User's Manual

32

RX210 Group (B Mask)

Renesas Starter Kit Software Manual

RENESAS MCU
RX Family / RX200 Series

Renesas Electronics
www.renesas.com

Rev. 1.00 May 2013

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 is 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 the RSK software samples and integration with the Renesas Peripheral Driver Library (RPDL). It is intended for users designing sample code on the RSK platform, using the many different incorporated peripheral devices.

The manual comprises of an overview of each software sample with a description of how the sample works and which RPDL API functions achieve the sample functionality, but does not intend to be a complete guide to the RPDL API. Further details regarding setting up the RSK and development environment can be found in the tutorial manual.

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 this RSK and the RX210 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.	RSKRX210B User's Manual	R20UT2604EG
Software Manual	Describes the functionality of the sample code, and its interaction with the Renesas Peripheral Driver Library (RPDL)	RSKRX210B Software Manual	R20UT2607EG
Tutorial	Provides a guide to setting up RSK environment, running sample code and debugging programs.	RSKRX210B Tutorial Manual	R20UT2605EG
Quick Start Guide	Provides simple instructions to setup the RSK and run the first sample, on a single A4 sheet.	RSKRX210B Quick Start Guide	R20UT2606EG
Schematics	Full detail circuit schematics of the RSK.	RSKRX210B Schematics	R20UT2603EG
Hardware Manual	Provides technical details of the RX210 microcontroller.	RX210 Group Hardware Manual	R01UH0037EJ

2. List of Abbreviations and Acronyms

Abbreviation	Full Form
CPU	Central Processing Unit
DOC	Data Operation Circuit
ELC	Event Link Controller
MCU	Microcontroller Unit
SCI	Serial Communication Interface
SPI	Serial Peripheral Interface
PWM	Pulse Width Modulation
DMAC	Direct Memory Access Controller
WDT	Watchdog Timer
RTC	Real Time Clock
DTC	Data Transfer Controller
IIC	Philips™ Inter-Integrated Circuit Connection Bus
CRC	Cyclic Redundancy Check

Table of Contents

1. Overview

1.1 Purpose

RSK is an evaluation tool for Renesas microcontrollers. This manual explains the operation of the sample code provided, and its interaction with the Renesas Peripheral Driver Library (RPDL). The Renesas Peripheral Driver Library (hereinafter "this library" or RPDL) is based upon a unified API (Application Programming Interface) for the microcontrollers made by Renesas Electronics Corporation.

This manual is not intended to be a tutorial on using RPDL, or how RPDL works – it simply aims to explain to the reader how the RPDL was used to create the sample code supplied with the RSK. For further information regarding RPDL, refer to the RPDL API User's Manual supplied with the RSK. Alternatively, visit the PDG (Peripheral Driver Group) section of the Renesas website:

http://www.renesas.com/pdg

1.2 Note Regarding Document

This document explains by text and diagrams the functionality of the sample code and its interaction with the Renesas Peripheral Driver Library (RPDL).

This manual aims to be as clear as possible, by matching the reference sample code as closely as possible. There may be some cases however where the function names in the code differ slightly this document. This does not change its functionality as described in this manual.

2. RSK Sample Code Concept

2.1 Sample Code Structure

The basic structure of all RSK sample code is shown in Figure 2-1 below. The first two functions, 'PowerOn_ResetPC' and 'HardwareSetup', configure the MCU before the main program code executes.

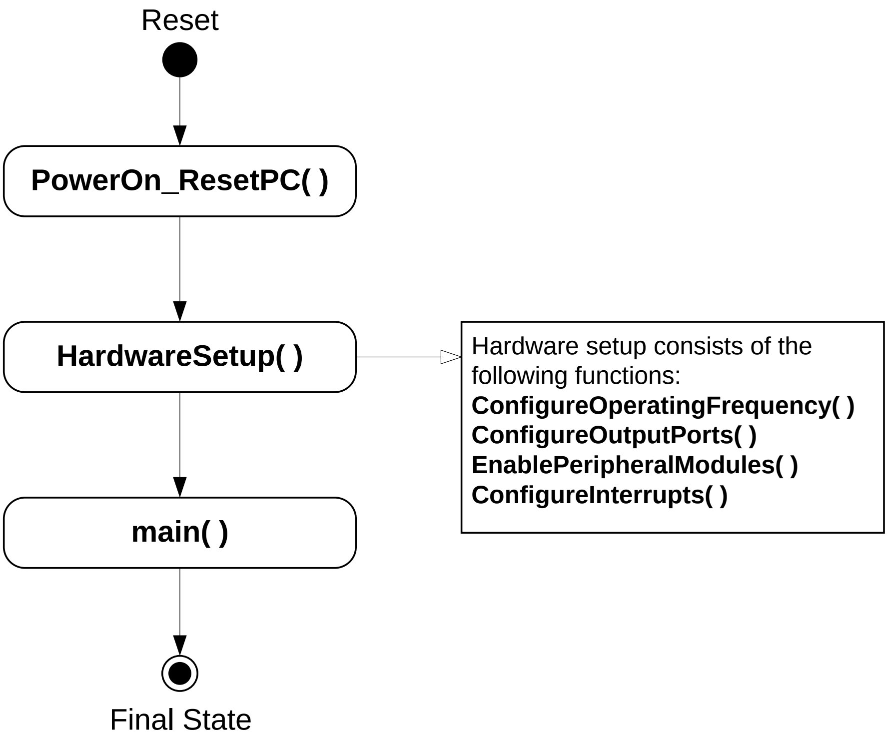

Figure 2-1: Sample Code Structure

The purpose of the functions included in the 'HardwareSetup' function are detailed in Table 2-1 below.

Function Name	Purpose	RPDL Functions Used
ConfigureOperatingFrequency	Initialises the main MCU, bus and peripheral clocks; as well as any real-time clocks and PLL settings.	R_CGC_Set R_CMT_CreateOneShot R_CGC_Control
ConfigureOutputPorts	Configures the MCU port pins as inputs or outputs, depending on the devices on the RSK and the intended function of the sample code. Also sets some pins to suitable initial logic levels.	R_IO_PORT_Set R_IO_PORT_Write
EnablePeripheralModules	Enables or disables peripheral modules on the MCU not controlled by RPDL, as required by the sample code.	Non-RPDL functions only*
ConfigureInterrupts	Configures interrupts external hardware interrupts required by the sample code.	R_INTC_CreateExtInterrupt** R_INTC_CreateFastInterrupt**

Table 2-1: Hardware Setup Functions

*RPDL functions cannot be used to manually enable/disable MCU peripherals, as this is controlled with the Create/Destroy functions for each RPDL group; therefore RPDL functions are not required in this section.

** RPDL functions indirectly called by the function ConfigureInterrupts.

3. Tutorial Samples

3.1 Tutorial

The sample code in this section is basic tutorial code; used to demonstrate basic usage of the RSK and help the user to begin writing their own basic sample code.

3.1.1 Description

The tutorial sample code demonstrates basic usage of the debugger and RSK hardware, and is common to all RSKs. This sample is supplied programmed onto the MCU, and executes out of the box when power is applied.

The sample calls three main functions to demonstrate port pin control, interrupt usage and C variable initialisation. These functions are shown in **Figure 3-1** below.

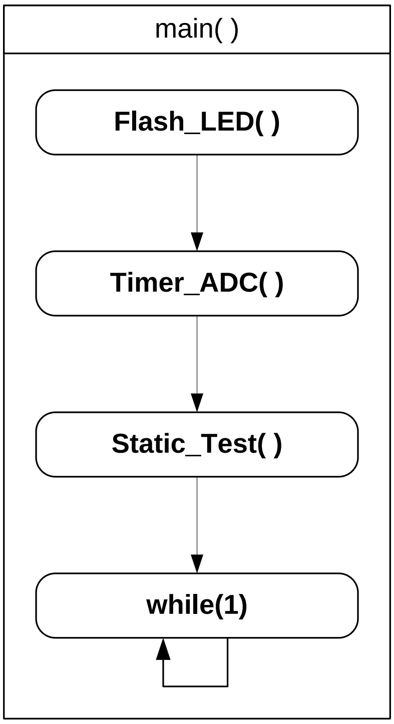

Figure 3-1: Tutorial Sample Flow

3.1.2 Operation

① The tutorial code initialises the LCD module, and displays ‘Renesas’ on the first line of the LCD, and the name of the MCU on the second line.

② The tutorial code calls the Flash_LED function which creates a timer interrupt, with callback function CB_TimerLED, to toggle the LEDs repeatedly and waits in a loop until either a switch is pressed or the LEDs flash 200 times.

③ The tutorial then calls the Timer_ADC function which configures the ADC unit, and a timer unit to periodically trigger an ADC conversion. The ADC unit is configured to call the function CB_ADConversion, after each AD conversion completes. The timer is configured to call the function CB_Timer.

④ The callback function CB_ADConversion is triggered by ADC interrupts. It fetches the ADC result, and uses it to set a new timer period. The callback function also toggles the user LEDs.

⑤ After calling Timer_ADC and setting up the timer & ADC interrupts, the tutorial calls the Static_Test function which runs in parallel to the callback functions in step 4. This function displays the string ‘STATIC’ which is fetched from a static variable and is replaced, letter by letter, by the string ‘TESTTEST’. The LCD then reverts to the original display on completion.

⑥ After the Static_Test function completes, the MCU waits in an infinite while loop; and is periodically interrupted with the callback function in step 4.

3.1.3 Sequence Diagram

Figure 3-2 below shows the program execution flow of the tutorial sample.

