Renesas RA Family

Low Power Application (Use of ADC and DTC at Snooze mode) for FPB-RA0E1

Introduction

This application note describes the features of RA0E1 MCU that are useful and typically required for low power application as a data logger, and its functionality can be dedicated for the application of the fitness trackers, fleet tracking devices, and so forth.

This application project is built with the use of 12-bit A/D Converter (ADC), Data Transfer Controller (DTC), 32-bit Interval Timer (TML32), Real-time Clock (RTC), Serial Array Unit (SAU), and Low-power Mode (LPM). The Snooze mode and Software Standby mode are used to reduce the average power consumption while keeping the data logger functionality. Application projects uses the e^2 studio Integrated Development Environment (IDE) and Flexible Software Package (FSP) provided for the RA family.

Prerequisites

We assume that you have developed Renesas e^2 studio IDE and Flexible Software Package (FSP). We recommend that you build and run the Blinky project according to section "Tutorial: Your First RA MCU Project – Blinky" in FSP User's Manual (R11UM0155) prior to trying out this application. You can then become familiar with the e^2 studio IDE and FSP and verify that the debugging connectivity to the boards is working correctly.

Required Resources

This application project is created for Renesas RA Family MCU RA0E1. When applying this application note to other MCUs, be sure to change it according to the specifications of the MCU and evaluate it carefully. The resources required for this application project are as follows.

Hardware

- Renesas RA Kit FPB-RA0E1
- Seeed Grove Base Shield V2.0 for Arduino (www.seeedstudio.com/Base-Shield-V2.html)
- Seeed Grove - Luminance Sensor (www.seeedstudio.com/Grove-Luminance-Sensor.html)
- Seeed Grove - Temperature Sensor (www.seeedstudio.com/Grove-Temperature-Sensor.html)
- USB – TTL Serial connector
- Grove Male Jumper wire

Development Tools and Software

- e^2 studio IDE version 2024-04 or later
- Renesas Flexible Software Package (FSP) version 5.3.0 or later
- GCC ARM Embedded Toolchain version 13.2
- Terminal software (like Tera Term)

Contents

1. Overview

1.1 Overview of Specifications

This application project acquires sensor data at regular intervals. These processes are intermittently executed though the Snooze mode transition during Software Standby mode to achieve operation as a low-power data logger.

In this application project, the device transfers to Snooze mode every 30 minutes during Software Standby mode. While the device is operating in Snooze mode, the A/D conversion and the result data copy to RAM buffer are performed. In addition, the low power mode is canceled every 24 hours by RTC alarm interrupt, and the measured data accumulated in the buffer is transmitted through UART communication. After the data transmission is done, the data logging operation repeats. When a user button is pressed, the low power mode is canceled, and the accumulated data is transmitted. Table 1 explains the details of these processes for each function.

Table 1. List of Functions

Function	Process Description
Data Acquisition Function	The ADC is triggered (software trigger from DTC) after saving the previous AD conversion result.
Data Output Function	When the RTC alarm interrupt or external IRQ interrupt is occurred, the device is woken up to Normal mode, and the data is output by UART communication (SAU).

Figure 1 shows MCU states and mode transition events, and Figure 2 shows a conceptual diagram of the operation modes and current consumption for RA0E1.

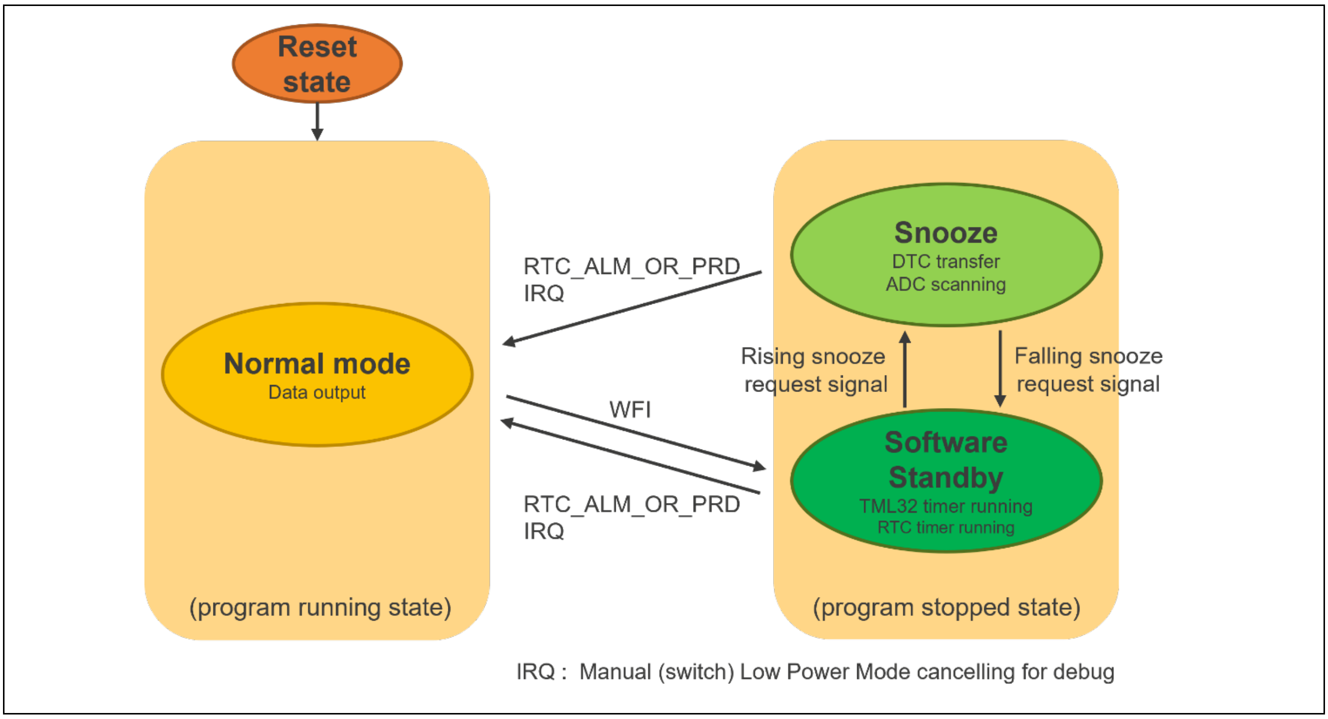


Figure 1 MCU Status and Mode Transition Events RA0E1

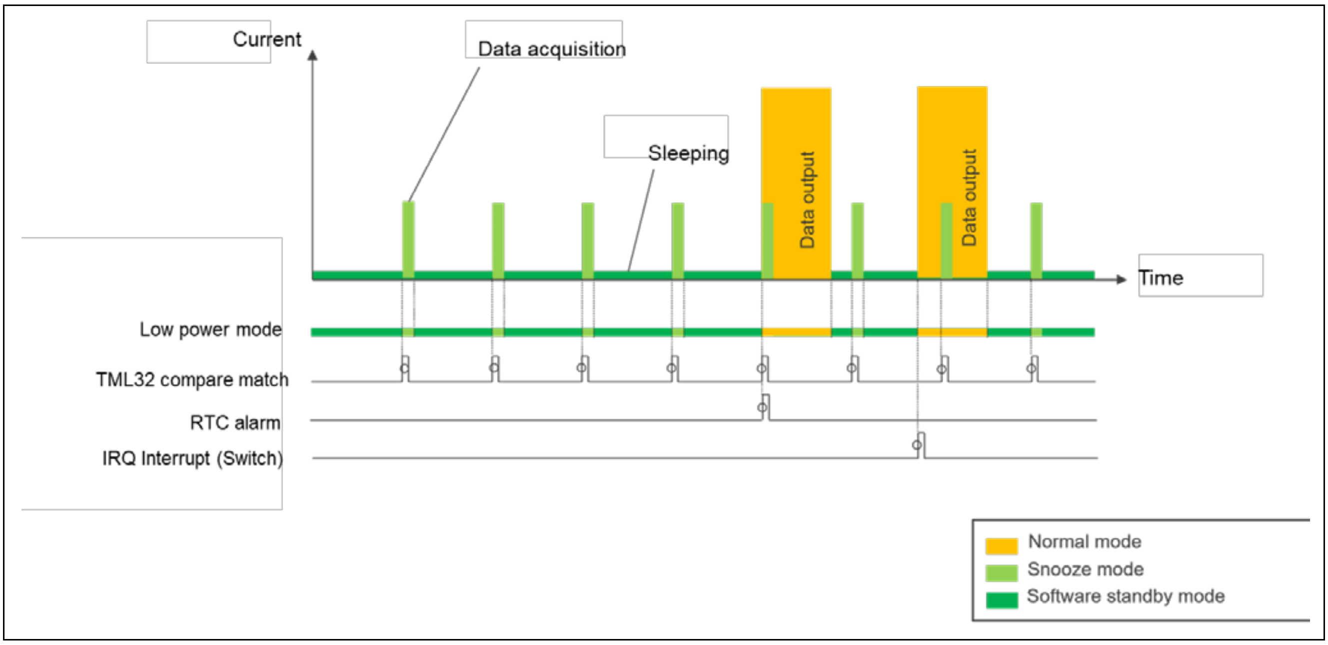


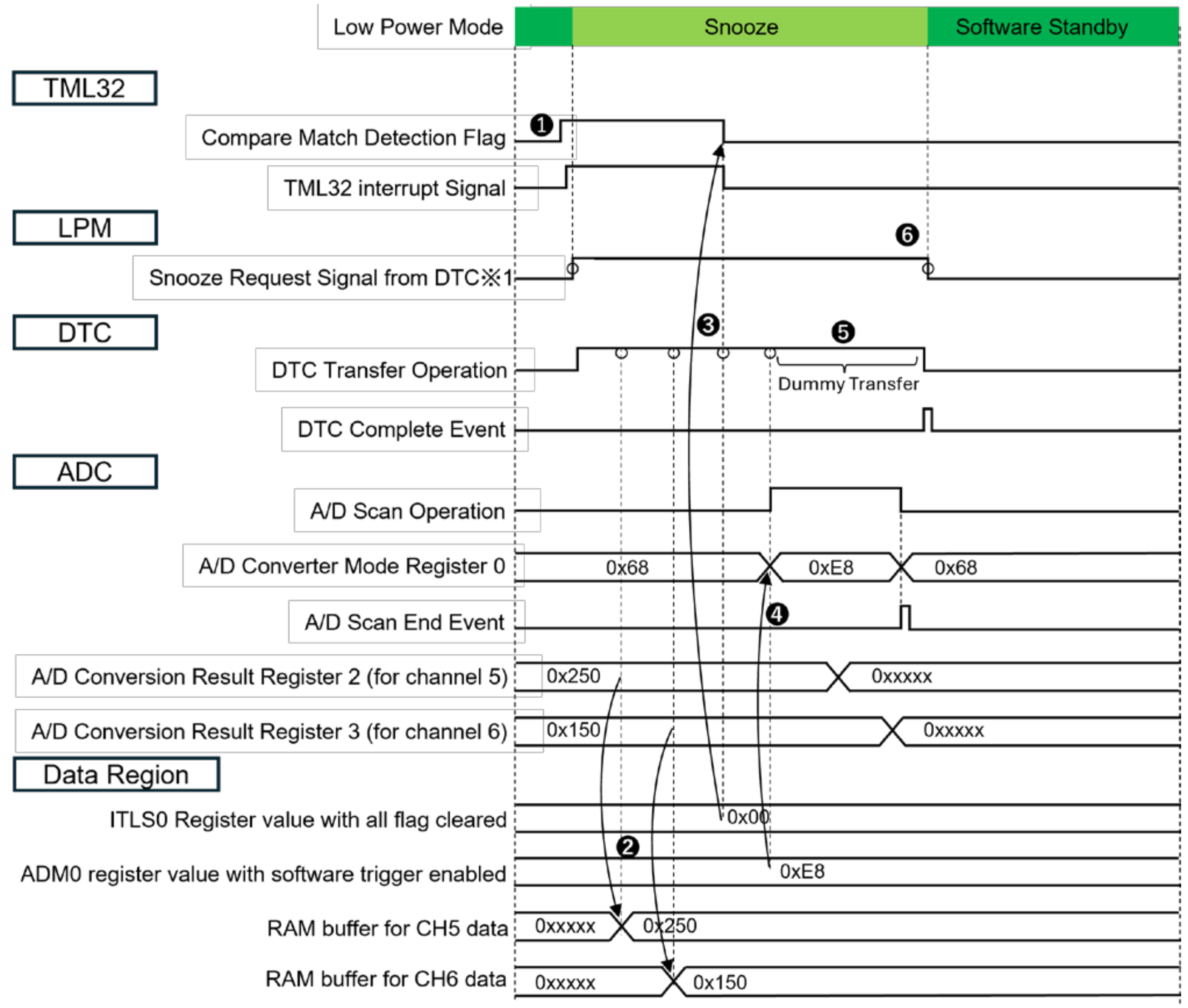
Figure 2 RA0E1 Conceptual Diagram of Operation Mode and Current Consumption

1.1.1 Data Acquisition Function

In the data acquisition operation, when the TML32 timer interrupt occurs, ICU request triggers the transition from Software Standby mode to Snooze mode, and the DTC chain transfer in snooze mode is started. By DTC chain transfer, firstly the value of the A/D conversion result register (A/D conversion result in previous turn) is transferred to a buffer in RAM, and secondly a compare match flag of TML32 is cleared, and thirdly a ADC scan is started by a software trigger. After the above transfer steps, DTC continues to perform dummy transfer so that the snooze mode signal does not fall while the ADC is operating. When the ADC Scan is ended and DTC chain transfer is completed, the device's low power mode returns to Software Standby mode.

