Renesas Synergy™ Platform

Checking Analog Inputs in Sleep State in S3A7

Introduction

This application note describes how to implement an application for reading analog inputs through a 14-bit ADC model during MCU Sleep/Standby. It also describes how to set up different ADC comparison functions for staying in LPM or returning to the normal running mode. The application program is created on DK-S3A7 (v1.1/v2.0) Renesas Synergy development boards, using the Renesas e^2 studio ISDE and Renesas Synergy Software Package (SSP).

Goals and Objectives

The goal of this application note is to illustrate a general framework for reading analog signals during Sleep/Standby modes, waking up the MCU, or keeping it in sleep state, with different ADC conditions.

The example application shows how to use a 14-bit ADC to read analog signals that are generated by adjusting the potentiometer when the Synergy S3A7 MCU is in Snooze mode, when only some select modules run, and the CPU is not active. This includes showing how to set different events for canceling the Snooze mode, to either return to the Software Standby mode, or return to the Normal mode.

You can easily substitute other analog sources such as light sensors, or motion sensors for the potentiometer, and re-define the ADC window functions for different waking up conditions to create more complex applications.

Prerequisites

As a user, you are assumed to have some experience with Renesas e^2 studio ISDE and SSP. For example, before you perform the procedure in this application note, you should follow the procedure in your board's Quick Start Guide to build and run the Blinky project. By doing so, you will become familiar with e^2 studio and the SSP and ensure that the debug connection on your board is functioning properly.

Required Resources

The example application targets Renesas Synergy S devices. To build and run the application, you will need:

- Renesas Synergy DK-S3A7 board (v2.0)
- A PC running Microsoft® Windows® 7, with the following Renesas Synergy software installed:
  - e^2 studio (ISDE) v7.3.0 or later
  - Synergy Software Package (SSP) v1.6.0 or later
  - IAR Embedded Workbench® for Renesas Synergy™ v8.23.3 or later
  - Synergy Standalone Configurator (SSC) v7.3.0 or later

You can download the required Renesas software from the Renesas Synergy Gallery (www.renesas.com/synergy/software).

Contents

1. Overview

This design demonstrates how to make a typical IoT low-power sensor hub where the CPU is in Sleep mode, but it checks an environmental sensor such as a motion, temperature, or smoke sensor periodically without waking up the CPU, until a sensor value meets a predefined condition. Figure 1 shows the DK-S3A7 board.

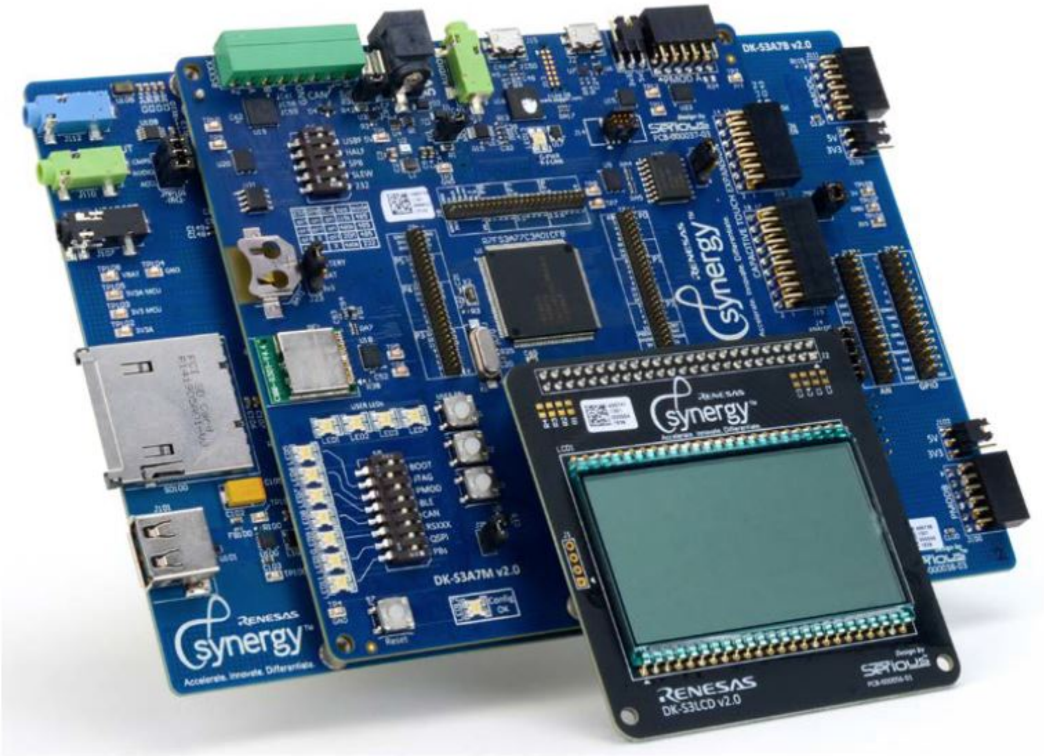


Figure 1. DK-S3A7 (v2.0) board

Figure 2. Checking the environment sensor during Sleep mode

This application uses the Low Power Modes (LPMs) and the ADC module in the Synergy S3A7 MCU.

