NAME

avarice − Provides an interface from avr-gdb to Atmel’s JTAGICE box.

SYNOPSIS

avarice [*OPTIONS*]... [[*HOST_NAME*]:*PORT*]

DESCRIPTION

AVaRICE runs on a POSIX machine and connects to gdb via a TCP socket and communicates via gdb’s "serial debug protocol". This protocol allows gdb to send commands like "set/remove breakpoint" and "read/write memory".

AVaRICE translates these commands into the Atmel protocol used to control the AVR JTAG ICE. Connection to the AVR JTAG ICE is via a serial port on the POSIX machine.

Because the GDB <---> AVaRICE connection is via a TCP socket, the two programs do not need to run on the same machine. In an office environment, this allows a developer to debug a target in the lab from the comfort of their cube (or even better, their home!)

NOTE: Even though you can run **avarice** and **avr−gdb** on different systems, it is not recommended because of the security risk involved. **avarice** was not designed to be a secure server. There is no authentication performed when a client connects to **avarice** when it is running in gdb server mode.

Supported Devices

avarice currently has support for the following devices:

atmega16
atmega162
atmega169
atmega323
atmega32
atmega64
atmega128
at90can128
atmega164p (**o**)
atmega324p (**o**)
atmega644 (*****)
atmega329 (**o**)
atmega3290 (**o**)
atmega649 (*****)
atmega6490 (*****)
atmega640 (*****)
atmega1280 (*****)
atmega1281 (*****)
atmega2560 (*****)
atmega2561 (*****)
at90usb1287 (*****)
atmega48 (**o**) (+)
atmega88 (**o**) (+)
atmega168 (**o**) (+)
attiny13 (**o**) (+)
attiny2313 (**o**) (+)
at90pwm2 (**o**) (+)
at90pwm3 (**o**) (+)
attiny24 (**o**) (+)
attiny44 (**o**) (+)
attiny25 (**o**) (+)
attiny45 (**o**) (+)
attiny261 (**o**) (+)
attiny461 (**o**) (+)

attiny861 (**o**) (+)

***** − Only supported by the JTAG ICE mkII device.
o − Only supported by the JTAG ICE mkII and AVR Dragon device.
+ − debugWire, see below

Supported File Formats

avarice uses libbfd for reading input files. As such, it can handle any file format that libbfd knowns about. This includes the Intel Hex, Motorola SRecord and ELF formats, among others. If you tell **avarice** to read an ELF file, it will automatically handle programming all of the sections contained in the file (e.g. flash, eeprom, etc.).

OPTIONS

−h, **−−help**
Print this message.

−1, **−−mkI**
Connect to JTAG ICE mkI (default).

−2, **−−mkII**
Connect to JTAG ICE mkII.

−B, **−−jtag-bitrate** <rate>
Set the bitrate that the JTAG box communicates with the AVR target device. This must be less than 1/4 of the frequency of the target. Valid values are 1 MHz, 500 kHz, 250 kHz or 125 kHz for the JTAG ICE mkI, anything between 22 kHz through approximately 6400 kHz for the JTAG ICE mkII. (default: 1 MHz)

−C, **−−capture**
Capture running program.
Note: debugging must have been enabled prior to starting the program. (e.g., by running avarice earlier)

−c, **−−daisy-chain** <ub,ua,bb,ba>
Setup JTAG daisy-chain information.
Four comma-separated parameters need to be provided, corresponding to *units before*, *units after*, *bits before*, and *bits after*.

−D, **−−detach**
Detach once synced with JTAG ICE

−d, **−−debug**
Enable printing of debug information.

−e, **−−erase**
Erase target.

−E, **−−event <eventlist>**
List of events that do not interrupt. JTAG ICE mkII and AVR Dragon only. Default is "none,run,target_power_on,wakeup"

−f, **−−file** <filename>
Specify a file for use with the --program and --verify options. If --file is passed and neither --program or --verify are given then --program is implied.

−g, **−−dragon**
Connect to an AVR Dragon. This option implies the **-2** option.

−I, **−−ignore-intr**
Automatically step over interrupts.
Note: EXPERIMENTAL. Can not currently handle devices fused for compatibility.

−j, **−−jtag** <devname>

Port attached to JTAG box (default: /dev/avrjtag). If the JTAG_DEV environmental variable is set, avarice will use that as the default instead.

If **avarice** has been configured with libusb support, the JTAG ICE mkII can be connected through USB. In that case, the string *usb* is used as the name of the device. If there are multiple JTAG ICE mkII devices connected to the system through USB, this string may be followed by the (trailing part of the) ICE's serial number, delimited from the *usb* by a colon.

