AVR 8-bit GNU Toolchain: Release 3.4.4.1229

The AVR 8-bit GNU Toolchain supports all AVR 8-bit devices. The AVR 8-bit Toolchain is based on the free and open-source GCC compiler. The toolchain includes compiler, assembler, linker and binutils (GCC and Binutils) Standard C library (AVR-libc) and GNU Debugger (GDB).

About this release

This is an update release that fixes some defects and upgrades binutils to higher version.

Atmel

8/32-bits Atmel Microcontrollers

Release 3.4.4.1229

Installation Instructions

System Requirements

AVR 8-bit GNU Toolchain is supported under the following configurations:

Hardware requirements

- Minimum processor Pentium 4, 1GHz
- Minimum 512 MB RAM
- Minimum 500 MB free disk space

AVR 8-bit GNU Toolchain has not been tested on computers with less resources, but may run satisfactorily depending on the number and size of the projects and the user's patience.

Software requirements

- Windows 2000, Windows XP, Windows Vista, Windows 7 (x86 or x86-64) or Windows 8 (x86 or x86-64).
- AVR 8-bit GNU Toolchain is not supported on Windows 98, NT or ME.
- Fedora 13 or 12 (x86 or x86-64), RedHat Enterprise Linux 4/5/6, Ubuntu Linux 10.04 or 8.04 (x86 or x86-64), or SUSE Linux 11.2 or 11.1 (x86 or x86-64). AVR 8-bit GNU Toolchain may very well work on other distributions. However those would be untested and unsupported.

Downloading and Installing

The package comes in two forms:

- As a standalone self extracting installer (.exe)
- As Atmel Studio Toolchain Extension

It may be downloaded from Atmel's website at *http://www.atmel.com* or from the Atmel Studio Extension Gallery http://gallery.atmel.com

Installing on Windows

In order to install using standalone installer, the AVR Toolchain installer can be downloaded from Atmel website. After downloading the installer, double-click the executable file to install. It will ask for a location to install and when entered, it will extract the toolchain binaries into the corresponding location. This will not add the toolchain path to the system environment variable "PATH". The user has to do it manually. Any number of installations is possible on a single machine. To uninstall, please remove the directory from the file system.

In order to install as extension, please refer to Atmel Studio documentation.

Configuring the toolchain in Atmel Studio

If you plan to use the standalone installer outside Atmel Studio, you can skip this section. To configure a standalone toolchain installation to be used inside Atmel Studio environment, do the following

1. Install the toolchain using the standalone self-extracting installer.
2. From Atmel Studio 6.0 or later, go to Tools menu -> Options.

3. From the dialog select Toolchain -> Package Configuration.
4. From the right pane select nature of the toolchain e.x AVR8 for C, ARM for C++ etc.
5. Click "Add Flavour".
6. From the dialog, enter the name and path to the toolchain executable. For example if it's AVR8 select the path till avr-gcc.exe. and click OK.

If you want support for other architecture/language, please remember to repeat the exercise by choosing the correct "Toolchain" within the "Package configuration" tab.

Now you are done with configuring a toolchain for use from within Atmel Studio. To configure a project to use this toolchain, do the following.

1. Open the project in Atmel Studio (6.0 or later)
2. Right click the project, go to Properties -> Advanced tab.
3. Select the toolchain you configured in the previous step.

Now build the project, and the toolchain should be picked from the configured location.

Installing on Linux

On Linux AVR 8-bit GNU Toolchain is available as a TAR.GZ archive which can be extracted using the 'tar' utility. In order to install, simply extract to the location where you want the toolchain to run from.

Upgrading from previous versions

Upgrading is not supported with the installer. But you are allowed to have any number of versions of the toolchain in your machine. If it is installed via Atmel Studio it can be upgraded through the extension manager in Atmel Studio. See Atmel Studio release notes for more information.

On Linux, if you have it unpacked to a local folder, you just delete the old folder and unpack the latest version in a new folder.

Layout

Listed below are some directories you might want to know about.

`<install_dir>` = The directory where you installed AVR 8-bit GNU Toolchain.

- <install_dir>\bin
  - The AVR software development programs. This directory should be in your `PATH` environment variable. This includes:
    - GNU Binutils
    - GCC
    - GDB

- <install_dir>\avr\lib
  - avr-libc libraries, startup files, linker scripts,and stuff.
- <install_dir>\avr\include
  - avr-libc header files for AVR 8-bit.
- <install_dir>\avr\include\avr
  - header files specific to the AVR 8-bit MCU. This is where, for example, #include <avr/io.h> comes from.
- <install_dir>\lib
  - GCC libraries, other libraries, headers and stuff.

