Eclipse IDE for Embedded AVR Software Development

Helsinki University of Technology
Jaakko Ala-Paavola

February 17th, 2006
Version 0.2

Abstract

This document describes how to set up Eclipse based Integrated Development Environment (IDE) for Atmel's AVR microcontroller family. These instructions are Linux-oriented, but tey are applicable for other OS platforms as well. Thus users of ofter systems may use this document too. Some hints are given for Windows installation.

This document is not an thorough manual of Eclipse. Only instructions on how to setup the system to develope embedded C code for the microcontroller.

Date	**Version**	**Change**
2006/02/09	0.1	Cross-compilation in eclipse
2006/02/17	0.2	ISP uploading

Table 1: Version history

Contents

1 Introduction

Eclipse is defined in the project's web page as follows:

> Eclipse is an open source community whose projects are focused on providing an extensible development platform and application frameworks for building software.

In the Eclipse documentation the same thing stated with more words:

> The Eclipse Platform is an open and extensible platform for anything and yet nothing in particular. It provides a foundation for constructing and running integrated software-development tools. The Eclipse Platform allows tool builders to independently develop tools that integrate with other people's tools so seamlessly you can't tell where one tool ends and another starts.

Due to Eclipse's extentable nature, it can be used for any development purposes. However, somewhat extensive configuration effort is required to make it work as a cross-compilation IDE for embedded C-code.

An IDE provides tools for all steps of software implementation frpm text editing to debugging. Even design and documentation can be done in an IDE.

2 Eclipse installation

Eclipse is available for download at projects official web page

```
http://www.eclipse.org
```

In some Linux distributions, like Ubuntu and derivatives, Eclipse is available as an installation package for the package manager of the system (`apt` in Debian based distros and `yum` in Red Hat based distributions). In an apt-system one may say:

```
# apt-get install eclipse
```

If your system doesn't provide Eclipse as an installation package, download the package from project's web page via mirror-server or peer-to-peer tool (torrent). At the moment of writing this document, the latest stable release of Eclipse is 3.1.2. Install the package by following instructions of your system. Note that you may have to install Java in order to run Eclipse.

2.1 Java Runtime Environment

Eclipse does not include Java Runtime Environment (JRE). You will need a 1.4.2 level or higher Java runtime or Java Development Kit (JDK) installed on your machine in order to run Eclipse 3.1. Java is available for download at Sun's official Java page:

```
http://java.sun.com
```

If your system has Eclipse available via package manager (apt, yum), the system will install required dependency packages automatically.

2.2 C/C++ plugin

Base Eclipse installation does not contain C/C++ development environment. The is a C/C++ Development Tools (CDT) project working towards providing a fully functional C and C++ Integrated Development Environment (IDE) for the Eclipse platform. For more information take a look at projects web site:

```
http://www.eclipse.org/cdt/
```

Eclipse provides a simple mechanism to install plugins/extension at runtime. Notice that you may need to have administrative priviledges to install plugins succesfully.

Select menu *Help - Software Updates - Find and Install*. Select *Search for new features to install* and click *Next*. Click *New Remote Site* and add the URL below. Name of the site can be chosen freely.

```
http://download.eclipse.org/tools/cdt/releases/eclipse3.1
```

Note: If you have installed a newer version of Eclipse, you must modify the URL accordingly. Check the project web page for correct address. Once you have the site corretly configured, as shown in figure 1, click *Finish*.

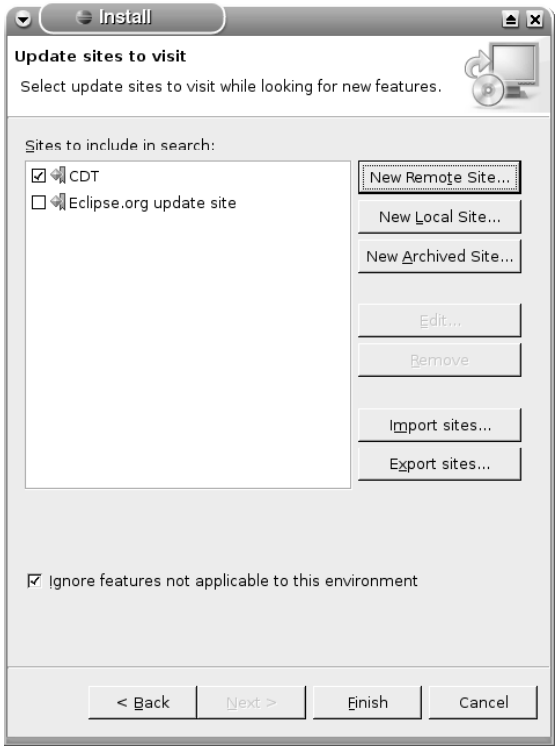

Figure 1: Eclipse software installation

A *Search Results* window will appear once Eclipse has search the given site for software updates, as shown in figure 2. Select the *cdt* version that corresponds to your Eclipse version and click *Next*.