The CPU is put in a sleep state using the Software Standby mode, one of the LPMs, where an RTC timer operating at 32 kHz generates an interrupt periodically.

Each RTC PRD interrupt puts the MCU in Snooze mode, another LPM, where the CPU is still in sleep state, but a 14-bit ADC can be triggered.

Using an ADC compare function window, different results or events can be set as conditions. For example, ADC140_WCMPM and ADC140_WCMPUM, which are used to decide if the CPU is still in sleep state, or is waking up to a running state.

Figure 3 shows a state diagram with the MCU states and LPM transitions used in this application.

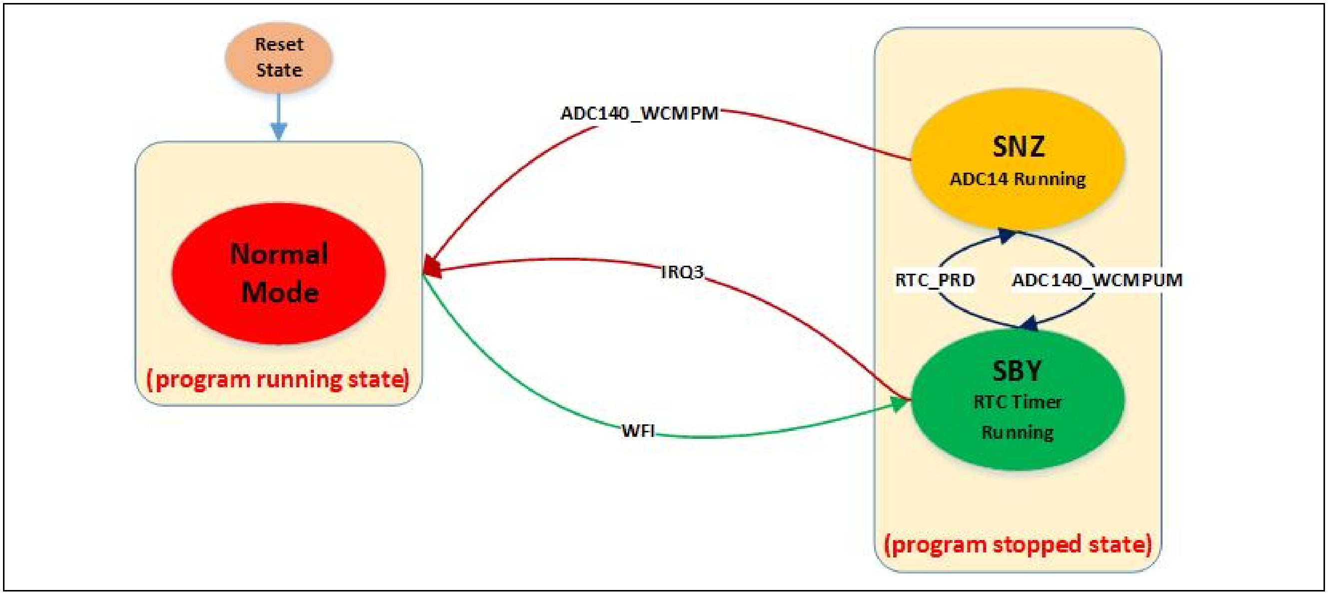


Figure 3. MCU states and LPM transitions used in this application

2. Peripheral Modules

This section briefly describes the Low Power Modes (LPMs), Event Link Controller (ELC), and ADC modules in Synergy S3A7 MCU, and how to configure them to achieve the expected functionality.

2.1 LPMs for Configuring Different Sleep States

This application shows how to reduce the MCU power consumption as much as possible, while checking the analog sensors, and adjusting the MCU states. The S3A7 has three low power modes: Sleep mode (SLP), Software Standby mode (SBY), and Snooze mode (SNZ). Each mode has a different configuration for clock sources, available peripheral modes, and power consumption, among other things.

2.1.1 LPMs to be used

Table 11.2 in the *S3A7 User's Manual* specifies the conditions to transition to LPMs.

The LPMs can be described as follows:

- Sleep mode (SLP):
  The CPU stops operating, but the contents of the internal registers are retained. Other peripheral functions do not stop. The CPU can be woken up by an interrupt, RES pin reset, a power-on reset, a voltage monitor reset, an SRAM parity error reset, a reset caused by an IWDT, or a WDT underflow.
- Software Standby mode (SBY):
  The CPU and most of the on-chip peripherals and oscillators stop. However, the contents of the CPU internal registers, SRAM data, the states of on-chip peripheral functions, and the I/O ports are retained. Software Standby mode allows a significant reduction in power consumption because most of the oscillators stop in this mode. Only those interrupts specified in the Wake Up Interrupt Enable register (WUPEN) can cancel the Software Standby mode. According to the operating and standby currents given in Table 50.12 in the *S3A7 User's Manual*, the SBY has the lowest power consumption among the three LPMs, and so it is used in this application.
- Snooze mode (SNZ):
  Similar to Sleep mode, some peripheral modules can operate without waking up the CPU. For example, the 14-bit ADC converter can read the analog sensors and check if some predefined conditions such as the ranges of ADC values are satisfied or not, and then stay in the LPM, or wake up the CPU. The conditions to enter the SNZ mode, called Snooze requests, are specified in Table 1. The conditions to transition from SNZ to SBY, called Snooze end conditions, are listed in Table 2.

