APPLICATION NOTE

Atmel AT01616: Using the WeX Timer/Counter Extension

Atmel AVR XMEGA E

Features

- Output matrix for timer/counter compare channels distribution
  - Configurable distribution of compare channel outputs across port pins
  - Redistribution of dead-time insertion resource between TC4 and TC5
- Four dead-time insertion (DTI) units
  - 8-bit resolution
  - Separated high and low side dead-time setting
  - Double buffered dead time
- Four swap (SWAP) units
  - Separated port pair or low/high side swap
  - Double buffered swap feature
- Pattern generator unit creating synchronized bit pattern across the port pins
  - Double buffered pattern generation
- Output disable

Introduction

This application note describes the various functions of the Waveform Extension (WeX) to Timer/Counter 4/5 available on the Atmel® XMEGA® E.

It details the differences and improvements according to the AWeX extension to Timer/Counters of the previous XMEGA (see AVR®1311 Application Note).

Included are code examples to simplify the use of WEX in typical applications.

All the software examples mentioned in this cookbook are provided in ASF (Atmel® Software Framework).

Table of Contents

1. Glossary

WeX	Waveform Extension
AWeX	Advanced Waveform Extension of previous XMEGA A, B, C, D
OTMX	Output Matrix
DTI	Dead-time insertion
ASF	Atmel Software Framework
Atmel Studio	Integrated Development Environment (IDE) for Atmel applications
SMPS	Switching Mode Power Supply

2. Pre-requisites

The solutions discussed in this document require basic familiarity with the following skills and technologies.

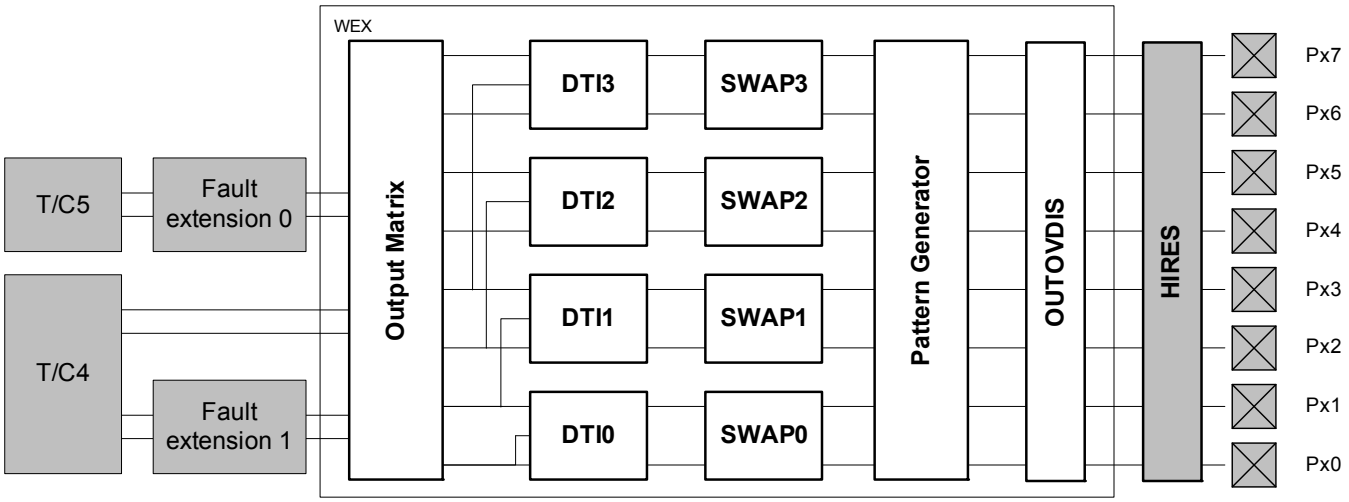
- XMEGA E Manual
- XMEGA E Datasheet
- AWeX AVR1311 Application Note
- Atmel Studio 5 or 6
- Atmel debugger AVR JTAGICE mkII or JTAGICE3
- Atmel STK®600 Starter Kit

3. WeX

3.1 WeX overview

The waveform extension (WEX) provides extra functions to the Timer/counter in waveform generation (WG) modes. It is primarily intended for use in different types of motor control, ballast, LED, H-bridge, power converter, and other types of power control applications. The WEX consists of five independent sub-functions, as shown in Figure 3-1. This overview example is the Atmel ATxmega32E5 configuration with two Timer/Counters (TC4 and TC5).

Figure 3-1. WeX module overview.



3.2 WeX versus AWeX

WeX provides the following improvements compared to standard AWeX module in ATxmega products:

New functions:

- Output Matrix
- Swap
- Output Disable

Improved functions:

- Pattern generator
- Dead-time insertion (DTI)
- Fault (see below)

Other changes:

- FAULT extension is a new standalone extension with improved features versus the Fault function included in standard AWeX description.

4. Output Matrix

New Output Matrix sub-function has been developed in WeX for Power control systems (such as SMPS) and Lighting applications.

The output matrix (OTMX) can distribute the waveform outputs of Timer/Counters across the port pins according to the configurations detailed in Table 4-1.

Table 4-1. Timer/counter 4 and 5 compare channel pin routing configuration.

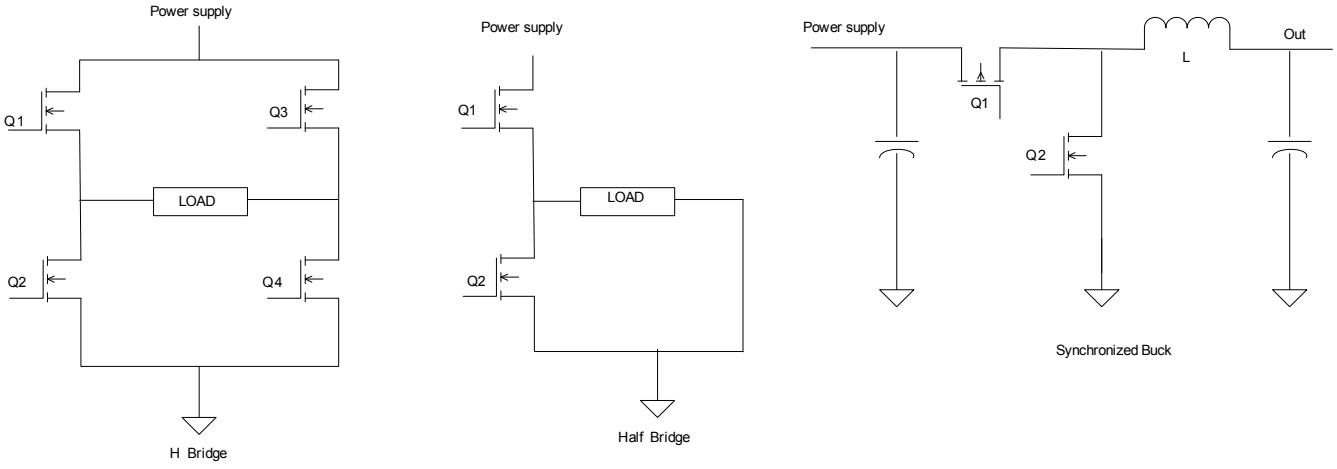
OTMX[2:0]	PIN 7	PIN 6	PIN 5	PIN 4	PIN 3	PIN 2	PIN 1	PIN 0	Main applications
000			TC5CCB	TC5CCA	TC4CCD	TC4CCC	TC4CCB	TC4CCA	Reset configuration
001	TC5CCB	TC5CCA	TC5CCB	TC5CCA	TC4CCD	TC4CCC	TC4CCB	TC4CCA	H bridges/SMPS + Motor control
010	TC5CCB	TC5CCA	TC4CCB	TC4CCA	TC5CCB	TC5CCA	TC4CCB	TC4CCA	H bridges/SMPS + Motor control
011	TC4CCA	TC4CCA	TC4CCA	TC4CCA	TC4CCA	TC4CCA	TC4CCA	TC4CCA	LED
100	TC4CCB	TC4CCA	TC4CCA	TC4CCA	TC4CCA	TC4CCA	TC4CCA	TC4CCA	LED

Caution: As the Output Matrix is the first sub-function in WeX, all the following sub-functions will be applied to the output signals of the Matrix (DTI, SWAP, Pattern generator and Port override).

