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 ..................................................................................................................... 2
2. Hardware Explanation ......................................................................................................... 3
3. API Functions ..................................................................................................................... 6
4. API Definitions .................................................................................................................... 8
   BULK division number ............................................................................................................. 11
   BULK division remainder ....................................................................................................... 11
5. Analog Front-end APIs ..................................................................................................... 26
6. Flash Memory API ............................................................................................................ 41
7. UART Communication API ............................................................................................... 43
8. Free Running Timer API .................................................................................................. 47
9. Key Judgement API ......................................................................................................... 48
10. Sample Code Operation Specifications .......................................................................... 54
11. How to Change Settings ................................................................................................. 67
12. API Usage Notes .............................................................................................................. 71

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products
........................................................................................................................................... 1
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>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation board</td>
<td>・ RL78/I1E analog evaluation board (RTK50011CCC00000BR)</td>
</tr>
<tr>
<td></td>
<td>・ RL78/I1E CPU board (FB-R5F11CCC-TB)</td>
</tr>
<tr>
<td>MCU used</td>
<td>R5F11CCC (RL78/I1E) 36 pins</td>
</tr>
<tr>
<td>Operating frequency</td>
<td>32MHz</td>
</tr>
<tr>
<td>Operating voltage</td>
<td>5.0V</td>
</tr>
<tr>
<td>Integrated Development</td>
<td>V3.01.00 (19 Aug 2015)</td>
</tr>
<tr>
<td>Environment (CS+ for CA,CX)</td>
<td></td>
</tr>
<tr>
<td>C compiler (CS+)</td>
<td>CA78K0R</td>
</tr>
<tr>
<td></td>
<td>V5.00.00.02 (03 Jul 2014)</td>
</tr>
<tr>
<td>RL78/I1E Code Library (CS+)</td>
<td>V1.02.00.06 (12 Aug 2015)</td>
</tr>
<tr>
<td>Data Flash Library</td>
<td>RL78Family data flash library Type04</td>
</tr>
<tr>
<td></td>
<td>Ver.1.05</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Peripheral Function</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Clock generator</strong></td>
<td><strong>Operating mode</strong></td>
</tr>
<tr>
<td><strong>Clock</strong></td>
<td><strong>Main system clock (fMAIN)</strong></td>
</tr>
<tr>
<td><strong>24-bit ΔΣA/D converter operating clock (fDSAD) source setting</strong></td>
<td><strong>RTC operating clock (fRTC) source setting</strong></td>
</tr>
<tr>
<td><strong>High-speed on-chip oscillator clock (fHOCO) setting</strong></td>
<td><strong>32(MHz)</strong></td>
</tr>
<tr>
<td><strong>High-speed system clock (fMX) setting</strong></td>
<td><strong>none</strong></td>
</tr>
<tr>
<td><strong>24-bit ΔΣA/D converter operating clock (fDSAD) setting</strong></td>
<td><strong>fHOCO 32(MHz)</strong></td>
</tr>
<tr>
<td><strong>RTC, interval timer/timer RJ operating clock</strong></td>
<td><strong>fIL 15(kHz)</strong></td>
</tr>
<tr>
<td><strong>CPU and peripheral clock (fCLK)</strong></td>
<td><strong>fHOCO 32000(kHz)</strong></td>
</tr>
<tr>
<td><strong>Port functions</strong></td>
<td><strong>Port13</strong></td>
</tr>
<tr>
<td><strong>Timer array Unit</strong></td>
<td><strong>Channel 0</strong></td>
</tr>
<tr>
<td><strong>Unit 1</strong></td>
<td><strong>Interval period</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Generates INTTM10 interrupt at start of count</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Interrupt</strong></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Timer array Unit</strong></td>
<td><strong>Channel 1</strong></td>
</tr>
<tr>
<td><strong>Unit 1</strong></td>
<td><strong>Operating mode setting</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Interval timer setting</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Interrupt</strong></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Watchdog timer</strong></td>
<td><strong>Watchdog timer operation setting</strong></td>
</tr>
<tr>
<td><strong>PGA+ΔΣA/D converter</strong></td>
<td><strong>Selected multiplexer setting</strong></td>
</tr>
<tr>
<td></td>
<td><strong>SBIAS output voltage</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Disconnection detection</strong></td>
</tr>
<tr>
<td></td>
<td><strong>ΔΣA/D converter operation mode setting</strong></td>
</tr>
<tr>
<td></td>
<td><strong>ΔΣA/D converter start trigger setting</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Auto-scan mode setting</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Interrupt setting</strong></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Peripheral Function</td>
<td>Setting</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------</td>
</tr>
<tr>
<td>PGA+ ΔΣA/D CONVERTER</td>
<td>Gain setting $G_{SET1}$ 1x</td>
</tr>
<tr>
<td></td>
<td>Gain setting $G_{SET2}$ 8x</td>
</tr>
<tr>
<td></td>
<td>Offset calibration voltage setting 16 (0mV)</td>
</tr>
<tr>
<td></td>
<td>Oversampling rate 256</td>
</tr>
<tr>
<td></td>
<td>A/D conversion count Specify 1 to 8092 times as the setting value of register PGA0CTL2 1 (1 time)</td>
</tr>
<tr>
<td></td>
<td>Averaging procedure Do not execute averaging procedure</td>
</tr>
<tr>
<td>Serial</td>
<td>Channel 1 UART0 transmission/reception function</td>
</tr>
<tr>
<td></td>
<td>Data bit length 8 bits</td>
</tr>
<tr>
<td></td>
<td>Data transfer direction LSB</td>
</tr>
<tr>
<td></td>
<td>Parity No parity</td>
</tr>
<tr>
<td></td>
<td>Stop bit length 1 bit</td>
</tr>
<tr>
<td></td>
<td>Receive data level Standard</td>
</tr>
<tr>
<td></td>
<td>Transfer rate 1000000(bps)</td>
</tr>
<tr>
<td></td>
<td>Interrupt Receive completed interrupt setting (INTSR1) Level 3 (lowest priority)</td>
</tr>
<tr>
<td></td>
<td>Call back function Receive complete</td>
</tr>
<tr>
<td>Data transfer control</td>
<td>DTC base address 0xffd00</td>
</tr>
<tr>
<td>DTCD0</td>
<td>Control data0 (DTCD0) UART1 reception</td>
</tr>
<tr>
<td></td>
<td>Control data1(DTCD1) UART1 transmission</td>
</tr>
<tr>
<td>Data transfer control</td>
<td>Transfer mode setting Normal mode</td>
</tr>
<tr>
<td>DTCD0</td>
<td>Transfer data size setting 8 bits</td>
</tr>
<tr>
<td></td>
<td>Transfer origin address 0xFF46 (fixed)</td>
</tr>
<tr>
<td></td>
<td>Transfer destination address 0xF900 (incremented)</td>
</tr>
<tr>
<td></td>
<td>Number of transfers 1</td>
</tr>
<tr>
<td></td>
<td>Block size 1</td>
</tr>
<tr>
<td>Data transfer control</td>
<td>Transfer mode setting Normal mode</td>
</tr>
<tr>
<td>DTCD1</td>
<td>Transfer data size setting 8 bits</td>
</tr>
<tr>
<td></td>
<td>Transfer origin address 0xF910 (incremented)</td>
</tr>
<tr>
<td></td>
<td>Transfer destination address 0xFF44 (fixed)</td>
</tr>
<tr>
<td></td>
<td>Number of transfers 1</td>
</tr>
<tr>
<td></td>
<td>Block size 1</td>
</tr>
<tr>
<td>Interrupt</td>
<td>INT10 INTP0 used</td>
</tr>
<tr>
<td>External interrupt</td>
<td>Falling edge detected</td>
</tr>
<tr>
<td></td>
<td>Level 3 (lowest priority)</td>
</tr>
</tbody>
</table>
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 reserved 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".