Table 1. Events that cause a transition from SBY to SNZ mode

Snooze request	Control register	Bit
PORT_IRQn (n = 0 to 15)	SNZREQCR	SNZREQENn (n = 0 to 15)
KEY_INTKR	SNZREQCR	SNZREQEN17
ACMP_LP0	SNZREQCR	SNZREQEN23
RTC_ALM	SNZREQCR	SNZREQEN24
RTC_PRD	SNZREQCR	SNZREQEN25
AGT1_AGTI	SNZREQCR	SNZREQEN28
AGT1_AGTCMAI	SNZREQCR	SNZREQEN29
AGT1_AGTCMBI	SNZREQCR	SNZREQEN30
RXD0 falling edge	SNZCR	RXDREQEN (RXDREQEN bit must not be set to 1 except in asynchronous mode.)

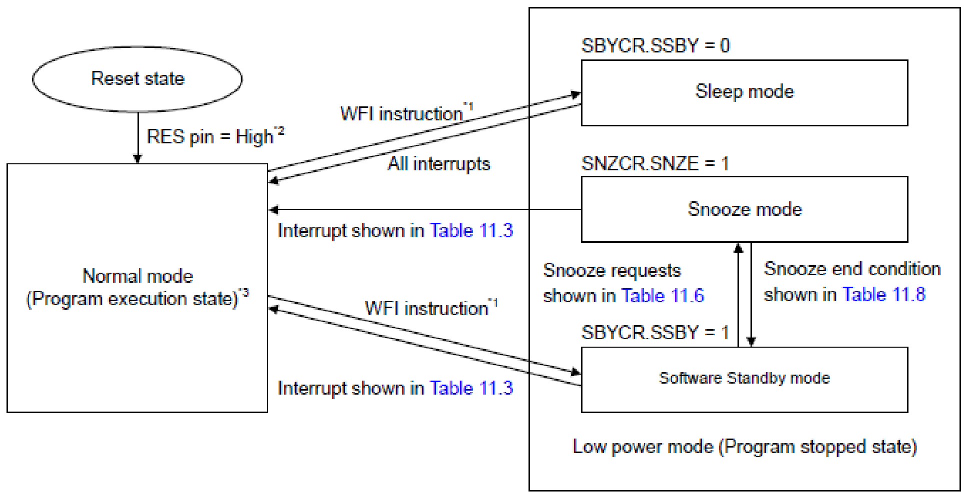
Table 2. Events that can end SNZ mode

<table>
<tr><th rowspan="2">Operating module when a Snooze end request occurs</th><th colspan="2">Snooze end request</th></tr>
<tr><th>AGT1 underflow</th><th>Other than AGT1 underflow</th></tr>
<tr><td>DTC</td><td rowspan="3">The MCU transfers to Software Standby mode after all the modules listed to the left complete operation</td><td rowspan="4">The MCU transfers to Software Standby mode after all the modules to the left complete operation</td></tr>
<tr><td>ADC140</td></tr>
<tr><td>CTSU</td></tr>
<tr><td>SCI0</td><td>The MCU transfers to Software Standby mode immediately after a Snooze end request is generated</td></tr>
<tr><td>Other than above</td><td colspan="2">The MCU transfers to Software Standby mode immediately after a Snooze end request is generated</td></tr>
</table>

Note: If the DTC is used to activate the ADC140, CTSU, or SCI, the MCU transitions to Software Standby mode after a Snooze end request is generated.

2.1.2 Possible power mode transitions

Figure 4 shows available transitions between Normal mode and LPMs. The conditions or interrupt sources for triggering such transitions are specified in the *S3A7 User's Manual*. Selecting different conditions create different applications. Figure 4 shows one such application of using the ADC140 as a Snooze end condition.



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.

Figure 4. Possible LPM transitions for S3A7 MCU

2.2 ELC for Triggering ADC without using CPU

To trigger an operation such as ADC in SNZ mode without waking the CPU, the Event Link Controller (ELC) can forward the event requests generated by various peripherals to some peripheral modules, where two sets of control registers, ELSEGR0, 1, and ELSRn should be set before entering SBY. For example, Figure 5 shows that, to trigger an ADC operation in SBY directly, you need to set the SYSTEM_SNZREG event in the ELSR0 register, which is the entry for the ADC14 peripheral.

