RZ/T1 Group

ADC Sample Program

R01AN2599EJ0140
Rev.1.40
Jun. 07, 2018

Introduction

This application note explains a sample program that performs A/D conversion by using the ADC function after the 12-bit A/D converter (S12ADCa unit 0) of RZ/T1.

The major features of the ADC sample program are listed below.

- The input voltage to the potentiometer is A/D converted.
- Conversion results are classified into four scales, which are displayed to each of LED0, LED1, LED2, or LED3.

Target Devices

RZ/T1

When applying the sample program covered in this application note to another microcomputer, modify the program according to the specifications for the target microcomputer and conduct an extensive evaluation of the modified program.

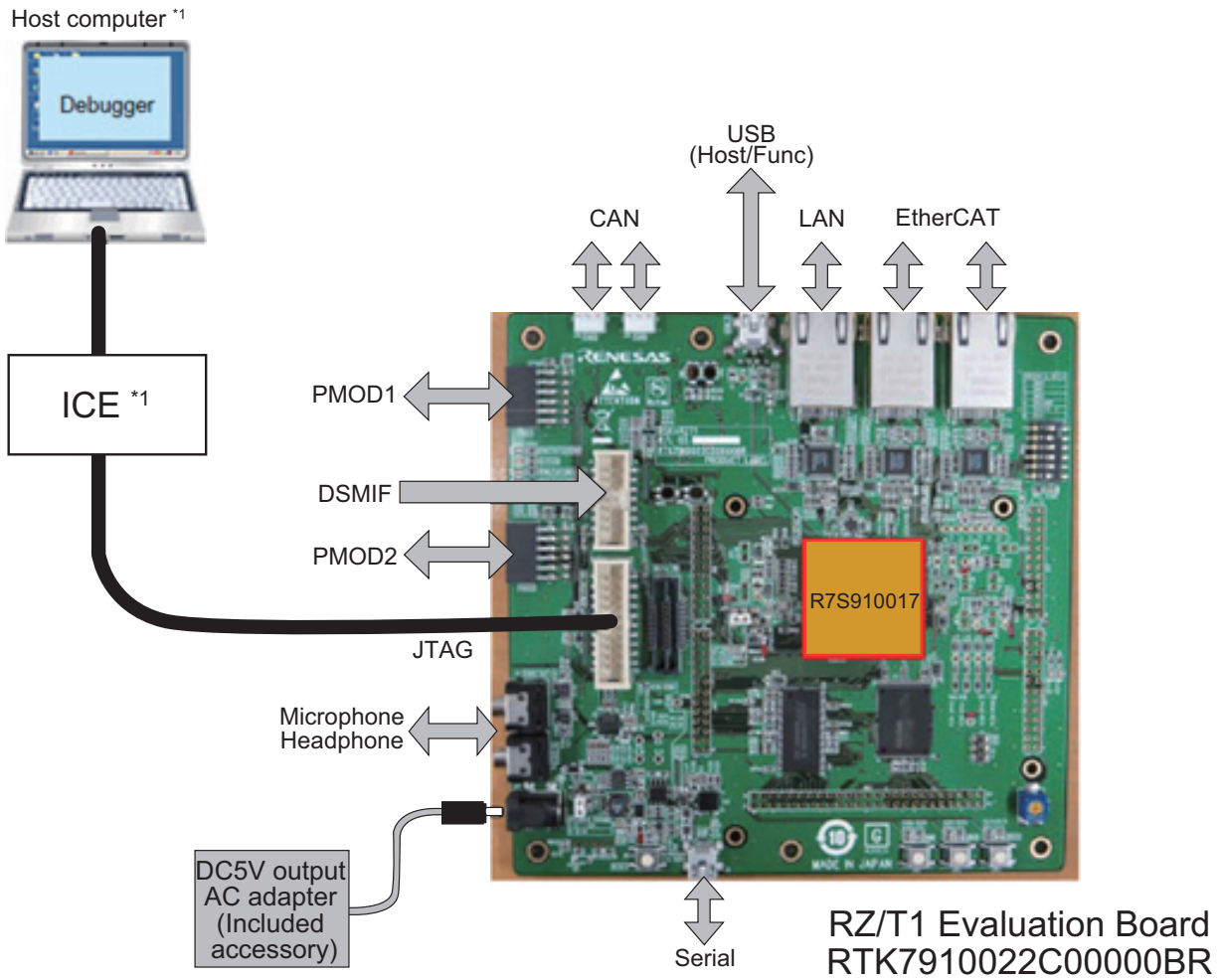
Table of Contents

1. Specifications

Table 1.1 lists the peripheral functions to be used and their applications and Figure 1.1 shows the operating environment when the sample code is being executed.

Table 1.1 Peripheral Functions and Applications

Peripheral Function	Application
Power consumption reduction function	Supplies and stops the clock to the ADC module
Interrupt controller (ICUA) (Interrupt ID: 35)	Accepts an A/D conversion complete interrupt request and issues an interrupt to the Cortex-R4.
Multi-function pin controller (MPC) (Assigned port: P44)	Set to the external trigger pin for starting A/D conversion ADTRG0#.
ADC (AN007)	A/D conversion of the input voltage to the potentiometer
Potentiometer	Inputs variable voltages to the ADC.



Note 1. Indicates the device that the user needs to prepare.

Figure 1.1 Operating Environment

2. Operating Environment

The sample code covered in this application note is for the environment below.

Table 2.1 Operating Environment

Item	Description
Microcomputer	RZ/T1 group
Operating frequency	CPUCLK = 450 MHz
Operating voltage	3.3 V
Integrated Development Environment	Manufactured by IAR Systems Embedded Workbench® for Arm Version 8.20.2 Manufactured by Arm DS-5™ 5.26.2 Manufactured by RENESAS e2studio 6.1.0
Operating mode	SPI boot mode 16-bit bus boot mode
Board	RZ/T1 Evaluation Board (RTK7910022C00000BR)
Device (functions to be used on the board)	• NOR flash memory (connected to CS0 and CS1 spaces) Manufacturer: Macronix International Co., Ltd. Model: MX29GL512FLT2I-10Q • SDRAM (connected to CS2 and CS3 spaces) Manufacturer: Integrated Silicon Solution Inc. Model: IS42S16320D-7TL • Serial flash memory Manufacturer: Macronix International Co., Ltd. Model: MX25L51245G • Potentiometer (AN007)

3. Related Application Note

The application note related to this application note is listed below for reference.

- Application Note: RZ/T1 Group Initial Settings (R01AN2554EJ)

Note: Registers not mentioned in this application note should be used at a value set in the Application Note: RZ/T1 Group Initial Settings.

4. Peripheral Functions

Refer to the RZ/T1 Group User's Manual: Hardware for the basics of the operating mode, power consumption reduction function, interrupt controller (ICUA), general-purpose input/output ports, and 12-bit A/D converter (S12ADCa).

5. Hardware

5.1 Hardware Configuration Example

Figure 5.1 shows a hardware configuration example.