![Memory Address Space Used by Sample Code](image-url)
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>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_I1E_Variable_Initialize</td>
<td>Variable initialization processings used in sample code</td>
</tr>
<tr>
<td>R_I1E_RingBuffer_Initialize</td>
<td>DSAD ring buffer initialization function</td>
</tr>
<tr>
<td>R_I1E_AFE_Calibration</td>
<td>AFE module calibration function</td>
</tr>
<tr>
<td>R_I1E_PGA_DSAD_GetResult_1Shot</td>
<td>24-bit ΔΣA/D converter specified channel 1-shot differential input A/D conversion function</td>
</tr>
<tr>
<td>R_I1E_PGA_DSAD_GainRegSet</td>
<td>24-bit ΔΣA/D converter gain setting function</td>
</tr>
<tr>
<td>R_I1E_PGA_DSAD_OffsetRegSet</td>
<td>24-bit ΔΣA/D converter offset setting function</td>
</tr>
<tr>
<td>R_I1E_PGA_DSAD_OscRegSet</td>
<td>24-bit ΔΣA/D converter OSR setting function</td>
</tr>
<tr>
<td>R_I1E_PGA_DSAD_SettingRegGet</td>
<td>Auto-scan mode input multiplexer settings retentive variable storage function</td>
</tr>
<tr>
<td>R_I1E_PGA_DSAD_SettingRegSet</td>
<td>Set structure variable information to DSAD setting</td>
</tr>
<tr>
<td>R_I1E_PGA_DSAD_CorrectValue</td>
<td>Get PGA offset error correction value function</td>
</tr>
<tr>
<td>R_I1E_PGA_DSAD_GetValue</td>
<td>Get 24-bit ΔΣA/D converted value function</td>
</tr>
<tr>
<td>R_I1E_PGA_DSAD_AutoGainInit</td>
<td>PGA auto-gain variable initialization</td>
</tr>
<tr>
<td>R_I1E_PGA_DSAD_AutoGainBufCheck</td>
<td>24-bit ΔΣA/D converter conversion complete PGA auto-gain adjustment detection processing function</td>
</tr>
<tr>
<td>R_I1E_PGA_DSAD_AutoGainExecute</td>
<td>24-bit ΔΣA/D converter auto gain calibration function</td>
</tr>
<tr>
<td>R_I1E_PGA_DSAD_OffsetAdjustment</td>
<td>PGA offset adjustment execution function</td>
</tr>
<tr>
<td>R_I1E_PGA_DSAD_DisconnCheck</td>
<td>Disconnection detection function</td>
</tr>
<tr>
<td>R_I1E_AFE_Stop</td>
<td>RL78/I1E AFE stop processing function</td>
</tr>
<tr>
<td>R_I1E_AFE_ReStart</td>
<td>RL78/I1E AFE restart processing function</td>
</tr>
<tr>
<td>R_I1E_CAMP_Calibration</td>
<td>Configuration amplifier trimming function</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_I1E_FlashCheck</td>
<td>Data flash memory storage data check function</td>
</tr>
<tr>
<td>R_I1E_FlashRewrite</td>
<td>Data flash memory storage data rewrite function</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_I1E_UartSend</td>
<td>PC transmission processing setting function (ASCII): limit of up to 256 bytes per transmission</td>
</tr>
<tr>
<td>R_I1E_UartSendBinary</td>
<td>PC transmission processing setting function (Binary): limit of up to 256 bytes per transmission</td>
</tr>
<tr>
<td>R_I1E_UartReceive</td>
<td>PC receive processing (DTC0): limit of up to 256 bytes per reception</td>
</tr>
</tbody>
</table>

### 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>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_TAU_FreeRunTimerInit</td>
<td>Free running timer setting initialization function</td>
</tr>
<tr>
<td>R_TAU_FreeRunTimerStop</td>
<td>Free running timer stop function</td>
</tr>
<tr>
<td>R_GetTickCount</td>
<td>Get free running timer tick count function</td>
</tr>
<tr>
<td>R_CmpTickCount</td>
<td>Free running timer count compare function</td>
</tr>
</tbody>
</table>

### 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>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_KEY_Scan</td>
<td>Key scan processing</td>
</tr>
<tr>
<td>R_KEY_Initialize</td>
<td>Key information initialization</td>
</tr>
<tr>
<td>R_KEY_WaitOneClick</td>
<td>Get key information (specified key only)</td>
</tr>
<tr>
<td>R_KEY_GetAll</td>
<td>Get key information (all keys)</td>
</tr>
<tr>
<td>R_KEY_WaitOneClick</td>
<td>Specified key click wait</td>
</tr>
</tbody>
</table>
4. API Definitions

This section describes the definitions used in the Application Program Interface 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>
<thead>
<tr>
<th>Table 4-1 Type Definitions and Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Definition</strong></td>
</tr>
<tr>
<td>int8_t</td>
</tr>
<tr>
<td>uint8_t</td>
</tr>
<tr>
<td>int16_t</td>
</tr>
<tr>
<td>uint16_t</td>
</tr>
<tr>
<td>int32_t</td>
</tr>
<tr>
<td>uint32_t</td>
</tr>
<tr>
<td>MD_STATUS</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Table 4-2 MD_STATUS Type Return Values Used in Sample Code</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type Name</strong></td>
</tr>
<tr>
<td>MD_STATUS</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Table 4-3 Macro Declarations for User Environment-dependent Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Macro Declaration</strong></td>
</tr>
<tr>
<td>D_BULK_NUM</td>
</tr>
<tr>
<td>D_BULK_COMMAND_NUM</td>
</tr>
<tr>
<td>D_STREAMHEADER</td>
</tr>
</tbody>
</table>

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

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

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

<table>
<thead>
<tr>
<th>Macro Declaration</th>
<th>Default Setting Value</th>
<th>Input Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEF_KEY_ACTIVE</td>
<td>DEF_KEY_ACTIVE_LOW,</td>
<td>1U, uint8_t</td>
<td>SW active level</td>
</tr>
<tr>
<td>KEY_SCAN_NORM</td>
<td>10U</td>
<td></td>
<td>Single push determination time (10ms)</td>
</tr>
<tr>
<td>KEY_SCAN_LONG</td>
<td>100U</td>
<td>uint8_t</td>
<td>Long push determination time (10ms)</td>
</tr>
<tr>
<td>KEY_SCAN_DEAD</td>
<td>5U</td>
<td>uint8_t</td>
<td>Dead zone time after change in key state (10ms)</td>
</tr>
<tr>
<td>KEY_SCAN_NOT</td>
<td>5U</td>
<td>uint8_t</td>
<td>Non-active time (10ms)</td>
</tr>
<tr>
<td>KEY_SCAN_DOUBLE</td>
<td>50U</td>
<td>uint8_t</td>
<td>Double click determination enable time (10ms)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Macro Declaration</th>
<th>Default Setting Value</th>
<th>Input Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D_DEBUG_LED_USE</td>
<td>0U</td>
<td>0U, 1U</td>
<td>Debug LED usage setting</td>
</tr>
<tr>
<td>D_DEBUG_LED_PORT</td>
<td>P1.5</td>
<td>(output port) Note 1</td>
<td>Debug LED port setting</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Macro Declaration</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D_BULK_SPLIT_NUM</td>
<td>D_BULK_NUM / _DSAD_VALUE_BUF_SIZE</td>
<td>BULK division number</td>
</tr>
<tr>
<td>D_BULK_SPLIT_MOD</td>
<td>D_BULK_NUM % _DSAD_VALUE_BUF_SIZE</td>
<td>BULK division remainder</td>
</tr>
<tr>
<td>D_NO_ERROR</td>
<td>0x00U</td>
<td>No error</td>
</tr>
<tr>
<td>D_OVER_FLOW_ERROR</td>
<td>0x02U</td>
<td>Overflow error</td>
</tr>
<tr>
<td>D_DISCONNECTION_ERROR</td>
<td>0x80U</td>
<td>Disconnect error</td>
</tr>
</tbody>
</table>
| D_ON_OFF_COMMAND  | @0
| | | Receive command size for PC communication |

