Linux
BSP (Board Support Package) Guide for the Diolan LPC4350-DB1

Cyril Fougeray

July 12, 2014

This document is largely based on the Emcraft Systems work on the Hitex LPC4350 Eval Board.
See http://www.emcraft.com/products/232

Contents

1 Overview

This document is a Linux LPC43XX BSP (Board Support Package) Guide for the Diolan LPC4350-DB1 board.
The BSP provides a software development environment for evaluation and development of Linux on the Cortex-M4 processor core of the NXP LPC43XX microcontroller using the Diolan LPC4350-DB1 board as a hardware platform.

2 Product contents

You need to purchase the Diolan LPC4350-DB1 board from Diolan, Arrow or their distributors.

2.1 Downloadable software materials

The following software materials are available for download from my *Github* :
`https://github.com/fouge/u-boot`.

1. **u-boot-dfu.bin** - prebuilt U-Boot file in the format suitable for installation into Flash or an execution from RAM if loaded via DFU (see `http://www.nxp.com/documents/user_manual/UM10503.pdf`, Chapter 5) on the Diolan LPC4350-DB1 board ;
2. **u-boot-not-dfu-usb.bin** - prebuilt U-Boot file in the format suitable for installation into Flash on the Diolan LPC4350-DB1 board ;
   - This U-Boot image is not loadable on the board using DFU and the console is accessible through the USB interface by default and not the serial interface.
3. **networking.uImage** - prebuilt Linux image, ready to be loaded to the board (from Emcraft Systems).
4. **Source code** - ready to be compiled with the options you want.

3 Software functionality

The following list summarizes the features and capabilities of Linux LPC43XX, Release 1.12.0:

- U-Boot firmware :
  - U-Boot v2010.03;
  - Target initialization from power-on / reset;
  - Runs from the internal eNVM and internal SRAM (no external memory required for standalone operation);
  - Serial & USB console;
  - Ethernet driver for loading images to the target;
  - Serial driver for loading images to the target;
  - Device driver for built-in Flash (eNVM) and self-upgrade capability;
  - Device driver for storing environment and Linux images in external Flash;
  - Autoboot feature, allowing boot of OS images from Flash or other storage with no operator intervention;

  - Persistent environment in Flash for customization of target operation;
  - Sophisticated command interface for maintenance and development of the target.

- Linux :
  - uClinux kernel v2.6.33;
  - Boot from compressed and uncompressed images;
  - Ability to run critical kernel code from integrated Flash of LPC43XX;
  - Serial device driver and Linux console;
  - Ethernet device driver and networking (ping, NFS, Telnet, FTP, ntpd, etc.);
  - busybox v1.17;
  - POSIX pthreads;
  - Hardened exception handling; an exception triggered by a process affects only the offending process;
  - Support for the hardware FPU;
  - Web server;
  - MTD-based Flash partitioning and persistent JFFS2 Flash file system for external Flash;
  - I2C device driver;
  - SPI controller master-mode device driver

NB. I didn't test all these functionalities.

3.1 New and changed features

There are 2 versions of U-Boot with different functionnalities :

1. u-boot-dfu.bin : loadable via DFU and accessible through the serial interface (see 4.2)
2. u-boot-not-dfu-usb.bin : not loadable via DFU, working only from Flash (must be loaded at address 0x1C000000) and accessible through USB (*/dev/ttyACMx* on Linux).

Note that uCLinux is only accessible through the serial port.

3.2 Known issues

Please go to `https://github.com/fouge/u-boot/issues` for an updated list of all known issues.

4 Hardware setup

4.1 Jumpers

Boot mode	P2 9	P2 8	P1 2	P1 1	Description
EMC 16-bit	LOW	LOW	HIGH	HIGH	Boot from external static memory (such as NOR flash) using CS0 and a 16-bit data bus.
USB0	LOW	HIGH	LOW	HIGH	Boot from USB0 (DFU mode).

Table 1: Useful boot modes

4.2 Connecting the serial interface

As configured, the PF.10 pin is used for TXD and the PF.11 pin is used for RXD (USART0). The baudrate is set to 115200, 1N8.