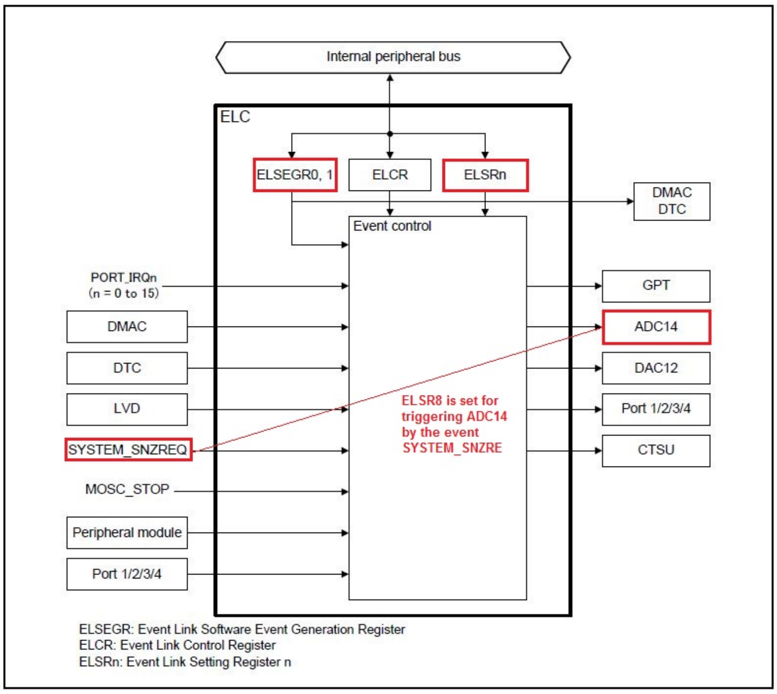


Figure 5. Selecting a proper source event to trigger a peripheral operation in ELC

Implementation using the SSP ELC driver API is as follows:

```
//set SYSTEM_SNZREQ to trigger ADC0
g_elc.p_api->linkSet(ELC_PERIPHERAL_ADC0, ELC_EVENT_LPM_SNOOZE_REQUEST);
```

2.3 ICU for Waking the CPU from LPM

Figure 6 shows the Interrupt Controller Unit (ICU) controling which event signals link to the Nested Vector Interrupt Controller/ Data Transfer Control (NVIC/DTC) module and wakes the CPU from the different LPMs.

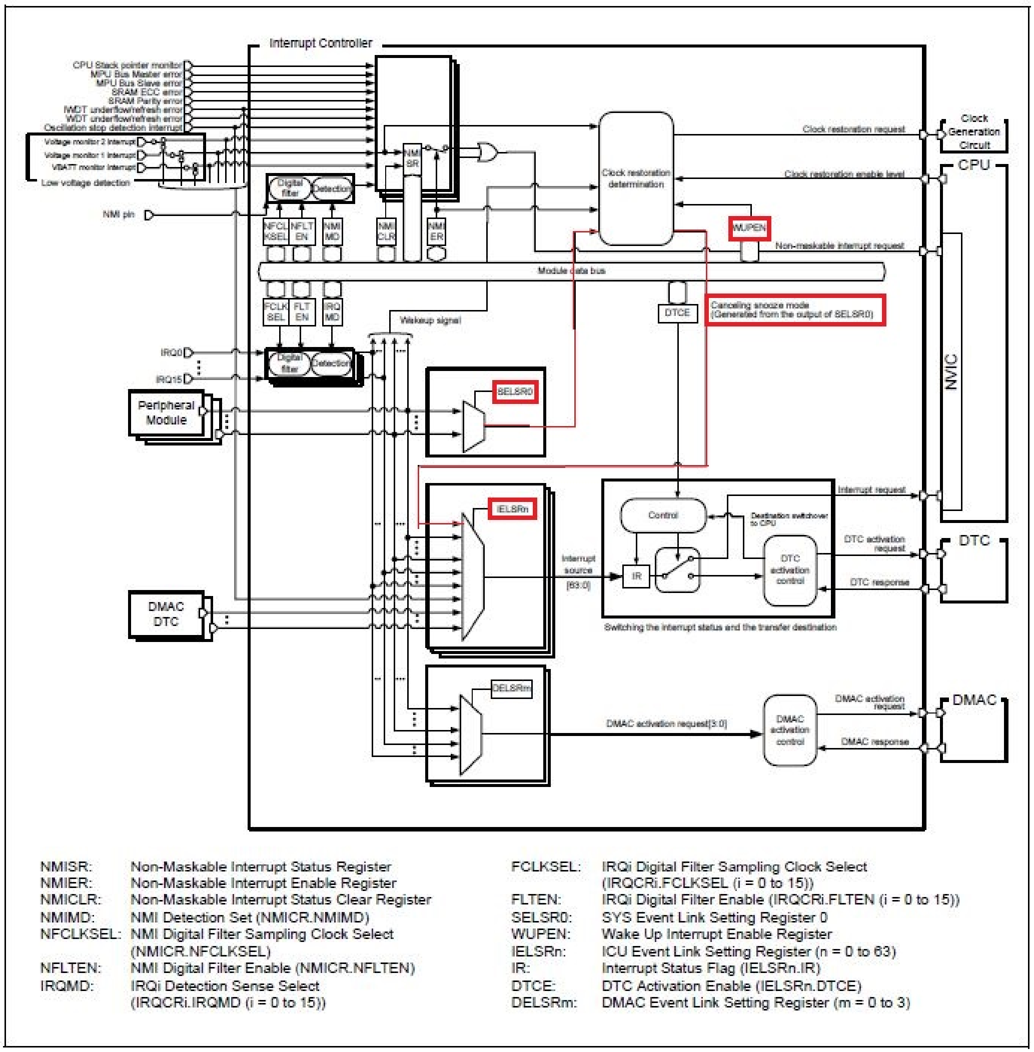


Figure 6. ICU functional blocks for waking up the LPM

Actions while waking up from different LPMs are as follows:

- From Sleep mode, return is initiated by non-maskable interrupts or any other interrupt source.
- For Software Standby mode, return is initiated by non-maskable interrupts. Interrupts can be selected in the WUPEN register.
- For Snooze mode, return is initiated by non-maskable interrupts. Interrupts can be selected in SELSR0 and WUPEN registers. The SELSR0 register selects events that wake up the CPU from Snooze mode. Two registers are set as follows:
  - Assign an event as listed in Table 14.4 of the *S3A7 User's Manual* under "Canceling Snooze", set in the SELSR0 register as in Table 11.3. For example, ADC140_WCMPM (0x4F) is used to wake up the CPU from the SNZ.
  - Assign an event ICU_SNZCANCEL (0x2D), set in IELSRn.ISEL, to enable an SELSR0 event interrupt.

2.4 ADC for Setting Analog Conditions for an LPM transition

The S3A7 MCU has a 14-bit A/D Converter (ADC14) unit, which can be used for reading up to 28 analog channels and the on-chip temperature sensor/ internal reference voltage. On the DK-S3A7 v2.0 or v1.1 board, a potentiometer is already connected to the analog channel 13. You can enable this channel to the ADC14, and easily generate different values by turning the potentiometer wheel. Figure 7 shows the potentiometer connections to the analog channel 13.

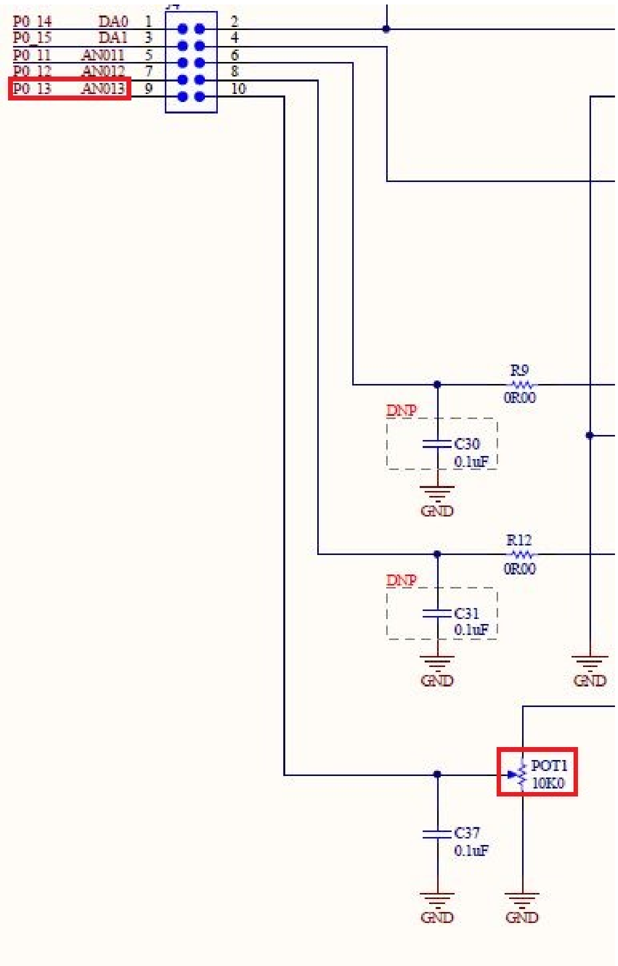


Figure 7. Potentiometer connection to the analog channel 13 on DK-S3A7

2.4.1 Scanning the analog channels

The ADC14 module supports three different scanning modes:

- Single scan, which goes through the analog channels in ascending order of channel number
- Continuous scan, which performs a single scan continuously
- Group scan, which partitions the analog channels into two groups (group A and group B), then performs a single scan for each group. This design only reads channel 7 once each time, and so the single scan mode is set for ADC14.

2.4.2 Starting ADC operation

Figure 8 shows the ADC14 control logic, where there are two different kinds of triggers — a synchronous event from ELC, and an asynchronous event from an input pin (ADTRG0).