The processing sequence for the above functions is described using Figure 3 Do not enable multiple Snooze requests at the same time. (Refer to Chapter 9.8 in the RA0E1 Hardware User's Manual (R01UH1040)).

1. The TML32 generates a compare match event at every 30 minutes.
2. The previous conversion result of ADC channel 5 or 6 is transferred to each data buffer.
3. The compare match flag of TML32 is cleared.
4. The ADC starts Scan by DTC writing 1 to A/D conversion operation bit (software trigger).
5. The DTC repeats dummy transfer to maintain output of a Snooze request signal.
6. When DTC dummy transfer is finished, the Snooze request signal is reset to low, and the low power mode returns to the Software Standby mode.



Note: The duration of each processing and event occurrence represents a concept.

Note 1: In RA0E1, Snooze request signal is issued by Snooze mode control circuit in each peripheral modules (SAU0 and ADC, ICU (for DTC)). When any of them receive start up request in Software Standby mode, Snooze request signal rises and system transfers to Snooze mode. Also Snooze mode end by similar way (falling Snooze request signal by peripherals). Do not enable multiple Snooze requests at the same time. (Refer to Chapter 9.8 in the RA0E1 Hardware User's Manual (R01UH1040)).

Figure 3 Timing Chart for Data Acquisition

1.1.2 Data Output Function

Once the device is woken up (either from software standby mode or snooze mode) to the Normal mode by either RTC alarm interrupt or external IRQ interrupt, the data output function is executed to transmit the data log through UART (Serial Array Unit (SAU)). After the data transmission is completed, the device returns to the software standby mode. In addition, if the device is woken up by the RTC alarm interrupt, the RTC alarm for the next transition to Normal mode is set, and then the mode transition to low power mode.

1.2 Peripheral Modules of RA0E1 MCU

Table 2 lists the main peripheral modules of RA0E1 MCU used in this application project, and the typical use case is described as well. The peripheral usage is also illustrated in the functional overview of RA0E1 MCU in Figure 4.

Table 2. Main Peripheral Modules

Module	Typical Uses
Low Power Modes	The Snooze mode and the Software Standby mode are used to achieve low power consumption.
12-bit ADC	Converts the analog continuous time signal from the sensor to discrete digital signal value.
DTC	The analog conversion result of the A/D converter is transferred to the buffer using the DTC in repeat mode. A DTC chain transfer is used to transfer the data from the ADC data register to RAM memory.
TML32	Generates a data acquisition timing every 30 minutes, and triggers the DTC transfer operation.
RTC	Measures the time starting from power on. The 24-hour alarm interrupt is used to cancel the low power mode and generate the timing of data output.
SAU	Performs UART communication with external equipment.
ICU	The Interrupt Controller Unit (ICU) controls which event signals are linked to the Nested Vector Interrupt Controller (NVIC), and the Data Transfer Controller (DTC) modules. The ICU also controls non-maskable interrupts.

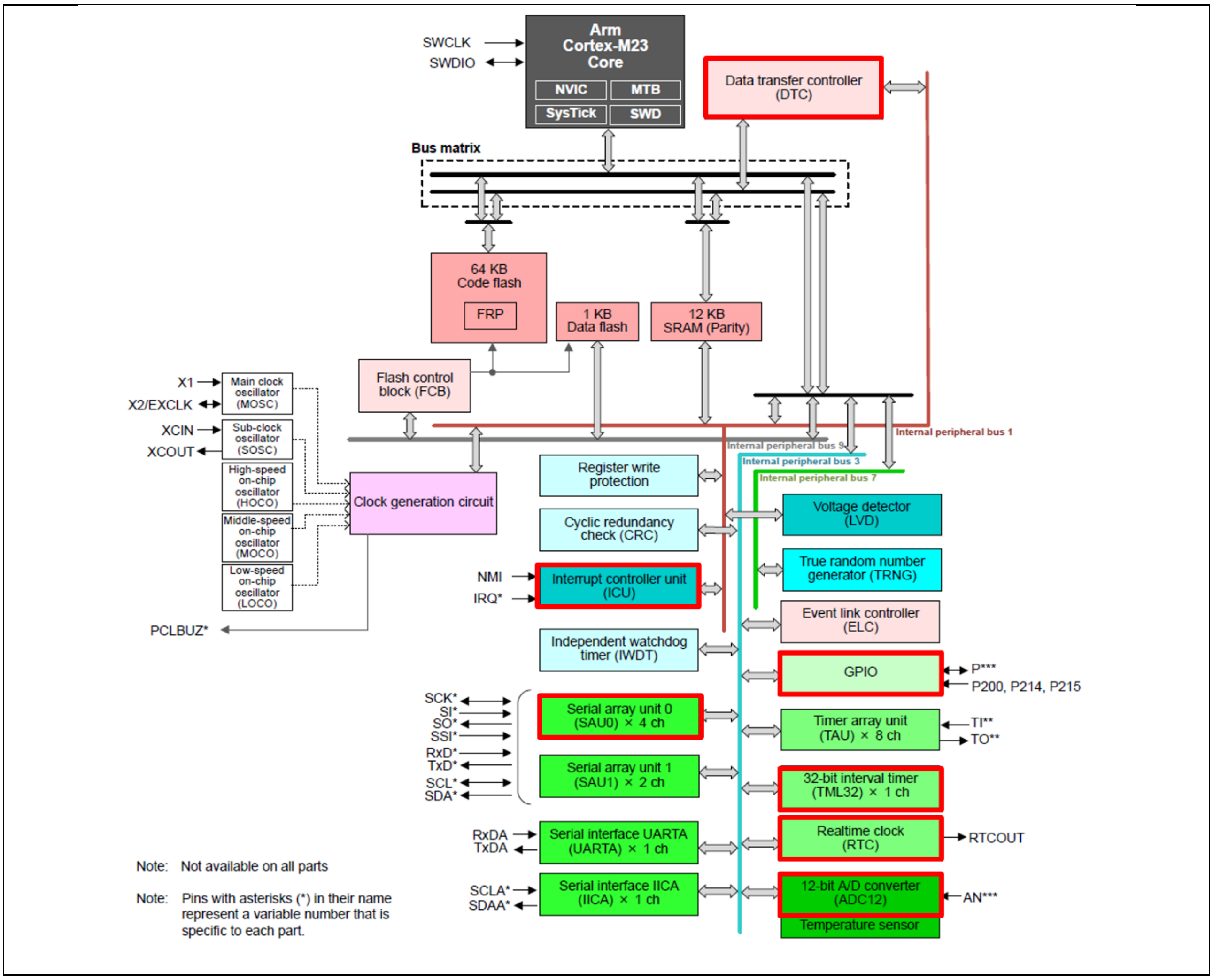


Figure 4 Functional Overview

2. Description of Functions Used

This section describes the functions of the LPM, ADC, DTC, TML32, and RTC modules of RA0E1 MCU and explains how to set them to achieve the expected operation.

2.1 Low Power Modes

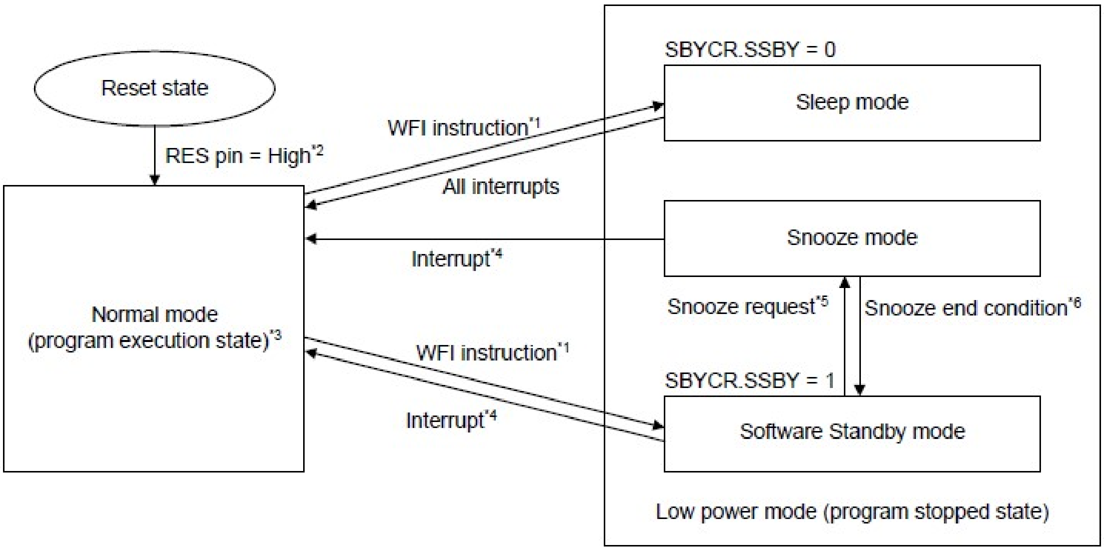
2.1.1 Available Low Power Modes

The table "Operating conditions of each low power mode" in the RA0E1 Hardware User's Manual (R01UH1040) describes the conditions to transition to low power modes, the states of the CPUs and peripheral modules, and the condition to cancel each mode.

The available low power modes are as follows:

- Sleep mode
- Software Standby mode
- Snooze mode

Figure 5 shows a schematic diagram of the low power mode transitions.



Note 1. When an interrupt that acts as a trigger for cancel is received during a transition to the program stopped state after the execution of a WFI instruction, the MCU executes interrupt exception handling instead of a transition to low power mode.
Note 2. The MOCO is the source of the operating clock following a transition from the reset state to Normal mode.
Note 3. The transition to Normal mode is made from an interrupt in Sleep mode, Software Standby mode, or Snooze mode. The clock source is the same as before entering the low power mode.
Note 4. See Table 9.3.
Note 5. See section 9.8.1. Transition to Snooze Mode.
Note 6. See section 9.8.3. Returning from Snooze Mode to Software Standby Mode.

Figure 5 Schematic Diagram of Low Power Mode Transitions

2.1.1.1 Sleep Mode

Generally, CPU operation is the most significant factor for the increase in power consumption. In this mode, the CPU halts operation but retains the value of the CPU's internal registers. Peripheral functions other than CPU are not stopped. The Sleep mode is canceled when an available reset or interrupt occurs in Sleep mode. All interrupt sources are available.

2.1.1.2 Software Standby Mode

This mode dramatically reduces power consumption by stopping the CPU, most on-chip peripheral functions, and oscillators. However, CPU's internal registers, SRAM data, the on-chip peripheral functions, and I/O port status are retained.

2.1.1.3 Snooze Mode

This mode is an extension of Software Standby mode that allows limited peripheral modules to operate with the CPU halted. This reduces current consumption by flexible operation of peripheral modules required by the application. Snooze mode can be entered from Software Standby mode by a specified interrupt request. Interrupt requests available in Snooze mode can also transition from Snooze mode to Normal mode or Software Standby mode.

2.1.2 Low Power Mode Transition Events

For details on the available low power mode transitions, refer to the relevant sections in the RA0E1 Hardware User's Manual (R01UH1040) below.

- Table 9.8: Available Snooze requests to switch to Snooze mode.
- Table 9.3: Available interrupt sources to transition to Normal mode from Snooze mode and Software Standby mode.

Figure 6 also shows the low power transition event settings on FSP configurator.

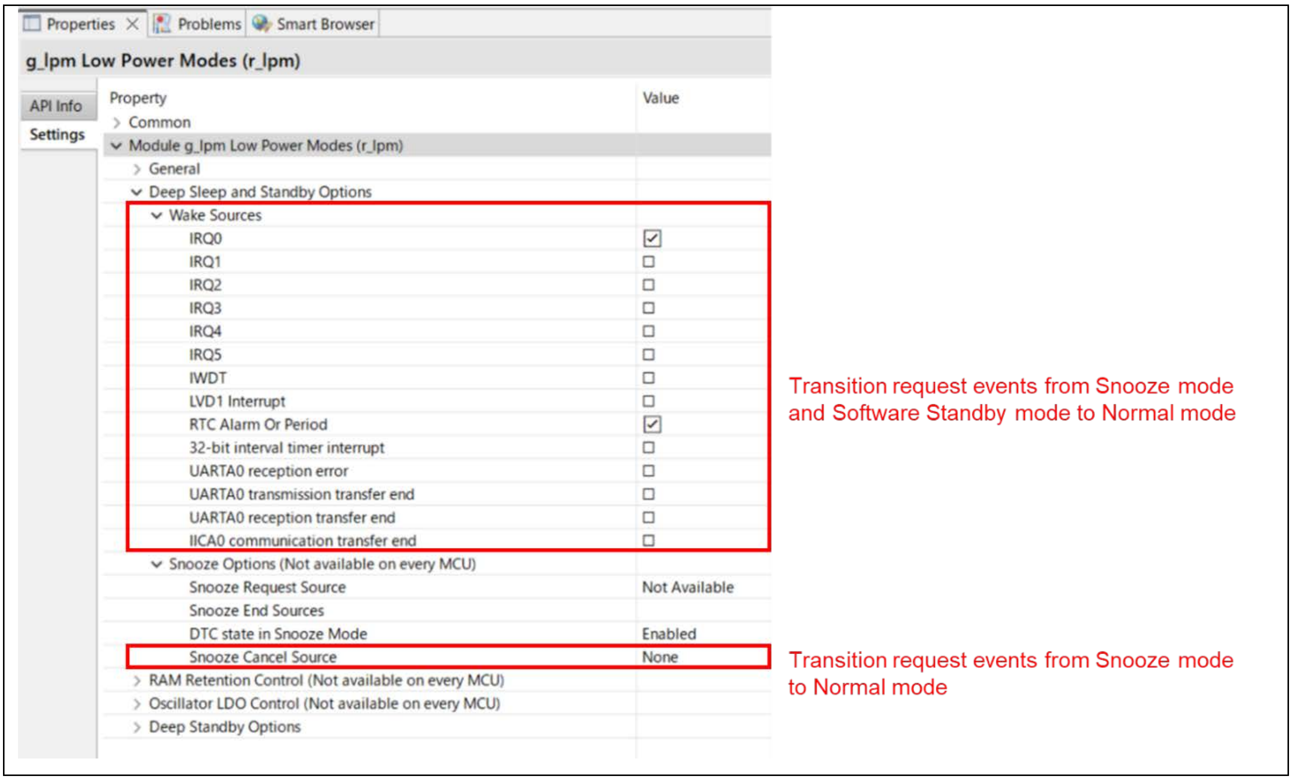


Figure 6 Low Power Mode Transition Event Settings on FSP Configurator

In this application project, Snooze request signal rises and the MCU enters snooze mode when DTC activation request is output from ICU, as Figure 7 shows.

