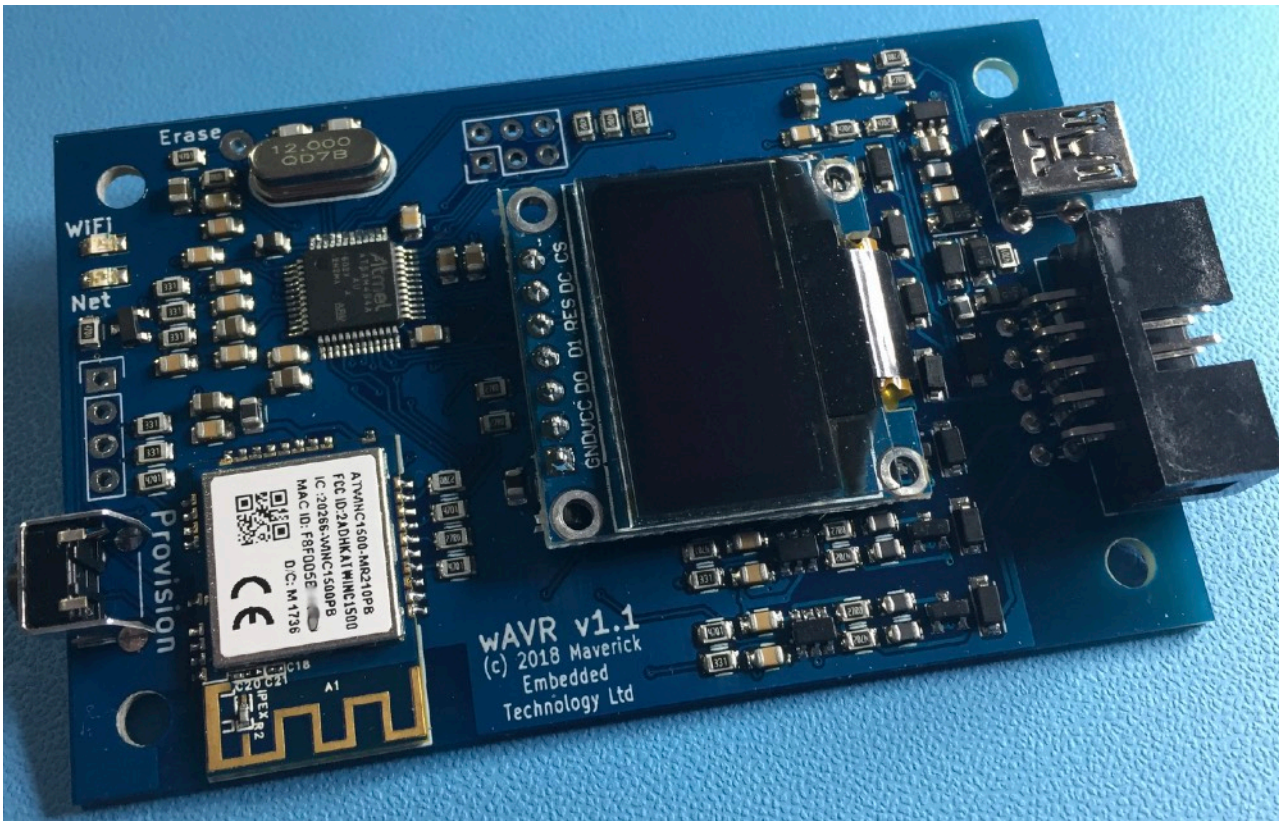
wAVR



Maverick Embedded Technology Ltd

Introduction

Thank you for purchasing wAVR by Maverick Embedded Technology Ltd. The wAVR Programmer is a small WiFi-enabled in-system programming device for the Microchip (previously Atmel) AVR™ range of micro-controllers. It supports AVR™ MCUs from the Tiny, Mega and XMega range using either ISP, PDI, or uPDI programming methods.