Table 4-1 is only valid if DTI, SWAP and Pattern generator are in Reset configuration. Else, the TCxCCy outputs are transformed according to the sub-extensions configurations until they reach the Output pins.

The different Output Matrix modes can be used to adapt the WeX outputs to different Application topologies. Some different power control topologies are shown Figure 4-2.

Figure 4-2. Power control topologies examples.



4.2 H bridge example

Lighting applications use Half or Full H bridges.

DC Motor control applications also use different switching schemes to control H bridges.

Some switching schemes examples are listed in Table 4-2 and in Figure 4-3.

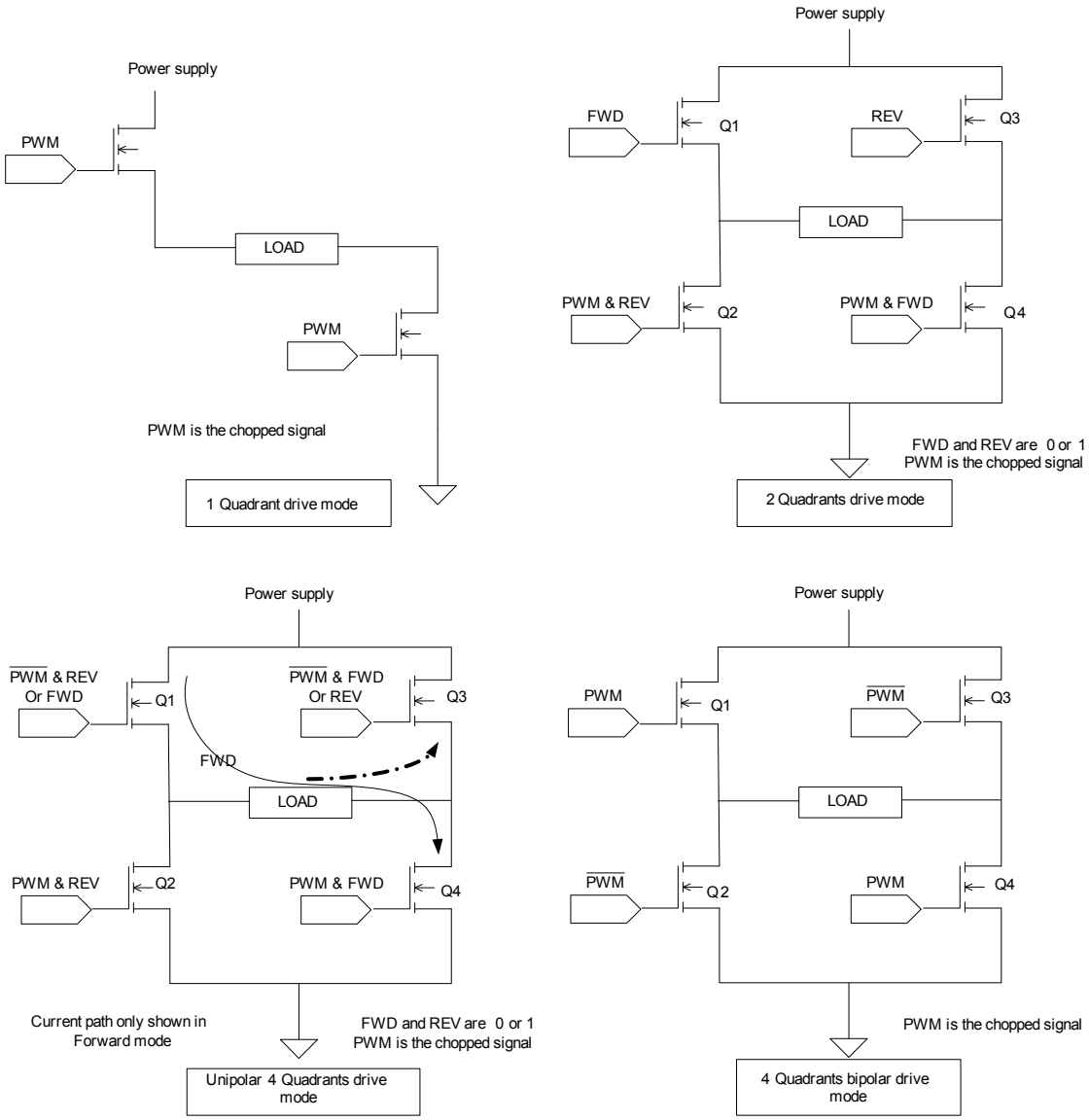
The different Output Matrix modes can be useful to adapt a generic control system to these different switching schemes.

Table 4-2. H bridge control schemes requirements.

Quadrants	Unipolar/Bipolar	PWM signals	DTI required	Pattern generator use	Timer mode
1	Unipolar	1 PWM	No		
2	Unipolar	1 PWM	No		
4	Unipolar	2 complimentary PWM	Yes	Yes	
4	Bipolar	2 complimentary PWM	Yes		Center-aligned mode

In the following figures, FWD and REV are exclusive logic levels 0/1.

Figure 4-3. H bridge control schemes figures.



All the six outputs of Timer4 and Timer5 are available and can be rerouted or overwritten with following configurations of Output Matrix.

4.3 Configuration 000

The 000 is the Reset default configuration.

4.4 Configuration 001

This configuration duplicates the waveform outputs from timer/counter 5 compare channel A and B (TC5CCA and TC5CCB) on two other pin locations.

OTMX[2:0]	PIN7	PIN 6	PIN 5	PIN 4	PIN 3	PIN 2	PIN 1	PIN 0	Main Applications
001	TC5CCB	TC5CCA	TC5CCB	TC5CCA	TC4CCD	TC4CCC	TC4CCB	TC4CCA	H bridges/SMPS + Motor control

So the TC5 (CCA and CCB) outputs are available at the same time on outputs respectively 4/6 and 5/7.

4.4.1 Applications

This mode can be used, for instance, to control the four transistors of a full H bridge.

4.5 Configuration 010

OTMX[2:0]	PIN7	PIN 6	PIN 5	PIN 4	PIN 3	PIN 2	PIN 1	PIN 0	Main Applications
010	TC5CCB	TC5CCA	TC4CCB	TC4CCA	TC5CCB	TC5CCA	TC4CCB	TC4CCA	H bridges/SMPS + Motor control

This mode distributes the waveform generator outputs from compare channels A and B (CCA and CCB) of both timer/counter 4 and 5 on two other pin locations.

4.5.1 Example of applications in this mode

This Matrix mode can be used:

- if user needs to get Compare channels outputs with DTI from Timer TC5. If DTI is set, the outputs will be:

OTMX[2:0]	PIN7	PIN 6	PIN 5	PIN 4	PIN 3	PIN 2	PIN 1	PIN 0	Main Applications
010	TC5CCB LS	TC5CCB HS	TC5CCA LS	TC5CCA HS	TC4CCB LS	TC4CCB HS	TC4CCA LS	TC4CCA HS	H bridges/SMPS + Motor control

- in applications which require dynamic braking. The mode provides redundant outputs with same PWM signal. This function saves external logic in customer application.
- in applications which require to drive 2 full H-bridges.

4.6 Configuration 011

This mode is equivalent to the Common Waveform mode.

It distributes the waveform output from timer/counter 4 compare channel A (TC4CCA) to all port pins.

4.6.1 Applications

This configuration can be used with Pattern generator to control a stepper motor.

For Lighting applications, this mode can be useful to control multiple LED strings.