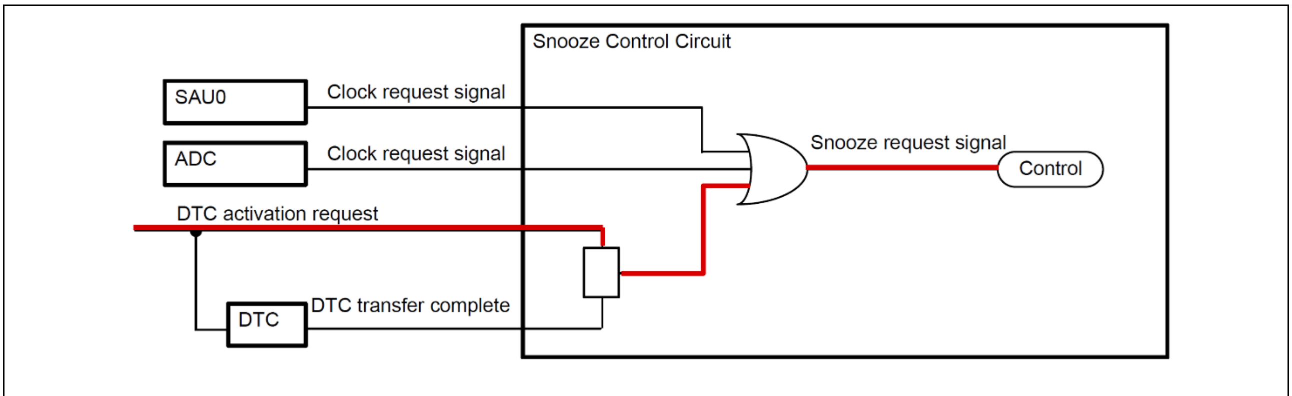


Figure 7 Snooze request signal form DTC

When DTC sets a software trigger of ADC in Snooze mode, A/D conversion starts. As shown in Figure 8, the DTC must keep raising Clock request signal (Snooze request signal) to provide HOCO to ADC module while ADC is operating. Therefore, DTC needs to create time during ADC operation, for example, creating multiple dummy transfers. After ADC operation is completed, DTC can fall snooze request signal.

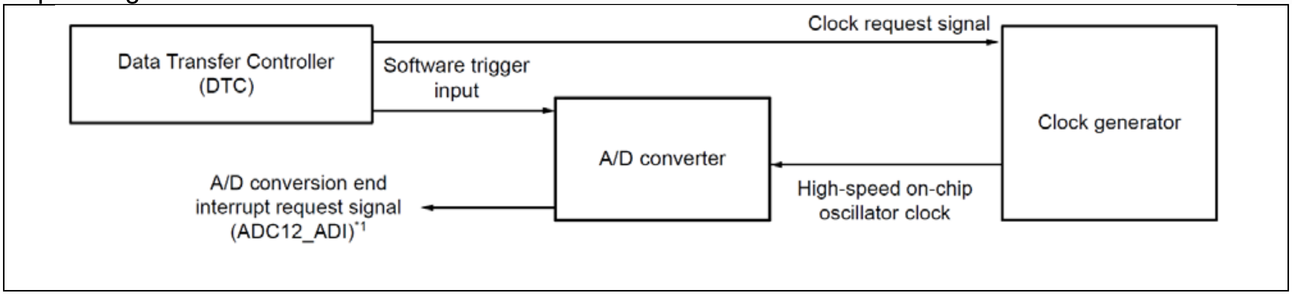


Figure 8 Block diagram when using Snooze mode in software trigger wait mode

Table 3 lists the mode transition events used in this application project.

Table 3. Mode Transition Events Used in this Application Project

Transition Request	Event
Request for transition to Snooze mode	Rising snooze request signal by ICU (for DTC).
Snooze mode end request	Falling snooze request signal by DTC
Request for transition to Normal mode from Snooze mode or Software Standby mode	PORT_IRQ0 RTC_ALM_OR_PRD

2.2 12-Bit A/D Converter

The RA0E1 MCU includes a 12-bit successive approximation A/D converter (ADC12) unit. There are 8 types of conversion modes as shown below.

- Trigger mode
  - Software trigger no-wait mode: Conversion is started by setting the ADCE bit to 1 by software, and then setting ADCS to 1 after the A/D power supply stabilization wait time has passed.
  - Software trigger wait mode: The power is turned on by setting the ADCS bit to 1 by software while A/D conversion is stopped and conversion is then started automatically after the A/D power supply stabilization wait time has passed.
  - Hardware trigger no-wait mode: Conversion is started by detecting a hardware trigger.
  - Hardware trigger wait mode: The power to the A/D converter is turned on by detecting a hardware trigger while the A/D converter is off and in the conversion standby state, and conversion is then started automatically after the stabilization wait time passes.
- Channel selection mode
  - Select mode: A/D conversion is performed on the analog input of one selected channel.
  - Scan mode: A/D conversion is performed on the analog input of four channels in order. Four consecutive channels can be selected from AN000 to AN007 as analog input channels
- Conversion operation mode
  - One-shot conversion mode: A/D conversion is performed on the selected channel once.
  - Sequential conversion mode: A/D conversion is sequentially performed on the selected channels until it is stopped by software.

Note1. Refer to Table “A/D conversion mode” in the RA0E1 Hardware User's Manual (R01UH1040) for a deep understanding of the A/D conversion modes.

2.2.1 A/D Conversion Channel

The 12-bit A/D converter unit (ADC12) can select up to 10 channels (AN000–AN007, AN021, AN022)[Note 1] of analog input, temperature sensor, and internal reference voltage.

- In Select mode, you can choose one channel from 10 channels analog input, a temperature sensor, and an internal reference voltage. Please refer to Table 25.12 in the RA0E1 Hardware User's Manual (R01UH0140).
- In Scan mode, you can select any continuous 4 channels from supported channels (AN000-007).

For details on input source selection, refer to the relevant sections in the RA0E1 Hardware User's Manual (R01UH1040) below.

- Table 25.12: Input source selection by ADS[4:0] bits and ADISS bit in select mode
- Table 25.13: Input source selection by ADS[4:0] bits and ADISS bit in scan mode

Figure 9 shows the ADC scan channels settings in FSP configurator.

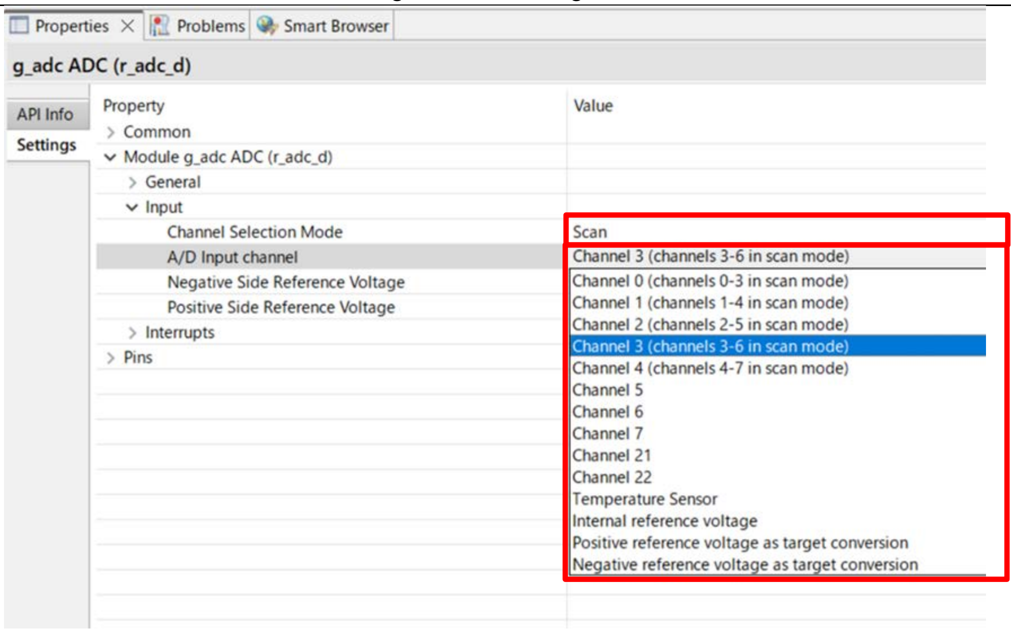


Figure 9 FPB-RA0E1 ADC Scan Channels Settings on FSP Configurator

The A/D conversion channel is selected by the ADS register. A/D conversion starts from the smallest channel number according to the conversion sequence for the analog input channel.

This application project uses Seeed's Grove Base Shield V2.0 for Arduino to connect the sensor through Arduino compatible connector of FPB-RA0E1 kit. J4-2 and J4-1 of Arduino compatible connector of FPB-RA0E1 are connected to analog channels 5 (AN005) and 6 (AN006) of the RA0E1 MCU, and can be read by selecting channels 5 and 6. The A/D conversion result is stored in the corresponding A/D data register (ADCR2, ADCR3)[Note 2].

Notes 1: The available input channels depend on the package type.

Notes 2: Channel 3-6 in Scan mode is selected in the ADS register.

Figure 10 shows the A/D conversion start trigger event settings in FSP configurator.

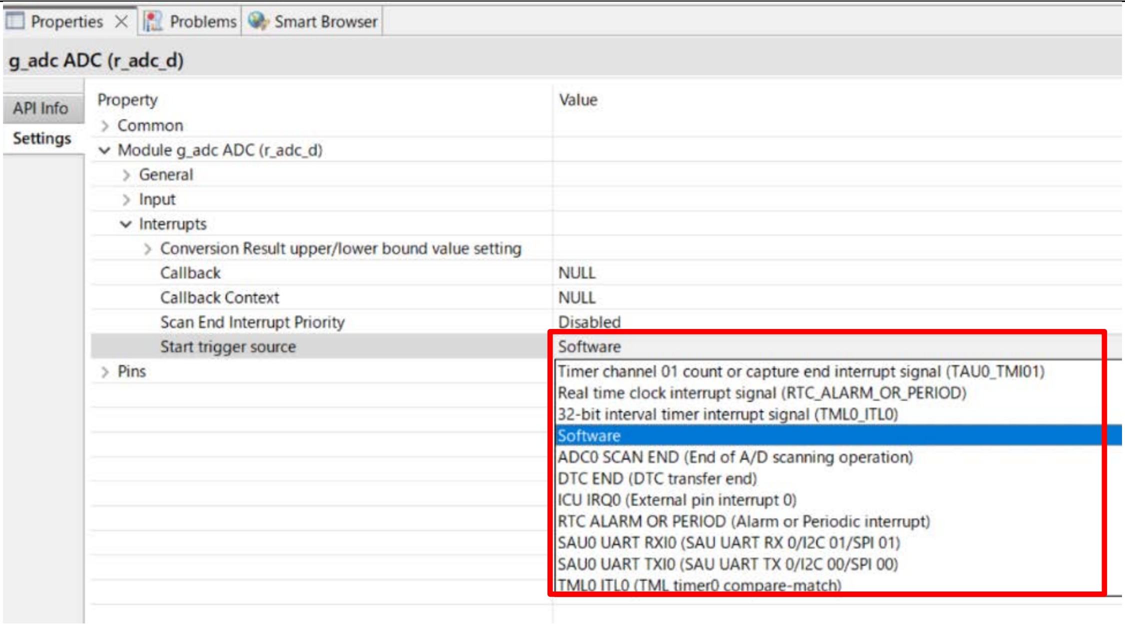


Figure 10 A/D convert Start Trigger Event Settings in FSP Configurator

2.2.2 Operation in Snooze mode

A/D conversion is stopped while in Software Standby mode. But, in Snooze mode, the ADC module can operate and can perform A/D conversion by inputting a software trigger or a hardware trigger without running the CPU.

2.2.2.1 A/D Conversion by Inputting a Software trigger

In Software Standby mode, ADC conversion is triggered by setting software trigger bit by DTC. The following conversion mode can be used.

- Trigger mode
  - Software trigger wait mode
- Channel selection mode
  - Select mode
  - Scan mode
- Conversion operation mode
  - One-shot conversion mode
  - Sequential conversion mode

When using the Snooze mode function with the software trigger, the ADM2.AWS bit must be 0. The Snooze request signal will not be generated by ADC module. Therefore, the DTC module needs to continue generating a Snooze request signal instead of ADC while the ADC conversion is running. For example, performing multiple dummy transfers.

2.2.2.2 A/D Conversion by Inputting a Hardware trigger

In the Software Standby mode, ADC conversion is triggered by inputting hardware trigger. The following conversion mode can be used.

