BP30
LED driver Specification

Edition 2006

Published by Neonseven s.r.l.,
Viale Stazione di Prosecco, 15

34010 Sgonico (Trieste) Italy

For questions on technology, delivery and prices please contact the Neonseven Offices in Italy Sgonico and Gorizia

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.

Warnings

Due to technical requirements components may contain dangerous substances. For information on the types in question please contact Neonseven.

Neonseven technologies may only be used in life-support devices or systems with the express written approval of Neonseven, if a failure of such technologies 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.

Table of Contents

1 Document Mission/Scope

1.1 Mission

This document contains the specification of the Led driver used in BP30 project. Such driver handles single and multicolor LED, it manages more than one led or groups of led and changes color and intensity of each led.

1.2 Scope

This document is addressed to SW developers who need to either to interface to or customize the LED driver module.

2 List of Acronyms

Abbreviation / Term	Explanation / Definition
CAPCOM	Capture Compare
GPIO	General Purpose Input Output
MMI	Man-Machine Interrupt

3 Introduction

The aim of the led driver is to handle service, keypad and display led. Service LED is used to reflect state of the mobile, or to indicate when charger is plugged in. To this purpose, different color or combination of color could be used when mobile changes its state from an idle state to receiving a call.
Keypad led consists of four multicolor led that are switched on when a key is pressed and switched off by a direct call from MMI or by led driver if fading action is set. Different actions, colors and intensity levels can be applied if supported by HW.

4 LED

In the following chapter, led architecture and led HW design developed on BP30 board will be described.

4.1 Architecture

The architecture of the led driver is illustrated in Figure 4-1. MMI has to provide the LED driver with information of the request action. The main function is LED_main_control that schedule all activity, programs HW and SW timers, activates CAPCOM interrupts.

The actions handled by LED driver are:

1. To turn on/off led
2. To flash with the specified duty cycle
3. to choose a sequence of 10 colors, and repeat it until led is switched off by MMI
4. To apply fading on LED_ON and LED_OFF actions, with an initial intensity, color and duration.

To perform the above cited action, LED driver uses the following HW and SW timers:

- Capcom HW Timers
- Paging period
- Tic timers

depends on kind of led handled and period of action as it will be explained in section 4.2.

