resets the SIM interface by writing 0 to **SIMCTRL.SIMEN**. Before setting **SIMEN**, the MCU must reset the **SIMCTRL** bits **SIMVCC**, **SIMON**, **SIMIOL** and **SIMRST**. This ensures LOW level on the SIM signals when auto power down mode is re-initialized.



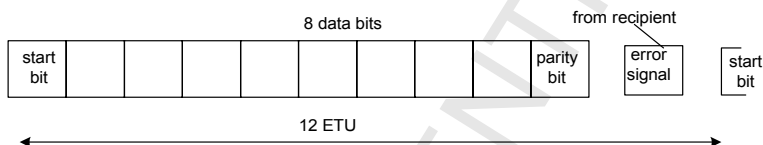
*Note: The SIMININT interrupt is generated, even if the SIM has been powered down.  
This can be used to detect if the SIM has been removed or inserted.*

### 10.8.9 SIM Character Mode

The mode used during ATR is called SIM Character Mode (SCM). In this mode, the SIM interface only ensures an error free character transmission/reception. Any overlaying protocol - such as the T=0 protocol - must be implemented in SW especially spacing between characters and timeouts. The SIM interface provides all the signals and data needed to implement such protocols.

**Figure 10-37** shows the basic structure of a SIM character. This format is also used in T=0 mode.

**Figure 10-37 SIM Character Data Structure**



#### 10.8.9.1 Receiving Characters

The basic functionality of the SIM interface is described in [Section 10.8.1 Functional Overview \(on Page 761\)](#). The following is more detailed description of the receive mode.

The SIM interface scans the I/O line for a start bit. When a start bit is detected, the following bits are shifted into an internal register and transferred to the **SIMRX** register when the last bit is received. The parity bit is checked after 9.5 ETU and an interrupt is generated. If no parity error is detected, the **SIMSTATUS.UARTOK** bit is set and a SIMOKINT is generated. In case of a parity error, the SIM interface generates an error condition lasting **one** eTu (from 10.5 ETU to 11.5 ETU) on the I/O line. The **SIMSTATUS.PARINT** bit is then set instead of the UARTOK bit and a SIMERRINT is generated. If a new byte is received before the previous one is read from the **SIMRX** register, the **SIMSTATUS.OVRUN** bit is set overriding all other **SIMSTATUS** interrupt bits - again a SIMERRINT is generated. If the overrun interrupt is enabled the **SIMRX** register must always be read on an **UARTOK** interrupt. If an overrun error occurs, the SIM session cannot continue.

Error signalling on received characters can operate without enabling the **SIMSTATUS.PARINT** interrupt. By keeping bit **SIMCTRL.ERROFF** at 0, and by resetting **SIMIRQEN.ENPAR**, the UART detects and signals parity errors without interrupting the MCU. To disable error detection **ERROFF** must be set.

### 10.8.9.2 Sending Characters

Setting the SIM interface in TX mode is straight forward. A transmission is initiated by writing the byte to be transmitted to the **SIMTX** register. The data is shifted out with the LSB first at a bit rate of 8.73kbaud as a default. Inverse convention can be chosen by setting the **SIMCTRL.INCON** bit, thus inverting the polarity and bit order of the data byte.

The interface generates the parity bit on the I/O line at the end of the data byte, according to the **INCON** bit. After 11 ETUs the status of the I/O line is checked for the receiver status. If no parity error is reported by the SIM (the I/O line is HIGH), the **SIMSTATUS.UARTOK** bit is set indicating that the interface is ready to accept a new byte to be transferred. If the SIM reports a parity error (the I/O line is LOW), the **SIMSTATUS.PARINT** bit is set and the data byte is retransmitted until a successful transmission has been performed. A parity error interrupt is generated before each retransmission. When a byte has been transmitted, the UART automatically enters RX mode. A new byte can be written to **SIMTX** right after the previous byte has been transmitted successfully, without considering the character spacing. The SIM UART ensures a character spacing of 12 ETUs when the MCU transmits characters continuously. This is however not the case when changing from RX- to TX-mode. After receiving a character, the MCU must wait for at least 2.5 ETU before writing to **SIMTX**. If the spacing of 12 ETUs between two successive characters is too tight, extra spacing can be programmed in the **SIMTXSPC** register in steps of one ETU.

Similar to error detection and signalling in RX-mode, character repetition can be enabled (**SIMCTRL.RPTOFF** = 0) while the **PARINT** interrupt is switched off. Character repetition is disabled by setting **RPTOFF**.

If a new byte is written to **SIMTX** before the previous byte has been transmitted, the **SIMSTATUS.OVRRUN** bit is set.

### 10.8.10 SIM T=0 Protocol Mode

When setting **SIMBRF** to 16 (corresponds to D = 8 and F = 512), the MCU interrupt load from the SIM interface increases approximately 6 times. To lower the interrupt-load, the T=0 protocol has been implemented in HW. This unit is called T=0 controller.

#### 10.8.10.1 Data Fetch Instructions

Data fetch instructions all receive data from the SIM card. The example below describes a READ BINARY instruction where the SIM sends 100 bytes. In receive mode, the header must be written by the MCU.

The max number of bytes that can be transferred in a data fetch command (refer to ISO 7816-3, section 8.2.1 and ISO 7816-3 - Amendment 2 - 2ed Edition, section 9.1) are 256 (P3 = 00).

### **10.8.10.2 Data Write Instructions**

Data write instructions all transfer data to the SIM card. The example below describes an UPDATE BINARY instruction where the SIM receives 100 bytes.

The max number of bytes that can be transferred in a data fetch command (refer to ISO 7816-3, section 8.2.1 and ISO 7816-3 - Amendment 2 - 2ed Edition, section 9.1) are 255 (P3 = FF).

### **10.8.10.3 Work Waiting Timer (WWT)**

As indicated above parity errors are handled by the T=0 controller during execution of an instruction. Additionally, the controller contains a timer unit to detect if the distance between received characters exceeds the Work Waiting Time (WWT) defined during ATR. WWT is always checked between the leading edges of any character sent by the card, and the previous character.

The timer is programmed through 24 bits in registers **SIMCHTIMERx** (SIM Character Timer) with a base resolution of 1 ETU.

The timer function is enabled by setting bit **SIMIRQEN.ENCHTIMER** meaning that the character timer interrupt is enabled when the SIM card interface block is operated in SIM T=0 protocol mode.

**CONFIDENTIAL**

**SIM Interface**

In T=0 mode, the character timer behaves as a WWT. The T=0 controller starts/restarts the character timer with the value defined in **SIMCHTIMERx**:

- When the timer is enabled with **ENCHTIMER**.
- Whenever a procedure byte (NULL, ACK or SW1) is received (also at the reception of retransmitted procedure bytes). If a character or a procedure byte has not been received before the expiration of the timer, a timeout interrupt is generated.
- Whenever a character is received (also at the reception of retransmitted characters) by the T=0 controller. If a character or a procedure byte has not been received before the expiration of the timer, a timeout interrupt is generated.
- Whenever a header or body character is transmitted (also at retransmission of characters) by the T=0 controller. If a procedure byte has not been received before the expiration of the timer, a timeout interrupt is generated.
- After the WWT value has been written to the **SIMCHTIMERx** registers.

The character timer is stopped without causing a timeout interrupt at the reception of the status byte SW2 (resulting in a command end interrupt) and at deactivation of the SIM – both with respect to MCU controlled deactivation but also when **SIMCTRL.SIMEN** is programmed with the value 0 (disabled following a automatic power down for example).

The character timer is also stopped when a character timeout interrupt is generated.

A character timeout results in a SIMERRINT interrupt causing the **SIMSTATUS.CHTIMEOUT** bit to be set. A character timeout does not change the behavior of the ongoing T=0 instruction handling and it is the responsibility of the MCU to determine whether further instruction operation can be accepted.

*Note: A character timeout interrupt is also generated during transmission of subsequent characters, if the time between two characters exceeds the defined timeout limit. (This is not strictly required by ISO/IEC 7816-3, but can be used to increase the robustness of the system)*

#### **10.8.10.4 Character Spacing**

When receiving characters from the SIM, proper spacing (minimum allowed) is handled by the card.

The spacing after receiving a character and before starting to transmit, is ensured to be 2.5 ETU. However, this can be changed through the **SIMRXSPC** register, where this spacing is calculated in 1/16 ETU, so the default value of 40 equals  $40/16 = 2.5$ .

When transmitting characters to the SIM, the T=0 controller ensures 12 ETUs between two consecutive transmitted characters. This spacing can be increased by programming the additional spacing in ETU to the **SIMTXSPC** register.

### **10.8.11 Connection of 5 V, 3 V, and 1.8 V SIM Cards**

Because the pad supply voltage for E-GOLDradio is limited to 3.0 V, an external level shifter is required to connect 5V Smart cards.

The E-GOLDradio can also support 1.8 V SIMs, but in this case, both the SIM interface, and the Smart Card interface has a voltage of 1.8 V. If in such a case, the Smart card needs a higher voltage (3.0 V for example), then a level shifter is also required.

### **10.8.12 SIM Card Pads**

During powering up the mobile it may happen, that the pads (Vdd2.x) are supplied earlier than the core (Vdd1.x). To avoid spikes on the SIM-interface due to a undefined reset status, the SIM interface pads have a special reset behavior.

If the chip reset (RESET\_IN = 0) is active, the following fixed logic states are driven at the SIM Card pads:

- CC\_IO = 0
- CC\_RST = 0
- CC\_CLK = 0
- CC\_VZ = 1.

In this state boundary scan data, which is latched into the boundary scan cells, does not appear at the SIM card pins.

CONFIDENTIAL

**CONFIDENTIAL**

**AFC**

## 10.9 AFC

History	
Current version: Rev. 1.06, 2005-11-04	
Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.01	
<b>Page 787</b>	Updated <b>System Integration</b> : WS00006682
Changes for Rev. 1.04	
all	Updated <b>Section 10.9 AFC</b> (13 MHz changed to 26 MHz) WS00008814
Changes for Rev. 1.06	

### System Integration:

- Supply domain: VDD\_MAIN
- Chip internal interfaces:
  - Clock domain: Refer to **Section 10.4.1.3 Sub-System Clocks and Enables (on Page 661)** and see **Figure 10-11 Clock and Enable Generation for MCU Sub-System (on Page 668)**.
  - Bus domain: X-Bus
- Interrupt sources:
- Monitor Pins: Refer to **Section 11.8.10 Internal Signal Monitoring (on Page 1209)**

### 10.9.1 Introduction

The Automatic Frequency Control (AFC) Unit generates a pulse number modulated (PNM) signal which can be low pass filtered externally to make a programmable DC voltage for controlling the external VCO.

The PNM signal repeats itself with a period depending on the number of bits used in register **AFCVAL (on Page 789)**. For standard applications the upper 11-bits are used and the PNM signal repeats every 2048 26-MHz clock cycles. (Approximately 78  $\mu$ s at a frequency of 12.69 kHz).

For each 26 MHz clockinput to the block, the output is driven to 1 or 0. The ratio of the number of 1's to 0's output in a period is proportional to the value of the **AFCVAL** register. The 1's are distributed equally in a PNM period. No pulse deviates more from a perfect distribution by more than 1 input clock period. For example, if the value 7 is programmed, the leading edge of the seven 38.5 ns wide pulses are separated by 293, 293, 292, 293, 292, 293, and 292 clock periods, which is near the ideal value of 292,57.

**CONFIDENTIAL****AFC**

The equal spacing in a period ensures that most of the energy in the signal appears at harmonics of 13.69 kHz, (in this case  $7 \times 13.69$  kHz), and very little at 13.69 kHz. Because of this the requirements for the low pass filter are considerably reduced compared to a pulse width modulated output or a pulse number modulated signal with unequal distribution. In this example which uses 11-bits, the lowest frequency component in the spectrum of any E-GOLDradio PNM sequence is 13.69 kHz and has an effective amplitude of less than 1 LSB.

The DC voltage which can be generated externally with a passive filter is proportional to the supply voltage, proportional to the number of 1's output in a period and inversely proportional to the number of clocks in a period.



**CONFIDENTIAL**

**AFC**

### 10.9.2 Use of AFCVAL Register for Standard Applications

For standard applications 11-bit resolution is used. In this case the bits **AFCVAL(3:0)** must be zero and the value of **AFCVAL(14:4)** is the number of 1s output in a period of 2048 clock cycles (26 MHz).

The data transmission can be enabled by setting **AFCVAL.ENAFC** = 1.

#### **AFCVAL**

#### **AFC Reference Value Register**

**Reset value: 0000<sub>H</sub>**

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>EN AFC</b>	<b>AFC(14:4)</b>												<b>AFC(3:0)</b>			

Field	Bits	Type	Description
<b>AFC(3:0)</b>	3:0	rw	<b>AFC Standard Applications Enable</b> 0 AFC set for standard 11-bit application. Any other value AFC set for 14-bit application. <i>Note: If <b>AFCVAL(14:0)</b> = 7FFF<sub>H</sub>, output does not remain high; there is one low impulse in every period.</i>
<b>AFC(14:4)</b>	14:4	rw	<b>AFC Standard Application Value</b> This number of high pulses generated in 2048 clock cycles.
<b>ENAFC</b>	15	rw	<b>AFC Enable</b> 0 Data transmission disabled (AFC pin is in tristate) 1 Data transmission enabled

The AFC-PNM transmits the current PNM value until it is disabled (via **ENAFC** = 0) or a new PNM reference value is loaded.

If the value 7FFF<sub>H</sub> is loaded into the reference value register, its output is set to 1.

The AFC-PNM output is tri-stated when disabled.

### 10.9.3 Use of AFCVAL Register for Special Applications

Standard applications have 11-bit resolution and use the upper 11 bits of **AFCVAL**. A lower (n bits) resolution value can be generated by loading n upper **AFCVAL** bits with the value and all lower bits with 0. This generates high pulses which are distributed over 2<sup>n</sup> clock cycles. The shorter cycle means that filter with a higher cut off frequency can be used and that the DC value after the filter can change faster.

CONFIDENTIAL

AFC

For a resolution of up to 11 bit, the differential non-linearity of the DAC is less than 1 LSB because at each voltage step only one additional pulse is activated and each clock period is approximately equal.

**Note: If a resolution of more than 11-bit is programmed the digital PNM circuit will correctly deliver a longer sequence, but the analog differential non linearity of the output voltage after the external low pass filter may be more than 1 LSB. In other words increasing the programmed output voltage by 1 LSB may result in a decrease of the output voltage instead of an increase.**

This undesirable effect may be caused by second order effects such as small amounts of non random jitter in the width of the 26 MHz clock if the jitter is synchronous to a sub-harmonic of the input clock frequency. Small synchronous variations in the supply voltages can cause a similar effect. This can be illustrated by a simplified example. If at 11-bit resolution and 2048 cycle length, every second input clock is 19 ps longer than its predecessor, then a 50% output voltage which requires a digital sequence where the output is high for every second input clock could be generated with 1024 short pulses or 1024 long pulses. The difference in the two possible "identical" output values is 0.5 LSB. If the same input clock is used for 12 or 13 bit resolution the accuracy would be only 1 or 2 LSBs which for most applications is not acceptable.

*Note: The AFC pin can be used to produce voltage up to 2.5 V defined by the voltage on pin VDD2.3. No additional supply voltage pin is provided. Because the DC load current is limited to a low value by the resistance in the RC filter, the maximum and minimum output high and low voltages are identical to the VSS and VDD 2.3 supply voltages.*

*Note: The worst case output for integral non-linearity is the 50% value. One output cycle consists of 1024 pulses each separated by a single space. Because the rise and fall times of the AFC output are not symmetrical, the area of the individual output pulses differs slightly from the ideal value. A difference in rise and fall times of only 1.5 ns (output loaded with 10 pF board capacitance) would lead to an integral non-linearity error of approximately  $(1,5 \text{ ns} / 77 \text{ ns}) \times 50\% = 1\% = 10 \text{ LSBs}$ . This is a typical value of the error. The worst case error including production tolerances and temperature is approximately 3%. This is only a small variation in the gain of the frequency control loop.*

CONFIDENTIAL

AFC

## 10.9.4 AFC Identification Register

AFCID

AFC Identification Register

Reset value: A701<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Module_ID								Revision_Number							

Field	Bits	Type	Description
Revision_Number	0:7	r	<b>AFC Revision Number</b> These hard-wired bits are used for module revision numbering.
Module_ID	8:15	r	<b>AFC Identification Number</b> These hard-wired bits are used for module identification numbering.

CONFIDENTIAL

**CONFIDENTIAL**

**RF Power Ramping**

## **10.10 RF Power Ramping**

<b>History</b>	
Current version: Rev. 1.06, 2005-11-04	
Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.03	
<b>Page 794</b>	Updated <b>Section 10.10.1 Introduction</b> WS00007560
Changes for Rev. 1.06	

### **System Integration:**

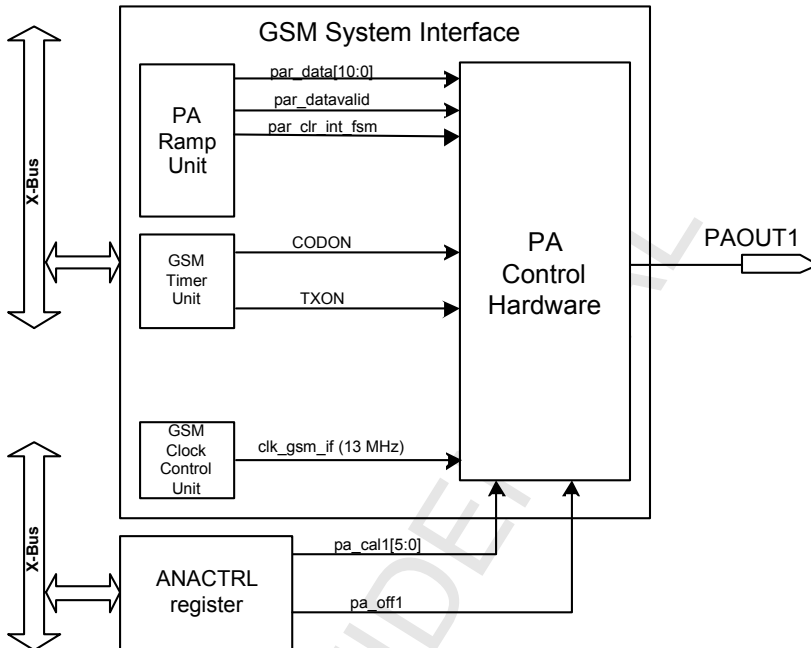
- Supply domain: VDD\_MAIN
- Chip internal interfaces:
  - Clock domain: clk\_gsm\_if
  - Bus domain: X-Bus
- Interrupt sources:
- Monitor Pins: Refer to **Section 11.8.10 Internal Signal Monitoring (on Page 1209)**

### **10.10.1 Introduction**

The RF power amplifier (PA) needs analog control voltages to set the output power of the mobile. The PA Control Hardware provides the possibility, to set the output power of the mobile according to the GSM requirements to fulfill the power time template specification and the adjacent channel power specification for switching transients.

For the PA adjustments an analog control voltage is supplied by the PMB7870 at the output pin PAOUT1. As shown in **Figure 10-38** the PA Control Hardware is connected to GSM System Interface units and to the Analog Control Registers (refer to **Section 10.6 Analog Control Registers (on Page 750)**).

Figure 10-38 Power Amplifier Control Hardware System Overview

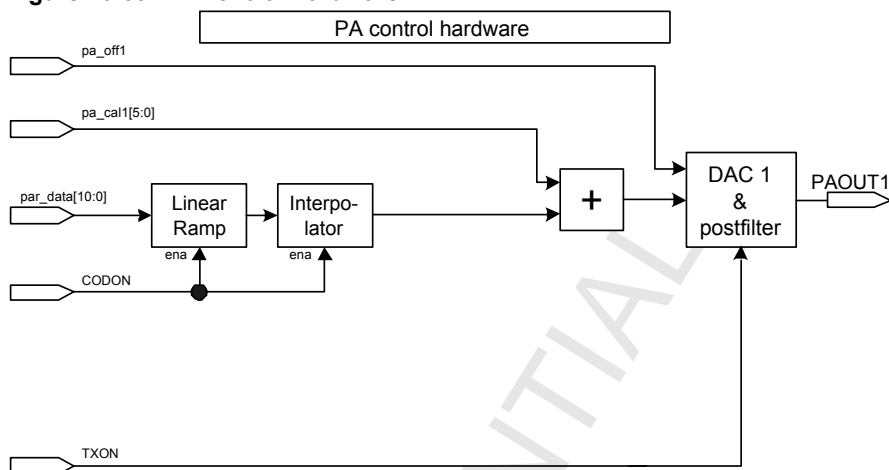


*Note:* The PA ramping data is stored by the controller within the RF RAM, refer to [Section 10.12.2 RF RAM \(on Page 836\)](#). The power ramping words are transferred from the PA Ramp Unit to the PA Control Hardware. *For information about the GSM CGU refer to [Section 10.4.1 Clock Generation Unit](#).*

CONFIDENTIAL

RF Power Ramping

Figure 10-39 PA Control Hardware



An additional feature of the PA Control Hardware is the possibility to correct the PA power with a linear slope during the active part of the burst. The PA power ramping is basically divided into three periods: the power ramping at the start and end of a burst and the linear slope during the active part of the burst. Intermediate ramping between two consecutive burst is possible as well when the correct ramping shape is programmed by the controller.

Internal details about the HW are shown in [Figure 10-39](#). The units shown in the block schematic have the following functions:

- **Linear Ramp** Performs the linear ramping during the active part of the burst.
- **Interpolator** Interpolates the data rate to 6.5 MSamples/s.
- **Adders** Adds offset value for offset compensation. Refer to [Section 10.10.1.3.1 Digital Correction of Analog DC Offset \(on Page 798\)](#).
- **DAC & filter** Digital-to-analog conversion and analog anti-aliasing filtering.

### 10.10.1.1 Power Ramping Shapes

Several types of power time templates are defined for the GSM system. The implementation of the rising and the falling part of a ramping curve is described in [Section 10.10.1.1.1 Up and Down Ramping \(on Page 796\)](#).

One complete set of ramping data in the RF RAM of the RF Control Unit comprises 16 values for the ramping shape and one 11-bit value (word 17) for the step size of the linear power interpolation for the ramping during the active part of the burst. The structure of

CONFIDENTIAL

RF Power Ramping

the data is described in the [Table 10-34 RF RAM Partitioning of the RF Control Unit \(Type 1\) \(on Page 837\)](#).

#### 10.10.1.1.1 Up and Down Ramping

In the RF RAM 16 specific RAM locations PAR[1:16] have to be loaded with 11-bit values by the controller. This represent the shape of the desired ramping curve as digital ramp samples and the PAINC value as word 17 (refer to [Section 10.12.2 RF RAM \(on Page 836\)](#)). The data is transferred from the RF Ramping Unit to the PA Ramping Hardware with the rate of  $6.5 \text{ MHz}/12 = 541.67 \text{ kSamples/s}$ .

With the trigger mechanism described in [Section 10.12 RF Control \(on Page 835\)](#) the ramping process is started and the 16 ramp samples and the PAINC value are transferred to the PA Ramping Hardware.

*Note: The data path delay from the RF Ramping Unit to the interpolator is approximately 2 symbols (7.385  $\mu\text{s}$ ).*

#### 10.10.1.1.2 Linear Power Ramping during Active Part of Burst

The output power of a PA without amplitude feedback control usual is changing during the active part of the burst. To compensate this deviation the RF Control Unit has to be able to correct this effect. The shape of the active part of an GMSK burst can be corrected by the linear slope.

The 18th value of a ramping data set is the value PAINC (11 bits, 0...2047) and defines the amount of increment cycles ( $f_{\text{clk\_gsm\_if}}/24 = 541.67 \text{ kHz}$ ) during the active part of the burst after which the output value of the multiplexer in the PA Ramping Hardware is incremented by 1 as shown in [Figure 10-40](#).

*Note: With PAINC = 0 the linear ramping functionality is deactivated. One  $f_{\text{clk\_gsm\_if}}/24$  cycle after the value PAINC is transferred to the PA Ramping Hardware the linear ramping is started if the value PAINC is any value except 0.*

**Figure 10-40 Linear Power Ramping during Active Part of the Burst**



**CONFIDENTIAL**

**RF Power Ramping**

### 10.10.1.1.3 Consecutive Bursts

When the 16 ramp samples and the value PAINC have been transferred and the linear ramping is deactivated the latest voltage output on PAOUT remains constant until a new set of 16 samples representing the falling part of the ramping shape is to be processed.

If the linear ramping is activated the output value is changing in steps of one LSB. It is SW responsibility to start the following up/down ramping sequence with a correct start value to avoid a step in the power ramp.

*Note: For the down ramping the value PAINC should be set to 0 in order to avoid unwanted changes of the PA output power when the PA power control is regarded as switched off.*

The start value of the down ramping section can be calculated as follows:

$$y = \text{floor}\left(\frac{t_{\text{burstlin}} \cdot 541.6 \text{ kHz} - 1}{\text{PAINC}}\right) + x \quad [36.3]$$

where:

x = end value of the up ramping sequence

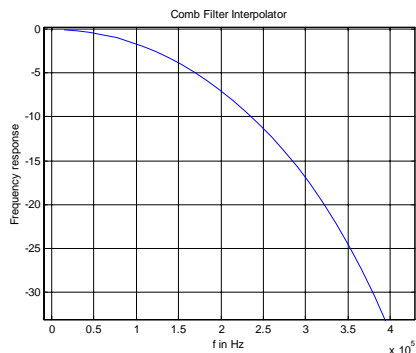
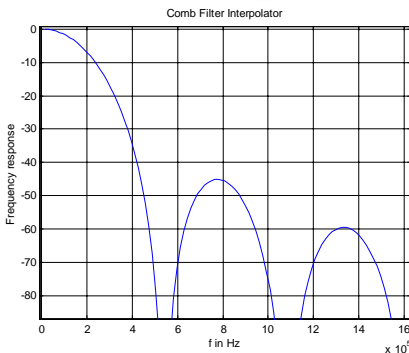
y = start value of the down ramping sequence

$t_{\text{burstlin}}$  = length of the active part of the burst in seconds.

### 10.10.1.2 Interpolation

The interpolator increases the sampling frequency from 541.6 kHz to 6.5 MHz thus, providing a 11-bit sample every 154 ns to the D-to-A converter.

### 10.10.1.3 Programming Sequence for Power Ramping



The MCU procedure to get the PA ramping path working after a boot up of the PMB7870:

CONFIDENTIAL

RF Power Ramping

The entire control of the PA ramping is done by the GSM system interface.

1. Refer to [Section 10.12 RF Control \(on Page 835\)](#) for programming the power ramping data into the RF control RAM.
2. Enable the signal CODON by programming the GSM timer unit in [Section 11.7 GPT 1 and 2 \(on Page 1145\)](#) to switch on the digital power ramping HW.
3. Enable TXON by programming the GSM timer unit in [Section 11.7 GPT 1 and 2](#) to switch on the analog power ramping HW.
4. Start the transfer of the power ramping telegrams.
5. To disable the PA ramping path in a correct way, first switch off TXON, then switch off the CODON.

*Note: The CODON signal enables the digital blocks; the TXON signals enables the analog blocks of the TX modulator as well.*

#### 10.10.1.3.1 Digital Correction of Analog DC Offset

The analog DC offset of the analog part of the PA Control Hardware is measured by the measurement interface. For a digital correction of the analog DC offsets digital adders are used. The advantage of separate adders compared to a software-only solution is that standard tables for each mobile can be used and only two calibration values, that is, the value of [ANA\\_CTRL2 \(on Page 753\)](#), [PA\\_CAL1](#) has to be measured and set.

The measurement procedure is described in [Section 10.5.7 Modulator Unit Offset Measurement TXOFI and TXOFQ \(on Page 729\)](#). The correction values [PA\\_CAL1](#) ( $= 0..63_D$ ) are added to the output of the PA Multiplexer Unit.

*Note: The input value to the adder must not exceed the range of  $0..1984_D$ .*

*This is  $2047_D - 63_D = 1984_D$  for a 6-bit wide correction word and leaves the necessary headroom for the DC correction.*

#### 10.10.1.4 D-to-A Conversion and Post Filtering

The digital-to-analog converters are R-string type. After interpolation the signal spectrum will be repeated around multiples of the sampling frequency (6.5 MHz). These frequency components have to be suppressed by a subsequent lowpass reconstruction filter. This post filters are active 1rd order filters with a cutoff frequency of 300 kHz. The resulting analog voltage is shown on the output pin PAOUT1.

The nominal linear output voltage range for PAOUT1 is  $U_{drop-}$  to  $VDD - U_{drop+}$ . ([Figure 10-41](#),  $I_{PAOUT} > 0$ ). More details for different currents are listed in [Table 12-51 Specification of PAOUTOF1 \(on Page 1377\)](#).  $I_{PAOUT} > 0$  denotes a current flow out of the pin PAOUT1 and  $I_{PAOUT} < 0$  vice versa. Therefore, the actual linear output range depends on  $I_{PAOUT}$ ,  $VDD$ , and the dropout voltages as shown in [Figure 10-41](#).

*Note: For transient-free power ramps it is a precondition to stay within the linear operating range of the PAOUT1 pin during the HF power amplifier unit is sensitive for the PAOUT1 output voltage.*

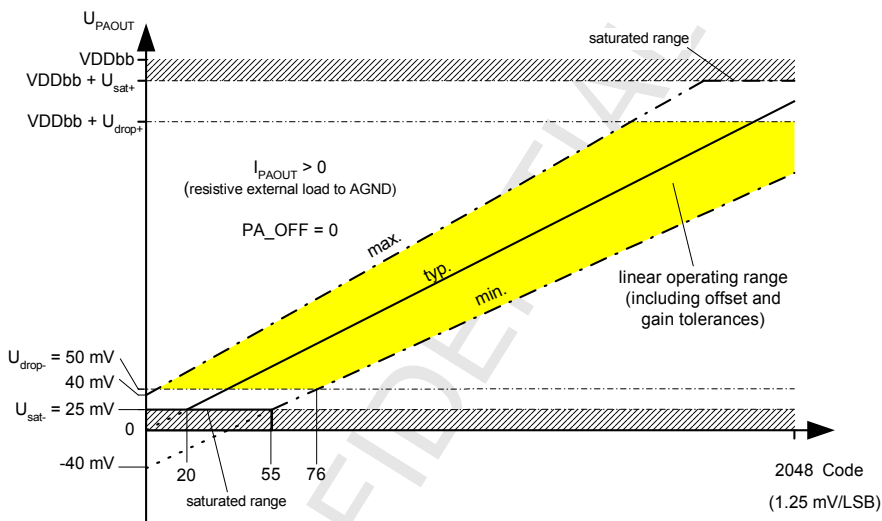
CONFIDENTIAL

RF Power Ramping

For **ANA\_CTRL2.PA\_OFF1** = 0 there's zero (no) voltage bias. For **PA\_OFF1** = 1 the output buffer is biased by nominal -50 mV for offset adjustment. As the PMB7870 can only output positive voltages, only digital values above up to 40 ensure an output voltage in the linear operating range in the case of **PA\_OFF1** = 0.

The signal TXON is used to switch on the analog voltage supply of the DAC1 and the postfilter. The PAOUT1 output is forced to 0 V during TXON = 0.

**Figure 10-41 Operating Range of PAOUT1 Output after Offset Calibration**



## 10.10.2 Application Notes

### 10.10.2.1 How to Program a Power Ramp Signal, after a POWER Up

1. Program the power ramping sequences (up and down ramping), the PAINC values into the RF RAM (refer to [Section 10.12.2 RF RAM \(on Page 836\)](#)).
2. Program the offset calibration information in [ANA\\_CTRL2 \(on Page 753\)](#).PA\_CAL1 and [ANA\\_CTRL1 \(on Page 751\)](#).PA\_OFF1.
3. Program the GSM timer signals CODON, TXON (refer to [Section 10.11.2 GSM Clock Control Unit \(on Page 828\)](#)).
4. Program the trigger signals for the PA ramping sequence telegrams (refer to [Section 10.12.2 RF RAM](#)).
5. After down ramping, switch TXON signal off.
6. Switch CODON signal off, the PAOUT1 pin is now in high impedance.

*Note: TXON and CODON are enabling the TX modulator hardware as well (refer to [Section 8.8 GMSK Modulator \(on Page 479\)](#)).*

*Note: For offset self calibration of PAOUT1 refer to [Section 10.6.3 Analog Control Register 2 \(on Page 753\)](#).*

*Note: The CODON signal must be enabled before the TXON signal is switched active to get a valid analog output signal with the rising edge of TXON.*

*Note: The ramping sequence must not be started within 2 symbols (7.385  $\mu$ s) after CODON is set active.*

*Note: The last word of the ramp down sequence is D-to-A converted until the falling edge of TXON. The last word of the ramp down sequence is again D-to-A converted starting with the rising edge of TXON until the first word of the ramp up sequence is received.*

*Note: To reduce power consumption, the clock for the PA Ramping Hardware is only enabled when needed (during a valid burst by the CODON signal). For correct operation of the PA Ramping Hardware the signal CODON must be switched on in advance to the beginning of the power ramp period and must remain active until the end of the ramp down period.*

*Note: The user must ensure that RF chips, such as the PA, are not powered on until a valid PAOUT1 signal is available.*

## 10.10.3 System Register

### Module Identification

#### TID

GSM System Interface Identification Register (refer to [TID \(on Page 811\)](#)).

CONFIDENTIAL

CONFIDENTIAL

**CONFIDENTIAL**

**GSM Timer Unit**

## 10.11 GSM Timer Unit

History	
Current version: Rev. 1.06, 2005-11-04	
Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 0.02	
<b>Page 811</b>	Updated register <b>TID</b>
Changes for Rev. 1.01	
<b>Page 803</b>	Updated <b>System Integration</b> : WS00006682
Changes for Rev. 1.02	
<b>Page 821</b>	Updated register <b>TEAPT</b> WS00007353
Changes for Rev. 1.04	
<b>Page 815</b> <b>Page 816</b>	Note added in descriptions for registers <b>TCOR</b> and <b>TOVF</b> WS00008441
Changes for Rev. 1.06	
all	TRIG[25:24] removed WS00009067

### System Integration:

- Supply domain: VDD\_MAIN
- Chip internal interfaces:
  - Clock domain: Refer to [Section 10.4.1.3 Sub-System Clocks and Enables \(on Page 661\)](#) and see [Figure 10-11 Clock and Enable Generation for MCU Sub-System \(on Page 668\)](#).
  - Bus domain: X-Bus
- Interrupt sources:
- Monitor Pins: Refer to [Section 11.8.10 Internal Signal Monitoring \(on Page 1209\)](#)

#### 10.11.1 Overview

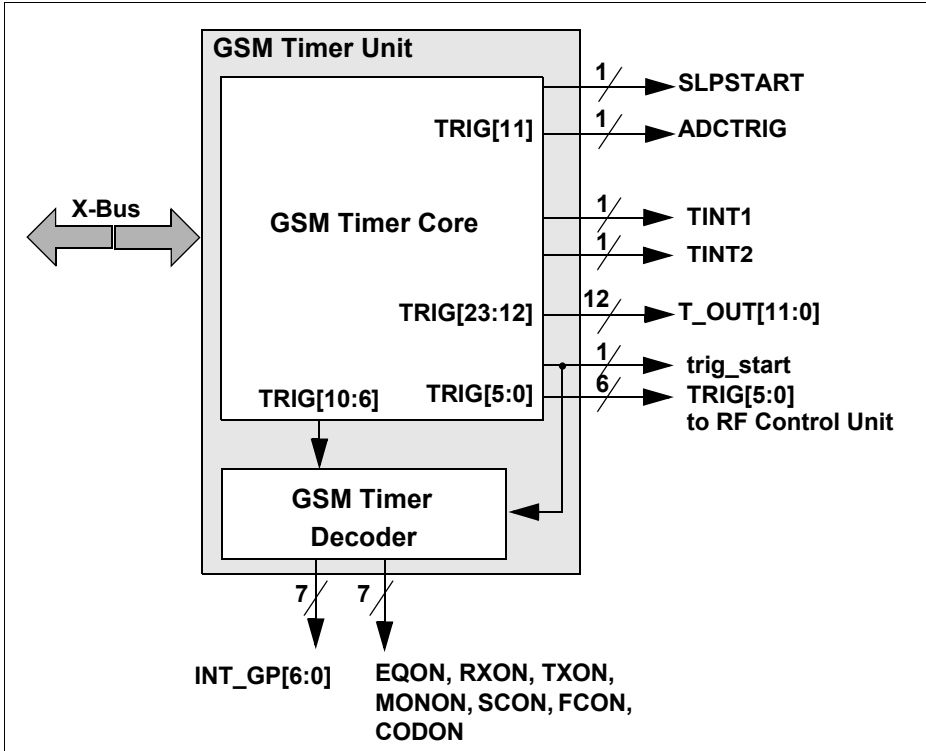
The GSM Timer Unit provides all timing signals which are periodically repeated in TDMA frames and thus off-loads the controller from scheduling events. These timing signals are chip internal signals (for example, trigger signals for equalizer, RF control, etc.) as well as chip external signals for control of the HF ICs. By means of a user programmable event table each timing signal can be programmed very flexible. Thus, the timing signals of the GSM Timer Core are not application specific. They are all generated in an identical

**CONFIDENTIAL**

**GSM Timer Unit**

way and the meaning is only determined by the GSM Timer Decoder and the controller software or DSP firmware (see [Figure 10-42](#)).

**Figure 10-42 External Interfaces of the GSM Timer Unit**





**Table 10-30 GSM Timer Unit Signals**

Signal Name	Derived from	Function
TRIG[5:0]	-	Selects the telegram which has to be transferred by the RF Control Unit
INT_GP[4:0]	TRIG[10:6] <sup>1)</sup>	5 general purpose interrupt signals connected to the controller
INT_GP[6:5]		2 general purpose interrupt signals connected to DSP: INT_GP[5] = FRAME_INT INT_GP[6] = SYS_MCU_INT
EQON MONON SCON FCON CODON		Signals for the DSP to control data acquisition
TXON RXON		Signals to enable/disable the hardware blocks of the RX path (analog and digital) and TX path (analog)
ADCTRIG	TRIG[11]	Trigger signal for measurement interface
T_OUT[11:0]	TRIG[23:12]	12 signals are provided to switch on/off RF ICs periodically each TDMA frame.
trig_start	-	Signal triggers transmission of telegrams or power ramps by the RF Control Unit and triggers the GSM Timer Decoder.
SLPSTART	-	Trigger signal for SCCU block
TINT1	-	Controller interrupts generated at a programmed timer value of the RTDMA Counter
TINT2	-	

<sup>1)</sup> The description of the decoding scheme of TRIG[10:6] is located in [Section 10.11.3 GSM Timer Decoder \(on Page 833\)](#).

The GSM Timer Unit reduces the controller load for scheduling events and, therefore, it supports full rate channels, half rate channels, and multislot configurations.

An easy reconfiguration is possible for:

- Timing advance adjustment in transmit time slots
- Fade-out of the TX and RX control signals every 26<sup>th</sup> TDMA frame
- Insertion or fade-out of monitoring time slots
- Battery measurements

- Hand-over in general.

### 10.11.1.1 Basic Operation

The reference counter of the GSM Timer Unit (RTDMA Counter) located in the TDMA Counter Unit of the TDMA Compare Unit. The RTDMA Counter is a programmable (usually) modulo 10000<sub>D</sub> 15-bit wide counter operated by the clock clk\_counter with a clock period of 0.461 μs and allows to measure the length of one TDMA frame (4.615 ms) with a resolution of one eighth of a bit (refer to [Section 10.11.1.5 TDMA Counter Unit](#)).

The Timer RAM of the GSM Timer Unit contains the information about the point in time (within a TDMA frame, called the Timing Compare Value) when the 24 output signals TRIG[23:0] may get their new values or change/toggle their values. The record of one Timing Compare Value and the corresponding values for the new output signals TRIG[23:0] is called a Timing Event.

The Timer RAM configured like a FIFO. The TDMA Timer Comparator compares the last Timing Compare Value of the FIFO with the value of the TDMA Counter Unit. When both values are identical, the corresponding output values for TRIG[23:0] are read out of the Timer RAM and forwarded to the output signals TRIG[23:0]. After a match the Timer RAM address is incremented by the address generator of the RAM Control Unit and the next Timing Compare Value is compared with the value of the TDMA Counter Unit. A simplified example with only 4 timer output signals TRIG[3:0] illustrates the basic operation of the GSM Timer Unit in [Figure 10-43](#). (In this example the Group Enable Unit with their corresponding entries in the Timer RAM have not been taken into account.)

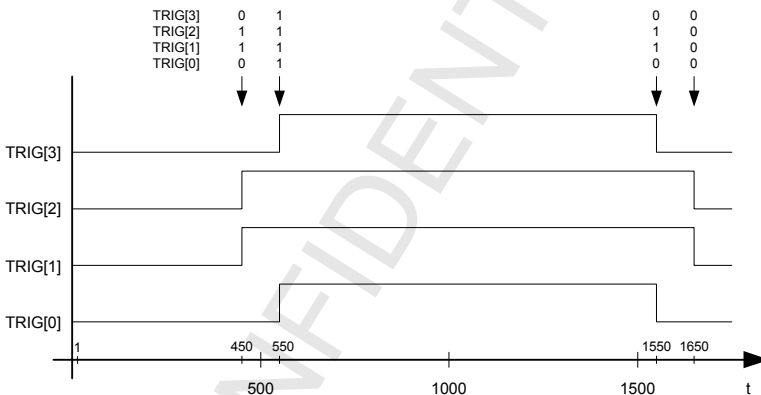
Figure 10-43 Basic Operation of the GSM Timer Unit

RAM content

timing compare value [14:0]

															TRIG[0]	TRIG[1]	TRIG[2]	TRIG[3]	
TE0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
TE1	0	0	0	0	0	0	1	1	1	0	0	0	0	1	0	0	1	1	0
TE2	0	0	0	0	0	1	0	0	0	1	0	0	1	1	0	1	1	1	1
TE3	0	0	0	0	1	1	0	0	0	0	0	1	1	1	0	0	1	1	0
TE4	0	0	0	0	1	1	0	0	1	1	1	0	0	1	0	0	0	0	0

Timing diagram



The Timer RAM in [Figure 10-43](#) stores 5 timing events TE0...TE4 each consisting of a 15-bit wide Timing Compare Value and the corresponding values for the 4 timer outputs. For example, the address generator points at the beginning to Timer RAM entry 0 with the Timing Compare Value of 0001. When the TDMA Counter Unit reaches this value the timer outputs TRIG[3:0] are changed according to the new output values stored in the Timer RAM. Afterwards the address generator is incremented and points to the second entry containing the Timing Compare Value 0450<sub>D</sub>. When the TDMA Counter Unit reaches this value the timer outputs change again, etc. The timing event values have to be stored in incremental and consecutive order in the GSM Timer RAM.

When the address generator has reached the top address (defined by the pointer [TEAPT](#)) it jumps to the bottom address (defined by the pointer [TEAPB](#)). While an address region is executed a new set of timing events can be written into the Timer RAM region not in use to prepare the next frame. Thus, a new set of timing events can be activated by changing the bottom and top addresses of the address generator only.

### 10.11.1.2 Timer RAM

The GSM Timer Unit comprises a dual port Timer RAM of 512 \* 16 bits. The RAM address range is defined in [Chapter 12 Register Lists and Mapping \(on Page 1265\)](#). The Timer RAM contains timing event entries as already mentioned according to [Table 10-31 Structure of Timing Events \(on Page 809\)](#) and timing offset entries for timing advance operations according to [Table 10-32 Structure of Timing Advance \(on Page 810\)](#).

Three different types of timing events are distinguished:

- Timing events in which all trigger signals TRIG[23:0] are identical to the bits TB[23:0] (refer to [Table 10-31](#)). This operation is explained in [Section 10.11.1.1 Basic Operation](#).
- Two other types of timing events allow marking only the trigger signals that have to be set or reset in comparison to the previous trigger values. This feature is important for using the Group Assign Bits GAB[4:0] (refer to [Table 10-31](#)).

The Timer RAM can hold a maximum number of 170 Timing Events and/or Timing Offset Values.

The allowed sequence of Timing Events and Timing Offset Values in the Timer RAM is: When the CTDMA Counter is reset the Timing Events or Offset Values are read out of the Timer RAM beginning at address **TEAPB** (refer to [Section 10.11.1.14 RAM Control Unit](#)). Therefore, the Timing Events and Timing Offset Values have to be stored in ascending order between the pointers **TEAPB** and **TEAPT** (the lowest timing compare value has to be stored at **TEAPB**, the highest timing compare value at entry [**TEAPT**-1]).

*Note: A timing event at the Timing Compare Value 0 is performed at the Timing Compare Value 1 and therefore it should **NOT** be programmed (this is a limitation of the chosen implementation).*

Within one 2.167 MHz clock cycle of the clock clk\_counter (which is the frequency of the CTDMA Counter) up to 6 accesses of the dual port Timer RAM can be performed. To enable the timing events at the correct time and, therefore, to exclude delayed enables, the following conditions have to be considered:

- Not more than **one enabled** timing event is allowed at one timing compare value.
- Between two enabled timing events which timing compare values differ by n up to 3(n-1) disabled timing events and/or Timing Offset Values are allowed. The Disabled Timing Events may even have the same Timing Compare Value.

**CONFIDENTIAL**

**GSM Timer Unit**

- Between the Timing Compare Value 0 and the first enabled timing event with the Timing Compare Value  $n$  ( $n > 0$ ) up to  $3(n-1)$  Disabled Timing Events and/or timing offset values are allowed.

**Table 10-31 Structure of Timing Events**

Word	Bit	Function
1	7:0	<b>Timer Bits TB[23:16]</b> The new signals $TRIG_{new}[23:16]$ are defined by TB[23:16] at the timing compare value TCV[14:0] specified in word 2.
	8	<b>Reserved</b> , must be set to 0.
	13:9	<b>Group Assign Bits GAB[4:0]</b> The bits GAB[4:0] are processed in the Group Enable Unit (refer to <a href="#">Section 10.11.1.17 Group Enable Unit (on Page 822)</a> ).
	15:14	<b>Entry Select Bits</b> 00 This word and the following two RAM words represent a timing event entry with $TRIG_{new}[23:0] = TB[23:0]$ . 01 This and the following two RAM words represent a timing event entry with $TRIG_{new}[11:0] = TB[11:0]$ and $TRIG_{new}[23:12] = TRIG_{old}[23:12]$ OR TB[23:12]. 10 This and the following two RAM words represent a timing event entry with $TRIG_{new}[11:0] = TB[11:0]$ and $TRIG_{new}[23:12] = TRIG_{old}[23:12]$ AND NOT(TB[23:12]). 11 Refer to <a href="#">Table 10-32 Structure of Timing Advance (on Page 810)</a> , do not use for timing events
2	14:0	<b>Timing compare value TCV[14:0]</b> Timing compare value which is compared with the CTDMA Counter value in the TDMA Timer Comparator.
	15	<b>RESERVED</b> , must be set to 0.
3	15:0	<b>Timer Bits TB[15:0]</b> The new signals $TRIG_{new}[15:0]$ are defined by TB[15:0] at the timing compare value TCV[14:0] specified in word 2.

**Table 10-32 Structure of Timing Advance**

Word	Bit	Function
1	8:0	<b>Reserved</b> , must be set to 0.
	13:9	<b>Group Assign Bits GAB[4:0]</b> The bits GAB[4:0] are processed in the Group Enable Unit (refer to <a href="#">Section 10.11.1.17 Group Enable Unit (on Page 822)</a> ).
	15:14	<b>Entry Select Bits</b> 00 Refer to <a href="#">Table 10-31 Structure of Timing Events (on Page 809)</a> , do not use for timing offset values 01 Refer to <a href="#">Table 10-31</a> , do not use for timing offset values 10 Refer to <a href="#">Table 10-31</a> , do not use for timing offset values 11 This word and the following two RAM words represent a timing offset entry.
	14:0	<b>Timing offset value TOV[14:0]</b> The timing offset value can be positive or negative. TOV[14] represents the sign. Negative values have to be expressed in two's complement notation.
2	15	<b>Reserved</b> , must be set to 0.
3	15:0	<b>Reserved</b> , must be set to 0.

**CONFIDENTIAL**

**GSM Timer Unit**

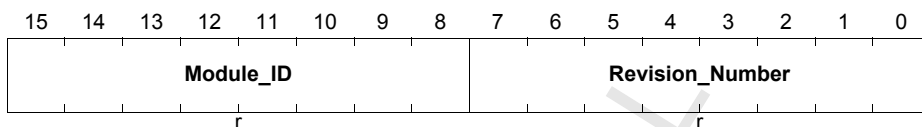
### 10.11.1.3 GSM Timer Identification Register

**TID**

**GSM Timer Identification Register**

**Reset values:**

**For E-GOLDradio Versions before V3.0: 2102<sub>H</sub>**



Field	Bits	Type	Description
Revision_Number	7:0	r	<b>GSM Timer Revision Number</b> These hard-wired bits are used for module revision numbering.
Module_ID	15:8	r	<b>GSM Timer Identification Number</b> These hard-wired bits are used for module identification numbering.

### 10.11.1.4 TDMA Compare Unit

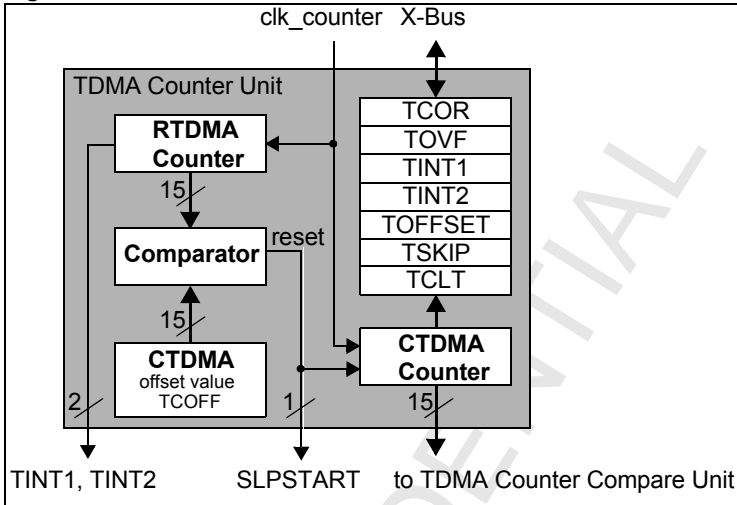
The TDMA Compare Unit consists of:

- **TDMA Counter Unit** ([Section 10.11.1.5 TDMA Counter Unit \(on Page 812\)](#)) that contains a programmable modulo 10 000<sub>D</sub> 15-bit counter operated by a 2.167 MHz clock and measures the length of one TDMA frame (4.615 ms).
- **Timing Advance Unit** ([Section 10.11.1.12 Timing Advance Unit \(on Page 819\)](#)) that stores the latest Timing Offset Value provided by the Timer RAM and adds this value to the Timing Compare Value.
- **TDMA Timer Comparator** ([Section 10.11.1.4 TDMA Compare Unit \(on Page 811\)](#)) which compares the value of the TDMA Counter Unit with the Timing Compare Value increased by the latest Timing Offset Value by the Timing Advance Unit.

### 10.11.1.5 TDMA Counter Unit

The block diagram of the TDMA Counter Unit is shown in [Figure 10-44](#).

**Figure 10-44 TDMA Counter Unit**

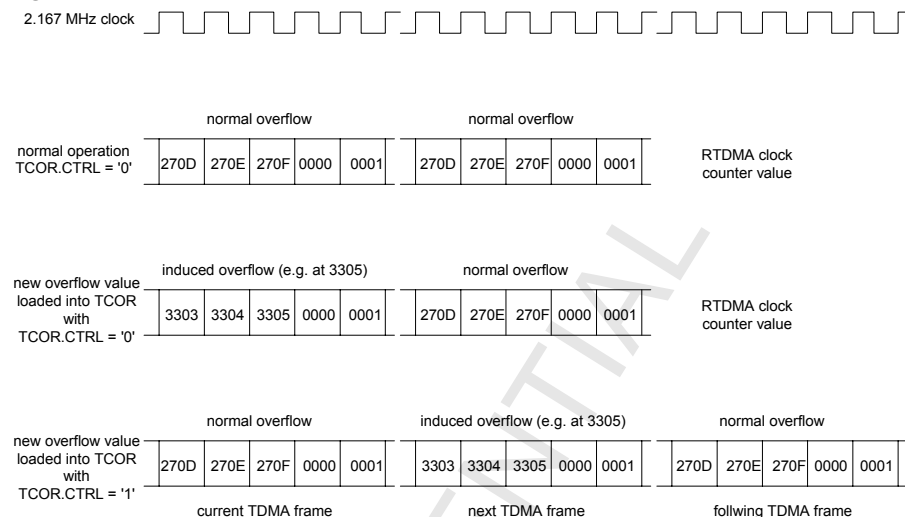


The reference counter of the GSM Timer Unit is the RTDMA Counter of the TDMA Counter Unit. The RTDMA Counter is a modulo  $10000_{10}$  15 bit wide counter operated by a clock with  $0.461 \mu\text{s}$  clock period and allows to measure the length of one TDMA frame ( $4.615 \text{ ms}$ ) with a resolution of one eights of a bit.

To synchronize the local TDMA frame timing of the mobile station with the global timing defined by the base station of the SCELL, the regular overflow value  $9999_{10}$  ( $270F_H$ ) of the RTDMA Counter can be overwritten temporarily by loading a 15-bit wide correction value into the RTDMA counter correction register **TCOR**. See [Figure 10-45](#).



**Figure 10-45 Overflow Behavior of RTDMA Counter**



The correction value to be stored in the register **TCOR** can be found as follows:  
A Frequency Correction Burst (FCB) search is started by the GSM Timer Unit (FCON goes high). When the DSP has found a FCB the number of transferred bits from the start of the FCB search up to the end of the detected FCB (end of time slot 0) is transferred to the shared memory. By means of this value the controller is able to evaluate the **TCOR** value.

$9999_D$  ( $270F_H$ ) is the overflow value of the RTDMA Counter being valid after a reset. If this value is modified (for example, for multiple monitoring during a TDMA frame), a new 15-bit wide overflow value ( $TOVF[14:0]$ ) can be loaded into the counter overflow register **TOVF**.

*Note: For modulo  $5000_D$  counting, the overflow value to be loaded into **TOVF** is  $5000_D - 1 = 4999_D$ , that is,  $TOVF[14:0] = 1387_H$ .*

Related to the counter value of the RTDMA Counter two interrupt signals TINT1 and TINT2 are provided to the interrupt control unit of the controller. The TINT1 and TINT2 are toggle interrupts, that is, every time the level of TINT1 or TINT2 toggles an interrupt is generated. The RTDMA Counter value where the interrupt TINT1 and TINT2 is generated is determined by the timer interrupt registers **TINT1** and **TINT2**.

Besides the RTDMA Counter a second 15-bit wide counter, the CTDMA (current TDMA) counter exists. Using the CTDMA counter offset register **TOFFSET** an offset TCOFF between the RTDMA Counter and CTDMA Counter can be programmed. Therefore, the RTDMA Counter can always be synchronized to the SCELL and during monitoring

CONFIDENTIAL

GSM Timer Unit

frames. By changing the **TOFFSET** register value, the CTDMA Counter can easily be synchronized with the neighboring cells (NCELLs):

$$\text{'CTDMA Counter value'} = \text{'RTDMA Counter value'} - \text{'TCOFF value'}. \quad [36.4]$$

By using the Frame Skip Register **TFSKIP**, the reset of the CTDMA Counter can be canceled once. Afterwards, the bit **TFSKIP.SKIP** is automatically reset internally. The bit **SKIP** is especially suited for the handling of the idle frame in TCH/FR channels. For this mode the bit **SKIP** should be set to 1 in the CTDMA frame 24 to make the CTDMA Counter perform only 25 frame cycles, each of them with one RX, TX, and neighbor cell monitoring mode, which eases the programming of the Timer RAM.

The CTDMA Counter value can be monitored via the Counter Latch Register **TCLT**.

**CONFIDENTIAL**

**GSM Timer Unit**

### 10.11.1.6 RTDMA Counter Correction Register

**TCOR**

**RTDMA Counter Correction Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT RL	TCOR														

Field	Bits	Type	Description
<b>TCOR</b>	14:0	rw	<b>Counter Correction Value</b> Temporarily new overflow value
<b>CTRL</b>	15	rw	<b>Control</b> 0    The next RTDMA Counter overflow occurs at the counter value equal to the correction value <b>TCOR</b> . After that, the regular overflow value is valid again.  <i>Note: If the correction value <b>TCOR</b> is lower than the current RTDMA Counter value the counter will overflow first at the regular overflow value (defined in register <b>TOVF</b>) and after that it overflows at the correction value.</i>  <i>Note: <b>TCOR</b> cannot be written to while a correction is active. When activating a correction with <b>TCOR</b>:</i> <i>If <b>CTRL</b> = 0, the <b>TCOR</b> register cannot be written until the end of the current RTDMA frame</i> <i>If <b>CTRL</b> = 1 the <b>TCOR</b> register cannot be written until the end of the next RTDMA frame.</i>  1    The RTDMA Counter first overflows at the regular overflow value (defined in register <b>TOVF</b> ) and then at the counter value equal to the correction value <b>TCOR</b> . After that, the regular overflow value is valid again.

**CONFIDENTIAL**

**GSM Timer Unit**

### 10.11.1.7 RTDMA Counter Overflow Register

**TOVF**

**RTDMA Counter Overflow Register**

**Reset value: 270F<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RES ERV ED</b>	<b>TOVF</b>														

Field	Bits	Type	Description
<b>TOVF</b>	14:0	rw	<b>Counter Overflow Value</b> User defined overflow value of the RTDMA Counter. <i>Note: <b>TOVF</b> cannot be written to while a <b>TCOR</b> correction is active. Access to <b>TOVF</b> register must wait until the completion the <b>TCOR</b> correction.</i>
<b>RESERVED</b>	15	r	Reserved; these bits must be left at their reset values.

### 10.11.1.8 RTDMA Timer Interrupt Registers

**TINT1**

**RTDMA Timer Interrupt Register 1**

**Reset value: 7FFF<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RES ERV ED</b>	<b>TINT1</b>														

Field	Bits	Type	Description
<b>TINT1</b>	14:0	rw	<b>Timer Interrupt 1 Value</b> User defined RTDMA Counter value at which an interrupt TINT1 is generated.
<b>RESERVED</b>	15	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**GSM Timer Unit**

## TINT2

### RTDMA Timer Interrupt Register 2

**Reset value: 7FFF<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>	<b>TINT2</b>														

Field	Bits	Type	Description
<b>TINT2</b>	14:0	rw	<b>Timer Interrupt 2 Value</b> User defined RTDMA Counter value at which an interrupt TINT2 is generated.
<b>RESERVED</b>	15	r	Reserved; these bits must be left at their reset values.

## 10.11.1.9 CTDMA Counter Offset Register

### TOFFSET

#### CTDMA Counter Offset Register

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>CTRL</b>	<b>TCOFF</b>														

Field	Bits	Type	Description
<b>TCOFF</b>	14:0	rw	<b>CTDMA Counter Offset Value</b> RTDMA Counter value at which CTDMA Counter is reset.
<b>CTRL</b>	15	r	<b>Control</b> 0 New CTDMA offset value is directly validated  <i>Note: If the CTDMA Counter offset value is lower than the current RTDMA Counter value the CTDMA Counter will not be reset before the next RTDMA frame.</i>  1 New CTDMA offset value validated the after next RTDMA counter overflow <sup>1)</sup>

<sup>1)</sup> If this register is read, the actual active value is read.

**CONFIDENTIAL**

**GSM Timer Unit**

### 10.11.1.10CTDMA Frame Skip Register

**TFSKIP**

**CTDMA Frame Skip Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>														<b>SKIP C</b>	<b>SKIP N</b>

Field	Bits	Type	Description
<b>SKIPN</b>	0	rw	<b>Skip Next CTDMA Counter Reset</b> 0 No action 1 The next reset of the CTDMA Counter is skipped. This bit is validated after the next CTDMA Counter overflow. The bit is reset by the unit with the skipped reset pulse.
<b>SKIPC</b>	1	r	<b>Skip Current CTDMA Counter Reset</b> 0 No action 1 The current reset of the CTDMA Counter is skipped. This bit is directly validated. The bit is reset by the unit with the skipped reset pulse.
<b>RESERVED</b>	15:2	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**GSM Timer Unit**

### 10.11.1.11 Counter Latch Register

**TCLT**

**Counter Latch Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RES ERV ED</b>	<b>TCLT</b>														

Field	Bits	Type	Description
<b>TCLT</b>	14:0	r	<b>Timer counter latch value</b> At every READ access the current CTDMA Counter can be monitored via the <b>TCLT</b> .
<b>RESERVED</b>	15	r	Reserved; these bits must be left at their reset values.

### 10.11.1.12 Timing Advance Unit

When a new timing event or Timing Offset Value is read out of the Timer RAM, the Timing Advance Unit is activated:

- If a new Timing Offset Value is read out of the Timer RAM that **is not disabled** via the corresponding Group Enable Bit, this value is written into the Timing Advance Register of the Timing Advance Unit (this register can not be directly accessed via the X-bus).
- If a new Timing Offset Value is read out of the Timer RAM that **is disabled** via the corresponding Group Enable Bit, this value is skipped.
- If a new timing event is read out of the Timer RAM that **is not disabled** via the corresponding Group Enable Bit, the corresponding timing compare value is written into the Timing Compare Register of the Timing Advance Unit (this register can not be directly accessed via the X-Bus). The Timing Compare Register value and the Timing Advance Register value are added and forwarded to the TDMA Timing Compare Unit.
- If a new timing event is read out of the Timer RAM that **is disabled** via the corresponding Group Enable Bit, this event is skipped and the next entry is requested from the RAM Control Unit.

*Note: When a timing event or Timing Offset Value was read out of the Timer RAM (disabled or not), the next entry in the Timer RAM is requested from the RAM Control Unit.*

*Note: For power consumption reasons it is recommended to have always one valid entry in the Timer RAM. Otherwise, the Timer RAM is read out with the maximum possible frequency without any effect on the output signals TRIG[23:0] of the GSM Timer Unit.*

### 10.11.1.13TDMA Timer Comparator

The TDMA Timer Comparator compares the value provided by the Timing Advance Unit with the CTDMA Counter value. If both values are equal, the next entry of the Timer RAM is requested from the RAM Control Unit.

### 10.11.1.14RAM Control Unit

The TDMA Compare Unit requests a new entry from the RAM Control Unit when a:

- Timing offset value has been read by the timing advance unit (no matter if the Timing Offset Value was enabled or disabled),
- Timing event has been read that was disabled by the Group Enable Unit or
- Timing event that was enabled has reached the current CTDMA Counter value.

The current address pointer in the register **TCEAP** of the RAM Control Unit always points to the next address that has to be read out of the RAM. When a new entry is requested by the TDMA Compare Unit (starting from the current address pointer), the next three RAM addresses are read out of the Timer RAM and the current address pointer is correspondingly incremented. When the upper address specified by the pointer in the registers **TEAPx** has been reached, the current address pointer is loaded with the lower address specified by the pointer in the register **TEAPB** and starts to increment again.

*Note: When a new value is written into the register **TEAPT** or **TEAPB**, this value is valid after the next reset of the CTDMA Counter.*

*Note: To prevent dead lock situations, the current address pointer is always reset to the **TEAPB** value when the CTDMA Counter is reset.*

*When the GSM Timer Unit is initialized via the bit **TPARA.TINI** in the Output Control Unit, the current RAM pointer is set to the **TEAPB** value.*

*Note: The RAM Control Unit works with only one timing event in the RAM.*



**CONFIDENTIAL**

**GSM Timer Unit**

### 10.11.1.15 Current Timer Event Address Pointer Register

**TCEAP**

**Current Timer Event Address Pointer Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>								<b>TCEAP</b>							

Field	Bits	Type	Description
<b>TCEAP</b>	8:0	r	<b>Current Timer Event Address Pointer</b> This is the address of the RAM where the next timing event will be read.  <i>Note: This register should only be used for debug purposes since the register value can be wrong when, during the read access of the controller, the register content is changed by the RAM Control Unit!</i>
<b>RESERVED</b>	15:9	r	Reserved; these bits must be left at their reset values.

### 10.11.1.16 Timer Event Address Pointer Registers

**TEAPx**

**TEAPT**

**Timer Event Top Address Pointer Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>								<b>TEAPT</b>							

Field	Bits	Type	Description
<b>TEAPT</b>	8:0	rw	<b>Timer Event Top Address Pointer Value</b> This is the address where the current address pointer is reset to <b>TEAPB</b> . Allowed values are 3 <sub>D</sub> , 6 <sub>D</sub> , ... , 510 <sub>D</sub> .  <i>Note: The entry at <b>TEAPT</b> is not executed. A new entry is active after the next reset of the CTDMA Counter.<sup>1)</sup></i>
<b>RESERVED</b>	15:9	r	Reserved; these bits must be left at their reset values.

<sup>1)</sup> If this register is read, the actual active value is read.

## TEAPB

### Timer Event Bottom Address Pointer Register

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>								<b>TEAPB</b>							

Field	Bits	Type	Description
<b>TEAPB</b>	8:0	rw	<b>Timer Event Bottom Address Pointer Value</b> This is the low address of the Timer RAM entry to be executed. A new entry is active after the next reset of the CTDMA Counter <sup>1)</sup> . Allowed values are 0 <sub>D</sub> , 3 <sub>D</sub> , 6 <sub>D</sub> , ..., 507 <sub>D</sub> .
<b>RESERVED</b>	15:9	r	Reserved; these bits must be left at their reset values.

<sup>1)</sup> If this register is read the actual active value is read.

### 10.11.1.17 Group Enable Unit

Each timing event or Timing Offset Value can be assigned to one of 32 groups by setting the corresponding Group Assign Bits GAB[4:0] of the Timer RAM entry (see word 1 in [Table 10-31 Structure of Timing Events \(on Page 809\)](#) and in [Table 10-32 Structure of Timing Advance \(on Page 810\)](#)). A group of events can be enabled or disabled by setting the corresponding Group Enable Bits GEB[31:0] in the Timer Group Enable Registers **TGERx**. 0 means that the event group is deactivated. 1 means that the event group is activated.

For example, all events necessary to be processed in TX bursts can be assigned to one group and disabled if needed. In the same way the events to be processed in RX bursts, monitoring bursts or to perform measurements can be grouped together and enabled or disabled whenever desired.

The timing event entries which mark the trigger signals that have to be set or reset, in addition to the previous timing events, are only necessary if, for example, a battery measurement is usually performed during a TX burst but also, sometimes, during a disabled TX burst.

**CONFIDENTIAL**

**GSM Timer Unit**

### 10.11.1.18 Timer Group Enable Registers

**TGERx**

**TGERT**

**Timer Group Enable Top Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>GEBT</b>															

**TGERB**

**Timer Group Enable Bottom Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>GEBB</b>															

Field	Bits	Type	Description
<b>GEBT</b> <b>GEBB</b>	15:0	rw	<b>Group Enable Bits</b> For each group enable bit: 0 The corresponding timing events belonging to this group are masked out. That means that they do not affect the TRIG[23:0] output signals. 1 The corresponding timing events belonging to this group will be executed. <i>Note: A new value is active after the next reset of the CTDMA Counter.<sup>1)</sup></i>

<sup>1)</sup> If this register is read, the actual active value is read.

### 10.11.1.19 Output Control Unit

The new values of the signals TRIG[23:0] are stored in the Select Unit of the Output Control Unit when both the:

- Output Control Unit is triggered by the RAM Control Unit
- Corresponding event is not masked out by the Group Enable Unit.

Valid entries are forwarded to the output signals TRIG[23:0] of the GSM Timer Unit if they are not faded out by the Fade Out Unit (**TPARA.TINI** = 1 and **TPARA.FDIS** = 1).

Using the Fade Out Unit the signal lines TRIG[23:0] of the GSM Timer Unit can be forced to the predefined values in **TFADEx.FTRIG**:

CONFIDENTIAL

GSM Timer Unit

- If the Fade Out Unit of the Output Control Unit is enabled (**TPARA.FDIS** = 0) and the RTDMA Counter and the CTDMA Counter is running (**TPARA.TINI** = 1), then the output signals of the Fade Out Unit (instead of the output signals of the Select Unit) are forwarded to the GSM Timer Unit output signals TRIG[23:0].  
This means that TRIG[23:0] = **TFADEx.FTRIG**[23:0].  
The signal SLPSTART and the interrupts TINT1 and TINT2 are still generated.
- If the GSM Timer Unit is in the initialization state (**TPARA.TINI** = 0), the RTDMA Counter value is 1, the CTDMA Counter value is 0 and the current address pointer is equal to **TEAPB**, then all state machines are reset but the registers accessible via the controller keep their values.  
If **TPARA.TINI** = 0, the output signals of the Fade Out Unit determined by the Timer Fade Out register **TFADEx** are forwarded to the output signals TRIG[23:0]. The signal SLPSTART is 0.

The registers **TGERx** and **TEAPx** keep their old values even if new values are written. The new values are not lost. They are stored in temporary registers. If the GSM is started (**TPARA.TINI** = 1), these registers are loaded with the new values if the RTDMA counter reaches the value of **TOFFSET** (for example, the CTDMA is reset).

If **TOFFSET** = 0, these registers get new values just after **TPARA.TINI** is set to 1.

If **TOFFSET** != 0, the old values are kept for as many timer cycles as the value of **TOFFSET**.

**CONFIDENTIAL**

**GSM Timer Unit**

After initial programming of the GSM Timer Unit, the reset values of the registers **TGERx** and **TEAPx** are zero. No timing event is executed until the RTDMA counter reaches the value of **TOFFSET** because, since both **TGERT** and **TGERB** are equal to 0, all timing events are disabled.

But, if the GSM Timer Unit is stopped and reprogrammed with new timing events and is restarted, the timing events are executed until the RTDMA counter reaches the value of **TOFFSET** (**TGERx** and **TEAPx** still have their old values).

To have the timer unit restart using immediately new timing event values, use the following procedure:

```

...           // previous GSM Timer Unit action
1. TPARA      = 0    // GSM Timer Unit is stopped
2. TGERx      = 0    // disable all groups
3. TEAPT      = 0
4. TEAPB      = 0
5. TOFFSET    = 0    // CTDMA counter reset just after restart
6. TPARA      = 1    // the registers TGERx are loaded with zero
7. TPARA      = 0

...// reprogramming of Timer RAM
8. TGERx      = x    // new values for the registers are set
9. TOFFSET    = x
10. TEAPT     = x
11. TEAPB     = x
12. TPARA     = 3
  
```

// restart GSM Timer Unit with new action

*Note: Power ramp sequences cannot be started while **TPARA.TINI** = 0.*

*Note: When **TPARA.FDIS** = 0 and registers **TFADEx** is written, the signal **trig\_start** is generated.*

*Note: When **TPARA.FDIS** = 0, the **TFADEx** registers can be used to send telegrams or generate interrupts during the initialization phase.*

### 10.11.1.20Timer Parameter Register

**TPARA**

**Timer Parameter Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														FDIS	TINI

**CONFIDENTIAL**

**GSM Timer Unit**

Field	Bits	Type	Description
<b>TINI</b>	0	rw	<b>Timer Init</b> 0 GSM Timer is in the initialization state 1 GSM Timer runs
<b>FDIS</b>	1	rw	<b>Fade Out Unit Disable</b> 0 Fade Out Unit enabled 1 Fade Out Unit disabled
<b>RESERVED</b>	15:2	r	Reserved; these bits must be left at their reset values.

CONFIDENTIAL

**CONFIDENTIAL**

**GSM Timer Unit**

### 10.11.1.21 Timer Fade Out Registers

**TFADEx**

**TFADE1**

**Timer Fade Out Register**

**Reset value: 0700<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>FTRIG(15:0)</b>															

Field	Bits	Type	Description
<b>FTRIG(15:0)</b>	15:0	rw	<b>Fade Out Trigger Signal</b> If ( <b>TPARA.FDIS</b> = 0 OR <b>TPARA.TINI</b> = 0), then TRIG[15:0] := <b>FTRIG</b> [15:0].

**TFADE2**

**Timer Fade Out Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>								<b>FTRIG(23:16)</b>							

Field	Bits	Type	Description
<b>FTRIG(23:16)</b>	7:0	rw	<b>Fade Out Trigger Signal</b> If ( <b>TPARA.FDIS</b> = 0 OR <b>TPARA.TINI</b> = 0), then TRIG[23:16] := <b>FTRIG</b> [23:16].
<b>RESERVED</b>	15:8	r	Reserved; these bits must be left at their reset values.

### 10.11.1.22 SLPSTART Signal

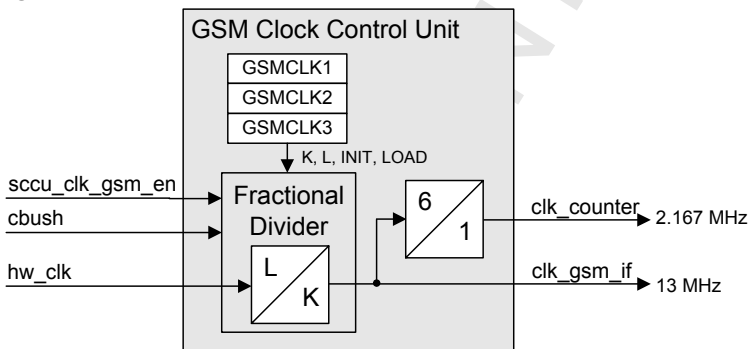
The SLPSTART signal determines the exact point in time when the GSM System Interface starts or ends a sleep phase of the SCCU block. When the CTDMA Counter is reset, SLPSTART is set and when the CTDMA Counter reaches the value 1024, SLPSTART is reset.

### 10.11.2 GSM Clock Control Unit

The clock  $hw\_clk$ <sup>1)</sup> is used to generate the  $clk\_gsm\_if$  with a frequency of 13 MHz for the GSM Timer Unit, the RF Control Unit, and the PA Control Hardware. This clock  $clk\_gsm\_if$  can be programmed by setting the corresponding control registers **GSMCLK1T**, **GSMCLK2T**, **GSMCLK3**.

Since the GSM System Interface, especially the GSM Timer Unit, is very sensitive to frequency errors of the system clock, either a voltage controlled quartz oscillator (VCXO) reference is required for the system or the frequency errors of a non-controlled quartz oscillator (XO) have to be compensated. Therefore, a programmable fractional divider with high resolution instead of a simple frequency divider with fixed ratio converts the clock  $hw\_clk$  into the clock  $clk\_gsm\_if$  (**Figure 10-46**). Furthermore, the clock  $clk\_gsm\_if$  is divided by a fixed ratio of 1/6 to the clock  $clk\_counter$  with 2.167 MHz used for the TDMA Counter Unit.

**Figure 10-46 GSM Clock Control Unit**



The fractional divider converts the frequency of  $hw\_clk$  according to:

$$f_{clk\_gsm\_if} = K/L * f_{hw\_clk} \quad [36.5]$$

The numerator K, which is a 30-bit unsigned integer value, is set in the GSM System Interface Clock Control Register 1 (**GSMCLK1T**). The denominator L which is also a 30-bit unsigned integer value is set in the GSM System Interface Clock Control Register 2 (**GSMCLK2T**).

Reprogramming K and L has not an immediate effect, because this would lead to undesired intermediate states. Activation of new K and L values is done by setting appropriate bits in register **GSMCLK3** or by toggling the signal  $cbush$  which is provided by the **Section 10.4.1 Clock Generation Unit (on Page 653)**.

<sup>1)</sup> Refer to the Clock Domain in **System Integration: (on Page 803)**.



**CONFIDENTIAL**

**GSM Timer Unit**

### 10.11.2.1 GSM System Interface Clock Control Registers

**GSMCLK1x**

**GSMCLK1T**

**GSM System Interface Clock Control Register 1 Top**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>		<b>K (29:16)</b>													

**GSMCLK1B**

**GSM System Interface Clock Control Register 1 Bottom**

**Reset value: 0001<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>K (15:0)</b>															

Field	Bits	Type	Description
<b>K (29:16)</b> <b>K (15:0)</b>	13:0 15:0	rw	<b>Numerator of the Fractional Divider</b> K is set according to <a href="#">Table 10-33 K and L Values for Various Kernel Clock Frequency Errors (on Page 833)</a> . <i>Note: It always has to be <math>K &lt; L</math> and <math>K &gt; 0</math>.  <math>K = L</math> is not allowed.</i> <i>Exception: <math>K = L = 0</math> only if INIT (see register <a href="#">GSMCLK3</a>) is used. For <math>K = L = 0</math> <math>clk\_gsm\_if</math> is equal to <math>clk\_kernel</math>.</i> <i>Note: K has to be programmed prior to any action of the GSM System Interface.</i>
<b>RESERVED</b>	15:14 in GSM CLK1T	r	Reserved; these bits must be left at their reset values.

**GSMCLK2x**

**GSMCLK2T**

**GSM System Interface Clock Control Register 2 Top**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>		<b>L (29:16)</b>													

CONFIDENTIAL

GSM Timer Unit

# GSMCLK2B

GSM System Interface Clock Control Register 2 Bottom

Reset value: 0002<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L (15:0)															

Field	Bits	Typ	Description
L (29:16) L (15:0)	13:0 15:0	rw	<b>Denominator of the Fractional Divider</b> The value of <b>L</b> is set in depends on the frequency of the system oscillator according to <a href="#">Table 10-33 K and L Values for Various Kernel Clock Frequency Errors (on Page 833)</a> . <i>Note: <b>L</b> has to be programmed prior to any action of the GSM System Interface.</i>
RESERVED	15:14 in GSM CLK2T	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**GSM Timer Unit**

### GSMCLK3

#### GSM System Interface Clock Control Register 3

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														INIT	LOAD

Field	Bits	Type	Description
<b>LOAD</b>	0	w	<p><b>Load Bit for K and L</b></p> <p>0     <b>LOAD</b> not used to load <b>K</b> and <b>L</b></p> <p><i>Note: <b>LOAD</b> always = 0 when read.</i></p> <p>1     <b>K</b> and <b>L</b> are loaded from <b>GSMCLK1x</b> and <b>GSMCLK2x</b> into the fractional divider and become valid.</p> <p><b>LOAD</b> is used if both:</p> <ul style="list-style-type: none"> <li>New values of <b>K</b> and <b>L</b> are needed because the ratio of the fractional divider has to be slightly adopted to a new deviation <math>\delta_{f_{hw\_clk}}</math> of the <b>hw_clk</b> frequency.</li> <li>Operation of the clock can not be stopped.</li> </ul> <p>If <b>LOAD</b> is set to 1, The internal state of the fractional divider is not initialized, only the new values for <b>K</b> and <b>L</b> become valid.</p> <p><i>Note: If <b>LOAD</b> is used the new values <math>K_{new}</math> and <math>L_{new}</math> have to fulfill the following conditions:</i></p> <p style="text-align: center;"><math>L_{new} - K_{new} &gt; K_{old}</math>; <math>K_{new} &lt; L_{new}</math>; <math>K_{new} &gt; 0</math>.</p> <p><i>Note: If <b>LOAD</b> is used, it is recommended to only change either <b>L</b> or <b>K</b>. Otherwise, a gap in the <b>clk_gsm_if</b> can occur.</i></p> <p><i>Note: If the internal state of the fractional divider has to also be initialized, use <b>INIT</b> instead.</i></p>

**CONFIDENTIAL**

**GSM Timer Unit**

Field	Bits	Type	Description
<b>INIT</b>	1	w	<p><b>INIT Bit for K and L</b></p> <p>0 <b>INIT</b> not used to load <b>K</b> and <b>L</b></p> <p>1 <b>K</b> and <b>L</b> are loaded from <b>GSMCLK1x</b> and <b>GSMCLK2x</b> into the fractional divider and the internal state of the fractional divider is initialized according on the values of <b>K</b> and <b>L</b>. If the frequency of clk_kernel (and clk_bus) changes due to changes in the <a href="#">Section 10.4.1 Clock Generation Unit (on Page 653)</a>, there are two ways to initialize the fractional divider:</p> <p><b>Asynchronous:</b> Configure the CGU, <b>GSMCLK1x</b>, and <b>GSMCLK2x</b>, use the <b>INIT</b> bit to load <b>K</b> and <b>L</b> and initialize the fractional divider.</p> <p><b>Synchronous:</b> Configure the CGU, <b>GSMCLK1x</b>, and <b>GSMCLK2x</b>, and initialize the fractional divider via the cbush signal from the CGU. This changes synchronously the frequency of clk_kernel or clk_bus and the ratio of the fractional divider.</p> <p><i>Note: <b>INIT</b> always = 1 when read.</i></p> <p><i>Note: If <b>INIT</b> = 1, <b>LOAD</b> = 'don't care' because <b>INIT</b> includes the load operation.</i></p>
<b>RESERVED</b>	15:2	r	Reserved; these bits must be left at their reset values.

If a VCXO reference is used, the fractional divider is used for simple integer division.

- For the 26 MHz clock clk\_kernel:

$$K = 0000\ 0001_H \text{ and } L = 0000\ 0002_H \quad [36.6]$$

- For a 52 MHz clock clk\_kernel:

$$K = 0000\ 0001_H \text{ and } L = 0000\ 0004_H \quad [36.7]$$

If an XO is used, the frequency error  $\delta f_{\text{clk\_kernel}}$  must be compensated by appropriate choice of **K** and **L**.

- For a 52 MHz clock clk\_kernel set:

$$K = 1000'0000_H \quad [36.8]$$

**CONFIDENTIAL**

**GSM Timer Unit**

- For a 26 MHz clock clk\_kernel set:

$$K = 0800'0000_H \quad [36.9]$$

In both cases the denominator is:

$$L = \text{round}[2000'0000_H * (1 + \delta f_{\text{clk\_kernel}}/10^6 \text{ ppm})]. \quad [36.10]$$

For an XO with 100 ppm accuracy the maximum error for clk\_gsm\_if is below 0.5 ppb with this setting.

A 13 MHz XO is not supported.

For a 13 MHz VCXO set:

$$K = L = 0. \quad [36.11]$$

**Table 10-33** shows several K and L values for a clock clk\_kernel with and without the minimal and maximal frequency error.

**Table 10-33 K and L Values for Various Kernel Clock Frequency Errors**

clk_kernel in MHz	$\delta f_{\text{clk\_kernel}}$ in ppm	clk_gsm_if in MHz	K	L	System Concept
13	0	13.0	0000 0000 <sub>H</sub>	0000 0000 <sub>H</sub>	VCXO
26	0	13.0	0000 0001 <sub>H</sub>	0000 0002 <sub>H</sub>	VCXO
26	-100	13.0	1000 0000 <sub>H</sub>	1FFF 2E49 <sub>H</sub>	XO
26	100	13.0	1000 0000 <sub>H</sub>	2000 D1B7 <sub>H</sub>	XO
52	0	13.0	0000 0001 <sub>H</sub>	0000 0004 <sub>H</sub>	VCXO
52	-100	13.0	0800 0000 <sub>H</sub>	1FFF 2E49 <sub>H</sub>	XO
52	100	13.0	0800 0000 <sub>H</sub>	2000 D1B7 <sub>H</sub>	XO

### 10.11.3 GSM Timer Decoder

The pseudo code below describes how the signals TRIG[10:6] are decoded to generate the following internal signals: INT\_GP[6:0], EQON, MONON, SCON, FCON, CODON, RXON, and TXON. The signals INT\_GP[6:0] are toggle interrupts, that is, every time the level of INT\_GP[i] toggles an interrupt is generated.

The function of the signals are described in **Table 10-30 GSM Timer Unit Signals**.

```
IF (signal trig_start is active AND TRIG[10:6] = 0)
    no action
ELSEIF (signal trig_start is active AND TRIG[10:6] = 1)
    signal EQON is set to 1
ELSEIF (signal trig_start is active AND TRIG[10:6] = 2)
    signal MONON is set to 1
ELSEIF (signal trig_start is active AND TRIG[10:6] = 3)
    signal SCON is set to 1
```

**CONFIDENTIAL**

**GSM Timer Unit**

```
ELSEIF (signal trig_start is active AND TRIG[10:6] = 4)
    signal FCON is set to 1
ELSEIF (signal trig_start is active AND TRIG[10:6] = 5)
    signals EQON, MONON, SCON and FCON are reset to 0
ELSEIF (signal trig_start is active AND TRIG[10:6] = 6)
    signal RXON is set to 1
ELSEIF (signal trig_start is active AND TRIG[10:6] = 7)
    signal RXON is reset to 0
ELSEIF (signal trig_start is active AND TRIG[10:6] = 8)
    signal TXON is set to 1
ELSEIF (signal trig_start is active AND TRIG[10:6] = 9 )
    signal TXON is reset to 0
ELSEIF (signal trig_start is active AND TRIG[10:6] = 10)
    INT_GP[0] is toggled
ELSEIF (signal trig_start is active AND TRIG[10:6] = 11)
    INT_GP[1] is toggled
ELSEIF (signal trig_start is active AND TRIG[10:6] = 12)
    INT_GP[2] is toggled
ELSEIF (signal trig_start is active AND TRIG[10:6] = 13)
    INT_GP[3] is toggled
ELSEIF (signal trig_start is active AND TRIG[10:6] = 14)
    INT_GP[4] is toggled
ELSEIF (signal trig_start is active AND TRIG[10:6] = 15)
    INT_GP[5] is toggled
ELSEIF (signal trig_start is active AND TRIG[10:6] = 16)
    INT_GP[6] is toggled
ELSEIF (signal trig_start is active AND TRIG[10:6] = 17)
    signal CODON is set to '1'
ELSEIF (signal trig_start is active AND TRIG[10:6] = 18)
    signal CODON is reset to '0'
ELSEIF (signal trig_start is active AND TRIG[10:6] > 18)
    no action
ENDIF
```

CONFIDENTIAL

RF Control

## 10.12 RF Control

History	
Design Spec.	Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.00	
<b>Page 842</b>	Updated <b>Section 10.12.3 Synchronous Serial Interface (3-wire RF interface)</b>
Changes for Rev. 1.01	
<b>Page 835</b>	Updated <b>System Integration</b> : WS00006682
Changes for Rev. 1.06	
<b>Page 843</b>	Updated <b>RFCON1</b> , only have 4 strobe lines WS00009101
<b>Page 836</b>	Updated <b>Section 10.12.2 RF RAM</b> , only have 4 strobe lines WS00009101
<b>Page 847</b>	Updated <b>Section 10.12.4.1 Telegram Control Block</b> , only have 4 strobe lines WS00009101

### System Integration:

- Supply domain: VDD\_MAIN
- Chip internal interfaces:
  - Clock domain: Refer to **Section 10.4.1.3 Sub-System Clocks and Enables (on Page 661)** and see **Figure 10-11 Clock and Enable Generation for MCU Sub-System (on Page 668)**.
  - Bus domain: X-Bus
- Interrupt sources:
- Monitor Pins: Refer to **Section 11.8.10 Internal Signal Monitoring (on Page 1209)**

### 10.12.1 Introduction

The RF Control Unit generates the control information which has to be provided to the RF chip set and the analog part. This information is provided by telegrams of 8, 16, or 24 bits via the RF Interface. The block diagram of the RF Control Unit is shown in **Figure 10-47**.

*Note: For information about the GSM CGU refer to **Section 10.4.1 Clock Generation Unit**.*

**CONFIDENTIAL**

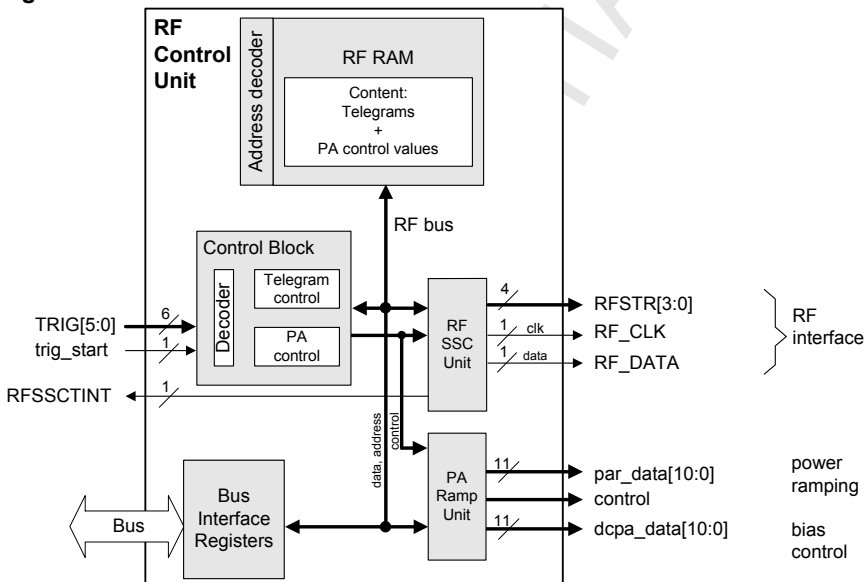
**RF Control**

By means of the signal 'trig\_start' coming from the GSM Timer Unit the transfer of a telegram or power ramp stored in the RF RAM of the RF Control Unit is initiated. Which telegram will be transferred is coded by the 6 signals TRIG[5:0].

The transmission of telegrams can also directly be programmed via the X-Bus interface similar to a SSC interface. This feature is especially suited for the initialization of the RF devices.

Within the RF Control Unit the PA power ramping data is generated as well. The interface to the PA Control Hardware are two 11-bit wide buses with corresponding control signals. Via this interface the power ramping data and additional a PA bias control signal (DCPA) are provided.

**Figure 10-47 RF Control Unit**



### 10.12.2 RF RAM

The RF Control Unit comprises a dual port RF RAM of 448 \* 11 bits which is programmable via the X-Bus. The RAM address range is given in [Table 10-34](#) and [Table 10-35](#). The GSM System Interface base address is defined in [Table 12-4 Address Mapping of X-Bus Peripherals \(on Page 1285\)](#). The RF RAM can store:

- With RF RAM Partitioning Type 1 (refer to [Table 10-34](#)):
  - Up to 40 telegrams (refer to [Table 10-36 Structure of Telegrams \(on Page 840\)](#)), each consisting up to 24 bits of data and 9 bits of control information



CONFIDENTIAL

RF Control

- 16 power ramp sequences, each consisting of 16 values of 11 bits, one 11 bit value for controlling the linear ramping during the active part of the burst, and one 10 bit value **DCPAdat**[a \(on Page 841\)](#) for controlling the power amplifier's bias (refer also to **Section 10.10 RF Power Ramping (on Page 793)**)
- With RF RAM Partitioning Type 2 (refer to **Table 10-35 on page 839**):
  - Up to 112 telegrams (refer to **Table 10-36**) each consisting up to 24 bits of data and 9 bits of control information.

The partitioning type is selected in the RF Control Register 1 (**RFCON1 (on Page 843)**).

*Note: RF RAM Partitioning Type 2 is especially suited for those cases where the internal power ramping and the output PAOUT1 is not needed, for example, if an external modulator with separate PA control is used. Usually, in this case more telegram capacity is needed to control the external modulator.*

**Table 10-34 RF RAM Partitioning of the RF Control Unit (Type 1)**

Physical address <sup>1)</sup> (Hex)	Name	Function
	RAM entries for telegrams T1 to T40	
000	T1	Definition of telegram 1: Word 1: 000 <sub>H</sub> Word 2: 002 <sub>H</sub> (trigger value: 8 <sub>D</sub> ) Word 3: 004 <sub>H</sub> Word 4: 006 <sub>H</sub>
008	T2	Definition of telegram 2 (trigger value: 9 <sub>D</sub> )
...	...	...
130	T39	Definition of telegram 39 (trigger value: 46 <sub>D</sub> )
138	T40	Definition of telegram 40 (trigger value: 47 <sub>D</sub> )
	RAM entries for power ramp sequence 1 start at 140 <sub>H</sub> (trigger value: 48 <sub>D</sub> )	
140	DCPAdat1	PA bias data for PA ramp 1 (refer to <b>DCPAdat</b> <a href="#">a (on Page 841)</a> )
142	PAR1(1)	1. power ramp value [10:0]
...	...	...
160	PAR1(16)	16. power ramp value [10:0]
162	PAINC1	1st incremental value for linear power ramping [10:0]
	RAM entries for power ramp sequence 2 start at 164 <sub>H</sub> (trigger value 49 <sub>D</sub> )	
164	DCPAdat2	PA bias data for PA ramp 2 (refer to Table DCPAdat)
166	PAR2(1)	1. power ramp value [10:0]
...	...	...

**CONFIDENTIAL**

**RF Control**

**Table 10-34 RF RAM Partitioning of the RF Control Unit (Type 1) (cont'd)**

Physical address <sup>1)</sup> (Hex)	Name	Function
184	PAR2(16)	16. power ramp value [10:0]
186	PAINC2	2nd incremental value for linear power ramping [10:0]
	RAM entries for power ramp sequence 3 start at 188 <sub>H</sub> (tigger value 50 <sub>D</sub> )	
	RAM entries for power ramp sequence 4 start at 1AC <sub>H</sub> (tigger value 51 <sub>D</sub> )	
	RAM entries for power ramp sequence 5 start at 1D0 <sub>H</sub> (tigger value 52 <sub>D</sub> )	
	RAM entries for power ramp sequence 6 start at 1F4 <sub>H</sub> (tigger value 53 <sub>D</sub> )	
	RAM entries for power ramp sequence 7 start at 218 <sub>H</sub> (tigger value 54 <sub>D</sub> )	
	RAM entries for power ramp sequence 8 start at 23C <sub>H</sub> (tigger value 55 <sub>D</sub> )	
	RAM entries for power ramp sequence 9 start at 260 <sub>H</sub> (tigger value 56 <sub>D</sub> )	
	RAM entries for power ramp sequence 10 start at 284 <sub>H</sub> (tigger value 57 <sub>D</sub> )	
	RAM entries for power ramp sequence 11 start at 2A8 <sub>H</sub> (tigger value 58 <sub>D</sub> )	
	RAM entries for power ramp sequence 12 start at 2CC <sub>H</sub> (tigger value 59 <sub>D</sub> )	
	RAM entries for power ramp sequence 13 start at 2F0 <sub>H</sub> (tigger value 60 <sub>D</sub> )	
	RAM entries for power ramp sequence 14 start at 314 <sub>H</sub> (tigger value 61 <sub>D</sub> )	
	RAM entries for power ramp sequence 15 start at 338 <sub>H</sub> (tigger value 62 <sub>D</sub> )	
	RAM entries for power ramp sequence 16 start at 35C <sub>H</sub> (tigger value 63 <sub>D</sub> )	

<sup>1)</sup> plus RAM base address

**Table 10-35 RF RAM Partitioning of the RF Control Unit (Type 2)**

Physical address <sup>1)</sup> (Hex)	Name	Function
RAM entries for telegrams T1 to T112		
000	T1	Definition of telegram 1: Word 1: 000 <sub>H</sub> Word 2: 002 <sub>H</sub> (trigger value 1 <sub>D</sub> ) Word 3: 004 <sub>H</sub> Word 4: 006 <sub>H</sub>
008	T2	Definition of telegram 2 (no trigger value)
010	T3	Definition of telegram 3 (trigger value 2 <sub>D</sub> )
...	...	...
288	T98	Definition of telegram 98 (no trigger value)
290	T99	Definition of telegram 99 (trigger value 50 <sub>D</sub> )
...	...	...
370	T111	Definition of telegram 111 (trigger value 62 <sub>D</sub> )
378	T112	Definition of telegram 112 (trigger value 63 <sub>D</sub> )

<sup>1)</sup> Plus RAM base address

**CONFIDENTIAL**

**RF Control**

**Table 10-36 Structure of Telegrams**

Word	Bit	Function
1	0	<b>Multiple Telegram Transmission Control (Burst Mode)</b> 0 No following telegram is automatically transmitted after this telegram 1 The telegram i + 1 is automatically transmitted after this telegram, that is, telegram i. Refer to <a href="#">Section 10.12.4 Control Block (on Page 847)</a> .
	3:1	<b>RF SSC Unit Strobe Select Bits (SSCSB[2:0])</b> 000 RFSTR[0] is active during transmission of telegram i 001 RFSTR[1] is active during transmission of telegram i 010 RFSTR[2] is active during transmission of telegram i 011 RFSTR[3] is active during transmission of telegram i 100 Reserved 101 Reserved 110 Reserved 111 Reserved
	5:4	<b>RF SSC Unit Telegram Length Bits (TLB[1:0], Corresponds to SSCBM)</b> 00 8 bit long telegram 01 16 bit long telegram 10 24 bit long telegram 11 Not allowed
	6	<b>Reserved<sup>1)</sup></b>
	7	<b>RF SSC Unit Clock Frequency Bit (SSCFB)</b> 0 Telegram transfer with 6.50 MHz bit clock 1 Telegram transfer with 3.25 MHz bit clock
	8	<b>RF SSC Unit Clock Phase Select Bit (SSCPB)</b> 0 Shift transmit data on leading clock edge 1 Shift transmit data on trailing edge
	9	<b>RF SSC Unit Heading Control Bit (SSCHB)</b> 0 Transmit LSB first 1 Transmit MSB first
	10	<b>Reserved<sup>1)</sup></b>
	7:0	<b>Telegram Data</b> IF (TLB[1:0] = 00) THEN bits [7:0] of telegram ELSIF (TLB[1:0] = 01) THEN bits [15:8] of telegram ELSIF (TLB[1:0] = 10) THEN bits [23:16] of telegram
	10:8	<b>Reserved<sup>1)</sup></b>

CONFIDENTIAL

RF Control

**Table 10-36 Structure of Telegrams (cont'd)**

Word	Bit	Function
3	7:0	<b>Telegram Data</b> IF (TLB[1:0] = 01) THEN bits [7:0] of telegram ELSIF (TLB[1:0] = 10) THEN bits [15:8] of telegram
	10:8	<b>Reserved<sup>1)</sup></b>
4	7:0	<b>Telegram Data</b> IF (TLB[1:0] = 10) THEN bits [7:0] of telegram
	10:8	<b>Reserved<sup>1)</sup></b>

<sup>1)</sup> always set to '0'.

### 10.12.2.1 DCPA Data Entry in RF Control Unit RAM

DCPAdata

DCPA Data Entry in RF Control Unit RAM

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED					SYN C	DCPAvalue									

Field	Bits	Type	Description
DCPAvalue	9:0	rw	<b>DAC Value for DCPA Output</b> The DAC Value is equal to 2 times DCPAvalue. 000 <sub>H</sub> DAC Value = 0 ... ... 3FF <sub>H</sub> DAC Value = 7FF <sub>H</sub>
SYNC	10	rw	<b>Synchronization of DAC with PA Ramp</b> 0 The DCPAvalue is getting valid with the beginning of the PA ramp it is associated with. 1 The DCPAvalue is getting valid after DELAY * 24 clk_gsm_if clock cycles after the beginning of the PA ramp it is associated with (refer to <a href="#">Table 10-34 on page 837</a> ).  <i>Note: Setting SYNC = 1 and DELAY = 0 is identical to setting SYNC = 0 and DELAY to any arbitrary value.</i>
RESERVED	15:11	r	Reserved; these bits must be left at their reset values.

### **10.12.3 Synchronous Serial Interface (3-wire RF interface)**

The Synchronous Serial Interface (RF SSC Unit) provides a high speed serial communication between the E-GOLDRadio and external peripherals like the RF ICs via a 3-wire interface. This interface is only accessible when in a certain test mode.

The RF SSC Unit generates the strobe signals RFSTR[3:0], the clock signal RF\_CLK and the serial data stream RF\_DATA.

The format of the data stream which has to be transferred has to be provided in the RF control registers **RFCON1** and **RFCON2 (on Page 843)** of the RF Control Unit in advance to the data transmission. The register **RFCON1** stores the control information as far as these control information is identical for all transferred data streams. The RF control register 2 (**RFCON2 (on Page 843)**) stores the control information as far as these control information can vary from telegram to telegram. The data bits which are transferred by the RF SSC Unit are stored in the RF SSC Unit transmit buffer register (**RFSSCTB (on Page 845)**).

CONFIDENTIAL

RF Control

### 10.12.3.1 RF Control Register 1

RFCON1

RF Control Register 1

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						RAM TY PE	RFI SSC P	RESERVED				STBSEL			

Field	Bits	Type	Description
STBSEL	3:0	rw	<b>Strobe Select</b> For each bit <b>STBSEL[i]</b> the meaning is: 0 RFSTR[i] is low active 1 RFSTR[i] is high active
RFISSCP	8	rw	<b>SSC Clock Polarity of RF Interface</b> 0 Idle clock line is low, leading clock edge is low to high transition 1 Idle clock line is high, leading clock edge is high to low transition
RAMTYPE	9	rw	<b>RF Control Unit RAM Partitioning Type</b> 0 Type 1 is selected (refer to <a href="#">Table 10-34 on page 837</a> ) 1 Type 2 is selected (refer to <a href="#">Table 10-35 on page 839</a> )
RESERVED	15:10, 7:4	r	Reserved; these bits must be left at their reset values.

### 10.12.3.2 RF Control Register 2

RFCON2

RF Control Register 2

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSC EN	RES ERV ED	SSC FB	RESERV ED	SSC SB			RESERV ED	SSC PB	SSC HB	SSC BM					

**CONFIDENTIAL**

**RF Control**

Field	Bits	Type	Description
<b>SSCBM</b>	3:0	rw	<b>RF SSC Unit Telegram Length Control</b> The telegram length is defined by <b>SSCBM + 1</b> . 0000 Do not use 0001 Telegram length is 2 bits 0010 Telegram length is 3 bits ... .. 1111 Telegram length is 16 bits
<b>SSCHB</b>	4	rw	<b>RF SSC Unit Heading Control</b> 0 Transmit LSB first 1 Transmit MSB first
<b>SSCPB</b>	5	rw	<b>RF SSC Unit Clock Phase Control</b> 0 Shift transmit data on leading clock edge 1 Shift transmit data on trailing edge
<b>SSCSB</b>	10:8	rw	<b>RF SSC Unit Strobe Select</b> 000 RFSTR[0] is active during transmission 001 RFSTR[1] is active during transmission 010 RFSTR[2] is active during transmission 011 RFSTR[3] is active during transmission 100 Reserved 101 Reserved 110 Reserved 111 Reserved
<b>SSCFB</b>	13	rw	<b>RF SSC Unit Clock Frequency</b> 0 Telegram transfer with 6.50 MHz bit clock 1 Telegram transfer with 3.25 MHz bit clock
<b>SSCEN</b>	15	rw	<b>RF SSC Unit Enable</b> 0 Transmission is disabled 1 Transmission is enabled
<b>RESERVED</b>	7:6, 12:11, 14	r	Reserved; these bits must be left at their reset values.



**CONFIDENTIAL**

**RF Control**

### 10.12.3.3 RF SSC Transmit Buffer

**RFSSCTB**

**RF SSC Transmit Buffer**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RFDATA</b>															

Field	Bits	Type	Description
<b>RFDATA</b>	15:0	rw	<b>RF Data</b> Data bits which are transferred via the RF SSC Unit <i>Note: This register cannot be accessed byte-wide.</i>

Whenever new data is transferred to the register **RFSSCTB** and the previous transmission is finished (for example, the 16-bit shift register is empty) and the bit **RFSSCTB.SSCEN** is set to 1, the content of the register **RFSSCTB** is copied to the 16-bit wide shift register, an interrupt **RFSSCTINT** is generated and the transmission is started according to the control parameters of the registers **RFCON1** and **RFCON2**.

The RF SSC Unit is programmed either from the Control Block of the RF Control Unit or directly via the X-Bus.

### 10.12.3.4 Programming via the Control Block

If the transmission of a telegram is initialized by the GSM Timer Unit the programming of the RF SSC Unit is mainly done by the Control Block of the RF Control Unit. Only the register **RFCON1** has to be programmed by the controller. The registers **RFCON2** and **RFSSCTB** are automatically programmed by the Control Block according to the data and control information of the RF Control RAM. Nevertheless, the interrupt **RFSSCTINT** is generated and has therefore to be disabled at the interrupt control unit of the controller.

### 10.12.3.5 Programming Directly via the X-Bus

A telegram can also directly be programmed via the X-Bus interface. A transmission is started when the bit **RFCON2.SSCEN** is set to 1 and a write operation has been performed in the register **RFSSCTB**. It does not matter whether the register **RFSSCTB** or **RFCON2** is programmed first. The interrupt **RFSSCTINT** is generated by the RF Control Unit when the **RFSSCTB** data has been copied to the shift register. Therefore, the software can rewrite the register **RFSSCTB** again after the interrupt has occurred. If the register **RFSSCTB** is rewritten before the transmission of the previous telegram - the content of which is now in the shift register - is finished, a continuous transmission of both telegrams is performed. By such means a 24-bit telegram to the RF IC can be generated by programming two 12-bit wide words into the register **RFSSCTB**. When the

CONFIDENTIAL

RF Control

transmission of the telegram is finished the bit **RFCON2.SSCEN** is cleared by hardware which enables the software to detect whether the transmission has been finished or not.

*Note: There is no hardware protection mechanism which takes care that the RF SSC Unit registers are NOT written at the same time by the Control Block of the RF Control Unit and the controller via the X-Bus. This has to be guaranteed by software. Therefore, it is recommended to program the RF SSC Unit directly via the X-Bus only for the initialization procedure of the RF devices when the GSM Timer Unit does not initiate any telegram transfer.*

### 10.12.3.6 Application Note

To transmit a telegram of 24 bits, it has to be split into two parts of 12 bits. After sending the first 12 bits, the second part has to be written immediately in the transmit buffer register **RFSSCTB** to avoid a gap in the telegram transmission. It is not necessary to set the bit **RFCON2.SSCEN** again if the second value is written fast enough.

This pseudo-code sends two 12-bit telegram with a 6.5 MHz bit clock:

(...)

**RFCON1** = 0000<sub>H</sub>;

**RFCON2** = 853B<sub>H</sub>;

**RFSSCTB** = 0222<sub>H</sub>;

**RFSSCTB** = 0FFF<sub>H</sub>;

(...)

*Note: The use of C code to send two 12-bit telegrams is not fast enough. If C code is used, only the first telegram is sent. Assembly code for the RFSSC must be used to be fast enough.*

## 10.12.4 Control Block

### 10.12.4.1 Telegram Control Block

The telegram control block has to take care that the control information for each telegram and the telegram data are transferred from the RF RAM at the appropriate point in time to the RF SSC Unit. In addition it has to trigger the start of the telegram transmission.

The principle timing of an 8 bit telegram with a low active strobe signal (STBSEL[i] = 0) and telegram control bits set to:

**RFCON2.SSCHB** = 0

TLB = 00 (**RFCON2.SSCBM** = 7<sub>D</sub>)

See **Figure 10-48**.

#### Single Telegram Transmission

Telegrams and power ramp data can be transmitted in parallel. When the signal 'trig\_start' is activated by the GSM Timer Unit, a telegram is transferred.

1. The GSM timer Unit sends a trigger signal to the RF Control Block, which initializes the telegram control part of the Control Block and initiates the telegram transmission. This trigger signal 'trig\_start' is provided by a timing event in the GSM Timer Unit.
2. The telegram data is transferred from the RF Control Unit RAM to the synchronous serial interface (RF SSC Unit).
3. In the RF SSC Unit each 8, 16 or 24 bit telegram is transmitted immediately via the 3-wire bus to the RF IC.

When the telegram is transferred via the serial interface with a 6.5MHz or 3.25MHz bit clock the serial transmission on RF\_DATA starts after a fixed delay of 11 clk\_gsm\_if clock cycles after the CTDMA Counter has reached the corresponding timing compare value.

*Note: For single telegram transmission the Multiple Telegram Transfer Control bit must always be set to 0.*

#### Multiple Telegram Transmission

In multiple telegram transmission mode a series of telegrams can be transmitted automatically only needing one trigger event by the GSM Timer Unit. The minimum series length is 1 (that is, single telegram transmission is also included), the maximum series length is 40 or 112 (depending on the RAM partitioning type). The RF RAM address of the first telegram of a series is encoded in the bits TRIG[5:0] of the timing event in the GSM Timer Unit and every telegram contains a Multiple Telegram Transfer Control Bit.

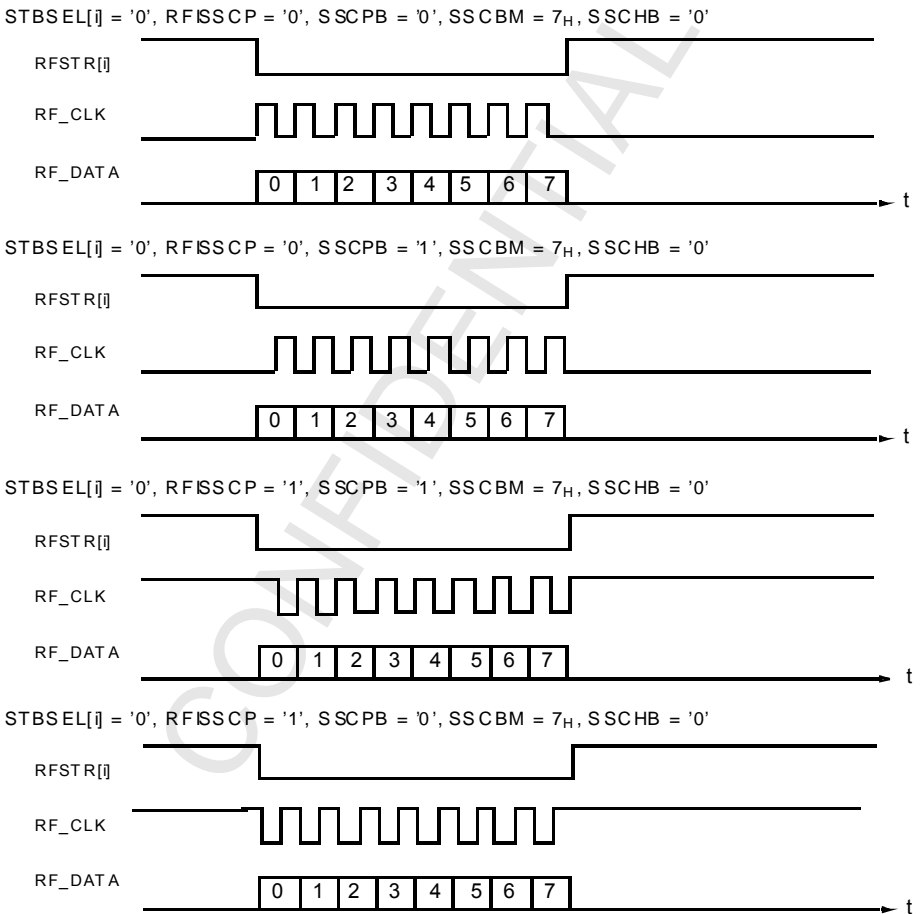
When initiated by the trigger signal 'trig\_start' the first telegram is read out of the RF RAM and the Multiple Telegram Transfer Control Bit is analyzed. If the Multiple Telegram

CONFIDENTIAL

RF Control

Transfer Control Bit is set to 1, the next telegram with index  $i + 1$  (this is not the trigger value) will be automatically transmitted after the telegram with index  $i$  without the signal 'trig\_start' signal being activated. In this way a series of consecutive telegrams with variable length can be transmitted with only one entry in the GSM Timer RAM. The last telegram of a series is indicated by setting the Multiple Telegram Transfer Control Bit to 0.

**Figure 10-48 Principle Timings of 8 Bit Telegrams**



*Note: In this implementation the number of telegrams to be transferred in multiple telegram transmission mode is **not** encoded in the first telegram to determine the end of transmission. By the use of the Multiple Telegram Transfer Control Bit the*

CONFIDENTIAL

RF Control

*end of the series of consecutive telegrams is determined by the first telegram index where this bit is set to 0, thus, avoiding the implementation of a telegram counter.*

When the signal 'trig\_start' is activated by the GSM Timer Unit, the first telegram of the series is transferred. If this telegram is transferred over the serial interface with 6.5 MHz or 3.25 MHz, the transmission starts after a fixed delay of 11 clk\_gsm\_if clock cycles after the CTDMA Counter has reached the corresponding timing compare value. The transmission of the telegram with index  $i + 1$  starts after a fixed delay of 5 clk\_gsm\_if clock cycles after the signal RFSTR of the telegram with index  $i$  goes inactive.

*Note: If the Multiple Telegram Control Bit is set to 1 in the last telegram of the RF RAM memory space, that is, telegram  $i = 40$  in RF RAM partitioning type 1 and  $i = 112$  in RF RAM partitioning type 2, the series of telegrams is aborted because there is no next telegram available.*

*Note: The RFSTR[i] signal is active during each telegram and inactive between the transmission of telegrams. It changes RFSTR(i) to RFSTR(j) from telegram to telegram to allow transmission of telegrams to different receivers during a series of telegrams.*

*Note: During transmission of telegrams (single or multiple mode) the GSM Timer Unit can generate all timer events except telegram transmission. The user has to be careful that the transmission of a series of telegrams is finished prior to any new telegram transmission event in the GSM Timer Unit. Otherwise, the new telegram transmission event is ignored.*

**Note: When the signal trig\_start is activated and the previous telegram (or a series of telegrams) is still being transmitted, the signal trig\_start is ignored. Therefore, the programmer of the GSM Timer Unit must ensure that the transmission of a telegram is never requested before the transmission of the last telegram is finished.**

*Note: Power ramp sequences cannot be transmitted during multiple telegram transmission.*

**CONFIDENTIAL**

**RF Control**

### Telegram Index Mapping

Which telegram is transferred is determined by the signal lines TRIG[5:0] with  $TRIG[5:0] = i$ .

For **RF RAM Partitioning Type 1** (refer to [Table 10-34 on page 837](#))  $i = TRIG[5:0]$  is decoded as follows:

```
IF (i < 8D) THEN
    don't transfer anything
ELSIF (i < 48D) THEN
    transmit telegram (i - 7D)
ELSIF
    transmit power ramp sequence (i - 47D)
ENDIF
```

For **RF RAM Partitioning Type 2** (refer to [Table 10-35 on page 839](#))  $i$  is decoded as follows:

```
IF (i = 0) THEN
    don't transfer anything
ELSIF (i < 50D) THEN
    transmit telegram (2*i - 1)
ELSIF
    transmit telegram (i + 49D)
ENDIF
```

*Note: TRIG[5:0] indicates either the index of a single telegram or the index of the first telegram of a series of telegrams in multiple telegram transmission mode.*

### 10.12.4.2 PA Control Block

The PA Control Block controls the data transfers and timing aspects within the RF Control Unit necessary to provide the PA Ramping Unit (refer to [Section 10.10 RF Power Ramping \(on Page 793\)](#)) with information used to generate the signals for the PA Control Hardware (refer to [Section 10.5.7.1 PA Control Hardware Offset Measurement PAOUTOF1 \(on Page 729\)](#)).

**CONFIDENTIAL**

**External Bus Unit**

## 10.13 External Bus Unit

<b>History</b>	
Design Spec.	Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.01	
<b>Page 877</b>	Updated <b>System Integration</b> : for Page Mode Flash Control Unit WS00005473
<b>Page 1115</b>	Updated <b>Figure 10-81 Demultiplexed Read Access Timing Diagram</b> WS00006716
<b>Page 852</b>	Updated <b>System Integration</b> : for EBU WS00006682
<b>Page 865</b>	Updated <b>Figure 10-54 Programmable External Bus Cycle</b> WS00006716
<b>Page 884</b>	Updated <b>Section 10.13.7.3.2 When to Use These Timers</b> WS00006816
Changes for Rev. 1.02	
<b>Page 854</b>	Updated <b>Section 10.13.3.1 Multiplexed Bus Modes (Not Supported)</b> WS00007327
<b>Page 877</b>	Updated <b>System Integration</b> : for Page Mode Flash Control Unit WS00006682
<b>Page 890</b>	Updated <b>Section 10-62 Example Read Access at 52MHz</b>
<b>Page 900</b>	Added <b>Section 10.13.8 Computation of Wait States and Examples of Read Accesses</b> WS00007373, WS00007323, and WS00007398
<b>Page 889</b> <b>Page 891</b>	Updated <b>Section 10.13.7.5.2 Page Mode Control Logic</b> and added <b>Figure 10-63 Example Read Access at 78MHz</b> WS00007398
Changes for Rev. 1.03	
<b>Page 866</b>	Updated <b>Section 10.13.4.1 ALE Length Control</b> WS00007321
<b>Page 900</b>	Updated <b>Section 10.13.8 Computation of Wait States and Examples of Read Accesses</b> WS00007327
<b>Page 862</b>	Updated <b>CS Signal Generation</b> WS00007602
<b>Page 888</b>	Added footnote to <b>PMC0/1.MEMCS</b> WS00007842
Changes for Rev. 1.04	
<b>Page 871</b>	Updated <b>BUSCONx</b> registers WS00008455
<b>Page 889</b>	Updated text in Note for <b>Section 10.13.7.5.2 Page Mode Control Logic</b> WS00008455
<b>Page 853</b>	Updated <b>Figure 10-49 SFRs and Port Pins Associated with the External Bus Interface</b> (ALE -> ADV) WS00008455

<b>History</b>	
Changes for Rev. 1.05	
<b>Page 877</b> <b>Page 885</b>	Page Mode Switch is not supported in E-GOLDradio V2.1F to V2.2x WS00008749, WS00008750
<b>Page 893</b>	Added <b>Section 10.13.8 EBU Application Notes</b>
Changes for Rev. 1.06	
<b>Page 870</b>	Node about LCD displays added to <b>Section 10.13.5.1 Bus Control Functions in the BUSCON Registers</b> WS00008326

### **System Integration:**

- Supply domain: VDD\_MAIN
- Chip internal interfaces:
  - Clock domain: cgu\_c166\_clk\_pos1\_o, cgu\_c166\_clk\_pos2\_o, and cgu\_c166\_clk\_neg\_o.
  - Bus domain: X-Bus
  - Interrupt sources:
- Monitor Pins: Refer to **Section 11.8.10 Internal Signal Monitoring (on Page 1209)**

### **10.13.1 Introduction**

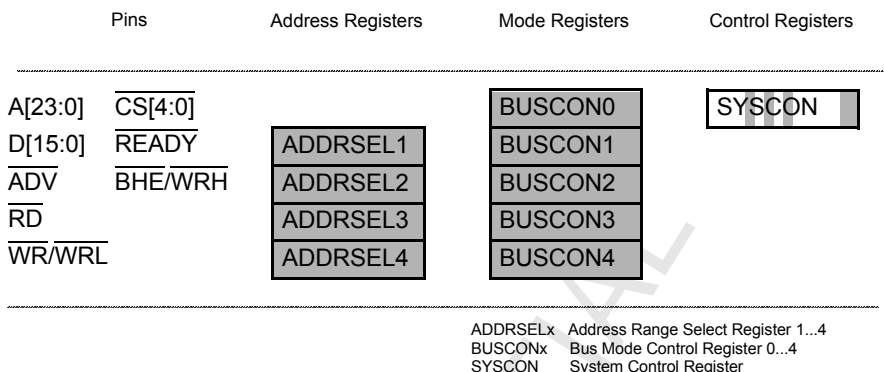
Although the E-GOLDradio C166S subsystem supports a powerful set of on-chip peripherals and on-chip RAM and ROM areas, these internal units cover only a small fraction of the chip address space (up to 16 MBytes). The external bus interface allows access to external peripherals and additional volatile and non-volatile memory. The external bus interface has a number of possible configurations, so it can be tailored to fit perfectly into a given application system.



CONFIDENTIAL

External Bus Unit

Figure 10-49 SFRs and Port Pins Associated with the External Bus Interface



### 10.13.3 External Bus Modes

When the external bus interface is enabled (bit **BUSCONx.BUSACTx** = 1) and configured (bitfield **BUSCONx.BTYP**), E-GOLDradio uses a subset of its port lines together with control lines to build the external bus.

**Table 10-37 Summary of External Bus Modes**

BTYP Encoding	External Data Bus Width	External Address Bus Mode
00	8-bit Data	Demultiplexed Addresses
01	-	Reserved
10	16-bit Data	Demultiplexed Addresses
11	-	Reserved

The bus configuration (BTYP) for the address windows is selected via software in **BUSCON4...BUSCON1**, typically during the initialization of the system.

The external bus configuration at RESET for **BUSCON0** is described in [Chapter 14 System Reset \(on Page 1401\)](#). Further information on external bus signals is contained in [Section 6.10 Configuring External Bus and MCU Signals \(on Page 313\)](#).

Otherwise, **BUSCON0** may be programmed via software just like the other **BUSCON** registers.

The 16-MByte address space of the C166S is divided into 256 segments of 64 kBytes each. The lower 22 address bits have dedicated pins. In addition 5 chip select (**CS**) lines may be used to select different memory banks or peripherals. Only 2 **CS** and **WR** and **RD** lines are available immediately after Reset.

#### 10.13.3.1 Multiplexed Bus Modes (Not Supported)

The multiplexed bus modes are not supported in the E-GOLDradio.

#### 10.13.3.2 Demultiplexed Bus Modes

In the demultiplexed bus modes no address latches are required.

The EBC initiates an external access by placing an address on the address bus. After a programmable period of time, the EBC activates the appropriate command signal (**RD**, **WR**, **WRL**, **WRH**). Data is driven onto the data bus either by the EBC (for write cycles) or by the external memory/peripheral (for read cycles). After a period of time determined by the access time of the memory/peripheral, data becomes valid.

**Read cycles:** Input data is latched and the command signal is deactivated. This causes the accessed device to remove its data from the data bus which is then tri-stated again (refer to [Table 10-51 Demultiplexed Bus, Read Access \(on Page 857\)](#)).

**CONFIDENTIAL**

**External Bus Unit**

**Write cycles:** The command signal is deactivated. If a subsequent external bus cycle is required, the EBC places the relevant address on the address bus. The data remain valid on the bus until the next external bus cycle is started (refer to [Table 10-50 Demultiplexed Bus, Write Access \(on Page 856\)](#)).

CONFIDENTIAL

Figure 10-50 Demultiplexed Bus, Write Access

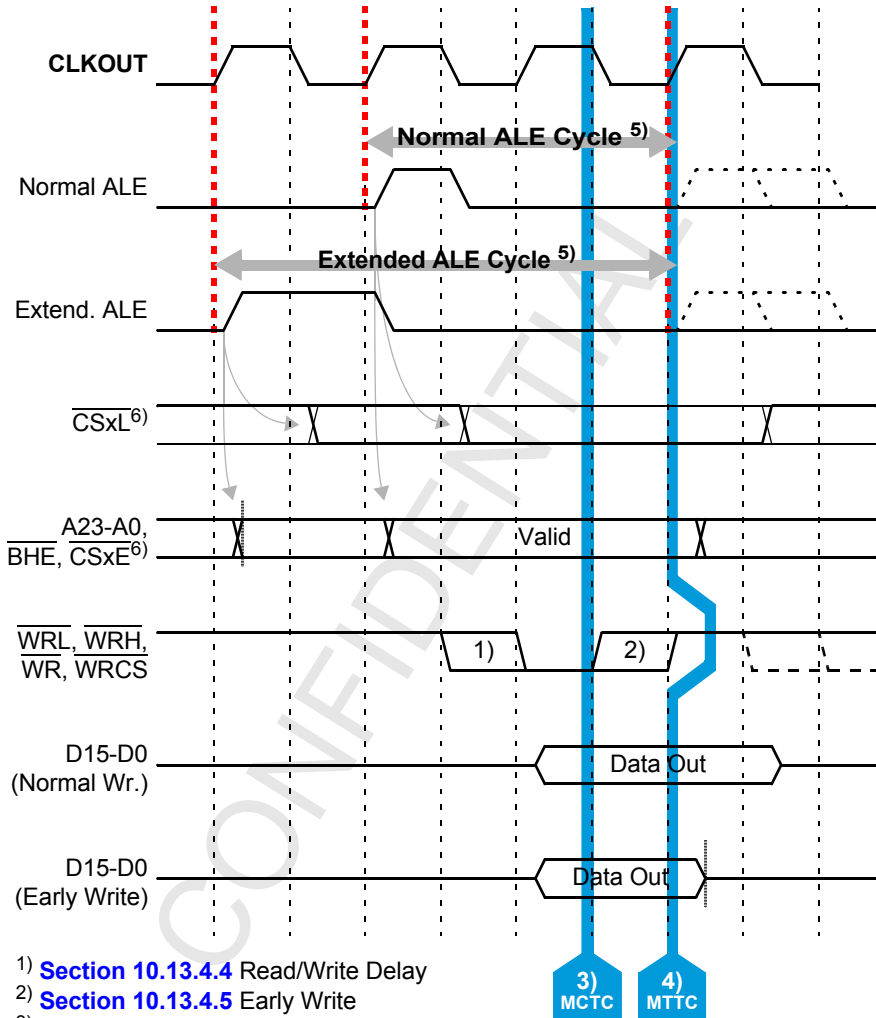
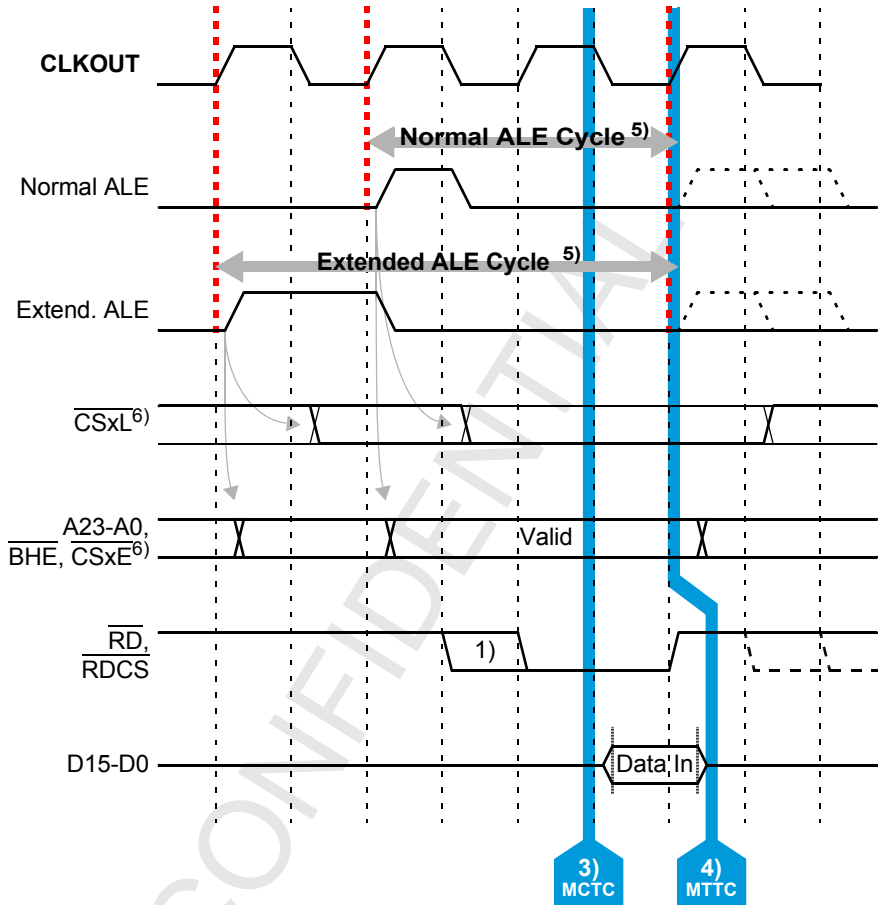


Figure 10-51 Demultiplexed Bus, Read Access



- 1) [Section 10.13.4.4 Read/Write Signal Delay](#)
- 3) [Section 10.13.4.2 Programmable Memory Cycle Time](#)
- 4) [Section 10.13.4.3 Programmable Memory Tri-State Time](#)
- 5) [Section 10.13.4.1 ALE Length Control](#)
- 6) [Section 10.13.3.3 Switching Among the Bus Modes: CS Signal Generation](#)

### 10.13.3.3 Switching Among the Bus Modes

The EBC allows dynamic switching among different bus modes, that is, subsequent external bus cycles may be executed in different ways. Certain address areas can use an 8-bit or 16-bit data bus and predefined waitstates.

A change of the external bus characteristics can be initiated in two different ways:

- **Reprogramming the `BUSCONx` and/or `ADDRSELx` registers**

This allows either:

- The bus mode to be changed for a given address window
- Changing the size of an address window that uses a certain bus mode.

Reprogramming makes it possible to use a great number of different address windows (more than BUSCONs are available), although there is some overhead for changing the registers and keeping appropriate tables.

- **Switching between predefined address windows**

This automatically selects the bus mode that is associated with the respective window. Predefined address windows allow the use of different bus modes without any overhead, but restrict the number of windows to the number of BUSCONs. However, as `BUSCON0` controls all address areas that are not covered by the other BUSCONs, there may be gaps between windows that use the bus mode of `BUSCON0`.

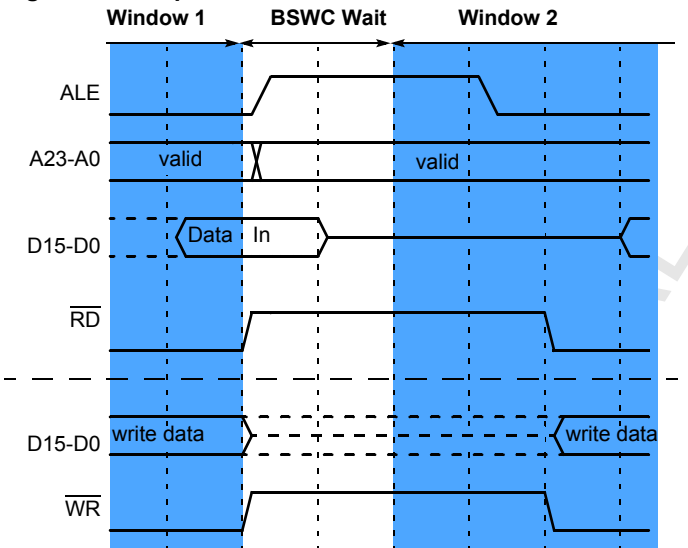
*Note: Never change the configuration for an address area that currently supplies the instruction stream. Due to internal pipelining, it is very difficult to determine the first instruction fetch that will use the new configuration. Only change the configuration for address areas that are not currently accessed. This applies to `BUSCONx` registers as well as to `ADDRSELx` registers.*

BUSCON/ADDRSEL register use is controlled via the addresses issued. When an access (code fetch or data) is initiated, the generated physical address determines whether the access is made internally, uses one of the address windows defined by `ADDRSEL4...ADDRSEL1` or uses the default configuration in `BUSCON0`. After initializing the active registers, they are selected and evaluated automatically by interpreting the physical address. No additional switching or selecting is necessary during run time, except when more than four address windows plus the default (`BUSCON0`) are to be used.

**Switching between external resources** (for example, for different peripherals) may create a problem if the previously-accessed resource needs too much time to switch off its output drivers (after a read), and if the resource to be accessed next switches on its output drivers very fast. In systems running on higher frequencies, this may lead to a bus conflict (switch-off delays normally are independent from the clock frequency).

In such a case, an additional waitstate can automatically be inserted when leaving a given address window, that is, when the next cycle accesses a different window. This is shown in [Figure 10-52](#).

Figure 10-52 Optional BSWC Wait between BUSCON Windows



BUSCON switch waitstates are enabled via bits **BUSCONx.BSWCx**. By enabling the automatic BUSCON switch waitstate (**BSWCx = 1**), there is no impact on the system performance as long as the external bus cycles access the same address window. If the following cycle accesses a different window, a waitstate is inserted between the last access to the previous window and the first access to the new window.

After reset, no BUSCON switch waitstates are selected.

### External Data Bus Width

The EBC can operate on a mixture of 8-bit- or 16-bit-wide external memory/peripherals. The 8-bit data accesses only use the lower data lines. If only 8-bit memories and peripherals are used, the upper lines can be reprogrammed to alternative functions. The EBC can control word accesses on an 8-bit data bus and byte accesses on a 16-bit data bus.

**Word accesses on an 8-bit data bus** are automatically split into two subsequent byte accesses in which the low byte is accessed first. The assembly of bytes to words and the disassembly of words into bytes is handled by the EBC and is transparent to the MCU and the programmer.

CONFIDENTIAL

External Bus Unit

**Byte accesses on a 16-bit data bus** require that the upper and lower half of the memory can be accessed individually. In this case, the upper byte is selected with the Byte High Enable  $\overline{\text{BHE}}$  signal, while the lower byte is selected with the A0 signal. The two bytes of the memory can be enabled either:

- Independently of each other
- Together when accessing words.

When writing bytes to an external 16-bit device that has a single  $\overline{\text{CS}}$  input and two  $\overline{\text{WR}}$  enable inputs (for the two bytes), the EBC can generate these two write control signals directly. This saves the external combination of the  $\overline{\text{WR}}$  signal with A0 or  $\overline{\text{BHE}}$ . In this case, pin  $\overline{\text{WR}}$  serves as  $\overline{\text{WRL}}$  (Write Low byte) and pin  $\overline{\text{BHE}}$  serves as  $\overline{\text{WRH}}$  (Write High byte). **SYSCON.WRCFG** selects the operating mode for pins  $\overline{\text{WR}}$  and  $\overline{\text{BHE}}$ . The respective byte is written on both data bus halves.

When reading bytes from an external 16-bit device, whole words may be read and the C166S automatically selects the byte to be input and discards the other. However, be careful when reading devices that change state when being read (such as FIFOs, interrupt status registers, etc.). In these cases, individual bytes must be selected using  $\overline{\text{BHE}}$  and A0.

#### Disable/Enable Control for Pin $\overline{\text{BHE}}$ (BYTDIS)

**SYSCON.BYTDIS** is provided for controlling the active low Byte High Enable ( $\overline{\text{BHE}}$ ) pin. The function of the  $\overline{\text{BHE}}$  pin is enabled if the **BYTDIS** bit contains a 0. Otherwise, it is disabled and the pin can be used as a standard I/O pin. The  $\overline{\text{BHE}}$  pin is used implicitly by the EBC to select one of two byte-organized memory chips, which are connected to the C166S via a word-wide external data bus. After reset, the  $\overline{\text{BHE}}$  function is automatically enabled (**BYTDIS** = 0) if a 16-bit data bus is selected during reset; otherwise it is disabled (**BYTDIS** = 1). It may be disabled if byte access to 16-bit memory is not required and if the  $\overline{\text{BHE}}$  signal is not used.

#### Summary of Use of $\overline{\text{WRL}}$ , $\overline{\text{WRH}}$ , A0, and $\overline{\text{BHE}}$

A0 is never used as an address bit for 16-bit memory. The E-GOLDradio address bit A1 is connected to memory A0.

A0, CSx#, and  $\overline{\text{BHE}}\#$  can be combined using external logic to generate CSL# and CSH# for memories with two chip select inputs or for two 8-bit memories.

It is better to use a 16-bit memory with  $\overline{\text{WRL}}\#$  and  $\overline{\text{WRH}}\#$  inputs. Setting bit **SYSCON.WRCFG** to 1 changes the function of  $\overline{\text{BHE}}\#$  to  $\overline{\text{WRH}}\#$  and  $\overline{\text{WR}}\#$  to  $\overline{\text{WRL}}\#$ . In this case A0 is not required. The write function is summarized in **Table 10-38 Use of  $\overline{\text{WRL}}\#$ ,  $\overline{\text{WRH}}\#$ , A0,  $\overline{\text{BHE}}$  for WRITE (on Page 861)** and the read function in **Table 10-39 Use of  $\overline{\text{WRL}}\#$ ,  $\overline{\text{WRH}}\#$ , A0,  $\overline{\text{BHE}}$  for READ (on Page 861)**. For a byte read the controller reads a word from the memory and internally selects and aligns the required byte.



**CONFIDENTIAL**

**External Bus Unit**

Memories that have the following input combination: CS#, LB# (low byte), UB# (upper byte), OE#, WE# must use the BHE# mode for a correct byte read and write. Refer to [Table 10-38](#) for a write and [Table 10-39](#) for a read.

**Table 10-38 Use of WRL#, WRH#, A0, BHE for WRITE**

<b>SYSCON.WRCFG</b>	<b>A0</b>	<b>BHE#</b>	<b>WR#</b>	<b>Mode</b>
0	0	0	0	Word
0	0	1	0	Low Byte on D(0:7)
0	1	0	0	High Byte on D(8:15)
0	x	x	1	No write Access
<b>SYSCON.WRCFG</b>	<b>A0</b>	<b>WRH#</b>	<b>WRL#</b>	<b>Mode</b>
1	0	0	0	Word
1	0	1	0	Low Byte on D(0:7)
1	1	0	1	High Byte on D(8:15)
1	x	1	1	No write Access

**Table 10-39 Use of WRL#, WRH#, A0, BHE for READ**

<b>SYSCON.WRCFG</b>	<b>A0</b>	<b>BHE#</b>	<b>RD#</b>	<b>Mode</b>
0	0	0	0	Word
0	0	1	0	Low Byte on D(0:7)
0	1	0	0	High Byte on D(8:15)
0	x	x	1	No read Access
<b>SYSCON.WRCFG</b>	<b>A0</b>	<b>WRH#, WRL#</b>	<b>RD#</b>	<b>Mode</b>
1	x	1	0	Word
1	x	1	0	Low Byte on D(0:7)
1	x	1	0	High Byte on D(8:15)
1	x	1	1	No read Access

**CONFIDENTIAL**

**External Bus Unit**

## Bus Mode Performance

**Table 10-40 Bus Mode Versus Performance**

Bus Mode	Transfer Rate (Speed factor for byte/word/dword access)	System Requirements	Free IO Lines
8-bit Demultipl.	Low (1/2/4)	Very low (no latch, byte bus)	D8:D15
16-bit Demultipl.	Very high (1/1/2)	Low (no latch, word bus)	---

## Segment Address Generation

The number of address lines used internally is always 24. The **RP0H.SALSEL** bits are always set to full segment address A23...A16 at Reset.

*Note: The total accessible address space may be increased by accessing several banks that are distinguished by individual chip select lines.*

## $\overline{\text{CS}}$ Signal Generation

CS0, CS1, and CS3 have dedicated output pins with alternate functions for CS1 and CS3. CS2 and CS4 are alternate functions of the ports OE\_n and T\_OUT11.

The  $\overline{\text{CSx}}$  signals identify accesses to different address ranges in the C16x memory map. The number of pins assigned to chip selects can be changed by configuring the pin logic by the SW after RESET.

During external accesses, the EBC can generate a (programmable) number of  $\overline{\text{CS}}$  lines, which make it possible to select external peripherals or memory banks directly without requiring an external decoder. The number of  $\overline{\text{CS}}$  lines is selected during reset in **RP0H.CSSEL** to the maximum of 5 (by default, there are only the three pins connected immediately after a reset).

The  $\overline{\text{CSx}}$  outputs are associated with the **BUSCONx** registers, and they are driven active low for any access within the address area defined for the respective BUSCON register. For any access outside this defined address area, the respective  $\overline{\text{CSx}}$  signal will go inactive high. At the beginning of each external bus cycle, the corresponding valid  $\overline{\text{CS}}$  signal is determined and activated. All other  $\overline{\text{CS}}$  lines are deactivated (driven high) at the same time.

*Note: The  $\overline{\text{CSx}}$  signals is not updated for an access to any internal address area (that is, when no external bus cycle is started), even if this area is covered by the respective **ADDRSELx** register. An internal bus interface access deactivates all external  $\overline{\text{CS}}$  signals.*

*On accesses to address windows without a selected  $\overline{\text{CS}}$  line, all selected  $\overline{\text{CS}}$  lines are deactivated.*

**CONFIDENTIAL**

**External Bus Unit**

The chip-select signals can be operated in four different modes (refer to [Table 10-41](#)) that are selected via bits **CSWENx** and **CSRENx** in the respective **BUSCONx** register.

**Table 10-41 Chip-Select Generation Modes**

CSWENx	CSRENx	Chip-Select Mode
0	0	Address chip select (default after reset)
0	1	Read chip select
1	0	Write chip select
1	1	Read/write chip select

**Read or Write Chip-Select** ( $\overline{CS}$  is renamed  $\overline{WRCS}$  or  $\overline{RDCS}$  in the protocol diagrams) signals remain active only as long as the associated control signal ( $\overline{RD}$  or  $\overline{WR}$ ) is active. This also includes the programmable read/write delay. Read chip select is activated only for read cycles; write chip select is activated only for write cycles; and read/write chip select is activated for both read and write cycles (write cycles are assumed if either of the signals  $\overline{WRH}$  or  $\overline{WRL}$  goes active). These modes save external glue logic when accessing external devices such as latches or drivers that have only a single enable input.

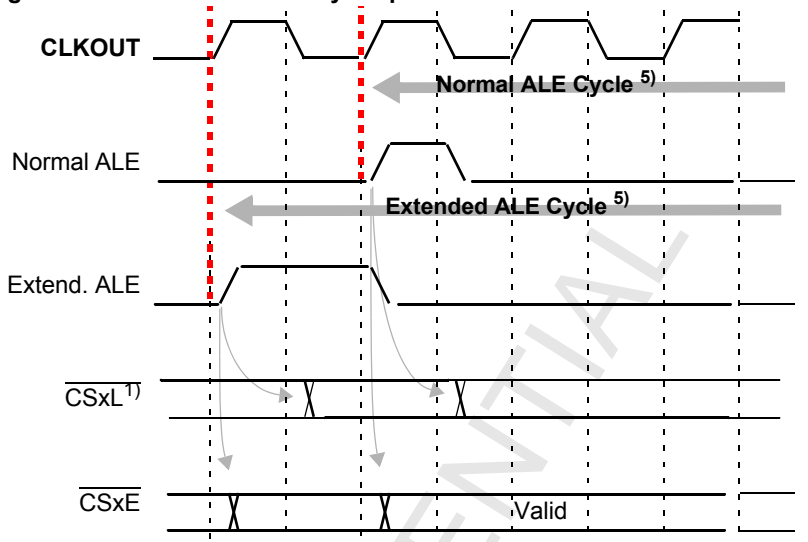
**Address Chip-Select** signals remain active during the complete bus cycle. For address chip select signals, two generation modes can be selected via bit **SYSCON.CSCFG** (see [Figure 10-53 Latched and Early Chip Select \(on Page 864\)](#)):

- A **latched** address chip-select signal ( $\overline{CS}$  is renamed in  $\overline{CSxL}$  in the protocol diagrams) (**CSCFG** = 0) becomes active with the falling edge of ALE and becomes inactive at the beginning of an external bus cycle that accesses a different address window. No spikes are generated on the chip-select lines, and no changes occur as long as locations within the same address window or within internal memory (excluding internal bus interface) are accessed.
- An **early** address chip-select signal ( $\overline{CS}$  is renamed in  $\overline{CSxE}$  in the protocol diagrams) (**CSCFG** = 1) becomes active together with the address and BHE (if enabled) and remains active until the end of the current bus cycle. Early address chip-select signals are not latched internally and may toggle intermediately while the address is changing.

*Note:  $\overline{CS0}$  provides a latched address chip select directly after reset (except for single-chip mode) when the first instruction is fetched internally.*

Internal pull-up devices are used to hold  $\overline{CS0}$  and  $\overline{CS1}$  lines high during reset.

Figure 10-53 Latched and Early Chip Select



<sup>5)</sup> [Section 10.13.4.1 ALE Length Control \(on Page 866\)](#)

## Segment Address Versus Chip Select

There are no limitations in the E-GOLDRadio on using all address lines and all chip selects.

The C166S can address a linear address space of 16 MByte. This allows implementation of a large sequential memory area and access to a great number of external devices using an external decoder. By increasing the number of CS lines, the C166S can access memory banks or peripherals without external glue logic. These two features may be combined to optimize the overall system performance.

## 10.13.4 Programmable Bus Characteristics

Important timing characteristics of the external bus interface have been made user-programmable to adapt it to a wide range of external bus and memory configurations with different types of memories and/or peripherals (see [Figure 10-54](#)).

