Technical Note
N7 BP30 Build System

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 3

1.1 Mission 3

1.2 Scope 3

2 List of Acronyms 3

3 References 3

4 Introduction 3

5 NeonSeven Build System Environment 3

5.1 Build Procedure 3

5.1.1 Prepare to Work 3

5.2 Settings 4

5.3 Start to Work 4

5.3.1 Build Settings 4

5.3.2 Useful Commands 8

6 Open Issues 8

7 Add/Remove Files to/from Build System 8

8 Compiler Defines 9

8.1 Defines types 9

8.2 Where add/remove a define to/from 9

9 Compiler Include 9

10 Release to 3rd Parties 10

11 Document change report 11

12 Approval 11

Annex 1 12

Annex2 12

Annex 3 12

1 Document Mission/Scope

1.1 Mission

This document will be explaining the build system environment.

1.2 Scope

The scope of this document is only for the person that can have access to the development VOBs of IFX. The build system instructions for customers that have not access to those VOBs will be provided in Release Letter document.

2 List of Acronyms

Abbreviation / Term	Explanation / Definition
CBE	Common Build Environment

3 References

- [1] Comneon – CBE Common Build Environment ver1.3

4 Introduction

This document is user documentation for CBE (Common Build Environment) and explains how it can be used to build software: an overview is provided about initial actions to be done, settings and commands.

5 NeonSeven Build System Environment

5.1 Build Procedure

5.1.1 Prepare to Work

- Create a view, set the correct config spec in the view, start the view;
- If buildtools VOB is not present the following softwares will be installed:
  - Tasking C166 compiler v7.5 release 5 + cp166l.lib modified library;
  - ActivePerl v5.6.1 or subsequent
- Install Visual C++;

5.2 Settings

- If the path where the view is stored is too long, you could substitute it with a virtual drive, e.g.
  ```
  subst p: M:\n7_bp30_username_generic.dev
  ```
- Start a DOS shell in VOB system-build from the directory make (e.g. `p:\system-build\make`)
- Call the `VCVARS32.BAT` batch file in order to set the environment variables for host build (this files must be present in the Microsoft Visual Studio installation directory into the VC98\Bin sub path)
- Add perl directory, cbe windows directory (present in VOB tools, sub dir WIN32\bin) and compiler directory in `PATH` environment variable. Note that the order of path inclusion is important so you MUST add with this sequence. E.g.
  ```
  PATH= p:\buildtools\perl\bin;
        p:\tools\WIN32\bin;
        p:\buildtools\c166\bin;
        %PATH%
  ```
- Set the variable `C166INC` with compiler include directory: e.g.
  ```
  set C166INC=p:\buildtools\c166\include
  ```
- Set the variable `M166INC` with the compiler include directory: e.g.
  ```
  set M166INC=p:\buildtools\c166\include
  ```

5.3 Start to Work

The command line for compiling is: `make -r`.
This will start a build with settings stored in the `project.cfg.default` file: stored values are project specific. Once the build process has completed, the executable file `${PROJECTNAME}.hex` can be found in the folder `system-build/${TARGETSYSTEM}/${PLATFORM}`. Also the abs file `${PROJECTNAME}.abs` and the map file `${PROJECTNAME}.map` are present here.

5.3.1 Build Settings

The following table lists the settings used in project.cfg.default and on commandline: some setting can be changed on commandline in order to have a build for host, for example, or to have a build with debugging informations.

Setting	Supported values	Description
the following settings are usually used on the commandline		
TARGETSYSTEM	HW	The output of the build process will run on a hardware platform.
	ALIHOST	The output of the build process will run on a host platform.
	HOST	The output of the build process will run on a host platform. Only for pure Stack simulation.