Features

Numerous significant features set it apart from other AVR™ programmers:

- Programs devices over your WiFi network using the open-source programmer “avrdude" (version 6.3 recommended).
- Programs devices over USB using "avrdude" or Atmel Studio.
- 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. wAVR 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.
- wAVR'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 can be used for programming - supporting the same protocols as the WiFi interface. The other provides access to the same target UART interface mentioned above.
- The USB interface can also be configured to emulate just enough of Atmel's STK600 programmer for Atmel Studio to be used to program your devices without the need to add avrdude as an external command.
- wAVR can provide your target with a recovery clock signal should you need it to reset the fuses on Tiny or Mega devices.
- All I/O signals between wAVR and your target are fully protected against electrostatic discharge, over-voltage and reverse voltage.
- In most cases wAVR can be powered by your target. Only when your target voltage is below around 3.1 volts will wAVR need a separate power connection. wAVR will show a message on the OLED display if its power-supply voltage is too low for reliable operation.
- Firmware updates for wAVR, should they be necessary, can be applied very easily over WiFi using a simple update program. Note that wAVR does not “phone home” to detect new firmware updates, or for any other reason.

2

Supported Protocols

wAVR

The ISP (for Tiny and Mega), PDI (XMega), and uPDI (latest Tiny MCUs) programming methods are supported in this mode. The mode is fully supported by the latest development version of avrdude over both WiFi and USB CDC serial port, after applying a small set of patches. The patches have been submitted to the avrdude developers in the hope that native wAVR support will be provided in a future release.

AVR™ Dragon

Both ISP (for Tiny and Mega) and PDI (XMega) programming are supported using this protocol. This mode does not support uPDI. The protocol is fully supported by avrdude over both WiFi and USB CDC serial port.

STK500v2/STK600

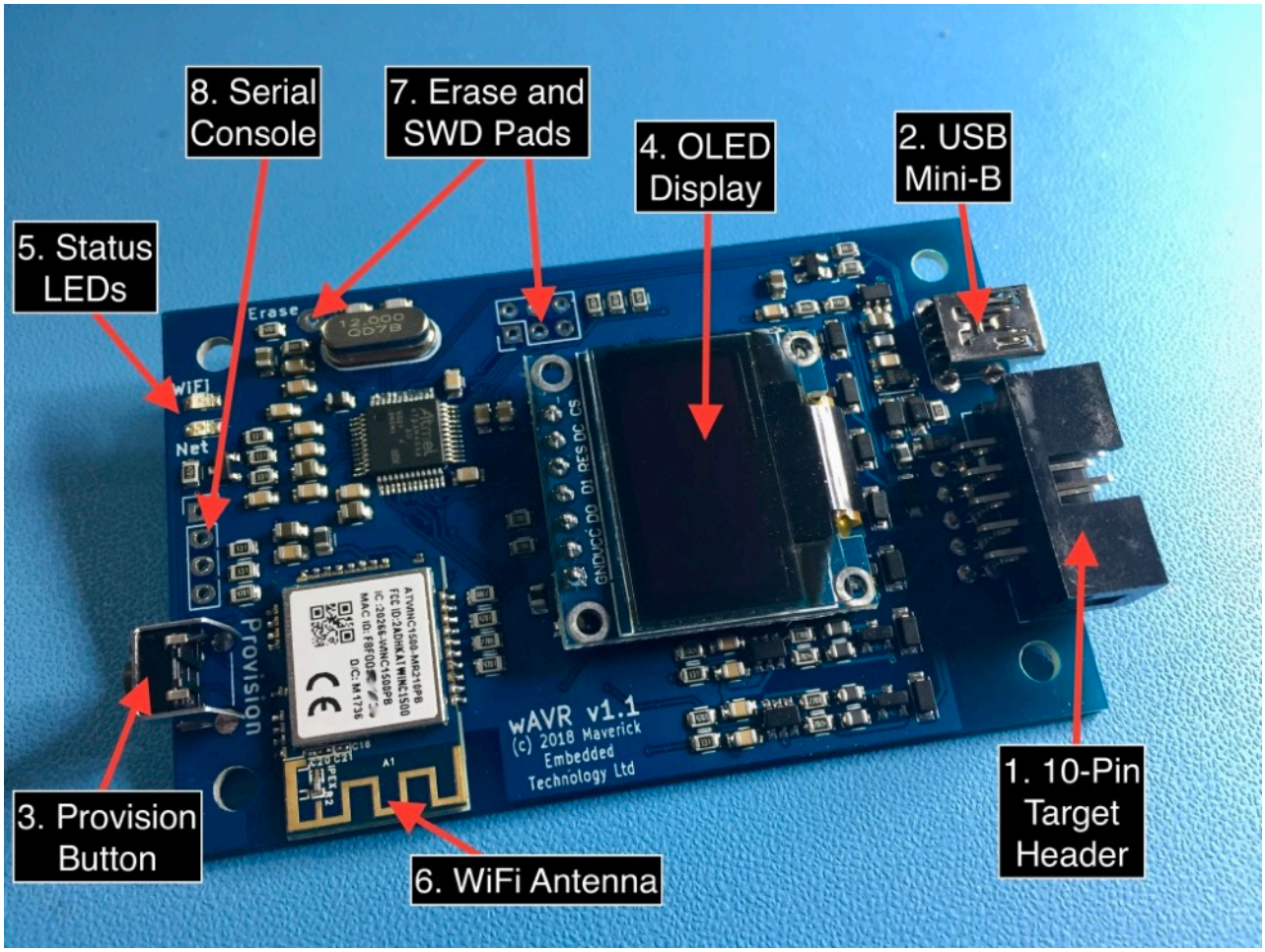
Both ISP and PDI programming are supported using this protocol. The protocol is fully supported by avrdude over both WiFi and USB CD serial port. Atmel Studio can also work with the STK500v2 emulation over USB however it requires installation of a USB CDC ".inf" file in order for Windows™ to recognise wAVR as a valid USB device, and only ISP programming is supported.