4.7 Configuration 100

This configuration distributes the waveform output from TC5CCA to pin 7 and the waveform output from TC4CCA to all other pins (Px0 to Px6).

4.7.1 Applications

Application examples using this configuration could be:

- Control of 1 to 7 LEDs strings
- Control, at the same time, of the Led strings and the DC/DC converter which powers the LEDs. In this case, TC5 is used to control the Transistor of the DC/DC converter (Buck or Boost) while the LED current regulation loop controls the TC4

4.8 Registers

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
+0x00	CTRL		OTMX[2]	OTMX[1]	OTMX[0]				
Read/Write	R/W		R/W	R/W	R/W				
Initial value	0		0	0	0				

- Bit 6:4 – OTMX[2:0]: Output Matrix

  These bits define the matrix routing of the timers/counter waveform generation outputs to the port pins, according to Table 4-1.

4.9 Example 1 (Output Matrix control)

This project provides the way to configure the Output Matrix to the different modes.

4.9.1 Drivers

The drivers to configure the Output Matrix are included in ASF:

\src\asf\xmega\drivers\tcdrivers/tc.h

Parameters:

- *WEX is the Pointer to WEX module (WEXC)
- Otmx is the Output Matrix mode

4.9.1.1 Write function

```
static inline void tc45_WEX_set_otmx(WEX_t *WEX, enum wex_otmx_mode_t otmx)
    ((WEX_t *)WEX)->CTRL = (((WEX_t *)WEX)->CTRL & ~WEX_OTMX_gm) | otmx;
```

It configures WEX in the specified output matrix mode

Examples:

```
tc45_WEX_set_otmx(&WEXC, WEX_OTMX_DEFAULT);
tc45_WEX_set_otmx(&WEXC, WEX_OTMX_1);
tc45_WEX_set_otmx(&WEXC, WEX_OTMX_2);
tc45_WEX_set_otmx(&WEXC, WEX_OTMX_3);
tc45_WEX_set_otmx(&WEXC, WEX_OTMX_4);
```

4.9.1.2 Read function

```
static inline uint16_t tc45_WEX_read_otmx(WEX_t *WEX)
    return (((WEX_t *)WEX)->CTRL & WEX_OTMX_gm);
```

4.9.2 Example

Configuration:

TCC4 Timer:

Compare/Capture modules A/B: Compare and waveform generation in One slope mode

Outputs of Port C0/C1 C2/C3 are with Duty-cycle: Period/4 and Period/2

WexC: OTMX configuration can be changed

TCC5 Timer

Compare/Capture modules A/B: Compare and waveform generation in One slope mode

Outputs of Port C4/C5 are with Duty-cycle: Period/8 (472µs @66MHz and F/4) and Period/6 (616µs @66MHz and F/4)

WexC: OTMX configuration can be changed

OTMX:

To configure the OTMX, the function: tc45_WEX_set_otmx(..) must be modified

Results:

In OTMX 0 mode, the Outputs will be:

- PC0/PC1: TC4 CCA/CCB
- PC2/PC3: TC4 CCC/CCD
- PC4/PC5: TC5 CCA/CCB

In OTMX 1 mode, the Outputs will be:

- PC0/PC1: TC4 CCA/CCB
- PC2/PC3: TC4 CCC/CCD
- PC4/PC5: TC5 CCA/CCB
- PC6/PC7: TC5 CCA/CCB

In OTMX2 mode, the Outputs will be:

- PC0/PC1: TC4 CCA/CCB
- PC2/PC3: TC5 CCA/CCB
- PC4/PC5: TC4 CCA/CCB
- PC6/PC7: TC5 CCA/CCB

In OTMX3 mode, the Outputs will be:

- PC0 to PC7: TC4 CCA

In OTMX4 mode, the Outputs will be:

- PC0 to PC6: TC4 CCA
- PC7: TC4 CCB

5. Dead-Time Insertion

5.1 Overview

In a system driven by a pair of transistors operating in the Complementary Output mode it is completely forbidden to enable simultaneously the two FETs on the same side. This would lead to Shoot Through (a short circuit from power supply to ground).

Because the power output devices cannot switch instantaneously, some amount of time must be provided between the turn-off event of one PWM output in a complementary pair and the turn-on event of the other transistor.

The dead time function in the PWM control avoids the drivers of the same set of PWMs (PWMxH and PWMxL) from being on simultaneously due to the operating speed of the driver during output generation.

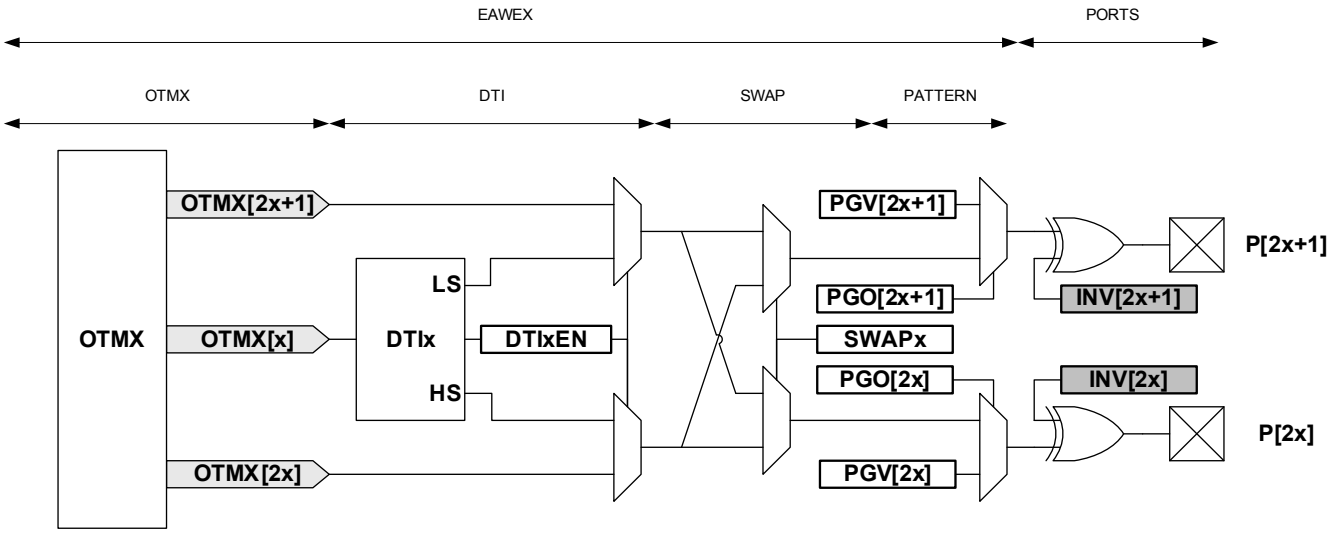
This requirement is also explained in AVR1311 Application Note which describes the DTI sub-function in AWeX of ATxmega.

Dead time must be inserted when any of the PWM I/O pin pairs are operating in the Complementary Output mode.

Four DTI insertion functions (DTI0 to DTI3) control the four lowest OTMX outputs (see Figure 3-1 WeX module overview.):

Figure 5-1 shows a diagram of one dead-time insertion until action on a port pin pair.

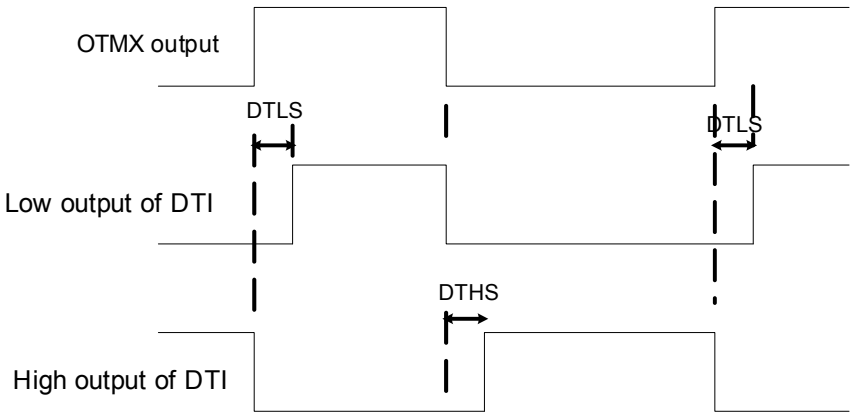
Figure 5-1. DTI overview.