BUILDMODE	RELEASE	By default the build mode is a non-debug build mode. It is also used when this field is left empty.
	DEBUG	Debugging information will be included in the objects and libraries.
CCCLIB	Y	Code reduction tool is used before assembling each file.
INT_STAGE	FSY	full system
	BSY	basic system (stack+apoxi+mmi)
	STT	just protocol stack
USESTACKLIBS	<empty>	stack has to be build
	Y, YES, any value	stack is provided as libs and therefore not build
USEAPOXILIBS	<empty>	apoxi has to be build
	Y, YES, any value	apoxi is provided as libs and therefore not build
DEPENDENCIES	NO	dependency files can be switched of because a) one is using clearmake b) one doesn't need it and want's to save hdd-space/time
VIAFILE	<empty>	by default all compiler/linker flags are handed over via commandline
	Y, YES, any value	compiler/linker flags are handed over via a file. Use this if you face the "line too long" problem
ASMCODEGEN	Y, YES, any value	several compilers allow to save intermediate assembler code for later review
ERRORLOG	<empty>	

	Y, YES, any value	compiler errors & warnings are put into an error file (located e.g. HW/SGOLD/err/<library>)
VERBOSE	<empty>	When the compiler commands are given all is done in quite mode.
	Y, YES, any value	Each compiler command is verbose (only for debug build system tool chain)
TMPDIR	<empty>	
	any valid path, e.g. c:/temp/cbe (win32) or ~/tmp/cbe (unix/linux)	the intermediate files (.o, .lib, .dep, …) are put into that directory. You may use this a) because you're using a dynamic view, don't want to winkin (clearmake) but want to speed up the build (by default the intermediate files go into the view in that scenario which slows down the build) b) you want to do several builds (e.g. debug & release) in one view. Using this setting can separate them.
the following settings are usually set in project.config.default and are not changed on commandline except in very special cases		
BUILDPCOS	WIN32	The PC on which the software is using a Windows NT or Windows 2000 operating system.
	LINUX	The PC on which the software is using a Linux operating system
PLATFORMOS	OSE166	OSE 16 bit operating system is utilized.

CORE	C166	C166 is used on the target platform.
CPU	EGOLDLITE	EGOLDLITE CPU is used on the target platform.
PLATFORM	BP30_GLOBE6	The term 'platform' relates to the platform on which the actual executable will run plus the specification of the HW: in this case the GLOBE6
	BP30_GLOBE3	GLOBE3 compilation
	BP30_GOLDFINCH_A1	GOLDFINCH A1 compilation; In this case also PROJECTNAME variable must be set (see below)
	BP30_GOLDFINCH_B1	GOLDFINCH B1 compilation; In this case also PROJECTNAME variable must be set (see below)
	BP30_EVABOARD	Infineon EVB. In this case also SMARTI_SD variable must be set (see below)
SMARTI_SD	Value	Specific Value of the radio that it's used with the Infineon EVB. The vaule must be in C like hex format with 4 digits. Eg: 0x02B1. Note. If the Hw is not the EVB this variable is not used.
PROJECTNAME	BP30	Name of the project. (Note: the executable, binary or output file of the build process will have the name of the project (for instance BP30.hex in case of a TASKING C166 build).
	Goldfinch	Name of the project. (Note: the executable, binary or output file of the build process will have the name of the project (for instance

		Goldfinch.hex in case of a TASKING C166 build).
APPLICATIONS	APOXI	Apoxi will be included in the build
	REFMMI	Building of the Reference MMI will be carried out during the build.
	APPADVPHONEBOOK	Application Advanced Phonebook will be included in the build
STACK	GPRS	The 'STACK' variable denotes the current stack.
COMPILER	TASKING166	TASKING C166 is used as the default compiler.
	MSVC	Microsoft Visual C++ V6 compiler

[1] Default parameters are marked in red.

5.3.2 Useful Commands

- `make -r PLATFORM=BP30_GLOBE6` complete build for GLOBE6;
- `make -r TARGETSYSTEM=ALIHOST` complete build for host;
- `make -r BUILDMODE=DEBUG` build with debugging information;
- `make -r INT_STAGE=STT` build with only stack;
- `make -rk` build doesn't stop if errors occur;
- `make -r ADD_SYSTEM_DEFS=RAINBOW_DISPLAY` define "`RAINBOW_DISPLAY`" added to build;

Combinations of these commands are also available and useful.
For more infomations please see [1].