CONFIDENTIAL

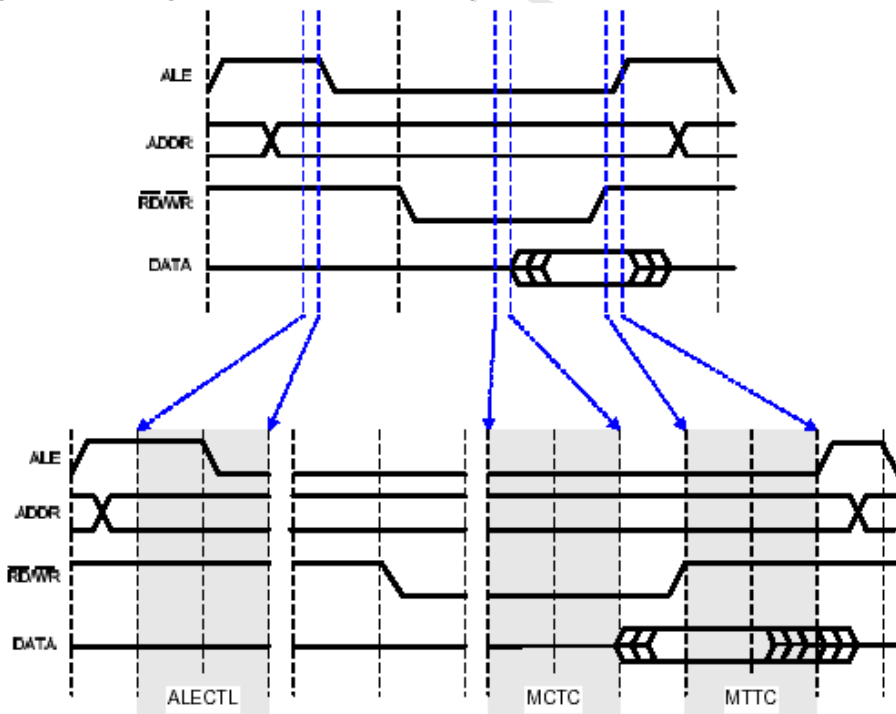
External Bus Unit

The following parameters of an external bus cycle are programmable:

- **ALE Control**  
This defines the internal ALE signal length and the address hold time after its falling edge.
- **Memory Cycle Time** (extendable with 1-15 waitstates)  
This defines the allowable access time.
- **Memory Tri-State Time** (extendable with 1 waitstate)  
This defines the time for a data driver to float.
- **Read/Write Delay Time**  
This defines when a command is activated after the falling edge of ALE.

*Note: External accesses use the slowest possible bus cycle after reset. The bus cycle timing can then be optimized by the initialization software.*

Figure 10-54 Programmable External Bus Cycle



#### 10.13.4.1 ALE Length Control

Even though the ALE signal is not available externally this feature can be used for special timing adjustments. The length of the internal ALE signal and the address hold time after its falling edge are controlled by the **BUSCONx.ALECTLx** bits:

- When **ALECTLx** is set to 0, then the Normal ALE length is 1/4 of a bus cycle at 52 Mhz, and it is 1/2 of a bus cycle at 26 MHz and 78 MHz.
- When **ALECTLx** is set to 1, then the Extended ALE length is 1 bus cycle in all cases.

*Note: **BUSCON0.ALECTL0** is 1 after reset to select the slowest possible bus cycle, the other ALECTLx bits are 0 after reset.*

#### 10.13.4.2 Programmable Memory Cycle Time

The C166S allows the user to adjust the controller external bus cycles to the access time of the respective memory or peripheral. This access time is the total time required to move the data to the destination. It represents the period of time during which the controller signals do not change.

The external bus cycles of the C166S can be extended by introducing wait states during access (see [Figure 10-54 Programmable External Bus Cycle \(on Page 865\)](#)) to compensate for a memory or peripheral that cannot keep pace with the controller maximum speed. During these memory cycle time wait states, the MCU is idle if this access is required for the execution of the current instruction.

The memory cycle time wait states can be programmed in increments of one MCU clock (2 TCLs) within a range from 0 to 15 (default after reset) via the Memory Cycle Time Control **BUSCONx.MCTC** fields. 15-<**MCTC**> wait states are inserted.

#### 10.13.4.3 Programmable Memory Tri-State Time

The C166S allows the user to adjust the time between two subsequent external accesses to account for the tri-state time of the external device. The tristate time defines when the external device has released the bus after deactivation of the read command ( $\overline{RD}$ ).

The output of the next address on the external bus can be delayed by introducing a wait state after the previous bus cycle to compensate for a memory or peripheral that needs more time to switch off its bus drivers (see [Figure 10-54 Programmable External Bus Cycle \(on Page 865\)](#)). During this memory tri-state time wait state the MCU is not idle, so MCU operations will be slowed down only if a subsequent external instruction or data fetch operation is required during the next instruction cycle.

The memory tristate time wait state requires one MCU clock (2 TCLs) and is controlled via is inserted if bit **BUSCONx.MTTCx** is 0 (default after reset).

#### 10.13.4.4 Read/Write Signal Delay

The C166S allows the user to adjust the timing of the read and write commands to account for timing requirements of external peripherals. The read/write delay controls the time between the falling edge of ALE and the falling edge of the command. Without read/write delay, the falling edges of ALE and command(s) are concurrent (except for propagation delays). With the delay enabled, the command(s) become active half a MCU clock (1 TCL) after the falling edge of ALE. The read/write delay does not extend the memory cycle time, and does not slow down the controller.

The read/write delay is controlled via the Read Write Delay Control **BUSCONx.RWDCx** bits. The command(s) will be delayed if bit **RWDCx** is 0 (default after reset).

#### 10.13.4.5 Early $\overline{\text{WR}}$

The duration of an external write access can be shortened by one TCL. The  $\overline{\text{WR}}$  signal is activated (driven low) in the standard way, but can be deactivated (driven high) one TCL earlier than defined in the standard timing. In this case, the data output drivers will also be deactivated one TCL earlier.

This is especially useful in systems that operate on higher MCU clock frequencies and employ external modules (memories, peripherals, etc.) that switch on their own data drivers very rapidly in response to, for example, a chip select signal.

Conflicts between the C166S and external peripheral output drivers can be avoided by selecting early  $\overline{\text{WR}}$  for the C166S.

*Note: Make sure that the reduced  $\overline{\text{WR}}$  low time still meets the requirements of the external peripheral or memory.*

Early  $\overline{\text{WR}}$  deactivation is controlled via the Early Write Enable **BUSCONx.EWENx** bits. The  $\overline{\text{WR}}$  signal is shortened if bit **EWENx** is 1 (default after reset is a standard  $\overline{\text{WR}}$  signal, that is, **EWENx** = 0).

#### 10.13.4.6 $\overline{\text{READY}}$ Controlled Bus Cycles

For situations in which the programmable wait states are not enough, or the response (access) time of a peripheral is not constant, the C166S has external bus cycles that are terminated via an asynchronous  $\overline{\text{READY}}$  input signal. In this case the C166S first inserts a programmable number of wait states (0-7) and then monitors the  $\overline{\text{READY}}$  line to determine the actual end of the current bus cycle. The external device drives  $\overline{\text{READY}}$  low to indicate that either data have been latched (write cycle) or are available (read cycle).

The  $\overline{\text{READY}}$  function is enabled via the Ready Enable **BUSCONx.RDYENx** bit. When this function is selected (**RDYENx** = 1), only the lower 3 bits of the respective **MCTC** bit field define the number of inserted waitstates (0-7), while the MSB of bit field **MCTC** is unused.

CONFIDENTIAL

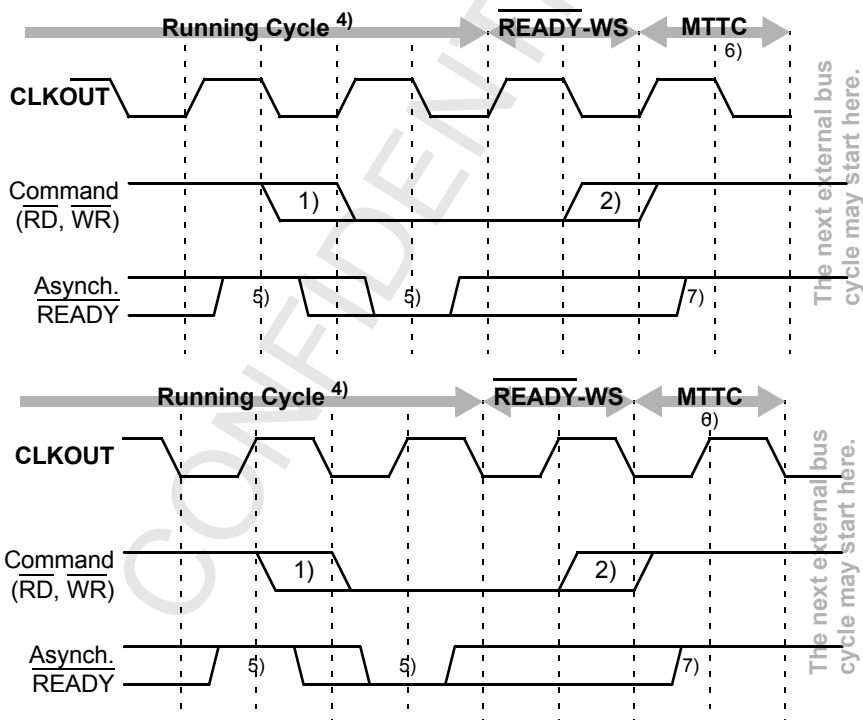
External Bus Unit

As shown in [Figure 10-55](#), the asynchronous  $\overline{\text{READY}}$  requires additional wait states caused by the internal synchronization. The asynchronous  $\overline{\text{READY}}$  is synchronized internally, and programmed wait states may be necessary to provide proper bus cycles (see notes on “normally-ready” peripherals on [Page 869](#)).

An asynchronous  $\overline{\text{READY}}$  signal that has been activated by an external device may be deactivated in response to the trailing (rising) edge of the respective command ( $\text{RD}$  or  $\text{WR}$ ).

*Note: When the  $\overline{\text{READY}}$  function is enabled for a specific address window, each bus cycle within this window must be terminated with an active  $\overline{\text{READY}}$  signal, otherwise, the controller hangs until the next reset. A time-out function is provided by the watchdog timer.*

Figure 10-55  $\overline{\text{READY}}$  Controlled Bus Cycles





**CONFIDENTIAL**

**External Bus Unit**

- 5)  $\overline{\text{READY}}$  sampled HIGH at this sampling point generates a  $\overline{\text{READY}}$  controlled waitstate,  $\overline{\text{READY}}$  sampled LOW at this sampling point terminates the currently running bus cycle.
- 6) For a demultiplexed bus **without MTTC** waitstate the delay here is zero.
- 7) If the next following bus cycle is  $\overline{\text{READY}}$  controlled, an active  $\overline{\text{READY}}$  signal must be disabled before the first valid sample point for the next bus cycle. This sample point depends on the **MTTC** waitstate of the current cycle, and on the **MCTC** waitstates and the ALE mode of the next cycle.

Combining the  $\overline{\text{READY}}$  function with predefined wait states is advantageous in two cases:

1. Memory components with a fixed access time and peripherals operating with  $\overline{\text{READY}}$  can be grouped into the same address window. The (external) waitstate control logic in this case would activate  $\overline{\text{READY}}$  either upon the memory chip select or with the peripheral  $\overline{\text{READY}}$  output. After the predefined number of wait states, the C166S checks its  $\overline{\text{READY}}$  line to determine the end of the bus cycle. For a memory access it is already low; for a peripheral access it may be delayed. As memories tend to be faster than peripherals, there should be no impact on system performance.
2. When using the  $\overline{\text{READY}}$  function with so-called "normally-ready" peripherals, erroneous bus cycles may occur if the  $\overline{\text{READY}}$  line is sampled too early. These peripherals pull their  $\overline{\text{READY}}$  output low while they are idle. When they are accessed, they deactivate  $\overline{\text{READY}}$  until the bus cycle is complete, then drive it low again. If, however, the peripheral deactivates  $\overline{\text{READY}}$  after the first sample point of the C166S, the controller samples an active  $\overline{\text{READY}}$  and terminates the current bus cycle too early. By inserting predefined waitstates, the first  $\overline{\text{READY}}$  sample point can be shifted to an interval in which the peripheral has safely controlled the  $\overline{\text{READY}}$  line.

### 10.13.5 Controlling the External Bus Controller

A set of registers controls the functions of the EBC. General features such as the usage of interface pins ( $\overline{\text{WR}}$ ,  $\overline{\text{BHE}}$ ), segmentation, and internal memory mapping are controlled via register **SYSCON**. The properties of a bus cycle, such as chip-select mode, length of ALE, external bus mode, read/write delay, and waitstates, are controlled via registers **BUSCON4...BUSCON0**. Four of these registers (**BUSCON4...BUSCON1**) have an address select register (**ADDRSEL4...ADDRSEL1**) associated with them, which makes it possible to specify up to four address areas and the individual bus characteristics within these areas. All accesses that are not covered by these four areas are controlled via **BUSCON0**. This allows the use of memory components or peripherals with different interfaces within the same system while optimizing accesses to each of them.

**CONFIDENTIAL**

**External Bus Unit**

### 10.13.5.1 Bus Control Functions in the BUSCON Registers

Registers **BUSCON1..BUSCON4**, which control the selected address windows, are completely under software control. Register **BUSCON0**, which is also used for the very first code access after reset, is partly controlled by hardware, that is, it is initialized via dedicated configuration signals during the reset sequence.

*Note: For reliable operation with certain LCD displays (for example, the EPSON display on the Evaluation Board), set the **BUSCONx.CSWEN** bit to 1.*

**Attention:** The bit descriptions are the same for registers **BUSCON1..BUSCON4** and are in the table below register **BUSCON4**.

#### **BUSCONx**

##### **BUSCON0**

##### **Bus Control Register 0**

Reset value<sup>1)</sup>: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSW EN0	CSR EN0	RES ERV ED	RDY EN0	BSW C0	BUS ACT 0	ALE CTL 0	EW EN0	BTYP		MTT C0	RWD C0	MCTC			

<sup>1)</sup> The reset values of BUS ACT0 and ALE CTL0 may not be 0 as they are controlled by pin MON2 at reset.

##### **BUSCON1**

##### **Bus Control Register 1**

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSW EN1	CSR EN1	RES ERV ED	RDY EN1	BSW C1	BUS ACT 1	ALE CTL 1	EW EN1	BTYP		MTT C1	RWD C1	MCTC			

##### **BUSCON2**

##### **Bus Control Register 2**

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSW EN2	CSR EN2	RES ERV ED	RDY EN2	BSW C2	BUS ACT 2	ALE CTL 2	EW EN2	BTYP		MTT C2	RWD C2	MCTC			

##### **BUSCON3**

##### **Bus Control Register 3**

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**CONFIDENTIAL**

**External Bus Unit**

### BUSCON3

#### Bus Control Register 3

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSW EN3	CSR EN3	RES ERV ED	RDY EN3	BSW C3	BUS ACT 3	ALE CTL 3	EW EN3	BTYP		MTT C3	RWD C3	MCTC			

### BUSCON4

#### Bus Control Register 4

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSW EN4	CSR EN4	RES ERV ED	RDY EN4	BSW C4	BUS ACT 4	ALE CTL 4	EW EN4	BTYP		MTT C4	RWD C4	MCTC			

Field	Bits	Type	Description
<b>MCTC</b>	3:0	rw	<b>Memory Cycle Time Control</b> <sup>1)</sup> (Number of memory cycle time waitstates) 0000: 15 waitstates ... (Number = 15 - <MCTC>) 1111: No waitstates
<b>RWDCx</b>	4	rw	<b>Read/Write Delay Control for BUSCONx</b> 0: With rd/wr delay: activate command 1 TCL after falling edge of ALE 1: No RD/WR delay: activate command with falling edge of ALE
<b>MTTCx</b>	5	rw	<b>Memory Tristate Time Control</b> 0: 1 wait state 1: No wait states
<b>BTYP</b>	7:6	rw	<b>External Bus Configuration</b> 00: 8-bit Demultiplexed Bus 01: Reserved, do not use 10: 16-bit Demultiplexed Bus 11: Reserved, do not use
<b>EWENx</b>	8	rw	<b>Early Write Enable</b> 0: Normal $\overline{WR}$ signal 1: Early write: The $\overline{WR}$ signal is deactivated and write data is tri-stated one TCL earlier

**CONFIDENTIAL**

**External Bus Unit**

Field	Bits	Type	Description
<b>ALECTLx</b>	9	rw	<b>ALE Lengthening Control</b> 0: Normal ALE signal 1: Lengthened ALE signal
<b>BUSACTx</b>	10	rw	<b>Bus Active Control</b> 0: External bus disabled 1: External bus enabled within respective address window ( <b>ADDRSELx</b> )
<b>BSWCx</b>	11	rw	<b>BUSCON Switch Control</b> 0: Address windows switched immediately 1: A tristate waitstate is inserted if the next bus cycle accesses a different window than the one controlled by this <b>BUSCONx</b> register <sup>2)</sup>
<b>RDYENx</b>	12	rw	<b>READY Input Enable</b> 0: External bus cycle is controlled by bit field <b>MCTC</b> only 1: External bus cycle is controlled by the $\overline{\text{READY}}$ input signal
<b>CSRENx</b>	14	rw	<b>Read Chip Select Enable</b> 0: The $\overline{\text{CS}}$ signal is independent of the read command ( $\overline{\text{RD}}$ ) 1: The $\overline{\text{CS}}$ signal is generated for the duration of the read command
<b>CSWENx</b>	15	rw	<b>Write Chip Select Enable</b> 0: The $\overline{\text{CS}}$ signal is independent of the write cmd. ( $\overline{\text{WR}}$ , $\overline{\text{WRL}}$ , $\overline{\text{WRH}}$ ) 1: The $\overline{\text{CS}}$ signal is generated for the duration of the write command
<b>RESERVED</b>	13	r	Reserved for future use; these bits must be left at their reset values.

<sup>1)</sup> When the READY function is selected (**BUSCONx.RDYENx** = 1), only the lower 3 bits of the respective MCTC bit field define the number of inserted wait states (0-7), while the MSB of bit field MCTC is unused

<sup>2)</sup> A BUSCON switch wait state is enabled by bit **BUSCONx.BSWCx** of the address window that is left.

**CONFIDENTIAL**

**External Bus Unit**

## 10.13.5.2 Address Select Registers

### ADDRSELx

#### ADDRSEL1

**Address Select Register 1**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RGSAD												RGSZ			

#### ADDRSEL2

**Address Select Register 2**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RGSAD												RGSZ			

#### ADDRSEL3

**Address Select Register 3**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RGSAD												RGSZ			

#### ADDRSEL4

**Address Select Register 4**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RGSAD												RGSZ			

Field	Bits	Type	Description
<b>RGSZ</b>	0:3	rw	<b>Range Size Selection</b> Defines the size of the address area controlled by the respective <b>BUSCONx/ADDRSELx</b> register pair. Refer to <a href="#">Table 10-42 Address Window Definition (on Page 874)</a> .
<b>RGSAD</b>	15:4	rw	<b>Range Start Address</b> Defines the upper bits of the start address of the respective address area. Refer to <a href="#">Table 10-42</a> .

CONFIDENTIAL

External Bus Unit

*Note: There is no register ADDRSEL0, as register **BUSCON0** controls all external accesses within the C166S address space but outside the four address windows of **BUSCON4...BUSCON1**.*

### Definition of Address Areas

The four register pairs **BUSCON4/ADDRSEL4...BUSCON1/ADDRSEL1** allow four address areas to be defined within the address space of the C166S. Within each of these address areas, external accesses can be controlled by one of the four different bus modes, independent of each other and of the bus mode specified in register **BUSCON0**. Each **ADDRSELx** register has an address window, within which the **BUSCONx** parameters are used to control external accesses. The start address of a **ADDRSELx** window defines the upper address bits, which are not used within the address window, of the specified size (refer to **Table 10-42**). For a given window size, only those upper address bits of the start address (marked "R") that are not implicitly used for addresses inside the window are used. The lower bits of the start address (marked "x") are disregarded.

**Table 10-42 Address Window Definition**

Bit field RGSZ	Resulting Window Size	Relevant Bits (R) of Start Addr. (A12...)
0 0 0 0	4kByte	R R R R R R R R R R R R R R
0 0 0 1	8kByte	R R R R R R R R R R R R R x
0 0 1 0	16kByte	R R R R R R R R R R R R x x
0 0 1 1	32kByte	R R R R R R R R R R R x x x
0 1 0 0	64kByte	R R R R R R R R R x x x x
0 1 0 1	128kByte	R R R R R R R R x x x x x
0 1 1 0	256kByte	R R R R R R R x x x x x x
0 1 1 1	512kByte	R R R R R x x x x x x x
1 0 0 0	1 MByte	R R R R x x x x x x x x
1 0 0 1	2 MByte	R R R x x x x x x x x x
1 0 1 0	4 MByte	R R x x x x x x x x x x
1 0 1 1	8 MByte	R x x x x x x x x x x x
1 1 x x	Reserved.	

### Address Window Arbitration

The address windows that can be defined within the C166S address space may partly overlap each other. Thus small areas may be cut out of bigger windows, for example, to utilize external resources effectively, especially within segment 0.

For each access, the EBC compares the current address with all address-select registers (programmable **ADDRSELx** and hardwired/programmable **XADRSx**). This comparison is done in three levels. The **XADRSx** registers have the highest priority

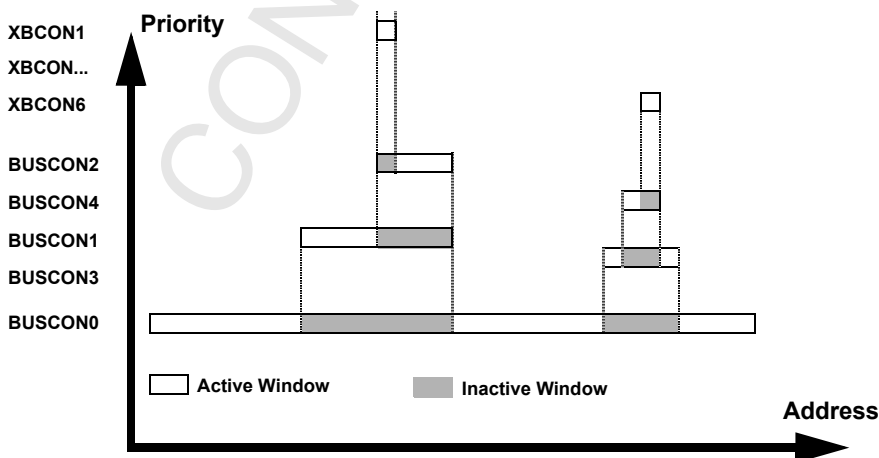
CONFIDENTIAL

External Bus Unit

(priority I). The ADDRSEL registers have the second highest priority (priority II). If there is no match with any **XADRSx** or **ADDRSELx** register, the access to the external bus uses register **BUSCON0** (priority III):

- Priority 1:  
The **XADRSx** registers are evaluated first. A match with one of these registers directs the access to the respective X-Peripheral using the corresponding **XBCONx** register and ignoring all other **ADDRSELx** registers. Priority of the **XADRSx** registers:
  - **XADRS1** (priority I.1)
  - **XADRS2** (I.2)
  - **XADRS3** (I.3)
  - **XADRS4** (I.4)
  - **XADRS5** (I.5)
  - **XADRS6** (I.6).
- Priority 2:  
A match with one of the registers **ADDRSELx** directs the access to the respective external area using the corresponding **BUSCONx** register. Priority of the **ADDRSELx** registers:
  - **ADDRSEL2** (priority II.1)
  - **ADDRSEL4** (II.2)
  - **ADDRSEL1** (II.3)
  - **ADDRSEL3** (II.4).
- Priority 3:  
If there is no match with any **XADRSx** or **ADDRSELx** register, the access to the external bus uses **BUSCON0**.

Figure 10-56 Address Window Arbitration Example



### **Precautions and Hints**

The external bus interface is enabled as long as at least one of the BUSCON registers has its **BUSACT** bit set.

The address windows defined via registers **ADDRSELx** may overlap internal address areas. In this case, internal accesses are executed.

For any access to an internal address area, the EBC remains inactive (refer to [Section 10.13.6 EBC Idle State](#)).

#### **10.13.6 EBC Idle State**

Refer to [Section 6.10 Configuring External Bus and MCU Signals \(on Page 313\)](#).

CONFIDENTIAL



CONFIDENTIAL

## 10.13.7 Page Mode Flash Control Unit

### System Integration:

- Supply domain: VDD\_MAIN
- Chip internal interfaces:
  - Clock domain: Refer to [Section 10.4.1.3 Sub-System Clocks and Enables \(on Page 661\)](#) and see [Figure 10-11 Clock and Enable Generation for MCU Sub-System \(on Page 668\)](#).
  - Interrupt sources:
- Monitor Pins: Refer to [Section 11.8.10 Internal Signal Monitoring \(on Page 1209\)](#)

### 10.13.7.1 Introduction to PMCU

*Note: The Page Mode Switch is not supported in E-GOLDradio V2.1F to V2.2x.*

Flash devices supporting page mode read accesses offer a significant reduction of access time for on-page accesses. When an access is on the same page as the previous access, the reduced on-page access time can be used.

Flash page sizes of 4 or 8 words are available.

In the address of a word:

- The upper bits are the page address
- The lowest 2 or 3 bits is the address on the page.

As long as the page address remains unchanged from the previous access, the reduced time on-page access can be used.

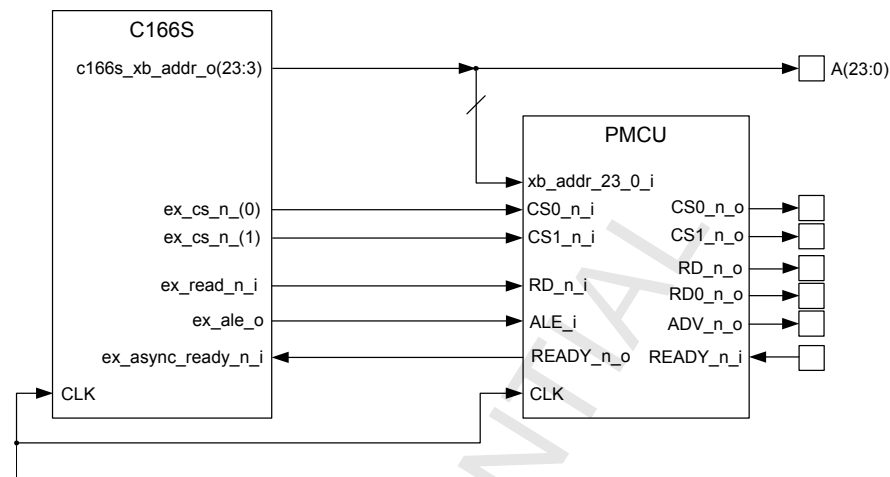
*Note: During write accesses to the Flash device, the Page Mode Control Unit (PMCU) has to be disabled as write accesses are not supported while in the paging mode.*

Depending upon the structure of the executed code, by using page mode access, performance can be increased compared to non-page mode accesses.

Page mode access is controlled by the PMCU shown in [Figure 10-57](#). It uses the asynchronous READY\_n\_i function of the C166S core. It compares the new address coming from the C166S with the previous one and generates an asynchronous READY\_n\_o signal (for the C166S) depending on the access type. The output from the PMCU and from the external READY\_n\_i (an alternate function of the RFSTR2 pad) are combined. If both the PMCU and the external READY\_n\_i are used, READY\_n\_i has to be kept asserted (low) during page mode accesses (PMCU enabled).

CONFIDENTIAL

Figure 10-57 Page Mode Control Unit (PMCU) Block Diagram



### 10.13.7.2 Page Mode Control Registers

Page mode access is available for the address regions controlled by signals `ex_cs_n_(0,1)`, see [Figure 10-57](#). The function of the block is controlled by registers **PMC0** and **PMC1** respectively.

The PMCU is clocked by the controller clock. All access times are defined by wait states defined as multiples of a controller clock cycle.

The PMCU can be configured to match the page mode flash memory specification. The PMCU allows the setting of:

- Page size
- Off-page wait states (off-page read access time)
- On-page wait states (on-page read access time).

The on-page access time is defined by the higher bit field value in either **PMC0/1.ONPWS** or **BUSCONx.MCTC**. If an on-page access is detected by the page mode control logic, signal `READY_n_o` is asserted **ONPWS** clock cycles after signal `RD_n_i` has been asserted.

The off-page access time is defined by the higher bit field value in either **PMC0/1.OFFPWS** or **BUSCONx.MCTC**. If an off-page access is detected, signal `READY_n_o` is asserted **OFFPWS** clock cycles after signal `RD_n_i` has been asserted.

In the page mode signal `RD_n_i` is modified. It is not de-asserted between on-page accesses if signal `ex_cs_n_(0,1)` stays asserted.

**CONFIDENTIAL**

The transition from asynchronous mode to page mode has to be done in such a way that the PMCU is set up and enabled first by setting up bit **PMC0/1.PAEN** enabled while the memory timing is still controlled by bit field **BUSCON0/BUSCON1.MCTC**. After register **PMC0/1** has been set up, bit **BUSCON0/BUSCON1.RDYEN** must be set and the value of wait states in bit field **BUSCON0/BUSCON1.MCTC** must be reduced to 0 (set **MCTC** to  $F_H$ ), therefore effectively enabling page mode.

*Note: **PMC0/1.MEMCS** must be set according to the external memory configuration.*

*Note: The complete page mode enabling procedure must be performed before code again in the page mode from the Flash.*

To terminate the page mode, **MCTC** has to be set to its original value. Then the PMCU can be disabled.

To minimize power consumption, bit **PMC0/1.PAEN** must be kept deasserted if the page mode access control block is not used. If bit **PAEN** is deasserted, the PMCU is transparent to signals **RD\_n\_o** and **READY\_n\_i** when the respective **ex\_cs\_n\_(0,1)** signal is asserted. The PMCU is always transparent if any one of **ex\_cs\_n\_(2,3,4)** is asserted.

The Page Mode must only be enabled if the optimum controller performance is required.

### 10.13.7.3 Page Mode Control Timer Registers

The two timers, **PMC\_TIMER0** and **PMC\_TIMER1**, that control the LOW PULSE WIDTH of **CS0\_n\_o** and **CS1\_n\_o** are available in the PMCU. They are only available when the PMCU is enabled (**PMC0/1.PAEN** = 1).

The counter of the timer starts or re-starts from 1 at each falling edge of **CSx\_n\_i**.

The value of the counter is incremented by 1, at each MCU clock cycle.

The timeout of the timer occurs when the value of the counter is equal to **PMC\_TIMER0/1.TIMER\_MAX**.

When the timeout of the timer occurs, the value of the counter remains equal to **PMC\_TIMER0/1.TIMER\_MAX** as long as the counter is not reset.

A new falling edge on **CSx\_n\_i** resets the counter.

**CONFIDENTIAL**

*Note: When **PMC\_TIMER0/1.EN** or **PMC0/1.PAEN** is set to 0 (disabled):*

- The counter is reset
- CSx\_n\_o equals CSx\_n\_i.

*Note: Set **PMC\_TIMER0/1.EN** to:*

- 1 (enabled) after setting **PMC0/1.PAEN** to 1 (enabled)
- 0 (disabled) before setting **PMC0/1.PAEN** to 0 (disabled).

### **10.13.7.3.1 PMCU Timer Scenarios**

**When **PMC\_TIMER0/1.EN** = 0 (Disabled, Default Value)**

CSx\_n\_o is equal to CSx\_n\_i.

**When **PMC\_TIMER0/1.EN** = 1 (Enabled)**

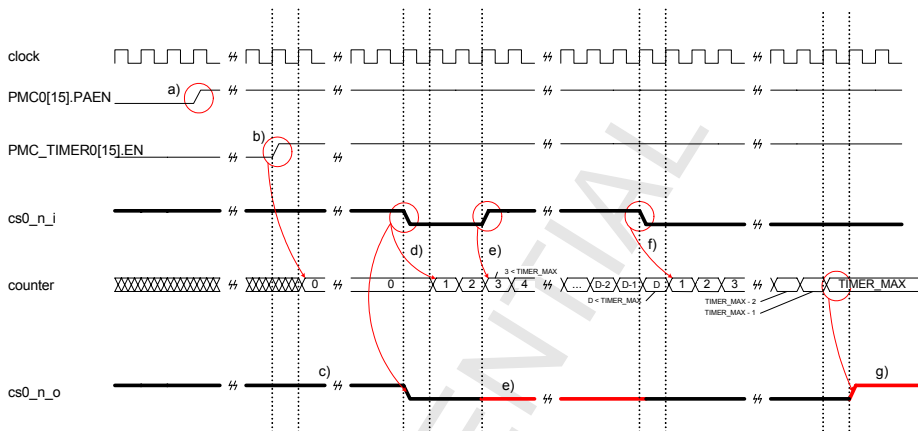
CSx\_n\_o is controlled internally by the PMCU:

1. CSx\_n\_o is equal to CSx\_n\_i as long as a falling edge of CSx\_n\_i is not detected (see the third item (c) in [Figure 10-58](#) and [Figure 10-59](#)).

**CONFIDENTIAL**

2. If CSx\_n\_o is set to 0 (asserted) when a falling edge of CSx\_n\_i is detected, CSx\_n\_o remains asserted as long as the timer has not timed out even if CSx\_n\_i is set to 1 (de-asserted), see **Figure 10-58**.

**Figure 10-58 De-assertion of CSx\_n\_i before Timeout**

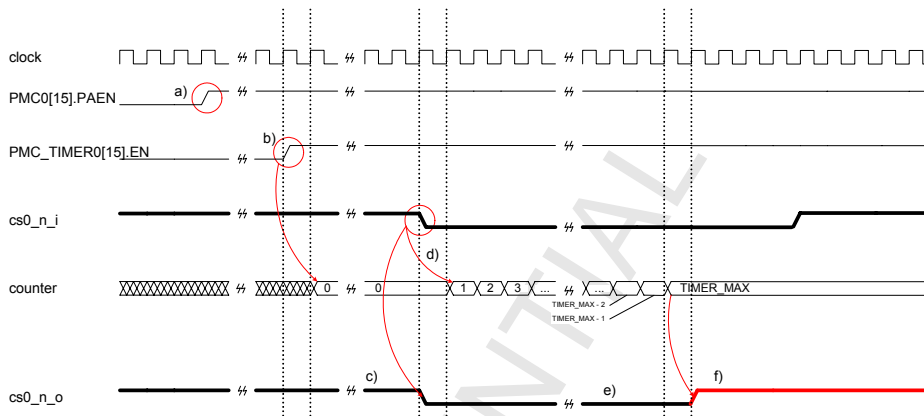


- a) **PMC0/1.PAEN** is set to 1 (enabled).
- b) **PMC\_TIMER0/1.EN** is set to 1 (enabled).
- c) CSx\_n\_o is equal to CSx\_n\_i as long as the falling edge of CSx\_n\_i is not detected.
- d) The counter starts at the next falling edge of CSx\_n\_i and CSx\_n\_o is set to 0 (asserted).
- e) CSx\_n\_o is equal to 0 (asserted) as long as the timeout has not occurred even if CSx\_n\_i is set to 1 (de-asserted).
- f) The counter re-starts from 1 on each falling edge of CSx\_n\_i. CSx\_n\_o remains equal to 0 (asserted).
- g) When the timeout occurs (counter = **PMC\_TIMER0/1.TIMER\_MAX**), CSx\_n\_o is set to 1 (de-asserted) and remains equal to 1 (de-asserted).

**CONFIDENTIAL**

3. CSx\_n\_o is set to 1 (de-asserted) when the timeout of the timer occurs even if CSx\_n\_i is still asserted. Refer to **Figure 10-59**.

**Figure 10-59 Timeout Occurs before CSx\_n\_i Is De-asserted**

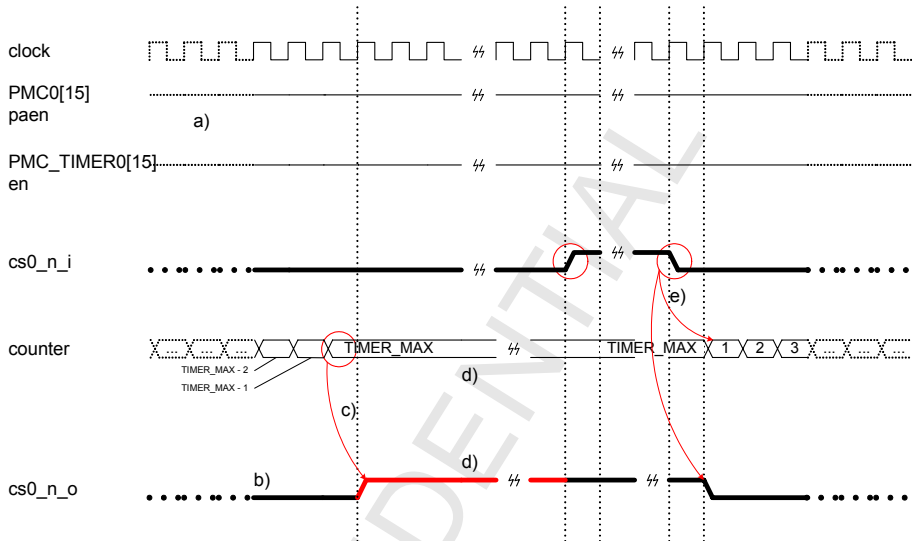


- PMC0[15].PAEN is set to 1 (enabled).
- PMC\_TIMER0[15].EN is set to 1 (enabled).
- CSx\_n\_o is equal to CSx\_n\_i as long as the falling edge of CSx\_n\_i is not detected.
- The counter starts at the next falling edge of CSx\_n\_i.
- CSx\_n\_o is equal to 0 (asserted) as long as the timeout has not occurred.
- When the timeout has occurred (counter = PMC\_TIMER0[15].TIMER\_MAX), CSx\_n\_o is set to 1 (de-asserted), even if CSx\_n\_i is still equal to 0 (asserted).

**CONFIDENTIAL**

4. When a new falling edge of CSx\_n\_i is detected, CSx\_n\_o is set to 0 (asserted). Refer to **Figure 10-60**.

**Figure 10-60 At the Next Falling Edge of CSx\_n\_i, When the Timeout Has Occurred**



For **Figure 10-60**:

- PMC0/1.PAEN and PMC\_TIMER0/1.EN are set to 1 (enabled).
- CSx\_n\_o is equal to 0 (asserted) as long as the timeout has not occurred.
- When the timeout has occurred (counter = PMC\_TIMER0/1.TIMER\_MAX), CSx\_n\_o is set to 1 (de-asserted) even if CSx\_n\_i is still equal to 0 (asserted).
- The counter stays equal to PMC\_TIMER0/1.TIMER\_MAX when the timeout has occurred and CSx\_n\_o stays equal to 1 (de-asserted).
- At the next falling edge of CSx\_n\_i, the counter re-starts from 1 and CSx\_n\_o is set to 0 (asserted).

**CONFIDENTIAL**

### **10.13.7.3.2 When to Use These Timers**

If the timers are used:

- The power consumption of the flash memory devices is reduced.
- The MCU performance is improved when external Flash and RAM is accessed alternately because the page on the Flash device remains open while the RAM is accessed. Subsequent accesses by the MCU are faster due to the reduced page mode access times (refer to [Section 10.13.7.4 Page Mode Switch Details \(on Page 885\)](#)).

CONFIDENTIAL



**CONFIDENTIAL**

### 10.13.7.4 Page Mode Switch Details

*Note: The Page Mode Switch is not supported in E-GOLDradio V2.1F to V2.2x.*

#### To Get the Best Performance

For the best performance, set the Flash in the page mode with following configuration for the appropriate **BUSCON<sub>x</sub> (on Page 870)**,  $x = 0$  or  $1$  depending on the location of the Flash: CS0 or CS1.

A **BUSCON<sub>x</sub>** value of  $14BF_H$  corresponds to the best access performance with 3 cycles (see **Figure 10-61**).

#### BUSCON<sub>x</sub>

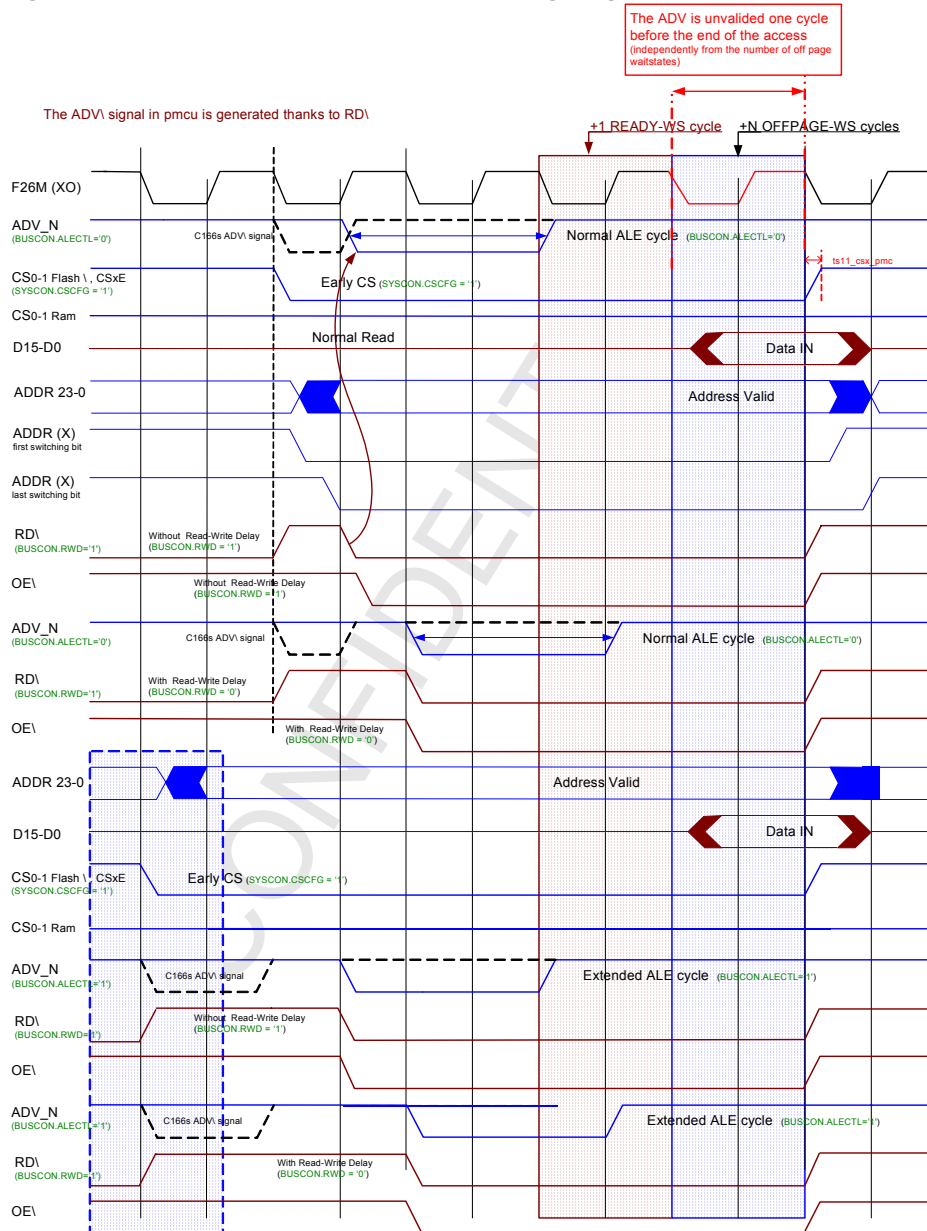
##### Bus Control Register 0 or 1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSW EN 0/1	CSR EN 0/1	RES ERV ED	RDY EN 0/1	BSW C0/1	BUS ACT 0/1	ALE CTL 0/1	EW EN 0/1	BTYP		MTT C0/1	RWD C0/1	MCTC			
0	0	0	1	0	1	0	0	1	0	1	1	1	1	1	1

- **MCTC** => 0 wait states
- **RWDC** => No RD/WR delay: activate command with falling edge of ALE
- **MTTC** => Takes into account bit **MCTC**
- **EWEN** => Normal  $\overline{WR}$  signal (No early write)
- **ALECTL** => Normal ALE signal
- **BSWC** => Address windows switched immediately (no tristate wait state is inserted when the window address is changed)
- **RDYEN** => External bus cycle is controlled by the  $\overline{READY}$  input signal. **RDYEN** must be used to program the number of on-page and off-page wait states, which depends on the Flash specification.  
**RDYEN** requires an additional wait state in the C166S for internal synchronization (refer to the *C166S V1 User's Manual* (May 2002), Section 8.3.6 "READY Controlled Bus Cycles").

**CONFIDENTIAL**

**Figure 10-61 Demultiplexed Read Access Timing Diagram**



**CONFIDENTIAL**

### 10.13.7.5 SFR Registers

#### PMC0/1

##### PMC0

##### Page Mode Control 0

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PAEN	RESERVED					PSIZE	ME MCS	ONPWS	RES ERV ED	OFFPWS			RES ERV ED		

##### PMC1

##### Page Mode Control 1

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PAEN	RESERVED						PSIZE	ME MCS	ONPWS	RES ERV ED	OFFPWS			RES ERV ED	

Field	Bits	Type	Description
OFFPWS	3:1	rw	<b>Off-Page Access Wait States</b> Number of wait states: 000 0 wait states 001 1 wait state 010 2 wait states 011 3 wait states 100 4 wait states 101 5 wait states 110 6 wait states 111 7 wait states
ONPWS	6:5	rw	<b>On-Page Access Wait States</b> Number of wait states: 000 0 wait states 001 1 wait state 010 2 wait states 011 3 wait states
MEMCS <sup>1)</sup>	7	rw	<b>Memory on External Chip Select</b> 0 Any other type of memory on Chip Select x 1 Flash memory on Chip Select x <i>Note: x = 0 for <b>PMC0</b>, 1 for <b>PMC1</b></i>

**CONFIDENTIAL**

Field	Bits	Type	Description
<b>PSIZE</b>	8	rw	<b>Memory Page Size</b> 0    4 words 1    8 words
<b>PAEN</b>	15	rw	<b>Page Mode Enable</b> 0    Disabled 1    Enabled
<b>RESERVED</b>	0, 4, 14:9	r	Reserved for future use; these bits must be left at their reset values.

1) If either:

- **PMC0.MEMCS** = 1 and **PMC1.MEMCS** = 0 or 1
- **PMC0.MEMCS** = 0 and **PMC1.MEMCS** = 0

Then the OE\_n output signal generation depends on the configuration of CS0.

If:

- **PMC0.MEMCS** = 0 and **PMC1.MEMCS** = 1

Then the OE\_n output signal generation depends on the configuration of CS1.

*Note: The PMCU does not permit use of 0 wait states because the controller adds 1 wait state in the ready control mode.*

*Note: The PMCU is only intended to be used for 16-bit wide, non-multiplexed, read accesses from external 16-bit wide devices.*

### 10.13.7.5.1 Page Mode Timer Control Registers

#### PMC\_TIMER0/1

##### PMC\_TIMER0

##### PMC Timer 0

**Reset value: 0000<sub>H</sub>**

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>EN</b>	<b>TIMER_MAX</b>															

Field	Bits	Type	Description
<b>TIMER_MAX</b>	14:0	rw	<b>Timer 0 Duration</b> Its unit is one cycle of the PD Bus clock. The timer starts or re-starts on each falling edge of CS0_n.
<b>EN</b>	15	rw	<b>Page Mode Timer 0 Enable</b> 0    Disables and resets the timer to 0 1    Enables the timer (starts from 0)

**CONFIDENTIAL**

## PMC\_TIMER1

### PMC Timer 1

**Reset value: 0000<sub>H</sub>**

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>EN</b>	<b>TIMER_MAX</b>															

Field	Bits	Type	Description
<b>TIMER_MAX</b>	14:0	rw	<b>Timer 1 Duration</b> Its unit is one cycle of the PD Bus clock. The timer starts or re-starts on each falling edge of CS01_n.
<b>EN</b>	15	rw	<b>Page Mode Timer 1 Enable</b> 0     Disables and resets the timer to 0. 1     Enables the timer (starts from 0).

#### 10.13.7.5.2 Page Mode Control Logic

**Figure 10-62** and **Figure 10-63** show timing examples with an initial access, an on-page access, and an off-page access.

*Note: The timing diagram in **Figure 10-63** is based on the performance to be achieved for E-GOLDradio V3.0 .*

*Note: To view these figures with the PDF reader, rotate counterclockwise, and then zoom as needed.*

Due to a propagation delay lower in pagemode than in normal mode, the data arrives after this ADV latching edge. That's why we add an additional waitstate

C166s data latching edge

ADV-2

$3 * 19.2ns = 57.6ns$  (52MHz)

The timing for ADV generation is defined in the Characterization. In pagemode the ADVI generators is much longer than in Normal Mode. Then we need 5 cycles of Poise access instead of 4 cycles in Normal Mode.

worst case off page Flash INTEL 28F32018 access time

The timing for Data propagation from Pad to C166s Pins

Off Page 5 cycles Read access

Demultiplexed off page read access in pagemode with BUSCON=14Bh and PMCO=8084h memory: INTEL 28F32018

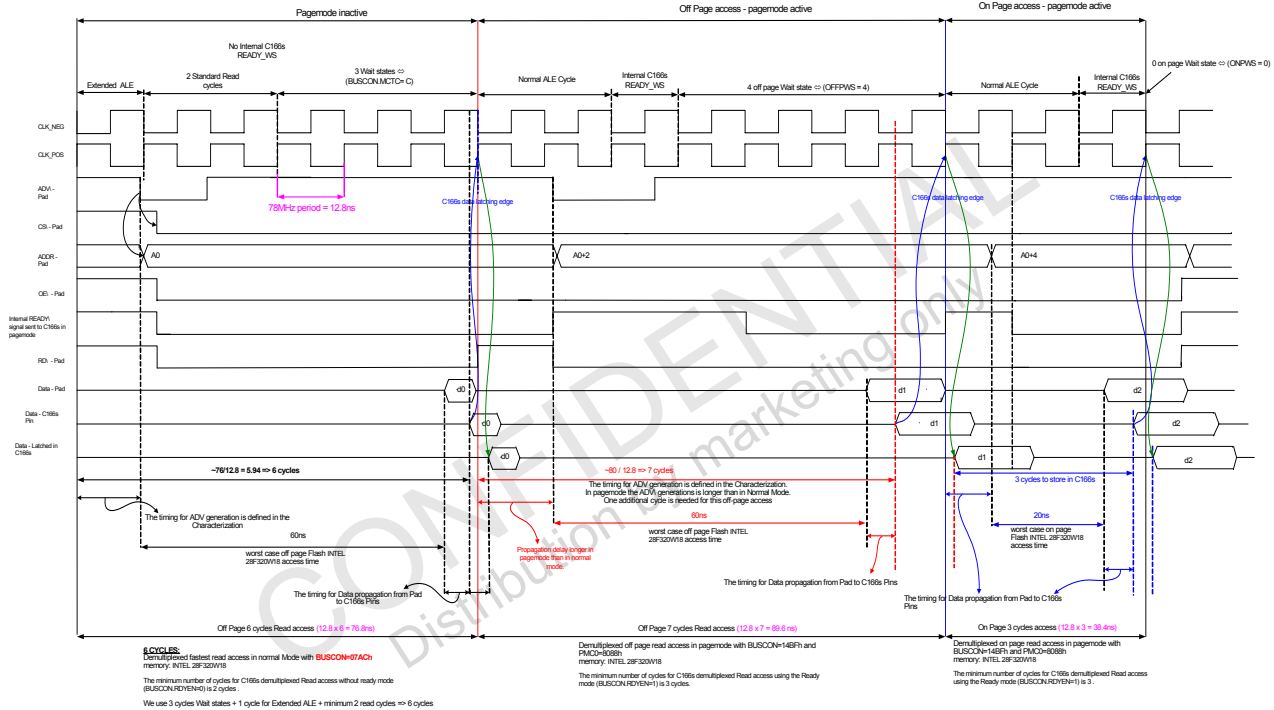
The minimum number of cycles for C166s demultiplexed Read access using the Ready mode (BUSCON RDYEN=1) is 3 cycles

Demultiplexed and PMCO= memory: INTEL 28F32018

The minimum Ready mode



Figure 10-63 Example Read Access at 78MHz



ADV\_N and OE\_N are also used for Flash devices on CS3 and CS4. For Flash devices on CS3 and CS4, page mode access is not supported.

**CONFIDENTIAL**

CONFIDENTIAL



**CONFIDENTIAL**

### 10.13.8 EBU Application Notes

History	
Design Spec.	Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDRadio Design Specification, Revision 1.03

#### 10.13.8.1 Computation of Wait States and Examples of Read Accesses

To be able to configure the External Bus Unit correctly the characteristics of the device connected to the bus interface and the specification of EGR must be analyzed.

Due to the fact that different devices have different access times, the number of wait states needed to access these devices must be carefully chosen. To speed up access to consecutive memory locations in these devices the Page Mode can be configured. Before explaining the computation of Wait States for the normal mode and the Page Mode, some more details are given about the PMCU and ADV<sub>1</sub> signal.

#### 10.13.8.2 Page Mode for Flash Memory

The Page Mode is used to speed up access to consecutive memory location in external memories (read only). The page size can be 4 or 8 depending on the memory used. The first of the 4 (8) page addresses will be read in the OFF Page Mode where all address bit are used on the address bus. During that access the higher address bits (bit 2(3) to bit 23) will be stored in the PMCU. The following address locations on the same page will then be read in the ON Page Mode which is a much faster way accessing the data where only the 2 (3) lower bit are modified on the address bus. An the ON Page Mode uses an internal signal (falling edge) generated in the PMCU that is sent to the CPU, which generates the next address of the page.

#### Asynchronous mode

The EBU bus is controlled through the Page Mode Controller Unit (PMCU). This PMCU is transparent if the Page Mode is deactivated and signals are passed through. The ALE signal of the C166 will become the ADV<sub>1</sub> signal which is delayed by the propagation delay but will have the same length. More information about ADV<sub>1</sub> signal can be found in [Section 10.13.8.2.2](#).

Only asynchronous and de-multiplexed modes are supported.

The 24 address bits are divided into page address bits (MSBs) and 2 or 3 address bits (LSBs) in the page.

**CONFIDENTIAL**

The Page Mode controller stores the page address bits. These page address bits will not be modified during on page accesses but only the 2 or 3 on page address bits (LSB). The PMCU generates at the first access on the page the ADV $\backslash$  signal (low active) which indicates to the Flash that these bits are valid and can be stored. This ADV $\backslash$  signal is reset and will stay inactive during further on page accesses. The RD\_n\_o signal is active at the first page access and will stay active during the next page accesses.

Depending on the device, the CS and OE signals are set active at the first access on the Flash independent off the Page Mode or not and stay active during the whole access time or CS can toggle after each access (depending on configuration).

Furthermore the PMCU creates a Ready $\backslash$  signal for the C166. This Ready $\backslash$  signal will become active at the end of the each read cycle (rising edge of CLK\_POS signal) to latch the data into the C166. On page access uses an internal signal (falling edge of READY $\backslash$ ) generated in the PMCU that is sent to the CPU, which generates the next address on the page. The synchronization of the Ready signal in the C166 needs an extra wait state, which is added automatically to the off page wait states.

On CS0 or CS1 only read access is supported in the Page Mode, for write accesses the Page Mode has to be inactive. No write access is possible due to the fact that the write pin is shared by all external memory devices.

#### **10.13.8.2.1 Transition from Asynchronous Mode to Page Mode**

Before switching to the Page Mode, the ALE (ADV $\backslash$ ) signal in the **BUSCON0** (on [Page 870](#)) and **BUSCON1** registers and the wait states in the bit field **BUSCON0/BUSCON1.MCTC** are programmed accordingly to the type of memory used.

##### **Transition from Asynchronous Mode to Page Mode**

This has to be done in such a way that the PMCU is set up and enabled first. For Flash which uses the ADV $\backslash$  signal, the ALE has to be programmed as Extended ALE before switching to the Page Mode:

1. Enable bits **PMC0/1.PAEN** while the memory timing is still controlled by bit field **BUSCON0/BUSCON1.MCTC**.
2. Set up registers **PMC0/1** accordingly to the external memory, program ON and OFF page wait states.
3. Set ALE (ADV $\backslash$ ) signal to Normal ALE signal in **BUSCON0/BUSCON1.ALECTL**.
4. Set bit **BUSCON0/BUSCON1.RDYEN** for READY control.
5. The value of wait states in bit field **BUSCON0/BUSCON1.MCTC** must be reduced to 0 (set **MCTC** to FH), therefore, effectively enabling the Page Mode.

##### **Transition from Page Mode to Asynchronous Mode**

To terminate the Page Mode, **MCTC** has to be set to its original value. Then the PMCU can be disabled:

**CONFIDENTIAL**

1. Put **BUSCON0/BUSCON1** to original values.
2. Disable bits **PMC0/1.PAEN**.

**10.13.8.2.2 Explanations on ADV\ Signal**

ALE is an internal signal generated directly by the CPU. This signal is inverted and propagated through the logic of EBU/PMCU to generate the ADV signal, which is accessible on the pad.

In normal mode the ADV\ signal is directly derived from the ALE signal of the C166 and arrives on the ADV\ pad with a certain propagation delay. The ADV\ signal will be used to store the address in the memory. This allows the flash to latch the address bits internally. (This is valid only for memories which use ADV\ signal)

The length of the ADV signal depends on the characteristics of the memory and can be configured in the **BUSCONx.ALECTLx** bits to define the lengths.

For certain types of memories the timing between CS and ADV has to be respected. CS and ADV signal have to be active at the same time for a certain duration. This can be read in the dedicated specifications of the device.

**Non-Page Mode**

If the Page Mode is deactivated, the length of ADV signal is half or a full clock cycle dependent on the **ALECTLx** bit. This is valid for 78 MHz and 26 MHz but for 52 MHz the duty cycle is 25% so the length of normal ADV signal is 25% of the clock cycle.

**Page Mode**

In the Page Mode the ADV\ signal is derived from the ALE and RD\ signal in the PMCU. The generation of the ADV signal is longer than in normal mode and specified in the EGR timing specifications. This means off-page access is longer than normal mode access.

If the Page Mode is active, the number of wait states needed to access the flash will be added to the specified access time. The rising edge of ALE generates with the propagation delay the falling edge of ADV\ and the signal is maintained for almost 1cycle + number of wait states cycles active.

In the Page Mode one extra WS is added by the PMCU if ready control mode (**BUSCONx**) was chosen or extended ALE was configured (**ALECTLx** bit).

**10.13.8.2.3 ADV\ Available Only for CS0 and CS1 in Page Mode**

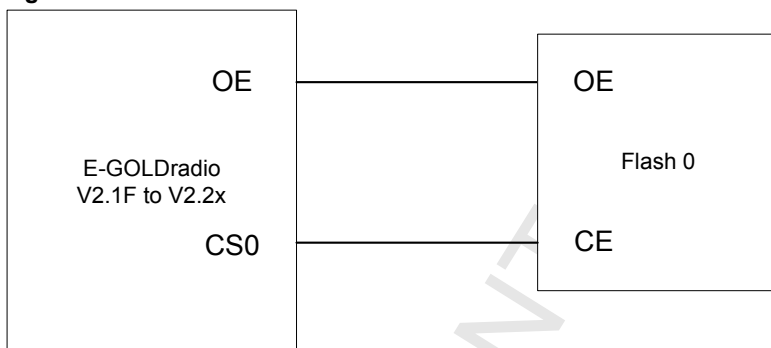
For E-GOLDradio V2.1F to V2.2x ADV\ signal is generated only for CS0 and CS1 in the PMCU. This means that Flash devices which need ADV\ input signal and are not connected to CS0 or CS1 will not be served when the Page Mode is active. If these devices which are connected to CS2 to CS4 need ADV\ input signal, the Page Mode must be disabled before each access.

**CONFIDENTIAL**

### 10.13.8.2.4 Alternate Function of OE/CS2 Pin

Due to the fact that OE pin can be used either for OE or as CS2 some precaution must be taken if the Page Mode is activated. Normally OE is connected to OE of the Flash 0 (see [Figure 10-64](#)).

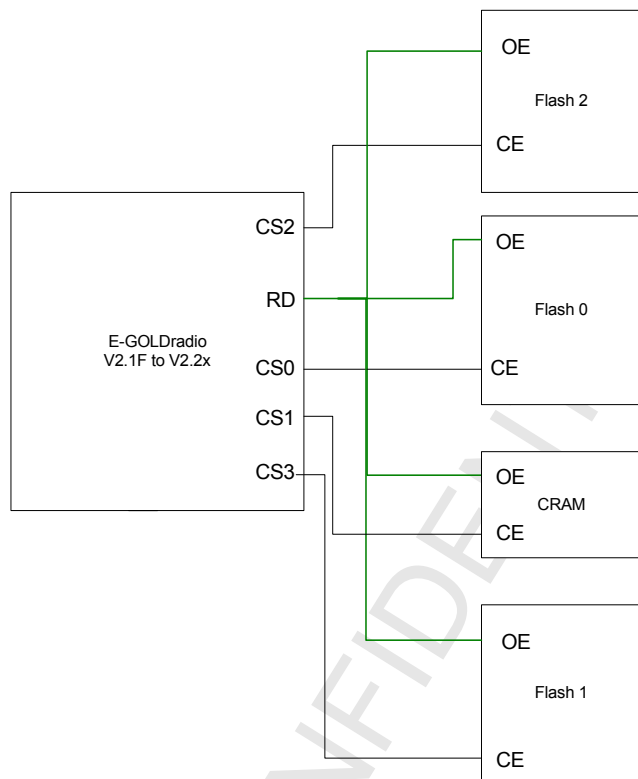
**Figure 10-64 Normal Connection of Flash to EGR**



If the OE pin is used as CS2 the Flash0/1 OE pin cannot be connected anymore to this pin but must be connected to RD signal. The CS2 will be used for Flash 2 CE pin. All OE pins of other devices connected must use the RD signal for their OE pins.

**CONFIDENTIAL**

**Figure 10-65 Connection of Multiple Devices Using CS2**



**CONFIDENTIAL**

### 10.13.8.2.5 Computation of the Number of Wait States

Due to the fact that manufacturers of memories defines the access time of their memories in different ways the calculation of wait states can cause problems. For the calculation of wait states not only the access time of the memory but some EGR specific parameters have to be taken into account.

Use the following formula to calculate the number of wait states:

$$NbWS = \text{ROUNDUP}\left(\frac{TS + t_{acc}}{t_c}\right) - T_{base} \quad [36.12]$$

Where:

- $t_{acc}$  is the access time of the memory
- $T_{base} = 2$  for a random access with a normal ALE and 3 for all other cases
- $t_c = \frac{1}{f_{CPU}}$
- TS: refer to [Calculation of TS](#).

#### Calculation of TS

The parameter TS is composed of the propagation delay of the signal from the internal signal to the pad (T) and the setup time for the data from the pad to the internal C166 (see [Figure 10-66](#) for more details).

The total time needed for accessing a device and getting stable data in the C166 is:

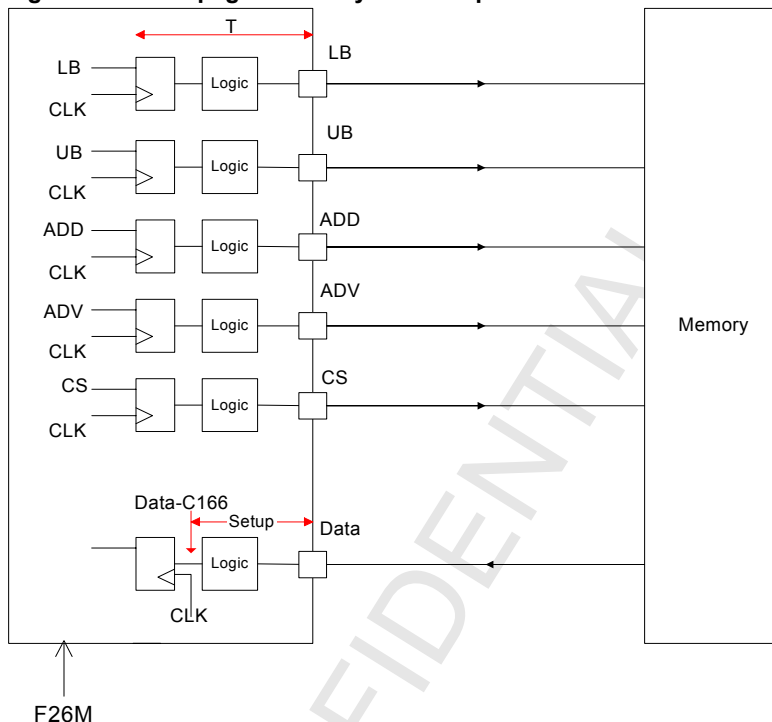
$$T_{total} = T + \text{setup time} + t_{acc} \quad [36.13]$$

where is:

$$TS = T + \text{setup time}$$

CONFIDENTIAL

Figure 10-66 Propagation Delay and Setup Time



Firstly the parameter in the memory specifications must be chosen which gives the most pessimistic access time (e.g: "ADV low to Output Delay"). Then the parameter TS is composed of the propagation delay (T) of this signal ("ADV ") and the setup time for the data. To the access time of the memory the corresponding TS must be added. This means if the ADV\ signal defines the access time the propagation delay of ADV and data setup time must be added. If CS is the defined signal the propagation delay of CS and data setup time must be added.

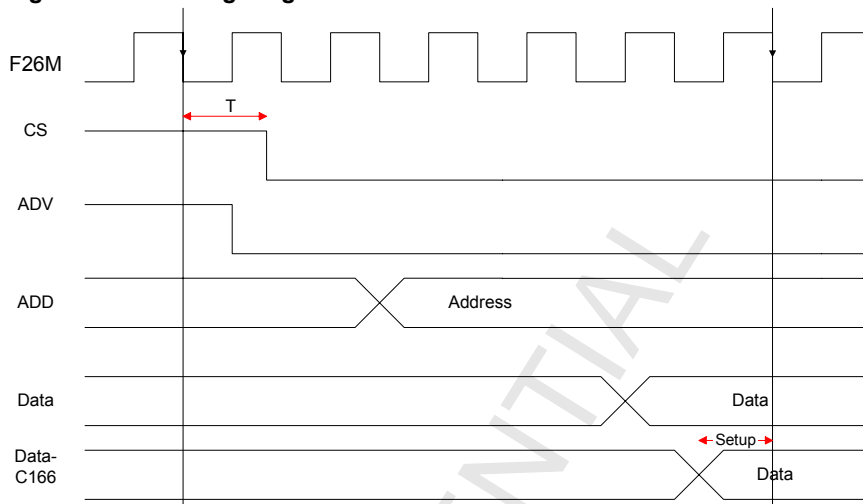
The start of the access time to the external memory is:

- The last falling edge between ADV/ or CS/ for random access or off page access in the Page Mode.
- Address change for the on page access.

In many memory specifications the access time defined by ADV, CS and Address are identical. In the shown case ([Figure 10-67](#)) the propagation delay of the address has to be added because it is the worst case of propagation delay.

CONFIDENTIAL

Figure 10-67 Timing Diagram



To decide which TS must be chosen in the [Equation \[36.13\]](#) for a device the following points have to be respected:

- TS =  $ts_{23}$  (**Address plus data set-up (on Page 1322)**)
  - if address specifies the access time for random access and for an off-page access
  - in the ON Page Mode (Flash without ADV signal)
  - if ADV, CS and address access times are identically.
- TS =  $ts_{21}$  (**Early CS plus data set-up (on Page 1322)**)
  - for a random access and for an OFF page access in the Page Mode where CS defines the access time
- TS =  $ts_{20}$  (**ADV plus data set-up (on Page 1322)**)
  - for an off-page access in the Page Mode if ADV\ defines the access time.

### 10.13.8.3 Examples of Read Accesses

All the following examples are for demultiplexed memory access in asynchronous mode.

In the following 4 examples the `cgu_c166_clk_pos1` duty cycle is 25%-75% for 52 MHz. For 26 MHz and 78 MHz it is 50%-50%.



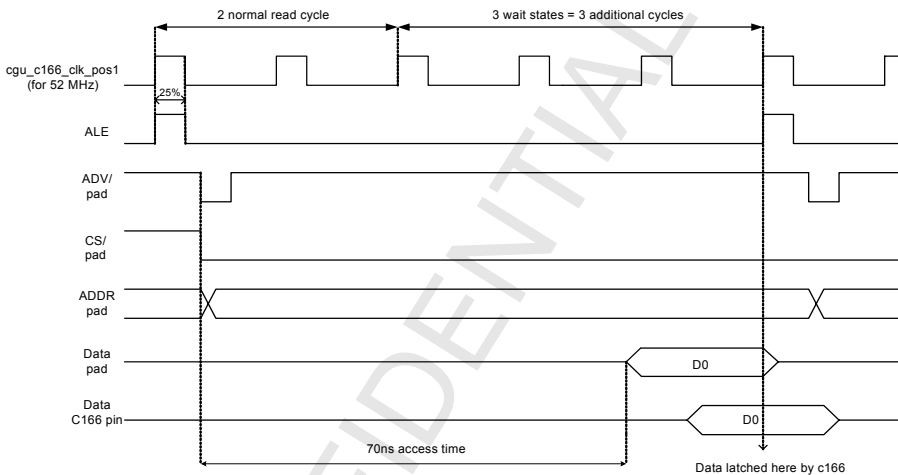
**CONFIDENTIAL**

### 10.13.8.3.1 Read Access in Non-Page Mode

#### Example 1

he following example shows random Read access with normal ALE.  $T_{base} = 2$  for a random access because of 2 normal read cycles. The wait states must be calculated using  $ts_{23}$  because all access times of address, CS and ADV are identically.

**Figure 10-68 Random Read Access for External Flash with Normal ALE**

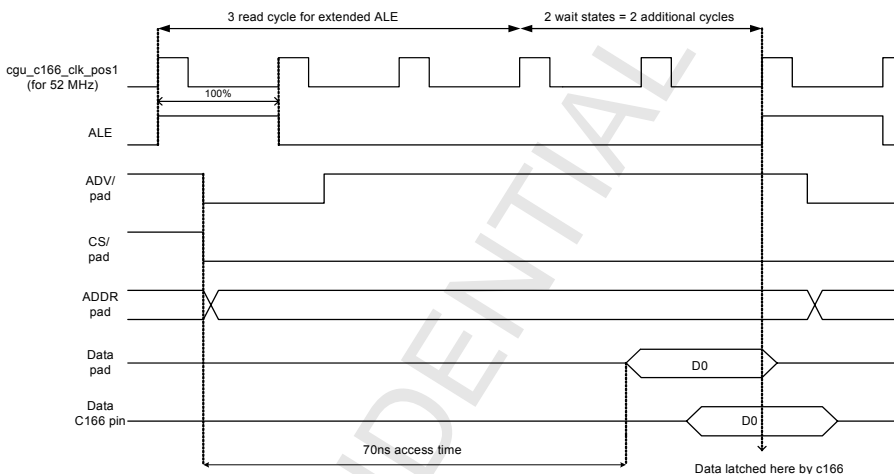


## CONFIDENTIAL

### Example 2

The following example shows random Read access with extended ALE.  $T_{base} = 3$  for a random access because of 2 normal read cycles plus 1 WS due to extended ALE. The wait states must be calculated using  $ts_{23}$  because all access times of address, CS and ADV are identically.

**Figure 10-69 Random Read Access for External Flash with Extended ALE**



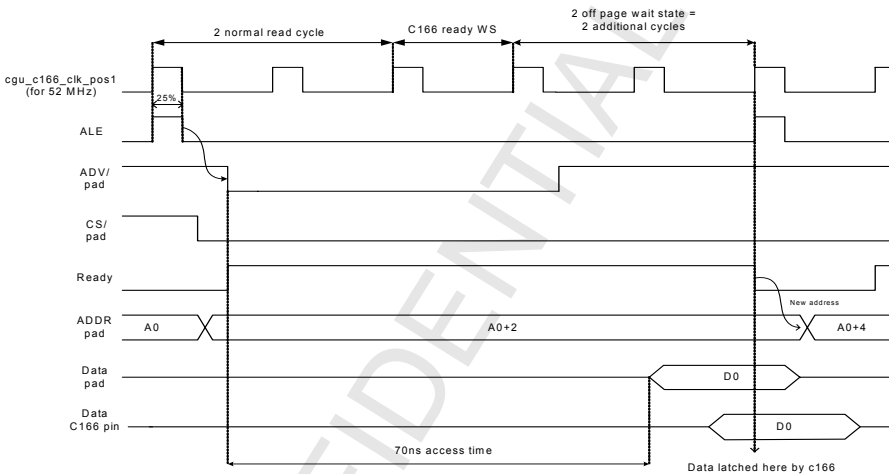
**CONFIDENTIAL**

### 10.13.8.3.2 Read Access in Page Mode

#### Example 3

In the Page Mode ADV\ signal is delayed in the PMCU.  $T_{base} = 3$  because of 2 normal read cycles plus C166 Ready WS (automatically added). Off page read is defined in this example by the ADV\ access time so  $ts_{20}$  must be used.

**Figure 10-70 Page Mode Enable - OFF page Read Access**

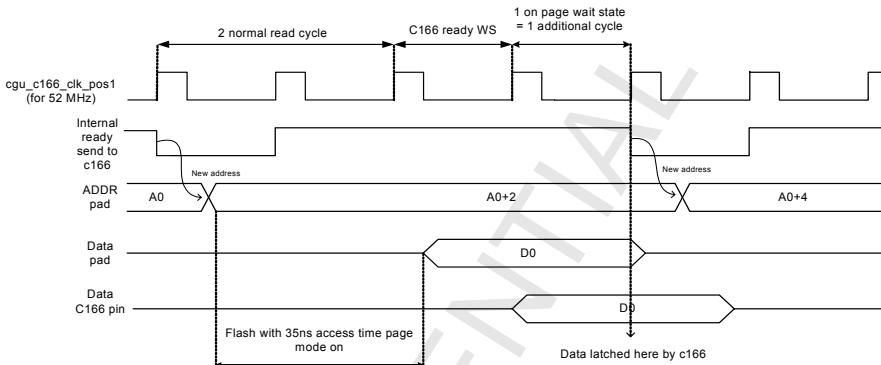


## CONFIDENTIAL

### Example 4

In the ON Page Mode  $T_{base} = 3$  because of 2 normal read cycles plus C166 Ready WS (automatically added). On page read is defined by the address access time so  $ts_{23}$  must be used in [Equation \[36.12\]](#).

**Figure 10-71 Page Mode Enable - ON page Read Access**



An ON page access uses an internal signal (falling edge) generated in the PMCU that is sent to the CPU, which generates the next address of the page.

### 10.13.8.4 Supported Memory Devices

Some examples of device which have been connected:

1. Intel RD38F2240WWYTQW18 memory combo:  
2 x 64Mbits Wireless Flash (top 8kbit parameter sector for boot): In the asynchronous Page Mode 65/25 ns access time 32 Mbits PSRAM (Pseudo-Static RAM). 1.8 V voltage core and I/O.
2. Combo AMD/ Fujitsu
3. Cellular RAM + Flash
4. Disk On Chip support MD5832-d256-V3Q18.

In [Figure 10-72](#) some characteristics of tested memory devices are shown. For these devices wait states were calculated in using [Equation \[36.12\]](#) and described in the following tables.

CONFIDENTIAL

Figure 10-72 Examples of Memory Specifications

Manufacturer	Type	Memory size	Asynchronous Mode				Latching Specifications					Page mode	Remarks
			Read Cycle time (min)	Address to Output delay time (max)	CE low to Output delay time (max)	UB, LB access time						Page Size Word	Remarks
							On Page	Access time Off Page					
							Page address access time	CE\ low to ADV\ high (min)	ADV low to Output Delay (max)	ADV\ Pulse width Low (min)	Address to output delay (no ADV)		
		MBit	ns	ns	ns	ns	ns	ns	ns	ns	ns		
INTEL	38F2240WWYTQW18 PSRAM spec	32	85	85	85 *)	85	n.a.	n.a.	n.a.	n.a.	n.a.		Combo: PSRAM spec
INTEL	28F640W18 **)	64	70	70	70	n.a.	20	10	70	10	n.a.	4	FLASH spec (in Combo)
INTEL	28F128W18	128	85	85	85	n.a.	25	10	85	10	n.a.	4	FLASH
Infineon	HYE18P16161AC-70	16	70	70	70 *)	70	n.a.	n.a.	n.a.	n.a.	n.a.		PSRAM
AMD	Am29LV640MH-101	64	100	100	100	n.a.	30	n.a.	n.a.	n.a.	100	4	Flash: No ADV pin

\*) CS to output delay

\*\*) specification of Feb. 2004

## CONFIDENTIAL

For the memory devices described in **Figure 10-72** the number of wait states can be calculated as shown in the following figures. The EGR parameters are given and must be chosen for the specified **Equation [36.12]**.

As shown in **Figure 10-73** for the Flash in the OFF Page Mode, the parameter ts20 must be taken because ADV\ signal defines the access time of the memory. In the ON Page Mode the address specifies the access time so parameter ts23 has to be taken. As in asynchronous mode (Flash with the Page Mode disabled) all specified so parameters of the memory are identically the ts23 parameter of EGR must be chosen because it is the worst case. The cellular RAM (CRAM) is specified by the address access time so ts23 is the parameter to use. the Tbase values are also given.

**Figure 10-73 Number of WS for INTEL Combo**

INTEL 38F2240WYYTQW18		Flash Off page	Flash On page	Flash with page mode disabled	CRAM
Tacc (ns)		70	20	70	85
fCPU (MHz)	26	0	0	0	1
fCPU (MHz)	52	3	0	3	4
fCPU (MHz)	78	6	1	5	7
EGR parameters to choose for INTEL 38F2240WYYTQW18		ts20	ts23	ts23	ts23
tsX (ns)		35	30	30	30
Tbase		3	3	3	2

Number of waitstates

**Figure 10-74** shows that for the Flash in the OFF Page Mode. The parameter ts20 must be used because ADV\ signal defines the access time of the memory.

In the ON Page Mode the address specifies the access time so the parameter ts23 has to be used.

All the specified memory parameters are identical to the asynchronous mode (Flash with the Page Mode disabled) so the ts23 parameter of EGR must be used because it is the worst case.

**Figure 10-74 Number of WS for INTEL Flash**

INTEL 38F128W18		Flash Off page	Flash On page	Flash with page mode disabled
Tacc (ns)		85	25	85
fCPU (MHz)	26	1	0	0
fCPU (MHz)	52	4	0	3
fCPU (MHz)	78	7	2	6
EGR parameters to choose for INTEL 38F128W18		ts20	ts23	ts23
tsX (ns)		35	30	30
Tbase		3	3	3

Number of waitstates

In **Figure 10-75** the AMD Flash does not use ADV\ signal.

In the OFF Page Mode the parameter ts23 must be used because the address defines the access time of this memory.

In the ON Page Mode the address specifies the access time so parameter ts23 has to be used.

## CONFIDENTIAL

All the specified memory parameters are identical to the asynchronous mode (Flash with the Page Mode disabled) so the ts23 parameter of EGR must be used because it is the worst case.

**Figure 10-75 Number of WS for AMD Flash**

AMD Am29LV640MH-101		Flash Off page	Flash On page	Flash with page mode disabled
Tacc (ns)		100	30	100
fCPU (MHz)	26	1	0	1
fCPU (MHz)	52	4	1	4
fCPU (MHz)	78	8	2	8
EGR parameters to choose for AMD Am29LV640MH-101		ts23	ts23	ts23
tsX (ns)		30	30	30
Tbase		3	3	3

Number of waitstates

**Figure 10-76** shows that the cellular RAM (CRAM) is specified by the address access time so ts23 is used. The Tbase values are also given.

**Figure 10-76 Number of WS for Infineon CRAM**

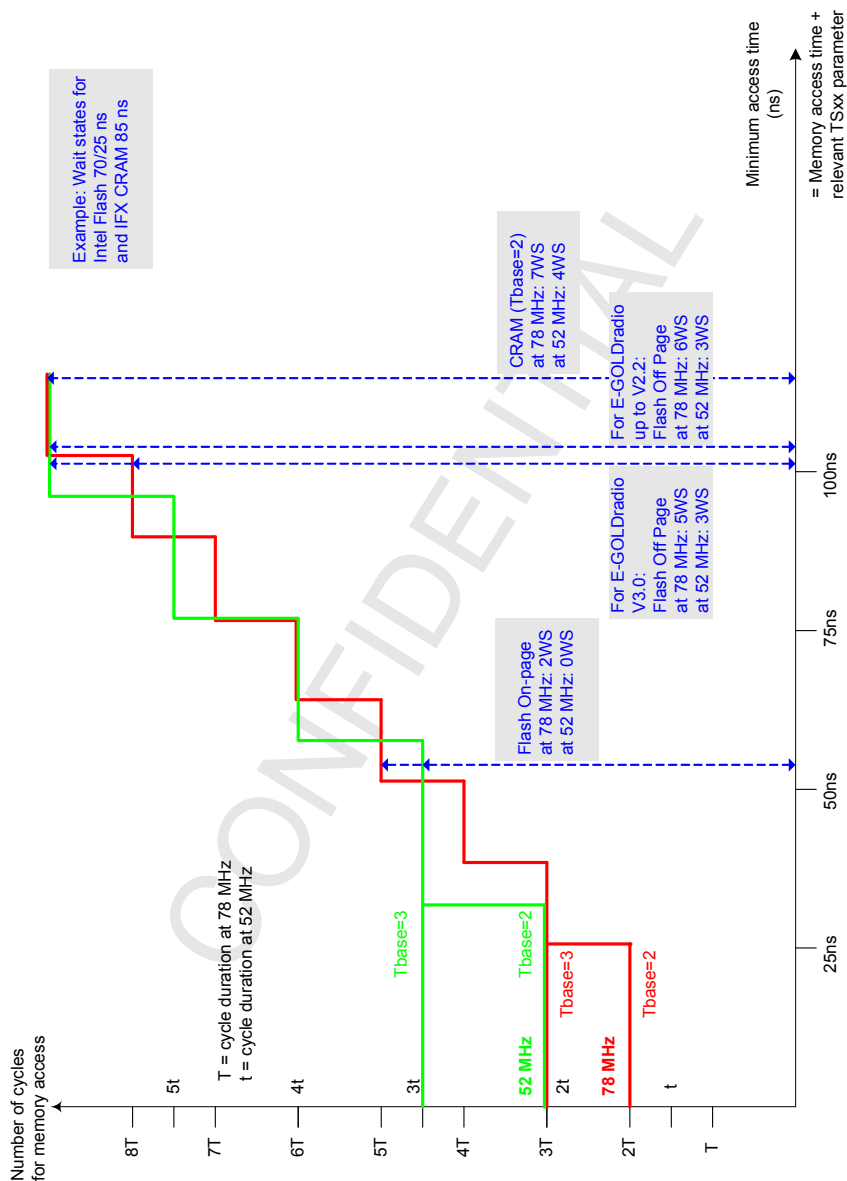
Infineon HYE18P16161AC-70		CRAM
Tacc (ns)		70
fCPU (MHz)	26	1
fCPU (MHz)	52	4
fCPU (MHz)	78	6
EGR parameters to choose for Infineon HYE18P16161AC-70		ts23
tsX (ns)		30
Tbase		2

Number of waitstates

**Figure 10-77** is a graphical view of memory access time vs. the number of cycles needed for a memory access.

**CONFIDENTIAL**

**Figure 10-77 Wait State Computation and Performance**





## CONFIDENTIAL

### 10.13.8.5 Definition of Address Areas

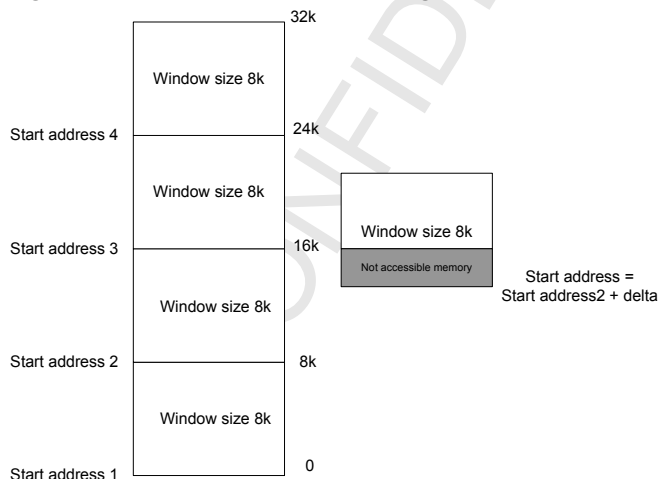
As already describes in previous chapter the four register pairs **BUSCON4/ADDRSEL4...BUSCON1/ADDRSEL1** allow four address areas to be defined within the address space of the C166S. Within each of these address areas, external accesses can be controlled by one of the four different bus modes, independent of each other and of the bus mode specified in register **BUSCON0**. Each **ADDRSELx** register has an address window, within which the **BUSCONx** parameters are used to control external accesses. The start address of a **ADDRSELx** window defines the upper address bits, which are not used within the address window, of the specified size. For a given window size, only those upper address bits (A12, A13,...) of the start address (marked "R") that are not implicitly used for addresses inside the window are used. The lower bits of the start address (marked "x") are disregarded.

This means that start address of a **ADDRSELx** window (Bit Field **RGSAD**) must be a multiple of the window size (Bit Field **RGSZ**) and cannot start at an arbitrary address. Example: Bit Field **RGSZ** = 0001 allows a window size of 8kbytes

The start address must be chosen:  $n \cdot 8\text{kbytes}$  with  $n=0,1,2,\dots$

If it is mapped on an arbitrary address the memory addresses below the start address of a multiple of the window size cannot be accessed (**Figure 10-78**).

**Figure 10-78 Address Areas Mapping**



### 10.13.8.6 Address Window Arbitration

Memory mapping can be done in a way that same address windows are used for different types of internal (SFR) or external memories. In that case the EBC takes care of prioritization of the windows. Internal memories defined in **XADRSx** have higher

## CONFIDENTIAL

priority than **ADDRSELx**. This means when an external memory and an internal memory are placed at the same address priority is given to the internal memory and **XADRSx** is evaluated. The EBC will write to internal memory ignoring **ADDRSELx**. If no window is defined in **XADRSx** for this address **ADDRSELx** will be used. If neither **XADRSx** nor **ADDRSELx** is mapped to this address **BUSCON0** will be used and external memory will be accessed.

**XADRS1** has highest priority followed by **XADRS2** until **XADRS6**.

For the priority of the **ADDRSELx** registers the priority is:

1. **ADDRSEL2** highest
2. **ADDRSEL4**
3. **ADDRSEL1**
4. **ADDRSEL3** lowest.

### 10.13.8.7 EBU Power Domain

All signals (including alternate functions) are connected to the same power supply VDDP\_EBU so no special care must be taken on this point.

### 10.13.8.8 Performances

#### 10.13.8.8.1 Dhrystone 2.1

The Dhrystone 2.1 algorithm is an enhancement of the Dhrystone 1.1:

The Dhrystone 1.1 test is a standard test (was first published in 1984) intended to measure and compare CPU performance. This short benchmark program, that contains mainly string operation, memory accesses, logic evaluation... should reflect the MCU activities in most of the SW applications. Loading this test in internal and external memory can be used to show performance gain using the Page Mode.

The Dhrystone result is determined by measuring the average time a processor takes to perform many iterations of a single loop containing a fixed sequence of instructions that make up the benchmark. When Dhrystone is referenced, it is usually quoted as 'DMIPS' or 'Dhrystone MIPS/MHz' (refer to *Benchmarking in context: Dhrystone* by Richard York, Arm Ltd.).

Dhrystone running in internal RAM:

**CONFIDENTIAL**

**Table 10-43** shows the best configuration. The dhrystone test is executed when the program is running in internal RAM with 0 wait states for access to write the internal memory.

**Table 10-43 Result of Dhrystone 2.1 for E-GOLDradio V1.0 with program running in internal RAM with 0 WS**

CPU Clock	26 MHz	52 MHz	78 MHz
Runtime RTC (ms)	6858	3456	2304
Dhrystone/sec	14581.5	28935.2	43402.8
DMIPS	8.299	16.469	24.703

**Table 10-44** shows the same test with the program running in internal memory and WS is used for data write.

**Table 10-44 Result of Dhrystone 2.1 for E-GOLDradio V1.0 with program running in internal RAM with 1 WS**

CPU clock	26 MHz	52 MHz	78 MHz
Runtime RTC (ms)	9238	4218	2812
Dhrystone/sec	10824.9	23707.9	3556.1
DMIPS	6.161	13.493	20.240

**Table 10-45:** The external RAM and the external flash used for this test have the same access time (70 ns), the total access time for number of wait states calculation is  $70 \text{ ns} + \text{TS23} = 100 \text{ ns}$ . The configuration of the external bus controller for CS0 and CS1 are: **BUSCON1** = 04B9<sub>H</sub> at 78 MHz, 04BB<sub>H</sub> at 52 MHz, and 04BE<sub>H</sub> at 26 MHz. **BUSCON0** = 06BA<sub>H</sub> at 78 MHz, 06BC<sub>H</sub> at 52 MHz, and 06BF<sub>H</sub> at 26 MHz.

**Table 10-45 Result of Dhrystone 2.1 for E-GOLDradio V1.0 with program running in external RAM or in external flash without the Page Mode with 0 WS in internal RAM**

CPU clock	26 MHz	52 MHz	78 MHz
Runtime RTC (ms)	18927	18629	16493
Dhrystone/sec	5283.5	5368	6063.2
DMIPS	3.007	3.055	3.451

**Table 10-46:** The external RAM and the external flash used for this test have the same access time (70 ns), the total access time for number of wait states calculation is  $70 \text{ ns} + \text{TS23} = 100 \text{ ns}$ . The configuration of the external bus controller for CS0 and 1 are:

**CONFIDENTIAL**

**BUSCON1** = 04B9<sub>H</sub> at 78MHz, 04BB<sub>H</sub> at 52 MHz, and 04BE<sub>H</sub> at 26 MHz. **BUSCON0** = 06BA<sub>H</sub> at 78 MHz, 06BC<sub>H</sub> at 52 MHz, and 06BF<sub>H</sub> at 26 MHz.

**Table 10-46 Result of Dhrystone 2.1 for E-GOLDradio V1.0 with program running in external RAM or in external flash without the Page Mode with 1 WS in internal RAM**

CPU clock	26 MHz	52 MHz	78 MHz
Runtime RTC (ms)	18927	19452	17042
Dhrystone/sec	5283.5	5140.9	5867.9
DMIPS	3.007	2.926	3.340

**Table 10-47:** Enabling the Page Mode at 78 MHz gives a performance improvement of:

- 33%
- 11% compared to the Page Mode at 52 MHz.

The external flash used has 70 ns access time. That means 70 ns + TS23(30 ns) must be used to calculate the number of wait states. The external bus configuration is

**BUSCON0** = 14BF<sub>H</sub> and **PMCO** = 80AA<sub>H</sub> at 78 MHz, 8086<sub>H</sub> at 52M Hz, and 8080<sub>H</sub> at 26 MHz.

**Table 10-47 Result of Dhrystone 2.1 for E-GOLDradio V1.0 with program running in external flash with the Page Mode enable with 0 WS in internal RAM**

CPU clock	26 MHz	52 MHz	78 MHz
Runtime RTC (ms)	18927	12435	10973
Dhrystone/sec	5283.5	8041.8	9113
DMIPS	3.007	4.577	5.187

**CONFIDENTIAL**

**Table 10-48:** Enabling the Page Mode at 78 MHz gives a performance improvement of:

- 35%
- 15% compared to the Page Mode at 52 MHz.

The external flash used has 70 ns access time. That means 70 ns + TS23(30 ns) must be used to calculate the number of wait states. The external bus configuration is BUSCON0 = 14BF<sub>H</sub> and PMC0 = 80AA<sub>H</sub> at 78 MHz, 8086<sub>H</sub> at 52 MHz, and 8080<sub>H</sub> at 26 MHz.

**Table 10-48 Result of Dhrystone 2.1 for E-GOLDradio V1.0 with program running in external flash with the Page Mode enable with 1 WS in internal RAM**

CPU clock	26 MHz	52 MHz	78 MHz
Runtime RTC (ms)	18927	13013	11055
Dhrystone/sec	5283.5	7684.6	9045
DMIPS	3.007	4.374	5.148

### 10.13.8.9 Idle Mode

For power saving reasons during MCU Idle Mode it is possible to release the external bus. Bit **EBU\_PDC.PDW** is used to release external bus by driving pull down.

Following procedure is recommended:

1. Set **EBU\_PDC.PDW** = 1 to release EBU during MCU IDLE Mode
2. Reset **EBU\_PDC.PDW** when MCU exits IDLE Mode to reactivate EBU pads

No other registers of the EBU have to be modified, the same configuration can be used after leaving the Idle Mode.

**CONFIDENTIAL**

CONFIDENTIAL

CONFIDENTIAL

Logic Arranger

## 10.14 Logic Arranger

History	
Current version: Rev. 1.06, 2005-11-04	
Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.01	
<b>Page 915</b>	Updated <b>System Integration</b> : WS00006682
Changes for Rev. 1.02	
<b>Page 917</b>	Updated <b>Section 10.14.2 Functional Description</b> WS00007114
Changes for Rev. 1.04	
<b>Page 920</b>	<b>LPASEL.IN7_SEL</b> updated WS00008455
Changes for Rev. 1.06	

### System Integration:

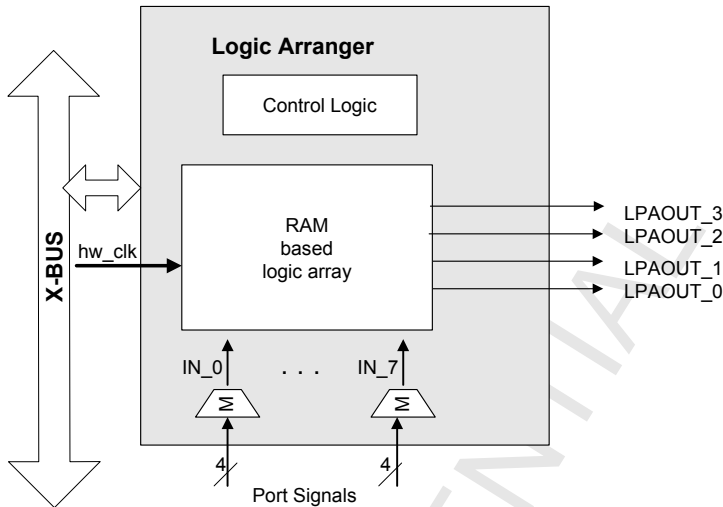
- Supply domain: VDD\_MAIN
- Chip internal interfaces:
  - Clock domain: Refer to **Section 10.4.1.3 Sub-System Clocks and Enables (on Page 661)** and see **Figure 10-10 Clock Enable (on Page 662)**.
  - Bus domain: X-Bus
- Interrupt sources:
- Monitor Pins: Refer to **Section 11.8.10 Internal Signal Monitoring (on Page 1209)**

### 10.14.1 Introduction

The Logic Port Signal Arranger (LPSA) allows selecting different port signals like GSM Timer outputs, CAPCOM outputs, GPIOs, etc. and to freely define programmable logic combination of those signals. The resulting output signals can be provided to external pins.

The LPSA is RAM based, which means that any logical combination between input and output signals are programmed in a RAM. During normal operation the value representing up to 8 input signals is taken as the address for this RAM and the corresponding pre-programmed values are transferred to 4 outputs on the next clock cycle. **LPA\_OUT(3:0)** are connected to the external pins of the baseband chip (see **Figure 10-79**).

Figure 10-79 Block Diagram of Port Signal Logic Arranger



## hw\_clk

Refer to the Clock Domain in [System Integration: \(on Page 915\)](#).

## System Integration

The LPSA is located on the X-Bus. Refer to [Table 12-4 Address Mapping of X-Bus Peripherals \(on Page 1285\)](#) for information on the X-Bus.

### 10.14.2 Functional Description

The LPSA unit is a single port RAM of 64\*16 bits which is programmable via the X-Bus. The RAM entries represent the desired logical combinations of the input signals IN\_0 to IN\_7. For each input IN\_0 .. 7 one out of 4 port signals can be selected by setting **LPASEL**.

**Table 10-49** is an example for mapping of two first words 0 and 1 of the LPSA RAM to the numeric values represented by the logic state of IN\_0...7 and to the LPA\_OUT0...3 signals. The LPSA RAM is write/readable. The RAM access is word oriented. The read option is mainly used for test purposes. The output signal level at LPA\_OUT0...3 toggles while reading the LPSA RAM content.



**Table 10-49 LPSA RAM Words 0 and 1**

	RAM Content							
	Word 0				Word 1			
LPA_OUT3	bit[3]	bit[7]	bit[11]	bit[15]	bit[3]	bit[7]	bit[11]	bit[15]
LPA_OUT2	bit[2]	bit[6]	bit[10]	bit[14]	bit[2]	bit[6]	bit[10]	bit[14]
LPA_OUT1	bit[1]	bit[5]	bit[9]	bit[13]	bit[1]	bit[5]	bit[9]	bit[13]
LPA_OUT0	bit[0]	bit[4]	bit[8]	bit[12]	bit[0]	bit[4]	bit[8]	bit[12]
RAM byte address	0		1		2		3	
IN_0 .. 7 value	0	1	2	3	4	5	6	7

For the Port Signal Logic Arranger RAM address area refer to [Table 12-4 Address Mapping of X-Bus Peripherals \(on Page 1285\)](#).

**Table 10-50** shows an example: If IN\_0 .. 7 represent the value of 5 (IN\_0 and IN\_2 = 1), the outputs LPA\_OUT 1 and 2 should be set to 1. In this case a value of 0060<sub>H</sub> has to be programmed in word 1 in the RAM address space.

**Table 10-50 LPSA RAM Example**

	RAM Content							
	Word 0				Word 1			
LPA_OUT3	0	0	0	0	0	0	0	0
LPA_OUT2	0	0	0	0	0	1	0	0
LPA_OUT1	0	0	0	0	0	1	0	0
LPA_OUT0	0	0	0	0	0	0	0	0
IN_0 .. 7 value	-	-	-	-	-	X	-	-

The output signals are continuously updated with each X-Bus clock cycle if the **LPACON.RAMCLKEN** bit is set to 1. The output values at LPA\_OUT 0..3 corresponding to any new state of IN 0..7 are available at the outputs latest after one X-Bus clock cycle.

If the RAM clock is switched off by setting **RAMCLKEN**, the last value should remain unchanged at the outputs LPA\_OUT(0:3). During write access to the RAM, the RAM clock is disabled and the outputs LPA\_OUT(0:3), therefore, remain unchanged.

**CONFIDENTIAL**

**Logic Arranger**

## 10.14.3 LPSA Registers

### 10.14.3.1 LPSA Identification Register

**LPSAID**

**LPSA Identification Register**

**Reset value: A101<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>Module_ID</b>								<b>Revision_Number</b>							

Field	Bits	Type	Description
<b>Revision_Number</b>	0:7	r	<b>LPSA Revision Number</b> These hard-wired bits are used for module revision numbering.
<b>Module_ID</b>	8:15	r	<b>LPSA Identification Number</b> These hard-wired bits are used for module identification numbering.

### 10.14.3.2 LPSA Control Register

**LPAON** stores the control information concerning clock and output enable.

**LPAON**

**LPSA Control Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>								<b>LA OUT 3EN</b>	<b>LA OUT 2EN</b>	<b>LA OUT 1EN</b>	<b>LA OUT 0EN</b>	<b>RESERVED</b>		<b>RAM CLK EN</b>	

Field	Bits	Type	Description
<b>RAMCLKEN</b>	0	rw	<b>Enables Clock for Arranger RAM</b> 0 Disabled; last value kept valid at LPAOUT0..3 1 Enabled; LPAOUT0..3 are updated with each clock cycle
<b>LAOUTxEN</b> (x = 0 to 3)	7:4	rw	<b>Enables Output LPAOUT x</b> 0 Disabled; LPAOUT x = 0 1 Enabled; LPAOUT x = programmed value
<b>RESERVED</b>	3:1, 15:8	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**Logic Arranger**

### 10.14.3.3 LPSA Select Register

**LPASEL** stores the control information for the multiplexers to select the input signals for IN 0..7.

**LPASEL**

**LPSA Select Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>IN7_SEL</b>	<b>IN6_SEL</b>	<b>IN5_SEL</b>	<b>IN4_SEL</b>	<b>IN3_SEL</b>	<b>IN2_SEL</b>	<b>IN1_SEL</b>	<b>IN0_SEL</b>								
Field	Bits	Type	Description												
<b>IN0_SEL</b>	0:1	rw	<b>Port Signal Selector for IN0</b> 00 GPIO48 01 TOUT0 10 CC00IO 11 DSPOUT0												
<b>IN1_SEL</b>	2:3	rw	<b>Port Signal Selector for IN1</b> 00 GPIO47 01 TOUT1 10 CC01IO 11 DSPOUT1												
<b>IN2_SEL</b>	4:5	rw	<b>Port Signal Selector for IN2</b> 00 GPIO46 01 TOUT9 10 CC22IO 11 MON1												
<b>IN3_SEL</b>	6:7	rw	<b>Port Signal Selector for IN3</b> 00 GPIO44 01 TOUT10 10 CC23IO 11 MON2												
<b>IN4_SEL</b>	8:9	rw	<b>Port Signal Selector for IN4</b> 00 GPIO50 (input) 01 Reserved 10 Reserved 11 NOT RTCOUT												

**CONFIDENTIAL**

**Logic Arranger**

Field	Bits	Type	Description
<b>IN5_SEL</b>	10:11	rw	<b>Port Signal Selector for IN5</b> 00 GPIO49 (input) 01 Reserved 10 Reserved 11 VCXO_EN
<b>IN6_SEL</b>	12:13	rw	<b>Port Signal Selector for IN6</b> 00 Reserved 01 RXON (from GSM Timer Unit) 10 Reserved 11 RSTOUT_Q
<b>IN7_SEL</b>	14:15	rw	<b>Port Signal Selector for IN7</b> 00 Reserved 01 TXON from GSM Timer Unit 10 Reserved 11 Reserved

**CONFIDENTIAL**

**GPRS Cipher Unit**

## 10.15 GPRS Cipher Unit

History	
Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.01	
<b>Page 921</b>	Updated <b>System Integration</b> : WS00006682
Changes for Rev. 1.02	
<b>Page 933</b> , <b>Page 935</b> , <b>Page 939</b> , <b>Page 941</b>	Updated <b>Interrupts Section 10.15.4</b> , <b>Section 10.15.5.2</b> , <b>Section 10.15.6.2</b> , <b>Section 10.15.6.4</b> WS00006682
Changes for Rev. 1.05	

### System Integration:

- Supply domain: VDD\_MAIN
- Chip internal interfaces:
  - Clock domain: Refer to **Section 10.4.1.3 Sub-System Clocks and Enables (on Page 661)** and see **Figure 10-10 Clock Enable (on Page 662)**.
  - Bus Domain: X-Bus
- Interrupt sources: gprs\_int\_o
- Chip external signals related to this block: none
- Monitor Pins: refer to **Section 11.8.10 Internal Signal Monitoring (on Page 1209)**

**Note: The GEA3 implementation is based on the May 2002 ETSI release.**

*Note: The GPRS block is a sub-block of the GPRS\_GEA3.*

### 10.15.1 GPRS\_GEA3 Overview

**Note:** The GPRS\_GEA3 block includes the GEA1, GEA2, and GEA3 algorithms.

All mentioned cycles refer to clock cycles of the GPRS\_GEA3 unit, not MCU cycles or wait states, that is, if the clock of the GPRS\_GEA3 unit is 52 MHz, one cycle corresponds to 19.2 ns. GPRS\_GEA3 unit is directly coupled to the X-Bus clock; their frequencies are always identical.

The GPRS\_GEA3 block is used for fast data communication according to the GPRS standard. The GPRS\_GEA3 unit basic structure is shown in the following block diagram. The GPRS block is used for data encoding (transmission) and decoding (reception). Data coding includes error detection (CRC) and data scrambling (Ciphering Unit) to insure data privacy. The GPRS data stream is block-oriented with a variable block length. The length step size is 1 Byte. The GPRS block consists of an LLC Header (2-37 Bytes), Information Field (0-1520 Bytes), and FCS Field (3 Bytes).

GPRS\_GEA3 is composed of two sub-blocks:

- GPRS\_KERNEL that provides GEA1/2 functionality
- GPRS\_GEA3\_KERNEL provides GEA3 functionality.

Prior any use, the block to be used has to be selected.

### 10.15.2 Functional Overview

**Figure 10-80** shows the GPRS\_GEA3 block and **Figure 10-81** is an example of one of the two sub-blocks.

**Figure 10-80 GPRS\_GEA3 Unit Block Diagram**

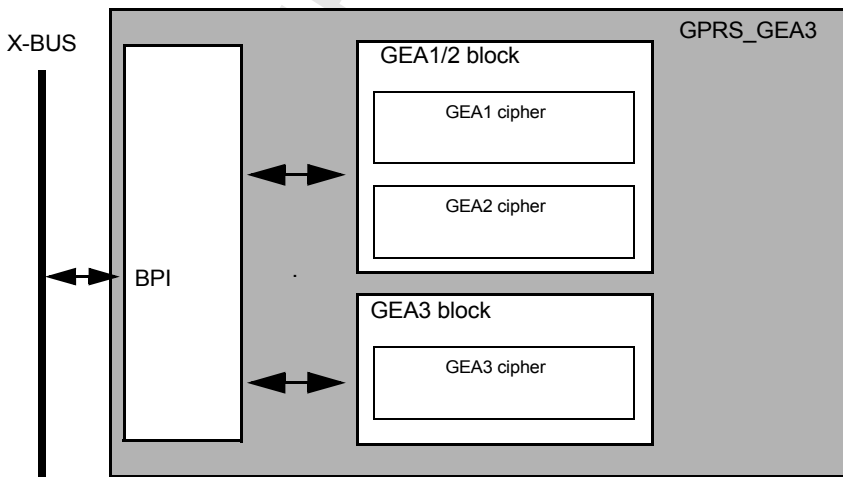
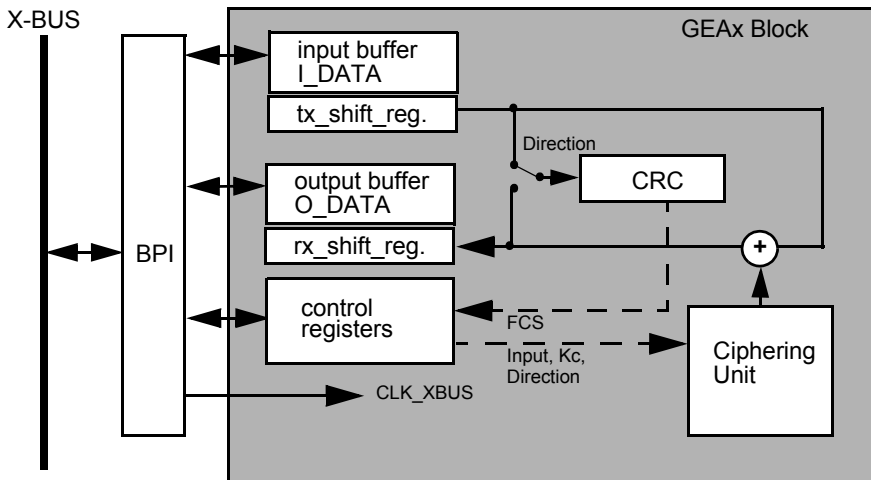


Figure 10-81 Example GEA1/GEA2 or GEA3 Block Diagram



### Initialization

Prior to using of the GPRS\_GEA3 block, in any mode, the block has to be initialized:

1. Select the type of processing to be used (GEA1, GEA2, or GEA3) via the [GPRSCTRL\\_12.GEA\\_MODE](#) or [GPRSCTRL\\_3.GEA\\_MODE](#).
2. Load the input and Kc input keys into the corresponding GPRS\_GEA3 block registers according to the mode selected in Step 1. The keys are processed (serial pre-shifted in the ciphering registers) by the block.
3. Depending on the selected block, the [GPRSCTRL\\_12](#) or [GPRSCTRL\\_3](#) register bits **DIRECTION**, **CRC\_CTRL**, and **CIPH\_CTRL** have to be set to select the initialization mode.

The **DIRECTION** bit configures the GPRS unit for either the encoding (uplink) or decoding (downlink).

The **CRC\_CTRL** and **CIPH\_CTRL** bits determine whether the CRC and ciphering algorithm are to be used or not. This is necessary because the GPRS header, Information Field, and parity check bits are processed with different settings.

At the end of the LLC block processing, the CRC register carries the FCS contents, which are shifted into the FCS0 and FCS1 registers:

- In the encoding mode, the MCU attaches the FCS content at the end of the Information Field.
- In the decoding mode the CRC register content is, for an error free transmission, equal to 6D 8930<sub>H</sub>.

CONFIDENTIAL

GPRS Cipher Unit

The CRC-24 generator polynomial used for GPRS (GEA1-2-3) application is given by:  
 $G(x) = x^{24} + x^{23} + x^{21} + x^{20} + x^{19} + x^{17} + x^{16} + x^{15} + x^{13} + x^8 + x^7 + x^5 + x^4 + x^2 + 1$

*Note: For GEA3 mode, this polynomial is hardcoded and cannot be changed. Polynomial can only be configured in GEA1-2 mode.*

- The initialization can be started by setting the respective **INIT** bit. This can be done at the same time as the **DIRECTION**, **CRC\_CTRL** and **CIPH\_CTRL** bits are set/reset, or at any point later in time. The initialization is finalized when the GPRS\_GEA3 block resets the **INIT** bit.

There are two independent parts of the initialization procedure:

- CRC Unit initialization when **CRC\_CTRL** = 1 (the FCS register has to be programmed by the MCU to 111...111 corresponding to GPRS requirement).
- Ciphering Unit initialization (when **CRC\_CTRL** = 1, the Input and Kc are serially shifted through the given feedback registers).

The initialization procedure duration depends on ciphering: if ciphering is required, initialization is longer than without ciphering.

Ciphering performed for GPRS1/2 is different from ciphering performed for GEA3 through the KGcore function.

## Processing

After initialization, the MCU processes the data by:

- Writing the length of the input data (1 or 2 bytes) to the GPRS\_GEA3 **GPRSCTRL\_12/3.IN\_SIZE**.
- Writing a data segment (the specified number of bytes [1 or 2] of the data block) into the input buffer, which starts the calculation on that segment.
- The **GPRSCTRL\_12/3.VALID\_IN** bit is set until the segment written to the input buffer is latched into the tx\_shift\_reg. After latching, **VALID\_IN** is reset.  
(A new data segment (1 or 2 bytes) can be written into the input buffer as soon as **VALID\_IN** is reset.)
- Reading the output data from the output buffer after the **GPRSCTRL\_12/3.VALID\_OUT** bit is set.

For more details on GPRS ciphering algorithm refer to the ETSI GEA specification. For more details on LLC layer specification refer to ETSI GSM 04.64 document, Version 6.1.0 Release 1997. Both encryption algorithms (GEA1 and GEA2) are implemented.

By selecting GEA3, the implemented GPRS\_GEA3 encryption algorithm is used.

For GPRS\_GEA3 encryption algorithm, registers **INPUT\_REG\_00\_3** through **INPUT\_REG\_11\_3** contain the 64 bits "A" value described in ETSI specification as an input for the KGcore function.



CONFIDENTIAL

GPRS Cipher Unit

### Initialization and Processing Durations

Initialization duration is about 300 cycles for GPRS\_GEA1/2 and about 220 cycles for GPRS\_GEA3.

**Note: The end of initialization must be found by polling the MCU register.**

The processing time is 10 cycles for 1 byte, and 18 cycles for 2 bytes. Processing time is identical for GPRS\_GEA1/2 and GPRS\_GEA3.

### Uplink

**Figure 10-51** shows an example for the use of the hardware unit from the MCU side with the following properties:

- LLC-Frame uplink
- 11 bytes header
- 101 bytes information field
- Ciphering on
- FCS calculation over header and information field.
- Input, Processing, and Output numbers are byte numbers.

*Note: Advice for software designer: because the data is word-aligned, align uneven start addresses in the processing first octet.*

**Table 10-51 Uplink Example**

Command from MCU	Input	Processing	Output
<b>Select GPRS_GEA1/2 / GPRS_GEA3</b>			
Set GEA_MODE			
<b>Init Ciphering-Unit</b>			
Write Kc (KC_REG_0 -KC_REG_3 for GPRS_GEA1/2) (KC_REG_0_L -KC_REG_3_H for GPRS_GEA3)			
Write Input (INPUT_REG_0 to INPUT_REG_1 for GPRS) (INPUT_REG00 to INPUT_REG11 for GPRS_GEA3)			

**CONFIDENTIAL**

**GPRS Cipher Unit**

<b>Command from MCU</b>	<b>Input</b>	<b>Processing</b>	<b>Output</b>
Clear DIRECTION bit (= uplink) and Set CIPHER_INIT			
Wait until CIPHER_INIT is 0			

#### **Init FCS-Unit**

Write FFFF <sub>H</sub> in FCS_REG_0			
Write 00FF <sub>H</sub> in FCS_REG_1			

#### **Init Control Flags**

Set IN_SIZE (= 16 bits) and Clear CIPH_CTRL (no ciphering for header) and Set CRC_CTRL (switch on CRC check)			
---	--	--	--

**CONFIDENTIAL**

**GPRS Cipher Unit**

Command from MCU	Input	Processing	Output
------------------	-------	------------	--------

**Processing LLC-Frame**

Write IN_DATA	Head[0,1]	Head[0,1]	
Write IN_DATA	Head[2,3]		
Wait 10 resp. 2 clocks <sup>1)</sup>			
Read OUT_DATA		Head[2,3]	Head[0,1]
Write IN_DATA	Head[4,5]		
Wait 10 resp. 2 clocks <sup>1)</sup>			

Read OUT_DATA		Head[4,5]	Head[2,3]
Write IN_DATA	Head[6,7]		
Wait 10 resp. 2 clocks <sup>1)</sup>			

Read OUT_DATA		Head[6,7]	Head[4,5]
Write IN_DATA	Head[8,9]		
Wait 10 resp. 2 clocks <sup>1)</sup>			

Read OUT_DATA		Head[8,9]	Head[6,7]
Clear IN_SIZE (= 8 bit input)			
Write IN_DATA	Head[10]		
Wait 10 resp. 2 clocks <sup>1)</sup>			

Read OUT_DATA		Head[10]	Head[8,9]
Set IN_SIZE (= 16 bit input) and Set CIPH_CTRL (switch ciphering on for information field)			
Write IN_DATA	Info[0,1]		
Wait 10 resp. 2 clocks <sup>1)</sup>			

Read OUT_DATA		Info[0,1]	Head[10]
Write IN_DATA	Info[2,3]		
Wait 10 resp. 2 clocks <sup>1)</sup>			

**CONFIDENTIAL**

**GPRS Cipher Unit**

Command from MCU	Input	Processing	Output
Read OUT_DATA		Info[96,97]	Info[94,95]
Write IN_DATA	Info[98,99]		
Wait 10 resp. 2 clocks <sup>1)</sup>			
Read OUT_DATA		Info[98,99]	Info[96,97]
Clear IN_SIZE (= 8 bit input)			
Write IN_DATA	Info[100]		
Wait 10 resp. 2 clocks <sup>1)</sup>			
Read OUT_DATA		Info[100]	Info[98,99]
Wait 10 resp. 2 clocks <sup>1)</sup>			
Read and Invert FCS[0]			
Read and Invert FCS[1]			
Set IN_SIZE (= 16 bit Input)			
Write IN_DATA	FCS_0	FCS_0	
Read OUT_DATA			Info[100]
Clear IN_SIZE (= 8 bit Input)			
Write IN_DATA	FCS_1		
Wait 10 resp. 2 clocks <sup>1)</sup>			
Read OUT_DATA		FCS_1	FCS_0
Write DUMMY	Dummy		
Wait 10 resp. 2 clocks <sup>1)</sup>			
Read OUT_DATA		Dummy	FCS_1

<sup>1)</sup> Wait 10 resp. 2 clocks: the GPRS hardware needs 18 clock cycles for 16-bit processing resp. 10 clocks for 8-bit. From this value 4 clocks for input data write and 4 clocks for output data read have to be subtracted

**CONFIDENTIAL**

**GPRS Cipher Unit**

**Downlink**

The following table shows an example for the usage of the hardware unit from the MCU side with the following properties:

- Selecting GPRS
- LLC-Frame downlink
- 11 bytes header
- 101 bytes information field
- Ciphering on
- FCS calculation over header and information field.

**Table 10-52 Downlink Example**

Command from MCU	Input	Processing	Output
<b>Select GPRS / GPRS_GEA3</b>			
Set GEA_MODE = 00 (GPRS_GEA1/2 chosen)			
<b>Init Ciphering-Unit</b>			
Write Kc to KC_REG_0 -KC_REG_3			
Write Input (32 bits) to INPUT_REG_0 and INPUT_REG_1			
Set DIRECTION bit (= downlink) and Set CIPHER_INIT			
Wait until CIPH_INIT is 0			
<b>Init FCS-Unit</b>			
Write Polynom to 32 bit Register (only once)			
Write FFFF <sub>H</sub> in FCS_REG_0			
Write 00FF <sub>H</sub> in FCS_REG_1			
<b>Init Control Flags</b>			
Set IN_SIZE (= 16 bits) and Clear CIPH_CTRL (no ciphering for header) and Set CRC_CTRL (switch on CRC check)			

**CONFIDENTIAL**

**GPRS Cipher Unit**

Command from MCU	Input	Processing	Output
------------------	-------	------------	--------

**Processing LLC-Frame**

Write IN_DATA	Head[0,1]	Head[0,1]	
Write IN_DATA	Head[2,3]		
Wait 10 resp. 2 clocks <sup>1)</sup>			
Read OUT_DATA		Head[2,3]	Head[0,1]
Write IN_DATA	Head[4,5]		
Wait 10 resp. 2 clocks <sup>1)</sup>			

Read OUT_DATA		Head[4,5]	Head[2,3]
Write IN_DATA	Head[6,7]		
Wait 10 resp. 2 clocks <sup>1)</sup>			

Read OUT_DATA		Head[6,7]	Head[4,5]
Write IN_DATA	Head[8,9]		
Wait 10 resp. 2 clocks <sup>1)</sup>			

Read OUT_DATA		Head[8,9]	Head[6,7]
Clear IN_SIZE (= 8 bit input)			
Write IN_DATA	Head[10]		
Wait 10 resp. 2 clocks <sup>1)</sup>			

Read OUT_DATA		Head[10]	Head[8,9]
Set IN_SIZE (= 16 bit input) and Set CIPH_CTRL (switch ciphering on for information field)			
Write IN_DATA	Info[0,1]		
Wait 10 resp. 2 clocks <sup>1)</sup>			

Read OUT_DATA		Info[0,1]	Head[10]
Write IN_DATA	Info[2,3]		
Wait 10 resp. 2 clocks <sup>1)</sup>			

**CONFIDENTIAL**

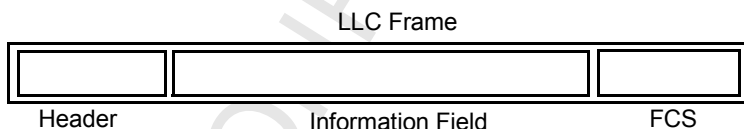
**GPRS Cipher Unit**

Command from MCU	Input	Processing	Output
Read OUT_DATA		Info[98,99]	Info[96,97]
Write IN_DATA	Info[100] + 8 bit FCS		
Wait 10 resp. 2 clocks <sup>1)</sup>			
Read OUT_DATA		Info[100] + 8 bit FCS	Info[98,99]
Write IN_DATA	16 bit FCS		
Wait 10 resp. 2 clocks <sup>1)</sup>			
Read OUT_DATA		16 bit FCS	Info[100] + 8 bit FCS
Wait 10 resp. 2 clocks <sup>1)</sup>			
Read FCS[0]			FCS_0
Read FCS[1]			FCS_1

<sup>1)</sup> Wait 10 resp. 2 clocks: the GPRS hardware needs 18 clock cycles for 16-bit processing resp. 10 clocks for 8-bit. From this value 4 clocks for input data write and 4 clocks for output data read have to be subtracted.

**Figure 10-82** shows a basic structure of a LLC frame.

**Figure 10-82 Structure of a LLC frame (Simplified Example)**

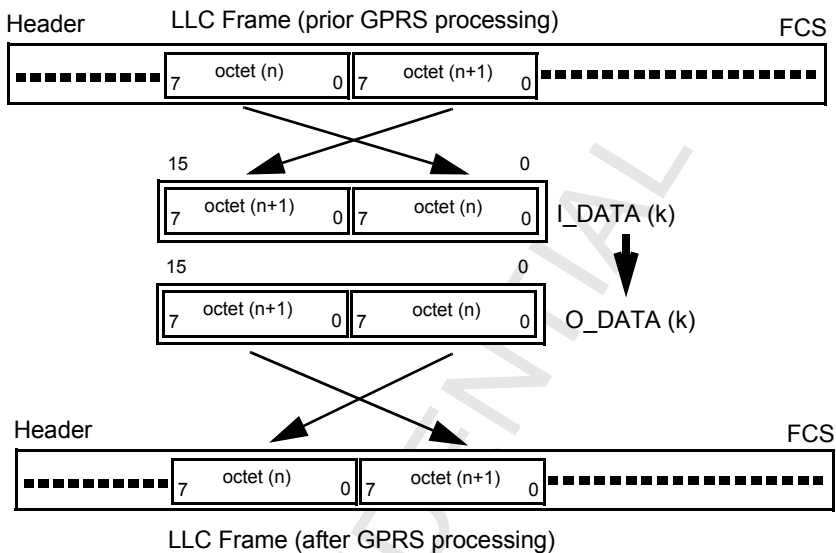


CONFIDENTIAL

GPRS Cipher Unit

**Figure 10-83** shows the word-splitting and MSB/LSB assignment in the LLC frame. Be aware of even address alignment while word-splitting.

**Figure 10-83 Splitting of the LLC Frame Into Words**



**Figure 10-84 MSB/LSB Processing Order within the GPRS Block**

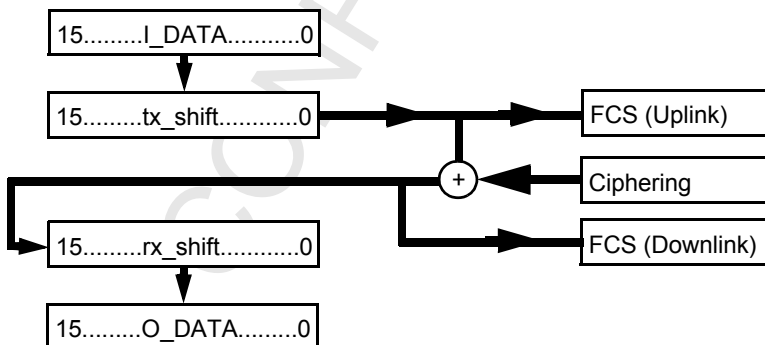
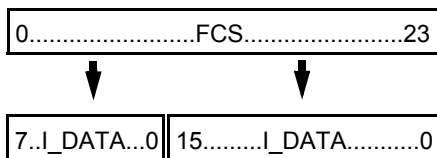




Figure 10-85 FCS Cipherring



### 10.15.3 Physical Interface

Table 10-53 Clock, Reset

Signal Name	Direction	Source/ Destination	Activity	Description
gea_clk <sup>1)</sup>	Input	MCU		X-Bus clock
gprs_clk_en_i	Input	MCU	1	Tied to 1
select_gea_i_s	Input	MCU		Activated for any BPI access
reset_n	Input	Core	Low	Asynchronous Reset

<sup>1)</sup> Refer to the Clock Domain in the [System Integration: \(on Page 921\)](#).

Table 10-54 Data Interface

Signal Name	Direction	Source/ Destination	Activity	Description
XBus Address	Input	MCU Core		
XBus Data	I/O	MCU Core		

### 10.15.4 Interrupts

The GEA3 block can generate an interrupt after the data have been written in the data out registers

#### GEA1/2

The interrupt is raised when the [GPRCTRL\\_12/3.VALID\\_OUT](#) is set to 1.  
The interrupt is cleared when read access to [O\\_DATA\\_12/3](#).

#### GEA3

The interrupt is raised when the [GPRCTRL\\_3.VALID\\_OUT](#) is set to 1.

CONFIDENTIAL

GPRS Cipher Unit

The interrupt is cleared when read access to [O\\_DATA\\_3](#).

### 10.15.5 MCU Register Interface for GPRS\_GEA1/2

*Note: GPRS\_GEA1/2 and GPRS\_GEA3 do not have the same MCU register interface.*

*The following description only apply for GPRS\_GEA1/2.*

*For GPRS\_GEA3 register interface, refer to [Section 10.15.6 MCU Register Interface for GPRS\\_GEA3 \(on Page 939\)](#).*

*Note: The same address range applies to both blocks. The register definitions are different depending on value of [GPRCTRL\\_12.GEA\\_MODE](#) field.*

Refer to [Section 12.3 X-Bus Register Addresses \(on Page 1285\)](#) for the addresses of these registers.

#### 10.15.5.1 GPRS\_GEA1/2 Identification Register

GPRSID\_12

GPRS\_GEA1/2 Identification Register

Reset value: A402<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Module_ID								Revision_Number							

Field	Bits	Type	Description
Revision_Number	0:7	r	<b>GPRS Revision Number</b> These hard-wired bits are used for module revision numbering.
Module_ID	8:15	r	<b>GPRS Identification Number</b> These hard-wired bits are used for module identification numbering.

**CONFIDENTIAL**

**GPRS Cipher Unit**

### 10.15.5.2 GPRS\_GEA1/2 Control Register

**GPRSCTRL\_12/3**

(**GPRSCTRL\_3** is on [Page 939](#).)

**GPRSCTRL\_12**

**GPRS\_GEA1/2 Control Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED							GEA MO DE	GEA MO DE	VALI D OUT	VALI D IN	CIPH CT RL	CRC CT RL	DIRE CTIO N	IN_S IZE	INIT

Field	Bits	Type	Description
INIT	0	rw	<b>Initialization Phase Indicator</b> Set to 1 by the MCU. Reset to 0 by GPRS block. Can be polled by the MCU. As a consequence of the Init phase, <a href="#">GPRSCTRL_3 (on Page 939)</a> .START_KASUMI is set to 1 because the Init phase sequence includes a Kasumi start. 0 Initialization finalized 1 Start of initialization
IN_SIZE	1	rw	MCU informs the GPRS unit if the lower 8-bit or the whole 16-bit word content are to be used for processing: 0 MCU transfers 8 bits from <a href="#">I_DATA_12</a> to <a href="#">O_DATA_12</a> 1 MCU transfers 16 bits from <a href="#">I_DATA_12</a> to <a href="#">O_DATA_12</a>
DIRECTION	2	rw	<b>Selects the encoding/decoding procedure:</b> 0 Uplink channel 1 Downlink channel
CRC_CTRL	3	rw	0 CRC calculation switched off, necessary for unprotected data stream 1 CRC calculation switched on
CIPH_CTRL	4	rw	0 Cipheryng switched off 1 Cipheryng switched on
VALID_IN	5	r	0 Data can be written into the <a href="#">I_DATA_12</a> 1 MCU has written the <a href="#">I_DATA_12</a> , the content is processed

**CONFIDENTIAL**

**GPRS Cipher Unit**

Field	Bits	Type	Description
<b>VALID_OUT</b>	6	r	0 Reset after the <b>O_DATA_12</b> content was picked-up by the MCU 1 <b>O_DATA_12</b> has a valid content When this bit is set to 1, interrupt GEA3_int is raised.
<b>GEA_MODE</b>	7:8	rw	00 GEA1 is selected 01 GEA2 is selected 1X GEA3 is selected
<b>RESERVED</b>	15:9	r	Reserved; these bits must be left at their reset values.

### 10.15.5.3 GPRS\_GEA1/2 Input Data Register

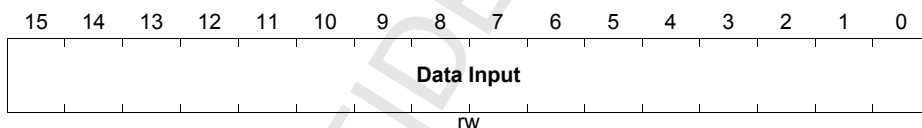
**I\_DATA\_12/3**

(**I\_DATA\_3** is on [Page 941](#).)

**I\_DATA\_12**

**GPRS\_GEA1/2 Input Data Register**

**Reset value: 0000<sub>H</sub>**



### 10.15.5.4 GPRS\_GEA1/2 Output Data Register

A read access to this register clears the GEA3\_int interrupt.

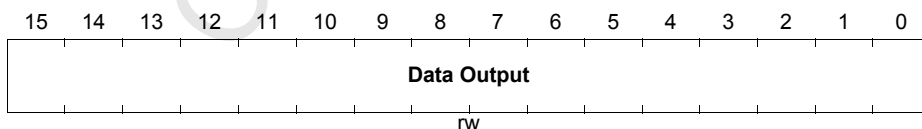
**O\_DATA\_12/3**

(**O\_DATA\_3** is on [Page 941](#).)

**O\_DATA\_12**

**GPRS\_GEA1/2 Output Data Register**

**Reset value: 0000<sub>H</sub>**



**CONFIDENTIAL**

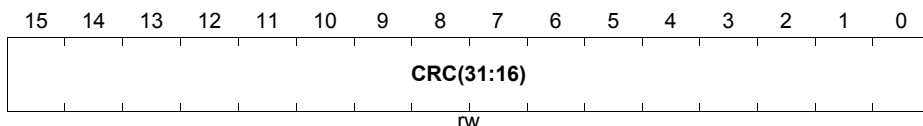
**GPRS Cipher Unit**

### 10.15.5.5 GPRS\_GEA1/2 Checksum Registers

**FCS\_REG\_0\_12**

**GPRS\_GEA1/2 Checksum Register 0**

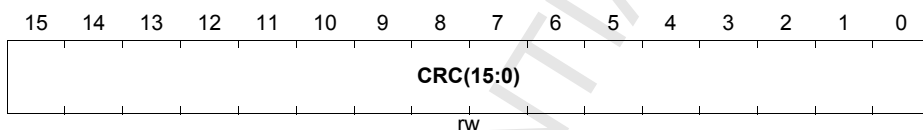
**Reset value: 0000<sub>H</sub>**



**FCS\_REG\_1\_12**

**GPRS\_GEA1/2 Checksum Register 1**

**Reset value: 0000<sub>H</sub>**

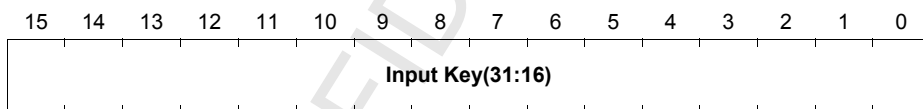


### 10.15.5.6 GPRS\_GEA1/2 Input Key Registers

**INPUT\_REG\_0\_12**

**GPRS\_GEA1/2 Input Key Register 0**

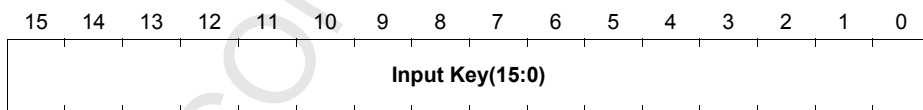
**Reset value: 0000<sub>H</sub>**



**INPUT\_REG\_1\_12**

**GPRS\_GEA1/2 Input Key Register 1**

**Reset value: 0000<sub>H</sub>**



Field	Bits	Type	Description
Input Key(31:0)	15:0	rw	<b>Input key for Initialization</b> The <b>INPUT_REG_0_12</b> and <b>INPUT_REG_1_12</b> (16 bits + 16 bits) have to be directly filled with the 32 bits input parameter as defined in 04.64 ETSI specification.

**CONFIDENTIAL**

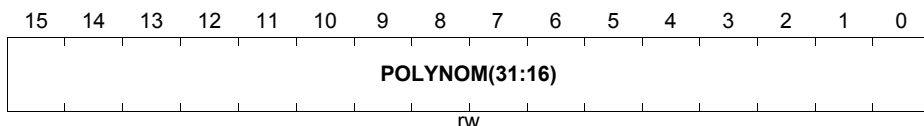
**GPRS Cipher Unit**

### 10.15.5.7 GPRS\_GEA1/2 Polynom Configuration Registers

**POLYNOM\_0\_12**

**GPRS\_GEA1/2 Polynom Configuration Register 0**

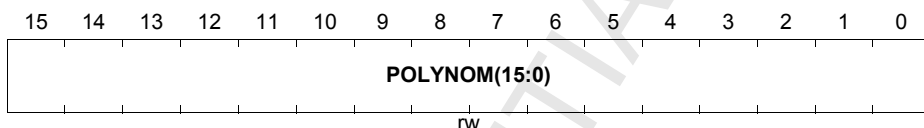
**Reset value: 0000<sub>H</sub>**



**POLYNOM\_1\_12**

**GPRS\_GEA1/2 Polynom Configuration Register 1**

**Reset value: 0000<sub>H</sub>**

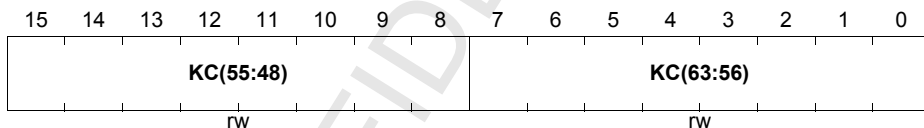


### 10.15.5.8 GPRS\_GEA1/2 Cipher Key Registers

**KC\_REG\_0\_12**

**GPRS\_GEA1/2 Cipher Key Register 0**

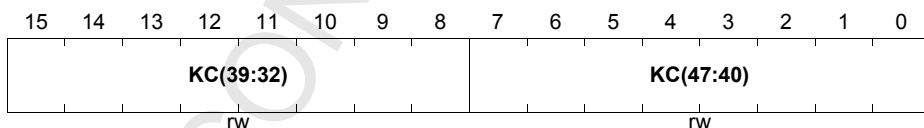
**Reset value: 0000<sub>H</sub>]**



**KC\_REG\_1\_12**

**GPRS\_GEA1/2 Cipher Key Register 1**

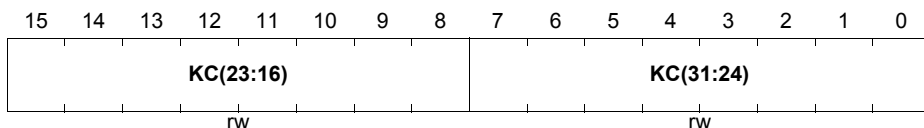
**Reset value: 0000<sub>H</sub>**



**KC\_REG\_2\_12**

**GPRS Cipher Key Register 2**

**Reset value: 0000<sub>H</sub>**



**CONFIDENTIAL**

**GPRS Cipher Unit**

### KC\_REG\_3\_12

**GPRS Cipher Key Register 3**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KC(7:0)								KC(15:8)							
rw								rw							

## 10.15.6 MCU Register Interface for GPRS\_GEA3

*Note: GPRS and GPRS\_GEA3 do not have the same MCU register interface. The following description only apply for GPRS\_GEA3.*

*For GPRS register interface, refer to [Section 10.15.5. MCU Register Interface for GPRS\\_GEA1/2 \(on Page 934\)](#)*

Refer to [Section 12.3 X-Bus Register Addresses \(on Page 1285\)](#) for the addresses of these registers.

### 10.15.6.1 GPRS\_GEA3 Identification Register

**GPRSID\_3**

**GPRS\_GEA3 Identification Register**

**Reset value: A402<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Module_ID								Revision_Number							

Field	Bits	Type	Description
Revision_Number	0:7	r	<b>GPRS Revision Number</b> These hard-wired bits are used for module revision numbering.
Module_ID	8:15	r	<b>GPRS Identification Number</b> These hard-wired bits are used for module identification numbering.

### 10.15.6.2 GPRS\_GEA3 Control Register

**GPRCTRL\_3**

**GPRS\_GEA3 Control Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED						START KAS UMI	GEA MO DE	GEA MO DE	VAL ID OUT	VAL ID IN	CIPH CT RL	CRC CT RL	DIRE CTIO N	IN S IZE	INIT

**CONFIDENTIAL**

**GPRS Cipher Unit**

Field	Bits	Type	Description
<b>INIT</b>	0	rw	<b>Initialization Phase Indicator</b> Set to 1 by the MCU. Reset to 0 by GPRS block. Can be polled by the MCU. As a consequence of the Init phase, <b>START_KASUMI</b> is set to 1 because the Init phase sequence includes a Kasumi start. 0     Initialization finalized 1     Start initialization
<b>IN_SIZE</b>	1	rw	MCU informs the GPRS unit if the lower 8-bit or the whole 16-bit word content are to be used for processing: 0     MCU transfers 8 bits from <b>I_DATA_3</b> to <b>O_DATA_3</b> 1     MCU transfers 16 bits from <b>I_DATA_3</b> to <b>O_DATA_3</b>
<b>DIRECTION</b>	2	rw	<b>Selects the encoding/decoding procedure:</b> 0     Uplink channel 1     Downlink channel
<b>CRC_CTRL</b>	3	rw	<b>CRC Calculation</b> 0     CRC calculation switched off, necessary for unprotected data stream 1     CRC calculation switched on
<b>CIPH_CTRL</b>	4	rw	<b>Ciphering Control</b> 0     Ciphering switched off 1     Ciphering switched on
<b>VALID_IN</b>	5	r	0     Data can be written into the <b>I_DATA_3</b> 1     MCU has written the <b>I_DATA_3</b> , the content will be processed
<b>VALID_OUT</b>	6	r	0     Reset after the <b>O_DATA_3</b> content was picked-up by the MCU 1 <b>O_DATA_3</b> has a valid content. When this bit is set to 1, interrupt GEA3_int is raised.
<b>GEA_MODE</b>	7:8	rw	<b>GEA Mode Selection</b> 00     GEA1 is selected 01     GEA2 is selected 1X     GEA3 is selected



**CONFIDENTIAL**

**GPRS Cipher Unit**

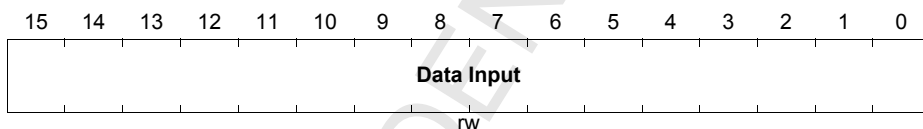
Field	Bits	Type	Description
<b>START_KASUMI</b>	9	rw	This indicates valid data input and cipher key registers and starts ciphering. This bit has to be set to 1 by software only for stand alone KASUMI operations. In normal use, it is "don't care". 0     Reset by hardware when ciphering is done 1     Set by software to start Kasumi or by hardware if INIT bit has been set to 1.
<b>RESERVED</b>	15:9	r	Reserved; these bits must be left at their reset values.

### 10.15.6.3 GPRS\_GEA3 Input Data Register

**I\_DATA\_3**

**GPRS\_GEA3 Input Data Register**

**Reset value: 0000<sub>H</sub>**



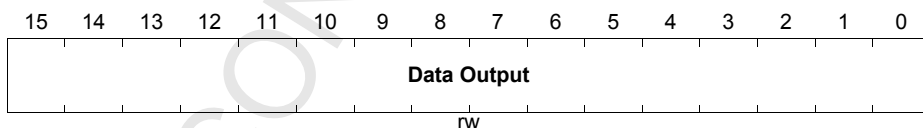
### 10.15.6.4 GPRS\_GEA3 Output Data Register

A read access to this register clears the GEA3\_int interrupt.

**O\_DATA\_3**

**GPRS\_GEA3 Output Data Register**

**Reset value: 0000<sub>H</sub>**

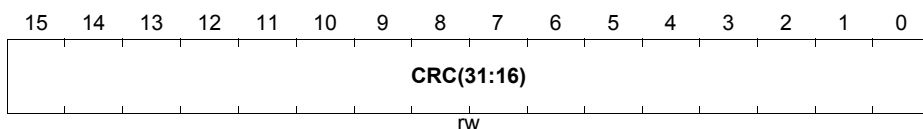


### 10.15.6.5 GPRS\_GEA3 Checksum Registers

**FCS\_REG\_0\_3**

**GPRS\_GEA3 Checksum Register 0**

**Reset value: 0000<sub>H</sub>**



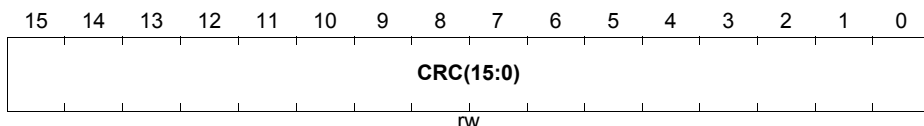
**CONFIDENTIAL**

**GPRS Cipher Unit**

### **FCS\_REG\_1\_3**

**GPRS\_GEA3 Checksum Register 1**

**Reset value: 0000<sub>H</sub>**



### **10.15.6.6 GPRS\_GEA3 Input Key Registers**

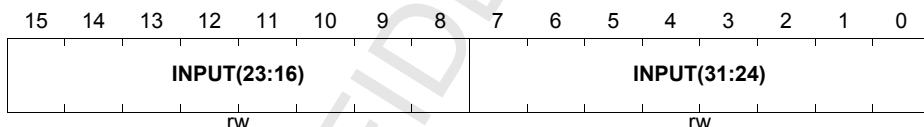
Input data for KGcore initialization.

For these registers be very careful with bit fields. Software developers must be careful about the data contained in [INPUT\\_REG\\_00\\_3](#), [INPUT\\_REG\\_01\\_3](#), [INPUT\\_REG\\_10\\_3](#) and [INPUT\\_REG\\_11\\_3](#) registers: the LSB bits are stored in the lower memory space of each register (for example, bit field **INPUT[7:0]** is located in the MSB bits of [INPUT\\_REG\\_01\\_3](#), although bit field **INPUT[8:15]** is located in the LSB bits of [INPUT\\_REG\\_01\\_3](#); then [INPUT\\_REG\\_00\\_3](#) contains the following input data with bit field **INPUT [23:16]** in the MSB and bit field **INPUT[31:24]** in the LSB).

#### **INPUT\_REG\_00\_3**

**GPRS\_GEA3 Input Key Register 0**

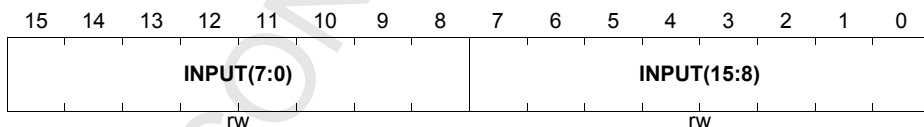
**Reset value: 0000<sub>H</sub>**



#### **INPUT\_REG\_01\_3**

**GPRS\_GEA3 Input Key Register 1**

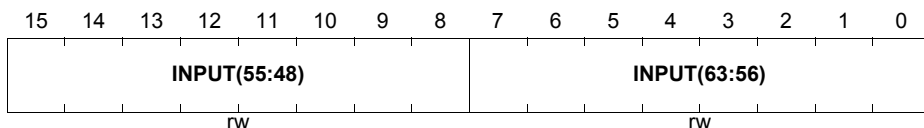
**Reset value: 0000<sub>H</sub>**



#### **INPUT\_REG\_10\_3**

**GPRS\_GEA3 Input Key Register 2**

**Reset value: 0000<sub>H</sub>**



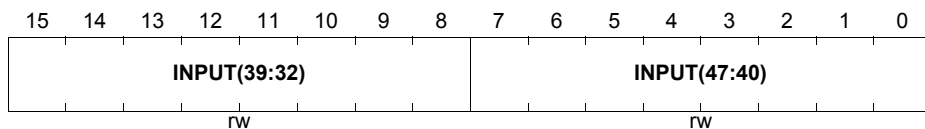
**CONFIDENTIAL**

**GPRS Cipher Unit**

**INPUT\_REG\_11\_3**

**GPRS\_GEA3 Input Key Register 3**

**Reset value: 0000<sub>H</sub>**



CONFIDENTIAL

**CONFIDENTIAL**

**GPRS Cipher Unit**

*Note: INPUT(63:0) is set to A according to:*

$A = CC \parallel CB \parallel CD \parallel 00 \parallel CA \parallel CE$   
MSB
LSB

*where:*

$CA[0]...CA[7] = 1111\ 1111$

$CB[0]...CB[4] = 0\ 0000$

$CC[0]...CC[31] = INPUT[0]...INPUT[31]$  (32 bits Input parameter as defined in 04.64 ETSI specification)

$CD[0] = DIRECTION[0]$  (1 bit Direction parameter as defined in 04.64 ETSI specification, duplicate of [GPRSCtrl\\_3.DIRECTION](#))

$CE[0]...CE[15] = 0000\ 0000\ 0000\ 0000$

### 10.15.6.7 GPRS\_GEA3 Cipher Key Register

These registers have to be set to the Kc value provided by the network before processing each new frame.

KC is the cipher key provided to the hardware.

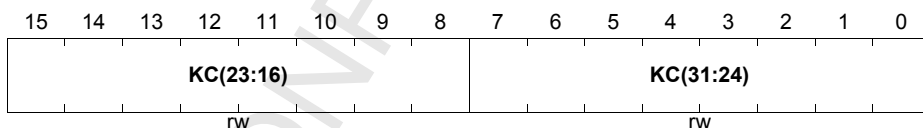
**Note: If the Kc length is < 128 bits, then:**

- KC bits 0 to (Kc length - 1) must be set to Kc range 0 to (Kc length - 1)
- KC bits (Kc length) to 127 must be set to Kc range 0 to (127 - Kc length).

