RZ/N1D Group

CONNECT IT! ETHERNET RZ/N1D-DB

Introduction

RZ/N1 is a family of scalable ARM® based industrial Ethernet switches. Due to the integrated performant ARM® Cortex®-A7 CPU, they are ideal choice for building products such as Programmable Logic Controllers (PLC), complex remote-IO slaves, industrial switches, gateways or operator terminals.

RZ/N1 Family consists of three devices, out of which the RZ/N1D, is an industrial ethernet communication System on Chip with 5-ports, in a 400BGA package.

This guide shall assist you to quickly start with your evaluation of the RZ/N1D device on the related Solution Kit board(s) and the Solution Kit software and documentation. It represents a short guide for the first time users with the instructions for the first startup of the board.

In addition to this document, please be aware of the extensive documentation on RZ/N1 devices, evaluation boards and software that you may find in the Solution Kit Documentation directory.

The Solution Kit boards can solely be used for evaluation purposes. It is not allowed to use the boards for commercial purposes.

List of reference documents

Document name
User's Manual: System Introduction, Multiplexing, Electrical and Mechanical Information
User's Manual: System Control and Peripheral
User's Manual: Peripherals
User's Manual: R-IN Engine and Ethernet Peripherals
User's Manual: Generic Open Abstraction Layer
RZ/N1D Development Board Schematic
RZ/N1D Development Board Setup Notes
RZ/N1 U-Boot User Manual
RZ/N1 Linux User Manual

List of Abbreviations and Acronyms

Abbreviation	Full Form
API	Application Programming Interface
BSP	Board Support Package
CO	CANopen Bus Protocol
DLR	Device Level Ring Protocol
EtherCAT / ECAT	EtherCAT Network Protocol
EtherNet/IP / EIP	EtherNet/IP Network Protocol
EPL	Ethernet POWERLINK Network Protocol
PNIO	PROFINET IO Protocol
PROFINET	Process Field Network Protocol
lwIP	Lightweight IP Protocol
GOAL	Generic OS Abstraction Layer, Communication Framework on RZ/N1x
LLDP	Link Layer Discovery Protocol
NVM	Non-volatile memory
OS	Operating System
RT	Real-Time
API	Application Programming Interface
HW	Hardware
SW	Software
HSR	High-Availability Seamless Redundancy Protocol
PRP	Parallel Redundancy Protocol
IBIS	I/O Buffer Information Specification for pin signal simulation

1. Overview

The following chapters will help you setup the hardware and software environment and run a sample application on your RZ/N1L evaluation board.

- The Solution Kit is being actively updated and new versions of it are iteratively released
- Current version of Solution Kit is providing a snapshot of the latest software and documentation versions at the release time

The **CONNECT IT! ETHERNET RZ/N1D** Solution Kit is based on the RZ/N1D, a 5-port, industrial Ethernet communication SoC in a 400BGA package. The device features two independent blocks integrated in a single package – a communication block based on the proven multi-protocol R-IN Engine, as well as an application block with a dual ARM® Cortex®-A7 and a variety of peripherals, including a DDR memory interface required for rich operating systems like Linux. Along this external memory the RZ/N1D additionally includes 2 MB internal SRAM. The RZ/N1D Solution Kit contains everything you need for fast evaluation and rapid prototyping of multiple industrial Ethernet protocols.
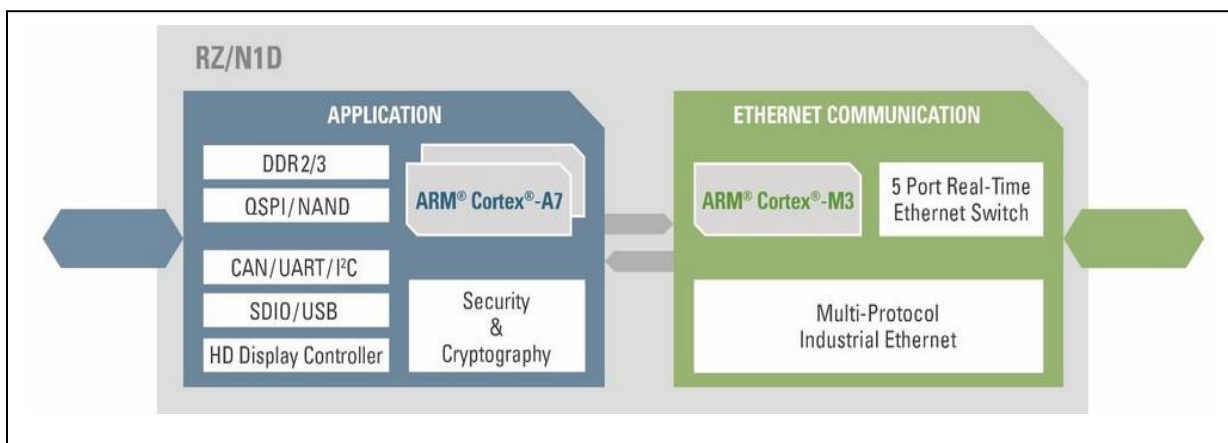
Figure 1-1: RZ/N1D Functional and HW overview

The RZ/N1D Solution Kit contains

- RZ/N1D CPU Development Board
- IAR I-jet Lite debugger (incl. 20pin flat ribbon and USB-micro/A cable)
- USB Cable Micro / Type-A
- DVD RZ/N1x-DB Solution Kit for all RZ/N1x devices / boards

Additionally, one can order the RZ/N1-EB Expansion Board separately, that provides the user with the possibility to use an expanded range of functionalities and RZ/N1D interfaces. RZ/N1-EB kit contains:

- RZ/N1-EB Expansion Board
- Power Supply 24V/750mA
- Power Supply Primary Adapters Euro/UK/JPN-US

The expansion board has no stand-alone capability and requires an RZ/N1D or RZ/N1S CPU board.

2.   Hardware Setup

2.1   CPU Demo Board Setup

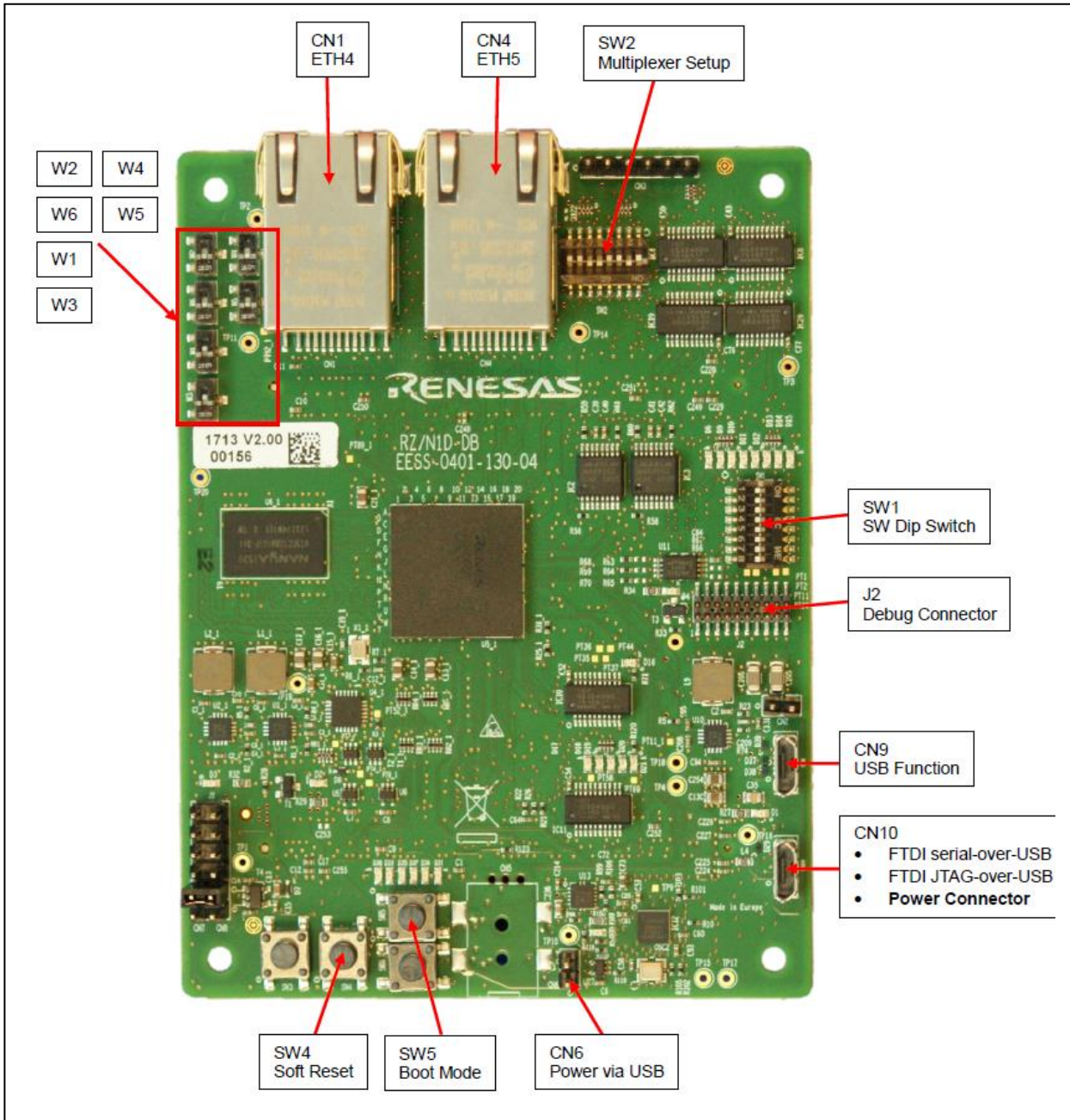To get started quickly with the board, please identify the connectors on the RZ/N1D-DB CPU Board.

