YRDKRX62N

Rev. 1.1
December 6, 2010

Project Generator Guide

Introduction

This document describes the applications created by the Project Generator for HEW that ship with the YRDKRX62N Demonstration Kit.

Target Device

Renesas YRDKRX62N

Contents

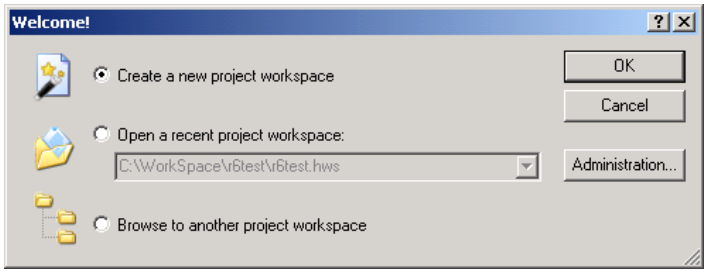
1. Introduction

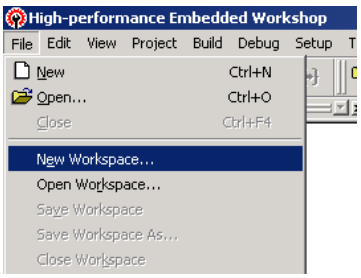
The Renesas High Performance Embedded Workshop (HEW) provides embedded systems developers with a complete, integrated development environment (IDE) dedicated to creating, building, and debugging firmware for Renesas MCUs. HEW has an open architecture that allows tools to be “plugged in” to the base installation to extend functionality and offer new features. This includes the capability to add Project Generators that help automate the creation of new firmware projects. Included on the CD that ships with the Renesas YRDKRX62 Demonstration Kit is a Project Generator that creates projects that demonstrate various peripherals on the RX62N MCU that is on the YRDK board.

This document details how to use the Project Generator, along with a brief description of the projects that the Generator creates.

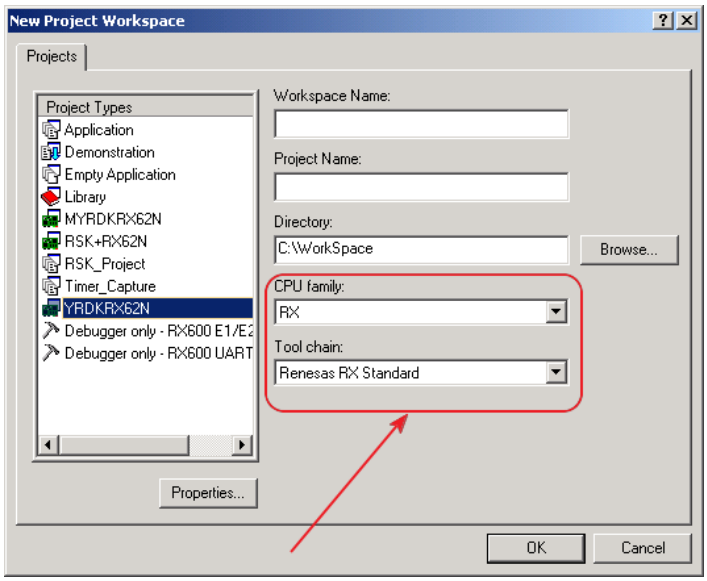
2. Using the Project Generator

To use the Project Generator, start HEW. The following screen is displayed. Choose “Create a new project workspace”. If you’ve already bypassed the welcome screen or are in another project, select the “Create new workspace” option from the file menu.

