Using the LPC32XX Chip Support Package (CSP) and the Phytec 3250 Board Support Package (BSP)

Table of contents

1 Introduction

The PHY3250 Board Support Package (BSP) provides a starting point for applications and software based on the LPC3250 MCU and the Phytec 3250 board. The BSP is based on the LPC32xx Chip Support Package (CSP) and includes examples on how to use the CSP drivers to control LPC32XX MCU and PHY3250 board functions. Sample startup code and applications are also provided that can be used as a basis for developing other applications for the board.

The examples included with the BSP show how to use the LPC32XX CSP drivers in an example application. The examples can be loaded and executed through the Stage 1 loader included on the PHY3250 board or through a debugger. Examples can also be binded to the startup code to make self-executing examples that are booted when the board is powered up.

This document explains how to build the examples and startup code and how to load and execute the examples with the Stage 1 loader (S1L).

1.1 Installing the BSP

To install the BSP, simply unzip the BSP in a work area on your storage device. It is recommended that the BSP be installed at the C:/NXPMCU directory to prevent possible problems with projects that require absolute path names.

After installation, the tress will look similar to the following:
C:/NXPMCU
C:/NXPMCU/SOFTWARE
C:/NXPMCU/SOFTWARE/MAKERULE
C:/NXPMCU/SOFTWARE/CSPS
C:/NXPMCU/SOFTWARE/TOOLS
C:/NXPMCU/SOFTWARE/CSPS/<CHIP>
…
…
…
…

1.2 Selecting a tool chain to use

To build code and examples from this BSP, a tool chain is needed. The tool chain consists of compilers, linkers, and other build tools necessary to make the libraries and executables from the source code. The PHY3250 BSP supports GNU (CodeSourcery), IAR (Embedded Workbench), Keil (uVision3), and ARM Realview 3.x.

At least 1 of these toolchains must be installed to build the CSP and BSP code. If a toolchain isn't currently available, a lite version of the GNU tools can be downloaded and used for free from www.codesourcery.com.

1.2.1 IAR toolchain path modifications

If IAR Embedded Workbench is used to develop the applications and the make based build system is going to be used, then the paths to the IAR tool executables location must be added to the PATH environment variable. This can be done by editing the setenv.bat file, adding a small batch file to add the path when a CMD shell is opened, or by changing the default user PATH value in Windows.

1.2.2 IAR and Keil include path additions

If using the Keil or IAR toolchains, locate the setenv.bat file and update the file path environment variables specified in that file. As of the current release, the environment variables are shown below with their default values.

SET KEIL_BIN_BASE=C:\Keil\ARM\BIN31
SET KEIL_RVCT=C:\Keil\ARM\RV31
SET IAR_ROOT=C:\Program Files\IAR Systems\Embedded Workbench 5.0 Evaluation

1.2.3 Pre-built IDE based projects

There are several pre-built IDE based projects for IAR Embedded Workbench, Keil uVision3, and ARM Realview 3.x. These projects are found in some of the driver and startup examples. If you have the correct tool installed, the project can be started by clicking on the project icon.

The pre-built IDE projects may need to be tweaked for the setup you are running on. For example, if you are using a different hardware debugger, the debugger options will need to be changed. The project assumes the BSP is installed in the /NXPMCU/SOFTWARE area. If it is installed elsewhere, you may need to change the paths in the project.

1.3 IMPORTANT INFORMATION

Prior to build any code with this BSP, the 3 system revision defines need to be setup to their correct values. Locate the phy3250_board.h file and set the following defines to the matching values indicated on the LCD module, CPU mode, and carrier board of your Phytec 3250 system. If these defines are set to the wrong values, the software may work correctly for your board.

```
/* Carrier board revision selection - The carrier board revision is a
   value of 1305.x, where x = 0 to 3 */
#define PHY3250_CARRIERBOARD_1305_X 0

/* Module board revision selection - The module board revision is a
   value of 1304.x, where x = 0 to 1 */
#define PHY3250_MODULEBOARD_1304_X 0
```

```
/* LCD revision selection - The LCD revision is a value of 1307.x,
   where x = 0 to 1 */
#define PHY3250_LCD_1307_X 0
```

2 BSP software

This BSP includes examples, startup code, and driver files applicable to the PHY3250 board and drivers files specific to the LPC32XX MCU. Examples that show how to use the CSP drivers provide a good starting point to understanding the LPC32XX MCU and its peripherals. Startup code and several startup examples are also provided to get up and running quickly if the kickstart or stage 1 loader are to be replaced.

2.1 Supported LPC32XX drivers

See the lpc32xx_readme.txt file included in the CSPS/LPC32XX directory for supported drivers and additional information on the CSP.

2.2 Example code

The example code shows how to configure and use the LPC32XX peripheral drivers for various functions. The examples may include functions from multiple drivers (such as the touchscreen example, which uses functions from the Interrupt, LCD, and ADC/Tocuhscreen drivers). Although these examples are intended to be executed from S1l, some of the examples may not have a visual output and are much better suited for single stepping through with a hardware debugger.

2.2.1 Specific toolchain examples

Some pre-built projects are also provided for specific toolchains such as Keil uVision3 or ARM Realview 3.x. If you have the supported toolchain installed, you can click on the toolchain project file to open a ready-made project for the example. *Not every example has a pre-built project.*

2.3 Startup code and examples

The board startup code is included in the PHY3250/STARTUP folder. This code provides the initial setup for the board including initial mux setup, SDRAM initialization, and clock setup. This code is usually considered a minimum set of functions to get a board up and running. The startup code is used with a application by calling c_entry in the application. The *simple* startup example shows a simple application (blinking LEDs) using the startup code.

2.3.1 Kickstart loader

The kickstart loader handles the boot ROM to stage 1 application transfer. The LPC32XX bootrom has a boot size limit of 1 FLASH block, or about 54Kbytes, whichever is smaller. The kickstart loader provides a mechanism to download larger applications into memory by bridging the size gap between the boot ROM and stage 1 application. The source code for the kickstart loader is included with the BSP.

Customers can install their own stage 1 application to be booted by the kickstart loader. See the phy32xx_bl.doc file for information on how kickstart loads the stage 1 application.

2.3.2 Stage 1 loader (S1L)

The stage 1 loader is based on the startup code included in the BSP. The stage 1 loader is the default stage 1 application loaded by the kickstart loader when the board is powered on or reset. The stage 1 loader application provides various capabilities to load and run applications, examine memory, or change system configuration. More information about the stage 1 loader can be found in the lpc32xx_bl.doc file.

2.4 Other applications

Other applications and demos are available for the Phytec 3250 board such as pre-built WinCE and Linux executables. Documentation is included with those applications on how to setup and execute them on the Phytec 3250 board.

2.4.1 LPC3250 serial loader (LSL)

The LPC3250 serial loader (LSL) is a Windows tool that connexts to the LPC3250 through a serial port. This tool communicates with the LPC3250 when the LPC3250 is reset and provides a method to get code to the chip and board without a JTAG device, or even if the board boot devices have been erased.

More information about the LSL can be found in Section 5.2

3 Building and deploying an application

Applications can be built for the Phytec board using one of the supported tool chains. Applications can be built to work with or without the supplied board startup code, depending on whether the S1L application will be used.

There are also various methods for getting your application to the Phytec board for execution such as S1L, LPC3250 serial loader (LSL) tool, or a hardware debugger. The application can be setup to automatically start when the board is powered up and can be programmed into NAND FLASH or boot via SD card or serial port.

3.1 Application types

3.1.1 Applications using the stage 1 loader

Applications built to use the S1L are the simplest to build and deploy. These types of applications are loaded through the S1L's *load* command and use the system environment and configuration already setup by the S1L. These types of applications typically require only 'light' startup code, as most of the system startup has been handled by the S1L application. Most of the default examples included with the BSP fall into this category.

To build an example to load and execute through S1L, read Section 4.1.

3.1.2 Applications replacing the stage 1 loader

Applications replacing the stage 1 loader are designed to be loaded and executed with the kickstart loader. These types of applications require more robust system startup code to initialize SDRAM, chip clocking, LPC3250 pin muxing, etc. The startup code included with the BSP is a great reference to start with if you are doing this type of application. Most of the examples included with the BSP can be built to execute as the stage 1 application by making a new startup project and copying the example code into that project.

See Section 4.2 for information on how to create and application of this type.

3.2 Application deployment methods

3.2.1 Using S1L

S1L provides methods for deploying an application through the serial port, SD/MMC cards, and NAND FLASH. Binary files and S-record files are supported. See the phy32xx_bl.doc file for more information on how to use S1L.

See Section 4.1 for how to deploy and application through S1L.

3.2.2 LPC3250 boot methods

The LPC3250 provides the ability to boot from the serial port, Synchronous Serial Port (SSP), or via NAND FLASH. This BSP supports the serial port and NAND FLASH boot methods.

4 Building and deploying applications

This section explains how to build and deploy application to the Phytec 3250 board using S1L or the LPC3250 serial loader (LSL) tool. Customers can build their own applications and restore the board to its original state if the application fails to boot.

All that’s needed is a PC with a serial port, a serial cable between the PC and the Phytec 3250 board, a terminal program (such as TeraTerm), and one of the supported tool chains to build the application. Optionally, an SD/MMC card can be used to store and deploy large applications that are booted and executed through S1L.

Applications can be built with startup code that completely initializes the board or without startup code. Application built without startup code requires that the board already be initialized before starting the application. Applications loaded through S1L already have the board configured when they are loaded and executed.

4.1 Building an application for use with S1L

Applications that are loaded through S1L do not require full startup code and can be loaded and executed without any special hardware. These types of applications are

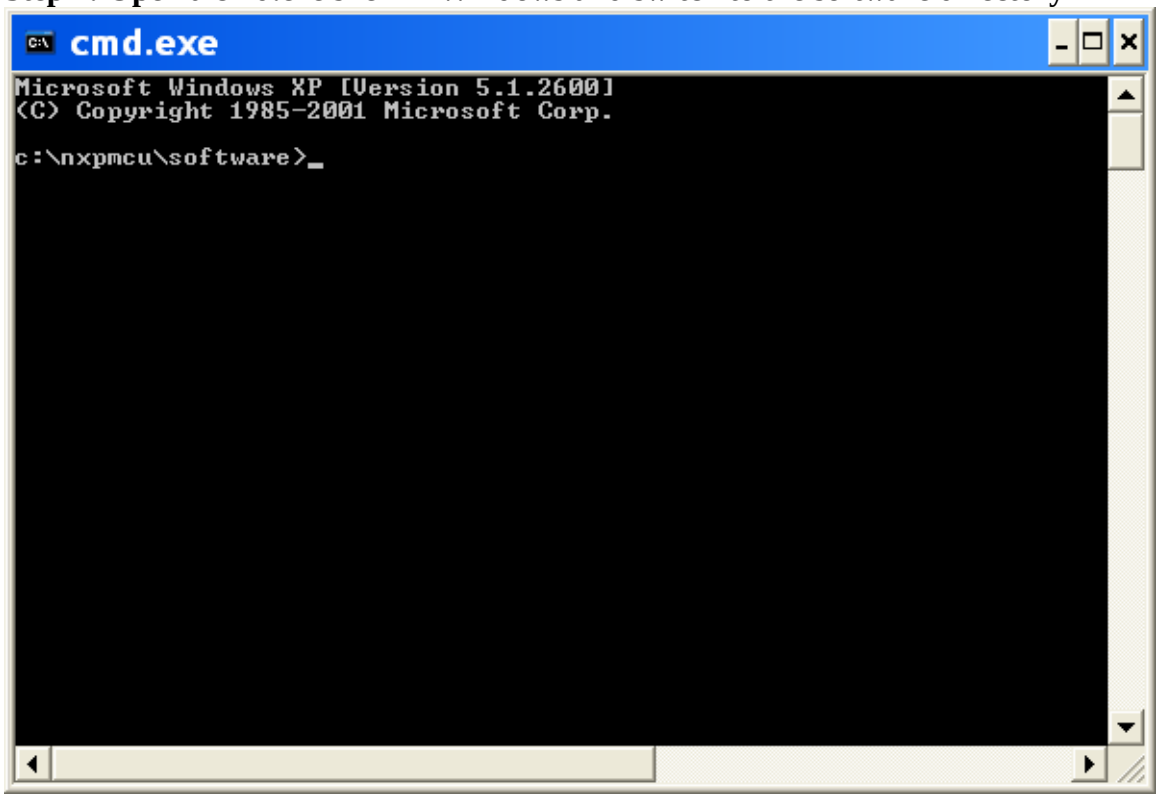
usually load and run applications. If they don't work correctly or lock up the board, simply resetting the board will usually return the board to a good state.

These types of applications can be saved into NAND FLASH and booted from NAND FLASH (through S1L), loaded through the serial port, or loaded from an SD/MMC card. These applications are simple and quick to develop and shield the end user from the details of FLASH organization and system startup.