The dead-time for high-side and low-side can be set individually through the DTHS and DTLS registers respectively (see Figure 5-2).

The dead-time DTHS and DTLS are common to all the OTMX outputs.

Figure 5-2. DTI action.



DTI0 transforms the OTMX0 output

- in two complementary PWM outputs
- with hardware dead-time:
- and outputs are Port pins 0 and 1

DTI1 transforms the OTMX1 output

- in two complementary PWM outputs
- with hardware dead-time:
- and outputs are Port pins 2 and 3

DTIn transforms the OTMXn output

- in two complementary PWM outputs
- with hardware dead-time:
- and outputs are Port pins 2x and 2x+1

As a shortcut, DTHS and DTLS can be set to the same value by writing to the DTBOTH register.

The dead-time value is given in main system clock cycles. The allowable range for the dead-time is thus 0-255 main system clock cycles.

The Dead Time length should not be too short otherwise the same driver pair may still be simultaneously activated. It should not be too long to ensure good efficiency. A proper dead-time length should be selected according to the speed of the selected driver.

This dead-time is usually 100ns to 2.5µs (real result is 1.5µs due to slopes of Gate voltages of MOSFET versus input capacitance). So the maximum dead-time can be configured with the 8-bit register.

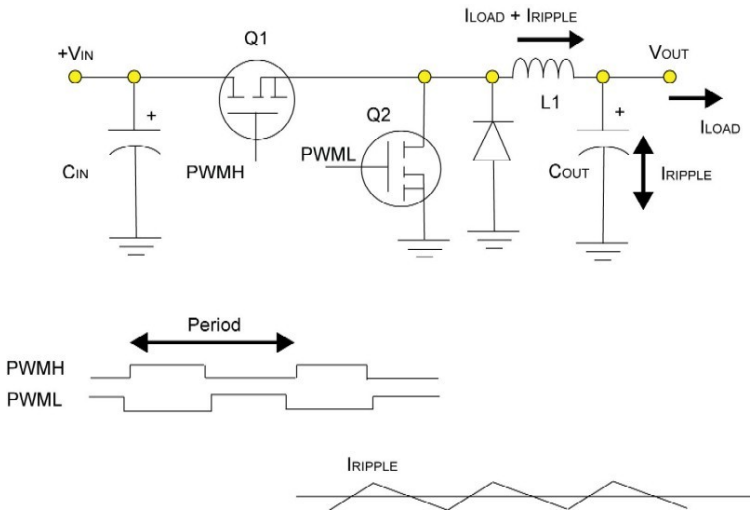
5.2 Applications view

Dead-time insertion is mandatory if a pair of transistors is operating in the Complementary Output mode.

This mode is used for instance in:

- Half H bridge
- H bridge in 2 and 4 Quadrant unipolar control
- Synchro buck converter (see Figure 5-3)
- Interleaved Boost converter

Figure 5-3. Synchro buck converter example.



5.3 WeX DTI improvement

The improvement of WeX DTI compared to AWeX DTI is that the DTI registers are no more shared with Pattern Generator sub-function.

So in WeX, Pattern Generator can be used with PWM signals which have built-in DTI.

Nevertheless, DTI registers are shared with blanking function in FAULT feature (see description in FAULT section of the XMEGA E Manual). So DTI in WeX and Blank in FAULT cannot be used at the same time.

5.4 Registers

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
+0x00	CTRL					DTI3EN	DTI2EN	DTI1EN	DTI0EN
Read/Write	R/W					R/W	R/W	R/W	R/W
Initial value	0					0	0	0	0

- Bit 3:0 – DTIxEN: Dead-Time Insertion Generator x Enable

  Setting any of these bits enables the dead-time insertion generator for the corresponding matrix output. This will override the related matrix outputs[2x] and [2x+1], with the low side and high side waveform respectively.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
+0x01	DTBOTH	DTBOTH[7]	DTBOTH[6]	DTBOTH[5]	DTBOTH[4]	DTBOTH[3]	DTBOTH[2]	DTBOTH[1]	DTBOTH[.0]
Read/Write	W	W	W	W	W	W	W	W	W
Initial value	0	0	0	0	0	0	0	0	0

- Bit 7:0 – DTBOTH[7:0]: Dead-time Both Sides

  Writing to this register will update the DTHS and DTLS registers at the same time (i.e., at the same I/O write access). Reading it, give 0x00 Value.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
+0x02	DTLS	DTLS[7]	DTLS[6]	DTLS[5]	DTLS[4]	DTLS[3]	DTLS[2]	DTLS[1]	DTLS[.0]
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0	0

- Bit 7:0 – DTLS[7:0]: Dead-time Low Side

  This register holds the number of peripheral clock cycles for the dead-time low side.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
+0x03	DTHS	DTHS[7]	DTHS[6]	DTHS[5]	DTHS[4]	DTHS[3]	DTHS[2]	DTHS[1]	DTHS[.0]
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0	0

- Bit 7:0 – DTHS[7:0]: Dead-time High Side

  This register holds the number of peripheral clock cycles for the dead-time high side.

5.5 Dead Time Insertion example

5.5.1 Drivers

Parameters

*WEX: Pointer to WEX module (WEXC)

Functions

- To enable Deadtime insertion on CCA

```
static inline void tc45_WEX_enable_cca_deadtime(WEX_t *WEX)
  ((WEX_t *)WEX)->CTRL |= WEX_DTI0EN_bm;
```

- To disable Deadtime insertion on CCA

```
static inline void tc45_WEX_disable_cca_deadtime(WEX_t *WEX)
  ((WEX_t *)WEX)->CTRL &= ~WEX_DTI0EN_bm;
```

- To enable Deadtime insertion on CCB

```
static inline void tc45_WEX_enable_ccb_deadtime(WEX_t *WEX)
  ((WEX_t *)WEX)->CTRL |= WEX_DTI1EN_bm;
```

- To disable Deadtime insertion on CCB

```
 static inline void tc45_WEX_disable_ccb_deadtime(WEX_t *WEX)
  ((WEX_t *)WEX)->CTRL &= ~WEX_DTI1EN_bm;
```

- To enable Deadtime insertion on ccC

```
static inline void tc45_WEX_enable_ccc_deadtime(WEX_t *WEX)
  ((WEX_t *)WEX)->CTRL |= WEX_DTI2EN_bm;
```

- To disable Deadtime insertion on ccD

```
static inline void tc45_WEX_disable_ccc_deadtime(WEX_t *WEX)
  ((WEX_t *)WEX)->CTRL &= ~WEX_DTI2EN_bm;
```

- To enable Deadtime insertion on ccD

```
static inline void tc45_WEX_enable_ccd_deadtime(WEX_t *WEX)
  ((WEX_t *)WEX)->CTRL |= WEX_DTI3EN_bm;
```

- To disable Deadtime insertion on ccD

```
 static inline void tc45_WEX_disable_ccd_deadtime(WEX_t *WEX)
  ((WEX_t *)WEX)->CTRL &= ~WEX_DTI3EN_bm;
```

- To configure the high side deadtime

parameter value: deadtime value

```
static inline void tc45_WEX_set_dti_high(WEX_t *WEX, int16_t value)
  ((WEX_t *)WEX)->DTHS = value;
```

- To configure the low side deadtime

```
static inline void tc45_WEX_set_dti_low(WEX_t *WEX, int16_t value)
  ((WEX_t *)WEX)->DTLS = value;
```

