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 (experimental)

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.

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

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

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

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

−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.

−j, **−−jtag** <devname>
Port attached to JTAG box (default: /dev/avrjtag).

−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 1MHz, 500KHz, 250KHz or 125KHz. (default: 1MHz)

−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.

−v, **−−verify**
Verify program in device against file specified with --file option.

−e, **−−erase**
Erase target.

−r, **−−read-fuses**
Read fuses bytes.

−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.

−l, **−−read-lockbits**
Read the lock bits from the target. The individual bits are also displayed with names.

−L, **−−write-lockbits** <ll>
Write lock bits. The lock byte data must be given in two digit hexidecimal format with zero padding if needed.

−P, **−−part** <name>
Target device name (e.g. atmega16)

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
```

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.

SEE ALSO

gdb(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.