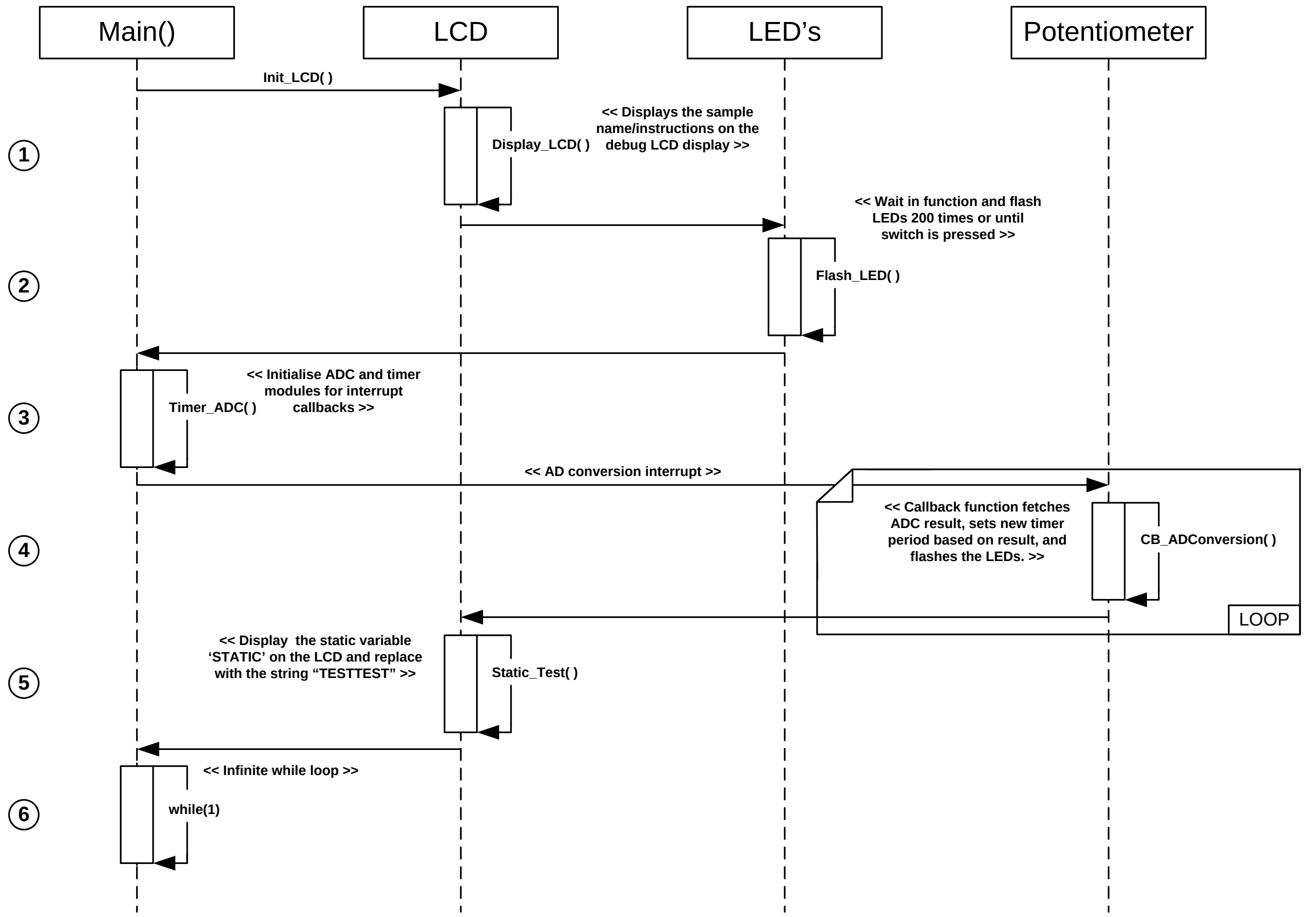

Figure 3-2: Tutorial Sequence Diagram

3.1.4 RPDL Integration

Table 3-1 below details the RPDL functions used in each sample code function shown in the sequence diagram.

Function	RPDL Function
Flash_LED	R_CMT_Create
	R_CMT_Destroy
Start_Timer	R_CMT_Create
Start_ADC	R_ADC_12_Set
	R_ADC_12_CreateUnit
	R_ADC_12_CreateChannel
	R_ADC_12_Control
CB_ADConversion	R_ADC_12_Read
	R_CMT_Control

Table 3-1: Tutorial Sample RPDL Integration

3.2 Application

3.2.1 Description

The application sample is intended as a starting platform for the user to write their own code. The sample includes all the necessary initialisation code and configuration settings from previous samples. The main( ) function contains no sample code, and performs no additional functionality.

For more information regarding the hardware initialisation performed before the main( ) function starts, refer back to §2.

4. Peripheral Samples

The sample code in this section provides examples of initialisation and usage of some of the MCU's peripheral modules. The sample code also provides examples of how to debug MCU peripherals.

4.1 ADC_Repeat

This sample code demonstrates usage of the on-chip analogue to digital converter (ADC), in scan repeat mode. The sample configures the ADC to repeatedly take readings of the potentiometer voltage (RV1). The sample then updates the conversion value displayed on the LCD, by periodic interrupts from the timer module.

4.1.1 Operation

1. The sample first initialises the debug LCD, and displays the sample name/instructions.

2. It then calls the function Init_ADC12Repeat, to initialise the ADC module to operate in repeat mode – the ADC unit will now continuously take readings. The function also initialises an interval timer interrupt, which calls the CB_Timer_ADC12Repeat callback function.

3. The sample then enters an infinite while loop. The timer interrupt function CB_Timer_ADC12Repeat is called every interval, and fetches the current AD conversion result. The function then displays the result onto the debug LCD. The callback function displays a different analogue input reading each time it is called.

4.1.2 Sequence Diagram

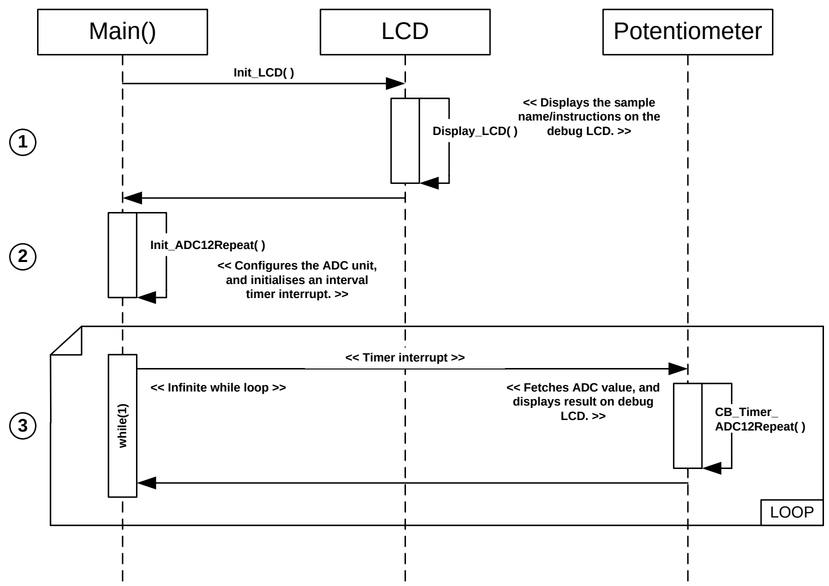

Figure 4-1: ADC_Repeat Sequence Diagram

4.1.3 RPDL Integration

Function	RPDL Function
Init_ADC12Repeat	R_ADC_12_Set
	R_ADC_12_CreateUnit
	R_ADC_12_CreateChannel
	R_CMT_Create
	R_ADC_12_Control
CB_Timer_ADC12Repeat	R_ADC_12_Read

Table 4-1: ADC_Repeat Sample RPDL Integration

4.2 Analog_Compare

This sample code demonstrates usage of the on-chip analog comparator module. The sample configures a comparator to compare the voltage with a set reference voltage.

4.2.1 Operation

This sample may require hardware modifications in order to operate. Refer to the sample code instructions for further information before proceeding.

1. The sample first initialises the debug LCD, and displays the sample name/instructions.

2. It then calls the function Init_AnalogCompare, which configures the comparator unit. The unit is configured to execute the callback function CB_Comparator_AnalogCompare function when the input voltage to comparator is higher than the reference voltage.

3. The sample then enters an infinite while loop. When the input voltage exceeds the reference, a comparator interrupt is generated which calls the function CB_Comparator_AnalogCompare. This function sets the LCD and user LEDs to indicate that the input voltage is higher than the reference voltage. It also reconfigures the comparator interrupt to trigger when the input goes below the reference voltage.

4. When the input voltage goes below the reference voltage, the comparator generates another interrupt and calls the CB_Comparator_AnalogCompare. The function sets the LCD and user LEDs to indicate that the input is lower than the reference voltage. The function also reconfigures the comparator interrupt to trigger when the inputs goes above the reference voltage. When the input goes high again, the sample repeats steps 3 & 4.

