Cortex-M3 ecosystem

Péter Csordás, Balázs Scherer

November 16, 2011

The following ecosystem is based on CodeSourcery gcc and Eclipse. The software library and programming packages have been constructed at the Department of Measurement and Information Systems, Budapest University of Technology and Economics (BUTE), Faculty of Electrical Engineering and Informatics (VIK). This package contains only open-source codes and free tools, so you are free to reuse it. (Added parts are under BSD license) Please, refer to this page, if you do so. This pdf and the software package is regularly developed and updated. Check the latest version at `http://home.mit.bme.hu/~csordas/CortexProg`!

1 Installation Guide

1.1 Installing CodeSourcery

Download the ARM GCC Cross Compiler from here! Choose "IA32 GNU/Linux Installer" or just download this local mirror!

The Installation Process is straightforward:

- Next, Accept License, Next, Next
- The minimal install set is enough
- Choose an arbitrary install directory - or just leave the default one, Next
- Path Should be modified for all users, Next
- "Don't create Icons", Next
- Install - Get a coffee
- View "Getting Started", (only) if you like it. Click Next and Done.

Now, you have all the command line tools needed for cross-compiling installed. The compilation itself is managed by make(files). You'll need an IDE for code editing and calling make. I prefer Eclipse, but you can choose any other editor.

1.2 Installing Eclipse

The Eclipse IDE for C/C++ is available here or you can download this locally mirrored file for Helios-SR2 release Just extract the zip wherever you like (e.g.: `c:\eclipse`). Putting a shortcut to `eclipse.exe` to your desktop is advisable.

Eclipse is a JAVA based IDE, but the downloaded zip file does not contain the JRE. If you have not installed Java yet, you do NOT need to do it now, because CodeSourcery has the JRE. Just add the directory `%CS%\Sourcery G++ Lite\jre\bin` to your `PATH` Environment variable! (`%CS%` stands for the CodeSourcery Installation directory you have selected before. Environment variable can be set in WinXP selecting `My Computer -> Properties -> Advanced`)

Adding GDB support to Eclipse

For Graphical GDB support (using OpenOCD and JTAG) an additional Eclipse plugin is needed. You may update your Eclipse and install the CDT feature called `C/C++ GDB Hardware Debugging`, but I prefer the Zylin package.

For installation start Eclipse! Choose a workspace, e.g. `C:\CortexProg\cortex_ws\ws_stm3210c`.
{TODO: Eclipse metadata for the other workspaces will be added}
Choose the menu `Help -> Install New Software -> Add` and browse for the downloaded Archive, or choose the site `http://www.zylin.com/zylincdt`! Choose `Zylin Embedded CDT 4.16.1`! Next, Next, Accept, Finish. Restart Eclipse.

1.3 Setting up the software library

Download the software library installer! and let it run. The only parameter to give is a root directory (You can leave the default). **It's path should not contain spaces!** - Makefiles do not like it.

Two Environment variables will be added automatically:

- `DEVENV_ROOT` gives the full path to the installed `devenv` subdirectory. This is used by the common makefiles.
- `OPENOCD_SERVER` is initialized to `localhost`. It should give the IP address of the machine running the Open OCD gdb server. (If you are using some kind of virtual machine without USB sharing, OpenOCD can be started on the host. In this case, the host address should be given in `OPENOCD_SERVER`.)

After installation, restarting the computer might be needed to make this variables visible for Eclipse.

Programming tools

The subdirectory `devenv\programming_tools` contains:

- The command-line version of STM32 FlashLoader utility. It can be used for ST boards for e.g. downloading with a serial cable.
- The openOCD compiled with libusb based ft2232 support. Configuration files for various JTAG cables and controllers are given, so a GDB server can be started. It can be used for downloading and debugging.

Software libraries

The subdirectory `devenv\Libraries` contains reusable makefiles, linker scripts, common and platform specific software libraries.

- `libdefs.mk` can be included in makefiles, it defines all the used library directories
- `gdbprog.mk` can be included in makefiles, it defines a target for downloading the program with GDB
- `common.mk` can be included in makefiles, it defines the targets and rules for building Cortex-M programs.
- The files in `Common` can be used for all Cortex-M core controllers: CMSIS core files, FreeRTOS (ver 6.0.4), A simple GNU libc (newlib) interface
- `Family` contains controller-specific linker script, startup code, CMSIS implementation and software libraries provided by the manufacturer.
- `Board` contains development board specific makefiles (these should be included in the projects) and further software libraries.

2 Revision History

Current Version: 0.02

Version 0.02: Released 21.04.2011.

- Libs in new structure - codes separated: Common / Controller Family / Board. (TODO: SD card and ethernet libs.)
- OpenOCD 0.4.0 added with BME-MITMOT JTAG support. Directory renamed
- LM3S library updated, STM32 SPD updated (V3.5.0), STM USB Dev. Lib. (V3.3.0)

Version 0.01: Released 01.04.2011.

References