TB3212

Getting Started with Timer/Counter Type D (TCD)

Introduction

Author: Catalin Visan, Microchip Technology Inc.

The AVR® microcontrollers are equipped with powerful timers designed to cover a wide area of applications, from signal measurement to events synchronization and waveform generation.

The Timer/Counter type D (TCD) is a 12-bit timer available in tinyAVR® 1-series and AVR DA devices. The TCD is designed to cover the need for a fast timer with multiple waveform generation capabilities in common embedded applications, such as motor control or Switch mode power supplies. One of the key features of the TCD is that in addition to running from the main clock, it can be set up to run directly from the internal 20 MHz oscillator (OSC20M) or used with an external clock source. Moreover, it has multiple configurable Waveform Generation modes.

This document describes some of the TCD operating modes, emphasizing this timer's particularities and providing initialization code snippets. For a deeper understanding of the functionality, refer to the tinyAVR® 1-series and AVR DA data sheet. The structure of the document comprises two specific use cases:

- **Generating complementary driving signals:**
  Initialize the timer to generate two complementary signals with 50 kHz frequency and 100 ns dead-time.
- **Controlling synchronous signals using input events:**
  Initialize the timer to generate four PWM signals with 10 kHz frequency and approximately 50% duty cycle, synchronized in pairs. Configure an input channel for fault detection purposes.

Note: For each of the use cases described in this document, there are two code examples: One bare metal developed on ATtiny817, and one generated with MPLAB® Code Configurator (MCC) developed on AVR128DA48.

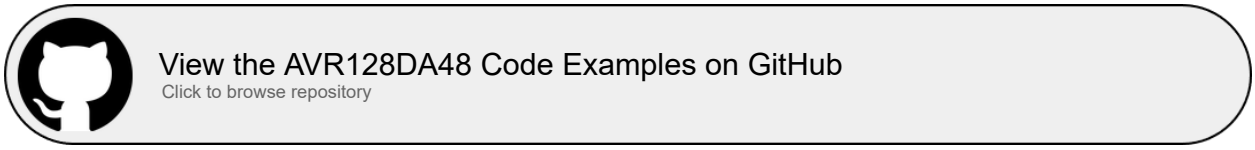


Table of Contents

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® 1-series Overview

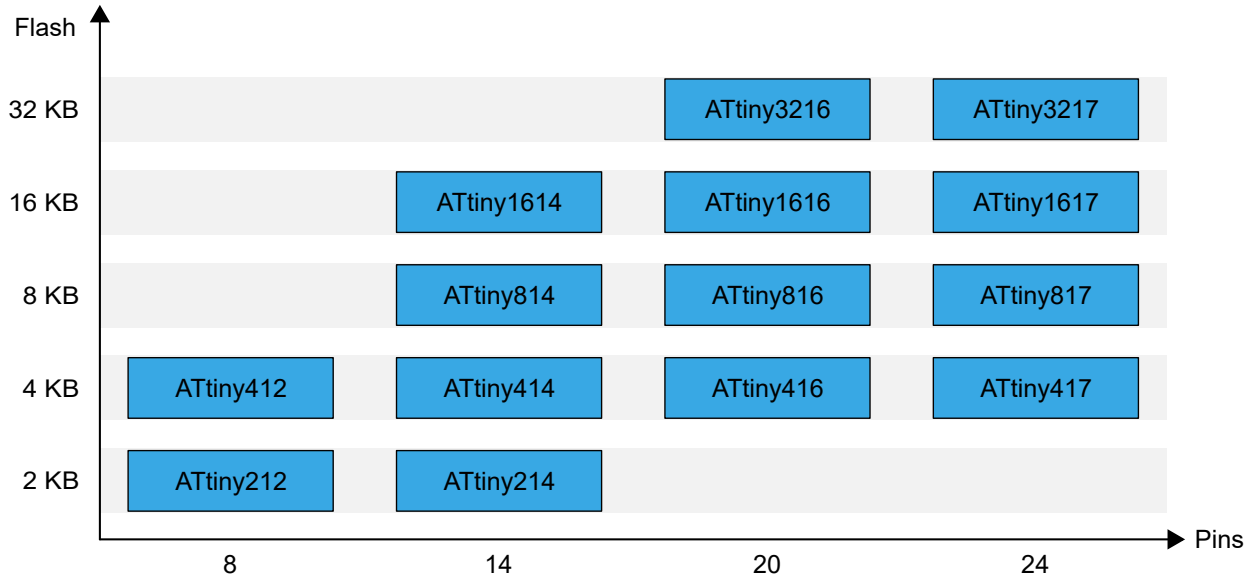
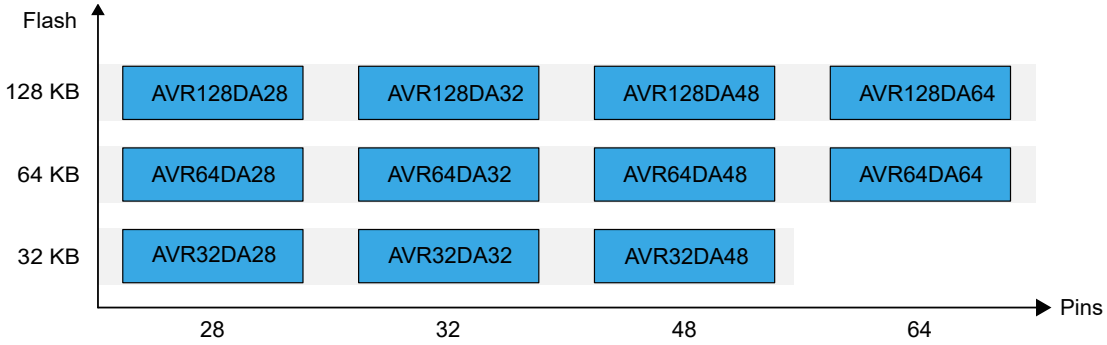


