TB3213

Getting Started with Real-Time Counter (RTC)

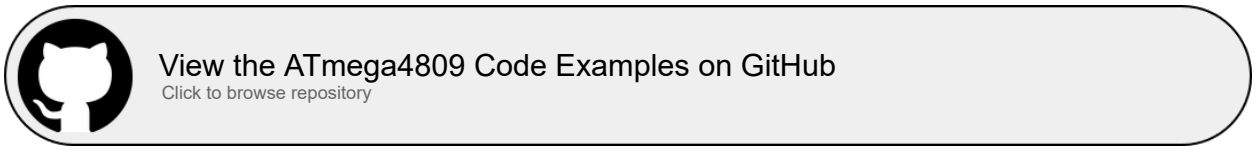
Introduction

Author: Victor Berzan, Microchip Technology Inc.

This technical brief describes how the Real-Time Counter (RTC) module works on tinyAVR® 0- and 1-series, megaAVR® 0-series and AVR® DA devices. It covers the following use cases:

- **RTC Overflow Interrupt:**
  Initialize the RTC, enable the overflow interrupt, and toggle an LED on each overflow.
- **RTC Periodic Interrupt:**
  Initialize the RTC PIT, enable the periodic interrupt, and toggle an LED on each periodic interrupt.
- **RTC PIT Wake from Sleep:**
  Initialize the RTC PIT, enable the periodic interrupt, configure device sleep mode, put CPU in Sleep, the PIT interrupt will wake the CPU.

Note: For each use case described in this document, there are two code examples: One bare metal developed on ATmega4809, and one generated with MPLAB® Code Configurator (MCC) developed on AVR128DA48.



© 2021 Microchip Technology Inc.

Table of Contents

1. Relevant Devices

This section lists the relevant devices for this document. The following figures show the different family devices, laying out pin count variants and memory sizes:

- Vertical migration upwards is possible without code modification, as these devices are pin-compatible and provide the same or more features. Downward migration on tinyAVR® 1-series devices may require code modification due to fewer available instances of some peripherals
- Horizontal migration to the left reduces the pin count and, therefore, the available features
- Devices with different Flash memory sizes typically also have different SRAM and EEPROM

