TB3214

Getting Started with Timer/Counter Type B (TCB)

Introduction

Author: Marius Nicolae, Microchip Technology Inc.

The tinyAVR® 0- and 1-series, megaAVR® 0-series, and AVR® DA-series of microcontrollers are equipped with powerful timers that enable them to cover a wide area of applications. Timer/Counter type B (TCB) offers a variety of features and operation modes, from periodic interrupts to 8-bit PWM or time-out. The various operation modes of the TCB can be used in correlation with the event system. The scope of this technical brief is to show the configuration of three operation modes.

- **Using TCB in 8-bit PWM mode:**
  TCB will be configured in this mode and will generate a PWM signal with a 50% duty cycle with a period of one second. A GPIO pin will be used as an output to showcase the signal.
- **Using TCB in Time-Out Check mode:**
  TCB will be configured in this mode and will be used to measure the signal time (edge-to-edge) generated by a GPIO pin configured as an input. An interrupt will be generated if the time-out (the period from edge-to-edge) expires.
- **Using TCB in sleep mode:**
  TCB will be configured to generate a periodic interrupt, even when the AVR® microcontroller is in Standby sleep mode.

Note: For each of the use cases 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.

View the ATmega4809 Code Example on GitHub
Click to browse repository

View the AVR128DA48 Code Example on GitHub
Click to browse repository

Table of Contents

Introduction....1

1. Relevant Devices....3
2. Overview....6
3. Using TCB in 8-Bit PWM Mode....7
4. Using TCB in Time-Out Check Mode....13
5. Using TCB in Sleep Mode....17
6. References....20
7. Appendix....21
8. Revision History....24

Microchip Information....25

- The Microchip Website....25
- Product Change Notification Service....25
- Customer Support....25
- Microchip Devices Code Protection Feature....25
- Legal Notice....25
- Trademarks....26
- Quality Management System....27
- Worldwide Sales and Service....28