4.2.2 Sequence Diagram

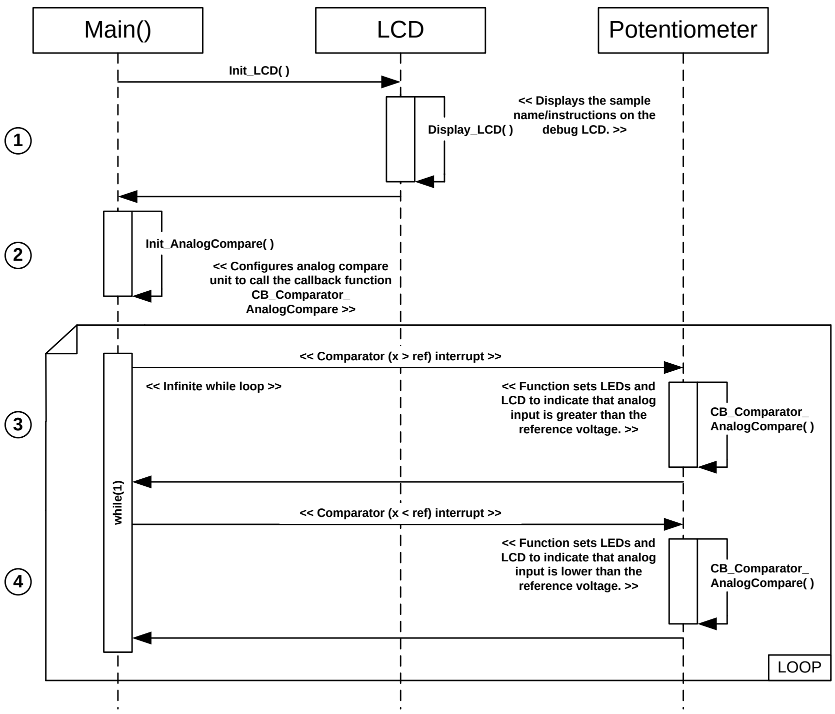

Figure 4-2: Analog_Compare Sequence Diagram

4.2.3 RPDL Integration

Function	RPDL Function
Init_AnalogCompare	R_CPA_Create
CB_Comparator_AnalogCompare	R_CPA_GetStatus
	R_IO_PORT_Write
	R_CMT_CreateOneShot

Table 4-2: Analog Compare Sample RPDL Integration

4.3 Async_Serial

This sample code demonstrates an asynchronous serial communication using the on-chip serial interface module.

4.3.1 Operation

➔ Before starting the sample, the user should connect the RSK to a PC via an RS-232 cable and start the terminal program (refer to the instructions in the sample code comments).

① The sample first initialises the debug LCD, and displays the sample name/instructions.

② The Init_Async function is called, which initialises the SCI unit to operate in asynchronous mode with the settings detailed in the sample code comments. The function also configures a SCI receive interrupt, which generates an interrupt every time data is received in the SCI unit. Finally a timer unit is configured, which generates interval interrupts, used to call the CB_Timer_Async callback function.

③ The sample then enters an infinite while loop. The callback function CB_Timer_Async is called with every timer interval interrupt. The function checks the status of the gSCI_Flag, and continues to transmit an incrementing number over SCI with the flag is true. The function exits without transmitting if the flag is false. Whilst transmitting, LED0 will remain on and LED1 off.

④ The callback function CB_SCIReceive_Async is called when data is received on the SCI unit. When a user enters a character in the terminal program, the callback function is called. The function fetches the keypress, and if the character is a ‘z’ it sets the gSCI_Flag variable to false. On all other keypresses, it sets the flag to true. Once ‘z’ has been pressed, LED1 will turn on and LED0 remain off until another key is pressed to resume transmission.

4.3.2 Sequence Diagram

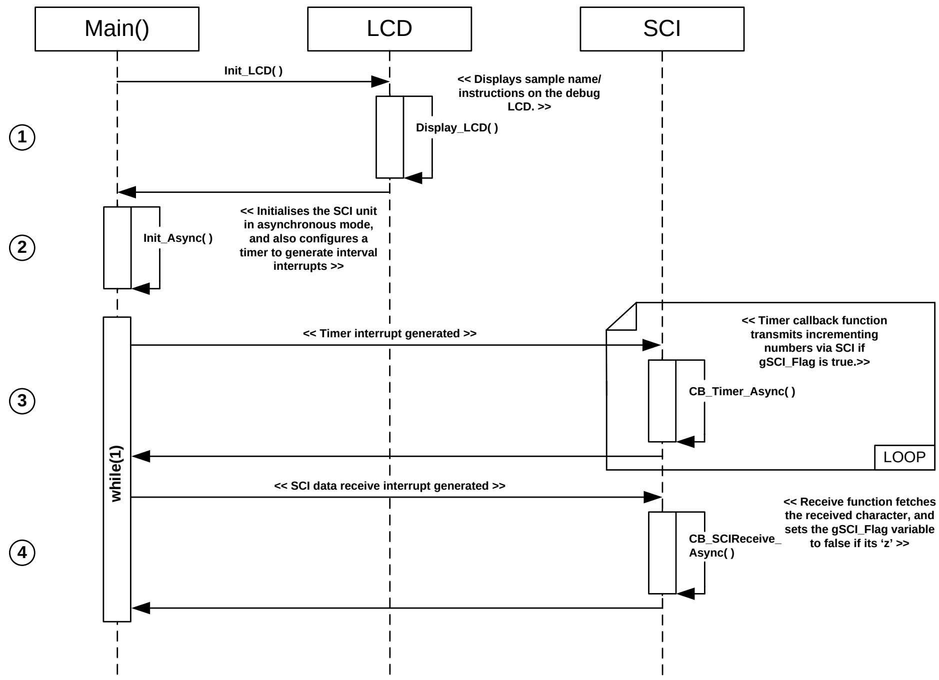

Figure 4-3: Async_Serial Sequence Diagram

4.3.3 RPDL Integration

Function	RPDL Function
Init_Async	R_SCI_Create
	R_SCI_Receive
	R_SCI_Send
	R_SCI_Set
	R_CMT_Create
Transmit_Async	R_SCI_Send
	R_IO_PORT_Write
CB_Timer_Async	R_SCI_GetStatus
CB_SCIReceive_Async	R_IO_PORT_Write
	R_SCI_Receive

Table 4-3: Async_Serial Sample RPDL Integration

4.4 CRC

4.4.1 Description

This sample demonstrates the CRC unit, by echoing typed characters from the SCI terminal with a corresponding checksum.

4.4.2 Operation

➔ Before starting this sample, the user should connect the RSK to the PC via an RS232 serial cable and run a suitable terminal program (see instructions in sample code comments).

① The sample first initialises the debug LCD, and displays the sample name/instructions.

② The sample then calls the function Init_CRC, which configures the CRC unit to produce checksums, and the SCI unit for asynchronous operation to and from the PC terminal.

③ The sample then enters an infinite while loop, and the rest of the sample's functionality is performed in interrupts. When the user presses a key in the terminal, the SCI interrupt executes the callback function CB_SCIReceive_CRC. This function takes the received character and calls the function Calculate_CRC to generate a checksum.

④ The function Calculate_CRC inserts the received character into the CRC registers, and fetches the calculated checksum, and returns it to the CB_SCIReceive_CRC function.

⑤ The sample returns from the Calculate_CRC function to the callback function and writes a string containing the received character and its checksum to the terminal. The sample then returns to the infinite while loop and waits until a key is entered into the terminal again.

4.4.3 Sequence Diagram

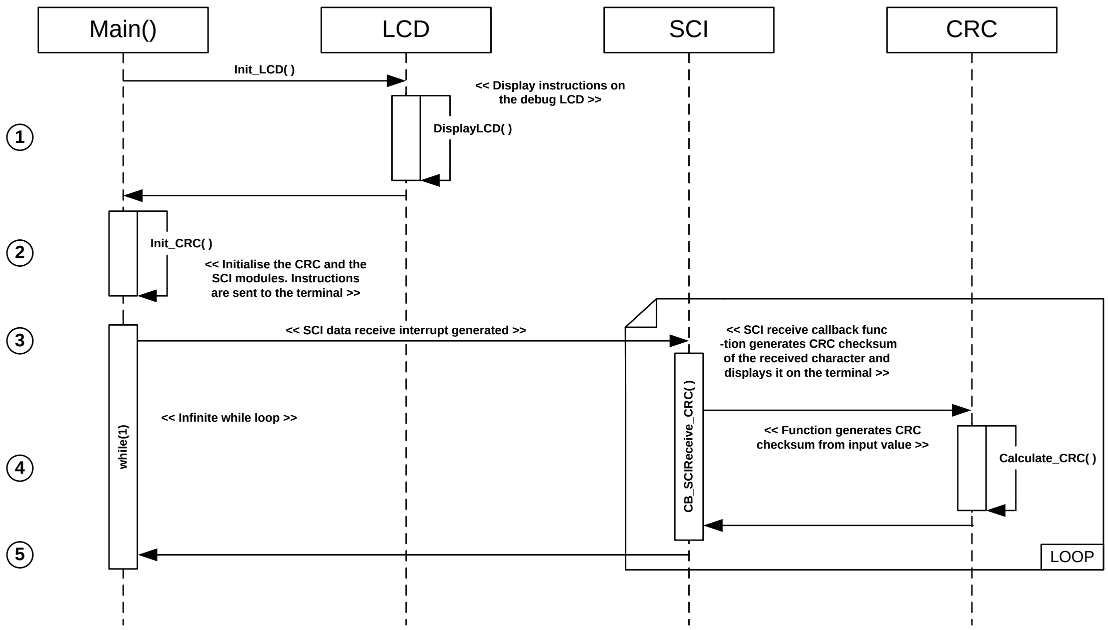

Figure 4-4: **CRC Sequence Diagram**

4.4.4 RPDL Integration

Function	RPDL Function
Init_CRC	R_CRC_Create
	R_SCI_Create
	R_SCI_Set
	R_SCI_Receive
	R_SCI_Send
CB_SCIReceive_CRC	R_SCI_GetStatus
	R_SCI_Send
	R_SCI_Receive
Calculate_CRC	R_CRC_Write
	R_CRC_Read

Table 4-4: CRC Sample RPDL Integration

4.5 DMAC

This sample code configures the DMAC unit to perform a data transfer to the global variable, gDMA_DataBuff.

4.5.1 Operation

1. The sample initialises the debug LCD and displays instructions on the screen.
2. The function Init_DMAC is called to configure a DMAC channel for consecutive data transfers. The transfer mode is set to automatically increment the data destination address after each transfer. A callback function CB_DMACTransferEnd_DMAC is configured to be called on completion of all transfers The DMAC is enabled and the transfer operation is started before the sample enters an infinite while loop.
3. On completion of all data transfers, the CB_DMACTransferEnd_DMAC callback function is called and turns on LED1 to indicate the operation has ended.