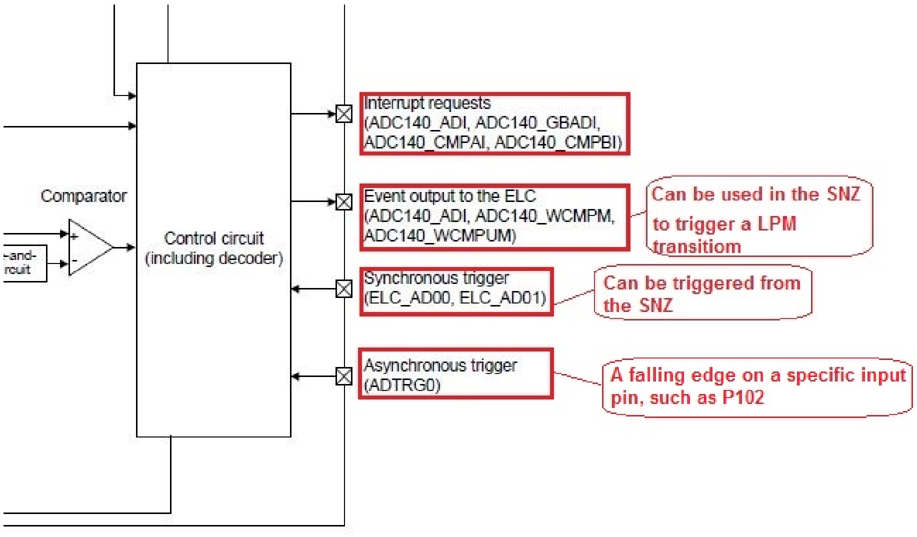


Figure 8. Input and output events for ADC14

As a synchronous trigger, ELC_AD00 can be considered as an output from the ELC and generated when an event SYSTEM_SNZREQ is provided. Such a relation is specified in the ELC.ELSR8 register. There is a critical link between the LPM and ADC module in this design.

2.4.3 Setting a compare function in the ADC

The ADC values can be compared with some predefined windows or ranges in each analog channel. Two different events are generated:

- ADC140_ADI: an interrupt for completing all selected channel conversions
- ADC_WCMPM: when the ADC value matches the comparison conditions on window A/B, or
- ADC_WCMPUM: when the ADC value does not match the comparison conditions on window A/B.

For example, for a compare function window A, there are four different comparison results by setting different values on two control register bits, as shown in Table 3.

Table 3. Four different compare conditions for ADC window A/B

Enable window A (ADCMPCR.WCMPE)	Compare conditions (ADCMPLR0.CMPLCHAn)	
Disabled (0)	Larger than the lower boundary (0)	ADCMPDR0 value > A/D-converted value
	Smaller than the lower boundary (1)	ADCMPDR0 value < A/D-converted value
Enabled (1)	Out of range (0)	A/D converted value < ADCMPDR0 value, or ADCMPDR1 value < A/D-converted value
	Within range (1)	ADCMPDR0 value < A/D-converted value < ADCMPDR1 value

A suitable range is selected for the potentiometer by setting the following ADC compare control register bits, to check if the ADC values match a condition, such as falling within the range of 12,000 to 16,000.

```
//--- Set the Window_A ------
      //Select a channel for Window_A Comparison
      do {
         R_S14ADC->ADCMPANSR0_b.CMPCHA13 = 0x1;
      } while (!(R_S14ADC->ADCMPANSR0_b.CMPCHA13));

      //Set the Window_A Lower boundary
      do {
         R_S14ADC->ADCMPDR0_b.ADCMPDR0 = POT_RD_LOW_A;
      } while (R_S14ADC->ADCMPDR0_b.ADCMPDR0 & ~POT_RD_LOW_A);

      //Set the Window_A Upper boundary
      do {
         R_S14ADC->ADCMPDR1_b.ADCMPDR1 = POT_RD_HIGH_A;
      } while (R_S14ADC->ADCMPDR1_b.ADCMPDR1 & ~POT_RD_HIGH_A);

//Set the Window_A comparison condition
      do {
         R_S14ADC->ADCMPLR0_b.CMPLCHA13 = 0x1;
      } while (!(R_S14ADC->ADCMPLR0_b.CMPLCHA13));

//Enable Window_A operation
      do {
         R_S14ADC->ADCMPCR_b.CMPAE = 0x1;
      } while (!(R_S14ADC->ADCMPCR_b.CMPAE));

//Enable Window_A interrupt for meeting the condition, ADC140_CMPAI
      do {
         R_S14ADC->ADCMPCR_b.CMPAIE = 0x1;
      } while (!(R_S14ADC->ADCMPCR_b.CMPAIE));

//Enable Window_A/B Comparison Function
      do {
         R_S14ADC->ADCMPCR_b.WCMPE = 0x1;
      } while (!(R_S14ADC->ADCMPCR_b.WCMPE));
```

Once the condition is matched, it generates an ADC140_WCMPM event, which is already set in the ICU as an event to wake up the CPU from SNZ.

```
//Enable Window_A/B composite condition
R_S14ADC->ADCMPCR_b.CMPAB = 0x0; //output ADC140_WCMPM
------------------------------------
//set ADC140_ADI as the event of ending SNZ
R_ICU->SELSR0_b.SELS = EVENT_TO_CANCEL_SNZ; //ADC140_WCMPM (0x4F) to cancel
SNZ
```

3. Application Implementation

Figure 9 shows the algorithm used that summarizes the setup steps and provides some usages information about this design on a DK-S3A7 (v2.0) board.