Figure 2-1: Board connectors overview

1. Switches and Jumpers
Please take care that the following switches are set according to the tables below.

Table 1: RZ/N1D GPIO Multiplexer DIP-Switch SW2

Number	SW ON (low)	SW OFF (high)	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	OFF

Table 2: RZ/N1D Config DIP-Switch

Switch	SW ON (white bar)	SW OFF	Setting
W1	JTAG Mode	ARM Coresight Mode	OFF
W2	RXCLK4 from PHY	RXCLK4 from GPIO61	**ON**
W3	Boot Mode NAND	Boot Mode QSPI/USB	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**

Table 3: Jumper Settings RZ/N1D-DB

USB Power

Jumper	On Setting (default)	Off Setting
CN6	Power from USB	Power from ext. Supply

SRESET from Debugger to RZ/N1D

Jumper	On Setting (default)	Off Setting
CN7	RZ/N1D is connected with the SRESET from the Debugger	Debugger cannot reset the CPU

MRESETout

Jumper	On Setting	Off Setting (default)
CN8	MRESETOUT is connected to the Debugger	Debugger cannot detect any internal CPU reset

USB Host/Device Mode

Jumper	On Setting	Off Setting (default)
CN2	OTG mode ID	Device Mode ID

Table 4: Boot config dip switches

The single dip switches are used to select the power on setup.

Boot Mode

Switch	On Setting (white bar) (default)	Off Setting
W3	Boot Mode NAND	Boot Mode QSPI / USB

Debug Mode

Switch	On Setting (white bar)	Off Setting (default)
W1	JTAG Mode	ARM Coresight Mode

GPIO Pull up

Switch	On Setting (white bar) (default)	Off Setting
W5	LCD [145:127,73:62] pull up	LCD [145:127,73:62] pull down

GPIO Pull up

Switch	On Setting (white bar) (default)	Off Setting
W6	LCD [111:105,59:36] pull up	LCD [111:105,59:36] pull down

Table 5: Ethernet RX Clock

ETH4: RX_CLK

Switch	On Setting (white bar) (default)	Off Setting
W2	RXCLK4 from Phy	RXCLK4 from GPIO61

ETH5: RX_CLK

Switch	On Setting (white bar) (default)	Off Setting
W4	RXCLK5 from Phy	RXCLK5 from GPIO61

Instead of QSPI it is possible to boot from USB, if SW5 is pressed down during the RESET.

Table 6: Push buttons

SW5:

Boot Mode Select

Switch	On Setting	Off Setting
SW5	Boot from USB	Boot from QSPI

Table 7: Reset and NMI switches

SW3:

Power On Clear

Switch	On Setting	Off Setting
SW3	POR Reset	Release POR Reset

SW4:

Soft Reset

Switch	On Setting	Off Setting
SW4	Soft Reset	Release Soft Reset

SW6:

NMI

Switch	On Setting	Off Setting
SW6	Generate CM3 NMI	Release CM3 NMI

If you are using the expansion board together with RZ/N1D CPU Board, please remove the jumper CN6.

1. **JTAG Debug Connector**
    This connection is required for software development purposes, to be able to run and debug the software on the target. The IAR Debugger cable has one unused key pin, therefore pin 7 of the **J2** connector should be cut or bent, as shown below. After this step, you can attach your I-jet debugger to the board via JTAG cable to the PC via USB.
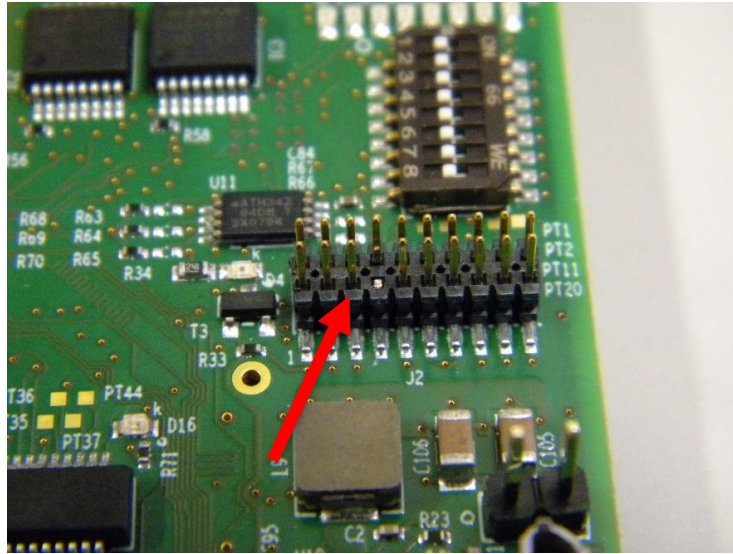

Figure 2-2: Connector J2 with removed pin 7
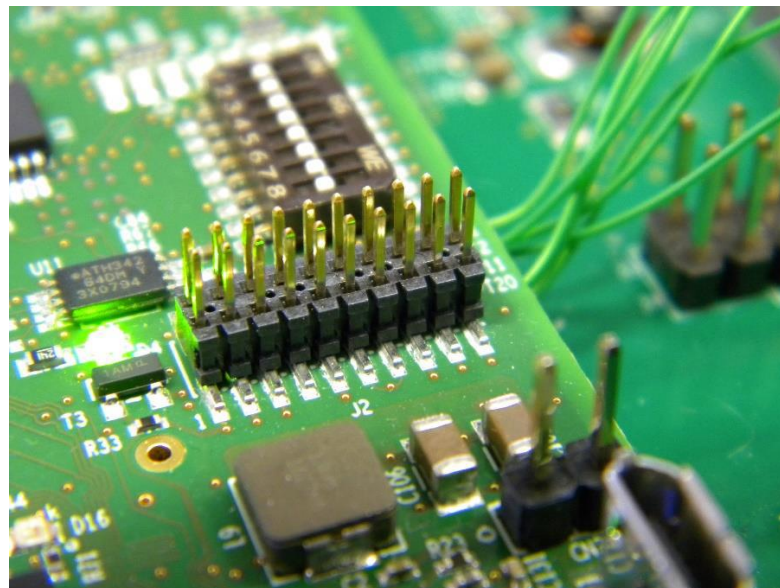
Figure 2-3: I-Jet Debugger Connector