(b) `r_rl78_i1e_common.h`

<table>
<thead>
<tr>
<th>Macro Declaration</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D_MCU_VOLTAGE_MODE</td>
<td>0U</td>
<td>Memory voltage mode (full-speed mode) (fixed)</td>
</tr>
<tr>
<td>D_PGA_DSAD_MPX0</td>
<td>0x00U</td>
<td>Input multiplexer 0</td>
</tr>
<tr>
<td>D_PGA_DSAD_MPX1</td>
<td>0x01U</td>
<td>Input multiplexer 1</td>
</tr>
<tr>
<td>D_PGA_DSAD_MPX2</td>
<td>0x02U</td>
<td>Input multiplexer 2</td>
</tr>
<tr>
<td>D_PGA_DSAD_MPX3</td>
<td>0x03U</td>
<td>Input multiplexer 3</td>
</tr>
<tr>
<td>D_DSAD_AUTO_GAIN_TRIGGER_COUNT</td>
<td>D_DSAD_AUTO_GAIN_TRIGGER_SEC * D_MCU_CLOCK_MHZ * 1000000</td>
<td>PGA auto-gain adjustment timing Counter value</td>
</tr>
<tr>
<td>D_GAIN_ERROR_REFERENCE_LSB</td>
<td>(int32_t)((0x800000L * D_GAIN_ERROR_REFERENCE_mV / 0.8 / 1000L)</td>
<td>PGA gain error measurement reference voltage (LSB)</td>
</tr>
<tr>
<td>D_DISCONNECTION_THRESHOLD_LSB</td>
<td>(int32_t)(0x800000L * D_DISCONNECTION_THRESHOLD_mV / 0.8 / 1000L)</td>
<td>Disconnection determination threshold voltage (LSB)</td>
</tr>
<tr>
<td>D_UART_SEND_BUFFER_NUMBER</td>
<td>2U</td>
<td>Number of transmission buffers for PC communication</td>
</tr>
<tr>
<td>D_FLASH_SUCCESS</td>
<td>0U</td>
<td>Flash memory processing success</td>
</tr>
<tr>
<td>D_FLASH_MATCH</td>
<td>0U</td>
<td>Flash memory and RAM match</td>
</tr>
<tr>
<td>D_FLASH_MISMATCH</td>
<td>1U</td>
<td>Flash memory mismatch</td>
</tr>
<tr>
<td>D_FLASH_INVALID</td>
<td>2U</td>
<td>Flash memory data invalid</td>
</tr>
<tr>
<td>D_FLASH_FAILURE</td>
<td>6U</td>
<td>Flash memory processing error</td>
</tr>
<tr>
<td>D_FLASH_DATA_STRUCT_SIZE</td>
<td>sizeof(str_flash_data_t) / sizeof(uint8_t)</td>
<td>Total structure size for flash storage variable (byte)</td>
</tr>
<tr>
<td>D_FLASH_DATA_CHECKSUM_SIZE</td>
<td>2U</td>
<td>Checksum size (byte)</td>
</tr>
<tr>
<td>D_CAMP_CH_NUMBER</td>
<td>3U</td>
<td>Number of configurable amp channels</td>
</tr>
</tbody>
</table>
### Table 4-9  Macro Declarations

<table>
<thead>
<tr>
<th>Macro Declaration</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D_DSAD_VALEU_MAX</td>
<td>(1L&lt;&lt;23) - 1</td>
<td>$\Delta$ΣA/D CONVERTER input voltage Max setting: $+800 \text{ mV/(G}_{\text{TOTAL}})$</td>
</tr>
<tr>
<td>D_DSAD_VALEU_MIN</td>
<td>-1*(1L&lt;&lt;23)</td>
<td>$\Delta$ΣA/D CONVERTER input voltage Min setting: $-800 \text{ mV/(G}_{\text{TOTAL}})$</td>
</tr>
<tr>
<td>D_DSAD_AUTO_GAIN_MAX_POS</td>
<td>0U</td>
<td>PGA auto-gain adjustment array index Max value of index</td>
</tr>
<tr>
<td>D_DSAD_AUTO_GAIN_MIN_POS</td>
<td>1U</td>
<td>PGA auto-gain calibration array index Min value of index</td>
</tr>
</tbody>
</table>

### Table 4-10  Macro Declarations

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

### Table 4-11  Macro Declarations

<table>
<thead>
<tr>
<th>Macro Declaration</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEF_KEY_ACTIVE_LOW</td>
<td>0U</td>
<td>Port active level = LOW</td>
</tr>
<tr>
<td>DEF_KEY_ACTIVE_HIGH</td>
<td>1U</td>
<td>Port active level = HIGH</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Type Name</th>
<th>Default Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>e_key_t</td>
<td>SW_MODE_CHANGE</td>
<td>This is the enumeration 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, SW_GAIN_DOWN, SW_OFFSET_UP, SW_OFFSET_DOWN.</td>
</tr>
<tr>
<td>SW_TYPE_MAX</td>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Type Name</th>
<th>Macro Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>e_bulk_control_t</td>
<td>E_BULKDATA_BULKSTART</td>
<td>BULKSTART format data generation</td>
</tr>
<tr>
<td></td>
<td>E_BULKDATA_BULK</td>
<td>BULK format data generation</td>
</tr>
<tr>
<td></td>
<td>E_BULKDATA_BULKEND</td>
<td>BULKEND format data generation</td>
</tr>
</tbody>
</table>

Table 4-14 Communication Command Generation Control Enumeration

<table>
<thead>
<tr>
<th>Type Name</th>
<th>Macro Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>e_communication_data_t</td>
<td>E_STREAMHEADER</td>
<td>STREAMHEADER format data generation</td>
</tr>
<tr>
<td></td>
<td>E_STREAM</td>
<td>STREAM format data generation</td>
</tr>
<tr>
<td></td>
<td>E_BULK</td>
<td>BULK format data generation</td>
</tr>
<tr>
<td></td>
<td>E_BINARY</td>
<td>BINARY format data generation</td>
</tr>
</tbody>
</table>
Table 4-15  Communication Command Judgement Enumeration

<table>
<thead>
<tr>
<th>Type Name</th>
<th>Macro Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>e_communication_data_t</td>
<td>E_ON_OFF_COMMAND</td>
<td>ON/OFF command</td>
</tr>
<tr>
<td></td>
<td>E_COMMAND_NONE</td>
<td>Determines no command is present</td>
</tr>
</tbody>
</table>

Table 4-16  Measurement Control Enumeration

<table>
<thead>
<tr>
<th>Type Name</th>
<th>Macro Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>e_measurement_control_t</td>
<td>E_TRANSMISSION_START</td>
<td>Serial transmission start</td>
</tr>
<tr>
<td></td>
<td>E_DATA_GENERATION</td>
<td>Generates measured value transmission data</td>
</tr>
<tr>
<td></td>
<td>E_ERROR_CHECK</td>
<td>Error check</td>
</tr>
<tr>
<td></td>
<td>E_WAIT</td>
<td>No operation</td>
</tr>
</tbody>
</table>

Table 4-17  MCU State Determination Enumeration

<table>
<thead>
<tr>
<th>Type Name</th>
<th>Macro Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>e_mcu_state_t</td>
<td>E_MCU_HALT</td>
<td>HALT mode</td>
</tr>
<tr>
<td></td>
<td>E_MCU_RUN</td>
<td>Now running</td>
</tr>
</tbody>
</table>

(b)  \texttt{r_rl78_i1e_common.h}

Table 4-18  Input Multiplexer Specification Enumeration

<table>
<thead>
<tr>
<th>Type Name</th>
<th>Macro Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>e_pga_dsad_mpx_t</td>
<td>E_PGA_DSAD_MPX0</td>
<td>0x00U</td>
<td>Input multiplexer 0</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_MPX1</td>
<td>0x01U</td>
<td>Input multiplexer 1</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_MPX2</td>
<td>0x02U</td>
<td>Input multiplexer 2</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_MPX3</td>
<td>0x03U</td>
<td>Input multiplexer 3</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_MPX4</td>
<td>0x04U</td>
<td>Input multiplexer 4 -&gt; internal temperature sensor (fixed)</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_MPX_MAX</td>
<td>0x05U</td>
<td>Input multiplexer max judgement</td>
</tr>
</tbody>
</table>

Table 4-19  Input Mode Specification Enumeration

<table>
<thead>
<tr>
<th>Type Name</th>
<th>Macro Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>e_pga_dsad_input_mode_t</td>
<td>E_PGA_DSAD_DIFF_INPUT</td>
<td>0x00U</td>
<td>Differential input mode</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_SINGLE_INPUT</td>
<td>0x01U</td>
<td>Single-ended input mode</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Type Name</th>
<th>Macro Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>e_pga_dsad_onoff_t</td>
<td>E_PGA_DSAD_OFF</td>
<td>0x01U</td>
<td>A/D conversion OFF</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_ON</td>
<td>0x00U</td>
<td>A/D conversion ON</td>
</tr>
</tbody>
</table>
### Table 4-21  PGA Gain Setting Specification Enumeration