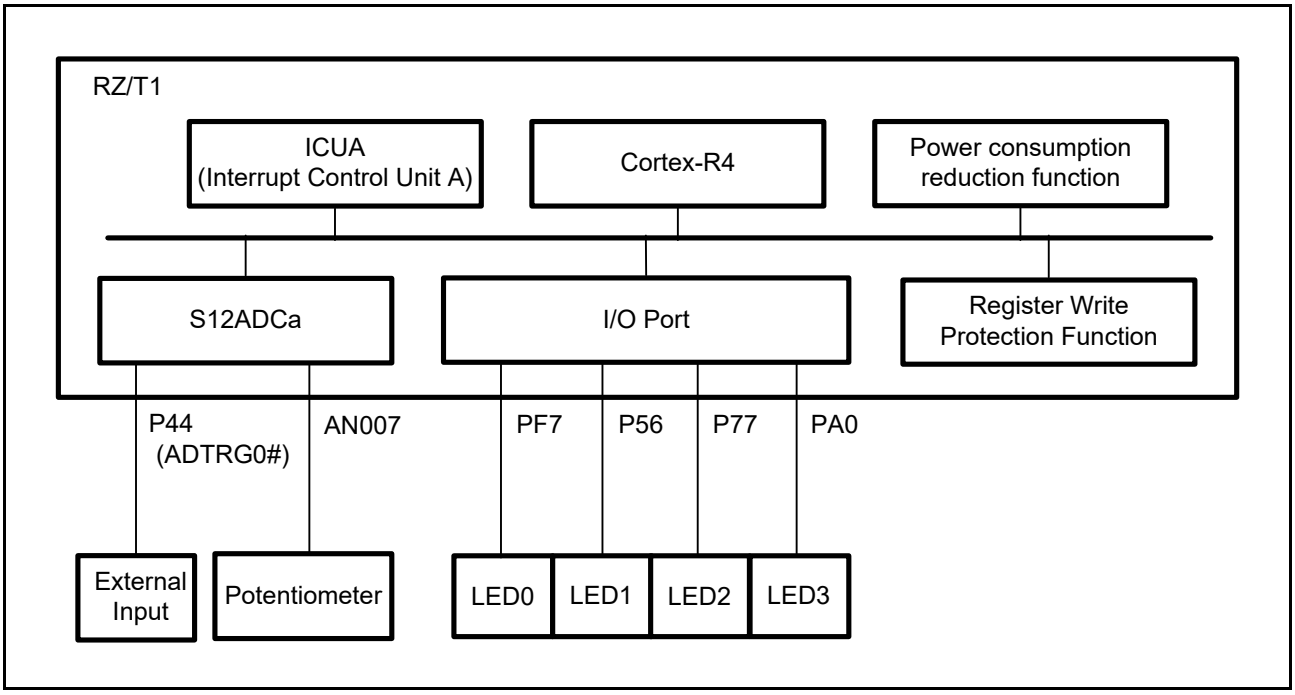


Figure 5.1 Hardware Configuration Example

5.2 Pins

Table 5.1 lists pins to be used and their functions.

Table 5.1 Pins and Functions

Pin Name	Input/Output	Function
AN007	Input	Potentiometer
P44/ADTRG0#	Input	External trigger pin for starting A/D conversion

6. Software

6.1 Operation Outline

Table 6.1 Operation Outline lists the functional outlines of the ADC sample program. Figure 6.1 shows the system block diagram.

Table 6.1 Operation Outline

Function	Outline
Input channel	Set to AN007, to which the potentiometer is connected.
Operation Mode	Set to the single scan mode for converting only AN007.
How to start A/D conversion	Select either of the following A/D conversion methods when building the program: • Software startup • External trigger startup
Acquisition of ADC conversion results	The results of A/D conversion are classified into four scales and displayed by using the LEDs*1 on the evaluation board that are assigned to each of the scales. Note 1. LED0, LED1, LED2, and LED3

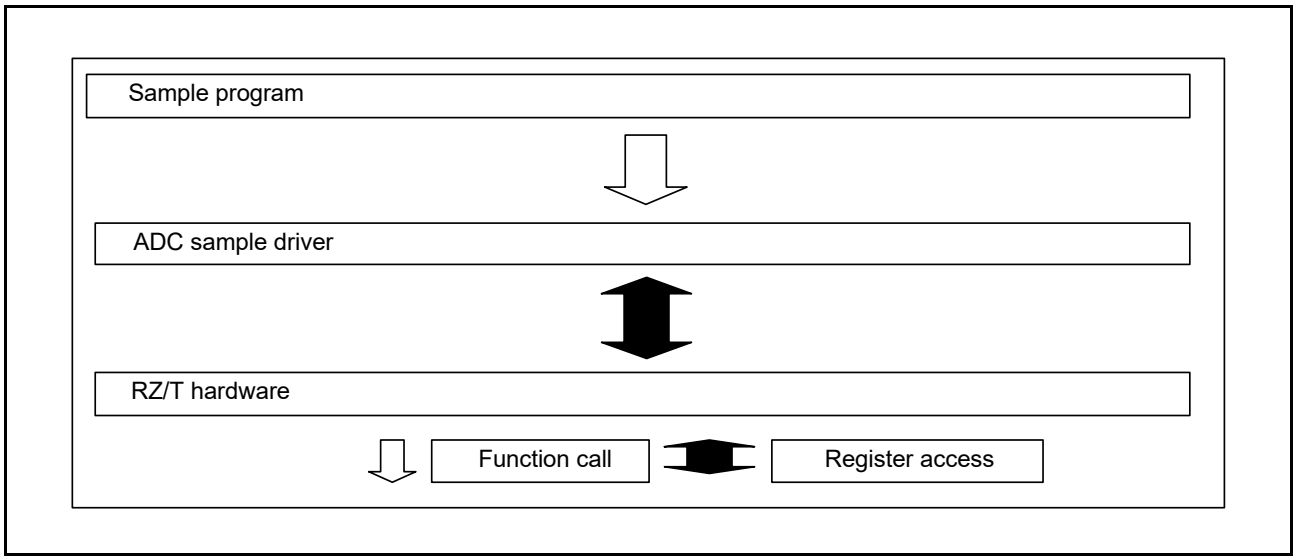


Figure 6.1 System Block Diagram

6.1.1 Project Settings

The project settings used on the development environment EWARM are described in the Application Note: RZ/T1 Group Initial Settings.

6.1.2 Preparation for Use

No preparation is required for executing this sample program.

6.2 Memory Map

The address space of the RZ/T1 group and the memory mapping of the RZ/T1 evaluation board are described in the Application Note: RZ/T1 Group Initial Settings.

6.2.1 Section Arrangement of the Sample Program

Sections used in the sample program, the section arrangement in the initial state of the sample program (load view), and the section arrangement after the scatter loading function is used (execution view) are described in the Application Note: RZ/T1 Group Initial Settings.

6.2.2 MPU Settings

The settings of the MPU are described in the Application Note: RZ/T1 Group Initial Settings.

6.2.3 Exception Handling Vector Table

The exception handling vector table is described in the Application Note: RZ/T1 Group Initial Settings.

6.3 Interrupts

Table 6.2 lists interrupts for the sample code.

Table 6.2 Interrupts for the Sample Code

Interrupt (Source ID)	Priority	Process Outline
A/D conversion complete interrupt	7	The callback function is called.

6.4 Fixed-Width Integer Types

Table 6.3 lists fixed-width integers for the sample code.