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

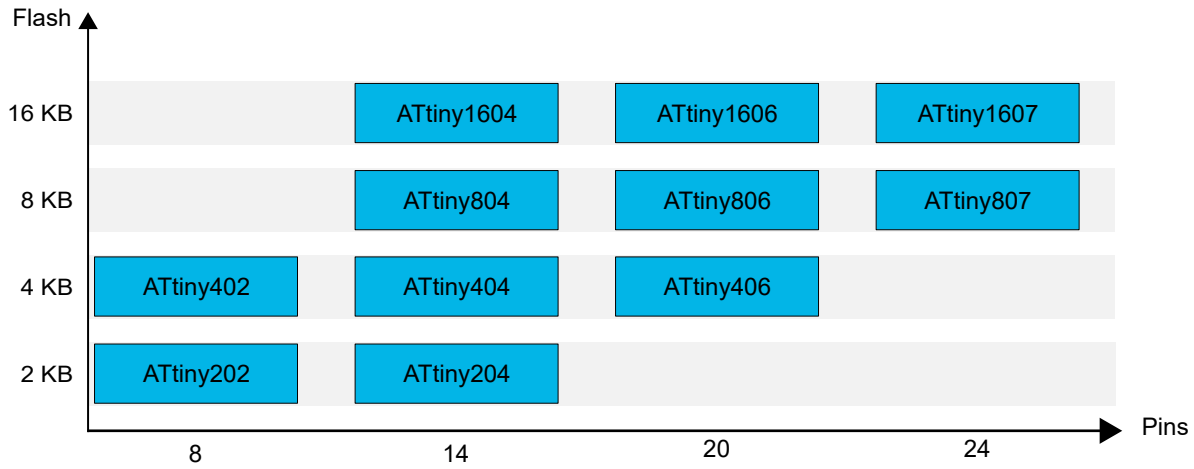


Figure 1-2. tinyAVR® 1-series Overview

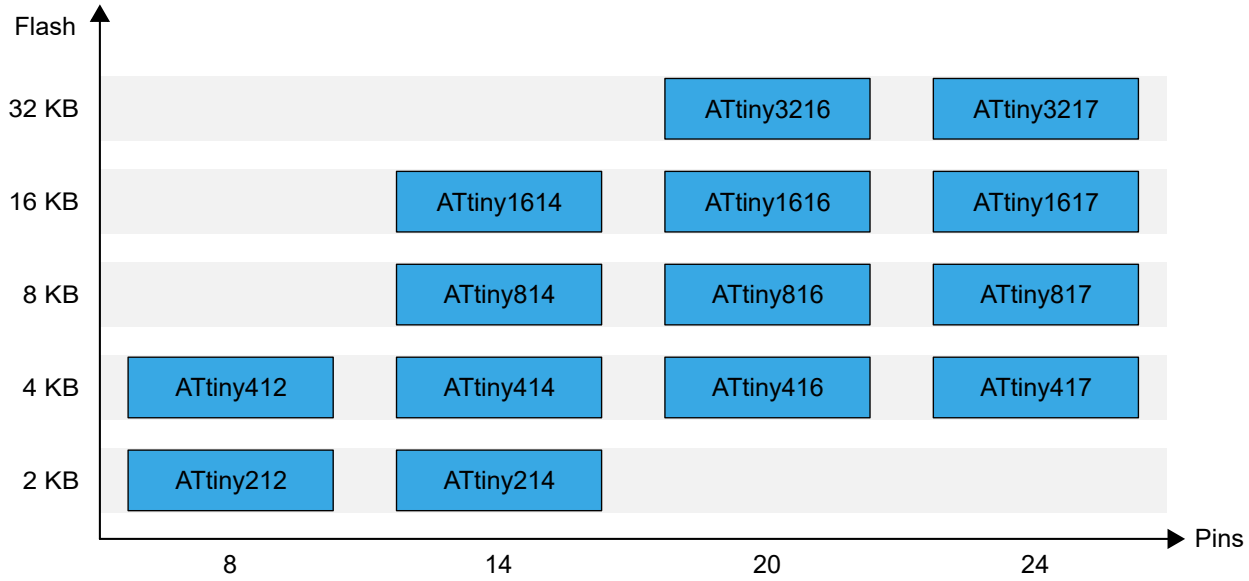


Figure 1-3. tinyAVR® 2 Family Overview

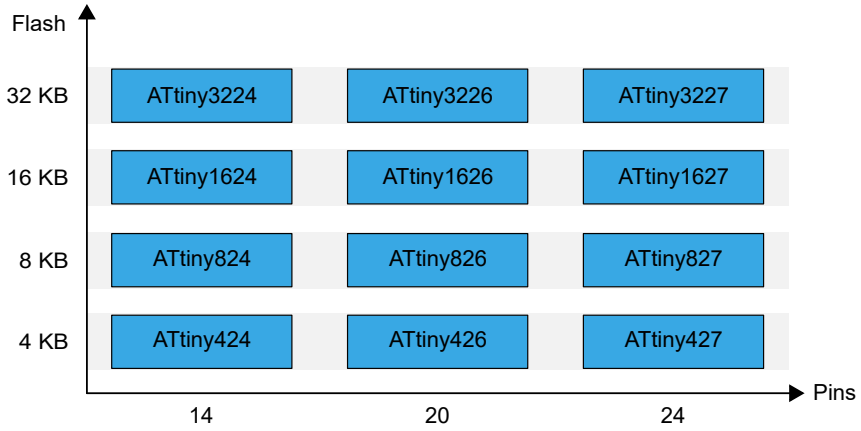


Figure 1-4. megaAVR® 0-series Overview

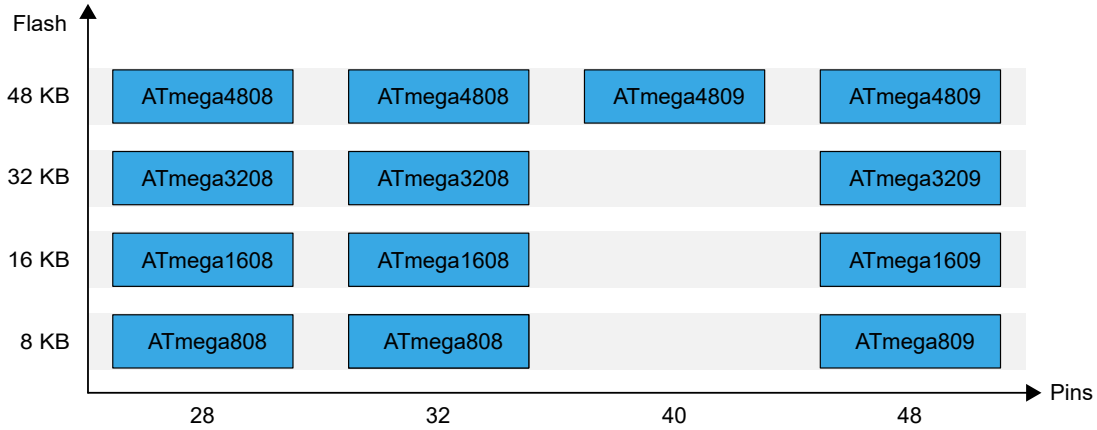


Figure 1-5. AVR® DA Family Overview

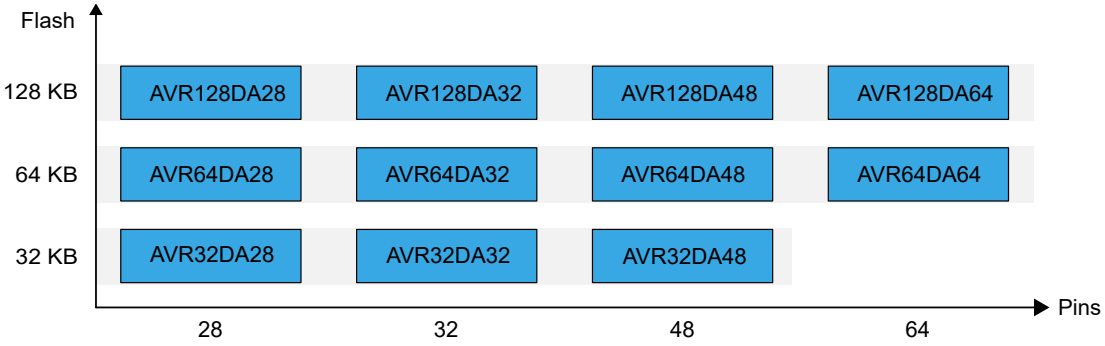
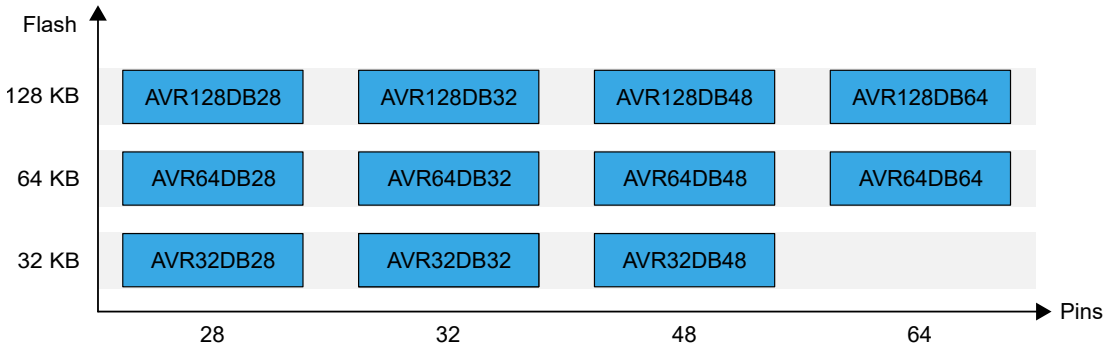


