

Renesas e² studio

Smart Configurator Application Examples: CMT, A/D, SCI, DMA, USB

Introduction

Smart Configurator (SC) is a GUI-based tool that has the functionalities of code generation and configuration for drivers, middleware and pins. SC generates suitable code for each Renesas MCU family and has the functionality to import code generated by FIT modules.

This application note guides user to use SC in the e² studio to configure drivers and generate codes.

Target Device and support compiler

Refer to the following URL for the range of supported devices:

<https://www.renesas.com/smart-configurator>

The following operating systems on the host computer are supported:

- Windows 7 32-bit / 64-bit
- Windows 8.1 32-bit / 64-bit
- Windows 10 32-bit / 64-bit

Software Components

Smart Configurator supports 2 types of software components: Code Generator (CG) and Firmware Integration Technology (FIT). Drivers and middleware supported by each software type are:

- Basic drivers: CG drivers (CMT, A/D Converter, SCI, etc.)
FIT modules (CMT, DTC, DMAC, RSPI, SCIFA, etc.)
- Middleware: FIT modules (USB, Ethernet, Flash Memory (programming the on-chip flash memory), etc.)

The basic driver is a control program for peripheral functions of microcomputer such as CMT, A/D converter, SCI, etc. It is convenient to embed software component (CG driver) using code generation (CG) function.

In addition, FIT modules can be embedded for using middleware such as USB, Ethernet, and Flash memory (programming the on-chip flash memory) as software components.

List of abbreviations:

CMT: Compare Match Timer

DTC: Data Transfer Controller

DMAC: Direct Memory Access Controller

RSPI: Serial Peripheral Interface

SCIFA: FIFO Embedded Serial Communications Interface

Contents

1. Overview	4
1.1 Purpose	4
1.2 Operating Environment	4
1.3 Basic operation steps of Smart Configurator project	5
2. Application Example 1 (Creating a basic program using Smart Configurator).....	6
2.1 Creating a Workspace	8
2.2 Creating a Project	8
2.3 Adding a peripheral driver	12
2.4 Generating codes	18
2.5 Adding definition codes in “r_cg_userdefine.h” file	19
2.6 Adding application codes in Compare Match Timer driver	19
2.7 Build and run on hardware board.....	21
2.8 Changing LED2 blinking interval.....	25
2.9 Operation on board	26
3. Application Example 2 (Adding a function using Smart Configurator)	27
3.1 Adding a peripheral driver	29
3.2 Generating codes	31
3.3 Adding application codes in Single Scan Mode S12AD driver	32
3.4 Adding application codes in <i>main()</i>	33
3.5 Build and run on hardware board.....	35
3.6 Operation on board	35
4. Application Example 3 (Changing a function in Smart Configurator)	36
4.1 Removing S12AD driver and related application codes	39
4.2 Adding a peripheral driver	40
4.3 Generating codes	43
4.4 Adding application codes in SCI/SCIF Asynchronous Mode driver	44
4.5 Adding application codes in <i>main()</i>	47
4.6 Build and run on hardware board.....	51
4.7 Operation on board	53
5. Application Example 4 (Transferring data using DMA)	54
5.1 Adding and modifying peripheral drivers.....	58
5.1.1 Add DMAC0 driver	58
5.1.2 Add DMAC1 driver	60
5.1.3 Modify SCI8 driver	62
5.2 Generating codes	62

5.3	Adding application codes in DMAC0 driver	63
5.4	Adding application codes in DMAC1 driver	64
5.5	Modifying application codes in SCI8 driver	67
5.6	Adding application codes in <i>main()</i>	68
5.7	Build and run on hardware board.....	71
5.8	Operation on board.....	71
6.	Application Example 5 (Integration with USB function using Smart Configurator and FIT module).....	72
6.1	Removing SCI/SCIF Asynchronous Mode driver and related application codes.....	75
6.2	Adding a peripheral driver	77
6.3	Generating codes.....	81
6.4	Adding header file of USB PCDC driver	82
6.5	Adding application codes in <i>main()</i>	90
6.6	Build and run on hardware board.....	93
6.7	Operation on board.....	95
6.8	Additional debugging assistance tool (QE)	95
	Website and Support.....	96

1. Overview

1.1 Purpose

This document guides user to use Smart Configurator in few application examples.

1.2 Operating Environment

Target devices	RX64M Group RX65N, RX651 Group RX130 Group
Debugger	E1 /E2 Lite
IDE	e ² studio v.7.0.0 or above
Toolchains	Renesas C/C++ compiler package for RX family

1.3 Basic operation steps of Smart Configurator project

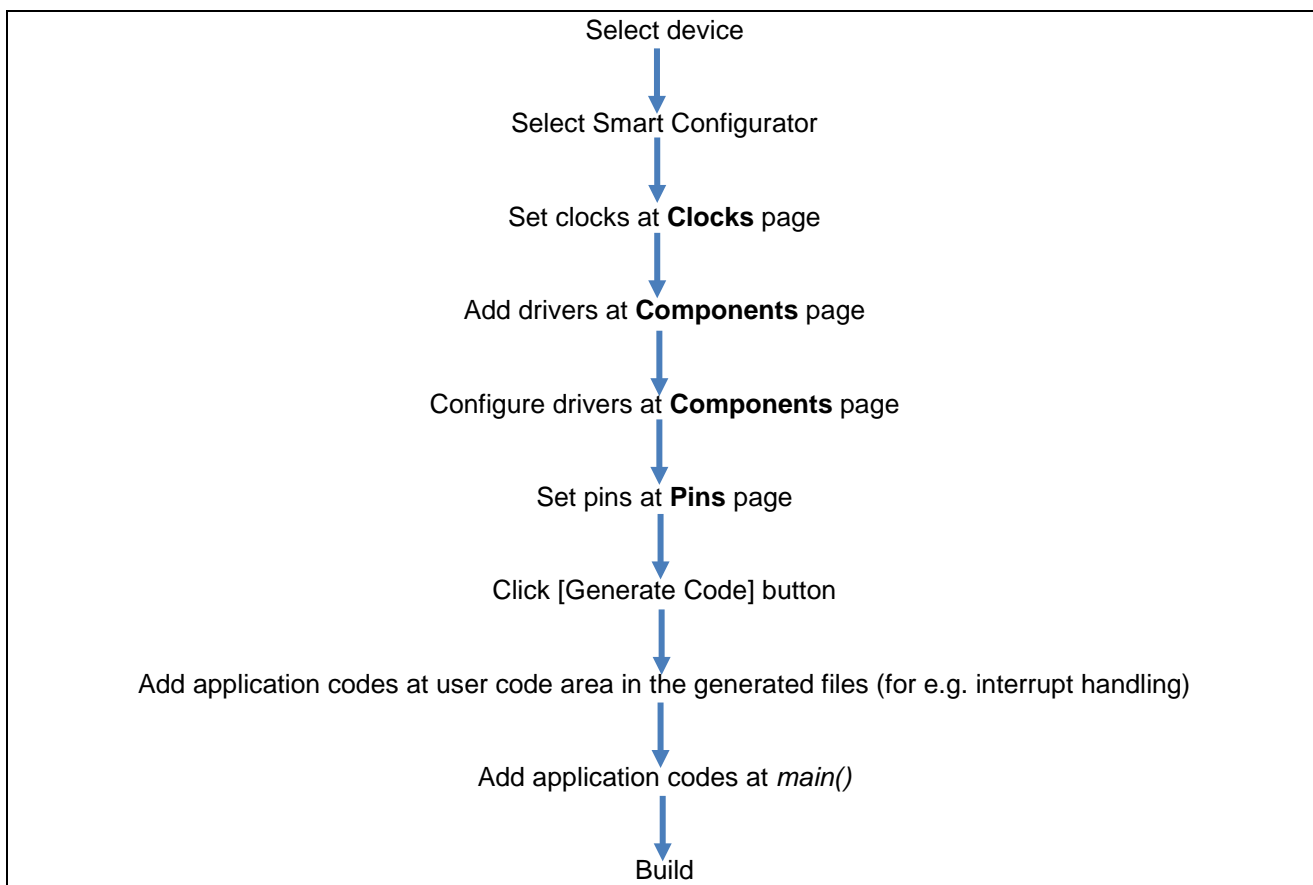


Figure 1-1-1 Basic operation

Refer to “Smart Configurator User Guide” document for detailed operations of Smart Configurator.

2. Application Example 1 (Creating a basic program using Smart Configurator)

This chapter describes how to create a LED blinking project using Code Generator drivers in Smart Configurator. The following resources are required for this example:

- IDE: e² studio v7.0 and above
- Toolchain: Renesas RXC Toolchain
- Board: Renesas Starter Kit+ for RX65N-2MB
- MCU: RX65N R5F565NEDxFC
- Debugger: E1 or E2 Lite

The schematic diagram of Renesas Starter Kit+ for RX65N 2MB is shown below. In this example, LED2 will be configured to blink every 5000 counts.

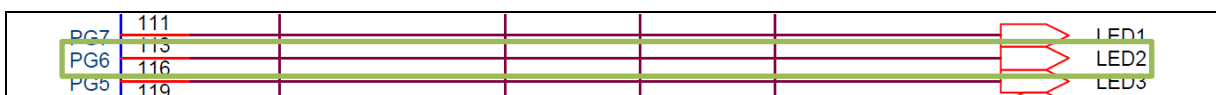


Figure 2-1 Schematic diagram of Renesas Starter Kit+ for RX65N 2MB

The table below shows CG drivers to be configured.

Drivers	Resource	Function
Ports	PORTG	Connected to LED2 (PG6) Output a low level '0' to turn on LED2
Compare Match Timer	CMT0	Toggle LEDs every 5000 counts with output compare interrupt handler

Based on the schematic below, RX65N main clock is connected to a 24MHz crystal resonator. This clock will be configured as main clock source in Chapter 2.7 on debug configuration setting.

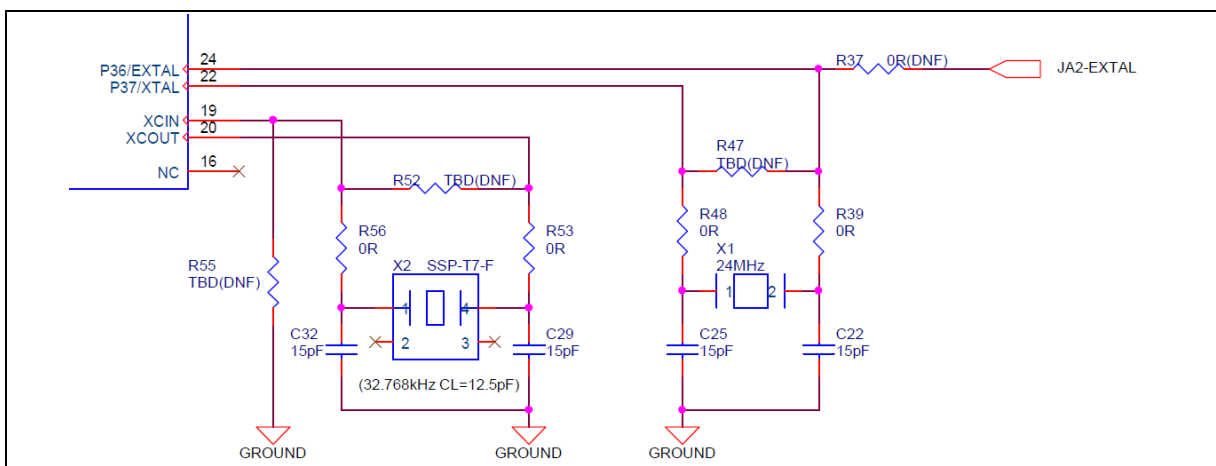


Figure 2-2 Schematic diagram of Renesas Starter Kit+ for RX65N-2MB

The program flowchart is shown as below:

a) Main function:

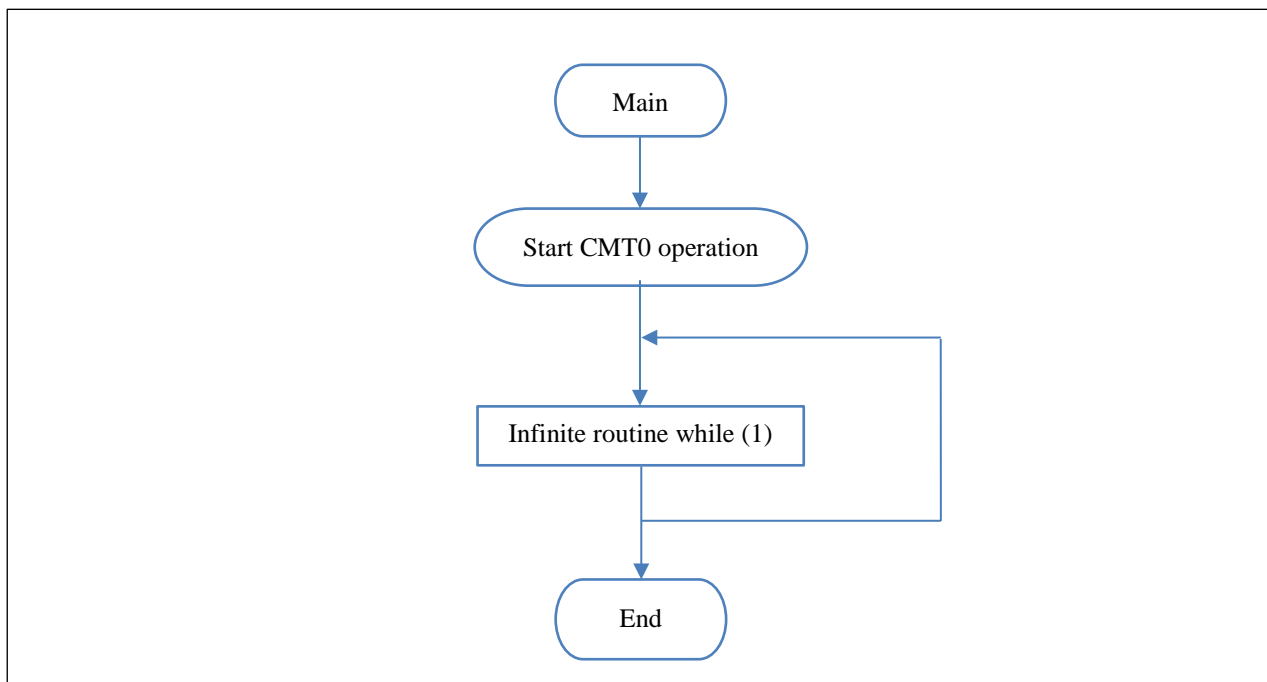


Figure 2-3 Main flowchart

b) CMT0 interrupt function:

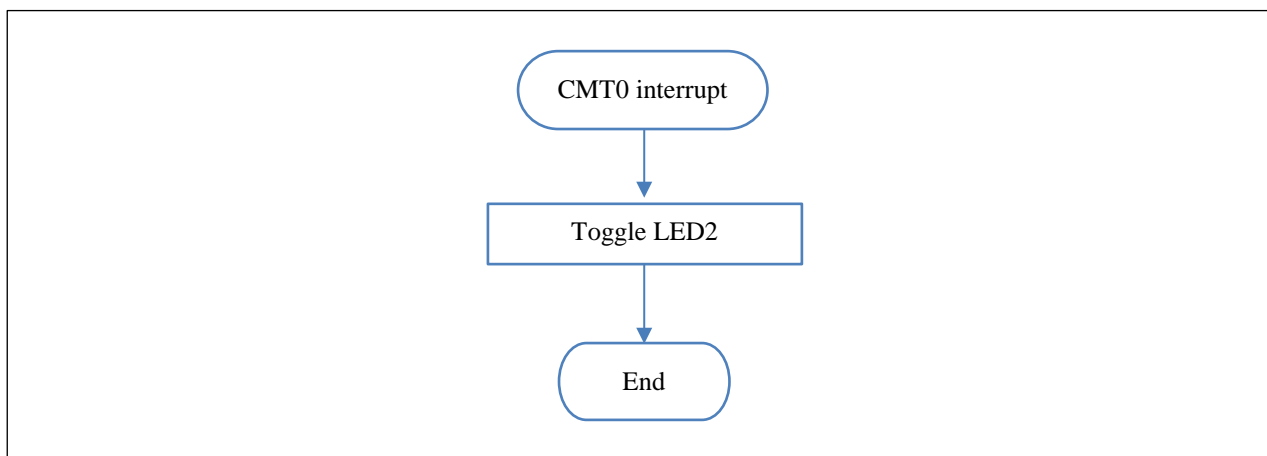


Figure 2-4 CMT0 interrupt flowchart

2.1 Creating a Workspace

- 1) Start e² studio from Windows® Start Menu. Use default workspace folder and click [OK].

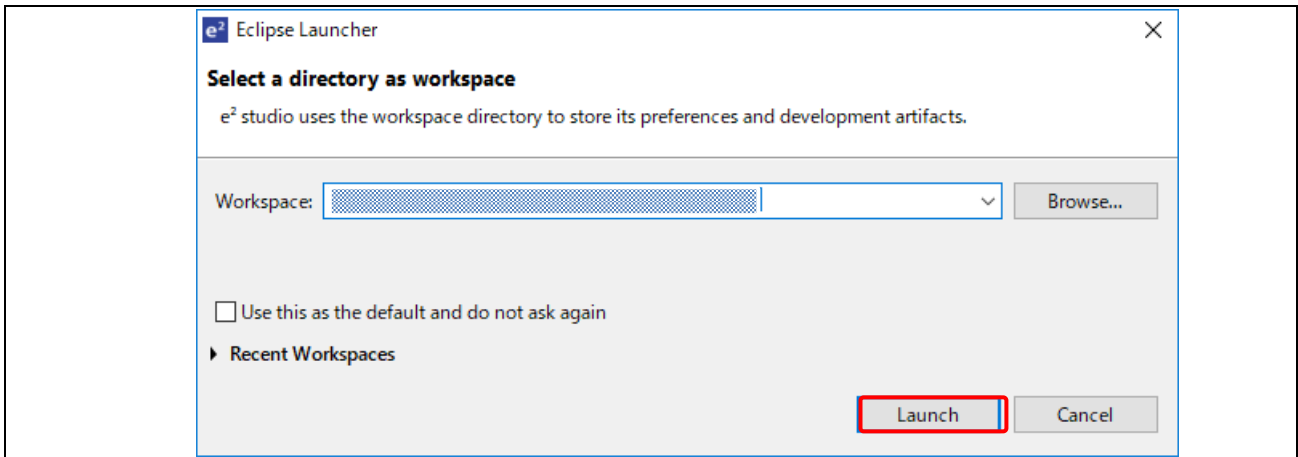


Figure 2-5 Workspace Launcher

2.2 Creating a Project

- 1) Create a new C project in e² studio.
Go to [File] → [New] → [C/C++ Project] to start new project generation

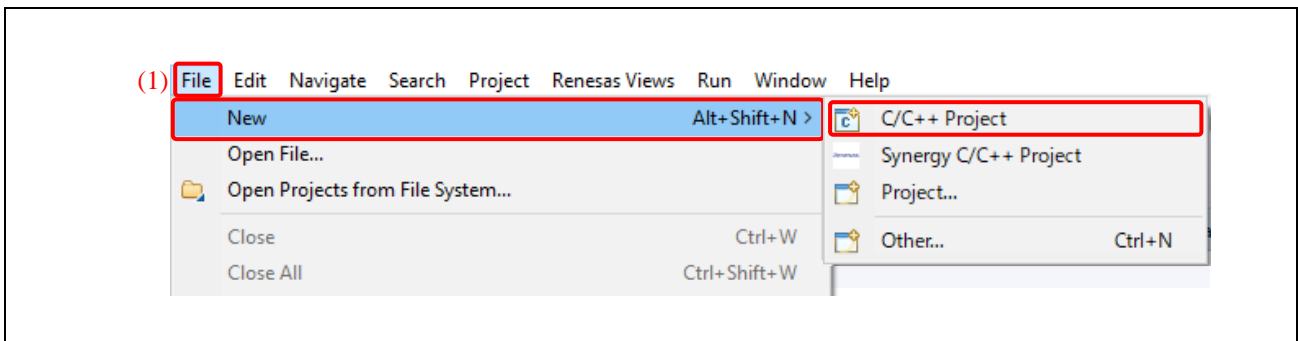


Figure 2-6 Creating project from File menu

- 2) Select [Renesas RX] → [Renesas CC-RX C/C++ Executable Project] → [Next]

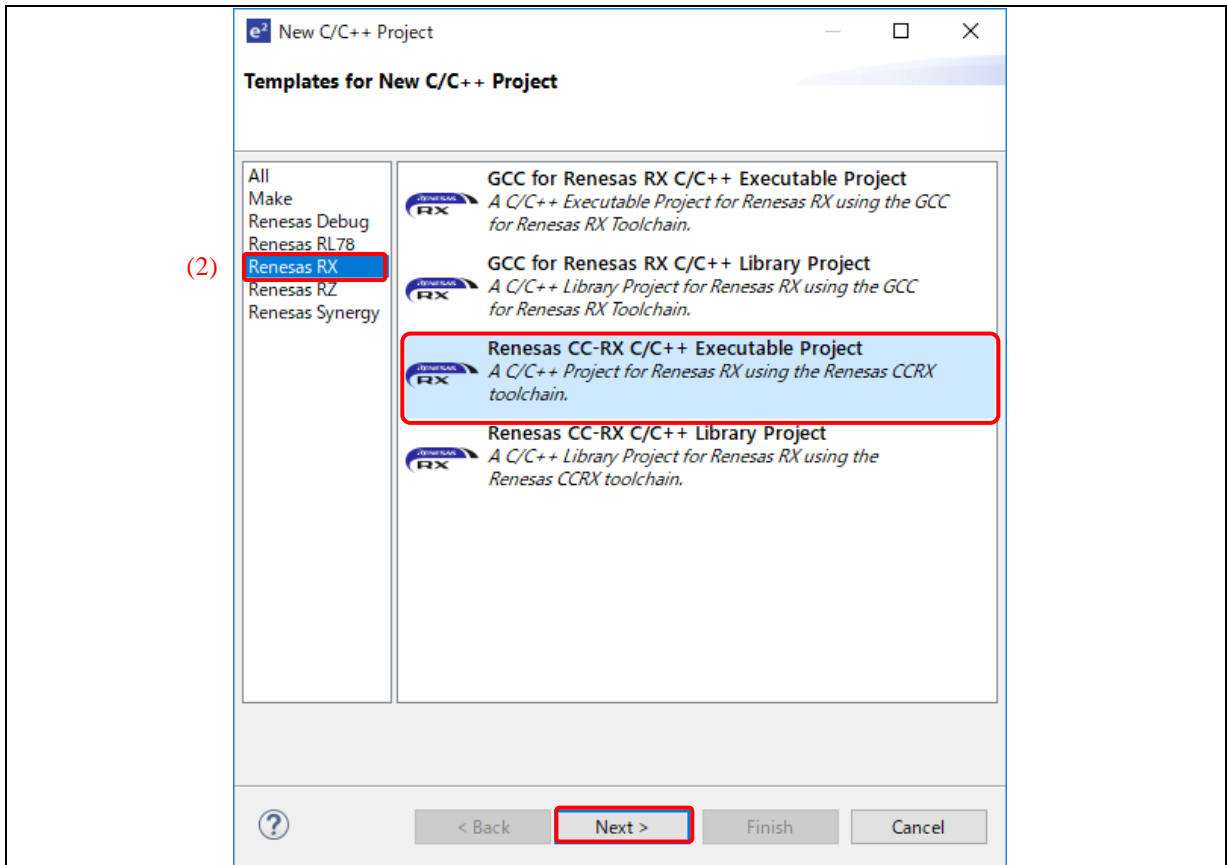


Figure 2-7 Creating project from File menu

- 3) Give an appropriate name to the project, for example “Smart_Configurator_Example” → [Next]

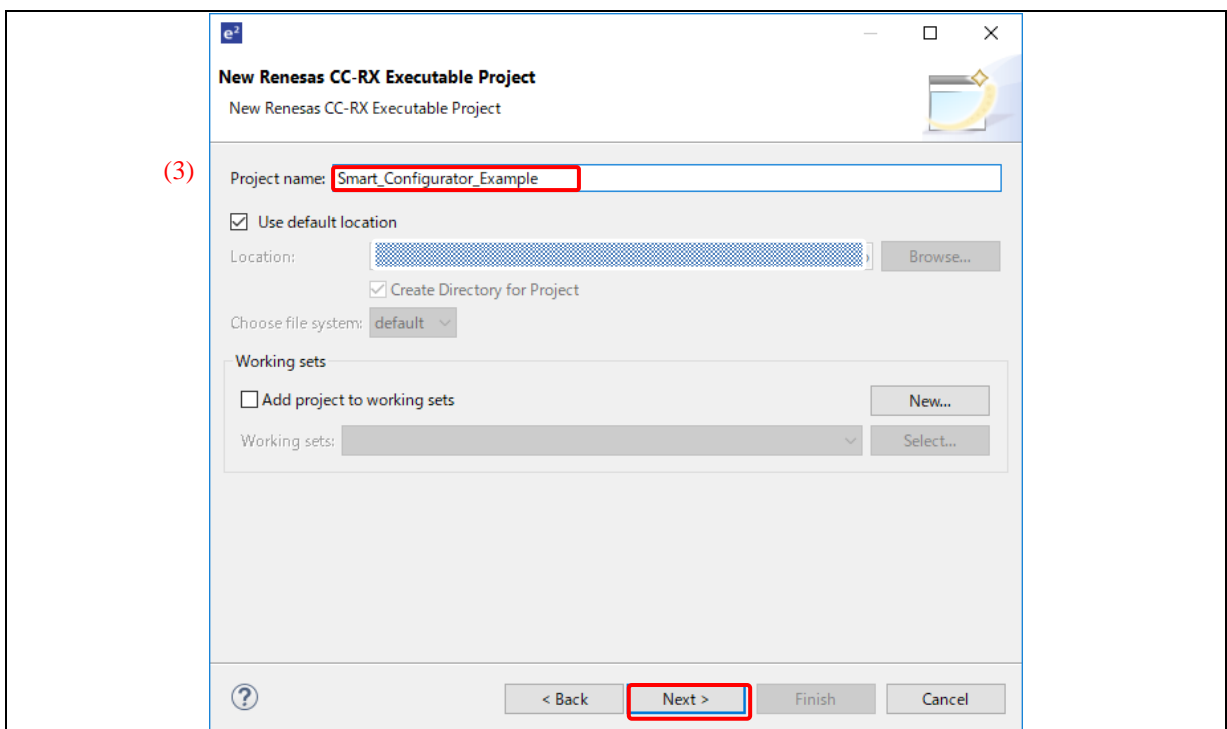


Figure 2-8 Creating project from File menu

- 4) Select "C" as Language
- 5) Select "Renesas CCRX" as Toolchain
- 6) Select Toolchain Version. For e.g. "v3.01.00"
- 7) Select Target Device: "RX600 > RX65N > RX65N - 176pin > R5F565NEDxFC"
- 8) Ensure [Create Hardware Debug Configuration] is ticked. Select emulator, for e.g. "E2 Lite(RX)".
- 9) Click [Next]

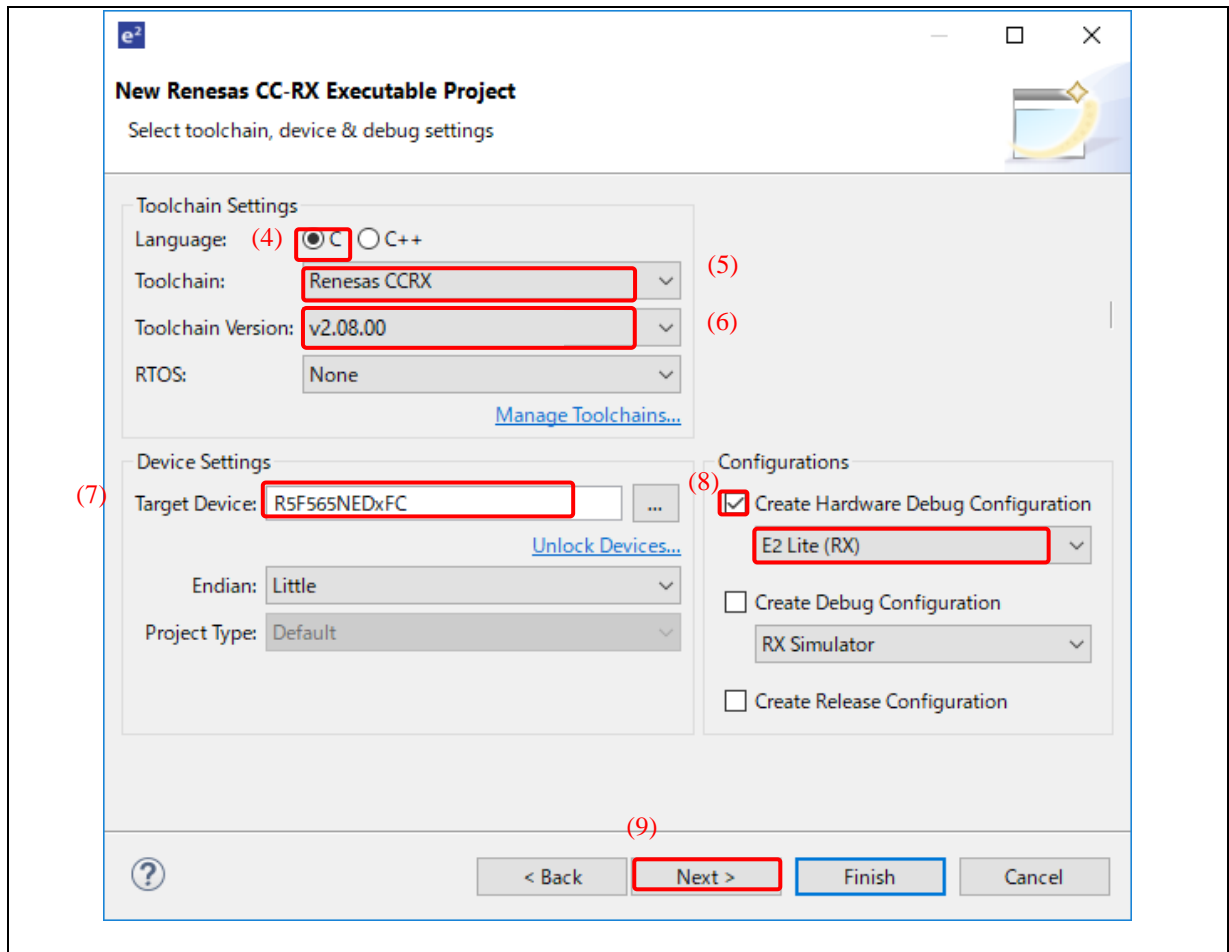


Figure 2-9 C Project - Target Specific Settings example with E2 Lite emulator

- 10) In the “Select Coding Assistant settings” dialog, select the checkbox of “Smart Configurator”
- 11) Click [Finish]

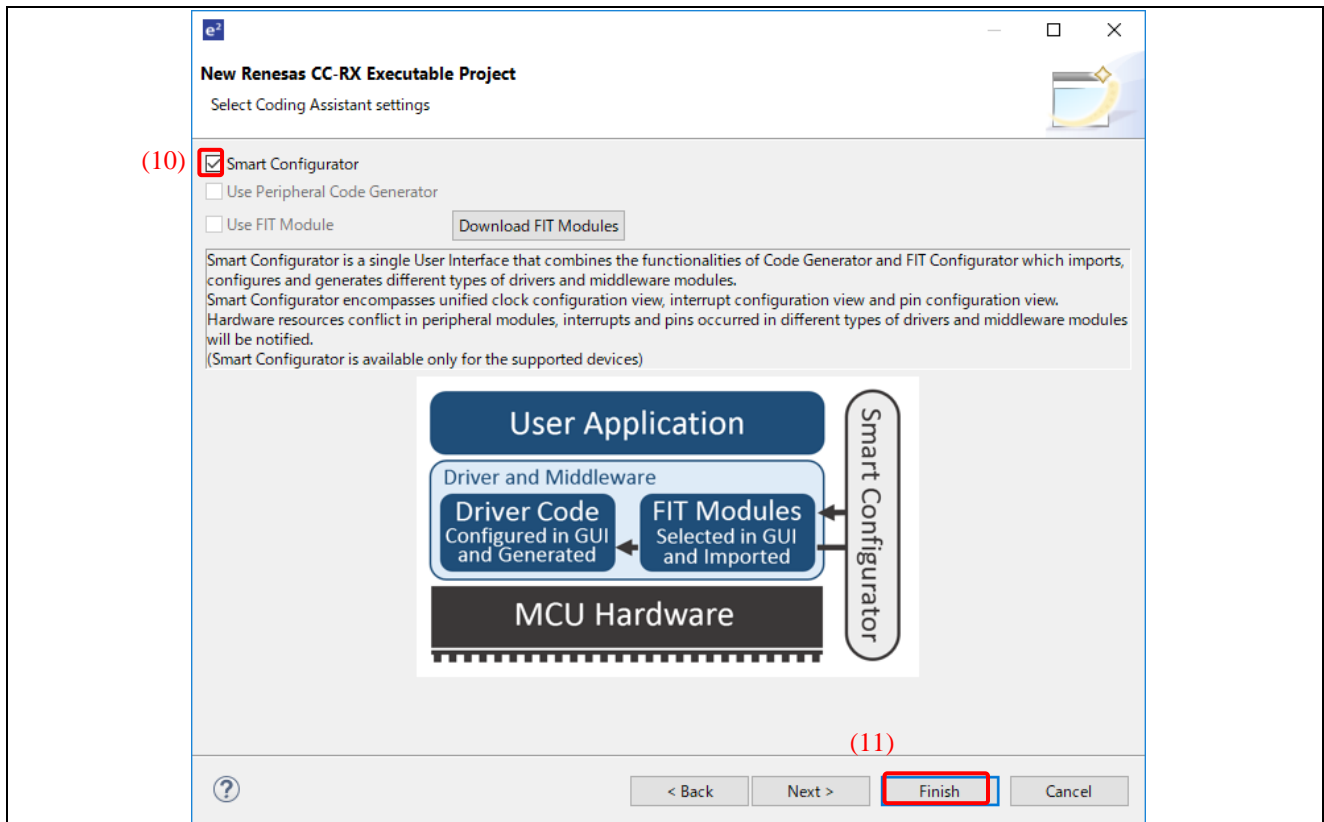


Figure 2-10 Select Coding Assistant Tool

2.3 Adding a peripheral driver

Smart Configurator perspective will be launched as shown below.

- 1) In Smart_Configurator_Example.scfg pane, click the [Components] tab

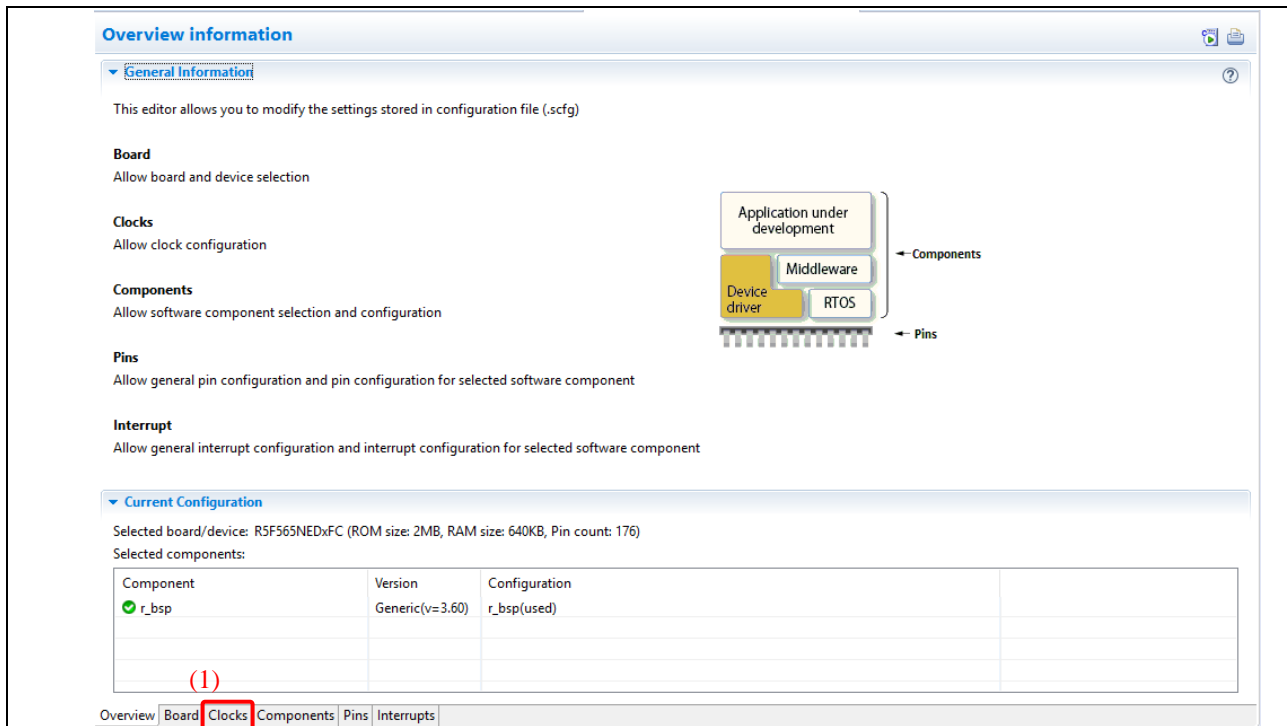


Figure 2-11 Smart Configurator perspective

- 2) Click  to add new component.

