RL78/I1E

Analog Characteristics Evaluation Sample Code Specification

R01AN2819EJ0100
Rev.1.00
Nov. 09, 2015

Introduction

This application note describes the specifications of the sample code used to evaluate the analog characteristics of RL78/I1E (R5F11CCC).

Target Device

RL78/I1E (R5F11CCC)

Contents

1. Specifications

1.1 Outline

The sample code described in this application note controls the 24-bit ΔΣA/D converter with programmable gain instrumentation amplifier in the analog front-end circuit and enables communication with the PC.

ΔΣA/D control mainly performs A/D conversion, gain error determination, offset calibration, and auto-gain adjustment.

1.2 Operating Conditions

The sample code operations have been confirmed under the following conditions.

Table 1-1 Sample Code Operating Conditions

Item	Description
Evaluation board	・RL78/I1E analog evaluation board (RTK50011CCC00000BR) ・RL78/I1E CPU board (FB-R5F11CCC-TB)
MCU used	R5F11CCC (RL78/I1E) 36 pins
Operationg frequency	32MHz
Operating voltage	5.0V
Intergrated Development Environment (CS+ for CA,CX)	V3.01.00 (19 Aug 2015)
C compiler (CS+)	CA78K0R V5.00.00.02 (03 Jul 2014)
RL78/I1E Code Library (CS+)	V1.02.00.06 (12 Aug 2015) Note 1
Data Flash Library	RL78Family data flash library Type04 Ver.1.05

Note 1: The CS+ code library is bundled in the code generator plugin. Operations described in this application note have been confirmed for CS+ Code_Generator for RL78_78K V2.05.00.

2. Hardware Explanation

2.1 Hardware Resources

Table 2-1 and Table 2-2 describe the settings for the peripheral hardware used for the sample code.

Table 2-1 Hardware Resources (1/2)

Peripheral Function		Setting
Clock generator └Clock	Operating mode	High-speed main mode 2.7(V) ≦VDD≦5.5(V)
	Main system clock (fMAIN)	High-speed on-chip oscillator clock (fHOCO)
	24-bit ΔΣA/D converter operating clock (fDSAD) source setting	high-speed on-chip oscillator clock (fHOCO)
	RTC operating clock (fRTC) source setting	High-speed on-chip oscillator clock (fHOCO)
	High-speed on-chip oscillator clock (fHOCO) setting	32(MHz)
	High-speed system clock (fMX) setting	none
	24-bit ΔΣA/D converter operating clock (fDSAD) setting	fHOCO 32(MHz)
	RTC, interval timer/timer RJ operating clock	fIL 15(kHz)
	CPU and peripheral clock (fCLK)	fHOCO 32000(kHz)
Port functions	Port13	P13.7 (INTP)
Timer array Unit └Unit 1	Channel 0	Interval timer
	Interval period	10(ms)
		Generates INTTM10 interrupt at start of count
	Interrupt	Generates interrupt when timer channel 0 count is completed
		Level 3 (lowest priority)
Timer array Unit └Unit 1	Channel 1	Interval timer
	Operating mode setting	16 bits
	Interval timer setting	100(us)
	Interrupt	Generates interrupt when timer channel 1 count is completed.
		Level 3 (lowest priority)
Watchdog timer	Watchdog timer operation setting	Not used
PGA+ΔΣA/D converter	Selected multiplexer setting	Multiplexer 0 differential input mode
		No other multiplexers used
	SBIAS output voltage	2.0V
	Disconnection detection	Not used
	ΔΣA/D converter operation mode setting	Normal operations
	ΔΣA/D converter start trigger setting	Software trigger
	Auto-scan mode setting	Single scan
	Interrupt setting	ΔΣA/D conversion interrupt enable (INTDSAD)
		Level 3 (lowest priority)
		ΔΣA/D scan interrupt enable (INTDSADS)
		Level 3 (lowest priority)

Table 2-2 Hardware Resources (2/2)

Peripheral Function		Setting
PGA+ ΔΣA/D CONVERTER └multiplexer 0	Gain setting G_{SET1}	1x
	Gain setting G_{SET2}	8x
	Offset calibration voltage setting	16 (0mV)
	Oversampling rate	256
	A/D conversion count	Specify 1 to 8092 times as the setting value of register PGA0CTL2
		1 (1 time)
	Averaging procedure	Do not execute averaging procedure
Serial └SAU0 └Channel	Channel 1	UART0 transmission/reception function
	Data bit length	8 bits
	Data transfer direction	LSB
Serial └SAU0 └UART1 └Receive	Parity	No parity
	Stop bit length	1 bit
	Receive data level	Standard
	Transfer rate	1000000(bps)
	Interrupt	Receive completed interrupt setting (INTSR1)
		Level 3 (lowest priority)
	Call back function	Receive complete
Serial └SAU0 └UART1 └Send	Transfer mode	Continuous transfer mode
	Data bit length	8 bits
	Data transfer direction	LSB
	Parity	No parity
	Stop bit length	1 bit
	Transmit data level	Standard
	Transfer rate	1000000(bps)
	Interrupt	Transfer complete interrupt setting (INTST1)
		Level 3 (lowest priority)
	Call back function	Transfer complete
Data transfer control	DTC base address	0xffd00
	Control data0 (DTCD0)	UART1 reception
	Control data1(DTCD1)	UART1 transmission
Data transfer control └DTCD0	Transfer mode setting	Normal mode
	Transfer data size setting	8 bits
	Transfer origin address	0xFF46 (fixed)
	Transfer destination address	0xF900 (incremented)
	Number of transfers	1
	Block size	1
Data transfer control └DTCD1	Transfer mode setting	Normal mode
	Transfer data size setting	8 bits
	Transfer origin address	0xF910 (incremented)
	Transfer destination address	0xFF44 (fixed)
	Number of transfers	1
	Block size	1
Interrupt └External interrupt	INT10	INTP0 used
		Falling edge detected
		Level 3 (lowest priority)

2.2 Memory Address Space

Figure 2.1 shows the memory address space used by the sample code.

Part of the fixed address area in RAM must be reservered for the data flash library and the DTC table. To set this area, create a link directive file (*.dr) and register it in the CS+ IDE. For more details, refer to section "4.7 Link Directive File".

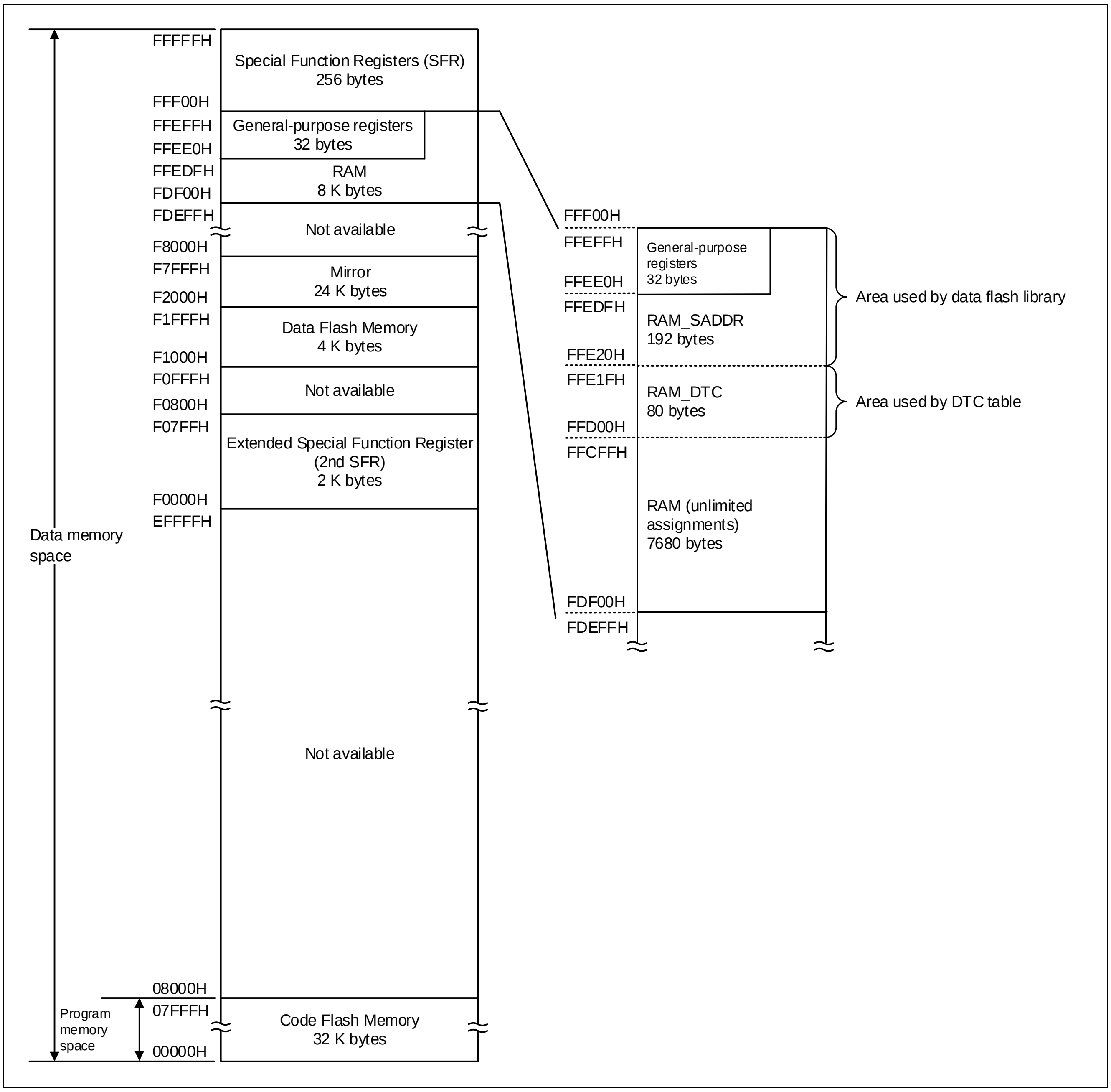


Figure 2.1 Memory Address Space Used by Sample Code

3. API Functions

3.1 Analog Front-end

Table 3-1 lists the API functions related to analog front-end (AFE).

Refer to r_rl78_i1e_common.c for the source files.

Table 3-1 Analog Front-end Related API Functions

Function Name	Description
R_I1E_Variable_Initialize	Variable initialization processings used in sample code
R_I1E_RingBuffer_Initialize	DSAD ring buffer initialization function
R_I1E_AFE_Calibration	AFE module calibration function
R_I1E_PGA_DSAD_GetResult_1Shot	24-bit ΔΣA/D converter specified channel 1-shot differential input A/D conversion function
R_I1E_PGA_DSAD_GainRegSet	24-bit ΔΣA/D converter gain setting function
R_I1E_PGA_DSAD_OffsetRegSet	24-bit ΔΣA/D converter offset setting function
R_I1E_PGA_DSAD_OsrRegSet	24-bit ΔΣA/D converter OSR setting function
R_I1E_PGA_DSAD_SettingRegGet	Auto-scan mode input multiplexer settings retentive variable storage function
R_I1E_PGA_DSAD_SettingRegSet	Set structure variable information to DSAD setting
R_I1E_PGA_DSAD_CorrectValue	Get PGA offset error correction value function
R_I1E_PGA_DSAD_GetValue	Get 24-bi tΔΣA/D converted value function
R_I1E_PGA_DSAD_AutoGainInit	PGA auto-gain variable initialization
R_I1E_PGA_DSAD_AutoGainBufCheck	24-bit ΔΣA/D converter conversion complete PGA auto-gain adjustment detection processing function
R_I1E_PGA_DSAD_AutoGainExecute	24-bit ΔΣA/D converter auto gain calibration function
R_I1E_PGA_DSAD_OffsetAdjustment	PGA offset adjustment execution function
R_I1E_PGA_DSAD_DisconnCheck	Disconnection detectionfunction
R_I1E_AFE_Stop	RL78/I1E AFE stop processing function
R_I1E_AFE_ReStart	RL78/I1E AFE restart processing function
R_I1E_CAMP_Calibration	Configuration amplifier tirmming function

3.2 Flash Memory-related Functions

Table 3-2 lists API functions related to the flash memory.

Refer to r_rl78_i1e_common.c for the source files.

Table 3-2 Flash Memory-related API Functions

Function Name	Description
R_I1E_FlashCheck	Data flash memory storage data check function
R_I1E_FlashRewrite	Data flash memory storage data rewrite function

Note Please refer to related document, Data Flash Library Type04.

3.3 UART Communication

Table 3-3 lists the API functions related to UART communication.

Refer to r_rl78_i1e_common.c for the source files.

Table 3-3 UART Communication API Functions

Function Name	Description
R_I1E_UartSend	PC transmission processing setting function (ASCII): limit of up to 256 bytes per transmission
R_I1E_UartSendBinary	PC transmission processing setting function (Binary): limit of up to 256 bytes per transmission
R_I1E_UartReceive	PC receive processing (DTC0): limit of up to 256 bytes per reception

3.4 Free Running Timer

Table 3-4 lists the API functions related to the free running timer.

Refer to r_rl78_common_util.c for the source files.

Table 3-4 Free Running Timer API Functions

Function Name	Description
R_TAU_FreeRunTimerInit	Free running timer setting initialization function
R_TAU_FreeRunTimerStop	Free running timer stop function
R_GetTickCount	Get free running timer tick count function
R_CmpTickCount	Free running timer count compare function

3.5 Key Judgement

Table 3-5 lists the API function related to key judgement.

Refer to r_keyscan.c for the source files.

Table 3-5 Key Judgement API Functions

Function Name	Description
R_KEY_Scan	Key scan processing
R_KEY_Initialize	Key information initialization
R_KEY_WaitOneClick	Get key information (specified key only)
R_KEY_GetAll	Get key information (all keys)
R_KEY_WaitOneClick	Specified key click wait

4. API Definitions

This section describes the definitions used in the Application Program Inteface functions, referred to as API functions.

4.1 Code Generating Function Definitions

This sample code includes header file r_cg_macrodriver.h, generated by the CS+ code generating function, and uses the following types.

Table 4-1 Type Definitions and Values

Definition	Value
int8_t	signed char
uint8_t	unsigned char
int16_t	signed short
uint16_t	unsigned short
int32_t	signed long
uint32_t	unsigned long
MD_STATUS	unsigned short

Most of the API functions included in this sample code return the status value in a common type name. The user application must judge the return value and continue with the processing as is if the return value is “successfully completed” or execute a correction process if “error.”

Table 4-2 MD_STATUS Type Return Values Used in Sample Code

Type Name	Macro Name	Constant Value	Description
MD_STATUS	MD_OK	00H	Successfully completed
	MD_ERROR	80H	Error
	MD_ARGERROR	81H	Paramter error

4.2 Macro Declarations for User Environment-dependent Settings

Areas that are dependent on the user environment and usage conditions are defined in this API. Please modify each definition as necessary according to your development environment.

(a) r_cg_main.c

Table 4-3 Macro Declarations for User Environment-dependent Settings

Macro Declaration	Default Setting Value	Input Range	Description
D_BULK_NUM	488U	1 or higher	Specify the total number of transfer data between the BULKSTART and BULKEND commands.
D_BULK_COMMAND_NUM	20U Note 1	1 or higher	Specify the number of transfer data for each BULK command.
D_STREAMHEADER	(header string) Note 2		STREAMHEADER transmission string

Note 1 Make sure the setting is within the range of the transmission buffer size (D_DSAD_VALUE_BUFFER_SIZE).

Note 2 Refer to the section about header transmission for STREAM transfers.

(b) r_rl78_i1e_common.h

Table 4-4 Macro Declarations for User Environment—dependent Settings

Macro Declaration	Default Setting Value	Input Range	Description
D_MCU_CLOCK_MHZ	32U	uint8_t [Note1]	Defines frequency for CPU and peripheral clocks 。[Note1, Note2] This is used to roughly calculate the time of the software wait used in the API. (MHz)
D_FLASH_MEMORY_DATA_USE	1U	0U,1U	Flash memory data use setting 0: Not used 1: Used
D_FLASH_FORCE_WRITING	0U	0U,1U	Flash data forced write setting 0: Do not forcibly overwrite (overwrite only when flash value is invalid) 1: Forcibly overwrite
D_DSAD_CORRECT_USE	1U	0U,1U	PGA error measurement enable setting 0: PGA error measurement disabled 1: PGA error measurement enabled
D_DSAD_CORRECT_MPXn	D_PGA_DSAD_MPX0	MPX0-MPX4[Note3]	Input multiplexer number used for PGA error measurement[Note 3]
D_DSAD_VALUE_BUFFER_SIZE	256U	uint16_t	Buffer size for DSAD converted value storage
D_DSAD_AUTO_GAIN_USE	1U	0U,1U	PGA auto-gain adjustment enable/disable setting 0: PGA auto-gain adjustment disabled 1: PGA auto-gain adjustment enabled
D_DSAD_AUTO_GAIN_TRIGGER_SEC	5U	uint8_t	Auto-gain adjustment timing (sec)
D_GAIN_ERROR_REFERENCE_mV	10.0F	float	PGA gain error measurement reference voltage (mV)
D_DISCONNECTION_CHECK_COUNT	3U[Note 1]	uint8_t	Disconnection check count (1 or more times)
D_DISCONNECTION_THRESHOLD_mV	10.0F	float	Disconnection measurement threshold (mV)
D_UART_SEND_USE	1U	0U,1U	UART transmission enable setting 0: UART transmission disabled 1: UART transmission enabled
D_UART_SEND_BUFFER_SIZE	256U	Max.256	Transmit buffer size for PC transmission

Note 1: Set a value higher than 0.
Note 2: Specify the setting value for the CPU clock of the MCU you are using.
Note 3: Specify the define declaration value of the input multiplexer number
D_PGA_DSAD_MPX0 = input multiplexer 0
D_PGA_DSAD_MPX1 = input multiplexer 1
D_PGA_DSAD_MPX2 = input multiplexer 2
D_PGA_DSAD_MPX3 = input multiplexer 3

(c) r_keyscan.h

Table 4-5 Macro Declarations for User Environment-dependent Settings

Macro Declaration	Default Setting Value	Input Range	Description
DEF_KEY_ACTIVE	DEF_KEY_ACTIVE_LOW	DEF_KEY_ACTIVE_LOW, DEF_KEY_ACTIVE_HIGH	SW active level
KEY_SCAN_NORM	10U	uint8_t	Single push determination time (10ms)
KEY_SCAN_LONG	100U	uint8_t	Long push determination time (10ms)
KEY_SCAN_DEAD	5U	uint8_t	Dead zone time after change in key state (10ms)
KEY_SCAN_NOT	5U	uint8_t	Non-active time (10ms)
KEY_SCAN_DOUBLE	50U	uint8_t	Double click determination enable time (10ms)

(d) r_cg_userdefine.h

Table 4-6 Macro Declarations for User Environment-dependent Settings

Macro Declaration	Default Setting Value	Input Range	Description
D_DEBUG_LED_USE	0U	0U,1U	Debug LED usage setting 0U : Not used 1U : Used
D_DEBUG_LED_PORT	P1.5	(output port) Note 1	Debug LED port setting

Note 1: Specify the digital output port connected to the pulled-up LED

4.3 Macro Declarations

This section describes the macro declarations defined in this API.

(a) r_cg_main.c

Table 4-7 Macro Declarations

Macro Declaration	Value	Description
D_BULK_SPLIT_NUM	D_BULK_NUM / _DSAD_VALUE_BUF_SIZE	BULK division number
D_BULK_SPLIT_MOD	D_BULK_NUM %D_DSAD_VALUE_BUF_SIZE	BULK division remainder
D_NO_ERROR	0x00U	No error
D_OVER_FLOW_ERROR	0x02U	Overflow error
D_DISCONNECTION_ERROR	0x80U	Disconnect error
D_ON_OFF_COMMAND	@0¥r¥n	ON/OFF command
D_COMMAND_SIZE	4U	Receive command size for PC communication

(b) r_rl78_i1e_common.h

Table 4-8 Macro Declarations

Macro Declaration	Value	Description
D_MCU_VOLTAGE_MODE	0U	Memory voltage mode (full-speed mode) (fixed)
D_PGA_DSAD_MPX0	0x00U	Input multiplexer 0
D_PGA_DSAD_MPX1	0x01U	Input multiplexer 1
D_PGA_DSAD_MPX2	0x02U	input multiplexer 2
D_PGA_DSAD_MPX3	0x03U	Input multiplexer 3
D_DSAD_AUTO_GAIN_TRIGGER_COUNT	D_DSAD_AUTO_GAIN_TRIGGER_SEC * D_MCU_CLOCK_MHZ * 1000000	PGA auto-gain adjustment timing Counter value
D_GAIN_ERROR_REFERENCE_LSB	(int32_t)((0x800000L * D_GAIN_ERROR_REFERENCE_mV / 0.8 / 1000L)	PGA gain error measurement reference voltage (LSB)
D_DISCONNECTION_THRESHOLD_LSB	(int32_t)(0x800000L * D_DISCONNECTION_THRESHOLD_mV / 0.8 / 1000L)	Disconnection determination threshold voltage (LSB)
D_UART_SEND_BUFFER_NUMBER	2U	Number of transmission buffers for PC communication
D_FLASH_SUCCESS	0U	Flash memory processing success
D_FLASH_MATCH	0U	Flash memory and RAM match
D_FLASH_MISSMATCH	1U	Flash memory mismatch
D_FLASH_INVALID	2U	Flash memory data invalid
D_FLASH_FAILURE	6U	Flash memory processing error
D_FLASH_DATA_STRUCT_SIZE	sizeof(str_flash_data_t) / sizeof(uint8_t)	Total structure size for flash storage variable (byte)
D_FLASH_DATA_CHECKSUM_SIZE	2U	Checksum size (byte)
D_CAMP_CH_NUMBER	3U	Number of configurable amp channels

(c) r_rl78_i1e_common.c

Table 4-9 Macro Declarations

Macro Declaration	Value	Description
D_DSAD_VALEU_MAX	(1L<<23) - 1	ΔΣA/D CONVERTER input voltage Max setting: +800 mV/(G_{TOTAL})
D_DSAD_VALEU_MIN	-1*(1L<<23)	ΔΣA/D CONVERTER input voltage Min setting: -800 mV/(G_{TOTAL})
D_DSAD_AUTO_GAIN_MAX_POS	0U	PGA auto-gain adjustment array index Max value of index
D_DSAD_AUTO_GAIN_MIN_POS	1U	PGA auto-gain calibration array index Min value of index

(d) r_rl78_common_util.h

Table 4-10 Macro Declarations

Macro Declaration	Value	Description
D_MCU_FREQUENCY_HZ	(D_MCU_CLOCK_MHZ * 1000000U)	Free running timer definition MCU operating clock frequency (Hz)
D_WAIT10S	(D_MCU_FREQUENCY_HZ * 10U - 1U)	Free running timer counter value: 10s
D_WAIT3S	(D_MCU_FREQUENCY_HZ * 3U - 1U)	Free running timer counter value: 3s
D_WAIT1S	(D_MCU_FREQUENCY_HZ * 1U - 1U)	Free running timer counter value: 1s
D_WAIT500MS	(D_MCU_FREQUENCY_HZ / 2U - 1U)	Free running timer counter value: 500ms
D_WAIT100MS	(D_MCU_FREQUENCY_HZ / 10U - 1U)	Free running timer counter value: 100ms
D_WAIT10MS	(D_MCU_FREQUENCY_HZ / 100U - 1U)	Free running timer counter value: 10ms
D_WAIT5MS	(D_MCU_FREQUENCY_HZ / 200U - 1U)	Free running timer counter value: 5ms
D_WAIT1MS	(D_MCU_FREQUENCY_HZ / 1000U - 1U)	Free running timer counter value: 1ms
D_WAIT100US	(D_MCU_FREQUENCY_HZ / 10000U - 1U)	Free running timer counter value: 100us
D_WAIT10US	(D_MCU_FREQUENCY_HZ / 100000U - 1U)	Free running timer counter value: 10us

(e) r_keyscan.h

Table 4-11 Macro Declarations

Macro Declaration	Value	Description
DEF_KEY_ACTIVE_LOW	0U	Port active level = LOW
DEF_KEY_ACTIVE_HIGH	1U	Port active level = HIGH

4.4 Type Declarations

This section describes independent types defined in the API.

4.4.1 Enumerations for User Environment-dependent Settings

The enumerations listed in Table 4-12 are used with the definitions stated in section “4.5 Global Constants for User Environment-dependent Settings” and must be modified according to those settings. Refer to section “4.5 Global Constants for User Environment-dependent Settings” for more details.

(a) r_keyscan.h

Table 4-12 Enumerations for Specifying Global Variables Storing Key Setting Information

Type Name	Default Definition	Description
e_key_t	SW_MODE_CHANGE	This is the enumurationg that indicates the index of the global variable structure array that stores key setting information such as the input port number. This is used when the get key information function is called. When adding a new key, also add a unique ENUM value. Example: SW_MODE_CHANGE = 0x00U, SW_GAIN_UP, // herein referred to as “added key” SW_GAIN_DOWN, SW_OFFSET_UP, SW_OFFSET_DOWN,
	SW_TYPE_MAX	This definition determines the size of the global variable structure array that stores key setting information. Therefore, do not change the name, delete or move from the end of the enumeration name.

Note: When making changes, make sure you match the above enumeration setting and the elements of the global variable structure array.

4.4.2 Enumerations

This section describes the enumeration declarations defined in the API.

