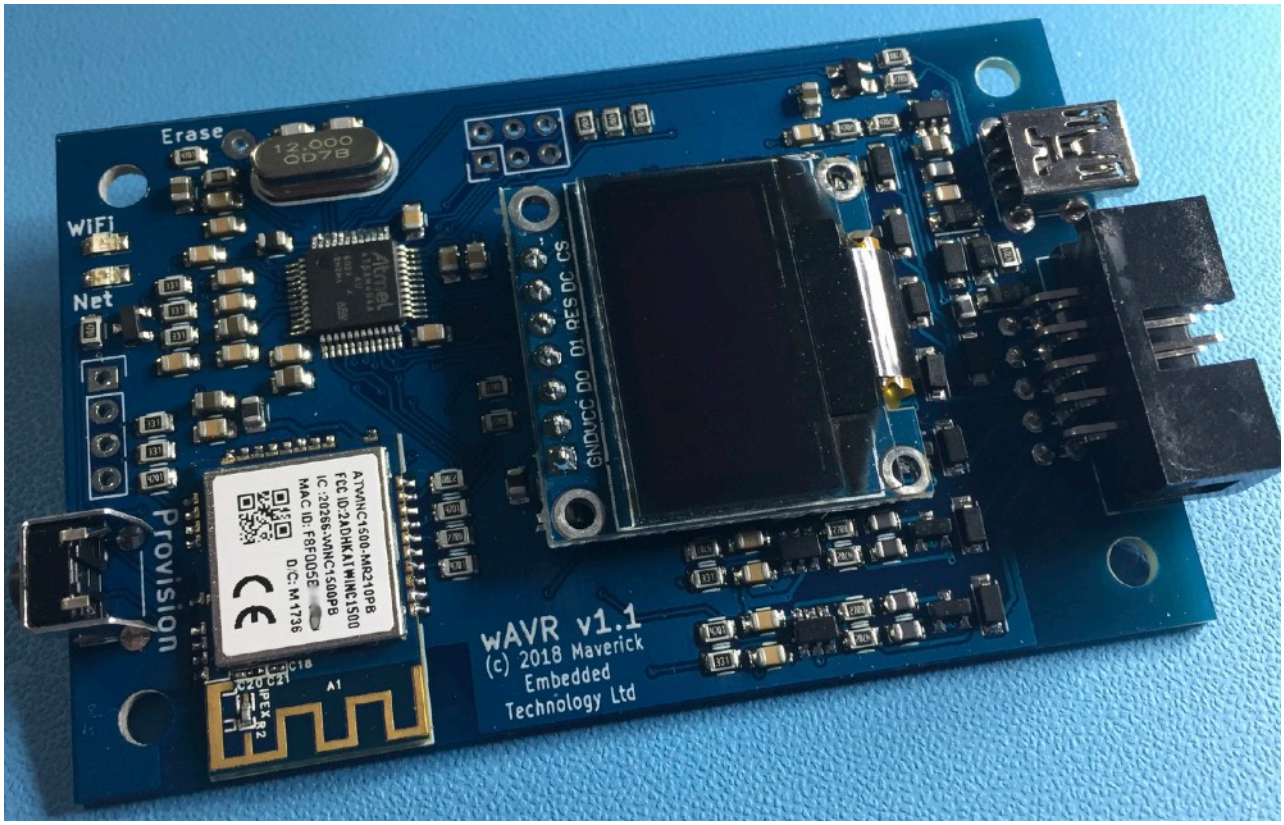
Maven



Maverick Embedded Technology Ltd

Introduction

Thank you for purchasing Maven by Maverick Embedded Technology Ltd. The Maven Programmer/Debugger is a small WiFi-enabled in-system programming and debugging device for ARM Cortex-M micro-controllers.

Features

Numerous significant features set it apart from other ARM programmers/debuggers:

- Built-in GDB server accessible over WiFi means Maven works with any network-connected host system which can run GDB for ARM.
- Does not require middleware running on the host, such as OpenOCD, and the associated configuration file(s).
- Supports Cortex-M TRACESWO serial data in NRZ mode at up to 3MHz.
- Supports target voltages between 1.65 volts and 5.5 volts.
- Communicates with your target using RS232 on a UART or bit-banged I/O pin. Maven will make the UART data available over WiFi using the telnet command on your host. Both RxD and TxD are supported at all the common baud rates.
- Maven's OLED display keeps you informed of both its status and various target parameters. It can also be configured to show the RS232 data received from the target.
- The USB interface provides two CDC-compatible RS232 interfaces. One of those is dedicated to TRACVESWO data. The other provides access to the same target UART interface mentioned above.
- All I/O signals between Maven and your target are fully protected against electrostatic discharge, over-voltage and reverse voltage.
- In many cases Maven can be powered by your target. Only when your target voltage is below around 3.1 volts will Maven need a separate power connection. Maven will show a message on the OLED display if its power-supply voltage is too low for reliable operation.
- Firmware updates for Maven can be applied very easily over WiFi using a simple update program. Note that Maven does not "phone home" to detect new firmware updates, or for any other reason.
- Maven's hardware is identical to the original wAVR board for programming Atmel/ Microchip AVR micro-controllers. A Maven board can be converted to a wAVR, and vice-versa, using a simple firmware update.

