RELEASE NOTE

RZ/T1 Group

Encoder I/F BiSS-C application package

R01AN3561EJ0110
Rev.1.10
April 2, 2018

Summary

This document explains about RZ/T1 Encoder I/F BiSS-C application package.

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

For the detailed technical information on the BiSS C mode, please contact iC-Haus to obtain the protocol specification (BiSS C Protocol Description) and encoder specifications.

Device that BiSS functionality is checked

RZ/T1 CPU Board（RTK7910022C00000BR）

Version History

Ver.	Date	Content	Note
1.1	April 2018	Update the RZ/T1 BiSS-C sample driver code. (1) Added ID macro definition for ch1. (2) Changed register definition for ch1. (3) Added the SCIFA sample program. Update the RZ/T1 Group BiSS Interface (BiSS) User’s Manual.	
1.0	January 2017	Update the RZ/T1 Group BiSS Interface (BiSS) User’s Manual.	
0.9	December 2016	Update the RZ/T1 BiSS-C sample driver code. (1) Changed the error processing in the interrupt operation. (2) Added the sample driver code for KPIT GCC. (3) Improved the stability of the module stop release operation. (4) Improved the stability of the interrupt operation. Update the RZ/T1 Group BiSS Interface (BiSS) User’s Manual. Update the RZ/T1 Group BiSS-C Sample Program Application Note.	
0.8	August 2015	Newly created	

Table of contents

1. Contents of package

Contents of this package are described in this chapter.

Configuration data and sample programs included in this package support only 1 channel of Encoder I/F. In order to use 2 channels of Encoder I/F, obtain the RZ/T1 group Encoder I/F 2ch Tool (R01AN4306) and change the Configuration Data and sample program.

1.1 Software

・Source code

No.	Title	Version
1	A set of RZ/T1 BiSS-C sample driver code	1.2

・Configuration data

No.	Title	Version
1	RZ/T1 Encoder I/F Configuration Data (BiSS)	1.0

1.2 Document

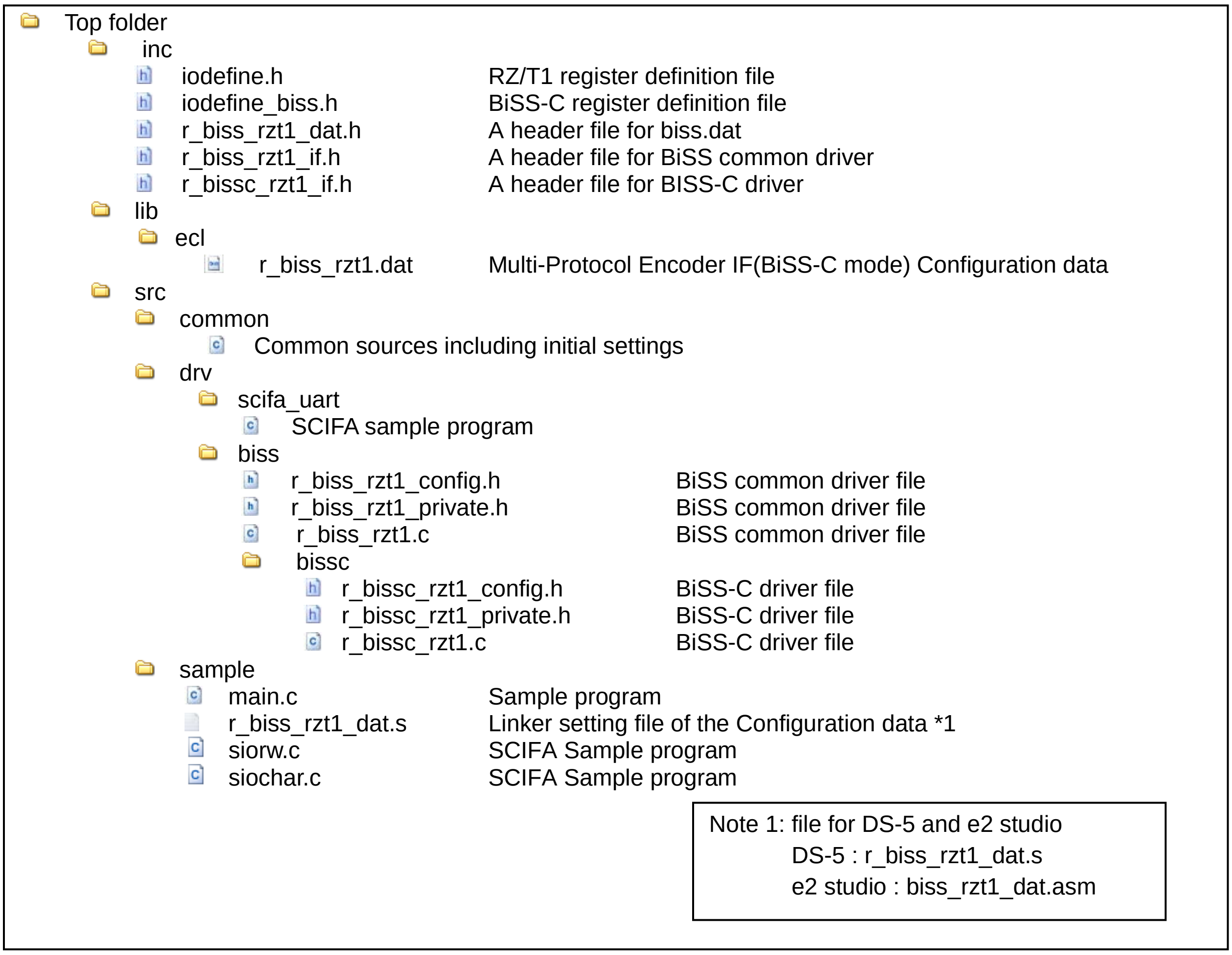
No.	Document name	Ver.	File name
1	RZ/T1 Encoder I/F BiSS-C sample program release note	1.10	(English) r01an3561ej0110-rzt1.pdf (this document) (Japanese) r01an3561jj0110-rzt1.pdf
2	RZ/T1 Group BiSS Interface (BiSS) User’s Manual	1.50	(English) r01uh0597ej0150_rzt1_biss.pdf (Japanese) r01uh0597jj0150_rzt1_biss.pdf
3	RZ/T1 Group BiSS-C Sample Program Application Note	1.20	(English) r01an2792ej0120_rzt1_biss-c.pdf (Japanese) r01an2792jj0120_rzt1_biss-c.pdf

