RL78/G14

Event Link Controller operation in STOP mode

R01AN1397EG0100
Rev.1.00
Oct 29, 2012

Introduction

This application note demonstrates peripheral operation and communication using the Event Link Controller (ELC) whilst in STOP mode. This allows the device to operate in the lowest power mode for as long as possible in order to minimize power consumption. The hardware used for this application note is the RSKRL78/G14 – please refer to the RSKRL78/G14 specifications for hardware configuration.

Target Device

RL78/G14

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Contents <contents header>

1. Operation Overview

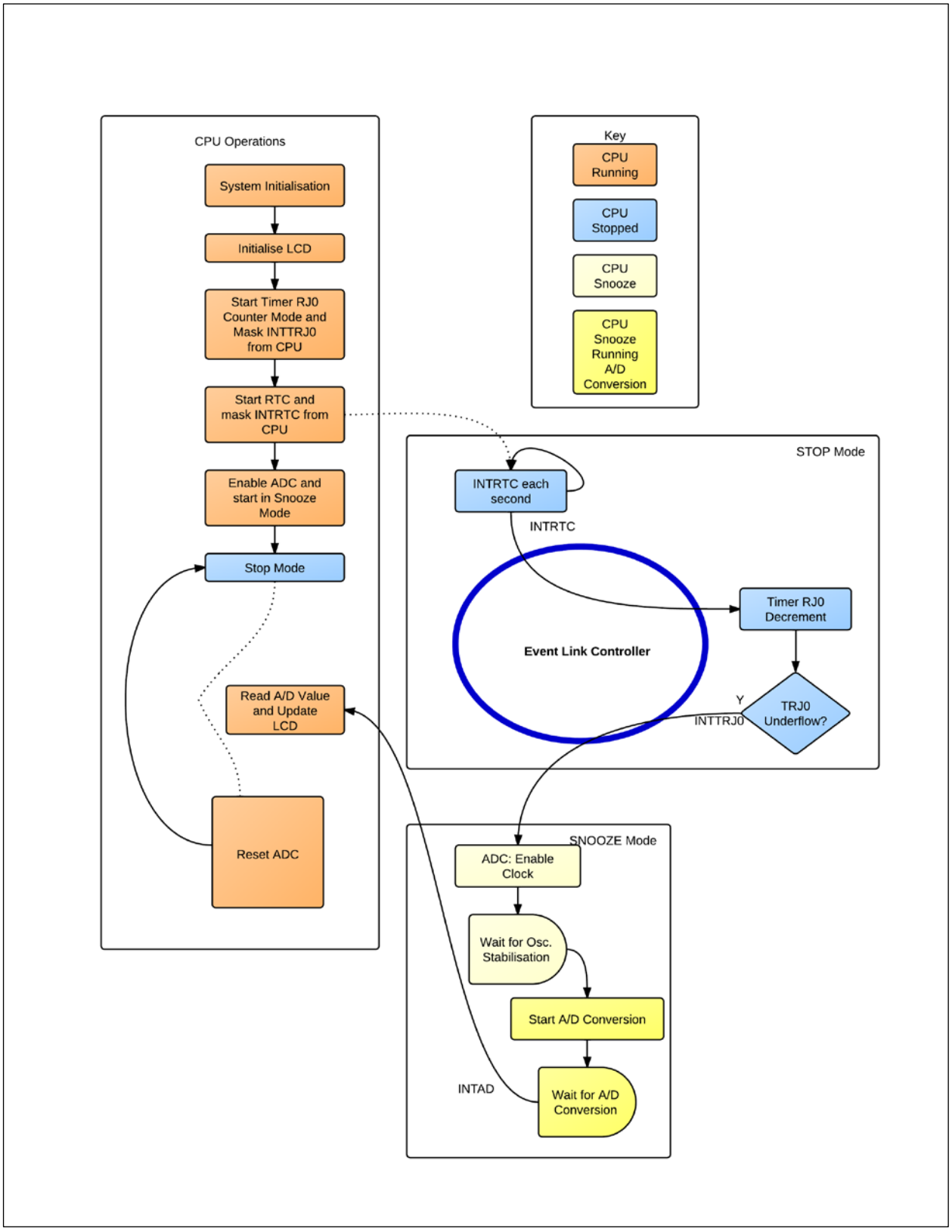

Figure 1 Operational flow

Once the System has initialized, the main() function sets up the LCD, the RTC (in 1 second fixed period interval mode), and Timer RJ0 (in Count Mode). Next the main code enters an infinite while loop which starts the A/D converter (in