2

Supported ARM Micro-Controllers

Maven's current firmware supports ARM Cortex-M micro-controllers from Microchip Inc (previously Atmel), and a selection of devices from ST Micro. This includes both the CPU core itself, memory layout (address of RAM/ROM/Flash) and programming of the on-chip Flash memory.

Cortex-M0/M0+

- Microchip SAMC20, SAMC21
- Microchip SAMD09, SAMD1x, SAMD2x
- Microchip SAMDA1
- Microchip SAML21, SAML22
- Microchip SAMR2x, SAMR3x
- STMicro STM32F0 series.

Cortex-M3

- Microship SAM3A, SAM3N, SAM3S, SAM3U, SAM3X
- STMicro STM32F1 series.

Cortex-M4

- Microchip SAM4E, SAM4N, SAM4S
- Microchip SAMD5x, SAME5x,
- Microchip SAMG51, SAMG53, SAMG54, SAMG55
- STMicro STM32F4 series.

Cortex-M7

- SAME70, SAMS70, SAMV70, SAMV71
- STMicro STM32F7 series.

Support for ARM Cortex-M micro-controllers from other manufacturers will be added in future firmware updates.

3 GDB Server Support

Maven's primary purpose is to act as a GDB server on behalf of the target micro-controller. In this role, the Gnu Debugger (GDB), or IDE supporting the GDB remote protocol, running on a host computer can control and inspect the state of the target's CPU core and its connected peripherals using the GDB 'extended-remote' target protocol. Maven translates instructions received from the debugger into the appropriate register and memory read/write requests over the ARM CPU's ADIv5 interface known as the Serial Wire Debug Port (SWD-DP).

Maven supports breakpoints both in RAM and in Flash memory. The latter utilising Cortex-M hardware breakpoints where available - most Cortex-M cores have around 6 hardware breakpoints. Maven supports up to 128 software breakpoints for code running in RAM. In most cases GDB will choose the appropriate type of breakpoint automatically, since Maven informs GDB of the address/size/type of memory regions for supported targets.

Hardware watchpoints are also supported, where available on the target Cortex-M core. These can be used to halt execution when firmware accesses specific memory addresses, and are an invaluable tool to help track down memory corruption and/or use-after-free type bugs.

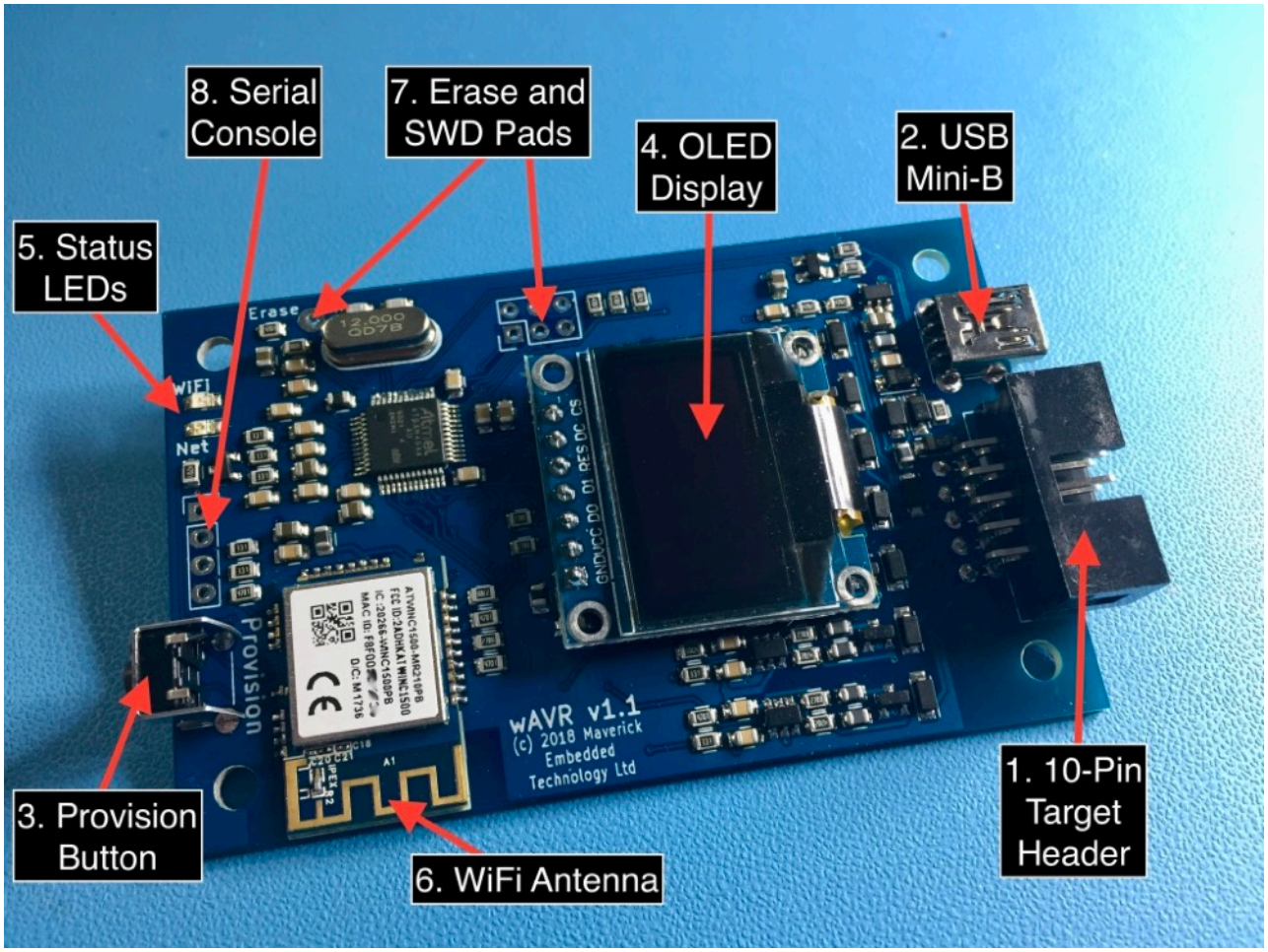
For more information on debugging embedded platforms with GDB, see:

https://sourceware.org/gdb/onlinedocs/gdb/Remote-Debugging.html

4

What's what?

A Quick Tour of Maven



1. 10-pin Target header

Connects Maven to the target device using the supplied cable and paddle board. The pins are assigned as follows:

Pin	Function
1	TRACESWO
2	Target Vcc
3	SWDIO
4	SWCLK
5	nRST (Target Reset pin)
6	Gnd

Pin	Function
7	Target Debug UART TxD
8	Current Maven firmware does not use this pin. It may be re-purposed as a user-controlled GPIO signal in a future firmware update.
9	Target Debug UART RxD
10	Gnd

Note that Maven can be powered by the target via pins 2 and 6. Your target must be able to provide at least 3.3 volts at a constant 150mA. Peak current demand is around 460mA with an average duty cycle of 15% when WiFi is active. The voltage on pin 2 should not exceed 5.5 volts.

If your target is incapable of meeting Maven's power requirements you will see a warning on the display. In this situation you should provide 5v power via the USB connector.

2. USB Connector

Provides additional 5 volt power when the target Vcc is below about 3.2 volts or the target is unable to meet the current demands of Maven.

The USB port also instantiates two CDC USB serial ports on compatible operating systems (for Windows, you'll need Windows 10 and above). The first CDC serial port instance optionally provides access to the raw TRACESWO data, if enabled. The second instance provides access to the target UART. Since the latter is primarily available over WiFi it is not expected that it will be used often. The former can be pressed into service to deal with high speed SWO data from the ARM TPIU.

3. Provision Switch

Press and hold the provision switch for 5 seconds to place Maven into WiFi Provision Mode. This allows you to change the WiFi network Maven connects to. See section 3.2 for details.

Press and hold the provision switch for 2 seconds to show to the device's IP address on the OLED display in a large, readable font.

4. OLED Display

The display is used to show status of Maven and the connected target. The top line always shows WiFi signal strength, USB and Network activity indicators, and the device's IPv4 address.

The second line shows the current target Vcc and Maven's current Vcc. If Maven's Vcc drops below 2.7 volts the second status line will show "LOW" next to the Vcc. This is your cue to connect Maven to external power via the USB connector.

The remaining four lines are used to show target details, Rx data from the target UART, or TRACESWO ITM.

5. Status LEDs

Colour	Name	Purpose
Green	WiFi	Indicates WiFi connection state. When connected to an Access Point the LED is on. When in Provision mode the LED blinks. Otherwise the LED is off to indicate no WiFi connection established.
Amber	Net	Blinks to indicate network activity over WiFi.

6. WiFi Antenna

Ensure this is not obstructed for best WiFi performance.

7. Erase and SWD pads

These are used during manufacturing and serve no end-user purpose. You should be especially careful not to connect the Erase pad to Vcc during power-up. This will completely erase the CPU's firmware and boot loader. Your device will need to be returned to Maverick Embedded Technology, at your expense, for re-imaging.

8. Serial Console

These pads serve no end-user purpose but an adventurous user might be tempted to connect the pins to see what they do. The pins provide access to the ARM CPU's debug serial console. There is a shell running on the console, providing exactly the same features as the shell available over the network (documented below) so there's really no point connecting to it. However, the pin-out is as follows:

Pin	Function	Comment
1 (Square pad)	RxD	Rx data to the ARM CPU. Connect to pin 4 briefly at power-on to force entry to the boot-loader.
2	TxD	Tx data from the ARM CPU. Baud rate is 38400, no parity, one stop bit.
3	Not Connected	
4	Gnd	Connect to pin 1 briefly at power-on to force entry to the boot-loader.

Note: TxD and RxD ***must not*** be driven at a higher voltage than the ARM CPU's Vcc. This is nominally 3.3 volts but may be less if Maven is drawing power from the Target Vcc pin. You should be especially careful not to back-feed power via TxD/RxD when Maven is powered off. Damage may ensue! In short, use of the debug serial console is not recommended.

