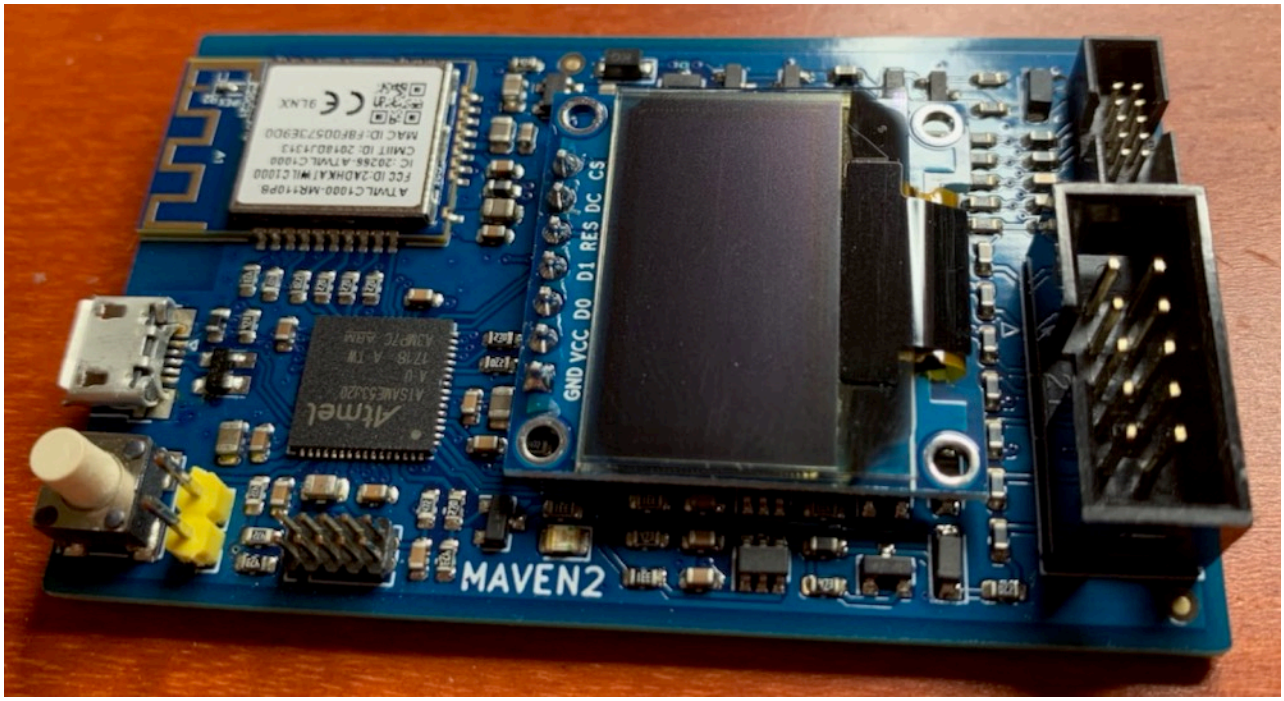
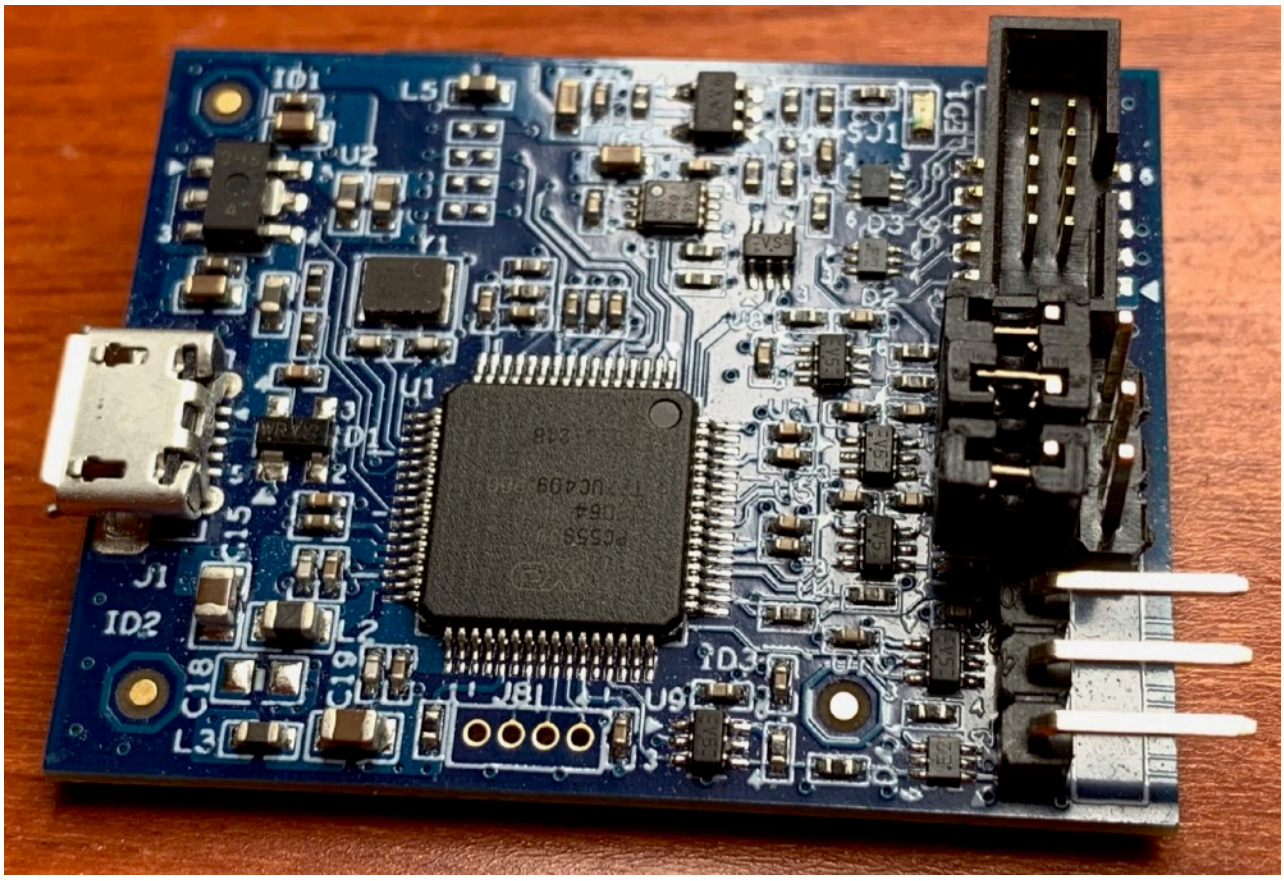
Maven Firmware

