

# PSAS-LPC2148 Development Setup Notes/Scripts

## Table of Contents

Introduction.....	1
Versions.....	2
Get the scripts.....	2
psas_lpc.sh – .....	2
psas_build.sh – .....	2
psas_fdtlib.sh – .....	3
psas_ocd.sh - .....	3
psas_lpc21isp.sh -.....	3
psas_eclipse.sh -.....	4
Default Directories.....	4
Running the scripts.....	4
Step 0: Start in the Tools directory.....	4
Step 1: Build all the cross compilation tools for arm.....	4
Step 2: ftdi libs for openocd. ....	5
Step 3: openocd (jtag programming).....	5
Step 3a: lpc21isp (serial port ISP programming).....	5
Step 4: Eclipse.....	5
Compiling and Programming.....	5
Flash programming script:.....	6
Example Programming.....	6
Example Continued. Running poke on OlimexLPC2148.....	9

## 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.)

There is very little system wide configuration, all compilers, IDE's, scripts etc., are located in a user directory, not a system directory where it would overwrite configurations for other projects.

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

This information could also be considered as notes created while gathering all this information.

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

## Versions

Source	Version
GCC_VERSION	4.3.2
BINUTILS_VERSION	2.19
GDB_VERSION	6.8
EXPAT_VERSION	2.0.1
NEWLIB_VERSION	1.16.0
libftd2xx0	libftd2xx0.4.13
Openocd OOCd_VERSION	1257
Eclipse	3.4.1 ("Ganymede")

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

## 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/PSAS/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 wherever/lpc-kit/PSAS/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\_ftdlib.sh, psas\_ocd.sh.

### **psas\_build.sh –**

Attempts to download and build software packages. Installs in wherever/lpc-kit/PSAS/LPC 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\_ftdlib.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\_ftdlib.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\_ftdlib.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](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: ... wherever/PSAS/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 -**

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](http://www.sparkfun.com/tutorial/ARM/ARM_Cross_Development_with_Eclipse.pdf)

## **Default Directories.**

.../PSAS

Top level directory.

.../PSAS/Tools

This is where the scripts are placed.

.../PSAS/IDE

This is where Eclipse IDE is installed.

.../PSAS/LPC/2148

This is where all the compilation tools are built.

.../PSAS/LPC/2148/OCD

This is where the openocd software is installed.

.../PSAS/Sources

Source files for openocd and lpc21isp are stored here.

.../PSAS/Config

Here are configuration files for openocd, and header files for compilation on lpc2148.

.../PSAS/Dev/2148/<project>

Development directory(ies) for 2148 software.

## **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.**

yourprompt > cd ~/PSAS/Tools

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

(Needs about 2-4GB.)<sup>1</sup>

---

<sup>1</sup> PSAS > du -s ./LPC  
3157216 ./LPC

```
myprompt > 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_ftdlib.sh` and `psas_ocd.sh`.

Then it will run `psas_eclipse.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.

```
myprompt > psas_ftdlib.sh -i
```

## Step 3: openocd (jtag programming)

Download and build the openocd software.

```
myprompt > psas_ocd.sh -i
```

## Step 3a: lpc21isp (serial port ISP programming)

```
myprompt > psas_lpc21isp.sh
```

## Step 4: Eclipse

```
myprompt > psas_eclipse.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=$HOME/PSAS/LPC/2148/bin:$HOME/PSAS/Tools:$PATH; echo New  
path is $PATH"
```

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

There are many compilation tools available in `$HOME/PSAS/LPC/2148/bin`.

```
~/PSAS/LPC/2148/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-gcbug*    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 expect 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

Find project directory in ~/PSAS/Dev/2148. This example is called 'poke'.

```
~/PSAS/Dev/2148 > ls
```

```
poke/
```

Set the path to the tools for cross compiling:

```
~/PSAS/Dev/2148/poke > export
```

```
PATH=/home/kwilson/PSAS/LPC/2148/bin:/home/kwilson/PSAS/Tools:/home/kwilson/Tools/bash:/usr/bin:/usr/local/bin:/usr/sbin:/bin:/sbin:/usr/local/share:/opt/gnome/bin; echo New path is $PATH
```

New path is

```
/home/kwilson/PSAS/LPC/2148/bin:/home/kwilson/PSAS/Tools:/home/kwilson/Tools/bash:/usr/bin:/usr/local/bin:/usr/sbin:/bin:/sbin:/usr/local/share:/opt/gnome/bin
```

Set the environment vars for Makefile:

```
PSAS_DEV=$HOME/PSAS/Dev/2148
```

```
export PSAS_DEV
PSAS_DIR=$HOME/PSAS
export PSAS_DIR
```

These files should be in the directory to compile.

Linker information:

```
2148_demo.cmd -> ../../../../Config/2148/2148_demo.cmd
```

Configuration script for openocd jtag software:

```
openocd_lpc2148.cfg
```

Script for programming flash:

```
oocd_flash_lpc2148.script
```

gdb initialization file (debugger-optional):

```
init.gdb
```

Example c – source code file:

```
poke.c
```

This is an explore the lpc type of program. It isn't very useful, except for testing the installation of the compilers etc... Used with the olimex LPC2148 development board.

