Click to verify

This is the standard library for Microchip (formerly Atmel) AVR devices together with the AVR-GCC compiler. The library contains most of the functionality required by the ISO C standard, with the notable exception of wchar_t support. It also contains a lot of auxiliary functionality targeted to the AVR controller family. Further, it includes customizable startup code, tailored to work together with the linker scripts provided by the GNU Binutils, so for most AVR applications, there is usually no need to provide project-specific files for that purpose. See the documentation for details. The library is distributed using a modified BSD-style license. The official source code repository is located at . AVR-LibC depends on GNU Binutils and GCC that should be built for the AVR target. We recommend to use the most recent versions of these tools. Detailed instructions on building these tools from source can be found in Building and Installing the GNU Tool Chain. Note that to build AVR-LibC directly by cloning the GitHub repository, you will need to run the bootstrap script in the root directory first. AVR-LibC uses autoconf so be sure to use recent versions of autoconf and automake to generate the configure script and the Makefiles. This also requires Python being installed and available in the system path. The latest version of this document is always available from The AVR Libc package provides a subset of the standard C library for Atmel AVR 8-bit RISC microcontrollers. In addition, the library provides the basic startup code needed by most applications. There is a wealth of information in this document which goes beyond simply describing the interfaces and routines provided by the library. We hope that this document provides enough information to get a new AVR developer up to speed quickly using the freely available development tools: binutils, gcc avr-libc and many others. If you find yourself stuck on a problem which this document doesn't quite address, you may wish to post a message to the avr-gcc mailing list. Most of the developers of the AVR binutils and gcc ports in addition to the devleopers of avr-libc subscribe to the list, so you will usually be able to get your problem resolved. You can subscribe to the list at . Before posting to the list, you might want to try reading the Frequently Asked Questions chapter of this document. Note: If you think you've found a bug, or have a suggestion for an improvement, either in this documentation or in the library itself, please use the bug tracker at to ensure the issue won't be forgotten. In general, it has been the goal to stick as best as possible to established standards while implementing this library. Commonly, this refers to the C library as described by the ANSI X3.159-1989 and ISO/IEC 9899:1990 ("ANSI-C") standard, as well as parts of their successor ISO/IEC 9899:1999 ("C99"). Some additions have been inspired by other standards like IEEE Std 1003.1-1988 ("POSIX.1"), while other extensions are purely AVR-specific (like the entire program-space string interface). Unless otherwise noted, functions of this library are not guaranteed to be reentrant. In particular, any functions that store local state are known to be non-reentrant, as well as functions that manipulate IO registers like the EEPROM access routines. If these functions are used within both standard and interrupt contexts undefined behaviour will result. See the FAQ for a more detailed discussion. The following is a list of AVR devices currently supported by the library. Note that actual support for some newer devices depends on the ability of the compiler/assembler to support these devices at library compile-time. megaAVR Devices: atmega103 atmega128 atmega128a atmega1280 atmega1281 atmega1284 atmega1284p atmega16 atmega161 atmega162 atmega163 atmega164a atmega164p atmega164pa atmega165 atmega165a atmega165p atmega165pa atmega168 atmega168a atmega168p atmega168pa atmega168pb atmega16a atmega2560 atmega2561 atmega32 atmega32a atmega323 atmega324a atmega324p atmega324pa atmega325 atmega325a atmega325p atmega325pa atmega3250 atmega3250a atmega3250p atmega3250pa atmega328 atmega328p atmega48 atmega48a atmega48pa atmega48pb atmega48p atmega64 atmega64a atmega640 atmega644 atmega644a atmega644p atmega644pa atmega645 atmega645a atmega645p atmega6450 atmega6450a atmega6450p atmega8 atmega8a atmega88 atmega88a atmega88p atmega88pa atmega88pb atmega8515 atmega8535 