#### KC\_REG\_0\_L\_3

**GPRS\_GEA3 Cipher Key Register Low 0**

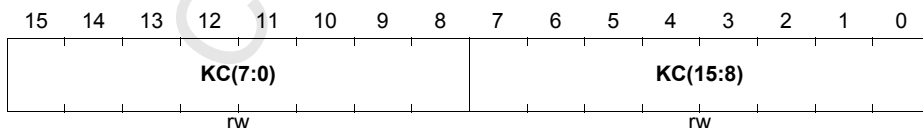
**Reset value: 0000<sub>H</sub>**



#### KC\_REG\_0\_H\_3

**GPRS\_GEA3 Cipher Key Register High 0**

**Reset value: 0000<sub>H</sub>**



**CONFIDENTIAL**

**GPRS Cipher Unit**

**KC\_REG\_1\_L\_3**

**GPRS\_GEA3 Cipher Key Register Low 1**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KC(55:48)								KC(63:56)							
rw								rw							

**KC\_REG\_1\_H\_3**

**GPRS\_GEA3 Cipher Key Register High 1**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KC(39:32)								KC(47:40)							
rw								rw							

**KC\_REG\_2\_L\_3**

**GPRS\_GEA3 Cipher Key Register Low 2**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KC(87:80)								KC(95:88)							
rw								rw							

**KC\_REG\_2\_H\_3**

**GPRS\_GEA3 Cipher Key Register High 2**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KC(71:64)								KC(79:72)							
rw								rw							

**KC\_REG\_3\_L\_3**

**GPRS\_GEA3 Cipher Key Register Low 3**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KC(119:112)								KC(127:120)							
rw								rw							

**KC\_REG\_3\_H\_3**

**GPRS\_GEA3 Cipher Key Register High 3**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KC(103:96)								KC(111:104)							
rw								rw							

CONFIDENTIAL

CONFIDENTIAL

## 11 PD-Bus

History	
Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on Automotive Design Spec, Revision 1.1, 2002-09
Changes for Rev. 1.01	
<b>Page 949</b>	Updated <b>System Integration</b> : WS00005665, WS00006682
Changes for Rev. 1.04	
<b>Page 973</b>	Removed "SRC Register" column in <b>Table 11-4 Interrupt Sources</b> WS00008455
Changes for Rev. 1.06	

## 11.1 I2C Bus Interface

### System Integration:

- Supply domain: VDD\_MAIN
- Chip internal interfaces:
  - Clock domain: Refer to [Section 10.4.1.3 Sub-System Clocks and Enables \(on Page 661\)](#) and see [Figure 10-10 Clock Enable \(on Page 662\)](#).
  - Bus domain: PD-Bus
- Interrupt sources:
- Monitor Pins: refer to [Section 11.8.10 Internal Signal Monitoring \(on Page 1209\)](#)

### 11.1.1 Introduction

IIC supports a protocol that allows devices to communicate directly with each other via two wires. One line is responsible for clock transfer and synchronization (SCL), the other is responsible for the data transfer (SDA).

The on-chip IIC Bus module connects the platform buses to other external controllers and/or peripherals via the two-line serial IIC interface. The IIC Bus module provides communication at data rates of up to 400kbit/s and features 7-bit addressing as well as 10-bit addressing. This module is fully compatible to the IIC bus protocol.



**CONFIDENTIAL**

**I2C Bus Interface**

The module can operate in three different modes:

- **Master Mode**, the IIC controls the bus transactions and provides the clock signal.
- **Slave Mode**, an external master controls the bus transactions and provides the clock signal.
- **Multimaster Mode**, several masters can be connected to the bus, that is, the IIC can be master or slave.

The on-chip IIC bus module allows efficient communication via the common IIC bus. The module unloads the CPU of low level tasks such as:

- (De)Serialization of bus data.
- Generation of start and stop conditions.
- Monitoring the bus lines in slave mode.
- Evaluation of the device address in slave mode.
- Bus access arbitration in multimaster mode.

## **Features**

- Extended buffer allows up to 4 send/receive data bytes to be stored.
- Selectable baud rate generation.
- Support of standard 100kBaud and extended 400kBaud data rates.
- Operation in 7-bit addressing mode or 10-bit addressing mode.
- Flexible control via interrupt service routines or by polling.
- Dynamic access to up to 4 physical IIC buses.

## **Applications**

- EEPROMs
- 7-Segment Displays
- Keyboard Controllers
- On-Screen Display
- Audio Processors.

### **11.1.2 Operational Overview**

Data is transferred by the 2-line IIC bus (SDA, SCL) using a protocol that ensures reliable and efficient transfers. This protocol clearly distinguishes regular data transfers from defined control signals which control the data transfers.

The following bus conditions are defined:

- **Bus Idle:** SDA and SCL remain high. The IIC bus is currently not used.
- **Data Valid:** SDA stable during the high phase of SCL. SDA then represents the transferred bit. There is one clock pulse for each transferred bit of data.

**CONFIDENTIAL**

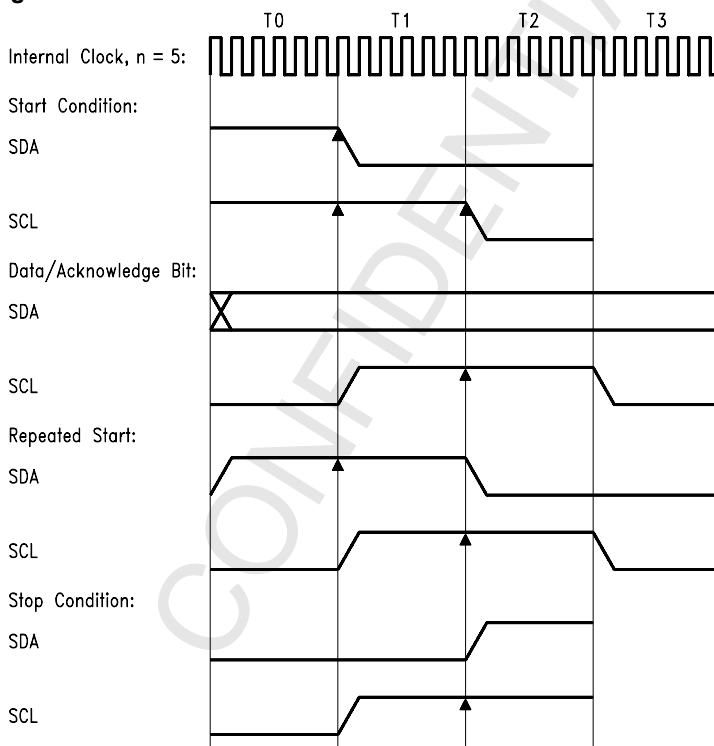
**I2C Bus Interface**

During data transfers, the SDA may only change while SCL is low. If SCL is high, the transfer stops (refer to Stop Transfer below).

- **Start Transfer:** A falling edge on SDA (↘) while SCL is high indicates a start condition. This start condition initiates a data transfer over the IIC bus.
- **Stop Transfer:** A rising edge on SDA (↗) while SCL is high indicates a stop condition. This stop condition terminates a data transfer. Between a start condition and a stop condition, an arbitrary number of bytes may be transferred.

**Figure 11-1** gives examples for these bus conditions.

**Figure 11-1 Bus Conditions**

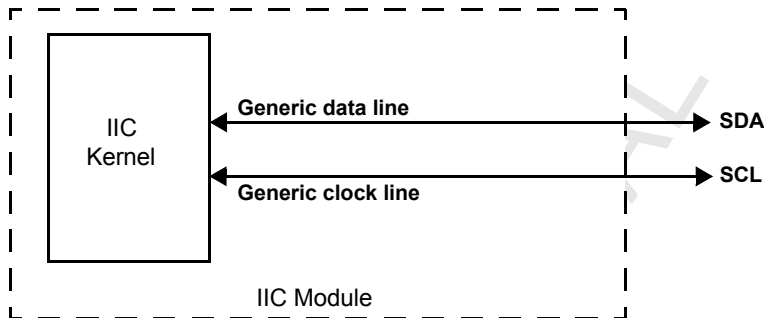


↑ The high level of the respective signal is verified.  
If the signal is low, the previous state (Ti) is repeated.

### 11.1.2.1 Physical IIC-Bus Interface

Communication via the IIC Bus uses two bidirectional lines, the serial data line SDA and the serial clock line SCL (see [Figure 11-2](#)). These two interface lines are connected to two I/O ports.

**Figure 11-2 IIC Bus Line Connections**



Register **IIC\_CFG** (on [Page 970](#)) selects the bus baud rate as well the activation of SDA and SCL lines. So an external IIC channel can be established (baud rate and physical lines) with one single register access.

*Note: Baud rate and physical channels must never be changed (via **IIC\_CFG**) during a transfer.*

### 11.1.2.2 Output Pin Configuration

The pin drivers that are assigned to the IIC channel(s) provide open drain outputs (that is, no upper transistor). This ensures that the IIC module does not put any load on the IIC bus lines while the C166S is not powered. The IIC bus lines, therefore, require pull-up resistors. Values for these resistors depend upon the capacitive load on the I2C lines. Rules for the selection of the values are given in the I2C bus specification.

All pins of the CB-Core that are to be used for IIC bus communication must be switched to output and their alternate function must be enabled (by setting the respective port output latch to 1) before any communication can be established.

If not driven by the IIC module (that is, the corresponding enable bit in register **IIC\_CFG** is 0), they then switch off their drivers (that is, driving it to an open drain output). Due to the external pull-up devices, the respective bus levels are then 1, which is idle.

The IIC module features digital input filters to improve the rejection of noise from the external bus lines.

## CONFIDENTIAL

## I2C Bus Interface

### 11.1.3 Functional Overview

For information about the registers in this section refer to [Section 11.1.4 Registers \(on Page 954\)](#).

#### 11.1.3.1 Operation in Master Mode

If the on-chip IIC module controls the IIC bus (that is, as a bus master), the master mode must be selected via bit field **IIC\_CON.MOD**. The physical channel is configured by a control word written to register **IIC\_CFG**, which activates the interface pins and the baud rate used. The address of the remote slave that is to be accessed is written to either **RTB\_LO.RTB(0,1)** or **RTB\_HI.RTB(2,3)**. The bus is claimed by setting bit **IIC\_CON.BUM**. This generates a start condition on the bus and automatically starts the transmission of the address in **RTB\_LO.RTB0**. Bit **IIC\_CON.TRX** defines the transfer direction (TRX = 1, that is, transmit, for the slave address). A repeated start condition is generated by setting bit **IIC\_CON.RSC**, which automatically starts the transmission of the address previously written to **RTB\_LO.RTB0**. This may be used to change the transfer direction. **IIC\_CON.RSC** is cleared automatically after the repeated start condition has been generated.

The bus is released by clearing bit **IIC\_CON.BUM**. This generates a stop condition on the bus.

#### 11.1.3.2 Operation in Multimaster Mode

If multimaster mode is selected via bit field **IIC\_CON.MOD** the on-chip IIC module can operate concurrently as a bus master or as a slave. The descriptions of these modes apply accordingly.

Multimaster mode implies that several masters are connected to the same bus. As more than one master may try to claim the bus at a given time an arbitration is done on the SDA line. When a master device detects a mismatch between the data bit to be sent and the actual level on the SDA (bus) line it loses the arbitration and automatically switches to slave mode (leaving the other device as the remaining master). This loss of arbitration is indicated by bit **IIC\_ST.AL**, which must be checked by the driver software when operating in multimaster mode. Lost arbitration is also indicated when the software tries to claim the bus (by setting bit **IIC\_CON.BUM**) while the IIC bus is active (indicated by bit **IIC\_ST.BB** = 1). Bit **IIC\_ST.AL** must be cleared via software.

#### 11.1.3.3 Operation in Slave Mode

If the on-chip IIC module shall be controlled via the IIC bus by a remote master (that is, be a bus slave) slave mode must be selected via bit field **IIC\_CON.MOD**. The physical channel is configured by a control word written to register **IIC\_ST**, defining the active interface pins and the baud rate used. It is recommended to have only one SDA and SCL

CONFIDENTIAL

I2C Bus Interface

line active at a time when operating in slave mode. The address for the slave module that can be selected is written to register **IIC\_ADR**.

The IIC module is selected by another master when it receives (after a start condition) either its own device address (stored in **IIC\_ADR**) or the general call address (00<sub>H</sub>). In this case an interrupt is generated and bit **IIC\_ST.SLA** is set indicating the valid selection. The desired transfer mode is then selected via bit **IIC\_CON.TRX** (TRX = 0 for reception, TRX = 1 for transmission).

**For a transmission** the respective data byte is placed into either buffer **RTB\_LO.RTB(0,1)** or **RTB\_HI.RTB(2,3)** (which automatically sets bit **IIC\_CON.TRX**) and the acknowledge behavior is selected via bit **IIC\_CON.ACKDIS**.

**For a reception** the respective data byte is fetched from either buffer **RTB\_LO.RTB(0,1)** or **RTB\_HI.RTB(2,3)** after **IIC\_ST.IRQD** has been activated.

**In both cases** the data transfer itself is enabled by clearing bits **IIC\_ST.IRQD**, **IIC\_ST.IRQP**, and **IIC\_ST.IRQE** which releases the SCL line.

**When a stop condition** is detected, bit **IIC\_CON.SLA** is cleared.

The IIC Bus Configuration Register **IIC\_CFG** selects the bus baud rate and activation of SDA and SCL lines. So an external IIC channel can be established (baud rate and physical lines) with one single register access.

*Note: Refer to [Section 11.1.2.1 Physical IIC-Bus Interface \(on Page 951\)](#).*

**CONFIDENTIAL**

**I2C Bus Interface**

### 11.1.4 Registers

For information about the IIC PD-Bus Register Mapping refer to [Section 12.2 PD-Bus Register Addresses \(on Page 1273\)](#).

All available module registers are summarized in [Table 11-1](#).

**Table 11-1 I2C Register List**

Name	Clock	Access Condition	Description
<a href="#">IIC_ID</a>	cfg_clk <sup>1)</sup>	not bit addressable	I2C Identification Register
<a href="#">IIC_PISEL</a>	cfg_clk <sup>1)</sup>	bit addressable	Input port selection register.
<a href="#">IIC_CON</a>	hw_clk <sup>1)</sup>	bit addressable	System Control Register
<a href="#">IIC_ST</a>	hw_clk <sup>1)</sup>	bit addressable	System Status Register
<a href="#">IIC_ADR</a>	hw_clk <sup>1)</sup>	bit addressable	Bus Address Register
<a href="#">IIC_CFG</a>	cfg_clk <sup>1)</sup>	bit addressable	Bus Configuration Register
<a href="#">RTB_HI</a>	hw_clk <sup>1)</sup>	none	High Receive/Transmit Buffer
<a href="#">RTB_LO</a>	hw_clk <sup>1)</sup>	none	Low Receive/Transmit Buffer

<sup>1)</sup> Refer to the Clock Domain in the [System Integration: \(on Page 948\)](#).

#### 11.1.4.1 I2C Identification Register

**IIC\_ID**

**IIC Identification Register**

**Reset value: 4604<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Module_ID								Revision_Number							
r								r							

Field	Bits	Type	Description
<b>Revision_Number</b>	0:7	r	<b>I2C Revision Number</b> These hard-wired bits are used for module revision numbering.
<b>Module_ID</b>	8:15	r	<b>I2C Identification Number</b> These hard-wired bits are used for module identification numbering.

**CONFIDENTIAL**

**I2C Bus Interface**

### 11.1.4.2 IIC Port Input Selection Register

This register does not need to be programmed. Just leave it at its reset value.

**IIC\_PISEL**

**Port Input Select Register**

**Reset values: 0001<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											SDA IS	RESERVED			SCL IS

Field	Bits	Type	Description
<b>SCLIS</b>	0	rw	<b>Select Input for Clock Signal</b> 0 Signal on pin SCL 1 Not used
<b>SDAIS</b>	4	rw	<b>Select Input for Data Signal</b> 0 Signal on pin SDA 1 Not used
<b>RESERVED</b>	15:8, 3:1	r	Reserved, these bits must be left at their reset values.

**CONFIDENTIAL**

**I2C Bus Interface**

### 11.1.4.3 System Control Registers

The operating mode of the IIC is controlled by the system control register **IIC\_CON** and system status register **IIC\_ST**. These registers contain control bits for mode and error check selection, and status flags for error identification.

Depending on bits **WMEN** and **RMEN**, either the write mirror (WM) or receive mirror (RM) is enabled.

For **IIC\_ST.RMEN** = 1

#### **IIC\_CON**

#### **System Control Register**

**Reset value: 0000<sub>H</sub>**




15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RM</b>								<b>TRX</b>	<b>INT</b>	<b>ACK DIS</b>	<b>BUM</b>	<b>MOD</b>	<b>RSC</b>	<b>M10</b>	

Field	Bits	Type	Description
<b>M10</b>	0	rw	<b>Address Mode</b> 0 7-bit addressing using <b>IIC_ADR.ICA(7..1)</b> 1 10-bit addressing using <b>IIC_ADR.ICA(9..0)</b> .
<b>RSC</b>	1	rwh	<b>Repeated Start Condition</b> 0 No operation. 1 Generate a repeated start condition in (multi) master mode. <b>RSC</b> can not be set in slave mode.  <i>Note: <b>RSC</b> is cleared automatically after the repeated start condition has been sent.</i>
<b>MOD</b>	2:3	rw	<b>Basic Operating Mode</b> 00 IIC module is disabled and initialized (Init-Mode). Transmissions under execution was aborted. 01 Slave mode. 10 Master mode. 11 Multi-Master mode.



**CONFIDENTIAL**

**I2C Bus Interface**

Field	Bits	Type	Description
<b>BUM</b>	4	rwh	<b>Busy Master</b> 0 Clearing bit <b>BUM</b> (  ) generates a stop condition immediately. 1 Setting bit <b>BUM</b> (  ) generates a start condition in (multi)master mode. <i>Note: Setting bit <b>BUM</b> (  ) while <b>BB</b> = 1 generates an arbitration lost situation. In this case <b>BUM</b> is cleared and bit <b>AL</b> is set. <b>BUM</b> can not be set in slave mode.</i>
<b>ACKDIS</b>	5	rwh	<b>Acknowledge Pulse Disable</b> 0 An acknowledge pulse is generated for each received frame. 1 No acknowledge pulse is generated. <i>Note: <b>ACKDIS</b> is automatically cleared by a stop condition.</i>
<b>INT</b>	6	rw	<b>Interrupt Delete Select</b> 0 Interrupt flag <b>IRQD</b> is deleted by a read/write to <b>RTB_LO.RTB(0,1)</b> or <b>RTB_HI.RTB(2,3)</b> . 1 Interrupt flag <b>IRQD</b> is not deleted by a read/write to <b>RTB_LO.RTB(0,1)</b> or <b>RTB_HI.RTB(2,3)</b> .
<b>TRX</b>	7	rwh	<b>Transmit Select</b> 0 No data is transmitted to the IIC bus. 1 Data is transmitted to the IIC bus. <i>Note: <b>TRX</b> is set automatically when writing to the transmit buffer. It is not allowed to delete this bit in the same buscycle. It is automatically cleared after last byte as slave transmitter.</i>
<b>IGE</b>	8	rw	<b>Ignore IRQE</b> Ignore <b>IRQE</b> (End of transmission) interrupt. 0 The IIC is stopped at <b>IRQE</b> interrupt. 1 The IIC ignores the <b>IRQE</b> interrupt. <i>Note: If <b>IIC_ST.RMEN</b> is set, <b>RM</b> is mirrored here.</i>
<b>RM</b>	15:8	rh	<b>Read Mirror</b> If <b>IIC_ST.RMEN</b> is set, <b>RTB_LO.RTB0</b> may be read here. Writing to <b>RM</b> has no effect in this mode.

**CONFIDENTIAL**

**I2C Bus Interface**

For **IIC\_ST.RMEN** = 0

**IIC\_CON**

**System Control Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WM EN	RESERVED			CI		STP	IGE	TRX	INT	ACK DIS	BUM	MOD		RSC	M10

Field	Bits	Type	Description
<b>M10</b>	0	rw	<b>Address Mode</b> 0 7-bit addressing using <b>IIC_ADR.ICA(7..1)</b> 1 10-bit addressing using <b>IIC_ADR.ICA(9..0)</b> .
<b>RSC</b>	1	rwh	<b>Repeated Start Condition</b> 0 No operation. 1 Generate a repeated start condition in (multi) master mode. <b>RSC</b> can not be set in slave mode.  <i>Note: <b>RSC</b> is cleared automatically after the repeated start condition has been sent.</i>
<b>MOD</b>	2:3	rw	<b>Basic Operating Mode</b> 00 IIC module is disabled and initialized (Init-Mode). Transmissions under execution was aborted. 01 Slave mode. 10 Master mode. 11 Multi-Master mode.
<b>BUM</b>	4	rwh	<b>Busy Master</b> 0 Clearing bit <b>BUM</b> ( $\overline{\text{L}}$ ) generates a stop condition immediately. 1 Setting bit <b>BUM</b> ( $\text{L}$ ) generates a start condition in (multi)master mode.  <i>Note: Setting bit <b>BUM</b> ( <math>\text{L}</math> ) while <b>BB</b> = 1 generates an arbitration lost situation.            In this case <b>BUM</b> is cleared and bit <b>AL</b> is set.  <b>BUM</b> can not be set in slave mode.</i>

**CONFIDENTIAL**

**I2C Bus Interface**

Field	Bits	Type	Description
<b>ACKDIS</b>	5	rwh	<b>Acknowledge Pulse Disable</b> 0 An acknowledge pulse is generated for each received frame. 1 No acknowledge pulse is generated. <i>Note: <b>ACKDIS</b> is automatically cleared by a stop condition.</i>
<b>INT</b>	6	rw	<b>Interrupt Delete Select</b> 0 Interrupt flag <b>IRQD</b> is deleted by a read/write to <b>RTB_LO.RTB(0,1)</b> or <b>RTB_HI.RTB(2,3)</b> . 1 Interrupt flag <b>IRQD</b> is not deleted by a read/write to <b>RTB_LO.RTB(0,1)</b> or <b>RTB_HI.RTB(2,3)</b> .
<b>TRX</b>	7	rwh	<b>Transmit Select</b> 0 No data is transmitted to the IIC bus. 1 Data is transmitted to the IIC bus. <i>Note: <b>TRX</b> is set automatically when writing to the transmit buffer. It is not allowed to delete this bit in the same buscycle. It is automatically cleared after last byte as slave transmitter.</i>
<b>IGE</b>	8	rw	<b>Ignore IRQE</b> Ignore <b>IRQE</b> (End of transmission) interrupt. 0 The IIC is stopped at <b>IRQE</b> interrupt. 1 The IIC ignores the <b>IRQE</b> interrupt. <i>Note: If <b>IIC_ST.RMEN</b> is set, <b>RM</b> is mirrored here.</i>
<b>STP</b>	9	rwh	<b>Stop Master</b> 0 Clearing bit <b>STP</b> generates no stop condition. 1 Setting bit <b>STP</b> generates a stop condition after next transmission. <b>BUM</b> is set to zero. <b>ACKDIS</b> is set to one. <i>Note: <b>STP</b> is automatically cleared by a stop condition. If <b>IIC_ST.RMEN</b> is set, <b>RM</b> is mirrored here.</i>

**CONFIDENTIAL**

**I2C Bus Interface**

Field	Bits	Type	Description
<b>CI</b>	11:10	rw	<b>Length of the Transmit Buffer</b> 00    1 Byte 01    2 Bytes 10    3 Bytes 11    4 Bytes <i>Note: If <b>IIC_ST.RMEN</b> is set, <b>RM</b> is mirrored here.</i>
<b>WMEN</b>	15	rwh	<b>Write Mirror Enable</b> 0      Write mirror is not active 1      Write mirror is active <i>If <b>IIC_ST.RMEN</b> is set, <b>WMEN</b> cannot be set and remains 0.</i> <i>If <b>WMEN</b> and <b>IIC_ST.RMEN</b> are simultaneously set to 1, both remain what they are, only one of them can be set to 1.</i>
<b>RESERVED</b>	14:12	r	If <b>IIC_ST.RMEN</b> is cleared, then this bitfield is reserved, these bits must be left at their reset values.

**CONFIDENTIAL**

**I2C Bus Interface**

For **IIC\_CON.WMEN** = 1

**IIC\_ST**

**System Status Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WM								IRQE	IRQP	IRQD	BB	LRB	SLA	AL	ADR

Field	Bits	Type	Description
<b>ADR</b>	0	rh	<b>Address</b> Bit <b>ADR</b> is set after a start condition in slave mode until the address has been received (1 byte in 7-bit address mode, 2 bytes in 10-bit address mode).
<b>AL</b>	1	rwh	<b>Arbitration Lost</b> Bit <b>AL</b> is set when the IIC module has tried to become master on the bus but has lost the arbitration. Operation is continued until the 9th clock pulse. If multimaster mode is selected the IIC module temporarily switches to slave mode after a lost arbitration. Bit <b>IRQP</b> is set along with bit <b>AL</b> . <b>AL</b> must be cleared via software.
<b>SLA</b>	2	rh	<b>Slave</b> 0 The IIC module is not selected as a slave, or the module is in master mode. 1 The IIC module has been selected as a slave (device address received).
<b>LRB</b>	3	rh	<b>Last Received Bit</b> Bit <b>LRB</b> represents the last bit (for example, the acknowledge bit) of the last transferred frame. It is automatically set to zero by a write or read access to buffers <b>RTB_LO.RTB(0,1)</b> and <b>RTB_HI.RTB(2,3)</b> . <i>Note: If LRB is high (no acknowledge) in slave mode, <b>TRX</b> bit is set automatically.</i>

**CONFIDENTIAL**

**I2C Bus Interface**

Field	Bits	Type	Description
<b>BB</b>	4	rh	<p><b>Bus Busy</b></p> <p>0 The IIC bus is idle, that is, a stop condition has occurred.</p> <p>1 The IIC bus is active, that is, a start condition has occurred.</p> <p><i>Note: Bit <b>BB</b> is always 0 while the IIC module is disabled.</i></p>
<b>IRQD</b>	5	rwh	<p><b>IIC Interrupt Request Bit for Data Transfer Events</b> <sup>1)</sup></p> <p>0 No interrupt request pending.</p> <p>1 A data transfer event interrupt request is pending.</p> <p><b>IRQD</b> is set after the acknowledge bit of the last byte has been received or transmitted, and is cleared automatically upon a complete read or write access to the buffers <b>RTB_LO.RTB(0,1)</b> and <b>RTB_HI.RTB(2,3)</b>.</p> <p>New data transfers will start immediately after clearing <b>IRQD</b>. Do not access any register until next interrupt.</p> <p>If a multi byte write could not be finished in slave mode because of missing acknowledge, then the data interrupt is followed by an end of transmission interrupt. The number of bytes sent can be read from <b>CO</b>. The data interrupt must have higher priority than <b>IRQE</b>.</p>

**CONFIDENTIAL**

**I2C Bus Interface**

Field	Bits	Type	Description
<b>IRQP</b>	6	rwh	<p><b>IIC Interrupt Request Bit for Protocol Events <sup>1)</sup></b></p> <p>0 No interrupt request pending.</p> <p>1 A protocol event interrupt request is pending.</p> <p><b>IRQP</b> is set when bit <b>SLA</b> or bit <b>AL</b> is set (✓), and must be cleared via software.</p> <p>If the IIC has been selected by an other master, the software must look up the required transmission direction by reading the received address and direction bit, stored in <b>RTB_LO.RTB0</b>. The <b>TRX</b> bit must be set by software correspondingly.</p>
<b>IRQE</b>	7	rwh	<p><b>IIC Interrupt Request Bit for Data Transmission End <sup>1)</sup></b></p> <p>0 No interrupt request pending.</p> <p>1 A receive end event interrupt request is pending (a stop is detected).</p> <p><b>IRQE</b> is automatically cleared upon a start condition. <b>IRQE</b> is not activated in init-mode. <b>IRQE</b> must always be deleted to continue transmission.</p> <p><i>Note: In slave mode <b>IRQE</b> is set after the transmission is finished. This can also be after a stop or RSC condition. In this case the slave is not selected any more. This bit is also set, if a transmission is stopped by a missing acknowledge. In this case the bit must be cleared by software.</i></p>
<b>WM</b>	15:8	wh	<p><b>Write Mirror</b></p> <p>If <b>IIC_CON.WMEN</b> is set, <b>RTB_LO.RTB0</b> may be written here. Reading <b>WM</b> will result in zero.</p>

**CONFIDENTIAL**

**I2C Bus Interface**

For **IIC\_CON.WMEN** = 0

**IIC\_ST**

**System Status Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RM EN</b>	<b>RESERVED</b>				<b>CO</b>		<b>IRQE</b>	<b>IRQP</b>	<b>IRQ D</b>	<b>BB</b>	<b>LRB</b>	<b>SLA</b>	<b>AL</b>	<b>ADR</b>	

Field	Bits	Type	Description
<b>ADR</b>	0	rh	<b>Address</b> Bit <b>ADR</b> is set after a start condition in slave mode until the address has been received (1 byte in 7-bit address mode, 2 bytes in 10-bit address mode).
<b>AL</b>	1	rwh	<b>Arbitration Lost</b> Bit <b>AL</b> is set when the IIC module has tried to become master on the bus but has lost the arbitration. Operation is continued until the 9th clock pulse. If multimaster mode is selected the IIC module temporarily switches to slave mode after a lost arbitration. Bit <b>IRQP</b> is set along with bit <b>AL</b> . <b>AL</b> must be cleared via software.
<b>SLA</b>	2	rh	<b>Slave</b> 0 The IIC module is not selected as a slave, or the module is in master mode. 1 The IIC module has been selected as a slave (device address received).
<b>LRB</b>	3	rh	<b>Last Received Bit</b> Bit <b>LRB</b> represents the last bit (for example, the acknowledge bit) of the last transferred frame. It is automatically set to zero by a write or read access to buffers <b>RTB_LO.RTB(0,1)</b> and <b>RTB_HI.RTB(2,3)</b> . <i>Note: If LRB is high (no acknowledge) in slave mode, <b>TRX</b> bit is set automatically.</i>



**CONFIDENTIAL**

**I2C Bus Interface**

Field	Bits	Type	Description
<b>BB</b>	4	rh	<p><b>Bus Busy</b></p> <p>0 The IIC bus is idle, that is, a stop condition has occurred.</p> <p>1 The IIC bus is active, that is, a start condition has occurred.</p> <p><i>Note: Bit <b>BB</b> is always 0 while the IIC module is disabled.</i></p>
<b>IRQD</b>	5	rwh	<p><b>IIC Interrupt Request Bit for Data Transfer Events <sup>1)</sup></b></p> <p>0 No interrupt request pending.</p> <p>1 A data transfer event interrupt request is pending.</p> <p><b>IRQD</b> is set after the acknowledge bit of the last byte has been received or transmitted, and is cleared automatically upon a complete read or write access to the buffers <b>RTB_LO.RTB(0,1)</b> and <b>RTB_HI.RTB(2,3)</b>.</p> <p>New data transfers will start immediately after clearing <b>IRQD</b>. Do not access any register until next interrupt.</p> <p>If a multi byte write could not be finished in slave mode because of missing acknowledge, then the data interrupt is followed by an end of transmission interrupt. The number of bytes sent can be read from <b>CO</b>. The data interrupt must have higher priority than <b>IRQE</b>.</p>
<b>IRQP</b>	6	rwh	<p><b>IIC Interrupt Request Bit for Protocol Events <sup>1)</sup></b></p> <p>0 No interrupt request pending.</p> <p>1 A protocol event interrupt request is pending.</p> <p><b>IRQP</b> is set when bit <b>SLA</b> or bit <b>AL</b> is set (✓), and must be cleared via software.</p> <p>If the IIC has been selected by an other master, the software must look up the required transmission direction by reading the received address and direction bit, stored in <b>RTB_LO.RTB0</b>. The <b>TRX</b> bit must be set by software correspondingly.</p>

**CONFIDENTIAL**

**I2C Bus Interface**

Field	Bits	Type	Description
IRQE	7	rwh	<p><b>IIC Interrupt Request Bit for Data Transmission End</b> <sup>1)</sup></p> <p>0 No interrupt request pending.</p> <p>1 A receive end event interrupt request is pending (a stop is detected).</p> <p><b>IRQE</b> is automatically cleared upon a start condition. <b>IRQE</b> is not activated in init-mode. IRQE must always be deleted to continue transmission.</p> <p><i>Note: In slave mode <b>IRQE</b> is set after the transmission is finished. This can also be after a stop or RSC condition. In this case the slave is not selected any more. This bit is also set, if a transmission is stopped by a missing acknowledge. In this case the bit must be cleared by software.</i></p>

**CONFIDENTIAL**

**I2C Bus Interface**

Field	Bits	Type	Description
<b>CO</b>	10:8	rh	<p><b>Counter of Transmitted Bytes Since Last Data Interrupt.</b></p> <p>If a multi byte transmission could not be finished because of missing acknowledge, the number of correctly transferred bytes can be read from <b>CO</b>. It is automatically set to zero by the correct number (defined by <b>CI</b>) of write/read accesses to the Receive Transmit Buffers <b>RTB_HI</b> and <b>RTB_LO</b>.</p> <p>000 No Bytes  001 1 Byte  010 2 Bytes  011 3 Bytes  100 4 Bytes</p> <p>The number of legal bytes depends on the data buffer size (<b>CI</b>). Writing to this bitfield does not affect its content.</p> <p>If <b>IIC_CON.WMEN</b> is set, <b>WM</b> is mirrored here.</p>
<b>RMEN</b>	15	rwh	<p><b>Read Mirror Enable</b></p> <p>0 Read mirror is not active  1 Read mirror is active</p> <p><i>Note: If <b>IIC_CON.WMEN</b> is set <b>RMEN</b> can not be set and will remain zero.</i></p> <p><i>If <b>RMEN</b> and <b>IIC_CON.WMEN</b> are set simultaneously to 1, both will remain what they are, only one of them can be set to 1.</i></p>
<b>RESERVED</b>	14:11	r	<p>If <b>IIC_CON.WMEN</b> is cleared, then this bitfield is reserved, these bits must be left at their reset values.</p>

- <sup>1)</sup> While either **IRQD**, **IRQP** or **IRQE** is set and the IIC module is in master mode or has been selected as a slave, the IIC clock line is held low which prevents further transfers on the IIC bus.  
The clock line of the IIC bus is released when **IRQD**, **IRQE** and **IRQP** are cleared. Only in this case the next IIC bus action can take place.  
Interrupt request bits may be set or cleared via software, for example, to control the IIC bus.

**CONFIDENTIAL**

**I2C Bus Interface**

### 11.1.4.4 IIC Bus Control Registers

For **IIC\_CON.M10** = 1

**IIC\_ADR**

**Bus Address Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>BRP MOD</b>	<b>PREDIV</b>	<b>RESERVED</b>				<b>ICA10</b>									

Field	Bits	Type	Description
<b>ICA10</b>	9:0	rw	<b>Node Address in 10-Bit Mode</b> <i>Note: Access is only possible in the 10-bit mode (IIC_CON.M10 = 1).</i>
<b>PREDIV</b>	14:13	rw	<b>Pre Divider for Baud Rate Generation</b> 00 Pre-divider is disabled 01 Pre-divider factor 8 is enabled 10 Pre-divider factor 64 is enabled 11 Reserved, do not use <i>Note: Refer to Table 11-2 on page 971.</i>
<b>BRPMOD</b>	15	rw	Baud Rate Prescaler Mode 0 Mode 0 is enabled (by default) 1 Mode 1 is enabled. <i>Note: Refer to Table 11-2 IIC-Bus Baud Rate Selection for BRPMOD = 0 (on Page 971).</i>
<b>RESERVED</b>	12:10	r	Reserved, these bits must be left at their reset values.

**CONFIDENTIAL**

**I2C Bus Interface**

For **IIC\_CON.M10** = 0

### IIC\_ADR

**Bus Address Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>BRP MOD</b>	<b>PREDIV</b>	<b>RESERVED</b>						<b>ICA7</b>						<b>RESERVED</b>	

Field	Bits	Type	Description
<b>ICA7</b>	7:1	rw	<b>Node Address in 7-Bit Mode</b> <i>Note: Access is limited to this bitfield in the 7-bit mode (<b>IIC_CON.M10</b> = 0).</i>
<b>PREDIV</b>	14:13	rw	<b>Pre Divider for Baud Rate Generation</b> 00 Pre-divider is disabled 01 Pre-divider factor 8 is enabled 10 Pre-divider factor 64 is enabled 11 Reserved, do not use <i>Note: See <a href="#">Table 11-2 IIC-Bus Baud Rate Selection for BRPMOD = 0 (on Page 971)</a>.</i>
<b>BRPMOD</b>	15	rw	<b>Baud Rate Prescaler Mode</b> 0 Mode 0 is enabled (by default) 1 Mode 1 is enabled. <i>Note: See <a href="#">Table 11-2</a>.</i>
<b>RESERVED</b>	12:8	r	Reserved, these bits must be left at their reset values.

**CONFIDENTIAL**

**I2C Bus Interface**

## IIC\_CFG

### Bus Configuration Register

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRP								RESERVED			SCL EN	RESERVED			SDA EN

Field	Bits	Type	Description
<b>SDAEN</b>	0	rw	<b>Enable Input for Data Pin</b> These bits determine if the SDA pin for the IIC data line is connected. 0 SDA pin is disconnected. 1 SDA pin is connected to IIC data line.
<b>SCLEN</b>	4	rw	<b>Enable Input for Clock Pin</b> These bits determine if the SCL pin for the IIC clock line is connected. 0 SCL pin is disconnected. 1 SCL pin is connected to IIC clock line.
<b>BRP</b>	15:8	rw	<b>Baud Rate Prescaler</b> Determines the baud rate for the IIC channel(s). The prescaler may operate in two modes. Bit <b>IIC_CON.BRPMOD</b> selects the actual mode. Bit field <b>IIC_CON.PREDIV</b> selects an additional predivider. <i>Note: Refer to <a href="#">Table 11-2 IIC-Bus Baud Rate Selection for BRPMOD = 0 (on Page 971)</a>.</i>
<b>RESERVED</b>	7:5, 3:1	r	Reserved, these bits must be left at their reset values.

### 11.1.4.4.1 Baud Rate Selection

To give the user high flexibility in selection of CPU frequency and baud rate without constraints to baud rate accuracy, a flexible baud rate generator has been implemented. It uses two different modes (**IIC\_ADR.BRPMOD**) and an additional predivider (**IIC\_ADR.PREDIV**). Low baud rates may be configured at high precision in mode 0 which is compatible with older versions. High baud rates may be configured precisely in mode 1.

#### Mode 0: Reciprocal Divider

The resulting baud rate is (refer to **IIC\_CFG.BRP**):

If **PREDIV** = 0:

$$B_{IIC} = \frac{f_{IIC}}{4 \cdot (BRP + 1)} \quad BRP = \frac{f_{IIC}}{4 \cdot B_{IIC}} - 1 \quad [36.14]$$

If **PREDIV** != 0:

$$B_{IIC} = \frac{f_{IIC}}{4 \cdot (BRP + 1)} \cdot \frac{1}{PREDIV} \quad BRP = \frac{f_{IIC}}{4 \cdot B_{IIC} \cdot PREDIV} - 1 \quad [36.15]$$

#### Mode 1: Fractional Divider

The resulting baud rate is:

$$B_{IIC} = \frac{f_{IIC} \cdot BRP}{1024} \cdot \frac{1}{PREDIV} \quad BRP = \frac{1024 \cdot B_{IIC} \cdot PREDIV}{f_{cpu}} \quad [36.16]$$

**Table 11-2** gives Baud Rate Selection.

**Table 11-2 IIC-Bus Baud Rate Selection for BRPMOD = 0**

<b>BRPMOD = 0</b> $f_{cpu}$ [MHz]	<b>BRP @ 100kBaud</b>		<b>BRP @ 400kBaud</b>	
	<b>PREDIV = 00<sub>B</sub></b>	<b>PREDIV = 01<sub>B</sub></b>	<b>PREDIV = 00<sub>B</sub></b>	<b>PREDIV = 01<sub>B</sub></b>
100	F9 <sub>H</sub>	1E <sub>H</sub>	3E <sub>H</sub>	07 <sub>H</sub>
50	7C <sub>H</sub>	0F <sub>H</sub>	1E <sub>H</sub>	03 <sub>H</sub>
24	3B <sub>H</sub>	06 <sub>H</sub>	0E <sub>H</sub>	-
20	31 <sub>H</sub>	05 <sub>H</sub>	0C <sub>H</sub>	-
16	27 <sub>H</sub>	04 <sub>H</sub>	09 <sub>H</sub>	-

**Table 11-3 IIC-Bus Baud Rate Selection for BRPMOD = 1**

<b>BRPMOD = 1</b>	<b>BRP @ 100kBaud</b>			<b>BRP @ 400kBaud</b>		
$f_{cpu}$ [MHz]	PREDIV= 00 <sub>B</sub>	PREDIV= 01 <sub>B</sub>	PREDIV= 10 <sub>B</sub>	PREDIV= 00 <sub>B</sub>	PREDIV= 01 <sub>B</sub>	PREDIV= 10 <sub>B</sub>
100	-	-	42 <sub>H</sub>	-	21 <sub>H</sub>	-
50	-	-	83 <sub>H</sub>	-	42 <sub>H</sub>	-
24	-	22 <sub>H</sub>	-	-	89 <sub>H</sub>	-
20	-	29 <sub>H</sub>	-	-	A4 <sub>H</sub>	-
16	-	33 <sub>H</sub>	-	-	CD <sub>H</sub>	-

#### 11.1.4.5 IIC Receive Transmit Buffers

##### RTB\_HI

High Receive Transmit Buffer

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTB3								RTB2							

##### RTB\_LO

Low Receive Transmit Buffer

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTB1								RTB0							

Field	Bits	Type	Description
<b>RTBx</b> (x = 0 to 3)	31:0	rwh	<b>Receive/Transmit Buffer <sup>1)</sup></b> The buffers contain the data to be sent/received. The buffer size can be set in bitfield <b>IIC_CON.CI</b> (from 1 up to 4 bytes). <b>RTB0</b> is sent/received first.

<sup>1)</sup> If bit **IIC\_CON.INT** is set to zero and all bytes (specified in **IIC\_CON.CI**) of **RTB0..3** are read/written (depending on bit **IIC\_CON.TRX**), **IIC\_ST.IRQD** is cleared by hardware after completion of this access.

#### 11.1.5 Reset Behavior

All resets are handled asynchronously. All registers are reset to their reset values.



**CONFIDENTIAL**

**I2C Bus Interface**

### 11.1.6 Interrupts

**Table 11-4 Interrupt Sources**

Interrupt	Signal	Description
Data	IC_INT_D_O	Interrupt is requested after the acknowledge bit of the last byte has been received or transmitted.
Data Error	IC_INT_D_O	Interrupt is requested if a multi byte write could not be finished in slave mode because of missing acknowledge, then the data interrupt is followed by an end of transmission interrupt.
Protocol: Arbitration Lost	IIC_INT_P_O	Interrupt is requested when the IIC module has tried to become master on the bus but has lost the arbitration.
Protocol: Slave Mode after Lost Arbitration	IIC_INT_P_O	Interrupt is requested if multimaster mode is selected and the IIC module temporarily switches to slave mode after a lost arbitration.
Protocol: Slave Mode after Device Address	IIC_INT_P_O	Interrupt is requested if multimaster mode is selected and the IIC module temporarily switches to slave mode after a lost arbitration.
Data Transmission End after Stop Condition	IIC_INT_E_O	Interrupt is requested after transmission is finished by a stop condition.
Data Transmission End after RSC Condition	IIC_INT_E_O	Interrupt is requested after transmission is finished by a repeated start condition (RSC).
Data Transmission End after missing Acknowledge	IIC_INT_E_O	Interrupt is also requested if a transmission is stopped by a missing acknowledge.

CONFIDENTIAL

I2C Bus Interface

### 11.1.7 Synchronization

In the Master mode, the SCL line is controlled by the IIC Module. Sent and received data is only valid if SCL is high. With SCL going down, all modules are starting to count down their low period. During the low period all connected modules are allowed to hold SCL low. As the physical bus connection is wired-AND, SCL remains low until the device with the longest low period enters the high state. Then the device with the shortest high period pulls SCL low again.

### 11.1.8 Programming

It is strictly recommended not to write to the IIC registers (except for interrupt handling) when the IIC is working. This is indicated by the **IIC\_CON.BUM** bit (in the master mode) and the interrupt flags. In the initial mode all registers can be written. In the master mode the IIC is working as long as bit **IIC\_CON.BUM** is set, in the slave mode the IIC is working from receiving a start condition until receiving the next stop condition. Change of transmit direction is possible only after a protocol interrupt (IRQP) or in the initialization mode (MOD = 00<sub>b</sub>).

#### 11.1.8.1 Initialization

Before data can be sent or received, the data buffer size must be set in the bit field **IIC\_CON.CI** (this is only necessary if buffer greater than one byte is available). To decide if the slave/master or multimaster mode is required, the **IIC\_CON.MOD** bits must be programmed.

#### 11.1.8.2 Repeated Start Condition

**IIC\_CON.RSC** must be set to one.

#### 11.1.8.3 Start Condition

To generate a start condition, the IIC must be in the master mode. If **IIC\_CON.BUM** is set, a start condition is sent and the transmission is started. The slave returns the acknowledge bit that is indicated by **IIC\_ST.LRB**.

#### 11.1.8.4 Sending Data Bytes

To sent bytes it is only necessary to write data bytes to the transmit buffer every time a data interrupt (**IIC\_ST.IRQD**) occurs.

#### 11.1.8.5 Stop Condition

To stop the transmission, the **IIC\_CON.BUM** bit must be set to zero, or the **IIC\_CON.STP** bit must be set to one.

#### 11.1.8.6 Receiving Data Bytes

To receive bytes it is necessary to set the **IIC\_CON.TRX** bit to zero. The bytes can be read after every data interrupt (**IIC\_ST.IRQD**). After a stop condition (protocol interrupt **IIC\_ST.IRQE**), the count bit field **IIC\_ST.CO** must be read (in case the buffer size, as defined in **IIC\_CON.CI**, is greater than one byte to decide which bytes in the receive buffer were received in the last transmission cycle).

CONFIDENTIAL

CONFIDENTIAL

**CONFIDENTIAL**

**Synchronous Serial Interface on PD Bus**

## 11.2 Synchronous Serial Interface on PD Bus

History	
Design Spec.	Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.01	
<b>Page 977</b>	Updated <b>System Integration</b> : WS00006682
Changes for Rev. 1.02	
<b>Page 977</b>	Updated <b>Section 11.2.1 Introduction</b> WS00007170
Changes for Rev. 1.03	
all	Updated for 78MHz WS00007170
Changes for Rev. 1.06	

### System Integration:

- Supply domain: VDD\_MAIN
- Chip internal interfaces:
  - Clock domain: gclk\_dsp\_per drives the internal module clock ssc1\_clk
  - Bus domain: TEAKLite Z-Bus
  - Interrupt sources: SSC\_TX\_INT, SSC\_RX\_INT, SSC\_ERR\_INT
- Monitor Pins: Refer to **Section 11.8.10 Internal Signal Monitoring (on Page 1209)**

### 11.2.1 Introduction

The Synchronous Serial Interface (SSC) supports both duplex and half-duplex serial synchronous communication up to 26 Mbps (@ 52 MHz module clock) and up to 39 Mbps (@78 Mhz master clock). The serial clock signal can be generated by the SSC itself (Master Mode) or can be received from an external master (Slave Mode). Data width, shift direction, clock polarity, and phase are programmable. This allows communication with SPI-compatible devices. Transmission and reception of data is double-buffered. A 16-bit baudrate generator provides the SSC with a separate serial clock signal.

**CONFIDENTIAL**

**Synchronous Serial Interface on PD Bus**

**Features:**

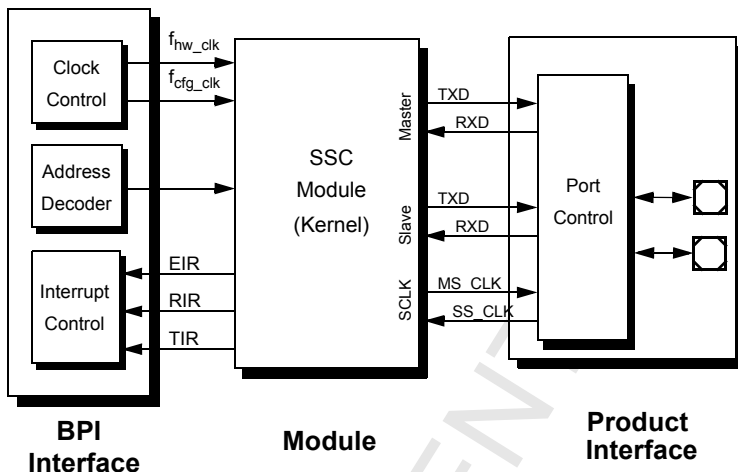
- Master and Slave Mode operation
  - Duplex or half-duplex operation
- Flexible data format
  - Programmable number of data bits: 2 to 16 bits
  - Programmable shift direction: LSB or MSB shift first
  - Programmable clock polarity: idle low or high state for the shift clock
  - Programmable clock/data phase: data shift with leading or trailing edge of the shift clock
- Baudrate generation from 396.72 up to 26 MBaud (@ 52 MHz module clock) and up to 39 MBaud (@ 78 MHz module clock)
- Interrupt generation
  - On a transmitter empty condition
  - On a receiver full condition
  - On an error condition (receive, phase, baudrate, transmit error)
- FIFO
  - Up to 32 stage receive FIFO (RXFIFO)
  - Up to 32 stage transmit FIFO (TXFIFO)
  - Independent control of RXFIFO and TXFIFO
  - 16-Bit FIFO data width
  - Programmable Receive/Transmit Interrupt Trigger Level
  - Receive and transmit FIFO filling level indication
  - Overrun error generation
  - Underflow error generation

CONFIDENTIAL

Synchronous Serial Interface on PD Bus

Figure 11-3 shows all functional relevant interfaces associated with the SSC Kernel.

Figure 11-3 SSC Interface Diagram<sup>1)</sup>



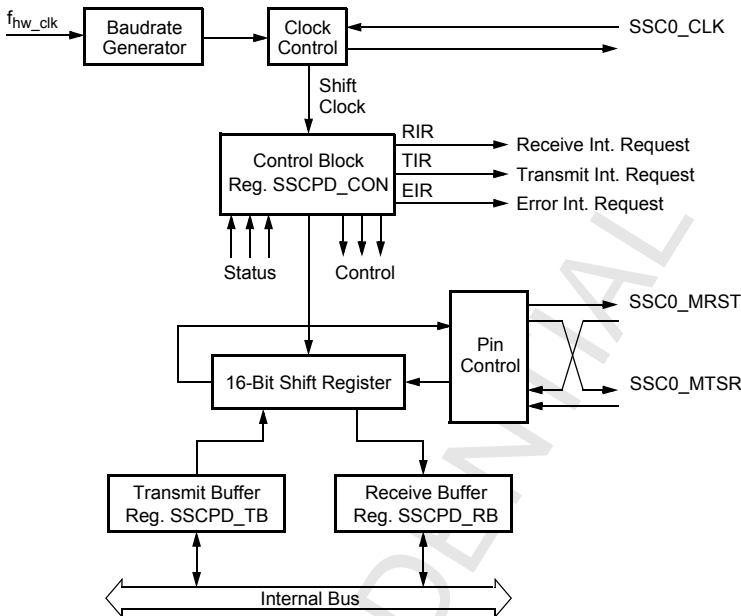
### 11.2.2 General Operation

The SSC supports duplex and half-duplex synchronous communication up to 26 MBaud (@ 52 MHz module clock) and up to 39 MBaud (@ 78 MHz module clock). The serial clock signal can be generated by the SSC itself (Master Mode) or can be received from an external master (Slave Mode). Data width, shift direction, clock polarity, and phase are programmable. This allows communication with SPI-compatible devices. Transmission and reception of data is double-buffered. A 16-bit baudrate generator provides the SSC with a separate serial clock signal.

The high-speed synchronous serial interface can be configured in a very flexible way, so it can be used with other synchronous serial interfaces, can serve for master/slave or multimaster interconnections or can operate compatible with the popular SPI interface. Thus, the SSC can be used to communicate with shift registers (IO expansion), peripherals (EEPROMs, etc.) or other controllers (networking). The SSC supports half-duplex and duplex communication. Data is transmitted or received on lines TXD and RXD, normally connected with pins MTSR (Master Transmit/Slave Receive) and MRST (Master Receive/Slave Transmit). The clock signal is output via line MS\_CLK (Master Serial Shift Clock) or input via line SS\_CLK (Slave Serial Shift Clock). Both lines are normally connected to pin SCLK. These pins are alternate functions of port pins.

<sup>1)</sup> For information about hw\_clk, refer to [Table 11-7 SSC Register Summary \(on Page 995\)](#).

Figure 11-4 Synchronous Serial Channel SSC Block Diagram



### 11.2.2.1 Operating Mode Selection

The operating mode of the serial channel SSC is controlled by its control register **SSCPD\_CON**. This register serves two purposes:

- During programming (SSC disabled by **SSCPD\_CON.EN** = 0), it provides access to a set of control bits
- During operation (SSC enabled by **SSCPD\_CON.EN** = 1), it provides access to a set of status flags.

The shift register of the SSC is connected to both the transmit lines and the receive lines via the pin control logic (see block diagram in [Figure 11-4](#)). Transmission and reception of serial data are synchronized and take place at the same time. That is, the same number of transmitted bits is also received. Transmit data is written into the Transmit Buffer Register (**SSCPD\_TB**). It is moved to the shift register as soon as it is empty. A SSC master (**SSCPD\_CON.MS** = 1) immediately begins transmitting, while an SSC slave

(**MS** = 0) will wait for an active shift clock. When the transfer starts, the busy flag (**SSCPD\_CON.BSY**) is set and the Transmit Interrupt Request line TIR is activated to indicate that register **SSCPD\_TB** may be reloaded again.



**CONFIDENTIAL**

**Synchronous Serial Interface on PD Bus**

When the programmed number of bits (2...16) has been transferred, the content of the shift register is moved to the Receive Buffer Register, (**SSCPD\_RB**), and the Receive Interrupt Request line RIR is activated. If no further transfer takes place (**SSCPD\_TB** is empty), **BSY** is cleared at the same time. Software should not modify **BSY** as it is hardware controlled.

*Note: Only one SSC can be master at a given time.*

The transfer of serial data bits can be programmed in many ways:

- Data width can be specified from 2 bits to 16 bits
- Transfer may start with either the LSB or the MSB
- Shift clock may be idle low or idle high
- Data bits may be shifted with the leading edge or the trailing edge of the shift clock signal
- Baudrate may be set from 396.72 Baud up to 26 MBaud (@ 52 MHz module clock) and up to 39 MBaud (@ 78 MHz module clock)
- Shift clock can be generated (MS\_CLK) or can be received (SS\_CLK)

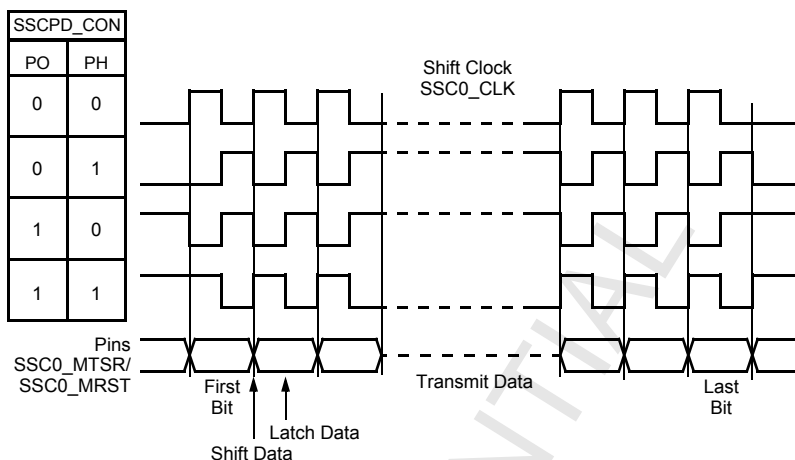
These features allow the adaptation of the SSC to a wide range of applications in which serial data transfer is required.

Data Width Selection supports the transfer of frames of any data length, from 2-bit "characters" up to 16-bit "characters". Starting with the LSB (**SSCPD\_CON.HB** = 0) allows communication with SSC devices in Synchronous Mode or with 8051 like serial interfaces for example. Starting with the MSB (**HB** = 1) allows operation compatible with the SPI interface.

Regardless of the data width selected and whether the MSB or the LSB is transmitted first, the transfer data is always right-aligned in registers **SSCPD\_TB** and **SSCPD\_RB**, with the LSB of the transfer data in bit 0 of these registers. The data bits are rearranged for transfer by the internal shift register logic. The unselected bits of **SSCPD\_TB** are ignored; the unselected bits of **SSCPD\_RB** will not be valid and should be ignored by the receiver service routine.

The Clock Control allows the adaptation of transmit and receive behavior of the SSC to a variety of serial interfaces. A specific shift clock edge (rising or falling) is used to shift out transmit data, while the other shift clock edge is used to latch in receive data. Bit **SSCPD\_CON.PH** selects the leading edge or the trailing edge for each function. Bit **SSCPD\_CON.PO** selects the level of the shift clock line in the idle state. Thus, for an idle-high clock, the leading edge is a falling one, a 1-to-0 transition (see **Figure 11-5**).

**Figure 11-5 Serial Clock Phase and Polarity Options**



### 11.2.2.2 Duplex Operation

The various devices are connected through three lines. The definition of these lines is always determined by the master: The line connected to the master's data output line TXD is the transmit line; the receive line is connected to its data input line RXD; the shift clock line is either MS\_CLK or SS\_CLK. Only the device selected for master operation generates and outputs the shift clock on line MS\_CLK. All slaves receive this clock; therefore, their pin SCLK must be switched to input mode. The output of the master's shift register is connected to the external transmit line, which in turn is connected to the slaves' shift register input. The output of the slaves' shift register is connected to the external receive line in order to enable the master to receive the data shifted out of the slave. The external connections are hard-wired, the function and direction of these pins is determined by the master or slave operation of the individual device.

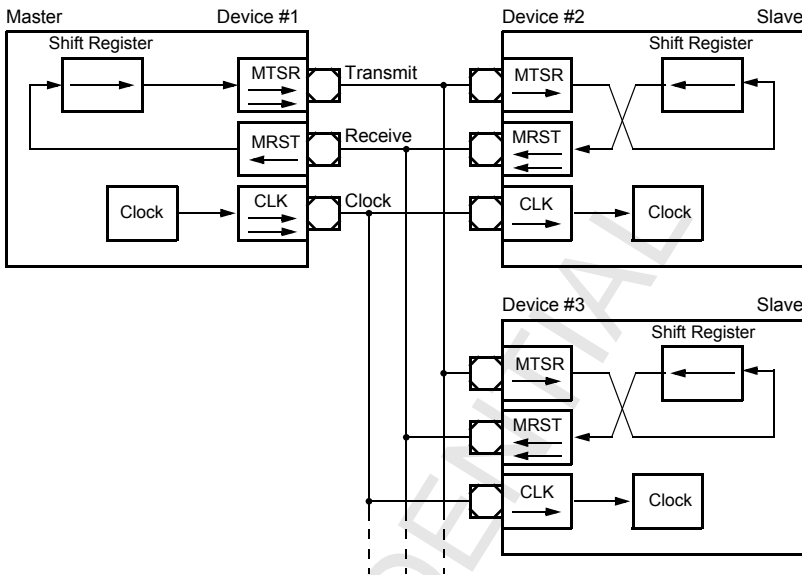
*Note: The shift direction shown in the figure applies for MSB-first operation as well as for LSB-first operation.*

When initializing the devices in this configuration, one device must be selected for master operation while all other devices must be programmed for slave operation. Initialization includes the operating mode of the device SSC and also the function of the respective port lines.

CONFIDENTIAL

Synchronous Serial Interface on PD Bus

Figure 11-6 SSC Duplex Configuration



The data output pins MRST of all slave devices are connected together onto the one receive line in the configuration shown in [Figure 11-6](#). During a transfer each slave shifts out data from its shift register. There are two ways to avoid collisions on the receive line due to different slave data:

- Only one slave drives the line, that is, enables the driver of its MRST pin. All the other slaves must have their MRST pins programmed as input so only one slave can put its data onto the master's receive line. Only receiving data from the master is possible. The master selects the slave device from which it expects data either by separate select lines, or by sending a special command to this slave. The selected slave then switches its MRST line to output until it gets a de-selection signal or command.
- The slaves use open drain output on MRST. This forms a wired-AND connection. The receive line needs an external pull-up in this case. Corruption of the data on the receive line sent by the selected slave is avoided when all slaves not selected for transmission to the master only send ones (1s). Because this high level is not actively driven onto the line, but only held through the pull-up device, the selected slave can pull this line actively to a low level when transmitting a zero bit. The master selects the slave device from which it expects data either by separate select lines or by sending a special command to this slave.

After performing the necessary initialization of the SSC, the serial interfaces can be enabled. For a master device, the alternate clock line will now go to its programmed

**CONFIDENTIAL****Synchronous Serial Interface on PD Bus**

polarity. The alternate data line will go to either 0 or 1 until the first transfer will start. After a transfer, the alternate data line will always remain at the logic level of the last transmitted data bit.

When the serial interfaces are enabled, the master device can initiate the first data transfer by writing the transmit data into register **SSCPD\_TB**. This value is copied into the shift register (assumed to be empty at this time), and the selected first bit of the transmit data will be placed onto the TXD line on the next clock from the baudrate generator (transmission starts only if **SSCPD\_CON.EN** = 1). Depending on the selected clock phase, a clock pulse will also be generated on the MS\_CLK line. At the same time, with the opposite clock edge, the master latches and shifts in the data detected at its input line RXD. This "exchanges" the transmit data with the receive data. Because the clock line is connected to all slaves, their shift registers will be shifted synchronously with the master's shift register—shifting out the data contained in the registers, and shifting in the data detected at the input line. After the preprogrammed number of clock pulses (via the data width selection), the data transmitted by the master is contained in all the slaves' shift registers, while the master's shift register holds the data of the selected slave. In the master and all slaves, the content of the shift register are copied into the receive buffer RB and the receive interrupt line RIR is activated.

A slave device will immediately output the selected first bit (MSB or LSB of the transfer data) at line RXD when the contents of the transmit buffer are copied into the slave's shift register. Bit **SSCPD\_CON.BSY** is not set until the first clock edge at SS\_CLK appears. The slave device will not wait for the next clock from the baudrate generator, as the master does. The reason for this is that, depending on the selected clock phase, the first clock edge generated by the master may already be used to clock in the first data bit. Thus, the slave's first data bit must already be valid at this time.

*Note: On the SSC, a transmission **and** a reception takes place at the same time, regardless of whether valid data has been transmitted or received.*

### **11.2.2.3 Half-Duplex Operation**

In a Half-Duplex Mode, only one data line is necessary for both receiving **and** transmitting of data. The data exchange line is connected to both the MTSR and MRST pins of each device, the shift clock line is connected to the SCLK pin.

The master device controls the data transfer by generating the shift clock, while the slave devices receive it. Due to the fact that all transmit and receive pins are connected to the one data exchange line, serial data may be moved between arbitrary stations.

Similar to the Duplex Mode, there are two ways to avoid collisions on the data exchange line:

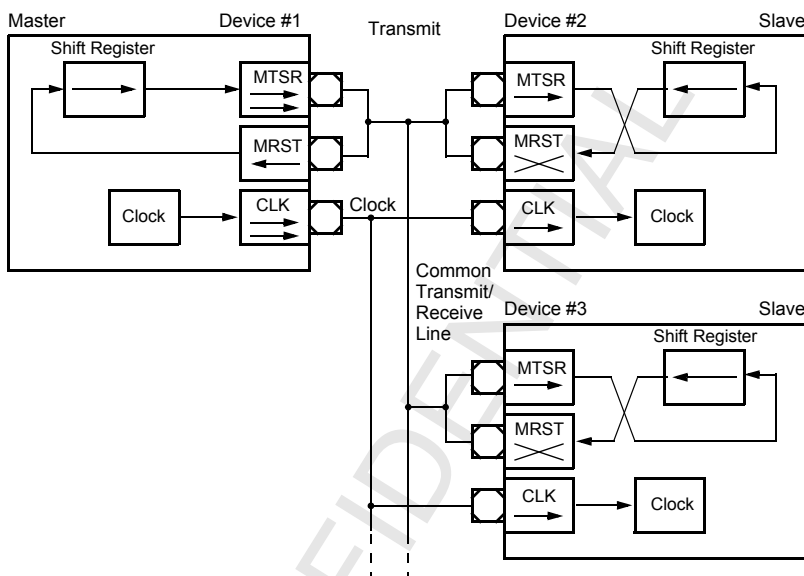
- only the transmitting device may enable its transmit pin driver
- the non-transmitting devices use open drain output and send only ones.

CONFIDENTIAL

Synchronous Serial Interface on PD Bus

Because the data inputs and outputs are connected together, a transmitting device will clock in its own data at the input pin (MRST for a master device, MTSR for a slave). By this method, any corruptions on the common data exchange line are detected if the received data is not equal to the transmitted data.

**Figure 11-7 SSC Half-Duplex Configuration**



### 11.2.2.4 Continuous Transfers

When the transmit interrupt request flag is set, it indicates that the transmit buffer **SSCPD\_TB** is empty and ready to be loaded with the next transmit data. If **SSCPD\_TB** has been reloaded by the time the current transmission is finished, the data is immediately transferred to the shift register and the next transmission will start without any additional delay. On the data line, there is no gap between the two successive frames. For example, two byte transfers would look the same as one word transfer. This feature can be used to interface with devices that can operate with or require more than 16 data bits per transfer. It is just a matter of software, how long a total data frame length can be. This option can also be used to interface to byte-wide and word-wide devices on the same serial bus, for instance.

*Note: Of course, this can only happen in multiples of the selected basic data width, because it would require disabling/enabling of the SSC to reprogram the basic data width on-the-fly.*

### 11.2.2.5 FIFO Operations

#### 11.2.2.5.1 Transmit FIFO

The transmit FIFO (TXFIFO) provides the following functionality:

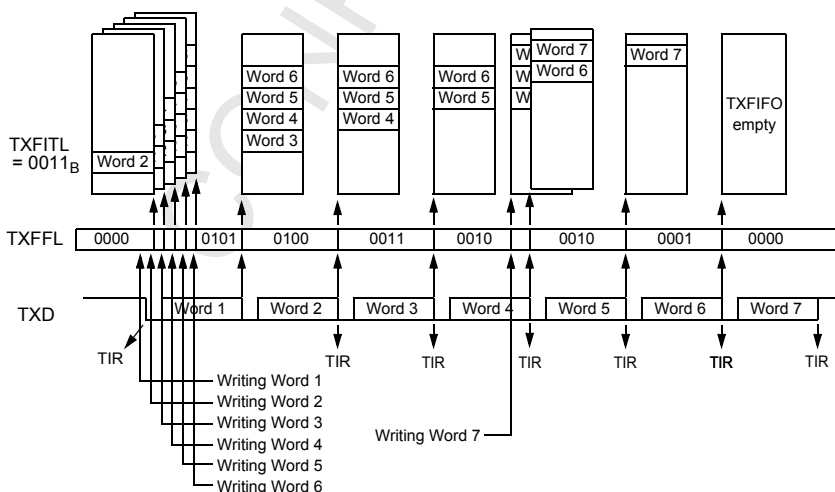
- Enable/disable control
- Programmable filling level for transmit interrupt generation
- Filling level indication
- FIFO clear (flush) operation
- FIFO overflow error generation

The 32 stage transmit FIFO is controlled by the **SSCPD\_TXFCON** control register. When bit **SSCPD\_TXFCON.TXFEN** is set, the transmit FIFO is enabled. The interrupt trigger level defined by **SSCPD\_TXFCON.TXFITL** defines the filling level of the TXFIFO at which a transmit interrupt TIR is generated. These interrupt is always generated when the filling level of the transmit FIFO is equal to or less than the value stored in **SSCPD\_TXFCON.TXFITL**.

Bit field **SSCPD\_FSTAT.TXFFL** indicates the number of entries that are actually written (valid) in the TXFIFO. Therefore, the software can verify, in the interrupt service routine, for instance, how many bytes can be still written into the transmit FIFO via register **SSCPD\_TB** without getting an overrun error.

The transmit FIFO cannot be accessed directly. All data write operations into the TXFIFO are executed by writing into the **SSCPD\_TB** register.

### Figure 11-8 Transmit FIFO Operation Example



**CONFIDENTIAL**

**Synchronous Serial Interface on PD Bus**

The example in **Figure 11-8** shows a typical transmit FIFO operation. In this example seven messages are transmitted via the TXD output line. The transmit FIFO interrupt trigger level **SSCPD\_TXFCON.TXFITL** is set to 0011<sub>B</sub>. The first message data written into the empty TXFIFO via **SSCPD\_TB** is directly transferred into the transmit shift register and is not written into the FIFO. After message 1, messages 2 to 6 are written into the transmit FIFO.

When message 3 is transferred from TXFIFO into the SSC shift register, 3 other messages remain in the TXFIFO. **SSCPD\_TXFCON.TXFITL** is then reached (= 3) and a transmit interrupt (TIR) is generated - at the beginning of the serial transmission. During the serial transmission of message 4, another message (message 7) is written into the TXFIFO (**SSCPD\_TB** write operation). Finally, after the start of the serial transmission of message 7, the TXFIFO is again empty (and a TIR is raised).

If the TXFIFO is full and additional bytes are written into **SSCPD\_TB**, the Error interrupt [EIR] will be generated with bit **SSCPD\_CON.TE** set if bit **SSCPD\_CON.TEN** was set. In this case, the data that was last written into the transmit FIFO is overwritten and the transmit FIFO filling level **SSCPD\_FSTAT.TXFFL** is set to maximum.

The TXFIFO can be flushed or cleared by setting bit **SSCPD\_TXFCON.TXFFLU**. After this TXFIFO flush operation, the TXFIFO is empty and the transmit FIFO filling level **SSCPD\_FSTAT.TXFFL** is set to 000000<sub>B</sub>. A running serial transmission is not aborted by a receive FIFO flush operation

*Note: The TXFIFO is flushed automatically with a reset operation of the SSC module and if the TXFIFO becomes disabled (resetting bit **SSCPD\_TXFCON.TXFEN**) after it was previously enabled.*

### **11.2.2.5.2 Receive FIFO Operation**

The receive FIFO (RXFIFO) provides the following functionality:

- Enable/disable control
- Programmable filling level for receive interrupt generation
- Filling level indication
- FIFO clear (flush) operation
- FIFO overflow error generation

The 32 stage receive FIFO is controlled by the **SSCPD\_RXFCON** control register. When bit **SSCPD\_RXFCON.RXFEN** is set, the receive FIFO is enabled. The interrupt trigger level defined by **SSCPD\_RXFCON.RXFITL** defines the filling level of RXFIFO at which a receive interrupt RIR is generated. RIR is always generated when the filling level of the receive FIFO is equal to or greater than the value stored in **SSCPD\_RXFCON.RXFITL**.

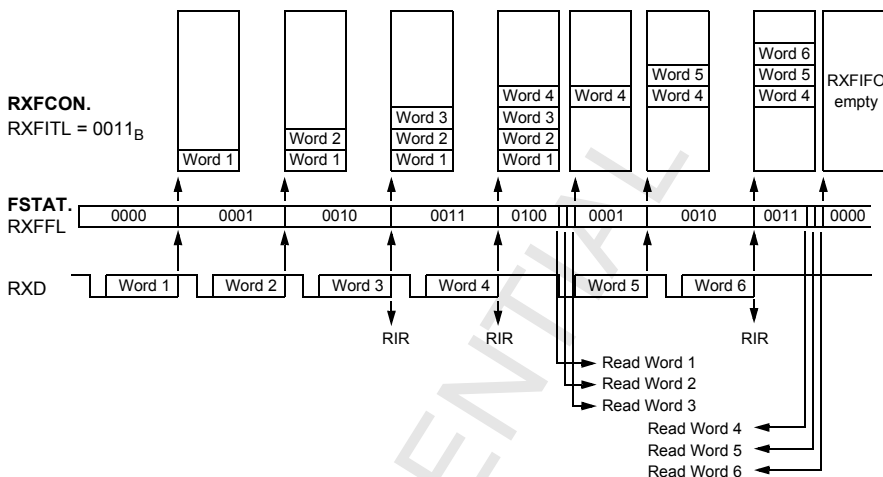
Bit field **SSCPD\_FSTAT.RXFFL** indicates the number of bytes that have been actually written into the FIFO and can be read out of the FIFO by a user program.

CONFIDENTIAL

Synchronous Serial Interface on PD Bus

The receive FIFO cannot be accessed directly. All data read operations from the RXFIFO are executed by reading the **SSCPD\_RB** register.

Figure 11-9 Receive FIFO Operation Example



The example in **Figure 11-9** shows a typical receive FIFO operation. In this example, six messages are received via the RXD input line. The receive FIFO interrupt trigger level **SSCPD\_RXFCON.RXFCTL** is set to 00011<sub>B</sub>. Therefore, the first receive interrupt RIR is generated after the reception of message 3 (RXFIFO is filled with three messages).

After the reception of message 4, three messages are read out of the receive FIFO. After this read operation, the RXFIFO still contains one message. RIR becomes again active after two more messages (messages 5 and 6) have been received (RXFIFO filled again with 3 messages). Finally, the FIFO is cleared after three read operations.

If the RXFIFO is full and additional data are received, the Error interrupt [EIR] is generated and bit **SSCPD\_CON.RE** is set if **SSCPD\_CON.REN** is not cleared. In this case, the data byte last written into the receive FIFO is overwritten. With the overrun condition, the receive FIFO filling level **SSCPD\_FSTAT.RXFFL** is set to maximum. If a **SSCPD\_RB** read operation is executed with the RXFIFO enabled but empty, a receive interrupt RIR will be generated. In this case, the receive FIFO filling level **SSCPD\_FSTAT.RXFFL** is set to 000000<sub>B</sub>.

If the RXFIFO is available but disabled (**SSCPD\_RXFCON.RXFEN** = 0) the receive operation is functionally equivalent to the receive operation of the SSC module without FIFO.

The RXFIFO can be flushed or cleared by setting bit **SSCPD\_RXFCON.RXFFLU**. After this RXFIFO flush operation, the RXFIFO is empty and the receive FIFO filling level **SSCPD\_FSTAT.RXFFL** is set to 000000<sub>B</sub>.



**CONFIDENTIAL**

**Synchronous Serial Interface on PD Bus**

The RXFIFO is flushed automatically with a reset operation of the SSC module and if the RXFIFO becomes disabled (resetting bit **SSCPD\_RXFCON.RXFEN**) after it was previously enabled. Resetting bit **SSCPD\_CON.REN** without resetting **SSCPD\_RXFCON.RXFEN** does not affect (reset) the RXFIFO state. This means that the receive operation of the SSC is stopped, in this case, without changing the content of the RXFIFO. After setting **SSCPD\_CON.REN** again, the RXFIFO with its content is again available.

### 11.2.2.5.3 FIFO Transparent Mode

In Transparent Mode, a specific interrupt generation mechanism is used for receive and transmit interrupts. In general, in Transparent Mode, receive interrupts are always generated if data bytes are available in the RXFIFO. The relevant conditions for interrupt generation in Transparent Mode are:

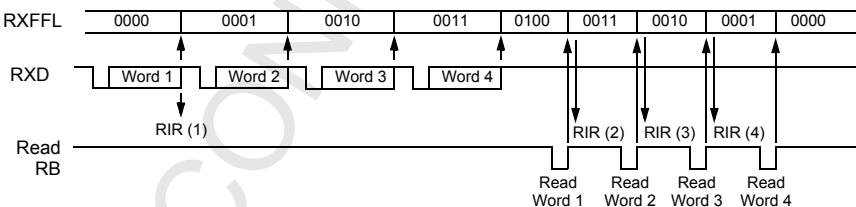
- FIFO filling levels
- Read operations on the **SSCPD\_RB** data register

**Note:** There is no interrupt when writing to **SSCPD\_TB**, only when the data is put on the Transmit line.

Interrupt generation for the receive FIFO depends on the RXFIFO filling level and the execution of read operations of register **SSCPD\_RB** (see [Figure 11-10](#)). Transparent Mode for the RXFIFO is enabled when bits **SSCPD\_RXFCON.RXTMEN** and **SSCPD\_RXFCON.RXFEN** are set.

**Figure 11-10 Transparent Mode Receive FIFO Operation**

Content of  
**SSCPD\_RXFCON**.



If the RXFIFO is empty, a receive interrupt RIR is always generated when the first message is written into an empty RXFIFO (**SSCPD\_FSTAT.RXFLL** changes from 000000<sub>B</sub> to 000001<sub>B</sub>). If the RXFIFO is filled with at least one message, the occurrence of further receive interrupts depends on the read operations of register **SSCPD\_RB**. The receive interrupt RIR will always be activated after a **SSCPD\_RB** read operation if the RXFIFO still contains data (**SSCPD\_FSTAT.RXFLL** is not equal to 000000<sub>B</sub>). If the RXFIFO is empty after a **SSCPD\_RB** read operation, no further receive interrupt will be generated.

**CONFIDENTIAL****Synchronous Serial Interface on PD Bus**

If the RXFIFO is full (**SSCPD\_TXFCON.TXTMEN** and **SSCPD\_TXFCON.TXFEN** = maximum) and additional messages are received, a Error interrupt [EIR] is generated. In this case, the message last written into the receive FIFO is overwritten. If a **SSCPD\_RB** read operation is executed with the RXFIFO enabled but empty (underflow condition), an Error interrupt EIR is generated in response to bit **SSCPD\_CON.RE** set (if **SSCPD\_CON.REN** has been previously set).

If the RXFIFO is flushed in Transparent Mode, the software must take care that a previous pending receive interrupt is ignored.

*Note: The Receive FIFO Interrupt Trigger Level bitfield **SSCPD\_RXFCON.RXFITL** is a don't care in Transparent Mode.*

Interrupt generation for the transmit FIFO depends on the TXFIFO filling level and the execution of write operations to the register TB. Transparent Mode for the TXFIFO is enabled when bits **SSCPD\_TXFCON.TXTMEN** and **SSCPD\_TXFCON.TXFEN**.

TIR is also activated after a TXFIFO flush operation or when the TXFIFO becomes enabled (**SSCPD\_TXFCON.TXTMEN** and **SSCPD\_TXFCON.TXFEN** set) when it was previously disabled. In these cases, the TXFIFO is empty and ready to be filled with data.

If the TXFIFO is full (**SSCPD\_TXFCON.TXTMEN** and **SSCPD\_TXFCON.TXFFL** = maximum) and an additional message is written into **SSCPD\_TB**, an Error interrupt [EIR] is generated after the **SSCPD\_TB** write operation. In this case the data byte last written into the transmit FIFO is overwritten and an Error interrupt (EIR) is generated in response to bit **SSCPD\_CON.TE** set (if **SSCPD\_CON.TEN** has been previously set).

*Note: The Transmit FIFO Interrupt Trigger Level bitfield **SSCPD\_TXFCON.TXFITL** is a don't care in Transparent Mode.*

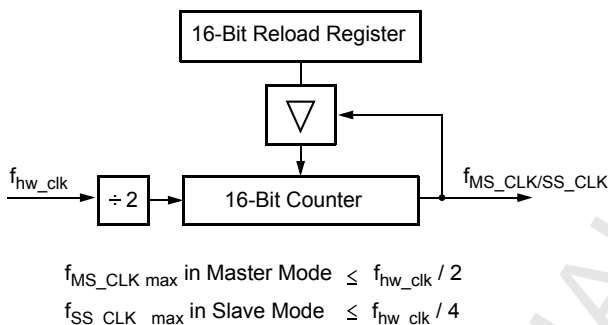
### **11.2.2.6 Baudrate Generation**

The serial channel SSC has its own dedicated 16-bit baudrate generator with 16-bit reload capability, allowing baudrate generation independent of the timers. **Figure 11-4 (on page 980)** shows the baudrate generator. **Figure 11-11** shows the baudrate generator of the SSC in more detail.

CONFIDENTIAL

Synchronous Serial Interface on PD Bus

Figure 11-11 SSC Baudrate Generator



The baudrate generator is clocked with the module clock  $f_{hw\_clk}$ . The timer counts downwards. Register **SSCPD\_BR** is the dual-function Baudrate Generator/Reload register. Reading **SSCPD\_BR**, while the SSC is enabled, returns the content of the timer. Reading **SSCPD\_BR**, while the SSC is disabled, returns the programmed reload value. In this mode the desired reload value can be written to **SSCPD\_BR**.

*Note: Never write to **SSCPD\_BR** while the SSC is enabled (**SSCPD\_CON.EN** must = 0 to write to **SSCPD\_BR**).*

The formulas below calculate either the resulting baudrate for a given reload value, or the required reload value for a given baudrate:

$$\text{Baudrate} = \frac{f_{hw\_clk}}{2 * (<SSCBR> + 1)} \quad \text{BR} = \frac{f_{hw\_clk}}{2 * \text{Baudrate}} - 1 \quad [36.17]$$

<SSCBR> represents the contents of the reload register, taken as unsigned 16-bit integer; while Baudrate is equal to  $f_{MS\_CLK/SS\_CLK}$  as shown in **Figure 11-11**.

The maximum baudrate that can be achieved when using a module clock of 52 MHz is 26 MBaud in Master Mode (with <SSCBR> = 0000<sub>H</sub>) or 13 MBaud in Slave Mode (with <SSCBR> = 0001<sub>H</sub>).

**Table 11-5** lists some possible baudrates together with the required reload values and the resulting bit times, assuming a module clock of 52 MHz.

**Table 11-5 Typical Baudrates of the SSC ( $f_{hw\_clk} = 52 \text{ MHz}$ )**

Reload Value	Baudrate (= $f_{MS\_CLK/SS\_CLK}$ )	Deviation
0000 <sub>H</sub>	26 MBaud (only in Master Mode)	0.0%
0001 <sub>H</sub>	13 MBaud	0.0%
FFFF <sub>H</sub>	396.72 Baud	0.0%

**CONFIDENTIAL**

**Synchronous Serial Interface on PD Bus**

The maximum baudrate that can be achieved when using a module clock of 78 MHz is 39 MBaud in Master Mode (with <SSCBR> = 0000<sub>H</sub>) or 19.5 MBaud in Slave Mode (with <SSCBR> = 0001<sub>H</sub>).

**Table 11-6** lists some possible baudrates together with the required reload values and the resulting bit times, assuming a module clock of 78 MHz.

**Table 11-6 Typical Baudrates of the SSC ( $f_{hw\_clk} = 78 \text{ MHz}$ )**

Reload Value	Baudrate (= $f_{MS\_CLK/SS\_CLK}$ )	Deviation
0000 <sub>H</sub>	39 MBaud (only in Master Mode)	0.0%
0001 <sub>H</sub>	19.5 MBaud	0.0%
FFFF <sub>H</sub>	595.09 Baud	0.0%

### 11.2.2.7 Error Detection Mechanisms

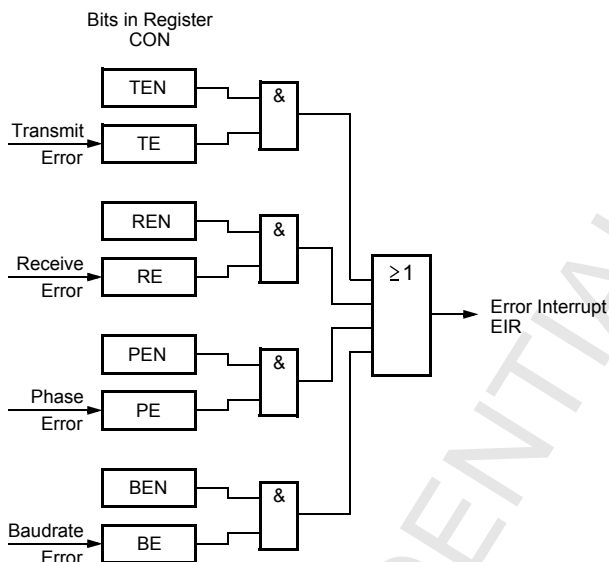
The SSC is able to detect four different error conditions:

- Receive Error and Phase Error are detected in all modes
- Transmit Error and Baudrate Error are detected only in Slave Mode.

When an error is detected, the respective error flag is set and an error interrupt request is generated (see **Figure 11-12**): EIR IRQ line is activated during 2 CPU clocks only and only if the respective EN bit (**TEN**, **REN**, **PEN**, **BEN**) in **SSCPD\_CON** has been set. The error flag is automatically reset by the Low to High IRQ pulse. No indication but a control by **SSCPD\_CON** programming mode indicates the IRQ cause.

*Note: The error interrupt handler must clear the associated (enabled) error flag(s) to prevent repeated interrupt requests.*

Figure 11-12 SSC Error Interrupt Control



A **Receive Error** (Master or Slave Mode) is detected when a new data frame is completely received but the previous data was not read out of the receive buffer register RB. This condition sets the error flag **SSCPD\_CON.RE** and, when enabled via **SSCPD\_CON.REN**, the error interrupt request line EIR. The old data in the receive buffer **SSCPD\_RB** will be overwritten with the new value and is irretrievably lost (also when FIFO overflows).

A **Phase Error** (Master or Slave Mode) is detected when the incoming data at pin MRST (Master Mode) or MTSR (Slave Mode), sampled with the same frequency as the module clock, changes between one cycle before and two cycles after the latching edge of the shift clock signal SCLK. This condition sets the error flag **SSCPD\_CON.PE** and, when enabled via **SSCPD\_CON.PEN**, the error interrupt request line EIR.

A **Baudrate Error** (Slave Mode) is detected when the incoming clock signal deviates from the programmed baudrate by more than 100%, that is, it is either more than twice or less than the half of the expected baudrate. This condition sets the error flag **SSCPD\_CON.BE** and, when enabled via **SSCPD\_CON.BEN**, the error interrupt request line EIR. Using this error detection capability requires that the slave's baudrate generator is programmed to the same baudrate as the master device. This feature detects false additional, or missing pulses on the clock line (within a certain frame).

*Note: If this error condition occurs and bit **SSCPD\_CON.AREN** = 1, an automatic reset of the SSC will be performed in case of this error. This is done to re-initialize the*

CONFIDENTIAL

Synchronous Serial Interface on PD Bus

*SSC if too few or too many clock pulses have been detected. In that case, no data are loaded from shift register to **SSCPD\_RB**.*

A **Transmit Error** (Slave Mode) is detected when a transfer was initiated by the master (SSC\_CLK gets active) but the transmit buffer **SSCPD\_TB** of the slave was not updated since the last transfer. This condition sets the error flag **SSCPD\_CON.TE** and, when enabled via **SSCPD\_CON.TEN**, the error interrupt request line EIR. If a transfer starts while the transmit buffer is not updated, the slave will shift out the 'old' contents of the shift register, which normally is the data received during the last transfer. This may lead to corruption of the data on the transmit/receive line in half-duplex mode (open drain configuration) if this slave is not selected for transmission. This mode requires that slaves not selected for transmission only shift out ones; that is, their transmit buffers must be loaded with  $FFFF_H$  prior to any transfer.

*Note: A slave with push/pull output drivers not selected for transmission, will normally have its output drivers switched. However, in order to avoid possible conflicts or misinterpretations, it is recommended to always load the slave's transmit buffer prior to any transfer.*

A **Transmit Error** (Master or Slave Mode) is generated when a new data frame is written but the previous data was not sent. This condition sets the error flag **SSCPD\_CON.TE** and, when enabled via **SSCPD\_CON.TEN**, the Error interrupt request line EIR. The old data in the transmit buffer **SSCPD\_TB** is overwritten with the new value and is irretrievably lost.

**CONFIDENTIAL**

**Synchronous Serial Interface on PD Bus**

### 11.2.3 SSC Kernel Registers

Refer to [Section 12.2 PD-Bus Register Addressses \(on Page 1273\)](#) for the addresses of the SSC registers.

**Table 11-7 SSC Register Summary**

Name	Clock	Description	Access Condition
<a href="#">SSCPD_PISEL</a>	hw_clk <sup>1)</sup>	Peripheral Input Select Register	Bit addressable
<a href="#">SSCPD_ID</a>	hw_clk <sup>1)</sup>	Peripheral Input Select Register	Bit addressable
<a href="#">SSCPD_CON</a>	hw_clk <sup>1)</sup>	Control Register	Bit addressable
<a href="#">SSCPD_BR</a>	cfg_clk <sup>1)</sup>	Baudrate Timer Reload Register	Bit addressable
<a href="#">SSCPD_TB</a>	cfg_clk <sup>1)</sup>	Transmit Buffer Register	Bit addressable
<a href="#">SSCPD_RB</a>	hw_clk <sup>1)</sup>	Receive Buffer Register	Bit addressable
<a href="#">SSCPD_RXFCON</a>	hw_clk <sup>1)</sup>	Receive FIFO Control Register	Bit addressable
<a href="#">SSCPD_TXFCON</a>	hw_clk <sup>1)</sup>	Transmit FIFO Control Register	Bit addressable
<a href="#">SSCPD_FSTAT</a>	hw_clk <sup>1)</sup>	FIFO Status Register	Bit addressable

<sup>1)</sup> Refer to Clock Domain in [System Integration: \(on Page 977\)](#).

**CONFIDENTIAL**

**Synchronous Serial Interface on PD Bus**

### 11.2.3.1 Port Input Select Register

Normally, the **SSCPD\_PISEL** register controls the receiver input selection of the SSC via an input multiplexer.

However, the E-GOLDradio uses only one of the input lines of each input type, the other input is tied to logical level 0. Therefore, all bits in this register must remain at their reset default values of 0 to select the lines connected the input pads.

#### SSCPD\_PISEL

#### Port Input Select Register

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED													CIS	SIS	MIS

Field	Bits	Type	Description
<b>MIS</b>	0	rw	<b>Master Mode Receiver Input Select</b> 0 SSC0_MRST0 selected (Default) 1 SSC0_MRST1 selected, there is no input over this line
<b>SIS</b>	1	rw	<b>Slave Mode Receiver Input Select</b> 0 SSC0_MTST0 selected (Default) 1 SSC0_MTST1 selected, there is no input over this line
<b>CIS</b>	2	rw	<b>Slave Mode Clock Input Select</b> 0 SSC0_CLK0 selected (Default) 1 SSC0_CLK1 selected, there is no input over this line
<b>RESERVED</b>	15:3	r	Reserved; these bits must be left at their reset values.



**CONFIDENTIAL**

**Synchronous Serial Interface on PD Bus**

### 11.2.3.1.1 SSC Identification Register

**SSCPD\_ID**

**SSC Identification Register**

**Reset value: 4526<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>Module_ID</b>								<b>Revision_Number</b>							

Field	Bits	Type	Description
<b>Revision_Number</b>	0:7	r	<b>CGU Revision Number</b> These hard-wired bits are used for module revision numbering.
<b>Module_ID</b>	8:15	r	<b>CGU Identification Number</b> These hard-wired bits are used for module identification numbering.

### 11.2.3.1.2 Configuration Register

The operating mode of the serial channel SSC is controlled by the control register **SSCPD\_CON**. This register contains control bits for mode and error check selection, and status flags for error identification. Depending on bit **EN**, either control functions or status flags and master/slave control is enabled.

**SSCPD\_CON.EN = 0: Programming Mode**

**SSCPD\_CON**

**Control Register**

**Reset Value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>EN</b>	<b>MS</b>	<b>RES ERV ED</b>	<b>A REN</b>	<b>BEN</b>	<b>PEN</b>	<b>REN</b>	<b>TEN</b>	<b>LB</b>	<b>PO</b>	<b>PH</b>	<b>HB</b>	<b>BM</b>			

Field	Bits	Type	Description
<b>BM</b>	3:0	rw	<b>Data Width Selection</b> 0000 Reserved. Do not use this combination. 0001 - 1111 Transfer Data Width is 2...16 bit (<BM>+1)
<b>HB</b>	4	rw	<b>Heading Control</b> 0 Transmit/Receive LSB First 1 Transmit/Receive MSB First

**CONFIDENTIAL**

**Synchronous Serial Interface on PD Bus**

Field	Bits	Type	Description
<b>PH</b>	5	rw	<b>Clock Phase Control</b> 0 Shift transmit data on the leading clock edge, latch on trailing edge 1 Latch receive data on leading clock edge, shift on trailing edge
<b>PO</b>	6	rw	<b>Clock Polarity Control</b> 0 Idle clock line is low, leading clock edge is low-to-high transition 1 Idle clock line is high, leading clock edge is high-to-low transition
<b>LB</b>	7	rw	<b>Loop Back Control</b> 0 Normal output 1 Receive input is connected with transmit output (half-duplex mode if connecting by wiring MTSR ST). <b>SSCPD_TB</b> => <b>SSCPD_RB</b> with TIR & RIR “normal” generation. This could be useful for SSC design testing (alone).
<b>TEN</b>	8	rw	<b>Transmit Error Enable</b> 0 Ignore transmit errors 1 Check transmit errors
<b>REN</b>	9	rw	<b>Receive Error Enable</b> 0 Ignore receive errors 1 Check receive errors
<b>PEN</b>	10	rw	<b>Phase Error Enable</b> 0 Ignore phase errors 1 Check phase errors
<b>BEN</b>	11	rw	<b>Baudrate Error Enable</b> 0 Ignore baudrate errors 1 Check baudrate errors
<b>AREN</b>	12	rw	<b>Automatic Reset Enable</b> 0 No additional action upon a baudrate error 1 The SSC is automatically reset upon a baudrate error ( <b>SSCPD_RB</b> is not loaded with data from received line)

**CONFIDENTIAL**

**Synchronous Serial Interface on PD Bus**

Field	Bits	Type	Description
<b>MS</b>	14	rw	<b>Master Select</b> 0 Slave Mode. Operate on shift clock received via SCLK. 1 Master Mode. Generate shift clock and output it via SCLK.
<b>EN</b>	15	rw	<b>Enable Bit = 0</b> Transmission and reception disabled. Access to control bits.
<b>RESERVED</b>	13	r	Reserved for future use; this bit must be left at their reset values.

CONFIDENTIAL

**CONFIDENTIAL**

**Synchronous Serial Interface on PD Bus**

**SSCPD\_CON.EN = 1: Operating Mode**

**SSCPD\_CON**  
**Control Register**

**Reset Value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN	MS	RES ERV ED	BSY	BE	PE	RE	TE	RESERVED				BC			

Field	Bits	Type	Description
<b>BC</b>	3:0	rh	<b>Bit Count Field</b> 0001 to 1111 Shift counter is updated with every shifted bit. <b>Note: Do not write to this bit field.</b>
<b>TE</b>	8	rwh	<b>Transmit Error Flag</b> 0 No error 1 Transfer starts with the slave's transmit buffer not being updated or a transmit buffer overflow
<b>RE</b>	9	rwh	<b>Receive Error Flag</b> 0 No error 1 Reception completed before the receive buffer was read
<b>PE</b>	10	rwh	<b>Phase Error Flag</b> 0 No error 1 Received data changes around sampling clock edge
<b>BE</b>	11	rwh	<b>Baudrate Error Flag</b> 0 No error 1 More than factor 2 or 0.5 between slave's actual and expected baudrate
<b>BSY</b>	12	rh	<b>Busy Flag</b> Set while a transfer is in progress. <b>Note: Do not write to this bit.</b>
<b>MS</b>	14	rw	<b>Master Select Bit</b> 0 Slave Mode. Operate on shift clock received via SCLK. 1 Master Mode. Generate shift clock and output it via SCLK.

**CONFIDENTIAL**

**Synchronous Serial Interface on PD Bus**

Field	Bits	Type	Description
<b>EN</b>	15	rw	<b>Enable Bit = 1</b> Transmission and reception enabled. Access to status flags and M/S control
<b>RESERVED</b>	7:4, 13	r	Reserved for future use; these bits must be left at their reset values.

*Note: The target of an access to **SSCPD\_CON** (control bits or flags) is determined by the state of **SSCPD\_CON.EN** prior to the access; that is, writing **C057<sub>H</sub>** to **SSCPD\_CON** in programming mode (**EN** = 0) initializes the SSC (**EN** was 0) and then turn it on (**EN** = 1). When writing to **SSCPD\_CON**, ensure that reserved locations receive zeros.*

### 11.2.3.1.3 Baudrate Timer Reload Register

The SSC baudrate timer reload register **SSCPD\_BR** contains the 16-bit reload value for the baudrate timer.

**SSCPD\_BR**

**Baudrate Timer Reload Register**

**Reset Value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>BR_VALUE</b>															

Field	Bits	Type	Description
<b>BR_VALUE</b>	15:0	rw	<b>Baudrate Timer/Reload Register Value</b> Reading <b>SSCPD_BR</b> returns the 16-bit content of the baudrate timer. Writing <b>SSCPD_BR</b> loads the baudrate timer reload register with <b>BR_VALUE</b> (write is only possible when <b>SSCPD_CON.EN</b> = 0).

**CONFIDENTIAL**

**Synchronous Serial Interface on PD Bus**

### 11.2.3.1.4 Transmitter Buffer Register

The SSC transmitter buffer register **SSCPD\_TB** contains the transmit data value.

**SSCPD\_TB**

**Transmitter Buffer Register**

**Reset Value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TB_VALUE															

Field	Bits	Type	Description
<b>TB_VALUE</b>	15:0	rw	<b>Transmit Data Register Value</b> <b>TB_VALUE</b> is the data value to be transmitted. Unselected bits of <b>SSCPD_TB</b> are ignored during transmission.

### 11.2.3.1.5 Receiver Buffer Register

The SSC receiver buffer register **SSCPD\_RB** contains the receive data value.

**SSCPD\_RB**

**Receiver Buffer Register**

**Reset Value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RB_VALUE															

Field	Bits	Type	Description
<b>RB_VALUE</b>	15:0	rh	<b>Receive Data Register Value</b> <b>SSCPD_RB</b> contains the received data value <b>RB_VALUE</b> . Unselected bits of <b>SSCPD_RB</b> will be not valid and should be ignored.

**CONFIDENTIAL**

**Synchronous Serial Interface on PD Bus**

### 11.2.3.1.6 Receive FIFO Control Register

**SSCPD\_RXFCON**

**Receive FIFO Control Register**

**Reset value: 0100<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
RESERVED		RXFITL						RESERVED						RXTMEN	RXFFLU	RXFEN

Field	Bits	Type	Description
<b>RXFEN</b>	0	rw	<b>Receive FIFO Enable</b> 0 Receive FIFO is disabled 1 Receive FIFO is enabled <i>Note: Resetting <b>RXFEN</b> automatically flushes the receive FIFO.</i>
<b>RXFLU</b>	1	rw	<b>Receive FIFO Flush</b> 0 No operation 1 Receive FIFO is flushed <i>Note: Setting <b>RXFFLU</b> clears bitfield <b>SSCPD_FSTAT.RXFFL</b>. <b>RXFFLU</b> is always read as 0.</i>
<b>RXTMEN</b>	2	rw	<b>Receive FIFO Transparent Mode Enable</b> 0 Receive FIFO Transparent Mode is disabled 1 Receive FIFO Transparent Mode is enabled <i>Note: This bit is don't care if the receive FIFO is disabled (<b>RXFEN</b> = 0).</i>

**CONFIDENTIAL**

**Synchronous Serial Interface on PD Bus**

Field	Bits	Type	Description
<b>RXFITL</b>	13:8	rw	<p><b>Receive FIFO Interrupt Trigger Level</b>  Defines a receive FIFO interrupt trigger level. A receive interrupt request (RIR) is always generated after the reception of a byte when the filling level of the receive FIFO is <u>equal to or greater</u> <b>RXFITL</b></p> <p>000000Reserved. Do not use this combination  000001Interrupt trigger level is set to one  000010Interrupt trigger level is set to two  ...  100000Interrupt trigger level is set to thirty two  100001Reserved Do not use this combination.  ...      Reserved Do not use this combination.  111111Reserved Do not use this combination.</p> <p><i>Note: In Transparent Mode this bitfield is don't care.</i></p> <p><i>Note: Combinations defining a interrupt trigger level greater then the configured FIFO size should not be used</i></p> <p><b>Note: 32 Bytes is the maximum authorized.</b></p>
<b>RESERVED</b>	7:3, 15:14	r	Reserved; these bits must be left at their reset values.



**CONFIDENTIAL**

**Synchronous Serial Interface on PD Bus**

### 11.2.3.1.7 Transmit FIFO Control Register

**SSCPD\_TXFCON**

**Transmit FIFO Control Register**

**Reset value: 0100<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
RESERVED		TXFITL						RESERVED						TX TM EN	TXF FLU	TXF EN

Field	Bits	Type	Description
<b>TXFEN</b>	0	rw	<b>Transmit FIFO Enable</b> 0 Transmit FIFO is disabled 1 Transmit FIFO is enabled <i>Note: Resetting <b>TXFEN</b> automatically flushes the transmit FIFO.</i>
<b>TXFLU</b>	1	rw	<b>Transmit FIFO Flush</b> 0 No operation 1 Transmit FIFO is flushed <i>Note: Setting <b>TXFFLU</b> clears bitfield <b>TXFFL</b> in register <b>SSCPD_FSTAT</b>. <b>TXFFLU</b> is always read as 0.</i>
<b>TXTMEN</b>	2	rw	<b>Transmit FIFO Transparent Mode Enable</b> 0 Transmit FIFO Transparent Mode is disabled 1 Transmit FIFO Transparent Mode is enabled <i>Note: This bit is don't care if the receive FIFO is disabled (<b>TXFEN</b> = 0).</i>

**CONFIDENTIAL**

**Synchronous Serial Interface on PD Bus**

Field	Bits	Type	Description
<b>TXFITL</b>	13:8	rw	<p><b>Transmit FIFO Interrupt Trigger Level</b>  Defines a transmit FIFO interrupt trigger level. A transmit interrupt request (TIR) is always generated after the transfer of a byte when the filling level of the transmit FIFO is <u>equal to or less than TXFITL</u>.</p> <p>000000Reserved. Do not use this combination.  000001Interrupt trigger level is set to one  000010Interrupt trigger level is set to two  ...  100000Interrupt trigger level is set to thirty two  100001Reserved Do not use this combination.  ...      Reserved Do not use this combination.  111111Reserved Do not use this combination.</p> <p><i>Note: In Transparent Mode this bitfield is don't care.</i></p> <p><i>Note: Combinations defining an interrupt trigger level greater then the configured FIFO size should not be used.</i></p> <p><b>Note: 32 bytes is the maximum authorized.</b></p>
<b>RESERVED</b>	7:3, 15:14	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**Synchronous Serial Interface on PD Bus**

### 11.2.3.1.8 FIFO Status Register

**SSCPD\_FSTAT**

**FIFO Status Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		TXFFL					RESERVED		RXFFL						

Field	Bits	Type	Description
RXFFL	5:0	rh	<p><b>Receive FIFO Filling Level</b></p> <p>000000Receive FIFO is filled with zero bytes</p> <p>000001Receive FIFO is filled with one byte</p> <p>...</p> <p>100000Receive FIFO is filled with thirty two bytes</p> <p>111110Reserved. Reserved Do not use this combination.</p> <p>111111Reserved. Reserved Do not use this combination.</p> <p><i>Note: <b>RXFFL</b> is cleared after a receive FIFO flush operation.</i></p> <p><i>Note: Combinations defining a interrupt trigger level greater then the configured FIFO size should not be used.</i></p> <p><b>Note: 32 bytes is the maximum read.</b></p>

**CONFIDENTIAL**

**Synchronous Serial Interface on PD Bus**

Field	Bits	Type	Description
<b>TXFFL</b>	13:8	rh	<p><b>Transmit FIFO Filling Level</b></p> <p>000000Receive FIFO is filled with zero bytes</p> <p>000001Receive FIFO is filled with one byte</p> <p>...</p> <p>100000Receive FIFO is filled with thirty two bytes</p> <p>111110Reserved. Reserved Do not use this combination.</p> <p>111111Reserved. Reserved Do not use this combination.</p> <p><i>Note: <b>TXFFL</b> is cleared after a receive FIFO flush operation.</i></p> <p><i>Note: Combinations defining a interrupt trigger level greater then the configured FIFO size should not be used.</i></p> <p><b>Note: 32 bytes is the maximum transmitted.</b></p>
<b>RESERVED</b>	7:6, 15:14	r	Reserved; these bits must be left at their reset values.

### 11.2.4 Interrupts

For a detailed description of the various interrupts refer to [Section 11.2.2 General Operation \(on Page 979\)](#). An overview is given in [Table 11-8](#).

**Table 11-8 SSC Interrupt Sources**

Interrupt	Signal	Description
Transmission starts (master only)	TIR	Indicates that the transmit buffer can be reloaded with new data. If a FIFO is configured for the SSC and <a href="#">SSCPD_TXFCON.TXTMEN</a> is cleared <a href="#">SSCPD_TXFCON.TXFIFL</a> defines when the interrupt is generated depending on the FIFO filling state (when Trigger level set, TIR occurs when FIFO = or <).
Transmission ends (master only)	RIR	The configured number of bits have been transmitted and shifted to the receive buffer. If a FIFO is configured for the SSC and <a href="#">SSCPD_RXFCON.RXTMEN</a> is cleared <a href="#">SSCPD_RXFCON.RXFIFL</a> defines when the interrupt is generated depending on the FIFO filling state (when Trigger level set, RIR occurs when FIFO = or >).

**CONFIDENTIAL**

**Synchronous Serial Interface on PD Bus**

**Table 11-8 SSC Interrupt Sources**

Interrupt	Signal	Description
Transparent Read Operation	RIR	In Transparent Mode a receive interrupt is always generated on a read operation from the MCU to the receive FIFO if the FIFO is not empty after this operation.
Receive Overflow	EIR	If an additional frame is received when the FIFO is completely full, an overflow error occurred. The interrupt is generated and the previously received frame is overwritten and therefore lost in the FIFO.
Transmit Overflow	EIR	If an additional frame is written to the transmit FIFO when it is completely full an overflow error occurred. The interrupt is generated and the previously written frame is overwritten and therefore lost in the FIFO.
Read to empty FIFO	RIR	A read operation from the MCU to an empty receive FIFO generated this interrupt.
Transparent first receive	RIR	In Transparent Mode the first received message in an empty receive FIFO generates this interrupt.
Flush Action	TIR	If <b>SSCPD_TXFCON.TXTMEN</b> is set, a transmit interrupt is generated when the transmit FIFO is flushed.
FIFO Enable	TIR	A transmit interrupt is generated when the transmit FIFO ( <b>SSCPD_TXFCON.TXTMEN</b> ) is enabled and <b>SSCPD_TXFCON.TXFEN</b> is set from 0 to 1.
Receive Error	EIR	This interrupt occurs if a new data frame is completely received and the last data in the receive buffer wasn't read yet. If a FIFO is configured for the SSC and <b>SSCPD_RXFCON.RXTMEN</b> is cleared <b>SSCPD_RXFCON.RXFIFL</b> defines when the interrupt is generated depending on the FIFO filling state. If a FIFO is configured for the SSC and <b>SSCPD_RXFCON.RXTMEN</b> is set the non FIFO interrupt generation in this case happened.
Phase Error	EIR	This interrupt is generated if the incoming data changes between one cycle before and two cycles after the latching edge of the shift clock signal SCLK.

CONFIDENTIAL

Synchronous Serial Interface on PD Bus

Table 11-8 SSC Interrupt Sources

Interrupt	Signal	Description
Baudrate Error	EIR	This interrupt is generated when the incoming clock signal deviates from the programmed baudrate by more than 100%.
Transmit Error (Slave Mode only)	EIR	This interrupt is generated when <b>SSCPD_TB</b> was not updated since the last transfer if a transfer is initiated by a master.

*Note: The interrupt request signals are connected to the central interrupt control unit; the control registers are also located there (XXXIC).*

**CONFIDENTIAL**

**ASC0**

## 11.3 ASC0

History	
Current version: Rev. 1.06, 2005-11-04	
Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.01	
<b>Page 1011</b>	Updated <b>System Integration</b> : WS00006682
Changes for Rev. 1.05	
<b>Page 1031</b>	Added Note about IrDA Baudrate re-configuration issue to <b>Section 11.3.5.14 Baudrate Generation</b>
Changes for Rev. 1.06	

### System Integration:

- Supply domain: VDD\_MAIN
- Chip internal interfaces:
  - Clock domain: Refer to **Section 10.4.1.3 Sub-System Clocks and Enables** and see **Figure 10-10 Clock Enable (on Page 662)**.
  - Bus domain: PD-Bus
- Interrupt sources:
- Monitor Pins: Refer to **Section 11.8.10 Internal Signal Monitoring (on Page 1209)**

**Note: The ASC0 is connected to the following pins:**

- RXD
- TXD
- ASC\_RTS\_n
- ASC\_CTS\_n

### 11.3.1 ASC0 Description

ASC0 is the acronym for Asynchronous Serial Communication interface. The ASC0 Subsystem consists of two blocks: one ASC0 controller itself and the AUTOSTART block. The ASC0 controller is a standard  $\mu$ C peripheral. The ASC0 subsystem also includes an echo mode (RxD0  $\rightarrow$  TxD0) which retransmits received data. The autostart block is watching for an edge at the RxD0 data line to wake-up the required circuitry. The AUTOSTART should only be used in the asynchronous operating mode.

### 11.3.2 Features of the ASC0 Controller

- Duplex asynchronous operating modes:
  - 8- or 9-bit data frames, LSB first
  - Parity bit generation/checking
  - One or two stop bits
  - Baudrate from 3.27 MBaud to 0.77 Baud (with a 52MHz module clock  $f_{hw\_clk}$ , for information about  $hw\_clk$ , refer to [Table 11-20 ASC0 Register Summary \(on Page 1047\)](#))
- Multiprocessor Mode for automatic address/data byte detection
- Loopback capability
- Support for IrDA data transmission up to 115.2 KBaud maximum
- Half-duplex 8-bit synchronous operating mode:
  - Baudrate from 6.5 MBaud to 662.5 Baud (with a 52 MHz module clock  $f_{hw\_clk}$ )
- Double buffered transmitter/receiver
- Interrupt generation:
  - On a transmitter buffer empty condition
  - On a transmit last bit of a frame condition
  - On a receiver buffer full condition
  - On an error condition (frame, parity, overrun error)
- Autobaud detection unit for asynchronous operating modes:
  - Detection of standard baudrates  
1200, 2400, 4800, 9600, 19 200, 38 400, 57 600, 115 200, and 230 400 Baud
  - Detection of non-standard baudrates
  - Detection of Asynchronous Modes
  - 7 bit, even parity; 7 bit, odd parity;  
8 bit, even parity; 8 bit, odd parity; 8 bit, no parity
  - Automatic initialization of control bits and baudrate generator after detection
  - Detection of a serial two-byte ASCII character frame
- FIFOs:
  - 8-stage receive FIFO (RXFIFO)



CONFIDENTIAL

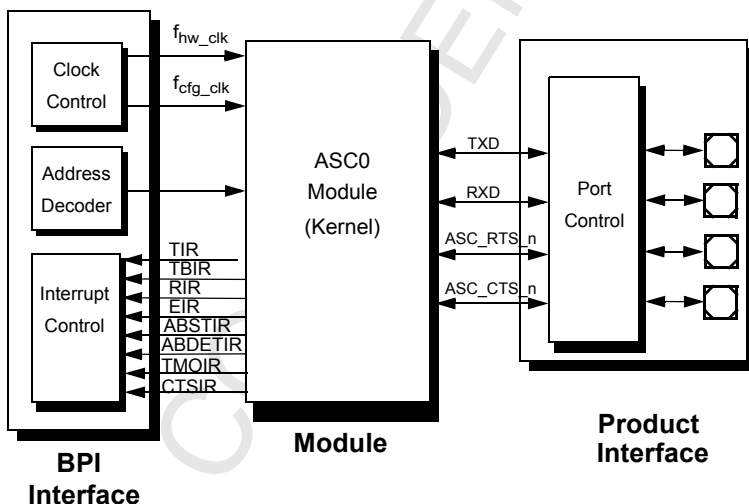
ASC0

- 8-stage transmit FIFO (TXFIFO)
- Independent control of RXFIFO and TXFIFO
- 9-Bit FIFO data width
- Programmable Receive/Transmit Interrupt Trigger Level
- Receive and transmit FIFO filling level indication
- Overrun error generation
- Underflow error generation
- HW Flow Control:
  - RTS/CTS handshaking
  - CTS interrupt
  - Programmable RTS de-assertion RXFIFO Trigger Level
- RX timeout

### 11.3.3 Functional Description of the ASC0

**Figure 11-13** shows all functional relevant interfaces associated with the ASC0 Kernel.

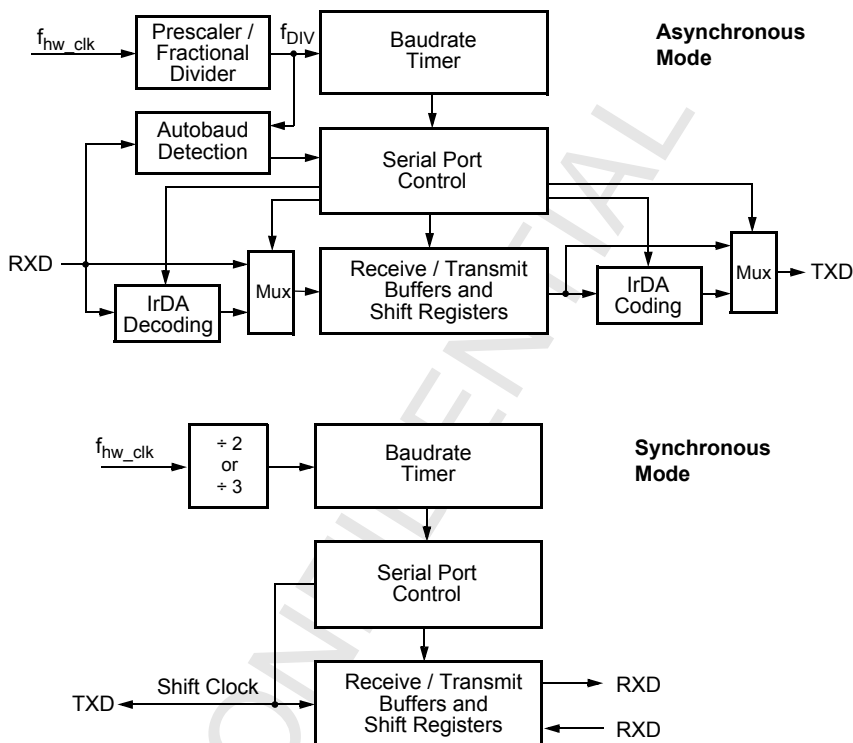
**Figure 11-13 ASC0 Interface Diagram**



### 11.3.4 Operational Overview

**Figure 11-14** shows a block diagram of the ASC0 with its operating modes (Asynchronous and Synchronous Mode).

**Figure 11-14 Block Diagram of the ASC0**



### 11.3.5 General Operation

In Synchronous Mode, data are transmitted or received synchronous to a shift clock that is generated by the microcontroller. In Asynchronous Mode, either 8- or 9-bit data transfer, parity generation, and the number of stop bits can be selected. Parity, framing, and overrun error detection is provided to increase the reliability of data transfers. Transmission and reception of data is double-buffered. For multiprocessor communication, a mechanism is provided to distinguish address bytes from data bytes. Testing is supported by a loop-back option. A 13-bit baudrate timer with a versatile input clock divider circuitry provides the serial clock signal. In a Special Asynchronous Mode, the ASC0 supports IrDA data transmission up to 115.2 KBaud with fixed or

**CONFIDENTIAL****ASC0**

programmable IrDA pulse width. Autobaud Detection allows to detect asynchronous data frames with its baudrate and mode with automatic initialization of the baudrate generator and the mode control bits.

A transmission is started by writing to the transmit buffer register **S0TBUF**. The selected operating mode determines the number of data bits that will actually be transmitted, so that, bits written to positions 9 through 15 of register **S0TBUF** are always insignificant. Data transmission is double-buffered, so a new character may be written to the transmit buffer register before the transmission of the previous character is complete. This allows the transmission of characters back-to-back without gaps.

Data reception is enabled by the Receiver Enable Bit **S0CON.REN**. After reception of a character has been completed, the received data can be read from the (read-only) receive buffer register **S0RBUF**; the received parity bit can also be read if provided by the selected operating mode. Bits in the upper half of **S0RBUF** that are not valid in the selected operating mode will be read as zeros.

Data reception is double-buffered, so that reception of a second character may already begin before the previously received character has been read out of the receive buffer register. In all modes, receive overrun error detection can be selected through bit **S0CON.OEN**. When enabled, the overrun error status flag **S0CON.OE** and the error interrupt request line EIR will be activated when the receive buffer register has not been read by the time reception of a ninth character is complete. The previously received character in the receive buffer is overwritten.

The Loopback Mode (selected by bit **S0CON.LB**) allows the data currently being transmitted to be received simultaneously in the receive buffer. This may be used to test serial communication routines at an early stage without having to provide an external network.

*Note: In Loopback Mode, the alternate input/output functions of the associated port pins are not necessary.*

*Note: Serial data transmission or reception is only possible when the Baudrate Generator Run bit **S0CON.R** is set. Otherwise, the serial interface is idle.*

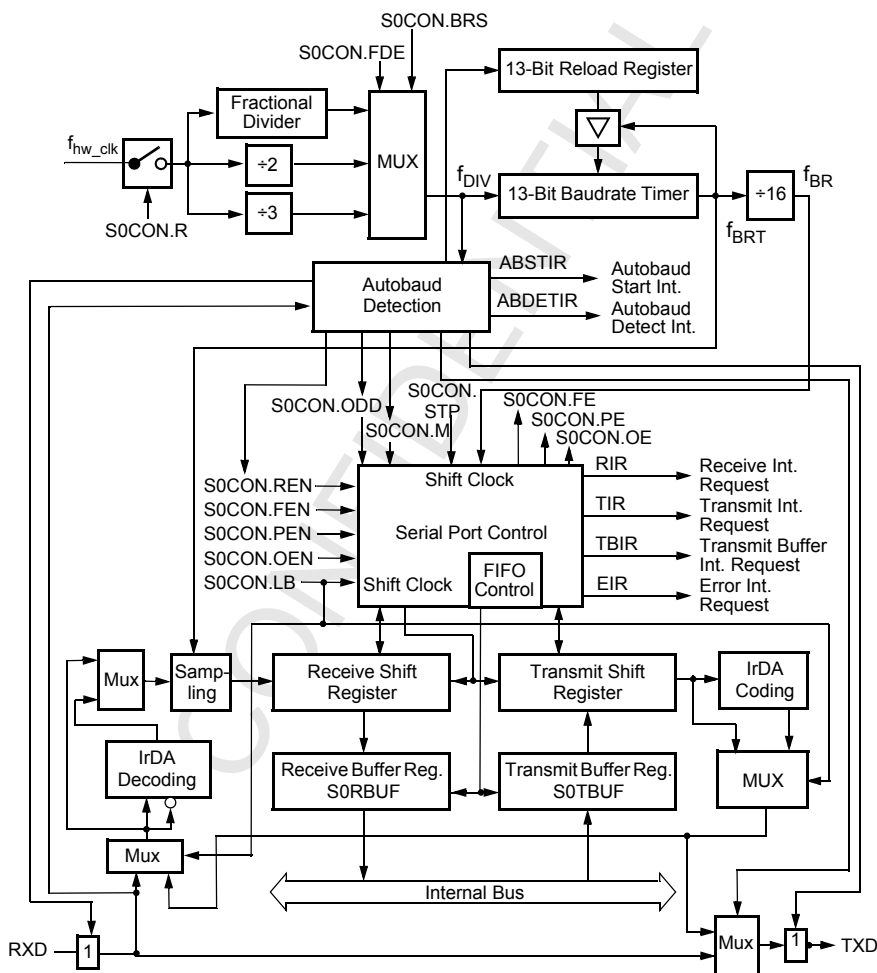
*Note: Do not program the Mode Control bit field **S0CON.M** to one of the reserved combinations to avoid unpredictable behavior of the serial interface.*

The operating mode of the serial channel ASC0 is controlled by its control register **S0CON**. This register contains control bits for mode and error check selection, and status flags for error identification.

### 11.3.5.1 Asynchronous Operation

Asynchronous Mode supports full-duplex communication in which both transmitter and receiver use the same data frame format and the same baudrate. Data is transmitted on line TXD and received on line RXD. IrDA data transmission/reception is supported up to 115.2 KBit/s. **Figure 11-15** shows the block diagram of the ASC0 when operating in Asynchronous Mode.

**Figure 11-15 Asynchronous Mode of Serial Channel ASC0**

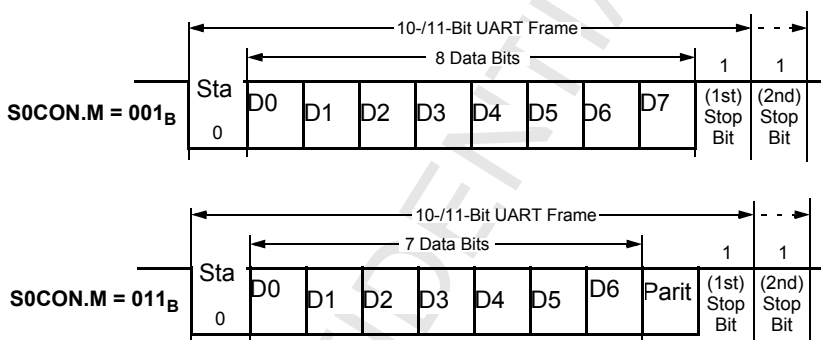


## 11.3.5.2 Asynchronous Data Frames

### 8-Bit Data Frames

8-bit data frames consist of either eight data bits D7...D0 (**S0CON.M** = 001<sub>B</sub>), or seven data bits D6...D0 plus an automatically generated parity bit (**S0CON.M** = 011<sub>B</sub>). Parity may be odd or even, depending on bit **S0CON.ODD**. An even parity bit will be set if the modulo-2-sum of the 7 data bits is 1. An odd parity bit will be cleared in this case. Parity checking is enabled via bit **S0CON.PEN** (always OFF in 8-bit data mode). The parity error flag **S0CON.PE** will be set, along with the error interrupt request flag, if a wrong parity bit is received. The parity bit itself will be stored in bit **S0RBUF.7**.

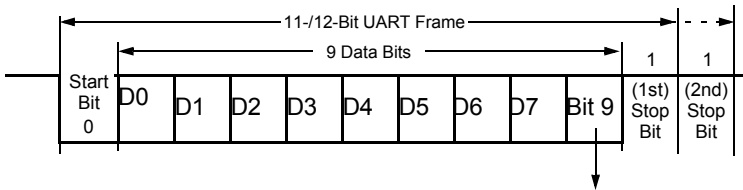
Figure 11-16 Asynchronous 8-Bit Frames



### 9-Bit Data Frames

9-bit data frames consist of either nine data bits D8...D0 (**S0CON.M** = 001<sub>B</sub>), eight data bits D7...D0 plus an automatically generated parity bit (**S0CON.M** = 111<sub>B</sub>), or eight data bits D7...D0 plus wake-up bit (**S0CON.M** = 101<sub>B</sub>). Parity may be odd or even, depending on bit **S0CON.ODD**. An even parity bit will be set if the modulo-2-sum of the 8 data bits is 1. An odd parity bit will be cleared in this case. Parity checking is enabled via bit **S0CON.PEN** (always OFF in 9-bit data and wake-up mode). The parity error flag **S0CON.PE** will be set, along with the error interrupt request flag, if a wrong parity bit is received. The parity bit itself will be stored in bit **S0RBUF.8**.

Figure 11-17 Asynchronous 9-Bit Frames



**S0CON.M = 100<sub>B</sub>:** Bit 9 = Data Bit D8  
**S0CON.M = 101<sub>B</sub>:** Bit 9 = Wake-up Bit  
**S0CON.M = 111<sub>B</sub>:** Bit 9 = Parity Bit

In wake-up mode, received frames are transferred to the receive buffer register only if the 9th bit (the wake-up bit) is 1. If this bit is 0, no receive interrupt request will be activated and no data will be transferred.

This feature may be used to control communication in a multi-processor system: When the master processor wants to transmit a block of data to one of several slaves, it first sends out an address byte to identify the target slave. An address byte differs from a data byte in that the additional 9th bit is a 1 for an address byte, but is a 0 for a data byte; so, no slave will be interrupted by a data 'byte'. An address 'byte' will interrupt all slaves (operating in 8-bit data + wake-up bit mode), so each slave can examine the eight LSBs of the received character (the address). The addressed slave will switch to 9-bit data mode (such as by clearing bit **S0CON.M[0]**), to enable it to also receive the data bytes that will be coming (having the wake-up bit cleared). The slaves not being addressed remain in 8-bit data + wake-up bit mode, ignoring the following data bytes.

## IrDA Frames

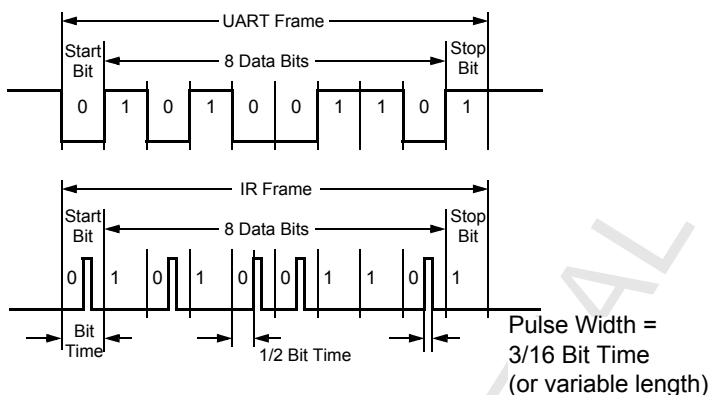
The modulation schemes of IrDA are based on standard asynchronous data transmission frames. The asynchronous data format in IrDA Mode (**S0CON.M = 010<sub>B</sub>**) is defined as follows:

1 start bit/8 data bits/1 stop bit

The coding/decoding of the asynchronous data frames is shown in **Figure 11-18**. In general, during IrDA transmissions, UART frames are encoded into IR frames and vice versa. A low level on the IR frame indicates an "LED off" state. A high level on the IR frame indicates an "LED on" state.

For a 0 bit in the UART frame, a high pulse is generated. For a 1 bit in the UART frame, no pulse is generated. The high pulse starts in the middle of a bit cell and has a fixed width of 3/16 of the bit time. The ASC0 also allows the length of the IrDA high pulse to be programmed. Further, the polarity of the received IrDA pulse can be inverted in IrDA Mode. **Figure 11-18** shows the non-inverted IrDA pulse scheme.

Figure 11-18 IrDA Frame Encoding/Decoding



The ASC0 IrDA pulse mode/width register PMW contains the 8-bit IrDA pulse width value and the IrDA pulse width mode select bit. This register is required in the IrDA operating mode only.

### 11.3.5.3 Asynchronous Transmission

Asynchronous transmission begins at the next overflow of the divide-by-16 baudrate timer (transition of the baudrate clock  $f_{BR}$ ), if bit **S0CON.R** is set and data has been loaded into **S0TBUF**. The transmitted data frame consists of three basic elements:

- Start bit
- Data field (eight or nine bits, LSB first, including a parity bit, if selected)
- Delimiter (one or two stop bits)

Data transmission is double-buffered. When the transmitter is idle, the transmit data loaded in the transmit buffer register is immediately moved to the transmit shift register, thus freeing the transmit buffer for the next data to be sent. This is indicated by the transmit buffer interrupt request line TBIR being activated. **S0TBUF** may now be loaded with the next data, while transmission of the previous data continues.

The transmit interrupt request line TIR will be activated before the last bit of a frame is transmitted, that is, before the first or the second stop bit is shifted out of the transmit shift register.

*Note: The transmitter output pin TXD must be configured for alternate data output.*

### 11.3.5.4 Transmit FIFO Operation

The transmit FIFO (TXFIFO) provides the following functionality:

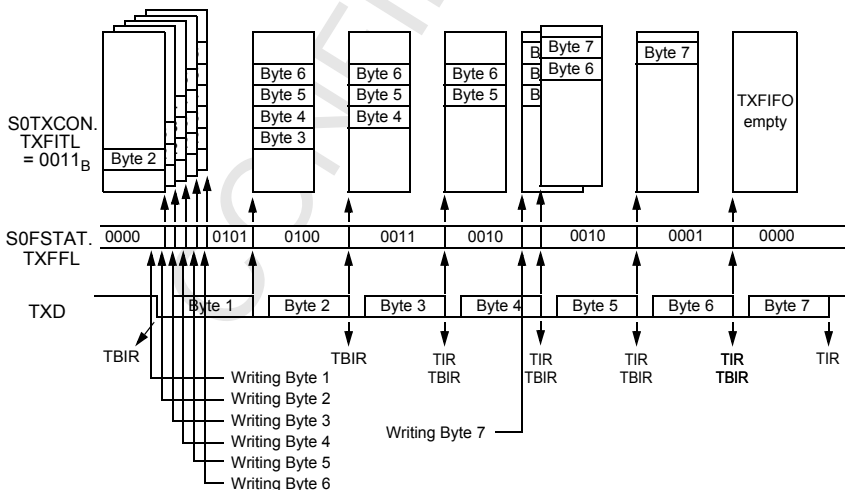
- Enable/disable control
- Programmable filling level for transmit interrupt generation
- Filling level indication
- FIFO clear (flush) operation
- FIFO overflow error generation.

The 8 stage transmit FIFO is controlled by the **S0TXFCON** control register. When bit **S0TXFCON.TXFEN** is set, the transmit FIFO is enabled. The interrupt trigger level defined by **S0TXFCON.TXFITL** defines the filling level of the TXFIFO at which a transmit buffer interrupt TBIR or a transmit interrupt TIR is generated. These interrupts are always generated when the filling level of the transmit FIFO is equal to or less than the value stored in **S0TXFCON.TXFITL**.

Bit field **S0FCSTAT.TXFFL** indicates the number of entries that are actually written (valid) in the TXFIFO. Therefore, the software can verify, for example, in the interrupt service routine, how many bytes can be still written into the transmit FIFO via register **S0TBUF** without getting an overrun error.

The transmit FIFO cannot be accessed directly. All data write operations into the TXFIFO are executed by writing into the **S0TBUF** register.

**Figure 11-19 Transmit FIFO Operation Example**



The example in **Figure 11-19** shows a typical 8 stage transmit FIFO operation. In this example seven bytes are transmitted via the TXD output line. The transmit FIFO interrupt



CONFIDENTIAL

ASC0

trigger level **S0TXFCON.TXFITL** is set to 000011<sub>B</sub>. The first byte written into the empty TXFIFO via **S0TBUF** is directly transferred into the transmit shift register and is not written into the FIFO. A transmit buffer interrupt will be generated in this case. After byte 1, bytes 2 to 6 are written into the transmit FIFO.

After the transfer of byte 3 from the TXFIFO into the transmit shift register of the ASC0, 3 bytes remain in the TXFIFO. Therefore, the value of **S0TXFCON.TXFITL** is reached and a transmit buffer interrupt will be generated at the beginning and a transmit interrupt at the end of the byte 3 serial transmission. During the serial transmission of byte 4, another byte (byte 7) is written into the TXFIFO (**S0TBUF** write operation). Finally, after the start of the serial transmission of byte 7, the TXFIFO is again empty.

If the TXFIFO is full and additional bytes are written into TBUF, the error interrupt will be generated with bit **S0CON.OE** set. In this case, the data byte that was last written into the transmit FIFO is overwritten and the transmit FIFO filling level **S0FCSTAT.TXFFL** is set to maximum.

The TXFIFO can be flushed or cleared by setting bit **S0TXFCON.TXFFLU**. After this TXFIFO flush operation, the TXFIFO is empty and the transmit FIFO filling level **S0FCSTAT.TXFFL** is set to 000000<sub>B</sub>. A running serial transmission is not aborted by a receive FIFO flush operation

*Note: The TXFIFO is flushed automatically with a reset operation of the ASC0 module and if the TXFIFO becomes disabled (resetting bit **S0TXFCON.TXFEN**) after it was previously enabled.*

### 11.3.5.5 Asynchronous Reception

Asynchronous reception is initiated by a falling edge (1-to-0 transition) on line RXD, provided that bits **S0CON.R** and **S0CON.REN** are set. The receive data input line RXD is sampled at 16 times the rate of the selected baudrate. A majority decision of the 7th, 8th, and 9th sample determines the effective bit value. This avoids erroneous results that may be caused by noise.

If the detected value is not a 0 when the start bit is sampled, the receive circuit is reset and waits for the next 1-to-0 transition at line RXD. If the start bit proves valid, the receive circuit continues sampling and shifts the incoming data frame into the receive shift register.

When the last stop bit has been received, the content of the receive shift register are transferred to the receive data buffer register **S0RBUF**. Simultaneously, the receive interrupt request line RIR is activated after the 9th sample in the last stop bit time slot (as programmed), regardless of whether valid stop bits have been received or not. The receive circuit then waits for the next start bit (1-to-0 transition) at the receive data input line.

*Note: The receiver input pin RXD must be configured for input.*

Asynchronous reception is stopped by clearing bit **S0CON.REN**. A currently received frame is completed including the generation of the receive interrupt request and an error interrupt request, if appropriate. Start bits that follow this frame will not be recognized.

*Note: In wake-up mode, received frames are transferred to the receive buffer register only if the 9th bit (the wake-up bit) is 1. If this bit is 0, no receive interrupt request will be activated and no data will be transferred.*

### 11.3.5.6 Receive FIFO Operation

The receive FIFO (RXFIFO) provides the following functionality:

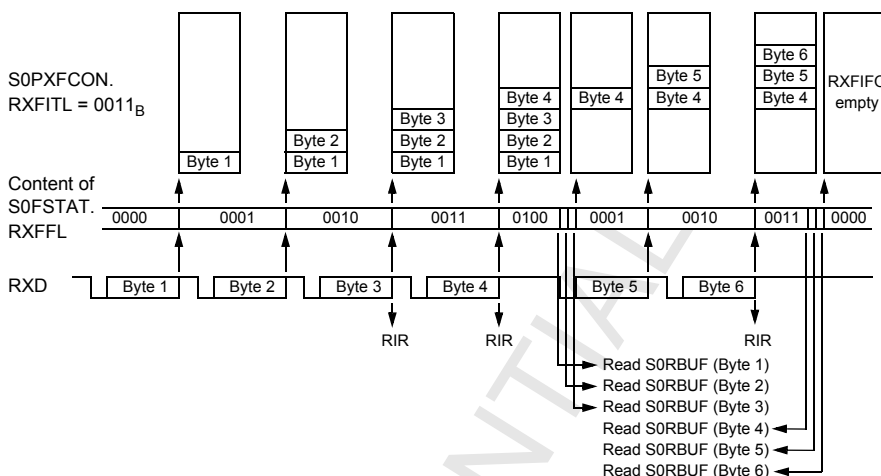
- Enable/disable control
- Programmable filling level for receive interrupt generation
- Filling level indication
- FIFO clear (flush) operation
- FIFO overflow error generation

The 8 stage receive FIFO is controlled by the **S0RXFCN** control register. When bit **S0RXFCN.RXFEN** is set, the receive FIFO is enabled. The interrupt trigger level defined by **S0RXFCN.RXFITL** defines the filling level of RXFIFO at which a receive interrupt RIR is generated. RIR is always generated when the filling level of the receive FIFO is equal to or greater than the value stored in **S0RXFCN.RXFITL**.

Bit field **S0FSTAT.RXFFL** in the FIFO status register FSTAT indicates the number of bytes that have been actually written into the FIFO and can be read out of the FIFO by a user program.

The receive FIFO cannot be accessed directly. All data read operations from the RXFIFO are executed by reading the **S0RBUF** register.

Figure 11-20 Receive FIFO Operation Example



The example in [Figure 11-20](#) shows a typical 8 stage receive FIFO operation. In this example, six bytes are received via the RXD input line. The receive FIFO interrupt trigger level [S0RXFCON.RXFITL](#) is set to 000011<sub>B</sub>. Therefore, the first receive interrupt RIR is generated after the reception of byte 3 (RXFIFO is filled with three bytes).

After the reception of byte 4, three bytes are read out of the receive FIFO. After this read operation, the RXFIFO still contains one byte. RIR becomes again active after two more bytes (byte 5 and 6) have been received (RXFIFO filled again with 3 bytes). Finally, the FIFO is cleared after three read operation.

If the RXFIFO is full and additional bytes are received, the receive interrupt RIR and the error interrupt EIR will be generated with bit [S0CON.OE](#) set. In this case, the data byte last written into the receive FIFO is overwritten. With the overrun condition, the receive FIFO filling level [S0FSTAT.RXFFL](#) is set to maximum. If a [S0RBUF](#) read operation is executed with the RXFIFO enabled but empty, an error interrupt EIR will be generated, and with bit [S0CON.OE](#) set. In this case, the receive FIFO filling level [S0FSTAT.RXFFL](#) is set to 000000<sub>B</sub>.

If the RXFIFO is available but disabled ([S0RXFCON.RXFEN](#) = 0) and the receive operation is enabled ([S0CON.REN](#) = 1), the asynchronous receive operation is functionally equivalent to the asynchronous receive operation of the ASC0 module.

The RXFIFO can be flushed or cleared by setting bit [S0RXFCON.RXFFLU](#). After this RXFIFO flush operation, the RXFIFO is empty and the receive FIFO filling level [S0FSTAT.RXFFL](#) is set to 000000<sub>B</sub>.

The RXFIFO is flushed automatically with a reset operation of the ASC0 module and if the RXFIFO becomes disabled (resetting bit [S0RXFCON.RXFEN](#)) after it was previously

enabled. Resetting bit **S0CON.REN** without resetting **S0RXFCON.RXFEN** does not affect (reset) the RXFIFO state. This means that the receive operation of the ASC0 is stopped, in this case, without changing the content of the RXFIFO. After setting **S0CON.REN** again, the RXFIFO with its content is again available.

*Note: After a successful autobaud detection sequence (if implemented), the RXFIFO should be flushed before data is received.*

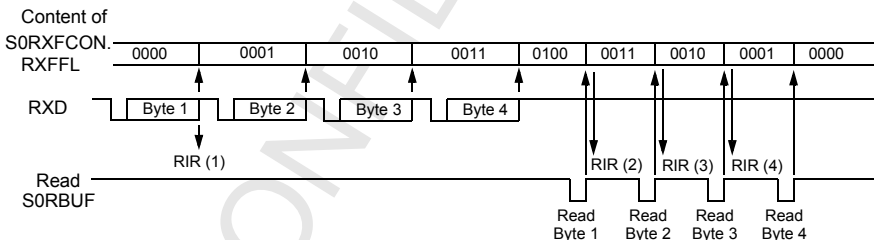
### 11.3.5.7 FIFO Transparent Mode

In Transparent Mode, a specific interrupt generation mechanism is used for receive and transmit buffer interrupts. In general, in Transparent Mode, receive interrupts are always generated if data bytes are available in the RXFIFO. Transmit buffer interrupts are always generated if the TXFIFO is not full. The relevant conditions for interrupt generation in Transparent Mode are:

- FIFO filling levels
- Read or write operations on the **S0RBUF** or **S0TBUF** data register.

Interrupt generation for the receive FIFO depends on the RXFIFO filling level and the execution of read operations of register **S0RBUF** (see **Figure 11-21**). Transparent Mode for the RXFIFO is enabled when bits **S0RXFCON.RXTMEN** and **S0RXFCON.RXFEN** are set.

**Figure 11-21 Transparent Mode Receive FIFO Operation**



If the RXFIFO is empty, a receive interrupt RIR is always generated when the first byte is written into an empty RXFIFO (**S0FSTAT.RXFFL** changes from 000000<sub>B</sub> to 000001<sub>B</sub>). If the RXFIFO is filled with at least one byte, the occurrence of further receive interrupts depends on the read operations of register **S0RBUF**. The receive interrupt RIR will always be activated after a **S0RBUF** read operation if the RXFIFO still contains data (**S0FSTAT.RXFFL** is not equal to 000000<sub>B</sub>). If the RXFIFO is empty after a **S0RBUF** read operation, no further receive interrupt will be generated.

If the RXFIFO is full (**S0FSTAT.RXFFL** = maximum) and additional bytes are received, an error interrupt EIR will be generated with bit **S0CON.OE** set. In this case, the data byte last written into the receive FIFO is overwritten. If a **S0RBUF** read operation is

executed with the RXFIFO enabled but empty (underflow condition), an error interrupt EIR will be generated as well, with bit **S0CON.OE** set.

If the RXFIFO is flushed in Transparent Mode, the software must take care that a previous pending receive interrupt is ignored.

*Note: The Receive FIFO Interrupt Trigger Level bit field **S0RXFCON.RXFITL** is a don't care in Transparent Mode.*

Interrupt generation for the transmit FIFO depends on the TXFIFO filling level and the execution of write operations to the register **S0TBUF**. Transparent Mode for the TXFIFO is enabled when bits **S0TXFCON.TXTMEN** and **S0TXFCON.TXFEN** are set.

A transmit buffer interrupt TBIR is always generated when the TXFIFO is not full (**S0FSTAT.TXFFL** not equal to maximum) after a byte has been written into register **S0TBUF**. TBIR is also activated after a TXFIFO flush operation or when the TXFIFO becomes enabled (**S0TXFCON.TXTMEN** and **S0TXFCON.TXFEN** set) when it was previously disabled. In these cases, the TXFIFO is empty and ready to be filled with data.

If the TXFIFO is full (**S0FSTAT.TXFFL** = maximum) and an additional byte is written into **S0TBUF**, no further transmit buffer interrupt will be generated after the **S0TBUF** write operation. In this case the data byte last written into the transmit FIFO is overwritten and an overrun error interrupt (EIR) will be generated with bit **S0CON.OE** set.

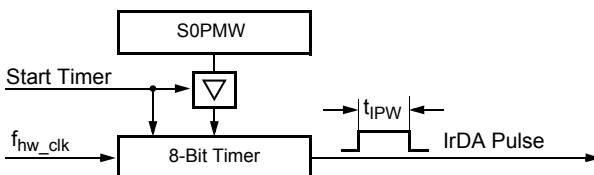
*Note: The Transmit FIFO Interrupt Trigger Level bit field **S0TXFCON.TXFITL** is don't care in Transparent Mode.*

### 11.3.5.8 IrDA Mode

The duration of the IrDA pulse is normally 3/16 of a bit period. The IrDA standard also allows the pulse duration to be independent of the baudrate or bit period. In this case, the width of the transmitted pulse always corresponds to the 3/16 pulse width at 115.2 kBaud, which is 1.627  $\mu$ s. Either fixed or bit-period-dependent IrDA pulse width generation can be selected. The IrDA pulse width mode is selected by bit **S0PWM.IRPW**.

In case of fixed IrDA pulse width generation, the lower eight bits in register PMW are used to adapt the IrDA pulse width to a fixed value such as 1.627  $\mu$ s. The fixed IrDA pulse width is generated by a programmable timer as shown in **Figure 11-22**.

**Figure 11-22 Fixed IrDA Pulse Generation**



**CONFIDENTIAL**

**ASC0**

The IrDA pulse width can be calculated according the formulas given in [Table 11-9](#).

**Table 11-9 Formulas for IrDA Pulse Width Calculation**

<b>S0PWM.PW_VALUE</b>		<b>Formulas</b>	
1 ... 255	0	$t_{IPW} = \frac{3}{16 \times \text{Baudrate}}$	$t_{IPW \min} = \frac{(\text{PMW} >> 1)}{f_{hw\_clk}}$
	1	$t_{IPW} = \frac{\text{PMW}}{f_{hw\_clk}}$	

*Note: The name PMW in the formulas of [Table 11-9](#) represents the contents of the pulse width/mode register bits **S0PWM.PW\_VALUE** taken as an unsigned 8-bit integer.*

The contents of **S0PWM.PW\_VALUE** further define the minimum IrDA pulse width ( $t_{IPW \min}$ ) that is still recognized as a valid IrDA pulse during a receive operation. This function is independent of the selected IrDA pulse width mode (fixed or variable) which is defined by bit **S0PWM.IRPW**. The minimum IrDA pulse width is calculated by a shift right operation of **S0PWM**{7:0} by one bit divided by the module clock  $f_{hw\_clk}$ .