Alternatively you can switch wAVR's USB controller into STK600 emulation mode. This mode provides just enough emulation of a real STK600 to make driver installation unnecessary and for both ISP and PDI programming to function correctly.

3

What's what?

A Quick Tour of wAVR



1. 10-pin Target header

Connects wAVR to the target device using the supplied cable. The pins are assigned as follows:

Pin	ISP Mode	PDI Mode	uPDI Mode
1	MISO	PDI Data	uPDI Data
2	Target Vcc	Target Vcc	Target Vcc
3	SCK	Unused. 47KΩ pull-up to Target Vcc	Unused. 47KΩ pull-up to Target Vcc
4	MOSI	Unused. 47KΩ pull-up to Target Vcc	Unused. 47KΩ pull-up to Target Vcc

Pin	ISP Mode	PDI Mode	uPDI Mode
5	Reset	PDI Clock	Unused. 47KΩ pull-up to Target Vcc
6	Gnd	Gnd	Gnd
7	Debug UART TxD	Debug UART TxD	Debug UART TxD
8	Rescue Clock. 47KΩ pull-up to Target Vcc if not used.	Unused. 47KΩ pull-up to Target Vcc	Unused. 47KΩ pull-up to Target Vcc
9	Debug UART RxD	Debug UART RxD	Debug UART RxD
10	Gnd	Gnd	Gnd

The layout of pins 1-6 corresponds to Atmel's 6-pin ISP header standard. The remaining pins are specific to wAVR.

Note that wAVR 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 wAVR's power requirements you will see a warning on the display. In this situation you should provided 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 wAVR.

The USB port also instantiates two CDC USB serial ports on compatible operating systems (basically anything except Windows). The first CDC serial port instance provides access to wAVR's Dragon/STK500 compatible programming interface. The second instance provides access to the target UART. Since these interfaces are primarily available over WiFi it is not expected that they will be used often.

The USB port can also be switched into a mode which emulates the Atmel STK600 programmer. In this mode, the target UART port is unavailable over USB but it does allow wAVR to be used as a programming device in Atmel Studio on Windows.