2. **Power connector and serial-over-USB**
    The board is powered via USB connector CN10, if not attached to the expansion board. If you use the CPU board standalone, please make sure that the jumper CN6 is ON, otherwise OFF. Please connect the board to your PC via USB cable to the connector **CN10**. The same interface is used for serial UART communication with the board, so if you have a windows PC, after connecting the

board via USB, you should be able to see a new device registered 4 new USB Serial COM ports. You may use any of the terminal emulator on your PC to open a serial port connection to the board. 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**, provided you do not have any other USB devices attached.
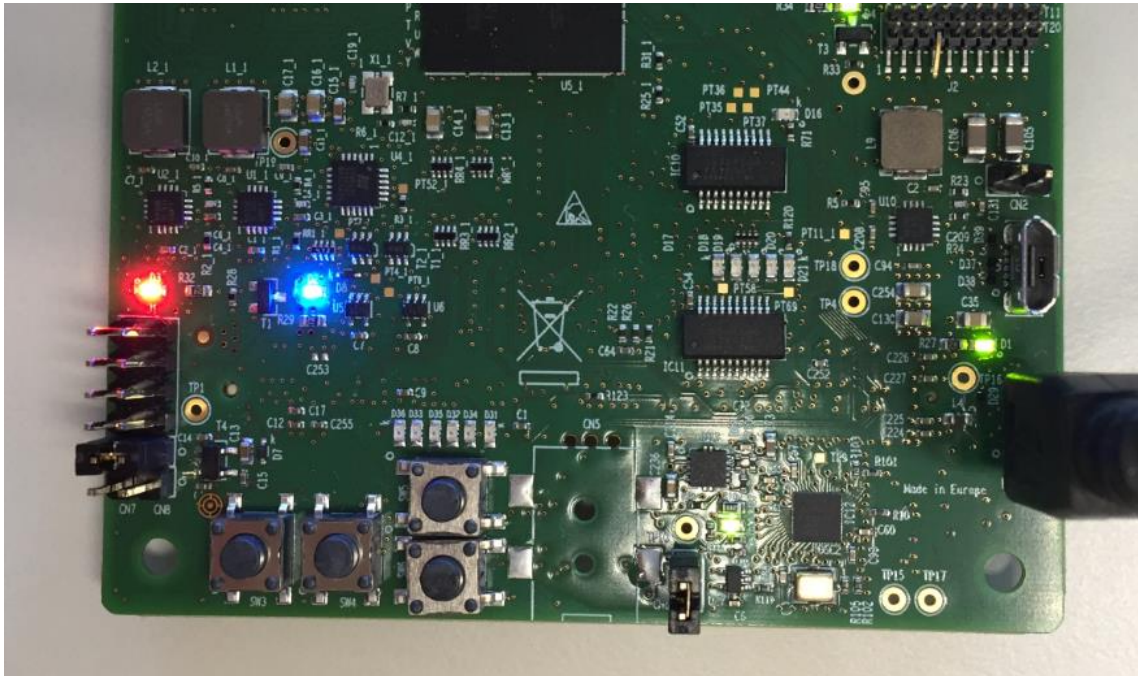

Figure 2-4: Power connector and serial over USB

If the board is not recognized by Windows, please install the FTDI driver that you may find under …\YCONNECT-IT-RZN_V1.x\Tools\FTDI.

You can perform a soft-reset of the board by pressing **SW4**.

3. **Ethernet Interface**

The Ethernet connection to the PC or a PLC can be prepared using one of the two RJ-45 connectors CN4 or CN1 (for an overview of connectors please check Figure 2-1). For running any Ethernet fieldbus sample program, you can use any of the two connectors. There are two PHY Micrel chips connected to these RJ-45 connectors.

If you are using the expansion board, Ports J23 and J24 can be used of real-time Ethernet and port J22 is reserved for Linux. You can connect J22 to a dedicated Network Interface Card (NIC) on your PC to open a secure shell to Linux. Linux accesses GMAC1 on the RZ/N1 device.

2.2   Expansion Board Setup

If you are using the expansion board together with your RZ/N1D CPU Board, here you may find some setup notes for it. The expansion board comes with an external power supply to drive the expansion board and the RZ/N1x CPU board plugged onto the connectors J1 and J2. Please note the position of the power supply selection jumper below and ensure that the 12…24V position is chosen. This board expands the RZ/N1D or RZ/N1S CPU boards with 3 Gbit PHY ports, an SD-Card slot, a USB Host as well as CAN, RS232, RS485, PMODs, LCD display interface with integrated touch screen, etc.
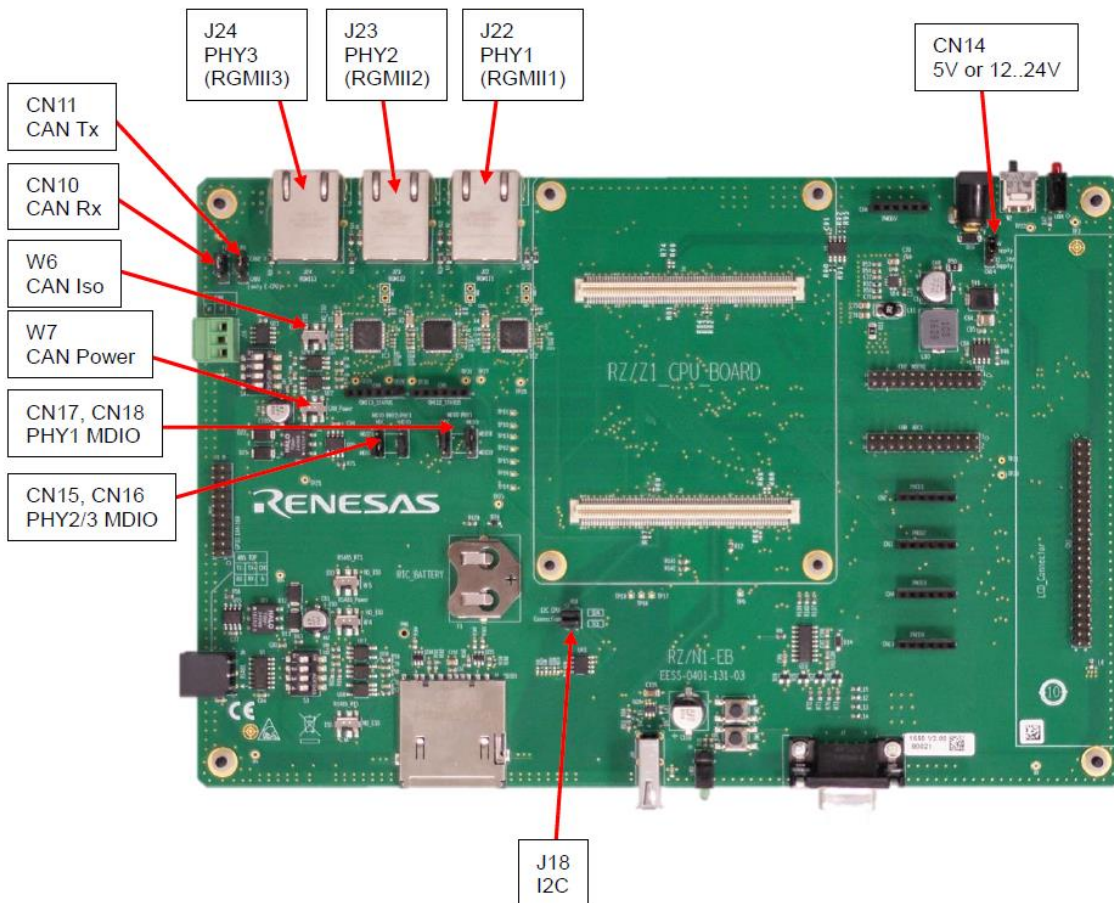

Figure 2-5: RZ/N1-EB Expansion Board Overview

If using the RZ/N1-EB expansion board, please ensure the following jumper and switch settings.

Table 8: Expansion Board switch settings

CN10	CAN Rx	Connect pins 1 and 2 (CAN2)
CN11	CAN Tx	Connect pins 1 and 2 (CAN2)
CN15	PHY2/PHY3 MDC	Connect pins 2 and 3 (MDIO2)
CN16	PHY2/PHY3 MDIO	Connect pins 2 and 3 (MDIO2)
CN17	PHY1 MDC	Connect pins 1 and 2 (MDIO1)
CN18	PHY1 MDIO	Connect pins 1 and 2 (MDIO1)
J18	I2C SDA	Connect pins 1 and 2
J18	I2C SCL	Connect pins 3 and 4
W6	CAN Isolation	NO_ISO
W7	CAN Power	NO_ISO

IMPORTANT INFORMATION: Please take care about using the right 5V power supply when modifiying the power scheme with CN14 from 12-24V to 5V! A wrong power supply can destroy the Expansion board and the CPU board powered via the circular power supply connector J4!