*Note: If **S0PWM.IRPW** is cleared (fixed IrDA pulse width), **S0PWM.PW\_VALUE** must be a value which assures that  $t_{IPW} > t_{IPW \min}$ .*

**Table 11-10** gives three examples for typical frequencies of  $f_{hw\_clk}$ .

**Table 11-10 IrDA Pulse Width Adaption to 1.627  $\mu$ s**

<b><math>f_{hw\_clk}</math></b>	<b>PMW</b>	<b><math>t_{IPW}</math></b>	<b>Error</b>	<b><math>t_{IPW \min}</math></b>
13 MHz	21	1.615 $\mu$ s	- 0.12 %	0.77 $\mu$ s
26 MHz	42	1.615 $\mu$ s	- 0.12 %	0.81 $\mu$ s
52 MHz	84	1.615 $\mu$ s	- 0.12 %	0.81 $\mu$ s

### 11.3.5.9 RXD/TXD Data Path Selection in Asynchronous Modes

The data paths for the serial input and output data in Asynchronous Mode are affected by several control bits in the registers **S0CON** and **S0ABCON** as shown in [Figure 11-23](#). The Synchronous Mode operation is not affected by these data path selection capabilities.

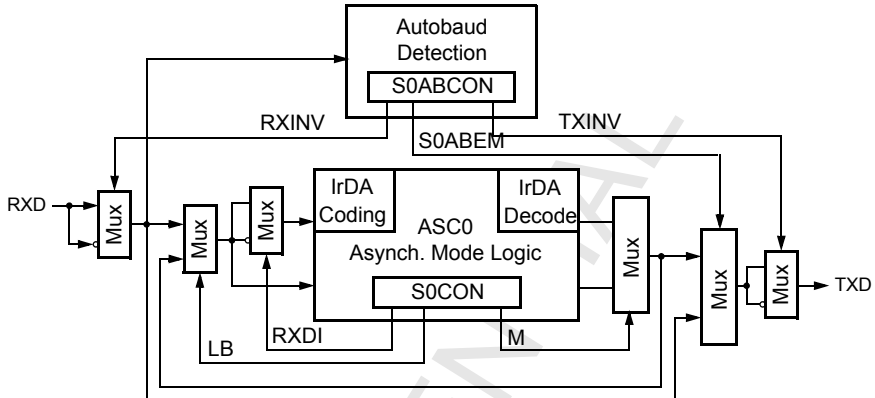
The input signal from RXD passes an inverter which is controlled by bit **S0ABCON.RXINV**. The output signal of this inverter is used for the Autobaud Detection and may bypass the logic in the Echo Mode (controlled by bit **S0ABCON.ABEM**). Further, two multiplexers are in the RXD input signal path for providing the Loopback Mode capability (controlled by bit **S0CON.LB**) and the IrDA receive pulse inversion capability (controlled by bit **S0CON.RXDI**).

CONFIDENTIAL

ASC0

Depending on the Asynchronous Mode (controlled by bit field **S0CON.M**), output signal or the RXD input signal in Echo Mode (controlled by bit **S0ABCON.ABEM**) is switched to the TXD output via an inverter (controlled by bit **S0ABCON.TXINV**).

**Figure 11-23 RXD/TXD Data Path in Asynchronous Modes**



*Note: In Echo Mode the transmit output signal is blocked by the Echo Mode output multiplexer. **Figure 11-23** shows that it is not possible to use an IrDA coded receiver input signal for Autobaud Detection.*

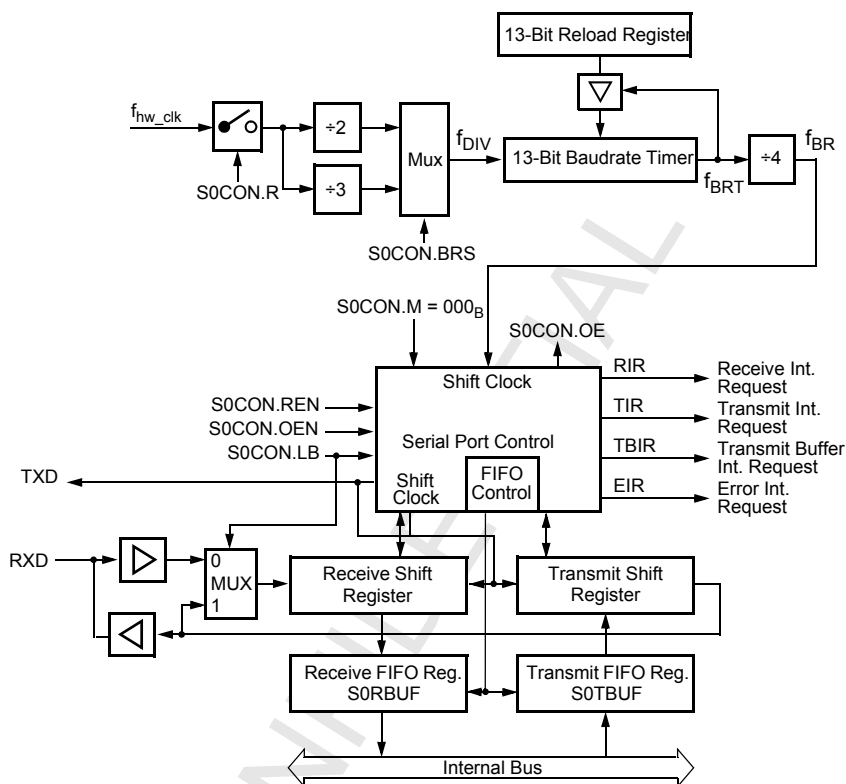
### 11.3.5.10 Synchronous Operation

Synchronous Mode supports half-duplex communication, basically for simple I/O expansion via shift registers. Data is transmitted and received via line RXD while line TXD outputs the shift clock.

Synchronous Mode is selected with **S0CON.M** = 000<sub>B</sub>.

Eight data bits are transmitted or received synchronous to a shift clock generated by the internal baudrate generator. The shift clock is active only as long as data bits are transmitted or received.

Figure 11-24 Synchronous Mode of Serial Channel ASC0



### 11.3.5.11 Synchronous Transmission

Synchronous transmission begins within four state times after data has been loaded into **S0TBUF**, provided that **S0CON.R** is set and **S0CON.REN** is cleared (half-duplex, no reception). Exception: in Loopback Mode (bit **S0CON.LB** set), **S0CON.REN** must be set for reception of the transmitted byte. Data transmission is double-buffered. When the transmitter is idle, the transmit data loaded into **S0TBUF** is immediately moved to the transmit shift register, thus freeing **S0TBUF** for more data. This is indicated by the transmit buffer interrupt request line TBIR being activated. **S0TBUF** may now be loaded with the next data, while transmission of the previous continuous. The data bits are transmitted synchronous with the shift clock. After the bit time for the eighth data bit, both the TXD and RXD lines will go high, the transmit interrupt request line TIR is activated, and serial data transmission stops.



*Note: Pin TXD must be configured for alternate data output in order to provide the shift clock. Pin RXD must also be configured for output during transmission.*

### 11.3.5.12 Synchronous Reception

Synchronous reception is initiated by setting bit **S0CON.REN**. If bit **S0CON.R** is set, the data applied at RXD is clocked into the receive shift register synchronous to the clock that is output at TXD. After the eighth bit has been shifted in, the contents of the receive shift register are transferred to the receive data buffer **S0RBUF**, the receive interrupt request line RIR is activated, the receiver enable bit **S0CON.REN** is reset, and serial data reception stops.

*Note: Pin TXD must be configured for alternate data output in order to provide the shift clock. Pin RXD must be configured as alternate data input.*

Synchronous reception is stopped by clearing bit **S0CON.REN**. A currently received byte is completed, including the generation of the receive interrupt request and an error interrupt request, if appropriate. Writing to the transmit buffer register while a reception is in progress has no effect on reception and will not start a transmission.

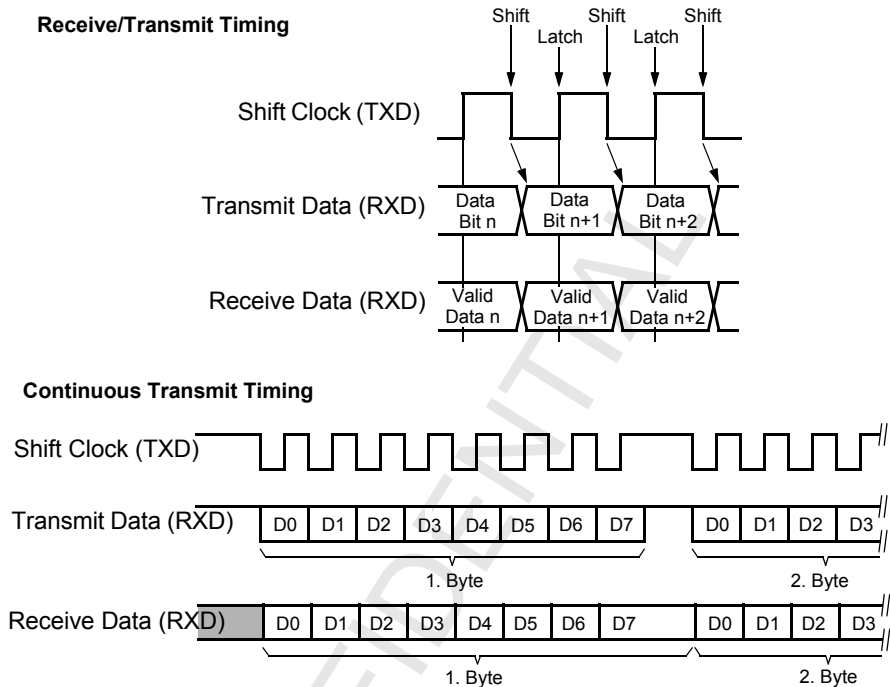
If a previously received byte has not been read out of a full receive buffer at the time the reception of the next byte is complete, both the error interrupt request line EIR and the overrun error status flag **S0CON.OE** will be activated/set, provided the overrun check has been enabled by bit **S0CON.OEN**.

### 11.3.5.13 Synchronous Timing

**Figure 11-25** shows timing diagrams of the ASC0 Synchronous Mode data reception and data transmission. In idle state, the shift clock level is high. With the beginning of a synchronous transmission of a data byte, the data is shifted out at RXD with the falling edge of the shift clock. If a data byte is received through RXD, data is latched with the rising edge of the shift clock.

Between two consecutive receive or transmit data bytes, one shift clock cycle ( $f_{BR}$ ) delay is inserted.

Figure 11-25 ASC0 Synchronous Mode Waveforms



#### 11.3.5.14 Baudrate Generation

The serial channel ASC0 has its own dedicated 13-bit baudrate generator with reload capability, allowing baudrate generation independent of other timers.

The baudrate generator is clocked with a clock ( $f_{DIV}$ ) derived via a prescaler from the ASC0 input clock  $f_{hw\_clk}$ . The baudrate timer counts downwards and can be started or stopped through the baudrate generator run bit **S0CON.R**. Each underflow of the timer provides one clock pulse to the serial channel. The timer is reloaded with the value stored in its 13-bit reload register each time it underflow. The resulting clock  $f_{BRT}$  is again divided by a factor for the baudrate clock (16 in Asynchronous Modes and 4 in Synchronous Mode). The prescaler is selected by the bits **S0CON.BRS** and **S0CON.FDE**. In addition to the two fixed dividers, a fractional divider prescaler unit is available in the Asynchronous Modes that allows selection of prescaler divider ratios of  $n/512$  with  $n = 0...511$ . Therefore, the baudrate of ASC0 is determined by the module clock, the content of **S0FDV**, the reload value of **S0BG**, and the operating mode (asynchronous or synchronous).

CONFIDENTIAL

ASC0

Register **S0BG** is the dual-function Baudrate Generator/Reload register. Reading **S0BG** returns the contents of the timer **S0BG.BR\_VALUE** (bits 15...13 return zero), while writing to **S0BG** always updates the reload register (bits 15...13 are insignificant).

An autoreload of the timer with the contents of the reload register is performed each time **S0CON.BG** is written to. However, if **S0CON.R** is cleared at the time a write operation to **S0CON.BG** is performed, the timer will not be reloaded until the first instruction cycle after **S0CON.R** was set. For a clean baudrate initialization, **S0CON.BG** should be written only if **S0CON.R** is reset. If **S0CON.BG** is written while **S0CON.R** is set, unpredictable behavior of the ASC0 may occur during running transmit or receive operations.

The ASC0 baudrate timer reload register **S0BG** contains the 13-bit reload value for the baudrate timer in Asynchronous and Synchronous modes.

*Note: IrDA Baudrate re-configuration causes unwanted pulses on the TXD PAD under following conditions:*

1. ASC0 is in IrDA Mode
2. **S0CON.R** is reset to change the Baudrate according the specification
3. ASC0 is configured
4. **S0CON.R** is set.

*During the configuration phase (point 3.) a high puls on TXD is generated, which disturbs the IrDAProtocol.*

*This problems occurs, because of the different Stop-bit active values:*

- Asynchronous Mode: HIGH-active
- IrDA Mode: LOW-active.

*When the **S0CON.R** is reseted during the reconfiguration of the Baudrate, the ASC changes to the Asynchronous Mode (--> HIGH Stop-Bit) and after reconfiguration the **S0CON.R** is set again (--> LOW Stop-Bit).*

*To avoid this Problem the ASC0 has to be decoupled during the configuration phase of the IrDA from the PAD, by using the GPIO functionality of the TXD Pin. --> Configure the according GPIO as OUTPUT and set the value to '0'.*

### 11.3.5.15 Baudrate in Asynchronous Mode

For Asynchronous Mode, the baudrate generator provides a clock  $f_{BRT}$  with sixteen times the rate of the established baudrate. Every received bit is sampled at the 7th, 8th, and 9th cycle of this clock. The clock divider circuitry, which generates the input clock for the 13-bit baudrate timer, is extended by a fractional divider circuitry that allows adjustment for more accurate baudrate and the extension of the baudrate range.

The baudrate of the baudrate generator depends on the following bits and register values:

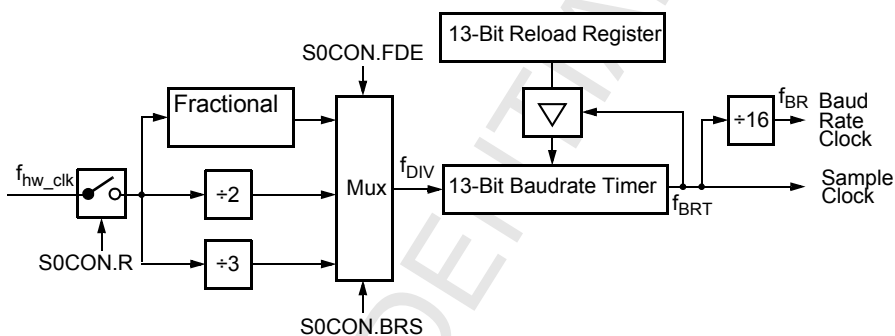
- Input clock  $f_{hw\_clk}$
- Selection of the baudrate timer input clock  $f_{DIV}$  by bits **S0CON.FDE** and **S0CON.BRS**

- If bit **S0CON.FDE** is set (fractional divider): value of register **S0FDV**
- Value of the 13-bit reload register **S0BG**.

The output clock of the baudrate timer with the reload register is the sample clock in the Asynchronous Modes of the ASC0. For baudrate calculations, this baudrate clock  $f_{BR}$  is derived from the sample clock  $f_{DIV}$  by a division by 16.

The ASC0 fractional divider register **S0FDV** contains the 9-bit divider value for the fractional divider (Asynchronous Mode only). It is also used for reference clock generation of the autobaud detection unit.

**Figure 11-26 ASC0 Baudrate Generator Circuitry in Asynchronous Modes**



S0CON.FDE	S0CON.BRS	Selected Divider
0	0	÷ 2
0	1	÷ 3
1	X	Fractional Divider

CONFIDENTIAL

ASC0

### Using the fixed Input Clock Divider

The baudrate for asynchronous operation of serial channel ASC0 when using the fixed input clock divider ratios (**S0CON.FDE** = 0) and the required reload value for a given baudrate can be determined by the following formulas:

**Table 11-11 Asynchronous Baudrate Formulas using Fixed Input Clock Dividers**

S0CON.FDE	S0CON.BRS	BG	Formula
0	0	0 ... 8191	$\text{Baudrate} = \frac{f_{\text{hw\_clk}}}{32 \times (\text{BG} + 1)}$ $\text{BG} = \frac{f_{\text{hw\_clk}}}{32 \times \text{Baudrate}} - 1$
	1		$\text{Baudrate} = \frac{f_{\text{hw\_clk}}}{48 \times (\text{BG} + 1)}$ $\text{BG} = \frac{f_{\text{hw\_clk}}}{48 \times \text{Baudrate}} - 1$

*Note: BG represents the contents of the reload register bits **S0BG.BR\_VALUE**, taken as unsigned 13-bit integer.*

**Table 11-12** lists various commonly used baudrates together with the required reload values and the deviation errors compared to the intended baudrate.

**Table 11-12 Typical Asynchronous Baudrates Using Fixed Input Clock Dividers**

Baudrate	S0CON.BRS = 0		S0CON.BRS = 1	
	Deviation Error	Reload Value	Deviation Error	Reload Value
	$f_{\text{hw\_clk}} = 26 \text{ MHz}$			
812.5 KBaud	---	0000 <sub>H</sub>	---	---
541.7 KBaud	---	---	---	0000 <sub>H</sub>
19.2 Baud	+0.8 %	0029 <sub>H</sub>	+0.8 %	001B <sub>H</sub>
9600 Baud	-0.4 %	0054 <sub>H</sub>	+0.8 %	0037 <sub>H</sub>
4800 Baud	+0.2 %	00A8 <sub>H</sub>	-0.1 %	0070 <sub>H</sub>
2400 Baud	+0.1 %	0152 <sub>H</sub>	-0.1 %	00E1 <sub>H</sub>
1200 Baud	+0.0 %	02A4 <sub>H</sub>	+0.1 %	01C2 <sub>H</sub>
110 Baud	+0.0 %	1CD9 <sub>H</sub>	+0.0 %	133B <sub>H</sub>
	$f_{\text{hw\_clk}} = 52 \text{ MHz}$			
	1625 KBaud	---	0000 <sub>H</sub>	---
	1083.3 KBaud	---	---	0000 <sub>H</sub>
	19.2 Baud	-0.4 %	0054 <sub>H</sub>	+0.8

CONFIDENTIAL

ASCO

**Table 11-12 Typical Asynchronous Baudrates Using Fixed Input Clock Dividers**

Baudrate	S0CON.BRS = 0		S0CON.BRS = 1	
	Deviation Error	Reload Value	Deviation Error	Reload Value
	$f_{hw\_clk} = 26 \text{ MHz}$			
9600 Baud	+0.2 %	00A8 <sub>H</sub>	-0.1 %	0070 <sub>H</sub>
4800 Baud	+0.1 %	0152 <sub>H</sub>	-0.1 %	00E1 <sub>H</sub>
2400 Baud	+0.0 %	02A4 <sub>H</sub>	-0.1 %	01C2 <sub>H</sub>
1200 Baud	+0.0 %	0549 <sub>H</sub>	+0.0 %	0386 <sub>H</sub>
110 Baud	+0.0 %	39B4 <sub>H</sub>	+0.0 %	2677 <sub>H</sub>

*Note: S0CON.FDE must be 0 to achieve the baudrates in Table 11-12. The deviation errors given in Table 11-12 are rounded. Using a baudrate crystal provides correct baudrates without deviation errors.*

### Using the Fractional Divider

When the fractional divider is selected, the input clock  $f_{DIV}$  for the baudrate timer is derived from the module clock  $f_{hw\_clk}$  by a programmable divider. If S0CON.FDE is set, the fractional divider is activated. It divides  $f_{hw\_clk}$  by a fraction of  $n/512$  for any value of  $n$  from 0 to 511. If  $n = 0$ , the divider ratio is 1, which means that  $f_{DIV} = f_{hw\_clk}$ . In general, the fractional divider allows the baudrate to be programmed with much more accuracy than with the two fixed prescaler divider stages.

**Table 11-13 Asynchronous Baudrate Formulas using the Fractional Input Clock Divider**

S0CON.FDE	S0CON.BRS	S0BG	S0FDV	Formula
1	-	0 ... 8191	1 ... 511	$\text{Baudrate} = \frac{\text{FDV}}{512} \times \frac{f_{\text{hw\_clk}}}{16 \times (\text{BG}+1)}$
			0	$\text{Baudrate} = \frac{f_{\text{hw\_clk}}}{16 \times (\text{BG}+1)}$

*Note: BG represents the contents of the reload register **S0BG.BR\_VALUE**, taken as an unsigned 13-bit integer.*

*FDV represents the contents of the fractional divider **S0FDV.FD\_VALUE** taken as an unsigned 9-bit integer.*

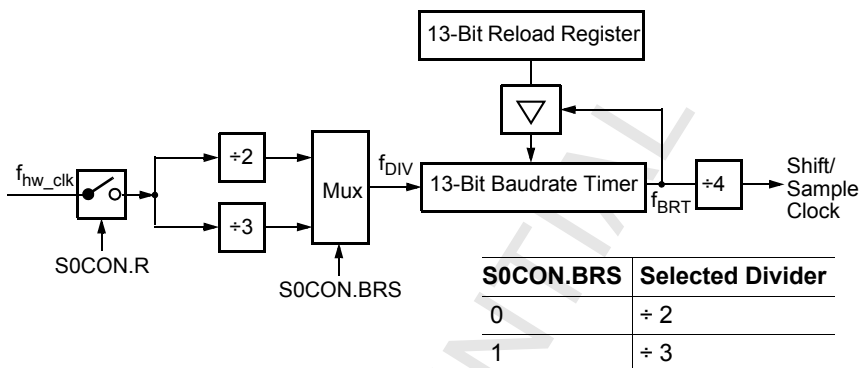
**Table 11-14 Typical Asynchronous Baudrates using the Fractional Input Clock Divider**

$f_{\text{hw\_clk}}$	Desired Baudrate	S0BG	S0FDV	Resulting Baudrate	Deviation
52 MHz	115.2 kBaud	0D <sub>H</sub>	0FE <sub>H</sub>	115.164 kBaud	0.03 %
	57.6 kBaud	0D <sub>H</sub>	07F <sub>H</sub>	57.582 kBaud	0.03 %
	38.4 kBaud	19 <sub>H</sub>	079 <sub>H</sub>	38.403 kBaud	0.01 %
	19.2 kBaud	28 <sub>H</sub>	07C <sub>H</sub>	19.175 kBaud	0.01 %

### 11.3.5.16 Baudrate in Synchronous Mode

For synchronous operation, the baudrate generator provides a clock with four times the rate of the established baudrate (see [Figure 11-27](#)).

**Figure 11-27 ASC0 Baudrate Generator Circuitry in Synchronous Mode**



The baudrate for synchronous operation of serial channel ASC0 can be determined by the formulas as shown in [Table 11-15](#).

**Table 11-15 Synchronous Baudrate Formulas**

S0CON.BRS	S0BG	Formula
0	0 ... 8191	$\text{Baudrate} = \frac{f_{hw\_clk}}{8 \times (BG+1)} \quad BG = \frac{f_{hw\_clk}}{8 \times \text{Baudrate}} - 1$
1		$\text{Baudrate} = \frac{f_{hw\_clk}}{12 \times (BG+1)} \quad BG = \frac{f_{hw\_clk}}{12 \times \text{Baudrate}} - 1$

*Note: BG represents the contents of the reload register [S0BG.BR\\_VALUE](#), taken as an unsigned 13-bit integers.*

### 11.3.5.17 Autobaud Detection

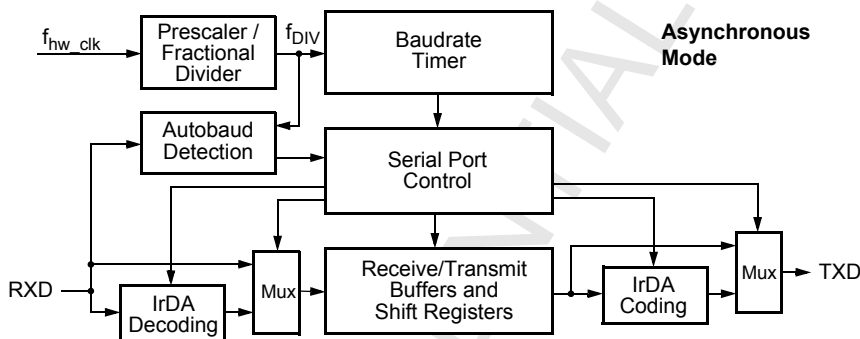
### 11.3.5.18 General Operation

Autobaud Detection provides a capability to recognize the mode and the baudrate of an asynchronous input signal at RXD. Generally, the baudrates to be recognized must be known by the application. With this knowledge always a set of nine baudrates can be detected. The Autobaud Detection is not designed to calculate a baudrate of an unknown asynchronous frame.



**Figure 11-28** shows how the Autobaud Detection is integrated into its Asynchronous Mode configuration. The RXD data line is an input to the autobaud detection unit. The clock  $f_{DIV}$ , generated by the fractional divider, is used by the autobaud detection unit as time base. After successful recognition of baudrate and Asynchronous Mode of the RXD data input signal, bits in the **S0CON** register and the value of the **S0BG** register in the baudrate timer are set to the appropriate values, and the ASC0 can start immediately with the reception of serial input data.

**Figure 11-28 Asynchronous Mode Block Diagram**



*Note: Autobaud detection is not available in Synchronous Mode*

The following sequence must be generally executed to start the autobaud detection unit for operation:

- Definition of the baudrates to be detected: standard or non-standard baudrates
- Programming of the prescaler/fractional divider to select a specific value of  $f_{DIV}$
- Starting the prescaler/fractional divider (setting **S0CON.R**)
- Preparing the interrupt system
- Enabling the autobaud detection (setting **S0ABCON.EN** and the interrupt enable bits in **S0ABCON** for interrupt generation, if required)
- Polling interrupt request flag or waiting for the autobaud detection interrupt

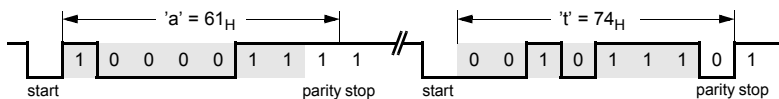
### 11.3.5.19 Serial Frames for Autobaud Detection

The Autobaud Detection is based on the serial reception of a specific two-byte serial frame. This serial frame is build up by the two ASCII bytes “at” or “AT” (“aT” or “At” are not allowed). Both byte combinations can be detected in five types of asynchronous frames. **Figure 11-29** and **Figure 11-30** show the serial frames which are detected at least.

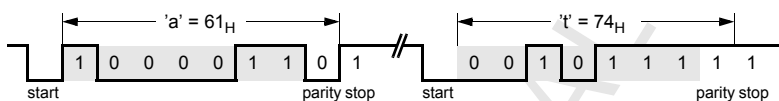
*Note: Some other two-byte combinations will be defined too.*

**Figure 11-29 Two-Byte Serial Frames with ASCII 'at'**

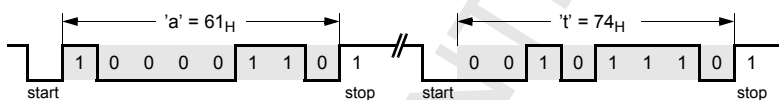
**7 bit, even parity**



**7 bit, odd parity**



**8 bit, no parity**



**8 bit, even parity**



**8 bit, odd parity**

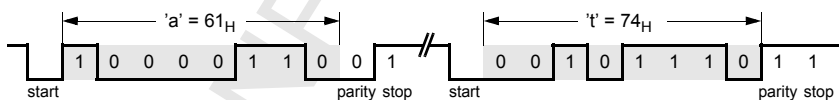
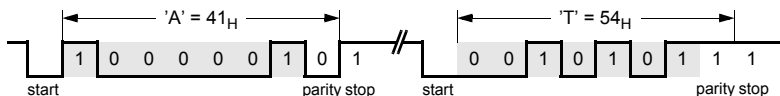
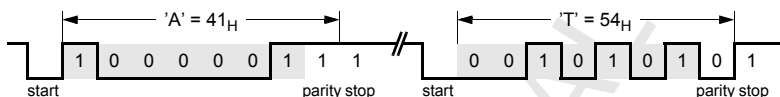


Figure 11-30 Two-Byte Serial Frames with ASCII 'AT'

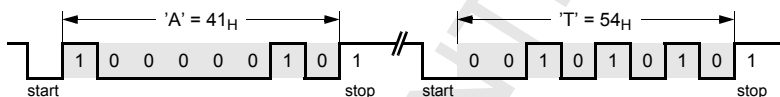
7 bit, even parity



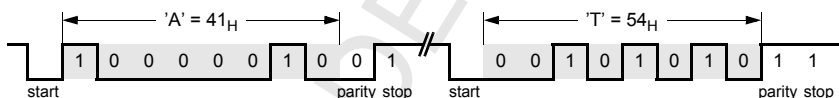
7 bit, odd parity



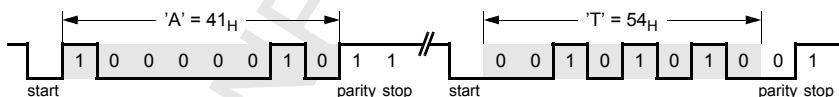
8 bit, no parity



8 bit, even parity



8 bit, odd parity



### 11.3.5.20 Baudrate Selection and Calculation

Autobaud Detection requires some calculations concerning the programming of the baudrate generator and the baudrates to be detected. Two steps must be considered:

- Defining the baudrate(s) to be detected
- Programming of the baudrate timer prescaler - setup of the clock rate of  $f_{DIV}$ .

In general, the baudrate generator in Asynchronous Mode is build up by two parts (see also [Figure 11-26](#)):

- The clock prescaler part which derives  $f_{DIV}$  from  $f_{hw\_clk}$
- The baudrate timer part which generates the sample clock  $f_{BRT}$  and the baudrate clock  $f_{BR}$ .

CONFIDENTIAL

ASC0

Prior to an Autobaud Detection the prescaler part has to be setup by the CPU while the baudrate timer is initialized with a 13-bit value (**S0BG.BR\_VALUE**) automatically after a successful autobaud detection. For the following calculations, the fractional divider is used (**S0CON.FDE** = 1).

*Note: It is also possible to use the fixed divide-by-2 or divide-by-3 prescaler. But the fractional divider allows to adapt  $f_{DIV}$  much more precise to the required value.*

## Standard Baudrates

For standard baudrate detection the baudrates as shown in **Table 11-16** can be e.g. detected. Therefore, the output frequency  $f_{DIV}$  of the baudrate generator must be set to a frequency derived from the module clock  $f_{hw\_clk}$  in a way that it is equal to 11.0592 MHz. The value to be written into register **S0FDV** is the nearest integer value which is calculated according the following formula:

$$FDV = \frac{512 \times 11.0592 \text{ MHz}}{f_{MOD}}$$

**Table 11-16** defines the nine standard baudrates (Br0 - Br8) which can be detected for  $f_{DIV} = 11.0592 \text{ MHz}$ .

**Table 11-16 Autobaud Detection using Standard Baudrates ( $f_{DIV} = 11.0592 \text{ MHz}$ )**

Baudrate Numbering	Detectable Standard Baudrate	Divide Factor $d_f$	BG is loaded after detection with value
Br0	230.400 kBaud	48	2 = 002 <sub>H</sub>
Br1	115.200 kBaud	96	5 = 005 <sub>H</sub>
Br2	57 600 kBaud	192	11 = 00B <sub>H</sub>
Br3	38 400 kBaud	288	17 = 011 <sub>H</sub>
Br4	19 200 kBaud	576	35 = 023 <sub>H</sub>
Br5	9600 Baud	1152	71 = 047 <sub>H</sub>
Br6	4800 Baud	2304	143 = 08F <sub>H</sub>
Br7	2400 Baud	4608	287 = 11F <sub>H</sub>
Br8	1200 Baud	9216	575 = 23F <sub>H</sub>

According **Table 11-16** a baudrate of 9600 Baud is achieved when register BG is loaded with a value of 047<sub>H</sub>, assuming that  $f_{DIV}$  has been set to 11.0592 MHz.

**Table 11-16** also lists a divide factor  $d_f$  which is defined with the following formula:

$$\text{Baudrate} = \frac{f_{DIV}}{d_f}$$

CONFIDENTIAL

ASC0

This divide factor  $d_f$  defines a fixed relationship between the prescaler output frequency  $f_{DIV}$  and the baudrate to be detected during the Autobaud Detection operation. This means, changing  $f_{DIV}$  results in a totally different baudrate table in means of baudrate values. For the baudrates to be detected, the following relations are always valid:

$$Br0 = f_{DIV} / 48_D, Br1 = f_{DIV}/96_D, \dots\dots \text{up to } Br8 = f_{DIV}/9216_D,$$

A requirement for detecting standard baudrates up to 230 400 kBaud is the  $f_{DIV}$  minimum value of 11.0592 MHz. With the value **S0FDV.FD\_VALUE** the fractional divider  $f_{DIV}$  is adapted to the module clock frequency  $f_{MOD}$ . **Table 11-17** defines the deviation of the standard baudrates when using autobaud detection depending on the module clock  $f_{hw\_clk}$ .

**Table 11-17 Standard Baudrates - Deviations and Errors for Autobaud Detection**

$f_{hw\_clk}$	S0FDV	Error in $f_{DIV}$
10 MHz		not possible
12 MHz	472	+ 0.03 %
13 MHz	436	+ 0.1 %
16 MHz	354	+ 0.03 %
18 MHz	315	+ 0.14 %
18.432 MHz	307	- 0.07 %
20 MHz	283	- 0.04 %
24 MHz	236	+ 0.03 %
25 MHz	226	- 0.22 %
26 MHz	218	+ 0.10 %
30 MHz	189	+ 0.14 %
33 MHz	172	+ 0.24 %
40 MHz	142	+ 0.31 %
50 MHz	113	- 0.23 %
52 MHz	109	- 0.10 %

*Note: If the deviation of the baudrate after autobaud detection is to high, the baudrate generator (fractional divider **S0FDV** and reload register **S0BG**) can be reprogrammed if required to get a more precise baudrate with less error.*

### Non-Standard Baudrates

Due to the relationship between Br0 to Br8 in **Table 11-16** concerning the divide factor  $d_f$  other baudrates than the standard baudrates can be also selected. For example, if a

CONFIDENTIAL

ASC0

baudrate of 50 kBaud has to be detected, Br2 is defined as baudrate for the 50 kBaud selection. This further results in:

$$f_{DIV} = 50 \text{ kBaud} \times d_f @ Br2 = 50 \text{ kBaud} \times 192 = 9.6 \text{ MHz}$$

Therefore, depending on the module clock frequency  $f_{hw\_clk}$ , the value of the fractional divider (register **S0FDV** must be set in this example according the formula:

$$FDV = \frac{512 \times f_{DIV}}{f_{hw\_clk}} \quad \text{with } f_{DIV} = 9.6 \text{ MHz}$$

Using this selection ( $f_{DIV} = 9.6 \text{ MHz}$ ), the detectable baudrates start at 200 kBaud (Br0) down to 1042 Baud (Br8). **Table 11-18** shows the baudrate table for this example.

**Table 11-18 Autobaud Detection using Non-Standard Baudrates ( $f_{DIV} = 9.6 \text{ MHz}$ )**

Baudrate Numbering	Detectable Non-Standard Baudrates	Divide Factor $d_f$	BG is loaded after detection with value
Br0	200.000 kBaud	48	2 = 002 <sub>H</sub>
Br1	100.000 kBaud	96	5 = 005 <sub>H</sub>
Br2	50 kBaud	192	11 = 00B <sub>H</sub>
Br3	33.333 kBaud	288	17 = 011 <sub>H</sub>
Br4	16.667 kBaud	576	35 = 023 <sub>H</sub>
Br5	8333 Baud	1152	71 = 047 <sub>H</sub>
Br6	4167 Baud	2304	143 = 08F <sub>H</sub>
Br7	2083 Baud	4608	287 = 11F <sub>H</sub>
Br8	1047 Baud	9216	575 = 23F <sub>H</sub>

### 11.3.5.21 Overwriting Registers on Successful Autobaud Detection

With a successful Autobaud Detection some bits in register **S0CON** and **S0BG** are automatically set to a value which corresponds to the mode and baudrate of the detected serial frame conditions (see **Table 11-19**). In control register **S0CON** the mode control bits **S0CON.M** and the parity select bit **S0CON.ODD** are overwritten. Register **S0BG** is loaded with the 13-bit reload value for the baudrate timer.

**Table 11-19 Autobaud Detection Overwrite Values for the CON Register**

Detected Parameters		S0CON.M	S0CON.ODD	BG.BR_VALUE
Operating Mode	7 bit, even parity	0 1 1	0	-
	7 bit, odd parity	0 1 1	1	-
	8 bit, even parity	1 1 1	0	-
	8 bit, odd parity	1 1 1	1	-
	8 bit, no parity	0 0 1	0	-
Baudrate	Br0	-	-	2 = 002 <sub>H</sub>
	Br1	-	-	5 = 005 <sub>H</sub>
	Br2	-	-	11 = 00B <sub>H</sub>
	Br3	-	-	17 = 011 <sub>H</sub>
	Br4	-	-	35 = 023 <sub>H</sub>
	Br5	-	-	71 = 047 <sub>H</sub>
	Br6	-	-	143 = 08F <sub>H</sub>
	Br7	-	-	287 = 11F <sub>H</sub>
	Br8	-	-	575 = 23F <sub>H</sub>

*Note: The autobaud detection interrupts are described in [Section 11.3.5.25 Interrupts \(on Page 1044\)](#).*

### 11.3.5.22 Hardware Flow Control

The ASC0 supports both software- and hardware-controlled flow control with Request to Send (RTS)/Clear to Send (CTS) handshaking. Flow control is only available in asynchronous mode. In software mode (**S0FCCON.RTSEN** = 0), the handshake line RTS\_N is controlled by the bit **S0FCCON.RTS**. In hardware mode (**S0FCCON.RTSEN** = 1), RTS is controlled depending on whether the receive FIFO is enabled (**S0RXFCON.RXFEN** = 1) or not. If the receive FIFO is enabled, RTS is active (RTS\_N = 0) as long as the receive FIFO level is below the programmable RTS trigger level **S0FCCON.RTSTL**. RTS is deactivated (RTS\_N = 1), if the receive FIFO level is equal to or greater than the RTS trigger level. The RXFIFO is described in [Section 11.3.5.6 Receive FIFO Operation \(on Page 1022\)](#). If the receive FIFO is disabled, RTS is active as long as the receive buffer is empty and inactive as soon as a frame is in the receive buffer, which has not been read yet. If automatic CTS recognition is enabled (**S0FCCON.CTSEN** = 1), data frames are transmitted only as long as the handshake line CTS is active. If CTS is deactivated while a frame is being transmitted, the transmission of this frame will still be completed. Independent of the CTS enable bit **S0FCCON.CTSEN**, the status of the CTS line can be monitored by reading the CTS status bit **S0FSTAT.CTS**. Also, the interrupt CTS\_INT indicates a change on the handshake line CTS\_N (see [Section 11.3.5.25 \(on page 1044\)](#)). In loop back mode (**S0CON.LB** = 1), RTS\_N is connected to CTS\_N internally.

### 11.3.5.23 Hardware Error Detection Capabilities

To improve the safety of serial data exchange, the serial channel ASC0 provides an error interrupt request flag to indicate the presence of an error, and three (selectable) error status flags in register **S0CON** to indicate which error has been detected during reception. Upon completion of a reception, the error interrupt request line EIR will be activated simultaneously with the receive interrupt request line RIR, if one or more of the following conditions are met:

- If the framing error detection enable bit **S0CON.FEN** is set and any of the expected stop bits is not high, the framing error flag **S0CON.FE** is set, indicating that the error interrupt request is due to a framing error (Asynchronous Mode only).
- If the parity error detection enable bit **S0CON.PEN** is set in the modes where a parity bit is received, and the parity check on the received data bits proves false, the parity error flag **S0CON.PE** is set, indicating that the error interrupt request is due to a parity error (Asynchronous Mode only).
- If the overrun error detection enable bit **S0CON.OEN** is set and the last character received was not read out of the receive buffer by software at the time the reception of a new frame is complete, the overrun error flag **S0CON.OE** is set indicating that the error interrupt request is due to an overrun error (Asynchronous and Synchronous Mode).

### 11.3.5.24 Receive Timeout Detection

A receive timeout detection functionality is provided to detect timeout conditions while the ASC0 is operating in reception mode. The submodule consists of a 16-bit timer and the timeout control register **S0TMO**. If the timeout control register is set to zero, the timeout detection is disabled. If the timeout detection is enabled, the timer is loaded with the contents of the timeout control register after the reception of the first character. Then the timer is decremented with each shift clock cycle generated by the baud rate generator. The timer is reloaded after the reception of each character. If the timer reaches zero, a timeout interrupt is generated. Thereby the timeout module has an inter-character-gap timeout functionality.

### 11.3.5.25 Interrupts

Eight interrupts sources are provided for serial channel ASC0:

- TIR indicates a Transmit Interrupt
- TBIR indicates a Transmit Buffer Interrupt
- RIR indicates a Receive Interrupt
- EIR indicates an Error Interrupt of the serial channel
- CTSIR indicates a change on the handshake line CTS\_N
- TMOIR indicates a receive timeout condition.



## CONFIDENTIAL

## ASC0

The autobaud detection unit provides two additional interrupts, the ABSTIR start of autobaud operation interrupt and the ABDETIR autobaud detected interrupt.

The interrupt output lines TBIR, TIR, RIR, EIR, CTSIR, TMOIR, ABSTIR, and ABDETIR are activated (active state) for two periods of the module clock  $f_{hw\_clk}$ .

The cause of an error interrupt request (framing, parity, overrun error) can be identified by the error status flags **S0CON.FE**, **S0CON.PE**, and **S0CON.OE**. For the two autobaud detection interrupts register **S0ABSTAT** provides status information.

*Note: Unlike the error interrupt request line EIR, the error status flags **S0CON.FE**, **S0CON.PE**, and **S0CON.OE** are not reset automatically but must be cleared by software.*

For normal operation (that is, besides the error interrupt) the ASC0 provides three interrupt requests to control data exchange via this serial channel:

- TBIR is activated when data is moved from **S0TBUF** to the transmit shift register.
- TIR is activated before the last bit of an asynchronous frame is transmitted, or after the last bit of a synchronous frame has been transmitted.
- RIR is activated when the received frame is moved to **S0RBUF**.

*Note: While the receive task is handled by a single interrupt handler, the transmitter is serviced by two interrupt handlers. This provides advantages for the servicing software.*

For single transfers it is sufficient to use the transmitter interrupt (TIR), which indicates that the previously loaded data has been transmitted, except for the last bit of an asynchronous frame.

For multiple back-to-back transfers it is necessary to load the following piece of data at last until the time the last bit of the previous frame has been transmitted. In Asynchronous Mode this leaves just one bit-time for the handler to respond to the transmitter interrupt request, in Synchronous Mode it is impossible at all.

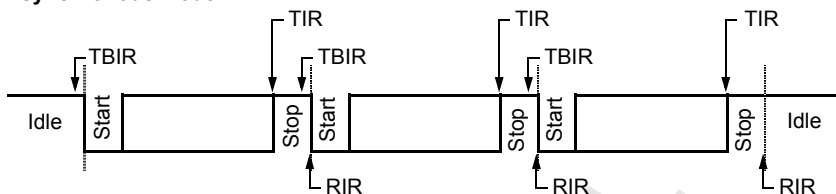
Using the transmit buffer interrupt (TBIR) to reload transmit data gives the time to transmit a complete frame for the service routine, as **S0TBUF** may be reloaded while the previous data is still being transmitted.

The start of autobaud operation interrupt ABSTIR is generated whenever the autobaud detection unit is enabled (**S0ABCON.ABEN** and **S0ABCON.ABDETEN** and **S0ABCON.ABSTEN** are set), and a start bit has been detected at RXD. In this case ABSTIR is generated during Autobaud Detection whenever a start bit is detected.

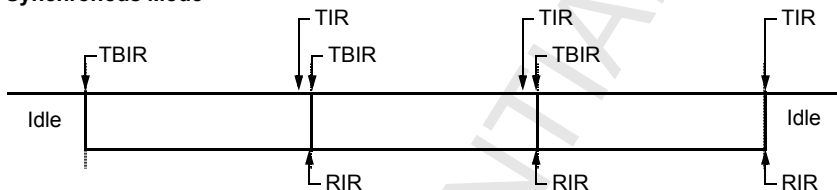
The autobaud detected interrupt ABDETIR is always generated after recognition of the second character of the two-byte frame, this means after a successful Autobaud Detection. If **S0ABCON.FCDETEN** is set the autobaud detected interrupt ABDETIR is also generated after the recognition of the first character of the two-byte frame.

Figure 11-31 ASC0 Interrupt Generation

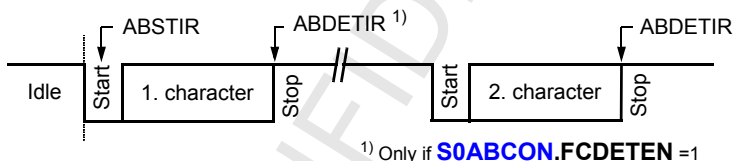
**Asynchronous Mode**



**Synchronous Mode**



**Asynchronous Modes  
Autobaud Detection**



<sup>1)</sup> Only if `S0ABCON.FCDETEN` = 1

As shown in [Figure 11-31](#), TBIR is an early trigger for the reload routine, while TIR indicates the completed transmission. Therefore, software using handshake should rely on TIR at the end of a data block to ensure that all data has actually been transmitted.

**CONFIDENTIAL**

**ASC0**

### 11.3.6 Registers

For the register addresses refer to [Section 12.2 PD-Bus Register Addresses \(on Page 1273\)](#).

**Table 11-20 ASC0 Register Summary**

Name	Clock	Access Condition	Description
<a href="#">S0PISEL</a>	hw_clk <sup>1)</sup>	bit addressable	Peripheral Input Select Register
<a href="#">S0PERID</a>	hw_clk <sup>1)</sup>	non bit addressable	Peripheral ID Number Register
<a href="#">S0CON</a>	hw_clk <sup>1)</sup>	bit addressable	Control Register
<a href="#">S0BG</a>	hw_clk <sup>1)</sup>	non bit addressable	Baudrate Timer Reload Register
<a href="#">S0FDV</a>	hw_clk <sup>1)</sup>	non bit addressable	Fractional Divider Register
<a href="#">S0PWM</a>	hw_clk <sup>1)</sup>	non bit addressable	IrDA Pulse Mode and Width Register
<a href="#">S0TBUF</a>	hw_clk <sup>1)</sup>	non bit addressable	Transmit Buffer Register
<a href="#">S0RBUF</a>	hw_clk <sup>1)</sup>	non bit addressable	Receive Buffer Register
<a href="#">S0ABCON</a>	hw_clk <sup>1)</sup>	bit addressable	Autobaud Control Register
<a href="#">S0ABSTAT</a>	hw_clk <sup>1)</sup>	non bit addressable	Autobaud Status Register
<a href="#">S0RXFCON</a>	hw_clk <sup>1)</sup>	non bit addressable	Receive FIFO Control Register
<a href="#">S0TXFCON</a>	hw_clk <sup>1)</sup>	non bit addressable	Transmit FIFO Control Register
<a href="#">S0FSTAT</a>	hw_clk <sup>1)</sup>	non bit addressable	FIFO Status Register
<a href="#">S0FCCON</a>	hw_clk <sup>1)</sup>	bit addressable	Flowcontrol Control Register
<a href="#">S0FCSTAT</a>	hw_clk <sup>1)</sup>	bit addressable	Flowcontrol Status Register
<a href="#">S0TMO</a>	hw_clk <sup>1)</sup>	non bit addressable	RX Timeout Control Register

<sup>1)</sup> Refer to the Clock Domain in the [System Integration: \(on Page 1011\)](#).

**CONFIDENTIAL**

**ASC0**

### 11.3.6.1 Port Input Select Register

The **S0PISEL** register selects the source to receiver data (RXD) of the ASC0.

**S0PISEL.RIS** must remain at the reset default value of 0 to select RXD. This ensures correct operation in Synchronous and Asynchronous modes.

#### **S0PISEL**

##### **Port Input Select Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															RIS

Field	Bits	Type	Description
<b>RIS</b>	0	rw	<b>Receiver Input Select</b> 0 RXD pin selected (default) 1 Tied to logical 0, no input received
<b>RESERVED</b>	15:1	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**ASC0**

### 11.3.6.2 ASC0 Identification Register

The **S0PERID** register contains the ASC0 Module Number, Configuration Number, and Revision Number.

#### **S0PERID**

**Peripheral Identification Register**

**Reset values: 44E4<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>MOD_NUMBER</b>								<b>CFG</b>			<b>MOD_REV</b>				

Field	Bits	Type	Description
<b>MOD_REV</b>	4:0	r	<b>ASC0 Revision Number</b> Value = 03 <sub>H</sub>
<b>CFG</b>	7:5	r	<b>Configuration Number</b> Bit 7 = 1 because Autobaud mode is implemented Bit 6 = 1 because Irda mode is implemented Bit 5 = 1 because FIFO mode is implemented Value = 7 <sub>H</sub>
<b>MOD_NUMBER</b>	15:8	r	<b>ASC0 Number</b> For ASC0, it is 44 <sub>H</sub>

**CONFIDENTIAL**

**ASC0**

### 11.3.6.3 Control Register

The operating mode of the serial channel ASC0 is controlled by its control register. This register contains control bits for mode and error check selection, and status flags for error identification.

#### S0CON

#### Control Register

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LB	BRS	ODD	FDE	OE	FE	PE	OEN	FEN	PEN/RXDI	REN	STP		M	

Field	Bits	Type	Description
<b>M</b>	2:0	rw	<b>Mode Control</b> 000 8-bit-data for synchronous operation 001 8-bit-data for asynchronous operation 010 8-bit-data IrDA Mode for asynchronous operation 011 7-bit-data and parity for asynchronous operation 100 9-bit-data for asynchronous operation 101 8-bit-data and wake up bit for asynchronous operation 110 Reserved. Do not use this combination 111 8-bit-data and parity for asynchronous operation
<b>STP</b>	3	rw	<b>Number of Stop Bits Selection</b> 0 One stop bit 1 Two stop bits
<b>REN</b>	4	rwh	<b>Receiver Enable Bit</b> 0 Receiver disabled 1 Receiver enabled  <i>Note: Bit is cleared by hardware after reception of a byte in the Synchronous Mode</i>

**CONFIDENTIAL**

**ASC0**

Field	Bits	Type	Description
<b>PEN</b>	5	rw	Any modes without the IrDA Mode <b>Parity Check Enable</b> 0 Ignore parity 1 Check parity
<b>RXDI</b>			Only in IrDA Mode <b>RXDI Invert in IrDA Mode</b> 0 RXD input is not inverted 1 RXD input is inverted
<b>FEN</b>	6	rw	<b>Framing Check Enable</b> (Asynchronous Mode only) 0 Ignore framing errors 1 Check framing errors
<b>OEN</b>	7	rw	<b>Overrun Check Enable</b> 0 Ignore overrun errors 1 Check overrun errors
<b>PE</b>	8	rwh	<b>Parity Error Flag</b> Set by hardware on a parity error ( <b>PEN</b> = 1). Must be cleared by software.
<b>FE</b>	9	rwh	<b>Framing Error Flag</b> Set by hardware on a framing error ( <b>FEN</b> = 1). Must be cleared by software.
<b>OE</b>	10	rwh	<b>Overrun Error Flag</b> Set by hardware on an overrun/underflow error ( <b>OEN</b> = 1). Must be cleared by software.
<b>FDE</b>	11	rw	<b>Fractional Divider Enable</b> 0 Fractional divider disabled 1 Fractional divider enabled and used as perscaler for baudrate generator (bit BRS is 'don't care')
<b>ODD</b>	12	rw	<b>Parity Selection</b> 0 Even parity selected (parity bit of 1 is included in data stream on odd number of 1 and parity bit of 0 is included in data stream on even number of 1) 1 Odd parity selected (parity bit of 1 is included in data stream on even number of 1 and parity bit of 0 is included in data stream on odd number of 1)

**CONFIDENTIAL**

**ASC0**

Field	Bits	Type	Description
<b>BRS</b>	13	rw	<b>Baudrate Selection</b> 0 Baud rate timer prescaler divide-by-2 selected 1 Baud rate timer prescaler divide-by-3 selected <i>Note: <b>BRS</b> is 'don't care' if <b>FDE</b> = 1 (fractional divider selected)</i>
<b>LB</b>	14	rw	<b>Loopback Mode Enabled</b> 0 Loopback Mode disabled. Standard transmit/receive Mode 1 Loopback Mode enabled
<b>R</b>	15	rw	<b>Baudrate Generator Run Control Bit</b> 0 Baudrate generator disabled (ASC0 inactive) 1 Baudrate generator enabled <i>Note: <b>S0BG.BR_VALUE</b> should only be written if <b>R</b> = 0.</i>

#### 11.3.6.4 Baudrate Register

The ASC0 baudrate timer reload register contains the 13-bit reload value for the baudrate timer in Asynchronous and Synchronous Mode.

##### **S0BG**

##### **Baudrate Timer/Reload Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			BR_VALUE												

Field	Bits	Type	Description
<b>BR_VALUE</b>	12:0	rw	<b>Baudrate Timer/Reload Value</b> <i>Note: <b>BR_VALUE</b> should only be written if <b>S0CON.R</b> = 0.</i>
<b>RESERVED</b>	15:13	r	Reserved; these bits must be left at their reset values.



CONFIDENTIAL

ASC0

### 11.3.6.5 Fractional Divider Register

The ASC0 fractional divider register contains the 9-bit divider value for the fractional divider (Asynchronous Mode only). It is also used for reference clock generation of the autobaud detection unit.

#### S0FDV

#### Fractional Divider Register

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED									FD_VALUE						

Field	Bits	Type	Description
FD_VALUE	8:0	rw	<b>Fractional Divider Register Value</b> <b>FD_VALUE</b> contains the 9-bit value of the fractional divider which defines the fractional divider ratio $n/512$ ( $n = 0-511$ ). With $n = 0$ , the fractional divider is switched off (input = output frequency, $f_{DIV} = f_{hw\_clk}$ , see <a href="#">Figure 11-26 ASC0 Baudrate Generator Circuitry in Asynchronous Modes (on Page 1032)</a> ).
RESERVED	15:9	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**ASC0**

### 11.3.6.6 IrDA Pulse Mode/Width Register

The ASC0 IrDA pulse mode and width register contains the 8-bit IrDA pulse width value and the IrDA pulse width mode select bit. This register is only required in the IrDA operating mode.

#### S0PWM

#### IrDA Pulse Mode/Width Register

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED							IRP W	PW_VALUE							

Field	Bits	Type	Description
<b>PW_VALUE</b>	7:0	rw	<b>IrDA Pulse Width Value</b> <b>PW_VALUE</b> is the 8-bit value n, which defines the variable pulse width of an IrDA pulse. Depending on the ASC0 input frequency $f_{hw\_clk}$ , this value can be used to adjust the IrDA pulse width to value which is not equal 3/16 bit time (e.g. 1.6 ms).
<b>IRPW</b>	8	rw	<b>IrDA Pulse Width Selection</b> 0 IrDA pulse width is 3/16 bit time 1 IrDA pulse width is defined by <b>PW_VALUE</b>
<b>RESERVED</b>	15:9	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**ASC0**

### 11.3.6.7 Transmitter Buffer Register

The ASC0 transmitter buffer register contains the transmit data value in Asynchronous and Synchronous Mode.

#### S0TBUF

#### Transmit Buffer Register

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								TD_VALUE							

Field	Bits	Type	Description
<b>TD_VALUE</b>	8:0	rw	<b>Transmit Data Register Value</b> <b>TD_VALUE</b> contains the data to be transmitted in asynchronous and synchronous operating mode of the ASC0. Data transmission is double buffered, Therefore, a new value can be written to <b>TD_VALUE</b> before the transmission of the previous value is complete.
<b>RESERVED</b>	15:9	r	Reserved; these bits must be left at their reset values.

CONFIDENTIAL

ASC0

### 11.3.6.8 Receiver Buffer Register

The ASC0 Receiver buffer register contains the transmit data value in Asynchronous and Synchronous Modes.

#### S0RBUF

#### Receive Buffer Register

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								RD_VALUE							

Field	Bits	Type	Description
RD_VALUE	8:0	rw	<b>Receive Data Register Value</b> <b>RD_VALUE</b> contains the received data bits and, depending on the selected mode, the parity bit in asynchronous and synchronous operating mode of the ASC0. In asynchronous operating mode if <b>S0CON.M</b> = 011 (7-bit data + parity), the received parity bit is written into <b>RD_VALUE[7]</b> . In asynchronous operating mode if <b>S0CON.M</b> = 111 (8-bit data + parity), the received parity bit is written into <b>RD_VALUE[8]</b> .
RESERVED	15:9	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**ASC0**

### 11.3.6.9 Autobaud Control Register

The autobaud control register of the ASC\_P module is used to control the autobaud detection operation. It contains its general enable bit, the interrupt enable control bits, and data path control bits.

#### **S0ABCON**

#### **Autobaud Control Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				RX INV	TX INV	ABEM		RESERVED			FC DET EN	AB DET EN	ABS T EN	AUR EN	AB EN

Field	Bits	Type	Description
<b>ABEN</b>	0	rwh	<b>Autobaud Detection Enable</b> 0 Autobaud detection is disabled 1 Autobaud detection is enabled  <i>Note: <b>ABEN</b> is reset by hardware after a successful Autobaud Detection; (with the stop bit detection of the second character). Resetting <b>ABEN</b> by software if it was set aborts the Autobaud Detection.</i>
<b>AUREN</b>	1	rw	<b>Automatic Autobaud Control of S0CON.REN</b> 0 <b>S0CON.REN</b> is not affected during autobaud detection 1 <b>S0CON.REN</b> is cleared (receiver disabled) when <b>ABEN</b> and <b>AUREN</b> are set together. <b>S0CON.REN</b> is set (receiver enabled) after a successful Autobaud Detection (with the stop bit detection of the second character)
<b>ABSTEN</b>	2	rw	<b>Start of Autobaud Detection Interrupt Enable</b> 0 Start of Autobaud Detection interrupt disabled 1 Start of Autobaud Detection interrupt enabled
<b>ABDETEN</b>	3	rw	<b>Autobaud Detection Interrupt Enable</b> 0 Autobaud Detection interrupt disabled 1 Autobaud Detection interrupt enabled

**CONFIDENTIAL**

**ASC0**

Field	Bits	Type	Description
<b>FCDETEN</b>	4	rw	<b>First Character of Two-Byte Frame Detected Enable</b> 0 Autobaud Detection interrupt ABDETIR becomes active after the two-byte frame recognition 1 Autobaud Detection interrupt ABDETIR becomes active after detection of the first <u>and</u> second byte of the two-byte frame
<b>ABEM</b>	9:8	rw	<b>Autobaud Echo Mode Enable</b> In Echo Mode the serial data at RXD is switched to TXD output. 00 Echo Mode disabled 01 Echo Mode is enabled during Autobaud Detection 10 Echo Mode is always enabled 11 Reserved; do not use this combination
<b>TXINV</b>	10	rw	<b>Transmit Inverter Enable</b> 0 Transmit inverter disabled 1 Transmit inverter enabled
<b>RXINV</b>	11	rw	<b>Receive Inverter Enable</b> 0 Receive inverter disabled 1 Receive inverter enabled
<b>RESERVED</b>	7:5, 15:12	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**ASC0**

### 11.3.6.10 Autobaud Status Register

The autobaud status register **S0ABSTAT** of the ASC\_P module indicates the status of the autobaud detection operation.

#### **S0ABSTAT**

**Autobaud Status Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>											<b>DET WA IT</b>	<b>SCC DET</b>	<b>SCS DET</b>	<b>FCC DET</b>	<b>FCS DET</b>

Field	Bits	Type	Description
<b>FCSDet</b>	0	rwh	<b>First Character with Small Letter Detected</b> 0 No small 'a' character detected 1 Small 'a' character detected Bit is cleared by hardware when <b>S0ABCON.ABEN</b> is set or if <b>FCCDET</b> or <b>SCSDet</b> or <b>SCCDET</b> is set. Bit can be also cleared by software.
<b>FCCDET</b>	1	rwh	<b>First Character with Capital Letter Detected</b> 0 No capital 'A' character detected 1 Capital 'A' character detected Bit is cleared by hardware when <b>S0ABCON.ABEN</b> is set or if <b>FCSDet</b> or <b>SCSDet</b> or <b>SCCDET</b> is set. Bit can be also cleared by software.
<b>SCSDet</b>	2	rwh	<b>Second Character with Small Letter Detected</b> 0 No small 't' character detected 1 Small 't' character detected Bit is cleared by hardware when <b>S0ABCON.ABEN</b> is set or if <b>FCSDet</b> or <b>FCCDET</b> or <b>SCCDET</b> is set. Bit can be also cleared by software.
<b>SCCDET</b>	3	rwh	<b>Second Character with Capital Letter Detected</b> 0 No capital 'T' character detected 1 Capital 'T' character detected Bit is cleared by hardware when <b>S0ABCON.ABEN</b> is set or if <b>FCSDet</b> or <b>FCCDET</b> or <b>SCSDet</b> is set. Bit can be also cleared by software.

CONFIDENTIAL

ASC0

Field	Bits	Type	Description
<b>DEWAIT</b>	4	rwh	<b>Autobaud Detection is Waiting</b> 0 Either character 'a', 'A', 't', or 'T' has been detected. 1 The autobaud detection unit waits for the first 'a' or 'A' Bit is cleared when either <b>FCSDET</b> or <b>FCCDET</b> is set ('a' or 'A' detected). Bit can be also cleared by software. <b>DEWAIT</b> is set by hardware when <b>S0ABCON.ABEN</b> is set.
<b>RESERVED</b>	15:5	r	Reserved for future use; these bits must be left at their reset values.

*Note: **SCSDET** or **SCCDET** are set when the second character has been recognized. **S0ABCON.ABEN** is reset and **ABDETIR** set after **SCSDET** or **SCCDET** have seen set.*



CONFIDENTIAL

ASC0

### 11.3.6.11 Receive FIFO Control Register

#### S0RXFCON

#### Receive FIFO Control Register

Reset value: 0100<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		RXFITL					RESERVED					RXTMEN	RXFFLU	RXFEN	

Field	Bits	Type	Description
RXFEN	0	rw	<b>Receive FIFO Enable</b> 0 Receive FIFO is disabled 1 Receive FIFO is enabled <i>Note: Resetting <b>RXFEN</b> automatically flushes the receive FIFO.</i>
RXFFLU	1	rw	<b>Receive FIFO Flush</b> 0 No operation 1 Receive FIFO is flushed <i>Note: Setting <b>RXFFLU</b> clears bit field <b>S0FSTAT.RXFFL</b>.  <b>RXFFLU</b> is always read as 0.</i>
RXTMEN	2	rw	<b>Receive FIFO Transparent Mode Enable</b> 0 Receive FIFO Transparent Mode is disabled 1 Receive FIFO Transparent Mode is enabled <i>Note: This bit is don't care if the receive FIFO is disabled (<b>RXFEN</b> = 0).</i>

**CONFIDENTIAL**

**ASC0**

Field	Bits	Type	Description
<b>RXFITL</b>	13:8	rw	<b>Receive FIFO Interrupt Trigger Level</b> Defines a receive FIFO interrupt trigger level. A receive interrupt request (RIR) is always generated after the reception of a byte when the filling level of the receive FIFO is equal to or greater <b>RXFITL</b> 000000Reserved. Do not use this combination 000001Interrupt trigger level is set to one 000010Interrupt trigger level is set to two ... 011111Interrupt trigger level is set to thirty one 100000Interrupt trigger level is set to thirty two <i>Note: In Transparent Mode this bit field is don't care.</i> <i>Note: Combinations defining an interrupt trigger level greater then the configured FIFO size should not be used</i>
<b>RESERVED</b>	7:3, 15:14	r	Reserved; these bits must be left at their reset values.

*Note: After a successful autobaud detection sequence (if implemented), the RXFIFO must be flushed before data is received.*

*Note: For smaller FIFO implementations than 64 stages also less bits in bit field **S0RXFCON.RXFITL** are implemented. The implemented width of bit field **RXFITL** depends on the size of the receive FIFO RXFIFO. Number of FIFO states =  $2^x$ ,  $x$  = number of bit in bit field **S0RXFCON.RXFITL**. Therefore, the bit field is located at [**S0RXFCON.8+x:S0RXFCON.8**].*

### 11.3.6.11.1 Transmit FIFO Control Register

#### **S0TXFCON**

#### **Transmit FIFO Control Register**

**Reset value: 0100<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
RESERVED		TXFITL						RESERVED						TX TM EN	TXF FLU	TXF EN

*Note: For smaller FIFO implementations than 64 stages also less bits in bit field **S0TXFCON.TXFITL** are implemented. The implemented width of bit field **TXFITL** depends on the size of the transmit FIFO TXFIFO. Number of FIFO states =  $2^x$ ,  $x$  = number of bit in bit field **TXFITL**. Therefore, the bit field is located at [**S0TXFCON.8+x:S0TXFCON.8**].*

**CONFIDENTIAL**

**ASC0**

Field	Bits	Type	Description
<b>TXFEN</b>	0	rw	<b>Transmit FIFO Enable</b> 0 Transmit FIFO is disabled 1 Transmit FIFO is enabled <i>Note: Resetting <b>TXFEN</b> automatically flushes the transmit FIFO.</i>
<b>TXFFLU</b>	1	rw	<b>Transmit FIFO Flush</b> 0 No operation 1 Transmit FIFO is flushed <i>Note: Setting <b>TXFFLU</b> clears bit field <b>S0FSTAT.TXFFL</b>.  <b>TXFFLU</b> is always read as 0.</i>
<b>TXTMEN</b>	2	rw	<b>Transmit FIFO Transparent Mode Enable</b> 0 Transmit FIFO Transparent Mode is disabled 1 Transmit FIFO Transparent Mode is enabled <i>Note: This bit is don't care if the receive FIFO is disabled (<b>TXFEN</b> = 0).</i>
<b>TXFITL</b>	13:8	rw	<b>Transmit FIFO Interrupt Trigger Level</b> Defines a transmit FIFO interrupt trigger level. A transmit interrupt request (TIR) is always generated after the transfer of a byte when the filling level of the transmit FIFO is equal to or lower <b>TXFITL</b> 000000Reserved. Do not use this combination 000001Interrupt trigger level is set to one 000010Interrupt trigger level is set to two ... 011111Interrupt trigger level is set to thirty one 100000Interrupt trigger level is set to thirty two <i>Note: In Transparent Mode this bit field is don't care.</i> <i>Note: Combinations defining an interrupt trigger level greater then the configured FIFO size should not be used</i>
<b>RESERVED</b>	7:3, 15:14	r	Reserved; these bits must be left at their reset values.

CONFIDENTIAL

ASC0

### 11.3.6.11.2 FIFO Status Register

**S0FSTAT**

**FIFO Status Register**

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		TXFFL						RESERVED		RXFFL					

Field	Bits	Type	Description
<b>RXFFL</b>	5:0	rh	<b>Receive FIFO Filling Level</b> 000 000 : Receive FIFO is filled with zero bytes 000 001 : Receive FIFO is filled with one byte ... 011 111 : Receive FIFO is filled with thirty one bytes 100 000 : Receive FIFO is filled with thirty two bytes <i>Note: <b>RXFFL</b> is cleared after a receive FIFO flush operation.</i>
<b>TXFFL</b>	13:8	rh	<b>Transmit FIFO Filling Level</b> 000 000 : Transmit FIFO is filled with zero bytes 000 001 : Transmit FIFO is filled with one byte ... 011 111 : Transmit FIFO is filled with thirty one bytes 100 000 : Transmit FIFO is filled with thirty two bytes <i>Note: <b>TXFFL</b> is cleared after a receive FIFO flush operation.</i>
<b>RESERVED</b>	7:6, 15:14	r	Reserved; these bits must be left at their reset values.

*Note: For smaller FIFO implementations than 64 stages also less bits in bit field **S0FSTAT.TXFFL** are implemented. The implemented width of bit field **TXFFL** depends on the size of the transmit FIFO TXFIFO. Number of FIFO states =  $2^x$ , x = number of bit in bit field **TXFFL**. Therefore, the bit field is located at **[S0FSTAT.8+x:S0FSTAT.8]**.*

*Note: For smaller FIFO implementations than 64 stages also less bits in bit field **S0FSTAT.RXFFL** are implemented. The implemented width of bit field **RXFFL** depends on the size of the transmit FIFO RXFIFO. Number of FIFO states =  $2^x$ , x = number of bit in bit field **RXFFL**. Therefore, the bit field is located at **[S0FSTAT.x:S0FSTAT.0]**.*

**CONFIDENTIAL**

**ASCO**

### 11.3.6.11.3 Flowcontrol Control Register

#### **S0FCCON**

#### **Flowcontrol Control Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		RTSTL						RESERVED			RTS	RESERVED	CTSEN	RTSEN	

Field	Bits	Type	Description
<b>RTSEN</b>	0	rw	<b>RTS Enable</b> 0: Disable RTS HW Flowcontrol 1: Enable RTS HW Flowcontrol
<b>CTSEN</b>	1	rw	<b>CTS Enable</b> 0: Disable CTS HW Flowcontrol 1: Enable CTS HW Flowcontrol
<b>RTS</b>	4	rw	<b>Request To Send Control Bit</b> 0: RTS is inactive (RTS_N = '1') 1: RTS is active (RTS_N = '0') The RTS_N pin is controlled by this bit only if hardware flow control is disabled ( <b>RTSEN</b> = 0).
<b>RTSTL</b>	13:8	rw	RTS Receive FIFO Trigger Level 000 000: Receive FIFO is filled with zero byte. 000 001: Receive FIFO is filled with one byte. ... 011 111: Receive FIFO is filled with thirty one bytes. 100 000: Receive FIFO is filled with thirty two bytes. <i>Note: <b>RTSTL</b> is cleared after a receive FIFO flush operation.</i> <i>Note: Combinations defining a RTS trigger level greater than the configured FIFO size should not be used.</i>
<b>RESERVED</b>	3:2, 7:5, 15:14	r	Reserved; these bits must be left at their reset values.

*Note: For smaller FIFO implementations than 64 stages also less bits in bit field **S0FCCON.RTSTL** are implemented. The implemented width of bit field **RTSTL** depends on the size of the receive FIFO RXFIFO. Number of FIFO states =  $2^x$ ,  $x$  = number of bit in bit field **RTSTL**. Therefor the bit field is located at [**S0FCCON.8+x:S0FCCON.8**].*

CONFIDENTIAL

ASC0

### 11.3.6.11.4 Flowcontrol Status Register

**S0FCSTAT**

**Flowcontrol Status Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															CTS

Field	Bits	Typ	Description
CTS	0	rh	<b>CTS Status</b> 0 CTS inactive (CTS_N = 1) 1 CTS active (CTS_N = 0)
RESERVED	15:1	r	Reserved; these bits must be left at their reset values.

*Note: The reset value depends on the input CTS\_N.*

### 11.3.6.11.5 RX Timeout Register

The timeout control register contains the 16-bit reload value for the timeout detection timer.

**S0TMO**

**RX Timeout Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TMO															

Field	Bits	Typ	Description
TMO	15:0	rw	<b>Timeout Detection Timer Reload Value</b> 0: timeout detection disabled 1 <sub>H</sub> ..FFFF <sub>H</sub> : Timeout after <TMO> shift clock cycles

**CONFIDENTIAL**

**ASC1**

## 11.4 ASC1

History	
Current version: Rev. 1.06, 2005-11-04	
Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.01	
<b>Page 1067</b>	Updated <b>System Integration</b> : WS00006682
Changes for Rev. 1.06	

### System Integration:

- Supply domain: VDD\_MAIN
- Chip internal interfaces:
  - Clock domain: Refer to **Section 10.4.1.3 Sub-System Clocks and Enables** and see **Figure 10-10 Clock Enable (on Page 662)**.
  - Bus Domain: PD-Bus
- Interrupt sources:
- Monitor Pins: Refer to **Section 11.8.10 Internal Signal Monitoring (on Page 1209)**

### 11.4.1 Features of the ASC1 Controller

- Duplex asynchronous operating modes:
  - 8- or 9-bit data frames, LSB first
  - Parity bit generation/checking
  - One or two stop bits
  - Baudrate from 3.27 MBaud to 0.77 Baud (with a 52MHz module clock  $f_{hw\_clk}$ , for information about  $hw\_clk$ , refer to [Table 11-21 ASC1 Registers Summary \(on Page 1070\)](#))
- Multiprocessor Mode for automatic address/data byte detection
- Loopback capability
- Half-duplex 8-bit synchronous operating mode:
  - Baudrate from 6.5 MBaud to 662.5 Baud (with a 52 MHz module clock  $f_{hw\_clk}$ )
- Double buffered transmitter/receiver
- Interrupt generation:
  - On a transmitter buffer empty condition
  - On a transmit last bit of a frame condition
  - On a receiver buffer full condition
  - On an error condition (frame, parity, overrun error)
- RX timeout

#### 11.4.1.1 Differences between ASC1 and ASC0

The ASC1 does not have the following ASC0 functions:

- Autobaud detection
- IRDA data transmit
- HW Flow Control

There are transmit and receive FIFOs:

- 8-stage receive FIFO (RXFIFO)
- 8-stage transmit FIFO (TXFIFO)
- Independent control of RXFIFO and TXFIFO
- 9-Bit FIFO data width
- Programmable Receive/Transmit Interrupt Trigger Level
- Receive and transmit FIFO filling level indication
- Overrun error generation
- Underflow error generation.

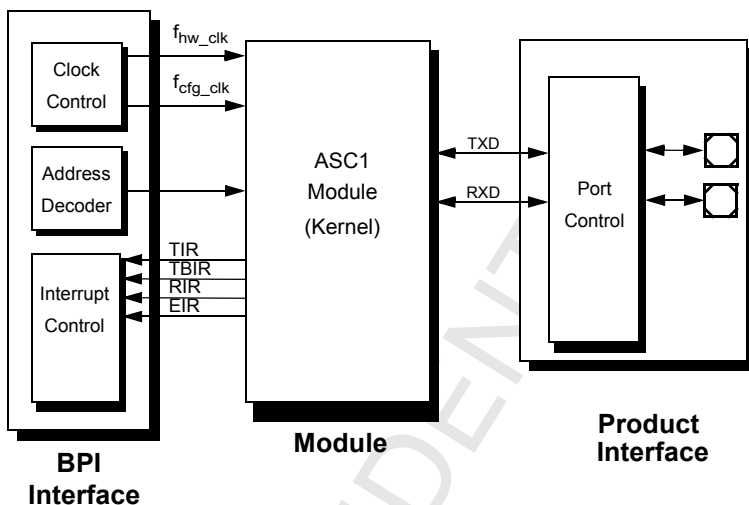


## 11.4.2 Functional Description of the ASC1

**Note:** The ASC1 is connected to the RXD1 and TXD1 pins.

**Figure 11-32** shows all functional relevant interfaces associated with the ASC1 Kernel.

**Figure 11-32 ASC1 Interface Diagram<sup>1)</sup>**



<sup>1)</sup> For information about hw\_clk, refer to [Table 11-21 ASC1 Registers Summary \(on Page 1070\)](#).

CONFIDENTIAL

ASC1

### 11.4.3 Registers

For the ASC1 register addresses refer to [Section 12.2 PD-Bus Register Addresses \(on Page 1273\)](#).

**Table 11-21 ASC1 Registers Summary**

Name	Clock	Access Condition	Description
<b>S1PERID</b>	hw_clk <sup>1)</sup>	non bit addressable	Peripheral ID Number Register
<b>S1PISEL</b>	hw_clk <sup>1)</sup>	bit addressable	Peripheral Input Select Register
<b>S1CON</b>	hw_clk <sup>1)</sup>	bit addressable	Control Register
<b>S1BG</b>	hw_clk <sup>1)</sup>	non bit addressable	Baudrate Timer Reload Register
<b>S1FDV</b>	hw_clk <sup>1)</sup>	non bit addressable	Fractional Divider Register
<b>S1TBUF</b>	hw_clk <sup>1)</sup>	non bit addressable	Transmit Buffer Register
<b>S1RBUF</b>	hw_clk <sup>1)</sup>	non bit addressable	Receive Buffer Register
<b>S1RXFCON</b>	hw_clk <sup>1)</sup>	non bit addressable	Receive FIFO Control Register
<b>S1TXFCON</b>	hw_clk <sup>1)</sup>	non bit addressable	Transmit FIFO Control Register
<b>S1FSTAT</b>	hw_clk <sup>1)</sup>	non bit addressable	FIFO Status Register

<sup>1)</sup> Refer to [Chip internal interfaces](#): Clock domain on [Page 1067](#).

#### 11.4.3.1 ASC1 Identification Register

The **S1PERID** register stores the Module Number and Revision Number for ASC0.

##### S1PERID

##### Peripheral Identification Register

Reset values: **4423<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOD_NUMBER								MOD_REV							

Field	Bits	Type	Description
<b>MOD_REV</b>	7:0	r	<b>ASC1 Revision Number</b>
<b>MOD_NUMBER</b>	15:8	r	<b>ASC1 Number</b>

**CONFIDENTIAL**

**ASC1**

### 11.4.3.2 Port Input Select Register

The **S1PISEL** register selects the source to receiver data (RXD1 pin) of the ASC0.

**S1PISEL.RIS** must remain at the reset default value of 0 to select the RXD1 pin. This ensures correct operation in Synchronous and Asynchronous modes.

#### **S1PISEL**

#### **Port Input Select Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED															RIS

Field	Bits	Type	Description
<b>RIS</b>	0	rw	<b>Receiver Input Select</b> 0 RXD1 pin selected (default) 1 Tied to 0, no input received
<b>RESERVED</b>	15:1	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**ASC1**

### 11.4.3.3 Control Register

The operating mode of the serial channel ASC0 is controlled by its control register. This register contains control bits for mode and error check selection, and status flags for error identification.

#### S1CON

#### Control Register

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LB	BRS	ODD	FDE	OE	FE	PE	OEN	FEN	PEN	REN	STP		M	

Field	Bits	Type	Description
<b>M</b>	2:0	rw	<b>Mode Control</b> 000 8-bit-data for synchronous operation 001 8-bit-data for asynchronous operation 010 8-bit-data IrDA Mode for asynchronous operation 011 7-bit-data and parity for asynchronous operation 100 9-bit-data for asynchronous operation 101 8-bit-data and wake up bit for asynchronous operation 110 Reserved. Do not use this combination 111 8-bit-data and parity for asynchronous operation
<b>STP</b>	3	rw	<b>Number of Stop Bits Selection</b> 0 One stop bit 1 Two stop bits
<b>REN</b>	4	rwh	<b>Receiver Enable Bit</b> 0 Receiver disabled 1 Receiver enabled <i>Note: Bit is cleared by hardware after reception of a byte in the Synchronous Mode</i>
<b>PEN</b>	5	rw	<b>Parity Check Enable</b> All Asynchronous Modes without IrDA Mode: 0 Ignore parity 1 Check parity

**CONFIDENTIAL**

**ASC1**

Field	Bits	Type	Description
<b>FEN</b>	6	rw	<b>Framing Check Enable</b> (Asynchronous Mode only) 0 Ignore framing errors 1 Check framing errors
<b>OEN</b>	7	rw	<b>Overrun Check Enable</b> 0 Ignore overrun errors 1 Check overrun errors
<b>PE</b>	8	rwh	<b>Parity Error Flag</b> Set by hardware on a parity error ( <b>PEN</b> = 1. Must be cleared by software.
<b>FE</b>	9	rwh	<b>Framing Error Flag</b> Set by hardware on a framing error ( <b>FEN</b> = 1). Must be cleared by software.
<b>OE</b>	10	rwh	<b>Overrun Error Flag</b> Set by hardware on an overrun/underflow error ( <b>OEN</b> = 1). Must be cleared by software.
<b>FDE</b>	11	rw	<b>Fractional Divider Enable</b> 0 Fractional divider disabled 1 Fractional divider enabled and used as perscaler for baudrate generator (bit BRS is 'don't care')
<b>ODD</b>	12	rw	<b>Parity Selection</b> 0 Even parity selected (parity bit of 1 is included in data stream on odd number of 1 and parity bit of 0 is included in data stream on even number of 1) 1 Odd parity selected (parity bit of 1 is included in data stream on even number of 1 and parity bit of 0 is included in data stream on odd number of 1)
<b>BRS</b>	13	rw	<b>Baudrate Selection</b> 0 Baud rate timer prescaler divide-by-2 selected 1 Baud rate timer prescaler divide-by-3 selected <i>Note: <b>BRS</b> is 'don't care' if <b>FDE</b> = 1 (fractional divider selected)</i>

CONFIDENTIAL

ASC1

Field	Bits	Type	Description
<b>LB</b>	14	rw	<b>Loopback Mode Enabled</b> 0 Loopback Mode disabled. Standard transmit/receive Mode 1 Loopback Mode enabled
<b>R</b>	15	rw	<b>Baudrate Generator Run Control Bit</b> 0 Baudrate generator disabled (ASC0 inactive) 1 Baudrate generator enabled <i>Note: <b>S1BG.BR_VALUE</b> should only be written if <b>R = 0</b>.</i>

#### 11.4.3.4 Baudrate Register

The ASC0 baudrate timer reload register contains the 13-bit reload value for the baudrate timer in Asynchronous and Synchronous Mode.

**S1BG**

**Baudrate Timer/Reload Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			BR_VALUE												

Field	Bits	Type	Description
<b>BR_VALUE</b>	12:0	rw	<b>Baudrate Timer/Reload Value</b> Reading returns the 13-bit content of the baudrate timer; writing loads the baudrate timer/reload value. <i>Note: <b>BR_VALUE</b> should only be written if <b>S1CON.R = 0</b>.</i>
<b>RESERVED</b>	15:13	r	Reserved; these bits must be left at their reset values.

CONFIDENTIAL

ASC1

### 11.4.3.5 Fractional Divider Register

The ASC0 fractional divider register contains the 9-bit divider value for the fractional divider (Asynchronous Mode only). It is also used for reference clock generation of the autobaud detection unit.

#### S1FDV

#### Fractional Divider Register

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED									FD_VALUE						

Field	Bits	Type	Description
FD_VALUE	8:0	rw	<b>Fractional Divider Register Value</b> <b>FD_VALUE</b> contains the 9-bit value of the fractional divider which defines the fractional divider ratio $n/512$ ( $n = 0-511$ ). With $n = 0$ , the fractional divider is switched off (input = output frequency, $f_{DIV} = f_{hw\_clk}$ , see <a href="#">Figure 11-27 ASC0 Baudrate Generator Circuitry in Synchronous Mode (on Page 1036)</a> ).
RESERVED	15:9	r	Reserved; these bits must be left at their reset values.

CONFIDENTIAL

ASC1

### 11.4.3.6 Transmitter Buffer Register

The ASC0 transmitter buffer register contains the transmit data value in Asynchronous and Synchronous Mode.

#### S1TBUF

#### Transmit Buffer Register

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								TD_VALUE							

Field	Bits	Type	Description
TD_VALUE	8:0	rw	<b>Transmit Data Register Value</b> TD_VALUE contains the data to be transmitted in asynchronous and synchronous operating mode of the ASC0. Data transmission is double buffered, Therefore, a new value can be written to TD_VALUE before the transmission of the previous value is complete.
RESERVED	15:9	r	Reserved; these bits must be left at their reset values.



CONFIDENTIAL

ASC1

### 11.4.3.7 Receiver Buffer Register

The ASC0 Receiver buffer register contains the transmit data value in Asynchronous and Synchronous Modes.

#### S1RBUF

#### Receive Buffer Register

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								RD_VALUE							

Field	Bits	Type	Description
RD_VALUE	8:0	rw	<b>Receive Data Register Value</b> <b>RD_VALUE</b> contains the received data bits and, depending on the selected mode, the parity bit in asynchronous and synchronous operating mode of the ASC0. In asynchronous operating mode if <b>S1CON.M</b> = 011 (7-bit data + parity), the received parity bit is written into <b>RD_VALUE[7]</b> . In asynchronous operating mode if <b>S1CON.M</b> = 111 (8-bit data + parity), the received parity bit is written into <b>RD_VALUE[8]</b> .
RESERVED	15:9	r	Reserved; these bits must be left at their reset values.

CONFIDENTIAL

ASC1

### 11.4.3.8 Receive FIFO Control Register

#### S1RXFCON

#### Receive FIFO Control Register

Reset value: 0100<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
RESERVED		RXFITL						RESERVED						RXTMEN	RXFFLU	RXFEN

Field	Bits	Type	Description
RXFEN	0	rw	<b>Receive FIFO Enable</b> 0 Receive FIFO is disabled 1 Receive FIFO is enabled <i>Note: Resetting <b>RXFEN</b> automatically flushes the receive FIFO.</i>
RXFFLU	1	rw	<b>Receive FIFO Flush</b> 0 No operation 1 Receive FIFO is flushed <i>Note: Setting <b>RXFFLU</b> clears bit field <b>S1FSTAT.RXFFL</b>.  <b>RXFFLU</b> is always read as 0.</i>
RXTMEN	2	rw	<b>Receive FIFO Transparent Mode Enable</b> 0 Receive FIFO Transparent Mode is disabled 1 Receive FIFO Transparent Mode is enabled <i>Note: This bit is don't care if the receive FIFO is disabled (<b>RXFEN</b> = 0).</i>

**CONFIDENTIAL**

**ASC1**

Field	Bits	Type	Description
<b>RXFITL</b>	13:8	rw	<p><b>Receive FIFO Interrupt Trigger Level</b></p> <p>Defines a receive FIFO interrupt trigger level. A receive interrupt request (RIR) is always generated after the reception of a byte when the filling level of the receive FIFO is equal to or greater <b>RXFITL</b></p> <p>000000Reserved. Do not use this combination</p> <p>000001Interrupt trigger level is set to one</p> <p>000010Interrupt trigger level is set to two</p> <p>...</p> <p>011111Interrupt trigger level is set to thirty one</p> <p>100000Interrupt trigger level is set to thirty two</p> <p><i>Note: In Transparent Mode this bit field is don't care.</i></p> <p><i>Note: Combinations defining an interrupt trigger level greater then the configured FIFO size should not be used</i></p>
<b>RESERVED</b>	7:3, 15:14	r	Reserved; these bits must be left at their reset values.

*Note: After a successful autobaud detection sequence (if implemented), the RXFIFO must be flushed before data is received.*

*Note: For smaller FIFO implementations than 64 stages also less bits in bit field **S1RXFCN.RXFITL** are implemented. The implemented width of bit field **RXFITL** depends on the size of the receive FIFO RXFIFO. Number of FIFO states =  $2^x$ ,  $x$  = number of bit in bit field **S1RXFCN.RXFITL**. Therefore, the bit field is located at [**S1RXFCN.8+x:S1RXFCN.8**].*

CONFIDENTIAL

ASC1

### 11.4.3.8.1 Transmit FIFO Control Register

#### S1TXFCON

#### Transmit FIFO Control Register

Reset value: 0100<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED		TXFITL						RESERVED					TX TM EN	TXF FLU	TXF EN

Field	Bits	Type	Description
TXFEN	0	rw	<b>Transmit FIFO Enable</b> 0 Transmit FIFO is disabled 1 Transmit FIFO is enabled <i>Note: Resetting <b>TXFEN</b> automatically flushes the transmit FIFO.</i>
TXFFLU	1	rw	<b>Transmit FIFO Flush</b> 0 No operation 1 Transmit FIFO is flushed <i>Note: Setting <b>TXFFLU</b> clears bit field <b>S1FSTAT.TXFFL</b>.  <b>TXFFLU</b> is always read as 0.</i>
TXTMEN	2	rw	<b>Transmit FIFO Transparent Mode Enable</b> 0 Transmit FIFO Transparent Mode is disabled 1 Transmit FIFO Transparent Mode is enabled <i>Note: This bit is don't care if the receive FIFO is disabled (<b>TXFEN</b> = 0).</i>

**CONFIDENTIAL**

**ASC1**

Field	Bits	Type	Description
<b>TXFITL</b>	13:8	rw	<p><b>Transmit FIFO Interrupt Trigger Level</b></p> <p>Defines a transmit FIFO interrupt trigger level. A transmit interrupt request (TIR) is always generated after the transfer of a byte when the filling level of the transmit FIFO is equal to or lower <b>TXFITL</b></p> <p>000000Reserved. Do not use this combination</p> <p>000001Interrupt trigger level is set to one</p> <p>000010Interrupt trigger level is set to two</p> <p>...</p> <p>011111Interrupt trigger level is set to thirty one</p> <p>100000Interrupt trigger level is set to thirty two</p> <p><i>Note: In Transparent Mode this bit field is don't care.</i></p> <p><i>Note: Combinations defining an interrupt trigger level greater then the configured FIFO size should not be used</i></p>
<b>RESERVED</b>	7:3, 15:14	r	Reserved; these bits must be left at their reset values.

*Note: For smaller FIFO implementations than 64 stages also less bits in bit field **S1TXFCON.TXFITL** are implemented. The implemented width of bit field **TXFITL** depends on the size of the transmit FIFO TXFIFO. Number of FIFO states =  $2^x$ ,  $x$  = number of bit in bit field **TXFITL**. Therefore, the bit field is located at [**S1TXFCON**.8+x:**S1TXFCON**.8].*

**CONFIDENTIAL**

**ASC1**

### 11.4.3.8.2 FIFO Status Register

**S1FSTAT**

**FIFO Status Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>		<b>TXFFL</b>						<b>RESERVED</b>		<b>RXFFL</b>					

Field	Bits	Type	Description
<b>RXFFL</b>	5:0	rh	<b>Receive FIFO Filling Level</b> 000000Receive FIFO is filled with zero bytes 000001Receive FIFO is filled with one byte ... 011111Receive FIFO is filled with thirty one bytes 100000Receive FIFO is filled with thirty two bytes <i>Note: <b>RXFFL</b> is cleared after a receive FIFO flush operation.</i>
<b>TXFFL</b>	13:8	rh	<b>Transmit FIFO Filling Level</b> 000000Transmit FIFO is filled with zero bytes 000001Transmit FIFO is filled with one byte ... 011111Transmit FIFO is filled with thirty one bytes 100000Transmit FIFO is filled with thirty two bytes <i>Note: <b>TXFFL</b> is cleared after a receive FIFO flush operation.</i>
<b>RESERVED</b>	7:6, 15:14	r	Reserved; these bits must be left at their reset values.

*Note: For smaller FIFO implementations than 64 stages also less bits in bit field **S1FSTAT.TXFFL** are implemented. The implemented width of bit field **TXFFL** depends on the size of the transmit FIFO TXFIFO. Number of FIFO states =  $2^x$ , x = number of bit in bit field **TXFFL**. Therefore, the bit field is located at [**S1FSTAT**.8+x:**S1FSTAT**.8].*

*Note: For smaller FIFO implementations than 64 stages also less bits in bit field **S1FSTAT.RXFFL** are implemented. The implemented width of bit field **RXFFL** depends on the size of the transmit FIFO RXFIFO. Number of FIFO states =  $2^x$ , x = number of bit in bit field **RXFFL**. Therefore, the bit field is located at [**S1FSTAT**.x:**S1FSTAT**.0].*

**CONFIDENTIAL**

**CAPCOM 1 and 2**

## 11.5 CAPCOM 1 and 2

History	
Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.01	
<b>Page 1083</b>	Updated <b>System Integration</b> : WS00006682
Changes for Rev. 1.04	
<b>Page 1112</b>	Updated <b>Table 11-27 CAPCOM 2 Input Signal Selection</b>
Changes for Rev. 1.06	

### System Integration:

- Supply domain: VDD\_MAIN
- Chip internal interfaces:
  - Clock domain: Refer to **Section 10.4.1.3 Sub-System Clocks and Enables (on Page 661)** and see **Figure 10-10 Clock Enable (on Page 662)**.
  - Bus domain: PD-Bus
- Interrupt sources:
  - Monitor Pins: refer to **Section 11.8.10 Internal Signal Monitoring (on Page 1209)**

**CONFIDENTIAL****CAPCOM 1 and 2****11.5.1 Introduction**

The E-GOLDradio provides two identical 8-channel CAPCOM units each of which interacts with 2 timers. A CAPCOM unit can either

- Capture the contents of a timer on specific internal or external events
- Compare a timer content with given values and modify output signals if there is a match.

With this mechanism it supports generation and control of timing sequences on up to 8 channels with a minimum of software intervention.

*Note: In the figures the x labels are used for timers and the y and z labels for channels.*

**Features for Each CAPCOM Unit**

- Module of timers registers and comparators for high speed pulse and waveform generation or time measurement.
- Two 16-bit timers with reload registers.
- 8 registers individually configurable for capture or compare function.
- 10 interrupts: 8 capture compare interrupts and two timer interrupts.
- Up to 8 software timers.
- Programmable clock with multiple sources.
- 154 ns maximum resolution @ 52 MHz master clock (hw\_clk) if staggered mode enabled.
- 19.2 ns maximum resolution @ 52 MHz master clock (hw\_clk) if staggered mode disabled.
- Double register compare function.
- Primary clock prescaler.
- Additional output register.
- Single event mode.

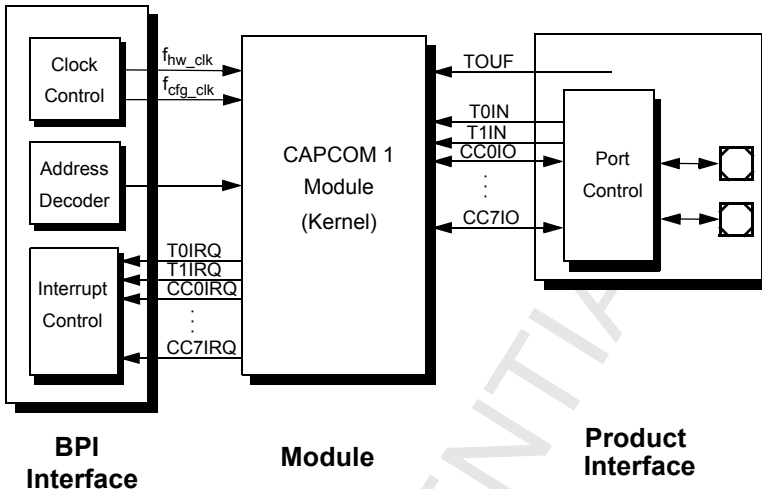
**11.5.2 Operational Overview**

From the programmer's point of view, the term 'CAPCOM Unit' refers to a set of registers which are associated with this peripheral, including the port pins which may be used for alternate input/output functions including their direction control bits (refer to [Table 11-24 CAPCOM Register Summary \(on Page 1100\)](#)).

[Figure 11-33](#) shows the interface diagram of CAPCOM Unit 1. CAPCOM Unit 2 is identical except that the timers are **T7** and **T8** (instead of **T0** and **T1**), and the channels are numbered from 16 to 23 (instead of 0 and 7).



Figure 11-33 CAPCOM 1 Interface Diagram



*Note: If required, it is also possible to connect CCxIO ( $x = 0$  to 7 or 16 to 23) channels directly with the ports, without going through a port control module.*

### 11.5.3 Functional Overview

A CAPCOM unit is typically used to handle high speed IO tasks such as pulse and waveform generation, pulse width modulation, or recording of the time when a specific event occurs. It also allows the implementation of up to eight software timers. The highest resolution of a CAPCOM Unit is 154 ns @ 52 MHz master clock ( $hw\_clk$ ) if staggered mode is enabled or 19.2 ns maximum resolution @ 52 MHz master clock ( $hw\_clk$ ) if staggered mode is disabled.

Each CAPCOM unit consists of two 16-bit timers, each with its own reload register and dual purpose 16-bit capture/compare registers.

The input clock for the CAPCOM timers is programmable to several prescaled values of the master clock ( $hw\_clk$ ), or it can be derived from an overflow/underflow ( $TOUF$ ) of an external timer. **T0** and **T7** may also operate in counter mode (from an external input) where they can be clocked by external events ( $TxIN$ ).

*Note: The external timer is usually the Timer 6 of the General Purpose Timer unit.*

Each capture/compare register may be programmed individually for capture or compare function, and each register may be allocated to either timer of the associated unit. Each capture/compare register has one signal associated with it which serves as an input signal for the capture function or as an output signal for the compare function. The capture function causes the current timer contents to be latched into the respective

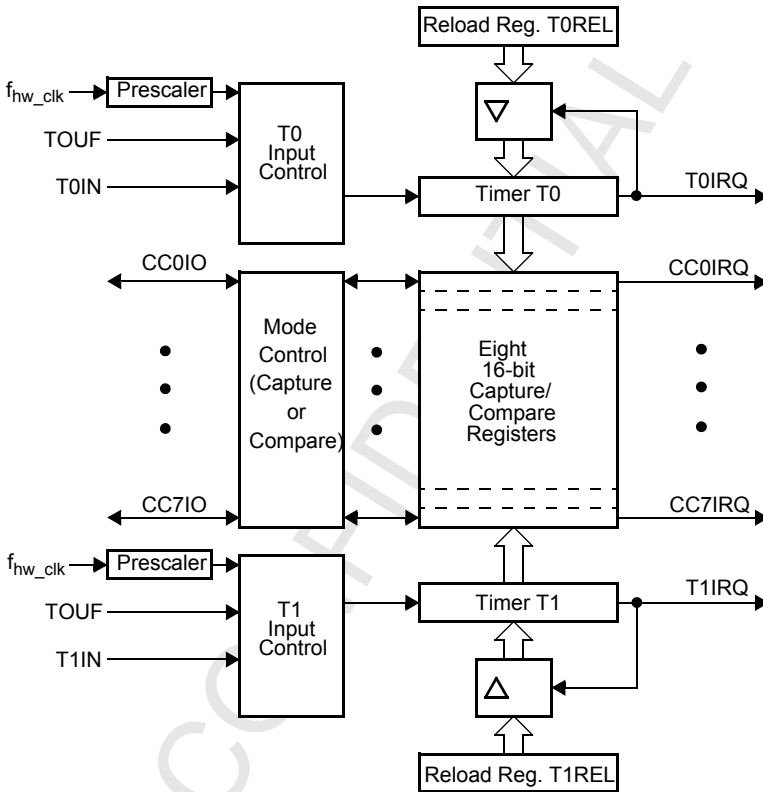
CONFIDENTIAL

CAPCOM 1 and 2

capture/compare register triggered by an event (transition) on its associated signal. The compare function may cause an output signal transition on that signal whose associated capture/compare register matches the current timer contents. Specific interrupt requests are generated upon each capture/compare event or upon timer overflow.

Figure 11-34 shows the basic structure of CAPCOM unit 1 (CAPCOM unit 1 is identical).

Figure 11-34 CAPCOM Unit 1 Block Diagram

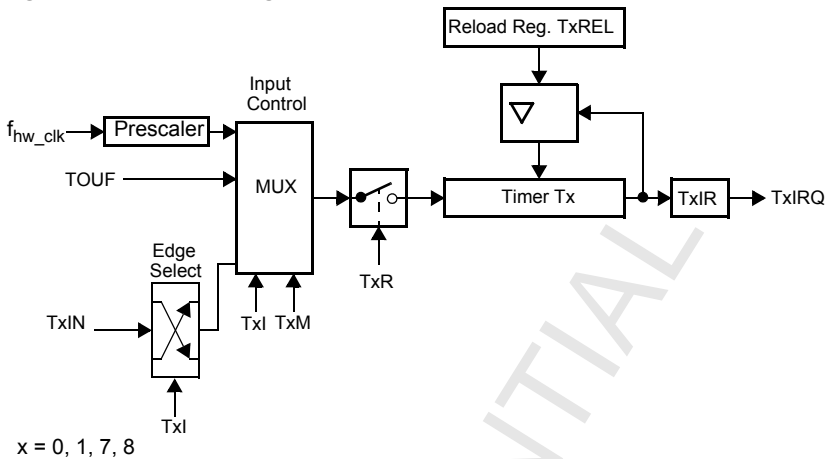


### 11.5.3.1 The Timers

The primary use of the timers **T0** and **T1** (**T7** and **T8**) is to provide two independent time bases with 154 ns maximum resolution @ 52 MHz master clock (**hw\_clk**) and staggered mode enabled or 19.2 ns maximum resolution @ 52 MHz master clock (**hw\_clk**) and staggered mode disabled for the capture/compare registers of a unit, but they may also be used independent of the capture/compare registers.

The basic structure of the two timers is identical (see Figure 11-35).

Figure 11-35 Block Diagram of Timers



The functions of the CAPCOM timers are controlled via the bit addressable control registers **T01CON** (for CAPCOM 1) and **T78CON** (for CAPCOM 2). The high-byte of **TxxCON** controls **T1** or **T7**, the low-byte of **TxxCON** controls **T0** or **T8**. The control options are identical for all timers.

The timer run flags **TxxCON.T0R** and **TxxCON.T1R** allow the starting and stopping of the timers. The following description of the timer modes and operation always applies to the enabled state of the timers, that is, the respective run flag is assumed to be set.

In all modes, the timers always count upward. The current timer values are accessible from the MCU in the timer registers Tx, which are non bit addressable registers. When the MCU writes to a register Tx in the state immediately before the respective timer increment or reload is to be performed, the MCU write operation has priority and the increment or reload is disabled to guarantee correct timer operation.

### Timer Mode

The bits **TxxCON.TxM** select between the timer or counter mode for the respective timer. In timer mode (**TxxCON.TxM** set), the input clock for a timer is derived from the internal hw\_clk clock divided by a programmable prescaler. The different options for the prescaler are selected separately for each timer by the bit field **TxxCON.Txl**.

The input frequencies  $f_{Tx}$  for Tx are determined as a function of the hw\_clk clock as follows, where  $\langle TxI \rangle$  represents the contents of the bit field **TxxCON.Txl**:

$$f_{Tx} = \frac{f_{hw\_clk}}{2^{(\langle TxI \rangle + 3)}}$$

Staggered Mode enabled

$$f_{Tx} = \frac{f_{hw\_clk}}{2^{\langle TxI \rangle}}$$

Staggered Mode disabled

**CONFIDENTIAL**

**CAPCOM 1 and 2**

When a timer overflows from  $FFFF_H$  to  $0000_H$  it is reloaded with the value stored in its respective reload register  $TxREL$ . The reload value determines the period  $P_{Tx}$  between two consecutive overflows of  $Tx$  as follows:

$$P_{Tx} = \frac{(2^{16} - <TxREL>) * 2^{(<TxI>+3)}}{f_{hw\_clk}}$$

Staggered Mode enabled

$$P_{Tx} = \frac{(2^{16} - <TxREL>) * 2^{<TxI>}}{f_{hw\_clk}}$$

Staggered Mode disabled

Examples for timer input frequencies, resolution, and periods which result from the selected prescaler option in **TxI** when using a 52 MHz  $hw\_clk$  clock are listed in [Table 11-22](#). The numbers for the timer periods are based on a reload value of  $0000_H$ .

*Note: Some numbers may be rounded.*

**Table 11-22 Timing Examples**

<b><math>f_{hw\_clk} = 52\text{ MHz}</math> Staggered Mode enabled</b>	<b>Timer Input Selection TxI</b>							
	000 <sub>B</sub>	001 <sub>B</sub>	010 <sub>B</sub>	011 <sub>B</sub>	100 <sub>B</sub>	101 <sub>B</sub>	110 <sub>B</sub>	111 <sub>B</sub>
<b>Prescaler for <math>f_{hw\_clk}</math></b>	8	16	32	64	128	256	512	1024
<b>Input Frequency</b>	6.5 MHz	3.25 MHz	1.625 MHz	812.5 kHz	406.25 kHz	203.125 kHz	101.562 kHz	50.781 kHz
<b>Resolution</b>	154 ns	308 ns	615 ns	1.23 $\mu$ s	2.46 $\mu$ s	4.92 $\mu$ s	9.85 $\mu$ s	19.7 $\mu$ s
<b>Period</b>	10.08 ms	20.164 ms	40.3 ms	80.6 ms	161.3 ms	322.6 ms	645.27 ms	1.29 s
<b><math>f_{hw\_clk} = 52\text{ MHz}</math> Staggered Mode disabled</b>	<b>Timer Input Selection TxI</b>							
	000 <sub>B</sub>	001 <sub>B</sub>	010 <sub>B</sub>	011 <sub>B</sub>	100 <sub>B</sub>	101 <sub>B</sub>	110 <sub>B</sub>	111 <sub>B</sub>
<b>Prescaler for <math>f_{hw\_clk}</math></b>	1	2	4	8	16	32	64	128
<b>Input Frequency</b>	52 MHz	26 MHz	13 MHz	6.50 MHz	3.25 MHz	1.625 MHz	0.8125 MHz	0.406 MHz
<b>Resolution</b>	19.2 ns	38.5 ns	76.9 ns	154 ns	308 ns	615 ns	1.23 $\mu$ s	2.46 $\mu$ s
<b>Period</b>	1.26 ms	2.52 ms	5.04 ms	10.08 ms	20.16 ms	40.32 ms	80.65 ms	161.3 ms

After a timer has been started by setting its run flag (**TxxCON.TxR**), the first increment occurs within the time interval which is defined by the selected timer resolution. All further increments occur exactly after the time defined by the timer resolution.

## CONFIDENTIAL

## CAPCOM 1 and 2

When both timers of a CAPCOM unit are to be incremented or reloaded at the same time **T0** or **T7** is always serviced one  $hw\_clk$  clock before **T1** or **T8**.

### Counter Mode

The bits **TxxCON.TxM** select between timer or counter mode for the respective timer. In Counter mode (**TxM** = 1) the input clock for a timer can be derived from the overflows/underflows of timer\_in.

In addition, the timers can be clocked by external events if a signal is connected to that module input. Either a positive, a negative, or both a positive and a negative transition can be selected to cause an increment of **T0** or **T7**.

*Note: When **T1** or **T8** is programmed to run in counter mode, bit field **TxxCON.T[1,8]I** is used to enable the overflows/underflows of the GPT Timer T6 as the count source. This is the only option for **T1** or **T8**, and it is selected by the combination **T[1,8]I** =  $X00_B$ . When bit field **T[1,8]I** is programmed to any other combination, the respective timer (**T1** or **T8**) stops.*

When **T0** or **T7** is programmed to run in counter mode, bit field **TxxCON.T[0,7]I** is used to select the count source and transition (if the source is the input pin) which should cause a count trigger (refer to **TxxCON** for the possible selections).

*Note: To use pin T0IN as external count input pin, the respective port pin must be configured as input, that is, the corresponding direction control bit (DP3.0) must be cleared (0).*

*If the respective port pin is configured as output, the associated timer may be clocked by modifying the port output latch P3.0 via software, for example, for testing purposes.*

The maximum external input frequency to **T0** in counter mode is  $f_{hw\_clk}/16$  (3.25 MHz @ 52 MHz  $f_{hw\_clk}$ ). To ensure that a signal transition is properly recognized at the timer input, an external count input signal should be held for at least eight  $hw\_clk$  clock cycles before it changes its level again. The incremented count value appears in **T0** within eight  $hw\_clk$  clock cycles after the signal transition at T0IN.

### Reload

A reload of a timer with the 16-bit value stored in its associated reload register in both modes is performed each time a timer would overflow from  $FFFF_H$  to  $0000_H$ . In this case the timer does not wrap around to  $0000_H$ , but rather is reloaded with the contents of the respective reload register TxREL. The timer then resumes incrementing starting from the reloaded value.

The reload registers TxREL are not bit addressable.

CONFIDENTIAL

CAPCOM 1 and 2

### 11.5.3.2 Timer Interrupt

Upon a timer overflow the corresponding timer interrupt request flag TxIR for the respective timer will be set. This flag can be used to generate an interrupt or trigger a interrupt controller service request, when enabled by the respective interrupt enable bit TxIE.

Each timer has its own bit addressable interrupt control register (TxIC) and its own interrupt vector (TxIRQ). The organization of the interrupt control registers TxIC is identical with the other interrupt control registers (refer to [Section 6.3.3 Interrupt and Exception Execution \(on Page 111\)](#)).

### 11.5.3.3 Capture/Compare Registers

The 16-bit capture/compare registers **CCy** are used as data registers for capture or compare operations with respect to two timers in each CAPCOM unit. The capture/compare registers are not bit addressable.

Each of the registers **CCy** may be individually programmed for capture mode or one of 4 different compare modes, and may be allocated individually to one of the two timers of the respective CAPCOM unit (**CC0** through **CC7** for CAPCOM 1; **CC16** through **CC23** for CAPCOM 2). A special combination of compare modes additionally allows the implementation of a 'double-register' compare mode. When capture or compare operation is disabled for one of the **CCy** registers, it may be used for general purpose variable storage.

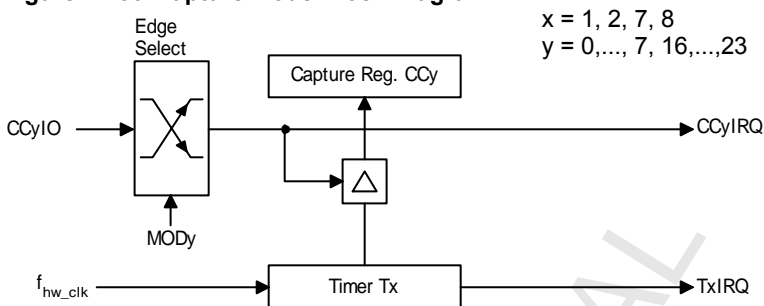
The functions of the capture/compare registers are controlled by four bit-addressable mode control registers **CCMz** (**CCM0** and **CCM1** for CAPCOM 1; **CCM4** and **CCM5** for CAPCOM 2). Each register contains bits for mode selection and timer allocation of four capture/compare registers.

### 11.5.3.4 Capture Mode

In response to an external event the content of the associated timer (**T0**, depending on the state of the allocation control bit **CCMz.ACCy**) is latched into the respective capture register **CCy**. The external event causing a capture can be programmed to be either a positive, a negative, or both a positive or a negative transition at the respective external input pin CCyIO.

The triggering transition is selected by the mode bits **CCMz.MODY** in the respective mode control register. In any case, the event causing a capture will also set the respective interrupt request flag CCyIR, which can cause an interrupt or an interrupt controller service request when enabled.

Figure 11-36 Capture Mode Block Diagram



*Note: In order to use the respective port pin as external capture input pin CCyIO for capture register CCy, this port pin must be configured as input, that is, the corresponding direction control bit must be cleared.*

To ensure that a signal transition is properly recognized, an external capture input signal should be held for at least eight hw\_clk clock cycles before it changes its level.

During these eight hw\_clk clock cycles the capture input signals are scanned sequentially. When a timer is modified or incremented during this process, the new timer contents will already be captured for the remaining capture registers within the current scanning sequence.

*Note: If pin CCyIO is configured as output, the capture function may be triggered by modifying the corresponding port output latch via software, for example, for testing purposes.*

### 11.5.3.5 Compare Modes

The compare modes allow triggering of events (interrupts and/or output signal transitions) with minimum software overhead. In all compare modes, the 16-bit value stored in compare register CCy (in the following also referred to as 'compare value') is continuously compared with the contents of the allocated timer (Tx). If the current timer contents match the compare value, an appropriate output signal, which is based on the selected compare mode, can be generated at the corresponding output pin CCyIO and the associated interrupt request flag CCyIR is set, which can generate an interrupt request (if enabled).

As for capture mode, the compare registers are also processed sequentially during compare mode. When any two compare registers are programmed to the same compare value, their corresponding interrupt request flags will be set and the selected output signals will be generated within eight hw\_clk clock cycles after the allocated timer is incremented to the compare value. Further compare events on the same compare value are disabled until the timer is incremented again or written to by software. After a reset, compare events for register CCy only becomes enabled if the allocated timer has been

CONFIDENTIAL

CAPCOM 1 and 2

incremented or written to by software and one of the compare modes described in the following has been selected for this register.

The different compare modes which can be programmed for a given compare register **CCy** are selected by the mode control field **CCMz.MODy** in the associated capture/compare mode control register. In the following, each of the compare modes, including the special 'double-register' mode, is discussed in detail.

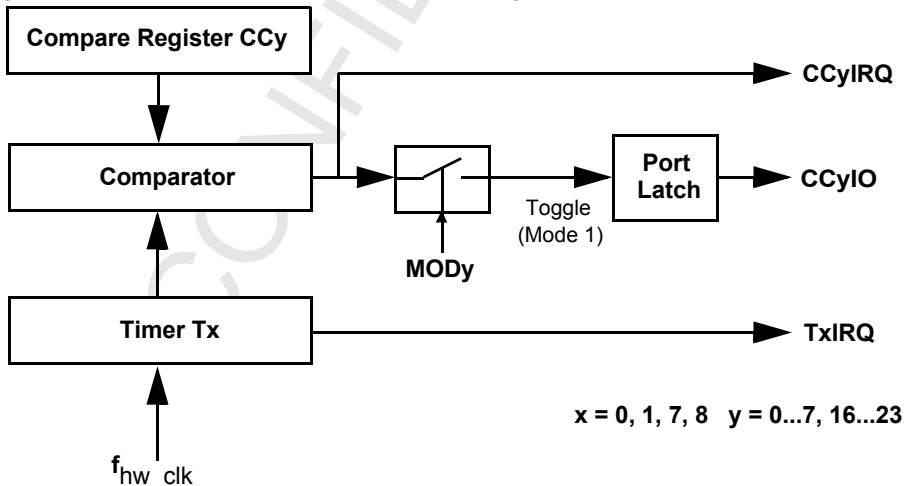
### Compare Mode 0

This is an interrupt-only mode which can be used for software timing purposes. Compare Mode 0 is selected for a given compare register **CCy** by setting bit field **CCMz.MODy** of the corresponding mode control register to 100<sub>B</sub>.

In this mode, the interrupt request flag CCyIR is set each time a match is detected between the content of compare register **CCy** and the allocated timer. Several of these compare events are possible within a single timer period, when the compare value in register **CCy** is updated during the timer period. The corresponding port signal CCyIO is not affected by compare events in this mode and can be used as general purpose IO.

*Note: If compare mode 0 is programmed for one of the registers CC...CC, the double-register compare mode may become enabled for this register if the corresponding bank 1 register is programmed to compare mode 1 (refer to [Section 11.5.3.6 Double-Register Compare Mode \(on Page 1096\)](#)).*

Figure 11-37 Compare Mode 0 and 1 Block Diagram



*Note: The OUT latch, the port latch, and the pin remain unaffected in compare mode 0.*

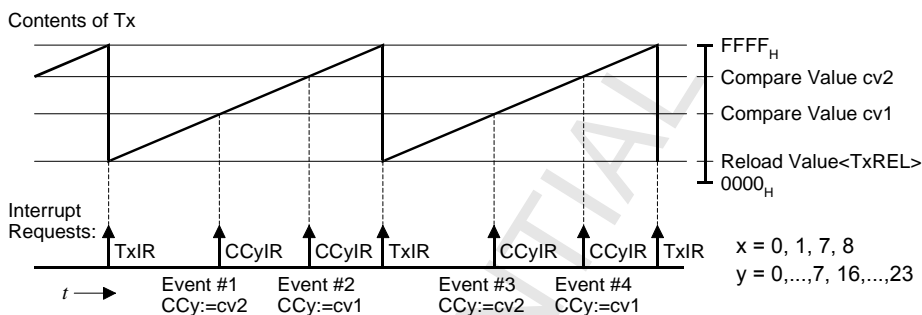


CONFIDENTIAL

CAPCOM 1 and 2

In the **Figure 11-38** example the compare value in register **CCy** is modified from cv1 to cv2 after compare events #1 and #3, and from cv2 to cv1 after events #2 and #4, etc. This results in periodic interrupt requests from timer Tx, and in interrupt requests from register **CCy** which occur at the time specified by the user through cv1 and cv2.

**Figure 11-38 Timing Example for Compare Modes 0 and 1**



*Note: Output signal CCyIO only effected in mode 1. No changes in mode 0.*

## Compare Mode 1

Compare Mode 1 is selected for register **CCy** by setting bit field **CCMz.MODy** of the corresponding mode control register to 101<sub>B</sub>.

When a match between the content of the allocated timer and the compare value in register **CCy** is detected in this mode, interrupt request flag CCyIR is set, and in addition the corresponding output signal CCyIO (alternate port output function) is toggled. For this purpose, the state of the respective port output latch (not the signal) is read, inverted, and then written back to the output latch.

Compare Mode 1 allows several compare events within a single timer period. An overflow of the allocated timer has no effect on the output signal nor does it disable or enable further compare events.

To use the respective port signal as compare signal output CCyIO for compare register **CCy** in Compare Mode 1, this port signal must be configured as output, that is, the corresponding direction control bit must be set. With this configuration, the initial state of the output signal can be programmed or its state can be modified at any time by writing to the port output latch.

In Compare Mode 1 the port latch is toggled upon each compare event (see **Figure 11-38** above).

*Note: If Compare Mode 1 is programmed for one of the registers **CC0...CC3** (**CC16...CC23**) or the double-register compare mode may become enabled for*

CONFIDENTIAL

CAPCOM 1 and 2

this register if the corresponding bank 1 register is programmed to Compare Mode 0 (refer to [Section 11.5.3.6 Double-Register Compare Mode \(on Page 1096\)](#)).

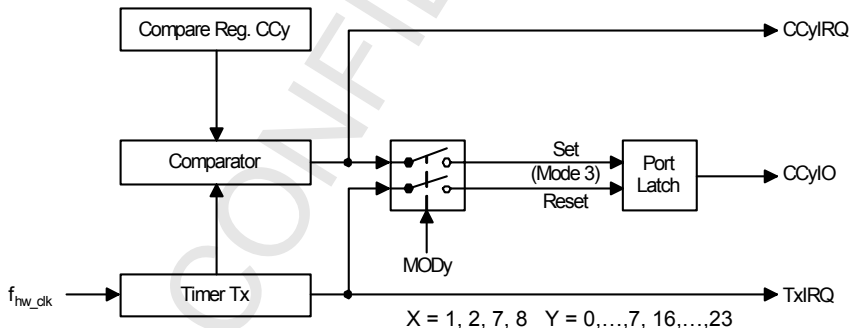
*Note: If the port output latch is written to by software at the same time it would be altered by a compare event, the software write has priority. In this case the hardware-triggered change does not become effective.*

## Compare Mode 2

Compare Mode 2 is an interrupt-only mode similar to Compare Mode 0, but only one interrupt request per timer period is generated. Compare Mode 2 is selected for register **CCy** by setting bit field **CCMz.MODy** of the corresponding mode control register to 110<sub>B</sub>. When a match is detected in Compare Mode 2 for the first time within a timer period, the interrupt request flag CCyIR is set. The corresponding port 2 signal is not affected and can be used for general purpose IO. However, after the first match has been detected in this mode all further compare events within the same timer period are disabled for compare register **CCy** until the allocated timer overflows. This means, that after the first match, even when the compare register is reloaded with a value higher than the current timer value, no compare event occurs until the next timer period.

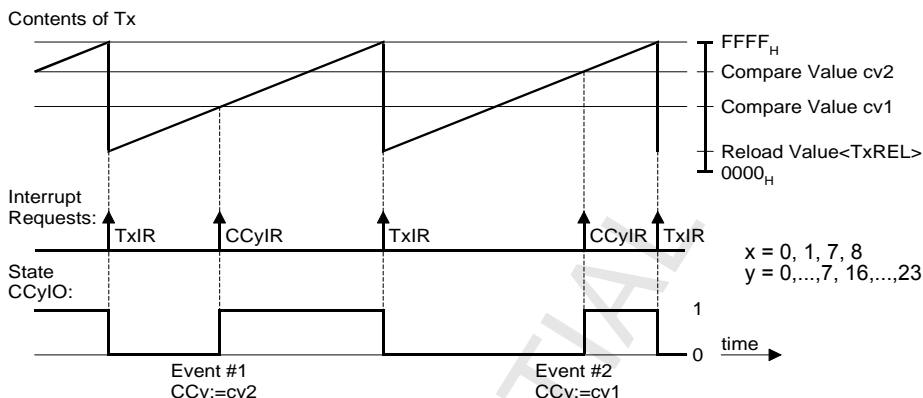
In the [Figure 11-40](#) example the compare value in register **CCy** is modified from cv1 to cv2 after compare event #1. Compare event #2, however, does not occur until the next period of timer Tx.

**Figure 11-39 Compare Mode 2 and 3 Block Diagram**



*The port latch and signal remain unaffected in compare mode 2.*

Figure 11-40 Timing Example for Compare Modes 2 and 3



Note: Only output signal CCyIO is effected in mode 3. There are no changes in mode 2.

### Compare Mode 3

Compare Mode 3 is selected for register **CCy** by setting bit field **CCMz.MODy** of the corresponding mode control register to  $111_B$ . In Compare Mode 3 only one compare event is generated per timer period.

When the first match within the timer period is detected the interrupt request flag CCyIR is set and the output pin CCyIO (alternate port function) is set. The signal is cleared when the allocated timer overflows.

If a match was found for register **CCy** in this mode, all further compare events during the current timer period are disabled for **CCy** until the corresponding timer overflows. If, after a match was detected, the compare register is reloaded with a new value, this value will not become effective until the next timer period.

To use the respective port signal as compare signal output signal CCyIO for compare register **CCy** in Compare Mode 3 this port signal must be configured as output, that is, the corresponding direction control bit must be set. With this configuration, the initial state of the output signal can be programmed or its state can be modified at any time by writing to the port output latch.

In Compare Mode 3 the port latch is set upon a compare event and cleared upon a timer overflow (see **Figure 11-40**).

However, when compare value and reload value for a channel are equal the respective interrupt requests are generated, only the output signal is not changed (set and clear would coincide in this case).

**CONFIDENTIAL**

**CAPCOM 1 and 2**

*Note: If the port output latch is written to by software at the same time it would be altered by a compare event, the software write has priority. In this case the hardware-triggered change does not become effective.*

### 11.5.3.6 Double-Register Compare Mode

In double-register compare mode two compare registers work together to control one output. This mode is selected by a special combination of modes for these two registers or via the **CCxDRM** register.

For double-register mode the eight capture/compare registers of a CAPCOM unit are regarded as two banks of 4 registers each. Registers **CC0...CC3** (**CC16...CC19**) form bank 1 while registers **CC4...CC7** (**CC20...CC23**) form bank 2. For double-register mode a bank 1 register and a bank 2 register form a register pair. Both registers of this register pair operate on the pin associated with the bank 1 register (pins **CC0IO...CC3IO**).

The relationship between the bank 1 and bank 2 register of a pair and the effected output pins for double-register compare mode is listed in **Table 11-23**.

**Table 11-23 Register Pairs for Double-Register Compare Mode**

<b>CAPCOM Unit 1</b>		
<b>Register Pair</b>		<b>Associated Output Pin</b>
<b>Bank 1</b>	<b>Bank 2</b>	
CC0	CC4	CC0IO
CC1	CC5	CC1IO
CC2	CC6	CC2IO
CC3	CC7	CC3IO
<b>CAPCOM Unit 2</b>		
<b>Register Pair</b>		<b>Associated Output Pin</b>
<b>Bank 1</b>	<b>Bank 2</b>	
CC16	CC20	
CC17	CC21	
CC18	CC22	
CC19	CC23	

The double-register compare mode can be programmed individually for each register pair. To enable double-register mode the respective bank 1 register (see **Table 11-23**) must be programmed to Compare Mode 1 and the corresponding bank 2 register must be programmed to Compare Mode 0 or via the **CCxDRM** register.

CONFIDENTIAL

CAPCOM 1 and 2

If the respective bits in **CCxDRM** are set to 00 and if the respective bank 1 compare register is disabled or programmed for a mode other than mode 1 the corresponding bank 2 register will operate in Compare Mode 0 (interrupt-only mode).

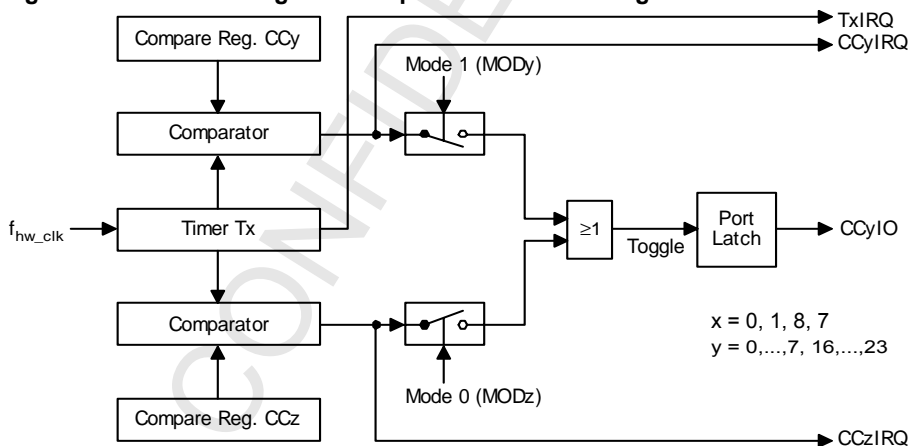
In the following, a bank 2 register (programmed to Compare Mode 0) is referred to as CCz while the corresponding bank 1 register (programmed to Compare Mode 1) will be referred to as CCy.

When a match is detected for one of the two registers in a register pair (CCy or CCz) the associated interrupt request flag (CCyIR or CCzIR) is set and pin CCyIO corresponding to bank 1 register CCy is toggled. The generated interrupt always corresponds to the register that caused the match.

*Note: If a match occurs simultaneously for both register CCy and register CCz of the register pair pin CCyIO are toggled only once but two separate compare interrupt requests are generated, one for vector CCyIRQ and one for vector CCzIRQ.*

To use the respective port signal as compare signal output signal CCyIO for compare register CCy in double-register compare mode, this port signal must be configured as output, that is, the corresponding direction control bit must be set. With this configuration, the output signal has the same characteristics as in Compare Mode 1.

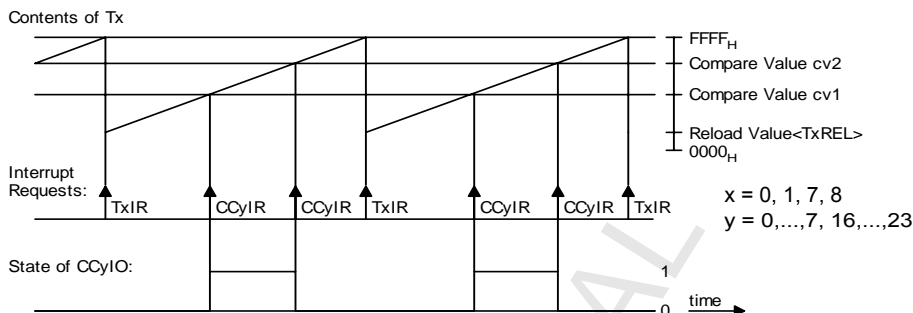
**Figure 11-41 Double-Register Compare Mode Block Diagram**



In this configuration example, the same timer allocation was chosen for both compare registers, but each register may also be individually allocated to one of the two timers of the respective CAPCOM unit. In the timing example for this compare mode (see **Figure 11-42**) the compare values in registers CCy and CCz are not modified.

*Note: The signals CCzIO (which do not serve for double-register compare mode) may be used for general purpose IO.*

**Figure 11-42 Timing Example for Double-Register Compare Mode**



### 11.5.3.7 Disabled Capture and Compare Mode

Both Capture and Compare modes are disabled by setting bit field **CCMz.MODy** of the corresponding mode control register to  $000_B$ .

The respective **CCy** registers then can serve for general variable storage. Also corresponding CCIOy pins can be used as General-Purpose Inputs/Outputs (GPIOs) and accessed directly through the **CCxOUT** register (refer to [Section 11.5.3.8.1 Alternative Implementation with Direct IO \(on Page 1099\)](#)).

### 11.5.3.8 I/O Control Register

Via the CAPCOM's I/O control register, the enhanced features of the CAPCOM I/Os are selected such as general variable storage and others.

Bit **CCyIOC.ORSEL** determines whether the port register or the **CCxOUT** register is visible at the pins (refer to [Chapter 11.5.3.8.1](#)). For compatibility reasons the port register is default output if no other module has a higher priority at the port (EBC).

Bit **CCyIOC.PL** disables changes of the port output register by the CAPCOM. This feature is only to be used when the port register is connected to the pins (**CCyIOC.ORSEL** = 0).

Bit **CCyIOC.STAG** enables the I/O stagger mechanism. The staggered mode is only used when the **CCxOUT** register is connected to the pins (**CCyIOC.ORSEL** = 1).

The maximum resolution of the CAPCOM is reduced to 1/8 if this mode was enabled and the timers are running with 1/8 of the hw\_clk clock. In this case the capture and compare functions are performed one after the other.

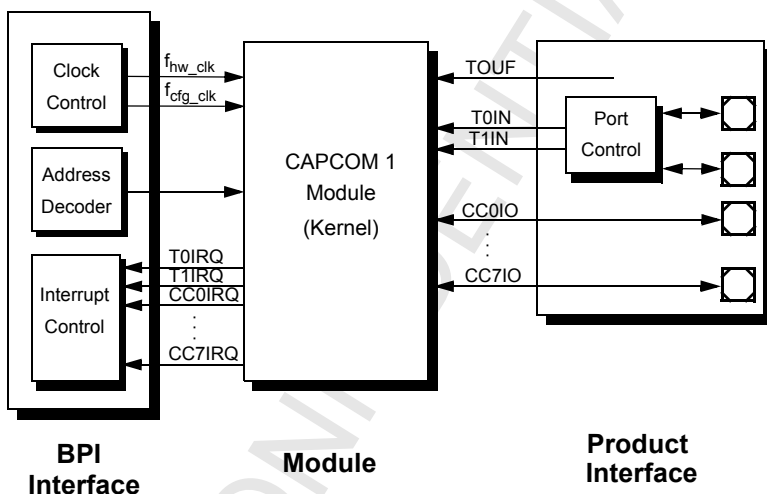
Bit **CCyIOC.PDS** selects the port direction when the CAPCOM is neither in Capture nor Compare mode (**CCMz.MODy** =  $000_B$ ), and when the **CCxOUT** register is connected to the pins (**CCyIOC.ORSEL** = 1). In such a case the Alternative Implementation is true.

### 11.5.3.8.1 Alternative Implementation with Direct IO

In addition to the default implementation described above (see [Figure 11-33](#)), the CAPCOM can also be implemented in a way that the CAPCOM registers are connected to the ports without going through the product interface's port control (**CCyIOC.ORSEL** = 1). Those bits of OUT register, currently in no capture/compare mode and hence serving as GPIOs, can be directly read/written from/to respective CCIOy pins. All such GPIOs are configured together by bit **CCyIOC.PDS**: either as outputs (**CCyIOC.PDS** = 0) or inputs (**CCyIOC.PDS** = 1).

[Figure 11-43](#) shows the alternative interface diagram of the CAPCOM Unit 1 (CAPCOM Unit 2 is identical).

**Figure 11-43 Interface Diagram for Alternative Implementation**



### 11.5.3.9 Interrupt Node

Upon a capture or compare event, the interrupt request flag CCyIR for the respective capture/compare register CCyIC is set. This flag can be used to generate an interrupt or trigger an interrupt service request when enabled by the interrupt enable bit CCyIE.

Capture interrupts can be regarded as external interrupt requests with the additional feature of recording the time at which the triggering event occurred (refer to [Section 11.5.5 Interrupts \(on Page 1118\)](#)).

Each of the 8 capture/compare registers (CC0...CC) has its own bit addressable interrupt control register (CC0IC...CCIC) and its own interrupt vector (CC0IRQ...CCIRQ). These registers are organized the same way as all other interrupt control registers (refer to [Section 6.3.3 Interrupt and Exception Execution \(on Page 111\)](#)).

**CONFIDENTIAL**

**CAPCOM 1 and 2**

### 11.5.3.10 Single Event Mode

The Single Event Mode is used to generate single edges or pulses and interrupts. It can be used with all compare modes.

For generating a single pulse the compare mode has to be set up and the **CCxSEM.SEMy** and **CCxSEE.SEEy** bits have to be set.

The **CCxSEE.SEEy** bit is cleared by HW after the event has occurred. By reading the register, the occurrence of the event can be controlled. Writing to this bit can be used to set and to clear it. Writing to this bit must be performed using bit protection.

### 11.5.4 Registers

The CAPCOM register mapping is in [Section 12.2 PD-Bus Register Addresses \(on Page 1273\)](#).

**Table 11-24 CAPCOM Register Summary**

Name	Access Condition	Clock	Description
<b>T01CON</b> <b>T78CON</b>	Bitaddressable	cfg_clk <sup>1)</sup>	CAPCOM 1 <b>T0</b> & <b>T1</b> control register CAPCOM 2 <b>T0</b> & <b>T1</b> control register
<b>CCM0</b> & <b>CCM1</b> <b>CCM4</b> & <b>CCM5</b>	Bitaddressable	cfg_clk <sup>1)</sup>	CAPCOM 1 Mode Control registers CAPCOM 2 Mode Control registers
<b>CC1ID</b> <b>CC2ID</b>	Bitaddressable	cfg_clk <sup>1)</sup>	CAPCOM 1 Identification register CAPCOM 2 Identification register
<b>CC1PISEL</b> <b>CC2PISEL</b>	Bitaddressable	cfg_clk <sup>1)</sup>	CAPCOM 1 Port Input Select register CAPCOM 2 Port Input Select register
<b>CC0</b> to <b>CC7</b> <b>CC16</b> to <b>CC23</b>	None	hw_clk <sup>1)</sup>	CAPCOM 1 Capture/Compare registers CAPCOM 2 Capture/Compare registers
<b>CC1SEE</b> <b>CC2SEE</b>	Bitaddressable, bit protected	hw_clk <sup>1)</sup>	CAPCOM 1 Single Event Enable register CAPCOM 2 Single Event Enable register
<b>CC1SEM</b> <b>CC2SEM</b>	Bitaddressable	cfg_clk <sup>1)</sup>	CAPCOM 1 Single Event Mode register CAPCOM 2 Single Event Mode register
<b>CC1DRM</b> <b>CC2DRM</b>	Bitaddressable	cfg_clk <sup>1)</sup>	CAPCOM 1 Double Register Mode register CAPCOM 2 Double Register Mode register
<b>CC1OUT</b> <b>CC2OUT</b>	Bitaddressable, bit protected	hw_clk <sup>1)</sup>	CAPCOM 1 Output register CAPCOM 2 Output register
<b>T0</b> <b>T7</b>	None	hw_clk <sup>1)</sup>	CAPCOM 1 Timer 0 registers CAPCOM 2 Timer 0 registers
<b>T0REL</b> <b>T7REL</b>	None	cfg_clk <sup>1)</sup>	CAPCOM 1 Timer 0 Reload registers CAPCOM 2 Timer 0 Reload registers



**CONFIDENTIAL**

**CAPCOM 1 and 2**

**Table 11-24 CAPCOM Register Summary**

<b>Name</b>	<b>Access Condition</b>	<b>Clock</b>	<b>Description</b>
<b>T1</b> <b>T8</b>	None	hw_clk <sup>1)</sup>	CAPCOM 1 Timer 1 registers CAPCOM 2 Timer 1 registers
<b>T1REL</b> <b>T8REL</b>	None	cfg_clk <sup>1)</sup>	CAPCOM 1 Timer 1 Reload registers CAPCOM 2 Timer 2 Reload registers
<b>CC1IOC</b> <b>CC2IOC</b>	Bitaddressable	cfg_clk <sup>1)</sup>	CAPCOM 1 I/O Control register CAPCOM 2 I/O Control register

<sup>1)</sup> Refer to Clock Domain in [System Integration: \(on Page 1083\)](#)

**CONFIDENTIAL**

**CAPCOM 1 and 2**

### 11.5.4.1 Timer Control Registers

#### TxxCON

#### T01CON

#### CAPCOM 1 Timer Control Registers

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	T1R	RESERVED	T1M					RESERVED	T0R	RESERVED		T0M			T0I

#### T78CON

#### CAPCOM 2 Timer Control Registers

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	T8R	RESERVED	T8M					RESERVED	T7R	RESERVED		T7M			T7I

Field	Bits	Type	Description
<b>TxI</b>	2:0, 10:8	rw	<b>Timer/Counter x Input Selection</b> <ul style="list-style-type: none"> <li>Timer Mode (<b>TxM</b> = 0): Input Frequency = <math>f_{hw\_clk}/2^{(&lt;TxI&gt;+3)}</math>. <i>Note: Refer to <a href="#">Table 11-22 Timing Examples (on Page 1088)</a> for examples.</i> </li> <li>Counter Mode (<b>TxM</b> = 1): <ul style="list-style-type: none"> <li>000 Overflow/Underflow of Timer_in</li> <li>001 Positive (rising) edge on TxIN</li> <li>010 Negative (falling) edge on TxIN</li> <li>011 Any edge (rising and falling) on TxIN</li> <li>1XX Reserved. Do not use these combinations.</li> </ul> </li> </ul>
<b>TxM</b>	11, 3	rw	<b>Timer/Counter x Mode Selection</b> <ul style="list-style-type: none"> <li>0 Timer Mode (input derived from internal clock)</li> <li>1 Counter Mode (input from external Input or T6)</li> </ul>
<b>TxR</b>	14, 6	rw	<b>Timer/Counter x Run Control</b> <ul style="list-style-type: none"> <li>0 Timer/Counter x is disabled</li> <li>1 Timer/Counter x is enabled</li> </ul>
<b>RESERVED</b>	15, 13:12, 7, 5:4	r	Reserved; these bits must be left at their reset values.

CONFIDENTIAL

CAPCOM 1 and 2

### 11.5.4.2 Capture/Compare Registers

#### CCy

**CC0** through **CC7** are the data registers for the capture or compare operations for CAPCOM1 Timers **T0** and **T1**.

**CC0**

**CC1**

**CC2**

**CC3**

**CC4**

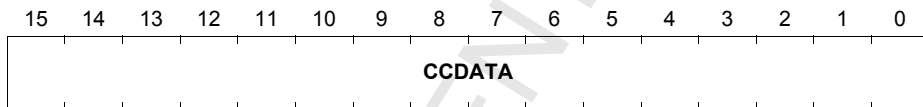
**CC5**

**CC6**

**CC7**

**Capture/Compare Registers for the CAPCOM 1**

**Reset value: 0000<sub>H</sub>**



**CC16** through **CC23** are the data registers for the capture or compare operations for CAPCOM 2 Timers **T7** and **T8**.

**CC16**

**CC17**

**CC18**

**CC19**

**CC20**

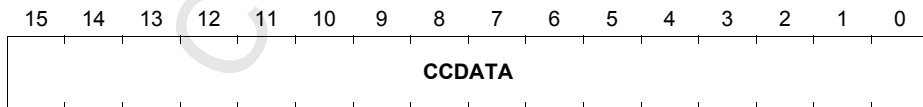
**CC21**

**CC22**

**CC23**

**Capture/Compare Registers for the CAPCOM 2**

**Reset value: 0000<sub>H</sub>**



Field	Bits	Type	Description
<b>CCDATA</b>	15:0	rw	<b>Capture/Compare Data Register</b>

**CONFIDENTIAL**

**CAPCOM 1 and 2**

### 11.5.4.3 Capture/Compare Mode Registers

#### CCMz

#### CCM0

**Capture/Compare Mode Registers for the CAPCOM 1 (CC0...CC3)**

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACC <sub>3</sub>	MOD3			ACC <sub>2</sub>	MOD2			ACC <sub>1</sub>	MOD1			ACC <sub>0</sub>	MOD0		

#### CCM1

**Capture/Compare Mode Registers for the CAPCOM 1 (CC4...CC7)**

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACC <sub>7</sub>	MOD7			ACC <sub>6</sub>	MOD6			ACC <sub>5</sub>	MOD5			ACC <sub>4</sub>	MOD4		

#### CCM4

**Capture/Compare Mode Registers for the CAPCOM 2 (CC16...CC19)**

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACC <sub>19</sub>	MOD19			ACC <sub>19</sub>	MOD18			ACC <sub>19</sub>	MOD17			ACC <sub>19</sub>	MOD16		

#### CCM5

**Capture/Compare Mode Registers for the CAPCOM 2 (CC20...CC23)**

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACC <sub>23</sub>	MOD23			ACC <sub>22</sub>	MOD22			ACC <sub>21</sub>	MOD21			ACC <sub>20</sub>	MOD20		

**CONFIDENTIAL**

**CAPCOM 1 and 2**

Field	Bits	Type	Description
<b>MODy</b> (y = 0 to 7 or 16 to 23)	2:0, 6:4, 10:8, 14:12	rw	<b>Mode Selection for Capture/Compare Register CCy</b> The available capture/compare modes are listed in <a href="#">Table 11-25</a> .
<b>ACCy</b> (y = 0 to 7 or 16 to 23)	15, 11, 7, 3	rw	<b>Allocation of Capture/Compare Register CCy</b> 0     CCy allocated to Timer <b>T0</b> . 1     CCy allocated to Timer <b>T1</b> .

**Table 11-25 Selection of Capture Modes and Compare Modes**

MODy	Selected Operating Mode
<b>000</b>	Disable Capture and Compare Modes. The respective CAPCOM registers may be used for general variable storage.
<b>001</b>	Capture on Positive Transition (Rising Edge) at Pin CCyIO.
<b>010</b>	Capture on Negative Transition (Falling Edge) at Pin CCyIO.
<b>011</b>	Capture on Positive and Negative Transition (Both Edges) at Pin CCyIO.
<b>100</b>	Compare Mode 0:     Interrupt Only Several interrupts per timer period; Enables double-register compare mode for registers CC4...CC7.
<b>101</b>	Compare Mode 1:     Toggle Output Pin on each Match. Several compare events per timer period; This mode is required for double-register compare mode for registers CC0...CC3.
<b>110</b>	Compare Mode 2:     Interrupt Only Only one interrupt per timer period.
<b>111</b>	Compare Mode 3:     Set Output Pin on each Match. Reset output pin on each timer overflow; Only one interrupt per timer period.

CONFIDENTIAL

CAPCOM 1 and 2

### 11.5.4.3.1 CAPCOM 1 & 2 Identification Registers

CCxID

CC1ID

CC2ID

CAPCOM Identification Register

Reset value: 5002<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Module_ID								Revision_Number							

Field	Bits	Type	Description
Revision_Number	0:7	r	<b>CAPCOM Revision Number</b> These hard-wired bits are used for module revision numbering.
Module_ID	8:15	r	<b>CAPCOM Identification Number</b> These hard-wired bits are used for module identification numbering.

**CONFIDENTIAL**

**CAPCOM 1 and 2**

#### 11.5.4.4 CAPCOM 1 & 2 Compare Output Registers

The CAPCOM's compare output serves two registers in parallel, the port output register for binary compatibility and a separate one for enhanced functionality. The CAPCOM compare output and the port output latch is muxed in the port logic.

*Note: Compare output is visible at the pin if Compare Mode 1 or 3 is programmed in **CCMz.MODy**.*

#### CCxOUT

#### CC1OUT

#### CC2OUT

#### Compare Output Registers

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								CC7/ 23 IO	CC6/ 22 IO	CC5/ 21 IO	CC4/ 20 IO	CC3/ 19 IO	CC2/ 18 IO	CC1/ 17 IO	CC0/ 16 IO

Field	Bits	Typ	Description
<b>CCyIO</b> (y = 0 to 7 or 16 to 23)	7:0	rwh	<b>Compare Output for Channel y</b> Alternative port output for port y. <i>Note: Refer to <b>CCyIOC</b>.</i>
<b>RESERVED</b>	15:8	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**CAPCOM 1 and 2**

### 11.5.4.5 Port Input Selection Registers

For switching between different port input sources the CAPCOM unit provides an input multiplexer. This multiplexer allows the selection between two input sources. The **CCxPISEL** registers control the switching of either each input channel or groups of input channels.

#### CCxPISEL

#### CC1PISEL

##### Port Input Select Register 1

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED										T1 INIS	T0 INIS	C7 C6 IS	C5 C4 IS	C3 C2 IS	C1 C0 IS

Field	Bits	Type	Description
<b>C1C0IS</b>	0	rw	<b>Select Source for Capture Input Channels</b>
<b>C3C2IS</b>	1		0 Default input port
<b>C5C4IS</b>	2		1 Alternate input port
<b>C7C6IS</b>	3		Refer to <a href="#">Figure 11-44</a> and <a href="#">Table 11-26</a> .
<b>TxINIS</b>	4	rw	<b>Select Source for External Input of Timer 0 Clock</b>
			0 T0IN pin selected (default)
			1 Tied to logical 0, no input received
<b>TxINIS</b>	5	rw	<b>Select Source for External Input of Timer 1 Clock</b>
			0 t_int1 signal from GSM Timer
			1 Tied to logical 0, no input received
<b>RESERVED</b>	15:6	r	Reserved; these bits must be left at their reset values.