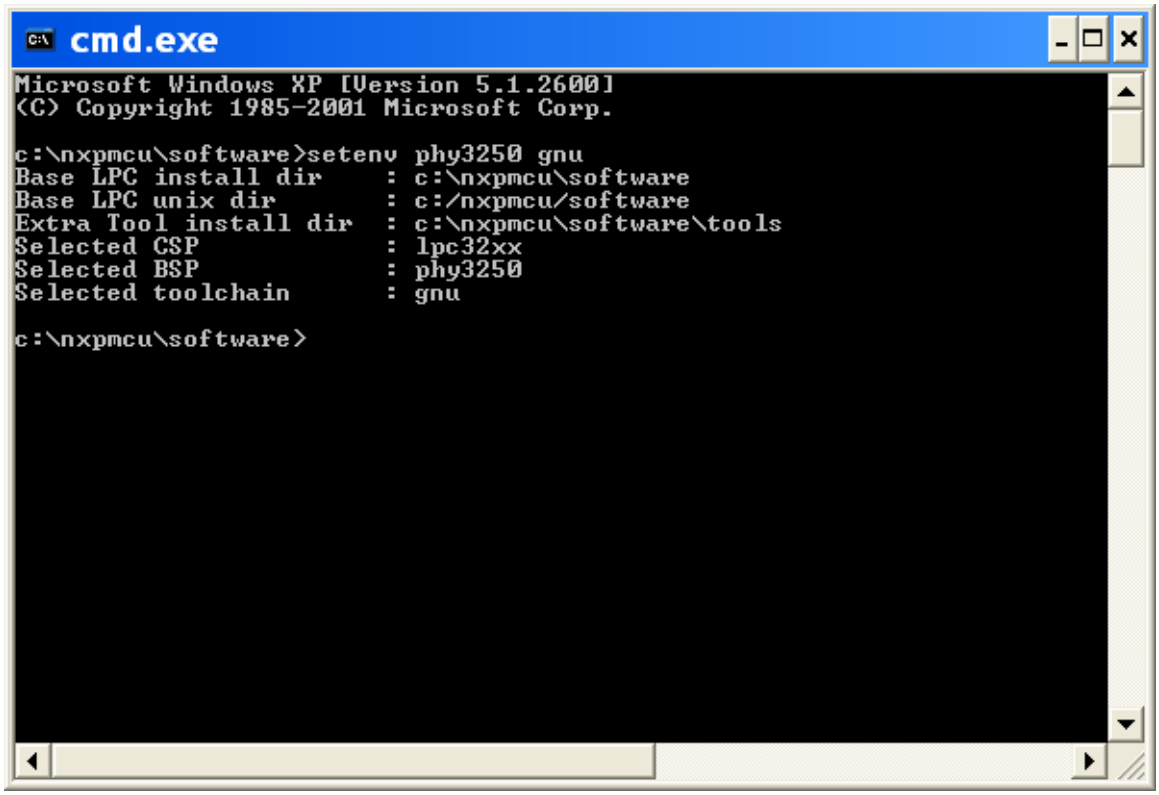
4.1.1 Building the application

These steps detail how to build the timer application using the CodeSourcery GNU toolchain. This type of application will be built to load and execute through S1L.

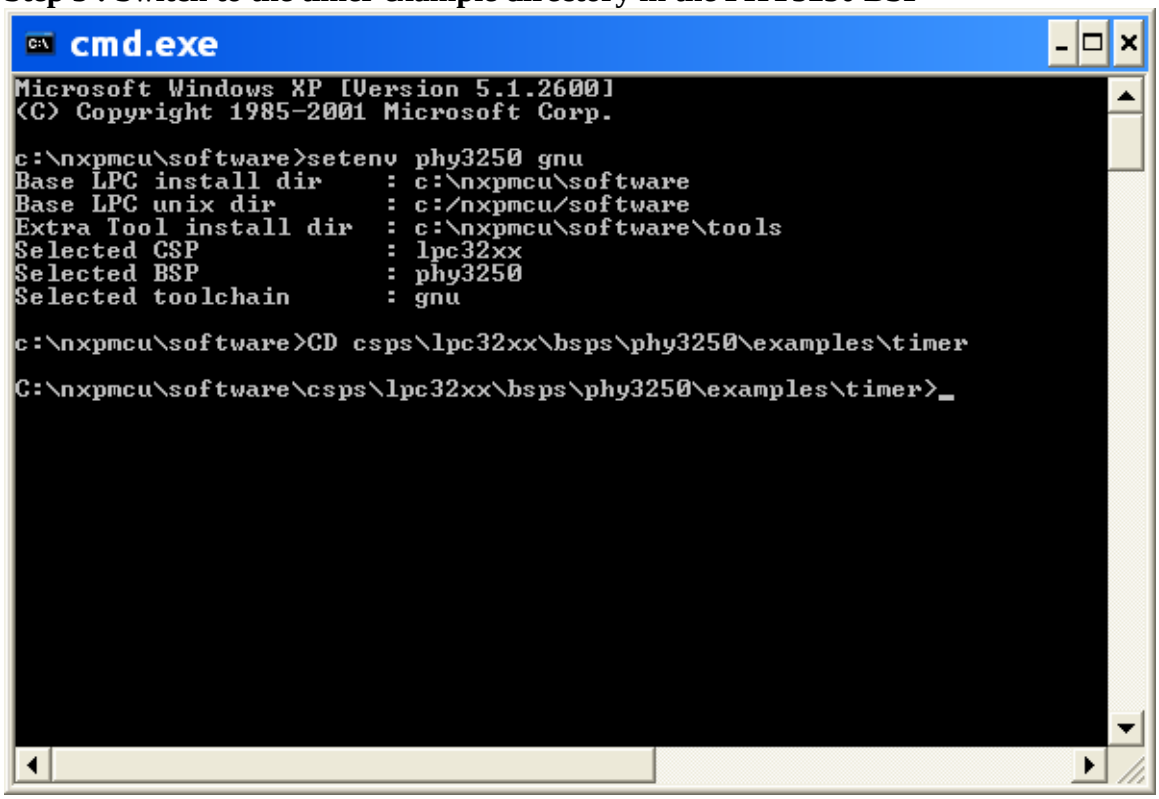
Step 1: Open a cmd.exe shell in Windows and switch to the software directory



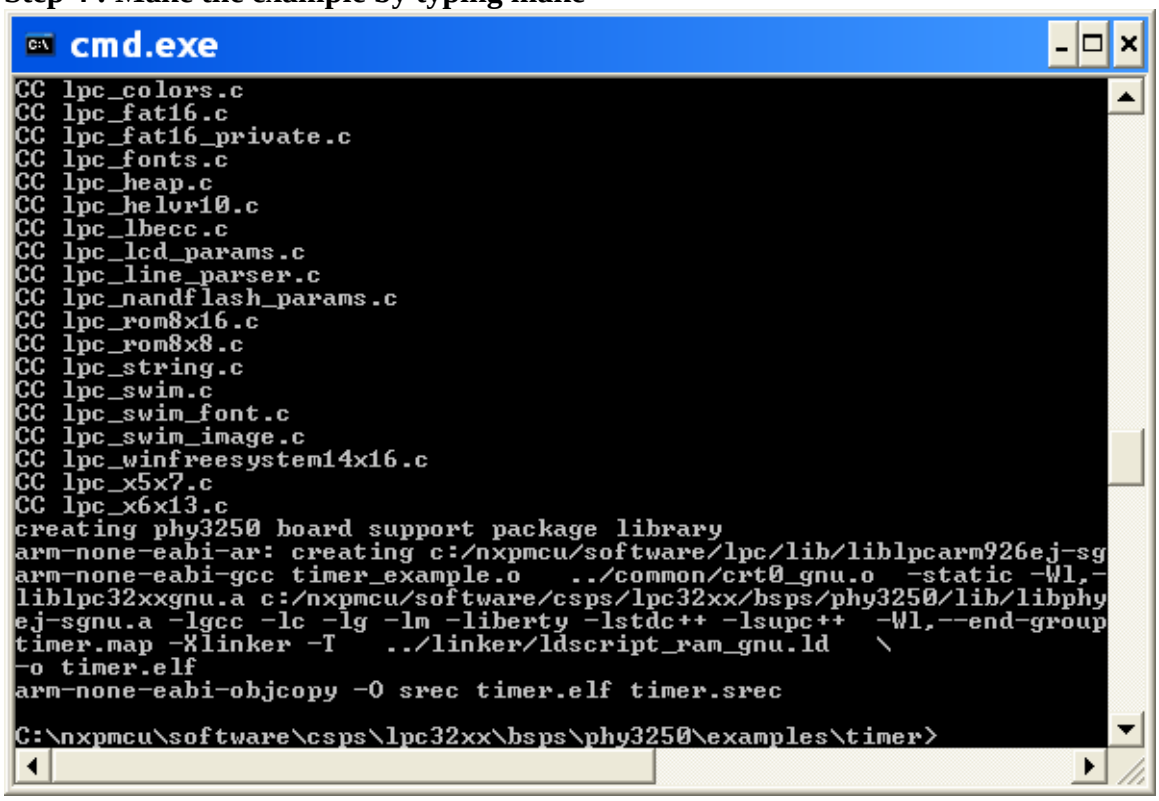
Step 2 : Setup the build environment for the Phytec 3250 (PHY325) BSP and GNU



Step 3 : Switch to the timer example directory in the PHY3250 BSP



Step 4 : Make the example by typing make



The example build is complete! The timer.elf file can be loaded and executed through a hardware debugger, while the timer.srec file can be loaded and executed through S1L.

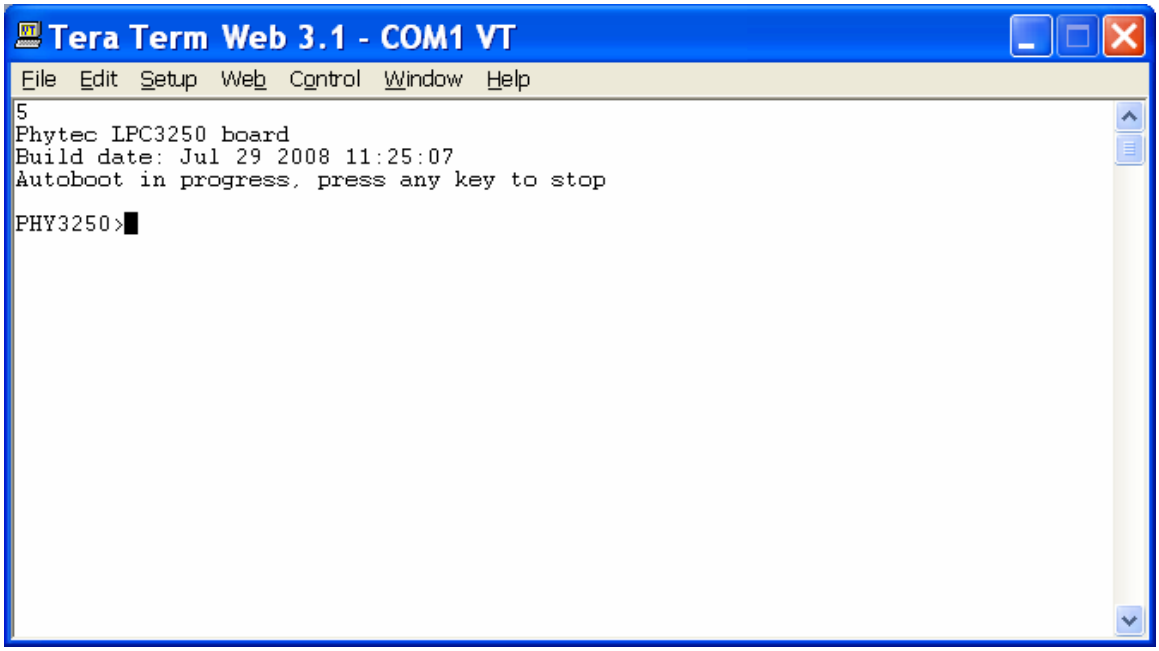
4.1.2 Deploying the application

The application can be loaded and executed on the board using S1L commands and load methods. The following 2 methods explain how to load and execute the timer example generated in Section 4.1.1 on the board using the serial port or an SD/MMC card.

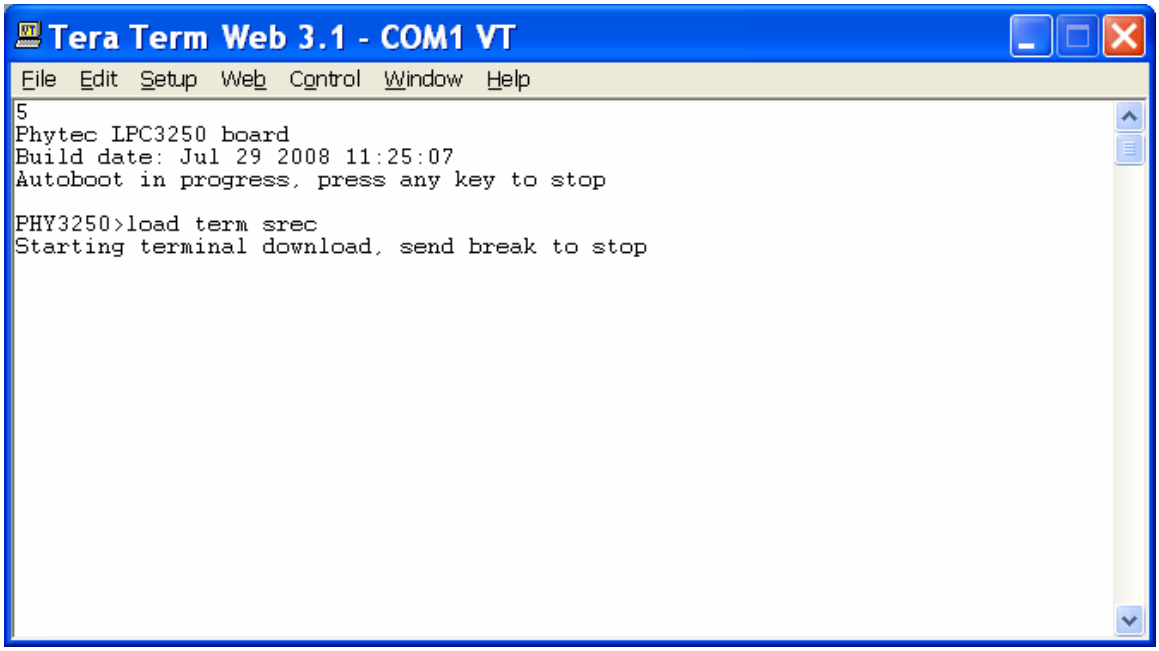
4.1.2.1 Deploying the application using the serial port

The following steps explain how to deploy an application to the Phytec board using S1L's serial port and load/exec commands.