- <install_dir>\libexec
  - GCC program components
- <install_dir>\doc
  - Various documentation.

Toolset Background

AVR 8-bit GNU Toolchain is a collection of executable, open source software development tools for the Atmel AVR 8-bit series of microcontrollers. It includes the GNU GCC compiler for C and C++.

Compiler

The compiler is the GNU Compiler Collection, or GCC. This compiler is incredibly flexible and can be hosted on many platforms, it can target many different processors/operating systems (back-ends), and can be configured for multiple different languages (front-ends).

The GCC included in AVR 8-bit GNU Toolchain is targeted for the AVR 8-bit microcontroller and is configured to compile C or C++.

" **CAUTION:** There are caveats on using C++. See the avr-libc FAQ. C++ language is not fully supported and has some limitations. libstdc++ is unsupported."

Because this GCC is targeted for the AVR 8-bit MCUs, the main executable that is created is prefixed with the target name: `avr-gcc` (with '.exe' extension on MS Windows). It is also referred to as AVR GCC.

`avr-gcc` is just a "driver" program only. The compiler itself is called `cc1.exe` for C, or `cc1plus.exe` for C++. Also, the preprocessor `cpp.exe` will usually automatically be prepended with the target name: `avr-cpp`. The actual set of component programs called is usually derived from the suffix of each source code file being processed.

GCC compiles a high-level computer language into assembly, and that is all. It cannot work alone. GCC is coupled with another project, GNU Binutils, which provides the assembler, linker, librarian and more. Since 'gcc' is just a "driver" program, it can automatically call the assembler and linker directly to build the final program.

Assembler, Linker, Librarian and More

GNU Binutils is a collection of binary utilities. This also includes the assembler, as. Sometimes you will see it referenced as GNU as or gas. Binutils includes the linker, ld; the librarian or archiver, ar. There are many other programs included that provide various functionality.

Note that while the assembler uses the same mnemonics as proposed by Atmel, the "glue" (pseudo-ops, operators, expression syntax) is derived from the common assembler syntax used in Unix assemblers, so it is not directly compatible to Atmel assembler source files.

Binutils is configured for the AVR target and each of the programs is prefixed with the target name. So you have programs such as:

- **avr-as**: The Assembler.
- **avr-ld**: The Linker.
- **avr-ar**: Create, modify, and extract from archives (libraries).
- **avr-ranlib**: Generate index to archive (library) contents.
- **avr-objcopy**: Copy and translate object files.
- **avr-objdump**: Display information from object files including disassembly.
- **avr-size**: List section sizes and total size.

- **avr-nm**: List symbols from object files.
- **avr-strings**: List printable strings from files.
- **avr-strip**: Discard symbols.
- **avr-readelf**: Display the contents of ELF format files.
- **avr-addr2line**: Convert addresses to file and line.
- **avr-c++filt**: Filter to demangle encoded C++ symbols.
- **avr-gdb**: GDB, the GNU debugger, allows you to see what is going on `inside' another program targeted to AVR, while it executes.

See the binutils user manual for more information on what each program can do.

C Library

avr-libc is the Standard C Library for AVR 8-bit GCC. It contains many of the standard C routines, and many non-standard routines that are specific and useful for the AVR 8-bit MCUs.

NOTE: The actual library is currently split into two main parts, libc.a and libm.a, where the latter contains mathematical functions (everything mentioned in <math.h>, and a bit more). Thus it is a good idea to always include the `-lm` linker option. Also, there are additional libraries which allow a customization of the printf and scanf function families.

avr-libc also contains the most documentation on how to use (and build) the entire toolset, including code examples. The avr-libc user manual also contains the FAQ on using the toolset.

Debugging

Atmel Studio provides a debugger and also provides simulators for the parts that can be used for debugging as well. Note that `Atmel Studio` is currently free to the public, but it is not Open Source. The GNU debugger is now shipped along with the toolchain.

Source Code

Atmel AVR 8-bit GNU Toolchain uses modified source code from GCC, Binutils and AVR-LibC. The source code and the build scripts used for building the packaged binaries are available at:

http://distribute.atmel.no/tools/opensource/Atmel-AVR-GNU-Toolchain/3.4.4/

Please refer to the README for the instructions on how to use the supplied script to build the toolchain.

New and Noteworthy

This chapter lists new and noteworthy items for the AVR 8-bit GNU Toolchain release.