- To configure a symmetrical deadtime

```
static inline void tc45_WEX_set_dti_both(WEX_t *WEX, int16_t value)
  ((WEX_t *)WEX)->DTBOTH = value;
```

5.5.2 Example

Functions

The program enables the dead-time of CCx outputs with following functions:

```
tc45_WEX_enable_cca_deadtime(&WEXC);
tc45_WEX_enable_ccb_deadtime(&WEXC);
tc45_WEX_enable_ccc_deadtime(&WEXC);
tc45_WEX_enable_ccd_deadtime(&WEXC);
```

and configures the dead-time of CCx outputs with following functions:

```
tc45_WEX_set_dti_high(&WEXC, 0x40);
tc45_WEX_set_dti_low(&WEXC, 0x40);/* 0x40=64 so DT time= 64 * 1/(Fextern / 4) -> DT= 64 / 66 * 4 = 3.87µs */
```

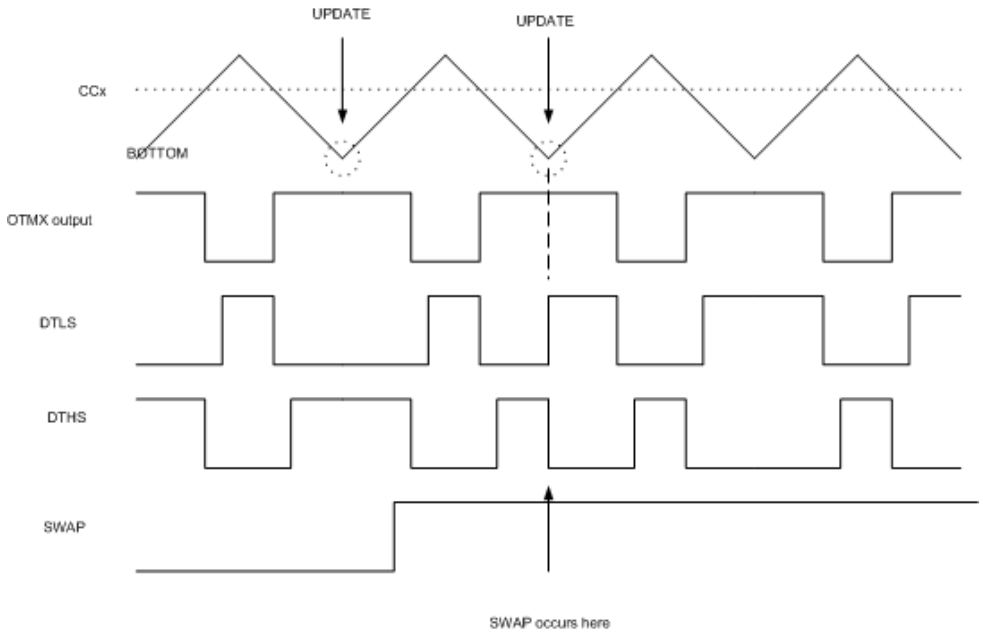
6. Swap

6.1 Overview

This new feature is useful to switch simultaneously two output signals when a Timer Update occurs.

Swap operation is achieved at UPDATE and is able to provide a very short timing constraint (some hundred ns) as shown in Figure 6-1.

Figure 6-1. Output swap.



As shown in Figure 3-1 and Figure 5-1, the DTI and SWAP units can be seen as a four port pair slices:

Slice 0 DTI0/ SWAP0 acting on port pins (Px[0],Px[1])

Slice 1 DTI1/ SWAP1 acting on port pins (Px[2],Px[3])

And more generally:

Slice n DTIn/ SWAPn acting on port pins (Px[2n],Px[2n+1])

6.2 Applications

The channel swap function is very useful in BLDC motor control. It allows the immediate change of top and bottom transistors in the phase. Using this function the rotor commutation and speed control can be splitted into two independent program parts. The state of the control signals can be changed immediately when required by the motor position (phase commutation) without changing the content of the PWM value registers. These changes can be accomplished asynchronously to the PWM duty cycle update.

Once the chopping current threshold is reached, the H-bridge can operate in two different current recirculation modes:

- an asynchronous mode if current recirculates through the diodes (in FETs or external). The user cannot control the occurrence of the alternate path creation
- a synchronous mode if enabling and disabling FETs in order to promote an alternate path

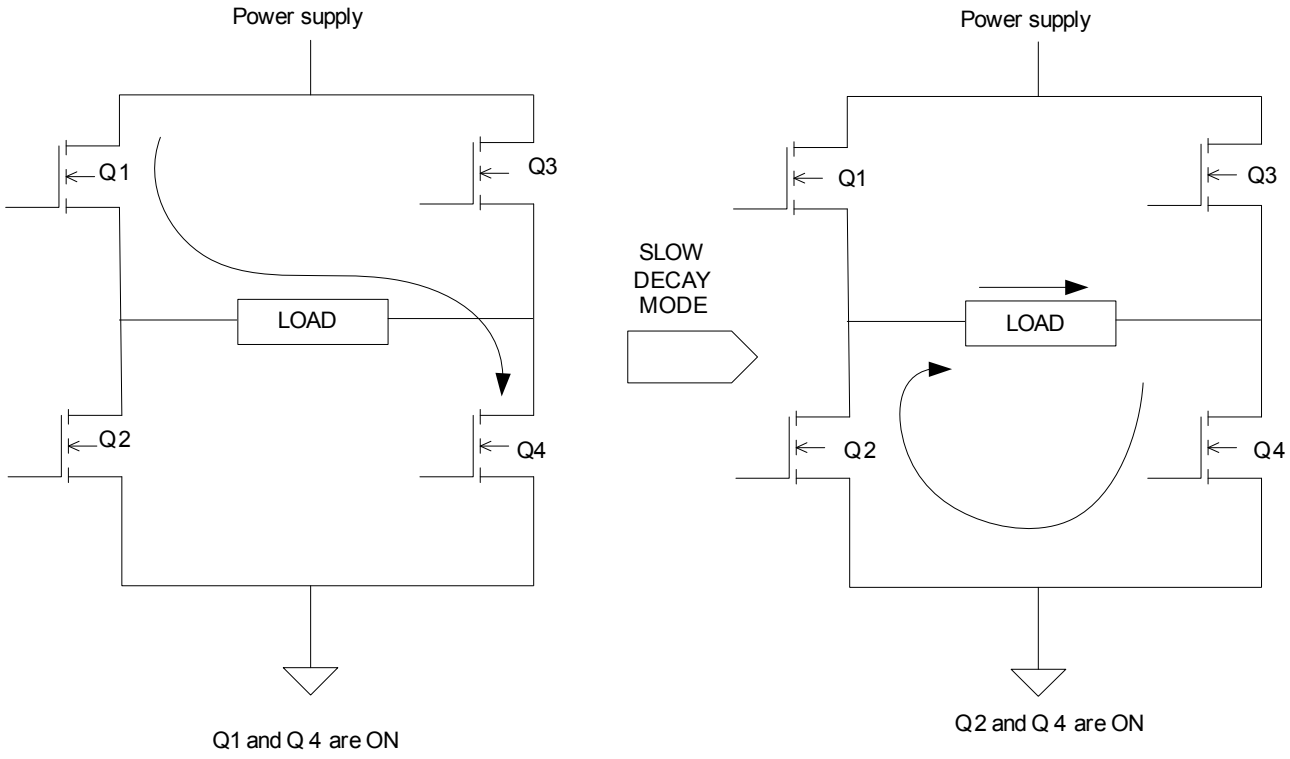
Two synchronous modes can be used: fast decay or slow decay. Fast and slow refer to the current decay mode and not the motor speed. It is the opposite for speed. In Fast decay mode, the motor will slow down in speed while in slow decay mode, the motor stops very quickly.

6.2.1 Slow decay mode

The Slow Decay mode uses the FETs on the same upper or lower H Bridge segment. More often the two low side FETs are used. The current inductor decreases to zero through the two FET's path. The decay time will depend on the RDSon of the FETs.