## CC2PISEL

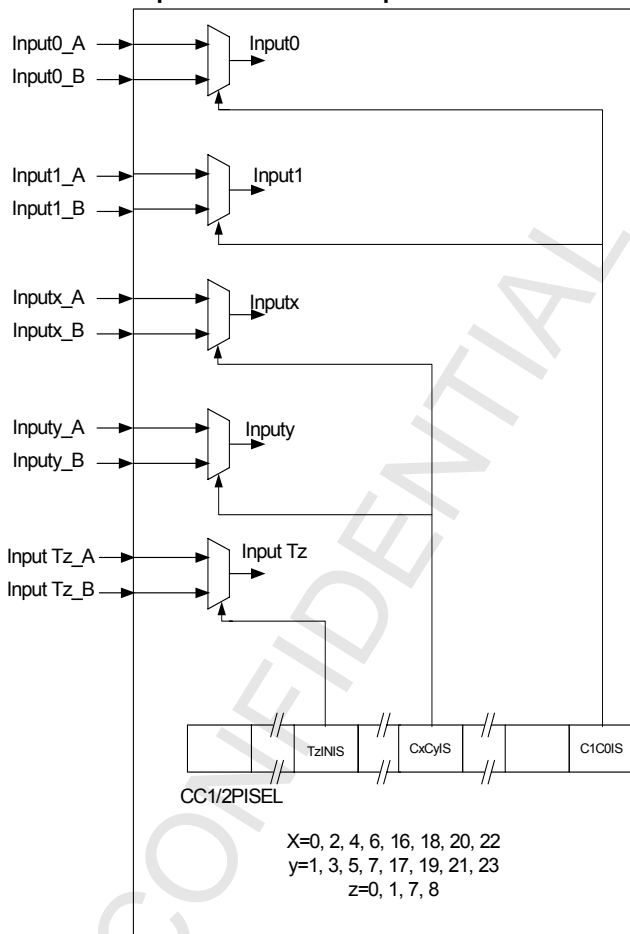
### Port Input Select Register 2

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED										T8 INIS	T7 INIS	C23 C22 IS	C21 C20 IS	C19 C18 IS	C17 C16 IS

Field	Bits	Type	Description
<b>C17C16IS</b>	0	rw	<b>Select Source for Capture Input Channels</b> 0 Default input port. 1 Alternate input port. Refer to <a href="#">Figure 11-44</a> and <a href="#">Table 11-27</a> .
<b>C19C18IS</b>	1		
<b>C21C20IS</b>	2		
<b>C23C22IS</b>	3		
<b>T0INIS</b>	4	rw	<b>Select Source for External Input of Timer 7 Clock</b> 0 T7IN pin selected (default) 1 T7IN pin selected (default)
<b>T1INIS</b>	5		<b>Select Source for External Input of Timer 8 Clock</b> 0 t_int2 signal from GSM Timer 1 Tied to logical 0, no input received
<b>RESERVED</b>	15:6	r	Reserved; these bits must be left at their reset values.

**Figure 11-44 Port Input Selection for Capture Channels**



**CONFIDENTIAL**
**CAPCOM 1 and 2**
**Table 11-26 CAPCOM 1 Input Signal Selection**

Pin Select Input (See <a href="#">Figure 11-44</a> )	Selected with Bit in <a href="#">CC1PISEL</a>	Ball Name <sup>1)</sup>	ALT Mode <sup>2)</sup>	Connected to Alternate Function
Input0_A	C1C0IS = 0	CC00IO SSC0_CLK	ALT 0 ALT 1	CC00IO CC00IO
Input0_B	C1C0IS = 1	ASC_CTS_n	ALT 2	CC00IOb
Input1_A	C1C0IS = 0	KP4	ALT 1	CC01IOa
Input1_B	C1C0IS = 1	KP4	ALT 1	CC01IOa
Input2_A	C3C2IS = 0	KP5 LPAOUT0	ALT 1 ALT 1	CC02IO CC02IO
Input2_B	C3C2IS = 1	RFSTR3	ALT 2	CC02IOb
Input3_A	C3C2IS = 0	SSC1_MTSR	ALT 2	CC03IO
Input3_B	C3C2IS = 1	SSC1_MTSR	ALT 2	CC03IO
Input4_A	C5C4IS = 0	TXD	ALT 1	CC04IO
Input4_B	C5C4IS = 1	TXD	ALT 1	CC04IO
Input5_A	C5C4IS = 0	I2S1_CLK0 I2S1_CLK	ALT 1 ALT 1	CC05IO CC05IO
Input5_B	C5C4IS = 1	I2S1_CLK0 I2S1_CLK	ALT 1 ALT 1	CC05IO CC05IO
Input6_A	C7C6IS = 0	I2S3_RX I2S3_TX	ALT 1 ALT 1	CC06IOa CC06IOa
Input6_B	C7C6IS = 1	I2S3_TX I2S3_WX	ALT 1 ALT 1	CC06IOb CC06IOb
Input7_A	C7C6IS = 0	I2S2_RX T_OUT2	ALT 1 ALT 1	CC07IO CC07IO
Input7_B	C7C6IS = 1	I2S2_RX T_OUT2	ALT 1 ALT 1	CC07IO CC07IO
InputT0_A	T0INIS = 0	I2S2_WA0	ALT 2	T0IN
InputT0_B	T0INIS = 1	Not connected		
InputT1_A	T1INIS = 0	Connected to GSM interrupt TINT1		
InputT1_B	T1INIS = 1	Not connected		

<sup>1)</sup> Refer to [Chapter 3 Pin Descriptions](#).

<sup>2)</sup> If there are two possible pads, choose one of the two. The pads are mutually exclusive.

**CONFIDENTIAL**

**CAPCOM 1 and 2**

**Table 11-27 CAPCOM 2 Input Signal Selection**

Pin Select Input (See <a href="#">Figure 11-44</a> )	Selected with Bit in <b>CC2PISEL</b>	Ball Name <sup>1)</sup>	ALT Mode <sup>2)</sup>	Connected to Alternate Function
Input0_A	C17C16IS = 0	SCCO_CLK	ALT1	CC00IO
Input0_B	C17C16IS = 1	ASC_CTS_n	ALT 2	CC00IO
Input1_A	C17C16IS = 0	SSC0_MTSR	ALT 1	CC17IO
Input1_B	C17C16IS = 1	SSC0_MTSR	ALT 1	CC17IO
Input2_A	C19C18IS = 0	KP9	ALT 1	CC18IOa
Input2_B	C19C18IS = 1	ASC_RTS_n	ALT 1	CC18IOb
Input3_A	C19C18IS = 0	T_OUT10	ALT 1	CC19IO
Input3_B	C19C18IS = 1	T_OUT10	ALT 1	CC19IO
Input4_A	C21C20IS = 0	ADV_n	ALT 1	CC20IOa
Input4_B	C21C20IS = 1	KP2	ALT 1	CC20IOb
Input5_A	C21C20IS = 0	OE_n	ALT 1	CC21IO
Input5_B	C21C20IS = 1	OE_n	ALT 1	CC21IO
Input6_A	C23C22IS = 0	KP8	ALT 1	CC22IOa
Input6_B	C23C22IS = 1	SSC1_MTSR	ALT 1	CC22IOb
Input7_A	C23C22IS = 0	SSC1_CLK	ALT 2	CC23IO
Input7_B	C23C22IS = 1	SSC1_CLK	ALT 2	CC23IO
InputT0_A	T7INIS = 0	KP7	ALT 1	T7IN
InputT0_B	T7INIS = 1	KP7	ALT 1	T7IN
InputT1_A	T8INIS = 0	Connected to GSM interrupt TINT2		
InputT1_B	T8INIS = 1	Not connected		

<sup>1)</sup> Refer to [Chapter 3 Pin Descriptions](#).

<sup>2)</sup> If there are two possible pads, choose one of the two. The pads are mutually exclusive.

CONFIDENTIAL

CAPCOM 1 and 2

### 11.5.4.6 Double-Register Compare Mode Register

CCxDRM

CC1DRM

CC2DRM

Double-Register Compare Mode Register

Reset value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								DR3M	DR2M	DR1M	DR0M				

Field	Bits	Type	Description
<b>DRxM</b> x = 0 to 3	1:0, 3:2, 5:4, 7:6	rw	<b>Double Register x compare Mode selection</b> 00 <b>DRxM</b> is controlled via compare modes 1 and 0. 01 <b>DRxM</b> disabled regardless of compare modes. 10 <b>DRxM</b> enabled regardless of compare modes. 11 Reserved <i>Note: x: 0 for registers 0 and 4, 1 for registers 1 and 5, etc.</i>
<b>RESERVED</b>	15:8	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**CAPCOM 1 and 2**

### 11.5.4.7 IOC Registers

**CCyIOC**

**CC1IOC**

**CC2IOC**

**I/O Control Register**

**Reset value: 0001<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												PDS	STA G	PL	OR SEL

Field	Bits	Type	Description
<b>ORSEL</b>	0	r	<b>Output Register Select</b> 0 The contents of the port register Py is visible at pad y (default). 1 The contents of the <b>CCxOUT</b> register is visible at the pad y. <i>Note: This bit is hardwired and it indicates which implementation is used for the CAPCOM (see <a href="#">Section 11.5.2</a> <a href="#">Section 11.5.2 (on page 1084)</a> and <a href="#">Section 11.5.3.8.1</a> <a href="#">Section 11.5.3.8.1 (on page 1099)</a>).</i>
<b>PL</b>	1	rw	<b>Port Lock</b> 0 The contents of the port register is changed by the CAPCOM unit (default). 1 The contents of the port register is not changed by the CAPCOM unit. <i>Note: Value of <b>PL</b> will be ignored when <b>ORSEL</b> = 1.</i>

**CONFIDENTIAL**

**CAPCOM 1 and 2**

Field	Bits	Type	Description
<b>STAG</b>	2	rw	<b>Stagger</b> 0 Staggered mode enabled (default). 1 Staggered mode disabled. <i>Note: <b>STAG</b> = 1 is reserved when <b>ORSEL</b> = 1.</i>
<b>PDS</b>	3	rw	<b>Port Direction Select</b> 0 Port direction is Output (default). 1 Port direction is Input. <i>Note: The <b>PDS</b> value is only relevant the following is true:</i> <ul style="list-style-type: none"> <li>When CAPCOM is in neither Capture nor Compare Mode (<b>CCMz.MODy</b> = 000<sub>B</sub>)</li> <li><b>CCxOUT</b> registers are connected to the pads (<b>CCxOUT.ORSEL</b> = 1).</li> </ul>
<b>RESERVED</b>	15:4	r	Reserved; these bits must be left at their reset values.

#### 11.5.4.8 CAPCOM 1 & 2 Single Event Mode Register

**CCxSEM**

**CC1SEM**

**CC2SEM**

**Single Event Mode Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								SEM 7	SEM 6	SEM 5	SEM 4	SEM 3	SEM 2	SEM 1	SEM 0

Field	Bits	Type	Description
<b>SEMy</b> (y = 0 to 7 or 16 to 23)	7:0	rw	<b>Single Event Mode selection</b> 0 Mode disabled for channel y 1 Mode enabled for channel y
<b>RESERVED</b>	15:8	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**CAPCOM 1 and 2**

### 11.5.4.9 CAPCOM 1 & 2 Single Event Enable Register

**CCxSEE**

**CC1SEE**

**CC2SEE**

**Single Event Enable Registers**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								SEE 7	SEE 6	SEE 5	SEE 4	SEE 3	SEE 2	SEE 1	SEE 0

Field	Bits	Type	Description
<b>SEE<sub>y</sub></b> (y = 0 to 7 or 16 to 23)	7:0	rwh	<b>Single Event Enable</b> 0 Event disabled for channel y. 1 Event enabled for channel y. This bit is cleared after the event has occurred by the HW.
<b>RESERVED</b>	15:8	r	Reserved; these bits must be left at their reset values.



CONFIDENTIAL

CAPCOM 1 and 2

## 11.5.4.10 Timer and Timer Reload Registers

### Timers

**T0** and **T1** (for CAPCOM 1), **T7** and **T8** (for CAPCOM 2) are the timer registers that store timer values.

**Tx**

**T0**

**T1**

**T7**

**T8**

**Timer Registers**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TxValue															

Field	Bits	Type	Description
<b>TxValue</b> x = 0,1,7, or 8	15:0	rw	Timer value

### Timer Reloads

**T0REL** and **T1REL** (for CAPCOM 1), **T7REL** and **T8REL** (for CAPCOM 2) are the timer reload registers that store the timer reload values.

**TxREL**

**T0REL**

**T1REL**

**T7REL**

**T8REL**

**Timer Reload Registers**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TxRELValue															

Field	Bits	Type	Description
<b>TxRELValue</b> x = 0,1,7, or 8	15:0	rw	Timer reload value

**CONFIDENTIAL**

**CAPCOM 1 and 2**

### 11.5.5 Interrupts

**Table 11-28** is an overview of the interrupts. For a detailed description refer to **Section 11.5.4 Registers (on Page 1100)**.

**Table 11-28 CAPCOM Interrupt Sources**

Interrupt	Signal	SRC Register	Description
Timer 0 overflow	int_T0_o		Interrupt is requested on overflow of Timer 0.
Timer 1 overflow	int_T1_o		Interrupt is requested on overflow of Timer 1.
Rising edge on Pin CCxIO	int_CCx_o		Interrupt is requested on positive transition at pin CCxIO in Capture Mode <b>TxxCON.MODy</b> = 001 <sub>B</sub> .
Falling edge on Pin CCxIO	int_CCx_o		Interrupt is requested on negative transition at pin CCxIO in Capture Mode <b>TxxCON.MODy</b> = 010 <sub>B</sub> .
Rising or falling edge on Pin CCxIO	int_CCx_o		Interrupt is requested on positive or negative transition at pin CCxIO in Capture Mode <b>TxxCON.MODy</b> = 011 <sub>B</sub> .
Match CCx and Ty	int_CCx_o		Interrupt is requested on detected match between content of compare register CCx and allocated Timer Ty in Compare Mode 0 ( <b>TxxCON.MODy</b> = 100 <sub>B</sub> ).
Match CCx and Ty	int_CCx_o		Interrupt is requested on detected match between content of compare register CCx and allocated Timer Ty in Compare Mode 1 ( <b>TxxCON.MODy</b> = 101 <sub>B</sub> ).
Match CCx and Ty	int_CCx_o		Interrupt is requested on detected match between content of compare register CCx and allocated Timer Ty in Compare Mode 2 ( <b>TxxCON.MODy</b> = 110 <sub>B</sub> ).
Match CCx and Ty	int_CCx_o		Interrupt is requested on detected match between content of compare register CCx and allocated Timer Ty in Compare Mode 3 ( <b>TxxCON.MODy</b> = 111 <sub>B</sub> ).
Match CCx and CCy	int_CCx_o or int_CCy_o		Interrupt is requested on detected match between contents of register pair CCx and CCy in Double Register Compare Mode.

CONFIDENTIAL

CONFIDENTIAL

**CONFIDENTIAL**

**RTC**

## 11.6 RTC

History	
Current version: Rev. 1.06, 2005-11-04	
Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev 0.02	
<b>Page 1133</b>	Update the <b>RTC_ID</b> register value
<b>Page 1142</b>	Updated <b>Table 11-30 Register Assignment to Clock and Reset Domains</b>
Changes for Rev. 1.01	
<b>Page 1132</b>	Updated <b>Figure 11-49 RTC Kernel Block Diagram</b> WS00006601
<b>Page 1142</b>	Updated <b>Table 11-30 Register Assignment to Clock and Reset Domains</b> WS00006601
Changes for Rev. 1.02	
<b>Page 1141</b>	Added <b>Figure 11-51 Differentiating Circuits for RTC interrupt</b> WS00006636
Changes for Rev. 1.04	
all	Corrected power supply domain to VDD_MAIN WS00008463
Changes for Rev. 1.06	

## **System Integration**

- Supply domain:
  - Register **RTCIF** & Bus Interface: VDD\_MAIN
  - All other parts: VDD\_RTC
- Chip internal interfaces:
  - Clock domains:
    - *pll\_clk\_i* for bus interface logic
    - software register controls selection of *pll\_clk\_i* or *rtc\_ref\_clk\_i* (32KHz) for internal counter logic
  - Bus domain: PD Bus
  - Interrupt sources: 2
  - Other interfaces: Padframe.
- Chip external signals related to this block :  
F32K, OSC32K, RTC\_OUT, RESET, PM\_INT.

### **11.6.1 Introduction**

The integrated Real Time Clock (RTC) is able to provide programmable alarm functions and external interrupts. Due to its extreme low power consumption the RTC can be supplied from a small backup battery. This allows the generation of external interrupts, even when the main PMB7870 supply voltage is switched off. For this purpose the RTC is powered by own voltage supply pins VDD\_RTC and VSS\_RTC.

The RTC shall be driven by a 32.768 kHz (32k) clock which needs to be applied via the PMB7870 F32K and OSC32K pins. The clock can be fed from either an external clock source or use the on chip 32 KHz oscillator module.

The low clock frequency and the optimized low power design give the possibility to run the chip with a minimum of power dissipation. For example, for this specific application the 26 MHz reference oscillator can be switched off during system standby and a low-power time reference can be kept when the 32k clock is provided to the RTC.

The RTC consists of an PMB7870 specific RTC shell (**p3\_rtc\_core**), containing the RTC macro (**p3\_rtc\_kernel**), as well as the 32 kHz oscillator, as described in the following sections. The module **p3\_rtc\_pre** solely performs level translation of the 32KHz clock to the VDD\_MAIN power supply domain, and is not functionally associated with the RTC.

In addition, another system function, the pad tristate control, is implemented in the RTC block. This is because this function needs the RTC power supply. Due to its strong relation with the system power management, the pad tristate control is described in

**Chapter 14 System Reset (on Page 1401).**

CONFIDENTIAL

RTC

## 11.6.2 RTC Register Overview

For the register addresses refer to [Section 12.2 PD-Bus Register Addresses \(on Page 1273\)](#).

**Table 11-29 RTC Register List**

Register Group	Register Name	Register Symbol
System Registers	RTC Identification	<a href="#">RTC_ID (on Page 1133)</a>
Control and Status Registers	RTC Shell Control RTC Control Register RTC Interface (Isolation)	<a href="#">RTC_CTRL (on Page 1124)</a> <a href="#">RTC_CON (on Page 1134)</a> <a href="#">RTCIF</a>
Data Registers	Timer T14 Count Timer T14 Reload RTC Count  RTC Reload  RTC Alarm  RTC Interrupt Sub Node Control.	<a href="#">RTC_T14_CNT (on Page 1136)</a> <a href="#">RTC_T14_REL (on Page 1136)</a> <a href="#">RTC_CNT_LO &amp; RTC_CNT_HI (on Page 1137)</a> <a href="#">RTC_REL_LO &amp; RTC_REL_HI (on Page 1138)</a> <a href="#">RTC_ALARM_LO &amp; RTC_ALARM_HI (on Page 1139)</a> <a href="#">RTC_ISNC (on Page 1140)</a>

## 11.6.3 RTC Shell

### 11.6.3.1 Register Descriptions

Only the registers of the RTC shell are described here. The main RTC register descriptions can be found in the RTC macro section.

*Note: The register [RTC\\_CTRL](#) (along with all other RTC registers) can only be accessed by software when the register [RTCIF](#) is set so that RTC isolation is disabled.*

CONFIDENTIAL

RTC

### 11.6.3.1.1 RTC Shell Control Register

RTC\_CTRL

RTC Shell Control Register

Reset value: 0200<sub>H</sub>  
on RTC SW reset only

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED					CLR RT CBA D	RTC BAD	CLR RT CINT	RESERVED			RTC CL K S EL	PU3 2K	32K EN	RTCI NT	RTC OUT EN

Field	Bits	Type	Description
RTCOUTEN	0	rw	<b>RTC External Interrupt Output Enable</b> 0: Disabled, RTC_OUT pin shows low level 1: Enabled
RTCINT	1	rh	<b>RTC Interrupt Status</b> 0: No RTC_INT interrupt occurred 1: RTC_INT interrupt occurred <i>Note: <b>RTCINT</b> needs to be cleared by the CPU by setting bit <b>CLR_RTCINT</b> in this register</i>
32KEN	2	rw	<b>32k Clock Enable</b> (See <a href="#">Figure 11-47 (on page 1129)</a> and <a href="#">Figure 11-48</a> ) 0: Disabled 1: Enabled
PU32K	3	rw	<b>32 kHz Oscillator Power Up</b> (See <a href="#">Figure 11-47</a> and <a href="#">Figure 11-48</a> ) 0: Disabled 1: Enabled



**CONFIDENTIAL**

**RTC**

Field	Bits	Type	Description
<b>RTC_CLK_SEL</b>	4	rw	<p><b>RTC Logic Clock Select</b></p> <p>0: 32 kHz clock operation mode (Asynchronous to microcontroller clock, low power, read only)</p> <p>1: Bus clock operation mode (Synchronous to microcontroller clock, required for register write operation for some registers, refer to <a href="#">Table 11-30 Register Assignment to Clock and Reset Domains</a> (on <a href="#">Page 1142</a>))</p> <p>Upon changing this bit from 32 kHz to BPI clock, RTC register access can only begin when <b>RTC_CON.ACCPOS</b> is active.</p> <p><i>Note: The clock must not be switched when the Bus clock frequency is less than or equal to 32 kHz.</i></p>
<b>CLR_RTCINT</b>	8	w	<p><b>Clears RTC_INT</b></p> <p>0 No action</p> <p>1 Clears <b>RTCINT</b></p> <p>Reading always returns 0.</p>
<b>RTCBAD</b>	9	rh	<p><b>RTC Content Inconsistent Due to Power Supply Drop Down</b></p> <p>0: RTC content consistent</p> <p>1: A error in the RTC counter has been detected. This bit is set by hardware.</p> <p><i>Note: To clear this bit software must perform write operations to the <a href="#">RTC_CNT_LO</a> &amp; <a href="#">RTC_CNT_HI</a> registers, followed by software asserting <b>CLR_RTCBAD</b> in this register. This must be done while the RTC operates in the synchronous mode.</i></p>
<b>CLR_RTCBAD</b>	10	w	<p><b>Clear RTCBAD</b></p> <p>0 No action</p> <p>1 Clears <b>RTCBAD</b> Also refer to <b>RTCBAD</b> for details on clearing this bit.</p> <p>Reading always returns 0.</p>

Field	Bits	Type	Description
RESERVED	7:5, 15:11	r	Reserved; these bits must be left at their reset values.

### 11.6.3.1.2 RTC Interface Register

#### RTCIF

#### RTC Interface Register

Reset value: 0000<sub>H</sub>

This register is defined in section [RTCIF](#). The value must be set to the Enabled value specified to allow access to the RTC registers.

*Note: The register [RTCIF](#) is not in the RTC voltage supply domain but in the same supply domain as the CGU. It is also reset with the reset for the CGU.*

### 11.6.3.2 Internal Interrupts

The RTC may generate 2 internal interrupt signals *rtc\_int* and *rtc\_t14int* from 6 interrupt sources. The interrupt sources are ALARM, RTC0..3 and T14.

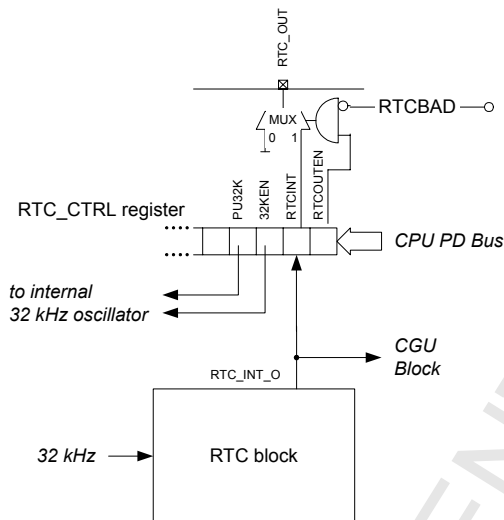
- *rtc\_t14int* is driven directly by the T14 interrupt
- *rtc\_int* is driven by combining the enabled 6 interrupt sources. Each of the 6 interrupt sources may be enabled independently, as described in [Section 11.6.5.4.11 \(on page 1140\)](#).

### 11.6.3.3 External Interrupt RTC\_OUT

The interrupt output *rtc\_int* of the RTC macro sets bit [RTC\\_CTRL.RTCINT](#). Once an interrupt has been detected, bit **RTCINT** remains at the logic HIGH level until it is reset by the CPU. The status of bit **RTCINT** is directed to the RTC\_OUT pin, if bit [RTC\\_CTRL.RTCOUTEN](#) is set and the [RTC\\_CON.RTCBAD](#) bit is not set (see [Figure 11-45](#)).

*Note: RTC\_OUT (as well as *rtc\_t14int* and *rtc\_int*) are undefined before the RTC is reset. The overall system has to handle this state appropriately.*

Figure 11-45 RTC Interrupt Generation



#### 11.6.3.4 Content Validity Check

A feature of the RTC block in the PMB7870 is the data content validity check capability. There are two registers capturing a part of the RTC counter in inverted and non-inverted form (6-bit part of the **RTC\_CNT\_HI** (on Page 1137) register field and its shadow register). Both registers are hardware-incremented/decremented according to the RTC operation. In the case of a RTC battery voltage drop-out or replacement, the content of those two registers will no longer be complementary. This error is latched into the **RTC\_CON.RTCBAD** register bit and can be detected by the software which may force the user to update the time and date information. The **RTC\_CON.RTCBAD** register bit then has to be cleared by software.

**11.6.3.5** *There is a VDD voltage range in which registers can not be incremented any more but they keep their value. When the VDD drops into this range, the registers stay complementary but they are not up-to-date any more. If the VDD voltage is stored externally in large capacitor, this is even likely to happen. Thus the content validity check may not be useful in some applications.*

**RTC Software Reset**

The RTC can be reset by software-reset only. By this means the RTC keeps its value during the PMB7870 hardware-reset. Refer to **Reset Control and Status Register** (on Page 683) for details. It becomes effective only if the **RTCIF** register is set so that RTC Isolation is disabled.

The RTC is in an undefined state when VDD\_RTC has dropped below the minimum specified value (first unit initialization or power-up after the RTC-backup cell died). During the time between the VDD power-up and controller software-reset, the RTC is in an undefined state.

### **11.6.3.6 RTC Isolation**

The RTC unit together with its related pads may be isolated electrically from the other parts of the E-GOLDradio and work completely autonomously in an asynchronous mode. This has to be done before the chip is put in the powerdown mode. The isolation helps also to save power on the RTC power supply because the BPI clock domain registers are not then clocked. Therefore the isolation may be used every time when the connection to the RTC is not required by the controller.

The isolation mode is controlled by the register **RTCIF**, which is in the VDD\_MAIN supply domain. **RTCIF** is actually implemented in the CGU. Isolation mode is enabled upon power on reset and can be removed only by writing "10101010" to the register. If any other value is written to **RTCIF**, the RTC is isolated. It is strongly recommended to write "00000000" to **RTCIF** for isolation, because a 1 may cause some current flow during the drop down phase of VDD\_MAIN.

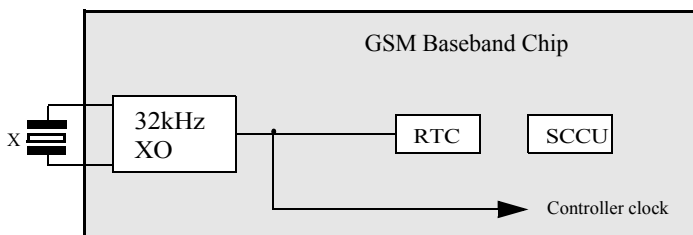
In the isolated state the RTC runs in the Asynchronous mode. Although the hardware forces this mode automatically when the isolated state is reached, the software should switch to Asynchronous mode before isolating the RTC in order to avoid undefined states during the transition from Not-Isolated to Isolated.

Isolation mode of the RTC is primarily intended to allow the RTC to operate when the supply voltage to the core has been removed. Before this occurs, the software must put the RTC in Asynchronous mode (i.e. select entire RTC operation from the 32KHz clock rather than the bus clock). The 8-bits from the **RTCIF** register are passed to permanently enabled level shifters located in the padframe, which have the characteristic of storing the input **RTCIF** value (on their outputs) before the inputs float down, as in the case when the core power is removed. By writing a value to **RTCIF** which selects isolation (preferably all zeros) prior to core power removal, all input level shifters within the RTC are disabled, which means that all input signals to the RTC will remain in a defined state during the time core power drops off, and allows the RTC to then operate autonomously when the core power has been removed.

### **11.6.4 32k Oscillator**

The 32k oscillator is used primarily for the RTC clock generation. It also supports the CGU block ([Section 10.4.1.5.7 Reset Control and Status Register \(on Page 683\)](#)) with a reference clock. Additionally the 32k clock can be used for controller operation in a power-saving mode. The major oscillator parameters are low power consumption and a wide operating voltage range. The oscillator is in the same supply domain as the RTC.

Figure 11-46 MS Clock Generation with On-Chip Oscillators



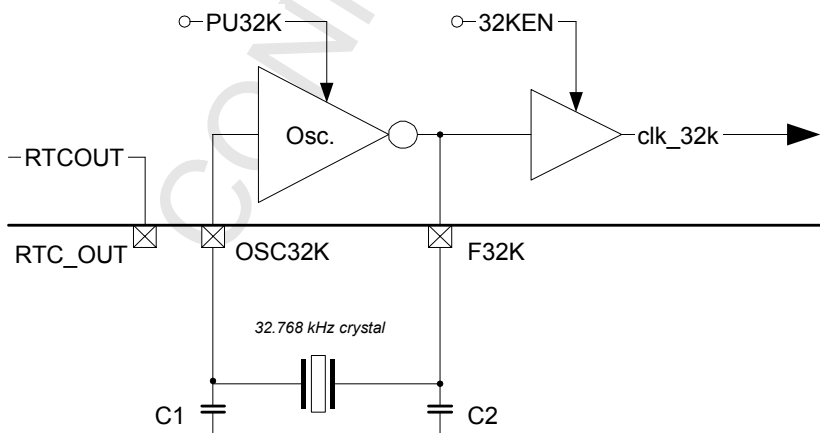
The user can use the on-chip oscillator or an external 32kHz circuitry (e.g. in a power management ASIC). If used, the 32k oscillator requires 2 pins for the external quartz.

The 32k oscillator block has a register interface to the controller via the PD Bus. The RTC block itself can operate in two modes: in an Asynchronous mode and a Synchronous mode. In the Asynchronous mode, the RTC block runs with the 32k clock. The register access is limited to those indicated as being in the Bus Clock domain in [Table 11-30](#). To enable the complete register access the RTC has to be (temporarily) switched into the Synchronous mode. In the Synchronous mode, the block logic and all registers run with the bus clock.

The figure below shows the basic circuitry required by the internal 32k oscillator.

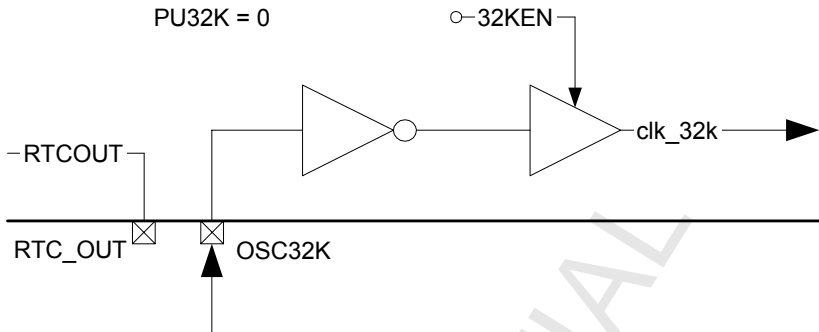
Externally connected are the crystal, the load capacitances  $C_1$  and  $C_2$  and the resistor  $R_1$  which is used to limit the drive level. Recommended values are given in [Section 13.2.1.2.9.1 Timing of RTC \(on Page 1360\)](#).

Figure 11-47 Circuitry to Use the Internal 32 kHz Oscillator



The Oscillator Block can be enabled by setting bit [RTC\\_CTRL.PU32K](#) of the RTC block. Bit [32KEN](#) in the same register enables the clock `clk_32k`.

**Figure 11-48 Circuitry to Use the External 32 kHz Oscillator**



If the 32kHz signal is fed in via the OSC32K input pad, the signal levels correspond to those specified in [Section 13.2.1.1.2 Voltages \(on Page 1303\)](#).

## 11.6.5 RTC Macro

The Real Time Clock (RTC) module is basically an independent timer chain and counts clock ticks. The base frequency of the RTC can be programmed via a reload counter. The RTC can work fully asynchronous to the system frequency and is optimized for low power consumption.

### Features

- Two reloadable timers, T14 (16-bit) and CNT (32-bit)
- Timers can operate in Synchronous or Asynchronous Mode.

### 11.6.5.1 Operational Overview

The real time clock module provides three different types of registers:

- Control registers for controlling RTC functionality
- Data registers for setting the clock divider for RTC base frequency programming and for flexible interrupt generation
- Counter registers, which contain the actual time and date.

The interrupts are programmed via one interrupt sub node register (refer to [Section 11.6.3.2 Internal Interrupts \(on Page 1126\)](#)).

### 11.6.5.2 Register Overview

Refer to [Table 11-29 RTC Register List \(on Page 1123\)](#).

### **11.6.5.3 Functional Overview**

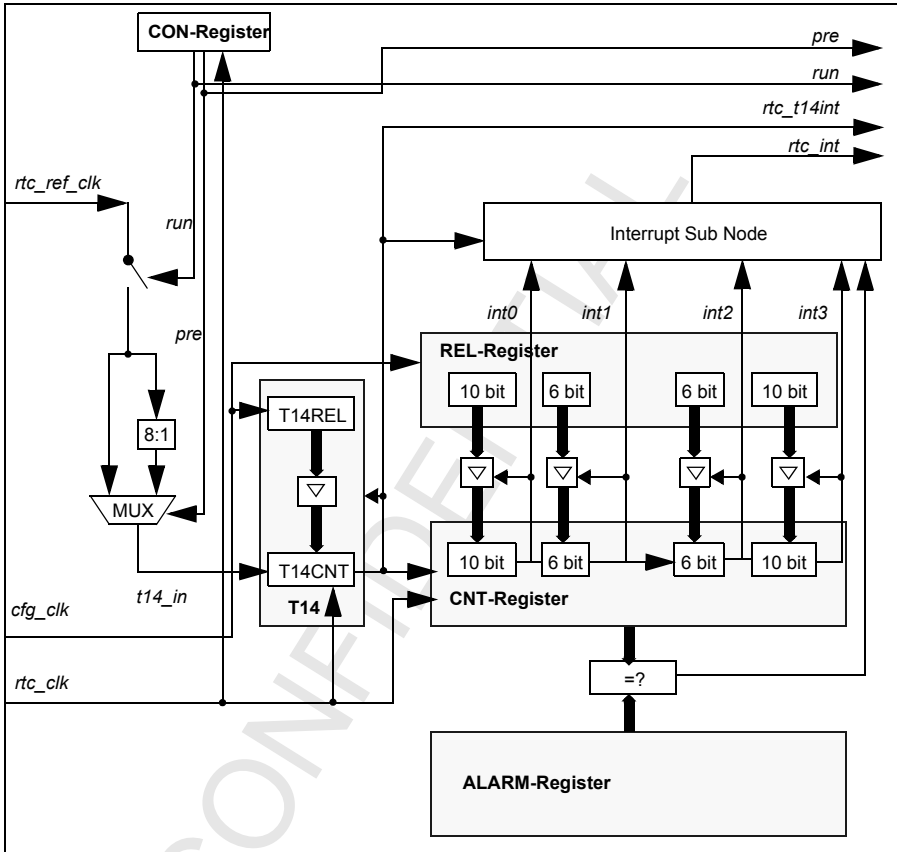
The RTC module consists of a chain of 2 divider blocks, the reloadable 16-bit timer T14 and the 32-bit RTC timer. Both timers count up. Each timer contains reload registers (**RTC\_T14\_REL** and **RTC\_REL\_LO/RTC\_REL\_HI**) and count registers (**RTC\_T14\_CNT** and **RTC\_CNT\_LO/RTC\_CNT\_HI**). **RTC\_T14\_CNT** is reloaded with the value of **RTC\_T14\_REL** on every **RTC\_T14\_CNT** overflow. The 32-bit RTC timer (**RTC\_CNT\_LO/RTC\_CNT\_HI**) is split into four smaller related sections (10-bit/6-bit/6-bit/10-bit). The reload parts of these sections are grouped to register **RTC\_REL\_LO/RTC\_REL\_HI** and the count parts to register **RTC\_CNT\_LO/RTC\_CNT\_HI**.

The count input of the RTC module can be optional divided by a prescaler with factor 8.

The RTC module operates in two different modes, an Asynchronous and a Synchronous Mode. In Synchronous Mode the RTC module is clocked with a synchronous clock derived from the bus clock. In Asynchronous Mode the RTC module is clocked by the 32K clock. The Asynchronous Mode is necessary to allow a lower frequency or disabled Bus Clock (for example, Power Down Mode). The accessibility of the RTC registers Asynchronous Mode is described in [Section 11.6.5.5.1 Clock Operation \(on Page 1142\)](#).

### 11.6.5.3.1 RTC Diagrams

**Figure 11-49 RTC Kernel Block Diagram**





CONFIDENTIAL

RTC

## 11.6.5.4 Registers

### 11.6.5.4.1 RTC Identification Register

#### RTC Identification Register

RTC\_ID

RTC Identification Register

Reset value: 4902<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Module_ID								Revision_Number							

Field	Bits	Type	Description
Revision_Number	0:7	r	<b>Revision Number</b> These hard-wired bits are used for module revision numbering.
Module_ID	8:15	r	<b>Module Identification Number</b> These hard-wired bits are used for module identification numbering.

**CONFIDENTIAL**

**RTC**

### 11.6.5.4.2 RTC Control Register

**RTC\_CON**

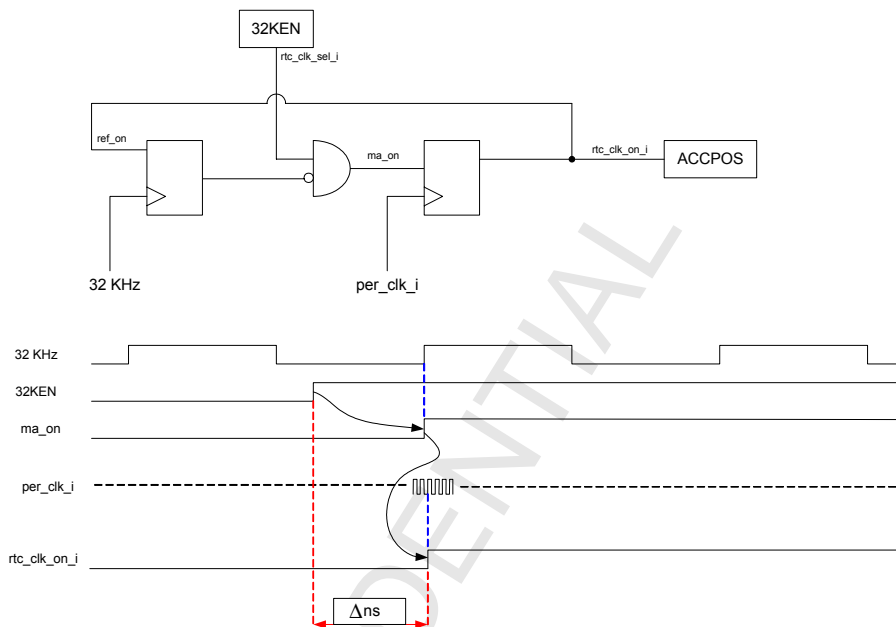
**RTC Control Register**

**Reset Value: 0003<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>ACC POS</b>	<b>RESERVED</b>											<b>T14 INC</b>	<b>T14 DEC</b>	<b>PRE</b>	<b>RUN</b>

Field	Bits	Type	Description
<b>RUN</b>	0	rw	<b>RTC Enable</b> 0 Stopped 1 Runs
<b>PRE</b>	1	rw	<b>RTC Input Source Pre-Scaler Enable</b> 0 Disabled 1 Enabled
<b>T14DEC</b>	2	wh	<b>Decrement T14 Timer Value</b> 0 No action 1 T14 decremented The bit is automatically cleared by hardware after decrementing T14. Reading returns a zero.
<b>T14INC</b>	3	wh	<b>Increment T14 Timer Value</b> 0 No action 1 T14 incremented The bit is automatically cleared by hardware after incrementing T14. Reading returns a zero.
<b>ACCPOS</b>	15	rh	<b>RTC Register Access Possible</b> This bit indicates that synchronous read / write access to RTC registers is possible. Note that the Asynchronous/Synchronous mode select bit <b>RTC_CTRL.RTC_CLK_SEL</b> can always be accessed in either mode (see <a href="#">Figure 11-50</a> ). 0 No write access is possible, only asynchronous reads 1 Read and Write accesses are possible
<b>RESERVED</b>	14:4	r	Reserved, these bits must be left at their reset values.

Figure 11-50 ACCPOS Bit Generation



*Note:*  $\Delta ns$  is required to get the ACCPOS bit set to one when switching from the asynchronous to synchronous mode.

#### 11.6.5.4.3 Prescaler Timer T14 Count Register

*Note:* **RTC\_T14\_CNT** and **RTC\_T14\_REL** are physically located inside one 32-bit register in the RTC. The 32-bit register is updated when register **RTC\_T14\_CNT** is written to by software. Software must write the required data to the **RTC\_T14\_REL** address immediately before this to ensure that both 16-bit values are (simultaneously) written to the 32-bit register.

CONFIDENTIAL

RTC

## RTC\_T14\_CNT

Timer T14 Count Register

Reset Value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T14CNT															

Field	Bits	Type	Description
T14CNT	15:0	rwh	T14 Counter

### 11.6.5.4.4 Prescaler Timer T14 Reload Register

Note: Refer to [RTC\\_T14\\_CNT](#) for instructions on how to update this register.

## RTC\_T14\_REL

Timer T14 Reload Register

Reset Value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T14REL															

Field	Bits	Type	Description
T14REL	15:0	rw	T14 Reload Value

### 11.6.5.4.5 RTC Count Register (High Word)

Note: [RTC\\_CNT\\_HI](#) and [RTC\\_CNT\\_LO](#) are physically located inside one 32-bit register in the RTC. The 32-bit register is updated when register [RTC\\_CNT\\_HI](#) is written to by software. Software must write the required data to the [RTC\\_CNT\\_LO](#) address immediately before this to ensure that both 16-bit values are (simultaneously) written to the 32-bit register.

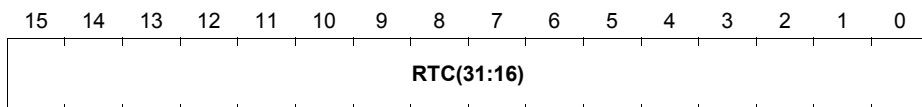
CONFIDENTIAL

RTC

#### RTC\_CNT\_HI

RTC Count Register - High

Reset Value: 0000<sub>H</sub>



Field	Bits	Type	Description
RTC(31:16)	15:0	rwh	RTC Counter Bits 31:16

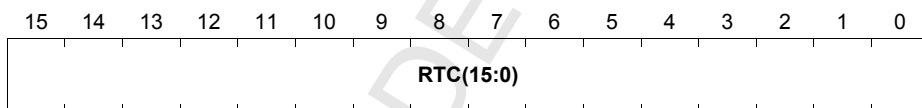
#### 11.6.5.4.6 RTC Count Register (Low Word)

Note: Refer to [RTC\\_CNT\\_HI](#) for instructions on how to update this register.

#### RTC\_CNT\_LO

RTC Count Register - Low

Reset Value: 0000<sub>H</sub>



Field	Bits	Type	Description
RTC(15:0)	15:0	rwh	RTC Counter Bits 15:0

#### 11.6.5.4.7 RTC Reload Register (High Word)

Note: [RTC\\_REL\\_HI](#) and [RTC\\_REL\\_LO](#) are physically located inside one 32-bit register in the RTC. The 32-bit register is updated when register [RTC\\_REL\\_HI](#) is written to by software. Software must write the required data to the [RTC\\_REL\\_LO](#) address immediately before this to ensure that both 16-bit values are (simultaneously) written to the 32-bit register.

#### RTC\_REL\_HI

##### RTC Reload Register - High

Reset Value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REL(31:16)															

Field	Bits	Type	Description
REL(31:16)	15:0	rw	RTC Reload Value Bits 31:16

#### 11.6.5.4.8 RTC Reload Register (Low Word)

Refer to [RTC\\_REL\\_HI](#) for instructions on how to update this register.

#### RTC\_REL\_LO

##### RTC Reload Register - Low

Reset Value: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REL(15:0)															

Field	Bits	Type	Description
REL(15:0)	15:0	rw	RTC Reload Value Bits 15:0

#### 11.6.5.4.9 RTC Alarm Register (High Word)

Refer to [RTC\\_ALARM\\_HI](#) and [RTC\\_ALARM\\_LO](#) are physically located inside one 32-bit register in the RTC. The 32-bit register is updated when register [RTC\\_ALARM\\_HI](#) is written to by software. Software must write the required data to the [RTC\\_ALARM\\_LO](#) address immediately before this to ensure that both 16-bit values are (simultaneously) written to the 32-bit register.

CONFIDENTIAL

RTC

## RTC\_ALARM\_HI

RTC Alarm Register - High Word

Reset Value: FFFF<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALARM(31:16)															

Field	Bits	Type	Description
ALARM(31:16)	15:0	rw	Alarm Time Bits 31:16

### 11.6.5.4.10 RTC Alarm Register (Low Word)

Refer to [RTC\\_ALARM\\_HI](#) for instructions on how to update this register.

## RTC\_ALARM\_LO

RTC Alarm Register - Low Word

Reset Value: FFFF<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALARM(15:0)															

Field	Bits	Type	Description
ALARM(15:0)	15:0	rw	Alarm Time Bits 15:0

**CONFIDENTIAL**

**RTC**

### 11.6.5.4.11 RTC Interrupt Sub Node Control

See [Figure 11-51 Differentiating Circuits for RTC interrupt](#).

#### RTC\_ISNC

#### RTC Interrupt Sub Node Control

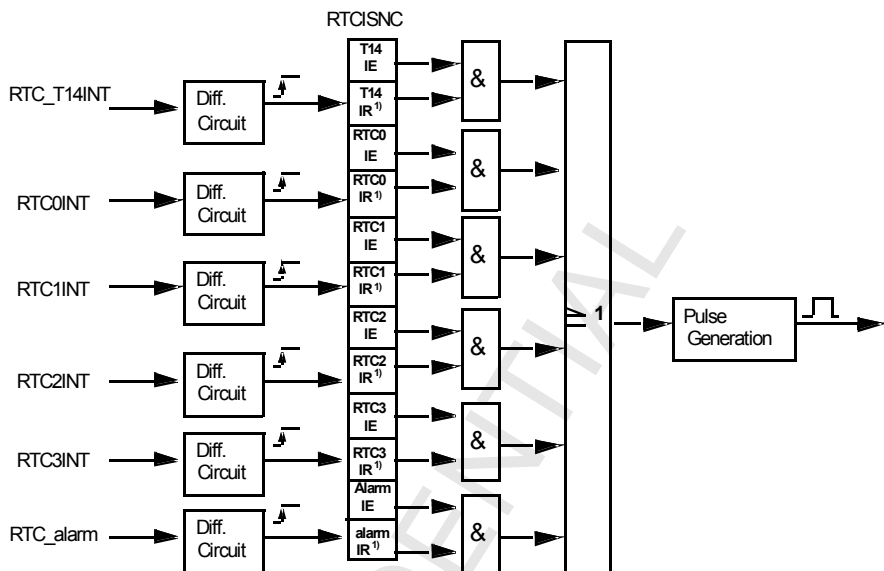
**Reset Value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				ALARMIR	ALARMIE	RTC3IR	RTC3IE	RTC2IR	RTC2IE	RTC1IR	RTC1IE	RTC0IR	RTC0IE	T14IR	T14IE

Field	Bits	Type	Description
<b>T14IE</b>	0	rw	<b>T14 Overflow Interrupt Enable Control</b> 0 Disables interrupt request 1 Enables interrupt request
<b>T14IR</b>	1	rw	<b>T14 Overflow Interrupt Request</b> 0 No request pending 1 This source has raised an interrupt request Write 0 to clear this interrupt bit.
<b>RTCxIE</b>	8,6,4,2	rw	<b>Interrupt Enable Control x</b> 0 Disables interrupt request 1 Enables interrupt request
<b>RTCxIR</b>	9,7,5,3	rw	<b>Interrupt Request</b> 0 No request pending 1 This source has raised an interrupt request Write 0 to clear this interrupt bit.
<b>ALARMIE</b>	10	rw	<b>Alarm Interrupt Enable Control</b> 0 Disables interrupt request 1 Enables interrupt request
<b>ALARMIR</b>	11	rw	<b>Alarm Interrupt Request (bit protected)</b> 0 No request pending 1 This source has raised an interrupt request Write 0 to clear this interrupt bit.
<b>RESERVED</b>	15:12	r	Reserved, these bits must be left at their reset values.



Figure 11-51 Differentiating Circuits for RTC interrupt



1) The ISR (Interrupt Service Routine) must delete the IR flag in register RTCISNC manually. Otherwise no further interrupts can be detected.

## 11.6.5.5 Functions

### 11.6.5.5.1 Clock Operation

The following clocking issues have to be considered when the RTC module runs in Synchronous Mode (**RTC\_CTRL.RTC\_CLK\_SEL** = 1).

The module derives its Kernel Clock (*rtc\_clk\_s*) and the Bus Clock (*bpi\_clk\_s*) from the *per\_clk\_i* input by Clock Gating with the input signals *rtc\_gating\_en\_inv\_i* and *bpi\_clk\_en\_i*. The following relation between the Kernel and Bus Clocks is required:

- The Kernel Clock has the same speed, or is faster than, the Bus Clock.
- When a Bus Clock pulse is active then a Kernel Clock pulse must be also active.

In Asynchronous Mode (**RTC\_CLK\_SEL** = 0) the Kernel Clock is driven by *clk\_32k* and no write accesses to the registers in this clock domain are possible. Write access is possible to those which are indicated as being in the Bus Clock domain in **Table 11-30**. A read access is possible, but correct read data can not be guaranteed due to the possibility of an asynchronous data change during the read access.

**Table 11-30 Register Assignment to Clock and Reset Domains**

Register	Kernel Clock	Bus Clock	Cfg Clock	RTC reset	Bus Reset
<b>RTC_ID</b>	-	-	-	-	-
<b>RTC_CON</b>	X	-	-	X	-
<b>RTC_T14_CNT</b>	X	-	-	X	-
<b>RTC_T14_REL</b>	-	-	X	X	-
<b>RTC_CNT_LO/ RTC_CNT_HI</b>	X	-	-	X	-
<b>RTC_REL_LO/ RTC_REL_HI</b>	-	-	X	X	-
<b>RTC_ISNC</b>	X	-	-	X	-
<b>RTC_CTRL</b>	-	X	-	X	-
<b>RTC_ALARM_LO/ RTC_ALARM_HI</b>	X	-	-	X	-

### 11.6.5.5.2 RTC Control

The operating behavior of the RTC module is controlled by the **RTC\_CON (on Page 1134)** register. The RTC starts counting when **RTC\_CON.RUN** is set. After a software reset the run bit is set and the RTC automatically starts operation (note that the RTC module is only reset by software, not by the power on reset). The bit

CONFIDENTIAL

RTC

**RTC\_CON.RTCPRE** selects a prescaler which divides the counting clock by factor 8. Activating the prescaler reduces the resolution of the reload counter T14. If the prescaler is not activated, the RTC may lose counting clocks on switching from asynchronous to synchronous mode and back. This effect can be avoided by activating the prescaler.

Setting the bits **RTC\_CON.T14DEC** or **RTC\_CON.T14INC** decrements or increments the T14 timer with the next count event. If at the next count event a reload has to be executed, then an increment operation is delayed until the next count event occurs. The in/decrement function can only be used if register T14REL is not equal to FFFF<sub>H</sub>. The in/decrement bits are cleared by hardware after the decrement/increment operation.

#### 11.6.5.5.3 System Clock Operation

A real time system clock can be maintained that represents the current time and date. If the RTC module is not effected by a system reset, it keeps running also during idle mode and power down mode.

The maximum resolution (minimum step width) for this clock information is determined by timer T14's input clock. The maximum usable time span is achieved when

**RTC\_T14\_REL.T14REL** is loaded with 0000<sub>H</sub> and so T14 is divided by  $2^{16}$ .

#### 11.6.5.5.4 Cyclic Interrupt Generation

The RTC module can generate an interrupt request RTC\_T14INT whenever timer T14 overflows and is reloaded. For example, this interrupt request may be used to provide a system time tick independent of the Kernel Clock frequency without loading the general purpose timers, or to wake up regularly from idle mode. The T14 overflow interrupt (RTC\_T14INT) cycle time can be adjusted via the timer T14 reload register

**RTC\_T14\_REL.T14REL**. This interrupt request is also OR'ed with all other interrupts of the RTC via the RTC interrupt sub node **RTC\_ISNC (on Page 1140)**.

#### 11.6.5.5.5 Alarm Interrupt Generation

The RTC module can also provide an alarm interrupt. The alarm time is written to the registers **RTC\_ALARM\_LO** and **RTC\_ALARM\_HI (on Page 1139)**. One **RTC\_CNT\_LO (on Page 1137)** cycle after the value in registers **RTC\_CNT\_HI** and **RTC\_CNT\_LO** equals the value programmed in **RTC\_ALARM\_HI** and **RTC\_ALARM\_LO**, an internal interrupt request is generated. It is OR'ed in the interrupt sub node register **RTC\_ISNC (on Page 1140)** with the other RTC interrupts to generate one interrupt request RTC\_INT.

#### 11.6.5.5.6 48-bit Timer Operation

The concatenation of the 16-bit reload timer T14 and the 32-bit RTC timer can be regarded as a 48-bit timer which counts with the RTC count input frequency, (RTC\_REF\_CLK) divided by the fixed prescaler, if the prescaler is selected. The reload

registers [RTC\\_T14\\_REL](#), [RTC\\_REL\\_LO](#), and [RTC\\_REL\\_HI](#) should be cleared to get the maximum usable time span of  $2^{48}$  ( $\approx 10^{14}$ ) T14 input clocks.

#### 11.6.5.5.7 Hardware Dependent RTC Accuracy

The RTC has different counting accuracies, depending on the operating mode (with or without prescaler). There is only an impact on the counting accuracy when switching the RTC from synchronous mode to asynchronous mode and back.

**Table 11-31 Impact on Counting Accuracy**

Operating Mode	Inaccuracy in T14 Counting Ticks
Without prescaler	+0 / -0.5
With prescaler	+0 / -0

#### 11.6.5.5.8 RTC Disable Functionality

The Peripheral Kernel of the RTC can be disabled, if the RTC functionality is not used. In this case only the bus interface, interrupt logic and pad tristate control is enabled. Disabling the RTC module reduces the power consumption and the generated noise of the complete system.

### 11.6.6 RTC Power Supply Concept

The RTC module logic and the internal 32k oscillator are supplied by VDD\_RTC. The VDD\_RTC operating range is specified in the chapter Operating Conditions. Generally there are two ranges:

1. RTC and oscillator operating range **for RTC stand-alone operation** (for example, mobile station off, only RTC is running).
2. RTC and oscillator operating range **for communication with MCU**.

The voltages are given in [Section 13.2.1.1.2 Voltages \(on Page 1303\)](#).

**CONFIDENTIAL**

**GPT 1 and 2**

## 11.7 GPT 1 and 2

History	
Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.01	
<b>Page 1145</b>	Updated <b>System Integration</b> : WS00006682
Changes for Rev. 1.04	
<b>Page 1189</b>	Removed "SRC Register" column from <b>Table 11-57 GPT1_2 Interrupt Sources</b> WS00008455
Changes for Rev. 1.06	

### System Integration:

- Supply domain: VDD\_MAIN
- Chip internal interfaces:
  - Clock domain: Refer to **Section 10.4.1.3 Sub-System Clocks and Enables (on Page 661)** and see **Figure 10-10 Clock Enable (on Page 662)**.
  - Bus domain: PD-Bus
- Interrupt sources:
- Monitor Pins: Refer to **Section 11.8.10 Internal Signal Monitoring (on Page 1209)**

### 11.7.1 Introduction

The General Purpose Timer Unit (GPT12) provides very flexible multifunctional timer structures that may be used for timing, event counting, pulse width measurement, pulse generation, frequency multiplication, and other purposes. The GPT12 incorporates five 16-bit timers grouped into the two timer blocks: Block1 (GPT1) and Block2 (GPT2). Each timer in each block can operate independently in a number of different modes such as Gated Timer Mode or Counter Mode; or, each timer can be concatenated with another timer of the same block.

Block 1 contains three timers/counters with a maximum resolution of  $f_{hw\_clk}/4$  (for information about  $hw\_clk$ , refer to **Table 11-46 GPT12 Register Summary (on Page 1176)**). The auxiliary Timers of GPT1 may optionally be configured as reload or capture registers for the core Timer.

Block 2 contains 2 timers/counters with a maximum resolution of  $f_{hw\_clk}/2$ . An additional **CAPREL register** supports capture and reload operation with extended functionality.

CONFIDENTIAL

GPT 1 and 2

*Note: The Core Timer T6 may be concatenated with timers of other on-chip peripherals such as a CAPCOM unit.*

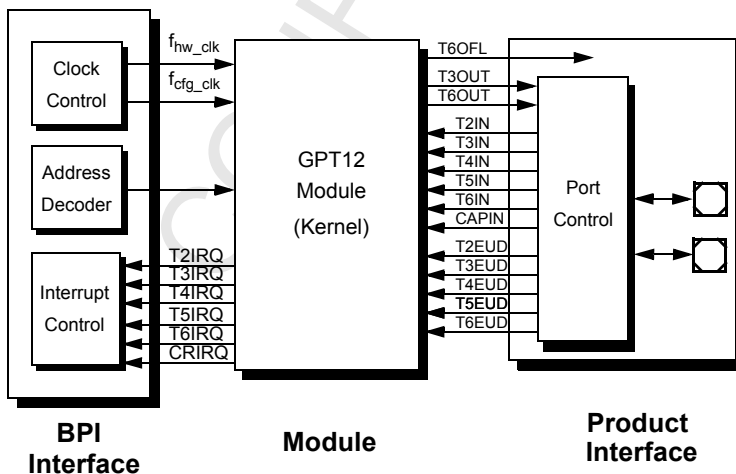
*Note: The GPT12 can receive two clocks:  $f_{hw\_clk}$  (the normal operation clock) and  $f_{cfg\_clk}$  (only used for the configuration registers).*

The following summary identifies all features to be supported by the GPT12:

- Timer Block 1:
  - Maximum resolution of  $f_{hw\_clk}/4$
  - Three independent timers/counters.
  - Concatenation of timers/counters can be done.
  - Four operating modes (Timer Mode, Gated Timer Mode, Counter Mode, Incremental Interface Mode).
  - Separate interrupt nodes.
- Timer Block 2:
  - Maximum resolution of  $f_{hw\_clk}/2$
  - Two independent timers/counters.
  - Concatenation of timers/counters can be done.
  - Three operating modes (Timer Mode, Gated Timer Mode, Counter Mode).
  - Extended capture/reload functions via 16-bit Capture/Reload register **CAPREL**.
  - Separate interrupt nodes.

## 11.7.2 Overview

Figure 11-52 GPT12 Interface Diagram



*Note: Bus Peripheral Interface (BPI) is the connection to the on-chip bus system.*

### **11.7.3 Kernel Description**

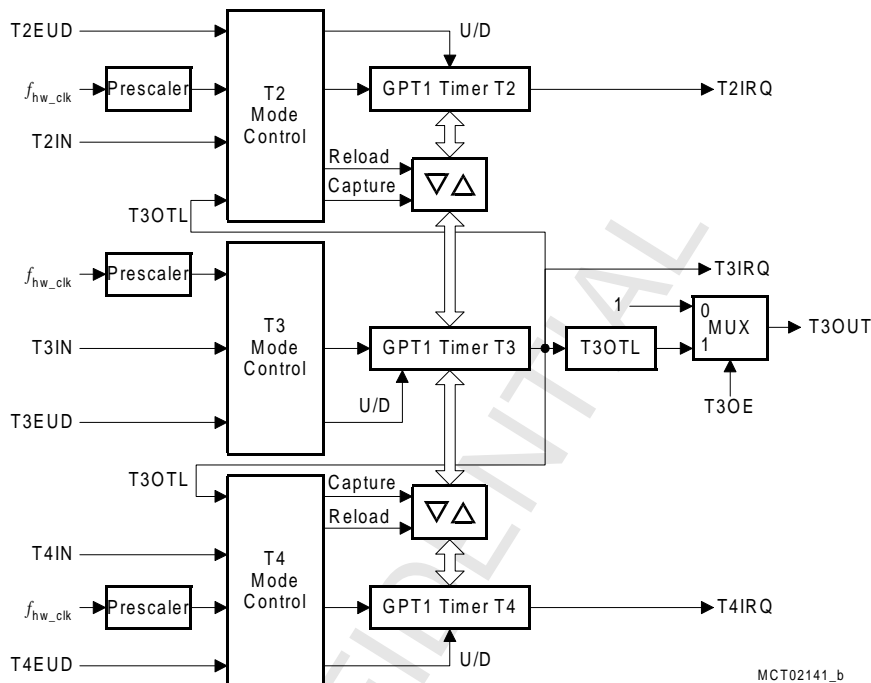
#### **11.7.3.1 Functional Description of Timer Block 1**

All three timers of Block 1 (T2, T3, T4) can run in 4 basic modes: Timer Mode, Gated Timer Mode, Counter Mode and Incremental Interface Mode. All timers can count up or down. Each timer of Block 1 is controlled by a separate control register **TxCON**.

Each timer has an input line, TxIN, associated with it which serves as the gate control in Gated Timer Mode, or as the count input in Counter Mode. The count direction (up/down) may be programmed via software or may be dynamically altered by a signal at an external control input line, External Up/Down Control Input TxEUD. An overflow/underflow of core Timer T3 is indicated by the Output Toggle Latch T3OTL whose state may be output on related line T3OUT. Additionally, the auxiliary Timers T2 and T4 may be concatenated with core Timer T3 or may be used as capture or reload registers for core Timer T3. Concatenation of T3 with other timers is provided through line T3OTL.

The current contents of each timer can be read or modified by the MCU by accessing the corresponding timer registers T2, T3, or T4, located in the non-bitaddressable Special Function Register (SFR) space. When any of the timer registers is written to by the MCU in the state immediately before a timer increment, decrement, reload, or capture is to be performed, the MCU write operation has priority in order to guarantee correct results.

Figure 11-53 Structure of Timer Block 1



### 11.7.3.1.1 Core Timer T3

The operation of core Timer T3 is controlled by its bitaddressable control register **T3CON**.

#### Run Control

The timer can be started or stopped by software through bit **T3CON.T3R**. Setting bit **T3CON.T3R** will start the timer; clearing **T3CON.T3R** stops the timer.

In Gated Timer Mode, the timer will run only if **T3CON.T3R** is set and the gate is active (high or low, as programmed).

*Note: When bit **T2CON.T2RC** or **T4CON.T4RC** is set, **T3CON.T3R** will also control (start and stop) auxiliary Timer T2/T4.*

#### Count Direction Control

The count direction of the core Timer T3 can be controlled either by software or by the external input line,  $T3EUD$ . These options are selected by bits **T3CON.T3UD** and



CONFIDENTIAL

GPT 1 and 2

**T3CON.T3UDE.** When the up/down control is set by software (bit **T3CON.T3UDE** is cleared), the count direction can be altered by setting or clearing bit **T3CON.T3UD**. When **T3CON.T3UDE** is set, line T3EUD is selected to be the controlling source of the count direction. However, bit **T3CON.T3UD** can still be used to reverse the actual count direction, as shown in **Table 11-32**. If **T3CON.T3UD** is cleared and line T3EUD shows a low level, the timer is counting up. With a high level at T3UD the timer is counting down. If **T3CON.T3UD** is set, a high level at line T3EUD specifies counting up, and a low level specifies counting down. The count direction can be changed whether or not the timer is running or not.

*Note: When line T3EUD is used as external count direction control input, its associated port pin must be configured as input.*

**Table 11-32 Core Timer T3 Count Direction Control**

Line T3EUD	Bit <b>T3CON.T3UDE</b>	Bit <b>T3CON.T3UD</b>	Count Direction
X	0	0	Counts Up
X	0	1	Counts Down
0	1	0	Counts Up
1	1	0	Counts Down
0	1	1	Counts Down
1	1	1	Counts Up

*Note: The direction control works in same way for core Timer T3 and for auxiliary Timers T2 and T4.*

### Timer 3 Overflow/Underflow Monitoring

An overflow or underflow of Timer T3 will clock bit **T3CON.T3OTL**. **T3CON.T3OTL** can also be set or reset by software. Bit **T3CON.T3OE** (overflow/underflow output enable) enables the state of **T3CON.T3OTL** to be monitored via an external line, T3OUT. If this line is linked to an external port pin (configured as output), T3OUT can be used to control external hardware.

Additionally, **T3CON.T3OTL** can be used in conjunction with auxiliary Timers T2 and T4. In this case **T3CON.T3OTL** serves as input for the counter function or as trigger source for the reload function of T2 and T4. T3OTL is internally connected for this functionality and it is not necessary to enable overflow/underflow output on T3OUT for this purpose.

### Timer 3 in Timer Mode

Timer Mode for the core Timer T3 is selected by setting bit field **T3CON.T3M** to 000<sub>B</sub>. A block diagram of T3 in Timer Mode is shown in **Figure 11-54**.

CONFIDENTIAL

GPT 1 and 2

In this mode, T3 is clocked with the module clock  $f_{hw\_clk}$  divided by a programmable prescaler block, controlled by bit field **T3CON.T3I** and **T3CON.T3BPS1**. The input frequency  $f_{T3}$  for Timer T3 and its resolution  $r_{T3}$  are scaled linearly with lower module clock frequencies, as can be seen from the following formula:

$$f_{T3} = \frac{f_{hw\_clk}}{BPS1 * 2^{<T3I>}} \quad r_{T3} [ms] = \frac{BPS1 * 2^{<T3I>}}{f_{hw\_clk} [MHz]}$$

Note: *<BPS1> represents the prescaler value of the prescaler part controlled by bit field **T3CON.T3BPS1**. For the values, see the bit description in register **T3CON**.*

**Table 11-33 Timer 3 Input Parameter Selection: Timer Mode and Gated Timer Mode**

(all bits in **T3CON**)

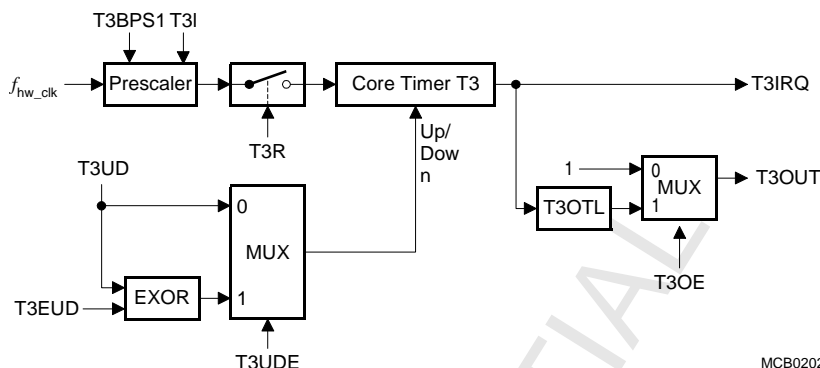
T3I	Prescaler for $f_{hw\_clk}$ (T3BPS1 = 00)	Prescaler for $f_{hw\_clk}$ (T3BPS1 = 01)	Prescaler for $f_{hw\_clk}$ (T3BPS1 = 10)	Prescaler for $f_{hw\_clk}$ (T3BPS1 = 11)
000	8	4	32	16
001	16	8	64	32
010	32	16	128	64
011	64	32	256	128
100	128	64	512	256
101	256	128	1024	512
110	512	256	2048	1024
111	1024	512	4096	2048

**Table 11-34 Example for Timer 3 Frequencies and Resolutions**

$f_{hw\_clk}$ [MHz]	<b>T3CON.T3I</b>	<b>T3CON.T3BPS1</b>	$f_{T3}$ [KHz]	$r_{T3}$ [μs]
40	7	10	9.77	102.4
40	0	01	10000	0.1
50	0	00	6250	0.16
50	4	11	195.31	5.12
50	7	10	12.20	81.97

This formula also applies to the Gated Timer Mode of T3 and to the auxiliary Timers T2 and T4 in Timer Mode and Gated Timer Mode.

Figure 11-54 Block Diagram of Core Timer T3 in Timer Mode



MCB02028\_d

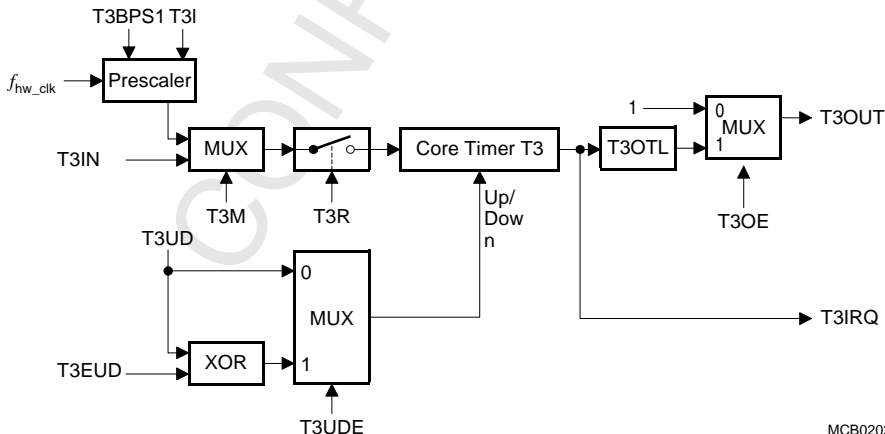
### Timer 3 in Gated Timer Mode

Gated Timer Mode for the core Timer T3 is selected by setting bit field **T3CON.T3M** to 010<sub>B</sub> or 011<sub>B</sub>.

Bit **T3CON.T3M(0)** (**T3CON(3)**) selects the active level of the gate input. The same options for the input frequency are available in Gated Timer Mode as for the Timer Mode. However, the input clock to the timer in this mode is gated by the external input line T3IN (Timer T3 External Input).

*Note: An associated port pin should be configured as input.*

Figure 11-55 Block Diagram of Core Timer T3 in Gated Timer Mode



MCB02029\_d

If **T3CON.T3M** = 010<sub>B</sub>, the timer is enabled when T3IN shows a low level. A high level at this line stops the timer. If **T3CON.T3M** = 011<sub>B</sub>, line T3IN must have a high level to

CONFIDENTIAL

GPT 1 and 2

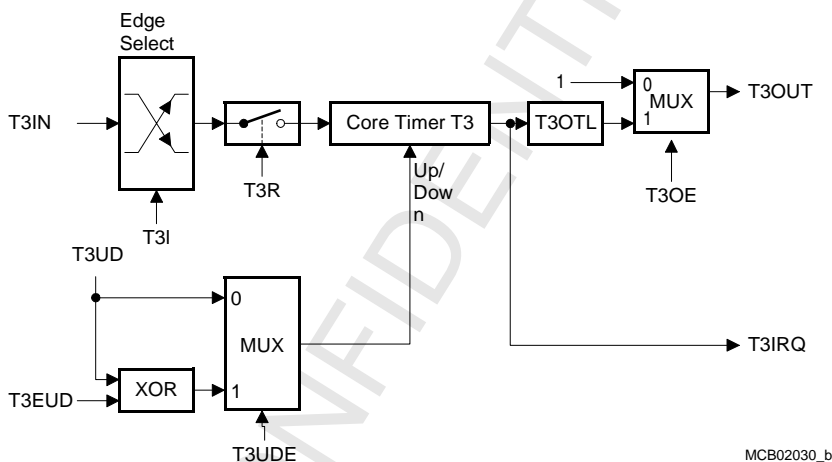
enable the timer. Additionally, the timer can be turned on or off by software using bit **T3CON.T3R**. The timer will run only if **T3CON.T3R** is set and the gate is active. It will stop if either T3R is cleared or the gate is inactive.

*Note: A transition of the gate signal at line T3IN does not cause an interrupt request.*

### Timer 3 in Counter Mode

Counter Mode for core Timer T3 is selected by setting bit field **T3CON.T3M** to 001<sub>B</sub>. In the Counter Mode, Timer T3 is clocked by a transition at the external input line T3IN. The event causing an increment or decrement of the timer can be a positive, a negative, or both a positive and a negative transition at this line. bit field **T3CON.T3I** selects the triggering transition (see [Table 11-35](#)).

**Figure 11-56 Block Diagram of Core Timer T3 in Counter Mode**



MCB02030\_b

**Table 11-35 Core Timer T3 (Counter Mode) Input Edge Selection**

<b>T3CON.T3I</b>	<b>Triggering Edge for Counter Increment / Decrement</b>
0 0 0	None. Counter T3 is disabled
0 0 1	Positive transition (rising edge) on T3IN
0 1 0	Negative transition (falling edge) on T3IN
0 1 1	Any transition (rising or falling edge) on T3IN
1 X X	Reserved. Do not use these combinations.

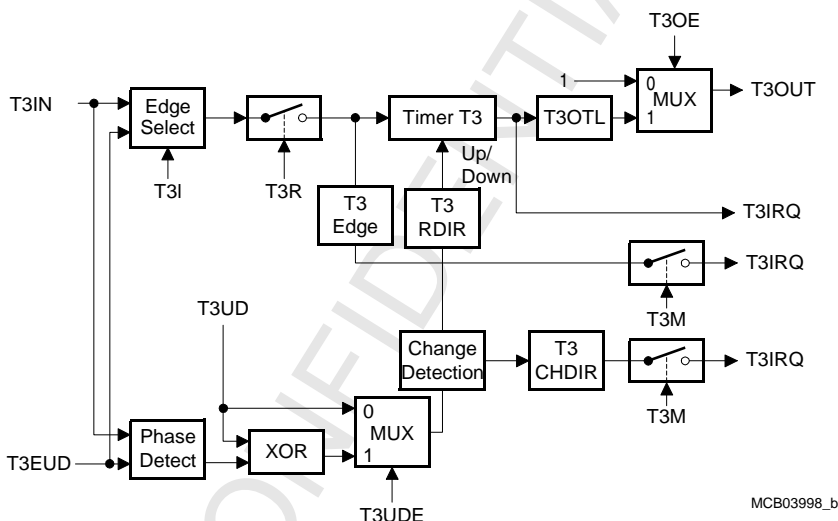
For Counter Mode operation, a port pin associated with line T3IN must be configured as input. The maximum input frequency allowed in Counter Mode is  $f_{\text{hw\_clk}}/8$

(**T3CON.T3BPS1** = 01). To ensure that a transition of the count input signal applied to T3IN is correctly recognized, its level should be held high or low for at least  $4 f_{hw\_clk}$  cycles (**T3CON.T3BPS1** = '01') before it changes.

### Timer 3 in Incremental Interface Mode

Incremental Interface Mode for the core Timer T3 is selected by setting bit field **T3CON.T3M** to 110<sub>B</sub> or 111<sub>B</sub>. In Incremental Interface Mode the two inputs associated with Timer T3 (T3IN, T3EUD) are used to interface to an external incremental encoder. T3 is clocked by each transition on one or both of the external input lines which gives 2-fold or 4-fold resolution of the encoder input.

**Figure 11-57 Block Diagram of Core Timer T3 in Incremental Interface Mode**



bit field **T3CON.T3I** selects the triggering transitions (see [Table 11-36](#)). The sequence of the transitions of the two input signals is evaluated and generates count pulses as well as the direction signal. Depending on whether Rotation Detection Mode (**T3CON.T3M** = 110<sub>B</sub>) or Edge Detection Mode (**T3CON.T3M** = 111<sub>B</sub>) is chosen, an interrupt request on T3IRQ is generated. For the Rotation Detection Mode, an interrupt is generated each time the count direction of Timer T3 changes. For the Edge Detection Mode an interrupt is generated each time a count action for Timer T3 occurs. Count direction, changes in the count direction and count requests are monitored by status bits **T3CON.T3RDIR**, **T3CON.T3CHDIR**, and **T3CON.T3EDGE**. T3 is modified automatically

CONFIDENTIAL

GPT 1 and 2

according to the speed and direction of the incremental encoder. Therefore, the contents of Timer T3 always represents the encoder's current position.

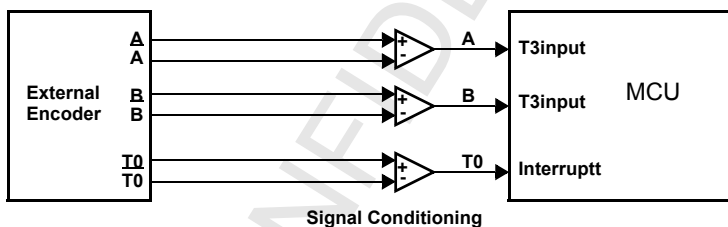
**Table 11-36 Core Timer T3 (Incremental Interface Mode) Input Edge Selection**

<b>T3CON.T3I</b>	<b>Triggering Edge for Counter Increment / Decrement</b>
0 0 0	None. Counter T3 stops.
0 0 1	Any transition (rising or falling edge) on T3IN.
0 1 0	Any transition (rising or falling edge) on T3EUD.
0 1 1	Any transition (rising or falling edge) on any T3 input (T3IN or T3EUD).
1 X X	Reserved. Do not use these combinations.

The incremental encoder can be connected directly to the microcontroller without external interface logic. In a standard system, however, comparators will be employed to convert the encoder's differential outputs (such as A,  $\bar{A}$ ) to digital signals (such as A in [Figure 11-58](#)). This greatly increases noise immunity.

*Note: The third encoder output T0, which indicates the mechanical zero position, may be connected to an external interrupt input and trigger a reset of Timer T3.*

**Figure 11-58 Interfacing the Encoder to the Microcontroller**



The following conditions must be met for Incremental Interface Mode operation:

- bit field **T3CON.T3M** must be 110<sub>B</sub> or 111<sub>B</sub>.
- Pins associated to lines T3IN and T3EUD must be configured as input.
- Bit **T3CON.T3UDE** must be set to enable external direction control.





The maximum input frequency allowed in Incremental Interface Mode is  $f_{\text{hw\_clk}}/8$  (**T3CON.T3BPS** = 01<sub>B</sub>). To ensure that a transition of any input signal is correctly recognized, its level should be held high or low for at least 4  $f_{\text{hw\_clk}}$  cycles (**T3CON.T3BPS** = 01<sub>B</sub>) before it changes.

CONFIDENTIAL

GPT 1 and 2

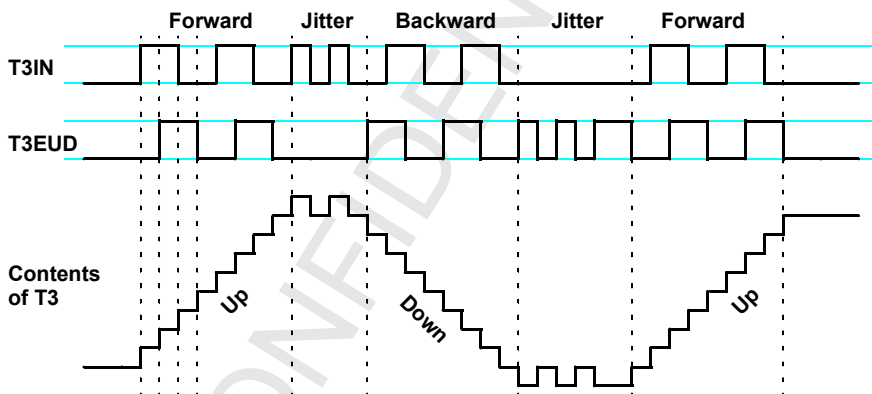
In Incremental Interface Mode the count direction is automatically derived from the sequence in which the input signals change, which corresponds to the rotation direction of the connected sensor. **Table 11-37** summarizes the possible combinations.

**Table 11-37 Core Timer T3 (Incremental Interface Mode) Count Direction**

Level on respective other input	T3IN Input		T3EUD Input	
	Rising 	Falling 	Rising 	Falling 
High	Down	Up	Up	Down
Low	Up	Down	Down	Up

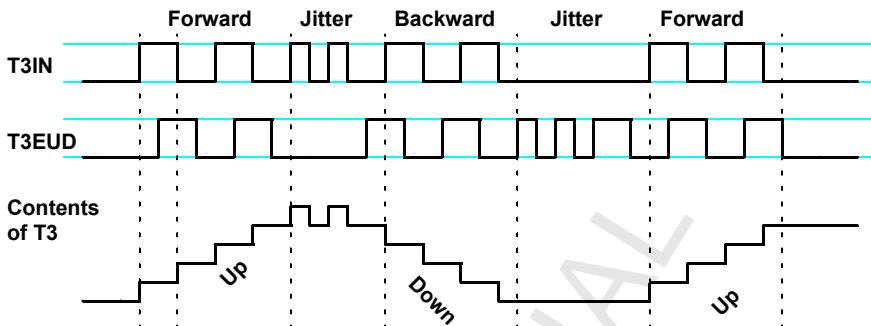
**Figure 11-59** and **Figure 11-60** give examples of the operation of T3 to illustrate count signal generation and direction control. Each example also shows how input jitter, which might occur if the sensor rests near to one of its switching points, is compensated.

**Figure 11-59 Evaluation of the Incremental Encoder Signals**



*Note: This example shows the timer behavior assuming that T3 counts upon any transition on any input, for example, **T3CON.T3I** = 011<sub>B</sub>.*

Figure 11-60 Evaluation of the Incremental Encoder Signals



Note: This example shows the timer behavior assuming that T3 counts upon any transition on input T3IN, for example, **T3CON.T3I** = 001<sub>B</sub>.

Note: Timer T3 operating in Incremental Interface Mode automatically provides information on the sensor's current position. Dynamic information (speed, acceleration, deceleration) may be obtained by measuring the incoming signal periods.

#### 11.7.3.1.2 Auxiliary Timers T2 and T4

Both auxiliary Timers T2 and T4 have exactly the same functionality. They can be configured for Timer Mode, Gated Timer Mode, Counter Mode, or Incremental Interface Mode with the same options for the timer frequencies and the count signal as the core Timer T3. In addition to these 4 counting modes, the auxiliary timers can be concatenated with the core Timer T3, or they may be used as reload or capture registers in conjunction with the core Timer T3.

The individual configuration for Timers T2 and T4 are determined by their bitaddressable control registers **T2CON** and **T4CON**, which are both organized identically. Note that functions which are present in all 3 timers of Timer Block 1 are controlled in the same bit positions and manner in each of the specific control registers.

Run control for auxiliary Timers T2 and T4 can be handled by the associated run control bit **T2CON.T2R** and **T4CON.T4R**. Alternatively, a remote control option (**T2CON.T2RC**, **T4CON.T4RC** set) may be enabled to start and stop T2/T4 via the run bit **T3CON.T3R** of core Timer T3.



**CONFIDENTIAL**

**GPT 1 and 2**

### Timers T2 and T4 in Timer Mode or Gated Timer Mode

When the auxiliary Timers T2 and T4 are programmed to Timer Mode or Gated Timer Mode, their operation is the same as described for the core Timer T3. The descriptions, figures and tables apply accordingly with two exceptions:

- There is no TxOUT output line for T2 and T4
- Overflow/underflow monitoring is not supported (no bit TxOTL in registers TxCON)

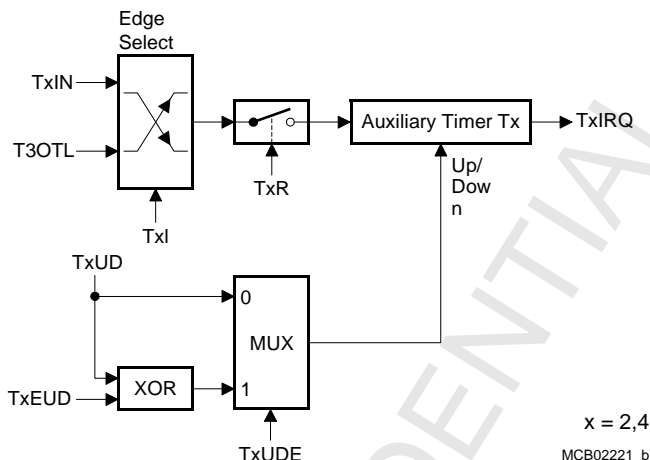
**Table 11-38 Timer 2,4 Input Parameter Selection: Timer Mode and Gated Timer Mode**

<b>T2CON. T2I or T4CON. T4I</b>	<b>Prescaler for <math>f_{hw\_clk}</math> (T3CON.T3BPS1 = 00)</b>	<b>Prescaler for <math>f_{hw\_clk}</math> (T3CON.T3BPS1 = 01)</b>	<b>Prescaler for <math>f_{hw\_clk}</math> (T3CON.T3BPS1 = 10)</b>	<b>Prescaler for <math>f_{hw\_clk}</math> (T3CON.T3BPS1 = 11)</b>
000	8	4	32	16
001	16	8	64	32
010	32	16	128	64
011	64	32	256	128
100	128	64	512	256
101	256	128	1024	512
110	512	256	2048	1024
111	1024	512	4096	2048

## Timers T2 and T4 in Counter Mode

In Counter Mode, Timers T2 and T4 can be clocked either by a transition at the respective external input line TxIN, or by a transition of T3OTL.

**Figure 11-61 Block Diagram of an Auxiliary Timer in Counter Mode**



The event causing an increment or decrement of a timer can be a positive, a negative, or both a positive and a negative transition at either the respective input line, or at the output toggle latch T3OTL.

bit fields **T2CON.T2I** and **T4CON.T4I** select the triggering transition (see [Table 11-39](#)).

**Table 11-39 Auxiliary Timer (Counter Mode) Input Edge Selection**

T2I/T4I	Triggering Edge for Counter Increment / Decrement
X 0 0	None. Counter T2 or T4 is disabled
0 0 1	Positive transition (rising edge) on T2IN or T4IN
0 1 0	Negative transition (falling edge) on T2IN or T4IN
0 1 1	Any transition (rising or falling edge) on T2IN or T4IN
1 0 1	Positive transition (rising edge) of T3OTL
1 1 0	Negative transition (falling edge) of T3OTL
1 1 1	Any transition (rising or falling edge) of T3OTL

*Note: Only state transitions of T3OTL caused by the overflow/underflow of T3 will trigger the counter function of T2/T4. Modifications of T3OTL via software will NOT trigger the counter function of T2/T4.*

For counter operation, an external pin associated to line TxIN must be configured as input. The maximum input frequency allowed in Counter Mode is  $f_{hw\_clk}/8$  (**T3CON.T3BPS1** = 01). To ensure that a transition of the count input signal which is applied to TxIN is correctly recognized, its level should be held for at least 4  $f_{hw\_clk}$  cycles (**T3CON.T3BPS1** = 01) before it changes.

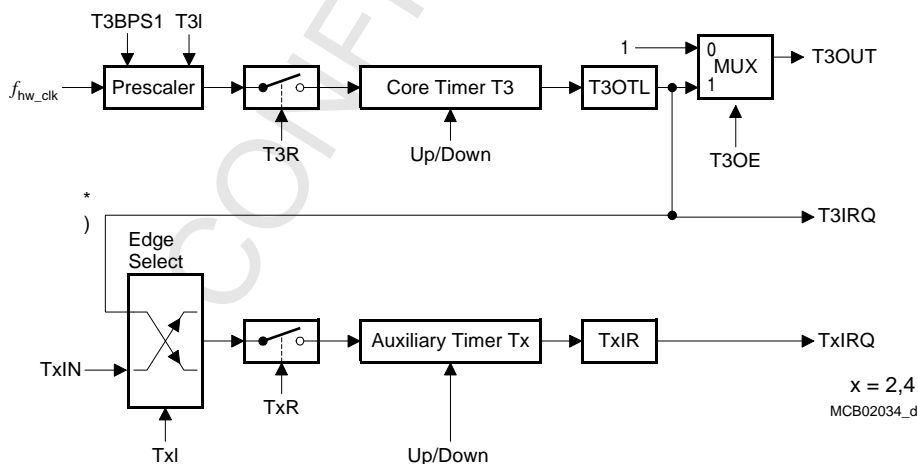
### 11.7.3.1.3 Timer Concatenation

Using T3OTL as a clock source for an auxiliary timer of Block 1 in Counter Mode concatenates the core Timer T3 with the respective auxiliary timer. Depending on which transition of T3OTL is selected to clock the auxiliary timer, this concatenation forms a 32-bit or a 33-bit timer/counter:

- **32-bit Timer/Counter:** If both a positive and a negative transition of T3OTL are used to clock the auxiliary Timer, this timer is clocked on every overflow/underflow of the core Timer T3. Thus, the two timers form a 32-bit timer.
- **33-bit Timer/Counter:** If either a positive or a negative transition of T3OTL is selected to clock the auxiliary Timer, this timer is clocked on every second overflow/underflow of the core Timer T3. This configuration forms a 33-bit timer (16-bit core Timer+T3OTL+16-bit auxiliary Timer).

The count directions is not required to be the same in the two concatenated timers. This offers a wide variety of different configurations. T3 can operate in Timer Mode, Gated Timer Mode or Counter Mode in this case.

**Figure 11-62 Concatenation of Core Timer T3 and an Auxiliary Timer**



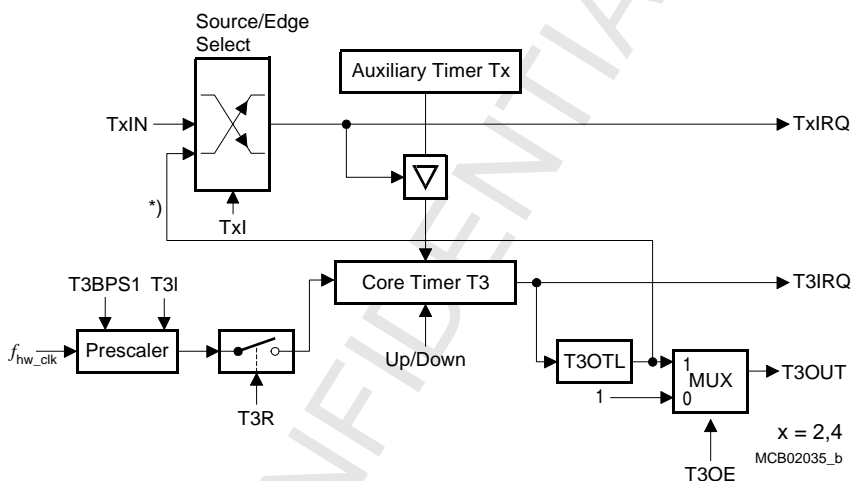
*Note: Line “\*” only affected by over/underflows of T3, but NOT by software modifications of T3OTL.*

## Auxiliary Timer in Reload Mode

Reload Mode for the auxiliary Timers T2 or T4 is selected by setting bit field **T2CON.T2I** or **T4CON.T4I** to 100<sub>B</sub>. In Reload Mode the core Timer T3 is reloaded with the contents of an auxiliary timer register, triggered by one of two different signals. The trigger signal is selected the same way as the clock source for Counter Mode (see [Table 11-39](#)). That is, a transition of the auxiliary Timer's input or the output toggle latch T3OTL may trigger the reload.

*Note: When programmed for Reload Mode, the respective auxiliary Timer T2 or T4 stops independent of its run flag T2R or T4R.*

**Figure 11-63 GPT1 Auxiliary Timer in Reload Mode**



*Note: Line '\*' only affected by over/underflows of T3, but NOT by software modifications of T3OTL.*

Upon a trigger signal, T3 is loaded with the contents of the respective timer register (T2 or T4) and T2IRQ or T4IRQ is set.

*Note: When a T3OTL transition is selected for the trigger signal, the interrupt request flag T3IRQ will be set upon a trigger, indicating T3's overflow or underflow.*

*Modifications of T3OTL via software will NOT trigger the counter function of T2/T4.*

The Reload Mode triggered by T3OTL can be used in a number of different configurations. Depending on the selected active transition, the following functions can be performed:

- If both a positive and a negative transition of T3OTL are selected to trigger a reload, the core Timer will be reloaded with the contents of the auxiliary Timer each time it

overflows or underflows. This is the standard Reload Mode (reload on overflow/underflow).

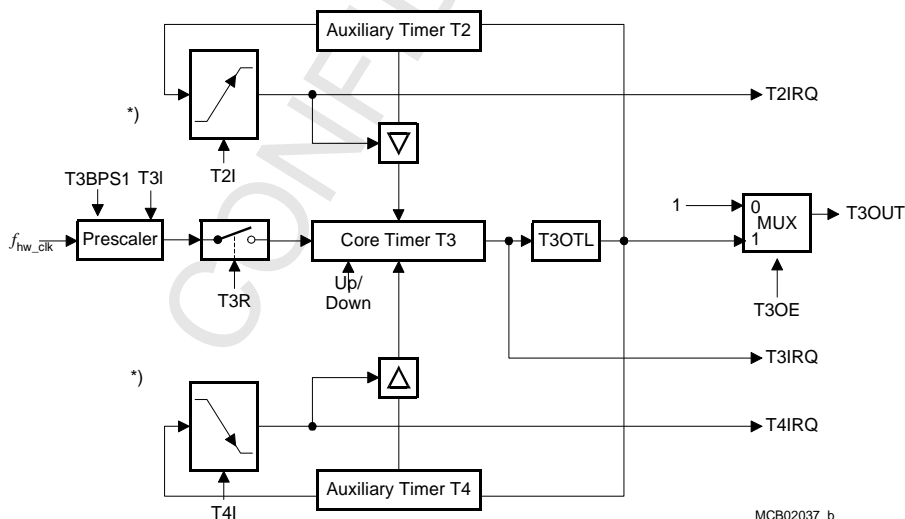
- If either a positive or a negative transition of T3OTL is selected to trigger a reload, the core Timer will be reloaded with the contents of the auxiliary Timer on every second overflow or underflow.
- Using this “single-transition” mode for both auxiliary Timers allows to perform very flexible pulse width modulation (PWM). One of the auxiliary Timers is programmed to reload the core Timer on a positive transition of T3OTL, the other is programmed for a reload on a negative transition of T3OTL. With this combination the core Timer is alternately reloaded from the two auxiliary Timers.

The **Figure 11-64** shows an example for the generation of a PWM signal using the alternate reload mechanism. T2 defines the high time of the PWM signal (reloaded on positive transitions) and T4 defines the low time of the PWM signal (reloaded on negative transitions). The PWM signal can be output on line T3OUT if the control bit **T3CON.T3OE** is set. With this method, the high and low time of the PWM signal can be varied over a wide range.

*Note: T3OTL is accessible via software and may be changed, if required, to modify the PWM signal. However, this will NOT trigger the reloading of T3.*

*Note: An associated port pin linked to line T3OUT should be configured as output.*

**Figure 11-64 GPT1 Timer Reload Configuration for PWM Generation**



*Note: Lines “\*” only affected by over/underflows of T3, but NOT by software modifications of T3OTL.*

*Note: It should be avoided to select the same reload trigger event for both auxiliary Timers. In this case both reload registers would try to load the core Timer at the same time. If this combination is selected, T2 is disregarded and the contents of T4 is reloaded.*

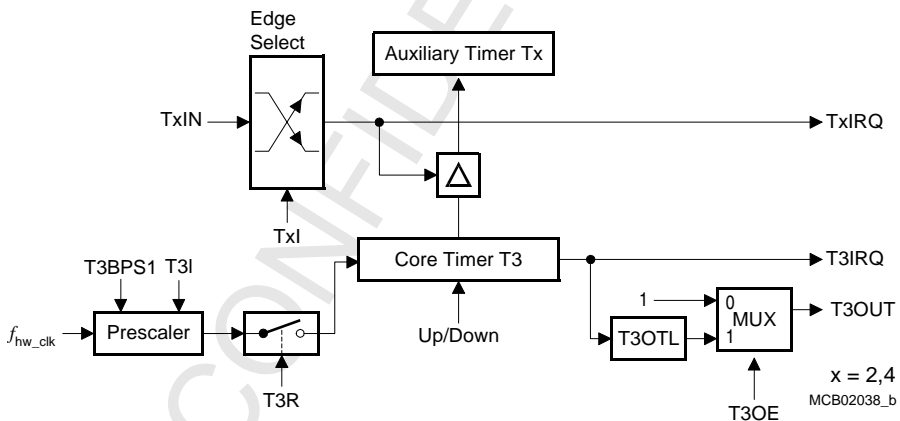
### Auxiliary Timer in Capture Mode

Capture Mode for the auxiliary Timers T2 or T4 is selected by setting bit field **T2CON.T2M** or **T4CON.T4M** to 101<sub>B</sub>. In Capture Mode the contents of the core Timer are latched into an auxiliary timer register in response to a signal transition at the respective auxiliary Timer's external input line TxIN. The capture trigger signal can be a positive, a negative, or both a positive and a negative transition.

The two least significant bits of bit field **T2CON.T2I** or **T4CON.T4I** are used to select the active transition (see [Table 11-39](#)), while the most significant bit **T2CON.T2I(2)** or **T4CON.T4I(2)** is irrelevant for Capture Mode. It is recommended to keep this bit cleared.

*Note: When programmed for Capture Mode, the respective auxiliary Timer (T2 or T4) stops independent of its run flag **T2CON.T2R** or **T4CON.T4R**.*

**Figure 11-65 Auxiliary Timer of Timer Block 1 in Capture Mode**



Upon a trigger (selected transition) at the corresponding input line TxIN, the contents of the core Timer are loaded into the auxiliary timer register and the associated interrupt request flag TxIRQ will be driven high.

*Note: Port pins associated with T2IN and T4IN must be configured to Input, and the level of the capture trigger signal should be held high or low for at least  $4 f_{clk}$  (**T3CON.T3BPS1** = 01) cycles before it changes to ensure correct edge detection.*

### Auxiliary in Incremental Interface Mode

When auxiliary Timers T2 and T4 are programmed to Incremental Interface Mode, their operation is the same as described for core Timer T3. The descriptions, figures, and tables apply accordingly with two exceptions:

- There is no TxOUT output line for T2 and T4.
- Overflow/underflow monitoring is not supported (no bit TxOTL).

**Table 11-40 Timer x Input Parameter Selection for Incremental Interface Mode**

<b>T2CON.T2I</b> <b>T4CON.T4I</b>	<b>Triggering Edge for Counter Update</b>
000	None. Counter Tx stops
001	Any transition (rising or falling edge) on TxIN
010	Any transition (rising or falling edge) on TxEUD
011	Any transition (rising or falling edge) on TxIN or TxEUD
1XX	<b>Reserved.</b> Do not use this combination!

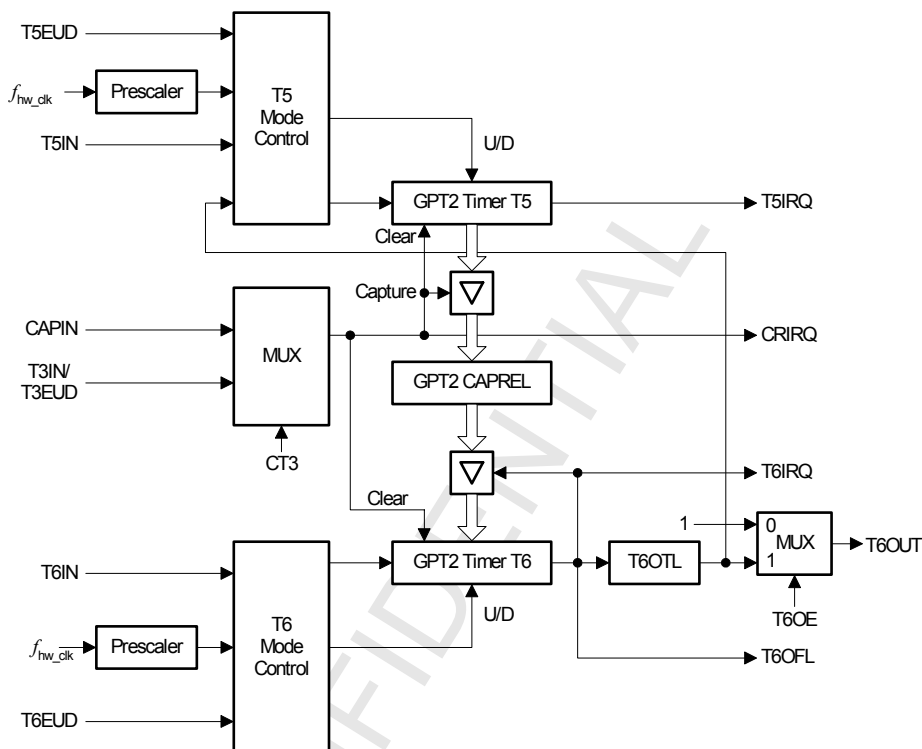
### 11.7.3.2 Functional Description of Timer Block 2

Timer Block 2 includes the two Timers T5 (referred to as the auxiliary Timer) and T6 (referred to as the core Timer), and the 16-bit capture/reload register CAPREL. Each Timer of Block 2 is controlled by a separate control register **T5CON** or **T6CON**.

Each timer has an input line (TxIN) associated with it which serves as the gate control in Gated Timer Mode, or as the count input in Counter Mode. The count direction (up/down) may be programmed via software or may be dynamically altered by a signal at an external control input line. An overflow/underflow of core Timer T6 is indicated by bit T6OTL whose state may be output on related line T6OUT and on line T6OFL. The core Timer T6 may be reloaded with the contents of CAPREL.

The toggle bit also supports the concatenation of T6 with auxiliary Timer T5, while concatenation of T6 with other timers is provided through line T6OFL. Triggered by an external signal, the contents of T5 can be captured into register CAPREL and T5 may optionally be cleared. Both Timer (T6 and T5) can count up or down, and the current timer value can be read or modified by the MCU in the non-bitaddressable SFRs T5 and T6.

Figure 11-66 Structure of Timer Block 2



### 11.7.3.2.1 Core Timer T6

The operation of the core Timer T6 is controlled by its bitaddressable control register **T6CON**.

#### Timer 6 Run Bit

The timer can be started or stopped by software through bit **T6CON.T6R** (Timer T6 Run Bit). Setting bit **T6CON.T6R** will start the timer; clearing **T6CON.T6R** stops the timer.

In Gated Timer Mode, the timer will only run if **T6CON.T6R** is set and the gate is active (high or low, as programmed).

*Note: When bit **T5CON.T5RC** is set bit **T6CON.T6R** will also control (start and stop) auxiliary Timer T5.*



CONFIDENTIAL

GPT 1 and 2

## Count Direction Control

The count direction of the core Timer can be controlled either by software or by the external up/down input line T6EUD). These options are selected by bits **T6CON.T6UD** and **T6CON.T6UDE**. When the up/down control is done by software (bit **T6CON.T6UDE** is cleared), the count direction can be altered by setting or clearing bit **T6CON.T6UD**. When **T6CON.T6UDE** is set, line T6EUD is selected to be the controlling source of the count direction. However, bit **T6CON.T6UD** can still be used to reverse the actual count direction, as shown in **Table 11-41**. If **T6CON.T6UD** is cleared and line T6EUD shows a low level, the timer is counting up. With a high level at T6EUD the timer is counting down. If **T6CON.T6UD** is set, a high level at line T6EUD specifies counting up, and a low level specifies counting down. The count direction can be changed regardless of whether the timer is running or not.

**Table 11-41 Core Timer T6 Count Direction Control (Tx = T5 or T6)**

TxEUD	TxUDE	TxUD	Count Direction
X	0	0	Count Up
X	0	1	Count Down
0	1	0	Count Up
1	1	0	Count Down
0	1	1	Count Down
1	1	1	Count Up

*Note: The direction control works the same for core Timer T6 and for auxiliary Timer T5.*

## Timer 6 Overflow/Underflow Monitoring

An overflow or underflow of Timer T6 will toggle **T6CON.T6OTL**. T6OTL can also be set or reset by software. Bit **T6CON.T6OE** enables the state of T6OTL to be monitored via the external output line T6OUT. An associated port pin must be configured as output.

Additionally, T6OTL can be used in conjunction with the timer over/underflow as an input for the counter function of auxiliary Timer T5. For this purpose, the state of T6OTL does not have to be available at line T6OUT, because an internal connection is provided for this option.

An overflow or underflow of Timer T6 can also be used to clock other timers. For this purpose, there is the special output line T6OFL.

## Timer 6 in Timer Mode

Timer Mode for the core Timer T6 is selected by setting bit field **T6CON.T6M** to 000<sub>B</sub>. In this mode, T6 is clocked with the module clock divided by a programmable prescaler, as selected by bit field T6I. The input frequency  $f_{T6}$  for Timer T6 and its resolution  $r_{T6}$  are

**CONFIDENTIAL**

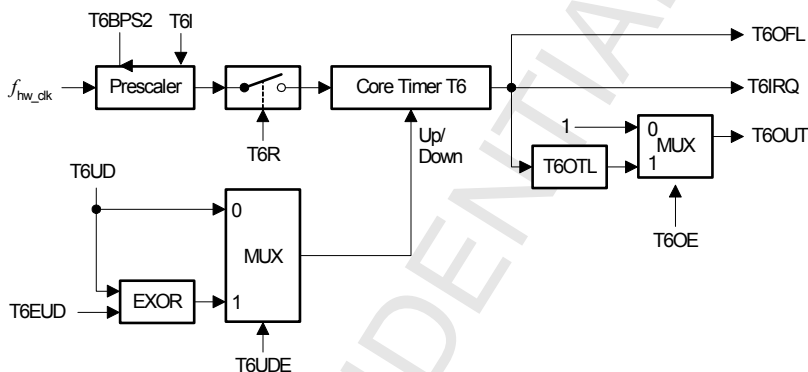
**GPT 1 and 2**

scaled linearly with lower clock frequencies  $f_{hw\_clk}$ , as can be seen from the following formula:

$$f_{T6} = \frac{f_{hw\_clk}}{BPS2 * 2^{<T6I>}} \quad r_{T6} [ms] = \frac{BPS2 * 2^{<T6I>}}{f_{hw\_clk} [MHz]}$$

*Note: <BPS2> represents the prescaler value of the prescaler part controlled by bit field **T6CON.BPS2**.*

**Figure 11-67 Block Diagram of Core Timer T6 in Timer Mode**



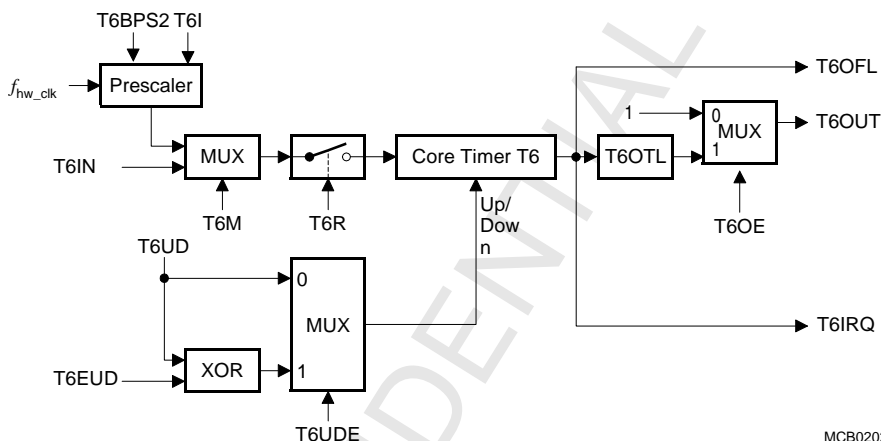
**Table 11-42 Timer 6 Input Parameter Selection: Timer Mode and Gated Timer Mode**

<b>T6CON.T6I</b>	Prescaler for $f_{hw\_clk}$ ( <b>T6CON.BPS2</b> = 00)	Prescaler for $f_{hw\_clk}$ ( <b>T6CON.BPS2</b> = 01)	Prescaler for $f_{hw\_clk}$ ( <b>T6CON.BPS2</b> = 10)	Prescaler for $f_{hw\_clk}$ ( <b>T6CON.BPS2</b> = 11)
000	4	2	16	8
001	8	4	32	16
010	16	8	64	32
011	32	16	128	64
100	64	32	256	128
101	128	64	512	256
110	256	128	1024	512
111	512	256	2048	1024

### Timer 6 in Gated Timer Mode

Gated Timer Mode for the core Timer T6 is selected by setting bit field **T6CON.T6M** to 010<sub>B</sub> or 011<sub>B</sub>. Bit **T6CON.T6M(0)** (**T6CON(3)**) selects the active level of the gate input. In Gated Timer Mode the same options for the input frequency as for the Timer Mode are available. However, the input clock to the timer in this mode is gated by the external input line T6IN.

**Figure 11-68 Block Diagram of Core Timer T6 in Gated Timer Mode**



MCB02029\_e

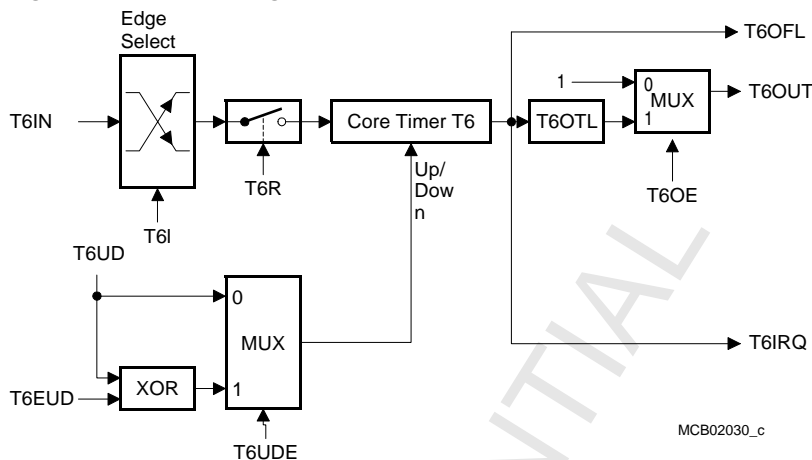
If **T6CON.T6M(0)** is cleared, the timer is enabled when T6IN shows a low level. A high level at this line stops the timer. If **T6CON.T6M(0)** is set, line T6IN must have a high level to enable the timer. Additionally, the timer can be turned on or off by software using bit T6R. The timer will only run, if **T6CON.T6R** is set and the gate is active. It will stop, if either **T6CON.T6R** is cleared or the gate is inactive.

*Note: A transition of the gate signal at line T6IN does not cause an interrupt request.*

### Timer 6 in Counter Mode

Counter Mode for the core Timer T6 is selected by setting bit field **T6CON.T6M** to 001<sub>B</sub>. In Counter Mode, Timer T6 is clocked by a transition at the external input line T6IN. The event causing an increment or decrement of the timer can be a positive, a negative, or both a positive and a negative transition at this line. bit field **T6CON.T6I** selects the triggering transition (see [Table 11-43](#)).

**Figure 11-69 Block Diagram of Core Timer T6 in Counter Mode**



**Table 11-43 Core Timer T6 (Counter Mode) Input Edge Selection**

<b>T6CON.T6I</b>	<b>Triggering Edge for Counter Increment/Decrement</b>
0 0 0	None. Counter T6 is disabled
0 0 1	Positive transition (rising edge) on T6IN
0 1 0	Negative transition (falling edge) on T6IN
0 1 1	Any transition (rising or falling edge) on T6IN
1 X X	Reserved. Do not use this combination

The maximum input frequency allowed in Counter Mode is  $f_{hw\_clk}/4$  (**T6CON.BPS2** = 01). To ensure that a transition of the count input signal which is applied to T6IN is correctly recognized, its level should be held high or low for at least 2  $f_{hw\_clk}$  cycles (**T6CON.BPS2** = 01) before it changes.

### 11.7.3.2.2 Auxiliary Timer T5

The auxiliary Timer T5 can be configured for Timer Mode, Gated Timer Mode, or Counter Mode with the same options for the timer frequencies and the count signal as core Timer T6. In addition to these 3 counting modes, the auxiliary Timer T5 can be concatenated with the core Timer T6.

The individual configuration for Timer T5 is determined by its bitaddressable control register **T5CON**. The functions present in both timers of Timer Block 2 are controlled in the same bit positions and in the same manner in each of the specific control registers.

CONFIDENTIAL

GPT 1 and 2

Run control for auxiliary Timer T5 can be handled by the associated Run Control Bit **T5CON.T5R**. Alternatively, a remote control option (**T5CON.T5RC** is set) may be enabled to start and stop T5 via the run bit **T5CON.T5R** of core Timer T6.

*Note: The auxiliary Timer has no T5OTL. Therefore, an output line for overflow/underflow monitoring is not provided.*

### Count Direction Control for Auxiliary Timer

The count direction of the auxiliary Timer can be controlled in the same way as for the core Timer T6. The description and the table apply accordingly.

### Timer T5 in Timer Mode or Gated Timer Mode

When auxiliary Timer T5 is programmed to Timer Mode or Gated Timer Mode, its operation is the same as described for the core Timer T6. The descriptions, figures and tables apply accordingly with two exceptions:

- There is no TxOUT line for T5.

Overflow/underflow monitoring is not supported (no bit T5OTL).

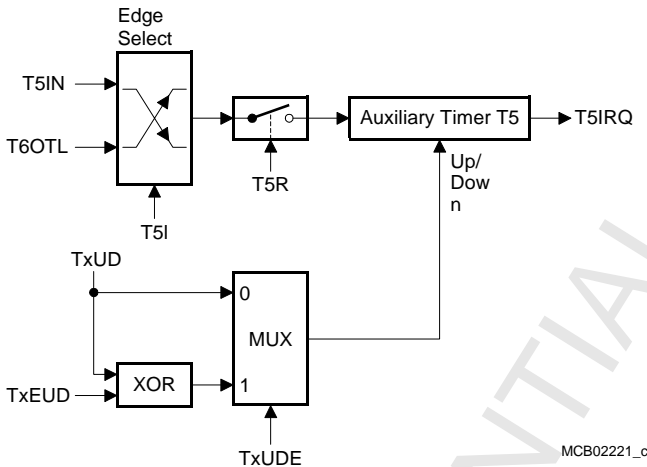
**Table 11-44 Timer 5 Input Parameter Selection: Timer Mode and Gated Timer Mode**

<b>T5CON.T5I</b>	<b>Prescaler for <math>f_{hw\_clk}</math> (<b>T6CON.BPS2</b> = 00)</b>	<b>Prescaler for <math>f_{hw\_clk}</math> (<b>T6CON.BPS2</b> = 01)</b>	<b>Prescaler for <math>f_{hw\_clk}</math> (<b>T6CON.BPS2</b> = 10)</b>	<b>Prescaler for <math>f_{hw\_clk}</math> (<b>T6CON.BPS2</b> = 11)</b>
000	4	2	16	8
001	8	4	32	16
010	16	8	64	32
011	32	16	128	64
100	64	32	256	128
101	128	64	512	256
110	256	128	1024	512
111	512	256	2048	1024

### Timer T5 in Counter Mode

Counter Mode for auxiliary Timer T5 is selected by setting bit field **T5CON.T5M** to 001<sub>B</sub>. In Counter Mode, Timer T5 can be clocked either by a transition at the external input line T5IN, or by a transition of Timer T6's output toggle latch T6OTL.

**Figure 11-70 Block Diagram of Auxiliary Timer T5 in Counter Mode**



The event causing an increment or decrement of the timer can be a positive, a negative, or both a positive and a negative transition at either the input line T5IN or at the toggle latch T6OTL.

bit field **T5CON.T5I** selects the triggering transition (see [Table 11-45](#)).

**Table 11-45 Auxiliary Timer (Counter Mode) Input Edge Selection**

<b>T5CON.T5I</b>	<b>Triggering Edge for Counter Increment/Decrement</b>
X 0 0	None. Counter T5 is disabled
0 0 1	Positive transition (rising edge) on T5IN
0 1 0	Negative transition (falling edge) on T5IN
0 1 1	Any transition (rising or falling edge) on T5IN
1 0 1	Positive transition (rising edge) of T6OTL
1 1 0	Negative transition (falling edge) of T6OTL
1 1 1	Any transition (rising or falling edge) of T6OTL

*Note: Only state transitions of T6OTL caused by the overflow/underflow of T6 will trigger the counter function of T5. Modifications of T6OTL via software will **NOT** trigger the counter function of T5.*

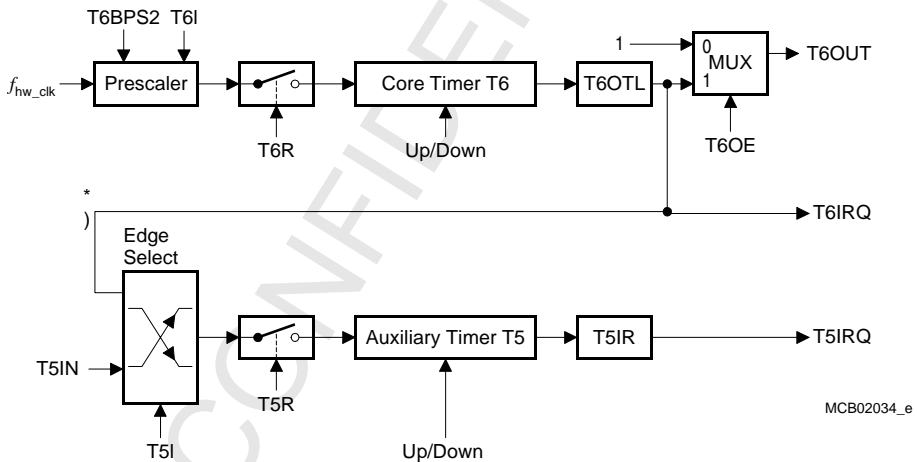
The maximum input frequency which is allowed in Counter Mode is  $f_{hw\_clk}/4$  (**T6CON.BPS2** = 01). To ensure that a transition of the count input signal which is applied to T5IN is correctly recognized, its level should be held high or low for at least 2  $f_{hw\_clk}$  cycles (**T6CON.BPS2** = 01) before it changes.

### 11.7.3.2.3 Timer Concatenation

Using the toggle bit **T6CON.T6OTL** as a clock source for the auxiliary Timer of Block 2 in Counter Mode concatenates core Timer T6 with auxiliary Timer T5. Depending on which transition of **T6CON.T6OTL** is selected to clock auxiliary Timer T5, this concatenation forms a 32-bit or a 33-bit timer/counter.

- 32-bit Timer/Counter: If both a positive and a negative transition of **T6CON.T6OTL** is used to clock the auxiliary Timer T5, this timer is clocked on every overflow/underflow of the core Timer T6. Thus, the two timers form a 32-bit timer.
- 33-bit Timer/Counter: If either a positive or a negative transition of **T6CON.T6OTL** is selected to clock the auxiliary Timer T5, this timer is clocked on every second overflow/underflow of the core Timer T6. This configuration forms a 33-bit timer (16-bit core Timer+**T6CON.T6OTL**+16-bit auxiliary Timer). The count directions of the two concatenated timers are not required to be the same. This offers a wide variety of configurations. T6 can operate in Timer Mode, Gated Timer Mode or Counter Mode in this case.

**Figure 11-71 Concatenation of Core Timer T6 and Auxiliary Timer T5**



MCB02034\_e

*Note: Line \*\* only affected by over/underflows of T6, but NOT by software modifications of **T6CON.T6OTL**.*

### Capture/Reload Register CAPREL in Capture Mode

This 16-bit register can be used as a capture register for auxiliary Timer T5. This mode is selected by setting bit **T5CON.T5SC**. Bit **T5CON.CT3** selects the external input line (CAPIN) or the input lines (T3IN and/or T3EUD) of Timer T3 as the source for a capture trigger. Either a positive, a negative, or both a positive and a negative transition at line

CONFIDENTIAL

GPT 1 and 2

CAPIN can be selected to trigger the capture function, or transitions on input T3IN or input T3EUD or both inputs T3IN and T3EUD. The active edge is controlled by bit field CI in register **T5CON**.

The maximum input frequency for the capture trigger signal at CAPIN is  $f_{hw\_clk}/2$  (**T6CON.BPS2** = 01). To ensure that a transition of the capture trigger signal is correctly recognized, its level should be held for at least 2  $f_{hw\_clk}$  cycles (**T6CON.BPS2** = 01) before it changes.

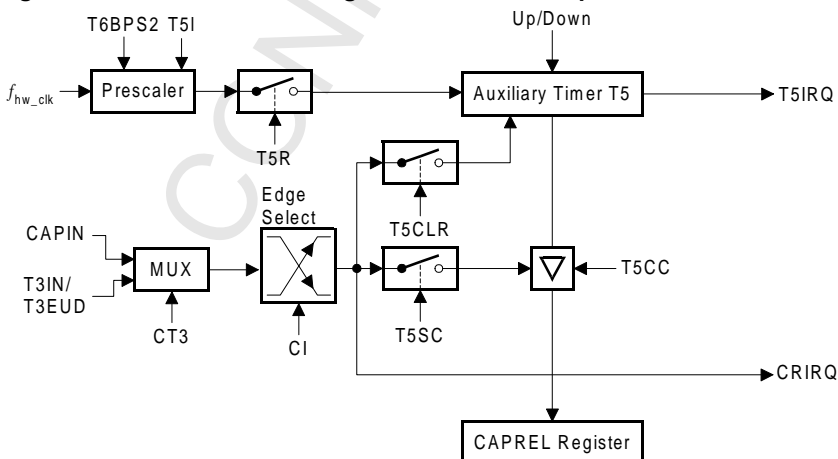
When the Timer T3 capture trigger is enabled (**T5CON.CT3** is set) register CAPREL captures the contents of T5 upon transitions of the selected input(s). These values can be used to measure input signals of T3. This is useful, for example, when T3 operates in Incremental Interface Mode, to derive dynamic information (speed acceleration) from the input signals.

When a selected transition at the external input line CAPIN is detected, the contents of auxiliary Timer T5 are latched into register CAPREL, and interrupt request line CRIRQ is driven at high level. With the same event, Timer T5 can be cleared to 0000<sub>H</sub>. This option is controlled by bit **T5CON.T5CLR**:

- If **T5CLR** is cleared, the contents of Timer T5 is not affected by a capture.
- If **T5CLR** is set, Timer T5 is cleared after the current timer value has been latched into register CAPREL.

*Note: Bit **T5CON.T5SC** only controls whether a capture is performed or not. If **T5CON.T5SC** is cleared the input line CAPIN can still be used to clear Timer T5 or as an external interrupt input. This interrupt is controlled by the CAPREL interrupt control register CRIC.*

Figure 11-72 Timer Block 2 Register CAPREL in Capture Mode



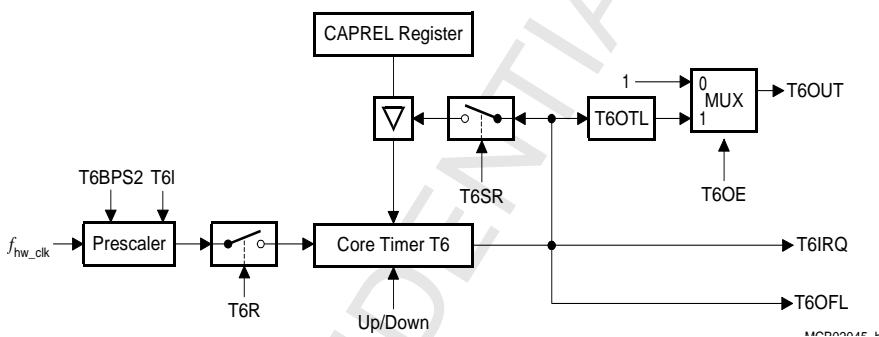


### Timer Block 2 Capture/Reload Register CAPREL in Reload Mode

This 16-bit register can be used as a reload register for core Timer T6. This mode is selected by setting bit **T6CON.T6SR**. The event causing a reload in this mode is an overflow or underflow of the core Timer T6.

When Timer T6 overflows from  $FFFF_H$  to  $0000_H$  (when counting up) or underflows from  $0000_H$  to  $FFFF_H$  (when counting down), the value stored in register CAPREL is loaded into Timer T6. This will not drive the interrupt request line CRIRQ associated with the CAPREL register. However, interrupt request line T6IRQ will be driven at high level to indicate the overflow/underflow of T6.

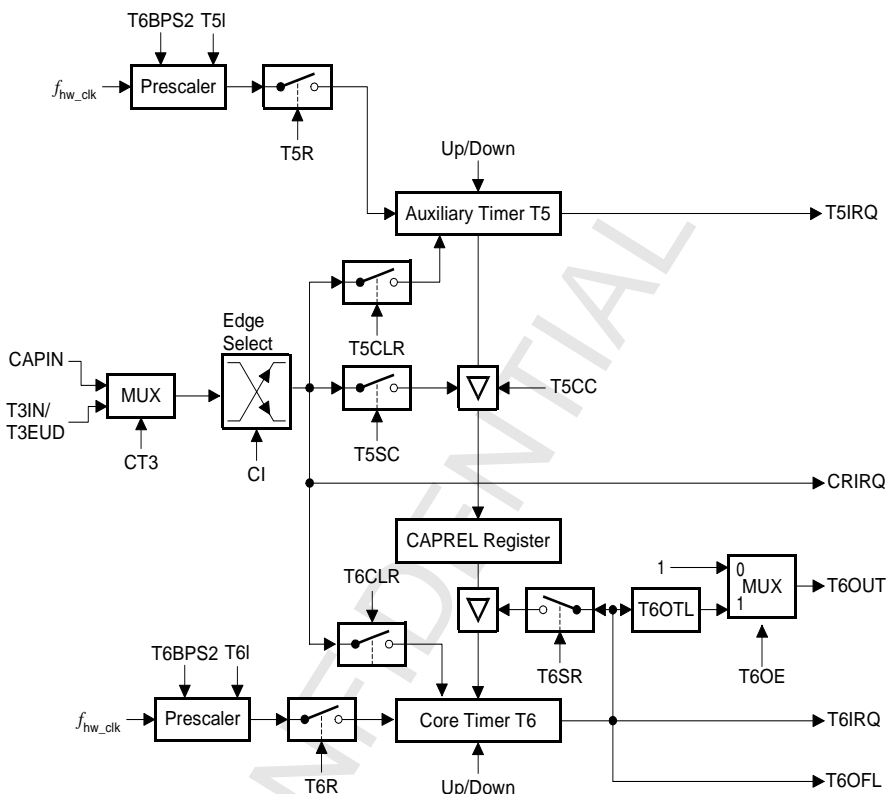
**Figure 11-73 Timer Block 2 Register CAPREL in Reload Mode**



### Timer Block 2 Capture/Reload Register CAPREL in Capture-And-Reload Mode

Since the reload and capture functions of register CAPREL can be enabled individually by bits **T5CON.T5SC** and **T6CON.T6SR**, the two functions can be enabled simultaneously by setting both bits. This feature can be used to generate an output frequency that is a multiple of the input frequency.

Figure 11-74 Timer Block 2 Register CAPREL in Capture-And-Reload Mode



This combined mode can be used to detect consecutive external events which may occur aperiodically, but where require a finer resolution (more 'ticks' within the time between two external events).

For this purpose, the time between the external events is measured using Timer T5 and the CAPREL register. Timer T5 runs in Timer Mode counting up with a frequency of  $f_{hw\_clk}/32$ , for example. The external events are applied to line CAPIN. When an external event occurs, the Timer T5 contents are latched into register CAPREL, and Timer T5 is cleared (**T5CON.T5CLR** cleared). Thus, register CAPREL always contains the correct time between two events, measured in Timer T5 increments. Timer T6, which runs in Timer Mode counting down with a frequency of  $f_{hw\_clk}/4$ , for example, uses the value in register CAPREL to perform a reload on underflow. This means, the value in register CAPREL represents the time between two underflows of Timer T6, now measured in Timer T6 increments. Since Timer T6 runs eight times faster than Timer T5, it will

**CONFIDENTIAL****GPT 1 and 2**

underflow eight times within the time between two external events. Thus, the underflow signal of Timer T6 generates eight 'ticks'. Upon each underflow, the interrupt request line T6IRQ will be driven at high level and bit **T6CON.T6OTL** will be toggled. The state of **T6CON.T6OTL** may be output on line T6OUT. This signal has eight times more transitions than the signal which is applied to line CAPIN.

A certain deviation of the output frequency is generated by the fact that Timer T5 will count actual time units (e.g. T5 running at 1 MHz will capture the value  $64_H/100_D$  for a 10 KHz input signal) while **T6CON.T6OTL** will only toggle upon an underflow of T6 (that is, the transition from  $0000_H$  to  $FFFF_H$ ). In the above mentioned example T6 would count down from  $64_H$  so the underflow would occur after 101 T6 timing ticks. The actual output frequency then is 79.2 KHz instead of the expected 80 KHz.

This can be solved by activating the Capture Correction (T5CC is set). If capture correction is active the content of T5 is decremented by 1 before being captured. The described deviation is eliminated (in the example, T5 would now capture  $63_H/99_D$  and the output frequency would be 80 KHz).

*Note: The underflow signal of Timer T6 can furthermore be used to clock one or more of the timers of the CAPCOM units. This makes it possible to set compare events based on a finer resolution than that of the external events. This connection is accomplished via signal T6OFL.*

**CONFIDENTIAL**

**GPT 1 and 2**

### 11.7.3.3 GPT12 Kernel Registers

All available kernel registers are summarized in the overview table below. .

**Table 11-46 GPT12 Register Summary**

Name	Clock	Access Condition	Description
<b>GPTID</b>	cfg_clk <sup>1)</sup>	None	Identification Register
<b>T2CON</b>	hw_clk <sup>1)</sup>	bitaddressable	Timer 2 Control Register
<b>T3CON</b>	hw_clk <sup>1)</sup>	bitaddressable	Timer 3 Control Register
<b>T4CON</b>	hw_clk <sup>1)</sup>	bitaddressable	Timer 4 Control Register
<b>T5CON</b>	cfg_clk <sup>1)</sup>	bitaddressable	Timer 5 Control Register
<b>T6CON</b>	hw_clk <sup>1)</sup>	bitaddressable	Timer 6 Control Register
CAPREL	hw_clk <sup>1)</sup>	None	Capture/Reload Register
T2	hw_clk <sup>1)</sup>	None	Timer 2 Register
T3	hw_clk <sup>1)</sup>	None	Timer 3 Register
T4	hw_clk <sup>1)</sup>	None	Timer 4 Register
T5	hw_clk <sup>1)</sup>	None	Timer 5 Register
T6	hw_clk <sup>1)</sup>	None	Timer 6 Register
<b>GPTISEL</b>	cfg_clk <sup>1)</sup>	None	Port Input Select Register

<sup>1)</sup> Refer to Clock Domain in [System Integration: \(on Page 1145\)](#)

#### 11.7.3.3.1 GPT Identification Register

**GPTID**

**GPT Identification Register**

**[Reset value: 5803<sub>H</sub>]**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>Module_ID</b>								<b>Revision_Number</b>							

Field	Bits	Type	Description
<b>Revision_Number</b>	0:7	r	<b>GPT Revision Number</b> These hard-wired bits are used for module revision numbering.
<b>Module_ID</b>	8:15	r	<b>GPT Identification Number</b> These hard-wired bits are used for module identification numbering.

**CONFIDENTIAL**

**GPT 1 and 2**

### 11.7.3.3.2 GPT12 Port Input Selection Register

Normally the **GPTPISEL** register switches between different port input sources the GPT12 unit via an input multiplexer.

But in the E-GOLDradio both values (0 or 1) for the **GPTPISEL.TxINIS** and **GPTPISEL.TxEUDIS** bits are tied to the same pin, therefore, the values chosen do not matter.

However, the **GPTPISEL.CAPINIS** bit is different. Because 0 selects a chip input pin, this register should be left at its reset value.

#### **GPTPISEL**

#### **Input Select Register**

**Reset Value 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>					<b>T6 EUD IS</b>	<b>T5 EUD IS</b>	<b>CAP IN IS</b>	<b>T6 IN IS</b>	<b>T5 IN IS</b>	<b>T4 EUD IS</b>	<b>T3 EUD IS</b>	<b>T2 EUD IS</b>	<b>T4 IN IS</b>	<b>T3 IN IS</b>	<b>T2 IN IS</b>

<b>Field</b>	<b>Bits</b>	<b>Type</b>	<b>Description</b>
<b>TxINIS</b> (x = 2,3,4,5,6)	0, 1, 2, 6, 7	rw	<b>Select Source for Timer Input</b> 0 Pin TxIN selected 1 Pin TxIN selected
<b>TxEUDIS</b> (x = 2,3,4,5,6)	3, 4, 5, 9, 10	rw	<b>Select Source for Timer External Up/Down</b> 0 Pin TxEUD selected 1 Pin TxEUD selected
<b>CAPINIS</b>	8	rw	<b>Select Source for Timer Capture Input</b> 0 Pin CAPIN selected 1 No input, tied to logical 0
<b>RESERVED</b>	15:11	r	Reserved, these bits must be left at their reset values.

CONFIDENTIAL

GPT 1 and 2

### 11.7.3.4 Function Control Registers

#### TxCON

The operating mode of the core Timer T3 is configured and controlled via its bitaddressable control register **T3CON**.

#### 11.7.3.4.1 Timer 3 Control Register

##### T3CON

##### Timer 3 Control Register

Reset Value 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T3 RDIR	T3 CH DIR	T3 EDG E	T3 BPS1	T3 OTL	T3 OE	T3 UDE	T3 UD	T3R	T3M			T3I			

Field	Bits	Type	Description																
T3I	2:0	rw	<b>Timer 3 Input Parameter Selection</b> <ul style="list-style-type: none"><li>• Timer Mode, see <a href="#">Table 11-47 on page 1180</a> for encoding</li><li>• Gated Timer Mode, see <a href="#">Table 11-47 on page 1180</a> for encoding</li><li>• Counter Mode, see Table 11-48 on page 1180 for encoding</li><li>• Incremental Interface Mode, see Table 11-49 on page 1180 for encoding</li></ul>																
T3M	5:3	rw	<b>Timer 3 Mode Control</b> <table><tr><td>000</td><td>Timer Mode</td></tr><tr><td>001</td><td>Counter Mode</td></tr><tr><td>010</td><td>Gated Timer Mode with Gate active low</td></tr><tr><td>011</td><td>Gated Timer Mode with Gate active high</td></tr><tr><td>100</td><td><b>Reserved.</b> Do not use this combination.</td></tr><tr><td>101</td><td><b>Reserved.</b> Do not use this combination.</td></tr><tr><td>110</td><td>Incremental Interface Mode (Rotation detection)</td></tr><tr><td>111</td><td>Incremental Interface Mode (Edge detection)</td></tr></table>	000	Timer Mode	001	Counter Mode	010	Gated Timer Mode with Gate active low	011	Gated Timer Mode with Gate active high	100	<b>Reserved.</b> Do not use this combination.	101	<b>Reserved.</b> Do not use this combination.	110	Incremental Interface Mode (Rotation detection)	111	Incremental Interface Mode (Edge detection)
000	Timer Mode																		
001	Counter Mode																		
010	Gated Timer Mode with Gate active low																		
011	Gated Timer Mode with Gate active high																		
100	<b>Reserved.</b> Do not use this combination.																		
101	<b>Reserved.</b> Do not use this combination.																		
110	Incremental Interface Mode (Rotation detection)																		
111	Incremental Interface Mode (Edge detection)																		
T3R	6	rw	<b>Timer 3 Run Bit</b> <table><tr><td>0</td><td>Timer/counter 3 stops</td></tr><tr><td>1</td><td>Timer/counter 3 runs</td></tr></table>	0	Timer/counter 3 stops	1	Timer/counter 3 runs												
0	Timer/counter 3 stops																		
1	Timer/counter 3 runs																		

**CONFIDENTIAL**

**GPT 1 and 2**

Field	Bits	Type	Description
<b>T3UD</b>	7	rw	<b>Timer 3 Up/Down Control</b> (when T3UDE is cleared) 0 Counting up 1 Counting down
<b>T3UDE</b>	8	rw	<b>Timer 3 External Up/Down Enable</b> 0 Counting direction is internally controlled by SW 1 Counting direction is externally controlled by line T3EUD
<b>T3OE</b>	9	rw	<b>Overflow/Underflow Output Enable</b> 0 T3 overflow/underflow can not be externally monitored 1 T3 overflow/underflow may be externally monitored via T3OUT
<b>T3OTL</b>	10	rwh	<b>Timer 3 Output Toggle Latch</b> Toggles on each overflow/underflow of T3. Can be set or reset by software.
<b>BPS1</b>	12:11	rw	<b>Timer Block Prescaler 1</b> The maximum input frequency 00 For Timer 2/3/4 is $f_{hw\_clk} / 8$ 01 For Timer 2/3/4 is $f_{hw\_clk} / 4$ 10 For Timer 2/3/4 is $f_{hw\_clk} / 32$ 11 For Timer 2/3/4 is $f_{hw\_clk} / 16$
<b>T3EDGE</b>	13	rwh	<b>Timer 3 Edge Detection</b> The bit is set on each successful edge detection. The bit has to be reset by SW. 0 No count edge was detected 1 A count edge was detected
<b>T3CHDIR</b>	14	rwh	<b>Timer 3 Count Direction Change</b> The bit is set on a change of the count direction of timer 3. The bit has to be reset by SW. 0 No change in count direction was detected 1 A change in count direction was detected
<b>T3RDIR</b>	15	rh	<b>Timer 3 Rotation Direction</b> 0 Timer 3 counts up 1 Timer 3 counts down

**Table 11-47 Timer 3 Input Parameter Selection for Timer Mode and Gated Timer Mode**

T3I	Prescaler for $f_{hw\_clk}$ (T3CON.BPS1 = 00)	Prescaler for $f_{hw\_clk}$ (T3CON.BPS1 = 01)	Prescaler for $f_{hw\_clk}$ (T3CON.BPS1 = 10)	Prescaler for $f_{hw\_clk}$ (T3CON.BPS1 = 11)
000	8	4	32	16
001	16	8	64	32
010	32	16	128	64
011	64	32	256	128
100	128	64	512	256
101	256	128	1024	512
110	512	256	2048	1024
111	1024	512	4096	2048

**Table 11-48 Timer 3 Input Parameter Selection for Counter Mode**

T3CON.T3I	Triggering Edge for Counter Update
000	None. Counter T3 is disabled
001	Positive transition ( raising edge ) on T3IN
010	Negative transition ( falling edge ) on T3IN
011	Any transition ( raising or falling edge ) on T3IN
1XX	<b>Reserved.</b> Do not use this combination!

**Table 11-49 Timer 3 Input Parameter Selection for Incremental Interface Mode**

T3CON.T3I	Triggering Edge for Counter Update
000	None. Counter T3 stops
001	Any transition ( raising or falling edge ) on T3IN
010	Any transition ( raising or falling edge ) on T3EUD
011	Any transition ( raising or falling edge ) on T3IN or T3EUD
1XX	<b>Reserved.</b> Do not use this combination!



**CONFIDENTIAL**

**GPT 1 and 2**

### 11.7.3.4.2 Timers 2 & 4 Control Register

**T2CON**

**T4CON**

**Timer 2/4 Control Register**

**Reset Value 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Tx RDIR	Tx CH DIR	Tx EDG E	Tx IR DIS	RESERV ED		Tx RC	Tx UDE	Tx UD	TxR	TxM			TxI		

Field	Bits	Type	Description
<b>TxI</b>	2:0	rw	<b>Timer x Input Parameter Selection</b> <ul style="list-style-type: none"><li>• Timer Mode, refer to Table 11-50 on page 1183 for encoding</li><li>• Gated Timer Mode, refer to Table 11-50 on page 1183 for encoding</li><li>• Counter Mode, refer to Table 11-51 on page 1183 for encoding</li><li>• Incremental Interface Mode, refer to Table 11-52 on page 1183 for encoding</li></ul>
<b>TxM</b>	5:3	rw	<b>Timer x Mode Control</b> (Basic Operating Mode) 000 Timer Mode 001 Counter Mode 010 Gated Timer Mode with Gate active low 011 Gated Timer Mode with Gate active high 100 Reload Mode 101 Capture Mode 110 Incremental Interface Mode (Rotation detection) 111 Incremental Interface Mode (Edge detection)
<b>TxR</b>	6	rw	<b>Timer x Run Bit</b> 0 Timer/counter x stops 1 Timer/counter x runs
<b>TxUD</b>	7	rw	<b>Timer x Up/Down Control</b> (when TxUDE is cleared) 0 Counting up 1 Counting down

**CONFIDENTIAL**

**GPT 1 and 2**

Field	Bits	Type	Description
<b>TxUDE</b>	8	rw	<b>Timer x External Up/Down Enable</b> 0 Counting direction is internally controlled by SW 1 Counting direction is externally controlled by line TxEUD
<b>TxRC</b>	9	rw	<b>Timer x Remote Control</b> 0 Timer/counter x is controlled by its own run bit TxR 1 Timer/counter x is controlled by the run bit of core Timer 3
<b>TxIRDIS</b>	12	rw	<b>Timer x Interrupt Disable</b> 0 Interrupt generation for TxCHDIR and TxEDGE interrupts in Incremental Interface Mode is enabled 1 Interrupt generation for TxCHDIR and TxEDGE interrupts in Incremental Interface Mode is disabled
<b>TxEDGE</b>	13	rwh	<b>Timer x Edge Detection</b> The bit is set on each successful edge detection. The bit has to be reset by SW. 0 No count edge was detected 1 A count edge was detected
<b>TxCHDIR</b>	14	rwh	<b>Timer x Count Direction Change</b> The bit is set on a change of the count direction of timer x. The bit has to be reset by SW. 0 No change in count direction was detected 1 A change in count direction was detected
<b>TxRDIR</b>	15	rh	<b>Timer x Rotation Direction</b> 0 Timer x counts up 1 Timer x counts down
<b>RESERVED</b>	11:10	r	Reserved, these bits must be left at their reset values.

**Table 11-50 Timer 2,4 Input Parameter Selection for Timer Mode and Gated Timer Mode**

T2CON. T2I or T4CON. T4I	Prescaler for $f_{hw\_clk}$ (T3CON.BPS1 = 00)	Prescaler for $f_{hw\_clk}$ (T3CON.BPS1 = 01)	Prescaler for $f_{hw\_clk}$ (T3CON.BPS1 = 10)	Prescaler for $f_{hw\_clk}$ (T3CON.BPS1 = 11)
000	8	4	32	16
001	16	8	64	32
010	32	16	128	64
011	64	32	256	128
100	128	64	512	256
101	256	128	1024	512
110	512	256	2048	1024
111	1024	512	4096	2048

**Table 11-51 Timer 2,4 Input Parameter Selection for Counter Mode**

T[2,4]I	Triggering Edge for Counter Update
X00	None. Counter Tx is disabled
001	Positive transition (raising edge) on TxIN
010	Negative transition (falling edge) on TxIN
011	Any transition (raising or falling edge) on TxIN
101	Positive transition (rising edge) of T3OTL
110	Negative transition (falling edge) of T3OTL
111	Any transition (rising or falling edge) of T3OTL

**Table 11-52 Timer 2,4 Input Parameter Selection for Incremental Interface Mode**

T[2,4]I	Triggering Edge for Counter Update
000	None. Counter Tx stops
001	Any transition (raising or falling edge) on TxIN
010	Any transition (raising or falling edge) on TxEUD
011	Any transition (raising or falling edge) on TxIN or TxEUD
1XX	<b>Reserved.</b> Do not use this combination!