AVR 8-bit GNU Toolchain

Known Issues

- Support for AVR Tiny architecture (ATTiny 4/5/9/10/20/40) has known limitations:
  - libgcc implementation has some known limitations
  - Standard C / Math library implementation are very limited or not present
- Program memory images beyond 128KBytes are supported by the toolchain, subject to the limitations mentioned in "3.17.4.1 EIND and Devices with more than 128 Ki Bytes of Flash" at http://gcc.gnu.org/onlinedocs/gcc/AVR-Options.html
- Named address spaces are supported by the toolchain, subject to the limitations mentioned in "6.16.1 AVR Named Address Spaces" at http://gcc.gnu.org/onlinedocs/gcc/Named-Address-Spaces.html#AVR%20Named%20Address%20Spaces

Updates and Issues Fixed

AVRTCDEV-616

- Include GDB along with the toolchain distribution

AVRTCDEV-678

- Binutils upgraded to 2.24

DEVXML-562

- wrong LFUSE_DEFAULT in iotn84a.h

DEVXML-561

- HFUSE_DEFAULT not defined for iotn84.h

DEVXML-527

- In tiny441/841 header file the REMAP register bit U0MAP and SPIMAP interchanged

DEVXML-523

- RSIG is missing from tiny4313 header

DEVXML-513

- IO view not showing TWIE for ATxmega32D3 and ATxmega64D3 devices during debugging

DEVXML-508

- ATtiny2313A, Tiny4313 : PCMSK0 register is missing in device header file

DEVXML-487

- TINY13A has misspelled BOD register bit names

DEVXML-486

- ATTINY24A uses WATCHDOG_vect instead of WDT_vect

DEVXML-467

- Update the production signature of Xmega E

DEVXML-448

- Update missing definitions in atxmega64d4def inc file

DEVXML-349

- Device signature for Mega164A is wrongly mentioned in device header file

DEVXML-311

- ATtiny167, ICR1 register definition error.

DEVXML-300

- ATmega328P has incorrect definitions of fuses in header file, should be BODLEVEL, but is defined as BOOTSZ0, BOOTSZ1, BOOTRST.

DEVXML-257

- ATTINY2313/ATTINY4313 missing alternative UCSRC register bit defines

DEVXML-174

- TIMCTRL should not be listed under DACB in ATxmega64A3U

DEVXML-171

- Missing external interrupt value option for EICRA

DEVXML-136

- The bitmask for PORTB for tiny25 is erroneously defined

DEVXML-119

- Datasheet for AT90PWM216/316 refers to Bit 1 of PRR as USART0 whereas the header files refers to the same as USART.

DEVXML-117

- Fix GIFR pin definitions and vector name of the Pin Change Interrupt for the device ATtiny4313

AVRTC-710

- Linking fails occasionally for TINY devices

AVRTC-701

- atxmega16x1 is recognized by gcc but device does not exist

AVRTC-698

- power_all_enable power macro broken for ATA664251

AVRTC-696

- AVR GCC compiler does not emit a LDS/STS instruction for ATtiny10

AVRTC-694

- Power macros for AT90PWM216/316 are broken - PRUSART missing

AVRTC-693

- AVR8 3.4.3 toolchain could not recognize XMEGA USB instructions (XCH,LAC,LAS,LAT)

AVRTC-657

- .BOOT section overlaps .data section load image

AVRTC-707

- Error in wdt_enable for XMEGA devices

AVRTC-713

- eeprom write/ update block functions are incorrect for atxmega32e5

Supported Devices

avr2			
at90s2313	at90s2323	at90s2333	at90s2343
attiny22	attiny26	at90s4414	at90s4433
at90s4434	at90s8515	at90c8534	at90s8535
avr25			
ata5272	ata6616c	attiny13	attiny13a
attiny2313	attiny2313a	attiny24	attiny24a
attiny4313	attiny44	attiny44a	attiny441
attiny84	attiny84a	attiny25	attiny45
attiny85	attiny261	attiny261a	attiny461
attiny461a	attiny861	attiny861a	attiny43u
attiny87	attiny48	attiny88	attiny828
attiny841	at86rf401		
avr3			
at43usb355	at76c711		
avr31			
atmega103	at43usb320		
avr35			
ata5505	ata6617c	ata664251	at90usb82
at90usb162	atmega8u2	atmega16u2	atmega32u2