Figure 1-2. AVR® DA Family Overview



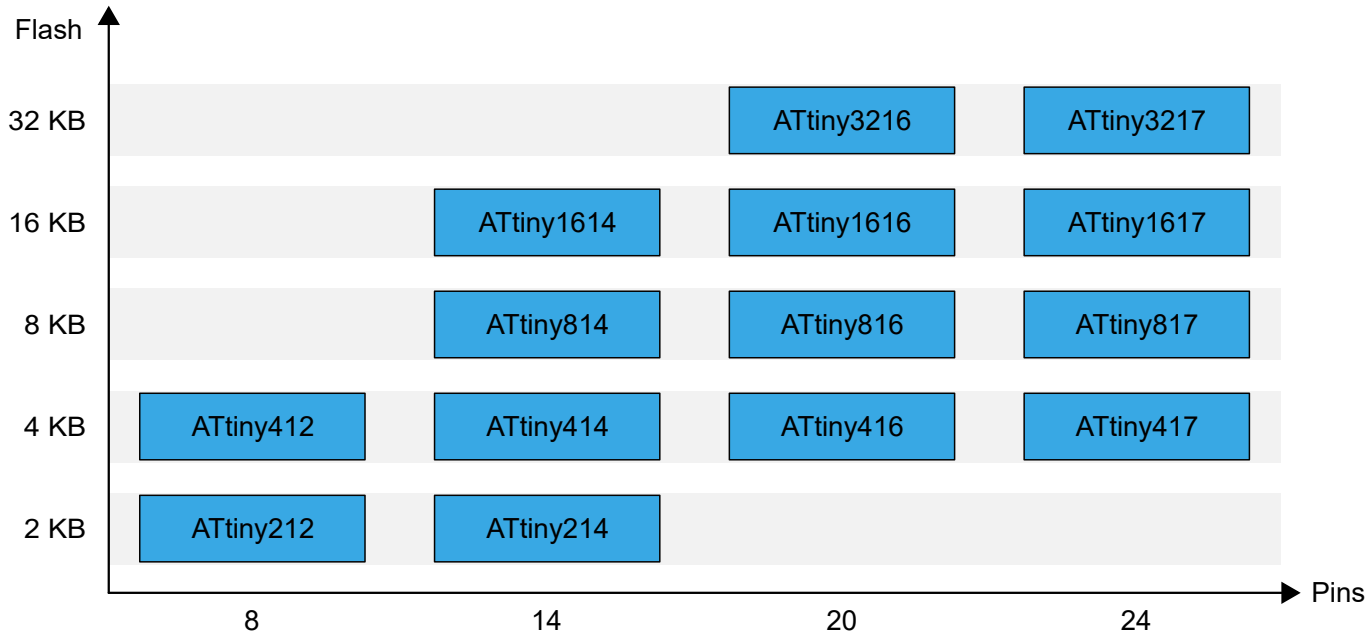
1.1 tinyAVR® 1-series

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 1-3. tinyAVR® 1-series Overview



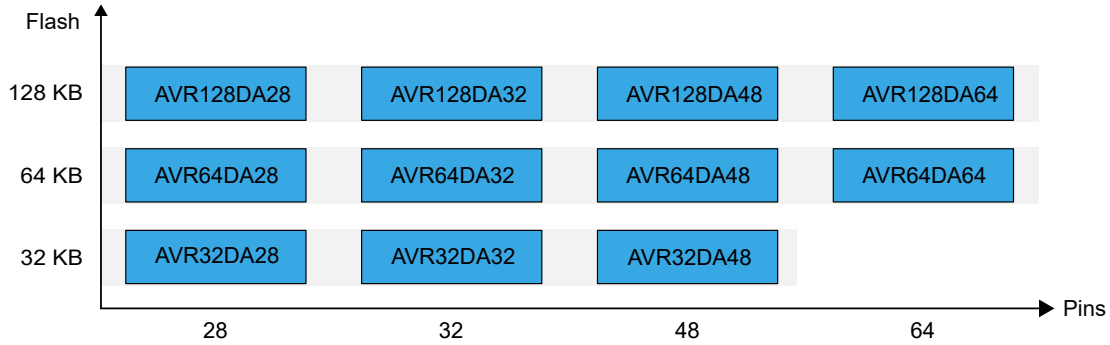
Devices with different Flash memory sizes typically also have different SRAM and EEPROM.