5

Setup

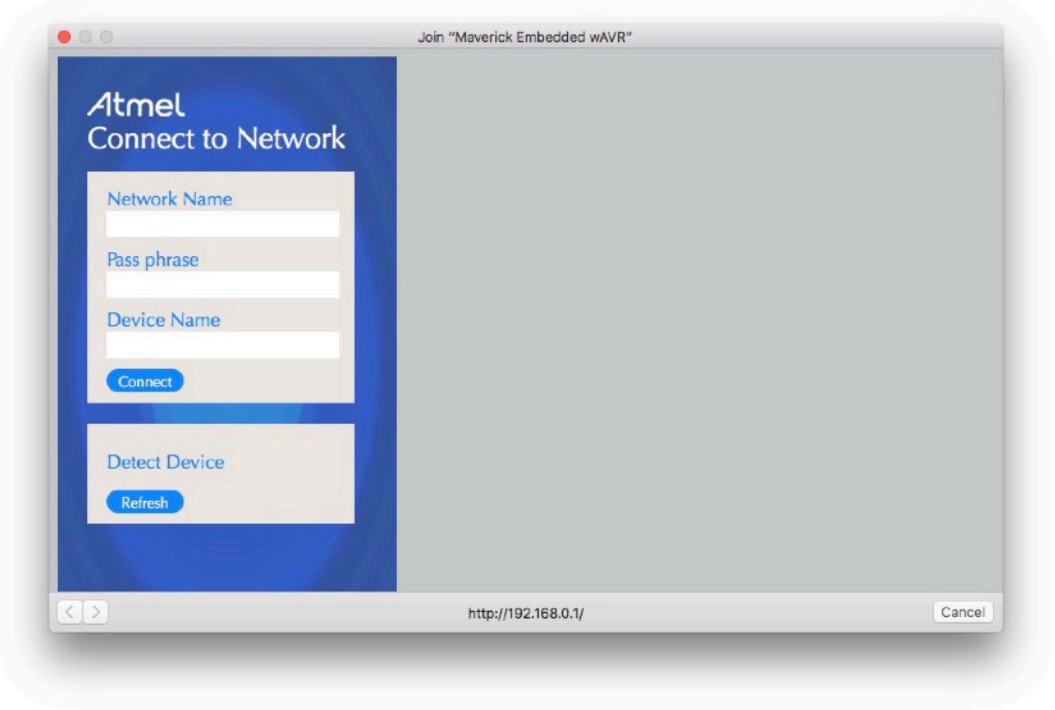
WiFi Provisioning

Before it can be used over WiFi, you must configure Maven to join your wireless network. The recommended way to do this is to power the device via USB.

When powered up for the first time, Maven will start in “Provisioning" mode. In this mode, Maven will act as a WiFi Access Point with the SSID "Maverick Embedded" and no password.

From your PC or smart phone, connect to "Maverick Embedded”. Depending on your PC's operating system you will either be presented with a captive network sign-on screen, or you will have to open up a web browser and navigate to http://192.168.0.1/

Use the sign-on screen to enter the details of your own WiFi network, as shown below.



Enter the Network Name (SSID) of your WiFi network and its associated pass phrase/key. Note that Maven supports WPA/WPA2 security only. WEP is not supported.

The Device Name field is optional. It is passed to your network's DHCP server as part of IP address negotiation and could be used to identify Maven in the DHCP server log files if they are available.

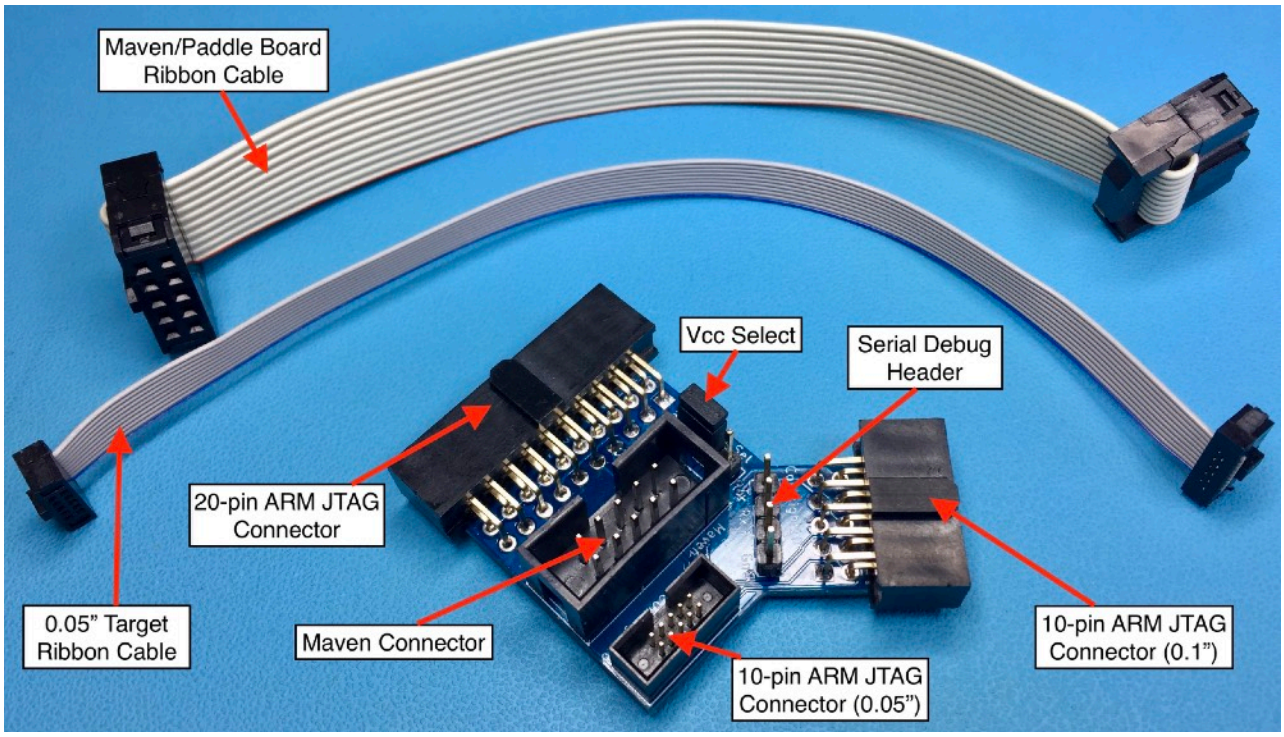
After clicking "Connect", Maven will switch off Access Point mode and attempt to connect to the configured WiFi network.