Figure 1-1. tinyAVR® 0-series Overview

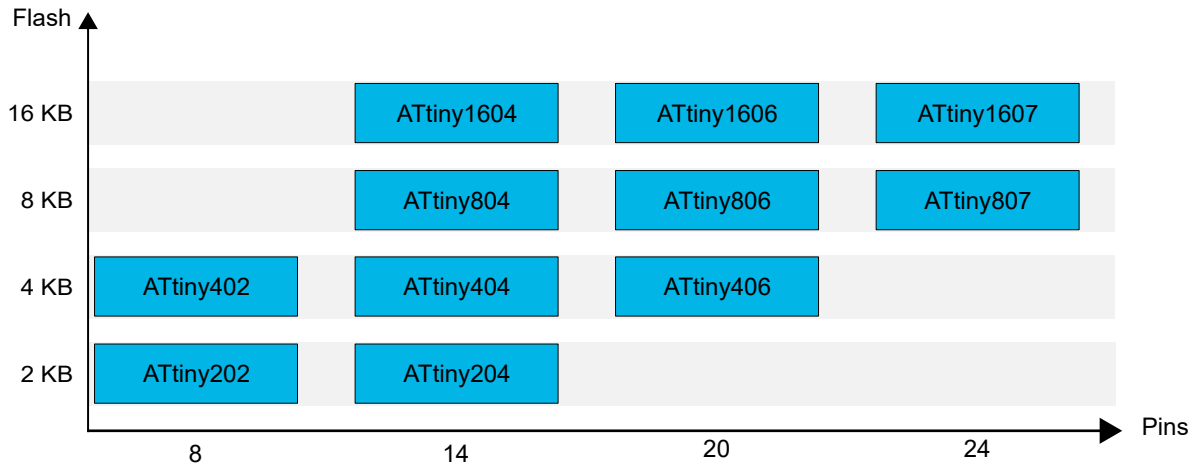


Figure 1-2. tinyAVR® 1-series Overview

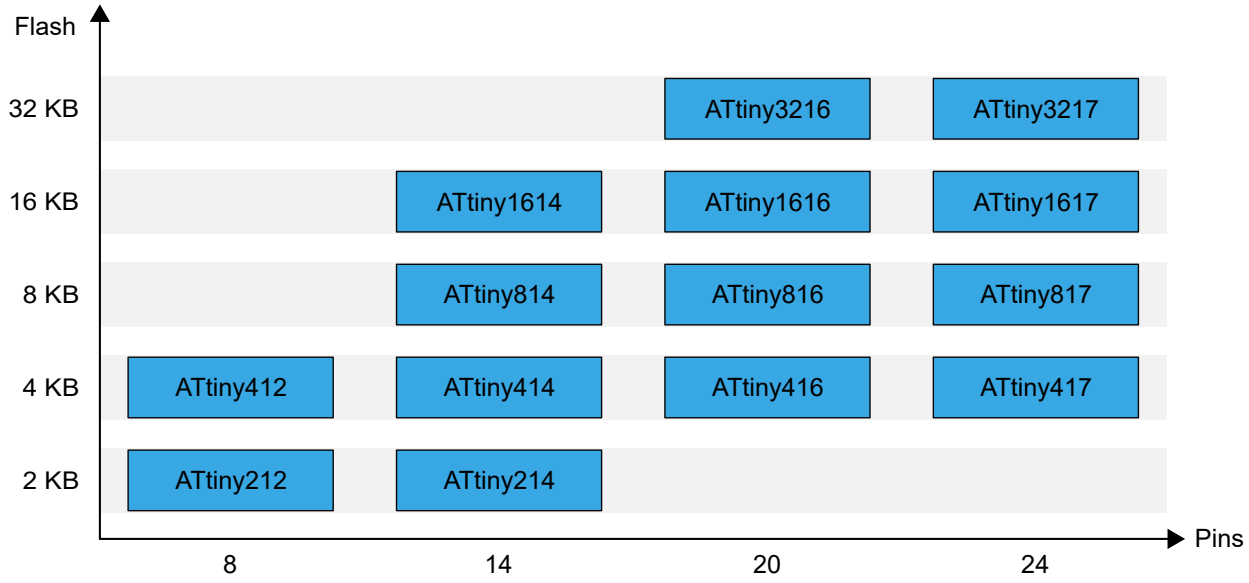


Figure 1-3. megaAVR® 0-series Overview

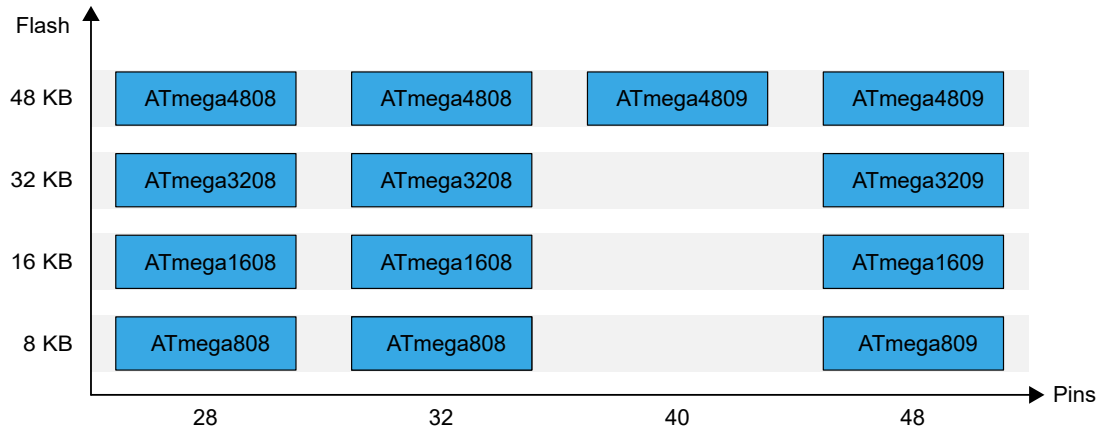
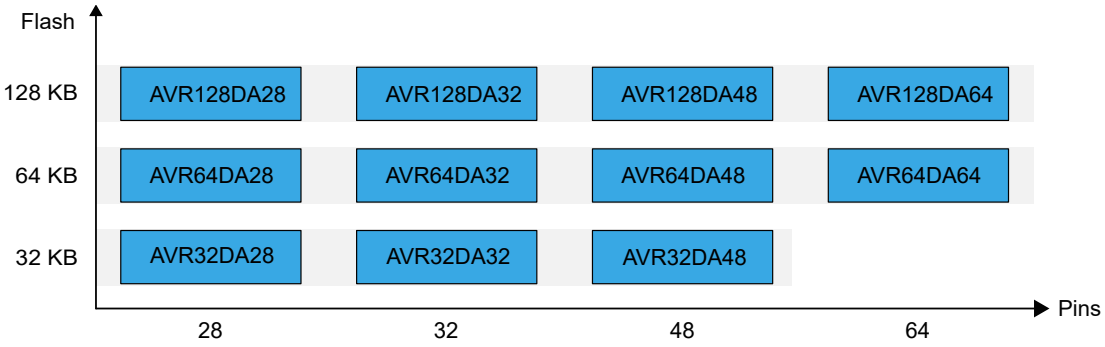


Figure 1-4. AVR® DA Family Overview

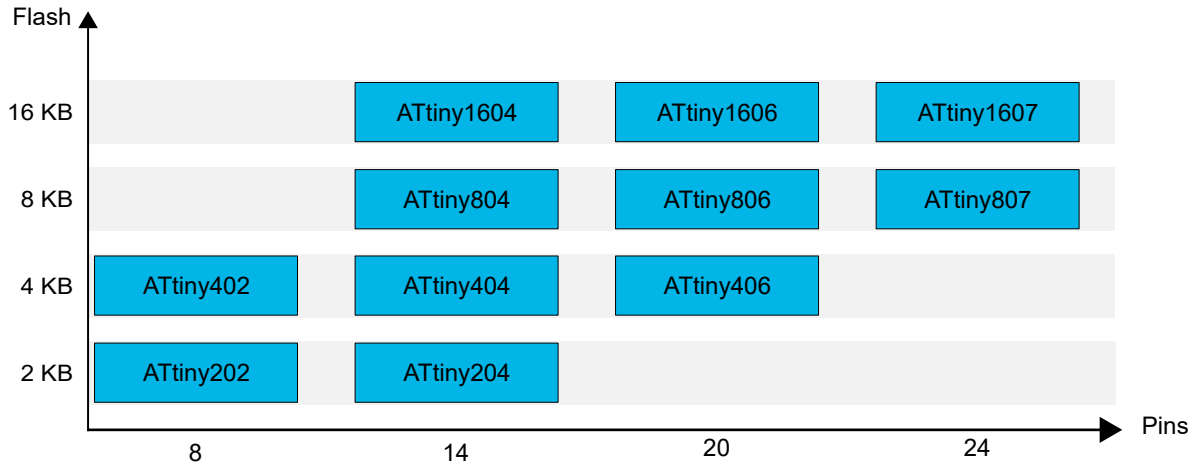


1.1 tinyAVR® 0-series

The figure below shows the tinyAVR® 0-series devices, laying out pin count variants and memory sizes:

- Vertical migration upwards is possible without code modification, as these devices are pin-compatible and provide the same or more features
- Horizontal migration to the left reduces the pin count and, therefore, the available features

Figure 1-5. tinyAVR® 0-series Overview



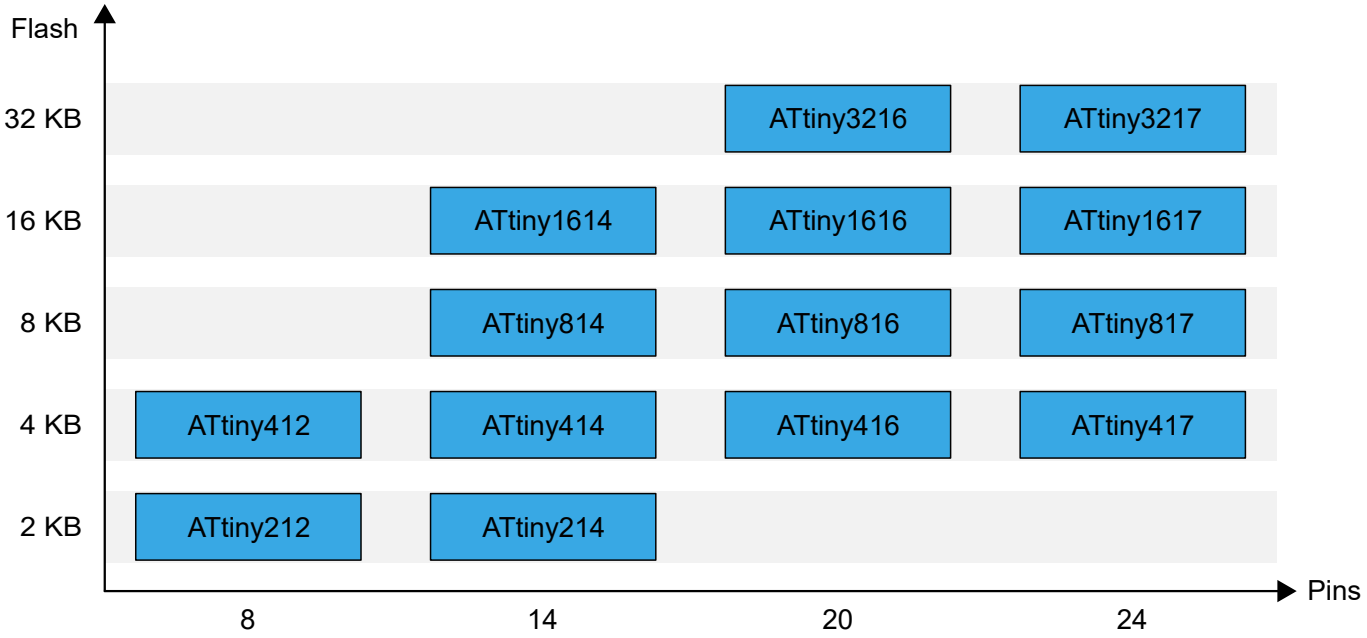
Devices with different Flash memory sizes typically also have different SRAM and EEPROM.