3. Provision Switch

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

4. OLED Display

The display is used to show status of wAVR 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 wAVR's current Vcc. If wAVR's Vcc drops below 2.7 volts the second status line will show "LOW" next to wAVR Vcc. This is your cue to connect wAVR to external power via the USB connector.

When programming a target, the second status line will also show the current interface clock rate used while programming.

The remaining four lines are used to show Rx data from the target UART connection.

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.

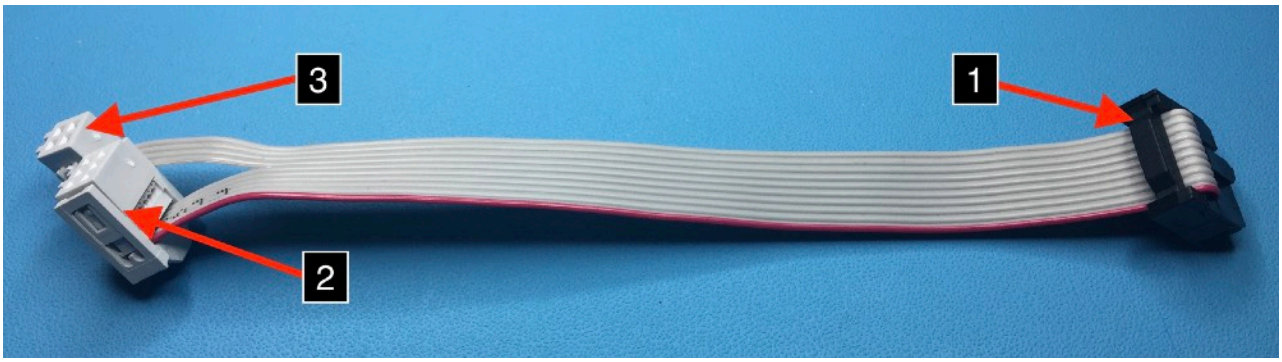
8. Serial Console

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	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 wAVR is drawing power from the Target Vcc pin. You should be especially careful not to back-feed power via TxD/RxD when wAVR is powered off. Damage may ensue! In short, use of the debug serial console is not recommended.

Programming Cable



1. 10-pin plug

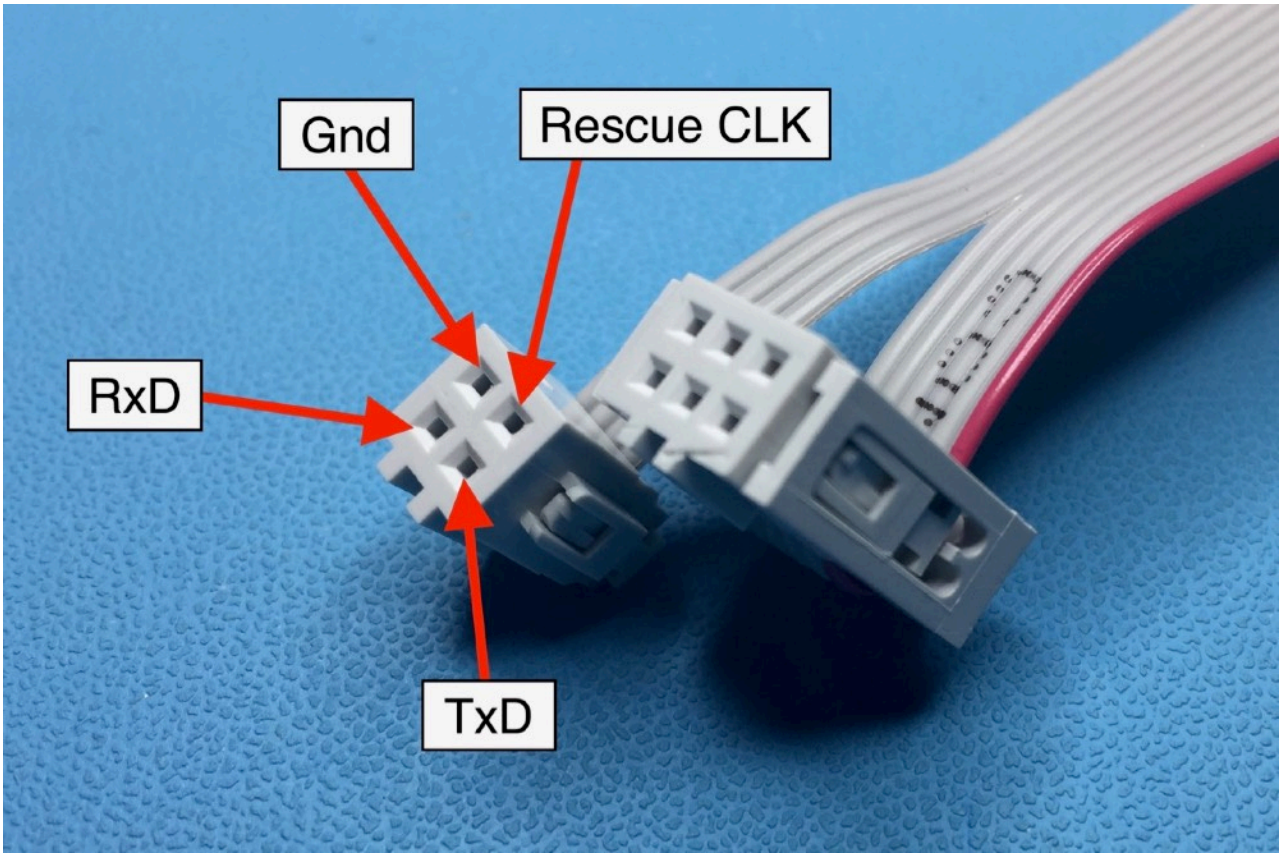
This connects to wAVR's 10-pin socket. Pin-out is as described earlier for wAVR's 10-pin header.