To wire these pins, see http://www.diolan.com/downloads/lpc4350-db1-schematics.pdf, page 2 (U0_TXD and U0_RXD). For a more visual overview, see below :

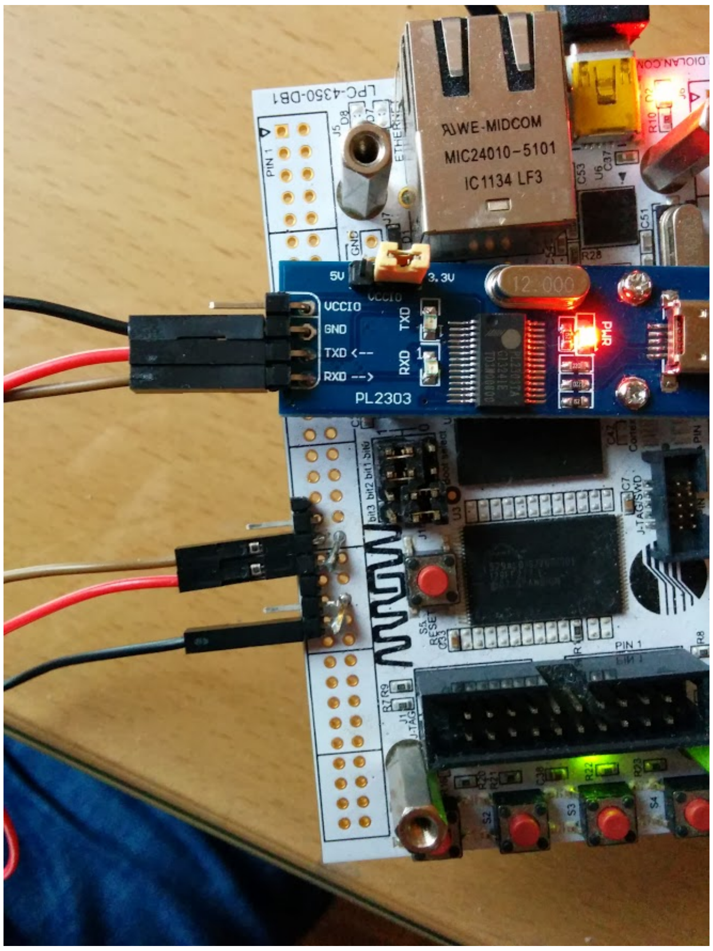


Figure 1: Serial wiring - USART0

4.3 Ethernet

In order to load the Linux image, connect the Ethernet port to your computer.

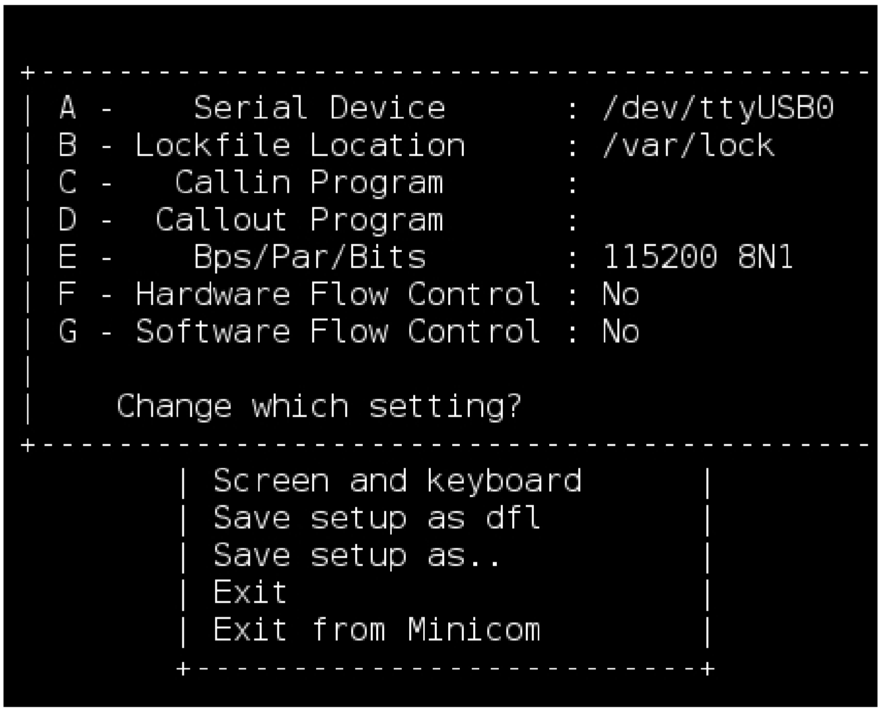
5 Linux software Set-up

5.1 U-Boot installation

To install U-Boot onto the Diolan LPC4350-DB1 board, follow the step-wise procedure documented below :

