NXP Common Driver Library (CDL) software use instructions

1 Purpose

This document describes the how to install and build the NXP software for the LH7 and LPC lines of embedded processors.

1.1 Overview of the LPC software package

The LPC software package is a collection of standardized software packages for various devices that have a common file and directory structure, build system, and usage model. The LPC software package consists of common code and tools for various NXP processors and platforms, chip specific code, and board specific code. The software and build system included in each package supports multiple tool chains such and GCC and ARM Realview.

At a minimum, the LPC software package consists of 3 sub-packages organized as a common device-independent sub-package, a chip specific package (CSP), and a board specific package (BSP). These sub-packages are bundled together for a specific chip and product and can be downloaded from the NXP website. A single package may contain support for multiple development boards.

1.2 Software organization

The software is organized into packages under a well-defined directory and file structure as shown below:

<BASE>
  TOOLS
  MAKERULE
  SOFTWARE
    DOCS
    LPC
      SOURCE
      INCLUDE
    CSPS
      <PROCESSOR>
        SOURCE
        INCLUDE
        BSPS
          <BOARD>
            SOURCE
            INCLUDE
            EXAMPLES
            STARTUP

Depending on the processor and board you are using, the <PROCESSOR> and <BOARD> names may vary.

All files are organized under the BASE directory. Although the BASE directory can be located anywhere, it is recommended that the BASE directory be named NXPMCU on the C: drive. This will allow some projects that use path absolute settings to work without changes.

The TOOLS directory contains free tools required to build code and executables. These tools provide the basis for the make system and are used in a command line environment.

The SOFTWARE directory contains the LPC directory, the CSPS (Chip Support Package) directory, the MAKERULE directory, and the DOCS directory.

The LPC directory contains common code and data used on all processors and boards.

The MAKERULE directory contains target specific makefiles used to build libraries and applications using the makefile system.

Under the CSPS directory is the selected processor directory (such as the LH7A404 or LH79520) and associated files. The BSPS directory exists under the CSPS directory and may contain one or more board or product directories.

The LPC, <PROCESSOR>, and <BOARD> directories share a structure that contains SOURCE, INCLUDE, LIB, and DOCS directories.

1.3 Software packages

A software package consists of the LPC common code, selected CSPS processors, and one or more BSPS products. Software packages also include tools required to build the software, all necessary make files, additional documentation or readme files, port files, and example code. Multiple processor and board software packages can coexist on a single machine.

1.3.1 LPC sub-package

The LPC sub-Package contains common code used on all processors and boards. These include common types and header files, common snippets of code, and generic support libraries. Files in the CSPs and BSPs may use definitions and code from this area.

1.3.2 CSP sub-package

The Chip Support sub-Package contains code specific to a certain chip. This consists of software drivers for chip functions and chip specific port files. Files in the BSP use definitions and code from this area.

1.3.3 BSP sub-package

The Board Support sub-Package contains code specific to a certain platform or board. This includes software drivers to handle board specific functions, example code, board demonstration code, board startup code, and board specific port files.

1.3.4 Other files

Each package is also released with a set of open-source tools used to control the build system. These free tools, based on UnixUtils, are used with the build system make files to generate the libraries, examples, and code in each package.

1.4 Software package downloads and versions

Software packages can be downloaded from the NXP website as zip files. Although the LPC, CSP, and BSP sub-packages are versioned separately inside the package, the downloaded file contains a single version number. Documentation included in the zip file detail the individual version numbers of the sub-packages. The filename of the zipped package is in the following format:

lpc_<processor>_<versions>.zip

Examples:

lpc_lh7a404_v1_02.zip
lpc_lh79520_v1_00.zip

2 LPC Package installation instructions

The LPC package is easy to install and requires no external tools or software to operate. This package is intended to be installed and used on a Microsoft Windows platform. It has been tested on Windows 2000 and XP platforms.