1.2 tinyAVR® 1-series

The following figure shows the tinyAVR 1-series devices, laying out pin count variants and memory sizes:

- Vertical migration upwards is possible without code modification, as these devices are pin-compatible and provide the same or more features. Downward migration may require code modification due to fewer available instances of some peripherals.
- Horizontal migration to the left reduces the pin count and, therefore, the available features

Figure 1-6. tinyAVR® 1-series Overview



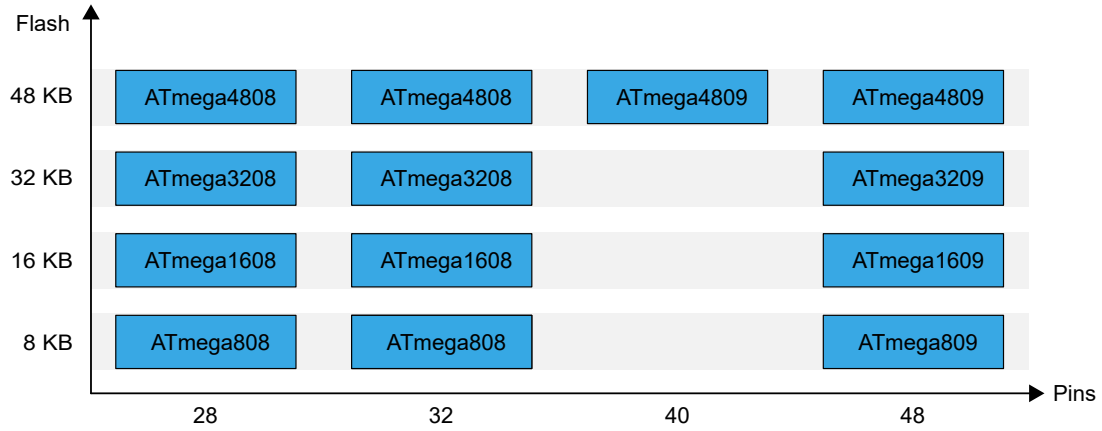
Devices with different Flash memory sizes typically also have different SRAM and EEPROM.

1.3 megaAVR® 0-series

The figure below shows the megaAVR® 0-series devices, laying out pin count variants and memory sizes:

- Vertical migration is possible without code modification, as these devices are fully pin and feature compatible
- Horizontal migration to the left reduces the pin count and, therefore, the available features

Figure 1-7. megaAVR® 0-series Overview



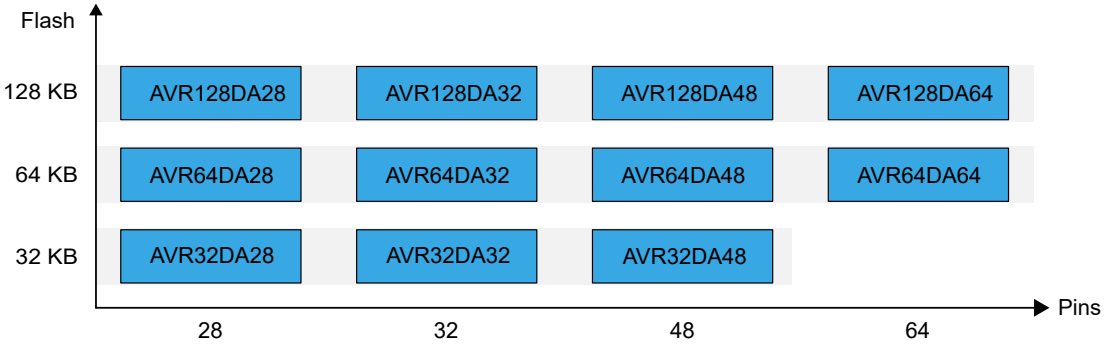
Devices with different Flash memory sizes typically also have different SRAM and EEPROM.

1.4 AVR® DA Family Overview

The figure below shows the AVR® DA devices, laying out pin count variants and memory sizes:

- Vertical migration is possible without code modification, as these devices are fully pin and feature compatible
- Horizontal migration to the left reduces the pin count, and therefore, the available features

Figure 1-8. AVR® DA Family Overview



Devices with different Flash memory sizes typically also have different SRAM.