2. 6-pin plug

This connects to the standard Atmel 6-pin ISP/PDI/uPDI header of you target device. Pin-out is as described earlier for wAVR's 10-pin header, though only the first 6 pins apply.

3. 4-pin plug

This carries the target UART, rescue clock and Gnd signals. The pin-out is essentially the same as pins 7-10 of wAVR's 10-pin header:



4

Setup

WiFi Provisioning

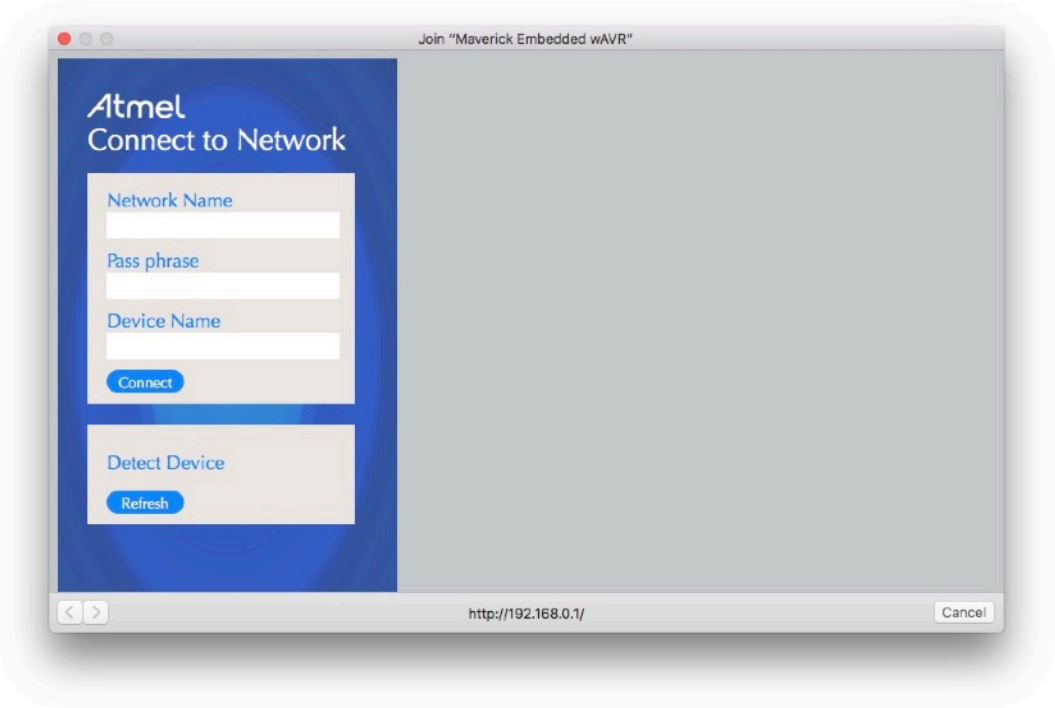
Before it can be used over WiFi, you must configure wAVR to join your wireless network. The recommended way to do this is to power the device via USB.