Hardware Trigger Wait, One-Shot conversion mode) calls the Stop_Mode() function which disables all unused peripheral clocks and executes a STOP instruction. This stops the CPU execution but allows the selected peripherals to continue running from the external subclock. The RTC generates masked INTRTC interrupt requests every second, which are passed on to Timer RJ0 by the Event Link Controller. For this application note, Timer RJ0 is initialised with a value of 3 so after every 3 RTC events, Timer RJ0 underflows and generates a masked INTTRJ0 interrupt which is passed on to the ADC by the Event Link Controller. This initiates the transition to SNOOZE mode, where the ADC clock is enabled, and after clock stabilization has occurred, the A/D conversion begins without waking the CPU. Once the A/D conversion completes the INTAD interrupt is generated and the CPU returns to run mode to handle the interrupt, which reads the A/D conversion value and outputs it to the LCD. After this, execution continues in the main() function, which resets the ADC and returns to stop mode. This process continues forever.

Note: On the RSKRL78/G14, the CPU current can be measured by removing R26 and connecting an ammeter to J7.

1.1 Main Code

```
/***********************************************************************
* Function Name: main
* Description  : This function implements main function.
* Arguments    : None
* Return Value : None
***********************************************************************/
void main(void)
{
    /* Start user code. Do not edit comment generated here */

    /* This project demonstrates peripheral operation and communication using
     * the Event Link Controller (ELC) whilst in STOP mode.
     * The CPU will be placed into STOP mode for low current consumption, but
     * the RTC will still generate a constant period (1 second) interrupt.
     * This interrupt request is masked to prevent waking the CPU but the
event
     * is passed onto a timer which is in counter mode by the ELC.
     * When the timer underflows (after 3 periods) it also generates an
     * interrupt request which is masked to prevent waking the CPU; again,
     * this signal is passed on by the ELC, this time to the A/D Converter.
     * The A/D converter is set to wake to SNOOZE mode, which means it enables
     * the HOCO for the A/D but does not wake the CPU until the A/D conversion
     * is complete.
     * This means that after each 3 seconds in STOP mode, the A/D conversion
     * will begin and the CPU will wake from STOP mode upon A/D completion.
The
     * LCD is then updated with the read A/D value (adjusted by the
     * potentiometer) and the process restarts. */

    /* (green) LED0 ON to indicate CPU RUN mode */
    LED0 = LED_ON;
    /* (red) LED2 OFF as this should only be on to indicate STOP mode */
    LED2 = LED_OFF;
    /* (red) LED3 OFF */
    LED3 = LED_OFF;

    /* Set the ADC to SNOOZE mode: This means that the ADC will not wake */
    /* the CPU fully from STOP until the A/D conversion completes.       */
    AWC = 1U;

    /* Initialise the LCD module. */
    Init_LCD();

    /* Display project information on the debug LCD. */
    Display_LCD(LCD_LINE1, "RTC->ADC");
    Display_LCD(LCD_LINE2, "Stop/ELC");

    /* Start Timer RJ0 */
    R_TMR_RJ0_Start();

    /* Mask INTTRJ0 - this does not mask the ELC from receiving the signal */
    TRJMK0 = 1U;

    /* Start the RTC */
    R_RTC_Start();
```

```
    /* RTC Interrupt Mask - Mask RTC interrupts */
    RTCMK = 1U;

    /* Loop forever */
    while (1U)
    {
        /* (amber) LED1 OFF - this will be switched on to indicate ADC
completion */
        LED1 = LED_OFF;

        /* Ensure A/D conversion is enabled */
        ADCS = 1U;

        /* Start the ADC */
        R_ADC_Start();

        /* Enter STOP mode */
        Stop_Mode();

        /* The CPU has awoken - now reset ADC */

        /* Disable ADC */
        ADCE = 0U;

        /* Stop the ADC */
        R_ADC_Stop();

        /* Enable ADC */
        ADCE = 1U;
    }
    /* End user code. Do not edit comment generated here */
}

/* Start user code for adding. Do not edit comment generated here */
```

