RX Family

DAC Module Using Firmware Integration Technology

Introduction

This application note describes the 12 bit Digital Analog Converter (DAC) module which uses Firmware Integration Technology (FIT). This module uses DAC to control the operation of the DAC peripheral driver. In this document, this module is referred to as the DAC FIT module.

Target Devices

- RX111, RX113 Groups
- RX130 Group
- RX13T Group
- RX140 Group
- RX230, RX231 Groups
- RX23T Group
- RX23W Group
- RX24T Group
- RX24U Group
- RX64M Group
- RX651, RX65N Groups
- RX66T Group
- RX66N Group
- RX660 Group
- RX71M Group
- RX72T Group
- RX72M Group
- RX72N Group

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Target Compilers

- Renesas Electronics C/C++ Compiler Package for RX Family
- GCC for Renesas RX
- IAR C/C++ Compiler for Renesas RX

For details of the confirmed operation contents of each compiler, refer to “6.1 Confirmed Operation Environment".

Contents

1. Overview 4
1.1 DAC FIT Module 4
1.2 Overview of the DAC FIT Module 4
1.3 Using the FIT DAC module 4
1.3.1 Using FIT DAC module in C++ project 4
1.4 API Overview 4

2. API Information 5
2.1 Hardware Requirements 5
2.2 Software Requirements 5
2.3 Limitations 5
2.3.1 RAM Location Limitations 5
2.4 Supported Toolchain 5
2.5 Interrupt Vector 5
2.6 Header Files 5
2.7 Integer Types 6
2.8 Configuration Overview 6
2.9 Code Size 6
2.10 Parameters 13
2.11 Return Values 13
2.12 Callback Function 13
2.13 Adding the FIT Module to Your Project 13
2.14 “for”, “while” and “do while” statements 14

3. API Functions 15
R_DAC_Open() 15
R_DAC_Close() 18
R_DAC_Write() 19
R_DAC_Control() 20
R_DAC_GetVersion() 22

4. Pin Setting 23

5. Demo Projects 24
5.1 dac_demo_rskrx113, dac_demo_rskrx113_gcc 24
5.2 dac_demo_rskrx231, dac_demo_rskrx231_gcc 25
5.3 dac_demo_rskrx64m, dac_demo_rskrx64m_gcc 25
5.4 dac_demo_rskrx71m, dac_demo_rskrx71m_gcc 26
5.5 dac_demo_rskrx65n, dac_demo_rskrx65n_gcc 27
5.6 dac_demo_rskrx65n_2m, dac_demo_rskrx65n_2m_gcc 27
5.7 dac_demo_rskrx72m, dac_demo_rskrx72m_gcc 28

5.8 Adding a Demo to a Workspace 29
5.9 Downloading Demo Projects 29

6. Appendices 30
6.1 Confirmed Operation Environment 30
6.2 Troubleshooting 36

7. Reference Documents 37

Related Technical Updates 37

Revision History 38

1. Overview

1.1 DAC FIT Module

The DAC FIT module can be used by being implemented in a project as an API. See section 2.13, Adding the FIT Module to Your Project for details on methods to implement this FIT module into a project.

1.2 Overview of the DAC FIT Module

This DAC driver supports the DAC peripheral on the RX111, RX113, RX130, RX13T, RX140, RX230, RX231, RX23T, RX24T, RX24U, RX64M, RX65N, RX651, RX66T, RX66N, RX660, RX71M, RX72T, RX72M and RX72N. The hardware functionality is detailed in the D/A Converter chapter in the User's Manual: Hardware for each MCU.

Data to convert to analog may be either left or right justified, and channels can be output independently. MCU-specific hardware features are also supported. This includes selecting the reference voltage, synchronizing conversions with the ADC peripheral, disabling conversions when the output is disabled, and enabling an internal amplifier for larger loads.

1.3 Using the FIT DAC module

1.3.1 Using FIT DAC module in C++ project

For C++ project, add FIT DAC module interface header file within extern "C"{}:

```
Extern "C"
{
  #include "r_smc_entry.h"
  #include "r_dac_rx_if.h"
}
```

1.4 API Overview

Table 1.1 lists the API functions included in this module.

Table 1.1 API Functions

Function	Description
R_DAC_Open()	Applies power to the DAC peripheral, initializes the associated registers, and configures MCU-specific options.
R_DAC_Close()	Removes power to the DAC peripheral.
R_DAC_Write()	Writes data to channel register for conversion.
R_DAC_Control()	Enables or disables channel output. Enables or disabled internal amplifier (RX64M and RX71M).
R_DAC_GetVersion()	Returns at runtime the driver version number.

2. API Information

This FIT module has been confirmed to operate under the following conditions.

2.1 Hardware Requirements

The MCU used must support the following functions:

- DAC
- GPIO

2.2 Software Requirements

This driver is dependent upon the following FIT module:

- Renesas Board Support Package (r_bsp) v5.20 or higher

2.3 Limitations

2.3.1 RAM Location Limitations

In FIT, if a value equivalent to NULL is set as the pointer argument of an API function, error might be returned due to parameter check. Therefore, do not pass a NULL equivalent value as pointer argument to an API function.

The NULL value is defined as 0 because of the library function specifications. Therefore, the above phenomenon would occur when the variable or function passed to the API function pointer argument is located at the start address of RAM (address 0x0). In this case, change the section settings or prepare a dummy variable at the top of the RAM so that the variable or function passed to the API function pointer argument is not located at address 0x0.

In the case of the CCRX project (e2 studio V7.5.0), the RAM start address is set as 0x4 to prevent the variable from being located at address 0x0. In the case of the GCC project (e2 studio V7.5.0) and IAR project (EWRX V4.12.1), the start address of RAM is 0x0, so the above measures are necessary.

The default settings of the section may be changed due to the IDE version upgrade. Please check the section settings when using the latest IDE.

2.4 Supported Toolchain

This driver has been confirmed to work with the toolchain listed in 6.1, Confirmed Operation EnvironmentConfirmed Operation Environment.

2.5 Interrupt Vector

None.

2.6 Header Files

All API calls and their supporting interface definitions are located in “r_dac_rx_if.h”.

2.7 Integer Types

This project uses ANSI C99. These types are defined in stdint.h.

2.8 Configuration Overview

The configuration option settings of this module are located in r_dac_rx_config.h. The option names and setting values are listed in the table below:

Configuration options in r_dac_rx_config.h	
DAC_CFG_PARAM_CHECKING_ENABLE 1	If this equate is set to 1, parameter checking is included in the build. If the equate is set to 0, the parameter checking is omitted from the build and code size is reduced. Setting this equate to BSP_CFG_PARAM_CHECKING_ENABLE utilizes the system default setting.

2.9 Code Size

Typical code sizes associated with this module are listed below.

The ROM (code and constants) and RAM (global data) sizes are determined by the build-time configuration options described in 2.8, Configuration Overview. The table lists reference values when the C compiler's compile options are set to their default values, as described in 2.4,Supported Toolchain. The compile option default values are optimization level: 2, optimization type: for size, and data endianness: little-endian. The code size varies depending on the C compiler version and compile options.