Table 6.3 Fixed-width Integers for the Sample Code

Symbol	Description
int8_t	8-bit signed integer (defined in the standard library)
int16_t	16-bit signed integer (defined in the standard library)
int32_t	32-bit signed integer (defined in the standard library)
int64_t	64-bit signed integer (defined in the standard library)
uint8_t	8-bit unsigned integer (defined in the standard library)
uint16_t	16-bit unsigned integer (defined in the standard library)
uint32_t	32-bit unsigned integer (defined in the standard library)
uint64_t	64-bit unsigned integer (defined in the standard library)

6.5 Constants/Error Codes

Table 6.4 shows the constants to be used in the sample code.

Table 6.4 Constants for Sample Code

Constant Name	Setting Value	Description
ADC_PORT_PDR_OUT	3u	I/O port output
ADC_PORT_PMR_IO_SET	0u	Setting the I/O ports to general-purpose input/output ports
ADC_LED_OFF	0u	Turning off LED0, LED1, LED2, and LED3.
ADC_LED_ON	1u	Turning on LED0, LED1, LED2, and LED3.
ADC_LED_COUNT	9999u	LED turn-on duty ratio generation counter constant
ADC_MPC_ADTRG0	0x09u	Assigning an alternative function to ADTRG#.
ADC_ADI_PRI	7u	Priority for the scan interrupt
ADC_LEVEL0	820u	Threshold A/D conversion value at which LED0 turns on
ADC_LEVEL1	1639u	Threshold A/D conversion value at which LED1 turns on
ADC_LEVEL2	2458u	Threshold A/D conversion value at which LED2 turns on
ADC_LEVEL3	3277u	Threshold A/D conversion value at which LED3 turns on
ADC_SAMPLE_TRIG	0	Compile switch for switching how to start up A/D conversion 0: Startup by a software trigger 1: Startup by an external trigger

6.6 Structures/Unions/Enumerated Types

Figure 6.2 shows structures/unions/enumerated types for the sample code.

```
typedef enum e_adc_mode
{
   ADC_MODE_SS_TEMPERATURE,            // single scan temperature sensor
   ADC_MODE_SS_ONE_CH,                 // single scan one channel
   ADC_MODE_SS_ONE_CH_DBLTRIG,         // on even triggers save to ADDBLDR & interrupt
   ADC_MODE_SS_MULTI_CH,               // 1 trigger source, scan multiple channels
   ADC_MODE_SS_MULTI_CH_GROUPED,       // 2 trigger sources, scan multiple channels
   ADC_MODE_SS_MULTI_CH_GROUPED_DBLTRIG_A,
   ADC_MODE_CONT_ONE_CH,               // continuous scan one channel
   ADC_MODE_CONT_MULTI_CH,             // continuous scan multiple channels
   ADC_MODE_MAX
} adc_mode_t;


typedef enum e_adc_trig                // trigger sources
{
   ADC_TRIG_ADTRG0 = 0,
   ADC_TRIG_TRGA0N = 1,
   ADC_TRIG_TRGA1N = 2,
   ADC_TRIG_TRGA2N = 3,
   ADC_TRIG_TRGA3N = 4,
   ADC_TRIG_TRGA4N = 5,
   ADC_TRIG_TRGA6N = 6,
   ADC_TRIG_TRGA7N = 7,
   ADC_TRIG_TRG0N = 8,
   ADC_TRIG_TRG4AN = 9,
   ADC_TRIG_TRG4BN = 10,
   ADC_TRIG_TRG4AN_OR_TRG4BN = 11,
   ADC_TRIG_TRG4ABN = 12,
   ADC_TRIG_TRG7AN = 13,
   ADC_TRIG_TRG7BN = 14,
   ADC_TRIG_TRG7AN_OR_TRG7BN = 15,
   ADC_TRIG_TRG7ABN = 16,
   ADC_TRIG_GTADTRA0N = 17,
   ADC_TRIG_GTADTRB0N = 18,
   ADC_TRIG_GTADTRA1N = 19,
   ADC_TRIG_GTADTRB1N = 20,
   ADC_TRIG_GTADTRA2N = 21,
   ADC_TRIG_GTADTRB2N = 22,
   ADC_TRIG_GTADTRA3N = 23,
   ADC_TRIG_GTADTRB3N = 24,
   ADC_TRIG_GTADTRA0N_OR_GTADTRB0N = 25,
   ADC_TRIG_GTADTRA1N_OR_GTADTRB1N = 26,
```

```
    ADC_TRIG_GTADTRA2N_OR_GTADTRB2N = 27,
    ADC_TRIG_GTADTRA3N_OR_GTADTRB3N = 28,
    ADC_TRIG_TPTRGAN_0 = 31,
    ADC_TRIG_TPTRG0AN_0 = 32,
    ADC_TRIG_TPTRGAN_1 = 33,
    ADC_TRIG_TPTRG6AN_1 = 34,
    ADC_TRIG_ELCTRG0 = 48,
    ADC_TRIG_SOFTWARE = 63
} adc_trig_t;


typedef enum e_adc_add
{
    ADC_ADD_OFF = 0,                        // addition is turned off for chans/sensors
    ADC_ADD_TWO_SAMPLES = 1,
    ADC_ADD_THREE_SAMPLES = 2,
    ADC_ADD_FOUR_SAMPLES = 3,
    ADC_ADD_MAX
} adc_add_t;


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


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


typedef struct st_adc_cfg
{
    adc_add_t       add_cnt;
    adc_align_t     alignment;          // ignored if addition used
    adc_clear_t     clearing;
    adc_trig_t      trigger;            // default and Group A trigger source
    adc_trig_t      trigger_groupb;             // valid only for group modes
    uint8_t         priority;           // S12ADIO interrupt priority; 0-15
    uint8_t         priority_groupb;            // GBADI interrupt priority; 0-15
} adc_cfg_t;
```

```
typedef enum e_adc_err      // ADC API error codes
{
   ADC_SUCCESS = 0,
   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_t;


typedef enum e_adc_cb_evt               // callback function events
{
   ADC_EVT_SCAN_COMPLETE,               // normal/Group A scan complete
   ADC_EVT_SCAN_COMPLETE_GROUPB     // Group B scan complete
} adc_cb_evt_t;


typedef struct st_adc_cb_args       // callback arguments
{
   adc_cb_evt_t   event;
} adc_cb_args_t;


typedef enum e_adc_cmd
{
   ADC_CMD_ENABLE_CHANS,              // enables chans and INT(s) if priority != 0
   ADC_CMD_ENABLE_TEMP_SENSOR,     // enables sensor and INT if priority != 0
   ADC_CMD_SET_SAMPLE_STATE_CNT,
   ADC_CMD_ENABLE_TRIG,               // allows an async/sync trigger to start scan
   ADC_CMD_DISABLE_TRIG,              // prevents an async/sync trigger to start scan
   ADC_CMD_SCAN_NOW,                    // issue software trigger
   ADC_CMD_DISABLE_INT,               // interrupt disable; ADCSR.ADIE=0
   ADC_CMD_ENABLE_INT,                 // interrupt enable;  ADCSR.ADIE=1
   ADC_CMD_DISABLE_INT_GROUPB,       // interrupt disable; ADCSR.GBADIE=0
   ADC_CMD_ENABLE_INT_GROUPB,         // interrupt enable;  ADCSR.GBADIE=1
   ADC_CMD_CHECK_SCAN_DONE,            // for Normal, GroupA or GroupB scan
   ADC_CMD_CHECK_SCAN_DONE_GROUPA,
   ADC_CMD_CHECK_SCAN_DONE_GROUPB,
   ADC_CMD_MAX
} adc_cmd_t;
```

```
typedef struct st_adc_ch_cfg          // bit 0 is ch0; bit 7 is ch7
{
    uint32_t       chan_mask;             // channels/bits 0-7
    uint32_t       chan_mask_groupb;   // valid for group modes
    uint32_t       add_mask;              // valid if add enabled in Open()
} adc_ch_cfg_t;

typedef enum e_adc_sst_reg         // sample state registers
{
    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_TEMPERATURE,
    ADC_SST_NUM_REGS
} adc_sst_reg_t;


typedef struct st_adc_time
{
    adc_sst_reg_t   reg_id;
    uint8_t        num_states;         // default=11
} adc_time_t;


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_TEMP = 8,
    ADC_REG_DBLTRIG = 9,
    ADC_REG_MAX
} adc_reg_t;
```

```
typedef struct st_adc_data
{
    uint16_t    chan[8];
    uint16_t    dbltrig;
} adc_data_t;
```