(a) r_cg_main.c

Table 4-13 BULK Format Data Generation Control Enumeration

Type Name	Macro Name	Description
e_bulk_control_t	E_BULKDATA_BULKSTART	BULKSTART format data generation
	E_BULKDATA_BULK	BULK format data generation
	E_BULKDATA_BULKEND	BULKEND format data generation

Table 4-14 Commnication Command Generation Control Enumeration

Type Name	Macro Name	Description
e_communication_data_t	E_STREAMHEADER	STREAMHEADER format data generation
	E_STREAM	STREAM format data generation
	E_BULK	BULK format data generation
	E_BINARY	BINARY format data generation

Table 4-15 Commnication Command Judgement Enumeration

Type Name	Macro Name	Description
e_communication_data_t	E_ON_OFF_COMMAND	ON/OFF command
	E_COMMAND_NONE	Determines no command is present

Table 4-16 Measurement Control Enumeration

Type Name	Macro Name	Description
e_measurement_control_t	E_TRANSMISSION_START	Serial transmission start
	E_DATA_GENERATION	Generates measured value transmission data
	E_ERROR_CHECK	Error check
	E_WAIT	No operation

Table 4-17 MCU State Determination Enumeration

Type Name	Macro Name	Description
e_mcu_state_t	E_MCU_HALT	HALT mode
	E_MCU_RUN,	Now running

(b) r_rl78_i1e_common.h

Table 4-18 Input Multiplexer Specification Enumeration

Type Name	Macro Name	Value	Description
e_pga_dsad_mpx_t	E_PGA_DSAD_MPX0	0x00U	Input multiplexer 0
	E_PGA_DSAD_MPX1	0x01U	Input multiplexer 1
	E_PGA_DSAD_MPX2	0x02U	Input multiplexer 2
	E_PGA_DSAD_MPX3	0x03U	Input multiplexer 3
	E_PGA_DSAD_MPX4	0x04U	Input multiplexer 4 -> internal temperature sensor (fixed)
	E_PGA_DSAD_MPX_MAX	0x05U	Input multiplexer max judgement

Table 4-19 Input Mode Specification Enumeration

Type Name	Macro Name	Value	Description
e_pga_dsad_input_mode_t	E_PGA_DSAD_DIFF_INPUT	0x00U	Differential input mode
	E_PGA_DSAD_SINGLE_INPUT	0x01U	Single-ended input mode

Table 4-20 ΔΣA/D Conversion Enable/Disable Enumeration

Type Name	Macro Name	Value	Description
e_pga_dsad_onoff_t	E_PGA_DSAD_OFF	0x01U	A/D conversion OFF
	E_PGA_DSAD_ON	0x00U	A/D conversion ON

Table 4-21 PGA Gain Setting Specification Enumeration

Type Name	Macro Name	Value	Description
e_pga_dsad_gain_t	E_PGA_DSAD_GAIN_1_1_1	0x00U	G_{SET1} = x1, G_{SET2} = x1, G_{TOTAL} = x 1
	E_PGA_DSAD_GAIN_2_1_2	0x04U	G_{SET1} = x2, G_{SET2} = x1, G_{TOTAL} = x 2
	E_PGA_DSAD_GAIN_3_1_3	0x08U	G_{SET1} = x3, G_{SET2} = x1, G_{TOTAL} = x 3
	E_PGA_DSAD_GAIN_4_1_4	0x0CU	G_{SET1} = x4, G_{SET2} = x1, G_{TOTAL} = x 4
	E_PGA_DSAD_GAIN_8_1_8	0x10U	G_{SET1} = x8, G_{SET2} = x1, G_{TOTAL} = x 8
	E_PGA_DSAD_GAIN_1_2_2	0x01U	G_{SET1} = x1, G_{SET2} = x2, G_{TOTAL} = x 2
	E_PGA_DSAD_GAIN_2_2_4	0x05U	G_{SET1} = x2, G_{SET2} = x2, G_{TOTAL} = x 4
	E_PGA_DSAD_GAIN_3_2_6	0x09U	G_{SET1} = x3, G_{SET2} = x2, G_{TOTAL} = x 6
	E_PGA_DSAD_GAIN_4_2_8	0x0DU	G_{SET1} = x4, G_{SET2} = x2, G_{TOTAL} = x 8
	E_PGA_DSAD_GAIN_8_2_16	0x11U	G_{SET1} = x8, G_{SET2} = x2, G_{TOTAL} = x16
	E_PGA_DSAD_GAIN_1_4_4	0x02U	G_{SET1} = x1, G_{SET2} = x4, G_{TOTAL} = x 4
	E_PGA_DSAD_GAIN_2_4_8	0x06U	G_{SET1} = x2, G_{SET2} = x4, G_{TOTAL} = x 8
	E_PGA_DSAD_GAIN_3_4_12	0x0AU	G_{SET1} = x3, G_{SET2} = x4, G_{TOTAL} = x12
	E_PGA_DSAD_GAIN_4_4_16	0x0EU	G_{SET1} = x4, G_{SET2} = x4, G_{TOTAL} = x16
	E_PGA_DSAD_GAIN_8_4_32	0x12U	G_{SET1} = x8, G_{SET2} = x4, G_{TOTAL} = x32
	E_PGA_DSAD_GAIN_1_8_8	0x03U	G_{SET1} = x1, G_{SET2} = x8, G_{TOTAL} = x 8
	E_PGA_DSAD_GAIN_2_8_16	0x07U	G_{SET1} = x2, G_{SET2} = x8, G_{TOTAL} = x16
	E_PGA_DSAD_GAIN_3_8_24	0x0BU	G_{SET1} = x3, G_{SET2} = x8, G_{TOTAL} = x24
	E_PGA_DSAD_GAIN_4_8_32	0x0FU	G_{SET1} = x4, G_{SET2} = x8, G_{TOTAL} = x32
	E_PGA_DSAD_GAIN_8_8_64	0x13U	G_{SET1} = x8, G_{SET2} = x8, G_{TOTAL} = x64

Table 4-22 Offset Voltage Setting Specification Enumeration

Type Name	Macro Name	Value	Description
e_pga_dsad_offset_t	E_PGA_DSAD_OFFSET_164p06	0x1FU	164.06/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_153p13	0x1EU	153.13/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_142p19	0x1DU	142.19/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_131p25	0x1CU	131.25/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_120p31	0x1BU	120.31/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_109p38	0x1AU	109.38/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_98p44	0x19U	98.44/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_87p50	0x18U	87.50/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_76p56	0x17U	76.56/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_65p63	0x16U	65.63/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_54p69	0x15U	54.69/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_43p75	0x14U	43.75/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_32p81	0x13U	32.81/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_21p88	0x12U	21.88/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_10p94	0x11U	10.94/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_0p00	0x10U	0.00/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_M10p94	0x0FU	-10.94/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_M21p88	0x0EU	-21.88/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_M32p81	0x0DU	-32.81/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_M43p75	0x0CU	-43.75/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_M54p69	0x0BU	-54.69/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_M65p63	0x0AU	-65.63/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_M76p56	0x09U	-76.56/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_M87p50	0x08U	-87.50/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_M98p44	0x07U	-98.44/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_M109p38	0x06U	-109.38/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_M120p31	0x05U	-120.31/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_M131p25	0x04U	-131.25/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_M142p19	0x03U	-142.19/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_M153p13	0x02U	-153.13/G_{SET1} [mV]
	E_PGA_DSAD_OFFSET_M164p06	0x01U	-164.06/G_{SET1} [mV]

Table 4-23 OSR Setting Specification Enumeration

Type Name	Macro Name	Value	Description
e_pga_dsad_osr_t	E_PGA_DSAD_OSR_64	0x00U	15625.000 [sps]
	E_PGA_DSAD_OSR_128	0x01U	7812.500 [sps]
	E_PGA_DSAD_OSR_256	0x02U	3906.250 [sps]
	E_PGA_DSAD_OSR_512	0x03U	1953.125 [sps]
	E_PGA_DSAD_OSR_1024	0x04U	976.563 [sps]
	E_PGA_DSAD_OSR_2048	0x05U	488.281 [sps]

Table 4-24 ΔΣA/D Conversion Count Calcuation Method Specification Enumeration

Type Name	Macro Name	Value	Description
e_pga_dsad_count_mode_t	E_PGA_DSAD_COUNT_CALCULATION	0x00U	Specify 1 to 8,032 times by using the value set in the PGAxCTL2 register (default)
	E_PGA_DSAD_COUNT_LINEAR	0x01U	Specify 1 to 255 times linearly by using the value set in the PGAxCTL2 register

Table 4-25 Averaging Procedure Operation Selection Enumeration

Type Name	Macro Name	Value	Description
e_pga_dsad_average_operation_t	E_PGA_DSAD_DO_NOT_AVERAGE_1	0x00U	Do not execute averaging procedure
	E_PGA_DSAD_DO_NOT_AVERAGE_2	0x01U	Do not execute averaging procedure
	E_PGA_DSAD_AVERAGE_INT_AN_ADC	0x02U	Execute averaging procedure, generate INTDSAD for each A/D conversion
	E_PGA_DSAD_AVERAGE_INT_UPDATE	0x03U	Execute averaging procedure, generate INTDSAD for each average update

Table 4-26 Averaging Data Selection Enumeration

Type Name	Macro Name	Value	Description
e_pga_dsad_average_number_t	E_PGA_DSAD_AVERAGE_8	0x00U	Number of data for averaging: 8
	E_PGA_DSAD_AVERAGE_16	0x01U	Number of data for averaging: 16
	E_PGA_DSAD_AVERAGE_32	0x02U	Number of data for averaging: 32
	E_PGA_DSAD_AVERAGE_64	0x03U	Number of data for averaging: 64

Table 4-27 ΔΣA/D Conversion (AUTOSCAN) Start/Stop Specification Enumeration

Type Name	Macro Name	Value	Description
e_pga_dsad_start_stop_t	E_PGA_DSAD_STOP	0x00U	Stop
	E_PGA_DSAD_START	0x01U	Start

Table 4-28 Auto Scan Mode Specification Enumeration

Type Name	Macro Name	Value	Description
e_pga_dsad_autoscan_mode_t	E_PGA_DSAD_CONTINUOUS	0x00U	Continuous scan mode
	E_PGA_DSAD_SINGLE	0x01U	Single scan mode

Table 4-29 PGA Auto-gain Adjustment Return Value Specification Enumeration

Type Name	Macro Name	Value	Description
e_auto_gain_return_t	E_AUTO_GAIN_NO_ADJUSTMENT	0x00U	No PGA auto-gain adjustment
	E_AUTO_GAIN_ADJUSTMENT	0x01U	Execute PGA auto-gain adjustment

Table 4-30 Disconnection Detection Processing Return Value Specification Enumeration

Type Name	Macro Name	Value	Description
e_disconnection_return_t	E_DISCONNECTION_NO_DETECT	0x00U	Disconnection not detected
	E_DISCONNECTION_DETECT	0x01U	Disconnection detected

(c) r_keyscan.h

Table 4-31 Key State Specification Enumeration

Type Name	Macro Name	Description
e_key_status_t	E_KEY_OFF	Key in OFF state
	E_KEY_OFF_TO_ON	Transition from key OFF to key normal push state
	E_KEY_ON_NORM	Key in normal push state
	E_KEY_NORM_TO_LONG	Transition from key in normal push state to key in long push state
	E_KEY_ON_LONG	Key in long push state
	E_KEY_LONG_TO_LONG	Transition from key in long push state to key in long push state
	E_KEY_DOUBLE_CLICK	Key in double click state

4.4.3 Structures

This section describes the structure declarations defined in the API.

(a) r_cg_main.c

Table 4-32 Structure for Variables Storing Measured Data

Structure Name	measurement_data_t		
Outline	Definition of variables storing measured data		
Member Variable	Type	Name	Description
	uint32_t	count	ΔΣA/D conversion counter
	str_pga_dsad_value_t	dsad_value	ΔΣA/D converted value structure variable
	e_mcu_state_t	mcu_state	MCU operating state
	uint8_t	error_state	Error state

Table 4-33 Structure for Variables Storing BULK Transfer Data

Structure Name	bulk_data_t		
Outline	Definition of variables storing bulk transfer data		
Member Variable	Type	Name	Description
	uint16_t	send_count	Bulk transfer send counter
	e_bulk_control_t	control	Bulk transfer control
	int32_t	buf[D_BULK_BUFFER_SIZE]	BULK transfer data buffer
	uint16_t	str_count	BULK transfer data buffer counter
	uint16_t	split_count	Bulk transfer split send counter

(b) r_rl78_i1e_common.h

Table 4-34 Structure for Variables Storing ΔΣA/D Conversion Information

Structure Name	str_pga_dsad_setting_t		
Outline	Definition of variables storing setting values for PGA and ΔΣA/D control registers		
Member Variable	Type	Name	Description
	e_pga_dsad_onoff_t	dsad_onoff	A/D conversion enabled/disabled flag
	e_pga_dsad_input_mode_t	dsad_input_mode	Input mode
	e_pga_dsad_offset_t	dsad_offset	Offset register setting value
	e_pga_dsad_osr_t	dsad_osr	Oversampling rate
	e_pga_dsad_gain_t	dsad_gain	Gain
	uint8_t	dsad_count	Auto-scan number counter
	e_pga_dsad_count_mode_t	dsad_count_mode	A/D conversion count calculation method specification
	e_pga_dsad_average_operation_t	dsad_average_operation	Selection of averaging process operation
	e_pga_dsad_average_number_t	dsad_average_number	Selection of number of data for averaging

Table 4-35 Structure for Buffer Storing ΔΣA/D Converted Value

Structure Name	str_pga_dsad_value_t		
Outline	Definition for variables storing each type of information such as ΔΣA/D converted value, gain, offset, etc.		
Member Variable	Type	Name	Description
	uint32_t	count	Measured number counter value
	uni_long_t	adc_value	A/D converted value (24 bits, right-aligned)
	uint8_t	ch	Number of channels
	uint8_t	overflow	Overflow flag
	uint8_t	gain_set_1	Preamplifier gain multiplier
	uint8_t	gain_set_2	Post amplifier gain multiplier
	uint8_t	gain_total	Total gain multiplier
	e_pga_dsad_offset_t	offset_reg	Offset register setting value
	int32_t	adc_correct	A/D conversion correction (24 bits, right-aligned)

Table 4-36 Structure for Flash Storage Variables

Structure Name	str_flash_data_t		
Outline	Definition for flash storage variables		
Member Variable	Type	Name	Description
	int32_t	gain_real[5U]	Actual gain setting value G_{SET1} x fixed G_{SET2} = G_{SET1} [x1x2x3x4x8] x G_{SET2} x 100 (fixed point)
	int32_t	offset_factor[5U][32U]	PGA offset error correction value Offset 31 stages x preamplifier gain 5 stages = 155 stages retained [m][n]: m = gain setting, n = offset setting value
	uint16_t	checksum	Checksum value (16bit)

(c) r_keyscan.h

Table 4-37 Structure forKey Information Setting Variables

Structure Name	key_setting_t		
Outline	Definition of key information setting variables		
Member Variable	Type	Name	Description
	uint8_t *	p_port_addr	Address of port register connected to SW pin
	uint8_t	port_bit_num	Bit number of port register connected to SW pin
	uint8_t	multiple_group	Group number for determining invalid simultaneous push

4.4.4 Unions

This section describes the union declarations defined in the API.

(a) r_rl78_i1e_common.h

Table 4-38 Signed Long/Unsigned Long Read Unions

Structure Name	uni_long_t			
Outline	Definition of union for reading signed/unsigned data			
Member Variable	Type	Name	Description	Type
	int32_t	32	LONG	signed long
	uint32_t	32	uLONG	unsigned long

Table 4-39 ΔΣA/D Conversion Result Register C Read Union

Structure Name	uni_pga_dsad_conversion_result_t			
Outline	Definition of union for reading ΔΣA/D conversion result register C			
Member Variable	Type	No. of bits	Name	Description
	uint8_t	8	BYTE	8-bit access
	BIT	4	-	Not used
		1	overflow	Overflow Flag
		3	ch	Ch number

Table 4-40 ΔΣA/D Conversion Results Register Read Union

Structure Name	uni_pga_dsad_value_t				
Outline	Definition of union for reading ΔΣA/D conversion results registers				
Member Variable	Type		No. of bits	Name	Description
	uint32_t		32	LONG	32-bit access
	uint16_t		32	WORD[2]	16-bit access
	BYTE	uni_pga_dsad_conversion_result_t	8	dsad_c	ΔΣA/D conversion results register C
		uint8_t	8	dsad_l	ΔΣA/D conversion results register L
		uint8_t	8	dsad_m	ΔΣA/D conversion results register M
		uint8_t	8	dsad_h	ΔΣA/D conversion results register H

Table 4-41 PGAxCTL0 Register Gain Setting Union

Structure Name	uni_pga_dsad_gain_t			
Outline	Definition of union for reading PGAxCTL0 register gain setting			
Member Variable	Type	No. of bits	Name	Description
	uint8_t	8	BYTE	8-bit access
	GAIN_BIT	2	gain_set_2	G_{SET2}
		3	gain_set_1	G_{SET1}
		3	-	Not used

Table 4-42 Register PGAxCTL0 Control Union

Structure Name	uni_reg_pgaxctl0_t			
Outline	Definition of union for register PGAxCTL0 control.			
Member Variable	Type	No. of bits	Name	Description
	uint8_t	8	BYTE	8-bit access
	PGAxCTL0_BIT	5	gain	GAIN
		3	osr	OSR

Table 4-43 Register PGAxCTL1 Control Union

Structure Name	uni_reg_pgaxctl1_t			
Outline	Definition of union for register PGAxCTL1 control.			
Member Variable	Type	No. of bits	Name	Description
	uint8_t	8	BYTE	8bit access
	PGAxCTL1_BIT	5	offset	offset
		1	-	Not used
		1	pga3tsel	Configurable amplifier self-correction
		1	input_mode	AINSEL

Table 4-44 Register PGAxCTL2 Control Union

Structure Name	uni_reg_pgaxctl2_t			
Outline	Definition of union for register PGAxCTL2 control.			
Member Variable	Type	No. of bits	Name	Description
	uint8_t	8	BYTE	8-bit access
	PGAxCTL2_BIT	5	low	Offset
		3	high	Not used

Table 4-45 Register PGAxCTL3 Control Union

Structure Name	uni_reg_pgaxctl3_t			
Outline	Definition of union for register PGAxCTL3 control.			
Member Variable	Type	No. of bits	Name	Description
	uint8_t	8	BYTE	8-bit access
	PGAxCTL3_BIT	2	pgaxave0_1	PGAXAV[1:0] Selection of data for averaging
		2	pgaxave3_2	PGAXAV[3:2] Selection of averaging processing operation
		3	-	Not used
		1	pgaxctm	Selection of A/D conversion count specification mode

Table 4-46 Register DSADCTL Control Union

Structure Name	uni_reg_pgaxctl3_t			
Outline	Definition of union for register DSADCTL control.			
Member	Type	No. of bits	Name	Description

Variable	uint8_t	8	BYTE	8-bit access
	DSADCTL_BIT	5	dsadbmp4_0	Signal from input multiplexer n (n=4-0)
		1	dsadscm	Auto-scan mode selection
		1	-	Not used
		1	dsadtst	A/D conversion (AUTOSCAN) selection

4.5 Global Constants for User Environment-dependent Settings

4.5.1 Global variables storing key setting information

Users can adjust global constants to meet the specifications of their hardware configuration (I/O ports) by modifying the constants in the r_keyscan.h file.

(a) r_keyscan.h

Table 4-47 Structure for Global Variables Storing Key Setting Information

Structure Name	key_setting_t		
Outline	Structure for global variables storing key setting information		
Member Variable	Type	Name	Description
	uint8_t *	p_port_addr	Address of port register connected to SW pin
	uint8_t	port_bit_num	Bit number of port register connected to SW pin
	uint8_t	multiple_group	Group number for determining invalid simultaneous push

(b) Description example

- Case 1

  Register RL78/I1E's P13.7 as SW_MODE_CHANGE key (default in this sample code).

```
typedef enum
{

    SW_MODE_CHANGE = 0x00U,

    SW_TYPE_MAX
} e_key_t;

const key_setting_t g_con_key_setting[SW_TYPE_MAX] =
{
/*                      port bit number group */
/* SW_MODE_CHANGE   */   {&P13,  7U,       0U},
};
```

- Case 2

  Register RL78/I1E’s P12.1 as SW_GAIN_UP key and P12.0 as SW_GAIN_DOWN key, and set them both as Group 1 (simultaneous press invalid). Register P1.2 as SW_OFFSET_UP key and P1.5 as SW_OFFSET_DOWN key, and set them both as Group 2 (simultaneous press invalid).

```
typedef enum
{

    SW_GAIN_UP = 0x00U,
    SW_GAIN_DOWN,
    SW_OFFSET_UP,
    SW_OFFSET_DOWN,

    SW_TYPE_MAX
} e_key_t;

const key_setting_t g_con_key_setting[SW_TYPE_MAX] =
{
/*                   port bit number group */
/* SW_GAIN_UP      */  {&P12,  1U,       1U},
/* SW_GAIN_DOWN    */  {&P12,  0U,       1U},
/* SW_OFFSET_UP    */  {&P1,   2U,       2U},
/* SW_OFFSET_DOWN  */  {&P1,   5U,       2U},
};
```

4.6 Global Variables

This section describes the global variables defined in the API.

(a) r_cg_main.c

Table 4-48 r_cg_main.c Global Variables

Type	Value	Description
bulk_data_t	g_bulk_data	BULK transfer data storage variable

(b) r_rl78_i1e_common.c

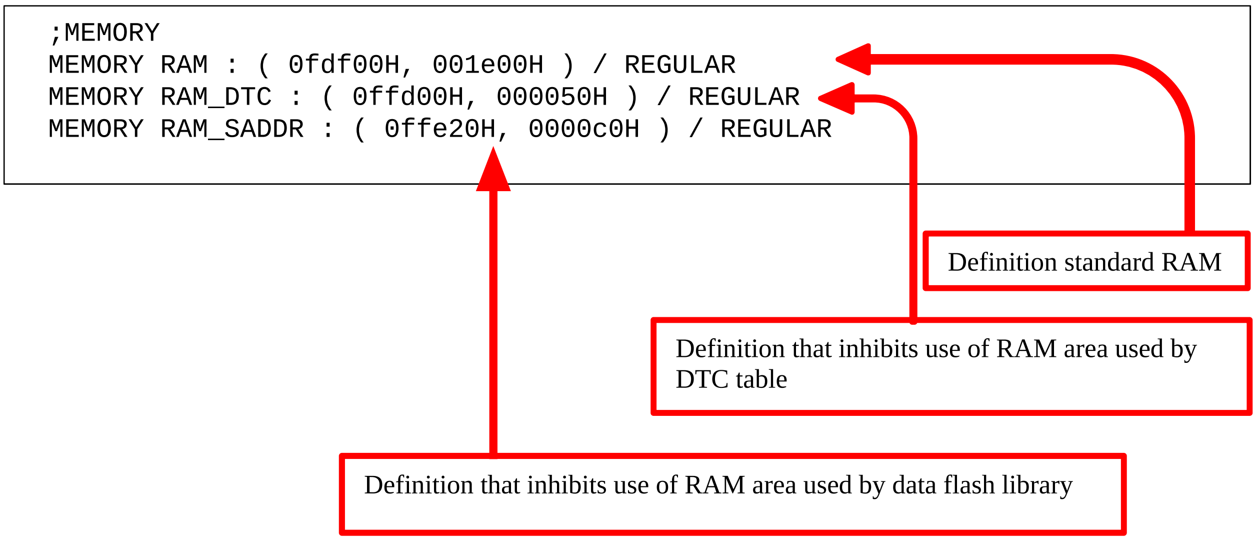
Table 4-49 r_rl78_i1e_common.c Global Variables

Variable Name	Type name	Description
g_flash_value	str_flash_data_t	Structure for data flash storage data
g_dsad_autoscan_mode	e_pga_dsad_autoscan_mode_t	Auto-scan mode setting
g_dsad_setting	str_pga_dsad_setting_t	Input multiplexer settings retentive variable
g_camp_trimming	uint8_t	Configurable amplifier trimming setting value storage variable
g_dsad_value	str_pga_dsad_value_t	DSAD converted value storage buffer
g_dsad_value_write_pos	uint16_t	DSAD converted value storage buffer write position
g_dsad_value_read_pos	uint16_t	DSAD converted value storage buffer read position
g_dsad_value_user_read_pos	uint16_t	DSAD converted value storage buffer user read position
g_dsad_measurement_count	uint32_t	DSAD conversion measurement counter
g_i1e_uart_send_buffer	int8_t	Transmit buffer for PC transmissions (DTC is used for transfers to UART)
g_buffer_number	uint8_t	Storage destination number of transmit buffer for PC transmissions

4.7 Link Directive File

This section explains how to set the link directive file to inhibit use of the RAM area used by the data flash library and DTC table.

