PSAS-LPC2148 Development Setup Notes/Scripts

Table of Contents

Introduction

This is another way to begin setting up for developing on the LPC2148. The context is primarily for developing LPC embedded software in the PSAS group, but it could be changed for other efforts.

This document assumes the user is involved with multiple projects, and is using a Linux-based OS. (BSD and other-nix may also work, they haven't been tested.)

It is also expected that the user understands shell scripting, and has no fear about editing scripts. Everything is changeable.

This setup has been tested on a 32Bit Debian Linux OS and a 64 bit Fedora Linux OS.

Versions

Source	Version

GCC_VERSION	4.2.1
BINUTILS_VERSION	2.19
GDB_VERSION	6.6
EXPAT_VERSION	Debian package
NEWLIB_VERSION	1.17.0
libftd2xx0	libftd2xx0.4.13
Openocd OOCD_VERSION	1257
Eclipse	N/A

Not all versions of these tools play nice together. This set seems to work OK.

Grab the development library packages

Install the Debian packages necessary to build the tools:

```
 # aptitude install libgmp3-dev libmpfr-dev subversion make autoconf
automake
```

Get the scripts

Use git:

(Reference: http://psas.pdx.edu/git/)

```
 $ git clone git://git.psas.pdx.edu/git/lpc-kit
```

Find the lpc-kit/Tools directory. Change the top level script PREFIX to reflect where you cloned the git repository.

They are .sh scripts, attack them with your favorite text editor.

Find the following scripts in the lpc-kit/Tools directory:

psas_lpc.sh –

starts psas_build.sh and logs the output to file(s). Run in the background and tail the logfile.

(e.g. psas_build.log). Runs psas_build.sh, psas_ftdilib.sh, psas_ocd.sh.

psas_build.sh –

Attempts to download and build software packages. Installs in /opt/cross directory. Generally called from psas_lpc.sh.

Download software sources. (GCC, binutils, expat, gdb, newlib, arm cross compile configuration.)

Verify gpg key for sources if available.

Builds: binutils, gcc, newlib, gcc (gcc), expat, gdb

Don't expect this all to work first time. If you are lucky, the script will exit nicely and tell you what broke. Also, this could take an hour or two to download/compile everything...

psas_ftdilib.sh –

Installs libs to support OpenOcd. This script **DOES MESS WITH SYSTEM DIRECTORIES**: /usr/local/include, /usr/local/lib, /usr/lib, /etc/fstab, /etc/ld.so.conf. And maybe more. It is the most ***irresponsible*** script in the set. Use at your own risk. If in doubt, just read the script to see what it does. (Uses sudo, you will need sudo authorization. Try 'man sudo' for more information.)

Command line args:

USAGE: psas_ftdilib.sh (-i | -u) [-h]

The arguments to use are:

-h: long help message

-i: Attempt an install of openOCD

-u: Attempt an uninstall of openOCD

type `psas_ftdilib.sh -h` for long help.

See also: http://psas.pdx.edu/OlimexLPC2148Setup/

This script does attempt to undo any changes it made if you run it with the -u switch.

More information:

http://www.ftdichip.com/Drivers/D2XX.htm

psas_ocd.sh -

This script downloads A VERSION (NOT THE LATEST VERSION, EDIT THE SCRIPT TO USE A DIFFERENT VERSION OR LATEST VERSION) of the open ocd software. Then it runs a set of commands to build. More information:

http://openocd.berlios.de/web/

http://openfacts.berlios.de/index-en.phtml?title=Open_On-Chip_Debugger

openocd is managed with subversion version control software.

psas_lpc21isp.sh -

lpc21isp: Program the 2148 (and others) using the serial port.

Download the source by hand. Put it in: lpc-kit/Sources/. The git repository may already have a version, you may want to check the directory first.

Get the source here:

http://tech.groups.yahoo.com/group/lpc21isp/files/Beta%20versions/

FILE: lpc21isp_155.zip
Yes, there are other versions.

psas_eclipse.sh -

The mirrors for eclipse do not remain stable. This script may not be functional at this time but might be good for notes on how to install.