1.2 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 1-4. AVR® DA Family Overview

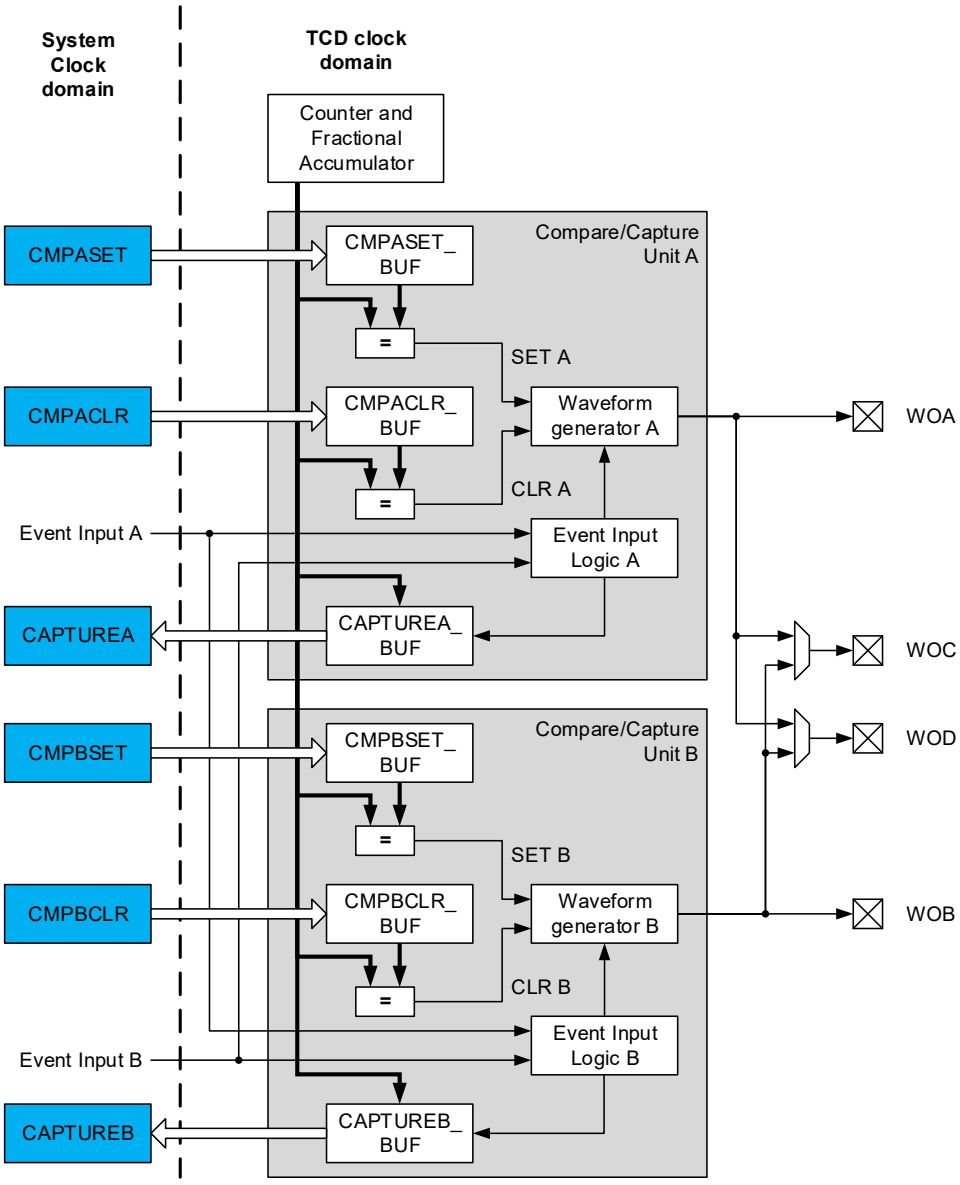


Devices with different Flash memory sizes typically also have different SRAM.

2. Overview

The TCD is a high-performance waveform controller that consists of an asynchronous counter, a prescaler, compare logic, capture logic, and control logic. The TCD contains a counter that can run on a clock that is asynchronous to the system clock. It contains compare logic that can generate two independent outputs with optional dead-time. It is connected to the event system for capture and deterministic fault control. The timer/counter can generate interrupts and events on compare match and overflow.

Figure 2-1. TCD Block Diagram



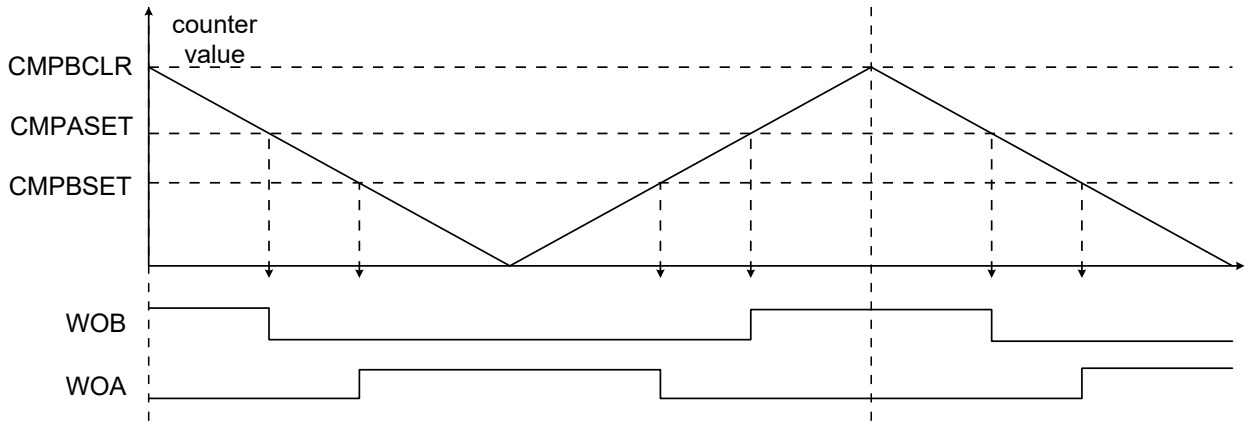
The TCD core is asynchronous to the system clock. The timer/counter consist of two compare/capture units, each with a separate waveform output. In addition, there are two extra waveform outputs that can be equal to the output from one of the units. The compare registers CMPxSET and CMPxCLR are stored in the respective registers (TCD.CMPxSET, TCD.CMPxCLR), which consist of both a low and a high byte. The registers are synchronized to the TCD domain after writing to the registers. During normal operation, the counter value is continuously compared to the compare registers. This is used to generate both interrupts and events.