<table>
<thead>
<tr>
<th>Type Name</th>
<th>Macro Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>E_PGA_DSAD_GAIN_1_1_1</td>
<td>0x00U</td>
<td>( G_{SET1} = x1, G_{SET2} = x1, G_{TOTAL} = x1 )</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_GAIN_2_1_2</td>
<td>0x04U</td>
<td>( G_{SET1} = x2, G_{SET2} = x1, G_{TOTAL} = x2 )</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_GAIN_3_1_3</td>
<td>0x08U</td>
<td>( G_{SET1} = x3, G_{SET2} = x1, G_{TOTAL} = x3 )</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_GAIN_4_1_4</td>
<td>0x0CU</td>
<td>( G_{SET1} = x4, G_{SET2} = x1, G_{TOTAL} = x4 )</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_GAIN_8_1_8</td>
<td>0x10U</td>
<td>( G_{SET1} = x8, G_{SET2} = x1, G_{TOTAL} = x8 )</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_GAIN_1_2_2</td>
<td>0x01U</td>
<td>( G_{SET1} = x1, G_{SET2} = x2, G_{TOTAL} = x2 )</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_GAIN_2_2_4</td>
<td>0x05U</td>
<td>( G_{SET1} = x2, G_{SET2} = x2, G_{TOTAL} = x4 )</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_GAIN_3_2_6</td>
<td>0x09U</td>
<td>( G_{SET1} = x3, G_{SET2} = x2, G_{TOTAL} = x6 )</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_GAIN_4_2_8</td>
<td>0x0DU</td>
<td>( G_{SET1} = x4, G_{SET2} = x2, G_{TOTAL} = x8 )</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_GAIN_8_2_16</td>
<td>0x11U</td>
<td>( G_{SET1} = x8, G_{SET2} = x2, G_{TOTAL} = x16 )</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_GAIN_1_4_4</td>
<td>0x02U</td>
<td>( G_{SET1} = x1, G_{SET2} = x4, G_{TOTAL} = x4 )</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_GAIN_2_4_8</td>
<td>0x06U</td>
<td>( G_{SET1} = x2, G_{SET2} = x4, G_{TOTAL} = x8 )</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_GAIN_3_4_12</td>
<td>0x0AU</td>
<td>( G_{SET1} = x3, G_{SET2} = x4, G_{TOTAL} = x12 )</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_GAIN_4_4_16</td>
<td>0x0EU</td>
<td>( G_{SET1} = x4, G_{SET2} = x4, G_{TOTAL} = x16 )</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_GAIN_8_4_32</td>
<td>0x12U</td>
<td>( G_{SET1} = x8, G_{SET2} = x4, G_{TOTAL} = x32 )</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_GAIN_1_8_8</td>
<td>0x03U</td>
<td>( G_{SET1} = x1, G_{SET2} = x8, G_{TOTAL} = x8 )</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_GAIN_2_8_16</td>
<td>0x07U</td>
<td>( G_{SET1} = x2, G_{SET2} = x8, G_{TOTAL} = x16 )</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_GAIN_3_8_24</td>
<td>0x0BU</td>
<td>( G_{SET1} = x3, G_{SET2} = x8, G_{TOTAL} = x24 )</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_GAIN_4_8_32</td>
<td>0x0FU</td>
<td>( G_{SET1} = x4, G_{SET2} = x8, G_{TOTAL} = x32 )</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_GAIN_8_8_64</td>
<td>0x13U</td>
<td>( G_{SET1} = x8, G_{SET2} = x8, G_{TOTAL} = x64 )</td>
</tr>
</tbody>
</table>
## Table 4-22  Offset Voltage Setting Specification Enumeration

<table>
<thead>
<tr>
<th>Type Name</th>
<th>Macro Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_PGA_DSAD_OFFSET_164p06</td>
<td>0xFU</td>
<td>164.06/G_SET1 [mV]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OFFSET_153p13</td>
<td>0xEU</td>
<td>153.13/G_SET1 [mV]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OFFSET_142p19</td>
<td>0xDU</td>
<td>142.19/G_SET1 [mV]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OFFSET_131p25</td>
<td>0xCU</td>
<td>131.25/G_SET1 [mV]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OFFSET_120p31</td>
<td>0xBU</td>
<td>120.31/G_SET1 [mV]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OFFSET_98p44</td>
<td>0x9U</td>
<td>98.44/G_SET1 [mV]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OFFSET_87p50</td>
<td>0x8U</td>
<td>87.50/G_SET1 [mV]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OFFSET_76p56</td>
<td>0x7U</td>
<td>76.56/G_SET1 [mV]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OFFSET_65p63</td>
<td>0x6U</td>
<td>65.63/G_SET1 [mV]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OFFSET_54p69</td>
<td>0x5U</td>
<td>54.69/G_SET1 [mV]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OFFSET_43p75</td>
<td>0x4U</td>
<td>43.75/G_SET1 [mV]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OFFSET_32p81</td>
<td>0x3U</td>
<td>32.81/G_SET1 [mV]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OFFSET_21p88</td>
<td>0x2U</td>
<td>21.88/G_SET1 [mV]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OFFSET_10p94</td>
<td>0x1U</td>
<td>10.94/G_SET1 [mV]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OFFSET_0p00</td>
<td>0x0U</td>
<td>0.00/G_SET1 [mV]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OFFSET_M10p94</td>
<td>0x0FU</td>
<td>-10.94/G_SET1 [mV]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OFFSET_M21p88</td>
<td>0x0EU</td>
<td>-21.88/G_SET1 [mV]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OFFSET_M32p81</td>
<td>0x0DU</td>
<td>-32.81/G_SET1 [mV]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OFFSET_M43p75</td>
<td>0x0CU</td>
<td>-43.75/G_SET1 [mV]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OFFSET_M54p69</td>
<td>0x0BU</td>
<td>-54.69/G_SET1 [mV]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OFFSET_M65p63</td>
<td>0x0AU</td>
<td>-65.63/G_SET1 [mV]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OFFSET_M76p56</td>
<td>0x09U</td>
<td>-76.56/G_SET1 [mV]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OFFSET_M87p50</td>
<td>0x08U</td>
<td>-87.50/G_SET1 [mV]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OFFSET_M98p44</td>
<td>0x07U</td>
<td>-98.44/G_SET1 [mV]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OFFSET_M109p38</td>
<td>0x06U</td>
<td>-109.38/G_SET1 [mV]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OFFSET_M120p31</td>
<td>0x05U</td>
<td>-120.31/G_SET1 [mV]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OFFSET_M131p25</td>
<td>0x04U</td>
<td>-131.25/G_SET1 [mV]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OFFSET_M142p19</td>
<td>0x03U</td>
<td>-142.19/G_SET1 [mV]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OFFSET_M153p13</td>
<td>0x02U</td>
<td>-153.13/G_SET1 [mV]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OFFSET_M164p06</td>
<td>0x01U</td>
<td>-164.06/G_SET1 [mV]</td>
<td></td>
</tr>
</tbody>
</table>

## Table 4-23  OSR Setting Specification Enumeration

<table>
<thead>
<tr>
<th>Type Name</th>
<th>Macro Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_PGA_DSAD_OSR_64</td>
<td>0x00U</td>
<td>15625.000 [sps]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OSR_128</td>
<td>0x01U</td>
<td>7812.500 [sps]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OSR_256</td>
<td>0x02U</td>
<td>3906.250 [sps]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OSR_512</td>
<td>0x03U</td>
<td>1953.125 [sps]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OSR_1024</td>
<td>0x04U</td>
<td>976.563 [sps]</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_OSR_2048</td>
<td>0x05U</td>
<td>488.281 [sps]</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Type Name</th>
<th>Macro Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_PGA_DSAD_COUNT_CALCULATION</td>
<td>0x00U</td>
<td>Specify 1 to 8,032 times by using the value set in the PGAxCTL2 register (default)</td>
<td></td>
</tr>
<tr>
<td>E_PGA_DSAD_COUNT_LINEAR</td>
<td>0x01U</td>
<td>Specify 1 to 255 times linearly by using the value set in the PGAxCTL2 register</td>
<td></td>
</tr>
</tbody>
</table>
### Table 4-25  Averaging Procedure Operation Selection Enumeration