1.2 Stop_Mode() Code

```
/***********************************************************************
* Function Name: Stop_Mode
* Description  : This function sends the MCU into Stop mode.
* Arguments    : None
* Return Value : None
***********************************************************************/
void Stop_Mode(void)
{
    /* Declare variable to store the peripheral clock enable registers */
    uint8_t per0_copy;
    uint8_t per1_copy;
    /* Declare variable to store operation speed mode control register state
*/
    uint8_t osmc_copy;

    /* Delay count variable */
    volatile uint16_t delay = 0;

    /* Store the PER0 state */
    per0_copy = PER0;
    /* Store the OSMC state */
    osmc_copy = OSMC;

    /* Stop all peripheral clock supply other than RTC and ADC */
    PER0 &= 0xA0;
    /* Selection of operation clock for real-time clock, 12-bit interval timer,
and timer RJ */
    OSMC = 0x00;

    /* Add delay before going into stop mode */
    while(delay++ != 0x00FF);

    /* Switch off (green) LED0 to indicate CPU inactive */
    LED0 = LED_OFF;

    /* Switch on (red) LED2 on to indicate STOP Mode */
    LED2 = LED_ON;

    /* Enter STOP mode */
    asm("STOP");

    /* Allow subclock oscillation to stabilise */
    while(--delay != 0);

    /* Switch on LED0 to show CPU active */
    LED0 = LED_ON;

    /* Switch LED2 off to show exit from STOP Mode */
    LED2 = LED_OFF;

    /* Restore the PER0 state */
    PER0 = per0_copy;

    /* Restore the OSMC state */
```

```
    OSMC = osmc_copy;
}
/* End user code. Do not edit comment generated here */
```

2. Event Link Controller

The Event Link Controller can receive interrupt request notifications from selected peripherals and link them to triggers of other selected peripherals without the CPU being required to pass these messages on. Additionally, the ELC continues to receive interrupt request notifications even if the interrupt request to the CPU is masked.

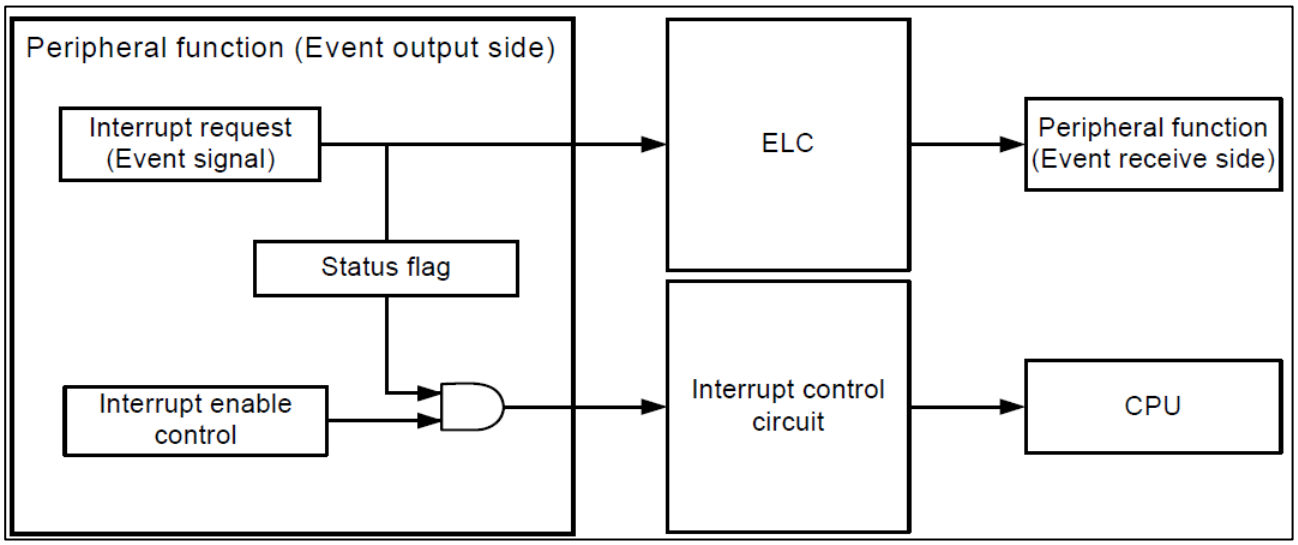

Figure 2 Interrupt Handling

2.1 ELC Source Events

The table below shows the peripheral events that can be used as the source of ELC events, along with which can be used in STOP mode. This application note will use the fixed cycle signal from the RTC and the Timer RJ0 Underflow signal for ELC inputs. There are 25 Event Output Destination Select Registers (ELSELRn) which control the ELC links between peripherals.