6 Open Issues

- **It's not possible to get a complete build with `BUILDMODE` parameter set to value `DEBUG` (this is due to a tasking linker symbol limit)**
- Dependencies implemented but disabled

7 Add/Remove Files to/from Build System

Build system is structured in different modules as follows:

- Stack modules;
- Drivers modules;
- APOXI modules;
- RefMMI modules;
- Applications modules;

From now on, we'll speak in tems of applications, but the whole description is valid for all the other types of modules.
In order to add/remove files to/from a module you have to add/remove the files to/from makefile which you want including them to (which already includes them in) to the variable ${BUILD_LIB}_FILES, e.g.

```
${BUILD_LIB}_FILES += \
                      <ApplicationFile1> \
                      <ApplicationFile2> \
                      <ApplicationFile3> \
                      <NewFile>
```

8 Compiler Defines

8.1 Defines types

In the build system different types of defines exists; they are listed below:

- `SYSTEM_DEFS`, visible to every module;
- `CC_DEFS`, visible to .c files;
- `CPP_DEFS`, visible to .cpp files;
- `ASM_DEFS`, visible to .asm files;

8.2 Where add/remove a define to/from

In the build system different files are dedicated to defines; so, if you have to add/remove a specific define, you have to choose the right place where put it in, according the following rules:

- Top level "`makeoptions.mk`" (path:system-build/make): it contains global defines which have to be applied to every module in the build;
- platform "`makeoptions.mk`" (EGOLDLITE): it contains BP30 platform specific defines which will be visible all over in the build.
- `Cbe_apoxi_makeoptions.mk` (path: lnz_apoxi/Apoxi): it contains only `APOXI_DEFS` defines which are visible only to every APOXI module;
- `Cbe_mmi_makeoptions.mk` (path: lnz_mmi/Mmi): it contains only `REFMMI_DEFS` defines which are visible only to every MMI module;
- `Cbe_mmi_plugins_makeoptions.mk` (path: lnz_mmi/MmiPlugins): it contains only `MMIPLUGIN_DEFS` defines which are visible only to every MMIPlugins module;
- `Cbe_app_makeoptions.mk` (path: lnz_apps_int/Applications): it contains only `APP_DEFS` defines which are visible only to every APPLICATIONS module;

So, for example, if you have to add a define for an APOXI module, you have to add it in the file `Cbe_apoxi_makeoptions.mk` modifying the `APOXI_DEFS` variable as follows:

```
APOXI_DEFS+= NEWDEFINE
```

9 Compiler Include

In order to add an include path to a module, path and compiler include path settings part in module makefile must be modificated. Follows an example:

If you want to include path "`dwddrv/AUD/src`" in some module, you have to add the following line to the include path part

```
${BUILD_LIB}_INC += \
    ..........
    ${GLOBALPATH}/dwddrv/AUD/src \
    ..........
    ..........
```