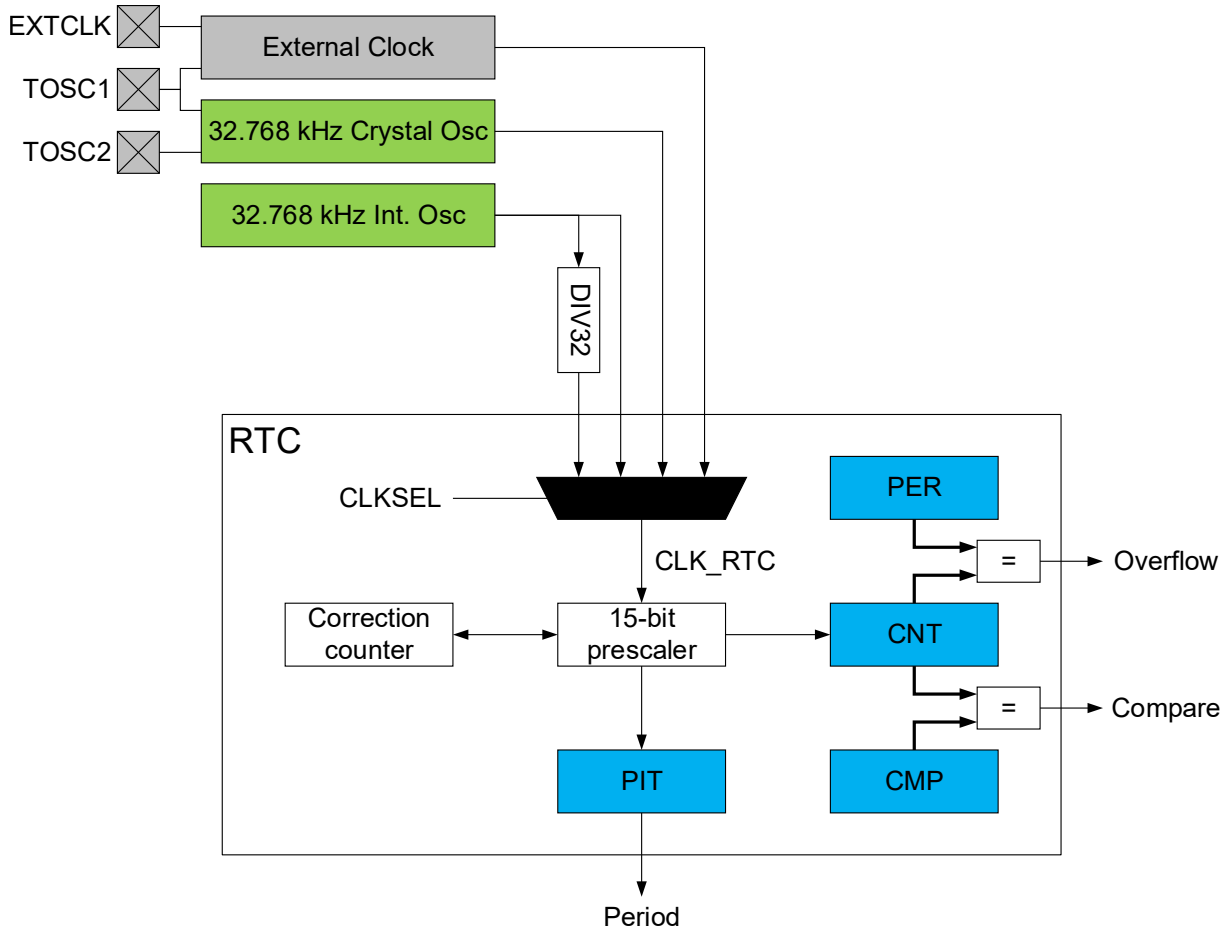
2. Overview

The RTC peripheral offers two timing functions: A Real-Time Counter (RTC) and a Periodic Interrupt Timer (PIT). The PIT functionality can be enabled independently of the RTC functionality.

The RTC counts (prescaled) clock cycles in a Counter register and compares the content of the Counter register to a Period register and a Compare register. The RTC can generate both interrupts and events on compare match or overflow. It will generate a compare interrupt and/or event at the first count after the counter equals the Compare register value, and an overflow interrupt and/or event at the first count after the counter value equals the Period register value. The overflow will also reset the counter value to zero.

Using the same clock source as the RTC function, the PIT can request an interrupt or trigger an output event on every n^{th} clock period ('*n*' can be selected from {4, 8, 16,.. 32768} for interrupts, and from {64, 128, 256,... 8192} for events).

Figure 2-1. Block Diagram



The PIT and RTC functions are running off the same counter inside the prescaler. Writing the PRESCALER bit field in the RTC.CTRLA register configures the period of the clock signal that increments the CNT. The PERIOD bit field in RTC.PITCTRLA selects the bit from the 15-bit prescaler counter to be used as PIT period output.

3. RTC Overflow Interrupt

This code example shows how to use the RTC with overflow interrupt enabled to toggle an LED. The overflow period is 500 ms. The on-board LED will be toggled each time the overflow interrupt occurs.

To operate the RTC, the source clock for the RTC counter must be configured before enabling the RTC peripheral and the desired actions (interrupt requests, output events). In this example, the 32.768 kHz external oscillator is used as the source clock.

To configure the oscillator, first, it must be disabled by clearing the ENABLE bit in the CLKCTRL.XOSC32KCTRLA register:

Figure 3-1. CLKCTRL.XOSC32KCTRLA – Clear the ENABLE Bit

The SEL and CSUT bits cannot be changed as long as the ENABLE bit is set or the XOSC32K Stable (XOSC32KS) bit in CLKCTRL.MCLKSTATUS is high.
To change settings safely: Write a '0' to the ENABLE bit and wait until XOSC32KS is '0' before re-enabling the XOSC32K with new settings.

Bit	7	6	5	4	3	2	1	0
			CSUT[1:0]			SEL	RUNSTDBY	ENABLE
Access			R/W	R/W		R/W	R/W	R/W
Reset			0	0		0	0	0

Bit 0 – ENABLE Enable
When this bit is written to '1', the configuration of the respective input pins is overridden to TOSC1 and TOSC2. Also, the Source Select (SEL) bit and Crystal Start-Up Time (CSUT) become read-only.
This bit is I/O protected to prevent any unintentional enabling of the oscillator.

```
uint8_t temp;
temp = CLKCTRL.XOSC32KCTRLA;
temp &= ~CLKCTRL_ENABLE_bm;
ccp_write_io((void*)&CLKCTRL.XOSC32KCTRLA, temp);
```

The user must then wait for the corresponding status bit to become '0':

Figure 3-2. CLKCTRL.MCLKSTATUS – Read XOSC32KS

Bit	7	6	5	4	3	2	1	0
	EXTS	XOSC32KS	OSC32KS	OSC20MS				SOSC
Access	R	R	R	R				R
Reset	0	0	0	0				0

Bit 6 – XOSC32KS XOSC32K Status
The Status bit will only be available if the source is requested as the main clock or by another module. If the oscillator RUNSTDBY bit is set, but the oscillator is unused/not requested, this bit will be 0.