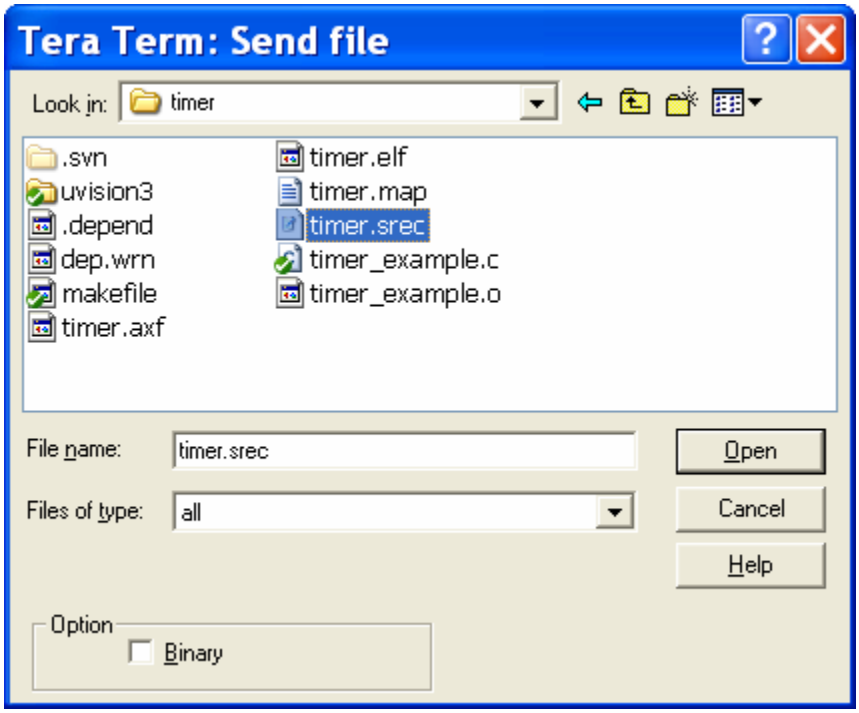
Step 1 : Start your terminal program (115.2K-8-N-1) and connect to the board. A message similar to what is shown below will appear. You may need to press a key to get to the PHY3250> S1L prompt.



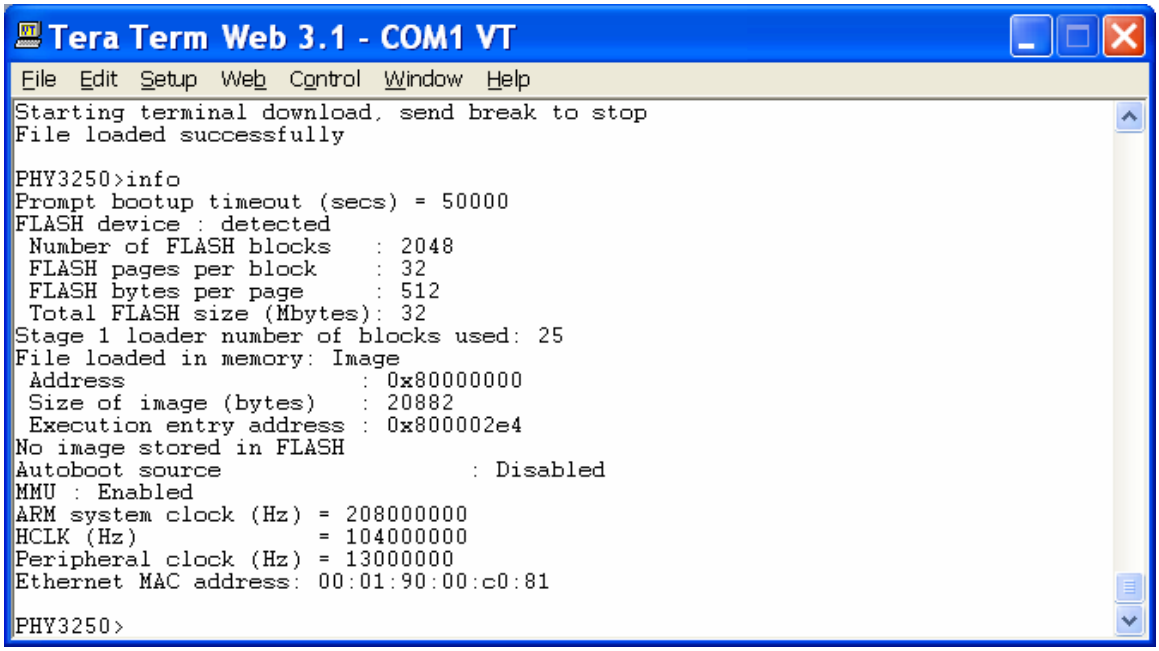
Step 2 : Using the load command, tell S1L that you want it to receive a S-record file from the terminal.



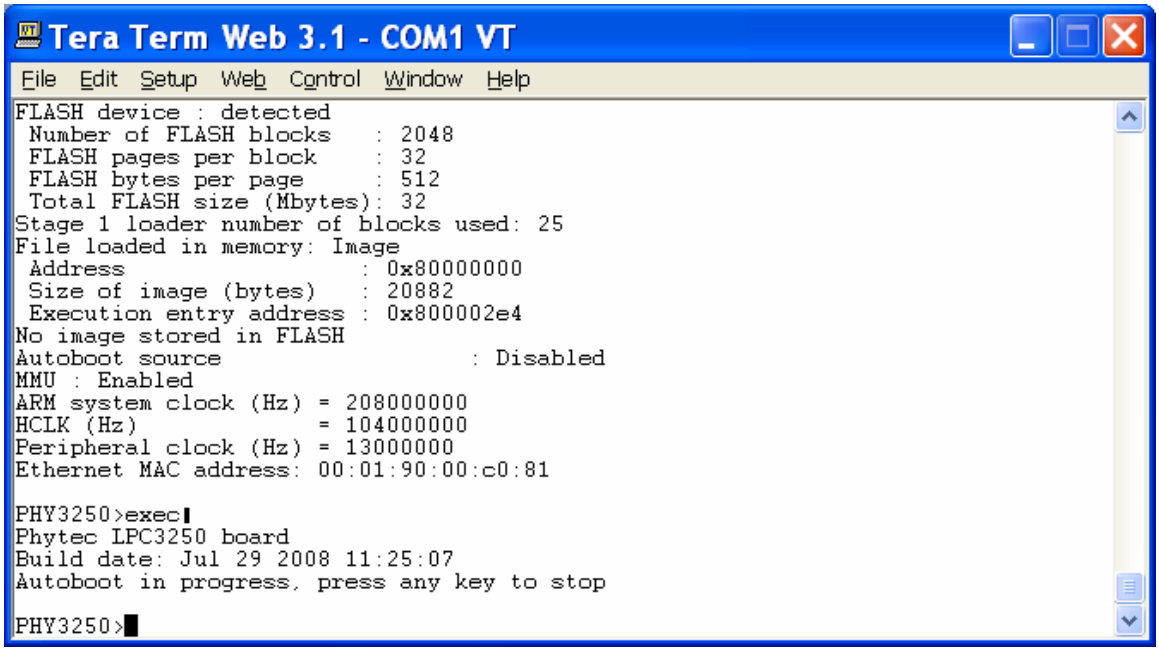
Step 3 : Send the file from the terminal program as a binary or text file (no protocol).



Step 4 : After the file has loaded, the prompt will return (binary files require a break to be sent from the terminal to return to the prompt). Type the info command to see the loaded file information.



Step 5 : The file loaded in memory is loaded at address 0x80000000 and is 20882 bytes in size. It is executed by starting the code at its execution entry address, which is 0x800002E4. Use the exec command to start the example. The example will run and after a few seconds, the prompt will return. The example has run successfully.

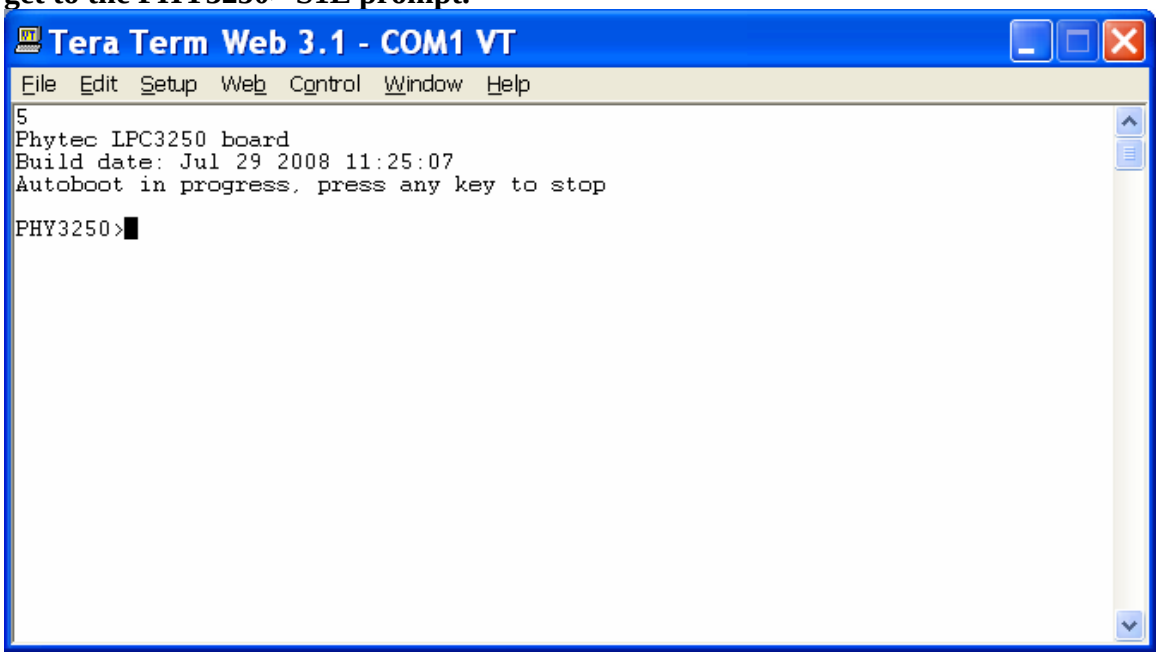


The same procedure can be used with the other examples included with the BSP.

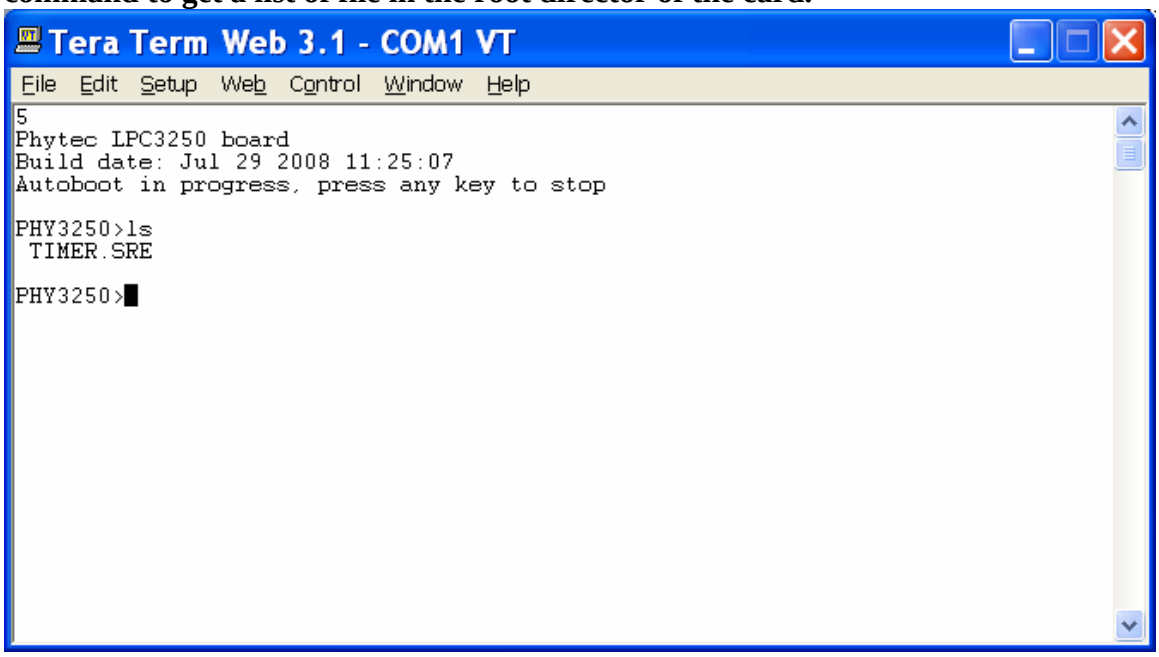
4.1.2.2 Deploying the application using an SD/MMC card

The following steps explain how to deploy an application to the Phytec board using Phytec board SD/MMC slot and an SD/MMC card.