ROM, RAM and Stack Code Sizes				
Device	Category	Memory Used		Remarks
		Renesas Compiler		
		With Parameter Checking	Without Parameter Checking	
RX130	ROM	314 bytes	282 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	32 bytes		R_DAC_Open function used
RX13T	ROM	284 bytes	257 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	60 bytes		R_DAC_Open function used
RX231	ROM	333 bytes	286 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	32 bytes		R_DAC_Open function used
RX23W	ROM	329 bytes	286 bytes	
	RAM	0	0	
	Maximum stack usage	32 bytes		R_DAC_Open function used
RX65N	ROM	409 bytes	367 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	32 bytes		R_DAC_Open function used
RX66T	ROM	405 bytes	373 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	32 bytes		R_DAC_Open function used
RX66N	ROM	551 bytes	497 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	8 bytes		R_DAC_Open function used
RX72T	ROM	405 bytes	373 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	32 bytes		R_DAC_Open function used
RX72M	ROM	407 bytes	365 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	32 bytes		R_DAC_Open function used
RX72N	ROM	551 bytes	510 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	8 bytes		R_DAC_Open function used

ROM, RAM and Stack Code Sizes				
Device	Category	Memory Used		Remarks
		Renesas Compiler		
		With Parameter Checking	Without Parameter Checking	
RX140	ROM	360 bytes	329 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	8 bytes		R_DAC_Open function used
RX660	ROM	449 bytes	418 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	8 bytes		R_DAC_Open function used

ROM, RAM and Stack Code Sizes				
Device	Category	Memory Used		Remarks
		GCC		
		With Parameter Checking	Without Parameter Checking	
RX130	ROM	576 bytes	536 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	-		
RX13T	ROM	440 bytes	400 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	-		
RX231	ROM	616 bytes	552 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	-		
RX65N	ROM	752 bytes	688 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	-		
RX66T	ROM	776 bytes	728 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	-		
RX66N	ROM	1048 bytes	975 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	-		
RX72T	ROM	776 bytes	728 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	-		
RX72M	ROM	746 bytes	704 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	-		
RX72N	ROM	1048 bytes	975 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	-		
RX140	ROM	640 bytes	600 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	-		

ROM, RAM and Stack Code Sizes				
Device	Category	Memory Used		Remarks
		Renesas Compiler		
		With Parameter Checking	Without Parameter Checking	
RX660	ROM	824 bytes	784 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	-		

ROM, RAM and Stack Code Sizes				
Device	Category	Memory Used		Remarks
		IAR Compiler		
		With Parameter Checking	Without Parameter Checking	
RX130	ROM	900 bytes	868 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	136 bytes		
RX13T	ROM	640 bytes	600 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	56 bytes		
RX231	ROM	932 bytes	876 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	136 bytes		
RX65N	ROM	1014 bytes	970 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	152 bytes		
RX66T	ROM	1034 bytes	1002 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	148 bytes		
RX66N	ROM	977 bytes	941 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	60 bytes		
RX72T	ROM	1038 bytes	1002 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	148 bytes		
RX72M	ROM	998 bytes	958 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	156 bytes		
RX72N	ROM	982 bytes	930 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	60 bytes		
RX140	ROM	728 bytes	696 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	56 bytes		

ROM, RAM and Stack Code Sizes				
Device	Category	Memory Used		Remarks
		Renesas Compiler		
		With Parameter Checking	Without Parameter Checking	
RX660	ROM	933 bytes	881 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	188 bytes		

2.10 Parameters

This section describes the parameter structure used by the API functions in this module. The structure is located in r_dac_rx_if.h as are the prototype declarations of API functions.

2.11 Return Values

This section describes return values of API functions. This enumeration is located in r_dac_rx_if.h as are the prototype declarations of API functions.

Below are the different error codes API functions can return. This enum is found in r_dac_rx_if.h along with the API function declarations.

```
/* DAC API ERROR CODE DEFINITIONS */
typedef enum e_dac_err
{
    DAC_SUCCESS=0,
    DAC_ERR_BAD_CHAN,          // non-existent channel number
    DAC_ERR_INVALID_CMD,       // non-existent operation command
    DAC_ERR_INVALID_ARG,       // argument is not valid for parameter
    DAC_ERR_NULL_PTR,          // received null ptr; missing required argument
    DAC_ERR_LOCK_FAILED,       // failed to lock DAC module (module already open)
    DAC_ERR_UNLOCK_FAILED      // failed to unlock DAC module
    DAC_ERR_ADC_NOT_POWERED,   // cannot sync because ADC is not powered
    DAC_ERR_ADC_CONVERTING     // cannot sync because ADC is converting
} dac_err_t;
```

2.12 Callback Function

None.

2.13 Adding the FIT Module to Your Project

This module must be added to each project in which it is used. Renesas recommends the method using the Smart Configurator described in (1) or (3) below. However, the Smart Configurator only supports some RX devices. Please use the methods of (2) or (4) for RX devices that are not supported by the Smart Configurator.

(1) Adding the FIT module to your project using the Smart Configurator in e^2 studio
By using the Smart Configurator in e^2 studio, the FIT module is automatically added to your project. Refer to “Renesas e^2 studio Smart Configurator User Guide (R20AN0451)” for details.

(2) Adding the FIT module to your project using the FIT Configurator in e^2 studio
By using the FIT Configurator in e^2 studio, the FIT module is automatically added to your project. Refer to “Adding Firmware Integration Technology Modules to Projects (R01AN1723)” for details.

(3) Adding the FIT module to your project using the Smart Configurator in CS+
By using the Smart Configurator Standalone version in CS+, the FIT module is automatically added to your project. Refer to “Renesas e^2 studio Smart Configurator User Guide (R20AN0451)” for details.

(4) Adding the FIT module to your project in CS+
In CS+, please manually add the FIT module to your project. Refer to “Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)” for details.

2.14 “for”, “while” and “do while” statements

In this module, “for”, “while” and “do while” statements (loop processing) are used in processing to wait for register to be reflected and so on. For these loop processing, comments with “WAIT_LOOP” as a keyword are described. Therefore, if user incorporates fail-safe processing into loop processing, user can search the corresponding processing with “WAIT_LOOP”.

The following shows example of description.

```
while statement example :
/* WAIT_LOOP */
while(0 == SYSTEM.OSCOVFSR.BIT.PLOVF)
{
    /* The delay period needed is to make sure that the PLL has stabilized. */
}

for statement example :
/* Initialize reference counters to 0. */
/* WAIT_LOOP */
for (i = 0; i < BSP_REG_PROTECT_TOTAL_ITEMS; i++)
{
    g_protect_counters[i] = 0;
}

do while statement example :
/* Reset completion waiting */
do
{
    reg = phy_read(ether_channel, PHY_REG_CONTROL);
    count++;
} while ((reg & PHY_CONTROL_RESET) && (count < ETHER_CFG_PHY_DELAY_RESET)); /* WAIT_LOOP */
```

3. API Functions

R_DAC_Open()

This function applies power to the DAC module, initializes the associated registers, and configures MCU-specific options.

Format

```
dac_err_t       R_DAC_Open (
        dac_cfg_t       * p_cfg
)
```

Parameters

*dac_cfg_t *p_cfg*

Pointer to the configuration structure.

Sample structure used for p_cfg:

```
typedef struct st_dac_cfg
{
    bool            fmt_flush_right;            // all MCUs
    bool            sync_with_adc;              // RX113/RX130/RX230/RX231/
// RX24U/RX63N/RX631/RX64M/
// RX65N/RX651/RX71M
    uint8_t         sync_unit;                  // 0 or 1; RX64M/RX71M
                                                // RX65N/RX651
    bool            ch_conv_off_when_output_off; // RX210/RX63N/RX631/RX64M/
 // RX65N/RX651/RX71M
    dac_refv_t      ref_voltage;                // RX113/RX230/RX231
} dac_cfg_t;

typedef enum e_dac_refv  // DAC reference voltage
{
    DAC_REFV_AVVC0_AVSS0    = 1,
    DAC_REFV_INTERNAL_AVSS0 = 3,
    DAC_REFV_VREFH_VREFL    = 6
} dac_refv_t;
```

Return Values

[DAC_SUCCESS]	*/* Successful; DAC initialized*	**/*
[DAC_ERR_NULL_PTR]	*/* p_cfg pointer is NULL*	**/*
[DAC_ERR_LOCK_FAILED]	*/* Failed to lock DAC module; already opened*	**/*
[DAC_ERR_INVALID_ARG]	*/* Invalid unit number for sync_unit*	**/*
[DAC_ERR_ADC_NOT_POWERED]	*/* Cannot sync because ADC is not powered*	**/*
[DAC_ERR_ADC_CONVERTING]	*/* Cannot sync because ADC is converting*	**/*

Properties

Prototyped in file “r_dac_rx_if.h”

Description

This function applies power to the DAC module, initializes the associated registers, and configures MCU-specific options.

Example

```
    dac_err_t  err;
    dac_cfg_t  config;

    /* Initialize RX63N DAC */
    config.fmt_flush_right = true;
    config.sync_with_adc = false;
    config.ch_conv_off_when_output_off = true;
    err = R_DAC_Open(&config);
```

Special Notes:

Data must be left or right justified by the application. The “fmt_flush_right” parameter just tells the DAC how to interpret the data.

To avoid a “DAC_ERR_ADC_CONVERTING”, open the DAC module after the ADC is opened but before scanning has begun on the ADC.

The DAC I/O pins must be configured prior to calling this function. An example initialization follows:

```
    R_BSP_RegisterProtectDisable(BSP_REG_PROTECT_MPC); // unlock
#ifdef BSP_MCU_RX113
    /*
     * Per Note 1 below Table 19.1 Allocation of Pin Functions to Multiple
* Pins (10/10) in the RX113 Group User's Manual: Hardware:
     *    Select general input (by setting the Bm bits for the given pin in the
     *    PDR and PMR for the given port to 0) for the pin if this pin function
     *    is to be used.
     */
    PORTJ.PDR.BIT.B0 = 0;
    PORTJ.PMR.BIT.B0 = 0;
    PORTJ.PDR.BIT.B2 = 0;
    PORTJ.PMR.BIT.B2 = 0;

  /* Set the pin function for PJ0 & PJ2 to be used as DAC analog output pins. */
    MPC.PJ0PFS.BIT.ASEL = 1;
    MPC.PJ2PFS.BIT.ASEL = 1;

    /*
     * Uncomment the two lines below if you want to use VREFH/VREFL for the DAC
     * reference voltage.
     */
    //MPC.P41PFS.BIT.ASEL = 1;   // Configure P41 as a VREFH analog pin
    //MPC.P42PFS.BIT.ASEL = 1;   // Configure P42 as a VREFL analog pin

#else  /* RX111, RX210, RX63N */

    /* Configure I/O port pins for analog outputs as general input pins.
    PORT0.PDR.BIT.B3 = 0;
    PORT0.PMR.BIT.B3 = 0;
    PORT0.PDR.BIT.B5 = 0;
    PORT0.PMR.BIT.B5 = 0;

  /* Set the pin function for P03 & P05 to be used as DAC analog output pins. */
    MPC.P03PFS.BIT.ASEL = 1;
    MPC.P05PFS.BIT.ASEL = 1;

#endif

    R_BSP_RegisterProtectEnable(BSP_REG_PROTECT_MPC);  // lock
```

Note when using an amplifier

When using an amp, set true in ch_conv_off_when_output_off.

Note when using the A/D converter
When the D/A A/D synchronous conversion (sync_with_adc = true) is enabled, if the A/D converter (1) is to be placed in the module stop state, first, execute the R_DAC_Close function.

Note 1. The intended A/D converter is unit 1 for RX64M/RX651/RX65N/RX66N/RX71M/RX72M/RX72N and unit 2 for RX24U.

R_DAC_Close()

This function removes power from the DAC peripheral.

Format
dac_err_t R_DAC_Close (void)

Parameters
None.

Return Values
[DAC_SUCCESS] / Successful; channels closed */*
[DAC_ERR_UNLOCK_FAILED] / Failed to unlock DAC module */*

Properties
Prototyped in file “r_dac_rx_if.h”.

Description
Disables DAC channel output and powers down the peripheral.

Example
```
    :
/* Initialize DAC Peripheral */
err = R_DAC_Open(&config);
 :
 :
/* Shut down DAC Peripheral */
err = R_DAC_Close();
```

Special Notes:
When the D/A A/D synchronous conversion (sync_with_adc = true) is enabled, if the A/D converter (1) is to be placed in the module stop state, first, execute the R_DAC_Close function.

Note 1. The intended A/D converter is unit 1 for RX64M/RX651/RX65N/RX66N/RX71M/RX72M/RX72N and unit 2 for RX24U.

R_DAC_Write()

This function writes data to channel data register.

Format

```
dac_err_t       R_DAC_Write (
        uint8_t const    chan,
        uint16_t                 data
)
```

Parameters

uint8_t const chan
Channel to write to.

uint16_t data
Data to write.

Return Values

[DAC_SUCCESS]	*/* Data written to channel register successfully*	*/*
[DAC_ERR_BAD_CHAN]	*/* Non-existent channel number*	*/*

Properties

Prototyped in file “r_dac_rx_if.h”

Description

Writes data to the channel register for conversion. Depending upon the MCU, this data may be 8-, 10-, or 12-bits in length. The data must be aligned properly for the selected format before issuing a Write().

Example

```
    dac_err_t     err;
    uint16_t      g_short;
      :
      :
    /* Write data for conversion to 0V on channel 1 */
    g_short = 0x0000;
    err = R_DAC_Write(DAC_CH1, g_short);
```

Special Notes:

None.

R_DAC_Control()

This function is used to enable/disable channel features.

Format

```
dac_err_t       R_DAC_Control (
        uint8_t const           chan,
        dac_cmd_t const                 cmd
)
```

Parameters

uint8_t const chan
Channel to operate on.

dac_cmd_t const cmd
Command to run (see enumeration below).

The *cmd* values are as follows:

```
typedef enum e_dac_cmd
{
    DAC_CMD_OUTPUT_ON,     // Analog output of channel is enabled
    DAC_CMD_OUTPUT_OFF,    // Analog output of channel is disabled
    DAC_CMD_AMP_ON,        // RX64M/RX71M: Gain of 1 amplifier. See Electrical
    DAC_CMD_AMP_OFF,       //   Characteristics in User's Manual: Hardware.
    DAC_CMD_ASW_ON,        // RX65N/RX66N,RX72M,RX72N: Wait for the channel 0
                              output buffer amplifier to become stable (the pin
                              is Hi-Z).
    DAC_CMD_ASW_OFF,       // A wait for stabilization of the channel 0
                              output buffer amplifier is released (output is
                              enabled).
    DAC_CMD_END_ENUM
} dac_cmd_t;
```

Return Values

```
[DAC_SUCCESS]           /* Successful; channel initialized */
[DAC_ERR_BAD_CHAN]      /* Non-existent channel number */
[DAC_ERR_INVALID_CMD]   /* Invalid command             */
```

Properties

Prototyped in file “r_dac_rx_if.h”.

Description

The output of conversion data written in data register by Write() function is enable by OUTPUT command while Amp is enable by AMP command, The output permission must be set after enabling amp.

Example

```
    dac_cfg_t     config;
    dac_err_t     err;

    /* Initialize RX64M,RX71M DAC */
    config.fmt_flush_right = true;
    config.sync_with_adc = true;
    config.sync_unit = 1;
    config.ch_conv_off_when_output_off = true;

    err = R_DAC_Open(&config);
```

```
    /* Write data for 0V on channel 0 */
    err = R_DAC_Write(DAC_CH0, 0x0);

    /* Drive a larger load */
    err = R_DAC_Control(DAC_CH0, DAC_CMD_AMP_ON);
    /*  It is necessary to wait more than 3us.  */

    /* Output converted data */
    err = R_DAC_Control(DAC_CH0, DAC_CMD_OUTPUT_ON);

    /* Write data for 3.3V on channel 0 */
    err = R_DAC_Write(DAC_CH0, 0x0FFF);
```

Special Notes:

Amp output is generated after R_DAC_Write(DAC_CHx, 0x0) is executed.
When amp out is in use (DAC_CMD_AMP_ON command running), set true in ch_conv_off_when_output_off.
When using an amp, follow the process below.

1. Execute DAC_CMD_ASW_ON command in R_DAC_Control function.*
2. Execute DAC_CMD_AMP_ON command in R_DAC_Control function.
3. Execute DAC_CMD_OUTPUT_ON command in R_DAC_Control function
4. Wait more than 3us
5. Execute DAC_CMD_ASW_OFF command in R_DAC_Control function.*
6. Write D/A output value in R_DAC_Write function.

Note *. (Only RX65N, RX66N, RX72M, RX72N)

The DAC_CMD_OUTPUT_ON and DAC_CMD_OUTPUT_OFF commands must be executed while the A/D converter (1) to be synchronized with is stopped when the D/A A/D synchronous conversion is enabled.

Note 1. For RX64M/RX651/RX65N/RX66N/RX71M/RX72M/RX72N, unit 1 of the A/D converter is to be stopped, and for RX24U, unit 2 is to be stopped. The other MCUs do not need to specify the unit to be stopped since they only have one unit.

R_DAC_GetVersion()

This function returns the driver version number at runtime.

Format
uint32_t R_DAC_GetVersion (void)

Parameters
None.

Return Values
Version number.

Properties
Prototyped in file “r_dac_rx.h”

Description
Returns the version of this module. The version number is encoded such that the top 2 bytes are the major version number and the bottom 2 bytes are the minor version number.

Example
```
uint32_t   version;
  :
version = R_DAC_GetVersion();
```

Special Notes:
None.

4. Pin Setting

To use the DAC FIT module, assign output signals of the peripheral function to pins with the multi-function pin controller (MPC). The pin assignment is referred to as the “Pin Setting” in this document. Please perform the pin setting before calling the R_DAC_Open() function.

5. Demo Projects

Demo projects include function main() that utilizes the FIT module and its dependent modules (e.g. r_bsp). This FIT module includes the following demo projects.

5.1 dac_demo_rskrx113, dac_demo_rskrx113_gcc

This is a simple demo of the RX113 D/A Converter (R12DAA) for the RSKRX113 starter kit (FIT module "r_dac_rx"). The demo uses the r_dac_rx API to enable, configure, and write to the DAC. A continuous loop is entered in which low, medium, and high data values are written to DAC channel 1 for 1 second each. LED 0 (green) is lit when a low value is written, LED 1 (orange) is lit when a medium value is written, and LED 2 (red) is lit when a high value is written. See the "Notes for Measuring the DAC Channel 0/1 Output Signals" section below for details on accessing and configuring the DAC output channel signals and reference voltages on the RSKRX113 board.

Setup and Execution

1. Compile and download the sample code.
2. Attach multimeter leads or oscilloscope probes to the DAC channel output pin(s) if desired.
3. Click 'Reset Go' to start the software. If PC stops at Main, press F8 to resume.
4. Set breakpoints and watch global variables

Boards Supported

RSKRX113

Notes For Measuring the DAC Channel 0/1 Output Signals

- DAC channel 0 (DA0) output uses I/O Port PJ0 which maps to pin 2 of the MCU.

  Per the RSKRX113 schematic, DA0 shares pin 2 with switch 1 (SW1). To use pin 2 for the DA0 analog output signal a 0 ohm resistor needs to be moved from R241 to R239. Once this has been done DA0 can be accessed via header JA1_13 or J1_2. Note that once this link configuration change is made SW1 will not be usable.

- DAC channel 1 (DA1) output uses I/O Port PJ2 which maps to physical pin 100.

  Per the RSKRX113 schematic, DA1 can be accessed via header JA1_14.

  DA1 can also be accessed via J4_25.

- GROUND can be accessed via JA1_2 (pin 4 next to it is also a ground pin)
- The RX113 supports three possible DAC reference voltages via the DAVREFCR register:

  1. AVCC0/AVSS0

     On the RSKRX113 board, AVCC0/AVSS0 are connected to UC_VCC (typ. 3.3V) and GROUND, respectively.

  2. Internal reference voltage/AVSS0

     Typically 1.4V and GROUND, respectively.

  3. VREFH/VREFL

     On the RSKRX113 board, VREFH/VREFL are not connected. They go to J4_18 (CON_VREFH) and J4_17 (CON_VREFL), respectively. In order to use VREFH/VREFL as the DAC reference voltage:

     - I/O pins P41 & P42 must be configured via the MPC as analog pins.
     - VREFH/VREEFL must be connected to high/low supply voltages:

       Refer to the "DAC Configuration" section of the RSKRX113 User's Manual (R20UT2762EJ0100) for details on the option links for configuring these signals.

     - An alternative option is:

Connect J3_12 (GROUND) to J4_17 (CON_VREFL)

Connect J3_10 (UC_VCC, 3.3V) to J4_18 (CON_VREFH)

5.2 dac_demo_rskrx231, dac_demo_rskrx231_gcc

This is a simple demo of the RX231 D/A Converter (R12DAA) for the RSKRX231 starter kit (FIT module "r_dac_rx"). The demo uses the r_dac_rx API to enable, configure, and write to the DAC. A continuous loop is entered in which low, medium, and high data values are written to DAC channel 1 for 1 second each. LED 0 (green) is lit when a low value is written, LED 1 (orange) is lit when a medium value is written, and LED 2 (red) is lit when a high value is written. See the "Operation" notes below for details on accessing and configuring the DAC output channel signals and reference voltages on the RSKRX231 board.

Setup and Execution

1. Compile and download the sample code.
2. Attach multimeter leads or oscilloscope probes to the DAC channel output pin(s) if desired.
3. Click 'Reset Go' to start the software. If PC stops at Main, press F8 to resume.
4. Set breakpoints and watch global variables

Boards Supported

RSKRX231

Notes For Measuring the DAC Channel 0/1 Output Signals

- DAC channel 0 (DA0) output uses I/O Port P03 which maps to pin 2 of the MCU. It can be accessed via J1_2.
- DAC channel 1 (DA1) output uses I/O Port P05 which maps to pin 100 of the MCU. It can be accessed via JA1_14 (or J4_25).
- GROUND can be accessed via JA1_2 (pin 4 next to it is also a ground pin)
- The RX231 supports three possible DAC reference voltages via the DAVREFCR register:

  1. AVCC0/AVSS0

     On the RSKRX231 board, AVCC0/AVSS0 are connected to UC_VCC (typ. 3.3V) and GROUND, respectively.

  2. Internal reference voltage/AVSS0

     Typically 1.4V and GROUND, respectively.

  3. VREFH/VREFL

     On the RSKRX231 board, VREFH/VREFL are connected to UC_VCC (typ. 3.3V) and GND, respectively. To connect an external reference voltage, CON_VREFH (J1_1) and CON_VREFL (J1_3) can be used after moving the following 0 ohm resistors:

     - For CON_VREFH move R68 to R67
     - For CON_VREFL move R65 to R66

5.3 dac_demo_rskrx64m, dac_demo_rskrx64m_gcc

This is a simple demo of the RX64M D/A Converter (R12DA) for the RSKRX64M starter kit (FIT module "r_dac_rx"). The demo uses the r_dac_rx API to enable, configure, and write to the DAC. A continuous loop is entered in which low, medium, and high data values are written to DAC channel 0 for 1 second each. LED 1 is lit when a low value is written, LED 2 is lit when a medium value is written, and LED 3 is lit when a high value is written. See the "Operation" notes below for details on accessing and configuring the DAC output channel signals on the RSKRX64M board.

Setup and Execution

1. Compile and download the sample code.
2. Attach multimeter leads or oscilloscope probes to the DAC channel output pin(s) if desired.
3. Click ‘Reset Go’ to start the software. If PC stops at Main, press F8 to resume.
4. Set breakpoints and watch global variables

Boards Supported

RSKRX64M

Notes For Measuring the DAC Channel 0/1 Output Signals

- DAC channel 0 (DA0) output uses I/O Port P03 which maps to pin 4 of the MCU.

  Per the RSKRX64M schematic, DA0 shares pin 4 with LED 0. To use pin 4 for the DA0 analog output signal a 0 ohm resistor needs to be moved from R277 to R189. Once this has been done DA0 can be accessed via header JA1_13. Note that once this link configuration change is made LED 0 will not be usable.

- DAC channel 1 (DA1) output uses I/O Port P05 which maps to pin 2 of the MCU.

  Per the RSKRX64M schematic, DA1 shares pin 2 with LED 1. To use pin 2 for the DA1 analog output signal a 0 ohm resistor needs to be moved from R280 to R188. Once this has been done DA1 can be accessed via header JA1_14. Note that once this link configuration change is made LED 1 will not be usable.

- On the RSKRX64M board the DA reference voltage AVCC1 (VREFH) and AVSS1 (VREFL) are connected to UC_VCC (typ. 3.3V) and GROUND, respectively.

5.4 dac_demo_rskrx71m, dac_demo_rskrx71m_gcc

This is a simple demo of the RX71M D/A Converter (R12DA) for the RSKRX71M starter kit (FIT module “r_dac_rx”). The demo uses the r_dac_rx API to enable, configure, and write to the DAC. A continuous loop is entered in which low, medium, and high data values are written to DAC channel 0 for 1 second each. LED 1 is lit when a low value is written, LED 2 is lit when a medium value is written, and LED 3 is lit when a high value is written. See the “Operation” notes below for details on accessing and configuring the DAC output channel signals on the RSKRX71M board.

Setup and Execution

1. Compile and download the sample code.
2. Attach multimeter leads or oscilloscope probes to the DAC channel output pin(s) if desired.
3. Click ‘Reset Go’ to start the software. If PC stops at Main, press F8 to resume.
4. Set breakpoints and watch global variables

Boards Supported

RSKRX71M

Notes For Measuring the DAC Channel 0/1 Output Signals

- DAC channel 0 (DA0) output uses I/O Port P03 which maps to pin 4 of the MCU.

  Per the RSKRX71M schematic, DA0 shares pin 4 with LED 0. To use pin 4 for the DA0 analog output signal a 0 ohm resistor needs to be moved from R281 to R195. Once this has been done DA0 can be accessed via header JA1_13. Note that once this link configuration change is made LED 0 will not be usable.

- DAC channel 1 (DA1) output uses I/O Port P05 which maps to pin 2 of the MCU.

  Per the RSKRX71M schematic, DA1 shares pin 2 with LED 1. To use pin 2 for the DA1 analog output signal a 0 ohm resistor needs to be moved from R284 to R194. Once this has been done DA1 can be

accessed via header JA1_14. Note that once this link configuration change is made LED 1 will not be usable.

- On the RSKRX71M board the DA reference voltage AVCC1 (VREFH) and AVSS1 (VREFL) are connected to UC_VCC (typ. 3.3V) and GROUND, respectively.

5.5 dac_demo_rskrx65n, dac_demo_rskrx65n_gcc

This is a simple demo of the RX65N D/A Converter (R12DA) for the RSKRX65N starter kit (FIT module "r_dac_rx"). The demo uses the r_dac_rx API to enable, configure, and write to the DAC. A continuous loop is entered in which low, medium, and high data values are written to DAC channel 0 for 1 second each. LED 1 is lit when a low value is written, LED 2 is lit when a medium value is written, and LED 3 is lit when a high value is written. See the "Operation" notes below for details on accessing and configuring the DAC output channel signals on the RSKRX65N board.

Setup and Execution

1. Compile and download the sample code.
2. Attach multimeter leads or oscilloscope probes to the DAC channel output pin(s) if desired.
3. Click 'Reset Go' to start the software. If PC stops at Main, press F8 to resume.
4. Set breakpoints and watch global variables

Boards Supported

RSKRX65N

Notes For Measuring the DAC Channel 0/1 Output Signals

- DAC channel 0 (DA0) output uses I/O Port P03 which maps to pin 4 of the MCU.

  Per the RSKRX65N schematic, DA0 shares pin 4 with LED 0. To use pin 4 for the DA0 analog output signal a 0 ohm resistor needs to be moved from R306 to R192. Once this has been done DA0 can be accessed via header JA1_13. Note that once this link configuration change is made LED 0 will not be usable.

- DAC channel 1 (DA1) output uses I/O Port P05 which maps to pin 2 of the MCU.

  Per the RSKRX65N schematic, DA1 shares pin 2 with LED 1. To use pin 2 for the DA1 analog output signal a 0 ohm resistor needs to be moved from R308 to R184. Once this has been done DA1 can be accessed via header JA1_14. Note that once this link configuration change is made LED 1 will not be usable.

  On the RSKRX65N board the DA reference voltage AVCC1 (VREFH) and AVSS1 (VREFL) are connected to UC_VCC (typ. 3.3V) and GROUND, respectively.

5.6 dac_demo_rskrx65n_2m, dac_demo_rskrx65n_2m_gcc

This is a simple demo of the RX65N-2MB D/A Converter (R12DA) for the RSKRX65N-2MB starter kit (FIT module "r_dac_rx"). The demo uses the r_dac_rx API to enable, configure, and write to the DAC. A continuous loop is entered in which low, medium, and high data values are written to DAC channel 0 for 1 second each. LED 1 is lit when a low value is written, LED 2 is lit when a medium value is written, and LED 3 is lit when a high value is written. See the "Operation" notes below for details on accessing and configuring the DAC output channel signals on the

RSKRX65N-2MB board.