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

### Table 4-26  Averaging Data Selection Enumeration

<table>
<thead>
<tr>
<th>Type Name</th>
<th>Macro Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>e_pga_dsad_average_number_t</td>
<td>E_PGA_DSAD_AVERAGE_8</td>
<td>0x00U</td>
<td>Number of data for averaging: 8</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_AVERAGE_16</td>
<td>0x01U</td>
<td>Number of data for averaging: 16</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_AVERAGE_32</td>
<td>0x02U</td>
<td>Number of data for averaging: 32</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_AVERAGE_64</td>
<td>0x03U</td>
<td>Number of data for averaging: 64</td>
</tr>
</tbody>
</table>

### Table 4-27  ∆ΣA/D Conversion (AUTOSCAN) Start/Stop Specification Enumeration

<table>
<thead>
<tr>
<th>Type Name</th>
<th>Macro Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>e_pga_dsad_start_stop_t</td>
<td>E_PGA_DSAD_STOP</td>
<td>0x00U</td>
<td>Stop</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_START</td>
<td>0x01U</td>
<td>Start</td>
</tr>
</tbody>
</table>

### Table 4-28  Auto Scan Mode Specification Enumeration

<table>
<thead>
<tr>
<th>Type Name</th>
<th>Macro Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>e_pga_dsad_autoscan_mode_t</td>
<td>E_PGA_DSAD_CONTINUOUS</td>
<td>0x00U</td>
<td>Continuous scan mode</td>
</tr>
<tr>
<td></td>
<td>E_PGA_DSAD_SINGLE</td>
<td>0x01U</td>
<td>Single scan mode</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Type Name</th>
<th>Macro Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>e_auto_gain_return_t</td>
<td>E_AUTO_GAIN_NO_ADJUSTMENT</td>
<td>0x00U</td>
<td>No PGA auto-gain adjustment</td>
</tr>
<tr>
<td></td>
<td>E_AUTO_GAIN_ADJUSTMENT</td>
<td>0x01U</td>
<td>Execute PGA auto-gain adjustment</td>
</tr>
</tbody>
</table>

### Table 4-30  Disconnection Detection Processing Return Value Specification Enumeration

<table>
<thead>
<tr>
<th>Type Name</th>
<th>Macro Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>e_disconnection_return_t</td>
<td>E_DISCONNECTION_NO_DETECT</td>
<td>0x00U</td>
<td>Disconnection not detected</td>
</tr>
<tr>
<td></td>
<td>E_DISCONNECTION_DETECT</td>
<td>0x01U</td>
<td>Disconnection detected</td>
</tr>
<tr>
<td>Type Name</td>
<td>Macro Name</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------</td>
<td>--------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>e_key_status_t</td>
<td>E_KEY_OFF</td>
<td>Key in OFF state</td>
<td></td>
</tr>
<tr>
<td></td>
<td>E_KEY_OFF_TO_ON</td>
<td>Transition from key OFF to key normal push state</td>
<td></td>
</tr>
<tr>
<td></td>
<td>E_KEY_ON_NORM</td>
<td>Key in normal push state</td>
<td></td>
</tr>
<tr>
<td></td>
<td>E_KEY_NORM_TO_LONG</td>
<td>Transition from key in normal push state to key in long push state</td>
<td></td>
</tr>
<tr>
<td></td>
<td>E_KEY_ON_LONG</td>
<td>Key in long push state</td>
<td></td>
</tr>
<tr>
<td></td>
<td>E_KEY_LONG_TO_LONG</td>
<td>Transition from key in long push state to key in long push state</td>
<td></td>
</tr>
<tr>
<td></td>
<td>E_KEY_DOUBLE_CLICK</td>
<td>Key in double click state</td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Member Variable</th>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>uint32_t</td>
<td>count</td>
<td>∆ΣA/D conversion counter</td>
</tr>
<tr>
<td>dsad_value</td>
<td>str_pga_dsad_value_t</td>
<td>dsad_value</td>
<td>∆ΣA/D converted value structure variable</td>
</tr>
<tr>
<td>mcu_state</td>
<td>e_mcu_state_t</td>
<td>mcu_state</td>
<td>MCU operating state</td>
</tr>
<tr>
<td>error_state</td>
<td>uint8_t</td>
<td>error_state</td>
<td>Error state</td>
</tr>
</tbody>
</table>

Table 4-33 Structure for Variables Storing BULK Transfer Data

<table>
<thead>
<tr>
<th>Member Variable</th>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>send_count</td>
<td>uint16_t</td>
<td>send_count</td>
<td>Bulk transfer send counter</td>
</tr>
<tr>
<td>control</td>
<td>e_bulk_control_t</td>
<td>control</td>
<td>Bulk transfer control</td>
</tr>
<tr>
<td>buf[D_BULK_BUFFER_SIZE]</td>
<td>int32_t</td>
<td>BULK transfer data buffer</td>
<td></td>
</tr>
<tr>
<td>str_count</td>
<td>uint16_t</td>
<td>str_count</td>
<td>BULK transfer data buffer counter</td>
</tr>
<tr>
<td>split_count</td>
<td>uint16_t</td>
<td>split_count</td>
<td>Bulk transfer split send counter</td>
</tr>
</tbody>
</table>

(b) r_rl78_i1e_common.h

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

<table>
<thead>
<tr>
<th>Member Variable</th>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dsad_onoff</td>
<td>e_pga_dsad_onoff_t</td>
<td>dsad_onoff</td>
<td>A/D conversion enabled/disabled flag</td>
</tr>
<tr>
<td>dsad_input_mode</td>
<td>e_pga_dsad_input_mode_t</td>
<td>dsad_input_mode</td>
<td>Input mode</td>
</tr>
<tr>
<td>dsad_offset</td>
<td>e_pga_dsad_offset_t</td>
<td>dsad_offset</td>
<td>Offset register setting value</td>
</tr>
<tr>
<td>dsad_osr</td>
<td>e_pga_dsad_osr_t</td>
<td>dsad_osr</td>
<td>Oversampling rate</td>
</tr>
<tr>
<td>dsad_gain</td>
<td>e_pga_dsad_gain_t</td>
<td>dsad_gain</td>
<td>Gain</td>
</tr>
<tr>
<td>auto-scan number</td>
<td>uint8_t</td>
<td>dsad_count</td>
<td>Auto-scan number counter</td>
</tr>
<tr>
<td>auto-scan number</td>
<td>e_pga_dsad_count_mode_t</td>
<td>dsad_count_mode</td>
<td>A/D conversion count calculation method specification</td>
</tr>
<tr>
<td>auto-scan number</td>
<td>e_pga_dsad_average_operation_t</td>
<td>dsad_average_operation</td>
<td>Selection of averaging process operation</td>
</tr>
<tr>
<td>auto-scan number</td>
<td>e_pga_dsad_average_number_t</td>
<td>dsad_average_number</td>
<td>Selection of number of data for averaging</td>
</tr>
<tr>
<td>Structure Name</td>
<td>str_pga_dsad_value_t</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>----------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Outline</td>
<td>Definition for variables storing each type of information such as ΔΣA/D converted value, gain, offset, etc.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Member Variable</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Name</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>uint32_t</td>
<td>count</td>
<td>Measured number counter value</td>
<td></td>
</tr>
<tr>
<td>uni_long_t</td>
<td>adc_value</td>
<td>A/D converted value (24 bits, right-aligned)</td>
<td></td>
</tr>
<tr>
<td>uint8_t</td>
<td>ch</td>
<td>Number of channels</td>
<td></td>
</tr>
<tr>
<td>uint8_t</td>
<td>overflow</td>
<td>Overflow flag</td>
<td></td>
</tr>
<tr>
<td>uint8_t</td>
<td>gain_set_1</td>
<td>Preamplifier gain multiplier</td>
<td></td>
</tr>
<tr>
<td>uint8_t</td>
<td>gain_set_2</td>
<td>Post amplifier gain multiplier</td>
<td></td>
</tr>
<tr>
<td>uint8_t</td>
<td>gain_total</td>
<td>Total gain multiplier</td>
<td></td>
</tr>
<tr>
<td>e_pga_dsad_offset_t</td>
<td>offset_reg</td>
<td>Offset register setting value</td>
<td></td>
</tr>
<tr>
<td>int32_t</td>
<td>adc_correct</td>
<td>A/D conversion correction (24 bits, right-aligned)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Structure Name</th>
<th>str_flash_data_t</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outline</td>
<td>Definition for flash storage variables</td>
</tr>
<tr>
<td>Member Variable</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Name</td>
</tr>
<tr>
<td>int32_t</td>
<td>gain_real[5U]</td>
</tr>
<tr>
<td>int32_t</td>
<td>offset_factor[5U][32U]</td>
</tr>
<tr>
<td>uint16_t</td>
<td>checksum</td>
</tr>
</tbody>
</table>

(c) r_keysan.h

<table>
<thead>
<tr>
<th>Structure Name</th>
<th>key_setting_t</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outline</td>
<td>Definition of key information setting variables</td>
</tr>
<tr>
<td>Member Variable</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Name</td>
</tr>
<tr>
<td>uint8_t *</td>
<td>p_port_addr</td>
</tr>
<tr>
<td>uint8_t</td>
<td>port_bit_num</td>
</tr>
<tr>
<td>uint8_t</td>
<td>multiple_group</td>
</tr>
</tbody>
</table>
4.4.4 Unions

This section describes the union declarations defined in the API.