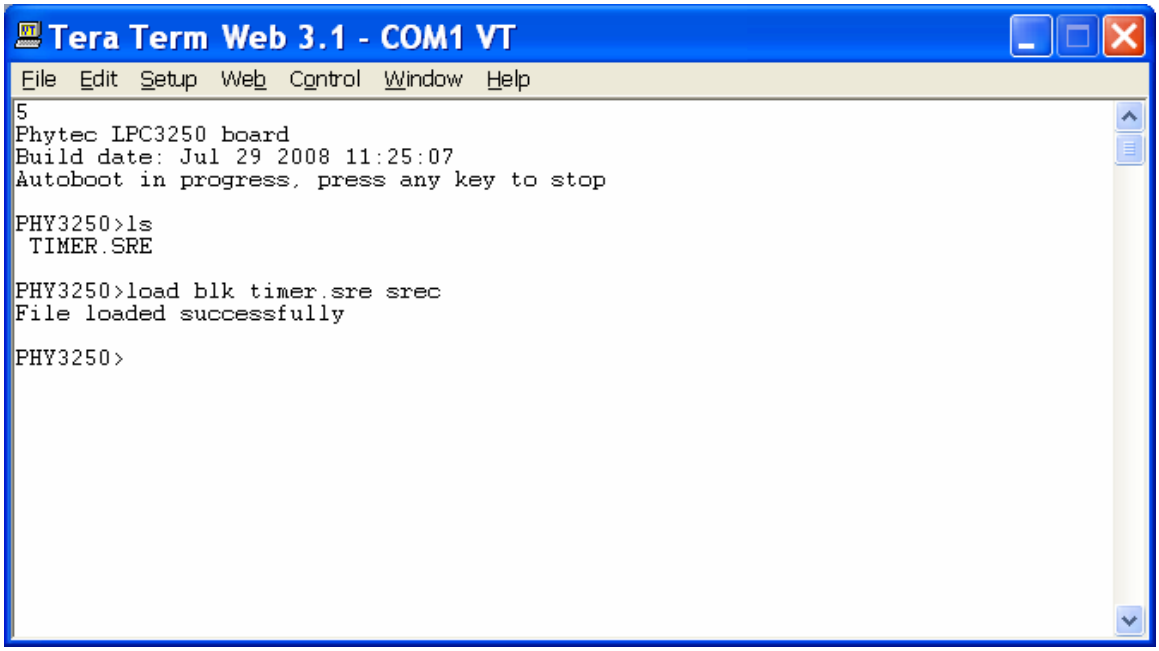
Step 1 : Start your terminal program (115.2K-8-N-1) and connect to the board. A message similar to what is shown below will appear. You may need to press a key to get to the PHY3250> S1L prompt.



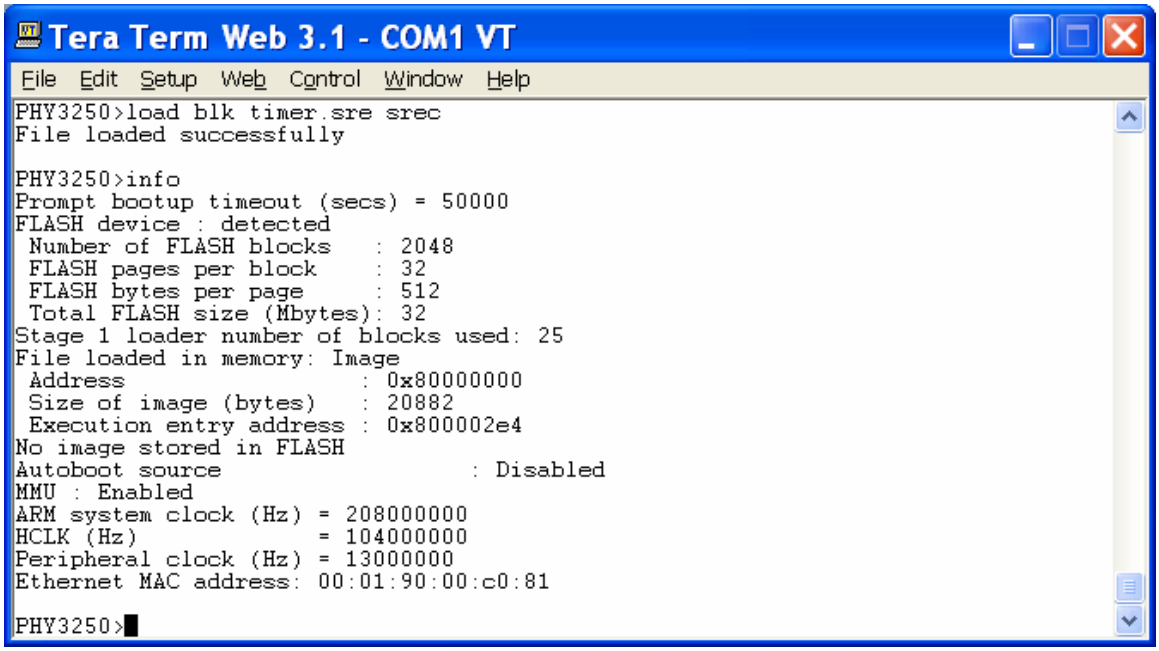
Step 2 : Place your program (timer.srec) into the root directory of an SD or MMC card and insert the card into the SD/MMC slot on the board. It is recommended that the name be renamed to an 8.3 format name (such as timer.sre) to make loading the file easier. Once the card has been inserted into the board, use the ls command to get a list of file in the root director of the card.



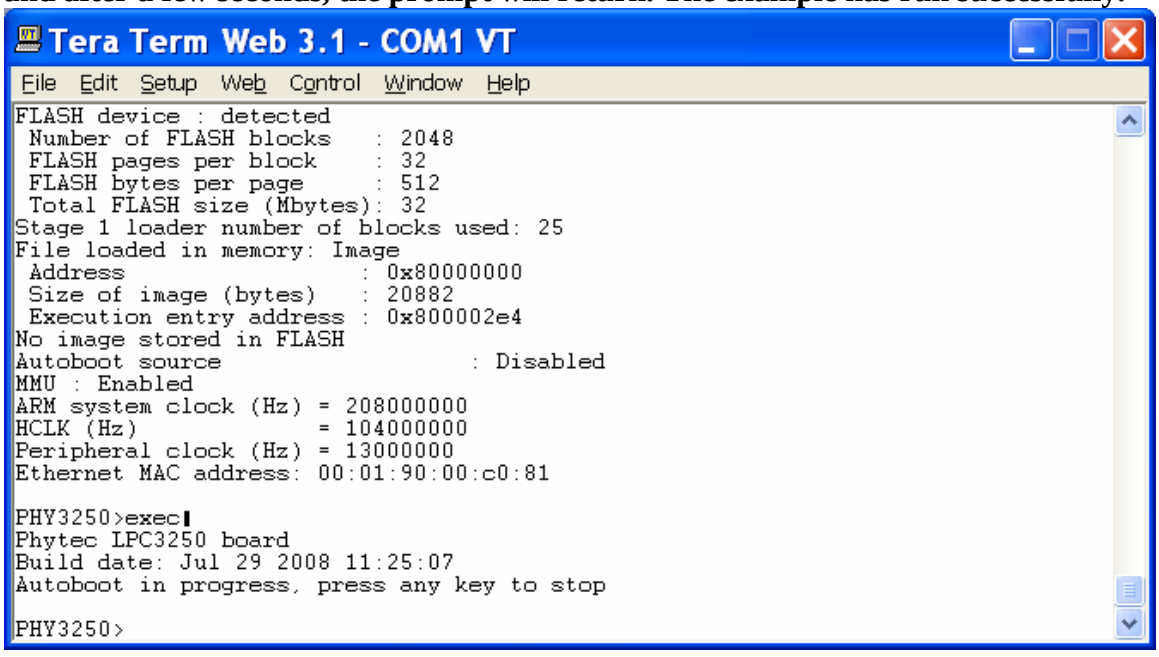
Step 3 : Use the load command to get the S-record file into the board's memory. The syntax of the load command changes slightly from the serial loaded version of the command.



Step 4 : After the file has loaded, the prompt will return. Type the info command to see the loaded file information.



Step 5 : The file loaded in memory is loaded at address 0x80000000 and is 20882 bytes in size. It is executed by starting the code at its execution entry address, which is 0x800002E4. Use the exec command to start the example. The example will run and after a few seconds, the prompt will return. The example has run successfully.



The same procedure can be used with the other examples included with the BSP.

4.2 Building a bootable application

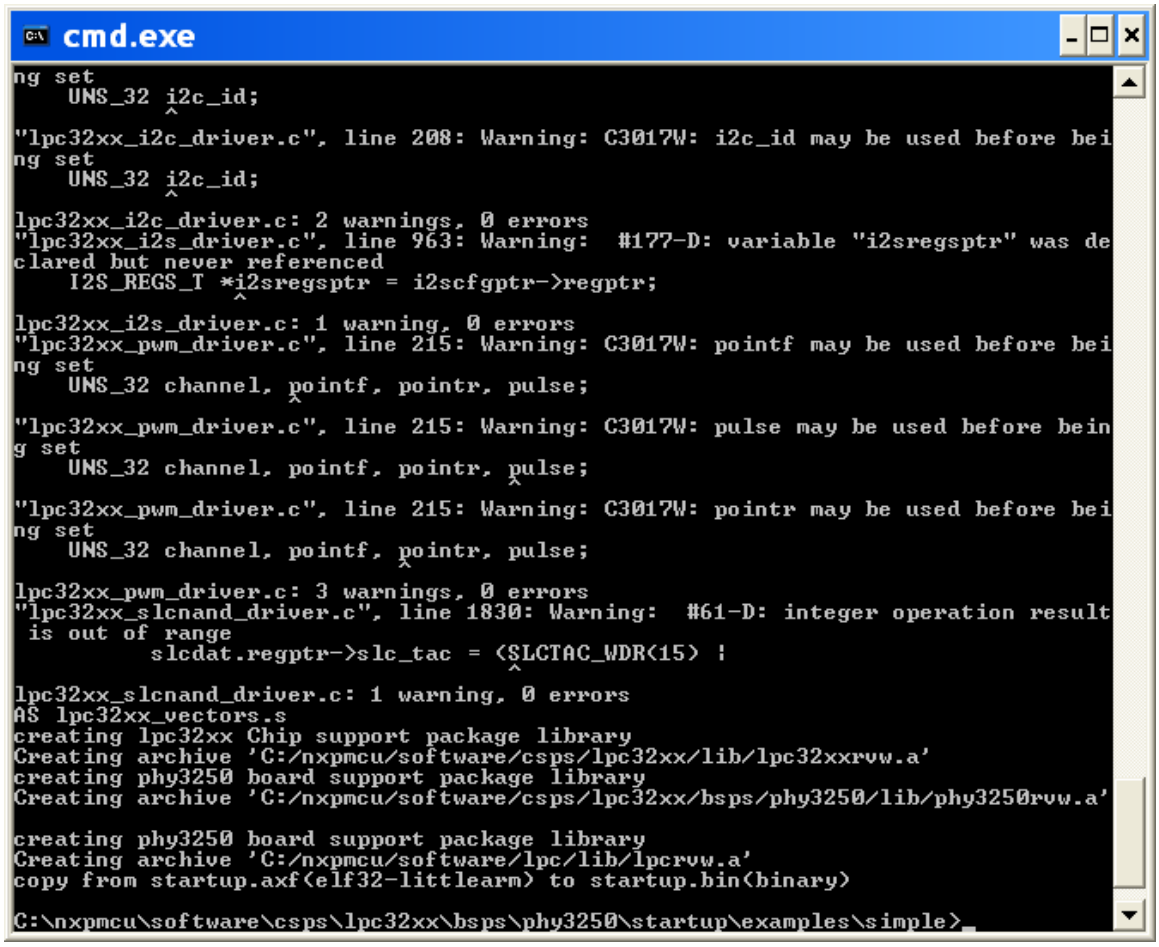
A bootable application is considered an application that is not loaded and executed through S1L. Bootable applications may be a stage 1 application that replaces the S1L, or an application that replaces the kickstart loader. A bootable application may also reside in the serial EEPROM on the SSP or be loaded through the serial port when the LPC3250 is reset. The procedure for booting software is specified in the LPC3250 User's Guide.

Several pre-made startup applications are provided with the BSP as an example of how to build a bootable application. Additionally, all the examples can be setup as bootable applications by simply creating a new project area for the application and copying the code from the example area.

4.2.1 Simple startup example

The *simple* startup example provides a good reference on how to build a bootable application. This example builds into a very small executable image loaded and executed at address 0x00000000 that can replace the kickstart loader or be used as the stage 1 application loaded from the kickstart loader.

To build the simple startup example, open a cmd.exe shell and switch to the software directory. Setup the environment using the setenv command (See Section 4.1.1). Switch to the "./bsps/phy3250/startup/examples/simple" directory. Type make to build the example and the simple.bin image file will be created similar as shown below.