Accept license agreement terms and select *Finish* to start the actual installation procedure. This may take a while depending on server load and available network bandtwith. In meanwhile you may enter next phase.

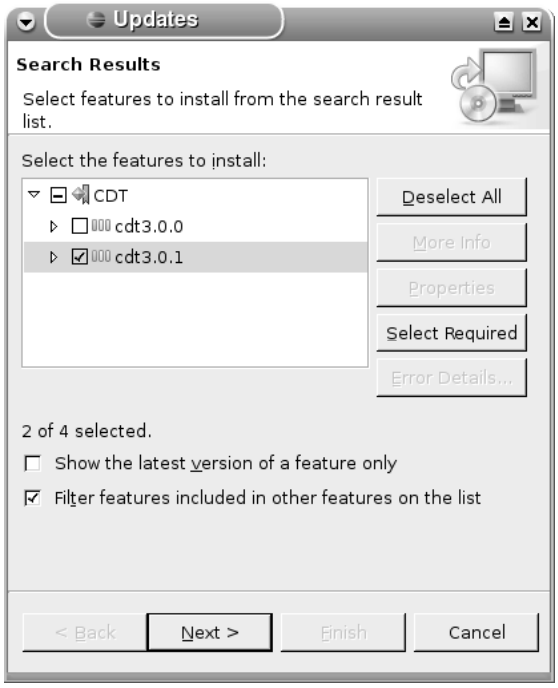

Figure 2: Installation search results

3 GNU C Cross-development tools

In order to develope code for an embedded target, cross-compilation tools are required. GNU tool-chain consists of the following components:

binutils Binary Utilities

gcc C Compiler

libc C Library

gdb Debugger

For most Linux distributions, there exists installation packages for the package management system. In Debian based systems these packages are named: `binutils-avr` `gcc-avr`, `avr-libc`, and `gdb-avr`. Use `apt-get install` or `yum install` commands to install these packages.

There exists a *WinAVR* projects that provides the GNU tools for Windows. Check the project web site for more information. For Mac OS/X and FreeBSD, there are pre-compiled ports available:

```
http://winavr.sourceforge.net/
http://avr-gcc.darwinports.com
http://www.freebsd.org/ports/devel.html
```

In systems that do not have installation packages available or the mechanism is broken, there is allways an option to build the tools from the scratch. Current GNU tools have AVR support included. Download sources of latest stable versions of *binutils*, *gcc* and *gdb* from GNU ftp-site or prefererable from local mirror (Funet in Finland).

```
ftp://ftp.gnu.org/pub/gnu/
ftp://ftp.funet.fi/pub/gnu/gnu/
```

Notice that *avr-libc* is not a GNU software, thus it is not available in the GNU sites. *avr-libc* is available at project's web site:

```
http://www.nongnu.org/avr-libc/
```

Extract source packages and build and install with following commands:

```
# ./configure --prefix=/usr/local/avr --target=avr
# make
# make install
```

where the `--prefix` indicates the directory where the tools will be installed. To run the *install* command, administrative priviledges (root rights) are required.