3.1 Algorithm

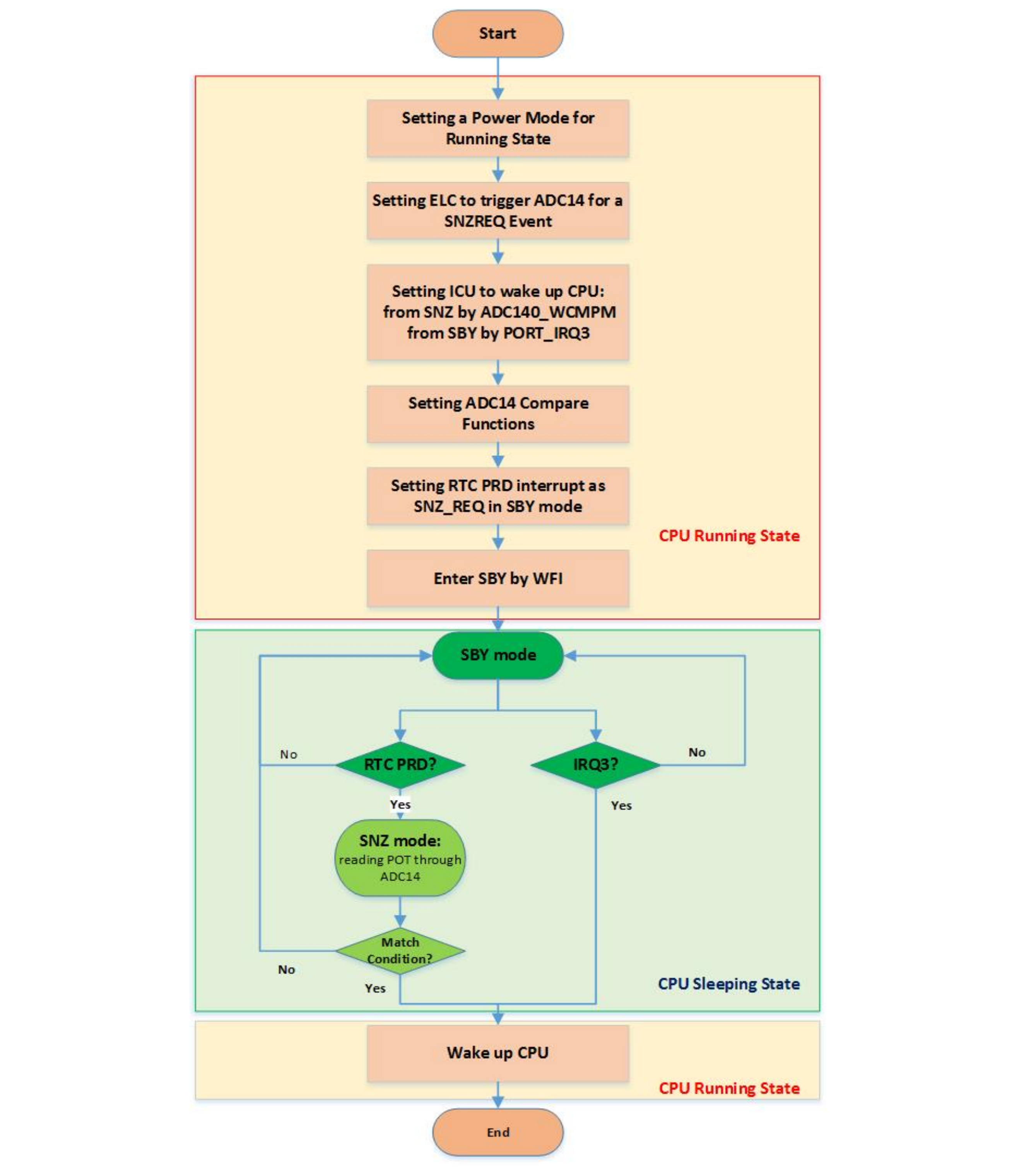


Figure 9. Algorithm used for checking ADC conditions in the Snooze mode

3.2 User Interface

The user interface for this design on DK-S3A7 v2.0 board is configured as follows:

- Debugging and programming the MCU
  - Connect a USB cable to the micro USB port (J15) from your host PC USB port.
- Switch function
  - Press switch S1 to enter Software Standby mode (SBY).
  - Turn POT1 to generate different analog values, which may wake up the MCU from Snooze mode (SNZ) if the value falls within the predefined range of 10,000 to 16,000.
  - Press switch S3 to wake up the MCU from SBY.
  - Press switch S7 to reset the application for a new round of testing after waking up the MCUfrom SBY or SNZ.

Note: On main board, in DIP switch S5, PBs should be in ON position for push button S1 and S3 to work.

- LED function
  - After the board is powered on or S3 is pressed, LED1 and LED2 will turn on.
  - When the MCU is in a low power mode, such as SBY/SNZ, the LED1 is off. It is on when the MCU is in normal mode. The LED1 will be blinking before the MCU changes state.
- Reset the board
  - A full power-on reset cycle, by unplugging and plugging back the power supply, is required for operating the LPM, therefore disconnecting the JTAG/J-Link from the Arm DAP (Debug Access Port).