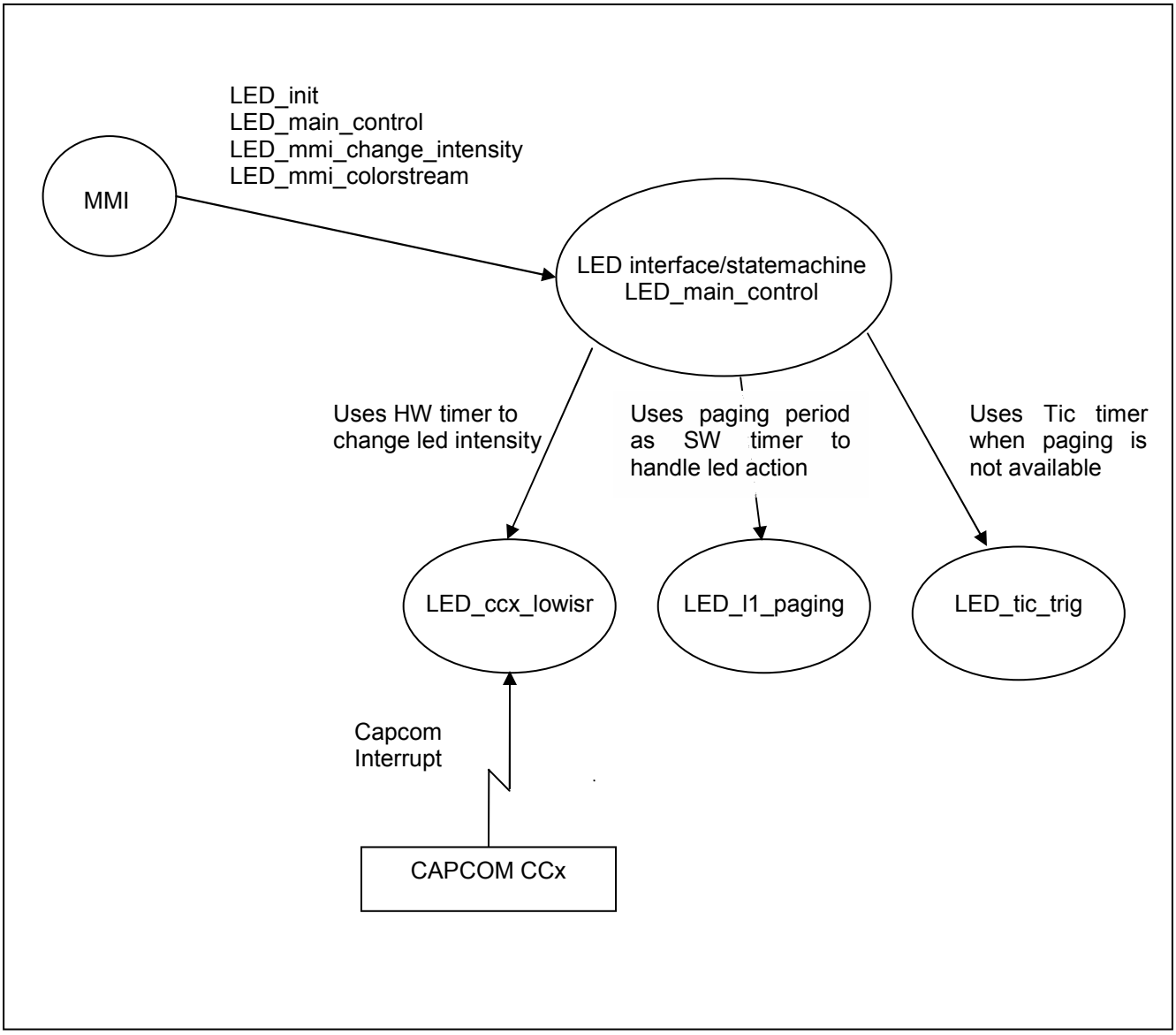

Figure 4-1 – LED architecture

The LED events are classified as continues events or single events. The timing in the continuous events must kept a subset of the minimum timing of 1.5 seconds (e.g. 3*1,5s = 4,5s OFF time) in order to save power.

Different LED events are proposed in Table 4-1

Single events	**FLASH**	**ON**	**OFF**	**Duration**
Power ON	x	40ms	250ms	6 sec. OR until power on complete
Power OFF	X	40ms	250ms	6 sec. OR until mobile is powered down
Receiving phone call		3 sec.		3 sec.
Receiving phone call connected		3 sec.		3 sec.
Receiving SMS	x	40ms	250ms	6 sec. Repeat when key is pressed
Making Call		2 sec.		2 sec.
Sending SMS		2 sec.		2 sec.
SMS full	x	40 ms		6 sec. Repeat when key is pressed
Charger events	**FLASH**	**ON**	**OFF**	**Duration**
Charged connected	x	40ms	1,5 sec	Until battery is recharged, charger disconnected on mobile powered up.
Charger disconnected			X	
Charging completed		x		Until charger is disconnected, or mobile is powered up.
Continuous events	**FLASH**	**ON**	**OFF**	**Duration**
In Service	x	40ms	6 sec	Periodic
Out of Service	x	40ms	6 sec	Periodic
Limited Service	x	40ms	6 sec	Periodic
Alarm		x		
Battery low			x	
Missed call	x	40ms	6 sec	Periodic

Table 4-1: LED operation

4.1.1 Timing

4.1.1.1 In Service

When the mobile is in service, the LED drivers uses the paging ticks. This way the driver is only active, when the mobile is awake anyway. The required LED ON time of 40 ms has been measured as the average time of the paging actions. The smallest timing interval will be between 1,1-2,1 seconds in total_time, because the LED trig event is aligned with the paging in order to save power. The paging ticks give the following limitations to the choices of LED OFF time (Table 4-1).

It is recommended to use a 6 seconds period when LED is in service.

DRX	Period	Nof periods closest to 1,5 sec
2	470,73 ms	3 (1,4 sec.)
3	706,10 ms	2 (1,4 sec.)
4	941,46 ms	2 (1,9 sec.)
5	1,177 sec	1
6	1,412 sec	1
7	1,648 sec	1
8	1,883 sec	1
9	2,118 sec	1

Table 4-2: Possible paging distances in GSM (network specific parameters).

4.1.1.2 Out of Service, Limited Service or in call

When the mobile is out of service, in limited service on in call, the LED driver must still be provided with a tick every 6th second, if the service led is flashing during these events. With regards to power consumption, it is not recommended to use less than a 6 second timing when the mobile is out of service or in limited service. A library containing high level timers [tic.c/tic.h] – provides high-level timers that allows power saving while in use – will be used to start a trigger with the required timeout. When the time has elapsed, a function will be called in the LED driver. The high level timer will only be started if no paging tick is available.