Follow the instructions listed in Section 2.1 to install the package. If you want to use the free CodeSourcery tools to build and run the code and examples, also see Section 3.0. After installation, go to Section 4.0 to build the package.

2.1 Installation steps for the NXP LPC software Package

Download the package and unzip it. It is recommended that the package be unzipped at the root directory of the C: drive in a folder called *NXPMCU*, although it can be placed anywhere. Multiple packages can co-exist by simply changing the root of where the package is unzipped. For example, you can create a directory called MYNXP on your desktop and unzip it there. If you download another software package in the future for another processor and board, you can create an MYNXP2 directory and unzip it there to keep the 2 packages separate. *Optionally, you can manually merge the 2 packages and they will co-exist fine; this is not recommended as different packages may contain different sub-package versions. Some projects may use absolute paths in their project files – these projects will not work without some changes if the C:/NXPMCU directory isn't used as the base directory.*

3 CodeSourcery GNU tools support

The CodeSourcery “Lite” GNU tools are supported in the LPC software packages in a make environment using a command line. Most code, examples, and demos can be built using these tools.

To get the CodeSourcery tools, go to http://www.codesourcery.com/. The version that needs to be downloaded has the following requirements:

Target platform:	ARM EABI
Host platform:	IA32 Windows

After installing the CodeSourcery tools, you may need to reset your system for the tools to work properly.

4 Building the software

This section explains how to setup the build system environment for a specific tool, processor, and board, and how to build the LPC/CSP/BSP sub-package libraries.

Most work will be performed in the BSP software area. There are lots of examples to demonstrate how to use the features of a specific processor.

4.1 Build system overview

The organization of the code into sub-packages allows a clear separation of common code, chip specific code, and board specific code. Included with these areas of code are scattered make files. These make files detail how a tool chain builds the code into object files, libraries, or executables when using the command line build system. This build system uses the tools located in the ./software/tools directory and its operation is designed to be transparent to the end-user.

Make file operations perform operations based on the directory you execute make in. For example, if you execute make at the C:/NXPMCU/SOFTWARE directory, it will build libraries. But if you execute make in a BSP example folder, it will build the libraries and build the executable associated with the example.

4.1.1 Preconfigured projects

Optionally, a Integrated Development Environment like ARM Realview or Keil uVision3 can be used instead of the command line environment. Many people prefer using an IDE over a command line make file driven environment. Use of an IDE is not covered in this document and setup and use vary per IDE, but most of them are easy to setup and use with the LPC software structure.

Some projects for specific tool chains and build environments may also be included in the BSP area. These projects do not use the make build system and require other commercial tool chains such as Keil uVision3, IAR Embedded Workbench, or ARM Realview. In

most cases, simply clicking on the project file will bring up the associated build environment (if it is installed) and the preconfigured project.

4.2 Software build areas using make

All of the code, examples, and demos are designed to be built from a Windows command (cmd.exe) shell. When the build system is properly setup, the make utility can be used to build libraries, startup code, examples, and demos. The make utility, as well as several other needed utilities, is provided with the LPC package.

4.2.1 Processor, Board, and tool selection

The SETENV.BAT file located in the ./NXPMCU/software directory is executed from the command shell to setup the build system environment. This script sets up the working directories for the LPC, CSP, and BSP sub-packages, and adds any needed paths for tool support. When a command shell is first opened, this script should be executed before any build operations are performed.

The syntax for the SETENV.BAT command is:

SETENV <BOARD> <TOOL> <VERBOSE>

A partial list of supported processors, boards, and tools are listed in the following table. See the individual readme files included with each CSP and BSP for the latest tool chain support. The <VERBOSE> argument is optional and is used to output more detail when compiling and linking code if <VERBOSE> is set to 1.