Table 1 ELC Source Events

Register Name	Event Generator (Output Origin of Event Input n)	Event Description	Operable in Stop Mode
ELSELR00	External interrupt edge detection 0	INTP0	Y
ELSELR01	External interrupt edge detection 1	INTP1	Y
ELSELR02	External interrupt edge detection 2	INTP2	Y
ELSELR03	External interrupt edge detection 3	INTP3	Y
ELSELR04	External interrupt edge detection 4	INTP4	Y
ELSELR05	External interrupt edge detection 5	INTP5	Y
ELSELR06	Key return signal detection	INTKR	Y
ELSELR07	**RTC fixed-cycle signal/Alarm match detection**	**INTRTC**	**Y**
ELSELR08	Timer RD0 Input capture A/Compare match A	INTTRD0	N
ELSELR09	Timer RD0 Input capture B/Compare match B	INTTRD0	N
ELSELR10	Timer RD1 Input capture A/Compare match A	INTTRD1	N
ELSELR11	Timer RD1 Input capture B/Compare match B	INTTRD1	N
ELSELR12	Timer RD1 Underflow	TRD1 underflow signal	N
ELSELR13	**Timer RJ0 Underflow**	**INTTRJ0**	**Y (Event Counter Mode)**
ELSELR14	Timer RG Input capture A/Compare match A	INTTRG	N
ELSELR15	Timer RG Input capture B/Compare match B	INTTRG	N
ELSELR16	TAU channel 00 Count end/Capture end	INTTM00	N
ELSELR17	TAU channel 01 Count end/Capture end	INTTM01	N
ELSELR18	TAU channel 02 Count end/Capture end	INTTM02	N
ELSELR19	TAU channel 03 Count end/Capture end	INTTM03	N
ELSELR20	TAU channel 10 Count end/Capture end For 80- and 100-pin products only	INTTM10	N
ELSELR21	TAU channel 11 Count end/Capture end For 80- and 100-pin products only	INTTM11	N
ELSELR22	TAU channel 12 Count end/Capture end For 80- and 100-pin products only	INTTM12	N
ELSELR23	TAU channel 13 Count end/Capture end For 80- and 100-pin products only	INTTM13	N
ELSELR24	A/D Comparator detection 0 >=96 KB flash memory products only	INTCMP0	Y
ELSELR25	A/D Comparator detection 1 >=96 KB flash memory products only	INTCMP1	Y

2.2 ELC Event Destinations

Each ELSELRn register corresponds with a particular interrupt source event, and the register value specifies the destination peripheral to pass the event notification on to:

Table 2 ELC Event Destinations

ELSELRn Register Value								Destination	Operable in STOP mode
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3 ELSEL3	Bit 2 ELSEL2	Bit 1 ELSEL1	Bit 0 ELSEL0		
-	-	-	-	0	0	0	1	A/D Converter (Start A/D Conversion)	Y (Wake up using Snooze mode)
-	-	-	-	0	0	1	0	TAU00 (Delay counter, input pulse interval measurement, external event counter)	N
-	-	-	-	0	0	1	1	TAU01 (Delay counter, input pulse interval measurement, external event counter)	N
-	-	-	-	0	1	0	0	Timer RJ0 (Count Source)	Y (Event Count)
-	-	-	-	0	1	0	1	Timer RG (TRGIOB input capture)	N
-	-	-	-	0	1	1	0	Timer RD0 (TRDIOD0 input capture, pulse output forced cutoff)	N
-	-	-	-	0	1	1	1	Timer RD1 (TRDIOD1 input capture, pulse output forced cutoff)	N
-	-	-	-	1	0	0	0	DA0 (Real-time output for >=96 KB code flash memory products only)	N (status before stop is retained)
-	-	-	-	1	0	0	1	DA1 (Real-time output for >=96 KB code flash memory products only)	N (status before stop is retained)