Maverick Embedded Technology Ltd





Introduction....4
Features....4
Supported ARM Micro-Controllers....5
GDB Server Support....7
Connecting to Maven over USB....8
Using Maven with GDB....9
Connecting to the Target UART....11
Configuring Maven....12
The Built-in Command Line....12
Command: adapter....13
Command: gdb_attach....13
Command: help....14
Command: reboot....14
Command: rstcfg....14
Command: rtos....15
Command: serial....16
Command: tc....16
Command: uart....17
Command: cortexm....17
Command: gpnvm....18
Command: info....18
Command: protect....19
Command: unlock....19
RTOS Support....20
Acknowledgements....23
GPIO Hardware Abstraction....23

ARM Cortex-M4 CMSIS Headers.....24
FreeRTOS 24

Introduction

Thank you for downloading Maven by Maverick Embedded Technology Ltd. The Maven firmware implements a Programmer/Debugger 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 USB means Maven works with any host system which can run GDB for ARM.
- GDB server automatically supports thread-aware debugging for firmware images containing either FreeRTOS or NuttX. Support for additional RTOSes will be added in due course.
- Does not require middleware running on the host, such as OpenOCD, and the associated configuration file(s).
- Supports Cortex-M SWO serial data in NRZ mode at up to 3MHz.
- Communicates with your target using RS232 on a UART or bit-banged I/O pin. Maven will make the UART data available over USB on your host. Both RxD and TxD are supported at all the common baud rates.
- The USB interface provides two CDC-compatible RS232 interfaces. One of those is dedicated to the GDB server, while the other is dedicated to SWO, target UART, and/or Semi-hosted console I/O. You can select SWO, target UART, or Semi-hosting either upon connection to the CDC device, or by a configuration command issued via GDB.
- Versions of Maven exist for several debug probes:
    - ✦ NXP's MCU-Link
    - ✦ Maverick Embedded Technology Ltd ***Maven2*** probe.