(a) `r_rl78_i1e_common.h`

<table>
<thead>
<tr>
<th>Structure Name</th>
<th>uni_long_t</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outline</td>
<td>Definition of union for reading signed/unsigned data</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Member Variable</th>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int32_t</td>
<td>32</td>
<td>LONG</td>
<td>signed long</td>
</tr>
<tr>
<td>uint32_t</td>
<td>32</td>
<td>uLONG</td>
<td>unsigned long</td>
</tr>
</tbody>
</table>

| Table 4-38 Signed Long/Unsigned Long Read Unions |

<table>
<thead>
<tr>
<th>Structure Name</th>
<th>uni_pga_dsad_conversion_result_t</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outline</td>
<td>Definition of union for reading ΔΣA/D conversion result register C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Member Variable</th>
<th>Type</th>
<th>No. of bits</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8_t</td>
<td>8</td>
<td>BYTE</td>
<td>8-bit access</td>
<td></td>
</tr>
<tr>
<td>BIT</td>
<td>4</td>
<td>-</td>
<td>Not used</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>overflow</td>
<td>Overflow Flag</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ch</td>
<td>Ch number</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Structure Name</th>
<th>uni_pga_dsad_value_t</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outline</td>
<td>Definition of union for reading ΔΣA/D conversion results registers</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Member Variable</th>
<th>Type</th>
<th>No. of bits</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int32_t</td>
<td>32</td>
<td>LONG</td>
<td>32-bit access</td>
<td></td>
</tr>
<tr>
<td>uint16_t</td>
<td>32</td>
<td>WORD[2]</td>
<td>16-bit access</td>
<td></td>
</tr>
<tr>
<td>BYTE</td>
<td>uni_pga_dsad_conversion_result_t</td>
<td>8</td>
<td>dsad_c</td>
<td>ΔΣA/D conversion results register C</td>
</tr>
<tr>
<td>uint8_t</td>
<td>8</td>
<td>dsad_l</td>
<td>ΔΣA/D conversion results register L</td>
<td></td>
</tr>
<tr>
<td>uint8_t</td>
<td>8</td>
<td>dsad_m</td>
<td>ΔΣA/D conversion results register M</td>
<td></td>
</tr>
<tr>
<td>uint8_t</td>
<td>8</td>
<td>dsad_h</td>
<td>ΔΣA/D conversion results register H</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Structure Name</th>
<th>uni_pga_dsad_gain_t</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outline</td>
<td>Definition of union for reading PGAxCTL0 register gain setting</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Member Variable</th>
<th>Type</th>
<th>No. of bits</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8_t</td>
<td>8</td>
<td>BYTE</td>
<td>8-bit access</td>
<td></td>
</tr>
<tr>
<td>GAIN_BIT</td>
<td>2</td>
<td>gain_set_2</td>
<td>G_SET2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>gain_set_1</td>
<td>G_SET1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>Not used</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 4-42 Register PGAxCTL0 Control Union

<table>
<thead>
<tr>
<th>Structure Name</th>
<th>uni_reg_pgaxctl0_t</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outline</td>
<td>Definition of union for register PGAxCTL0 control.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Member Variable</th>
<th>Type</th>
<th>No. of bits</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8_t</td>
<td>8</td>
<td>BYTE</td>
<td>8-bit access</td>
<td></td>
</tr>
<tr>
<td>PGAxCTL0_BIT</td>
<td>5</td>
<td>gain</td>
<td>GAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>osr</td>
<td>OSR</td>
<td></td>
</tr>
</tbody>
</table>

### Table 4-43 Register PGAxCTL1 Control Union

<table>
<thead>
<tr>
<th>Structure Name</th>
<th>uni_reg_pgaxctl1_t</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outline</td>
<td>Definition of union for register PGAxCTL1 control.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Member Variable</th>
<th>Type</th>
<th>No. of bits</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8_t</td>
<td>8</td>
<td>BYTE</td>
<td>8-bit access</td>
<td></td>
</tr>
<tr>
<td>PGAxCTL1_BIT</td>
<td>5</td>
<td>offset</td>
<td>offset</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>-</td>
<td>Not used</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>pga3tsel</td>
<td>Configurable amplifier self-correction</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>input_mode</td>
<td>AINSEL</td>
<td></td>
</tr>
</tbody>
</table>

### Table 4-44 Register PGAxCTL2 Control Union

<table>
<thead>
<tr>
<th>Structure Name</th>
<th>uni_reg_pgaxctl2_t</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outline</td>
<td>Definition of union for register PGAxCTL2 control.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Member Variable</th>
<th>Type</th>
<th>No. of bits</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8_t</td>
<td>8</td>
<td>BYTE</td>
<td>8-bit access</td>
<td></td>
</tr>
<tr>
<td>PGAxCTL2_BIT</td>
<td>5</td>
<td>low</td>
<td>Offset</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>high</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

### Table 4-45 Register PGAxCTL3 Control Union

<table>
<thead>
<tr>
<th>Structure Name</th>
<th>uni_reg_pgaxctl3_t</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outline</td>
<td>Definition of union for register PGAxCTL3 control.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Member Variable</th>
<th>Type</th>
<th>No. of bits</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8_t</td>
<td>8</td>
<td>BYTE</td>
<td>8-bit access</td>
<td></td>
</tr>
<tr>
<td>PGAxCTL3_BIT</td>
<td>2</td>
<td>pgaxave0_1</td>
<td>PGAXAV[1:0] Selection of data for averaging</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>pgaxave3_2</td>
<td>PGAXAV[3:2] Selection of averaging processing operation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>-</td>
<td>Not used</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>pgaxctm</td>
<td>Selection of A/D conversion count specification mode</td>
<td></td>
</tr>
</tbody>
</table>

### Table 4-46 Register DSADCTL Control Union

<table>
<thead>
<tr>
<th>Structure Name</th>
<th>uni_reg_dsadctl_t</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outline</td>
<td>Definition of union for register DSADCTL control.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Member Variable</th>
<th>Type</th>
<th>No. of bits</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8_t</td>
<td>8</td>
<td>BYTE</td>
<td>8-bit access</td>
<td></td>
</tr>
<tr>
<td>DSADCTL_BIT</td>
<td>2</td>
<td>pgaxave0_1</td>
<td>PGAXAV[1:0] Selection of data for averaging</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>pgaxave3_2</td>
<td>PGAXAV[3:2] Selection of averaging processing operation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>-</td>
<td>Not used</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>pgaxctm</td>
<td>Selection of A/D conversion count specification mode</td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Structure Name</th>
<th>key_setting_t</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outline</td>
<td>Structure for global variables storing key setting information</td>
</tr>
<tr>
<td>Member Variable</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Name</td>
</tr>
<tr>
<td>uint8_t * p_port_addr</td>
<td>p_port_addr</td>
</tr>
<tr>
<td>uint8_t port_bit_num</td>
<td>port_bit_num</td>
</tr>
<tr>
<td>uint8_t multiple_group</td>
<td>multiple_group</td>
</tr>
</tbody>
</table>

(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

<table>
<thead>
<tr>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bulk_data_t</td>
<td>g_bulk_data</td>
<td>BULK transfer data storage variable</td>
</tr>
</tbody>
</table>