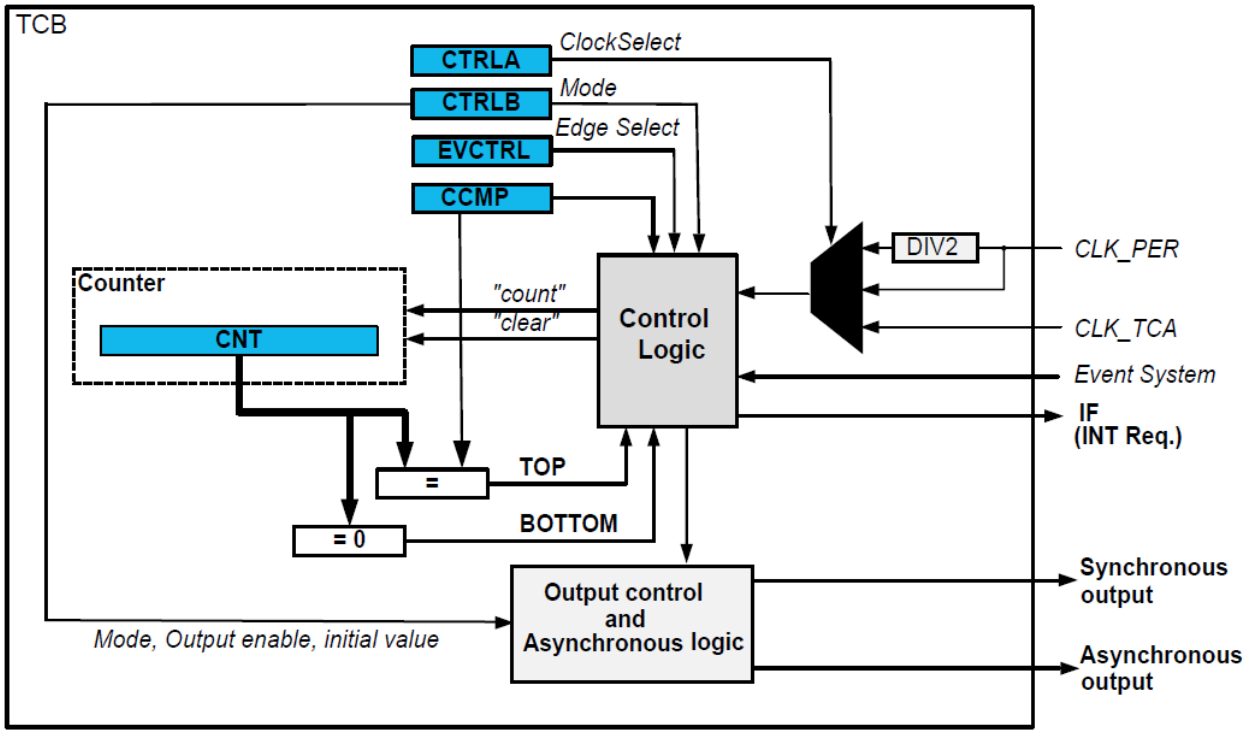
Figure 1-6. AVR® DB Family Overview



2. Overview

The capabilities of the 16-bit Timer/Counter type B (TCB) include frequency, waveform generation, and input capture on an event with time and frequency measurement of the digital signals. The TCB consists of a base counter and a control logic that can be set in one of the eight different modes, each mode providing unique functionality. The base counter is clocked by the peripheral clock with optional prescaling, as shown below.

Figure 2-1. Timer/Counter Type B Block Diagram



3. Using TCB in 8-Bit PWM Mode

Use case description: TCB (the TCB3 instance) will be configured in 8-bit PWM mode and generate a one second period PWM signal, at a 50% duty cycle. A GPIO pin (Port B pin 5 - PB5) will be used as an output to showcase the signal.

Result: TCB will generate a 50% duty cycle PWM signal with a period of one second. The on-board LED (the PB5 pin) will toggle every 500 ms.

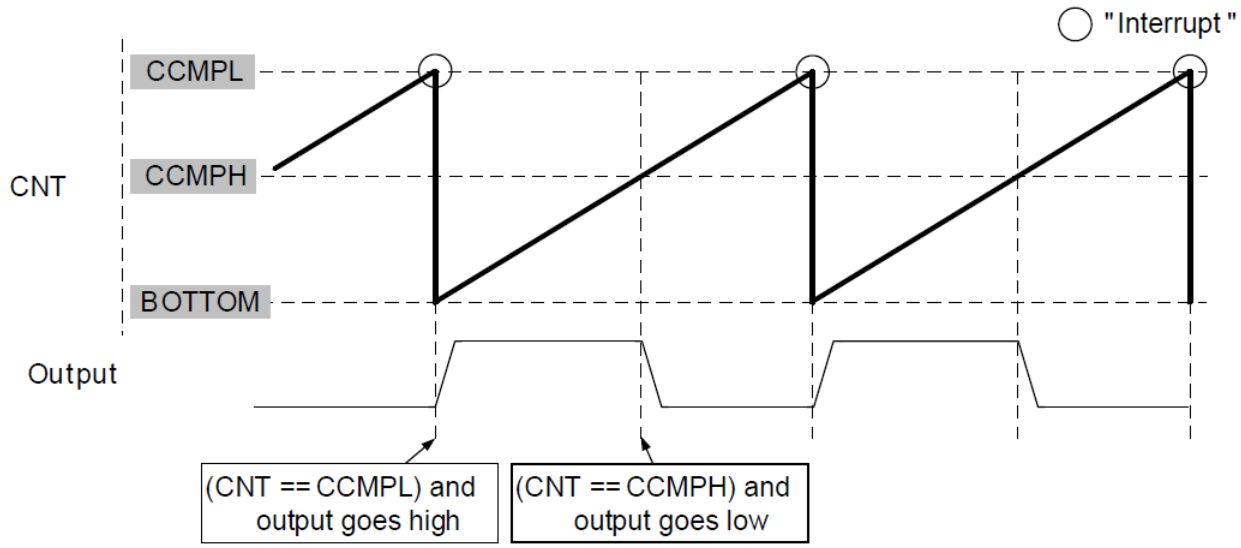
This timer can be configured to run in 8-bit PWM mode where each of the register pairs in the 16-bit Capture/Compare register (TCBn.CCMPH and TCBn.CCMPL) is used as an individual Compare register. The counter will continuously count from zero to CCMPL and the output will be set at BOTTOM and cleared when the counter reaches CCMPH.

When this peripheral is in PWM mode and enabled, changing the value of the Capture/Compare register will change the output, but the transition may render invalid values. It is hence recommended to:

1. Disable the peripheral.
2. Write Capture/Compare register to {CCMPH, CCMPL}.
3. Write `0x0000` to the Count register.
4. Re-enable the module.

CCMPH is the number of cycles for which the output will be driven high, while CCMPL+1 is the period of the output pulse.

Figure 3-1. 8-Bit PWM Mode



For example, for a 256 Hz frequency clock that serves as input to the timer, the result is a period of one second (CCMPL is an 8-bit register, which means it can have values from 0 to 255).