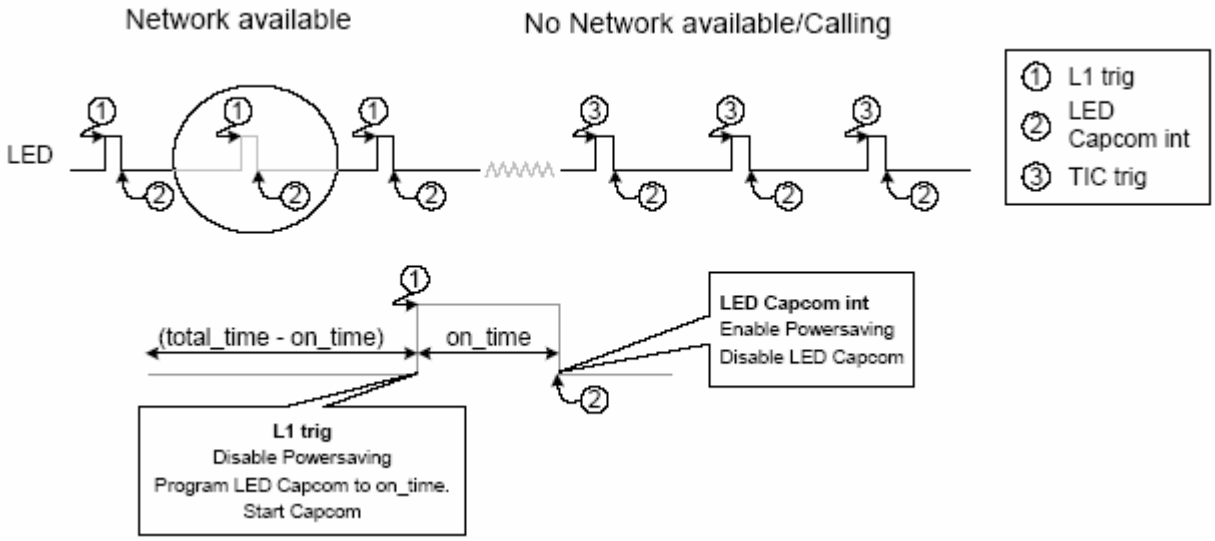

Figure 4-2: The concept of triggers and power saving

4.2 Functional description

In BP30 GLOBE board, the LED driver handles two different groups of led: service led and Keypad led.

4.2.1 Signaling Led

Signaling led consist of a red and a green led that are switched on/off by setting the bits SLED1ON and SLED2On in register LEDCTRL2 of EPOWERlite driver as is shown in Figure 4-3.
The main applications of signaling led are:

1) Show charger events:
- connected,
- disconnected,
- battery low,
- charging completed

2) Show MS status:
- In service,
- out of service,
- Limited Service
- in Call

The main characteristics are:

1) Led outputs are switchable low side constant current sources, therefore is not possible to change the intensity levels of led.

2) The actions ON, OFF, FLASH, COLORSTREAM and FADING are supported by using a SW timer (paging or Tic timer) for OFF timer and using a CAPCOM timer programmed in compare mode 2 (is an interrupt-only mode; only one interrupt request per timer period is generated) for ON time. In the following section there is a description of sequences made to achieve the flash action with very low power consumption.

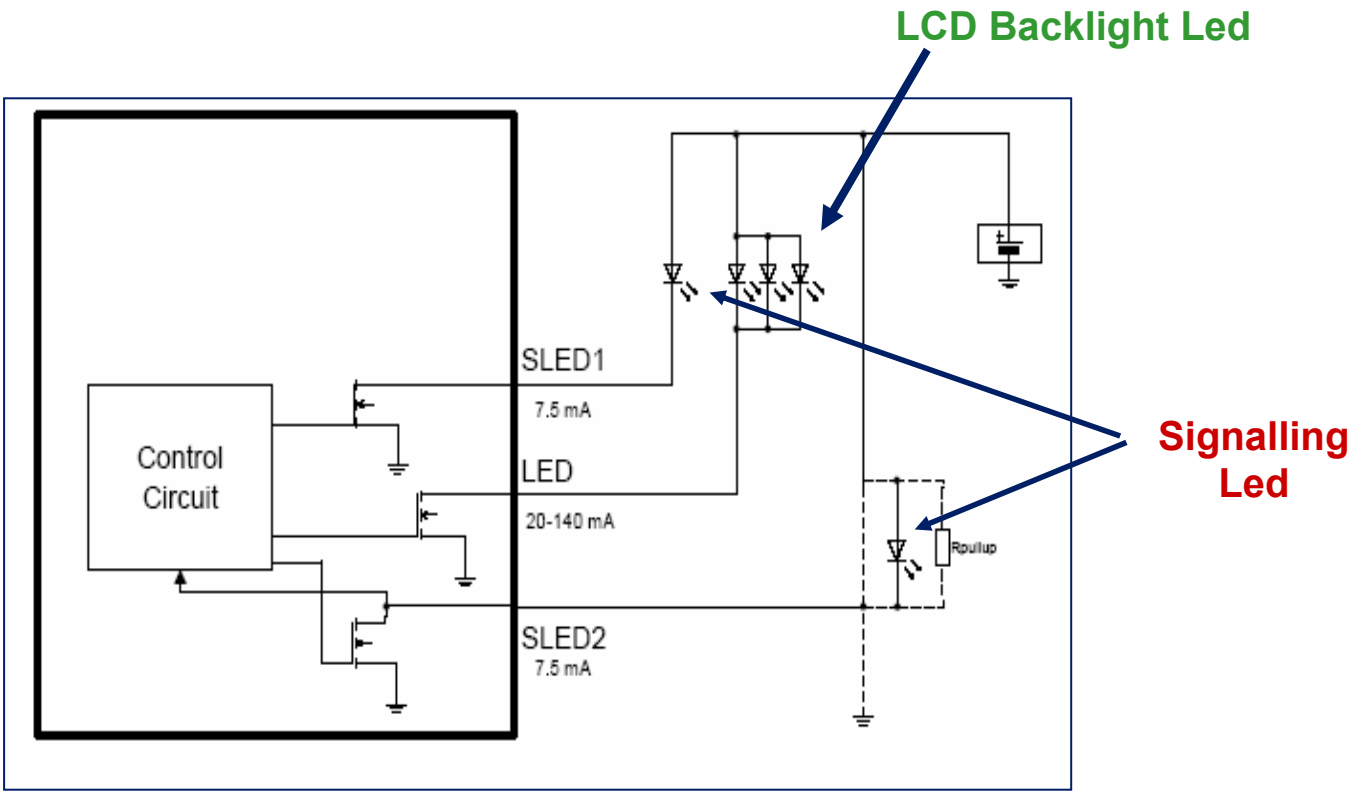

Figure 4-3: Signalling Led controlled by EPOWERlite driver

4.2.1.1 Example of flash action

If the LED is required to flash with a certain color and interval, and T_{ON} and T_{OFF} are the time for led on and off with T_{OFF} >>$T_{ON,}$ the following operations will be performed:

1) When n* paging-period >= T_{OFF} (or when Tic timer elapse):
   - n is reset
   - Led is turned on
   - Capcom is programmed in compare mode 2 in order to get an interrupt event on T_{ON} time.
   - Power saving is disabled to get a stable HW timer at 13MHz.

2) When the capcom interrupt occurs:
   - Led is turned off
   - Capcom is disabled
   - Power saving is enabled.

With regards to power consumption, the ON period should be equal or less than 40 ms, which is the time for the paging to complete and OFF period should be a multiple of paging period. The flow of the function calls is shown in Figure 4-4

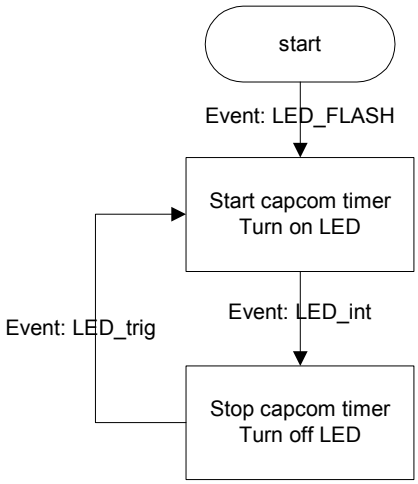

Figure 4-4: Flashing the LEDs

4.2.2 Keypad Backlight LED

Keypad Led consists of four multicolor led. Each led contains three color led red-green-blue programmed with a RGB coding (8 bit for each color).
To change the intensity level of the multicolor led, three different CCy capcoms (one for each color) are used. Those capcoms are configured in compare mode 3 to achieve only one compare event per timer period. The capcoms depend on a continuous timer started at initialization, and afterwards left untouched, this way providing a stable timer for other drivers to use. When the first match within the timer period is detected, the interrupt request flag CCyIR is set and the output pin CCyIO is set. The signal is cleared when the allocated timer overflows.
In Figure 4-5 and Figure 4-6 is shown the block diagram and timing example of capcoms in compare mode 3.