tinyAVR Devices: attiny4 attiny5 attiny10 attiny11 [1] attiny12 [1] attiny13 attiny13a attiny15 [1] attiny20 attiny22 attiny24 attiny24a attiny25 attiny26 attiny261 attiny261a attiny28 [1] attiny2313 attiny2313a attiny40 attiny4313 attiny43u attiny44 attiny44a attiny441 attiny45 attiny461 attiny461a attiny48 attiny828 attiny84 attiny84a attiny841 attiny85 attiny861 attiny861a attiny87 attiny88 attiny1634 Automotive AVR Devices: atmega16m1 atmega32c1 atmega32m1 atmega64c1 atmega64m1 attiny167 ata5505 ata5272 ata5702m322 ata5782 ata5790 ata5790n ata5791 ata5795 ata5831 ata6612c ata6613c ata6614q ata6616c ata6617c ata664251 ata8210 ata8510 CAN AVR Devices: at90can32 at90can64 at90can128 LCD AVR Devices: atmega169 atmega169a atmega169p atmega169pa atmega329 atmega329a atmega329p atmega329pa atmega3290 atmega3290a atmega3290p atmega3290pa atmega649 atmega649a atmega6490 atmega6490a atmega6490p atmega649p Lighting AVR Devices: at90pwm1 at90pwm2 at90pwm2b at90pwm216 at90pwm3 at90pwm3b at90pwm316 at90pwm161 at90pwm81 Smart Battery AVR Devices: atmega8hva atmega16hva atmega16hva2 atmega16hvb atmega16hvbrevb atmega32hvb atmega32hvbrevb atmega64hve atmega64hve2 atmega406 USB AVR Devices: at90usb82 at90usb162 at90usb646 at90usb647 at90usb1286 at90usb1287 atmega8u2 atmega16u2 atmega16u4 atmega32u2 atmega32u4 atmega32u6 XMEGA Devices: atxmega8e5 atxmega16a4 atxmega16a4u atxmega16c4 atxmega16d4 atxmega16e5 atxmega32a4 atxmega32a4u atxmega32c3 atxmega32c4 atxmega32d3 atxmega32d4 atxmega32e5 atxmega64a1 atxmega64a1u atxmega64a3 atxmega64a3u atxmega64a4u atxmega64b1 atxmega64b3 atxmega64c3 atxmega64d3 atxmega64d4 atxmega128a1 atxmega128a1u atxmega128a3 atxmega128a3u atxmega128a4u atxmega128b1 atxmega128b3 atxmega128c3 atxmega128d3 atxmega128d4 atxmega192a3 atxmega192a3u atxmega192c3 atxmega192d3 atxmega256a3 atxmega256a3u atxmega256a3b atxmega256a3bu atxmega256c3 atxmega256d3 atxmega384c3 atxmega384d3 Wireless AVR devices: atmega644rfr2 atmega64rfr2 atmega128rfa1 atmega1284rfr2 atmega128rfr2 atmega2564rfr2 atmega256rfr2 Miscellaneous Devices: at94K [2] at76c711 [3] at43usb320 at43usb355 at86rf401 at90scr100 ata6285 ata6286 ata6289 m3000 [4] Classic AVR Devices: at90s1200 [1] at90s2313 at90s2323 at90s2333 at90s2343 at90s4414 at90s4433 at90s4434 at90s8515 at90c8534 at90s8535 Note: [1] Assembly only. There is no direct support for these devices to be programmed in C since they do not have a RAM based stack. Still, it could be possible to program them in C, see the FAQ for an option. Note: [2] The at94K devices are a combination of FPGA and AVR microcontroller. [TRoth-2002/11/12: Not sure of the level of support for these. More information would be welcomed.] Note: [3] The at76c711 is a USB to fast serial interface bridge chip using an AVR core. Note: [4] The m3000 is a motor controller AVR ASIC from Intelligent Motion Systems (IMS) / Schneider Electric. avr-libc can be freely used and redistributed, provided the following license conditions are met. Portions of avr-libc are Copyright (c) 1999-2016 Werner Boellmann, Dean Camera, Pieter Conradie, Brian Dean, Keith Gudger, Wouter van Gulik, Bjoern Haase, Steinar Haugen, Peter Jansen, Reinhard Jessich, Magnus Johansson, Harald Kipp, Carlos Lamas, Cliff Lawson, Artur Lipowski, Marek Michalkiewicz, Todd C. Miller, Rich Neswold, Colin O'Flynn, Bob Paddock, Andrey Pashchenko, Reiner Patommel, Florin-Viorel Petrov, Alexander Popov, Michael Rickman, Theodore A. Roth, Juergen Schilling, Philip Soeberg, Anatoly Sokolov, Nils Kristian Strom, Michael Stumpf, Stefan Swanepoel, Helmut Wallner, Eric B. Weddington, Joerg Wunsch, Dmitry Xmelkov, Atmel Corporation, egnite Software GmbH, The Regents of the University of California. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. * Neither the name of the copyright holders nor the names of contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. AVR-LibC is a Free Software project whose goal is to provide a high quality C library for use with GCC on Microchip (formerly Atmel) AVR microcontrollers. Together, AVR-binutils, AVR-GCC, and AVR-LibC form the heart of the Free Opensource Software toolchain for the Microchip AVR microcontrollers. AVR-LibC is licensed under a single unified license. This so-called modified Berkeley license is intented to be compatible with most Free Software licenses like the GPL, yet impose as little restrictions for the use of the library in closed-source commercial applications as possible.