10 Release to 3rd Parties

In order to deliver to 3rd parties a sw release, the following rules, which don't allow 3rd part to have look to stack and Apoxi C and Cpp files but only to header files, must be added at the top of the config spec:

```
element -directory /vobs/lnz_tools/bin                              -none
element -directory /vobs/lnz_tools/src                              -none

element -directory /vobs/lnz_apoxi/Apoxi/UnitTest                   -none
element -directory /vobs/lnz_apoxi/Apoxi/AddOns/CodecManager        -none
element -directory /vobs/lnz_apoxi/Apoxi/AddOns/Lcs                 -none
element -directory /vobs/lnz_apoxi/Apoxi/AddOns/Svg                 -none
element -directory /vobs/lnz_apoxi/Apoxi/AddOns/VoiceRecognition    -none
element -directory /vobs/lnz_apoxi/Apoxi/AddOns/Wbxml               -none
element -directory /vobs/lnz_apoxi/Apoxi/AddOns/XmlParser           -none
element -directory /vobs/lnz_apoxi/Apoxi/AddOns/Zi8                 -none
element -directory /vobs/lnz_apoxi/Apoxi/t32                        -none
element -directory /vobs/lnz_mmi/Mmi/MmiAudioTest                   -none

element /vobs/lnz_apoxi/.../*.c                                     -none
element /vobs/lnz_apoxi/.../*.cpp                                   -none
element /vobs/3p_lwip_ops/.../*.c*                                  -none
element /vobs/a-gps/.../*.c                                         -none
element /vobs/drv/.../*.c                                           -none
element /vobs/dwddrv/.../*.c                                        -none
element /vobs/dwddrv/.../*.cpp                                      -none
element /vobs/dwdsrc/.../*.c                                        -none
element /vobs/dwdtools/.../*.c*                                     -none
element /vobs/lnz_tools/.../*.cpp                                   -none
element /vobs/ms-ap-src/.../*.c                                     -none
element /vobs/ms-bt-src/.../*.c                                     -none
element /vobs/ms-bt-src/.../*.cpp                                   -none
element /vobs/ms-ds-src/.../*.c                                     -none
element /vobs/ms-gprs-gm-sm-sn-src/.../*.c                          -none
element /vobs/ms-gprs-l1-src/.../*.c                                -none
element /vobs/ms-gprs-l2-src/.../*.c                                -none
element /vobs/ms-gprs-pch-src/.../*.c                               -none
element /vobs/ms-hosttest/.../*.c                                   -none
element /vobs/ms-interface-src/.../*.c                              -none
element /vobs/ms-ipr-src/.../*.c                                    -none
element /vobs/ms-l2-src/.../*.c                                     -none
element /vobs/ms-l3-src/.../*.c                                     -none
element /vobs/ms-mi-src/.../*.c                                     -none
element /vobs/ms-mn-src/.../*.c                                     -none
element /vobs/ms-om-src/.../*.c                                     -none
element /vobs/ms-si-src/.../*.c                                     -none
element /vobs/ms-src/.../*.c                                        -none
element /vobs/ms-target/.../*.c                                     -none
```

```
element /vobs/ms-test-src/.../*.c                          -none
element /vobs/os-src/.../*.c                               -none
element /vobs/os-src/OSE166/text/os166.tmp                 -none
element /vobs/platform-src/.../*.c                         -none
element /vobs/platform-src/sio/text/sio_scc.c_             -none
element /vobs/sdl-build/.../*.c                            -none
element /vobs/sim-interface/.../*.c                        -none
element /vobs/stack-interface/.../*.c                      -none
element /vobs/system-build/.../*.c                         -none
element /vobs/tools/.../*.c                                -none
element /vobs/util-src/.../*.c                             -none
```

For the complete delivery to 3rd parties, precompiled stack and Apoxi libraries must be added. Ref.MMI and Appications files are all availables to 3rd parties.

Another useful way for delivery to 3rd parties is running GDS (Global Delivery Script), which importall the files listed in a Configuration Identification file in delivery vobs.

11 Document change report

	Change Reference		Record of changes made to previous released version	
Rev	**Date**	**CR**	**Section**	**Comment**
1.0	02/052005		Creation	
1.1	04/05/2005		Update	
1.2	09/05/2005		Added „Release to 3rd parties“ chapter	
1.3	12/05/2005		Update “open issue” chapter	
1.4	30/05/2005		Added “How to have a build for EGOLDRADIO” chapter	
1.5	28/06/2005		Remove “How to have a build for EGOLDRADIO” chapter. Change PLATFORM and PROJECTNAME variables, add SMARTI_SD and VERBOSE variables.	
1.6	05/07/2005		Add CCCLIB case for BUILDMODE variable	
1.7	05/12/2005		Added “Compiler Defines” and “Compiler include” chapters	
1.8	15/02/2006		Updated list of acronyms	

12 Approval

Revision	Approver(s)	Date	Source/signature
1.0	Stefano Godeas	02/05/2005	
1.1	Stefano Godeas	04/05/2005	
1.2	Stefano Godeas	09/05/2005	
1.3	Stefano Godeas	19/05/2005	
1.4	Stefano Godeas	30/05/2005	
1.7	Stefano Godeas	05/12/2005	

Annex 1

None.

Annex2

None.

Annex 3

None.