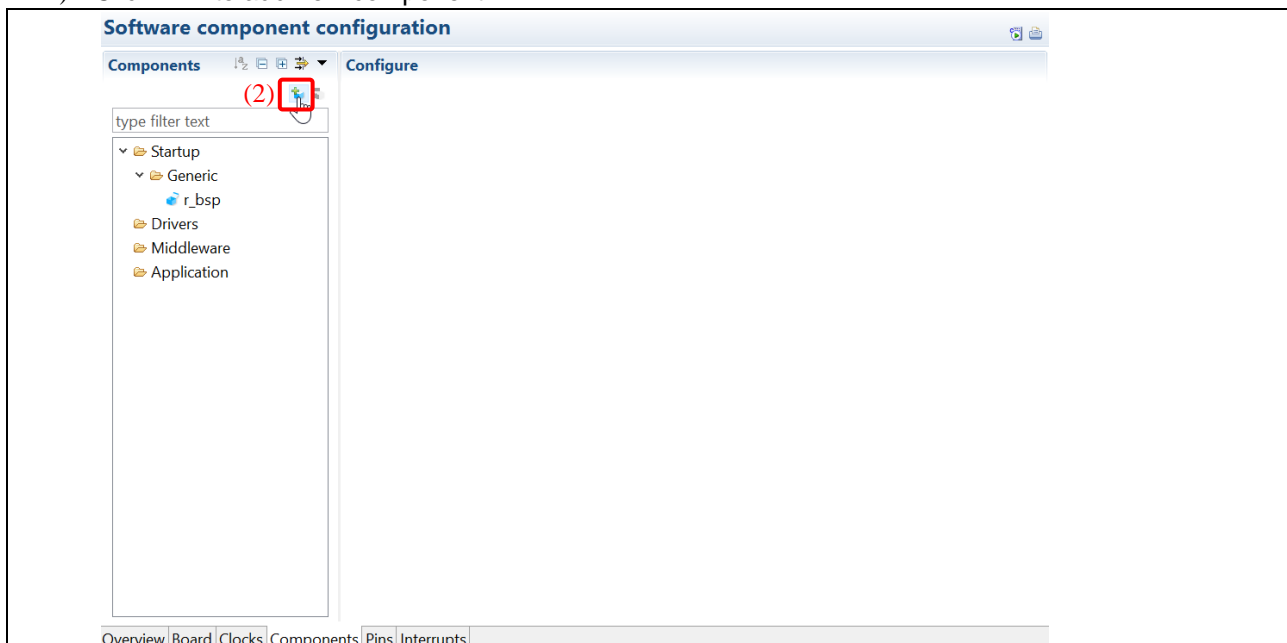


Figure 2-12 Software component configuration in Smart Configurator

- 3) Add Compare Match Timer driver into the project
 - a. Select “Timers (Drivers)” from the “Function” option
 - b. Navigate the component list and select *Compare Match Timer*
 - c. Click [Next] to continue

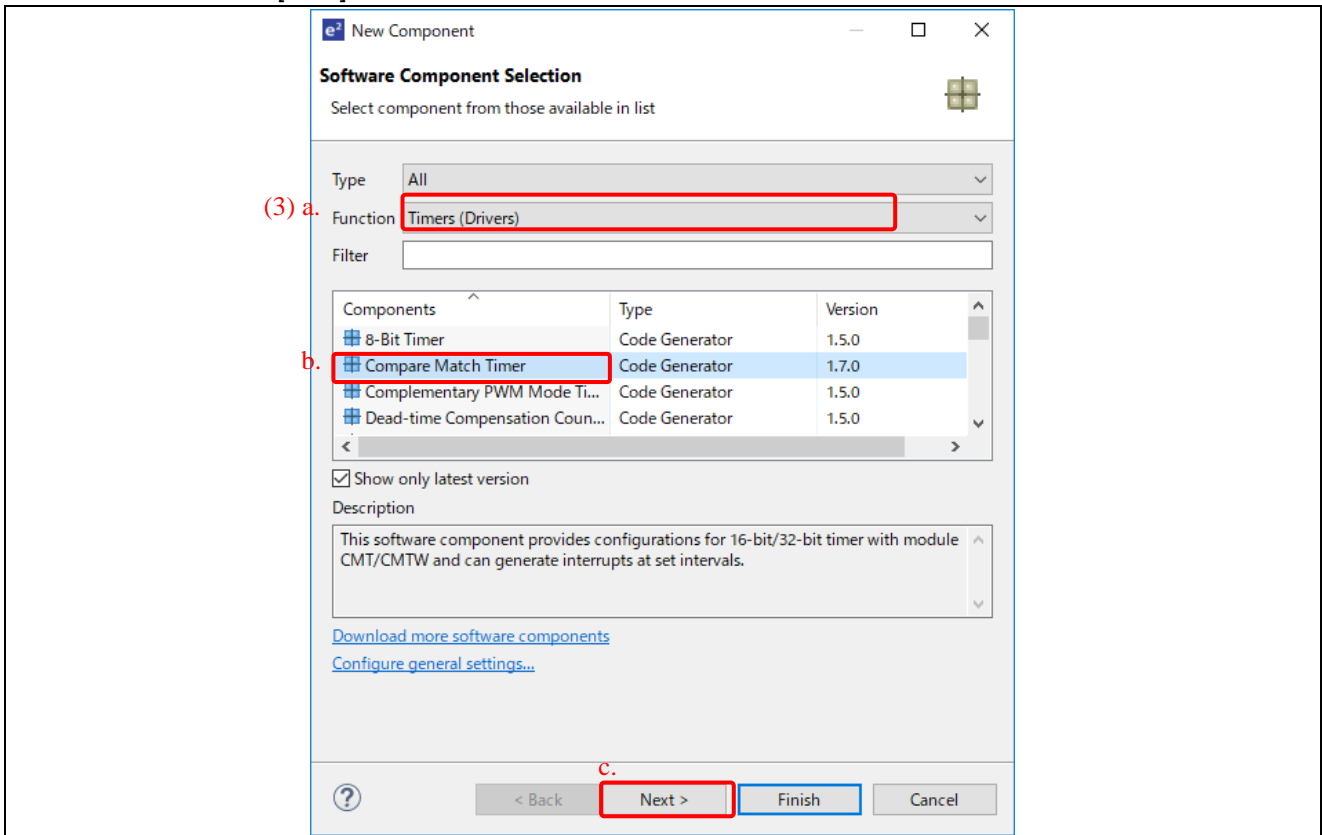


Figure 2-13 Select software component

- d. Select CMT0 as resource
- e. Keep the default configuration name. Source files and API will be generated based on this configuration name
- f. Click [Finish]

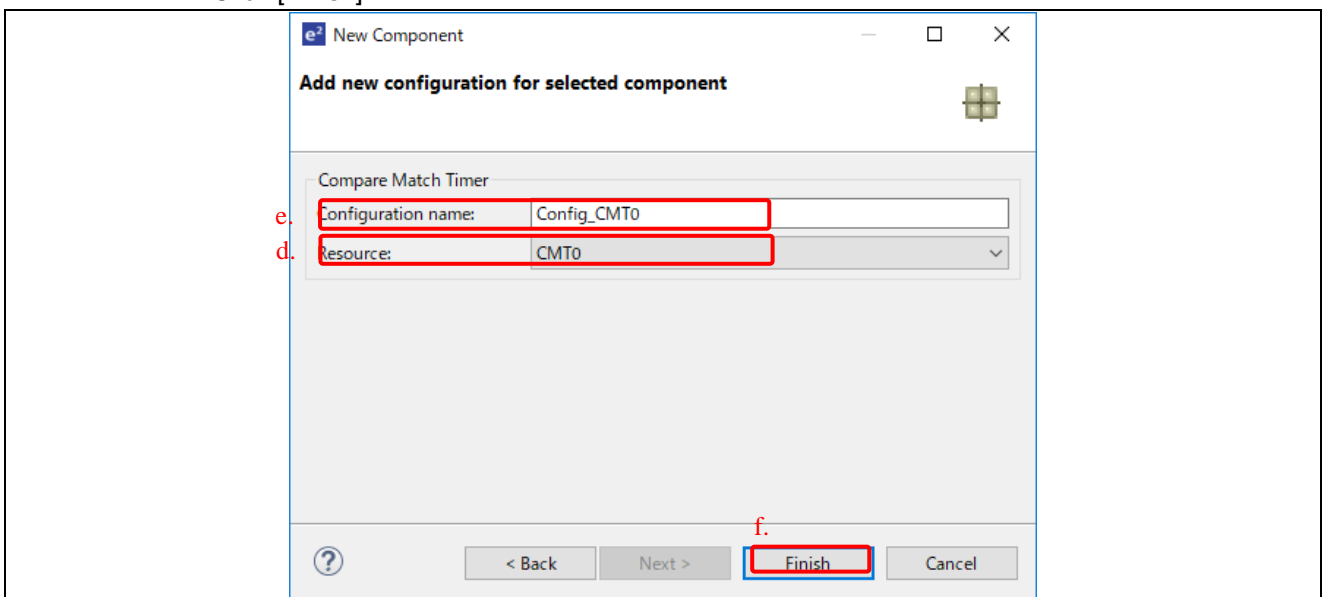


Figure 2-14 Add new configuration to the project

- 4) In the Configure panel of *Config_CMT0*,
 - a. Click Count clock setting as [PCLK/512] and set the Interval value to [5000 count]
 - b. Ensure the checkbox of "Enable compare match interrupt (CMI0)" is ticked

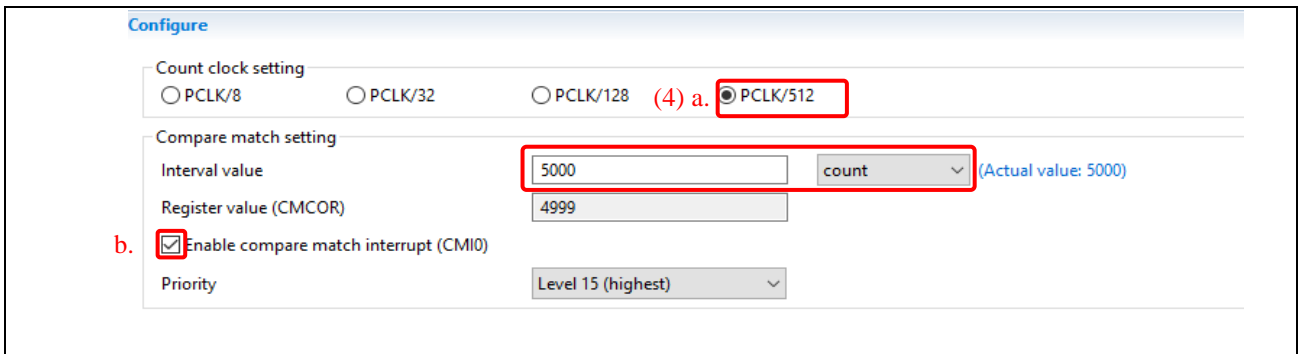



Figure 2-15 Compare Match Timer configuration

- 5) Add *Ports* driver into the project
 - a. At [Components] tab, click  to add *Ports*
 - b. Select "I/O Ports (Drivers)" from the "Function" option
 - c. Select "Ports" driver
 - d. Click [Next] to continue

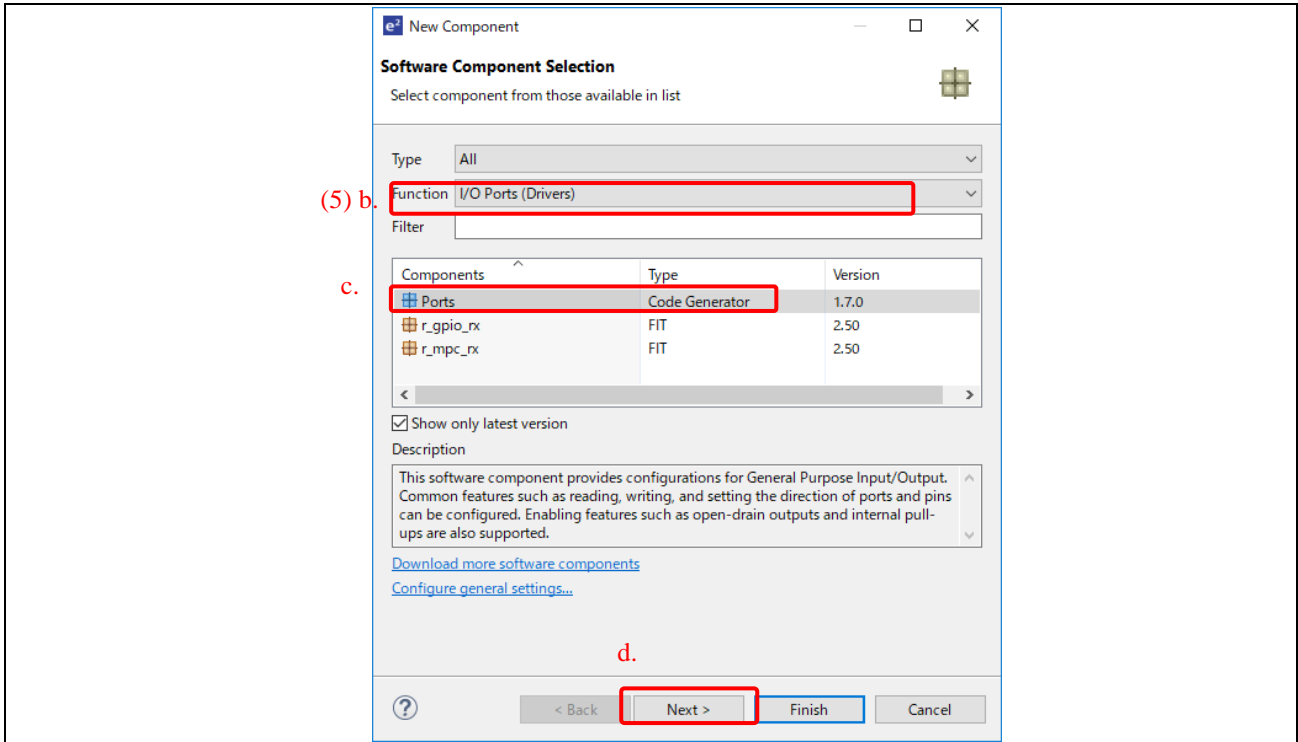


Figure 2-16 Select software component

- e. Keep the default configuration name and click [Finish]

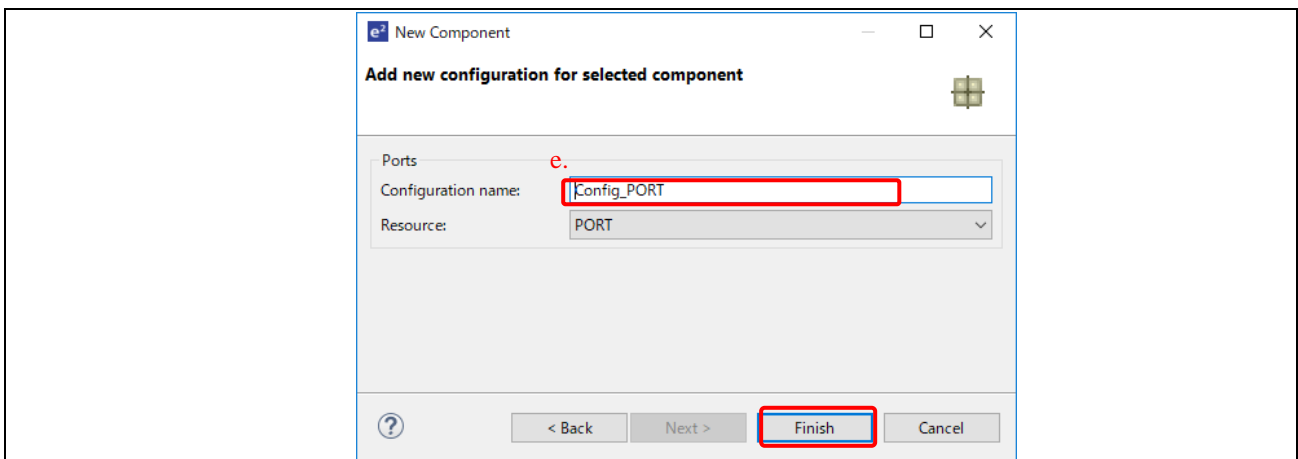


Figure 2-17 Add new configuration to the project

6) At [Port selection] tab, check PORTG checkbox.

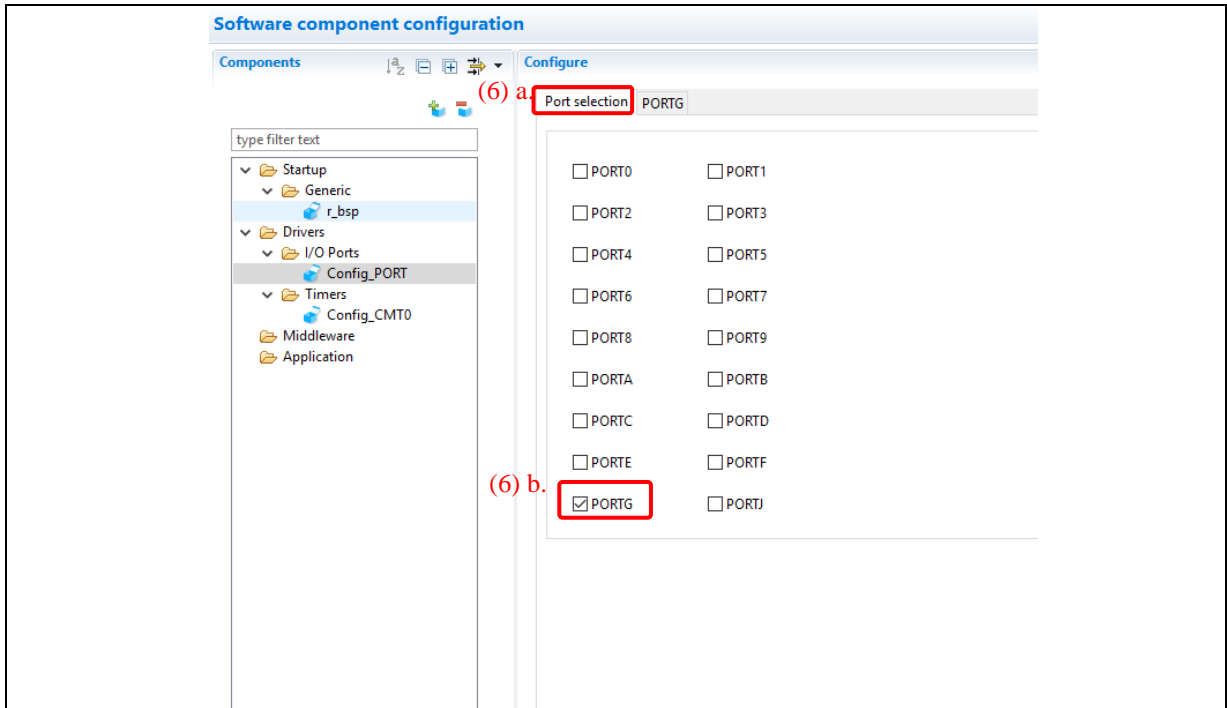


Figure 2-18 Port selection tab

7) At [PORTG] tab, set PG6 as output pin.

- a. At PG6 section, select [Out].
- b. Check [Output 1] checkbox to set LED2 to off status initially.

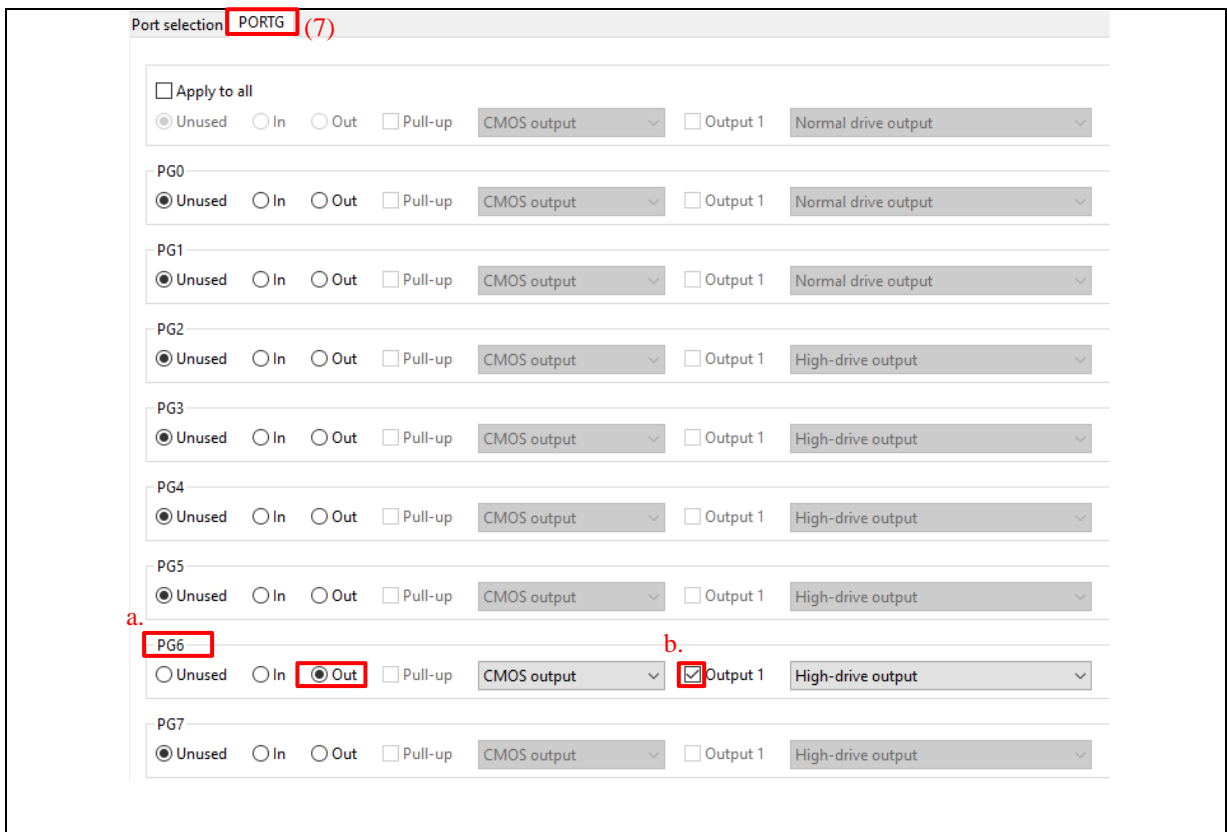



Figure 2-19 PORTG tab

- 8) At [Pins] tab, click  button to switch tree to Software Components view

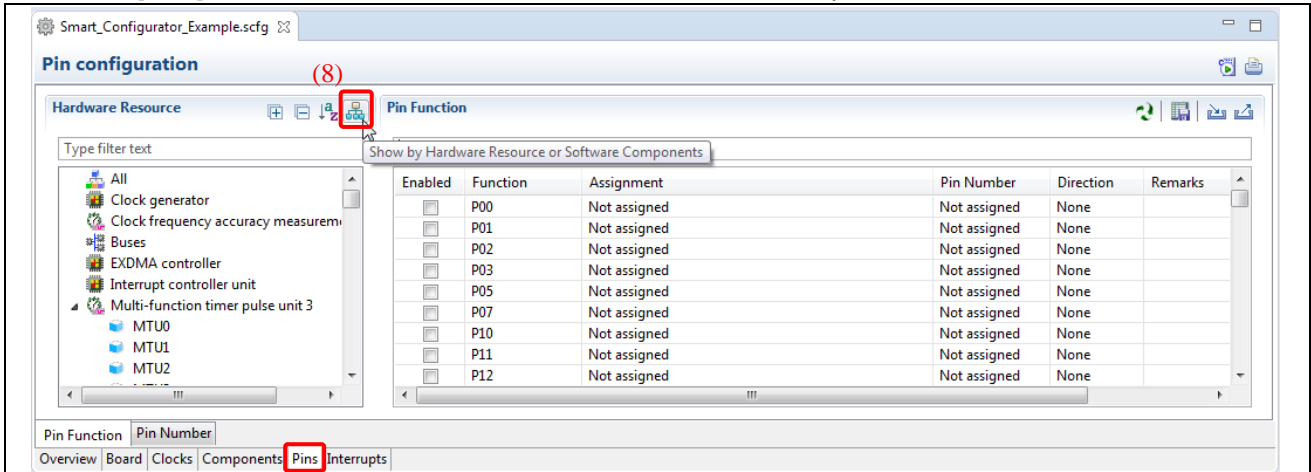


Figure 2-20 Pins tab

- 9) At the tree, click *Config_PORT* to view pin configurations
 10) Type “PG” in the Pin Function filter
 11) Ensure PG6 pin is enabled:

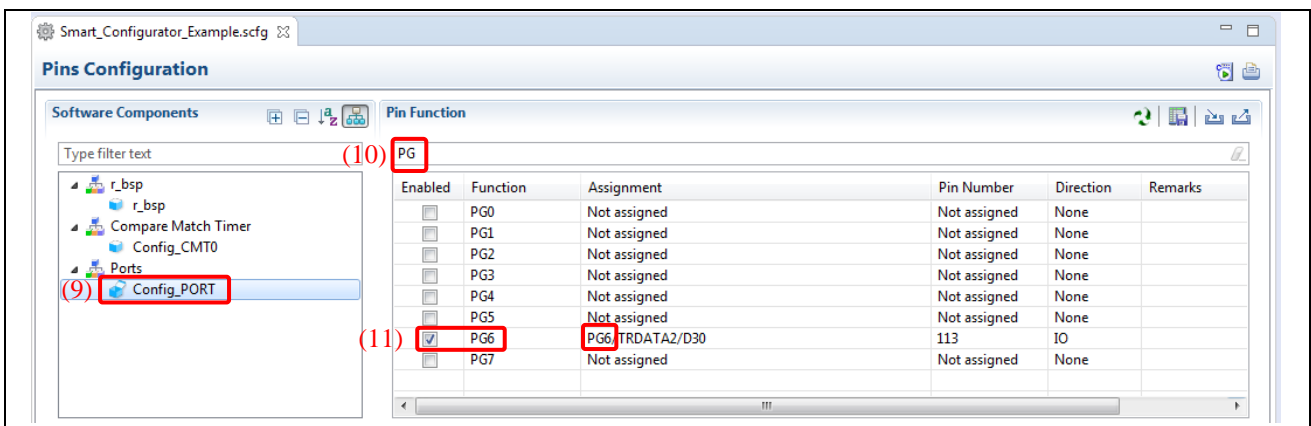



Figure 2-21 Pin configuration of *Config_PORT*

2.4 Generating codes

- 1) Click  to generate codes

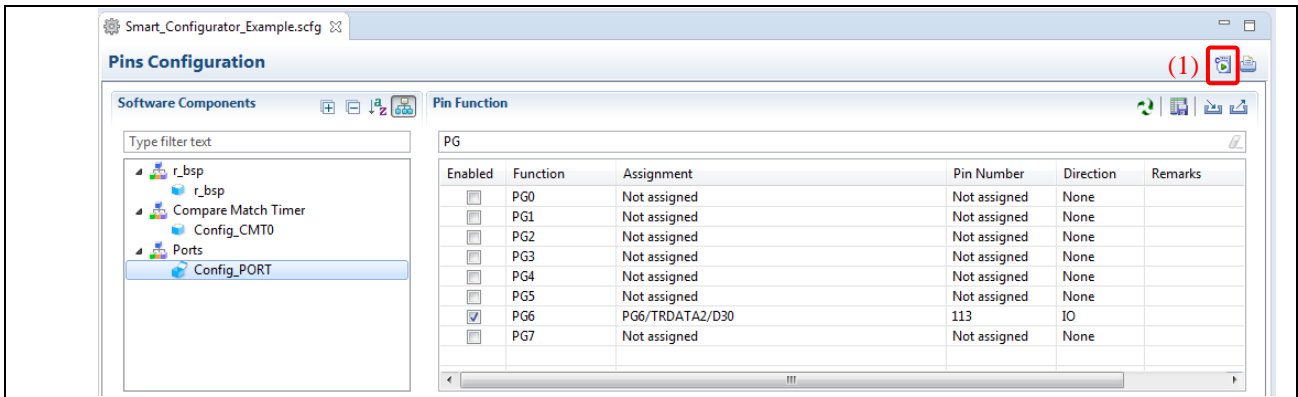


Figure 2-22 Generate codes

- 2) Message 'Code generation is successful will be shown at Console
- 3) Files generated into \src\smc_gen folder of the project

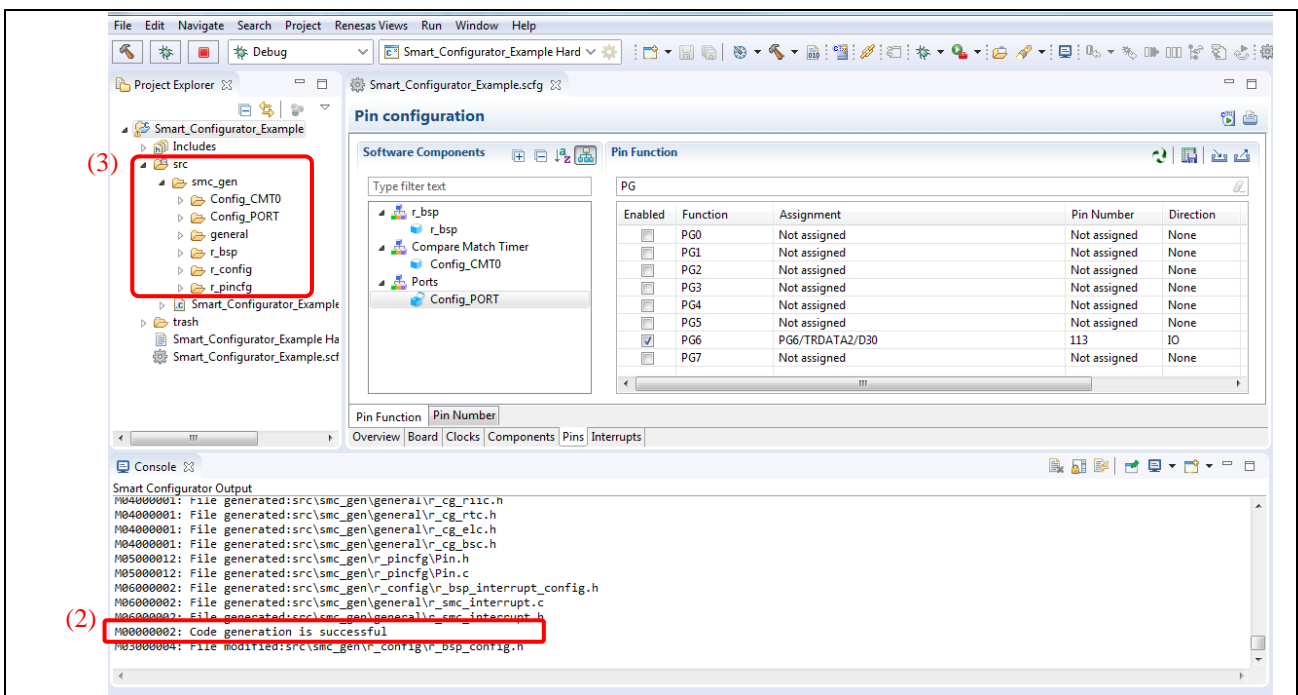


Figure 2-23 Successful code generation

2.5 Adding definition codes in “r_cg_userdefine.h” file

- 1) At the project tree, double-click to open “r_cg_userdefine.h” file in \src\smc_gen\general folder
 - a. Add LED2 port setting in the part between the beginning and the end of the comment as shown in the figure below

```

/* LED port setting */
#define LED2 (PORTG.PODR.BIT.B6)
    
```

Your source file should look like this:

```

/*****
Macro definitions (Register bit)
*****/
/* Start user code for register. Do not edit comment generated here */

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

/*****
Macro definitions
*****/
/* Start user code for macro define. Do not edit comment generated here */
/* LED port settings */
#define LED2 ((PORTG.PODR.BIT.B6))
/* End user code. Do not edit comment generated here */

/*****
Typedef definitions
*****/
/* Start user code for type define. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */

/*****
Global functions
*****/
/* Start user code for function. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
#endif
    
```

Figure 2-24 r_cg_userdefine.h

2.6 Adding application codes in Compare Match Timer driver

- 1) Open “Config_CMT0_user.c” in \src\smc_gen\Config_CMT0 folder of the project
 - a. Toggle LED2 in r_Config_CMT0_cmi0_interrupt (void) function

```

/* Toggle LED2 whenever timer is up */
LED2 ^= 1U;
    
```

Your source file should look like this:

```

/*****
 * Function Name: r_Config_CMT0_cmi0_interrupt
 * Description  : This function is CMI0 interrupt service routine
 * Arguments   : None
 * Return Value: None
 *****/
-#if FAST_INTERRUPT_VECTOR == VECT_CMT0_CMI0
-#pragma interrupt r_Config_CMT0_cmi0_interrupt(vect=VECT(CMT0,CMI0),fint)
-#else
-#pragma interrupt r_Config_CMT0_cmi0_interrupt(vect=VECT(CMT0,CMI0))
-#endif
-#static void r_Config_CMT0_cmi0_interrupt(void)
-#
-# {
-#     /* Start user code for r_Config_CMT0_cmi0_interrupt. Do not edit comment generated here */
-#     /* Toggle LED2 whenever timer is up */
-#     LED2 ^= 1U;
-#     /* End user code. Do not edit comment generated here */
-# }

```

Figure 2-25 Config_CMT0_user.c

2) Start CMT0 operation.

a. Open "Smart_Configurator_Example.c" in \src folder, add below code into *main()* function:

```

/* Start CMT0 counter operation */
R_Config_CMT0_Start();

while(1U)
{
    nop();
}

```

Your source file should look like this:

```

+* FILE      : Smart_Configurator_Example.c
#include "r_smc_entry.h"

void main(void);

-void main(void)
-#
-# {
-#     /* Start CMT0 counter operation */
-#     R_Config_CMT0_Start();
-#
-#     while(1U)
-#     {
-#         nop();
-#     }
-# }

```

Figure 2-26 Start CMT0 operation in main file

2.7 Build and run on hardware board

- 1) Build the project, click [Project] → [Build Project]

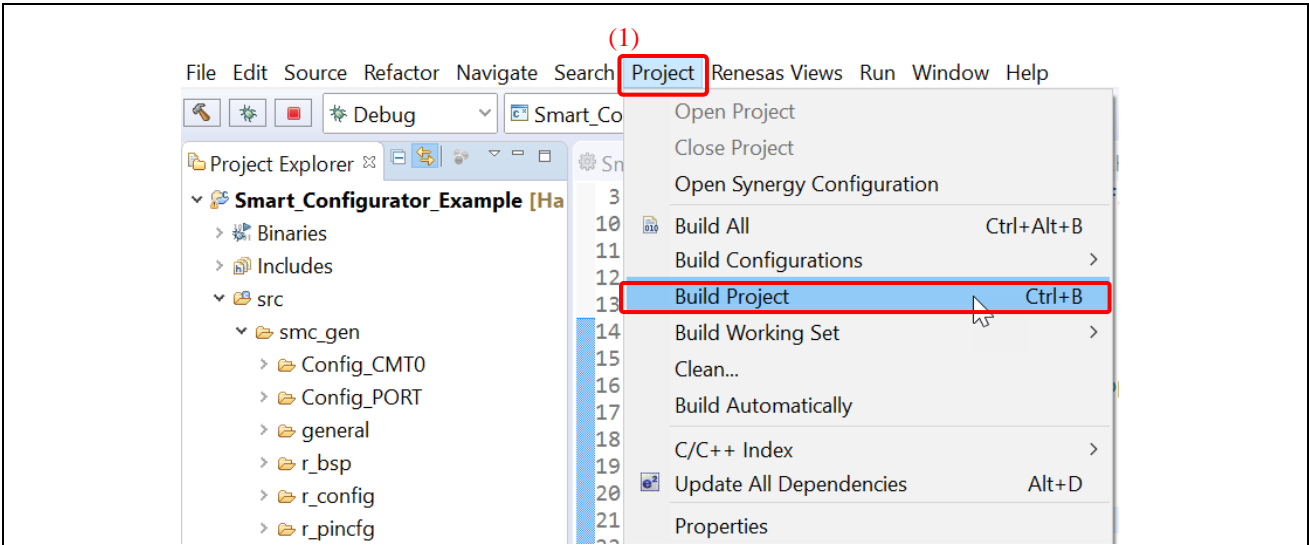



Figure 2-27 Build project

- 2) Debug the code
 - a. Connect RSK+ RX65N 2MB board to E1 / E2 Lite emulator and connect the E1 / E2 Lite emulator to PC
 - b. [Run] → [Debug Configurations...] or the downward arrow by the side of  icon → [Debug Configurations...] to open the “Debug Configurations” window

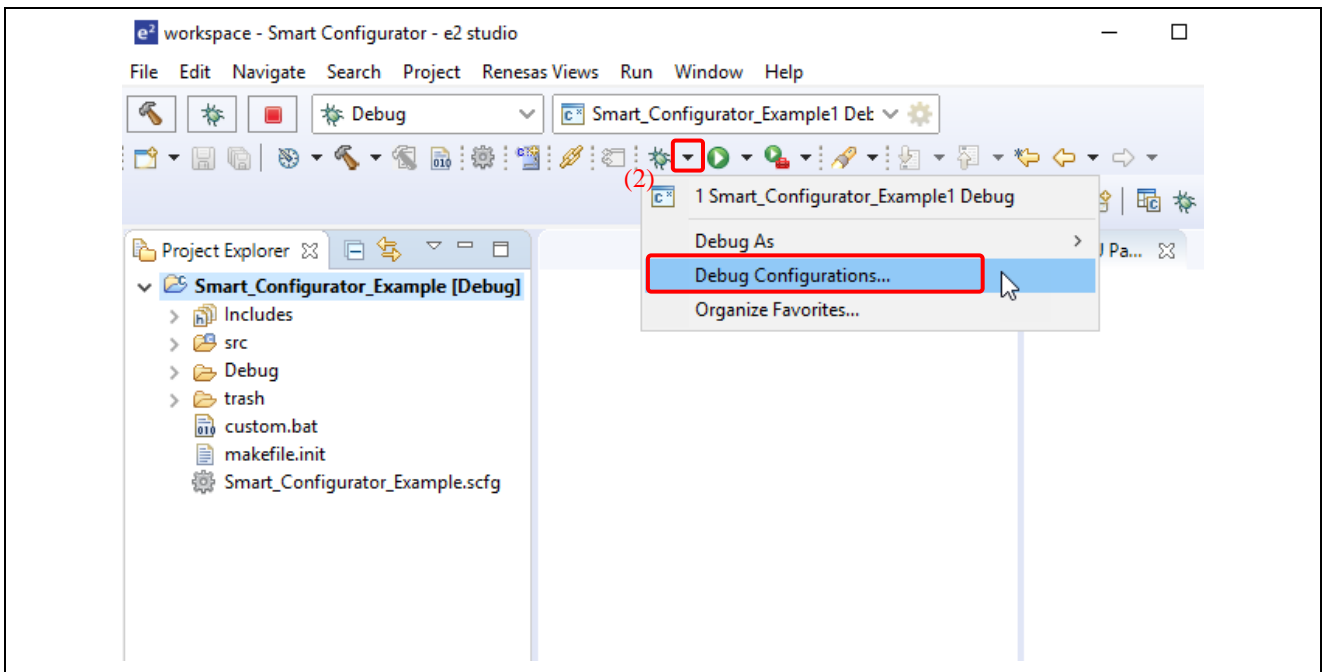


Figure 2-28 Open Debug Configurations window

- 3) Expand [Renesas GDB Hardware Debugging] and click on [Smart_Configurator_Example_Hardware Debug]. Go to the [Debugger] Tab, click on [Connection Settings] tab, make sure:
 - a. Debug Hardware: Select E1 (RX) or E2 Lite (RX)
 - b. Target Device: R5F565NEDxFC
 - c. Main Clock Source: EXTAL
 - d. Extal Frequency[MHz]: 24MHz
 - e. Permit Clock Source Change On Writing Internal Flash Memory: Yes
 - f. Power Target From The Emulator (MAX 200mA): Yes

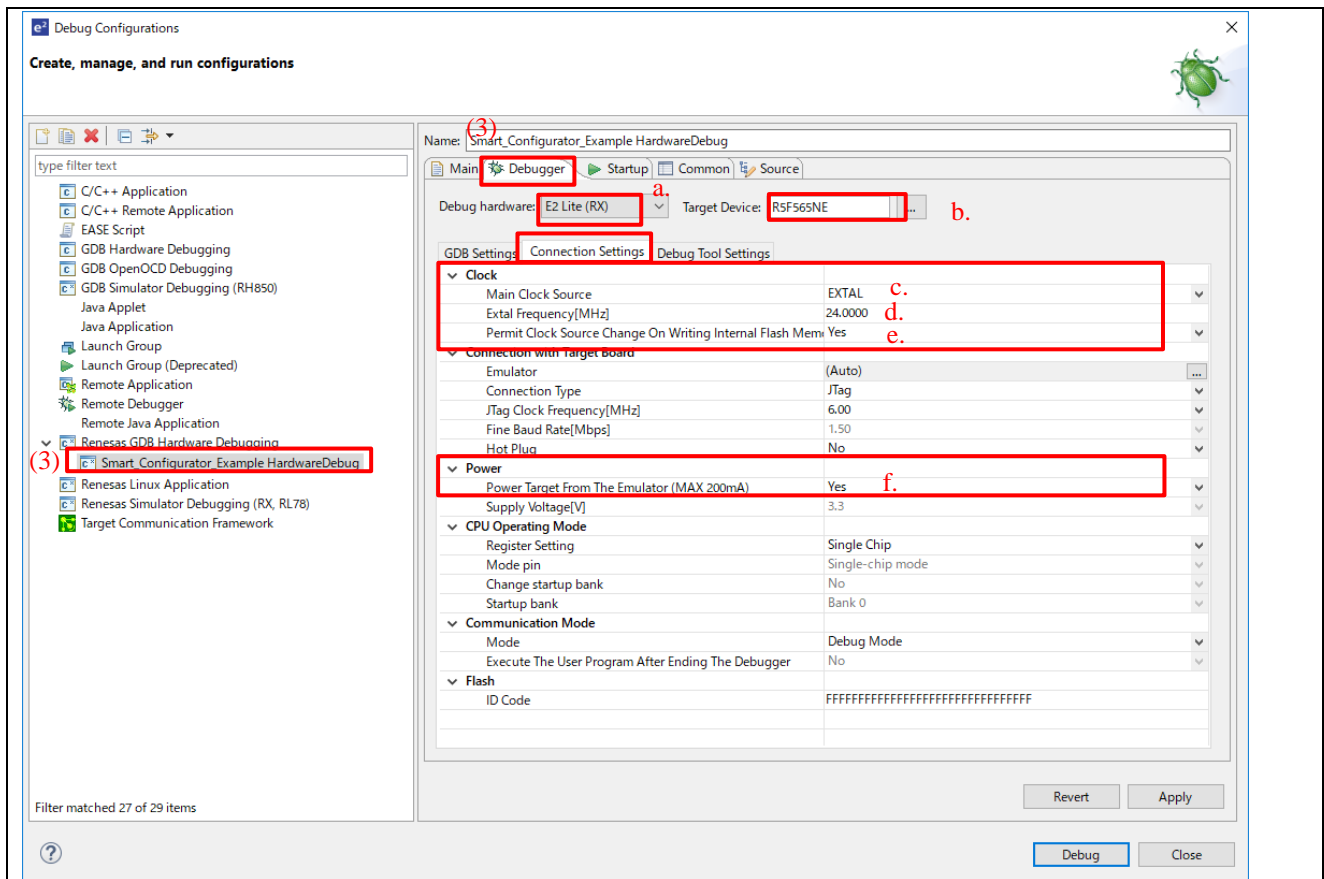


Figure 2-29 Debug Configurations example when connecting to E2 (Lite) emulator

- 4) At Startup, uncheck “Set breakpoint at: main”
- 5) Click [Debug] to start debugging.