Figure 6.2 Structures/Unions/Enumerated Types for the Sample Code

6.7 Global Variables

Table 6.5 lists global variables.

Table 6.5 Global Variable

Type	Variable Name	Description	Function
volatile static bool	adc_end_flg	A/D conversion complete interrupt notification flag	main.c mian() adc_sample_callback()

6.8 Functions

Figure 6.7 lists functions.

Table 6.6 Functions

Function Name	Page Number
main	18
adc_sample_led_init	18
adc_sample_adtrg_init	19
adc_sample_callback	19
R_ADC_Open	20
R_ADC_Control	21
R_ADC_Read	21
R_ADC_ReadAll	22
R_ADC_Close	22
R_ADC_GetVersion	23
adc_s12adi0_isr	23
adc_gbadi_isr	23

6.9 Specification of Functions

The following shows the function specifications of the sample code.

6.9.1 main

main	
Synopsis	A/D conversion of the input voltage to the potentiometer
Header	–
Declaration	int_t main(void);
Description	This function performs the following processing. The A/D conversion results of the input voltages to the potentiometer that is connected on the evaluation board are classified into four scales and displayed by using the LEDs*1 on the evaluation board that are assigned to each of the scales. Note 1. LED0, LED1, LED2, and LED3
Arguments	None
Return values	None
Remarks	The A/D conversion start condition can be selected from software trigger or external trigger by changing the value of ADC_SAMPLE_TRIG in Table 6.4. The default setting is software trigger.

6.9.2 adc_sample_led_init

adc_sample_led_init	
Synopsis	Initialization of the pins connected to the LEDs
Header	–
Declaration	static void adc_sample_led_init(void);
Description	The following pins are set to general-purpose output ports. PF7: Connected to LED0 P56: Connected to LED1 P77: Connected to LED2 PA0: Connected to LED3
Arguments	None
Return values	None

6.9.3 adc_sample_led_off

adc_sample_led_off	
Synopsis	Turning of the LEDs
Header	–
Declaration	static void adc_sample_led_off (void);
Description	This function turns off LED0, LED1, LED2, and LED3.
Arguments	None
Return values	None

6.9.4 adc_sample_adtrg_init

adc_sample_adtrg_init	
Synopsis	Initialization of the external trigger ADTRG#0 pin
Header	–
Declaration	static void adc_sample_adtrg_init(void);
Description	The ADTRG#0 function is assigned to the P44 pin.
Arguments	None
Return values	None
Remarks	This function is executed when the A/D conversion start condition is set to external trigger (ADC_SAMPLE_TRIG is set to 1).

6.9.5 adc_sample_callback

adc_sample_callback	
Synopsis	ADC sample program callback function
Header	–
Declaration	static void adc_sample_callback(void *p_args_adc);
Description	This function is called from the adc_s12adi0_isr function, turning off LEDs that has been turned on for the display of the input voltage to the potentiometer.
Arguments	None
Return values	None
Remarks	Use the R_ADC_Open function to register this function. Enable the A/D conversion complete interrupt.

6.9.6 R_ADC_Open

R_ADC_Open		
Synopsis	ADC driver initialization function	
Header	r_adc_rzt1_if.h	
Declaration	adc_err_t R_ADC_Open(adc_mode_t const mode, adc_cfg_t * const p_cfg, void (* const p_callback)(void *p_args));	
Description	This function performs the following processing. • Checking the arguments • Setting the power consumption reduction function - Supplying a clock to the ADC (releasing S12ADCa of the module stop function) - Supplying a clock to the temperature sensor • Initial settings of the ADC - Setting the operating mode - Setting the A/D conversion startup condition - Setting the A/D-converted-value accumulation mode - Setting the format of the A/D data register • Initial settings of the ICUA - Setting the priority of the A/D conversion complete interrupt Setting the priority of the group B A/D conversion complete interrupt	
Arguments	adc_mode_t const mode	ADC driver initialization parameter The operating mode of the ADC driver is set.
	adc_cfg_t * const p_cfg	Initial setting of the ADC
	void (* const p_callback) (void *p_args)	The address of the callback function to be called from the A/D conversion complete interrupt handler
Return values	ADC_SUCCESS ADC_ERR_AD_NOT_CLOSED ADC_ERR_INVALID_ARG ADC_ERR_ILLEGAL_ARG ADC_ERR_MISSING_PTR	: The initialization of the ADC is completed normaly. : Duplicate initialization : Incorrect argument : Incorrect mode setting argument : Incorrect pointer argument
Remarks	This function must be executed before executing any API functions of the ADC driver. After this function is executed, a 1-us wait must be inserted before A/D conversion is started.	

6.9.7 R_ADC_Control

R_ADC_Control		
Synopsis	ADC function setting function	
Header	r_adc_rzt1_if.h	
Declaration	adc_err_t R_ADC_Control(adc_cmd_t const cmd, void * const p_args);	
Description	This function sets the functions of S12ADCa. Refer to the adc_cmd_t enumerated type.	
Arguments	adc_cmd_t const cmd	Setting the ADC functions to be used
	void * const p_args	Setting the A/D conversion channel Setting the sampling state
Return values	ADC_SUCCESS	: Function setting is completed normally.
	ADC_ERR_MISSING_PTR	: Pointer argument is NULL
	ADC_ERR_INVALID_ARG	: Incorrect argument value
	ADC_ERR_ILLEGAL_ARG	: Incorrect cmd
	ADC_ERR_SCAN_NOT_DONE	: A/D conversion not completed
Remarks	This function must be executed after the R_ADC_Open function.	

6.9.8 R_ADC_Read

R_ADC_Read		
Synopsis	Read function of an A/D converted value from a specified channel	
Header	r_adc_rzt1_if.h	
Declaration	adc_err_t R_ADC_Read(adc_reg_t const reg_id, uint16_t * const p_data);	
Description	This function reads out conversion results from the A/D data register, the A/D data duplicated register, and the A/D temperature sensor data register.	
Arguments	adc_reg_t const reg_id	Channel specified for reading out an A/D converted value
	uint16_t * const p_data	Pointer to the variable that stores an A/D converted value
Return values	ADC_SUCCESS	: Normal termination
	ADC_ERR_INVALID_ ARG	: Incorrect reg_id
	ADC_ERR_MISSING _PTR	: p_data is null
Remarks	This function must be executed while the ADC is not in operation.	