4 Eclipse configuration

4.1 New project

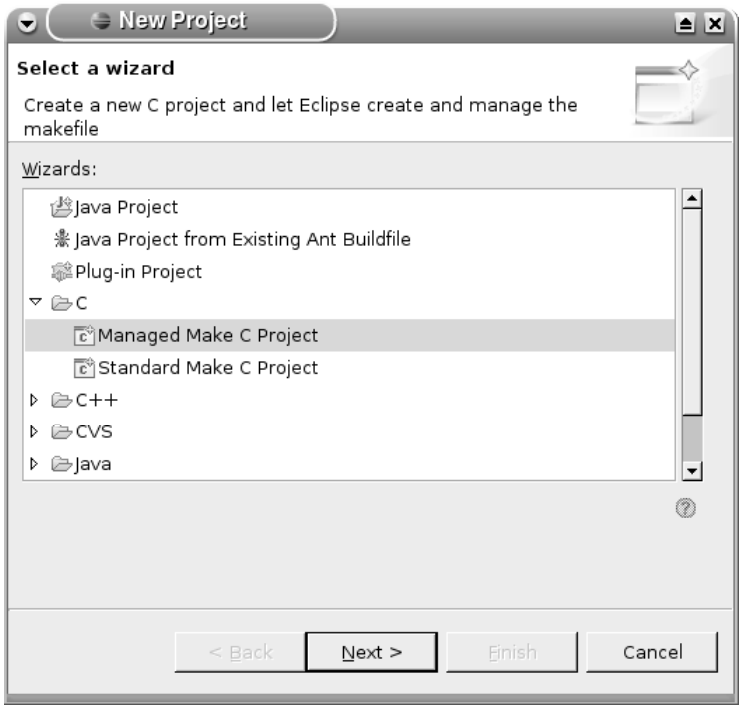

Figure 3: Create new project

To create a new project in Eclipse, select menu *File - New - Project*. Select folder *C* and *Managed Make C Project*, as shown in figure 3. In managed project, one does not need to take care of writing the Makefile, the system will generate apropriate Makefile depending on configurations. Click *Next* and enter a name for your project, then click *Finish*. In the case of this document, the name of the project is *avr-test*, and that will be refered in later configuration settings. If Eclipse asks for opening C/C++ perspective, answer *Yes*.

4.2 Properties

Select menu *Project - Properties* and in the properties window page *C/C++ Build*, as shown in figure 4. In the page, do the following settings:

Tools settings

GCC C Compiler Command: `avr-gcc`

Directories Include paths: `/usr/avr/include`
Set according to your setup. This is default for Debian distros.

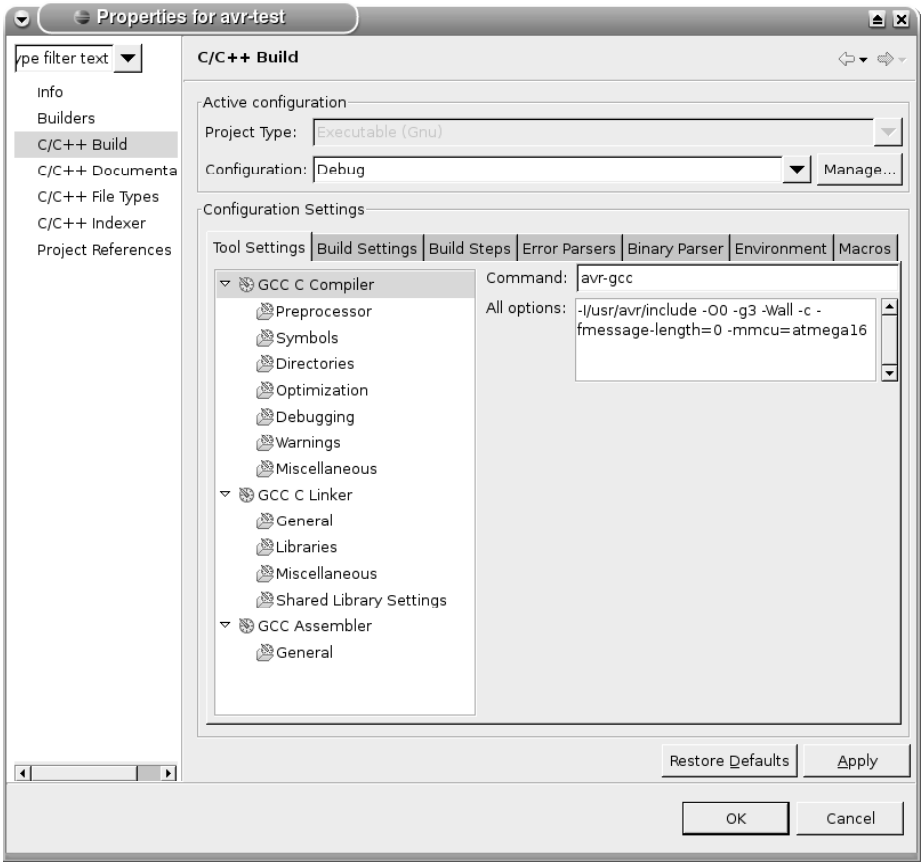

Figure 4: Project properties

Miscallaneous Other flags: add `-mmcu=atmega16`
Change to conform your target microcontroller.

GCC C Linker Command: `avr-gcc`

GCC Assembler Command: `avr-as`

Build Settings

Build output Artifact extension: `elf`

Build steps

Post-build steps Command: `avr-objcopy -O srec avr-test.elf avr-test.rom`
Change the target name to conform with the output of your project.

Environment

Environment variables If your tools are not in the default search path, add here your path extension, example: Variable: `PATH` value:`/usr/local/avr:$PATH`.

In *C/C++ Indexer* page, select *CTAG Indexer*. The default *Full C/C++ Indexer* does not understand avr-libc libraries, thus generating extra error messages when compiling.