Configuring a Pin as Output for Visualizing the PWM Signal

To visualize the PWM, a pin will be configured in the Output mode. The following code sets PB5 as output low.

```
PORTB_DIR |= PIN5_bm;
PORTB_OUT |= PIN5_bm;
```

Configuring the System Clock

According to the diagram in Figure 2-1, there are two main clock sources for TCB:

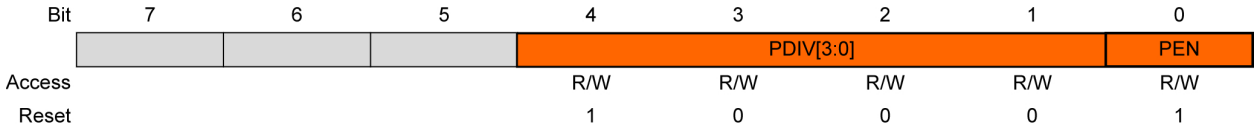
- CLK_PER (the peripheral clock, derived from main clock CLK_MAIN)
- CLK_TCA (the TCA clock, which can be derived from CLK_PER)

For this use case of TCB, the CLK_PER clock source and instance 3 of TCB will be used (TCB3). To get a period of one second, the input clock must be as low as possible. The following configuration must be made:

- Internal 32 kHz ultra-low power oscillator for CLKSEL must be selected
- The clock prescaler (PEN) must be enabled
- The highest prescaler division (PDIV 64) must be used

To use the 32 kHz internal oscillator as the clock source for TCB, the user must configure the following registers and bits or bit fields in the following register.

Figure 3-2. MCLKCTRLB Register Configuration



bits 4:1 PDIV[3:0]: Prescaler Division bits
If the Prescaler Enable (PEN) bit is written to '1', these bits define the division ratio of the main clock prescaler.

Value	Description
Value	Division
0x5	64

bit 0 PEN: Prescaler Enable bit
This bit must be written '1' to enable the prescaler. When enabled, the division ratio is selected by the PDIV bit field.

The Main Clock and Prescaler Configuration (CLKCTRL.MCLKCTRLA, CLKCTRL.MCLKCTRLB) registers are protected by the Configuration Change Protection (CCP) mechanism, employing a timed-write procedure for changing these registers. To write to these registers, a certain key must be written to the CPU.CCP register first, followed by a write access to the protected bits within four CPU instructions.

Since the desired frequency is the lowest possible, the Prescaler (PEN) must be enabled and PDIV must be set to '64' divider.

The key that must be written to the CPU.CPP register is IOREG. For more details, check the *Configuration Change Protection (CCP)* section in the megaAVR 0-series family data sheet.

Writing to a protected register is done by using the `ccp_write_io` function, which translates into the line of code from below for enabling the '64' prescaler divider.

Note: To use the `ccp_write_io` function, the `avr/cpufunc.h` header file must be included.

```
ccp_write_io((void *) &(CLKCTRL.MCLKCTRLB), (CLKCTRL_PDIV_64X_gc | CLKCTRL_PEN_bm));
```

Figure 3-3. MCLKCTRLA Register Configuration

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

bits 1:0 CLKSEL[1:0]: Clock Select bits
This bit field selects the source for the Main Clock (CLK_MAIN).

Value	Name	Description
0x0	OSC20M	20 MHz internal oscillator
0x1	OSCULP32K	32 KHz internal ultra low-power oscillator
0x2	XOSC32K	32.768 kHz external crystal oscillator
0x3	EXTCLK	External clock

OSCULP32K must be selected, which means the CLKSEL bit field must be set to the value `0x1`. This translates into the following code.

```
ccp_write_io((void *) &(CLKCTRL.MCLKCTRLA) , (CLKCTRL_CLKSEL_OSCULP32K_gc));
```

Figure 3-4. MCLKSTATUS Register Configuration

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 0 SOSC: Main Clock Oscillator Changing bit

Value	Description
0	The clock source for CLK_MAIN is not undergoing a switch
1	The clock source for CLK_MAIN is undergoing a switch and will change as soon as the new source is stable

The clock switching process is indicated by the SOSC bit. The program must halt during an undergoing switch of the clock source, so a wait until the switch is over will be implemented:

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

Configuring the TCB Input Clock and Operation Mode

To generate the 8-bit PWM signal to PB5, the following registers must be changed:

- TCBn.CCMP
- TCBn.CTRLA
- TCBn.CTRLB

The TCBn.CCMPL and TCBn.CCMPH register pair represents the 16-bit value TCBn.CCMP. 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`. In 8-bit PWM mode, TCBn.CCMPL and TCBn.CCMPH act as two independent registers.

Figure 3-5. TCBn.CCMP Register Configuration

Bit	15	14	13	12	11	10	9	8
	CCMP[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

bits 15:8 CCMP[15:8]: Capture/Compare Value High Byte
These bits hold the MSB of the 16-bit compare, capture, and top value.

bits 7:0 CCMP[7:0]: Capture/Compare Value Low Byte
These bits hold the LSB of the 16-bit compare, capture, and top value.

When running TCB in 8-bit PWM mode, TCBn.CCMPL must be loaded with the PWM signal period, of one second in this case. Since the period of the output pulse is defined by TCBn.CCMPL+1, the value that must be loaded into the TCBn.CCMPL register is `0xFF` (255 in decimal).

Subsequently, TCBn.CCMPH must be loaded with the number of cycles for which the output will be driven high. The goal is to set the duty cycle at 50% to make PB5 toggle every 500 ms.

$$CCMPH = \left(CCMPL + 1\right) \times \frac{50}{100}$$

$$CCMPH = 128$$

$$\text{This results in TCBn.CCMPH} = 128 = 0\text{x}80$$

This means that TCBn.CCMP must be loaded with the following value, obtained from TCBn.CCMPH and TCBn.CCMPL:

```
TCB3.CCMP = 0x80FF;
```

Figure 3-6. TCBn.CTRLA Register Configuration

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

Bit 6 – RUNSTDBY Run in Standby
Writing a '1' to this bit will enable the peripheral to run in Standby sleep mode. Not applicable when CLKSEL is set to 0x2 (CLK_TCA).

Bit 4 – SYNCUPD Synchronize Update
When this bit is written to '1', the TCB will restart whenever TCA0 is restarted or overflows. This can be used to synchronize capture with the PWM period.

Bits 2:1 – CLKSEL[1:0] Clock Select
Writing these bits selects the clock source for this peripheral.

Value	Name	Description
0x0	CLKDIV1	CLK_PER
0x1	CLKDIV2	CLK_PER / 2
0x2	CLKTCA	Use CLK_TCA from TCA0
0x3		Reserved

Bit 0 – ENABLE Enable
Writing this bit to '1' enables the Timer/Counter type B peripheral.