6.9.9 R_ADC_ReadAll

R_ADC_ReadAll		
Synopsis	Read out function of A/D converted values from all specified channels	
Header	r_adc_rzt1_if.h	
Declaration	adc_err_t R_ADC_ReadAll(adc_data_t * const p_all_data);	
Description	This function reads out conversion results from all channels of the A/D data register and the A/D data duplicated register.	
Arguments	adc_data_t * const p_all_data	Start address of the array that stores the A/D converted values
Return values	ADC_SUCCESS ADC_ERR_MISSING_PTR	: Normal termination : p_all_data is null
Remarks	This function must be executed while the ADC is not in operation.	

6.9.10 R_ADC_Close

R_ADC_Close	
Synopsis	End processing function of the ADC driver
Header	r_adc_rzt1_if.h
Declaration	void R_ADC_Close(void);
Description	This function performs the following processing. • End processing of the ICUA - Disabling the interrupt of S12ADI - Disabling the interrupt of S12GBADI • End processing of the ADC - Setting the A/D control register to the value after a reset • End processing of the temperature sensor - Setting the temperature sensor control register to the value after a reset • Setting the power consumption reduction function - Stopping the clock supply to the temperature sensor Stopping the clock supply to the ADC (disabling S12ADCa of the module stop function)
Arguments	None
Return values	None
Remarks	This function must be executed after the R_ADC_Open function. When asynchronous trigger or synchronous trigger is selected as the A/D conversion start condition, this function must be executed after the trigger is stopped. When ELC and/or EMU2 are used, this function must be executed after the following setting is made. Disabling event input from the ADC at the ELC Disabling data transfer from the ADC at the EMU2

6.9.11 R_ADC_GetVersion

R_ADC_GetVersion	
Synopsis	Acquisition function of the ADC driver version information
Header	r_adc_rzt1_if.h
Declaration	uint32_t R_ADC_GetVersion(void);
Description	The version of the ADC driver is returned as the return value.
Arguments	None
Return values	Version of the ADC driver

6.9.12 adc_s12adi0_isr

adc_s12adi0_isr	
Synopsis	A/D conversion complete interrupt handler
Header	–
Declaration	void adc_s12adi0_isr (void);
Description	The callback function that has been registered with the R_ADC_Open function is called.
Arguments	None
Return values	None

6.9.13 adc_gbadi_isr

adc_gbadi_isr	
Synopsis	Group B A/D conversion complete interrupt handler
Header	–
Declaration	static void adc_gbadi_isr(void);
Description	The callback function that has been registered with the R_ADC_Open function is called.
Arguments	None
Return values	None

6.10 Flowchart

6.10.1 Main Processing

Figure 6.3 shows the flowchart of the main processing.

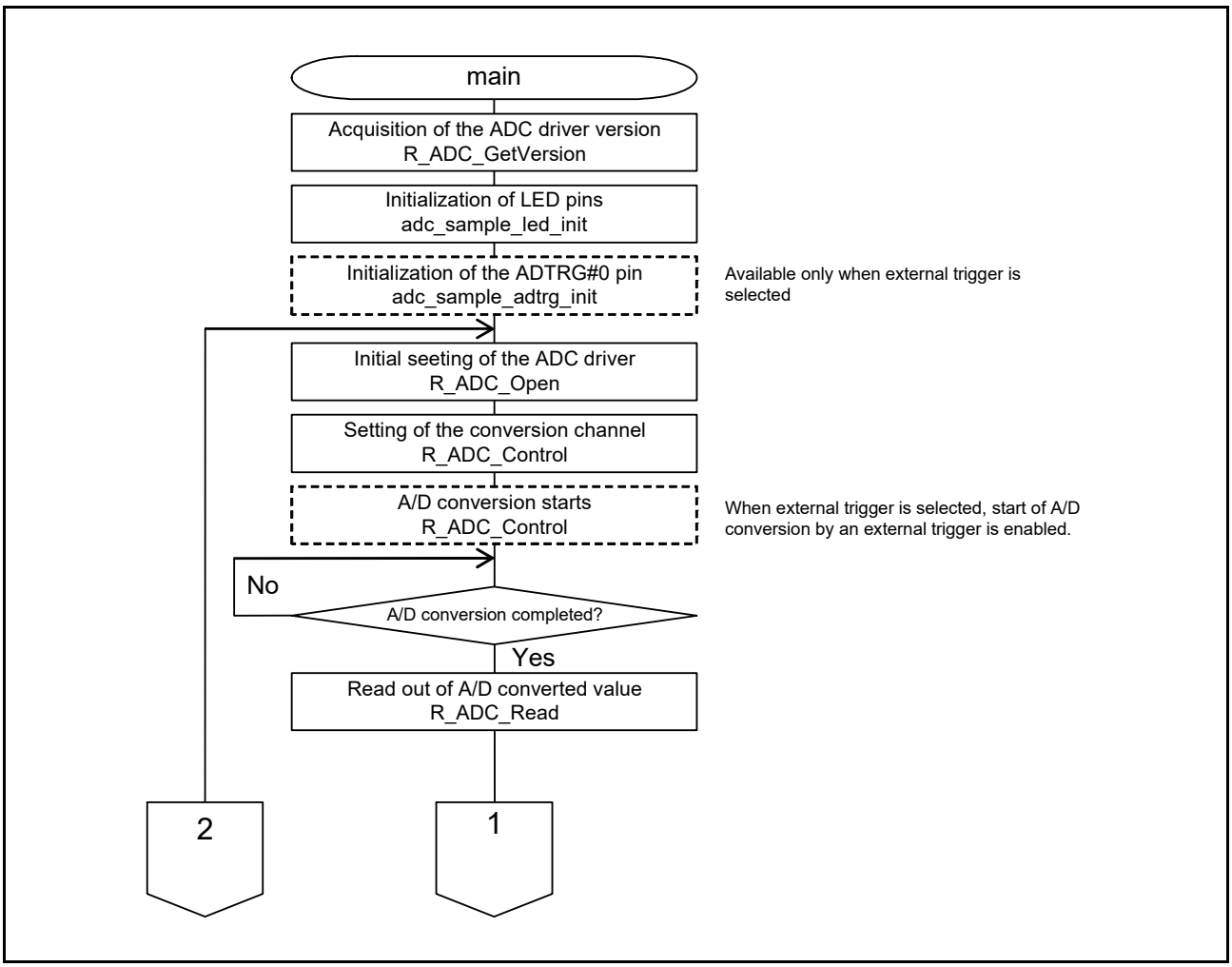


Figure 6.3 Main Processing (1 / 2)