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.
- Raspberry Pi Pico (RP2040).

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.

Cortex-M23

- Microchip SAML11 (still experimental)

Cortex-M33

- NXP LPC55S69

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 '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 64 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.

Maven can also auto-detect if you are debugging a firmware image containing a real-time operating system such as FreeRTOS or NuttX. Maven will use GDB's thread awareness feature to make it possible to view detailed task information such as name, state, priority, and full back-trace for all extant tasks.

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

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

7

Connecting to Maven over USB

Maven has a simple command-line interface from which you can monitor and configure various aspects of Maven's behaviour. The command-line interface is available from within GDB using the "monitor" command. Anything prefixed with "monitor" is passed to Maven and executed. 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, you first need to determine how the USB CDC ports are instantiated on your host computer. On Linux and MacOS host, this will be two entries under /dev on the root filesystem. In Windows, this will be as to COM ports. For example, on MacOS the following two device files are automatically created:

```
/dev/cu.usbmodemMCL1ZZ124C3E1
/dev/cu.usbmodemMCL1ZZ124C3E3
```

Maven's GDB server is accessed via the first file, while SWO/target UART are accessed via the second. The /dev/cu.usbmodem prefix will be constant. The suffix will vary depending on the serial number of your Maven device.

Add the following line to your .gdbinit file to configure GDB to connect to Maven's GDB server:

```
target extended-remote /dev/cu.usbmodemMCL1ZZ124C3E1
```

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= /dev/cu.usbmodemMCL1ZZ124C3E1
bin="${1}"
if [ "$2" != "" ]; then
        target="$2"
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 USB CDC device associated with 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 using your favourite terminal server program, such as TerraTerm.

Simply connect the Maven 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 115200 baud, 8 data bits, no parity and one stop bit. The speed, parity and stop bit settings can be changed via Maven's command line or by your terminal server program.

8

Configuring Maven

The Built-in Command Line

Maven has a simple command-line interface available via the GDB monitor command.

The available commands are:

`adapter`	View/Modify debug port parameters
`gdb_attach`	Configures GDB attach behaviour
`help`	Display this help
`reboot`	Reboot Maven
`rescan`	Initiates/configures target rescan
`rstcfg`	Configure sRST behaviour
`rtos`	Configure and show RTOS parameters
`serial`	Show serial number of your Maven device
`tc`	Configure target communications
`uart`	Display UART configuration

The default settings for most of the above commands will be fine. The most common changes you are likely to make will be target UART baud rate using the 'tc' command, and ADIv5 clock speed using the 'adapter' command.

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 SWCLK frequency for communicating with the target. The available frequencies range between about 300 KHz and 13 MHz, depending on the hardware Maven is running on. The default is around 2 MHz.

`adaptor idlecycles <value>`

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

`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: 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.

`gdb_attach auto`

When GDB connects to Maven, a connection to the target will take place automatically, without the need for “attach 1” at the GDB command prompt. This is the default setting for single-core targets. For dual-core targets, you must always issue a GDB attach command, specifying the desired core.

`gdb_attach manual`

When GDB connects to Maven, you will need to issue a GDB “attach N” command at the GDB prompt, where “N” is the CPU core number plus 1.

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

Command: `help`

`help`

Show list of available commands.

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.

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.

Command: `rtos`

`rtos`

Show current RTOS configuration. If invoked from within GDB (using GDB's "mon" command), this will also display details of any RTOS detected.

`rtos support <auto|off|name>`

Configures RTOS detection:

- `auto` Target will be probed for all supported RTOSes (default).
- `off` RTOS support is disabled.
- `name` Only the named RTOS will be probed.