In slow decay mode the rotor stops very quickly.

Figure 6-2. Slow decay mode.



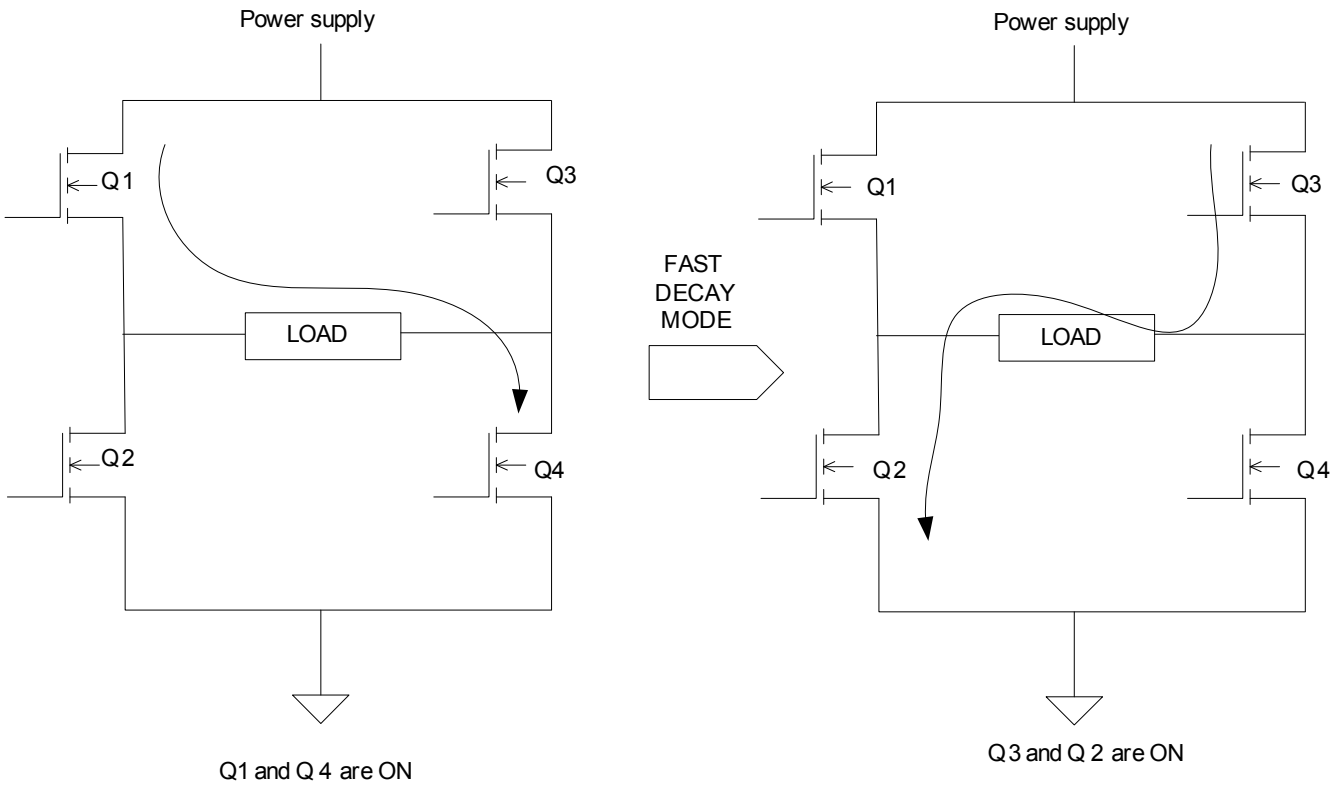
6.2.2 Fast decay mode

In fast decay mode, once the PWM chopping current level has been reached, the H-bridge reverses state to allow winding current to flow in a reverse direction. The opposite FETs are used as alternate path for the current to flow through which produces the fast decrease of the current in the H bridge. We are applying a voltage which fights, in an opposite way, the inductor current.

As the winding current approaches zero, the bridge is disabled to prevent any reverse current flow.

Fast decay mode is also named synchronous rectification and is mainly used in Stepper Motor applications (Microstepping…). Fast decay does not stress the internal diodes of FET or external diodes in parallel with FETs.

Figure 6-3. Fast decay mode.



6.2.3 Mixed decay mode

A third current decay mode is called Mixed Decay Mode. It is a mixture of Slow and Fast Decay modes. It is also mainly used in stepper motors control, especially microstepping.

6.3 SWAP WeX improvement

This mode was not available in previous AWeX.

The Swap can be achieved at Update of Timer4 or Timer5.

A Bit UPSEL (CTRL Register) allows selecting the UPDATE event from Timer4 or Timer5.

6.4 Registers

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
+0x06	SWAP	-	-	-	-	SWAP3	SWAP2	SWAP1	SWAP0
Read/Write	R/W	-	-	-	-	R/W	R/W	R/W	R/W
Initial value	0	-	-	-	-	0	0	0	0

- Bit 3:0 – SWAPx: Swap DTI output pair

  Setting these bits enables output swap of DTI outputs [2x] and [2x+1]. The outputs will be swapped independently of DTI enable bit (DTIxEN) setting.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
+0x0A	SWAPBUF	-	-	-	-	SWAP3BUF	SWAP2BUF	SWAP1BUF	SWAP0BUF
Read/Write	R/W	-	-	-	-	R/W	R/W	R/W	R/W
Initial value	0	-	-	-	-	0	0	0	0

- Bit 3:0 – SWAPxBUF: Swap DTI output pair

  These register bits are the buffer for the SWAP register bits. If double buffering is used, valid content in these bits are copied to the corresponding SWAPx bits on an UPDATE condition.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
+0x04/0x05	STATUSCLR/STATUSSET	-	-	-	-	-	SWAPBUFV	-	-
Read/Write	R/W	-	-	-	-	-	R/W	-	-
Initial value	0	-	-	-	-	-	0	-	-

- Bit 2 – SWAPBUFV: SWAP Buffer Valid

  If this bit is set, the swap buffer is written and contains valid data that will be copied into the SWAP register on the next UPDATE condition. If this bit is zero, no action will be taken. The connected timer/counter lock update (LUPD) flag also affects the update for the swap registers.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
+0x00	CTRL	7		-	-	-	-	-	-
Read/Write	R/W	UPSEL	-	-	-	-	-	-	-
Initial value	0	R	-	-	-	-	-	-	-

- Bit 7 – UPSEL: Update Source Selection

  By default the timer/counter 5 update condition is used by the swap and pattern generation units to also update their register content. Setting this bit, makes the timer/counter 5 update condition the source.

6.5 SWAP example

6.5.1 Drivers

Parameters:

- *WEX: Pointer to WEX module (WEXC)

Functions:

- Enable Swap on OTMX 0 and 1

```
static inline void tc45_WEX_enable_swap0(WEX_t *WEX)
  ((WEX_t *)WEX)->SWAP |= WEX_SWAP0_bm;
```

- Disable Swap on OTMX 0 and 1

```
static inline void tc45_WEX_disable_swap0(WEX_t *WEX)
  ((WEX_t *)WEX)->SWAP &= ~WEX_SWAP0_bm;
```

- Enable Swap Buffer on OTMX 0 and 1

```
static inline void tc45_WEX_enable_swap0_buffer(WEX_t *WEX)
  ((WEX_t *)WEX)->SWAPBUF |= WEX_SWAP0BUF_bm;
```

- Disable Swap on OTMX 0 and 1

```
static inline void tc45_WEX_disable_swap0_buffer(WEX_t *WEX)
  ((WEX_t *)WEX)->SWAPBUF &= ~WEX_SWAP0BUF_bm;
```

- Enable Swap on OTMX 2 and 3

```
static inline void tc45_WEX_enable_swap1(WEX_t *WEX)
  ((WEX_t *)WEX)->SWAP |= WEX_SWAP1_bm;
```