The TCD can use the input events in ten different input modes, selected separately for the two input events (see Figure 4-6). The input mode defines how the input event will affect the outputs.

3. Generating Complementary Driving Signals

The 8-bit microcontrollers are commonly used in Switch mode power supplies. This use case shows how the TCD can be configured to generate complementary waveforms for the power MOSFET transistors. In this example, the TCD instance is configured to generate two complementary signals with 50 kHz frequency and 100 ns dead-time.

Figure 3-1. Dual Slope Mode



- The maximum counter value is stored in the CMPBLCR register (see Figure 3-1)
- The WOA output is set when the TCD counter counts down and matches the CMPASET value
- WOA is cleared when the TCD counter counts up and matches the CMPASET value
- The WOB output is set when the TCD counter counts up and matches the CMPBSET value
- WOB is cleared when the TCD counter counts down and matches the CMPBSET value

1. The Waveform Generation mode will be set to dual slope using the CTRLB register.

```
TCD0.CTRLB = TCD_WGMODE_DS_gc;
```

Figure 3-2. CTRLB Register

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

bits 1:0 WGMODE[1:0]: Waveform Generation Mode bits
These bits select the waveform generation.

Value	Name	Description
0x0	ONERAMP	One Ramp mode
0x1	TWORAMP	Two Ramp mode
0x2	FOURRAMP	Four Ramp mode
0x3	DS	Dual-Slope mode

2. The signal's period must be deduced from the following formula and written to the CMPBCLR register.

$$f_{signal} = \frac{f_{CLK}}{Prescaler_{CNT} \times 2 \times (CMPBCLR + 1)}$$

The peripheral clock will be set as the 20 MHz internal oscillator and the counter prescaler will be set to '1'.

$$CMPBCLR = \frac{f_{CLK}}{Prescaler_{CNT} \times 2 \times f_{signal}} - 1 = \frac{20 \times 10^6}{1 \times 2 \times 50 \times 10^3} - 1 \cong 200 = 0xC8$$

```
TCD0.CMPBCLR = 0xC8;
```

3. To set the dead-time, the counting period must be determined, as well as the number of counting periods that match the dead-time.

$$CNT_{period} = \frac{1}{f_{CNT}} = \frac{Prescaler_{CNT}}{f_{CLK}} = \frac{1}{20 \times 10^6} = 50ns$$

$$Dead\ time = 2 \times CNT_{period} = 100ns$$

Assuming the user wants both signals to have the same duty cycle.

$$CMPBSET = \frac{CMPBCLR}{2} + \frac{Dead\ time}{2} = 0x65$$

$$CMPASET = \frac{CMPBCLR}{2} - \frac{Dead\ time}{2} = 0x63$$

```
TCD0.CMPBSET = 0x65;

TCD0.CMPASET = 0x63;
```

4. Before enabling the timer, the ENRDY bit of the STATUS register must be verified to have the value '1'.

   The hardware sets this bit when it is safe to start the timer. This mechanism prevents synchronization issues between the TCD time domain and the system time domain.

```
while(!(TCD0.STATUS & TCD_ENRDY_bm))
{
    ;
}
```

Figure 3-3. STATUS Register

Bit	7	6	5	4	3	2	1	0
	PWMACTB	PWMACTA					CMDRDY	ENRDY
Access	R/W	R/W					R	R
Reset	0	0					0	0

5. After the ENRDY bit is set, the timer can be enabled from the CTRLA register.

   From the same register, the clock source and the prescaler can also be set. All the bits from CTRLA except the ENABLE bit are enable-protected, they can only be written when ENABLE is set to '0' before the writing operation.

```
TCD0.CTRLA = TCD_CLKSEL_20MHZ_gc
           | TCD_CNTPRES_DIV1_gc
           | TCD_ENABLE_bm;
```

Figure 3-4. CTRLA Register

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

bits 6:5 CLKSEL[1:0]: Clock Select bits
The clock select bits select the clock source of the TCD clock.

Value	Description
0x0	OSC20M
0x1	Reserved
0x2	External clock
0x3	System clock

bits 4:3 CNTPRES[1:0]: Counter Prescaler bits
The Counter Prescaler bits select the division factor of the TCD counter clock.