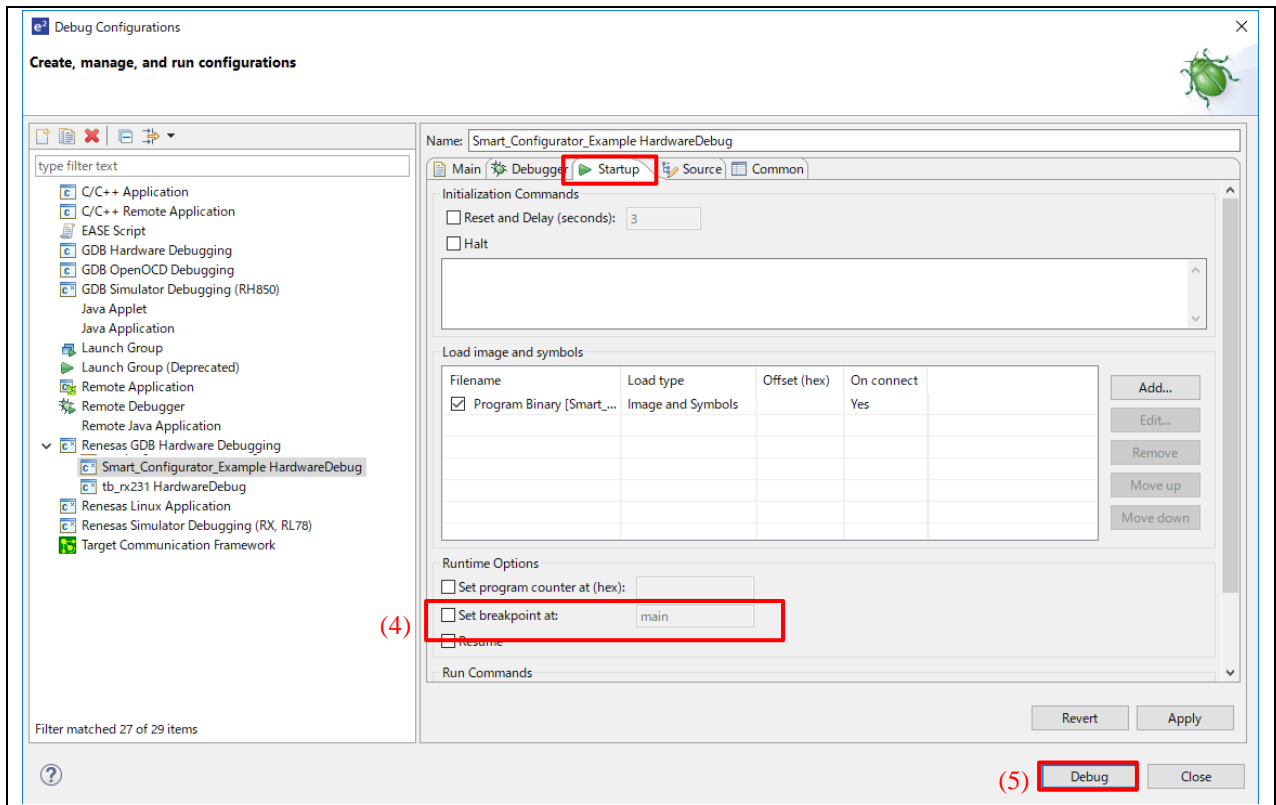



Figure 2-30 Remove breakpoint at main

- 6) ‘Confirm Perspective Switch’ dialog may pop up, click [Yes] to continue.
- 7) Click  to start project execution.

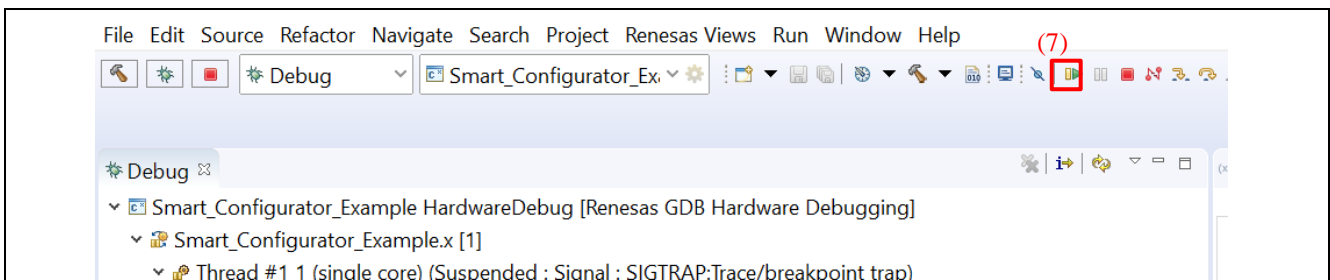


Figure 2-31 Start execution

- 8) Observed that LED2 is blinking. Refer to chapter 2.9 “Operation on board” for position of LED2.

9) To suspend the execution, click suspend button 

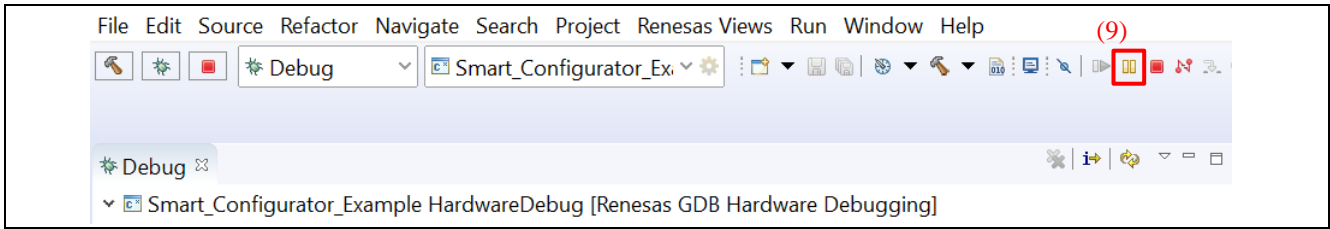



Figure 2-32 Suspend execution

10) To stop the execution, click disconnect button  to end debug session

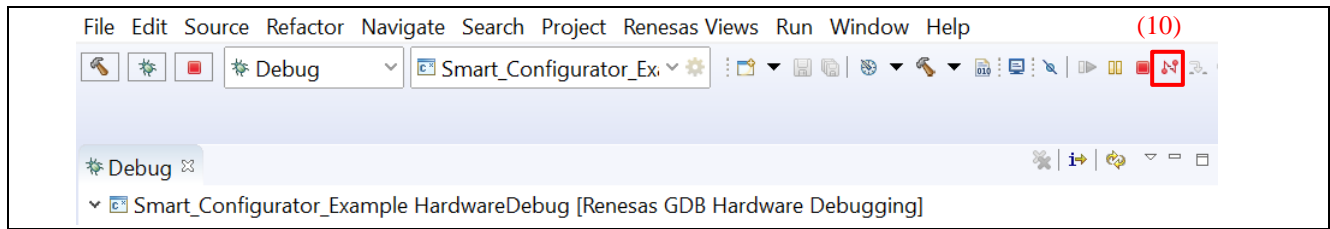


Figure 2-33 Stop debug

2.8 Changing LED2 blinking interval




- 1) Click  at the right hand side of the toolbar to switch to Smart Configurator perspective



Figure 2-34 Change to Smart Configurator perspective

- 2) In Smart_Configurator_Example.scfg pane, click the [Components] tab and select *Config_CMT0* from the tree
- 3) Change the Interval value to [50000 count] to slow down LED2 blinking speed
- 4) Click  on the toolbar to save changes and click  to regenerate codes

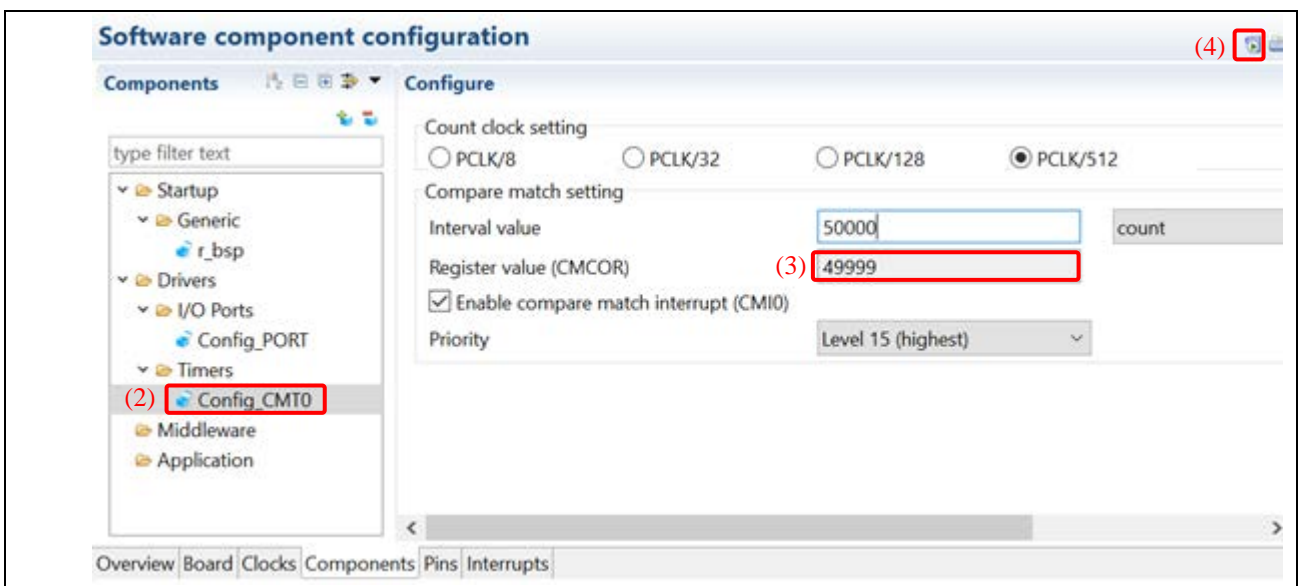


Figure 2-35 Compare Match Timer configuration

- 5) Click [Project] → [Build Project] to build the project
- 6) Click [Yes] to reload the updated project

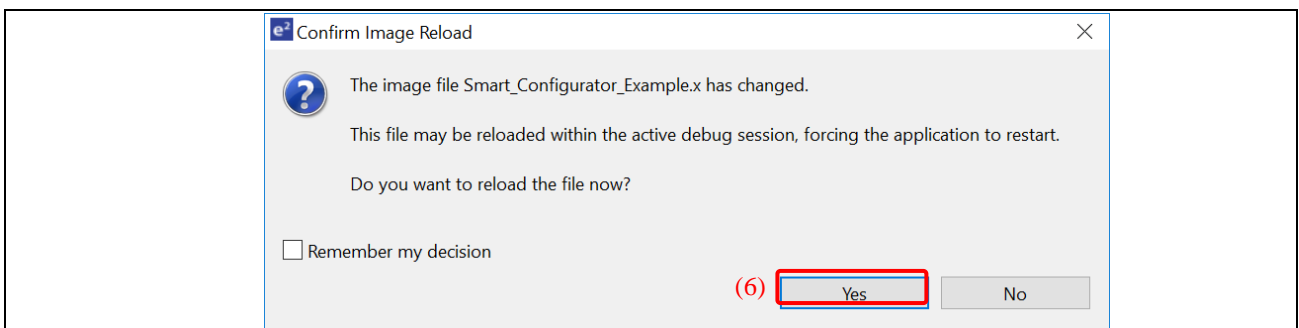



Figure 2-36 Confirm image reload

- 7) 'Confirm Perspective Switch' dialog may pop up, click [Yes] to continue.

8) Click  icon to start the execution

LED2 is blinking at a lower speed. Refer to chapter 2.9 “Operation on board” for position of LED2.

2.9 Operation on board

On RSK+ RX65N 2MB board, LED2 is blinking.

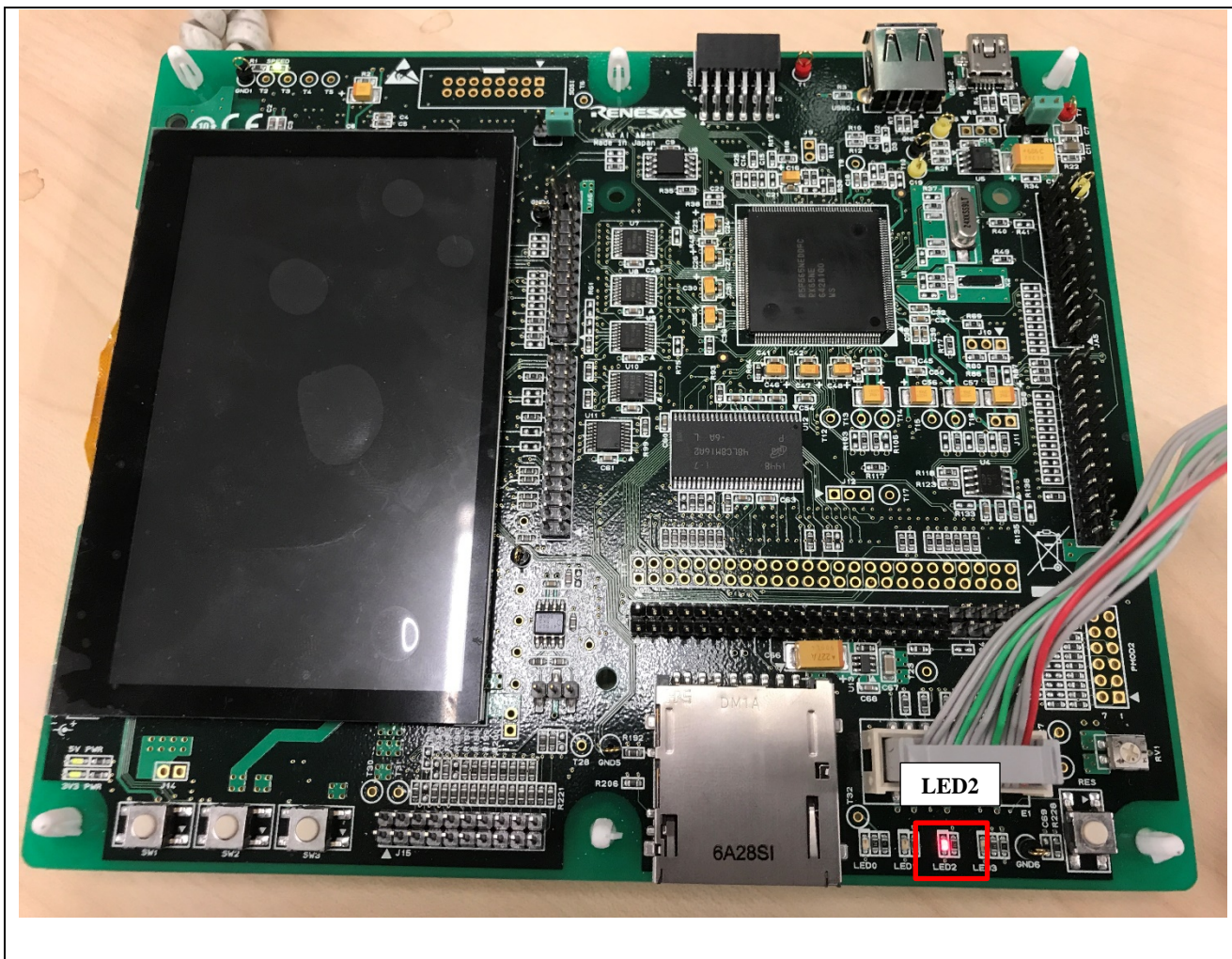


Figure 2-37 LEDs on RSK+ RX65N 2MB board

3. Application Example 2 (Adding a function using Smart Configurator)

This chapter describes how to use potentiometer to control the LED2 blinking rate.

The table below shows CG drivers to be configured in this example.

Drivers			Resource	Function
Single S12AD	Scan	Mode	S12AD	Read potentiometer value for controlling LED blinking rate

S12AD application codes will be added to the program as below:

a) Main function:

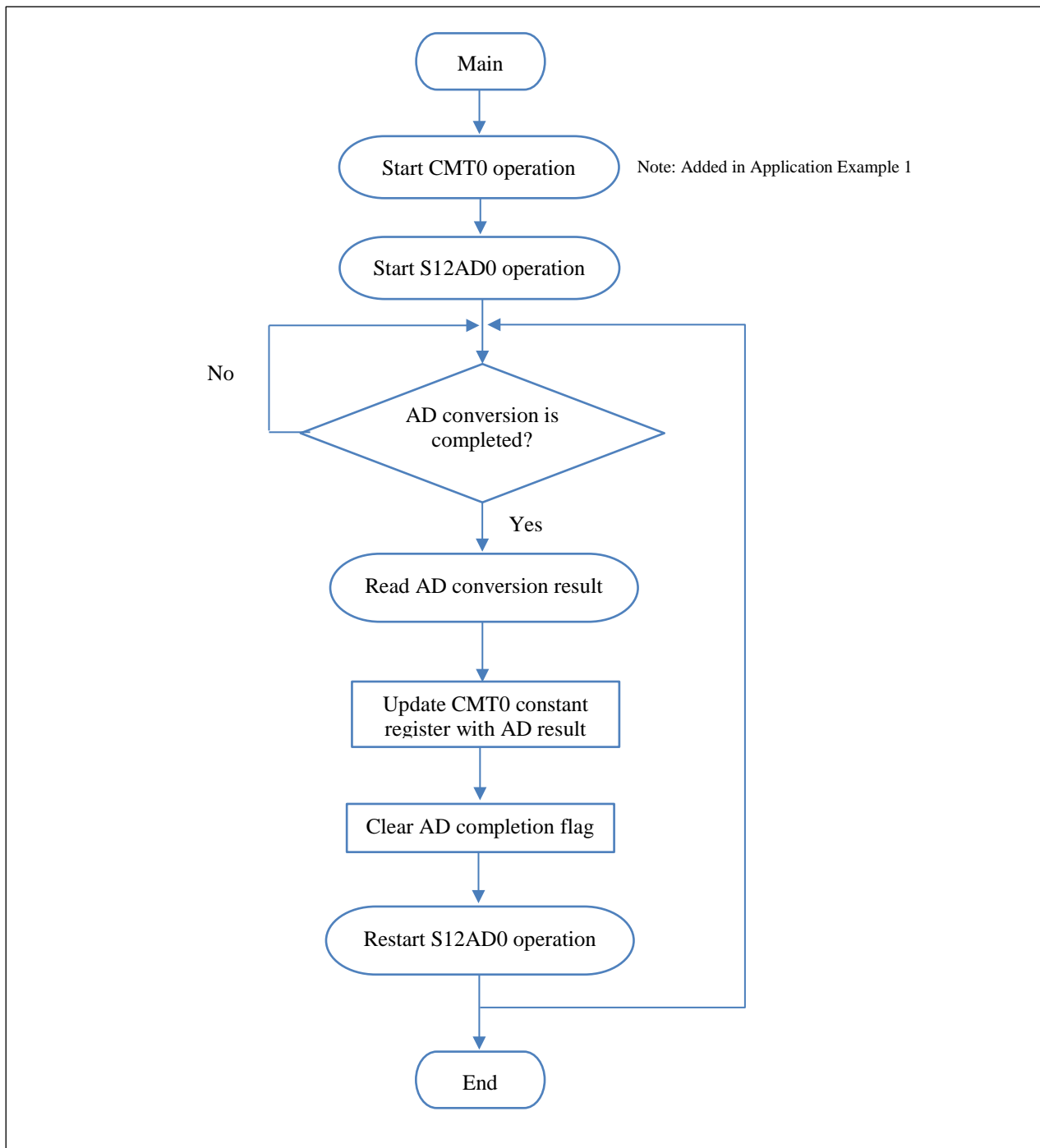


Figure 3-1 Main flowchart

b) S12AD0 user function:

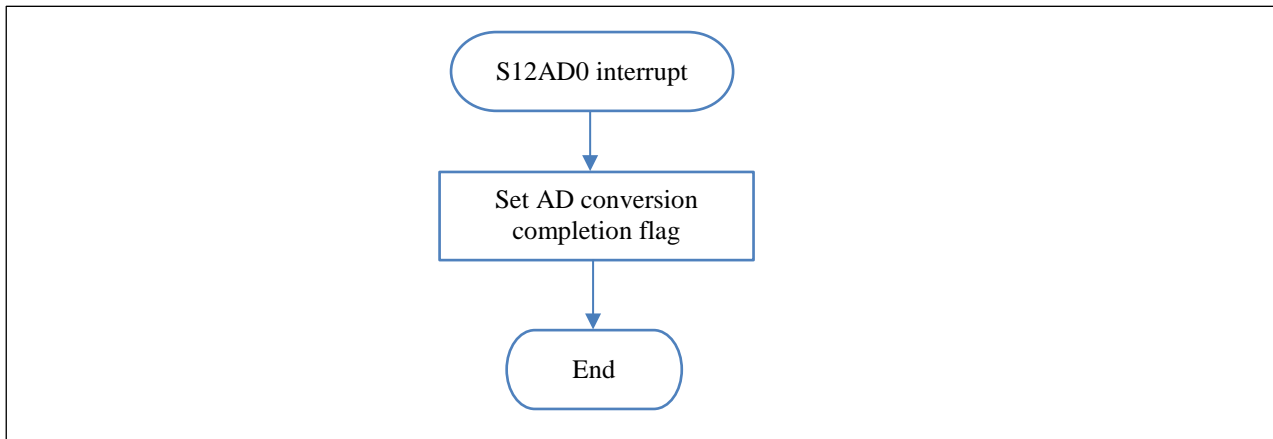



Figure 3-2 S12AD0 interrupt flowchart

3.1 Adding a peripheral driver

- 1) In Smart_Configurator_Example.scfg pane, click the [Components] tab
- 2) Click  to add new component

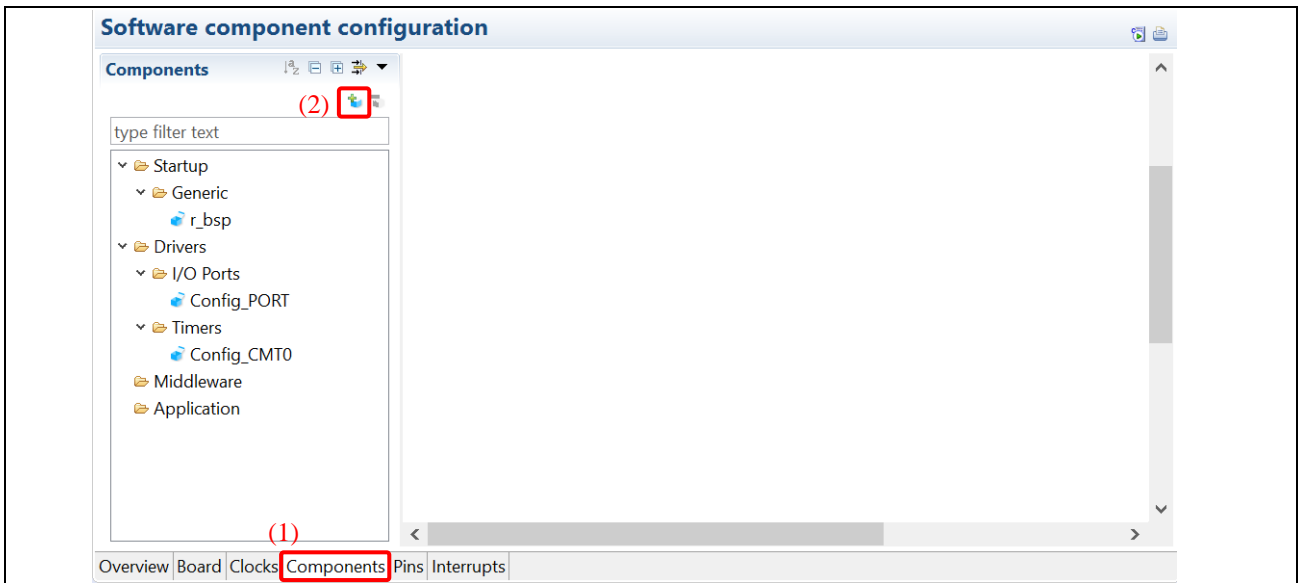


Figure 3-3 Software component configuration in Smart Configurator

- 3) Add *Single Scan Mode S12AD* driver into the project
 - a. Select “A/D Converter (Drivers)” from the “Function” option
 - b. Navigate the component list and select *Single Scan Mode S12AD*
 - c. Click [Next] to continue

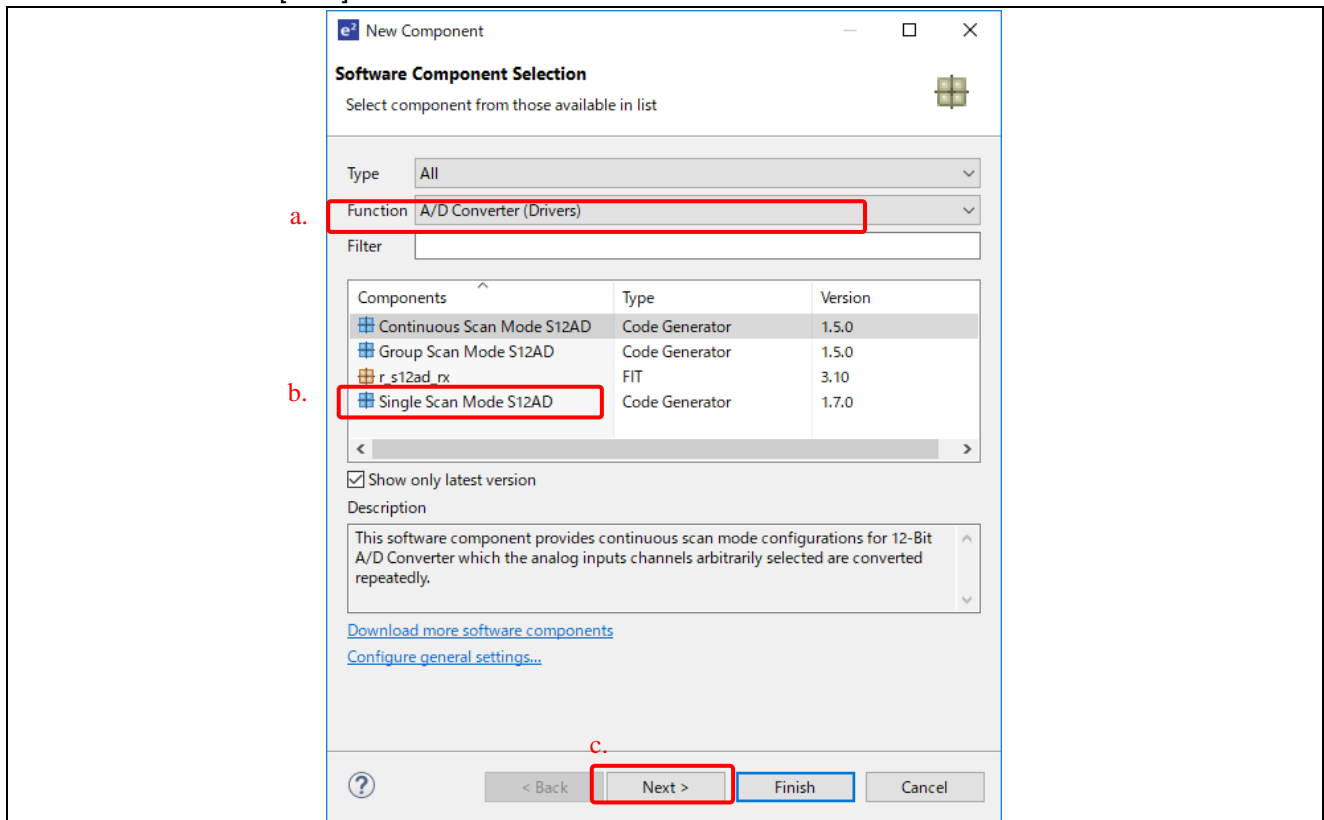


Figure 3-4 Select software component

- d. Select S12AD0 as resource
- e. Keep the default configuration name. Source files and API will be generated based on this configuration name
- f. Click [Finish]

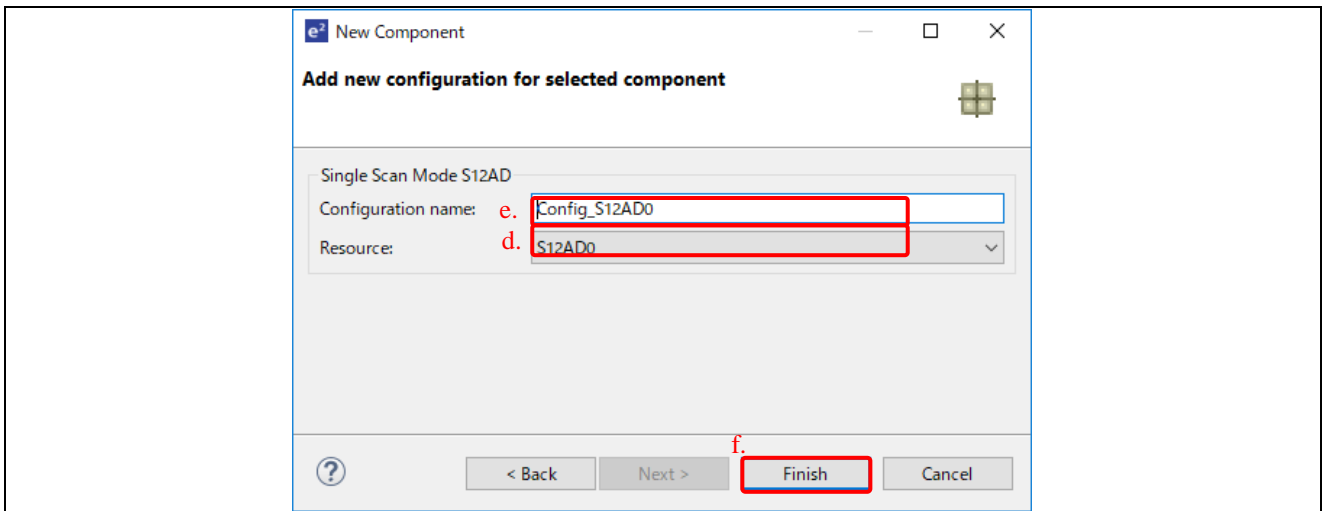


Figure 3-5 Add new configuration to the project

- 4) At Configure pane, select [AN000] at the Analog input channel setting
- 5) Ensure that the Start trigger source as [Software trigger] and the checkbox of [Enable AD conversion end interrupt (S12ADI)] is ticked

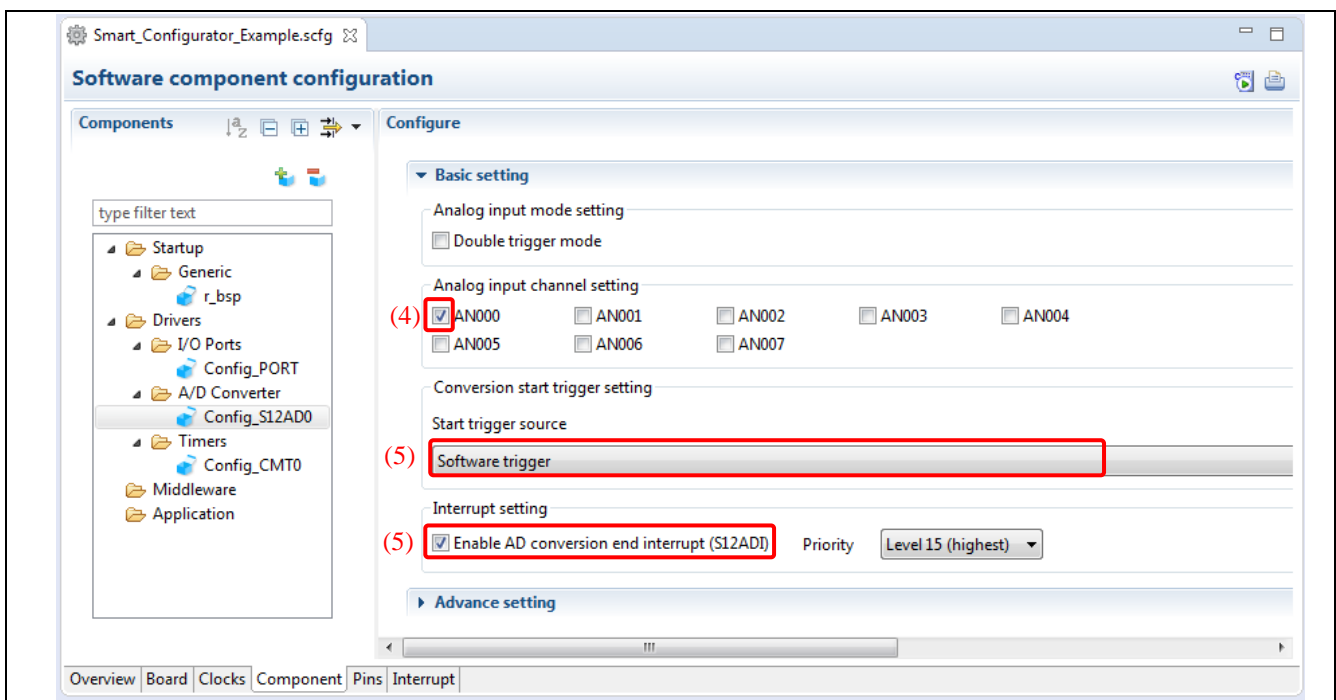


Figure 3-6 Single Scan Mode S12AD configuration

- 6) At [Pins] tab, click to select Config_S12AD0 from the tree
- 7) Clear typing in the Pin Function filter if there is any
- 8) Ensure AN000 pin is assigned to P40; Analog power pins (AVCC0, AVSS0, VREFH0, VREFL0) are enabled

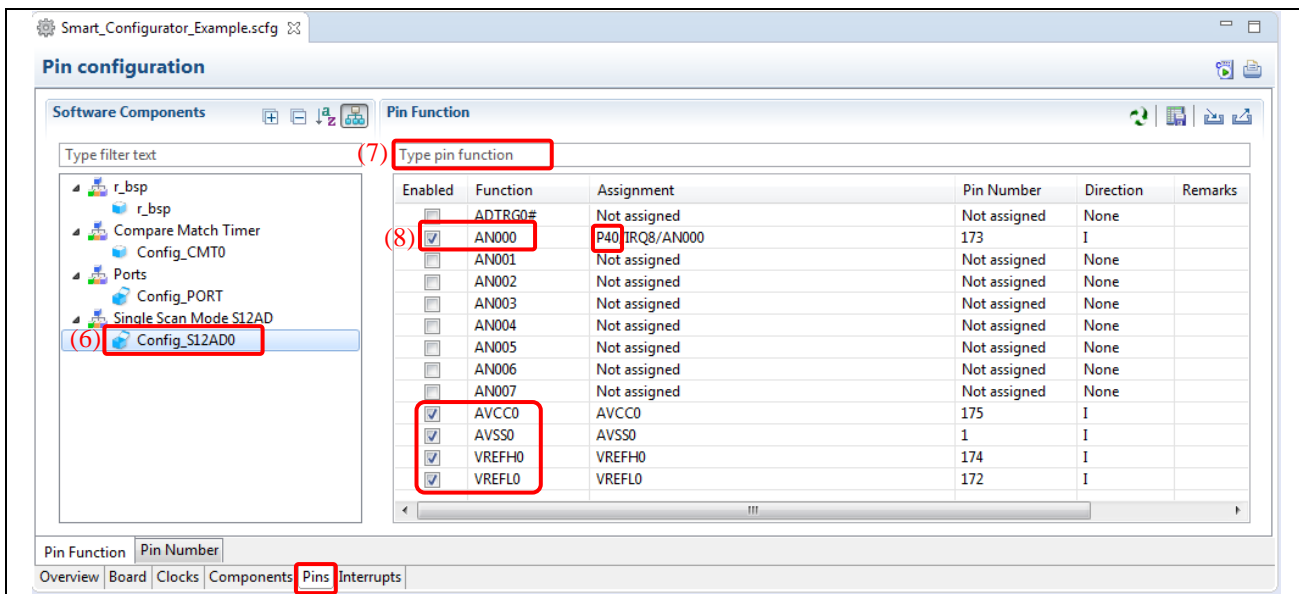


Figure 3-7 Pin configuration of Config_S12AD0

3.2 Generating codes

- 1) Click  to generate codes

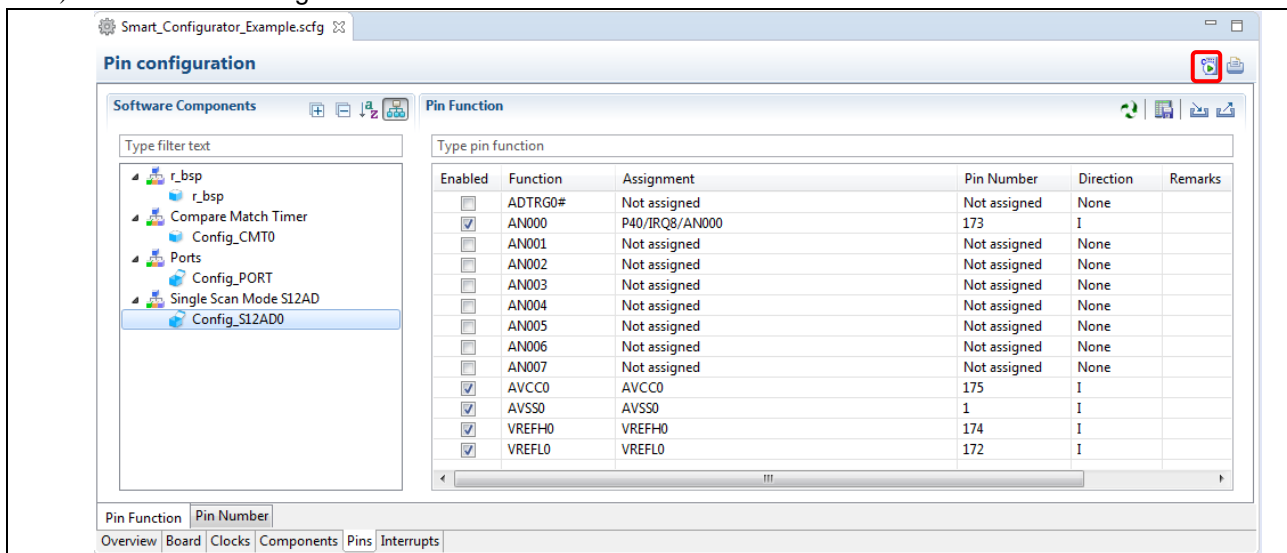


Figure 3-8 Generate codes

3.3 Adding application codes in Single Scan Mode S12AD driver

- 1) Open "Config_S12AD0_user.c" in \src\smc_gen Config_S12AD0 folder of the project
 - a. Add declaration of global flag and variable at the user code area near the top of the file

```
/* Global flag to indicate A/D conversion operation is completed */
uint8_t g_adc_flag;
```

- b. Set A/D conversion completion flag in *r_Config_S12AD0_interrupt* (void) function

```
/* Set A/D conversion completion flag */
g_adc_flag = 1U;
```

Your source file should look like this:

```
+ * File Name      : Config_S12AD0_user.c
+ Pragma directive
- /* Start user code for pragma. Do not edit comment generated here */
  /* End user code. Do not edit comment generated here */
+ Includes
  #include "r_cg_macrodriver.h"
  #include "r_cg_userdefine.h"
  #include "Config_S12AD0.h"
- /* Start user code for include. Do not edit comment generated here */
  /* End user code. Do not edit comment generated here */
+ Global variables and functions
- /* Start user code for global. Do not edit comment generated here */
  /* Global flag to indicate A/D conversion operation is completed */
  uint8_t g_adc_flag;
  /* End user code. Do not edit comment generated here */
+ * Function Name: R_Config_S12AD0_Create_UserInit
- void R_Config_S12AD0_Create_UserInit(void)
  {
    /* Start user code for user init. Do not edit comment generated here */
    /* End user code. Do not edit comment generated here */
  }
+ * Function Name: r_Config_S12AD0_interrupt
- #if FAST_INTERRUPT_VECTOR == VECT_PERIB_INTB186
  #pragma interrupt r_Config_S12AD0_interrupt(vect=VECT(PERIB,INTB186),fint)
- #else
  #pragma interrupt r_Config_S12AD0_interrupt(vect=VECT(PERIB,INTB186))
  #endif
- static void r_Config_S12AD0_interrupt(void)
  {
    /* Start user code for r_Config_S12AD0_interrupt. Do not edit comment generated here */
    /* Set A/D conversion completion flag */
    g_adc_flag = 1U;
    /* End user code. Do not edit comment generated here */
  }
- /* Start user code for adding. Do not edit comment generated here */
  /* End user code. Do not edit comment generated here */
```

Figure 3-8 Config_S12AD0_user.c

3.4 Adding application codes in *main()*

- 1) Start CMT0 and S12AD0 operation
 - a. At the project tree, open "Smart_Configurator_Example.c" in \src folder
 - b. Add declarations of global variables at the user code area near the top of the file after

```
#include "r_smc_entry.h"
```

```
/* Global variable for changing CMT0 interval */
uint16_t interval_level;

/* Global variable for storing the A/D conversion result */
uint16_t g_adc_result;

/* Global flag to indicate A/D conversion operation is completed */
extern uint8_t g_adc_flag;
```

- c. Add below code into *main()* function before the *while* loop to start S12AD0 operation:

```
/* Start performing A/D conversion */
R_Config_S12AD0_Start();
```

- d. Read A/D conversion result and update into CMCOR constant register when A/D conversion is completed

Replace the *nop()* instruction in the *while* loop with the codes below:

```
/* A/D conversion is completed */
if (g_adc_flag == 1U)
{
    /* Read the A/D conversion result and store in variable */
    R_Config_S12AD0_Get

t_ValueResult(ADCHANNEL0, &g_adc_result);

    /* Get new blink interval level from A/D conversion result */
    interval_level = g_adc_result / 500;

    /* Limit minimum level to 1 */
    if (interval_level < 1)
    {
        interval_level = 1;
    }
    else
    {
        /* Do nothing*/
    }

    /* Change blinking rate */
    CMT0.CMCOR = (uint16_t)(5000 * interval_level);

    /* Reset A/D conversion flag */
    g_adc_flag = 0U;

    /* Restart A/D conversion operation */
    R_Config_S12AD0_Start();
```

Your source file should look like this:

```

#include "r_smc_entry.h"

/* Global variable for changing CMT0 interval */
uint16_t interval_level;

/* Global variable for storing the A/D conversion result */
uint16_t g_adc_result;

/* Global flag to indicate A/D conversion operation is completed */
extern uint8_t g_adc_flag;

void main(void);

void main(void)
{
    /* Start CMT0 counter operation */
    R_Config_CMT0_Start();

    /* Start performing A/D conversion */
    R_Config_S12AD0_Start();

    while(1U)
    {
        /* A/D conversion is completed */
        if (g_adc_flag == 1U)
        {
            /* Read the A/D conversion result and store in variable */
            R_Config_S12AD0_Get_ValueResult(ADCHANNEL0, &g_adc_result);

            /* Get new blink interval level from A/D conversion result */
            interval_level = g_adc_result / 500;

            /* Limit minimum level to 1 */
            if (interval_level < 1)
            {
                interval_level = 1;
            }
            else
            {
                /* Do nothing*/
            }

            /* Change blinking rate */
            CMT0.CMCOR = (uint16_t)(5000 * interval_level);

            /* Reset A/D conversion flag */
            g_adc_flag = 0U;

            /* Restart A/D conversion operation */
            R_Config_S12AD0_Start();
        }
        else
        {
            /* Do nothing*/
        }
    }
}

```

Figure 3-9 Start CMT0 and S12AD0 operation in *main* code

3.5 Build and run on hardware board

Refer to *Chapter 2.7* to build and debug this project.