Startup code for lpc:

```
crt.s
```

Set the path and 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.

Start openocd.

```
sudo ~/PSAS/LPC/2148/OCD/bin/openocd -s ~/PSAS/Config/2148 -f
openocd_lpc2148.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
```

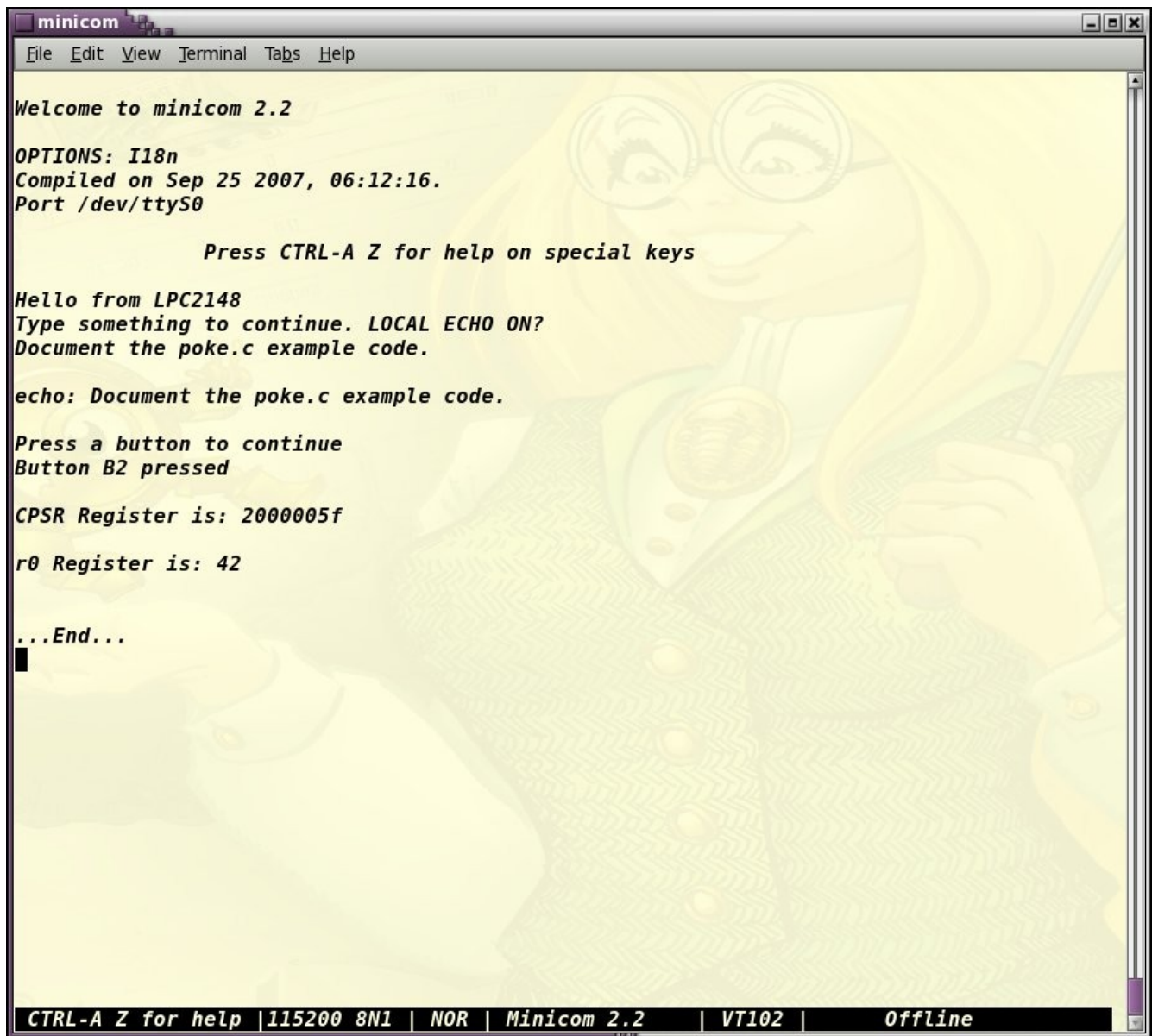
The expect script automates the telnet process. The effect is the same, but less typing.



## Example Continued. Running poke on OlimexLPC2148.

Execute the program. If you are still working with openocd and telnet type: reset run.

Otherwise hit the reset button. Here is an example using minicom.



```
minicom
File Edit View Terminal Tabs Help

Welcome to minicom 2.2

OPTIONS: I18n
Compiled on Sep 25 2007, 06:12:16.
Port /dev/ttyS0

    Press CTRL-A Z for help on special keys

Hello from LPC2148
Type something to continue. LOCAL ECHO ON?
Document the poke.c example code.

echo: Document the poke.c example code.

Press a button to continue
Button B2 pressed

CPSR Register is: 2000005f
r0 Register is: 42

...End...
█

CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.2 | VT102 | Offline
```