Quick Start Guide

RZ/N1D-DB Board U-Boot and Linux

System-on-Chip

Target Device RZ/N1D

Renesas Electronics
www.renesas.com

Renesas Electronics Corporation

R01QS0007EG0104
October 16, 2018

1 INTRODUCTION

This guide aims to quickly get U-Boot and Linux running on your Renesas RZ/N1D-DB board. This guide is written for software engineers who may be new to Linux. Experienced Linux engineers will likely just need the information on how to program U-Boot onto a bare board.

There are several pieces of software that are used to load and run Linux on the RZ/N1, these are discussed below. Pre-built binaries are provided to get you up and running quicker.

1.1 Renesas BootROM

The BootROM is internal to the device and is always run on reset. The BootROM loads the first valid SPKG image from one of three sources, QSPI, NAND or USB DFU, the choice of which is done via boot mode pins, and starts executing it. DFU is a protocol used with USB to update software on embedded products; the RZ/N1 device will act as USB Device and is attached to a USB Host, e.g. a PC.

An SPKG is a Renesas proprietary format that includes information on the size of the binary payload, the destination for the payload, and optional signature information. For the purposes of this document, we assume that the device allows SPKGs without a signature.

Note: The BootROM will only allow an SPKG payload to be written to internal SRAM as writing to QSPI and setting up DDR is board specific.

Normally, on the RZ/N1D-DB board, when the board is reset the BootROM will load an SPKG from QSPI. Typically, this SPKG contains U-Boot or U-Boot/SPL.

However, by changing the boot mode pins state by holding down a switch when resetting the board, the BootROM will start in USB DFU mode and will wait for the host PC to upload an SPKG.

1.2 U-Boot

U-Boot is a boot loader that allows you to run commands that can read and write to/from QSPI or SD cards, load files from a TFTP server, program QSPI using USB DFU, start Linux or other OS and start the Cortex-M3 processor. It also initialises the DDR controller.

Typically, U-Boot will load the Device Tree Blob (dtb) and Linux kernel image from QSPI, and pass arguments to the kernel.

1.3 Linux Kernel

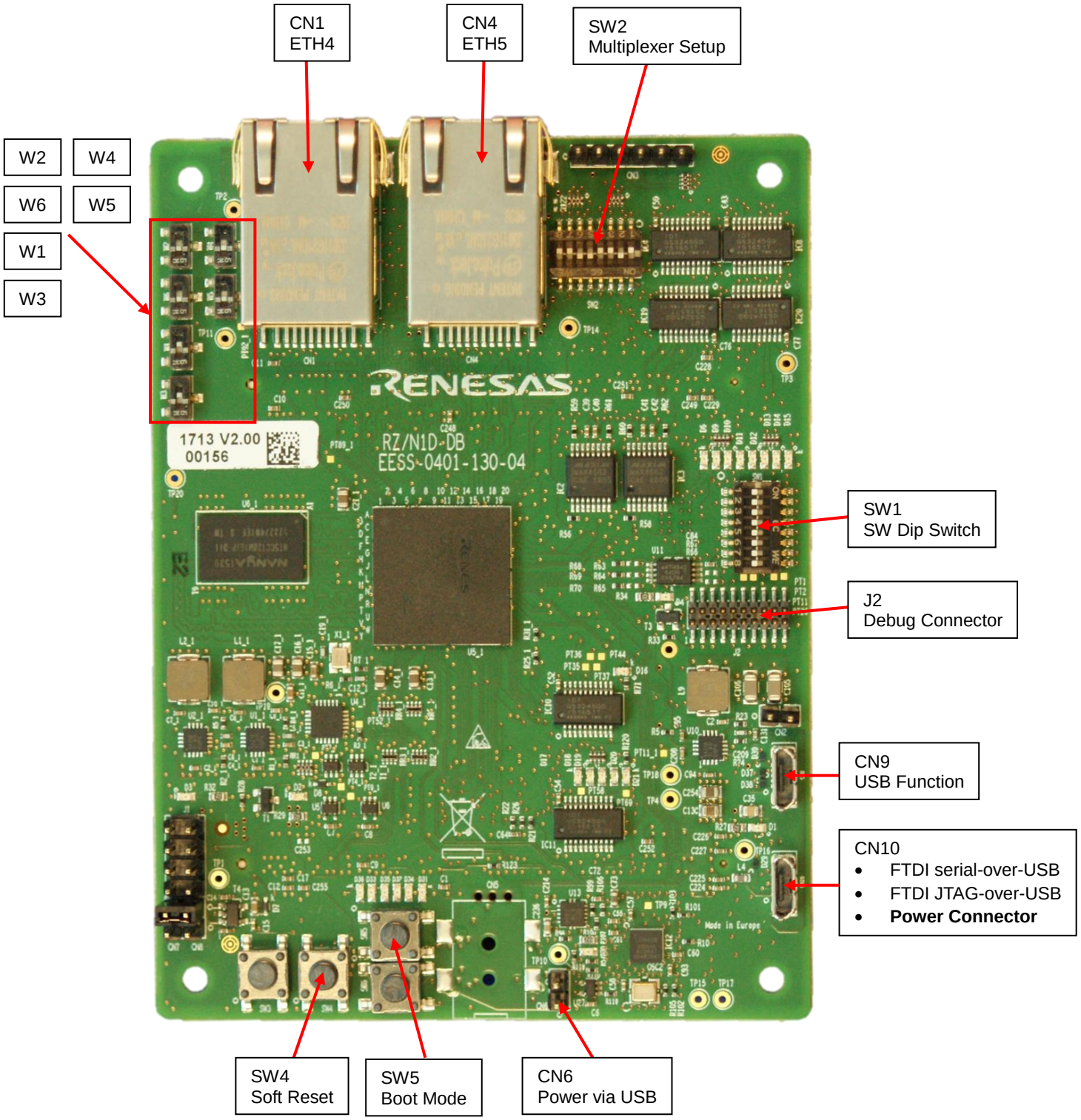
The Linux kernel will load drivers for all of the peripherals and provide you with a console to operate it.

Linux needs a root file system (rootfs) that contains user-space applications and other files. The rootfs provided as a SquashFS file system and is written to QSPI.

2 SETUP

2.1 RZ/N1D-DB Board

The main connectors and switches of the board are shown below.



The following switches must be set.

Switch bank SW2

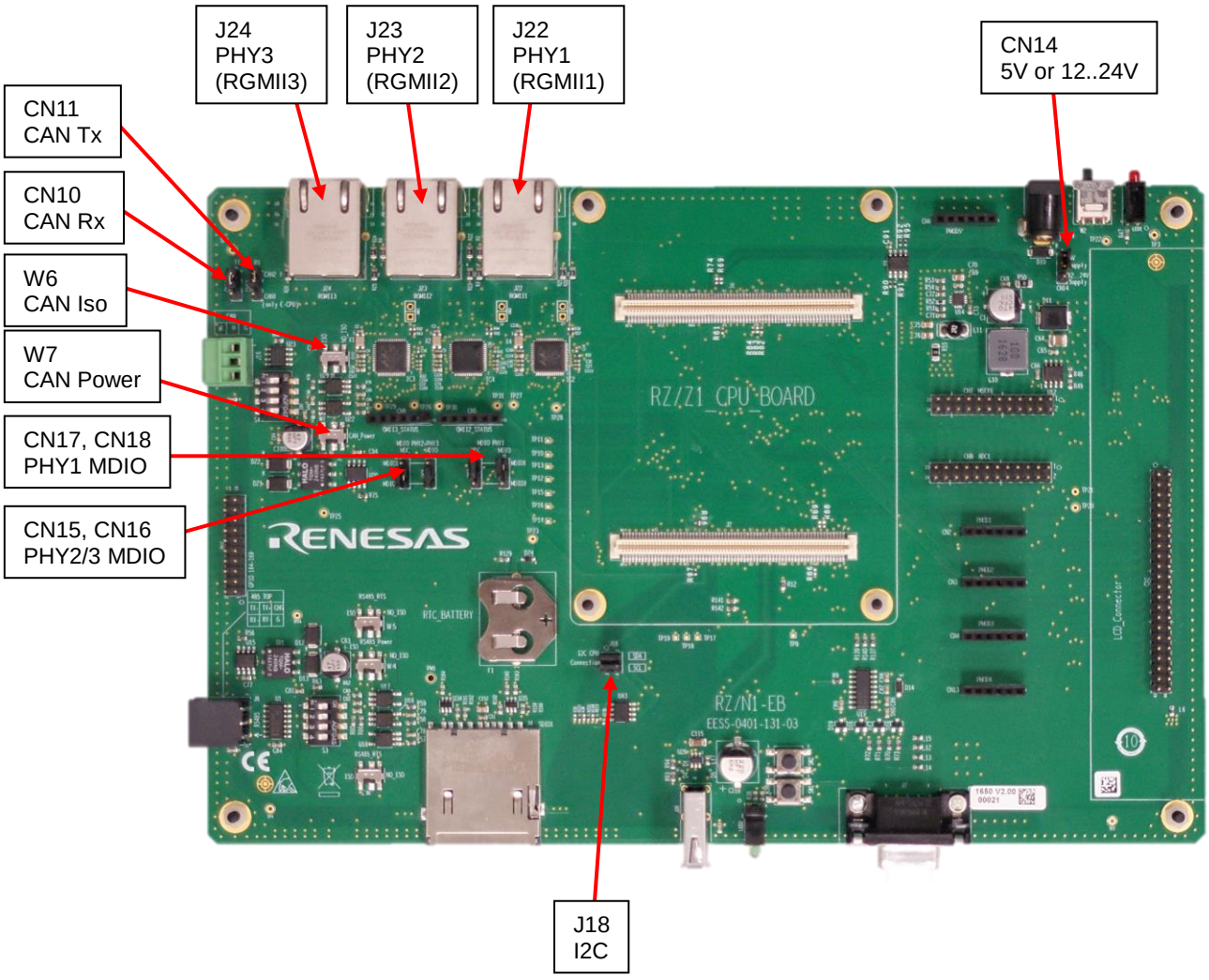
Number	SW ON (low)	SW OFF (high)	Default Setting
1	RMII/MII	LCD	OFF
2	CAT/S3	PMOD	**ON**
3	MSEBI	Mixed	OFF
4	RMII2	SPI5	OFF
5	USB 1x Host 1x Device	USB 2 x Host	**ON**
6	ARM Debug	FTDI Debug	**ON**
7	Segger Debugger	No connection	OFF
8	I-Jet Debugger	No connection	**ON**

Other switches

Switch	SW ON (white bar)	SW OFF	Default Setting
W1	JTAG Mode	ARM Coresight Mode	OFF
W2	RXCLK4 from PHY	RXCLK4 from GPIO61	**ON**
W3	Boot Mode NAND	Boot Mode QSPI	OFF
W4	RXCLK5 from PHY	RXCLK5 from GPIO61	**ON**
W5	LCD pull up	LCD pull down	**ON**
W6	LCD pull up	LCD pull down	**ON**

Extension Board

The relevant switches and connectors of the Extension Board are shown below.



If using the Extension Board, please ensure the following jumper and switch settings are made:

Jumper	Description	Default Setting
CN10	CAN Rx source	Connect pins 1 and 2 (CAN2 Rx)
CN11	CAN Tx source	Connect pins 1 and 2 (CAN2 Tx)
CN15	PHY2/PHY3 MDC source	Connect pins 2 and 3 (MDC2)
CN16	PHY2/PHY3 MDIO source	Connect pins 2 and 3 (MDIO2)
CN17	PHY1 MDC source	Connect pins 1 and 2 (MDC1)
CN18	PHY1 MDIO source	Connect pins 1 and 2 (MDIO1)
J18	I2C SDA	Connect pins 1 and 2 (enable I2C SDA on Ext Board)
J18	I2C SCL	Connect pins 3 and 4 (enable I2C SCL on Ext Board)
W6	CAN Signals	NO_ISO (do not use isolated CAN signals)
W7	CAN Power	NO_ISO (do not use isolated CAN power)

Board Connections

Connect the following to your PC:

- Connect CN9 on the board to a USB Host connector on your PC. This provides USB DFU.
- Connect CN10 on the board to a USB Host connector on your PC. This provides Serial-over-USB and JTAG-over-USB services. After the FTDI driver has been installed on your PC, four additional virtual serial ports will exist. The board uses the 3rd port for UART output at 115200,8,n,1. On Linux PCs, if you have no other serial-over-USB devices attached, this is accessed using /dev/ttyUSB2.
- If the board is powered by USB, press switch SW4 to perform a soft reset.
- If you wish to use Ethernet, but do not have an Extension Board, you have to use a special Device Tree file that allows Linux to use GMAC2 via the 5-Port Switch on the RZ/N1 device. Where this document tells you to use uImage-rzn1d400-db.dtb, instead use uImage-rzn1d400-db-no-cm3.dtb. Note that the Linux driver for the 5-Port Switch simply configures it as an unmanaged switch. When using this DTB, connect CN1 on the board to a dedicated Network Interface Card (NIC) on your PC. This setup does not correspond to the default use case of RZ/N1 in industrial switch applications.
- If you wish to use Ethernet, and are using the Extension Board, you can connect J22 of the Extension Board to a dedicated Network Interface Card (NIC) on your PC. This is used to access GMAC1 on the RZ/N1 device.
- By default, the board uses static IP addresses, so please ensure your host's NIC is set up with a static IP address of 192.168.1.30. This address is set by default in the U-Boot serverip environment variable.

2.2 Write U-Boot to QSPI

This section provides instructions to program QSPI flash on a new board. You will need a Windows or Linux host PC for this. The steps performed are:

- Use the BootROM DFU mode to load U-Boot (in SPKG format) into SRAM.
- Use the U-Boot dfu command to write U-Boot (in SPKG format) into QSPI.

1. On your Linux PC, install the 'dfu-util' package, e.g.:

```
sudo apt-get install dfu-util
```

   If using a Windows PC, follow the instructions in the U-Boot User Manual for installing dfu-util. For all of the subsequent Linux commands below that start 'sudo dfu-util', please replace with Windows commands starting with 'dfu-util-static.exe'.

2. On the board, hold down switch SW5 (to select DFU boot mode instead of QSPI) and press switch SW4 (soft reset). The RZ/N1 serial port should output:

```
** BOOTLOADER STAGE0 for RZN1 **
Boot source: USB
```

3. Download U-Boot to SRAM. On your host PC run:

```
sudo dfu-util -D u-boot-rzn1d400-db.bin.spkg
```

4. U-Boot should run and the RZ/N1 serial port presents you with a console, similar to this:

```
U-Boot 2017.01

Model: RZ/N1D Demo Board
DRAM:  256 MiB
MMC:   sdhci@0x40100000: 0
SF: Detected mx25l25635f with page size 256 Bytes, erase size 64 KiB,
total 32 MiB, mapped at 10000000
In:    serial@0x40060000
Out:   serial@0x40060000
Err:   serial@0x40060000
Net:   dwmac.44000000, dwmac.44002000
Hit any key to stop autoboot:  0
=>
```

   ***Note*:** If your board has previously been used and already has U-Boot environment variables programmed into QSPI, U-Boot may attempt to start running the commands specified by the bootcmd env variable. Interrupt this by pressing any key.

5. If your board has been programmed with an older version of U-Boot, the dfu_ext_info environment variable may be incompatible. If so, please run:

```
env default -f dfu_ext_info
saveenv
```

6. Ensure the U-Boot/SPL region of QSPI Flash is erased, run:

```
sf probe
sf erase 0 10000
```

7. On the U-Boot console, run:

```
dfu
```

8. Write U-Boot to QSPI. On your host PC run:

```
sudo dfu-util -a "sf_uboot" -D u-boot-rzn1d400-db.bin.spkg
```

   Wait until it completes, the U-Boot console will prompt you to press Ctrl-C when done.

***Note*:** The "sf_uboot" DFU target corresponds to the second region of the QSPI Flash. If there is a valid SPKG written into the first region ("sf_spl"), the BootROM will load this instead of U-Boot. Otherwise the BootROM will output messages whilst it looks for the first valid SPKG, similar to:

```
STATUS: Valid SPKG header not found (100 QSPI Flash 256-byte blocks read)
```

9. Press switch SW4 to reset the board, the BootROM will load and run U-Boot showing the following output on the terminal:

```
** BOOTLOADER STAGE0 for RZN1 **
Boot source: QSPI
00 BOOTLOADER STAGE0 Success
*** Bootloader stage0 END ***
*** Execute 2nd Stage Bootloader which has been loaded and verified ***

U-Boot 2017.01

Model: RZ/N1D Demo Board
DRAM:  256 MiB
MMC:   sdhci@0x40100000: 0
SF: Detected mx25l25635f with page size 256 Bytes, erase size 64 KiB,
total 32 MiB, mapped at 10000000
In:    serial@0x40060000
Out:   serial@0x40060000
Err:   serial@0x40060000
Net:   dwmac.44000000, dwmac.44002000
Hit any key to stop autoboot:  0
=>
```

2.3 Write Cortex M3 image to QSPI

The following instructions use the U-Boot dfu command to write the Cortex M3 code image into QSPI. Please see the GOAL Management Software Quick Start Guide for details of the Cortex M3 code image.

1. On the board, press switch SW4 (soft reset). U-Boot should run and the RZ/N1 serial interface presents you with a console.

2. From U-Boot, run:

```
dfu
```

3. Write the dtb to QSPI. On your host PC run:

```
sudo dfu-util -a "sf_cm3" -D rzn1d_demo_board_eb.bin
```

4. Wait until it completes, the U-Boot console will prompt you to press Ctrl-C when done.

2.4 Write Linux to QSPI

The following instructions use the U-Boot dfu command to write the kernel dtb, kernel image, and JFFS2 rootfs into QSPI.