Setup and Execution

1. Compile and download the sample code.
2. Attach multimeter leads or oscilloscope probes to the DAC channel output pin(s) if desired.
3. Click ‘Reset Go’ to start the software. If PC stops at Main, press F8 to resume.
4. Set breakpoints and watch global variables

Boards Supported

RSKRX65N-2MB

Notes For Measuring the DAC Channel 0/1 Output Signals

- DAC channel 0 (DA0) output uses I/O Port P03 which maps to pin 4 of the MCU.

  Per the RSKRX65N-2MB schematic, DA0 shares pin 4 with Switch 1. To use pin 4 for the DA0 analog output signal a 0 ohm resistor needs to be moved from R480 to R120. Once this has been done DA0 can be accessed via header JA1_13. Note that once this link configuration change is made Switch 1 will not be usable.

- DAC channel 1 (DA1) output uses I/O Port P05 which maps to pin 2 of the MCU.

  Per the RSKRX65N-2MB schematic, DA1 shares pin 2 with Switch 2. To use pin 2 for the DA1 analog output signal a 0 ohm resistor needs to be moved from R479 to R119. Once this has been done DA1 can be accessed via header JA1_14. Note that once this link configuration change is made Switch 2 will not be usable.

- On the RSKRX65N-2MB board the DA reference voltage AVCC1 (VREFH) and AVSS1 (VREFL) are connected to UC_VCC (typ. 3.3V) and GROUND, respectively.

5.7 dac_demo_rskrx72m, dac_demo_rskrx72m_gcc

This is a simple demo of the RX72M D/A Converter (R12DA) for the RSKRX72M starter kit (FIT module “r_dac_rx”). The demo uses the r_dac_rx API to enable, configure, and write to the DAC. A continuous loop is entered in which low, medium, and high data values are written to DAC channel 0 for 1 second each. LED 1 is lit when a low value is written, LED 2 is lit when a medium value is written, and LED 3 is lit when a high value is written. See the “Operation” notes below for details on accessing and configuring the DAC output channel signals on the

RSKRX72M board.

Setup and Execution

5. Compile and download the sample code.
6. Attach multimeter leads or oscilloscope probes to the DAC channel output pin(s) if desired.
7. Click ‘Reset Go’ to start the software. If PC stops at Main, press F8 to resume.
8. Set breakpoints and watch global variables

Boards Supported

RSKRX72M

Notes For Measuring the DAC Channel 0/1 Output Signals

- DAC channel 0 (DA0) output uses I/O Port P03 which maps to pin 4 of the MCU.

  Per the RSKRX72M schematic, DA0 shares pin 4 with SERIAL-CTS. To use pin 4 for the DA0 analog output signal a 0 ohm resistor needs to be moved from R275 to R218. Once this has been done DA0 can be accessed via header JA1_13. Note that once this link configuration change is made SERIAL-CTS will not be usable.

- DAC channel 1 (DA1) output uses I/O Port P05 which maps to pin 2 of the MCU.

  Per the RSKRX72M schematic, DA1 shares pin 2 with DSW-CATID3. To use pin 2 for the DA1 analog output signal a 0 ohm resistor needs to be moved from R201 to R214. Once this has been done DA1 can be accessed via header JA1_14. Note that once this link configuration change is made DSW-CATID3 will not be usable.

On the RSKRX72M board the DA reference voltage AVCC1 (VREFH) and AVSS1 (VREFL) are connected to UC_VCC (typ. 3.3V) and GROUND, respectively.

5.8 Adding a Demo to a Workspace

Demo projects are found in the FITDemos subdirectory of the distribution file for this application note. To add a demo
project to a workspace, select *File* >> *Import* >> *General* >> *Existing Projects into Workspace*, then click "Next". From the Import Projects dialog, choose the "Select archive file" radio button. "Browse" to the FITDemos subdirectory, select the desired demo zip file, then click "Finish".

5.9 Downloading Demo Projects

Demo projects are not included in the RX Driver Package. When using the demo project, the FIT module needs to be downloaded. To download the FIT module, right click on this application note and select "Sample Code (download)" from the context menu in the *Smart Browser* >> *Application Notes* tab.

6. Appendices

6.1 Confirmed Operation Environment

This section describes confirmed operation environment for the DAC FIT module.

Table 6.1 Confirmed Operation Environment (Rev.4.80)

Item	Contents
Integrated development environment	Renesas Electronics e2 studio Version 2022-04 IAR Embedded Workbench for Renesas RX 4.20.3
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.04.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
	GCC for Renesas RX 8.3.0.202104 Compiler option: The following option is added to the default settings of the integrated development environment. -std=gnu99 Linker option: The following user defined option should be added to the default settings of the integrated development environment, if “Optimize size (-Os)” is used: -Wl,--no-gc-sections This is to work around a GCC linker issue whereby the linker erroneously discard interrupt functions declared in FIT peripheral module
	IAR C/C++ Compiler for Renesas RX version 4.20.3 Compiler option: The default settings of the integrated development environment.
Endian	Big endian/little endian
Revision of the module	Rev.4.80
Board used	Renesas Starter Kit for RX660 (product No.: RTK556609HCxxxxxBJ)

Table 6.2 Confirmed Operation Environment (Rev.4.70)

Item	Contents
Integrated development environment	Renesas Electronics e2 studio Version 2021-10 IAR Embedded Workbench for Renesas RX 4.20.3
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.04.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
	GCC for Renesas RX 8.3.0.202104 Compiler option: The following option is added to the default settings of the integrated development environment. -std=gnu99 Linker option: The following user defined option should be added to the default settings of the integrated development environment, if “Optimize size (-Os)” is used: -Wl,--no-gc-sections This is to work around a GCC linker issue whereby the linker erroneously discard interrupt functions declared in FIT peripheral module
	IAR C/C++ Compiler for Renesas RX version 4.20.3 Compiler option: The default settings of the integrated development environment.
Endian	Big endian/little endian
Revision of the module	Rev.4.70
Board used	Renesas Starter Kit for RX66T (product No: RTK50566T0SxxxxxBE)

Table 6.3 Confirmed Operation Environment (Rev.4.60)

Item	Contents
Integrated development environment	Renesas Electronics e2 studio Version 2021-07 IAR Embedded Workbench for Renesas RX 4.20.3
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.03.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
	GCC for Renesas RX 8.3.0.202004 Compiler option: The following option is added to the default settings of the integrated development environment. -std=gnu99 Linker option: The following user defined option should be added to the default settings of the integrated development environment, if “Optimize size (-Os)” is used: -Wl,--no-gc-sections This is to work around a GCC linker issue whereby the linker erroneously discard interrupt functions declared in FIT peripheral module
	IAR C/C++ Compiler for Renesas RX version 4.20.3 Compiler option: The default settings of the integrated development environment.
Endian	Big endian/little endian
Revision of the module	Rev.4.60
Board used	Target board for RX140 (Part Number.: RTK5RX140xxxxxxxxx)

Table 6.4 Confirmed Operation Environment (Rev.4.50)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 7.8.0
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.02.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
	GCC for Renesas RX 8.3.0.201904 Compiler option: The following option is added to the default settings of the integrated development environment. -std=gnu99 Linker option: The following user defined option should be added to the default settings of the integrated development environment, if “Optimize size (-Os)” is used: -Wl,--no-gc-sections This is to work around a GCC linker issue whereby the linker erroneously discard interrupt functions declared in FIT peripheral module
Endian	Little endian
Revision of the module	Rev.4.50
Board used	Renesas Starter Kit+ for RX72M (product No.: RTK5572Mxxxxxxxxxx) Renesas Starter Kit+ for RX65N-2MB (product No.: RTK50565N2CxxxxxBR) Renesas Starter Kit+ for RX65N (product No.: RTK50565NCxxxxxBE) Renesas Starter Kit+ for RX64M (product No.: RTK50564Mxxxxxxxx) Renesas Starter Kit+ for RX71M (product No.: RTK50571Mxxxxxxxx) Renesas Starter Kit+ for RX113 (product No.: RTK505113xxxxxxxx) Renesas Starter Kit+ for RX231 (product No.: RTK505231xxxxxxxx)

Table 6.5 Confirmed Operation Environment (Rev.4.40)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 7.7.0 IAR Embedded Workbench for Renesas RX 4.12.1
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
	GCC for Renesas RX 4.8.4.201902 Compiler option: The following option is added to the default settings of the integrated development environment. -std=gnu99 Linker option: The following user defined option should be added to the default settings of the integrated development environment, if “Optimize size (-Os)” is used: -Wl,--no-gc-sections This is to work around a GCC linker issue whereby the linker erroneously discard interrupt functions declared in FIT peripheral module
	IAR C/C++ Compiler for Renesas RX version 4.12.1 Compiler option: The default settings of the integrated development environment.
Endian	Big endian/little endian
Revision of the module	Rev.4.40
Board used	Renesas Starter Kit+ for RX72N (product No.: RTK5572Nxxxxxxxxx)

Table 6.6 Confirmed Operation Environment (Rev.4.30)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 7.7.0 IAR Embedded Workbench for Renesas RX 4.12.1
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
	GCC for Renesas RX 4.8.4.201902 Compiler option: The following option is added to the default settings of the integrated development environment. -std=gnu99 Linker option: The following user defined option should be added to the default settings of the integrated development environment, if “Optimize size (-Os)” is used: -Wl,--no-gc-sections This is to work around a GCC linker issue whereby the linker erroneously discard interrupt functions declared in FIT peripheral module
	IAR C/C++ Compiler for Renesas RX version 4.12.1 Compiler option: The default settings of the integrated development environment.
Endian	Big endian/little endian
Revision of the module	Rev.4.30
Board used	RX13T CPU Card (product No.: RTK0EMXA10C00000BJ)

Table 6.7 Confirmed Operation Environment (Rev.4.20)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 7.5.0 IAR Embedded Workbench for Renesas RX 4.12.1
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
	GCC for Renesas RX 4.8.4.201902 Compiler option: The following option is added to the default settings of the integrated development environment. -std=gnu99 Linker option: The following user defined option should be added to the default settings of the integrated development environment, if “Optimize size (-Os)” is used: -Wl,--no-gc-sections This is to work around a GCC linker issue whereby the linker erroneously discard interrupt functions declared in FIT peripheral module
	IAR C/C++ Compiler for Renesas RX version 4.12.1 Compiler option: The default settings of the integrated development environment.
Endian	Big endian/little endian
Revision of the module	Rev.4.20
Board used	Renesas Starter Kit+ for RX72M (product No.: RTK50572Mxxxxxxxxx)

Table 6.8 Confirmed Operation Environment (Rev.4.10)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 7.5.0
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
Endian	Big endian/little endian
Revision of the module	Rev.4.10
Board used	Renesas Solution Starter Kit+ for RX23W (product No.: RTK5523Wxxxxxxxxx)

Table 6.9 Confirmed Operation Environment (Rev.4.00)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 7.4.0 IAR Embedded Workbench for Renesas RX 4.10.1
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
	GCC for Renesas RX 4.8.4.201803 Compiler option: The following option is added to the default settings of the integrated development environment. -std=gnu99 Linker option: The following user defined option should be added to the default settings of the integrated development environment, if “Optimize size (-Os)” is used: -Wl,--no-gc-sections This is to work around a GCC linker issue whereby the linker erroneously discard interrupt functions declared in FIT peripheral module
	IAR C/C++ Compiler for Renesas RX version 4.10.1 Compiler option: The default settings of the integrated development environment.
Endian	Big endian/little endian
Revision of the module	Rev.4.00
Board used	Renesas Starter Kit+ for RX65N-2MB (product No.: RTK50565Nxxxxxxxxx)

Table 6.10 Confirmed Operation Environment (Rev.3.30)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 7.3.0
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00
	Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
Endian	Big endian/little endian
Revision of the module	Rev.3.30
Board used	Renesas Starter Kit for RX72T (product No.: RTK5572Txxxxxxxxx)

Table 6.11 Confirmed Operation Environment (Rev.3.21)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 7.3.0
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00
	Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
Endian	Big endian/little endian
Revision of the module	Rev.3.21
Board used	Renesas Starter Kit for RX66T (product No.: RTK50566T0SxxxxxBE) Renesas Starter Kit+ for RX65N-2M (product No.: RTK50565N2CxxxxxBR) Renesas Starter Kit+ for RX130-512KB (product No.: RTK5051308CxxxxxBR)

Table 6.12 Confirmed Operation Environment (Rev.3.20)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 7.0.0
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.00.00
	Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
Endian	Big endian/little endian
Revision of the module	Rev.3.20
Board used	Renesas Starter Kit for RX66T (product No.: RTK50566T0SxxxxxBE) Renesas Starter Kit+ for RX65N-2M (product No.: RTK50565N2CxxxxxBR) Renesas Starter Kit+ for RX130-512KB (product No.: RTK5051308CxxxxxBR)

Table 6.13 Confirmed Operation Environment (Rev.3.11)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 6.0.0
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V2.07.00
	Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
Endian	Big endian/little endian
Revision of the module	Rev.3.11
Board used	Renesas Starter Kit+ for RX65N-2M (product No.: RTK50565N2CxxxxxBR) Renesas Starter Kit+ for RX130-512KB (product No.: RTK5051308CxxxxxBR)

Table 6.14 Confirmed Operation Environment (Rev.3.10)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 6.0.0
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V2.07.00
	Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
Endian	Big endian/little endian
Revision of the module	Rev.3.10
Board used	Renesas Starter Kit+ for RX65N-2M (product No.: RTK50565N2CxxxxxBR) Renesas Starter Kit+ for RX130-512KB (product No.: RTK5051308CxxxxxBR)

6.2 Troubleshooting

(1) Q: I have added the FIT module to the project and built it. Then I got the error: Could not open source file "platform.h".

A: The FIT module may not be added to the project properly. Check if the method for adding FIT modules is correct with the following documents:

- Using CS+:

  Application note "Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)"

- Using e^2 studio:

  Application note "Adding Firmware Integration Technology Modules to Projects (R01AN1723)"

When using a FIT module, the board support package FIT module (BSP module) must also be added to the project. Refer to the application note "Board Support Package Module Using Firmware Integration Technology (R01AN1685)".

(2) Q: I have added the FIT module to the project and built it. Then I got the error: This MCU is not supported by the current r_dac_rx module.

A: The FIT module you added may not support the target device chosen in your project. Check the supported devices of added FIT modules.

(3) Q: I have added the FIT module to the project and built it. Then I got an error for when the configuration setting is wrong.

A: The setting in the file "r_dac_rx_config.h" may be wrong. Check the file "r_dac_rx_config.h". If there is a wrong setting, set the correct value for that. Refer to 2.8, Configuration Overview for details.

(4) Q: Analog output is not done.

A: The pin setting may not be performed correctly. When using this FIT module, the pin setting must be performed. Refer to 4, Pin Setting for details.

7. Reference Documents

User's Manual: Hardware
The latest version can be downloaded from the Renesas Electronics website.

Technical Update/Technical News
The latest information can be downloaded from the Renesas Electronics website.

User's Manual: Development Tools
RX Family Compiler CC-RX User's Manual (R20UT3248)
The latest versions can be downloaded from the Renesas Electronics website.

Related Technical Updates

This module reflects the content of the following technical updates.
None

Revision History

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Nov.15.13	—	First edition issued
2.00	Apr.02.14	—	Updated for new API and RX210 & RX63N/631 support
2.10	Sep.08.14	—	Added RX64M support
2.20	Jan.20.15	—	Added RX113 support
2.30	Mar.19.15	—	Added RX71M support
2.40	Jun.30.15	—	Added RX231 support
2.50	Sep.30.15	—	Added RX23T support
2.60	Oct.1.15	—	Added RX130 support
2.70	Dec.1.15	—	Added RX230 support
		1, 5	Changed the document number for the “Board Support Package Firmware Integration Technology Module” application note.
		3	Changed the description in section 2.
		3	Removed “DAA” from the required peripheral lists in sections 2.1 and 2.2.
		7, 11	Modified some code examples shown in the Parameters and Example in sections 3.3 and 3.6.
		14	Added “4. Demo Projects”.
2.80	Feb.1.16	—	Added RX24T support
		18	Added “Related Technical Updates”.
2.91	Oct.1.16	—	Added RX65N Group support
		5	ROM, RAM and stack Code Sizes description change
		8	Added The notice when using an amplifier
		11, 12	Description, Example change, Special Notes addition
3.00	Feb.28.17	—	Added support for the RX24T (including ROM 512 KB version) and RX24U Groups.
		3	Added RXC v2.06.00 to “2.5 Supported Toolchains”.
		9, 10, 13	3.3 R_DAC_Open(), 3.4 R_DAC_Close(), and 3.6 R_DAC_Control(): Added the note when enabling the D/A A/D synchronous conversion in the Special Notes.
		Program	For RX64M, RX71M, and RX65N, the code has been modified to set unit 0 as the unit to be synchronized with and also generate an error when the D/A A/D synchronous conversion is enabled.
3.10	Jul.21.17	—	Added support for the RX130-512KB and RX65N-2MB.
		3	Added RXC v2.07.00 to “2.5 Supported Toolchains”.
		6	Changed section 2.11 “Adding the FIT Module to Your Project”
		15	Added section 4. Pin Setting.
3.11	Oct.31.17	18	5.4 dac_demo_rskrx71m Notes For Measuring the DAC Channel 0/1 Output Signals: Change resistors to R281, R195 for channel 0, R284, R194 for channel 1.
		19, 20	Added RSKRX65N, RSKRX65N-2M to “5. Demo Projects”
		20	Added 5.8 Downloading Demo Projects
		21	Added 6. Appendices
3.20	Sep 28, 2018	1, 3	Added support for the RX66T.
		5	Added code size corresponding to RX66T
		22	6.1 Confirmed Operation Environment: Added table for Rev.3.20
3.21	Nov 16, 2018	—	Added document number in XML
		22	Changed Renesas Starter Kit Product No for RX66T.

			Added table for Rev.3.21
3.30	Feb 01, 2019	Program	Added support for RX72T
		1, 3	Added support for RX72T
		5	Added code size corresponding to RX72T
		8-15	Removed ‘Reentrant’ description in each API function.
		22	6.1 Confirmed Operation Environment: Added table for Rev.3.30
4.00	May.20.19	—	Supported the following compilers: - GCC for Renesas RX - IAR C/C++ Compiler for Renesas RX
		1	Deleted the RX210, RX631, and RX63N in Target Devices for end of update these devices. Added the section of Target compilers. Deleted related documents.
		3	1.2 Overview of the DAC FIT Module Deleted the description of RX210, RX631, and RX63N.
		4	2.2 Software Requirements Requires r_bsp v5.20 or higher
		5-7	Updated the section of 2.8 Code Size.
		24	Table 6.1 Confirmed Operation Environment: Added table for Rev.4.00
		28	Deleted the section of Website and Support.
		Program	Changed below for support GCC and IAR compiler: Deleted the inline expansion of the R_DAC_GetVersion function.
4.10	Jun.28.19	1	Added support for RX23W
		5	Added code size corresponding to RX23W
		24	6.1 Confirmed Operation Environment: Added Table for Rev.4.10
		Program	Added support for RX23W.
4.20	Aug.15.19	1	Added support for RX72M.
		5-7	Added code size corresponding to RX72M
		24	6.1 Confirmed Operation Environment: Added Table for Rev.4.20 Table 6.2: Corrected board name for RX23W
		Program	Added support for RX72M.
4.30	Nov.25.19	1,3	Added support for RX13T.
		4	2.3 Limitations Added limitations.
		6-8	Added code size corresponding to RX13T
		25	6.1 Confirmed Operation Environment: Added Table for Rev.4.30
		Program	Added support for RX13T. Changed the comment of API functions to the doxygen style.
4.40	Dec.30.19	1,3	Added support for RX66N, RX72N.
		6-8	Added code size corresponding to RX66N, RX72N
		16	Added new cmd values in dac_cmd_t for RX65N, RX66N, RX72M, RX72N
		17	Added additional steps in Special Notes if using amplifier stabilization Wait
		25	6.1 Confirmed Operation Environment: Added Table for Rev.4.40
		Program	Added support for RX66N, RX72N. Added support for amplifier stabilization wait for RX65N,

			RX66N, RX72M, RX72N
4.50	Jun.30.20	20-25	Updated and added new demo project Added RSKRX72M to “5. Demo Projects”.
		26	6.1 Confirmed Operation Environment: Added Table for Rev.4.50
		Program	Updated and added new demo project
4.60	Apr.15.21	1, 4	Added support for RX140
		4	Added 1.3 Using the FIT DAC module. Added 1.3.1 Using FIT DAC module in C++ project.
		7-10	Added code size corresponding to RX140
		28	6.1 Confirmed Operation Environment: Added Table for Rev.4.60
		Program	Added support for RX140 Added CS+ support for demo project.
4.70	Mar.14.22	28	6.1 Confirmed Operation Environment: Added Table for Rev.4.70.
		Program	Added support for RX66T-48Pin.
4.80	Mar.31.22	1, 4	Added support for RX660
		8, 10, 12	Added code size corresponding to RX660
		30	6.1 Confirmed Operation Environment: Added Table for Rev.4.80
		Program	Added support for RX660.

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 VIL (Max.) and VIH (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 VIL (Max.) and VIH (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 be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. 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.
6. 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.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. 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.5.0-1 October 2020)

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.