6

Connecting the Target

Maven is supplied with three accessories for connecting ARM targets.

- A standard 0.1 inch pitch ribbon cable with a 10-pin IDC plug on each end. This is designed to connect Maven to the supplied paddle board.
- A 0.05 inch pitch ribbon cable with a 10-pin 0.05 inch pitch IDC plug on each end. This is intended for use with targets which have an onboard half-pitch CoreSight-10 connector. One end plugs into Maven's paddle board while the other plugs into the target.
- A paddle board which interfaces to three styles of ARM debug/JTAG connector and provides a header for the serial debug Tx/Rx pins. See the picture below for details.



The paddle board itself then breaks out the relevant signals to 5 separate connectors:

- *20-pin ARM JTAG Connector:* Plugs directly into target boards equipped with 20-pin 0.1" pitch debug headers.
- *10-pin ARM JTAG Connector:* Plugs directly into target boards equipped with 10-pin 0.1" pitch debug headers.
- *10-pin ARM JTAG Connector (0.05"): Connects to target boards equipped with a 10-pin 0.05" pitch debug header using the supplied 0.05" target ribbon cable.*

- *Vcc Select:* This jumper determines which Vcc pin of the 20-pin JTAG connector is used to provide Target Vcc to Maven. The default (shown) corresponds to pin 1 of the JTAG connector. However on some targets this pin is supplied via a resistor and so would not provide enough power to Maven should you be powering Maven from the target. The standard also suggests that pin 2 of the 20-pin JTAG connector should be connected directly to Target Vcc specifically to power debug hardware. If your target is configured in this way, you should move the jumper to prove Target Vcc via pin 2. Note that the jumper does not affect Target Vcc from either of the 10-pin JTAG connectors.
- *Serial Debug Header:* Breaks out Maven's RS232 target serial debug signals. The function of each pin is marked on the paddle board. Pin 1 (nearest to the Vcc Select header) is TxD and corresponds to transmit data from Maven to the target. Pin 2, RxD, corresponds to receive data from the target to Maven. Pin 3 corresponds to pin-8 of Maven's 10-pin target connector. As mentioned earlier, a future firmware update may enable this pin as a user-controllable GPIO. Ground is provided via pin 4.

7

Connecting to Maven over WiFi

The OLED display will show the connection status in the top-right of the screen. If all goes well it will show an IPv4 address. This is the address by which Maven can be reached from your host PC.

You can verify the connection status using your host's “telnet" command. For example, if wAVR's IPv4 address is 192.168.1.50 then from a terminal window on your PC you can type `telnet 192.168.1.50`.

If all is well you will see the following:

```
Trying 192.168.1.50...
Connected to 192.168.1.50.
Escape character is '^]'.

Maven WiFi/USB Programmer/Debugger for ARM Cortex-M SoCs
Copyright (c) 2019-2010 Maverick Embedded Technology Ltd.

Firmware Version: 1.5.0
Firmware Build Date: May 19 2020, 14:35:43

Maven>
```

This is Maven’s simple command-line "shell" from which you can monitor and configure various aspects of Maven’s behaviour. To exit the shell type `CTRL-]` followed by `quit`. The commands available from the shell are described in more detail later in this document.

Maven’s shell commands are also available from within GDB using the “monitor” command. Anything prefixed with “monitor” is passed to Maven and executed just as if it had been typed at Maven’s shell prompt. Command output is passed back to GDB and displayed like any other GDB command.

Using Maven with GDB

Maven was designed from the outset to work with the open-source Gnu Debugger, GDB. Other debuggers which support the GDB remote protocol can also be used. If you already use ARM GDB with a different programmer then a simple change to your .gdbinit file is all that is needed to get up and running with Maven. The minimum recommended version of GDB is 8.0.50. Earlier versions will probably work, but have not been extensively tested with Maven.

To configure GDB for Maven, add the following line to your .gdbinit file:

```
target extended-remote 192.168.2.50:3333
```

Substitute the actual IPv4 address of your Maven as required.

