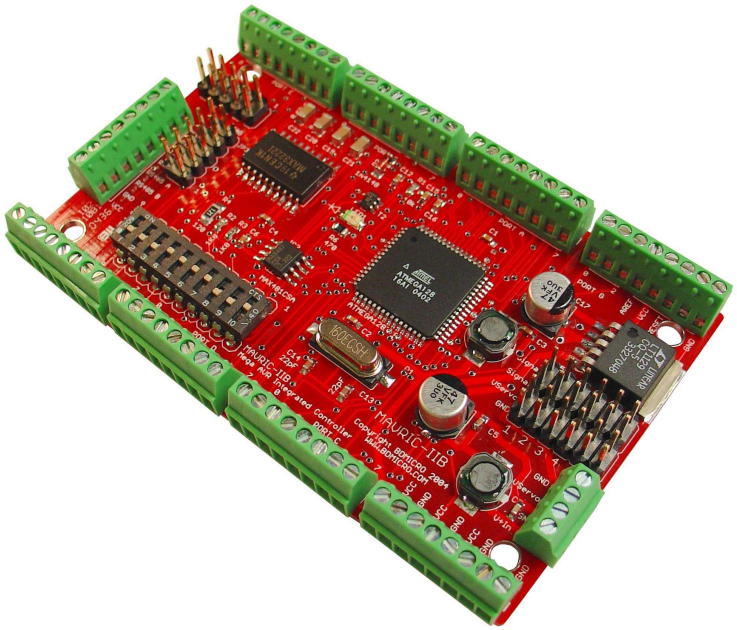

GNU-AVR

Building the GNU AVR Toolchain for Mac OS X and Linux

BDMICRO
http://www.bdmicro.com/

Brian S. Dean
bsd@bdmicro.com

April 24, 2007

Contents

1 Introduction

The GNU GCC tool chain is a powerful development environment developed on Unix systems and has been in existance for nearly two decades. The tools have been ported to many platforms and CPU architectures over the years, from high end Unix servers to small 8-bit embedded MCUs. The Atmel AVR port has been in existance for several years and sees active, continual development and improvements.

This document describes the steps necessary to build the AVR cross development tools from their source. There are many reasons why you might want to build the tools from source instead of installing pre-built packages. The first and foremost is that pre-built packages may not be available on your platform. Since the tools are not that difficult to build, the need for pre-built packages is not that great. However, building the tools from source may seem a little daunting to one who is not familiar with the process. That is what this document is intended to address.

Another reason you might want to build the tools from source, even though pre-built packages may exist, is because you may need to build from source in order to obtain the latest release available. It takes time from when sources are released until pre-built packages are available on all platforms. If you don't want to wait, you can build from source directly.

Note that the compiler is only one component needed for development. The C runtime environment is an essential component and provides the run-time system initialization as well as the C Standard Library facilities. The AVR-LIBC project provides this component with a full featured C run-time and standard library. For more information, see: `http://www.nongnu.org/avr-libc/`.

Once you build a program to run on the AVR, you will need to download it to the MCU. One does this with a downloader tool. AVRDUDE serves this purpose. AVRDUDE loads the firmware program into the AVR MCU using one of several possible programmers. Three major categories of programmers are supported by AVRDUDE including serial programmers like the STK500 and Atmel AVRISP, simple parallel port cable programmers, and Atmel's JTAGICE MkII device. Many AVR based boards include either a 10-pin ISP header for use with serial port and parallel port programmers, while other boards include the 10-pin JTAG header for use with a JTAGICE device.

If you have an older Atmel JTAGICE device (JTAGICE revision prior to the JTAGICE MkII) or equivalent programmer, you will need to build the AVaRICE tool. AVaRICE is similar to AVRDUDE in that it downloads the firmware built by GCC into the MCU where it can execute. AVaRICE also serves double duty in that it works with the GNU Debugger to facilitate single-step debugging. If you wish to use your older JTAGICE device for debugging, you will also need to build GDB - the GNU Debugger. Not all AVR parts support the JTAG protocol. Check the data sheet of the device you are using or with your board manufacturer to determine whether your target supports JTAG.

2 Order of Installation

Some tools rely on other tools in order to build and be installed. For example, GCC must be built before AVR-LIBC, and likewise, BINUTILS must be built and installed before GCC. For best results, build and install the tools in the following order:

- `binutils`
- `gcc`
- `avr-libc`
- `avrdude`

You can stop here if you are not using a JTAG-ICE. If you are using a JTAG-ICE, continue with the following additional tools:

- `gdb`
- `avarice`

3 Build Location

You will need an area on your hard disk to extract the source files and to build the tools. An area of about 700 Megs should be sufficient. For the purposes of these instructions, I will refer to this location as `/usr/local/src/avrtools`. This is the top level directory containing all of the sources for the tools. If that directory does not exist, you can create it as follows:

```
% mkdir -p /usr/local/src/avrtools
```

4 Installation Location

For all tools, the installation location will be specified to be `/usr/local/avr`. This is typically referred to as the *prefix* directory - the root location of the tool installation. All of the tools share this common location which makes it easy to know exactly what was installed, and more importantly, makes it easy to upgrade the tools chain when new releases become available.