The following provides a basic outline of the link directive file (SampleCode.dr) used in the sample code.



```
;MEMORY
MEMORY RAM : ( 0fdf00H, 001e00H ) / REGULAR
MEMORY RAM_DTC : ( 0ffd00H, 000050H ) / REGULAR
MEMORY RAM_SADDR : ( 0ffe20H, 0000c0H ) / REGULAR
```

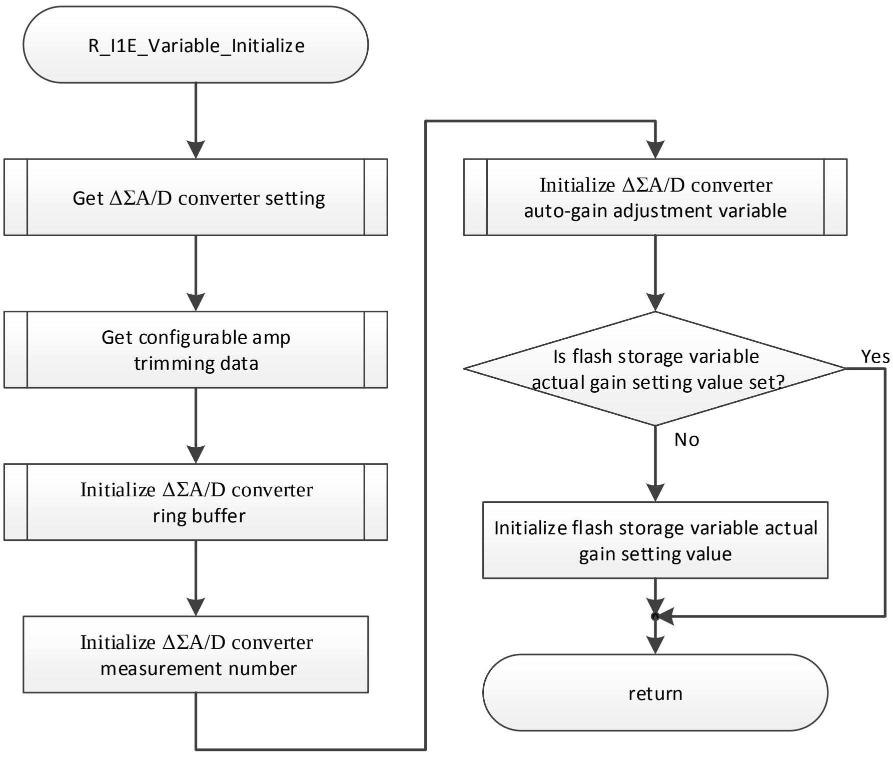
Definition standard RAM
Definition that inhibits use of RAM area used by DTC table
Definition that inhibits use of RAM area used by data flash library

5. Analog Front-end APIs

5.1 Sample code variable initialization processing

void R_I1E_Variable_Initialize(void)

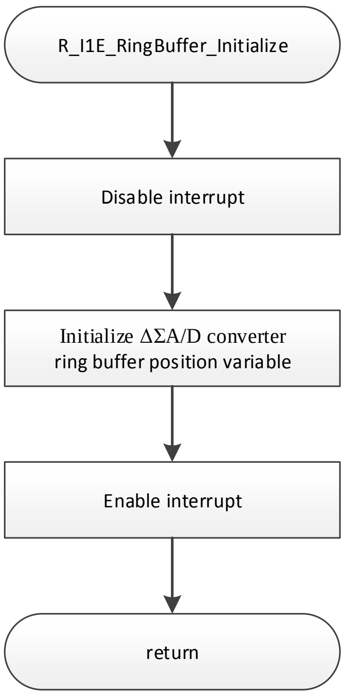
Description	Sample code variable initialization processing
Argument	None
Global Variables	g_dsad_autoscan_mode: Auto-scan mode setting g_dsad_setting[]: Input multiplexer settings retentive variable g_camp_trimming: Configurable amp n trimming setting value g_flash_value: Data flash storage data structure gs_con_gain_set_1_size: Number of preamplifier gain multiplier elements gs_con_gain_set_1[]: Post amplifier gain multiplier
SFR	None
Return Value	None
Processing	Get each setting for ΔΣA/D converter. Get configurable amp trimming data. Initialize ΔΣA/D converted value ring buffer. Initialize ΔΣA/D converter measurement counter. (0 = cleared) Initialize variable for PGA auto gain adjustment. If actual gain setting value array for flash storage structure is not set, initialize it.


R_I1E_Variable_Initialize
Get ΔΣA/D converter setting
Get configurable amp trimming data
Initialize ΔΣA/D converter ring buffer
Initialize ΔΣA/D converter measurement number
Initialize ΔΣA/D converter auto-gain adjustment variable
Is flash storage variable actual gain setting value set?
Yes
No
Initialize flash storage variable actual gain setting value
return

5.2 DSAD ring buffer initialization function

void R_I1E_RingBuffer_Initialize(void)

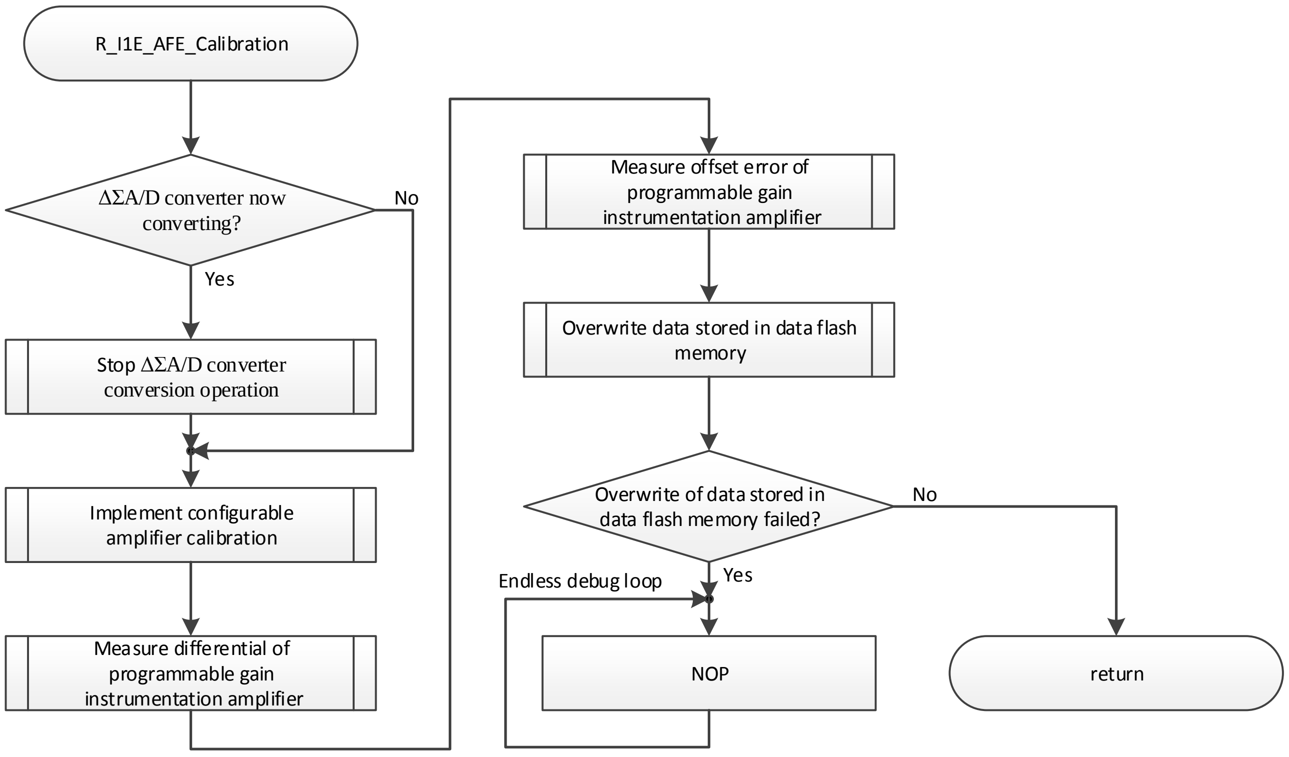
Description	DSAD ring buffer initialization function
Argument	None
Global Variables	g_dsad_value_write_pos: DSAD converted value storage buffer write position g_dsad_value_read_pos: DSAD converted value storage buffer read position g_dsad_value_user_read_pos: DSAD converted value storage buffer user read position
SFR	None
Return Value	None
Processing	Initialize memory variable for write/read positions of ring buffer that stores ΔΣA/D converted values.


R_I1E_RingBuffer_Initialize
Disable interrupt
Initialize ΔΣA/D converter ring buffer position variable
Enable interrupt
return

5.3 AFE Module Calibration Function

void R_I1E_AFE_Calibration(void)

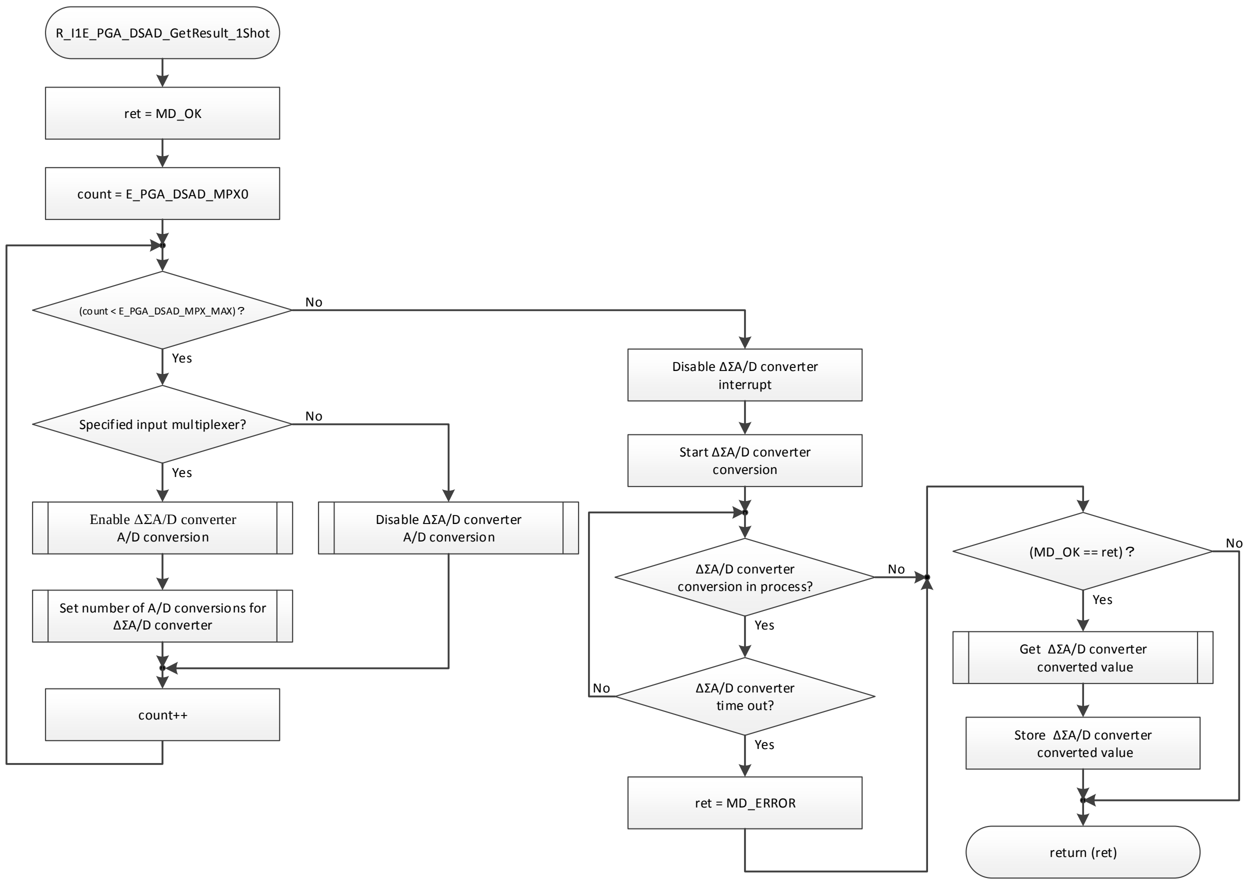
Description	AFE module calibration function
Argument	None
Global Variables	g_camp_trimming: Configurable amplifier n trimming setting value g_flash_value: Data flash storage data structure
SFR	None
Return Value	None
Processing	Execute configurable amp offset calibration, store results in g_camp_trimming. Measure gain error of programmable gain instrumentation amplifier, store measured value in actual gain setting value array of flash storage structure. Measure offset error of programmable gain instrumentation amplifier, store measured value in offset error correction value array of flash storage structure. Write obtained data in flash memory.


R_I1E_AFE_Calibration
ΔΣA/D converter now converting?
No
Yes
Stop ΔΣA/D converter conversion operation
Implement configurable amplifier calibration
Measure differential of programmable gain instrumentation amplifier
Measure offset error of programmable gain instrumentation amplifier
Overwrite data stored in data flash memory
Overwrite of data stored in data flash memory failed?
No
Yes
Endless debug loop
NOP
return

5.4 24-bit ΔΣA/D Converter/Specified Ch1-shot Differential Input A/D Converstion Function

MD_STATUS R_I1E_PGA_DSAD_GetResult_1Shot
(e_pga_dsad_mpx_t dsad_mpx, int32_t *p_dsad_value)

Description	24-bit ΔΣA/D converter/specified Ch 1-shot differential input A/D conversion function
Argument	dsad_mpx: 24-bit ΔΣA/D converter input multiplexer number *p_dsad_value: 24-bit ΔΣA/D converter 24-bit A/D value storage pointer
Global Variables	None
SFR	DSADIF: ΔΣA/D converter conversion complete interrupt request flag DSADSIF: ΔΣA/D converter scan complete interrupt request flag DSADMK: ΔΣA/D converter conversion complete interrupt mask flag DSADSMK: ΔΣA/D converter scan complete interrupt mask flag DSADST: A/D converter (AUTOSCAN) control
Return Value	MD_STATUS: MD_OK MD_ERROR
Processing	Execute A/D conversion withΔΣA/D converter based on input from channel specified by input multiplexer, store results in *p_dsad_value. Return MD_ERROR when A/D conversion goes to time out.


R_I1E_PGA_DSAD_GetResult_1Shot
ret = MD_OK
count = E_PGA_DSAD_MPX0
(count < E_PGA_DSAD_MPX_MAX)？
No
Yes
Specified input multiplexer?
No
Yes
Enable ΔΣA/D converter A/D conversion
Disable ΔΣA/D converter A/D conversion
Set number of A/D conversions for ΔΣA/D converter
count++
Disable ΔΣA/D converter interrupt
Start ΔΣA/D converter conversion
ΔΣA/D converter conversion in process?
No
Yes
ΔΣA/D converter time out?
No
Yes
ret = MD_ERROR
(MD_OK == ret)？
No
Yes
Get ΔΣA/D converter converted value
Store ΔΣA/D converter converted value
return (ret)

5.5 24-bit ΔΣA/D Converter Gain Setting Function

void R_I1E_PGA_DSAD_GainRegSet(e_pga_dsad_mpx_t dsad_mpx, e_pga_dsad_gain_t dsad_gain)

Description	24-bit ΔΣA/D converter gain setting function
Argument	dsad_mpx: 24-bit ΔΣA/D converter input multiplexer number dsad_gain: 24-bit ΔΣA/D converter/gain setting value
Global Variables	None
SFR	PGAxCTL0: Input multiplexer x (x = 0 to 4) setting register 0
Return Value	None
Processing	Set the specified value in the programmable gain instrumentation amp gain register for the channel specified by theΔΣA/D converter input multiplexer

5.6 24-bit ΔΣA/D Converter Offset Setting Function

void R_I1E_PGA_DSAD_OffsetRegSet
(e_pga_dsad_mpx_t dsad_mpx, e_pga_dsad_offset_t dsad_offset)

Description	24-bit ΔΣA/D converter offset setting function
Argument	dsad_mpx: 24-bit ΔΣA/D converter input multiplexer number dsad_offset: 24-bit ΔΣA/D converter offset setting value
Global Variables	None
SFR	PGAxCTL1: Input multiplexer x (x = 0 to 4) setting register 1
Return Value	None
Processing	Set the specified value in the offset voltage register for the channel specified by the ΔΣA/D converter input multiplexer.

5.7 24-bit ΔΣA/D Converter OSR Setting Function

void R_I1E_PGA_DSAD_OsrRegSet(e_pga_dsad_mpx_t dsad_mpx, e_pga_dsad_osr_t dsad_osr)

Description	24-bit ΔΣA/D converter OSR setting function
Argument	dsad_mpx: 24-bit ΔΣA/D converter input multiplexer number osr: 24-bit ΔΣA/D converter/OSR setting function
Global Variables	None
SFR	PGAxCTL0: input multiplexer x (x = 0 to 4) setting register 0
Return Value	None
Processing	Set the specified value in the oversampling rate register for the channel specified by the ΔΣA/D converter input multiplexer.

5.8 Auto-scan Mode Input Multiplexer Settings Retentive Variable Storage Function

void R_I1E_PGA_DSAD_SettingRegGet
(e_pga_dsad_autoscan_mode_t *dsad_autoscan, str_pga_dsad_setting_t dsad_setting[])

Description	Auto-scan mode input multiplexer settings retentive variable storage function
Argument	*dsad_autoscan: Auto-scan mode setting dsad_setting[]: 24-bit ΔΣA/D converter setting value
Global Variables	gs_gain_set_2: Post amplifier stage gain multiplier
SFR	None
Return Value	None
Processing	Get ΔΣA/D converter auto-scan mode settings and store in *dsad_autoscan. Get ΔΣA/D converter all input multiplexer settings and store in dsad_setting[] array. Get post amplifier gain multiplier of programmable gain instrumentation amplifier for input multiplexers used for PGA error measurement, set in global variable gs_gain_set_2.

5.9 Set Structure Variable Information to DSAD Setting

void R_I1E_PGA_DSAD_SettingRegSet
(e_pga_dsad_autoscan_mode_t dsad_autoscan, str_pga_dsad_setting_t dsad_setting[])

Description	Set structure variable information to DSAD setting
Argument	dsad_autoscan: Auto-scan mode setting dsad_setting[]: 24-bit ΔΣA/D converter setting value
Global Variables	None
SFR	None
Return Value	None
Processing	Set ΔΣA/D converter auto-scan mode to specified mode. Set all settings for input multiplexers of the ΔΣA/D converter to the specified values in the dsad_setting[] array.

5.10 Get PGA offset error correction value function

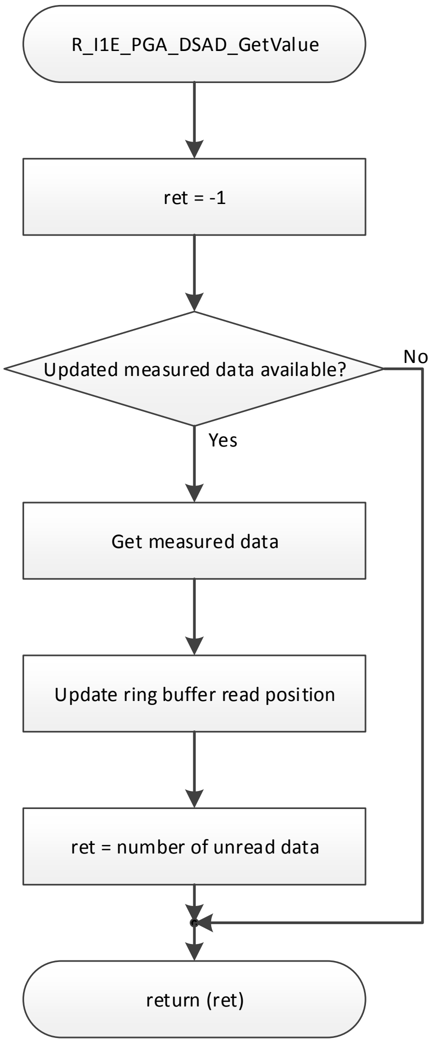
int32_t R_I1E_PGA_DSAD_CorrectValue(str_pga_dsad_value_t *p_value)

Description	Get PGA offset error correction value function
Argument	*p_value: 24-bit ΔΣA/D converter converted value pointer
Global Variables	g_flash_value: Data flash storage data structure
SFR	None
Return Value	int32_t: PGA offset error correction value. Store same value in p_value >adc_correct
Processing	Calculate the offset error correction value based on the data stored in the specified A/D conversion information storage structure and return the resulting value. At the same time, also store the value in member adc_correct of the specified structure.

5.11 24-bit ΔΣA/D Converter Converted Value Get Function

int16_t R_I1E_PGA_DSAD_GetValue(str_pga_dsad_value_t *p_value)

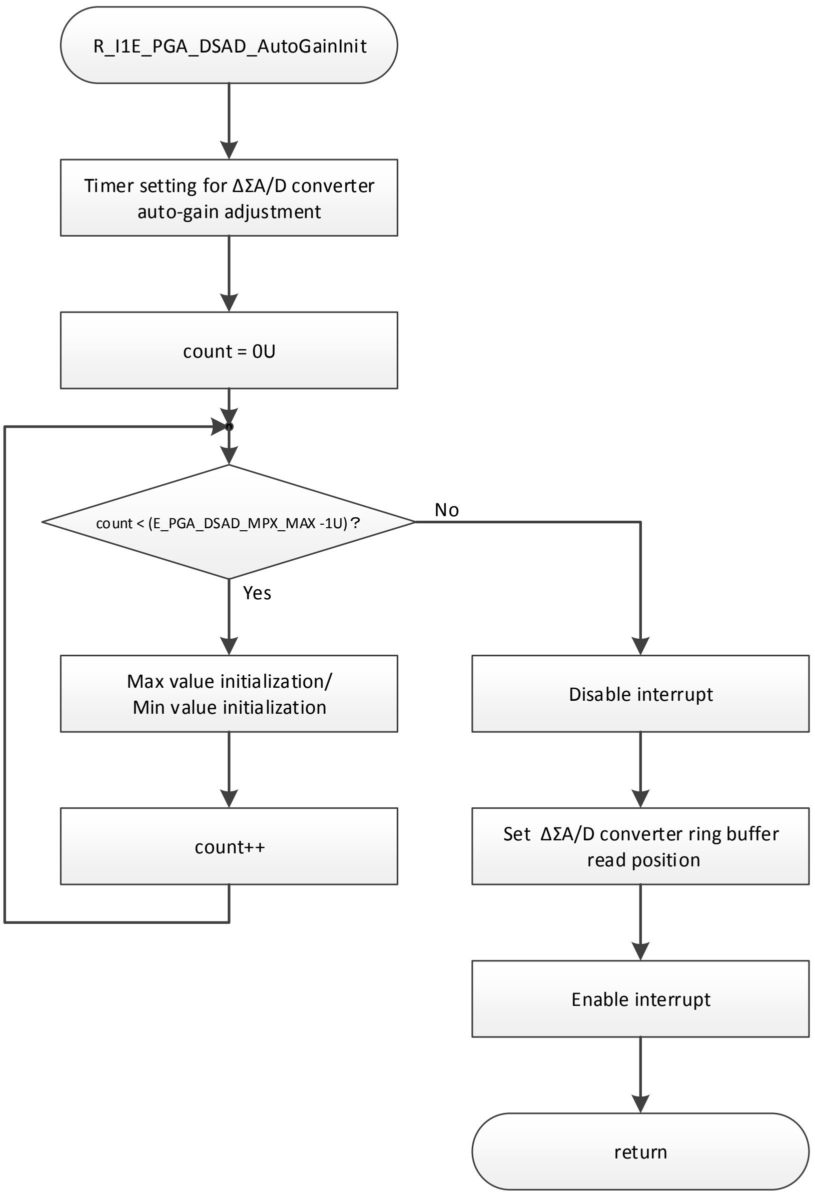
Description	24-bit ΔΣA/D converter converted value get function
Argument	*p_value: 24-bit ΔΣA/D converter converted value pointer
Global Variables	g_dsad_value_write_pos: DSAD converted value storage buffer write position g_dsad_value_user_read_pos: DSAD converted value storage buffer user read position g_dsad_value[]: DSAD converted value storage buffer
SFR	None
Return Value	int16_t: 0 or higher: number of data read completed + unread data 1: buffer empty
Processing	Copy data from ΔΣA/D converter converted value ring buffer to *p_value, update ring buffer read position. If the data read operation is successfully completed, return the number of unread data remaining. (0 or higher) If no unread data remains, do nothing and return -1.


R_I1E_PGA_DSAD_GetValue
ret = -1
Updated measured data available?
No
Yes
Get measured data
Update ring buffer read position
ret = number of unread data
return (ret)

5.12 PGA auto-gain variable initialization

void R_I1E_PGA_DSAD_AutoGainInit(void)

Description	PGA auto-gain variable initialization
Argument	None
Global Variables	gs_dsad_auto_gain_buffer[]: PGA auto-gain adjustment buffer g_dsad_value_write_pos: DSAD converted value storage buffer write position g_dsad_value_read_pos: DSAD converted value storage buffer read position
SFR	None
Return Value	None
Processing	Initialize PGA auto-gain adjustment variable.