Now whenever you start a GDB session it will automatically connect to Maven and communicate with the target.

You can also use GDB to program Flash memory on the target device; either as a standalone programmer or as part of your interactive debugging session.

For standalone programming, create a simple shell script:

```
#!/bin/sh
target=192.168.2.50:3333
bin="${1}"
if [ "$2" != "" ]; then
        target="$2:3333"
fi
arm-none-eabi-gdb -nx --batch -ex "target extended-remote
${target}" -ex "load ${bin}" -ex 'kill' -ex quit
```

Place the script somewhere in your search PATH. Invoke the script with the first parameter set to the firmware image filename. The script has an optional 2nd parameter - the IP address of Maven. Without the 2nd parameter, the default `target` specified in the script is used.

To program Flash from within GDB as part of your debugging procedure, start GDB as normal with the firmware image executable supplied on the command line. At GDB's prompt, issue the `load` command before any other debug commands. Your firmware image will now be in Flash and the program counter will point to the reset vector.

Maven supports GDB's `monitor` command. All command line text following the `monitor` keyword is passed to Maven for interpretation by Maven's shell. Output from shell commands is passed back to GDB for display on its console.

See https://sourceware.org/gdb/onlinedocs/gdb/ for more information.

Connecting to the Target UART

A very useful feature of Maven is the ability to send and receive RS232 serial data to and from your target device over the network using your host's "`telnet`" command.

Simply connect the paddle board's TxD/RxD/Gnd pins to the RxD/TxD/Gnd pins of your target. Note that Maven's UART operates at your target's logic voltage levels rather than true RS232 levels as you'd find on a real RS232 serial port. Connecting Maven to a real RS232 serial port risks damaging some components on Maven if the real serial port is able to drive a large current through Maven's over-voltage protection devices.

By default Maven uses a serial port speed of 38400 baud, 8 data bits, no parity and one stop bit. The speed and parity settings can be changed via Maven's command line shell however the number of data and stop bits are fixed due to limitations of the CPU's UART hardware. This should not be a problem as most applications will use the standard configuration.

To connect to the UART from the host:

```
$ telnet 192.168.1.50 2000
Trying 192.168.1.50...
Connected to 192.168.1.50.
Escape character is '^]'.
```

Assuming the target's baud rate and Maven's baud rate match you should now see whatever serial data is sent by the target. The received data will also be shown on the OLED display (this can be disabled by a shell command).

To exit the telnet session, type `CTRL-]` followed by `quit`.

8

Configuring Maven

The Built-in Shell

Maven has a simple command-line interface available over WiFi on TCP port 23 - the standard "telnet" port.

```
$ telnet 192.168.1.50

Trying 192.168.1.50...
Connected to 192.168.1.50.
Escape character is '^]'.

Maven>
```

The available commands are:

`adapter`	View/Modify debug port parameters
`display`	Manipulate the onboard display
`gdb_attach`	Configures GDB attach behaviour
`help`	Display this help
`net`	Display network status
`reboot`	Reboot Maven
`rstcfg`	Configure sRST behaviour
`semihost`	Configure semi-hosting
`serial`	Show serial number of your Maven device
`swo`	Configure TARGETSWO behaviour
`termserv`	Configure the terminal server
`uart`	Display/modify target UART configuration
`wifi`	Display/configure WiFi Info
`wincota`	Update WiFi module firmware Over-The-Air

Assuming you have provisioned the WiFi connection as described earlier, then the default settings for most of the above commands will be fine. The most common changes you are likely to make will be display orientation and UART baud rate.

When a valid target is connected, some additional commands are available:

`cortexm`	Configure Cortex-M behaviour
`gpnvm`	Clear/Set/Show GPNVM bits (Some SAM targets only)
`info`	Display target details
`protect`	Display/manipulate target protection (Not all targets)
`reset`	Performs a hard reset of the target

Note that the `gpnvm` command is available only for targets which support GPNVM bits, such as the Microchip SAM4S series. The `protect` command applies only to targets which have a configurable security level, such as STMicro STM32Fx series.

Command: `adapter`

`adapter`

Show current adapter configuration

`adaptor speed <value>`

Sets the SWD/JTAG clock speed for communicating with the target. The following values (between 1 and 9) are recognised (corresponding frequency is approximate):

1. `250 KHz`
2. `500 KHz`
3. `1.0 MHz`
4. `1.5 MHz`
5. `2.0 MHz`
6. `3.0 MHz`
7. `3.5 MHz`
8. `4.5 MHz`
9. `6.5 MHz`

The default is 2, meaning 500 KHz.

`adaptor idlecycles <value>`

Specifies the number of idle cycles inserted after completing an SWD transaction. Legal values are 0 to 255. The default is zero.

`adaptor turnaround <cycles>`

Specifies the number of clock cycles for the turnaround period when changing the SWDIO pin between input and output. Note that not all targets support changing this value. The status output of this command will display a warning if the current target did not honour the requested turnaround period. Legal values are 1 to 4. The default is one.

Changes to the above settings are persistent across reboots but will not affect the current connection with an existing target. Power-cycling or disconnecting/reconnecting the target will cause the changes to take effect.

Command: `display`

`display`

Show current display configuration.

`display say [text] ...`

Prints some text onto the display.