TCB3 can be enabled by setting the ENABLE bit to '1' in the TCB3.CTRLA register. This translates into the following code.

```
TCB3.CTRLA |= TCB_ENABLE_bm;
```

To get the lowest possible frequency, CLK_PER will be further divided by 2, by configuring the CLKSEL bit field in the TCB3.CTRLA register. The corresponding value for CLKSEL, in this case, is 0x1. This translates into the following code.

```
TCB3.CTRLA |= TCB_CLKSEL_CLKDIV2_gc;
```

Figure 3-7. TCBn.CTRLB Register Description

Bit	7	6	5	4	3	2	1	0
		ASYNC	CCMPINIT	CCMPEN		CNTMODE[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		0	0	0

bit 4 CCMPEN: Compare/Capture Output Enable bit
This bit is used to enable the output signal of the Compare/Capture.

Value	Description
0	Compare/Capture Output is zero
1	Compare/Capture Output has a valid value

bits 2:0 CNTMODE[2:0]: Timer Mode bits
Writing these bits selects the Timer mode.

Value	Description
0x0	Periodic Interrupt mode
0x1	Time-out Check mode
0x7	8-Bit PWM mode

CCMPEN must be enabled. This translates into the following code.

```
TCB3.CTRLB |= TCB_CCMPEN_bm;
```

TCB must be configured for the 8-bit PWM mode. This translates into the following code.

```
TCB3.CTRLB |= TCB_CNTMODE_PWM8_gc;
```

Tip: The full code example is also 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

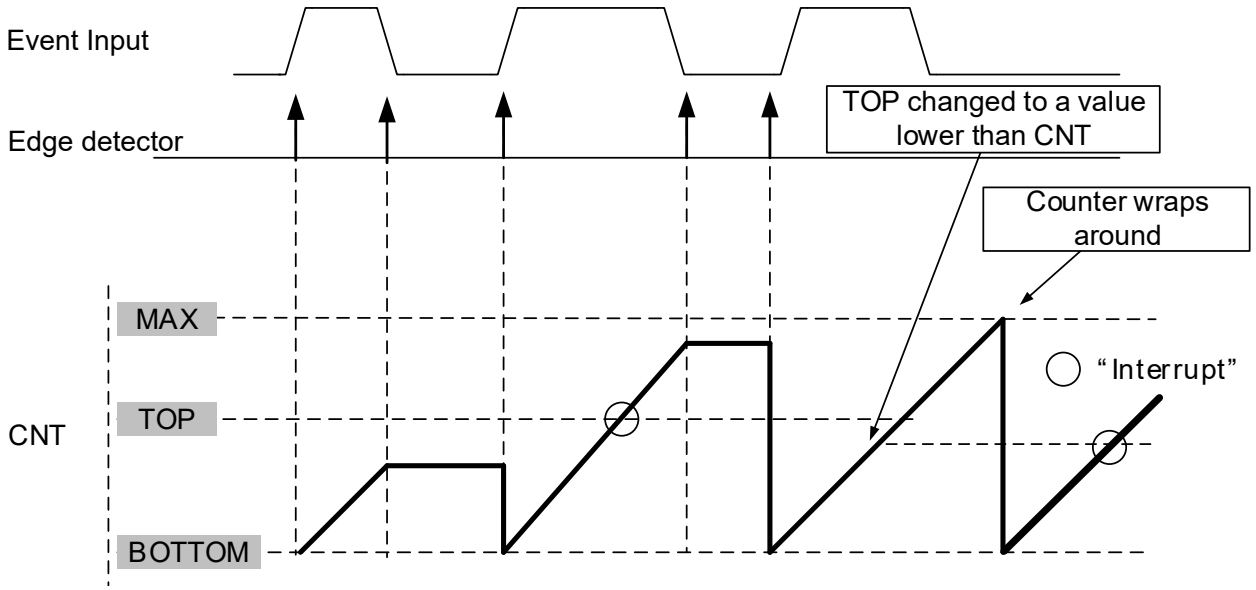
4. Using TCB in Time-Out Check Mode

Use case description: Configure TCB in Time-Out Check mode and measure the signal time generated by a GPIO (configured as an input). The time-out will be set to one second and the event system will be used to detect the rising and falling edges of the signal generated by a GPIO (e.g., a push button in a real application). If the time-out expires, another GPIO (e.g., an LED in a real application) will be toggled in the interrupt.

Result: Use a GPIO as input to generate an event that serves as the input signal for TCB. Generate an interrupt when the time-out expires.

In the Time-Out Check mode, TCB relies on the interaction with the event system, as displayed in the figure below. In this mode, the counter counts to MAX and wraps around. On the first edge, the counter is restarted. On the second edge, the counter is stopped. If the Count (TCBn.CNT) register reaches TOP before the second edge, an interrupt will be generated. In the Freeze state, the counter will restart on a new edge. Reading count (TCBn.CNT) or the Capture/Compare (TCBn.CCMP) register, or writing the RUN bit (RUN in TCBn.STATUS) in Freeze state will have no effect.

Figure 4-1. Time-Out Check Mode



Configuring the System Clock

To obtain a period of one second for the 16-bit TCB timer, the input frequency must be as low as possible. For this, the internal 32 kHz oscillator can be used. There is no need to use a frequency divider, so the prescaler for CLK_PER must be disabled. There are three steps in the configuration process:

1. **Disable the CLK_PER prescaler** - The following code snippet will demonstrate how to disable the CLK_PER prescaler.

```
ccp_write_io( (void *) &CLKCTRL.MCLKCTRLB , (0 << CLKCTRL_PEN_bp));
```

2. **Select the internal 32 kHz oscillator** - The following code snippet will switch the system clock source to the internal 32 kHz oscillator.

```
ccp_write_io( (void *) &CLKCTRL.MCLKCTRLA , (CLKCTRL_CLKSEL_OSCULP32K_gc));
```

3. **Wait for the clock switch process to complete** - The following code snippet will demonstrate how to wait for the clock source switching process to finish.

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

Notes:

1. Writing to the MCLKCTRLA and MCLKCTRLB registers requires the IOREG key to be written to the CPU.CCP register. This is done by using the `ccp_write_io` function from the `avr/cpufunc.h` header file.
2. The Clock Control registers are described in 3. Using TCB in 8-Bit PWM Mode.

Configuring TCB in Time-Out Check Mode

To generate a signal using a GPIO pin, the event system must be set up accordingly. Port B Pin 2 (PB2) will be used as an example in this document.

The following registers must be configured:

- TCB0.CCMP
- TCBn.INTCTRL
- TCBn.EVCTRL
- EVSYS.CHANNEL
- EVSYS.USER

1. **TCB0.CCMP Configuration**
   The TCB Capture/Compare register must be loaded with the value of the time-out. For this document, a value of one second for the time-out has been chosen. With an input frequency of 32.768 kHz, the 16-bit counter will make a complete cycle in two seconds (the maximum value of a 16-bit number is 65535).

