
AVR 8-bit GNU Toolchain: Release 3.4.0.663

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) and Standard C library (AVRLibC).



8/32-bit **AVR**[®]
Microcontrollers

Release 3.4.0.663





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 or Windows 7 (x86 or x86-64).
- Fedora 13 or 12 (x86 or x86-64), RedHat Enterprise Linux 4 or 5, 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 as well work on the other distributions. However those would be untested and unsupported.

AVR 8-bit GNU Toolchain is not supported on Windows 98, NT or ME.

Downloading and Installing

The package comes in two forms:

- As part of a standalone installer (avr-toolchain-installer)
- As part of Atmel Studio 6 Installer

This can be downloaded from Atmel's website at <http://www.atmel.com>

Installing on Windows

Atmel Studio 6 installation procedure will also install AVR Toolchain. See Atmel Studio 6 release notes for more information.

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. You may use "Custom Installation" in order to install in a specific 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

If it is installed via Atmel Studio 6 it will be upgraded through the Atmel Studio 6 upgrade. See Atmel Studio 6 release notes for more details.

If you used the standalone installer on MS-Windows, you might do a clean upgrade by first un-installing the old version or just upgrade using the latest installer.

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.

Manifest

1. AVR 8-bit GNU Binutils 2.22
 - Binary utilities for AVR 8-bit target (including assembler, linker, etc.).
2. AVR 8-bit GNU Compiler Collection (avr-gcc) 4.6.2
 - C language and C++ language compiler for AVR 8-bit target.
3. AVRLibC 1.8.0
 - C Standard Library for AVR 8-bit

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
- `<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.
- `<install_dir>\source`
 - Documentation on where to find the source code for the various projects and source code patches that were used to build the tools.

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 Micro Controller Units (MCU). 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 MCUs, and is configured to compile C, or C++.

CAUTION: There are caveats on using C++. See the `avr-libc` FAQ.

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.

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

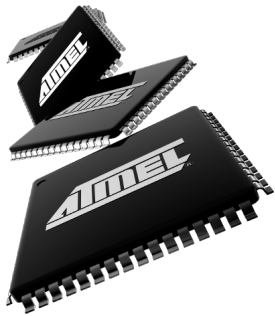
Atmel Studio 6 provides a debugger and also provides simulators for the parts that can be used for debugging as well. Note that `Atmel Studio 6` is currently free to the public, but it is not Open Source.

New and Noteworthy

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

AVR 8-bit GNU Toolchain

Supported Devices



AVR 8-bit GNU Toolchain supports the following devices:

Note:- Devices which are newly supported in this release are marked with *

at90s2313	at90s2323	at90s2333	at90s2343	attiny22
at90s4414	at90s4433	at90s4434	at90s8515	at90c8534
ata5272*	ata5505*	ata5790*	ata5795*	ata6285*
ata6286*	ata6289	attiny13	attiny13a	attiny2313
attiny2313a	attiny24a	attiny4313	attiny44	attiny44a
attiny828*	attiny84	attiny25	attiny45	attiny85
attiny261	attiny261a	attiny461a	attiny861	attiny861a
attiny43u	attiny87	attiny88	at86rf401	at43usb355
at76c711	atmega103	at90usb82	at90usb162	atmega8u2
atmega16u2	atmega32u2	attiny1634	atmega8	atmega8a
atmega48	atmega48a	atmega48pa	atmega88	atmega88a
atmega88p	atmega88pa	atmega8515	atmega8hva	at90pwm1
at90pwm2	at90pwm2b	at90pwm3	at90pwm81	at90pwm161
atmega16	atmega16a	atmega161	atmega163	atmega164a
atmega164p	atmega164pa	atmega165	atmega165a	atmega165pa
atmega168	atmega168a	atmega168p	atmega168pa	atmega169
atmega169a	atmega169pa	atmega16hva	atmega16hva2	atmega16hvb
atmega16hvbrevb	atmega16u4	atmega32	atmega32a	atmega323
atmega324a	atmega324p	atmega325	atmega325a	atmega325p
atmega325pa	atmega3250	atmega3250p	atmega3250pa	atmega328
atmega328p	atmega329	atmega329p	atmega329pa	atmega3290
atmega3290a	atmega3290p	atmega32c1	atmega32m1	atmega32u4
atmega32u6	atmega406	atmega640	atmega644	atmega644a
atmega644p	atmega644pa	atmega645p	atmega645	atmega6450
atmega6450a	atmega6450p	atmega649a	atmega649p	atmega6490
atmega6490a	atmega6490a	atmega64a	atmega64c1	atmega64m1
atmega64hve	atmega32hvb	atmega32hvbrevb	at90can64	at90pwm216
at90pwm316	atmega16c1	atmega32c1	atmega16m1	atmega32m1

atmega16u4	atmega32u4	at90scr100	at90usb646	at90usb647
at94k	m3000	atmega128a	atmega1280	atmega1281
atmega1284	atmega1284p	atmega128rfa1	at90can128	at90usb1287
atmega2560	atmega2561	atxmega16a4	atxmega16a4u	atxmega16d4
atxmega32a4	atxmega32a4u	atxmega32d4	atxmega32x1	atxmega64a3
atxmega64a3u	atxmega64d3	atxmega64a1u	atxmega64a4u	atxmega64b1
atxmega64b3	atxmega64c3	atxmega128a3	atxmega128a3u	atxmega128b1
atxmega128b3	atxmega128d3	atxmega128d4	atxmega192a3	atxmega192a3u
atxmega192c3	atxmega256a3	atxmega256a3b	atxmega256a3bu	atxmega256a3u
atxmega256d3	atxmega128a1	attiny4	attiny5	attiny9
attiny10	attiny20	at90s1200	attiny11	attiny12
attiny15				

Component Upgrades

- Binutils upgraded to 2.22 from 2.20.1
- GCC upgraded to 4.6.2 from 4.5.1
- AVR-LibC upgraded to 1.8.0 from 1.7.1

Known Issues

- AVR Tiny Architecture (ATTiny 4/5/9/10/20/40) support has some limitations. This is expected to be fixed soon. Some notable known issues are:
 - libgcc implementation has some known limitations
 - Standard C / Math library implementation are very limited or not present
 - Attribute signal has some known issues
- Support for ATtiny1634 does not include `clock_prescaler_set()` and `wdt_enable()` macros
- Program memory beyond 128KBytes is not efficiently supported for Mega and Xmega devices

Improvements

- PR 259: Support for Xmega RMW instructions `lat/las/lac/xch`
- PR 232: Dwarf2 CFI (Call frame information) generation

Issues Fixed

- PR 441: Undefined Reference to `__BV` error when `set_sleep_mode` is called for specific devices. This is now corrected.
- PR 424: Linker error "relocation truncated to fit: R_AVR_13_PCREL against symbol XXX defined in YYYY section in ZZZ.o" issued as wrong candidates are taken for relaxation. This issue is now fixed.
- PR 419: Segmentation fault when linker relaxation is enabled. This is caused since the relocations were not read properly (side effect of a change done for binutils PR 12161). This is now fixed.
- PR 206: Incomplete support in `power.h` for specific devices are corrected.
- PR 195: The `clock_prescale_set` function is implemented for tiny devices.
- PR 381: Calling `_delay_ms()` function causes the compiler to crash after showing error for ATtiny10. This is now fixed
- PR 193: Support LDS/STS 16 bit variants as per AVR8L specifications in assembler, linker and disassembler
- PR 444: Support `.config` section for ATTiny 4/5/9/10/20/40



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 2006-2012 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.