VALLEYFORGE TOOLCHAIN USER GUIDE

Zac Frank

January 25, 2012

Abstract

The abstract goes here.

Contents

List of Figures

List of Tables

1 Getting Started

This section will show you how to install the ValleyForge toolchain on your computer, and how to start using it to write code for your microprocessor.

1.1 Installation

It is assumed that the user has the latest version of Ubuntu installed. If not, the latest version can be downloaded from `http://ucmirror.canterbury.ac.nz/linux/ubuntu-releases/`. This image can be burned onto a disc, the computer booted from the disc, and on-screen instructions followed.

Once Ubuntu has been installed, we need to install the packages that do not come preinstalled with Ubuntu but are necessary for the toolchain to run. Most of these packages can be installed using the Debian Package repository. To install these packages, open the terminal, and enter:

```
sudo apt-get install byacc flex gcc-avr avr-libc
```

If the user is programming the AT32UC3C0512C, don't worry, the avr32 compiler and headers are included in the toolchain itself.

Next we will install the toolchain itself. Navigate to the directory where you want the toolchain to reside. This will contain a subdirectory called "src" where all your source code will reside. In the command line, type

```
sudo git clone (PUT REPOSITORY URL HERE)
```

to download the toolchain to your working directory.

1.2 Using the Toolchain

You are now ready to start using the ValleyForge Toolchain. Navigate to the Toolchain Directory, and into the subdirectory "bld".

If the toolchain has been used by someone else betwen being cloned into its directory and now, then run the script "resettc", by typing

```
./resettc
```

into the terminal.

Now run the "VFstart" script. This will be your main interface to the toolchain, and will take care of creating and deleting components, as well as keeping track of project properties.
A prompt will appear asking you provide your user information. This will aid in creating template source files for you to start programming with. You may choose not to provide this information, but will have to fill it in yourself in template files should you let the interface create them.

Follow the onscreen prompts to let the toolchain carry out its operations.

1.2.1 Creating a new Component

To create a new Component, run the VFstart script, and enter "C" into the terminal. Choose the microprocessor you are programming on, and then when prompted, enter the name of your project. The name should not contain any spaces and should not be one of the words "none", "all", or "exit". Follow the onscreen prompts to complete the creation of a new component.

If a bootloader is available for your component, you will be asked to specify which bootloader you wish to use for your component. You will then be prompted to choose some other parameters. If you aren't sure about these, don't worry, you can change them later by going to your component source folder, opening the config file, and editing the required fields. A detailed description of the bootloader options can be found in section [INSERT SECTION HERE].

Electing to create template files will place two source files (a component.h and a component.cpp or .c) files in your component's source folder, containing component details, and outlining a structure for your source files' layout. The bare metal templates contain no code other than an empty main function, while the FreeRTOS templates contain an example task to help get you started.

You will then be prompted to decide whether this component will be your default component. Your default, or active component is the one that will be acted on when you don't specify a component, for example when the build script is called with no arguments. If you have no other components, the component being created will automatically become your active component and you will not be prompted.

Once you have created your new component, a configuration file will also be placed in its source folder. It contains information such as which compiler to use, which micro the project will be programmed on to, and which platform is being used.

You can type exit at any time to exit back to the terminal.

1.2.2 Deleting Projects

To delete a component, start the VFstart script, and choose "Delete". A list of all your projects should appear. Select the component you wish to delete, and confirm the deletion when prompted.

You can type exit at any time to exit back to the terminal.

1.2.3 Changing the Active Project

To change which project is the active project, run the uavstart script, choose Edit, and select "Default Project". The list of projects will come up, and you will be able to choose which one you want to set as your active project.

1.2.4 Changing User Info

To change your user info, run the uavstart script, choose Edit, and select "User Info". Follow the on screen prompts to change the user information.

1.3 Compiling Projects

To compile a project, navigate through the terminal to the “bld” directory. Type

```
./build
```

into the terminal to compile the active project. The output hex file, if compilation is successful, will be found in the bin folder of the toolchain directory. To compile a project other than the active one, add the project’s name after the ./build command. To retain all output files in the “tmp” directory type “retain” as the last argument of the build command, or to build all projects, type “all” as the only argument. For example,

```
./build radio retain
```

will build the project “radio” with output files retained in the tmp directory.

while

```
./build retain
```

will build the default project and retain all files for it (including the makefile) in the “tmp” subdirectory.

```
./build all
```

will build all projects.

2 Bootloaders

The options prompted when creating a new component with bootloader are described below:

- **Blink LED Port**. Specify which port will contain the pin that you will use as a bootloader status LED.
- **Blink LED Pin**. Specify which pin on this port will be the status LED.
- **LED Logic**. Specify which voltage level will turn the status LED on or off.
- **Force Bootloader Port**. Specify which port will contain the pin that will be used by the external to signal to your microprocessor that new firmware is available.
- **Force Bootloader Pin**. Specify which pin on this port will be the Force Bootloader pin.
- **Force Bootloader Logic**. Specify which voltage level will signal new firmware is available.
- **Clock Speed**. Specify, in MHz, the clock speed of the microprocessor.

- **Startup State EEPROM Address**. Elect an EEPROM address which the microprocessor will use to determine whether the processor last shut down cleanly. This will affect how the bootloader behaves.