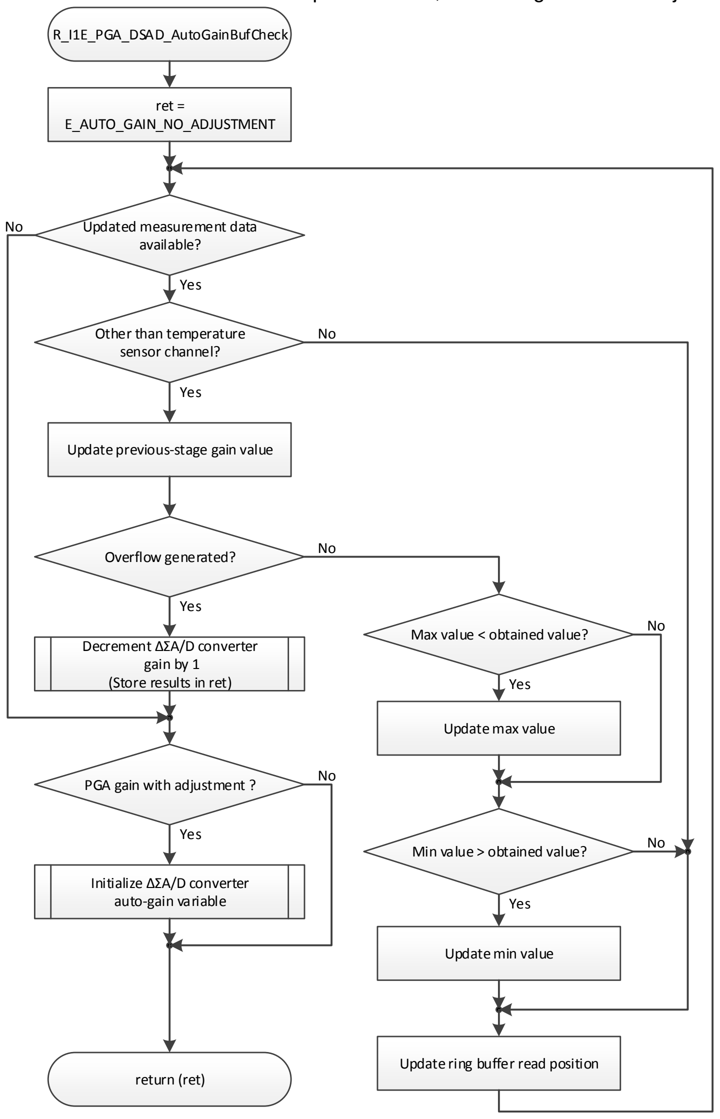


R_I1E_PGA_DSAD_AutoGainInit
Timer setting for ΔΣA/D converter auto-gain adjustment
count = 0U
count < (E_PGA_DSAD_MPX_MAX -1U)?
No
Yes
Max value initialization/ Min value initialization
count++
Disable interrupt
Set ΔΣA/D converter ring buffer read position
Enable interrupt
return

5.13 24-bit ΔΣA/D Converter Conversion Complete Auto-Gain Adjustment Detection Processing Function

e_auto_gain_return_t R_I1E_PGA_DSAD_AutoGainBufCheck(void)

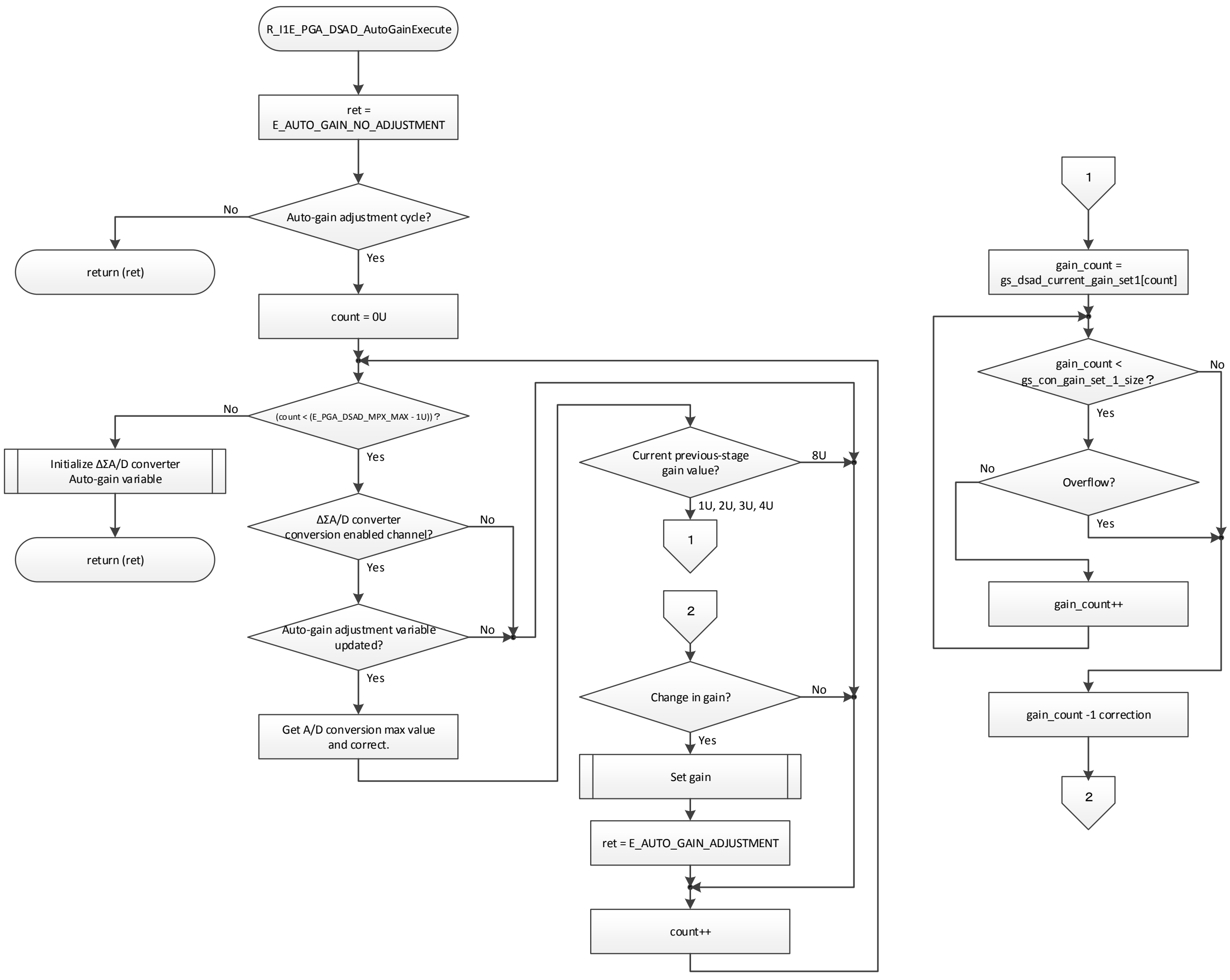
Description	4-bit ΔΣA/D converter conversion complete auto-gain adjustment detection processing function
Argument	None
Global Variables	g_dsad_value_write_pos: DSAD converted value storage buffer write position g_dsad_value_read_pos: DSAD converted value storage buffer read position g_dsad_value[]: DSAD converted value storage buffer gs_dsad_current_gain_set1[]: Current preamplifier gain value gs_dsad_auto_gain_buffer[]: PGA auto-gain adjustment buffer
SFR	None
Return Value	e_auto_gain_return_t: E_AUTO_GAIN_NO_ADJUSTMENT PGA gain, no adjustment E_AUTO_GAIN_ADJUSTMENT PGA gain, with adjustment
Processing	Update maximum and minimum values from ΔΣA/D converter measurement results. If the gain is judge unsuitable based on the updated value, the PGA gain is auto-adjusted.


R_I1E_PGA_DSAD_AutoGainBufCheck
ret = E_AUTO_GAIN_NO_ADJUSTMENT
Updated measurement data available?
No
Yes
Other than temperature sensor channel?
No
Yes
Update previous-stage gain value
Overflow generated?
No
Yes
Decrement ΔΣA/D converter gain by 1 (Store results in ret)
Max value < obtained value?
No
Yes
Update max value
PGA gain with adjustment ?
No
Yes
Initialize ΔΣA/D converter auto-gain variable
Min value > obtained value?
No
Yes
Update min value
Update ring buffer read position
return (ret)

5.14 24-bit ΔΣA/D Converter Auto-Gain Adjustment Function

e_auto_gain_return_t R_I1E_PGA_DSAD_AutoGainExecute(void)

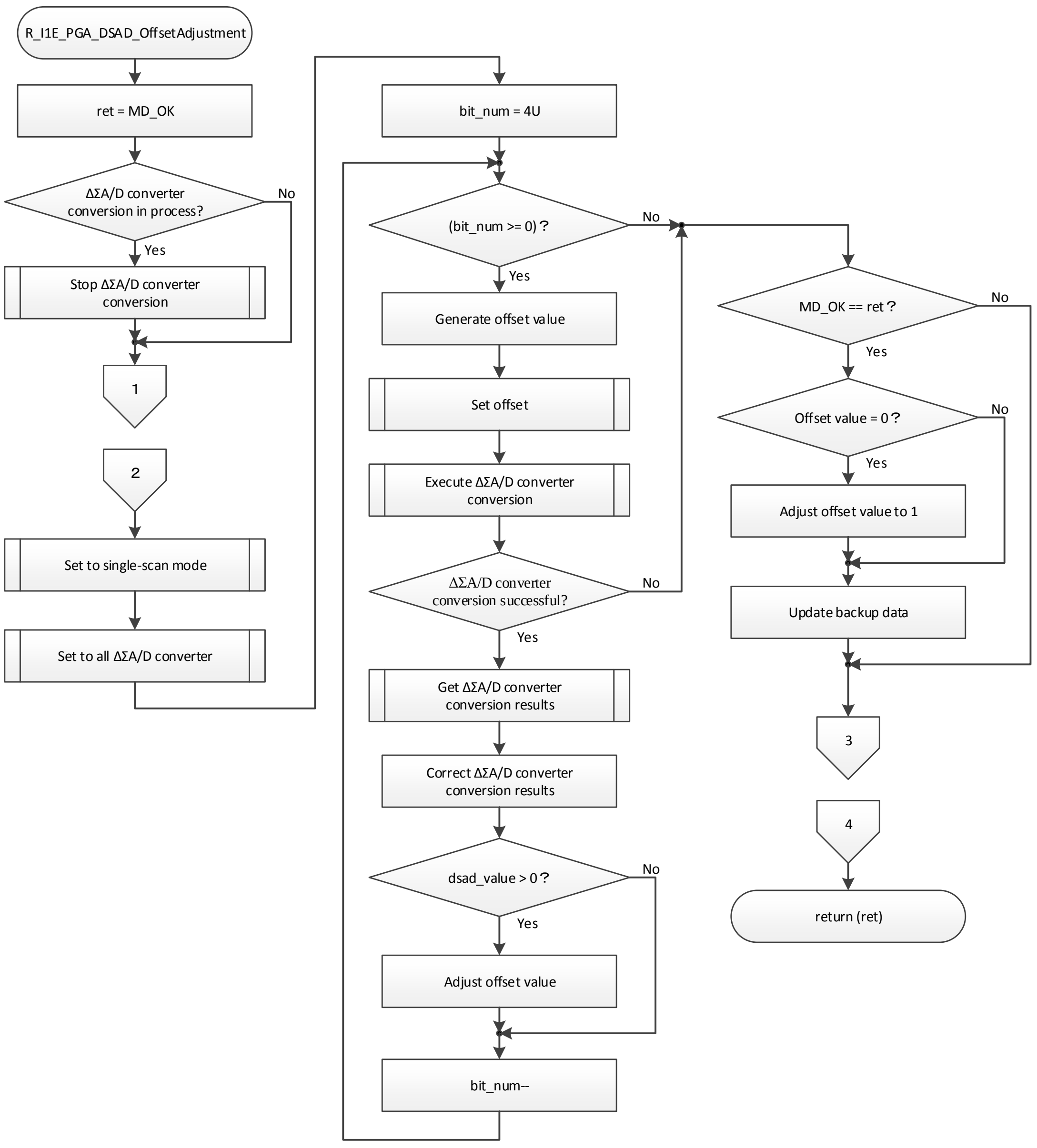
Description	24-bit ΔΣA/D converter auto-gain adjustment function
Argument	None
Global Variables	gs_dsad_auto_gain_timer: PGA auto-gain adjustment timer g_dsad_setting[]: 24-bit ΔΣA/D converter setting value gs_dsad_auto_gain_buffer[]: PGA auto-gain adjustment buffer gs_dsad_current_gain_set1[]: Current p gain value gs_con_gain_set_1_size: Preamplifier gain multiplier
SFR	None
Return Value	e_auto_gain_return_t: E_AUTO_GAIN_NO_ADJUSTMENT PGA gain, no adjustment E_AUTO_GAIN_ADJUSTMENT PGA gain, with adjustment
Processing	Execute PGA auto-gain adjustment for input multiplexer specified by ΔΣA/D converter.


R_I1E_PGA_DSAD_AutoGainExecute
ret = E_AUTO_GAIN_NO_ADJUSTMENT
Auto-gain adjustment cycle?
No
Yes
return (ret)
count = 0U
(count < (E_PGA_DSAD_MPX_MAX - 1U)) ?
No
Yes
Initialize ΔΣA/D converter Auto-gain variable
return (ret)
ΔΣA/D converter conversion enabled channel?
No
Yes
Auto-gain adjustment variable updated?
No
Yes
Get A/D conversion max value and correct.
Current previous-stage gain value?
8U
1U, 2U, 3U, 4U
1
2
Change in gain?
No
Yes
Set gain
ret = E_AUTO_GAIN_ADJUSTMENT
count++
1
gain_count = gs_dsad_current_gain_set1[count]
gain_count < gs_con_gain_set_1_size ?
No
Yes
No
Overflow?
Yes
gain_count++
gain_count -1 correction
2

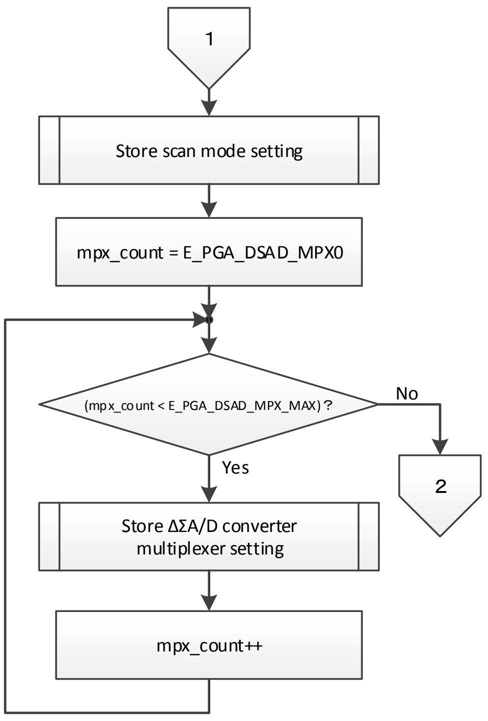
5.15 PGA Offset Adjustment Execution Function

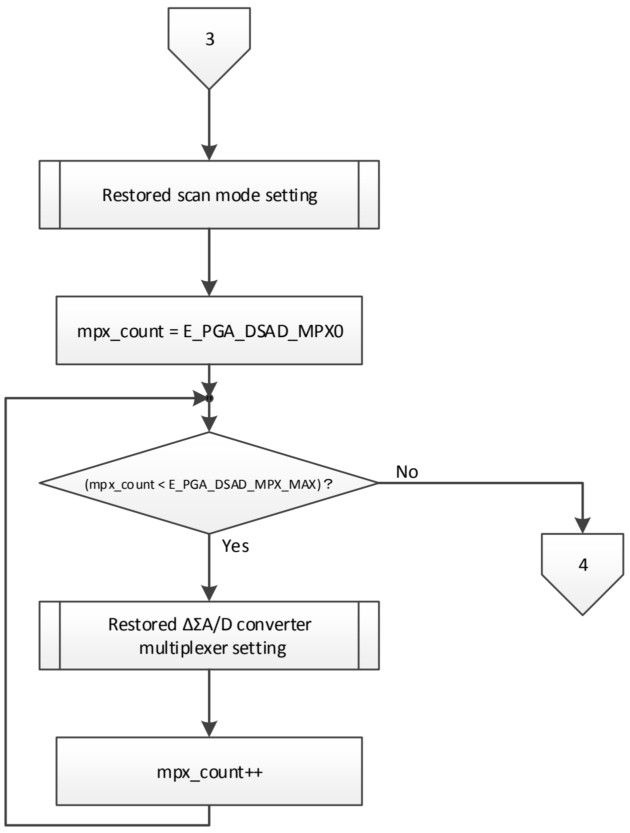
MD_STATUS R_I1E_PGA_DSAD_OffsetAdjustment(e_pga_dsad_mpx_t dsad_mpx)

Description	PGA offset adjustment execution function
Argument	dsad_mpx: 24-bit ΔΣA/D converter input multiplexer number
Global Variables	None
SFR	None
Return Value	MD_STATUS: MD_OK MD_ERROR
Processing	Execute PGA offset adjustment for input multiplexer specified by ΔΣA/D converter. Store the results in the offset error correction value array in the flash storage structure. If offset calibration fails, return MD_ERROR.



R_I1E_PGA_DSAD_OffsetAdjustment
ret = MD_OK
ΔΣA/D converter conversion in process?
No
Yes
Stop ΔΣA/D converter conversion
1
2
Set to single-scan mode
Set to all ΔΣA/D converter
bit_num = 4U
(bit_num >= 0)？
No
Yes
Generate offset value
Set offset
Execute ΔΣA/D converter conversion
ΔΣA/D converter conversion successful?
No
Yes
Get ΔΣA/D converter conversion results
Correct ΔΣA/D converter conversion results
dsad_value > 0？
No
Yes
Adjust offset value
bit_num--
MD_OK == ret？
No
Yes
Offset value = 0？
No
Yes
Adjust offset value to 1
Update backup data
3
4
return (ret)

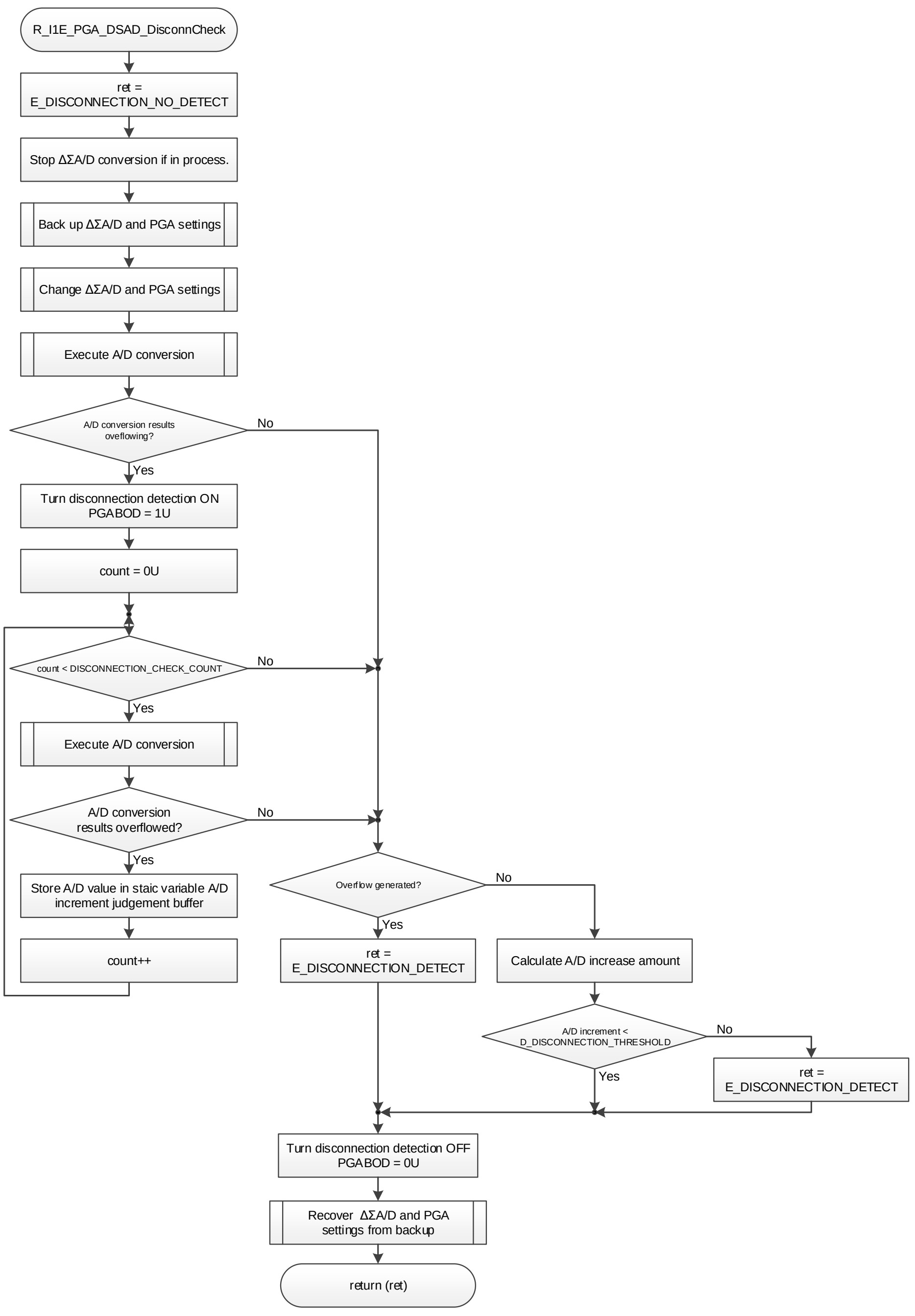

1
Store scan mode setting
mpx_count = E_PGA_DSAD_MPX0
(mpx_count < E_PGA_DSAD_MPX_MAX)？
No
Yes
2
Store ΔΣA/D converter multiplexer setting
mpx_count++


3
Restored scan mode setting
mpx_count = E_PGA_DSAD_MPX0
(mpx_count < E_PGA_DSAD_MPX_MAX)？
No
Yes
4
Restored ΔΣA/D converter multiplexer setting
mpx_count++

5.16 Disconnection Detection Function

e_disconnection_return_t R_I1E_PGA_DSAD_DisconnCheck(e_pga_dsad_mpx_t dsad_mpx)

Description	Disconnection detection function	
Argument	dsad_mpx:	
	24-bit ΔΣA/D converter input multiplexer number	
Global Variables	None	
SFR	None	
Return Value	e_disconnection_return_t:	
	E_DISCONNECTION_NO_DETECT	Disconnection not detected
	E_DISCONNECTION_DETECT	Disconnection detected
Processing	Checks for disconnection of input multiplexer specified by ΔΣA/D converter	
	Returns E_DISCONNECTION_DETECT when disconnection is detected.	


R_I1E_PGA_DSAD_DisconnCheck
ret = E_DISCONNECTION_NO_DETECT
Stop ΔΣA/D conversion if in process.
Back up ΔΣA/D and PGA settings
Change ΔΣA/D and PGA settings
Execute A/D conversion
A/D conversion results overflowing?
No
Yes
Turn disconnection detection ON PGABOD = 1U
count = 0U
count < DISCONNECTION_CHECK_COUNT
No
Yes
Execute A/D conversion
A/D conversion results overflowed?
No
Yes
Store A/D value in staic variable A/D increment judgement buffer
count++
Overflow generated?
No
Yes
ret = E_DISCONNECTION_DETECT
Calculate A/D increase amount
A/D increment < D_DISCONNECTION_THRESHOLD
No
Yes
ret = E_DISCONNECTION_DETECT
Turn disconnection detection OFF PGABOD = 0U
Recover ΔΣA/D and PGA settings from backup
return (ret)

5.17 RL78/I1E AFE Stop Processing Function

void R_I1E_AFE_Stop(void)

Description	RL78/I1E AFE stop processing function
Argument	None
Global Variables	None
SFR	PER0: Peripheral enable register 0 AFEPWS: Analog front-end power supply selection register AFEPON: Control of power supplied to AFE reference power supply (ABGR) block PER1: Peripheral enable register 1 AFEEN: Control of input clock supplied to AFE power supply/clock control block
Return Value	None
Processing	Put analog front-end in stop state.

5.18 RL78/I1E AFE Restart Processing Function

void R_I1E_AFE_ReStart(void)

Description	RL78/I1E AFE restart processing function
Argument	None
Global Variables	None
SFR	AFEEN: Control of input clock supplied to AFE power supply/clock control block AFEPON: Control of power supplied to AFE reference power supply (ABGR) block AFESTAT: Status of power supplied to AFE reference power supply (ABGR) block AFECKS: Analog front-end clock selection register PGAEN: Control of input clock supplied to PGA and 24-bit ΔΣ A/D converter DSADMR: ΔΣA/D converter mode register PGAPON: Control of power supplied to programmable gain instrumentation amplifier (PGA) block PGASTAT: Status of power supplied to AFE reference power supply (ABGR) block
Return Value	None
Processing	Enable use of analog front-end.

5.19 Configurable amplifier trimming function

void R_I1E_CAMP_Calibration(uint8_t *camp_trimming)

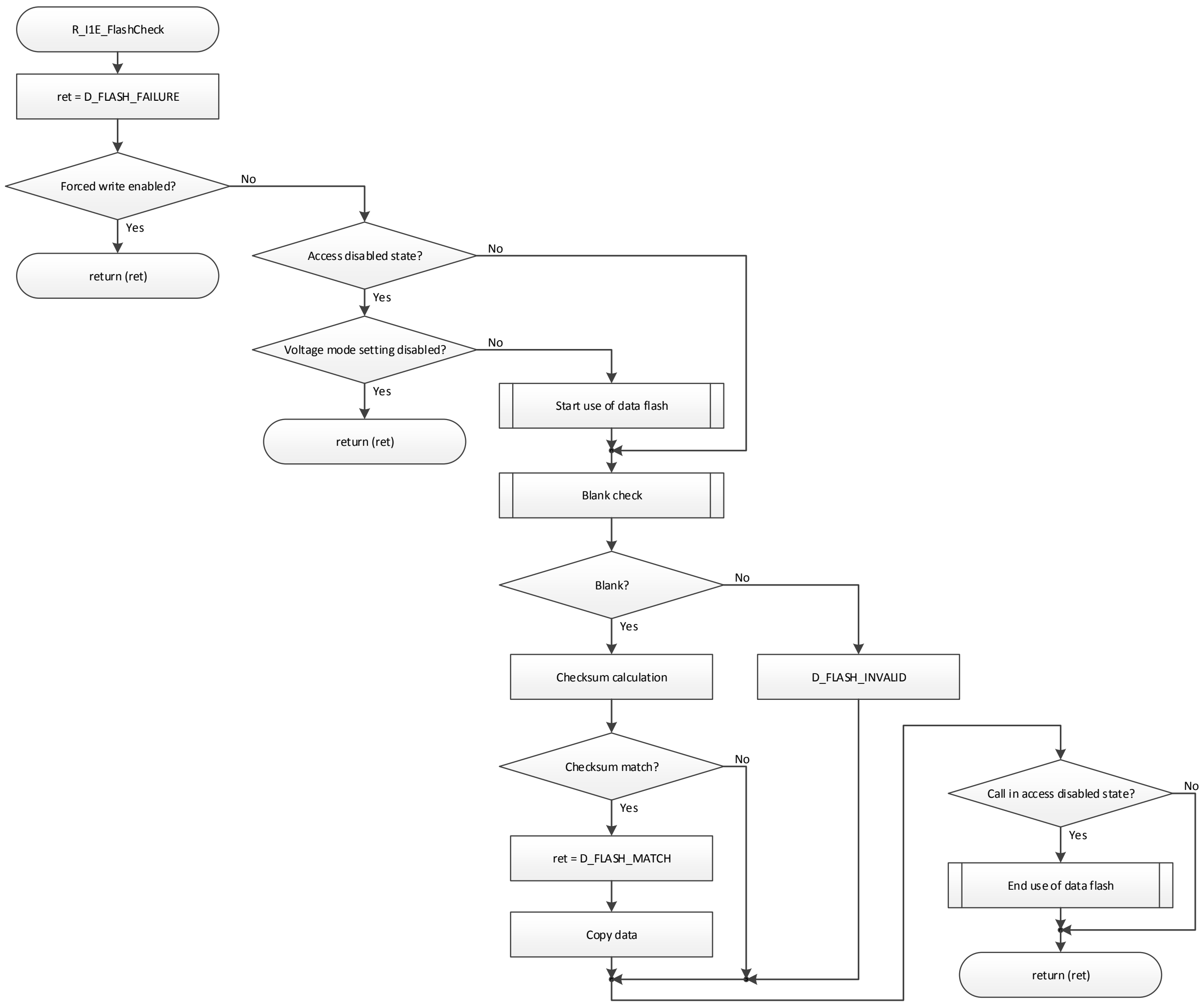
Description	Configurable amplifier trimming function
Argument	*camp_trimming: Configurable amplifier n trimming setting value
Global Variables	None
SFR	AMP0CAL: Configurable amplifier 0 trimming register AMP1CAL: Configurable amplifier 1 trimming register AMP2CAL: Configurable amplifier 2 trimming register
Return Value	None
Processing	Execute input offset trimming for configurable amplifier, then store results in *camp_trimming.