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

From your PC or smart phone, connect to "Maverick Embedded wAVR”. 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 wAVR 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 the wAVR device in the DHCP server log files if they are available.

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

5

Connecting to wAVR 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 wAVR 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 '^]'.

wAVR WiFi/USB Programmer for Microchip AVR MCUs
Copyright (c) 2018 Maverick Embedded Technology Ltd.

Firmware Version: 1.1.0
Firmware Build Date: Feb 15 2018, 17:28:37

NetShell>
```

This is wAVR's simple command-line "shell" from which you can monitor and configure various aspects of wAVR'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.

Using wAVR with avrdude

wAVR was designed from the outset to work with the open-source AVR programmer called "avrdude". If you already use avrdude-6.3 with a different programmer then a simple command-line change is all that is needed to get up and running with wAVR. The minimum recommended version of avrdude is 6.3. Earlier versions are not guaranteed to function correctly with wAVR.

Two command line options are relevant; "`-P`" and "`-c`".

1. `-P <port>`

This normally specifies the host serial port which is used to communicate with the controller. However the option has a handy feature which allows you to specify a network address rather than a physical serial port:

```
-P net:192.168.1.50:3000
```

The above -P option tells avrdude to establish a TCP connection to IPv4 address 192.168.1.50, port 3000. Port 3000 is where wAVR listens for avrdude connections.

2. `-c <programmer-id>`

This is the standard "programmer type" option for avrdude. To work with wAVR, simply set `<programmer-id>` to "`dragon_isp`" or "`dragon_pdi`". Use the latter for XMega devices and the former for Tiny and Mega devices.

An example command line to program an ATMega168p would be:

```
$ avrdude -p atmega168p -P net:192.168.1.50:3000 -c dragon_isp -U flash:w:main.hex
```

For an XMega device:

```
$ avrdude -p atxmega128a3u -P net:192.168.1.50:3000 -c dragon_pdi -U flash:w:main.hex
```

That's all there is to it.

If you're using the latest avrdude with our uPDI patches then you will need to substitute "wavr_`isp`" in place of "`dragon_isp`" and "`wavr_pdi`" in place of "`dragon_pdi`". In addition, uPDI mode is accessed by specifying "wavr_`updi`". You will also need to switch wAVR into native mode using the following command at wAVR's shell prompt:

```
NetShell> prog mode wavr
```

See the command reference section, below, for more information.

Connecting to the Target UART

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

By default wAVR 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 wAVR'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 wAVR'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`.

6

Configuring wAVR

The Built-in Shell

wAVR 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 '^]'.

NetShell>
```

The available commands are:

`display`	Manipulate the onboard display
`help`	Display this help
`isp`	Configure ISP programmer
`net`	Display network status
`pdi`	Configure PDI programming
`prog`	Some generic programmer settings
`reboot`	Reboot wAVR
`serial`	Show wAVR's serial number
`uart`	Display/modify target UART configuration
`updi`	Configure uPDI programming
`usb`	Configure USB mode
`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.

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: `help`

`help`

Show list of available commands.