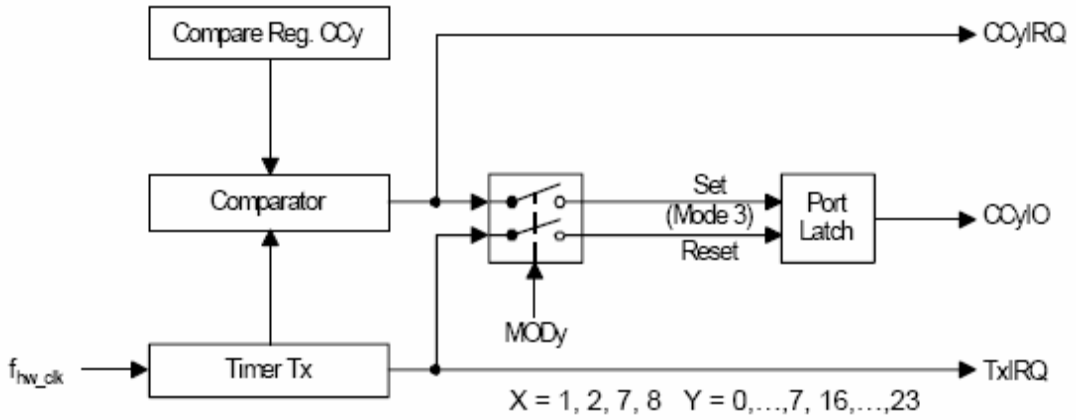

Figure 4-5: Compare mode 3 block diagram

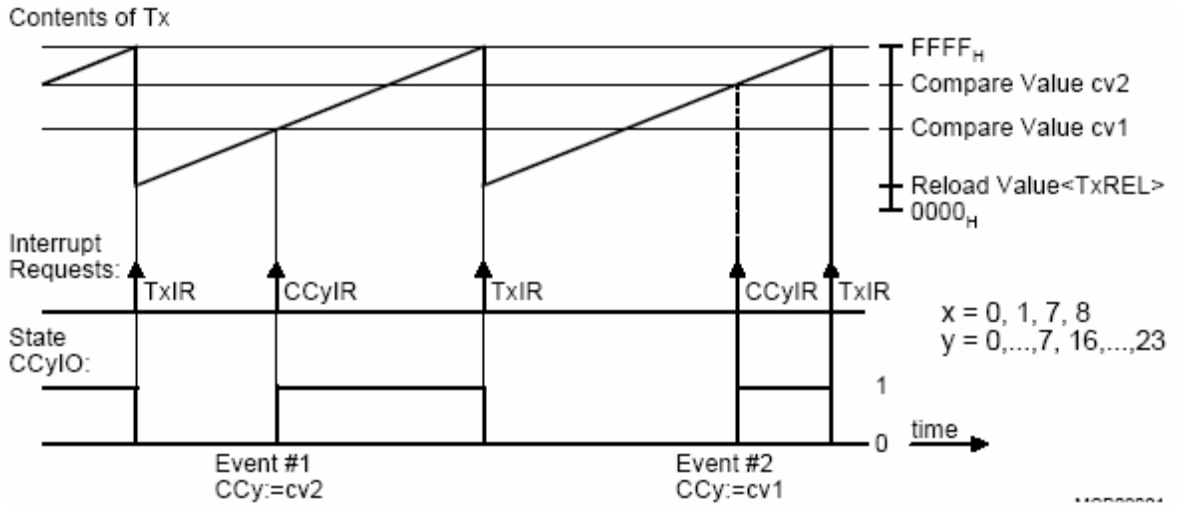

Figure 4-6: Timing example of compare mode 3

The output signal CCyIO is used to modulate the base current of BJT to switch transistor from interdiction to saturation state and to achieve different intensity levels as illustrated in Figure 9-1.
To change the intensity level for each color of the multicolor led, a good approximation is to consider that LED intensity is linearly proportional to time intervals during which the led is on, so the intensity can be increased or decreased by adjusting the duty cycle D of the voltage supplied, defined as

$$D = \frac{t_{ON}}{t_{ON} + t_{OFF}} = \frac{t_{ON}}{Ts} = fs\ t_{ON}$$

where t_{ON} and t_{OFF} are the time interval during which the LED is on and off; Ts = t_{ON} + t_{OFF} is the duration of a cycle.
This pulse width modulation is generated using capcom in compare mode 3, in fact in this mode, 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 (see Figure 4-6). The following section describes an example of capcoms configuration to handle the keypad led.

4.2.2.1 Example of Capcoms configuration

In order to change multicolor led intensity the following registers shall be configured:

- Configure CCMz to allocate capcom CCy to the respective timer Tx and to set CCy in compare mode 3
- Set TxREL = FF00H to get 255 different intensity levels (Timer Tx overflows from FF00H to 0000H)
- Set the period P_{Tx} between two consecutive overflows of Tx greaten than 5ms (1/200Hz), to avoid flickering.

$$P_{Tx} = \frac{(2^{16} - FF00H) \cdot 2^{(<TxL>+3)}}{fhw\_clk}$$

- Set CCyIO as output
- Change Led intensity by setting in CCy register a value in range FF00H-FFFFH.

In BP30 GLOBE6 board the output CC2IO, CC5IO and CC6IO are used for blue, green and red led respectively, therefore the capcoms configuration shall be:

- Set CCM0.MOD2 = 111, CCM1.MOD1 = 111, CCM1.MOD2 = 111 to configure CC2,CC5 and CC6 in compare mode 3
- Set CCM0.ACC0 = 1, CCM1.ACC1= 1, CCM1.ACC2 = 1 to allocate CC2,CC5 and CC6 to timer T1
- Set T1REL= FF00H to get 255 different intensity levels.
- Set T1M = 0 to configure timer T1 in Timer Mode (input derived from internal clock)
- Set T1L = 6 to get PTx > 5ms (with fhw_clk = 13MHz)
- Set CC2IO, CC5IO and CC6IO as output

To get the maximum intensity level the registers must be set as
CC2 = CC5= CC6= T1REL+1,
instead the minimum is obtained by setting
CC2 = CC5= CC6= T1REL-1

4.2.2.2 Action on Keypad Led

All actions ON,OFF, FLASH, COLORSTREAM and FADING can be applied to keypad led using a SW and capcom timer as already described in section 3 for signaling led.

5 Interface specification

5.1 Initialisation

The LED is initialised from Application_Initialize with a call to LED_init(). This function initialises the driver and the HW

Void **LED_init(**void**)**
Called from Application_Initialize on start up. Initialises the driver and the HW.

5.2 MMI Interface

The MMI is in charge of all LED operation. If a LED is started, it must be stopped from the MMI. The only exception is fading action, in fact led is switched on by MMI and switched off by led driver when capcom timer elapse.
Led driver manages more than one led or groups of led, in particular is possible to apply different action to signaling and keypad led. All actions are performed by calling the LED_main_control function.