6. Flash Memory API

6.1 Data Flash Memory Storage Data Check Function

uint8_t R_I1E_FlashCheck(void)

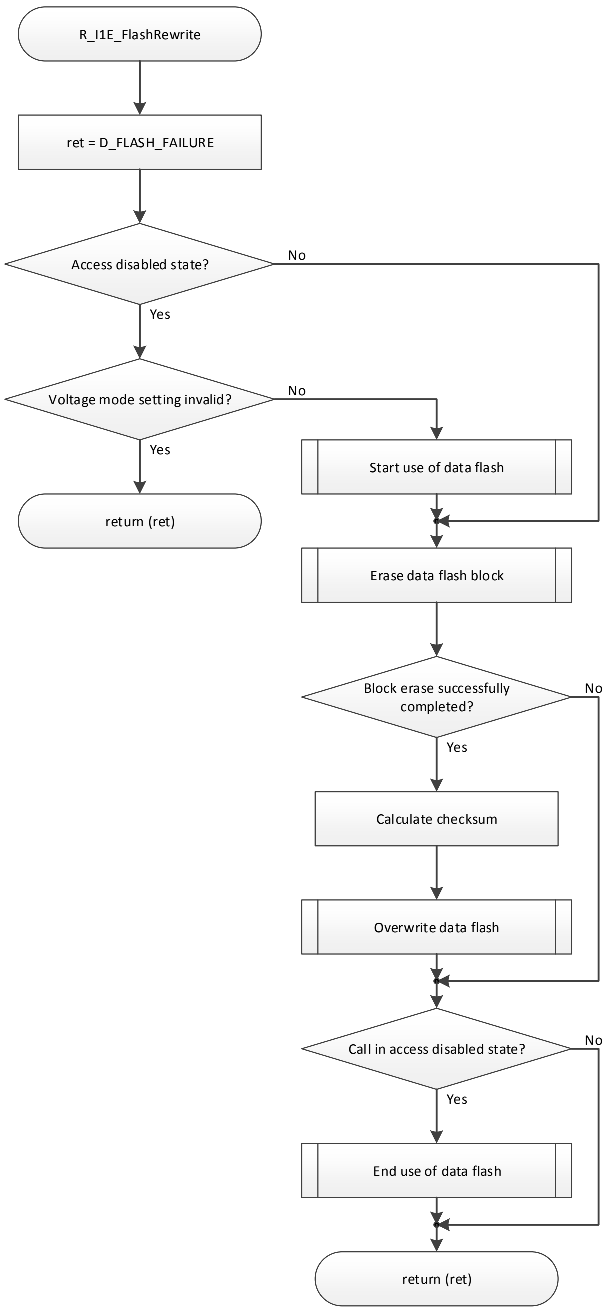
Description	Data flash memory storage data check function
Argument	None
Global Variables	g_flash_value: Data flash storage data structure
SFR	DFLCTL: Data flash control register
Return Value	uint8_t: D_FLASH_MATCH D_FLASH_FAILURE D_FLASH_INVALID
Processing	When the data flash dedicated block is not blank and there is no difference between the data written to the data flash and the checksum value, the function reads the data in the flash storage structure and returns D_FLASH_MATCH. When the data flash dedicated block is blank, data written to the data flash is not read and D_FLASH_INVALID is returned. When the data recorded in the data flash is abnormal, access to the data flash is blocked, or forced flash data write is enabled, the flash storage structure read operation is not executed and D_FLASH_FAILURE is returned.



6.2 Data Flash Memory Storage Data Overwrite Function

uint8_t R_I1E_FlashRewrite(str_flash_data_t *p_flash_value)

Description	Data flash memory storage data overwrite function
Argument	*p_flash_value: Flash memory write data storage buffer pointer
Global Variables	None
SFR	DFLCTL: Data flash control register
Return Value	uint8_t: D_FLASH_SUCCESS D_FLASH_FAILURE
Processing	Overwrite specified data and checksum to dedicated block of data flash. If write operation fails, return D_FLASH_FAILURE.

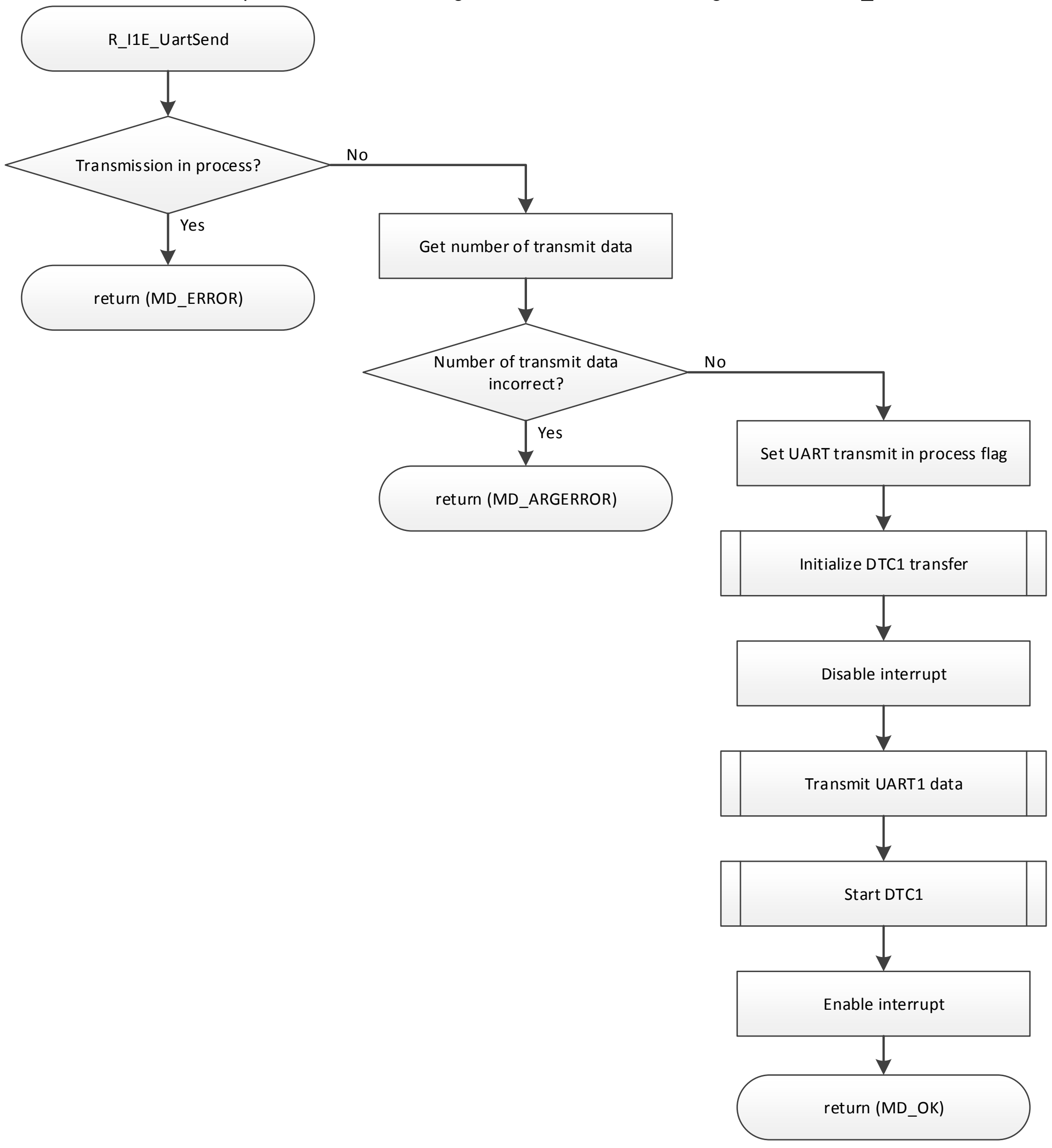


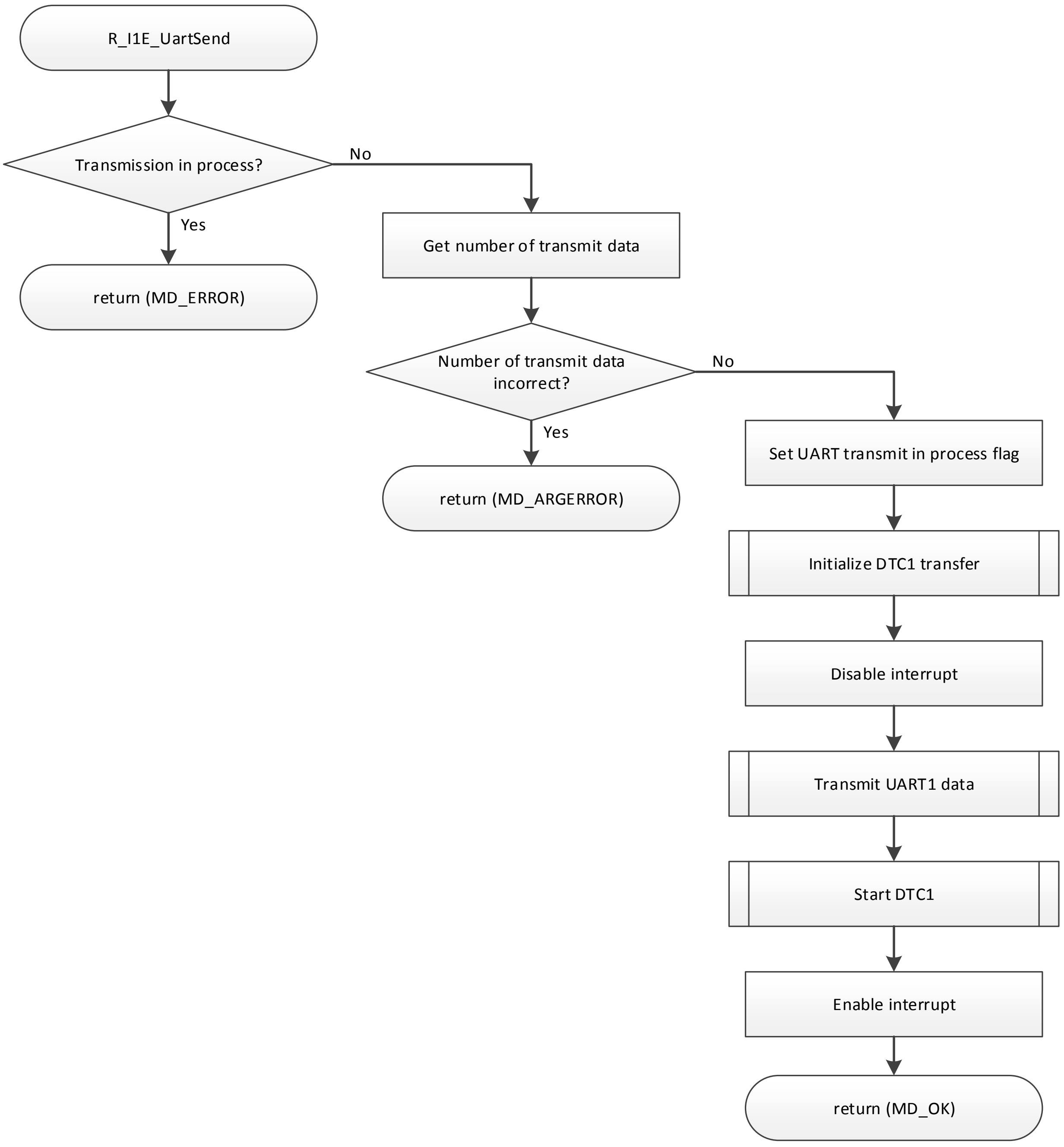
7. UART Communication API

7.1 PC Transmission Processing Setting Function (ASCII): limit of up to 256 bytes per transmission

MD_STATUS R_I1E_UartSend(int8_t *p_send_data)

Description	PC transmission processing setting function (ASCII): limit of up to 256 bytes per transmission
Argument	*p_send_data: Transmit buffer address
Global Variables	g_tx_in_process_flag: UART transmission in process flag
SFR	None
Return Value	MD_STATUS: MD_OK MD_ERROR MD_ARGERROR
Processing	Use Control Data 1 of data transfer controller to send specified character string from UART1. If UART transmission is already in process, do nothing and return MD_ERROR. If specified character string size is incorrect, do nothing and return MD_ARGERROR.

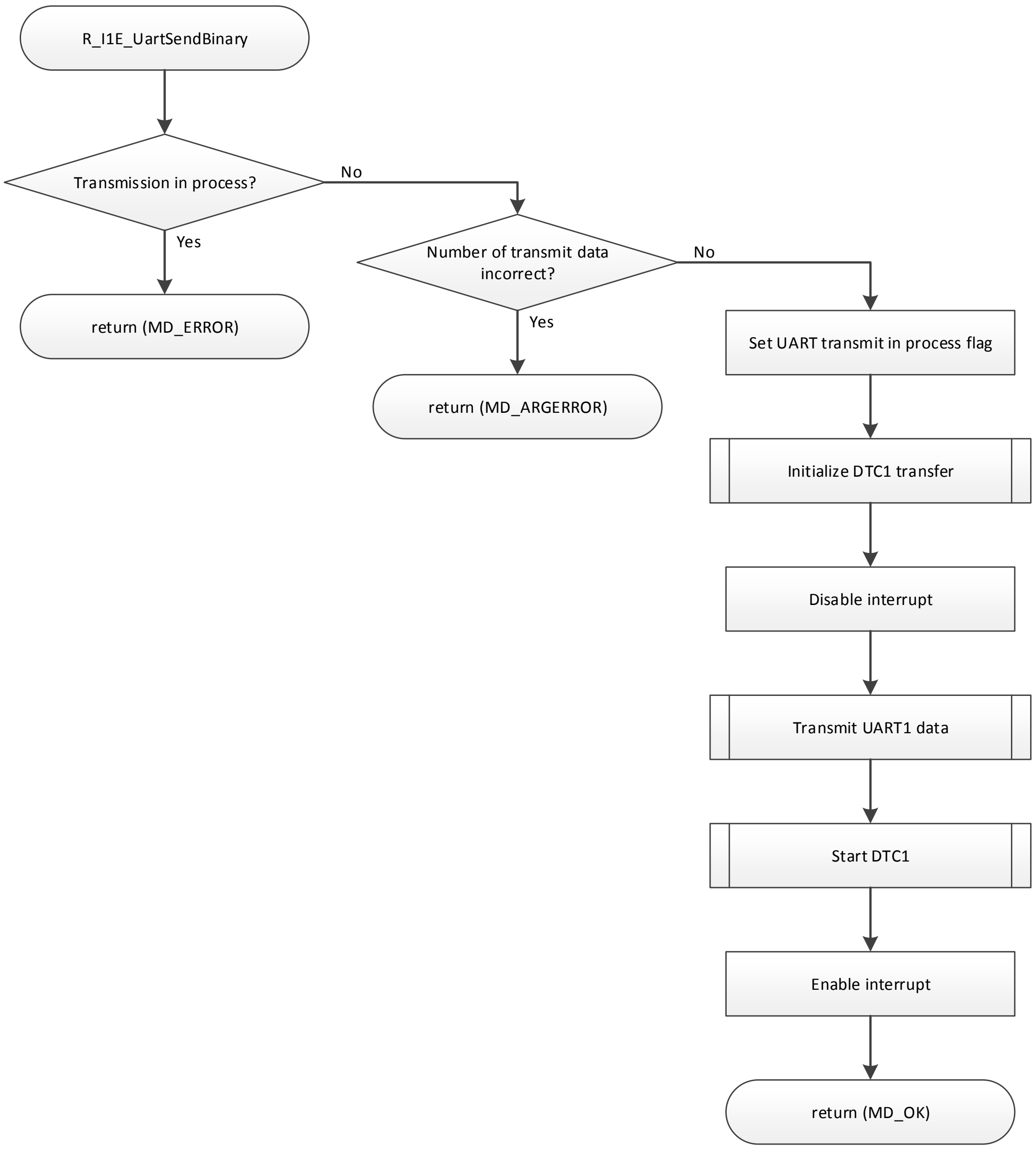



R_I1E_UartSend
Transmission in process?
No
Yes
return (MD_ERROR)
Get number of transmit data
Number of transmit data incorrect?
No
Yes
return (MD_ARGERROR)
Set UART transmit in process flag
Initialize DTC1 transfer
Disable interrupt
Transmit UART1 data
Start DTC1
Enable interrupt
return (MD_OK)

7.2 PC Transmission Processing Setting Function (Binary): limit of up to 256 bytes per transmission

MD_STATUS R_I1E_UartSendBinary(int8_t *p_send_data, uint16_t send_length)

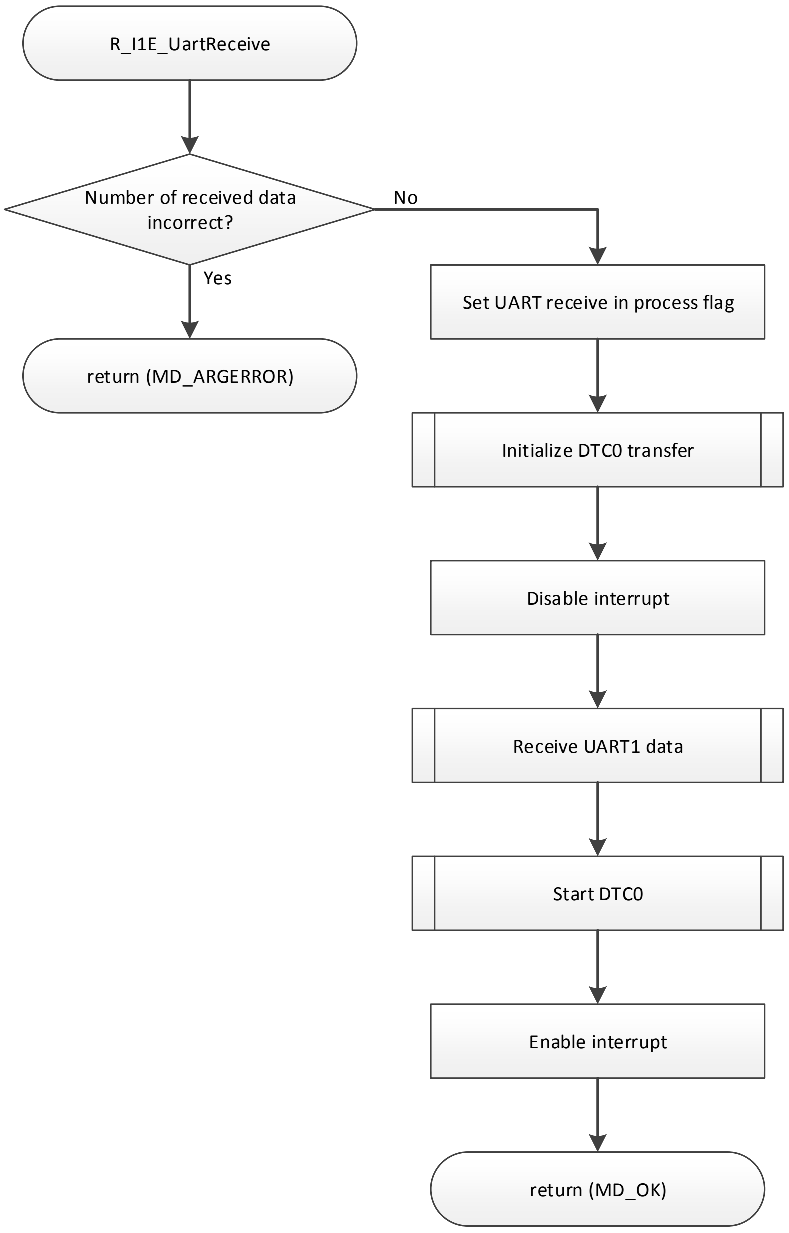
Description	PC transmission processing setting function (Binary): limit of up to 256 bytes per transmission
Argument	*p_send_data: Transmit buffer address send_length: Transmission size
Global Variables	g_tx_in_process_flag: UART transmission in process flag
SFR	None
Return Value	MD_STATUS: MD_OK MD_ERROR MD_ARGERROR
Processing	Use Control Data 1 of data transfer controller to send specified number of bytes from UART1. If transmission is already in process, do nothing and return MD_ERROR. If transmission size is incorrect, do nothing and return MD_ARGERROR.



7.3 Receive Processing from PC (using DTC0): limit of up to 256 bytes per reception

MD_STATUS R_I1E_UartReceive(int8_t *p_receive_data, uint8_t receive_length)

Description	Receive processing from PC (using DTC0).
Argument	*p_receive_data: Receive buffer address receive_length: Specified receive size
Global Variables	g_rx_in_process_flag: UART reception in process flag
SFR	None
Return Value	MD_STATUS: MD_OK MD_ARGERROR
Processing	Use Control Data 0 of data transfer controller to receive specified number of bytes from UART1. If specified receive size is incorrect, do nothing and return MD_ARGERROR.



8. Free Running Timer API

8.1 Free Running Timer Setting Initialization Function

void R_TAU_FreeRunTimerInit(void)

Description	Free running timer setting initialization function
Argument	None
Global Variables	None
SFR	TDR11: Tmer data register 11
Return Value	None
Processing	Initializes free running timer setting, starts timer operations.

8.2 Free Running Timer Stop Function

void R_TAU_FreeRunTimerStop(void)

Description	Free running timer stop function
Argument	None
Global Variables	None
SFR	None
Return Value	None
Processing	Stops free running timer.

8.3 Get Free Running Timer Tick Count Function

uint32_t R_GetTickCount(void)

Description	Get free running timer tick count function
Argument	None
Global Variables	g_freerun_timer: Free running timer counter variable
SFR	TCR11: Timer counter register 11
Return Value	uint32_t: 32-bit free running timer value
Processing	Get free running timer tick count and return this.

8.4 Free Running Timer Count Value Compare Function

int32_t R_CmpTickCount(uint32_t freerun_counter)

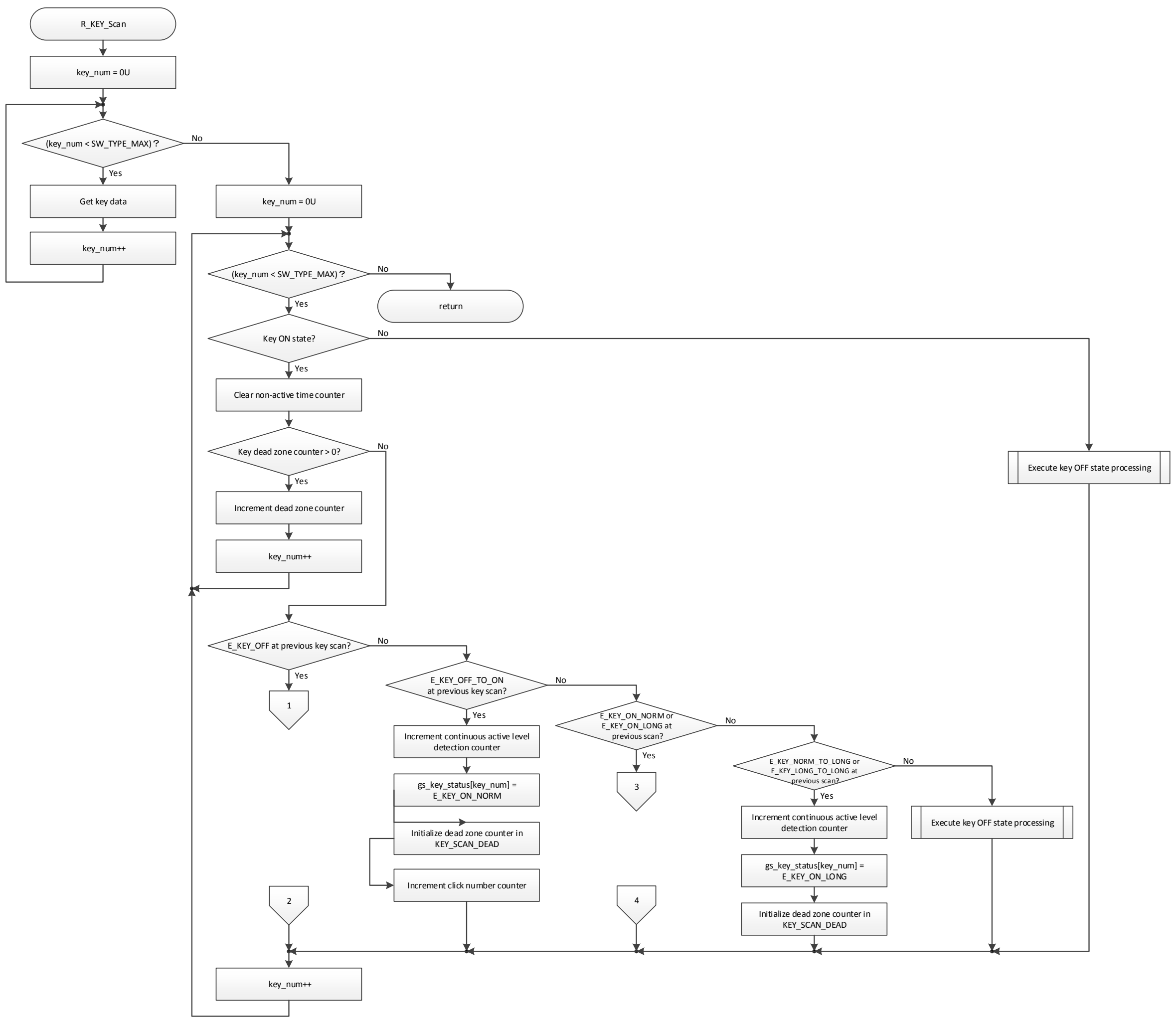
Description	Free running timer count value compare function
Argument	freerun_counter: Timer count value to be compared
Global Variables	None
SFR	None
Return Value	int32_t: Less than 0: unprocessed state 0 or higher: processed state
Processing	Compare values of free running timer tick count and specified count, return results.