Command: `isp`

`isp`

Displays current ISP programming clock rate

`isp clock <rate>`

Sets the ISP programming clock to <rate>. This is the bit clock used when programming ISP devices and a non-zero value overrides any bit clock supplied by the programmer (avrdude). The default is 0, which means use a safe default value or whatever value is specified by the programmer. Non-zero values must lie between 1000 and 1800000 inclusive. Take note of the maximum safe value specified in the target device's data sheet in case you exceed the limit it is capable of supporting. Typically this is CPU clock / 4 for most ISP-capable targets.

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: pdi

`pdi`

Display's current PDI clock frequency.

`pdi clock <frequency>`

Sets the PDI clock rate to <frequency>. This is the communication rate used when programming XMega devices if it is not specified by the programmer. The default is `1000000` (1 MHz). The minimum supported clock is `100000` (100 KHz) and the maximum is `10000000` (10 MHz). You may need to experiment to determine the maximum rate at which your target device can be reliably programmed. For reference, it's worth noting that Atmel-ICE's max PDI clock rate is 7.5 MHz...

Command: prog

`prog`

Display's current status.

`prog rclk <rescue-clock-frequency>`

Set the rescue clock (pin 8 of wAVR's 10-pin target header) to the specified frequency. Supported rates are `1-2000000` (1 Hz to 2 MHz), where `0` (zero) disables the rescue clock. Note that the actual rate will be as close as possible to the requested frequency but due to timer constraints it will usually not be exact. The clock is available when programming ISP targets only.

`prog mode [<wavr> <dragon> | <stk500>]`

Set protocol supported over the network.

"wavr" is supported in the development version of avrdude plus our patches, and permit ISP, PDI and uPDI targets to be programmed by your wAVR device.

"dragon" is suitable for programming ISP and PDI targets via avrdude. wAVR will identify itself as an AVR Dragon in this case.

"stk500" is provided for Atmel Studio compatibility, although its use is discouraged as sadly Atmel Studio does not support programming in either STK500 mode or Dragon mode. In fact it doesn't know that either of these devices can be reached over the network, so wAVR is best used as an external programmer. If you really want to use wAVR on Atmel Studio as a native programmer then you need to set the device ID to "STK600" using the “prog stkid" command and connect via USB with wAVR's USB mode set to "stk600". This is not a supported configuration and we may remove USB emulation of STK600 in a future firmware update.

`prog stkid <stk500-id-string>`

Set the ID string reported to the host when the stk500 mode is selected. Default is "STK500_2. The appropriate ID will be chosen automatically for the other two modes.

Command: `reboot`

`reboot`

Immediately reboots wAVR.

`reboot defaults yesplease`

Reboots wAVR 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 wAVR.

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: `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: `updi`

`updi`

Displays current uPDI baud rate.

`updi baud <rate>`

Sets the uPDI baud rate to <rate>. This is the communication rate used when programming uPDI devices if it is not specified by the programmer. The default is zero, which means use a safe minimum of 225Kbaud. For other baud rates, wAVR will set the target's UPDICLKSEL field automatically to a value appropriate for the specified rate. Note that this

might fail to work if Brown Out Detect is set to anything other that its highest level. See the target's data sheet for details. The lowest acceptable value for the uPDI baud rate is 1200. The highest is 900000.

Command: `usb`

`usb`

Display current USB mode.

`usb mode [ CDC | STK600 ]`

Change USB mode to either CDC (USB Serial Port) or STK600 emulation (for Atmel Studio's benefit). Note that a mode change will take effect after a reboot.

Command: `wifi`

`wifi`

Display current WiFi settings.

`wifi ssid <wifi-network-name>`

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

Note: at the moment wAVR does not support network names containing spaces. This will be addressed in a future firmware update. You can work around this by setting wAVR into WiFi provision mode and entering the details through the captive portal or web page.

`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, wAVR does not support white-space characters in the passphrase and this, too, will be addressed in a future firmware update.

`wifi name <dhcp-client-name>`

Specifies a human-friendly name for wAVR. 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 wAVR driver in your device is incompatible with the new firmware. Recovering from this situation could entail desoldering and replacing the WiFi module!

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.