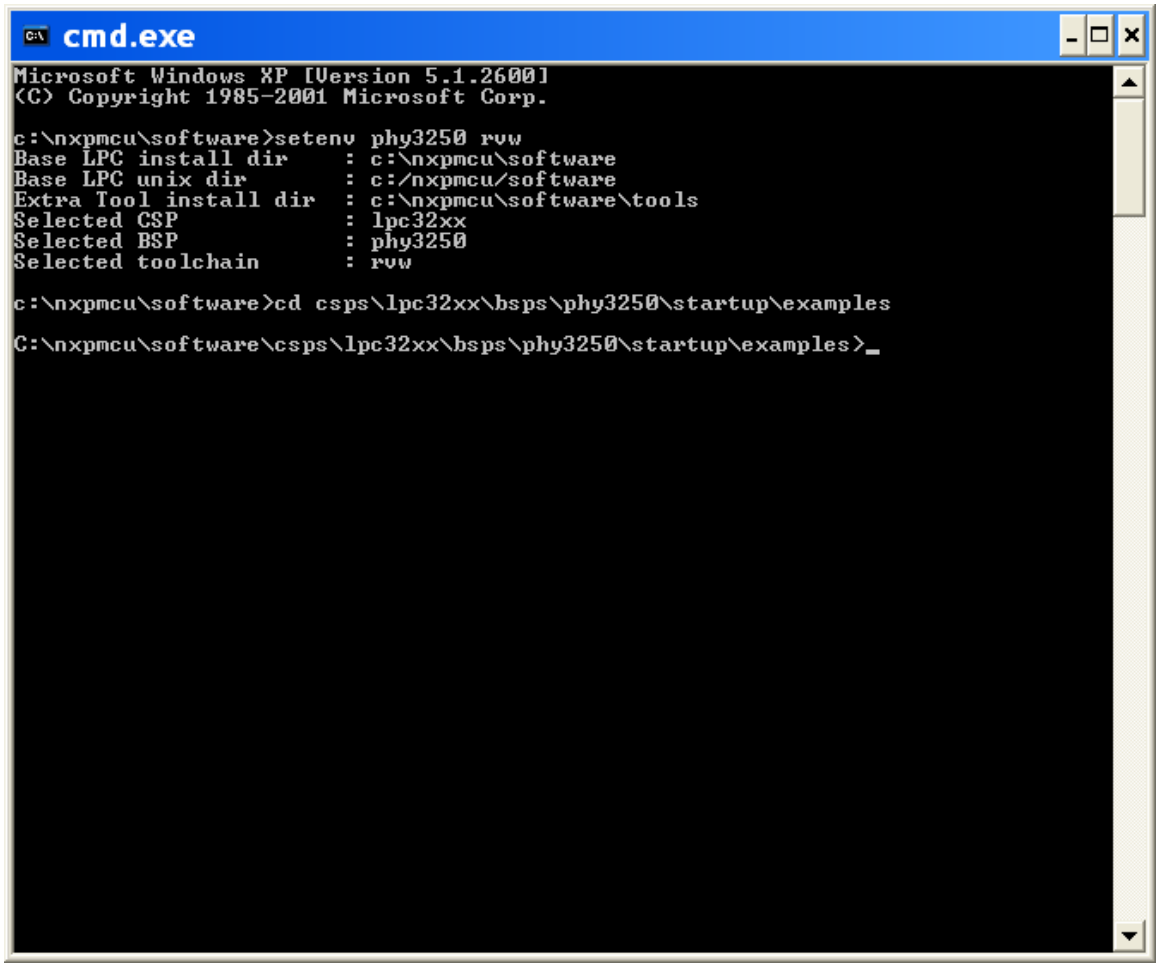
The startup.bin file generated with this build is an executable image that can be burnt into ROM, loaded and executed with the LPC3250 serial loader (LSL) tool, or programmed into NAND FLASH as a replacement ti the kickstart loader or stage 1 application.

4.2.1.1 Making a bootable image with the examples

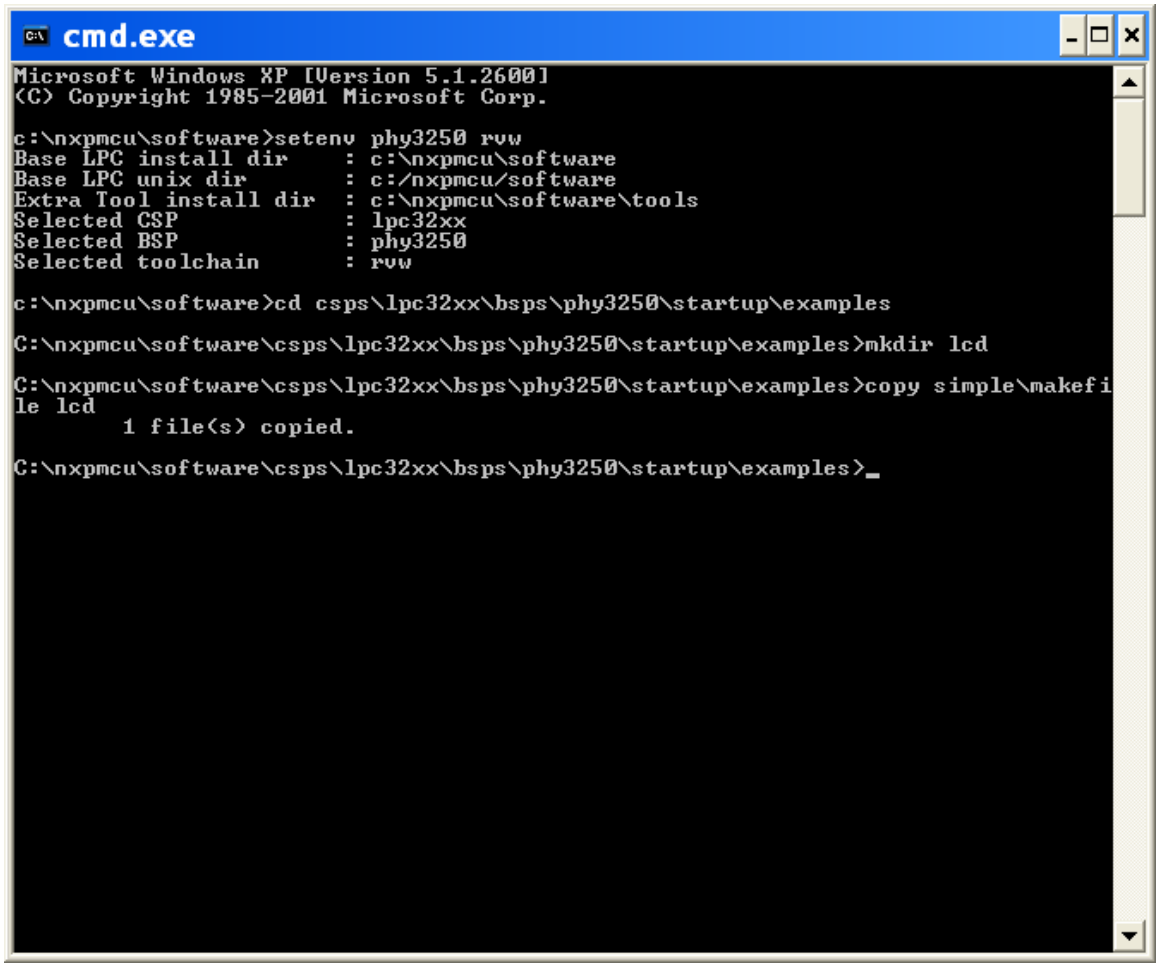
Most of the examples included with the BSP can easily be made into bootable images with the following steps. These example images can be setup to boot when the board is powered on or reset without using S1L.

Be sure you really want to create a bootable application. S1L can easily be setup to automatically load and execute all the examples without the hassles of updating FLASH or trying to restore your board if FLASH is accidentally erased or burnt with a non-functional program. It would be more productive to develop your application with S1L handling system startup, loading, and execution and then binding your application to startup code and booting it once it is near completion.

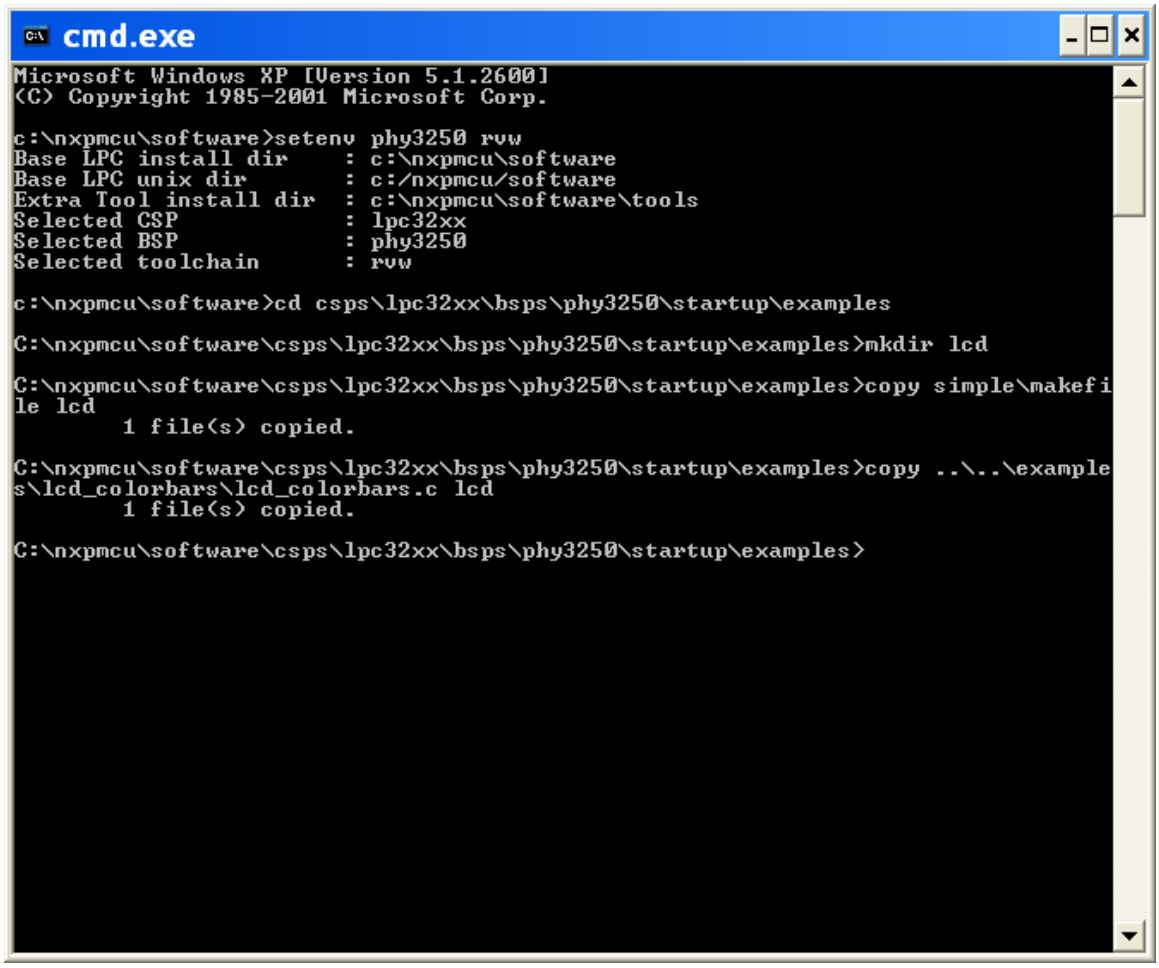
Step 1 : Open a cmd.exe shell and setup the build environment with setenv. Switch to the "./bsps/phy3250/startup/examples" directory.



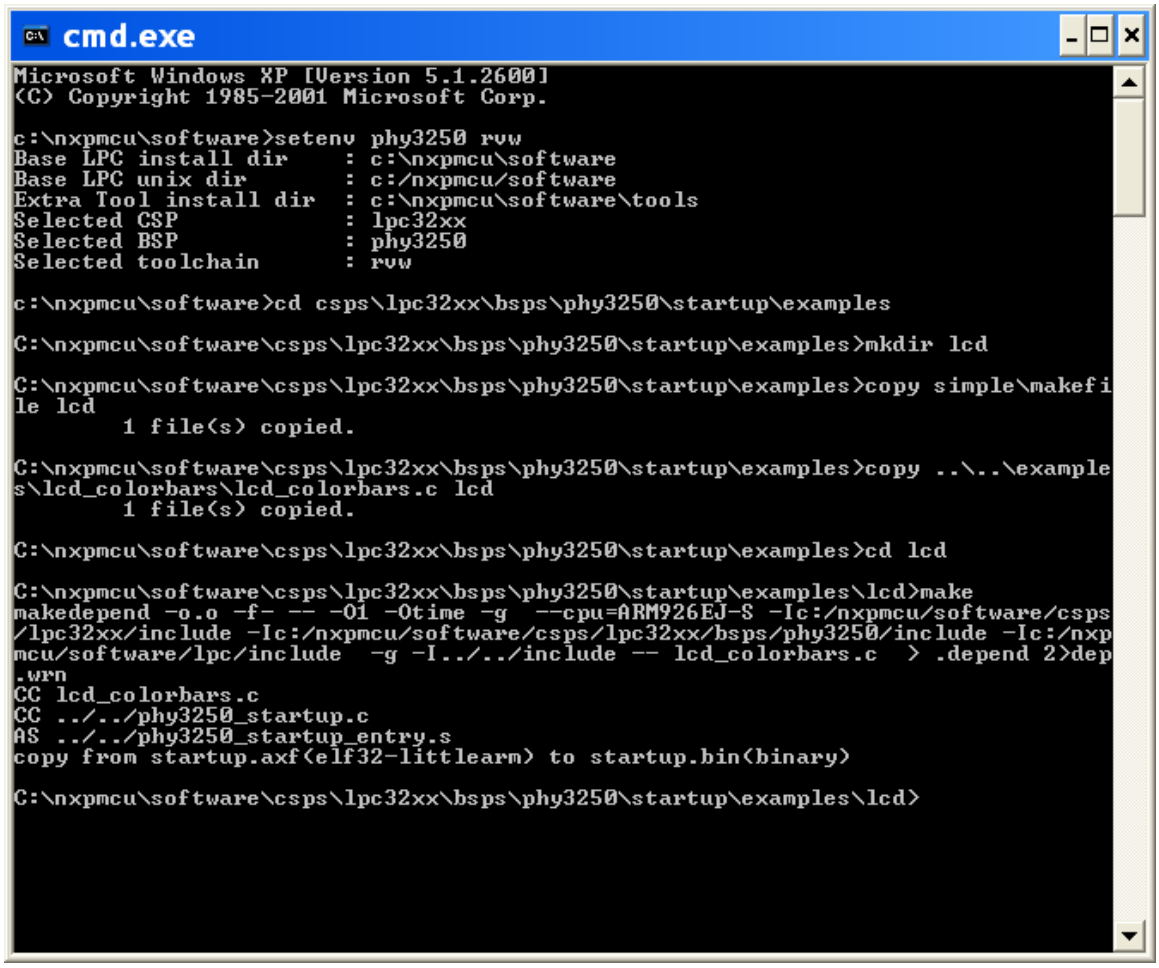
Step 2 : Create a new folder in this area. For the demo, the folder lcd will be created. Copy the makefile file from the simple folder to the new folder.



Step 3 : Add the example from the "./bsps/phy3250/examples" area to the new folder.



Step 4 : Type make in the new directory to build the example as a bootable image.



Congratulations! You've just made a new bootable image from one of the examples. The startup.bin file can be easily setup to boot when the system is reset or powered on.

4.2.2 Deploying a bootable applications

Bootable applications can be booted from a variety of sources and methods. This area explains how to boot applications using the kickstart loader, the LPC3250 NAND FLASH boot capability, and the LPC3250 serial loader (LSL) tool (through the LPC3250 serial boot capability).

4.2.2.1 Deploying as a stage 1 application

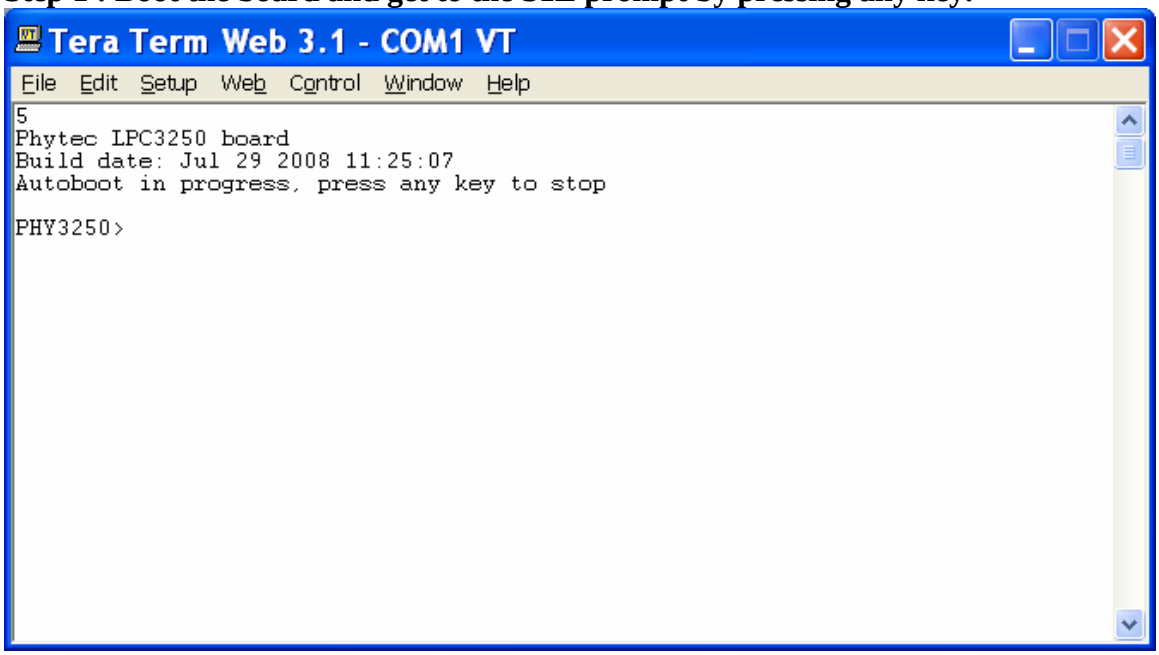
The simple startup example can be deployed as a stage 1 application that is loaded and booted from the kickstart loader. As a stage 1 application, the application image has restrictions on how it is programmed into FLASH for the kickstart loader to use. These restrictions are explained in the phy32xx_bl.doc file.

The stage 1 application will replace S1L and is loaded and executed by the kickstart loader. Although S1L will allow an automatic update of the stage 1 application, the stage 1 application may not be easily updatable after S1L has been erased. For this reason, the

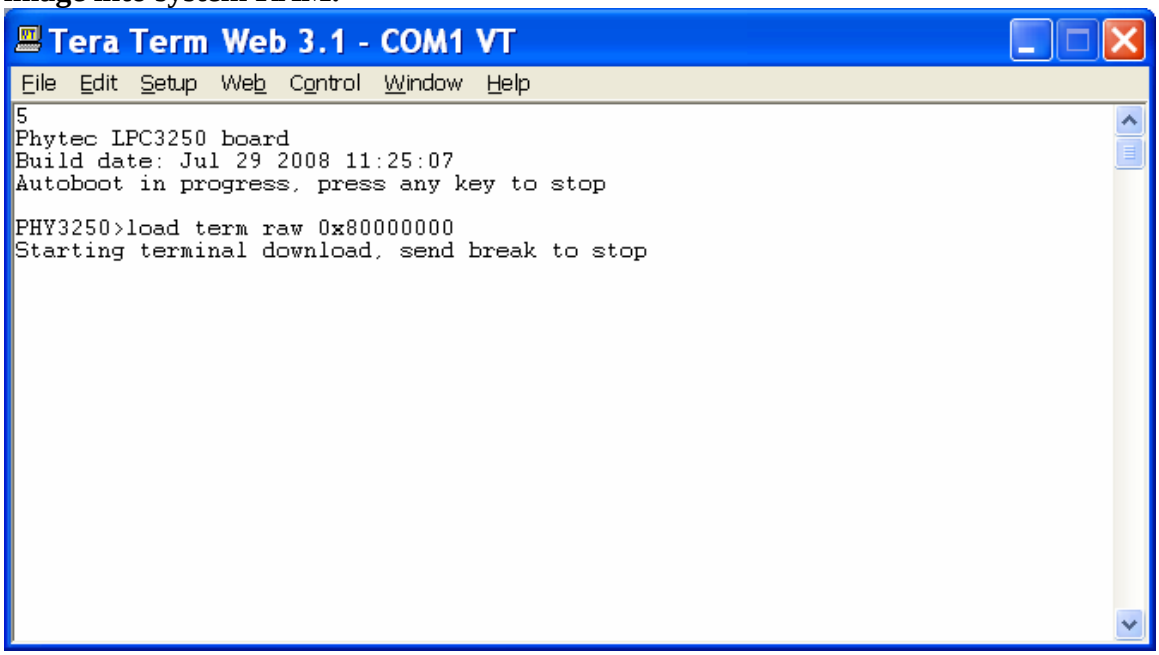
kickstart source code is provided as a reference on how to use the NAND FLASH MLC controller to program NAND FLASH for boot.

These steps assume S1L is currently loaded on the board. If S1L is not the bootable application, follow the steps in Section 0 to restore S1L to your board.

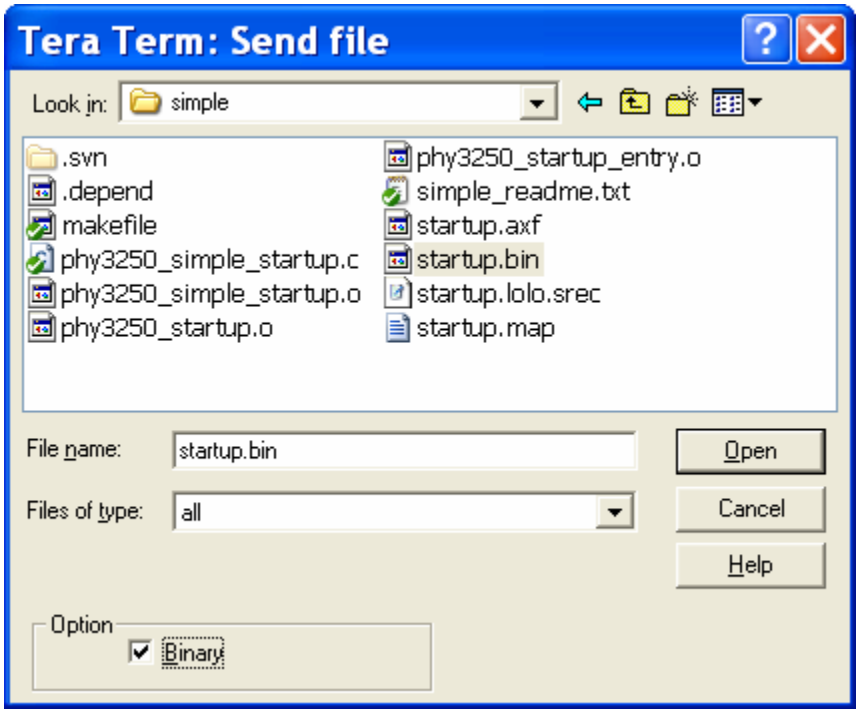
Step 1 : Boot the board and get to the S1L prompt by pressing any key.



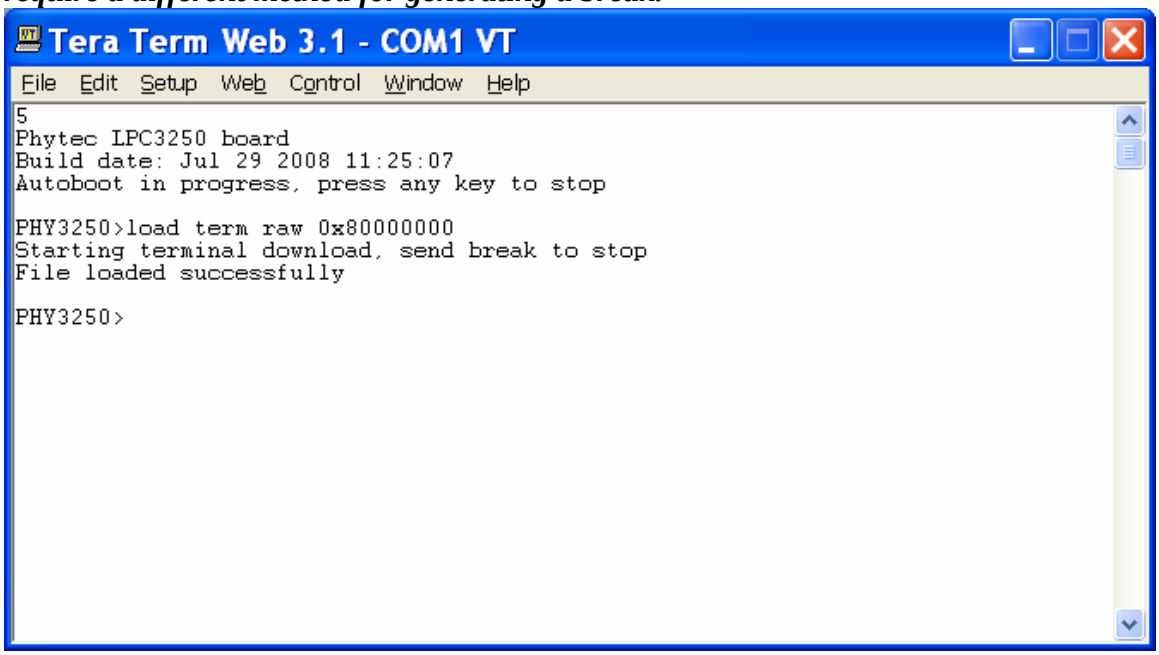
Step 2 : At the S1L prompt, type “load term raw 0x80000000” to load the bootable image into system RAM.