Value	Description
0x0	Division factor 1
0x1	Division factor 4
0x2	Division factor 32
0x3	Reserved

6. To enable the output channels, certain bits in the FAULTCTRL register may be set.
   This register is under Configuration Change Protection (CCP). Thus, a key must be written in the CCP register of the CPU before writing to the FAULTCTRL register. In this case, only channels A and B will be enabled.

```
void TCD0_enableOutputChannels(void)
{
        CPU_CCP = CCP_IOREG_gc;

        TCD0.FAULTCTRL = TCD_CMPAEN_bm
                       | TCD_CMPBEN_bm;
}
```

Figure 3-5. FAULTCTRL Register

Bit	7	6	5	4	3	2	1	0
	CMPDEN	CMPCEN	CMPBEN	CMPAEN	CMPD	CMPC	CMPB	CMPA
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	x	x	x	x	x	x	x	x

Info: After a Reset, the FAULTCTRL register loads its value from the fuses.

7. In the targeted microcontroller, the TCD output channels A and B are linked to PA4 and PA5. These pins must be configured as outputs:

```
PORTA.DIR |= PIN4_bm
           | PIN5_bm;
```

This application will configure the TCD instance to generate two complementary signals with 50 kHz frequency and 100 ns dead-time.

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

View the ATtiny817 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. Controlling Synchronous Signals Using Input Events

In motor control applications, it is often necessary to use perfectly synchronized signals. Also, there is a narrow class of applications that, for driving purposes, requires just a little bit more current than a pin can provide. In these cases, the possibility to map the same signal on multiple pins can reduce the bill of materials.

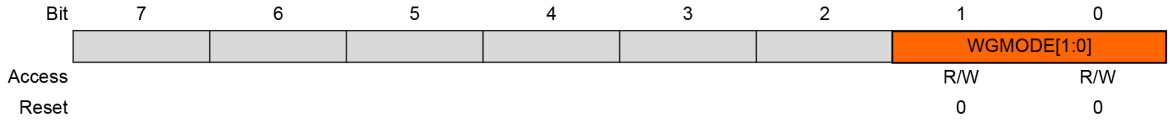
In most embedded systems there are different feedback loops and fault control components that ensure everything works properly. For reliability reasons, it is better to design these feedback mechanisms without the use of the MCU core. TCD has two event inputs that can be used to monitor the activity of other peripherals and components, and alter the output accordingly.

In this example, the TCD instance will be configured to generate four PWM signals with 10 kHz frequency and approximately 50% duty cycle, synchronized in pairs. In case of a fault signal on an input channel, the timer will stop and wait until the signal changes to the Safe state (no fault detected).

1. The timer will be configured from the CTRLB register (as in the previous use case), but this time in the Four Ramp mode:

```
TCD0.CTRLB = TCD_WGMODE_FOURRAMP_gc;
```

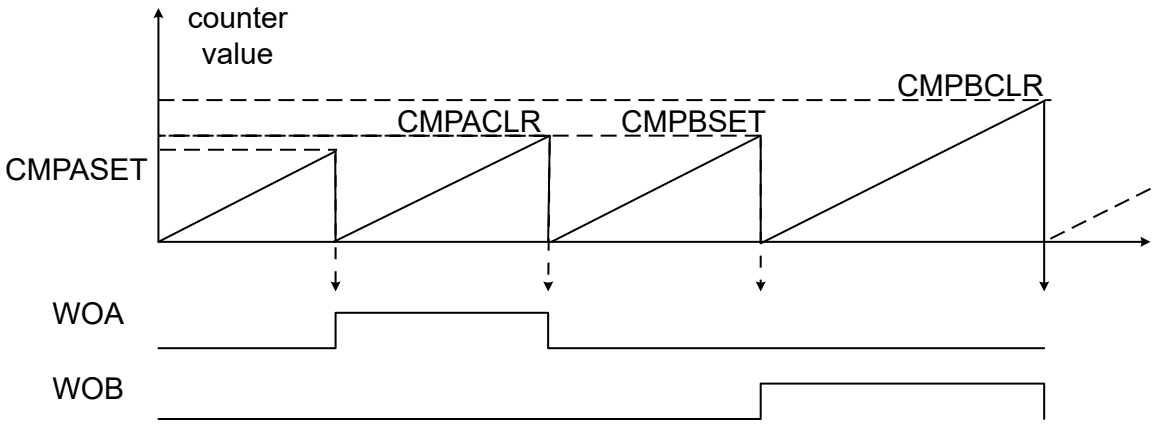
Figure 4-1. CTRLB Register



bits 1:0 WGMODE[1:0]: Waveform Generation Mode bits
These bits select the waveform generation.