attiny167	attiny1634		

avr4

ata6285	ata6286	ata6289	ata6612c
atmega8	atmega8a	atmega48	atmega48a
atmega48p	atmega48pa	atmega88	atmega88a
atmega88p	atmega88pa	atmega8515	atmega8535
atmega8hva	at90pwm1	at90pwm2	at90pwm2b
at90pwm3	at90pwm3b	at90pwm81	

avr5

ata5702m322	ata5790	ata5790n	ata5795
ata6613c	ata6614q	atmega16	atmega16a
atmega161	atmega162	atmega163	atmega164a
atmega164p	atmega164pa	atmega165	atmega165a
atmega165p	atmega165pa	atmega168	atmega168a
atmega168p	atmega168pa	atmega169	atmega169a
atmega169p	atmega169pa	atmega16hvb	atmega16hvbrevb
atmega16m1	atmega16u4	atmega32a	atmega32
atmega323	atmega324a	atmega324p	atmega324pa
atmega325	atmega325a	atmega325p	atmega325pa
atmega3250	atmega3250a	atmega3250p	atmega3250pa
atmega328	atmega328p	atmega329	atmega329a
atmega329p	atmega329pa	atmega3290	atmega3290a
atmega3290p	atmega3290pa	atmega32c1	atmega32m1
atmega32u4	atmega32u6	atmega406	atmega64
atmega64a	atmega640	atmega644	atmega644a
atmega644p	atmega644pa	atmega645	atmega645a
atmega645p	atmega6450	atmega6450a	atmega6450p
atmega649	atmega649a	atmega649p	atmega6490
atmega16hva	atmega16hva2	atmega32hvb	atmega6490a
atmega6490p	atmega64c1	atmega64m1	atmega64hve
atmega64hve2	atmega64rfr2	atmega644rfr2	atmega32hvbrevb
at90can32	at90can64	at90pwm161	at90pwm216
at90pwm316	at90scr100	at90usb646	at90usb647
at94k	m3000		

avr51

atmega128	atmega128a	atmega1280	atmega1281
atmega1284	atmega1284p	atmega128rfa1	atmega128rfr2
atmega1284rfr2	at90can128	at90usb1286	at90usb1287

avr6

atmega2560	atmega2561	atmega256rfr2	atmega2564rfr2

avr7

ata5782	ata5831		

avrxmega2

atxmega8e5	atxmega16a4	atxmega16a4u	atxmega16c4
atxmega16d4	atxmega16e5	atxmega32a4	atxmega32a4u
atxmega32c3	atxmega32c4	atxmega32d3	atxmega32d4
atxmega32e5			

avrxmega4

atxmega64a3	atxmega64a3u	atxmega64a4u	atxmega64b1
atxmega64b3	atxmega64c3	atxmega64d3	atxmega64d4

avrxmega5

atxmega64a1	atxmega64a1u		

avrxmega6

atxmega128a3	atxmega128a3u	atxmega128b1	atxmega128b3
atxmega128c3	atxmega128d3	atxmega128d4	atxmega192a3
atxmega192a3u	atxmega192c3	atxmega192d3	atxmega256a3
atxmega256a3b	atxmega256a3bu	atxmega256a3u	atxmega256c3
atxmega256d3	atxmega384c3	atxmega384d3	

avrxmega7

atxmega128a1	atxmega128a1u	atxmega128a4u	

avrtiny

attiny4	attiny5	attiny9	attiny10
attiny20	attiny40		

avr1

at90s1200	attiny11	attiny12	attiny15

}

Contact Information

For support on AVR 8-bit GNU Toolchain please contact avr@atmel.com.

Users of AVR 8-bit GNU Toolchain are also welcome to discuss on the AVRFreaks website forum for AVR Software Tools.

Disclaimer and Credits

AVR 8-bit GNU Toolchain is distributed free of charge for the purpose of developing applications for Atmel AVR processors. Use for other purposes are not permitted; see the software license agreement for details. AVR 8-bit GNU Toolchain comes without any warranty.

Copyright 2014 Atmel Corporation. All rights reserved. ATMEL, logo and combinations thereof, Everywhere You Are, AVR, AVR32, and others, are the registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Windows, Internet Explorer and Windows Vista are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Linux is the registered trademark of Linus Torvalds in the United States and other countries. *Built on Eclipse* is a trademark of Eclipse Foundation, Inc. Sun and Java are registered trademarks of Sun Microsystems, Inc. in the United States and other countries. Mozilla and Firefox are registered trademarks of the Mozilla Foundation. Fedora is a trademark of Red Hat, Inc. SUSE is a trademark of Novell, Inc. Other terms and product names may be the trademarks of others.