Gcc Tools for AVR Studio User's Guide

- **Avr-gcc 3.4.0**
- **avr-libc**
- **avr binutils**
- **avr-debugdump**
- **Elf/Dwarf Parser component for AVR Studio**

Beta1-2004-05-26

1 Introduction

Gcc tools for AVR Studio is a collection of tools and libraries necessary for producing and debugging AVR elf/dwarf object files on windows platforms.
It contains:

- avr-gcc 3.4.0 with Dwarf debug format support
- Gnu binutils compiled for avr.
- avr-libc
- avr-debugdump
- An Elf/Dwarf parser component for AVR Studio

The motivation for this release is to provide AVR Studio users with a compiler that is able to produce elf/dwarf object files and a debugger (AVR Studio) that is capable of debugging these object files.

The current WinAVR package (20040404) at http://sourceforge.net/projects/winavr/ includes the 3.3.2 version of gcc with no support for Dwarf debugging information.Thus, it was necessary to create a new distribution of gcc and related tools for windows.

Also included in the package are the AVR binutils and a simple debug dumper developed by Atmel, (avr-debugdump).

2 Installation

Prerequisites:
- AVR studio version 4.09 or newer

Just execute the AvrElfDwarfTool.exe file to install the package. The installer figures out where AVR Studio is installed and the gcc tools package is installed in a subdirectory of the AVR Tools directory.

3 Avr-gcc

Avr-gcc is just a build of gcc configured for the AVR.
To compile the source file functions.c with dwarf debug information, do something like this:

```
>avr-gcc -gdwarf-2 -o functions.elf functions.c
```

For further information on how to set up a Makefile, and using gcc in larger projects see the file doc/avr-gcc-make-tutorial.txt.

4 The Elf/Dwarf parser

The ElfDwarf parser allows debugging of elf/dwarf object files in AVR Studio.
The current version of studio, (4.09, Build 340) does not acknowledge .elf files as a known format. To overcome this, use the project wizard (Project->Project Wizard), and press “open”. Select “All files” in the combo-box to display all files and double click the elf file you want to open.

5 avr-debugdump

avr-debugdump is a simple tool for dumping the debug information contained in an object file. The following options are available:

-a	Dump everything (all)
-c <pc>	Dump condensed call frame commands at <pc>
-f	Dump sourcefiles
-h <file>	Dump high-level statements in <file>.
-h all	Dump all high-level statements.
-i	Display target information.
-m <memtype>	Display information on <memtype> content(progmem, sram, eeprom)
-M <memtype>	Dump content of <memtype> segment. (progmem, sram, eeprom)
-n <name>	Dump *all* symbols with name <name>
-s <scope>	Dump information on <scope>.
-s all	Dump all scopes.
-S <scope>	Dump all symbols contained in <scope>.
-t <scope>	Dump all types of the symbols contained in <scope>

The syntax for avr-debugdump is:

```
>avr-debugdump [options] objectfile
```

6 Known Issues

- Variables residing in segments other than sram will not get a correct location. avr-gcc does not provide segment information in dwarf debug information for variables (DW_AT_segment). AVR has multiple memory spaces, and gcc does not know how to handle this.

- avr-gcc does not yet generate the .debug_frame section for avr. Thus, the support for callstack information in the new elf/dwarf parser is unused.

- The "step over" mechanism in studio misbehaves in some cases:

  To step into a function you will have to step into it as usual. Then, when the source view shows the function, a second "step into" is needed to get to single stepping its statements. Also, stepping over functions fails sometimes.

  These problems do not occur on the simulator platform.

  This is due to a combination of the way avr-gcc generates code and how some platforms implement the step-over mechanism.

- "step out of" is not working.

  This is due to a combination of avr-gcc's code generation and the implementation of step out of on some platforms.

- c++ support:

  There has been made no effort to support c++ in the new parser component.

  - -There is no libstdc++ for avr-gcc.
  - operators new and delete are not implemented.
  - Some header files that come with avr-libc are not c++ safe.
  - Exceptions are not supported. They must be turned off using -fno-exceptions.

- bitfield support is not yet integrated with the rest of AvrStudio

- Multi-dimensional arrays are laid out all in one level.

  Debug info for arrays from avr-gcc is laid out so that it is hard to display multi-dimensional arrays in a more structured manner.

7 Getting started

A batch file is provided to set up an environment for using the tools in this package and start a command prompt . This batch file is called **AvrGcc.bat** and resides in the **AVR Tools** directory of your AVR Studio installation.

If you prefer to use cygwin and want to use this package, two paths must be added to the $PATH variable:

AVR Tools directory/AvrGcc/bin, and

AVR Tools directory/AvrGcc/utils

There is an example and a short tutorial for getting started using gnu make and avr-gcc included in the package under the *AvrGcc\doc* directory.

8 Support

This is a BETA release and as such not supported via the normal ATMEL support channels. Please send feedback and bug reports directly to the developers of this package at avrbeta@atmel.com

The *AVR Studio 4 forum* at http://www.avrfreaks.com is also a suitable forum for providing feedback or discussing this release.

9 Resources

Gcc documentation:	http://gcc.gnu.org/onlinedocs/
Avr-libc	http://www.nongnu.org/avr-libc/user-manual/index.html