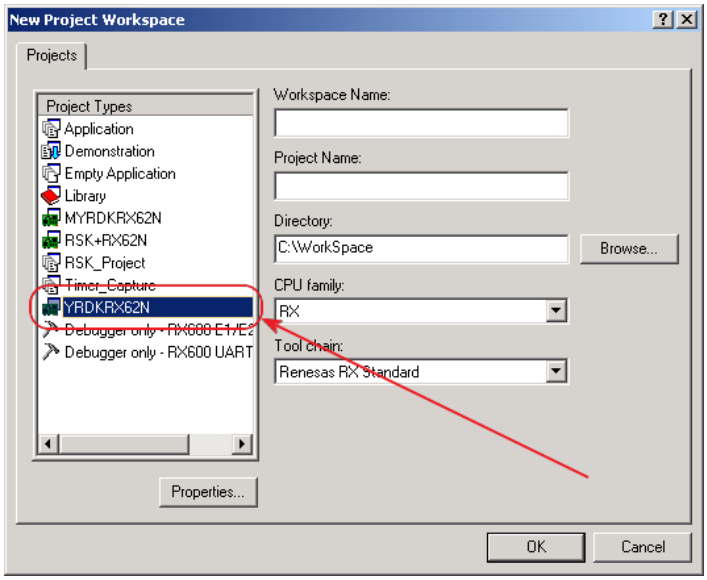




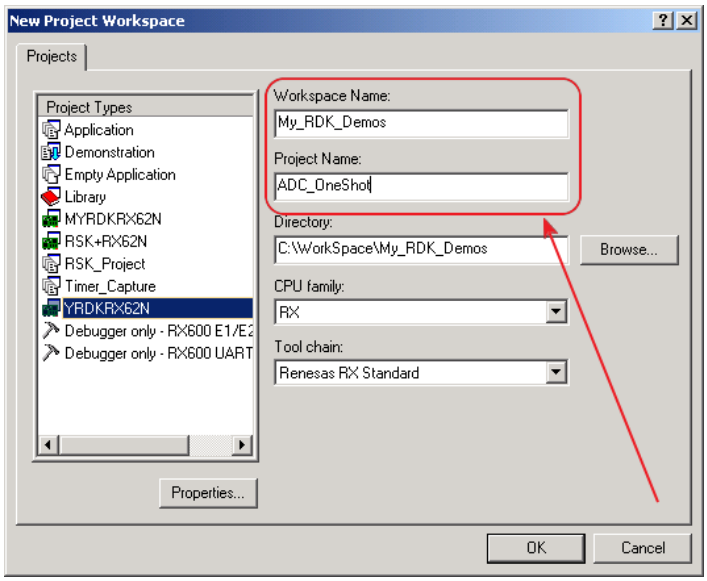
The New Project Workspace dialog is shown. To show the Project Generator for the YRDKRX62N, first select the correct CPU (“RX”) and compiler (“Renesas RX Standard”) as shown. Your list of Project types may be different, and it changes based on the selected CPU and Tool chain.



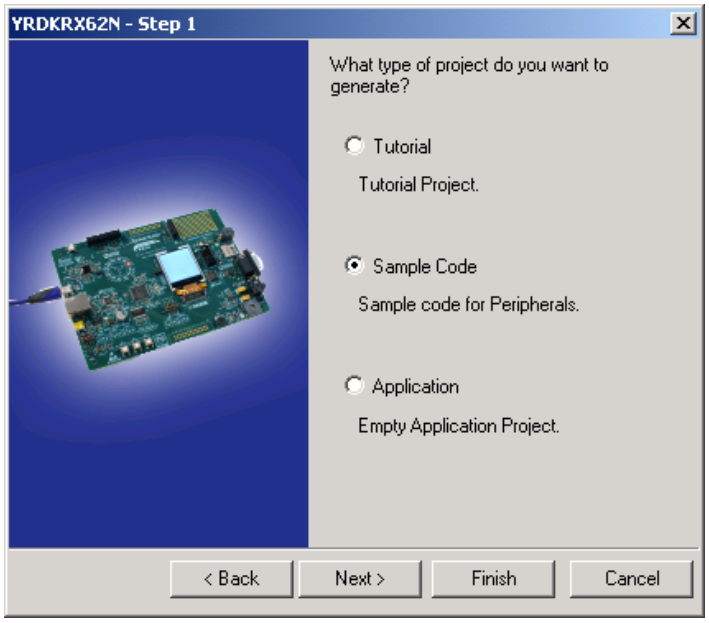
Next, choose “YRDKRX62N” from the list of Project Types on the left:



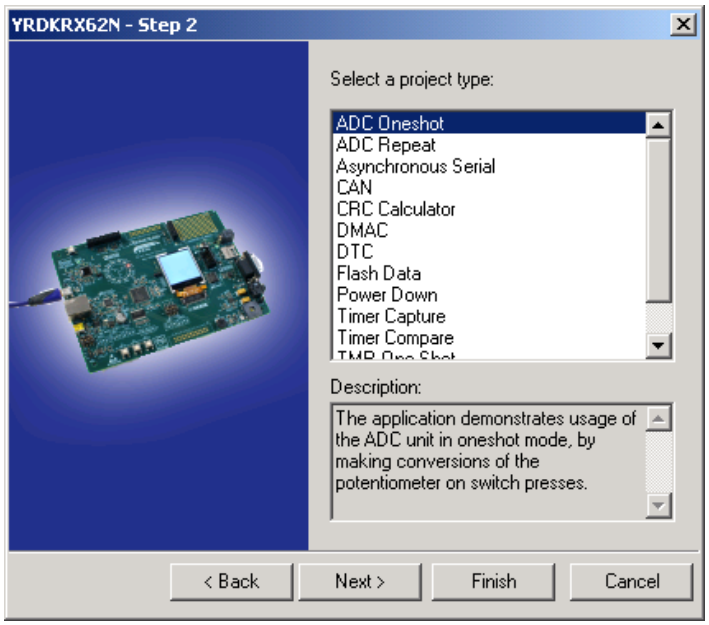
Now give your new workspace a name and give the first project in the workspace a name. An example is shown below:



Click “OK” and the Project Generator starts, displaying a dialog that offers three different types of projects: a tutorial, a sample code project that demonstrates one or more peripherals or an empty application with not much more than startup code. In this example, a Sample Code project is selected. Click “Next” to continue.