2. File Structures

File structures and contents of this package are described below.

```
Top
├──r01an3561ej0110-rzt1.pdf
├──r01an3561jj0110-rzt1.pdf
└──workspace
      ├──Software
      │    ├──armcc
      │    │    └──RZ_T1_biss.zip：A set of RZ/T1 BiSS-C sample driver code (DS-5)
      │    ├──iccarm
      │    │    └──RZ_T1_biss.zip：A set of RZ/T1 BiSS-C sample driver code (IAR)
      │    └──kpitgcc
      │         └──RZ_T1_biss.zip：A set of RZ/T1 BiSS-C sample driver code (e2 studio)
      └──Documentation
           ├──r01an2792ej0120_rzt1_biss-c.pdf
           ├──r01an2792jj0120_rzt1_biss-c.pdf
           ├──r01uh0597ej0150_rzt1_biss.pdf
           └──r01uh0597jj0150_rzt1_biss.pdf
```

The file structures of “RZ_T1_biss.zip” are indicated below.



```
Top folder
    inc
        iodefine.h                RZ/T1 register definition file
        iodefine_biss.h           BiSS-C register definition file
        r_biss_rzt1_dat.h         A header file for biss.dat
        r_biss_rzt1_if.h          A header file for BiSS common driver
        r_bissc_rzt1_if.h         A header file for BISS-C driver
    lib
        ecl
            r_biss_rzt1.dat       Multi-Protocol Encoder IF(BiSS-C mode) Configuration data
    src
        common
            Common sources including initial settings
        drv
            scifa_uart
                SCIFA sample program
            biss
                r_biss_rzt1_config.h          BiSS common driver file
                r_biss_rzt1_private.h         BiSS common driver file
                r_biss_rzt1.c                 BiSS common driver file
                bissc
                    r_bissc_rzt1_config.h     BiSS-C driver file
                    r_bissc_rzt1_private.h    BiSS-C driver file
                    r_bissc_rzt1.c            BiSS-C driver file
        sample
            main.c                Sample program
            r_biss_rzt1_dat.s     Linker setting file of the Configuration data *1
            siorw.c               SCIFA Sample program
            siochar.c             SCIFA Sample program

                                  Note 1: file for DS-5 and e2 studio
                                          DS-5 : r_biss_rzt1_dat.s
                                          e2 studio : biss_rzt1_dat.asm
```

3. Information about BiSS sample program

This chapter describes information to use a set of BiSS sample program.

3.1 Software information

3.1.1 Operating System

This software is independent from operating system.

3.1.2 Memory footprint

Section name			Memory Size		
			IAR [bytes]	DS-5 [bytes]	e2 studio [bytes]
BiSS-C sample driver	Code		3076	4492	7912
	Data (with initial value)		8	46	8
	Data (without initial value)		104	66	112
	Constant Data		170	172	172
	Stack size of function	R_BISS_Open	60	68	104
		R_BISS_Close	36	44	80
		R_BISS_Control	64	88	120
		R_BISS_GetVersion	0	16	0
		bissc0_rx_int_isr	144+n *1	88+n *1	128+n *1
BiSS-C Configuration data	Code		0	0	0
	Data (with initial value)		0	0	0
	Data (without initial value)		0	0	0
	Constant Data		20684	20684	20684
Sample program	Code		2084	2908	4496
	Data (with initial value)		32	29	16
	Data (without initial value)		367	356	372
	Constant Data		1100	41	1089

*1 “n” is the Maximum stack size of user defined callback functions that are registered to R_BISS_Control function

3.2 Hardware information

3.2.1 Device

RZ/T1

3.2.2 Target Board

(1) Board name

RZ/T1 CPU Board (RTK7910022C00000BR)

(2) Settings of CPU Board

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

SW4-7: OFF

JP2: 2-3 Connect

JP7: 1-2 Connect

3.3 Procedure on Development Environments

3.3.1 Preparation for the execution of the sample program

This sample program communicates with the PC. And for setting the PC, please refer to 6.1.2 Preparations of "RZ/T1 Group FIFO Integrated Serial Communication Interface (SCIFA) Application Note".

3.3.2 EWARM from IAR systems

- Build environment

  IAR Embedded Workbench for ARM v7.80.2

- Execution environment

  I-jet

- How to build sample program

  1. Extract files from RZ_T1_biss.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_biss_serial_nor¥RZ_T1_bissc_****_boot.eww

NOR version	RZ_T1_bissc_nor_boot.eww
Serial Flash version	RZ_T1_bissc_serial_boot.eww

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

     Following file is generated.

     RZ_T1_biss_serial_nor¥Debug¥Exe¥RZ_T1_bissc_****_boot.out

NOR version	RZ_T1_bissc_nor_boot.out
Serial Flash version	RZ_T1_bissc_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]

- Execution result of sample program

  After executing a sample program, input the command to "Terminal I/O" window.
  Please refer to “RZ/T1 Group BiSS-C Sample Program Application Note” about the command.

3.3.3 DS-5 from ARM

- Build environment

  ARM Development Studio 5 (DS-5) Version 5.25.0

  ARM Compiler 5.06 update 3

- Execution environment

  ULINK2 (v2.01)