**CONFIDENTIAL**

**GPT 1 and 2**

### 11.7.3.4.3 Timer 6 Control Register

**T6CON**

**Timer 6 Control Register**

**Reset Value 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T6 SR	T6 CLR	RES ERV ED	BPS2		T6 OTL	T6 OE	T6 UDE	T6 UD	T6R	T6M			T6I		

Field	Bits	Type	Description
T6I	2:0	rw	<b>Timer 6 Input Parameter Selection</b> <ul style="list-style-type: none"><li>• Timer Mode, see Table 11-53 on page 1185 for encoding</li><li>• Gated Timer Mode, see Table 11-53 on page 1185 for encoding</li><li>• Counter Mode, see Table 11-54 on page 1186 for encoding</li></ul>
T6M	5:3	rw	<b>Timer 6 Mode Control</b> (Basic Operating Mode) 000 Timer Mode 001 Counter Mode 010 Gated Timer Mode with Gate active low 011 Gated Timer Mode with Gate active high 1XX <b>Reserved.</b> Do not use this combination!
T6R	6	rw	<b>Timer 6 Run Bit</b> 0 Timer/counter 6 stops 1 Timer/counter 6 runs
T6UD	7	rw	<b>Timer 6 Up/Down Control</b> (when T6UDE is cleared) 0 Counting up 1 Counting down
T6UDE	8	rw	<b>Timer 6 External Up/Down Enable</b> 0 Counting direction is internally controlled by SW 1 Counting direction is externally controlled by line TxEUD
T6OE	9	rw	<b>Overflow/Underflow Output Enable</b> 0 T6 overflow/underflow can not be externally monitored 1 T6 overflow/underflow may be externally monitored via T6OUT

**CONFIDENTIAL**

**GPT 1 and 2**

Field	Bits	Type	Description
<b>T6OTL</b>	10	rwh	<b>Timer 6 Output Toggle Latch</b> Toggles on each overflow/underflow of T6. Can be set or reset by software.
<b>BPS2</b>	12:11	rw	<b>Timer Block Prescaler 2</b> The maximum input frequency 00 For Timer 5/6 is $f_{hw\_clk} / 4$ 01 For Timer 5/6 is $f_{hw\_clk} / 2$ 10 For Timer 5/6 is $f_{hw\_clk} / 16$ 11 For Timer 5/6 is $f_{hw\_clk} / 8$
<b>T6CLR</b>	14	rw	<b>Timer 6 Clear Bit</b> 0 Timer 6 is not cleared on a capture event 1 Timer 6 is cleared on a capture event
<b>T6SR</b>	15	rw	<b>Timer 6 Reload Mode Enable</b> 0 Reload from register CAPREL disabled 1 Reload from register CAPREL enabled
<b>RESERVED</b>	13	r	Reserved, these bits must be left at their reset values.

**Table 11-53 Timer 6 Input Parameter Selection for Timer Mode and Gated Timer Mode**

<b>T6CON.</b> <b>T6I</b>	Prescaler for $f_{hw\_clk}$ <b>(T6CON.BPS2 = 00)</b>	Prescaler for $f_{hw\_clk}$ <b>(T6CON.BPS2 = 01)</b>	Prescaler for $f_{hw\_clk}$ <b>(T6CON.BPS2 = 10)</b>	Prescaler for $f_{hw\_clk}$ <b>(T6CON.BPS2 = 11)</b>
000	4	2	16	8
001	8	4	32	16
010	16	8	64	32
011	32	16	128	64
100	64	32	256	128
101	128	64	512	256
110	256	128	1024	512
111	512	256	2048	1024

**CONFIDENTIAL**

**GPT 1 and 2**

**Table 11-54 Timer 6 Input Parameter Selection for Counter Mode**

<b>T6CON.T6I</b>	<b>Triggering Edge for Counter Update</b>
000	None. Counter T6 is disabled
001	Positive transition (raising edge) on T6IN
010	Negative transition (falling edge) on T6IN
011	Any transition ( raising or falling edge ) on T6IN
1XX	<b>Reserved.</b> Do not use this combination!

**T5CON**

**Timer 5 Control Register**

**Reset Value 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>T5 SR</b>	<b>T5 CLR</b>	<b>CI</b>	<b>T5 CC</b>	<b>CT3</b>	<b>T5 RC</b>	<b>T5 UDE</b>	<b>T5 UD</b>	<b>T5R</b>	<b>T5M</b>			<b>T5I</b>			

<b>Field</b>	<b>Bits</b>	<b>Type</b>	<b>Description</b>
<b>T5I</b>	2:0	rw	<b>Timer 5 Input Parameter Selection</b> <ul style="list-style-type: none"> <li>Timer Mode, refer to Table 11-55 on page 1188 for encoding</li> <li>Gated Timer Mode, refer to Table 11-55 on page 1188 for encoding</li> <li>Counter Mode, refer to Table 11-56 on page 1188 for encoding</li> </ul>
<b>T5M</b>	5:3	rw	<b>Timer 5 Mode Control (Basic Operating Mode)</b> <ul style="list-style-type: none"> <li>000 Timer Mode</li> <li>001 Counter Mode</li> <li>010 Gated Timer Mode with Gate active low</li> <li>011 Gated Timer Mode with Gate active high</li> <li>1XX <b>Reserved.</b> Do not use this combination!</li> </ul>
<b>T5R</b>	6	rw	<b>Timer 5 Run Bit</b> <ul style="list-style-type: none"> <li>0 Timer/counter 5 stops</li> <li>1 Timer/counter 5 runs</li> </ul>
<b>T5UD</b>	7	rw	<b>Timer 5 Up/Down Control</b> (when T5UDE is cleared) <ul style="list-style-type: none"> <li>0 Counting up</li> <li>1 Counting down</li> </ul>

**CONFIDENTIAL**

**GPT 1 and 2**

Field	Bits	Type	Description
<b>T5UDE</b>	8	rw	<b>Timer 5 External Up/Down Enable</b> 0 Counting direction is internally controlled by SW 1 Counting direction is externally controlled by line TxEUD
<b>T5RC</b>	9	rw	<b>Timer 5 Remote Control</b> 0 Timer/counter x is controlled by its own run bit T5R 1 Timer/counter 5 is controlled by the run bit of core Timer 6
<b>CT3</b>	10	rw	<b>Timer 3 Capture Trigger Enable</b> 0 Capture trigger from input line CAPIN 1 Capture trigger from T3 input lines
<b>T5CC</b>	11	rw	<b>Timer 5 Capture Correction</b> 0 T5 is just captured without any correction 1 T5 is decremented by 1 before being captured
<b>CI</b>	13:12	rw	<b>Register CAPREL Capture Trigger Selection</b> (depending in bit CT3) 00 Capture disabled 01 Positive transition (rising edge) on CAPIN or any transition on T3IN 10 Negative transition (falling edge) on CAPIN or any transition on T3EUD 11 Any transition (rising or falling edge) on CAPIN or any transition on T3IN or T3EUD
<b>T5CLR</b>	14	rw	<b>Timer 5 Clear Bit</b> 0 Timer 5 is not cleared on a capture event 1 Timer 5 is cleared on a capture event
<b>T5SC</b>	15	rw	<b>Timer 5 Capture Mode Enable</b> 0 Capture into register CAPREL disabled 1 Capture into register CAPREL enabled

**Table 11-55 Timer 5 Input Parameter Selection for Timer Mode and Gated Timer Mode**

<b>T5CON.T5I</b>	Prescaler for $f_{hw\_clk}$ (T6CON.BPS2 = 00)	Prescaler for $f_{hw\_clk}$ (T6CON.BPS2 = 01)	Prescaler for $f_{hw\_clk}$ (T6CON.BPS2 = 10)	Prescaler for $f_{hw\_clk}$ (T6CON.BPS2 = 11)
000	4	2	16	8
001	8	4	32	16
010	16	8	64	32
011	32	16	128	64
100	64	32	256	128
101	128	64	512	256
110	256	128	1024	512
111	512	256	2048	1024

**Table 11-56 Timer 5 Input Parameter Selection for Counter Mode**

<b>T5CON.T5I</b>	Triggering Edge for Counter Update
X00	None. Counter T5 is disabled
001	Positive transition (rising edge) on T5IN
010	Negative transition (falling edge) on T5IN
011	Any transition ( rising or falling edge ) on T5IN
101	Positive transition (rising edge) of T3OTL
110	Negative transition (falling edge) of T3OTL
111	Any transition (rising or falling edge) of T3OTL

### 11.7.3.5 Register Mapping

For information about the GPT12 PD-Bus Register Mapping refer to [Section 12.2 PD-Bus Register Addresses \(on Page 1273\)](#).



**CONFIDENTIAL**

**GPT 1 and 2**

## 11.7.4 Interrupts

For a detailed description of the various interrupts refer to [Section 11.7.3 Kernel Description \(on Page 1147\)](#). An overview is given in [Table 11-57](#).

**Table 11-57 GPT1\_2 Interrupt Sources**

Interrupt	Signal	Description
Timer 2 Overflow	int_t2_o	Interrupt is requested on overflow of Timer 2 if counting up.
Timer 2 Underflow	int_t2_o	Interrupt is requested on underflow of Timer 2 if counting down.
Timer 3 Overflow	int_t3_o	Interrupt is requested on overflow of Timer 3 if counting up.
Timer 3 Underflow	int_t3_o	Interrupt is requested on underflow of Timer 3 if counting down.
Timer 4 Overflow	int_t4_o	Interrupt is requested on overflow of Timer 4 if counting up.
Timer 4 Underflow	int_t4_o	Interrupt is requested on underflow of Timer 4 if counting down.
Timer 5 Overflow	int_t5_o	Interrupt is requested on overflow of Timer 5 if counting up.
Timer 5 Underflow	int_t5_o	Interrupt is requested on underflow of Timer 5 if counting down.
Timer 6 Overflow	int_t6_o	Interrupt is requested on overflow of Timer 6 if counting up.
Timer 6 Underflow	int_t6_o	Interrupt is requested on underflow of Timer 6 if counting down.
Rotation Direction Change Timer 2	int_t2_o	Interrupt is requested on a change of the count direction in the Incremental Interface Mode ( <b>T2CON.T2I</b> = 110).
Edge Detection Timer 2	int_t2_o	Interrupt is requested on a successful detected edge resulting in a timer count action ( <b>T2CON.T2I</b> = 111).
Rotation Direction Change Timer 3	int_t3_o	Interrupt is requested on a change of the count direction in the Incremental Interface Mode ( <b>T3CON.T3I</b> = 110).
Edge Detection Timer 3	int_t3_o	Interrupt is requested on a successful detected edge resulting in a timer count action ( <b>T3CON.T3I</b> = 111).

**CONFIDENTIAL**

**GPT 1 and 2**

**Table 11-57 GPT1\_2 Interrupt Sources**

Interrupt	Signal	Description
Rotation Direction Change Timer 4	int_t4_o	Interrupt is requested on a change of the count direction in the Incremental Interface Mode ( <b>T4CON.T4I</b> = 110).
Edge Detection Timer 4	int_t4_o	Interrupt is requested on a successful detected edge resulting in a timer count action ( <b>T4CON.T4I</b> = 111).
Reload Action Timer 2	int_t2_o	Interrupt is requested on a trigger signal for reloading Timer 3 in Reload Mode ( <b>T2CON.T2I</b> = 100).
Reload Action Timer 4	int_t4_o	Interrupt is requested on a trigger signal for reloading Timer 3 in Reload Mode ( <b>T4CON.T4I</b> = 100).
Capture Action Timer 2	int_t2_o	Interrupt is requested on a trigger signal for a capture action to capture Timer 3 in Timer 2 Capture Mode ( <b>T2CON.T2I</b> = 101).
Capture Action Timer 4	int_t4_o	Interrupt is requested on a trigger signal for a capture action to capture Timer 3 in Timer 4 Capture Mode ( <b>T4CON.T4I</b> = 101).
Capture Action Timer Block 2	int_cr_o	Interrupt is requested on a trigger signal for a capture action of Timer 5 to register CAPREL in Capture Mode ( <b>T5CON.T5SC</b> = 1).

**CONFIDENTIAL**

**Port Control Logic**

## 11.8 Port Control Logic

History	
Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 0.02	
<b>Page 1202</b>	Updated <b>PCL_ID</b> register
<b>Page 1227</b>	Updated <b>Table 11-68 Pad Signals</b>
Changes for Rev. 1.01	
<b>Page 1224</b>	Updated <b>Table 11-67 Signals from the CGU</b> WS00006502, WS00006641
<b>Page 1192</b>	Updated <b>System Integration</b> : WS00006682
<b>Page 1209</b>	Removed System Interrogation from <b>Section 11.8.10 Internal Signal Monitoring</b> WS00006682
Changes for Rev. 1.02	
<b>Page 1197</b>	Updated reset value for GPIO pad 51 WS00005866
Changes for Rev. 1.04	
<b>Page 1224</b>	Updated footnote for <b>Table 11-67 Signals from the CGU</b> WS00006643
<b>Page 1205</b> to <b>Page 1207</b>	Renamed <b>SCU_NUMx</b> to <b>ID_NUMx</b>
<b>Page 1225</b>	Corrected frequencies in tststate32 remarks
Changes for Rev. 1.05	
<b>Page 1212</b>	Block Select -> Signal Select in heading of <b>Table 11-64</b> to <b>Table 11-72</b>
<b>Page 1226</b>	Signal added to <b>Table 11-67 Signals from the CGU</b> WS00008954
Changes for Rev. 1.06	
<b>Page 1239</b>	Note added to <b>gsm_mon_iqram</b> signal WS00009067
<b>Page 1234</b>	Added CSWITCH_N_i (05) and changed name of OE_N_O (03) in <b>Table 11-70 PCL_PER Signals</b> WS00008905

### System Integration:

- Supply domain: VDD\_MAIN
- Chip internal interfaces:
  - Clock domain: Refer to [Section 10.4.1.3 Sub-System Clocks and Enables \(on Page 661\)](#) and see [Figure 10-10 Clock Enable \(on Page 662\)](#).
  - Bus domain: PD-Bus  
Bus interface common with internal signal monitoring
  - Interrupt sources: no
  - Other interface:
- Chip external signals related to this block (refer to [Chapter 3](#) for pin configuration options): all configurable pins.

### 11.8.1 Functional Overview

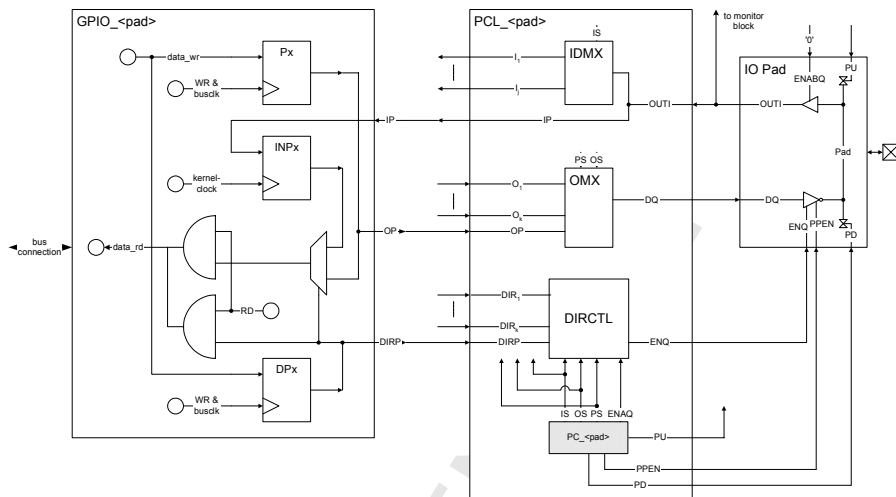
Most digital pads may be configured to be either a general purpose IO port (GPIO) or be connected to one of several inputs or to one of several outputs from chip internal blocks.

The connection and the GPIO function are realized individually for each pad <pad> with the blocks PCL\_<pad> and GPIO\_<pad> (denoted PCL and GPIO in the rest of this Section). Both blocks are accessible for the MCU via the bus.

The principal structure of PCL and GPIO is shown in [Figure 11-75](#).

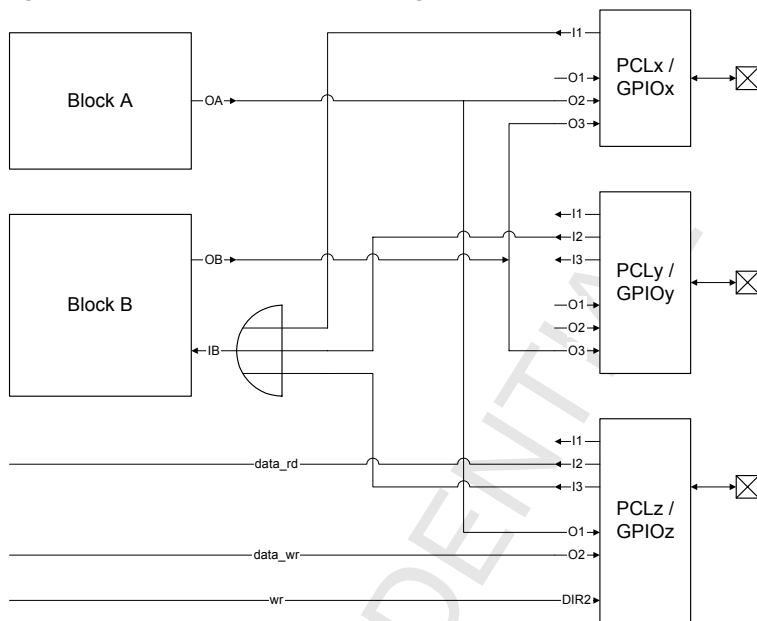
The main functional blocks of the PCL are the pad control register [PCL\\_<pad>](#), the input demultiplexer IDMX, the output multiplexer OMX and the direction control block DIRCTL.

**Figure 11-75 Architecture of Pad Control and Port Logic**



A typical connection of several PCL blocks and chip internal blocks is shown in [Figure 11-76](#).

Figure 11-76 Example for Connecting PCL Blocks and Chip Internal Blocks



*Note: Only the principal circuitry for pads with GPIO and additional function is described here. The control circuitry for other pads like analog pads or digital pads with dedicated non multiplexed function is described in the related module sections (for example, RTC, Measurement, ...).*

**CONFIDENTIAL**

**Port Control Logic**

## 11.8.2 Register Description PCL\_<pad>

The **PCL\_<pad>** registers select the alternate functions of the GPIO chip pads. For example, **PCL\_00** is the register that selects Alt0, Alt1, or Alt2 for the GPIO\_00 pad (ball name KP0), refer to [Table 3-1 Pin List \(on Page 51\)](#).

Certain pads do not use all the bits (they do not have alternate functions).

### PCL\_<pad>

**Control and GPIO Register for pad <pad>**

**Reset value: refer to [Table 11-58](#)**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENAQ	PDPN	PPE	RESERVED	DPx	Px	PS	RESERVED			OS		RESERVED			IS

Field	Bits	Type	Description
<b>IS</b>	1:0	rw	<b>Input Selection</b> Connects the pad input to one of up to 3 chip internal inputs (see <a href="#">Figure 11-75 (on page 1193)</a> ): 00 No input selected 01 Alt0 10 Alt1 11 Alt2  <b>Note: If an alternate is selected for a pad that does not have the alternate function, the pad behaves as if 00 (no input) were selected.</b>
<b>OS</b>	5:4	rw	<b>Output Selection</b> Connects one of up to 3 chip internal outputs to the pad output (see <a href="#">Figure 11-75</a> ): 00 No output selected 01 Alt0 02 Alt1 03 Alt2  <b>Note: If an alternate is selected for a pad that does not have the alternate function, the pad behaves as if 00 (no output) were selected.</b>
<b>PS</b>	8	rw	<b>Port Selection</b> 0: The internal signal connected to the pad cell and the pad direction are determined by the bit fields <b>IS</b> and <b>OS</b> and related internal direction signals. 1: The pad is connected to its associated GPIO cell. The direction is determined by the corresponding DP register bit.

**CONFIDENTIAL**

**Port Control Logic**

Field	Bits	Type	Description
<b>Px</b>	9	rwh	<b>Data of GPIOx</b> The written value and the hardware (pad) determined value are stored internally in different registers. Thus no conflict is possible when the register is written. The read delivers either of these values, depending on DPx, as explained in <a href="#">Section 11.8.6 General Purpose I/O (on Page 1200)</a> .
<b>DPx</b>	10	rw	<b>Direction Control for GPIOx</b> 0 GPIOx is input. The value applied on the input pad updates the Px field value. 1 GPIOx is output. The value set in the Px field drives the pad output value.
<b>PPEN</b>	12	rw	<b>Push/Pull Enable</b> 0 Push/pull function of the pad is activated 1 Open drain function of the pad is activated
<b>PDPU</b>	14:13	rw	<b>Pullup/Pulldown Selection</b> 00 Pullup and pulldown resistors of the pad are deactivated 01 Pullup resistor of the pad is activated 10 Pulldown resistor of the pad is activated 11 Reserved
<b>ENAQ</b>	15	rw	0 Output function of the pad as determined by the other bits of this register 1 Output of the pad is set to tristate
<b>RESERVED</b>	3:2, 7:6, 11, 31:16	r	Reserved; these bits must be left at their reset values.



**Table 11-58 GPIO Pad Reset Values**

GPIO Pad Number	Reset Value (Hex)	GPIO Pad Number	Reset Value (Hex)	GPIO Pad Number	Reset Value (Hex)
00	8900 <sup>1)</sup>	20	C900 <sup>2)</sup>	40	C900 <sup>2)</sup>
01	8900 <sup>1)</sup>	21	C900 <sup>2)</sup>	41	C900 <sup>2)</sup>
02	8900 <sup>1)</sup>	22	C900 <sup>2)</sup>	42	C900 <sup>2)</sup>
03	8900 <sup>1)</sup>	23	C900 <sup>2)</sup>	43	A900 <sup>3)</sup>
04	8900 <sup>1)</sup>	24	C900 <sup>2)</sup>	44	C900 <sup>2)</sup>
05	8900 <sup>1)</sup>	25	C900 <sup>2)</sup>	45	C900 <sup>2)</sup>
06	8900 <sup>1)</sup>	26	C900 <sup>2)</sup>	46	C900 <sup>2)</sup>
07	8900 <sup>1)</sup>	27	C900 <sup>2)</sup>	47	C900 <sup>2)</sup>
08	8900 <sup>1)</sup>	28	C900 <sup>2)</sup>	48	A900 <sup>3)</sup>
09	8900 <sup>1)</sup>	29	C900 <sup>2)</sup>	49	C900 <sup>2)</sup>
10	8900 <sup>1)</sup>	30	C900 <sup>2)</sup>	50	C900 <sup>2)</sup>
11	8900 <sup>1)</sup>	31	0010 <sup>4)</sup>	51	A900 <sup>3)</sup>
12	A800 <sup>5)</sup>	32	0010 <sup>4)</sup>	52	0000 <sup>6)</sup>
13	8900 <sup>1)</sup>	33	0010 <sup>4)</sup>	53	C900 <sup>2)</sup>
14	8900 <sup>1)</sup>	34	0010 <sup>4)</sup>	54	C900 <sup>2)</sup>
15	8900 <sup>1)</sup>	35	0010 <sup>4)</sup>	55	C900 <sup>2)</sup>
16	9900 <sup>5)</sup>	36	0010 <sup>4)</sup>	56	C900 <sup>2)</sup>
17	9900 <sup>5)</sup>	37	C900 <sup>2)</sup>	57	8900 <sup>1)</sup>
18	0010 <sup>7)</sup>	38	C900 <sup>2)</sup>		
19	4001	39	C900 <sup>2)</sup>		

1) Tristate.

2) Tristate and Pull Down

3) Tristate and Pull Up.

4) Pad configured as Alt 0 Output.

5) Tristate and Open Drain.

6) Pad drives the value either Low or High.

7) Pad configured as Alt 0 Output.

CONFIDENTIAL

Port Control Logic

### 11.8.3 IDMX

The input demultiplexer IDMX connects the internal pad output OUTI to an input of a chip internal block input via one of the lines I<sub>x</sub>. The maximal number of I<sub>x</sub> lines depends on the actual size of the input select word **PCL\_<pad>.IS**. The size may be 1 or 2 bits, allowing 1 or 3 different input lines. If all **IS** bits are 0, no input is selected. Others return 0 when read. The same is true for output select word **PCL\_<pad>.OS**.

The respective port input is always connected to OUTI. Thus a read access to the port shows always the value of the respective input line, even if the port function of the pad is not activated.

The function of IDMX is described in [Table 11-59](#).

**Table 11-59 Input Programming**

IS	I1 ... I3
00	Depending on the connected block input: inactive state of this input.
others	I <sub>x</sub> for x = <b>PCL_&lt;pad&gt;.IS</b> : OUTI I <sub>x</sub> for x ≠ <b>PCL_&lt;pad&gt;.IS</b> : Inactive state of connected block input

If different alternative input lines shall be connected to one input of a block, these lines are "ORed" or 'ANDed' before entering the block, depending on the inactive state of the block input. No other hardware is used outside of PCL. The actual input pad is selected by programming its respective input line I<sub>x</sub> to be connected to OUTI and not selecting the respective I<sub>x</sub> lines of all other related pads. For example, in case of inactive input state 0, these I<sub>x</sub> lines are set to 0 and do not influence the output of the OR gate.

### 11.8.4 OMX

The output multiplexer OMX connects the GPIO output OP or one of several outputs of chip internal blocks to the pad data input DQ. The maximal number of block outputs lines O<sub>x</sub> depends on the actual size of the output select word OS. OS is programmed in the register **PCL\_<pad>**. The size may be 1 or 2 bits, allowing 1 or 3 different output lines in addition to OP. If all OS bits are 0, no output is selected.

The function of OMX is described in [Table 11-60](#).

**Table 11-60 Output Programming**

PS	OS	DQ
1	xx	OP
0	00	0
0	others	DQ = O <sub>OS</sub>

Each block output may be connected to an arbitrary number of pads without additional hardware. The actual output pad is selected by programming its respective output line

**CONFIDENTIAL**

**Port Control Logic**

Ox to be connected to DQ and deselecting the respective Ox lines of all other related pads.

During switching of OS, spikes may occur on the internal DQ line. If this is a problem, the output driver can be disabled temporarily as described in the following DIRCTL section.

Logical combinations of outputs: some logical combinations of block outputs are possible. If required, refer to [Table 3-1 Pin List \(on Page 51\)](#). They are selected like other output signals.

### 11.8.5 DIRCTL

The direction control block DIRCTL determines the pad direction out of the PS, IS and OS selection and the related DIRx input and sets ENQ correspondingly, as described hereafter.

If neither the GPIO nor an input or an output are selected, that is,  $PS = IS = OS = 0$ , then the pad is set to tristate, that is, ENQ is deactivated.

If GPIO is selected, that is,  $PS = 1$ , the direction is determined by DIRP.

If no GPIO, no input and one output are selected, that is,  $PS = IS = 0$  and  $OS > 0$ , then the pad is used as an output, that is, ENQ is activated.

If no GPIO, one input and no output are selected, that is,  $PS = 0$ ,  $IS > 0$  and  $OS = 0$ , then the pad is used as an input, that is, ENQ is deactivated.

If no GPIO, one input and one output are selected, that is,  $PS = 0$ ,  $IS > 0$  and  $OS > 0$ , then the pad is used as an IO pad with the direction control signal DIR(OS). If the pad shall be used as a true bidirectional pad, the direction control signal will be delivered by the respective peripheral. If the pad shall be used as an output only, the direction control signal has a fixed value, always enabling the output driver. In this case setting  $IS > 0$  allows to feed back the output signal to an internal E-GOLDradio signal.

**Table 11-61 Pad Function Depending on Programming of PCL\_<pad>**

ENAQ	PS	IS	OS	ENQ	Pad function	
					Input	Output
1	x	x	x	1	Refer to <a href="#">Table 11-59 Input Programming (on Page 1198)</a>	Tristate
0	1	x	x	not DIRP	GPIO, direction depending on DIRP	
x	0	0	0	1	Not active	Tristate
0	0	0	>0	0	Not active	DQ ( <a href="#">Table 11-59</a> )

**CONFIDENTIAL**

**Port Control Logic**

**Table 11-61 Pad Function Depending on Programming of PCL\_<pad>**

ENAQ	PS	IS	OS	ENQ	Pad function	
					Input	Output
x	0	>0	0	1	Refer to <a href="#">Table 11-59</a>	Tristate
0	0	>0	>0	not DIR <sub>OS</sub>	Bidirectional, direction depending on DIR <sub>OS</sub>	

ENAQ allows setting the pad to tristate and programming the desired value for OS at the same time. Thus the spikes which may occur on the OMX output DQ during switching will not be visible on the chip output. The output level can be asserted by setting PU or PD. In a following cycle, the output may be activated by de-asserting ENAQ and PU or PD while OS remains unchanged.

### 11.8.6 General Purpose I/O

In order to accept or generate external control signals, the PMB7870 provides a number of general purpose IO (GPIO) lines. Each GPIO is related to one pad, as shown in [Table 3-1 Pin List \(on Page 51\)](#). The pad may be configured to show the GPIO function or some other function, as shown in the subsections above in principle and in [Table 3-1](#) in detail.

A general purpose IO cell x associated to a pad contains the data direction definition bit DPx and the data bit Px (see [Figure 11-75](#)).

It is always possible to write to or read from Px and DPx, even if the pad is not configured as port.

If DPx is set to 0, the data direction of the GPIO x is input. In this case reading of bit Px delivers the pad value.

If DPx is set to 1, the data direction of the GPIO x is output. In this case reading of bit Px delivers the value written to Px.

The EBU control block in the PCL controls the EBU Pads:

- A0 .. A23
- D0 .. D15
- CS0\_N, CS1\_N, CS3\_N
- ADV\_N
- WR\_N
- RD\_N
- OE\_N
- BHE\_N.

**CONFIDENTIAL**

**Port Control Logic**

### 11.8.6.1 EBU Pull Down Control

**EBU\_PDC**

**EBU Pull Down Control Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED							PDW	RESERVED							

Field	Bits	Type	Description
<b>PDW</b>	8	rw	<b>Pull Down Weak</b> <b>PDW</b> is used in the Minimum Power Consumption Optimization and must be used the MCU IDLE command. 0 No input selected 1 A Pull-Down is set on the Address (A0..A23) and Data (D0..D15) EBU pads Refer to <a href="#">Section 6.7.6.1 Idle Mode (on Page 259)</a> .
<b>RESERVED</b>	15:9, 7:0	r	Reserved; these bits must be left at their reset values.

**CONFIDENTIAL**

**Port Control Logic**

## 11.8.7 Control Registers

### 11.8.7.1 Port Control Logic Identification Register

**PCL\_ID**

**Port Control Logic Identification Register**

**Reset value: 2303<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>Module_ID</b>								<b>Revision_Number</b>							

Field	Bits	Type	Description
<b>Revision_Number</b>	0:7	r	<b>Port Control Logic Revision Number</b> These hard-wired bits are used for module revision numbering.
<b>Module_ID</b>	8:15	r	<b>Port Control Logic Identification Number</b> These hard-wired bits are used for module identification numbering.

## 11.8.8 Flash Overwrite Protection

### 11.8.8.1 Functional Description

The external memory device is normally expected to be a flash device. Therefore, the names used in this section are focused on the flash device. It is however possible to utilize other type of external memory. All external memory is treated in the same manner.

The external flash memory device must be protected again an accidental overwrite. An overwrite is especially dangerous in the write-suspend mode when the flash on-chip write protection mechanism is already deactivated.

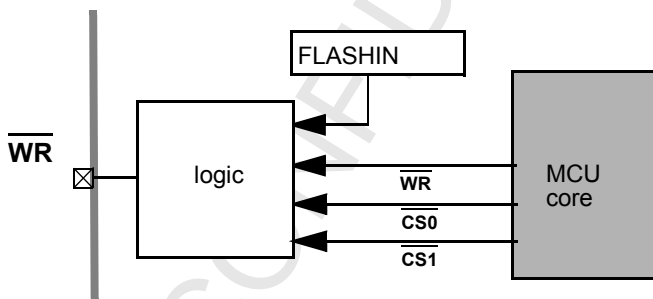
The overwrite protection mechanism suppresses the activation of the PMB7870 external WRITE pin **WR** during accesses to the CS0 or CS1 memory ranges. The WRITE pin **WR** can only be activated when the core output write signal is set and a dedicated write-enable register for the dressed memory page contains a special pattern.

As this protection controls the **WR** signal directly, it does not require a flash memory with an extra protection pin.

*Note: The extra protection pin can be controlled by software driving a general purpose output pin.*

**Figure 11-77** shows the block schematic of the flash overwrite protection.

**Figure 11-77 Flash Write Protection Functional Diagram**



**CONFIDENTIAL**

**Port Control Logic**

### 11.8.8.2 Flash Write Enable Register

**FLASHIN**

**Flash Write Enable Register**

**Reset value: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								WRITE_ENABLE_CS1				WRITE_ENABLE_CS0			

Field	Bits	Type	Description
WRITE_ENABLE_CS0	3:0	rw	6 <sub>H</sub> : Write on CS0 enabled Any other value: Write disabled
WRITE_ENABLE_CS1	7:4	rw	6 <sub>H</sub> : Write on CS1 enabled Any other value: Write disabled
RESERVED	15:8	rw	Reserved; these bits must be left at their reset values.

*Note: Writing 0066<sub>H</sub> to this register enables writing to both CS0 and CS1.*



**CONFIDENTIAL**

**Port Control Logic**

### 11.8.9 Fuse Boxes and Corresponding Registers

The PMB7870 provides fuse boxes available via registers. These five registers total 80 bits. Their values are hard wired for each individual chip during fabrication (refer to [Table 11-62 Wafer Character Coding \(on 6 bits\) \(on Page 1208\)](#)).

*Note: The 6 bits of the 7th Wafer Identifier character (**WafldChar6[0:3]**, **Wfld6[4]**, and **Wfld6[5]**) are spread over three registers.*

#### ID\_SNUM0

##### Serial Number Register 0

**Reset value: xxxx<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>WafldChar6[0:3]</b>				<b>ProdCodeChar1</b>						<b>ProdCodeChar0</b>					

Field	Bits	Type	Description
<b>ProdCodeChar0</b>	5:0	r	1st character of the Product Code <sup>1)</sup>
<b>ProdCodeChar1</b>	11:6	r	2ed character of Product Code <sup>2)</sup>
<b>WafldChar6[0:3]</b>	15:12	r	First 4 bits of the 7th alphanumeric character of the Wafer Identifier <i>Note: Bit 0 is the LSB.</i>

<sup>1)</sup> N (1E<sub>H</sub>, 011110<sub>B</sub>) for E-GOLDradio.

<sup>2)</sup> 9 (09<sub>H</sub>, 001001<sub>B</sub>) for E-GOLDradio.

**CONFIDENTIAL**

**Port Control Logic**

## ID\_SNUM1

### Serial Number Register 1

**Reset value: xxxx<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CodingType			Waf Id Char 6[4]	WafldChar1						WafldChar0					

Field	Bits	Type	Description
WafldChar0	5:0	r	1st alphanumeric character of Wafer Identifier
WafldChar1	11:6	r	2ed alphanumeric character of Wafer Identifier
WafldChar6[4]	12	r	5th bit of the 7th alphanumeric character of Wafer Identifier
CodingType	15:13	r	Type of coding used for the <b>SCU_SNUMx</b> registers in this chip

## ID\_SNUM2

### Serial Number Register 2

**Reset value: xxxx<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	FabCode			WafldChar3						WafldChar2					

Field	Bits	Type	Description
WafldChar2	5:0	r	3rd alphanumeric character of Wafer Identifier
WafldChar3	11:6	r	4th alphanumeric character of Wafer Identifier
FabCode	12:14	r	Fabrication Code
1	15	r	Always fused to 1

**CONFIDENTIAL**

**Port Control Logic**

### ID\_SNUM3

#### Serial Number Register 3

**Reset value: xxxx<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TestUse				WafldChar5						WafldChar4					

Field	Bits	Type	Description
WafldChar4	5:0	r	5th alphanumeric character of Wafer Identifier
WafldChar5	11:6	r	6th alphanumeric character of Wafer Identifier
TestUse	12:15	r	Reserved for testing

### ID\_SNUM4

#### Serial Number Register 4

**Reset value: xxxx<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Waf Id Char 6[5]	RESERVED	y_coordinate						x_coordinate							

Field	Bits	Type	Description
x_coordinate	6:0	r	X Coordinate on wafer
y_coordinate	13:7	r	Y Coordinate on wafer
WafldChar6[5]	15	r	6th bit of the 7th alphanumeric character of Wafer Identifier
RESERVED	14	r	Reserved; these bits must be left at their reset values.

**Table 11-62 Wafer Character Coding (on 6 bits)**

Character	Hex Value	Character	Hex Value
1	01	J	1A
2	02	K	1B
3	03	L	1C
4	04	M	1D
5	05	N	1E
6	06	O	1F
7	07	P	20
8	08	Q	21
9	09	R	22
A	11	S	23
B	12	T	24
C	13	U	25
D	14	V	26
E	15	W	27
F	16	X	28
G	17	Y	29
H	18	Z	2A
I	19		

## 11.8.10 Internal Signal Monitoring

### 11.8.10.1 Functionality

This block is used to monitor internal signals.

The selected internal signals are redirected to two external signals: MON1 and MON2. The selection is done in two steps:

1. The selection of the block done with **MON\_CRx.BLOCK\_SELECT**.
2. The selection of the signal from the block done with **MON\_CRx.SIGNAL\_SELECT**.

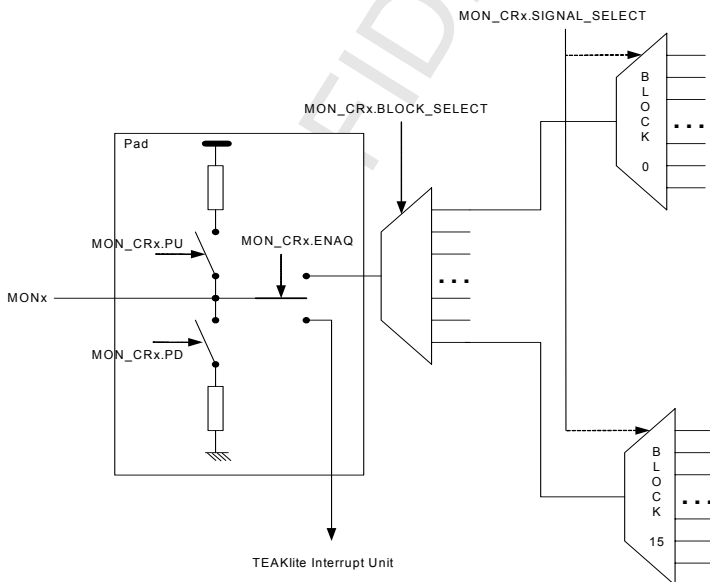
*Note: The signals MON1 and MON2 can be configured as an external input signal, in this case the MON1 and MON2 signals are connected to the TEAKlite TEAKLite Interrupt Unit (on Page 337).*

*The configuration of the MON1 signal and MON2 signal is independent; it is done by writing 1 into the **MON\_CR1.ENAQ** for the MON1 signal and by writing 1 into the **MON\_CR2.ENAQ** for the MON2 signal.*

*Note: The pads MON1 and MON2 have a different usage during the system reset, refer to [Chapter 14 System Reset \(on Page 1401\)](#).*

**Figure 11-78 (on page 1209)** shows the signals path and control.

**Figure 11-78 Monitoring of Internal Signals at the Monitor Pads**



CONFIDENTIAL

Port Control Logic

### 11.8.10.2 Control Registers

**MON\_CR1** controls the signal MON1.

**MON\_CR2** controls the signal MON2.

**MON\_CRx**

**MON\_CR1**

**Control Register 1**

**Reset value: C900<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENAQ	PD	PU	RESERVED	BLOCK_SELECT				RESERVED	SIGNAL_SELECT						

**MON\_CR2**

**Control Register 2**

**Reset value: C900<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENAQ	PD	PU	RESERVED	BLOCK_SELECT				RESERVED	SIGNAL_SELECT						

Field	Bits	Type	Description
<b>SIGNAL_SELECT</b>	6:0	rw	<b>Signal Selection</b>
<b>BLOCK_SELECT</b>	11:8	rw	<b>Block Selection</b> Refer to <a href="#">Table 11-63</a> .
<b>PD, PU</b>	14:13	rw	<b>Pullup and Pulldown Enable</b> These bits activate pullup and pulldown resistors of the associated pad. 00 Pullup and pulldown resistors are deactivated 01 Pullup resistor of pad is activated 10 Pulldown resistor of pad is activated 11 Reserved
<b>ENAQ</b>	15	rw	<b>Pad Mode</b> 0 Pad in Output mode. 1 Pad in Input mode.
<b>RESERVED</b>	7, 12	r	Reserved; these bits must be left at their reset values.

**Table 11-63 Block List**

<b>BLOCK_SELECT</b>	<b>Block Name</b>
0000	<b>DSP_INT</b>
0001	<b>DSP_ANA</b>
0010	<b>Cipher A53</b>
0011	<b>CGU</b>
0100	<b>PADS</b>
0101	<b>INTR_EXT</b>
0110	<b>PCL_PER</b>
0111	<b>GPRS_GEA3</b>
1000	<b>MCU_PER</b>
1001	Reserved
1010	Reserved
1011	Reserved
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

**CONFIDENTIAL**

**Port Control Logic**

### 11.8.10.2.1 DSP\_INT

**Table 11-64 Signals from the DSP (Mainly Interrupts)**

Signal Name	Signal Select (hex)	Remarks
mcu0_int	01	DSP Interrupt A0
mcu1_int	02	DSP Interrupt A0
mcu2_int	03	DSP Interrupt A0
mcu3_int	04	DSP Interrupt A0
frame_int	05	DSP Interrupt A0
-	06	reserved
-	07	reserved
-	08	reserved
-	09	reserved
-	0A	reserved
codon_int	0B	DSP Interrupt A0
modu_int	0C	DSP Interrupt A0
chadec_int	0D	DSP Interrupt A0
eq_int	0E	DSP Interrupt A0
eqon_int	0F	DSP Interrupt A0
fcon_int	10	DSP Interrupt A0
monon_int	11	DSP Interrupt A0
scon_int	12	DSP Interrupt A0
bb_full_int	13	DSP Interrupt A0
i2s1rx_int	14	DSP Interrupt B0
i2s1tx_int	15	DSP Interrupt B0
i2s2rx_int	16	DSP Interrupt B0
i2s2tx_int	17	DSP Interrupt B0
-	18	reserved
i2s3tx_int	19	DSP Interrupt B0
vbtx_int	1A	DSP Interrupt B0
vbrx_int	1B	DSP Interrupt B0
-	1C	reserved
-	1D	reserved



**CONFIDENTIAL**

**Port Control Logic**

**Table 11-64 Signals from the DSP (Mainly Interrupts)**

Signal Name	Signal Select (hex)	Remarks
-	1E	reserved
-	1F	reserved
ssc1_rir_int	20	DSP Interrupt B0
ssc1_tir_int	21	DSP Interrupt B0
ssc1_eir_int	22	DSP Interrupt B0
sysmcu_int	23	DSP Interrupt B0
ciph_int	24	DSP Interrupt 1
dtmr10_int	25	DSP Interrupt 1
dtmr11_int	26	DSP Interrupt 1
dtmr2_int	27	DSP Interrupt 1
dspin0_int	28	DSP Interrupt 1
dspin1_int	29	DSP Interrupt 1
monin1_int	2A	DSP Interrupt 1
monin2_int	2B	DSP Interrupt 1
monin3_int	2C	DSP Interrupt 1
monin4_int	2D	DSP Interrupt 1
intfw0	2E	DSP Interrupt 2
intfw1	2F	DSP Interrupt 2
intfw2	30	DSP Interrupt 2
intfw3	31	DSP Interrupt 2
intfw4	32	DSP Interrupt 2
intfw5	33	DSP Interrupt 2
intfw6	34	DSP Interrupt 2
intfw7	35	DSP Interrupt 2
intfw8	36	DSP Interrupt 2
intfw9	37	DSP Interrupt 2
intfw10	38	DSP Interrupt 2
intfw11	39	DSP Interrupt 2
intfw12	3A	DSP Interrupt 2
intfw13	3B	DSP Interrupt 2

**CONFIDENTIAL**

**Port Control Logic**

**Table 11-64 Signals from the DSP (Mainly Interrupts)**

Signal Name	Signal Select (hex)	Remarks
intfw14	3C	DSP Interrupt 2
intfw15	3D	DSP Interrupt 2
int0	3E	DSP Interrupt Line 0
int1	3F	DSP Interrupt Line 1
int2	40	DSP Interrupt Line 2
tomcu0_int	41	Interrupt to MCU
tomcu1_int	42	Interrupt to MCU
tomcu2_int	43	Interrupt to MCU
tomcu3_int	44	Interrupt to MCU
-	45 to 7F	reserved

#### 11.8.10.2.2 DSP\_ANA

**Table 11-65 Signals from DSP**

Signal Name	Signal Select (hex)	Remarks
mon_rxclear	01	Audio-FE
mon_txclear	02	Audio-FE
mon_data_rrq_rx	03	Audio-FE
mon_data_wrq_tx	04	Audio-FE
mon_powerup_int_asmd_tx	05	Audio-FE
mon_powerup_int_asmd_rx	06	Audio-FE
mon_zero_detect	07	Audio-FE
mon_data_vbrx_left	08	Audio-FE
mon_data_vbrx_lsb4(0)	09	Audio-FE
mon_data_vbrx_lsb4(1)	0A	Audio-FE
mon_data_vbrx_lsb4(2)	0B	Audio-FE
mon_data_vbrx_lsb4(3)	0C	Audio-FE
mon_clk_vbrx	0D	Audio-FE
mon_data_vbtx	0E	Audio-FE
mon_clk_vbtx_dith	0F	Audio-FE
mon_clk_vbtx	10	Audio-FE

**CONFIDENTIAL**

**Port Control Logic**

**Table 11-65 Signals from DSP**

Signal Name	Signal Select (hex)	Remarks
mon_gclock_pll	11	BB-Receive
mon_clk_dither	12	BB-Receive
mon_gclock_mix	13	BB-Receive
mon_pdm1q	14	BB-Receive
mon_pdm2q	15	BB-Receive
mon_pdm1i	16	BB-Receive
mon_pdm2i	17	BB-Receive
mon_clk_bbrx	18	BB-Receive
mon_ciphering_ready	19	GSM-Cipher
mon_start_ciphering	1A	GSM-Cipher
mon_ecry_dcry_data	1B	GSM-Cipher
mon_dcry_valid	1C	GSM-Cipher
mon_ecry_valid	1D	GSM-Cipher
mon_gclock_dsp	1E	GSM-Cipher
mon_gclock_ram	1F	GSM-Cipher
mon_gclock_cipher	20	GSM-Cipher
mon_rx_state(0)	21	I2S1 (DSP)
mon_rx_state(1)	22	I2S1 (DSP)
mon_rx_state(2)	23	I2S1 (DSP)
mon_rx_state(3)	24	I2S1 (DSP)
mon_tx_state(0)	25	I2S1 (DSP)
mon_tx_state(1)	26	I2S1 (DSP)
mon_tx_state(2)	27	I2S1 (DSP)
mon_tx_state(3)	28	I2S1 (DSP)
mon_cl2_state(0)	29	I2S1 (DSP)
mon_cl2_state(1)	2A	I2S1 (DSP)
mon_cl2_state(2)	2B	I2S1 (DSP)
mon_cl1_state(0)	2C	I2S1 (DSP)
mon_cl1_state(1)	2D	I2S1 (DSP)
mon_cl1_state(2)	2E	I2S1 (DSP)

**CONFIDENTIAL**

**Port Control Logic**

**Table 11-65 Signals from DSP**

<b>Signal Name</b>	<b>Signal Select (hex)</b>	<b>Remarks</b>
mon_i2s_data_wrq	2F	I2S1 (DSP)
mon_i2s_data_rrq	30	I2S1 (DSP)
mon_rx_state(0)	31	I2S2 (DSP)
mon_rx_state(1)	32	I2S2 (DSP)
mon_rx_state(2)	33	I2S2 (DSP)
mon_rx_state(3)	34	I2S2 (DSP)
mon_tx_state(0)	35	I2S2 (DSP)
mon_tx_state(1)	36	I2S2 (DSP)
mon_tx_state(2)	37	I2S2 (DSP)
mon_tx_state(3)	38	I2S2 (DSP)
mon_cl2_state(0)	39	I2S2 (DSP)
mon_cl2_state(1)	3A	I2S2 (DSP)
mon_cl2_state(2)	3B	I2S2 (DSP)
mon_cl1_state(0)	3C	I2S2 (DSP)
mon_cl1_state(1)	3D	I2S2 (DSP)
mon_cl1_state(2)	3E	I2S2 (DSP)
mon_i2s_data_wrq	3F	I2S2 (DSP)
mon_i2s_data_rrq	40	I2S2 (DSP)
mon_tx_state(0)	41	I2S3 (DSP)
mon_tx_state(1)	42	I2S3 (DSP)
mon_tx_state(2)	43	I2S3 (DSP)
mon_tx_state(3)	44	I2S3 (DSP)
mon_cl1_state(0)	45	I2S3 (DSP)
mon_cl1_state(1)	46	I2S3 (DSP)
mon_cl1_state(2)	47	I2S3 (DSP)
mon_i2s_data_rrq	48	I2S3 (DSP)
mon_a53_cipherring_ready	49	CIPHER_A53
mon_a53_start_cipherring	4A	CIPHER_A53
mon_a53_ecry_dcry_data	4B	CIPHER_A53
mon_a53_dcry_valid	4C	CIPHER_A53

**CONFIDENTIAL**

**Port Control Logic**

**Table 11-65 Signals from DSP**

Signal Name	Signal Select (hex)	Remarks
mon_a53_ecry_valid	4D	CIPHER_A53
mon_a53_gclock_dsp	4E	CIPHER_A53
mon_a53_gclock_ram	4F	CIPHER_A53
mon_a53_gclock_cipher	50	CIPHER_A53
mon_dpram_data(0)	51	BB-Transmit
mon_dpram_req	52	BB-Transmit
mon_gclk_dsp_per_mod	53	BB-Transmit
mon_clk_bbtx	54	BB-Transmit
mon_vh_status	55	Viterbi-Channel-Dec.
mon_dec_busy	56	Viterbi-Channel-Dec.
mon_dec_active	57	Viterbi-Channel-Dec.
mon_pointer_dec_new	58	Viterbi-Channel-Dec.
mon_res_dec	59	Viterbi-Channel-Dec.
mon_res_dsp_int	5A	Viterbi-Channel-Dec.
mon_vh_status	5B	Viterbi-Equalizer
mon_eq_busy	5C	Viterbi-Equalizer
mon_pointer_right_hs_new	5D	Viterbi-Equalizer
mon_pointer_left_hs_new	5E	Viterbi-Equalizer
mon_res_eq	5F	Viterbi-Equalizer
mon_res_dsp_int	60	Viterbi-Equalizer
mon_en_reg_read	61	DSP
mon_ppap_page	62	DSP
mon_dxap_page	63	DSP
mon_dspdis	64	DSP
pstatusp(0)	65	OCEM (DSP)
pstatusp(1)	66	OCEM (DSP)
pstatusp(2)	67	OCEM (DSP)
pstatusp(3)	68	OCEM (DSP)
psftp	69	OCEM (DSP)
ppwp	6A	OCEM (DSP)

**CONFIDENTIAL**

**Port Control Logic**

**Table 11-65 Signals from DSP**

Signal Name	Signal Select (hex)	Remarks
pprp	6B	OCEM (DSP)
Reserved	6C	
Reserved	6D	
pdummp	6E	OCEM (DSP)
ddtvm	6F	OCEM (DSP)
odebugp	70	OCEM (DSP)
obootp	71	OCEM (DSP)
oabortn	72	OCEM (DSP)
btrapreqp	73	OCEM (DSP)
mon_rst_mix_n	74	DSP
mon_rst_dsp_n	75	DSP
mon_rst_dsp_per_n	76	DSP
mon_rst_dsp_mem_n	77	DSP
seib_j_tdo_p	78	SEIB (DSP)
seib_j_abort_iut_n	79	SEIB (DSP)
waitp	7A	SEIB (DSP)
enocemn	7B	SEIB (DSP)
state_kgcore(0)	7C	CIPHER_A53
state_kgcore(1)	7D	CIPHER_A53
state_kgcore(2)	7E	CIPHER_A53
state_kgcore(3)	7F	CIPHER_A53

**CONFIDENTIAL**

**Port Control Logic**

### 11.8.10.2.3 Cipher A53

**Table 11-66 Signals from A53 Peripherals**

Signal Name	Signal Select (hex)	Remarks
mon_init_kgcore	01	Cipher_A53
mon_clear_init	02	Cipher_A53
mon_state_ctrl(0)	03	Cipher_A53
mon_state_ctrl(1)	04	Cipher_A53
mon_state_ctrl(2)	05	Cipher_A53
mon_state_kgcore(0)	06	Cipher_A53
mon_state_kgcore(1)	07	Cipher_A53
mon_state_kgcore(2)	08	Cipher_A53
mon_state_kgcore(3)	09	Cipher_A53
mon_block_count(0)	0A	Cipher_A53
mon_block_count(1)	0B	Cipher_A53
mon_block_count(2)	0C	Cipher_A53
mon_block_count(3)	0D	Cipher_A53
mon_register_A_value(0)	0E	Cipher_A53
mon_register_A_value(1)	0F	Cipher_A53
mon_register_A_value(2)	10	Cipher_A53
mon_register_A_value(3)	11	Cipher_A53
mon_register_A_value(4)	12	Cipher_A53
mon_register_A_value(5)	13	Cipher_A53
mon_register_A_value(6)	14	Cipher_A53
mon_register_A_value(7)	15	Cipher_A53
mon_register_A_value(8)	16	Cipher_A53
mon_register_A_value(9)	17	Cipher_A53
mon_register_A_value(10)	18	Cipher_A53
mon_register_A_value(11)	19	Cipher_A53
mon_register_A_value(12)	1A	Cipher_A53
mon_register_A_value(13)	1B	Cipher_A53
mon_register_A_value(14)	1C	Cipher_A53
mon_register_A_value(15)	1D	Cipher_A53

**CONFIDENTIAL**

**Port Control Logic**

**Table 11-66 Signals from A53 Peripherals**

Signal Name	Signal Select (hex)	Remarks
mon_register_A_value(16)	1E	Cipher_A53
mon_register_A_value(17)	1F	Cipher_A53
mon_register_A_value(18)	20	Cipher_A53
mon_register_A_value(19)	21	Cipher_A53
mon_register_A_value(20)	22	Cipher_A53
mon_register_A_value(21)	23	Cipher_A53
mon_register_A_value(22)	24	Cipher_A53
mon_register_A_value(23)	25	Cipher_A53
mon_register_A_value(24)	26	Cipher_A53
mon_register_A_value(25)	27	Cipher_A53
mon_register_A_value(26)	28	Cipher_A53
mon_register_A_value(27)	29	Cipher_A53
mon_register_A_value(28)	2A	Cipher_A53
mon_register_A_value(29)	2B	Cipher_A53
mon_register_A_value(30)	2C	Cipher_A53
mon_register_A_value(31)	2D	Cipher_A53
mon_register_A_value(32)	2E	Cipher_A53
mon_register_A_value(33)	2F	Cipher_A53
mon_register_A_value(34)	30	Cipher_A53
mon_register_A_value(35)	31	Cipher_A53
mon_register_A_value(36)	32	Cipher_A53
mon_register_A_value(37)	33	Cipher_A53
mon_register_A_value(38)	34	Cipher_A53
mon_register_A_value(39)	35	Cipher_A53
mon_register_A_value(40)	36	Cipher_A53
mon_register_A_value(41)	37	Cipher_A53
mon_register_A_value(42)	38	Cipher_A53
mon_register_A_value(43)	39	Cipher_A53
mon_register_A_value(44)	3A	Cipher_A53
mon_register_A_value(45)	3B	Cipher_A53



**CONFIDENTIAL**

**Port Control Logic**

**Table 11-66 Signals from A53 Peripherals**

Signal Name	Signal Select (hex)	Remarks
mon_register_A_value(46)	3C	Cipher_A53
mon_register_A_value(47)	3D	Cipher_A53
mon_register_A_value(48)	3E	Cipher_A53
mon_register_A_value(49)	3F	Cipher_A53
mon_register_A_value(50)	40	Cipher_A53
mon_register_A_value(51)	41	Cipher_A53
mon_register_A_value(52)	42	Cipher_A53
mon_register_A_value(53)	43	Cipher_A53
mon_register_A_value(54)	44	Cipher_A53
mon_register_A_value(55)	45	Cipher_A53
mon_register_A_value(56)	46	Cipher_A53
mon_register_A_value(57)	47	Cipher_A53
mon_register_A_value(58)	48	Cipher_A53
mon_register_A_value(59)	49	Cipher_A53
mon_register_A_value(60)	4A	Cipher_A53
mon_register_A_value(61)	4B	Cipher_A53
mon_register_A_value(62)	4C	Cipher_A53
mon_register_A_value(63)	4D	Cipher_A53
mon_kgcore_key_stored(0)	4E	Cipher_A53
mon_kgcore_key_stored(1)	4F	Cipher_A53
mon_kgcore_key_stored(2)	50	Cipher_A53
mon_kgcore_key_stored(3)	51	Cipher_A53
mon_kgcore_key_stored(4)	52	Cipher_A53
mon_kgcore_key_stored(5)	53	Cipher_A53
mon_kgcore_key_stored(6)	54	Cipher_A53
mon_kgcore_key_stored(7)	55	Cipher_A53
mon_kgcore_key_stored(8)	56	Cipher_A53
mon_kgcore_key_stored(9)	57	Cipher_A53
mon_kgcore_key_stored(10)	58	Cipher_A53
mon_kgcore_key_stored(11)	59	Cipher_A53

**CONFIDENTIAL**

**Port Control Logic**

**Table 11-66 Signals from A53 Peripherals**

<b>Signal Name</b>	<b>Signal Select (hex)</b>	<b>Remarks</b>
mon_kgcore_key_stored(12)	5A	Cipher_A53
mon_kgcore_key_stored(13)	5B	Cipher_A53
mon_kgcore_key_stored(14)	5C	Cipher_A53
mon_kgcore_key_stored(15)	5D	Cipher_A53
mon_kgcore_key_stored(16)	5E	Cipher_A53
mon_kgcore_key_stored(17)	5F	Cipher_A53
mon_kgcore_key_stored(18)	60	Cipher_A53
mon_kgcore_key_stored(19)	61	Cipher_A53
mon_kgcore_key_stored(20)	62	Cipher_A53
mon_kgcore_key_stored(21)	63	Cipher_A53
mon_kgcore_key_stored(22)	64	Cipher_A53
mon_kgcore_key_stored(23)	65	Cipher_A53
mon_kgcore_key_stored(24)	66	Cipher_A53
mon_kgcore_key_stored(25)	67	Cipher_A53
mon_kgcore_key_stored(26)	68	Cipher_A53
mon_kgcore_key_stored(27)	69	Cipher_A53
mon_kgcore_key_stored(28)	6A	Cipher_A53
mon_kgcore_key_stored(29)	6B	Cipher_A53
mon_kgcore_key_stored(30)	6C	Cipher_A53
mon_kgcore_key_stored(31)	6D	Cipher_A53
mon_kgcore_key_stored(32)	6E	Cipher_A53
mon_kgcore_key_stored(33)	6F	Cipher_A53
mon_kgcore_key_stored(34)	70	Cipher_A53
mon_kgcore_key_stored(35)	71	Cipher_A53
mon_kgcore_key_stored(36)	72	Cipher_A53
mon_kgcore_key_stored(37)	73	Cipher_A53
mon_kgcore_key_stored(38)	74	Cipher_A53
mon_kgcore_key_stored(39)	75	Cipher_A53
mon_kgcore_key_stored(40)	76	Cipher_A53
mon_kgcore_key_stored(41)	77	Cipher_A53

**CONFIDENTIAL**

**Port Control Logic**

**Table 11-66 Signals from A53 Peripherals**

Signal Name	Signal Select (hex)	Remarks
mon_kgcore_key_stored(42)	78	Cipher_A53
mon_kgcore_key_stored(43)	79	Cipher_A53
mon_kgcore_key_stored(44)	7A	Cipher_A53
mon_kgcore_key_stored(45)	7B	Cipher_A53
mon_kgcore_key_stored(46)	7C	Cipher_A53
mon_kgcore_key_stored(47)	7D	Cipher_A53
mon_ecry_dcry_add	7E	Cipher_A53
-	7F	Reserved

**CONFIDENTIAL**

**Port Control Logic**

### 11.8.10.2.4 CGU

**Table 11-67 Signals from the CGU**

Signal Name	Signal Select (hex)	Remarks
cgu_rtc_clk_en	01	RTC clock enable <sup>1)</sup>
cgu_capcom2_clk_en	02	CAPCOM2 clock enable <sup>1)</sup>
cgu_capcom1_clk_en	03	CAPCOM1 clock enable <sup>1)</sup>
cgu_ssc_clk_en	04	SSC clock enable <sup>1)</sup>
cgu_asc1_clk_en	05	ASC1 clock enable <sup>1)</sup>
cgu_asc0_clk_en	06	ASC0 clock enable <sup>1)</sup>
cgu_master_access	07	Master Access clock enable <sup>1)</sup>
cgu_xper_clk_en	08	X-Bus Peripheral clock enable <sup>1)</sup>
cgu_xbus_clk_en	09	X-Bus clock enable <sup>1)</sup>
cgu_pdbus_clk_en	0A	PD-Bus clock enable <sup>1)</sup>
cgu_clk_ms	0B	Mixed Signal Clock. The output frequency is divided by 2.
cgu_clk_dsp	0C	DSP Clock. The output frequency is divided by 2.
cgu_clk_afc	0D	AFC Clock. The output frequency is divided by 2.
cgu_clk_mem	0E	Local Memory Bus Clock. The output frequency is divided by 2.
cgu_clk_x	0F	X-Bus Clock. The output frequency is divided by 2.
cgu_clk_pd	10	PD-Bus Clock. The output frequency is divided by 2.
cgu_c166_clk_neg	11	C166S Negative Clock. The output frequency is divided by 2.
cgu_c166_clk_pos2	12	C166S Positive 2 Clock. The output frequency is divided by 2.
cgu_c166_clk_pos1	13	C166S Positive 1 Clock. The output frequency is divided by 2.
cgu_clk_master	14	Master Access Clock. The output frequency is divided by 2.

**CONFIDENTIAL**

**Port Control Logic**

**Table 11-67 Signals from the CGU**

Signal Name	Signal Select (hex)	Remarks
pll_clk_phs1	15	Phase Shifter 1 Clock. The output frequency is divided by 2.
pll_clk_pll	16	PLL Clock. The output frequency is divided by 2.
pll_clk_in	17	Shaper Output clock. The output frequency is divided by 2.
rtc_clk_32k	18	32kHz clock from RTC. The output frequency is divided by 2.
sccu_resd	19	SCCU
sccu_resa	1A	SCCU
slp_vcxo_off	1B	SCCU
pre_wakeup	1C	SCCU
hwwup	1D	SCCU
sccu_vcxo_en_del	1E	SCCU
fr_dsp_out	1F	SCCU
fr_ana_out	20	SCCU
-	21	Reserved
slprst	22	Sleep counter reset
reset_32	23	Power on reset for clk_32
reset_13	24	Power on reset for clk_13
-	25	Reserved
-	26	Reserved
res_ucslp	27	SCCU
res_ucwup	28	SCCU
clk_cpu_en_del	29	SCCU
clk_gsm_on	2A	SCCU
sccu_shap_en_del	2B	SCCU
-	2C	Reserved
tststate13m(0)	2D	13mHz state machine for sleep function
tststate13m(1)	2E	13mHz state machine for sleep function
tststate32k(0)	2F	32 kHz state machine for sleep function

**CONFIDENTIAL**

**Port Control Logic**

**Table 11-67 Signals from the CGU**

Signal Name	Signal Select (hex)	Remarks
tsstate32k(1)	30	32 kHz state machine for sleep function
status(0)	31	Main state machine status
status(1)	32	Main state machine status
status(2)	33	Main state machine status
status(3)	34	Main state machine status
status(4)	35	Main state machine status
prewup_int_o	36	Pre-wakeup interrupt
pll_clk_phs2	37	For second phase shifter output clock

<sup>1)</sup> The output clock is the result of the logical AND combination from the source clock in the clock enable signal. With a enable clock equal to 1 (logical value) the output clock will be equal to the source clock, that is, the visible enable signal values:

- If output clk is equal to source clk, monitor signal is set to constant '1'
- If output clk is half of source clk, monitor signal is set to '1' every second source clk, else set to '0'
- etc.

**CONFIDENTIAL**

**Port Control Logic**

### 11.8.10.2.5 PADS

**Table 11-68 Pad Signals**

Signal Name	Signal Select (hex)	Remarks
KP0	01	Keypad
KP2	02	Keypad
KP3	03	Keypad
KP4	04	Keypad
KP5	05	Keypad
KP6	06	Keypad
KP7	07	Keypad
KP8	08	Keypad
KP9	09	Keypad
RXD	0A	ASC0
TXD	0B	ASC0
ASC_RTS_n	0C	ASC0
ASC_CTS_n	0D	ASC0
SSC0_CLK	0E	PD SSC
SSC0MRST	0F	PD SSC
SSC0_MTSR	10	PD SSC
SCL	11	I2C
SDA	12	I2C
CCVZ_n	13	SIM Card I/F
CCIN	14	SIM Card I/F
I2S1_CLK0	15	I2S1: DAI-PCM
I2S1_RX	16	I2S1: DAI-PCM
I2S1_TX	17	I2S1: DAI-PCM
I2S1_WA0	18	I2S1: DAI-PCM
I2S2_CLK0	19	I2S2 : Dig. HIQ Audio I/F
I2S2_RX	1A	I2S2 : Dig. HIQ Audio I/F
I2S2_TX	1B	I2S2 : Dig. HIQ Audio I/F
I2S2_WA0	1C	I2S2 : Dig. HIQ Audio I/F
I2S3_CLK	1D	I2S3 : Dig. BB I/F

**CONFIDENTIAL**

**Port Control Logic**

**Table 11-68 Pad Signals**

Signal Name	Signal Select (hex)	Remarks
I2S3_TX	1E	I2S3 : Dig. BB I/F
I2S3_WA	1F	I2S3 : Dig. BB I/F
A22	20	External Bus I/F
CS1_n	21	External Bus I/F
CS3_n	22	External Bus I/F
BHE_n	23	External Bus I/F
ADV	24	External Bus I/F
OE_N	25	External Bus I/F
RFSTR1	26	RF Control Unit
RFSTR2	27	RF Control Unit
RFSTR3	28	RF Control Unit
T_OUT1	29	Gsm TDMA Timer
T_OUT2	2A	Gsm TDMA Timer
T_OUT3	2B	Gsm TDMA Timer
T_OUT4	2C	Gsm TDMA Timer
T_OUT5	2D	Gsm TDMA Timer
T_OUT7	2E	Gsm TDMA Timer
T_OUT8	2F	Gsm TDMA Timer
T_OUT10	30	Gsm TDMA Timer
T_OUT11	31	Gsm TDMA Timer
CLKOUT	32	Other Functional Pins: Clocks and control
RSTOUT_n	33	Other Functional Pins: Clocks and control
VCXO_EN	34	Other Functional Pins: Clocks and control
WAKEUP	35	Other Functional Pins: Clocks and control
SSC1_CLK	36	Other Functional Pins: Clocks and control
SSC1_MTRS	37	Other Functional Pins: Clocks and control



**CONFIDENTIAL**

**Port Control Logic**

**Table 11-68 Pad Signals**

Signal Name	Signal Select (hex)	Remarks
SSC1_MRST	38	Other Functional Pins: Clocks and control
CC00IO	39	Other Functional Pins: Clocks and control
LPAOUT0	3A	Other Functional Pins: Clocks and control
CCIO	3B	SIM Card I/F
CCLK	3C	SIM Card I/F
CCRST	3D	SIM Card I/F
RFSTR0	3E	RF Control Unit
RFDATA	3F	RF Control Unit
RFCLK	40	RF Control Unit
T_OUT0	41	GSM TDMA Timer
AFC	42	Other Functional Pins: Clocks and control
D0	43	External Bus I/F
D1	44	External Bus I/F
D2	45	External Bus I/F
D3	46	External Bus I/F
D4	47	External Bus I/F
D5	48	External Bus I/F
D6	49	External Bus I/F
D7	4A	External Bus I/F
D8	4B	External Bus I/F
D9	4C	External Bus I/F
D10	4D	External Bus I/F
D11	4E	External Bus I/F
D12	4F	External Bus I/F
D13	50	External Bus I/F
D14	51	External Bus I/F
D15	52	External Bus I/F

**CONFIDENTIAL**

**Port Control Logic**

**Table 11-68 Pad Signals**

Signal Name	Signal Select (hex)	Remarks
A0	53	External Bus I/F
A1	54	External Bus I/F
A2	55	External Bus I/F
A3	56	External Bus I/F
A4	57	External Bus I/F
A5	58	External Bus I/F
A6	59	External Bus I/F
A7	5A	External Bus I/F
A8	5B	External Bus I/F
A9	5C	External Bus I/F
A10	5D	External Bus I/F
A11	5E	External Bus I/F
A12	5F	External Bus I/F
A13	60	External Bus I/F
A14	61	External Bus I/F
A15	62	External Bus I/F
A16	63	External Bus I/F
A17	64	External Bus I/F
A18	65	External Bus I/F
A19	66	External Bus I/F
A20	67	External Bus I/F
A21	68	External Bus I/F
CS0_n	69	External Bus I/F
RD_n	6A	External Bus I/F
WR_n	6B	External Bus I/F
A22	6C	External Bus I/F

**CONFIDENTIAL**

**Port Control Logic**

### 11.8.10.2.6 INTR\_EXT

**Table 11-69 INTR\_EXT Signals**

Signal Name	Signal Select (hex)	Remarks
CC0_INT	01	CAPCOM-Register-0
CC1_INT	02	CAPCOM-Register-1
CC2_INT	03	CAPCOM-Register-2
CC3_INT	04	CAPCOM-Register-3
CC4_INT	05	CAPCOM-Register-4
CC5_INT	06	CAPCOM-Register-5
CC6_INT	07	CAPCOM-Register-6
CC7_INT	08	CAPCOM-Register-7
CC16_INT	09	CAPCOM-register-16
CC17_INT	0A	CAPCOM-register-17
CC18_INT	0B	CAPCOM-register-18
CC19_INT	0C	CAPCOM-register-19
CC20_INT	0D	CAPCOM-register-20
CC21_INT	0E	CAPCOM-register-21
CC22_INT	0F	CAPCOM-register-22
CC23_INT	10	CAPCOM-register-23
T0_INT	11	CAPCOM-Timer-0
T1_INT	12	CAPCOM-Timer-1
T2_INT	13	GPT1-Timer-2
T3_INT	14	GPT1-Timer-3
T4_INT	15	GPT1-Timer-4
T5_INT	16	GPT2-Timer-5
T6_INT	17	GPT2-Timer-6
T7_INT	18	CAPCOM-Timer-7
T8_INT	19	CAPCOM-Timer-8
CR_INT	1A	GPT2-CAPREL-Register
RTC_INT	1B	from RTC
RTC_T14_INT	1C	from RTC
S0T_INT	1D	ASC0-Transmit

**CONFIDENTIAL**

**Port Control Logic**

**Table 11-69 INTR\_EXT Signals**

Signal Name	Signal Select (hex)	Remarks
S0R_INT	1E	ASC0-Receive
S0E_INT	1F	ASC0-Error
S0TB_INT	20	ASC0-Transmit-Buffer
S0ABST_INT	21	ASC0-Autobaud-Start
S0ABDET_INT	22	ASC0-Autobaud-Detect
S0CTS_INT	23	ASC0-Cts
S0TM0_INT	24	ASC0-Tm0
S1T_INT	25	ASC1-Transmit
S1R_INT	26	ASC1-Receive
S1E_INT	27	ASC1-Error
S1TB_INT	28	ASC1-Transmit-Buffer
SSCT_INT	29	SSC-Transmit
SSCR_INT	2A	SSC-Receive
SSCE_INT	2B	SSC-Error
RES1_INT	2C	Reserved
RES2_INT	2D	Reserved
RES3_INT	2E	Reserved
KPD_INT	2F	Keypad-interrupt
ECO_INT	30	Power-Management
S0AS_INT	31	uC-int from autostart
T_INT1	32	T_INT1-of-GSM-Timer
T_INT2	33	T_INT2-of-GSM-Timer
TIM0_INT	34	INT_GP(0)-of-GSM-Timer
TIM1_INT	35	INT_GP(1)-of-GSM-Timer
TIM2_INT	36	INT_GP(2)-of-GSM-Timer
TIM3_INT	37	INT_GP(3)-of-GSM-Timer
TIM4_INT	38	INT_GP(4)-of-GSM-Timer
MEAS0_INT	39	Measurement interrupt
MEAS1_INT	3A	Measurement interrupt
MEAS0_TOGGLE_INT	3B	Measurement int. toggled

**CONFIDENTIAL**

**Port Control Logic**

**Table 11-69 INTR\_EXT Signals**

Signal Name	Signal Select (hex)	Remarks
MEAS1_TOGGLE_INT	3C	Measurement int. toggled
RFSSOT_INT	3D	RFSSSTINT-of-GSM-Timer
I2CD_INT	3E	I2C data
IIC_P_INT	3F	I2C protocol
IIC_E_INT	40	I2C error
SIMOK_INT	41	SIM Card-Interface-OK
SIMERR_INT	42	SIM Card-Interface-BAD
SIMIN_INT	43	SIM Card-in-(pad CCIN => from SIMCARD)
EX0_INT	44	External-Interrupt-0 and/or s0rint
EX1_INT	45	External-Interrupt-1 and/or siminint
EX2_INT	46	External-Int-2 and/or kp dint
EX3_INT	47	External-Interrupt-3 and/or rtc_int
EX4_INT	48	External-Interrupt-4 and/or ex4bint
EX5_INT	49	External-Interrupt-5 and/or ex5bint
EX6_INT	4A	External-Int-6 and/or tim2int
EX7_INT	4B	External-Int-7 and/or t3int
FROM_DSP_TO_MCU0_INT	4C	DSP-Interrupt-0
FROM_DSP_TO_MCU1_INT	4D	DSP-Interrupt-1
FROM_DSP_TO_MCU2_INT	4E	DSP-Interrupt-2
FROM_DSP_TO_MCU3_INT	4F	DSP-Interrupt-3
PECLIN_INT	50	PEC-link
EX4B_INT	51	EX4B
EX5B_INT	52	EX5B
GPRS_INT	53	GPRS Interrupt

**CONFIDENTIAL**

**Port Control Logic**

### 11.8.10.2.7 PCL\_PER

**Table 11-70 PCL\_PER Signals**

Signal Name	Signal Select (hex)	Description
READY_N_O	01	PMCU
RD_N_O	02	PMCU
OE_N_O	03	PMCU
CS0_N_O	04	PMCU
CSWITCH_N_I	05	PMCU
CS1_N_O	05	PMCU
ADV_N_O	06	PMCU
RTC0	07	RTC0 interrupt request
RTC1	08	RTC1 interrupt request
RTC2	09	RTC2 interrupt request
RTC3	0A	RTC3 interrupt request
RTCALARM	0B	RTC alarm interrupt request
RTCBAD	0C	RTC bad signal
RTC_CLK	0D	RTC kernel clock through toggle

### 11.8.10.2.8 GPRS\_GEA3

**Table 11-71 GPRS\_GEA3 Signals**

Signal Name	Signal Select (hex)	Remarks
monitortate_kgcore(0)	01	
monitortate_kgcore(1)	02	
monitortate_kgcore(2)	03	
monitortate_kgcore(3)	04	
monitor_cipher_out_bit	05	
monitor_cipher_in_bit	06	
monitor_init_kgcore	07	
monitor_clear_init	08	
monitortate_ctrl(0)	09	
monitortate_ctrl(1)	0A	
monitortate_ctrl(2)	0B	
select_gea_i	0C	

**CONFIDENTIAL**

**Port Control Logic**

**Table 11-71 GPRS\_GEA3 Signals**

Signal Name	Signal Select (hex)	Remarks
monitor_block_count(0)	0D	
monitor_block_count(1)	0E	
monitor_block_count(2)	0F	
monitor_block_count(3)	10	
monitor_A_value(0)	11	
monitor_A_value(1)	12	
monitor_A_value(2)	13	
monitor_A_value(3)	14	
monitor_A_value(4)	15	
monitor_A_value(5)	16	
monitor_A_value(6)	17	
monitor_A_value(7)	18	
monitor_A_value(8)	19	
monitor_A_value(9)	1A	
monitor_A_value(10)	1B	
monitor_A_value(11)	1C	
monitor_A_value(12)	1D	
monitor_A_value(13)	1E	
monitor_A_value(14)	1F	
monitor_A_value(15)	20	
monitor_A_value(16)	21	
monitor_A_value(17)	22	
monitor_A_value(18)	23	
monitor_A_value(19)	24	
monitor_A_value(20)	25	
monitor_A_value(21)	26	
monitor_A_value(22)	27	
monitor_A_value(23)	28	
monitor_A_value(24)	29	
monitor_A_value(25)	2A	

**CONFIDENTIAL**

**Port Control Logic**

**Table 11-71 GPRS\_GEA3 Signals**

Signal Name	Signal Select (hex)	Remarks
monitor_A_value(26)	2B	
monitor_A_value(27)	2C	
monitor_A_value(28)	2D	
monitor_A_value(29)	2E	
monitor_A_value(30)	2F	
monitor_A_value(31)	30	
monitor_A_value(32)	31	
monitor_A_value(33)	32	
monitor_A_value(34)	33	
monitor_A_value(35)	34	
monitor_A_value(36)	35	
monitor_A_value(37)	36	
monitor_A_value(38)	37	
monitor_A_value(39)	38	
monitor_A_value(40)	39	
monitor_A_value(41)	3A	
monitor_A_value(42)	3B	
monitor_A_value(43)	3C	
monitor_A_value(44)	3D	
monitor_A_value(45)	3E	
monitor_A_value(46)	3F	
monitor_A_value(47)	40	
monitor_A_value(48)	41	
monitor_A_value(49)	42	
monitor_A_value(50)	43	
monitor_A_value(51)	44	
monitor_A_value(52)	45	
monitor_A_value(53)	46	
monitor_A_value(54)	47	
monitor_A_value(55)	48	



**CONFIDENTIAL**

**Port Control Logic**

**Table 11-71 GPRS\_GEA3 Signals**

Signal Name	Signal Select (hex)	Remarks
monitor_A_value(56)	49	
monitor_A_value(57)	4A	
monitor_A_value(58)	4B	
monitor_A_value(59)	4C	
monitor_A_value(60)	4D	
monitor_A_value(61)	4E	
monitor_A_value(62)	4F	
monitor_A_value(63)	50	
monitor_kgcore_keytored(0)	51	
monitor_kgcore_keytored(1)	52	
monitor_kgcore_keytored(2)	53	
monitor_kgcore_keytored(3)	54	
monitor_kgcore_keytored(4)	55	
monitor_kgcore_keytored(5)	56	
monitor_kgcore_keytored(6)	57	
monitor_kgcore_keytored(7)	58	
monitor_kgcore_keytored(8)	59	
monitor_kgcore_keytored(9)	5A	
monitor_kgcore_keytored(10)	5B	
monitor_kgcore_keytored(11)	5C	
monitor_kgcore_keytored(12)	5D	
monitor_kgcore_keytored(13)	5E	
monitor_kgcore_keytored(14)	5F	
monitor_kgcore_keytored(15)	60	
monitor_kgcore_keytored(16)	61	
monitor_kgcore_keytored(17)	62	
monitor_kgcore_keytored(18)	63	
monitor_kgcore_keytored(19)	64	
monitor_kgcore_keytored(20)	65	
monitor_kgcore_keytored(21)	66	

**CONFIDENTIAL**

**Port Control Logic**

**Table 11-71 GPRS\_GEA3 Signals**

Signal Name	Signal Select (hex)	Remarks
monitor_kgcore_keytored(22)	67	
monitor_kgcore_keytored(23)	68	
monitor_kgcore_keytored(24)	69	
monitor_kgcore_keytored(25)	6A	
monitor_kgcore_keytored(26)	6B	
monitor_kgcore_keytored(27)	6C	
monitor_kgcore_keytored(28)	6D	
monitor_kgcore_keytored(29)	6E	
monitor_kgcore_keytored(30)	6F	
monitor_kgcore_keytored(31)	70	
monitor_kgcore_keytored(32)	71	
monitor_kgcore_keytored(33)	72	
monitor_kgcore_keytored(34)	73	
monitor_kgcore_keytored(35)	74	
monitor_kgcore_keytored(36)	75	
monitor_kgcore_keytored(37)	76	
monitor_kgcore_keytored(38)	77	
monitor_kgcore_keytored(39)	78	
monitor_kgcore_keytored(40)	79	
monitor_kgcore_keytored(41)	7A	
monitor_kgcore_keytored(42)	7B	
monitor_kgcore_keytored(43)	7C	
monitor_kgcore_keytored(44)	7D	
monitor_kgcore_keytored(45)	7E	
monitor_kgcore_keytored(46)	7F	

**CONFIDENTIAL**

**Port Control Logic**

### 11.8.10.2.9 MCU\_PER

**Table 11-72 MCU\_PER Signals**

Signal Name	Signal Select (hex)	Remarks
gsm_dsp_int_gp(0)	01	to DSP
gsm_dsp_int_gp(1)	02	to DSP
gsm_mon_pasw (Not used in E-GOLDradio)	03	GSM PORTS
gsm_mon_iqramp <sup>1)</sup>	04	GSM PORTS
gsm_mon_par_clr_int_fsm	05	GSM PORTS
gsm_mon_par_dac_clk_n	06	GSM PORTS
gsm_mon_par_dac(0)	07	GSM PORTS
gsm_mon_par_dac(1)	08	GSM PORTS
gsm_mon_par_dac(2)	09	GSM PORTS
gsm_mon_par_dac(3)	0A	GSM PORTS
gsm_mon_par_dac(4)	0B	GSM PORTS
gsm_mon_par_dac(5)	0C	GSM PORTS
gsm_mon_par_dac(6)	0D	GSM PORTS
gsm_mon_par_dac(7)	0E	GSM PORTS
gsm_mon_par_dac(8)	0F	GSM PORTS
gsm_mon_par_dac(9)	10	GSM PORTS
gsm_mon_par_dac(10)	11	GSM PORTS
gsm_mon_dcpa_datavalid	12	GSM PORTS
gsm_mon_dcpa_data(0)	13	GSM PORTS
gsm_mon_dcpa_data(1)	14	GSM PORTS
gsm_mon_dcpa_data(2)	15	GSM PORTS
gsm_mon_dcpa_data(3)	16	GSM PORTS
gsm_mon_dcpa_data(4)	17	GSM PORTS
gsm_mon_dcpa_data(5)	18	GSM PORTS
gsm_mon_dcpa_data(6)	19	GSM PORTS
gsm_mon_dcpa_data(7)	1A	GSM PORTS
gsm_mon_dcpa_data(8)	1B	GSM PORTS
gsm_mon_dcpa_data(9)	1C	GSM PORTS

**CONFIDENTIAL**

**Port Control Logic**

**Table 11-72 MCU\_PER Signals**

Signal Name	Signal Select (hex)	Remarks
gsm_mon_dcpa_data(10)	1D	GSM PORTS
gsm_mon_adctrig	1E	GSM PORTS
gsm_dsp_eqon	1F	GSM PORTS
gsm_dsp_monon	20	GSM PORTS
gsm_dsp_scon	21	GSM PORTS
gsm_dsp_fcon	22	GSM PORTS
gsm_dsp_rxon	23	GSM PORTS
gsm_dsp-ana_txon	24	GSM PORTS
gsm_dsp_codon	25	GSM PORTS
gsm_eco_slpstart	26	GSM PORTS
meas_ana-pcb_clk_meas	27	MEAS PORTS
meas_ana-pcb_soc	28	MEAS PORTS
ana_meas-pcl_eoc	29	MEAS PORTS
ana_meas-pcb_penirq_n	2A	MEAS PORTS
com_mon(0)	2B	SM MCU Set Comm. Flag, Bit 2
com_mon(1)	2C	SM MCU Reset Comm. Flag, Bit 2
com_mon(2)	2D	SM DSP Set Comm. Flag, Bit 2
com_mon(3)	2E	SM DSP Reset Comm. Flag, Bit 2
com_mon(4)	2F	SM MCU Comm. Flag Status, Bit 2
sem_mon(0)	30	SM DSP Semaphore Register, Bit 4
sem_mon(1)	31	SM MCU Semaphore Register, Bit 4
sem_mon(2)	32	SM DSP Semaphore Read Data, Bit 4
sem_mon(3)	33	SM MCU Semaphore Read Data, Bit 4
biu_mon(0)	34	BIU
biu_mon(1)	35	BIU
biu_mon(2)	36	BIU
biu_mon(3)	37	BIU
biu_mon(4)	38	BIU
biu_mon(5)	39	BIU
biu_mon(6)	3A	BIU

**CONFIDENTIAL**

**Port Control Logic**

**Table 11-72 MCU\_PER Signals**

Signal Name	Signal Select (hex)	Remarks
biu_mon(7)	3B	BIU
pcl_sim_ccin	3C	SIM PORTS
sim_pcl_ccrst	3D	SIM PORTS
sim_pcl_ccclk	3E	SIM PORTS
sim_pcl_ccvz_n	3F	SIM PORTS
sim_pcl_cciosw	40	SIM PORTS
ssc_ms_out	41	SSC SIGNALS
ssc_sh_clk	42	SSC SIGNALS
ssc_sl_out	43	SSC SIGNALS
asc0_sel_n	44	ASC0 SIGNALS
asc0_ex_hw_disreq	45	ASC0 SIGNALS
asc1_cgu_sel_n	46	ASC1 SIGNALS
ram_mon(0)	47	SHARED RAM
ram_mon(1)	48	SHARED RAM
ram_mon(2)	49	SHARED RAM
ram_mon(3)	4A	SHARED RAM
ram_mon(4)	4B	SHARED RAM
ram_mon(5)	4C	SHARED RAM
ram_mon(6)	4D	SHARED RAM
ram_mon(7)	4E	SHARED RAM
ram_mon(8)	4F	SHARED RAM
ram_mon(9)	50	SHARED RAM
ram_mon(10)	51	SHARED RAM
ram_mon(11)	52	SHARED RAM
ram_mon(12)	53	SHARED RAM
ram_mon(13)	54	SHARED RAM
ram_mon(14)	55	SHARED RAM
ram_mon(15)	56	SHARED RAM
ram_mon(16)	57	SHARED RAM
ram_mon(17)	58	SHARED RAM

**CONFIDENTIAL**

**Port Control Logic**

**Table 11-72 MCU\_PER Signals**

Signal Name	Signal Select (hex)	Remarks
ram_mon(18)	59	SHARED RAM
ram_mon(19)	5A	SHARED RAM
ram_mon(20)	5B	SHARED RAM
ram_mon(21)	5C	SHARED RAM
ram_mon(22)	5D	SHARED RAM
ram_mon(23)	5E	SHARED RAM
ram_mon(24)	5F	SHARED RAM
ram_mon(25)	60	SHARED RAM
ram_mon(26)	61	SHARED RAM
ram_mon(27)	62	SHARED RAM
ram_mon(28)	63	SHARED RAM
ram_mon(29)	64	SHARED RAM
ram_mon(30)	65	SHARED RAM
ram_mon(31)	66	SHARED RAM
ram_mon(32)	67	SHARED RAM
ram_mon(33)	68	SHARED RAM
ram_mon(34)	69	SHARED RAM
ram_mon(35)	6A	SHARED RAM

<sup>1)</sup> The TRIG[25] signal is the gsm\_mon\_iqramp signal, it is an internal monitoring signal of MCU\_PER.

**CONFIDENTIAL**

**TAP Controller and Break Switch**

## 11.9 TAP Controller and Break Switch

History	
Current version: Rev. 1.06, 2005-11-04	
Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 0.02	
<b>Page 1258</b>	Added <b>Section 11.9.3.1.6 RF Test Modes</b>
<b>Page 1251</b>	Updated <b>Table 11-73 JTAG Instructions</b>
<b>Page 1243</b>	Updated Chip external signals in <b>System Integration:</b>
Changes for Rev. 1.01	
<b>Page 1243</b>	Updated <b>System Integration:</b> WS00006682
Changes for Rev. 1.03	
<b>Page 1255</b>	Updated text for <b>ID Register</b>
Changes for Rev. 1.06	

### System Integration:

- Supply domain: VDD\_MAIN
- Chip internal interfaces:
  - Break Switch clock domain: Refer to **Section 10.4.1.3 Sub-System Clocks and Enables (on Page 661)** and see **Figure 10-10 Clock Enable (on Page 662)**.
  - JTAG clock domain: tck from the external port  
cgu\_clk\_x\_o and clk\_dsp\_o for scan use
  - Bus domain: PD-Bus, JTAG  
PD-Bus interface for break\_switch module
  - Interrupt sources: No
- Chip external signals:  
TCK, TMS, TDI, TDO, TRST\_n, TRIG\_IN, TRIG\_OUT  
RF\_CLK, RF\_DATA and RF\_STR[0:3]
- Monitor Pins: Refer to **Section 11.8.10 Internal Signal Monitoring (on Page 1209)**.

### 11.9.1 Introduction

The TCU Test Control Unit includes the Primary Tap controller, the **Break Switch** for multicore debug, and some **JTAG** data registers for test and debug purposes.

### 11.9.2 Break Switch

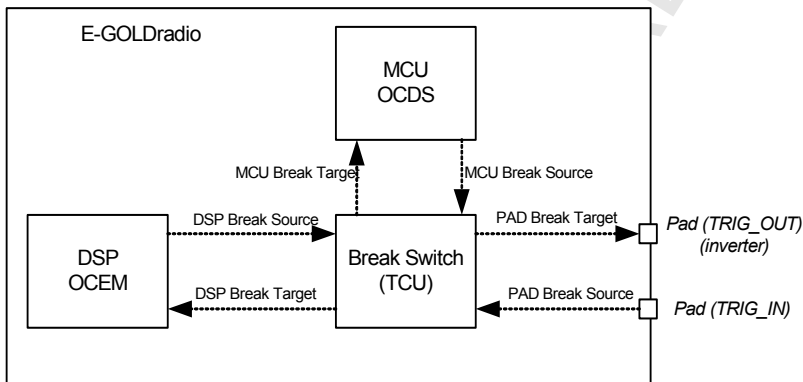
The Break Switch is used to perform the multicore debug; it is linked to the **OCEM/SEIB (on Page 595)** and to **OCDS (on Page 272)**.

The break switch:

Receives break source signals from the OCEM, OCDS, and the external pad

Sends break target signals to the OCEM, OCDS, and the external pad (see **Figure 11-79**).

**Figure 11-79 Break Switch Interface.**





**CONFIDENTIAL**

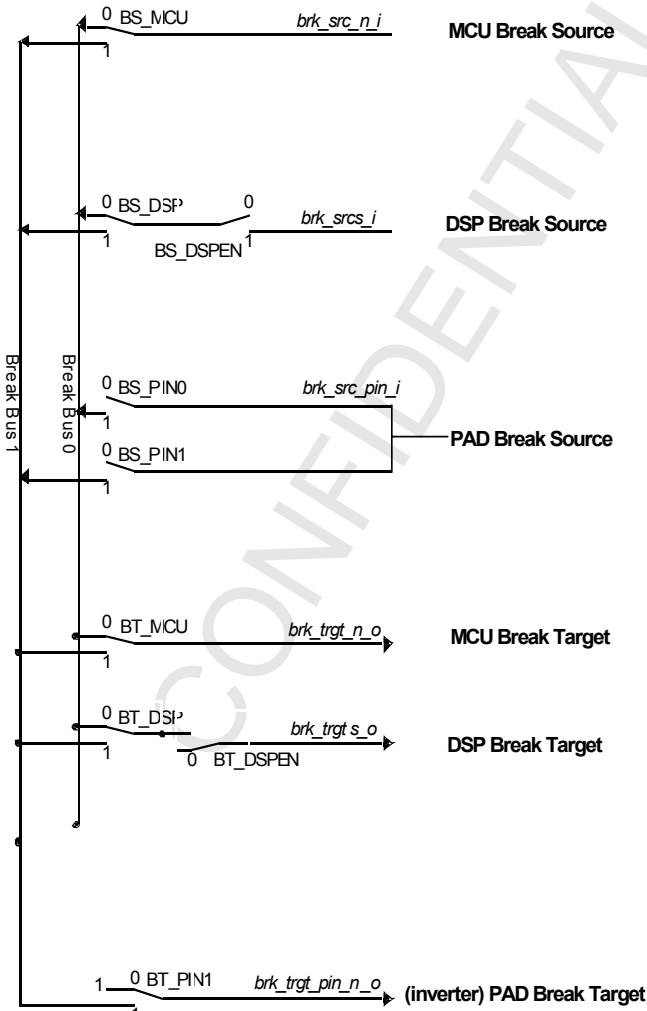
**TAP Controller and Break Switch**

**Features**

- Break sources: MCU, DSP, external (TRIG IN)
- Break targets: MCU, DSP, external (TRIG OUT)

Depending on the configuration of the register **MCD\_BBS**, any active break source signal generates an active break target signal. The break target signal(s) remains active as long as the break condition is true.

**Figure 11-80 Break Bus Switch**



**CONFIDENTIAL**

**TAP Controller and Break Switch**

### 11.9.2.1 Registers

There is one register (**MCD\_BBS**) to configure the break switch. It is a control and status register.

#### **MCD\_BBS**

**Break Bus Switch Status and Control**

**Reset value: F000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>BSS<sub>1</sub></b>	<b>BSS<sub>0</sub></b>	<b>BSS<sub>DSP</sub></b>	<b>BSS<sub>MCU</sub></b>	<b>RESERVED</b>	<b>TRIG<sub>IN</sub> LATCH</b>	<b>BT<sub>PIN1</sub></b>	<b>BS<sub>PIN1</sub></b>	<b>BS<sub>PIN0</sub></b>	<b>BT<sub>DSPEN</sub></b>	<b>BT<sub>DSP</sub></b>	<b>BT<sub>MCU</sub></b>	<b>BS<sub>DSPEN</sub></b>	<b>BS<sub>DSP</sub></b>	<b>BS<sub>MCU</sub></b>	

Field	Bits	Type	Description
<b>BS_MCU</b>	0	rw	<b>MCU Break Source Switch (Figure 11-80)</b> 0 Break source connected to break bus 0 1 Break source connected to break bus 1
<b>BS_DSP</b>	1	rw	<b>DSP Break Source Switch (Figure 11-80)</b> 0 Break source connected to break bus 0 1 Break source connected to break bus 1
<b>BS_DSPEN</b>	2	rw	<b>DSP Break Source Switch Enable</b> 0 Break source DSP disabled 1 Break source DSP enabled
<b>BT_MCU</b>	3	rw	<b>MCU Break Target Switch (Figure 11-80)</b> 0 Break target connected to break bus 0 1 Break target connected to break bus 1
<b>BT_DSP</b>	4	rw	<b>DSP Break Target Switch (Figure 11-80)</b> 0 Break target connected to break bus 0 1 Break target connected to break bus 1
<b>BT_DSPEN</b>	5	rw	<b>DSP Break Source Target Enable</b> 0 Break target DSP disabled 1 Break target DSP enabled
<b>BS_PIN0</b>	6	rw	<b>Pad Break Source TRIG_IN Enable for Bus 0</b> 0 Break source disabled. 1 Break source enabled.
<b>BS_PIN1</b>	7	rw	<b>Pad Break Source TRIG_IN Enable for Bus 1</b> 0 Break source disabled. 1 Break source enabled.

**CONFIDENTIAL**

**TAP Controller and Break Switch**

Field	Bits	Type	Description
<b>BT_PIN1</b>	8	rw	<b>Pad Break Target TRIG_OUT Enable (for Break Bus 1)</b> 0 Break target disabled. 1 Break target enabled.
<b>TRIG_IN_LATCH<sup>1)</sup></b>	9	rh	<b>NAND FLASH Detection at Reset for MCU Boot</b> 0 Other (depending on MON1/MON2 latch value) 1 NAND FLASH
<b>BSS_MCU</b>	12	rh	<b>Status of Break Source MCU</b>
<b>BSS_DSP</b>	13	rh	<b>Status of Break Source DSP</b>
<b>BBS0</b>	14	rh	<b>Status of Break Bus 0</b>
<b>BBS1</b>	15	rh	<b>Status of Break Bus 1</b>
<b>RESERVED</b>	11:10	r	Reserved, these bits must be left at their reset values.

<sup>1)</sup> TRIG\_IN\_LATCH does not correspond to any debug concept but it is a MCU boot configuration latched at reset. When reset is released, it has no more effect and fills the TRIG\_IN pad function.

### 11.9.2.1.1 Application Examples

Here are three configuration examples:

1. All break sources are connected to bus 1  
All break targets are connected to bus 0.  
The break source TRIG\_IN and the break target TRIG\_OUT are disabled.  
The value to be written in **MCD\_BBS** register is 0027<sub>H</sub>.
2. The MCU break source is connected to bus 1.  
The external break target TRIG\_OUT is connected to bus 1.  
The external break source TRIG\_IN is connected to bus 0.  
The other break sources and break targets are disabled.  
The value to be written in **MCD\_BBS** register is 0141<sub>H</sub>.
3. The MCU break source is connected to bus 1.  
The external break target TRIG\_OUT is connected to bus 1.  
DSP break source and external break source TRIG\_IN are connected to bus 0.  
MCU break target is connected to bus 0.  
The value to be written in **MCD\_BBS** register is 0165<sub>H</sub>  
(binary value: xxxx 0001 **0110 0101**. All bit positions that enable break sources and break targets are shown in bold).

## **11.9.3 JTAG**

### **11.9.3.1 General**

This specification defines the JTAG module and the protocol for a special JTAG mode to be used for On Chip Debug Support (OCDS) purposes, for Boundary Scan, and for Tests.

This specification refers to IEEE JTAG Standard (IEEE Std. 1149, October 21, 1993).

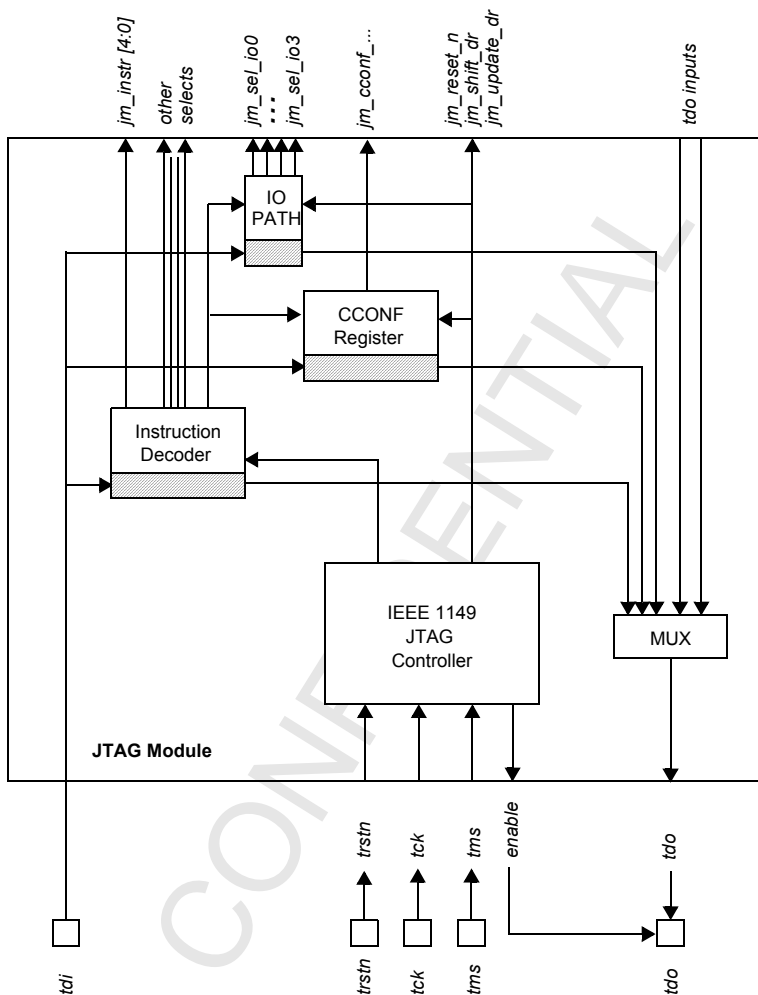
#### **Features**

- Based on the IEEE 1149 JTAG standard.
- 8-bit wide JTAG instruction register.
- Supports access to multiple CPUs (JTAG IO clients) on the same chip.
- Optional error protection for all IO mode data transfers.
- Generic memory access functionality.

#### **11.9.3.1.1 Block Diagram**

**Figure 11-81** shows a block diagram with all IO mode related signals of the JTAG module.

Figure 11-81 JTAG Module Block Diagram



CONFIDENTIAL

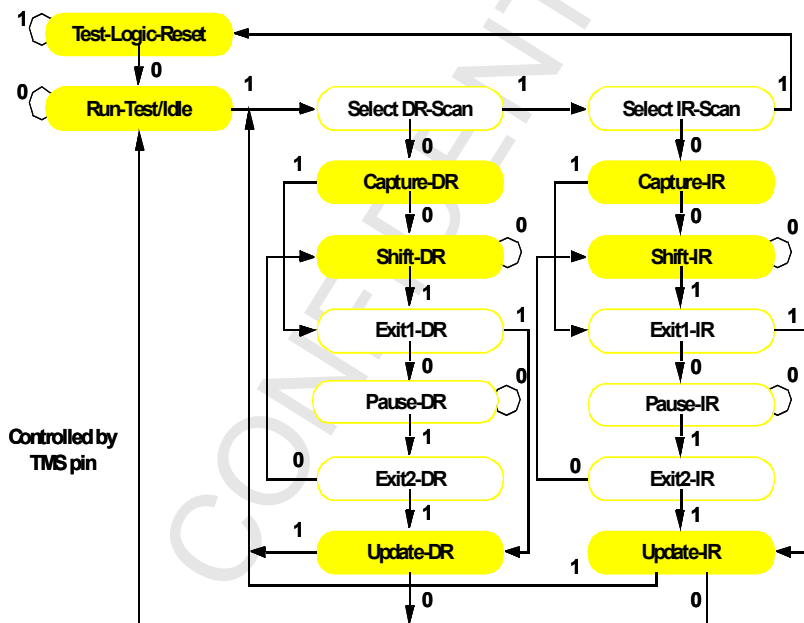
TAP Controller and Break Switch

### 11.9.3.1.2 TAP Controller

In E-GOLDRadio a Test Access Port (TAP) controller is integrated providing the standard functionality described in the IEEE 1149.1 standard.

To control the TAP controller five pins are necessary: TRST\_N, TMS, TCK, TDI, TDO. The TAP controller can be reset independently from the chip reset by an external TRST pin. The device works according to the typical tap controller state diagram (see [Figure 11-82](#)). State transitions are performed according to the inputs TMS and TCK. Test data on TDI will be loaded with a less or equal 4 MHz clock signal on TCK. 1 or 0 on TMS will cause a transition from one controller state to another; a constant 1 on TMS guarantees normal operation of the chip. If no boundary scan testing is desired, TMS and TDI do not need to be connected since internal pull-up transistors ensure high input levels in this case. A reset forces the TAP controller to the Test Logic Reset state.

Figure 11-82 TAP Controller State Diagram



**CONFIDENTIAL**

**TAP Controller and Break Switch**

### 11.9.3.1.3 JTAG Instructions

The instruction register (IR) contains the opcode which are transferred during a JTAG IR-scan. In the following table there is a list of the JTAG instructions.

**Table 11-73 JTAG Instructions**

Opcode	Range	Type	Instruction	Select signal
0000 0000	00 <sub>H</sub> - 07 <sub>H</sub> 8 instr.	IEEE1149	EXTEST	
0000 0001			INTEST	
0000 0010			SAMPLE/PRELOAD - not implemented	
0000 0011			RUNBIST- not implemented	
0000 0100			IDCODE	
0000 0101			USERCODE - not implemented	
0000 0110			CLAMP - not implemented	
0000 0111			HIGHZ - not implemented	
0000 1000 ...	08 <sub>H</sub> - 0F <sub>H</sub> 8 instr.	Reserved		
0001 0000	10 <sub>H</sub>	Chip config.	CCONF_SET	
0001 0001	11 <sub>H</sub>	Reserved		
0001 0010 ...	12 <sub>H</sub> - 1F <sub>H</sub> 14 instr.	Reserved		

**CONFIDENTIAL**

**TAP Controller and Break Switch**

**Table 11-73 JTAG Instructions (cont'd)**

Opcode	Range	Type	Instruction	Select signal
0010 0000 ...	20 <sub>H</sub> - 3F <sub>H</sub> 32 instr.	Test Modes	20 <sub>H</sub> RTC key 21 <sub>H</sub> Debug DSP 22 <sub>H</sub> Scan long 23 <sub>H</sub> Scan fast 24 <sub>H</sub> DSP Misc2 25 <sub>H</sub> Transio test 26 <sub>H</sub> Bist test 27 <sub>H</sub> Analog test 1 28 <sub>H</sub> Analog test 2 29 <sub>H</sub> PLL + Speed monitor 2A <sub>H</sub> Ram Access 2B <sub>H</sub> DC test 2C <sub>H</sub> DSP test 2D <sub>H</sub> Monitor signals test 2E <sub>H</sub> Short Reset 30 <sub>H</sub> RF test 1 31 <sub>H</sub> RF test 2 32 <sub>H</sub> RF test 3 33 <sub>H</sub> RF scan test	
0100 0000 ...	40 <sub>H</sub> - 5F <sub>H</sub> 32 instr.	Reserved		
0110 0000 ...	60 <sub>H</sub> - 7F <sub>H</sub> 32 instr.	Reserved		
1000 0000 ...	80 <sub>H</sub> - 9F <sub>H</sub> 32 instr.	Reserved		
1010 0000 ...	A0 <sub>H</sub> - BF <sub>H</sub> 32 instr.	Reserved		
1100 0000	C0 <sub>H</sub>	JTAG IO mode	JTAG_IO_SELECT_PATH	jm_sel_ioX <sup>1)</sup>
1100 0001	C1 <sub>H</sub> - CF <sub>H</sub>		JTAG_IO_INSTRUCTION1 <sup>2)</sup>	
...			... JTAG_IO_INSTRUCTION15	
1101 0000 ...	D0 <sub>H</sub> - FE <sub>H</sub> 47 instr.	Reserved		
1111 1111	FF <sub>H</sub>	IEEE1149	BYPASS	

<sup>1)</sup> Defined by the content of the [IOPATH Register](#)

<sup>2)</sup> Instructions C1<sub>H</sub> to CF<sub>H</sub> "isolate" the selected client, that is, tdi/tdo will be dedicated to that client (with its own [I/O Instructions](#), for example, CERBERUS or OCEM).



**CONFIDENTIAL**

**TAP Controller and Break Switch**

### 11.9.3.1.4 JTAG Control Pins

**Table 11-74** defines how the JTAG signals shall be connected to port pins.

**Table 11-74 JTAG Module Port Pins**

Pin	Signal	I/O	Internal Pull Up/Down		Remark
			IEEE1149	Recommended	
TDI	tdi	I	Up	Up	
TMS	tms	I	Up	Up	
TCK	tck	I	-	Down	Keep in defined state.
TDO	tdo	O	-	-	
TRST_N	trstn	I	Up	Down	The pull-down avoids, that an external pull-down is required during normal operation. This reduces cost and power consumption.

The JTAG standard defines an internal pull up for the TRST\_N reset pin. In normal operation, when the JTAG interface is not used, this pin has to be connected externally to ground. This means, that the pin permanently draws current. To avoid this, an internal pull down is recommended (**Table 11-74**). This has the benefit of reduced power consumption and the JTAG is kept in a defined state by default.

There are no compatibility problems with this non-standard solution. This means it will work with standard JTAG compliant tools, since those actively drive the TRST\_N pin. Thus the pull-up/down is irrelevant.

*Note: The JTAG standard requires dedicated pins.*

**CONFIDENTIAL**

**TAP Controller and Break Switch**

### 11.9.3.1.5 General JTAG Registers

The JTAG module contains the standard **JTAG Instructions** (8 bits) and **BYPASS Register** and the IO mode specific **IOPATH Register**.

**Table 11-75 JTAG Module Register Overview**

Register	Width	Reset ( <i>trstn</i> )	Description
BYPASS	1 bit	X	JTAG standard bypass register. If selected (BYPASS instruction) the <i>tdo</i> output is equal to <i>tdi</i> , delayed by one <i>tck</i> cycle.
CCONF	16 bit	0000 <sub>H</sub>	Chip Configuration Register.
ID	32 bit	-	Optional JTAG standard chip ID register. The ID is shifted out, when INSTRUCTION contains the IDCODE instruction. This register is not a part of the JTAG module
INSTRUCTION	8 bit	04 <sub>H</sub>	JTAG standard instruction register. In difference to all other registers it is set with an IR scan. The reset value is the IDCODE instruction.
IOPATH	2 bit	0 <sub>H</sub>	Selects the IO client.

*Note: All JTAG registers are shifted in and out with the LSB first.*

#### **BYPASS Register**

This is a mandatory JTAG register. If selected, the *tdo* output is the *tdi* input delayed by one *tck* cycle.

**CONFIDENTIAL**

## TAP Controller and Break Switch

## ID Register

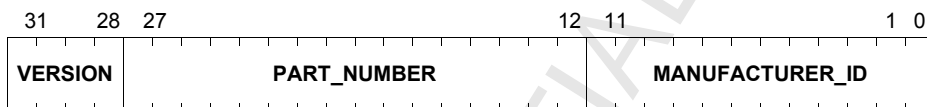
The **ID** register is not part of the JTAG module, its implementation is a product specific decision. This allows to maintain one central version and part number register, which can be accessed across JTAG with the IDCODE instruction. According to the JTAG standard, the IDCODE instruction has to have the following structure:

## ID

## JTAG ID Register

### Reset values:

**100F A083<sub>H</sub>**



Field	Bits	Type	Description
VERSION	31:28	r	Version of the chip.
PART_NUMBER	27:12	r	Part number of the chip.
MANUFACTURER_ID	11:0	r	Infineon's Manufacturer ID. The value is based on the JEDEC standard (JEP-106-G) manufacturer ID and the IEEE JTAG Standard.

## IOPATH Register

The **IOPATH** register is a modified JTAG scan register to provide error protection. For **IOPATH** *tdo* represents not the output of the **IOPATH** register but the input (**Figure 11-83**). This allows to detect transmission bit errors.

The *tdi/tdo* behavior is exactly like for a JTAG BYPASS instruction except, that the first bit output (state Capture-DR) is 1 and not 0. This difference is important in the case, that there was a bit error when the JTAG instruction was shifted in. In the most probable case, that this faulty JTAG instruction is not implemented, the JTAG module would set the BYPASS mode which could otherwise not be distinguished from the JTAG IO SELECT PATH instruction.

The **IOPATH** register is used to select the IO client (**Figure 11-85**). If the JTAG instruction is in the IO address range and not C0<sub>H</sub> (**Table 11-73**) the associated select signal is active. **IOPATH** register is 2 bits wide and set like a regular JTAG scan chain register with the JTAG IO SELECT PATH instruction.

CONFIDENTIAL

TAP Controller and Break Switch

Figure 11-83 IOPATH Register

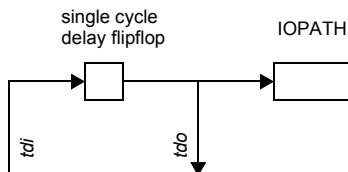


Table 11-76 IOPATH Register

Value (Binary)	Select Signal	Assignment Recommendation
00	jm_sel_io0	Reserved.
01	jm_sel_io1	Teaklite DSP (SEIB)
10	jm_sel_io2	C166S MCU (CERBERUS)
11	jm_sel_io3	Reserved

### CCONF Register

The **CCONF** register is provided to configure special chip states. It can be considered as an alternate mechanism to reset configurations. All configuration bits have associated protection bits. This allows a very straightforward access by different tools to their dedicated bits, sharing the JTAG interface.

The **CCONF** register is set with the CCONF\_SET JTAG instruction, refer to [Table 11-73](#), with the same behavior as the **IOPATH Register** (see [Figure 11-83](#)).

**CONFIDENTIAL**

**TAP Controller and Break Switch**

Two bits (**RST\_HLT**, **NET\_ENABLE**) have dedicated meanings, all others are reserved.

## **CCONF**

### **Chip Configuration Register**

(Reset value: 0000<sub>H</sub>)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RESERVED</b>												<b>NET EN. P</b>	<b>NET EN.</b>	<b>RST HLT P</b>	<b>RST HLT</b>

Field	Bits	Type	Description
<b>RST_HLT</b>	0	w	<b>Halt After Reset</b> 0 No effect 1 Halt mode after reset
<b>RST_HLT_P</b>	1	w	0 Bit protection: <b>CCONF.RST_HLT</b> unchanged 1 <b>RST_HLT</b> will be changed
<b>NET_ENABLE</b>	2	w	<b>Enable NET Interface</b> 0 Disabled 1 Enabled
<b>NET_ENABLE_P</b>	3	w	0 Bit protection: <b>CCONF.NET_ENABLE</b> unchanged 1 <b>NET_ENABLE</b> will be changed
<b>RESERVED</b>	15:4	r	Reserved, these bits must be left at their reset values.

**CONFIDENTIAL**

**TAP Controller and Break Switch**

### 11.9.3.1.6 RF Test Modes

#### DEBUG\_DSP

Debug control for DSP

(Reset value: 000<sub>H</sub>)

10	9	8	7	6	5	4	3	2	1	0
RESERVED				START	RESERVED	DSP_OCDS_EN	RESERVED	ocem_abort	RESERVED	dsp_rst_jtag

Field	Bits	Type	Description
<b>dsp_rst_jtag</b>	0	rw	<b>Reset DSP</b> This bit is OR'ed with the system reset and goes to the DSP.
<b>ocem_abort</b>	2	rw	<b>Send DSP to Monitor Program (start debug)</b> This bit is OR'ed with brk_trgts_o (from TCU) and goes to POCEM.oabort.
<b>dsp_ocds_en</b>	4	rw	Activates <b>DSP_DEBUG.OCEM</b> bit; active high
<b>START</b>	6	rw	<b>Restart DSP Synchronously from Break</b> 0 Do not restart 1 Restart
<b>RESERVED</b>	1, 3, 5, 10:7	r	Reserved, these bits must be left at their reset values.

### 11.9.3.2 Core Debug (JTAG I/O)

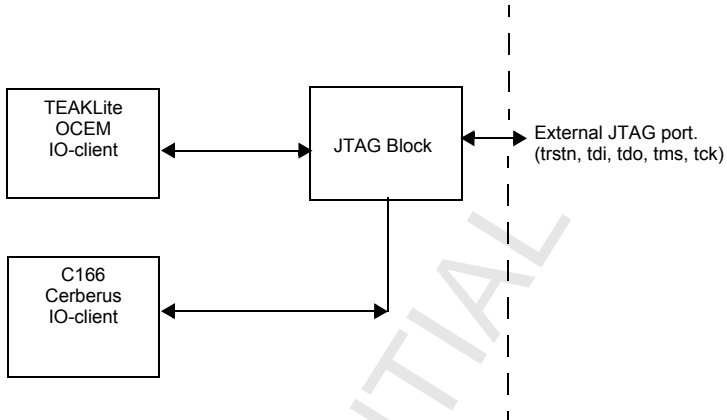
An important functionality of JTAG is the possibility of access to DSP and MCU module for debugging purpose.

The JTAG block is connected to the **OCDS (on Page 272)** C166 Cerberus and to the **OCEM/SEIB (on Page 595)** TEAKlite.

The selection between the TEAKlite OCEM (Named OCEM IO-client) and the C166 Cerberus (Named Cerberus IO-client) is done with the **IOPATH Register**.

The selected IO-client is addressed by the external JTAG port. There is only one IO-client selected.

**Figure 11-84 JTAG IO Mode Application Example**

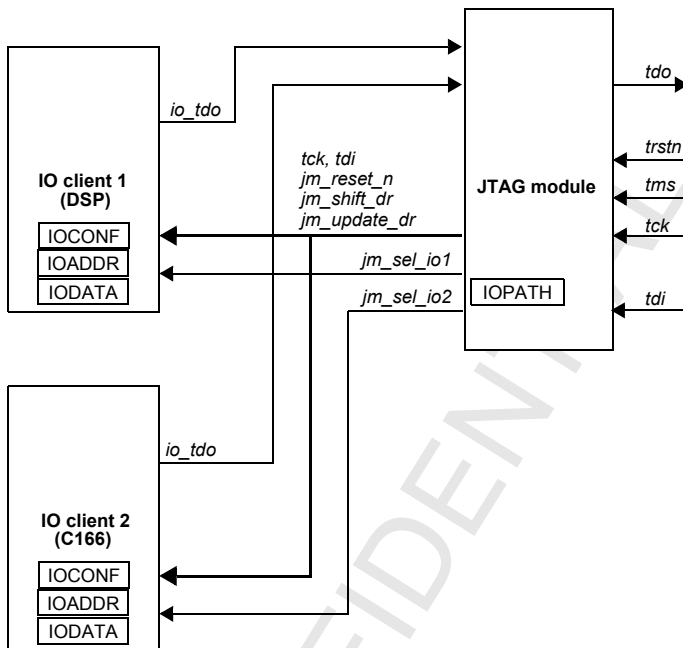


The IO mode is implemented in two parts ([Figure 11-85](#)):

- The extension of the JTAG module: IO mode JTAG instructions and IOPATH register to select the IO module. The extension is described in [Block Diagram](#).
- The IO client modules (one or multiple). These IO clients are the generic IO mode part of, for example, the OCDS module in [Figure 11-84](#). The selected client (IOPATH) is controlled with default chip internal JTAG signals by the JTAG module. Depending on the JTAG instruction, the configuration register IOCONF or the address register for the access IOADDR can be written, or the data register IODATA can be read or written.

Figure 11-85 shows only the JTAG IO mode related signals.

Figure 11-85 IO Mode Basic Architecture



Note: IO client 2 is now CERBERUS port C166 core.

Note: You will see further that IO client 1 (DSP) is implemented into TCU block (TCU test mode block) whereas IO client 2 is inside C166 (CERBERUS port)

### 11.9.3.2.1 Application Example for C166 Cerberus Debug

In order to connect the C166 Cerberus directly to the external JTAG port, do the following procedure:

1. Load the **JTAG IO mode** instruction: C0<sub>H</sub>
2. Load the corresponding **IOPATH Register** with the data 10<sub>B</sub>.
3. Load the **JTAG IO mode** instruction: C1<sub>H</sub> (to lead Cerberus tdo visible at external JTAG tdo port)
4. Deal with C166 Cerberus **I/O Instructions (on Page 294)** - IO\_CONFIG to IO\_CLIENT\_ID

Figure 11-86 shows IO-client selection and setting.





## CONFIDENTIAL

## TAP Controller and Break Switch

### 11.9.3.2.3 Multi Core Debug Session

There are two debug session choices:

- MCU
- MCU + DSP

#### Debug Session

##### MCU

To enter MCU into debug:

- HW means => “*jm\_sel\_io2*” signal (= **IOPATH Register** value) enables MCU OCDS.

A wait loop is defined into the monitor polling any debugger command. A low level of **PSW.USR0** makes the MCU leave the monitor.

#### Start a MCU debug session

1. Plug JTAG debugger
2. Power on the chip
3. Give a debug system reset via JTAG trstn
4. Deactivate DSP debug system by resetting **DEBUG\_DSP.dsp\_ocds\_en** and setting **DEBUG\_DSP.START** (Data associated with JTAG instruction 21<sub>H</sub>)
5. Activate MCU debug system by loading JTAG instruction C0<sub>H</sub> (communication mode) with its associated DATA\_DR for C166 IO-client (2<sub>H</sub>)

MCU is now in monitor program waiting for any debugger command, polling for **PSW.USR0** bit activity.

#### Interrupt the MCU debug session

Assume the debugger is “idle” waiting for a command (breakpoint...), MCU is in its monitor program => **PSW.USR0** is set.

1. Load JTAG instruction C1<sub>H</sub> to enter Cerberus debug commands (**I/O Instructions (on Page 294)**)
2. Execute the debug command (debugger is running) => **PSW.USR0** is reset. As a consequence, MCU leaves the monitor program to process the command. When debugger stops into the breakpoint, MCU has returned to monitor program => **PSW.USR0** is set. Into the debugger, the user can access different registers and carry on with an other debugger commands.

##### MCU + DSP

To enable the DSP OCEM block, refer to **Figure 8-71 OCEM Enabling Block (on Page 613)**: “*jm\_sel\_io1*” signal (= **IOPATH Register** value) and **DEBUG\_DSP.dsp\_ocds\_en** bit (both high active) allows the DSP debug.

**CONFIDENTIAL**

**TAP Controller and Break Switch**

When system reset is released, the DSP boot ROM scans for **STATUS1.DBG** bit (OCEM register) and for **STATUS1.DBG** bit to know if DSP must enter monitor program or not.

**Start a MCU + DSP Debug Session**

1. Plug JTAG debugger
2. Power on the chip
3. Give a debug system reset via JTAG trstn
4. Activate DSP debug system by setting **DEBUG\_DSP.dsp\_ocds\_en** and resetting **DEBUG\_DSP.START** (Data associated with JTAG instruction 21<sub>H</sub>) and by loading JTAG instruction C0<sub>H</sub> (communication mode) with associated **IOPATH Register** for Teaklite IO-client (1<sub>H</sub>)
5. Activate MCU debug system by loading JTAG instruction C0<sub>H</sub> (communication mode) with its associated **IOPATH Register** for C166 IO-client (2<sub>H</sub>)
6. Give a system reset via RESET\_N
7. Load JTAG instruction 21<sub>H</sub>
8. Set **DEBUG\_DSP.DSP\_OCDS\_EN** to 1
9. Perform a DSP RST by loading JTAG instruction 21<sub>H</sub> to set **dsp\_rst\_jtag** to 1 => Monitor program is entered as **STATUS1.DBG** bit and **STATUS1.DBG** are both active (mailbox memory buffer will be used to dump all internal/OCEM registers)
10. Load instruction C1<sub>H</sub> => mailbox will poll for any debugger command

**In a DSP Debug Session**

There are two HW interrupts sources and one internal interrupt source for DSP core:

1. **DEBUG\_DSP.ocem\_abort** activated (= 1) as a consequence of JTAG instruction
2. **dsp\_ocem\_breakinp\_i** signal activated in MCD
3. An address/data match (refer to **Section 8.15 OCEM/SEIB (on Page 595)**)

In any cases, monitor program is ended as soon as **DSP\_DEBUG.START** is active (= 1) => OCEM/internal registers are reloaded from the mailbox.

**Start a MCU debug session**

1. Power on the chip
2. Plug JTAG debugger
3. Perform a HW JTAG trstn
4. Load JTAG instruction C0<sub>H</sub> (communication mode)
5. Load **IOPATH Register** with C166 IO-client (2<sub>H</sub>)

**11.9.3.2.4** *Configure the **MCD\_BBS** register (break switch) via PD-Bus to use multicore debug.*

CONFIDENTIAL

CONFIDENTIAL

TEAKLite Bus Register Addresses

## 12 Register Lists and Mapping

History	
Current version: Rev. 1.06, 2005-11-04	
Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 0.02	
<b>Page 1293</b>	Added <b>PHX2_CTRL</b> register
Changes for Rev. 1.02	
<b>Page 1277</b>	Updated SSC(PD) register names WS00007169
Changes for Rev. 1.03	
<b>Page 1268</b>	Updated <b>Table 12-1 DSP Register Addresses</b> WS00007507
Changes for Rev. 1.06	

### 12.1 TEAKLite Bus Register Addresses

#### 12.1.1 DSP Memory Maps

The DSP memory map is described in **Table 12-1 DSP Register Addresses (on Page 1266)**.

**CONFIDENTIAL**

**TEAKLite Bus Register Addresses**

### 12.1.2 Registers in the DSP Memory Space

In [Table 12-1](#) all register used by the DSP are shown. All registers are reset asynchronously.

**Table 12-1 DSP Register Addresses**

DSP Peripheral	Name	Address	Clock Domain	Type	Modify HW = Hardware FW = Firmware
TEAKLite Interrupt Unit	<a href="#">INT_FINTA0</a>	E600	DSP	RD only	HW
	<a href="#">INT_EINTA0</a>	E601	DSP	RD/WR	FW
	<a href="#">INT_RINTA0</a>	E602	DSP	WR only	FW
	<a href="#">INT_SINTA0</a>	E603	DSP	WR only	FW
	<a href="#">INT_FINTB0</a>	E604	DSP	RD only	HW
	<a href="#">INT_EINTB0</a>	E605	DSP	RD/WR	FW
	<a href="#">INT_RINTB0</a>	E606	DSP	WR only	FW
	<a href="#">INT_SINTB0</a>	E607	DSP	WR only	FW
	<a href="#">INT_FINT1</a>	E608	DSP	RD only	HW
	<a href="#">INT_EINT1</a>	E609	DSP	RD/WR	FW
	<a href="#">INT_RINT1</a>	E60A	DSP	WR only	FW
	<a href="#">INT_SINT1</a>	E60B	DSP	WR only	FW
	<a href="#">INT_FINT2</a>	E60C	DSP	RD only	FW
	<a href="#">INT_EINT2</a>	E60D	DSP	RD/WR	FW
	<a href="#">INT_RINT2</a>	E60E	DSP	WR only	FW
	<a href="#">INT_SINT2</a>	E60F	DSP	WR only	FW
	<a href="#">INT_TOMCU</a>	E610	DSP	WR only	FW

**CONFIDENTIAL**

**TEAKLite Bus Register Addresses**

**Table 12-1 DSP Register Addresses**

DSP Peripheral	Name	Address	Clock Domain	Type	Modify HW = Hardware FW = Firmware
<b>GSM Cipher Unit (A51 and A52)</b> <b>CIPH_CSTAT and CIPH_KEY0 to CIPH_KEY3</b> use the same addresses as in A53.	<b>CIPH_CSTAT</b>	E620	DSP	RD/WR	HW/FW
	<b>CIPH_KEY0</b>	E621	DSP	RD/WR	FW
	<b>CIPH_KEY1</b>	E622	DSP	RD/WR	FW
	<b>CIPH_KEY2</b>	E623	DSP	RD/WR	FW
	<b>CIPH_KEY3</b>	E624	DSP	RD/WR	FW
	<b>CIPH_TMOD26</b>	E625	DSP	RD/WR	FW
	<b>CIPH_TMOD51</b>	E626	DSP	RD/WR	FW
	<b>CIPH_SFNUM</b>	E627	DSP	RD/WR	FW
<b>GSM Cipher Unit (A53)</b>	<b>CIPH_CSTAT</b>	E620	DSP	RD/WR	HW / FW
	<b>CIPH_KEY0</b>	E621	DSP	RD/WR	FW
	<b>CIPH_KEY1</b>	E622	DSP	RD/WR	FW
	<b>CIPH_KEY2</b>	E623	DSP	RD/WR	FW
	<b>CIPH_KEY3</b>	E624	DSP	RD/WR	FW
	Reserved	E625	DSP	RD/WR	FW
	Reserved	E626	DSP	RD/WR	FW
	Reserved	E627	DSP	RD/WR	FW
	<b>CIPH_KEY4</b>	E628	DSP	RD/WR	FW
	<b>CIPH_KEY5</b>	E629	DSP	RD/WR	FW
	<b>CIPH_KEY6</b>	E62A	DSP	RD/WR	FW
	<b>CIPH_KEY7</b>	E62B	DSP	RD/WR	FW
	<b>CIPH_KDATA1</b>	E62C	DSP	RD/WR	FW
	<b>CIPH_KDATA2</b>	E62D	DSP	RD/WR	FW
	<b>CIPH_KDATA3</b>	E62E	DSP	RD/WR	FW
	<b>CIPH_KDATA4</b>	E62F	DSP	RD/WR	FW

**CONFIDENTIAL**

**TEAKLite Bus Register Addresses**

**Table 12-1 DSP Register Addresses**

<b>DSP Peripheral</b>	<b>Name</b>	<b>Address</b>	<b>Clock Domain</b>	<b>Type</b>	<b>Modify</b> HW = Hardware FW = Firmware
<b>Timer</b>	<b>TMR1_CTRL</b>	E630	DSP	RD/WR	HW/FW
	<b>TMR1_CNT</b>	E631	DSP	RD only	HW
	<b>TMR1_INT0</b>	E632	DSP	RD/WR	FW
	<b>TMR1_INT1</b>	E633	DSP	RD/WR	FW
	<b>TMR2_CTRL</b>	E634	DSP	RD/WR	HW/FW
	<b>TMR2_CNT</b>	E635	DSP	RD/WR	HW/FW
	<b>TMR2_MAX</b>	E636	DSP	RD/WR	FW
<b>Equalizer Accelerator</b>	<b>VH_CONF1_EQ</b>	E640	DSP	WR only	FW
	<b>VH_CONF2_EQ</b>	E641	DSP	WR only	FW
	<b>VH_STATUS_EQ</b>	E642	DSP	RD only	HW
	<b>VH_CONF_CNT_E</b>	E643	DSP	RD/WR	FW
	<b>VH_STAT_CNT_E</b>	E644	DSP	RD only	HW
	<b>VH_SC_SOUT</b>	E645	DSP	RD/WR	FW
	<b>VH_SQUAL</b>	E646	DSP	RD only	FW
	Reserved	E647	DSP	RD/WR	FW
	Reserved	E648	DSP	RD/WR	FW
	Reserved	E649	DSP	RD/WR	FW



**CONFIDENTIAL**

**TEAKLite Bus Register Addresses**

**Table 12-1 DSP Register Addresses**

DSP Peripheral	Name	Address	Clock Domain	Type	Modify HW = Hardware FW = Firmware
Channel Decoder Accelerator	VH_CONF1_DEC	E650	DSP	WR only	FW
	VH_CONF2_DEC	E651	DSP	RD/WR	FW
	VH_STATUS_DEC	E652	DSP	RD only	HW
	VH_CONF_CNT_D	E653	DSP	RD/WR	FW
	VH_STAT_CNT_D	E654	DSP	RD only	HW
	VH_REF_BR_BFLYx (x = 0 to 7)	E655	DSP	RD/WR	FW
		E656	DSP	RD/WR	FW
		E657	DSP	RD/WR	FW
		E658	DSP	RD/WR	FW
		E659	DSP	RD/WR	FW
		E65A	DSP	RD/WR	FW
		E65B	DSP	RD/WR	FW
		E65C	DSP	RD/WR	FW
Voice (Audio Front End)	AFE_INTPTTR	E670	PLL	WR only	FW
	AFE_RWADDR	E671	PLL	RD only	HW <sup>1)</sup>
	AFE_BCON	E672	PLL	RD/WR	FW
	AFE_VRXCTRL1	E673	PLL	RD/WR	FW
	AFE_VRXCTRL2	E674	PLL	RD/WR	FW
	AFE_VTXCTRL	E675	PLL	RD/WR	FW
	AFE_RINGCTRL	E676	PLL	RD/WR	FW

**CONFIDENTIAL**

**TEAKLite Bus Register Addresses**

**Table 12-1 DSP Register Addresses**

<b>DSP Peripheral</b>	<b>Name</b>	<b>Address</b>	<b>Clock Domain</b>	<b>Type</b>	<b>Modify</b> HW = Hardware FW = Firmware
<b>Baseband Receive Unit</b>	<b>BB_CTRL</b>	E680	PLL	RD/WR	HW/FW
	<b>BB_INT_POINTER</b>	E681	PLL	RD/WR	FW
	<b>BB_WR_POINTER</b>	E682	PLL	RD only	HW <sup>1)</sup>
	<b>BB_STATUS</b>	E683	PLL	RD only	HW
	<b>BB_DCOFFSET_I</b>	E684	DSP	RD/WR	FW
	<b>BB_DCOFFSET_Q</b>	E685	DSP	RD/WR	FW
	<b>BB_FSHIFT</b>	E686	DSP	RD/WR	FW
	<b>BB_BRFILTER_CTRL</b>	E687	DSP	RD/WR	FW
	<b>BB_Pbase_MSB</b>	E688	PLL	RD only	HW
	<b>BB_Pbase_LSB</b>	E689	PLL	RD only	HW
	<b>BB_Padj_MSB</b>	E68A	PLL	RD only	HW
	<b>BB_Padj_LSB</b>	E68B	PLL	RD only	HW
	<b>BB_IQ_IMBALANCE</b>	E68C	DSP	RD/WR	FW
<b>MCS BLOCK for DSP</b>	<b>DSP_CFSTA</b>	E690	DSP	RD only	FW
	<b>DSP_CFSET</b>	E691	DSP	WR only	FW
	<b>DSP_CFR</b>	E692	DSP	WR only	FW
	<b>MCU_SEM (read)</b>	E693	DSP	RD only	FW
	<b>MCU_SEMS (set)</b>	E694	DSP	WR only	FW
	<b>MCU_SEMR (reset)</b>	E695	DSP	WR only	FW
<b>XBIU</b>	<b>BIU_CTRL</b>	E696	DSP	WR only	FW
	<b>SD_ADDRL</b>	E697	DSP	WR only	FW
	<b>SD_ADDRH</b>	E698	DSP	WR only	FW
	<b>BIU_DATA_W</b>	E699	DSP	WR only	FW
	<b>BIU_DATA_R</b>	E699	DSP	RD only	HW

**CONFIDENTIAL**

**TEAKLite Bus Register Addresses**

**Table 12-1 DSP Register Addresses**

<b>DSP Peripheral</b>	<b>Name</b>	<b>Address</b>	<b>Clock Domain</b>	<b>Type</b>	<b>Modify</b> HW = Hardware FW = Firmware
<b>DSP: General Part</b>	<b>DSP_ID</b>	E6A0	DSP	RD only	Hard wired
	<b>DSP_CTRL</b>	E6A1	DSP	WR only	HW/FW
	<b>DSP_DEBUG</b>	E6A2	DSP	RD/WR	FW
	<b>DSP_PAGE</b>	E6A3	DSP	RD/WR	FW
	Not used	E6A4	DSP	WR only	FW
	Reserved	E6A6	DSP	WR only	FW
	Reserved	E6A7	DSP	WR only	FW
	<b>DSP_DSPOUT</b>	E6A8	DSP	RD/WR	HW/FW
<b>GMSK Modulator Unit on DSP</b>	<b>MOD_CTRL</b>	E6B0	PLL	RD/WR	FW
	<b>MOD_STAT</b>	E6B1	PLL	RD only	HW
	<b>MOD_INT_ADDR</b>	E6B3	PLL	RD/WR	FW
	<b>MOD_OCI</b>	E6B4	PLL	RD/WR	FW
	<b>MOD_OCQ</b>	E6B5	PLL	RD/WR	FW
	<b>MOD_ACI</b>	E6B6	PLL	RD/WR	FW
	<b>MOD_ACQ</b>	E6B7	PLL	RD/WR	FW
	<b>MOD_FC</b>	E6B8	PLL	RD/WR	FW
<b>Synchronous Serial Controller on DSP</b>	<b>SSCDSP_CON</b>	E6C0	DSP	RD/WR	HW/FW
	Not Used	E6C1	DSP	RD/WR	FW
	<b>SSCDSP_TXB</b>	E6C2	DSP	RD/WR	FW
	<b>SSCDSP_RXB</b>	E6C3	DSP	RD/WR	HW
	<b>SSCDSP_RXFCON</b>	E6C4	DSP	RD/WR	FW
	<b>SSCDSP_TXFCON</b>	E6C5	DSP	RD/WR	FW
	<b>SSCDSP_FSTAT</b>	E6C6	DSP	RD only	HW
	<b>SSCDSP_BR</b>	E6C7	DSP	RD/WR	FW
	<b>SSCDSP_FDV</b>	E6C8	DSP	RD/WR	FW

**CONFIDENTIAL**

**TEAKLite Bus Register Addresses**

**Table 12-1 DSP Register Addresses**

<b>DSP Peripheral</b>	<b>Name</b>	<b>Address</b>	<b>Clock Domain</b>	<b>Type</b>	<b>Modify</b> HW = Hardware FW = Firmware
<b>I2S 1 (DAI)</b> <b>(Bi-directional Serial Audio Interface)</b>	<b>I2S1_CTRL</b>	E6D0	DSP	RD/WR	HW/FW
	<b>I2S1_CSEL</b>	E6D1	DSP	RD/WR	FW
	<b>I2S1_RWADDR</b>	E6D2	DSP	RD only	HW
	<b>I2S1_NUM0</b>	E6D3	DSP	RD/WR	FW
	<b>I2S1_DEN0</b>	E6D4	DSP	RD/WR	FW
	<b>I2S1_NUM1</b>	E6D5	DSP	RD/WR	FW
	<b>I2S1_DEN1</b>	E6D6	DSP	RD/WR	FW
	<b>I2S1_RXCONF</b>	E6D7	DSP	RD/WR	FW
	<b>I2S1_RXINTADDR</b>	E6D8	DSP	RD/WR	FW
	<b>I2S1_TXCONF</b>	E6D9	DSP	RD/WR	FW
	<b>I2S1_TXINTADDR</b>	E6DA	DSP	RD/WR	FW
<b>I2S 2 (Audio)</b> <b>(Bi-directional Serial Audio Interface)</b>	<b>I2S2_CTRL</b>	E6E0	DSP	RD/WR	HW/FW
	<b>I2S2_CSEL</b>	E6E1	DSP	RD/WR	FW
	<b>I2S2_RWADDR</b>	E6E2	DSP	RD only	HW
	<b>I2S2_NUM0</b>	E6E3	DSP	RD/WR	FW
	<b>I2S2_DEN0</b>	E6E4	DSP	RD/WR	FW
	<b>I2S2_NUM1</b>	E6E5	DSP	RD/WR	FW
	<b>I2S2_DEN1</b>	E6E6	DSP	RD/WR	FW
	<b>I2S2_RXCONF</b>	E6E7	DSP	RD/WR	FW
	<b>I2S2_RXINTADDR</b>	E6E8	DSP	RD/WR	FW
	<b>I2S2_TXCONF</b>	E6E9	DSP	RD/WR	FW
	<b>I2S2_TXINTADDR</b>	E6EA	DSP	RD/WR	FW
<b>I2S 3 (Unidirectional)</b> <b>(Unidirectional Serial Audio Interface)</b>	<b>I2S3_CTRL</b>	E6F0	DSP	RD/WR	HW/FW
	<b>I2S3_CSEL</b>	E6F1	DSP	RD/WR	FW
	<b>I2S3_RADDR</b>	E6F2	DSP	RD only	HW
	<b>I2S3_NUM</b>	E6F3	DSP	RD/WR	FW
	<b>I2S3_DEN</b>	E6F4	DSP	RD/WR	FW
	<b>I2S3_TXCONF</b>	E6F9	DSP	RD/WR	FW
	<b>I2S3_TXINTADDR</b>	E6FA	DSP	RD/WR	FW

**CONFIDENTIAL**

**PD-Bus Register Addresses**

- <sup>1)</sup> Normally the DSP has read/write access to the registers with 2 wait states. Only registers marked with (\*) need 4 wait states for the read access by the DSP.

## 12.2 PD-Bus Register Addresses

### 12.2.1 Register Addresses

The following list of SFRs and extended SFRs (ESFR) shows all Core and PD-Bus SFRs and their addresses and differentiates between bitaddressable SFR/ESFRs and non-bitaddressable SFR/ESFRs.