3. Software Setup

RZ/N1D Solution Kit contains a software package on a DVD (also downloadable on Renesas' official website) that enables the user to evaluate various functionalities of our industrial automation multi-core chip.

The software for RZ/N1D comes as either IAR compilable code for Cortex A7 and Cortex M3, or as Linux and U-boot that run on Cortex A7. ThreadX and the software running on Cortex M3 are compiled with IAR compiler. Linux and U-Boot are provided as ready binaries to run on the board. If you have the desire to compile the Linux kernel or U-boot, you need GNU environment and you can refer to the documentation in YCONNECT-IT-RZN_V1.X\Software\U-Boot-and-Linux\Documentation.

Since the protocol stacks are running on the Cortex-M3 part, you need to install IAR Embedded Workbench for ARM (EWARM) for compiling and running the software on Cortex M3.

Table 9 gives an overview of development environments and available operating systems for both cores.

Table 9: Development Environment for RZ/N1D

Device	RZ/N1D		
CPU	Cortex-A7		Cortex-M3 (R-IN engine)
OS	Linux **or** ThreadX		uITRON (HWRTOS)
IDE	Linux	ThreadX	IAR EWARM
Bootloader	U-Boot*		

*U-Boot is used as bootloader, it runs on Cortex A7, but it can boot Images to be executed on both cores.

Please follow the below steps to setup the software environment for RZ/N1D.

1. **Install IAR EWARM**
   As the first step in software setup, please install the IAR EWARM -> installation file for IAR EWARM 8.30.2 can be found under **YCONNECT-IT-RZN_V1.x\Tools\IAREWARM**.

   Please note that the current RZ/N1D software from the Solution Kit was tested only with IAR EWARM v8.30.2. During the installation process, you will be prompted to select your license type. For evaluation purposes, you can first choose the 30-day license, if you need the license for a longer period, please contact RZ/N1 support team - rzn_support-eu@lm.renesas.com.

During the installation process, you can continue with the steps below.

2. **Locate the RZ/N1D IAR software projects in the Solution Kit**
The software is stored in the folder **YCONNECT-IT-RZN_V1.x\Software** in the Solution Kit. This folder consists of software packages for all RZ/N1 platforms, not only for RZ/N1D. It is therefore important to mention which projects are meant for being executed on RZ/N1D and how to differentiate between projects running on Cortex A7 and on Cortex M3. Table 10 gives an overview about that.

More detailed software documentation can be found under **YCONNECT-IT-RZN_V1.x\Documents\RZN.Software**. The IAR projects represent a collection of required source files to build an executable and run it on the target.

Table 10: Overview of IAR Projects for RZ/N1D

IAR Project	Description	CPU	Path to the workspace file in the Sol. Kit YCONNECT-IT-RZN_V1.x\Software\...
x-ware*	Set of projects running on the Cortex A7 including ThreadX, NetX Duo and GUIx sample applications	Cortex **A7**	\ThreadX\rzn1\iar\x-ware_platform.eww ThreadX\rzn1_smp\iar\x-ware_platform.eww
GOAL Projects**	Several example projects showing the different functionalities for the switch management and GOAL	Cortex **M3**	\GOAL\goal\projects**00410_goal**
Protocol Stack Projects***	Several Protocol Stack slave application examples, which are running I/O communication between a PLC and RZ/N1D	Cortex **M3**	\GOAL\goal\projects**goal_co_lib** \GOAL\goal\projects**goal_ecat** \GOAL\goal\projects**goal_eip_lib** \GOAL\goal\projects**goal_epl_lib** \GOAL\goal\projects**goal_mbs** \GOAL\goal\projects**goal_pnio_lib**

* x-ware User Guide:

\YCONNECT-IT-RZN_V1.X\Documents\RZN.Software\ThreadX\X-Ware_Platform_RZ_N1_User_Guide.pdf

Please be aware the projects ending with **_smp** in their names are meant to be executed on RZ/N1D only (Symmetric Multi-Processing).

** GOAL application notes document:

\YCONNECT-IT-RZN_V1.X\Software\GOAL\doc\r11qs0008ed0131-rzn1-goal-quick-startguide-management.pdf

*** Protocol Stack quick start guides:

\YCONNECT-IT-RZN_V1.X\Documents\RZN.Software\GOAL\

r01an4239ej0110_rzn1_EtherCAT.pdf

r11qs0012ed0131-rzn1-goal-quick-startguide-canopen.pdf

r11qs0007ed0131-rzn1-goal-quick-startguide-eip.pdf

r11qs0010ed0131-rzn1-goal-quick-startguide-profinet.pdf

r01an4412ej0120-rzn1_Modbus.pdf

r11qs0009ed0131-rzn1-goal-quick-startguide-powerlink.pdf

Please **note** that there are multiple variants of IAR GOAL projects, one for each board configuration:

…**rzn1d_demo_board**\\rzn1d_demo_board.eww   – RZ/N1D CPU Board

…**rzn1d_demo_board_eb**\\rzn1d_demo_board_eb.eww   – RZ/N1D CPU Board + Expansion B.

…\\rzn1l_demo_board\\rzn1l_demo_board.eww   – RZ/N1L CPU Board

…\\rzn1s_demo_board\\rzn1s_demo_board.eww   – RZ/N1S CPU Board

…\\rzn1s_demo_board_eb\\rzn1s_demo_board_eb.eww   – RZ/N1S CPU Board + Expansion B.

Please also **note** that ThreadX IAR workspace from the table above consists of IAR projects for both RZ/N1S and for RZ/N1D. RZ/N1S can run only **non "_smp"** applications.

3. **Locate Linux, DTB and U-Boot binaries**
Linux and U-Boot binaries are to be found in the solution kit under:

| U-Boot for RZ/N1D	YCONNECT-IT-RZN_V1.X\\Software\\U-Boot-and-Linux\\u-boot\\binaries**u-boot-rzn1d400-db.bin.spkg**
Linux Kernel	YCONNECT-IT-RZN_V1.X\\Software\\U-Boot-and-Linux\\kernel\\binaries**uImage**
Device Tree Blob	YCONNECT-IT-RZN_V1.X\\Software\\U-Boot-and-Linux\\kernel\\binaries**uImage-rzn1d400-db.dtb**

Linux on RZ/N1 runs on Cortex A7 core and cannot be used parallely with x-ware applications, since both are running on Cortex A7.

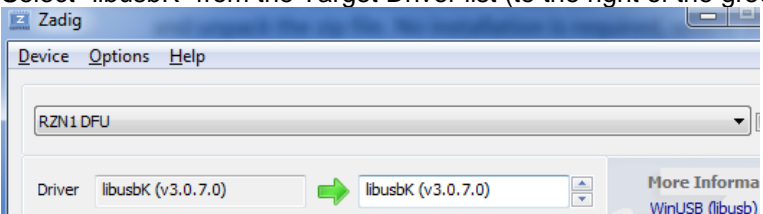4. **Write U-Boot to QSPI flash on the board with DFU utility**

As you can see in Table 9, U-Boot is chosen as the bootloader for RZ/N1D. It runs on the Cortex A7, initializes some of the required interfaces and peripherals and it can be used to load software images for both cores, by setting the corresponding parameters in the U-boot user console. Without U-Boot, you cannot initiate the load of your application via IAR I-Jet Debugger to SRAM for executing on the target. Figure 3-1 provides an overview of execution stages during programming U-Boot in Flash and later when U-Boot is configured to load Cortex M3 and Cortex A7 Images from Flash to SRAM (ThreadX, R-IN Engine) or DDR (Linux) and execute them.