<PROCESSOR>	<BOARD>	<TOOL>
LH7A404	SDK7A404 (LogicPD SDK7A404 board)	ADS, RVW, GNU
LH7A400	SDK7A400	ADS, RVW, GNU
LH79520	SDK79520	ADS, RVW, GNU
LH79524	SDK79524	ADS, RVW, GNU
LH754XX	SDK75401	ADS, RVW, GNU
LPC32XX	PHY3250 (Phytec LPC3250 board)	ADS, RVW, GNU, KEIL, IAR

Example that selects the lh7a404 processor, sdk7a404 board, and Realview tools:

setenv sdk7a404 rvw

Example that selects the lh79520 processor, sdk79520 board, and CodeSourcery GNU tools:

setenv sdk79520 gnu

Example that selects the lpc32XX processor, Phytec LPC3250 board, and CodeSourcery GNU tools with verbose compiler and linker messages:

setenv phy3250 gnu 1

4.2.1.1 Supported 3rd party tool chains

The following tool chains may be supported in the LPC software packages. See the readme file included with a downloaded package to see what tool chains are supported by that package.

ARM Developers Suite version 2.x (ads)
ARM Realview Suite version 3.0 (rvw)
www.arm.com

IAR Embedded Workbench (iar)
http://www.iar.com/

Codesourcery Sourcery G++ Lite for EABI
http://www.codesourcery.com/

Keil uVision3
http://www.keil.com/

4.2.1.2 Supported boards

Information on the various supported SDK boards is available at www.logicpd.com.

4.3 Building libraries and executables

Once a command shell is opened and the environment setup with SETENV.BAT, the libraries and various code examples can be built. The build system relies on make files and the make utility to build software components.

4.3.1 Make file commands

The make command will perform different functions based on the directory where it is executed. For example, if you type **make clean** in the ./NXPMCU/SOFTWARE directory, it will clean the temporary build files in the LPC, CSP, and BSP sub-packages. However, if you type **make clean** in the ./NXPMCU/SOFTWARE/LPC directory, it will only clean the temporary build files in the LPC sub-package.

All make files support the **make** and **make clean** commands. See the readme files included in the BSP for additional supported commands. *For example, some BSPs allow special LOLO builds that generate files that can load through the SDK loader on LH7 boards.*

4.3.2 Build dependencies

The build system will automatically determine dependencies for examples and code and will rebuild any necessary files and libraries as needed.

4.4 Building and executing examples and demos

Each BSP includes an example area that details how to use interfaces on the selected CSP. These examples are usually simple code programs that use just one peripheral on the device. Some BSPs include demos; demos are more encompassing than examples and usually work with multiple peripherals simultaneously or kernels such as uCos-II or ThreadX.

4.4.1 Board initialization

Some boards require initialization before running examples or demos. For example, the PHY3250 examples require initialization to setup the SDRAM and MMU before examples can even be loaded into main memory. Although example initialization code is provided with each BSP, the best way to initialize the board is to let the built-in bootloader perform initialization before stopping the board. This allows simple examples and demos to work with a pre-existing board environment.

4.4.2 Building an example

To build an example, simply type *make* in the example's directory. The build system will generate a map file and an executable fileSome BSPs amy also generate an S-record file. The executable can be loaded with debuggers such as Abatron BDI2000 or Realview ICE. The S-record file can be used with the board loader/monitor programs. Some examples may also generate other file types such as binaries.

BSPs may differ slightly in the file type generated and the make commands needed to generate an executable. See the documentation included with the BSP for more information on how to build, load, and run examples and types of files generated.