2.3 Demonstrated ELC configuration

This application note demonstrates use of the Real Time Clock, Timer RJ0 and the A/D Converter in stop mode, linked by the Event Link Controller in the following way:

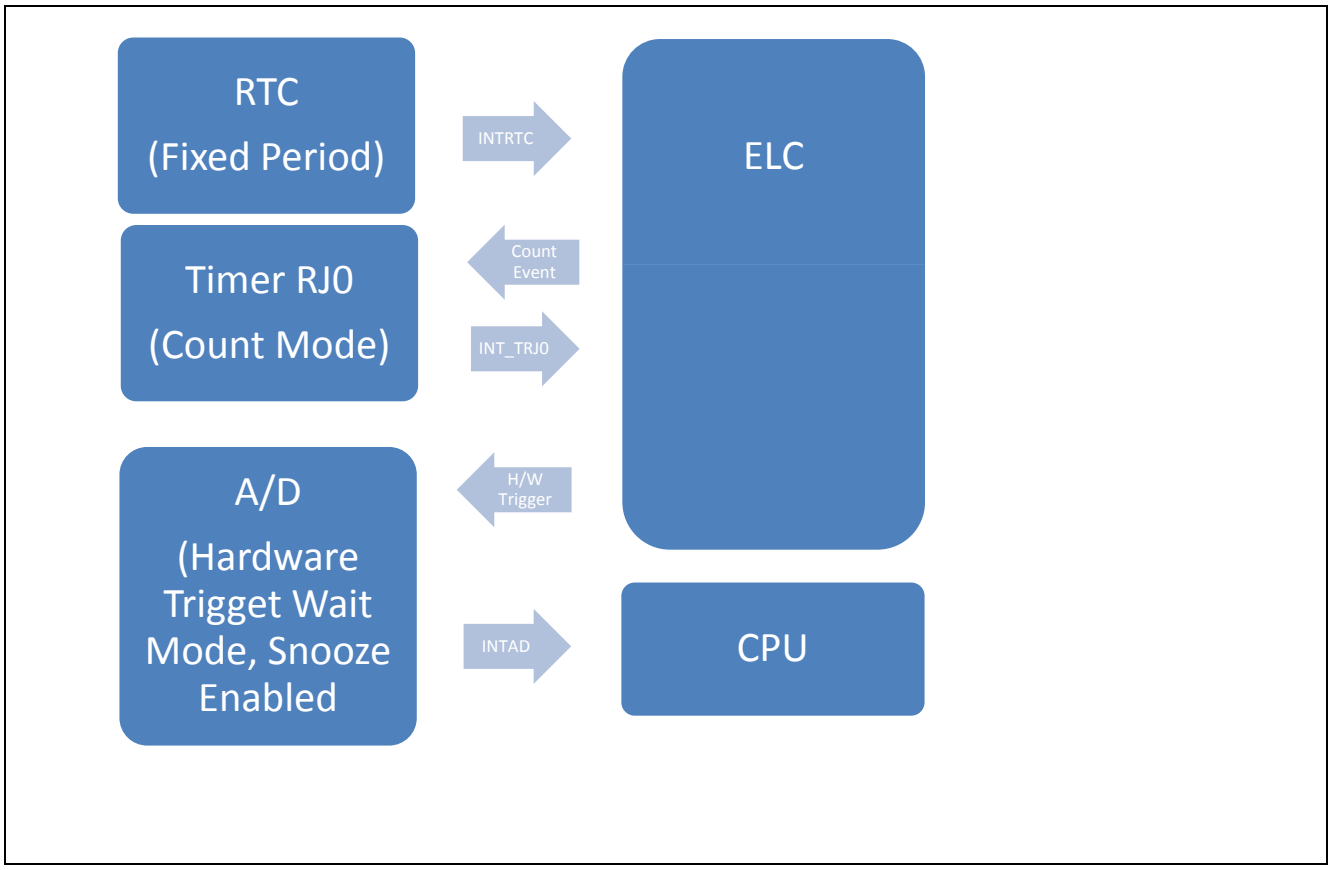

Figure 3 ELC Configuration

2.4 ELC Code

```
/* Event link selection (ELSELn3 - ELSELn0) */
#define _00_ELC_EVENT_LINK_OFF      (0x00U) /* prohibit event link    */
#define _01_ELC_EVENT_LINK_AD       (0x01U) /* link destination AD    */
#define _02_ELC_EVENT_LINK_TAU00    (0x02U) /* link destination TAU00 */
#define _03_ELC_EVENT_LINK_TAU01    (0x03U) /* link destination TAU01 */
#define _04_ELC_EVENT_LINK_RJ0      (0x04U) /* link destination RJ0   */
#define _05_ELC_EVENT_LINK_RG       (0x05U) /* link destination RG    */
#define _06_ELC_EVENT_LINK_RD0      (0x06U) /* link destination RD0   */
#define _07_ELC_EVENT_LINK_RD1      (0x07U) /* link destination RD1   */
#define _08_ELC_EVENT_LINK_DA0      (0x08U) /* link destination DA0   */
#define _09_ELC_EVENT_LINK_DA1      (0x09U) /* link destination DA1   */

/***********************************************************************
* Function Name: R_ELC_Create
* Description  : This function initializes the ELC module.
* Arguments    : None
* Return Value : None
***********************************************************************/
void R_ELC_Create(void)
{
    ELSELR13 = _01_ELC_EVENT_LINK_AD;
    ELSELR07 = _04_ELC_EVENT_LINK_RJ0;
}
```

3. Real-Time Clock

For this application note we need the RTC to be configured with a constant period interrupt of 1 second. This interrupt is masked by setting the RTCMK bit of the MK1H register in order to prevent the interrupt request from waking the CPU.

3.1 RTC Code

This is the code used to set up the RTC.

```
#define _02_RTC_INTRTC_CLOCK_1        (0x02U)    /* once per 1 s */

/***********************************************************************