After hardware reset, BootROM will start looking for the U-Boot package in QSPI flash. Therefore, you need to write U-Boot in QSPI flash. This is done via USB DFU (Device Firmware Update). **SW5 button pressed together SW4** signals to BootROM, to switch the boot mode from QSPI to USB DFU. This is required only once, to load U-Boot to SRAM. In the next step, you will write the U-Boot in flash. **Please make sure to connect CN9 on the board to a USB Host connector on your PC. This provides USB DFU.**

RENESAS

a. Find the Windows version of dfu-util in the solution kit under YCONNECT-IT-RZN_V1.X \Tools\dfu-util. Alternatively you can download this dfu-util command line version on https://sourceforge.net/projects/dfu-util/files and unpack the zip file. No installation is required, where this document tells you to run dfu-util, simply run the dfu-util-static.exe – the Windows executable from a Windows command prompt. If you are running Linux on your PC, install the "dfu-util" package: `sudo apt-get install dfu-util`.
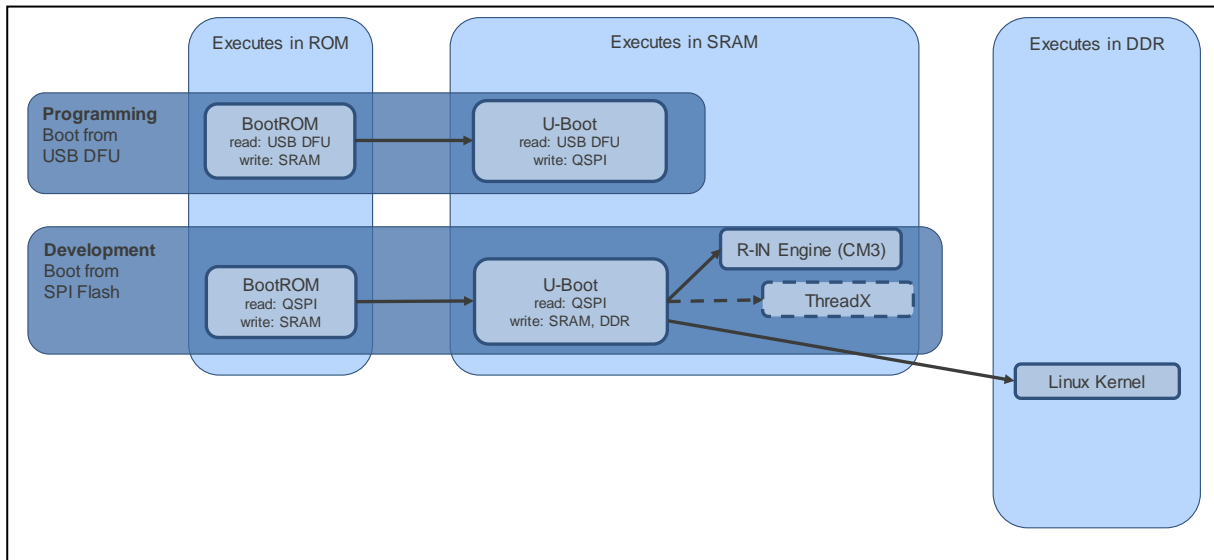
Figure 3-1: Flash and Boot Sequence

b. Start the DFU target
 Connect your RZ/N1 board to your Windows PC using the USB cable, and reset your board in "Boot from USB" mode (i.e. using DFU) -> **press SW5 and SW4 together**.

c. Register the USB device (should be done only once)
 You must register the USB Download Gadget device with the libusbk driver, you only need to do this once. On your Windows PC:

- o Download Zadig from http://zadig.akeo.ie/downloads/zadig-2.3.exe
- o Launch Zadig, select "List All Devices" from the Options menu.
- o Select "RZN1 DFU" from the pull-down list. Please note that the device might appear as "USB Download Gadget"
- o Select "libusbK" from the Target Driver list (to the right of the green arrow).

- o Click the "Install Driver" button, this will take some time.
- o Quit Zadig

d. Connect CN10 on the board to a USB Host connector on your PC. This provides Serial-over-USB and JTAG-over-USB services. Open a terminal emulator client like TeraTerm or putty, choose the corresponding serial port, where you board is registered to and set the following serial port settings. On Linux PCs, if you have no other serial-over-USB devices attached, this is accessed using /dev/**ttyUSB2**.
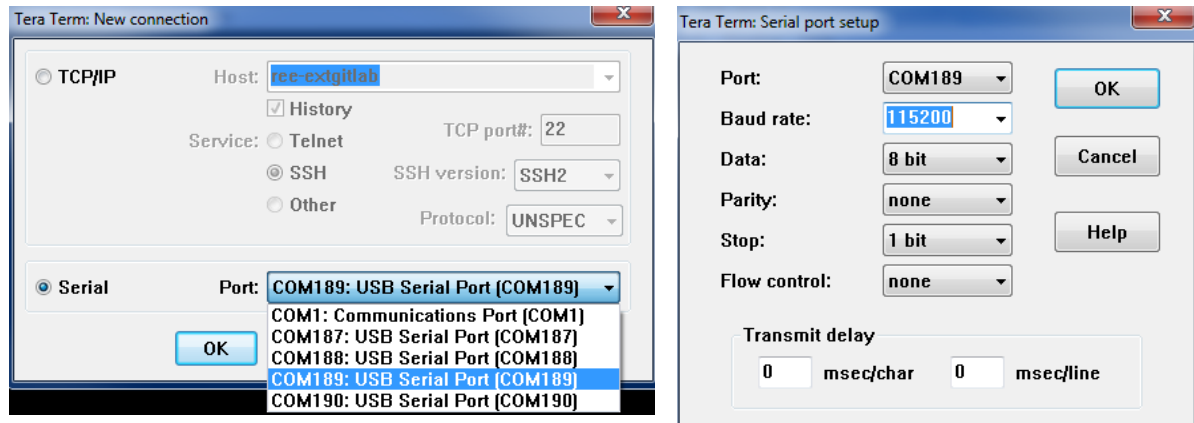
e. On the board, hold down switch SW5 (to select DFU boot mode instead of QSPI) and press switch SW4 (soft reset). The RZ/N1 serial port should output:
```
** BOOTLOADER STAGE0 for RZN1 **
```

Boot source: USB

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

`dfu-util.exe -D "u-boot-rzn1d400-db.bin.spkg"` in Windows from a Windows command prompt

`sudo dfu-util -D u-boot-rzn1d400-db.bin.spkg` in Linux from a terminal

U-Boot should now run in SRAM and the RZ/N1 serial port presents you a console, like this:

```
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.44002000

Hit any key to stop autoboot:  1
```

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

In addition to the above DFU command line tool, RZ/N1 Solution Kit provides a graphical dfu_prog tool. You may find it under YCONNECT-IT-RZN_V1.X\Tools\dfu-util.exe and YCONNECT-ITRZN_V1.X \Tools\dfu-util\dfu_prog\dfu_prog.exe

dfu_prog.exe opens a window, that allows the user to select the file, that he wants to flash, so in a way it abstracts the command line tool by providing the graphical user interface to select the files that user wants to flash. The program generates a Windows command for dfu-util.exe and executes this

command in a terminal. This way the user does not need to open the command prompt and type the dfu commands on his own, since they are automatically generated and executed by the program on user's host PC.

This tool is for Windows users only and is not compatible with Linux OS.

Prior to executing the command, the user needs to take care that the correct boot mode pins are set and that the USB connector on the board (CN9 for RZ/N1D and CN3 for RZ/N1S) are connected to the USB Host on his PC and that U-Boot or BootROM are set in a DFU (USB) mode.