3.6 Operation on board

On the RSK+ RX65N 2MB board, LED2 blinking rate is changing when turning the potentiometer

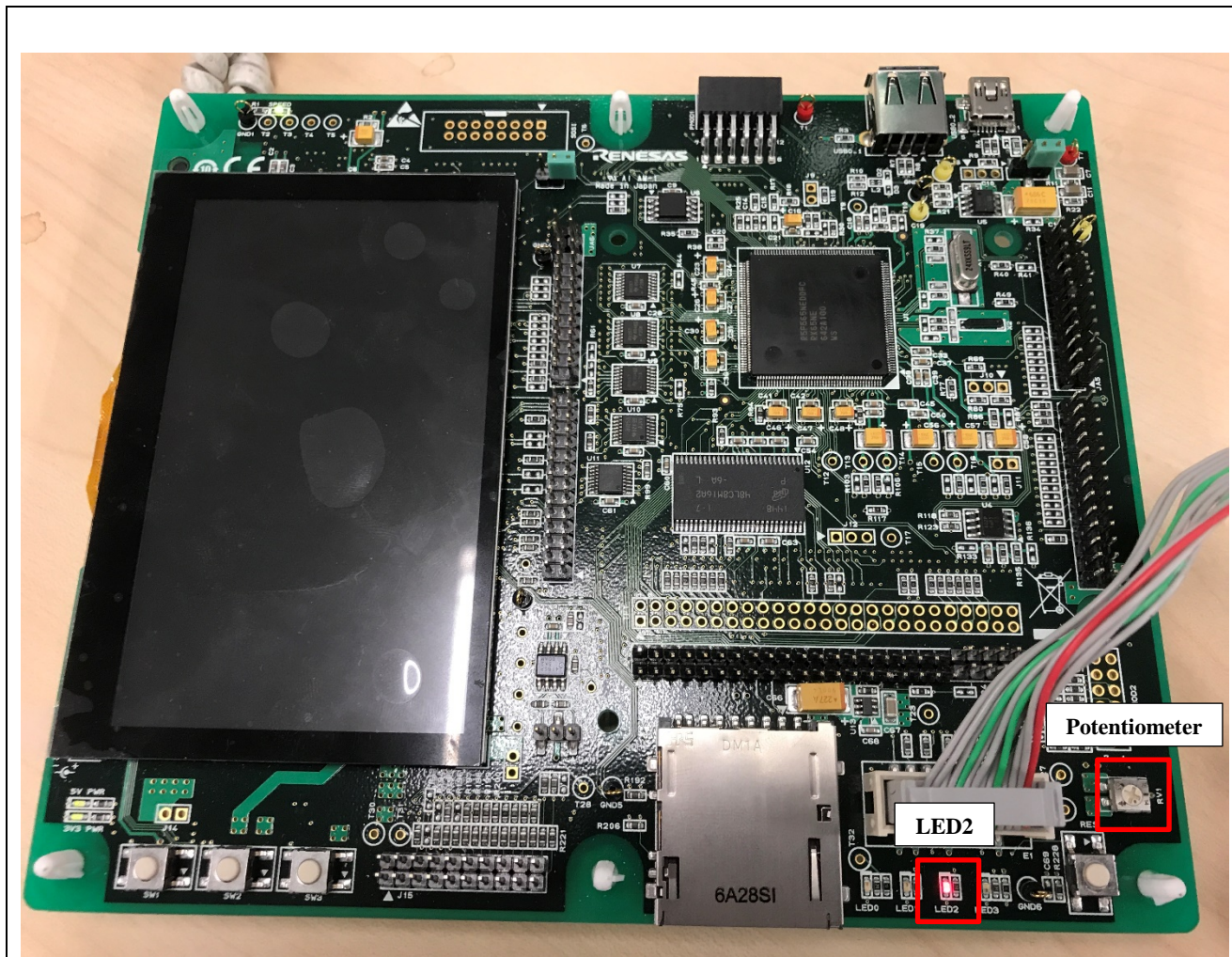


Figure 3-10 LEDs and Potentiometer on RSK+ RX65N 2MB board

4. Application Example 3 (Changing a function in Smart Configurator)

In previous chapter, potentiometer is used to control LED2 blinking rate.

This chapter describes how to control LED2 blinking rate with terminal program on PC via serial communication. In this application example, Serial Communication Interface module (SCI8) of RX65N is connected to PC via a USB-to-serial converter (implemented in RL78/G1C) on RSK+ board.

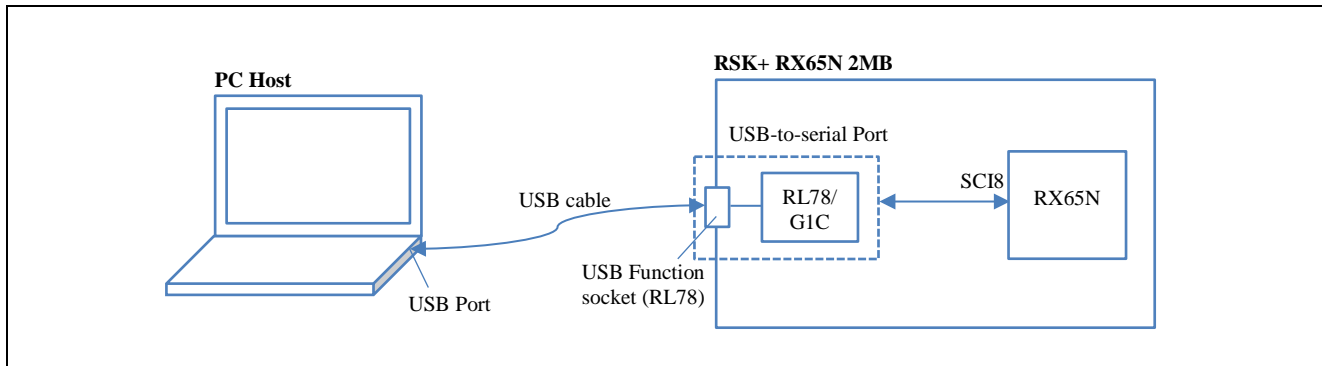


Figure 4-1 Control LED blinking rate by terminal program via serial communication

The schematic diagram of Renesas Starter Kit+ for RX65N 2MB is shown below. In this example, SCI8 will be configured to communicate with RL78/G1C.

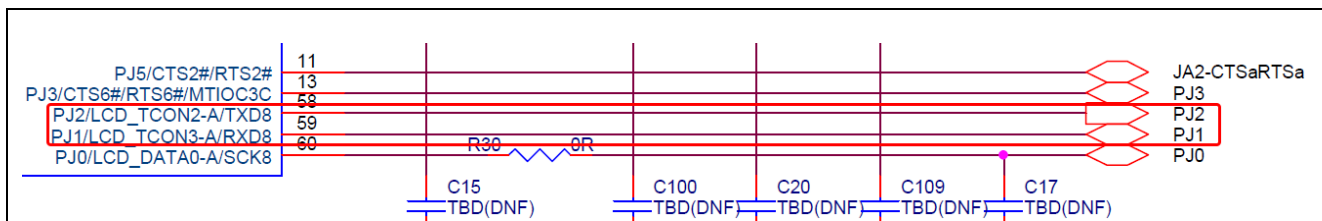


Figure 4-2 Schematic diagram of Renesas Starter Kit+ for RX65N 2MB

The table below shows changes of the driver selection:

	Drivers	Resource	Function
1. Remove driver	Single Scan Mode S12AD	S12AD	Read potentiometer value for controlling LED blinking rate
2. Add new driver	SCI/SCIF Asynchronous Mode	SCI8	Communication between RX65N and PC terminal via an on-board USB-to-serial converter (RL78/G1C) Data received from PC terminal is used to control LED blinking rate

SCI/SCIF Asynchronous Mode application codes will be added to the program as below:

a) Main function:

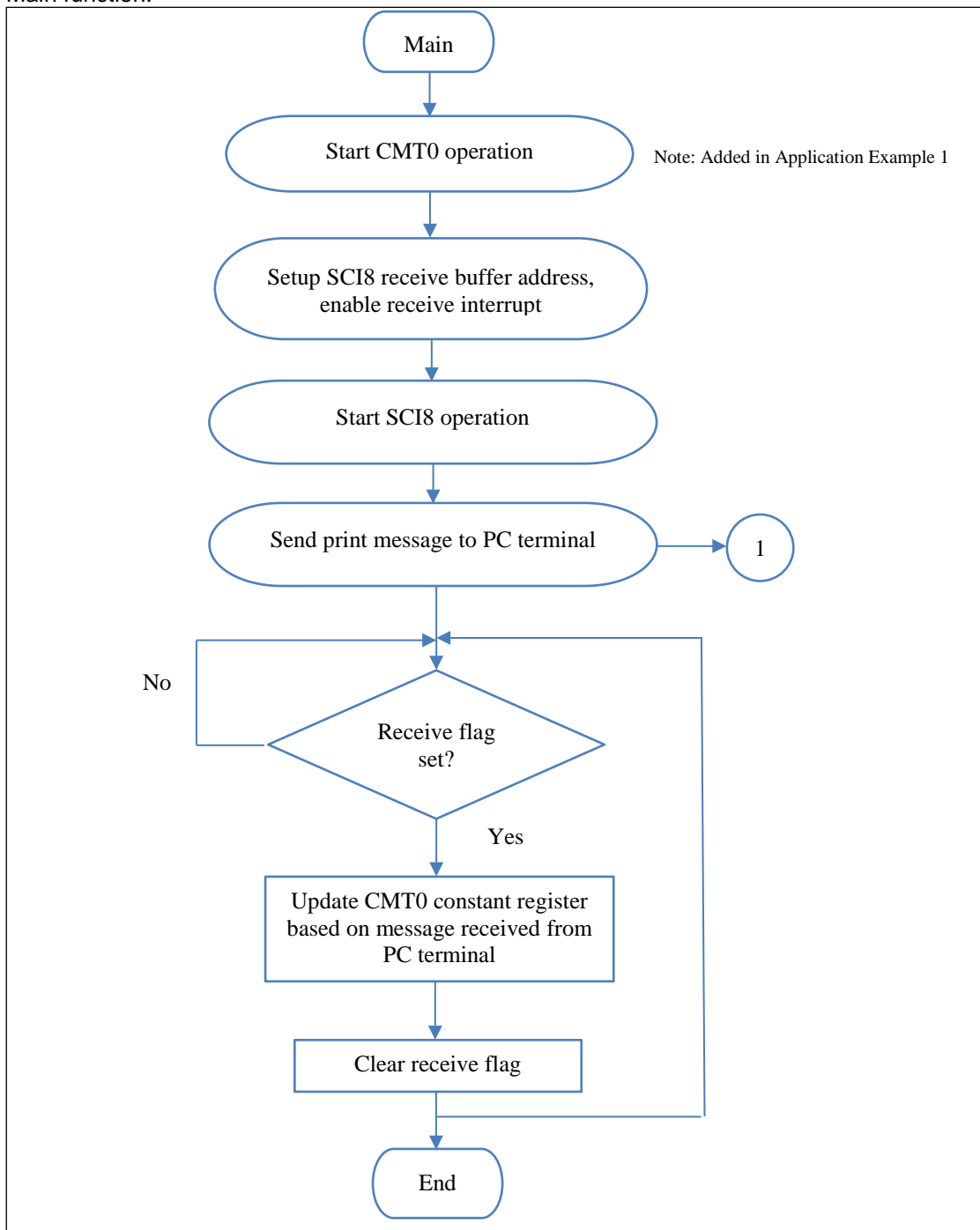


Figure 4-3 Main flowchart

b) SCI8 user functions:

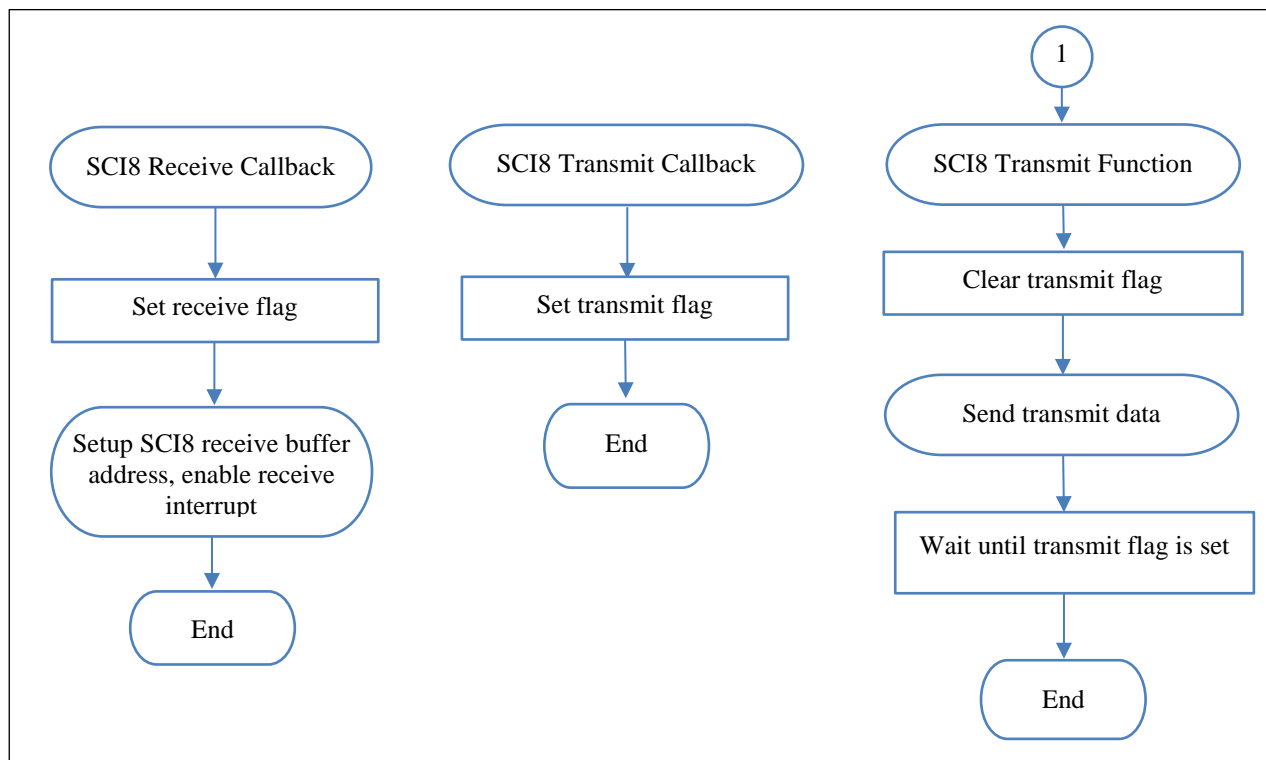



Figure 4-4 SCI user functions flowchart

4.1 Removing S12AD driver and related application codes

- 1) In Smart_Configurator_Example.scfg pane, click the [Components] tab
- 2) At Components tree, click to select *Config_S12AD0* configuration
- 3) Click  to remove this configuration

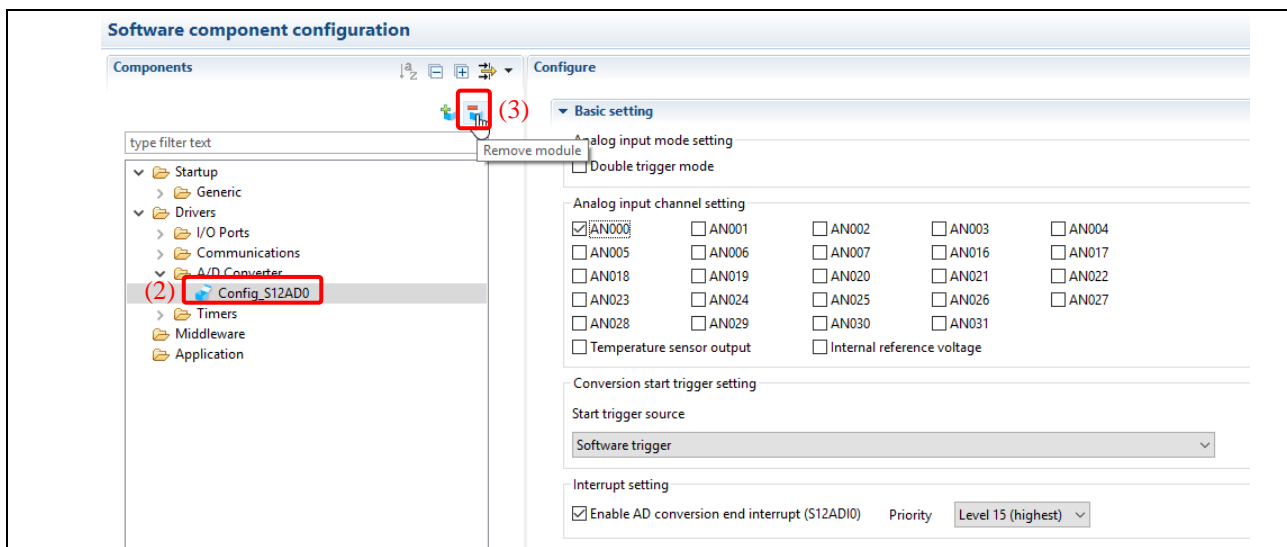



Figure 4-5 Software component configuration in Smart Configurator

- 4) Click  to generate codes
The source folder *Config_S12AD0* will be moved to project \trash folder before the code generation operation
- 5) At the project tree, open "Smart_Configurator_Example.c" in \src folder
- 6) Remove the application codes which are related to S12AD0
 - a. Remove declarations for global variables

```

/* Global variable for changing CMT0 interval */
uint16_t interval_level;

/* Global variable for storing the A/D conversion result */
uint16_t g_adc_result;

/* Global flag to indicate A/D conversion operation is completed */
extern uint8_t g_adc_flag;
    
```

- b. Remove all codes after *R_Config_CMT0_Start* function

After deleted the codes related to S12AD0, the main file will look like following:

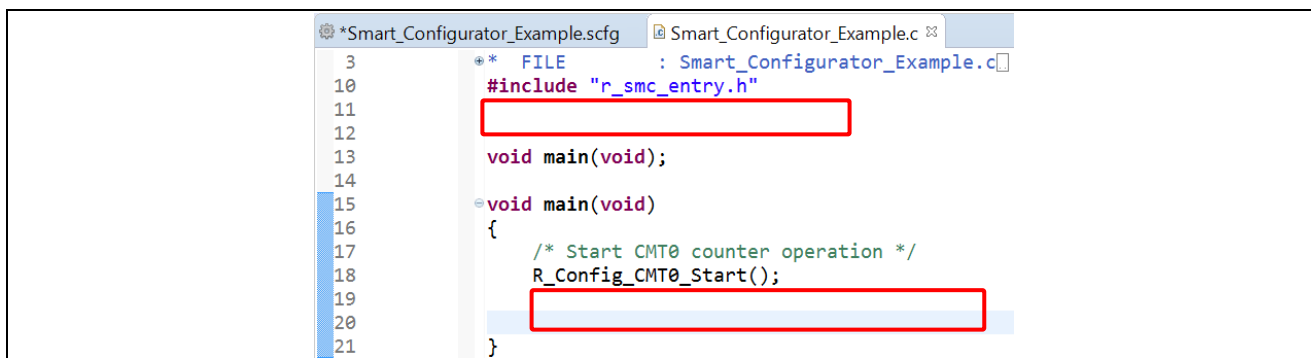



Figure 4-6 Main file after deleted codes related to S12AD0

4.2 Adding a peripheral driver

- 1) In the Smart_Configurator_Example.scfg pane, click [Components] tab
- 2) Click  to add new component

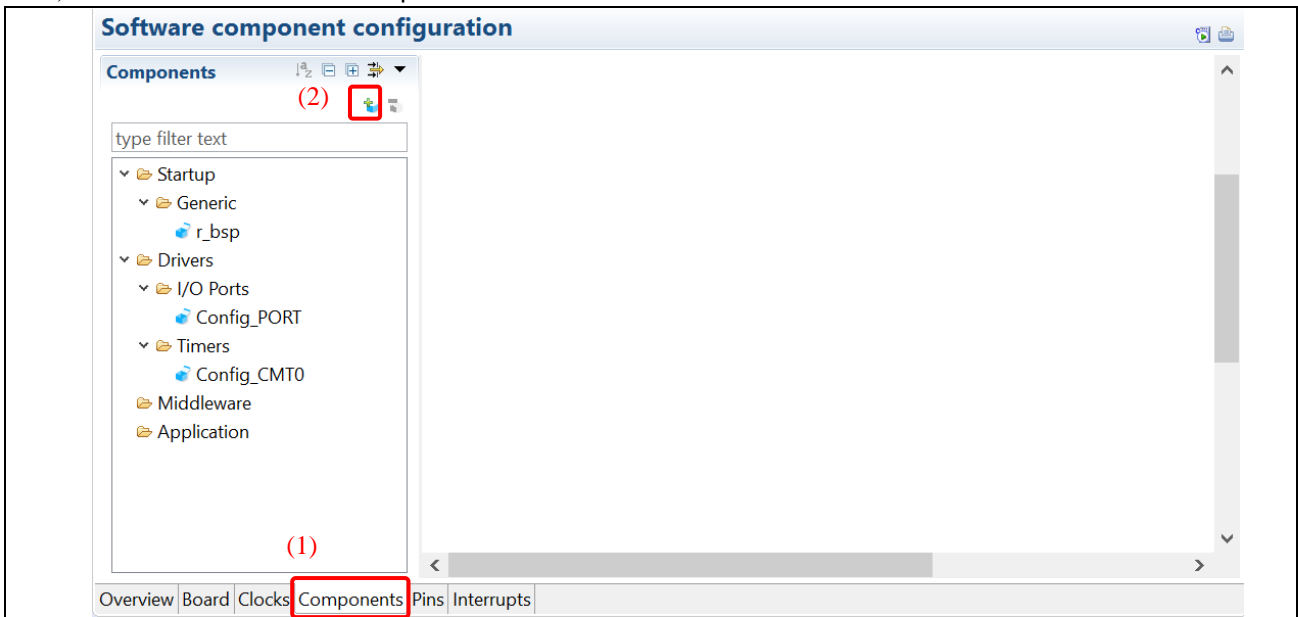


Figure 4-7 Software component configuration in Smart Configurator

- 3) Add *Serial Communication Interface* driver into the project
 - Select “Communications (Drivers)” from the “Function” option
 - Navigate the component list and select *SCI/SCIF Asynchronous Mode* driver
 - Click [Next] to continue

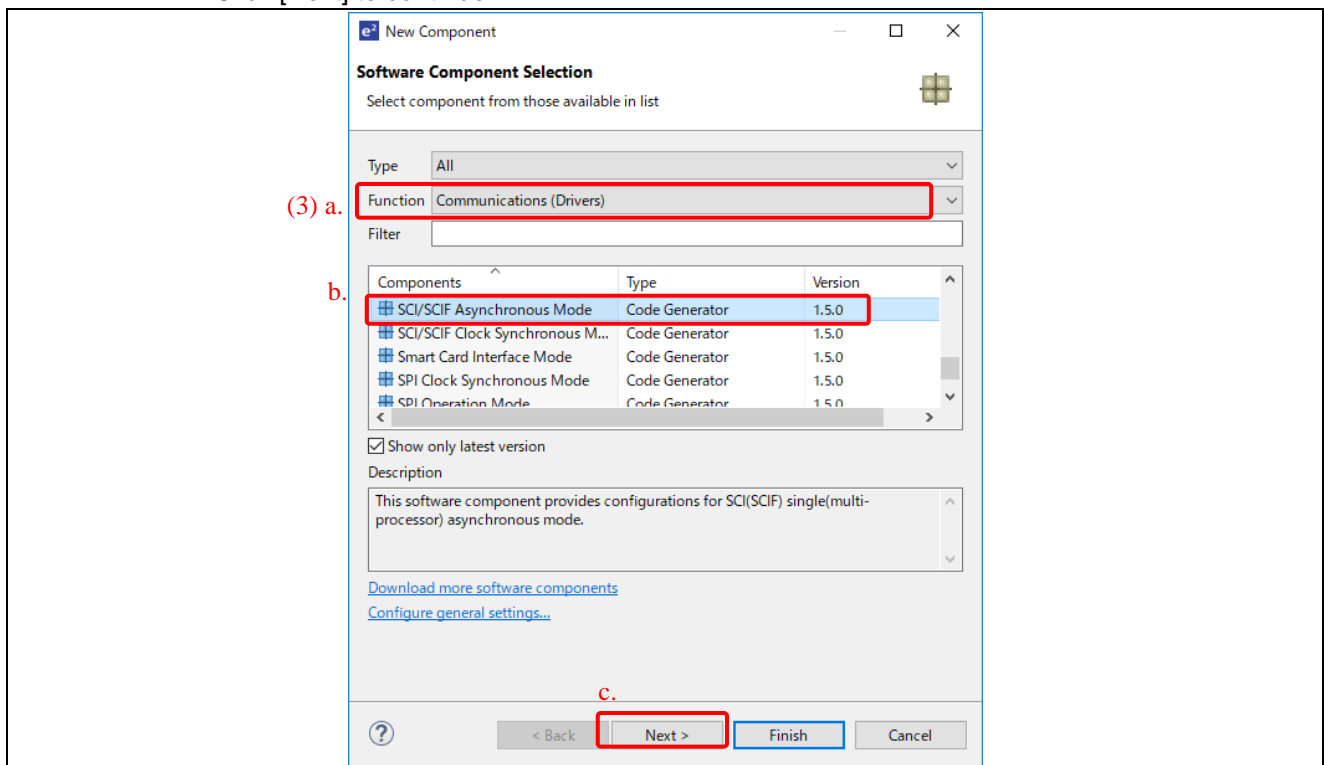


Figure 4-8 Select software component

- Select SCI8 as resource
- Select Work mode as “Transmission/Reception”
- Keep the default configuration name. Source files and API will be generated based on this configuration name
- Click [Finish]

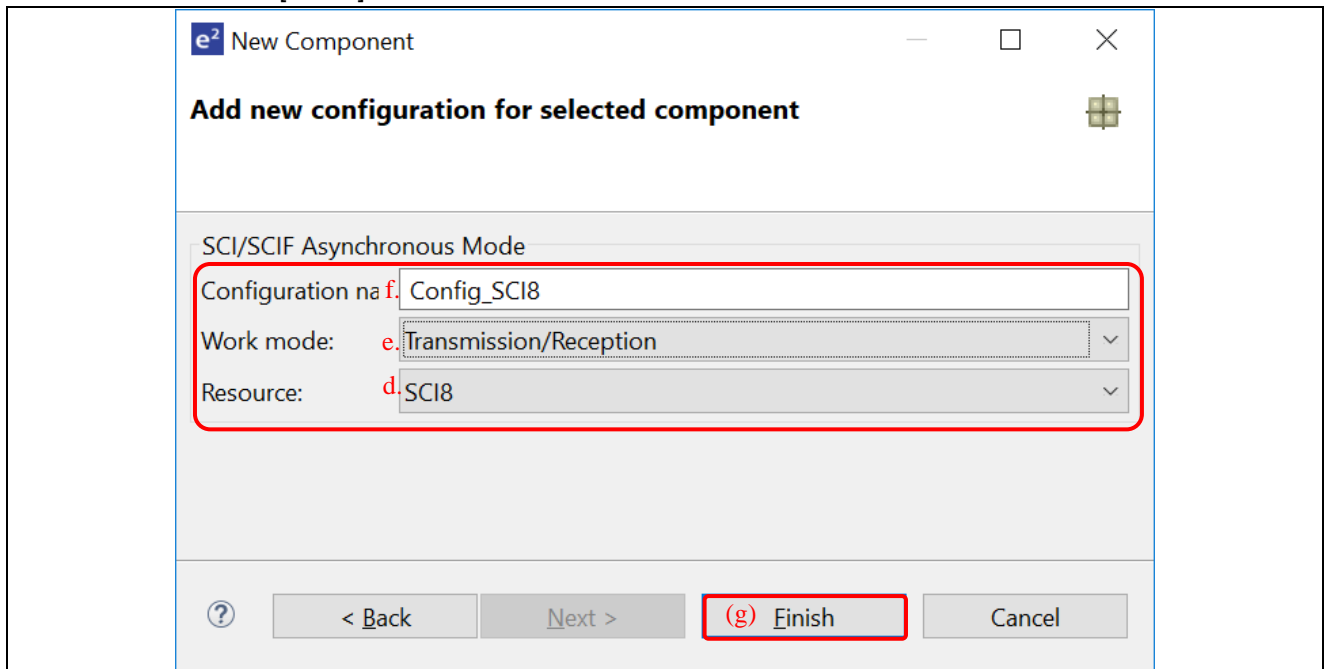


Figure 4-9 Add new configuration to the project

- 4) At Configure pane, ensure settings as the following:
 - Start bit edge detection setting: Falling edge on RXD8 pin
 - Data length setting: 8 bits
 - Parity setting: None
 - Stop bit length setting: 1 bit
 - Bit rate: 9600 bps

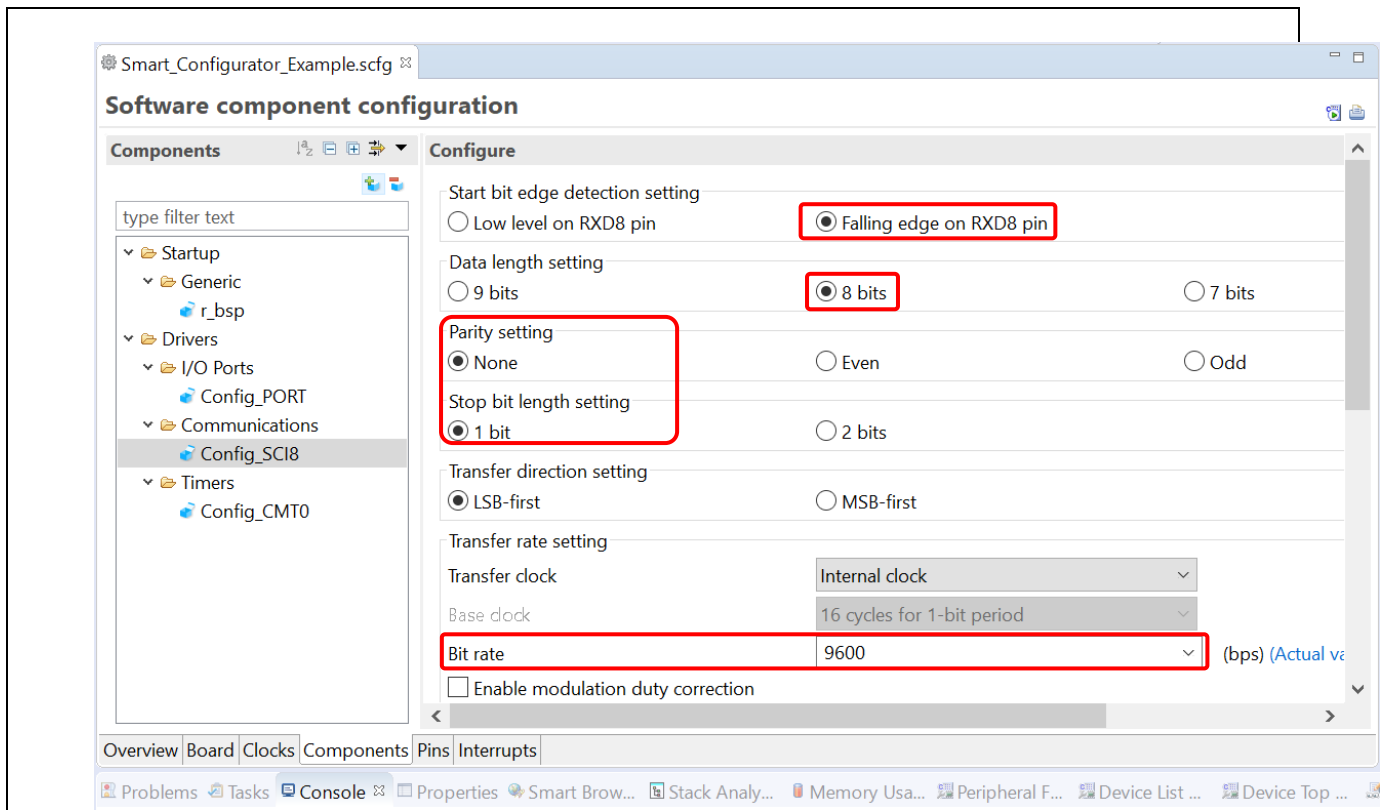



Figure 4-10 SCI/SCIF Asynchronous Mode driver configuration

- 5) At [Pins] tab, click  button to switch tree to Software Components view

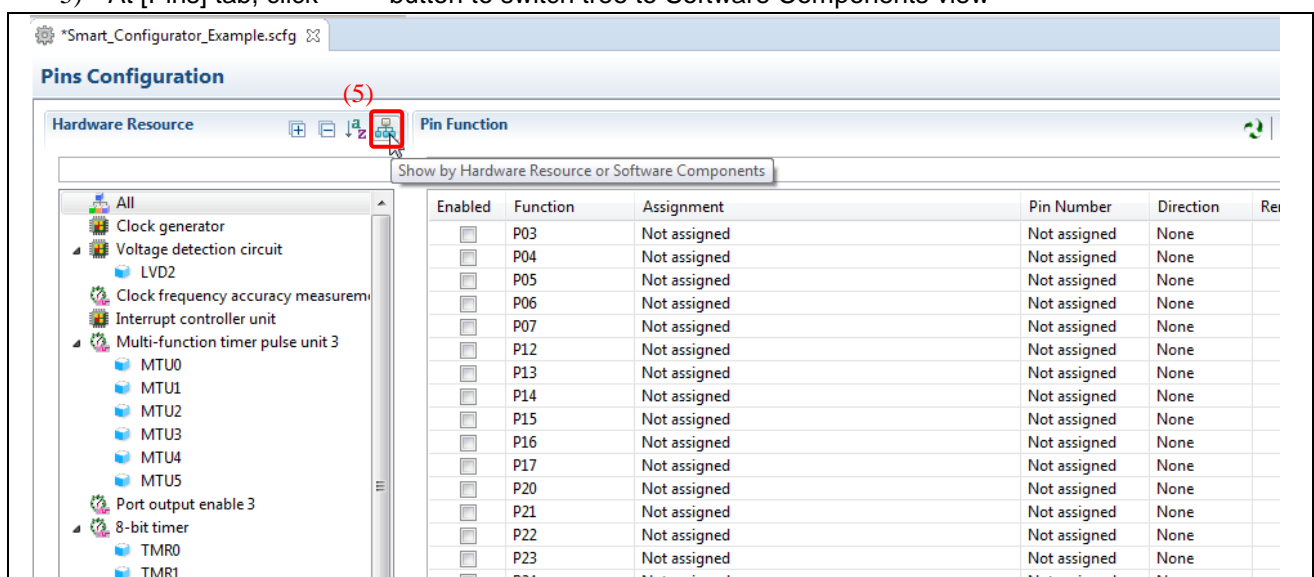


Figure 4-11 Pins tab

- 6) At the tree, click Config_SCI8 to view pin configurations
- 7) Ensure RXD8 is assigned to PJ1 and TXD8 is assigned to PJ2:

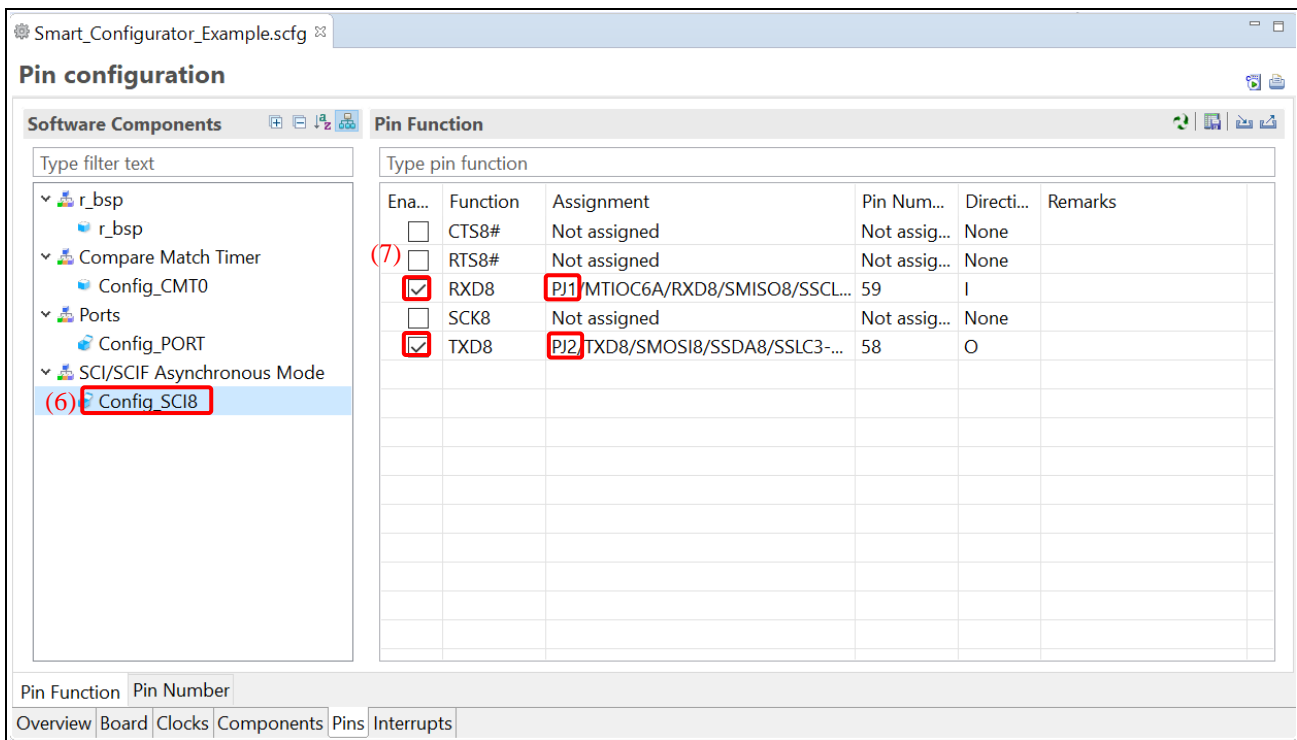



Figure 4-12 Pins configuration of Config_SCI8

4.3 Generating codes

- 1) At [Software Component Configuration] tab, click  to generate codes

4.4 Adding application codes in SCI/SCIF Asynchronous Mode driver

- 1) At Project Explore tree, open "Config_SCI8.h" in \src\smc_gen\Config_SCI8 folder. Add the declarations below at the user code area at the bottom of the file:

```
/* Sends SCI8 data and waits for transmit end flag */
MD_STATUS R_SCI8_AsyncTransmit(uint8_t * const tx_buf, const uint16_t
tx_num);
```

- 2) At Project Explore tree, open "Config_SCI8_user.c" in \src\smc_gen\Config_SCI8 folder
 - a. Add the declarations below at the user code area near the top of the file:

```
/* Global variable used to receive a character from PC terminal */
volatile uint8_t g_rx_char;

/* Flag used to detect whether data is received */
volatile uint8_t g_rx_flag;

/* Flag used to detect completion of transmission */
static volatile uint8_t SCI8_txdone;
```

- b. Set the transmission completion flag in *r_Config_SCI8_callback_transmitend* (void) function

```
/* Set transmit completion flag */
SCI8_txdone = 1U;
```

- c. Set the reception completion flag in *r_Config_SCI8_callback_receiveend* (void) function and restart reception

```
/* Set receive completion flag */
g_rx_flag = 1U;

/* Set SCI8 receive buffer address and restart reception */
R_Config_SCI8_Serial_Receive((uint8_t *)&g_rx_char, 1);
```

- d. Add a function to send SCI8 data and waits for the transmit end flag at user code area at the bottom of the file

```

/*****
*****
* Function Name: R_SCI8_AsyncTransmit
* Description  : This function sends SCI8 data and waits for the
transmit end flag
* Arguments    : tx_buf -
*                transfer buffer pointer
*                tx_num -
*                buffer size
* Return Value : status -
*                MD_OK or MD_ARGERROR
*****
*****/
MD_STATUS R_SCI8_AsyncTransmit (uint8_t * const tx_buf, const uint16_t
tx_num)
{
    MD_STATUS status = MD_OK;

    /* Clear transmit completion flag before transmission */
    SCI8_txdone = 0U;

    /* Set SCI8 transmit buffer address and start transmission */
    status = R_Config_SCI8_Serial_Send(tx_buf, tx_num);

    /* Wait for transmit end flag */
    while (0U == SCI8_txdone)
    {
        /* Wait */
    }
    return (status);
}
/*****
*****
* End of function R_SCI8_AsyncTransmit
*****
*****/

```

Your source file should look like this:

```

File Name : Config_SCI8_user.c

#pragma directive
/* Start user code for pragma. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */

#include "r_cg_macrodriver.h"
#include "Config_SCI8.h"
/* Start user code for include. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
#include "r_cg_userdefine.h"

Global variables and functions
extern volatile uint8_t * gp_sci8_tx_address; /* SCI8 transmit buffer address */
extern volatile uint16_t g_sci8_tx_count; /* SCI8 transmit data number */
extern volatile uint8_t * gp_sci8_rx_address; /* SCI8 receive buffer address */
extern volatile uint16_t g_sci8_rx_count; /* SCI8 receive data number */
extern volatile uint16_t g_sci8_rx_length; /* SCI8 receive data length */
/* Start user code for global. Do not edit comment generated here */
/* Global variable used to receive a character from PC terminal */
volatile uint8_t g_rx_char;

/* Flag used to detect whether data is received */
volatile uint8_t g_rx_flag;

/* Flag used to detect completion of transmission */
static volatile uint8_t SCI8_txdone;
/* End user code. Do not edit comment generated here */

*****
Function Name: r_Config_SCI8_callback_transmitend
Description : This function is a callback function when SCI8 finishes transmission
Arguments : None
Return Value : None
*****
static void r_Config_SCI8_callback_transmitend(void)
{
/* Start user code for r_Config_SCI8_callback_transmitend. Do not edit comment generated here */
/* Set transmit completion flag */
SCI8_txdone = 1U;
/* End user code. Do not edit comment generated here */
}

*****
Function Name: r_Config_SCI8_callback_receiveend
Description : This function is a callback function when SCI8 finishes reception
Arguments : None
Return Value : None
*****
static void r_Config_SCI8_callback_receiveend(void)
{
/* Start user code for r_Config_SCI8_callback_receiveend. Do not edit comment generated here */
/* Set receive completion flag */
g_rx_flag = 1U;

/* Set SCI8 receive buffer address and restart reception */
R_Config_SCI8_Serial_Receive((uint8_t *)&g_rx_char, 1);
/* End user code. Do not edit comment generated here */
}

```

```

/* Start user code for adding. Do not edit comment generated here */
= /*****
* Function Name: R_SCI8_AsyncTransmit
* Description : This function sends SCI8 data and waits for the transmit end flag
* Arguments : tx_buf -
*             transfer buffer pointer
*             tx_num -
*             buffer size
* Return Value : status -
*               MD_OK or MD_ARGERROR
*****/
= MD_STATUS R_SCI8_AsyncTransmit (uint8_t * const tx_buf, const uint16_t tx_num)
{
    MD_STATUS status = MD_OK;

    /* Clear transmit completion flag before transmission */
    SCI8_txdone = 0U;

    /* Set SCI8 transmit buffer address and start transmission */
    status = R_Config_SCI8_Serial_Send(tx_buf, tx_num);

    /* Wait for transmit end flag */
    = while (0U == SCI8_txdone)
    {
        /* Wait */
    }
    return (status);
}
= /*****
* End of function R_SCI8_AsyncTransmit
*****/

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

```

Figure 4-13 Config_SCI8_user.c

4.5 Adding application codes in *main()*

- 1) In Smart_Configurator_Example.c, add header files and declarations near the top of the file

```

#include <stdio.h>
#include <string.h>

/* Global variable for changing CMT0 interval */
volatile uint16_t interval_level = 5;

/* String used to print message at PC terminal */
static char print_str[80];

/* Flag used to detect whether data is received from PC terminal */
extern volatile uint8_t g_rx_flag;

/* Global variable used for storing data received from PC terminal */
extern volatile uint8_t g_rx_char;

```

2) Add below codes into *main()* function after *R_Config_CMT0_Start()* to start communication with terminal program:

```

/* Set SCI8 receive buffer address and enable receive interrupt */
R_Config_SCI8_Serial_Receive((uint8_t *)&g_rx_char, 1);

/* Enable SCI8 operation */
R_Config_SCI8_Start();

/* Print instruction onto PC terminal */
sprintf(print_str, "\r\n Press keyboard to control the blinking rate of LED2 (not case sensitive)\r\n");
R_SCI8_AsyncTransmit((uint8_t *)print_str, (uint16_t)strlen(print_str));
sprintf(print_str, "\r\n a ----> Slower\r\n");
R_SCI8_AsyncTransmit((uint8_t *)print_str, (uint16_t)strlen(print_str));
sprintf(print_str, "\r\n b ----> Faster\r\n");
R_SCI8_AsyncTransmit((uint8_t *)print_str, (uint16_t)strlen(print_str));

while (1U)
{
    /* Get new blink interval level from PC terminal */
    if (g_rx_flag == 1U)
    {
        /* Instruction to slow down blinking rate is received */
        if (('A' == g_rx_char) || ('a' == g_rx_char))
        {
            R_Config_SCI8_Serial_Receive((uint8_t *)&g_rx_char, 1);

            /* Notify the character received and inform next action */
            sprintf(print_str, "\r\n Character 'a' or 'A' is received. LED2 will blink at
            slower rate.\r\n");
            R_SCI8_AsyncTransmit((uint8_t *)print_str, (uint16_t)strlen(print_str));

            /* Get new blink interval level. Maximum level is 9. */
            if (interval_level < 10)
            {
                interval_level++;
            }
            else
            {
                sprintf(print_str, "\r\n This is minimum. Please press 'b' or 'B' to
                blink faster. \r\n");
                R_SCI8_AsyncTransmit((uint8_t *)print_str,
                (uint16_t)strlen(print_str));
            }
        }
        /* Instruction to increase blinking rate is received */
        else if (('B' == g_rx_char) || ('b' == g_rx_char))
        {
            R_Config_SCI8_Serial_Receive((uint8_t *)&g_rx_char, 1);

            /* Notify the character received and inform next action */
            sprintf(print_str, "\r\n Character 'b' or 'B' is received. LED2 will blink at
            faster rate.\r\n");
            R_SCI8_AsyncTransmit((uint8_t *)print_str, (uint16_t)strlen(print_str));

            /* Get new blink interval level. Minimum level is 1 */
            if (interval_level > 1)
            {
                interval_level--;
            }
            else
            {
                sprintf(print_str, "\r\n This is maximum. Please press 'a' or 'A' to
                blink slower. \r\n");
                R_SCI8_AsyncTransmit((uint8_t *)print_str,
                (uint16_t)strlen(print_str));
            }
        }
        else
        {
            /* Do nothing */
        }

        /* Change blinking rate */
        CMT0.CMCOR = (uint16_t)(5000 * interval_level);

        /* Reset SCI8 reception flag*/
        g_rx_flag = 0U;
    }
    else
    {
        /* Do nothing */
    }
}

```


Your source file should look like this:

```

⊕ * FILE      : Smart_Configurator_Example.c
#include "r smc entry.h"
#include <stdio.h>
#include <string.h>

/* Global variable for changing CMT0 interval */
volatile uint16_t interval_level = 5;

/* String used to print message at PC terminal */
static char print_str[80];

/* Flag used to detect whether data is received from PC terminal */
extern volatile uint8_t g_rx_flag;

/* Global variable used for storing data received from PC terminal */
extern volatile uint8_t g_rx_char;

void main(void);
void main(void)
{
    /* Start CMT0 counter operation */
    R_Config_CMT0_Start();

    /* Set SCI8 receive buffer address and enable receive interrupt */
    R_Config_SCI8_Serial_Receive((uint8_t *) &g_rx_char, 1);

    /* Enable SCI8 operation */
    R_Config_SCI8_Start();

    /* Print instruction onto PC terminal */
    sprintf(print_str, "\r\n Press keyboard to control the blinking rate of LED2 (not case sensitive)\r\n");
    R_SCI8_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));
    sprintf(print_str, "\r\n a ----> Slower\r\n");
    R_SCI8_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));
    sprintf(print_str, "\r\n b ----> Faster\r\n");
    R_SCI8_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));

    while (1U)
    {
        /* Get new blink interval level from PC terminal */
        if (g_rx_flag == 1U)
        {
            /* Instruction to slow down blinking rate is received */
            if (('A' == g_rx_char) || ('a' == g_rx_char))
            {
                R_Config_SCI8_Serial_Receive((uint8_t *) &g_rx_char, 1);

                /* Notify the character received and inform next action */
                sprintf(print_str, "\r\n Character 'a' or 'A' is received. LED2 will blink at slower rate.\r\n");
                R_SCI8_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));

                /* Get new blink interval level. Maximum level is 9. */
                if (interval_level < 10)
                {
                    interval_level++;
                }
            }
            else
            {
                sprintf(print_str, "\r\n This is minimum. Please press 'b' or 'B' to blink faster. \r\n");
                R_SCI8_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));
            }
        }
    }
}

```

```

/* Instruction to increase blinking rate is received */
else if (('B' == g_rx_char) || ('b' == g_rx_char)) {
    R_Config_SCI8_Serial_Receive((uint8_t *) &g_rx_char, 1);

    /* Notify the character received and inform next action */
    sprintf(print_str,
            "\r\n Character 'b' or 'B' is received. LED2 will blink at faster rate.\r\n");
    R_SCI8_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));

    /* Get new blink interval level. Minimum level is 1 */
    if (interval_level > 1)
    {
        interval_level--;
    }
    else
    {
        sprintf(print_str, "\r\n This is maximum. Please press 'a' or 'A' to blink slower. \r\n");
        R_SCI8_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));
    }
} else
{
    /* Do nothing */
}

/* Change blinking rate */
CMT0.CMCOR = (uint16_t) (5000 * interval_level);

/* Reset SCI8 reception flag*/
g_rx_flag = 0U;
}
else
{
    /* Do nothing */
}
}
}

```

Figure 4-14 Asynchronous application codes in *main* function

4.6 Build and run on hardware board

After build the project, perform the following steps before debug the project.

Step1. Use a USB to mini B cable to connect PC COM port and USB Serial connector (G1CUSB0) on Renesas Starter Kit+ for RX65N 2MB (refer to Chapter 4.7 for the serial port location)

Step 2. Open a terminal program in PC (for e.g. Tera Term). Perform the following steps:

- Select USB Serial Device

Note: Port name may vary depends on driver in PC

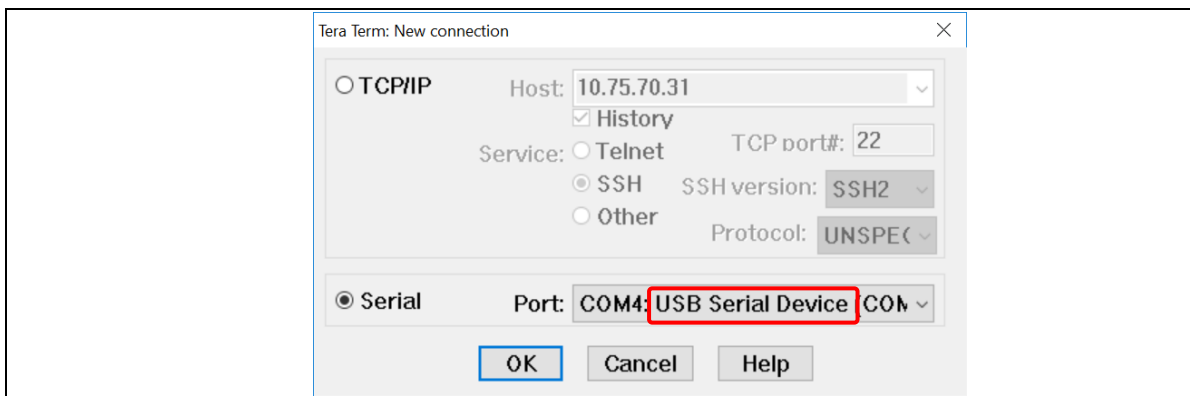


Figure 4-15 USB Serial Port

- Click [Setup] from the menu and select [Serial port...]

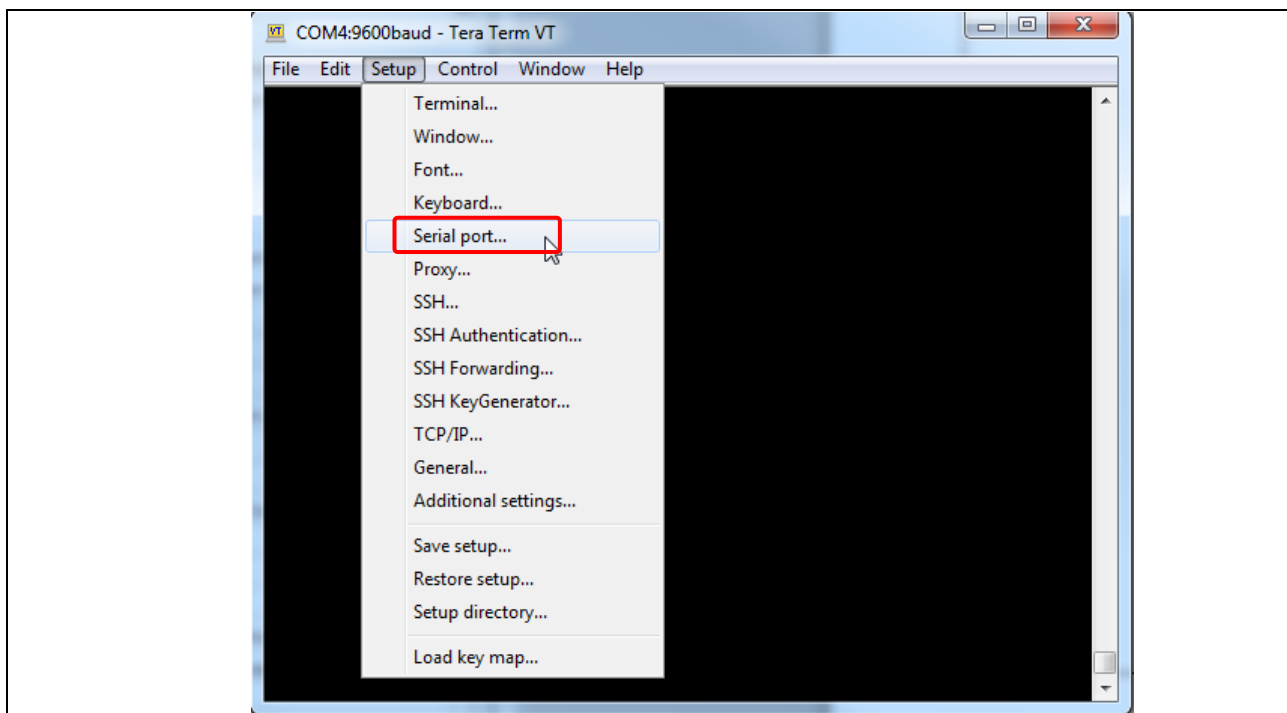


Figure 4-16 Terminal window in PC

- In [Serial port setup] popup window, ensure the settings below match the setting in *SCI/SCIF Asynchronous Mode* driver and click [OK] to proceed
 - Data length setting: 8 bits
 - Parity setting: None
 - Stop bit length setting: 1 bit
 - Bit rate: 9600 bps

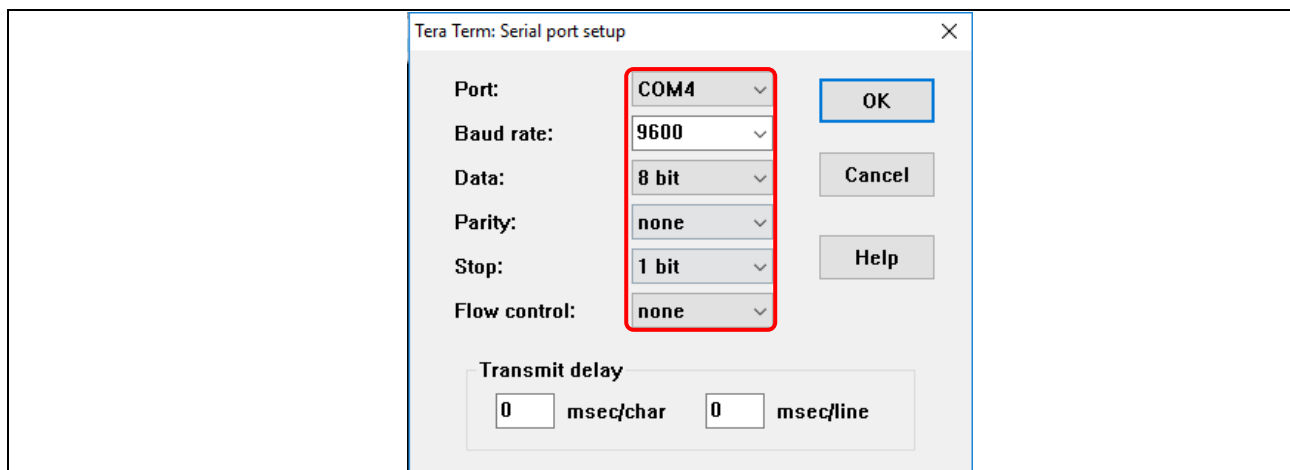




Figure 4-17 Serial port setup

Step 3. At e² studio, click debug  icon

Step 4. At debug perspective, click resume icon  icon to execute the project

The PC terminal window will show the following message. Follow the message to increase or decrease LED2 blinking speed.

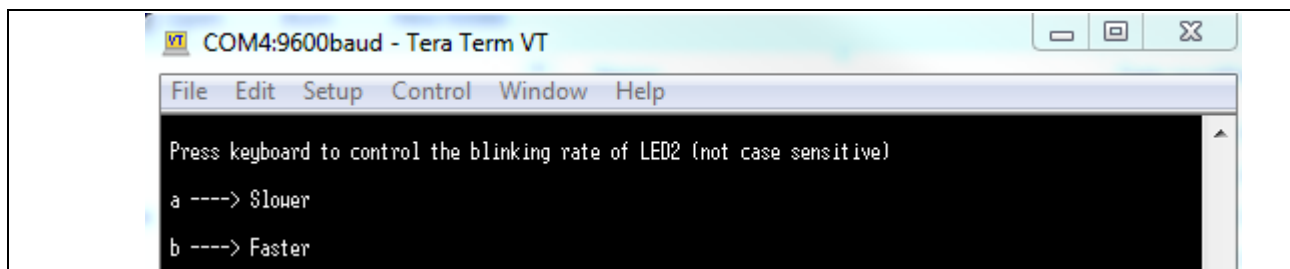


Figure 4-18 Terminal window in PC

These messages will be shown if RSK board receives an input from PC.

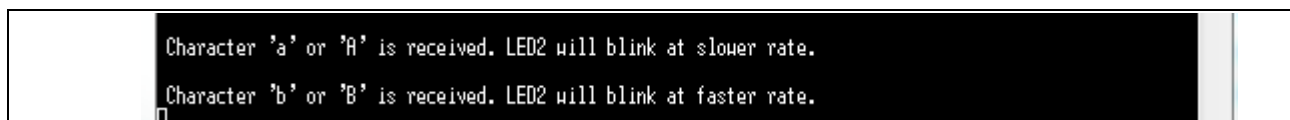


Figure 4-19 Response from RSK board when an input is received from PC

4.7 Operation on board

On the Renesas Starter Kit+ for RX65N 2MB board:

Blinking rate of LED2 is changed when key in character 'a' or 'b' at the PC terminal:

- a : Blinking of LED2 will slow down
- b : Blinking of LED2 will become faster

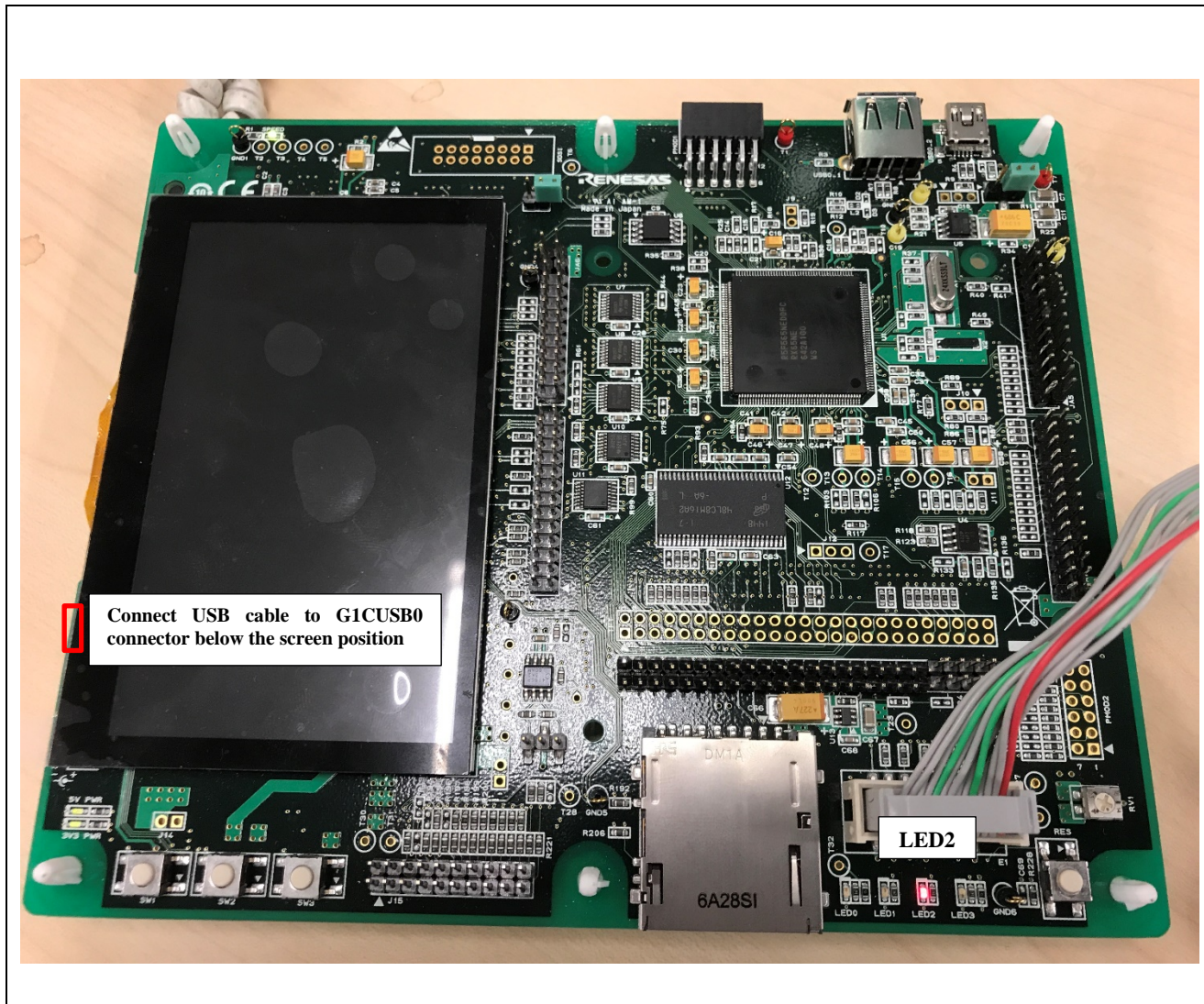


Figure 4-20 LED2 and Serial Port on RX65N 2MB board

5. Application Example 4 (Transferring data using DMA)

In the previous chapter, we use interrupt service routine to transfer the data between SCI8 and on-chip RAM. The data transfer is handled by CPU.

In this chapter, we introduce DMA controller to transfer the data. The memory operation is speeded up and the CPU is freed from data transfer.

Note:

Direct memory access (DMA) is a method that allows an input/output (I/O) device to send or receive data directly to or from the main memory, bypassing the CPU to speed up memory operations.

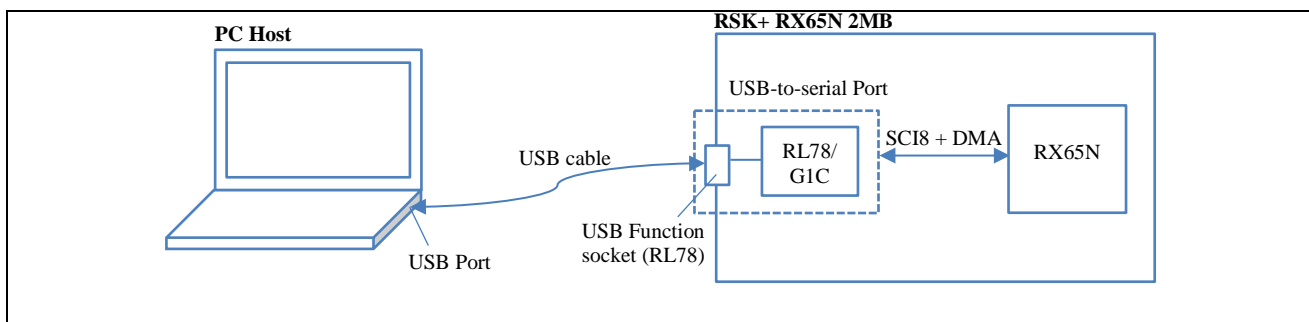


Figure 5-1 Data handled by DMAC

The table below shows changes of the driver selection:

	Drivers	Resource	Function
1. Add new driver	DMA Controller	DMAC0	Transfer data from SCI8 to RAM (Received)
		DMAC1	Transfer data from RAM to SCI8 (Transmit)
2. Modify driver	SCI/SCIF Asynchronous Mode	SCI8	Communication between PC terminal and target board for controlling LED blinking rate

The SCI-DMA application codes will be added to the program as below:

a) Main function:

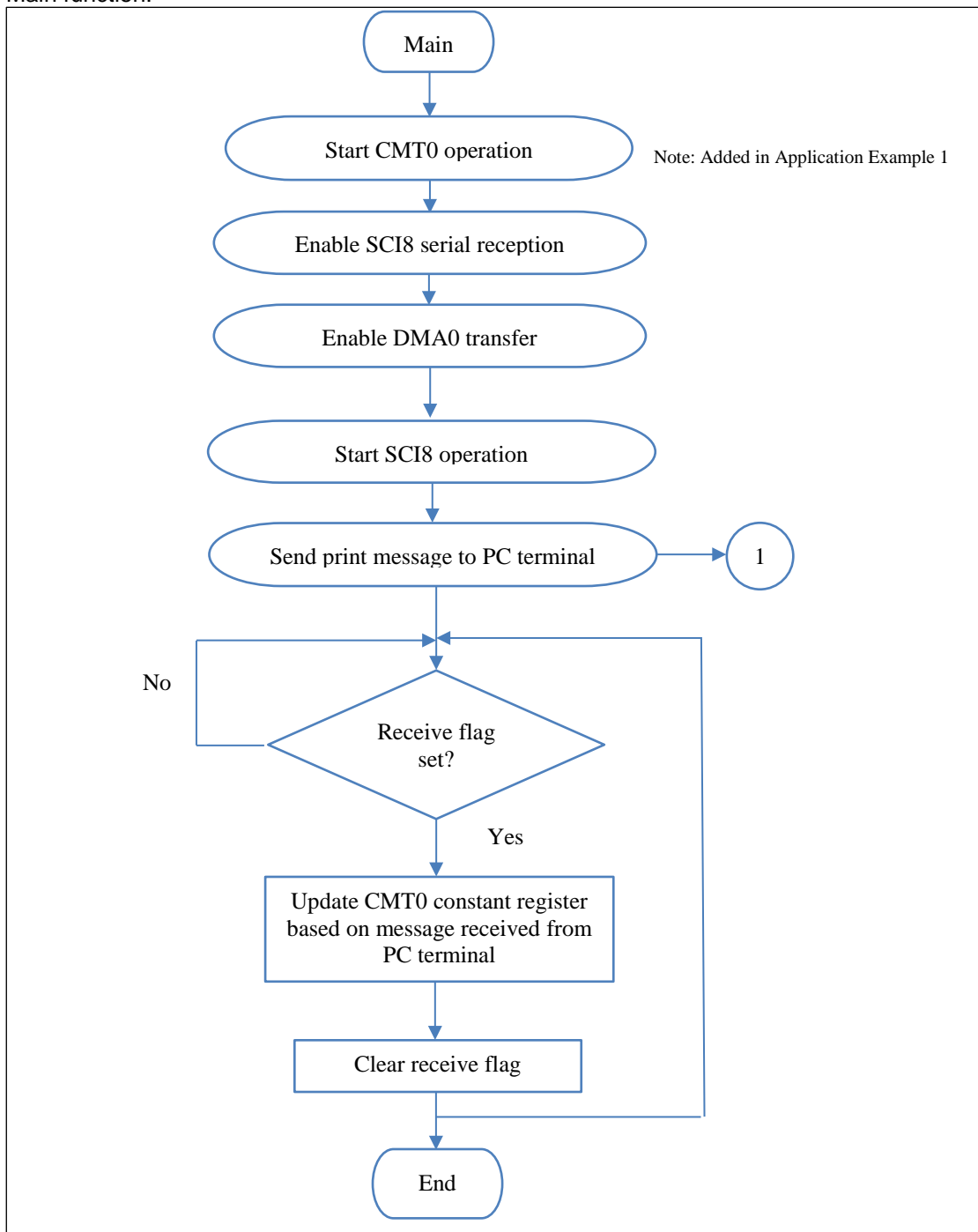


Figure 5-2 Main flowchart

b) SCI8 user functions:

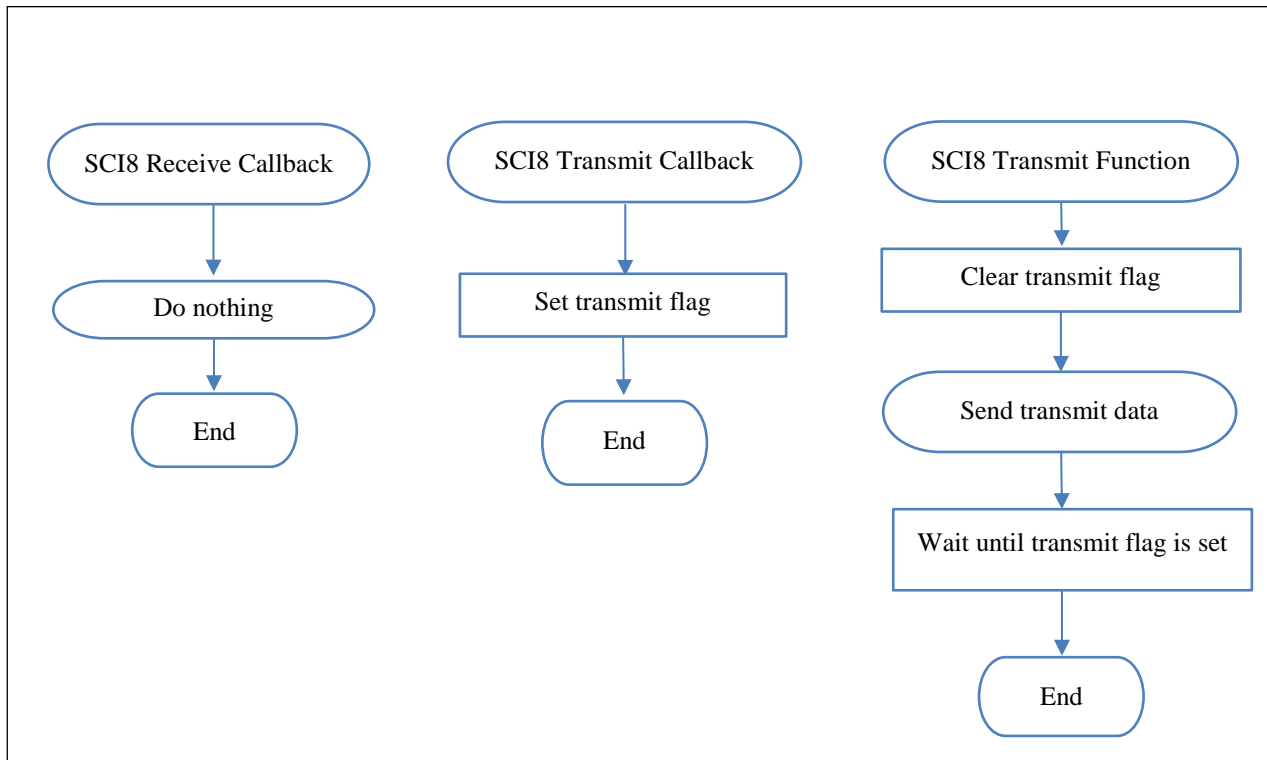


Figure 5-3 SCI user functions flowchart

c) DMAC0 user functions:

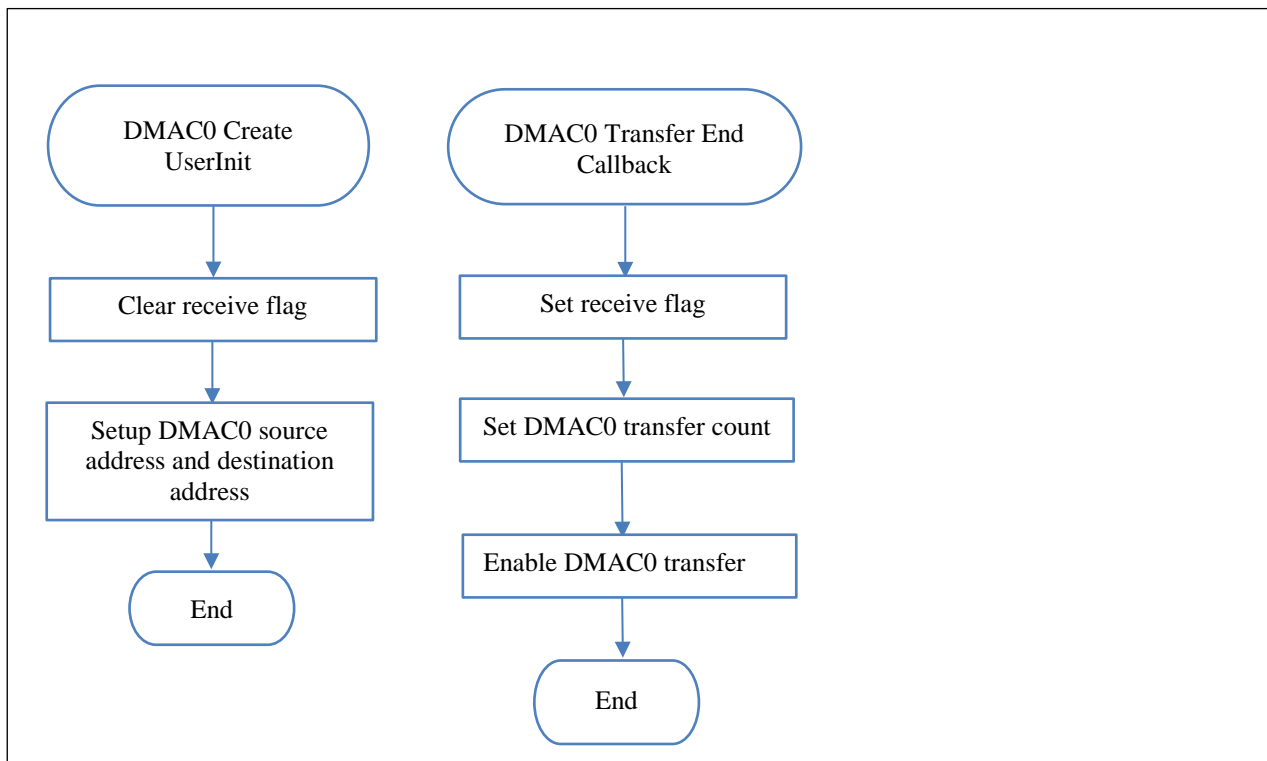


Figure 5-4 DMAC0 user functions flowchart

d) DMAC1 user functions:

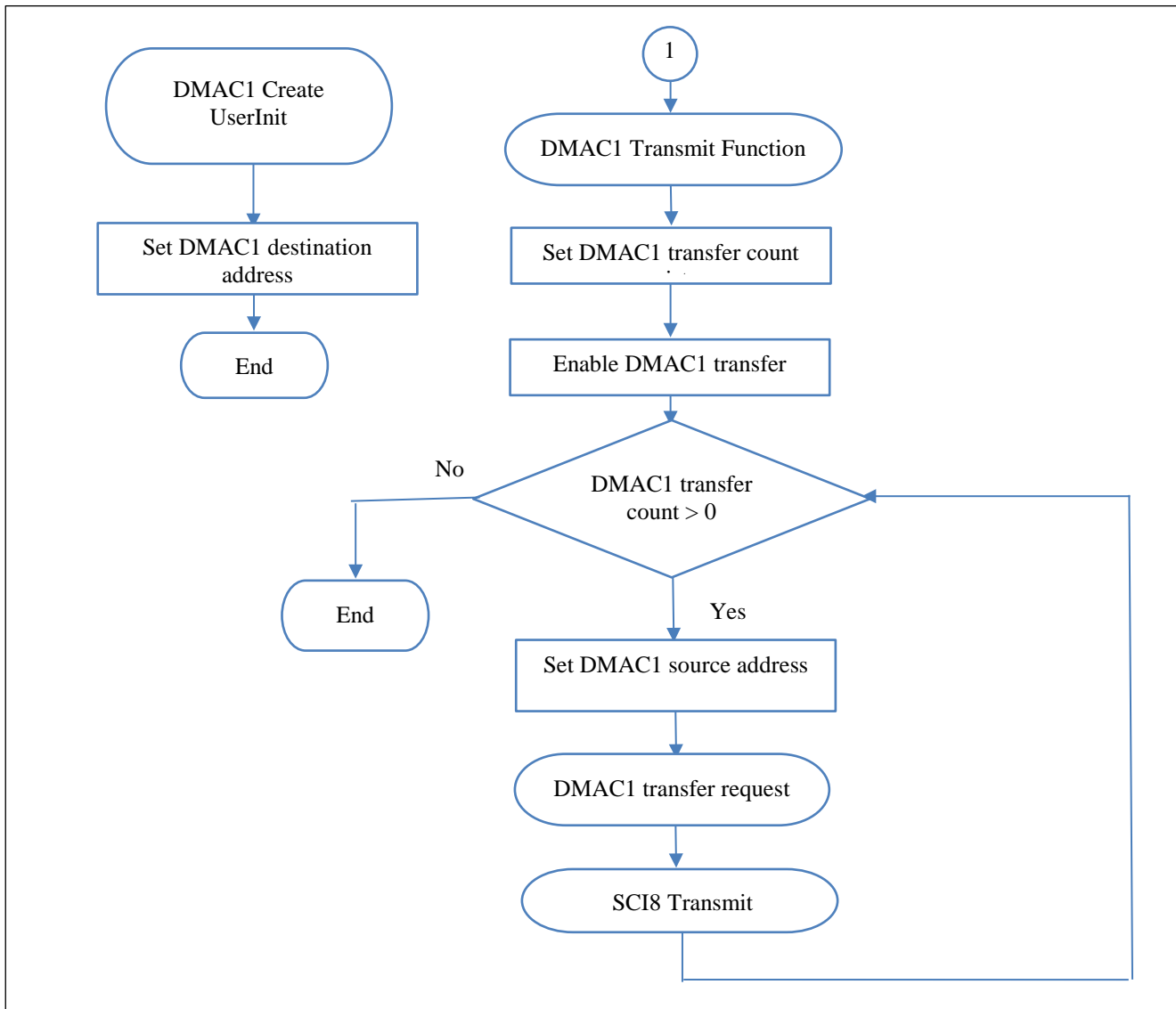



Figure 5-5 DMAC1 user functions flowchart

5.1 Adding and modifying peripheral drivers

5.1.1 Add DMAC0 driver

- 1) In the Smart_Configurator_Example.scfg pane, click [Components] tab
- 2) Click  to add new component

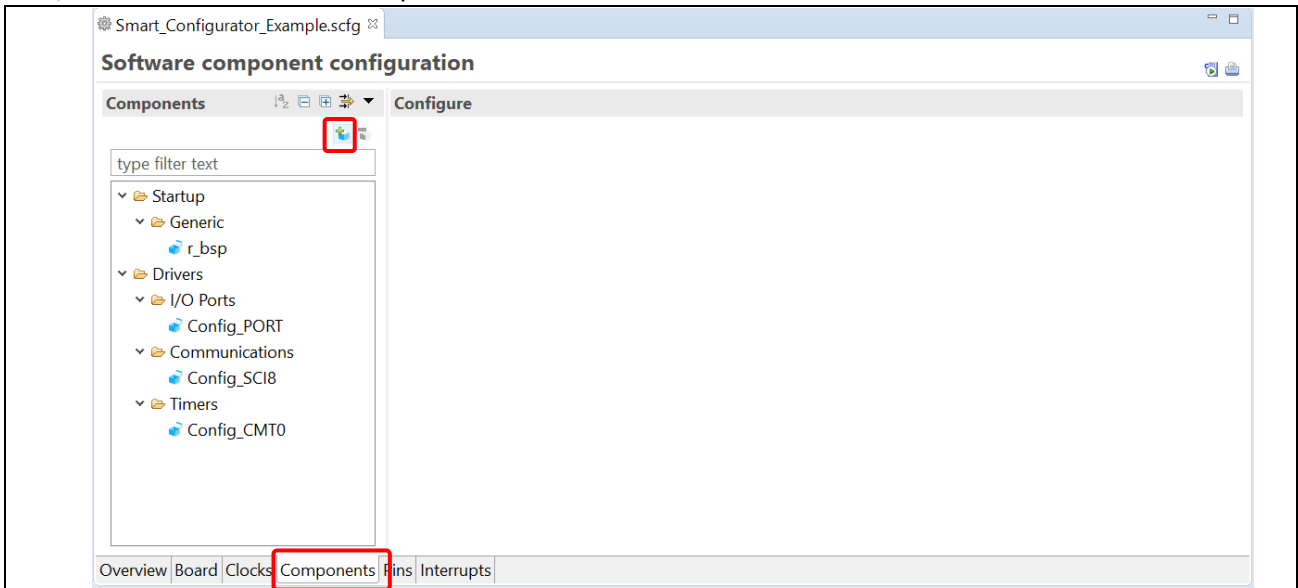


Figure 5-6 Software component configuration in Smart Configurator

- 3) Add *DMA Controller* driver into the project
 - Type “DMA” to the “Filter” box.
 - Navigate the component list and select *DMA Controller* driver
 - Click [Next] to continue

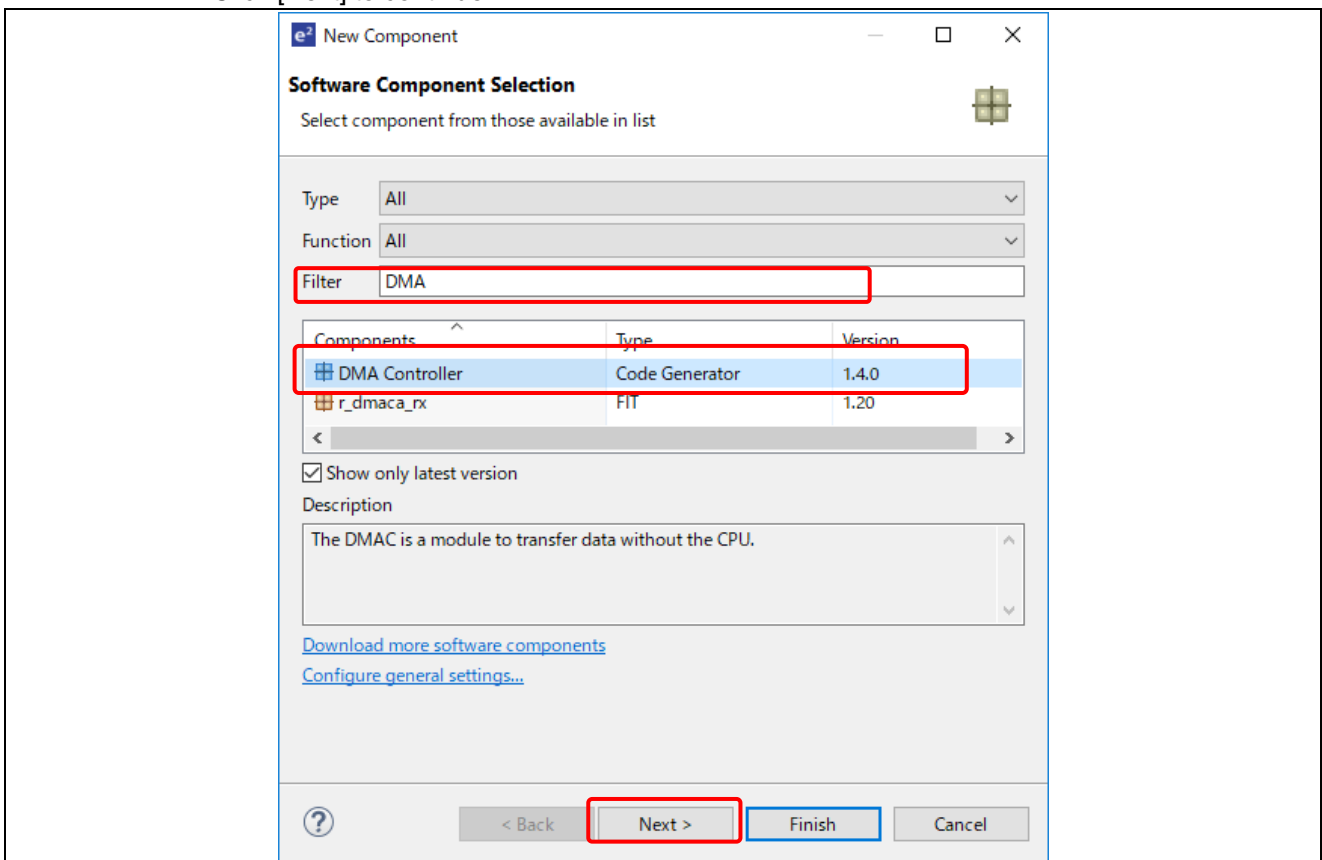


Figure 5-7 Select software component

- 4) In the [New Component] dialog box
 - Select *DMAC0* as resource
 - Keep the default configuration name. Source files and API will be generated based on this configuration name
 - Click [Finish]

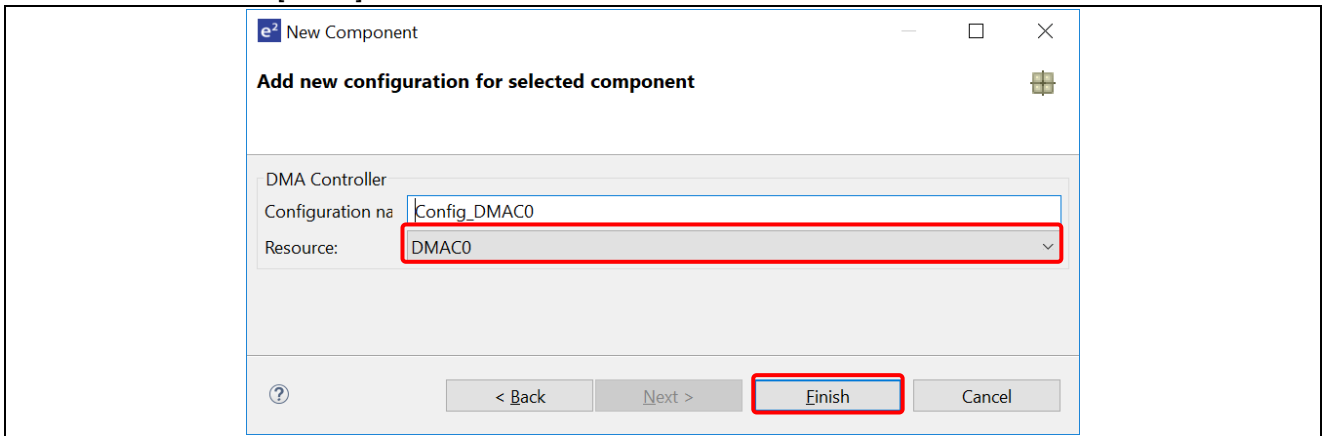


Figure 5-8 Add new configuration to the project

- 5) At Configure pane, ensure settings as the following:
 - Activation source: SCI8(RX18)
 - Enable interrupt on transfer end: Checked
 - Keep other default settings

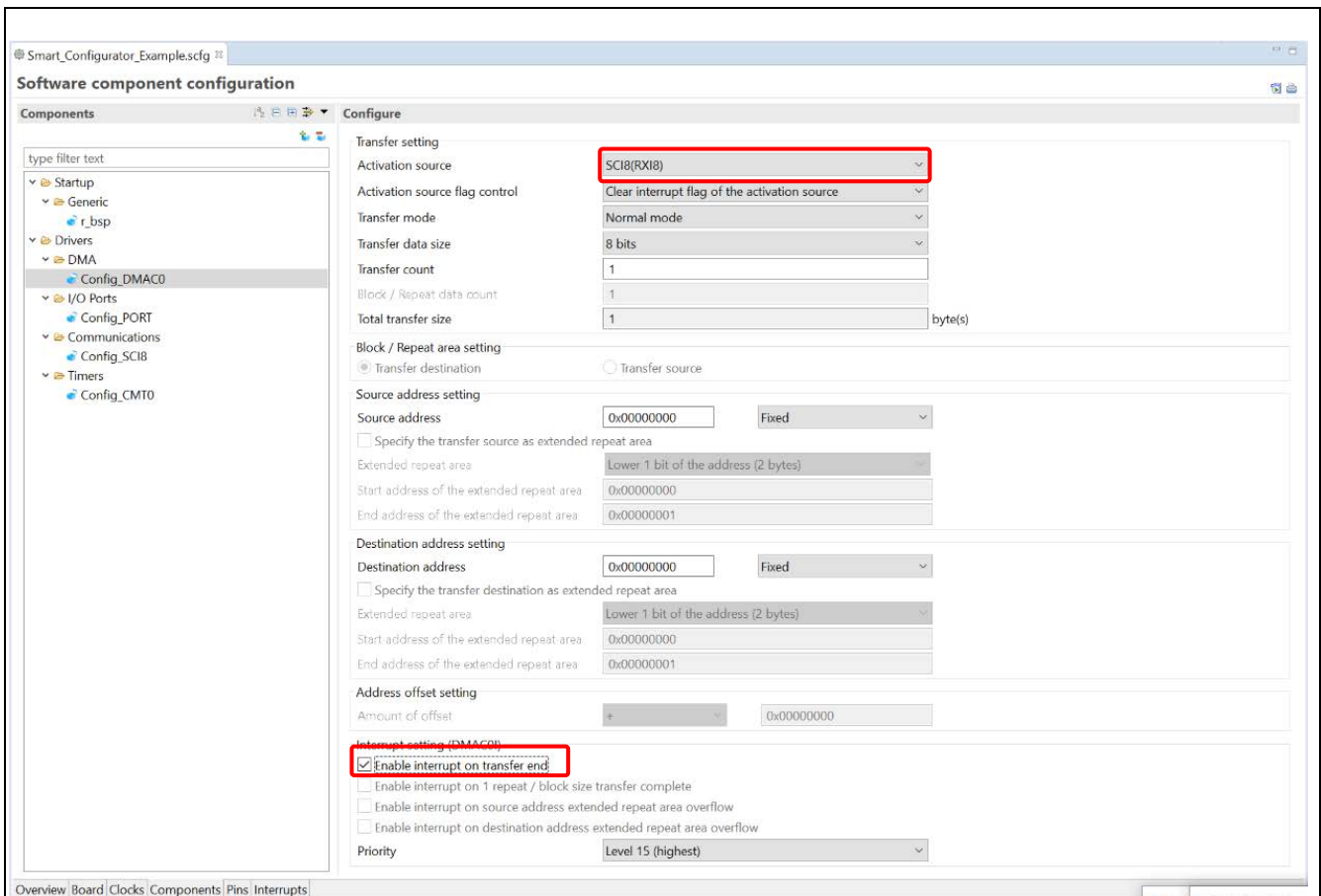



Figure 5-9 DMAC0 driver configuration

5.1.2 Add DMAC1 driver

- 1) In the Smart_Configurator_Example.scfg pane, click [Components] tab
- 2) Click  to add new component

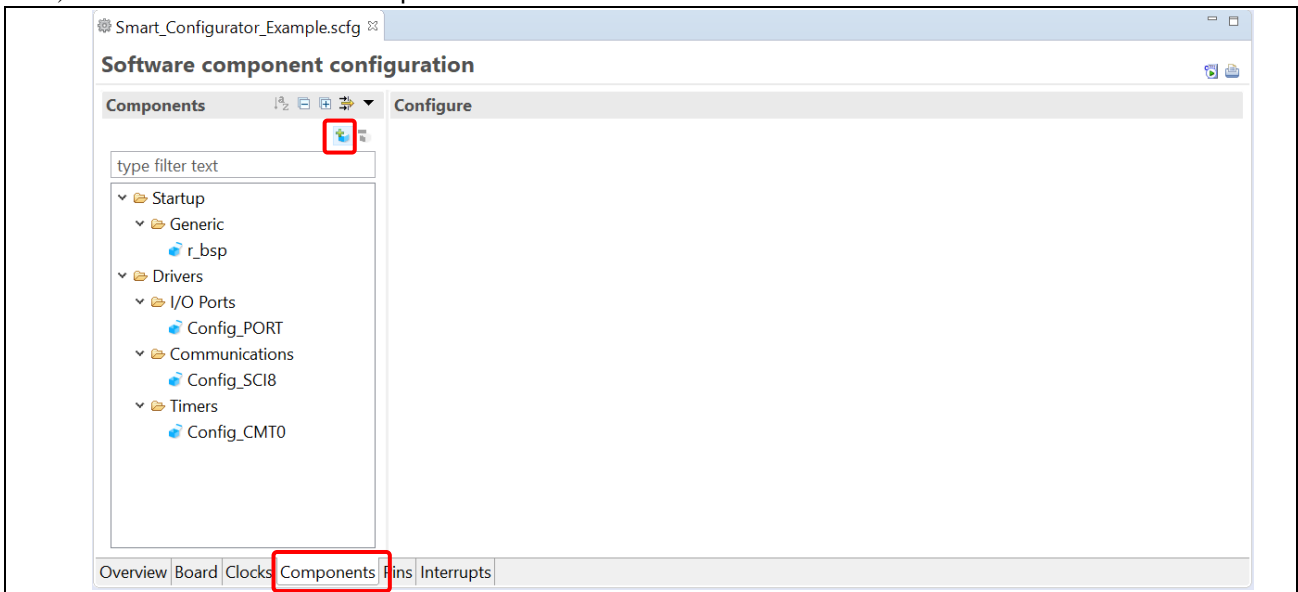


Figure 5-10 Software component configuration in Smart Configurator

- 3) Add DMA Controller driver into the project
 - Type “DMA” to the “Filter” box.
 - Navigate the component list and select *DMA Controller* driver
 - Click [Next] to continue

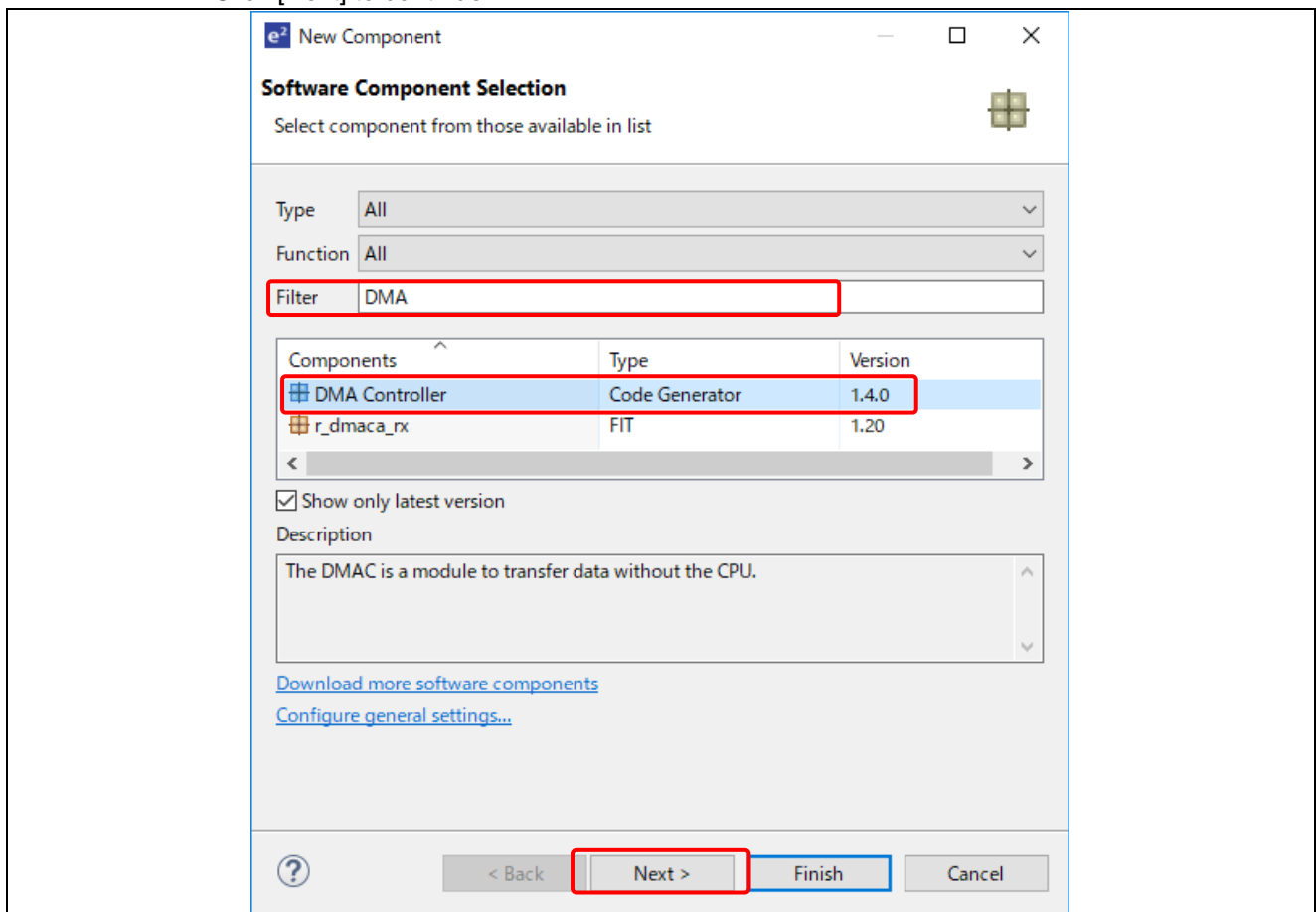


Figure 5-11 Select software component

- 4) In the [New Component] dialog box
 - Select *DMAC1* as resource
 - Keep the default configuration name. Source files and API will be generated based on this configuration name
 - Click [Finish]

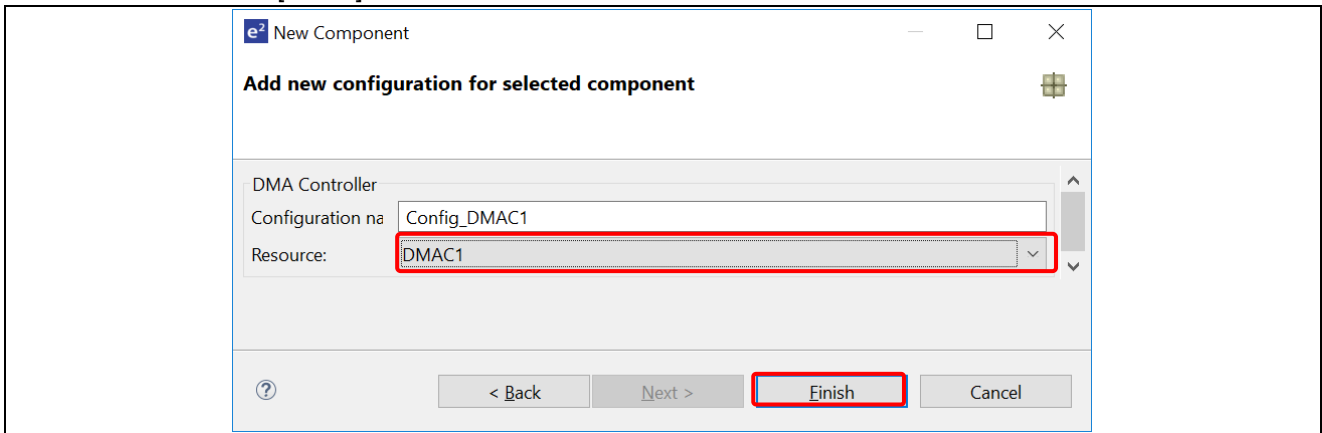


Figure 5-12 Add new configuration to the project

- 5) At Configure pane, ensure settings as the following:
 - Activation source: Software trigger
 - Keep other default settings

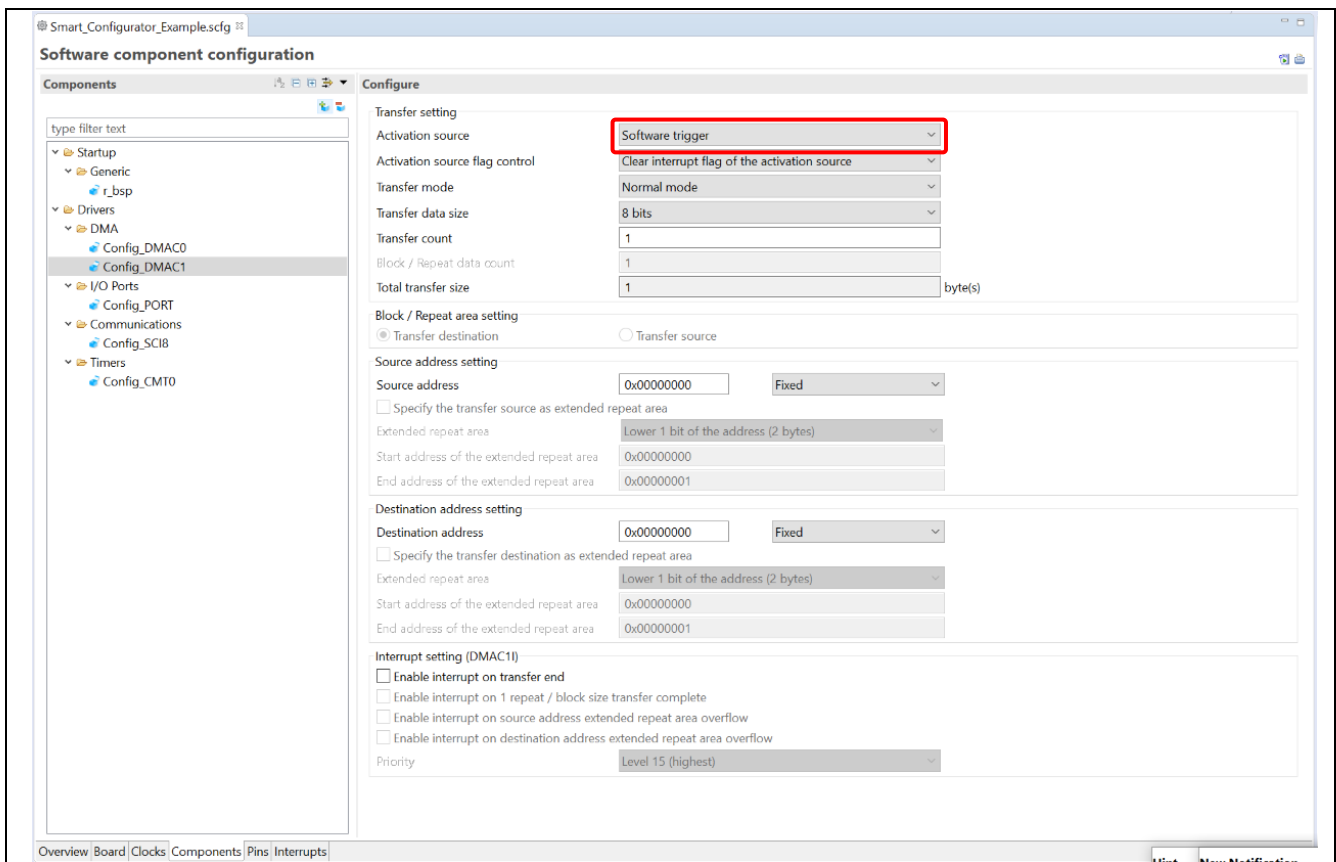


Figure 5-13 DMAC1 driver configuration

5.1.3 Modify SCI8 driver

- 1) In the Smart_Configurator_Example.scfg pane, click [Components] tab
- 2) Select *Config_SCI8* in the [Component] panel
- 3) At Configure pane, modify settings as the following:
 - Transmit data handling: Data handled by DMAC
 - Receive data handling: Data handled by DMAC
 - Keep other settings

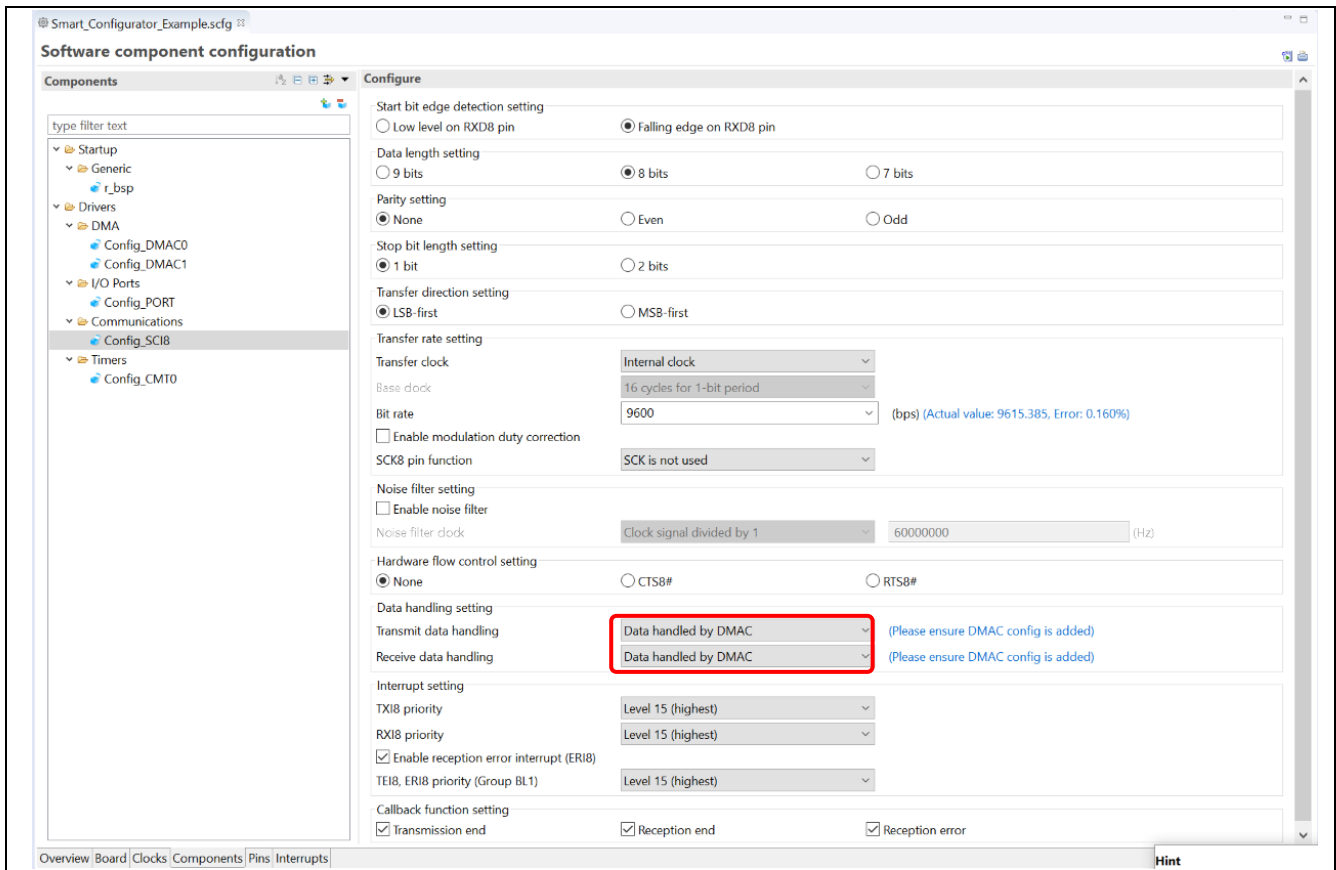



Figure 5-14 SCI8 driver configuration

5.2 Generating codes

At [Software Component Configuration] tab, click  to generate codes

5.3 Adding application codes in DMAC0 driver

At Project Explore tree, open "Config_DMAC0_user.c" in \src\smc_gen\Config_DMAC0 folder

- a. Add the declarations below at the user code area near the top of the file:

```
/* Global variable used to receive a character from SCI8 */
volatile uint8_t g_rx_char_dmac;

/* Flag used to detect whether data is received */
volatile uint8_t g_rx_flag_dmac;
```

- b. Add code to set the initial value for the reception completion flag, DMAC0 source address and DMAC0 destination address in *R_Config_DMAC0_Create_UserInit(void)* function

```
/* Clear receive flag */
g_rx_flag_dmac = 0U;

/* Set DMAC0 source address */
DMAC0.DMSAR = (void *)&SCI8.RDR;

/* Set DMAC0 destination address */
DMAC0.DMDAR = (void *)&g_rx_char_dmac;
```

- c. In *r_dmac0_callback_transfer_end(void)* function:

- i. Set the reception completion flag
- ii. Restart reception by setting the transfer count and enable DMAC0 transfer

```
/* Set receive completion flag */
g_rx_flag_dmac = 1U;

/* Set DMAC0 transfer count */
DMAC0.DMCRA = _00000001_DMAC0_DMCRA_COUNT;

/* Enable DMAC0 transfer */
R_Config_DMAC0_Start();
```

Your source file should look like this:

```

2      /* DISCLAIMER */
19
21      /* File Name      : Config_DMACE0_user.c */
27
29      /*Pragma directive*/
31      /* Start user code for pragma. Do not edit comment generated here */
32      /* End user code. Do not edit comment generated here */
33
35      /*Includes*/
37      #include "r_cg_macrodriver.h"
38      #include "Config_DMACE0.h"
39      /* Start user code for include. Do not edit comment generated here */
40      /* End user code. Do not edit comment generated here */
41      #include "r_cg_userdefine.h"
42
44      /*Global variables and functions*/
46      /* Start user code for global. Do not edit comment generated here */
47      /* Global variable used to receive a character from SCI8 */
48      volatile uint8_t g_rx_char_dmac;
49
50      /* Flag used to detect whether data is received */
51      volatile uint8_t g_rx_flag_dmac;
52      /* End user code. Do not edit comment generated here */
53
55      /* Function Name: R_Config_DMACE0_Create_UserInit */
60
61      void R_Config_DMACE0_Create_UserInit(void)
62      {
63          /* Start user code for user init. Do not edit comment generated here */
64          /* Clear receive flag */
65          g_rx_flag_dmac = 0U;
66
67          /* Set DMACE0 source address */
68          DMACE0.DMSAR = (void *) &SCI8.RDR;
69
70          /* Set DMACE0 destination address */
71          DMACE0.DMDAR = (void *) &g_rx_char_dmac;
72          /* End user code. Do not edit comment generated here */
73      }
74
-----
97      /* Function Name: r_dmac0_callback_transfer_end */
102
103      static void r_dmac0_callback_transfer_end(void)
104      {
105          /* Start user code for r_dmac0_callback_transfer_end. Do not edit comment generated here */
106          /* Set receive completion flag */
107          g_rx_flag_dmac = 1U;
108
109          /* Set DMACE0 transfer count */
110          DMACE0.DMCRA = _00000001_DMACE0_DMCRA_COUNT;
111
112          /* Enable DMACE0 transfer */
113          R_Config_DMACE0_Start();
114          /* End user code. Do not edit comment generated here */
115      }
116
117      /* Start user code for adding. Do not edit comment generated here */
118      /* End user code. Do not edit comment generated here */

```

Figure 5-15 Config_DMACE0_user.c

5.4 Adding application codes in DMACE1 driver

At Project Explore tree, open "Config_DMACE1_user.c" in \src\smc_gen\Config_DMACE1 folder

- a. Add header file at the user code area near the top of the file:

```
#include "Config_SCI8.h"
```

- b. Add code to set the initial value for DMACE1 destination address in R_Config_DMACE1_Create_UserInit(void) function

```
/* Set DMACE1 destination address */
DMACE1.DMDAR = (void *)&SCI8.TDR;
```


- c. Add a function to transfer data to SCI8 and call SCI8 function to send data to PC at user code area at the bottom of the file

```

/*****
*****
* Function Name: R_DMAM1_AsyncTransmit
* Description : This function transfer data to SCI8 and call SCI8
function to send data to PC.
* Arguments : tx_buf -
*             transfer buffer pointer
*             tx_num -
*             buffer size
* Return Value : status -
*               MD_OK or MD_ARGERROR
*****
*****/
MD_STATUS R_DMAM1_AsyncTransmit(uint8_t * const tx_buf, const uint16_t
tx_num)
{
    MD_STATUS status = MD_OK;

    uint8_t * source_address = tx_buf;

    // Set DMAM1 transfer count
    DMAM1.DMCRA = tx_num;

    // Enables DMA transfer
    R_Config_DMAM1_Start();

    while(DMAM1.DMCRA > 0)
    {
        // Set DMAM1 source address
        DMAM1.DMSAR = (void *) source_address;
        source_address++;

        // DMA transfer request
        R_Config_DMAM1_Set_SoftwareTrigger();

        // Sends SCI8 data
        R_SCI8_AsyncTransmit(NULL, 1);
    }

    return (status);
}
/*****
*****
* End of function R_DMAM1_AsyncTransmit
*****
*****/

```

Your source file should look like this:

```

2
19
21
27
29
31
32
33
35
37
38
39
40
41
42
43
45
47
48
49
51
56
57
58
59
60
61
62
63
64
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
25
```

5.5 Modifying application codes in SCI8 driver

- 1) At Project Explore tree, open "Config_SCI8_user.c" in \src\smc_gen\Config_SCI8 folder
 - a. Remove the declarations "g_rx_char" and "g_rx_flag" at the user code area near the top of the file:

```

/* Global variable used to receive a character from PC terminal */
volatile uint8_t g_rx_char;

/* Flag used to detect whether data is received */
volatile uint8_t g_rx_flag;

/* Flag used to detect completion of transmission */
static volatile uint8_t SCI8_txdone;


```

- b. Remove code in `r_Config_SCI8_callback_receiveend` (void) function

```

/* Set receive completion flag */
g_rx_flag = 1U;

/* Set SCI8 receive buffer address and restart reception */
R_Config_SCI8_Serial_Receive((uint8_t *)&g_rx_char, 1);


```

Your source file should look like this:

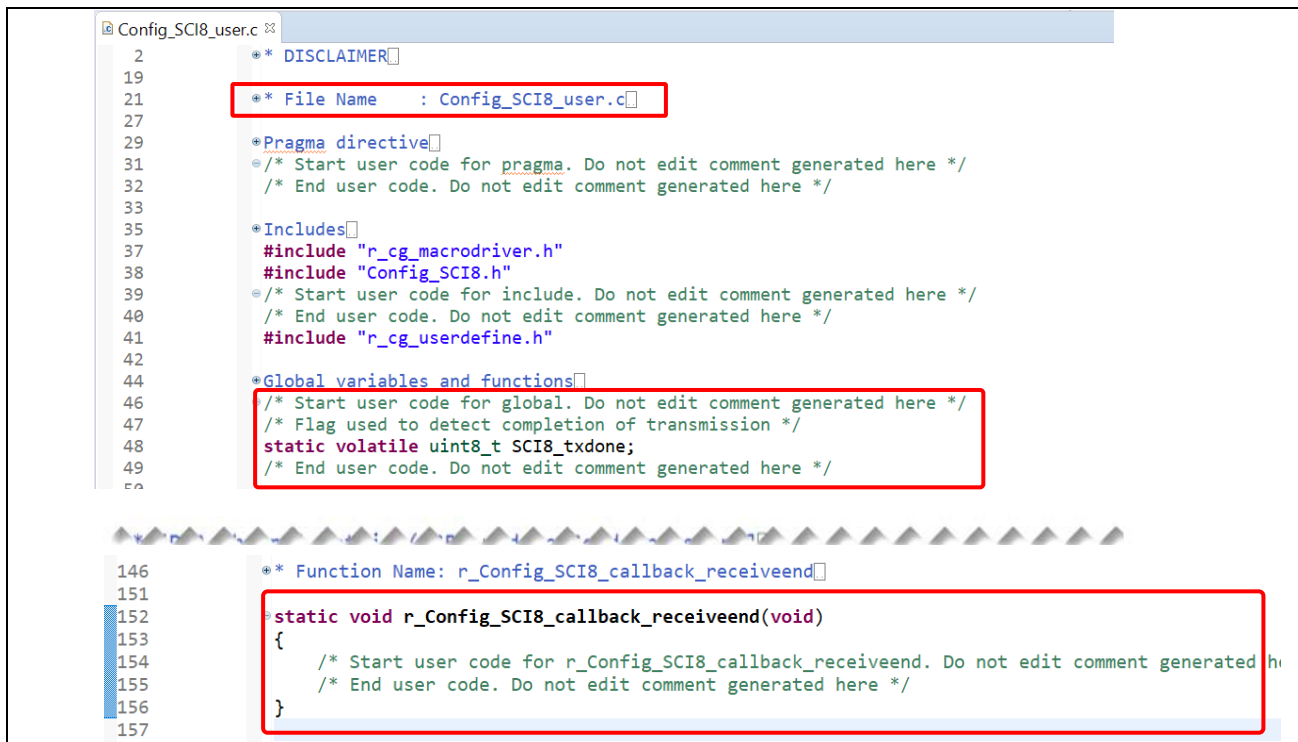


Figure 5-17 Config_SCI8_user.c

5.6 Adding application codes in *main()*

- 1) Near the top of the file Smart_Configurator_Example.c, there are 2 external declarations (g_rx_char and g_rx_flag) which are used for SCI8 and are defined in file Config_SCI8_user.c. Because we use DMAC instead of SCI8, they become obsolete now. Remove them.

```
/* Flag used to detect whether data is received from PC terminal */  
extern volatile uint8_t g_rx_flag;  
  
/* Global variable used for storing data received from PC terminal */  
extern volatile uint8_t g_rx_char;
```

Add new external declarations for DMAC0 variables. (Defined in DMAC Config_DMAC0_user.c in section 5.3)

```
/* Flag used to detect whether data is received from PC terminal */  
extern volatile uint8_t g_rx_flag_dmac;  
  
/* Global variable used for storing data received from PC terminal */  
extern volatile uint8_t g_rx_char_dmac;
```

2) Replace all codes inside *main()* function with below codes:

```

/* Start CMT0 counter operation */
R_Config_CMT0_Start();

/* Set SCI8 receive buffer address and enable receive interrupt */
R_Config_SCI8_Serial_Receive(NULL, 0);

/* Enable DMAC0 operation */
R_Config_DMAC0_Start();

/* Enable SCI8 operation */
R_Config_SCI8_Start();

/* Print instruction onto PC terminal */
sprintf(print_str, "\r\n Press keyboard to control the blinking rate of LED2 (not case sensitive)\r\n");
R_DMAC1_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));
sprintf(print_str, "\r\n a ----> Slower\r\n");
R_DMAC1_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));
sprintf(print_str, "\r\n b ----> Faster\r\n");
R_DMAC1_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));

while (1U)
{
    /* Get new blink interval level from PC terminal */
    if (g_rx_flag_dmac == 1U)
    {
        /* Instruction to slow down blinking rate is received */
        if (('A' == g_rx_char_dmac) || ('a' == g_rx_char_dmac))
        {
            R_Config_SCI8_Serial_Receive(NULL, 0);

            /* Notify the character received and inform next action */
            sprintf(print_str, "\r\n Character 'a' or 'A' is received. LED2 will blink at slower
rate.\r\n");

            R_DMAC1_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));

            /* Get new blink interval level. Maximum level is 9. */
            if (interval_level < 10)
            {
                interval_level++;
            }
            else
            {
                sprintf(print_str, "\r\n This is minimum. Please press 'b' or 'B' to blink faster.
\r\n");

                R_DMAC1_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));
            }
        }
        /* Instruction to increase blinking rate is received */
        else if (('B' == g_rx_char_dmac) || ('b' == g_rx_char_dmac))
        {
            R_Config_SCI8_Serial_Receive(NULL, 0);

            /* Notify the character received and inform next action */
            sprintf(print_str, "\r\n Character 'b' or 'B' is received. LED2 will blink at faster
rate.\r\n");

            R_DMAC1_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));

            /* Get new blink interval level. Minimum level is 1 */
            if (interval_level > 1)
            {
                interval_level--;
            }
            else
            {
                sprintf(print_str, "\r\n This is maximum. Please press 'a' or 'A' to blink slower.
\r\n");

                R_DMAC1_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));
            }
        }
        else
        {
            /* Do nothing */
        }

        /* Change blinking rate */
        CMT0.CMCOR = (uint16_t) (5000 * interval_level);

        /* Reset SCI8 reception flag*/
        g_rx_flag_dmac = 0U;
    }
    else
    {
        /* Do nothing */
    }
}

```

Your source file should look like this:

```

3  * * FILE      : Smart_Configurator_Example.c[]
10 #include "r_smc_entry.h"
11 #include <stdio.h>
12 #include <string.h>
13
14 /* Global variable for changing CMT0 interval */
15 volatile uint16_t interval_level = 5;
16
17 /* String used to print message at PC terminal */
18 static char print_str[80];
19
20 /* Flag used to detect whether data is received from PC terminal */
21 extern volatile uint8_t g_rx_flag_dmac;
22
23 /* Global variable used for storing data received from PC terminal */
24 extern volatile uint8_t g_rx_char_dmac;
25
26 void main(void);
27
28 void main(void)
29 {
30     /* Start CMT0 counter operation */
31     R_Config_CMT0_Start();
32
33     /* Set SCI8 receive buffer address and enable receive interrupt */
34     R_Config_SCI8_Serial_Receive(NULL, 0);
35
36     /* Enable DMAC0 operation */
37     R_Config_DMAC0_Start();
38
39     /* Enable SCI8 operation */
40     R_Config_SCI8_Start();
41
42     /* Print instruction onto PC terminal */
43     sprintf(print_str, "\r\n Press keyboard to control the blinking rate of LED2 (not case sensitive)\r\n");
44     R_DMAC1_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));
45     sprintf(print_str, "\r\n a ----> Slower\r\n");
46     R_DMAC1_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));
47     sprintf(print_str, "\r\n b ----> Faster\r\n");
48     R_DMAC1_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));
49
50     while (1U)
51     {
52         /* Get new blink interval level from PC terminal */
53         if (g_rx_flag_dmac == 1U)
54         {
55             /* Instruction to slow down blinking rate is received */
56             if (('A' == g_rx_char_dmac) || ('a' == g_rx_char_dmac))
57             {
58                 R_Config_SCI8_Serial_Receive(NULL, 0);
59
60                 /* Notify the character received and inform next action */
61                 sprintf(print_str, "\r\n Character 'a' or 'A' is received. LED2 will blink at slower rate.\r\n");
62                 R_DMAC1_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));
63
64                 /* Get new blink interval level. Maximum level is 9. */
65                 if (interval_level < 10)
66                 {
67                     interval_level++;
68                 }
69                 else
70                 {
71                     sprintf(print_str, "\r\n This is minimum. Please press 'b' or 'B' to blink faster. \r\n");
72                     R_DMAC1_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));
73                 }
74             }
75             /* Instruction to increase blinking rate is received */
76             else if (('B' == g_rx_char_dmac) || ('b' == g_rx_char_dmac))
77             {
78                 R_Config_SCI8_Serial_Receive(NULL, 0);
79
80                 /* Notify the character received and inform next action */
81                 sprintf(print_str, "\r\n Character 'b' or 'B' is received. LED2 will blink at faster rate.\r\n");
82                 R_DMAC1_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));
83
84                 /* Get new blink interval level. Minimum level is 1 */
85                 if (interval_level > 1)
86                 {
87                     interval_level--;
88                 }
89                 else
90                 {
91                     sprintf(print_str, "\r\n This is maximum. Please press 'a' or 'A' to blink slower. \r\n");
92                     R_DMAC1_AsyncTransmit((uint8_t *) print_str, (uint16_t) strlen(print_str));
93                 }
94             } else
95             {
96                 /* Do nothing */
97             }
98
99             /* Change blinking rate */
100            CMT0.CMCOR = (uint16_t) (5000 * interval_level);
101
102            /* Reset SCI8 reception flag*/
103            g_rx_flag_dmac = 0U;
104        }
105        else
106        {
107            /* Do nothing */
108        }
109    }
110 }
111

```

5.7 Build and run on hardware board

Refer to section 4.6

5.8 Operation on board

Refer to section 4.7

6. Application Example 5 (Integration with USB function using Smart Configurator and FIT module)

In previous chapter, serial communication (asynchronous mode) is used to control LED2 blinking rate.

This chapter describes how to control LED2 blinking rate with terminal program on PC via USB serial communication. In this application example, two FIT modules will be integrated to the project to emulate a serial COM port on RX65N for the communication with PC:

R01AN2025: USB Basic Host and Peripheral Driver

R01AN2030: USB Peripheral Communications Device Class Driver (PCDC)

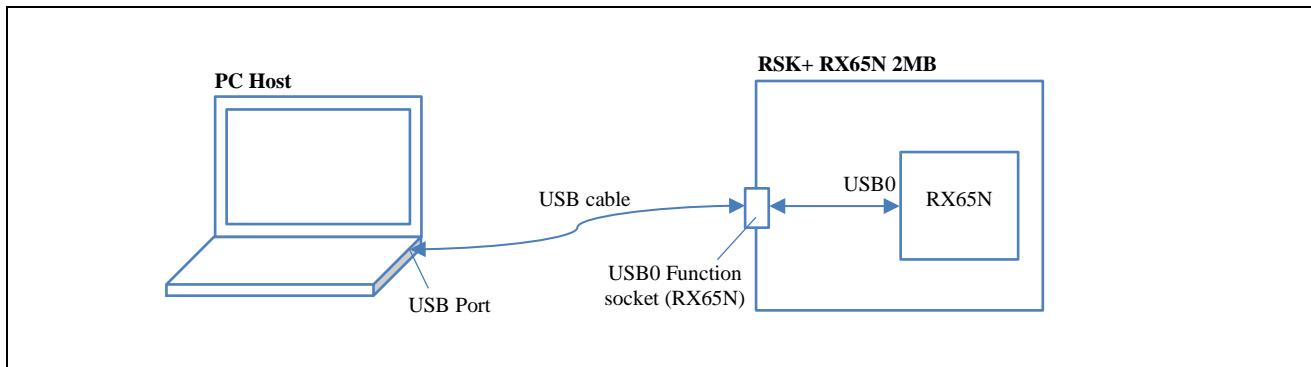


Figure 6-1 Control LED blinking rate by terminal program via USB serial communication

The schematic diagram of Renesas Starter Kit+ for RX65N 2MB is shown below. In this example, USB0_VBUS (P16) will be configured before using USB0 function. Ensure J7 Pin2-Pin3 are shorted (refer to Chapter 5.8 for J7 location).

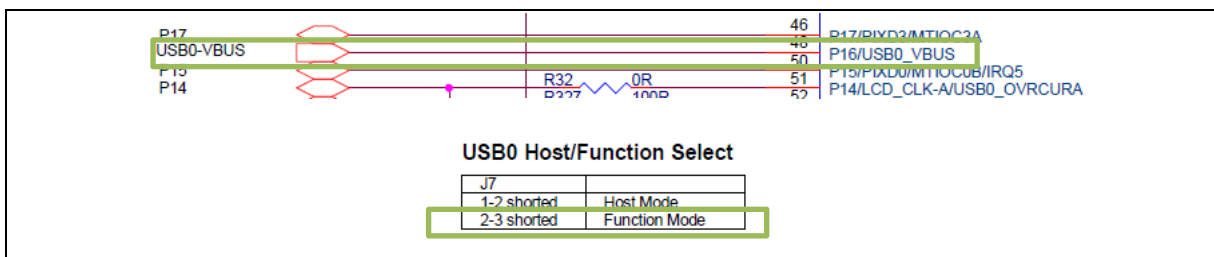


Figure 6-2 Schematic diagram of Renesas Starter Kit+ for RX65N 2MB

The table below shows changes of the driver selection:

	Drivers	Resource	Function
1. Remove driver	SCI/SCIF Asynchronous Mode	SCI8	Communication between PC terminal and target board for controlling LED blinking rate
	DMA Controller	DMAC0	Transfer data from SCI8 to RAM (Received)
		DMAC1	Transfer data from RAM to SCI8 (Transmit)
2. Add new driver	r_usb_basic	USB0	USB basic host and peripheral firmware Performs USB hardware control
	r_usb_pcdc	USB0	Operates as a USB peripheral communications device class driver (PCDC) Enables communication with a USB host

USB PCDC Mode application codes will be added to the program as below:

a) Main function:

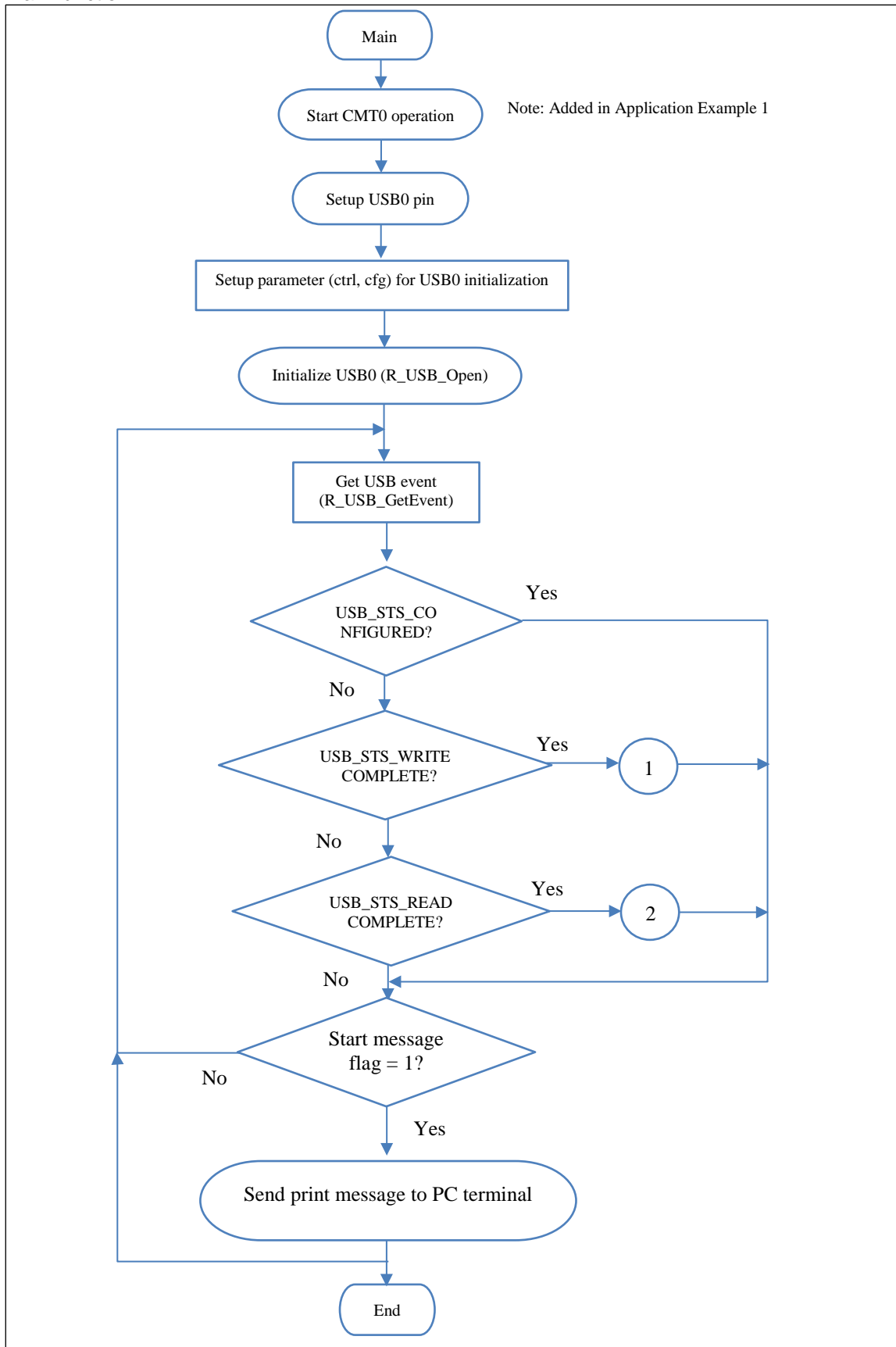


Figure 6-3 Main flowchart

b) Process when USB_STS_WRITE_COMPLETE is detected:

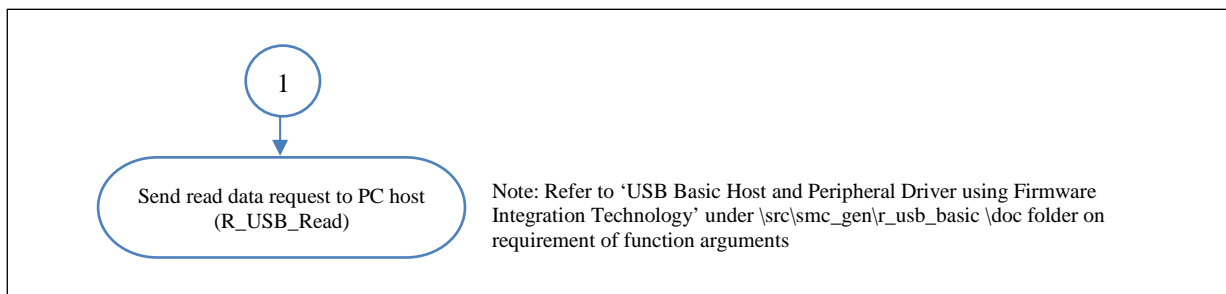


Figure 6-4 Process when USB_STS_WRITE_COMPLETE is detected

c) Process when USB_STS_READ_COMPLETE is detected:

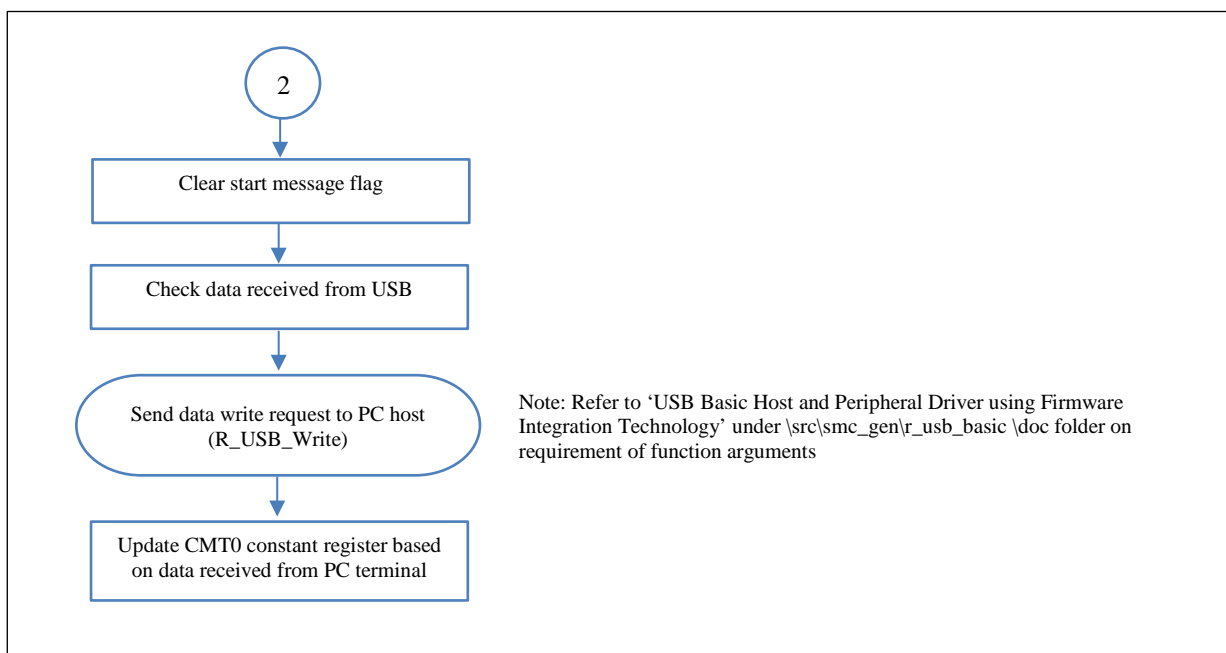



Figure 6-5 Process when USB_STS_READ_COMPLETE is detected

6.1 Removing SCI/SCIF Asynchronous Mode driver and related application codes

- 1) In Smart_Configurator_Example.scfg pane, click the [Components] tab
- 2) At Components tree, click to select *Config_SCI8* configuration. Click  to remove this configuration. Repeat the same steps to remove *Config_DMAC0* and *Config_DMAC1*.

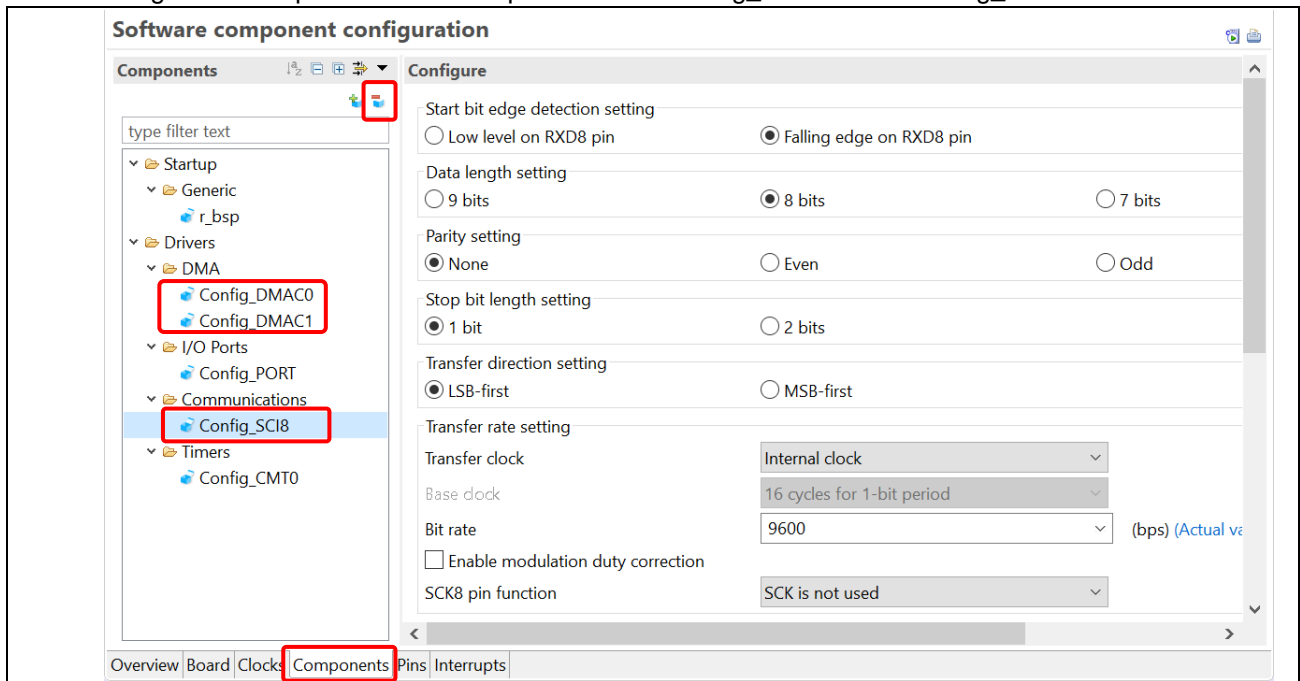



Figure 6-6 Software component configuration in Smart Configurator

- 3) Click  to generate codes
The source folder *Config_SCI8*, *Config_DMAC0* and *Config_DMAC1* will be moved to project \trash folder before the code generation operation
- 4) At the project tree, open "Smart_Configurator_Example.c" in \src folder

- 5) Remove the application codes which are related to previous example
 - a. Remove declarations for variables

```

#include <stdio.h>
#include <string.h>

/* Global variable used for changing CMT0 interval */
volatile uint16_t interval_level = 5;

/* String used to print message at PC terminal */
static char print_str[70];

/* Flag used to detect whether data is received from PC terminal */
extern volatile uint8_t g_rx_flag_dmac;

/* Global variable used for storing data received from PC terminal */
extern volatile uint8_t g_rx_char_dmac;


```

- b. Remove all codes after *R_Config_CMT0_Start* function

After deleted the codes related to SCI8, the main file will look like following:


```

3      /* FILE      : Smart_Configurator_Example.c
10     #include "r_smc_entry.h"
11
12
13     void main(void);
14
15     void main(void)
16     {
17         /* Start CMT0 counter operation */
18         R_Config_CMT0_Start();
19
20
21     }
22

```

Figure 6-7 Main file after deleted codes related to SCI8

6.2 Adding a peripheral driver

- 1) In the Smart_Configurator_Example.scfg pane, click [Components] tab
- 2) Click  to add new component

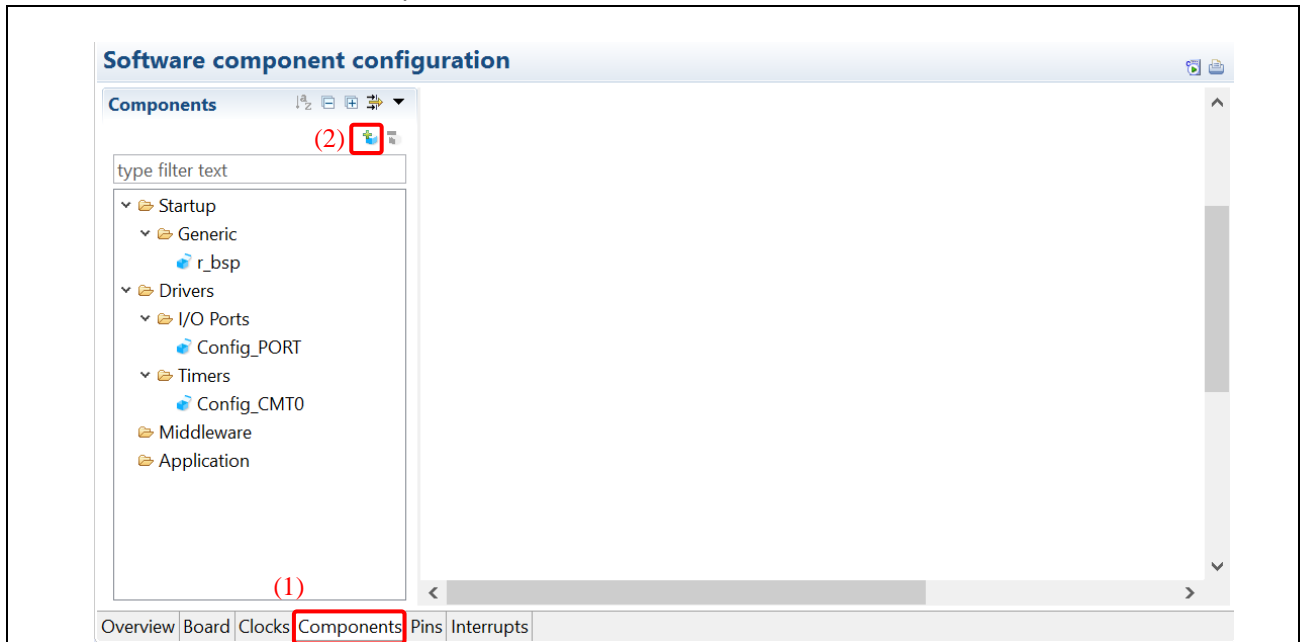


Figure 6-8 Software component configuration in Smart Configurator

- 3) Add *USB Basic* and *USB PCDC* drivers into the project
 - a. Enter “usb” in the “Filter” column
 - b. Navigate the component list and select *r_usb_basic* driver
 - c. Press and hold Ctrl key, click on *r_usb_pcdc* driver
 Note: If above drivers were not available in the list, click [Download more software components] to download the FIT modules.
 - d. Click [Finish]

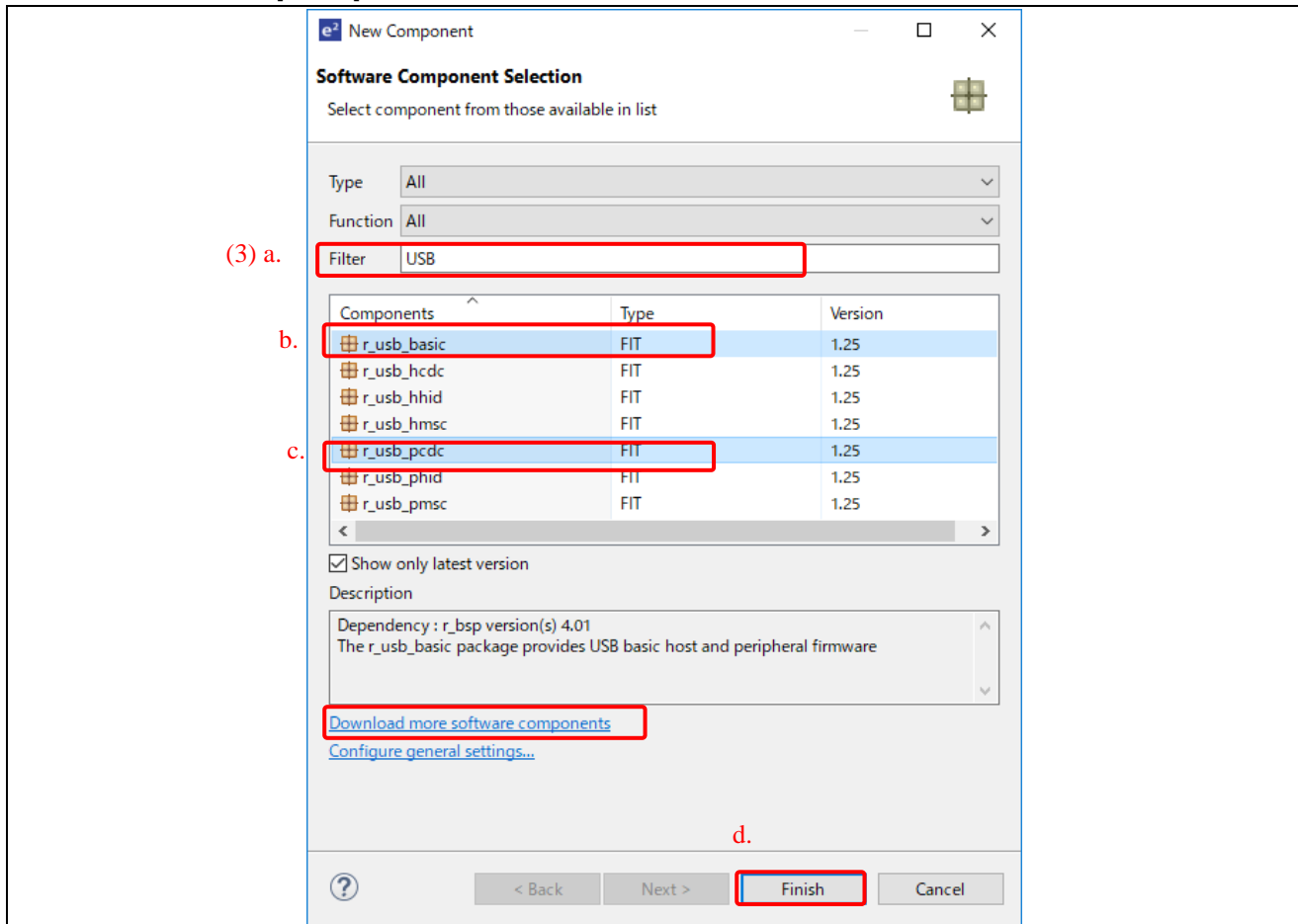


Figure 6-9 Select software component

- 4) In the component page
 - a. Click *r_usb_basic* to open the configure tab
 - b. Under “Configuration”, set [Setting whether the received class request is supported.] to “Not supporting the class request”.
 - c. Tick the checkbox for [USB0_PERI] & [USB0_VBUS Pin]

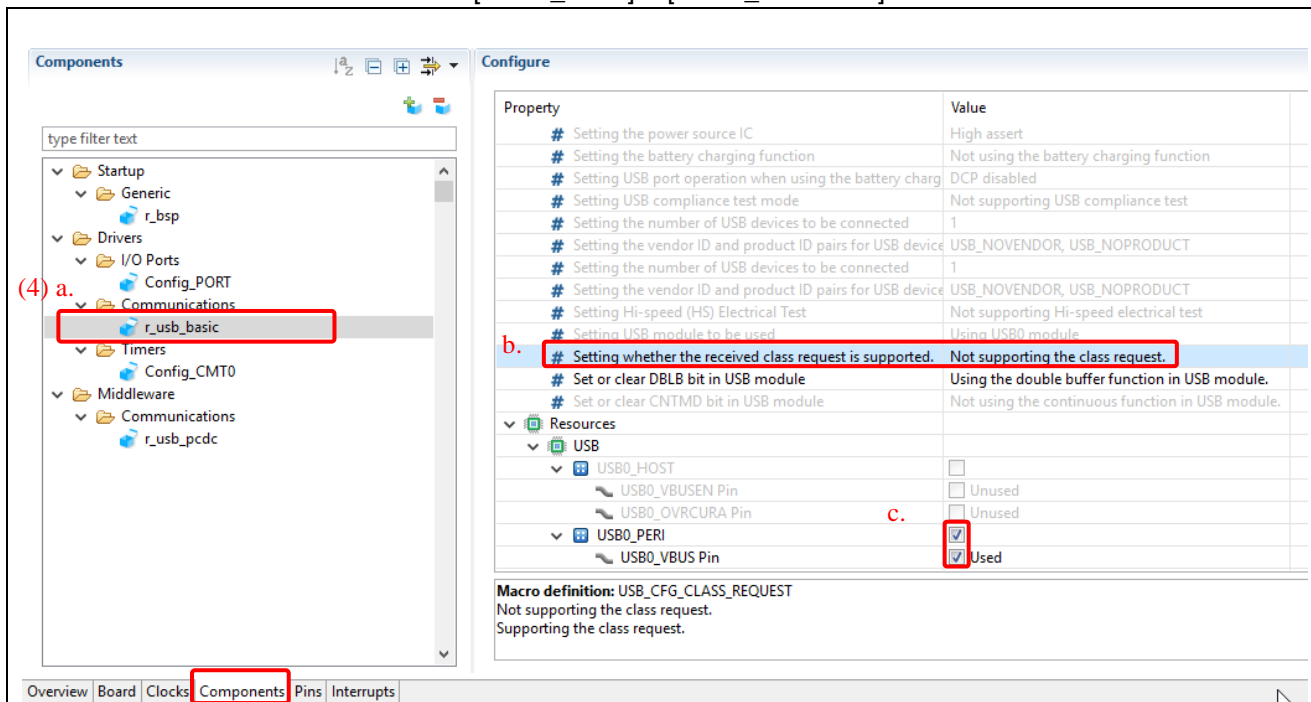



Figure 6-10 configure software component

- 5) In the pins page
 - a. Click 
 - b. Click *r_usb_basic* to open the corresponding Pin Function tab
 - c. Tick the checkbox for [USB0_VBUS]
 - d. Click on the corresponding [Assignment] row to assign P16 to function [USB0_VBUS]

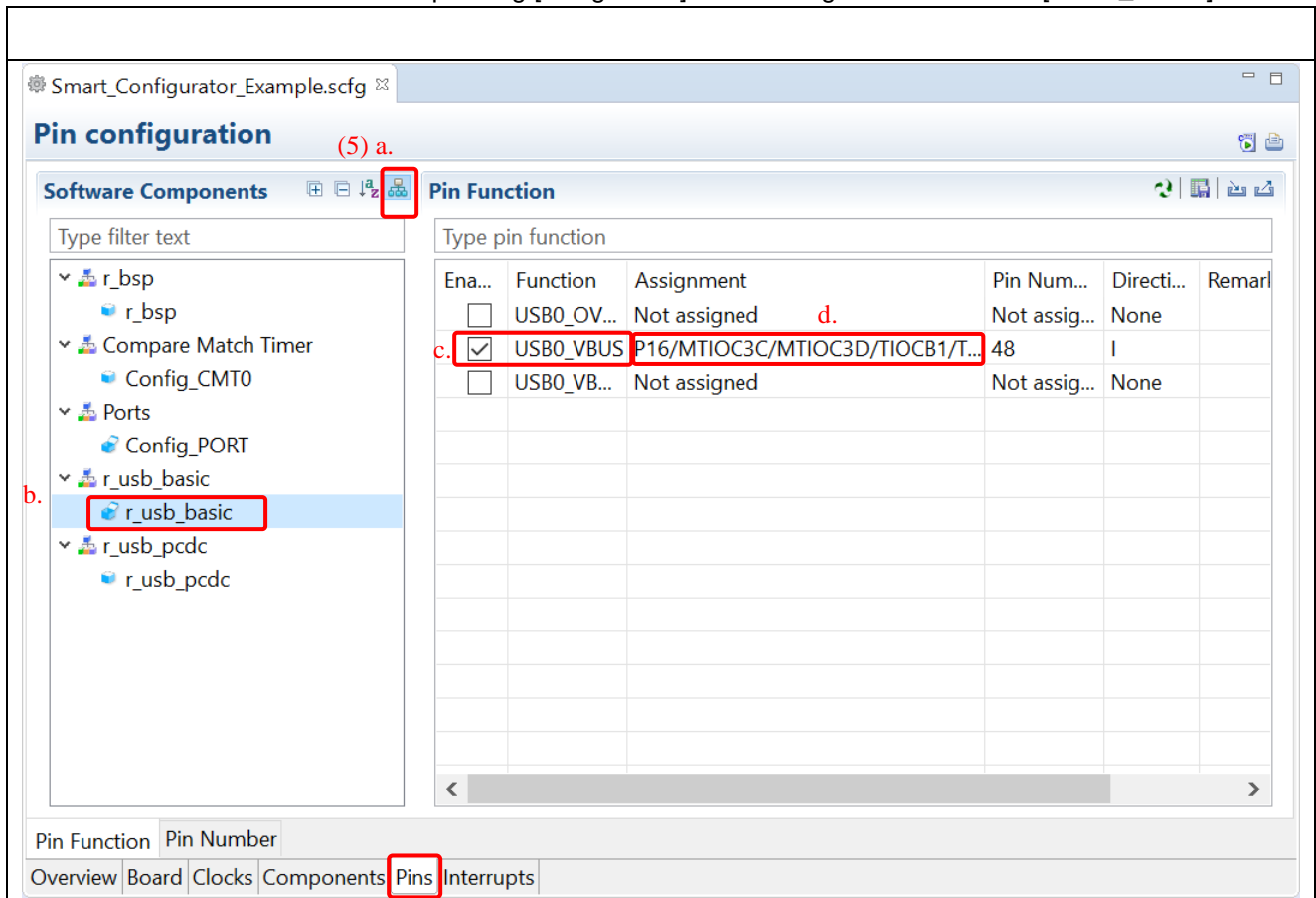



Figure 6-11 configure software component

6.3 Generating codes

- 1) At [Software Component Configuration] tab, click  to generate codes
- 2) *USB Basic Host and Peripheral Driver* and *USB PCDC Driver* codes are generated into three folders under \src\smc_gen:

Folder	Sub-folder/File	Description
r_usb_basic	doc	Application Note of <i>USB Basic Host and Peripheral Driver (r_usb_basic)</i>
	ref	Reference of configuration file
	src	Source files and header files
	<i>r_usb_basic_if.h</i>	List of all API calls and interface definitions of <i>r_usb_basic</i> Refer to this file for operation in application
r_usb_pcdc	doc	Application Note of <i>USB PCDC Driver (r_usb_pcdc)</i>
	ref	Reference of configuration file
	src	Source files and header files
	utilities	Contains system definition file (CDC_Demo_Win7.inf) Install this driver for Windows 7 PC host
	<i>r_usb_pcdc_if.h</i>	List of all API calls and interface definitions of <i>r_usb_pcdc</i> Refer to this file for operation in application
r_config	<i>r_usb_basic_config.h</i>	Configuration of <i>r_usb_basic</i> Configure this file as the requirement of application Refer to Application Note of <i>r_usb_basic</i> before modifying this file
	<i>r_usb_pcdc_config.h</i>	Configuration of <i>r_usb_pcdc</i> Configure this file as the requirement of application Refer to Application Note of <i>r_usb_basic</i> and <i>r_usb_pcdc</i> before modifying this file

- 3) In this application example:
 - a. USB driver is configured to peripheral mode (device)
 - b. A header file is created for construction of descriptor: *r_usb_pcdc_descriptor.h*
 - c. API functions that are called in application codes: *R_USB_Open*, *R_USB_Read*, *R_USB_Write*
 - d. USB driver status that are checked for operation in application codes:
USB_STS_WRITE_COMPLETE, *USB_STS_READ_COMPLETE*

6.4 Adding header file of USB PCDC driver

- 1) At Project Explorer tree, right click [src] folder, select [New] → [Header file]

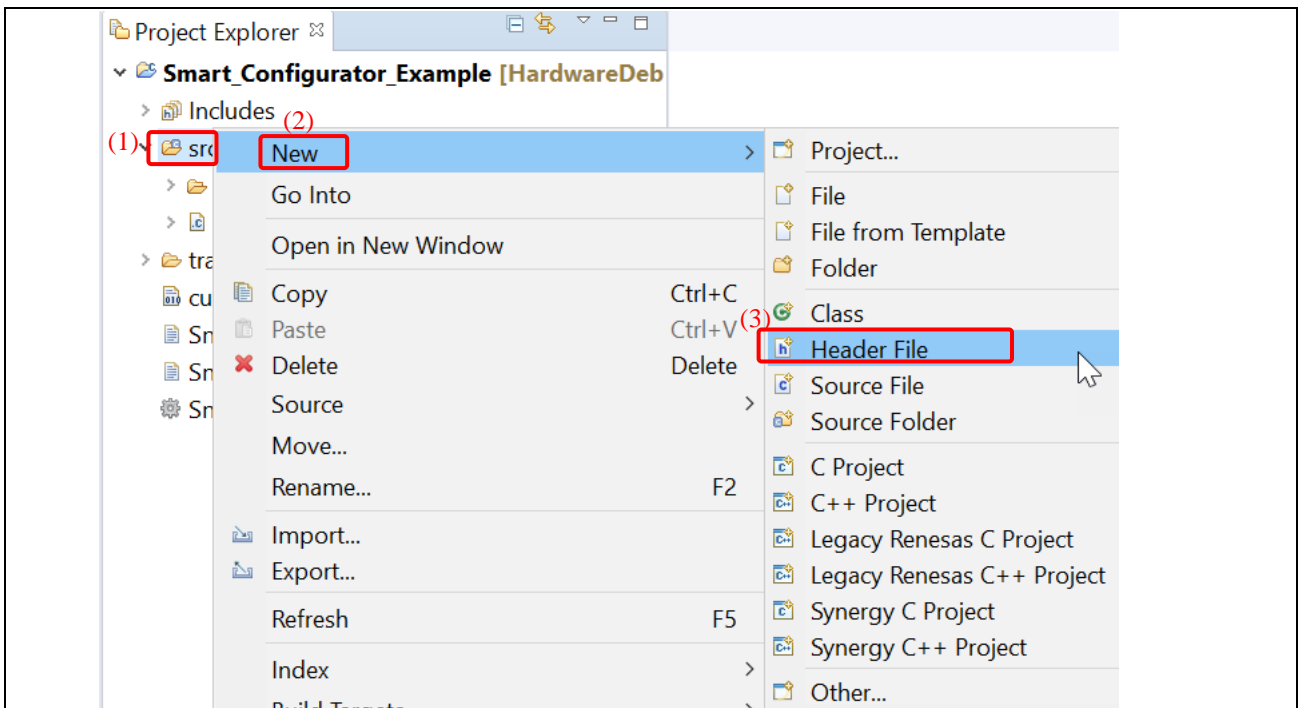


Figure 6-12 Adding head file

- 2) Input header file name (e.g. `r_usb_pcdc_descriptor.h`), click [Finish]

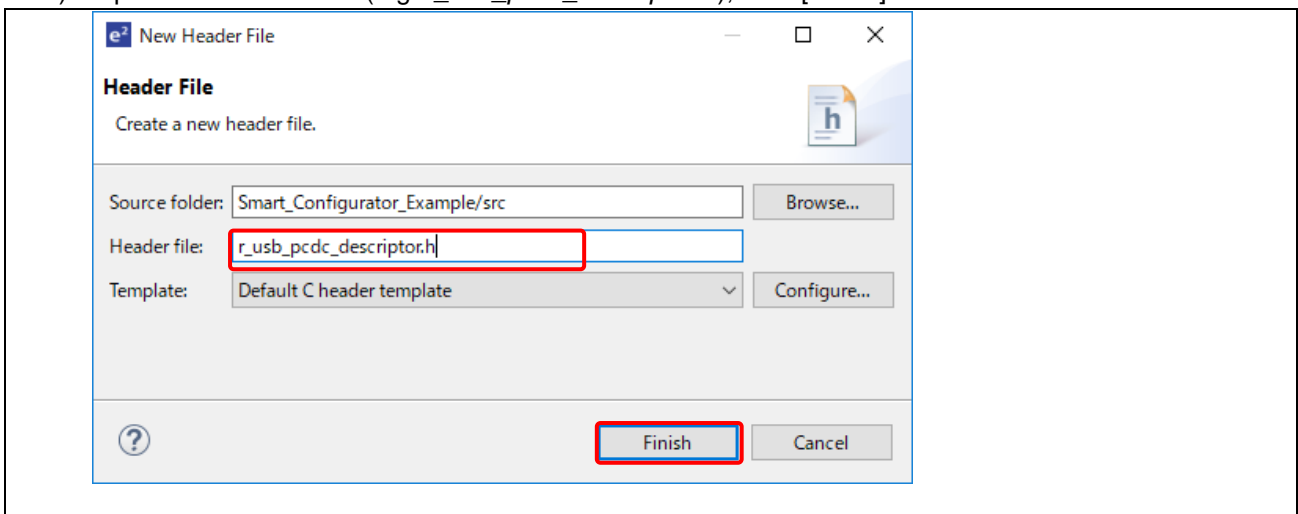


Figure 6-13 Adding header file

- 3) Open “`r_usb_pcdc_descriptor.h`” file in `\src` folder

Refer to the application note of ‘USB Basic Host and Peripheral Driver using Firmware Integration Technology’ under `\src\smc_gen\r_usb_basic\doc` folder or USB-IF website (<http://www.usb.org>) for more information on USB 2.0 specification