- Trigger mode
  - Hardware trigger wait mode
- Channel selection mode
  - Select mode
  - Scan mode
- Conversion operation mode
  - One-shot conversion mode

The ADC can issue a snooze request in the hardware trigger wait mode. Set ADM2.AWC bit to 1 to enable Snooze mode function of the ADC module right before entering Software Standby mode. Also, the ADM2.AWC bit needs to be set to 0 after returning from Software Standby mode to Normal mode. If the ADM2.AWC bit is left set to 1, A/D conversion will not start normally despite the subsequent mode.

2.3 DTC Transfer

When the data transfer controller (DTC) is activated by an interrupt request, it transfers data according to the transfer information.

The following three transfer modes are available:

- Normal transfer mode: One data transfer is performed by one activation.
- Repeat transfer mode: One data transfer is performed by one activation. When data of repeat size is transferred, it returns to the address at the start of transfer.
- Block transfer mode: One block is transferred at one activation.

2.3.1 Transfer Start Event

The DTC can be activated by an interrupt request. When DTCENSETn.SET bit in ICU is set to 1, DTC starts transfer operation by the corresponding interrupt. To start DTC transfer operation with TML32 interrupt, set DTCENSET1.SET33 bit to 1.

Figure 11 shows DTC activation by an interrupt request from a peripheral module.

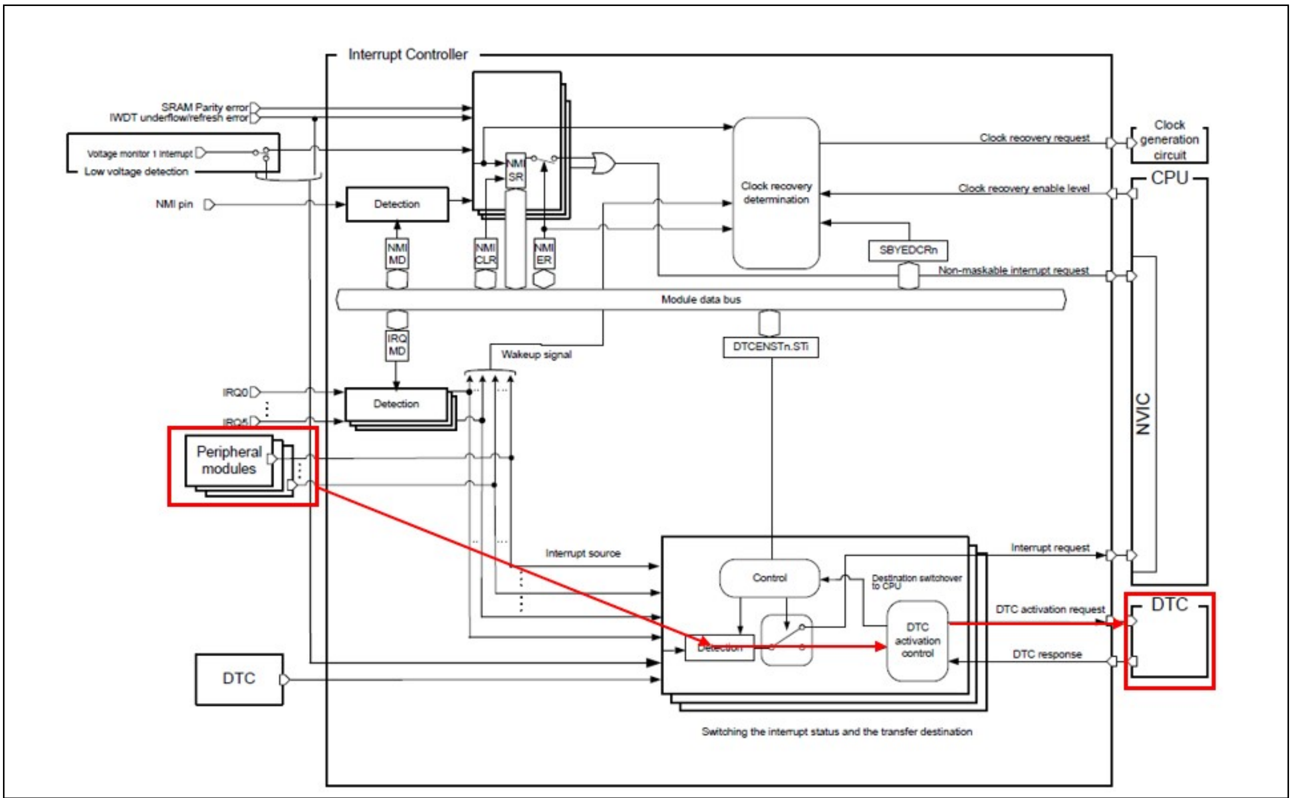


Figure 11 DTC Activation by an Interrupt Request from a Peripheral Module

2.3.2 Chain Transfer

The DTC can perform chain transfers in which multiple data transfers are performed continuously at a single activation source. When CHNE bit in MRB register is set to 1, the chain transfer function is enabled.

Figure 12 shows the chain transfer operation in this case.

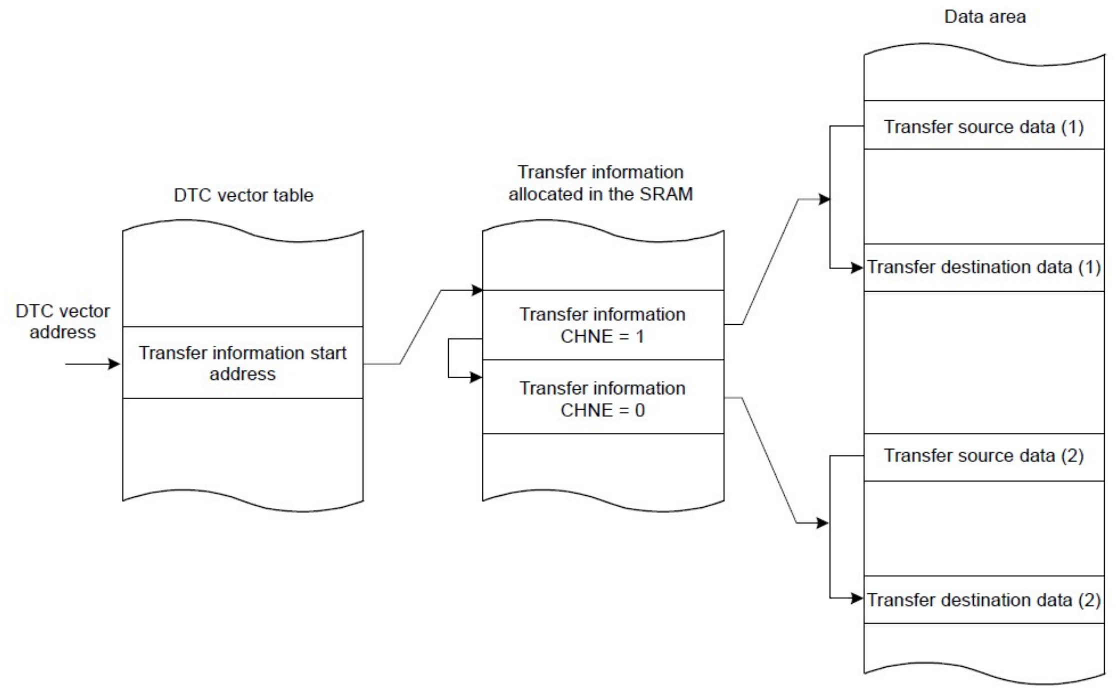


Figure 12 Chain Transfer Operation

In this application project, the purpose of DTC operation is the A/D conversion results copy to the data buffer, software trigger set for ADC, and compare match flag clearing for TML32 timer. Also, dummy transfers are performed multiple times since DTC needs to generate continuous Snooze request signal until ADC conversion operation is completed. The number of dummy transfers can be calculated using the information described in the following sections in the RA0E1 Hardware User’s Manual (R01UH1040).

- Section 25.2.1: ADM0 : A/D Converter Mode Register 0
- Section 14.4.8: Execution cycles of DTC

In FSP based project, the transfer information of chain transfer function can be set as follows.

```
transfer_info_t g_dtc_data_transfer_info[] = {
{ /* ADC data register 2 => Light sensor data buffer in RAM */
  .transfer_settings_word_b.dest_addr_mode = TRANSFER_ADDR_MODE_INCREMENTED,
  .transfer_settings_word_b.repeat_area = TRANSFER_REPEAT_AREA_DESTINATION,
  .transfer_settings_word_b.irq = TRANSFER_IRQ_EACH,
  .transfer_settings_word_b.chain_mode = TRANSFER_CHAIN_MODE_EACH,
  .transfer_settings_word_b.src_addr_mode = TRANSFER_ADDR_MODE_FIXED,
  .transfer_settings_word_b.size = TRANSFER_SIZE_2_BYTE,
  .transfer_settings_word_b.mode = TRANSFER_MODE_REPEAT,
  .p_dest = Address of RAM data buffer for channel 5,
  .p_src = Address of ADC data register 2 for channel 5 in Scan mode,
  .num_blocks = 0,
  .length = Size of RAM data buffer,
},
{ /* ADC data register 3 => Temperature sensor data buffer in RAM */
  .transfer_settings_word_b.dest_addr_mode = TRANSFER_ADDR_MODE_INCREMENTED,
  .transfer_settings_word_b.repeat_area = TRANSFER_REPEAT_AREA_DESTINATION,
  .transfer_settings_word_b.irq = TRANSFER_IRQ_EACH,
  .transfer_settings_word_b.chain_mode = TRANSFER_CHAIN_MODE_EACH,
  .transfer_settings_word_b.src_addr_mode = TRANSFER_ADDR_MODE_FIXED,
  .transfer_settings_word_b.size = TRANSFER_SIZE_2_BYTE,
  .transfer_settings_word_b.mode = TRANSFER_MODE_REPEAT,
  .p_dest = Address of RAM data buffer for channel 6,
  .p_src = Address of ADC data register 3 for channel 6 in Scan mode,
  .num_blocks = 0,
  .length = Size of RAM data buffer,
},
{ /* TML compare match detection flag clear */
  .transfer_settings_word_b.dest_addr_mode = TRANSFER_ADDR_MODE_FIXED,
  .transfer_settings_word_b.repeat_area = TRANSFER_REPEAT_AREA_DESTINATION,
  .transfer_settings_word_b.irq = TRANSFER_IRQ_EACH,
  .transfer_settings_word_b.chain_mode = TRANSFER_CHAIN_MODE_EACH,
  .transfer_settings_word_b.src_addr_mode = TRANSFER_ADDR_MODE_FIXED,
  .transfer_settings_word_b.size = TRANSFER_SIZE_1_BYTE,
  .transfer_settings_word_b.mode = TRANSFER_MODE_REPEAT,
  .p_dest = Address of TML32 Interval Timer Status Register,
  .p_src = Address of data variable that contains all flag clearing value,
  .num_blocks = 0,
  .length = 1,
},
```

```
{ /* ADC software trigger set */
  .transfer_settings_word_b.dest_addr_mode = TRANSFER_ADDR_MODE_FIXED,
  .transfer_settings_word_b.repeat_area = TRANSFER_REPEAT_AREA_DESTINATION,
  .transfer_settings_word_b.irq = TRANSFER_IRQ_EACH,
  .transfer_settings_word_b.chain_mode = TRANSFER_CHAIN_MODE_EACH,
  .transfer_settings_word_b.src_addr_mode = TRANSFER_ADDR_MODE_FIXED,
  .transfer_settings_word_b.size = TRANSFER_SIZE_1_BYTE,
  .transfer_settings_word_b.mode = TRANSFER_MODE_REPEAT,
  .p_dest = Address of A/D Converter Mode Register 0,
  .p_src = Address of data variable that contains A/D Converter Mode Register 0 value with software
trigger,
  .num_blocks = 0,
  .length = 1,
},
{ /* Dummy transfer */
  .transfer_settings_word_b.dest_addr_mode = TRANSFER_ADDR_MODE_FIXED,
  .transfer_settings_word_b.repeat_area = TRANSFER_REPEAT_AREA_DESTINATION,
  .transfer_settings_word_b.irq = TRANSFER_IRQ_EACH,
  .transfer_settings_word_b.chain_mode = TRANSFER_CHAIN_MODE_EACH,
  .transfer_settings_word_b.src_addr_mode = TRANSFER_ADDR_MODE_FIXED,
  .transfer_settings_word_b.size = TRANSFER_SIZE_1_BYTE,
  .transfer_settings_word_b.mode = TRANSFER_MODE_REPEAT,
  .p_dest = Address of data variable for dummy transfer destination,
  .p_src = Address of data variable for dummy transfer source,
  .num_blocks = 0,
  .length = 1,
},
:
: Repeat the dummy transfer multiple times.
:
{ /* last dummy transfer (with chain transfer end) */
  .transfer_settings_word_b.dest_addr_mode = TRANSFER_ADDR_MODE_FIXED,
  .transfer_settings_word_b.repeat_area = TRANSFER_REPEAT_AREA_DESTINATION,
  .transfer_settings_word_b.irq = TRANSFER_IRQ_EACH,
  .transfer_settings_word_b.chain_mode = TRANSFER_CHAIN_MODE_DISABLED,
  .transfer_settings_word_b.src_addr_mode = TRANSFER_ADDR_MODE_FIXED,
  .transfer_settings_word_b.size = TRANSFER_SIZE_1_BYTE,
  .transfer_settings_word_b.mode = TRANSFER_MODE_REPEAT,
  .p_dest = Address of data variable for dummy transfer destination,
  .p_src = Address of data variable for dummy transfer source,
```

```
  .num_blocks = 0,
  .length = 1,
}
};
```

2.4 32-bit Interval Timer (TML32)

The RA0E1 MCU implements a 32-bit Interval Timer with four channels (n=0~3). This TML32 can be used over a four-channel connection to generate a long-period interrupt.