To use this tool, please make sure that the solution kit folder structure is maintained and double click on YCONNECT-IT-RZN_V1.X\Tools\dfu-util\dfu_prog\dfu_prog.exe.
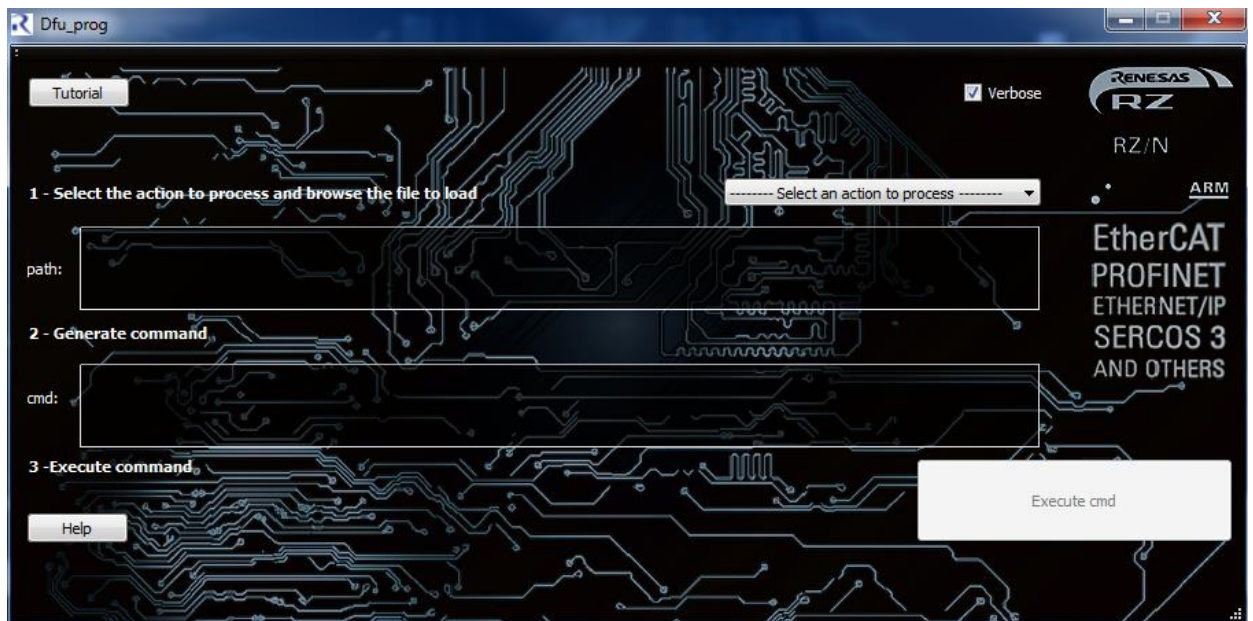
You may find some brief tool user guidelines about this graphical tool in …\YCONNECT-IT-RZN_V1.X\Tools\dfu-util\dfu_prog\r30uz0135ed0002-rzn1-dfu-prog-guideline.pdf

f. 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
```

g. Ensure the U-Boot/SPL region of QSPI Flash is erased, run:
```
sf probe

sf erase 0 10000
```

h. On the U-Boot console, run:
```
dfu
```

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

```
dfu-util.exe -a "sf_uboot" -D "u-boot-rzn1d400-db.bin.spkg"
```
in Windows

```
sudo dfu-util -a "sf_uboot" -D u-boot-rzn1s324-db.bin.spkg
```
in Linux

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, like:

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

j. 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
```

k. 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
```

3.1   Bring up the board with Linux and U-Boot

5. **Write Linux to QSPI flash on the board**
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:

```
dfu-util.exe -a "sf_dtb" -D "uImage-rzn1d400-db.dtb"
```
in Windows
```
sudo dfu-util -a "sf_dtb" -D uImage-rzn1d400-db.dtb
```
in Linux

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

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

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

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

6. **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 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_dtb` and `sf_kernel` targets.

```
setenv bootcmd "sf probe && sf read 0x8ffe0000 b0000 20000 && sf read
0x80008000 1d0000 600000 && bootm 0x80008000 - 0x8ffe0000"
```

Save the environment variables:

```
saveenv
```

7. **Run 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
```

You have now successfully started U-Boot and Linux on the Cortex A7. To run software on Cortex M3, you need to use IAR Embedded Workbench, that you hopefully installed already.

If you wish to use Ethernet within Linux, but <u>do not have an Extension Board</u>, you should 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.

NOTE: this setup does not correspond to the default use case of RZ/N1 in industrial switch applications and with the ***-no_cm3.dtb**, Cortex M3 (communication core) cannot use industrial ethernet interfaces.

- 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.

3.2 Running a sample application with IAR EWARM

In the previous section we saw how you can run Linux on Cortex A7. We can now proceed with running a sample application on the Cortex M3. You need to follow the steps below to load Cortex M3 image in SRAM and execute it from there. Additionally, you can write the images in flash and load them to SRAM after reset, by setting the corresponding U-Boot parameters, as described in chapter 3.3.

3.2.1 Run and debug Cortex M3 software from RAM with IAR Debugger

1. Power the board and open a serial terminal to U-Boot as explained in the previous chapters and reset the board - SW4
2. Open the IAR EWARM 8.30.2 that you previously installed
3. Click on File -> Open Workspace as depicted below
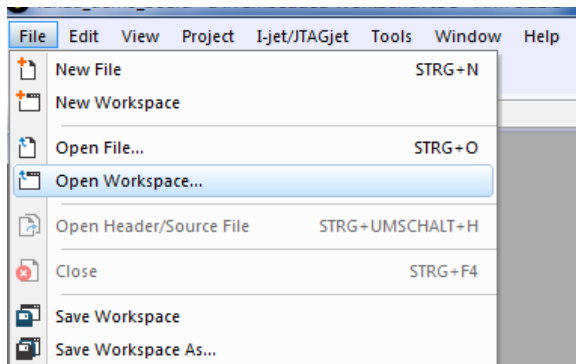
4. Select the workspace file
   *YCONNECT-IT-RZN_V1.x\Software\GOAL\goal\projects\00410_goal**chase_lights**\iar\renesas\rzn1d_demo_board\rzn1d_demo_board.eww* if you are using the CPU Board standalone, otherwise if you are using it in combination with expansion board, open the *rzn1d_demo_board_**eb**.eww* workspace.

   As already mentioned, there are 5 variants of project settings for each board configuration:

   …**rzn1d_demo_board**\rzn1d_demo_board.eww   – RZ/N1D CPU Board

   …**rzn1d_demo_board_eb**\rzn1d_demo_board_eb.eww   – RZ/N1D CPU Board + Expansion B.

   …\rzn1l_demo_board\rzn1l_demo_board.eww   – RZ/N1L CPU Board

   …\rzn1s_demo_board\rzn1s_demo_board.eww   – RZ/N1S CPU Board

   …\rzn1s_demo_board_eb\rzn1s_demo_board_eb.eww   – RZ/N1S CPU Board + Expansion B.

5. Select rzn1d_demo_board as shown below, right-click on the project name and click on "Make".
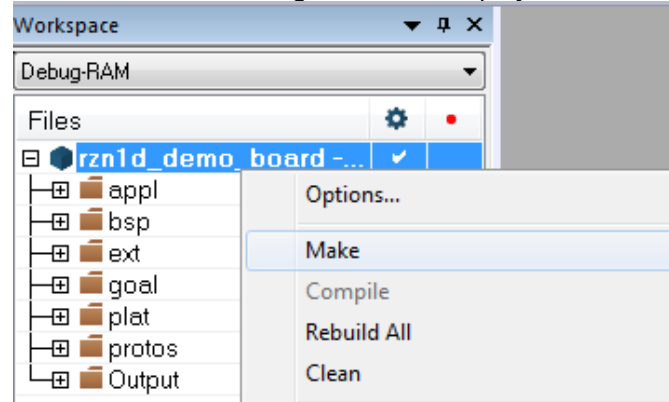
6. After the project compiled with no errors nor warnings, make sure your board is powered on and that the IAR I-Jet Debugger is connected to the JTAG connector.

7. In the U-Boot console (you might need to stop the execution of U-Boot after reset, by pressing any key during the booting) type:

```
=> setenv cm3 "sf probe&&mw 0x04000004 1&&rzn1_start_cm3"
=> saveenv
=> run cm3
```

   Press enter.

