RELEASE NOTE

RZ/T1 Group

ENCOUT application package

R01AN3806EJ0100
Rev.1.00
August 4, 2017

Summary

This document explains about RZ/T1 ENCOUT application package.

To use this application package, please obtain release package of "RZ/T1 Encoder I/F Configuration Library".

Device that ENCOUT functionality is checked

RZ/T1 CPU Board (RTK7910018C00000BE)

Version History

Ver.	Date	Content	Note
1.0	August 2017	Updated RZ/T1 ENCOUT sample program to Ver.1.0. (1) Changed carrier period which can be specified as argument of "R_ENCOUT_Control(R_ENCOUT_CMD_INIT)" function up to 3276999 ns. (2) Updated return value of "R_ENCOUT_GetVersion" function to Ver.1.0. (3) Correction of mistakes etc.	
		Updated "RZ/T1 ENCOUT Configuration Data" to Ver.1.0. (1) Added a set of ABZ phase output terminals. (2) Updated value of VER register to Ver.1.0.	
		Updated "RZ/T1 Group Encoder Divided-Output Module (ENCOUT) User's Manual" to Rev.1.00.	
0.5	April 2017	Newly created	

Table of contents

1. Contents of package

Contents of this package are described in this chapter.

1.1 Software

・Source code

No.	Title	Version
1	A set of RZ/T1 ENCOUT sample program code	1.0

・Configuration data

No.	Title	Version
1	RZ/T1 ENCOUT Configuration Data	1.0

1.2 Document

No.	Document name	Ver.	File name
1	RZ/T1 Group ENCOUT application package release note	1.00	(English) r01an3806ej0100-rzt1.pdf (this document)
2	RZ/T1 Group Encoder Divided-Output Module (ENCOUT) User's Manual	1.00	(English) r01uh0701ej0100_rzt1_encout.pdf (Japanese) r01uh0701jj0100_rzt1_encout.pdf

2. File structures

File structures and contents of this package are described below.

```
Top
├─r01an3806ej0100-rzt1.pdf
└─workspace
    ├─Documentation
    │   ├─r01uh0701ej0100_rzt1_encout.pdf
    │   └─r01uh0701jj0100_rzt1_encout.pdf
    └─Software
        ├─armcc
        │   └─RZ_T1_encout.zip：A set of RZ/T1 ENCOUT sample program code (DS-5)
        ├─iccarm
        │   └─RZ_T1_encout.zip：A set of RZ/T1 ENCOUT sample program code (EWARM)
        └─kpitgcc
            └─RZ_T1_encout.zip：A set of RZ/T1 ENCOUT sample program code (e2 studio)
```

The file structures of "RZ_T1_encout.zip" are shown below.

```
Top folder
    inc
        iodefine.h                 RZ/T1 register definition file
        iodefine_encout.h          ENCOUT register definition file
        r_encout_rzt1_dat.h        Header file for r_encout_rzt1.dat
        r_encout_rzt1_if.h         ENCOUT driver header file
        Common header files including initial settings
    lib
        ecl
            r_encout_rzt1.dat      RZ/T1 ENCOUT Configuration Data
    src
        common
            Common sources including initial settings
        drv
            encout
                r_encout_rzt1_config.h     ENCOUT driver file
                r_encout_rzt1.c            ENCOUT driver file
            scifa_uart
                SCIFA driver file
        sample
            main.c                 Main program file
            encout_dat.s           Linker setting file for the configuration data *1
            nestintr_wraps.s       Sample program for initial settings (for DS-5 only)
            siorw.c                SCIFA sample program
            siochar.c              SCIFA sample program
            retarget.c             SCIFA sample program (for DS-5 only)
```

Note 1: file for DS-5 and e2 studio
DS-5: encout_dat.s
e2 studio: encout_dat.asm

3. Information about ENCOUT sample program

This chapter describes information to use ENCOUT sample program.

3.1 Operating environment

ENCOUT sample program is for the environment below.

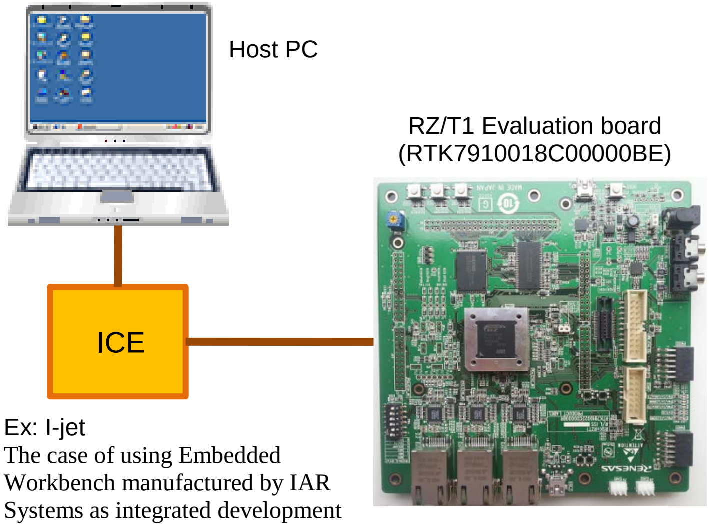
Item	Description
Microcomputer	RZ/T1 Group
Operating frequency	CPUCLK = 450MHz
Operating voltage	3.3V
Integrated development environment	Manufactured by IAR Systems Embedded Workbench® for ARM Version 7.80.2 (ICE: I-jet) Manufactured by ARM ARM Development Studio 5 (DS-5™) Version 5.25.0 ARM Compiler 5.06 update 3 (ICE: ULINK2) Manufactured by RENESAS RENESAS e2 studio 5.2.0.020 KPIT GNUARM-NONE-EABI Toolchain v16.01 (ICE: J-Link BASE)
Operating modes	SPI boot mode 16-bit bus boot mode
Board	RZ/T1 Evaluation board (RTK7910018C00000BE)
Devices (functions to be used on the board)	Serial interface (USB-Mini B connector J8) NOR flash memory (connected to CS0/CS1 space) Manufacturer: Macronix International Co. Ltd. Model: MX29GL512FLT2I-10Q SDRAM (connected to CS2/CS3 space) Manufacturer: Integrated Silicon Solution Inc. Model: IS42S16320D-7TL Serial flash memory Manufacturer: Macronix International Co. Ltd. Model: MX25L51245G
Operating system	This software is independent from operating system.