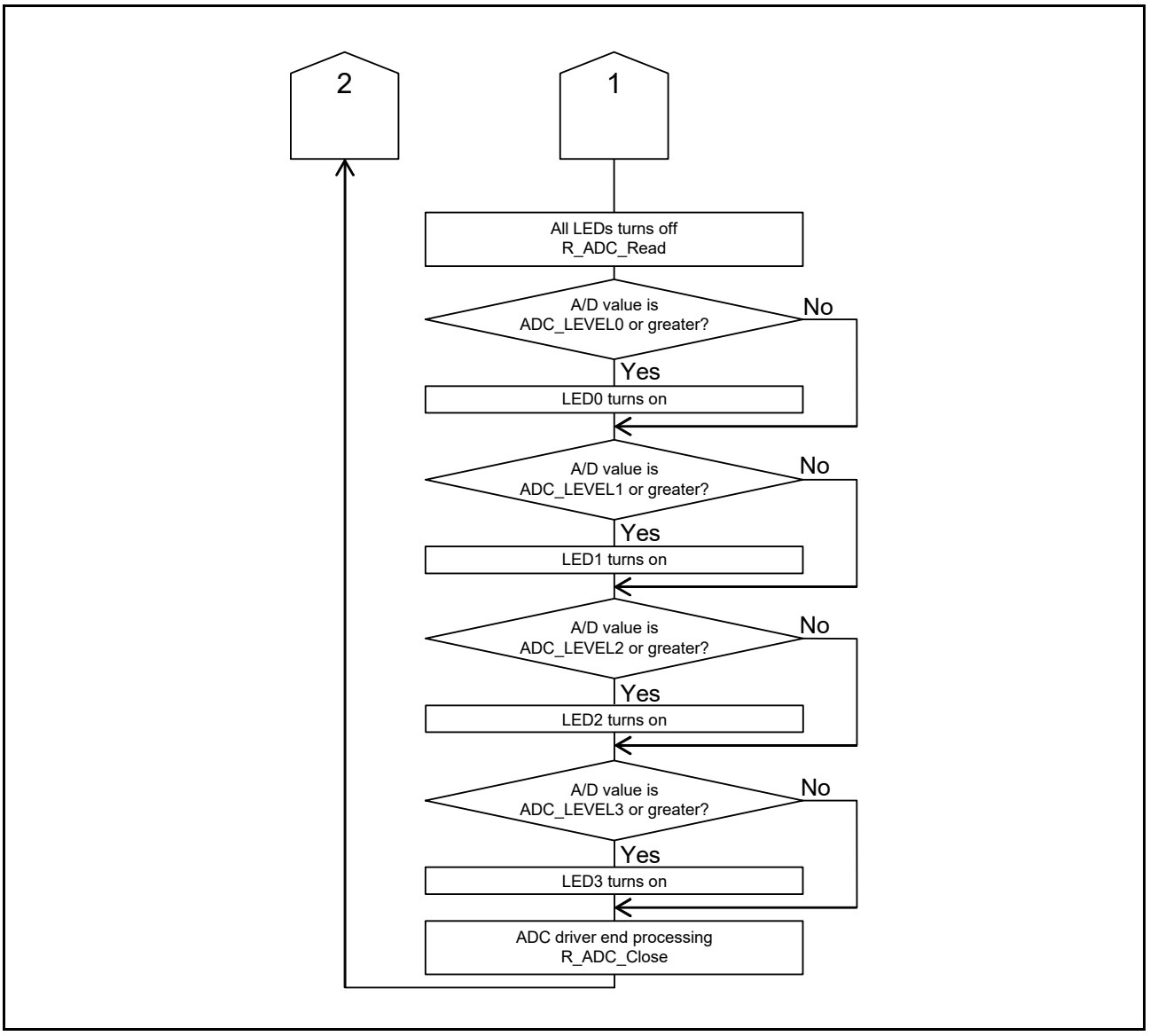


Figure 6.3 Main Processing (2 / 2)