   The reload value of TCB0.CCMP can be calculated, as shown below:

   $$CCMP = \frac{CNT\ \max}{2}$$
   $$CCMP = \frac{65535}{2}$$
   $$CCMP = 32767.5$$

   A value of 32767.5 cannot be translated into hex, so 32767 will be used instead.

   $$CCMP = 32767.5 = \sim 32767 = 0 \times 7FFF$$

   The following code snippet will load TCB0.CCMP with `0x7FFF` time-out value.

   ```
   TCB0.CCMP = 0x7fff;
   ```

2. **TCBn.INTCTRL Configuration**

   Figure 4-2. TCBn.INTCTRL Register Configuration

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

   bit 0 CAPT: Capture Interrupt Enable bit
   Writing this bit to '1' enables the Capture interrupt.

   TCB0.CNT will start increasing as soon as a signal edge is detected on the event system bus. If the complementary edge of the signal is not detected within the time-out, TCB0 will trigger an interrupt. In this regard, capture or time-out interrupt must be enabled in the TCB0.INCTRL register. The following code snippet enables the interrupt.

   ```
   TCB0.INTCTRL = TCB_CAPT_bm;
   ```

   Note: The global interrupts must also be enabled. This can be done at a later step in the software program.

3. **TCBn.EVCTRL Configuration**

Figure 4-3. TCBn.EVCTRL Register Configuration

Bit	7	6	5	4	3	2	1	0
		FILTER		EDGE				CAPTEI
Access		R/W		R/W				R/W
Reset		0		0				0

bit 4 EDGE: Event Edge bit
This bit is used to select the event edge. The effect of this bit is dependent on the selected Count Mode (CNTMODE) in TCBn.CTRLB. "-" means that an event or edge has no effect in this mode.

Count Mode	EDGE	Positive Edge	Negative Edge
Timeout Check mode	0	Start counter	Stop counter
	1	Stop counter	Start counter

bit 0 CAPTEI: Capture Event Input Enable bits
Writing this bit to '1' enables the input capture event.

The target is to measure the signal time between the falling edge and the rising edge of the PB2 input pin (in Idle, the pin is kept in high state). This means both CAPTEI and EDGE bits must be set to '1'. The following code snippet enables both bits in the TCB0.EVCTRL register.

```
TCB0.EVCTRL = TCB_CAPTEI_bm | TCB_EDGE_bm;
```

4. **EVSYS.CHANNEL Configuration**
Each channel can be connected to a one-event generator. Not all generators can be connected to all channels. Refer to the table below to see which generator sources can be routed onto each channel and the generator value that must be written to EVSYS.CHANNELn to achieve this routing. The value 0x00 in EVSYS.CHANNELn turns the channel off.

Figure 4-4. EVSYS.CHANNEL Register Configuration

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

bits 7:0 GENERATOR[7:0]: Channel Generator Selection bits

GENERATOR		INPUT	Async/Sync	CH0	CH1	CH2	CH3	CH4	CH5	CH6	CH7
binary	hex										
0100_0nnn	0x40-0x47	PORT0_PINn	Async	PORTA_PINn		PORTC_PINn		PORTE_PINn			
0100_1nnn	0x48-0x4F	PORT1_PINn	Async	PORTB_PINn		PORTD_PINn		PORTF_PINn			

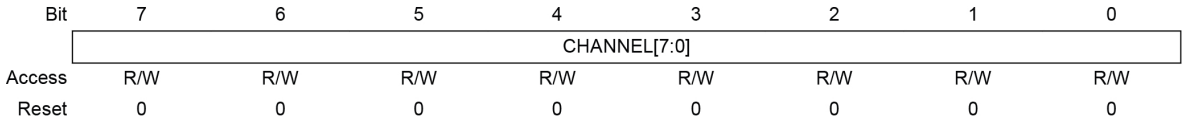
EVSYS.CHANNEL must be loaded with the value 0x4A for PB2 to generate an input event, as represented in the following code snippet.

```
EVSYS.CHANNEL0 = EVSYS_GENERATOR_PORT1_PIN2_gc;
```

5. **EVSYS.USER Configuration**
Each event user can be connected to one channel. Several users can be connected to the same channel. The following table lists all event system users, with their corresponding user ID number. This ID number corresponds to the USER register index, e.g., the user with ID 2 is controlled by the EVSYS.USER2 register.

Figure 4-5. EVSYS.USERTCB0 Register Configuration

USER #	User Name	Async/Sync	Description
20	TCB0	Async	TCB0 Event in



bits 7:0 CHANNEL[7:0]: User Channel Selection bits
Describes which event system channel the user is connected to.

Value	Description
0	OFF, no channel is connected to this event system user
n	Event user is connected to CHANNEL(n-1)

The user channel must be linked to the event channel already configured previously (in EVSYS.CHANNEL). The following code snippet illustrates how this can be done.

```
EVSYS.USERTCB0 = EVSYS_CHANNEL_CHANNEL0_gc;
```

Tip: The full code example is also 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

5. Using TCB in Sleep Mode

Use case description: Configure TCB to generate an overflow interrupt every second, even when the microcontroller is in Standby sleep mode, and toggle a GPIO pin (PB5).

Result: TCB overflow interrupt will be triggered every one second, regardless of the running mode of the microcontroller (Idle/Standby). In the Interrupt Service Routine (ISR), the on-board LED, corresponding to PB5, will be toggled.

TCBn is by default disabled in Standby sleep mode; as soon as the microcontroller enters sleep mode, TCB will be halted. The module can stay fully operational in Standby sleep mode if the Run In Standby (RUNSTDBY) bit in the TCBn.CTRLA register is written to '1'.

Configuring the System Clock for Sleep Mode

To obtain an interrupt of one second for the 16-bit TCB timer, the input frequency must be as low as possible. For this, the internal 32 kHz oscillator can be used. There is no need to use a frequency divider, so the prescaler for CLK_PER must be disabled. There are three steps in the configuration process:

1. **Disable the CLK_PER prescaler** - The following code snippet will demonstrate how to disable the CLK_PER prescaler.

```
ccp_write_io( (void *) &CLKCTRL.MCLKCTRLB , (0 << CLKCTRL_PEN_bp));
```

2. **Select the internal 32 kHz oscillator** - The following code snippet will switch the system clock source to the internal 32 kHz oscillator.

```
ccp_write_io( (void *) &CLKCTRL.MCLKCTRLA, (CLKCTRL_CLKSEL_OSCULP32K_gc));
```

3. **Wait for the clock switch process to complete** - The following code snippet will demonstrate how to wait for the clock source switching process to finish.