#### 12.2.1.1 SFR Description

**Table 12-2 Bit and Non-Bit Addressable SFR Areas**

Bitaddressable SFR Area				Non Bitaddressable SFR Area			
Long Address (Hex)	Short Address (Hex)	Register Name	Where	Long Address (Hex)	Short Address (Hex)	Register Name	Where
none	FF	R15	core				
none	FE	R14	core				
none	FD	R13	core				
none	FC	R12	core				
none	FB	R11	core				
none	FA	R10	core				
none	F9	R9	core				
none	F8	R8	core				
none	F7	R7	core				
none	F6	R6	core				
none	F5	R5	core				
none	F4	R4	core				
none	F3	R3	core				
none	F2	R2	core				
none	F1	R1	core				
none	F0	R0	core				

**CONFIDENTIAL**

**PD-Bus Register Addresses**

**Table 12-2 Bit and Non-Bit Addressable SFR Areas**

Bitaddressable SFR Area				Non Bitaddressable SFR Area			
Long Address (Hex)	Short Address (Hex)	Register Name	Where	Long Address (Hex)	Short Address (Hex)	Register Name	Where
FFFE	none	Reserved		FEFE	7F	<a href="#">PECC15</a>	Core
FFFC	none	Reserved		FEFC	7E	<a href="#">PECC14</a>	Core
FFFA	none	Reserved		FEFA	7D	<a href="#">PECC13</a>	Core
FFF8	none	Reserved		FEF8	7C	<a href="#">PECC12</a>	Core
FFF6	none	Reserved		FEF6	7B	<a href="#">PECXC6</a>	Core
FFF4	none	Reserved		FEF4	7A	<a href="#">PECXC4</a>	Core
FFF2	none	Reserved		FEF2	79	<a href="#">PECXC2</a>	Core
FFF0	none	Reserved		FEF0	78	<a href="#">PECXC0</a>	Core
FFEE	none	Reserved		FEED	77	<a href="#">PECC11</a>	Core
FFEC	none	Reserved		FEED	76	<a href="#">PECC10</a>	Core
FFEA	none	Reserved		FEED	75	<a href="#">PECC9</a>	Core
FFE8	none	Reserved		FEE8	74	<a href="#">PECC8</a>	Core
FFE6	none	Reserved		FEE6	73	<a href="#">PECSN11</a>	Core
FFE4	none	Reserved		FEE4	72	<a href="#">PECSN10</a>	Core
FFE2	none	Reserved		FEE2	71	<a href="#">PECSN9</a>	Core
FFE0	none	Reserved		FEE0	70	<a href="#">PECSN8</a>	Core
FFDE	EF	<a href="#">RTB_HI</a>	IIC	FEDE	6F	<a href="#">PECSN7</a>	Core
FFDC	EE	<a href="#">RTB_LO</a>	IIC	FEDC	6E	<a href="#">PECSN6</a>	Core
FFDA	ED	<a href="#">IIC_CON</a>	IIC	FEDA	6D	<a href="#">PECSN5</a>	Core
FFD8	EC	<a href="#">IIC_CFG</a>	IIC	FED8	6C	<a href="#">PECSN4</a>	Core
FFD6	EB	<a href="#">IIC_ADR</a>	IIC	FED6	6B	<a href="#">PECSN3</a>	Core
FFD4	EA	<a href="#">IIC_ST</a>	IIC	FED4	6A	<a href="#">PECSN2</a>	Core
FFD2	E9	Reserved	IIC	FED2	69	<a href="#">PECSN1</a>	Core
FFD0	E8	<a href="#">RTC_ISNC</a>	RTC	FED0	68	<a href="#">PECSN0</a>	Core
FFCE	E7	<a href="#">CC2OUT</a>	CC2	FECE	67	<a href="#">PECC7</a>	Core
FFCC	E6	<a href="#">CC2DRM</a>	CC2	FECC	66	<a href="#">PECC6</a>	Core
FFCA	E5	<a href="#">CCM5</a>	CC2	FECA	65	<a href="#">PECC5</a>	Core

**CONFIDENTIAL**

**PD-Bus Register Addresses**

**Table 12-2 Bit and Non-Bit Addressable SFR Areas**

Bitaddressable SFR Area				Non Bitaddressable SFR Area			
Long Address (Hex)	Short Address (Hex)	Register Name	Where	Long Address (Hex)	Short Address (Hex)	Register Name	Where
FFC8	E4	<b>CCM4</b>	CC2	FEC8	64	<b>PECC4</b>	Core
FFC6	E3	<b>T78CON</b>	CC2	FEC6	63	<b>PECC3</b>	Core
FFC4	E2	<b>CC1OUT</b>	CC1	FEC4	62	<b>PECC2</b>	Core
FFC2	E1	<b>CC1DRM</b>	CC1	FEC2	61	<b>PECC1</b>	Core
FFC0	E0	<b>CCM1</b>	CC1	FEC0	60	<b>PECC0</b>	Core
FFBE	DF	<b>CCM0</b>	CC1	FEBE	5F	<b>PECSN15</b>	Core
FFBC	DE	<b>T01CON</b>	CC1	FEBE	5E	<b>PECSN14</b>	Core
FFBA	DD	<b>PECXISNC</b>	Core	FEBA	5D	<b>PECSN13</b>	Core
FFB8	DC	<b>T6CON</b>	GPT12	FEB8	5C	<b>PECSN12</b>	Core
FFB6	DB	<b>T5CON</b>	GPT12	FEB6	5B	<b>PECXC14</b>	Core
FFB4	DA	<b>T4CON</b>	GPT12	FEB4	5A	<b>PECXC12</b>	Core
FFB2	D9	<b>T3CON</b>	GPT12	FEB2	59	<b>PECXC10</b>	Core
FFB0	D8	<b>T2CON</b>	GPT12	FEB0	58	<b>PECXC8</b>	Core
FFAE	D7	<b>WDTCON</b>	Core	FEAE	57	<b>WDT</b>	Core
FFAC	D6	<b>TFR</b>	Core	FEAC	56		
FFAA	D5	<b>FOCON</b>	Core	FEAA	55		
FFA8	D4	<b>PECISNC</b>	Core	FEA8	54		
FFA6	D3	<b>PCL_15(PCL_&lt;pad&gt;)</b>	PCL	FEA6	53		
FFA4	D2	<b>PMC1</b>	PMCU	FEA4	52		
FFA2	D1	<b>PMC0</b>	behind PCL BPI	FEA2	51		
FFA0	D0	<b>MCU_DSP_INT_ACK</b>	intr_ext	FEA0	50		
FF9E	CF	<b>T1IC</b>	Int	FE9E	4F		
FF9C	CE	<b>T0IC</b>	Int	FE9C	4E		
FF9A	CD	<b>CRIC</b>	Int	FE9A	4D		
FF98	CC	<b>T8IC</b>	Int	FE98	4C		
FF96	CB	<b>CC23IC</b>	Int	FE96	4B		

**CONFIDENTIAL**

**PD-Bus Register Addresses**

**Table 12-2 Bit and Non-Bit Addressable SFR Areas**

Bitaddressable SFR Area				Non Bitaddressable SFR Area			
Long Address (Hex)	Short Address (Hex)	Register Name	Where	Long Address (Hex)	Short Address (Hex)	Register Name	Where
FF94	CA	CC22IC	Int	FE94	4A		
FF92	C9	CC21IC	Int	FE92	49		
FF90	C8	CC20IC	Int	FE90	48		
FF8E	C7	CC19IC	Int	FE8E	47		
FF8C	C6	CC18IC	Int	FE8C	46		
FF8A	C5	CC17IC	Int	FE8A	45		
FF88	C4	CC16IC	Int	FE88	44		
FF86	C3	CC7IC	Int	FE86	43		
FF84	C2	CC6IC	Int	FE84	42		
FF82	33	Reserved	Core	FE82	41		
FF80	33	Reserved	Core	FE80	40		
FF7E	33	Reserved	Core	FE7E	3F		
FF7C	33	Reserved	Core	FE7C	3E		
FF7A	33	Reserved	Core	FE7A	3D		
FF78	33	Reserved	Core	FE78	3C		
FF76	33	Reserved	Core	FE76	3B		
FF74	33	Reserved	Core	FE74	3A		
FF72	33	Reserved	Core	FE72	39		
FF70	33	Reserved	Core	FE70	38		
FF6E	33	Reserved	Core	FE6E	37		
FF6C	33	Reserved	Core	FE6C	36		
FF6A	33	Reserved	Core	FE6A	35		
FF68	33	Reserved	Core	FE68	34		
FF66	33	Reserved	Core	FE66	33		
FF64	B2	T4IC	Int	FE64	32	IIC_PISEL	IIC
FF62	B1	T3IC	Int	FE62	31	reserved	IIC
FF60	B0	T2IC	Int	FE60	30	IIC_ID	IIC



**CONFIDENTIAL**

**PD-Bus Register Addresses**

**Table 12-2 Bit and Non-Bit Addressable SFR Areas**

Bitaddressable SFR Area				Non Bitaddressable SFR Area			
Long Address (Hex)	Short Address (Hex)	Register Name	Where	Long Address (Hex)	Short Address (Hex)	Register Name	Where
FF5E	AF	<b>S1CON</b>	ASC1	FE5E	2F		
FF5C	AE	<b>S1PISEL</b>	ASC1	FE5C	2E		
FF5A	AD	<b>S0FCSTAT</b>	ASC0	FE5A	2D		
FF58	AC	<b>S0FCCON</b>	ASC0	FE58	2C		
FF56	AB	<b>S0ABCON</b>	ASC0	FE56	2B		
FF54	AA	<b>S0CON</b>	ASC0	FE54	2A		
FF52	A9	<b>S0PISEL</b>	ASC0	FE52	29	<b>RTC_ALARM_HI</b>	RTC
FF50	A8	<b>SSCPD_FSTAT</b>	SSC(PD)	FE50	28	<b>RTC_ALARM_LO</b>	RTC
FF4E	A7	<b>SSCPD_TXFCON</b>	SSC(PD)	FE4E	27	Reserved	RTC
FF4C	A6	<b>SSCPD_RXFCON</b>	SSC(PD)	FE4C	26	<b>RTC_CTRL</b>	RTC
FF4A	A5	<b>SSCPD_CON</b>	SSC(PD)	FE4A	25	Reserved	RTC
FF48	A4	<b>MON_CR2</b>	PCL	FE48	24	Reserved	RTC
FF46	A3	<b>MON_CR1</b>	PCL	FE46	23	Reserved	RTC
FF44	A2	<b>PCL_57(PCL_&lt;pad&gt;)</b>	PCL	FE44	22	Reserved	RTC
FF42	A1	<b>PCL_56(PCL_&lt;pad&gt;)</b>	PCL	FE42	21	<b>RTC_REL_HI</b>	RTC
FF40	A0	<b>PCL_55(PCL_&lt;pad&gt;)</b>	PCL	FE40	20	<b>RTC_REL_LO</b>	RTC
FF3E	9F	<b>PCL_54(PCL_&lt;pad&gt;)</b>	PCL	FE3E	1F	<b>RTC_CNT_HI</b>	RTC
FF3C	9E	<b>PCL_53(PCL_&lt;pad&gt;)</b>	PCL	FE3C	1E	<b>RTC_CNT_LO</b>	RTC
FF3A	9D	<b>EBU_PDC</b>	PCL	FE3A	1D	<b>RTC_T14_CNT</b>	RTC
FF38	9C	<b>PCL_51(PCL_&lt;pad&gt;)</b>	PCL	FE38	1C	<b>RTC_T14_REL</b>	RTC
FF36	9B	<b>PCL_50(PCL_&lt;pad&gt;)</b>	PCL	FE36	1B	Reserved	RTC
FF34	9A	<b>PCL_49(PCL_&lt;pad&gt;)</b>	PCL	FE34	1A	<b>RTC_CON</b>	RTC
FF32	99	<b>PCL_48(PCL_&lt;pad&gt;)</b>	PCL	FE32	19	Reserved	RTC
FF30	98	<b>PCL_47(PCL_&lt;pad&gt;)</b>	PCL	FE30	18	RTCID (not used)	RTC
FF2E	97	<b>PCL_46(PCL_&lt;pad&gt;)</b>	PCL	FE2E	17		
FF2C	96	<b>PCL_45(PCL_&lt;pad&gt;)</b>	PCL	FE2C	16		
FF2A	95	<b>PCL_44(PCL_&lt;pad&gt;)</b>	PCL	FE2A	15		

**CONFIDENTIAL**

**PD-Bus Register Addresses**

**Table 12-2 Bit and Non-Bit Addressable SFR Areas**

Bitaddressable SFR Area				Non Bitaddressable SFR Area			
Long Address (Hex)	Short Address (Hex)	Register Name	Where	Long Address (Hex)	Short Address (Hex)	Register Name	Where
FF28	94	PCL_43(PCL_<pad>)	PCL	FE28	14		
FF26	93	PCL_42(PCL_<pad>)	PCL	FE26	13		
FF24	92	PCL_41(PCL_<pad>)	PCL	FE24	12		
FF22	91	PCL_40(PCL_<pad>)	PCL	FE22	11		
FF20	90	PCL_39(PCL_<pad>)	PCL	FE20	10		
FF1E	8F	<b>ONES</b>	Core	FE1E	0F	<b>ADDRSEL4</b>	EBU
FF1C	8E	<b>ZEROS</b>	Core	FE1C	0E	<b>ADDRSEL3</b>	EBU
FF1A	8D	<b>BUSCON4</b>	EBU	FE1A	0D	<b>ADDRSEL2</b>	EBU
FF18	8C	<b>BUSCON3</b>	EBU	FE18	0C	<b>ADDRSEL1</b>	EBU
FF16	8B	<b>BUSCON2</b>	EBU	FE16	0B	<b>STKUN</b>	Core
FF14	8A	<b>BUSCON1</b>	EBU	FE14	0A	<b>STKOV</b>	Core
FF12	89	<b>SYSCON</b>	Core	FE12	09	<b>SP</b>	Core
FF10	88	<b>PSW</b>	Core	FE10	08	<b>CP</b>	Core
FF0E	87	<b>MDC</b>	Core	FE0E	07	<b>MDL</b>	Core
FF0C	86	<b>BUSCON0</b>	Core	FE0C	06	<b>MDH</b>	Core
FF0A	85	Reserved		FE0A	05	Reserved	Core
FF08	84	Reserved		FE08	04	<b>CSP</b>	Core
FF06	83	P1H	Core	FE06	03	<b>DPP3</b>	Core
FF04	82	P1L	Core	FE04	02	<b>DPP2</b>	Core
FF02	81	P0H	Core	FE02	01	<b>DPP1</b>	Core
FF00	80	P0L	Core	FE00	00	<b>DPP0</b>	Core

CONFIDENTIAL

PD-Bus Register Addresses

### 12.2.1.2 ESFR Description

Table 12-3 Bit and Non-Bit Addressable ESFR Areas

Bitaddressable ESFR Area			Non-Bitaddressable ESFR Area		
Long Address (Hex)	Short Address (Hex)	Register Name	Long Address (Hex)	Short Address (Hex)	Register Name
none	FF	R15			
none	FE	R14			
none	FD	R13			
none	FC	R12			
none	FB	R11			
none	FA	R10			
none	F9	R9			
none	F8	R8			
none	F7	R7			
none	F6	R6			
none	F5	R5			
none	F4	R4			
none	F3	R3			
none	F2	R2			
none	F1	R1			
none	F0	R0			
F1FE	none		F0FE	7F	Reserved
F1FC	none		F0FC	7E	<b>DBGSR</b>
F1FA	none		F0FA	7D	<b>DIPX</b>
F1F8	none		F0F8	7C	<b>DIP</b>
F1F6	none		F0F6	7B	Reserved
F1F4	none		F0F4	7A	<b>DSWEVT</b>
F1F2	none		F0F2	79	<b>DEXEVT</b>
F1F0	none		F0F0	78	<b>DTREVT</b>
F1EE	none		F0EE	77	<b>DCMPDP</b>
F1EC	none		F0EC	76	<b>DCMPSP</b>

**CONFIDENTIAL**

**PD-Bus Register Addresses**

**Table 12-3 Bit and Non-Bit Addressable ESFR Areas**

Bitaddressable ESFR Area				Non-Bitaddressable ESFR Area			
Long Address (Hex)	Short Address (Hex)	Register Name		Long Address (Hex)	Short Address (Hex)	Register Name	
F1EA	none			F0EA	75		
F1E8	none			F0E8	74	<b>SSCPD_ID</b>	SSC
F1E6	none	Reserved	Core	F0E6	73	<b>SSCPD_TB</b>	SSC
F1E4	none			F0E4	72	<b>SSCPD_RB</b>	SSC
F1E2	none			F0E2	71	<b>SSCPD_BR</b>	SSC
F1E0	none	<b>RSTCON</b>	Core	F0E0	70	<b>SSCPD_PISEL</b>	SSC
F1DE	EF	Reserved	Core	F0DE	6F		
F1DC	EE	<b>SYSCON1</b>	Core	F0DC	6E		
F1DA	ED	<b>EXISEL</b>	Core	F0DA	6D		
F1D8	EC	Reserved		F0D8	6C	<b>DTIDR</b>	Core
F1D6	EB	Reserved		F0D6	6B	<b>CC2PISEL</b>	CC2
F1D4	EA	SYSCON3 (not used)	Core	F0D4	6A	<b>CC2IOC</b>	CC2
F1D2	E9	PCL_14 ( <b>PCL_&lt;pad&gt;</b> )	PCL	F0D2	69	<b>GPTPISEL</b>	GPT12
F1D0	E8	SYSCON2 (not used)	Core	F0D0	68	CAPREL	GPT12
F1CE	E7	PCL_38 ( <b>PCL_&lt;pad&gt;</b> )	PCL	F0CE	67	T6	GPT12
F1CC	E6	PCL_37 ( <b>PCL_&lt;pad&gt;</b> )	PCL	F0CC	66	T5	GPT12
F1CA	E5	PCL_36 ( <b>PCL_&lt;pad&gt;</b> )	PCL	F0CA	65	T4	GPT12
F1C8	E4	PCL_35 ( <b>PCL_&lt;pad&gt;</b> )	PCL	F0C8	64	T3	GPT12
F1C6	E3	PCL_34 ( <b>PCL_&lt;pad&gt;</b> )	PCL	F0C6	63	T2	GPT12
F1C4	E2	PCL_33 ( <b>PCL_&lt;pad&gt;</b> )	PCL	F0C4	62	<b>GPTID</b>	GPT12
F1C2	E1	PCL_32 ( <b>PCL_&lt;pad&gt;</b> )	PCL	F0C2	61	<b>SCUSLS</b>	Core
F1C0	E0	<b>EXICON</b>	Core	F0C0	60	<b>SCUSLC</b>	Core
F1BE	DF	PCL_31 ( <b>PCL_&lt;pad&gt;</b> )	PCL	F0BE	5F	<b>CC2SEM</b>	CC2
F1BC	DE	PCL_30 ( <b>PCL_&lt;pad&gt;</b> )	PCL	F0BC	5E	<b>CC2SEE</b>	CC2
F1BA	DD	PCL_29 ( <b>PCL_&lt;pad&gt;</b> )	PCL	F0BA	5D	T8	CC2
F1B8	DC	PCL_28 ( <b>PCL_&lt;pad&gt;</b> )	PCL	F0B8	5C	T7	CC2
F1B6	DB	PCL_27 ( <b>PCL_&lt;pad&gt;</b> )	PCL	F0B6	5B	T8REL	CC2

**CONFIDENTIAL**

**PD-Bus Register Addresses**

**Table 12-3 Bit and Non-Bit Addressable ESFR Areas**

Bitaddressable ESFR Area				Non-Bitaddressable ESFR Area			
Long Address (Hex)	Short Address (Hex)	Register Name		Long Address (Hex)	Short Address (Hex)	Register Name	
F1B4	DA	PCL_26 (PCL_<pad>)	PCL	F0B4	5A	T7REL	CC2
F1B2	D9	PCL_25 (PCL_<pad>)	PCL	F0B2	59	CC23	CC2
F1B0	D8	PCL_24 (PCL_<pad>)	PCL	F0B0	58	CC22	CC2
F1AE	D7	PCL_23 (PCL_<pad>)	PCL	F0AE	57	CC21	CC2
F1AC	D6	PCL_22 (PCL_<pad>)	PCL	F0AC	56	CC20	CC2
F1AA	D5	PCL_21 (PCL_<pad>)	PCL	F0AA	55	CC19	CC2
F1A8	D4	PCL_20 (PCL_<pad>)	PCL	F0A8	54	CC18	CC2
F1A6	D3	PCL_19 (PCL_<pad>)	PCL	F0A6	53	CC17	CC2
F1A4	D2	PCL_18 (PCL_<pad>)	PCL	F0A4	52	CC16	CC2
F1A2	D1	PCL_17 (PCL_<pad>)	PCL	F0A2	51	CC2ID	CC2
F1A0	D0	PCL_16 (PCL_<pad>)	PCL	F0A0	50	CC1PISEL	CC1
F19E	CF	TIM0IC	Int	F09E	4F	CC1IOC	CC1
F19C	CE	TIM4IC	Int	F09C	4E	CC1SEM	CC1
F19A	CD	MEAS1_TOGGLEIC	Int	F09A	4D	CC1SEE	CC1
F198	CC	IIC_PIC	Int	F098	4C	T1	CC1
F196	CB	T_INT2IC	Int	F096	4B	T0	CC1
F194	CA	TIM3IC	Int	F094	4A	T1REL	CC1
F192	C9	MEAS0_TOGGLEIC	Int	F092	49	T0REL	CC1
F190	C8	IIC_DIC	Int	F090	48	CC7	CC1
F18E	C7	T_INT1IC	Int	F08E	47	CC6	CC1
F18C	C6	TIM2IC	Int	F08C	46	CC5	CC1
F18A	C5	MEAS1IC	Int	F08A	45	CC4	CC1
F188	C4	IIC_EIC	int	F088	44	CC3	CC1
F186	C3	S0ASIC	Int	F086	43	CC2	CC1
F184	C2	TIM1IC	Int	F084	42	CC1	CC1
F182	C1	MEAS0IC	Int	F082	41	CC0	CC1
F180	C0	RFSSOTIC	Int	F080	40	CC1ID	CC1

**CONFIDENTIAL**

**PD-Bus Register Addresses**

**Table 12-3 Bit and Non-Bit Addressable ESFR Areas**

Bitaddressable ESFR Area				Non-Bitaddressable ESFR Area			
Long Address (Hex)	Short Address (Hex)	Register Name		Long Address (Hex)	Short Address (Hex)	Register Name	
F17E	BF	ECOIC	Int	F07E	3F	IDMANUF	ID
F17C	BE	KPDIC	Int	F07C	3E	IDCHIP	ID
F17A	BD	RES3IC	Int	F07A	3D	IDMEM	ID
F178	BC	RES2IC	Int	F078	3C	IDPROG	ID
F176	BB	RES1IC	Int	F076	3B	IDMEM2	ID
F174	BA	SSC0EIC	Int	F074	3A		
F172	B9	SSC0RIC	Int	F072	39		
F170	B8	SSC0TIC	Int	F070	38	IDRT	Core
F16E	B7	S1TBIC	Int	F06E	37	Reserved	
F16C	B6	S1EIC	Int	F06C	36	IOSR	Core
F16A	B5	S1RIC	Int	F06A	35	RWDATA	Core
F168	B4	S1TIC	Int	F068	34	COMDATA	Core
F166	B3	S0TM0IC	Int	F066	33		
F164	B2	S0CTSIC	Int	F064	32		
F162	B1	S0ABDETIC	Int	F062	31		
F160	B0	S0ABSTIC	Int	F060	30		
F15E	AF	EOPIC	Int	F05E	2F	S1RXFCON	ASC1
F15C	AE	From_DSP_to_MCU3IC	Int	F05C	2E	S1TXFCON	ASC1
F15A	AD	From_DSP_to_MCU2IC	Int	F05A	2D	S1FSTAT	ASC1
F158	AC	From_DSP_to_MCU1IC	Int	F058	2C	ID_SNUM4	PCL
F156	AB	From_DSP_to_MCU0IC	Int	F056	2B	ID_SNUM3	PCL
F154	AA	FEX7IC	Int	F054	2A	ID_SNUM2	PCL
F152	A9	FEX6IC	Int	F052	29	ID_SNUM1	PCL
F150	A8	FEX5IC	Int	F050	28	ID_SNUM0	PCL
F14E	A7	FEX4IC	Int	F04E	27		
F14C	A6	FEX3IC	Int	F04C	26		
F14A	A5	FEX2IC	Int	F04A	25	S0TMO	ASC0

**CONFIDENTIAL**

**PD-Bus Register Addresses**

**Table 12-3 Bit and Non-Bit Addressable ESFR Areas**

Bitaddressable ESFR Area				Non-Bitaddressable ESFR Area			
Long Address (Hex)	Short Address (Hex)	Register Name		Long Address (Hex)	Short Address (Hex)	Register Name	
F148	A4	FEX1IC	Int	F048	24	<b>S0FSTAT</b>	ASC0
F146	A3	FEX0IC	Int	F046	23	<b>S0TXFCON</b>	ASC0
F144	A2	SIMINIC	Int	F044	22	<b>S0RXFCON</b>	ASC0
F142	A1	SIMERRIC	Int	F042	21	<b>S0ABSTAT</b>	ASC0
F140	A0	SIMOKIC	Int	F040	20	<b>S0RBUF</b>	ASC0
F13E	9F	<b>FLASHIN</b>	Core	F03E	1F	<b>S0TBUF</b>	ASC0
F13C	9E	Not used	Core	F03C	1E	<b>S0PWM</b>	ASC0
F13A	9D	PCL_13 ( <b>PCL_&lt;pad&gt;</b> )	PCL	F03A	1D	<b>S0FDV</b>	ASC0
F138	9C	PCL_12 ( <b>PCL_&lt;pad&gt;</b> )	PCL	F038	1C	<b>S0BG</b>	ASC0
F136	9B	PCL_11 ( <b>PCL_&lt;pad&gt;</b> )	PCL	F036	1B	<b>S0PERID</b>	ASC0
F134	9A	PCL_10 ( <b>PCL_&lt;pad&gt;</b> )	PCL	F034	1A	<b>S1RBUF</b>	ASC1
F132	99	PCL_09 ( <b>PCL_&lt;pad&gt;</b> )	PCL	F032	19	<b>S1TBUF</b>	ASC1
F130	98	PCL_08 ( <b>PCL_&lt;pad&gt;</b> )	PCL	F030	18	<b>S1FDV</b>	ASC1
F12E	97	PCL_07 ( <b>PCL_&lt;pad&gt;</b> )	PCL	F02E	17	<b>S1BG</b>	ASC1
F12C	96	PCL_06 ( <b>PCL_&lt;pad&gt;</b> )	PCL	F02C	16	<b>S1PERID</b>	ASC1
F12A	95	PCL_05 ( <b>PCL_&lt;pad&gt;</b> )	PCL	F02A	15	<b>PMC_TIMER1</b>	PMCU behind PCL BPI
F128	94	PCL_04 ( <b>PCL_&lt;pad&gt;</b> )	PCL	F028	14	<b>PMC_TIMER0</b>	
F126	93	PCL_03 ( <b>PCL_&lt;pad&gt;</b> )	PCL	F026	13		Break switch behind intr_ext BPI
F124	92	PCL_02 ( <b>PCL_&lt;pad&gt;</b> )	PCL	F024	12	Reserved	Core
F122	91	PCL_01 ( <b>PCL_&lt;pad&gt;</b> )	PCL	F022	11		
F120	90	PCL_00 ( <b>PCL_&lt;pad&gt;</b> )	PCL	F020	10	<b>PCL_ID</b>	PCL
F11E	8F	<b>XBCON6</b>	Core	F01E	0F	<b>XADRS6</b>	Core
F11C	8E	<b>XBCON5</b>	Core	F01C	0E	<b>XADRS5</b>	Core

**CONFIDENTIAL**

**PD-Bus Register Addresses**

**Table 12-3 Bit and Non-Bit Addressable ESFR Areas**

Bitaddressable ESFR Area				Non-Bitaddressable ESFR Area			
Long Address (Hex)	Short Address (Hex)	Register Name		Long Address (Hex)	Short Address (Hex)	Register Name	
F11A	8D	<b>XBCON4</b>	Core	F01A	0D	<b>XADRS4</b>	Core
F118	8C	<b>XBCON3</b>	Core	F018	0C	<b>XADRS3</b>	Core
F116	8B	<b>XBCON2</b>	Core	F016	0B	<b>XADRS2</b>	Core
F114	8A	<b>XBCON1</b>	Core	F014	0A	<b>XADRS1</b>	Core
F112	89	GEA_int	Core	F012	09	reserved	
F110	88	Reserved		F010	08	reserved	
F10E	87	Reserved		F00E	07	<b>SCUID</b>	Core
F10C	86	Reserved		F00C	06	<b>CPUID</b>	Core
F10A	85	<b>RP0L</b>	Core	F00A	05	reserved	
F108	84	<b>RP0H</b>	Core	F008	04	reserved	
F106	83	DP1H	Core	F006	03	reserved	
F104	82	DP1L	Core	F004	02	reserved	
F102	81	DP0H	Core	F002	01	reserved	
F100	80	DP0L	Core	F000	00	reserved	



**CONFIDENTIAL**

**X-Bus Register Addresses**

### 12.3 X-Bus Register Addresses

**Table 12-4 Address Mapping of X-Bus Peripherals**

X-Bus Peripheral	Physical Address	Number of Bytes	Register Access Clock
	XADRS1: If XBIU is used by DSP, RGSAD must be 0EF <sub>H</sub> , RGSZ: 0000 (256 bytes) Explanation: refer to <a href="#">Section 10.3.5.6 TEAKLite Read/Write Request Response Time (on Page 650)</a> (If XBIU is not used by DSP, RGSAD is freely programmable, RGSZ: 0000 <sub>H</sub> )		
Chip Card Interface	00 EFFF <sub>H</sub> - 00 EFC0 <sub>H</sub>	(connected to one BPI)	X-Bus
SCCU Block	00 EFBF <sub>H</sub> - 00 EF80 <sub>H</sub>	(connected to one BPI)	X-Bus
Measurement	00 EF7F <sub>H</sub> - 00 EF40 <sub>H</sub>	(connected to one BPI)	X-Bus
Semaphore, Communication Register	00 EF3F <sub>H</sub> - 00 EF00 <sub>H</sub>	64 (connected to one BPI)	X-Bus
XBIU (includes BB filter & Cordic)			
Keypad Block			
AFC			
	XADRS2: RGSAD: freely programmable, for example, 0E0 <sub>H</sub> RGSZ: 0011 (2 Kbyte)		
GSM Timer + RF-Control	00 E7FF <sub>H</sub> - 00 E000 <sub>H</sub>	1920 bytes for RAM, 128 bytes for registers	13 MHz
	XADRS3: RGSAD: freely programmable, for example, 0D0 <sub>H</sub> RGSZ: 0100(4 Kbyte)		
Shared memory (1.5Kword(16bits) RAM)	00 DBFF <sub>H</sub> - 00 D000 <sub>H</sub>	1.5k for RAM	X-Bus
	XADRS4: RGSAD: freely programmable, for example, 0EC <sub>H</sub> RGSZ: 0001 (512 Byte)		

**CONFIDENTIAL**

**X-Bus Register Addresses**

**Table 12-4 Address Mapping of X-Bus Peripherals**

<b>X-Bus Peripheral</b>	<b>Physical Address</b>	<b>Number of Bytes</b>	<b>Register Access Clock</b>
<b>FREE</b>	00 EDFF <sub>H</sub> - 00 ED80 <sub>H</sub>	128 bytes free	Free
<b>GPRS Control Register</b>	00 ED7F <sub>H</sub> - 00 ED40 <sub>H</sub>	64 (connected to one BPI)	X-Bus
<b>Clock Generation Unit</b>	00 ED3F <sub>H</sub> - 00 ED00 <sub>H</sub>	64 (connected to one BPI)	13 MHz
<b>Port Signal Logic Arranger</b>	00 ECFF <sub>H</sub> - 00 EC80 <sub>H</sub>	128 bytes RAM	X-Bus
<b>FREE</b>	00 EC7F <sub>H</sub> - 00 EC40 <sub>H</sub>	64 bytes free.	Free
<b>Port Signal Logic Arranger</b>	00 EC3F <sub>H</sub> - 00 EC00 <sub>H</sub>	64 bytes for registers	X-Bus

**CONFIDENTIAL**

**X-Bus Register Addresses**

**Table 12-5 Register Overview for XADRS1**

<b>Name</b>	<b>Address Offset (Hex)</b>	<b>Functional Block</b>
<b>AFCID</b>	08	AFC
<b>AFCVAL</b>	0A	AFC
unused	0C	
unused	0E	
unused	10	
unused	12	
unused	14	
unused	16	
<b>KBID</b>	18	Keypad
<b>KBDINP</b>	1A	Keypad
<b>KBDOUT</b>	1C	Keypad
unused	1E	
unused	20	
unused	22	
unused	24	
unused	26	
<b>BIUID</b>	28	XBIU
<b>BIU_DATA_W</b>	2A	XBIU
<b>BIU_DATA_R</b>	2A	XBIU
<b>MCU_CFS</b>	2C	ComRAM
<b>MCU_CFR</b>	2E	ComRAM
<b>MCU_CFSTA</b>	30	ComRAM
<b>MCU_CFID</b>	32	ComRAM
<b>DSP_SEMS</b>	34	ComRAM
<b>DSP_SEMR</b>	36	ComRAM
<b>DSP_SEM</b>	38	ComRAM
<b>MCU_SEMID</b>	3A	ComRAM
unused	3C	
unused	3E	
<b>MEAS_ID</b>	40	Measurement

**CONFIDENTIAL**

**X-Bus Register Addresses**

**Table 12-5 Register Overview for XADRS1**

<b>Name</b>	<b>Address Offset (Hex)</b>	<b>Functional Block</b>
unused	42	Measurement
Reserved	44	Measurement
unused	46	Measurement
<a href="#">ANA_CTRL1</a>	48	Measurement
<a href="#">ANA_CTRL2</a>	4A	Measurement
<a href="#">MEAS_CTRL1</a>	4C	Measurement
<a href="#">MEAS_CTRL2</a>	4E	Measurement
<a href="#">MEAS_STAT</a>	50	Measurement
unused	52	Measurement
<a href="#">MEAS_DATA0</a>	54	Measurement
unused	56	Measurement
<a href="#">MEAS_DATA1</a>	58	Measurement
unused	5A	Measurement
<a href="#">MEAS_DATA2</a>	5C	Measurement
unused	5E	Measurement
<a href="#">MEAS_DATA3</a>	60	Measurement
unused	62	Measurement
<a href="#">MEAS_DATA4</a>	64	Measurement
unused	66	Measurement
<a href="#">MEAS_DATA5</a>	68	Measurement
unused	6A	Measurement
<a href="#">MEAS_DATA6</a>	6C	Measurement
unused	6E	Measurement
<a href="#">MEAS_DATA7</a>	70	Measurement
unused	72	Measurement
<a href="#">MEAS_CLK</a>	74	Measurement
unused	76	Measurement
unused	78	Measurement
unused	7A	Measurement
unused	7C	Measurement

**CONFIDENTIAL**

**X-Bus Register Addresses**

**Table 12-5 Register Overview for XADRS1**

Name	Address Offset (Hex)	Functional Block
unused	7E	Measurement
<b>SCCUID</b>	80	SCCU Block
	82	SCCU Block
<b>SCCUTDMINL</b>	84	SCCU Block
<b>Not Used</b>	86	SCCU Block
<b>SCCUSLPCTRL</b>	88	SCCU Block
<b>Not Used</b>	8A	SCCU Block
<b>SCCUSCTRL</b>	8C	SCCU Block
<b>Not Used</b>	8E	SCCU Block
<b>SCCUREFINL</b>	90	SCCU Block
<b>Not Used</b>	92	SCCU Block
<b>SCCUNQTZ</b>	94	SCCU Block
<b>Not Used</b>	96	SCCU Block
<b>SCCUWAITL</b>	98	SCCU Block
<b>SCCUWAITH</b>	9A	SCCU Block
<b>SCCUHWWAKEUPL</b>	9C	SCCU Block
<b>SCCUHWWAKEUPH</b>	9E	SCCU Block
<b>SCCUTDMOUTL</b>	A0	SCCU Block
<b>Not Used</b>	A2 Not used	SCCU Block
<b>SCCUREFL</b>	A4	SCCU Block
<b>SCCUREFH</b>	A6	SCCU Block
<b>SCCUCLKSTA</b>	A8	SCCU Block
unused	AA	SCCU Block
<b>SCCUSMSTA</b>	AC	SCCU Block
unused	AE	SCCU Block
<b>Reserved</b>	B0	SCCU Block
unused	B2	SCCU Block
<b>Reserved</b>	B4	SCCU Block
unused	B6	SCCU Block
<b>Reserved</b>	B8	SCCU Block

**CONFIDENTIAL**

**X-Bus Register Addresses**

**Table 12-5 Register Overview for XADRS1**

<b>Name</b>	<b>Address Offset (Hex)</b>	<b>Functional Block</b>
unused	BA	SCCU Block
<b>SCCUSPCR</b>	BC	SCCU Block
unused	BE	SCCU Block
<b>SIMID</b>	C0	Chip Card
unused	C2	Chip Card
<b>SIMCTRL</b>	C4	Chip Card
unused	C6	Chip Card
<b>SIMBRF</b>	C8	Chip Card
unused	CA	Chip Card
<b>SIMSTATUS</b>	CC	Chip Card
unused	CE	Chip Card
<b>SIMIRQEN</b>	D0	Chip Card
unused	D2	Chip Card
<b>SIMRXSPC</b>	D4	Chip Card
unused	D6	Chip Card
<b>SIMTXSPC</b>	D8	Chip Card
unused	DA	Chip Card
<b>SIMCHTIMER1</b>	DC	Chip Card
<b>SIMCHTIMER2</b>	DE	Chip Card
<b>Reserved</b>	E0	Chip Card
unused	E2	Chip Card
<b>Reserved</b>	E4	Chip Card
<b>SIMBWT2</b>	E6	Chip Card
<b>SIMTX</b>	E8	Chip Card
unused	EA	Chip Card
<b>SIMRX</b>	EC	Chip Card
unused	EE	Chip Card
<b>SIMINS</b>	F0	Chip Card
unused	F2	Chip Card
<b>SIMP3</b>	F4	Chip Card

**CONFIDENTIAL**

**X-Bus Register Addresses**

**Table 12-5 Register Overview for XADRS1**

Name	Address Offset (Hex)	Functional Block
unused	F6	Chip Card
<b>SIMSW1</b>	F8	Chip Card
unused	FA	Chip Card
<b>SIMSW2</b>	FC	Chip Card
unused	FE	Chip Card

**Table 12-6 Register Overview for XADRS2**

Name	Address Offset (Hex)	Functional Block
<b>TID</b>	008	GSM IF
RFCON1	010	GSM IF
unused	012	GSM IF
RFCON2	014	GSM IF
unused	016	GSM IF
RFSSCTB	018	GSM IF
unused	01A	GSM IF
<b>TCOR</b>	01C	GSM IF
unused	01E	GSM IF
<b>TOVF</b>	020	GSM IF
unused	022	GSM IF
<b>TINT1</b>	024	GSM IF
unused	026	GSM IF
<b>TINT2</b>	028	GSM IF
unused	02A	GSM IF
<b>TOFFSET</b>	02C	GSM IF
unused	02E	GSM IF
<b>TFSKIP</b>	030	GSM IF
unused	032	GSM IF
<b>TCLT</b>	034	GSM IF
unused	036	GSM IF
<b>TCEAP</b>	038	GSM IF

**CONFIDENTIAL**

**X-Bus Register Addresses**

**Table 12-6 Register Overview for XADRS2**

Name	Address Offset (Hex)	Functional Block
unused	03A	GSM IF
<b>TEAPT</b>	03C	GSM IF
unused	03E	GSM IF
<b>TEAPB</b>	040	GSM IF
unused	042	GSM IF
<b>TGERB</b>	044	GSM IF
<b>TGERT</b>	046	GSM IF
<b>TPARA</b>	048	GSM IF
unused	04A	GSM IF
<b>TFADE1</b>	04C	GSM IF
<b>TFADE2</b>	04E	GSM IF
<b>Not Used</b>	050	GSM IF
unused	052	GSM IF
<b>GSMCLK1B</b>	054	GSM IF
<b>GSMCLK1T</b>	056	GSM IF
<b>GSMCLK2B</b>	058	GSM IF
unused		GSM IF
<b>GSMCLK2T</b>	05A	GSM IF
<b>GSMCLK3</b>	05C	GSM IF
Free	05E-7F	GSM IF
RF Control RAM	080-3FF	GSM IF
GSM Timer RAM	400-7FF	GSM IF

**Table 12-7 Register Overview for XADRS4**

Name	Address Offset (Hex)	Functional Block
<b>LPSAID</b>	008	LPSA
<b>LPACON</b>	010	LPSA
<b>LPASEL</b>	012	LPSA
LPA RAM	080-0FF	LPSA
<b>CGUID</b>	100	CGU Block



**CONFIDENTIAL**

**X-Bus Register Addresses**

**Table 12-7 Register Overview for XADRS4**

<b>Name</b>	<b>Address Offset (Hex)</b>	<b>Functional Block</b>
<b>RTCIF</b>	102	CGU Block
<b>MST_CLK_CTRL</b>	104	CGU Block
<b>PLL_CTRL</b>	106	CGU Block
<b>PHX_CTRL</b>	108	CGU Block
<b>SIFCLKS</b>	10A	CGU Block
<b>RST_CTRL_STA</b>	10C	CGU Block
<b>PHX2_CTRL</b>	10E	CGU Block
Free	110-13E	
GEA1/GEA2		
Unused	140-146	GPRS
<b>GPRSID_12</b>	148	GPRS
<b>GPRSCTRL_12</b>	150	GPRS
<b>I_DATA_12</b>	152	GPRS
<b>O_DATA_12</b>	154	GPRS
<b>FCS_REG_0_12</b>	156	GPRS
<b>FCS_REG_1_12</b>	158	GPRS
<b>INPUT_REG_0_12</b>	15A	GPRS
<b>INPUT_REG_1_12</b>	15C	GPRS
<b>POLYNOM_0_12</b>	15E	GPRS
<b>POLYNOM_1_12</b>	160	GPRS
<b>KC_REG_0_12</b>	162	GPRS
<b>KC_REG_1_12</b>	164	GPRS
<b>KC_REG_2_12</b>	166	GPRS
<b>KC_REG_3_12</b>	168	GPRS
<b>Reserved</b>	16A	GPRS
<b>Reserved</b>	16C	GPRS
	16E	
	170	
	172	
	174	

**CONFIDENTIAL**

**X-Bus Register Addresses**

**Table 12-7 Register Overview for XADRS4**

<b>Name</b>	<b>Address Offset (Hex)</b>	<b>Functional Block</b>
	176	
	178	
	17A	
	17C	
	17E	
<b>GEA3</b>		
<b>GPRSID_3</b>	148	GPRS
<b>GPRSCTRL_3</b>	150	GPRS
<b>I_DATA_3</b>	152	GPRS
<b>O_DATA_3</b>	154	GPRS
<b>FCS_REG_0_3</b>	156	GPRS
<b>FCS_REG_1_3</b>	158	GPRS
<b>INPUT_REG_00_3</b>	15A	GPRS
<b>INPUT_REG_01_3</b>	15C	GPRS
<b>INPUT_REG_10_3</b>	15E	GPRS
<b>INPUT_REG_11_3</b>	160	GPRS
<b>KC_REG_0_L_3</b>	162	GPRS
<b>KC_REG_0_H_3</b>	164	GPRS
<b>KC_REG_1_L_3</b>	166	GPRS
<b>KC_REG_1_H_3</b>	168	GPRS
<b>KC_REG_2_L_3</b>	16A	GPRS
<b>KC_REG_2_H_3</b>	16C	GPRS
<b>KC_REG_3_L_3</b>	16E	GPRS
<b>KC_REG_3_H_3</b>	170	GPRS
<b>Reserved</b>	172	GPRS
<b>Reserved</b>	174	GPRS
<b>Reserved</b>	176	GPRS
<b>Reserved</b>	178	GPRS
<b>Reserved</b>	17A	GPRS

**CONFIDENTIAL**

**X-Bus Register Addresses**

**Table 12-7 Register Overview for XADRS4**

<b>Name</b>	<b>Address Offset (Hex)</b>	<b>Functional Block</b>
<b>Reserved</b>	17C	GPRS
Free	17E	GPRS
Free	180-1FF	

CONFIDENTIAL

CONFIDENTIAL

**CONFIDENTIAL**

## 13 Electrical and Temperature Characteristics

<b>History</b>	
Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.00	
<b>Page 1300</b>	Updated <b>Table 12-10 Maximum ESD</b>
Changes for Rev. 1.01	
<b>Page 1307</b>	Updated <b>Table 12-18 Input Capacitances and Resistors</b> and <b>Table 12-19 External Load Capacitances and Resistors</b> , WS00006913
<b>Page 1301</b> <b>Page 1303</b>	Updated <b>Table 12-12 Maximum Power Supply Voltages</b> and <b>Table 12-15 Normal Power Supply Voltages</b> WS00006914
<b>Page 1345</b>	Updated <b>Table 12-26 Timing Characteristic of I2C-Bus Interface</b> WS00005887
<b>Page 1379</b>	Changed "W" to "Ω" in <b>Table 12-54 Specification of Pin PAOUT1 (TXON = 1)</b> WS00007132
<b>Page 1312</b>	F26M to XO warning added to <b>Section 13.2.1.2 Dynamic (AC Characteristics)</b> WS00007126
<b>Page 1362</b>	Updated <b>Section 13.2.1.2.11 Output Clock CLKOUT</b> WS00007126
<b>Page 1691</b>	Updated <b>Section 13.2.2.13 Input Clock F26M (XO)</b> WS00007126
Changes for Rev. 1.02	
all	"USIM" changed to "SIM" except in signal and directory names WS00007145
<b>Page 1330</b> to <b>Page 1342</b>	Updated <b>Section 13.2.1.2.3.4 Timing Diagrams for PMCU Parameter Characterization</b> text and figures. The figures are now readable. WS00007377
<b>Page 1315</b> <b>Page 1322</b>	Updated <b>Table 12-23 EBU and PMCU Measured Parameters</b> and <b>Table 12-24 EBU and PMCU Derived Parameters</b> Note added to <b>Section 13.2.1.2.3 EBU</b> WS00007326
<b>Page 1325</b> to <b>Page 1329</b>	Changed "Normal ALE" signal to "ADV\ Pad" in <b>Figure 13-4</b> , <b>Figure 13-5</b> , <b>Figure 13-6</b> , <b>Figure 13-7</b> , <b>Figure 13-8</b> WS00007373
<b>Page 1385</b>	Updated the names of VDD[1..6] in <b>Table 12-57 AC/DC Characteristics</b> WS00007484
<b>Page 1330</b>	Added duty cycle Note to <b>Section 13.2.1.2.3.4 Timing Diagrams for PMCU Parameter Characterization</b> WS00007377

**CONFIDENTIAL**

<b>History</b>	
<b>Page 1331 to Page 1333</b>	Updated ts <sub>1_1_adv_r_pmc</sub> and ts <sub>1_1_adv_f_pmc</sub> signals in <b>Figure 13-9</b> , <b>Figure 13-10</b> , and <b>Figure 13-11</b> WS00007326
Changes for Rev. 1.03	
<b>Page 1379</b>	Updated <b>Table 12-54 Specification of Pin PAOUT1 (TXON = 1)</b> WS00007540
<b>Page 1322</b>	Added parameter ts <sub>20</sub> to <b>Table 12-24 EBU and PMCU Derived Parameters</b> WS0007327 Updated definitions WS00007801
<b>Page 1363</b>	Updated <b>Table 12-38 General Electrical Characteristics of Audio Receive Path</b> WS00007729
<b>Page 1325, Page 1329, Page 1331, Page 1332 Page 1315</b>	Updated <b>Figure 13-4</b> , <b>Figure 13-8</b> , <b>Figure 13-9</b> , and <b>Figure 13-10</b>  Updated definitions in <b>Table 12-23 EBU and PMCU Measured Parameters</b> WS00007798
<b>Page 1359</b>	Updated <b>Table 12-34 Timing Characteristic of SSC in Slave Mode</b> WS00007934
Changes for Rev. 1.04	
<b>Page 1368</b>	Updated <b>Table 12-44 Electrical Characteristics of Microphone Supply</b> WS00008378
<b>Page 1322</b>	Updated <b>Table 12-24 EBU and PMCU Derived Parameters</b> WS00008449
<b>Page 1309 Page 1311</b>	Updated <b>Table 12-20 Pad Output Current</b> and <b>Table 12-21 Pull-Up/Pull-Down Currents</b> WS00008454
<b>Page 1374</b>	Added <b>Table 12-49 Specification of pins M0 to M2 and M7 to M10</b> WS00008367
<b>Page 1303</b>	Updated <b>Table 12-15 Normal Power Supply Voltages</b> WS00008539
<b>Page 1343</b>	Removed <b>Section 13.2.2.4.5 Parameters at VDDP_EBU = 3.0 V</b> WS00008455
<b>Page 1314</b>	Updated <b>Section 13.2.1.2.3.1 Measured Parameters for VDDP_EBU = 1.8 V and 3 V</b> WS00008455
<b>Page 1309</b>	Removed VDDP_ETM from <b>Table 12-20 Pad Output Current</b> WS00008595
<b>Page 1309</b>	<b>Pad Characteristics</b> section moved to <b>Section 13.2.1.1.5</b> WS00008595
<b>Page 1302</b>	<b>Absolute Maximum Ratings</b> section moved to <b>Section 13.1.5</b> WS00008595
<b>Page 1344</b>	Added note to <b>Figure 13-20 SIM Timing</b> WS00008670
<b>Page 1305</b>	Ball names and some bit names changed and several cross-references added to <b>Table 12-17 Analog Power Supply Currents (IDD measured with VDD = 2.75 V)</b>
<b>Page 1385</b>	Updated <b>Table 12-57 AC/DC Characteristics</b> WS0008800

**CONFIDENTIAL**

<b>History</b>	
<b>Page 1323</b>	Updated <b>Table 12-23 EBU and PMCU Measured Parameters</b> WS00008783
<b>Page 1368</b>	Updated <b>Table 12-44 Electrical Characteristics of Microphone Supply</b> WS00008378
Changes for Rev. 1.05	
<b>Page 1316</b>	Updated <b>ts<sub>5_csx</sub></b> Maximum Limit WS00008897
<b>Page 1310</b>	X=A,B,C,D,E -> x=A,B
<b>Page 1318</b>	Updated <b>ts<sub>8_wr_f</sub></b> Maximum Limit WS00008899
<b>Page 1331</b> <b>Page 1332</b> <b>Page 1333</b> <b>Page 1334</b> <b>Page 1335</b> <b>Page 1337</b> <b>Page 1338</b>	CS1 removed from <b>Figure 13-9</b> , <b>Figure 13-10</b> , <b>Figure 13-11</b> , <b>Figure 13-12</b> , <b>Figure 13-13</b> <b>Figure 13-14</b> <b>Figure 13-15</b> WS00008783
Changes for Rev. 1.06	
<b>Page 1304</b>	Updated <b>Table 12-16 Digital Power Supply Currents</b> WS00009104
<b>Page 1315</b> <b>Page 1322</b> <b>Page 1343</b> <b>Page 1355</b> <b>Page 1357</b>	Updated <b>Table 12-23 EBU and PMCU Measured Parameters</b> , <b>Table 12-24 EBU and PMCU Derived Parameters</b> , <b>Table 12-25 SIM Timing</b> , <b>Table 12-32 Timing Characteristic of ASC with D Drivers in Synchronous Mode</b> and <b>Table 12-33 Timing Characteristic of SSC in Master Mode</b> WS00009104
<b>Page 1322</b>	Removed 7 parameters from <b>Table 12-24 EBU and PMCU Derived Parameters</b> WS00009104
<b>Page 1385</b>	Updated <b>Table 12-57 AC/DC Characteristics</b> WS0009106

**CONFIDENTIAL**

**Maximum Values (Destruction limits)**

*Note: All values mentioned in this chapter are target values and have to be confirmed.*

### 13.1 Maximum Values (Destruction limits)

This section contains the temperature, voltage, and ESD limit values.



**WARNING**

The maximum ratings may not be exceeded under any circumstances, not even momentarily and individually, as permanent damage to the device will result.

#### 13.1.1 Maximum ESD

**Table 12-10 Maximum ESD**

Parameter	Symbol	Limit Values		Unit
		Minimum	Maximum	
ESD		-	1000	V
ESD exception for certain pins			500	V

#### 13.1.2 Maximum Temperature

**Table 12-11 Maximum Temperature**

Parameter	Symbol	Limit Values		Unit
		Minimum	Maximum	
Temperature		-55	150	°C



**CONFIDENTIAL**

**Maximum Values (Destruction limits)**

### 13.1.3 Maximum Voltages

**Table 12-12 Maximum Power Supply Voltages**

Parameter	Symbol	Limit Values		Unit
		Minimum	Maximum	
<b>Digital Power Supply</b>	VDD_MAIN	-0.15	1.7	V
	VDD_DSP	-0.15	1.7	V
	VDD_RTC	-0.15	2.5	V
	VDDP_SIM	-0.3	3.6	V
	VDD_PLL	-0.15	1.7	V
	VDDP_EBU	-0.15	3.6	V
	VDDP_DIGA	-0.3	3.6	V
	VDDP_DIGB	-0.3	3.6	V
<b>Analog Power Supply</b>	VDDDB	-0.15	3	V
	VDDVBR	-0.15	3	V
	VDDVBT	-0.15	3	V
	VDDD	-0.15	3	V
	VDDM	-0.15	3	V
	VDDBG	-0.15	3	V
	AGND	-0.15	0.15	V
	VREFn	-0.15	3	V

### 13.1.4 Maximum Current

Shorts at the output components of standard digital drivers may cause sink or source currents up to 100 mA per pin. Exposure to these currents may destroy power buses or pad cells inside the PMB7870.

The sum of the sink or source currents at all pads belonging to the same digital I/O supply domain ( $V_{DDP\_DIGA}$ ,  $V_{DDP\_DIGB}$ ,  $V_{DDP\_EB}$ ,  $V_{DDP\_SIM}$ ) must not exceed [TBD: 20mA] at any time.

The sum of sinked or sourced currents in the connection between the bumps and the balls must not exceed (TBD: 20 mA @ 2.7 V). This corresponds to a maximum thermal dissipation inside the chip of approximately. (TBD: 55 mW).

**CONFIDENTIAL**

**Maximum Values (Destruction limits)**

### 13.1.5 Absolute Maximum Ratings

**Table 12-13 Absolute Maximum Ratings,  $T_{AMB} = -30^{\circ}\text{C} \dots +85^{\circ}\text{C}$**

#	Parameter	Symbol	Limit Values		Unit	Remarks
			min	max		
1	Supply Voltages 1.5V	$V_{15}$	-0.3	1.6	V	
2	Supply Voltages 2.5V	$V_{25}$	-0.3	3.0	V	
3	Digital Input Voltage	$V_{I15}$	-0.3	3.0	V	CLK, DA, EN
4	Analog Input Voltage	$V_{A25}$	-0.3	3.0	V	A, AX, B, BX
5	Total Power Dissipation	$P_{tot}$		400	mW	
6	Junction Temperature	$T_j$		125	$^{\circ}\text{C}$	
7	Storage Temperature	$T_S$	-55	125	$^{\circ}\text{C}$	
8	Thermal Resistance (junction to ambient)	$R_{thJA}$		40	K/W	Soldered diepad
9	ESD-Integrity <sup>1)</sup>	$V_{ESD}$	1000		V	
10	ESD-Integrity	$V_{ESD2}$	500		V	RX1, RX1X, RX2, RX2X, RX3, RX3X, RX4, RX4X, TX1, TX2, XO, XOX
11	RX1/RX2 Receiver Input Level	$Pin_{RX1/RX2}$		+5	dBm	Single Ended, GSM850/E- GSM900
12	RX3/RX4 Receiver Input Level	$Pin_{RX3/RX4}$		+8	dBm	Single Ended, DCS1800/ GSM1900

<sup>1)</sup> HBM according MIL-Std 883D, method 3015.7, and EOS/ESD assn. Standard S5.1-1993- only CMOS input/output pins.

**CONFIDENTIAL**

**Normal Operation Values**

## 13.2 Normal Operation Values

### 13.2.1 Baseband Electrical Data

#### 13.2.1.1 Static (DC Characteristics)

##### 13.2.1.1.1 Temperature

**Table 12-14** contains the operating temperature range.

**Table 12-14 Operating Temperature**

Parameter	Symbol	Limit Values		Unit
		Minimum	Maximum	
Temperature		-30	+85	°C

##### 13.2.1.1.2 Voltages

**Table 12-15** contains the normal operating voltage range of the power supplies.

**Table 12-15 Normal Power Supply Voltages**

Parameter	Symbol	Limit Values			Unit
		Minimum	Typical	Maximum	
<b>Digital Power Supply</b>	VDD_MAIN	1.35	1.5	1.6	V
	VDD_DSP	1.35	1.5	1.6	V
	VDD_RTC	If VDD_MAIN = 0 then 1, else 1.8	2	2.25	V
	VDDP_SIM	1.7	2.5	3.3	V
	VDD_PLL	1.35	1.5	1.6	V
	VDDP_EBU	1.62 2.7	1.8 3.0	1.98 3.3	V
	VDDP_DIGA	1.7	2.3	2.9	V
	VDDP_DIGB	1.7	2.3	2.9	V
<b>Analog Power Supply</b>	VDDDB	2.25	2.5	2.75	V
	VDDVBR	2.25	2.5	2.75	V
	VDDVBT	2.25	2.5	2.75	V
	VDDD	2.25	2.5	2.75	V

**CONFIDENTIAL**

**Normal Operation Values**

**Table 12-15 Normal Power Supply Voltages (cont'd)**

Parameter	Symbol	Limit Values			Unit
		Minimum	Typical	Maximum	
	VDDM	2.25	2.5	2.75	V
	VDDBG	2.25	2.5	2.75	V

### 13.2.1.1.3 Currents

**Table 12-16** and **Table 12-17** contain the input Power Supply currents.

**Table 12-16 Digital Power Supply Currents**

Parameter	Ball Name	Limit Values			Unit
		Minimum	Typical	Maximum	
With CPU Master Clock running at 52 MHz	VDD_MAIN		60 <sup>1) 2)</sup>		mA
With CPU Master Clock running at 26 MHz			30 <sup>1) 2)</sup>		mA
With CPU Master Clock running at 32 KHz			0.16 <sup>1) 2)</sup>		mA
With DSP Master Clock running at 104 MHz	VDD_DSP		75.3 <sup>1) 2)</sup>		mA
With CPU Master Clock running at 26MHz and the RTC reference clock 32 kHz	VDD_RTC		0.5		mA
With CPU Master Clock running at 32 kHz and the RTC reference clock 32 kHz		0.003			mA
	VDDP_SIM				mA
	VDD_PLL				mA
	VDDP_EBU				mA
	VDDP_DIGA				mA
	VDDP_DIGB				mA

<sup>1)</sup> These values are measured in a specific way and under a specific configuration.

<sup>2)</sup> These values are not guaranteed.

**CONFIDENTIAL**

**Normal Operation Values**

**Table 12-17 Analog Power Supply Currents (IDD measured with VDD = 2.75 V)**

Ball Name	Limit Values			Unit	Comments
	Minimum	Typical	Maximum		
VDDBB (Transmitter active)	1.7	3.5	4.2	mA	TXONMOD = 1 clk_bbtX = 13 MHz <b>AFE_VTXCTRL.TXMODE</b> = 01 DC digital input = half of fullscale
VDDBB (Receiver active, Standard mode)	5.1	6.4	7.7	mA	DC analog input clk_bbrX = 13 MHz RXON = 1 <b>BB_CTRL.BB_ADCMODE</b> = 0
VDDBB (Receiver active, Enhanced mode)	13	16.5	20.5	mA	DC analog input clk_bbrX = 26 MHz RXON = 1 <b>BB_CTRL.BB_ADCMODE</b> = 1
VDDBB (Power Amplifier)	0.3	0.75	1.1	mA	TXON = 1 clk_pa1 = 6.5 MHz clk_par = 6.5 MHz DC digital input = half of full scale
VDDVBT (Only Voiceband Receive DAC on)	2	2.5	3	mA	<b>AFE_VRXCTRL2.RXDAC</b> = 1 digital input = 0 F26M (XO) = 52 MHz
VDDVBT (DAC on + Highpower Output Amplifier on)	1.8	3.4	4.2	mA	<b>AFE_VRXCTRL2.VEPPA</b> = 1 <b>AFE_VRXCTRL2.RXDAC</b> = 1 digital input = 0 F26M (XO) = 52 MHz
VDDVBT (DAC on + Highpower Output Amplifier on)	1.6	3	4.2	mA	<b>AFE_VRXCTRL1.EPSAV</b> = 1

**CONFIDENTIAL**

**Normal Operation Values**

**Table 12-17 Analog Power Supply Currents (IDD measured with VDD = 2.75 V)**

Ball Name	Limit Values			Unit	Comments
	Minimum	Typical	Maximum		
VDDVBT (DAC on + Low Power Output Amplifier on)	2	2.5	3	mA	<b>AFE_VRXCTRL2.RXDAC</b> = 1 digital input pattern F26M (XO) = 52 MHz
VDDVBT (Only Voiceband Receive ADC on)	0.8	1.6	2	mA	<b>AFE_VTXCTRL.TXMODE</b> = 01
VDDVBR (DAC on + Highpower Output Amplifier on)	5.4	6.8	8.2	mA	<b>AFE_VRXCTRL2.VEPPA</b> = 1 <b>AFE_VRXCTRL2.RXDAC</b> = 1 digital input pattern F26M (XO) = 52 MHz
VDDVBR (DAC on + Highpower Output Amplifier on)	1.6	3.2	4.2	mA	<b>AFE_VRXCTRL1.EPSAV</b> = 1
VDDVBR (DAC + Highpower Output Amplifier)	0.2	0.5	1.2	mA	<b>AFE_VRXCTRL2.VEPPA</b> = 1 <b>AFE_VRXCTRL2.RXDAC</b> = 1
VDDD	0	0.06	0.2	mA	Average of I_VDDD values in all above tests
VDDM	0.5	1	1.5	mA	<b>MEAS_CTRL2.ADCON</b> = 1
VDDBG	0.75	1.2	1.8	mA	<b>ANA_CTRL1.BG_PWUP</b> = 1 <b>AFE_VTXCTRL.VMIC</b> != 00 (Mic. supply voltage on) (wait 150 µs)

The mapping between the signals and the pad is in **Chapter 3 Pin Descriptions**.

**CONFIDENTIAL**

**Normal Operation Values**

### 13.2.1.1.4 Capacitances and Resistors

**Table 12-18 Input Capacitances and Resistors**

Parameter	Symbol	Limit Values			Unit
		Minimum	Typical	Maximum	
Digital I/O; DC value	$C_{I/O\_DIG}$			10	pF
Oscillator (F32K); Input driven	$C_{F32k}$			25	pF

*Note: These values are not measured by production tests or characterization procedures. They are derived from simulations.*

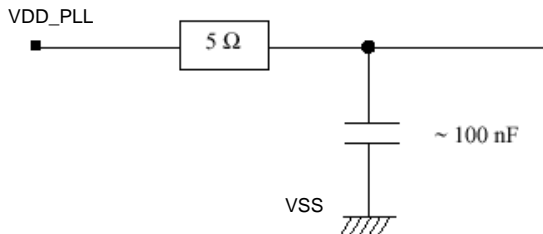
**Table 12-19 External Load Capacitances and Resistors**

Parameter	Symbol	Limit Values		Unit
		Minimum	Maximum	
I <sup>2</sup> C Signal / Clock Pins			50	pF
Oscillator (F32K, OSC32K) Values have to be calculated depending on the external quartz	$C_{32k}$	-	-	pF

#### Further Requirements and Notes:

- EBU:** For external pull up resistances and PCB integration guidelines on EBU pins (see [Section 10.13 External Bus Unit \(on Page 851\)](#) and [Section 13.2.1.2.3 EBU \(on Page 1314\)](#)).
- Mixed Signal and Analog Interfaces:** The description of external load conditions of the mixed signal interfaces and the measurement interface are contained in:
  - [Section 13.2.2.1.1 Audio Receive Part \(on Page 1362\)](#)
  - [Section 13.2.2.3 Baseband Receive Unit \(on Page 1371\)](#)
  - [Section 13.2.2.2 GMSK Modulator \(on Page 1369\)](#), etc.
- PLL:** For  $V_{DD\_PLL}$  an external filter must be applied. The blocking capacitor has to be located as close as possible to the pins  $V_{DD\_PLL}$ ,  $V_{SS\_PLL}$  (see [Figure 13-1](#)).

Figure 13-1 External Filter at PLL Supply



4. **I<sup>2</sup>C**: The dedicated I<sup>2</sup>C pads (pins SCL and SDA) have open drain functionality. Pull up resistors have to be implemented externally. Note that those signals also appear as alternative functions at standard I/O pads. Those pads can be configured by the port control logic to show OD functionality or active driver behavior. But those pads do not fulfill the I<sup>2</sup>C standard.
5. **Keypads**: The total sum of possible source currents from the keypad outputs to V<sub>SS</sub> should be restricted to maximum 20 mA Dynamic.



**CONFIDENTIAL**

**Normal Operation Values**

**13.2.1.1.5 Pad Characteristics**

**Table 12-20 Pad Output Current**

Parameter	Symbol	Limit Values		Unit	Test Condition
		Minimum	Maximum		
Digital Pins in the I/O supply domain of the EBU					
EBU: L-input voltage for memory Interface	V <sub>IL_PEBU</sub>	- 0.2	0.3	V	
EBU: H-input voltage for memory Interface	V <sub>IH_PEBU</sub>	V <sub>DDP_EBU</sub> - 0.3	V <sub>DDP_EBU</sub> + 0.2	V	
EBU: L-output voltage for pad classes B & C of Memory Interface at a load current of 100 μA	V <sub>OL_PEBU</sub>		V <sub>SSP_EBU</sub> + 0.1	V	I <sub>OH_PEBU</sub> = +100 μA C) 1)
EBU: H-output voltage for pad classes B & C of memory interface at a load current of 100 μA	V <sub>OH_PEBU</sub>	V <sub>DDP_EBU</sub> - 0.1		V	I <sub>OH_PEBU</sub> = -100 μA C) 1)
Input/Output leakage current	I <sub>IZ</sub>		0,7	μA	0.2 V < V <sub>IN</sub> < V <sub>IHmax</sub>
Digital Pins in the I/O supply domain of the SIM Interface					
L-input voltage for SIM Interface	V <sub>IL_SIM</sub>	0	0.2 * V <sub>DDP_SIM</sub>	V	
L-input for SIM Interface	V <sub>IL_SIM</sub>		0.37	V	V <sub>DDP_SIM</sub> = 1.76 V
L-Input for SIM Interface	V <sub>IL_SIM</sub>		0.60	V	V <sub>DDP_SIM</sub> = 2.96 V
H-input voltage for SIM Interface	V <sub>IH_SIM</sub>	0.7 x V <sub>DDP_SIM</sub>	3.3	V	
H-input for SIM Interface	V <sub>IH_SIM</sub>	1.22		V	V <sub>DDP_SIM</sub> = 1.84 V
H-input voltage for SIM Interface	V <sub>IH_SIM</sub>	1.95		V	V <sub>DDP_SIM</sub> = 2.96 V
Input/Output leakage current	I <sub>IZ</sub>		0,7	μA	0.2 V < V <sub>IN</sub> < V <sub>IHmax</sub>
Full Swing Input of RTC					
L-input voltage for F32K input (PU32KEN = 0)	V <sub>IL_32KFS</sub>	-0.25 V	+0.2 V	V	

**CONFIDENTIAL**

**Normal Operation Values**

**Table 12-20 Pad Output Current**

Parameter	Symbol	Limit Values		Unit	Test Condition
		Minimum	Maximum		
H-input voltage for F32K input (PU32KEN = 0)	$V_{IH\_32KFS}$	$V_{DD\_RTC} - 0.2$	$V_{DD\_RTC} + 0.25$	V	
<b>I<sup>2</sup>C Interface (only for special OD drivers, Not valid for alternative functions)</b>					
L-input voltage for inputs I2C_SCL and I2C_SDA (V <sub>DD</sub> -related levels)	$V_{ILI2C}$	-0.3	$0.3 \times V_{DDP\_DIGA}$	V	
H-input voltage for inputs I2C_SCL and I2C_SDA (V <sub>DDP</sub> -related levels)	$V_{IHI2C}$	$0.7 \times V_{DDP\_DIGA}$	$\max[V_{DDP\_DIGA} + 0.5, 3.3 \text{ V}]$	V	
Hysteresis for inputs I2C_SCL and I2C_SDA	$V_{HI2C}$	$0.05 \times V_{DDP\_DIGA}$			In accordance with I2C bus specification
L-output voltage for outputs I2C_SCL and I2C_SDA	$V_{OLI2C}$		0.4 V		$I_{OL} = +3.0 \text{ mA}$
Input/Output leakage current	$I_Z$			μ A	$0.2 \text{ V} < V_{IN} < V_{IHmax}$
<b>Standard Digital I/Os in I/O Domains VDDP_DIGx, x=A,B; standard digital I/Os in domain VDD_RTC (only valid in the range of 1,8V&lt;VDD_RTC&lt;2,25V)</b> <b>Not valid for special OD pads of I<sup>2</sup>C interface and pins F32k, OSC32k.</b>					
L-input voltage for general digital pads (except OD pads for I <sup>2</sup> C)	$V_{IL}$	- 0.2	$0.2 \times V_{DDP\_DIG}$	V	
H-input voltage for general digital pads (except OD pads I2C_SCL, I2C_SDA)	$V_{IH}$	$0.7 \times V_{DDP\_DIG}$	3.3	V	
L-output voltage for pad class B	$V_{OLB}$		0.2	V	$I_{OL} = +2.5 \text{ mA P)}$
			0.35	V	$I_{OL} = +5.0 \text{ mA C)}$
L-output voltage for pad class C	$V_{OLC}$		0.2	V	$I_{OL} = +2.0 \text{ mA P)}$
			0.35	V	$I_{OL} = +4.0 \text{ mA C)}$
L-output voltage for pad class D	$V_{OLD}$		0.2	V	$I_{OL} = +1.0 \text{ mA P)}$
			0.35	V	$I_{OL} = +2.0 \text{ mA C)}$
L-output voltage for pad class E/F	$V_{OLE}$		0.2	V	$I_{OL} = +1.0 \text{ mA P)}$
			0.35	V	$I_{OL} = +1.5 \text{ mA C)}$
H-output voltage for pad class B slow	$V_{OHBslow}$				$I_{OH} = -15 \text{ mA}$ $V_{DDP\_DIGB} = 1.7 \text{ V C)}$
H-output voltage for pad class B slow	$V_{OHBslow}$				$I_{OH} = -10 \text{ mA}$ $V_{DDP\_DIGB} = 1.7 \text{ V C)}$

**CONFIDENTIAL**

**Normal Operation Values**

**Table 12-20 Pad Output Current**

Parameter	Symbol	Limit Values		Unit	Test Condition
		Minimum	Maximum		
H-output voltage for pad class B	$V_{OHB}$	$V_{DDP\_DIG} - 0.35$		V	$I_{OH} = -5.0 \text{ mA C)}$
		$V_{DDP\_DIG} - 0.2$			$I_{OH} = -2.5 \text{ mA P)}$
H-output voltage for pad class C	$V_{OHC}$	$V_{DDP\_DIG} - 0.35$		V	$I_{OH} = -4.0 \text{ mA C)}$
		$V_{DDP\_DIG} - 0.2$		V	$I_{OH} = -2.0 \text{ mA P)}$
H-output voltage for pad class D	$V_{OHD}$	$V_{DDP\_DIG} - 0.35$		V	$I_{OH} = -2.0 \text{ mA C)}$
		$V_{DDP\_DIG} - 0.2$		V	$I_{OH} = -1.0 \text{ mA P)}$
H-output voltage for pad class E/F	$V_{OHE}$	$V_{DDP\_DIG} - 0.35$		V	$I_{OH} = -1.5 \text{ mA C)}$
		$V_{DDP\_DIG} - 0.2$		V	$I_{OH} = -1.0 \text{ mA P)}$
Input/Output leakage current	$I_{IZ}$	I	$\pm 0.7$	$\mu\text{A}$	$0.2 \text{ V} < V_{IN} < V_{IHmax}$

<sup>1)</sup> This value only applies to the DC characteristics. AC characteristics, such as ripples and switching noise, are not included in this definition.

Table legend:

P) Production test program

C) Characterization test program

### I/O Supply Domains

All digital I/O supply domains of the E-GOLDradio (including the RTC) have to be operated under operating conditions. It is not possible to power down pad domains separately.

*Note: All values in [Table 12-20](#) and [Table 12-21](#) are derived from simulations.*

**Table 12-21 Pull-Up/Pull-Down Currents**

Parameter	Symbol	Limit Values		Unit	Test Condition $V_{S_d}, V_{DD\_PAD}$
		Minimum	Maximum		
Pull-up input current for pull class A (1)	$I_{PUa}$		-450 -150	$\mu\text{A}$	0 V, $V_{DD\_PAD} = 3.30 \text{ V C}$ 0 V, $V_{DD\_PAD} = 1.90 \text{ V P}$
		-40 -15		$\mu\text{A}$	1.90 V, $V_{DD\_PAD} = 2.75 \text{ V C}$ 1.20 V, $V_{DD\_PAD} = 1.70 \text{ V P}$
Pull-up input current for pull class B (2)	$I_{PUb}$		-100 -35	$\mu\text{A}$	0 V, $V_{DD\_PAD} = 3.30 \text{ V C)}$ 0 V, $V_{DD\_PAD} = 1.90 \text{ V P)}$
		-5 -1		$\mu\text{A}$	1.90 V, $V_{DD\_PAD} = 2.75 \text{ V C)}$ 1.20 V, $V_{DD\_PAD} = 1.70 \text{ V P)}$

**CONFIDENTIAL**

**Normal Operation Values**

**Table 12-21 Pull-Up/Pull-Down Currents**

Parameter	Symbol	Limit Values		Unit	Test Condition $V_{S_d}, V_{DD\_PAD}$
		Minimum	Maximum		
Pull-up input current for pull class C (3)	$I_{PUc}$		-30 -10	$\mu A$	0 V, $V_{DD\_PAD} = 3.30$ V C) 0 V, $V_{DD\_PAD} = 1.90$ V P)
		-0.5 -0.0		$\mu A$	1.90 V, $V_{DD\_PAD} = 2.75$ V C) 1.20 V, $V_{DD\_PAD} = 1.70$ V P)
Pull-down input current for pull class A (1)	$I_{PDa}$		+450 +200	$\mu A$	3.30 V, $V_{DD\_PAD} = 3.30$ V C) 1.90 V, $V_{DD\_PAD} = 1.90$ V P)
		+7 +2		$\mu A$	0.2 V, $V_{DD\_PAD} = 2.75$ V C) 0.2 V, $V_{DD\_PAD} = 1.70$ V P)
Pull-down input current for pull class B (2)	$I_{PDb}$		+100 +35	$\mu A$	3.30 V, $V_{DD\_PAD} = 3.30$ V C) 1.90 V, $V_{DD\_PAD} = 1.90$ V P)
		+1.0 +0.5		$\mu A$	0.2 V, $V_{DD\_PAD} = 2.75$ V C) 0.2 V, $V_{DD\_PAD} = 1.70$ V P)
Pull-down input current for pull class C (3)	$I_{PDc}$		+30 +10	$\mu A$	3.30 V, $V_{DD\_PAD} = 3.30$ V C) 1.90 V, $V_{DD\_PAD} = 1.90$ V P)
		+0.5 +0.0		$\mu A$	0.2 V, $V_{DD\_PAD} = 2.75$ V C) 0.2 V, $V_{DD\_PAD} = 1.70$ V P)

### 13.2.1.2 Dynamic (AC Characteristics)

**Warning: F26M now refers to the XO pad. The major 26 MHz reference clock is now generated by the internal RF oscillator.**

#### 13.2.1.2.1 Rise and Fall Time of Signals

The level reached at a certain time is basically determined by following components:

- Supply voltage drop from ideal voltage at pad supply pin to pad due to:
  - Noise on power supply (determined in the application)
  - Voltage drop over pad supply rail: max. 30 mV
- Pad resistance  $R_i$  and external load C (RC loading curve with timing constant  $R_i * C$ ).

**Table 12-22** shows the resistances  $R_i$  for the different driver classes.

**Table 12-22 Pad Resistances (1,75 V to 2,75 V)**

Parameter 2,25-2,75 V	Symbol	Limit Values			Unit
		minimum	typical	maximum	

**CONFIDENTIAL**

**Normal Operation Values**

**Table 12-22 Pad Resistances (1,75 V to 2,75 V)**

Pad resistance at 2.5 mA - 5 mA load / rising edge for Pad Class B	RiB_r			50	$\Omega$
Pad resistance at 2.5 mA - 5 mA load/falling edge for Pad Class B	RiB_f			50	$\Omega$
Pad resistance at 2.5 mA - 5 mA load/rising edge for Pad Class Bslow	RiBslow_r			70	$\Omega$
Pad resistance at 2.5 mA - 5 mA load/falling edge for Pad Class Bslow	RiBslow_f			70	$\Omega$
Pad resistance at 2 mA - 4 mA load/rising edge for Pad Class C	RiC_r			70	$\Omega$
Pad resistance at 2 mA - 4 mA load/falling edge for Pad Class C	RiC_f			70	$\Omega$
Pad resistance at 1 mA - 2 mA load/rising edge for Pad Class D	RiD_r			115	$\Omega$
Pad resistance at 1 mA - 2 mA load/falling edge for Pad Class D	RiD_f			115	$\Omega$
Pad resistance at 1 mA - 1.5 mA load/rising edge for Pad Class E	RiE_r			130	$\Omega$
Pad resistance at 1 mA - 1.5 mA load/falling edge for Pad Class E	RiE_f			120	$\Omega$
Pad resistance at 1 mA - 1.5 mA load/rising edge for Pad Class F	RiF_r			180	$\Omega$
Pad resistance at 1 mA - 1.5 mA load/falling edge for Pad Class F	RiF_f			180	$\Omega$

### 13.2.1.2.2 Testing Waveforms

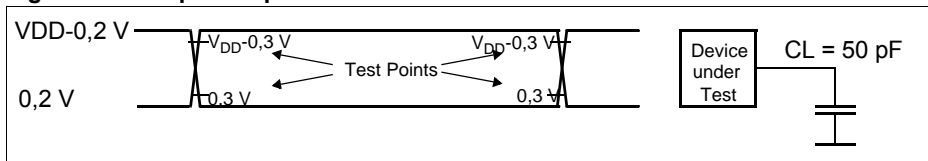
#### 13.2.1.2.2.1 Testing Driven Values for All Digital I/Os Except EBU\_AHB I/Os

Input signals during AC testing are driven to  $V_{DD} - 0.2$  V for a logical 1 and to 0.2 V for a logical 0.

Output signals during AC testing are measured at  $V_{DD} - 0.3$  V for a logical 1 and at 0.3 V for a logical 0.

The AC testing input/output waveforms are shown in [Table 13-2](#).

**Figure 13-2 Input Output Waveform for AC Test**



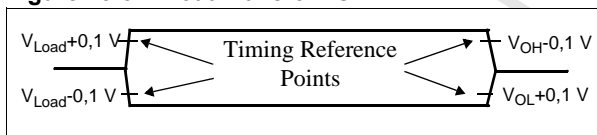
#### 13.2.1.2.2.2 Testing Float Waveforms

A load voltage  $V_{LOAD}$  and a load current ( $I_{OH} = -5 \text{ mA}$ ,  $I_{OL} = 5 \text{ mA}$ ) are applied to the pin during the tristate phases (see [Figure 13-3](#)).

The end of floating is detected at the latest when a 100 mV change from  $V_{LOAD}$  occurs.

The begin of floating is detected at the latest when a 100 mV change from the loaded  $V_{OH}/V_{OL}$ -level occurs.

**Figure 13-3 Float Waveforms**



#### 13.2.1.2.3 EBU

##### 13.2.1.2.3.1 Measured Parameters for VDDP\_EBU = 1.8 V and 3 V

**Note:** The values in Table 13-15 and Table 13-16 will be validated and updated when the E-GOLDradio characterization is finished.

In [Table 12-23](#) BUSCON refers only to [BUSCON0 \(on Page 870\)](#) and [BUSCON1](#). In this table the limits are the same for both VDDP\_EBU voltages

The characterization for EBU and PMCU is done at VDDP\_EBU = 1.8 V and 3 V. These results give the same limits for both voltages.

**Table 12-23 EBU and PMCU Measured Parameters**

Parameter Symbol	Timing Diagram	Limits		Unit	
		Minimum	Maximum		
Symbol Description					
$ts_{1\_adv\_f\_pmc}$	Page 1332		28.0	ns	
ADV_n low after F26M (XO) rising edge in <b>pagemode during Nth (with N = 1) offpage access:</b> <ul style="list-style-type: none"><li>Without Read-Write Delay (BUSCON.RWD = 1)</li><li>Without extended ALE (BUSCON.ALE = 0)</li></ul>					
$ts_{1\_adv\_r\_pmc}$	Page 1332		28.0	ns	
ADV_n high after F26M (XO) falling edge in <b>pagemode during Nth (with N = 1) offpage access:</b> <ul style="list-style-type: none"><li>Without Read-Write Delay (BUSCON.RWD = 1)</li><li>Without extended ALE (BUSCON.ALE = 0)</li></ul>					
$ts_{2\_adv\_f\_pmc}$	Page 1337		28.0	ns	
ADV_n low after F26M (XO) falling edge in <b>pagemode during Nth (with N = 1) offpage access:</b> <ul style="list-style-type: none"><li>With Read-Write Delay (BUSCON.RWD = 0)</li><li>Without extended ALE (BUSCON.ALE = 0)</li></ul>					
$ts_{2\_adv\_r\_pmc}$	Page 1337		28.0	ns	
ADV_n high after F26M (XO) rising edge in <b>pagemode during Nth (with N = 1) offpage access:</b> <ul style="list-style-type: none"><li>With Read-Write Delay (BUSCON.RWD = 0)</li><li>Without extended ALE (BUSCON.ALE = 0)</li></ul>					
$ts_{3\_adv\_f\_pmc}$	Page 1339		28.0	ns	
ADV_n low after F26M (XO) rising edge in <b>pagemode during Nth (with N = 1) offpage access:</b> <ul style="list-style-type: none"><li>Without Read-Write Delay (BUSCON.RWD = 1)</li><li>With extended ALE (BUSCON.ALE = 1)</li></ul>					
$ts_{3\_adv\_r\_pmc}$	Page 1339		28.0	ns	
ADV_n high after F26M (XO) falling edge in <b>pagemode during Nth (with N = 1) offpage access:</b> <ul style="list-style-type: none"><li>Without Read-Write Delay (BUSCON.RWD = 1)</li><li>With extended ALE (BUSCON.ALE = 1)</li></ul>					

**CONFIDENTIAL**

**Normal Operation Values**

**Table 12-23 EBU and PMCU Measured Parameters**

Parameter Symbol	Timing Diagram	Limits		Unit	
		Minimum	Maximum		
Symbol Description					
$ts_{4\_adv\_f\_pmc}$	Page 1341		28.0	ns	
ADV_n low after F26M (XO) falling edge in <b>pagemode</b> during Nth (with N = 1) <b>offpage</b> access:					
<ul style="list-style-type: none"><li>• With Read-Write Delay (BUSCON.RWD = 0)</li><li>• With extended ALE (BUSCON.ALE = 1)</li></ul>					
$ts_{4\_adv\_r\_pmc}$	Page 1341		28.0	ns	
ADV_n high after F26M (XO) rising edge in <b>pagemode</b> during Nth (with N = 1) <b>offpage</b> access:					
<ul style="list-style-type: none"><li>• With Read-Write Delay (BUSCON.RWD = 0)</li><li>• With extended ALE (BUSCON.ALE = 1)</li></ul>					
$ts_{1\_2\_adv\_f\_pmc}$	Page 1334		28.0	ns	
ADV_n low after F26M (XO) falling edge in <b>pagemode</b> during Nth (with N > 1) <b>offpage</b> access:					
<ul style="list-style-type: none"><li>• With or without Read-Write Delay</li><li>• With or without extended ALE</li></ul>					
$ts_{4\_data\_fl\_e}$	Page 1326		20	ns	
Write Data valid after F26M (XO) falling edge:					
<ul style="list-style-type: none"><li>• First switching data bit from Z to valid)</li><li>• Without or with Early Write (BUSCON.EW = 0)</li><li>• With or without Read-Write Delay (BUSCON.RWD = 0 or 1)</li></ul>					
$ts_{5\_addr}$	Page 1326		30.0	ns	
Address valid after F26M (XO) falling edge:					
<ul style="list-style-type: none"><li>• Last switching address bit from invalid to valid</li></ul>					
$ts_{5\_csx}$	Page 1326		24.0 (Except for pad OE_N & T_OUT11)	ns	
CSx_n valid after F26M (XO) falling edge:					
<ul style="list-style-type: none"><li>• Early CS (SYSCON.CSCFG = 1)</li></ul>					
$ts_{5\_oen}$			28.0	ns	
oen valid after F26M (XO) falling edge:					
<ul style="list-style-type: none"><li>• Early CS (SYSCON.CSCFG = 1)</li></ul>					



**CONFIDENTIAL**

**Normal Operation Values**

**Table 12-23 EBU and PMCU Measured Parameters**

Parameter Symbol	Timing Diagram	Limits		Unit	
		Minimum	Maximum		
Symbol Description					
$ts_{5\_tout11}$			28.0	ns	
tout11 valid after F26M (XO) falling edge: <ul style="list-style-type: none"><li>• Early CS (<b>SYSCON.CSCFG</b> = 1)</li></ul>					
$ts_{5\_csx\_pmc}$	Page 1332		24.0	ns	
Flash CSx_n valid after F26M (XO) falling edge in <b>pagemode</b> : <ul style="list-style-type: none"><li>• With x = 0 if Flash is on CS0 and x = 1 if flash on CS1</li><li>• Early CS (<b>SYSCON.CSCFG</b> = 1, <b>PMC_TIMER0/1.EN</b> = 0, <b>PMC0/1.PAEN</b> = 1, <b>PMC0/1.MEMCS</b> = 1)</li></ul>					
$ts_{5\_data}$	Page 1326		22	ns	
Write Data valid after F26M (XO) falling edge: <ul style="list-style-type: none"><li>• Last switching data bit from Z to valid</li><li>• Without or with Early Write (<b>BUSCON.EW</b> = 0)</li><li>• With or without Read-Write Delay (<b>BUSCON.RWD</b> = 0 or 1)</li></ul>					
$ts_{6\_csx}$	Page 1328		28.0	ns	
CSx_n valid after F26M (XO) rising edge: <ul style="list-style-type: none"><li>• Normal CS (<b>SYSCON.CSCFG</b> = 0)</li></ul>					
$ts_{6\_oen}$			28.0	ns	
oen valid after F26M (XO) rising edge: <ul style="list-style-type: none"><li>• Early CS (<b>SYSCON.CSCFG</b> = 1)</li></ul>					
$ts_{6\_tout11}$			28.0	ns	
tout11 valid after F26M (XO) falling edge: <ul style="list-style-type: none"><li>• Early CS (<b>SYSCON.CSCFG</b> = 1)</li></ul>					
$ts_{7\_wr\_f}$	Page 1327		23.0	ns	
WR_n valid after F26M (XO) rising edge: <ul style="list-style-type: none"><li>• Without Read-Write Delay (<b>BUSCON.RWD</b> = 1)</li></ul>					
$ts_{7\_wr\_r}$	Page 1326		23.0	ns	
WR_n valid after F26M (XO) rising edge: <ul style="list-style-type: none"><li>• Without Read-Write Delay (<b>BUSCON.RWD</b> = 1)</li></ul>					
$ts_{7\_rd\_f}$	Page 1327		20.0	ns	
WR_n invalid after F26M (XO) rising edge: <ul style="list-style-type: none"><li>• With Early Write (<b>BUSCON.EW</b> = 1)</li></ul>					

**CONFIDENTIAL**

**Normal Operation Values**

**Table 12-23 EBU and PMCU Measured Parameters**

Parameter Symbol	Timing Diagram	Limits		Unit	
		Minimum	Maximum		
Symbol Description					
<i>ts7_rd_f_pmc</i>	Page 1332		25.0	ns	
RD_n valid after F26M (XO) rising edge in <b>pagemode</b> : <ul style="list-style-type: none"><li>Without Read-Write Delay (BUSCON.RWD = 1)</li><li>Without extended ALE (BUSCON.ALE = 0)</li></ul>					
<i>ts7_oe_f_pmc</i>	Page 1332		28.0	ns	
OE_n valid after F26M (XO) rising edge in <b>pagemode</b> : <ul style="list-style-type: none"><li>Without Read-Write Delay (BUSCON.RWD = 1)</li><li>Without extended ALE (BUSCON.ALE = 0)</li></ul>					
<i>ts7_rd_r_pmc</i>	Page 1332		31.0	ns	
RD_n invalid after F26M (XO) falling edge in <b>pagemode</b> : <ul style="list-style-type: none"><li>With or without Read-Write Delay</li><li>With or without extended ALE</li></ul>					
<i>ts7_oe_r_pmc</i>	Page 1332		31.0	ns	
OE_n invalid after F26M (XO) falling edge in <b>pagemode</b> : <ul style="list-style-type: none"><li>With or without Read-Write Delay</li><li>With or without extended ALE</li></ul>					
<i>ts8_rd_f</i>	Page 1326		20.0	ns	
RD_n valid after F26M (XO) falling edge: <ul style="list-style-type: none"><li>With Read-Write Delay (BUSCON.RWD = 0)</li></ul>					
<i>ts8_rd_r</i>	Page 1326		20.0	ns	
RD_n invalid after F26M (XO) falling edge: <ul style="list-style-type: none"><li>With or without Read-Write Delay (BUSCON.RWD = 0 or 1)</li></ul>					
<i>ts8_wr_f</i>	Page 1326		26.0	ns	
WR_n valid after F26M (XO) falling edge: <ul style="list-style-type: none"><li>With Read-Write Delay (BUSCON.RWD = 0)</li></ul>					
<i>ts8_rd_f_pmc</i>	Page 1337		25.0	ns	
RD_n valid after F26M (XO) falling edge in <b>pagemode</b> : <ul style="list-style-type: none"><li>With Read-Write Delay (BUSCON.RWD = 0)</li><li>Without extended ALE (BUSCON.ALE = 0)</li></ul>					

**CONFIDENTIAL**

**Normal Operation Values**

**Table 12-23 EBU and PMCU Measured Parameters**

Parameter Symbol	Timing Diagram	Limits		Unit	
		Minimum	Maximum		
Parameter Symbol	Symbol Description				
<i>ts8_oe_f_pmc</i>	Page 1337		28.0	ns	
	OE_n valid after F26M (XO) falling edge <b>in pagemode</b> : <ul style="list-style-type: none"><li>• With Read-Write Delay (BUSCON.RWD = 0)</li><li>• Without extended ALE (BUSCON.ALE = 0)</li></ul>				
<i>ts9_rd_f_pmc</i>	Page 1339		31.0	ns	
	RD_n valid after F26M (XO) rising edge <b>in pagemode</b> : <ul style="list-style-type: none"><li>• Without Read-Write Delay (BUSCON.RWD = 1)</li><li>• With extended ALE (BUSCON.ALE = 1)</li></ul>				
<i>ts9_oe_f_pmc</i>	Page 1339		31.0	ns	
	OE_n valid after F26M (XO) rising edge <b>in pagemode</b> : <ul style="list-style-type: none"><li>• Without Read-Write Delay (BUSCON.RWD = 1)</li><li>• With extended ALE (BUSCON.ALE = 1)</li></ul>				
<i>ts9_data</i>	Page 1329	Not measurable		ns	
	Read Data invalid after F26M (XO) falling edge <ul style="list-style-type: none"><li>• First switching data bit from valid to Z.</li></ul>				
<i>ts10_rd_f_pmc</i>	Page 1341		25.0	ns	
	RD_n valid after F26M (XO) falling edge <b>in pagemode</b> : <ul style="list-style-type: none"><li>• With Read-Write Delay (BUSCON.RWD = 0)</li><li>• With extended ALE (BUSCON.ALE = 1)</li></ul>				
<i>ts10_oe_f_pmc</i>	Page 1341		28.0	ns	
	OE_n valid after F26M (XO) falling edge <b>in pagemode</b> : <ul style="list-style-type: none"><li>• With Read-Write Delay (BUSCON.RWD = 0)</li><li>• With extended ALE (BUSCON.ALE = 1)</li></ul>				
<i>ts10_data</i>	Page 1329		5.0	ns	
	Read Data valid before F26M (XO) falling edge <ul style="list-style-type: none"><li>• Last switching data bit from Z to valid.</li></ul>				
<i>ts11_addr</i>	Page 1326		22.0	ns	
	Address invalid after F26M (XO) falling edge: <ul style="list-style-type: none"><li>• First switching address bit from valid to invalid</li></ul>				

**CONFIDENTIAL**

**Normal Operation Values**

**Table 12-23 EBU and PMCU Measured Parameters**

	Timing Diagram	Limits		Unit	
		Minimum	Maximum		
Parameter Symbol	Symbol Description				
$ts_{11\_csx}$	Page 1328		28.0	ns	
	CSx_n invalid after F26M (XO) rising edge: <ul style="list-style-type: none"><li>• Early CS (<b>SYSCON.CSCFG</b> = 1)</li></ul>				
$ts_{11\_oen}$			28.0	ns	
	oen valid after F26M (XO) falling edge: <ul style="list-style-type: none"><li>• Early CS (<b>SYSCON.CSCFG</b> = 1)</li></ul>				
$ts_{11\_tout11}$			28.0	ns	
	tout11 valid after F26M (XO) rising edge: <ul style="list-style-type: none"><li>• Early CS (<b>SYSCON.CSCFG</b> = 1)</li></ul>				
$ts_{11\_2\_csx}$	Page 1326		31.0	ns	
	CSx_n invalid after F26M (XO) falling edge: <ul style="list-style-type: none"><li>• Normal C (<b>SYSCON.CSCFG</b> = 0)</li></ul>				
$ts_{11\_2\_oen}$			31.0	ns	
	oen valid after F26M (XO) falling edge: <ul style="list-style-type: none"><li>• Early CS (<b>SYSCON.CSCFG</b> = 1)</li></ul>				
$ts_{11\_2\_tout11}$			31.0	ns	
	tout11 valid after F26M (XO) falling edge: <ul style="list-style-type: none"><li>• Early CS (<b>SYSCON.CSCFG</b> = 1)</li></ul>				
$ts_{11\_csx\_pmc}$	Page 1332		31.0	ns	
	Flash CSx_n invalid after F26M (XO) falling edge in <b>pagemode</b> : <ul style="list-style-type: none"><li>• With x = 0 if Flash is on CS0 and x = 1 if flash on CS1):</li><li>• Early CS (<b>SYSCON.CSCFG</b> = 1, <b>PMC_TIMER0/1.EN</b> = 0, <b>PMC0/1.PAEN</b> = 1, <b>PMC0/1.MEMCS</b> = 1)</li></ul>				
$ts_{12\_data\_f}$	Page 1326	Not measurable		ns	
	Write Data invalid after F26M (XO) falling edge: <ul style="list-style-type: none"><li>• First switching data bit from valid to Z</li><li>• With Early Write (<b>BUSCON.EW</b> = 1)</li></ul>				
$ts_{12\_data\_r}$	Page 1329	Not measurable		ns	
	Write Data invalid after F26M (XO) rising edge: <ul style="list-style-type: none"><li>• First switching data bit from valid to Z</li><li>• Without Early Write (<b>BUSCON.EW</b> = 0)</li></ul>				

**CONFIDENTIAL**

**Normal Operation Values**

**Table 12-23 EBU and PMCU Measured Parameters**

Parameter Symbol	Timing Diagram	Limits		Unit	
		Minimum	Maximum		
Symbol Description					
$ts_{13\_bus\_tris}$	Page 1326	Not measurable		ns	
Write Data invalid after F26M (XO) rising edge: <ul style="list-style-type: none"><li>• Last switching data bit from valid to Z</li><li>• With Early Write (BUSCON.EW = 1)</li></ul>					
$ts_{18\_wr}$	Page 1329		25.0	ns	
WR_n invalid after F26M (XO) falling edge: <ul style="list-style-type: none"><li>• Without Early Write (BUSCON.EW = 0)</li></ul>					
$ts_{19\_csx\_pmc}$			25.0	ns	
Flash CSx_n invalid after F26M (XO) falling edge in <b>pagemode</b> : <ul style="list-style-type: none"><li>• With x = 0 if Flash is on CS0 and x = 1 if flash on CS1</li><li>• Early CS (<b>SYSCON.CSCFG</b> = 1, <b>PMC_TIMER0/1.EN</b> = 1, <b>PMC0/1.PAEN</b> = 1, <b>PMC0/1.MEMCS</b> = 1)</li></ul>					
$ts_{0\_clk}$	Page 1326	Not measured		ns	
CLKOUT rising edge after F26M (XO) falling edge with: <ul style="list-style-type: none"><li>• <b>MST_CLK_CTRL.CPUH</b> = 0</li><li>• <b>MST_CLK_CTRL.CPUPRE</b> = 0</li><li>• <b>MST_CLK_CTRL.CLK_OUTL_EN</b> = 10</li></ul>					

If **SYSCON.WRCFG** = 1, pin  $\overline{WR}$  acts as  $\overline{WRL}$  and pin  $\overline{BHE}$  acts as  $\overline{WRH}$ , else pin  $\overline{WR}$  and  $\overline{BHE}$  retain their normal function.

If **BUSCONx.CSWENx** = 1, pin  $\overline{CSx}$  acts as  $\overline{WRCSx}$ , else pin  $\overline{CSx}$  retains its normal function.

**CONFIDENTIAL**

**Normal Operation Values**

**13.2.1.2.3.2 Derived Parameters for VDDP\_EBU = 1.8 and 3 V**

The timing parameters in **Table 12-24** are not characterized directly, but calculated from **Table 12-23 EBU and PMCU Measured Parameters (on Page 1315)**.

**Table 12-24 EBU and PMCU Derived Parameters**

Symbol Descriptions	Parameter Symbol	Limits		Unit
		Minimum	Maximum	
ADV plus data set-up ( $ts_{1\_adv\_f\_pmc} + ts_{10\_data}$ )	$ts_{20}$		28	ns
Early $\overline{CS}$ plus data set-up ( $ts_{5\_csx} + ts_{10\_data}$ )	$ts_{21}$		25 (Except for pad T_OUT11)	ns
Normal $\overline{CS}$ plus data set-up ( $ts_{6\_csx} + ts_{10\_data}$ )	$ts_{22}$		28	ns
Address plus data set-up ( $ts_{5\_addr} + ts_{10\_data}$ )	$ts_{23}$		30 (Except for pad T_OUT11)	ns
Early write timing ( $ts_{5\_data} - ts_{7\_wr\_r}$ )	$ts_{24}$		5	ns
Normal write referred to chip select ( $ts_{5\_data} - ts_{11\_csx}$ )	$ts_{25}$		5	ns
Read enable timing with RW delay ( $ts_{8\_rd\_f} + ts_{10\_data}$ )	$ts_{28}$		22	ns
Address hold after write ( $ts_{11\_addr} - ts_{18\_wr}$ ) <sup>1)</sup>	$ts_{30}$		2	ns
Write within chip select ( $ts_{11\_csx} - ts_{18\_wr}$ ) <sup>1)</sup> <i>Note: This timing is measured between Vdd/2</i>	$ts_{32}$		23	ns
Data write read spacing ( $ts_{8\_rd\_f} - ts_{13\_data\_fl\_b}$ )	$ts_{33}$		10	ns
Data hold to ADDR/RD/CS ( $ts_{10\_data} - \text{MIN} [ts_{5\_addr}, ts_{8\_rd\_r}, ts_{5\_csx}]$ )	$ts_{34}$		-15 (Except for pad T_OUT11)	ns
Address hold after write ( $ts_{7\_wr\_r} - ts_{11\_addr}$ )	$ts_{36}$		2	ns
$\overline{RD}$ high to high Z ( $ts_{8\_rd\_r} - ts_{4\_data\_fl}$ )	$ts_{38}$		2	ns
$\overline{WR}$ begins high and $\overline{CS}$ begins high ( $ts_{7\_wr\_r} - ts_{11\_csx}$ )	$ts_{42}$		18	ns
Read cycle ( $ts_{5\_addr} - ts_{11\_addr}$ )	$ts_{44}$		12	ns

**CONFIDENTIAL**

**Normal Operation Values**

**Table 12-24 EBU and PMCU Derived Parameters**

Symbol Descriptions	Parameter Symbol	Limits		Unit
		Minimum	Maximum	
$\overline{\text{CS}}$ low to $\overline{\text{RD}}$ low for CS0,1,2 ( $ts_{5\_csx\_max} - ts_{7\_rd\_f\_min}$ )	$ts_{46}$		23	ns
$\overline{\text{CS}}$ low to $\overline{\text{RD}}$ low for CS3,4 ( $ts_{5\_csx\_max} - ts_{7\_rd\_f\_min}$ )	$ts_{46}$		23	ns
$\overline{\text{CS}}$ low to $\overline{\text{WR}}$ low ( $ts_{5\_csx\_max} - ts_{8\_wr\_f}$ )	$ts_{47}$		23	ns
ADDR valid to $\overline{\text{WR}}$ going high ( $ts_{5\_addr\_max} - ts_{20\_wr\_r}$ )	$ts_{48}$			ns
$\overline{\text{WR}}$ high to $\overline{\text{CS}}$ hold ( $ts_{7\_wr\_r} - ts_{5\_csx\_min}$ )	$ts_{49}$		18	ns
$\overline{\text{WR}}$ high pulse width ( $ts_{7\_wr\_r} - ts_{8\_wr\_f}$ )	$ts_{50}$		18	ns
$\overline{\text{WR}}$ high to $\overline{\text{RD}}$ going low ( $ts_{7\_wr\_r} - ts_{3}$ )	$ts_{51}$		13	ns
ADDR valid to $\overline{\text{WR}}$ going low ( $ts_{5\_addr\_max} - ts_{8\_wr\_f\_min}$ )	$ts_{58}$		12	ns
$\overline{\text{CS}}$ high after read to data valid for write ( $ts_{5\_csx\_max} - ts_{4\_data\_fl\_e}$ )	$ts_{59}$		18	ns
$\overline{\text{RD}}$ high pulse width ( $ts_{8\_rd\_r\_max} - ts_{8\_rd\_f\_min}$ )	$ts_{60}$		15	ns
ADDR valid to $\overline{\text{WR}}$ going low ( $ts_{5\_addr\_max} - ts_{7\_wr\_f\_min}$ )	$ts_{63}$		20	ns
Read enable timing without RW delay ( $ts_{7\_rd\_f\_max} + ts_{9\_data}$ )	$ts_{65}$		40	ns
Write width ( $ts_{7\_wr\_f\_max} - ts_{20\_wr\_r\_min}$ )	$ts_{67}$			ns
Data from $\overline{\text{WR}} / \overline{\text{CS}}$ ( $ts_{18\_wr\_r\_max} - ts_{12\_data\_f}$ )	$ts_{68}$		5	ns

<sup>1)</sup> Normal write not supported.

**Notes:**

- $ts_{20}$ ,  $ts_{21}$ , and  $ts_{23}$  can be used to calculate the access time required for an external memory.
- $ts_{24}$ ,  $ts_{25}$ , and  $ts_{26}$  can be used to calculate the data set up time for external memory.
- $ts_{28}$  can be used to calculate the time that the external memory has to enable its output for a read.

IF **BUSCONx.CSRENx** = 1, pin  $\overline{\text{CSx}}$  acts as  $\overline{\text{RDCSx}}$ , else pin  $\overline{\text{CSx}}$  retains its normal function.

**CONFIDENTIAL**

**Normal Operation Values**

Muxed Bus is not supported in E-GOLDradio and is not included within these timings. Read data are latched with the same clock edge that triggers the address change and the rising  $\overline{RD}$  edge. Therefore, address changing before the end of  $\overline{RD}$  have no impact on read cycles.

**Note:** The X-Bus clock (XCLK) is an internal clock signal which is measured via pin CLKOUT (the address and data bus measurement is done with XCLK = 26 MHz).

The X-Bus clock is identical to the MCU clock.

When the MCU clock is derived directly from the shaper clock (refer to [Section 10.4.1 Clock Generation Unit \(on Page 653\)](#)), XCLK can be asymmetrical (asymmetrical duty cycle) which will degrade the performance of the address and data bus.

Only when the X-Bus clock is derived from the PLL can a symmetrical XCLK can be guaranteed.

**Note:** The normal write operation is not supported, use the early write operation.

#### 13.2.1.2.3.3 Timing Diagrams for EBU Parameter Characterization

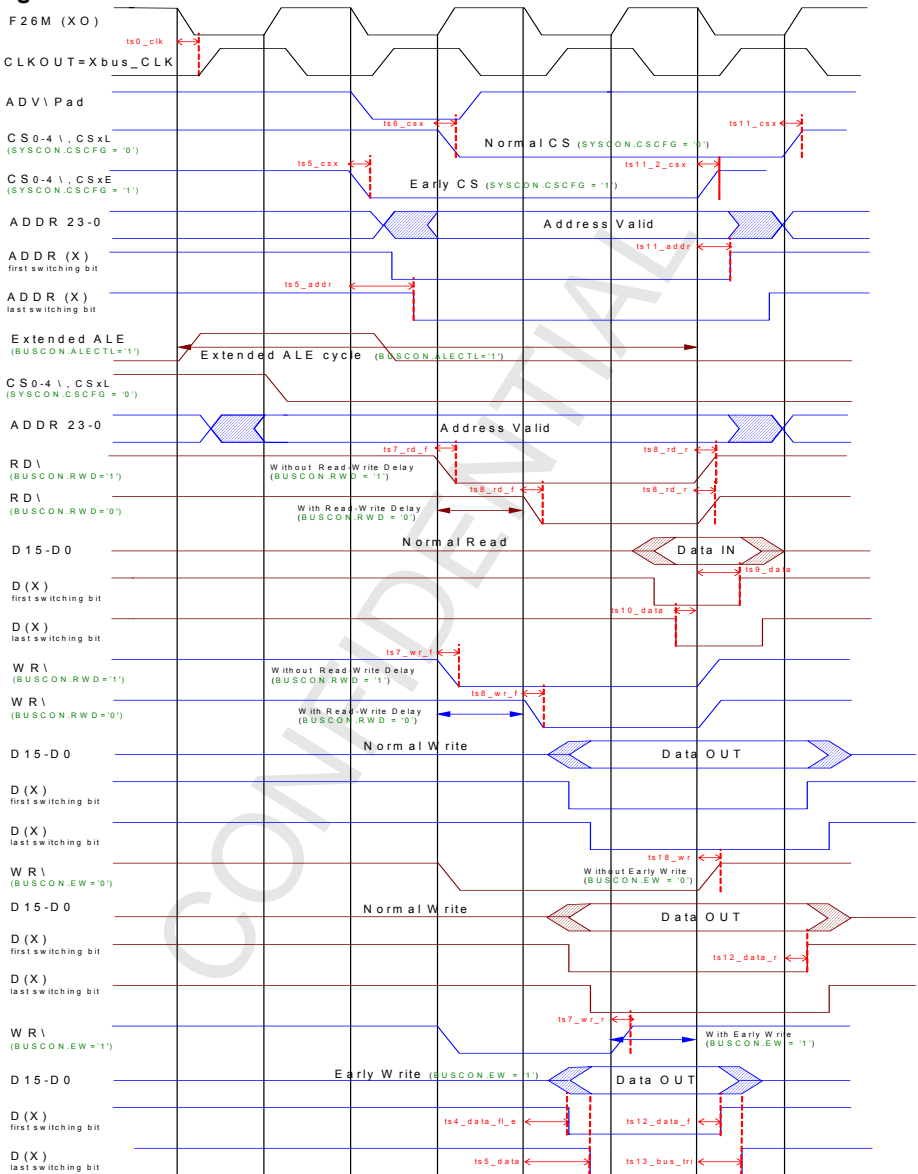
BUSCON in this section refers to [BUSCON0 \(on Page 870\)](#) and [BUSCON1](#).

SYSCON in this section refers to [SYSCON \(on Page 109\)](#).

[Figure 13-4](#) shows timing for the EBU parameters.



**Figure 13-4 Overview of EBU Parameter Characterizations**



## CONFIDENTIAL

## Normal Operation Values

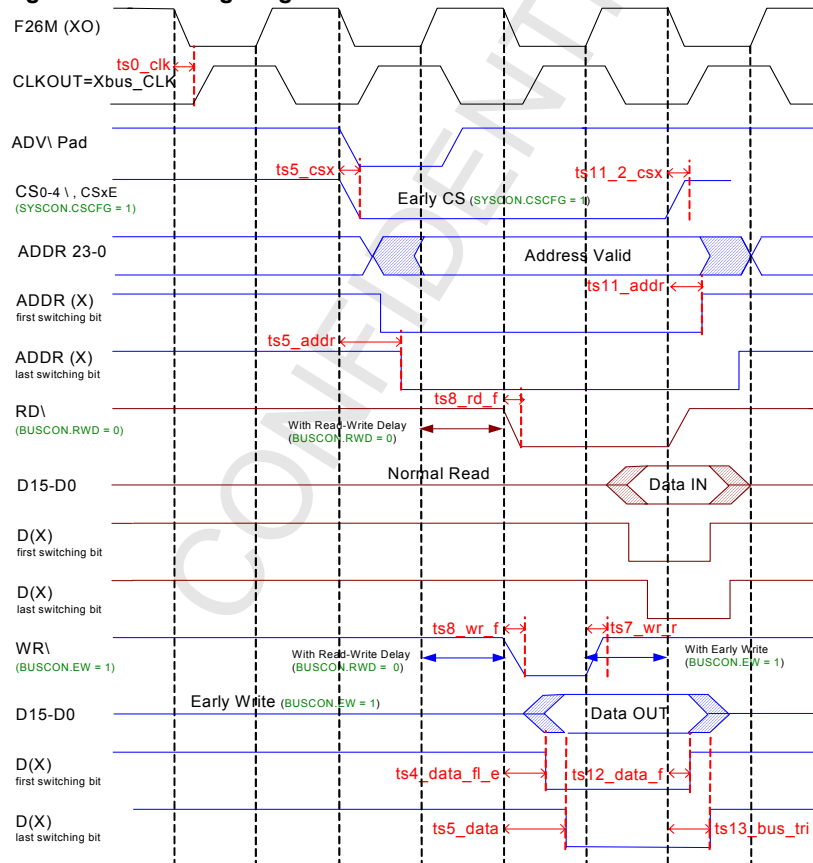
### Parameters

$t_{s0\_clk}$     $t_{s4\_data\_fl\_e}$     $t_{s5\_addr}$     $t_{s5\_csx}$     $t_{s5\_data}$     $t_{s7\_wr\_r}$   
 $t_{s8\_rd\_f}$     $t_{s8\_wr\_f}$     $t_{s11\_2\_csx}$     $t_{s11\_addr}$     $t_{s12\_data\_f}$     $t_{s13\_bus\_tri}$

are characterized under the following conditions (see [Figure 13-5](#)):

- **BUSCON** = 05AF<sub>H</sub>:
  - **BUSCON.RWD** = 0 => with Read Write Delay
  - **BUSCON.ALE** = 0 => without Extended ALE
  - **BUSCON.BSW** = 0 => without Tristate inserted when address window changes
- **SYSCON** = 7444<sub>H</sub>:
  - **SYSCON.CSCFG** = 1 => with Early Chip Select

**Figure 13-5 Timing Diagram of Parameters**



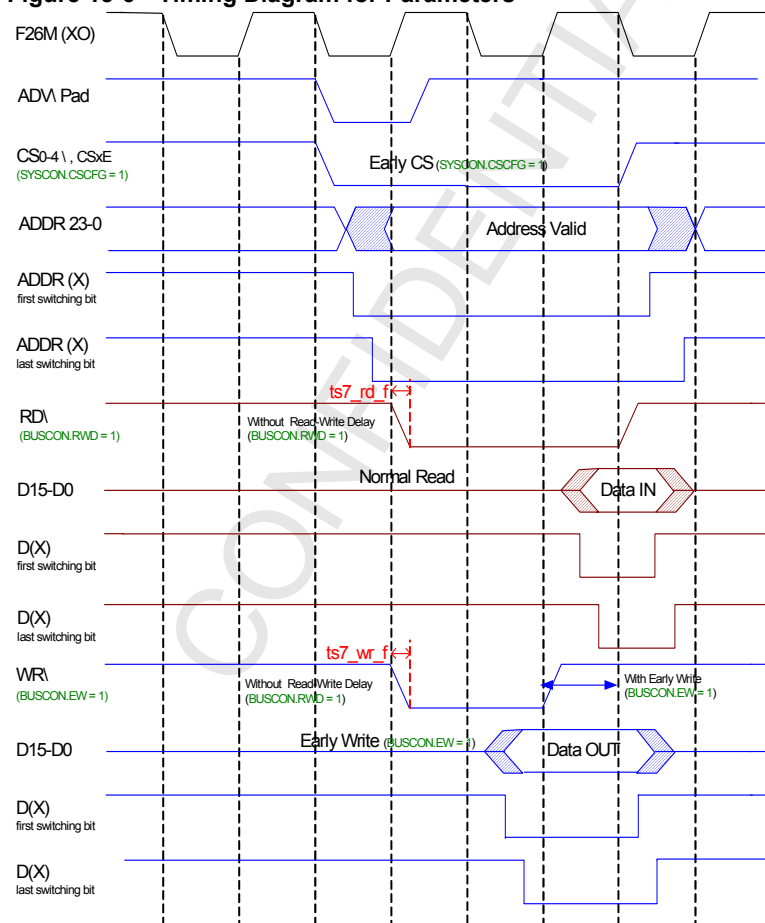
**CONFIDENTIAL**

**Normal Operation Values**

Parameters of  $ts7\_rd\_f$  and  $ts7\_wr\_f$  are characterized under the following conditions (see **Figure 13-6**):

- **BUSCON** = 05BF<sub>H</sub>:
  - **BUSCON.RWD** = 1 => without Read Write Delay
  - **BUSCON.EW** = 1 => with Early Write
  - **BUSCON.ALE** = 0 => without Extended ALE
  - **BUSCON.BSW** = 0 => without Tristate inserted when address window changes
- **SYSCON** = 7444<sub>H</sub>:
  - **SYSCON.CSCFG** = 1 => with Early Chip Select

**Figure 13-6 Timing Diagram for Parameters**



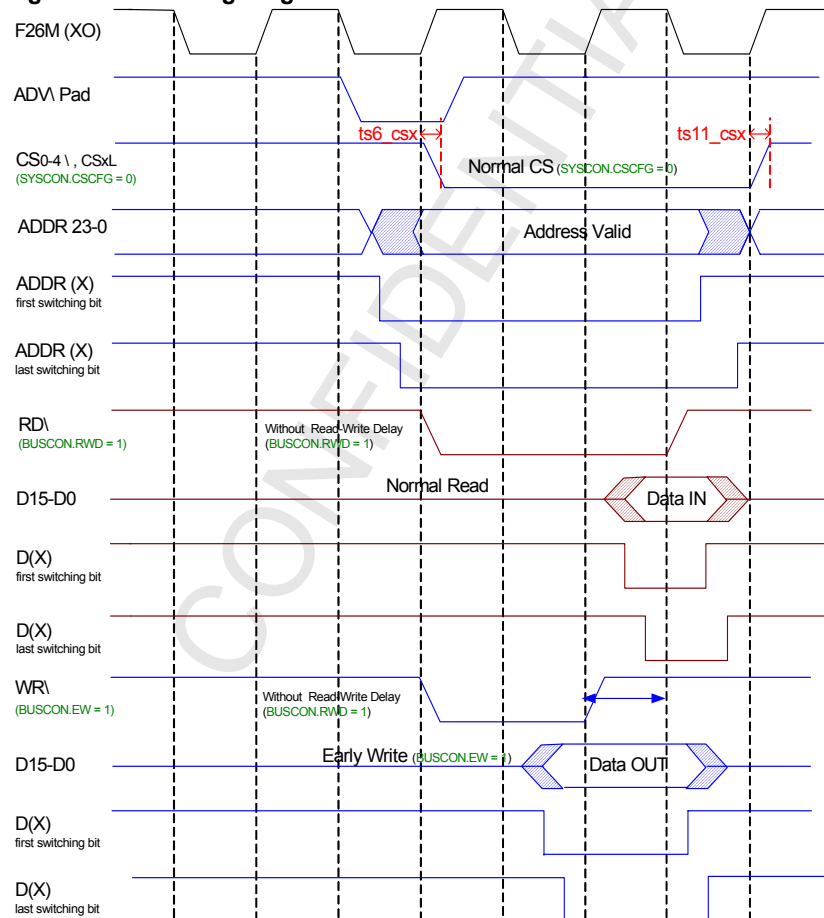
**CONFIDENTIAL**

**Normal Operation Values**

Parameters  $ts_{6\_csx}$  and  $ts_{11\_csx}$  are characterized under the following conditions (see **Figure 13-7**):

- **BUSCON** = 05AF<sub>H</sub>:
  - **BUSCON.RWD**= 0 => without Read Write Delay
  - **BUSCON.EW**= 1 => with Early Write
  - **BUSCON.ALE**= 0 => without Extended ALE
  - **BUSCON.BSW**= 0 => without Tristate inserted when address window changes
- **SYSCON** = 7404<sub>H</sub>:
  - **SYSCON.CSCFG**= 0 => without Early Chip Select

**Figure 13-7 Timing Diagram for Parameters**



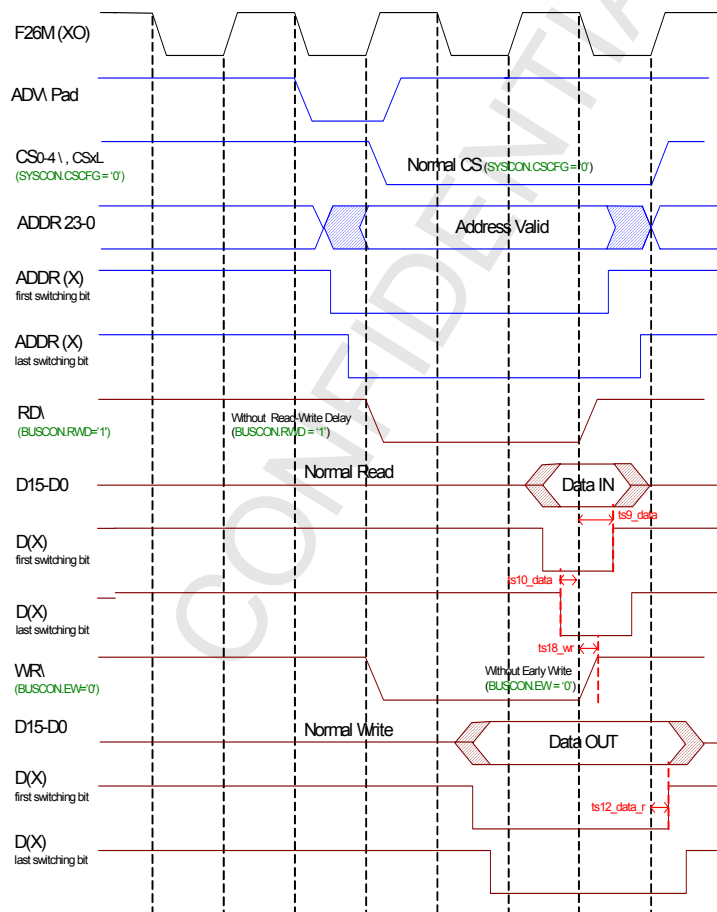
**CONFIDENTIAL**

**Normal Operation Values**

Parameters of  $ts_{12\_data\_r}$ ,  $ts_{18\_wr}$ ,  $ts_{9\_data}$ , and  $ts_{10\_data}$  are characterized under the following conditions (see [Figure 13-8](#)):

- **BUSCON** = 05AF<sub>H</sub>:
  - **BUSCON.RWD**= 0 => without Read Write Delay
  - **BUSCON.EW**= 0 => with Early Write
  - **BUSCON.ALE**= 0 => without Extended ALE
  - **BUSCON.BSW**= 0 => without Tristate inserted when address window changes
- **SYSCON** = 7404<sub>H</sub>:
  - **SYSCON.CSCFG**= 0 => without Early Chip Select

**Figure 13-8 Timing Diagram for Parameters**



#### 13.2.1.2.3.4 Timing Diagrams for PMCU Parameter Characterization

**Note:** In the following PMCU timing diagrams the F26M (XO) signal can be:

- 52 MHz with a 25% - 75% duty cycle
- 78 MHz with a 50% - 50% duty cycle.

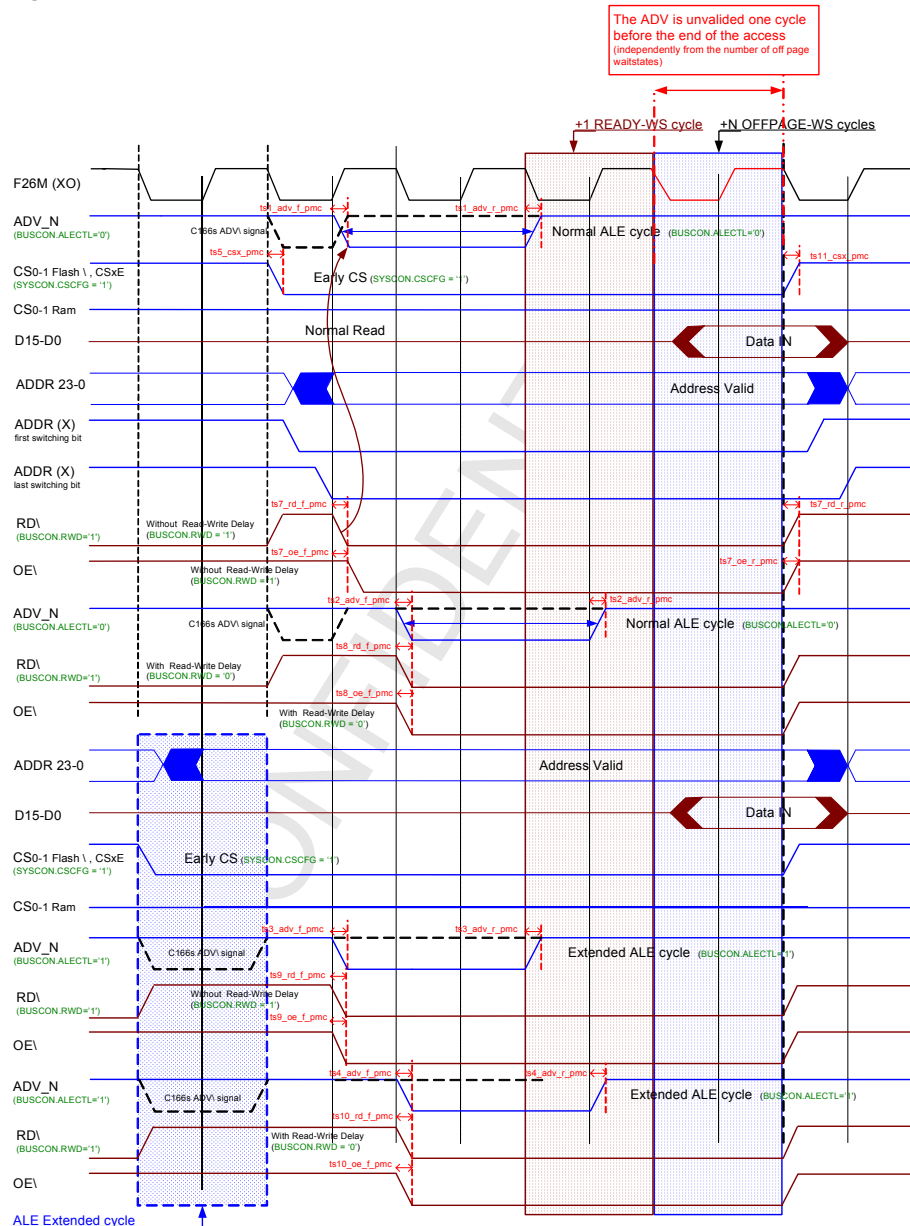
BUSCON in this section refers to **BUSCON0** and **BUSCON1**.

SYSCON in this section refers to **SYSCON (on Page 109)**.

**Figure 13-9** shows timing for the PMCU parameters.

CONFIDENTIAL

### Figure 13-9 Overview of PMCU Parameter Characterizations



**CONFIDENTIAL**

**Normal Operation Values**

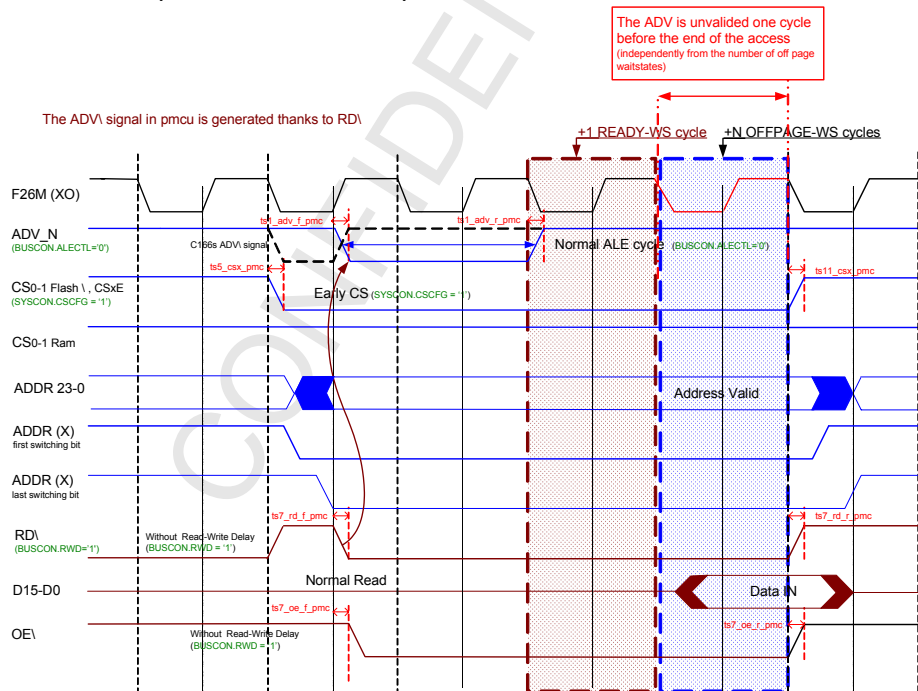
Parameters:

$ts1\_adv\_f\_pmc$        $ts1\_adv\_r\_pmc$        $ts5\_csx\_pmc$        $ts7\_oe\_f\_pmc$        $ts7\_rd\_f\_pmc$   
 $ts7\_oe\_r\_pmc$        $ts7\_rd\_r\_pmc$        $ts11\_csx\_pmc$

are characterized in **Figure 13-10** under the following conditions:

- **BUSCON** = 14BF<sub>H</sub>:
  - **BUSCON.RWD** = 1 => without Read Write Delay
  - **BUSCON.EW** = 0 => without Early Write
  - **BUSCON.ALE** = 0 => without Extended ALE
  - **BUSCON.BSW** = 0 => without Tristate inserted when address window changes
  - **BUSCON.RDYEN** = 1 => external bus cycles controlled by Ready signal generated by the PMCU
- **SYSCON** = 7444<sub>H</sub>:
  - **SYSCON.CSCFG** = 1 => with Early Chip Select

**Figure 13-10 First OFF-PAGE Access after Flash CS\ Validation 1**  
**(PMCx.OFFPWS > 00<sub>H</sub>)**





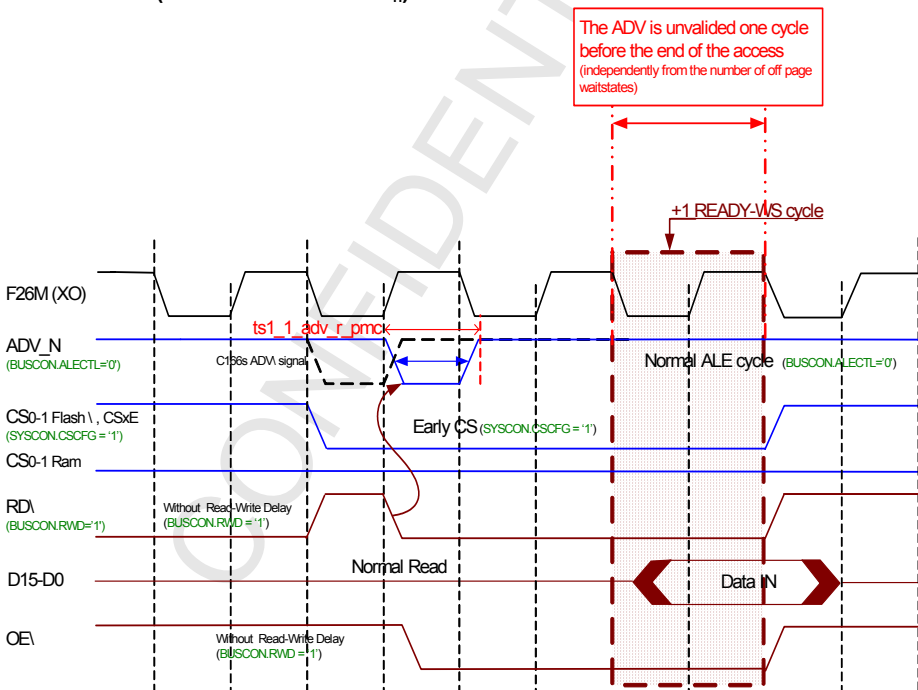
**CONFIDENTIAL**

**Normal Operation Values**

Parameter  $ts_{1\_1\_adv\_r\_pmc}$  is characterized in **Figure 13-11** under the following conditions:

- **BUSCON** = 14BF<sub>H</sub>:
  - **BUSCON.RWD** = 1 => without Read Write Delay
  - **BUSCON.EW** = 0 => without Early Write
  - **BUSCON.ALE** = 0 => without Extended ALE
  - **BUSCON.BSW** = 0 => without Tristate inserted when address window changes
  - **BUSCON.RDYEN** = 1 => external bus cycles controlled by Ready signal generated by the PMCU
- **SYSCON** = 7444<sub>H</sub>:
  - **SYSCON.CSCFG** = 1 => with Early Chip Select

**Figure 13-11 First OFF-PAGE Access after Flash CS\ Validation 2**  
(PMCx.OFFPWS = 00<sub>H</sub>)



**CONFIDENTIAL**

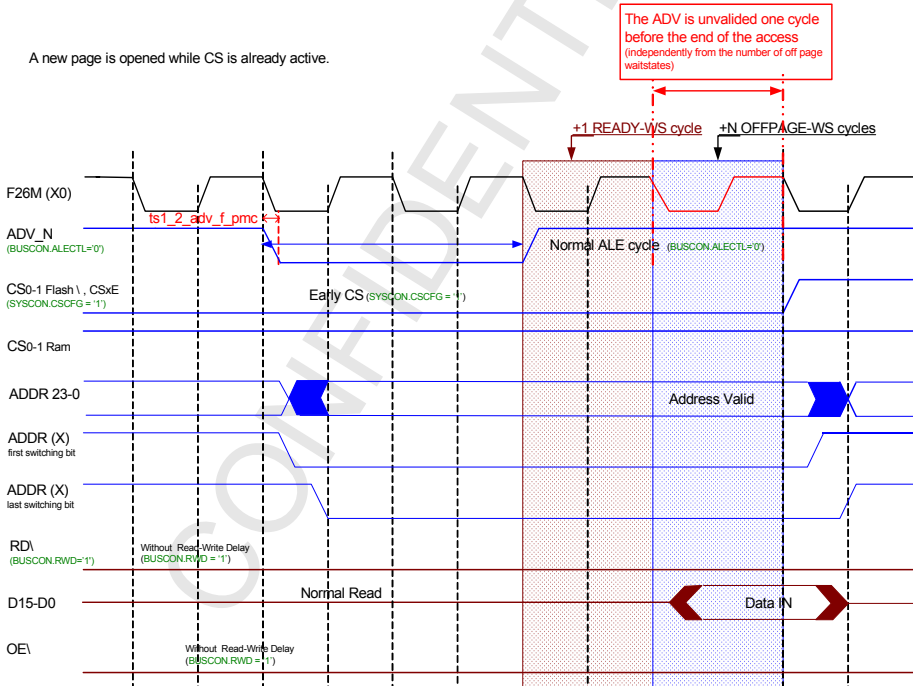
**Normal Operation Values**

Parameter  $ts_{1\_2\_adv\_f\_pmc}$  is characterized in **Figure 13-12** under the following conditions:

- **BUSCON** = 14BF<sub>H</sub>:
  - **BUSCON.RWD** = 1 => without Read Write Delay
  - **BUSCON.EW** = 0 => without Early Write
  - **BUSCON.ALE** = 0 => without Extended ALE
  - **BUSCON.BSW** = 0 => without Tristate inserted when address window changes
  - **BUSCON.RDYEN** = 1 => external bus cycles controlled by Ready signal generated by the PMCU
- **SYSCON** = 7444<sub>H</sub>:
  - **SYSCON.CSCFG** = 1 => with Early Chip Select

**Figure 13-12 OFF-PAGE Access Number N after Flash CS\ Validation (with N > 1)**

A new page is opened while CS is already active.



**CONFIDENTIAL**

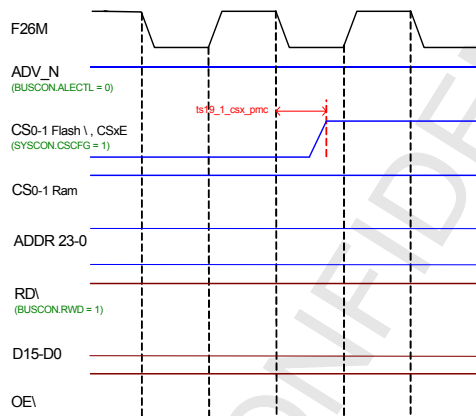
**Normal Operation Values**

Parameters  $ts_{19\_1\_csx\_pmc}$  is characterized in **Figure 13-13** under the following conditions:

- **BUSCON** =  $14BF_H$ :
  - **BUSCON.RWD** = 1 => without Read Write Delay
  - **BUSCON.EW** = 0 => without Early Write
  - **BUSCON.ALE** = 0 => without Extended ALE
  - **BUSCON.BSW** = 0 => without Tristate inserted when address window changes
  - **BUSCON.RDYEN** = 1 => external bus cycles controlled by Ready signal generated by the PMCU
- **SYSCON** =  $7444_H$ :
  - **SYSCON.CSCFG** = 1 => with Early Chip Select

**Figure 13-13 Switch of CS\ from Valid to Invalid**

(When Timeout of PMC\_TIMER is Reached)



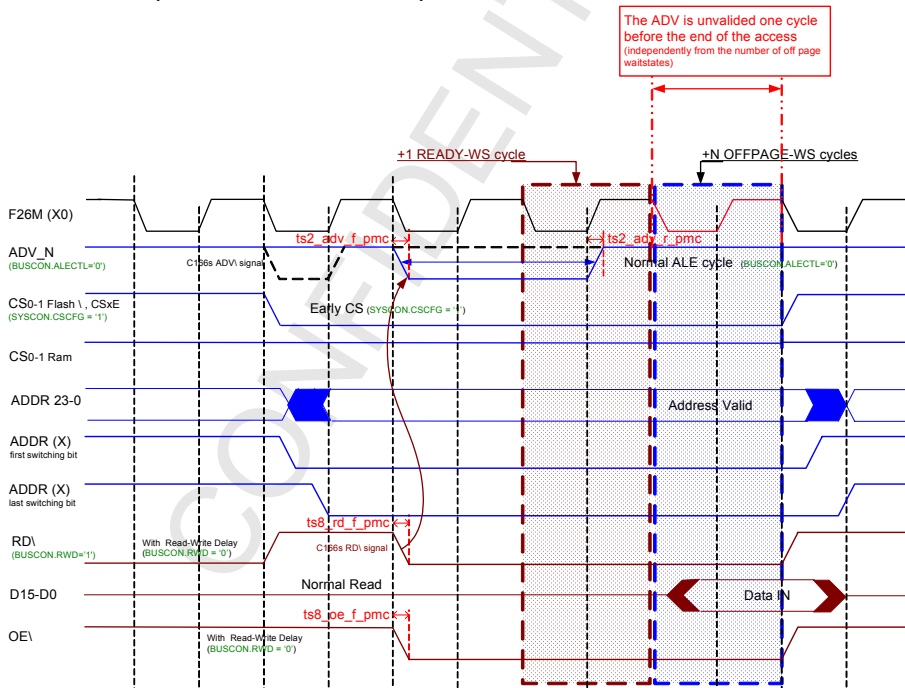
CONFIDENTIAL

**CONFIDENTIAL**

Parameters  $ts_{2\_adv\_f\_pmc}$   $ts_{2\_adv\_r\_pmc}$   $ts_{8\_oe\_f\_pmc}$  and  $ts_{8\_rd\_f\_pmc}$  are characterized in **Figure 13-14** under the following conditions:

- **BUSCON = 14AF<sub>H</sub>:**
  - **BUSCON.RWD** = 0 => with Read Write Delay
  - **BUSCON.EW** = 0 => without Early Write
  - **BUSCON.ALE** = 0 => without Extended ALE
  - **BUSCON.BSW** = 0 => without Tristate inserted when address window changes
  - **BUSCON.RDYEN** = 1 => external bus cycles controlled by Ready signal generated by the PMCU
- **SYSCON = 7444<sub>H</sub>:**
  - **SYSCON.CSCFG** = 1 => with Early Chip Select

**Figure 13-14 First OFF-PAGE Access after Flash CS\ Validation 1**  
**(PMC0/1.OFFPWS > 00<sub>H</sub>)**

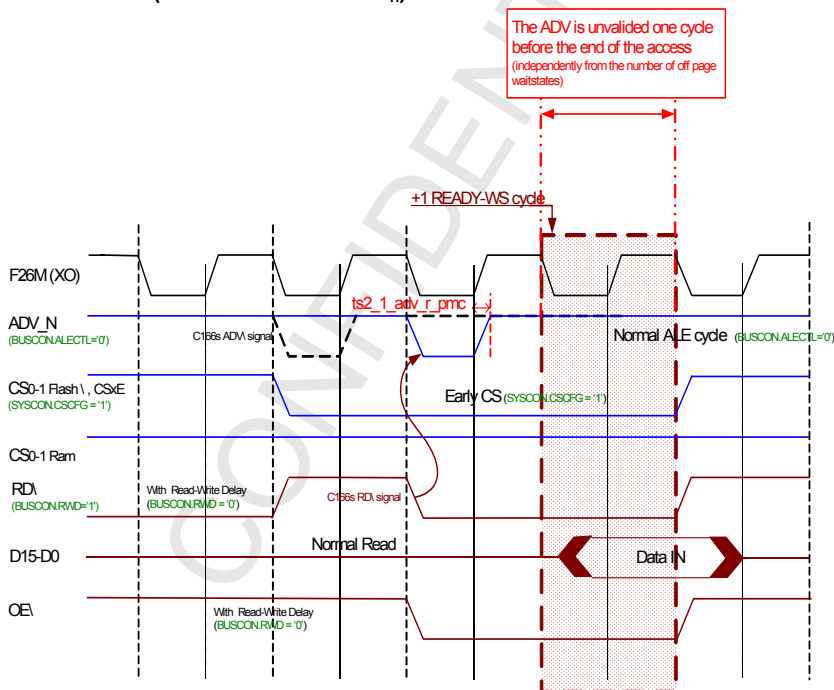


## CONFIDENTIAL

Parameter  $ts_{2\_1\_adv\_r\_pmc}$  is characterized in **Figure 13-15** under the following conditions:

- **BUSCON = 14AF<sub>H</sub>:**
  - **BUSCON.RWD** = 0 => with Read Write Delay
  - **BUSCON.EW** = 0 => without Early Write
  - **BUSCON.ALE** = 0 => without Extended ALE
  - **BUSCON.BSW** = 0 => without Tristate inserted when address window changes
  - **BUSCON.RDYEN** = 1 => external bus cycles controlled by Ready signal generated by the PMCU
- **SYSCON = 7444<sub>H</sub>:**
  - **SYSCON.CSCFG** = 1 => with Early Chip Select

**Figure 13-15 First OFF-PAGE Access after Flash CS\ Validation 2**  
(**PMCx.OFFPWS = 00<sub>H</sub>**)



## CONFIDENTIAL

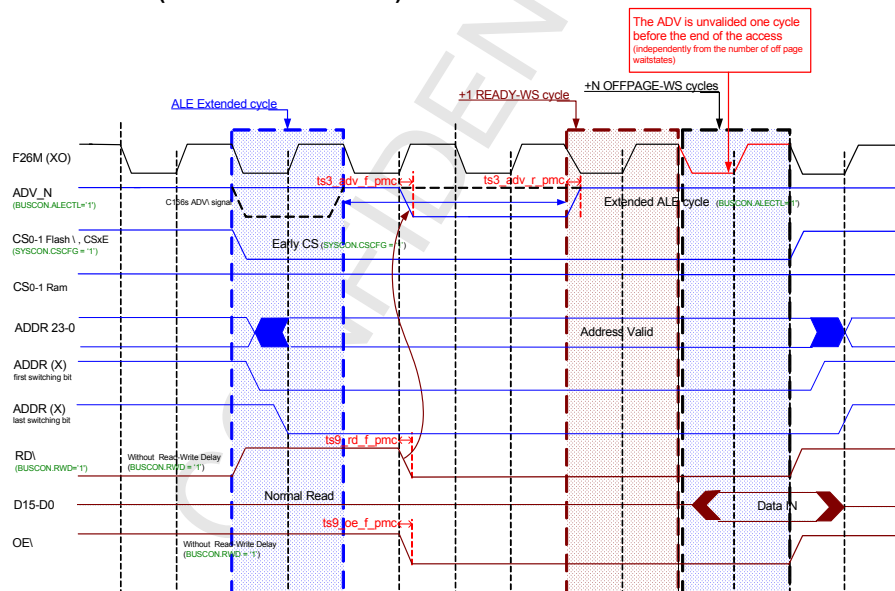
Parameters:

ts3\_adv\_f\_pmc ts3\_adv\_r\_pmc ts9\_oe\_f\_pmc ts9\_rd\_f\_pmc

are characterized in **Figure 13-16** under the following conditions:

- **BUSCON** = 16BF<sub>H</sub>:
  - **BUSCON.RWD** = 1 => with Read Write Delay
  - **BUSCON.EW** = 0 => without Early Write
  - **BUSCON.ALE** = 1 => with Extended ALE
  - **BUSCON.BSW** = 0 => without Tristate inserted when address window changes
  - **BUSCON.RDYEN** = 1 => external bus cycles controlled by Ready signal generated by the PMCU
- **SYSCON** = 7444<sub>H</sub>:
  - **SYSCON.CSCFG** = 1 => with Early Chip Select

**Figure 13-16 First OFF-PAGE Access after the Flash CS\ Validation 3**  
(PMCx.OFFPWS > 00)

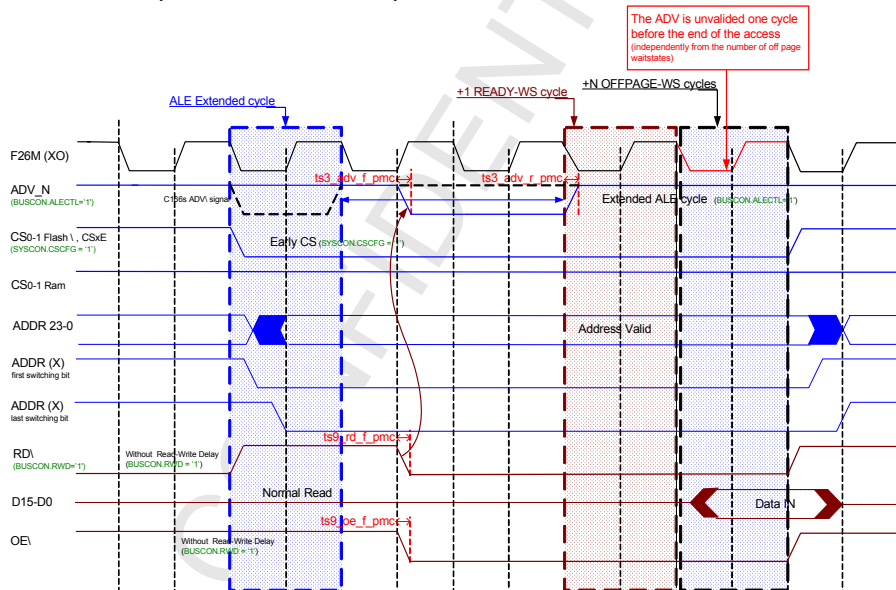


**CONFIDENTIAL**

Parameter  $ts_{3\_1\_adv\_r\_pmc}$  is characterized in **Figure 13-17** under the following conditions:

- **BUSCON** = 16BF<sub>H</sub>:
  - **BUSCON.RWD** = 1 => with Read Write Delay
  - **BUSCON.EW** = 0 => without Early Write
  - **BUSCON.ALE** = 1 => with Extended ALE
  - **BUSCON.BSW** = 0 => without Tristate inserted when address window changes
  - **BUSCON.RDYEN** = 1 => external bus cycles controlled by Ready signal generated by the PMCU
- **SYSCON** = 7444<sub>H</sub>:
  - **SYSCON.CSCFG** = 1 => with Early Chip Select

**Figure 13-17 First OFF-PAGE Access after the Flash CS\ Validation 4**  
(PMCx.OFFPWS = 00)





## CONFIDENTIAL

Parameters:

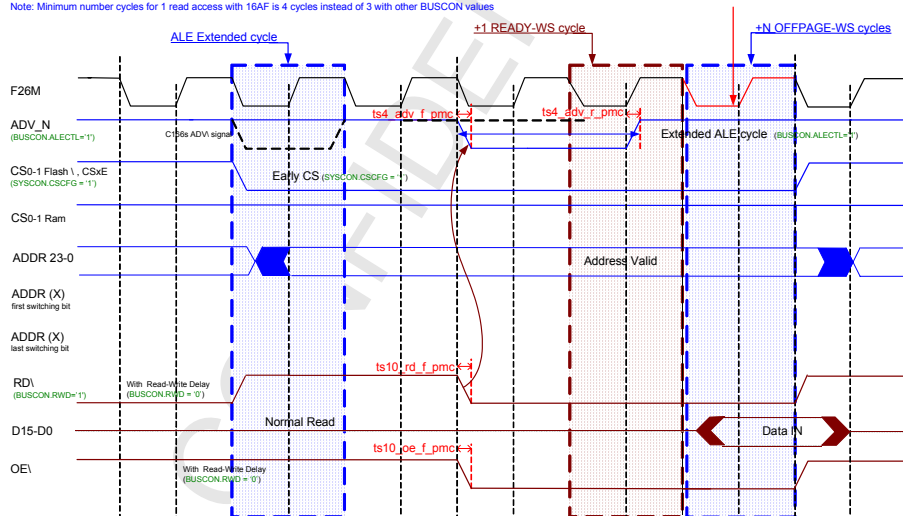
ts4\_adv\_f\_pmc ts4\_adv\_r\_pmc ts10\_oe\_f\_pmc ts10\_rd\_f\_pmc

are characterized in **Figure 13-18** under the following conditions:

- **BUSCON = 16AF<sub>H</sub>:**
  - **BUSCON.RWD** = 0 => without Read Write Delay
  - **BUSCON.EW** = 0 => without Early Write
  - **BUSCON.ALE** = 1 => *with Extended ALE*
  - **BUSCON.BSW** = 0 => without Tristate inserted when address window changes
  - **BUSCON.RDYEN** = 1 => *external bus cycles controlled by Ready signal generated by the PMCU*
- **SYSCON = 7444<sub>H</sub>:**
  - **SYSCON.CSCFG** = 1 => with Early Chip Select

**Figure 13-18 First OFF-PAGE Access after the Flash CS\ Validation 5**  
**(PMCx.OFFPWS > 00<sub>H</sub>)**

Note: Minimum number cycles for 1 read access with 16AF is 4 cycles instead of 3 with other BUSCON values



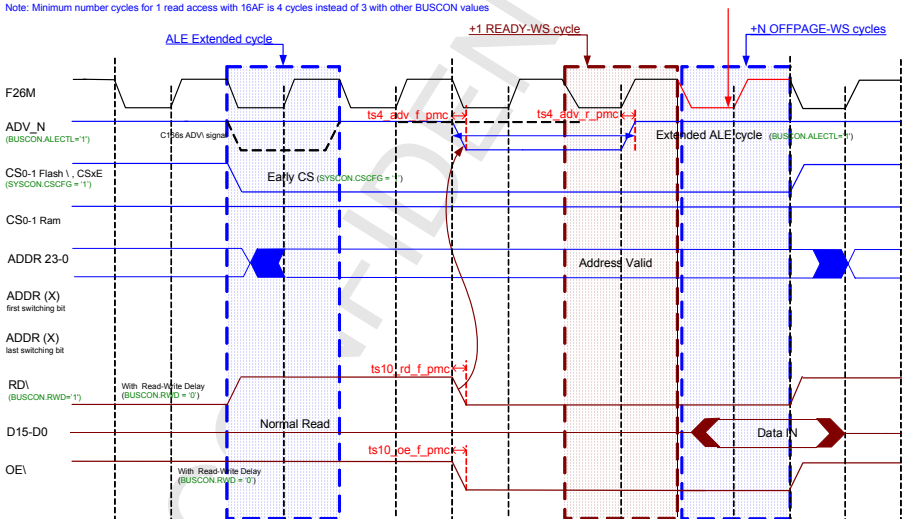
## CONFIDENTIAL

Parameter  $ts_{4\_1\_adv\_r\_pmc}$  is characterized in **Figure 13-19** under the following conditions:

- **BUSCON = 16AF<sub>H</sub>:**
  - **BUSCON.RWD** = 0 => without Read Write Delay
  - **BUSCON.EW** = 0 => without Early Write
  - **BUSCON.ALE** = 1 => *with Extended ALE*
  - **BUSCON.BSW** = 0 => without Tristate inserted when address window changes
  - **BUSCON.RDYEN** = 1 => *external bus cycles controlled by Ready signal generated by the PMCU*
- **SYSCON = 7444<sub>H</sub>:**
  - **SYSCON.CSCFG** = 1 => with Early Chip Select

**Figure 13-19 First OFF-PAGE Access after the Flash CS\ Validation 6**  
(**PMCx.OFFPWS = 00<sub>H</sub>**)

Note: Minimum number of cycles for 1 read access with 16AF is 4 cycles instead of 3 with other BUSCON values



**CONFIDENTIAL**

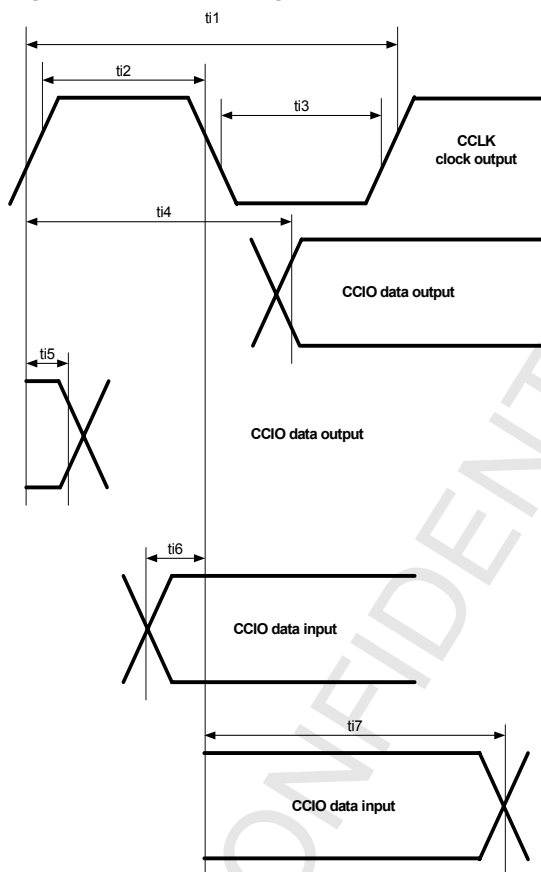
### 13.2.1.2.4 SIM

**Table 12-25 SIM Timing**

Parameter	Symbol	Limit Values			Unit
		Minimum	Typical	Maximum	
CCLK clock period (output)	$t_{i1}$		307		ns
CCLK high time	$t_{i2}$	110		184	ns
CCLK low time	$t_{i3}$	110		184	ns
CCIO (output) valid after CCLK low end	$t_{i4}$			200	ns
CCIO (output) still stable after CCLK low end	$t_{i5}$	30			ns
CCIO (input, 50%) setup before CCLK high end	$t_{i6}$	45			ns
CCIO (input, 50%) hold after CCLK low begin	$t_{i7}$	45			ns

CONFIDENTIAL

Figure 13-20 SIM Timing



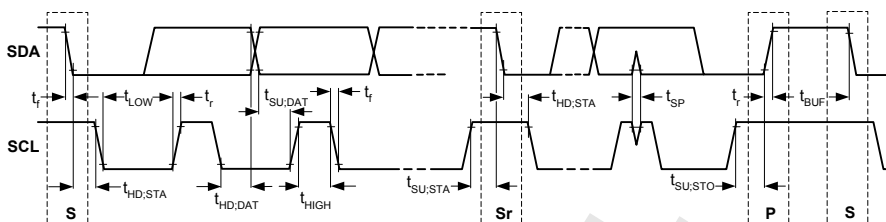
*Note: For information about changing the CCLK frequency, refer to [Section 10.4.1.2.4 Changing the MCU Sub-System Frequencies During Operation \(on Page 661\)](#).*

### 13.2.1.2.5 I2C

The I<sup>2</sup>C bus interface meets the requirements of the STANDARD-MODE and FAST-MODE as described in the I<sup>2</sup>C-bus specification Version 2.1 as published by Phillips Semiconductors available on the World-Wide-Web. The timing definition of [Figure 13-21](#) is equal to the timing definition in Fig. 31 of the Phillips I<sup>2</sup>C-bus specification.