5.2.1 LED_main_control

BOOL **LED_main_control (**led_color_type **color,** ubyte **action,** ushort **total_time,** ushort **on_time, led_type_of_peripherals led_peripherals)**		
Led_color_type led	Described in the enum led_types in	E.g. red, blue, green or combinations

	led.h	
Ubyte action	Defined in led.h	LED_OFF, LED_ON, LED_FLASH, LED_COLORSTREAM, LED_FADING (if supported by HW)
Ushort total_time	Led_on [ms] + led_off [ms]	Only used if action is LED_FLASH and LED_FADING Range [250; 10000]
Ushort on_time	Led_on [ms]	Only used if action is LED_FLASH and LED_FADING If action is LED_ON, it is possible to turn on a led just once, for a period of led_on [ms]. Range [40; 1499], on_time < off_time.
led_type_of_peripherals led_peripherals	Defined in led.h	led_peripherals parameter is HW dependend. It is used to apply the action onto specific group of led. Possible values are led_service, led_kbd, led_display, led_all_peripherals.
Returns BOOL	TRUE: the parameters in the function call were inside the specified limits FALSE: The parameters were outside limit and have been truncated.	

LED_main_control is called with the chosen led color (colors are described in the enum led_types in led.h). The action can be any of LED_OFF, LED_ON, LED_FLASH, LED_COLORSTREAM or LED_FADING (defined in led.h).

If the requested action is LED FLASH, the chosen LED will flash with a LED_ON time of **on_time** and a LED_OFF time as (**total_time** – **on_time**). It will continue to flash until stopped with a new call to LED_main_control, with the action LED_OFF.

If the requested action is LED_COLORSTREAM, then a combination of up to 10 colors (e.g. led_green, led_red_green, led_no_color, led_green, led_red_green,..) can be flashed. First, the colorstream should be updated using the LED_mmi_colorstream() with the necessary colors as well as the on_time and total_time associated with each before calling LED_main_control. The arguments color, total_time and on_time are ignored for the LED_COLORSTREAM action.

If the leds are to be turned on only once for a specific period of time, this can be controlled by specifying action as LED_ON and the period in the **on_time** parameter.

If the requested action is LED_FADING, the chosen led is switched on with an initial led intensity set by LED_mmi_change_intensity function, and then it will be switched off after a total_time second.
During this period the led intensity is decreased on every on_time by a step calculated as:

$$\text{StepFading} = \frac{\text{InitialLedIntensity}}{\frac{\text{total\_time}}{\text{on\_time}}}$$

For example, if the initial led intensity is 200, on_time = 100ms and total_time = 5s, the led intensity will be decrease by 4 on every 100ms. After 5s led intensity will be zero and the led will be completely switched off.

5.2.1.1 Examples of LED actions

5.2.1.1.1 In service area

Flash service led with green color::
Green flash, on_time = 40 ms, off_time = 6 sec.
LED_main_control(led_green, LED_FLASH, 6000, 40,led_service);

Change to Limited Service:
Stop green led*:*
LED_main_control(led_green, LED_OFF, 0, 0, led_service);

Start service led with red color:
Change LED to red flash, on_time = 40 ms, off_time = 6 sec.
LED_main_control(led_red, LED_FLASH, 6000, 40,led_service);

The LED driver will use a timer provided by the TIC library, while the mobile is out of service. (No paging ticks available)

5.2.1.1.2 On Keypad led

Switch on keypad led with blue color:
LED_main_control(led_blue, LED_ON, 0, 0,led_kbd);

Switch on keypad led with yellow color for 100ms
LED_main_control(led_red_blue, LED_ON, 100, 100, led_kbd);

Apply fading action to blue color with an initial intensity equals 255, total_time =4s, on_time = 40ms
Led_mmi_change_intensity(led_blue,255,led_kbd);
Led_main_control(led_blue,LED_FADING,4000,40,led_kbd)

5.2.1.1.3 Using the COLORSTREAM functionality:

Set up the number and order of colors and the time periods for each:
E.g. for streaming 3 colors, with different time periods, update the led_stream_data structure as: -
led_stream_data_type led_stream_data[3];
led_stream_data[0].color = led_blue;
led_stream_data[0].total_time = 500;
led_stream_data[0].on_time = 200;
led_stream_data[1].color = led_green;
led_stream_data[1].total_time = 2000;
led_stream_data[1].on_time = 500;
led_stream_data[2].color = led_red;
led_stream_data[2].total_time = 0;
led_stream_data[2].on_time = 0;
LED_mmi_colorstream (3, led_stream_data);

Start color_stream:
LED_main_control(not used, LED_COLORSTREAM, not used, not used,led_kbd);.

5.2.2 LED_mmi_colorstream

void **LED_mmi_colorstream(**ushort **len**, led_stream_data_type ***led_stream_data)**		
Ushort len	Number of colors in colorstream	Max. length is 10
led_stream_data_type *led_stream_data	Defined in led.h	Store the color, on_time and total_time for each.