The peripheral functions used are below.

Peripheral function	Application
Encoder divided output (ENCOUT)	Output ABZ phase signal.
Compare match timer (CMT) Unit 0 channel 1	Generate the carrier period.
Event link controller (ELC)	Input the carrier period to ENCOUT.
Serial communication interface with FIFO (SCIFA)	Output debug information.

3.2 Target board

The connection of host PC and target board “RZ/T1 Evaluation board (RTK7910018C00000BE)” is as follows.



Ex: I-jet
The case of using Embedded Workbench manufactured by IAR Systems as integrated development environment.

Phase A output port: Pin 13 of connector JA2
Phase B output port: Pin 14 of connector JA2
Phase Z output port: Pin 15 of connector JA2

The settings of the target board are below.

SW4-1: ON

SW4-2: ON in case of serial flash memory is used, OFF in case of NOR flash memory is used

SW4-3: ON

SW4-4: ON

SW4-5: ON

SW4-6: OFF

JP2: 2-3 Connect

JP7: 1-2 Connect

3.3 Preparation before executing this sample program

This sample program communicates with the Host PC. Before executing the sample program, install the USB serial driver and set the terminal software.

Download “RZ/T1 Group FIFO Integrated Serial Communication Interface (SCIFA) Sample Program” from Renesas Electronics web site and install the bundled USB serial driver.

The setting of terminal software is as follows.

Baud rate: 115200kbps

Data: 8 bit

Parity: None

Stop bit: 1 bit

Flow control: None

3.4 Procedure on development environments

3.4.1 EWARM from IAR systems

- How to build sample program

1. Extract files from RZ_T1_encout.zip and copy the files to arbitrary holder
2. Copy the following files of "RZ/T1 Encoder I/F Configuration Library" (for IAR EWARM) to each folder

   lib\ecl\r_ecl_rzt1.a

   inc\r_ecl_rzt1_if.h
3. Launch EWARM
4. Select [File]menu -> [Open] -> [Workspace]
5. Open RZ_T1_encout_boot\RZ_T1_encout_****_boot.eww

NOR version	RZ_T1_encout_nor_boot.eww
Serial Flash version	RZ_T1_encout_serial_boot.eww

6. Select [Project]menu -> [Rebuild all]

   Following file is generated.

   RZ_T1_encout_boot\Debug\Exe\RZ_T1_encout_****_boot.out

NOR version	RZ_T1_encout_nor_boot.out
Serial Flash version	RZ_T1_encout_serial_boot.out

- How to execute sample program

  After executing "How to build sample program", connect the target board and the debugger properly, and execute the following operations.

1. Select [Project] menu-> [Download and Debug]
2. Select [Debug] menu-> [Go]

3.4.2 DS-5 from ARM

- How to build sample program

1. Extract files from RZ_T1_encout.zip and copy the files to arbitrary holder
2. Copy the following files of "RZ/T1 Encoder I/F Configuration Library" (for ARM DS-5) to each folder

   lib\ecl\r_ecl_rzt1.a

   inc\r_ecl_rzt1_if.h
3. Launch DS-5
4. Select [Window]menu -> [Show View] -> [Project Explorer]
5. Click right button on [Project Explorer]view and then select [Import] of popup menu
6. Select [General] -> [Existing Projects into Workspace] of [Import] dialog and then click [Next] button
7. Click [Browse…] of [Import] dialog
8. Select holder (the arbitrary holder of procedure 1 above) in [Browse For Folder] dialog and then click [OK].
9. Select [Copy projects into workspace] of [Import] dialog
10. Click [Finish] of [Import] dialog
11. Select [Project] menu -> [Build All]

    Following file is generated.

    Debug\RZ_T_nor_sample.axf

    (In case of serial flash, use the "RZ_T_sflash_sample.axf" instead of the "RZ_T_nor_sample.axf")