Refer to the relevant part of the RA0E1 Hardware User's Manual (R01UH1040) for more information. Figure 13 also shows TML32 mode selection and count source settings on FSP configurator.

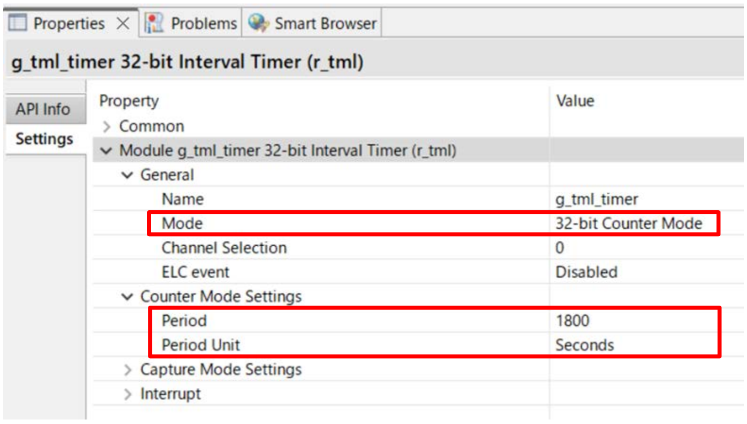


Figure 13 TML32 Mode Selection and Count Source Settings on FSP Configurator

In this application project, 32-bit Counter mode (four-channel connection) is used because it is necessary to generate a long period time (30 minutes) to perform periodic Snooze mode operation.

2.5 Realtime clock (RTC)

The RA0E1 RTC supports Calendar count mode and two count sources. The Calendar count mode retains the 100-year calendar and automatically corrects the leap year date. The following two interrupts are the source of the Realtime Clock interrupt signal (RTC_ALM_OR_PRD).

- Fixed-cycle interrupt: Period selectable from among 0.5 of a second, 1 second, 1 minute, 1 hour, 1 day, or 1 month
- Alarm interrupt: Alarm set by day of week, hour, and minute

For more information, refer to Table "RTC specifications" in RA0E1 Hardware User's Manual (R01UH1040).

In this application note, the RTC is configured so that the alarm interrupt is generated every 24 hours pass.

Note: The count of years, months, weeks, days, hours, minutes, and seconds can only proceed when the sub-clock oscillator (SOSC = 32.768 kHz) is selected as the operating clock of the Realtime Clock (RTCCLK). When the low-speed on-chip oscillator clock (LOCO = 32.768 kHz) is selected, only the fixed-cycle interrupt is available.

3. Low-Power Data Logger Application

This chapter describes the detailed design of the low-power data logger application using the module features described in the previous chapter.

3.1 Functional Specification

This application project is implemented based on the following specifications. Figure 14 also shows the overall algorithm for the low-power data logger.

3.1.1 Data Acquisition Function

- Analog signals of two sensors (Illuminance sensor Seeed Grove-Luminance Sensor, Temperature sensor Seeed Grove-Temperature Sensor) connected to external pins are acquired by A/D converter module.
- Analog signal input values are between the low-potential reference voltage (VSS) and the high-potential reference voltage (VCC) set between 0 V and 3.3 V respectively.
- Acquisition of data from each sensor is performed every 30 minutes regardless of the low power mode status.
- The size of the measurement data buffer is 96 samples (48 hours of sampling every 30 minutes).

3.1.2 Data Output Function

- Perform data output every 24 hours triggered by RTC alarm interrupt or external IRQ interrupt generated by user switch.
- UART communication format is 115,200 bps baud rate, 8-bit data length, no parity, 1 stop bit.

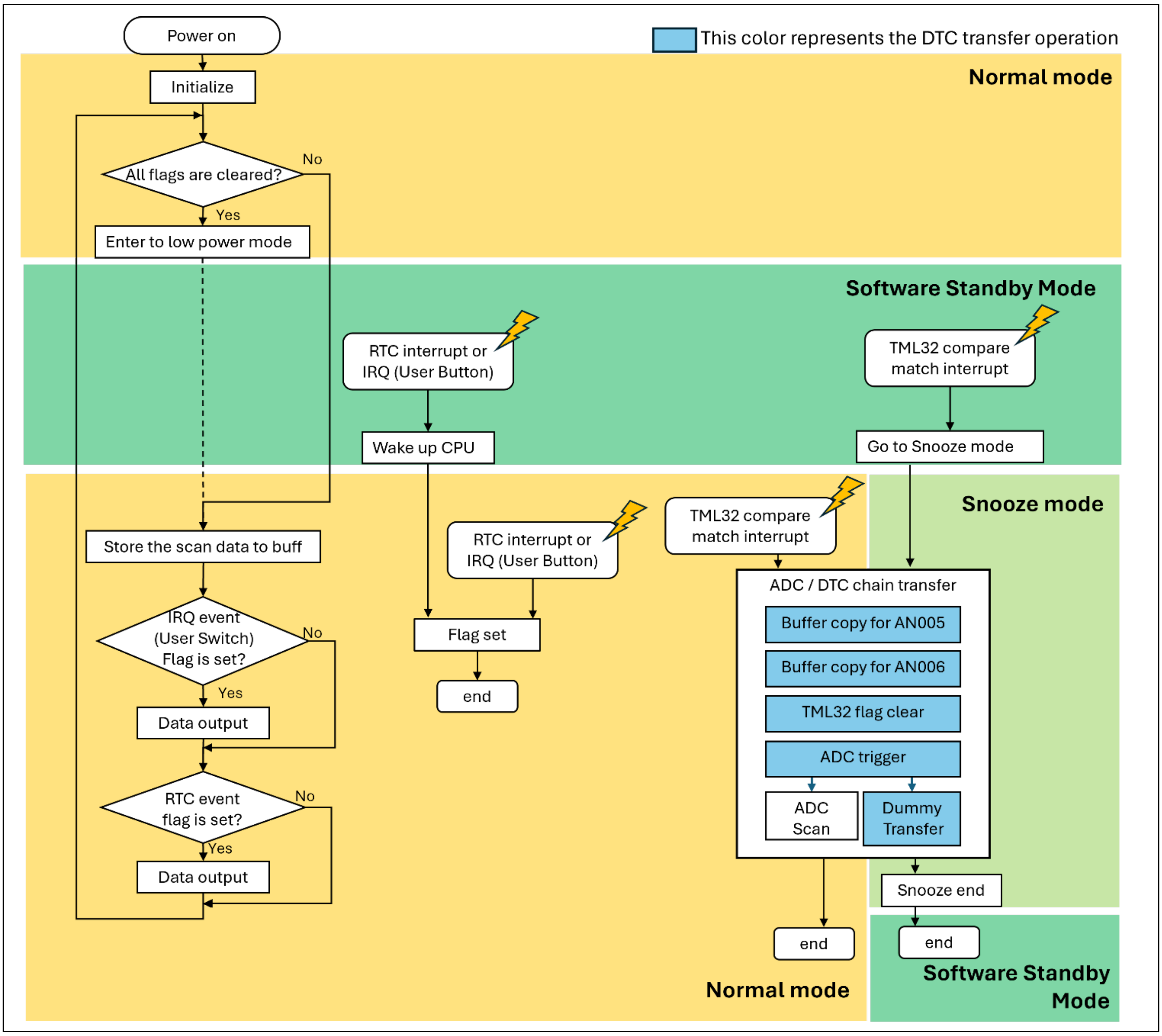


Figure 14 Overall Algorithm of the Low-Power Data Logger

3.2 User Interface

This section describes the user interface for the Low-Power Data Logger application. See chapter 4.4 "Procedure for Checking the Operation of the Low-Power Data Logger" for more information of checking the program behavior.

Figure 15 shows the system overview of this application project.

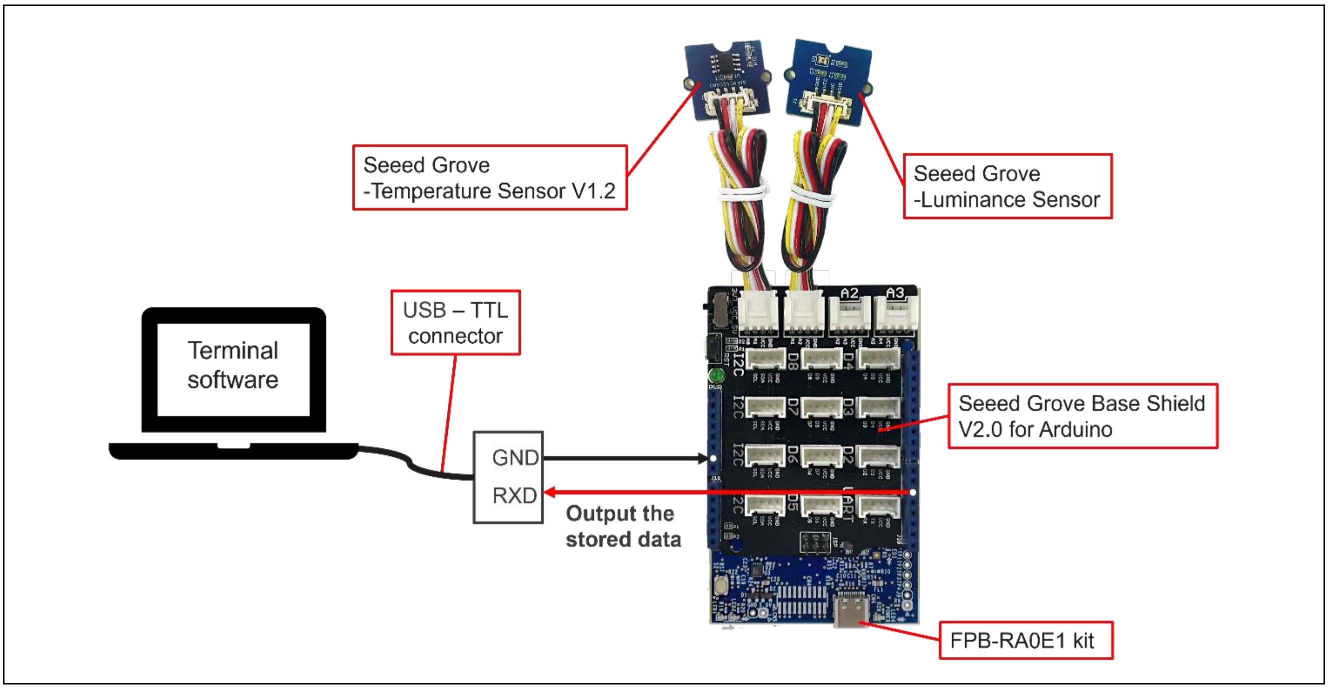


Figure 15 System Overview

3.2.1 Connecting the Sensor

Attach Seeed's Grove Base Shield V2.0 for Arduino-to-Arduino compatible connector of FPB-RA0E1 kit. Attach the sensor module to the A0 and A1 connectors of this Base Shield with Grove cable. This connects the analog signal pins (AN005 and AN006) of RA0E1 MCU with the analog output pins of the two sensors. Refer to Figure 16.

- Light Sensor Module: Seeed Grove – Luminance Sensor
  Equipped with illuminance sensor APDS-9002, Operating voltage: 2.4~5V, Measurement range: 0–1000Lux
- Temperature Sensor Module: Seeed Grove – Temperature Sensor
  Equipped with NTC thermistor NCP18WF104F03RC, Operating voltage: 3.3~5V, Operating temperature range: -40 °C ~ 125°C

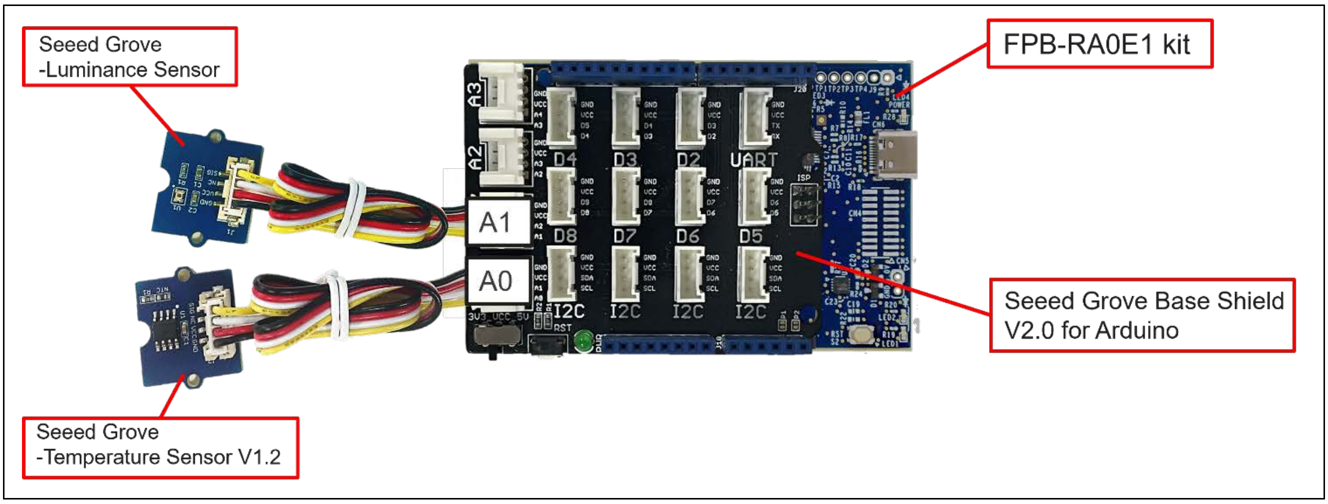


Figure 16 FPB-RA0E1 Sensor Connection

3.2.2 Connecting the USB-TTL Connector

Table 4 and Figure 17 show the connection pins to the USB-TTL connector. Connect Arduino-compatible connector of FPB-RA0E1 kit using jumper wires.