Add below codes in “`r_usb_pcdc_descriptor.h`” to construct the descriptor for this application example:

Note:

Please be sure to specify your vendor ID and product ID in the device descriptor and INF file.

```

#include "r_usb_basic_if.h"
/*****
Macro definitions
*****/
/* bcdUSB */
#define USB_BCDNUM      (0x0200u)      /* 0x0100 (USB1.0), 0x0110 (USB1.1), 0x0200 (USB2.0) */
/* Release Number */
#define USB_RELEASE     (0x0200u)      /* 0x0100 (USB1.0), 0x0110 (USB1.1), 0x0200 (USB2.0) */
/* DCP max packet size */
#define USB_DCPMAXP     (64u)          /* Max packet size for endpoint 0. 8/16/32/64 */
/* Configuration number */
#define USB_CONFIGNUM   (1u)           /* Total number of possible configurations for the device */
/* Vendor ID */
#define USB_VENDORID    (0x0000u)      /* Must be obtained from USB-IF */
/* Product ID*/
#define USB_PRODUCTID   (0x0002u)      /* Assigned by the manufacturer */

/* Class-Specific Configuration Descriptors */
#define USB_PCDC_CS_INTERFACE (0x24u) /* Assigned by USB-IF*/

/* bDescriptor SubType in Communications Class Functional Descriptors */
/* Header Functional Descriptor */
#define USB_PCDC_DT_SUBTYPE_HEADER_FUNC (0x00u) /* Assigned by USB-IF */
/* Call Management Functional Descriptor. */
#define USB_PCDC_DT_SUBTYPE_CALL_MANAGE_FUNC (0x01u) /* Assigned by USB-IF */
/* Abstract Control Management Functional Descriptor. */
#define USB_PCDC_DT_SUBTYPE_ABSTRACT_CTR_MANAGE_FUNC (0x02u) /* Assigned by USB-IF */
/* Union Functional Descriptor */
#define USB_PCDC_DT_SUBTYPE_UNION_FUNC (0x06u) /* Assigned by USB-IF */

/* Communications Class Subclass Codes */
#define USB_PCDC_CLASS_SUBCLASS_CODE_ABS_CTR_MDL (0x02u) /*SubClass Code assigned by USB-IF */

/* USB Class Definitions for Communications Devices Specification
release number in binary-coded decimal. */
#define USB_PCDC_BCD_CDC (0x0110u) /* Assigned by USB-IF */

/* Descriptor length */
#define USB_PCDC_DD_LEN (18u)
#define USB_PCDC_QD_LEN (10u)
#define USB_PCDC_CD1_LEN (67u)
#define STRING_DESCRIPTOR0_LEN (4u)
#define STRING_DESCRIPTOR1_LEN (16u)
#define STRING_DESCRIPTOR2_LEN (8u)
#define STRING_DESCRIPTOR3_LEN (22u)
#define STRING_DESCRIPTOR4_LEN (22u)
#define STRING_DESCRIPTOR5_LEN (18u)
#define STRING_DESCRIPTOR6_LEN (28u)

/* Standard Device Descriptor */
uint8_t g_apl_device[USB_PCDC_DD_LEN + ( USB_PCDC_DD_LEN % 2)] =
{
    USB_PCDC_DD_LEN, /* 0:bLength */
    USB_DT_DEVICE, /* 1:bDescriptorType */
    (USB_BCDNUM & (uint8_t) 0xffu), /* 2:bcdUSB_lo */
    ((uint8_t) (USB_BCDNUM >> 8) & (uint8_t) 0xffu), /* 3:bcdUSB_hi */
    USB_IFCLS_CDCC, /* 4:bDeviceClass */
    0, /* 5:bDeviceSubClass */
    0, /* 6:bDeviceProtocol */
    (uint8_t) USB_DCPMAXP, /* 7:bMAXPacketSize(for DCP) */
    (USB_VENDORID & (uint8_t) 0xffu), /* 8:idVendor_lo */
    ((uint8_t) (USB_VENDORID >> 8) & (uint8_t) 0xffu), /* 9:idVendor_hi */
    ((uint16_t) USB_PRODUCTID & (uint8_t) 0xffu), /* 10:idProduct_lo */
    ((uint8_t) (USB_PRODUCTID >> 8) & (uint8_t) 0xffu), /* 11:idProduct_hi */
    (USB_RELEASE & (uint8_t) 0xffu), /* 12:bcdDevice_lo */
    ((uint8_t) (USB_RELEASE >> 8) & (uint8_t) 0xffu), /* 13:bcdDevice_hi */
    1, /* 14:iManufacturer */
    2, /* 15:iProduct */
    6, /* 16:iSerialNumber */
    USB_CONFIGNUM /* 17:bNumConfigurations */
};
    
```

```

/*****
 * Configuration or other Speed Configuration Descriptor
 *****/
/* For Full-Speed */
uint8_t g_apl_configuration[USB_PCDC_CD1_LEN + ( USB_PCDC_CD1_LEN % 2)] =
{
    9, /* 0:bLength */
    USB_SOFT_CHANGE, /* 1:bDescriptorType */
    USB_PCDC_CD1_LEN % 256, /* 2:wTotalLength(L) */
    USB_PCDC_CD1_LEN / 256, /* 3:wTotalLength(H) */
    2, /* 4:bNumInterfaces */
    1, /* 5:bConfigurationValue */
    0, /* 6:iConfiguration */
    USB_CF_RESERVED | USB_CF_SELFP, /* 7:bmAttributes */
    (10 / 2), /* 8:MAXPower (2mA unit) */

    /* Interface Descriptor */
    9, /* 0:bLength */
    USB_DT_INTERFACE, /* 1:bDescriptor */
    0, /* 2:bInterfaceNumber */
    0, /* 3:bAlternateSetting */
    1, /* 4:bNumEndpoints */
    USB_IFCLS_CDCC, /* 5:bInterfaceClass */
    USB_PCDC_CLASS_SUBCLASS_CODE_ABS_CTR_MDL, /* 6:bInterfaceSubClass */
    1, /* 7:bInterfaceProtocol */
    0, /* 8:iInterface */

    /* Communications Class Functional Descriptors */
    5, /* 0:bLength */
    USB_PCDC_CS_INTERFACE, /* 1:bDescriptorType */
    USB_PCDC_DT_SUBTYPE_HEADER_FUNC, /* 2:bDescriptorSubtype */
    USB_PCDC_BCD_CDC % 256, /* 3:bcdCDC_lo */
    USB_PCDC_BCD_CDC / 256, /* 4:bcdCDC_hi */

    /* Communications Class Functional Descriptors */
    4, /* 0:bLength */
    USB_PCDC_CS_INTERFACE, /* 1:bDescriptorType */
    USB_PCDC_DT_SUBTYPE_ABSTRACT_CTR_MANAGE_FUNC, /* 2:bDescriptorSubtype */
    2, /* 3:bmCapabilities */

    /* Communications Class Functional Descriptors */
    5, /* 0:bLength */
    USB_PCDC_CS_INTERFACE, /* 1:bDescriptorType */
    USB_PCDC_DT_SUBTYPE_UNION_FUNC, /* 2:bDescriptorSubtype */
    0, /* 3:bMasterInterface */
    1, /* 4:bSlaveInterface0 */

    /* Communications Class Functional Descriptors */
    5, /* 0:bLength */
    USB_PCDC_CS_INTERFACE, /* 1:bDescriptorType */
    USB_PCDC_DT_SUBTYPE_CALL_MANAGE_FUNC, /* 2:bDescriptorSubtype */
    /* D1:1-Device can send/receive call management
    information over a Data Class interface. */
    /* D0:1-Device handles call management itself. */
    3, /* 3:bmCapabilities */
    1, /* 4:bDataInterface */

    /* Endpoint Descriptor 0 */
    7, /* 0:bLength */
    USB_DT_ENDPOINT, /* 1:bDescriptorType */
    USB_EP_IN | USB_EP3, /* 2:bEndpointAddress */
    USB_EP_INT, /* 3:bmAttribute */
    16, /* 4:wMAXPacketSize_lo */
    0, /* 5:wMAXPacketSize_hi */
    0x10, /* 6:bInterval */

```

```

/* Interface Descriptor */
9, /* 0:bLength */
USB_DT_INTERFACE, /* 1:bDescriptor */
1, /* 2:bInterfaceNumber */
0, /* 3:bAlternateSetting */
2, /* 4:bNumEndpoints */
USB_IFCLS_CDCD, /* 5:bInterfaceClass */
0, /* 6:bInterfaceSubClass */
0, /* 7:bInterfaceProtocol */
0, /* 8:iInterface */

/* Endpoint Descriptor 0 */
7, /* 0:bLength */
USB_DT_ENDPOINT, /* 1:bDescriptorType */
USB_EP_IN | USB_EP1, /* 2:bEndpointAddress */
USB_EP_BULK, /* 3:bmAttribute */
64, /* 4:wMAXPacketSize_lo */
0, /* 5:wMAXPacketSize_hi */
0, /* 6:bInterval */

/* Endpoint Descriptor 1 */
7, /* 0:bLength */
USB_DT_ENDPOINT, /* 1:bDescriptorType */
USB_EP_OUT | USB_EP2, /* 2:bEndpointAddress */
USB_EP_BULK, /* 3:bmAttribute */
64, /* 4:wMAXPacketSize_lo */
0, /* 5:wMAXPacketSize_hi */
0, /* 6:bInterval */
};

/*****
 * String Descriptor
 *****/
/* UNICODE 0x0409 English (United States) */
uint8_t g_cdc_string_descriptor0[STRING_DESCRIPTOR0_LEN + (STRING_DESCRIPTOR0_LEN % 2)] =
{
    STRING_DESCRIPTOR0_LEN, /* 0:bLength */
    USB_DT_STRING, /* 1:bDescriptorType */
    0x09, 0x04 /* 2:wLANGID[0] */
};

/* iManufacturer */
uint8_t g_cdc_string_descriptor1[STRING_DESCRIPTOR1_LEN + (STRING_DESCRIPTOR1_LEN % 2)] =
{
    STRING_DESCRIPTOR1_LEN, /* 0:bLength */
    USB_DT_STRING, /* 1:bDescriptorType */
    'R', 0x00, /* 2:wLANGID[0] */
    'E', 0x00,
    'N', 0x00,
    'E', 0x00,
    'S', 0x00,
    'A', 0x00,
    'S', 0x00,
};

/* iProduct */
uint8_t g_cdc_string_descriptor2[STRING_DESCRIPTOR2_LEN + (STRING_DESCRIPTOR2_LEN % 2)] =
{
    STRING_DESCRIPTOR2_LEN, /* 0:bLength */
    USB_DT_STRING, /* 1:bDescriptorType */
    'U', 0x00,
    'S', 0x00,
    'B', 0x00,
};

```

```

/* iInterface */
uint8_t g_cdc_string_descriptor3[STRING_DESCRIPTOR3_LEN + (STRING_DESCRIPTOR3_LEN % 2)] =
{
    STRING_DESCRIPTOR3_LEN,          /* 0:bLength */
    USB_DT_STRING,                  /* 1:bDescriptorType */
    'C', 0x00,
    'o', 0x00,
    'm', 0x00,
    ' ', 0x00,
    'D', 0x00,
    'e', 0x00,
    'v', 0x00,
    'i', 0x00,
    'c', 0x00,
    'e', 0x00,
};

/* iConfiguration */
uint8_t g_cdc_string_descriptor4[STRING_DESCRIPTOR4_LEN + (STRING_DESCRIPTOR4_LEN % 2)] =
{
    STRING_DESCRIPTOR4_LEN,          /* 0:bLength */
    USB_DT_STRING,                  /* 1:bDescriptorType */
    'F', 0x00,                      /* 2:wLANGID[0] */
    'u', 0x00,
    'l', 0x00,
    'l', 0x00,
    '-', 0x00,
    'S', 0x00,
    'p', 0x00,
    'e', 0x00,
    'e', 0x00,
    'd', 0x00
};

uint8_t *g_apl_string_table[] =
{
    g_cdc_string_descriptor0,
    g_cdc_string_descriptor1,
    g_cdc_string_descriptor2,
    g_cdc_string_descriptor3,
    g_cdc_string_descriptor4,
};

/*****
Renesas Abstracted Peripheral Communications Devices Class Driver API functions
*****/

```

Your source file should look like this:

```

* r_usb_pcdc_descriptor.h
#ifdef R_USB_PCDC_DESCRIPTOR_H
#define R_USB_PCDC_DESCRIPTOR_H
#include "r_usb_basic_if.h"
/*****
Macro definitions
*****/
/* bcdUSB */
#define USB_BCDNUM                (0x0200u)/* 0x0100 (USB1.0), 0x0110 (USB1.1), 0x0200 (USB2.0)*/
/* Release Number */
#define USB_RELEASE                (0x0200u)/* 0x0100 (USB1.0), 0x0110 (USB1.1), 0x0200 (USB2.0)*/
/* DCP max packet size */
#define USB_DCPMAXP                (64u) /*Max packet size for endpoint 0.Must be 8, 16, 32 or 64*/
/* Configuration number */
#define USB_CONFIGNUM              (1u)/*Specifies the total number of possible configurations for the device.*/
/* Vendor ID */
#define USB_VENDORID              (0x0000u)/*must be obtained from USB-IF*/
/* Product ID*/
#define USB_PRODUCTID             (0x0002u)/*assigned by the manufacturer */

/* Class-Specific Configuration Descriptors */
#define USB_PCDC_CS_INTERFACE      (0x24u) /*assigned from USB-IF*/

/* bDescriptor SubType in Communications Class Functional Descriptors */
/* Header Functional Descriptor */
#define USB_PCDC_DT_SUBTYPE_HEADER_FUNC (0x00u)/*assigned from USB-IF*/
/* Call Management Functional Descriptor. */
#define USB_PCDC_DT_SUBTYPE_CALL_MANAGE_FUNC (0x01u)/*assigned from USB-IF*/
/* Abstract Control Management Functional Descriptor. */
#define USB_PCDC_DT_SUBTYPE_ABSTRACT_CTR_MANAGE_FUNC (0x02u)/*assigned from USB-IF*/
/* Union Functional Descriptor */
#define USB_PCDC_DT_SUBTYPE_UNION_FUNC (0x06u)/*assigned from USB-IF*/

/* Communications Class Subclass Codes */
#define USB_PCDC_CLASS_SUBCLASS_CODE_ABS_CTR_MDL (0x02u) /*SubClass Code assigned by USB-IF*/

/* USB Class Definitions for Communications Devices Specification
release number in binary-coded decimal. */
#define USB_PCDC_BCD_CDC          (0x0110u) /*assigned from USB-IF*/

/* Descriptor length */
#define USB_PCDC_DD_LEN           (18u)
#define USB_PCDC_QD_LEN           (10u)
#define USB_PCDC_CD1_LEN          (67u)
#define STRING_DESCRIPTOR0_LEN    (4u)
#define STRING_DESCRIPTOR1_LEN    (16u)
#define STRING_DESCRIPTOR2_LEN    (8u)
#define STRING_DESCRIPTOR3_LEN    (22u)
#define STRING_DESCRIPTOR4_LEN    (22u)
#define STRING_DESCRIPTOR5_LEN    (18u)
#define STRING_DESCRIPTOR6_LEN    (28u)

/* Standard Device Descriptor */
uint8_t g_apl_device[USB_PCDC_DD_LEN + ( USB_PCDC_DD_LEN % 2)] =
{
    USB_PCDC_DD_LEN,                /* 0:bLength */
    USB_DT_DEVICE,                 /* 1:bDescriptorType */
    (USB_BCDNUM & (uint8_t) 0xffu), /* 2:bcdUSB_lo */
    ((uint8_t) (USB_BCDNUM >> 8) & (uint8_t) 0xffu), /* 3:bcdUSB_hi */
    USB_IFCLS_CDCC,                /* 4:bDeviceClass */
    0,                             /* 5:bDeviceSubClass */
    0,                             /* 6:bDeviceProtocol */
    (uint8_t) USB_DCPMAXP,          /* 7:bMAXPacketSize(for DCP) */
    (USB_VENDORID & (uint8_t) 0xffu), /* 8:idVendor_lo */
    ((uint8_t) (USB_VENDORID >> 8) & (uint8_t) 0xffu), /* 9:idVendor_hi */
    ((uint16_t) USB_PRODUCTID & (uint8_t) 0xffu), /* 10:idProduct_lo */
    ((uint8_t) (USB_PRODUCTID >> 8) & (uint8_t) 0xffu), /* 11:idProduct_hi */
    (USB_RELEASE & (uint8_t) 0xffu), /* 12:bcdDevice_lo */
    ((uint8_t) (USB_RELEASE >> 8) & (uint8_t) 0xffu), /* 13:bcdDevice_hi */
    1,                             /* 14:iManufacturer */
    2,                             /* 15:iProduct */
    6,                             /* 16:iSerialNumber */
    USB_CONFIGNUM                  /* 17:bNumConfigurations */
};

```

```

* Configuration Or Other_Speed_Configuration Descriptor *[]
/* For Full-Speed */
uint8_t g_apl_configuration[USB_PCDC_CD1_LEN + ( USB_PCDC_CD1_LEN % 2)] =
{
    9, /* 0:bLength */
    USB_SOFT_CHANGE, /* 1:bDescriptorType */
    USB_PCDC_CD1_LEN % 256, /* 2:wTotalLength(L) */
    USB_PCDC_CD1_LEN / 256, /* 3:wTotalLength(H) */
    2, /* 4:bNumInterfaces */
    1, /* 5:bConfigurationValue */
    0, /* 6:iConfiguration */
    USB_CF_RESERVED | USB_CF_SELFP, /* 7:bmAttributes */
    (10 / 2), /* 8:MAXPower (2mA unit) */

    /* Interface Descriptor */
    9, /* 0:bLength */
    USB_DT_INTERFACE, /* 1:bDescriptorType */
    0, /* 2:bInterfaceNumber */
    0, /* 3:bAlternateSetting */
    1, /* 4:bNumEndpoints */
    USB_IFCLS_CDC, /* 5:bInterfaceClass */
    USB_PCDC_CLASS_SUBCLASS_CODE_ABS_CTR_MDL, /* 6:bInterfaceSubClass */
    1, /* 7:bInterfaceProtocol */
    0, /* 8:iInterface */

    /* Communications Class Functional Descriptors */
    5, /* 0:bLength */
    USB_PCDC_CS_INTERFACE, /* 1:bDescriptorType */
    USB_PCDC_DT_SUBTYPE_HEADER_FUNC, /* 2:bDescriptorSubtype */
    USB_PCDC_BCD_CDC % 256, /* 3:bcdCDC_lo */
    USB_PCDC_BCD_CDC / 256, /* 4:bcdCDC_hi */

    /* Communications Class Functional Descriptors */
    4, /* 0:bLength */
    USB_PCDC_CS_INTERFACE, /* 1:bDescriptorType */
    USB_PCDC_DT_SUBTYPE_ABSTRACT_CTR_MANAGE_FUNC, /* 2:bDescriptorSubtype */
    2, /* 3:bmCapabilities */

    /* Communications Class Functional Descriptors */
    5, /* 0:bLength */
    USB_PCDC_CS_INTERFACE, /* 1:bDescriptorType */
    USB_PCDC_DT_SUBTYPE_UNION_FUNC, /* 2:bDescriptorSubtype */
    0, /* 3:bMasterInterface */
    1, /* 4:bSlaveInterface0 */

    /* Communications Class Functional Descriptors */
    5, /* 0:bLength */
    USB_PCDC_CS_INTERFACE, /* 1:bDescriptorType */
    USB_PCDC_DT_SUBTYPE_CALL_MANAGE_FUNC, /* 2:bDescriptorSubtype */
    /* D1:1-Device can send/receive call management
    information over a Data Class interface. */
    /* D0:1-Device handles call management itself. */
    3, /* 3:bmCapabilities */
    1, /* 4:bDataInterface */

    /* Endpoint Descriptor 0 */
    7, /* 0:bLength */
    USB_DT_ENDPOINT, /* 1:bDescriptorType */
    USB_EP_IN | USB_EP3, /* 2:bEndpointAddress */
    USB_EP_INT, /* 3:bmAttribute */
    16, /* 4:wMAXPacketSize_lo */
    0, /* 5:wMAXPacketSize_hi */
    0x10, /* 6:bInterval */

    /* Interface Descriptor */
    9, /* 0:bLength */
    USB_DT_INTERFACE, /* 1:bDescriptorType */
    1, /* 2:bInterfaceNumber */
    0, /* 3:bAlternateSetting */
    2, /* 4:bNumEndpoints */
    USB_IFCLS_CDC, /* 5:bInterfaceClass */
    0, /* 6:bInterfaceSubClass */
    0, /* 7:bInterfaceProtocol */
    0, /* 8:iInterface */

    /* Endpoint Descriptor 0 */
    7, /* 0:bLength */
    USB_DT_ENDPOINT, /* 1:bDescriptorType */
    USB_EP_IN | USB_EP1, /* 2:bEndpointAddress */
    USB_EP_BULK, /* 3:bmAttribute */
    64, /* 4:wMAXPacketSize_lo */
    0, /* 5:wMAXPacketSize_hi */
    0, /* 6:bInterval */

```



```

        /* Endpoint Descriptor 1 */
        7, /* 0:bLength */
        USB_DT_ENDPOINT, /* 1:bDescriptorType */
        USB_EP_OUT | USB_EP2, /* 2:bEndpointAddress */
        USB_EP_BULK, /* 3:bmAttribute */
        64, /* 4:wMAXPacketSize_lo */
        0, /* 5:wMAXPacketSize_hi */
        0, /* 6:bInterval */
    };
    /*****
    * String Descriptor
    *****/
    /* UNICODE 0x0409 English (United States) */
    uint8_t g_cdc_string_descriptor0[STRING_DESCRIPTOR0_LEN + ( STRING_DESCRIPTOR0_LEN % 2)] =
    {
        STRING_DESCRIPTOR0_LEN, /* 0:bLength */
        USB_DT_STRING, /* 1:bDescriptorType */
        0x09, 0x04 /* 2:wLANGID[0] */
    };
    /* iManufacturer */
    uint8_t g_cdc_string_descriptor1[STRING_DESCRIPTOR1_LEN + ( STRING_DESCRIPTOR1_LEN % 2)] =
    {
        STRING_DESCRIPTOR1_LEN, /* 0:bLength */
        USB_DT_STRING, /* 1:bDescriptorType */
        'R', 0x00, /* 2:wLANGID[0] */
        'E', 0x00,
        'N', 0x00,
        'E', 0x00,
        'S', 0x00,
        'A', 0x00,
        'S', 0x00,
    };
    /* iProduct */
    uint8_t g_cdc_string_descriptor2[STRING_DESCRIPTOR2_LEN + ( STRING_DESCRIPTOR2_LEN % 2)] =
    {
        STRING_DESCRIPTOR2_LEN, /* 0:bLength */
        USB_DT_STRING, /* 1:bDescriptorType */
        'U', 0x00,
        'S', 0x00,
        'B', 0x00,
    };
    /* iInterface */
    uint8_t g_cdc_string_descriptor3[STRING_DESCRIPTOR3_LEN + ( STRING_DESCRIPTOR3_LEN % 2)] =
    {
        STRING_DESCRIPTOR3_LEN, /* 0:bLength */
        USB_DT_STRING, /* 1:bDescriptorType */
        'C', 0x00,
        'o', 0x00,
        'm', 0x00,
        'i', 0x00,
        'D', 0x00,
        'e', 0x00,
        'v', 0x00,
        'i', 0x00,
        'c', 0x00,
        'e', 0x00,
    };
    /* iConfiguration */
    uint8_t g_cdc_string_descriptor4[STRING_DESCRIPTOR4_LEN + ( STRING_DESCRIPTOR4_LEN % 2)] =
    {
        STRING_DESCRIPTOR4_LEN, /* 0:bLength */
        USB_DT_STRING, /* 1:bDescriptorType */
        'F', 0x00, /* 2:wLANGID[0] */
        'u', 0x00,
        'l', 0x00,
        'l', 0x00,
        '-', 0x00,
        'S', 0x00,
        'p', 0x00,
        'e', 0x00,
        'e', 0x00,
        'd', 0x00
    };
    uint8_t *g_apl_string_table[] =
    {
        g_cdc_string_descriptor0,
        g_cdc_string_descriptor1,
        g_cdc_string_descriptor2,
        g_cdc_string_descriptor3,
        g_cdc_string_descriptor4,
    };
    /*****
    Renesas Abstracted Peripheral Communications Devices Class Driver API functions
    *****/
#endif /* R_USB_PCDC_DESCRIPTOR_H */

```

Figure 6-14 r_usb_pcdc_descriptor.h

6.5 Adding application codes in *main()*

- 1) In `Smart_Configurator_Example.c`, add header files includes and declarations near the top of the file

```
#include "r_usb_basic_if.h"
#include "r_usb_pcdc_if.h"
#include "r_usb_pcdc_descriptor.h"
#include <stdio.h>
#include <string.h>

void R_USB_PinSet_USB0_PERI();           /* Initialize USB0_VBUS pin */
static uint8_t g_buf[1];                 /* Variable to store input
character from PC terminal */
volatile unsigned int flag_start = 1;    /* Flag to print start message
*/
volatile uint16_t interval_level = 1;    /* Variable to change CMT0
interval */
static char print_str[120];              /* String to print message at
PC terminal */
volatile uint32_t slength;               /* String length */

extern uint8_t g_apl_device[];
extern uint8_t g_apl_configuration[];
extern uint8_t *g_apl_string_table[];
const static usb_descriptor_t usb_descriptor =
{
    g_apl_device,                        /* Pointer to the device descriptor
*/
    g_apl_configuration,                 /* Pointer to the configuration
descriptor for Full-speed */
    USB_NULL,                             /* Pointer to the configuration
descriptor for Hi-speed */
    USB_NULL,                             /* Pointer to the qualifier
descriptor */
    g_apl_string_table                   /* Pointer to the string descriptor
table */
};
```

- 2) Add below codes into *main()* function before `R_Config_CMT0_Start()`

```
usb_ctrl_t  ctrl;
usb_cfg_t   cfg;
```

- 3) Add below codes in the same function after `R_Config_CMT0_Start()` to initialize USB0 and start the communication with terminal program on the PC host:

```

R_USB_PinSet_USB0_PERI();    /* USB MCU pin setting */

ctrl.module    = USB_IP0;    /* USB0 module */
ctrl.type      = USB_PCDC;   /* Peripheral Communication Device Class*/
cfg.usb_speed  = USB_FS;     /* USB_HS/USB_FS */
cfg.usb_mode   = USB_PERI;
cfg.p_usb_reg  = (usb_descriptor_t *)&usb_descriptor;
R_USB_Open(&ctrl, &cfg);     /* Initializes the USB module */

/* Loop back between PC Terminal and USB MCU */
while (1)
{
    switch (R_USB_GetEvent(&ctrl))
    {
        case USB_STS_CONFIGURED :
            break;

        case USB_STS_WRITE_COMPLETE :
            /* Read the input from PC terminal*/
            R_USB_Read(&ctrl, g_buf, 1);
            break;

        case USB_STS_READ_COMPLETE :
            /* Clear start message flag*/
            flag_start = 0;
            /* Instruction to slow down blinking rate is received */
            if (('a' == *g_buf) || ('A' == *g_buf))
            {
                /* Get new blink interval level. Maximum level is 9. */
                if (interval_level < 10)
                { /* Notify the character received and inform next action */
                    printf(print_str, "\r\n Character 'a' or 'A' is received. LED2 will blink
at slower rate.\r\n");
                    interval_level++;
                }
                else
                {
                    printf(print_str, "\r\n This is minimum. Please press 'b' or 'B' to blink
faster. \r\n");
                }
            }
            else if (('b' == *g_buf) || ('B' == *g_buf))
            {
                /* Get new blink interval level. Minimum level is 1 */
                if (interval_level > 1)
                {
                    /* Instruction to increase blinking rate is received */
                    printf(print_str, "\r\n Character 'b' or 'B' is received. LED2 will blink
at faster rate.\r\n");
                    interval_level--;
                }
                else
                {
                    printf(print_str, "\r\n This is maximum. Please press 'a' or 'A' to blink
slower. \r\n");
                }
            }
            else
            {
                printf(print_str, "\r\n Press keyboard to control the blinking rate of LED2 (not
case sensitive)\n\r\n a ----> Slower\n\r\n b ----> Faster\n\r\n");
            }
            /* Print message at PC terminal */
            slength = strlen(print_str);
            R_USB_Write(&ctrl, (uint8_t *)print_str, slength);

            /* Change blinking rate */
            CMT0.CMCOR = (uint16_t)(5000 * interval_level);
            break;

        default :
            break;
    }

    if (flag_start == 1)
    {
        /* Print start message until the 1st input from PC host is received */
        printf(print_str, "Press space bar twice to start ... \r ");
        slength = strlen(print_str);
        R_USB_Write(&ctrl, (uint8_t *)print_str, slength);
    }
}
    
```

Your source file should look like this:

```
#include "r_smc_entry.h"

#include "r_usb_basic_if.h"
#include "r_usb_pcdc_if.h"
#include "r_usb_pcdc_descriptor.h"
#include <stdio.h>
#include <string.h>

void R_USB_PinSet_USB0_PERI(); /* Initialize USB0_VBUS pin */
static uint8_t g_buf[1]; /* Variable to store input character from PC terminal */
volatile unsigned int flag_start = 1; /* Flag to print start message */
volatile uint16_t interval_level = 1; /* Variable to change CMT0 interval */
static char print_str[120]; /* String to print message at PC terminal */
volatile uint32_t slength; /* String length */

extern uint8_t g_apl_device[];
extern uint8_t g_apl_configuration[];
extern uint8_t *g_apl_string_table[];
const static usb_descriptor_t usb_descriptor =
{
    g_apl_device, /* Pointer to the device descriptor */
    g_apl_configuration, /* Pointer to the configuration descriptor for Full-speed */
    USB_NULL, /* Pointer to the configuration descriptor for Hi-speed */
    USB_NULL, /* Pointer to the qualifier descriptor */
    g_apl_string_table /* Pointer to the string descriptor table */
};

void main(void)
{
    usb_ctrl_t ctrl;
    usb_cfg_t cfg;

    /* Start CMT0 counter operation */
    R_Config_CMT0_Start();

    usb_pin_setting(); /* USB MCU pin setting */

    ctrl.module = USB_IP0; /* USB0 module */
    ctrl.type = USB_PCDC; /* Peripheral Communication Device Class*/
    cfg.usb_speed = USB_FS; /* USB_HS/USB_FS */
    cfg.usb_mode = USB_PERI;
    cfg.p_usb_reg = (usb_descriptor_t *)&usb_descriptor;
    R_USB_PinSet_USB0_PERI(); /* Initializes the USB module */

    /* Loop back between PC Terminal and USB MCU */
    while (1)
    {
        switch (R_USB_GetEvent(&ctrl))
        {
            case USB_STS_CONFIGURED :
                break;

            case USB_STS_WRITE_COMPLETE :
                /* Read the input from PC terminal*/
                R_USB_Read(&ctrl, g_buf, 1);
                break;

            case USB_STS_READ_COMPLETE :
                /* Clear start message flag*/
                flag_start = 0;
                /* Instruction to slow down blinking rate is received */
                if (('a' == *g_buf) || ('A' == *g_buf))
                {
                    /* Get new blink interval level. Maximum level is 9. */
                    if (interval_level < 10)
                    {
                        /* Notify the character received and inform next action */
                        sprintf(print_str, "\r\n Character 'a' or 'A' is received. LED2 will blink at slower rate.\r\n");
                        interval_level++;
                    }
                }
                else
                {
                    sprintf(print_str, "\r\n This is minimum. Please press 'b' or 'B' to blink faster. \r\n");
                }
            }
        }
    }
}
```

```

else if (('b' == *g_buf)|| ('B' == *g_buf))
{
    /* Get new blink interval level. Minimum level is 1 */
    if (interval_level > 1)
    {
        /* Instruction to increase blinking rate is received */
        sprintf(print_str, "\r\n Character 'b' or 'B' is received. LED2 will blink at faster rate.\r\n");
        interval_level--;
    }
    else
    {
        sprintf(print_str, "\r\n This is maximum. Please press 'a' or 'A' to blink slower. \r\n");
    }
}
else
{
    sprintf(print_str, "\r\n Press keyboard to control the blinking rate of LED2 (not case sensitive)\n\r\n a ----> Slower\n\r\n b ----> Faster\n\r\n");
}
/* Print message at PC terminal */
length = strlen(print_str);
R_USB_Write(&ctrl, (uint8_t *)print_str, length);

/* Change blinking rate */
CMT0.CMCOR = (uint16_t)(5000 * interval_level);
break;


default :
break;
}

if (flag_start == 1)
{
    /* Print start message until the 1st input from PC host is received */
    sprintf(print_str, "Press space bar twice to start ...\r ");
    length = strlen(print_str);
    R_USB_Write(&ctrl, (uint8_t *)print_str, length);
}
}
}


```

Figure 6-15 USB PCDC application codes in *main* function

6.6 Build and run on hardware board

After build the project, click debug  icon to debug the project.

Step1. Use a USB to mini B cable to connect PC host and USB function connector on Renesas Starter Kit+ for RX65N 2MB (refer to Chapter 5.8 for the USB function port location)

Step 2. At e² studio Debug perspective, click  to execute the project, the project status will show in Running status **Running** at the left bottom of IDE.

Step 3. Open a terminal program in PC (for e.g. Tera Term). Perform the following steps:

- Select Serial radio button
- In the Port option, select COM that is connected to *USB Serial Device*
- Click [OK]

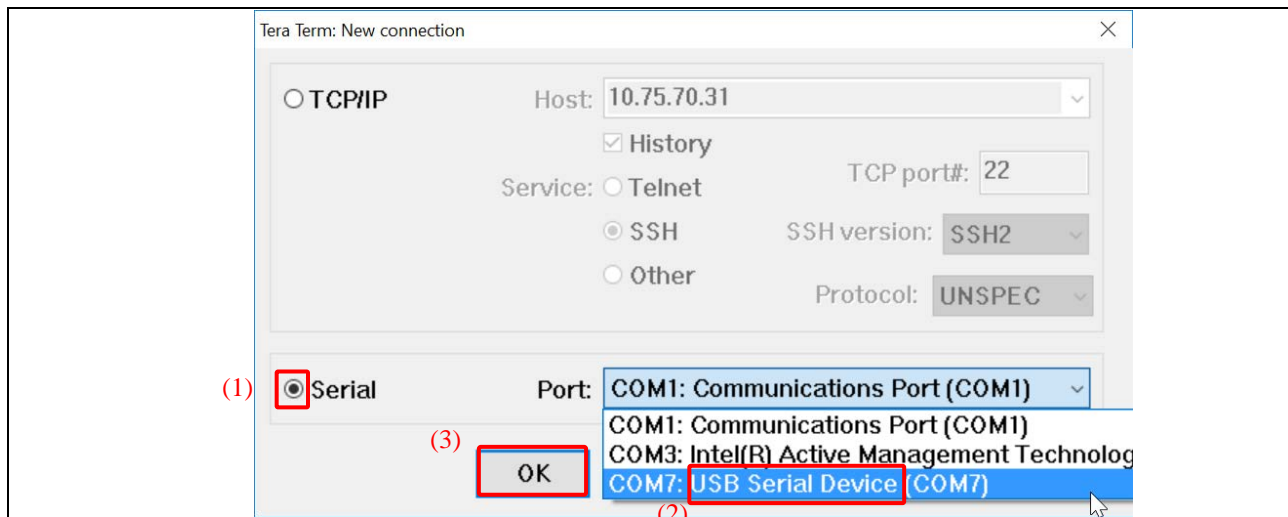


Figure 6-16 Select USB Serial Device

Note:

For Windows 7 PC, device driver software may not be installed successfully.

Install the system definition file separately from
 \src\smc_gen\r_usb_pcdc\utilities\CDC_Demo_Win7.inf

After installation, terminal program will display COM as “CDC USB Demonstration”.

Step 4. Follow the message on the terminal program to increase or decrease LED2 blinking rate:

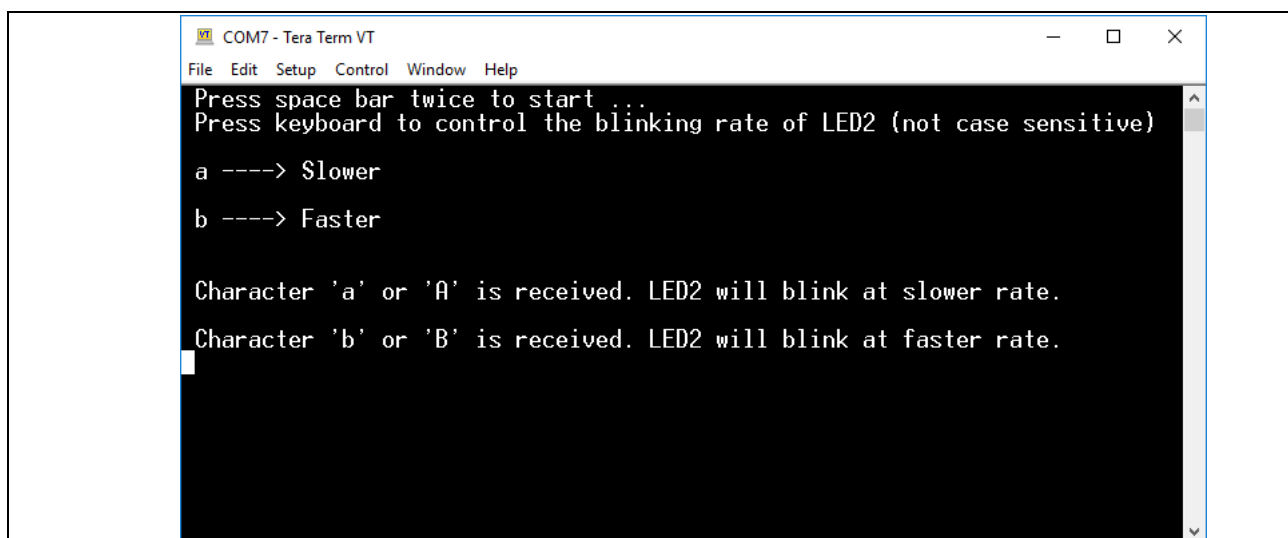


Figure 6-17 Terminal window in PC

6.7 Operation on board

On the Renesas Starter Kit+ for RX65N 2MB board:

Blinking rate of LED2 is changed when key in character 'a' or 'b' at the PC terminal:

- a : Blinking of LED2 will slow down
- b : Blinking of LED2 will become faster



Figure 6-18 LED2 and USB Function Port on RX65N 2MB board

6.8 Additional debugging assistance tool (QE)

Renesas has a range of Quick and Effective Tool Solutions (QEs) as development tools for particular applications to assist and improve efficiency during development. Specific to this example of integrating USB function, QE for USB is recommended as it has several features to assist in debugging application based on USB function.

Please refer to the following link for more information on QE and the type of supported applications for Renesas IDE:

QE: <https://www.renesas.com/qe>

QE for USB: <https://www.renesas.com/qe-usb>

Website and Support

- Renesas Electronics Website
<http://www.renesas.com/>
- Inquiries
<http://www.renesas.com/contact/>

All trademarks and registered trademarks are the property of their respective owners.

Revision History

Rev.	Date	Description			
		Page	Summary		
1.00	July 31, 2017	-	First creation		
1.10	Mar 30,2018	1	Document title, changed		
		10	Updated based on e ² studio V6.2 Figure 2-9 C Project - Target Specific Settings example with E2 Lite emulator, changed		
		13	Updated based on e ² studio V6.2 Figure 2-13 Select software component, changed		
		15	Updated based on e ² studio V6.2 Figure 2-16 Select software component, changed		
		29	Updated based on e ² studio V6.2 Figure 3-4 Select software component, changed		
		40	Updated based on e ² studio V6.2 Figure 4-8 Select software component, changed		
		61	Updated based on e ² studio V6.2 Figure 6-9 Select software component, changed		
		62	Updated based on e ² studio V6.2 Figure 6-10 configure software component, added		
		63	Updated based on e ² studio V6.2 Figure 6-11 configure software component, added		
		65	5.4 Modifying the default configuration, deleted		
		74	Updated based on e ² studio V6.2 Revise code to adjust to new smart configurator		
		1.20	Jun 20,2019	36	Updated Application example 3 to use SCI8
				54	Added Application 4 – Transferring data using DMA

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

2. Processing at power-on

The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.

3. Input of signal during power-off state

Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.

4. Handling of unused pins

Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.

5. Clock signals

After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.

6. Voltage application waveform at input pin

Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).

7. Prohibition of access to reserved addresses

Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.

8. Differences between products

Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. 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 or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
 2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving 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, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
 3. 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 shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
 5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended 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; industrial robots; etc.
 - "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.
- Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or 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 damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
 7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, 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 and impractical, you are responsible for evaluating 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. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
 9. Renesas Electronics products and technologies shall 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 shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
 10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
 11. This document shall not be reprinted, 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.

(Note1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.