4.4.2.1 Full build example

The following text is the output from a cmd.exe session doing a build with CodeSourcery GNU tools for the 'timer' example with the PHY3250 BSP. Most examples are built using this same method for different tool chains, BSPs, and CSPs. The build was done for the first time on a clean system, so all of the files (CSP and BSP) were compiled and built when the make command was used with the example. At the end of the output, the current directory is dumped and the executable (timer.elf) and the S-record (timer.srec) files are shown.

```
C:\nxpmcu\software>cd \nxpmcu\software

C:\nxpmcu\software>setenv phy3250 gnu
Base LPC install dir    : C:\nxpmcu\software
Base LPC unix dir       : C:/nxpmcu/software
Extra Tool install dir  : C:\nxpmcu\software\tools
Selected CSP            : lpc32xx
Selected BSP            : phy3250
Selected toolchain      : gnu

C:\nxpmcu\software>cd csps

C:\nxpmcu\software\csps>cd lpc32xx

C:\nxpmcu\software\csps\lpc32xx>cd bsps
```

```
C:\nxpmcu\software\csps\lpc32xx\bsps>cd phy3250

C:\nxpmcu\software\csps\lpc32xx\bsps\phy3250>cd examples

C:\nxpmcu\software\csps\lpc32xx\bsps\phy3250\examples>cd timer

C:\nxpmcu\software\csps\lpc32xx\bsps\phy3250\examples\timer>make
CC timer_example.c
arm-none-eabi-gcc -c -mcpu=arm926ej-s -Wall -O3 -mapcs-frame -DLPC_CHIP= -mno-sc
hed-prolog -fno-hosted -mno-thumb-interwork -IC:/nxpmcu/software/csps/lpc32xx/in
clude -IC:/nxpmcu/software/csps/lpc32xx/bsps/phy3250/include -IC:/nxpmcu/softwar
e/lpc/include -gdwarf-2 timer_example.c -o timer_example.o
CC lpc32xx_adc_driver.c
CC lpc32xx_clcdc_driver.c
CC lpc32xx_clkpwr_driver.c
lpc32xx_clkpwr_driver.c: In function 'clkpwr_check_pll_setup':
lpc32xx_clkpwr_driver.c:146: warning: 'cfreq' may be used uninitialized in this
function
lpc32xx_clkpwr_driver.c:146: warning: 'fref' may be used uninitialized in this f
unction
lpc32xx_clkpwr_driver.c:146: warning: 'fcco' may be used uninitialized in this f
unction
CC lpc32xx_dma_driver.c
CC lpc32xx_gpio_driver.c
CC lpc32xx_i2s_driver.c
CC lpc32xx_intc_driver.c
CC lpc32xx_kscan_driver.c
CC lpc32xx_mlcnand_driver.c
lpc32xx_mlcnand_driver.c: In function 'mlcnand_read_page':
lpc32xx_mlcnand_driver.c:721: warning: 'dwptr' is used uninitialized in this fun
ction
lpc32xx_mlcnand_driver.c: In function 'mlcnand_write_page':
lpc32xx_mlcnand_driver.c:1037: warning: 'dwptr' is used uninitialized in this fu
nction
CC lpc32xx_mstimer_driver.c
CC lpc32xx_pwm_driver.c
lpc32xx_pwm_driver.c: In function 'pwm_open':
lpc32xx_pwm_driver.c:111: warning: unused variable 'tmp'
lpc32xx_pwm_driver.c: In function 'pwm_ioctl':
lpc32xx_pwm_driver.c:212: warning: unused variable 'tmp'
CC lpc32xx_rtc_driver.c
CC lpc32xx_sdcard_driver.c
lpc32xx_sdcard_driver.c: In function 'sd0_cmd_interrupt':
lpc32xx_sdcard_driver.c:587: warning: unused variable 'tmp'
lpc32xx_sdcard_driver.c: In function 'sd_start_data_write':
lpc32xx_sdcard_driver.c:736: warning: unused variable 'tmp'
CC lpc32xx_slcnand_driver.c
CC lpc32xx_ssp_driver.c
lpc32xx_ssp_driver.c: In function 'ssp_standard_interrupt':