Download and install Eclipse IDE. C/C++ and Zylin embedded tools packages are installed through the Eclipse IDE interface. See these resources:
http://psas.pdx.edu/OlimexLPC2148Setup/
http://www.sparkfun.com/tutorial/ARM/ARM_Cross_Development_with_Eclipse.pdf

Default Directories.

lpc-kit/

Top level directory.

lpc-kit/Tools

This is where the scripts are placed.

lpc-kit/IDE

This is where Eclipse IDE is installed.

/opt/cross

This is where all the compilation tools are built.

/opt/ocd

This is where the openocd software is installed.

lpc-kit/toolchain/src
lpc-kit/toolchain/ocd_src
Source files are downloaded/stored here.

lpc-kit/Config
Here are configuration files for openocd, and header files for compilation on lpc2148.

lpc-kit/Dev/2148/<project>
Example development directory(ies) for 2148 software. Use for introduction to programming the lpc2148 or for testing the install of the cross compile and programming toolchain.

Running the scripts.

This is a high level view of what happens to begin the setup of a development environment:

Step 0: Start in the Tools directory.

```
$ cd ~/lpc-kit/Tools
```

Step 1: Build all the cross compilation tools for arm.

```
$ psas_lpc.sh &
```

Follow the progress with tail -f psas_build.log.
This will build arm compiler tools and gdb.
Then it will run psas_ftdilib.sh and psas_ocd.sh.
These will generate log files as well for reference.
You can also just run psas_build.sh >& psas_build.log & and follow steps
2 and 3 below if you want to go more carefully.

Step 2: ftdi libs for openocd.

These libraries are needed by openocd.

```
$ psas_ftdilib.sh -i
```

Step 3: openocd (jtag programming)

Download and build the openocd software.

```
$ psas_ocd.sh -i
```

Step 3a: lpc21isp (serial port ISP programming)

```
$ psas_lpc21isp.sh
```

Compiling and Programming

How to compile code.
See: http://psas.pdx.edu/OlimexLPC2148Setup/

Here is an example alias to set your path for a shell.
alias psas="export PATH=/opt/cross/bin:$HOME/lpc-kit/LPC/2148/ISP:$HOME/lpc-kit/Tools: $PATH; echo New path is \$PATH"

You could also set this path permanently in your `.bashrc` or whatever shell rc file you use.

There are many compilation tools available in /opt/cross/bin:

```
 $ ls
arm-elf-addr2line*  arm-elf-g++*        arm-elf-gdbtui*   arm-elf-readelf*
arm-elf-ar*         arm-elf-gcc*        arm-elf-ld*       arm-elf-run*
arm-elf-as*         arm-elf-gcc-4.2.3*  arm-elf-nm*       arm-elf-size*
arm-elf-c++*        arm-elf-gccbug*     arm-elf-objcopy*  arm-elf-strings*
arm-elf-c++filt*    arm-elf-gcov*       arm-elf-objdump*  arm-elf-strip*
arm-elf-cpp*        arm-elf-gdb*        arm-elf-ranlib*
```

How to use jtag to program olimex 2148 board

See: http://psas.pdx.edu/OlimexLPC2148Setup/

Flash programming script:

Here is an example script for programming the flash with openocd. (openocd must already be running).

```
#!/usr/bin/expect


set timeout 20 ;# just in case.

set host localhost ;# 127.0.0.0

set port 4444 ;# openocd port

set cwd [pwd]

spawn telnet "$host" "$port";

expect ">"  ;# prompt

send "script $cwd/oocd_flash_lpc2148.script\r";

interact;
```

Example Programming

First, add the script and cross-compilation tool paths to the PATH variable in your .bashrc (or other shell start up script). E.g.

```
export PATH=/opt/cross/bin:/opt/ocd/bin:$DIR/lpc-kit/LPC/2148/bin:
$DIR/lpc-kit/Tools:$DIR/LPC/2148/OCD/bin:$PATH
```

Replace $DIR with the directory where you ran git-clone for lpc-kit.