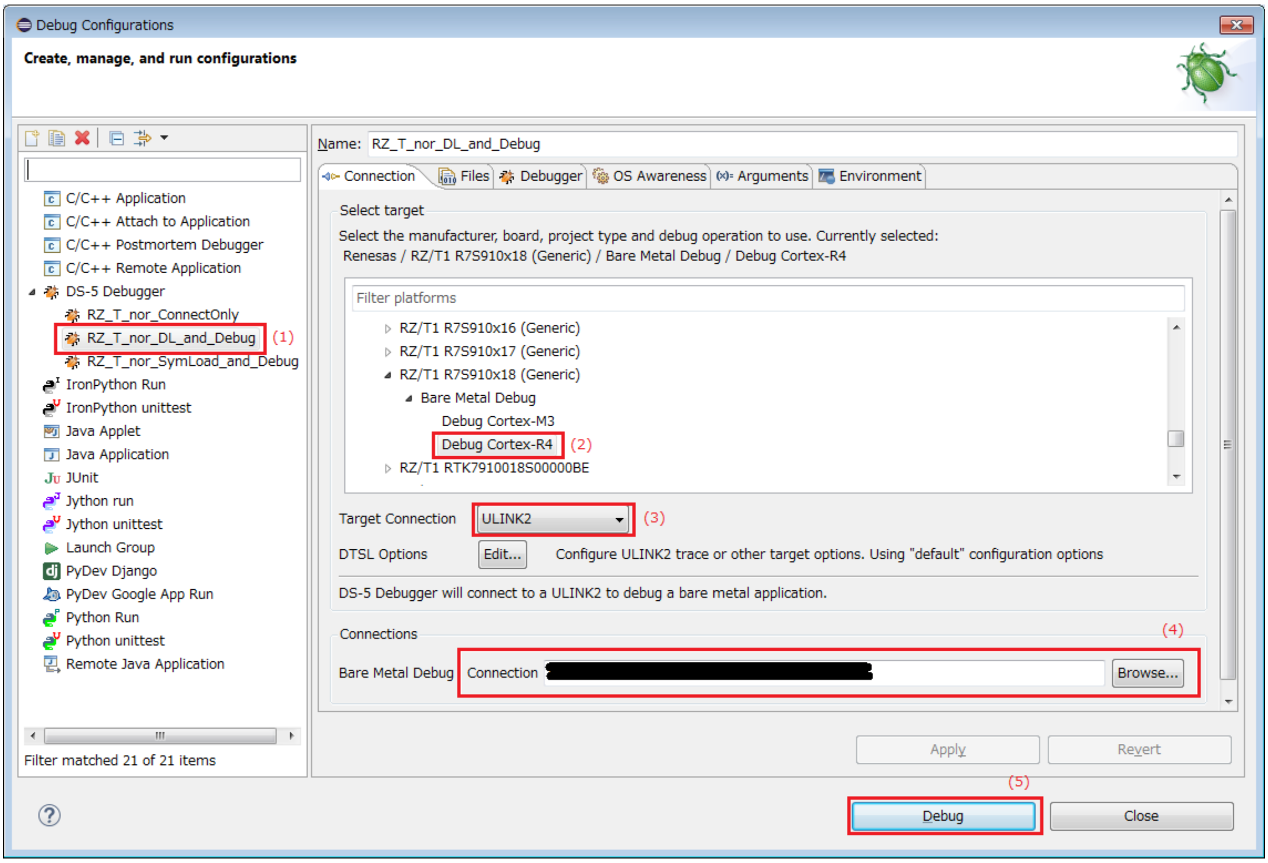
- How to execute sample program

After executing "How to build sample program", connect the target board and the debugger properly, and execute the following operations.

1. Open the debug configuration from the [Run] -> [Debug Configurations...], select the configuration window for "RZ_T_nor_DL_and_Debug". (In case of serial flash, use the "RZ_T_sflash_DL_and_Debug" instead of the "RZ_T_nor_DL_and_Debug")

   Select "Debug Cortex-R4" of "RZ/T1 R7S910x18 (Generic)" in [Select target].

   Select the ULINK2 of [Target Connection] in [Connection] tab, click on [Browse] and select the target connection from the list in the window. Click on [Debug] in the debug configurations window and start debugging.



Debug Configurations

Create, manage, and run configurations

- C/C++ Application
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- C/C++ Remote Application
- DS-5 Debugger
  - RZ_T_nor_ConnectOnly
  - RZ_T_nor_DL_and_Debug (1)
  - RZ_T_nor_SymLoad_and_Debug
- IronPython Run
- IronPython unittest
- Java Applet
- Java Application
- JUnit
- Jython run
- Jython unittest
- Launch Group
- PyDev Django
- PyDev Google App Run
- Python Run
- Python unittest
- Remote Java Application

Filter matched 21 of 21 items

Name: RZ_T_nor_DL_and_Debug

Select target

Select the manufacturer, board, project type and debug operation to use. Currently selected:
Renesas / RZ/T1 R7S910x18 (Generic) / Bare Metal Debug / Debug Cortex-R4

Filter platforms

- RZ/T1 R7S910x16 (Generic)
- RZ/T1 R7S910x17 (Generic)
- RZ/T1 R7S910x18 (Generic)
  - Bare Metal Debug
    - Debug Cortex-M3
    - Debug Cortex-R4 (2)
- RZ/T1 RTK7910018S00000BE

Target Connection: ULINK2 (3)

DTSL Options: Edit... Configure ULINK2 trace or other target options. Using "default" configuration options

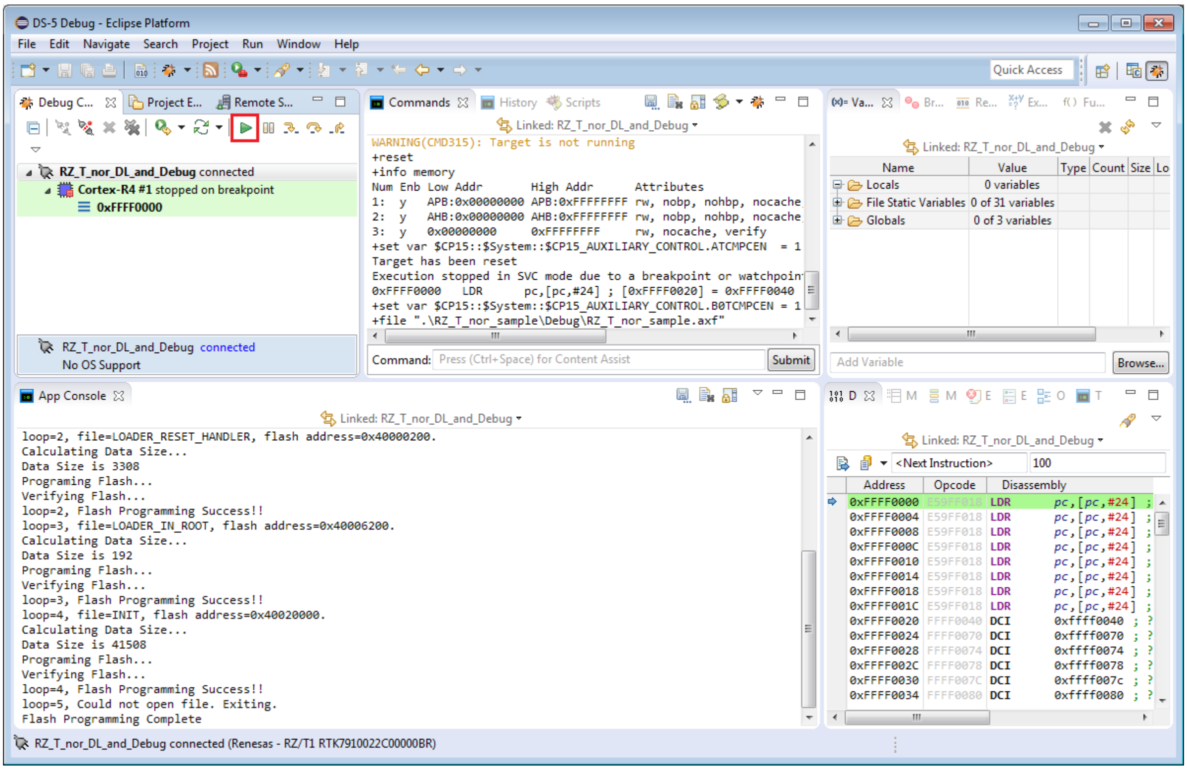
DS-5 Debugger will connect to a ULINK2 to debug a bare metal application.

Connections (4)

Bare Metal Debug Connection [redacted] Browse...

(5)

2. On completion of writing to the flash memory by the script, the message "Flash Programming Complete" appears in the application console window. Debugging can then start.


DS-5 Debug - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Quick Access
Debug C... Project E... Remote S...
Commands History Scripts
Va... Br... Re... Ex... Fu...
App Console
D M M E E O T
Submit
Browse...

RZ_T_nor_DL_and_Debug connected
Cortex-R4 #1 stopped on breakpoint
0xFFFF0000

RZ_T_nor_DL_and_Debug connected
No OS Support

Linked: RZ_T_nor_DL_and_Debug

```
WARNING(CMD315): Target is not running
+reset
+info memory
Num Enb Low Addr       High Addr      Attributes
1:  y   APB:0x00000000 APB:0xFFFFFFFF rw, nobp, nohbp, nocache
2:  y   AHB:0x00000000 AHB:0xFFFFFFFF rw, nobp, nohbp, nocache
3:  y   0x00000000     0xFFFFFFFF     rw, nocache, verify
+set var $CP15::$System::$CP15_AUXILIARY_CONTROL.ATCMPCEN  = 1
Target has been reset
Execution stopped in SVC mode due to a breakpoint or watchpoin
0xFFFF0000   LDR      pc,[pc,#24] ; [0xFFFF0020] = 0xFFFF0040
+set var $CP15::$System::$CP15_AUXILIARY_CONTROL.B0TCMPCEN = 1
+file ".\RZ_T_nor_sample\Debug\RZ_T_nor_sample.axf"
```

Command: Press (Ctrl+Space) for Content Assist

Linked: RZ_T_nor_DL_and_Debug

Name	Value	Type	Count	Size	Lo
Locals	0 variables				
File Static Variables	0 of 31 variables				
Globals	0 of 3 variables				

Add Variable

Linked: RZ_T_nor_DL_and_Debug

```
loop=2, file=LOADER_RESET_HANDLER, flash address=0x40000200.
Calculating Data Size...
Data Size is 3308
Programing Flash...
Verifying Flash...
loop=2, Flash Programming Success!!
loop=3, file=LOADER_IN_ROOT, flash address=0x40006200.
Calculating Data Size...
Data Size is 192
Programing Flash...
Verifying Flash...
loop=3, Flash Programming Success!!
loop=4, file=INIT, flash address=0x40020000.
Calculating Data Size...
Data Size is 41508
Programing Flash...
Verifying Flash...
loop=4, Flash Programming Success!!
loop=5, Could not open file. Exiting.
Flash Programming Complete
```

Linked: RZ_T_nor_DL_and_Debug

<Next Instruction> 100

Address	Opcode	Disassembly
0xFFFF0000	E59FF018	LDR pc,[pc,#24] ;
0xFFFF0004	E59FF018	LDR pc,[pc,#24] ;
0xFFFF0008	E59FF018	LDR pc,[pc,#24] ;
0xFFFF000C	E59FF018	LDR pc,[pc,#24] ;
0xFFFF0010	E59FF018	LDR pc,[pc,#24] ;
0xFFFF0014	E59FF018	LDR pc,[pc,#24] ;
0xFFFF0018	E59FF018	LDR pc,[pc,#24] ;
0xFFFF001C	E59FF018	LDR pc,[pc,#24] ;
0xFFFF0020	FFFF0040	DCI 0xffff0040 ; ?
0xFFFF0024	FFFF0070	DCI 0xffff0070 ; ?
0xFFFF0028	FFFF0074	DCI 0xffff0074 ; ?
0xFFFF002C	FFFF0078	DCI 0xffff0078 ; ?
0xFFFF0030	FFFF007C	DCI 0xffff007c ; ?
0xFFFF0034	FFFF0080	DCI 0xffff0080 ; ?

RZ_T_nor_DL_and_Debug connected (Renesas - RZ/T1 RTK7910022C00000BR)

3.4.3 e2 studio from RENESAS

- How to build sample program

1. Extract files from RZ_T1_encout.zip and copy the files to arbitrary holder
2. Copy the following files of "RZ/T1 Encoder I/F Configuration Library" (for KPIT GCC) to each folder

   lib\ecl\r_ecl_rzt1.a

   inc\r_ecl_rzt1_if.h
3. Launch the e2studio
4. Select [Window]menu -> [Show View] -> [Project Explorer]
5. Click right button on [Project Explorer]view and then select [Import] of popup menu
6. Select [General] -> [Existing Projects into Workspace] of [Import] dialog and then click [Next] button
7. Click [Browse...] of [Import] dialog
8. Select holder (the arbitrary holder of procedure 1 above) in [Browse For Folder] dialog and then click [OK].
9. Select [Copy projects into workspace] of [Import] dialog
10. Click [Finish] of [Import] dialog
11. Select [Project] menu -> [Build All]

    Following file is generated.

    HardwareDebug\ RZ_T_nor_sample.x

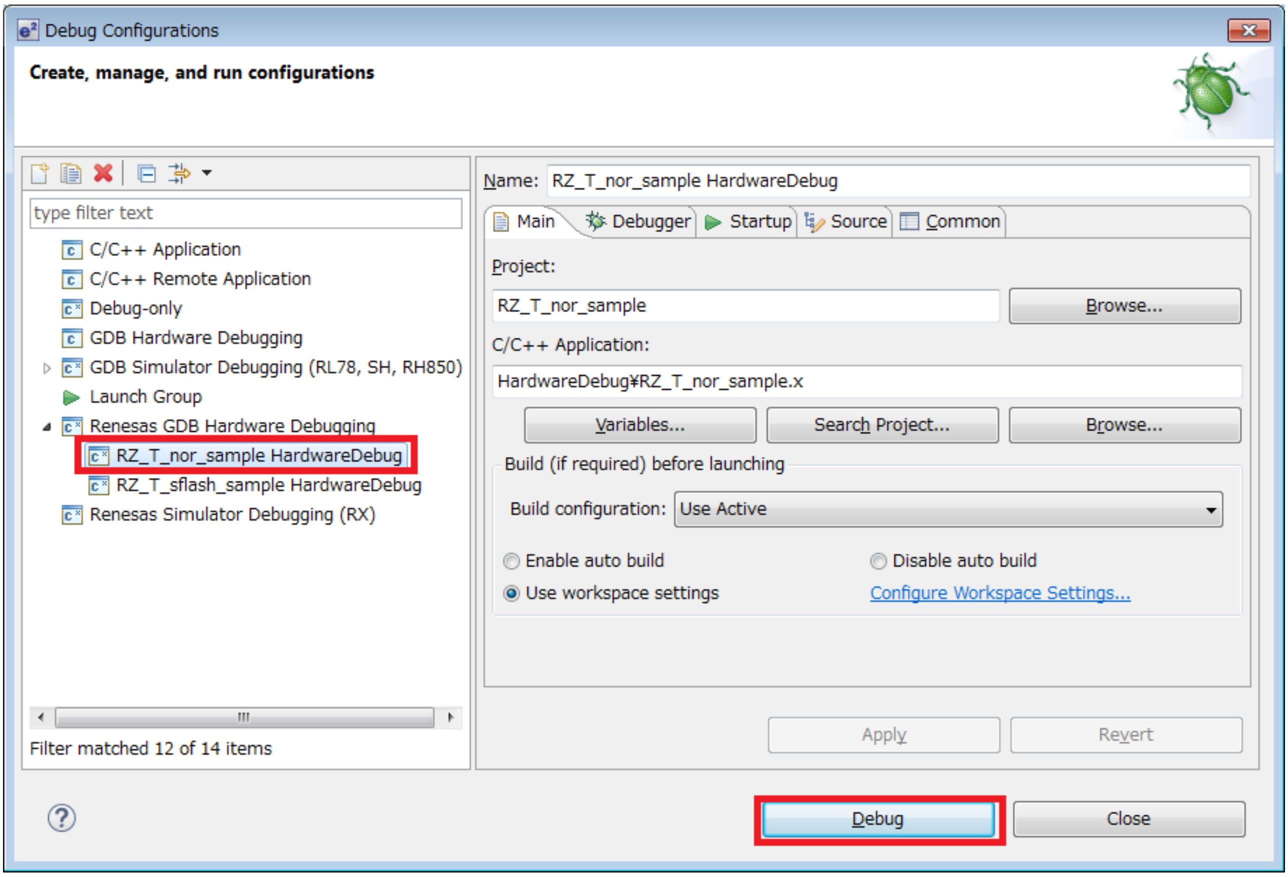
    (In case of serial flash, use the "RZ_T_sflash_sample.x" instead of the "RZ_T_nor_sample.x")

- How to execute sample program

  After executing "How to build sample program", connect the target board and the debugger properly, and execute the following operations.

1. Select [Run] from the [Project] menu and then select [Debug Configurations].
2. Select the [RZ_T_nor_sample_HardwareDebug] in the following screen. Click the [Debug] and start the download to flash memory.

   (In case of serial flash, use the [RZ_T_sflash_sample_HardwareDebug] instead of the [RZ_T_nor_sample_HardwareDebug])



Debug Configurations

Create, manage, and run configurations

type filter text

- C/C++ Application
- C/C++ Remote Application
- Debug-only
- GDB Hardware Debugging
- GDB Simulator Debugging (RL78, SH, RH850)
- Launch Group
- Renesas GDB Hardware Debugging
  - RZ_T_nor_sample HardwareDebug
  - RZ_T_sflash_sample HardwareDebug
- Renesas Simulator Debugging (RX)

Filter matched 12 of 14 items

Name: RZ_T_nor_sample HardwareDebug

Project: RZ_T_nor_sample

C/C++ Application: HardwareDebug¥RZ_T_nor_sample.x

Build (if required) before launching

Build configuration: Use Active

○ Enable auto build ○ Disable auto build

● Use workspace settings Configure Workspace Settings...

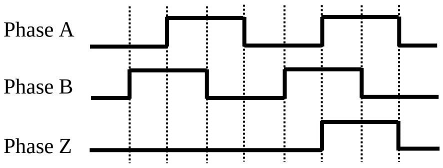
3. Click the [Resume] from the [Run] to start execution of the sample program.