`display flip`

Flips the display orientation.

`display uart <"on" | "off">`

Enable/Disable showing Rx data from the target serial port on the attached display. Default is On.

`display rssi <"on" | "off">`

Controls whether the WiFi signal strength is shown as a simple bar graph or as the RSSI value provided by the WiFi controller.

Command: `gdb_attach`

`gdb_attach`

Show current GDB attach behaviour.

`gdb_attach reset`

When GDB attaches, reset the target and halt it on the reset vector. This is useful in conjunction with GDB's "load" command to download and run/debug a new firmware image.

`gdb_attach halt`

When GDB attaches, forcibly halt the target without performing a reset. This is useful when using GDB to "break" into a running program to determine its current state.

The setting is persistent; it will be preserved across Maven reboots.

Command: `help`

`help`

Show list of available commands.

Command: `net`

`net`

Shows current network configuration and socket states.

Note that wAVR does not support static IP configuration due to limitations in Atmel Software Framework's WiFi driver. Therefore TCP/IP configuration is performed via DHCP only.

Command: `reboot`

`reboot`

Immediately reboots Maven.

`reboot defaults yesplease`

Reboots Maven after erasing all stored settings. When the firmware restarts, all settings will be restored to factory default values.

`reboot bl`

Reboots into the boot loader. This is used when applying firmware updates over WiFi. Details will be provided with the update image. The only way to

exit the boot loader, other than uploading a valid firmware image, is to power-cycle Maven.

Command: `rstcfg`

`rstcfg`

Show current reset behaviour.

`rstcfg driver <option>`

Possible values for 'option' are:

- `opendrain` sRST driver is open-drain (default).
- `pushpull` sRST driver is push-pull.

`rstcfg extended <option>`

Possible values for 'option' are:

- `on` Perform extended reset sequence during attach.
- `off` Normal reset sequence during attach.

Note that the extended command may be required for some Microchip SAM devices which have a Device Service Unit (DSU).

Command: `serial`

`serial`

This will display the device's unique serial number. You will be asked to quote this number when purchasing firmware upgrades from Maverick Embedded Technology since paid-for upgrades are locked to a specific device.

Note: this applies only to up*grades*, not up*dates*. The latter are used to fix bugs and/or add simple new features. The former will be used for significant new value-added features.

Command: `swo`

`swo`

Display the current SWO configuration.

`swo termserve`

This is the simplest way to use the SWO pin for debug/diagnostic purposes. It is designed to work with the ITM_SendChar() function available in ARM CMSIS. When in this mode the usual target UART normally available via telnet to Maven is replaced by the data available from the SWO pin, minus the ITM header.

`swo usb`

This mode re-directs SWO data to Maven's USB port rather than the telnet server, and is designed to be used when the target's SWO port is generating a large amount of data, such as when using the DWT and ITM in conjunction with the TPIU. In this situation, you will likely have the SWO port configured for high speed, and the WiFi module will have insufficient bandwidth/buffering to prevent data loss.

`swo speed <baudrate>`

Set the baudrate for Maven's SWO port. The command accepts arbitrary values for 'baudrate' and will display the actual rate that will be used. This may not be exactly the rate you requested, particularly for rates above about 1MHz.

Note that only NRZ format SWO data is currently supported; Manchester format is expected in a future update.

Command: `termserv`

`termserv`

Display configuration for the terminal server.

`termserv crlf <on|off>.`

Enable or disable the addition of carriage return (CR) for each line-feed (LF) sent by the target.

Command: `uart`

`uart`

Display configuration and statistics for the Target UART.

`uart speed <baudrate>`

Set speed to the specified baud rate. Supported rates are: 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.

`uart fmt <character-format>`

Set character format. Specified as char SizeParityStopbits, where: size is always 8, parity is N, O, or E and stopbits is always 1.

For example "`8N1`" is character size 8, no parity, 1 stop.

Command: `wifi`

`wifi`

Display current WiFi settings.

`wifi ssid <wifi-network-name>`

Sets the name of the WiFi network to which Maven should connect.

Enclose the network name in double quotes, such as "`net name`" if it contains spaces.

`wifi key <encryption-type> <wifi-network-passphrase>`

Configure the encryption type and passphrase. Encryption type is one of "`wep`" or "`wpa`". Note that WEP is insecure and support is likely to be dropped in future firmware updates.

The `wifi-network-passphrase` is the password used to secure your WiFi network. Again, enclose the password in double-quotes if the password contains spaces.

`wifi name <dhcp-client-name>`

Specifies a human-friendly name for Maven. Most good DHCP servers will log the client name against the assigned IP address and may populate the local DNS with the name for ease of access. This is blank by default.

Command: `wincota`

`wincota`

Initiates an Over The Air update of firmware in the WINC1500 WiFi controller. The specified URL must be in the form:

http://server-name/firmware-file

where "server-name" will usually be winc-fw.maverick-embedded.co.uk and "firmware-file" specifies the location and name of the file on the server. The purpose of this command is to enable in-the-field patching of the WiFi controller in cases where serious WiFi security- related updates are needed, such as the 2017 WPA2 "Krack" incident. (Don't worry - the patch for that one has already been applied!) WARNING: You are very strongly cautioned against uploading a firmware image which has not been tested by Maverick Embedded Technology Ltd. Doing so could render your device completely unusable if the Maven driver in your device is incompatible with the new firmware. Recovering from this situation could entail desoldering and replacing the WiFi module!

