User's Manual

16

RL78/L1C Group

Renesas Starter Kit Code Generator Tutorial Manual For CubeSuite+

RENESAS MCU
RL78 Family / L1X Series

Renesas Electronics
www.renesas.com

Rev. 1.02 Apr 2014

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation 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.
2. 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.
3. 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.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.

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

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.

   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). 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 for which it is not intended. 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 which the product is not intended by Renesas Electronics.
6. 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.
7. 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 systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. 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.
9. 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. You should not use Renesas Electronics products or 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. When exporting the Renesas Electronics 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.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
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.

(2012.4)

Disclaimer

By using this Renesas Starter Kit (RSK), the user accepts the following terms:

The RSK is not guaranteed to be error free, and the entire risk as to the results and performance of the RSK is assumed by the User. The RSK is provided by Renesas on an “as is” basis without warranty of any kind whether express or implied, including but not limited to the implied warranties of satisfactory quality, fitness for a particular purpose, title and non-infringement of intellectual property rights with regard to the RSK. Renesas expressly disclaims all such warranties. Renesas or its affiliates shall in no event be liable for any loss of profit, loss of data, loss of contract, loss of business, damage to reputation or goodwill, any economic loss, any reprogramming or recall costs (whether the foregoing losses are direct or indirect) nor shall Renesas or its affiliates be liable for any other direct or indirect special, incidental or consequential damages arising out of or in relation to the use of this RSK, even if Renesas or its affiliates have been advised of the possibility of such damages.

Precautions

The following precautions should be observed when operating any RSK product:

This Renesas Starter Kit is only intended for use in a laboratory environment under ambient temperature and humidity conditions. A safe separation distance should be used between this and any sensitive equipment. Its use outside the laboratory, classroom, study area or similar such area invalidates conformity with the protection requirements of the Electromagnetic Compatibility Directive and could lead to prosecution.

The product generates, uses, and can radiate radio frequency energy and may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment causes harmful interference to radio or television reception, which can be determined by turning the equipment off or on, you are encouraged to try to correct the interference by one or more of the following measures;

- ensure attached cables do not lie across the equipment
- reorient the receiving antenna
- increase the distance between the equipment and the receiver
- connect the equipment into an outlet on a circuit different from that which the receiver is connected
- power down the equipment when not in use
- consult the dealer or an experienced radio/TV technician for help NOTE: It is recommended that wherever possible shielded interface cables are used.

The product is potentially susceptible to certain EMC phenomena. To mitigate against them it is recommended that the following measures be undertaken;

- The user is advised that mobile phones should not be used within 10m of the product when in use.
- The user is advised to take ESD precautions when handling the equipment.

The Renesas Starter Kit does not represent an ideal reference design for an end product and does not fulfil the regulatory standards for an end product.

How to Use This Manual

1. Purpose and Target Readers

This manual is designed to provide the user with an understanding of how to use Application Leading Tool (Applilet) for RL78 together with the CubeSuite+ IDE to create a working project for the RSK platform. It is intended for users designing sample code on the RSK platform, using the many different incorporated peripheral devices.

The manual comprises of step-by-step instructions to generate code and import it into CubeSuite+, but does not intend to be a complete guide to software development on the RSK platform. Further details regarding operating the RL78/L1C microcontroller may be found in the Hardware Manual and within the provided sample code.

Particular attention should be paid to the precautionary notes when using the manual. These notes occur within the body of the text, at the end of each section, and in the Usage Notes section.

The revision history summarizes the locations of revisions and additions. It does not list all revisions. Refer to the text of the manual for details.

The following documents apply to the RL78/L1C Group. Make sure to refer to the latest versions of these documents. The newest versions of the documents listed may be obtained from the Renesas Electronics Web site.

Document Type	Description	Document Title	Document No.
User's Manual	Describes the technical details of the RSK hardware.	RSKRL78L1C User's Manual	R20UT2203EG
Tutorial	Provides a guide to setting up RSK environment, running sample code and debugging programs.	RSKRL78L1C Tutorial Manual	R20UT2204EG
Code Generator Tutorial	Provides a guide to code generation and importing into the CubeSuite+ IDE.	RSKRL78L1C Code Generator Tutorial Manual	R20UT2887EG
Quick Start Guide	Provides simple instructions to setup the RSK and run the first sample.	RSKRL78L1C Quick Start Guide	R20UT2205EG
Schematics	Full detail circuit schematics of the RSK.	RSKRL78L1C Schematics	R20UT2202EG
Hardware Manual	Provides technical details of the RL78/L1C microcontroller.	RL78/L1C Group Hardware Manual	R01UH0409EJ

2. List of Abbreviations and Acronyms

Abbreviation	Full Form
ADC	Analog-to-Digital Converter
API	Application Programming Interface
CPU	Central Processing Unit
DVD	Digital Versatile Disc
E1	On-chip Debugger
GUI	Graphical User Interface
LCD	Liquid Crystal Display
LED	Light Emitting Diode
MCU	Micro-controller Unit
RSK	Renesas Starter Kit
ROM	Read-Only Memory
SAU	Serial Array Unit
TAU	Timer Array Unit
UART	Universal Asynchronous Receiver/Transmitter
WDT	Watchdog Timer

Table of Contents

1. Overview 7
1.1 Purpose 7
1.2 Features 7

2. Introduction 8

3. Code Generation Using Applilet 9
3.1 Introduction 9
3.2 Applilet Tour 9
3.3 Code Generation 12

4. Importing into CubeSuite+ 25
4.1 Starting CubeSuite+ and Importing Applilet Code 25
4.2 Project Settings 27
4.3 LCD Panel Code Integration 29
4.4 Switch Code Integration 30
4.5 Debug Code Integration 37
4.6 UART Code Integration 38
4.7 LED Code Integration 41

5. Debugging the Project 43

6. Running the Code Generator Tutorial 44
6.1 Running the Tutorial 44

7. Additional Information 45

1. Overview

1.1 Purpose

This RSK is an evaluation tool for Renesas microcontrollers. This manual describes how to Application Leading Tool (Applilet) for RL78 together with the CubeSuite+ IDE to create a working project for the RSK platform.

1.2 Features

This RSK provides an evaluation of the following features:

- Code Generation using Applilet for RL78/L1C.
- Project Creation and Building with CubeSuite+
- User circuitry such as switches, LEDs and a potentiometer

The RSK board contains all the circuitry required for microcontroller operation.

2. Introduction

This manual is designed to answer, in tutorial form, how to use Applilet for the RL78 family together with the CubeSuite+ IDE to create a working project for the RSK platform. The tutorials help explain the following:

- Detailed use of Applilet for MCU peripheral configuration and code generation
- Importing generated code into CubeSuite+ projects
- Integration with custom code
- Building the project CubeSuite+

The project generator will create a tutorial project with three selectable build configurations:

- ‘DefaultBuild’ is a project with debug support and optimisation level set to two.
- ‘Debug’ is a project built with the debugger support included. Optimisation is set to zero.
- ‘Release’ is a project with optimised compile options, producing code suitable for release in a product.

Some of the illustrative screenshots in this document will show text in the form RL78XXX. These are general screenshots and are applicable across the whole RL78 family. In this case, simply substitute for RL78XXX RL78/L1C

These tutorials are designed to show you how to use the RSK and are not intended as a comprehensive introduction to the CubeSuite+ debugger, compiler toolchains or the E1 emulator. Please refer to the relevant user manuals for more in-depth information.

3. Code Generation Using Applilet

3.1 Introduction

Applilet is a Windows™ GUI tool for generating template 'C' source code and project settings for the RL78 family. When using Applilet, the engineer is able to configure various MCU features and operating parameters using intuitive GUI controls, thereby bypassing the need in most cases to refer to sections of the Hardware Manual.

Once the engineer has configured the project, the 'Generate Code' function is used to generate three code modules for each specific MCU feature selected. These code modules are name 'r_cg_xxx.h', 'r_cg_xxx.c', and 'r_cg_xxx_user.c', where 'xxx' is a three letter acronym for the relevant MCU feature, for example 'adc'. Within these code modules, the engineer is then free to add custom code to meet their specific requirement. Custom code should be added, whenever possible, in between the following comment delimiters:

```
/* Start user code for adding. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```

Applilet will locate these comment delimiters, and preserve any custom code inside the delimiters on subsequent code generation operations. This is useful if, after adding custom code, the engineer needs to re-visit Applilet to change any MCU operating parameters.

Applilet is released with this RSK, and is available via a web download at:

http://www.renesas.com/applilet_download

By following the steps detailed in this Tutorial, the user will generate an CubeSuite+ project called CG_Tutorial. The fully completed Tutorial project is contained on the DVD and may be imported into CubeSuite+ by following the steps in the Quick Start Guide. This Tutorial is intended as a learning exercise for users who wish to use the Applilet code generator to generate their own custom projects for CubeSuite+.

The CG_Tutorial project uses interrupts for switch inputs, the ADC module, the Timer Array Unit (TAU), the Serial Array Unit (SAU) and uses these modules to perform A/D conversion and display the results via the UART to a terminal program and also on the on-board LCD panel on the RSK.

Following a tour of the key user interface features of Applilet in §3.2, the reader is guided through each of the peripheral function configuration dialogs in §3.3. In §4, the reader is shown how to import the project into CubeSuite+, where the reader will be familiarised with the structure of the template code, as well as how to add their own code to the user code areas provided by the code generator.

The Applilet installer is contained on the DVD. This installer must be run before proceeding to the next section.

3.2 Applilet Tour

In this section a brief tour of Applilet is presented. For further details of the Applilet paradigm and reference, refer to the Application Leading Tool Common Operations manual (R20UT2663EJ).

Launch Applilet from Start -> All Programs -> Renesas Electronics Application Leading Tool-> RL78-> Vx.xx.xx-> RL78 Vx.xx.xx Application Leading Tool. Vx.xx.xx represents the installed version number, different versions of Applilet can be started from here. On first launch, the user should be presented with the new project dialog as shown in Figure 3-1. To get to this dialog on subsequent launches, from the Applilet menus select 'File -> New'.

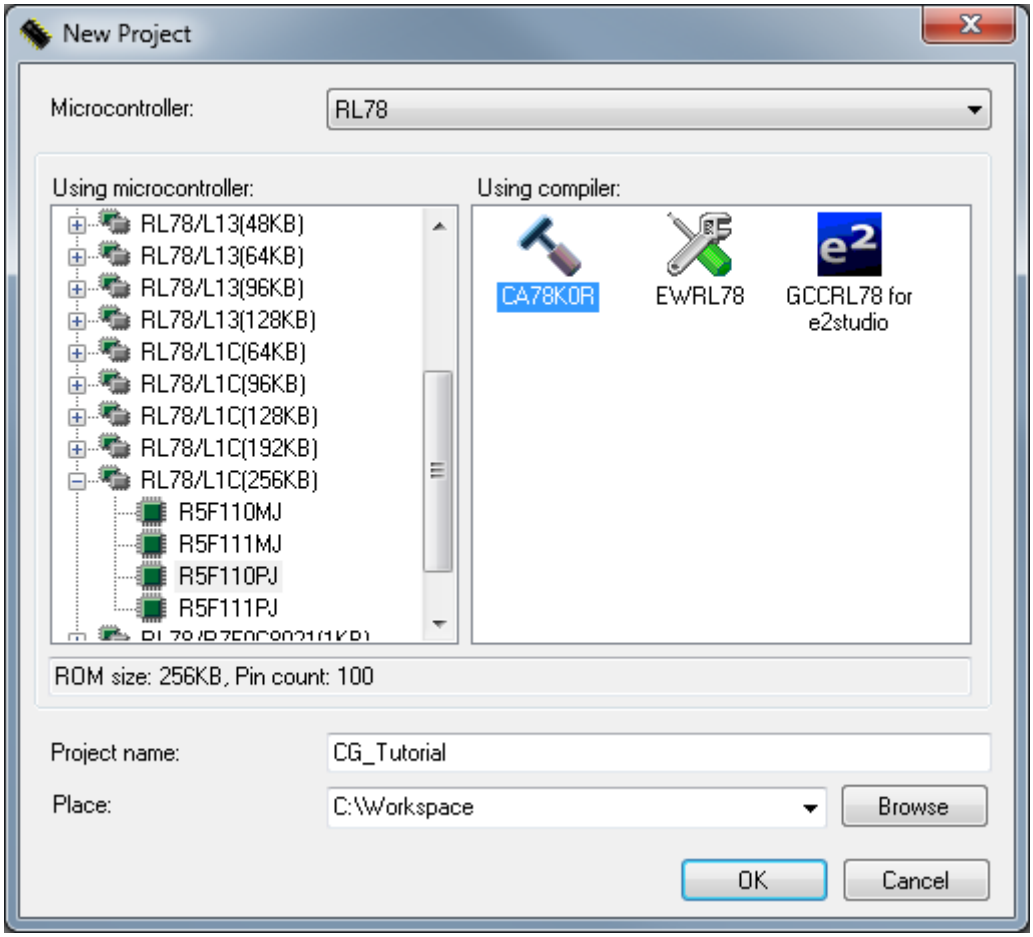


Figure 3-1 New Project Dialog

In the ‘Using Microcontroller’ pane, locate and expand the ‘RL78/L1C(256KB)’ item and select ‘R5110PJ’. In the ‘Using Compiler’ pane, ensure that ‘CA78K0R is selected. Choose a suitable location for storing the Applilet-generated files, in the example shown in Figure 3-1 this is ‘C:\Workspace’. Also, choose a suitable name for the project (CG_Tutorial in our example).

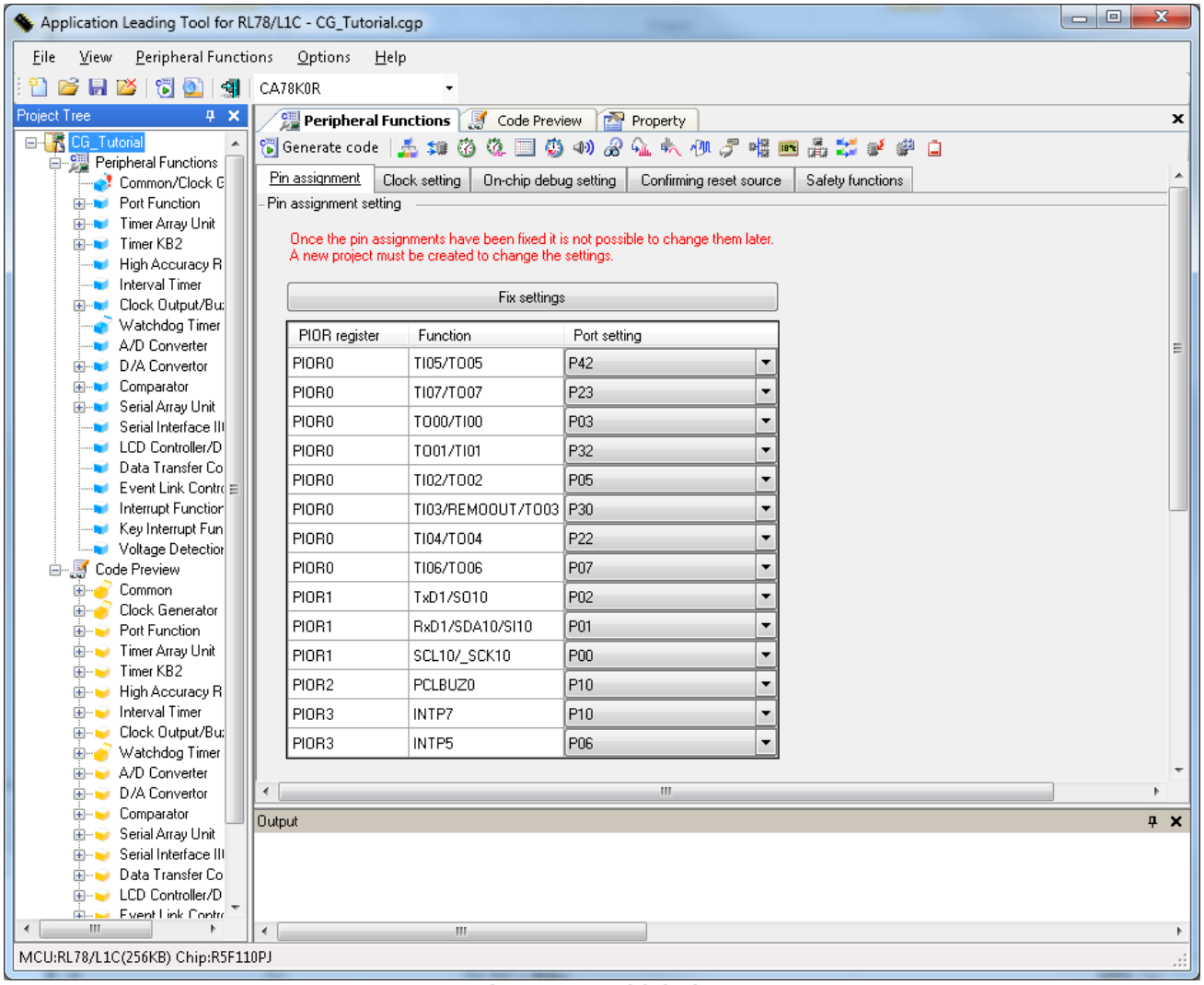


Figure 3-2 Initial View

Applilet provides GUI features for configuration of MCU sub systems. Once the user has configured all required MCU sub systems and peripherals, the user can click the 'Generate Code' button, resulting in a fully configured CubeSuite+ project that builds and runs without error.

Navigation to the MCU peripheral configuration screens may be performed in three different ways:

- By double-clicking the required function in the Project Tree -> Project Name -> Peripheral Function on the left.
- By using the menu item 'Peripheral Functions'
- By using the graphical toolbar in the main application area, when the 'Peripheral Functions' tab is selected.

It is also possible to see a preview of the code that will be generated for the current peripheral function settings by double-clicking the required function in the Project Tree -> Project Name -> Code Preview on the left.

The 'View' menu item may also be used to switch between main area tabs for Peripheral Function, Code Preview and Property tabs.

When Applilet is launched for the first time the user is presented with the ‘Pin assignment’ sub-tab under the ‘Peripheral Functions’ tab. Certain MCU pins in the RL78/L1C are configurable for different peripheral functions. In order to proceed to setting up the MCU peripheral functions, the user must first fix these pin assignments using the ‘Fix settings’ button. Once fixed, these pin assignments may not be changed and it will be necessary to re-start Applilet with a new project if different pin assignments are required.

For the purposes of this Code Generator Tutorial using the RSK, the default settings shown in Figure 3-2 above are applicable. The reader may click ‘Fix settings’ and proceed onto to the next Code Generation section.

3.3 Code Generation

In the following sub-sections, the reader is guided through the steps to configure the MCU for a simple project containing interrupts for switch inputs, timers, ADC and a UART.

3.3.1 Common/Clock Generator

Figure 3-3 shows a screenshot of Applilet with the Common/Clock Generator function open. Click on the ‘Clock setting’ sub tab. Configure the system clocks as shown in the figure. In this tutorial we are using the on board 12 MHz crystal for our main system clock fMAIN and the on board 32.768 kHz crystal for our sub clock fSUB. The CPU and peripheral use fMAIN and the RTC/Interval Timer/LCD use fSUB.

The selections for sub tabs ‘On-chip debug setting’, ‘Confirming reset source’ and ‘Safety functions’ can be left at their defaults.

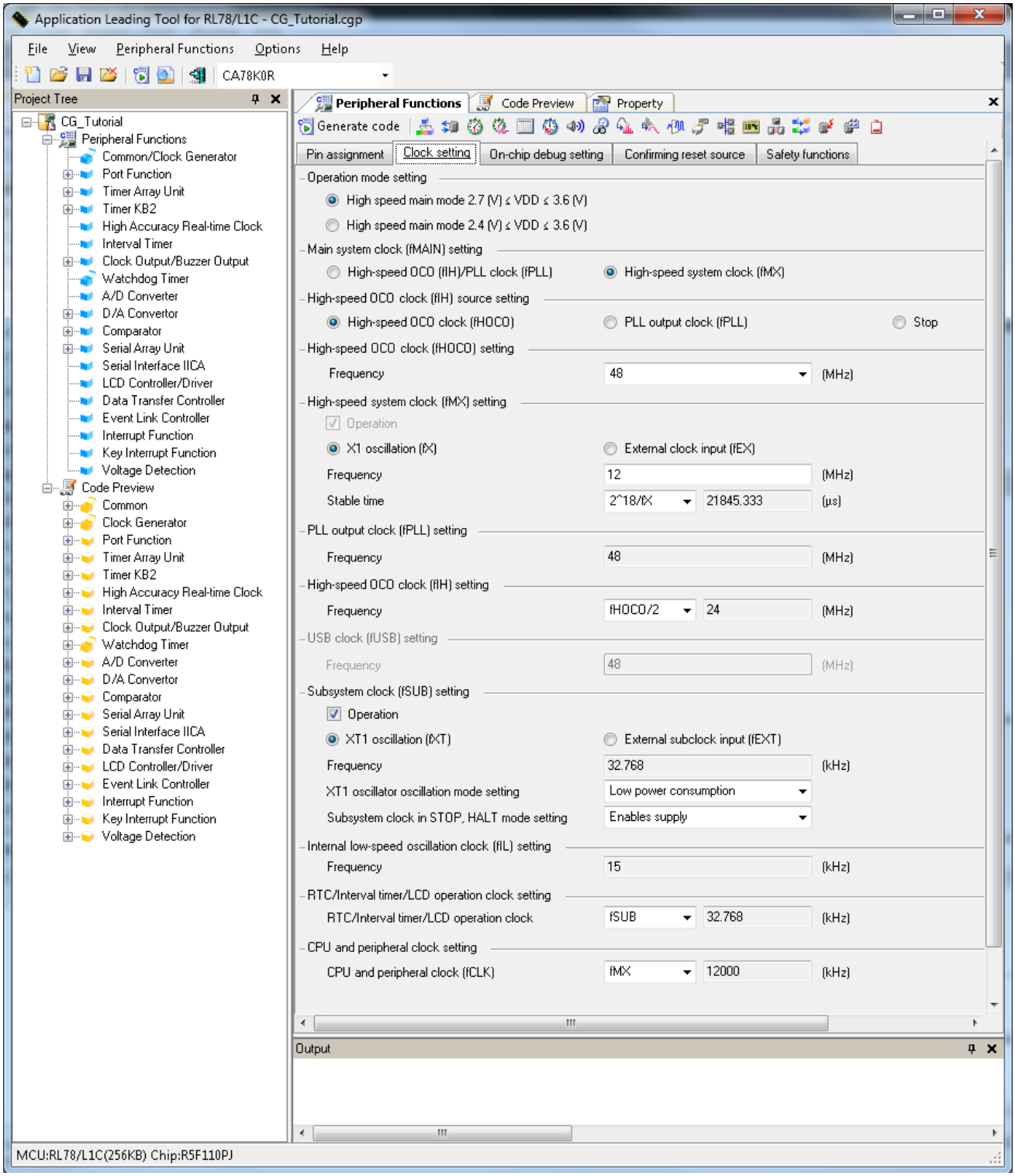


Figure 3-3 Clock setting tab

Proceed to the next section on Interrupt Functions. Although the next item in Applilet is Port Function, it is instructive to first configure the other peripherals, thereby reserving some of the I/O pins such that they may not be selected for normal I/O in the tab Port Function later.

3.3.2 Interrupt Functions

Referring to the RSK schematic, SW1 is connected to INTP0, SW2 is connected to INTP1 and SW3 is connected to INTP2. Navigate to the 'Interrupt Functions' tab in Applilet and configure these three interrupts as shown in Figure 3-4 below.

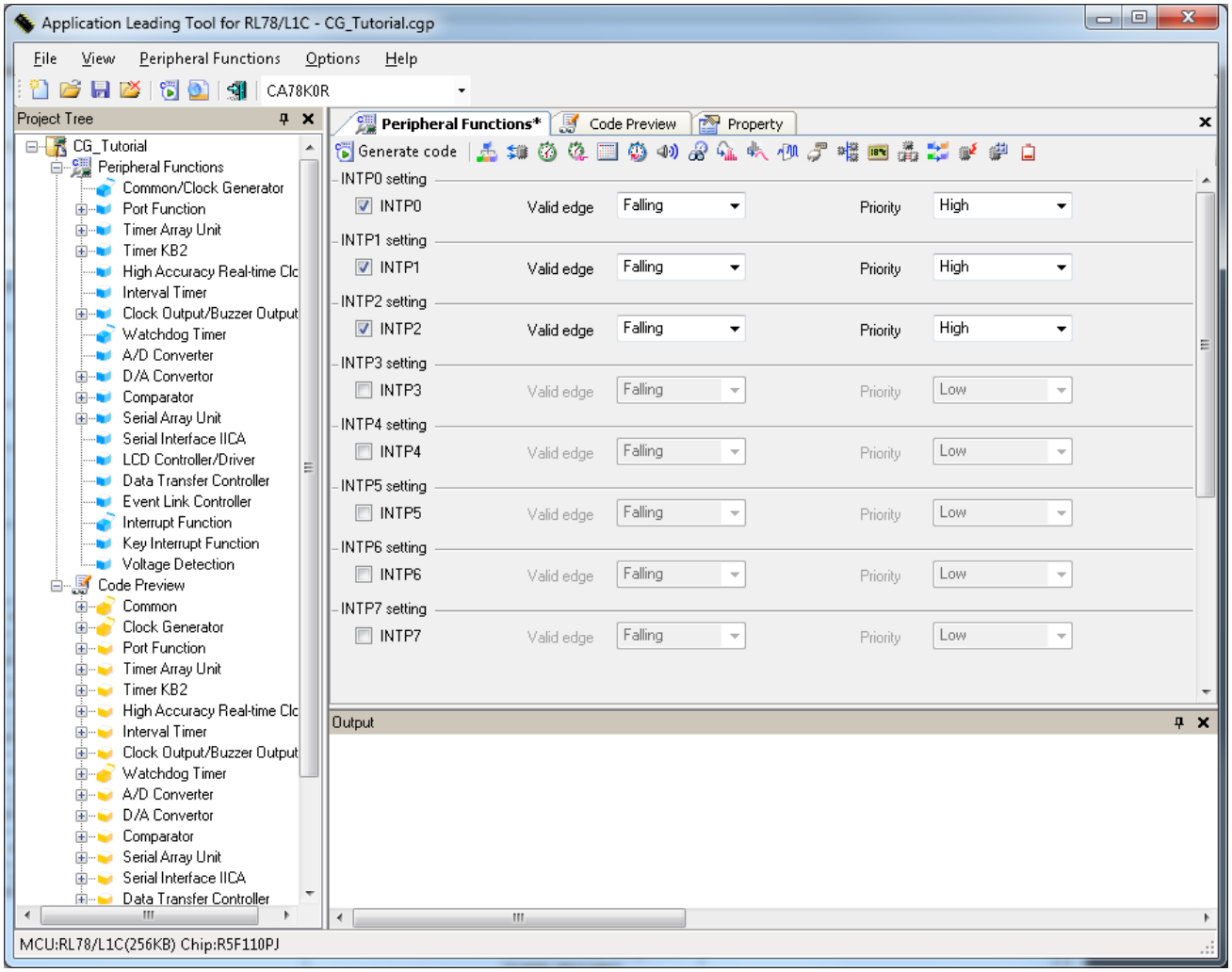


Figure 3-4 Interrupt Functions tab

3.3.3 LCD Controller/Driver

Navigate to the 'LCD Controller/Driver' tab in Applilet and configure the LCD as shown in Figure 3-5 below. Note the exclamation mark next to 'SEG51' and SEG52'. Move the mouse pointer over these exclamation marks to see the tool tip pop-ups, indicating that these pins have already been configured for INTP1 and INTP2 respectively.

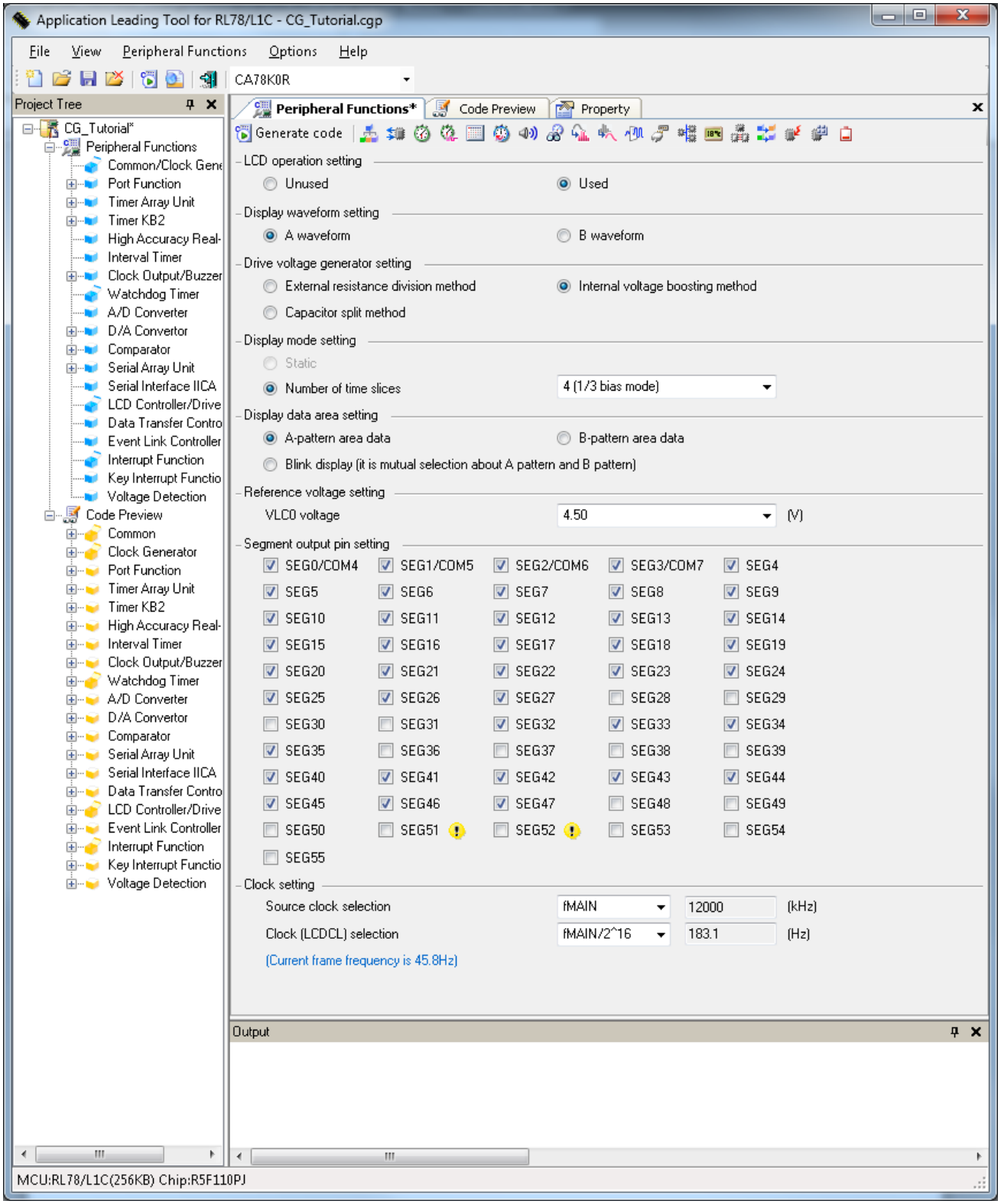


Figure 3-5 LCD Controller/Driver tab

3.3.4 Timer Array Unit

Navigate to the 'Timer Array Unit' tab in Applilet. Refer to the screenshot shown in Figure 3-6. Use the pull-down controls to configure Channels 0-3 as Interval Timer as shown. Channel 0 will be used as an interval timer for generation of accurate delays. Channels 1 and 2 will be used as timers in de-bouncing of switch interrupts.

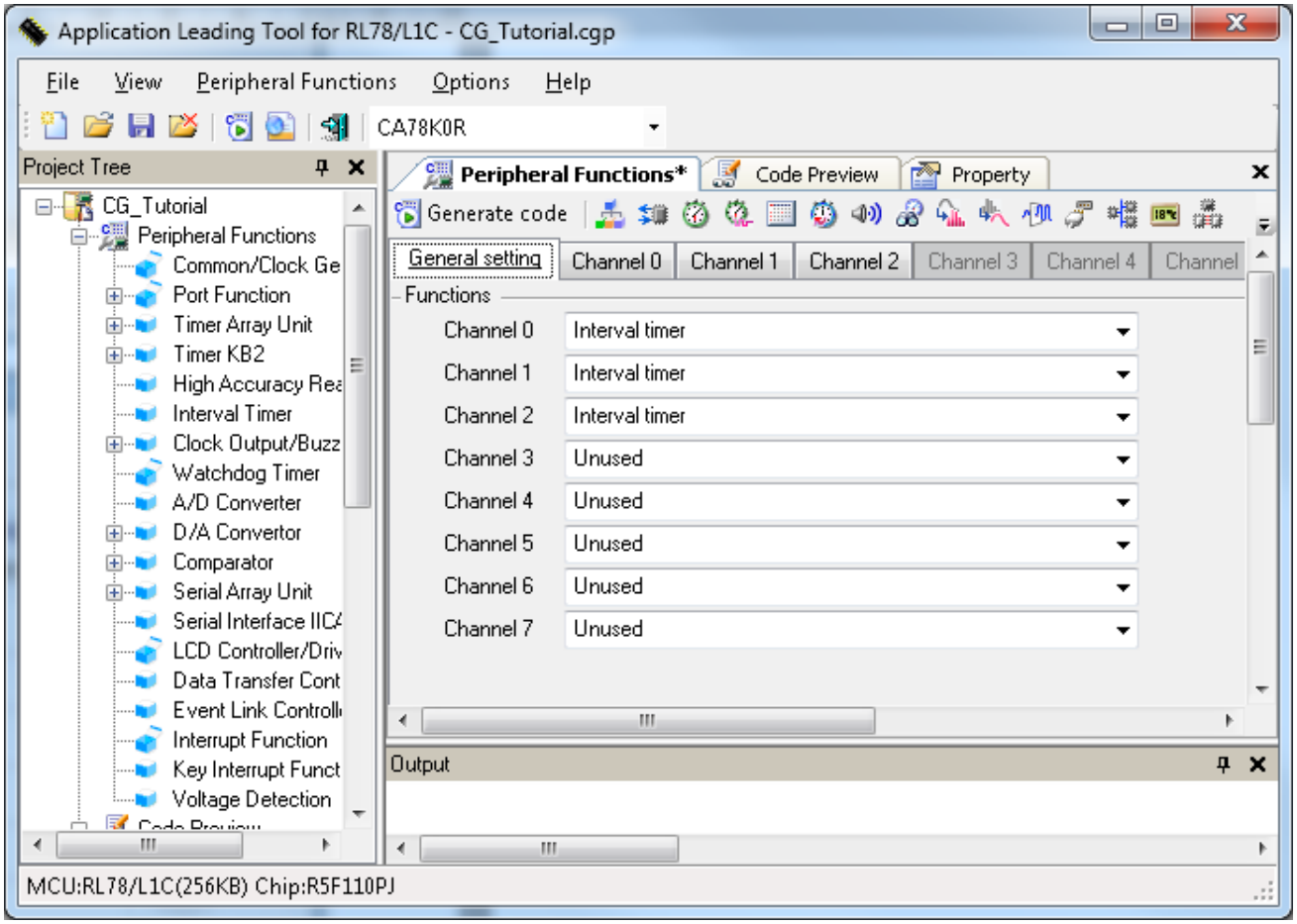


Figure 3-6 Timer Array Unit tab – General setting

Navigate to the 'Channel 0' tab and configure channel 0 as shown in Figure 3-7. This timer is configured to generate a High priority interrupt every 1ms. We will use this interrupt later in the tutorial to provide an API for generating high accuracy delays required in our application.

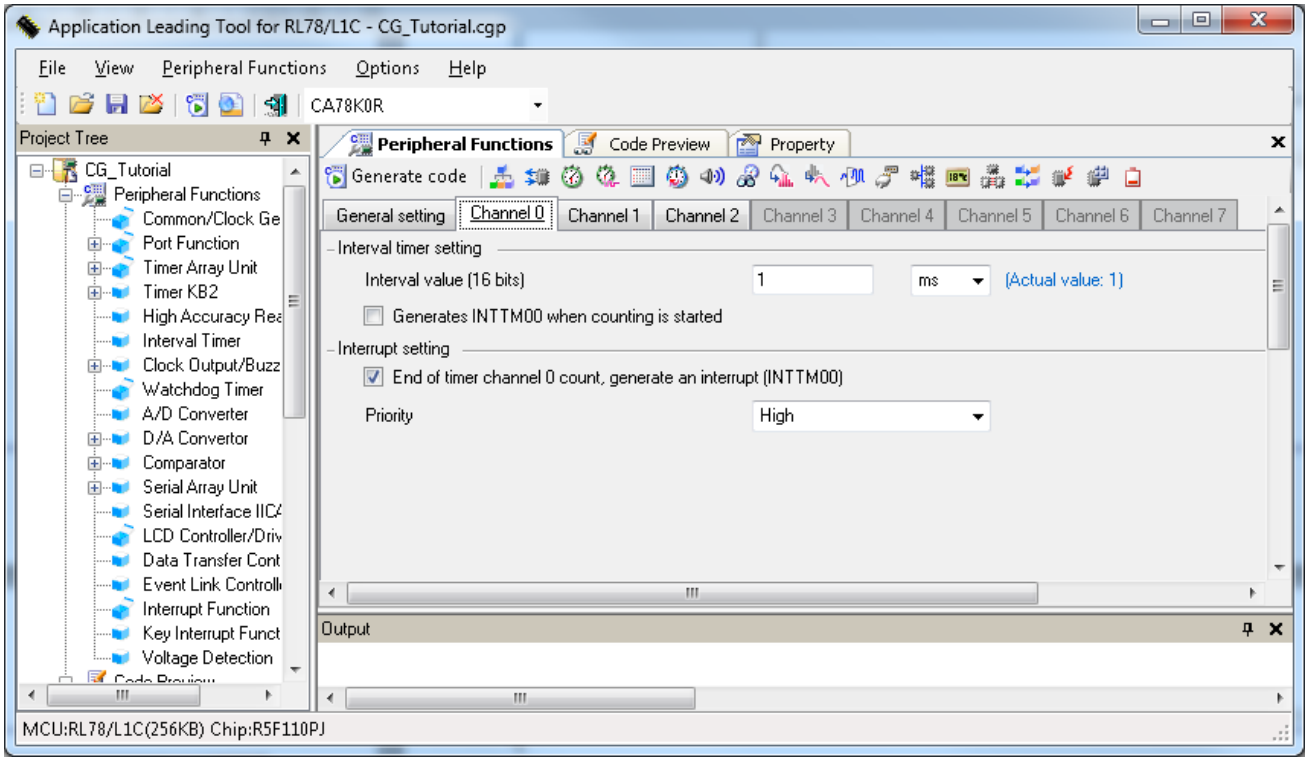


Figure 3-7 Timer Array Unit tab – Channel 0

Navigate to the 'Channel 1' tab and configure channel 1 as shown in Figure 3-8. This timer is configured to generate a High priority interrupt after 20ms. This timer is used as our short switch de-bounce timer later in this tutorial.

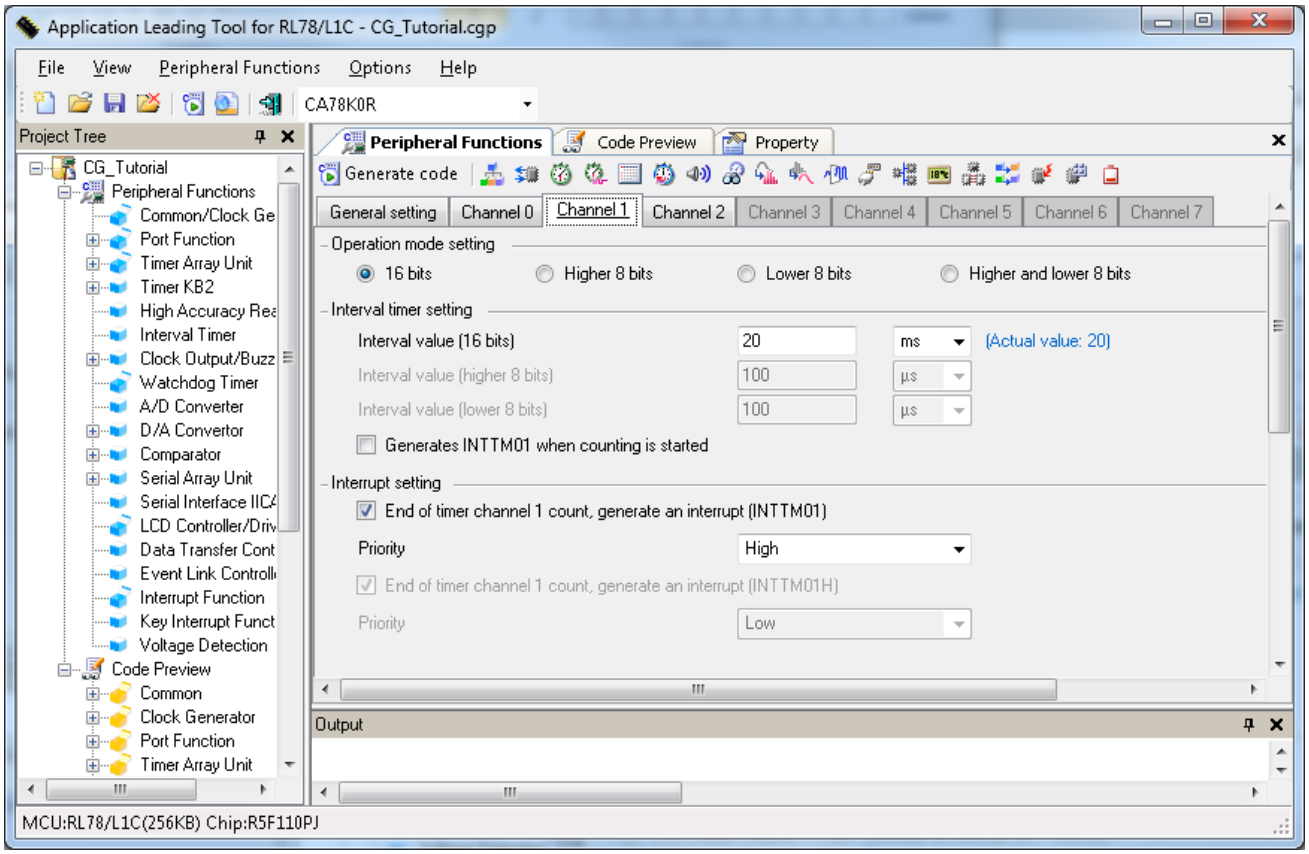


Figure 3-8 Timer Array Unit tab – Channel 1

Navigate to the ‘Channel 2’ tab and configure channel 2 as shown in Figure 3-9. This timer is configured to generate a High priority interrupt after 200ms. This timer is used as our long switch de-bounce timer later in this tutorial.

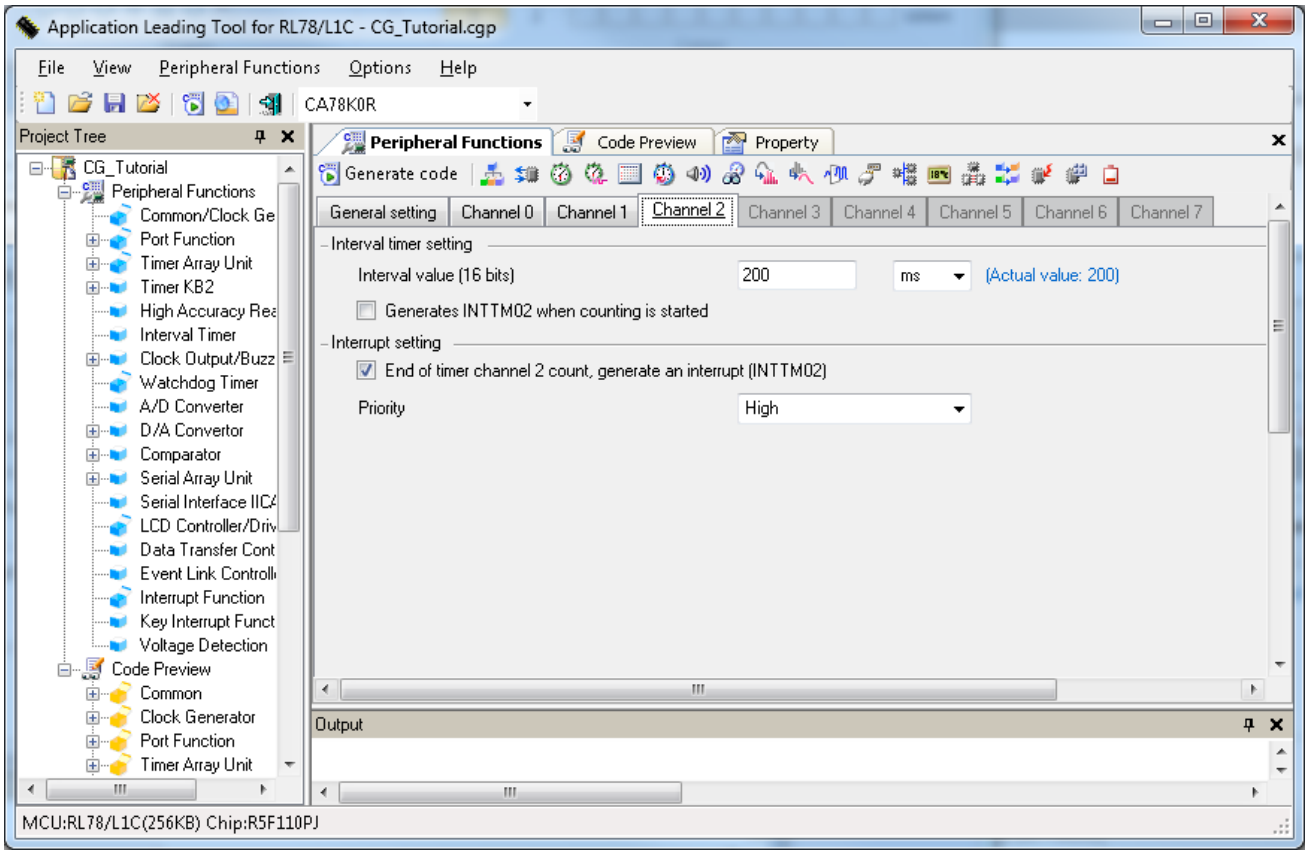


Figure 3-9 Timer Array Unit tab – Channel 2

3.3.5 Watchdog Timer

The Watchdog Timer is enabled by default. Navigate to the Watchdog Timer tab and select ‘Unused’ for the Watchdog timer operation setting.

3.3.6 A/D Converter

Navigate to the ‘A/D Converter’ tab in Applilet. Refer to the screenshot shown in Figure 3-10 and configure the ADC as shown. We will be using the ADC in 12-bit one shot mode on the ANI0 input, which is connected to the RV1 potentiometer output on the RSK.

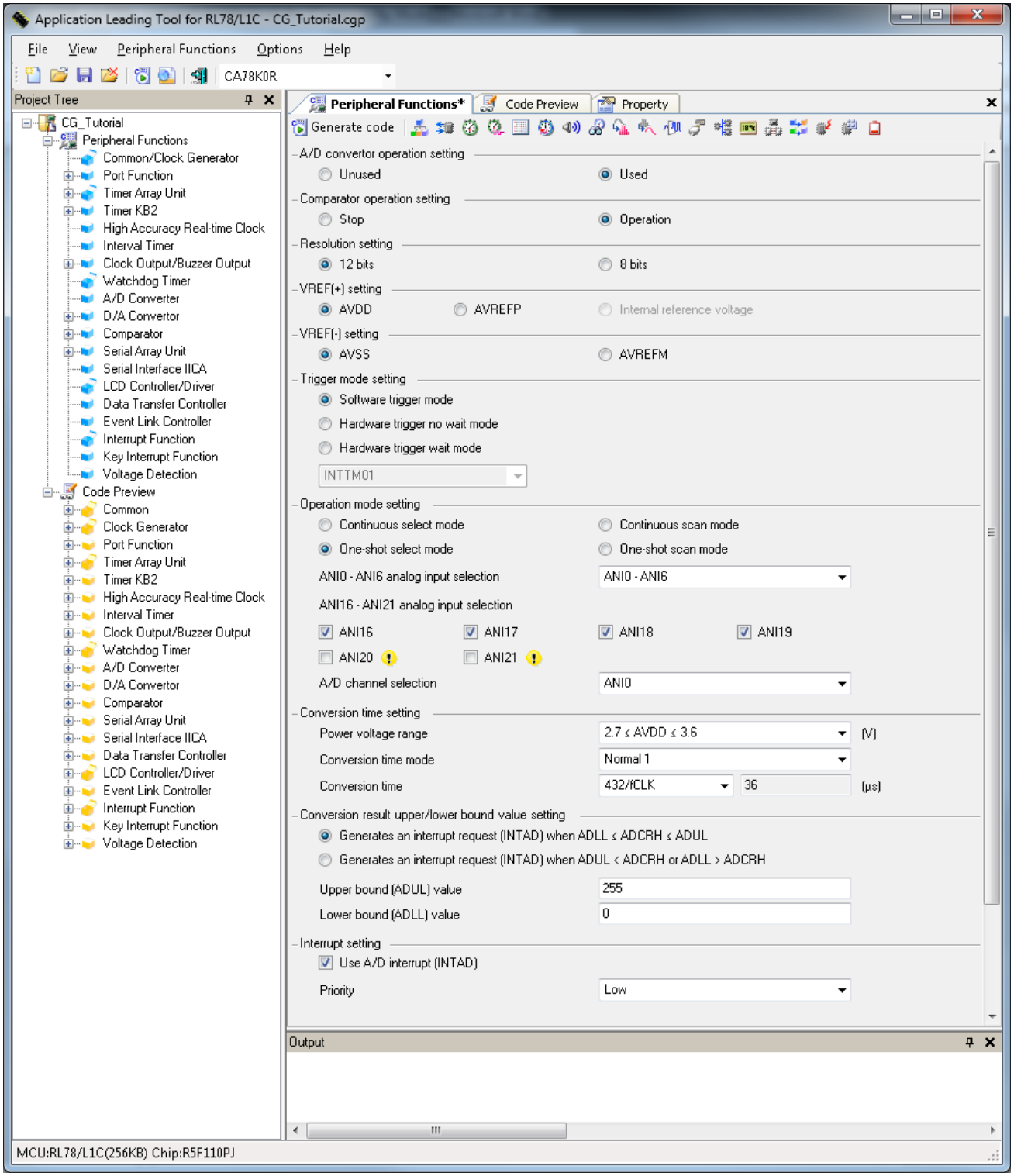


Figure 3-10 A/D Converter tab

3.3.7 Serial Array Unit

Navigate to the 'Serial Array Unit' tab in Applilet and refer to the screenshot shown in Figure 3-11. In the RSKRL78L1C UART1 lines TXD1 and RXD1 are connected to the MAX232R RS-232 line transceiver as shown in the schematic. Internally in the RL78/L1C MCU, UART1 is implemented in SAU0 Channel 2. As shown in Figure 3-11, use the pull-down control for Channel 2 and select UART1. Configure this UART for Transmit/receive function. The UART1 tab above will now be activated. Select this tab and refer to Figure 3-12.

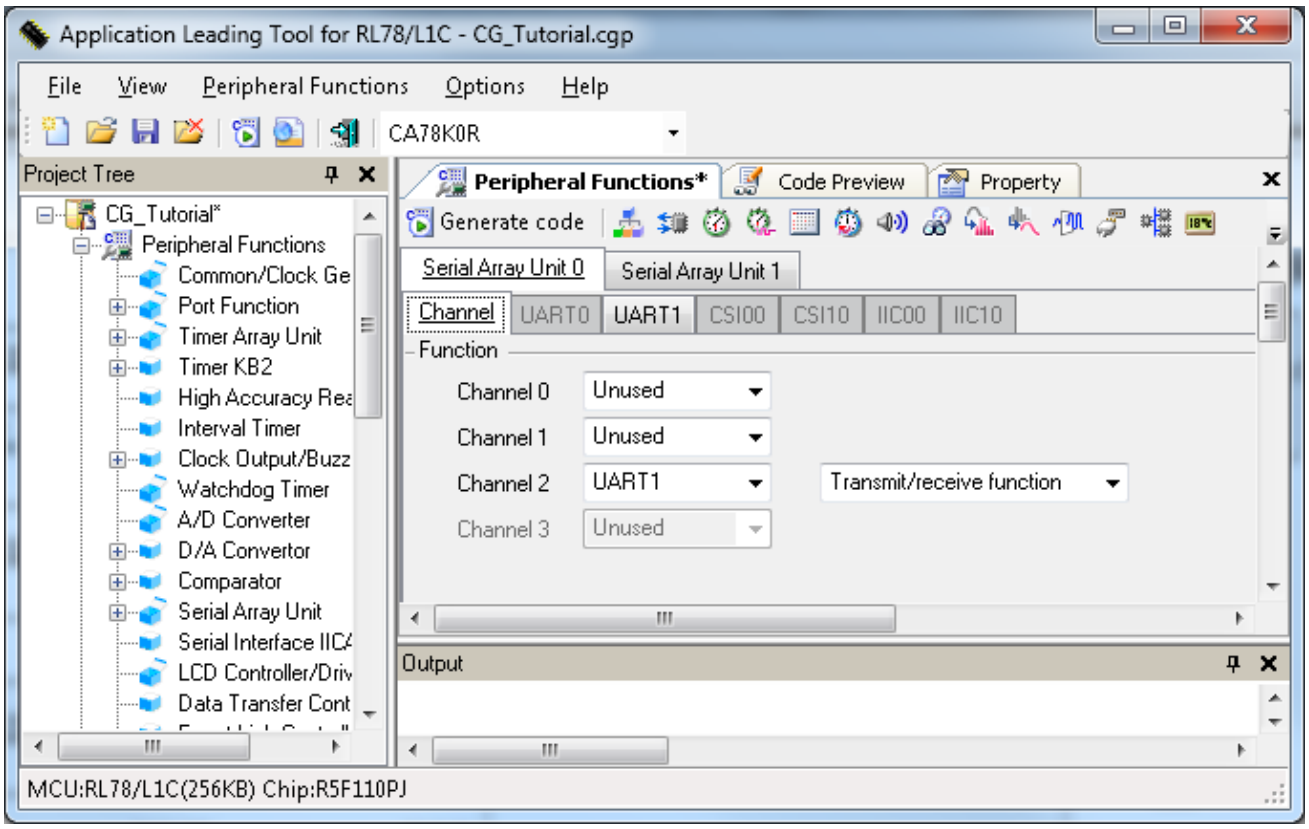


Figure 3-11 Serial Array Unit 0 tab – Channel function selection

Configure UART1 for 19200, 8, N, 1 as shown in Figure 3-12. Ensure that both callback function settings are selected. Ensure the same settings are made for the Transmit tab and that the Transfer mode is set for 'Single Transfer Mode'.

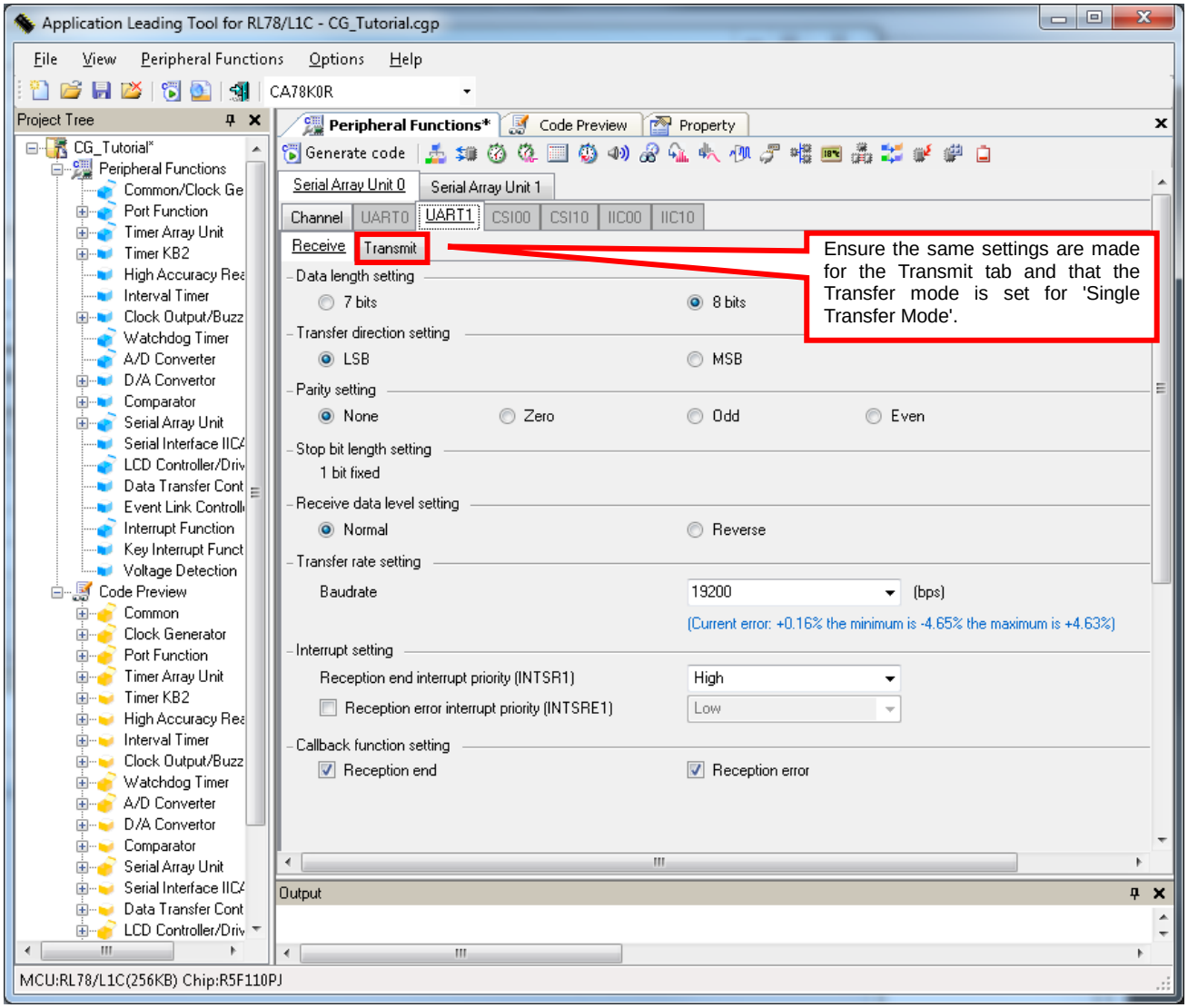


Figure 3-12 Serial Array Unit 0 tab – UART1 settings

3.3.8 Port Function

Referring to the RSK schematic, LED0 is connected to P05, LED1 is connected to P07, LED2 is connected to P41 and LED3 is connected to P42. Navigate to the 'Port Function' tab in Applilet and configure these four I/O lines as shown in Figure 3-13 and Figure 3-14 below. Ensure that the 'Output 1' tick box is selected. This ensures that the code is generated to set LEDs initially off.

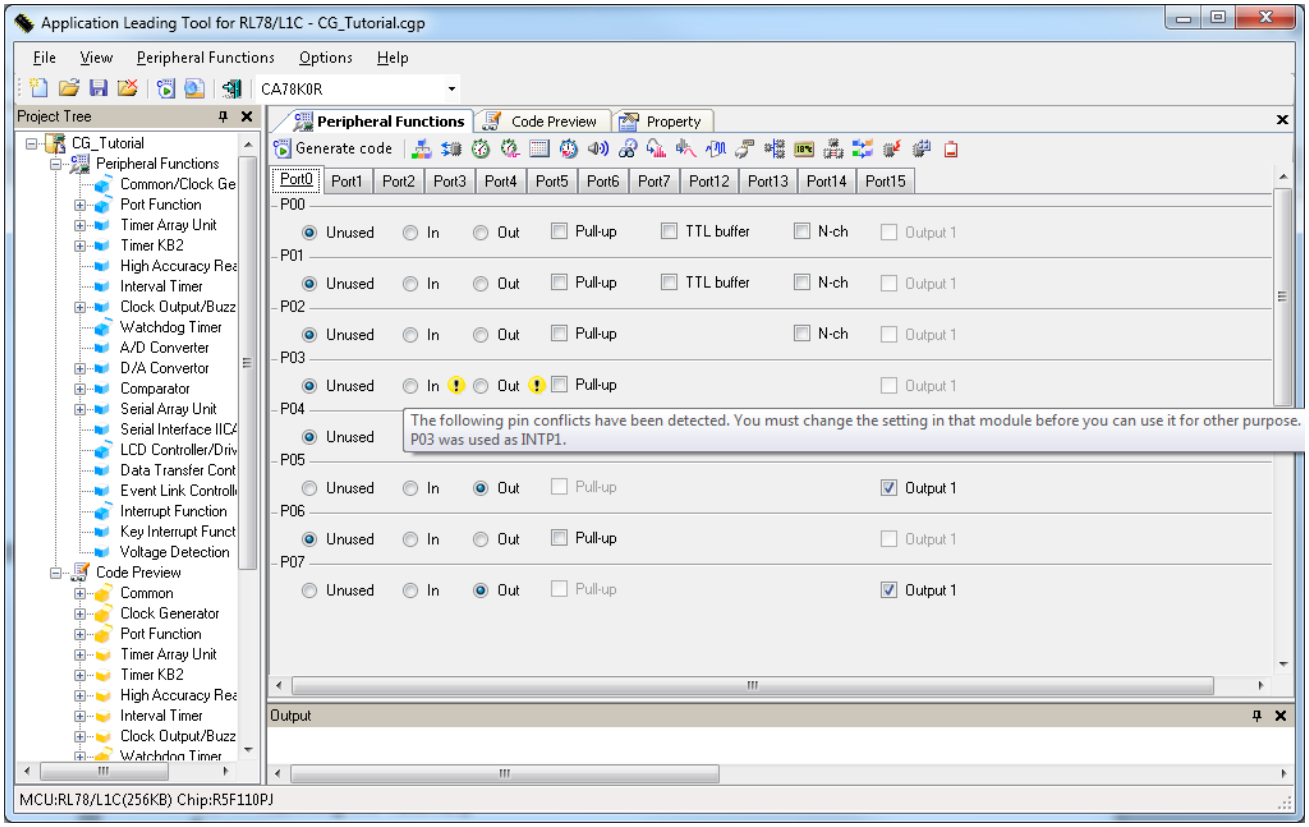


Figure 3-13 Port function tab – Port0

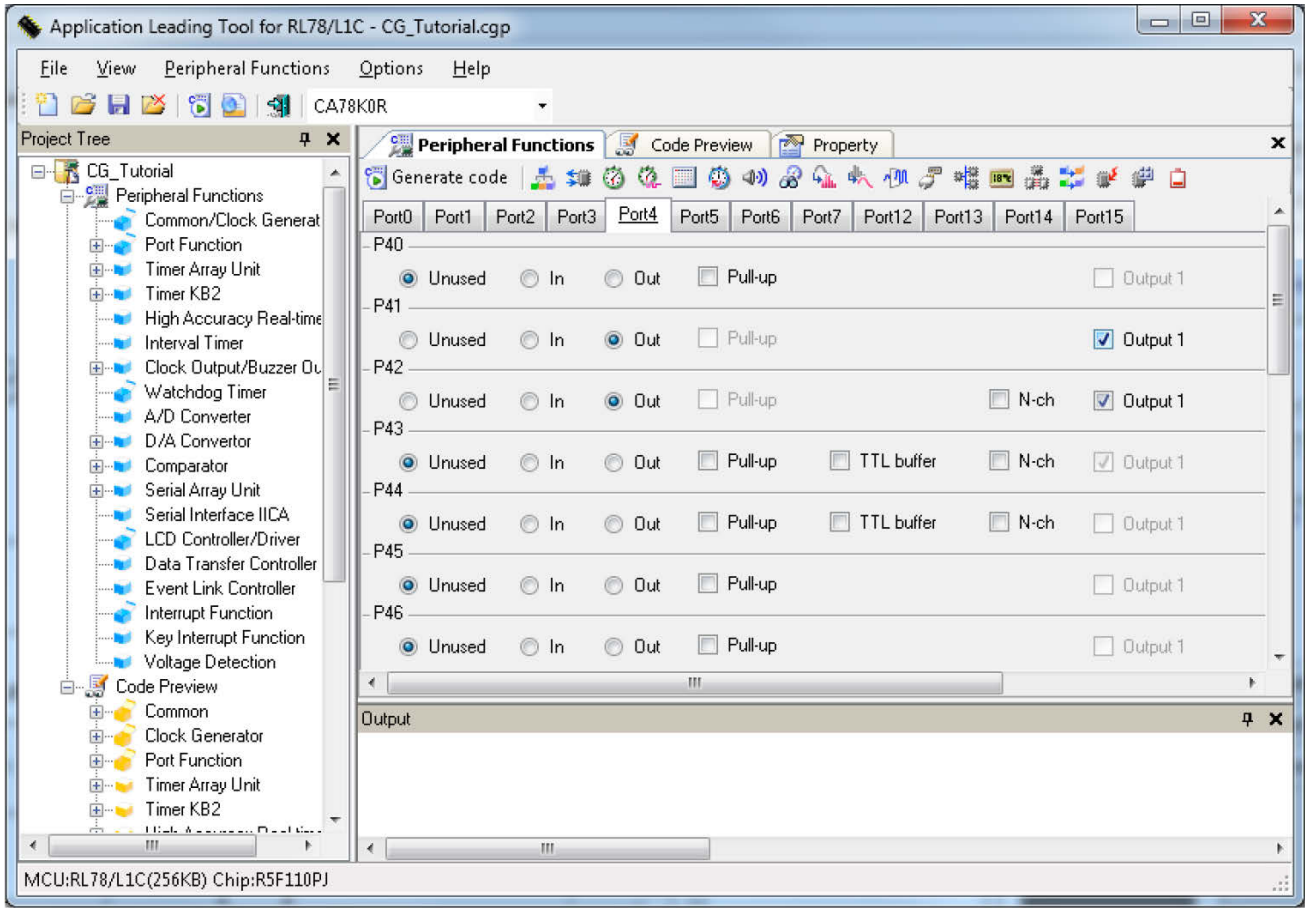


Figure 3-14 Port function tab – Port4

Peripheral function configuration is now complete. Save the project using the File -> Save menu item, then click ‘Generate Code’. The Output pane should report ‘The operation of generating file was successful, as shown Figure 3-15 below.

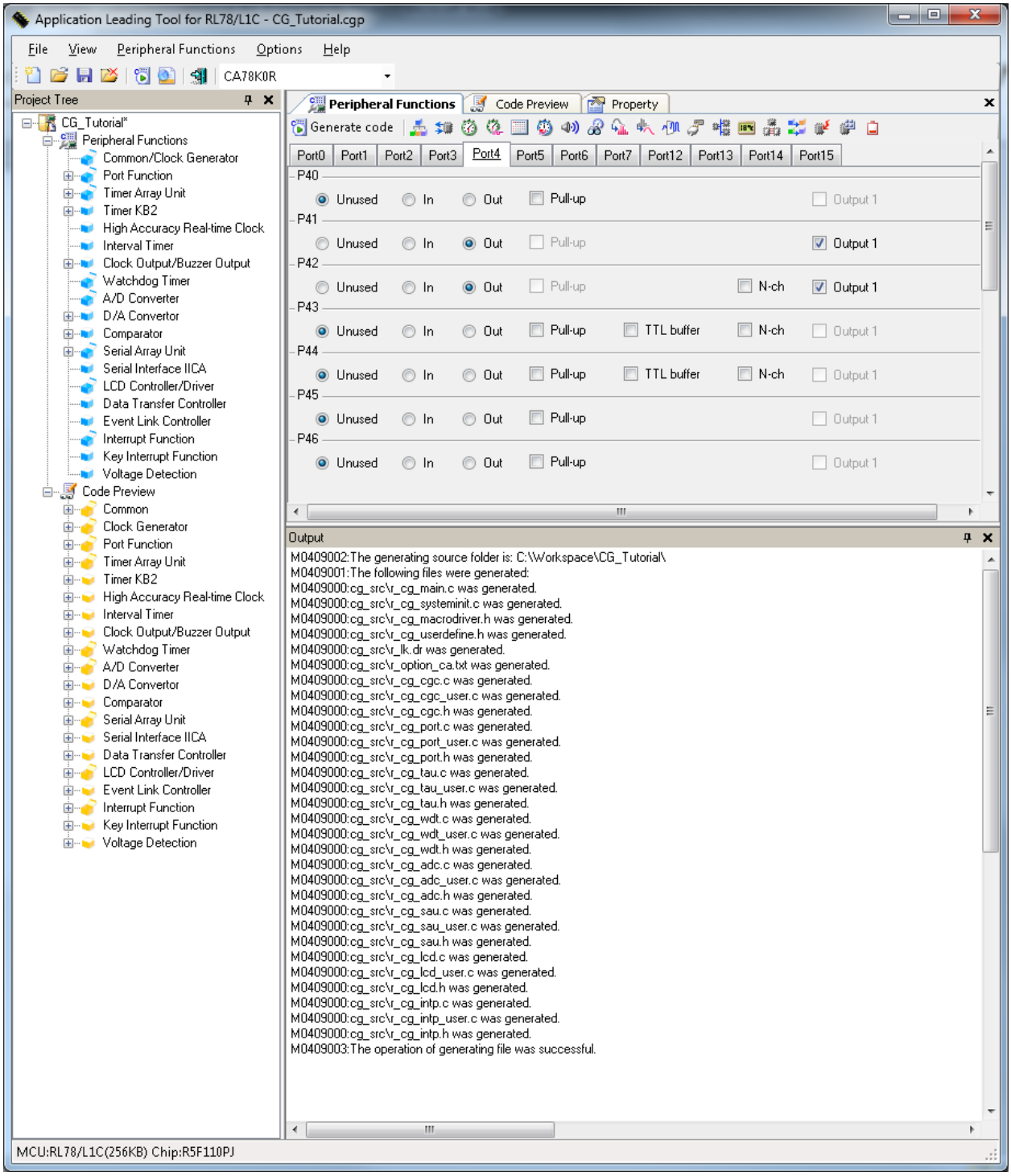
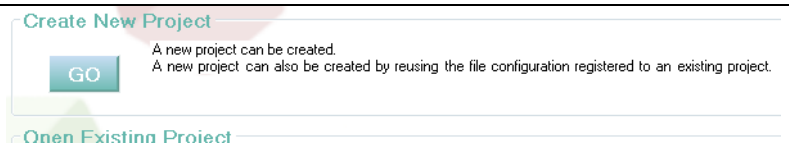


Figure 3-15 Code generation

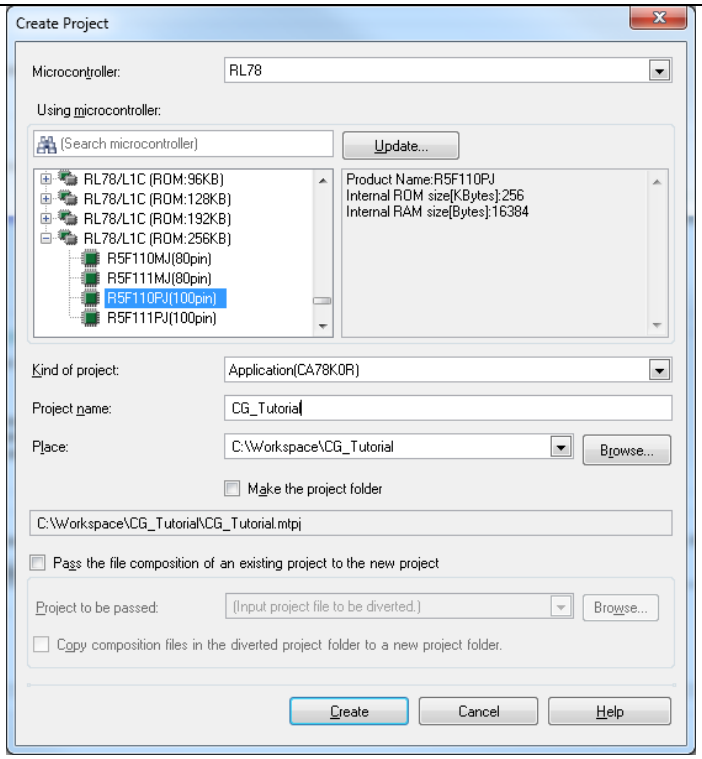
4. Importing into CubeSuite+

4.1 Starting CubeSuite+ and Importing Applilet Code

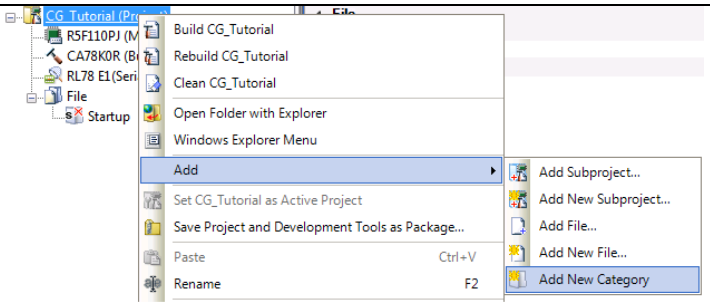
- Start CubeSuite+ by selecting it from the Start Menu. CubeSuite+ will show the Start Page. Use the 'GO' button to create a new project



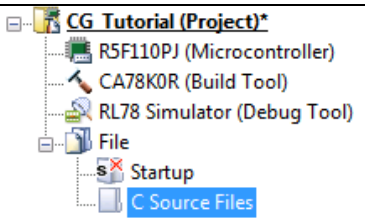
- In the 'Create Project' dialog, select 'RL78' from the 'Microcontroller' pull-down.
- In the 'Using Microcontroller' list control, scroll down to 'RL78/L1C(ROM:256KB)' and expand the tree control by clicking '+'. Select 'R5F110PJ(100pin)'.
- Choose an appropriate name and location for the project, then click 'Create'.



(1) In the Project Tree pane, right-click the CG_Tutorial project and select 'Add -> Add New Category'.

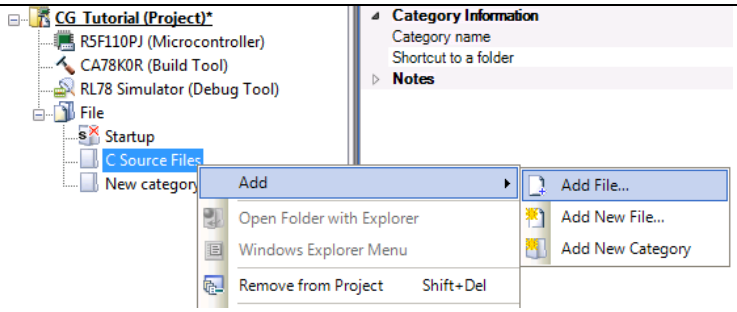


(2) Rename the newly-created 'New Category' folder to 'C Source Files'.

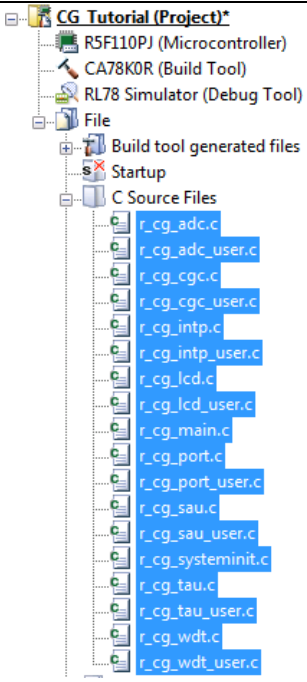


- Repeat step (1) and step (2) to create a new category folder for 'Dependencies'.

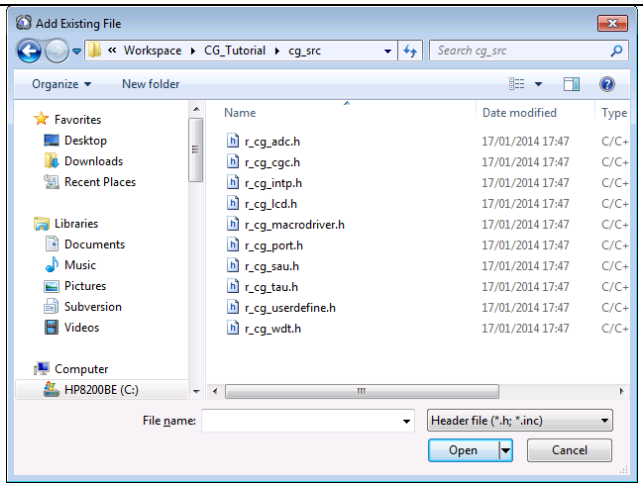
(1) Right-click the ‘C Source Files’ folder and select ‘Add -> Add File…’. Browse to the ‘cg_src’ subdirectory, select all of the ‘.c’ files and click ‘Open’.



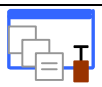
(2) CubeSuite+ will add all of the Applilet-generated C source files the ‘C Source Files’ folder.



- Repeat step (1) and step (2) for the ‘Dependencies’ folder, making sure to select ‘Header file (*.h; *.inc)’ from the file filter pull-down. Select all ‘.h’ header files and add them to the project.

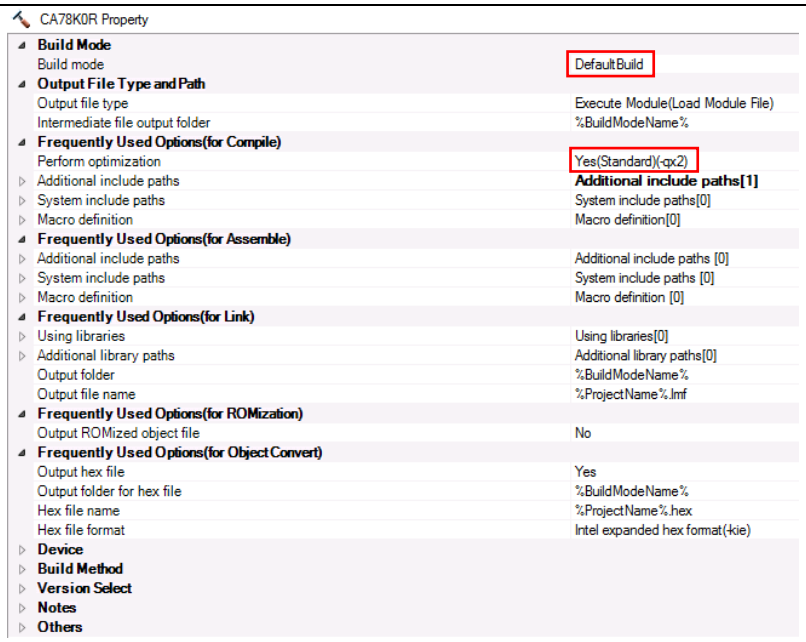


- Select ‘Build Project’ from the ‘Build’ menu, or press F7. CubeSuite+ will build the project with no errors.

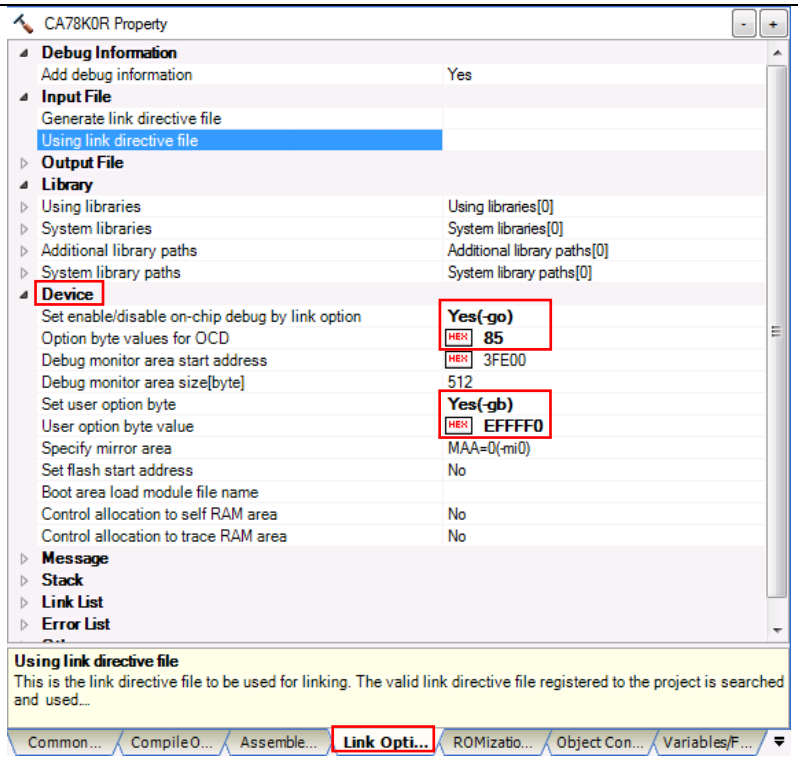


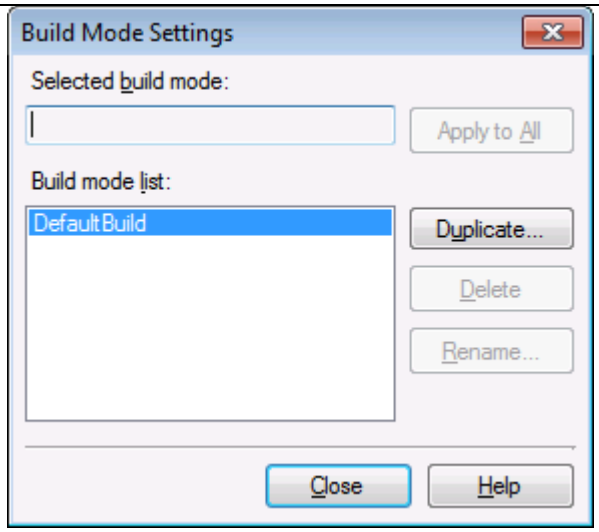
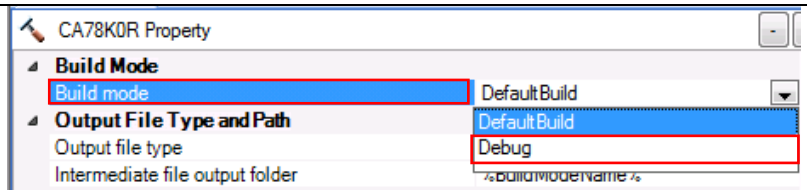
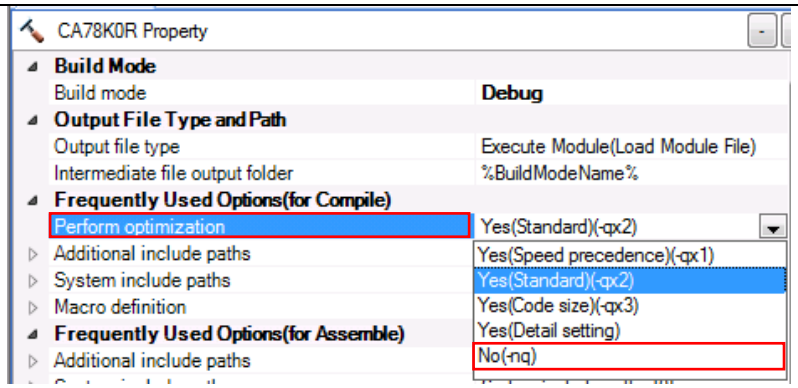
4.2 Project Settings

- In the ‘Project Tree’ pane, select ‘CA780KR (Build Tool)’. The build properties will appear in the main window.
- CubeSuite+ creates a single build configuration called ‘Default Build’ for the project. This has standard code optimisation turned on by default.

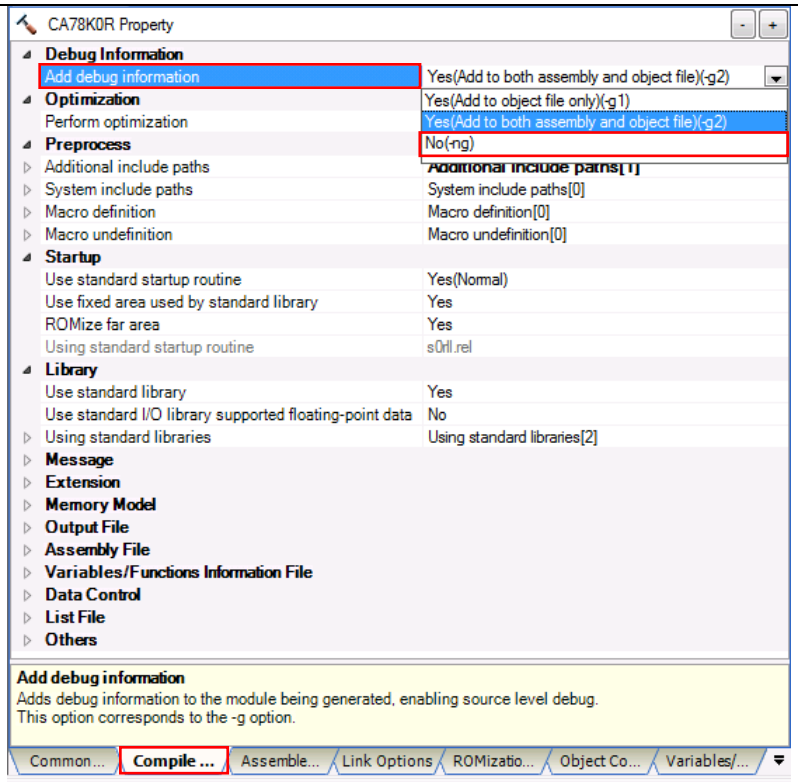


- Select the ‘Link Options’ tab at the bottom of the properties window pane. Change the ‘Device Settings’ as shown in the screenshot opposite.



• From the ‘Build’ menu, select ‘Build Mode Settings…’. Click ‘Duplicate’ and in the resulting ‘Character String Input’ dialog, enter ‘Debug’ for the name of the duplicate build mode.	
• The new ‘Debug’ build mode will be added to the Build mode list. Click ‘Close’. Now, in the main CA780KR Property window, click on the line containing ‘Build Mode’, click the pull-down arrow and select ‘Debug’ from the pull-down’.	
• For the ‘Perform Optimization’ option, select ‘No(-nq) from the pull-down. We have now created a ‘Debug’ build mode with no code optimisation and will be using the Build mode to create and debug the project.	

- All of the sample code projects contained in this RSK are configured with three Build modes; ‘DefaultBuild’, ‘Debug’ and ‘Release’. ‘Release’ is created in the same way as above; by duplicating ‘Default Build’. ‘Release’ build mode leaves code optimisation turned on and removes debug information from the output file.
- To remove debug information from the build mode, in the ‘CA780KR Property’ window, select the ‘Compile Options’ tab at the bottom of the window pane. For the ‘Add debug information’ option, select ‘No(-ng).
- From the menus, select ‘File -> Save All’ to save all project settings.



4.3 LCD Panel Code Integration

API functions for the LCD panel are provided with the RSK. Locate the files lcd_panel.h and lcd_panel.c on the RSK DVD. These files can be found in the RSKRL78L1C_Tutorial project for CubeSuite+. Copy these files into the C:\Workspace\CG_Tutorial\cg_src directory. Add these files to the ‘C Source Files’ and ‘Dependencies’ folder as shown in §4.1.

Code must be inserted in to the user code area in many files in this project, in the areas delimited by comments as follows:

```
/* Start user code for _xxxxx_. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```

Where _xxxx_ depends on the particular area of code, i.e. ‘function’ for insertion of user functions and prototypes, ‘global’ for insertion of user global variable declarations, or ‘include’ for insertion of pre-processor include directives. User code inserted inside these comment delimiters is protected from being overwritten by Applilet, should the user need to subsequently change any of the Applilet-generated code.

In the CubeSuite+ Project Tree, expand the ‘C Source Files’ folder and open the file ‘r_cg_main.c’ by double-clicking on it. Insert #include "lcd_panel.h" in between the user code delimiter comments as shown below.

```
/* Start user code for include. Do not edit comment generated here */
#include "lcd_panel.h"
/* End user code. Do not edit comment generated here */
```

Scroll down to the ‘main()’ function and insert the 2 lines of code as shown below into the beginning of the user code area of the main() function:

```
void main(void)
{
    R_MAIN_UserInit();
    /* Start user code. Do not edit comment generated here */
    /* Enable and configure LCD display. */
    Init_Display_Panel();

    /* Display the device family name on LCD.*/
    Display_Panel_String(PANEL_LCD_LINE1, " RL78");
    while (1U)
    {
        ;
    }
    /* End user code. Do not edit comment generated here */
}
```

Select ‘Build Project’ from the ‘Build’ menu, or press F7. CubeSuite+ will build the project with no errors.

4.4 Switch Code Integration

API functions for user switch control are provided with the RSK. Locate the files rskrl78l1cdef.h, switch.h and switch.c on the RSK DVD. These files can be found in the RSKRL78L1C_Tutorial project for CubeSuite+. Copy these files into the C:\Workspace\CG_Tutorial\cg_src directory. Add these files to the 'C Source Files' and 'Dependencies' folder as shown in §4.1.

The switch code uses interrupt code in the files r_cg_intp.h, r_cg_intp.c and r_cg_intp_user.c and timer code in the files r_cg_tau.h, r_cg_tau.c and r_cg_tau_user.c, as described in §3.3.2 and §3.3.4. It is necessary to provide additional user code in these files to implement the switch press/release detection and de-bouncing required by the API functions in switch.c.

4.4.1 Interrupt Code

In the CubeSuite+ Project Tree, expand the ‘Dependencies’ folder and open the file ‘r_cg_intp.h’ by double-clicking on it. Insert the following code in the user code area at the end of the file:

```
/* Function prototypes for detecting and setting the edge trigger of INTP0 */
uint8_t R_INTC0_IsFallingEdge(void);
void R_INTC0_SetFallingEdge(const uint8_t set_f_edge);
void R_INTC0_SetRisingEdge(const uint8_t set_r_edge);

/* Function prototypes for detecting and setting the edge trigger of INTP1 */
uint8_t R_INTC1_IsFallingEdge(void);
void R_INTC1_SetFallingEdge(const uint8_t set_f_edge);
void R_INTC1_SetRisingEdge(const uint8_t set_r_edge);

/* Function prototypes for detecting and setting the edge trigger of INTP2 */
uint8_t R_INTC2_IsFallingEdge(void);
void R_INTC2_SetFallingEdge(const uint8_t set_f_edge);
void R_INTC2_SetRisingEdge(const uint8_t set_r_edge);
```

Now, open the r_cg_intp.c file and insert the following code in the user code area at the end of the file:

```
/***********************************************************************************
* Function Name: R_INTC0_IsFallingEdge
* Description  : This function returns 1 if the INTP0 is set to falling edge
*                triggered, otherwise 0.
* Arguments    : None
* Return Value : None
***********************************************************************************/
uint8_t R_INTC0_IsFallingEdge (void)
{
    uint8_t falling_edge_trig = 0x0;

    if (EGN0 & _01_INTP0_EDGE_FALLING_SEL)
    {
        falling_edge_trig = 1;
    }

    return falling_edge_trig;

}
/***********************************************************************************
* End of function R_INTC0_IsFallingEdge
***********************************************************************************/

/***********************************************************************************
* Function Name: R_INTC0_SetFallingEdge
* Description  : This function sets/clears the falling edge trigger for INTP0.
* Arguments    : uint8_t set_f_edge, 1 if setting falling edge triggered, 0 if
*                clearing
* Return Value : None
***********************************************************************************/
void R_INTC0_SetFallingEdge (const uint8_t set_f_edge)
{
    if (1 == set_f_edge)
    {
        EGN0 |= _01_INTP0_EDGE_FALLING_SEL;
    }
    else
    {
        EGN0 &= (uint8_t) ~_01_INTP0_EDGE_FALLING_SEL;
    }
}
/***********************************************************************************
* End of function R_INTC0_SetFallingEdge
***********************************************************************************/

/***********************************************************************************
* Function Name: R_INTC0_SetRisingEdge
* Description  : This function sets/clear the rising edge trigger for INTP0.
* Arguments    : uint8_t set_r_edge, 1 if setting rising edge triggered, 0 if
*                clearing
* Return Value : None
***********************************************************************************/
void R_INTC0_SetRisingEdge (const uint8_t set_r_edge)
{
    if (1 == set_r_edge)
    {
        EGP0 |= _01_INTP0_EDGE_RISING_SEL;
    }
    else
    {
        EGP0 &= (uint8_t) ~_01_INTP0_EDGE_RISING_SEL;
    }
}
/***********************************************************************************
* End of function R_INTC0_SetRisingEdge
***********************************************************************************/

/***********************************************************************************
* Function Name: R_INTC1_IsFallingEdge
* Description  : This function returns 1 if the INTP1 is set to falling edge
*                triggered, otherwise 0.
* Arguments    : None
* Return Value : None
```

```
***********************************************************************************/
uint8_t R_INTC1_IsFallingEdge (void)
{
    uint8_t falling_edge_trig = 0x0;

    if (EGN0 & _02_INTP1_EDGE_FALLING_SEL)
    {
        falling_edge_trig = 1;
    }

    return falling_edge_trig;

}
/***********************************************************************************
* End of function R_INTC1_IsFallingEdge
***********************************************************************************/

/***********************************************************************************
* Function Name: R_INTC1_SetFallingEdge
* Description  : This function sets/clears the falling edge trigger for INTP1.
* Arguments    : uint8_t set_f_edge, 1 if setting falling edge triggered, 0 if
*                clearing
* Return Value : None
***********************************************************************************/
void R_INTC1_SetFallingEdge (const uint8_t set_f_edge)
{
    if (1 == set_f_edge)
    {
        EGN0 |= _02_INTP1_EDGE_FALLING_SEL;
    }
    else
    {
        EGN0 &= (uint8_t) ~_02_INTP1_EDGE_FALLING_SEL;
    }
}
/***********************************************************************************
* End of function R_INTC1_SetFallingEdge
***********************************************************************************/

/***********************************************************************************
* Function Name: R_INTC1_SetRisingEdge
* Description  : This function sets/clear the rising edge trigger for INTP1.
* Arguments    : uint8_t set_r_edge, 1 if setting rising edge triggered, 0 if
*                clearing
* Return Value : None
***********************************************************************************/
void R_INTC1_SetRisingEdge (const uint8_t set_r_edge)
{
    if (1 == set_r_edge)
    {
        EGP0 |= _02_INTP1_EDGE_RISING_SEL;
    }
    else
    {
        EGP0 &= (uint8_t) ~_02_INTP1_EDGE_RISING_SEL;
    }
}
/***********************************************************************************
* End of function R_INTC1_SetRisingEdge
***********************************************************************************/

/***********************************************************************************
* Function Name: R_INTC2_IsFallingEdge
* Description  : This function returns 1 if the INTP2 is set to falling edge
*                triggered, otherwise 0.
* Arguments    : None
* Return Value : None
***********************************************************************************/
uint8_t R_INTC2_IsFallingEdge (void)
{
    uint8_t falling_edge_trig = 0x0;

    if (EGN0 & _04_INTP2_EDGE_FALLING_SEL)
    {
        falling_edge_trig = 1;
    }
```

```
    return falling_edge_trig;

}
/***********************************************************************************
* End of function R_INTC2_IsFallingEdge
***********************************************************************************/

/***********************************************************************************
* Function Name: R_INTC2_SetFallingEdge
* Description  : This function sets/clears the falling edge trigger for INTP2.
* Arguments    : uint8_t set_f_edge, 1 if setting falling edge triggered, 0 if
*                clearing
* Return Value : None
***********************************************************************************/
void R_INTC2_SetFallingEdge (const uint8_t set_f_edge)
{
    if (1 == set_f_edge)
    {
        EGN0 |= _04_INTP2_EDGE_FALLING_SEL;
    }
    else
    {
        EGN0 &= (uint8_t) ~_04_INTP2_EDGE_FALLING_SEL;
    }
}
/***********************************************************************************
* End of function R_INTC2_SetFallingEdge
***********************************************************************************/

/***********************************************************************************
* Function Name: R_INTC2_SetRisingEdge
* Description  : This function sets/clear the rising edge trigger for INTP2.
* Arguments    : uint8_t set_r_edge, 1 if setting rising edge triggered, 0 if
*                clearing
* Return Value : None
***********************************************************************************/
void R_INTC2_SetRisingEdge (const uint8_t set_r_edge)
{
    if (1 == set_r_edge)
    {
        EGP0 |= _04_INTP2_EDGE_RISING_SEL;
    }
    else
    {
        EGP0 &= (uint8_t) ~_04_INTP2_EDGE_RISING_SEL;
    }
}
/***********************************************************************************
* End of function R_INTC2_SetRisingEdge
***********************************************************************************/
```

Open the r_cg_intp_user file.c file and insert the following code in the user code area for include near the top of the file:

```
/* Defines switch callback functions required by interrupt handlers */
#include "switch.h"
```

In the same file insert the following code in the user code area inside the function r_intc0_interrupt():

```
/* Switch 1 callback handler */
Switch1IsrCallback();

/* clear INTP0 interrupt flag */
PIF0 = 0U;
```

In the same file insert the following code in the user code area inside the function r_intc1_interrupt():

```
/* Switch 2 callback handler */
Switch2IsrCallback();

/* clear INTP1 interrupt flag */
PIF1 = 0U;
```

In the same file insert the following code in the user code area inside the function r_intc2_interrupt():

```
/* Switch 3 callback handler */
Switch3IsrCallback();

/* clear INTP2 interrupt flag */
PIF2 = 0U;
```

4.4.2 De-bounce Timer Code

Open the r_cg_tau_user.c file and insert the following code in the user code area for include near the top of the file:

```
/* Defines switch callback functions required by interrupt handlers */
#include "switch.h"
```

In the same file insert the following code in the user code area inside the function r_tau0_channel1_interrupt():

```
/* Stop this timer - we start it again in the de-bounce routines */
R_TAU0_Channel1_Stop();

/* Call the de-bounce call back routine */
SwitchDebounceIsrCallback();
```

In the same file insert the following code in the user code area inside the function r_tau0_channel2_interrupt():

```
/* Stop this timer - we start it again in the de-bounce routines */
R_TAU0_Channel2_Stop();

/* Call the de-bounce call back routine */
SwitchDebounceIsrCallback();
```

4.4.3 Main Switch and ADC Code

In the CubeSuite+ Project Tree, expand the ‘Dependencies’ folder and open the file ‘r_cg_userdefine.h’ by double-clicking on it. Insert the following code the user code area, resulting in the code shown below

```
/* Start user code for function. Do not edit comment generated here */
#define TRUE            (1)
#define FALSE           (0)

extern volatile uint8_t g_adc_trigger;
/* End user code. Do not edit comment generated here */
```

Open the file ‘r_cg_main.c’ by double-clicking on it. Insert #include "switch.h" and #include "r_cg_adc.h" in the user code area for include, resulting in the code shown below:

```
/* Start user code for include. Do not edit comment generated here */
#include "lcd_panel.h"
#include "switch.h"
#include "r_cg_adc.h"
/* End user code. Do not edit comment generated here */
```

Next add the switch module initialisation function call highlighted in the user code area inside the main() function, resulting in the code shown below:

```
void main(void)
{
    R_MAIN_UserInit();
    /* Start user code. Do not edit comment generated here */

    /* Initialise the switch module */
    Switch_Init();

    /* Enable and configure LCD display. */
    Init_Display_Panel();
```

```
    /* Display the device family name on LCD.*/
    Display_Panel_String(PANEL_LCD_LINE1, " RL78");

    while (1U)
    {
        ;
    }
    /* End user code. Do not edit comment generated here */
}
```

In the same file, insert the declarations in the user code area for global, resulting in the code shown below:

```
/* Start user code for global. Do not edit comment generated here */

/* Prototype declaration for cb_switch_press */
static void cb_switch_press (void);

/* Prototype declaration for get_adc */
static uint16_t get_adc(void);

/* Prototype declaration for lcd_display_adc */
static void lcd_display_adc (const uint16_t adc_result);

/* Variable for flagging user requested ADC conversion */
volatile uint8_t g_adc_trigger = FALSE;
```

Next add the switch module call back registration function call in the user code area inside the main() function and the code inside the while loop, resulting in the code shown below:

```
void main(void)
{
    R_MAIN_UserInit();
    /* Start user code. Do not edit comment generated here */

    /* Initialise the switch module */
    Switch_Init();

    /* Set the call back function when SW3 is pressed */
    SetSwitchPressCallback(cb_switch_press);

    /* Enable and configure LCD display. */
    Init_Display_Panel();

    /* Display the device family name on LCD.*/
    Display_Panel_String(PANEL_LCD_LINE1, " RL78");

    while (1U)
    {
        /* Wait for user requested A/D conversion flag to be set */
        if (TRUE == g_adc_trigger)
        {
            uint16_t adc_result;

            /* Call the function to perform an A/D conversion  */
            adc_result = get_adc();

            /* Display the result on the LCD */
            lcd_display_adc(adc_result);

           /* Reset the flag */
           g_adc_trigger = FALSE;
        }
    }
    /* End user code. Do not edit comment generated here */
}
```

Then add the definition for the switch call-back, get_adc() and lcd_display_adc() functions in the user code area at the end of the file, resulting in the code shown below:

```
/***********************************************************************
* Function Name : cb_switch_press
* Description   : Switch press callback function. Sets g_adc_trigger flag.
* Argument      : none
* Return value  : none
***********************************************************************/
static void cb_switch_press (void)
{
    /* Check if switch 3 was pressed */
    if (g_switch_flag & SWITCHPRESS_3)
    {
        /* set the flag indicating a user requested A/D conversion is required */
        g_adc_trigger = TRUE;

        /* Clear flag */
        g_switch_flag = 0x0;
    }
}
/***********************************************************************
* End of function cb_switch_press
***********************************************************************/

/***********************************************************************
* Function Name : get_adc
* Description   : Reads the ADC result, converts it to a string and displays
*                 it on the LCD panel.
* Argument      : none
* Return value  : uint16_t adc value
***********************************************************************/
static uint16_t get_adc (void)
{
    /* A variable to retrieve the adc result */
    uint16_t adc_result;

    /* Start a conversion */
    R_ADC_Start();

    /* Wait for the A/D conversion to complete */
    while (FALSE == g_adc_complete)
    {
        /* Wait */
    }

    /* Stop conversion */
    R_ADC_Stop();

    /* Clear ADC flag */
    g_adc_complete = FALSE;

    R_ADC_Get_Result(&adc_result);

    return adc_result;
}
/***********************************************************************
* End of function get_adc
***********************************************************************/

/***********************************************************************
* Function Name : lcd_display_adc
* Description   : Converts adc result to a string and displays
*                 it on the LCD panel.
* Argument      : uint16_t adc result
* Return value  : none
***********************************************************************/
static void lcd_display_adc (const uint16_t adc_result)
{
    /* Declare a temporary variable */
    uint8_t a;

    /* Declare temporary character string */
    char    lcd_buffer[4] = "XYZ";
```

```
    /* Convert ADC result into a character string, and store in the local.
       Casting to ensure use of correct data type. */
    a = (uint8_t)((adc_result & 0x0F00) >> 8);
    lcd_buffer[0] = (char)((a < 0x0A) ? (a + 0x30) : (a + 0x37));
    a = (uint8_t)((adc_result & 0x00F0) >> 4);
    lcd_buffer[1] = (char)((a < 0x0A) ? (a + 0x30) : (a + 0x37));
    a = (uint8_t)(adc_result & 0x000F);
    lcd_buffer[2] = (char)((a < 0x0A) ? (a + 0x30) : (a + 0x37));

    /* Display the contents of the local string lcd_buffer */
    Display_Panel_String(PANEL_LCD_LINE3, lcd_buffer);

}
/***********************************************************************
* End of function lcd_display_adc
***********************************************************************/

/* End user code. Do not edit comment generated here */
```

Open the file ‘r_cg_adc.h’ by double-clicking on it. Insert the following code in the in the user code area for function, resulting in the code shown below:

```
/* Start user code for function. Do not edit comment generated here */
/* Flag indicates when serial transmission is in progress */
extern volatile uint8_t g_adc_complete;
/* End user code. Do not edit comment generated here */
```

Open the file ‘r_cg_adc_user.c’ by double-clicking on it. Insert the following code in the in the user code area for global, resulting in the code shown below:

```
/* Start user code for global. Do not edit comment generated here */
volatile uint8_t g_adc_complete;
/* End user code. Do not edit comment generated here */
```

Insert the following code in the in the user code area of the r_adc_interrupt() function, resulting in the code shown below:

```
void r_adc_interrupt(void)
{
    /* Start user code. Do not edit comment generated here */
    g_adc_complete = TRUE;
    /* End user code. Do not edit comment generated here */
}
```

Select ‘Build Project’ from the ‘Build’ menu, or press F7. CubeSuite+ will build the project with no errors.

The project may now be run using the debugger as described in §5. When SW3 is pressed, the program will perform an A/D conversion of the voltage level on the ADPOT line and display the result on the LCD panel. Return to this point in the Tutorial to add the UART user code.

4.5 Debug Code Integration

API functions for trace debugging via the RSK serial port are provided with the RSK. Locate the files debug.h and debug.c on the RSK DVD. These files can be found in the RSKRL78L1C_Tutorial project for CubeSuite+. Copy these files into the C:\Workspace\CG_Tutorial\cg_src directory. Add these files to the 'C Source Files' and 'Dependencies' folder as shown in §4.1.

In the debug.h file, ensure the following macro definition is included:

```
/* Macro for definition of serial debug transmit function - user edits this */
#define SerialDbgWrite R_UART1_Transmit
```

4.6 UART Code Integration

4.6.1 Serial Array Unit Code

In the CubeSuite+ Project Tree, expand the 'Dependencies' folder and open the file 'r_cg_sau.h' by double-clicking on it. Insert the following code in the user code area at the end of the file:

```
/* Start user code for function. Do not edit comment generated here */

/* Function prototype for R_UART1_Transmit */
MD_STATUS R_UART1_Transmit(uint8_t * const tx_buf, const uint16_t tx_num);

/* Flag indicates when serial transmission is in progress */
extern volatile uint8_t g_uart1_tx_busy;

/* Character is used to receive key presses from PC terminal */
extern uint8_t g_rx_char;

/* End user code. Do not edit comment generated here */
```

Open the file 'r_cg_sau.c. Insert the following code in the user code area at the end of the file:

```
/* Start user code for adding. Do not edit comment generated here */
/***********************************************************************
* Function Name: R_UART1_Transmit
* Description  : This function transmits data through UART1, but first waiting
*                for the tx_busy to clear.
* Arguments    : tx_buf -
*                    transfer buffer pointer
*                tx_num -
*                    buffer size
* Return Value : status -
*                    MD_OK or MD_ARGERROR
***********************************************************************/
MD_STATUS R_UART1_Transmit (uint8_t * const tx_buf, const uint16_t tx_num)
{
    MD_STATUS status = MD_OK;

    if (tx_num < 1U)
    {
        status = MD_ARGERROR;
    }
    else
    {
        /* Wait for the g_uart1_tx_busy flag to clear, to avoid overwriting of
           the transmit buffer */
        while (g_uart1_tx_busy)
        {
            /* Wait */
        }

        /* Set the tx busy flag, this is cleared in the transmit end callback
           function */
        g_uart1_tx_busy = 1;

        /* Send the data using the R_UART1_Send function */
        R_UART1_Send(tx_buf, tx_num);
    }

    return (status);
}
/***********************************************************************
* End of function R_UART1_Transmit
***********************************************************************/

/* End user code. Do not edit comment generated here */
```

Open the file 'r_cg_sau_user.c. Insert the following code in the user area for global near the beginning of the file:

```
/* Start user code for global. Do not edit comment generated here */
/* Flag indicates when serial transmission is in progress */
volatile uint8_t g_uart1_tx_busy = 0;

/* Global used to receive a character from the PC terminal */
uint8_t g_rx_char;
/* End user code. Do not edit comment generated here */
```

In the same file, insert the following code in the user code area inside the r_uart1_callback_receiveend() function:

```
static void r_uart1_callback_receiveend(void)
{
    /* Start user code. Do not edit comment generated here */
    /* Check the contents of g_rx_char */
    if (('c' == g_rx_char) || ('C' == g_rx_char))
    {
        g_adc_trigger = TRUE;
    }
    /* Set up UART1 receive buffer and callback function again */
    R_UART1_Receive((uint8_t *)&g_rx_char, 1);
     /* End user code. Do not edit comment generated here */
}
```

In the same file, insert the following code in the user code area inside the r_uart1_callback_sendend() function:

```
static void r_uart1_callback_sendend(void)
{
    /* Start user code. Do not edit comment generated here */
    /* Clear the g_uart1_tx_busy flag, this facilitates correct serialisation of
       tranmsit strings using the R_UART1_Transmit function */
    g_uart1_tx_busy = 0;
    /* End user code. Do not edit comment generated here */
}
```

4.6.2 Main UART code

Open the file 'r_cg_main.c'. Add the following declaration to the user code area for include near the top of the file:

```
#include "debug.h"
```

Add the following declaration to the user code area for global near the top of the file:

```
/* Prototype declaration for uart_display_adc */
static void uart_display_adc(const uint8_t adc_count, const uint16_t adc_result);

/* Variable to store the A/D conversion count for user display */
uint8_t adc_count = 0;
```

Add the following highlighted code to the user code area in the main function:

```
void main(void)
{
    R_MAIN_UserInit();
    /* Start user code. Do not edit comment generated here */

    /* Initialise the switch module */
    Switch_Init();
```

```
    /* Set the call back function when SW3 is pressed */
    SetSwitchPressCallback(cb_switch_press);

    /* Enable and configure LCD display. */
    Init_Display_Panel();

    /* Display the device family name on LCD.*/
    Display_Panel_String(PANEL_LCD_LINE1, " RL78");

    /* Set up UART1 receive buffer and callback function */
    R_UART1_Receive((uint8_t *)&g_rx_char, 1);

    /* Enable UART1 operations */
    R_UART1_Start();

    while (1U)
    {
        /* Wait for user requested A/D conversion flag to be set */
        if (TRUE == g_adc_trigger)
        {
            uint16_t adc_result;

            /* Call the function to perform an A/D conversion  */
            adc_result = get_adc();

            /* Display the result on the LCD */
            lcd_display_adc(adc_result);

            /* Increment the adc_count */
            if (16 == ++adc_count)
            {
                adc_count = 0;
            }

            /* Send the result to the UART */
            uart_display_adc(adc_count, adc_result);

            /* Reset the flag */
            g_adc_trigger = FALSE;
        }
    }
    /* End user code. Do not edit comment generated here */
}
```

Then, add the following function definition in the user code area at the end of the file:

```
/***********************************************************************
* Function Name : uart_display_adc
* Description   : Converts adc result to a string and sends it to the UART1.
* Argument      : uint8_t : adc_count
*                 uint16_t: adc result
* Return value  : none
***********************************************************************/
static void uart_display_adc (const uint8_t adc_count, const uint16_t adc_result)
{
    /* Declare a temporary variable */
        char a;

    /* Declare temporary character string */
    static char uart_buffer[] = "ADC xH Value: xxxH\r\n";

    /* Convert ADC result into a character string, and store in the local.
       Casting to ensure use of correct data type. */
    a = (char)(adc_count & 0x000F);
    uart_buffer[4] = (char)((a < 0x0A) ? (a + 0x30) : (a + 0x37));
    a = (char)((adc_result & 0x0F00) >> 8);
    uart_buffer[14] = (char)((a < 0x0A) ? (a + 0x30) : (a + 0x37));
    a = (char)((adc_result & 0x00F0) >> 4);
    uart_buffer[15] = (char)((a < 0x0A) ? (a + 0x30) : (a + 0x37));
    a = (char)(adc_result & 0x000F);
    uart_buffer[16] = (char)((a < 0x0A) ? (a + 0x30) : (a + 0x37));

    /* Send the string to the UART */
    DebugPrint(uart_buffer);

}
/***********************************************************************
* End of function uart_display_adc
***********************************************************************/
```

Select ‘Build Project’ from the ‘Build’ menu, or press F7. CubeSuite+ will build the project with no errors.

The project may now be run using the debugger as described in §5. Connect the RSK serial port to a COM port on a PC and open a terminal program, such as HyperTerminal, on the PC with the same settings as for the UART (see §3.3.7). When SW3 is pressed, or when ‘c’ is sent , the program will perform an A/D conversion of the voltage level on the ADPOT line and display the result on the LCD panel and send the result to the PC terminal program via the UART. Return to this point in the Tutorial to add the LED user code.

4.7 LED Code Integration

Open the file ‘r_cg_main.c’. Add the following declaration to the user code area for include near the top of the file:

```
#include "rskrl78l1cdef.h"
```

Add the following declaration to the user code area for global near the top of the file:

```
/* Prototype declaration for led_display_count */
static void led_display_count(const uint8_t count);
```

Add the following highlighted code to the user code area in the main function:

```
void main(void)
{
    R_MAIN_UserInit();
    /* Start user code. Do not edit comment generated here */

    /* Initialise the switch module */
    Switch_Init();

    /* Set the call back function when SW3 is pressed */
    SetSwitchPressCallback(cb_switch_press);
```

```
    /* Enable and configure LCD display. */
    Init_Display_Panel();

    /* Display the device family name on LCD.*/
    Display_Panel_String(PANEL_LCD_LINE1, " RL78");

    /* Set up UART1 receive buffer and callback function */
    R_UART1_Receive((uint8_t *)&g_rx_char, 1);

    /* Enable UART1 operations */
    R_UART1_Start();

    while (1U)
    {
        /* Wait for user requested A/D conversion flag to be set */
        if (TRUE == g_adc_trigger)
        {
            uint16_t adc_result;

            /* Call the function to perform an A/D conversion  */
            adc_result = get_adc();


            /* Display the result on the LCD */
            lcd_display_adc(adc_result);

            /* Increment the adc_count and display using the LEDs */
            if (16 == ++adc_count)
            {
                adc_count = 0;
            }
            led_display_count(adc_count);

            /* Send the result to the UART */
            uart_display_adc(adc_count, adc_result);

            /* Reset the flag */
            g_adc_trigger = FALSE;
        }
    }
    /* End user code. Do not edit comment generated here */
}
```

Then, add the following function definition in the user code area at the end of the file:

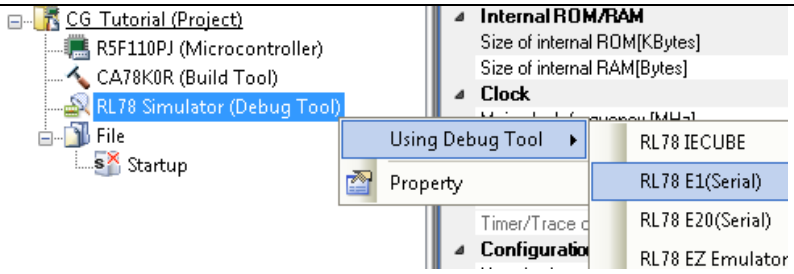
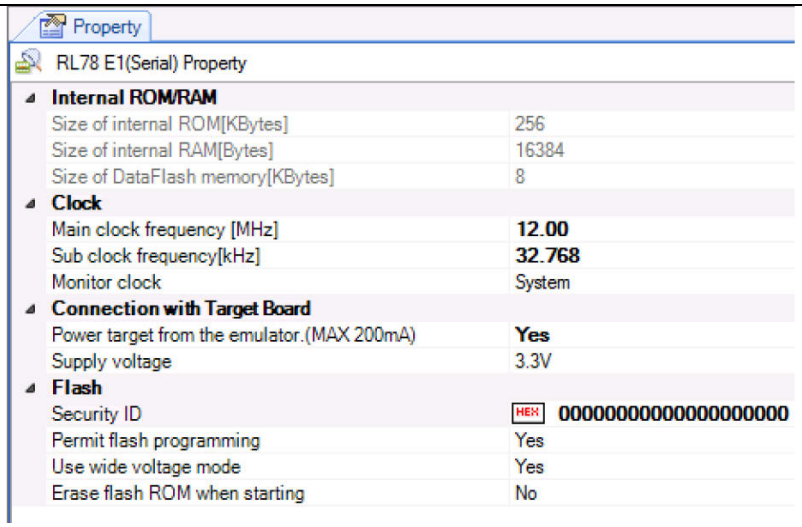
```
/***********************************************************************
* Function Name : led_display_count
* Description   : Converts count to binary and displays on 4 LEDS0-3
* Argument      : uint8_t count
* Return value  : none
***********************************************************************/
static void led_display_count (const uint8_t count)
{
    /* Set LEDs according to lower nibble of count parameter */
    LED0 = (count & 0x01) ? LED_ON : LED_OFF;
    LED1 = (count & 0x02) ? LED_ON : LED_OFF;
    LED2 = (count & 0x04) ? LED_ON : LED_OFF;
    LED3 = (count & 0x08) ? LED_ON : LED_OFF;

}
/***********************************************************************
* End of function led_display_count
***********************************************************************/
```

Select ‘Build Project’ from the ‘Build’ menu, or press F7. CubeSuite+ will build the project with no errors.

The project may now be run using the debugger as described in §5. The code will perform the same but now the LEDs will display the adc_count in binary form.

5. Debugging the Project

• In the 'Project Tree' pane, right-click the 'RL78 Simulator (Debug Tool)'. Select 'Using Debug Tool -> RL78 E1(Serial)'.	
• Double-click 'RL78 E1(Serial) (Debug Tool)' to display the debugger tool properties. Under 'Clock', change the main clock frequency to 12 MHz and the Sub clock frequency to 32.768 KHz. • Under 'Connection with Target Board' change 'Power target from the emulator.(MAX 200mA)' to 'Yes'. • All other settings can remain at their defaults.	
• Connect the E1 to the PC and the RSK E1 connector. From the 'Debug' menu select 'Download' to start the debug session and download code to the target.	

6.Running the Code Generator Tutorial

6.1 Running the Tutorial

Once the program has been downloaded onto the RSK device, the program can be executed. Click the 'Go' button or press F5 to begin the program from the current program counter position. It is recommended that you run through the program once first, and then continue to the Tutorial manual to review the code.	

7.Additional Information

Technical Support
For details on how to use CubeSuite+, refer to the manual available on the DVD or from the web site.

For information about the RL78/L1C series microcontrollers refer to the RL78/L1C Group Hardware Manual.

For information about the RL78 assembly language, refer to the RL78 Series Software Manual.

Technical Contact Details

***Please refer to the contact details listed in section 9 of the* "Quick Start Guide"**

General information on Renesas microcontrollers can be found on the Renesas website at:
http://www.renesas.com/

Trademarks
All brand or product names used in this manual are trademarks or registered trademarks of their respective companies or organisations.

Copyright

REVISION HISTORY	RSK RL78L1C Tutorial Manual

Rev.	Date	Description	
		Page	Summary
1.00	Jan 17, 2014	—	First Edition issued
1.01	Mar 19, 2014	—	[2. List of Abbreviations and Acronyms] was updated.
		37	Directory information of Section 4.5 was fixed.
			Folder information of Section 4.6.1 was fixed.
1.02	Apr 04, 2014	—	[2. List of Abbreviations and Acronyms] was updated.
		—	[Table of Contents] was updated.
		21	Additional explanation was added into the Figure 3-12.
		25	Explanation about Category creation was updated.
		25 to 29, 43, 44	Frames were added to some explanations and figures.
		26	Explanations about File addition and Build were updated.
		27 to 29	Highlight frames were added to some figures.
		29 to 42	Highlight frames were added to text part an addition or ensuring.
		30	Explanations about the file addition method to category folders of [4.4 Switch Code Integration] were updated.
		37	Explanations about the file addition method to category folders of [4.5 Debug Code Integration] were updated.

Renesas Starter Kit Manual: Tutorial Manual

Publication Date: Rev. 1.02 Apr 04, 2014

Published by: Renesas Electronics Corporation

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.
2801 Scott Boulevard Santa Clara, CA 95050-2549, 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.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

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: +852-2265-6688, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

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.
12F., 234 Teheran-ro, Gangnam-Ku, Seoul, 135-920, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 3.0

RL78/L1C Group

Renesas Electronics Corporation

R20UT2887EG0102