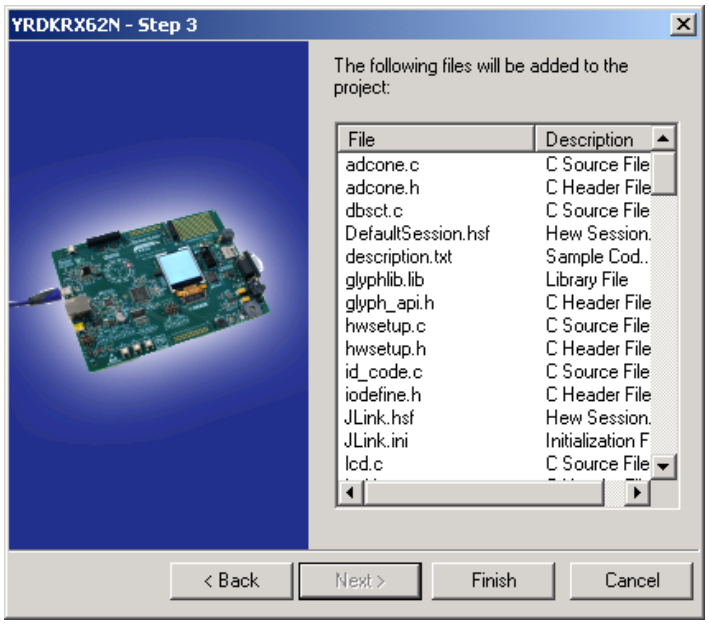


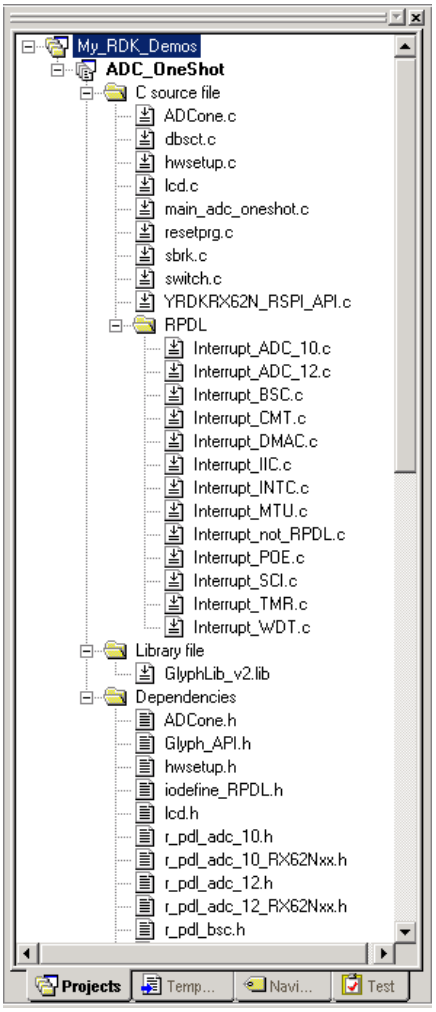
For the Tutorial and Application types there are no other options. With the Sample Code project type, a list of project types is offered. Each type generates a complete running program that demonstrates the use of one or more peripherals. As you highlight each type in the top list, a short description is shown. For this example, we’re going to create a project that shows how to read the ADC in single conversion mode. Click “Next” to continue.



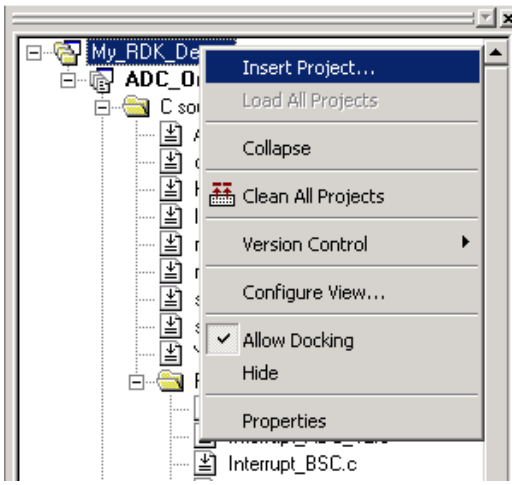
The next dialog shows a list of the files that will be added to your new project. Click “Finish” to continue.

The Project Generator goes to work and creates your new project. The Projects pane in HEW shows the result:





To add more Project Generator code to the same workspace, right click on the workspace name in the Projects pane and choose “Insert New Project…” Repeat the steps above to create the new project.



3. Description of Generated Applications

3.1 Application

This is an empty project with just a “while” loop in the main function. The project demonstrates basic hardware initialization, utilizing RPDL to access the hardware. Review the file HWSetup.c to see how the YRDK hardware is initialized. Only basic setup is performed such as setting the direction of the I/O pins.

3.2 ADC_One_Shot

This sample demonstrates use of the ADC in single conversion mode. The program takes a sample reading of the voltage created by the potentiometer VR1 and displays it on the LCD.

3.2.1 Operation

1. Build and download the sample code to the YRDK.
2. Click 'Reset Go' to start the software.
3. Instructions are displayed on the LCD. Adjust the potentiometer VR1, and press switch SW3 to trigger an AD conversion.
4. The ADC result will be displayed on the second line of the LCD in the format =H'0000.
5. Re-adjust VR1, and press SW3 again to perform another conversion.
6. The value of the AD conversion can be watched in the variable usADC_Result. Right click on the variable, select “Instant Watch” and click “yes” to the following dialogs.

3.3 ADC_Repeat

This sample demonstrates use of the ADC in continuous scan mode. After the ADC is started, a timer is triggered to periodically fetch the result from the conversion. The potentiometer VR1 is connected to the ADC; adjust it and watch the results on the LCD.

3.3.1 Operation

1. Build and download the sample code to the YRDK.
2. Click 'Reset Go' to start the software.
3. The LCD will show the name of the program and the current ADC value.
4. Adjust the potentiometer, VR1, and observe the value displayed on the LCD change.
5. The user may examine the AD conversion result in the global array usADC_Result[ ].

3.4 CAN

This CAN sample demonstrates receive and transmit using the CAN API.

The demo can be run in three ways:

1. Program two boards and connect them together over the CAN bus.
2. With CANPORT_TEST_1_INT_LOOPBACK used in the R_CAN_PortSet API you can communicate internally, no external bus needed!
3. Use a CAN bus monitor, e.g. SysTec's low-cost monitor 3204000, to send and receive frames to/from the demo. Remote frames can also be demonstrated if CAN interrupts are enabled.

3.4.1 Operation

The demo transmits and receives frames with CAN-ID 1 by default. The default demo CAN-ID value is set in api_demo.h by TX_ID_DEFAULT. The software starts up by immediately sending ten test frames. This has two purposes, to check the link and to demonstrate how messages are sent back-to-back as quickly as possible.

Press SW1 to send one CAN frame. The frame will be received and displayed by the other RDK as long as that board's receive ID (RxID) matches the sending boards transmit ID (TxID).

Press SW2 to display the current demo TxID on the LCD. To increment the TxID, hold SW2 down and press SW3. The actual send command is invoked by the Sw1Func function.

Press SW3 to display current demo RxID on the LCD. To change RxID hold SW3 down and press SW2.

By default, polled CAN is used. To use the CAN interrupts for faster processing, uncomment USE_CAN_POLL in file config_r_can_rapi.h.

REMOTE frames:

Besides demonstrating Tx and Rx of Standard CAN frames, the demo will also send remote frame responses for CAN-ID 50h remote frame requests. Remote frames demo is only done in interrupt mode; with USE_CAN_POLL commented in the CAN API configuration file. Remote requests are not sent by this demo as it is, and so must come from an outside source, e.g. the CAN monitor mentioned above.

3.5 CMT_Compare

This sample demonstrates the use of a CMT configured to generate an interrupt every 100 milliseconds. The interrupt triggers a callback function to toggle the user LEDs.

3.5.1 Operation

1. Build and download the sample code to the YRDK.
2. Click 'Reset Go' to start the software.
3. Observe the user LEDs, which will flash every time a compare match occurs in the CMT channel.

3.6 CRC

This sample demonstrates the CRC unit by echoing characters typed in the terminal with their checksums.

3.6.1 Operation

1. Build and download the sample code to the YRDK.
2. Connect the YRDK to the host PC via an RS232 serial cable.
3. Launch a suitable terminal program (e.g. HyperTerminal) and configure it to operate on the channel the YRDK is connected to with the following settings:

   Baud.....19200
   Data Bits.....8
   Parity.....None
   Stop Bits.....1
   Flow Control.....None

4. Click 'Reset Go' to start the software.
5. Observe the terminal window - instructions will be displayed at the top, asking the user to input a character.
6. Type any letter (printable characters only) into the terminal, and the character will be echoed back to the terminal display, along with its corresponding CRC checksum.

3.7 Dhrystone

This project runs the Dhrystone benchmark on the YRDK highlighting the 1.65 DMIPS/MHz performance of the RX.

3.7.1 Operation

1. Build and download the program to the YRDK.
2. Click “Reset Go” to start the benchmark.
3. When the LED’s change and the display indicates, the Dhrystone is complete. Break the program by clicking the “Stop” button on the toolbar and examine the value of the variables “DMIPS_per_MHz” and “Dhrystones_Per_Second” to see the results.
4. If you uncomment the calls to “S12ADC_Init” and “S12ADC_Start” at the beginning of the main function, the RX will use the DMAC to take ADC samples at 400 kHz while the Dhrystone is running. The samples are saved to two ping-pong buffers of 5K samples each. Compare the Dhrystone results with and without the ADC running. This is an interesting experiment that shows how the advanced bus architecture of the RX plus its smart peripherals work together to offload the CPU.

3.8 DMAC

This sample demonstrates the DMAC by performing a DMA transfer from one variable and duplicating it into a 1024 element array.

3.8.1 Operation

1. Build and download the sample code to the YRDK.
2. Right-click the variable 'gDMA_DataBuff' and select 'Instant Watch', clicking 'Add' on the subsequent dialog.
3. The contents of 'gDMA_DataBuff can now be observed in the variable watch window.
4. Click 'Reset Go' to start the software.
5. Observe the contents of 'gDMA_DataBuff' being filled with the contents of the variable 'gDMA_DataSource'.

Note: In order to watch the contents of the gDMA_DataBuff array during execution, the sample must be built in 'debug' mode.

3.9 DTC

This sample demonstrates the DTC; by utilizing the DTC to transfer an ADC conversion to a 16 element array.

3.9.1 Operation

1. Build and download the sample code to the YRDK.
2. Right-click the variable 'gDTC_Destination' and select ‘Instant Watch', clicking 'Add' on the subsequent dialog.
3. The contents of the variable 'gDTC_Destination' can be viewed in the watch window.
4. Click 'Reset Go' to start the software. Instructions will be displayed on the debug LCD.
5. Adjust the potentiometer, RV1, and press the switch SW3. The result from the ADC conversion will be transferred into the first element of the transfer destination array.
6. Repress the switch SW3, and subsequent results will be transferred into the next array element.
7. Once the last array has been filled, the next transfer will clear the array contents and start from the first element.

Note: In order to watch the contents of the gDTC_Destination array during execution, the sample must be built in 'debug' mode.

3.10 Ethernet_uIP

This sample shows the open-source uIP Ethernet stack ported to the YRDK.

3.10.1 Operation

1. Build and download the sample code to the YRDK.

2. Connect the YRDK to your network (be sure there is a DHCP server accessible from the network).
3. Click “Reset Go” so start the software.
4. The YRDK will look for a DHCP server to obtain an IP address. Once an address is obtained, it is displayed on the LCD.
5. Type the IP address into the address bar of your web browser to contact the web server running on the YRDK.

3.11 Flash_Data

This sample demonstrates the FCU by writing ADC results to incrementing data flash memory locations.

3.11.1 Operation

1. Build and download the sample code to the YRDK.
2. Open the memory viewing, right click in the window and click 'address'. Enter the Display address 0x00100000, and then click OK.
3. Ensure the 'Auto Refresh' option in the memory viewer is disabled before proceeding. (It is not possible to use this feature when observing flash memory).
4. Click 'Reset Go' to start the sample. Instructions will be displayed on the debug LCD.
5. Push the switch SW1 to trigger an ADC conversion. The value will be written into the data flash memory (click refresh in the memory viewer).
6. The data written to flash and its memory location are displayed on the debug LCD.
7. Repress switch SW1 to trigger another ADC conversion. The result will be stored in an incremented memory location, displayed on the debug LCD.
8. Press switch SW3 to reset the contents of the entire flash memory block, and set the flash write pointer back to the start of the block (0x00100000).

Note: Memory locations which have not initialized do not have a defined state, and will continuously change with random values.

3.12 FPU_Bouncing_Ball

Demonstrates hardware Floating Point Unit (FPU) versus software floating point performance.

3.12.1 Operation

1. Build and download the sample code to the YRDK.
2. Click 'Reset Go' to start the software. Two small bouncing balls will appear on the LCD. The top ball is being moved using the FPU. The lower ball’s position is updated using the floating point emulation library instead of the FPU hardware. Each ball is given an equivalent time slice to perform as many position calculations as possible in the allotted time. It takes thousands of these calculations to move the ball one pixel on the display.

3.13 IIC_Master

This sample demonstrates use of IIC in master mode by reading from 2 separate IIC devices:

- ADXL345 digital accelerometer
- ADT7420 thermal sensor

3.13.1 Operation

1. Build and download the sample code to the YRDK.
2. Click 'Reset Go' to start the software. The name of the sample will be displayed on the debug LCD, along with the current board temperature in Celsius.

3. Place a finger over the ADT720 device. It is located just below the APPLICATION HEADER located along the top edge of the board. Observe that the temperature is updated four times per second. When you remove your finger, the temperature returns to normal.
4. Pick up the board and tilt the board forward/backward and left/right.
5. As you slowly roll the board in a circular motion, observe the circle LEDs light to indicate the direction of the tilt.

3.14 MTU_Timer

This sample uses an MTU to generate a 1KHz square waveform.

3.14.1 Operation

1. Build and download the sample code to the YRDK.
2. Click 'Reset Go' to start the program.
3. The 1 KHz waveform can be observed with an oscilloscope connected to the header JN2, pin 8. (JN2 pin 2 can be used for a ground.)

3.15 Power_Down

This sample demonstrates the MCU power down modes by entering standby when a switch is pressed. The MCU then wakes up by any interrupt.

3.15.1 Operation

1. Build and download the sample code to the YRDK.
2. Click 'Reset Go' to start the software.
3. The user LEDs will start to flash and the current MCU power mode will be displayed on the debug LCD.
4. Press switch SW1 to enter the MCU into standby mode. The debug LCD will update with the current power mode and the LEDs will stop flashing.
5. Holding down another switch whilst releasing SW1 will prevent the MCU from entering standby until all switches are released. If another switch is held down whilst SW1 is released, the red LEDs will be lit to indicate that the MCU cannot enter standby. Once all switches are released, the LEDs will turn off.
6. Press any switch to wake the MCU from standby mode. The debug LCD will display the new power mode, and the LEDs will stay constantly lit.
7. To repeat the sample, restart the software by clicking 'Reset Go'.

3.16 SCI_Async

This sample demonstrates the SCI in asynchronous mode by transmitting data to a connected terminal.

3.16.1 Operation

1. Build and download the sample code to the YRDK.
2. Connect the YRDK to the host PC via an RS232 serial cable.
3. Launch a suitable terminal program (e.g. HyperTerminal) and configure it to operate on the channel the YRDK is connected to with the following settings:

Baud.....19200
Data Bits.....8
Parity.....None
Stop Bits.....1
Flow Control.....None

4. Click 'Reset Go' to start the software.

5. Observe the terminal window - instructions will be displayed at the top, then the numbers 0 - 9 will be written across the screen repeatedly.
6. Type 'z' (lowercase) into the terminal to pause the SCI transmission, and any other key to resume it.

3.17 SPI_Flash

Demonstrates use of Renesas Peripheral Development Library (RPDL) to write to/read from the RSPI flash device. This program writes 2048 bytes of data to the flash device located on RSPI-0, 64 bytes at a time. It then proceeds to read the data back 512 bytes at a time and compares it to the original data. Upon successful completion, LED 4 is turned on. However, if the programming fails, LED 14 is turned on.

3.17.1 Operation

Build and download the sample code to the YRDK.

3.18 Timer_Capture

Demonstrates capture a timer value from the Compare Match Timer (CMT)

3.18.1 Operation

1. Build and download the sample code to the YRDK.
2. Click 'Reset Go' to start the software. Instructions will be displayed on the debug LCD.
3. Press switch SW1, and observe the result on the debug LCD. The time duration, in milliseconds (ms), will be displayed.
4. Press switch SW1 again to repeat the test.

3.19 TMR_Oneshot

This sample demonstrates use of the TMR unit, by creating one timer to restore the debug LCD.

3.19.1 Operation

1. Build and download the sample code to the YRDK.
2. Click 'Reset Go' to start the software.
3. The debug LCD will show:

   "TMR 1 Shot "
   "Press SW3 "

4. Press switch 3 and the debug LCD will display a new message and the one shot timer is started.
5. When the one shot timer fires, the debut LCD is restored to the original display.

3.20 Tutorial

This tutorial sample will demonstrate basic use of the YRDK hardware and the J-Link debugger.

3.20.1 Operation

1. Build and download the tutorial project to the YRDK.
2. Click 'Reset Go' to start the software.
3. "Renesas RX62N" will be displayed on the debug LCD, and the user LEDs will flash.
4. The user LEDs will flash at a fixed rate until either a switch is pressed, or the LEDs have flashed 200 times.
5. The software will then vary the rate in which the LEDs flash by the position of the potentiometer, VR1. Turn the potentiometer in both directions, and observe the change in flash rate.

6. While the LEDs flash at a varying rate, the second line of the debug LCD will show “STATIC". The second LCD line will slowly be replaced, letter by letter, with the string "TESTTEST".
7. Once the second line of the debug LCD shows "TESTTEST" fully, it will return back to showing "RX62N". The LEDs will continue to flash at a varying rate until the tutorial is stopped.
8. In order to repeat the tutorial, click 'Reset Go'.

3.21 USB_CDC

This Application demonstrates the USB function controller. The code implements the CDC class. This means the device will appear as a virtual COM port to the host.

3.21.1 Operation

1. Build and download the project to the YRDK.
2. Connect a second USB cable to the USB jack below the Ethernet jack. Connect the other end to your PC.
3. Click “Reset Go” to start the software.
4. When Windows displays the “Found New Hardware” dialogs, point it to the INF file in the project directory. The board will then enumerate as a virtual COM port on the PC.
5. Open a terminal program and select the virtual COM port for the board. Follow the instruction that are displayed.

3.22 USB_HID

A USB HID Application. This USB HID application allows a host to perform the following operations by sending an OUTPUT report:

1. Toggle a LED.
2. Read the ADC.
3. Write to the LCD.

The Device will tell the host, via an INPUT report, when a user has pressed a switch.

3.22.1 Operation

1. Build and download the program
2. Click “Reset & Go” on the tool bar.
3. Connect a second USB cable to the jack just below the Ethernet connector; connect the other end to your PC.
4. In the “Host” folder of the project (you can see it in the Project pane), run the program “RSK_HID.EXE” by double clicking on it. Click “Connect”, accept the defaults, and then use the buttons in the program to turn an LED on and off, read the ADC, and update the LCD via the USB HID connection to the board.

3.23 WDT

This sample demonstrates use of the watchdog timer (WDT) by configuring the timer to require periodic resets to prevent a timer underflow. The sample configures a periodic timer reset which has a period that varies with the position of the potentiometer, VR1. The LEDs flash to indicate each period.

3.23.1 Operation

1. Build and download the sample code to the YRDK.
2. Ensure the potentiometer, VR1, is turned to the fully counter-clockwise position before starting the sample. To start the sample, click 'Reset Go'.
3. The debug LCD will display the sample name, and the user LEDs will begin to flash.
4. Slowly turn the potentiometer VR1 clockwise, and observe the gradual decrease the LED flash rate. Each time the LEDs toggle output state, the watchdog timer is reset, preventing it from triggering an overflow interrupt.

5. Continue to turn the potentiometer clockwise - the LED flashing will slow down to a point where the watchdog timer is allowed to overflow.
6. When the timer has overflowed, the LEDs will stay constantly lit, and the program will wait in an infinite while loop. The debug LCD will display "Watchdog Overflow".
7. To repeat the sample, ensure VR1 is fully counter-clockwise and click 'Reset Go'.

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/inquiry

YRDKRX

Revision Record

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Dec.02.2010	—	First edition issued
1.10	Dec.06.2010	—	Minor editorial corrections.

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins
   Handle unused pins in accord 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 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. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on
   The state of the product is undefined at the moment 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 moment 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 moment 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 moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has 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. Moreover, 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.
5. Differences between Products
   Before changing from one product to another, i.e. to one with a different type number, confirm that the change will not lead to problems.
   — The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of 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. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. 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 of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may 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.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
   - "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
   - "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
   - "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, 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, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be 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, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

RENESAS

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
7F, No. 363 Fu Shing North Road Taipei, Taiwan, R.O.C.
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141