RX600 Series

Using RPDL & PDG

R01AN0615EU0100
Rev.1.00
Mar 24, 2011

Introduction

The Renesas Peripheral Driver Library (RPDL) is a unified API for configuring and controlling the peripheral modules on Renesas microcontrollers. Examples of peripherals supported are timers, watchdogs, DMA, DTC, SPI, I2C, ADC, and DAC. As shown in the figure below RPDL is a software abstraction layer that sits between the user's application and the Target MCU. RPDL takes care of reading and writing all of the supported peripheral registers on the Target MCU. While RPDL covers most peripherals, more complex peripherals like USB, Ethernet, and CAN are not supported and require their own drivers. These drivers are free to use RPDL.

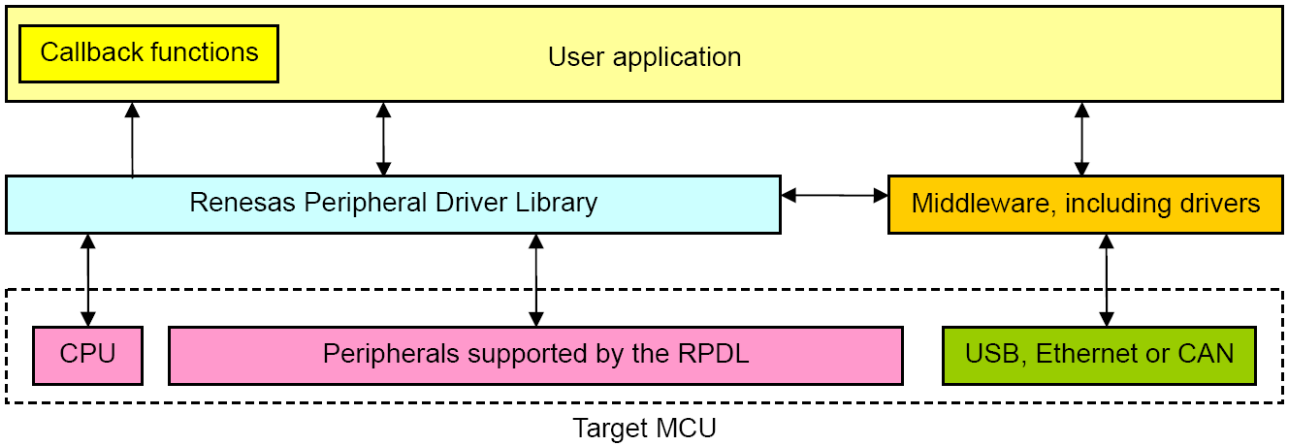


Figure: MCU with Software Layers

Peripheral Driver Generator (PDG) is a graphical tool that runs on a PC and produces RPDL function calls. PDG offers users the ability to configure their target device's peripherals through an easy to use, graphical interface. Once the user has configured their peripherals in PDG they can generate the code and PDG will automatically incorporate these files into their HEW project.

This application note discusses the usage of the RPDL and the added value provided by the PDG. This document first takes you through an example of manually implementing support for a peripheral in the HEW using RPDL. After this it will demonstrate the simplicity of generating the same support via PDG.

Target Device

YRDKRX62N

Contents

1. Required Tools 2
2. Creating an RPDL Workspace 3
3. Using Renesas Peripheral Driver Library (RPDL) 8
4. Using Peripheral Driver Generator (PDG) 13

1. Required Tools

This application note is written for the YRDKRX62N platform but the basic steps of using RPDL and PDG can be applied to any project.

Before running through the steps in this lab the user should have installed the following:

- The RX62N RDK DVD
  - http://am.renesas.com/products/tools/introductory_evaluation_tools/renesas_demo_kits/yrdkrx62n/child_folder/downloads_child.jsp
- Peripheral Driver Generator v2
  - http://www.renesas.com/pdg_download

The rest of the code, including RPDL, that will be referenced in this document can be found in the 'Source' directory that is packaged with this application note.

2. Creating an RPDL Workspace

This section will cover setting up a workspace to use with RPDL.

Procedural Steps:

Step 2.1 Start HEW by going to Start >> All Programs >> Renesas >> High Performance Embedded Workshop >> High Performance Embedded Workshop.

Step 2.2 In the Welcome window select “Create a new project workspace” and click “OK”.

Step 2.3 Choose “RX” from the “CPU family” drop down.

Step 2.4 Choose “Renesas RX Standard” from the “Tool chain” drop down.

Step 2.5 Choose “Application” from the entries under “Project Types”.

Step 2.6 Type “RX_RPDL_Demo” into the “Workspace Name” field. Your window should look like Figure 2-1. Click “OK”.

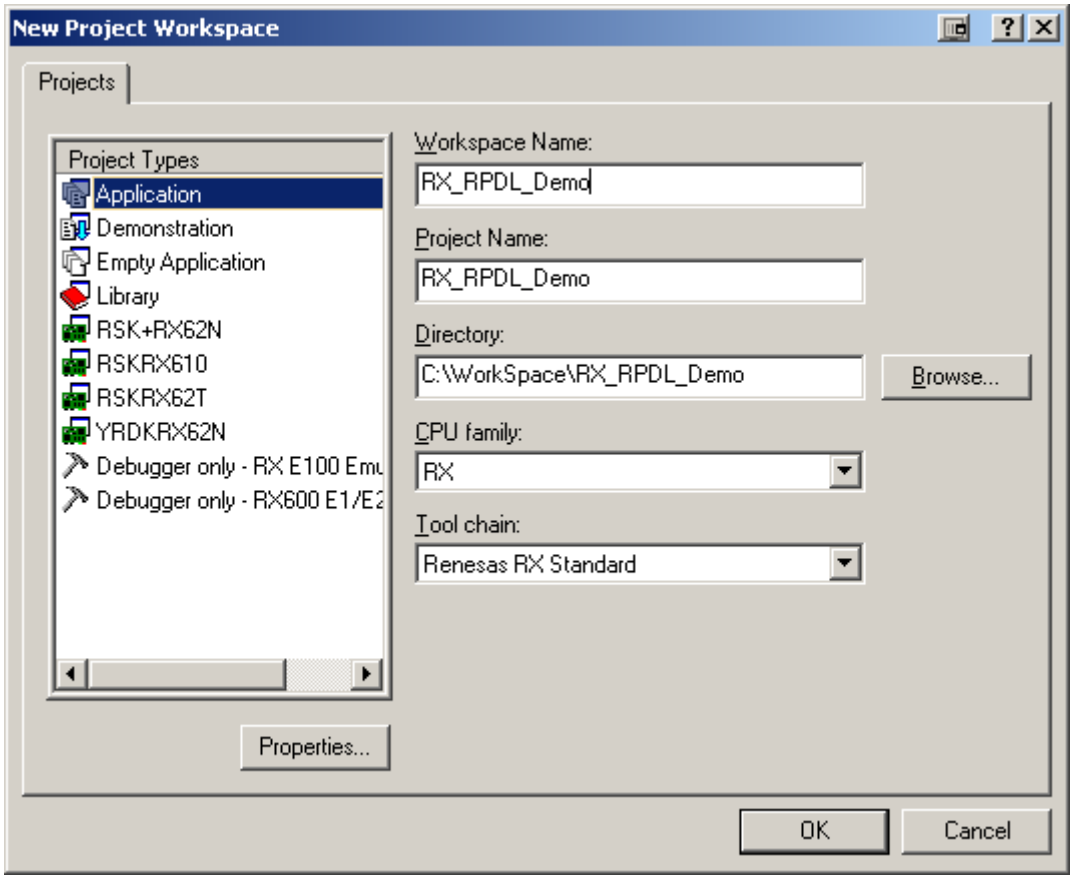


Figure 2-1 : Creating a new workspace

Step 2.7 In the window choose “RX62N” from the list underneath “CPU Type”. Click “Next”.

Step 2.8 Continue to hit “Next” until you get to the window that says “New Project-3/10…” in the title bar. In this window select “None” from the “Changes code generation” drop down box.

Step 2.9 Continue to hit “Next” until you get to a window that says “New Project-4/10…” in the title bar and looks like Figure 2-2. Uncheck the box next to “I/O Register Definition Files”. Click “Next”.

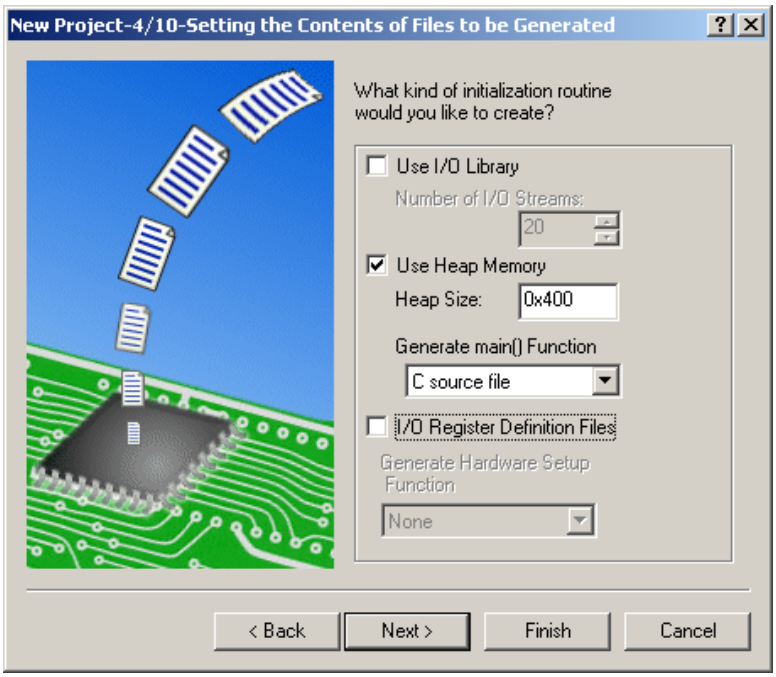


Figure 2-2 : New Project Window 4

Step 2.10 Continue to hit “Next” until you get to a window that says “New Project-8/10…” in the title bar and looks like Figure 2-3. Check the box next to “RX600 Segger J-Link” and click “Finish”.

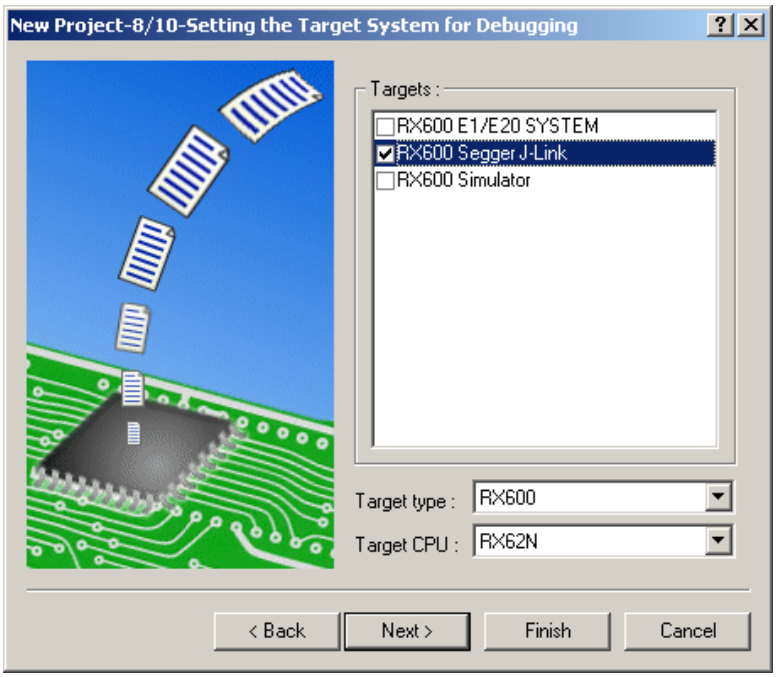


Figure 2-3 : Select Debugger

Step 2.11 Click “OK” in the Summary window.

Step 2.12 Now, navigate via an Explorer window to the ‘Source/RPDL_RX62N’ folder underneath the folder where you extracted this application note’s contents.

Step 2.13 Double-click on the file Copy_RPDL_RX62N.bat. A command window will open.

Step 2.14 When asked to select the device use the number corresponding to the 100 pin LQFP part and press “Enter”.

Step 2.15 When asked for where to install RPDL enter “C:\Workspace\RX_RPDL_Demo\RX_RPDL_Demo” and press “Enter”. The RPDL files will be copied and when done the window should look like Figure 2-4. Press any key to close the command window.

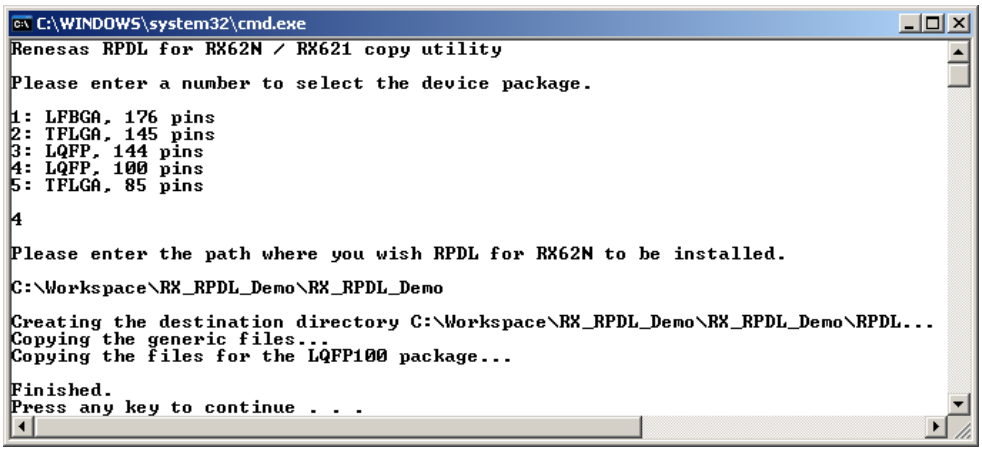


Figure 2-4 : Copying RPDL to your workspace

This batch file simply copies the appropriate RPDL library and required C source and header files to your workspace. You can copy the files manually if needed.

Step 2.16 Browse to the directory “C:\Workspace\RX_RPDL_Demo\RX_RPDL_Demo” and verify that there is a folder named “RPDL”. If not, then repeat Step 2.15 and make sure to type the correct installation path.

Step 2.17 Go back to your HEW workspace and click Build >> RX Standard Toolchain.

Step 2.18 Select the “C/C++” tab and change the “Show entries for” drop down to “Include file directories”.

Step 2.19 Click the “Add” button.

Step 2.20 In the window that comes up change the “Relative to” drop down to “Project directory” and for the “Sub-Directory” entry use “RPDL”. Your window should look like Figure 2-5. Click “OK”.

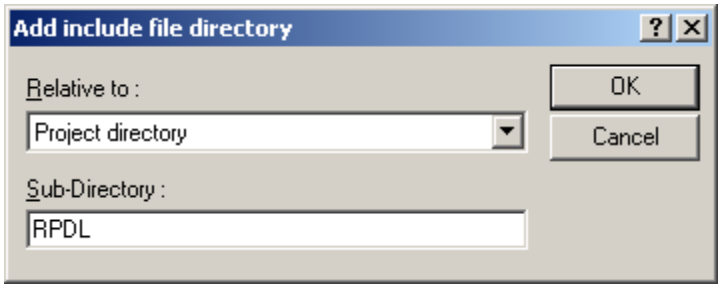


Figure 2-5 : Include the RPDL folder

Step 2.21 Follow the same directions to add another directory and this time set the "Sub-Directory" entry to "." (one "period" means "current directory" which will be the project directory).

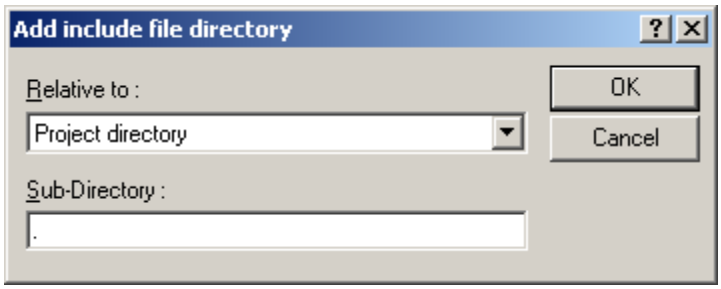


Figure 2-6 : Include the project's root folder

Step 2.22 Click on the "Link/Library" tab at the top of the window.

Step 2.23 Verify that "Input" is chosen from the "Category" drop down and that "Library files" is chosen for "Show entries for".

Step 2.24 Click the "Add" button.

Step 2.25 Change the "Relative to" drop down to "Project directory".

Step 2.26 For the "File path" entry enter "RPDL\RX62N_library.lib" and click "OK".

Step 2.27 Click "OK" to exit the RX Standard Toolchain window.

Step 2.28 Click Project >> Add Files. In the window that comes up select "C source file" from the "Files of type" drop down.

Step 2.29 Navigate to the "RPDL" folder underneath the project folder and highlight all the C source files (*.c) except *Interrupt_EXDMAC.c*. *Interrupt_EXDMAC.c* was not included because the 100-pin and 85-pin RX62N products do not support the EXDMA controller. After you have highlighted the files click "Add".

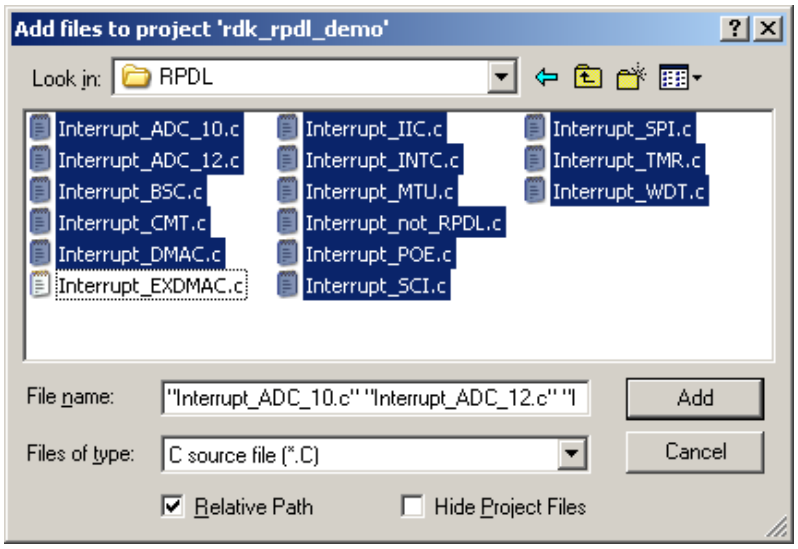


Figure 2-7 : Adding RPDL files to project

After adding the RPDL C source files, your project pane can look unwieldy. To help organize the files, you can make a folder to hold them all. Right click on the “C source file” folder and select “Add Folder”. Input “RPDL” for the folder name and click ‘OK’. You can then drag and drop the RPDL files into the new folder. The ‘RPDL’ files are easy to spot since they all start with “Interrupt_”. When done, the project pane will look like the following:

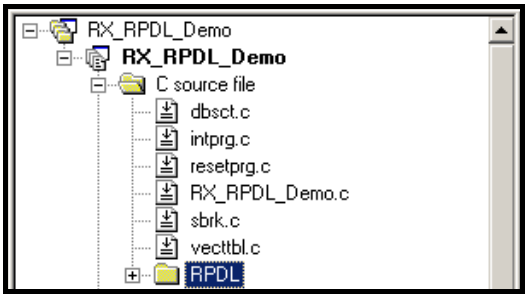


Figure 2-8 : Creating an RPDL folder

Step 2.30 RPDL handles device interrupts and therefore will conflict with the *intprg.c* and *vecttbl.c* files that are automatically generated with the RX “Application” project. Exclude both of these files by highlighting them in the project navigation pane and right clicking and selecting “Exclude Build …”.

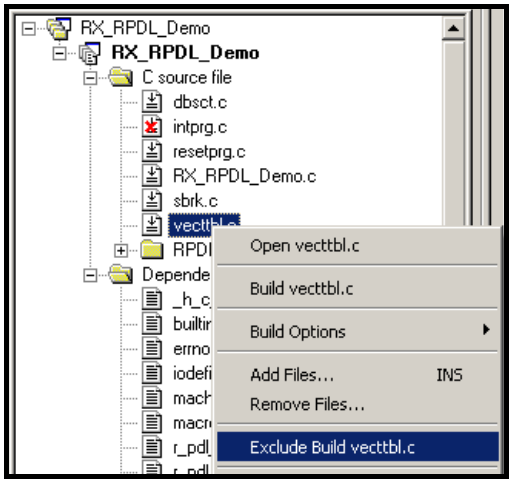


Figure 2-9 : Excluding vector handling files

Step 2.31 RPDL has now been added to your project and you can build it. Press the Build All button  or go to Build >> Build All.

When building the project you will get a warning stating ‘L1100 (W) Cannot find "PIntPRG" specified in option "start"’. This is a result of excluding the ‘intprg.c’ file earlier. You can safely ignore this warning since RPDL is taking care of the interrupts, or you can remove the warning by going into the toolchain menu and removing the ‘PIntPRG’ section name. You can see the sections for your project by going to Build >> RX Standard Toolchain, switch to the “Link/Library” tab, and change “Category” to “Section”.

3. Using Renesas Peripheral Driver Library (RPDL)

This section of the lab will focus on using RPDL. The user will create a timer to trigger the ADC. After the ADC conversion has finished it will be displayed on the LCD.

Procedural Steps:

Step 3.1 Go to the "Source" directory underneath the folder where you extracted this application note's contents.

Step 3.2 Copy the following files from the "Source" directory to your HEW workspace project directory ("C:\Workspace\RX_RPDL_Demo\RX_RPDL_Demo").

- *glyph_api.h*
- *Glyphlib_v2.lib*
- *lcd.c*
- *lcd.h*
- *lcd_utilities.c*
- *lcd_utilities.h*
- *YRDKRX62N.h*
- *YRDKRX62N_RSPI_API.c*
- *YRDKRX62N_RSPI_API.h*

Step 3.3 Add the files *lcd.c*, *lcd_utilities.c*, *YRDKRX62N_RSPI_API.c*, and *Glyphlib_v2.lib* to your current project by going to Project >> Add Files as was done in Section 2.

Step 3.4 Open up the file *RX_RPDL_Demo.c* in HEW.

We have included the RPDL library in our project but we also need to include the appropriate header files in order to use the RPDL functions and definitions. In every RPDL source file that you use there will be 2 different types of header files that you use:

1. **r_pdl_definitions.h**
   a. This file contains device specific definitions
2. **r_pdl_*PERIPHERAL*.h**
   a. These files contain driver function prototypes
   b. Examples: *r_pdl_adc_10.h*, *r_pdl_cmt.h*, *r_pdl_tmr.h*, etc…

The headers that declare the driver function prototypes are always included in the source file before the *r_pdl_definitions.h* file.

Step 3.5 We are going to use the 10-bit ADC and 8-bit Timer together in this lab. We also use the Clock Generation Circuit (CGC) to configure our system clocks. Include the appropriate header files at the top of *RX_RPDL_Demo.c*.

```
#include "r_pdl_tmr.h"
#include "r_pdl_adc_10.h"
#include "r_pdl_cgc.h"
```

Step 3.6 After these include statements, add an include for the file *r_pdl_definitions.h*.

```
#include "r_pdl_definitions.h"
```

Step 3.7 To get all the basic hardware support necessary to display the ADC value on the LCD, we will use the function UpdateLCD() found in *lcd_utilities.h*. The RSPI API library allows for shared access on the RSPI bus, which is where the LCD resides. Include these files after the previous include statements:

```
/* RSPI API library support */
#include "YRDKRX62N_RSPI_API.h"

#include "lcd_utilities.h"
#include "lcd.h"
```

Step 3.8 If you have not yet installed PDG as mentioned in Section 1, do so now.

Step 3.9 We are now ready to start using the RPDL API. Open up the RPDL User's Manual found under the PDG install directory here: 'C:\Renesas\PDG2\manuals\r20ut0084ee0103_RX62N.pdf'.

A convenient way to easily access the RPDL User's Manual is to add it to your HEW project. You do this the same way as you do when adding a source file. Just make sure that the "Files of type" drop down is set to "All Files" otherwise the file won't show up. After you add the file it will show up in the project navigation pane and you can double-click it to open it up in your default PDF reader.

Step 3.10 The first thing that needs to be done to use RPDL is to setup the system clocks. This is required because clock settings are needed when setting up other peripherals. In the RPDL User's Manual go to Section 4: Library Reference. Continue to the subsection 4.2.1 which is the Clock Generation Circuit reference.

Step 3.11 The first function listed is R_CGC_Set(). Take some time to read over the API and get a feel for how it is setup.

Step 3.12 Find the main() function in *RX_RPDL_Demo.c* and add a RPDL function call to set up the system clocks with these parameters:

- An input frequency of 12MHz
- System clock = 96MHz
- Peripheral module clock = 48MHz
- External bus clock = 24MHz
- Disable BCLK output

Step 3.13 According to the RPDL User's Manual this would be done using the line of code below:

```
/* Setup system clocks */
R_CGC_Set( 12E6, 96E6, 48E6, 24E6, PDL_CGC_BCLK_DISABLE );
```

Step 3.14 We can now proceed to setup the 10-bit ADC. Refer to section 4.2.25 of the RPDL User's Manual.

Step 3.15 After the GPIO initialization add a function call to R_ADC_10_Create( ) to setup the ADC with these values:

- Unit 1
- Channel AN4
- Single Mode
- TMR0 is the trigger
- Right alignment (LSB aligned)
- 48MHz conversion clock
- 0.6us for sampling time (0.5us is listed as minimum in HW manual)
- Call the function ADC_Callback() when conversion has finished
- Use an IPL of 4

Step 3.16 According to the RPDL User's Manual this would be done using the line of code below.

```
/* Setup 10bit ADC */
R_ADC_10_Create(1,
                PDL_ADC_10_CHANNELS_OPTION_1 |
                PDL_ADC_10_MODE_SINGLE |
                PDL_ADC_10_TRIGGER_TMR0_CM_A,
                48E6,
                6E-7,
                ADC_Callback,
                4 );
```

Step 3.17 We will now setup the 8-bit timer to trigger the ADC conversion. Refer to section 4.2.15 for available API functions. Keep going through the pages until you find the description for the R_TMR_CreatePeriodic() function.

Step 3.18 After the R_ADC_10_Create() function call add a function call to R_TMR_CreatePeriodic() to setup the ADC with these values:

- Unit 0 (takes TMR0 and TMR1 and cascades them to make a 16-bit counter)
- Set a frequency of 4Hz
- ADC triggering
- 50% duty cycle
- No callback functions
- 0 for IPL since there is no callback

Step 3.19 According to the RPDL User's Manual this would be done using the line of code below.

```
/* Setup TMR */
R_TMR_CreatePeriodic( PDL_TMR_UNIT0,
                      PDL_TMR_FREQUENCY | PDL_TMR_ADC_TRIGGER_ON,
                      4,
                      0.5,
                      PDL_NO_FUNC,
                      PDL_NO_FUNC,
                      0 );
```

Step 3.20 The last initialization code we will use is to setup the LCD. These functions are provided in the *lcd_utilities.c* file. After the function call to R_TMR_CreatePeriodic() add a function call to YRDKRX62N_RSPI_INIT() and InitializeLCD(). Also add an infinite loop after this call so execution will not leave the main() function.

```
/* Initialise the LCD display on RSPI bus */
YRDKRX62N_RSPI_Init( 0 );

/* Setup LCD */
InitialiseLCD();

/* Infinite loop */
while(1);
```

Step 3.21 Now that we have initialized our peripherals we just have to write the callback routines. In the *RX_RPDL_Demo.c* file create the function below.

```
/* Callback function when ADC conversion has finished */
void ADC_Callback(void)
{
    uint16_t ADC_value;

    /* Read ADC value */
    //Fill in this line

    /* Update ADC value on LCD */
    UpdateLCD(ADC_value);
}
```

Step 3.22 Refer to the 10-bit ADC section in the RPDL User's Manual and find the function R_ADC_10_Read(). Use this function to fill in the spot in the code where you need to read the ADC value. Notice that R_ADC_10_Read() takes in a pointer to where you wish to store the data. In the C programming language putting a '&' in front of a variable name is used to get the address of that variable. As you can see in the code the 'ADC_value' variable is already setup to hold the ADC reading. Refer to Step 3.15 for which ADC unit is being used.

Step 3.23 After adding the RPDL call, add a function prototype for the ADC_Callback() function to the top of *RX_RPDL_Demo.c* file underneath the #include's.

```
void ADC_Callback(void);
```

Step 3.24 Press the Build button  or use the shortcut F7.

Step 3.25 Connect the YRDKRX62N board to your workstation using the supplied USB cable. Make sure to use the USB header on the board labeled "J-Link USB".

Step 3.26 Change the session from "DefaultSession" to "SessionRX600_Segger_J-Link". If a window pops up asking you to save the current Debug Session click "Yes".

Figure 3-1 : Change debugging session

Step 3.27 If the connection window does not automatically come up then click the Connection button or go to Debug >> Connect.

Step 3.28 Configure your windows so they look like the ones shown in Figure 3-2 and hit OK until the connection has finished. If a window pops up asking you to specify a JTAG clock choose 16.5MHz. If a window pops up asking you to upgrade the firmware, select OK. When it has finished upgrading the firmware, unplug the USB cable from the YRDK and then plug it back in. At this point follow Step 3.27 to connect to the board again.

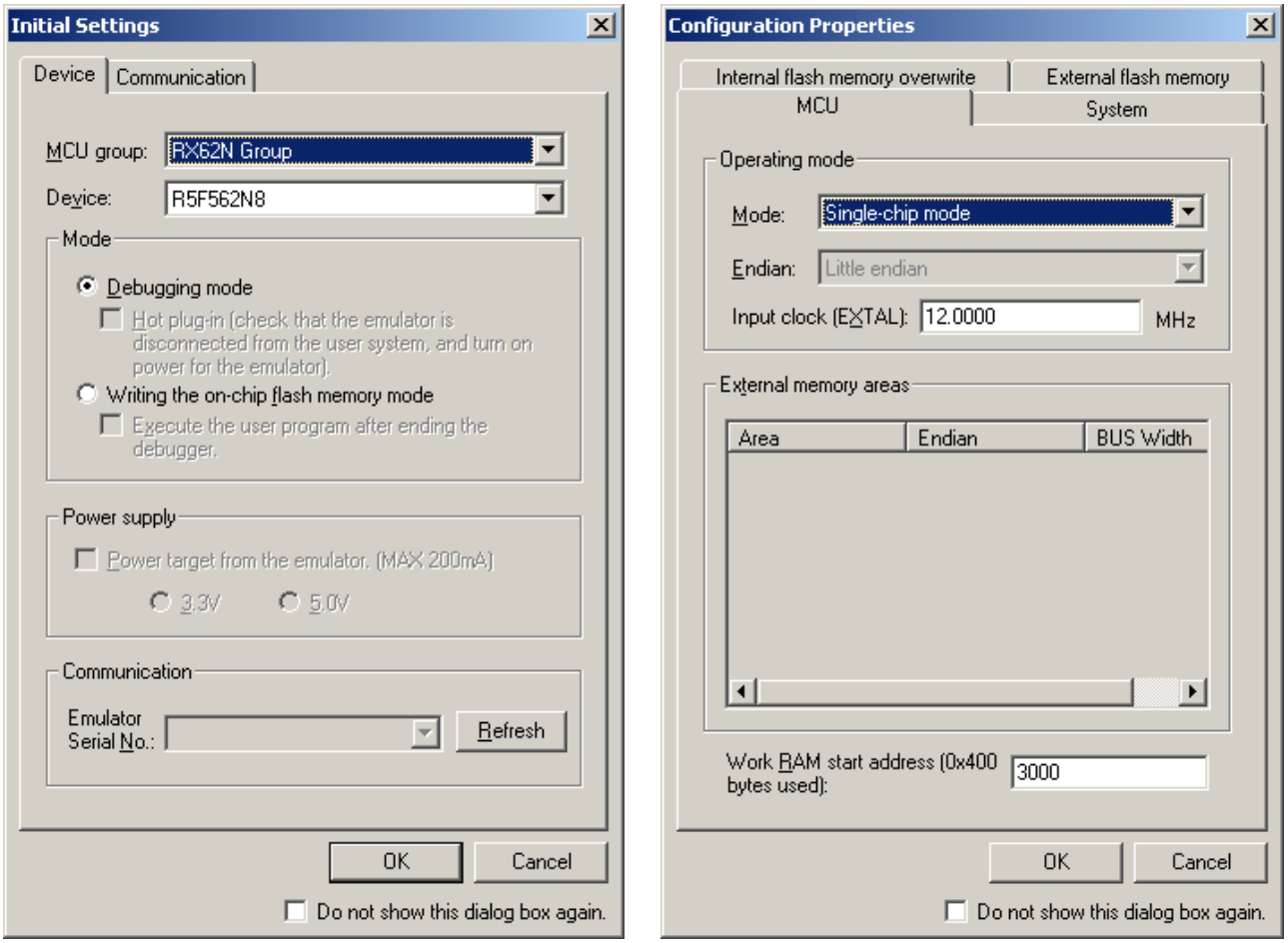


Figure 3-2 : Connect to RDK

Step 3.29 Download the code to the board by double clicking, or right click and select Download, on the *RX_RPDL_Demo.abs* file in the project navigation screen.

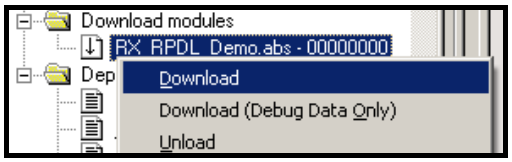


Figure 3-3 : Download code

Step 3.30 Click the Reset-Go button or click Debug >> Reset Go.

Step 3.31 Verify that the ADC value is shown on the board's LCD. If you rotate the potentiometer you should see that the value is automatically updated.

Step 3.32 Stop MCU execution, disconnect, and close HEW.

4. Using Peripheral Driver Generator (PDG)

This section of the lab will setup the same code as in the last project, except instead of writing the RPDL code manually we will use PDG. PDG will generate the code containing the RPDL invocations that we coded by hand in the previous exercise.

Procedural Steps

Step 4.1 Before beginning this example of how to use PDG, we must first verify that Hew Target Server is registered. Start HEW and then go to Tools >> Administration. Expand the Extension Components folder and verify that "HewTargetServer" is listed.

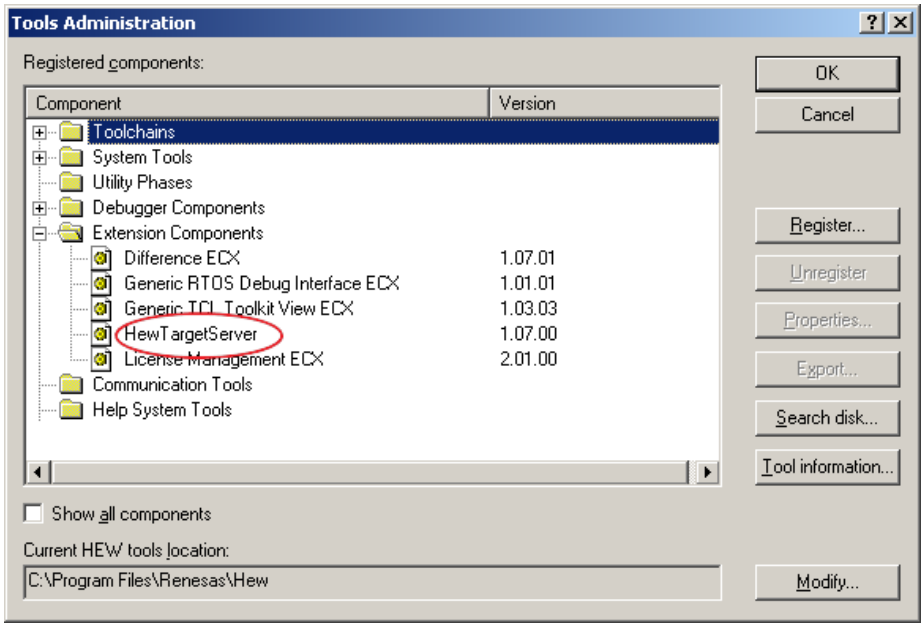


Figure 4-1 : Verify HTS Registry

Step 4.2 If it is not listed, click on "Search disk…" and then click "Start". When searching has finished select "HewTargetServer" from the list and click "Register". When done, close HEW.

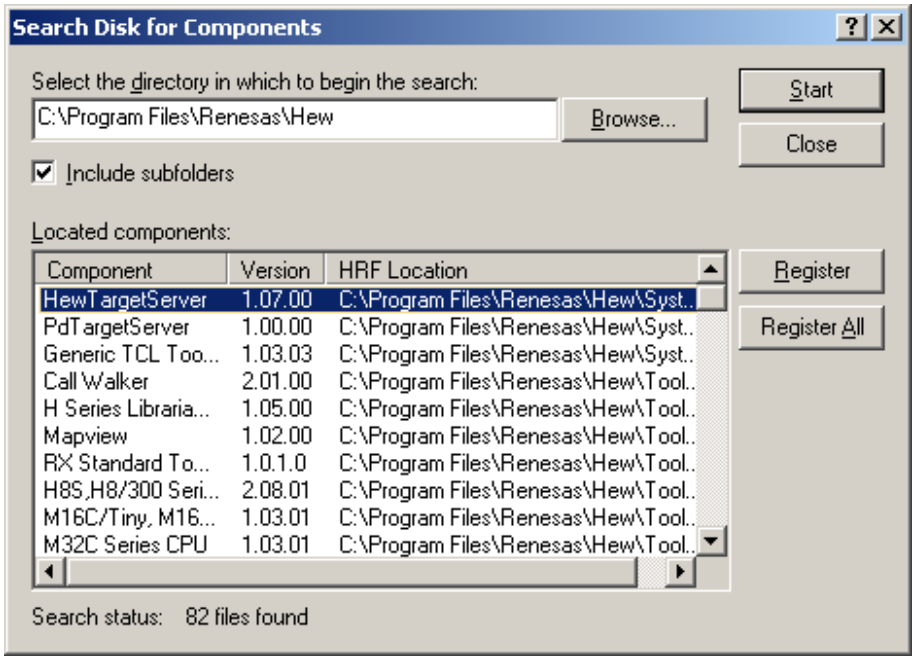


Figure 4-2: Adding HTS to Registry

Step 4.3 Start PDG by going to Start >> All Programs >> Renesas >> Peripheral Driver Generator 2 >> Peripheral Driver Generator 2.

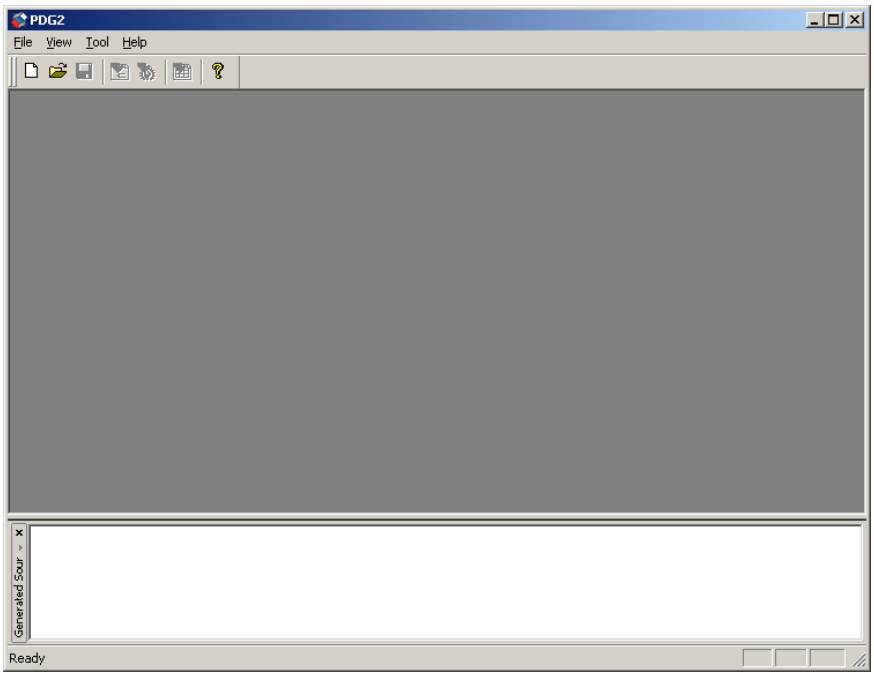


Figure 4-3: PDG Application

Step 4.4 Start a new PDG project by going to File >> New Project. The window below should pop up.

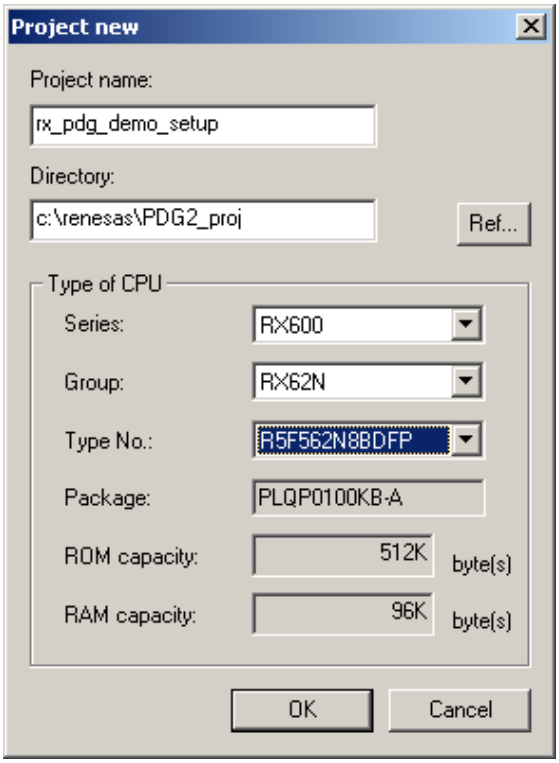


Figure 4-4: PDG New Project Window

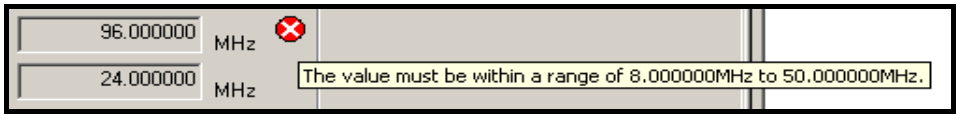
Step 4.5 Make your window match that of Figure 4-4 and click “OK”.

PDG projects are separate from HEW projects and therefore they do not have to be in the same directory.

Step 4.6 The window for controlling the RX peripherals will now show up. You can switch which peripheral you are currently editing by using the tabs at the bottom of the edit pane. By default the first pane selected will be "SYSTEM" which controls the system clocks. Just like before setup the clocks with these parameters:

- An input frequency of 12MHz on the EXTAL pin
- System clock = 96MHz
- Peripheral module clock = 48MHz
- External bus clock = 24MHz
- Dedicated USB clock = 48Mhz
- Disable BCLK output

Try to set the PCLK to 96MHz. Notice that PDG gives a red error sign noting that this is not possible. If you hover your mouse over the error sign then it will tell you why there is an error and what settings are valid. Also note that whenever there is an error with a peripheral PDG will show the red error sign on the peripheral tab.



Step 4.7 Next, switch to the 'ADa' tab and configure AD1 the same as was done with RPDL with the settings below. When done your window should look like Figure 4-5:

- Unit 1 (AD1)
- Channel AN4
- Single Mode
- Padded at the LSB
- TMR0 compare match is the trigger
- 48MHz conversion clock
- 0.6us for sampling time (0.5us is listed as minimum in HW manual)
- Call the function ADC_Callback() when conversion has finished
- Use an IPL of 4

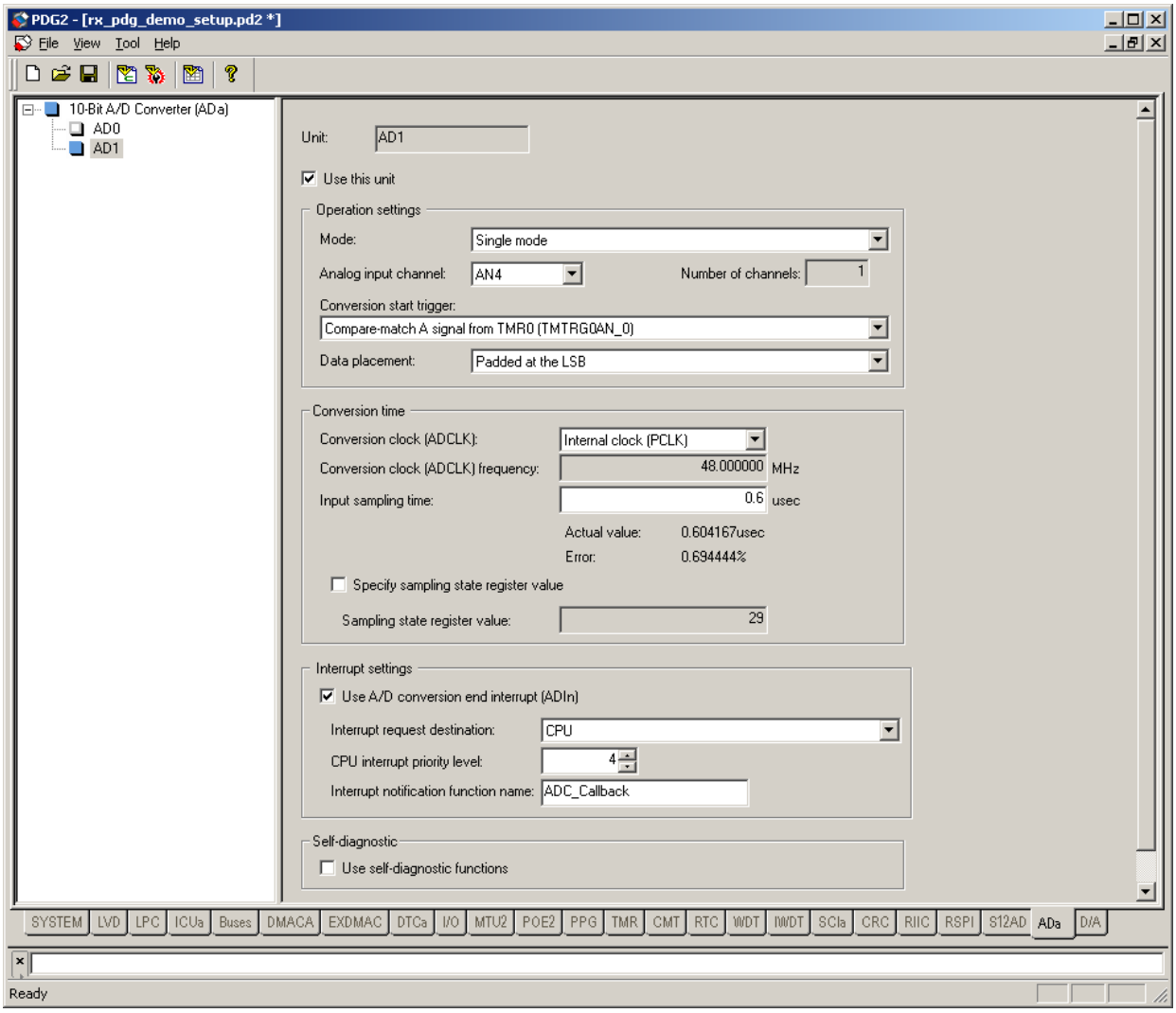


Figure 4-5: Configure AD1 with PDG

Step 4.8 Switch to the TMR tab and configure it with the following settings:

- Unit 0 (takes TMR0 and TMR1 and cascades them to make a 16-bit counter)
- Use PCLK divided by 8192
- Set a frequency of 4Hz (250 msec period)
- Clear on Compare match A
- ADC triggering
- 50% duty cycle
- No callback functions