4.1 Update Your PATH

Now is a good time to update your login script to update your PATH environment variable to include the `/usr/local/avr/bin` directory. Your login script could go under a number of different names depending on your login shell. Typical names are `.profile`, `.bashrc`, `.zshrc`, `.kshrc`, among others. The following line placed after the PATH is assigned should be sufficient for most common shells:

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

If you are running a “C” type shell, use this instead:

```
setenv PATH "${PATH}:/usr/local/avr/bin"
```

5 Upgrading the Tools to a Newer Release

Upgrading the tool chain is easy. If you are performing an upgrade as opposed to a new install, simply move the old installation location out of the way:

```
% mv /usr/local/avr /usr/local/avr.old
```

Next, simply build and install each tool as normal. You can always go back to the old tool versions by moving the new `/usr/local/avr` out of the way, and moving the old set back into place:

```
% mv /usr/local/avr.old /usr/local/avr
```

6 Fetch the Source, Configure, Build, and Install

In general, each tool requires three steps that vary slightly from tool to tool. First, the sources need to be fetched from the hosting server. The tool needs to be configured and built for your platform. And finally, the tool needs to be installed. Each of these steps will be detailed for each tool in the following sections.

6.1 Binutils

6.1.1 Fetch

```
% cd /usr/local/src/avrtools
% ftp ftp.gnu.org       (username anonymous)
ftp> cd pub/gnu/binutils
ftp> bin
ftp> get binutils-2.17.tar.gz
ftp> quit
```

6.1.2 Extract, Configure, Build, and Install

```
% tar xzvf binutils-2.17.tar.gz
% cd binutils-2.17
% mkdir obj
% cd obj
% ../configure --prefix=/usr/local/avr --target=avr
% make
% make install
```

6.2 GCC

6.2.1 Fetch

```
% ftp ftp.gnu.org       (username anonymous)
ftp> cd pub/gnu/gcc/gcc-4.1.2
ftp> bin
ftp> get gcc-4.1.2.tar.bz2
ftp> quit
```

6.2.2 Extract, Configure, Build, and Install

```
% bunzip2 gcc-4.1.2.tar.bz2
% tar xvf gcc-4.1.2.tar
% cd gcc-4.1.2
% mkdir obj
% cd obj
% ../configure --prefix=/usr/local/avr --target=avr --enable-languages=c,c++ --disable-libssp \
               --disable-nls
% make
% make install
```

6.3 Avr-libc

6.3.1 Fetch

Use your web browser and go to `http://savannah.nongnu.org/download/avr-libc/`. Select `avr-libc-1.4.5.tar.gz` and copy it to `/usr/local/src/avrtools`.

6.3.2 Extract, Configure, Build, and Install

```
% tar xzvf avr-libc-1.4.5.tar.gz
% cd avr-libc-1.2.3
% mkdir obj
% cd obj
% ../configure --prefix=/usr/local/avr
% make
% make install
```

6.4 AVRDUDE

6.4.1 Fetch

Use your web browser and go to `http://savannah.nongnu.org/download/avrdude/`. Select `avrdude-5.0.tar.gz` and copy it to `/usr/local/src/avrtools`.

6.4.2 Extract, Configure, Build, and Install

```
% tar xzvf avrdude-5.0.tar.gz
% cd avrdude-5.0
% mkdir obj
% cd obj
% ../configure --prefix=/usr/local/avr
% make
% make install
```

6.5 GDB

6.5.1 Fetch

```
% ftp ftp.gnu.org       (username anonymous)
ftp> cd pub/gnu/gdb
ftp> bin
ftp> get gdb-6.3.tar.gz
ftp> quit
```

6.5.2 Extract, Configure, Build, and Install

```
% tar xzvf gdb-6.3.tar.gz
% cd gdb-6.3
% ./configure --prefix=/usr/local/avr --target=avr
% make
% make install
```

6.6 AVaRICE

6.6.1 Fetch

Use your web browser and go to `http://sourceforge.net/projects/avarice`. Click the "Download" link for avarice-2.3 in the "Latest File Releases" section, then select `avarice.2.3.tar.bz2` and copy it to `/usr/local/src/avrtools`.

6.6.2 Extract, Configure, Build, and Install

```
% bunzip2 avarice.2.3.tar.bz2
% tar xvf avarice.2.3.tar
% cd avarice.2.3
% mkdir obj
% cd obj
% ../configure --prefix=/usr/local/avr
% make
% make install
```