5 Uploading

Eclipse can upload compiled code into the target hardware using AVR In-System Programming (ISP) at build time. An external uploader software and ISP cable are required

5.1 Installation

UISP and *AvrDude* are the most common uploader software for AVR. The choice between the two is matter of taste. They are available at their web sites:

```
http://www.nongnu.org/uisp/
http://www.nongnu.org/avrdude/
```

In an debian based system, both softwares are included as an installation package. In this instructions *UISP* is applied. Use the package manager application to install it:

```
# apt-get install uisp
```

There are ten's of different uploader cable designs available. Both uploaders support a number of different cables. The most simple one is parallel port connected direct access cable *dapa*. Test your setup with following command:

```
# uisp -dprog=dapa
```

If you have a different programming cable, modify the command accordingly. Consult UISP documentation for further help. If you have a serial port ISP-cable, you may need to specify serial port settings as well in

If the programmer reports error related to parallel port (parport od ppdev), make sure you have parallel port device installed and correct permissions set. Notice that these commands must be executed with root priviledges.

```
# modprobe ppdev
# chmod a+rw /dev/parport0
```

Once your system is correctly set up, the uploader should report something similar to:

> Atmel AVR ATmega16 is found.

5.2 Configuration

In Eclipse, select menu *Project - Properties*. Select *C/C++ Build* page and *Build steps* tab. In the *Post-build steps* section add command:

```
avr-objcopy -O srec avr-test.elf avr-test.rom;
uisp -dprog=dapa --erase --upload --verify if=avr-test.rom
```

Modify the project name and according to your project. Make sure you have correct cable definition. Now, if the cable is connected properly and the target hardware is powered, the Eclipse build procedure should upload the binary into the AVR.

If the target hardware is not available at build time, the build will report failure. Thus it is convenient to create a new build target for uploading purposes. In *Porject - Properties* at *C/C++ Build page* in *Active configuration* window, select *Manage* button. Create a new upload configuration. Copy settings from your previos (Debug) configuration. Now modify *Build steps* in a way that you have `uisp` command in the upload configuration and not in other configuration. This way you may control wether code will be uploaded at build time or not by selecting proper configuration.

6 Debugging

Most microcontroller system are equiped with industry standard JTAG-interface. AVR mega-series do no exception. The interface provides in-system programming and debugging facilities. This part of the document expects that the user has Atmel's AVR JTAG ICE compatible JTAG adapter available.

```
http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2737
```

6.1 Installation

AvaRICE is a program which interfaces the GNU Debugger GDB with the AVR JTAG ICE available from Atmel. There are some third party clones of the Atmel jtagice available for purchase via the web for prices much less than the Atmel's offering. Avarice is available at project's web page:

```
http://avarice.sourceforge.net/
```

Before install avarice, the *bfd* library from binutils or gdb must be compiled and present. In Debian based Linux systems, the library is included in `binutils-dev` package, that must be installed.

```
# apt-get install binutils-dev
```

The *bfd* library is included in *binutils* and *gdb* sources. Configure and build the library with following commands:

```
# cd bfd
# ./configure --target=avr \
              --prefix=/usr/local/avr \
              --disable-nls \
              --enable-install-libbfd
# make
# make install
```

At the moment of writing this document, the latest stable version of AvaRICE is 2.4. It is not available as an installation package, thus on have to build it by oneself. Download source from a sourceforge mirror site. Configure and and compile the tool with following commands:

```
# tar xvjf avarice-2.4.tar.bz2
# cd avarice-2.4
# ./configure
# make
# make install
```

By default, AvaRICE looks for a device `/dev/avrjtag`. It is recommended to create a symbolic link from the actual device where JTAG-adapter is connected. Give users permissions to access the device. Change the `ttyS0` to correspond your setup.

```
# chmod a+rw /dev/ttyS0
# ln -s /dev/ttyS0 /dev/avrjtag
```

avr-gdb delivered with Linux packages doesn't seems to cooperate with `avarice` properly. Thus it must be compiled from sources. In order to compile `gdb`, `termcap` library is required. The termcap is not available as a packaget, thus it must be compiled too. Fetch sources for `gdb` and `termcap`. Notice that `gdb` version must be 6.4 or greater!

```
ftp://ftp.gnu.org/pub/pub/gnu/termcap
ftp://ftp.gnu.org/pub/pub/gnu/gdb
```

Build and install `termcap` with following commands:

```
# tar xvjf termcap-1.3.1.tar.gz
# cd termcap-1.3.1
# ./configure
# make
# make install
# ldconfig
```

Build and compile `gdb` with following commands:

```
# tar xvjf gdb-6.4.tar.bz2
# cd gdb-6.4
# ./configure --target=avr --prefix=/usr/avr
# make
# make install
```

In the *configure* command, change the `prefix` argument to the destination folder of your choise, `/usr/avr` is the default in Debian.