The AVR Dragon can only be connected through USB, so this option defaults to "usb" in that case.

−L, **−−write-lockbits** <ll>

Write lock bits. The lock byte data must be given in two digit hexidecimal format with zero padding if needed.

−l, **−−read-lockbits**

Read the lock bits from the target. The individual bits are also displayed with names.

−P, **−−part** <name>

Target device name (e.g. atmega16)

−p, **−−program**

Program the target. Binary filename must be specified with --file option.

NOTE: The old behaviour of automatically erasing the target before programming is no longer done. You must explicitly give the --erase option for the target to be erased.

−r, **−−read-fuses**

Read fuses bytes.

−V, **−−version**

Print version information.

−v, **−−verify**

Verify program in device against file specified with --file option.

−w, **−−debugwire**

Connect to JTAG ICE mkII (or AVR Dragon), talking debugWire protocol to the target. This option implies the **-2** option. See the DEBUGWIRE section below.

−W, **−−write-fuses** <eehhll>

Write fuses bytes. **ee** is the extended fuse byte, **hh** is the high fuse byte and **ll** is the low fuse byte. The fuse byte data must be given in two digit hexidecimal format with zero padding if needed. All three bytes must currently be given.

NOTE: Current, if the target device doesn't have an extended fuse byte (e.g. the atmega16), the you should set ee==ll when writing the fuse bytes.

HOST_NAME defaults to 0.0.0.0 (listen on any interface) if not given.

:*PORT* is required to put avarice into gdb server mode.

EXAMPLE USAGE

```
avarice --erase --program --file test.bin --jtag /dev/ttyS0 :4242
```

Program the file *test.bin* into the JTAG ICE (mkI) connected to /dev/ttyS0 after erasing the device, then listen in GDB mode on the local port 4242.

```
avarice --jtag usb:1234 --mkII :4242
```

Connect to the JTAG ICE mkII attached to USB which serial number ends in *1234*, and listen in GDB mode on local port 4242.

DEBUGGING WITH AVARICE

The JTAG ICE debugging environment has a few restrictions and changes:

- No "soft" breakpoints, and only three hardware breakpoints. The break command sets hardware breakpoints. The easiest way to deal with this restriction is to enable and disable breakpoints as needed.

- Two 1-byte hardware watchpoints (but each hardware watchpoint takes away one hardware breakpoint). If you set a watchpoint on a variable which takes more than one byte, execution will be abysmally slow. Instead it is better to do the following:

  ```
  watch *(char *)&myvariable
  ```

  which watches the least significant byte of **myvariable**.
- The Atmel AVR processors have a Harvard architecture (separate code and data buses). To distinguish data address 0 from code address 0, **avr-gdb** adds 0x800000 to all data addresses. Bear this in mind when examining printed pointers, or when passing absolute addresses to gdb commands.

DEBUGWIRE

The *debugWire* protocol is a proprietary protocol introduced by Atmel to allow debugging small AVR controllers that don't offer enough pins (and enough chip resources) to implement full JTAG. The communication takes place over the */RESET* pin which needs to be turned into a debugWire connection pin by programming the *DWEN* fuse (debugWire enable), using a normal programmer connection (in-system programming, high-voltage programming). Note that by enabling this fuse, the standard reset functionality of that pin will be lost, so any in-system programming will cease to work as it requires a functional */RESET* pin. Thus it should be made **absolutely sure there is a way back**, like a device (as the STK500, for example) that can handle high-voltage programming of the AVR. Currently, **avarice** offers no option to turn off the DWEN fuse. However, **avrdude** offers the option to turn it off either through high-voltage programming, or by using the JTAG ICE mkII to first turn the target into an ISP-compatible mode, and then using normal ISP commands to change the fuse settings.

Note that the debugWire environment is further limited, compared to JTAG. It does not offer hardware breakpoints, so all breakpoints have to be implemented as software breakpoints by rewriting flash pages using *BREAK* instructions. (Software breakpoints are currently not implemented by **avarice**.) Some memory spaces (fuse and lock bits) are not accessible through the debugWire protocol.

SEE ALSO

gdb(1), **avrdude**(1), **avr−gdb**(1), **insight**(1), **avr−insight**(1), **ice−gdb**(1), **ice−insight**(1)

AUTHORS

Avarice (up to version 1.5) was originally written by Scott Finneran with help from Peter Jansen. They did the work of figuring out the jtagice communication protocol before Atmel released the spec (appnote AVR060).

David Gay made major improvements bringing avarice up to 2.0.

Joerg Wunsch reworked the code to abstract the JTAG ICE communication from the remainder, and then extended the code to support the JTAG ICE mkII protocol (see Atmel appnote AVR067).