4.5.2 Sequence Diagram

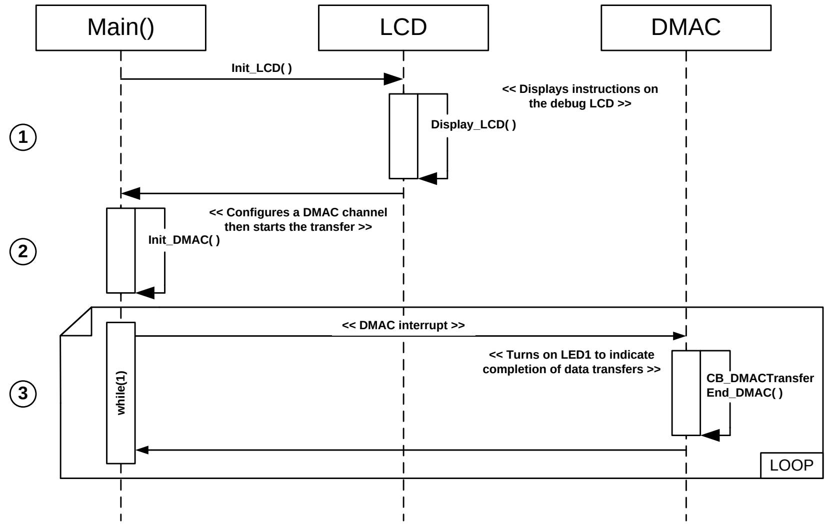

Figure 4-5: DMAC Sequence Diagram

4.5.3 RPDL Integration

Function	RPDL Function
Init_DMAC	R_DMAC_Create
	R_DMAC_Control
	R_INTC_Modify
CB_DMACTransferEnd_DMAC	R_DMAC_GetStatus
	R_IO_PORT_Write
	R_DMAC_Control

Table 4-5: DMAC Sample RPDL Integration

4.6 Data Operation Circuit (DOC)

This sample code demonstrates usage of the data operation circuit (DOC), by comparing a user input from a PC terminal, via SCI, to a set reference integer and reporting the comparison result back to the PC terminal.

4.6.1 Operation

- Before starting the sample, the user should connect the RSK to a PC via an RS-232 cable and start the terminal program (refer to the instructions in the sample code comments).

1. The sample first initialises the debug LCD, and displays the sample name/instructions.

2. It then calls the function Init_DOC, which configures the SCI unit to communicate with a PC terminal, and configures the SCI unit to execute the callback function CB_SCIRecieve_DOC when data is received. It also configures the DOC unit to perform comparisons of an input integer to a reference integer (set in the global variable gCompareReference_DOC).

3. The sample then enters an infinite while loop. When the user enters a key in the PC terminal, the callback function CB_SCIReceive_DOC function is called. The function fetches the key entered, and converts it to an ASCII equivalent integer. The function then passes the value to the DOC unit. The DOC unit performs a comparison of the user input and the set reference integer, and updates the PC terminal indicating whether the input was a match or not. The program then returns to the infinite while loop, and repeats step 3 on the next user input.

4.6.2 Sequence Diagram

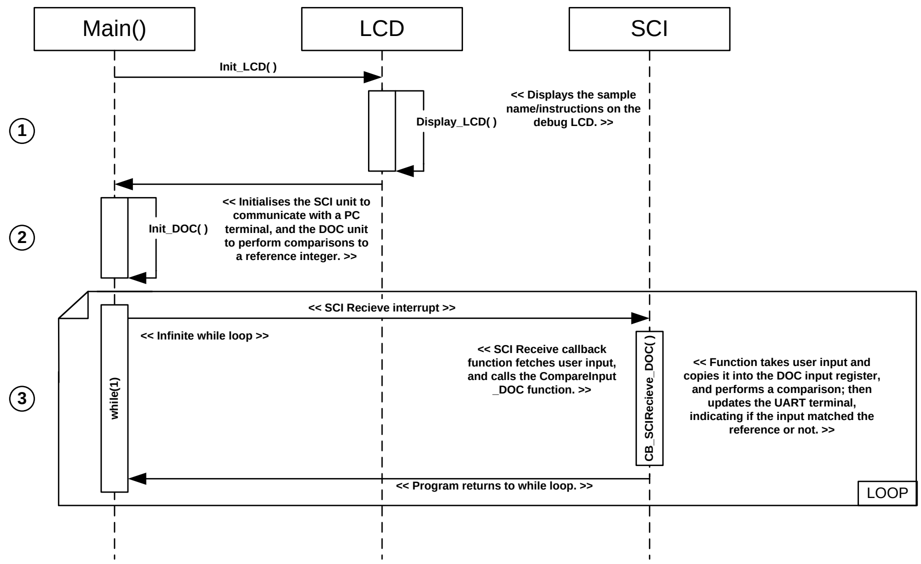

Figure 4-6: Data Operation Circuit (DOC) Sequence Diagram

4.6.3 RPDL Integration

Function	RPDL Function
Init_DOC	R_DOC_Create
	R_SCI_Create
	R_SCI_Set
	R_SCI_Receive
	R_SCI_Send
	R_SCI_Control
CB_SCIReceive_DOC	R_SCI_Send
	R_DOC_Write
	R_DOC_Read
	R_SCI_GetStatus
	R_SCI_Receive

Table 4-6: Data Operation Circuit (DOC) Sample RPDL Integration

4.7 DTC

This sample demonstrates usage of the DTC unit, by performing a DTC transfer of an ADC result to an incrementing location in an array when a switch is pressed.

4.7.1 Operation

① The sample initialises the debug LCD, and displays instructions on the screen.

② The sample calls the function Init_DTC, which configures the DTC unit and also configures an ADC channel which will trigger a DTC transfer after a successful conversion. The DTC transfer is configured to transfer the content of the ADC result register to incrementing locations in the global array, gDTC_Destination.

③ The sample then enters an infinite while loop, with the rest of the sample's functionality completed in interrupts. When switch SW3 is pressed, the callback function CB_Switch_DTC is called. The callback function checks the number of remaining transfers, and triggers an AD conversion. If there are no more remaining transfers, the function clears the contents of the gDTC_Destination array and reconfigures the DTC transfer to start from the beginning.

4.7.2 Sequence Diagram

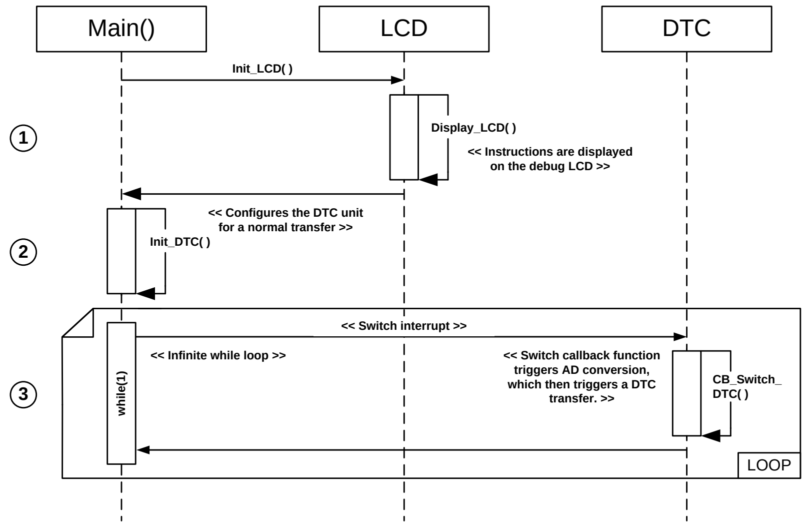

Figure 4-7: DTC Sequence Diagram

4.7.3 RPDL Integration

Function	RPDL Function
Init_DTC	R_DTC_Set
	R_DTC_Create
	R_DTC_Control
	R_ADC_12_Set
	R_ADC_12_CreateUnit
	R_ADC_12_CreateChannel
	R_INTC_Write
	R_INTC_Modify
CB_Switch_DTC	R_DTC_GetStatus
	R_DTC_Control
	R_INTC_Write
	R_ADC_12_Control
	R_IO_PORT_Modify

Table 4-7: DTC Sample RPDL Integration

4.8 Event Link Controller (ELC)

This sample code demonstrates usage of the event link controller (ELC). The sample triggers an A/D conversion when a user presses a switch. After the A/D conversion completes, the ELC unit automatically triggers a port pin to toggle. The result of the A/D operation is displayed on the debug LCD.

4.8.1 Operation

This sample may require hardware modifications in order to operate. Refer to the sample code instructions for further information before proceeding.

1. The sample first initialises the debug LCD, and displays the sample name/instructions.
2. It then calls the function Init_ELC. This function configures the ADC and the ELC units. The ELC unit is set to trigger the event of toggling a port pin after each ADC conversions. A switch interrupt is also set to call the callback function CB_Switch_ELC.
3. The program then enters an infinite while loop. When a user switch is pressed, the function CB_Switch_ELC executes. If the switch pressed is SW1, the function starts an A/D conversion, reads the result and converts it to a string. The string is then displayed on the LCD. The function then returns to the while loop.
4. As soon as an A/D conversion is completed, the ELC toggles the selected port pin without interrupting the CPU.