6.10.2 adc_sample_led_init

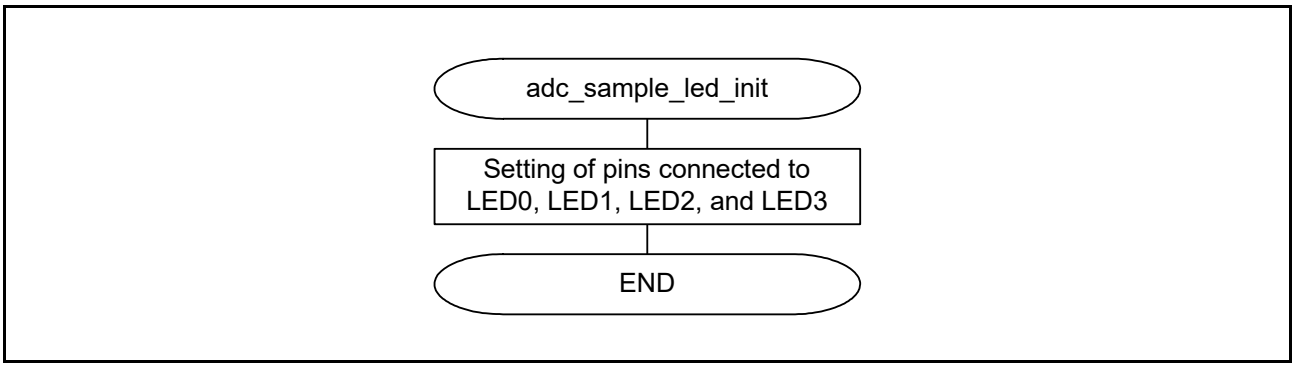


Figure 6.4 adc_sample_led_init

6.10.3 adc_sample_adtrg_init

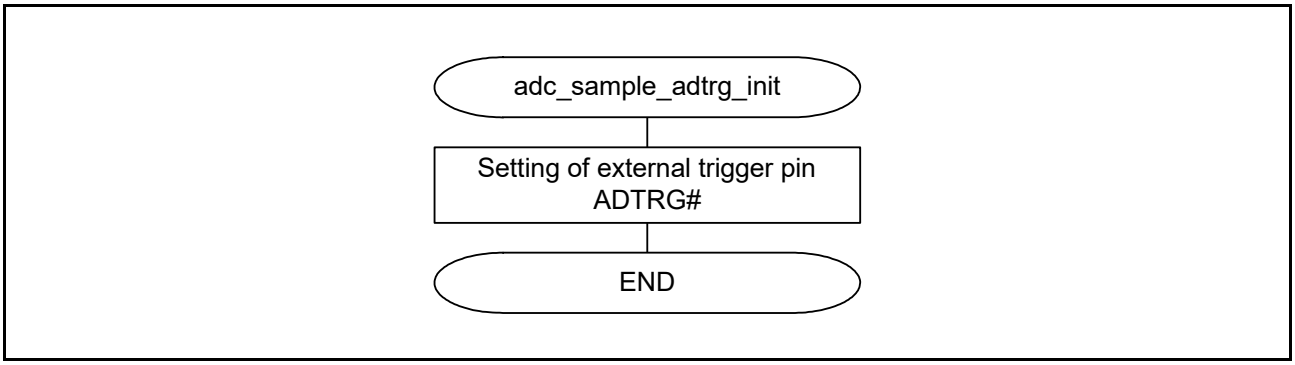


Figure 6.5 adc_sample_adtrg_init

6.10.4 adc_sample_callback

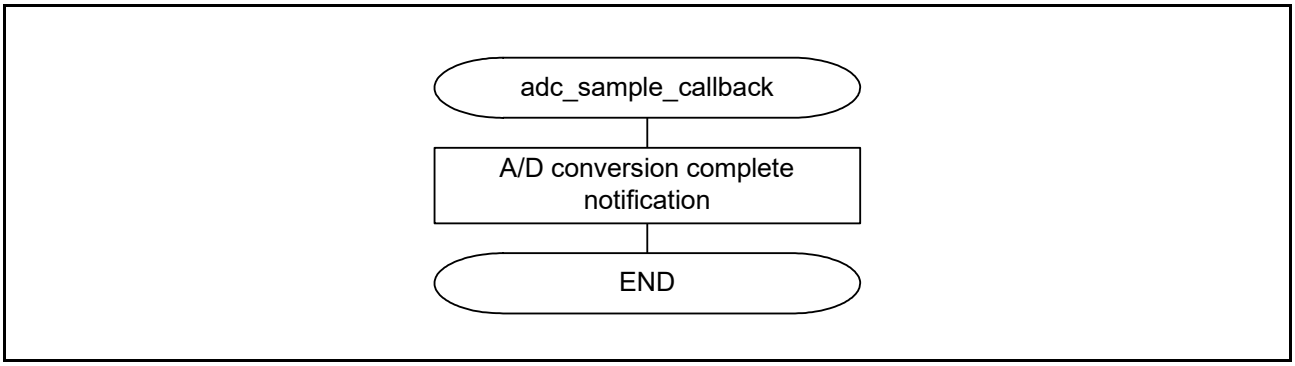


Figure 6.6 adc_sample_callback

6.10.5 adc_s12adi0_isr

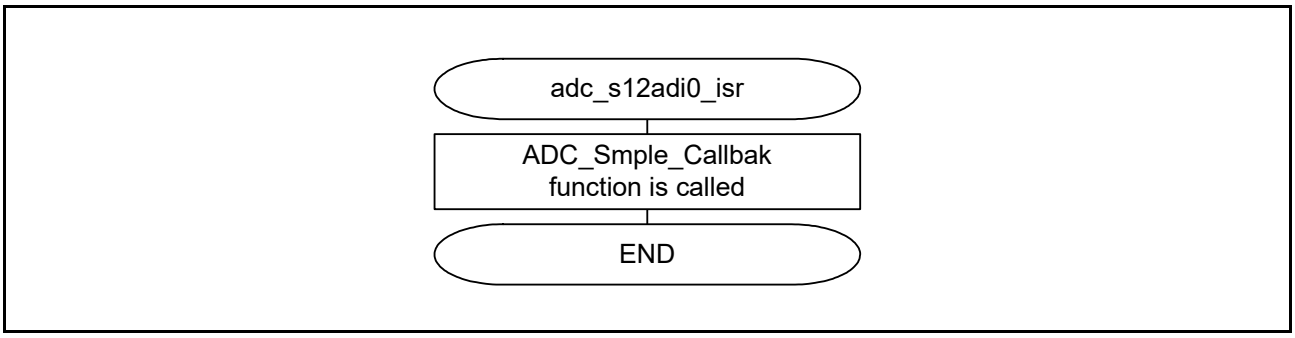


Figure 6.7 adc_s12adi0_isr

6.11 R_ADC_Control Commands

The following table lists commands for the R_ADC_Control function.

Table 6.7 Commands

Command	Outline
ADC_CMD_ENABLE_CHANS	Specification of the A/D conversion channel
ADC_CMD_ENABLE_TEMP_SENSOR	Initial settings of the temperature sensor
ADC_CMD_SET_SAMPLE_STATE_CNT	Setting of the sampling time for the analog input
ADC_CMD_ENABLE_TRIG	Enabling A/D conversion to be started by a synchronous or asynchronous trigger
ADC_CMD_DISABLE_TRIG	Disabling A/D conversion to be started by a synchronous or asynchronous trigger
ADC_CMD_SCAN_NOW	Starting A/D conversion by a software trigger
ADC_CMD_ENABLE_INT	Enabling S12ADI interrupt to be generated after scanning
ADC_CMD_DISABLE_INT	Disabling S12ADI interrupt to be generated after scanning
ADC_CMD_ENABLE_INT_GROUPB	Enabling S12GBADI interrupt to be generated after group B scanning
ADC_CMD_DISABLE_INT_GROUPB	Disabling S12GBADI interrupt to be generated after group B scanning
ADC_CMD_CHECK_SCAN_DONE	Checking A/D conversion
ADC_CMD_CHECK_SCAN_DONE_GROUPA	Checking group A scanning
ADC_CMD_CHECK_SCAN_DONE_GROUPB	Checking group B scanning

6.11.1 ADC_CMD_ENABLE_CHANS

ADC_CMD_ENABLE_CHANS		
Synopsis	Specification of the A/D conversion channel	
Header	r_adc_rzt1_if.h	
Description	This command specifies the A/D conversion channel. The parameters are delivered in a form of an adc_ch_cfg_t type valuable.	
Parameter	adc_ch_cfg_t p_args	The channel to which A/D conversion is conducted is specified.
Return values	ADC_SUCCESS	: Succeeded in the channel specification
	ADC_ERR_MISSING_PTR	: Pointer argument is null
	ADC_ERR_ILLEGAL_ARG	: The mode of the R_ADC_Open function is ADC_MODE_SS_TEMPERATURE
	ADC_ERR_INVALID_ARG	: Incorrect argument value
Remarks	–	

6.11.2 ADC_CMD_ENABLE_TEMP_SENSOR

ADC_CMD_ENABLE_TEMP_SENSOR		
Synopsis	Initial settings of the temperature sensor	
Header	r_adc_rzt1_if.h	
Description	This command initializes the temperature sensor. Null must be specified to the parameter because no parameter is required.	
Parameter	NULL	
Return values	ADC_SUCCESS	: Succeeded in the initialization of the temperature sensor
	ADC_ERR_ILLEGAL_ARG	: The mode of the R_ADC_Open function is ADC_MODE_SS_TEMPERATURE
Remarks	–	

6.11.3 ADC_CMD_SET_SAMPLE_STATE_CNT

ADC_CMD_SET_SAMPLE_STATE_CNT		
Synopsis	Setting of the sampling time for the analog input	
Header	r_adc_rzt1_if.h	
Description	This command specifies the sampling time for the analog input. The parameters are delivered in a form of an adc_time_t type valuable.	
Parameter	adc_time_t p_args	The channel to which a sampling time is set and its sampling time are specified. Refer to the adc_time_t structure.
Return values	ADC_SUCCESS	: Succeeded in the setting of the sampling time for the analog input
	ADC_ERR_MISSING_PTR	: Pointer argument is null
	ADC_ERR_ILLEGAL_ARG	: Incorrect argument value
Remarks	–	

6.11.4 ADC_CMD_ENABLE_TRIG

ADC_CMD_ENABLE_TRIG	
Synopsis	Enabling A/D conversion to be started by a synchronous or asynchronous trigger
Header	r_adc_rzt1_if.h
Description	This command enables A/D conversion to be started by a synchronous or asynchronous trigger. Null must be specified to the parameter because no parameter is required.
Parameter	NULL
Return values	ADC_SUCCESS : Succeeded in enabling A/D conversion to be started by a synchronous or asynchronous trigger
Remarks	–