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

Configuring the Sleep Mode Operation

Sleep modes are used to shut down peripherals and clock domains in the device to save power. The Sleep Controller (SLPCTRL) controls and handles the transitions between Active and sleep mode. Four modes are available: One Active mode in which software is executed and three sleep modes. The available sleep modes are Idle, Standby, and Power-Down.

The interrupts are used to wake the device from sleep. The available interrupt wake-up sources depend on the configured sleep mode. When an interrupt occurs, the device will wake up and execute the interrupt service routine before continuing normal program execution from the first instruction following the `SLEEP` instruction.

Figure 5-1. SLPCTRL.CTRLA Register

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 bits
Writing these bits selects the sleep mode entered when the Sleep Enable bit (SEN) 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 bit
This bit must be written to '1' before the SLEEP instruction is executed to make the MCU enter the selected Sleep mode.

The TCB module can run while the microcontroller is in Standby sleep mode. The Power-Down sleep mode disables TCB completely.

For this use case, the ATmega4809 microcontroller will be configured to enter sleep in Standby sleep mode. Entering sleep must also be enabled. The following code enables the sleep operation and configures the ATmega4809 microcontroller to enter Standby sleep mode when the SLEEP instruction is executed.

```
SLPCTRL.CTRLA = SLPCTRL_SMODE_gm | SLPCTRL_SMODE_STDBY_gc;
```

Configuring TCB in Periodic Interrupt Mode

The following configuration must be made to the registers:

- TCBn.CCMP
- TCBn.CTRLA

The TCB Capture/Compare register must be loaded with a comparison value which will trigger an interrupt. For this document, a value of one second for the periodic interrupt has been chosen. With an input frequency of 32.768 kHz, the 16-bit counter will make a complete cycle in two seconds (the maximum value of a 16-bit number is 65535). Since the maximum value the 16-bit TCB counter can achieve is 65535 for the two seconds maximum period, the CCMP register must be loaded with half of the maximum value – 32768 (0x7FFF). The following code snippet loads the TCB0.CCMP register with 0x7FFF.

```
TCB0.CCMP = 0x7fff;
```

Figure 5-2. TCBn.CTRLA Register Configuration

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

bit 6 RUNSTDBY: Run in Standby bit
Writing a '1' to this bit will enable the peripheral to run in Standby Sleep mode. Not applicable when CLKSEL is set to 0x2 (CLK_TCA).

Prescaling is not needed, so the divider value will be '0'. Also, the timer must be enabled. For the TCB to still generate an interrupt while in Standby sleep mode, the RUNSTDBY bit must also be enabled. The following code snippet sets the clock divider to '0', enables the timer, and allows the timer to run in Standby sleep mode.

```
TCB0.CTRLA = TCB_CLKSEL_CLKDIV1_gc | TCB_ENABLE_bm | TCB_RUNSTDBY_bm;
```

To trigger a periodic interrupt, the CAPT bit of the TCB0.INTCTRL register must be enabled. The following code snippet enables the interrupt.

```
TCB0.INTCTRL = TCB_CAPT_bm;
```

Notes:

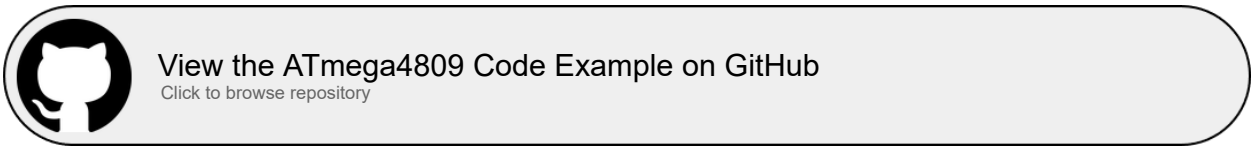
1. The global interrupts must be also enabled. This can be done at a later step in the software program.
2. All operations are halted in Power-Down sleep mode.

Configuring a Pin as Output for Visualizing Interrupt Occurrence

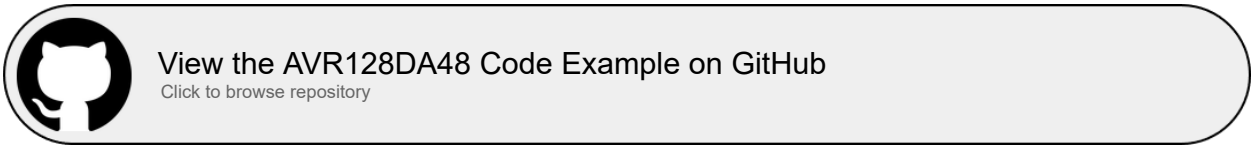
To visualize the periodic interrupt occurrence, the PB5 pin will be configured in Output mode. The following code sets PB5 as output low.

```
PORTB_DIR |= PIN5_bm;
PORTB_OUT |= PIN5_bm;
```

Tip: The full code example is also 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:



6. References

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. TCB 8-Bit PWM Mode Code Example

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

#define TCB_CMP_EXAMPLE_VALUE   (0x80FF)

void CLOCK_init (void);
void PORT_init (void);
void TCB3_init (void);

void CLOCK_init (void)
{
    /* Enable Prescaler and set Prescaler Division to 64 */
    ccp_write_io((void *) &(CLKCTRL.MCLKCTRLB), (CLKCTRL_PDIV_64X_gc |
CLKCTRL_PEN_bm));

    /* Select 32KHz Internal Ultra Low Power Oscillator (OSCULP32K) */
    ccp_write_io((void *) &(CLKCTRL.MCLKCTRLA) ,
(CLKCTRL_CLKSEL_OSCULP32K_gc));

    /* Wait for system oscillator changing to finish */
    while (CLKCTRL.MCLKSTATUS & CLKCTRL_SOSC_bm)
    {
        ;
    }
}

void PORT_init (void)
{
    PORTB_DIR |= PIN5_bm;
    PORTB_OUT |= PIN5_bm;
}

void TCB3_init (void)
{
    /* Load CCMP register with the period and duty cycle of the PWM */
    TCB3.CCMP = TCB_CMP_EXAMPLE_VALUE;

    /* Enable TCB3 and Divide CLK_PER by 2 */
    TCB3.CTRLA |= TCB_ENABLE_bm;
    TCB3.CTRLA |= TCB_CLKSEL_CLKDIV2_gc;

    /* Enable Pin Output and configure TCB in 8-bit PWM mode */
    TCB3.CTRLB |= TCB_CCMPEN_bm;
    TCB3.CTRLB |= TCB_CNTMODE_PWM8_gc;
}

int main(void)
{
    CLOCK_init();
    PORT_init();
    TCB3_init();

    while (1)
    {
        ;
    }
}
```

Example 7-2. TCB Time-Out Check Mode Code Example

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

void CLOCK_init (void);
void PORT_init (void);
void EVENT_SYSTEM_init (void);
void TCB0_init (void);
```

```
void CLOCK_init (void)
{
    /* Disable CLK_PER Prescaler */
    ccp_write_io( (void *) &CLKCTRL.MCLKCTRLB , (0 << CLKCTRL_PEN_bp));

    /* Select 32KHz Internal Ultra Low Power Oscillator (OSCULP32K) */
    ccp_write_io( (void *) &CLKCTRL.MCLKCTRLA , (CLKCTRL_CLKSEL_OSCULP32K_gc));

    /* Wait for system oscillator changing to finish */
    while (CLKCTRL.MCLKSTATUS & CLKCTRL_SOSC_bm)
    {
        ;
    }
}

void PORT_init (void)
{
    PORTB.DIR |= PIN5_bm;  /* Configure PB5 as digital output */
    PORTB.OUT |= PIN5_bm;  /* Set initial level of PB5 */
    PORTB.DIR &= ~PIN2_bm; /* Configure PB2 as digital input */
    PORTB.PIN2CTRL = PORT_PULLUPEN_bm; /* Enable the internal pullup */
}

void EVENT_SYSTEM_init (void)
{
    EVSYS.CHANNEL0 = EVSYS_GENERATOR_PORT1_PIN2_gc; /* Set Port 1 Pin 2 (PB2)
as input event*/
    EVSYS.USERTCB0 = EVSYS_CHANNEL_CHANNEL0_gc; /* Connect user to event
channel 0 */
}

void TCB0_init (void)
{
    /* Load the Compare or Capture register with the timeout value*/
    TCB0.CCMP = 0x7fff;

    /* Enable TCB and set CLK_PER divider to 1 (No Prescaling) */
    TCB0.CTRLA = TCB_CLKSEL_CLKDIV1_gc | TCB_ENABLE_bm;

    /* Configure TCB in Periodic Timeout mode */
    TCB0.CTRLB = TCB_CNTMODE_TIMEOUT_gc;

    /* Enable Capture or Timeout interrupt */
    TCB0.INTCTRL = TCB_CAPT_bm;

    /* Enable Event Input and Event Edge*/
    TCB0.EVCTRL = TCB_CAPTEI_bm | TCB_EDGE_bm;
}

ISR(TCB0_INT_vect)
{
    TCB0.INTFLAGS = TCB_CAPT_bm; /* Clear the interrupt flag */
    PORTB.IN = PIN5_bm; /* Toggle PB5 GPIO */
}

int main(void)
{
    CLOCK_init();
    PORT_init();
    EVENT_SYSTEM_init();
    TCB0_init();

    sei(); /* Enable Global Interrupts */

    while (1)
    {
        ;
    }
}
```

Example 7-3. TCB Sleep Mode Operation Code Example

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

#define TCB_CMP_EXAMPLE_VALUE   (0x7fff)

void CLOCK_init (void);
void SLPCTRL_init (void);
void PORT_init (void);
void TCB0_init (void);

void CLOCK_init (void)
{
    /* Disable CLK_PER Prescaler */
    ccp_write_io( (void *) &CLKCTRL.MCLKCTRLB , (0 << CLKCTRL_PEN_bp));

    /* Select 32KHz Internal Ultra Low Power Oscillator (OSCULP32K) */
    ccp_write_io( (void *) &CLKCTRL.MCLKCTRLA, (CLKCTRL_CLKSEL_OSCULP32K_gc));

    /* Wait for system oscillator changing to finish */
    while (CLKCTRL.MCLKSTATUS & CLKCTRL_SOSC_bm)
    {
        ;
    }
}

void SLPCTRL_init (void)
{
    /* Enable sleep mode and select Standby mode */
    SLPCTRL.CTRLA = SLPCTRL_SMODE_gm | SLPCTRL_SMODE_STDBY_gc;
}

void PORT_init (void)
{
    PORTB.DIR |= PIN5_bm; /* Configure PB5 as digital output */
    PORTB.OUT |= PIN5_bm; /* Set initial level of PB5 */
}

void TCB0_init (void)
{
    /* Load the Compare or Capture register with the timeout value*/
    TCB0.CCMP = TCB_CMP_EXAMPLE_VALUE;

    /* Enable TCB and set CLK_PER divider to 1 (No Prescaling) */
    TCB0.CTRLA = TCB_CLKSEL_CLKDIV1_gc | TCB_ENABLE_bm | TCB_RUNSTDBY_bm;

    /* Enable Capture or Timeout interrupt */
    TCB0.INTCTRL = TCB_CAPT_bm;
}

ISR(TCB0_INT_vect)
{
    TCB0.INTFLAGS = TCB_CAPT_bm; /* Clear the interrupt flag */
    PORTB.IN = PIN5_bm; /* Toggle PB5 GPIO */
}

int main(void)
{
    CLOCK_init();
    SLPCTRL_init();
    PORT_init();
    TCB0_init();

    sei(); /* Enable Global Interrupts */

    while (1)
    {
        sleep_mode();
    }
}
```

8. Revision History

Document Revision	Date	Comments
C	06/2022	Added AVR® DB and tinyAVR® 2 to relevant devices.
B	02/2021	Updated the GitHub repository links, the *References* section, and the use cases sections. Added the *AVR® DA Family Overview* and *Revision History* sections. Added MCC versions for each use case, running on AVR128DA48. Other minor corrections.
A	05/2019	Initial document release.

Microchip Information

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 products:

- 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, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip product is strictly prohibited and may violate the Digital Millennium Copyright Act.
- 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. Microchip is committed to continuously improving the code protection features of our products.

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications 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. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

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, CryptoMemory, CryptoRF, dsPIC, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, 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, TrueTime, 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, GridTime, IdealBridge, In-Circuit Serial Programming, ICSP, INICnet, Intelligent Paralleling, Inter-Chip Connectivity, JitterBlocker, Knob-on-Display, maxCrypto, maxView, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, NVM Express, NVMe, 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, SmartHLS, 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, Symmcom, and Trusted Time 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.

ISBN: 978-1-6683-0538-6

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