4.8.2 Sequence Diagram

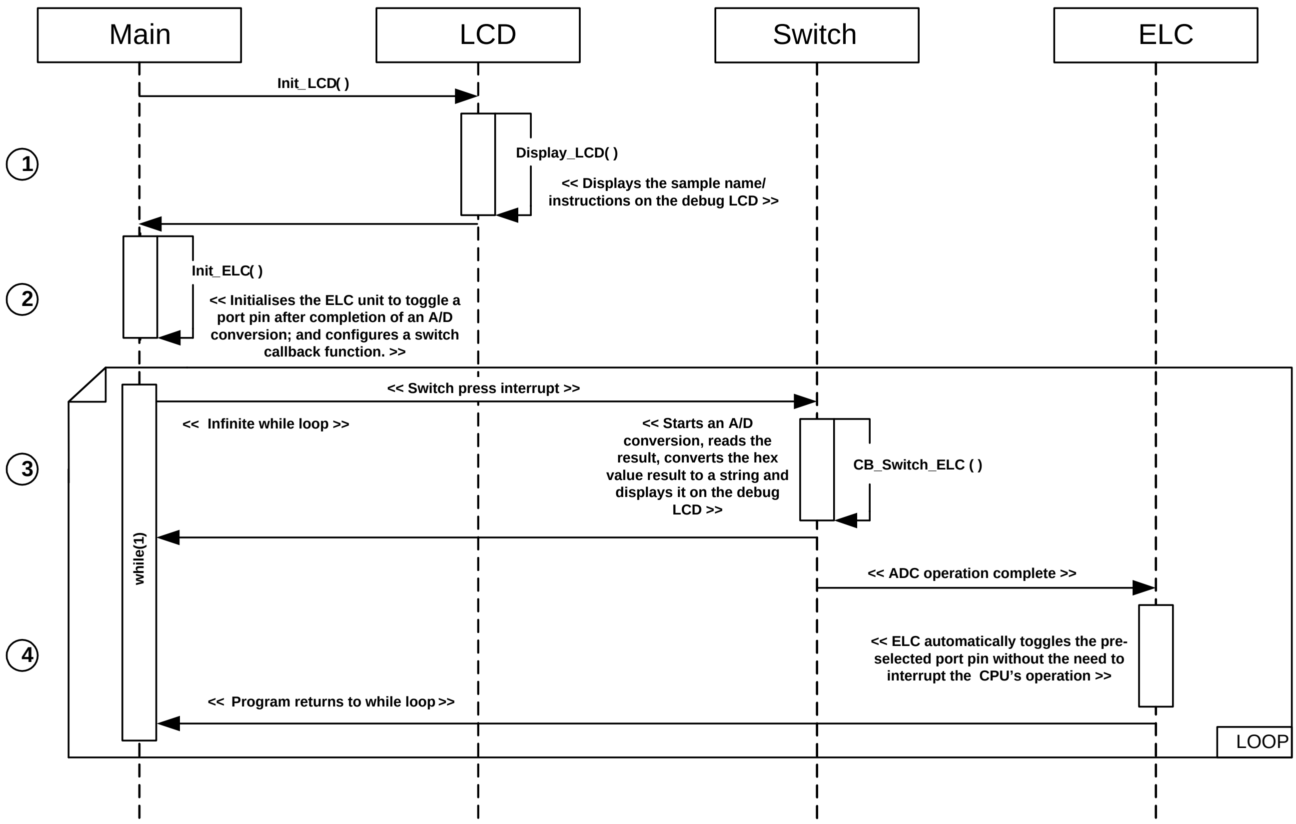

Figure 4-8: Event Link Controller (ELC) Sequence Diagram

4.8.3 RPDL Integration

Function	RPDL Function
Init_ELC	R_ADC_12_Set
	R_ELC_Create
	R_ELC_Control
	R_ADC_12_CreateUnit
	R_ADC_12_CreateChannel
	R_IO_PORT_Set
	R_IO_PORT_Write
CB_Switch_ELC	R_ADC_12_Control
	R_ADC_12_Read

Table 4-8: Event Link Controller (ELC) Sample RPDL Integration

4.9 IIC_Master

This sample demonstrates usage of the IIC unit in master mode, by performing read and write operations to an on-board EEPROM memory device.

4.9.1 Operation

① The sample initialises the debug LCD and displays the sample name/instructions on the screen.

② The sample then enters the main IIC master sequence loop, where it first calls the Init_EEPROM_Master function which configures the IIC unit to operate in master mode.

③ The master sequence then waits in an infinite while loop, polling the user switch flag. When switch SW2 is pressed, an EEPROM device write operation is executed using the Write_EEPROM_Master function. The write operation writes the string "XXRenesas IIC ", where XX is replaced with an ASCII data identifier, which increments with every write operation. If the write operation fails, the debug LCD displays "Error W."

④ When switch SW3 is pressed, an EEPROM device read operation is executed using the Read_EEPROM_Master function. The first 16 bytes of the EEPROM device's memory are read. If the read data matches the expected data string, "XXRenesas IIC ", the data identifier (XX) is displayed on the debug LCD to indicate a successful read operation. If the read operation fails, the debug LCD displays "Error R.".

4.9.2 Sequence Diagram

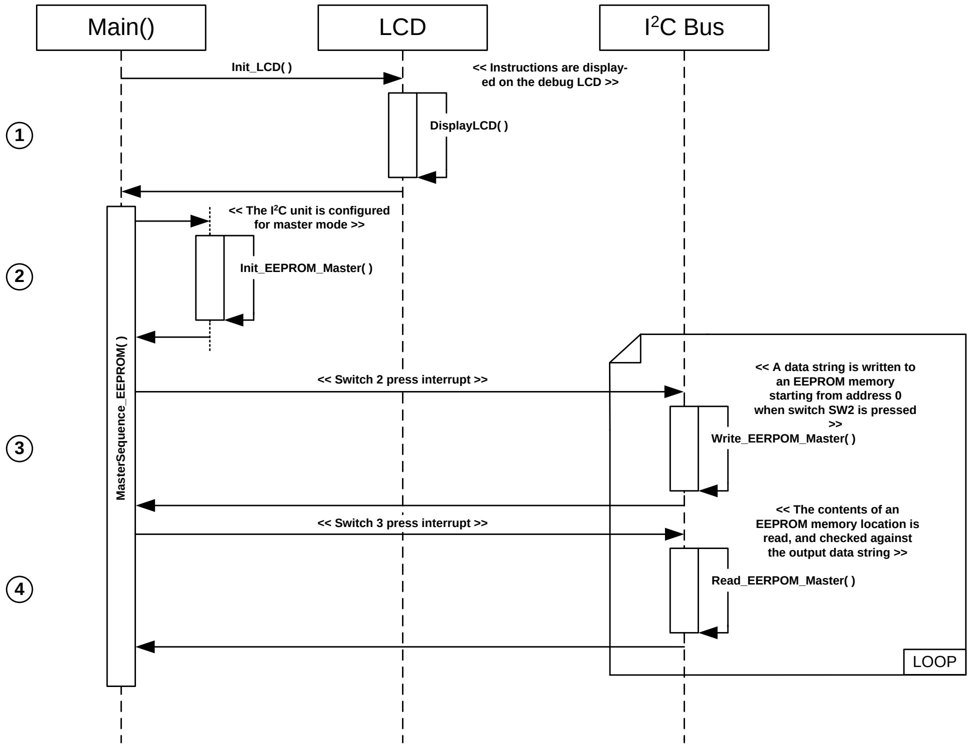

Figure 4-9: IIC_Master Sequence Diagram

4.9.3 RPDL Integration

Function	RPDL Function
Init_EEPROM_Master	R_IIC_Set
	R_IIC_Create
Write_EEPROM_Master	R_IIC_MasterSend
	R_CMT_CreateOneShot
Read_EEPROM_Master	R_IIC_MasterSend
	R_IIC_MasterReceive
	R_CMT_CreateOneShot
CheckStatus_EEPROM_Master	R_IIC_GetStatus
BusActivity_IIC	R_IO_PORT_Write

Table 4-9: IIC_Master Sample RPDL Integration

4.10 Low_Power

The Low Power sample demonstrates the low power functionality of the MCU. The MCU toggles between low power (CPU active), sleep mode (CPU inactive) and normal full power mode.

4.10.1 Operation

➔ This sample should be flashed onto the MCU, and run without the debugger. Before starting the sample, remove the debug LCD from the RSK as it is not used in the sample and can skew any power measurement readings. Refer to the main_low_power.c file for instructions on measuring the MCU power consumption.

① The sample calls the function Init_LowPower to configure the low power consumption functions. It also shuts down any unneeded peripherals to reduce power consumption, and configures a switch press interrupt to call the callback function CB_Switch_LowPower.

② The sample then executes the IdleFunction_LowPower. This function loops infinitely, and flashes LED0 to indicate CPU activity. This function is interrupted when a user switch is pressed, which calls the CB_Switch_LowPower function. This callback function changes the MCU's power mode depending on which switch is pressed.

SW1:	SW2:	SW3:
Power Mode: Normal	Power Mode: Software Standby	Power Mode: Normal
Operating Mode: Low Speed 2	Operating Mode: X *	Operating Mode: High Speed

* Operating mode is preserved from previous setting

4.10.2 Sequence Diagram

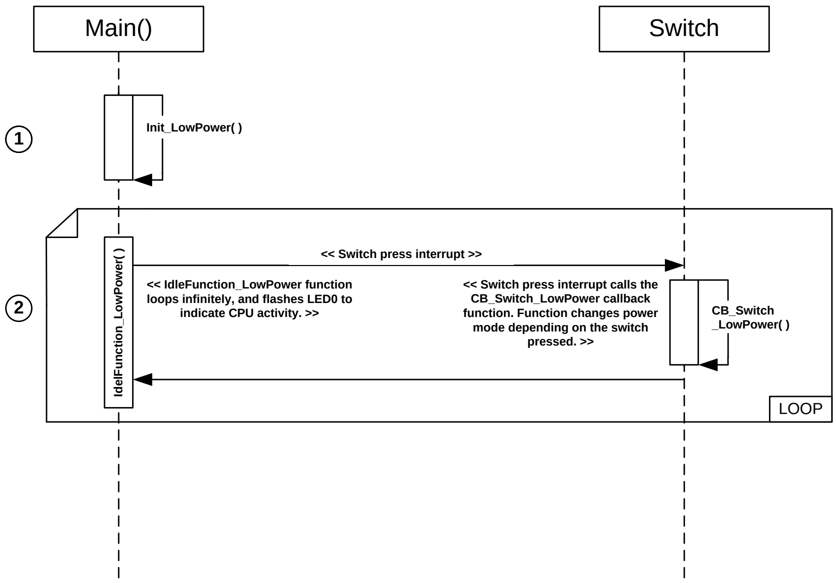

Figure 4-10: Low Power Sequence Diagram

4.10.3 RPDL Integration

Function	RPDL Function
Init_LowPower	R_LPC_Create
	R_CGC_Control
	R_DMAC_Destroy
IdleFunction_LowPower	R_IO_PORT_Modify
	R_CMT_CreateOneShot
	R_CGC_Set
	R_CGC_Control
	R_LPC_Create
	R_IO_PORT_Write
	R_LPC_Control

Table 4-10: Low Power Sample RPDL Integration

4.11 Power_Down

This sample code configures the MCU's low power consumption's settings and enters the MCU into standby mode. The RTC is configured to interrupt and wake the MCU at intervals of 1 second, to update the LCD with the time elapsed since the program was started.