(b) r_rl78_i1e_common.c

Table 4-49  r_rl78_i1e_common.c Global Variables

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Type name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>g_flash_value</td>
<td>str_flash_data_t</td>
<td>Structure for data flash storage data</td>
</tr>
<tr>
<td>g_dsad_autoscan_mode</td>
<td>e_pga_dsad.autoscan_mode_t</td>
<td>Auto-scan mode setting</td>
</tr>
<tr>
<td>g_dsad_setting</td>
<td>str_pga_dsad_setting_t</td>
<td>Input multiplexer settings retentive variable</td>
</tr>
<tr>
<td>g_camp_trimming</td>
<td>uint8_t</td>
<td>Configurable amplifier trimming setting value storage variable</td>
</tr>
<tr>
<td>g_dsad_value</td>
<td>str_pga_dsad_value_t</td>
<td>DSAD converted value storage buffer</td>
</tr>
<tr>
<td>g_dsad_value_write_pos</td>
<td>uint16_t</td>
<td>DSAD converted value storage buffer write position</td>
</tr>
<tr>
<td>g_dsad_value_read_pos</td>
<td>uint16_t</td>
<td>DSAD converted value storage buffer read position</td>
</tr>
<tr>
<td>g_dsad_value_user_read_pos</td>
<td>uint16_t</td>
<td>DSAD converted value storage buffer user read position</td>
</tr>
<tr>
<td>g_dsad_measurement_count</td>
<td>uint32_t</td>
<td>DSAD conversion measurement counter</td>
</tr>
<tr>
<td>g_i1e_uart_send_buffer</td>
<td>int8_t</td>
<td>Transmit buffer for PC transmissions (DTC is used for transfers to UART)</td>
</tr>
<tr>
<td>g_buffer_number</td>
<td>uint8_t</td>
<td>Storage destination number of transmit buffer for PC transmissions</td>
</tr>
</tbody>
</table>
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 that inhibits use of RAM area used by data flash library

Definition that inhibits use of RAM area used by DTC table

Definition standard RAM
5. Analog Front-end APIs

5.1 Sample code variable initialization processing

```c
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 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 ∆ΣA/D converter.
- Get configurable amp trimming data.
- Initialize ∆ΣA/D converted value ring buffer.
- Initialize ∆ΣA/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.

```plaintext
R_I1E_Variable_Initialize

Get ∆ΣA/D converter setting

Get configurable amp trimming data

Initialize ∆ΣA/D converter ring buffer

Initialize ∆ΣA/D converter measurement number

Is flash storage variable actual gain setting value set?

Yes

Initialize ∆ΣA/D converter auto-gain adjustment variable

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 ΔΣΔ/D converted values.

```
R_I1E_RingBuffer_Initialize

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.

---

**Flowchart Description**

1. **ΔΣ/D converter now converting?**
   - **Yes**
     - Implement configurable amplifier calibration
     - Measure differential of programmable gain instrumentation amplifier
   - **No**
     - Stop ΔΣ/D converter conversion operation

2. **ΔΣ/D converter now converting?**
   - **Yes**
     - Overwrite data stored in data flash memory
   - **No**
     - Overwrite of data stored in data flash memory failed?
       - **Yes**
         - Endless debug loop
       - **No**
         - NOP

3. **NOP return**
5.4 24-bit $\Delta\Sigma$/A/D Converter/Specified Ch1-shot Differential Input A/D Conversion Function

MD_STATUS R_I1E_PGA_DSAD_GetResult_1Shot
(e_pga_dsad_mpx_t dsad_mpx, int32_t *p_dsad_value)

Description
24-bit $\Delta\Sigma$/A/D converter/specifed Ch 1-shot differential input A/D conversion function

Argument
- dsad_mpx: 24-bit $\Delta\Sigma$/A/D converter input multiplexer number
- *p_dsad_value: 24-bit $\Delta\Sigma$/A/D converter 24-bit A/D value storage pointer

Global Variables
None

SFR
- DSADIF: $\Delta\Sigma$/A/D converter conversion complete interrupt request flag
- DSADSIF: $\Delta\Sigma$/A/D converter scan complete interrupt request flag
- DSADMK: $\Delta\Sigma$/A/D converter conversion complete interrupt mask flag
- DSADSMK: $\Delta\Sigma$/A/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 with $\Delta\Sigma$/A/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.

---

![Flowchart](flowchart.png)
### 5.5 24-bit ΔΣA/D Converter Gain Setting Function

```c
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 the ΔΣA/D converter input multiplexer.

### 5.6 24-bit ΔΣA/D Converter Offset Setting Function

```c
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

```c
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

```c
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

**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

```c
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

**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

```c
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

**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 ∆ΣA/D Converter Converted Value Get Function

```c
int16_t R_I1E_PGA_DSAD_GetValue(str_pga_dsad_value_t *p_value)
```

**Description**
24-bit ∆ΣA/D converter converted value get function

**Argument**
*p_value:
24-bit ∆ΣA/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 ∆ΣA/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.

---

**Flowchart**

1. **R_I1E_PGA_DSAD_GetValue**
2. **ret = -1**
3. **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:
1. Initialize PGA auto-gain adjustment variable.
2. Timer setting for ΔΣ A/D converter auto-gain adjustment
3. count = 0U
4. If count < [E_PGA_DSAD_MFX_MAX - 1U] then:
   a. Max value initialization/Min value initialization
   b. count++
   c. Set ΔΣ A/D converter ring buffer read position
   d. Enable interrupt
5. Disable interrupt
6. 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 judged unsuitable based on the updated value, the PGA gain is auto-adjusted.

```
R_I1E_PGA_DSAD_AutoGainBufCheck)

ret = E_AUTO_GAIN_NO_ADJUSTMENT

No

Yes

Updated measurement data available?

No

Yes

Other than temperature sensor channel?

No

Yes

Update previous stage gain value

No

Overflow generated?

Yes

No

Decrement ΔΣA/D converter gain by 1 (Store results in ret)

Yes

No

PGA gain with adjustment?

Yes

No

Initialize ΔΣA/D converter auto-gain variable

return (ret)
```

Max value < obtained value?

Yes

No

Update max value

Min value > obtained value?

Yes

No

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.

```
ret = E_AUTO_GAIN_NO_ADJUSTMENT

count = 0U

while (count < (E_PGA_DSAD_MPX_MAX - 1U))
{
    gain_count = gs_dsad_current_gain_set1[count]
    gain_count = gain_count < gs_con_gain_set_1_size ?
        gain_count++
    
    ret = E_AUTO_GAIN_NO_ADJUSTMENT
    return(ret)
}
```
5.15 PGA Offset Adjustment Execution Function

**MD_STATUS R_I1E_PGA_DSAD_OffsetAdjustment(e_pga_dsad_mpx_t dsad_mpx)**

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

```
if (ret == MD_OK) {
    bit_num = 4UL;
    if (bit_num > 0) {
        Generate offset value;
        Set offset;
        Execute ∆ΣA/D converter conversion;
        if (∆ΣA/D converter conversion successful?) {
            Get ∆ΣA/D converter conversion results;
            Correct ∆ΣA/D converter conversion results;
            if (dsad_value > 0) {
                Adjust offset value;
            }
        }
    }
    MD_OK = ret;
    Offset value = 0;
    Adjust offset value to 1;
    Update backup data;
    return (ret);
}
```
1

Store scan mode setting

mpx_count = E_PGA_DSAD_MPX0

[mpx_count < E_PGA_DSAD_MPX_MAX]?

No

Yes

Store ΔΣ/A/D converter multiplexer setting

mpx_count++

2

3

Restored scan mode setting

mpx_count = E_PGA_DSAD_MPX0

[mpx_count < E_PGA_DSAD_MPX_MAX]?

No

Yes

Restored ΔΣ/A/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  

<table>
<thead>
<tr>
<th>e_disconnection_return_t</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_DISCONNECTION_NO_DETECT</td>
<td>Disconnection not detected</td>
</tr>
<tr>
<td>E_DISCONNECTION_DETECT</td>
<td>Disconnection detected</td>
</tr>
</tbody>
</table>

Processing  
Checks for disconnection of input multiplexer specified by ΔΣA/D converter  
Returns E_DISCONNECTION_DETECT when disconnection is detected.
5.17 RL78/I1E AFE Stop Processing Function

```c
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

```c
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
- DΣADMR:
  - 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

```c
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`. 

```c
// Processing example
void R_I1E_CAMP_Calibration(uint8_t *camp_trimming)
{
    // Example code
    uint8_t trim_value = 0x0A; // Example trimming value
    *camp_trimming = trim_value;
}
```
6. Flash Memory API

6.1 Data Flash Memory Storage Data Check Function

```c
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.

![Flowchart](chart.png)
6.2 Data Flash Memory Storage Data Overwrite Function

```c
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.

```
R_I1E_FlashRewrite

ret = D_FLASH_FAILURE

Access disabled state?
  No
  - Start use of data flash
  - Erase data flash block

  Voltage mode setting invalid?
    No
    - Block erase successfully completed?
      No
      - Calculate checksum
      - Overwrite data flash
      - Call in access disabled state?
        Yes
        - End use of data flash
        return (ret)
      Yes
      - Overwrite data flash
      - Calculate checksum
      - Overwrite data flash
      - Call in access disabled state?
        Yes
        - End use of data flash
        return (ret)
    Yes
    - Block erase successfully completed?
      Yes
      - Calculate checksum
      - Overwrite data flash
      - Call in access disabled state?
        Yes
        - End use of data flash
        return (ret)
      No
      - Overwrite data flash
      - Calculate checksum
      - Overwrite data flash
      - Call in access disabled state?
        Yes
        - End use of data flash
        return (ret)
```
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.

```
if (R_I1E_UartSend(p_send_data)) {
  // Transmitted successfully
} else {
  // Error
}
```
R_I1E_UartSend

Transmission in process?