1. Configure the boot select jumpers for booting from USB0 (see 4.1).
2. Connect the UART port of the board (see 4.2) to your host computer.
3. Run a terminal program (e.g *HyperTerminal* on Windows or *minicom* on Linux). Depending if you are using a Serial to USB converter or not, configure the terminal

program accordingly. Here is my configuration using *minicom* (`$ sudo minicom -s`) :



```
+-----------------------------------------
| A -    Serial Device      : /dev/ttyUSB0
| B - Lockfile Location     : /var/lock
| C -   Callin Program      :
| D -  Callout Program      :
| E -    Bps/Par/Bits       : 115200 8N1
| F - Hardware Flow Control : No
| G - Software Flow Control : No
|
|    Change which setting?
+-----------------------------------------
        | Screen and keyboard      |
        | Save setup as dfl        |
        | Save setup as..          |
        | Exit                     |
        | Exit from Minicom        |
        +--------------------------+
```

Figure 2: Minicom Setup

4. Connect the board to a Linux host by plugging a USB mini-B cable into the USB connector of the board.

5. Install the free tool *dfu-util* on the Linux host (go to http://wiki.openmoko.org/wiki/Dfu-util for more information).

6. Boot U-Boot on the board by running *dfu-util* on the Linux host and observe the U-Boot prompt on the terminal program connected to the LPC4350-DB1 UART.

```
$ dfu-util -R -D u-boot-dfu.bin
dfu-util 0.5

(C) 2005-2008 by Weston Schmidt, Harald Welte and OpenMoko Inc.
(C) 2010-2011 Tormod Volden (DfuSe support)
This program is Free Software and has ABSOLUTELY NO WARRANTY

dfu-util does currently only support DFU version 1.0

Opening DFU USB device... ID 1fc9:000c
Run-time device DFU version 0100
Claiming USB DFU Runtime Interface...
Determining device status: state = dfuIDLE, status = 0
WARNING: Runtime device already in DFU state ?!?
Found Runtime: [1fc9:000c] devnum=0, cfg=1, intf=0, alt=0, name="DFU"
Claiming USB DFU Interface...
Setting Alternate Setting #0 ...
Determining device status: state = dfuIDLE, status = 0
dfuIDLE, continuing
```

```
DFU mode device DFU version 0100
Device returned transfer size 2048
No valid DFU suffix signature
Warning: File has no DFU suffix
bytes_per_hash=1884
Copying data from PC to DFU device
Starting download: [##################################################] finished!
unable to read DFU status
can't detach
Resetting USB to switch back to runtime mode
$
```

7. Start a TFTP server in the local network of the Diolan LPC4350-DB1 board and make the *u-boot-dfu.bin* image available on this TFTP server (this tutorial seems great for configuring your TFTP server: https://linuxlink.timesys.com/docs/linux_tftp).

8. Set the U-Boot environment variables *serverip* and *ipaddr* to the IP address of the TFTP server and the desired IP address of the Diolan board, respectively.

   ```
   Diolan-LPC4350-DB1> setenv ipaddr 192.168.0.5
   Diolan-LPC4350-DB1> setenv serverip 192.168.0.4
   ```

   Then, set the environment variables *flashaddr* and *loadaddr*.

   ```
   Diolan-LPC4350-DB1> setenv flashaddr 1C000000
   Diolan-LPC4350-DB1> setenv loadaddr 28000000
   ```

   Now, we have to configure the board to load the right image (you can load *u-boot-dfu.bin* or *u-boot-not-dfu-usb.bin*)

   ```
   Diolan-LPC4350-DB1> setenv image u-boot-dfu.bin
   ```

   You can then verify the environment variables using the *printenv* command.