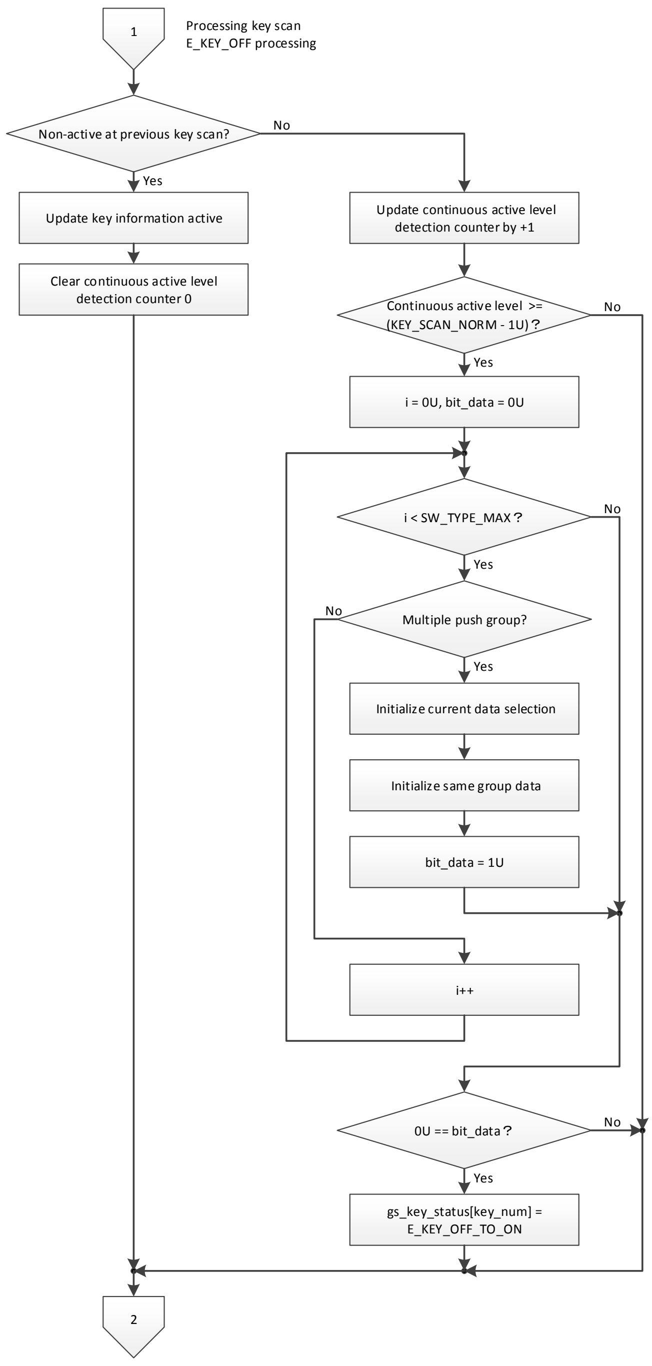
9. Key Judgement API

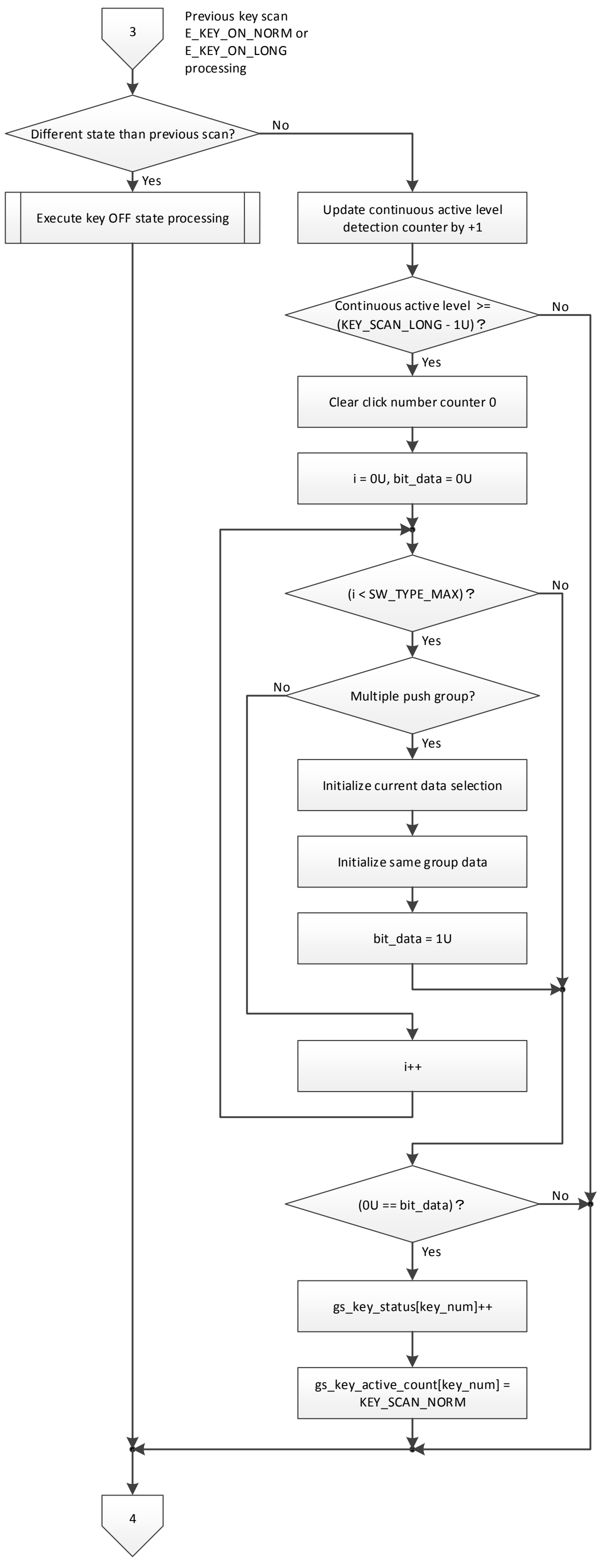
9.1 Key Scan Processing

void R_KEY_Scan(void)

Description	Key scan processing
Argument	None
Global Variables	g_con_key_setting[]: User setting key information gs_key_status[]: Key information array gs_key_setting_old: Last key information gs_key_active_count[]: Continuous active level detection counter array gs_key_dead_zone_count[]: Switch state post-change dead zone counter array gs_key_non_active_count[]: Continuous non-active level detection counter array gs_key_click_count[]: Click counter array
SFR	None
Return Value	None
Processing	Confirm input state of port set as key input, update key information.


R_KEY_Scan
key_num = 0U
(key_num < SW_TYPE_MAX)？
No
Yes
Get key data
key_num++
key_num = 0U
(key_num < SW_TYPE_MAX)？
No
Yes
return
Key ON state?
No
Yes
Clear non-active time counter
Key dead zone counter > 0?
No
Yes
Increment dead zone counter
key_num++
Execute key OFF state processing
E_KEY_OFF at previous key scan?
No
Yes
1
E_KEY_OFF_TO_ON at previous key scan?
No
Yes
Increment continuous active level detection counter
gs_key_status[key_num] = E_KEY_ON_NORM
Initialize dead zone counter in KEY_SCAN_DEAD
Increment click number counter
E_KEY_ON_NORM or E_KEY_ON_LONG at previous scan?
No
Yes
3
4
E_KEY_NORM_TO_LONG or E_KEY_LONG_TO_LONG at previous scan?
No
Yes
Increment continuous active level detection counter
gs_key_status[key_num] = E_KEY_ON_LONG
Initialize dead zone counter in KEY_SCAN_DEAD
Execute key OFF state processing
2
key_num++

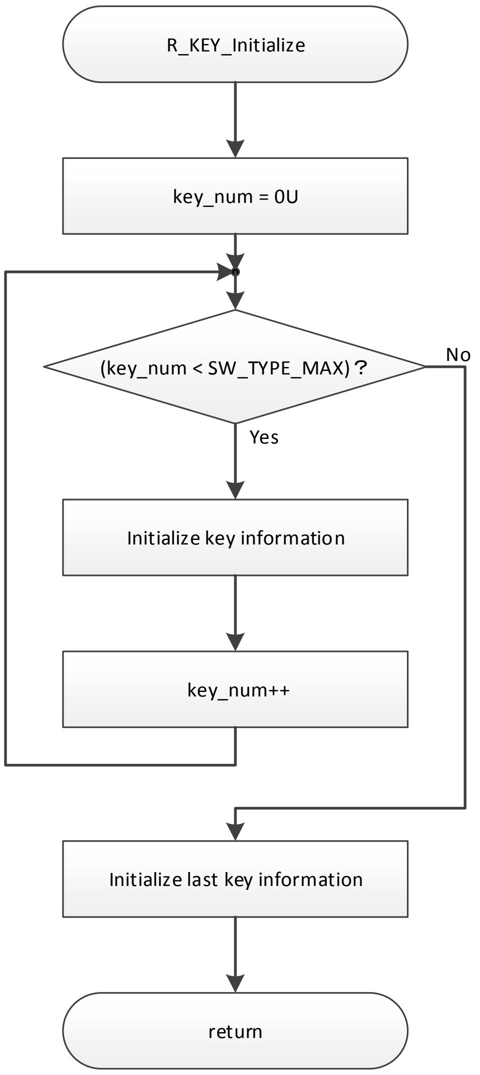

1
Processing key scan
E_KEY_OFF processing
Non-active at previous key scan?
No
Yes
Update key information active
Clear continuous active level detection counter 0
Update continuous active level detection counter by +1
Continuous active level >= (KEY_SCAN_NORM - 1U)？
No
Yes
i = 0U, bit_data = 0U
i < SW_TYPE_MAX？
No
Yes
Multiple push group?
No
Yes
Initialize current data selection
Initialize same group data
bit_data = 1U
i++
0U == bit_data？
No
Yes
gs_key_status[key_num] = E_KEY_OFF_TO_ON
2


3
Previous key scan
E_KEY_ON_NORM or
E_KEY_ON_LONG
processing
Different state than previous scan?
No
Yes
Execute key OFF state processing
Update continuous active level detection counter by +1
Continuous active level >= (KEY_SCAN_LONG - 1U)？
No
Yes
Clear click number counter 0
i = 0U, bit_data = 0U
(i < SW_TYPE_MAX)？
No
Yes
Multiple push group?
No
Yes
Initialize current data selection
Initialize same group data
bit_data = 1U
i++
(0U == bit_data)？
No
Yes
gs_key_status[key_num]++
gs_key_active_count[key_num] = KEY_SCAN_NORM
4

9.2 Key Information Initialization

void R_KEY_Initialize(void)

Description	Key information initialization
Argument	None
Global Variables	gs_key_setting_old: Last key information gs_key_active_count[]: Continuous active level detection counter array gs_key_dead_zone_count[]: Switch state post-change dead zone counter array gs_key_non_active_count[]: Continuous non-active level detection counter array gs_key_click_count[]: Click counter array gs_key_status[]: Key information array
SFR	None
Return Value	None
Processing	Initialize all key information.



9.3 Get Key Information (specified keys only)

e_key_status_t R_KEY_Get(e_key_t const key)

Description	Get key information (specified keys only)
Argument	key: SW names specified in e_key_t
Global Variables	gs_key_status[]: Key information array
SFR	None
Return Value	e_key_status_t:

E_KEY_OFF	0	The key is not pushed
E_KEY_OFF_TO_ON	1	The key was pushed (off to normal push)
E_KEY_ON_NORM	2	The key was pushed (normal push)
E_KEY_NORM_TO_LONG	3	The key was pushed (normal to long push)
E_KEY_ON_LONG	4	The key was pushed (push and hold)
E_KEY_LONG_TO_LONG	5	The key was pushed (long to long push)
E_KEY_DOUBLE_CLICK	6	The key was double pushed

Processing Get specified key information and send as return value.

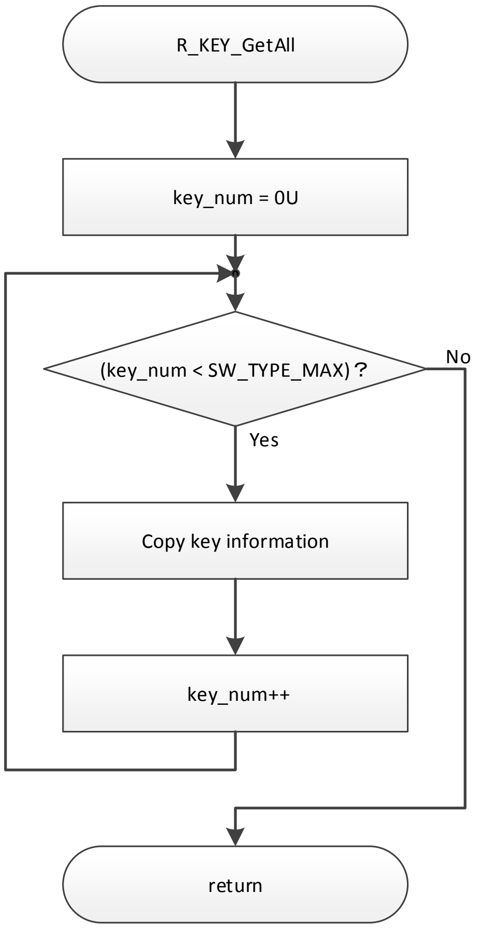
* This function simply returns the retrieved key information, therefore R_KEY_Scan() must be executed before this function.

9.4 Get Key Information (all keys)

void R_KEY_GetAll(e_key_status_t * p_status)

Description	Get key information (all keys)
Argument	p_status: Key information array pointer
Global Variables	gs_key_status[]: Key information array
SFR	None
Return Value	None
Processing	Store all key information in specified key information array.

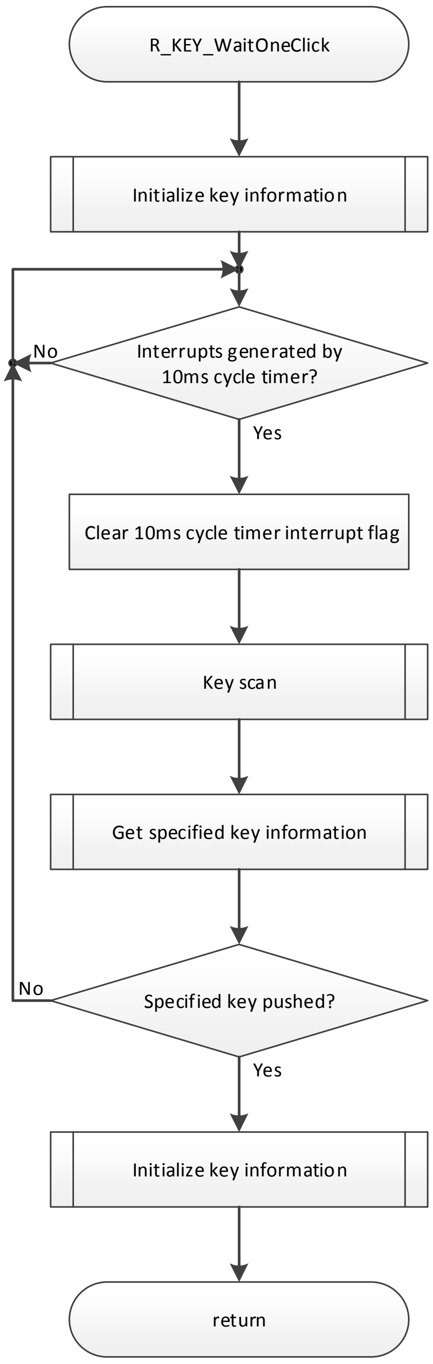
* This function simply copies key information already retreived, therefore R_KEY_Scan() must be executed before this function.



9.5 Specified Key Click Wait

void R_KEY_WaitOneClick(e_key_t const key)

Description	Specified key click wait
Argument	key: SW name set in e_key_t
Global Variables	g_10ms_timer_flag: 11ms timer cycle flag
SFR	None
Return Value	None
Processing	Wait in loop until specified key is pushed


R_KEY_WaitOneClick
Initialize key information
Interrupts generated by 10ms cycle timer?
No
Yes
Clear 10ms cycle timer interrupt flag
Key scan
Get specified key information
Specified key pushed?
No
Yes
Initialize key information
return

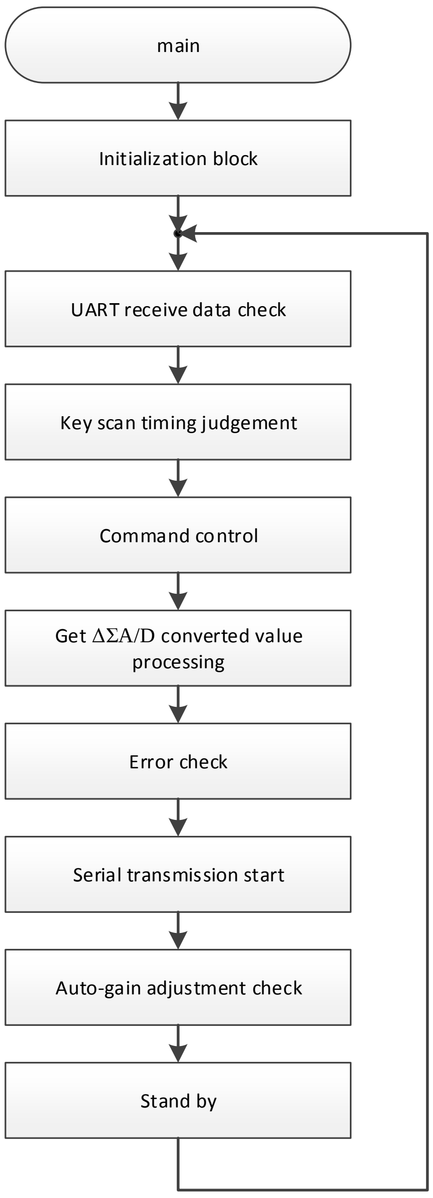
10. Sample Code Operation Specifications

The sample code described in this application note carries out 24-bit ΔΣA/D converter operations on a periodic basis, sending data to the PC through the serial array unit (UART1). The user can also select the 24-bit ΔΣA/D converter auto-scan mode during CS+ code generation to enable additional intermittent or continuous A/D conversion operations.

The following describes operations of the main function and communication specifications regarding data transmission to the PC.

10.1 Main Function Operations

Processing in the main function after initialization constitutes nine major blocks, as shown in the following flowchart.

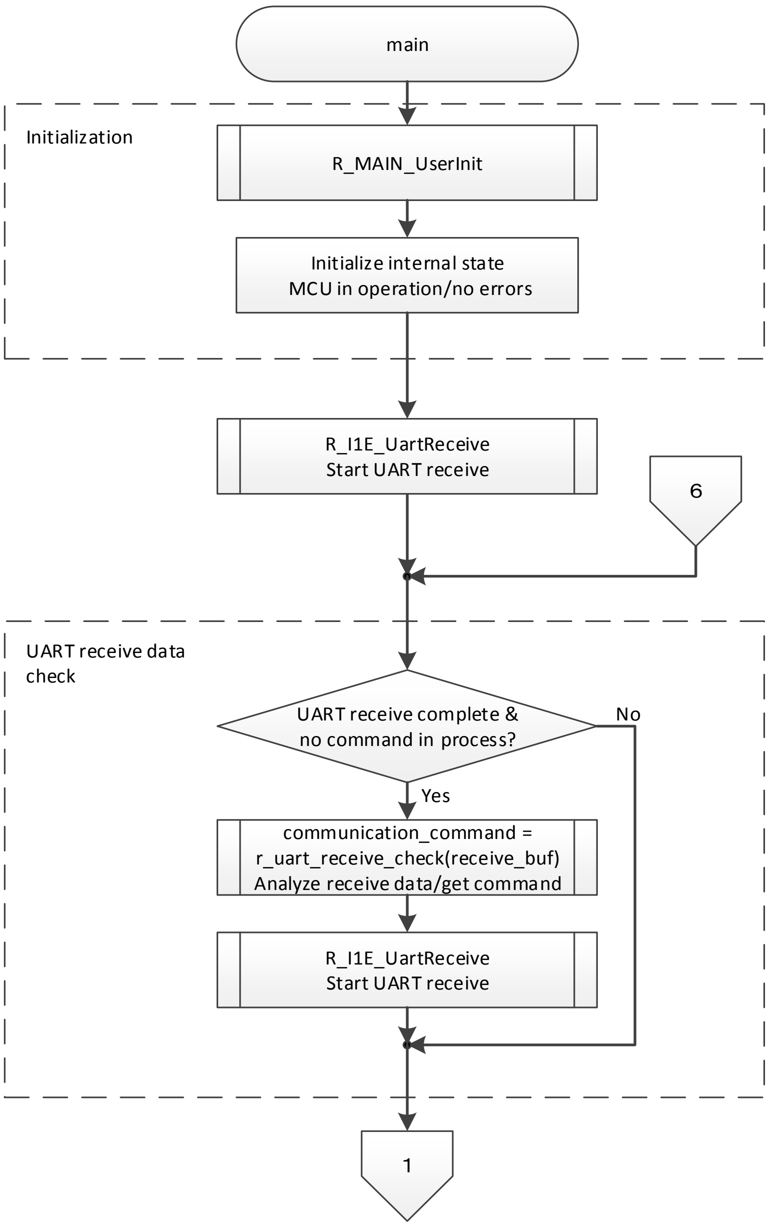

main
Initialization block
UART receive data check
Key scan timing judgement
Command control
Get ΔΣA/D converted value processing
Error check
Serial transmission start
Auto-gain adjustment check
Stand by

10.1.1 UART Receive data check block

The “UART receive data check block” executes the following: confirms state of data receive operation from UART, analyzes and obtains commands when data is received, and executes UART receive restart process.

The actual UART data receive operation is performed by the data transfer controller. This block is only executed when data receive has successfully completed.

The following flowchart shows details of the operation.

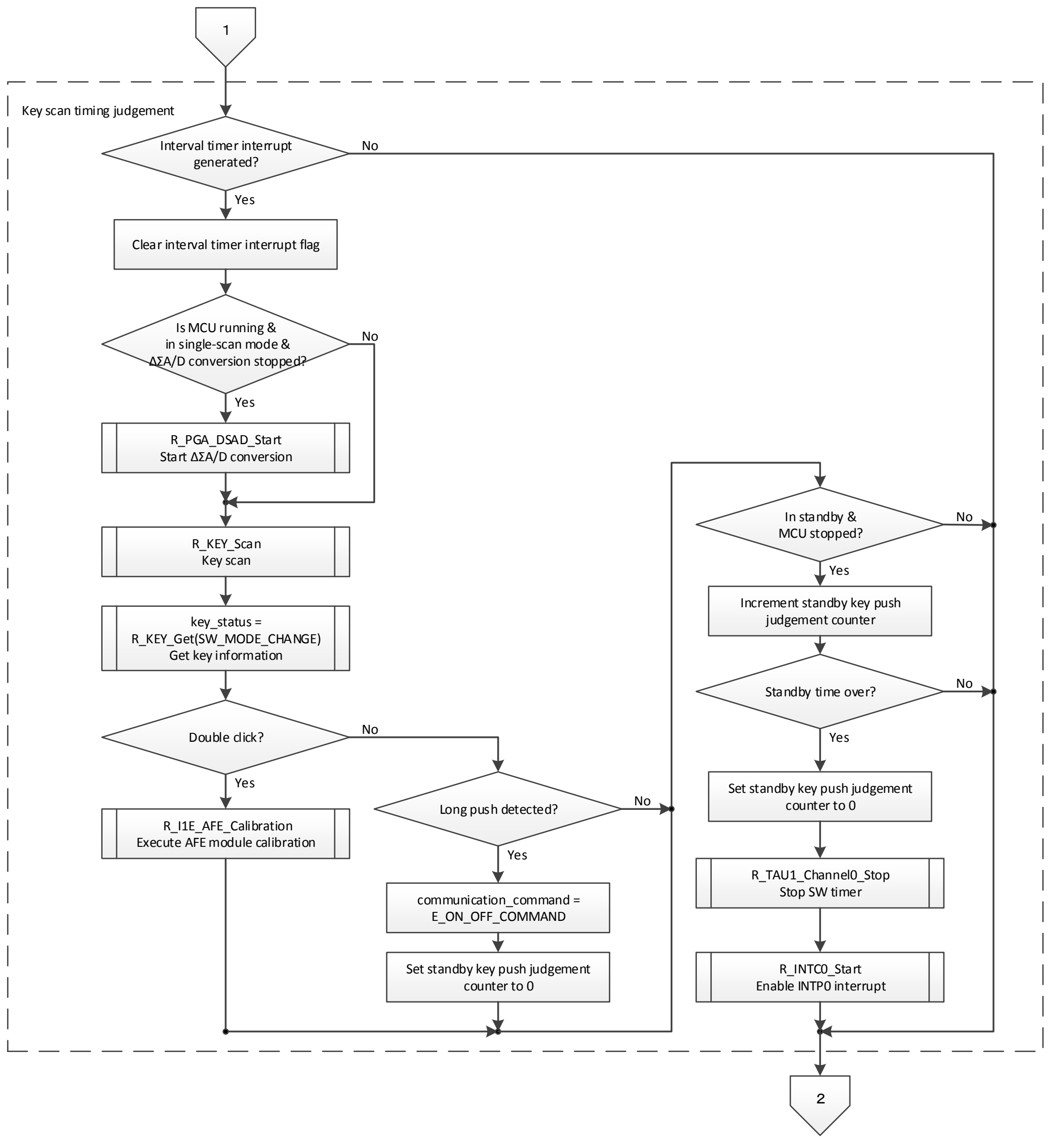

main
Initialization
R_MAIN_UserInit
Initialize internal state
MCU in operation/no errors
R_I1E_UartReceive
Start UART receive
6
UART receive data check
UART receive complete & no command in process?
No
Yes
communication_command = r_uart_receive_check(receive_buf)
Analyze receive data/get command
R_I1E_UartReceive
Start UART receive
1

10.1.2 Key scan timing judgement block

The “key scan timing judgement block” performs the following processes based on the periodic interrupt timer interrupt timing.