1. On the board, press switch SW4 (soft reset). U-Boot should run and the RZ/N1 serial presents you with a console.

2. From U-Boot, run dfu:

```
dfu
```

3. Write the dtb to QSPI. On your host PC run:

```
sudo dfu-util -a "sf_dtb" -D uImage-rzn1d400-db.dtb
```

4. Write the kernel to QSPI. On your host PC run:

```
sudo dfu-util -a "sf_kernel" -D uImage
```

5. Write the rootfs to QSPI. On your host PC run:

```
sudo dfu-util -a "sf_data" -D core-image-minimal-rzn1.squashfs
```

6. Wait until it completes, the U-Boot console will prompt you to press Ctrl-C when done.

2.5 Setup U-Boot environment variables

This section sets up the U-Boot environment variables so that it automatically reads the Linux kernel and DTB from QSPI, then starts Linux.

From U-Boot, set the MAC addresses corresponding to the MAC address sticker on the board, for example:

```
setenv -f ethaddr  74:90:50:02:00:FD
setenv -f eth1addr 74:90:50:02:00:FE
```

The following Linux bootargs setting specifies a read-only rootfs in the QSPI, a read-write JFFS2 file system in QSPI, and a static IP address for GMAC1 in Linux. The `clk_ignore_unused` option ensures Linux does not turn off any clocks to IP blocks that are used by the software running on the Cortex M3.

```
setenv bootargs "console=ttyS0,115200 root=/dev/mtdblock7 init=/init
rootwait ip=192.168.1.50:::::eth0 earlyprintk clk_ignore_unused"
```

The following bootcmd setting makes U-Boot load and start the Cortex M3 image from QSPI, then load a 128KB DTB and a 6MB kernel from QSPI, and start Linux automatically on start up. The offsets into the SPI flash correspond to the DFU `sf_cm3, sf_dtb` and `sf_kernel` targets.

```
setenv bootcmd "sf probe  &&  sf read 0x4000000 d0000 80000  &&
rzn1_start_cm3  &&  sleep 4  &&  sf read 0x8ffe0000 b0000 20000  &&
sf read 0x80008000 1d0000 600000  &&  bootm 0x80008000 - 0x8ffe0000"
```

Save the environment variables:

```
saveenv
```

2.6 Using Linux

Reset the board, you will see the boot messages followed by a log on prompt. This can take quite a while.

```
rzn1d400-db login:
```

Log on with username `root`.

Now you have Linux running, you can ping your PC's NIC.

```
ping 192.168.1.30
```

2.7 OpenOCD Debugger

This section provides basic information on how to connect a debugger to the ARM Cortex A7 CPUs on the board. The board uses an FTDI device to provide JTAG over USB which is supported via the OpenOCD software. Note that openocd v0.10.0 adds support for MMU address translation and cache flushing which is required to debug the Linux kernel or other OS that uses the MMU to remap memory ranges. However, if you connect to the target after it has executed code to enable the MMU and cache, download performance will be significantly slower due to the MMU table look ups. OpenOCD v0.10.0 has issues debugging ARM Thumb2 code, so please build the latest version.

```
git clone  http://repo.or.cz/openocd.git
cd openocd
./bootstrap
./configure --disable-jlink
make clean
make
sudo make install
```

OpenOCD provides a “gdbserver” so you can connect to Eclipse or other debuggers that supports this protocol. The following instructions detail how to connect to the gdbserver using the gdb command line debugger.

Ensure switch SW2-6 is OFF and switch W1 is OFF, i.e. away from the white bar.

Below we show basic commands to get you started with GDB. Further details on using OpenOCD with GDB can be found at http://openocd.org/doc/html/GDB-and-OpenOCD.html.

Connect to the board using the configuration file provided by Renesas. This starts a GDBServer which you can connect to from gdb or any other debugger that supports connecting to a GDBServer.

```
openocd -f renesas-rzn1d-openocd.cfg
```

In a separate terminal, you can now connect using gdb. Specify the U-Boot elf file on the command line. You can use the -tui option to show source code in a simplified GUI:

```
arm-linux-gnueabihf-gdb -tui u-boot
...
(gdb) target remote localhost:3333
```

Read symbols from the ELF file specified on the gdb command line, u-boot in this example, and download the code to the RZ/N1 SRAM:

```
(gdb) load
Loading section .text, size 0x1ffc0 lma 0x200a0000
...
Start address 0x200a0000, load size 217553
```

Note: The U-Boot elf file is actually different to the binary image, so download the binary

```
(gdb) mon load_image u-boot.bin 0x200a0000 bin
```

Step into the code:

```
(gdb) s
```

Set a breakpoint at the start of a function:

```
(gdb) b board_init
```

Run the code until you hit a breakpoint:

```
(gdb) c
```

RZ/N1D-DB Board U-Boot and Linux