Quick Start Guide

RZ/N1S-DB Board U-Boot

System-on-Chip

Target Device RZ/N1S

Renesas Electronics
www.renesas.com

Renesas Electronics Corporation

R01QS0008EG0104
June 8, 2018

1 INTRODUCTION

This guide aims to quickly get U-Boot running on your Renesas RZ/N1S-DB board.

There are several pieces of software that are used to load and run software 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 is board specific.

Normally, on the RZ/N1S-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

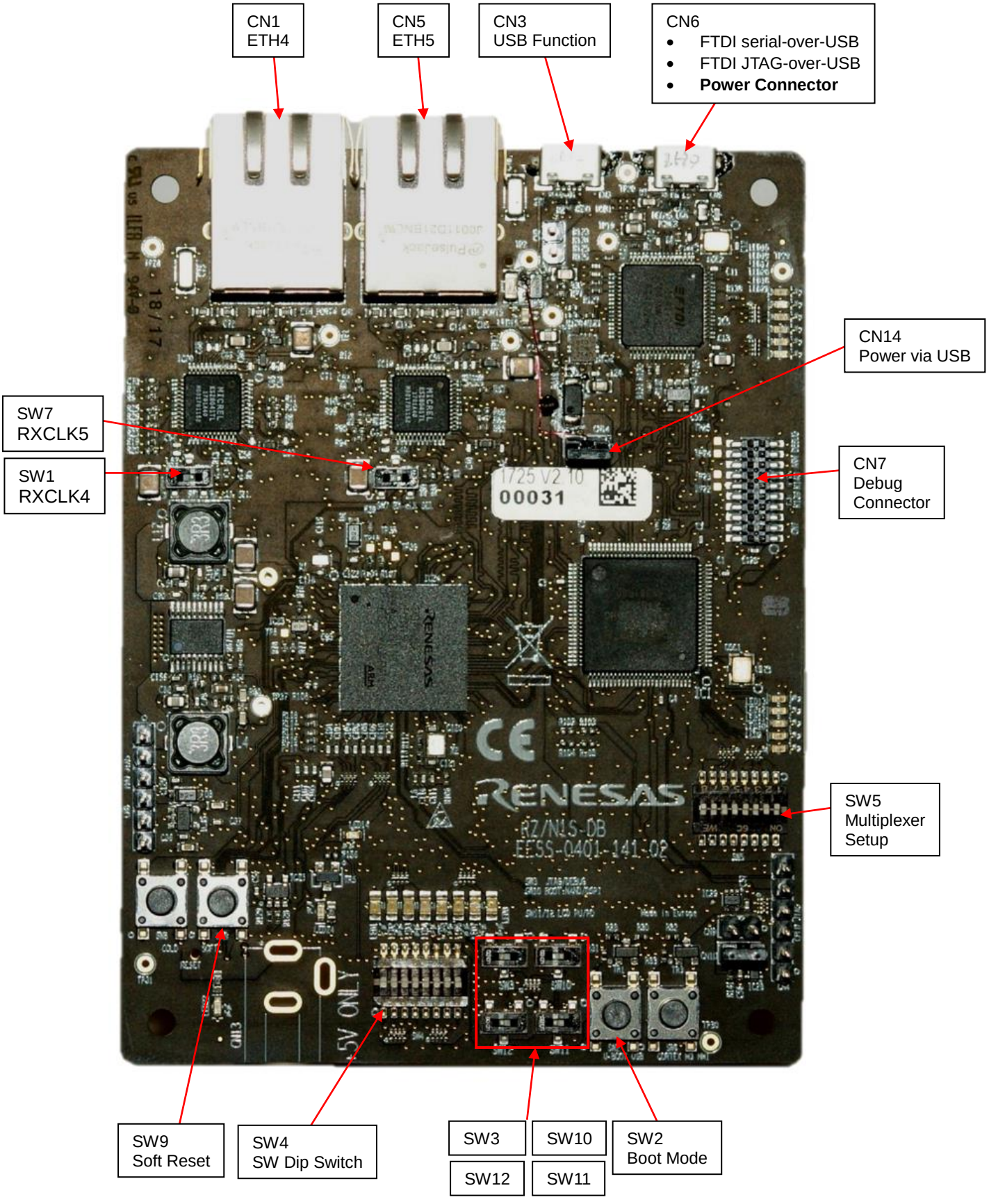
U-Boot is a boot loader that allows you to run commands that can read and write to/from QSPI or SD cards, load files from a TFTP server, program QSPI using USB DFU, start an OS and start the Cortex-M3 processor.

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

2 SETUP

2.1 RZ/N1S-DB Board

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



The following switches must be set.