The MMI uses LED_mmi_colorstream() to initialise and/or change the order of the colors used when colorstreaming. A color_stream is a continuous change of led colors until stopped by the MMI. Each color has an on_time and total_time specified for itself. The data structure **led_stream_data** is used to hold this data. Only these values will be used to perform the color streaming and the color streaming always starts with the first color in the stream. When the on_time is 0 for any color then that color shall be streamed using the default period of 250 ms.(i.e. on_time = off_time = 250ms). The following is an example led_stream_data and the streaming shown in Figure 5-1

Index	Color	Total_time	On_time
0	Led_blue	500	200
1	Led_green	2000	500
2	Led_red	0	0
…	….	…	…
9(max)			

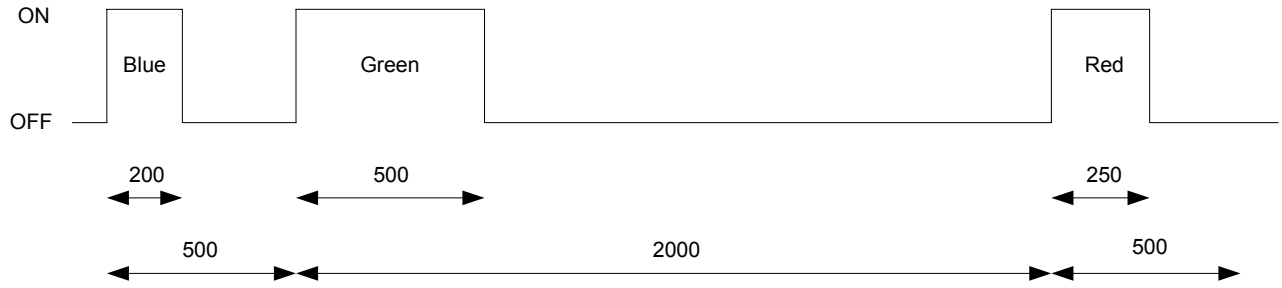

Figure 5-1: Led color streaming

5.2.3 LED_mmi_disable_all_leds

void **LED_mmi_disable_all_leds(**void**)**
Turns off all LEDs and all ongoing led functionality

This function is used to switch off all led and disable all SW and HW timers.

5.2.4 LED_mmi_set_eep_colors

BOOL **LED_mmi_set_eep_colors(**led_event_type **event**, led_conf_type ***config)**		
Led_event_type event	Defined in led.h	E.g. led_call, led_service…
Led_conf_type *config	color, on_time and total_time	The requested colour and timing to be saved in the eeprom.

The function returns TRUE if the parameters are saved successfully.

This function is used for saving user defined LED settings in the eeprom.

5.2.5 LED_mmi_get_eep_colors

led_conf_type *LED_mmi_get_eep_colors(led_event_type **event**)		
Led_event_type event	Defined in led.h	Get colors and timing related to this specific event
The function returns a pointer to the colour, on_time and off_time, the change parameter describes whether the specified event can be changed by the MMI. If the function returns a NULL pointer, the event is not stored in EEP.		

This function gets the stored LED parameters from the eeprom.

5.2.6 LED_mmi_change_intensity

Void LED_mmi_change_intensity(led_color_type color, ushort intensity,led_type_of_peripherals led_peripherals)		
Led_color_type color	Defined in led.h	The color that needs a change in intensity. Led_blue, led_red or led_green. Combinations are not possible.
Ushort intensity	Possible values can be found in the led_current_enum_type in led.h	The range for the intensity is HW dependent. Values will be truncated to maximum if called with a value outside scope.
led_type_of_peripherals led_peripherals	Defined in led.h	led_peripherals parameter is HW dependend. It is used to change the intensity of a specific group of led. Possible values are led_service, led_kbd, led_display, led_all_peripherals.

This function changes the intensity if supported by HW. If not supported, the settings will be ignored. Call the function before calling LED_main_control(). The intensity can be changed on the fly as well.

5.2.7 LED_mmi_set_charger_led

void **LED_trig**(ushort **page_mode)**	
Ushort page_mode	Page_mode can be any of 2-9 multiframes between each paging. Function is used to time the LED flashes when the mobile is in service. (See Table 4-1 and Table 4-2).

When the mobile is in service, the paging period can be used as SW timer to trig LED actions. To this purpose LED_trig function is called and the page_mode is used to calculate if the LED should be triggered.

5.3 Interface to L1

5.3.1 LED_trig

void **LED_trig**(ushort **page_mode)**	
Ushort page_mode	Page_mode can be any of 2-9 multiframes between each paging. Function is used to time the LED flashes when the mobile is in service. (See Table 4-1 and Table 4-2).

When the mobile is in service, the paging period can be used as SW timer to trig LED actions. To this purpose LED_trig function is called and the page_mode is used to calculate if the LED should be triggered.

5.4 Interface to timer library

5.4.1 LED_tic_trig

When the mobile is out of service, the timer library provides the LED trig event, which ensures the timing of the LED flashes.

void **LED_tic_trig(**void**)**
Called from tic.c with the requested period. Only used if the mobile is in limited service, Out of Service or during a call.

5.5 Interface to charger driver

5.5.1 LED_get_status

ushort **LED_get_status(**void**)**		
Returns	0x1	Blue led
	0x2	Red led
	0x4	Green led

This function returns the color(s) of the leds that are constantly ON.

6 References

6.1 External

[1] Infineon Tecnology AG, E-GOLDlite GSM/GPRS Baseband System manual, PMB 7860 Design Specification, Rev.2.01, 2004-04-15
[2] Jette Nordstrøm Kjældgaard, LED interface Specification, Rev. 0.21, 2004-17-02

6.2 Internal

None.