- Disable Swap on OTMX 2 and 3

```
static inline void tc45_WEX_disable_swap1(WEX_t *WEX)
  ((WEX_t *)WEX)->SWAP &= ~WEX_SWAP1_bm;
```

- Enable Swap buffer on OTMX 2 and 3

```
static inline void tc45_WEX_enable_swap1_buffer(WEX_t *WEX)
  ((WEX_t *)WEX)->SWAPBUF |= WEX_SWAP1BUF_bm;
```

- Disable Swap on OTMX 2 and 3

```
static inline void tc45_WEX_disable_swap1_buffer(WEX_t *WEX)
  ((WEX_t *)WEX)->SWAPBUF &= ~WEX_SWAP1BUF_bm;
```

- Enable Swap on OTMX 4 and 5

```
static inline void tc45_WEX_enable_swap2(WEX_t *WEX)
  ((WEX_t *)WEX)->SWAP |= WEX_SWAP2_bm;
```

- Disable Swap on OTMX 4 and 5

```
static inline void tc45_WEX_disable_swap2(WEX_t *WEX)
  ((WEX_t *)WEX)->SWAP &= ~WEX_SWAP2_bm;
```

- Enable Swap buffer on OTMX 4 and 5

```
static inline void tc45_WEX_enable_swap2_buffer(WEX_t *WEX)
  ((WEX_t *)WEX)->SWAPBUF |= WEX_SWAP2BUF_bm;
```

- Disable Swap on OTMX 4 and 5

```
static inline void tc45_WEX_disable_swap2_buffer(WEX_t *WEX)
  ((WEX_t *)WEX)->SWAPBUF &= ~WEX_SWAP2BUF_bm;
```

- Enable Swap on OTMX 6 and 7

```
static inline void tc45_WEX_enable_swap3(WEX_t *WEX)
  ((WEX_t *)WEX)->SWAP |= WEX_SWAP3_bm;
```

- Disable Swap on OTMX 6 and 7

```
static inline void tc45_WEX_disable_swap3(WEX_t *WEX)
  ((WEX_t *)WEX)->SWAP &= ~WEX_SWAP3_bm;
```

- Enable Swap buffer on OTMX 6 and 7

```
static inline void tc45_WEX_enable_swap3_buffer(WEX_t *WEX)
  ((WEX_t *)WEX)->SWAPBUF |= WEX_SWAP3BUF_bm;
```

- Disable Swap on OTMX 6 and 7

```
static inline void tc45_WEX_disable_swap3_buffer(WEX_t *WEX)
  ((WEX_t *)WEX)->SWAPBUF &= ~WEX_SWAP3BUF_bm;
```

6.5.2 Example

Configuration:

The CCx Interrupts are enabled with following functions,

```
/* Declares the interrupt functions which will be called when CCA and CCB
interrupts will occur */
tc45_set_cca_interrupt_callback(&TIMER_EXAMPLE_C,example_cca_interrupt_callback);
tc45_set_ccb_interrupt_callback(&TIMER_EXAMPLE_C,example_ccb_interrupt_callback);

/* Configures the interrupt level of CCA CCB CCC and CCD modules of Timer4: low */
tc45_set_cca_interrupt_level(&TIMER_EXAMPLE_C, TC45_INT_LVL_LO);
tc45_set_ccb_interrupt_level(&TIMER_EXAMPLE_C, TC45_INT_LVL_LO);
tc45_set_ccc_interrupt_level(&TIMER_EXAMPLE_C, TC45_INT_LVL_LO);
tc45_set_ccd_interrupt_level(&TIMER_EXAMPLE_C, TC45_INT_LVL_LO);
```

OTMX mode 2 is used:

Outputs in "unswapped mode" are:
PC7...0 = TC5 CCB/ TC5 CCA/ TC4 CCB / TC4CCA / TC5 CCB/ TC5 CCA/ TC4 CCB / TC4CCA.

The following instruction will enable C0/C1 to be swapped and C2/C3 to be swapped as well:

```
tc45_set_cca_interrupt_callback(&TIMER_EXAMPLE_C, example_cca_interrupt_callback);
```

The following instruction will enable C4/C5 to be swapped and C6/C7 to be swapped as well:

```
tc45_set_ccb_interrupt_callback(&TIMER_EXAMPLE_C, example_ccb_interrupt_callback);
```

Functions:

- CCx Interrupt example

The Compare/Capture detection interrupt callback function is called when an interrupt occurs on a Compare A channel (TIMER_C).

It increments the CC detection index and thus forces the pattern generator output one of two times. Example of CCA interrupt function is following (CCB interrupt uses the same instructions sequence):

```
static void example_cca_interrupt_callback(void)
{
    cca_pwm_index += 1;
    if (cca_pwm_index == 2)
        {
        cca_pwm_index = 0;
        tc45_WEX_enable_swap2(&WEXC);/* enable swap C4 and C5*/
        tc45_WEX_disable_swap3(&WEXC); /* disable swap C6 and C7*/
        }
        else
        {
            tc45_WEX_disable_swap2(&WEXC);/* disable swap C4 and C5*/
            tc45_WEX_enable_swap3(&WEXC); /* enable swap C6 and C7*/
        }
}
}
```

7. Pattern Generator

The pattern generation unit is used to generate synchronized output waveforms with constant logic level.

Using pattern generation, some of the eight outputs can be overwritten by a constant level.

As with other double buffered timer/counter registers, the register update is synchronized to the UPDATE condition set by the timer/counter waveform generation mode. If the application does not need synchronization, the application code can simply access the PGO, PGV or PORTx registers directly.

7.1 Applications

Pattern generator can be used, for the control of:

- Stepper motors
- Power H bridges: H bridge can be controlled in a flexible way in all quadrant configurations, see Table 4-2 H bridge control schemes requirements.

7.2 Pattern Generator WeX improvements

7.2.1 Pattern Generator buffers

The output enable buffer (PGOBUF) and the Output value buffer (PGVBUF) of the new Patter Generator are now specific compared to previous DTBUFLS and DTBUFHS of AWeX which were shared with DTI. This way Pattern generator can be used also with DTI signals.

7.2.2 Registers update

The update of Pattern Generator Registers (PGO and PGV) with content of PGOBUF and PGVBUF Buffers can be achieved at Timer update of Timer4 or Timer5.

This update was only possible at UPDATE of Timer0 in previous AWeX.

A new Bit UPSEL (CTRL register) allows selecting the UPDATE event from Timer4 or Timer5.

7.3 Registers

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
+0x07	PGO	PGO[7]	PGO[6]	PGO[5]	PGO[4]	PGO[3]	PGO[2]	PGO[1]	PGO[.0]
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0	0

- Bit 7:0 – PGO[7:0]: Pattern Generation Override

  This register holds the enables of pattern generation for each output. A bit position at one overrides the corresponding SWAP output with the related PGV bit value.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
+0x08	PGV	PGV[7]	PGV[6]	PGV[5]	PGV[4]	PGV[3]	PGV[2]	PGV[1]	PGV[.0]
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0	0

- Bit 7:0 – PGV[7:0]: Pattern Generation Value

  This register holds the values of pattern for each output.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
+0x0B	PGOBUF	PGOBUF[7]	PGOBUF[6]	PGOBUF[5]	PGOBUF[4]	PGOBUF[3]	PGOBUF[2]	PGOBUF[1]	PGOBUF[.0]
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0	0

- PGOBUF – Pattern Bit 7:0 – PGOBUF[7:0]: Pattern Generation Override Buffer

  This register is the buffer for the PGO register. If double buffering is used, valid content in this register is copied to the PGO register on an UPDATE condition.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
+0x0C	PGVBUF	PGVBUF[7]	PGVBUF[6]	PGVBUF[5]	PGVBUF[4]	PGVBUF[3]	PGVBUF[2]	PGVBUF[1]	PGVBUF[.0]
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0	0

- PGVBUF – Pattern Bit 7:0 – PGVBUF[7:0]: Pattern Generation Value Buffer

  This register is the buffer for the PGV register. If double buffering is used, valid content in this register is copied to the PGV register on an UPDATE condition.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
+0x04/0x05	STATUSCLR/STATUSSET	-	-	-	-	-	-	PGVBUFV	PGOBUFV
Read/Write	R/W	-	-	-	-	-	-	R/W	R/W
Initial value	0	-	-	-	-	-	-	0	0

- Bit 1 – PGVBUFV: Pattern Generator Value Buffer Valid

  If this bit is set, the pattern generation value (PGV) buffer is written and contains valid data that will be copied into the PGV register on the next UPDATE condition. If this bit is zero, no action will be taken. The connected timer/counter lock update (LUPD) flag also affects the update of the PGV buffer.