Command: `cortexm`

`cortexm`

Display current configuration of features specific to the target's Cortex-M CPU core.

`cortexm maskisr <on | off>`

Configures interrupt masking behaviour during single-step. Default is on.

`cortexm vector_catch <vector> [...]`

Configures which error vectors to hook when target is running. Options for <vector> are:

`harderr`	Debug trap on HardFault exception
`interr`	Debug trap on fault during exception entry/exit
`buserr`	Debug trap on BusFault exception
`staterr`	Debug trap on UsageFault exception (state error)
`chkerr`	Debug trap on UsageFault exception (check error)
`nocperr`	Debug trap on UsageFault exception (copro error)
`mmerr`	Debug trap on MemManage exception
`all`	Enable all of the above

`none` Disable all of the above

The default is 'harderr'. You can specify multiple vectors, each separated by a space. Note that ARMv6-M devices (Cortex-M0, for example) only support debug trap on 'harderr'; all other trap types are ignored.

If execution branches to a hooked vector, Maven will halt the CPU core and report the status to GDB.

Command: `gpnvm`

`gpnvm`

Available when supported by the target, this command shows the current status of GPNVM bits. For example, the output for a SAM4S4A SoC (as used by Maven itself):

```
GPNVM0: 0 (Security Bit: Disabled)

GPNVM1: 1 (Boot Mode: Flash)
```

The first field shows the GPNVM bit name. The second field shows the current value for the bit. The third field describes the function of the bit, where appropriate.

`gpnvm [set | clear] <bit-number>`

Set or clear the specified GPNVM bit number. The bit number is usually 0 or 1 but can be 2 in devices with larger Flash memory. The current status will indicate how may bits are implemented.

Command: `info`

`info`

Displays some useful information about the target SoC. For example, the information shown for the SAM4S4A SoC on Maven is:

```
Vendor: Microchip
Device: ATSAM4S4A
Additional Info: CHIP ID 288B09E0
CPU Core: Cortex-M4 (r0p1), ARMv7-M Mainline implementation
MPU: present
FPU: not present
Cache: not present
Hardware breakpoints: 6
Hardware watchpoints: 4
Memories:
  Flash: 00400000-0043ffff (256 KB
    ROM: 00800000-00807fff (32 KB, SAM-BA ROM)
```

```
    RAM: 20000000-2000ffff (64 KB)
Power Supply Voltages:
Maven: 3.3 volts
Target: 3.30 volts
```

Command: protect

`protect [level [permanent]]`

Available when supported by the target, with no parameters, this command displays the target's current protection level along with a description of the possible target-specific values for 'level'. If specified, 'level' is used to change the current protection level of the target. Note that this command can only increase the protection level. Use the 'unlock' command to remove protection, if possible. Specify '1' or 'set' to enable protection. Other non-zero values for 'level' are interpreted in a target-specific way, as described when the current protection level is displayed.

Note that changing the protection level may require the target to be power-cycled for the change to be effective. If 'level' is followed by the word "permanent" then, for some targets, the combination may render the target irreversibly locked, so be sure to understand the consequences of your actions!

Valid range for level is 0-2

Command: unlock

`unlock`

Available when a locked target is connected, this command will perform the necessary steps to unlock a device which has been protected from debugger access. In effect, it undos the results of the `protect` command. In most cases, this will cause the device's Flash and volatile memories to be completely erased, and is usually followed up with a device reset. However, if the device has been locked permanently (if supported by the device) then this command will have no effect.

7

Acknowledgements

wAVR firmware contains open-source software from several third parties. Their license details are included below.

The Display Driver

Universal 8bit Graphics Library (http://code.google.com/p/u8glib/)

GPIO Hardware Abstraction

Specifically `hal_gpio.h` by Alex Taradov (https://github.com/ataradov/mcu-starter-projects)

ARM Cortex-M4 CMSIS Headers

- Neither the name of ARM nor the names of its 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 COPYRIGHT HOLDERS AND 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.

Atmel Software Framework

Small portions of Atmel Software Framework are included in the firmware.

Copyright (c) 2012-2016 Atmel Corporation. All rights reserved.

Redistribution and use in source and binary forms, with or without

modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice,this list of conditions and the following disclaimer.
2. 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.
3. The name of Atmel may not be used to endorse or promote products derived from this software without specific prior written permission.
4. This software may only be redistributed and used in connection with an Atmel microcontroller product.

THIS SOFTWARE IS PROVIDED BY ATMEL "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT ARE EXPRESSLY AND SPECIFICALLY

DISCLAIMED. IN NO EVENT SHALL ATMEL 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.

FreeRTOS

wAVR Firmware uses a wholly unmodified version of FreeRTOS-10.0.0 which is licensed under the terms of the MIT Open Source License, below:

Copyright (C) 2017 Amazon.com, Inc. or its affiliates. All Rights Reserved. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software. If you wish to use our Amazon FreeRTOS name, please do so in a fair use way that does not cause confusion.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.