Switch bank SW5

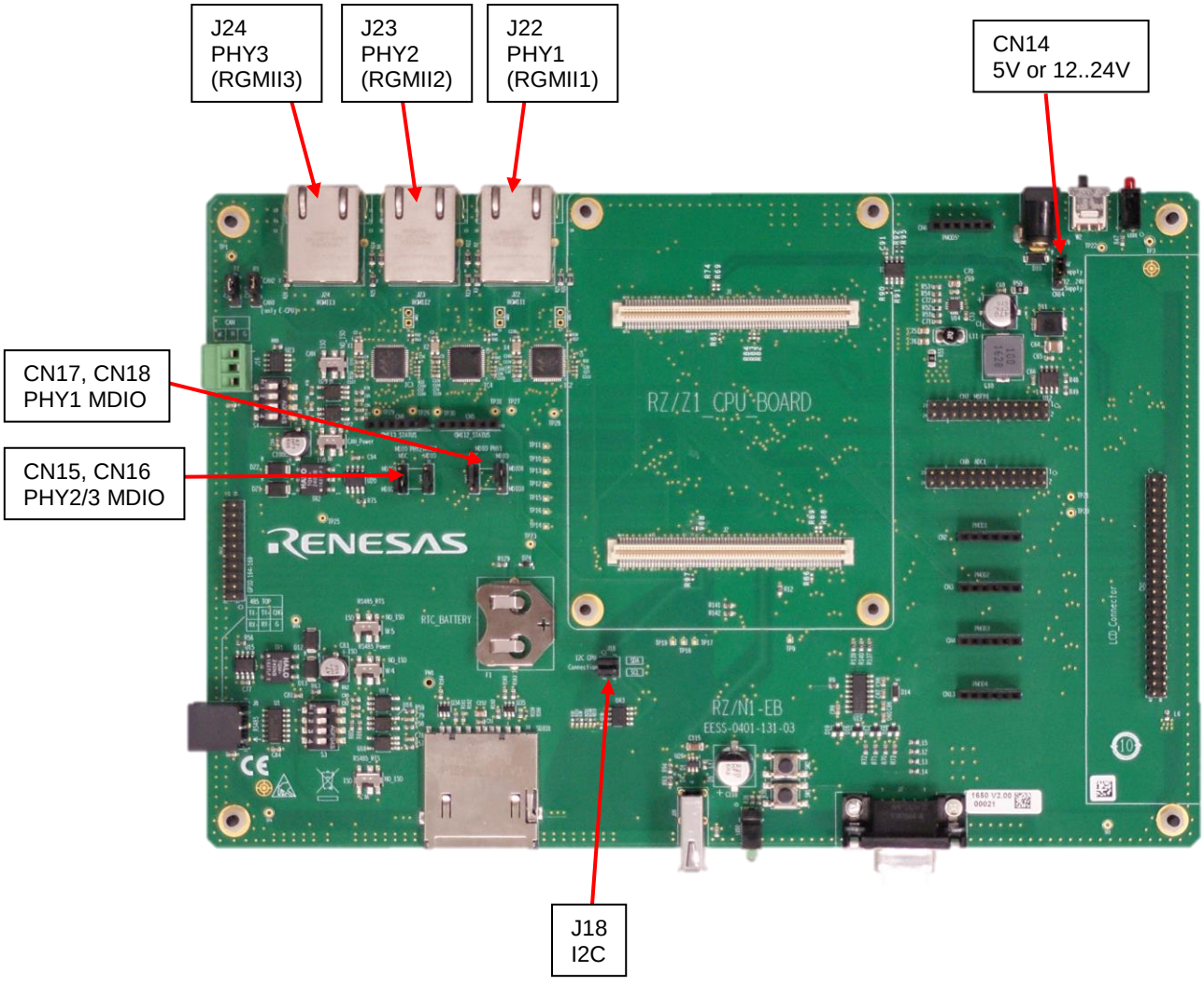
Number	SW ON (low)	SW OFF (high)	Default Setting
1	RMII/MII	LCD	OFF
2	CAT/S3	PMOD	**ON**
3	MSEBI	Mixed	OFF
4	RMII2 / MDIO3	QSPI2-Memory	OFF
5	USB 1x Host 1x Device	USB 2 x Host	**ON**
6	ARM Debug	FTDI Debug	**ON**
7	Segger Debugger	I-Jet Debugger	OFF
8	Not used	Not used	OFF

Other switches

Switch	SW ON (white bar)	SW OFF	Default Setting
SW3	JTAG Mode	ARM Coresight Mode	OFF
SW10	Boot from NAND	Boot from QSPI	OFF
SW11	LCD pull up	LCD pull down	**ON**
SW12	LCD pull up	LCD pull down	**ON**
SW1	RXCLK4 from PHY	RXCLK4 from GPIO61	**ON**
SW7	RXCLK5 from PHY	RXCLK5 from GPIO61	**ON**

Extension Board

The relevant switches 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
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)

Board Connections

Connect the following to your PC:

- Connect CN3 on the board to a USB Host connector on your PC. This provides USB DFU.
- Connect CN6 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 SW9 to perform a soft reset.
- If you wish to use Ethernet, but do not have an Extension Board, you can connect CN1 on the board to a dedicated Network Interface Card (NIC) on your PC. This is used by U-Boot to access GMAC2 via the 5-Port Switch on the RZ/N1 device. Note that the U-Boot driver for the 5-Port Switch simply configures it as an unmanaged switch.
- 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 Linux or Windows 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 SW2 (to select DFU boot mode instead of QSPI) and press switch SW9 (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-rzn1s324-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/N1S-DB board
DRAM:  4 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, at the U-Boot console 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-rzn1s324-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 SW9 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/N1S-DB board
DRAM:  4 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 Setup U-Boot environment variables

This sets up the U-Boot environment variables for your board. 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
```

Save the environment variables:

```
saveenv
```

2.4 OpenOCD Debugger

This section provides basic information on how to connect a debugger to the ARM Cortex A7 CPU 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 SW5-6 is OFF and switch SW3 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-rzn1s-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/N1S-DB Board U-Boot