- In single-scan mode, executes intermittent ΔΣ A/D conversion
- Obtains key input information
- Performs processes according to the obtained key input information
  - Double click → executes AFE module calibration
  - Long push → A/D conversion ON/OFF toggle processing
- Monitors enabled key input during standby and disables key input when there is no input for a specified amount of time

The following flowchart shows details of the operation.

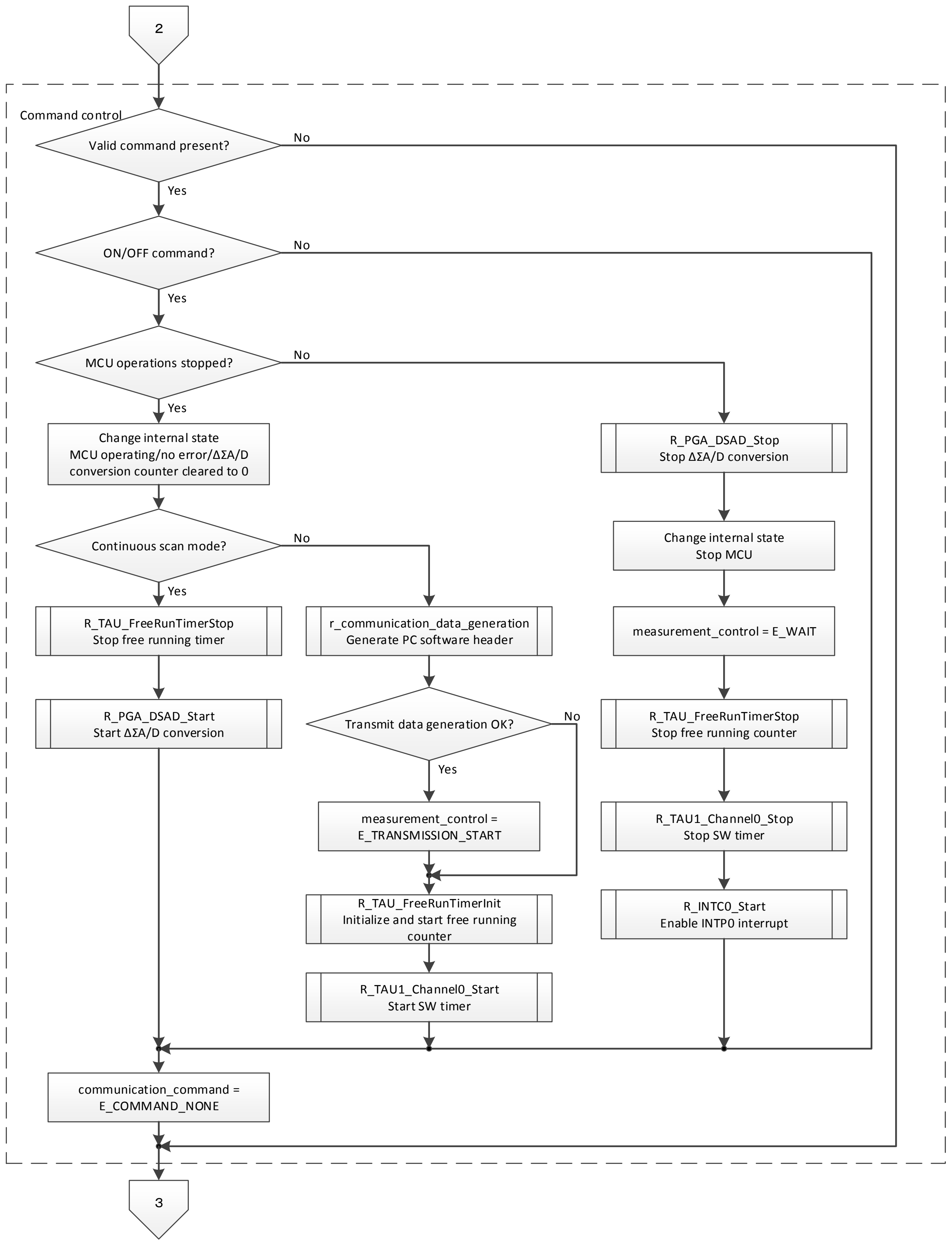

1
Key scan timing judgement
Interval timer interrupt generated?
No
Yes
Clear interval timer interrupt flag
Is MCU running & in single-scan mode & ΔΣA/D conversion stopped?
No
Yes
R_PGA_DSAD_Start
Start ΔΣA/D conversion
R_KEY_Scan
Key scan
key_status = R_KEY_Get(SW_MODE_CHANGE)
Get key information
Double click?
No
Yes
R_I1E_AFE_Calibration
Execute AFE module calibration
Long push detected?
No
Yes
communication_command = E_ON_OFF_COMMAND
Set standby key push judgement counter to 0
In standby & MCU stopped?
No
Yes
Increment standby key push judgement counter
Standby time over?
No
Yes
Set standby key push judgement counter to 0
R_TAU1_Channel0_Stop
Stop SW timer
R_INTC0_Start
Enable INTP0 interrupt
2

10.1.3 Command control block

The "command control block" performs each process as instructed by the received commands.

This sample only provides the ON/OFF command.

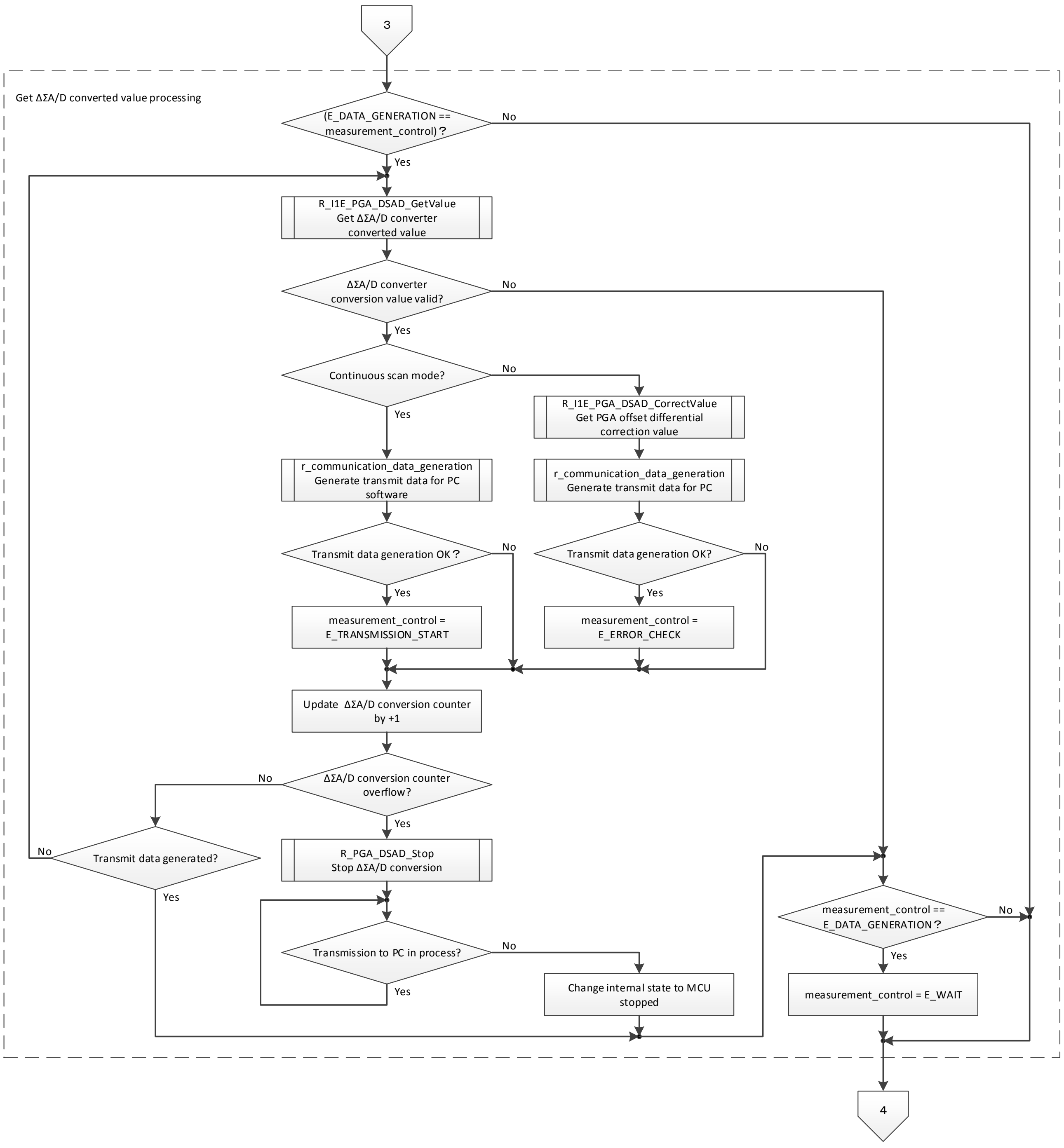
The following flowchart shows details of the operation.


2
Command control
Valid command present?
No
Yes
ON/OFF command?
No
Yes
MCU operations stopped?
No
Yes
Change internal state MCU operating/no error/ΔΣA/D conversion counter cleared to 0
R_PGA_DSAD_Stop Stop ΔΣA/D conversion
Change internal state Stop MCU
Continuous scan mode?
No
Yes
R_TAU_FreeRunTimerStop Stop free running timer
r_communication_data_generation Generate PC software header
measurement_control = E_WAIT
R_PGA_DSAD_Start Start ΔΣA/D conversion
Transmit data generation OK?
No
Yes
R_TAU_FreeRunTimerStop Stop free running counter
measurement_control = E_TRANSMISSION_START
R_TAU1_Channel0_Stop Stop SW timer
R_TAU_FreeRunTimerInit Initialize and start free running counter
R_INTC0_Start Enable INTP0 interrupt
R_TAU1_Channel0_Start Start SW timer
communication_command = E_COMMAND_NONE
3

10.1.4 Get ΔΣA/D converted value processing block

The “get ΔΣA/D converted value processing block” generates transmit data to send to the PC in scan mode. It also counts the number of transmit data and stops the ΔΣA/D conversion process after the specified number of conversions.

The following flowchart shows details of the operation.


3
Get ΔΣA/D converted value processing
(E_DATA_GENERATION == measurement_control)？
No
Yes
R_I1E_PGA_DSAD_GetValue Get ΔΣA/D converter converted value
ΔΣA/D converter conversion value valid?
Continuous scan mode?
R_I1E_PGA_DSAD_CorrectValue Get PGA offset differential correction value
r_communication_data_generation Generate transmit data for PC software
r_communication_data_generation Generate transmit data for PC
Transmit data generation OK？
Transmit data generation OK?
measurement_control = E_TRANSMISSION_START
measurement_control = E_ERROR_CHECK
Update ΔΣA/D conversion counter by +1
ΔΣA/D conversion counter overflow?
Transmit data generated?
R_PGA_DSAD_Stop Stop ΔΣA/D conversion
Transmission to PC in process?
Change internal state to MCU stopped
measurement_control == E_DATA_GENERATION？
measurement_control = E_WAIT
4

10.1.5 Error check block

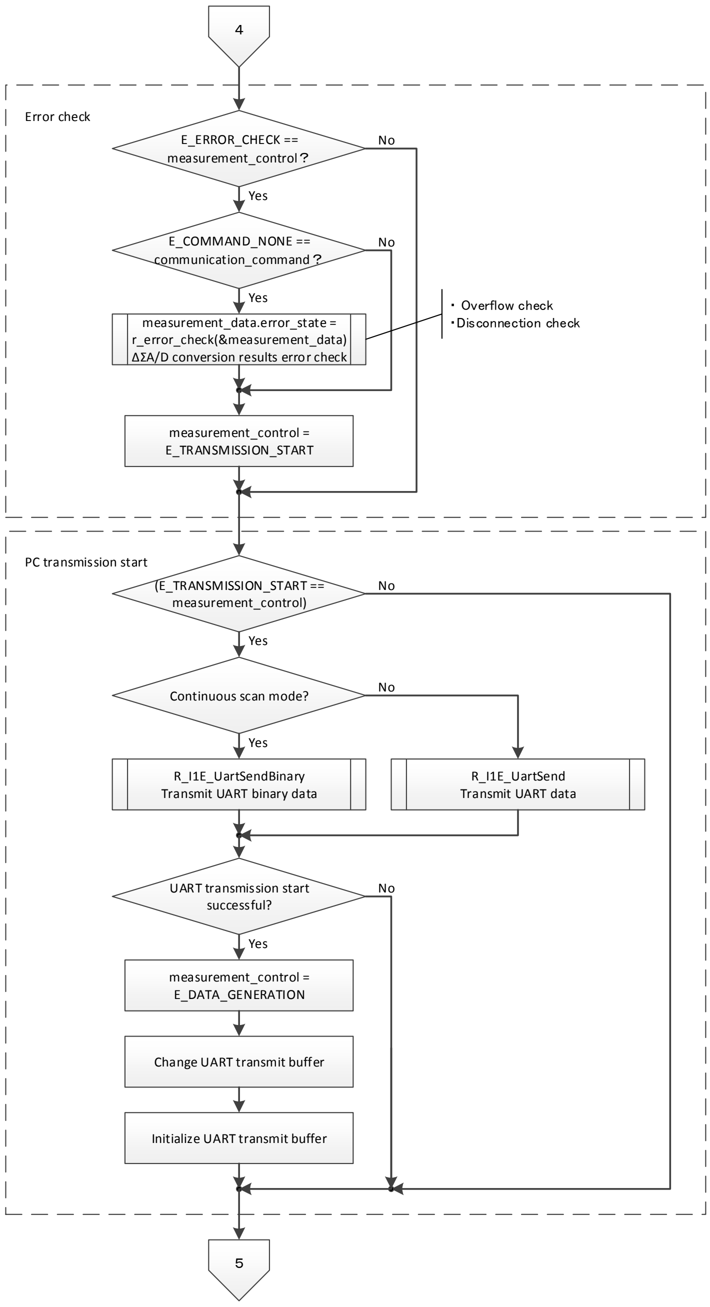
The “error check block” performs error check on the ΔΣA/D conversion results.

10.1.6 Serial transmission block

The “serial transmission block” starts data transmission to the PC in the method (Binary/ASCII) required for scan mode. The actual UART data transmission operation is performed by the data transfer controller. This block is only executes the data transmit start processing.

This block also controls the transmit buffer. When data transmit is successfully started, this block performs the transmit buffer for data storage change processing.

The following flowchart shows details of the operation.


4
Error check
E_ERROR_CHECK == measurement_control？
No
Yes
E_COMMAND_NONE == communication_command？
No
Yes
measurement_data.error_state = r_error_check(&measurement_data) ΔΣA/D conversion results error check
・Overflow check
・Disconnection check
measurement_control = E_TRANSMISSION_START
PC transmission start
(E_TRANSMISSION_START == measurement_control)
No
Yes
Continuous scan mode?
No
Yes
R_I1E_UartSendBinary Transmit UART binary data
R_I1E_UartSend Transmit UART data
UART transmission start successful?
No
Yes
measurement_control = E_DATA_GENERATION
Change UART transmit buffer
Initialize UART transmit buffer
5

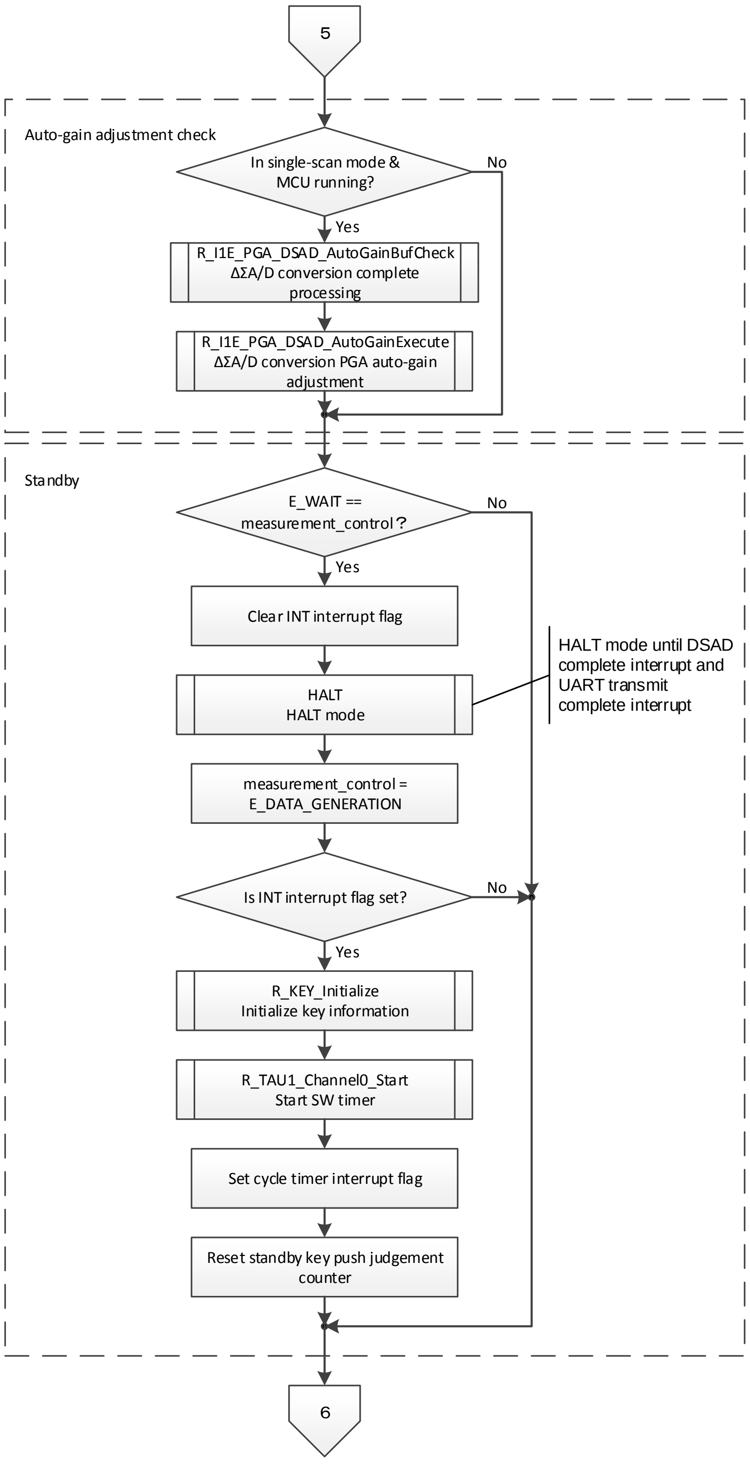
10.1.7 PGA auto-gain adjustment check block

The “PGA auto-gain adjustment check block” performs each process required for PGA auto-gain adjustment when operating in single-scan mode.

10.1.8 Standby block

In the “standby block,” operations are transitioned to the standby state by executing the HALT instruction after another block completeing processing. The standby state is released by all types of interrupts and initialization processes are carried out based on the return factor.

The following flowchart shows details of the operation.


5
Auto-gain adjustment check
In single-scan mode & MCU running?
No
Yes
R_I1E_PGA_DSAD_AutoGainBufCheck ΔΣA/D conversion complete processing
R_I1E_PGA_DSAD_AutoGainExecute ΔΣA/D conversion PGA auto-gain adjustment
Standby
E_WAIT == measurement_control?
No
Yes
Clear INT interrupt flag
HALT HALT mode
HALT mode until DSAD complete interrupt and UART transmit complete interrupt
measurement_control = E_DATA_GENERATION
Is INT interrupt flag set?
No
Yes
R_KEY_Initialize Initialize key information
R_TAU1_Channel0_Start Start SW timer
Set cycle timer interrupt flag
Reset standby key push judgement counter
6

10.1.9 Timing Charts

Two types of measurement operation timing can be selected by setting the ΔΣA/D converter DSADSCM bit (in auto-scan mode). Figure 1 shows timing in the single-scan mode and Figure 2 shows timing in the continuous scan mode.

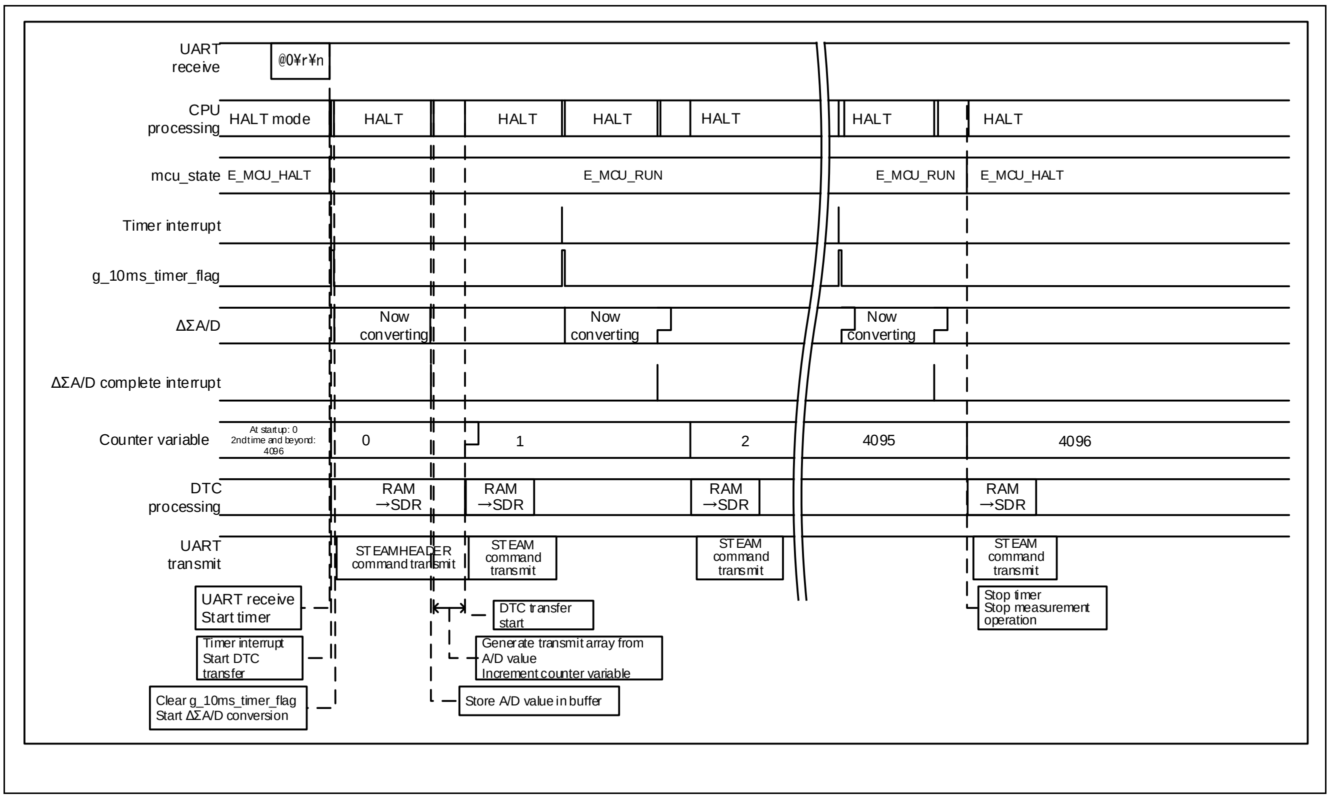

UART receive
@0¥r¥n
CPU processing
HALT mode
HALT
mcu_state
E_MCU_HALT
E_MCU_RUN
Timer interrupt
g_10ms_timer_flag
ΔΣA/D
Now converting
ΔΣA/D complete interrupt
Counter variable
At startup: 0 2nd time and beyond: 4096
0
1
2
4095
4096
DTC processing
RAM →SDR
UART transmit
STEAMHEADER command transmit
STEAM command transmit
UART receive Start timer
Timer interrupt Start DTC transfer
Clear g_10ms_timer_flag Start ΔΣA/D conversion
DTC transfer start
Generate transmit array from A/D value Increment counter variable
Store A/D value in buffer
Stop timer Stop measurement operation

Figure 1 Single Scan Mode Timing

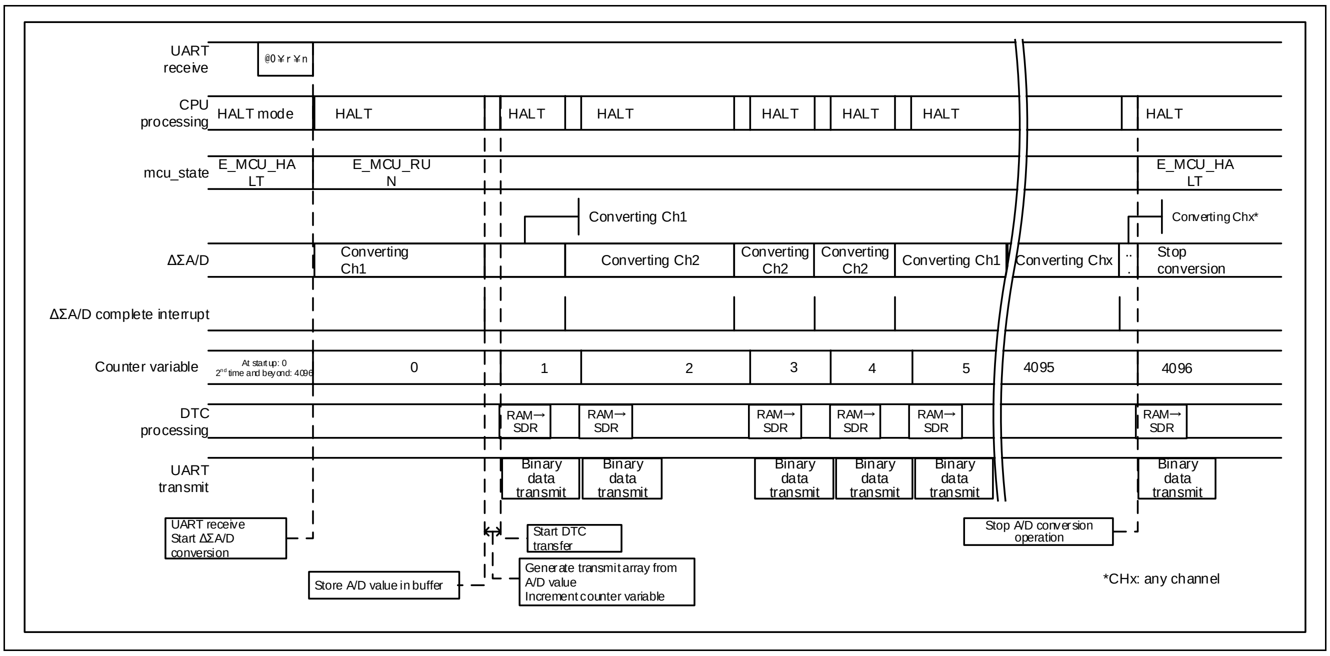


Figure 2 Continuous Scan Mode Timing (Ch1 x 2 times, Ch2 x 3 times)

10.2 Communication Specifications

The sample code interfaces with the PC, starts/stops operations, and transmits measurement results. Data is transmitted using UART communication, as indicated in the communication specifications below.

