RX Family

ADC Module Using Firmware Integration Technology

R01AN1666EJ0230
Rev. 2.30
July 24, 2017

Introduction

This module (ADC FIT module) provides support for all features of the 12-bit A/D Converter (S12AD) on the RX110, RX111, RX113, RX130, RX210, RX230, RX231, RX631, RX63N, RX64M, RX65N and RX71M.

Target Device

The following is a list of devices that are currently supported by this API:

- **RX110, RX111, RX113, RX130 Groups**
- **RX210 Group**
- **RX230, RX231 Groups**
- **RX631, RX63N Groups**
- **RX64M Group**
- **RX65N Group**
- **RX71M Group**

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

Related Documents

- Firmware Integration Technology User's Manual (R01AN1833)
- Board Support Package Module Using Firmware Integration Technology (R01AN1685)
- Adding Firmware Integration Technology Modules to Projects (R01AN1723)
- Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)
- Renesas e^2 studio Smart Configurator User Guide (R20AN0451)

Contents

1. Overview

This A/D Converter (ADC) driver supports the S12ADa peripheral on the RX63x, the S12ADb peripheral on the RX110/RX111/RX113/RX210, the S12ADC peripheral on the RX64M/RX71M, the S12ADE peripheral on the RX130/RX230/RX231, and the S12ADFa peripheral on the RX65x.

Depending on the MCU chosen, some features include, but are not limited to single scans, grouped single scans, and continuous scanning. Peripheral features include register left or right alignment, clearing data after register reads, summation and average of conversion results, and the ability to store data on alternate triggers of a channel. Channel, temperature and internal reference voltage sensor specific features include setting sampling time using state counts, and opting out of summation of samples. There is no dependency on any other software except for the board support package (r_bsp module).

The S12AD begins conversion when it receives a trigger. When the conversion is complete, a flag is set and an interrupt issued if enabled. If the S12AD is operating in a single scan mode, only one scan takes place per trigger. If the S12AD is operating in a continuous mode, scans continue indefinitely after the initial trigger occurs.

The majority of the driver serves to initialize the A/D peripheral and provide functions to read conversion results. With the ADC FIT module, settings which are common to all channels such as conversion alignment or addition count are set in the R_ADC_Open() call. Specific channel enabling is done via the R_ADC_Control() function. To retrieve conversion results, use the R_ADC_Read() function which retrieves a single conversion value or the R_ADC_ReadAll() function which retrieves all conversion registers.

The ADC FIT module supports the following 12-bit A/D Converter (S12AD) for each RX MCU.

Table 1.1 S12AD Supported by Each MCU

	S12ADa	S12ADb	S12ADC	S12ADE	S12ADFa
RX110		✓			
RX111		✓			
RX113		✓			
RX130				✓	
RX210		✓			
RX230				✓	
RX231				✓	
RX63x	✓				
RX64M			✓		
RX65x					✓
RX71M			✓		

2. API Information

The sample code in this application note has been run and confirmed under the following conditions.

2.1 Hardware Requirements

This driver requires your MCU support the following features:

- S12ADa, S12ADb, S12ADC, S12ADE or S12ADFa peripheral

2.2 Hardware Resource Requirements

This section details the hardware peripherals that this driver requires. Unless explicitly stated, these resources must be reserved for the driver and the user cannot use them.

2.2.1 S12ADa/S12ADb/S12ADC/S12ADE/S12ADFa

This driver makes use of all features available on these peripherals.

2.2.2 GPIO

This driver utilizes port pins corresponding to each individual analog channel.

2.3 Software Requirements

This driver is dependent upon the following packages:

- Renesas Board Support Package (r_bsp)

2.4 Limitations

Registers, settings, or usage notes vary depending on the mode used in the 12-bit A/D converter. APIs in this application note must be used according to the 12-bit A/D converter chapter in the User's Manual: Hardware for the MCU used.

For RX130-512KB, use the Renesas Board Support Package (r_bsp) Ver. 3.60 or later.

2.5 Supported Toolchains

This driver is tested and working with toolchains listed in 6.1 “Operation Confirmation Environment”.

2.6 Interrupt Vector

When the interrupt priority level is set to a value other than 0 in the R_ADC_Open() function, the interrupt (S12ADIn, S12GBADIn or GCADIn) for the interrupt source will be enabled.

Table 2.1 lists the interrupt vector used in the ADC FIT Module.

Table 2.1 Interrupt Vector Used in the ADC FIT Module

Device	Interrupt Vector
RX110, RX111, RX113, RX130, RX210, RX230, RX231	S12ADI0 interrupt (vector no.: 102) GBADI interrupt (vector no.: 103)
RX631, RX63N	S12ADI0 interrupt (vector no.: 102)
RX64M, RX71M	S12ADI0 interrupt (vector no.: 190) [(1)] S12ADI1 interrupt (vector no.: 192) [(1)] S12GBADI0 interrupt (vector no.: 191) [(1)] S12GBADI1 interrupt (vector no.: 193) [(1)] GROUPBL1 interrupt (vector no.: 111) ● S12CMPI0 interrupt (group interrupt source no.: 20) ● S12CMPI1 interrupt (group interrupt source no.: 22)
RX65N	S12ADI0 interrupt (vector no.: 186) [(1)] S12ADI1 interrupt (vector no.: 189) [(1)] S12GBADI0 interrupt (vector no.: 187) [(1)] S12GBADI1 interrupt (vector no.: 190) [(1)] S12GCADI0 interrupt (vector no.: 188) [(1)] S12GCADI1 interrupt (vector no.: 191) [(1)] GROUPBL1 interrupt (vector no.: 111) ● S12CMPAI interrupt (group interrupt source no.: 20) ● S12CMPBI interrupt (group interrupt source no.: 21) ● S12CMPAI1 interrupt (group interrupt source no.: 22) ● S12CMPBI1 interrupt (group interrupt source no.: 23)

Note 1. The interrupt vector numbers for software configurable interrupt B shown here are the default values specified in the board support package FIT module (BSP module).

2.7 Header Files

All API calls and their supporting interface definitions are located in the file “r_s12ad_rx_if.h” and this file should be included by the User’s application.

Build-time configuration options are selected or defined in the file “r_s12ad_rx_config.h”.

2.8 Integer Types

This project uses ANSI C99 “Exact width integer types” in order to make the code clearer and more portable. These types are defined in *stdint.h*.

2.9 Configuration Overview

All configurable options that can be set at build time are located in the file “r_s12ad_rx_config.h”. A summary of these settings are provided in the following table:

Configuration options in *r_s12ad_rx_config.h*	
`#define ADC_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. Setting this equate to BSP_CFG_PARAM_CHECKING_ENABLE utilizes the system default setting.

`// 1.8V <= AVcc0 < 2.7V`
`#define ADC_CFG_PGA_GAIN      0`
`// 2.7V <= AVcc0 < 3.6V`
`//#define ADC_CFG_PGA_GAIN    1`
`// 3.6V <= AVcc0 < 4.5V`
`//#define ADC_CFG_PGA_GAIN    2`
`// 4.5V <= AVcc0 <= 5.5V`
`//#define ADC_CFG_PGA_GAIN    3`
	This equate is for the Temperature Sensor gain amplifier on the RX210. The default is a value of 0 which is good for all target voltages. For best temperature resolution, the voltage range which most accurately reflects the AVcc0 should have its #define uncommented.

2.10 Code Size

The code size is based on optimization level 2 and optimization type for size for the RXC toolchain in Section 2.5. The ROM (code and constants) and RAM (global data) sizes are determined by the build-time configuration options set in the module configuration header file.

ROM and RAM code sizes		
	With Parameter Checking	**Without Parameter Checking**
RX110	ROM: 1354 bytes	ROM: 983 bytes
	RAM: 12 bytes	RAM: 12 bytes
RX111	ROM: 1234 bytes	ROM: 950 bytes
	RAM: 124 bytes	RAM: 124 bytes
RX113	ROM: 1471 bytes	ROM: 1100 bytes
	RAM: 12 bytes	RAM: 12 bytes
RX130	ROM: 2674 bytes	ROM: 2125 bytes
	RAM: 12 bytes	RAM: 12 bytes
RX210	ROM: 1671 bytes	ROM: 1200 bytes
	RAM: 12 bytes	RAM: 12 bytes

RX230, RX231	ROM: 2676 bytes	ROM: 2127 bytes
	RAM: 12 bytes	RAM: 12 bytes
RX63N	ROM: 1030 bytes	ROM: 792 bytes
	RAM: 12 bytes	RAM: 12 bytes
RX64M	ROM: 3667 bytes	ROM: 2962 bytes
	RAM: 32 bytes	RAM: 32 bytes
RX65N	ROM: 5325 bytes	ROM: 4284 bytes
	RAM: 40 bytes	RAM: 40 bytes
RX71M	ROM: 3667 bytes	ROM: 2962 bytes
	RAM: 32 bytes	RAM: 32 bytes

2.11 API Data Structures

This section details the data structures that are used with the driver's API functions.

To provide strong type checking and reduce errors, many parameters used in API functions require arguments to be passed using the provided type definitions. Allowable values are defined in the public interface files:

2.11.1 Callback Function Events (Common to All MCUs)

```
/* CALLBACK FUNCTION ARGUMENT DEFINITIONS */
typedef enum e_adc_cb_evt           // callback function events
{
    ADC_EVT_SCAN_COMPLETE,
    ADC_EVT_SCAN_COMPLETE_GROUPB,
#if (defined(BSP_MCU_RX65_ALL))
    ADC_EVT_SCAN_COMPLETE_GROUPC,
#endif
#if (defined(BSP_MCU_RX64M) || defined(BSP_MCU_RX71M) || defined(BSP_MCU_RX65_ALL))
    ADC_EVT_CONDITION_MET
#endif
defined(BSP_MCU_RX65_ALL))
    ADC_EVT_CONDITION_METB
#endif
} adc_cb_evt_t;
```

Enumeration Name	Description
ADC_EVT_SCAN_COMPLETE	Indicates completion of A/D conversion in single scan mode or group scan mode (group A).
ADC_EVT_SCAN_COMPLETE_GROUPB	Indicates completion of A/D conversion in group scan mode (group B).
ADC_EVT_SCAN_COMPLETE_GROUPC	Indicates completion of A/D conversion in group scan mode (group C). Only valid for S12ADFa.
ADC_EVT_CONDITION_MET	Indicates the comparison condition for window A has been met. Only valid for S12ADC and S12ADFa.
ADC_EVT_CONDITION_METB	Indicates the comparison condition for window B has been met. Only valid for S12ADFa.

2.11.2 Callback Function Arguments (Common to All MCUs)

```
typedef struct st_adc_cb_args       // callback arguments
{
    adc_cb_evt_t   event;
#if (defined(BSP_MCU_RX64M) || defined(BSP_MCU_RX71M) || defined(BSP_MCU_RX65_ALL))
    uint32_t       compare_flags;
#if (defined(BSP_MCU_RX65_ALL))
    uint32_t       compare_flagsb;
#endif
    uint8_t        unit;
#endif
} adc_cb_args_t;
```

Member	Description
event	Indicates the event occurred.
compare_flags	Stores the window A comparison result for each channel. The comparison result for channel n correspond to bit n. 0: Comparison condition not met. 1: Comparison condition met. Only valid for S12ADC and S12ADFa.
compare_flagsb	Stores the window B comparison result for each channel. The comparison result for channel n correspond to bit n. 0: Comparison condition not met. 1: Comparison condition met. Only valid for S12ADFa.
unit	Indicates the unit where the event occurred. Only valid for S12ADC and S12ADFa.

2.11.3 S12AD Operation Mode (S12ADb, A12ADE)

```
/* ADC_OPEN() ARGUMENT DEFINITIONS */
typedef enum e_adc_mode
{
    ADC_MODE_SS_TEMPERATURE,
    ADC_MODE_SS_INT_REF_VOLT,
    ADC_MODE_SS_ONE_CH,
    ADC_MODE_SS_MULTI_CH,
    ADC_MODE_CONT_ONE_CH,
    ADC_MODE_CONT_MULTI_CH,
    ADC_MODE_SS_ONE_CH_DBLTRIG,
    ADC_MODE_SS_MULTI_CH_GROUPED,
    ADC_MODE_SS_MULTI_CH_GROUPED_DBLTRIG_A,
    ADC_MODE_MAX        // DO NOT USE this definition for R_ADC_Open() argument.
} adc_mode_t;
```

Enumeration Name	Description
ADC_MODE_SS_TEMPERATURE	A/D conversion with temperature sensor in single scan mode. Select temperature sensor for the channel.
ADC_MODE_SS_INT_REF_VOLT	A/D conversion with internal reference voltage in single scan mode. Select internal reference voltage for the channel.
ADC_MODE_SS_ONE_CH	A/D conversion for one channel in single scan mode. Select only one channel for the channel. Neither internal reference voltage nor temperature sensor can be selected.
ADC_MODE_SS_MULTI_CH	A/D conversion for multiple channels in single scan mode. Neither internal reference voltage nor temperature sensor can be selected.
ADC_MODE_CONT_ONE_CH	A/D conversion for one channel in continuous scan mode. Select only one channel for the channel. Neither internal reference voltage nor temperature sensor can be selected.
ADC_MODE_CONT_MULTI_CH	A/D conversion for multiple channels in continuous scan mode. Neither internal reference voltage nor temperature sensor can be selected.
ADC_MODE_SS_ONE_CH_DBLTRIG	A/D conversion for one channel in double trigger mode. Select only one channel for the channel. Neither internal reference voltage nor temperature sensor can be selected.
ADC_MODE_SS_MULTI_CH_GROUPED	A/D conversion for multiple channels in group scan mode. Select different channels for group A and group B. Neither internal reference voltage nor temperature sensor can be selected.
ADC_MODE_SS_MULTI_CH_GROUPED_DBLTRIG_A	A/D conversion for multiple channels in group scan mode. Operates in double trigger mode for group A. Select one channel only for group A. For group B, select a channel other than the one selected for group A. Neither internal reference voltage nor temperature sensor can be selected.

2.11.4 S12AD Operation Mode (S12ADa)

```
/* ADC_OPEN() ARGUMENT DEFINITIONS */
typedef enum e_adc_mode
{
    ADC_MODE_SS_TEMPERATURE,
    ADC_MODE_SS_INT_REF_VOLT,
    ADC_MODE_SS_ONE_CH,
    ADC_MODE_SS_MULTI_CH,
    ADC_MODE_CONT_ONE_CH,
    ADC_MODE_CONT_MULTI_CH,
    ADC_MODE_MAX        // DO NOT USE this definition for R_ADC_Open() argument.
} adc_mode_t;
```

Enumeration Name	Description
ADC_MODE_SS_TEMPERATURE	A/D conversion with temperature sensor in single scan mode. Select temperature sensor for the channel.
ADC_MODE_SS_INT_REF_VOLT	A/D conversion with internal reference voltage in single scan mode. Select internal reference voltage for the channel.
ADC_MODE_SS_ONE_CH	A/D conversion for one channel in single scan mode. Select only one channel for the channel. Neither internal reference voltage nor temperature sensor can be selected.
ADC_MODE_SS_MULTI_CH	A/D conversion for multiple channels in single scan mode. Neither internal reference voltage nor temperature sensor can be selected.
ADC_MODE_CONT_ONE_CH	A/D conversion for one channel in continuous scan mode. Select only one channel for the channel. Neither internal reference voltage nor temperature sensor can be selected.
ADC_MODE_CONT_MULTI_CH	A/D conversion for multiple channels in continuous scan mode. Neither internal reference voltage nor temperature sensor can be selected.

2.11.5 S12AD Operation Mode (S12ADC)

```
typedef enum e_adc_mode
{
    ADC_MODE_SS_ONE_CH,
    ADC_MODE_SS_MULTI_CH,
    ADC_MODE_CONT_ONE_CH,
    ADC_MODE_CONT_MULTI_CH,
    ADC_MODE_SS_ONE_CH_DBLTRIG,
    ADC_MODE_SS_MULTI_CH_GROUPED,
    ADC_MODE_SS_MULTI_CH_GROUPED_DBLTRIG_A,
    ADC_MODE_MAX        // DO NOT USE this definition for R_ADC_Open() argument.
} adc_mode_t;
```

Enumeration Name	Description
ADC_MODE_SS_ONE_CH	A/D conversion for one channel in single scan mode. Select only one channel for the channel.
ADC_MODE_SS_MULTI_CH	A/D conversion for multiple channels in single scan mode.
ADC_MODE_CONT_ONE_CH	A/D conversion for one channel in continuous scan mode. Select only one channel for the channel.
ADC_MODE_CONT_MULTI_CH	A/D conversion for multiple channels in continuous scan mode.
ADC_MODE_SS_ONE_CH_DBLTRIG	A/D conversion for one channel in double trigger mode. Select only one channel for the channel.
ADC_MODE_SS_MULTI_CH_GROUPED	A/D conversion for multiple channels in group scan mode. Select different channels for group A and group B.
ADC_MODE_SS_MULTI_CH_GROUPED_DBLTRIG_A	A/D conversion for multiple channels in group scan mode. Operates in double trigger mode for group A. Select one channel only for group A. For group B, select a channel other than the one selected for group A.

2.11.6 S12AD Operation Mode (S12ADFa)

```
typedef enum e_adc_mode
{
    ADC_MODE_SS_ONE_CH,
    ADC_MODE_SS_MULTI_CH,
    ADC_MODE_CONT_ONE_CH,
    ADC_MODE_CONT_MULTI_CH,
    ADC_MODE_SS_ONE_CH_DBLTRIG,
    ADC_MODE_SS_MULTI_CH_GROUPED,
    ADC_MODE_SS_MULTI_CH_GROUPED_GROUPC,
    ADC_MODE_SS_MULTI_CH_GROUPED_DBLTRIG_A,
    ADC_MODE_SS_MULTI_CH_GROUPED_DBLTRIG_A_GROUPC,
    ADC_MODE_MAX        // DO NOT USE this definition for R_ADC_Open() argument
} adc_mode_t;
```

Enumeration Name	Description
ADC_MODE_SS_ONE_CH	A/D conversion for one channel in single scan mode. Select only one channel for the channel.
ADC_MODE_SS_MULTI_CH	A/D conversion for multiple channels in single scan mode.
ADC_MODE_CONT_ONE_CH	A/D conversion for one channel in continuous scan mode. Select only one channel for the channel.
ADC_MODE_CONT_MULTI_CH	A/D conversion for multiple channels in continuous scan mode.
ADC_MODE_SS_ONE_CH_DBLTRIG	A/D conversion for one channel in double trigger mode. Select only one channel for the channel.
ADC_MODE_SS_MULTI_CH_GROUPED	A/D conversion for multiple channels with two groups (group A and B). Select different channels for group A and group B.
ADC_MODE_SS_MULTI_CH_GROUPED_GROUPC	A/D conversion for multiple channels with three groups (group A, B, and C). Select a different channel for each group.
ADC_MODE_SS_MULTI_CH_GROUPED_DBLTRIG_A	A/D conversion for multiple channels with two groups (group A and B). Operates in double trigger mode for group A. Select one channel only for group A. For group B, select a channel other than the one selected for group A.
ADC_MODE_SS_MULTI_CH_GROUPED_DBLTRIG_A_GROUPC	A/D conversion for multiple channels with three groups (group A, B, and C). Select one channel only for group A. Then, select a different channel for each group.

2.11.7 S12AD Function Configuration Structure (S12ADb except RX210)

```
typedef struct st_adc_cfg
{
    adc_add_t        add_cnt;
    adc_align_t      alignment;
    adc_clear_t      clearing;
    adc_speed_t      conv_speed;
    adc_trig_t       trigger;
    adc_trig_t       trigger_groupb;
    uint8_t          priority;
    uint8_t          priority_groupb;
} adc_cfg_t;
```

Member	Description
add_cnt	Specifies addition mode.
alignment	Specifies the format. When addition mode is used, the setting of this member is ignored.
clearing	Enables/disables A/D data register automatic clearing.
conv_speed	Specifies A/D conversion mode (normal conversion/high-speed conversion).
trigger	Specifies A/D conversion start trigger.
trigger_groupb	Specifies A/D conversion start trigger for group B.
priority	Specifies the interrupt priority level (0 to 15) for the S12ADI0 interrupt. Setting to 0 disables the S12ADI0 interrupt.
priority_groupb	Specifies the interrupt priority level (0 to 15) for the GBADI interrupt. Setting to 0 disables the GBADI interrupt.

2.11.8 S12AD Function Configuration Structure (RX210 only)

```
typedef struct st_adc_cfg
{
    adc_add_t        add_cnt;
    adc_align_t      alignment;
    adc_clear_t      clearing;
    adc_trig_t       trigger;
    adc_trig_t       trigger_groupb;
    uint8_t          priority;
    uint8_t          priority_groupb;
} adc_cfg_t;
```

Member	Description
add_cnt	Specifies addition mode.
alignment	Specifies the format. When addition mode is used, the setting of this member is ignored.
clearing	Enables/disables A/D data register automatic clearing.
trigger	Specifies A/D conversion start trigger.
trigger_groupb	Specifies A/D conversion start trigger for group B.
priority	Specifies the interrupt priority level (0 to 15) for the S12ADI0 interrupt. Setting to 0 disables the S12ADI0 interrupt.
priority_groupb	Specifies the interrupt priority level (0 to 15) for the GBADI interrupt. Setting to 0 disables the GBADI interrupt.

2.11.9 S12AD Function Configuration Structure (S12ADE)

```
typedef struct st_adc_cfg
{
    adc_speed_t      conv_speed;
    adc_align_t      alignment;
    adc_add_t        add_cnt;
    adc_clear_t      clearing;
    adc_trig_t       trigger;
    adc_trig_t       trigger_groupb;
    uint8_t          priority;
    uint8_t          priority_groupb;
} adc_cfg_t;
```

Member	Description
conv_speed	Specifies A/D conversion mode (normal operating mode/high speed operating mode).
alignment	Specifies the format.
add_cnt	Specifies addition mode.
clearing	Enables/disables A/D data register automatic clearing.
trigger	Specifies A/D conversion start trigger.
trigger_groupb	Specifies A/D conversion start trigger for group B.
priority	Specifies the interrupt priority level (0 to 15) for the S12ADI0 interrupt. Setting to 0 disables the S12ADI0 interrupt.
priority_groupb	Specifies the interrupt priority level (0 to 15) for the GBADI interrupt. Setting to 0 disables the GBADI interrupt.

2.11.10 S12AD Function Configuration Structure (S12ADa)

```
typedef struct st_adc_cfg
{
    adc_add_t        add_cnt;
    adc_align_t      alignment;
    adc_clear_t      clearing;
    adc_speed_t      conv_speed;
    adc_trig_t       trigger;
    uint8_t          priority;
} adc_cfg_t;
```

Member	Description
add_cnt	Specifies addition mode.
alignment	Specifies the format. When addition mode is used, the setting of this member is ignored.
clearing	Enables/disables A/D data register automatic clearing.
conv_speed	Specifies A/D conversion mode (normal operating mode/high speed operating mode).
trigger	Specifies A/D conversion start trigger.
priority	Specifies the interrupt priority level (0 to 15) for the S12ADI0 interrupt. Setting to 0 disables the S12ADI0 interrupt.

2.11.11 S12AD Function Configuration Structure (S12ADC)

```
typedef struct st_adc_cfg
{
    adc_res_t       resolution;
    adc_align_t     alignment;
    adc_add_t       add_cnt;
    adc_clear_t     clearing;
    adc_trig_t      trigger;
    adc_trig_t      trigger_groupb;
    uint8_t         priority;
    uint8_t         priority_groupb;
} adc_cfg_t;
```

Member	Description
resolution	Specifies A/D conversion accuracy (8-bit, 10-bit, or 12-bit accuracy). The conversion time becomes shorter with lower resolution.
alignment	Specifies the format.
add_cnt	Specifies addition mode.
clearing	Enables/disables A/D data register automatic clearing.
trigger	Specifies A/D conversion start trigger.
trigger_groupb	Specifies A/D conversion start trigger for group B.
priority	Specifies the interrupt priority level (0 to 15) for the S12ADI interrupt. Setting to 0 disables the S12ADI interrupt.
priority_groupb	Specifies the interrupt priority level (0 to 15) for the S12GBADI interrupt. Setting to 0 disables the S12GBADI interrupt.

2.11.12 S12AD Function Configuration Structure (S12ADFa)

```
typedef struct st_adc_cfg
{
    adc_res_t        resolution;
    adc_align_t      alignment;
    adc_add_t        add_cnt;
    adc_clear_t      clearing;
    adc_trig_t       trigger;
    adc_trig_t       trigger_groupb;
    adc_trig_t       trigger_groupc;
    uint8_t          priority;
    uint8_t          priority_groupb;
    uint8_t          priority_groupc;
} adc_cfg_t;
```

Member	Description
resolution	Specifies A/D conversion accuracy (8-bit, 10-bit, or 12-bit accuracy). The conversion time becomes shorter with lower resolution.
alignment	Specifies the format.
add_cnt	Specifies addition mode.
clearing	Enables/disables A/D data register automatic clearing.
trigger	Specifies A/D conversion start trigger.
trigger_groupb	Specifies A/D conversion start trigger for group B.
trigger_groupc	Specifies A/D conversion start trigger for group C.
priority	Specifies the interrupt priority level (0 to 15) for the S12ADI interrupt. Setting to 0 disables the S12ADI interrupt.
priority_groupb	Specifies the interrupt priority level (0 to 15) for the S12GBADI interrupt. Setting to 0 disables the S12GBADI interrupt.
priority_groupc	Specifies the interrupt priority level (0 to 15) for the S12GCADI interrupt. Setting to 0 disables the S12GCADI interrupt.

2.11.13 A/D Conversion Trigger

Triggers available on S12AD vary depending on the MCU used. The following shows the definitions of triggers in RX110.

```
typedef enum e_adc_trig          // trigger sources
{
    ADC_TRIG_ASYNC_ADTRG    = 0,
    ADC_TRIG_SYNC_TRG0AN    = 1,
    ADC_TRIG_SYNC_TRG0BN    = 2,
    ADC_TRIG_SYNC_TRGAN     = 3,
    ADC_TRIG_SYNC_TRG0EN    = 4,
    ADC_TRIG_SYNC_TRG0FN    = 5,
    ADC_TRIG_SOFTWARE       = 16
} adc_trig_t;
```

The following table lists the S12AD triggers and available MCUs for each trigger.

Enumeration Name	Description	Available MCU
ADC_TRIG_NONE	Trigger source de-selection	RX130, RX230, RX231, RX64M, RX71M, RX65N
ADC_TRIG_SOFTWARE	Software trigger	All MCUs
ADC_TRIG_ASYNC_ADTRG	External trigger (ADTRG#)	All MCUs
ADC_TRIG_SYNC_TRG0AN	MTU0 TRGA	All MCUs
ADC_TRIG_SYNC_TRG0BN	MTU0 TRGB	RX110, RX111, RX113, RX210, RX63x, RX130, RX230, RX231
ADC_TRIG_SYNC_TRG1AN	MTU1 TRGA	RX64M, RX71M, RX65N
ADC_TRIG_SYNC_TRG2AN	MTU2 TRGA	RX64M, RX71M, RX65N
ADC_TRIG_SYNC_TRG3AN	MTU3 TRGA	RX64M, RX71M, RX65N
ADC_TRIG_SYNC_TRGAN	MTUx TRGA	RX110, RX63x
ADC_TRIG_SYNC_TRGAN_OR_UDF4N	MTUx TRGA or MTU4 underflow (complementary PWM)	RX111, RX113, RX210, RX130, RX230, RX231
ADC_TRIG_SYNC_TRG4AN_OR_UDF4N	MTU4 TRGA or MTU4 underflow (complementary PWM)	RX64M, RX71M, RX65N
ADC_TRIG_SYNC_TRG6AN	MTU6 TRGA	RX64M, RX71M, RX65N
ADC_TRIG_SYNC_TRG7AN_OR_UDF7N	MTU7 TRGA or MTU7 underflow (complementary PWM)	RX64M, RX71M, RX65N
ADC_TRIG_SYNC_TRG0EN	MTU0 TRGE	All MCUs
ADC_TRIG_SYNC_TRG0FN	MTU0 TRGF	RX110, RX111, RX113, RX210, RX63x, RX130, RX230, RX231
ADC_TRIG_SYNC_TRG4AN	MTU4 TADCORA	RX111, RX113, RX210, RX130, RX230, RX231, RX64M, RX71M, RX65N
ADC_TRIG_SYNC_TRG4BN	MTU4 TADCORB	RX111, RX113, RX210, RX130, RX230, RX231, RX64M, RX71M, RX65N
ADC_TRIG_SYNC_TRG4AN_OR_TRG4BN	MTU4 TADCORA or TADCORB	RX63x, RX64M, RX71M, RX65N
ADC_TRIG_SYNC_TRG4AN_AND_TRG4BN	MTU4 TADCORA and TADCORB	RX111, RX113, RX210, RX130, RX230, RX231, RX64M, RX71M, RX65N
ADC_TRIG_SYNC_TRG7AN	MTU7 TADCORA	RX64M, RX71M, RX65N
ADC_TRIG_SYNC_TRG7BN	MTU7 TADCORB	RX64M, RX71M, RX65N
ADC_TRIG_SYNC_TRG7AN_OR_TRG7BN	MTU7 TADCORA or TADCORB	RX64M, RX71M, RX65N
ADC_TRIG_SYNC_TRG7AN_AND_TRG7BN	MTU7 TADCORA and TADCORB	RX64M, RX71M, RX65N
ADC_TRIG_SYNC_GTADTR0AN	GPT0 GTADTRA	RX64M, RX71M
ADC_TRIG_SYNC_GTADTR0BN	GPT0 GTADTRB	RX64M, RX71M
ADC_TRIG_SYNC_GTADTR1AN	GPT1 GTADTRA	RX64M, RX71M
ADC_TRIG_SYNC_GTADTR1BN	GPT1 GTADTRB	RX64M, RX71M
ADC_TRIG_SYNC_GTADTR2AN	GPT2 GTADTRA	RX64M, RX71M
ADC_TRIG_SYNC_GTADTR2BN	GPT2 GTADTRB	RX64M, RX71M
ADC_TRIG_SYNC_GTADTR3AN	GPT3 GTADTRA	RX64M, RX71M
ADC_TRIG_SYNC_GTADTR3BN	GPT3 GTADTRB	RX64M, RX71M

Enumeration Name	Description	Available MCU
ADC_TRIG_SYNC_GTADTR0AN_OR_GTADTR0BN	GPT0 GTADTRA or GTADTRB	RX64M, RX71M
ADC_TRIG_SYNC_GTADTR1AN_OR_GTADTR1BN	GPT1 GTADTRA or GTADTRB	RX64M, RX71M
ADC_TRIG_SYNC_GTADTR2AN_OR_GTADTR2BN	GPT2 GTADTRA or GTADTRB	RX64M, RX71M
ADC_TRIG_SYNC_GTADTR3AN_OR_GTADTR3BN	GPT3 GTADTRA or GTADTRB	RX64M, RX71M
ADC_TRIG_SYNC_TMRTRG0AN	TMR0 TCORA	RX63x, RX64M, RX71M, RX65N
ADC_TRIG_SYNC_TMRTRG2AN	TMR2 TCORA	RX63x, RX64M, RX71M, RX65N
ADC_TRIG_SYNC_TPUTRG0AN	TPU0 TRGA	RX210, RX230, RX231, RX63x, RX64M, RX71M, RX65N
ADC_TRIG_SYNC_TPUTRGAN	TPUx TRGA	RX210, RX230, RX231, RX63x, RX64M, RX71M, RX65N
ADC_TRIG_SYNC_TEMPS	Temperature sensor	RX210
ADC_TRIG_SYNC_ELC	ELC	RX111, RX113, RX210, RX130, RX230, RX231, RX64M, RX71M, RX65N

2.11.14 Addition Mode (S12ADa, S12ADb)

```
typedef enum e_adc_add
{
    ADC_ADD_OFF = 0,
    ADC_ADD_TWO_SAMPLES = 1,
    ADC_ADD_THREE_SAMPLES = 2,
    ADC_ADD_FOUR_SAMPLES = 3
} adc_add_t;
```

Enumeration Name	Description
ADC_ADD_OFF	Addition mode not used.
ADC_ADD_TWO_SAMPLES	Performs A/D conversion twice (adding up once).
ADC_ADD_THREE_SAMPLES	Performs A/D conversion three times (adding up twice).
ADC_ADD_FOUR_SAMPLES	Performs A/D conversion four times (adding up three times).

2.11.15 Addition Mode (S12ADC)

```
typedef enum e_adc_add
{
    ADC_ADD_OFF = 0,
    ADC_ADD_TWO_SAMPLES = 1,
    ADC_ADD_THREE_SAMPLES = 2,
    ADC_ADD_FOUR_SAMPLES = 3,
    ADC_ADD_AVG_2_SAMPLES = 0x81,
    ADC_ADD_AVG_4_SAMPLES = 0x83,
} adc_add_t;
```

Enumeration Name	Description
ADC_ADD_OFF	Addition mode not used.
ADC_ADD_TWO_SAMPLES	Performs A/D conversion twice (adding up once).
ADC_ADD_THREE_SAMPLES	Performs A/D conversion three times (adding up twice).
ADC_ADD_FOUR_SAMPLES	Performs A/D conversion four times (adding up three times).
ADC_ADD_AVG_2_SAMPLES	Average two samples.
ADC_ADD_AVG_4_SAMPLES	Average four samples.

2.11.16 Addition Mode (S12ADE, S12ADFa)

```
typedef enum e_adc_add
{
    ADC_ADD_OFF              = 0,
    ADC_ADD_TWO_SAMPLES      = 1,
    ADC_ADD_THREE_SAMPLES    = 2,
    ADC_ADD_FOUR_SAMPLES     = 3,
    ADC_ADD_SIXTEEN_SAMPLES = 5,
    ADC_ADD_AVG_2_SAMPLES    = 0x81,
    ADC_ADD_AVG_4_SAMPLES    = 0x83,
} adc_add_t;
```

Enumeration Name	Description
ADC_ADD_OFF	Addition mode not used.
ADC_ADD_TWO_SAMPLES	Performs A/D conversion twice (adding up once).
ADC_ADD_THREE_SAMPLES	Performs A/D conversion three times (adding up twice).
ADC_ADD_FOUR_SAMPLES	Performs A/D conversion four times (adding up three times).
ADC_ADD_ SIXTEEN _SAMPLES	Performs A/D conversion sixteen times (adding up 15 times).
ADC_ADD_AVG_2_SAMPLES	Average two samples.
ADC_ADD_AVG_4_SAMPLES	Average four samples.

2.11.17 A/D Conversion Accuracy (S12ADC, S12ADFa)

```
typedef enum e_adc_res
{
    ADC_RESOLUTION_12_BIT = 0,
    ADC_RESOLUTION_10_BIT = 1,
    ADC_RESOLUTION_8_BIT = 2
} adc_res_t  ;
```

Enumeration Name	Description
ADC_RESOLUTION_12_BIT	Performs A/D conversion with 12-bit accuracy.
ADC_RESOLUTION_10_BIT	Performs A/D conversion with 10-bit accuracy.
ADC_RESOLUTION_8_BIT	Performs A/D conversion with 8-bit accuracy.

2.11.18 Format (Common in All MCUs)

```
typedef enum e_adc_align
{
    ADC_ALIGN_RIGHT = 0x0000,
    ADC_ALIGN_LEFT  = 0x8000
} adc_align_t;
```

Enumeration Name	Description
ADC_ALIGN_RIGHT	Stores A/D conversion result in right alignment.
ADC_ALIGN_LEFT	Stores A/D conversion result in left alignment.

2.11.19 Auto Clear (Common in All MCUs)

```
typedef enum e_adc_clear
{
    ADC_CLEAR_AFTER_READ_OFF = 0x0000,
    ADC_CLEAR_AFTER_READ_ON  = 0x0020
} adc_clear_t;
```

Enumeration Name	Description
ADC_CLEAR_AFTER_READ_OFF	A/D data register is not auto cleared.
ADC_CLEAR_AFTER_READ_ON	A/D data register is auto cleared.

2.11.20 A/D Conversion Mode (S12ADb except RX210)

```
typedef enum e_adc_speed
{
    ADC_CONVERT_SPEED_DEFAULT = 0x0000,
    ADC_CONVERT_SPEED_NORM = 0x0000,
    ADC_CONVERT_SPEED_HIGH = 0x0400
} adc_speed_t;
```

Enumeration Name	Description
ADC_CONVERT_SPEED_DEFAULT	Default setting (normal conversion)
ADC_CONVERT_SPEED_NORM	Normal conversion
ADC_CONVERT_SPEED_HIGH	High-speed conversion

2.11.21 A/D Conversion Mode (S12ADE)

```
typedef enum e_adc_speed
{
    ADC_CONVERT_SPEED_DEFAULT = 0,
    ADC_CONVERT_SPEED_HIGH    = 0,
    ADC_CONVERT_CURRENT_LOW   = 1
} adc_speed_t;
```

Enumeration Name	Description
ADC_CONVERT_SPEED_DEFAULT	Default setting (high-speed conversion)
ADC_CONVERT_SPEED_ HIGH	High-speed conversion
ADC_CONVERT_CURRENT_LOW	Low-current conversion

2.11.22 A/D Conversion Mode (S12ADa)

```
typedef enum e_adc_speed
{
    ADC_CONVERT_SPEED_PCLK_DIV8 = 0x00,
    ADC_CONVERT_SPEED_PCLK_DIV4 = 0x01,
    ADC_CONVERT_SPEED_PCLK_DIV2 = 0x02,
    ADC_CONVERT_SPEED_PCLK = 0x03
} adc_speed_t;
```

Enumeration Name	Description
ADC_CONVERT_SPEED_PCLK_DIV8	Selects PCLK/8 as the A/D conversion clock.
ADC_CONVERT_SPEED_PCLK_DIV4	Selects PCLK/4 as the A/D conversion clock.
ADC_CONVERT_SPEED_PCLK_DIV2	Selects PCLK/2 as the A/D conversion clock.
ADC_CONVERT_SPEED_PCLK	Selects PCLK as the A/D conversion clock.

2.11.23 Commands Used in the R_ADC_Control Function

Commands used in the R_ADC_Control function vary depending on the MCU. The following shows the definitions to specify triggers in RX110.

```
typedef enum e_adc_cmd
{
    // Commands for special hardware configurations
    ADC_CMD_SET_SAMPLE_STATE_CNT,

    // Commands to enable channels or sensors
    ADC_CMD_ENABLE_CHANS,
    ADC_CMD_ENABLE_TEMP_SENSOR,
    ADC_CMD_ENABLE_VOLT_SENSOR,

    // Commands to enable hardware triggers or cause software trigger
    ADC_CMD_ENABLE_TRIG,
    ADC_CMD_SCAN_NOW,

    // Commands to poll for scan completion
    ADC_CMD_CHECK_SCAN_DONE,
    ADC_CMD_CHECK_SCAN_DONE_GROUPA,
    ADC_CMD_CHECK_SCAN_DONE_GROUPB,

    // Advanced control commands
    ADC_CMD_DISABLE_TRIG,
    ADC_CMD_DISABLE_INT,
    ADC_CMD_ENABLE_INT,
    ADC_CMD_DISABLE_INT_GROUPB,
    ADC_CMD_ENABLE_INT_GROUPB,
} adc_cmd_t;
```

The argument specified for the third parameter (p_args) varies depending on the command used. The table below lists the commands and the available MCUs for each command. For the command which does not use the third parameter of the R_ADC_Control() function, set FIT_NO_PTR for the parameter.

Enumeration Name	Description	Available MCU
ADC_CMD_USE_INT_VOLT_AS_HVREF	Uses the internal reference voltage as the high-side reference voltage. Parameter not used.	RX113
ADC_CMD_USE_VREFL0	Uses VREFL0 as the low-side reference voltage. Parameter not used.	RX130, RX230, RX231
ADC_CMD_USE_VREFH0	Uses VREFH0 as the high-side reference voltage. Parameter not used.	RX130, RX230, RX231
ADC_CMD_SET_DDA_STATE_CNT	Configures the disconnection detection function. Set the disconnection detection configuration structure (adc_dda_t) for the parameter.	RX130, RX210, RX230, RX231, RX64M, RX65N, RX71M
ADC_CMD_SET_SAMPLE_STATE_CNT	Changes the A/D sampling state. Set the sampling state configuration structure (adc_time_t or adc_sst_t) for the parameter.	All MCUs

Enumeration Name	Description	Available MCU
ADC_CMD_ENABLE_CHANS	Configures channels used for A/D conversion. Set the conversion channel configuration structure (adc_ch_cfg_t) for the parameter.	All MCUs
ADC_CMD_ENABLE_TEMP_ SENSOR	Enables the temperature sensor. Parameter not used.	RX110, RX111, RX113, RX130, RX210, RX230, RX231, RX63N
ADC_CMD_ENABLE_VOLT_ SENSOR	Enables the internal reference voltage sensor. Parameter not used.	RX110, RX111, RX113, RX130, RX210, RX230, RX231, RX63N
ADC_CMD_EN_ COMPARATOR_LEVEL	Compare function is used with the window function disabled (threshold compare). Set the compare function configuration structure (adc_cmpwin_t) for the parameter.	RX130, RX230, RX231, RX64M, RX65N, RX71M
ADC_CMD_EN_ COMPARATOR_WINDOW	Compare function is used with the window function enabled (range compare). Set the compare function configuration structure (adc_cmpwin_t) for the parameter.	RX130, RX230, RX231, RX64M, RX65N, RX71M
ADC_CMD_COMP_COMB_ STATUS	Obtains the comparison result for window A/B complex conditions. Set the pointer to the combination result monitoring variable (adc_comp_stat_t) for the parameter.	RX65N
ADC_CMD_ENABLE_TRIG	Enables to start A/D conversion with sync/async trigger. Parameter not used.	All MCUs
ADC_CMD_SCAN_NOW	Starts A/D conversion with software trigger. Parameter not used.	All MCUs
ADC_CMD_CHECK_SCAN_ DONE	Checks for completion of A/D conversion in single scan mode. Parameter not used.	All MCUs
ADC_CMD_CHECK_SCAN_ DONE_GROUPA	Checks for completion of A/D conversion for group A in group scan mode. Parameter not used.	RX110, RX111, RX113, RX130, RX210, RX230, RX231, RX64M, RX65N, RX71M
ADC_CMD_CHECK_SCAN_ DONE_GROUPB	Checks for completion of A/D conversion for group B in group scan mode. Parameter not used.	RX110, RX111, RX113, RX130, RX210, RX230, RX231, RX64M, RX65N, RX71M
ADC_CMD_CHECK_SCAN_ DONE_GROUPC	Checks for completion of A/D conversion for group C in group scan mode. Parameter not used.	RX65N

Enumeration Name	Description	Available MCU
ADC_CMD_CHECK_ CONDITION_MET	Obtains the comparison result of the compare function. [1] For the parameter, set the pointer to the uint32_t variable which stores the comparison result.	RX130, RX230, RX231, RX64M, RX65N, RX71M
ADC_CMD_CHECK_ CONDITION_METB	Obtains the comparison result of the compare function for group B. [1] For the parameter, set the pointer to the uint32_t variable which stores the comparison result.	RX65N
ADC_CMD_DISABLE_TRIG	Disables to start A/D conversion with sync/async trigger. Parameter not used.	All MCUs
ADC_CMD_DISABLE_INT	Disables the S12ADI interrupt. Parameter not used.	All MCUs
ADC_CMD_ENABLE_INT	Enables the S12ADI interrupt. Parameter not used.	All MCUs
ADC_CMD_DISABLE_INT_ GROUPB	Disables the GBADI interrupt. Parameter not used.	RX110, RX111, RX113, RX130, RX210, RX230, RX231, RX64M, RX65N, RX71M
ADC_CMD_ENABLE_INT_ GROUPB	Enables the GBADI/S12GBADI interrupt. Parameter not used.	RX110, RX111, RX113, RX130, RX210, RX230, RX231, RX64M, RX65N, RX71M
ADC_CMD_DISABLE_INT_ GROUPC	Disables the S12GCADI interrupt. Parameter not used.	RX65N
ADC_CMD_ENABLE_INT_ GROUPC	Enables the S12GCADI interrupt. Parameter not used.	RX65N

Note 1. The comparison result is reset to 0 (comparison condition is not satisfied) after this command is executed. Therefore, execute this command once each time A/D conversion is complete.

2.11.24 Disconnection Detection Configuration Structure (S12ADC, S12ADE, S12ADFa)

```
typedef struct st_adc_dda
{
    adc_charge_t    method;
    uint8_t         num_states;
} adc_dda_t;
```

Member	Description
method	Configures the setting for the disconnection detection assist function (discharge/precharge).
num_states	Specifies the discharge/precharge period. Valid values are 0, and 2 to 15. Setting to 0 disables the disconnection detection assist function.

2.11.25 Disconnection Detection Configuration Structure (RX210)

```
typedef struct st_adc_dda
{
    adc_charge_t    method;
    uint8_t         num_states;
} adc_dda_t;
```

Member	Description
method	Configures the setting for the disconnection detection assist function (discharge/precharge).
num_states	Specifies the discharge/precharge period. Valid values are 0 to 15. Setting to 0 disables the disconnection detection assist function.

2.11.26 Sampling State Configuration Structure (S12ADb except RX210)

```
typedef struct st_adc_time
{
    adc_sst_reg_t   reg_id;
    uint8_t         num_states;
} adc_time_t;
```

Member	Description
reg_id	Selects the channel to set the sampling state.
num_states	Specifies the sampling time. Valid values are 6 to 255.

2.11.27 Sampling State Configuration Structure (RX210)

```
typedef struct st_adc_time
{
    adc_sst_reg_t   reg_id;
    uint8_t         num_states;
} adc_time_t;
```

Member	Description
reg_id	Selects the channel to set the sampling state.
num_states	Specifies the sampling time. Valid values are 12 to 255.

2.11.28 Sampling State Configuration Structure (S12ADa)

```
typedef struct st_adc_time
{
    adc_sst_reg_t   reg_id;
    uint8_t         num_states;
} adc_time_t;
```

Member	Description
reg_id	Selects the channel to set the sampling state.
num_states	Specifies the sampling time. Valid values are 10 to 255.

2.11.29 Sampling State Configuration Structure (S12ADC, S12ADE, S12ADFa)

```
typedef struct st_adc_time
{
    adc_sst_reg_t   reg_id;
    uint8_t         num_states;
} adc_sst_t;
```

Member	Description
reg_id	Selects the channel to set the sampling state.
num_states	Specifies the sampling time. Valid values are 5 to 255. (1)

Note 1. When PCLK to ADCLK frequency ratio is 1:2 or 1:4 in S12ADE, set a value greater than 6.

2.11.30 Conversion Channel Configuration Structure (S12ADb except RX210)

```
typedef struct st_adc_ch_cfg            // bit 0 = ch0; bit 15 = ch15
{
    uint32_t        chan_mask;
    uint32_t        chan_mask_groupb;
    uint32_t        add_mask;
} adc_ch_cfg_t;
```

Member	Description
chan_mask (1)	Selects channels to be used. Example when selecting channel 1 and channel 3: (ADC_MASK_CH1 | ADC_MASK_CH3)
chan_mask_groupb (1)	Selects channels to be used for group B. When not using group B, set 'ADC_MASK_GROUPB_OFF'. Example when selecting channel 4 and channel 5: (ADC_MASK_CH4 | ADC_MASK_CH5)
add_mask (1)	Selects channels to be used for addition mode. When not using addition mode, set 'ADC_MASK_ADD_OFF'. When using addition mode, select a channel from channels specified with chan_mask.

Note 1. Use 'ADC_MASK_CHn' (n = channel number) to specify channels.

2.11.31 Conversion Channel Configuration Structure (RX210)

```
typedef struct st_adc_ch_cfg
{
    uint32_t        chan_mask;
    uint32_t        chan_mask_groupb;
    uint32_t        add_mask;
    adc_diag_t      diag_method;
    uint8_t         sample_hold_mask;
    uint8_t         sample_hold_states;
} adc_ch_cfg_t;
```

Member	Description
chan_mask [(1)]	Selects channels to be used. Example when selecting channel 1 and channel 3: (ADC_MASK_CH1 | ADC_MASK_CH3)
chan_mask_groupb [(1)]	Selects channels to be used for group B. When not using group B, set 'ADC_MASK_GROUPB_OFF'. Example when selecting channel 4 and channel 5: (ADC_MASK_CH4 | ADC_MASK_CH5)
add_mask [(1)]	Selects channels to be used for addition mode. When not using addition mode, set 'ADC_MASK_ADD_OFF'. When using addition mode, select a channel from channels specified with chan_mask.
diag_method	Configures self-diagnosis.
sample_hold_mask	Specifies channels to be used for the sample-and-hold circuit. Bits 2 to 0 corresponds to channels 2 to 0. 0: Bypass the channel-dedicated sample-and-hold circuits. 1: Use the channel-dedicated sample-and-hold circuits.
sample_hold_states	Specifies the sampling time. Valid values are 4 to 255.

Note 1. To specify channels, use 'ADC_MASK_CHn' (n = channel number) in combination.

2.11.32 Conversion Channel Configuration Structure (S12ADE)

```
typedef struct st_adc_ch_cfg
{
    uint32_t        chan_mask;
    uint32_t        chan_mask_groupb;
    adc_grpa_t      priority_groupa;
    uint32_t        add_mask;
    adc_diag_t      diag_method;
    adc_elc_t       signal_elc;
} adc_ch_cfg_t;
```

Member	Description
chan_mask (1)	Selects channels to be used. Example when selecting channel 1 and channel 3: (ADC_MASK_CH1 | ADC_MASK_CH3)
chan_mask_groupb (1)	Selects channels to be used for group B. When not using group B, set 'ADC_MASK_GROUPB_OFF'. Example when selecting channel 4 and channel 5: (ADC_MASK_CH4 | ADC_MASK_CH5)
priority_groupa	Configures group-A priority control operation.
add_mask (1)	Selects channels to be used for addition mode. When not using addition mode, set 'ADC_MASK_ADD_OFF'. When using addition mode, select a channel from channels specified with chan_mask.
diag_method	Configures self-diagnosis.
signal_elc	Configures the ELC event condition on scan completion.

Note 1. Use 'ADC_MASK_CHn' (n = channel number) to specify channels.
When using the temperature sensor in single scan mode (ADC_MODE_SS_TEMPERATURE), set 'ADC_MASK_TEMP'.
When using the internal reference voltage sensor in single scan mode (ADC_MODE_SS_INT_REF_VOLT), set 'ADC_MASK_VOLT'.

2.11.33 Conversion Channel Configuration Structure (S12ADa)

```
typedef struct st_adc_ch_cfg
{
    uint32_t        chan_mask;
    uint32_t        add_mask;
} adc_ch_cfg_t;
```

Member	Description
chan_mask (1)	Selects channels to be used. Example when selecting channel 1 and channel 3: (ADC_MASK_CH1 | ADC_MASK_CH3)
add_mask (1)	Selects channels to be used for addition mode. When not using addition mode, set 'ADC_MASK_ADD_OFF'. When using addition mode, select a channel from channels specified with chan_mask.

Note 1. Use 'ADC_MASK_CHn' (n = channel number) to specify channels.

2.11.34 Conversion Channel Configuration Structure (S12ADC)

```
typedef struct st_adc_ch_cfg
{
    uint32_t        scan_mask;
    uint32_t        scan_mask_groupb;
    adc_grpa_t      priority_groupa;
    uint32_t        add_mask;
    adc_diag_t      diag_method;
    bool            anex_enable;
    uint8_t         sample_hold_mask;
    uint8_t         sample_hold_states;
} adc_ch_cfg_t;
```

Member	Description
scan_mask [1]	Selects channels to be used. Example when selecting channel 1 and channel 3: (ADC_MASK_CH1 | ADC_MASK_CH3)
scan_mask_groupb [1]	Selects channels to be used for group B. When not using group B, set 'ADC_MASK_GROUPB_OFF'. Example when selecting channel 4 and channel 5: (ADC_MASK_CH4 | ADC_MASK_CH5)
priority_groupa	Configures group-A priority control operation.
add_mask [1]	Selects channels to be used for addition mode. When not using addition mode, set 'ADC_MASK_ADD_OFF'. When using addition mode, select a channel from channels specified with scan_mask.
diag_method	Configures self-diagnosis.
anex_enable	Specifies extended analog input (ANEX1) usage.
sample_hold_mask [1]	Specifies channels to be used for the sample-and-hold circuit. Select a channel used for channel-dedicated sample-and-hold circuit from channel 0 to 2.
sample_hold_states	Specifies the sampling time. Valid values are 4 to 255.

Note 1. To specify channels, use either of 'ADC_MASK_CHn' (n = channel number), 'ADC_MASK_TEMP' (temperature sensor), or 'ADC_MASK_VOLT' (internal reference voltage sensor), or use them in combination.

2.11.35 Conversion Channel Configuration Structure (S12ADFa)

```
typedef struct st_adc_ch_cfg
{
    uint32_t        scan_mask;
    uint32_t        scan_mask_groupb;
    uint32_t        scan_mask_groupc;
    adc_grpa_t      priority_groupa;
    uint32_t        add_mask;
    adc_diag_t      diag_method;
    bool            anex_enable;
    uint8_t         sample_hold_mask;
    uint8_t         sample_hold_states;
} adc_ch_cfg_t;
```

Member	Description
scan_mask [1]	Selects channels to be used. Example when selecting channel 1 and channel 3: (ADC_MASK_CH1 | ADC_MASK_CH3)
scan_mask_groupb [1]	Selects channels to be used for group B. When not using group B, set 'ADC_MASK_GROUPB_OFF'. Example when selecting channel 4 and channel 5: (ADC_MASK_CH4 | ADC_MASK_CH5)
scan_mask_groupc [1]	Selects channels to be used for group C. When not using group C, set 'ADC_MASK_GROUPC_OFF'. Example when selecting channel 8 and channel 9: (ADC_MASK_CH8 | ADC_MASK_CH9)
priority_groupa	Configures group priority control operation.
add_mask	Selects channels to be used for addition mode. When not using addition mode, set 'ADC_MASK_ADD_OFF'. When using addition mode, select a channel from channels specified with scan_mask.
diag_method	Configures self-diagnosis.
anex_enable	Specifies extended analog input (ANEX1) usage.
sample_hold_mask [1]	Specifies channels to be used for the sample-and-hold circuit. Selects a channel used for channel-dedicated sample-and-hold circuit from channel 0 to 2.
sample_hold_states	Specifies the sampling time. Valid values are 4 to 255.

Note 1. To specify channels, use either of 'ADC_MASK_CHn' (n = channel number), 'ADC_MASK_TEMP' (temperature sensor), or 'ADC_MASK_VOLT' (internal reference voltage), or use them in combination.

2.11.36 Compare Function Configuration Structure (S12ADE)

```
typedef struct st_adc_cmpwin_cfg
{
    uint32_t        compare_mask;
    uint32_t        inside_window_mask;

    uint16_t        level_lo;
    uint16_t        level_hi;
    bool            windowa_enable;
} adc_cmpwin_t;
```

Member	Description
compare_mask [1]	Selects channels to be used for the compare function. Example when selecting channel 1 and channel 3: (ADC_MASK_CH1 | ADC_MASK_CH3)
inside_window_mask	Selects the compare condition for each channel. Bit n corresponds to channel n. ● When the window function is disabled (ADC_CMD_EN_COMPARATOR_LEVEL command): 0: Condition met when level_lo > A/D conversion value. 1: Condition met when level_lo < A/D conversion value. ● When the window function is enabled (ADC_CMD_EN_COMPARATOR_WINDOW command): 0: Condition met when A/D conversion value < level_lo or level_hi < A/D conversion value. 1: Condition met when level_lo < A/D conversion value < level_hi.
level_lo [2]	Specifies the lower-side level of window A.
level_hi [2]	Specifies the higher-side level of window A. Valid only when the 'ADC_CMD_EN_COMPARATOR_WINDOW' command is used.
windowa_enable	Enables/disables the compare window A function.

Note 1. Use 'ADC_MASK_CHn' (n = channel number) to specify channels.
When using the temperature sensor in single scan mode (ADC_MODE_SS_TEMPERATURE), set 'ADC_MASK_TEMP'.
When using the internal reference voltage sensor in single scan mode (ADC_MODE_SS_INT_REF_VOLT), set 'ADC_MASK_VOLT

Note 2. The meaning of the setting differs depending on the A/D data register format (flush-right or flush-left) selected or the setting of A/D conversion addition mode. Refer to the User's Manual: Hardware for details.

2.11.37 Compare Function Configuration Structure (S12ADC)

```
typedef struct st_adc_cmpwin_cfg
{
    uint32_t        compare_mask;
    uint32_t        inside_window_mask;

    uint16_t        level_lo;
    uint16_t        level_hi;
    uint8_t         int_priority;
} adc_cmpwin_t;
```

Member	Description
compare_mask [(1)]	Selects channels to be used for the comparison. Example when selecting channel 1 and channel 3: (ADC_MASK_CH1 | ADC_MASK_CH3)
inside_window_mask	Selects the comparison condition for each channel. Bit n corresponds to channel n. ● When the window function is disabled (ADC_CMD_EN_COMPARATOR_LEVEL command): 0: Condition met when level_lo > A/D conversion value. 1: Condition met when level_lo < A/D conversion value. ● When the window function is enabled (ADC_CMD_EN_COMPARATOR_WINDOW command): 0: Condition met when A/D conversion value < level_lo or level_hi < A/D conversion value. 1: Condition met when level_lo < A/D conversion value < level_hi.
level_lo [(2)]	Specifies the lower-side level of window A.
level_hi [(2)]	Specifies the higher-side level of window A. Valid only when the 'ADC_CMD_EN_COMPARATOR_ WINDOW' command is used.
int_priority	Specifies the S12CMPI interrupt priority level (0 to 15). Setting to 0 disables the S12CMPI interrupt.

Note 1. To specify channels, use either of 'ADC_MASK_CHn' (n = channel number), 'ADC_MASK_TEMP' (temperature sensor), or 'ADC_MASK_VOLT' (internal reference voltage sensor), or use them in combination.'.

Note 2. The meaning of the setting differs depending on the A/D data register format (flush-right or flush-left) selected or the setting of A/D conversion addition mode. Refer to the User's Manual: Hardware for details.

2.11.38 Compare Function Configuration Structure (S12ADFa)

```
typedef struct st_adc_cmpwin_cfg
{
    uint32_t        compare_mask;
    uint32_t        compare_maskb;
    uint32_t        inside_window_mask;
    uint32_t        inside_window_maskb;

    uint16_t        level_lo;
    uint16_t        level_lob;
    uint16_t        level_hi;
    uint16_t        level_hib;
    adc_comp_cond_t comp_cond;
    uint8_t         int_priority;

    bool            windowa_enable;
    bool            windowb_enable;
} adc_cmpwin_t;
```

Member	Description
compare_mask [(1)]	Selects channels to be used for the comparison window A. Example when selecting channel 1 and channel 3: (ADC_MASK_CH1 | ADC_MASK_CH3)
compare_maskb [(2)]	Selects channels to be used for the comparison window B. Example when selecting channel 4: ADC_COMP_WINB_CH4
inside_window_mask	Selects the window A comparison condition for each channel. Bit n corresponds to channel n. ● When the window function is disabled (ADC_CMD_EN_COMPARATOR_LEVEL command): 0: Condition met when level_lo > A/D conversion value. 1: Condition met when level_lo < A/D conversion value. ● When the window function is enabled (ADC_CMD_EN_COMPARATOR_WINDOW command): 0: Condition met when A/D conversion value < level_lo or level_hi < A/D conversion value. 1: Condition met when level_lo < A/D conversion value < level_hi.
inside_window_maskb	Selects the window B comparison condition. ● When the window function is disabled (ADC_CMD_EN_COMPARATOR_LEVEL command): • ADC_COMP_WINB_COND_BELOW: Condition met when level_lo > A/D conversion value. • ADC_COMP_WINB_COND_ABOVE: Condition met when level_lo < A/D conversion value. ● When the window function is enabled (ADC_CMD_EN_COMPARATOR_WINDOW command): • ADC_COMP_WINB_COND_BELOW: Condition met when A/D conversion value < level_lo or level_hi < A/D conversion value. • ADC_COMP_WINB_COND_ABOVE: Condition met when level_lo < A/D conversion value < level_hi.

Member	Description
level_lo (3)	Specifies the lower level of window A.
level_lob (3)	Specifies the lower level of window B. Valid only when the ‘ADC_CMD_EN_COMPARATOR_ WINDOW’ command is used.
level_hi (3)	Specifies the upper level of window A.
level_hib (3)	Specifies the upper level of window B. Valid only when the ‘ADC_CMD_EN_COMPARATOR_ WINDOW’ command is used.
comp_cond	Specifies window A/B complex conditions.
int_priority	Specifies the interrupt priority level (0 to 15) for the S12CMPAI and S12CMPBI interrupts. Setting to 0 disables the S12CMPAI and S12CMPBI interrupts.
windowa_enable	Enables or disables the comparison window A.
windowb_enable	Enables or disables the comparison window B.

Note 1. To specify channels, use either of ‘ADC_MASK_CHn’ (n = channel number), ‘ADC_MASK_TEMP’ (temperature sensor), or ‘ADC_MASK_VOLT’ (internal reference voltage sensor), or use them in combination.

Note 2. To specify channels for window B, use either of ‘ADC_COMP_WINB_CHn’ (n = channel number), ‘ADC_COMP_WINB_TEMP’ (temperature sensor), or ‘ADC_COMP_WINB_VOLT’ (internal reference voltage sensor), or use them in combination.

Note 3. The meaning of the setting differs depending on the A/D data register format (flush-right or flush-left) selected or the setting of A/D conversion addition mode. Refer to the User's Manual: Hardware for details.

2.11.39 Window A/B Complex Condition (S12ADFa)

```
typedef enum e_adc_comp_cond
{
    ADC_COND_OR = 0x00,
    ADC_COND_EXOR = 0x01,
    ADC_COND_AND = 0x02,
} adc_comp_cond_t;
```

Enumeration Name	Description
ADC_COND_OR	Window A comparison condition matched OR window B comparison condition matched
ADC_COND_EXOR	Window A comparison condition matched EXOR window B comparison condition matched
ADC_COND_AND	Window A comparison condition matched AND window B comparison condition matched

2.11.40 Window A/B Combination Result Monitoring (S12ADFa)

```
typedef enum e_adc_comp_stat
{
    ADC_COMP_COND_NOTMET = 0x00,
    ADC_COMP_COND_MET    = 0x01
} adc_comp_stat_t;
```

Enumeration Name	Description
ADC_COMP_COND_NOTMET	Window A/B complex condition is not satisfied.
ADC_COMP_COND_MET	Window A/B complex condition is satisfied.

2.11.41 Analog Input Disconnection Detection Function (Discharge/Precharge) (S12ADC, S12ADE, S12ADFa, and RX210)

```
typedef enum e_adc_charge
{
    ADC_DDA_DISCHARGE = 0x00,
    ADC_DDA_PRECHARGE = 0x01,
    ADC_DDA_OFF = 0x02,
} adc_charge_t;
```

Enumeration Name	Description
ADC_DDA_DISCHARGE	Selects discharge.
ADC_DDA_PRECHARGE	Selects precharge.
ADC_DDA_OFF	Disconnection detection function not used.

2.11.42 Sampling State Channel (S12ADb)

Channels that can be specified for sampling state vary depending on the MCU. The following shows the definitions for RX113.

```
typedef enum e_adc_sst_reg
{
    ADC_SST_CH0 = 0,
    ADC_SST_CH1,
    ADC_SST_CH2,
    ADC_SST_CH3,
    ADC_SST_CH4,
    ADC_SST_CH5,
    ADC_SST_CH6,
    ADC_SST_CH7,
    ADC_SST_CH8_TO_15,
    ADC_SST_CH21,
    ADC_SST_TEMPERATURE,
    ADC_SST_VOLTAGE,
    ADC_SST_REG_MAX = ADC_SST_VOLTAGE
} adc_sst_reg_t;
```

Enumeration Name	Description
ADC_SST_CHn	Selects channel n. RX110, RX111: n = 0 to 4, 6 RX113: n = 0 to 7, 21 RX210: n = 0 to 7 Use channels available on the MCU used.
ADC_SST_CH8_TO_15	Selects channels 8 to 15.
ADC_SST_TEMPERATURE	Selects the temperature sensor.
ADC_SST_VOLTAGE	Selects the internal reference voltage.

2.11.43 Sampling State Channel (S12ADa)

```
typedef enum e_adc_sst_reg
{
    ADC_SST_CH0_TO_20,
    ADC_SST_TEMPERATURE,
} adc_sst_reg_t;
```

Enumeration Name	Description
ADC_SST_CH0_TO_20	Selects channels 0 to 20. Use channels available on the MCU used.
ADC_SST_TEMPERATURE	Selects the temperature sensor.

2.11.44 Sampling State Channel (S12ADE)

```
typedef enum e_adc_sst_reg
{
    ADC_SST_CH0 = 0,
    ADC_SST_CH1,
    ADC_SST_CH2,
    ADC_SST_CH3,
    ADC_SST_CH4,
    ADC_SST_CH5,
    ADC_SST_CH6,
    ADC_SST_CH7,
    ADC_SST_CH16_TO_31,
    ADC_SST_TEMPERATURE,
    ADC_SST_VOLTAGE,
    ADC_SST_REG_MAX = ADC_SST_VOLTAGE
} adc_sst_reg_t;
```

Enumeration Name	Description
ADC_SST_CHn	Selects channel n. (n = 0 to 7). Use channels available on the MCU used.
ADC_SST_CH16_TO_31	Selects channels 16 to 31.
ADC_SST_TEMPERATURE	Selects the temperature sensor.
ADC_SST_VOLTAGE	Selects the internal reference voltage.

2.11.45 Sampling State Channel (S12ADC)

```
typedef enum e_adc_sst_reg
{
    ADC_SST_CH0 = 0,
    ADC_SST_CH1,
    ADC_SST_CH2,
    ADC_SST_CH3,
    ADC_SST_CH4,
    ADC_SST_CH5,
    ADC_SST_CH6,
    ADC_SST_CH7,
    ADC_SST_CH8_TO_20,
    ADC_SST_TEMPERATURE,
    ADC_SST_VOLTAGE,
    ADC_SST_REG_MAX = ADC_SST_VOLTAGE
} adc_sst_reg_t;
```

Enumeration Name	Description
ADC_SST_CHn	Selects channel n. (n = 0 to 7). Use channels available on the MCU used.
ADC_SST_CH8_TO_20	Selects channels 8 to 20. (Only unit 1 can be used.)
ADC_SST_TEMPERATURE	Selects the temperature sensor.
ADC_SST_VOLTAGE	Selects the internal reference voltage.

2.11.46 Sampling State Channel (S12ADFa)

```
typedef enum e_adc_sst_reg
{
    ADC_SST_CH0 = 0,
    ADC_SST_CH1,
    ADC_SST_CH2,
    ADC_SST_CH3,
    ADC_SST_CH4,
    ADC_SST_CH5,
    ADC_SST_CH6,
    ADC_SST_CH7,
    ADC_SST_CH8,
    ADC_SST_CH9,
    ADC_SST_CH10,
    ADC_SST_CH11,
    ADC_SST_CH12,
    ADC_SST_CH13,
    ADC_SST_CH14,
    ADC_SST_CH15,
    ADC_SST_CH16_TO_20,
    ADC_SST_TEMPERATURE,
    ADC_SST_VOLTAGE,
    ADC_SST_REG_MAX
} adc_sst_reg_t;
```

Enumeration Name	Description
ADC_SST_CHn	Selects channel n. (n = 0 to 15). Use channels available on the MCU used. For unit 1, channels 0 to 7 can be selected.
ADC_SST_CH16_TO_20	Selects channels 16 to 20. (Only unit 1 can be used.)
ADC_SST_TEMPERATURE	Selects the temperature sensor.
ADC_SST_VOLTAGE	Selects the internal reference voltage.

2.11.47 Group-A Priority Control Operation (S12ADC, S12ADE)

```
typedef enum e_adc_grpa
{
    ADC_GRPA_PRIORITY_OFF = 0,
    ADC_GRPA_GRPB_WAIT_TRIG = 1,
    ADC_GRPA_GRPB_RESTART_SCAN = 3,
    ADC_GRPA_GRPB_CONT_SCAN= 0x8001,
} adc_grpa_t;
```

Enumeration Name	Description
ADC_GRPA_PRIORITY_OFF	Operation is without group-A priority control.
ADC_GRPA_GRPB_WAIT_TRIG	Scanning for group B is not restarted after having been discontinued due to group-A priority control.
ADC_GRPA_GRPB_RESTART_SCAN [(1)]	Scanning for group B is restarted after having been discontinued due to group-A priority control.
ADC_GRPA_GRPB_CONT_SCAN	Single scan for group B is continuously activated. (If scanning for group A is requested, group A has a priority.)

Note 1. With this setting, the frequency ratio of peripheral module clock PCLK to A/D conversion clock ADCLK should be set to 1:1.

2.11.48 Group Priority Control Operation (S12ADFa)

```
typedef enum e_adc_grpa
{
    ADC_GRPA_PRIORITY_OFF = 0,
    ADC_GRPA_GRPB_GRPC_WAIT_TRIG = 1,
    ADC_GRPA_GRPB_GRPC_TOP_RESTART_SCAN = 3,
    ADC_GRPA_GRPB_GRPC_RESTART_TOP_CONT_SCAN = 0x8003,
    ADC_GRPA_GRPB_GRPC_RESTART_SCAN = 0x4003,
    ADC_GRPA_GRPB_GRPC_TOP_CONT_SCAN = 0x8001,
    ADC_GRPA_GRPB_GRPC_RESTART_CONT_SCAN = 0xC003,
} adc_grpa_t;
```

Enumeration Name	Description
ADC_GRPA_PRIORITY_OFF	Operation is without group priority control.
ADC_GRPA_GRPB_GRPC_WAIT_TRIG	Scanning for the low-priority group is not restarted after having been discontinued due to group priority control.
ADC_GRPA_GRPB_GRPC_TOP_RESTART_SCAN	Scanning for the low-priority group is restarted from the scan start channel after having been discontinued due to group priority control.
ADC_GRPA_GRPB_GRPC_RESTART_TOP_CONT_SCAN [(1)]	Single scan for the lowest-priority group is continuously activated. Scanning for the low-priority group is restarted from the scan start channel after having been discontinued due to group priority control.
ADC_GRPA_GRPB_GRPC_RESTART_SCAN	Scanning for the lowest-priority group is restarted from the channel on which A/D conversion is not completed after having been discontinued due to group priority control.
ADC_GRPA_GRPB_GRPC_TOP_CONT_SCAN [(1)]	Single scan for the lowest-priority group is continuously activated. Scanning for the low-priority group is not restarted after having been discontinued due to group priority control.
ADC_GRPA_GRPB_GRPC_RESTART_CONT_SCAN [(1)]	Single scan for the lowest-priority group is continuously activated. Scanning for the lowest-priority group is restarted from the channel on which A/D conversion is not completed after having been discontinued due to group priority control.

Note 1. With this setting, the frequency ratio of peripheral module clock PCLK to A/D conversion clock ADCLK should be set to 1:1.

2.11.49 Self-Diagnosis Mode (S12ADC, S12ADE, S12ADFa)

```
typedef enum e_adc_diag
{
    ADC_DIAG_OFF = 0x00,
    ADC_DIAG_0_VOLT = 0x01,
    ADC_DIAG_HALF_VREFH0 = 0x02,
    ADC_DIAG_VREFH0 = 0x03,
    ADC_DIAG_ROTATE_VOLTS = 0x04
} adc_diag_t;
```

Enumeration Name	Description
ADC_DIAG_OFF	Self-diagnosis is not executed.
ADC_DIAG_0_VOLT	Performs self-diagnosis using the voltage of 0 V.
ADC_DIAG_HALF_VREFH0	Performs self-diagnosis using the voltage of reference power supply × 1/2.
ADC_DIAG_VREFH0	Performs self-diagnosis using the voltage of reference power supply.
ADC_DIAG_ROTATE_VOLTS	Self-diagnosis rotation mode is used.

2.11.50 Channel Definitions in the R_ADC_Read Function (S12ADb except RX210)

```
typedef enum e_adc_reg
{
    ADC_REG_CH0 = 0,
    ADC_REG_CH1 = 1,
    ADC_REG_CH2 = 2,
    ADC_REG_CH3 = 3,
    ADC_REG_CH4 = 4,
    ADC_REG_CH6 = 6,
    ADC_REG_CH8 = 8,
    ADC_REG_CH9 = 9,
    ADC_REG_CH10 = 10,
    ADC_REG_CH11 = 11,
    ADC_REG_CH12 = 12,
    ADC_REG_CH13 = 13,
    ADC_REG_CH14 = 14,
    ADC_REG_CH15 = 15,
    ADC_REG_TEMP = 16,
    ADC_REG_VOLT = 17,
    ADC_REG_DBLTRIG = 18,
    ADC_REG_MAX = ADC_REG_DBLTRIG
} adc_reg_t;
```

Enumeration Name	Description
ADC_REG_CHn [(1)]	Specifies the A/D conversion value of channel n.
ADC_REG_TEMP	Specifies the A/D conversion value of the temperature sensor.
ADC_REG_VOLT	Specifies the A/D conversion value of the internal reference voltage sensor.
ADC_REG_DBLTRIG	Specifies the A/D conversion value in double trigger mode.

Note 1. Available channels vary depending on the MCU or the number of pins on the MCU. The above shows the definitions in RX110.

2.11.51 Channel Definitions in the R_ADC_Read Function (S12ADE, RX210)

```
typedef enum e_adc_reg
{
    ADC_REG_CH0  = 0,
    ADC_REG_CH1,
    ADC_REG_CH2,
    ADC_REG_CH3,
    ADC_REG_CH4,
    ADC_REG_CH5,
    ADC_REG_CH6,
    ADC_REG_CH7,
    ADC_REG_CH16,
    ADC_REG_CH17,
    ADC_REG_CH18,
    ADC_REG_CH19,
    ADC_REG_CH20,
    ADC_REG_CH21,
    ADC_REG_CH24,
    ADC_REG_CH25,
    ADC_REG_CH26,
    ADC_REG_TEMP,
    ADC_REG_VOLT,
    ADC_REG_DBLTRIG,
    ADC_REG_SELF_DIAG,
    ADC_REG_MAX = ADC_REG_SELF_DIAG
} adc_reg_t;
```

Enumeration Name	Description
ADC_REG_CHn [1]	Specifies the A/D conversion value of channel n.
ADC_REG_TEMP	Specifies the A/D conversion value of the temperature sensor.
ADC_REG_VOLT	Specifies the A/D conversion value of the internal reference voltage sensor.
ADC_REG_DBLTRIG	Specifies the A/D conversion value in double trigger mode.
ADC_REG_SELF_DIAG	Specifies the A/D conversion value for self-diagnosis.

Note 1. Available channels vary depending on the MCU or the number of pins on the MCU. The above shows the definitions in RX130.

2.11.52 Channel Definitions in the R_ADC_Read Function (S12ADa)

```
typedef enum e_adc_reg
{
    ADC_REG_CH0 = 0,
    ADC_REG_CH1,
    ADC_REG_CH2,
    ADC_REG_CH3,
    ADC_REG_CH4,
    ADC_REG_CH5,
    ADC_REG_CH6,
    ADC_REG_CH7,
    ADC_REG_CH8,
    ADC_REG_CH9,
    ADC_REG_CH10,
    ADC_REG_CH11,
    ADC_REG_CH12,
    ADC_REG_CH13,
    ADC_REG_CH14,
    ADC_REG_CH15,
    ADC_REG_CH16,
    ADC_REG_CH17,
    ADC_REG_CH18,
    ADC_REG_CH19,
    ADC_REG_CH20,
    ADC_REG_TEMP,
    ADC_REG_VOLT,
    ADC_REG_MAX = ADC_REG_VOLT
} adc_reg_t;
```

Enumeration Name	Description
ADC_REG_CHn [1]	Specifies the A/D conversion value of channel n.
ADC_REG_TEMP	Specifies the A/D conversion value of the temperature sensor.
ADC_REG_VOLT	Specifies the A/D conversion value of the internal reference voltage sensor.

Note 1. Available channels vary depending on the MCU or the number of pins on the MCU. The above shows the definitions in RX63N.

2.11.53 Channel Definitions in the R_ADC_Read Function (S12ADC, S12ADFa)

```
typedef enum e_adc_reg
{
    ADC_REG_CH0 = 0,
    ADC_REG_CH1 = 1,
    ADC_REG_CH2 = 2,
    ADC_REG_CH3 = 3,
    ADC_REG_CH4 = 4,
    ADC_REG_CH5 = 5,
    ADC_REG_CH6 = 6,
    ADC_REG_CH7 = 7,
    ADC_REG_CH8 = 8,
    ADC_REG_CH9 = 9,
    ADC_REG_CH10 = 10,
    ADC_REG_CH11 = 11,
    ADC_REG_CH12 = 12,
    ADC_REG_CH13 = 13,
    ADC_REG_CH14 = 14,
    ADC_REG_CH15 = 15,
    ADC_REG_CH16 = 16,
    ADC_REG_CH17 = 17,
    ADC_REG_CH18 = 18,
    ADC_REG_CH19 = 19,
    ADC_REG_CH20 = 20,
    ADC_REG_TEMP,
    ADC_REG_VOLT,
    // unit 0 and unit 1
    ADC_REG_DBLTRIG,
    ADC_REG_DBLTRIGA,
    ADC_REG_DBLTRIGB,
    ADC_REG_SELF_DIAG,
    ADC_REG_MAX = ADC_REG_SELF_DIAG
} adc_reg_t;
```

Enumeration Name	Description
ADC_REG_CHn (1)	Specifies the A/D conversion value of channel n.
ADC_REG_TEMP	Specifies the A/D conversion value of the temperature sensor.
ADC_REG_VOLT	Specifies the A/D conversion value of the internal reference voltage sensor.
ADC_REG_DBLTRIG	Specifies the A/D conversion value in double trigger mode.
ADC_REG_DBLTRIGA	Specifies the A/D conversion value (ADDBLDRA register) in extended double trigger mode.
ADC_REG_DBLTRIGB	Specifies the A/D conversion value (ADDBLDRB register) in extended double trigger mode.
ADC_REG_SELF_DIAG	Specifies the A/D conversion value for self-diagnosis.

Note 1. Available channels vary depending on the MCU or the number of pins on the MCU. The above shows the definitions in RX64M.

2.11.54 A/D Conversion Result Storage Structure in the R_ADC_ReadAll Function (S12ADb except RX210)

```
typedef struct st_adc_data
{
    uint16_t    chan[ADC_REG_ARRAY_MAX];
    uint16_t    temp;
    uint16_t    volt;
    uint16_t    dbltrig;
} adc_data_t;
```

Member	Description
chan[ADC_REG_ARRAY_MAX]	A/D conversion result of each channel [1].
temp	A/D conversion result of the temperature sensor.
volt	A/D conversion result of the internal reference voltage.
dbltrig	A/D conversion result in double trigger mode.

Note 1. Specify the channel with ADC_REG_CHn (n = channel number).

2.11.55 A/D Conversion Result Storage Structure in the R_ADC_ReadAll Function (S12ADE, RX210)

```
typedef struct st_adc_data
{
    uint16_t    chan[ADC_REG_ARRAY_MAX];
    uint16_t    temp;
    uint16_t    volt;
    uint16_t    dbltrig;
    uint16_t    self_diag;
} adc_data_t;
```

Member	Description
chan[ADC_REG_ARRAY_MAX]	A/D conversion result of each channel [1].
temp	A/D conversion result of the temperature sensor.
volt	A/D conversion result of the internal reference voltage.
dbltrig	A/D conversion result in double trigger mode.
self_diag	A/D conversion result for self-diagnosis.

Note 1. Specify the channel with ADC_REG_CHn (n = channel number).

2.11.56 A/D Conversion Result Storage Structure in the R_ADC_ReadAll Function (S12ADa)

```
typedef struct st_adc_data
{
    uint16_t    chan[ADC_REG_ARRAY_MAX];
    uint16_t    temp;
    uint16_t    volt;
} adc_data_t;
```

Member	Description
chan[ADC_REG_ARRAY_MAX]	A/D conversion result of each channel [1].
temp	A/D conversion result of the temperature sensor.
volt	A/D conversion result of the internal reference voltage.

Note 1. Specify the channel with ADC_REG_CHn (n = channel number).

2.11.57 A/D Conversion Result Storage Structure in the R_ADC_ReadAll Function (S12ADC, S12ADFa)

```
typedef struct st_adc_data
{
    adc_unit0_data_t   unit0;
    adc_unit1_data_t   unit1;
} adc_data_t;
```

Member	Description
unit0	A/D conversion result storage structure for unit 0
unit1	A/D conversion result storage structure for unit 1

2.11.58 A/D Conversion Result Storage Structure for Unit 0 (S12ADC, S12ADFa)

```
typedef struct st_adc_unit0_data
{
    uint16_t    chan[ADC_0_REG_ARRAY_MAX];
    uint16_t    dbltrig;
    uint16_t    dbltrigA;
    uint16_t    dbltrigB;
    uint16_t    self_diag;
} adc_unit0_data_t;
```

Member	Description
chan[ADC_REG_ARRAY_MAX]	A/D conversion result of each channel (1).
dbltrig	A/D conversion result in double trigger mode.
dbltrigA	A/D conversion result (ADDBLDRA register) in extended double trigger mode.
dbltrigB	A/D conversion result (ADDBLDRB register) in extended double trigger mode.
self_diag	A/D conversion result for self-diagnosis

Note 1. Specify the channel with ADC_REG_CHn (n = channel number).

2.11.59 A/D Conversion Result Storage Structure for Unit 1 (S12ADC, S12ADFa)

```
typedef struct st_adc_unit1_data
{
    uint16_t    chan[ADC_1_REG_ARRAY_MAX];
    uint16_t    temp;
    uint16_t    volt;
    uint16_t    dbltrig;
    uint16_t    dbltrigA;
    uint16_t    dbltrigB;
    uint16_t    self_diag;
} adc_unit1_data_t;
```

Member	Description
chan[ADC_REG_ARRAY_MAX]	A/D conversion result of each channel (1).
temp	A/D conversion result of the temperature sensor.
volt	A/D conversion result of the internal reference voltage.
dbltrig	A/D conversion result in double trigger mode.
dbltrigA	A/D conversion result (ADDBLDRA register) in extended double trigger mode.
dbltrigB	A/D conversion result (ADDBLDRB register) in extended double trigger mode.
self_diag	A/D conversion result for self-diagnosis

Note 1. Specify the channel with ADC_REG_CHn (n = channel number).

2.12 Return Values

These are the different error codes API functions can return. The enum is found in r_s12ad_rx_if.h along with the API function declarations.

```
typedef enum e_adc_err          // ADC API error codes
{
    ADC_SUCCESS = 0,
    ADC_ERR_AD_LOCKED,          // Open() call is in progress elsewhere
    ADC_ERR_AD_NOT_CLOSED,      // peripheral still running in another mode
    ADC_ERR_MISSING_PTR,        // missing required pointer argument
    ADC_ERR_INVALID_ARG,        // argument is not valid for parameter
    ADC_ERR_ILLEGAL_ARG,        // argument is illegal for mode
    ADC_ERR_SCAN_NOT_DONE,      // default, Group A, or Group B scan not done
    ADC_ERR_TRIG_ENABLED,       // scan running, cannot configure comparator
    ADC_ERR_CONDITION_NOT_MET,  // no chans/sensors passed comparator condition
    ADC_ERR_UNKNOWN             // unknown hardware error
} adc_err_t;
```

2.13 Adding the FIT Module to Your Project

This module must be added to each project in which it is used. Renesas recommends using “Smart Configurator” described in (1) or (3). However, “Smart Configurator” only supports some RX devices. Please use the methods of (2) or (4) for unsupported RX devices.

(1) Adding the FIT module to your project using “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 “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 “Smart Configurator” on 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.

3. API Functions

3.1 Summary

The following functions are included in this design:

Function	Description
R_ADC_Open()	Applies power to the A/D peripheral (and TEMPS peripheral on RX210 and RX63x), sets the operational mode, trigger sources, interrupt priority, and configurations common to all channels and sensors. Optionally takes a callback function pointer for notifying the user at interrupt level whenever a scan has completed or a comparator condition is met.
R_ADC_Control()	Provides commands for enabling channels and sensors, and for runtime operations. These include enabling/disabling trigger sources and interrupts, initiating a software trigger, and checking for scan completion.
R_ADC_Read()	Reads conversion results from a single channel, sensor, double trigger, or self-diagnosis register.
R_ADC_ReadAll()	Reads conversion results from all potential channel sources, enabled or not.
R_ADC_Close()	Ends any scan in progress, disables interrupts, and removes power to the A/D peripheral.
R_ADC_GetVersion()	Returns at runtime the driver version number.

3.2 R_ADC_Open()

This function applies power to the A/D peripheral, sets the operational mode, trigger sources, interrupt priority, callback function, and configurations common to all channels and sensors. If interrupt priority is non-zero, the function takes a callback function pointer for notifying the user at interrupt level whenever a scan has completed or a comparator condition is met. The function initializes the ADC FIT module. This function must be called before calling any other API functions.

Format

```
adc_err_t  R_ADC_Open(uint8_t                unit,
                      adc_mode_t const       mode,
                      adc_cfg_t * const      p_cfg,
                      void                   (* const  p_callback)(void *p_args));
```

Parameters

unit

0 or 1. For MCUs with only one unit, 0 should be passed (only the RX64M/RX71M/RX65x have 2 units).

mode

Operational mode. See 2.11.3 “S12AD Operation Mode (S12ADb, A12ADE)” to 2.11.6 “S12AD Operation Mode (S12ADFa)” for details on operational modes.

mode indicates the type of scan to be performed. For ADC_MODE_SS_MULTI_CH_ GROUPED_ DBLTRIG_A or ADC_MODE_SS_MULTI_CH_GROUPED_DBLTRIG_A_GROUPC, only one channel can be specified for group A.

p_cfg

Pointer to function configuration structure. See 2.11.7 “S12AD Function Configuration Structure (S12ADb except RX210)” to 2.11.12 “S12AD Function Configuration Structure (S12ADFa)” for details on function configuration structures.

p_callback

Optional pointer to function called from interrupt when a scan completes or a comparator condition is met. When not using this parameter, set ‘FIT_NO_PTR’.

Return Values

ADC_SUCCESS:	*Successful*
ADC_ERR_AD_LOCKED:	*Open() call is in progress elsewhere*
ADC_ERR_AD_NOT_CLOSED:	*Peripheral is still running in another mode; Perform R_ADC_Close() first*
ADC_ERR_INVALID_ARG:	*Element of p_cfg structure has invalid value*
ADC_ERR_ILLEGAL_ARG:	*an argument is illegal based upon mode*
ADC_ERR_MISSING_PTR:	*p_cfg pointer is FIT_NO_PTR/NULL*

Properties

Prototyped in file “r_s12ad_rx_if.h”

Description

Applies power to the A/D peripheral, sets the operational mode, trigger sources, interrupt priority, and configurations common to all channels and sensors. With a non-zero interrupt priority (interrupt usage), a callback function is called by the interrupts whenever a scan has completed or a comparator condition is met. When setting the interrupt priority to 0, a callback function is not called. In this case, poll for scan completion with the R_ADC_ Control() function when necessary.

To set values of parameters used in this function, first clear all members of parameters to 0, and then set values.

Reentrant

No.

Example (S12ADb except RX210)

```
adc_cfg_t   config;

/* Clear all members of the adc_cfg_t structure */
memset(&config, 0, sizeof(config));

    /* INITIALIZE FOR SINGLE SCAN OF TEMPERATURE SENSOR
     *  - use software trigger to start scan; poll for completion
     *  - don't do any summing of conversion values
     *  - keep the data registers aligned right, and clear after reading is off
     *  - use normal speed conversion
     */
    config.trigger = ADC_TRIG_SOFTWARE;
    config.priority = 0;                // denotes polling!
    config.add_cnt = ADC_ADD_OFF;
    config.alignment = ADC_ALIGN_RIGHT;
    config.clearing = ADC_CLEAR_AFTER_READ_OFF;
    config.conv_speed = ADC_CONVERT_SPEED_NORM;

    R_ADC_Open(0, ADC_MODE_SS_TEMPERATURE, &config, FIT_NO_FUNC);
```

Special Notes (RX Family Common):

The application must complete MPC and PORT initialization prior to calling R_ADC_Open(). Refer to the User’s Manual: Hardware about limitations of using output pins on the same port as analog pins. The following is a sample initialization for an RSKRX111 Rev 1 board:

```
    R_BSP_RegisterProtectDisable(BSP_REG_PROTECT_MPC);

      PORT4.PDR.BIT.B0 = 0;   // set direction of A/D conversion port to input
      PORT4.PMR.BIT.B0 = 0;   // set A/D conversion port to general I/O
      MPC.P40PFS.BYTE = 0x80; // set P40 function to A/D conversion port (AN000)

      MPC.PB0PFS.BIT.PSEL = 0x09; // set PB0 function to ADTRIG0
                                  // (SW3 on RSKRX111)
      PORTB.PDR.BIT.B0 = 0;   // set ADTRIG0 pin direction to input
      PORTB.PMR.BIT.B0 = 1;   // set ADTRIG0 pin mode to peripheral

    R_BSP_RegisterProtectEnable(BSP_REG_PROTECT_MPC);
```

The application must set the A/D conversion clock prior to calling R_ADC_Open() function.

To stop A/D conversion which is started in continuous scan mode, call the R_ADC_Close function.

If continuous scan mode is selected, it is recommended not to use the S12ADI interrupt since scan completion occurs continuously. That causes the majority of the processing time to be spent at the interrupt level.

If interrupts are in use, a callback function is required which takes a single argument. This is a pointer to a structure which is cast to a void pointer (provides consistency with other FIT module callback functions). Cast to the adc_cb_args_t pointer in the interrupt handling. See 2.11.1 “Callback Function Events (Common to All MCUs)” for details on ‘adc_cb_args_t’.

An example template for a callback function is provided here:

```
void MyCallback(void *p_args)
{
adc_cb_args_t    *args;


    args = (adc_cb_args_t *)p_args;

    if (args->event == ADC_EVT_SCAN_COMPLETE)
    {
        // Read results here
        nop();
    }
    else if (args->event == ADC_EVT_GROUPB_SCAN_COMPLETE)
    {
        // Read Group B results here
        nop();
    }
    else if (args->event == ADC_EVT_CONDITION_MET)
    {
        // Process chans/sensors indicated in args->compare_flags
        nop();
    }
}
```

Special Notes (S12ADa):

Even if register automatic clearing is enabled only for temperature sensor output and internal reference voltage, the A/D conversion result is not cleared.

After the R_ADC_Open() function is executed, wait at least 10 ms before executing A/D conversion.

Special Notes (S12ADb, A12ADC, S12ADE, S12ADFa):

After the R_ADC_Open() function is executed, wait at least 1 µs before executing A/D conversion.

3.3 R_ADC_Control()

This function configures 12-bit A/D converter functions.

Format

```
adc_err_t   R_ADC_Control(uint8_t            unit,
                          adc_cmd_t const    cmd,
                          void * const       p_args);
```

Parameters

unit

0 or 1. For MCUs with only one unit, 0 should be passed (only the RX64M/RX71M/RX65x have 2 units).

cmd

Command to run. See 2.11.23 “Commands Used in the R_ADC_Control Function” for details on commands and arguments used by the commands.

p_args

Pointer to optional configuration structure. Clear all members of the argument to 0 before setting values to them. If the command requires no argument, set FIT_NO_PTR.

Return Values

ADC_SUCCESS:	*Successful*
ADC_ERR_MISSING_PTR:	*p_args pointer is FIT_NO_PTR/NULL when required as an argument*
ADC_ERR_INVALID_ARG:	*Invalid value is specified to p_args structure*
ADC_ERR_ILLEGAL_ARG:	*cmd is illegal based upon mode*
ADC_ERR_SCAN_NOT_DONE:	*The requested scan has not completed*
ADC_ERR_TRIG_ENABLED:	*Cannot configure comparator because scan still running*
ADC_ERR_CONDITION_NOT_MET:	*No channels/sensors met the comparison condition*

Properties

Prototyped in file “r_s12ad_rx_if.h”

Description

Provides commands for enabling channels and sensors and for runtime operations. These include enabling/disabling trigger sources and interrupts, initiating a software trigger, and checking for scan completion.

After R_ADC_Open() call, the following commands can be issued with R_ADC_Control(). Only required No. of commands should be issued in the order listed in the table.

No.	Command	Description	3rd Parameter (p_args)
1	ADC_CMD_SET_DDA_STATE_CNT	For S12ADC, S12ADE, S12ADFa, RX210. Configures the disconnection detection function. When not using the disconnection detection function, this command does not need to be issued.	Disconnection detection configuration structure (adc_dda_t)
2	ADC_CMD_SET_SAMPLE_STATE_CNT	Specifies the sampling time of an analog input. If the initial value (the ADSSTRn register value after reset) is not changed, this command does not need to be issued.	Sampling state configuration structure (adc_sst_t or adc_time_t)

No.	Command	Description	3rd Parameter (p_args)
3	ADC_CMD_USE_ VREFL0	For S12ADE Sets the high-potential reference voltage to VREFH0. When selecting AVCC0, this command does not need to be issued.	FIT_NO_PTR
4	ADC_CMD_USE_ VREFH0	For S12ADE Sets the low-potential reference voltage to VREFL0. When selecting AVSS0, this command does not need to be issued.	FIT_NO_PTR
5	ADC_CMD_ENABLE_ CHANS	Selects and configures A/D conversion channels.	Conversion channel configuration structure (adc_ch_cfg_t)
6	ADC_CMD_ENABLE_ TEMP_SENSOR	For S12ADa, S12ADb, S12ADE Enables the temperature sensor. When not using the temperature sensor, this command does not need to be issued.	FIT_NO_PTR
7	ADC_CMD_ENABLE_ VOLT_SENSOR	For S12ADa, S12ADb, S12ADE Enables the internal reference voltage sensor. When not using the internal reference voltage sensor, this command does not need to be issued.	FIT_NO_PTR
8	ADC_CMD_EN_ COMPARATOR_LEVEL	For S12ADC, S12ADE, S12ADFa Configures the compare function (window function disabled). When not using the comparison, this command does not need to be issued.	Compare function configuration structure (adc_cmpwin_t)
9	ADC_CMD_EN_COMPARATOR_WINDOW	For S12ADC, S12ADE, S12ADFa Configures the compare function (window function enabled). When not using the comparison, this command does not need to be issued.	Compare function configuration structure (adc_cmpwin_t)
10	ADC_CMD_TRIG_ ENABLE	Enables hardware trigger. When selecting software trigger, this command does not need to be issued.	FIT_NO_PTR
11	ADC_CMD_SCAN_ NOW	Starts A/D conversion with software trigger. When selecting hardware trigger, this command does not need to be issued.	FIT_NO_PTR
12	ADC_CMD_CHECK_ SCAN_DONE	Checks for completion of A/D conversion. Use this command when not using the callback function but polling for completion.	FIT_NO_PTR
13	ADC_CMD_CHECK_SCAN_DONE_GROUPA	For S12ADb, S12ADC, S12ADE, S12ADFa Checks for completion of A/D conversion for group A. Use this command when setting the group A interrupt priority level to 0 and polling for completion.	FIT_NO_PTR
14	ADC_CMD_CHECK_SCAN_DONE_GROUPB	For S12ADb, S12ADC, S12ADE, S12ADFa Checks for completion of A/D conversion for group B. Use this command when setting the group B interrupt priority level to 0 and polling for completion.	FIT_NO_PTR

No.	Command	Description	3rd Parameter (p_cfg)
15	ADC_CMD_CHECK_SCAN_DONE_GROUPC	For S12ADFa Checks for completion of A/D conversion for group C. Use this command when setting the interrupt priority level for group C to 0, and polling for completion.	FIT_NO_PTR
16	ADC_CMD_CHECK_CONDITION_MET	For S12ADC, S12ADE, S12ADFa Obtains the comparison result in the variable specified by the argument. The comparison result for channel n is stored in bit n. (1) 0: Comparison condition is not met. 1: Comparison condition is met.	Pointer to uint32_t variable which stores the comparison result.
17	ADC_CMD_CHECK_CONDITION_METB	For S12ADFa Obtains the comparison result for group B. The result is stored in the variable specified by the argument. (1) 0x0000: Comparison condition is not met. 0x0001: Comparison condition is met.	Pointer to uint32_t variable which stores the comparison result.
18	ADC_CMD_COMP_COMB_STATUS	For S12ADFa Obtains the Window A/B combination result. The result is stored in the variable specified by the argument. ADC_COMP_COND_NOTMET: Window A/B complex condition is not satisfied. ADC_COMP_COND_MET: Window A/B complex condition is satisfied.	Pointer to adc_comp_stat_t variable which stores the Window A/B combination result.

Note 1. After this command is executed, the comparison result is initialized to 0 (comparison condition not met). Therefore execute this command once each time A/D conversion is complete.

Reentrant

No.
However, Yes only when the ADC_CMD_CHECK_SCAN_DONE_GROUPA, ADC_CMD_CHECK_SCAN_DONE_GROUPB, or ADC_CMD_CHECK_SCAN_DONE_GROUPC command is being executed.

Example 1: Single Channel Polling Unit 0 (RX64M, RX71M, RX65x only)

```
uint16_t        data;
adc_cfg_t       config;
adc_ch_cfg_t    ch_cfg;
adc_err_t       err;

/* OPEN ADC */

/* Clear all members of the adc_cfg_t structure */
memset(&config, 0, sizeof(config));


/* Open ADC for software trigger, single scan of one channel, and polling */
config.resolution = ADC_RESOLUTION_12_BIT;
config.trigger    = ADC_TRIG_SOFTWARE;
config.priority   = 0;                            // denotes polling
config.add_cnt    = ADC_ADD_OFF;
config.alignment  = ADC_ALIGN_RIGHT;
config.clearing   = ADC_CLEAR_AFTER_READ_OFF;
err = R_ADC_Open(0,ADC_MODE_SS_ONE_CH, &config, NULL);

/* ENABLE CHANNELS */

/* Clear all members of the adc_ch_cfg_t structure */
memset(&ch_cfg, 0, sizeof(ch_cfg));

/* Specify and enable potentiometer channel on RSKRX64M */
ch_cfg.chan_mask        = ADC_MASK_CH0;
ch_cfg.diag_method      = ADC_DIAG_OFF;
ch_cfg.anex_enable      = false;
ch_cfg.sample_hold_mask = 0;
err = R_ADC_Control(0, ADC_CMD_ENABLE_CHANS, &ch_cfg);

/* After open, wait 1 us or longer before A/D conversion starts */

/* Repeatedly trigger, poll for completion, and read result */
while(1)
{
  /* CAUSE SOFTWARE TRIGGER */
  err = R_ADC_Control(0, ADC_CMD_SCAN_NOW, NULL);

  /* WAIT FOR SCAN TO COMPLETE */
  while (R_ADC_Control(0,ADC_CMD_CHECK_SCAN_DONE,NULL) == ADC_ERR_SCAN_NOT_DONE)
  {
  }

  /* READ RESULT */
  err = R_ADC_Read(0, ADC_REG_CH0, &data);
}
```

Example 2: Temperature Sensor Polling and Set Sample State Count Unit 1 (RX64M, RX71M, RX65x)

```
uint16_t       data;
adc_cfg_t      config;
adc_sst_t      sst;           // sample state
adc_ch_cfg_t   ch_cfg;
adc_err_t      adc_err;

/* OPEN ADC */

/* Clear all members of the adc_cfg_t structure */
memset(&config, 0, sizeof(config));

/* Open ADC for software trigger, single scan temperature sensor, and polling */
config.resolution = ADC_RESOLUTION_10_BIT;
config.trigger  = ADC_TRIG_SOFTWARE;
config.priority = 0;                           // denotes polling
config.add_cnt  = ADC_ADD_OFF;
config.alignment = ADC_ALIGN_RIGHT;
config.clearing  = ADC_CLEAR_AFTER_READ_OFF;
adc_err = R_ADC_Open(1, ADC_MODE_SS_ONE_CH, &config, NULL);


/* DO SPECIAL HARDWARE CONFIGURATION */

/* Clear all members of the adc_sst_t structure */
memset(&sst, 0, sizeof(sst));

/* Clear all members of the adc_ch_cfg_t structure */
memset(&ch_cfg, 0, sizeof(ch_cfg));

/* Set number of sampling states for 4us sample *
/* For PCLKD=60MHz, 1 state = 1/60MHz = 16.7ns, 4us/16.7ns = 240 states */
sst.reg_id = ADC_SST_TEMPERATURE;
sst.num_states = 240;
adc_err = R_ADC_Control(1, ADC_CMD_SET_SAMPLE_STATE_CNT, &sst);

/* CONFIGURE SCAN */
ch_cfg.chan_mask = ADC_MASK_TEMP;
ch_cfg.diag_method = ADC_DIAG_OFF;
ch_cfg.anex_enable = false;
ch_cfg.sample_hold_mask = 0;        // not available on unit 1
adc_err = R_ADC_Control(1, ADC_CMD_ENABLE_CHANS, &ch_cfg);

/* After open, wait 1 us or longer before A/D conversion starts */

/* CAUSE SOFTWARE TRIGGER */
adc_err = R_ADC_Control(1, ADC_CMD_SCAN_NOW, NULL);

/* WAIT FOR SCAN TO COMPLETE */
while (R_ADC_Control(1, ADC_CMD_CHECK_SCAN_DONE, NULL) == ADC_ERR_SCAN_NOT_DONE)
{
}

/* READ RESULT */
adc_err = R_ADC_Read(1, ADC_REG_TEMP, &data);
```

Example 3: Grouped Channels with Interrupt Triggers, Double Trigger on Group A, and Averaging Four Samples (RX64M, RX71M, RX65x)

```
adc_cfg_t      config;
adc_ch_cfg_t   ch_cfg;

/* INITIALIZE MTU HERE (USED FOR TRIGGER SOURCES) */

/* OPEN ADC */

/* Clear all members of each structure */
memset(&config, 0, sizeof(config));
memset(&ch_cfg, 0, sizeof(ch_cfg));

/* INITIALIZE ADC FOR GROUP SCANNING WITH DOUBLE TRIGGER
 *  - use synchronous trigger TRGA0N to start Group A scan; int priority 4
 *  - use synchronous trigger TRG0N to start Group B scan; int priority 5
 *  - allow each channel to be scanned four times and averaged before continuing
 *  - do not clear registers after reading
 */
config.resolution = ADC_RESOLUTION_8_BIT;
config.trigger    = ADC_TRIG_SYNC_TRG0AN;
config.priority   = 4;
config.trigger_groupb = ADC_TRIG_SYNC_TRG0EN;
config.priority_groupb= 5;
config.add_cnt    = ADC_ADD_AVG_4_SAMPLES;
config.alignment  = ADC_ALIGN_RIGHT;
config.clearing   = ADC_CLEAR_AFTER_READ_OFF;
R_ADC_Open(1, ADC_MODE_SS_MULTI_CH_GROUPED_DBLTRIG_A, &config, MyCallback);

/* CONFIGURE SCAN */

/* Can only have 1 channel for double triggering, and is only channel in Group A
   Have channel 8 as Group A, have 2, 3, and 9 as Group B
   Perform addition/average on all channels except 9
*/
ch_cfg.chan_mask = ADC_MASK_CH8;
ch_cfg.chan_mask_groupb = ADC_MASK_CH2 | ADC_MASK_CH3 | ADC_MASK_CH9;
ch_cfg.priority_groupa = ADC_GRPA_PRIORITY_OFF;
ch_cfg.add_mask = ADC_MASK_CH8 | ADC_MASK_CH2 | ADC_MASK_CH3;
ch_cfg.diag_method = ADC_DIAG_OFF;
ch_cfg.anex_enable = false;
ch_cfg.sample_hold_mask = 0;
R_ADC_Control(1, ADC_CMD_ENABLE_CHANS, &ch_cfg);

/* After open, wait 1 us or longer before A/D conversion starts */

/* ENABLE TRIGGERS */
R_ADC_Control(1, ADC_CMD_ENABLE_TRIG, NULL);

/* INTERRUPT OCCURS UPON SCAN COMPLETION */

/* The callback is called twice from interrupt level- once after each
 *  group scan completes. The order depends upon the trigger order.
 */
void MyCallback(void *p_args)
{
adc_cb_args_t  *args;
uint16_t        dbltrg,data2,data3,data8,data9;
```

```
    args = (adc_cb_args_t *)p_args;

    /* READ RESULTS */

    if (args->event == ADC_EVT_SCAN_COMPLETE)
    {
        /* From S12ADIO interrupt, Group A scan complete, read registers */
        R_ADC_Read(1, ADC_REG_CH8, &data8);
        R_ADC_Read(1, ADC_REG_DBLTRIG, &dbltrg);
    }
    else if (args->event == ADC_EVT_SCAN_COMPLETE_GROUPB)
    {
        /* From GBADI interrupt, Group B scan complete, read registers */
        R_ADC_Read(1, ADC_REG_CH2, &data2);
        R_ADC_Read(1, ADC_REG_CH3, &data3);
        R_ADC_Read(1, ADC_REG_CH9, &data9);
    }

    /* process data, or set flag for application level to do so */
}
```

Example 4: Grouped Channels with Interrupt Triggers (RX65x)

```
adc_cfg_t      config;
adc_ch_cfg_t   ch_cfg;

/* INITIALIZE MTU HERE (USED FOR TRIGGER SOURCES) */

/* OPEN ADC */

/* Clear all members of each structure */
memset(&config, 0, sizeof(config));

/* INITIALIZE ADC FOR GROUP SCANNING WITH DOUBLE TRIGGER
 *  - use synchronous trigger TRGA0N to start Group A scan; int priority 4
 *  - use synchronous trigger TRGA1N to start Group B scan; int priority 5
 *  - use synchronous trigger TRGA2N to start Group C scan; int priority 6
 *  - allow each channel to be scanned four times and averaged before continuing
 *  - do not clear registers after reading
 */
config.resolution = ADC_RESOLUTION_8_BIT;
config.trigger    = ADC_TRIG_SYNC_TRG0AN;
config.priority   = 4;
config.trigger_groupb = ADC_TRIG_SYNC_TRG1AN;
config.priority_groupb= 5;
config.trigger_groupc = ADC_TRIG_SYNC_TRG2AN;
config.priority_groupc= 6;
config.add_cnt    = ADC_ADD_OFF;
config.alignment  = ADC_ALIGN_RIGHT;
config.clearing   = ADC_CLEAR_AFTER_READ_OFF;
R_ADC_Open(0, ADC_MODE_SS_MULTI_CH_GROUPED_GROUPC, &config, MyCallback);

/* CONFIGURE SCAN */

/* Clear all members of the adc_ch_cfg_t structure */
memset(&ch_cfg, 0, sizeof(ch_cfg));
```

```
/* Have channel 1 and 2 as Group A, have 3 and 4 as Group B,
   have 5 and 6 as Group C
   Perform addition/average on all channels except 9
*/
ch_cfg.scan_mask = ADC_MASK_CH1 | ADC_MASK_CH2;
ch_cfg.scan_mask_groupb = ADC_MASK_CH3 | ADC_MASK_CH4;
ch_cfg.scan_mask_groupc = ADC_MASK_CH5 | ADC_MASK_CH6;
ch_cfg.priority_groupa = ADC_GRPA_PRIORITY_OFF;
ch_cfg.add_mask = 0;
ch_cfg.diag_method = ADC_DIAG_OFF;
ch_cfg.anex_enable = false;
ch_cfg.sample_hold_mask = 0;
R_ADC_Control(0, ADC_CMD_CONFIGURE_SCAN, &ch_cfg);

/* After open, wait 1 us or longer before A/D conversion starts */

/* ENABLE TRIGGERS */
R_ADC_Control(0, ADC_CMD_ENABLE_TRIG, NULL);

/* INTERRUPT OCCURS UPON SCAN COMPLETION */



/* The callback is called twice from interrupt level- once after each
 *  group scan completes. The order depends upon the trigger order.
 */
void MyCallback(void *p_args)
{
adc_cb_args_t  *args;
uint16_t        data1,data2,data3,data4,data5,data6;


    args = (adc_cb_args_t *)p_args;

    /* READ RESULTS */

    if (args->event == ADC_EVT_SCAN_COMPLETE)
    {
        /* From S12ADIO interrupt, Group A scan complete, read registers */
        R_ADC_Read(0, ADC_REG_CH1, &data1);
        R_ADC_Read(0, ADC_REG_CH2, &data2);
    }
    else if (args->event == ADC_EVT_SCAN_COMPLETE_GROUPB)
    {
        /* From GBADI interrupt, Group B scan complete, read registers */
        R_ADC_Read(0, ADC_REG_CH3, &data3);
        R_ADC_Read(0, ADC_REG_CH4, &data4);
    }
    else if (args->event == ADC_EVT_SCAN_COMPLETE_GROUPC)
    {
        /* From GCADI interrupt, Group C scan complete, read registers */
        R_ADC_Read(0, ADC_REG_CH5, &data5);
        R_ADC_Read(0, ADC_REG_CH6, &data6);
    }
    /* process data, or set flag for application level to do so */
}
```

Example 5: Multiple Channels with Interrupt Trigger and Comparator Checking (RX64M, RX71M)

```
adc_cfg_t      config;
adc_ch_cfg_t   ch_cfg;
adc_cmpwin_t   cmpwin;


/* INITIALIZE MTU HERE (USED FOR TRIGGER SOURCES) */

/* OPEN UNIT 0 */

/* Clear all members of the adc_cfg_t structure */
memset(&config, 0, sizeof(config));

config.resolution = ADC_RESOLUTION_12_BIT;
config.trigger = ADC_TRIG_SYNC_TRG0AN;
config.priority = 4;
config.add_cnt = ADC_ADD_OFF;
config.alignment = ADC_ALIGN_RIGHT;
config.clearing = ADC_CLEAR_AFTER_READ_OFF;
R_ADC_Open(0, ADC_MODE_SS_MULTI_CH, &config, MyCallback);


/* CONFIGURE SCAN OF CHANNELS 3-5 */

/* Clear all members of the adc_ch_cfg_t structure */
memset(&ch_cfg, 0, sizeof(ch_cfg));

ch_cfg.chan_mask = ADC_MASK_CH3 | ADC_MASK_CH4 | ADC_MASK_CH5;
ch_cfg.diag_method = ADC_DIAG_OFF;
ch_cfg.anex_enable = false;
ch_cfg.sample_hold_mask = 0;
R_ADC_Control(0, ADC_CMD_ENABLE_CHANS, &ch_cfg);


/* HAVE COMPARATOR CHECK ON CHANNELS 3-4 FOR DROPPING BELOW 1.65V */

/* Clear all members of the adc_cmpwin_t structure */
memset(&cmpwin, 0, sizeof(cmpwin));

cmpwin.compare_mask = ADC_MASK_CH3 | ADC_MASK_CH4;
cmpwin.inside_window_mask = 0;          // condition met when below level
cmpwin.level_lo = 0x7FF;                // 12-bit 3.3V=0xFFF, 1.65V=0x7FF
cmpwin.int_priority = 3;
R_ADC_Control(0, ADC_CMD_EN_COMPARATOR_LEVEL, &cmpwin);


/* ENABLE TRIGGERS */
R_ADC_Control(0, ADC_CMD_ENABLE_TRIG, NULL);


/* INTERRUPT OCCURS UPON SCAN COMPLETION */

  :
```

```
/* Callback called from interrupt level: */

void MyCallback(void *p_args)
{
adc_cb_args_t  *args;
uint16_t        data3,data4,data5;


    args = (adc_cb_args_t *)p_args;

    /* READ RESULTS */

    if (args->event == ADC_EVT_SCAN_COMPLETE)
    {
        R_ADC_Read(0, ADC_REG_CH3, &data3);
        R_ADC_Read(0, ADC_REG_CH4, &data4);
        R_ADC_Read(0, ADC_REG_CH5, &data5);
    }

    if (args->event == ADC_EVT_CONDITION_MET)
    {
        if (args->compare_flags & ADC_MASK_CH3)
        {
            // processing when channel 3 voltage is too low
        }
        else
        {
            // processing when channel 4 voltage is too low
        }
    }
}
```

Example 6: Multiple Channels with Interrupt Trigger and 2 Comparator Checking (RX65x)

```
adc_cfg_t      config;
adc_ch_cfg_t   ch_cfg;
adc_cmpwin_t   cmpwin;

/* Clear all members of each structure */
memset(&config, 0, sizeof(config));
memset(&ch_cfg, 0, sizeof(ch_cfg));
memset(&cmpwin, 0, sizeof(cmpwin));

/* INITIALIZE MTU HERE (USED FOR TRIGGER SOURCES) */

/* OPEN UNIT 0 */

config.resolution = ADC_RESOLUTION_12_BIT;
config.trigger = ADC_TRIG_SYNC_TRG0AN;
config.priority = 4;
config.add_cnt = ADC_ADD_OFF;
config.alignment = ADC_ALIGN_RIGHT;
config.clearing = ADC_CLEAR_AFTER_READ_OFF;
R_ADC_Open(0, ADC_MODE_SS_MULTI_CH, &config, MyCallback);
```

```
/* CONFIGURE SCAN OF CHANNELS 3-4 */

ch_cfg.chan_mask = ADC_MASK_CH3 | ADC_MASK_CH4 | ADC_MASK_CH5;
ch_cfg.diag_method = ADC_DIAG_OFF;
ch_cfg.anex_enable = false;
ch_cfg.sample_hold_mask = 0;
R_ADC_Control(0, ADC_CMD_CONFIGURE_SCAN, &ch_cfg);


/* HAVE COMPARATOR CHECK ON CHANNELS 3-4 FOR DROPPING BELOW 1.65V */

cmpwin.compare_mask = ADC_MASK_CH3 | ADC_MASK_CH4;
cmpwin.compare_maskb = ADC_COMP_WINB_CH5;
cmpwin.inside_window_mask = 0;            // Condition met when below level
cmpwin.inside_window_maskb = ADC_COMP_WINB_COND_BELOW;
cmpwin.level_lo = 0x7FF;                // 12-bit 3.3V=0xFFF, 1.65V=0x7FF
cmpwin.level_lob = 0x7FF;               // 12-bit 3.3V=0xFFF, 1.65V=0x7FF
cmpwin.int_priority = 3;
cmpwin.windowa_enable = true;
cmpwin.windowb_enable = true;
R_ADC_Control(0, ADC_CMD_EN_COMPARATOR_LEVEL, &cmpwin);

/* After open, wait 1 us or longer before A/D conversion starts */

/* ENABLE TRIGGERS */
R_ADC_Control(0, ADC_CMD_ENABLE_TRIG, NULL);

/* INTERRUPT OCCURS UPON SCAN COMPLETION */

  :

/* Callback called from interrupt level: */

void MyCallback(void *p_args)
{
adc_cb_args_t  *args;
uint16_t        data3,data4,data5;


    args = (adc_cb_args_t *)p_args;

    /* READ RESULTS */

    if (args->event == ADC_EVT_SCAN_COMPLETE)
    {
        R_ADC_Read(0, ADC_REG_CH3, &data3);
        R_ADC_Read(0, ADC_REG_CH4, &data4);
        R_ADC_Read(0, ADC_REG_CH5, &data5);
    }

    if (args->event == ADC_EVT_CONDITION_MET)
    {
        if (args->compare_flags & ADC_MASK_CH3)
        {
            // processing when channel 3 voltage is too low
        }
        else
        {
            // processing when channel 4 voltage is too low
        }
    }

    if (args->event == ADC_EVT_CONDITION_METB)
    {
        // processing when channel 5 voltage is too low
    }
}
```

Special Notes (RX Family Common):

When the A/D conversion start (ADST) bit is 1, settings such as mode must not be changed using this function. However, the conversion status or the comparison result can be obtained.
When switching channels used for A/D conversion or settings, call the R_ADC_Close() function once and then call the R_ADC_Open() function again to start.

When waiting completion of A/D conversion using the R_ADC_Control function, use the following commands.

A/D Conversion Channel Settings			Commands for the R_ADC_Control Function	
Mode	A/D conversion start trigger	Interrupt	Starts A/D conversion	Waits completion of A/D conversion
Single scan	Software trigger	–	ADC_CMD_SCAN_NOW	ADC_CMD_CHECK_SCAN_DONE
	Other than software trigger	Disabled	ADC_CMD_ENABLE_TRIG	ADC_CMD_CHECK_SCAN_DONE_GROUPA (1)
Continuous scan	Software trigger	Disabled	ADC_CMD_SCAN_NOW	ADC_CMD_CHECK_SCAN_DONE_GROUPA (1)
	Other than software trigger	Disabled	ADC_CMD_ENABLE_TRIG	ADC_CMD_CHECK_SCAN_DONE_GROUPA (1)
Group scan	Other than software trigger	Disabled	ADC_CMD_ENABLE_TRIG	ADC_CMD_CHECK_SCAN_DONE_GROUPA (1) ADC_CMD_CHECK_SCAN_DONE_GROUPB (2) ADC_CMD_CHECK_SCAN_DONE_GROUPC (3)

Note 1. ADC_CMD_CHECK_SCAN_DONE_GROUPA cannot be used with S12ADa. Please check the interrupt request flag directly for completion of A/D conversion.

Note 2. Use ADC_CMD_CHECK_SCAN_DONE_GROUPB when waiting completion of A/D conversion for Group B.

Note 3. ADC_CMD_CHECK_SCAN_DONE_GROUPC can be used only with S12ADFa.

When A/D conversion interrupts are enabled, the R_ADC_Control() function cannot be used to wait completion of A/D conversion except when using single scan mode with software trigger. In this case, use the callback function for the A/D conversion interrupt to wait completion of A/D conversion.

Special Notes (S12ADC, S12ADFa):

Channels and sensors can be combined in the same unit.

ELC is only for S12ADI, not S12GBADI or S12CMPI. (S12ADC)

ELC is only for S12ADI, not GBADI, GCADI, S12CMPAI or S12CMPBI. (S12ADFa)

The application should wait 30 µs after configuring the scan before enabling the trigger for Temperature Sensor for best results.

If Group A Priority is selected such that Group B operates in continuous scan mode, it is recommended not to use the S12GBADI interrupt (S12ADC) and GBADI interrupt (S12ADFa) since the interrupt handling will be processed so often. That causes the majority of the processing time to be spent at the interrupt level.

Enabling the comparator should be done prior to enabling the triggers.

Some features may not be used with others. The following table illustrates this.

	Dbl Trig	Group Scan	Self-Diag	Add/ Avg	ANEX	Sample &Hold	Priority GroupA	Sensors	Comparator	DDA
Double Trigger			X			*B		X	X	
Group Scan					X	*S				
Self-Diagnosis	X			X	X				X	X
Add/Avg			X							
ANEX		X	X					X		X
Sample &Hold	*B	*S					*A			
Priority GroupA						*A				
Sensors	X				X					X
Comparator	X		X							
Disconnect Detection Assist			X		X			X		

X - Combination may not be used. For example, ANEX may not be used with group scan modes, Self-Diagnosis, sensors or Disconnect Detection Assist.
*A - Sample and Hold channels must be in Group A.
*B - Sample and Hold channels must be in Group B or Group C.
*S - Sample and Hold channels cannot be split across groups.

Special Notes (S12ADE):

This function does not support following features.

- Compare function window B
- Compare function window A/B composite condition setting

Special Notes (S12ADC/S12ADE/S12ADFa):

When using the comparison, configure the comparison after the channel configuration.

Special Notes (S12ADa):

Only AN008 to AN020 can be used for setting the number of sampling states. AN000 to AN007 are fixed to 20 states regardless of the setting.

Special Notes (S12ADb except RX210):

For temperature sensor output and internal reference voltage, the number of sampling states must be set to 5 µs or greater.

3.4 R_ADC_Read()

This function reads conversion results from a single channel, sensor, double trigger, or self-diagnosis register.

Format

```
adc_err_t  R_ADC_Read(uint8_t                unit,
                      adc_reg_t const        reg_id,
                      uint16_t * const       p_data);
```

Parameters

unit

0 or 1. For MCUs with only one unit, 0 should be passed (only the RX64M/RX71M/RX65x have 2 units).

reg_id

Id for the register to read. See 2.11.50 "Channel Definitions in the R_ADC_Read Function (S12ADb except RX210)" to 2.11.53 "Channel Definitions in the R_ADC_Read Function (S12ADC, S12ADFa)" for details on register ID.

p_data

Pointer to variable to load value into.

Return Values

ADC_SUCCESS:	*Success*
ADC_ERR_INVALID_ARG:	*unit or reg_id contains an invalid value.*
ADC_ERR_MISSING _PTR:	*p_data is FIT_NO_PTR/NULL*

Properties

Prototyped in file "r_s12ad_rx_if.h"

Description

Reads conversion results from a single channel, sensor, double trigger, or self-diagnosis register.

Reentrant

Yes.

Example

```
uint16_t data;
     :
    /* Read channel 0 on unit 0 */
    R_ADC_Read(0, ADC_CH0_REG, &data);    // conversion value placed in "data"
```

Special Notes (S12ADb except RX210):

For temperature sensor output and internal reference voltage, discard the first A/D conversion result after the open, and use the second and the subsequent A/D conversion results.

3.5 R_ADC_ReadAll()

This function reads conversion results from all potential sources, enabled or not.

Format

adc_err_t R_ADC_ReadAll(adc_data_t * const p_data);

Parameters

p_data

Pointer to structure to load register values into. See 2.11.54 "A/D Conversion Result Storage Structure in the R_ADC_ReadAll Function (S12ADb except RX210)" to 2.11.57 "A/D Conversion Result Storage Structure in the R_ADC_ReadAll Function (S12ADC, S12ADFa)" for details on structures.

Return Values

ADC_SUCCESS:	*Success*
ADC_ERR_MISSING_PTR:	*p_data is FIT_NO_PTR/NULL*

Properties

Prototyped in file "r_s12ad_rx_if.h"

Description

Reads conversion results from all potential sources, enabled or not.

Reentrant

Yes.

Example

```
adc_data_t data;
     :
    /* Read all channel registers available on hardware */
    R_ADC_ReadAll(&data);     // "data" loaded with all conversion reg values
```

Special Notes:

None.

3.6 R_ADC_Close()

This function ends any scan in progress, disables interrupts, and removes power to the A/D peripheral.

Format

adc_err_t R_ADC_Close(uint8_t unit);

Parameters

unit

0 or 1. For MCUs with only one unit, 0 should be passed (only the RX64M/RX71M/RX65x have 2 units).

Return Values

ADC_ERR_INVALID_ARG:	*Unit not 0 or 1*

Properties

Prototyped in file “r_s12ad_rx_if.h”

Description

Ends any scan in progress, disables interrupts, and removes power to the A/D peripheral. When changing scan configurations, call the R_ADC_Open() function again after this function is called.

Reentrant

This may only be called once per unit after the R_ADC_Open() function is performed.

Example

```
    :
  err = R_ADC_Open(1, ADC_MODE_SS_MULTI_CH_GROUPED, &config, MyCallback);
    :
  R_ADC_Close(1);
```

Special Notes:

This function will abort any scan that may be in progress.

3.7 R_ADC_GetVersion()

This function returns the driver version number at runtime.

Format

uint32_t R_ADC_GetVersion(void)

Parameters

None

Return Values

Version number.

Properties

Prototyped in file “r_s12ad_rx_if.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.

Reentrant

Yes

Example

```
uint32_t   version;
  :
version = R_ADC_GetVersion();
```

Special Notes:

This function is in-lined using the “#pragma inline” directive

4. Pin Setting

To use the ADC FIT module, assign input/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 after calling the R_ADC_Open function.

When performing the Pin Setting in the e^2 studio, the Pin Setting feature of the FIT configurator or the Smart Configurator can be used. When using the Pin Setting feature, a source file is generated according to the option selected in the Pin Setting window in the FIT configurator or the Smart Configurator. Pins are configured by calling the function defined in the source file. Refer to Table 4.1 for details.

Table 4.1 Function Output by the FIT Configurator

MCU Used	Option Selected	Function to be Output	Remarks
RX64M, RX71M, RX65N	Unit 0	R_ADC_PinSet_S12AD0()	
	Unit 1	R_ADC_PinSet_S12AD1()	
RX110, RX111, RX113, RX130, RX210, RX230, RX231, RX63x	Unit 0	R_ADC_PinSet_S12AD0()	

5. Demo Projects

Demo projects are complete stand-alone programs. They include function main() that utilizes the module and its dependent modules (e.g. r_bsp). The standard naming convention for the demo project is <module>_demo_<board> where <module> is the peripheral acronym (e.g. s12ad, cmt, sci) and the <board> is the standard RSK (e.g. rskrx113). For example, s12ad FIT module demo project for RSKRX113 will be named as s12ad_demo_rskrx113. Similarly the exported .zip file will be <module>_demo_<board>.zip. For the same example, the zipped export/import file will be named as s12ad_demo_rskrx113.zip.

5.1 s12ad_int_demo_rskrx113

This demo uses periodic interrupts from MTU0 to trigger the ADC module to scan the potentiometer on the board. Each time a scan completes, the program reads the converted value at interrupt level in a callback function and places it into a global variable called "data". This variable should be added to the Expressions window and made into a Real-time Watch (double-click to make real-time). As the program runs, change the potentiometer position and observe the corresponding changes in the variable.

5.2 s12ad_poll_demo_rskrx113

This demo scans the potentiometer on the board via a software trigger in an endless loop. Each time a scan completes, the program reads the converted value at the application level and places it into a global variable called "data". This variable should be added to the Expressions window and made into a Real-time Watch (double-click to make real-time). As the program runs, change the potentiometer position and observe the corresponding changes in the variable.

5.3 s12ad_poll_demo_rskrx130

This demo scans the potentiometer on the board via a software trigger in an endless loop. Each time a scan completes, the program reads the converted value at the application level and places it into a global variable called "data". This variable should be added to the Expressions window and made into a Real-time Watch (double-click to make real-time). As the program runs, change the potentiometer position and observe the corresponding changes in the variable.

5.4 s12ad_demo_rskrx64m

This is a simple demo of the RX64M A/D Converter (S12AD) for the RSKRX64M starter kit (FIT module "r_s12ad_rx"). The demo uses the Multi-Function Timer Pulse Unit (MTU3a) to periodically trigger the ADC module to perform conversion on channel 0 which is connected to the on-board potentiometer. Each time a scan completes, the program reads the converted value at interrupt level in a callback function and places it into a global variable called "g_data". This variable should be added to the Expressions window and made into a Real-time Watch (double-click to make real-time). As the program runs, change the potentiometer position and observe the corresponding changes in the variable.

5.5 s12ad_demo_rskrx71m

This is a demo of the RX71M A/D Converter (S12AD) for the RSKRX71M starter kit (FIT module "r_s12ad_rx"). The demo uses the Multi-Function Timer Pulse Unit 3 (MTU3a) to periodically trigger the ADC module to perform conversion on channel 0 which is connected to the on-board potentiometer. Each time a scan completes, the program reads the converted value at interrupt level in a callback function and places it into a global variable called "g_data". This variable should be added to the Expressions window and made into a Real-time Watch (double-click to make real-time). As the program runs, change the potentiometer position and observe the corresponding changes in the variable.

5.6 s12ad_demo_rskrx231

This is a demo of the RX231 A/D Converter (S12ADE) for the RSKRX231 starter kit (FIT module "r_s12ad_rx"). The demo uses the Multi-Function Timer Pulse Unit 2 (MTU2a) to periodically trigger the ADC module to perform a conversion on channel 0, which is connected to the on-board potentiometer. Each time a scan completes, the program reads the converted value at interrupt level in a callback function and places it into a global variable called "g_data". This variable should be added to the Expressions window and made into a Real-time Watch. To do that, add it to the Expressions window then right-click it. From the drop-down menu click on "Real-time Refresh". Right click again and select "Real-time Refresh Interval" and set the refresh value to 200 ms. As the program runs, change the potentiometer position and observe the corresponding changes in the variable.

5.7 Adding a Demo to a Workspace

Demo projects are found in the FITDemos subdirectory of the e^2 studio installation directory. 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 demo subdirectory, select the desired demo zip file, then click “Finish”.

5.8 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 the required application note and select “Sample Code (download)” from the context menu in the *Smart Brower >> Application Notes* tab.

6. Appendices

6.1 Operation Confirmation Environment

This section describes operation confirmation environment for the ADC FIT module.

Table 6.1 Operation Confirmation Environment (Rev. 2.30)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 5.4.0 (WS Patch)
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.2.30
Board used	Renesas Starter Kit+ for RX 65N-2MB (product No.: RTK50565N2SxxxxxBE) Renesas Starter Kit for RX130-512KB (product No.: RTK5051308SxxxxxBE)

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:

- When using CS+:
  Application note "Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)"
- When 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. For this, 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_s12ad_rx module.

A: The FIT module you added may not support the target device chosen in the user project. Check if the FIT module supports the target device for the project used.

(3) Q: The voltage input to the analog input pin and the A/D conversion result do not match.

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.

Related Technical Updates

This module reflects the content of the following technical updates.

- TN-RX*-A124A/E
- TN-RX*-A117A/E

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/inquiry

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

Revision Record

Rev.	Date	Description Page	Summary
1.00	Nov.15.2013	—	First edition issued
1.20	Apr.21.2014	1,3	Added mention of support for RX110/63x.
		11,12	Added interface for RX210 Sample&Hold, Self-Diagnosis, and Disconnect Detection Assist (DDA)
1.30	Jun.05.2014	—	Fixed bug in code that eliminated channels 8-15.
1.40	Nov.07.2014	—	Added RX113 support.
2.00	Mar.30.2015	—	Added RX64M/RX71M support. Modified interface to include a unit number.
2.10	Jun.15.2015	—	Added RX231 support. Added an RX231 demo.
2.11	Mar.01.2016	—	Added RX130 and RX230 support.
2.20	Dec.01.2016	—	Added RX65N support.
		5	2.9 Code Size: - Changed code sizes for RX111. - Added code sizes for RX65N.
		53 to 64	3.2 R_ADC_Open(), 3.3 R_ADC_Control(): Added the following code in each Example section. - Code to clear all fields of each structure. - Comment regarding a wait time before A/D conversion starts after open.
		55	3.2 R_ADC_Open(): Added the Special Notes (RX 63x) and Special Notes (RX110/RX111/RX113/RX210/RX130/RX230/RX231/RX65x).
		56	3.3 R_ADC_Control(): Added the sentence to clear all members of parameters in the Description.
		65	3.3 R_ADC_Control(): Added and modified the following items: - Special Notes (RX Family Common): Added four special notes. - Special Notes (RX64M/RX71M/RX65x): Added a special note regarding operation under Group A Priority Control and modified the table. - Special Notes (RX63x) and Special Notes (RX110/RX111/RX113): Added.
		67	3.4 R_ADC_Read(): Added Special Notes (RX110/RX111/RX113).
		71	4. Pin Setting: Added.
		72	5.3 s12ad_poll_demo_rskrx130: Added.
		Program	Fixed typo on comment lines.
			Revised the initialization in the R_ADC_Open function.

		Description	
Rev.	Date	Page	Summary
2.20	Dec.01.2016	Program	Fixed the following issue: Target Device: RX64M/RX71M/RX230/RX231 Description: There is an error in checking the range of the arguments. Thus, when the trigger source de-selection state is set as the trigger for group B, the R_ADC_Open function returns an error. Condition: The following combination of arguments for the R_ADC_Open function is set. Second parameter (mode) ADC_MODE_SS_MULTI_CH_GROUPED or ADC_MODE_SS_MULTI_CH_GROUPED_DBLTRIG_A Third parameter (p_cfg->trigger_groupb) ADC_TRIG_NONE_GROUPB. Measure: Modified the code for checking the arguments of the adc_check_open_cfg function. Use Rev. 2.20 or later version of the ADC FIT module.
			Fixed the following issue: Target Device: RX230/RX231 Description: The compare window A operation enable bit is not set to be enabled. Thus comparison for levels and windows does not work. Condition: Comparison does not work under any condition. Measure: Modified the code to enable the CMPAE bit using the adc_control function when the compare function is selected. Use Rev. 2.20 or later version of the ADC FIT module.
			Fixed the following issue: Target Device: RX64M/RX71M/RX230/RX231 Description: After Disconnection Detection Assist (DDA) is set, the register is not reset. Thus the Disconnection Detection Assist (DDA) setting remains and this causes a combination error when setting self-diagnosis. Then the R_ADC_Control function returns an error. Condition: After Disconnection Detection Assist (DDA) is set, the FIT module is closed and re-opened, and then self-diagnosis is set. Measure: Added processing to reset all S12AD related registers in the adc_open function and deleted the check during Disconnection Detection Assist (DDA) operation from the check with self-diagnosis set in the adc_check_scan_config function. Use Rev. 2.20 or later version of the ADC FIT module.

Rev.	Date	Description Page	 Summary
2.20	Dec.01.2016	Program	Fixed the following issue: Target Device: RX230/RX231 Description: The numbers of arguments (enum value) for an index of the register table do not match and the indexed value becomes out of range. Then the R_ADC_Read function cannot obtain the result of self-diagnosis. Condition: Occurs under any conditions. Measure: Deleted unnecessary definitions from the enum (abc_reg_t) for an index of the register table. Use Rev. 2.20 or later version of the ADC FIT module.
			Fixed the following issue: Target Device: RX210 Description: A parameter needed for compiling was deleted in rev. 2.10, thus a build error occurs when compiling with RX210. Condition: A project with Rev.2.10 or Rev.2.11 of the ADC FIT module is built. Measure: Added ADC_CFG_PGA_GAIN to r_s12ad_rx_config.h. Use Rev. 2.20 or later version of the ADC FIT module.
			Deleted unnecessary definitions.
			Deleted unnecessary members.
			Modified the following procedures according to the User's Manual: Hardware: - Procedure for when A/D conversion stops - Procedure for when entering low power consumption modes - Procedure to rewrite the ADHSC bit
			Fixed the following issue: Target Device: RX64M/RX71M/RX230/RX231 Description: Since the operator is incorrect in processing to avoid the upper limit voltage becoming less than the lower limit voltage, the upper and lower limit voltages cannot be set to the same value in the comparison setting. Condition: The comparison (window comparison) is used. Measure: Use Rev. 2.20 or later version of the ADC FIT module.
			Modified the code to set the delay time properly when converting the temperature sensor in RX64M and RX71M.
			Modified processing for checking an invalid channel when using the extended analog input in RX64M and RX71M.

		Description	
Rev.	Date	Page	Summary
2.20	Dec.01.2016	Program	Unify the name of definitions that have same meanings but have different names among MCU Groups. RX63x ADC_TRIG_ASYNC_ADTRG0 → ADC_TRIG_ASYNC_ADTRG ADC_TRIG_SYNC_TRG0AN_0 → ADC_TRIG_SYNC_TRG0AN ADC_TRIG_SYNC_TRG0BN_0 → ADC_TRIG_SYNC_TRG0BN ADC_TRIG_SYNC_TRGAN_0 → ADC_TRIG_SYNC_TRGAN ADC_TRIG_SYNC_TRGAN_1 → ADC_TRIG_SYNC_TPUTRGAN ADC_TRIG_SYNC_TRG0EN_0 → ADC_TRIG_SYNC_TRG0EN ADC_TRIG_SYNC_TRG0FN_0 → ADC_TRIG_SYNC_TRG0FN ADC_TRIG_SYNC_TRG4ABN_0 → ADC_TRIG_SYNC_TRG4AN_OR_TRG4BN ADC_TRIG_SYNC_TRG4ABN_1 → ADC_TRIG_SYNC_TPUTRG0AN ADC_TRIG_SYNC_TMRTRG0AN_0 → ADC_TRIG_SYNC_TMRTRG0AN ADC_TRIG_SYNC_TMRTRG0AN_1 → ADC_TRIG_SYNC_TMRTRG2AN RX110 ADC_CONVERT_SPEED_HI → ADC_CONVERT_SPEED_HIGH ADC_TRIG_NONE_GROUPB → Deleted ADC_TRIG_ASYNC_ADTRG0 → ADC_TRIG_ASYNC_ADTRG RX111 ADC_CONVERT_SPEED_HI → ADC_CONVERT_SPEED_HIGH ADC_TRIG_NONE_GROUPB → Deleted ADC_TRIG_ASYNC_ADTRG0 → ADC_TRIG_ASYNC_ADTRG ADC_TRIG_SYNC_TRGAN → ADC_TRIG_SYNC_TRGAN_OR_UDF4N ADC_TRIG_SYNC_TRG4ABN → ADC_TRIG_SYNC_TRG4AN_AND_TRG4BN RX113 ADC_CONVERT_SPEED_HI → ADC_CONVERT_SPEED_HIGH ADC_TRIG_NONE_GROUPB → Deleted ADC_TRIG_ASYNC_ADTRG0 → ADC_TRIG_ASYNC_ADTRG ADC_TRIG_SYNC_TRGAN → ADC_TRIG_SYNC_TRGAN_OR_UDF4N ADC_TRIG_SYNC_TRG4ABN → ADC_TRIG_SYNC_TRG4AN_AND_TRG4BN RX210 ADC_TRIG_NONE_GROUPB → Deleted ADC_TRIG_ASYNC_ADTRG0 → ADC_TRIG_ASYNC_ADTRG ADC_TRIG_SYNC_TRGAN → ADC_TRIG_SYNC_TRGAN_OR_UDF4N ADC_TRIG_SYNC_TRG4ABN → ADC_TRIG_SYNC_TRG4AN_AND_TRG4BN ADC_TRIG_PLACEHOLDER → ADC_TRIG_SYNC_TEMPS ADC_TRIG_SYNC_TRGAN1 → ADC_TRIG_SYNC_TPUTRGAN ADC_TRIG_SYNC_TRG4ABN1 → ADC_TRIG_SYNC_TPUTRG0AN

		Description	
Rev.	Date	Page	Summary
2.20	Dec.01.2016	Program	RX64M ADC_CMD_CONFIGURE_SCAN → ADC_CMD_ENABLE_CHANS ADC_TRIG_NONE_GROUPB → ADC_TRIG_NONE ADC_TRIG_ASYNC_ADTRG0 → ADC_TRIG_ASYNC_ADTRG ADC_TRIG_SYNC_TRGA0N → ADC_TRIG_SYNC_TRG0AN ADC_TRIG_SYNC_TRGA1N → ADC_TRIG_SYNC_TRG1AN ADC_TRIG_SYNC_TRGA2N → ADC_TRIG_SYNC_TRG2AN ADC_TRIG_SYNC_TRGA3N → ADC_TRIG_SYNC_TRG3AN ADC_TRIG_SYNC_TRGA4N → ADC_TRIG_SYNC_TRG4AN_OR_UDF4N ADC_TRIG_SYNC_TRGA6N → ADC_TRIG_SYNC_TRG6AN ADC_TRIG_SYNC_TRGA7N → ADC_TRIG_SYNC_TRG7AN_OR_UDF7N ADC_TRIG_SYNC_TRG0N → ADC_TRIG_SYNC_TRG0EN ADC_TRIG_SYNC_TRG4ABN → ADC_TRIG_SYNC_TRG4AN_AND_TRG4BN ADC_TRIG_SYNC_TRG7ABN → ADC_TRIG_SYNC_TRG7AN_AND_TRG7BN ADC_TRIG_SYNC_GTADTRA0N → ADC_TRIG_SYNC_GTADTR0AN ADC_TRIG_SYNC_GTADTRB0N → ADC_TRIG_SYNC_GTADTR0BN ADC_TRIG_SYNC_GTADTRA1N → ADC_TRIG_SYNC_GTADTR1AN ADC_TRIG_SYNC_GTADTRB1N → ADC_TRIG_SYNC_GTADTR1BN ADC_TRIG_SYNC_GTADTRA2N → ADC_TRIG_SYNC_GTADTR2AN ADC_TRIG_SYNC_GTADTRB2N → ADC_TRIG_SYNC_GTADTR2BN ADC_TRIG_SYNC_GTADTRA3N → ADC_TRIG_SYNC_GTADTR3AN ADC_TRIG_SYNC_GTADTRB3N → ADC_TRIG_SYNC_GTADTR3BN ADC_TRIG_SYNC_GTADTRA0N_OR_GTADTRB0N → ADC_TRIG_SYNC_GTADTR0AN_OR_GTADTR0BN ADC_TRIG_SYNC_GTADTRA1N_OR_GTADTRB1N → ADC_TRIG_SYNC_GTADTR1AN_OR_GTADTR1BN ADC_TRIG_SYNC_GTADTRA2N_OR_GTADTRB2N → ADC_TRIG_SYNC_GTADTR2AN_OR_GTADTR2BN ADC_TRIG_SYNC_GTADTRA3N_OR_GTADTRB3N → ADC_TRIG_SYNC_GTADTR3AN_OR_GTADTR3BN ADC_TRIG_SYNC_TMTRG0AN_0 → ADC_TRIG_SYNC_TMRTRG0AN ADC_TRIG_SYNC_TMTRG0AN_1 → ADC_TRIG_SYNC_TMRTRG2AN ADC_TRIG_SYNC_TPTRGAN → ADC_TRIG_SYNC_TPUTRGAN ADC_TRIG_SYNC_TPTRG0AN → ADC_TRIG_SYNC_TPUTRG0AN ADC_TRIG_SYNC_ELCTRG → ADC_TRIG_SYNC_ELC

Rev.	Date	Description	
		Page	Summary
2.20	Dec.01.2016	Program	RX71M ADC_CMD_CONFIGURE_SCAN → ADC_CMD_ENABLE_CHANS ADC_TRIG_NONE_GROUPB → ADC_TRIG_NONE ADC_TRIG_ASYNC_ADTRG0 → ADC_TRIG_ASYNC_ADTRG ADC_TRIG_SYNC_TRGA0N → ADC_TRIG_SYNC_TRG0AN ADC_TRIG_SYNC_TRGA1N → ADC_TRIG_SYNC_TRG1AN ADC_TRIG_SYNC_TRGA2N → ADC_TRIG_SYNC_TRG2AN ADC_TRIG_SYNC_TRGA3N → ADC_TRIG_SYNC_TRG3AN ADC_TRIG_SYNC_TRGA4N → ADC_TRIG_SYNC_TRG4AN_OR_UDF4N ADC_TRIG_SYNC_TRGA6N → ADC_TRIG_SYNC_TRG6AN ADC_TRIG_SYNC_TRGA7N → ADC_TRIG_SYNC_TRG7AN_OR_UDF7N ADC_TRIG_SYNC_TRG0N → ADC_TRIG_SYNC_TRG0EN ADC_TRIG_SYNC_TRG4ABN → ADC_TRIG_SYNC_TRG4AN_AND_TRG4BN ADC_TRIG_SYNC_TRG7ABN → ADC_TRIG_SYNC_TRG7AN_AND_TRG7BN ADC_TRIG_SYNC_GTADTRA0N → ADC_TRIG_SYNC_GTADTR0AN ADC_TRIG_SYNC_GTADTRB0N → ADC_TRIG_SYNC_GTADTR0BN ADC_TRIG_SYNC_GTADTRA1N → ADC_TRIG_SYNC_GTADTR1AN ADC_TRIG_SYNC_GTADTRB1N → ADC_TRIG_SYNC_GTADTR1BN ADC_TRIG_SYNC_GTADTRA2N → ADC_TRIG_SYNC_GTADTR2AN ADC_TRIG_SYNC_GTADTRB2N → ADC_TRIG_SYNC_GTADTR2BN ADC_TRIG_SYNC_GTADTRA3N → ADC_TRIG_SYNC_GTADTR3AN ADC_TRIG_SYNC_GTADTRB3N → ADC_TRIG_SYNC_GTADTR3BN ADC_TRIG_SYNC_GTADTRA0N_OR_GTADTRB0N → ADC_TRIG_SYNC_GTADTR0AN_OR_GTADTR0BN ADC_TRIG_SYNC_GTADTRA1N_OR_GTADTRB1N → ADC_TRIG_SYNC_GTADTR1AN_OR_GTADTR1BN ADC_TRIG_SYNC_GTADTRA2N_OR_GTADTRB2N → ADC_TRIG_SYNC_GTADTR2AN_OR_GTADTR2BN ADC_TRIG_SYNC_GTADTRA3N_OR_GTADTRB3N → ADC_TRIG_SYNC_GTADTR3AN_OR_GTADTR3BN ADC_TRIG_SYNC_TMTRG0AN_0 → ADC_TRIG_SYNC_TMRTRG0AN ADC_TRIG_SYNC_TMTRG0AN_1 → ADC_TRIG_SYNC_TMRTRG2AN ADC_TRIG_SYNC_TPTRGAN → ADC_TRIG_SYNC_TPUTRGAN ADC_TRIG_SYNC_TPTRG0AN → ADC_TRIG_SYNC_TPUTRG0AN ADC_TRIG_SYNC_ELCTRG → ADC_TRIG_SYNC_ELC

		Description	
Rev.	Date	Page	Summary
2.20	Dec.01.2016	Program	RX130 ADC_TRIG_NONE_GROUPB → ADC_TRIG_NONE ADC_TRIG_ASYNC_ADTRG0 → ADC_TRIG_ASYNC_ADTRG ADC_TRIG_SYNC_TRGAN → ADC_TRIG_SYNC_TRGAN_OR_UDF4N ADC_TRIG_SYNC_TRG4ABN → ADC_TRIG_SYNC_TRG4AN_AND_TRG4BN ADC_TRIG_SYNC_ELCTRG0 → ADC_TRIG_SYNC_ELC RX230 ADC_TRIG_NONE_GROUPB → ADC_TRIG_NONE ADC_TRIG_ASYNC_ADTRG0 → ADC_TRIG_ASYNC_ADTRG ADC_TRIG_SYNC_TRGAN → ADC_TRIG_SYNC_TRGAN_OR_UDF4N ADC_TRIG_SYNC_TRG4ABN → ADC_TRIG_SYNC_TRG4AN_AND_TRG4BN ADC_TRIG_SYNC_ELCTRG0N_OR_ELCTRG1N → ADC_TRIG_SYNC_ELC ADC_TRIG_SYNC_TRGAN1 → ADC_TRIG_SYNC_TPUTRGAN ADC_TRIG_SYNC_TRG4ABN1 → ADC_TRIG_SYNC_TPUTRG0AN RX231 ADC_TRIG_NONE_GROUPB → ADC_TRIG_NONE ADC_TRIG_ASYNC_ADTRG0 → ADC_TRIG_ASYNC_ADTRG ADC_TRIG_SYNC_TRGAN → ADC_TRIG_SYNC_TRGAN_OR_UDF4N ADC_TRIG_SYNC_TRG4ABN → ADC_TRIG_SYNC_TRG4AN_AND_TRG4BN ADC_TRIG_SYNC_ELCTRG0N_OR_ELCTRG1N → ADC_TRIG_SYNC_ELC ADC_TRIG_SYNC_TRGAN1 → ADC_TRIG_SYNC_TPUTRGAN ADC_TRIG_SYNC_TRG4ABN1 → ADC_TRIG_SYNC_TPUTRG0AN
			Unify the member names in the adc_ch_cfg_t structure that are different among MCU Groups. RX64M/RX71M scan_mask → chan_mask scan_mask_groupb → chan_mask_groupb Deleted processing for checking the range of enum value to simplify the processing. * See the warning on compiling to check the enum range.

Rev.	Date	Description Page	 Summary
2.20	Dec.01.2016	Program	Fixed the following issue: Target Device: RX210 Description: In the processing for checking arguments, ADC_TRIG_SYNC_TEMPS is checked with “trigger” instead of "trigger_groupb". Then the R_ADC_Open function returns an error even if the ADC_TRIG_SYNC_TEMPS setting is valid. Condition: ADC_TRIG_SYNC_TEMPS is set as the trigger of A/D conversion. Measure: Deleted the code for checking ADC_TRIG_SYNC_TEMPS in the adc_open function. * “trigger_groupb” is ignored in modes other than group scan mode. In group scan mode, if ADC_TRIG_SYNC_TEMPS is set to trigger_groupb, an error is returned. Thus the checking process for ADC_TRIG_SYNC_TEMPS is unnecessary. Use Rev. 2.20 or later version of the ADC FIT module.
			Added the temperature sensor (temp) and internal reference voltage (volt) to the adc_data_t structure in the RX63x, RX110, RX111, RX113, and RX210 Groups to unify the behavior of the R_ADC_ReadAll function over all MCU groups.
			Fixed the following issue: Target Device: RX64M/RX71M Description: In the processing for checking arguments, ADC_TRIG_NONE is checked with "trigger". Then the R_ADC_Open function returns an error even if the ADC_TRIG_NONE setting is valid. Condition: ADC_TRIG_NONE is set as the trigger of A/D conversion. Measure: Deleted the code for checking ADC_TRIG_NONE in the adc_open function since ADC_TRIG_NONE can be set to the TRSA register as well as the TRSB register. Use Rev. 2.20 or later version of the ADC FIT module.
			Modified the code to reset the ADGSPCR register when setting a mode other than group scan mode in the RX130, RX230, RX231, RX64M, and RX71M.
			Changed the structure for arguments of comparison in RX130/RX230/RX231 to similar to the structure in RX65N. The adc_cmplvl_t structure has been discarded, accordingly. Please use the adc_cmpwin_t structure when using the level comparison.

		Description	
Rev.	Date	Page	Summary
2.20	Dec.01.2016	Program	Fixed the following issue: Target Device: RX130/RX230/RX231 Description: No processing is provided to set the compare window operation enable bit to "disabled". Thus once the compare function is enabled, only the way to disable it is reopening. However, please note that reopening does not work for RX230 and RX231. Condition: Always occurs when the compare function is used. Measure: Added "windowa_enable" to the structure for arguments of the compare function. Now the compare window operation enable bit can be set to "enabled" or "disabled" according to true/false setting of "windowa_enable", i.e. same processing as RX65N. Use Rev. 2.20 or later version of the ADC FIT module.
			Fixed the following issue: Target Device: RX64M/RX71M Description: No processing is provided to set the WCMPE bit to “0” (level comparison). Thus once window comparison is enabled, the comparison cannot be set to level comparison. Condition: The comparison is reset to level comparison after setting to window comparison. Measure: Modified the code to properly set the WCMPE bit according to the selection of window or level comparison. Use Rev. 2.20 or later version of the ADC FIT module.
			Modified the code to use the interface provided in the BSP (R_BSP_InterruptControl function) for specifying the interrupt enable bit and interrupt priority level when the compare interrupt is used in RX64M and RX71M.
			Fixed the following issue: Target Device: RX64M/RX71M Description: No processing is provided to set the compare interrupt enable bit to “disabled”. Thus once the comparison is enabled, the compare interrupt cannot be disabled. Condition: The interrupt priority level is set to "1" or greater while the comparison is enabled. Measure: Modified the code to disable the compare interrupt when executing the adc_close function and to disable group interrupts if no FIT module uses group interrupts. Use Rev. 2.20 or later version of the ADC FIT module.

Rev.	Date	Description: Page	Description: Summary
2.20	Dec.01.2016	Program	Fixed the following issue: Target Device: RX130/RX230/RX231/RX64M/RX71M Description: An unspecified callback function (NULL) is executed and improper interrupt occurs. Condition: After the R_ADC_Open function is executed with interrupts disabled, the interrupt priority level of the compare interrupt is set to “1” or greater. Measure: Modified the code to check the callback function before executing it. If the callback function is NULL, the interrupt handler is exited without performing any processing. Use Rev. 2.20 or later version of the ADC FIT module.
			Deleted unnecessary processing to reset the register when enabling an output of the temperature sensor in RX210 since the register is already reset to "0".
			Replaced the RX113 provided wait function (adc_delay) with the BSP provided wait function (R_BSP_SoftwareDelay). * The RX113 provided wait function (adc_delay) has been deleted.
			Fixed the following issue: Target Device: RX210 Description: An unnecessary error determination is performed. Because of this, when specifying a setting with the channel-dedicated sample-and-hold function, the R_ADC_Control function returns an error. Condition: In group scan mode, A/D conversion channels for group A and group B are set with the channel-dedicated sample-and-hold function. Measure: Deleted an unnecessary error determination as no limitation regarding it is described in the User’s Manual: Hardware. Use Rev. 2.20 or later version of the ADC FIT module.
			Fixed the following issue: Target Device: RX210 Description: Since an error determination processing is not provided, if self-diagnosis is enabled in a mode where self-diagnosis does not work, the R_ADC_Control function cannot return an error. Condition: Self-diagnosis is enabled when double trigger mode is selected in single scan mode or group scan mode. Measure: Added the error determination processing for when self-diagnosis is enabled. Use Rev. 2.20 or later version of the ADC FIT module.

		Description	
Rev.	Date	Page	Summary
2.20	Dec.01.2016	Program	Fixed the following issue: Target Device: RX130/RX230/RX231 Description: An unnecessary error determination is performed. Because of this, when setting the disconnection detection assist function after self-diagnosis is enabled, the R_ADC_Control function returns an error. Condition: Discharge or precharge is selected for the disconnection detection assist function after self-diagnosis is enabled. Measure: Deleted unnecessary determination processing described in the Description above. Use Rev. 2.20 or later version of the ADC FIT module.
			Fixed the following issue: Target Device: RX63x Description: The definition to determine a valid channel is incorrect and channel 20 cannot be selected. Condition: A chip with 177, 176, 145, or 144 pins is selected. Measure: Modified the definition to determine a valid channel. Use Rev. 2.20 or later version of the ADC FIT module.
			Fixed the following issue: Target Device: RX631 Description: There is no definition to determine a valid channel and this causes a compiling error. Condition: A chip with 64 pins or 48 pins is selected. Measure: Added the definition to determine a valid channel. Use Rev. 2.20 or later version of the ADC FIT module.
			Fixed the following issue: Target Device: RX64M/RX71M/RX65x Description: When obtaining the compare match result, the compare channel is cleared. Then, the subsequent compare match is not performed. Condition: When any of the unit 1 channel from channel 16 to channel 20 is specified as the compare channel, the condition is met and the compare match interrupt occurs, or the R_ADC_Control function is executed by setting ADC_CMD_CHECK_CONDITION_MET. Measure: Modified the register that was initialized when obtaining the compare match result. Use Rev. 2.20 or later version of the ADC FIT module.

Rev.	Date	Description	
		Page	Summary
2.20	Dec.01.2016	Program	Fixed the following issue: Target Device: RX64M/RX71M/RX65x/RX130/RX230/RX231 Description: When enabling self-diagnosis under a prohibited setting condition, the operation ends normally. Condition: Self-diagnosis is enabled in double trigger mode with single scan mode selected. Measure: Modified processing for checking the error condition when self-diagnosis is enabled. Use Rev. 2.20 or later version of the ADC FIT module.
			In RX63x and RX210, the TEMPS register is now modified only when the temperature sensor module is enabled.
			Added the definition "ADC_CONVERT_SPEED_DEFAULT" for conversion speed of A/D conversion in RX110, RX111, and RX113. "ADC_CONVERT_SPEED_DEFAULT" has the same value as "ADC_CONVERT_SPEED_NORM".
			Fixed the following issue: Target Device: RX110 Description: An error occurs when attempting to set the minimum value for the number of sampling states. Condition: An error occurs whenever the number of sampling states can be set. Measure: Modified the definition of the minimum value for the number of sampling states. Use Rev. 2.20 or later version of the ADC FIT module.
			In RX64M, RX71M, RX65x, RX130, RX230, and RX231, some function declarations differed from prototypes. These function declarations now correspond to the prototypes.
2.30	Jul.24.2017	—	Applications of descriptions are now indicated by the S12AD peripherals (not MCUs).
		—	Added support for RX65N-2MB (177 pins and 176 pins).
		—	Added support for RX130-512KB (100 pins).
		1	Related Documents: Added the following document: “Renesas e^2 studio Smart Configurator User Guide (R20AN0451)”
		3	1. Overview: Revised the descriptions.
		4	2.5 Supported Toolchains: The information of the toolchains are now described in 6.1.
		5	2.6 Interrupt Vector: Added
		6-7	2.10 Code Size: Updated the sizes according to changes in the program.
		7-43	2.11 API Data Structures: Revised. Now descriptions have given by each structure.
		44	2.13 Adding a FIT Module to Your Project: Revised.
		46-48	3.2 R_ADC_Open(): Revised.
		49-61	3.3 R_ADC_Control(): Revised.
		66	4. Pin Setting: Revised.

Rev.	Date	Description Page	 Summary
2.30	Jul.24.2017	68	5.8 Downloading Demo Projects: Added.
		69, 70	6. Appendices: Added.
		Program	In RX65N, deleted processing for checking the range of enum value to simplify the processing. * See the warning on compiling to check out-of-range for enum.
			Fixed the following issue: Target Device: RX130/RX230/RX231/RX64M/RX71M/RX65N Description: When a channel is opened in a mode other than group scan mode, even if the parameter only available for group scan mode is set for the channel, an error does not occur. Condition: When in a mode other than group scan mode, a channel for group B, channel for group C (RX65N only), and group priority control is set. Measure: Modified processing to check invalid combination in group scan mode and return an error. Use Rev. 2.30 or later version of the ADC FIT module.
			Fixed the following issue: Target Device: RX130/RX230/RX231/RX64M/RX71M/RX65N Description: The procedure to specify the register for group priority control does not follow the procedure in the User's Manual: Hardware. Due to this, scanning operation and the result stored cannot be guaranteed. Condition: Group priority control is used. Measure: Modified the register setting procedure for group priority control. Use Rev. 2.30 or later version of the ADC FIT module.
			Fixed the following issue: Target Device: RX65N Description: When the interrupt priority level is set (interrupt enabled) without specifying the callback function, an error does not occur. Condition: The interrupt priority level is set to 1 or greater. Measure: Modified the checking procedure at open to return an error. Use Rev. 2.30 or later version of the ADC FIT module.

		Description	
Rev.	Date	Page	Summary
2.30	Jul.24.2017	Program	Fixed the following issue: Target Device: RX65N Description: Even if addition mode is specified with an invalid combination, an error does not occur. Condition: When "sixteen samples" is selected for addition mode, 10-bit accuracy or 8-bit accuracy is selected. Measure: Modified the checking procedure at open to return an error. Use Rev. 2.30 or later version of the ADC FIT module.
			Fixed the following issue: Target Device: RX65N Description: The procedure to stop A/D conversion does not follow the procedure described in the User's Manual: Hardware. Due to this, an unexpected operation may be performed. Condition: Close processing is performed with group priority control enabled. Measure: Modified the register setting procedure at close. Use Rev. 2.30 or later version of the ADC FIT module.
			Fixed the following issue: Target Device: RX65N Description: Window B comparison condition may not be specified correctly. Condition: With comparison function, window B comparison condition is set to 2 or greater. Measure: Window B comparison condition does not have range check. Thus, the code has been modified to return an error when an out-of-range error occurs. Use Rev. 2.30 or later version of the ADC FIT module. Fixed the following issue: Target Device: RX65N Description: The trigger for group A cannot be set to the external trigger. Condition: Double trigger is disabled in group scan mode. Measure: The external trigger was disabled in RX65N (same as the RX64M). For RX65N, the external trigger now can be set only for group A. Use Rev. 2.30 or later version of the ADC FIT module.

Rev.	Date	Description	
		Page	Summary
2.30	Jul.24.2017	Program	Fixed the following issue: Target Device: RX65N Description: The result cannot be obtained when the window A/B complex condition is set. Condition: Occurs at any time. Measure: Added I/F for obtaining the comparison result with window A/B complex condition to the R_ADC_Control function. Use Rev. 2.30 or later version of the ADC FIT module.

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. 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 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. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment 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 moment 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 moment 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 moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has 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. Moreover, 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.

5. Differences between Products

   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of Microprocessing unit or Microcontroller unit products in the same group but having a different part number may differ in terms of the 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 disputes 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, drawing, chart, program, algorithm, application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or otherwise misappropriate 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, copy or otherwise misappropriation of Renesas Electronics products.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and 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.

   Renesas Electronics products are neither intended nor 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 damages (space and 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 third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. When using the 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 radiation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions or failure or accident arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please ensure to implement safety measures to guard them against the possibility of bodily injury, injury or damage caused by fire, and social damage in the event of 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 by your own responsibility as warranty for your products/system. Because the evaluation of microcomputer software alone is very difficult and not practical, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please investigate applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive carefully and sufficiently and use Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall not use Renesas Electronics products or technologies for (1) any purpose relating to the development, design, manufacture, use, stockpiling, etc., of weapons of mass destruction, such as nuclear weapons, chemical weapons, or biological weapons, or missiles (including unmanned aerial vehicles (UAVs)) for delivering such weapons, (2) any purpose relating to the development, design, manufacture, or use of conventional weapons, or (3) any other purpose of disturbing international peace and security, and you shall not sell, export, lease, transfer, or release Renesas Electronics products or technologies to any third party whether directly or indirectly with knowledge or reason to know that the third party or any other party will engage in the activities described above. When exporting, selling, transferring, etc., Renesas Electronics products or technologies, you shall comply with any applicable export control laws and regulations promulgated and administered by the governments of the countries asserting jurisdiction over the parties or transactions.
10. Please acknowledge and agree that you shall bear all the losses and damages which are incurred from the misuse or violation of the terms and conditions described in this document, including this notice, and hold Renesas Electronics harmless, if such misuse or violation results from your resale or making Renesas Electronics products available any third party.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.3.0-1 November 2016)

RENESAS

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL II Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141