Yes

return (MD_ERROR)

No

Get number of transmit data

Number of transmit data incorrect?

Yes

return (MD_ARGERROR)

No

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

```c
void R_TAU_FreeRunTimerInit(void)
```

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

8.2 Free Running Timer Stop Function

```c
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

```c
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

```c
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

```c
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.

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

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

3. Update continuous active level detection counter by +1
   - No

4. Processing key scan E_KEY_OFF processing
   - No

---

**Processing key scan E_KEY_OFF processing**

- Non-active at previous key scan?
  - Yes
  - Update key information active
  - Update continuous active level detection counter by +1
  - Clear continuous active level detection counter 0
- Continuous active level >= KEY_SCAN_NORM - 1U?
  - Yes
  - i = 0U, bit_data = 0U
  - i < SW_TYPE_MAX?
    - Yes
    - Multiple push group?
      - Yes
      - Initialize current data selection
      - Initialize same group data
      - bit_data = 1U
      - i++
    - No
    - OU == bit_data?
      - Yes
      - gs_key_status[key_num] = E_KEY_OFF_TO_ON
      - No
  - No
  - Continuous active level < KEY_SCAN_NORM - 1U?
    - Yes
    - i = 0U, bit_data = 0U
    - i < SW_TYPE_MAX?
      - Yes
      - Multiple push group?
        - Yes
        - Initialize current data selection
        - Initialize same group data
        - bit_data = 1U
        - i++
      - No
      - OU == bit_data?
        - Yes
        - gs_key_status[key_num] = E_KEY_OFF_TO_ON
        - No
    - No
- Different state than previous scan?
  - Yes
  - Execute key OFF state processing
  - Update continuous active level detection counter by +1
  - Continuous active level >= KEY_SCAN_LONG - 1U?
    - Yes
    - Clear click number counter 0
    - i = 0U, bit_data = 0U
    - i < SW_TYPE_MAX?
      - Yes
      - Multiple push group?
        - Yes
        - Initialize current data selection
        - Initialize same group data
        - bit_data = 1U
        - i++
      - No
      - OU == bit_data?
        - Yes
        - gs_key_status[key_num] = E_KEY_ON_NORM or E_KEY_ON_LONG
        - No
    - No
    - Continuous active level < KEY_SCAN_LONG - 1U?
      - Yes
      - i = 0U, bit_data = 0U
      - i < SW_TYPE_MAX?
        - Yes
        - Multiple push group?
          - Yes
          - Initialize current data selection
          - Initialize same group data
          - bit_data = 1U
          - i++
        - No
        - OU == bit_data?
          - Yes
          - gs_key_status[key_num] = E_KEY_ON_NORM or E_KEY_ON_LONG
          - gs_key_active_count[key_num]++
          - No
    - No
9.2 Key Information Initialization

```c
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**:

1. `key_num = 0U`
2. If `(key_num < SW_TYPE_MAX)` then:
   - Initialize key information
   - `key_num++`
3. Initialize last key information
4. `return`

Diagram:
- Start with `R_KEY_Initialize`
- `key_num = 0U`
- Check `(key_num < SW_TYPE_MAX)`
  - If true, initialize key information and increment `key_num`
  - If false, initialize last key information

Flowchart:
- Start: `R_KEY_Initialize`
- `key_num = 0U`
- Decision: `(key_num < SW_TYPE_MAX)`
  - Yes: Initialize key information, `key_num++`
  - No: Initialize last key information
- End: `return`
### 9.3 Get Key Information (specified keys only)

#### `e_key_status_t R_KEY_Get(e_key_t const key)`

<table>
<thead>
<tr>
<th>Description</th>
<th>Get key information (specified keys only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Argument key:</td>
<td>SW names specified in <code>e_key_t</code></td>
</tr>
<tr>
<td>Global Variables gs_key_status[][]:</td>
<td>Key information array</td>
</tr>
<tr>
<td>SFR</td>
<td>None</td>
</tr>
<tr>
<td>Return Value e_key_status_t:</td>
<td></td>
</tr>
<tr>
<td>E_KEY_OFF</td>
<td>0  The key is not pushed</td>
</tr>
<tr>
<td>E_KEY_OFF_TO_ON</td>
<td>1  The key was pushed ( off to normal push )</td>
</tr>
<tr>
<td>E_KEY_ON_NORM</td>
<td>2  The key was pushed ( normal push )</td>
</tr>
<tr>
<td>E_KEY_NORM_TO_LONG</td>
<td>3  The key was pushed ( normal to long push )</td>
</tr>
<tr>
<td>E_KEY_ON_LONG</td>
<td>4  The key was pushed ( push and hold )</td>
</tr>
<tr>
<td>E_KEY_LONG_TO_LONG</td>
<td>5  The key was pushed ( long to long push )</td>
</tr>
<tr>
<td>E_KEY_DOUBLE_CLICK</td>
<td>6  The key was double pushed</td>
</tr>
</tbody>
</table>

**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)

```c
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 retrieved, therefore R_KEY_Scan() must be executed before this function.

```
R_KEY_GetAll

key_num = 0U

(key_num < SW_TYPE_MAX)?
  Yes
  Copy key information
  key_num++
  No

return
```
9.5 Specified Key Click Wait

```c
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

---

**Flowchart**:
- **R_KEY_WaitOneClick**
  - Initialize key information
  - **No**: Intermittent generated by 10ms cycle timer?
    - **Yes**: Clear 10ms cycle timer interrupt flag
      - **Yes**: Key scan
        - **No**: Specified key pushed?
          - **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.
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.
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.
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.
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.
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 completing 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.
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.

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/stop 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>
<thead>
<tr>
<th>Parameter</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baud rate (speed)</td>
<td>1Mbps</td>
</tr>
<tr>
<td>Data bits</td>
<td>8</td>
</tr>
<tr>
<td>Parity</td>
<td>None</td>
</tr>
<tr>
<td>Stop bits</td>
<td>1</td>
</tr>
<tr>
<td>Order</td>
<td>LSB first</td>
</tr>
</tbody>
</table>

Table 10-1 UART Communication Settings

10.2.1 Communication sequence

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

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

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,...

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
```

- **value1**: Specify number of data.
- **value2**: Data ID
- **string**: Data name

ASCII code “colon” separates command and argument.
Use commas to separate arguments.

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,...
```

- **value1**: Data ID
- **value2**: Command
- **value3**: ASCII code “colon” separates command and argument.

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.
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.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Transmit number counter</td>
</tr>
<tr>
<td></td>
<td>1 byte</td>
</tr>
<tr>
<td>2</td>
<td>24-bit A/D converted value</td>
</tr>
<tr>
<td></td>
<td>3 bytes</td>
</tr>
</tbody>
</table>

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.

```
@value
```

Carriage return + linefield (CR+LF)
Command: 1 byte
Preamble: 1 byte

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 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.xlsx 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 transmission 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 Chapter 10.2.2)

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.

![CS+ Debug Tool Setting Window](image)
Website and Support

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

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

All trademarks and registered trademarks are the property of their respective owners.
<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Page</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Nov. 09, 2015</td>
<td>---</td>
<td>First edition issued</td>
</tr>
</tbody>
</table>
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.