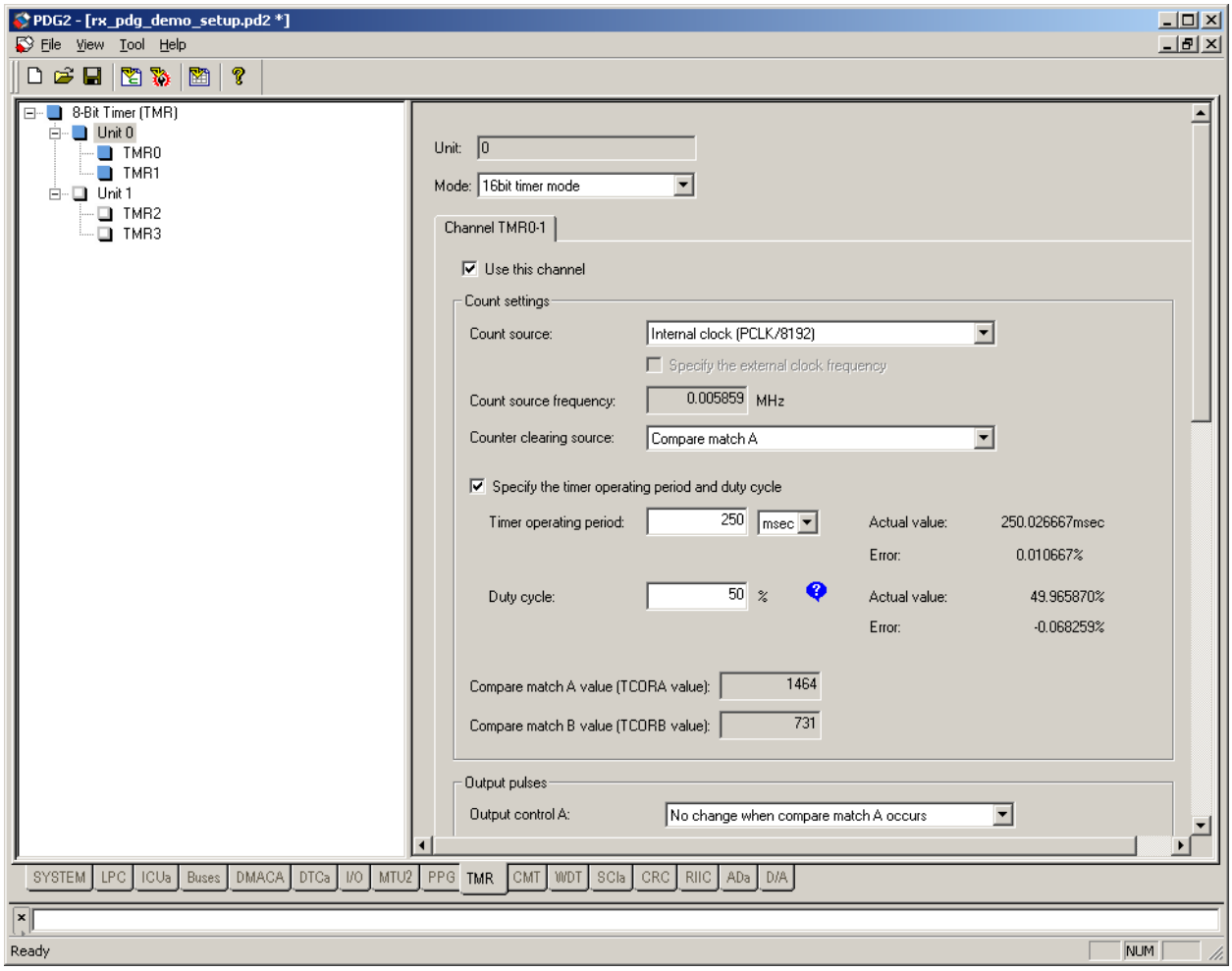


Figure 4-6: Configuring the 8-bit Timer with PDG

Don't forget to enable ADC triggering by checking the box at the bottom of the TMR pane.

Step 4.9 Save the project by going to File >> Save.

Step 4.10 Open up HEW and create a new workspace just like was done in Section 2 but this time name it "RX_PDG_Demo" and do not add RPDL (just repeat steps Step 2.1 through Step 2.11) after HEW creates the workspace.

Step 4.11 Copy the following files to your project as was done at the beginning of Section 3:

- *glyph_api.h*
- *Glyphlib_v2.lib*
- *lcd.c*
- *lcd.h*
- *lcd_utilities.c*
- *lcd_utilities.h*
- *YRDKRX62N.h*
- *YRDKRX62N_RSPI_API.c*
- *YRDKRX62N_RSPI_API.h*

Step 4.12 Add the files *lcd.c*, *lcd_utilities.c*, *YRDKRX62N_RSPI_API.c*, and *Glyphlib_v2.lib* to your current project.

Step 4.13 We are now ready to generate the code and register it in HEW. You could register the files manually in HEW but PDG uses HEW Target Server to automate this service. The first thing to do is generate the source files. In PDG click Tool >> Generate source files. A window telling you that it completed successfully will pop up. Click "OK" to close it.

When you generate source files with PDG they are placed underneath your PDG project folder. Even after registering them in HEW, they are not moved. This is convenient so that if you are using the source files with multiple projects they will always have the latest files.

Step 4.14 The next step is to register the generated files and RPDL with your HEW project. To do this in PDG click Tool >> Register source files in HEW project.

Step 4.15 A window will pop up asking you to make sure you have the desired workspace open. Click "OK".

Step 4.16 Next a window will pop up asking you to set the order in which libraries are linked. Since we are only using one library in this demo we do not need to worry about this. Click "OK".

Step 4.17 A window will pop up telling you that the files have been registered successfully. Click "OK" to close it.

Step 4.18 Switch back to HEW and you will notice these changes:

- There is a new folder "AddFromPDG" that contains the generated source files from PDG.
- The files *intprg.c* and *vecttbl.c* have automatically been excluded (we had to do this manually the earlier project).
- If you look in the Link/Library tab of the toolchain options window you will see that the RPDL library has been registered. (In PDG, the RPDL library for this MCU is named RX62N_library_LQFP_100.lib)

Step 4.19 In order to use the RPDL function calls in the file *YRDKRX62N_RSPI_API.c* we need to add an "include" directory. Go to Build >> RX Standard Toolchain.

Step 4.20 Make sure the "C/C++" tab is chosen and change the "Show entries for" drop down to "Include file directories".

Step 4.21 Click "Add". In the window that comes up change the "Relative to" drop down to "Custom directory".

Step 4.22 Click "Browse" and navigate to "C:\Renesas\PDG2\source\RX\RX62N\i_src" and click "Select". Click "OK".

Step 4.23 Switch the "Show entries for" drop down to "Defines".

Step 4.24 Click the "Add" button. In the window that pops up put "DEVICE_PACKAGE_LQFP_100" for the "Macro" entry and nothing for the "Replacement" entry. Click "OK".

The reason we added the "i_src" path and added the DEVICE_PACKAGE_LQFP_100 define was to be able to use the RPDL function calls inside the *YRDKRX62N_RSPI_API.c* source file. We could have added RPDL to the project like we did earlier but it was done this way to keep a common RPDL library. Since the PDG files used the RPDL library in the PDG directory, we used that directory as well. The #define was used to select what MCU we were using. The RPDL batch took care of this earlier when it asked us at the command prompt which MCU we were using. If you are not using RPDL outside of PDG, these steps are not needed.

Step 4.25 Click "OK" until you get back to the HEW project.

Step 4.26 To use the function generated by PDG we need to include the header file associated with our project. Open up *RX_PDG_Demo.c* and at the top of the file include the header file *r_pg_rx_pdg_demo_setup.h*. While doing this also include the header file for that we used before for accessing the LCD:

```
#include "r_pg_rx_pdg_demo_setup.h"

/* RSPI API library support */
#include "YRDKRX62N_RSPI_API.h"
#include "lcd_utilities.h"
#include "lcd.h"
```

To help with identifying PDG files all the files it generates will start with 'R_PG'. To find the code PDG generated for ADC channel 1 you would look in the file 'R_PG_ADC_10_AD1.c'.

Step 4.27 Now, we can call the functions generated by PDG. The easiest way to figure out these function names is to open the associated header file. We will follow the same setup as done with the RPDL project and first setup the system clocks. Open the header file *R_PG_Clock.h.*

Step 4.28 Notice there are 3 functions and one is named R_PG_Clock_Set(). Even though there are other functions (R_PG_Clock_Start_SUB() and R_PG_Clock_Stop_SUB()) it is easy to distinguish that the other functions pertain to the subclock. Go to the main() function inside of *RX_PDG_Demo.c* and add a function call to R_PG_Clock_Set().

Step 4.29 Open up the file *R_PG_Clock.c* and find the function R_PG_Clock_Set(). Verify that it is using the same RPDL call we created earlier in Section 2.

Step 4.30 Continue to build the project by using the PDG functions to setup the ADC and then the TMR.

Step 4.31 Add a function call after the PDG functions to initialize the RSPI bus and the LCD using the YRDKRX62N_RSPI_Init() and InitialiseLCD() function call. Also add an infinite loop at the end of the main() function.

```
void main(void)
{
    R_PG_Clock_Set();

    R_PG_ADC_10_Set_AD1();
    R_PG_Timer_Start_TMR_U0();

    /* Initialise the LCD display on RSPI bus */
    YRDKRX62N_RSPI_Init( 0 );

    /* Setup LCD */
    InitialiseLCD();

    while(1);
}
```

Step 4.32 Copy the same ADC_Callback() function from Step 3.21 and add it *RX_PDG_Demo.c*.

Step 4.33 This time instead of using the R_ADC_10_Read() RPDL function, use the R_PG_ADC_10_GetResult_AD1() PDG function found in *R_PG_ADC_10_AD1.h*.

Step 4.34 Click the Build button  or go to Build >> Build.

Step 4.35 Follow the directions in Section 3 to change your debugging session and to connect to the board.

Step 4.36 Download the code to the MCU and click Reset-Go  or go to Debug >> Reset Go.

Step 4.37 Verify that the code runs the same as it did in Section 3.

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	Mar.24.11	—	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.

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.

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