Table 10-1 provides the default settings for UART communications used by the sample code.

Table 10-1 UART Communication Settings

Baud rate (speed)	1Mbps
Data bits	8
Parity	None
Stop bits	1
Order	LSB first

10.2.1 Communication sequence

Figure 3 shows an outline of the communication sequence used by the sample code.

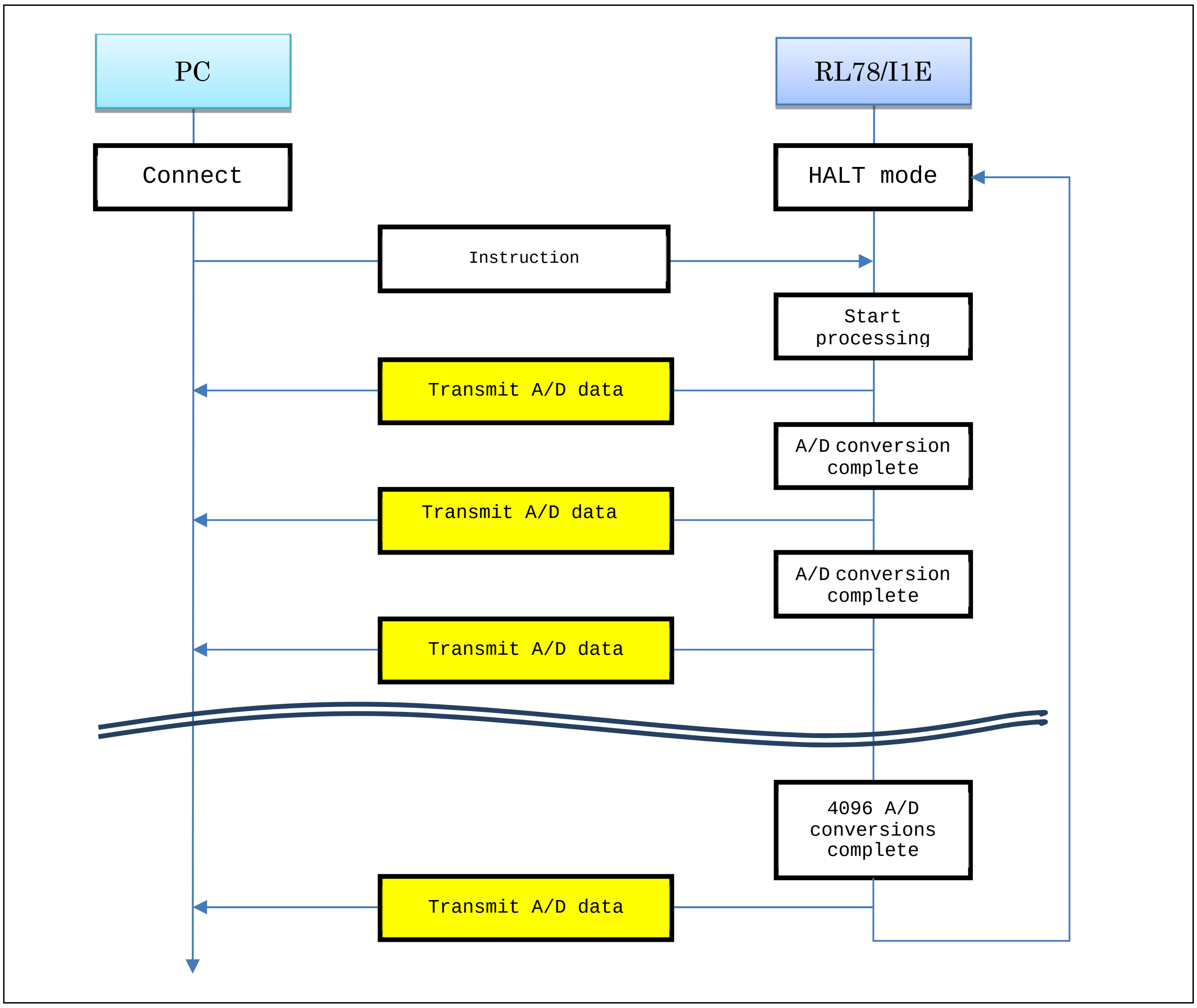


Figure 3 Communication Sequence

The sample code generates the data to be sent from the MCU to the PC using the r_communication_data_generation function. This function can generate data in one of three transmission formats (stream transfer, bulk transfer, and binary transfer) as detailed below.

Note that the sample code uses stream and binary transfers for data transmission.

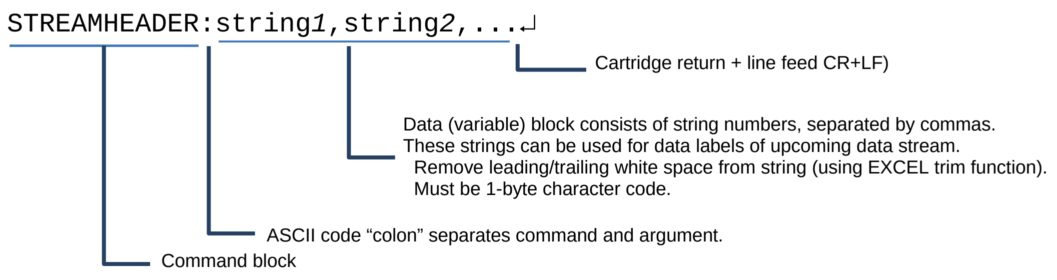
10.2.2 Stream Transfer

Stream transfer is normally used to send each measurement result from the MCU on a regular basis, allowing the data to be reflected in the graph as soon as it is received.

Stream transfer employs ASCII code to send numeric data, such as A/D converted results, by converting that data into a character string in sprintf or similar format. Stream transfers have two commands, STREAMHEADER and STREAM.

(1) STREAMHEADER Command Format

The STREAMHEADER command notifies the PC of the number and name of transmit data it will receive.


STREAMHEADER:string1,string2,...↵
Cartridge return + line feed CR+LF)
Data (variable) block consists of string numbers, separated by commas.
These strings can be used for data labels of upcoming data stream.
Remove leading/trailing white space from string (using EXCEL trim function).
Must be 1-byte character code.
ASCII code "colon" separates command and argument.
Command block

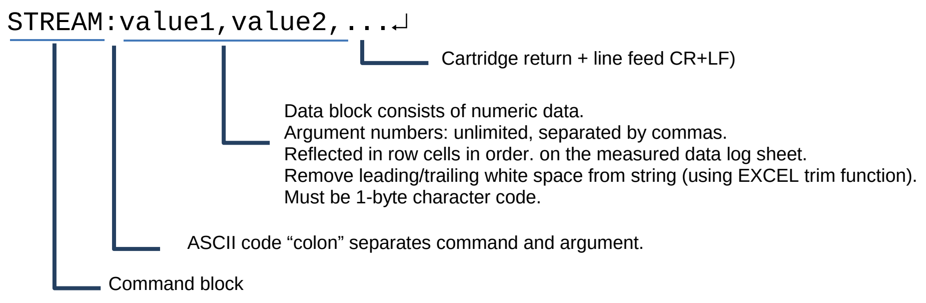
The data name can include multiple arguments in a string of characters separated by commas. The specified data name is recorded as the data header of the PC app.
*Please modify the define declaration D_UART_SEND_HEADER of file r_cg_main.c.

*Do not insert a space before or after commas used to separate string numbers.

```
Example)
STREAMHEADER:count,gain_set1,gain_set2,offset,rawdata,correct,error
STREAMHEADER:Count,Filter,Gram,Stability,Status
```

(2) STREAM Command Receive

The STREAM command sends the measured data as an ASCII base value.


STREAM:value1,value2,...↵
Cartridge return + line feed CR+LF)
Data block consists of numeric data.
Argument numbers: unlimited, separated by commas.
Reflected in row cells in order. on the measured data log sheet.
Remove leading/trailing white space from string (using EXCEL trim function).
Must be 1-byte character code.
ASCII code "colon" separates command and argument.
Command block

The data can include multiple arguments in a string of characters separated by commas. Record the data according to the data name specified in the header.

The transmit data format can be switched to DEC/HEX based on the PC app settings. However, you cannot mix ASCII and DEC/HEX formats.

*Do not insert a space before or after commas used to separate string numbers.

Example)

```
STREAM:0,1,8,16,363278,45409,0
```

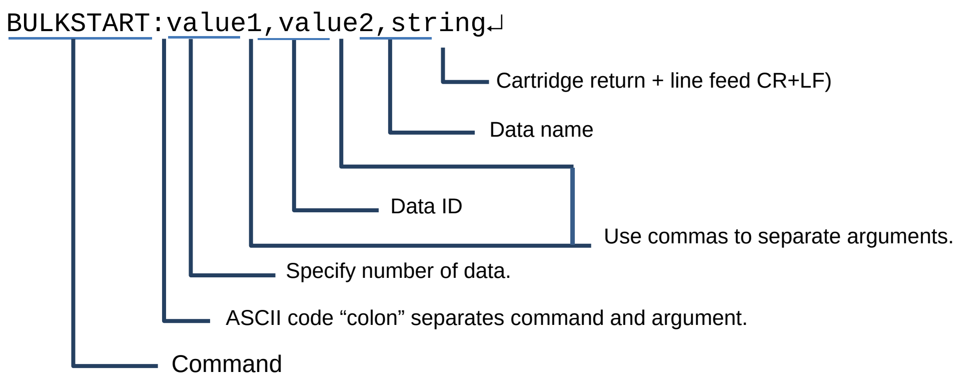
10.2.3 Bulk Transfer

Bulk transfer is used to send a large set of accumulated data from the MCU as requested, rather than sending one data at a time on a regular basis.

Bulk transfers are sent in a set of three commands, BULKSTART, BULK, and BULKEND.

(1) BULKSTART Command Format

When the PC app receives the BULKSTART command, it prepares a buffer for the number of receive data specified in the argument The BULKSTART command notifies the PC of the number and type of transmit data in the BULK command it will receive.


BULKSTART:value1,value2,string↵
Cartridge return + line feed CR+LF)
Data name
Data ID
Use commas to separate arguments.
Specify number of data.
ASCII code “colon” separates command and argument.
Command

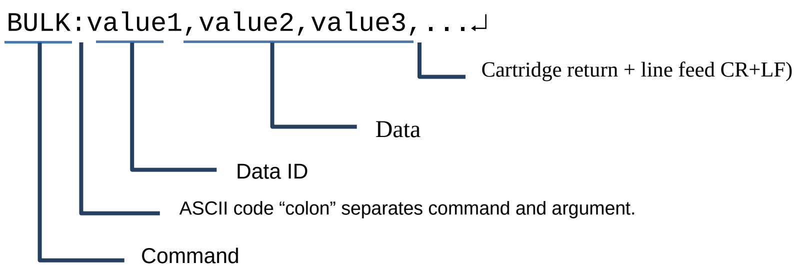
Example)

```
BULKSTART:100,0,RAWDATA
```

(2) BULK Command Format

The BULK command sends the measured data as an ASCII base value.

If the received value does not match with the data ID specified in the BULKSTART command, the PC app discards all data received up to that point.

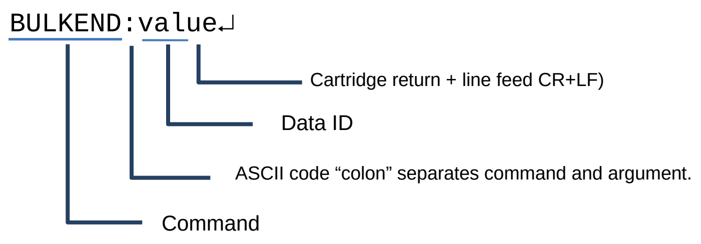

BULK:value1,value2,value3,...↵
Cartridge return + line feed CR+LF)
Data
Data ID
ASCII code “colon” separates command and argument.
Command

Example)

```
BULK:0,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19
BULK:0,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39
BULK:0,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59
BULK:0,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79
BULK:0,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99
```

BULKEND Command format

The BULKEND command notifies the PC that the BULK transfer will stop.


BULKEND:value↵
Cartridge return + line feed CR+LF)
Data ID
ASCII code “colon” separates command and argument.
Command

Example)

```
BULKEND:0
```

10.2.4 Binary Transfer

When in continuous scan mode, 4-byte binary data is sent to the PC.

Binary formats are described below.

① Transmit number counter 1 byte	② 24-bit A/D converted value 3 bytes

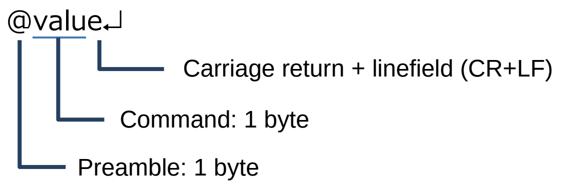
① Transmit number counter
This 1-byte transmit counter increments the count for each transmitted data, and returns to 0 when the count reaches 255.

② 24-bit A/D converted value
This 24-bitΔΣA/D converted value is for use in the differential input mode of the PC app.

10.2.5 Commands from the PC

These are command specifications for sending data when issuing instructions from the PC to the MCU. As MCU resources are limited, this is an extremely simple structure (4-byte fixed).

The format is shown below.



When @0↵ is received, the sample code performs the same processing that is executed for a SW long push on the RL78/I1E board (start/stop A/D conversion processing).

11. How to Change Settings

The sample code operations can be modified by changing the hardware resources used for CS+ code generation and the user definitions declared in the sample code. The following describes how to change these settings.

(1) Changes to main system clock

- Change setting here: Code generation → Clock setting → Main system clock (fMAIN) source
- Change the D_MCU_CLOCK_MHZ definition (refer to Table 4-4).

(2) Changes to SBIAS OUTPUT VOLTAGE

Change setting here: Code generation → PGA+ΔΣA/D converter → SBIAS OUTPUT VOLTAGE

(3) Changes to PGA + ΔΣA/D converter

① 24-bit ΔΣA/D converter operating clock
Change setting here: Code generation → Clock setting → 24-bit ΔΣA/D converter operating clock (fDSAD) source setting

② ΔΣA/D converter operating mode setting
Change setting here: Code generation → PGA +ΔΣA/D converter → ΔΣA/D converter operating mode setting

③ Auto-scan mode setting
Change setting here: Code generation → PGA +ΔΣA/D converter → Auto-scan mode setting

④ Setting the multiplexer in use
Change selection here: Code generation → PGA +ΔΣA/D converter → Input multiplexer setting (remove check)
Note: The sample code only supports 1 channel but the API supports multiple channels.

⑤ Change individual multiplexers

(a) Input mode
Change setting here: Code generation → PGA +ΔΣA/D converter → Input mode of multiplexer in use

(b) PGA gain setting
- Change settings here: Code generation → PGA +ΔΣA/D converter → Multiplexer n → GSET1, GSET2 settings
- Change the D_DSAD_AUTO_GAIN_USE definition to PGA auto-gain adjustment disabled setting (refer to Table 4-4).

(c) PGA gain setting: auto-gain adjustment setting (GSET1 auto setting)
- Change settings here: Code generation → PGA +ΔΣA/D converter → Multiplexer n → GSET1, GSET2 settings
- Change the D_DSAD_AUTO_GAIN_USE definition to PGA auto-gain adjustment enabled setting (refer to Table 4-4).
- Change the D_DSAD_AUTO_GAIN_TRIGGER_SEC definition (refer to Table 4-4).

(d) PGA offset adjustment voltage setting
Change setting here: Code generation → PGA +ΔΣA/D converter → Multiplexer n → offset adjustment voltage

(e) Oversampling rate
Change setting here: Code generation → PGA +ΔΣA/D converter → Multiplexer n → oversampling rate

(f) ΔΣA/D converter: A/D conversion number

Change number of conversions here: Code generation → PGA +ΔΣA/D converter → Multiplexer n → A/D conversion number

(g) ΔΣA/D converter: averaging processing

Change number of data here: Code generation → PGA +ΔΣA/D converter → Multiplexer n → Averaging processing, number of data for averaging.

(4) Changes to A/D conversion cycle

① When operating in auto-scan mode selection→ continuous scan mode setting
Refer to changes made to OSR setting and ΔΣA/D converter operating mode setting

② When operating in auto-scan mode selection→ single-scan mode setting
Change interval period here: Code generation → Timer array unit 1 → Channel 0 → Interval period
Note: The sample code uses the same timer for A/D start timing and KEY scan timing in single-scan mode.

(5) Changes to DSAD conversion value storage buffer size

Change the D_DSAD_VALUE_BUFFER_SIZE definition (refer to Table 4-4).

(6) Changes to serial communication setting

① UART transmission enable/disable
Change D_UART_SEND_USE setting (refer to Table 4-4).

② UART communication baud rate

- Change setting here: Code generation → Serial array unit→ Serial array unit 0 → UART1 → Baud rate
- Change the baud rate in the RL78I1E_evaluation tool_analysis.xlsm file. Go to Serial Communication Settings->Baudrate:

③ Change UART to ch0

- Select UART0 here: Code generation → Serial array unit→ Serial array unit 0 → Channel 1 pull-down menu → select UART0
- Copy content set in UART1 to UART0.
- Add the following process to the r_uart0_callback_receiveend function in r_cg_sau_user.c.
  g_rx_in_process_flag = 0U;
- Add the following process to the r_uart0_callback_sendend function in r_cg_sau_user.c.
  g_tx_in_process_flag = 0U;
- Code generation → Data transfer controller → DTC setting: change start factor from UART0 transmmission to UART0 reception
- Code generation → Data transfer controller → DTCD0 → transfer source: change from 0xFF46 to 0xFF12
- Code generation → Data transfer controller → DTCD1 → transfer source: change from 0xFF44 to 0xFF10
- Change contents of the R_DTC0_Init function of r_cg_dtc.c to the following
  dtc_controldata_0.dtsar = _FF12_DTCD0_SRC_ADDRESS;
- Change contents of the R_DTC1_Init function of r_cg_dtc.c to the following
  dtc_controldata_1.dtdar = _FF10_DTCD1_DEST_ADDRESS;
- Replace all R_UART1 functions in r_cg_main.c with R_UART0 functions

④ UART transmission buffer size
Change the D_UART_SEND_BUFFER_SIZE definition (refer to Table 4-4).

(7) Changes to transmission data

① Stream data

- Change the D_STREAMHEADER definition (refer to Table 4-3)
- Change the r_communication_data_generation function (refer to Chapter10.2.2)

② Bulk data

- Change the r_communication_data_generation (refer to Chapter 10.2.3)

③ Total number of transfer data from BULKSTART command to BULKEND command
Change the D_BULK_NUM definition (refer to Table 4-3).

④ Number of transfer data of each BULK command
Change the D_BULK_COMMAND_NUM definition (refer to Table 4-3).

(8) Changes to flash memory

① Enable/disable use of flash memory
Change the D_FLASH_MEMORY_DATA_USE definition (refer to Table 4-4).

② Data to be stored in flash memory
Change the str_flash_data_t definition (refer to Table 4-4).

③ Enable/disable writing value to flash memory in R_MAIN_UserInit function

- Change the D_FLASH_FORCE_WRITING definition (refer to Table 4-4).
- Change initial value of data flash memory storage data structure variable (g_flash_value).

(9) Changes to PGA error measurement

① Enable/disable error measurement
Change the D_DSAD_CORRECT_USE definition (refer to Table 4-4).

② Measurement reference voltage used for error measurement
Change the D_GAIN_ERROR_REFERENCE_mV definition (refer to Table 4-4).

③ Error measurement input multiplexer setting
Change the D_DSAD_CORRECT_MPXn definition (refer to Table 4-4).

(10) Changes to disconnection detection processing

① Number of disconnection checks
Change the D_DISCONNECTION_CHECK_COUNT definition (refer to Table 4-4).

② Disconnection determination voltage
Change the D_DISCONNECTION_THRESHOLD_mV definition (refer to Table 4-4).

(11) Changes to debug LED

① Enable/disable use of debug LED
Change the D_DEBUG_LED_USE definition (refer to Table 4-6).

② LED drive port for debug LED
Change the D_DEBUG_LED_PORT definition (refer to Table 4-6).

(12) Changes to KEYSCAN

① Key port enable level
Change the DEF_KEY_ACTIVE definition (refer to Table 4-5).

② Key port assignments, processing assignments, group information

- Change e_key_t enumeration (refer to Table 4-12).
- Change g_con_key_setting global variables (refer to section 4.5.1).

③ Key determination time

- Change the KEY_SCAN_NORM definition (refer to Table 4-5).
- Change the KEY_SCAN_LONG definition (refer to Table 4-5).
- Change the KEY_SCAN_DEAD definition (refer to Table 4-5).
- Change the KEY_SCAN_NOT definition (refer to Table 4-5).
- Change the KEY_SCAN_DOUBLE definition (refer to Table 4-5).

④ Key scan timing

Change here: Code generation → Timer array unit 1 → Channel 0 → Interval period

Note: The sample code uses the same timer for A/D start timing and key scan timing in single-scan mode.

12. API Usage Notes

This section explains the limitations and usage notes that apply to use of this API to control RL78/I1E. Please read this section carefully before using the API.

12.1 API Usage Notes

12.1.1 Calibration data

This sample code stores the calibration data in the data flash memory. If incorrect calibration data is stored in the data flash memory by mistake, the following method is recommended for erasing the incorrect data.

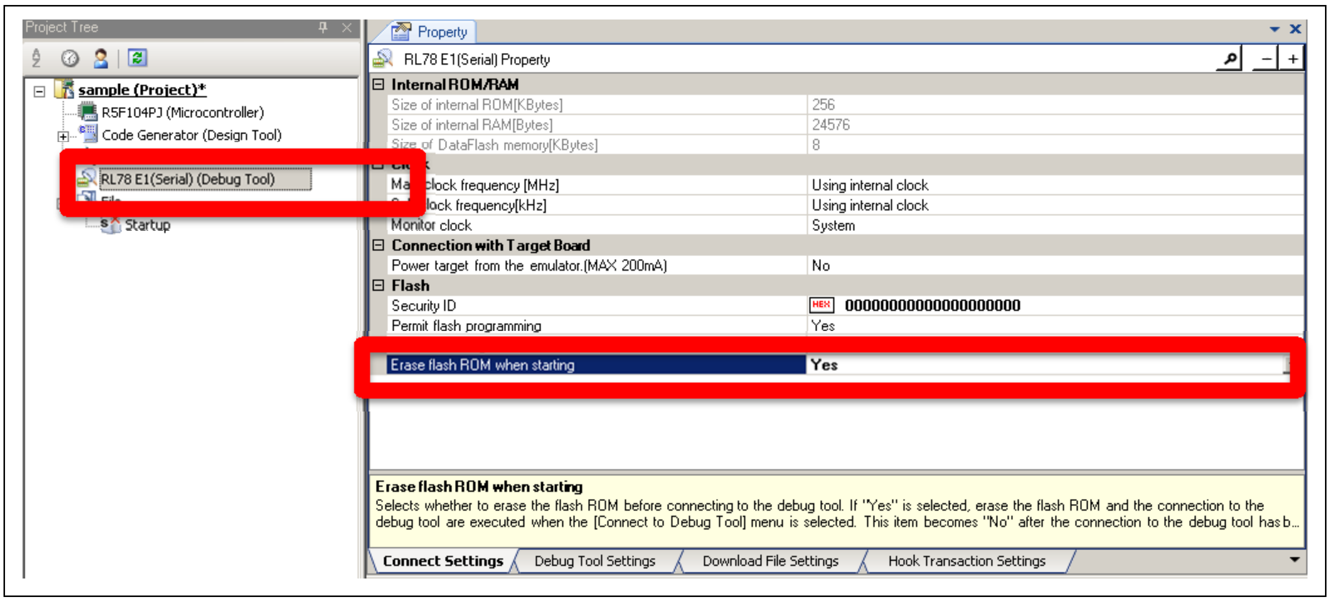
・Change the CS+ debug tool settings and erase the calibration data.

① Make sure debug tool is in disconnected state

② Double click **RL78 E1(Serial) (debug tool)** in CS+ project tree

③ In the **Property → Connection Setting** tab, select **YES** for **Erase flash ROM at startup**. (Refer to Figure 12.1).

④ From the Under the **Debug** tab located at the top of the window, select **Download debug tool**.



sample (Project)*
R5F104PJ (Microcontroller)
Code Generator (Design Tool)
RL78 E1(Serial) (Debug Tool)
Startup

Internal ROM/RAM	
Size of internal ROM[KBytes]	256
Size of internal RAM[Bytes]	24576
Size of DataFlash memory[KBytes]	8
Ma clock frequency [MHz]	Using internal clock
lock frequency[kHz]	Using internal clock
Monitor clock	System
Connection with Target Board	
Power target from the emulator.(MAX 200mA)	No
Flash	
Security ID	00000000000000000000
Permit flash programming	Yes
Erase flash ROM when starting	Yes

Erase flash ROM when starting
Selects whether to erase the flash ROM before connecting to the debug tool. If "Yes" is selected, erase the flash ROM and the connection to the debug tool are executed when the [Connect to Debug Tool] menu is selected. This item becomes "No" after the connection to the debug tool has b...

Figure 12.1 CS+ Debug Tool Setting Window

Website and Support

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

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

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Nov. 09, 2015	---	First edition issued

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.

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, HALII 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