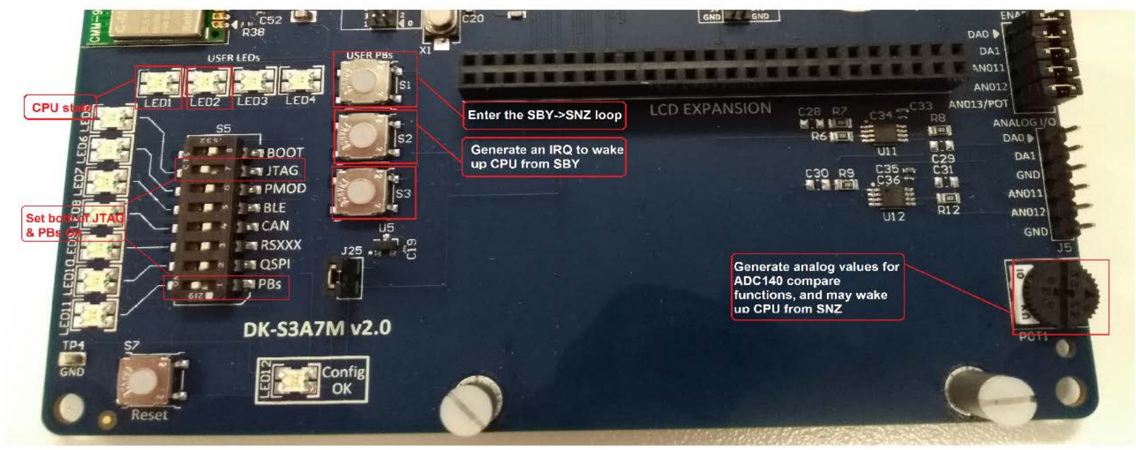


Figure 10. User interface for this design

4. Importing the Project into e^2 studio

See the *Renesas Synergy™ Project Import Guide* (r11an0023eu0121-synergy-ssp-import-guide.pdf) in the package for instructions on importing the project into e^2 studio to build and run the project.

5. Conclusion

This application note demonstrates a general platform using a predefined logic condition on the ADC compare functions to decide the LPM mode transitions of Synergy S3A7 MCU, such as staying in the sleep state, or waking up the CPU. The following configurations are selected as examples:

- The potentiometer, POT1, on the DK-S3A7 board is selected to generate different analog values.
- An ADC value range of 10,000 to 16,000 is set as the compare functions on both of ADC Window A/B.
- A 2-second RTC periodic interrupt enables an LPM transition from SBY to SNZ, where the ADC reads and checks the condition defined above, without activating the CPU. If the ADC value lies within the range, the CPU wakes up to execute high performance tasks.
- If the conditions do not match, that is, if the potentiometer value is out of the defined range, the MCU returns to the SBY mode to keep the power consumption at the lowest value.

- If the condition matches, that is, if the potentiometer value is within the defined range, the MCU wakes up and is ready for any high-performance tasks. The ADC value range for the potentiometer is defined between 10,000 and 16,000. To verify the application is working, turn the potentiometer to one end and verify the results provided. If the CPU is waking up, reset the board and try the other end of the potentiometer to verify the results.

This example is implemented with the Synergy SSP v1.4.0 or later and some bare-metal code, which is completely integrated with the next release of the SSP packages.

You can make further extensions to other analog sensors through AN011 and AN012 on the DK-S3A7 board. You can also replace the existing High-speed mode with other power modes such as the Middle-speed, Low-speed, or Low-voltage modes, and experience complex power saving strategies supported by the Synergy MCUs.

Website and Support

Visit the following vanity URLs to learn about key elements of the Synergy Platform, download components and related documentation, and get support.

Synergy Software	www.renesas.com/synergy/software
Synergy Software Package	www.renesas.com/synergy/ssp
Software add-ons	www.renesas.com/synergy/addons
Software glossary	www.renesas.com/synergy/softwareglossary
Development tools	www.renesas.com/synergy/tools
Synergy Hardware	www.renesas.com/synergy/hardware
Microcontrollers	www.renesas.com/synergy/mcus
MCU glossary	www.renesas.com/synergy/mcuglossary
Parametric search	www.renesas.com/synergy/parametric
Kits	www.renesas.com/synergy/kits
Synergy Solutions Gallery	www.renesas.com/synergy/solutionsgallery
Partner projects	www.renesas.com/synergy/partnerprojects
Application projects	www.renesas.com/synergy/applicationprojects
Self-service support resources:	
Documentation	www.renesas.com/synergy/docs
Knowledgebase	www.renesas.com/synergy/knowledgebase
Forums	www.renesas.com/synergy/forum
Training	www.renesas.com/synergy/training
Videos	www.renesas.com/synergy/videos
Chat and web ticket	www.renesas.com/synergy/resourcelibrary

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	May.24.16	—	Initial version
1.10	Oct.31.16	—	Migrated to SSP v1.2.0-b.1
1.11	Nov.30.16	—	Added IAR EW support
1.21	Feb.16.17	—	Migrated to SSP v1.2.0
1.22	Aug.04.17	—	Migrated to SSP v1.3.0
1.23	Sep.27.17	1, 12	Required resources of SSP version changed
1.24	Jan.18.18	—	Updated to SSP v1.3.3
1.25	Mar.16.18	—	Updated to SSP v1.4.0
1.26	Mar.06.19	—	Updated to SSP v1.6.0

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/.