7 Document change report

	Change Reference		Record of changes made to previous released version	
Rev	**Date**	**CR**	**Section**	**Comment**
1.0	18/06/04	N.A	Document created	
1.1	24/06/04	N.A	§4.1, Annex1	Correction on Led architecture Introduced schematic used to control led intensity
1.2	15/12/05	N.A	Doc updated to BP30 platform	

8 Approval

Revision	Approver(s)	Date	Source/signature
1.0	Stefano Godeas	18/06/04	Document stored on server
1.1	Stefano Godeas	29/06/04	Document stored on server
1.2	Stefano Godeas	15/12/05	Document stored on server

9 Annex 1: Example of schematic for controlling Led intensity

In order to change the led intensity three CAPCOMS are used to generate the width pulse modulation that controls the base current of BJT transistor (see section 4.2.2).

The led intensity can be decrease or increased by adjusting the duty cycle D of the voltage supplied to LED defined as:

$$D = \frac{t_{ON}}{t_{ON} + t_{OFF}} = \frac{t_{ON}}{Ts} = fs\ t_{ON}$$

where t_{ON} is the time interval during which the LED is on.

Ts is the duration of the cycle defined as

$$Ts = \frac{fhw\_clk}{(2^{16} - TxREL)\ 2^{(<TxL>+3)}}$$

The maximum intensity is obtained by setting the capcom register to CCx = TxREL+1

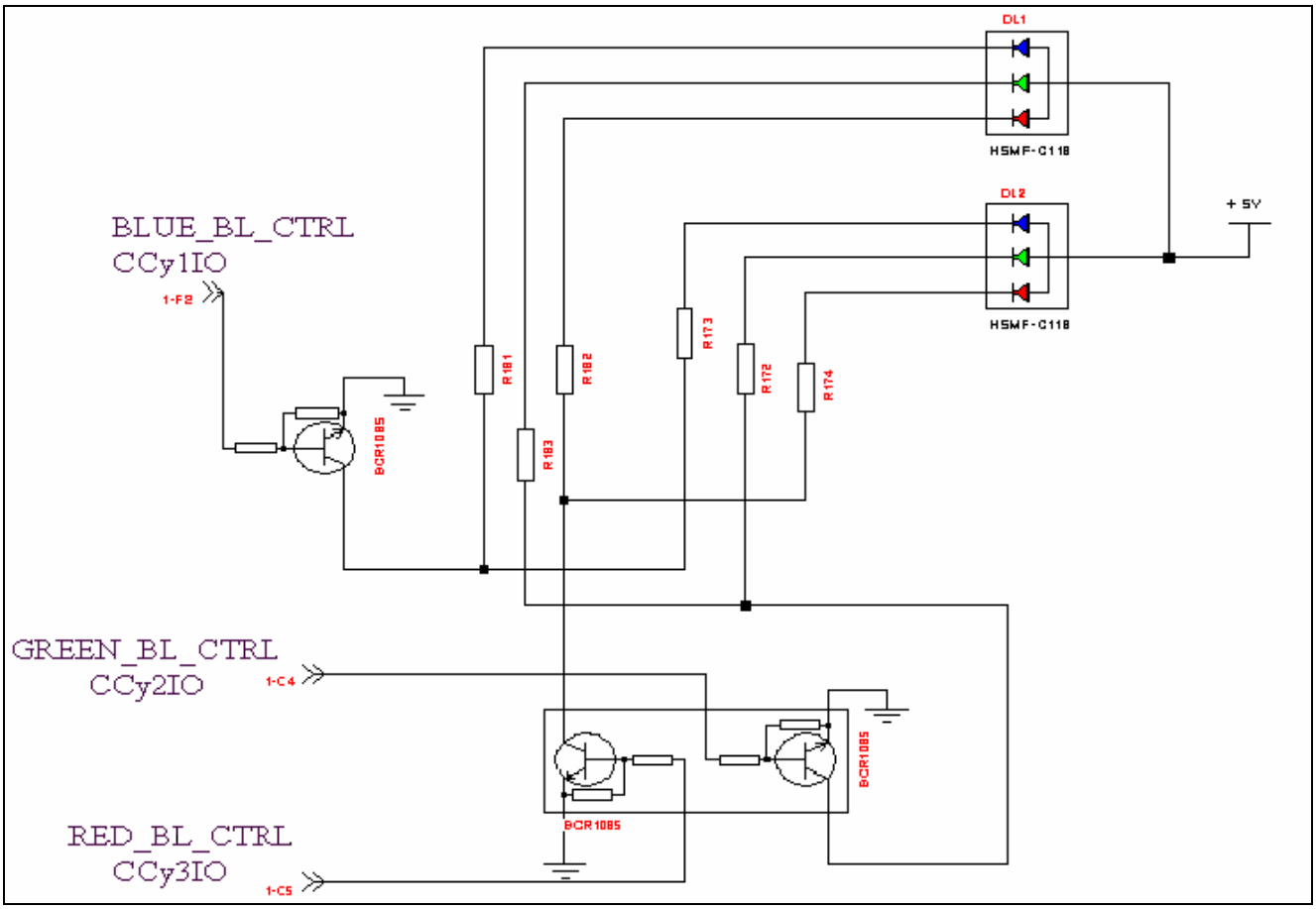

Figure 9-1*:* Method used to change Led intensity