Value	Description
0	XOSC32K is not stable
1	XOSC32K is stable

```
while(CLKCTRL.MCLKSTATUS & CLKCTRL_XOSC32KS_bm)
{
    ;
}
```

Select the external oscillator by clearing the SEL bit in the CLKCTRL.XOSC32KCTRLA register:

Figure 3-3. CLKCTRL.XOSC32KCTRLA – Clear the SEL Bit

Bit	7	6	5	4	3	2	1	0
			CSUT[1:0]			SEL	RUNSTDBY	ENABLE
Access			R/W	R/W		R/W	R/W	R/W
Reset			0	0		0	0	0

Bit 2 – SEL Source Select
This bit selects the type of external source. It is write protected when the oscillator is enabled (ENABLE = 1).

Value	Description
0	External crystal
1	External clock on TOSC1 pin

```
temp = CLKCTRL.XOSC32KCTRLA;
temp &= ~CLKCTRL_SEL_bm;
ccp_write_io((void*)&CLKCTRL.XOSC32KCTRLA, temp);
```

Then, enable the oscillator by setting the ENABLE bit in the CLKCTRL.XOSC32KCTRLA register:

```
temp = CLKCTRL.XOSC32KCTRLA;
temp |= CLKCTRL_ENABLE_bm;
ccp_write_io((void*)&CLKCTRL.XOSC32KCTRLA, temp);
```

Afterward, the user must wait for all registers to be synchronized:

Figure 3-4. RTC.STATUS

Bit	7	6	5	4	3	2	1	0
					CMPBUSY	PERBUSY	CNTBUSY	CTRLABUSY
Access					R	R	R	R
Reset					0	0	0	0

Bit 3 – CMPBUSY: Compare Synchronization Busy
This bit is indicating whether the RTC is busy synchronizing the Compare register (RTC.CMP) in the RTC clock domain.

Bit 2 – PERBUSY: Period Synchronization Busy
This bit is indicating whether the RTC is busy synchronizing the Period register (RTC.PER) in the RTC clock domain.

Bit 1 – CNTBUSY: Counter Synchronization Busy
This bit is indicating whether the RTC is busy synchronizing the Count register (RTC.CNT) in the RTC clock domain.

Bit 0 – CTRLABUSY: Control A Synchronization Busy
This bit is indicating whether the RTC is busy synchronizing the Control A register (RTC.CTRLA) in the RTC clock domain.

```
while (RTC.STATUS > 0)
{
    ;
}
```

The RTC period is set in the RTC.PER register:

Figure 3-5. RTC.PER – Set Period

The RTC.PERL and RTC.PERH register pair represents the 16-bit value, PER. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset `+0x01`. For more details on reading and writing 16-bit registers, refer to *Accessing 16-bit Registers* in the CPU section.
Due to synchronization between the RTC clock and system clock domains, there is a latency of two RTC clock cycles from updating the register until this has an effect. The application software needs to check that the PERBUSY flag in RTC.STATUS is cleared before writing to this register.

Bit	15	14	13	12	11	10	9	8
	PER[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	7	6	5	4	3	2	1	0
	PER[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bits 15:8 – PER[15:8] Period High Byte
These bits hold the MSB of the 16-bit Period register.

Bits 7:0 – PER[7:0] Period Low Byte
These bits hold the LSB of the 16-bit Period register.

The 32.768 kHz External Crystal Oscillator clock is selected in the RTC.CLKSEL register:

Figure 3-6. RTC.CLKSEL – Clock Selection

Bit	7	6	5	4	3	2	1	0
							CLKSEL[1:0]	
Access							R/W	R/W
Reset							0	0

Bits 1:0 – CLKSEL[1:0] Clock Select bits
Writing these bits select the source for the RTC clock (CLK_RTC).
When configuring the RTC to use either XOSC32K or the external clock on TOSC1, XOSC32K needs to be enabled, and the Source Select (SEL) bit and Run Standby (RUNSTDBY) bit in the XOSC32K Control A register of the Clock Controller (CLKCTRL.XOSC32KCTRLA) must be configured accordingly.

Value	Name	Description
0x0	INT32K	32.768 kHz from OSCULP32K
0x1	INT1K	1.024 kHz from OSCULP32K
0x2	TOSC32K	32.768 kHz from XOSC32K or external clock from TOSC1
0x3	EXTCLK	External clock from the EXTCLK pin

```
RTC.CLKSEL = RTC_CLKSEL_TOSC32K_gc;
```

To enable the RTC to also run in Debug mode, the DBGRUN bit is set in the RTC.DBGCTRL register:

Figure 3-7. RTC.CLKSEL – Set the DBGRUN Bit

Bit	7	6	5	4	3	2	1	0
								DBGRUN
Access								R/W
Reset								0

Bit 0 – DBGRUN: Debug Run bit

Value	Description
0	The peripheral is halted in Break Debug mode and ignores events
1	The peripheral will continue to run in Break Debug mode when the CPU is halted

```
RTC.DBGCTRL |= RTC_DBGRUN_bm;
```

The RTC prescaler is set in the RTC.CTRLA register. Set the RUNSTDBY bit in RTC.CTRLA to enable the RTC to also run in Standby mode. Set the RTCEN bit in RTC.CTRLA to enable the RTC.

Figure 3-8. RTC.CTRLA – Set the Prescaler, RUNSTDBY Bit, RTCEN Bit

Bit	7	6	5	4	3	2	1	0
	RUNSTDBY	PRESCALER[3:0]				CORREN		RTCEN
Access	R/W	R/W	R/W	R/W	R/W	R/W		R/W
Reset	0	0	0	0	0	0		0

Bit 7 – RUNSTDBY Run in Standby

Value	Description
0	RTC disabled in Standby sleep mode
1	RTC enabled in Standby sleep mode

Bits 6:3 – PRESCALER[3:0] Prescaler bits
These bits define the prescaling of the CLK_RTC clock signal. Due to synchronization between the RTC clock and system clock domains, there is a latency of two RTC clock cycles from updating the register until this has an effect. The application software needs to check that the CTRLABUSY flag in RTC.STATUS is cleared before writing to this register.