6.11.5 ADC_CMD_DISABLE_TRIG

ADC_CMD_DISABLE_TRIG	
Synopsis	Disabling A/D conversion to be started by a synchronous or asynchronous trigger
Header	r_adc_rzt1_if.h
Description	This command disables A/D conversion to be started by a synchronous or asynchronous trigger. Null must be specified to the parameter because no parameter is required.
Parameter	NULL
Return values	ADC_SUCCESS : Succeeded in disabling A/D conversion to be started by a synchronous or asynchronous trigger
Remarks	–

6.11.6 ADC_CMD_SCAN_NOW

ADC_CMD_SCAN_NOW	
Synopsis	Starting A/D conversion by a software trigger
Header	r_adc_rzt1_if.h
Description	This command starts A/D conversion by as software trigger. Null must be specified to the parameter because no parameter is required.
Parameter	NULL
Return values	ADC_SUCCESS : Succeeded in starting A/D conversion ADC_ERR_SCAN_NOT_DONE : A/D conversion in progress
Remarks	–

6.11.7 ADC_CMD_ENABLE_INT

ADC_CMD_ENABLE_INT		
Synopsis	Enabling an S12ADI interrupt to be generated after scanning	
Header	r_adc_rzt1_if.h	
Description	This command enables S12ADI interrupt to be generated after scanning. Null must be specified to the parameter because no parameter is required.	
Parameter	NULL	
Return values	ADC_SUCCESS	: Succeeded in enabling S12ADI interrupt to be generated after scanning
	ADC_ERR_ILLEGAL_ARG	: No callback function has been registered
Remarks	–	

6.11.8 ADC_CMD_DISABLE_INT

ADC_CMD_DISABLE_INT		
Synopsis	Disabling an S12ADI interrupt to be generated after scanning	
Header	r_adc_rzt1_if.h	
Description	This command disables S12ADI interrupt to be generated after scanning. Null must be specified to the parameter because no parameter is required.	
Parameter	NULL	
Return values	ADC_SUCCESS	: Succeeded in disabling S12ADI interrupt to be generated after scanning.
Remarks	–	

6.11.9 ADC_CMD_ENABLE_INT_GROUPB

ADC_CMD_ENABLE_INT_GROUPB		
Synopsis	Enabling S12GBADI interrupt to be generated after group B scanning	
Header	r_adc_rzt1_if.h	
Description	This command enables S12GBADI interrupt to be generated after group B scanning. Null must be specified to the parameter because no parameter is required.	
Parameter	NULL	
Return values	ADC_SUCCESS	: Succeeded in enabling S12GBADI interrupt to be generated after group B scanning.
	ADC_ERR_ILLEGAL_ARG	: No callback function has been registered
Remarks	–	

6.11.10 ADC_CMD_DISABLE_INT_GROUPB

ADC_CMD_DISABLE_INT_GROUPB	
Synopsis	Disabling S12GBADI interrupt to be generated after group B scanning
Header	r_adc_rzt1_if.h
Description	This command disables S12GBADI interrupt to be generated after group B scanning. Null must be specified to the parameter because no parameter is required.
Parameter	NULL
Return values	ADC_SUCCESS : Succeeded in disabling S12GBADI interrupt to be generated after group B scanning
Remarks	–

6.11.11 ADC_CMD_CHECK_SCAN_DONE

ADC_CMD_CHECK_SCAN_DONE	
Synopsis	Checking A/D conversion
Header	r_adc_rzt1_if.h
Description	This command checks if A/D conversion is in progress. Null must be specified to the parameter because no parameter is required.
Parameter	NULL
Return values	ADC_SUCCESS : A/D conversion has been completed ADC_ERR_SCAN_NOT_DONE : A/D conversion is in progress
Remarks	–

6.11.12 ADC_CMD_CHECK_SCAN_DONE_GROUPA

ADC_CMD_CHECK_SCAN_DONE_GROUPA	
Synopsis	Checking group A scanning
Header	r_adc_rzt1_if.h
Description	This command checks if group A scanning has been completed. Null must be specified to the parameter because no parameter is required.
Parameter	NULL
Return values	ADC_SUCCESS : Group A scanning has been completed ADC_ERR_SCAN_NOT_DONE : Group A scanning is in progress
Remarks	–

6.11.13 ADC_CMD_CHECK_SCAN_DONE_GROUPB

ADC_CMD_CHECK_SCAN_DONE_GROUPB		
Synopsis	Checking group B scanning	
Header	r_adc_rzt1_if.h	
Description	This command checks if group B scanning has been completed. Null must be specified to the parameter because no parameter is required.	
Parameter	NULL	
Return values	ADC_SUCCESS	: Group B scanning has been completed
	ADC_ERR_SCAN_NOT_DONE	: Group B scanning is in progress
Remarks	–	

7. Sample Code

The sample code can be downloaded from the Renesas Electronics website.

8. Related Documents

- User's manual: Hardware
  RZ/T1 Group User's Manual: Hardware
  (Download the latest version from the Renesas Electronics website.)

  RZ/T1 Evaluation Board RTK7910022C00000BR User's Manual
  (Download the latest version from the Renesas Electronics website.)

- Technical Updates/Technical News
  (Download the latest information from the Renesas Electronics website.)

- User's Manual: Development Environment
  Download the IAR Embedded Workbench® for Arm from the IAR website.
  (Download the latest version from the IAR website.)

Website and Support

Renesas Electronics website

http://www.renesas.com/

Inquiries

http://www.renesas.com/inquiry

Revision History	Application Note: ADC Sample Program

Rev.	Date	Description	
		Page	Summary
0.10	Apr. 02, 2015	—	First Edition issued
1.00	Apr. 10, 2015	—	Only the revision number was changed to be posted on a website.
1.10	Jul. 16, 2015	2. Operating Environment	
		5	Table 2.1 Operating Environment: Description added to Integrated Development Environment
		6. Software	
		10	6.2.4 Required Memory Size: Description and reference added
		10	Table 6.2: Table title and size description were partially amended
		10	Table 6.2 Required Memory Size: Description on the Note and Size, changed
		11	Table 6.3 added
		11	Table 6.4 added
1.20	Dec. 04, 2015	2. Operating Environment	
		5	Table 2.1 Operating Environment: Integrated Development Environment, information partially amended
1.30	Jul. 13, 2017	All	"Cortex-R4F" changed to "Cortex-R4"
		1	Introduction: Description changed (unit 0 of S12ADCa added, LED numbers changed)
		2. Operating Environment	
		5	Table 2.1 Operating Environment: Integrated Development Environment, modified
		5. Hardware	
		8	Figure 5.1 Hardware Configuration Example: Pin names of the I/O port and LED numbers changed
		6. Software	
		—	6.2.4 Required Memory Size, deleted
1.40	Jun. 07, 2018	2. Operating Environment	
		5	Table 2.1 Operating Environment: The description on the integrated development environment, modified
		8. Related Documents	
		35	The name of IAR Embedded Workbench, modified

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 claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.

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

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

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

(Rev.4.0-1 November 2017)

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.
1001 Murphy Ranch Road, Milpitas, CA 95035, U.S.A.
Tel: +1-408-432-8888, Fax: +1-408-434-5351

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: +44-1628-651-700, Fax: +44-1628-651-804

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, 200333 P. R. China
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 2nd Stage, Indiranagar, Bangalore 560 038, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
17F, KAMCO Yangjae Tower, 262, Gangnam-daero, Gangnam-gu, Seoul, 06265 Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5338

Colophon 7.0