4.11.1 Operation

1. The sample initialises the debug LCD and displays 'PWR MODE' (Power Mode) on the first line of the LCD, and the current power mode, 'Active' on the second line.

2. The sample then calls the function Init_PowerDown which configures the low power consumption and real time clock registers.

3. The sample then enters an infinite while loop. The while loop immediately enters the MCU into standby mode. After a second, the RTC unit triggers an interrupt which wakes the MCU from standby mode and calls the callback function CB_RTC_PowerDown.

4. The function CB_RTC_PowerDown reads the time from the RTC unit, and updates the LCD display with the current time (time elapsed since program began). The program then returns back to the while loop, and repeats step 3 & 4 infinitely.

4.11.2 Sequence Diagram

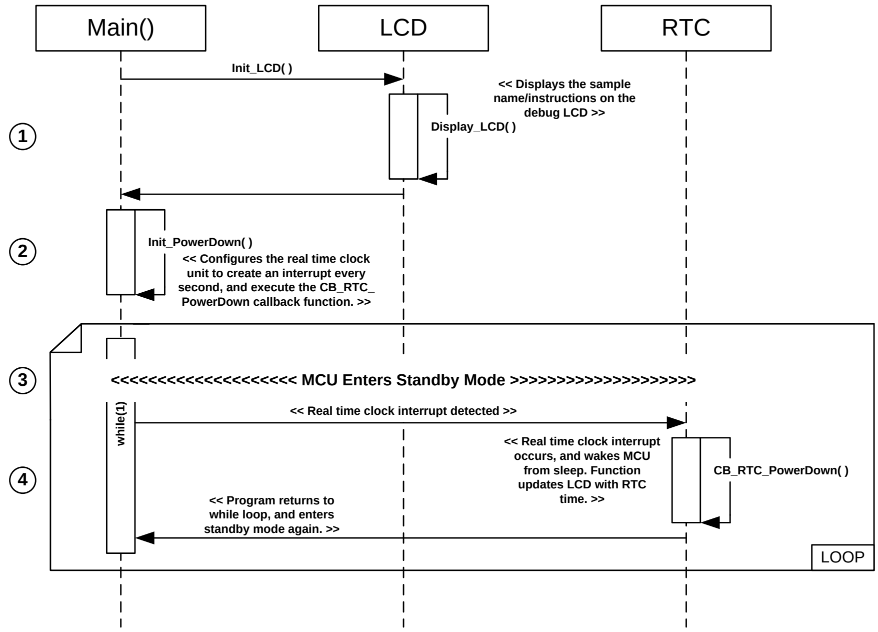

Figure 4-11: Power_Down Sequence Diagram

4.11.3 RPDL Integration

Function	RPDL Function
Init_PowerDown	R_RTC_Create
	R_RTC_Control
	R_LPC_Create
CB_RTC_PowerDown	R_RTC_Read
	R_CMT_CreateOneShot

Table 4-11: Power_Down Sample RPDL Integration

4.12 PWM

This sample code configures the timer to generate a 1KHz waveform, with a constantly changing duty cycle. The duty cycle begins at 10% and gradually increases to 90%, then cycles back to 10% continuously until the user presses switch 1 and freezes the duty cycle. The value at this freeze-point is displayed on the debug LCD.

4.12.1 Operation

➔ Before starting this sample, the user should refer to the sample code instructions in order to connect the oscilloscope to the correct location.

① The sample initialises the debug LCD and displays the sample name/instructions on the screen.

② It then calls the function Init_PWM which configures a timer channel to generate an initial periodic 1 KHz output signal with a 10% duty cycle. The function configures the timer to generate an interrupt at the end of each period, serviced by the callback function CB_Timer_PWM. The function also configures a switch interrupt and callback function.

③ The CB_Timer_PWM callback function increments the duty cycle each time it is executed if the duty cycle is less than 90%, otherwise it resets the duty cycle to 10%.

④ When the user switch is pressed, the callback function CB_Switch_PWM is called. The function prevents the duty cycle from changing, and displays the current duty cycle on the debug LCD.

4.12.2 Sequence Diagram

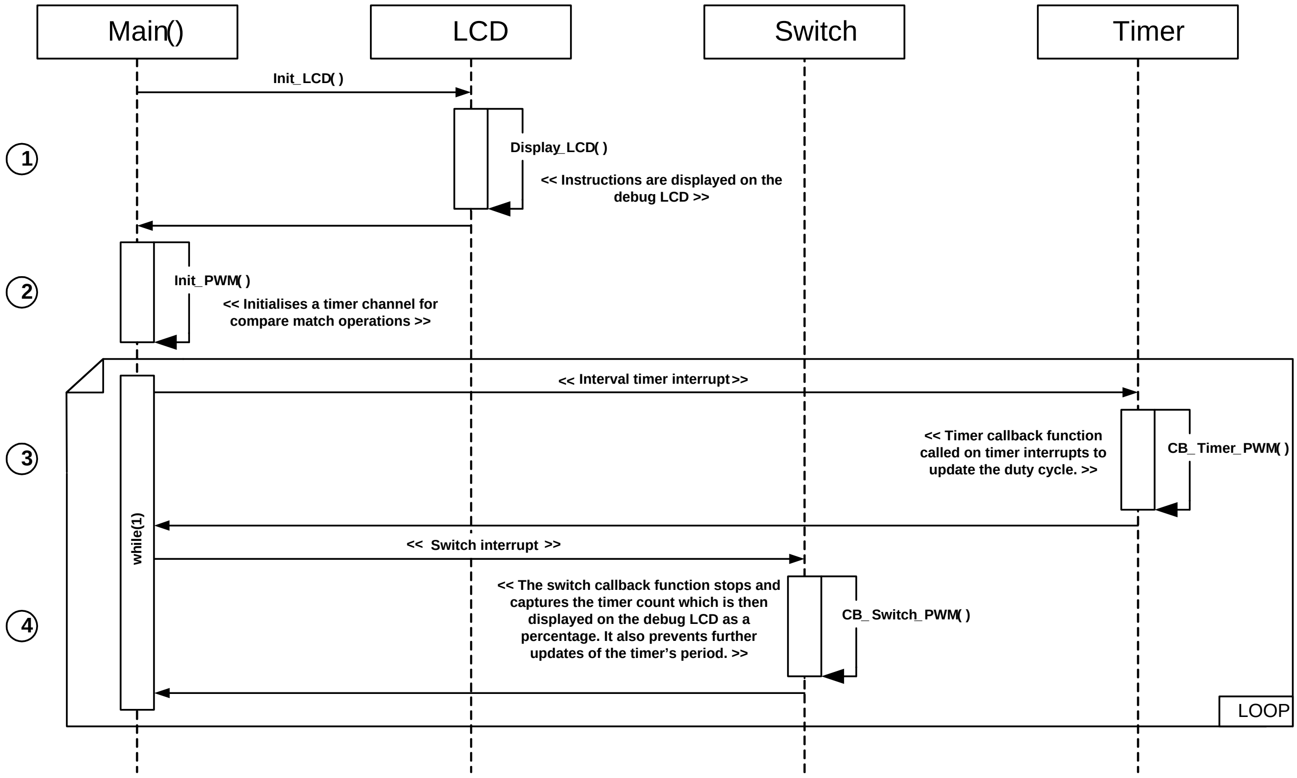

Figure 4-12: PWM Sequence Diagram

4.12.3 RPDL Integration

Function	RPDL Function
Init_PWM	R_TMR_Set
	R_TMR_CreatePeriodic
CB_Timer_PWM	R_TMR_ControlPeriodic

Table 4-12: PWM Sample RPDL Integration

4.13 RTC

This sample code demonstrates the usage of the real time clock's functionality by running a timer from a 32.768 KHz clock source. The time is displayed on the debug LCD using the 24-hour digital format hh:mm:ss, starting from 11:59:30.

4.13.1 Operation

1. The sample initialises the debug LCD and displays the sample name/instructions on the screen.
2. The sample then calls the Init_RTC to initialise the RTC settings and configures two callback functions CB_Alarm_RTC and CB_1HZ_RTC.
3. The sample enters an infinite while loop, which is interrupted every second by the RTC callback function, CB_1HZ_RTC. This function fetches the time (time elapsed from sample start) from the RTC unit, and displays it on the debug LCD.
4. The while loop is interrupted again when the time matches the alarm time. The interrupt executes the callback function CB_Alarm_RTC, which turns on LED1.