Value	Name	Description
0x0	DIV1	RTC clock/1 (no prescaling)
0x1	DIV2	RTC clock/2
0x2	DIV4	RTC clock/4
0x3	DIV8	RTC clock/8
0x4	DIV16	RTC clock/16
0x5	DIV32	RTC clock/32
0x6	DIV64	RTC clock/64
0x7	DIV128	RTC clock/128
0x8	DIV256	RTC clock/256
0x9	DIV512	RTC clock/512
0xA	DIV1024	RTC clock/1024
0xB	DIV2048	RTC clock/2048
0xC	DIV4096	RTC clock/4096
0xD	DIV8192	RTC clock/8192
0xE	DIV16384	RTC clock/16384
0xF	DIV32768	RTC clock/32768

Bit 0 – RTCEN RTC Peripheral Enable

Value	Description
0	RTC peripheral disabled
1	RTC peripheral enabled

```
RTC.CTRLA = RTC_PRESCALER_DIV32_gc | RTC_RTCEN_bm | RTC_RUNSTDBY_bm;
```

Enable the overflow interrupt by setting the OVF bit in the RTC.INTCTRL register:

Figure 3-9. RTC.INTCTRL – Set the OVF Bit

Bit	7	6	5	4	3	2	1	0
							CMP	OVF
Access							R/W	R/W
Reset							0	0

Bit 0 – OVF Overflow Interrupt Enable
Enable interrupt-on-counter overflow (i.e., when the Counter value (CNT) matches the Period value (PER) and wraps around to zero).

```
RTC.INTCTRL |= RTC_OVF_bm;
```

For the interrupt to occur, the global interrupts must be enabled:

```
sei();
```

The Interrupt Service Routine (ISR) for the RTC overflow will toggle an LED in the example below:

```
ISR(RTC_CNT_vect)
{
    RTC.INTFLAGS = RTC_OVF_bm;
    LED0_toggle();
}
```

Note: Clear the OVF bit from the RTC.INTFLAGS register by writing a '1' to it inside the ISR function.

Tip: The full code example is available in the Appendix section.

View the ATmega4809 Code Example on GitHub
Click to browse repository

An MCC generated code example for AVR128DA48, with the same functionality as the one described in this section, can be found here:

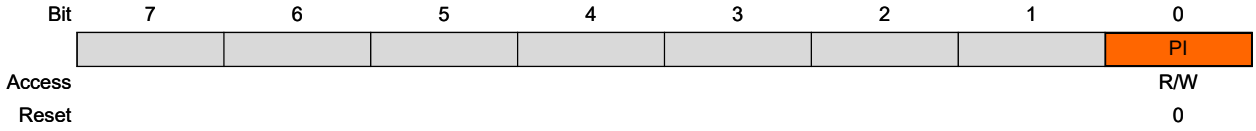
View the AVR128DA48 Code Example on GitHub
Click to browse repository

4. RTC Periodic Interrupt

This code example shows how to use the PIT timing function of the RTC. The on-board LED will be toggled each second when the periodic interrupt occurs.

The source clock configuration for this particular example is the same as for the RTC overflow interrupt example. Enable the periodic interrupt by setting the PI bit in the RTC.PITINTCTRL register.

Figure 4-1. RTC.PITINTCTRL – Set the PI Bit



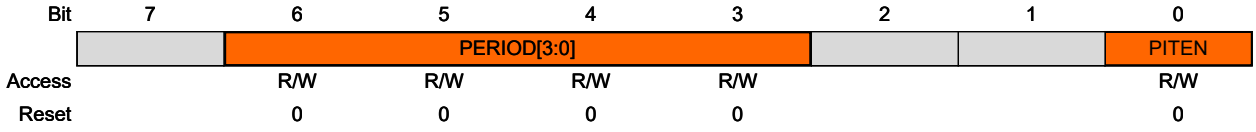
Bit 0 – PI: Periodic Interrupt bit

Value	Description
0	The periodic interrupt is disabled
1	The periodic interrupt is enabled

```
RTC.PITINTCTRL = RTC_PI_bm;
```

The PIT period is set in the RTC.PITCTRLA register. Enable the PIT by setting the PITEN bit in RTC.PITCTRLA.

Figure 4-2. RTC.PITCTRLA – Set the PITEN Bit



Bits 6:3 – PERIOD[3:0]: Period bits
Writing this bit field selects the number of RTC clock cycles between each interrupt.

Value	Name	Description
0x0	OFF	No interrupt
0x1	CYC4	4 cycles
0x2	CYC8	8 cycles
0x3	CYC16	16 cycles
0x4	CYC32	32 cycles
0x5	CYC64	64 cycles
0x6	CYC128	128 cycles
0x7	CYC256	256 cycles
0x8	CYC512	512 cycles
0x9	CYC1024	1024 cycles
0xA	CYC2048	2048 cycles
0xB	CYC4096	4096 cycles
0xC	CYC8192	8192 cycles
0xD	CYC16384	16384 cycles
0xE	CYC32768	32768 cycles
0xF	-	Reserved

Bit 0 – PITEN: Periodic Interrupt Timer Enable bit
Writing a '1' to this bit enables the Periodic Interrupt Timer.

```
RTC.PITCTRLA = RTC_PERIOD_CYC32768_gc | RTC_PITEN_bm;
```

For the interrupt to occur, the global interrupts must be enabled:

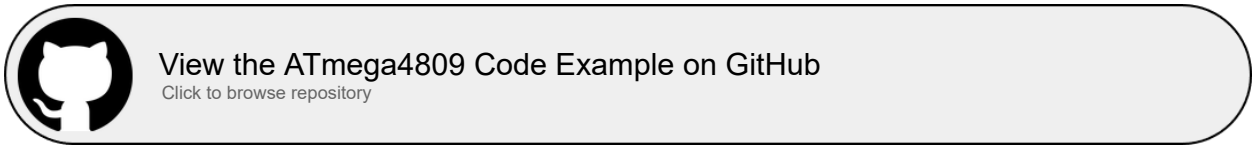
```
sei();
```

The Interrupt Service Routine (ISR) for the RTC PIT will toggle an LED in the example below:

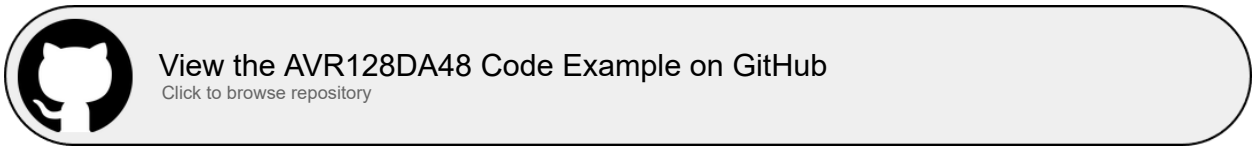
```
ISR(RTC_PIT_vect)
{
    RTC.PITINTFLAGS = RTC_PI_bm;
    LED0_toggle();
}
```

Note: Clear the PI bit from the RTC.PITINTFLAGS register by writing a '1' to it inside the ISR function.

Tip: The full code example is available in the Appendix section.



An MCC generated code example for AVR128DA48, with the same functionality as the one described in this section, can be found here:



5. RTC PIT Wake from Sleep

This code example shows how to use the PIT timing function of the RTC to wake up the CPU from sleep. The on-board LED will be toggled each second when the periodic interrupt occurs, meaning it will be toggled when the CPU wakes up from sleep.