CONFIDENTIAL

Figure 13-21 Definition of Timing for F/S-Mode Devices on I<sup>2</sup>C-Bus



Note: The timings given in [Table 12-26](#) apply only to the special I<sup>2</sup>C Open Drain pins (Function1 in pin list). They do not apply to pins, where the I<sup>2</sup>C functionality is used as alternative function (Function 1, 2,3,... in pin list).

Table 12-26 Timing Characteristic of I<sup>2</sup>C-Bus Interface

Parameter	Symbol	Standard-Mode <sup>1)</sup>		Fast-Mode <sup>2)</sup>		Unit
		Min.	Max.	Min.	Max.	
SCL clock frequency	$f_{SCL}$	0	100	0	400	kHz
Hold time (repeated) START condition. After this period, the first clock pulse is generated	$t_{HD;STA}$	4.0	-	0.6	-	$\mu s$
LOW period of the SCL clock	$t_{LOW}$	4.7	-	1.3 <sup>3)</sup>	-	$\mu s$
HIGH period of the SCL clock	$t_{HIGH}$	4.0	-	0.6	-	$\mu s$
Set-up time for a repeated START condition	$t_{SU;STA}$	4.7	-	0.6	-	$\mu s$
Data hold time	$t_{HD;DAT}$	0	3.45	0	0.9	$\mu s$
Data setup time	$t_{SU;DAT}$	250		100		$\mu s$
Rise time of both SDA and SCL signals	$t_r$	-	1000	27 <sup>4)</sup>	150 <sup>5)</sup>	ns
Fall time of both SDA and SCL signals	$t_f$	-	300	27	150	ns
Set-up time for STOP condition	$t_{SU;STO}$	4.0	-	0.6	-	$\mu s$

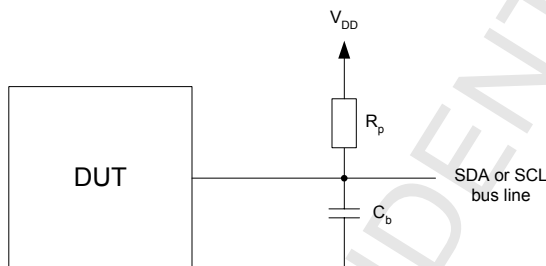
**CONFIDENTIAL**

**Table 12-26 Timing Characteristic of I<sup>2</sup>C-Bus Interface**

Bus free time between a STOP and START condition	$t_{BUF}$	4.7	-	1.3	-	$\mu s$
Capacitive load for each bus line	$C_b$		100		100	pF

- 1) Settings of register **IIC\_CON**: BRP = 81<sub>H</sub>, PREDIV = 01<sub>H</sub>, BRPMOD = 0
- 2) Settings of register **IIC\_CON**: BRP = 3F<sub>H</sub>, PREDIV = 08<sub>H</sub>, BRPMOD = 1
- 3) 1.3  $\mu s$  is the Phillips I2C-bus specification, but the E-GOLDradio value is 1  $\mu s$ .
- 4) Capacitive load  $C_h$  = 70 pF and pull-up resistor  $R_n$  = 1 kOhm
- 5) Capacitive load  $C_b$  = 100 pF and pull-up resistor  $R_p$  = 9 kOhm

**Figure 13-22 Test circuitry of I2C-bus**



### 13.2.1.2.6 I2S

This timing characterization is valid for both bi-directional and unidirectional I<sup>2</sup>S interfaces.

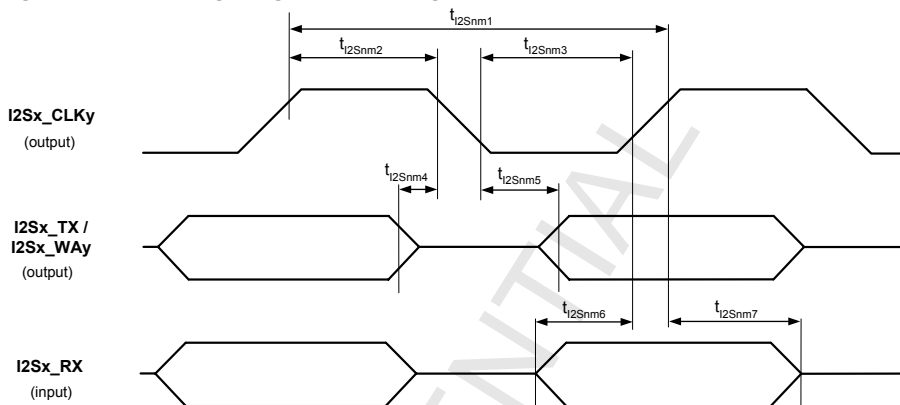
The description of pin alternatives are in [Chapter 3 Pin Descriptions](#).

CONFIDENTIAL

### 13.2.1.2.6.1 Normal Mode

#### Master Mode

Figure 13-23 Timing Diagram of I<sup>2</sup>S Signals in Normal Mode - Master Mode



**CONFIDENTIAL**

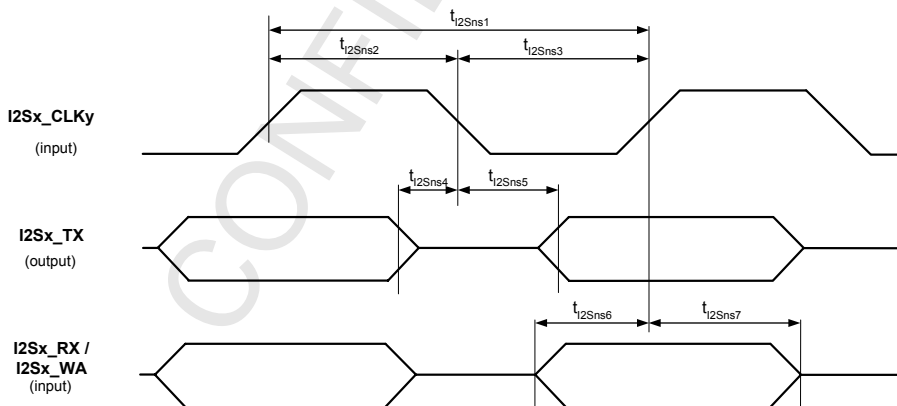
**Table 12-27 Timing Characteristic of I<sup>2</sup>S in Normal Mode - Master Mode**

Parameter	Symbol	Limit Values			Unit
		Minimum	Typical	Maximum	
I2Sx_CLKy clock period	$t_{I2Snm1}$	153 <sup>1)</sup>			ns
I2Sx_CLKy high time	$t_{I2Snm2}$	61			ns
I2S_CLKy low time	$t_{I2Snm3}$	61			ns
I2Sx_TX invalid before I2S_CLK0 high end	$t_{I2Snm4}$			24	ns
I2Sx_TX valid after I2S_CLK0 low begin	$t_{I2Snm5}$			24	ns
I2Sx_RX setup time before I2S_CLK1 low end	$t_{I2Snm6}$	27			ns
I2Sx_RX hold time after I2S_CLK1 high begin	$t_{I2Snm7}$	0			ns

<sup>1)</sup> Values are characterized for a maximum clock rate of 6.5 MHz

### Slave Mode

**Figure 13-24 Timing Diagram of I<sup>2</sup>S Signals in Normal Mode - Slave Mode**





**CONFIDENTIAL**

**Table 12-28 Timing Characteristic of I<sup>2</sup>S in normal mode - slave mode**

Parameter	Symbol	Limit Values			Unit
		Minimum	Typical	Maximum	
I2Sx_CLKy clock period	$t_{I2Sns1}$	306 <sup>1)</sup>			ns
I2Sx_CLKy high time	$t_{I2Sns2}$	n.a.			ns
I2S_CLKy low time	$t_{I2Sns3}$	n.a.			ns
I2Sx_TX invalid before I2S_CLKy falling edge	$t_{I2Sns4}$			0	ns
I2Sx_TX valid after I2S_CLKy falling edge	$t_{I2Sns5}$			80	ns
I2Sx_RX setup time before I2S_CLK1 rising edge	$t_{I2Sns6}$	24			ns
I2Sx_RX hold time after I2S_CLK1 rising edge	$t_{I2Sns7}$	24			ns

<sup>1)</sup> Values are characterized for a maximum clock rate of 3.25 MHz

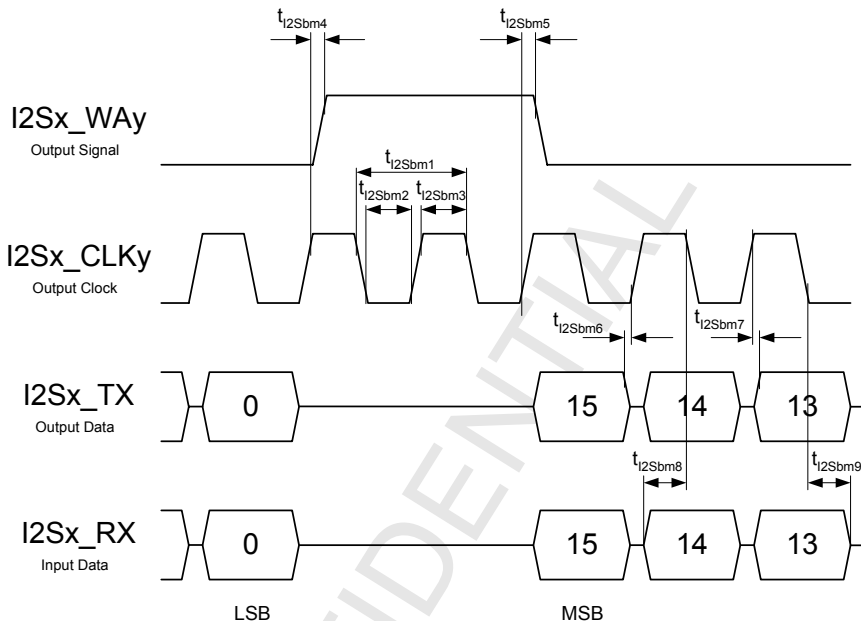
#### **13.2.1.2.6.2 Burst Mode**

The values in [Figure 13-25](#) and [Figure 13-26](#) are valid for the I<sup>2</sup>S interface in the burst mode.

CONFIDENTIAL

## Master Mode

Figure 13-25 Timing Diagram of I<sup>2</sup>S Signals in Burst Mode - Master Mode



**CONFIDENTIAL**

**Table 12-29 Timing Characteristics of I<sup>2</sup>S in Burst Mode - Master Mode**

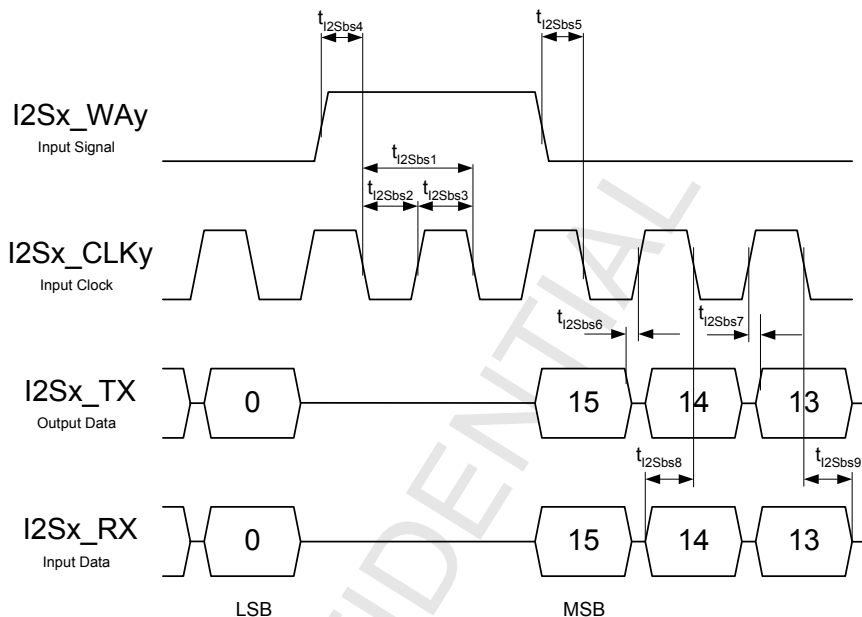
Parameter	Symbol	Limit Values			Unit
		Minimum	Typical	Maximum	
I2Sx_CLKy clock period	t <sub>I2Sbm1</sub>	153 <sup>1)</sup>			ns
I2Sx_CLKy low time	t <sub>I2Sbm2</sub>	61			ns
I2S_CLKy high time	t <sub>I2Sbm3</sub>	61			ns
I2Sx_CLKy low begin to I2Sx_WAy high begin	t <sub>I2Sbm4</sub>	-24		24	ns
I2Sx_WAy low begin to I2Sx_CLKy low begin	t <sub>I2Sbm5</sub>	-24		24	ns
I2Sx_TX invalid before I2S_CLK0 low end	t <sub>I2Sbm6</sub>			24	ns
I2Sx_TX valid after I2S_CLK0 high begin	t <sub>I2Sbm7</sub>			24	ns
I2Sx_RX setup time before I2S_CLK1 high end	t <sub>I2Sbm8</sub>	27			ns
I2Sx_RX hold time after I2S_CLK1 low begin	t <sub>I2Sbm9</sub>	0			ns

<sup>1)</sup> Values are characterized for a maximum clock rate of 6.5 MHz

CONFIDENTIAL

Slave Mode

Figure 13-26 Timing Diagram of I<sup>2</sup>S Interface Signals in Burst Mode - Slave Mode



**CONFIDENTIAL**

**Table 12-30 Timing Characteristics of I<sup>2</sup>S in Burst Mode - Slave Mode**

Parameter	Symbol	Limit Values			Unit
		Minimum	Typical	Maximum	
I2Sx_CLKy clock period	t <sub>I2Sbs1</sub>	306 <sup>1)</sup>			ns
I2Sx_CLKy low time	t <sub>I2Sbs2</sub>	n.a.			ns
I2S_CLKy high time	t <sub>I2Sbs3</sub>	n.a.			ns
I2Sx_CLKy low begin to I2Sx_WAy high begin	t <sub>I2Sbs4</sub>	36			ns
I2Sx_WAy low begin to I2Sx_CLKy low begin	t <sub>I2Sbs5</sub>	36			ns
I2Sx_TX invalid before I2S_CLKy rising edge	t <sub>I2Sbs6</sub>			0	ns
I2Sx_TX valid after I2S_CLKy rising edge	t <sub>I2Sbs7</sub>			80	ns
I2Sx_RX setup time before I2S_CLK1 falling edge	t <sub>I2Sbs8</sub>	24			ns
I2Sx_RX hold time after I2S_CLK1 falling edge	t <sub>I2Sbs9</sub>	24			ns

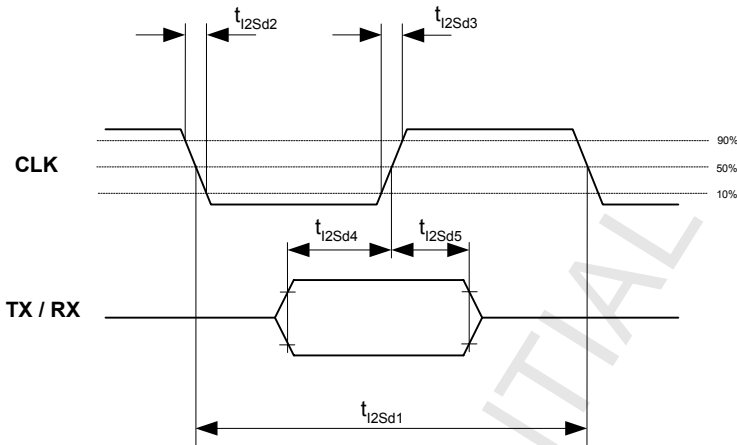
<sup>1)</sup> Values are characterized for a maximum clock rate of 3.25 MHz

### 13.2.1.2.6.3 DAI Mode

**Figure 13-27** and **Table 12-31** shows the timing requirements for a DAI interface according to the GSM standard. These requirements are met by design. Thus no electrical characterization is necessary.

**CONFIDENTIAL**

**Figure 13-27 Timing Diagram of I<sup>2</sup>S Interface Signals in DAI Mode**



**Table 12-31 Timing Characteristics of I<sup>2</sup>S in DAI Mode**

Parameter	Symbol	Limit Values			Unit
		Minimum	Typical	Maximum	
Clock period time <sup>1)</sup>	$t_{I2Sd1}$	9.61520	9.61538	9.61557	us
Clock falling edge time	$t_{I2Sd2}$			1	us
Clock rising edge time	$t_{I2Sd3}$			1	us
Data setup time before rising edge of CLK	$t_{I2Sd4}$	3			us
Data hold time after rising edge of CLK	$t_{I2Sd5}$	1			us

<sup>1)</sup> The clock frequency needs to be 104 kHz +/- 20 ppm

### 13.2.1.2.7 ASC

The following timing characteristic is valid for the ASC0 and the ASC1, they apply only in the synchronous mode.

*Note: In the asynchronous mode there is no timing given, since there is only one asynchronous transmission data line and one asynchronous reception data line. For the timing of the TX signal, the rise and fall times of the driver used must be considered.*

CONFIDENTIAL

Figure 13-28 Timing of ASC in Synchronous Mode

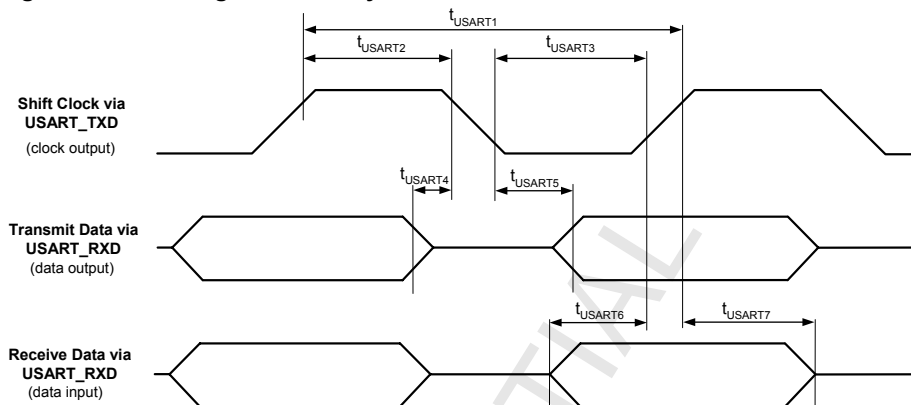


Table 12-32 Timing Characteristic of ASC with D Drivers in Synchronous Mode

Parameter	Symbol	Limit Values			Unit
		Minimum	Typical	Maximum	
Shift clock period	$t_{USART1}$	150 <sup>1)</sup>			ns
Shift high time	$t_{USART2}$	50			ns
Shift clock low time	$t_{USART3}$	50			ns
Transmit data invalid before shift clock high end	$t_{USART4}$			10	ns
Transmit data valid after shift clock low begin	$t_{USART5}$			10	ns
Receive data setup time before shift clock low end	$t_{USART6}$	50			ns
Receive data hold time after shift clock high begin	$t_{USART7}$	0			ns

<sup>1)</sup> Maximum characterized baud rate of ASC with D drivers in synchronous mode is 6.5 Mbps

**Note:** The USARTs in alternative functions with E drivers should only be used in asynchronous mode. Timings for synchronous modes are not guaranteed in this case.

## CONFIDENTIAL

### 13.2.1.2.8 SSC

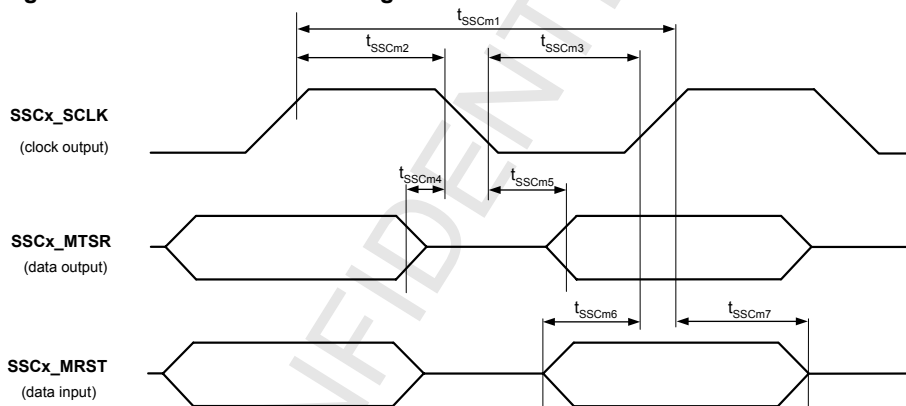
#### 13.2.1.2.8.1 Master Mode

In the master mode the maximum programmable baud rates of the MCU and DSP SSC modules are 26 MBaud and 44.55 MBaud, respectively. But the actual achievable baud rates are limited by the following timing characteristics.

*Note: The settings for this specification are:*

- SSC operates in master mode
- Idle clock line is high, the leading clock edge is high-to-low => **SSCPD\_CON.PO = 1**
- Shift transmit data on the leading edge, latch on trailing edge => **SSCPD\_CON.PH = 0.**

**Figure 13-29 SSC Interface Timing in Master Mode**





**CONFIDENTIAL**

**Table 12-33 Timing Characteristic of SSC in Master Mode**

Parameter	Symbol	Limit Values			Unit
		Minimum	Typical	Maximum	
SCLK clock period	$t_{SSCm1}$	38			ns
SCLK high time	$t_{SSCm2}$	n.a. <sup>1)</sup>			ns
SCLK low time	$t_{SSCm3}$	n.a.			ns
MTSR invalid before SCLK high end	$t_{SSCm4}$			5	ns
MTSR valid after SCLK low begin	$t_{SSCm5}$			7	ns
MRST setup time before SCLK low end for 6.5MHz internal clock	$t_{SSCm6}$ for MCU	165			ns
MRST setup time before SCLK low end for 13MHz internal clock		85			ns
MRST setup time before SCLK low end for 26MHz internal clock		45			ns
MRST setup time before SCLK low end for 52MHz internal clock		25			ns
MRST setup time before SCLK low end for 104MHz internal clock ( <i>clock_dsp</i> )	$t_{SSCm6}$ for DSP	15			ns
MRST hold time after SCLK high begin	$t_{SSCm7}$	0			ns

<sup>1)</sup> The duty cycle of SCLK is 50%. Thus the resulting high and low times are by design one half of the clock period minus the slope. The slope depends on the driver class of the used pad (see [Chapter 3 Pin Descriptions](#)).

#### 13.2.1.2.8.2 Slave Mode

In slave mode the maximum programmable baud rates of the MCU and DSP SSC modules are 13 MBaud and 22.25 MBaud, respectively. But the actual achievable baud rates are limited by the following timing characteristics.

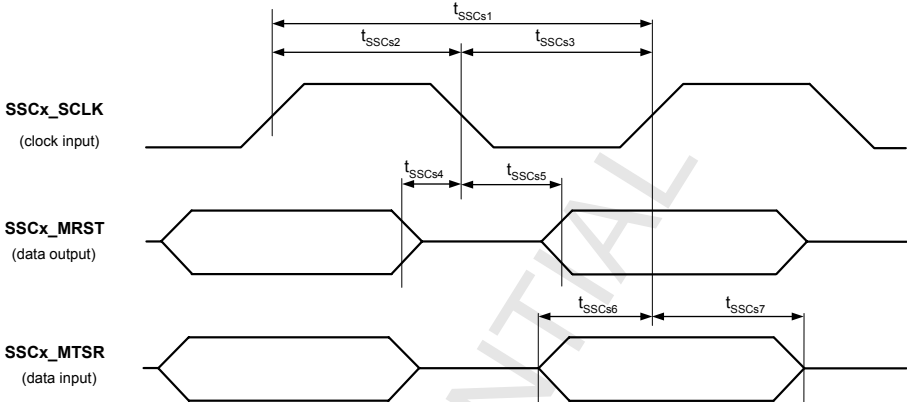
*Note: The settings for this specification are:*

- SSC operates in slave mode
- Idle clock line is high, the leading clock edge is high-to-low => **SSCPD\_CON.PO = 1**

**CONFIDENTIAL**

- Shift transmit data on the leading edge, latch on trailing edge  
=> **SSCPD\_CON.PH** = 0.

**Figure 13-30 SSC Interface Timing in Slave Mode**



**CONFIDENTIAL**

**Table 12-34 Timing Characteristic of SSC in Slave Mode**

Parameter	Symbol	Limit Values			Unit
		Minimum	Typical	Maximum	
SCLK clock period	$t_{SSCs1}$	76.9			ns
SCLK high time	$t_{SSCs2}$	n.a. <sup>1)</sup>			ns
SCLK low time	$t_{SSCs3}$	n.a.			ns
MRST invalid before SCLK falling edge	$t_{SSCs4}$			0	ns
MRST valid after SCLK falling edge (with C/D/E driver)	$t_{SSCs5}$			32/42/45	ns
MTSR setup time before SCLK low end for 6.5MHz internal clock	$t_{SSCs6}$ for MCU	165			ns
MTSR setup time before SCLK low end for 13MHz internal clock		85			ns
MTSR setup time before SCLK low end for 26MHz internal clock		45			ns
MTSR setup time before SCLK low end for 52MHz internal clock		25			ns
MTSR setup time before SCLK low end for 104MHz internal clock ( <i>clock_dsp</i> )	$t_{SSCs6}$ for DSP	15			ns
MTSR hold time after SCLK high begin	$t_{SSCs7}$	5			ns

<sup>1)</sup> The duty cycle should be approximately 50%. The minimum values of  $t_{SSCs2}$  and  $t_{SSCs3}$  are basically constrained by the values of  $t_{SSCs2...7}$ .

**CONFIDENTIAL**

### 13.2.1.2.9 RTC and 32 kHz Pad Oscillator

#### 13.2.1.2.9.1 Timing of RTC

**Table 12-35** gives the timing for the optional standby clock that is used with the PMB7870.

**Table 12-35 Standby Clock Timings**

Parameter	Symbol	Limit Values			Unit
		Minimum	Typical	Maximum	
Input clock period f32k	$f_{32k}$	18	32	55	kHz
Input clock high time f32k	$t_{H32k}$	8			μs
Input clock low time f32k	$t_{L32k}$	8			μs

The duty cycle must be between 45%:55% and 55%:45%

#### 13.2.1.2.9.2 External Circuitry and Recommended Crystal

For using the internal oscillator of the PMB7870 together with an external crystal the crystal parameters load capacitance  $C_L$ , static capacitance  $C_0$  and internal serial resistance  $R_s$ , the parameters  $C_1$  and  $C_2$  of **Figure 11-47 Circuitry to Use the Internal 32 kHz Oscillator (on Page 1129)** and the board capacitances must be related as follows.

The real part of the amplitude impedance (= negative resistive component of input impedance between F32k and OSC32K) of the 32k on chip oscillator is  $R_{int} = -240 \text{ kOhm}$  (that is,  $|R_{int}| \geq 240 \text{ kOhm}$ ). It is strongly recommended to use a crystal with a maximal serial resistance of:

$$R_{s\_max} < (-R_{int} / 3) = 80 \text{ kOhm} \quad [36.18]$$

Further it is recommended to choose the typical value of  $R_{s\_typ}$ :

$$R_{s\_typ} < (-R_{int}/5) \quad [36.19]$$

Values of  $R_{s\_max}$  above 80 kOhm have to be verified in the application and are not guaranteed.

The parameter  $C_L$  must have the following range:

$$9\text{pF} < C_L < 12.5\text{pF} \quad [36.20]$$

The external load capacities must be kept in the range of:

$$17\text{pF} < C_1 + C_{1b} < 20.5 \text{ pF} \quad [36.21]$$

$$17\text{pF} < C_2 + C_{2b} < 20.5 \text{ pF} \quad [36.22]$$

## CONFIDENTIAL

The relation between these values is given by:

$$C_{1t} * C_{2t} / (C_{1t} + C_{2t}) + C_{i0} = C \quad [36.23]$$

Where:

- $C_{1t} = C_1 + C_{1b} + C_{1p}$ ;  $1\text{pF} < C_{1p} < 2.5\text{ pF}$ 
  - $C_{1p}$  = capacitance from pin OSC32k to ground added by the package and internal circuitry;
  - $C_{1b}$  = capacitance from pin OSC32K to ground added by the board-design
- $C_{2t} = C_2 + C_{2b} + C_{2p}$ ;  $1\text{pF} < C_{2p} < 2.5\text{ pF}$ 
  - $C_{2b}$  = capacitance from pin F32K to ground added by the package and internal circuitry;
  - $C_{2p}$  = capacitance from pin F32K to ground added by the board-design
- $C_{1t} = C_{2t}$
- $C_{i0} = C_0 + C_{xb} + C_{xp}$ ;  $0,05\text{ pF} < C_{xp} < 0,5\text{ pF}$ 
  - $C_{xp}$  = capacitance parallel to the crystal added by the package and internal circuitry;
  - $C_{xb}$  = capacitance parallel to the crystal added by the board-design

*Note: The value  $C_{i0}$  (and thus  $C_{xb}$ ) should be as small as possible to reduce the startup time. That is, the external crystal should be connected as close as possible to the PMB7870.*

*Note: A 32 kHz jitter problem has been demonstrated mainly related to the PCB layout. **To avoid this 32kHz jitter, the external crystal must be protected by shielding.***

### 13.2.1.2.10 Pad Tristate Control Timing

*Note: The values given below are safe values guaranteed by design insuring proper function of the circuit.*

**Table 12-36 Pad Tristate Control Timing**

Parameter	Symbol	Limit Values		Unit
		Min.	Max.	
PM_INT High to Outputs in Tristate (RESET_n = L)	$t_{pr1}$	0	200	ns
PM_INT Low to Outputs Enabled (RESET_n = L)	$t_{pr2}$	0	200	ns
RESET_n Low to Outputs in Reset State (PM_INT = L)	$t_{pr3}$	0	200	ns
RESET_n Low to Outputs in Tristate (PM_INT = H)	$t_{pr4}$	0	200	ns

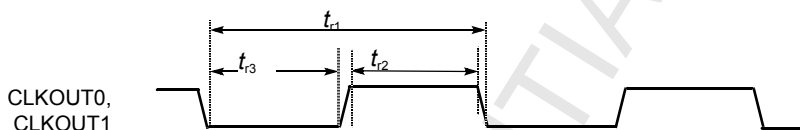
**CONFIDENTIAL**

### 13.2.1.2.11 Output Clock CLKOUT

**Table 12-37 Output Clock CLKOUT**

Parameter	Symbol	Limit Values			Unit
		Minimum	Typical	Maximum	
CLKOUT period at 13 MHz	$t_{r1}$		77		ns
CLKOUT high time at 13 MHz	$t_{r2}$	26			ns
CLKOUT low time at 13 MHz	$t_{r3}$	26			ns

**Figure 13-31 Output Clocks: CLKOUT Timing**



## 13.2.2 Mixed Signals and Other Values

### 13.2.2.1 Audio

#### 13.2.2.1.1 Audio Receive Part

*Note: If not specified otherwise, all parameters are measured with a bandwidth of 20Hz,...,20kHz and gain setting  $g_s = 0dB$ .*

**Table 12-38 General Electrical Characteristics of Audio Receive Path**

Parameter	Symbol	Limit values			Unit	Remark
		Minimum	Typical	Maximum		
Maximal differential output voltage		3.3	3.7	4.1	Vpp	Full scale differential open circuit voltage
Maximal single-ended output voltage		1.65	1.85	2.05	Vpp	Full scale single-ended open circuit voltage
Current consumption of audio receive path				12	mA	without any external load
Output differential DC offset				+/- 50	mV	

**CONFIDENTIAL**
**Table 12-38 General Electrical Characteristics of Audio Receive Path**

Parameter	Symbol	Limit values			Unit	Remark
		Minimum	Typical	Maximum		
Output load resistance		14			$\Omega$	
Signal to noise	S/N	70	80		dB	for $R_L = 16 \Omega$ gs = 0dB; input signal 0dBFS, code 0, A-weighted
Idle channel noise			-90	-80	dBFS	code 0, A-weighted
Signal to distortion	THD	60	70		dB	for $R_L = 16 \Omega$ with input signal 0dBFS at $V_{dd} = 2.5V \dots 2.75V$
Signal to distortion	THD	60	70		dB	for $R_L = 16 \Omega$ with input signal -1dBFS at $V_{dd} = 2.25V \dots 2.5V$
Signal to distortion	THD	60			dB	for $R_L = 16 \Omega$ with input signal -6dBFS at $V_{dd} = 2.25V \dots 2.5V$
Signal to distortion	THD		60		dB	for $R_L = 16 \Omega$ with input signal 0dBFS at $V_{dd} = 2.25V \dots 2.5V$
Power supply rejection - EPPA1/EPPA2	PSR	60	66		dB	$U_{VDDV}(t) = 2.5V + 0.15V \sin(2\pi \cdot 1 \text{ kHz} \cdot t)$ and $g_S = 0 \text{ dB}$
Power supply rejection - EPP1/EPN1	PSR	50	56		dB	$U_{VDDV}(t) = 2.5V + 0.15V \sin(2\pi \cdot 1 \text{ kHz} \cdot t)$ and $g_S = 0 \text{ dB}$
Cross talk (between receive and transmit channel)				-65	dB	$U_{TX}(t) = 1.075V + 0.15V \sin(2\pi \cdot 1 \text{ kHz} \cdot t)$ $U_{RX}(t) = 0.15V \sin(2\pi \cdot 33 \text{ kHz} \cdot t)$

**CONFIDENTIAL**

**Table 12-38 General Electrical Characteristics of Audio Receive Path**

Parameter	Symbol	Limit values			Unit	Remark
		Minimum	Typical	Maximum		
Passband ripple				0.5	dB	$f < 0.45f_s$
Stopband attenuation		50			dB	$f > 0.55f_s$
Analog post-filter cut-off frequency			800		kHz	
Absolute gain drift				+/- 2	%	Variation due to change in VDD, temperature and life time

**Table 12-39 Characteristics of Audio Rx Path for Power Buffer EP<sub>pa1</sub> and EP<sub>pa2</sub>**

Parameter	Symbol	Limit values			Unit	Remark
		Minimum	Typical	Maximum		
Maximal differential output voltage		3.3	3.7	4.1	V <sub>pp</sub>	Full scale differential open circuit voltage
Maximal single-ended output voltage		1.65	1.85	2.05	V <sub>p</sub>	Full scale single-ended open circuit voltage
Maximal output current				200	mA	
Internal output resistance			1.7	4	Ω	
Single-ended output load capacitance				10	nF	
Single-ended output load capacitance					μF	Between output pins and GND with Ω series resistance



**CONFIDENTIAL**

**Table 12-40 Characteristics of Audio Rx Path for Differential Line-Out Drivers  
EP<sub>p1</sub> and EP<sub>n1</sub>**

Parameter	Symbol	Limit values			Unit	Remark
		Minimum	Typical	Maximum		
Maximal differential output voltage		3.3	3.7	4.1	V <sub>pp</sub>	Full scale differential open circuit voltage
Differential output voltage			925		mV <sub>pp</sub>	at load resistance R <sub>L</sub> = 16 Ω and -12dBFS
Maximal output current				64	mA	
Internal output resistance			4		Ω	
Single-ended output load capacitance				250	pF	Between output pins and GND
Single-ended output load capacitance					μF	Between output pins and GND with Ω series resistance

**Table 12-41 Electrical characteristics of Ringer Support**

Parameter	Symbol	Limit Values			Unit	Test Condition/ Remark
		Minimum	Typical	Maximum		
Differential output voltage in <b>RING</b> Mode	V <sub>r</sub>		2 V <sub>dd</sub>		V	

CONFIDENTIAL

Figure 13-32 Timing Diagram for Analog Ringer Mode

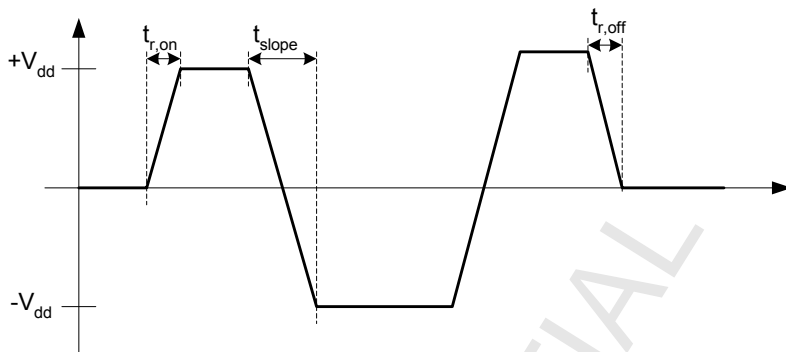


Table 12-42 Timing Parameters of Ringer Output

Parameter	Symbol	Limit Values		
		Minimum	Typical	Maximum
Rise time for first slope		11.25		$\mu s$
Rise and fall times for slopes during ringer operation		22.5		$\mu s$
Fall time for last slope (transition to <b>RINGHOLD</b> mode)		11.25		$\mu s$

*Note: Rise and fall times of the ringer output signal depend on the actual supply voltage, whereas the slew rates are constant. For minimal supply voltage the typical rise and fall times are some 10% smaller than for nominal supply voltage. For maximal supply voltage the typical rise and fall times are some 10% greater than in the nominal case. Values given here are preliminary. Final values will be available as soon as measurements will be available.*

**CONFIDENTIAL**

### 13.2.2.1.2 Audio Transmit Path

**Table 12-43 Electrical Characteristics of Audio Transmit Path**

Parameter	Symbol	Limit Values			Unit	Remark
		Minimum	Typical	Maximum		
Differential input voltage				1.03	V <sub>pp</sub>	
Differential input resistance			50		kΩ	
Input capacitance			5	10	pF	
Offset					mV	
S/D		65			dB	
Signal-to-noise ratio	S/N	75			dB	gs = +12dB, bandwidth 300-3900Hz (GSM mode)
Signal-to-noise ratio	S/N	72			dB	gs = +12dB, bandwidth 300Hz-7.0kHz (WAMR mode, )
Power supply rejection		66	85		dB	gs = 24 dB
		62			dB	gs = 18 dB
		45			dB	gs = 0 dB,
						$U_{VDDV}(t) = 2.5 V + 0.15V \sin(2\pi \cdot 1 \text{ kHz} \cdot t)$
Cross talk (between receive and transmit channel)				- 65	dB	$U_{TX}(t) = 1.075 V + U_{RX}(t) = 0.775 V \sin(2\pi \cdot 1 \text{ kHz} \cdot t)$
cut-off frequency of anti-alias filter		16			kHz	
Gain of ADC					%/V	Definition see general remark
LSB step size					μV	
Absolute gain drift				+/- 2	%	Variation due to change in VDD, temperature and life time

**CONFIDENTIAL**

### 13.2.2.1.3 Microphone Supply

**Table 12-44 Electrical Characteristics of Microphone Supply**

Parameter	Symbol	Limit Values			Unit	Remark
		Minimum	Typical	Maximum		
Output voltage of pin VMICH			1.8 2.0 2.2		V	VDD (typ.) = 2.25 V ... 2.75 V VDD (typ.) = 2.4 V ... 2.75 V VDD (typ.) = 2.5 V ... 2.75 V
Output voltage of pin VMICL			0		V	
Microphone supply current				2.0	mA	
Load capacitance		1		2	nF	
Load resistance		1			kΩ	
Power supply rejection of microphone supply			75		dB	<sup>1)</sup>

<sup>1)</sup>  $UVDDbg(t) = 2.6\text{ V} + 0.10\text{ V} \sin(2\pi \cdot 1\text{ kHz} \cdot t)$  and  $gs = 0\text{ dB}$  in crosstalk free conditions at board level

*Note: EN\_L2N must be disabled before entering the standby powerdown mode because the PLL is powered down and, thus, de-asserting the LOCKED signal.*

**CONFIDENTIAL**

### 13.2.2.2 GMSK Modulator

**Table 12-45 Baseband Transmit Path (Signal Outputs I/IX, Q/QX) for TXON = 1**

Parameter	Symbol	Limit Values			Unit	Test Condition/ Remark
		Minimum	Typical	Maximum		
Differential load resistance		20.0			kΩ	
Output load capacitance				20.0	pF	single ended
Differential output voltage	I ,  Q	0.95	1.00	1.05	V <sub>pp</sub>	maximum digital input values
Differential offset (not calibrated)	DC_I, DC_Q	-25.0		25.0	mV	
Differential offset after calibration	DC_I, DC_Q	-3.0	0	3.0	mV	Includes VDD and meas. accuracy <sup>1)</sup>
Differential RMS offset (not calibrated) <sup>2)</sup>	DC <sub>RMS</sub>			36.0 <sup>3)</sup>	mV	$[(DC\_I)^2 + (DC\_Q)^2]^{1/2}$
I&Q matching of amplitudes				1.2	%	(I-Q)/(I+Q)
GMSK-Ampl. variation of IQ vector <sup>4)</sup>	AM		1.3	2.0	%	(IQmin - IQmax) / (IQmin + IQmax)
GMSK phase error of IQ vector <sup>3)</sup>	ph_err <sub>RMS</sub>			1.0	°	
GMSK phase error of IQ vector <sup>3)</sup>	ph_err <sub>MAX</sub>			3.0	°	
Power of spurious spectra					dB	See spectrum mask
Accuracy of internal reference voltage for BB-TX		Nominal value - 140	Nominal value	Nominal value +140	mV	Nominal value programmed via T <sub>REF</sub> bits

<sup>1)</sup> Approximately 100 μV/K additional offset can be expected over temperature. For this reason customer should take care about the temperature change and periodically compensate in dependence of temperature change.

<sup>2)</sup> 0000<sub>H</sub> loaded into DAC

<sup>3)</sup> This variation is due to change of V<sub>DD</sub>, temperature and life time (nominal condition: 25 °C).

<sup>4)</sup> Sending pseudo-random GMSK data with pre-compensated offset.

**CONFIDENTIAL**

**Table 12-46 Baseband Transmit Path (Signal Outputs I/IX, Q/QX) for TXON=0**

Parameter	Symbol	Limit Values			Unit	Test Condition/ Remark
		Minimum	Typical	Maximum		
Differential output resistance		96.0	120.0	144.0	kΩ	TXON=0
Leakage current of ESD structure				10.0	uA	TXON=0

CONFIDENTIAL

**CONFIDENTIAL**

### 13.2.2.3 Baseband Receive Unit

**Table 12-47 Baseband Receiver Path (Signal Inputs IR/IRX, QR/QRX)**

Parameter	Symbol	Limit Values			Unit	Test Condition/ Remark
		Minimum	Typical	Maximum		
Characteristics						
Input resistance (Single ended)		90 5	150 <sup>1)</sup> 10 <sup>2)</sup>		kΩ	RXON = 1
Input capacitance (Single ended)			5	10	pF	
Differential offset introduced by ADC		-70		70	mV	
Signal-to-noise + distortion (standard mode)	S/(N+D)		66 <sup>3)</sup>		dB	1.0 Vpp desired (@ 20 kHz) (0... 100kHz)
Signal-to-noise + distortion (standard mode)	S/(N+D)		46 <sup>3)</sup>		dB	0.1 Vpp desired (@ 20 kHz) 1 Vpp interferer (@ 200 kHz)
Signal-to-noise + distortion (enhanced mode)	S/(N+D)		19.5 <sup>4)</sup>		dB	1.0 mVpp desired (@ 20 kHz) (0...100 kHz)
Signal-to-noise + distortion (enhanced mode)	S/(N+D)		33.5		dB	0.1 Vpp desired (@ 20kHz) 1.8 Vpp interferer (400kHz) (0...100 kHz)
3 dB corner frequency of anti- aliasing filter		200.0 <sup>1)</sup> 3000 <sup>2)</sup>	400 6000	600.0 9000	kHz	
Matching of I&Q		-0.25		0.25	dB	20*log(I/Q)
Overall Gain Definition			37095 <sup>5)</sup> 25969 <sup>6)</sup>		ΔLSB / ΔV	CORDIC ON = 0

**CONFIDENTIAL**

**Table 12-47 Baseband Receiver Path (Signal Inputs IR/IRX, QR/QRX)**

Parameter	Symbol	Limit Values			Unit	Test Condition/ Remark
		Minimum	Typical	Maximum		
Gain drift				±3	%	7)
Value of LSB			27 <sup>8)</sup> 39 <sup>9)</sup>		μV	CORDIC ON = 0
<b>Conditions</b>						
Max differential input voltage		1.90	2.0		V <sub>pp</sub>	Diff. offset by ADC must not be subtracted
Input voltage range		0		V <sub>DD</sub>	V	for each pin IR, IRX, QR, QRX
Common mode input voltage range		0.9		1.5	V	

1) Standard mode

2) Enhanced mode. Due to the small input resistance a non-zero external resistance influences also the gain of the BBRX input buffer and therefore the SNR of the baseband receive

3) Corresponds to a noise and distortion floor of 71.6 dB below 1.9 V<sub>pp</sub>

4) Corresponds to a noise and distortion floor of 85 dB below 1.9 V<sub>pp</sub>

5) Compromise channelization filter with BRFilterScaling = 3 for standard mode enhanced mode

6) Adaptive channelization filter with BRFilterScaling = 3 for standard mode and enhanced mode

7) Variation due to change in V<sub>DD</sub>, temperature and life time.

8) With compromise filter

9) With adaptive filter



**CONFIDENTIAL**

### 13.2.2.4 Measurement Interface

**Table 12-48 ADC Characteristics**

Parameter	Symbol	Limit Values			Unit	Remark
		Minimum	Typical	Maximum		
Resolution			12		Bits	
Differential linearity error	DNL			±0.5	LSB	
Integral linearity error	INL			±4	LSB	
Offset error	$U_{dig,0}$			±10	LSB	ADC input $U_2 = 0\text{ V}$
Gain	$g_{ADC}$	1966	2048	2130	LSB/V	<sup>1)</sup>
Absolute gain drift				±2	%	<sup>2)</sup>
Conversion time			60		cycle	$clk\_meas^3)$ cycles
Throughput rate				$f_M \cdot S/480$	Hz	$S = K/L^4)$
Acquisition time			32		cycle	$clk\_kernel\_2^4)$ cycles
Acquisition delay			39		cycle	$clk\_kernel\_2^5)$ cycles
Acquisition jitter		0		1	cycle	$clk\_kernel\_2^6)$ cycles
Wake-up time from power save			50		µs	<sup>7)</sup>

<sup>1)</sup> Variation due to process tolerances and change in VDDm, temperature and life time.

<sup>2)</sup> Variation due to change in VDDm, temperature and life time.

<sup>3)</sup>  $f_M = clk\_bus$  (52 MHz) and  $K = 02_H$ ,  $L = 0D_H$  (typical). Refer to [Section 10.5.12 Clock Control of Measurement Interface \(on Page 748\)](#).

<sup>4)</sup> During acquisition time the track and hold circuit is settling and tracking the signal (aperture). At the end of the acquisition period the track and hold circuit changes from track to hold and settles to its final value.

<sup>5)</sup> The acquisition delay is defined as the period in time from the occurrence of the rising edge of ADCTRIG (or setting **MEAS\_CTRL1.START**) to the beginning of the acquisition period. Acquisition is finished after the acquisition time is over. Especially, for PWPA measurement care has to be taken that the timing advance of ADCTRIG matches with the desired end of the acquisition period.

<sup>6)</sup> The acquisition jitter is defined as the uncertainty in time where the acquisition period begins (ends) in comparison to the ideal point in time defined by the signal ADCTRIG.

**CONFIDENTIAL**

- 7) Before ADC operation, the reference voltage has to be turned on (refer to [Section 10.6 Analog Control Registers](#)).

Wake-up time of reference voltage (band-gap) has to be considered separately.

**Table 12-49 Specification of pins M0 to M2 and M7 to M10<sup>1)</sup>**

Parameter	Symbol	Limit Values			Unit	Remarks
		Minimum	Typical	Maximum		
Characteristics						
Input resistance		1			MΩ	2)3)
Input leakage current				0.1	μA	
Equiv. network cross resistance	R <sub>x</sub>	1440	2400	3360	kΩ	Modes MxMyA <sup>4)</sup>
		960	1600	2240		Modes MxMyB <sup>4)</sup>
Common mode input resistance	R <sub>cm</sub>	360	600	840	kΩ	Modes MxMyA <sup>4)</sup>
		480	800	1120		Modes MxMyB <sup>4)</sup>
Internal common mode voltage	U <sub>cmi</sub>	0.58	0.60	0.62	V	Modes MxMyA <sup>2)4)</sup>
		0.58	0.60	0.62		Modes MxMyB <sup>2)4)</sup>
Input resistance in measurement mode	R <sub>eq</sub>	288	480	672	kΩ	Modes MxA <sup>2)4)</sup>
		320	533	747		Modes MxB <sup>2)4)</sup>
		70	117	163		Mode M10 <sup>2)4)</sup>
Internal voltage	U <sub>eq</sub>	0.46	0.48	0.50	V	Modes MxA <sup>2)4)</sup>
		0.38	0.40	0.42		Modes MxB <sup>2)4)</sup>
		0.48	0.50	0.52		Mode M10 <sup>2)4)</sup>
Pre-amplifier gain	G		0.5			Modes MxA, MxMyA
			1.0			Modes MxB, MxMyB
			0.4			Mode M10
ADC gain	g <sub>ADC</sub>	1966	2048	2130	LSB / V	4)
Absolute gain drift				±2	%	5)
Current tolerance	IR, IS			±4	%	TC[2:0] > 0 <sub>H</sub> <sup>4)</sup>
Absolute current drift				±2	%	TC[2:0] > 0 <sub>H</sub> <sup>5)</sup>

**CONFIDENTIAL**

**Table 12-49 Specification of pins M0 to M2 and M7 to M10<sup>1)</sup>** (cont'd)

Current ratio error				$\pm 1$	%	TC[2:0] > 0 <sub>H</sub> <sup>4)6)</sup>
Pre-amplifier and multiplexer settling time	CSEL = LOW <sup>7)</sup>			18.8	$\mu\text{s}$	Modes MxA, MxMyA
				37.5		Modes MxB, MxMyB
				3.8		Mode M10
	CSEL = HIGH <sup>7)</sup>			8.4		Modes MxA, MxMyA
				16.7		Modes MxB, MxMyB
				1.7		Mode M10
	-			1.0		Mode ADCCAL <sup>7)</sup>
Conditions						
Input voltage		0		VDDm	V	Modes MxMyA/B, ADCCAL <sup>2)8)</sup>
		0		1.92		Modes MxA <sup>8)9)</sup>
		0		0.96		Modes MxB <sup>8)9)</sup>
		0		2.40		Mode M10 <sup>8)9)</sup>
		0		VDDm - 0.6 V		Also for all modes if TC[2:0] > 0 <sup>8)</sup>
Differential input voltage		-1.92		1.92	V	Also for modes MxMyA <sup>10)</sup>
		-0.96		0.96		Also for modes MxMyB <sup>10)</sup>
		-0.96		0.96		Also for mode ADCCAL <sup>10)</sup>
Common mode input voltage		0	0.96	1.40	V	Also for modes MxMyA <sup>11)</sup>
		0	0.48	1.40		Also for modes MxMyB <sup>11)</sup>
		0.5		1.8		Also for mode ADCCAL <sup>11)</sup>

1) For measurement modes MxA, MxB and M10 as well as MxMyA, MxMyB and ADCCAL.

2) With respect to AGND.

3) If mode OFF is selected.

4) Variation due to process tolerances and change in VDD, temperature, and life time.

5) Variation due to change in VDDm, temperature, and life time.

6) Error of ratio ITC1/ITC2 with TC[2:0]1 not equal to TC[2:0]2.

**CONFIDENTIAL**

- 7) MX[5:0] has to be set in advance to the beginning of the acquisition time in order to guarantee that the external circuitry, the pre-amplifier, and the multiplexer are settled when the acquisition period begins. It is recommended to set CSEL = LOW to ensure 12 bit resolution of the ADC.
- 8) UMx/y for all measurement modes.
- 9) Pin may be left unconnected but measurement result is not specified.
- 10) UMx - UMy
- 11) (UMx + UMy)/2

**Table 12-50 Specification of BB\_I, BB\_Ix and BB\_Q, BB\_Qx**

Parameter	Symbol	Limit Values			Unit	Remark
		Minimum	Typical	Maximum		
Characteristics						
Input resistance		1			MΩ	1)
Overall gain		7864	8192	8520	LSB/ V	2)3)4)
Absolute gain drift				±2	%	2)5)
LSB step size			122		μV	
Pre-amplifier and multiplexer settling time				37.5	μs	MEAS_CTRL1. CSEL = 0 <sup>6)</sup>
				16.7		CSEL = 1 <sup>6)</sup>
Conditions						
Input voltage		0		VDDm	V	
Differential input voltage		-0.24		0.24	V	3)
Common mode input voltage		0.3		1.4	V	7)

- 1) For BB\_I <-> BB\_Ix and BB\_Q <-> BB\_Qx, if mode OFF (MEAS\_CTRL1.MX = 0<sub>H</sub>) is selected.
- 2) including amplifier and ADC without external load
- 3) Measurement:  
- Mode TXOFI:  $U_{BB\_Ix} - U_{BB\_I}$   
- Mode TXOFQ:  $U_{RR\_Qx} - U_{RR\_Q}$
- 4) Variation due to process tolerances and change in VDDm, temperature and life time.
- 5) Variation due to change in VDDm, temperature and life time.
- 6) MEAS\_CTRL1.MX has to be set before the start of the acquisition time to guarantee that the external circuitry, the pre-amplifier, and the multiplexer are settled when the acquisition period begins.

**CONFIDENTIAL**

- 7) Measurement:  
 - Mode TXOFI:  $(U_{BB\_I} + U_{BB\_IX})/2$   
 - Mode TXOFQ:  $(U_{BB\_Q} + U_{BB\_QX})/2$

**Table 12-51 Specification of PAOUTOF1**

Parameter	Symbol	Limit Values			Unit	Remark
		Minimum	Typical	Maximum		
Characteristics						
Input resistance		1			MΩ	1)2)
Overall gain		1966	2048	2130	LSB/ V	U <sub>PAOUTx</sub> <sup>3)4)</sup>
Absolute gain drift				±2	%	3)5)
LSB step size			488		mV	
Pre-amplifier and multiplexer settling time				37.5	μs	MEAS_CTRL1.CSEL = 0 <sup>6)</sup>
				16.7		CSEL = 1 <sup>6)</sup>
Conditions						
Input voltage		0		0.95	V	2)

1) If mode OFF (**MEAS\_CTRL1.MX** = 0<sub>H</sub>) is selected.

2) With respect to AGND

3) including amplifier and ADC, without external load

4) Variation due to process tolerances and change in VDDm, temperature and life time.

5) Variation due to change in VDDm, temperature and life time.

6) **MEAS\_CTRL1.MX** has to be set before the start of the acquisition time to guarantee that the external circuitry, the pre-amplifier, and the multiplexer are settled when the acquisition period begins.

**Table 12-52 On Chip Temperature Measurement TIC**

Parameter	Symbol	Limit Values			Unit	Remark
		Minimum	Typical	Maximum		
Temperature range		-30		125	°C	
Relative accuracy			±1	±2	°C	1)

**CONFIDENTIAL**

**Table 12-52 On Chip Temperature Measurement TIC (cont'd)**

Absolute accuracy				±5	°C	1)
Multiplexer settling time				1	µs	2)

1) guaranteed for at least to two years after calibration according [Section 10.5.6 On-Chip Temperature Measurement \(on Page 728\)](#).

2) **MEAS\_CTRL1.MX** has to be set before the start of the acquisition time to guarantee that the external circuitry, the pre-amplifier, and the multiplexer are settled when the acquisition period begins.

CONFIDENTIAL

**CONFIDENTIAL**

### 13.2.2.5 Analog Control Registers

**Table 12-53 Reference Voltage Generation (Band-Gap)**

Parameter	Symbol	Limit Values			Unit	Test Condition/ Remark
		Minimum	Typical	Maximum		
Characteristics						
Reference voltage	VREF		1.1		V	
Settling time of VREF after BG_PWUP is set HIGH				150	μs	99% of final voltage
Reference current (through external resistor on pin IREF)	IREF		50		μA μA	R <sub>IREF</sub> = 22 kΩ, max. load 5 pF, BG_PWUP = 1
			0			BG_PWUP = 0
Conditions						
External resistor on pin IREF	R <sub>IREF</sub>	21.5	22	22.4	kΩ	required external device
External capacitor on pin VREF	C <sub>VREF</sub>	47	220	250	nF	required external device
			47			required external device if internal voice band is not used.

### 13.2.2.6 RF Power Ramping

**Table 12-54 Specification of Pin PAOUT1 (TXON = 1)**

Parameter	Symbol	Limit Values			Unit	Test Condition/ Remark
		Minimum	Typical	Maximum		
Characteristics						
Gain		1.18	1.25	1.32	mV/ LSB	1)
Gain drift				±1	%	2)
Offset		-20 -70	0 -50	40 -10	mV	PA_OFF = 0 <sup>1)</sup> PA_OFF = 1
Offset drift				±5	mV	2)

**CONFIDENTIAL**

**Table 12-54 Specification of Pin PAOUT1 (TXON = 1)**

Parameter	Symbol	Limit Values			Unit	Test Condition/ Remark
		Minimum	Typical	Maximum		
Output resistance				10	$\Omega$	Within linear operating range
Saturation voltage	$U_{sat+}$	-25 -50 -270			mV	$I_{PAOUT} = +0.05 \text{ mA}$ $I_{PAOUT} = +0.50 \text{ mA}$ $I_{PAOUT} = +5.00 \text{ mA}$ $U_{sat+}$ is referred to $VDDbb^{(3)}$
	$U_{sat-}$			25	mV	$U_{sat-}$ is referred to AGND <sup>(3)</sup>
Integral nonlinearity	INL			$\pm 10$	LSB	
Differential nonlinearity	DNL			$\pm 1$	LSB	
Power supply rejection		50			dB	$U_{VDDr}(t) = 2.5 \text{ V} + 0.1 \text{ V} \sin(2\pi \cdot 1 \text{ kHz} \cdot t)^{(4)}$
- 3 dB freq. of postfilter		180	300	450	kHz	1 <sup>st</sup> order filter
<b>Conditions</b>						
Load capacitance				40	pF	
Load current	$I_{PAOUT}$	0		5	mA	
Dropout voltage	$U_{drop+}$	-50 -100 -330			mV	$I_{PAOUT} = +0.05 \text{ mA}$ $I_{PAOUT} = +0.50 \text{ mA}$ $I_{PAOUT} = +5.00 \text{ mA}$ $U_{drop+}$ is referred to $VDDbb^{(5)}$
	$U_{drop-}$			50	mV	$U_{drop-}$ is referred to AGND <sup>(5)</sup>

<sup>1)</sup> Variation due to process tolerances and change in  $VDDbb$ , temperature and lifetime.

<sup>2)</sup> Variation due to change in  $VDDbb$ , temperature and life time.



**CONFIDENTIAL**

- 3) Saturation voltage is the output voltage if the output is completely saturated. Nonlinear settling to be expected.  
Saturation voltages are given with respect to AGND and VDDbb.  
Saturation voltage is independent on the offset voltage.  
Voltage drops on wire resistances on the chip and within the package are included.  
Saturation voltage has to be considered with respect to the actual output voltage, i.e. offset and gain errors as well as tolerances of VDDbb have to be considered.
- 4) Valid within the linear operating range between  $AGND + U_{dron-}$  and  $VDDbb + U_{dron+}$ .
- 5) Dropout voltage is a condition for the maximum/minimum output voltage for which a linear signal processing is achieved. Dropout voltages are given with respect to AGND and VDDbb.  
Dropout voltage is independent on the offset voltage.  
Voltage drops on wire resistances on the chip and within the package are included.  
Dropout voltage has to be considered with respect to the actual output voltage, i.e. offset and gain errors as well as tolerances of VDDbb have to be considered.

**Table 12-55 Specification of Pins PAOUT1 (TXON = 0)**

Parameter	Symbol	Limit Values			Unit	Test Condition/ Remark
		Minimum	Typical	Maximum		
Leakage current of ESD structure				10	$\mu A$	
Output resistance				500	$\Omega$	

**CONFIDENTIAL**

CONFIDENTIAL

**CONFIDENTIAL**

## 13.2.3 RF Electrical Data

### 13.2.3.1 Operating Ranges

Within the operational range (refer to [Table 12-56](#)) the IC operates as explained in the functional description.

The 2.5 V supply domains are within 2.4 to 2.6 V.

The 1.5 V supply domains are within 1.4 to 1.6 V.

**Table 12-56 Operating Range,  $T_{AMB} = -30^{\circ}\text{C} \dots +85^{\circ}\text{C}$**

#	Parameter	Symbol	Limit Values			Unit	Test Conditions
			min	typ	max		
RF Receiver Front-end (LNA)							
1	RX1/RX1X Input Frequency	f <sub>RX1</sub>	869		894	MHz	GSM900 operation under investigation
2	RX2/RX2X Input Frequency	f <sub>RX2</sub>	925		960	MHz	GSM850 operation under investigation
3	RX3/RX3X Input Frequency	f <sub>RX3</sub>	1805		1880	MHz	GSM1900 operation under investigation
4	RX4/RX4X Input Frequency	f <sub>RX4</sub>	1930		1990	MHz	GSM1800 operation under investigation
5	Required Amplitude Balance at RX1/ RX1X and RX2/ RX2X Inputs		-1.0		1.0	dB	
6	Required Phase Balance at RX1/RX1X and RX2/RX2X Inputs		-10		10	deg.	
7	Required Amplitude Balance at RX3/ RX3X and RX4/ RX4X Inputs		-1.5		3.0	dB	

**CONFIDENTIAL**

**Table 12-56 Operating Range,  $T_{AMB} = -30^{\circ}\text{C} \dots +85^{\circ}\text{C}$**

#	Parameter	Symbol	Limit Values			Unit	Test Conditions
			min	typ	max		
8	Required Phase Balance at RX3/RX3X and RX4/RX4X Inputs		-12		15	deg	
<b>RF Receiver Outputs A/AX, B/BX</b>							
9	Maximum Load Capacitance	$C_{load}$		10	30	pF	Differential
<b>Modulator Section Inputs A/AX, B/BX</b>							
10	Common Mode I/Q Input Voltage	$V_{CM}$	0.9		1.4	V	
11	Differential Mode I/Q Input Voltage	$V_{DM}$	0.78	1	1.1	$V_{PP}$	
<b>Crystal Oscillator Output Buffers</b>							
12	FSYS2 Load (R    C) to GND	$C_{LFSYS2}$ $R_{LFSYS2}$	2		12	pF k $\Omega$	
13	FSYS3 Load (R    C) to GND	$C_{LFSYS3}$ $R_{LFSYS3}$	2		12	pF k $\Omega$	
<b>Crystal Requirements</b>							
14	Nominal Load Capacitance	$C_{Lnom}$		8 10		pF	
15	Motional Capacitance	$C_{18pF}$ $C_{110pF}$	2.88 5.3	3.6 6.6	4.3 7.9	fF	Acc. to Nominal Load Capacitance
16	Shunt Capacitance	$C_{08pF}$ $C_{010pF}$	0.8 1.3	1.0 1.6	1.2 1.9	pF	Acc. to Nominal Load Capacitance

**CONFIDENTIAL**

**Table 12-56 Operating Range,  $T_{AMB} = -30^{\circ}\text{C} \dots +85^{\circ}\text{C}$**

#	Parameter	Symbol	Limit Values			Unit	Test Conditions
			min	typ	max		
17	Equivalent Series Resistance	Rr			40	$\Omega$	1)
18	Overall Frequency Tolerance	$\Delta f_{XTAL}$	-26		26	ppm	All Crystal Specific Contributions (Base Tolerance, Aging, Temperature, etc.)

1) Rr is the real part of the crystal impedance.

### 13.2.3.2 AC/DC Characteristics

AC/DC characteristics involve the spread of values valid within the specified supply voltage and ambient temperature, refer to [Table 12-57](#). Typical characteristics are the median of the production. All typical values are valid for 1.5V and 2.5V supply voltage at  $T_A = 25^{\circ}\text{C}$ .

**Table 12-57 AC/DC Characteristics**

#	Parameter	Symbol	Limit Values			Unit	Test Conditions	Item
			Min	Typ	Max			
1. Receiver Supply Current								
	2.5V supply	VDDRX		25.5	29.5	mA		
	2.5V supply, DCXO	VDDXO		1.3	2	mA	Z <sub>L</sub> = 12pF//30kΩ, XOcal = 4, FSYS1 = ON FSYS2 = OFF FSYS3 = OFF	
	1.5V supply	VDDDIG		0.5	1	mA		
	1.5V supply	VDDTRX		12.5	15	mA		
	1.5V supply	VDDLO		37 49	60	mA	Typical values: 37: GSM 850/900 49: GSM 1800/1900	
	2.5V supply	VDDVCO		41.5	50	mA		

**CONFIDENTIAL**

**Table 12-57 AC/DC Characteristics**

#	Parameter	Symbol	Limit Values			Unit	Test Conditions	Item
			Min	Typ	Max			
	Subtotal 1.5V			49 61	76	mA	<sup>1)</sup> Typical values: 49: GSM 850/900 61: GSM 1800/1900	
	Subtotal 2.5V			69	82	mA	<sup>2)</sup>	
	Total			257	339	mW		

## 2. Transmitter Supply Current

	2.5V supply	VDDRX		6.5	7.5	mA		
	2.5V supply, DCXO	VDDXO		1.3	2.0	mA	$Z_L = 12\text{pF}/30\text{k}\Omega$ , XOcal = 4, FSYS1 = ON FSYS2 = OFF FSYS3 = OFF	
	1.5V supply	VDDDIG		0.55	1	mA		
	1.5V supply	VDDTRX		34	49	mA		
	1.5V supply	VDDLO		12	18	mA		
	2.5V supply	VDDVCO		41.5	50	mA		
	Subtotal 1.5V			47	69	mA	<sup>1</sup>	
	Subtotal 2.5V			50.3	60.5	mA	<sup>2</sup>	
	Total			209	268	mW		

## 3. Standby Current

	Standby Mode Current Consumption	$I_{\text{sby}}$			100	$\mu\text{A}$	Sum of Standby Currents on all 1.5 V Supply Lines, $T_A = 25^\circ\text{C}$	
--	--	------------------	--	--	-----	---------------	--	--

## 4. Overall Receiver Characteristics

RX1/RX1X;  $f_{\text{RX1}} = 869\text{ MHz}..894\text{ MHz}$  (GSM850)

RX2/RX2X;  $f_{\text{RX2}} = 925\text{ MHz}..960\text{ MHz}$  (EGSM900)

All values valid for RXGAIN[1:0] = 11, RXGS[3:0] = 0000, RXCORR[3:0] = 0100 (default = 0dB) unless otherwise stated. All measurements refer to **Figure 13-35 RX Measurement Setup (on Page 1397)** (differential receiver output A/AX or B/BX).

	Adjusted Overall Gain	$G_{\text{RX1/2\_H\_adj}}$	56.1	57.0	57.9	dB	RXCORR[3:0] applied <sup>3)</sup>	<sup>4)</sup>
	Overall High Gain	$G_{\text{RX1/2\_H}}$	51.5	57.0	62.5	dB	<b>3</b>	

**CONFIDENTIAL**

**Table 12-57 AC/DC Characteristics**

#	Parameter	Symbol	Limit Values			Unit	Test Conditions	Item
			Min	Typ	Max			
	Overall Medium Gain	$G_{RX1/2\_M}$	37.5	43.0	48.5	dB	RXGAIN[1:0] = 013	
	Overall Low Gain	$G_{RX1/2\_L}$	16.5	22.0	27.5	dB	RXGAIN[1:0] = 003	
	Noise Figure	$NF_{RX1/2}$		2.4	3.7	dB	Spot NF @80 kHz <sup>5)</sup>	4
	Noise Figure	$NF_{RX1/2\_M}$			8.5	dB	Spot NF5 @80 kHz, RXGAIN[1:0] = 01	4
	Noise Figure	$NF_{RX1/2\_L}$			30	dB	Spot NF5 @80 kHz, RXGAIN[1:0] = 00	4
	Input Referred Integral Noise Power	$N_{intRX1/2}$		-123.6	-120.8	dBm	1 kHz..30 kHz <sup>6)</sup>	
	Desensitization NF (incl. LO Phase Noise) $NF_{tot} @ P_{Blocker}$	$NFDES_{RX1/2\_0M6}$		3.5	8.0	dB	$\Delta f = 600$ kHz $P_{in} = -45$ dBm <sup>7)</sup>	4
	Desensitization NF (incl. LO Phase Noise) $NF_{tot} @ P_{Blocker}$	$NFDES_{RX1/2\_1M6}$		3.5	8.0	dB	$\Delta f = 1.6$ MHz $P_{in} = -35$ dBm <sup>7)</sup>	4
	Desensitization NF (incl. LO Phase Noise) $NF_{tot} @ P_{Blocker}$	$NFDES_{RX1/2\_3M}$		4.5	8.0	dB	$\Delta f = 3$ MHz, $P_{in} = -25$ dBm <sup>7)</sup>	4
	Desensitization NF (incl. LO Phase Noise) $NF_{tot} @ P_{Blocker}$	$NFDES_{RX1/2\_20M}$		3.5	7.5	dB	$\Delta f = 20$ MHz, $P_{in} = -25$ dBm <sup>7)</sup>	4
	3rd Order Input Intercept Point	$IIP3_{RX1/2}$	-15	-12		dBm	9	4
	AM Suppression	$AM_{RX1/2}$	82	90		dB	$\Delta f > 6$ MHz, $T_A = 25^\circ C$ <sup>11)</sup>	
	Harmonic Suppression @ n x 26MHz	$a_{n26\_RX1/2}$		70		dB	Harmonics within RX Band <sup>11)</sup>	4

**CONFIDENTIAL**

**Table 12-57 AC/DC Characteristics**

#	Parameter	Symbol	Limit Values			Unit	Test Conditions	Item
			Min	Typ	Max			
	Harmonic Suppression $0.5 \cdot f_{LO}$	$a_{05LO\_RX1/2}$		55		dB	$T_A = 25^\circ\text{C}$ , $P_{in,IC} = -50 \text{ dBm}^{11}$	4
	Harmonic Suppression $2 \cdot f_{LO}$	$a_{2LO\_RX1/2}$		55		dB	$T_A = 25^\circ\text{C}^{11}$	4
	Harmonic Suppression $3 \cdot f_{LO}$	$a_{3LO\_RX1/2}$		20		dB	Differential Mode Blocker, $T_A = 25^\circ\text{C}^{11}$	4
	Harmonic Suppression $3 \cdot f_{LO}$	$a_{3LO\_CM\_RX1/2}$		40		dB	Common Mode Blocker, $T_A = 25^\circ\text{C}^{11}$	4
	Harmonic suppression $4 \cdot f_{LO}$	$a_{4LO\_RX1/2}$		50		dB	$T_A = 25^\circ\text{C}^{11}$	4
	Harmonic Suppression $5 \cdot f_{LO}$	$a_{5LO\_RX1/2}$		30		dB	Differential Mode Blocker, $T_A = 25^\circ\text{C}^{11}$	4
	Harmonic Suppression $5 \cdot f_{LO}$	$a_{5LO\_CM\_RX1/2}$		50		dB	Common Mode Blocker, $T_A = 25^\circ\text{C}^{11}$	4
	Spurious Emission at RF Port RX1/RX2			-43	-33	dBm	@VCO-Frequency <sup>11</sup>	4
	Spurious Emission at RF Port RX1/RX2			N/A	-57	dBm	Spurious $\leq 1 \text{ GHz}^{11}$	4
	Spurious Emission at RF Port RX1/RX2			-50	-47	dBm	Spurious $> 1 \text{ GHz}^{11}$	4
	RX1 Input Impedance (differential)	$Z_{RX1}$		50-j204		$\Omega$	$f = 882 \text{ MHz}^{11}$	4
	RX2 Input Impedance (differential)	$Z_{RX2}$		57-j220		$\Omega$	$f = 942 \text{ MHz}^{11}$	4



**CONFIDENTIAL**

**Table 12-57 AC/DC Characteristics**

#	Parameter	Symbol	Limit Values			Unit	Test Conditions	Item
			Min	Typ	Max			
5. Overall Receiver Characteristics								
RX3/RX3X; f <sub>RX3</sub> = 1805MHz..1880MHz (GSM1800)								
RX4/RX4X; f <sub>RX4</sub> = 1930MHz..1990MHz (GSM1900)								
All values valid for RXGAIN[1:0] = 11, RXGS[3:0] = 0000, RXCORR[3:0] = 0100 (default = 0dB) unless otherwise stated. All measurements refer to <b>Figure 13-35 RX Measurement Setup (on Page 1397)</b> (differential receiver output A/AX or B/BX).								
	Adjusted Overall Gain	G <sub>RX3/4_H_adj</sub>	55.1	56	56.9	dB	RXCORR[3:0] applied <sup>3</sup>	4
	Overall High Gain	G <sub>RX3/4_H</sub>	50.5	56.0	61.5	dB	3	
	Overall Medium Gain	G <sub>RX3/4_M</sub>	36.5	42.0	47.5	dB	RXGAIN[1:0] = 01 <sup>3</sup>	
	Overall Low Gain	G <sub>RX3/4_L</sub>	16.5	22.0	27.5	dB	RXGAIN[1:0] = 00 <sup>3</sup>	
	Noise Figure	NF <sub>RX3/4</sub>		2.7	4.3	dB	Spot NF @80 kHz <sup>5</sup>	4
	Noise Figure	NF <sub>RX3/4_M</sub>			10	dB	Spot NF <sup>5</sup> @80 kHz, RXGAIN[1:0] = 01	4
	Noise Figure	NF <sub>RX3/4_L</sub>			30	dB	Spot NF <sup>5</sup> @80 kHz, RXGAIN[1:0] = 00	4
	Input Referred Integral Noise Power	N <sub>intRX3/4</sub>		-122.8	-120.0	dBm	1 kHz..30 kHz <sup>8)</sup>	
	Desensitization NF (incl. LO Phase Noise) NF <sub>tot</sub> @ P <sub>Blocker</sub>	NFDES <sub>RX3/4_0M6</sub>		6	8	dB	T <sub>A</sub> = 25°C, Δf = 600 kHz, Pin = -45 dBm <sup>7</sup>	4
	Desensitization NF (incl. LO Phase Noise) NF <sub>tot</sub> @ P <sub>Blocker</sub>	NFDES <sub>RX3/4_1M6</sub>		6	8	dB	T <sub>A</sub> = 25°C, Δf = 1.6 MHz, Pin = -35 dBm <sup>7</sup>	4
	Desensitization NF (incl. LO Phase Noise) NF <sub>tot</sub> @ P <sub>Blocker</sub>	NFDES <sub>RX3/4_3M</sub>		6	8	dB	T <sub>A</sub> = 25°C, Δf = 3 MHz, Pin = -28 dBm <sup>7</sup>	4
	Desensitization NF (incl. LO Phase Noise) NF <sub>tot</sub> @ P <sub>Blocker</sub>	NFDES <sub>RX4_20M</sub>		4	7.5	dB	Δf = 20 MHz, Pin = -27 dBm <sup>7</sup> GSM1800 only	4

**CONFIDENTIAL**

**Table 12-57 AC/DC Characteristics**

#	Parameter	Symbol	Limit Values			Unit	Test Conditions	Item
			Min	Typ	Max			
	Desensitization NF (incl. LO Phase Noise) NF <sub>tot</sub> @ P <sub>Blocker</sub>	NFDES <sub>RX4_20M</sub>		4	7.5	dB	$\Delta f = 20 \text{ MHz}$ , Pin = -26 dBm <sup>7</sup> GSM1900 only	4
	3rd Order Input Intercept Point	IIP3 <sub>RX3/4</sub>	-15	-12		dBm	<sup>9)</sup>	4
	AM Suppression	AM <sub>RX3/4</sub>	82	90		dB	$\Delta f > 6 \text{ MHz}$ , T <sub>A</sub> = 25°C <sup>10)</sup>	
	Harmonic Suppression @n x 26MHz	a <sub>n26_RX3/4</sub>		70		dB	Harmonics within RX Band <sup>11)</sup>	4
	Harmonic Suppression 0.5*f <sub>LO</sub>	a <sub>05LO_RX3/4</sub>		55		dB	T <sub>A</sub> = 25°C, P <sub>in,IC</sub> = -56 dBm <sup>11</sup>	4
	Harmonic Suppression 2*f <sub>LO</sub>	a <sub>2LO_RX3/4</sub>		55		dB	T <sub>A</sub> = 25°C <sup>11</sup>	4
	Harmonic Suppression 3*f <sub>LO</sub>	a <sub>3LO_RX3/4</sub>		25		dB	Differential Mode Blocker, T <sub>A</sub> = 25°C <sup>11</sup>	4
	Harmonic Suppression 3*f <sub>LO</sub>	a <sub>3LO_CM_RX3/4</sub>		45		dB	Common Mode Blocker, T <sub>A</sub> = 25°C <sup>11</sup>	4
	Harmonic Suppression 4*f <sub>LO</sub>	a <sub>4LO_RX3/4</sub>		55		dB	T <sub>A</sub> = 25°C <sup>11</sup>	4
	Harmonic Suppression 5*f <sub>LO</sub>	a <sub>5LO_RX3/4</sub>		40		dB	Differential Mode Blocker, T <sub>A</sub> = 25°C <sup>11</sup>	4
	Harmonic Suppression 5*f <sub>LO</sub>	a <sub>5LO_CM_RX3/4</sub>		60		dB	Common Mode Blocker, T <sub>A</sub> = 25°C <sup>11</sup>	4
	Spurious Emission at RF Port RX3/RX4			-46	-33	dBm	@VCO-Frequency <sup>11</sup>	4
				N/A	-57	dBm	Spurious ≤ 1GHz <sup>11</sup>	4
				-63	-47	dBm	Spurious > 1GHz <sup>11</sup>	4
	RX3 Input Impedance (differential)	Z <sub>RX3</sub>		22-j150		Ω	f = 1842 MHz <sup>11</sup>	4

**CONFIDENTIAL**

**Table 12-57 AC/DC Characteristics**

#	Parameter	Symbol	Limit Values			Unit	Test Conditions	Item
			Min	Typ	Max			
	RX4 Input Impedance (differential)	$Z_{RX4}$		20-j138		$\Omega$	$f = 1960 \text{ MHz}^{11}$	<b>4</b>

## 6. Baseband Filter Response

3dB Roll Off Frequency	$f_{3dB}$	132	142	152	kHz		<b>4</b>
Group Delay Ripple	$\Delta\tau_g$		530	620	ns	0..80kHz	<b>4</b>
Passband Ripple	$\Delta a_p$		0.2	0.3	dB	0..80 kHz	<b>4</b>
Stopband Attenuation (Unmodulated Source)	$a_{s\_200k}$	7.0	8.5		dB	at 200 kHz	<b>4</b>
	$a_{s\_400k}$	22.0	24.0		dB	at 400 kHz	
	$a_{s\_600k}$	32.0	34.0		dB	at 600 kHz	<b>4</b>
	$a_{s\_800k}$	39.0	41.0		dB	at 800 kHz	
	$a_{s\_1.6M}$	57.0	59.0		dB	at 1.6 MHz	<b>4</b>
	$a_{s\_3M}$	73.0	75.0		dB	at 3 MHz	<b>4</b>
	$a_{s\_13M}$	96.0	120		dB	at 12.9 MHz	<b>4</b>
Stopband Attenuation (GSM Modulated Source)	$a_{sGSM\_200k}$	5.5	6.5		dB	at 200 kHz	<b>4</b>
	$a_{sGSM\_400k}$	21.0	22.5		dB	at 400 kHz	<b>4</b>
	$a_{sGSM\_600k}$	31.0	33.0		dB	at 600 kHz	<b>4</b>

## 7. PGC, Output Signal A/AX and B/BX

Gain Correction Range	$\Delta G_{GCR}$	-6		+6	dB		
Gain Correction Step	$\Delta G_{GCS}$		1		dB		
Gain Step 0	$\Delta G_{RXGS0}$	11.8	12	12.2	dB		
Gain Step 1	$\Delta G_{RXGS1}$	5.8	6	6.2	dB		
Gain Step 2	$\Delta G_{RXGS2}$	-5.8	-6	-6.2	dB		
Gain Step 3	$\Delta G_{RXGS3}$	-2.8	-3	-3.2	dB		
DC Output level (Common Mode)	$V_{DC\_CM}$		0.95		V	RXCM[1:0] = 00	
			1.25		V	RXCM[1:0] = 01	
			1.35		V	RXCM[1:0] = 10	
			1.425		V	RXCM[1:0] = 11	

**CONFIDENTIAL**

**Table 12-57 AC/DC Characteristics**

#	Parameter	Symbol	Limit Values			Unit	Test Conditions	Item
			Min	Typ	Max			
	DC Offset (Differential Mode)	$V_{DC\_DM}$		< 0.02	0.15	V	RXGAIN[1:0] = 11 RXCORR[3:0] = 0100 RXGS[3:0] = 0010 OFC = 0	
	DC Offset (Differential Mode) after Compensation	$V_{DC\_DM\_adj}$		< 0.02	0.2	V	RXGAIN[1:0] = 11 RXCORR[3:0] = 0100 RXGS[3:0] = 0011 OFC = 1	
	I/Q Phase Error	$\Delta\phi_{I/Q}$		< 0.01	3	deg	<sup>12)</sup> RXGAIN[1:0] = 11	
	I/Q Amplitude Mismatch	$\Delta V_{I/Q}$		< 0.1	0.5	dB	<sup>12)</sup> RXGAIN[1:0] = 11	
	Short Term Offset Drift (Differential Mode)	$V_{Drift}$		< 100	400	$\mu V$	<sup>13)</sup> RXGAIN[1:0] = 11 RXCORR[3:0] = 0100 RXGS[3:0] = 0010 OFC = 0	<b>4</b>
	Output Resistance			200	500	$\Omega$		

## 8. Overall Transmitter Characteristics

GSM850/900 TX1;  $f_{TX}=824\text{ MHz} \dots 849\text{ MHz}$ ;  $f_{TX}=880\text{ MHz} \dots 915\text{ MHz}$

	Output Level	$P_{TX1}$	1.5	3.5	5.5	dBm	Measurement Setup according to <a href="#">Figure 13-35</a>	
	Output Impedance	$Z_{TX1}$		50		W		<b>4</b>
	Output Return Loss	$RL_{TX1}$	15	22		dB		<b>4</b>
	Output Noise Floor	$N_{TX1\_20M}$		-165.5	-163	dBc/Hz	$ \Delta f  = 20\text{ MHz}^{14)}$ RBW = 100 kHz $T_A = 25^\circ C$	<b>4</b>
	RMS Phase Error	$E_{RMS\_TX1}$		0.7	2.5	$^\circ$ rms		<b>4</b>
	Peak Phase Error	$E_{peak\_TX1}$		2	7.5	$^\circ$ peak		<b>4</b>

**CONFIDENTIAL**

**Table 12-57 AC/DC Characteristics**

#	Parameter	Symbol	Limit Values			Unit	Test Conditions	Item
			Min	Typ	Max			
	PRBS Modulation Spectrum <sup>15)</sup> @200 kHz offset @250 kHz offset @400 kHz offset @600 kHz offset @1800 kHz offset @3000 kHz offset @6000 kHz offset			-37 -41 -69 -75 -86 -87 -90	-33 -36 -62- 67 -73 -75 -80	dB	<sup>16)</sup>  RBW = 30 kHz RBW = 30 kHz RBW = 30 kHz <b>4</b>  RBW = 30 kHz <b>4</b> RBW = 100 kHz <b>4</b>  RBW = 100 kHz <b>4</b> RBW = 100 kHz <b>4</b>	

**9. Overall Transmitter characteristics**

GSM1800/1900 TX2;  $f_{TX}=1710\text{ MHz}...1785\text{ MHz}$ ;  $f_{TX} = 1850\text{ MHz}...1910\text{ MHz}$

Output Level	$P_{TX2}$	1.5	3.5	5.5	dBm	Measurement Setup according to <b>Figure 13-35</b>	
Output Impedance	$Z_{TX2}$		50		$\Omega$		
Output Return Loss	$RL_{TX2}$	15	21		dB		
Output Noise Floor	$N_{TX2\_20M}$		-159	-154	dBc/Hz	$ \Delta f  = 20\text{ MHz}$ <b>14</b> RBW = 100 kHz $T_A = 25^\circ\text{C}$	
RMS Phase Error	$E_{RMS\_TX2}$		1.2	2.8	$^\circ\text{ rms}$		
Peak Phase Error	$E_{peak\_TX2}$		4	7.5	$^\circ\text{ peak}$		
PRBS Modulation Spectrum <sup>17)</sup> @200 kHz offset @250 kHz offset @400 kHz offset @600 kHz offset @1800 kHz offset @6000 kHz offset			-37 -41 -67 -73 -83 -85	-33 -36 -62- 65 -73 -90	dB	<sup>18)</sup>  RBW = 30 kHz RBW = 30 kHz RBW = 30 kHz <b>4</b>  RBW = 30 kHz <b>4</b> RBW = 100 kHz <b>4</b>  RBW=100 kHz <b>4</b>	

**10. Transmitter Baseband Inputs A/AX, B/BX**

**CONFIDENTIAL**

**Table 12-57 AC/DC Characteristics**

#	Parameter	Symbol	Limit Values			Unit	Test Conditions	Item
			Min	Typ	Max			
	Input Resistance	$R_{inTX}$		100		$k\Omega$	TX Mode only Single Ended	
	Input Capacitance	$C_{inTX}$		5		pF	TX Mode only Single Ended	

### 11. Sigma-Delta Synthesizer

Settling time (5° residual phase error)	$t_{settle}$		150	190	$\mu s$	5° refer to $f_{TX} = 1990$ MHz	
SSB Inband Phase Noise of PLL @ 1 kHz	$PN_{PLL\_1K}$		-77	-75	dBc/Hz	Measured at TX2	
SSB Inband Phase Noise of PLL @ 40 kHz	$PN_{PLL\_40K}$		-92	-88.5	dBc/Hz	Measured at TX2	
Peak Phase Error due to Drop of Supply Voltages	$ E_{peak\_drop} $		5	10	°peak	Measured at TX2 either 1.5 or 2.5 V or both Supply Domains Drop Linearly by 10 mV within 2 $\mu s$	

### 12. Crystal Oscillator 26 MHz (DCXO)

#### Digitally Controlled Crystal Oscillator Core

Startup Time	$t_{start}$			5	ms	$ \Delta f  \leq 1.5$ ppm 90% Amplitude	
Tuning Sensitivity	$k_{XO}$		0.01	0.025	ppm/LSB		
Frequency Pushing	$k_{push\_XO}$		0.25	1.5	ppm/V	Measured with Nominal VDDXO $\pm 100$ mV	
Negative Resistance	$R_{N\_1f}$		-200	-160	$\Omega$	Fundamental, Measured at -40dBm in Subrange 1	
	$R_{N\_3f}$	$(R_{N\_1f})/6$				3rd Harmonic	

**CONFIDENTIAL**

**Table 12-57 AC/DC Characteristics**

#	Parameter	Symbol	Limit Values			Unit	Test Conditions	Item
			Min	Typ	Max			
	Tuning Nonlinearity			3.5	12	%	20 Subranges compared with Full Range <sup>19)</sup>	
	Crystal Drive Level	P <sub>XTAL</sub>		20	60	μW		

**FSYS2 and FSYS3 Output Buffers**

	FSYS2, FSYS3 Output Voltage	V <sub>out</sub>	700	950	1200	mVpp	R <sub>L</sub> = 12 pF    2 kΩ	
	Duty Cycle	t <sub>H</sub>	40	50	60	% t <sub>sc</sub>		
	Peak Phase Jitter			8	50	ps		
	Phase Noise					dBc/ Hz	R <sub>L</sub> = 12 pF    2 kΩ  Δf  = 10 Hz	
		L <sub>FSYS2_10</sub>		-96	-85		Δf  = 100 Hz	
		L <sub>FSYS2_100</sub>		-132	-115		Δf  = 1 kHz	
		L <sub>FSYS2_1K</sub>		-141	-125		Δf  = 10 kHz As Bluetooth is connected only	
		L <sub>FSYS2_10K</sub>		-148	Pot. -132			
	Phase Pushing	φ <sub>pushFSYS2</sub>		0.12	0.25	°peak	VDDXO Step from 2.5 to 2.4 V	
	FSYS2, FSYS3 Current Consumption			580	700	mA	R <sub>L</sub> = 10pF    2kΩ	
				620	750		R <sub>L</sub> = 12pF    2kΩ	

1) Including PLL supply current I<sub>PLL t<sub>vn</sub></sub> = 14mA / I<sub>PLL max</sub> = 17mA

2) Including PLL supply current I<sub>PLL t<sub>vn</sub></sub> = 42mA / I<sub>PLL max</sub> = 51mA

3) Voltage gain according to [Figure 13-35 RX Measurement Setup \(on Page 1397\)](#).

4) Not subject to production test - verified by characterization and design.

5) Including worst case flicker noise contribution at 80kHz.

6) According to f<sub>c2kT0</sub> = 11.5kHz/NF<sub>enth</sub> = 3.5dB/α = 1, the mean value of channel A and B.

7) Desensitization measured with notch filter.

8) According to f<sub>c2kT0</sub> = 15kHz/NF<sub>enth</sub> = 4.0dB/α = 1, mean value of channel A and B.

9) P1, P2 = -49dBm, f1 = 800kHz, f2 = 1.6MHz.

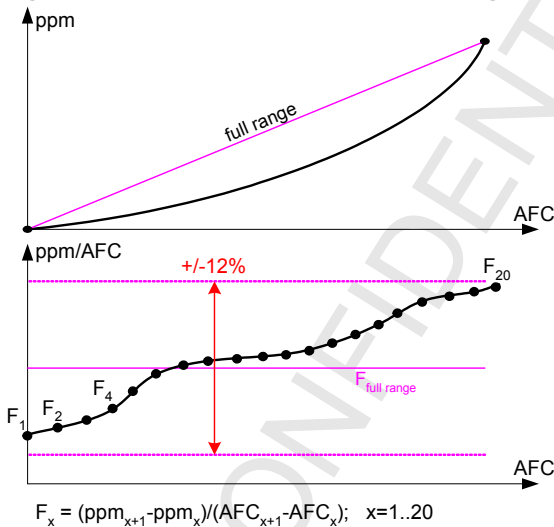
10) Single tone P<sub>R</sub> = -33dBm, P<sub>W</sub> = -101 dBm.

11) Measured on IFX S-parameter board.

**CONFIDENTIAL**

- 12)  $P_{in} = -70 \text{ dBm}$ ;  $f_{CW} = f_{LO} + 20 \text{ kHz}$ , measured from RF input to A,AX; B,BX.
- 13) Measurement starts  $100\mu\text{s}$  after sending RXTX word within  $577\mu\text{s}$ .
- 14)  $P_{carrier}$  and  $P_{noise}$  measured with "normal dBm marker" of spectrum analyzer  
 $N = P_{noise} - P_{carrier} - 10 \log (\text{RBW/Hz})$ .
- 15) Refer to GSM spec. 51.010.
- 16) VDDTRX has to fulfill the ripple mask according to [Figure 13-36 Ripple Mask for 1.5 V Supply \(on Page 1398\)](#).
- 17) Refer to GSM spec. 45.005.
- 18) VDD = 1.5V has to fulfill the ripple mask according to [Figure 13-36](#).
- 19) Refer to [Figure 13-33 Definition of the DCXO Tuning Nonlinearity \(on Page 1396\)](#).

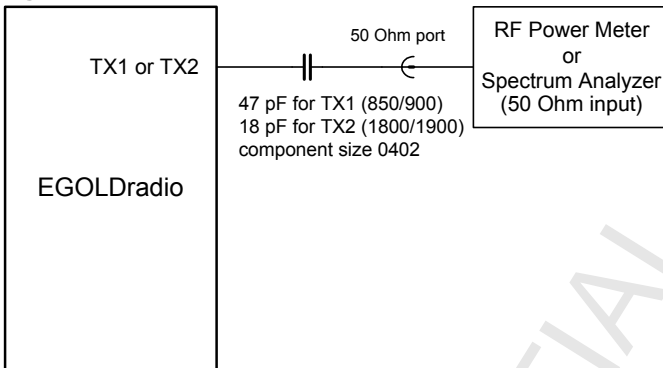
**Figure 13-33 Definition of the DCXO Tuning Nonlinearity**





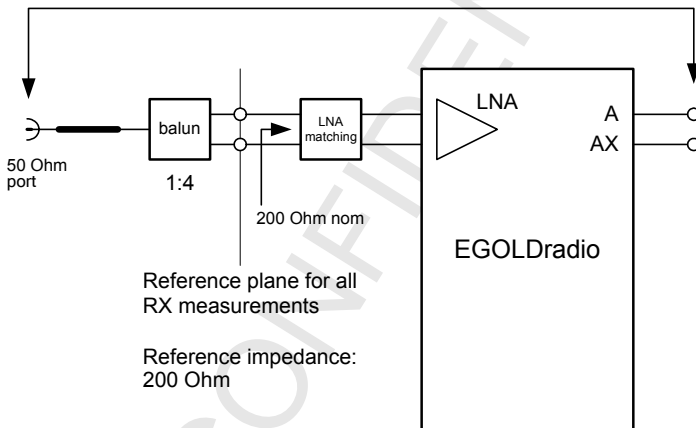
**CONFIDENTIAL**

**Figure 13-34 TX Power and Spectrum Measurement Setup**



**Figure 13-35 RX Measurement Setup**

Gain and NF measurements refer to the 50 Ohm port and are corrected for the measured balun and line loss



The receiver gain is defined as the voltage gain from single ended 50 Ohm input to differential voltage at A/AX output with an ideal (lossless) balun

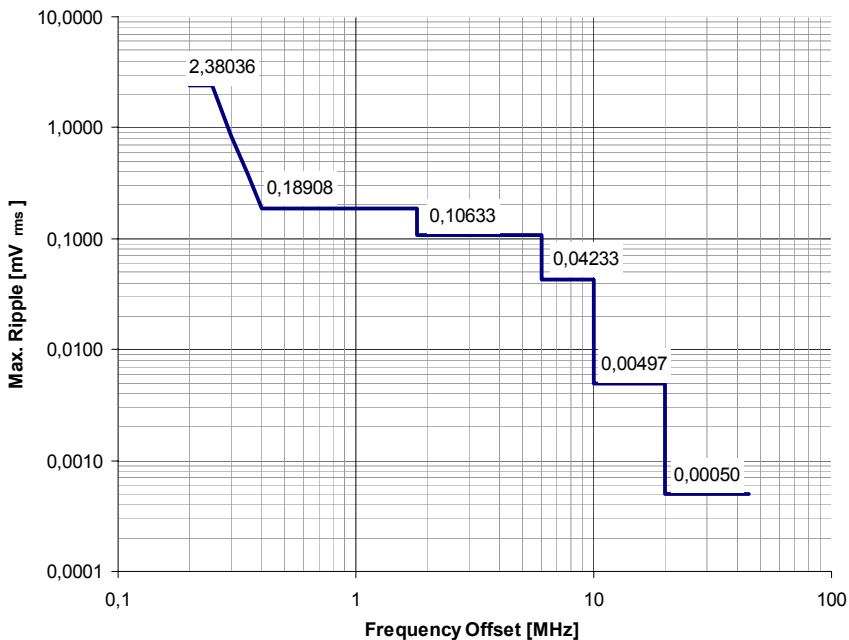
LNA matching circuits for all 4 bands:

Band	Balun type
850 MHz	Murata LDB15C201A881 with blocking capacitor 68pF/0402
900 MHz	Murata LDB15C201A942 with blocking capacitor 68pF/0402
1800 MHz	Murata LDB15C201A1800 with blocking capacitor 15pF/0402
1900 MHz	Murata LDB15C201A1900 with blocking capacitor 12pF/0402

CONFIDENTIAL

**Figure 13-36 Ripple Mask for 1.5 V Supply**

(To Avoid Unwanted TX Sidebands Caused by Remaining Ripple from DC/DC Converter)



*Note: The values given in [Figure 13-36](#) are preliminary and subject to be verified by first evaluation of E-GOLDradio.*

**CONFIDENTIAL**

$$DCstep = \frac{1}{\sqrt{2}} \left| \sqrt{\left(\frac{V_{DCstep}(I)}{V_{pp}}\right)^2 + \left(\frac{V_{DCstep}(Q)}{V_{pp}}\right)^2} \right| \cdot 100,$$

$$AMsuppression = AMinterferer - RFlevel - 20 \cdot \log\left(\frac{DCstep}{100}\right),$$

$$AMsuppression = AMinterferer - RFlevel - 20 \cdot \log\left(\frac{1}{\sqrt{2}} \cdot \left| \sqrt{\left(\frac{V_{DCstep}(I)}{V_{pp}}\right)^2 + \left(\frac{V_{DCstep}(Q)}{V_{pp}}\right)^2} \right| \right)$$

**CONFIDENTIAL**

CONFIDENTIAL

## 14 System Reset

History	
Current version: Rev. 1.06, 2005-11-04	
Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.06	

### 14.1 Introduction

The internal system reset function initializes the C166S into a defined state and is invoked in the following ways:

- By asserting a hardware reset signal RESET\_IN\_N (Hardware Reset input)
- On the execution of the SRST instruction (refer to [Section 14.1.1 Software Reset](#))
- By an overflow of the watchdog timer (refer to [Section 14.1.2 Watchdog Timer Reset](#)).

#### 14.1.1 Software Reset

The reset sequence can be triggered at any time via the protected instruction SRST. The instruction can be either:

- Executed deliberately within a program
- In a hardware trap routine that reveals a system failure.

The software reset performs the same reset functionality as the hardware reset except that the SIM Card block (and its corresponding pins) is not reset by the software reset. The SIM card interface can be reset via the CGU register [RST\\_CTRL\\_STA \(on Page 683\)](#).

#### 14.1.2 Watchdog Timer Reset

When the watchdog timer is not disabled during the initialization or serviced regularly during program execution it will overflow and trigger the reset sequence. The watchdog timer resets the same E-GOLDradio blocks as the software reset.

### **14.1.3 Reset of RTC, SIM Cards, DSP, and Analog**

The Real Time Clock, SIM card, DSP can separately be reset via the CGU register **RST\_CTRL\_STA** of the XBUS

The Analog part can be reset via the **SCCUSPCR (on Page 706)** (Standby Power Control register)

### **14.1.4 CGU Reset Block**

The Hardware reset from signal RESET\_IN\_n enters the CGU reset block. It is then delayed and re-synchronized on the external oscillator clock (13 MHz or 26 MHz). This signal "reset\_in\_sync\_s" is used internally to reset PLL and clock generator.

A delay line of 16 clock cycles is applied to make the signal long enough for the CPU. The delay line is synchronized on the external oscillator. This signal is the CPU C166S hardware reset. It enters the C166 SCU, which generates the reset signal "c166\_reset\_n". It has three sources:

1. HWD for hardware reset
2. SW for software reset
3. WDT for watchdog reset.

The reset sequence default duration is 4096 clock cycles. For the SW and WDT resets, the sequence duration can be adjusted to 1024, 2048, or 4096 clock cycles by programming the **RSTCON (on Page 237)** register. The SW reset is generated in the C166S core. The WDT reset is sent to the core when a timer overflow occurs.

The signal "c166\_reset\_n" is then switched back to the CGU and is used to reset PCL and peripherals on the XBus and PDbus. Therefore, all these devices are still in reset state after the CPU has finished its reset.

Once the reset sequence is finished, the C166S starts its init sequence. At the end of the init sequence (EOI). The signal ex\_rstout is switched back to the CGU and is used as a source for the DSP reset. At the end of the init sequence (EOI) of the MCU, the signal ex\_rstout is deactivated.

For the DSP there are four different reset sources:

1. Hardware
2. General software (external reset from C166S)
3. Specific software (from a register)
4. A programmable bit from the SCCU (Standby Clock Control Unit).

The reset is then re-synchronized inside DSP subsystem. A flag/interrupt handshake occurs between DSP and CPU to wait for DSP boot (see **Figure 14-1 (on page 1405)**). The DSP is responsible for resetting its peripherals including shared memories.

The Analog part has four different reset sources: one hardware, two software (SW and WDT from C166 SCU) and a programmable bit from the SCCU .

CONFIDENTIAL

Introduction

Two reset request lines are routed from the SCCU to the CGU. These signals correspond to bits **SCCUSPCR.DPDN** and **SCCUSPCR.ADPN**.

The RTC can only be reset by a specific software reset (**RST\_CTRL\_STA.RTC\_RESET**).

The SIM card is reset using either hardware reset or a specific software reset (**RST\_CTRL\_STA.SIM\_RESET**).

*Note: Other devices (such as Timer, ASC, ...) do not have specific software resets.*

The Pad frame is reset by an external reset except for SIM card, which is reset by either a specific external signal or RTC (Power Isolation).

MON1 and MON2 on PCL are latched on the same hardware reset as the C166S reset.

The PCL has two control signal: EOI (End of Init, the C166S external reset) and EOR (End of Reset, the C166S reset).

**CONFIDENTIAL**

**Reset**

## 14.2 Reset

**Table 13-1 Block Reset Actions**

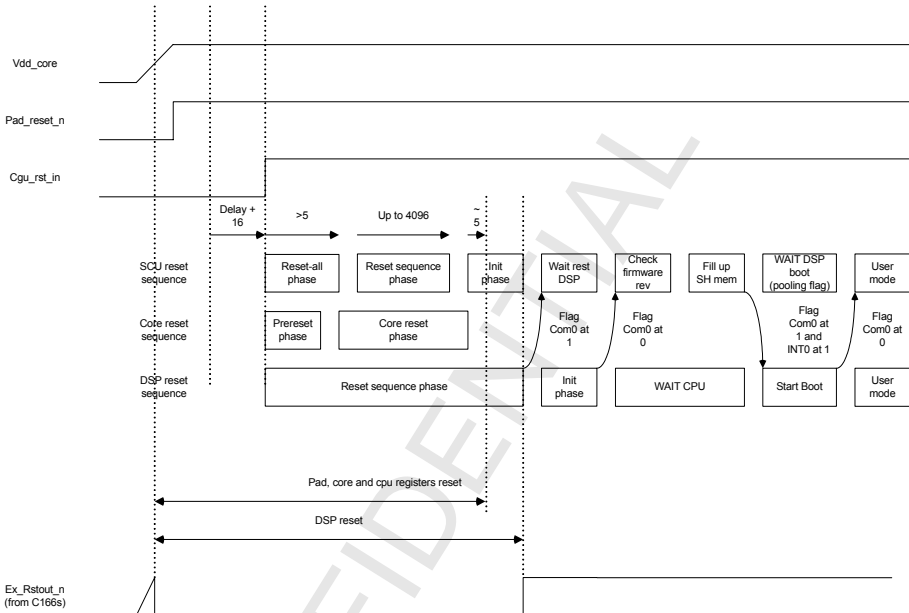
Block	MCU Output	External Reset	SW or WDT reset	DSP Standby power	Analog Standby Power	Module Reset
SCCU	Cgu_reset (active low)	R	Not affected	Not affected	Not affected	Not affected
DSP subsystem	Cgu_reset_dsp (active low)	R	R	R	Not affected	R
Analog block interfaces	Cgu_reset_ana (active low)	R	R	Not affected	R	Not affected
Xbus and PDbus peripherals	Cgu_reset (active low)	R	R	Not affected	Not affected	Not affected
RTC	Cgu_reset_rtc (active low)	Not affected	Not affected	Not affected	Not affected	R
SIM card	Cgu_reset_sim (active low)	R	Not affected	Not affected	Not affected	R
SSC	Cgu_reset	R	R	Not affected	Not affected	Not affected
Timer Unit	Cgu_reset	R	R	Not affected	Not affected	Not affected
GSM unit	Cgu_reset	R	R	Not affected	Not affected	Not affected
Voice	Cgu_reset	R	R	Not affected	Not affected	Not affected
Equalizer Accelerator	Cgu_reset	R	R	Not affected	Not affected	Not affected
TAP	Cgu_reset	R	R	Not affected	Not affected	Not affected
SEIB	Cgu_reset	R	R	Not affected	Not affected	Not affected
Shared memory	From DSP	R	R	R	Not affected	Not affected



### 14.2.1 Reset Timing Diagrams

**Figure 14-1** represents the reset sequence between the C166 MCU and TEAKlite DSP.

### Figure 14-1 Reset Timing Diagram



After DSP reset deactivation, the software performs a handshake exchange between DSP and CPU using communication flag 0 and interrupt 0. This guaranties that DSP has booted properly before any further treatment takes place.

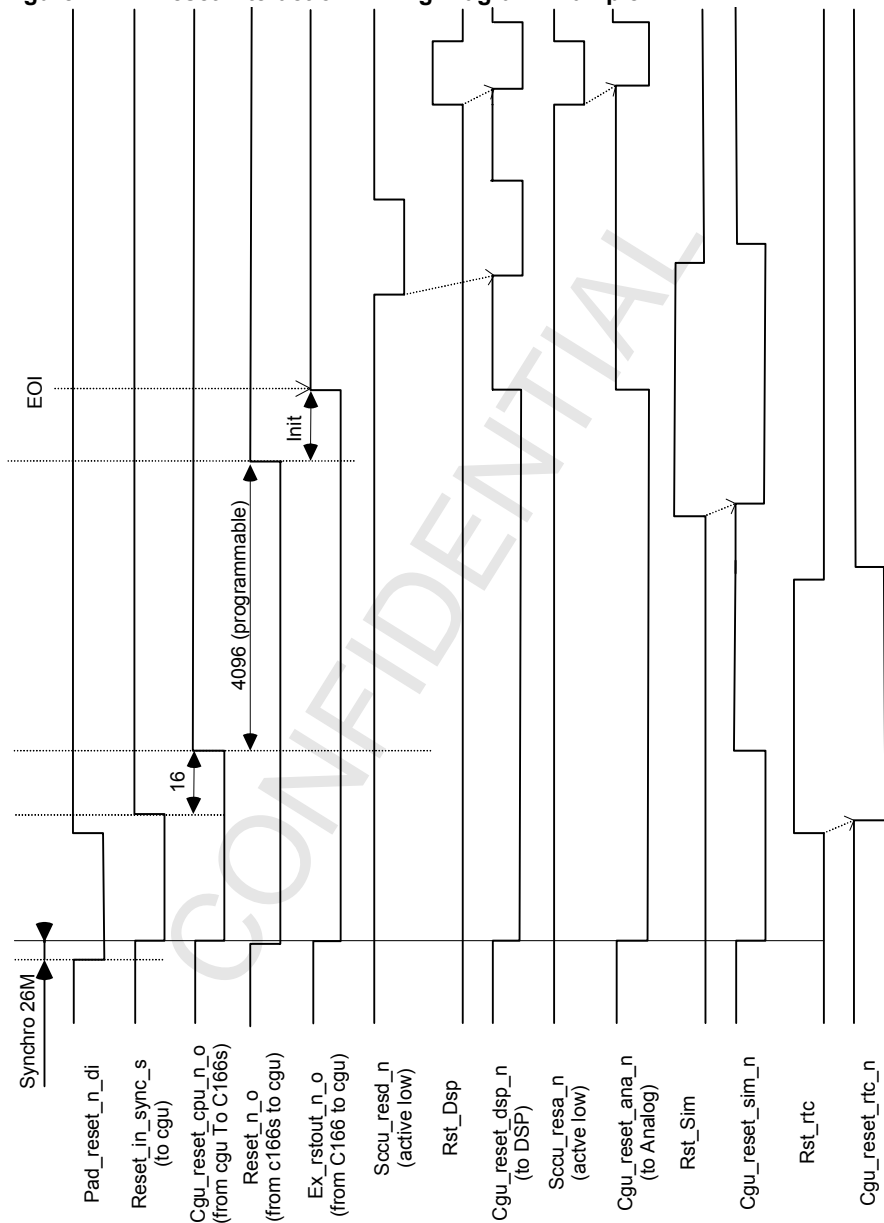
**Figure 14-2** is an example of a reset interaction timing diagram.

The reset from C166S has a minimum length of 1024 clock cycles. The length can be adjusted using register **RSTCON (on Page 237)** and has a default value of 4096 clock cycles. At the end of the init phase (EOI), the signal RSTOUT is deactivated.

The SW reset length of the SIM, RTC and DSP modules is controlled by software. Each reset signal is activated as soon as the corresponding bit in register

**RST\_CTRL\_STA (on Page 683)** is set to one. When the bit is reset to zero, the module exits.

Figure 14-2 Reset Interaction Timing Diagram Example



## 14.2.2 Power On/Off Sequences

For the major part of the outputs, the power-off tristate function permits deactivation of the outputs when proper operation cannot be guaranteed.

This function makes sure that the controlled PMB7870 outputs will never drive an undefined level even when the PMB7870 is switched off and the supply rail of an output pad is powered by an outside power source<sup>1)</sup>. It might pull up an PMB7870 output pad to a high level; Current could flow across the ESD protection circuitry of this pad and pull up the voltage of the pad supply rail. Then other pads supplied by the same rail could possibly drive a high output level. By switching all outputs into tristate this problem can be avoided.

The power-off pad tristate function does not apply to:

- SIM interface pins since special circuitry is provided there
- RTCOUT pin since this has to activate the power management IC
- TDO pin since it is a test pin that has to be available for testing under all conditions and is not used for control functions on the application board.

The power-off tristate function is controlled by the inputs RESET\_n and PM\_INT. These input pad cells are supplied by the RTC supply voltage. They retain full functionality even if (the mobile and) PMB7870 is powered off. To put the output pins of PMB7870 in tristate in power-off, these pins have to be controlled appropriately by the power management IC.

Pin PM\_INT retains its normal function (external interrupt) used by the power management IC as long as pin RESET\_n is deasserted. When pin RESET\_n is asserted pin PM\_INT can be used to set all outputs to tristate as shown in [Table 13-2](#). As long as input RESET\_n is deasserted the outputs are enabled.

The state of the outputs is controlled by the internal control signal PRG1 which is asserted whenever the outputs are tristated.

**Table 13-2 Power-Off Output Tristate Control**

input RESET_n	input PM_INT		Output State	Internal Control Signal PRG1
	Function	Value		
0	Output tristate control	0	Enabled	0
		1	Tristate	1
1	Interrupt	X	Enabled	0

The appropriate control sequence for signals PM\_INT and RESET\_n is described in [Section 14.2.2.3 Power-Up \(on Page 1408\)](#) for power-up transitions and [Section 14.2.2.1 Power-Off \(on Page 1408\)](#) for power-off transitions.

<sup>1)</sup> This power source might be an accessory attached to the powered down handset.

CONFIDENTIAL

Reset

PM\_INT, RESET\_IN, and RTC\_OUT have the specified IO functionality only if both:

- VDD\_RTC is in the Operating Range  $1.8\text{ V} < \text{VDD\_RTC} < 2.25\text{ V}$
- The other VDDs (such as VDD\_MAIN and VDD\_DSP) are also in operating range.

*Note: If  $\text{VDD\_RTC} < 1.8\text{ V}$  and  $\text{VDD\_MAIN}$  and  $\text{VDD\_DSP}$  are turned off, the functionality of these pins is not specified.*

### 14.2.2.1 Power-Off

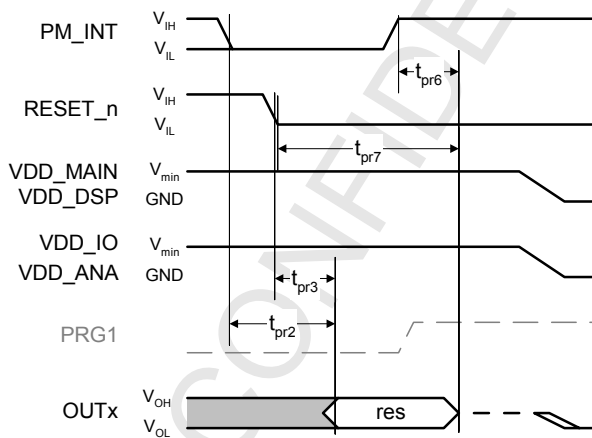
An example for a power-off sequence is shown in [Figure 14-3](#)

To drive the output signals to reset levels before power-off, both input signals PM\_INT and RESET\_n have to be driven low. Within less than  $t_{\text{pr2-max}} = 1\text{ }\mu\text{s}$  and  $t_{\text{pr3-max}} = 1\text{ }\mu\text{s}$ , respectively, the outputs drive their reset levels.

To set the output levels to tristate, input signal RESET\_n has to be driven low and input signal PM\_INT has to be driven high. Within less than  $t_{\text{pr7-max}} = 1\text{ }\mu\text{s}$  and  $t_{\text{pr6-max}} = 1\text{ }\mu\text{s}$ , respectively, the outputs will be tristated.

Thereafter, all regulators (except usually the RTC regulator) can be turned off.

**Figure 14-3 Power-Off**



### 14.2.2.2 Insertion of Battery

When the battery is inserted into the handset, the power management IC has to power up the RTC regulator and drive input signal PM\_INT high and input signal RESET\_n low to keep the outputs with the tristate function enabled.

### 14.2.2.3 Power-Up

The recommended power-up timing sequence is shown in [Figure 14-4](#).

**CONFIDENTIAL**

**Reset**

When the handset is turned on first VDD\_MAIN and VDD\_DSP must be powered up. The RESET\_n input buffer is supplied by the RTC supply voltage, therefore, the low level on input RESET\_n will be applied properly to the core logic. Due to the asynchronous internal reset, the core logic is in reset state within  $t_{pr4} = 1 \mu s$  max. after the core logic supply voltage reaches the specified level.

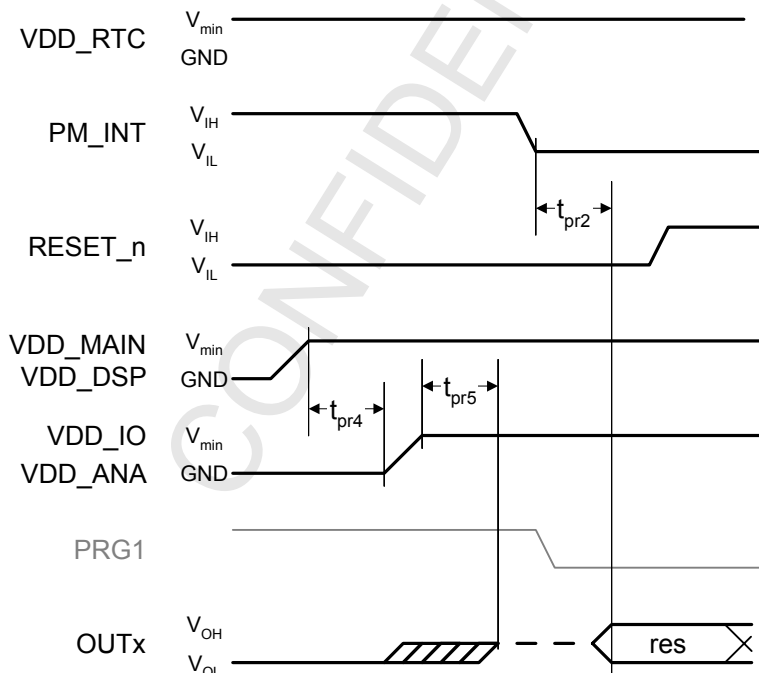
$t_{pr4-min} = 1 \mu s$  after turn-on of VDD\_MAIN and VDD\_DSP, the I/O- and analog supply voltages (VDD\_IO and VDD\_ANA) can be turned on. The output drivers will enter tristate within  $t_{pr5-max} = 1 \mu s$  as soon as their supply voltage has reached the minimum required level.

When the output drivers are enabled, input PM\_INT is set low and less then  $t_{pr2-max} = 1 \mu s$  later the outputs driven to the defined output levels.

Tristated outputs with pull-up or pull-down resistors may take longer to reach their levels (100  $\mu s$  ... 100 ms depending upon pull type and the external load capacity on the output).

Finally, input signal RESET\_n may be deasserted.

**Figure 14-4 Power-Up Sequence**



*Note: It is recommended to keep the output signals tristated until VDD\_IO has reached its minimum specified level and the output signals are in a stable tristate ( $t_{pr5}$*

**CONFIDENTIAL**

**Reset**

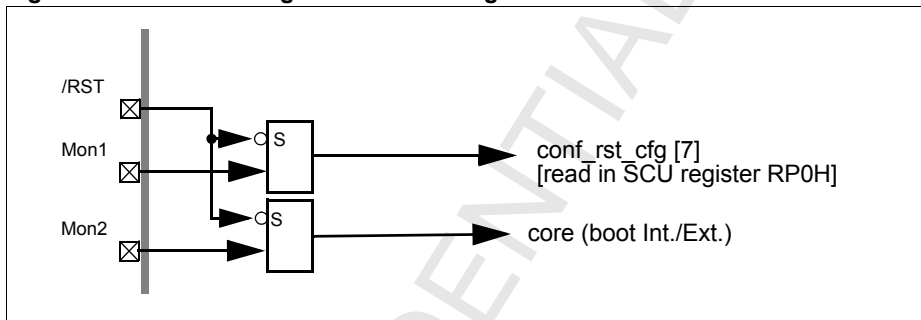
*elapsed). This minimizes the risk of spikes on the output signals during ramp-up of VDD\_IO. Signal PM\_INT must, therefore, be pulled low only after time  $t_{pr5}$  has elapsed.*

CONFIDENTIAL

### 14.2.3 CPU Boot Configuration

The basic PMB7870 CPU configuration setting (boot from internal ROM and 16-bit external data bus) is defined during **HW** RESET by the input levels on the pins MON1 and MON2 (see [Figure 14-5](#)). If the pins are not connected, then internal pull-downs ensure that the system starts with the default configuration, internal boot, and 16-bit external bus. This can be overridden by pulling-up the MON1 and/or MON2 pins while the RESET\_IN pin is active (for example, via external pull-up resistors), MON2 = 1 selects boot from external ROM and MON1 = 1 selects the 8-bit data bus.

**Figure 14-5 Boot Configuration Latch Register**



CONFIDENTIAL



**CONFIDENTIAL**

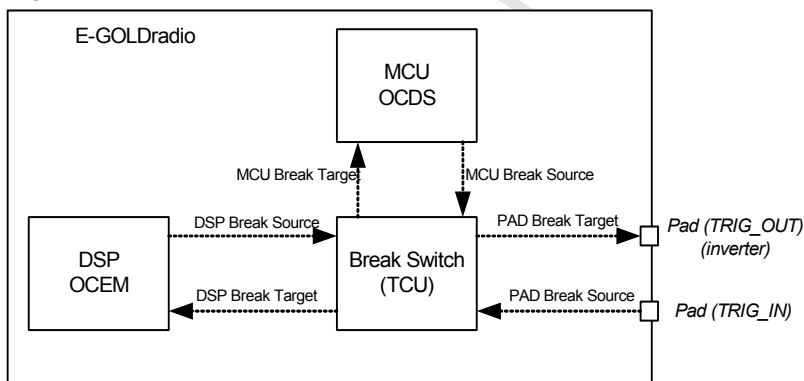
## 15 Debug

History	
Current version: Rev. 1.06, 2005-11-04	
Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.06	

This chapter contains links to the different debug features used in the E-GOLDradio:

- [Section 11.9.2 Break Switch \(on Page 1244\)](#)

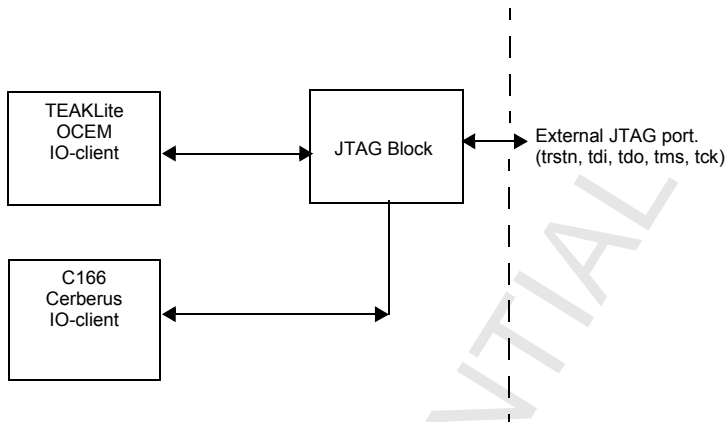
**Figure 15-1 Break Switch Interface**



**CONFIDENTIAL**

- [Section 11.9.3 JTAG \(on Page 1248\)](#)

**Figure 15-2 JTAG IO Mode Application Example**



- [Section 6.8 OCDS \(on Page 272\)](#)
- [Section 8.15 OCEM/SEIB \(on Page 595\)](#)
- [Section 6.9 Cerberus \(on Page 291\)](#)
- [Section 11.8.10 Internal Signal Monitoring \(on Page 1209\)](#)
- [Section 5.7 DSP Debug Register: DSP\\_DEBUG \(on Page 79\)](#)
- [Section 5.8 Pad Access Register: DSP\\_DSPOUT \(on Page 81\).](#)

## 16 Software Boot Code

History	
Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.04	
<b>Page 1418</b>	Updated <b>Table 15-1 Addresses of the Routines (in hex)</b> WS00008426
Changes for Rev. 1.05	
<b>Page 1418</b>	Updated <b>Table 15-1 Addresses of the Routines (in hex)</b> WS00008767
Changes for Rev. 1.06	

*Note: Additional information about the boot code is available on request from the E-GOLDradio support team.*

### 16.1 Functionality

This chapter describes the MCU Bootcode.

The bootcode is the first code to be executed after restart of an embedded system. The MCU bootcode for E-GOLDradio performs the following main tasks:

- Basic system initialization
- Reset source detection
- Code download via serial interface (ASC0)
- Bootstrap of application programs
- Utility routines for faster code execution

#### 16.1.1 Features

- Code download via serial interface ASC0
  - data transmission/reception rate: 9.6, 19.2, 28.8, 38.4, 57.6 and 115.2 kBits/s
  - asynchronous 8 bit data, no parity, no flow support
- Fast jump to code execution from external memory
- Flash device access
  - Address mode detection for standard flash types: 8-bit non-muxed NOR, 16-bit non-muxed NOR, 8-bit non-muxed NAND and 16-bit non-muxed NAND

- EBU pre-configuration

## 16.1.2 Utility Routines Interface

The following routines are available in the LM-ROM in addition to the boot code. These routines have been selected among the more time critical routines used by the protocol stack. Using these routines rather than code in Flash memory can increase significantly the C166 bandwidth as these are accessed without wait state.

### 16.1.2.1 idle\_loop

The following assembly code is extracted from the protocol stack.

*Note: “ZEROS” refers to the **ZEROS (on Page 185)** register in the bit addressable SFR area.*

```

ATOMIC #03h          ; Avoid to answer to interrupts starting from
                    ; now, to restore bus after idle

BCLR IEN
MOV R4, ZEROS        ; Make a dummy read from external address
                    ; #000000h to place address low

EXTS #00,#1
MOV R4,[R4]
MOV EBU_PDC, #0100h ; Attach Pull Down Weak on address and data bus
                    ; to avoid floating

IDLE                 ; The C166 pos1/pos2/neg clock are stopped
                    ; completely (0 Hz)

MOV EBU_PDC, ZEROS   ; Remove Pull Down Weak on address and data bus
BSET IEN             ; Allow to answer the awakening interrupt from
                    ; 2nd instruction after

```

### 16.1.2.2 memcpy

void \* memcpy(void \*destination, const void \*source, unsigned int size);

Copies data from source memory to destination memory *without* checking for overlapping spaces.

### 16.1.2.3 memcmp

int memcmp( const void \*cs, const void \*ct, size\_t n );

Compares the first n bytes of cs with the contents of ct.

Returns:

< 0 if cs < ct,

0 if cs == ct,

> 0 if cs > ct.

**CONFIDENTIAL**

**Functionality**

#### **16.1.2.4 memmove**

```
void *memmove( void *s, const void *ct, size_t n );
```

Copies n characters from ct to s. Overlapping objects are handled correctly.

Returns: s

#### **16.1.2.5 memset**

```
void *memset( void *location, int value, unsigned int size);
```

Fills a block of memory of a given size at a given location with a given value.

#### **16.1.2.6 strcat**

```
char *strcat( char *s, const char *c );
```

Concatenates string c to string s, including the trailing NULL character.

Returns: s

#### **16.1.2.7 strchr**

```
char *strchr( const char *cs, int c );
```

Returns a pointer to the first occurrence of character c in the string cs. If not found, NULL is returned.

#### **16.1.2.8 strcmp**

```
int strcmp( const char *s1, const char *s2 );
```

Compares string s1 to string s2.

Returns:

<0 if s1 < s2,

0 if s1 == s2,

>0 if s1 > s2.

#### **16.1.2.9 strcpy**

```
char *strcpy( char *s, const char *ct );
```

Copies string ct into the string s, including the trailing NULL character.

Returns: s

**CONFIDENTIAL**

**Functionality**

### 16.1.2.10 strlen

```
size_t strlen( const char *cs );
```

Returns:

The length of the string in cs, not counting the NULL character.

### 16.1.2.11 Bootcode Routine Address Table

**Table 15-1**    **Addresses of the Routines (in hex)**

<b>Routine</b>	<b>For Bootcode &lt; V2.0</b>	<b>For Bootcode V2.0</b>	<b>For Bootcode V2.2</b>
<a href="#">idle_loop</a>	06DC	06DE	N/A
<a href="#">memcpy</a>	06F6	06F8	06C6
<a href="#">memcmp</a>	05A2	05A4	05A4
<a href="#">memset</a>	N/A	N/A	078A
<a href="#">memmove</a>	05D0	05D2	05D2
<a href="#">strcat</a>	0624	0626	0626
<a href="#">strchr</a>	0660	0662	0662
<a href="#">strcmp</a>	067C	067E	067E
<a href="#">strcpy</a>	06A4	06A6	06A6
<a href="#">strlen</a>	06C4	06C6	N/A

**CONFIDENTIAL**

CONFIDENTIAL

**CONFIDENTIAL**

CONFIDENTIAL



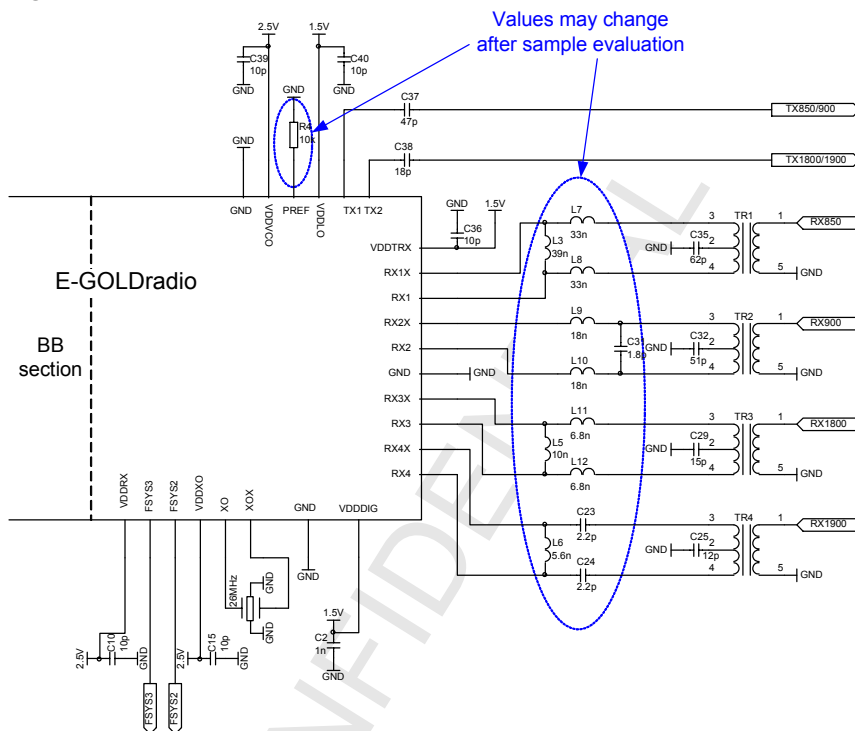
CONFIDENTIAL

## 17 Application

History	
Current version: Rev. 1.06, 2005-11-04	
Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on SMARTi-SD2 PMB6271 Target Specification, Rev 2.03
Changes for Rev. 1.02	
<a href="#">Page 1422</a>	Updated VDD names in <a href="#">Figure 17-1 Application Circuit</a> WS00007484
Changes for Rev. 1.05	

## 17.1 Circuits

### Figure 17-1 Application Circuit



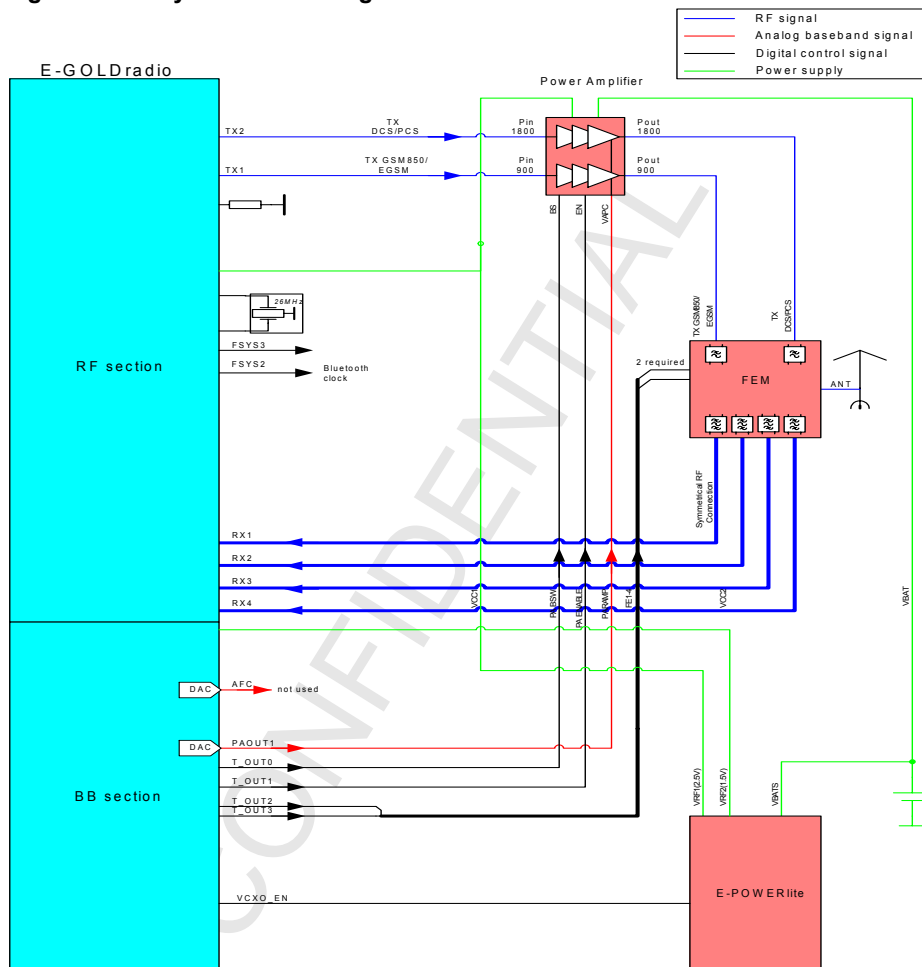
*Note: Component values and topology for RX Matching, blocking capacitors and RREF Resistor if required are subject to be optimized after first evaluation of GSM/GPRS Baseband System.*

**CONFIDENTIAL**

**System Block Diagram**

## 17.2 System Block Diagram

**Figure 17-2 System Block Diagram**

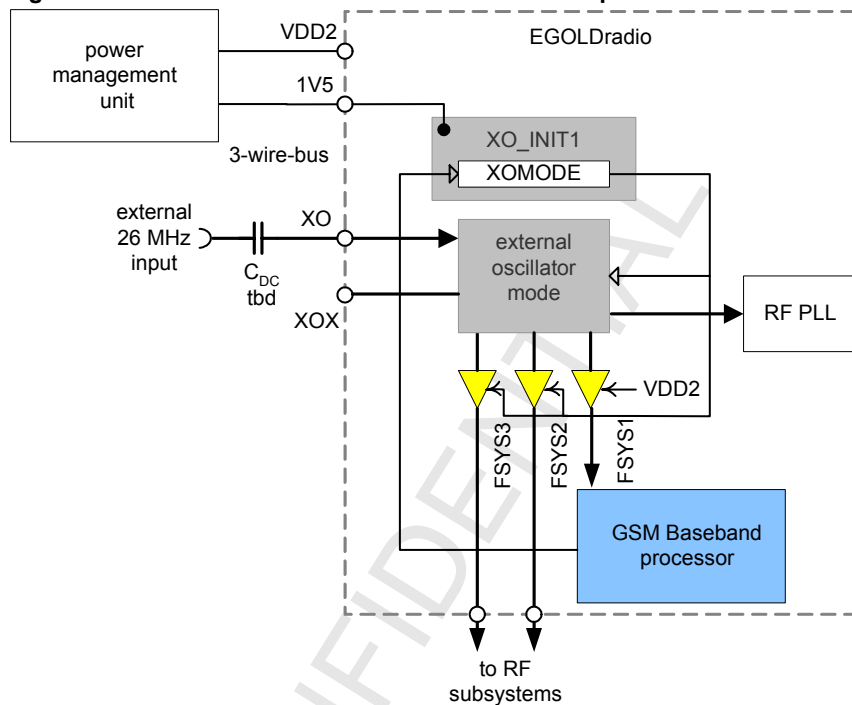


CONFIDENTIAL

External 26 MHz Clock Application Circuit

### 17.3 External 26 MHz Clock Application Circuit

Figure 17-3 External 26 MHz Module and FSYS Outputs



**CONFIDENTIAL**

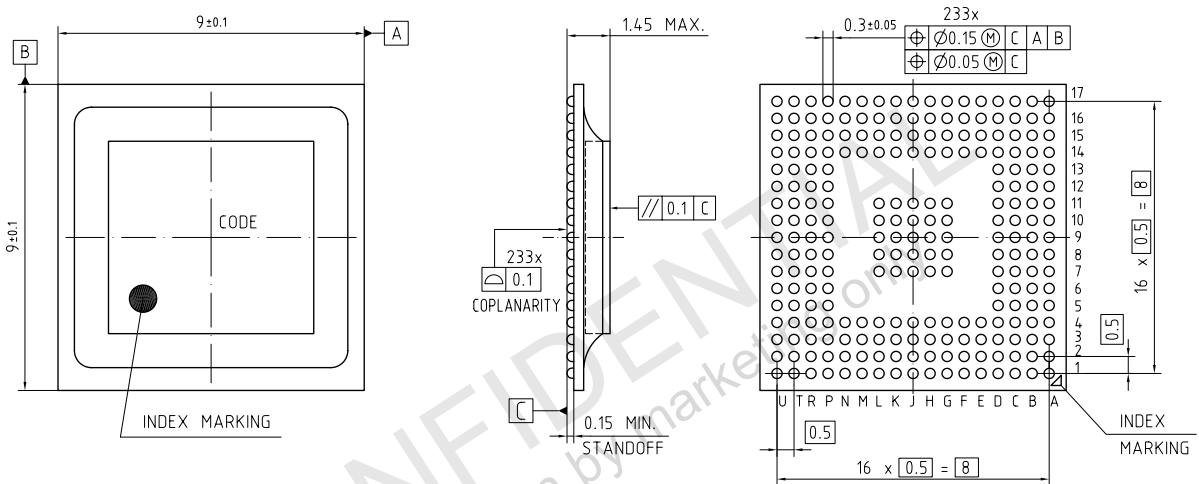
## 18 Package Outline

History	
Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.00	
<b>Page 1756</b>	Updated <b>Figure 19-1 E-GOLDradio Package Overview</b>
<b>Page 1804</b>	Updated <b>Figure 19-1 Package Labels (Character height = 1.0 mm)</b>
Changes for Rev. 1.01	
<b>Page 1762</b>	Updated <b>Figure 19-1 E-GOLDradio Package Overview</b> WS00006846
Changes for Rev. 1.02	
<b>Page 1426</b>	Added <b>Figure 18-1 E-GOLDradio: LF2BGA-233 Package Details and Ball Labelling</b> WS00007256
<b>Page 1804</b>	Updated Legend for <b>Figure 19-1 Package Labels (Character height = 1.0 mm)</b> WS00007493
Changes for Rev. 1.04	
<b>Page 1426</b>	Removed figure "E-GOLDradio Package Overview", it is the same as <b>Figure 18-1 E-GOLDradio: LF2BGA-233 Package Details and Ball Labelling</b> WS00007256
Changes for Rev. 1.06	
<b>Page 1426</b>	Updated <b>Figure 18-1 E-GOLDradio: LF2BGA-233 Package Details and Ball Labelling</b>
<b>Page 1425</b>	Replaced old <b>Figure 19-1 Package Label (Character height = 1.0 mm)</b> with a cross-reference WS00009111

For a description of the package label see **Figure 1-2 Package Label (Character height = 1.0 mm) (on Page 40)**.

CONFIDENTIAL

**Figure 18-1 E-GOLDradio: LF2BGA-233 Package Details and Ball Labelling**



**CONFIDENTIAL**

CONFIDENTIAL

**CONFIDENTIAL**

CONFIDENTIAL



**CONFIDENTIAL**

## Appendix A Glossary

History	
Current version: Rev. 1.06, 2005-11-04 Previous version: Rev. 1.05, 2005-08-02	
Page	Subjects (major changes since last revision)
	Initial revision based on E-GOLDlite Design Specification, Revision 2.03
Changes for Rev. 1.06	

**Table A-1 Glossary of Terms**

Term	Description
4PPM	Four Pulse Position Modulation
AFC	Adaptive Frequency Correction Automatic Frequency Correction
ALU	Arithmetic Logic Unit
AMR	Adaptive Multi-Rate
ASC	Asynchronous Serial Interface Controller
BFO	Bit Field Operations
CAPCOM	Capture Compare
CGU	Clock Generation Unit
CPS	CPU Slave
CPU	Control Processing Unit
CRC	Cyclic Redundancy Check
DAC	Digital Analog Converter
DAI	Digital Audio Interface
DSP	Digital Signal Processor
DTX	Discontinues Transmission
EBU	External Bus Interface Unit
EDGE	Enhanced Data Rates for GSM Evolution
EFR	Enhanced Full-Rate
EGPRS	Enhanced GPRS
FCS	Frame Check Sequence

**CONFIDENTIAL**

**Table A-1 Glossary of Terms**

<b>Term</b>	<b>Description</b>
FIR	Fast Infrared Mode
FR	Full Rate
GEA	GPRS Encryption Algorithm
GMSK	Gauss Minimum Shift Keying
GPIO	General Purpose IO cell
GPRS	General Packet Radio Service
GPTU	General Purpose Timer Unit
GSM	Global System for Mobile communication
HR	Half Rate
HSCSD	High Speed Circuit Switched Data
I2C	Inter IC
I2S	Inter IC Sound
IDMX	Input demultiplex
IrDA	Infrared Data Association
IrLAP	Infrared Link Access Protocol
ISA	Instruction Set Architecture
JTAG	Joint Test Action Group
LLC	Logical Link Control
LMB	Local Memory Bus
LNA	Low Noise Amplifier
MAC	Multiply Accumulate Module
MCS	Modulation and Coding Scheme
MCU	Micro Controller Unit
MIR	Medium Infrared Mode
MMU	Memory Management Unit
MPU	Micro Processor Unit
TEAKLite	DSP Core
OCDS	On Chip Debug Support
OCEM	On Chip Emulation Mode
PCL	Port Control Logic

**CONFIDENTIAL**

**Table A-1      Glossary of Terms**

<b>Term</b>	<b>Description</b>
PCP	Peripheral Control Processor
PEC	Peripheral Event Controller
PSK	Phase Shift Keying
RTC	Real Time Clock
RTOS	Real-Time Operating System
RZI	Return to Zero Inverted
SAPP	Sound APPLication
SCCU	Standby Clock Control Unit
SCU	System Control Unit
SEIB	System Emulation In Board
SIM	Subscriber Identity Module
SIR	Serial Infrared Mode
SPI	Serial Peripheral Interface
SRR	Service Request
SSC	Serial Synchronous Interface Controller
STM	System Timer Module
TAP	Test Access Port
TCL	Half of a MCU clock period
TDMA	Time Division Multiple Access
TOS	Type of Service
UART	Universal Asynchronous Receiver Transmitter
USART	Universal Synchronous Asynchronous Receiver Transmitter
USB	Universal Serial Bus
USF	Uplink Status Flag
VAD	Voice Activity Detection
XAB	X Address Bus
YAB	Y Address Bus
ZAB	Z Address Bus
XDB	X Data Bus
YDB	Y Data Bus

**CONFIDENTIAL**

**Table A-1      Glossary of Terms**

<b>Term</b>	<b>Description</b>
ZDB	Z Data Bus
PAB	Program Address Bus
PDB	Program Data Bus

CONFIDENTIAL

**CONFIDENTIAL**

CONFIDENTIAL

**CONFIDENTIAL**

CONFIDENTIAL