Table 4 . Pins to be Connected to the USB-TTL Connector

RA0E1 Pin	Terminal number on FPB-RA0E1	Header number on Arduino expansion bord	Typical Uses
P101/TXD0	J5-6	J20-6	Serial communication (transmission). Connect to the RXD pin of the USB-TTL connector.
VSS	GND	GND	Ground Connect to the GND pin of the USB-TTL connector.

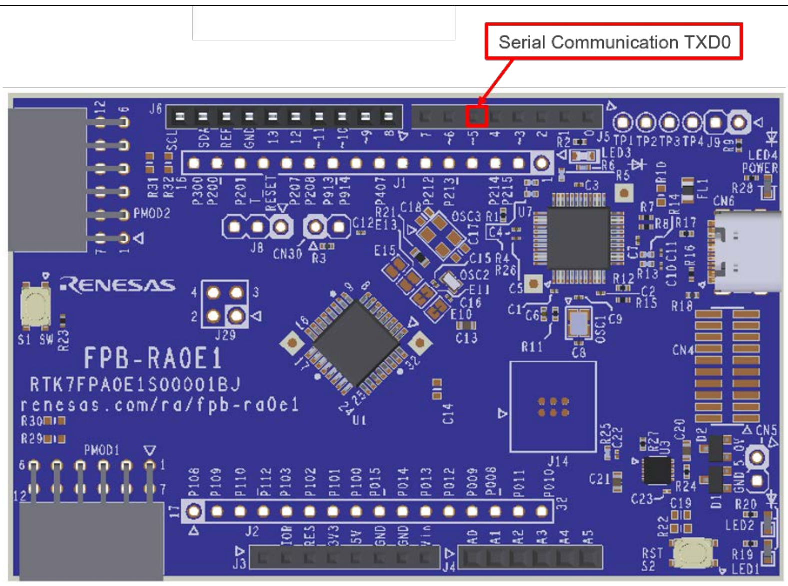


Figure 17 FPB-RA0E1 Connecting to the External Equipment

3.2.3 Data Communication Specifications

This application project performs data communication according to the following procedure and format.

3.2.3.1 Communication Data Format

- Transmission Format
  The following is the packet format/protocol of data transmitted by the FPB-RA0E1.
  Packet format:

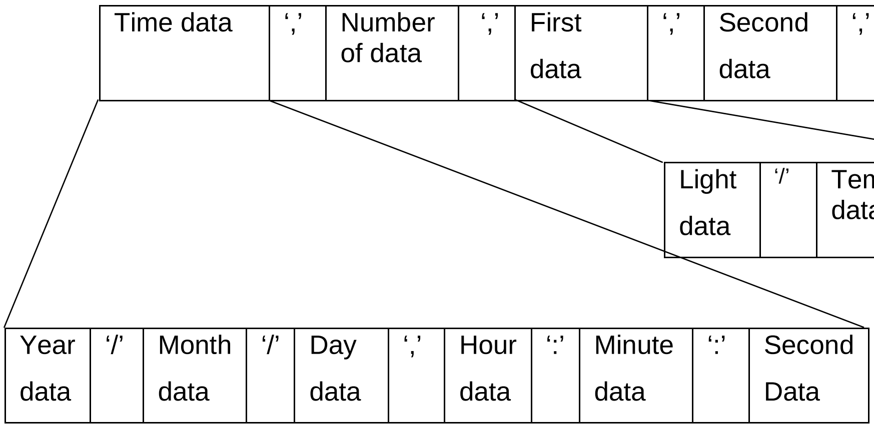
(a) Start code	(b) Attribute code	(c) Delimiter code	(d) Data length code	(e) Data	(f) End code	(g) Linefeed code

Description	Code	Length	Function
(a) Start code	'^'	1 byte	Start of the packet
(b) Attribute code	ASCII Code	1 byte	Represent a data output occurrence event '1': RTC alarm event '2': External IRQ event
(c) Delimiter code	‘,’	1 byte	Code delimiting code
(d) Data length code	ASCII Code	4 bytes	Indicates the length of the data section.
(e) Data	ASCII Code	Number of bytes specified in (d)	See the following Data Format
(f) End code	'$'	1 byte	End of the packet
(g) Linefeed code	‘\n’	1 byte	Linefeed

- (e) Data format

The format of the data part varies depending on the attribute code as shown below.

Data for up to 24 hours can be output. Every 24 hours, the output data is reset, and only the data accumulated after that time is output.



The data comprising the format of attribute codes 1 or 2 is generated as follows.

- The year/month/day/hour/minute/second data consists of 0 to 9 ASCII codes. The data are separated into 3 types of delimiters ‘/’ or ‘,’ or ‘:’. The year data is only lower 2 digits. Example: When the system transmits a time value “24/01/01,00:00:00”, it sets 0x32, 0x34, 0x2F, 0x30, 0x31. 0x2F, 0x30, 0x31, 0x2C, 0x30, 0x30, 0x3A, 0x30, 0x30, 0x3A, 0x30, 0x30.
- The number of data consists of 0 to 9 ASCII codes. The length is 2 bytes. Example: When the system transmits 10 data, it sets 0x31, 0x30.
- The light and temperature data are composed of 4-digit decimal number representing ADC read value. The data consists of 0 to 9 ASCII codes. The delimiter between the data is '/'. Example: When the system transmits data “2000/0300”, it sets 0x32, 0x30, 0x30, 0x30, 0x2F, 0x30, 0x33, 0x30, 0x30.

3.2.4 Input/Output Pins for Debugging

Table 5 and Figure 18 show the I/O pins for debugging this application project. The debug pins can be observed and toggled as shown below to cancel the low power modes and check the status of the modes.

Table 5. Debugging I/O Pins

RA0E1 Pin	FPB-RA0E1 Pins or Connecting Components	Typical Uses
P008	LED1	Turn on in Normal mode
P009	LED2	Turn on when program stops due to an error
P200/IRQ0	User switch S1	Manual cancellation of low power mode

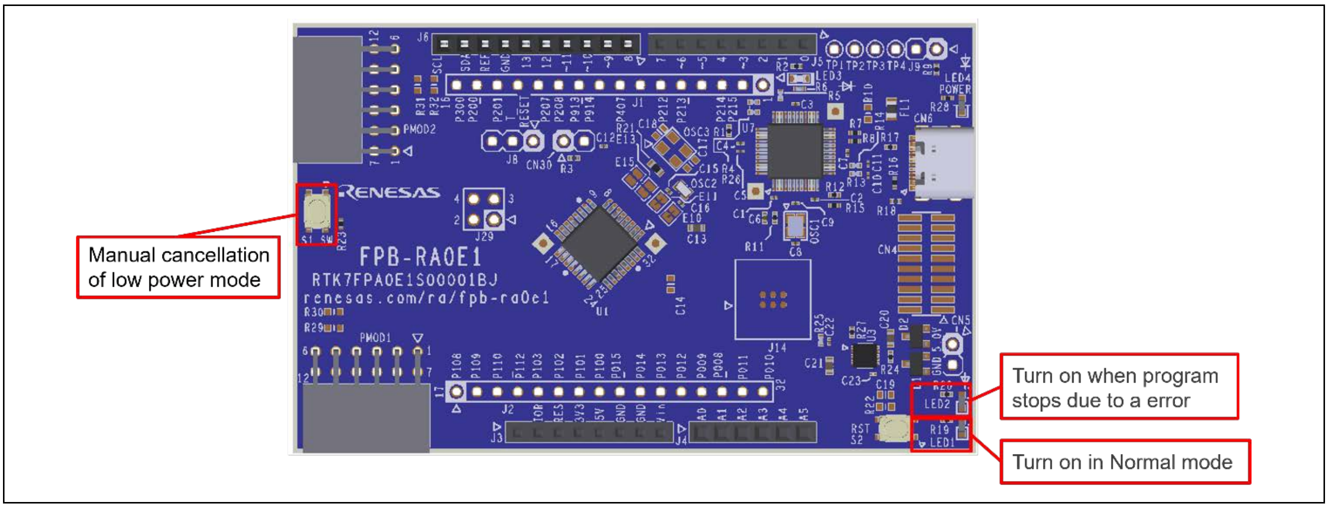


Figure 18 Debugging I/O Pins

3.3 Debugging

This application project implements pin input/output function for checking the low power mode status, fast cycle debug function, and disable sensor dependent process function.

3.3.1 Canceling the Low Power Mode by Pressing Down the User Switch

The user switch S1 of FPB-RA0E1 can be pressed to generate an IRQ0 interrupt and cancel the low power mode. Setting a breakpoint after canceling stops the program being executed, and each register and data can be checked.

3.3.2 Checking the Low Power Mode Status

In this application project, the LED1 is turned on during Normal mode (Table 6). To distinguish between Software Standby mode and Snooze mode, we recommend measuring the actual current consumption of MCU power supply pin. Refer to section 4.5 for how to evaluate the current consumption.

Table 6. Pin Output States in each Low Power Mode

Low Power Modes	P008/LED1
Normal mode	High
Software Standby mode, Snooze mode	Low

3.3.3 Fast Cycle Debug

To shorten the operation duration for easy debugging, the example project offers different implementations to reduce the execution cycle of data acquisition and data process. To enable this feature, define the `DEBUG_FAST_CYCLE` macro in the `app_common.h`. When this `DEBUG_FAST_CYCLE` macro is defined, the following process is enabled, and the data acquisition period is set every 30 seconds, and the data processing period is set every 24 minutes.

- Resetting the TML32 timer count (TML32)
- Changing the Added Value of the RTC Alarm Setting

3.4 Flowchart

This section describes the flowchart of the Low-Power Data Logger Application.

Figure 19 shows the overall flow of the application project.

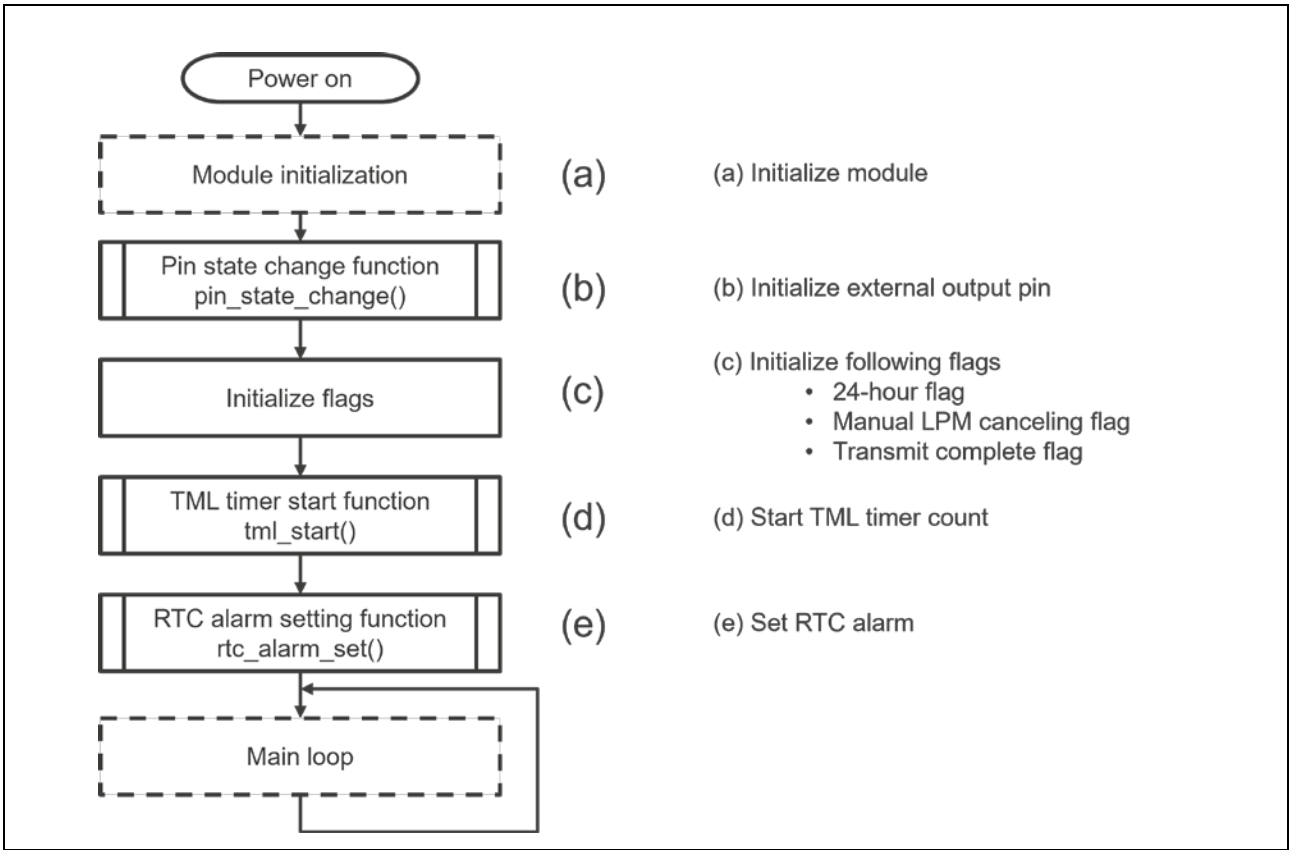


Figure 19 Overall flow

Figure 20 shows the module initialization processing flow.

