Trampoline on Olimex LPC-E2294

Release 1.0

Florent PAVIN

March 31, 2010

CONTENTS

CHAPTER

ONE

Trampoline on Olimex LPC-E2294

This document describes how to get started a Trampoline [1] application on Olimex LPC-2294 The following steps will be overviewed :

- Trampoline installation
- GCC compilation for ARM platform
- Installation of drivers and softwares to be able to upload programs on the Olimex LPC-E2294.

This document is explained from a UNIX side. If you are on a Windows system, download and install Cygwin as it is explained on NXTOSEK website (http://lejos-osek.sourceforge.net/installation_windows.htm) first.

1.1 Trampoline

To download Trampoline, type in a terminal (login:anonymous - password:anonymous) :

```
$svn checkout https://trampoline.rts-software.org/svn/trunk
```

Download the libpm (http://galgas.rts-software.org/download/) for your achitecture and copy it in trunk/. To compile GOIL, go in goil/makefile_[ARCH] depending on your architecture and type in a terminal (you can add -j2 if you have 2 processors) :

```
$make goil
```

This will generate the GOIL executable you'll need. You can add the path to your GOIL (in ~/.profile), adding :

[1] Trampoline is an open source RTOS which, once certified, could be compliant with the OSEK/VDX specification. Currently it is not the case, so while Trampoline has the same API as OSEK/VDX, it is not officially compliant. Trampoline is available under the GNU Lesser General Public License V2 and **is maintained by gcc-4.0.1**. Trampoline runs on several platforms like POSIX, C166 with Keil compiler, Star12X (courtesy of Geensys), PowerPC, ARM (Olimex for example)

```
export PATH=[TRAMPOLINE_REPOSITORY]/goil/makefile_[ARCH]/:$PATH
```

You can also do it typing ”$sudo make install goil” (this will copy the goil executable in /usr/local/bin).

You can now use Trampoline on Unix system by executing the tests as described in Annex 2.1.

1.2 GNUARM

1.2.1 MAC OS

Download and unpack the necessary packages: binutils, gcc, newlib and gdb.

```
$ mkdir ~/crossgcc && cd ~/crossgcc
$ wget ftp://sourceware.org/pub/binutils/snapshots/binutils-2.18.50.tar.bz2
$ tar jxf binutils-2.18.50.tar.bz2
$ wget http://ftp.gnu.org/pub/gnu/gcc/gcc-4.2.3/gcc-4.2.3.tar.bz2
$ tar jxf gcc-4.2.3.tar.bz2
$ wget ftp://sources.redhat.com/pub/newlib/newlib-1.16.0.tar.gz
$ tar zxf newlib-1.16.0.tar.gz
$ wget http://ftp.gnu.org/pub/gnu/gdb/gdb-6.6.tar.gz
$ tar zxf gdb-6.6.tar.gz
```

The installation directory should be /usr/local/crossgcc.

```
$ sudo mkdir /usr/local/crossgcc
$ sudo chmod 777 /usr/local/crossgcc
```

First we build the binutils:

```
$ mkdir build-binutils && cd build-binutils
$ ../binutils-2.18.50/configure --target=arm-elf \
--prefix=/usr/local/crossgcc/ 2>&1 | tee configure.log
$ make all install 2>&1 | tee make.log
$ export PATH=$PATH:/usr/local/crossgcc/bin
```

Build the gcc compiler with C/C++ support:

```
$ cd ../gcc-4.2.3
$ ln -s ../newlib-1.16.0/newlib .
$ ln -s ../newlib-1.16.0/libgloss .
$ cd ..
$ mkdir build-gcc && cd build-gcc
$ ../gcc-4.2.3/configure --target=arm-elf \
--prefix=/usr/local/crossgcc/ --with-newlib \
--with-gnu-as --with-gnu-ld --enable-languages=c,c++ 2>&1 | tee configure.log
$ make all install 2>&1 | tee make.log
```

Build the gdb debugger:

```
$ cd ..
$ mkdir build-gdb && cd build-gdb
$ ../gdb-6.6/configure --target=arm-elf --prefix=/usr/local/crossgcc/
$ make all install 2>&1 | tee make.log
```

Add arm-elf tp your path, adding

```
export PATH=/usr/local/crossgcc/bin:$PATH
```

into ~/.profile

You can now compile an application for ARM as described in Annex 2.2.