The sleep mode is configured in the SLPCTRL.CTRLA register. The sleep feature is enabled by setting the SEN bit in SLPCTRL.CTRLA.

Figure 5-1. SLPCTRL.CTRLA – Set the Sleep Mode, SEN Bit

Bit	7	6	5	4	3	2	1	0
						SMODE[1:0]		SEN
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 2:1 – SMODE[1:0] Sleep Mode
Writing these bits selects the sleep mode entered when the Sleep Enable (SEN) bit is written to '1', and the SLEEP instruction is executed.

Value	Name	Description
0x0	IDLE	Idle sleep mode enabled
0x1	STANDBY	Standby sleep mode enabled
0x2	PDOWN	Power-Down sleep mode enabled
other	-	Reserved

Bit 0 – SEN Sleep Enable
This bit must be written to '1' before the SLEEP instruction is executed to make the MCU enter the selected sleep mode.

```
SLPCTRL.CTRLA |= SLPCTRL_SMODE_PDOWN_gc;
SLPCTRL.CTRLA |= SLPCTRL_SEN_bm;
```

The CPU can be put to sleep by calling the following function:

```
sleep_cpu();
```

The PIT interrupt will wake the CPU from sleep. For the interrupt to occur, the global interrupts must be enabled:

```
sei();
```

The Interrupt Service Routine (ISR) for the RTC PIT will toggle an LED in the example below:

```
ISR(RTC_PIT_vect)
{
    RTC.PITINTFLAGS = RTC_PI_bm;
    LED0_toggle();
}
```

Note: Clear the PI bit from the RTC.PITINTFLAGS register by writing a '1' to it inside the ISR function.

Tip: The full code example is available in the Appendix section.

View the ATmega4809 Code Example on GitHub
Click to browse repository

An MCC generated code example for AVR128DA48, with the same functionality as the one described in this section, can be found here:

View the AVR128DA48 Code Example on GitHub
Click to browse repository

6. References

More information about the RTC and PIT operation modes can be found at the following links:

1. AVR128DA48 product page: www.microchip.com/wwwproducts/en/AVR128DA48
2. AVR128DA48 Curiosity Nano Evaluation Kit product page: https://www.microchip.com/Developmenttools/ProductDetails/DM164151
3. AVR128DA28/32/48/64 Data Sheet
4. Getting Started with the AVR® DA Family
5. ATmega4809 product page: www.microchip.com/wwwproducts/en/ATMEGA4809
6. megaAVR® 0-series Family Data Sheet
7. ATmega809/1609/3209/4809 – 48-Pin Data Sheet megaAVR® 0-series
8. ATmega4809 Xplained Pro product page: https://www.microchip.com/developmenttools/ProductDetails/atmega4809-xpro

7. Appendix

Example 7-1. RTC Overflow Interrupt Code Example

```
/* RTC Period */
#define RTC_EXAMPLE_PERIOD            (511)

#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/cpufunc.h>

void RTC_init(void);
void LED0_init(void);
inline void LED0_toggle(void);

void RTC_init(void)
{
    uint8_t temp;

    /* Initialize 32.768kHz Oscillator: */
    /* Disable oscillator: */
    temp = CLKCTRL.XOSC32KCTRLA;
    temp &= ~CLKCTRL_ENABLE_bm;
    /* Writing to protected register */
    ccp_write_io((void*)&CLKCTRL.XOSC32KCTRLA, temp);

    while(CLKCTRL.MCLKSTATUS & CLKCTRL_XOSC32KS_bm)
    {
        ; /* Wait until XOSC32KS becomes 0 */
    }

    /* SEL = 0 (Use External Crystal): */
    temp = CLKCTRL.XOSC32KCTRLA;
    temp &= ~CLKCTRL_SEL_bm;
    /* Writing to protected register */
    ccp_write_io((void*)&CLKCTRL.XOSC32KCTRLA, temp);

    /* Enable oscillator: */
    temp = CLKCTRL.XOSC32KCTRLA;
    temp |= CLKCTRL_ENABLE_bm;
    /* Writing to protected register */
    ccp_write_io((void*)&CLKCTRL.XOSC32KCTRLA, temp);

    /* Initialize RTC: */
    while (RTC.STATUS > 0)
    {
        ; /* Wait for all register to be synchronized */
    }

    /* Set period */
    RTC.PER = RTC_EXAMPLE_PERIOD;

    /* 32.768kHz External Crystal Oscillator (XOSC32K) */
    RTC.CLKSEL = RTC_CLKSEL_TOSC32K_gc;

    /* Run in debug: enabled */
    RTC.DBGCTRL |= RTC_DBGRUN_bm;

    RTC.CTRLA = RTC_PRESCALER_DIV32_gc  /* 32 */
    | RTC_RTCEN_bm             /* Enable: enabled */
    | RTC_RUNSTDBY_bm;         /* Run In Standby: enabled */

    /* Enable Overflow Interrupt */
    RTC.INTCTRL |= RTC_OVF_bm;
}

void LED0_init(void)
{
    /* Make High (OFF) */
    PORTB.OUT |= PIN5_bm;
    /* Make output */
    PORTB.DIR |= PIN5_bm;
}
```

```
inline void LED0_toggle(void)
{
    PORTB.OUTTGL |= PIN5_bm;
}

ISR(RTC_CNT_vect)
{
    /* Clear flag by writing '1': */
    RTC.INTFLAGS = RTC_OVF_bm;

    LED0_toggle();
}

int main(void)
{
    LED0_init();
    RTC_init();

    /* Enable Global Interrupts */
    sei();

    while (1)
    {
    }
}
```

Example 7-2. RTC Periodic Interrupt Code Example

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/cpufunc.h>

void RTC_init(void);
void LED0_init(void);
inline void LED0_toggle(void);

void RTC_init(void)
{
    uint8_t temp;

    /* Initialize 32.768kHz Oscillator: */
    /* Disable oscillator: */
    temp = CLKCTRL.XOSC32KCTRLA;
    temp &= ~CLKCTRL_ENABLE_bm;
    /* Writing to protected register */
    ccp_write_io((void*)&CLKCTRL.XOSC32KCTRLA, temp);

    while(CLKCTRL.MCLKSTATUS & CLKCTRL_XOSC32KS_bm)
    {
        ; /* Wait until XOSC32KS becomes 0 */
    }

    /* SEL = 0 (Use External Crystal): */
    temp = CLKCTRL.XOSC32KCTRLA;
    temp &= ~CLKCTRL_SEL_bm;
    /* Writing to protected register */
    ccp_write_io((void*)&CLKCTRL.XOSC32KCTRLA, temp);