8. Click on the "Download and Debug" button in the program control panel in IAR EWARM, as shown below.
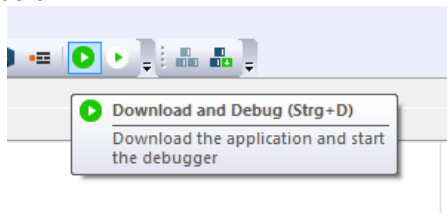
9. In the Debug Log window there should be no errors nor warnings and the program should be stopped at the beginning of the main() now.

```
82          **************************************************
83    └ */
84          #pragma required=__vector_table
  ➡  85      void __iar_program_start( void )
86   ⊟ {
87
88              SystemInit();
89
90              __iar_data_init3();
91
92              //--------------------------
93              // Replace vectors address
94              //--------------------------
95              SCB->VTOR = (uint32_t)__vector_table;
96
97              main();
98
99              exit();
100   └ }
```

10. At this stage you can click F5 on your keyboard or "Go" button in the IAR program flow panel.
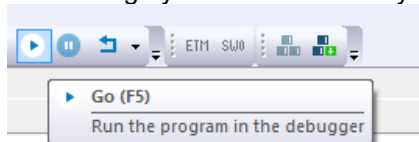
The program should be running now on your board, the user LEDs are toggling from one direction to another and you should see some info output on your serial console as shown below.

```
=> run cm3

SF: Detected mx25l25635f with page size 256 Bytes, erase size 64 KiB, total 32 MiB, mapped at 10000000

=> [CC_I|goal_lmLogLegacy:1152] [I|goal_cmInit:209] Calculated config size of 656 in 3 modules

[CC_I|goal_lmLogLegacy:1152] [I|goal_queuePoolBufsReq:776] ID(35) requests buffers[1556]

[CC_I|goal_lmLogLegacy:1152] [I|goal_queuePoolBufsReq:777]        fixed/temp = 1, 0

[CC_I|goal_lmLogLegacy:1152] [I|goal_queuePoolBufsReq:776] ID(35) requests buffers[1556]

[CC_I|goal_lmLogLegacy:1152] [I|goal_queuePoolBufsReq:777]        fixed/temp = 2, 0

[CC_I|goal_lmLogLegacy:1152] [I|goal_queuePoolBufsReq:776] ID(35) requests buffers[1556]

[CC_I|goal_lmLogLegacy:1152] [I|goal_queuePoolBufsReq:777]        fixed/temp = 1, 0

[CC_I|goal_lmLogLegacy:1152] [I|goal_taskCreate:74] creating task: Timer

[CC_I|goal_lmLogLegacy:1152] [I|goal_init:190] GOAL initialized

[CC_I|goal_lmLogLegacy:1152] [I|appl_setup:85] Chase Lights application started.
```

```
[CC_I|goal_lmLogLegacy:1152] [I|goal_memInitDone:128] fixed memory usage: 16020/196608 bytes
[CC_I|goal_lmLogLegacy:1152] [I|goal_memInitDone:129] fixed memory usage: (9%)
```

### 3.2.2   Run and debug Cortex A7 software from RAM with IAR Debugger

The steps are almost the same as the ones described in chapter3.2.1. Reset the board by pressing SW4. Follow the steps 1-4 from the previous chapter with the difference that you should open a Cortex A7 workspace file in IAR EWARM (step 4):

YCONNECT-IT-RZN_V1.X\Software\ThreadX\rzn1\iar\x-ware_platform.eww

Please note that the projects ending with "**_smp**" are meant for the execution only on RZ/N1D. SMP stands for Synchronous Multiprocessing and it matches the dual-core Cortex A7. Nevertheless, you can also run the non-smp projects on RZ/N1D, however smp projects will not run on the RZ/N1S, since RZ/N1S contains a single-core Cortex A7. The smp projects you may find under:

YCONNECT-IT-RZN_V1.X\Software\ThreadX\rzn1_smp\iar\x-ware_platform.eww

5. After opening the workspace, choose one of the projects in the workspace as shown below. Choose demo_threadx project as an example.
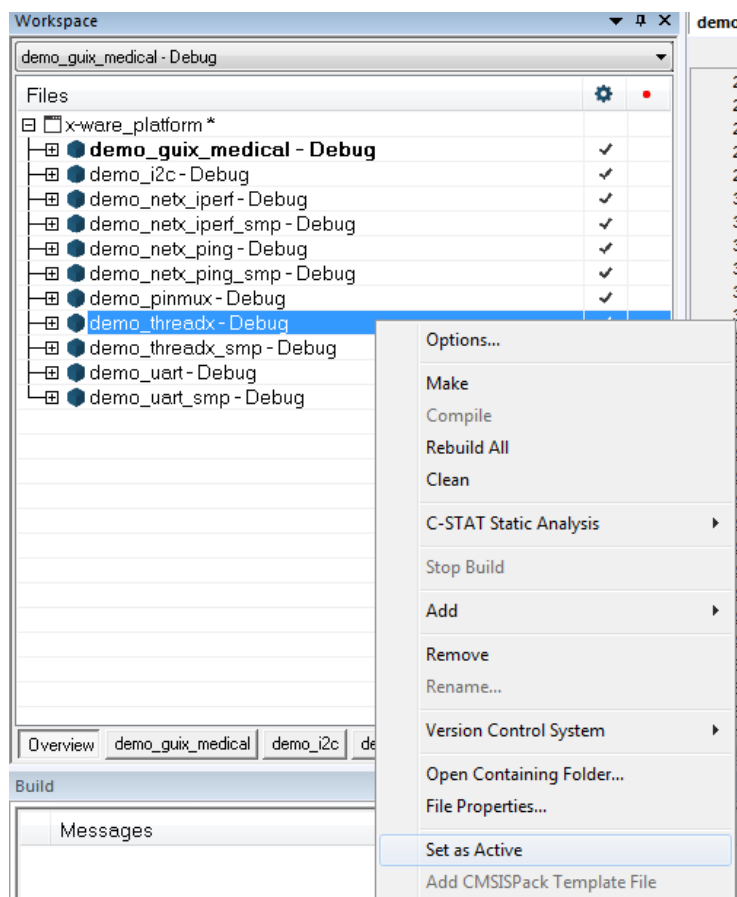

6. After the project compiled with no errors nor warnings, make sure your board is powered on and that the IAR I-Jet Debugger is connected to the JTAG connector.

7. Click on the "Download and Debug" button in the program control panel in IAR EWARM, as shown below.
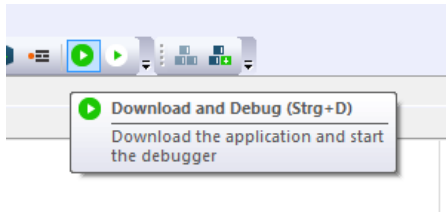
8. In the Debug Log window there should be no errors nor warnings and the program should be stopped at the beginning of the main() now.

9. At this stage you can click F5 on your keyboard or "Go" button in the IAR program flow panel.
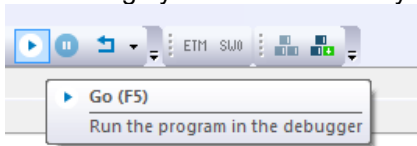
    a. If you are running the smp version of the project (demo_threadx_smp), please check that the debugger is rightly configured for multicore, first you need to select 'run/step/stop affect all cores' mode (the last icon on the right of the toolbar)
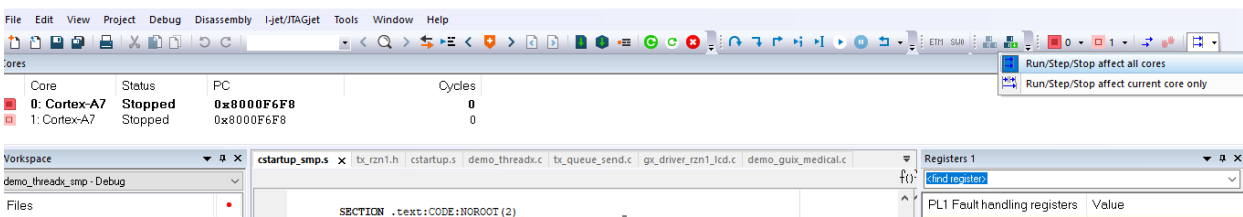
    then when you press 'go' the two cores should be running (in green):
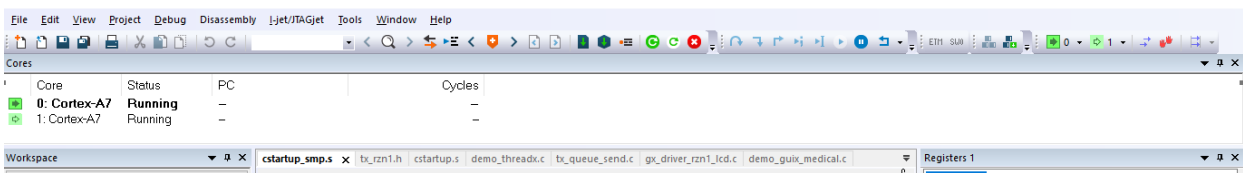
The program should be running now on your board and you should see some info output on your serial console as shown below.