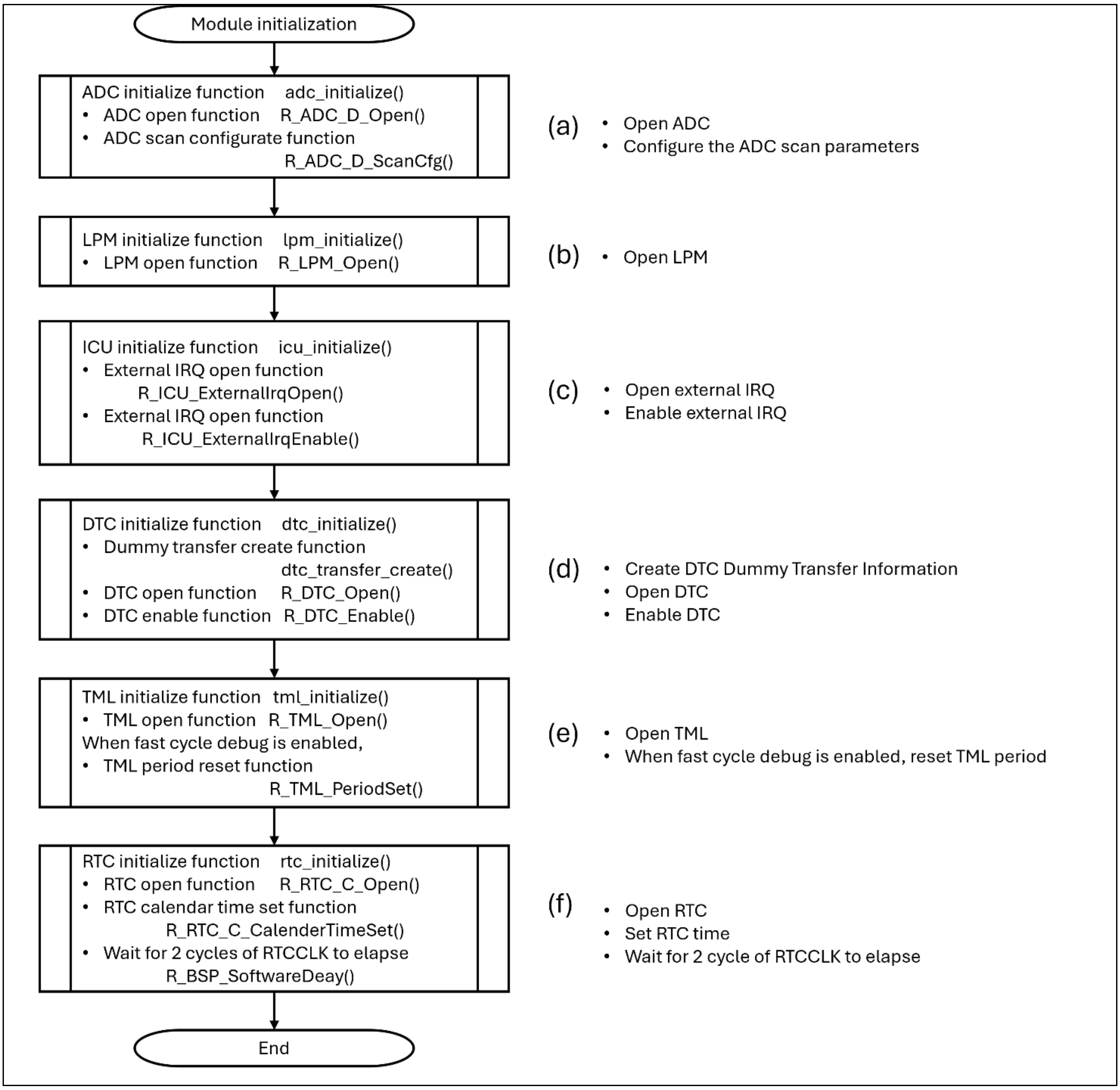


Figure 20 Module initialization Processing

Figure 21 shows the main loop processing flow.

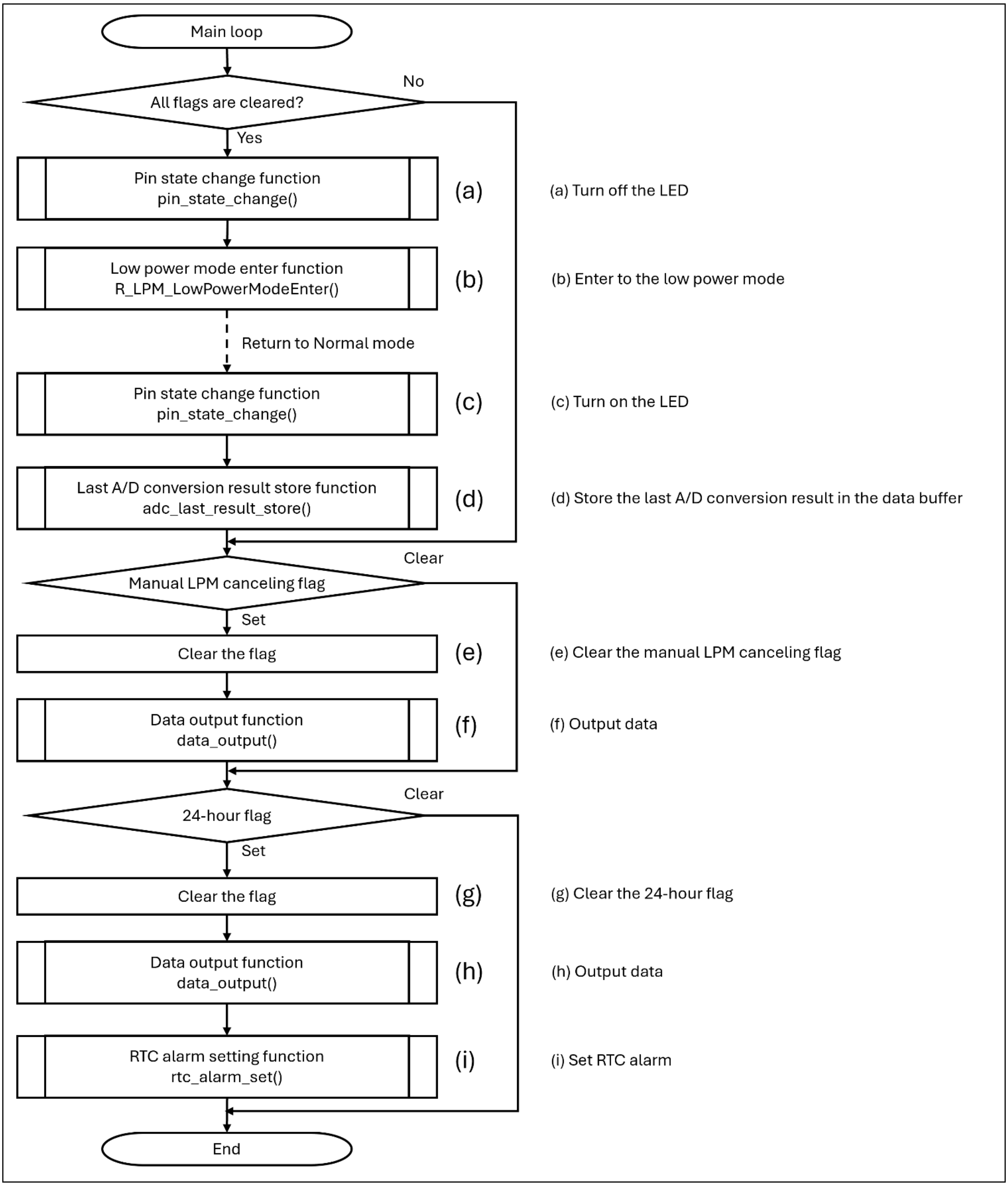


Figure 21 Main Loop Processing

Figure 22 shows the flow of data output processing.

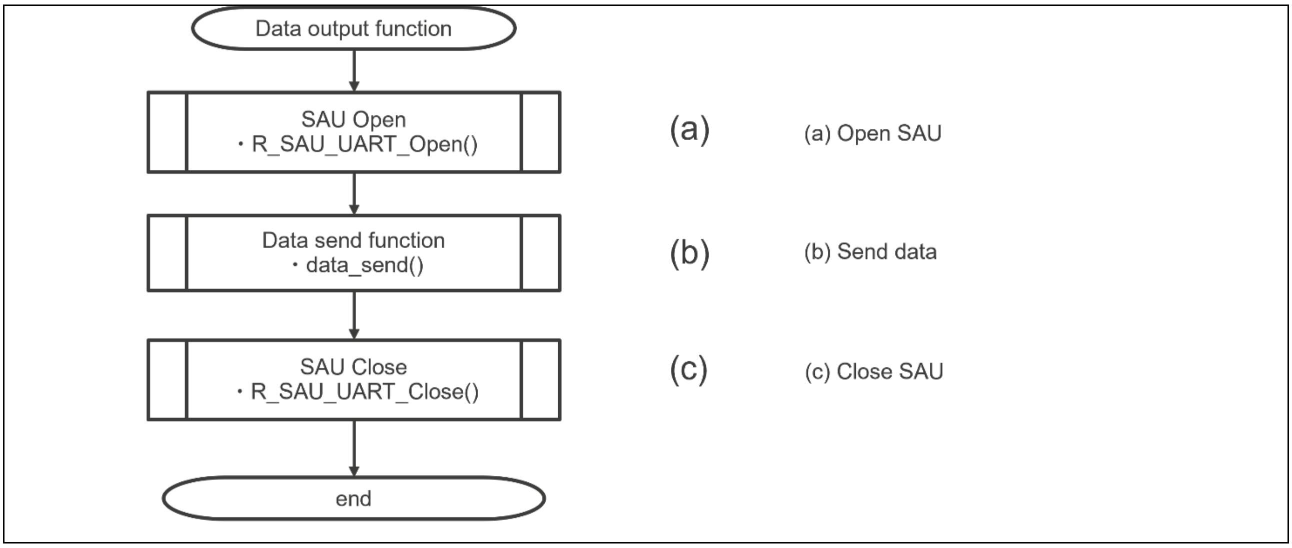


Figure 22 Data output Processing

Figure 23 shows the flow of send data processing.

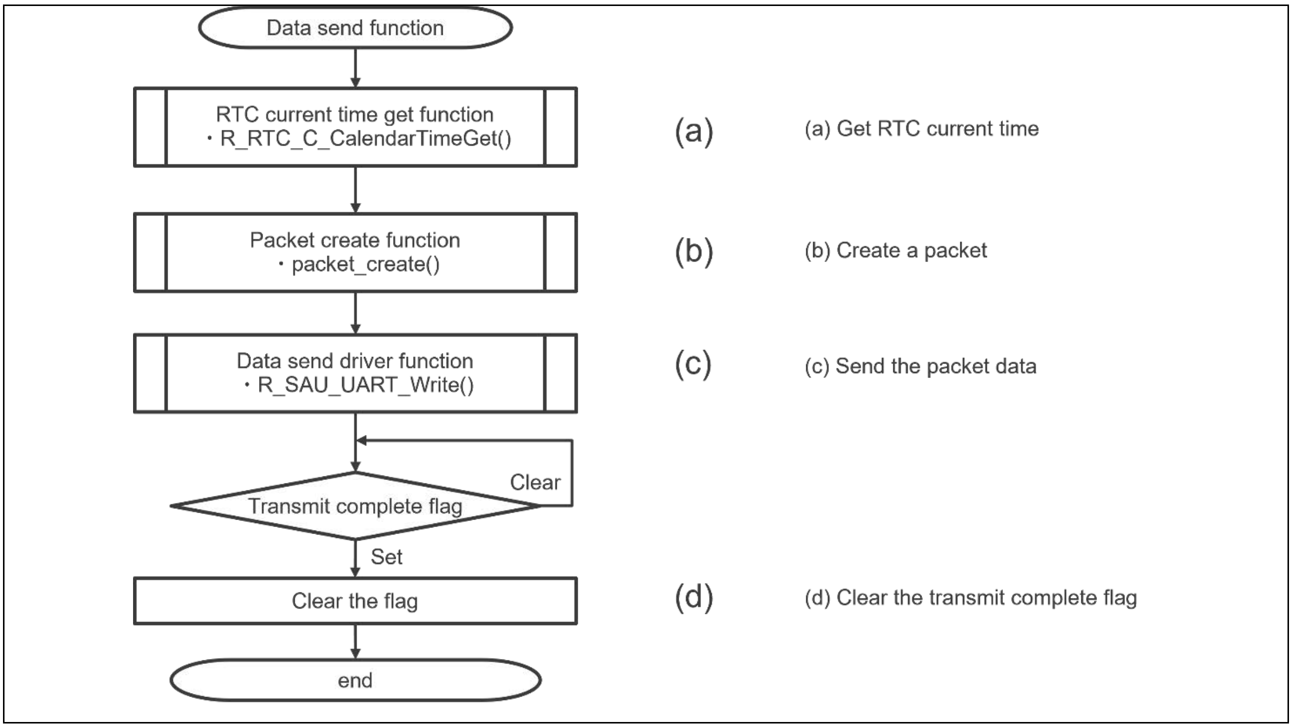


Figure 23 Data send Processing

Figure 24 shows the flow of RTC alarm interrupt processing.

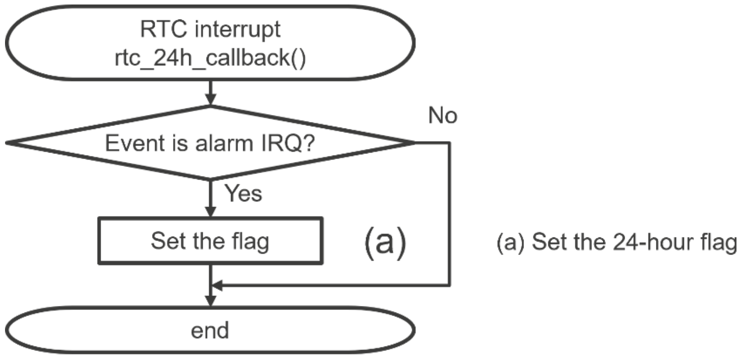


Figure 24 RTC Alarm Interrupt Processing

Figure 25 shows the flowchart for External IRQ Interrupt Processing.