`rtos params [-s] [-r <name>] [new-params]`

Shows parameters for the RTOS specified by "-r <name>", or the current RTOS is invoked from within GDB. If the target's RTOS requires slightly different parameters, to handle structure padding, for example, then these can be specified with [new-params]. This is a comma-separated list of integers pairs in the form I:V, where 'I' specifies the parameter index, and 'V' is the parameter value. Valid indices per RTOS are shown by 'rtos params -r <name>'.

If '-s' is specified, the new params are saved to non-volatile storage so need only be specified once. Otherwise, they will affect the current GDB session only.

The default RTOS parameters can be restored by specifying 'default' in place of <new-params>.

`rtos init-syms <on|off>`

If an RTOS is detected, this flag indicates whether or not the memory locations referenced by volatile RTOS symbols should be initialised to zero when 'gdb_attach' mode is configured for 'reset'. The default is 'on', and will ensure stale/invalid thread state is not used when a GDB session first starts.

Command: `serial`

`serial`

This will display the device's unique serial number.

Command: `tc`

`tc`

This will display the target comms settings for each service (GDB, SWO, UART).

`tc <service> usb <usb-port>`

Assign the numbered USB CDC port instance <usb-port> to the specified service. Valid values are 0 - 1 inclusive, or '-' to deassign USB. It is possible to share certain services on a single USB instance. Upon connection, you will be asked which service you wish to connect to. Note that the GDB server service cannot be shared in this way. Changes to this setting require a reboot to take effect.

`tc <service> mode <mode-string>`

Configure the specified service's data mode. Supported modes are:

"crlf" Convert an LF character into a CRLF pair.

"raw" Disable CRLF translation. Data is passed as-is to the host.

Mode strings can be prefixed with an optional "+" or "-" character to imply addition or removal of the mode from the service. Note that services sharing a USB port will use the lowest common mode. So if one of the services is in "raw" mode, all shared services on the same port will be in "raw" mode. Changes to these settings will not affect an established connection.

`tc <service> fmt <uart-character-format>`

For services provided by a UART, this command configures the UART character format (number of data bits, parity, and stop bits). For example, "8N1" specifies 8 bits, no parity, 1 stop bit. "5E2" is 5 bits, even parity and 2 stop bits. Note that the character format for the "swo" service is fixed at "8N1" as per the standard.

`tc <service> baud <uart-baudrate>`

Configure UART baud rate. Valid rates are > 0 and <= a probe-specific maximum value (usually around 2500000). If Maven cannot get to within +/- 1.5% of the required rate then an error will be displayed. Rates above 2 Mbaud are at higher risk of receiver overrun errors. See the "uart" command for the actual baud rate used, together with other useful statistics.

Command: `uart`

`uart`

Display configuration and statistics for Maven's UART devices.

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.

RTOS Support

Maven currently supports task-awareness for the following RTOSes:

- FreeRTOS
- NuttX

For FreeRTOS, the default parameters (see command: rtos params) will be adequate for almost all FreeRTOS ports to Cortex-M devices. It's unlikely you will need to adjust them, so thread-awareness for your FreeRTOS images should work right out the box.