3.3 Boot a sample application from QSPI flash

3.3.1 Boot Cortex M3 IAR binary from flash

To store your binary in QSPI flash and execute it from there after every new reset, you need to write that binary in flash and set the corresponding U-Boot parameters. This procedure was already explained for Linux in chapter 3, bullet point 5 *"Write Linux to QSPI flash on the board"*.

The same principle you need to follow for booting the binaries that are built with IAR. Please follow the steps below to do that.

1. Open an rzn1d_demo_board project for Cortex M3 in IAR, for example:
   YCONNECT-IT-RZN_V1.x\Software\GOAL\goal\projects\00410_goal\chase_lights\iar\renesas\rzn1d_demo_board\rzn1d_demo_board.eww
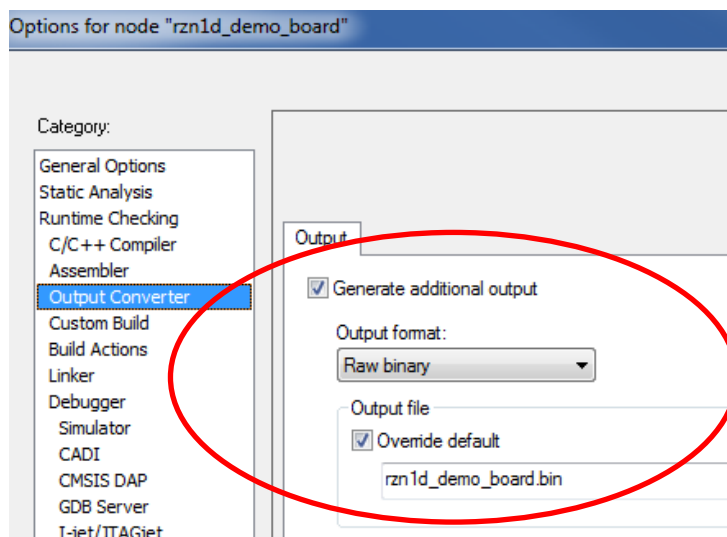
2. Right-click on the rzn1d_demo_board and click on "Options". Click on "Output Converter" in the left tab of the options window and check if the the box "Generate additional output" is ticked and select raw binary output format, if not already set per default for that project.

3. Compile the project. Your raw binary is in the exe folder of the project directory. In the build log you will see something like:

   Saving binary file to …\goal\projects\00410_goal\chase_lights\iar\renesas\rzn1d_demo_board\Debug-RAM\Exe\rzn1d_demo_board.bin

4. Connect CN9 on the board to a USB Host connector on your PC. This provides USB DFU. Reset the board (SW4) and open the serial terminal to the U-boot console. In the U-boot console type: **dfu**

```
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.44002000

Hit any key to stop autoboot:  0

=> dfu
```

5. Open the command window in Windows or terminal in Linux and type the command to write the raw binary in flash:

   dfu-util.exe -a "sf_cm3" -D "rzn1d_demo_board.bin"   in Windows

   sudo dfu-util -a "sf_cm3" -D "rzn1d_demo_board.bin"   in Linux

```
dfu-util 0.9


Copyright 2005-2009 Weston Schmidt, Harald Welte and OpenMoko Inc.

Copyright 2010-2016 Tormod Volden and Stefan Schmidt

This program is Free Software and has ABSOLUTELY NO WARRANTY

Please report bugs to http://sourceforge.net/p/dfu-util/tickets/


Invalid DFU suffix signature

A valid DFU suffix will be required in a future dfu-util release!!!

Opening DFU capable USB device...

ID 045b:0239

Run-time device DFU version 0110

Claiming USB DFU Interface...

Setting Alternate Setting #5 ...

Determining device status: state = dfuIDLE, status = 0

dfuIDLE, continuing

DFU mode device DFU version 0110

Device returned transfer size 4096

Copying data from PC to DFU device

Download        [=========================] 100%        40414 bytes

Download done.

state(7) = dfuMANIFEST, status(0) = No error condition is present

state(2) = dfuIDLE, status(0) = No error condition is present

Done!
```

6. Now your Cortex M3 image is stored in flash and you need to set the U-boot environment variables to boot it after reset. Please type the following commands in the U-boot console:

```
=> setenv bootcmd "sf probe && sf read 0x4000000 d0000 80000 &&
rzn1_start_cm3"

=> saveenv
```

After the next reset (SW4), the Cortex M3 image is booted and executed automatically by U-Boot.

3.3.2 Boot Cortex A7 IAR binary from flash

If you wish to write an IAR Cortex A7 binary instead of Linux Kernel, you need to follow the previous steps 1-4, with the difference that you need to select the corresponding Cortex A7 project in step 1, instead of Cortex M3 one, for example demo_threadx project from the workspace *YCONNECT-IT-RZN_V1.X\Software\ThreadX\rzn1\iar\x-ware_platform.eww*. The binary file you may find under

YCONNECT-IT-RZN_V1.X\Software\ThreadX\rzn1\iar\demo_threadx\Debug\Exe\demo_threadx.bin

5. Open the command window in Windows or terminal in Linux and type the command to write the Cortex A7 raw binary in flash:

    dfu-util.exe -a "sf_kernel" -D "demo_threadx.bin"   in Windows

    sudo dfu-util -a "sf_kernel" -D "demo_threadx.bin"   in Linux

6. Now your Cortex A7 binary is stored as well in flash and you need to set the U-boot environment variables to boot it after reset. Please type the following commands in the U-boot console:

```
=> setenv bootcmd "dcache off&&sf probe&&sf read 0x80008000 1d0000
600000&&go 0x80008000"

=> saveenv
```

After the next reset (SW4), the Cortex A7 image is booted and executed automatically by U-Boot.

3.3.3 Boot both Cortex A7 and Cortex M3 image from flash

7. If you followed steps 1 to 5 from the previous 2 chapters, now you have both IAR binaries (images) stored in flash, you could theoretically boot and execute both from U-Boot. To do that, please type the following commands in U-boot console:

```
=> setenv bootcmd "sf probe && sf read 0x4000000 d0000 80000 &&
rzn1_start_cm3&&sleep 4&&dcache off&&sf read 0x80008000 1d0000 600000&&go
0x80008000"

=> saveenv
```

8. Alternatively, you can run Linux on Cortex A7 and GOAL Sample Application on Cortex M3 simultaneously, provided you wrote Linux kernel and dtb in flash as described in chapter 3, sections 5. **Write Linux to QSPI flash on the board**. For doing so, please type the following commands in U-Boot console and reset the board (SW4):

```
=> 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"

=> saveenv
```

Reset the board – SW4. After the autoboot period of couple of seconds, you have both binaries running simultaneously.

Revision History

		Description	
Rev.	Date	Page	Summary
0.1	22.Feb.2018	-	Draft – first revision
0.2	23.Mar.2018	-	Added eth rx clk and push button settings table and dfu-util driver information
0.3	25.Jul.2018	-	Typo fixes, grammar check and adaptation to the GOAL for Solution Kit v1.3.1
0.4	11.Sep.2018	-	Inserted important notes regarding the flexible power supply structure on the RZ/N1 Expansion Board
0.5	24.Sep.2019	-	Modified licensing description
0.6	31.Jan.2020	-	Changed format

RENESAS

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

    A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

2. Processing at power-on

    The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.

3. Input of signal during power-off state

    Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.

4. Handling of unused pins

    Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.

5. Clock signals

    After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.

6. Voltage application waveform at input pin

    Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).

7. Prohibition of access to reserved addresses

    Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.

8. Differences between products

    Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.

2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.

3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.

4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.

5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

    "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.

    "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

    Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.

6. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.

7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.

8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.

9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.

10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.

11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.

12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.