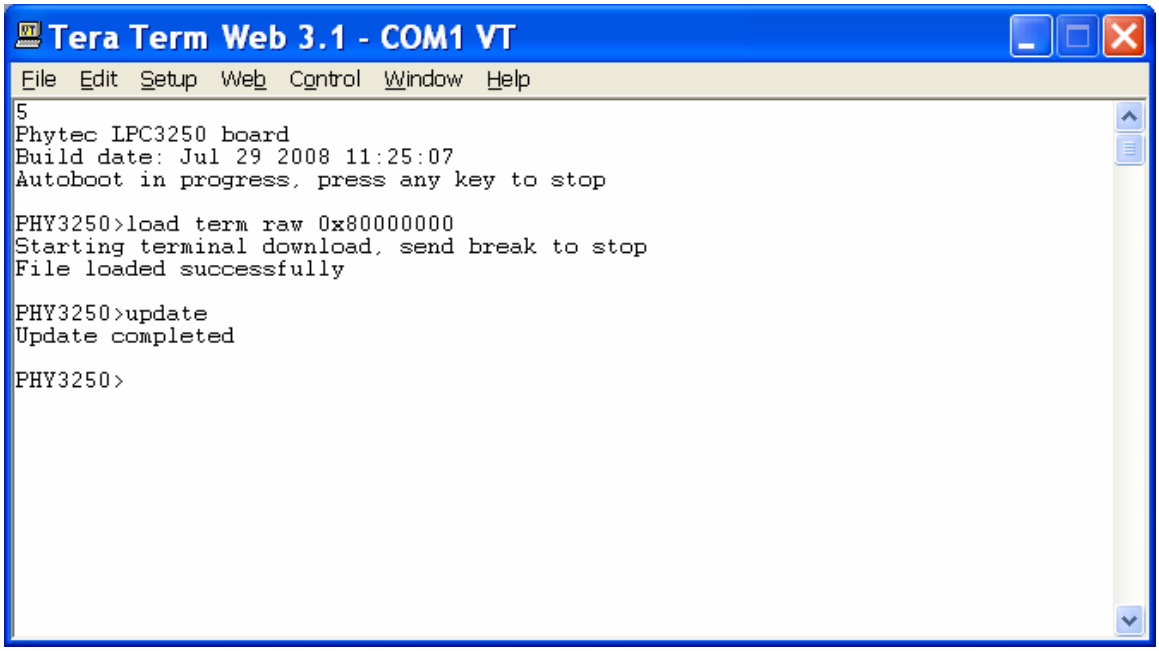
Step 3 : Send the startup.bin from the terminal to S1L as a binary image.



Step 4 : After the file transfer has complete, press ALT-B to send a break to return to the S1L prompt. ***ALT-B only applies to TeraTerm – other terminal programs may require a different method for generating a break.***



Step 5 : Type update to replace S1L with the current loaded application. The next time the board is reset, the application will boot instead of S1L.



4.2.2.2 Deploying as a kickstart application

The simple startup example can be deployed as a direct bootable application replacing the kickstart loader. There are restrictions on this type of boot method as explained in the phy32xx_bl.doc file.

These steps assume S1L is currently loaded on the board. If S1L is not the bootable application, follow the steps in Section 0 to restore S1L to your board.

Steps 1 – 4 : Follow steps 1 to 4 as explained in Section 4.2.2.1.

Step 5 : Type "update kick" to replace the kickstart loader with the current loaded application. The next time the board is reset, the application will boot instead of the kickstart loader.

If your application is greater than 15.5Kbytes in size, the kickstart loader cannot be replaced.

4.2.2.3 Deploying using the LPC3250 serial loader (LSL) tool

The LSL can also be used to load applications into the board. Applications loaded using this method must initialize the board into a known state. In SDRAM is needed, SDRAM must be initialized prior to using it. The startup code included with the BSP is a good place to start for binding you application to board bring up code that can be used with LSL.

Step 1 : Build the simple startup example as explained in Section 4.2.1.

Step 2 : Modify the LPC3250_Loader.ini file as follows, replacing <FILENAME> with the filename and extension of your executable binary file.

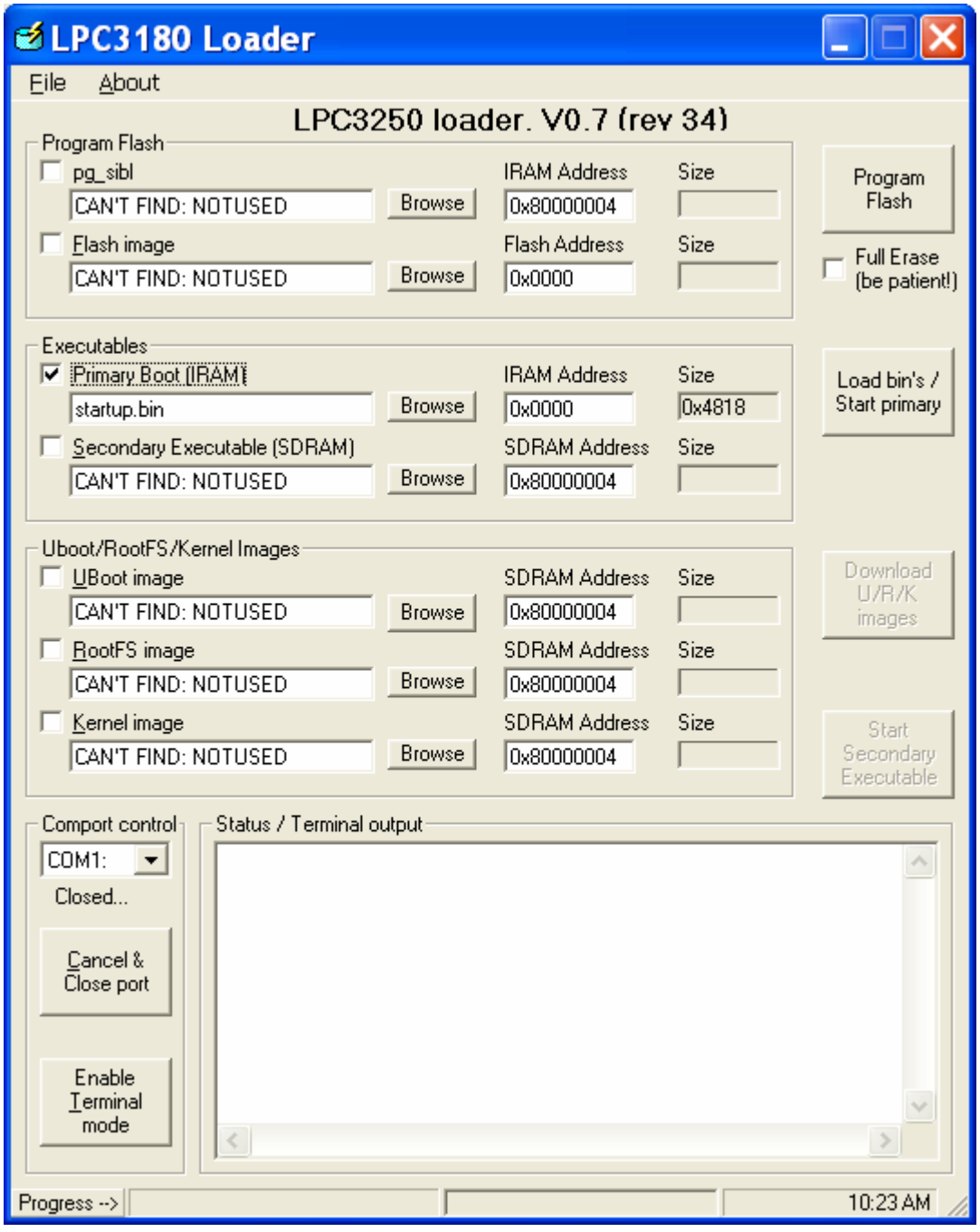
[BINFILES]
PRIMARY=<FILENAME>, 0x00000000
SECONDARY=NOTUSED, 0x00000000
FLASH=NOTUSED, 0x00000000
PG_SIBL=NOTUSED, 0x00000000

[IMGFILES]
UBOOT=NOTUSED, 0x00000000
ROOTFS=NOTUSED, 0x00000000
KERNEL=NOTUSED, 0x00000000

Step 3 : Place your binary file in the same directory as the LSL files (LPC3250_Loader.exe and LPC3250_Loader.ini)

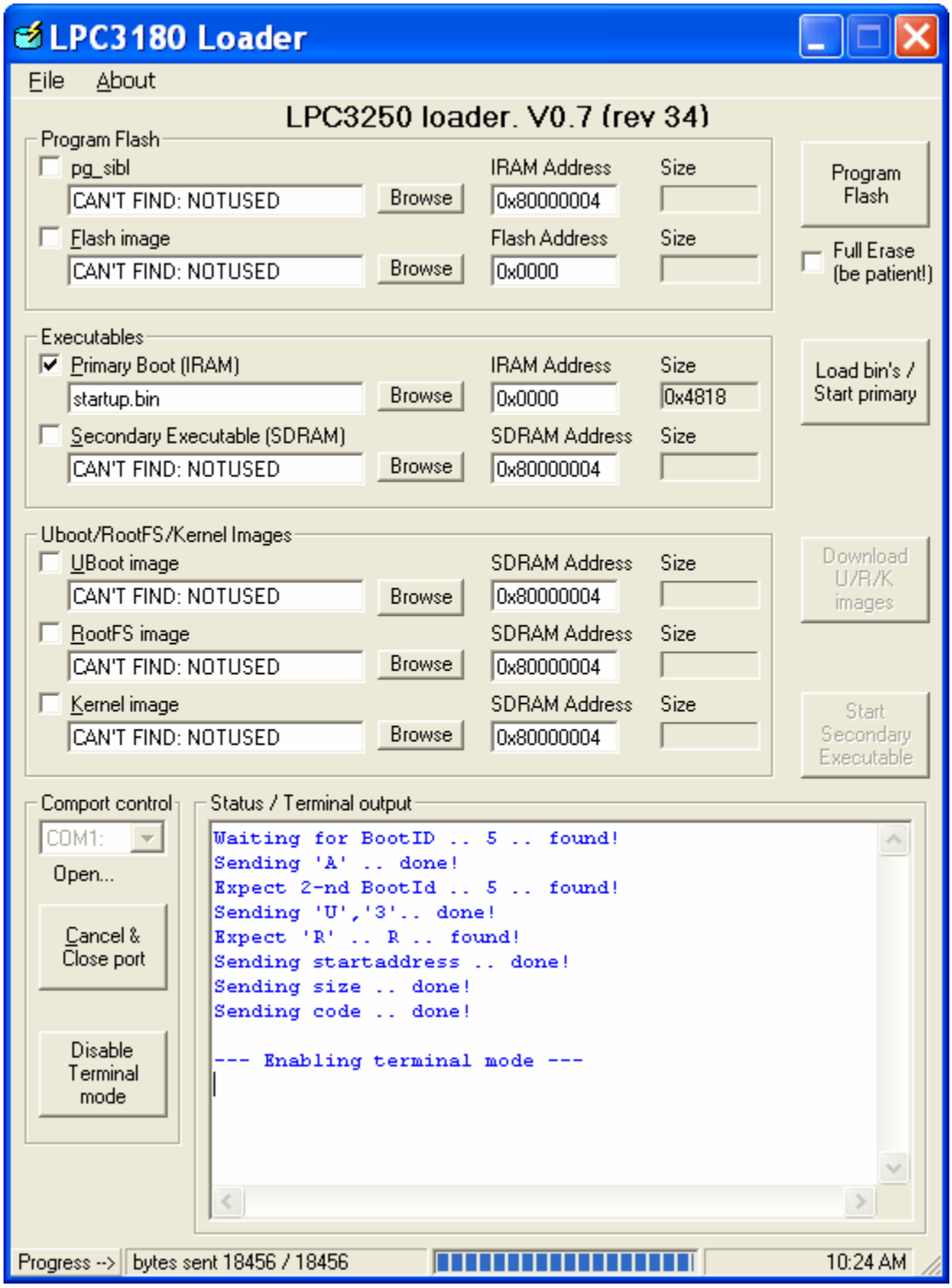
Step 4 : Make sure a serial cable is connected between UART3 on the Phytec board (bottom serial connector) and a PC.

Step 5 : Start LSL and setup the correct COM: port for the PC.



Step 6 : Press the “Load bin/Start primary button” on the LSL Window.

Step 7 : Reset the Phytec board and the image will load and run. After the image has started, the LSL will switch to terminal mode.



4.3 Restoring the kickstart and stage 1 applications

If the boot code stored in NAND FLASH is erased, the board will no longer correctly boot. The original kickstart/S1L image can be restored to the board using LSL and the pre-built binaries included with the BSP. A restoration project included with the BSP

shows how to restore the kickstart and stage 1 loader in a single executable that can be easily loaded and executed with LSL These steps explain how to restore the original kickstart and stage 1 loader application to your board using LSL:

Step 1 : Place the restore.bin file in the same directory as the LSL files (LPC3250_Loader.exe and LPC3250_Loader.ini).