4.13.2 Sequence Diagram

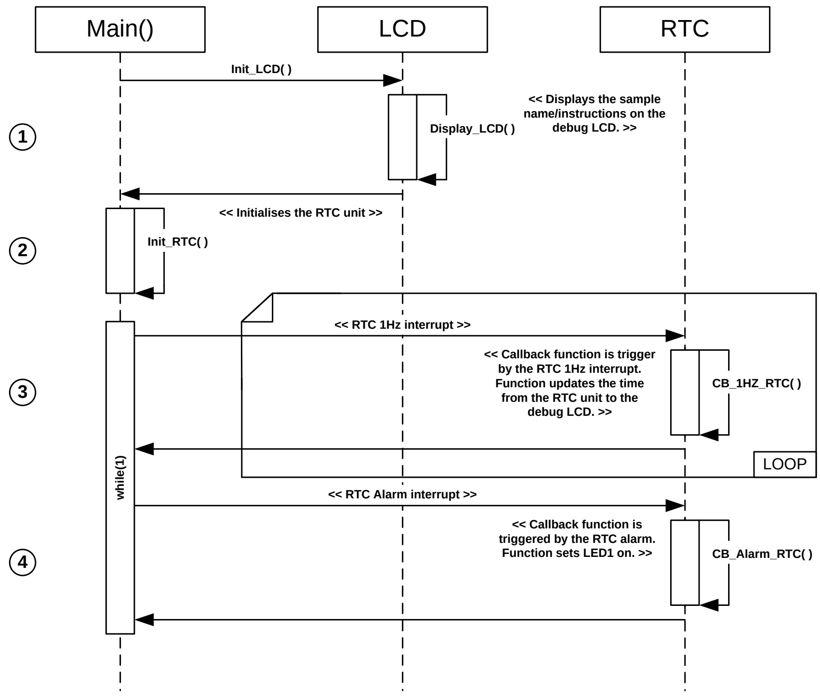

Figure 4-13: RTC Sequence Diagram

4.13.3 RPDL Integration

Function	RPDL Function
Init_RTC	R_RTC_Create
	R_RTC_Control
CB_1HZ_RTC	R_RTC_Read
CB_Alarm_RTC	R_IO_PORT_Write

Table 4-13: RTC Sample RPDL Integration

4.14 SPI

This sample code demonstrates usage of the SPI unit by performing a simple loopback test.

4.14.1 Operation

① The sample first initialises the debug LCD, and displays the sample name/instructions.

② The Init_SPI function is called, which initialises the SPI unit and configures an SPI receive interrupt callback. An ADC unit is configured for single scan operations. The function also configures a switch callback function.

③ The sample then enters an infinite while loop. When the user presses the switch specified (see sample code comments), the callback function CB_Switch_SPI is called. This callback function takes an ADC reading, and transmits it via the SPI loopback. Once the transfer completes, the received data is compared against the original value sent. If they match, the result is displayed on the debug LCD. If there is a mismatch, an error is reported on the debug LCD.

4.14.2 Sequence Diagram

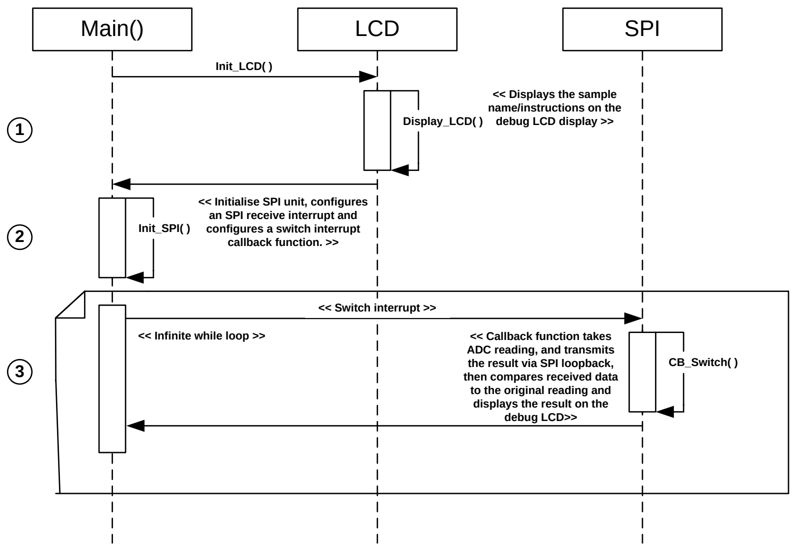

Figure 4-14: SPI Loopback Sequence Diagram

4.14.3 RPDL Integration

Function	RPDL Function
Init_SPI	R_SPI_Create
	R_SPI_Control
	R_SPI_Command
	R_ADC_12_Set
	R_ADC_12_CreateUnit
	R_ADC_12_CreateChannel
CB_Switch	R_ADC_12_Control
	R_ADC_12_Read
	R_SPI_Transfer

Table 4-14: SPI Sample RPDL Integration

4.15 Temperature Sensor

This sample code demonstrates usage of the on-chip temperature sensor. The sample takes repeat readings of the on-chip temperature sensor using an ADC channel, and converts it into degrees Celsius.

4.15.1 Operation

① The sample first initialises the debug LCD, and displays the sample name/instructions.

② It then calls the function Init_TempSensor, to initialise the ADC module to operate in repeat mode – the ADC unit will now continuously take readings of the temperature sensor. The function also initialises an interval timer interrupt, which calls the CB_Timer_TempSensor callback function every interval.

③ The sample then enters an infinite while loop. The timer interrupt function CB_Timer_TempSensor is called every interval, and fetches the current AD conversion result. The result is then converted into degrees Celsius, and displayed on the debug LCD.

4.15.2 Sequence Diagram

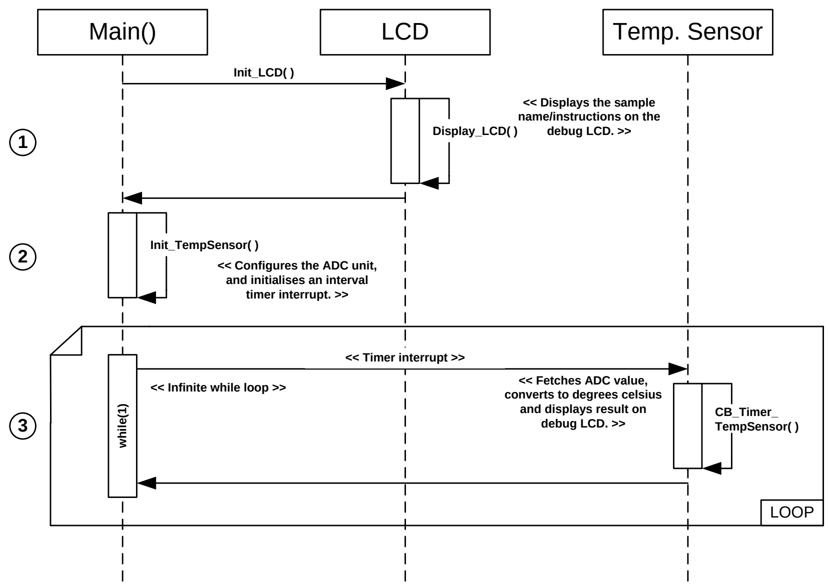

Figure 4-15: Temperature Sensor Sequence Diagram

4.15.3 RPDL Integration

Function	RPDL Function
Init_TempSensor	R_RWP_Control
	R_ADC_12_CreateUnit
	R_CMT_Create
	R_TS_Create
	R_TS_Control
CB_Timer_TempSensor	R_ADC_12_Read
	R_ADC_12_Control

Table 4-15: Temperature Sensor Sample RPDL Integration

4.16 Timer_Mode

This sample code configures the timer to generate 1 KHz waveform. The waveform can be seen on the oscilloscope.

4.16.1 Operation

➔ Before starting this sample, the user should refer to the sample code instructions in order to connect the oscilloscope to the correct location.

① The sample initialises the debug LCD and displays the sample name/instructions on the screen.

② The sample then configures a timer channel to output a 1 KHz periodic square wave with 50% duty.

③ The sample then enters an infinite while loop. The timer unit will continue to produce the waveform.

4.16.2 Sequence Diagram

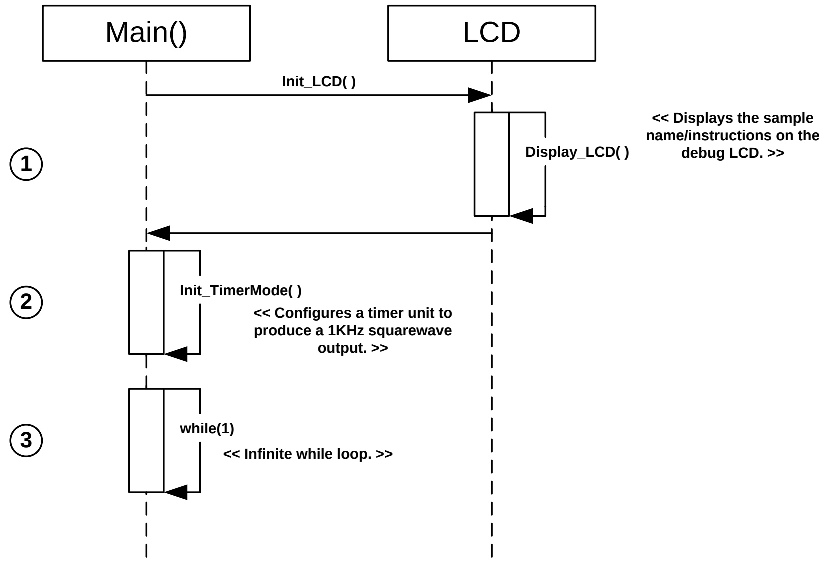

Figure 4-16: Timer Mode Sequence Diagram

4.16.3 RPDL Integration

Function	RPDL Function
Init_TimerMode	R_MTU2_Set
	R_MTU2_Create
	R_MTU2_ControlChannel
CB_Timer_TimerMode	R_IO_PORT_Modiy

Table 4-16: Timer Mode Sample RPDL Integration

4.17 Voltage Detect

4.17.1 Description

In this sample, the LVD (Low Voltage Detection) circuit is configured to generate an interrupt when the power supply equals or falls below the detection level.