1.2.2 Linux

The complete process should take around 1 hour and finish with the message "Build complete !". Alternatively, you could use a pre-build version of this toolchain (see below).

Install the required libraries by running the following command:

```
$ sudo apt-get install tk-dev ncurses-dev libmpfr-dev texinfo
```

on ubuntu 8.10 only: gcc-4.2 (and dependencies) is also required.

Download the script build_arm_toolchain.sh (http://lejos-osek.sourceforge.net/installation_linux_files/build_arm_toolchain.sh). This script is a modified version of the toolchain build script from the NxOS project. It will:

- download sources of binutils, gcc, newlib and gdb
- compile them in place with the right options

Give a look at the script code before execution (safe behaviour), change GCC_BINARY_VERSION to

4.2 if you are under Ubuntu 8.10, and then run it:

```
$ sh ./build_am_toolchain.sh
```

The arm-elf-gcc is now in your path but if you start a new terminal, don't forget to add it typing :

```
export PATH=[the_path_to_your_gnuarm_directory]/bin:$PATH
```

You can now compile an application for ARM as described in Annex 2.2.

1.2.3 Windows

- Download a GCC-4.0.2 tool chain (http://www.gnuarm.com/bu-2.16.1_gcc-4.0.2-c-c++_nl-1.14.0_gi-6.4.exe).
- Install only selected components in the below picture. ARM7(ATMEL AT91SAM7S256) in the Olimex is Little Endian and does not have FPU.
- Do not check "Install Cygwin DLLs..." because Cygwin was already installed.
- At the end of the installation, you were asked about adding the tool path to Windows Environment Variables, but it would not be needed.

1.3 OPENOCD + drivers

openOCD is a software that communicates via the JTAG (Open On-Chip Debug Solution for Embedded Target Systems based on the ARM7 and ARM9 Family). It's an open source software by BerliOS (http://openocd.berlios.de/web/).
Download and compile openOCD for your achitecture (it appears revision after 1200 are not able to compile correcty under MacOS X).
As you don't need just openOCD because it needs the USB driver, you've got two choice :

- install libusb et libftdi
- install D2XX

CHAPTER

TWO

Appendix

2.1 Launching Trampoline tests

To launch the tests you have to compile ViPER [1] first. Go in viper/ and type in a terminal :

```
$make
```

To launch the tests, go in check/ and type in a terminal :

```
$./tests.sh
```

At the end of the tests you should see :

```
...
Compare results with the expected ones...
Functional tests Succeed!!
GOIL tests Succeed!!
```

If an error occurs, you can visit Trampoline's forum (http://trampoline.rts-software.org/bb/).

2.2 Cross-Compile an application

To cross-compile a Trampoline application for ARM ports, you need to set :

```
COMPILER = "arm-elf-gcc";
ASSEMBLER = "arm-elf-as";
LINKER = "arm-elf-ld";
```

in your oil file as you can see in examples/arm/olimex_lpc_e2294/olimex_simple.oil.

[1] Virtual Processor Emulator, ViPER is used on Posix system to send interrupts to Trampoline to emulate the timers. It is launched by Trampoline.

And then, compile your application typing in a terminal (from the example in examples/arm/olimex_lpc_-e2294) :

```
$goil -t=arm/olimex_lpc_e2294 --templates=../../../../goil/templates -g -i
  olimex_simple.oil
```

This will generate the Makefile needed, thus type :

```
$make
```

Then you need to upload the olimex_simple_exe.rxe file (see Annex 2.3- after installing drivers and softwares on your platform (1.3)

2.3 Upload a program

To upload a program double click on 2-run-openocd.command to open the connection between your computer and the board.
And double click on 2a-debug-external-ram.command if you want to upload your program in external ram with debug.
In the debug consol type :

- ”c” to continue
- ”b” to place a breakpoint
- ”si” to step instruction
- ”display/i $pc” to display assembler code on the pc register
- ”x/i $pc” to visualise the i blocs from pc
- ”info registers” to see the registers state