    /* Enable oscillator: */
    temp = CLKCTRL.XOSC32KCTRLA;
    temp |= CLKCTRL_ENABLE_bm;
    /* Writing to protected register */
    ccp_write_io((void*)&CLKCTRL.XOSC32KCTRLA, temp);

    /* Initialize RTC: */
    while (RTC.STATUS > 0)
    {
        ; /* Wait for all register to be synchronized */
    }

    /* 32.768kHz External Crystal Oscillator (XOSC32K) */
    RTC.CLKSEL = RTC_CLKSEL_TOSC32K_gc;
```

```
    /* Run in debug: enabled */
    RTC.DBGCTRL = RTC_DBGRUN_bm;

    RTC.PITINTCTRL = RTC_PI_bm; /* Periodic Interrupt: enabled */

    RTC.PITCTRLA = RTC_PERIOD_CYC32768_gc /* RTC Clock Cycles 32768 */
                 | RTC_PITEN_bm; /* Enable: enabled */
}

void LED0_init(void)
{
    /* Make High (OFF) */
    PORTB.OUT |= PIN5_bm;
    /* Make output */
    PORTB.DIR |= PIN5_bm;
}

inline  void LED0_toggle(void)
{
    PORTB.OUTTGL |= PIN5_bm;
}

ISR(RTC_PIT_vect)
{
    /* Clear flag by writing '1': */
    RTC.PITINTFLAGS = RTC_PI_bm;

    LED0_toggle();
}

int main(void)
{
    LED0_init();
    RTC_init();

    /* Enable Global Interrupts */
    sei();

    while (1)
    {
    }
}
```

Example 7-3. RTC PIT Wake from Sleep Code Example

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <avr/cpufunc.h>

void RTC_init(void);
void LED0_init(void);
inline void LED0_toggle(void);
void SLPCTRL_init(void);

void RTC_init(void)
{
    uint8_t temp;

    /* Initialize 32.768kHz Oscillator: */
    /* Disable oscillator: */
    temp = CLKCTRL.XOSC32KCTRLA;
    temp &= ~CLKCTRL_ENABLE_bm;
    /* Writing to protected register */
    ccp_write_io((void*)&CLKCTRL.XOSC32KCTRLA, temp);

    while(CLKCTRL.MCLKSTATUS & CLKCTRL_XOSC32KS_bm)
    {
        ; /* Wait until XOSC32KS becomes 0 */
    }

    /* SEL = 0 (Use External Crystal): */
```

```
    temp = CLKCTRL.XOSC32KCTRLA;
    temp &= ~CLKCTRL_SEL_bm;
    /* Writing to protected register */
    ccp_write_io((void*)&CLKCTRL.XOSC32KCTRLA, temp);

    /* Enable oscillator: */
    temp = CLKCTRL.XOSC32KCTRLA;
    temp |= CLKCTRL_ENABLE_bm;
    /* Writing to protected register */
    ccp_write_io((void*)&CLKCTRL.XOSC32KCTRLA, temp);

    /* Initialize RTC: */
    while (RTC.STATUS > 0)
    {
        ; /* Wait for all register to be synchronized */
    }

    /* 32.768kHz External Crystal Oscillator (XOSC32K) */
    RTC.CLKSEL = RTC_CLKSEL_TOSC32K_gc;

    /* Run in debug: enabled */
    RTC.DBGCTRL = RTC_DBGRUN_bm;

    RTC.PITINTCTRL = RTC_PI_bm; /* Periodic Interrupt: enabled */

    RTC.PITCTRLA = RTC_PERIOD_CYC32768_gc /* RTC Clock Cycles 32768 */
                 | RTC_PITEN_bm; /* Enable: enabled */
}

void LED0_init(void)
{
    /* Make High (OFF) */
    PORTB.OUT |= PIN5_bm;
    /* Make output */
    PORTB.DIR |= PIN5_bm;
}

inline void LED0_toggle(void)
{
    PORTB.OUTTGL |= PIN5_bm;
}

ISR(RTC_PIT_vect)
{
    /* Clear flag by writing '1': */
    RTC.PITINTFLAGS = RTC_PI_bm;

    LED0_toggle();
}

void SLPCTRL_init(void)
{
    SLPCTRL.CTRLA |= SLPCTRL_SMODE_PDOWN_gc;
    SLPCTRL.CTRLA |= SLPCTRL_SEN_bm;
}

int main(void)
{
    LED0_init();
    RTC_init();
    SLPCTRL_init();

    /* Enable Global Interrupts */
    sei();

    while (1)
    {
        /* Put the CPU in sleep */
        sleep_cpu();

        /* The PIT interrupt will wake the CPU */
    }
}
```

8. Revision History

Document Revision	Date	Comments
C	02/2021	Updated the GitHub repository links, the *References* section, and the use cases sections. Added the *AVR® DA Family Overview* section. Added MCC versions for each use case, running on AVR128DA48. Other minor corrections.
B	12/2019	A write to the CCP registers is replaced by the `ccp_write_io()` function in the code.
A	05/2019	Initial document release.

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods being used in attempts to breach the code protection features of the Microchip devices. We believe that these methods require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Attempts to breach these code protection features, most likely, cannot be accomplished without violating Microchip's intellectual property rights.
- Microchip is willing to work with any customer who is concerned about the integrity of its code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable." Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

© 2021 Microchip Technology Inc.

Legal Notice

Information contained in this publication is provided for the sole purpose of designing with and using Microchip products. Information regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications.

THIS INFORMATION IS PROVIDED BY MICROCHIP “AS IS”. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION. Use of Microchip devices in life support and/or safety applications is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PackeTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AgileSwitch, APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, FlashTec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, WinPath, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, Augmented Switching, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, Espresso T1S, EtherGREEN, IdealBridge, In-Circuit Serial Programming, ICSP, INICnet, Intelligent Paralleling, Inter-Chip Connectivity, JitterBlocker, maxCrypto, maxView, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, RTAX, RTG4, SAM-ICE, Serial Quad I/O, simpleMAP, SimpliPHY, SmartBuffer, SMART-I.S., storClad, SQI, SuperSwitcher, SuperSwitcher II, Switchtec, SynchroPHY, Total Endurance, TSHARC, USBCheck, VariSense, VectorBlox, VeriPHY, ViewSpan, WiperLock, XpressConnect, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2021, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN: 978-1-5224-7542-2

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: www.microchip.com/support Web Address: www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4485-5910 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-72400 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820