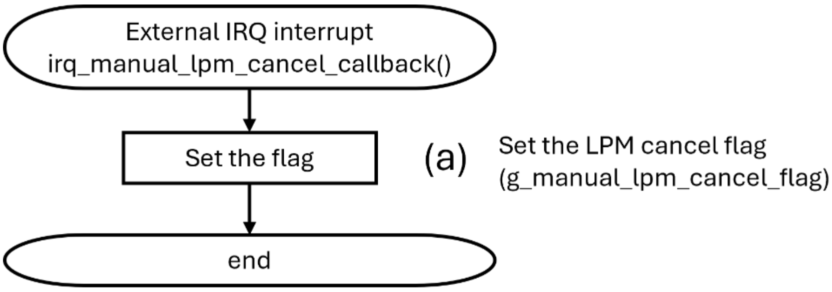


Figure 25 LPM canceling Interrupt Processing

Figure 26 shows the flowchart for SAU Interrupt Processing.

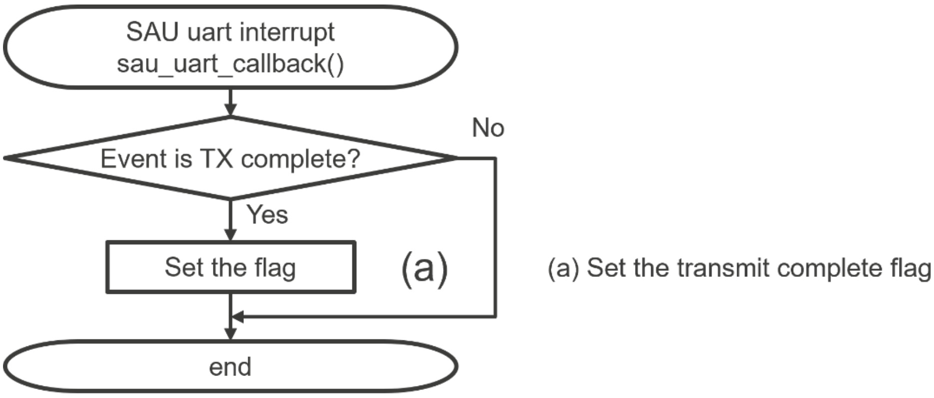


Figure 26 SAU Interrupt Processing

4. Evaluating Applications

4.1 Import and Building a Project

To build an application project with e[2] studio IDE, proceed as follows:

1. Launch e[2] studio IDE.
2. Select any workspace in **Workspace launcher**.
3. Close **Welcome** window.
4. Select **File** > **Import**.
5. Select **Existing Projects into Workspace** from the **Import** dialog box.
6. Select archive file.
7. Select the project you want to import and click **Finish**.
8. Open configuration.xml and click **Generate Project Content** in the **Configurator** window.
9. Select **Project** > **Build Project**.

4.2 Script Settings for Low Power Mode Debugging

Change the debug settings to debug applications using low power modes. If you use Segger J-Link emulator to debug low power modes for RA devices, the default debug configuration does not allow debugging. If the application tries to enter Software Standby mode, the connection between the CPU and the IDE is disconnected and the IDE debug session may be terminated. You can continue to debug in low power modes by specifying a debug script and configuring low power handling option as shown in Figure 27.

This debug script is provided to develop applications. Therefore, **please note that accurate Icc values cannot be measured when measuring the current consumption Icc**. Disable OCD (debug function) and execute a power on reset when measuring the current.

The script file is included with this application project and can be used by specifying it in the debug settings.

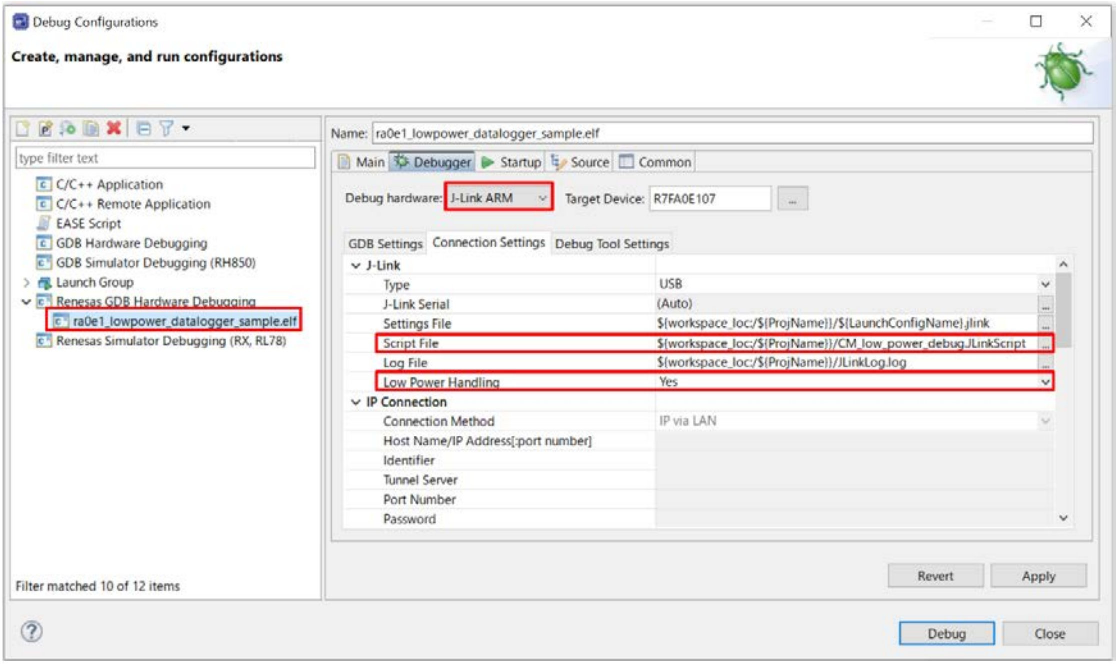


Figure 27 Debug Configuration

4.3 Download and Debug a Project

To download and debug an application project using e[2] studio and onboard J-Link debugger, proceed as follows:

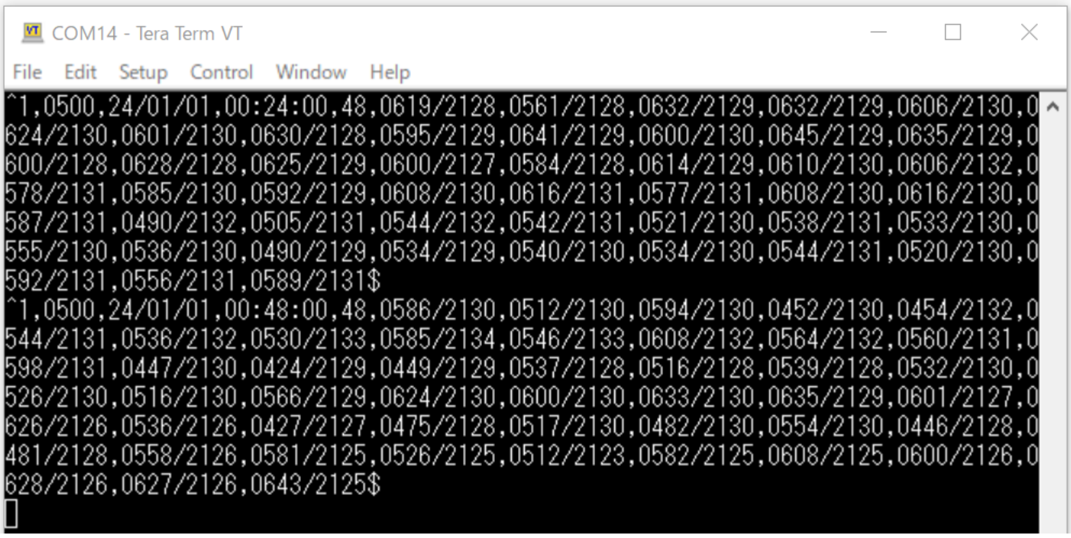
1. Connect the connector CN6 of FPB-RA0E1 and PC using USB cable.
2. Open the **Debug Configuration** window and confirm the low power mode related settings.
   For more detail, refer to the section 4.2.
3. Clicking **Debug**

4.4 Procedure for Checking the Operation of the Low-Power Data Logger

To check the operation of the low-power data logger, proceed as follows:

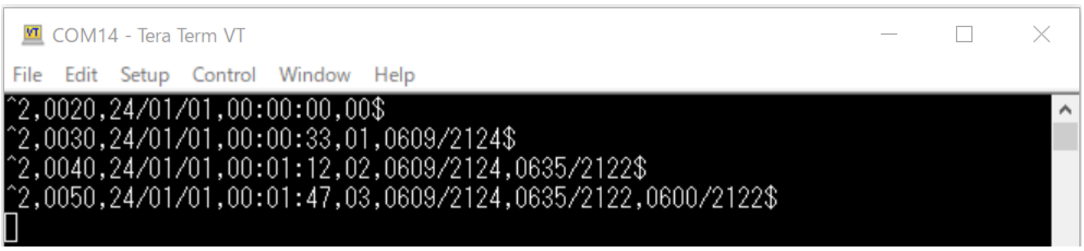
1. Connect the FPB-RA0E1 and PC using USB - TTL connector and jumper wires. Refer to section 3.2.2 for detailed connections.
2. Launch terminal software on the PC and change the settings as follows:
   - Transfer speed: 115,200 bps
   - Data length: 8 bits
   - Parity: None
   - Stop-bit length: 1 bit
3. Power on the kit.
4. Confirm the operation.

This application project performs data output every 24 hours (every 24 minutes in fast cycle debug is enabled) (See Figure 28). The output data can be viewed on terminal software on the PC. In addition, pressing the user switch S1 of the FPB-RA0E1 kit will output the progress of accumulated measurement data at any time (See Figure 29 and Figure 29).



Note: A fast cycle debug is enabled

Figure 28 FPB-RA0E1 Example of Data Output Triggered by 24 Hours Passing



Note: A fast cycle debug is enabled

Figure 29 FPB-RA0E1 Example of Data Output Triggered by Pressing the User Switch

4.5 Evaluating the Current Consumption

FPB-RA0E1 includes a resistor R3 and test connector CN30 (not fitted) for measuring the current of the MCU power of 3.3 V. When measuring the current amount of the MCU, remove resistor R3 and connect the measuring equipment to the test connector.

See Figure 30 and Figure 31 for the location of R3 and CN30.

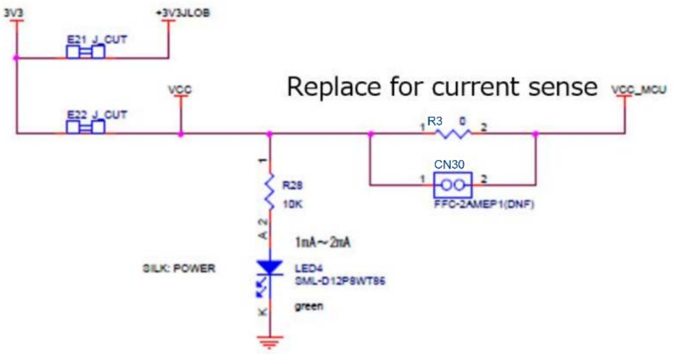


Figure 30 MCU Current Measuring Points (Circuit Diagram)

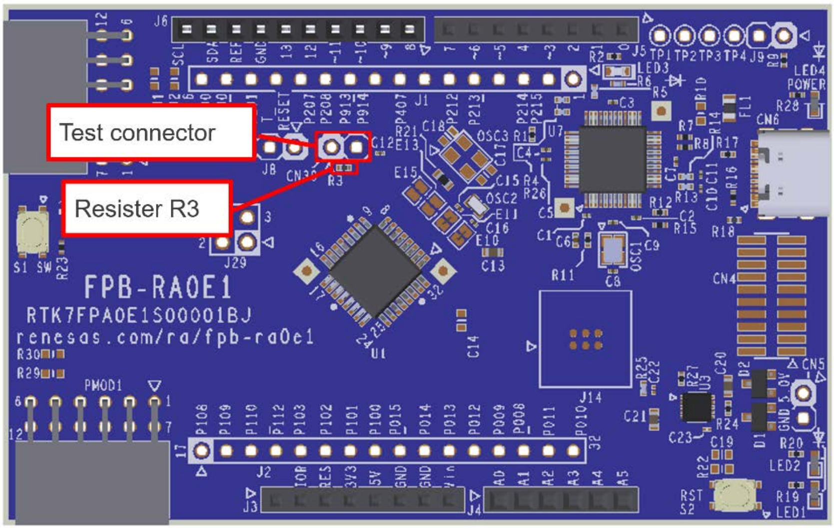


Figure 31 MCU Current Measuring Points

5. References

- Renesas FSP User's Manual — renesas.github.io/fsp
- Renesas RA MCU datasheet — Select the relevant MCUs from the www.renesas.com/ra
- Example Projects — github.com/renesas/ra-fsp-examples

6. Website and Support

Visit the following URLs to learn about key elements of the RA family, download components and related documentation, and get support.

RA Product Information	www.renesas.com/ra
RA Product Support Forum	www.renesas.com/ra/forum
RA Flexible Software Package	www.renesas.com/FSP
Renesas Support	www.renesas.com/support

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Apr 30, 24	-	Initial version

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.