- How to build sample program

  1. Extract files from RZ_T1_biss.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”)

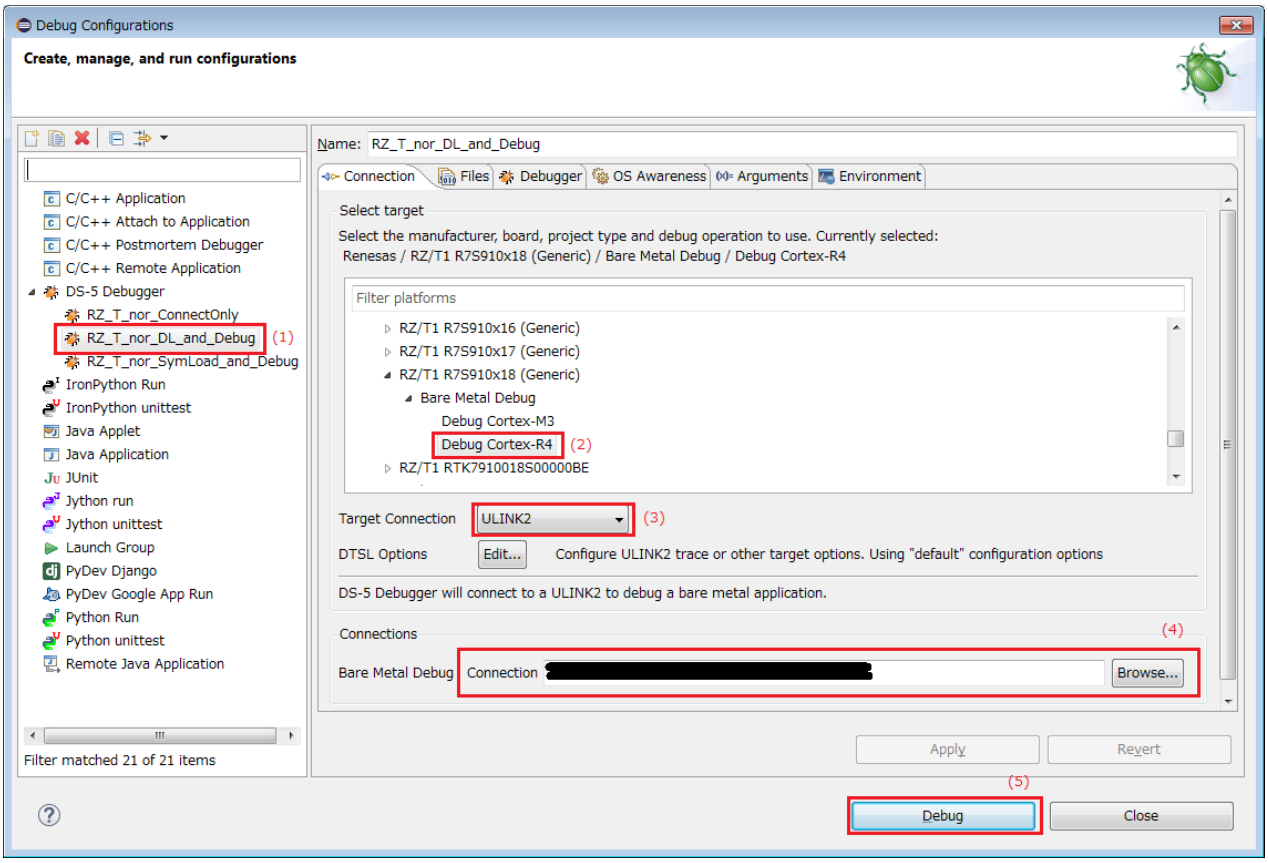
- How to execute sample program

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)

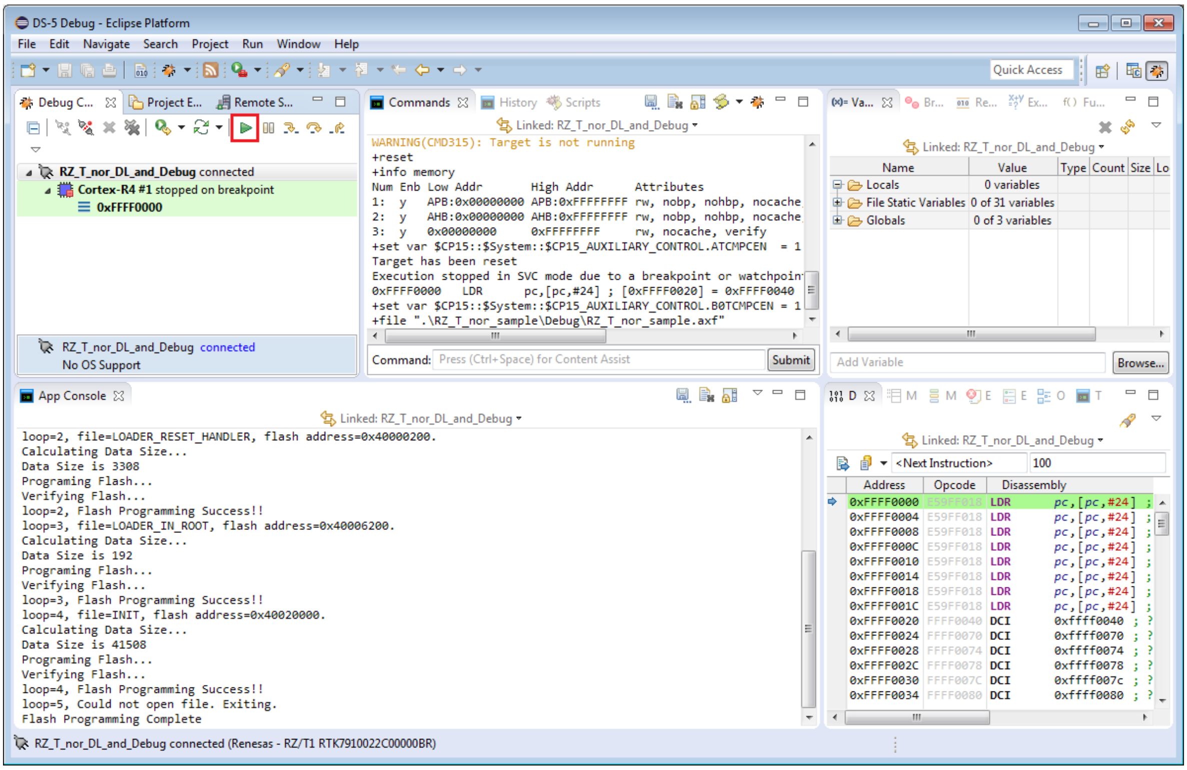
DTSL Options: Edit... Configure ULINK2 trace or other target options. Using "default" configuration options

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

Connections (4)

Bare Metal Debug Connection [redacted] Browse...

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
(x)= Va... Br... Re... Ex... f() 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)

➢ Execution result of sample program

After executing a sample program, input the command to "Terminal I/O" window.
Please refer to “RZ/T1 Group BiSS-C Sample Program Application Note” about the command.

3.3.4 e2 studio from RENESAS

- Build environment

  RENESAS e2 studio 5.2.0.020

  KPIT GNUARM-NONE-EABI Toolchain v16.01

- Execution environment

  J-Link BASE

- How to build sample program

  1. Extract files from RZ_T1_biss.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

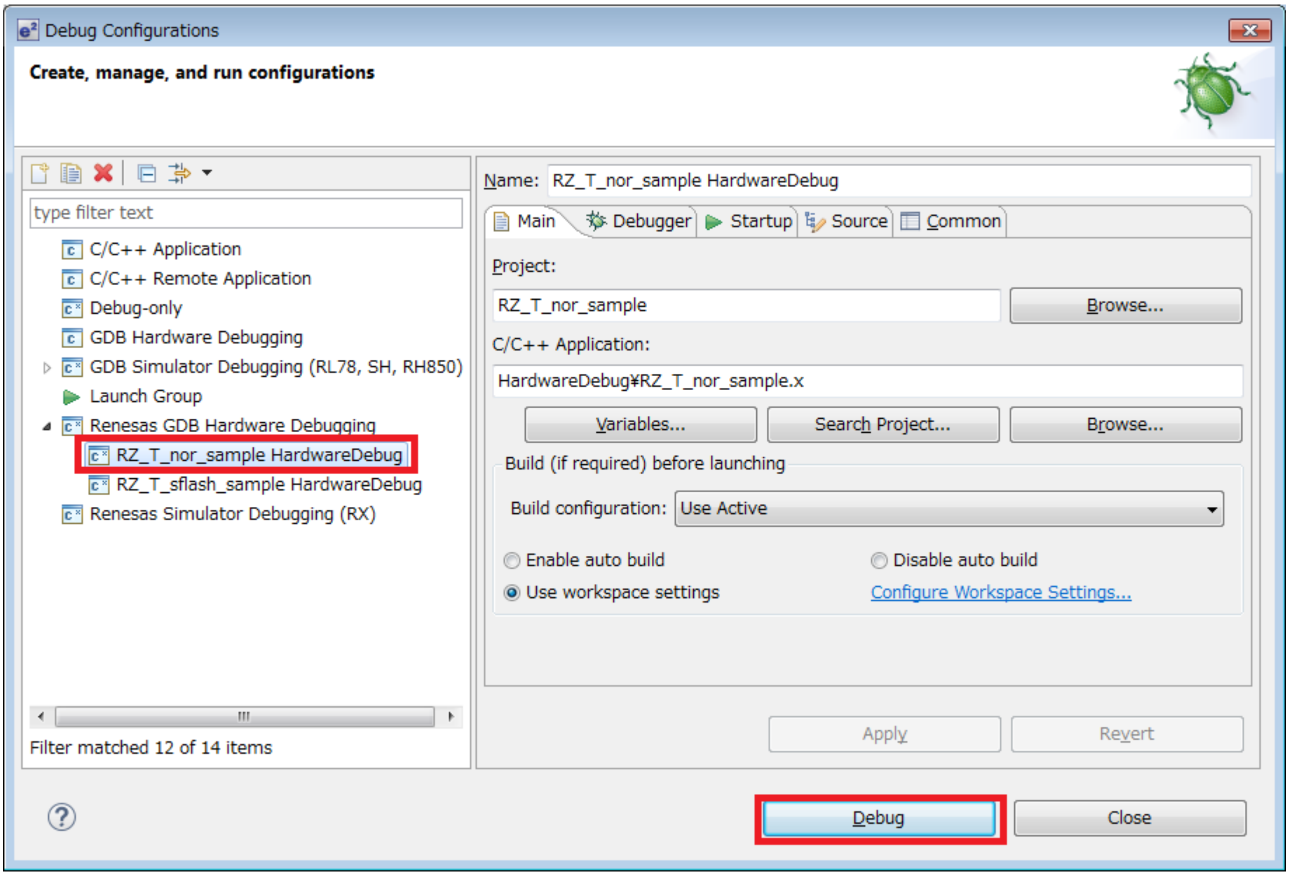
      (In case of serial flash, use the “RZ_T_sflash_sample.x” instead of the “RZ_T_nor_sample.x”)

- 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

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

Filter matched 12 of 14 items

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

- Execution result of sample program

After executing a sample program, input the command to "Terminal I/O" window.
Please refer to RZ/T1 Group BiSS-C Sample Program Application Note about the command.

4. Restriction

None.

5. Note

5.1 Processing time

Available time for user processing of Encoder I/F BiSS-C 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 control cycle is 62.5us is indicated below.

The time used by the sample program is about 7.7 us (13%) of 62.5us, and available time for user processing is about 54.8 us (87%).

Processing		Time		Occupancy rate
BiSS-C sample processing *2	Time setting registers for transmission	about 2.7 us	about 7.7us	13%
	Interrupt time	about 5 us		
Available time for user processing		about 54.8 us *1		87%

Note 1. For communication time with the encoder in available time for user processing, refer to section 6.1 AC Characteristics of “RZ/T1 Group BiSS Interface (BiSS) User's Manual”.

Note2. Initial setting time is not included.

5.2 Verified Encoders

Verified encoders by a production are indicated below.

Frequency of the transmission clock	Function	Verified Encoder				
		Danaher (HENGSTLER) AD36	RENISHAW RTLA-S	Kuebler F3663	Lika AM36	WACHENDORFF WDGF 58M
10 MHz	Acquisition of positional information	-	√	-	▨	-
	Register access	-	▨	-	▨	-
8.33 MHz	Acquisition of positional information	√	√	√	√	√
	Register access	√	▨	√	√*1	√
4 MHz	Acquisition of positional information	√	√	√	√	√
	Register access	√	▨	√	√*1	√
2.5 MHz	Acquisition of positional information	√	√	√	√	√
	Register access	√	▨	√	√*1	√
1 MHz	Acquisition of positional information	√	-	-	-	-
	Register access	√	▨	-	-	-
400 kHz	Acquisition of positional information	√	-	-	-	-
	Register access	√	▨	-	-	-
299.4 kHz	Acquisition of positional information	√	√	-	-	-
	Register access	√	▨	-	-	-
200 kHz	Acquisition of positional information	√	▨	-	√	-
	Register access	√	▨	-	√*1	-
100 kHz	Acquisition of positional information	√	▨	-	▨	-
	Register access	√	▨	-	▨	-
80.12 kHz	Acquisition of positional information	√	▨	√	▨	√
	Register access	√	▨	√	▨	√

√	: Verified.
√*1	: Consecutive register access not verified.
-	: Not verified.
▨	: Encoder not support.