Value	Name	Description
0x0	ONERAMP	One Ramp mode
0x1	TWORAMP	Two Ramp mode
0x2	FOURRAMP	Four Ramp mode
0x3	DS	Dual-Slope mode

Figure 4-2. Four Ramp Mode



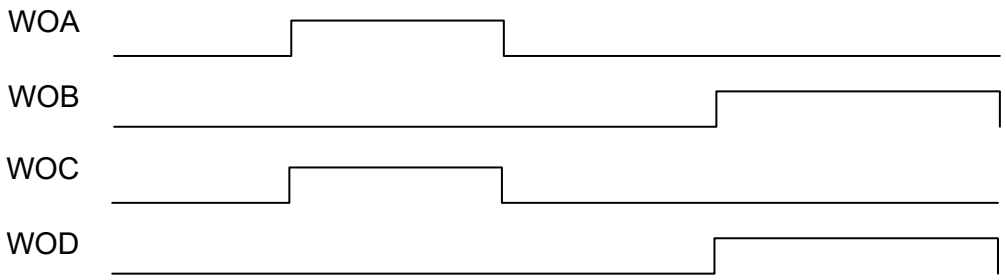
2. By default, channels C and D are linked to channel A. For the signals to be synchronized in pairs, the CMPDSEL bit from the CTRLC register must be set to link channel D to channel B (see Figure 4-4):

```
TCD0.CTRLC = TCD_CMPDSEL_bm;
```

Figure 4-3. CTRLC Register

Bit	7	6	5	4	3	2	1	0
	CMPDSEL	CMPCSEL			FIFTY		AUPDATE	CMPOVR
Access	R/W	R/W			R/W		R/W	R/W
Reset	0	0			0		0	0

Figure 4-4. Full-Bridge Output



3. The frequency can be computed using the following formula:

$$f_{PWM} = \frac{f_{CLK}}{Prescaler_{CNT} \times (CMPASET + CMPACLR + CMPBSET + CMPBCLR + 4)}$$

$$CMPASET + CMPACLR + CMPBSET + CMPBCLR = \frac{f_{CLK}}{f_{Prescaler} \times f_{PWM}} - 4 =$$

$$= \frac{20000000}{4 \times 10000} - 4 = 496 = 0x1F0$$

The targeted duty cycle is approximately 50%, so CMPACLR = CMPBCLR. For most applications, there is a settling time. Thus, the user must consider that and manually configure the peripheral to include a little dead-time. So CMPASET = CMPBSET = 2 for 0.4 µs settling time:

$$CMPACLR = CMPBCLR = \frac{0x1F0 - CMPASET - CMPBSET}{2} = \frac{0x1F0 - 2 - 2}{2} = 0xF6$$

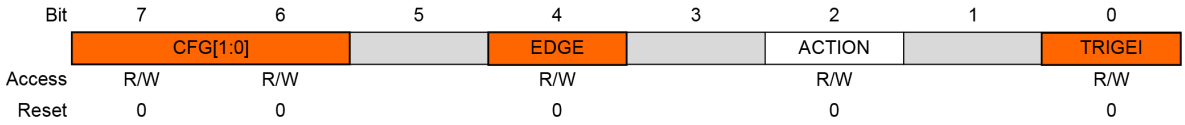
```
TCD0.CMPASET = 0x02;

TCD0.CMPACLR = 0xF6;

TCD0.CMPBSET = 0x02;

TCD0.CMPBCLR = 0xF6;
```

4. For fault detection purposes, input channel A of the timer will be configured to be active-low and the digital filter of the channel will be enabled to filter the potential spikes (see Figure 4-5).

   For this use case, the fault signal will be externally triggered by the press of a button, connected to pin PC5:

```
TCD0.EVCTRLA = TCD_CFG_FILTER_gc
             | TCD_EDGE_FALL_LOW_gc
             | TCD_TRIGEI_bm;
```

Figure 4-5. EVCTRL Register

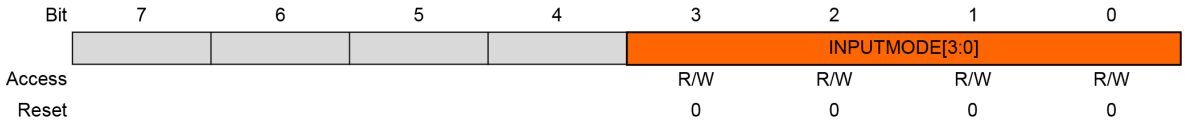


bits 7:6 CFG[1:0]: Event Configuration bits

Value	Name	Description
0x0	NEITHER	Neither filter nor asynchronous event is enabled.
0x1	FILTERON	Input capture noise cancellation filter enabled.
0x2	ASYNCON	Asynchronous event output qualification enabled.
other	-	Reserved.

5. The timer can be configured to respond to the input signal in various ways, using the INPUTCTRLA register.
   In this case, when a falling edge is detected on channel A, the timer is stopped and reset. The counter will restart at the next rising edge of the input signal.

Figure 4-6. INPUTCTRLA Register



bits 3:0 INPUTMODE[3:0]: Input Mode bits

Value	Name	Description
0x0	NONE	Input has no action
0x1	JMPWAIT	Stop output, jump to opposite compare cycle and wait
0x2	EXECWAIT	Stop output, execute opposite compare cycle and wait
0x3	EXECFAULT	Stop output, execute opposite compare cycle while fault active
0x4	FREQ	Stop all outputs, maintain frequency
0x5	EXECDT	Stop all outputs, execute dead time while fault active
0x6	WAIT	Stop all outputs, jump to next compare cycle and wait
0x7	WAITSW	Stop all outputs, wait for software action
0x8	EDGETRIG	Stop output on edge, jump to next compare cycle
0x9	EDGETRIGFREQ	Stop output on edge, maintain frequency
0xA	LVLTRIGFREQ	Stop output at level, maintain frequency