* Function Name: R_RTC_Create
* Description  : This function initializes the real-time clock module.
* Arguments    : None
* Return Value : None
***********************************************************************/
void R_RTC_Create(void)
{
    RTCEN = 1U;    /* supply RTC clock            */
    RTCE = 0U;     /* disable RTC clock operation */
    RTCMK = 1U;    /* disable INTRTC interrupt    */
    RTCIF = 0U;    /* clear INTRTC interrupt flag */
    /* Set INTRTC low priority */
    RTCPR1 = 1U;
    RTCPR0 = 1U;
    RTCC0 = _02_RTC_INTRTC_CLOCK_1;
}

/***********************************************************************
* Function Name: R_RTC_Start
* Description  : This function enables the real-time clock.
* Arguments    : None
* Return Value : None
***********************************************************************/
void R_RTC_Start(void)
{
    RTCIF = 0U;    /* clear INTRTC interrupt flag */
    RTCE = 1U;     /* enable RTC clock operation  */
}

/***********************************************************************
* Function Name: R_RTC_Stop
* Description  : This function disables the real-time clock.
* Arguments    : None
* Return Value : None
***********************************************************************/
void R_RTC_Stop(void)
{
    RTCE = 0U;    /* disable RTC clock operation */
    RTCMK = 1U;   /* disable INTRTC interrupt    */
    RTCIF = 0U;   /* clear INTRTC interrupt flag */
}
```

4. A/D Converter

When setting up the A/D converter it is important to ensure that it is in Hardware Trigger Wait Mode (ADTMD1 = 1, ADTMD0 = 1), that it is in One-Shot conversion mode (ADSCM = 1), and that the trigger event is set to ELC (ADTRS1 = 0, ADTRS0 = 0):

ADM1 = 0xE1

Table 3 ADM1 Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADTMD1	ADTMD0	ADSCM	-	-	-	ADTRS1	ADTRS0
1	1	1	0	0	0	0	1

Ensure that Snooze Mode is enabled by setting the AWC bit of the ADM2 register high.