3.4.4 Result of execution

When you run this sample program, it will appear on the terminal software as follows.

```
ENCOUT sample program start
  EC-Lib Ver.1.2
  ENCOUT Ver.1.0
  ENCOUT driver Ver.1.0
```

In addition, the signals as follows are output to the port described “3.2 Target board”.



3.5 Specification

3.5.1 Memory footprint

The estimates of memory footprint are below.

Items			Size [bytes]		
			EWARM	DS-5	e2 studio
ENCOUT driver	Code		700	1428	2052
	Data (with initial value)		0	0	0
	Data (without initial value)		18	18	18
	Constant data		48	48	48
	Stack size of functions	R_ENCOUT_Open	8	16	48
		R_ENCOUT_Close	16	24	48
		R_ENCOUT_Control	36	96	104
		R_ENCOUT_GetVersion	0	0	4
RZ/T1 ENCOUT Configuration Data	Constant data		8276		
Main program	Code		1084	1752	1836
	Data (with initial value)		20	0	20
	Data (without initial value)		4	24	4
	Constant data		328	0	328

3.5.2 API of ENCOUT driver

(1) R_ENCOUT_Open

R_ENCOUT_Open		
Synopsis	Initialization of ENCOUT driver	
Header	r_encout_rzt1_if.h r_encout_rzt1_dat.h	
Declaration	r_encout_err_t R_ENCOUT_Open(const int32_t id);	
Description	This function initializes ENCOUT driver. Before using ENCOUT driver, be sure to call this function.	
Arguments	id : Specify R_ENCOUT0_ID	
Return value	R_ENCOUT_SUCCESS	: Normal termination
	R_ENCOUT_ERR_INVALID_ARG	: Abnormal termination (a value of argument “id” is an undefined value)
	R_ENCOUT_ERR_ACCESS	: Abnormal termination (ENCOUT driver is already initialized)

(2) R_ENCOUT_Close

R_ENCOUT_Close		
Synopsis	Termination of ENCOUT driver	
Header	r_encout_rzt1_if.h r_encout_rzt1_dat.h	
Declaration	r_encout_err_t R_ENCOUT_Close(const int32_t id);	
Description	This function terminates ENCOUT driver. If this function is called while ENCOUT is running, it will terminate after stopping ENCOUT.	
Arguments	id : Specify R_ECNOUT0_ID	
Return value	R_ENCOUT_SUCCESS	: Normal termination
	R_ENCOUT_ERR_INVALID_ARG	: Abnormal termination (a value of argument “id” is an undefined value)

(3) R_ENCOUT_GetVersion

R_ENCOUT_GetVersion		
Synopsis	Acquiring the version number of ENCOUT driver	
Header	r_encout_rzt1_if.h	
Declaration	uint32_t R_ENCOUT_GetVersion(void);	
Description	This function acquires the version number of ENCOUT driver.	
Arguments	None	
Return value	Version information	: The major part of the version number is stored in the sixteen higher-order bits and the minor part of the version number is stored in the sixteen lower-order bits. Ex.) 0x00010002 is returned for Ver.1.2.

(4) R_ENCOUT_Control

R_ENCOUT_Control	
Synopsis	Operation of ENCOUT
Header	r_encout_rzt1_if.h r_encout_rzt1_dat.h
Declaration	r_encout_err_t R_ENCOUT_Control(const int32_t id, const r_encout_cmd_t cmd, void *const pbuf);
Description	This function operates ENCOUT. This function behaves differently depending on the argument “cmd”. Refer to “(a) R_ENCOUT_CMD_INIT”, “(b) R_ENCOUT_CMD_START”, “(c) R_ENCOUT_CMD_STOP” and “(d) R_ENCOUT_CMD_SET” for each operation.
Arguments	id : Specify R_ENCOUT0_ID cmd : Specify R_ENCOUT_CMD_INIT, R_ENCOUT_CMD_START, R_ENCOUT_CMD_STOP or R_ENCOUT_CMD_SET. pbuf : Depends on argument “cmd”
Return value	Refer to “(a) R_ENCOUT_CMD_INIT”, “(b) R_ENCOUT_CMD_START”, “(c) R_ENCOUT_CMD_STOP” and “(d) R_ENCOUT_CMD_SET”.

(a) R_ENCOUT_CMD_INIT

R_ENCOUT_CMD_INIT

Synopsis	Initialization of ENCOUT
Header	Refer to “(4) R_ENCOUT_Control”
Declaration	Refer to “(4) R_ENCOUT_Control”
Description	This function initializes ENCOUT. This function executes the processing of “2 Making initial settings for the ENCOUT” and “3 Setting the initial value in POSCNT” described in “4.1 Initialization” of “RZ/T1 Group Encoder Divided-Output Module (ENCOUT) User’s Manual”. Refer to “RZ/T1 Group Encoder Divided-Output Module (ENCOUT) User’s Manual” for details.

Arguments

id	: Refer to “(4) R_ENCOUT_Control”	
cmd	: Specify R_ENCOUT_CMD_INIT	
pbuf	: Specify a pointer to the “r_encout_init_t” structure storing the setting value. The member variables of the “r_encout_init_t” structure are below.	
	bool reverse_b	: Specify the value to be set in bit POL in register CTL. When “false” is specified, “0” is set. When “true” is specified, “1” is set.
	uint16_t position_max	: Specify the value to be set in register POSMAX. Refer to “RZ/T1 Group Encoder Divided-Output Module (ENCOUT) User’s Manual” for details.
	uint16_t encoder_count	: Specify the initial position value in the range 0 to ENCODER_RESOLUTION – 1. Refer to “3.5.4 How to change the setting value” for macro ENCODER_RESOLUTION.
	uint32_t carrier_period	: Specify the carrier period in ns. It can be specified in the range from 50000 to 3276999.

Return value

R_ENCOUT_SUCCESS	: Normal termination
R_ENCOUT_ERR_INVALID_ARG	: Abnormal termination (a value of the argument “id”, “cmd”, “encoder_count” or “carrier_period” is an undefined value, or a value of argument “position_max” is a prohibited value)
R_ENCOUT_ERR_ACCESS	: Abnormal termination (ENCOUT driver is not initialized)
R_ENCOUT_ERR_BUSY	: Abnormal termination (ENCOUT is running)

(b) R_ENCOUT_CMD_START

R_ENCOUT_CMD_START		
Synopsis	Start of ENCOUT	
Header	Refer to “(4) R_ENCOUT_Control”	
Declaration	Refer to “(4) R_ENCOUT_Control”	
Description	This function starts ENCOUT. This function executes the processing of “5 Starting the AB-phase and Z output” described in “4.1 Initialization” of “RZ/T1 Group Encoder Divided-Output Module (ENCOUT) User’s Manual”. Refer to “RZ/T1 Group Encoder Divided-Output Module (ENCOUT) User’s Manual” for details.	
Arguments	id	: Refer to “(4) R_ENCOUT_Control”
	cmd	: Specify R_ENCOUT_CMD_START
	pbuf	: not used
Return value	R_ENCOUT_SUCCESS	: Normal termination
	R_ENCOUT_ERR_INVALID_ARG	: Abnormal termination (a value of the argument “id” or “cmd” is an undefined value)
	R_ENCOUT_ERR_ACCESS	: Abnormal termination (ENCOUT driver is not initialized)
	R_ENCOUT_ERR_BUSY	: Abnormal termination (ENCOUT is running)

(c) R_ENCOUT_CMD_STOP

R_ENCOUT_CMD_STOP		
Synopsis	Stop of ENCOUT	
Header	Refer to “(4) R_ENCOUT_Control”	
Declaration	Refer to “(4) R_ENCOUT_Control”	
Description	This function stops ENCOUT.	
Arguments	id	: Refer to “(4) R_ENCOUT_Control”
	cmd	: Specify R_ENCOUT_CMD_STOP
	pbuf	: not used
Return value	R_ENCOUT_SUCCESS	: Normal termination
	R_ENCOUT_ERR_INVALID_ARG	: Abnormal termination (a value of the argument “id” or “cmd” is an undefined value)
	R_ENCOUT_ERR_ACCESS	: Abnormal termination (ENCOUT driver is not initialized)

(d) R_ENCOUT_CMD_SET

R_ENCOUT_CMD_SET	
Synopsis	Setting of ENCOUT
Header	Refer to “(4) R_ENCOUT_Control”
Declaration	Refer to “(4) R_ENCOUT_Control”
Description	This function sets OUTCNT register when ENCOUT is running. This function executes the processing described in “3 Calculating the values in the OUTCNT register” and “4 Setting the OUTCNT register” of “4.2 Main Processing”. Refer to “RZ/T1 Group Encoder Divided-Output Module (ENCOUT) User’s Manual” for details.
Arguments	id : Refer to “(4) R_ENCOUT_Control” cmd : Specify R_ENCOUT_CMD_SET pbuf : Specify a pointer to the “r_encout_set_t” structure storing the setting value. The member variables of the “r_encout_set_t” structure are below. uint32_t encoder_count : Specify the position value in the range 0 to ENCODER_RESOLUTION – 1. Refer to “3.5.4 How to change the setting value” for macro ENCODER_RESOLUTION.
Return value	R_ENCOUT_SUCCESS : Normal termination R_ENCOUT_ERR_INVALID_ARG : Abnormal termination (a value of the argument “id”, “cmd” or “encoder_count” is an undefined value) R_ENCOUT_ERR_ACCESS : Abnormal termination (ENCOUT is not running)

3.5.3 Overview of the processing

The procedure described in “4.1 Initalization” of “RZ/T1 Group Encoder Divided-Output Module (ENCOUT) User’s Manual” is implemented in “encout_start” function of “main.c” and the procedure described in “4.2 Main Processing” of “RZ/T1 Group Encoder Divided-Output Module (ENCOUT) User’s Manual” is implemented in “encout_main” function of “main.c”. However, the processing of “1 Acquiring positional information” described in “4.2 Main Processing” is realized by referring to array “encoder_data” in order instead of acquiring positional information. Also, the processing of “2 Control processing (system dependent)” is not implemented.

Refer to “RZ/T1 Group Encoder Divided-Output Module (ENCOUT) User’s Manual” for details.

The flowcharts of “main” function to initialize the sample program and “timer_isr” function which are activated for each carrier period and do main processing are shown below.

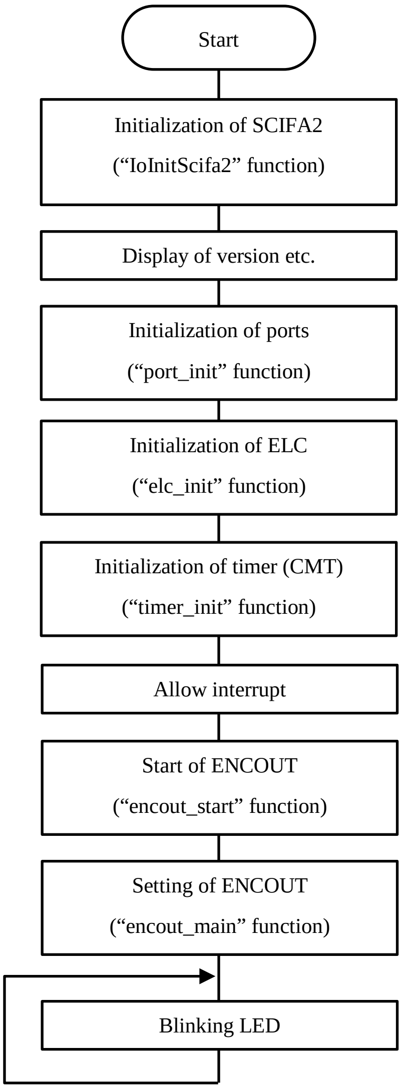


Figure 3-1 The flowchart of “main” function

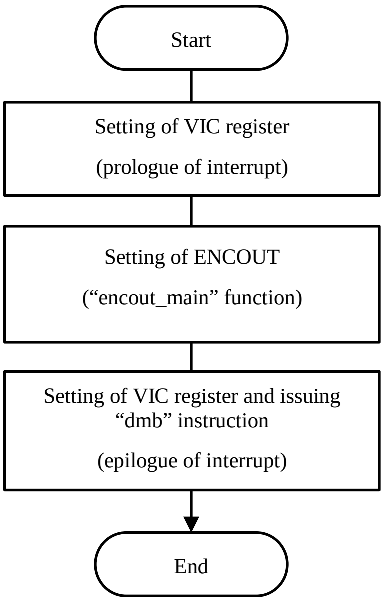


Figure 3-2 The flowchart of "tiemr_isr" function

3.5.4 How to change the setting value

Setting values of ENCOUT sample program can be changed as follows.

Setting value	file	How to change
Encoder resolution	r_encout_rzt1_config.h	Encoder resolution used to calculate the position by ENCOUT driver can be set. Specify encoder resolution to macro ENCODER_RESOLUTION. 32-bit value except “0” can be specified. For example, if the encoder resolution is 20 bits (position value is 0 to 1048575), specify 1048576. The default value is 1048576.
Carrier period	main.c	Carrier period of ABZ phase signals can be set. Specify carrier period to macro CARRIER_PERIOD in units of ns. The setting range is 50000 to 3276999. The default value is 100000ns (100us).
Polarity of phase B	main.c	Polarity of phase B output by ENCOUT can be set. If macro REVERSE_B is set to “false”, polarity of phase B is not reversed. If macro REVERSE_B is set to “true”, polarity of phase B is reversed. The default value is “false”.
Maximum position	main.c	Maximum position of ABZ phase signals output by ENCOUT can be set. Specify maximum position to macro POSITION_MAX. This value is set to register POSMAX of ENCOUT. Refer to “2.3 Maximum Position Register (POSMAX)” of “RZ/T1 Group Encoder Divided-Output Module (ENCOUT) User’s Manual” for the range of setting value. The default value is “99”.
ABZ phase output terminals	main.c	ABZ phase output terminals used by sample program can be changed. Change “port_reset” function and “port_set” function. Refer to “1.2 I/O Pins”, “1.3 Correspondence between I/O Pins and I/O Ports” and “4.1 Initialization” of “RZ/T1 Group Encoder Divided-Output Module (ENCOUT) User’s Manual” for available terminals and detailed setting method. The default terminals used by sample program are POUTA0, POUTB0 and POUTZ0.

3.6 How to combine Encoder I/F and ENCOUT

This chapter describes how to combine Encoder I/F and ENCOUT sample programs.

1. Copy the following files of ENCOUT sample program to Encoder I/F sample program.

```
Top folder
    inc
        iodefine_encout.h          ENCOUT register definition file
        r_encout_rzt1_dat.h        Header file for r_encout_rzt1.dat
        r_encout_rzt1_if.h         ENCOUT driver header file
    lib
        ecl
            r_encout_rzt1.dat      RZ/T1 ENCOUT Configuration Data
    src
        drv
            encout
                r_encout_rzt1_config.h     ENCOUT driver file
                r_encout_rzt1.c            ENCOUT driver file
        sample
            encout_dat.s           Linker setting file for configuration data *1
```

Note 1: file for DS-5/e2 studio
DS-5 : encout_dat.s
e2 studio : encout_dat.asm

When using EWARM manufactured by IAR Systems, add “r_encout_rzt1.c” to the project (It is recommended to create “encout” group under “drv” group and add it to the “encout” group.) and check “Use command line options” on menu “Project” -> “Options…” -> Category “Linker” -> tab “Extra Options” and input “`--image_input $PROJ_DIR$\lib\ecl\r_encout_rzt1.dat,g_encout_conf,ENCOUT_CONF_SEC,4`” to “Command line options”.

2. Change the part of main.c where “R_ECL_Configure” function is called as follows.

before	after

`...`
`extern const uint32_t g_xxx_config[];`
`...`
`ret_code = R_ECL_Configure(g_xxx_config);`
`...`
	`...`
	`extern const uint32_t g_xxx_config[];`
	`extern const uint32_t g_encout_config[];`
	`...`
	`const void *conf_array[2] =`
	`  { g_xxx_config, g_encout_config };`
	`ret_code = R_ECL_ConfigureMulti(conf_array, 2);`
	`...`

* The part of xxx depends on the type of Encoder I/F.

3. Modify main.c by referring to the “3.5.2 API of ENCOUT driver”, “3.5.3Overview of the processing” and “3.5.4 How to change the setting value”

4. Execute the program according to the procedure of Encoder I/F.

Note: To combine Encoder I/F and ENCOUT, “RZ/T1 Encoder I/F Configuration Library” Ver.2.0 Preliminary or later is required.

4. Restriction

None.

5. Note

5.1 Processing time

Available time for user processing of ENCOUT sample program in a control loop is as follows. Please confirm that there are no problems in your environment.

The example of the case that the carrier period is 62.5 us is indicated below.

The time used by the sample program is about 1 us (1.6%) of 62.5 us, and available time for user processing is about 61.5 us (98.4%).

Processing		Time	Occupancy rate
Processing of ENCOUT sample program ∗	Time setting OUTCNT register	about 1 us	1.6%
	Available time for user processing	about 61.5 us	98.4%

Note: Initial setting time is not included.