lpc32xx_ssp_driver.c:218: warning: suggest parentheses around comparison in oper
and of &
lpc32xx_ssp_driver.c: In function 'ssp_write':
lpc32xx_ssp_driver.c:613: warning: unused variable 'tmp1'
CC lpc32xx_timer_driver.c
CC lpc32xx_tsc_driver.c
lpc32xx_tsc_driver.c: In function 'tsc_read_ring':
lpc32xx_tsc_driver.c:561: warning: unused variable 'tscregsptr'
CC lpc32xx_uart_driver.c
AS lpc32xx_vectors.asm
creating lpc32xx Chip support package library
arm-none-eabi-ar: creating C:/nxpmcu/software/csps/lpc32xx/lib/liblpc32xxgnu.a
CC libnosys_gnu.c
CC phy3250_board.c
CC phy3250_nand.c
creating phy3250 board support package library
arm-none-eabi-ar: creating C:/nxpmcu/software/csps/lpc32xx/bsps/phy3250/lib/libp
hy3250gnu.a
CC lpc_api.c
CC lpc_arm922t_cp15_driver.c
```

```
CC lpc_bmp.c
CC lpc_colors.c
CC lpc_fat16.c
CC lpc_fat16_private.c
CC lpc_fonts.c
CC lpc_heap.c
CC lpc_helvr10.c
CC lpc_lbecc.c
CC lpc_lcd_params.c
CC lpc_nandflash_params.c
CC lpc_rom8x16.c
CC lpc_rom8x8.c
CC lpc_swim.c
CC lpc_swim_font.c
CC lpc_swim_image.c
CC lpc_winfreesystem14x16.c
CC lpc_x5x7.c
CC lpc_x6x13.c
creating phy3250 board support package library
arm-none-eabi-ar: creating C:/nxpmcu/software/lpc/lib/liblpcarm926ej-sgnu.a
arm-none-eabi-gcc timer_example.o   ../common/crt0_gnu.o  -static -Wl,--start-gr
oup C:/nxpmcu/software/csps/lpc32xx/lib/liblpc32xxgnu.a C:/nxpmcu/software/csps/
lpc32xx/bsps/phy3250/lib/libphy3250gnu.a C:/nxpmcu/software/lpc/lib/liblpcarm926
ej-sgnu.a -lgcc -lc -lg -lm -liberty -lstdc++ -lsupc++  -Wl,--end-group  -Xlinke
r -Map -Xlinker \
timer.map -Xlinker -T   ../linker/ldscript_ram_gnu.ld   \
-o timer.elf
arm-none-eabi-objcopy -O srec timer.elf timer.srec

C:\nxpmcu\software\csps\lpc32xx\bsps\phy3250\examples\timer>ls -la
total 273
drwxrwxrwx   1 user     group           0 May  9 10:30 .
drwxrwxrwx   1 user     group           0 May  8 13:37 ..
-rw-rw-rw-   1 user     group         979 May  9 10:29 .depend
drwxrwxrwx   1 user     group           0 May  8 11:16 .svn
-rw-rw-rw-   1 user     group        1340 Apr 18 12:19 makefile
-rw-rw-rw-   1 user     group       90101 May  9 10:30 timer.elf
-rw-rw-rw-   1 user     group       68524 May  9 10:30 timer.map
-rw-rw-rw-   1 user     group       99828 May  9 10:30 timer.srec
-rw-rw-rw-   1 user     group        7313 Apr 18 12:19 timer_example.c
-rw-rw-rw-   1 user     group        8752 May  9 10:30 timer_example.o

C:\nxpmcu\software\csps\lpc32xx\bsps\phy3250\examples\timer>
```

5 Miscellaneous

5.1 Build systems with multiple tool chains

For developers that are building or testing with multiple tool chains or multiple CSPs, always be sure to do a **make clean** when switching between tool chains or CSPs. This may clear object code from one tool chain that is incompatible with another tool chain or processor.

5.2 Additional documentation

Examine the readme files and PDF documents included in the LPC, CSP, and BSP documentation directories for additional information not included in this document. The readme files include topics such as build notes, tool chain issues, debugger help, etc.