ADM2 = 0x05

Table 3 ADM2 Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADREFP1	ADREFP0	ADREFM	-	ADRCK	AWC	-	ADTYP
0	0	0	0	0	1	0	1

4.1 A/D Converter Code

```
#define _04_AD_WAKEUP_ON                (0x04U) /* use wakeup function */
#define _01_AD_RESOLUTION_8BIT          (0x01U) /* 8 bits              */

/***********************************************************************
* Function Name: R_ADC_Create
* Description  : This function initializes the AD converter.
* Arguments    : None
* Return Value : None
***********************************************************************/
void R_ADC_Create(void)
{
    ADCEN = 1U;  /* supply AD clock */
    /* disable AD conversion and clear ADM0 register */
    ADM0 = _00_AD_ADM0_INITIALVALUE;
    ADMK = 1U;  /* disable INTAD interrupt */
    ADIF = 0U;  /* clear INTAD interrupt flag */
    /* Set INTAD low priority */
    ADPR1 = 1U;
    ADPR0 = 1U;
    /* Set ANI0 - ANI8 pin as analog input */
    PM2 |= 0xFFU;
    PM15 |= 0x01U;
    ADM0 = _00_AD_CONVERSION_CLOCK_64 | _00_AD_TIME_MODE_NORMAL_1 |
_00_AD_OPERMODE_SELECT;
    ADM1 = _C0_AD_TRIGGER_HARDWARE_WAIT | _20_AD_CONVMODE_ONESELECT |
_01_AD_TRIGGER_ELC;
    ADM2 = _00_AD_POSITIVE_VDD | _00_AD_NEGATIVE_VSS | _00_AD_AREA_MODE_1 |
_04_AD_WAKEUP_ON | _01_AD_RESOLUTION_8BIT;
    ADUL = _FF_AD_ADUL_VALUE;
    ADLL = _00_AD_ADLL_VALUE;
    ADS = _08_AD_INPUT_CHANNEL_8;
    ADCE = 1U;  /* enable AD comparator */
}
```

```
/***********************************************************************
* Function Name: R_ADC_Start
* Description  : This function starts the AD converter.
* Arguments    : None
* Return Value : None
***********************************************************************/
void R_ADC_Start(void)
{
    ADIF = 0U;  /* clear INTAD interrupt flag */
    ADMK = 0U;  /* enable INTAD interrupt */
}

/***********************************************************************
* Function Name: R_ADC_Stop
* Description  : This function stops the AD converter.
* Arguments    : None
* Return Value : None
***********************************************************************/
void R_ADC_Stop(void)
{
    ADCS = 0U;  /* disable AD conversion */
    ADMK = 1U;  /* disable INTAD interrupt */
    ADIF = 0U;  /* clear INTAD interrupt flag */
  }
```

The INTAD interrupt is not masked so the CPU will be woken from STOP mode upon receipt of this interrupt.

```
/***********************************************************************
* Function Name: INT_AD
* Description  : This function is INTAD interrupt service routine.
* Arguments    : None
* Return Value : None
***********************************************************************/
void INT_AD(void)
{
    /* Start user code. Do not edit comment generated here */

    /* Variable to store the ADC value text */
    static int8_t adc_display[] = "ADC:    ";

    /* Declare variable to hold the integer to BCD conversion bytes */
    static uint8_t bcd_value[3] = {0,0,0};
    uint8_t adc_value;

    R_ADC_Get_Result_8bit(&adc_value);

    /* Switch on (amber) LED1 to indicate ADC completion */
    LED1 = LED_ON;

    /* Convert the integer to BCD and copy the digits into the elc_count array
locations */
    bcd_value[0] = (uint8_t)(adc_value % 10);
    bcd_value[1] = (uint8_t)((adc_value / 10) % 10);
    bcd_value[2] = (uint8_t)((adc_value / 100) % 10);


    /* Store the ELC count */
    adc_display[5] = (int8_t)(0x30 + bcd_value[2]);
    adc_display[6] = (int8_t)(0x30 + bcd_value[1]);
    adc_display[7] = (int8_t)(0x30 + bcd_value[0]);

    /* Display the ADC value */
    Display_LCD(LCD_LINE1, adc_display);

    /* Clear ADC interrupt flag */
    ADIF  = 0;

    /* End user code. Do not edit comment generated here */
}
```

After the interrupt handler has completed, code execution will continue from the instruction following the STOP instruction.

Website and Support <website and support,ws>

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

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

Revision Record

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Oct 29, 2012	—	First edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins
   Handle unused pins in accord 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 one with a different type number, confirm that the change will not lead to problems.
   — The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

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

RENESAS

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700, Fax: +44-1628-651-804

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 2.2