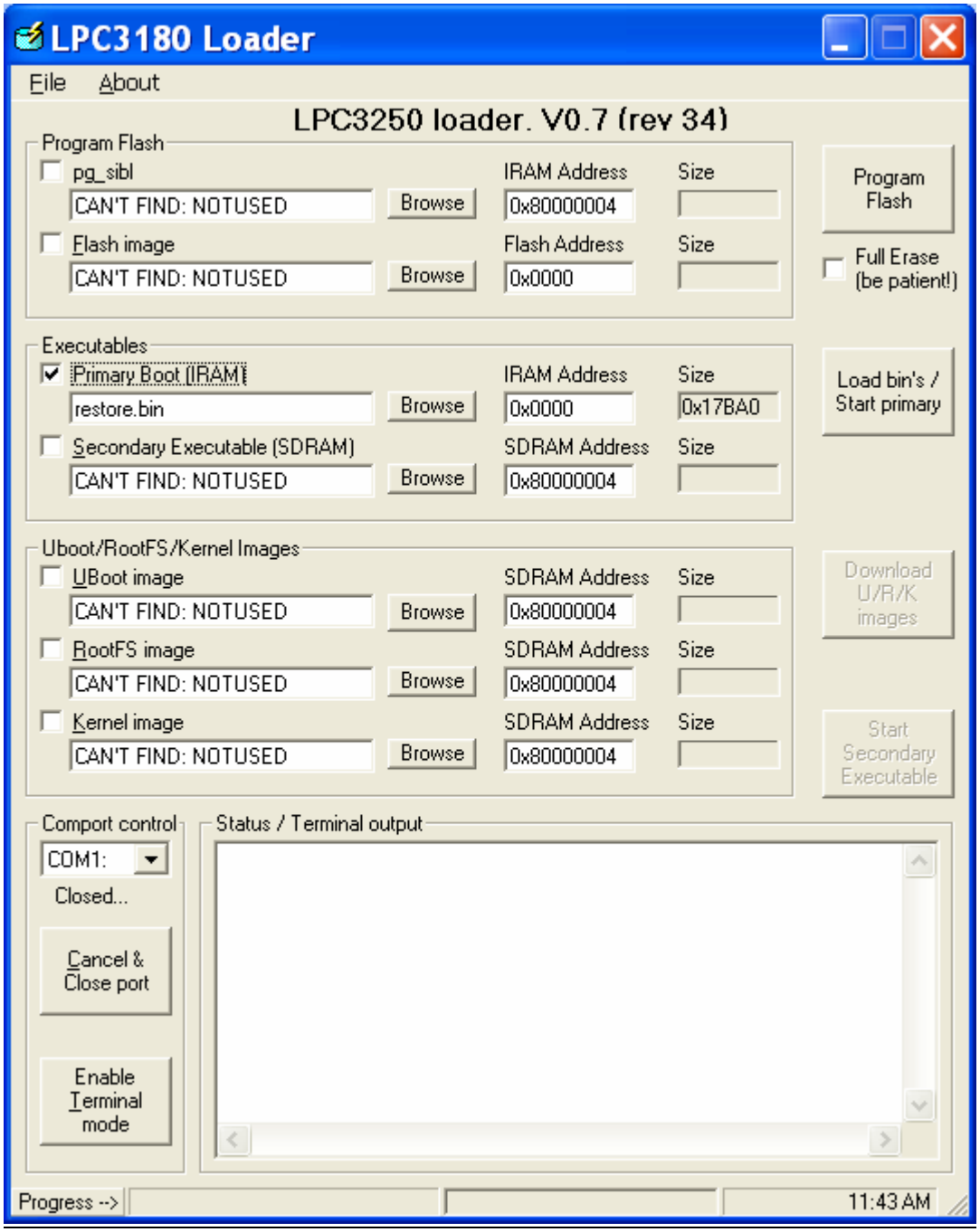
Step 2 : Modify the LPC3250_Loader.ini file as follows.

[BINFILES]
PRIMARY=restore.bin, 0x00000000
SECONDARY=NOTUSED, 0x00000000
FLASH=NOTUSED, 0x00000000
PG_SIBL=NOTUSED, 0x00000000

[IMGFILES]
UBOOT=NOTUSED, 0x00000000
ROOTFS=NOTUSED, 0x00000000
KERNEL=NOTUSED, 0x00000000

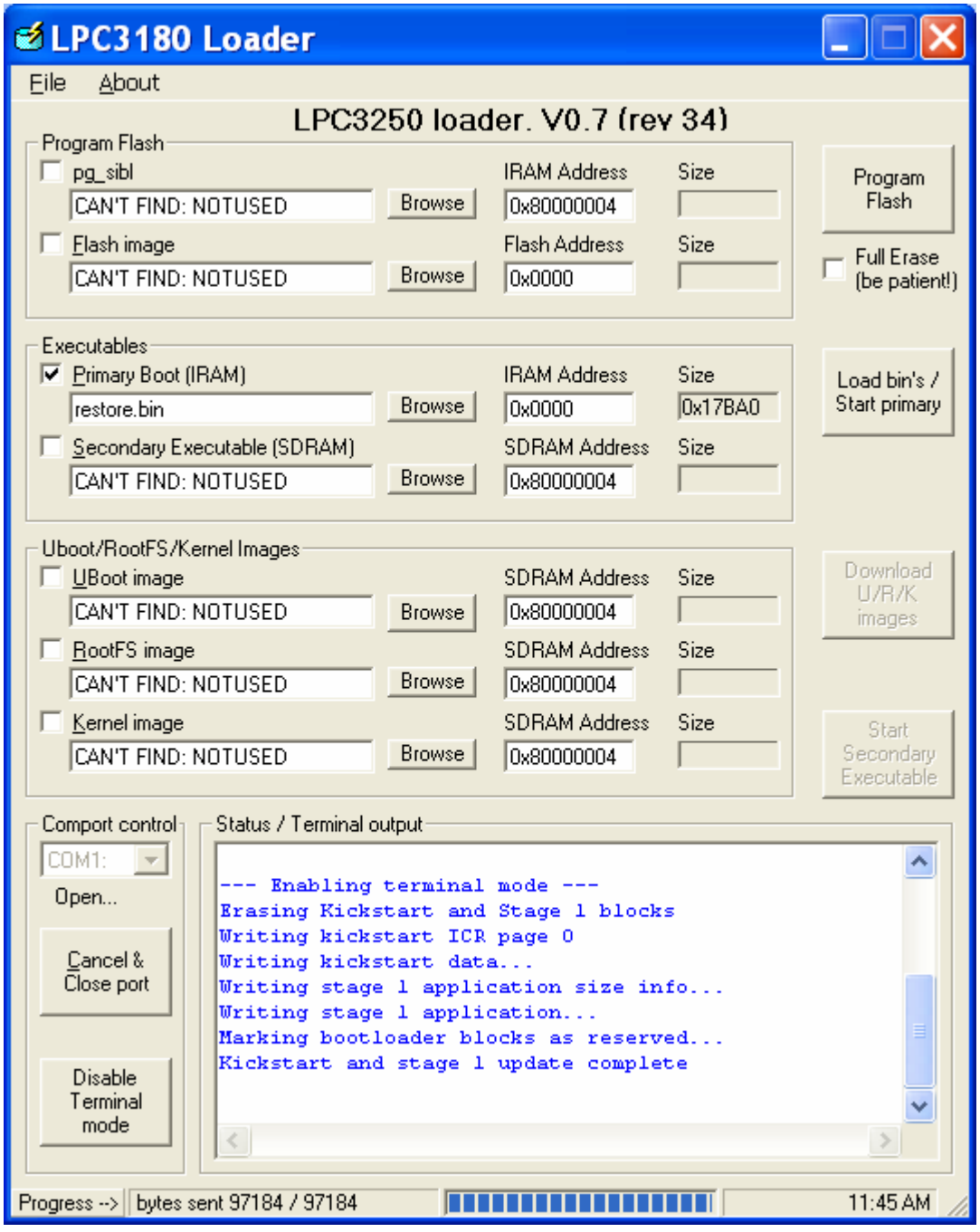
Step 3 : Make sure a serial cable is connected between UART3 on the Phytec board (bottom serial connector) and a PC.

Step 4 : Start LSL and setup the correct COM: port for the PC.

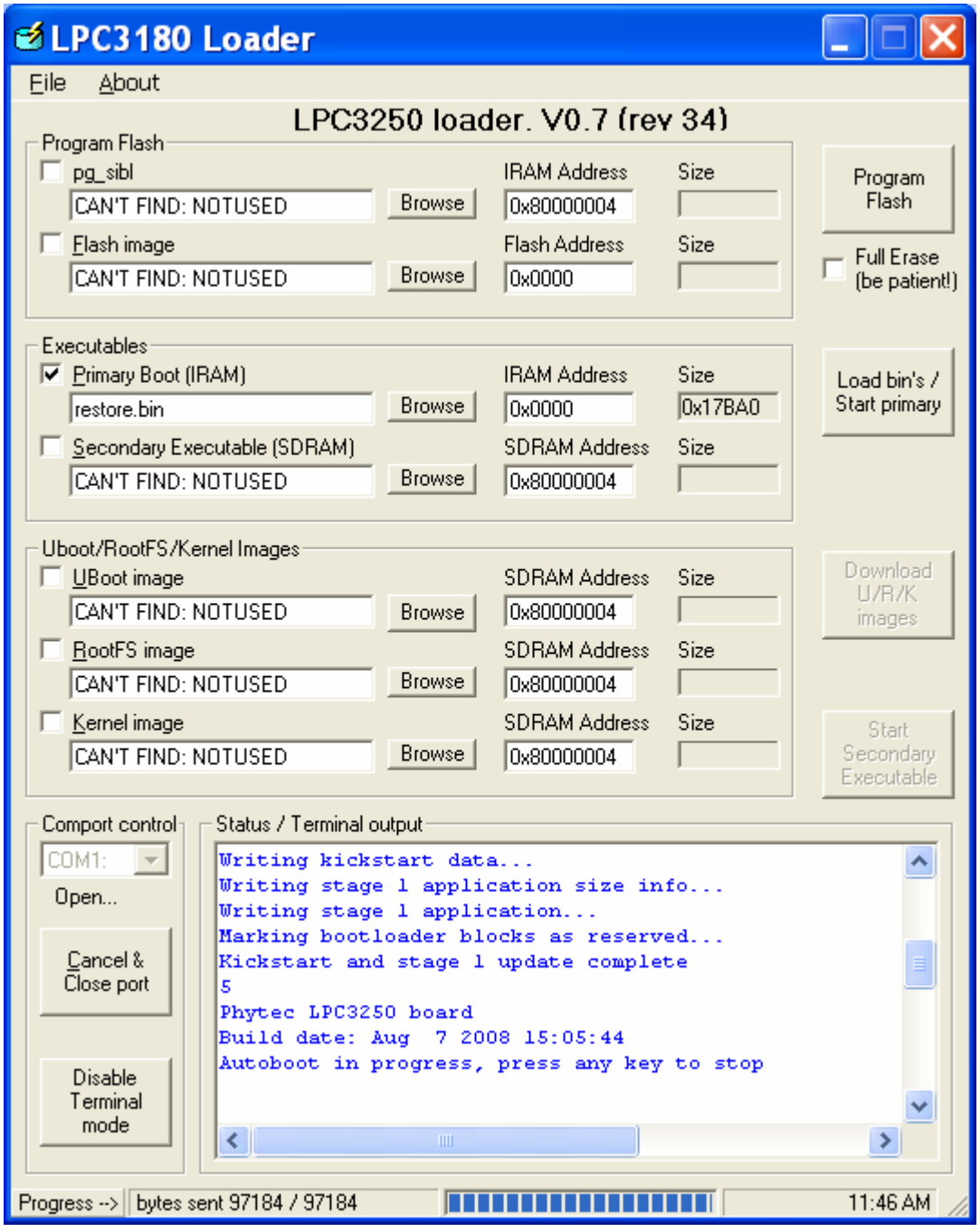


Step 5 : Press the “Load bin/Start primary button” on the LSL Window.

Step 6 : Reset the Phytec board and the image will load and run. After the image has started, the LSL will switch to terminal mode.



Step 7 : Reset the Phytec board. The S1L message header and prompt should appear.



4.3.1 Using the restoration project to FLASH custom applications

The code used to restore the kickstart loader and stage 1 loader applications through LSL is provided in the .\startup\examples\ Board restoration images\ Restoration executable directory. This project builds a single executable with the kickstart and stage 1 loader images in it and burns them into FLASH. *This project currently only works with ARM Realview and Keil uVision3 toolchains.*

Custom applications can be setup for the kickstart and/or stage 1 areas in FLASH by adding the custom binaries to the directory and editing the images.s file. Replace the kickstart.nim and stage1.bin strings in the INCBIN lines to include them in the build. The restore application can then be loaded and executed through LSL to burn the customer images.

```
__ks_start
    INCBIN ..\kickstart.bin
__ks_end
__s1l_start
    INCBIN ..\stage1.bin
__s1l_end
```

5 Miscellaneous

5.1 Where to get software

The latest Phytec 3250 BSP and LPC3250 CSP are available from NXP's website.

TeraTerm is free and can be downloaded from a number of sources such as SourceForge and tucows.

CodeSourcery GNU can be downloaded from www.codesourcery.com.

5.2 Using the LPC3250 serial loader (LSL) tool

The LSL is meant to be used on various LPC devices and all of it's functionality may not apply to the LPC3250 or the Phytec 3250 board. *As of this time, only the primary boot (IRAM) option is supported with the LSL.*

Using the LSL requires setting up the primary boot executable filename. The image must always be built to load and execute starting at address 0x00000000. Once LSL is setup, the Load bin's/Start primary button can be pressed and the LSL will wait for a character from the LPC3250 boot ROM on reset.

The boot sequence is as follows:

LPC3250	LPC3250 serial loader
LPC3250 is reset	
LPC3250 sends a '5' on UART3.	LSL responds with a 'A'.
LPC3250 sends a '5' on UART3.	LSL responds with a 'U'.
	LSL sends the address to the board (0x00000000)
	LSL sends the image size to the board in bytes
	LSL sends the code to the board
LPC3250 boots loaded code at address 0x00000000	

5.3 Issues

5.3.1 IAR toolchain

Examples built with IAR and executed through S1L will not return to the S1L prompt once execution is complete. This is because the system resources shared between S1L and the examples (stacks) are modified by the IAR example. For IAR examples, press reset on the board to return to the S1L prompt.

Startup code examples will not currently work with IAR.