NuttX support is a little more complicated due to its plethora of build-time CONFIG_* options. These make it very difficult to locate the key members of the "struct tcb_s" data structure assigned to each task. To overcome this problem, you must apply the following patch to the NuttX source tree:

```
diff --git a/include/debug.h b/include/debug.h
index f75a1fab84..fa69b2049d 100644
--- a/include/debug.h
+++ b/include/debug.h
@@ -1067,6 +1067,52 @@ void _info(const char *format, ...);
 #endif
 #endif /* CONFIG_CPP_HAVE_VARARGS */

+#ifndef CONFIG_DEBUG_DISABLE_DEBUGGER_HINTS
+/* This structure is used by thread-aware debuggers in order to locate key
+ * task state within a NuttX firmware image. Additions to this structure
+ * will need corresponding changes to the initializer in nx_start.c.
+ *
+ * ================================ NOTE ================================
+ * New hints must be APPENDED to this structure only. Existing hints
+ * must not be removed.
+ *
+ * The debugger will assume elements are naturally aligned, regardless
+ * of the native alignment constraints of the target device. Be careful.
+ * ================================ NOTE ================================
+ */
+struct nuttx_debugger_hints_s {
+  uint32_t magic;
+
+  /* sizeof(FAR void *) */
+  uint16_t sizeof_far_pointer;
+
+  /* offsetof(struct tcb_s, pid) */
+  uint16_t offsetof_tcb_pid;
+
+  /* offsetof(struct tcb_s, sched_priority) */
+  uint16_t offsetof_tcb_sched_priority;
+
+  /* offsetof(struct tcb_s, xcp) */
+  uint16_t offsetof_tcb_xcp;
+
+  /* offsetof(struct tcb_s, name) */
```

```
+  uint16_t offsetof_tcb_name;  /* '0' if CONFIG_TASK_NAME_SIZE == 0 */
+
+  /* offsetof(struct xcptcontext, regs) */
+  uint16_t offsetof_xcptcontext_regs;
+
+  /* sizeof(struct xcptcontext) */
+  uint16_t sizeof_xcptcontext_regs;
+
+  /* CONFIG_MAX_TASKS */
+  uint16_t config_max_tasks;
+
+  /* New hints go here. */
+};
+#define        NUTTX_DEBUGGER_HINTS_MAGIC      \
+  (0xDEBF0000u + sizeof(struct nuttx_debugger_hints_s))
+#endif /* CONFIG_DEBUG_DISABLE_DEBUGGER_HINTS */
+
 #if defined(__cplusplus)
 }
 #endif
diff --git a/sched/init/nx_start.c b/sched/init/nx_start.c
index 771522069d..483ba34d3a 100644
--- a/sched/init/nx_start.c
+++ b/sched/init/nx_start.c
@@ -330,6 +330,24 @@ static FAR char *g_idleargv[CONFIG_SMP_NCPUS][2];
 static FAR char *g_idleargv[1][2];
 #endif

+#ifndef CONFIG_DEBUG_DISABLE_DEBUGGER_HINTS
+const struct nuttx_debugger_hints_s g_nuttx_debugger_hints = {
+  .magic                      = NUTTX_DEBUGGER_HINTS_MAGIC,
+  .sizeof_far_pointer         = sizeof(FAR void *),
+  .offsetof_tcb_pid           = offsetof(struct tcb_s, pid),
+  .offsetof_tcb_sched_priority = offsetof(struct tcb_s, sched_priority),
+  .offsetof_tcb_xcp           = offsetof(struct tcb_s, xcp),
+#if (CONFIG_TASK_NAME_SIZE > 0)
+  .offsetof_tcb_name          = offsetof(struct tcb_s, name),
+#else
+  .offsetof_tcb_name          = 0,
+#endif
+  .offsetof_xcptcontext_regs   = offsetof(struct xcptcontext, regs),
+  .sizeof_xcptcontext_regs     = sizeof(((struct xcptcontext *)0)->regs),
+  .config_max_tasks           = CONFIG_MAX_TASKS,
+};
+#endif /* CONFIG_DEBUG_DISABLE_DEBUGGER_HINTS */
+
 /****************************************************************************
  * Public Functions
  ****************************************************************************/
@@ -355,6 +373,13 @@ void nx_start(void)
   int cpu = 0;
   int i;

+#ifndef CONFIG_DEBUG_DISABLE_DEBUGGER_HINTS
+  /* Reference nuttx_debugger_hints_s to ensure it is not removed from the
+   * image.
+   */
+  (void) g_nuttx_debugger_hints;
+#endif
```

```
+
   sinfo("Entry\n");

   /* Boot up is complete */
```

The patch adds a short 20-byte data structure to the firmware image. The structure contains the offsets to all the relevant fields of “struct tcb_s” necessary for Maven to support NuttX tasks under GDB.

7

Acknowledgements

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

GPIO Hardware Abstraction

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

ARM Cortex-M4 CMSIS Headers

FreeRTOS

wAVR Firmware uses a wholly unmodified version of FreeRTOS-10 which is licensed under the terms of the MIT Open Source License, below: