RX600 Series

Adding Printf and Scanf Support

R01AN0509EU0100
Rev.1.00
Feb 1, 2011

Introduction

This document explains how to add printf/scanf support to your HEW project. Examples are shown where standard input and output are directed to a simulated console, debug console, and a standard serial port.

Target Device

Renesas RX600 Series

Contents

1. Overview

Being able to write to and read from a console can be an integral part of your program and development process. Perhaps you need printf to display run time information; or scanf to be able to collect configuration parameters. Additionally, depending on your hardware architecture or what phase of development you are in, you may not have physical access to a serial port to run your terminal emulation.

The purpose of this application note and the attached *RX_printf* project is to provide examples of how to get the `printf()` and `scanf()` functions in your development environment directed to various consoles. We will show you how your serial I/O can be targeted to the HEW simulator console in a simulation environment, the HEW debug console, or to your hardware's serial port in a hardware debug environment.

The common denominator no matter what your targeted console, are the low-level functions of `charput()` and `charget()`. The *RX_printf* project provides separate build configurations to demonstrate these three scenarios. Each configuration brings in a different low-level source file to provide these functions for directing the I/O to the appropriate consoles:

Serial I/O Direction	Source Code	Build Configuration
HEW Simulator Console	lowlvl_Sim.src	Sim_Output
HEW Debug Console	lowlvl_DebugConsole.src	DebugConsole_Output
RSK+RX62N RS-232 port	serial_printf.c	Serial_Output

NOTE: The project *RX_printf* associated with this application note is intended to run on the RSK+RX62N reference platform.

2. Directing Standard I/O to the Simulator Console

First, let's take a look at directing the serial I/O to the HEW Simulator Console. As mentioned earlier, the I/O stream direction is controlled in the low-level software. In this example, we're going to use *lowlvl_sim.src*.

2.1 lowlvl_sim.src

The functions in this source code are generated when you create a new project in HEW. It shows up in your project directory as *lowlvl.src*. For this demonstration, it's been renamed so that its intended use is easily identifiable.

There are no modifications required to this code. The `charput()` and `charget()` functions interact with HEW to pass characters between your program and HEW running on your workstation.

2.2 Building the Sample Project

Copy the *RX_printf* project to your workspace directory. Open the workspace in HEW and select "Sim_Output" from the Build Configuration pull down and "SimSessionRX600" from the Debug Session pull down.

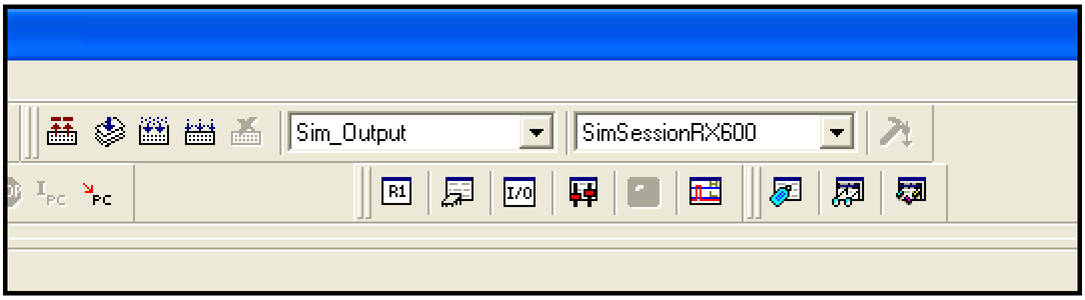


Figure 2-1 : Choosing build & debug configurations

Use the defaults in the "Set Simulator" pop up as shown in Figure 2-2.

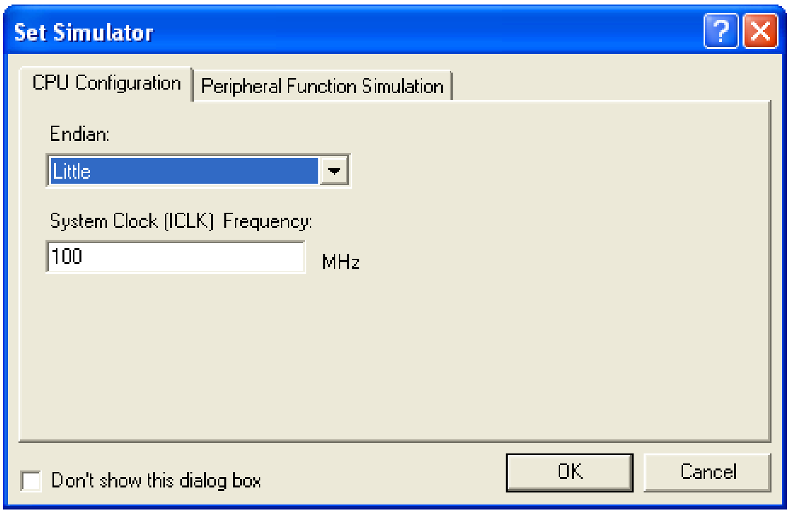


Figure 2-2 : Configuring the RX Simulator

Notice in the Project Navigation pane shown in Figure 2-3 that *lowlvl_Sim.src* is included in the build and the other two files are excluded. Which printf/scanf files are included and excluded is controlled by the currently selected Build Configuration.

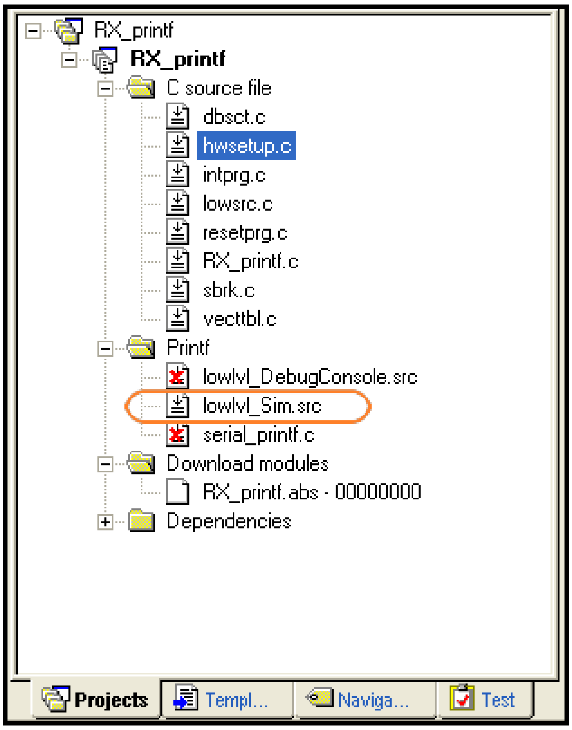
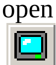


Figure 2-3 : Project Navigation Pane

Build the project by either pressing F7 or going to Build >> Build All. After the project has been built 'download' the module to the RX simulator by double clicking on 'RX_printf.abs'. If not already opened, open the Simulated I/O window by going to View >> CPU >> Simulated I/O or by clicking the Simulated I/O icon  in HEW.

Run the code by going to Debug >> Reset Go. Notice that the `printf()` and `scanf()` in `main()` are directed to/from the HEW simulator console as shown in Figure 2-4. When asked to enter an integer value, input a value and then hit Enter on your keyboard.

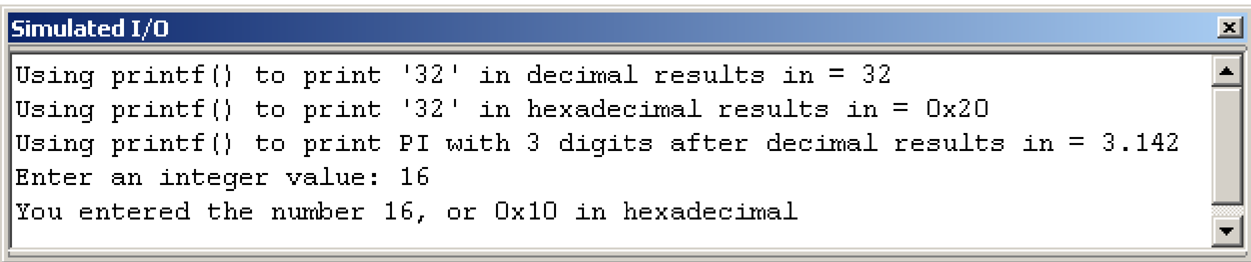


Figure 2-4 : Simulator Console Output

3. Directing Standard I/O to the HEW Debug Console

The next example demonstrates using standard I/O through the HEW debug console. This example's `charput()` and `charget()` functions come from *lowlvl_DebugConsole.src*.

3.1 lowlvl_DebugConsole.src

The low-level interface routines `charput()` and `charget()` in *lowlvl_DebugConsole.src* come directly from the example code provided in the E1/E20 Supplementary Guide for the RX62N. They do not require any modification to work in your development environment. For details on how these functions work consult "Using the Debug Console" in the above document.

3.2 Building the Sample Project

Select "DebugConsole_Output" from the Build Configuration pull down and "SessionRX600_E1_E20_SYSTEM" from the Debug Session pull down.

Figure 3-1 : Debug Console Project Configuration

Notice that `low_lvl_Debug_Console.src` is now included in the build:

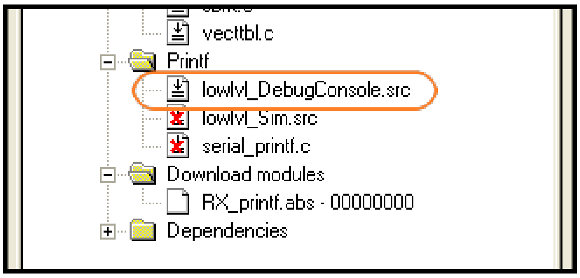


Figure 3-2 : Debug Console file included

Build and load the project and open the HEW Debug Console by clicking on the same button that you used to open the Simulator Console in the previous example. Run the code and notice that the `printf()` and `scanf()` in `main()` are directed to/from the HEW debug console. The Debug Console should look like Figure 2-4.

4. Directing Standard I/O to the Platform Serial Port

The final example demonstrates using standard I/O via an RS-232 Serial Port. This scenario requires knowledge of the hardware and custom software to make it work. The RX Serial Communication Interface (SCI) on the RSK reference board is initialized by `InitSCI()` in *hwsetup.c*. The "get" and "put" functions are provided by *serial_printf.c*.

4.1 serial_printf.c

This source code in this file is written specifically for the RSK+RX62N hardware. It does polling on a RX SCI for simple serial communications.

It's important to note that your implementation may require a more sophisticated solution involving semaphores, interrupt handlers or other various mechanisms typically found in RTOS-like environments. Additionally, these functions may have to be tailored to interface your custom hardware.

4.2 Building the Sample Project

Select "Serial_Output" and "SessionRX600_E1_E20_SYSTEM" on the Build Configuration and Debug Session pull downs.

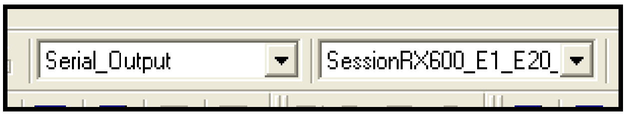


Figure 4-1 : Serial Output Project Configuration

The file *serial_printf.c* should now be included in the build. Both this file and *hwsetup.c* are platform dependent files that have been modified to work on the RSK+RX62N reference platform.

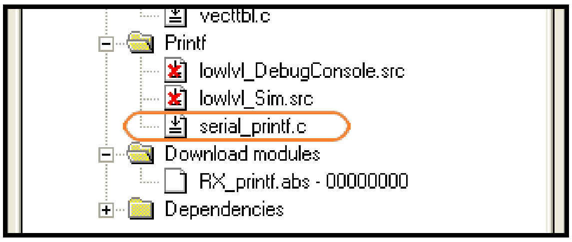


Figure 4-2 : serial_printf.c included

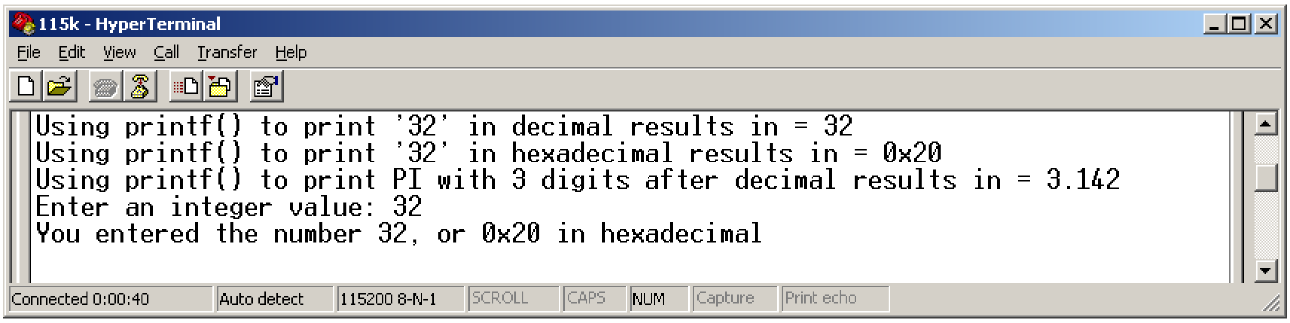
Next, connect a serial cable between the RSK board and a workstation and start a terminal emulator. Configure the workstation with the following parameters:

- 115200 Bits per second
- 8 Data bits
- No Parity
- 1 Stop bit
- No Flow Control

NOTE: The RSK+RX62N reference platform has jumpers that need to be set to utilize the serial port:

- J6 & J15 must be set in the TxD2-A position
- J5 & J16 must be set in the RxD2-A position

Build and load the project. Run the code and notice that now the `charget()` and `charput()` in `main()` are directed to/from the terminal emulator.



5. Adding Support to Your Own Project

When incorporating printf/scanf functionality to your own project, there are a couple of modifications that you need to be sure to include in your project configuration.

5.1 Creating a New Project

If creating a new project, as you step through the setup windows be sure to check the “Use I/O Library” option as shown in Figure 5-1.

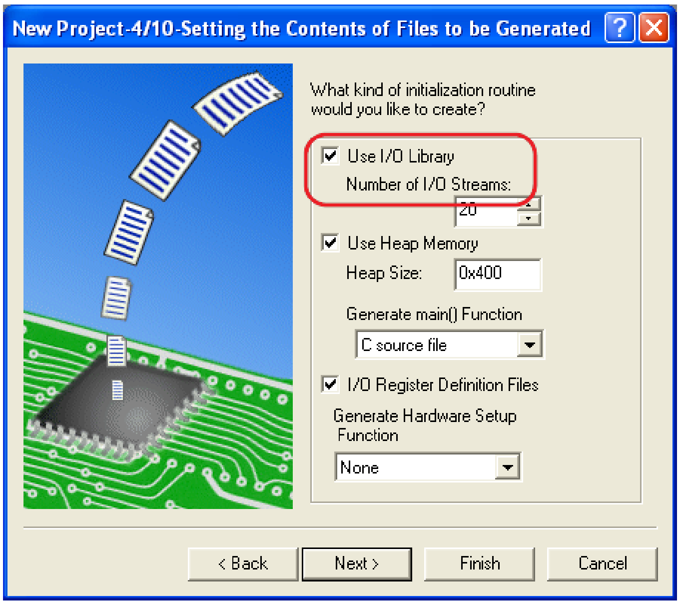


Figure 5-1 : Enable use of I/O Library

Also, on the next panel, make sure that *stdio.h* is selected.

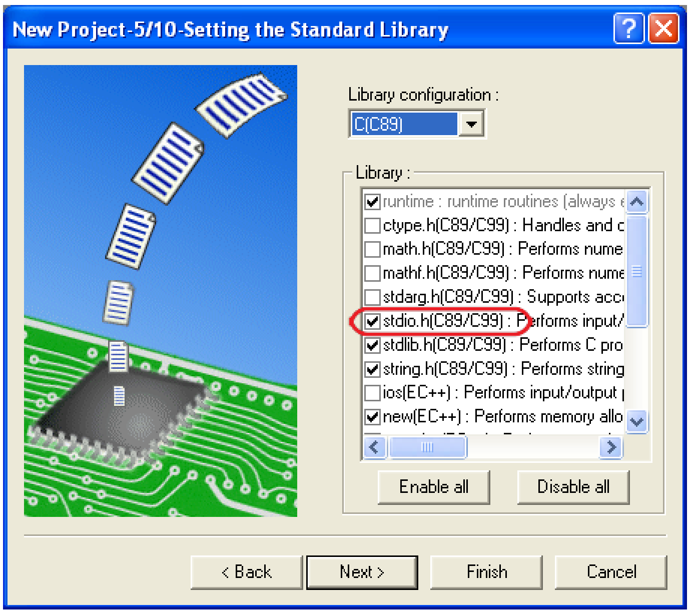
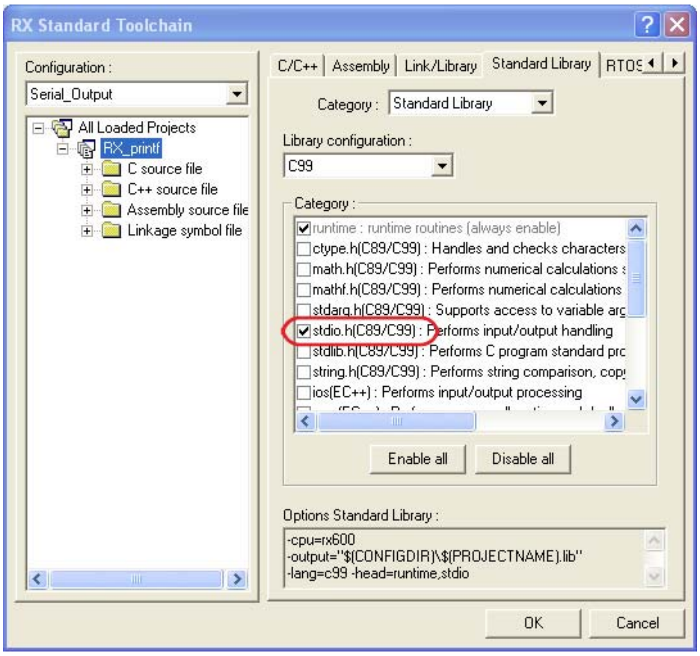


Figure 5-2 : Enable use of stdio.h

5.2 Updating an Existing Project

Likewise, if updating an existing project, *stdio.h* needs to be included in the build. This can be done by selecting Build >> RX Standard Toolchain from the HEW toolbar, and then clicking on the “Standard Library” tab of the presented configuration window.

Select “Standard Library” in the “Category” pull down. Verify that *stdio.h* `is` checked:



Be sure to integrate the appropriate `charget()` and `charget()` functions to achieve the desired serial I/O functionality.

Website and Support

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

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

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Feb.01.11	—	First edition issued

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 document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

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 a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the 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.

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