9. Download the U-Boot image into the on-board RAM and copy it into the on-board Flash using a single command (`> run update`):

   ```
   Diolan-LPC4350-DB1> run update
   Auto-negotiation...completed.
   LPC18XX_MAC: link UP (100/Full)
   Using LPC18XX_MAC device
   TFTP from server 192.168.0.4; our IP address is 192.168.0.5
   Filename 'u-boot-dfu.bin'.
   Load address: 0x28000000
   Loading: ###################
   done
   Bytes transferred = 94224 (17010 hex)
   Un-Protected 2 sectors

   .. done
   Erased 2 sectors
   Copy to Flash... done
   Diolan-LPC4350-DB1>
   ```

10. Reconfigure the boot select jumpers according to the EMC 16-bit mode (see 4.1).

11. Reboot the board and see the U-Boot start-up banner in the terminal program. If you loaded the non-DFU USB image, you need to run another instance of *minicom* and configure it to read from the right USB port (*/dev/ttyACM0* for example).

    ```
    U-Boot 2013.06 (juil. 08 2014 - 14:46:11)

    CPU  : LPC43xx series (Cortex-M4/M0)
    Freqs: SYSTICK=204MHz,CCLK=204MHz
    Board: Diolan LPC4350-DB1 rev 1
    DRAM:   8 MB
    Flash:  2 MB
    In:    serial
    Out:   serial
    Err:   serial
    Net:   LPC18XX_MAC
    Hit any key to stop autoboot:  0
    Diolan-LPC4350-DB1>
    ```

You now have U-Boot running from Flash.

5.2 Linux execution

The Diolan LPC4350-DB1 doesn't have enough Flash to have Linux installed on-board but you still can execute Linux.
First, you have to configure the board as in the previous step in order to load the Linux image using TFTP. Then, set the name of the image to load. In our case, it's "networking.uImage".

```
Diolan-LPC4350-DB1> setenv image networking.uImage
```

Once you have the board ready to load the Linux image, save the environment variables in flash :

```
Diolan-LPC4350-DB1> saveenv
```

Then we will start Linux. Make sure the image is available on your TFTP directory and execute the following command :

```
Diolan-LPC4350-DB1> run netboot
Auto-negotiation...completed.
LPC18XX_MAC: link UP (100/Full)
Using LPC18XX_MAC device
TFTP from server 192.168.0.4; our IP address is 192.168.0.5
Filename 'networking.uImage'.
Load address: 0x28000000
Loading: #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         ####################
done
Bytes transferred = 2098624 (2005c0 hex)
```

```
## Booting kernel from Legacy Image at 28000000 ...
   Image Name:   Linux-2.6.33-arm1
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    2098560 Bytes =  2 MB
   Load Address: 28008000
   Entry Point:  28008001
   Verifying Checksum ... OK
   Loading Kernel Image ... OK
OK

Starting kerne l ...
```

You can run Linux everytime you boot using this method. To do so, modify the *bootcmd* environment variable to “run netboot”. Obviously, your board must be connected to your host each time you reboot.

5.3 Using Telnet

In the uClinux console, you need to turn on the telnet deamon executing the command telnetd :

```
~ # ./bin/telnetd
```

Then, from your host, you can connect to the remote Linux using `$ telnet 192.168.xxx.xxx`. In our case :

```
$ telnet 192.168.0.5
Trying 192.168.0.5...
Connected to 192.168.0.5.
Escape character is '^]'.

lpc4350-db1 login: root
Password:
~ #
```

Login is "root", there is no password so keep the field empty and type enter. You are now connected to your board !

6 To go further

6.1 Compiling U-Boot

If you need to compile U-Boot from source, you will have to configure your distribution in order to set-up cross-compilation. I used to work with the LPCXpresso provided toolchain :

```
$ export CROSS_COMPILE=/usr/local/lpcxpresso_6.1.0_164/lpcxpresso/tools/bin
/arm-none-eabi-
```

Then, you should be able to compile :

```
$ cd .../u-boot/
$ make lpc4350-db1_config
$ make
```

6.2 USB or Serial interface

If you installed *u-boot-not-dfu-usb.bin*, you can configure U-Boot to get the console through the serial interface setting up environment variables (in this order if you don't want to get into trouble) “stderr”, “stdout” and finally “stdin” to “serial”. The console should now pop up from the serial interface. Execute “saveenv” if you want to save this configuration.

6.3 Support

I welcome any and all feedback, comments, or help : cyril.fougeray@gmail.com