- Bit 0 – PGOBUFV: Pattern Generator Overwrite Buffer Valid

  If this bit is set, the pattern generation overwrite (PGO) buffer is written and contains valid data that will be copied into the PGO register on the next UPDATE condition. If this bit is zero, no action will be taken. The connected timer/counter lock update (LUPD) flag also affects the update for the PGO buffers.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
+0x00	CTRL	7	-	-	-	-	-	-	-
Read/Write	R/W	UPSEL	-	-	-	-	-	-	-
Initial value	0	R	-	-	-	-	-	-	-

- Bit 7 – UPSEL: Update Source Selection

  By default the timer/counter 5 update condition is used by the swap and pattern generation units to also update their register content. Setting this bit, makes the timer/counter 5 update condition the source.

7.4 Pattern Generator example

7.4.1 Drivers

Parameters:

* WEX: Pointer to WEX module (WEXC)

Param:

- PGO enable

- PGV value

Functions:

Configures Pattern Generator output enable

```
static inline void tc45_WEX_write_pgo(WEX_t *WEX, int16_t value)
    ((WEX_t *)WEX)->PGO = (((WEX_t *)WEX)->PGO & 0x00) |
        value;
```

Configures Pattern Generator Buffer output enable

```
static inline void tc45_WEX_write_pgo_buffer(WEX_t *WEX, int16_t value)
 ((WEX_t *)WEX)->PGOBUF =value;
```

Reads the Pattern generator Output buffer

return Patter Generator Buffer PGOBUF

```
static inline uint16_t tc45_WEX_read_pgo_buffer(volatile void *WEX)
        return (((WEX_t *)WEX)->PGOBUF);
```

Configures Pattern Generator Value

Parameter: PGV value

```
static inline void tc45_WEX_write_pgv(WEX_t *WEX, int16_t value)
    ((WEX_t *)WEX)->PGV = (((WEX_t *)WEX)->PGV & 0x00) |
        value;
```

Configures Pattern Generator Value Buffer

Parameter: PGVBUF value

```
static inline void tc45_WEX_write_pgv_buffer(WEX_t *WEX, int16_t value)
    ((WEX_t *)WEX)->PGVBUF = (((WEX_t *)WEX)->PGVBUF & 0x00) |
        value;
```

7.4.2 Example

Configuration:

Configures the Pattern Generator values

```
tc45_WEX_write_pgv(&WEXC, 0xF0); /* writes a "1" pattern value
tc45_WEX_write_pgv_buffer(&WEXC, 0xF0);
```

Configures the interrupt level of CCA CCB CCC and CCD modules of Timer4: low

```
tc45_set_cca_interrupt_level(&TIMER_EXAMPLE_C, TC45_INT_LVL_LO);
tc45_set_ccb_interrupt_level(&TIMER_EXAMPLE_C, TC45_INT_LVL_LO);
tc45_set_ccc_interrupt_level(&TIMER_EXAMPLE_C, TC45_INT_LVL_LO);
tc45_set_ccd_interrupt_level(&TIMER_EXAMPLE_C, TC45_INT_LVL_LO);
```

Declares the interrupt functions which will be called when CCA and CCB interrupts will occur

```
tc45_set_cca_interrupt_callback(&TIMER_EXAMPLE_C,example_cca_interrupt_callback);
tc45_set_ccb_interrupt_callback(&TIMER_EXAMPLE_C,example_ccb_interrupt_callback);
tc45_set_ccc_interrupt_callback(&TIMER_EXAMPLE_C,example_ccc_interrupt_callback);
tc45_set_ccd_interrupt_callback(&TIMER_EXAMPLE_C,example_ccd_interrupt_callback);
```

Functions:

Compare/Capture detection interrupt callback functions are called when an interrupt occurs on a Compare channel (TIMER_C).

It increments the CC detection index and thus forces the pattern generator output 1 of 2 times.

Example for CCA Channel:

```
static void example_cca_interrupt_callback(void)
{
    cca_pwm_index += 1;
    if (cca_pwm_index == 2)
        {
        cca_pwm_index = 0;
        value = tc45_WEX_read_pgo_buffer(&WEXC);
        value_tmx = tc45_WEX_read_otmx(&WEXC);
        if (value_tmx == 0x20) value |= 0x11; /* if Output matrix is 010, duplicate bits 0 and 4 (TC4 CCA) */
        else value |= 0x01;
        tc45_WEX_write_pgo_buffer(&WEXC, value);/* enables the Pattern Generator 1 time of 2 */
        }
        else
        {
        value = tc45_WEX_read_pgo_buffer(&WEXC);
        value_tmx = tc45_WEX_read_otmx(&WEXC);
        if (value_tmx == 0x20) value &= 0xEE;
        else
        value &= 0xFE;
        tc45_WEX_write_pgo_buffer(&WEXC, value);        /* disables the Pattern Generator the other time */
        }

}
```

8. Output Override Disable

The output override disable unit can be used to disable the waveform output on selectable port pins.

This function provides the benefit to optimize the pins usage. Selected pins can be let free for other functional use when the application does not need the waveform output spread across all the port pins.

This port override logic is common for all the timer/counter extensions.

The WeX extension controls the Port pins as soon as all the following configurations are present:

- Compare channel of Timer/Counter is enable for waveform generation (COMP or BOTHCC)
- Compare channel is swapped or duplicated through the sub-functions of WeX
  - AVR1330: Using the WeX Timer/Counter extension

8.1 Registers

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
+0x0F	OUTOVDIS	OUTOVDIS[7]	OUTOVDIS[6]	OUTOVDIS[5]	OUTOVDIS[4]	OUTOVDIS[3]	OUTOVDIS[2]	OUTOVDIS[1]	OUTOVDIS[.0]
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0	0

- Bit 7:0 – OUTOVDIS[7:0]: Output Override Disable

  These bits disable override of the corresponding port output register (i.e., one-to-one bit relation to pin position).

8.2 Output override disable example

8.2.1 Drivers

Parameters:

* WEX: Pointer to WEX module (WEXC)

param: Output value

Functions:

Configures Output override

```
 static inline void tc45_WEX_set_output_override(WEX_t *WEX, int16_t value)
      ((WEX_t *)WEX)->OUTOVDIS = (((WEX_t *)WEX)->OUTOVDIS & 0x00) |
           value;
```

8.2.2 Example

```
      /* Configures the Output Disable */
      tc45_WEX_set_output_override(&WEXC, 0x0F); /* as a 1 disable: only highest 4 outputs are enable */
//    tc45_WEX_set_output_override(&WEXC, 0xF0); /* as a 1 disable: only lowest 4 outputs are enable */
```

9. Revision History

Doc. Rev.	Date	Comments
42086A	04/2013	Initial document release

Atmel | Enabling Unlimited Possibilities®

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan G.K.
16F Shin-Osaki Kangyo Building
1-6-4 Osaki, Shinagawa-ku
Tokyo 141-0032
JAPAN
Tel: (+81)(3) 6417-0300
Fax: (+81)(3) 6417-0370