6. The ENRDY bit of the STATUS register must be '1' before starting the timer from the CTRLA register.
   Also, the 20 MHz clock is selected from CTRLA together with a prescaler of 4.

```
while(!(TCD0.STATUS & TCD_ENRDY_bm))
{
    ;
}

TCD0.CTRLA = TCD_CLKSEL_20MHZ_gc | TCD_CNTPRES_DIV4_gc | TCD_ENABLE_bm;
```

   In this example, all four output channels are used. The procedure is detailed within the previous use case.

```
void TCD0_enableOutputChannels(void)
{
        CPU_CCP = CCP_IOREG_gc;

        TCD0.FAULTCTRL = TCD_CMPAEN_bm | TCD_CMPBEN_bm | TCD_CMPCEN_bm | TCD_CMPDEN_bm;
}
```

7. The event system needs to be configured to link the event generator to the TCD instance.

In this case, the fault signal is simulated by pressing a button that is connected to PC5. The initialization code is provided below, but the event system configuration details are beyond the scope of this document.

```
void EVENT_SYSTEM_init(void)
{
        EVSYS.ASYNCCH2 = EVSYS_ASYNCCH2_PORTC_PIN5_gc;

        EVSYS.ASYNCUSER6 = EVSYS_ASYNCUSER6_ASYNCCH2_gc;
}
```

8. Channels A and B are linked with PA4 and PA5 pins, and channels C and D are linked with PC0 and PC1 pins. These pins must be configured as outputs.

   PC5, which is configured as input, is connected to the ATtiny817 Xplained Mini user button and has an internal pull-up resistor enabled.

   A pull-up resistor provides a default state of '1' to the pin. Thus, the button will be used to drive the pin low (logical '0') when it is pressed.

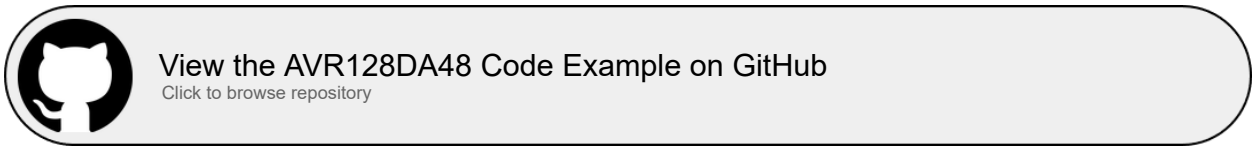
```
PORTA.DIR |= PIN4_bm
           | PIN5_bm;

PORTC.DIR |= PIN0_bm
           | PIN1_bm;

PORTC.DIR &= ~PIN5_bm;

PORTC.PIN5CTRL = PORT_PULLUPEN_bm;
```

This application will configure the TCD instance to generate four PWM signals with 10 kHz frequency at an approximately 50% duty cycle, synchronized in pairs.

Moreover, when a fault signal appears on an input channel, the timer will stop and wait until the signal changes to the Safe state.

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

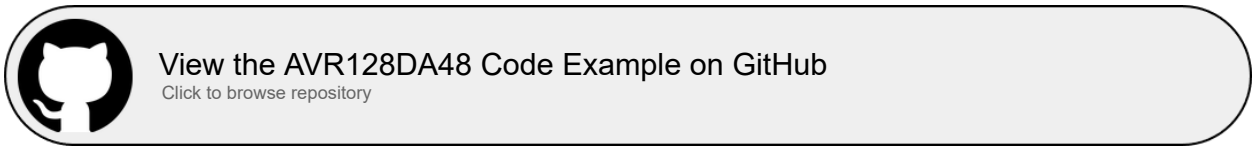
An MCC generated code example for AVR128DA48, with the same functionality as the one described in this section, can be found here:



5. Dithering Feature

The Timer/Counter type D (TCD) is equipped with a dithering feature. If it is not possible to achieve the desired frequency, dithering can be used to approximate the desired frequency and reduce the waveform drift. The dither accumulates the fractional error of the counter clock for each cycle. When the fractional error overflows, an additional clock cycle is added to the selected part of the TCD cycle.

An MCC generated code example for AVR128DA48, which emphases the dithering feature of the TCD, can be found here:



6. References

1. ATtiny817 web page: www.microchip.com/wwwproducts/en/ATTINY817
2. ATtiny417/817 - AVR® Microcontroller with Core Independent Peripherals and picoPower® Technology (DS40001901)
3. ATtiny817 Xplained Mini web page: www.microchip.com/DevelopmentTools/ProductDetails/ATTINY817-XMINI
4. AVR128DA48 product page: www.microchip.com/wwwproducts/en/AVR128DA48
5. AVR128DA48 Curiosity Nano Evaluation Kit product page: https://www.microchip.com/Developmenttools/ProductDetails/DM164151
6. AVR128DA28/32/48/64 Data Sheet
7. Getting Started with the AVR® DA Family

7. Appendix

Example 7-1. Generating Complementary Driving Samples Source Code

```
#define SIGNAL_PERIOD_EXAMPLE_VALUE            (0xC8)
#define SIGNAL_DUTY_CYCLE_EXAMPLE_VALUE        (0x64)

#include <avr/io.h>
/*Using default clock 3.3 MHz */

void TCD0_init(void);
void TCD0_enableOutputChannels(void);
void PORT_init(void);

void TCD0_init(void)
{
    /* set the waveform mode */
    TCD0.CTRLB = TCD_WGMODE_DS_gc;

    /* set the signal period */
    TCD0.CMPBCLR = SIGNAL_PERIOD_EXAMPLE_VALUE;

    /* the signals are alternatively active and a small symmetric dead time is
needed */
    TCD0.CMPBSET = SIGNAL_DUTY_CYCLE_EXAMPLE_VALUE + 1;
    TCD0.CMPASET = SIGNAL_DUTY_CYCLE_EXAMPLE_VALUE - 1;

    /* ensure the ENRDY bit is set */
    while(!(TCD0.STATUS & TCD_ENRDY_bm))
    {
        ;
    }

    TCD0.CTRLA = TCD_CLKSEL_20MHZ_gc        /* choose the timer's clock */
               | TCD_CNTPRES_DIV1_gc        /* choose the prescaler */
               | TCD_ENABLE_bm;             /* enable the timer */
}

void TCD0_enableOutputChannels(void)
{
    /* enable write-protected register */
    CPU_CCP = CCP_IOREG_gc;

    TCD0.FAULTCTRL = TCD_CMPAEN_bm          /* enable channel A */
                   | TCD_CMPBEN_bm;         /* enable channel B */
}

void PORT_init(void)
{
    PORTA.DIR |= PIN4_bm                    /* set pin 4 as output */
              | PIN5_bm;                    /* set pin 5 as output */
}

int main(void)
{
    PORT_init();

    TCD0_enableOutputChannels();

    TCD0_init();

    /* Replace with your application code */
    while (1)
    {
        ;
    }
}
```

Example 7-2. Controlling Synchronous Signals Using Input Events Source Code

```
#define SETTLING_TIME_EXAMPLE_VALUE         (0x02)
#define DUTY_CYCLE_EXAMPLE_VALUE            (0xF6)

#include <avr/io.h>
/*Using default clock 3.3 MHz */

void TCD0_init(void);
void TCD0_enableOutputChannels(void);
void EVENT_SYSTEM_init(void);
void PORT_init(void);

void TCD0_init(void)
{
    /* set the waveform mode */
    TCD0.CTRLB = TCD_WGMODE_FOURRAMP_gc;

    /* set channel D to match channel B */
    TCD0.CTRLC = TCD_CMPDSEL_bm;

    /* set the settling time and duty cycle for the signals*/
    TCD0.CMPASET = SETTLING_TIME_EXAMPLE_VALUE;
    TCD0.CMPACLR = DUTY_CYCLE_EXAMPLE_VALUE;
    TCD0.CMPBSET = SETTLING_TIME_EXAMPLE_VALUE;
    TCD0.CMPBCLR = DUTY_CYCLE_EXAMPLE_VALUE;

    TCD0.EVCTRLA = TCD_CFG_FILTER_gc            /* set the anti-spike filter */
                 | TCD_EDGE_FALL_LOW_gc         /* set the 'fault' state */
                 | TCD_TRIGEI_bm;               /* enable input channel A */

    /* set the input mode */
    TCD0.INPUTCTRLA = TCD_INPUTMODE_WAIT_gc;

    /* ensure the ENRDY bit is set */
    while(!(TCD0.STATUS & TCD_ENRDY_bm))
    {
        ;
    }

    TCD0.CTRLA = TCD_CLKSEL_20MHZ_gc            /* choose the timer's clock */
               | TCD_CNTPRES_DIV4_gc            /* choose the prescaler */
               | TCD_ENABLE_bm;                 /* enable the timer */
}

void TCD0_enableOutputChannels(void)
{
    /* enable write-protected register */
    CPU_CCP = CCP_IOREG_gc;

    TCD0.FAULTCTRL = TCD_CMPAEN_bm              /* enable channel A */
                   | TCD_CMPBEN_bm              /* enable channel B */
                   | TCD_CMPCEN_bm              /* enable channel C */
                   | TCD_CMPDEN_bm;             /* enable channel D */
}

void EVENT_SYSTEM_init(void)
{
    EVSYS.ASYNCCH2 = EVSYS_ASYNCCH2_PORTC_PIN5_gc;

    EVSYS.ASYNCUSER6 = EVSYS_ASYNCUSER6_ASYNCCH2_gc;
}

void PORT_init(void)
{
    /* set pin 4 and pin 5 of port A as output */
    PORTA.DIR |= PIN4_bm
              | PIN5_bm;

    /* set pin 0 and pin 1 of port C as output */
    PORTC.DIR |= PIN0_bm
              | PIN1_bm;

    /* set pin 5 of port C as input */
    PORTC.DIR &= ~PIN5_bm;
```

```
    /* enable pull-up resistor for pin 5 of port C */
    PORTC.PIN5CTRL = PORT_PULLUPEN_bm;
}

int main(void)
{
    PORT_init();

    EVENT_SYSTEM_init();

    TCD0_enableOutputChannels();

    TCD0_init();

    /* Replace with your application code */
    while (1)
    {
        ;
    }
}
```

8. Revision History

Document Revision	Date	Comments
B	02/2021	Updated the GitHub repository links, the *References* section, and the use cases sections. Added the *AVR® DA Family Overview*, *Dithering Feature* and *Revision History* sections. Added MCC versions for each use case, running on AVR128DA48. Other minor corrections.
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.

Legal Notice

Trademarks

ISBN: 978-1-5224-7715-0

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