4.17.2 Operation

➔ Before starting this sample, follow the instructions in main source file to ensure the RSK is configured correctly. Connect a 5V variable power supply to the power socket, and set the initial voltage to 5V.

① The sample initialises the debug LCD, and displays instructions on the screen.

② The sample then calls the Init_VDET function, which initialises the LVD unit to generate interrupts when VCC drops to 4.0V and 1.9V. The function also configures a CMT timer, to periodically toggle the user LEDs.

③ The sample then enters an infinite while loop. The while loop periodically interrupt by the callback function CB_Timer_VDET. If the global gLEDsync_flg is set, it toggles LEDs LED0 to LED2 synchronously, and LED3 asynchronously. If gLEDsync_flg is not set, it toggles all the user LEDs together.

④ When the input voltage is lowered to below 4.0V, the CB_LVD2_VDET callback function is called by the voltage detect interrupt. This function sets the gLEDsync_flg, turns off LEDs LED0 to LED2, and waits until the supply voltage returns to over 4.0V.

⑤ When the input voltage is lowered to below 1.9V, the CB_NMI_VDET callback function is called by the non-maskable voltage detect interrupt. This interrupts any other callback functions that may be running. This function turns off all the user LEDs, and waits until the voltage level rises above 1.9V before exiting.

4.17.3 Sequence Diagram

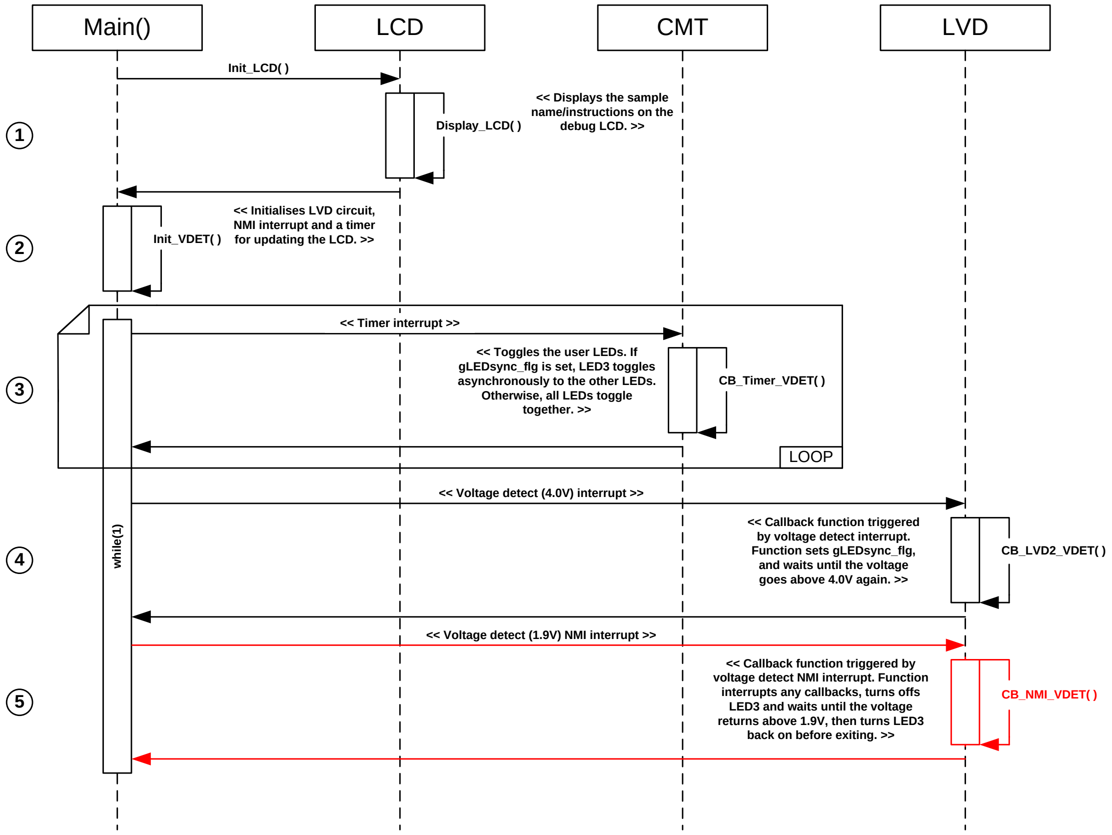

Figure 4-17: LVD Sequence Diagram

4.17.4 RPDL Integration

Function	RPDL Function
Init_VDET	R_LVD_Create
	R_INTC_CreateExtInterrupt
	R_CMT_Create
CB_Timer_VDET	R_IO_PORT_Modify
CB_LVD2_VDET	R_CMT_Control
	R_IO_PORT_Write
	R_LVD_GetStatus
	R_LVD_Control
	R_INTC_Write
CB_NMI_VDET	R_IO_PORT_Write
	R_LVD_GetStatus
	R_LVD_Control
	R_INTC_ControlExtInterrupt

Table 4-17: Low Voltage Detection Sample RPDL Integration

4.18 WDT

This sample code demonstrates the usage of the watchdog timer functionality; by resetting the WDT count regularly, at a rate controlled by the potentiometer to prevent its period from timing out. When the WDT's period is reached, the LEDs stop flashing and the program halts in an infinite while loop.

4.18.1 Operation

➔ Before starting this sample, ensure that the potentiometer shaft is turned anti-clockwise fully.

1. The sample initialises the debug LCD and displays the sample name/instructions on the screen.

2. The Init_WDT function configures the watchdog timer unit, the ADC unit and a timer unit. The ADC unit is configured to operate in continuous mode, and the timer is configured to generate an interval interrupt that executes the callback function CB_Timer_WDT.

3. The sample then enters an infinite while loop. The while loop is interrupted by the CB_Timer_WDT callback function. This function resets the WDT count, toggles the LEDs and fetches the ADC result. This result is used to set the new timer interval period. By adjusting the potentiometer, the length between WDT resets is varied.

4. When the WDT reset interval is too long, the watchdog timer will overflow and trigger an interrupt, which calls the callback function CB_Overflow_WDT. This function sets the LEDs to static, displays a watchdog overflow message on the debug LCD and waits in an infinite while loop.

4.18.2 Sequence Diagram

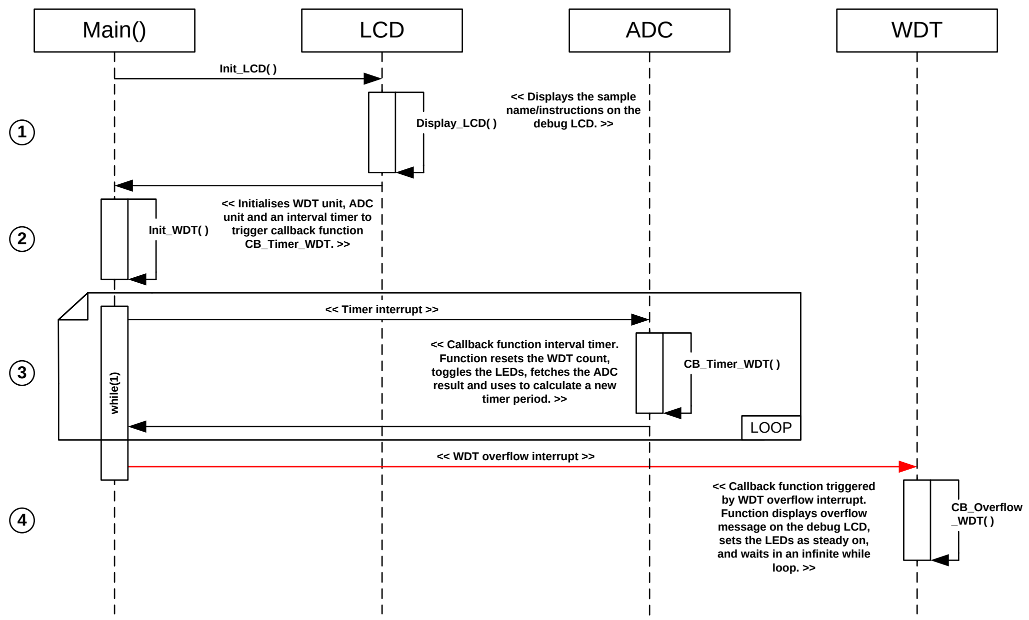

Figure 4-18: WDT Sequence Diagram

4.18.3 RPDL Integration

Function	RPDL Function
Init_WDT	R_ADC_12_Set
	R_ADC_12_CreateUnit
	R_ADC_12_CreateChannel
	R_ADC_12_Control
	R_CMT_Create
	R_WDT_Set
	R_INTC_CreateExtInterrupt
	R_WDT_Control
CB_Timer_WDT	R_WDT_Control
	R_ADC_12_Read
	R_IO_PORT_Modify
	R_CMT_Control
CB_WDTOverflow	R_IO_Port_Write

Table 4-18: WDT Sample RPDL Integration

5. Additional Information

Technical Support
For details on how to use High-performance Embedded Workshop (HEW), refer to the HEW manual available on the CD or from the web site.

For information about the RX210 series microcontrollers refer to the RX210 Group Hardware Manual.

For information about the RX assembly language, refer to the RX200 Series Software Manual.

Online technical support and information is available at: http://www.renesas.com/rskrx210b

Technical Contact Details

Please refer to the contact details listed in section 7 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 RX210B Software Manual

Rev.	Date	Description	
		Page	Summary
1.00	May 02, 2013	—	First Edition issued

Renesas Starter Kit User's Manual: Software Manual

Publication Date: Rev. 1.00 May 02, 2013

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.
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: +44-1628-651-700, Fax: +44-1628-651-804
Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, 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.
13F, No. 363, Fu Shing North Road, Taipei, 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.
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

Colophon 1.3

RX210 Group (B Mask)

Renesas Electronics Corporation