Next, find the example project directory `lpc-kit/Dev/2148/poke/src` Inside you'll find several files:

- Script for programming flash: `oocd_flash_lpc2148.script`
- Example c – source code file: `poke.c`
- Startup code for lpc: `crt.s`

This is just a very simple example. It isn't very useful, except for testing the installation of the compilers etc... It should be used with the olimex LPC2148 development board.

Type 'make'

```
Dev/2148/poke > make
.assembling
arm-elf-as -g -ahls -mapcs-32 -o crt.o crt.s > crt.lst
.compiling
arm-elf-gcc -I./ -c -fno-common -O0 -g poke.c
..linking
arm-elf-ld -v -Map poke.map -T2148_demo.cmd -o poke.out  crt.o poke.o
GNU ld version 2.17
.........making poke.s
arm-elf-gcc -s -I./ -fno-common -O0 -o poke.s -g poke.c
...copying
arm-elf-objcopy -O binary poke.out poke.bin
arm-elf-objdump -x --syms poke.out > poke.dmp
...building hex
arm-elf-objcopy -O ihex poke.out poke.hex
```

Plug in the development board and the jtag programming interface. See http://psas.pdx.edu/OlimexLPC2148Setup/ to verify the current cabling and jumper directions:

- Make sure to set the Olimex JTAG programmer to 9V by putting the jumper on the right-most pins (the two farthest away from the JTAG cable)
- Plug a USB cable from the JTAG programmer to your computer.
- Power the LPC2148 dev board by plugging it into a 9V DC wall wort. Powering it over the white cable provided with the JTAG programmer does not seem to provide enough power with some laptops.
- Make sure the LPC2148 debug jumper is jumpered (the DBG_E pins are next to the JTAG port)
- Set both dip-switches on the LPC2148 board to the 'off' position

Start openocd from the lpc-kit/Dev/2148/poke/src/ directory:

```
sudo openocd -s ../../../../Config/2148 -f openocd_lpc2148_v1257.cfg
```

```
Open On-Chip Debugger 1.0 (2008-06-16-12:21) svn:709
$URL: svn://svn.berlios.de/openocd/trunk/src/openocd.c $
```

In **another terminal :**

Either run ~/PSAS/Tools/lpc_flash.exp

or

```
telnet localhost 4444
script oocd_flash_lpc2148.script
```

```
Dev/2148/poke > telnet localhost 4444
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Open On-Chip Debugger
> script oocd_flash_lpc2148.script
```

The expect script automates the telnet process. The effect is the same, but less typing.

The JTAG programmer may beep when it is done programming the board.

Example Continued. Running poke on OlimexLPC2148.

Now connect a serial (or USB to serial) cable to the LPC2148 so you can talk to the reprogrammed board.

There are two ways to do this; either is valid. You can connect a serial cable from the serial port on the JTAG programmer to the LPC2148. The JTAG programmer contains an FTDI chip that provides a JTAG and a serial link from the USB cable attached to the host computer. The second way to communicate with the board is to attach a serial cable (or USB to serial cable) directly from the LPC2148 board to the host PC. You'll probably want to unscrew and remove the metal covers on the LPC2148 serial ports, as most USB to serial adapters do not include retractable screws.

Open minicom and configure to use the USB serial port. If you've attached the LPC2148 serial port to the JTAG connector, there will be only one USB serial device available so use /dev/ttyUSB0. If you've connected the LPC2148 to your computer with a USB to serial cable, you will have two USB serial devices, usually /dev/ttyUSB0 and /dev/ttyUSB1. You might need to unplug and replug in the USB to serial adaptor to figure out which /dev entry is the LPC2148 and which is for the JTAG programmer. You may also need to unplug the JTAG and just use a USB to serial adapter if the board seems to hang.

Once you have minicom configured, make the LPC2148 execute the poke program. If you are still working with openocd and telnet type: reset run. Otherwise hit the reset button on the LPC2148 (it's next to the on/off switches). Here is an example using minicom: