LECTURE 6 : DIGITAL TO ANALOG CONVERSION

REALTIDSSYSTEMER OG PROGRAMMERINGSSPROG

Sergiu Spataru
ssp@et.aau.dk

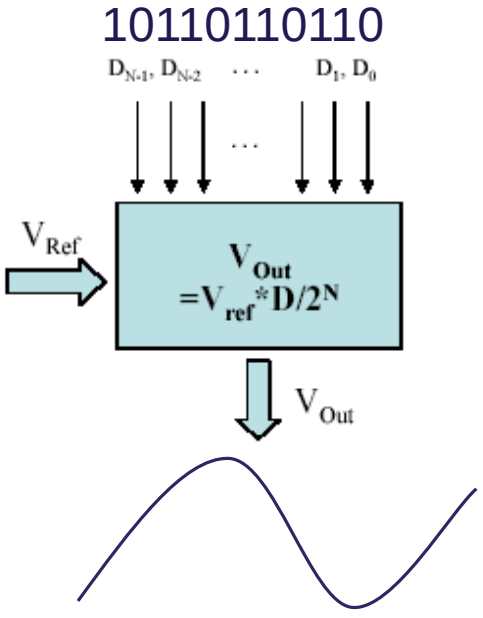
DEPARTMENT OF ENERGY TECHNOLOGY
AALBORG UNIVERSITY

Agenda

- **What is a DAC?**
- Main parameters of a DAC
- Pulse-Width Modulation (PWM)
- PWM module of F2806x

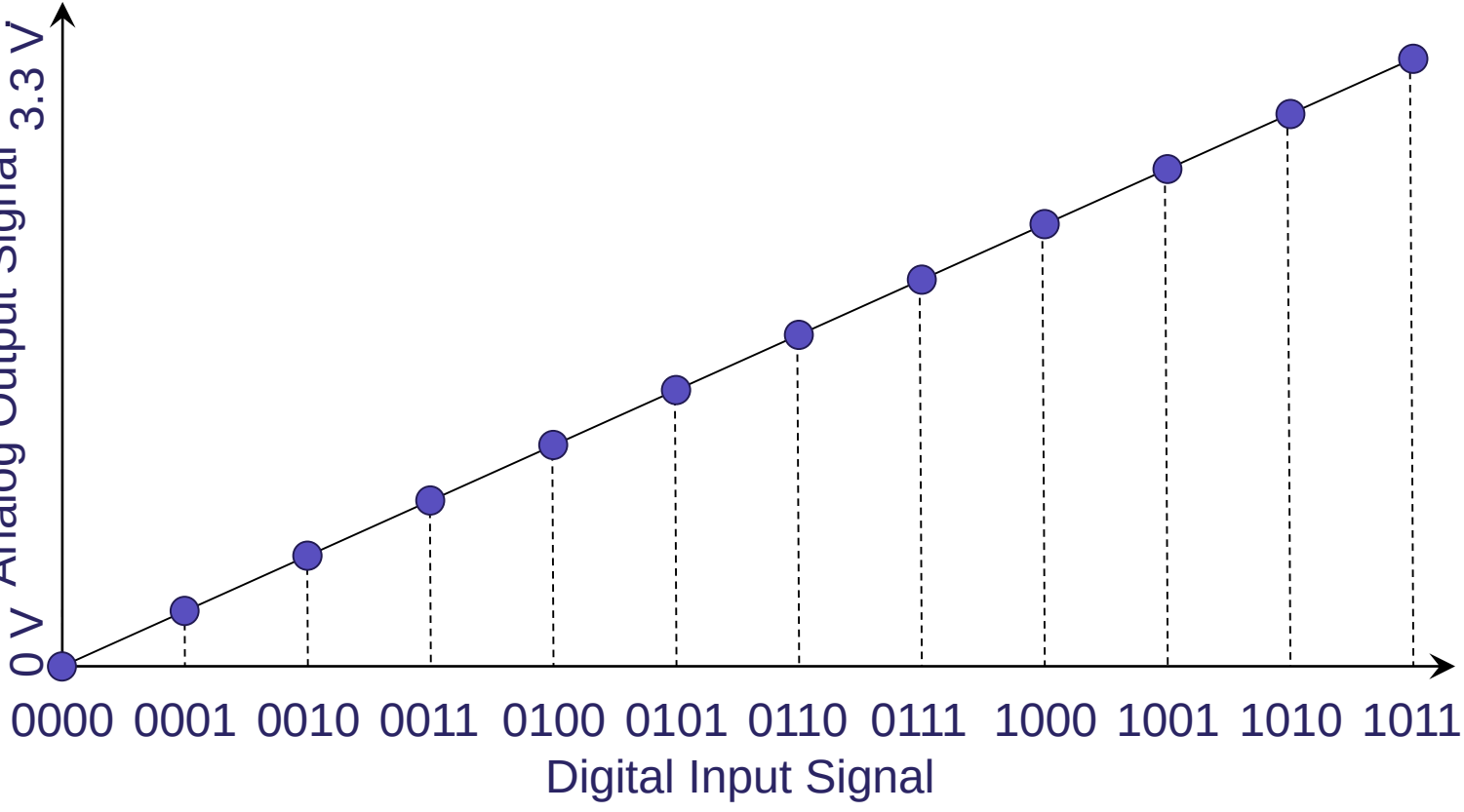
What is a Digital to Analog Converter?

- A digital to analog converter (DAC) is a device that converts digital numbers (binary) into an analog voltage or current output.



What is a Digital to Analog Converter?

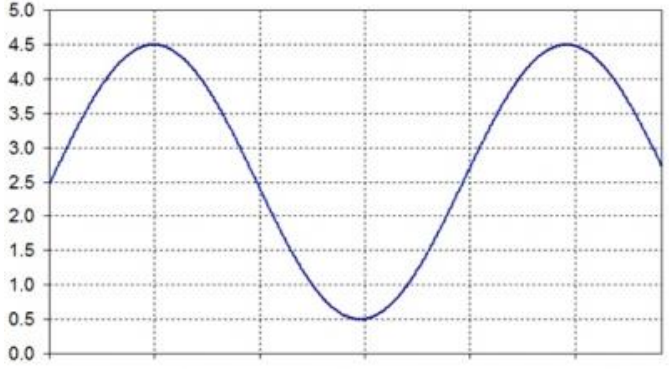
- Each binary number sampled by the DAC corresponds to a different output level.



Typical DAC output

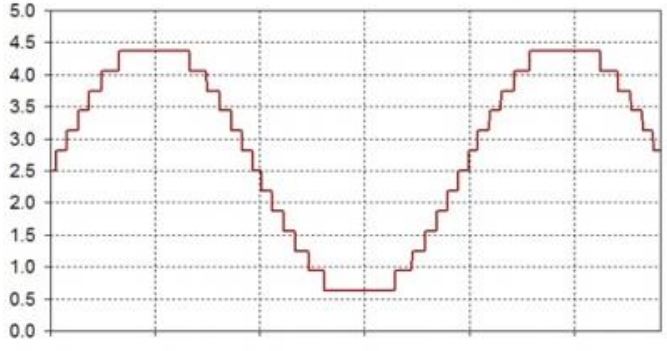
- DACs **convert** a digital number to a physical signal, and **hold** that value for a given sample interval.
- This is known as a zero-order hold and results in a piecewise constant output.

Ideal/Reference signal



DAC

Resulted DAC signal



Agenda

- What is a DAC?
- **Main parameters of a DAC**
- Pulse-Width Modulation (PWM)
- PWM module of F2806x

Main parameters of a DAC

- There are three main parameters that should be considered when choosing a DAC for a particular project:
  - Reference voltage
  - Resolution
  - Speed

DAC reference voltage

- **Vref** sets maximum DAC output voltage (if not amplified)

- Can be set externally or generated inside DAC

- **Multiplier DAC** – external **Vref**
  - **Vref** is not fixed
  - DAC is more flexible
  - Output opamp circuit must be properly designed

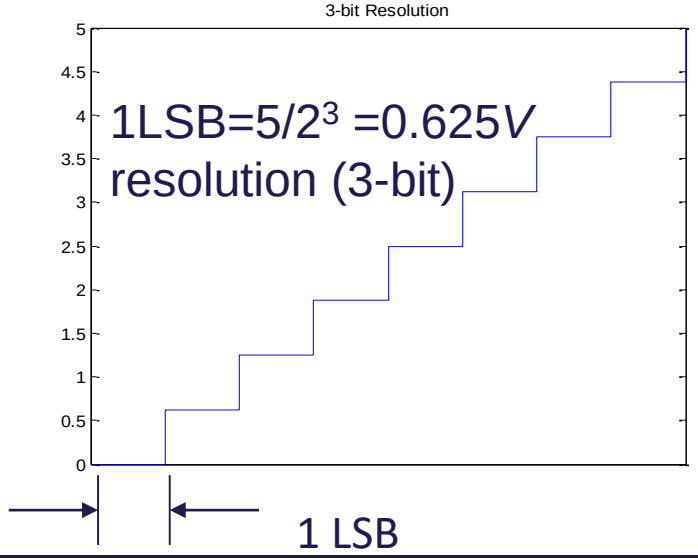
- **Non-Multiplier DAC** – internal **Vref**
  - **Vref** is constant and is set by the manufacturer.
  - Qualified for specified temperature range

DAC Resolution

- 1 LSB (digital)=1 step size for DAC output (analog)

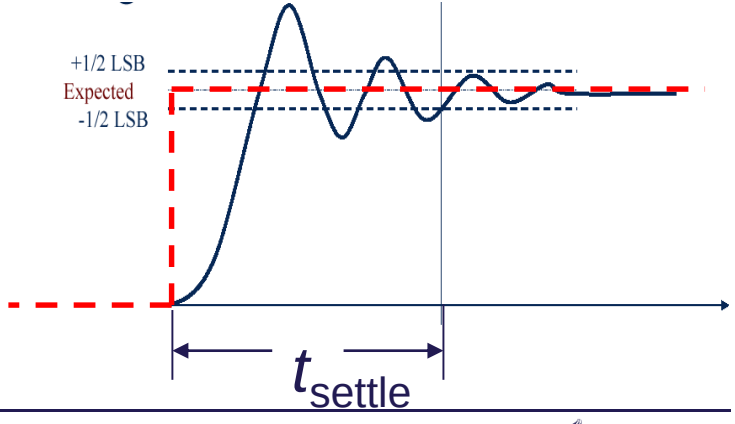
$$\text{Resolution} = \frac{V_{ref}}{2^n}$$

- Increasing the number of bits results in a finer resolution
- Most DACs - 8 to 16-bits (256 to 65,536 steps)



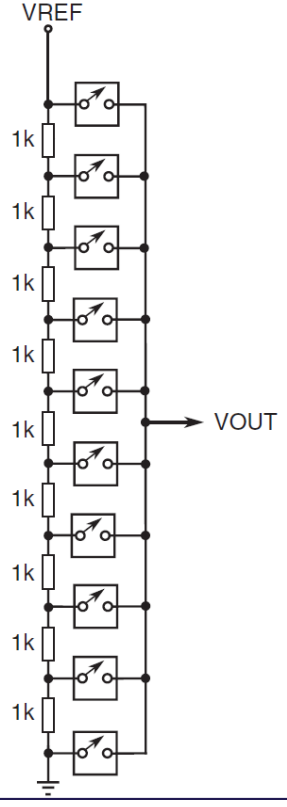
DAC speed and settling time

- **DAC speed** - The maximum rate at which DAC is reproducing usable analog output from digital input register
- **Speed** is **limited** by the clock speed of the microcontroller (input clock speed) and the **settling time** of the DAC
- **Settling time** -The interval between a command to update (change) its output value and the instant it reaches its final value, within a specified percentage (± ½ LSB)
- High speed DACs are defined as operating at greater than 1 millisecond per sample (1MHz).

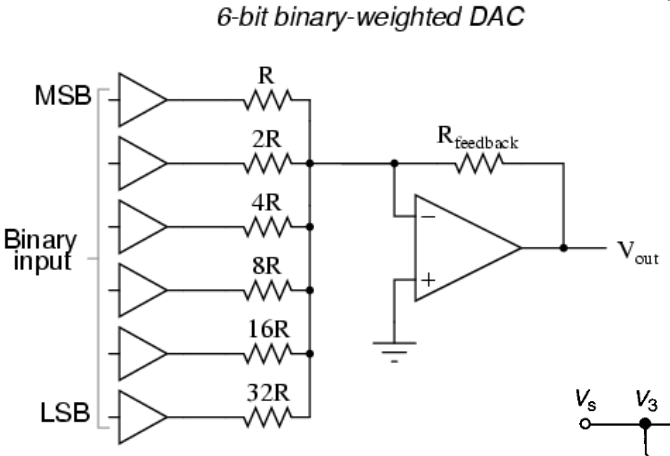


Typical resistor based DAC circuits

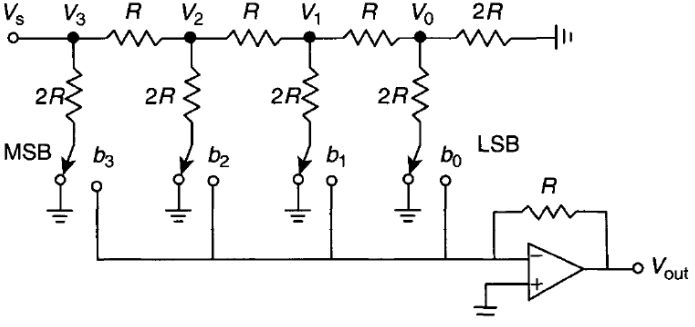
Digital potentiometer



Binary weighted resistors



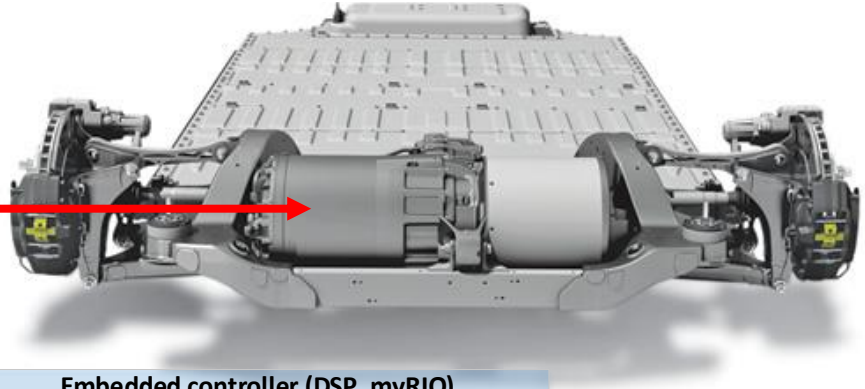
R2R Ladder

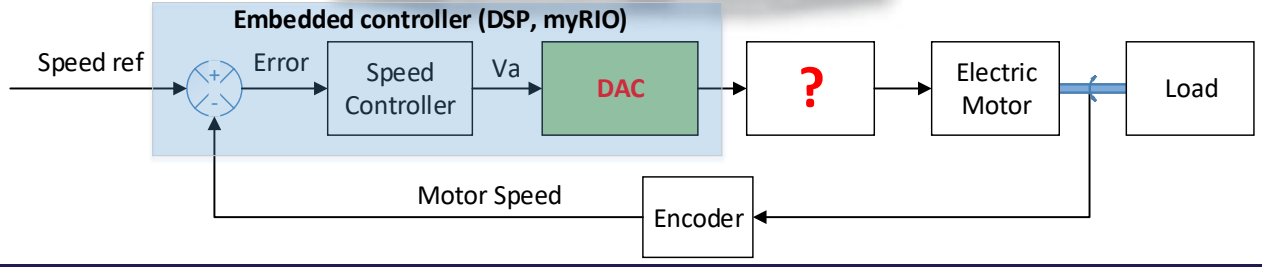


DACs in High Power Applications – How do we amplify the DAC signal?

Tesla Model S

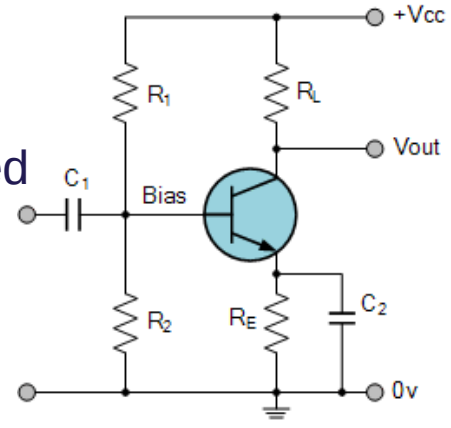
310 kW Electric Motor

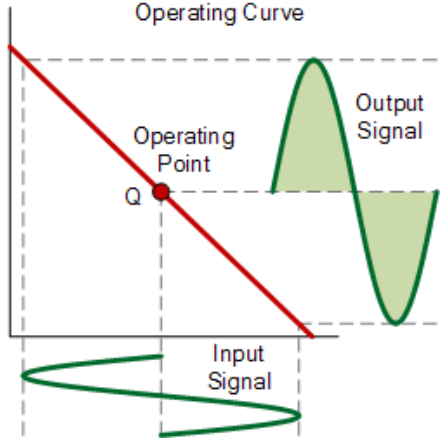




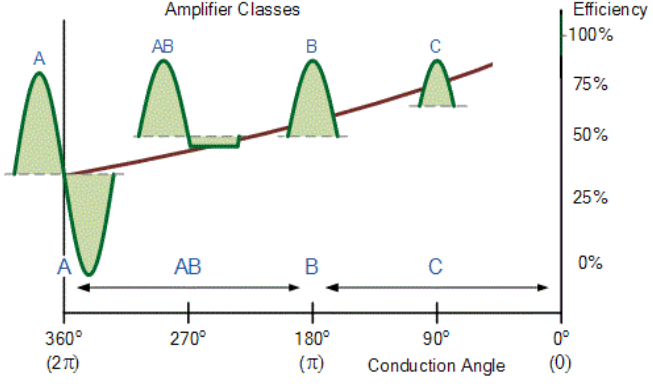
Are linear amplifiers a solution?

Transistor based linear power amplifier





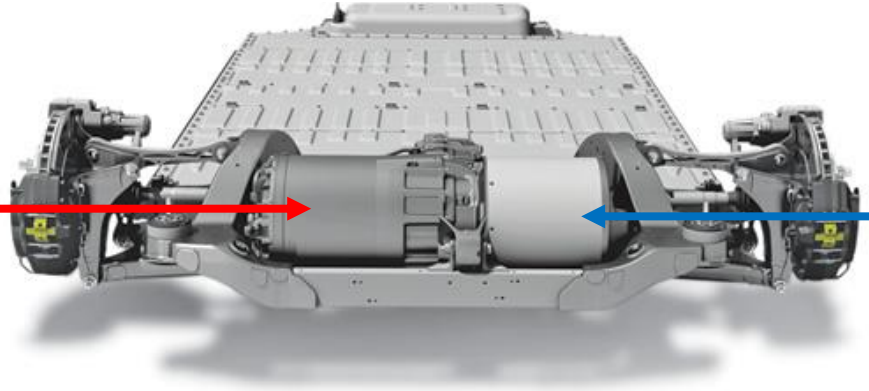
Linear amplifiers have low efficiencies



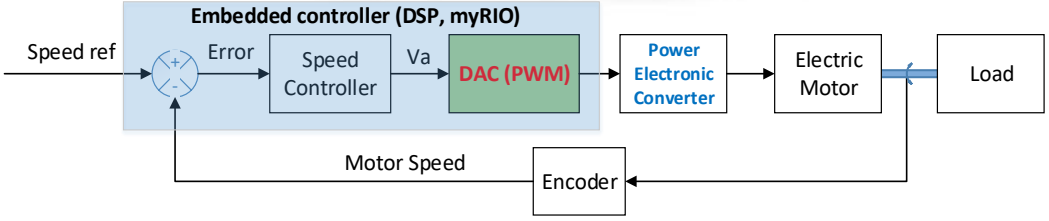
DACs in High Power Applications – PWM + Power Electronics

Tesla Model S

310 kW Electric Motor



Power Electronic Converter

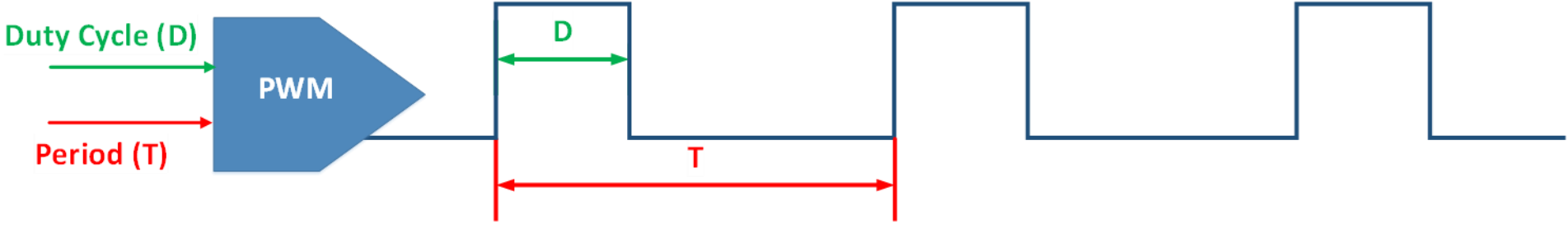


Agenda

- What is a DAC?
- Main parameters of a DAC
- **Pulse-Width Modulation (PWM)**
- PWM module of F2806x

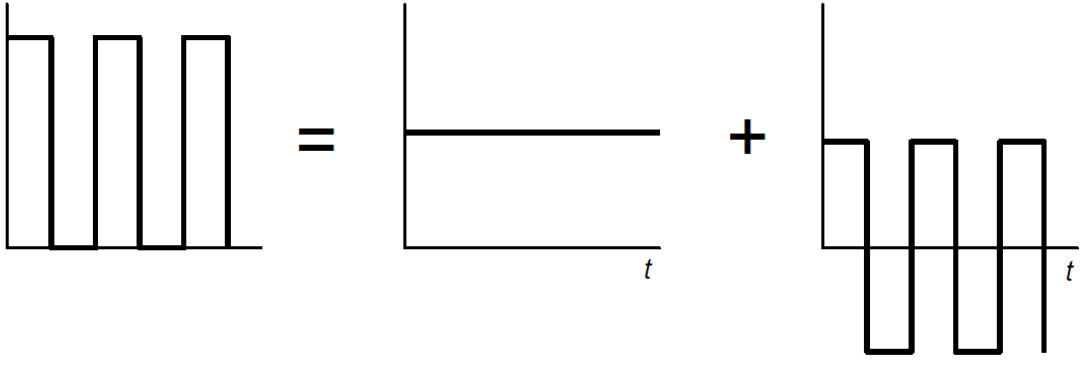
What is Pulse With Modulation (PWM)

- PWM is a technique used to **encode** a **information** into a **pulsing signal**
- A PWM signal consists of two main components that define its behavior: a **duty cycle** and a **period** (frequency)
- Typically we use the **duty cycle** to encode the information we want to transmit.



PWM as a DAC

- A PWM signal is the composed of **DC component** (A_0) and an infinite number of harmonics

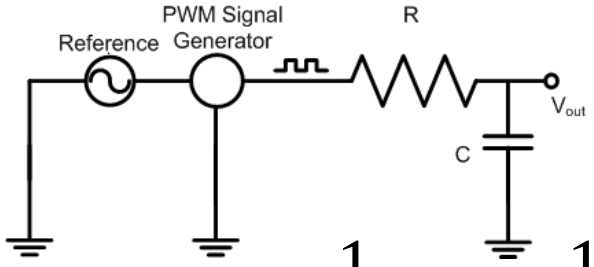


Original PWM signal **D.C. Component** **Zero average square wave**

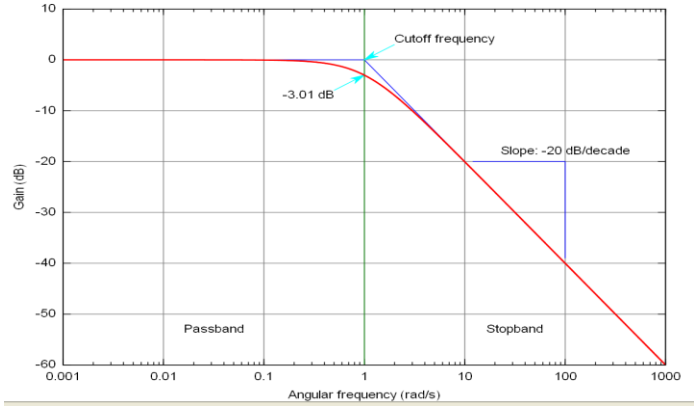
$$f(t) = A_0 + \sum_{n=1}^{\infty}\left[A_n \cos\left(\frac{2n\pi t}{T}\right) + B_n \sin\left(\frac{2n\pi t}{T}\right)\right]$$

$$A_0 = \frac{1}{2T}\int_{-T}^{T} f(t)dt$$

PWM as a DAC (by filtering)



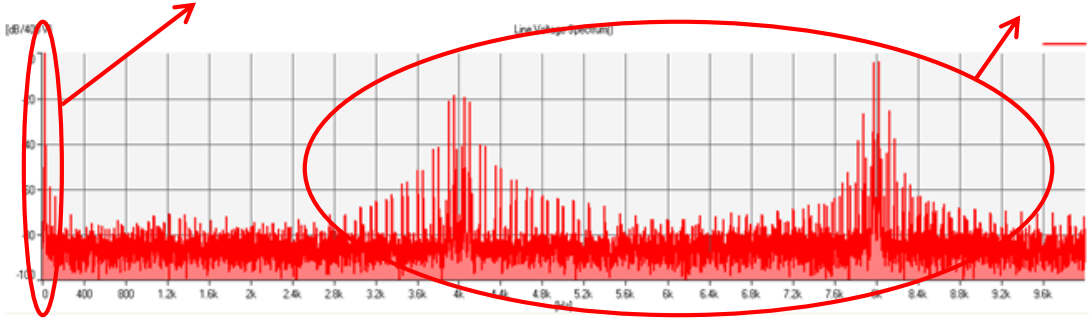
$$f_c = \frac{1}{2\pi\tau} = \frac{1}{2\pi RC}$$



Voltage spectrum

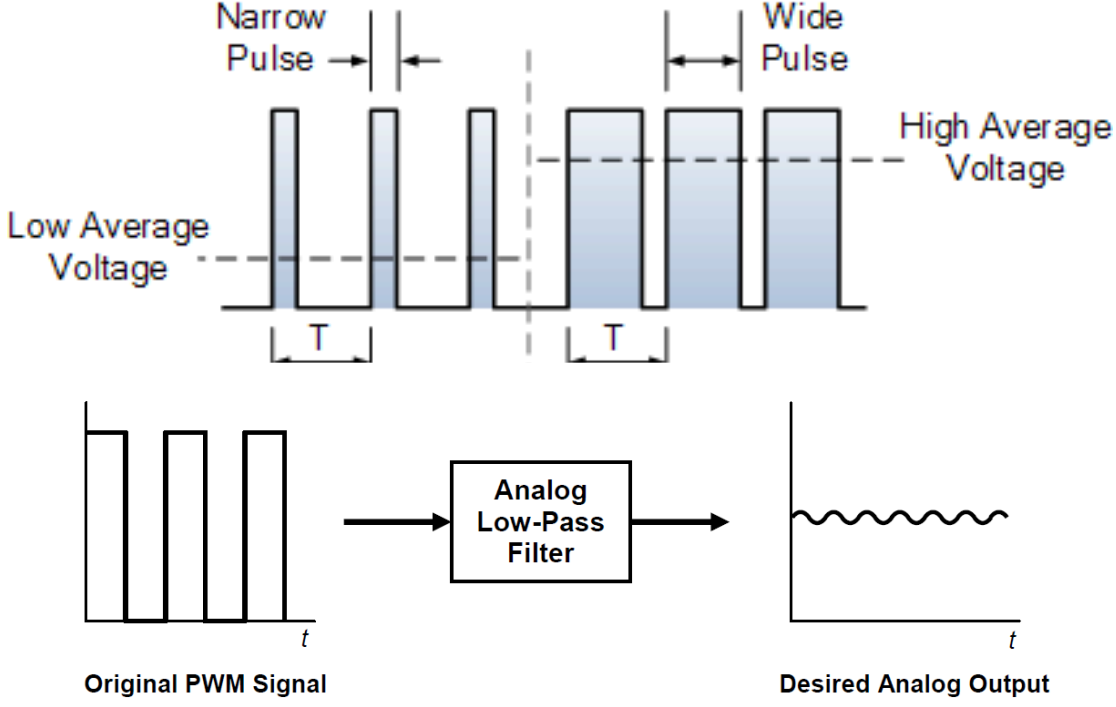
Fundamental

Upper Harmonics



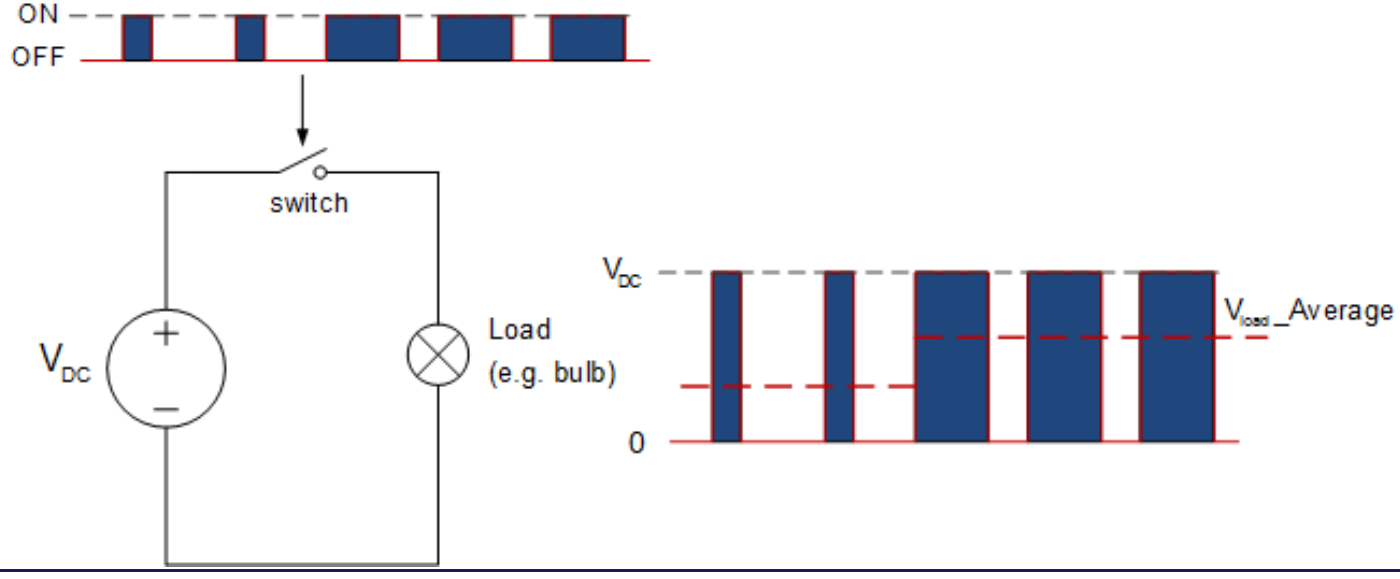
PWM as a DAC

- We can use the **duty cycle** of the PWM to control the **DC component** of the PWM signal
- And we can **remove** the **higher order harmonics** by **low pass filtering**



PWM as a DAC in practice

- Approximation of an analogue signal by switching in an on/off manner a DC voltage
- The average of the output voltage in a period is proportional to the reference voltage
- Often the process to control acts as low pass filter

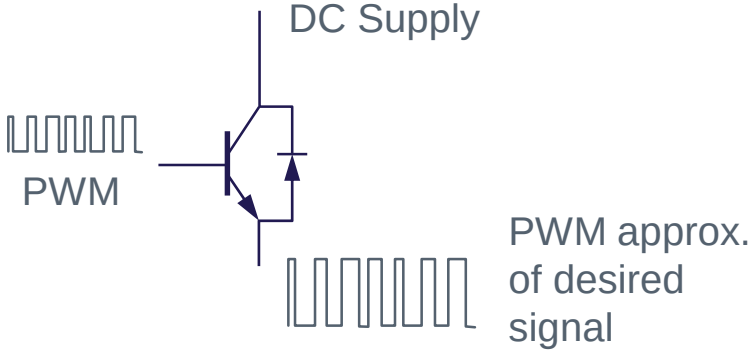


PWM Lecture Exercise

- See Exercise 1 in Lab6.pdf

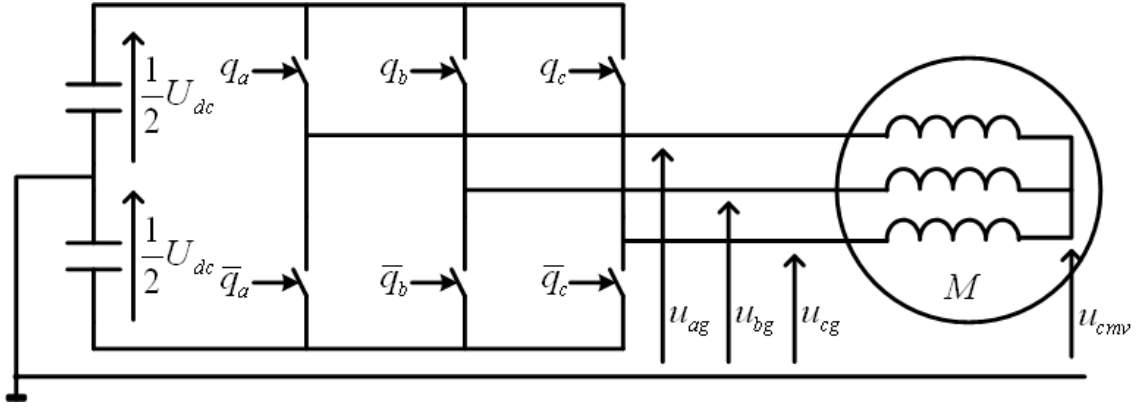
Application of PWM with Power Switching Devices

- Power switching devices are **Transistors**
  - Difficult to control in proportional region
  - Easy to control in saturated region
- Power converters (amplifiers) built with **power switching devices** are **called power electronic converters**
- PWM is a digital signal -> easy for MCU/DSP to output

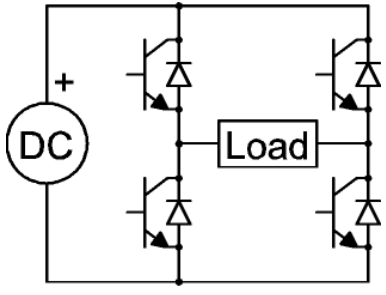


Typical Applications for PWM + PE Converters

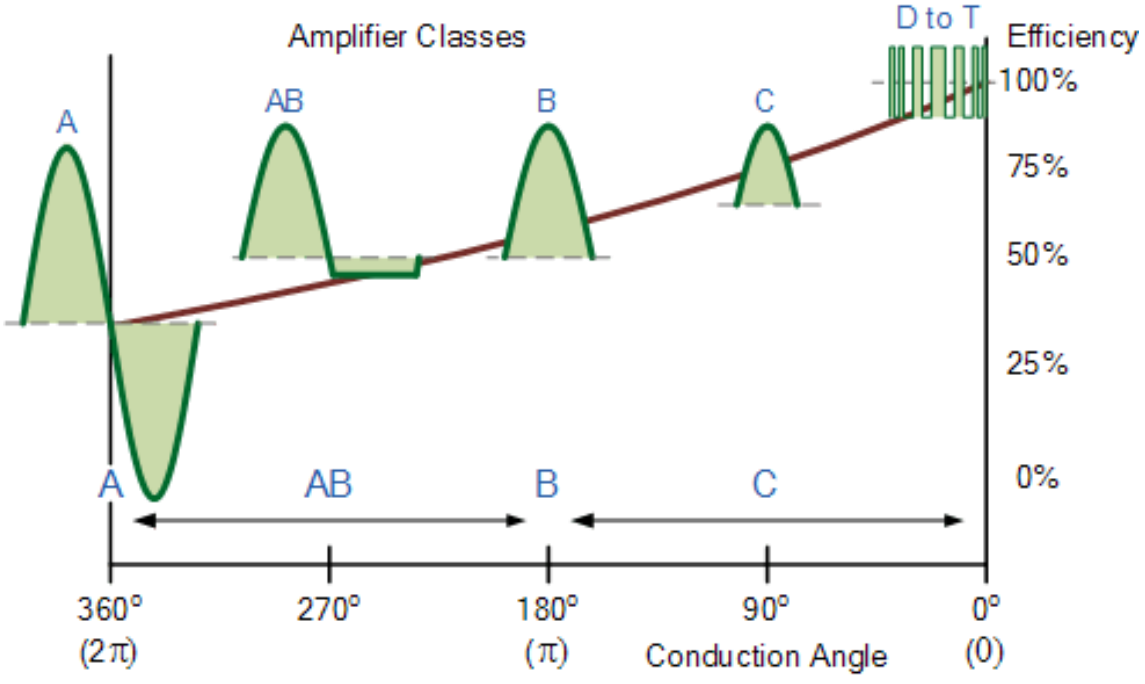
- AC motor control



- DC motor control

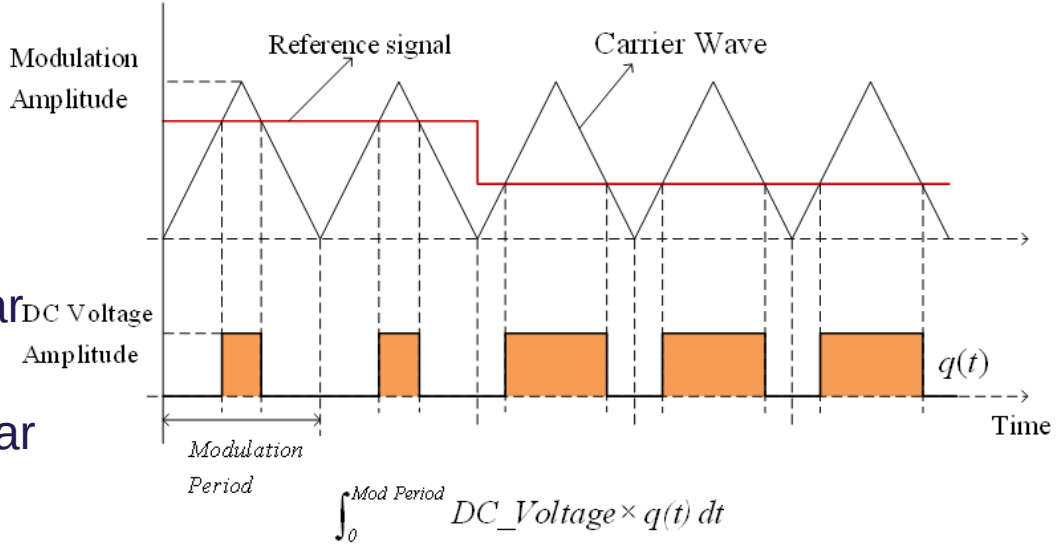


Linear Power Amplifiers vs. PWM + PE Converters



How is PWM generated?

- **Carrier wave**
  - **Type:** triangular or sawtooth
  - **Frequency:** – Will set the PWM frequency
- **Reference signal –** used to set the duty cycle
- **Comparator**
  - Reference signal >Triangular carrier wave – Pulse OFF
  - Reference signal < Triangular carrier wave – Pulse ON



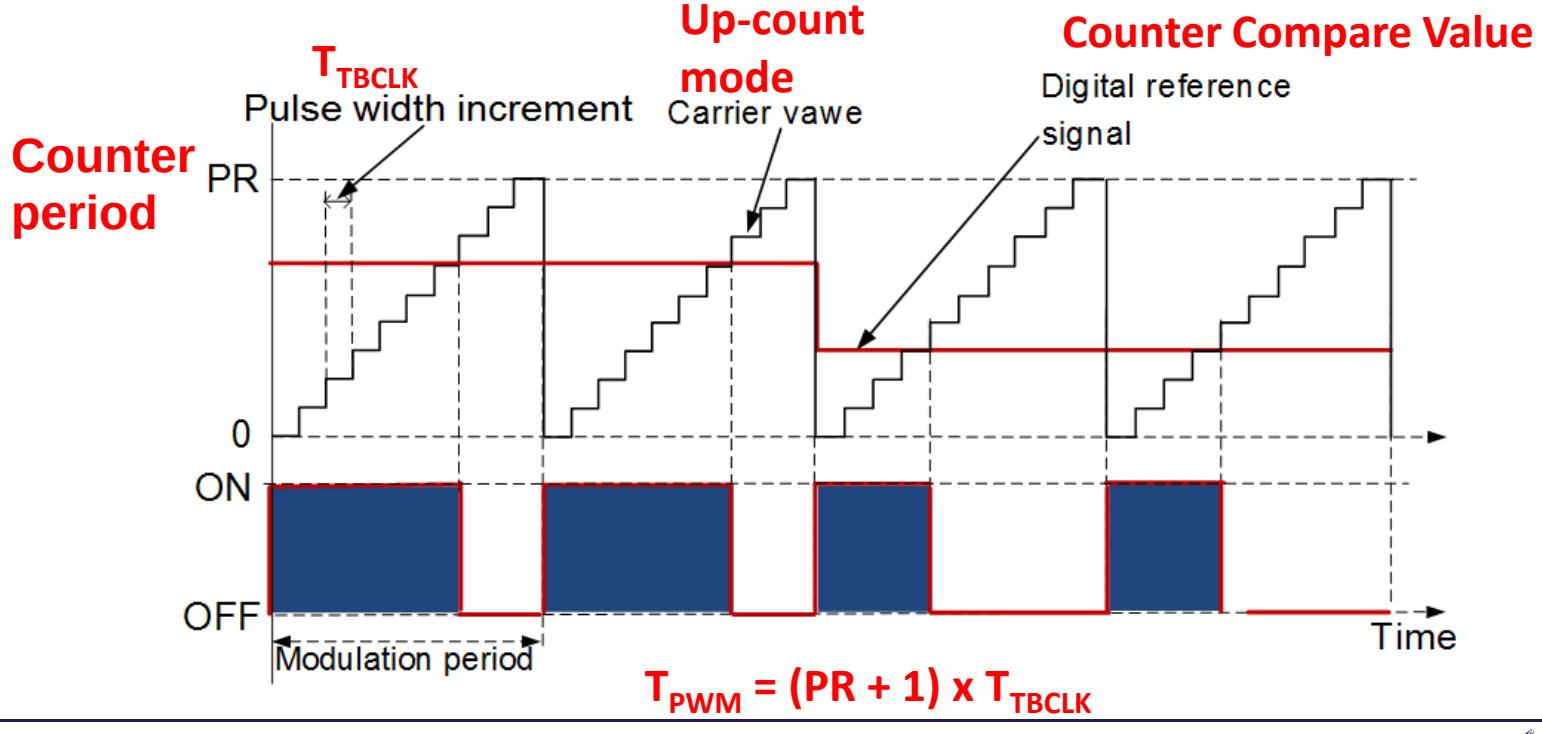
How is PWM generated digitally?

- The **carrier wave** is generated by a **timer/counter**
  - Has a **timer clock** (TBCLK) with the **time base** (T_{TBCLK})
  - Has a **counter period** (PR) which determines the **PWM period** (T_{PWM})
  - Has a **counting mode** which determined the carrier wave type
    - **Up-count mode** or **down-count mode** – **sawtooth carrier**
    - **Up-down-count mode** – **triangular carrier**

  - Has one or more **digital comparators -** called **Counter-Compare** units
  - The **reference signal** used to set the **duty cycle** is called the **Counter Compare Value** (CCV) and is set in a **Counter Compare Register**

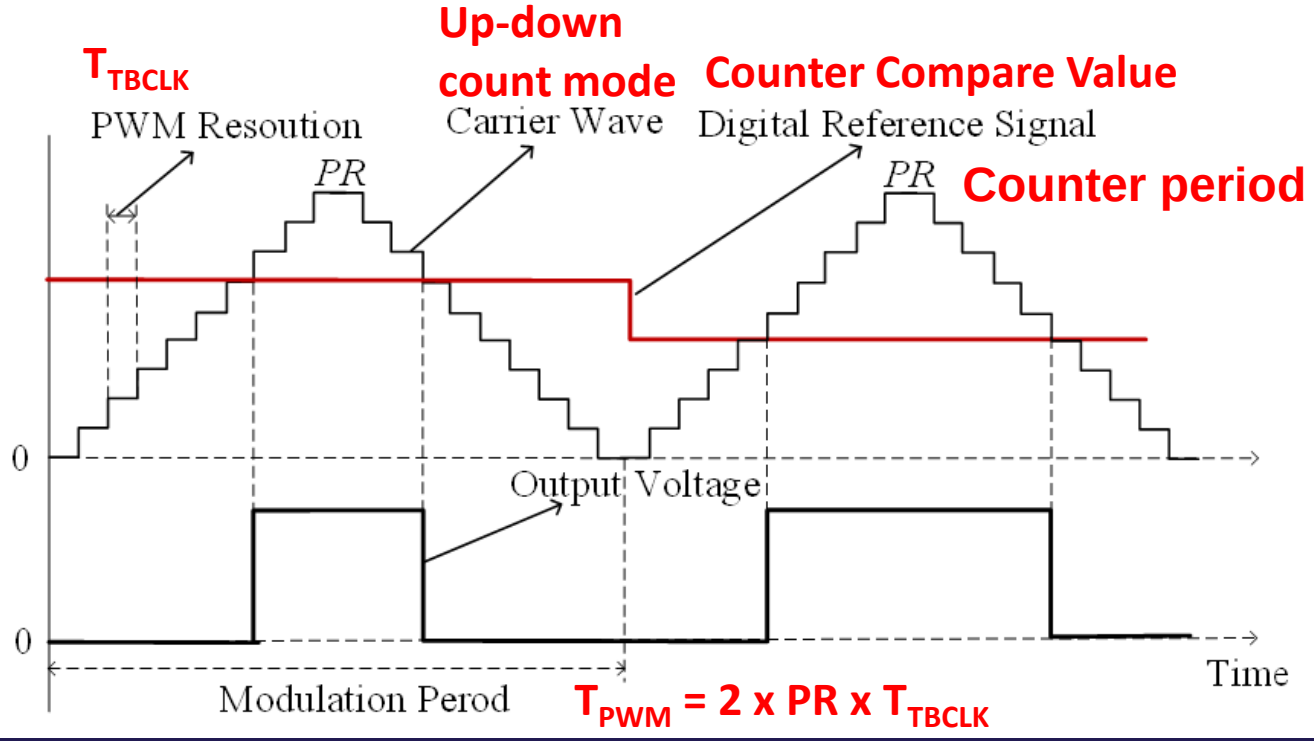
How is PWM generated digitally - **sawtooth carrier**?

- Reference value (CCV) > Timer count value – Pulse OFF
- Timer count value is zero – Pulse ON



How is PWM generated digitally - **triangular carrier**?

- Reference value (CCV) > Timer count value – Pulse ON
- Reference value (CCV) < Timer count value – Pulse OFF



PWM switching period/frequency

- The PWM switching period/frequency is determined by:
- The **timer clock TBCLK** which has a frequency $\mathbf{f_{TBCLCK}}$ and period (time base) $\mathbf{T_{TBCLK}}$ $\mathbf{f_{TBCLCK} = 1/T_{TBCLK}}$
  - **TBCLK** is the **= SYSCLK** or a pre-scaled of the **SYSCLK**
- **Timer counter period PR**
- **Timer counter mode**
  - **Up-count mode** or **down-count mode**

$$\mathbf{T_{PWM} = (PR + 1) \times T_{TBCLK}}$$

  - **Up-down-count mode – triangular carrier**

$$\mathbf{T_{PWM} = 2 \times PR \times T_{TBCLK}}$$

$$\mathbf{f_{PWM} = 1/T_{PWM}}$$

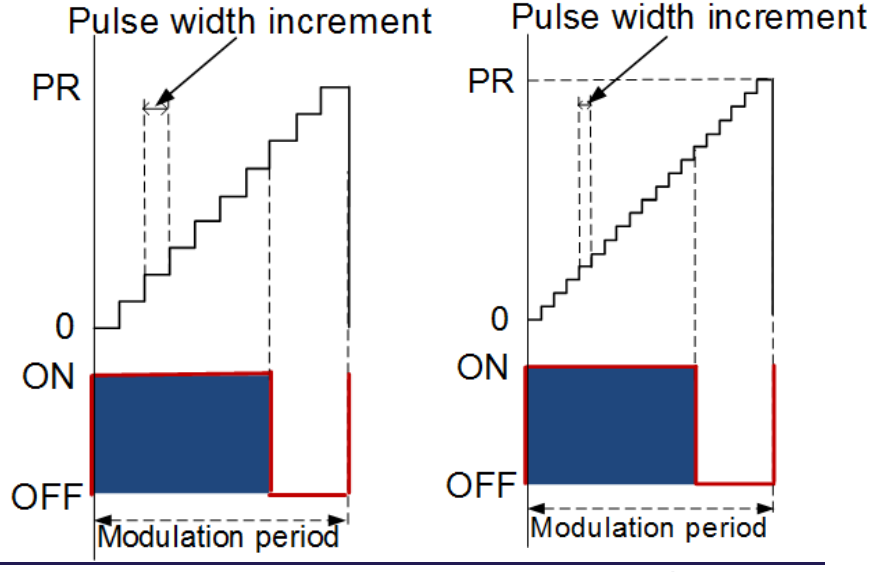
PWM resolution

- PWM resolution indicates the granularity of the duty cycle

$$PWM\_resolution = \log_2\left(\frac{f_{TBCLK}}{f_{PWM}}\right) \quad (bits)$$

- The higher the PWM resolution, the more precise approximation of a given reference is possible

For example, a 10-bit resolution PWM can have 1024 different widths for the pulses



PWM duty cycle

- The duty cycle is the compare value divided by the counter period:
- For up-count mode

$$Duty\,cycle = \frac{CCV}{PR+1}$$

- For symmetric up-down count mode

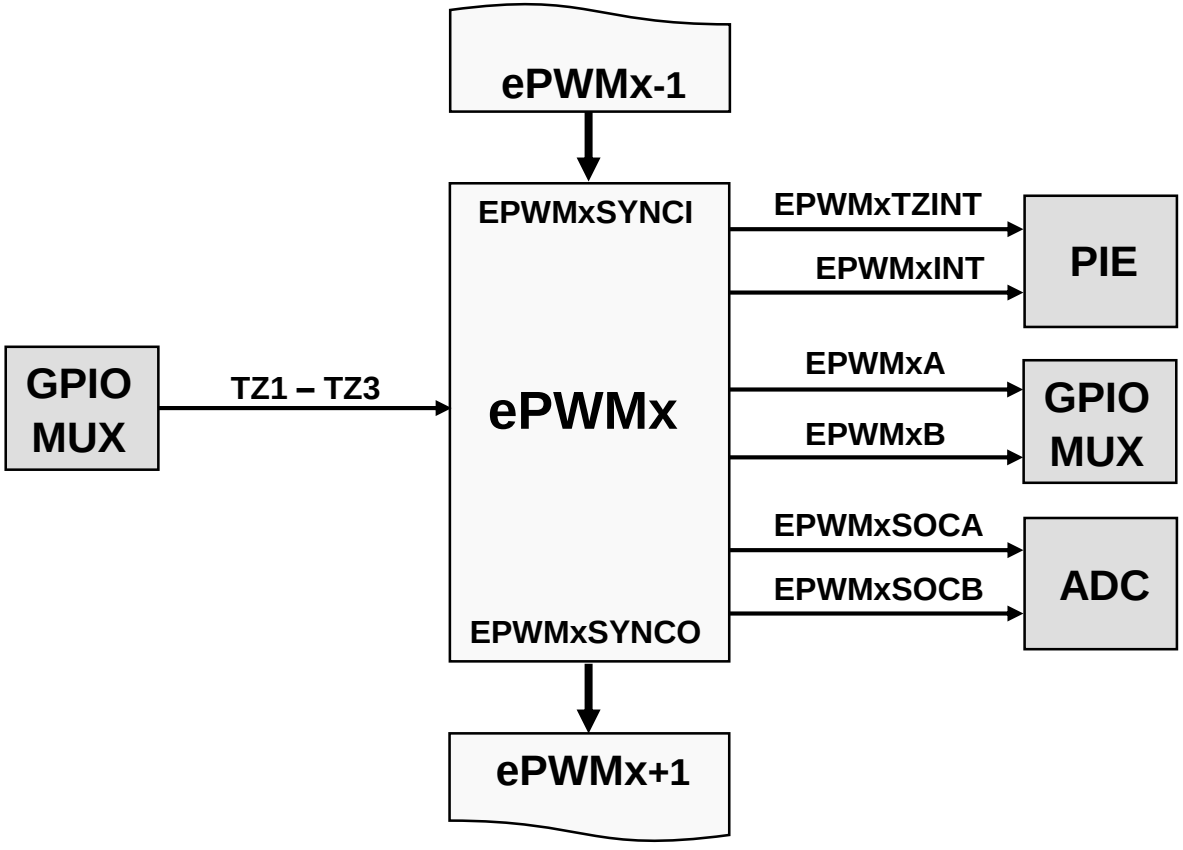
$$Duty\,cycle = \frac{PR-CCV}{PR} \qquad \text{or} \qquad Duty\,cycle = \frac{CCV}{PR}$$

- Duty Cycle can vary between 0-1 (in our case when it is 0 the I/O pin stays permanently in 0V while when is 1 is permanently 3.3V)

Agenda

- What is a DAC?
- Main parameters of a DAC
- Pulse-Width Modulation (PWM)
- **PWM module of F2806x**

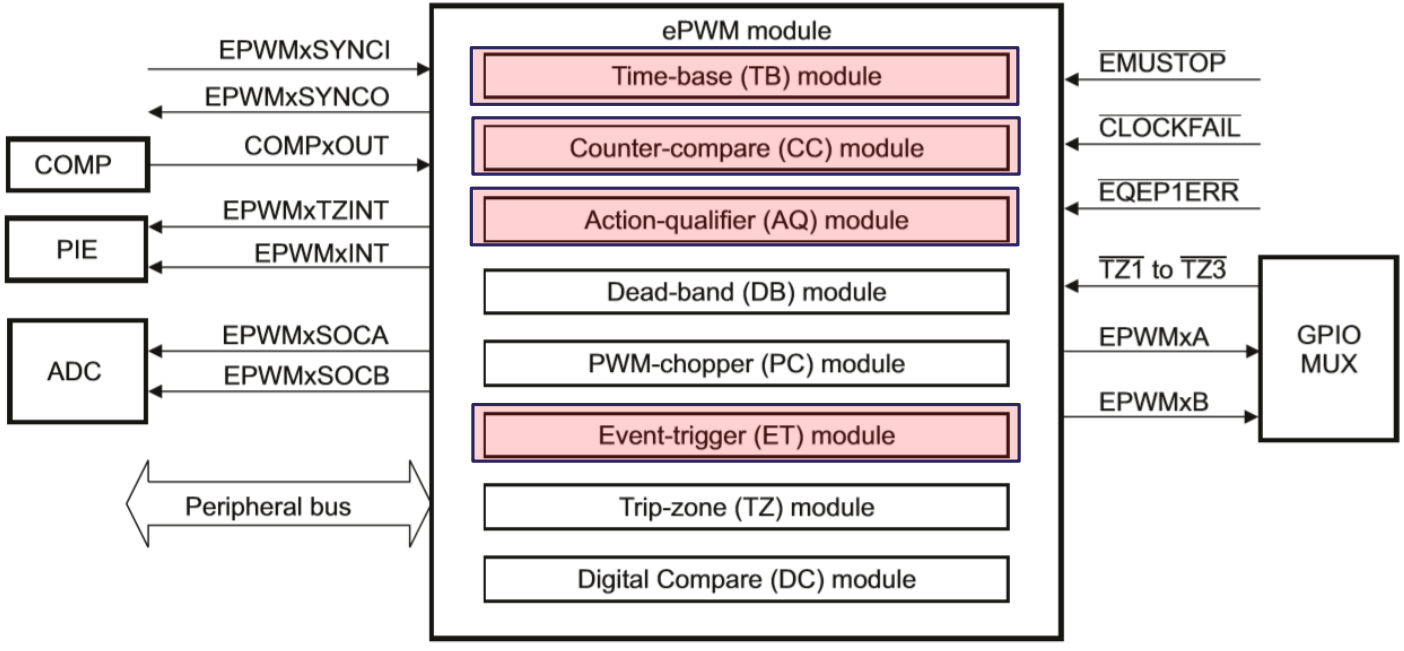
F2806x ePWM Block Diagram



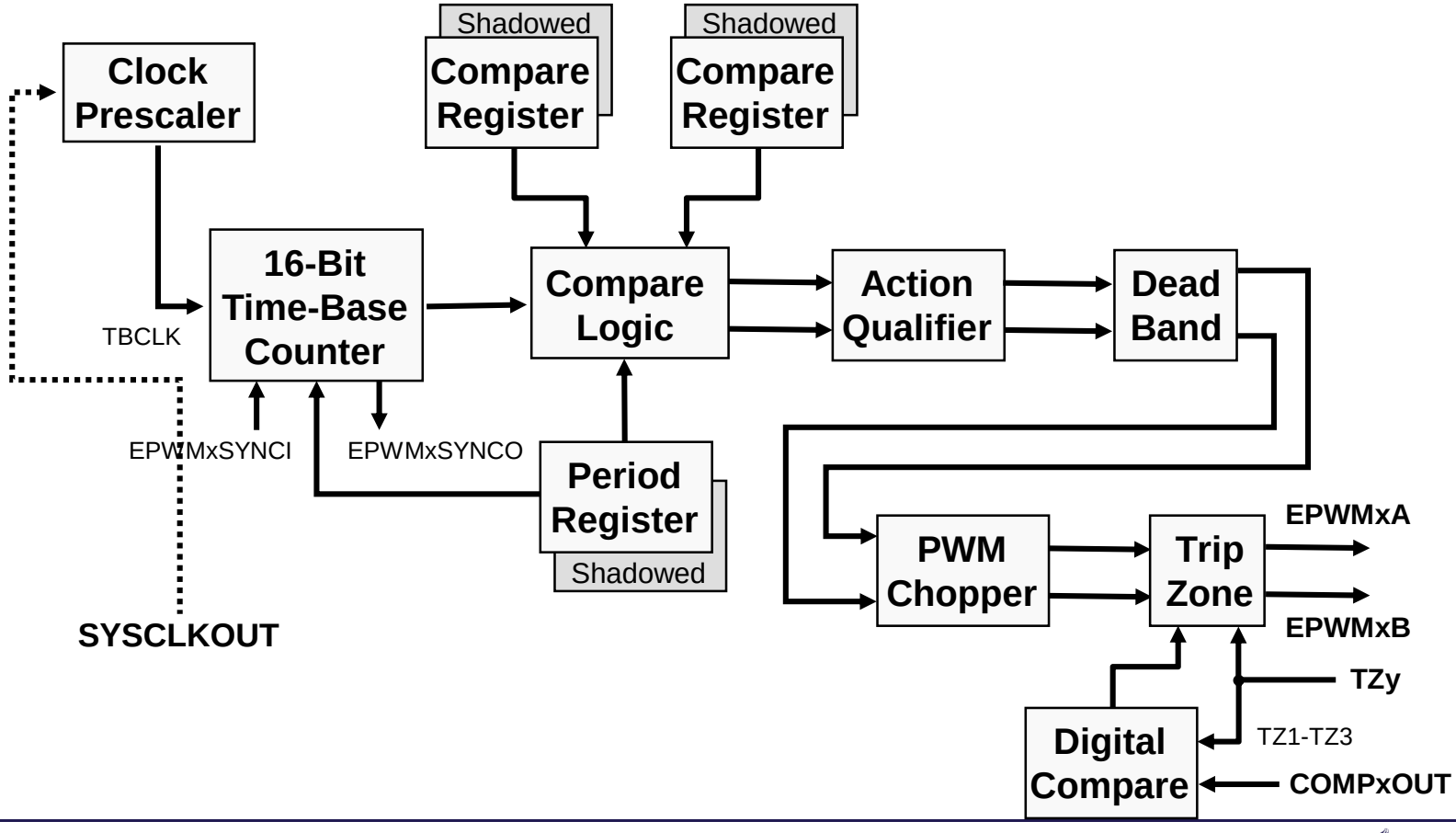
- 8 PWM modules
- 16 bit time-based counter
- Two independent outputs for each (A and B)
- ADC start event generation
- CPU interrupts
- Programmable trip zone events
- And more…

Submodules of the ePWM

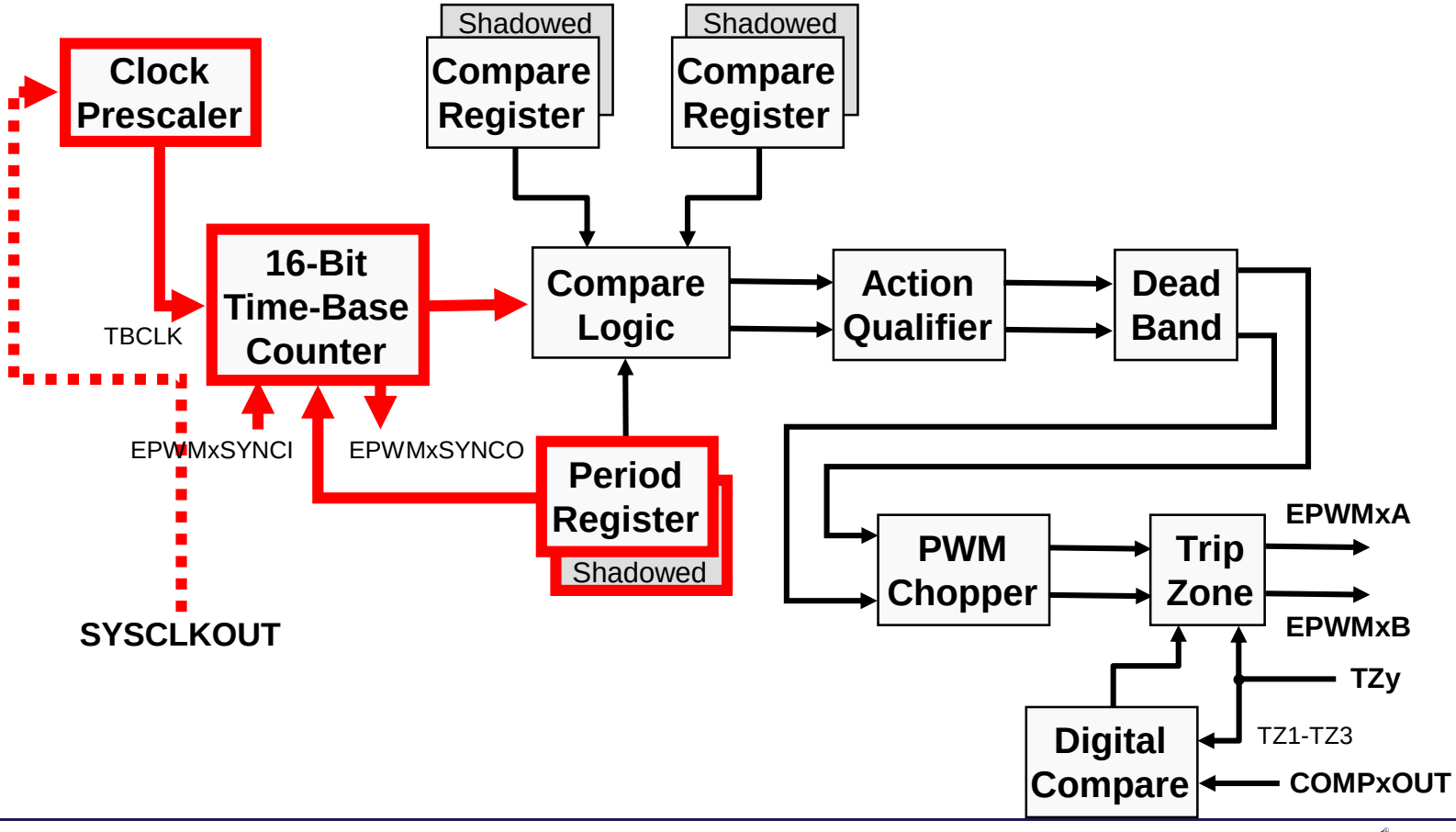
- Each of the six ePWM units consist of 8 submodules that need to be configure
- Some are optional depending on the requirements of the application



ePWM Block Diagram



ePWM Time-base (TB) submodule



ePWM Time-base (TB) submodule

- **Prescale** the **time-base clock** (TBCLK) relative to the system clock (SYSCLKOUT).
- Configure the **PWM time-base counter** (TBCTR) period (called PR in previous slides)
- Configure the **carrier wave type** = set the **mode** for the **time-base counter**:
  - **count-up mode**: used for asymmetric PWM
  - **count-down mode**: used for asymmetric PWM
  - **count-up-and-down mode**: used for symmetric PWM
- Configure the time-base phase relative to another ePWM module.
- Synchronize the time-base counter between modules through hardware or software.

ePWM Time-Base Sub-Module Registers

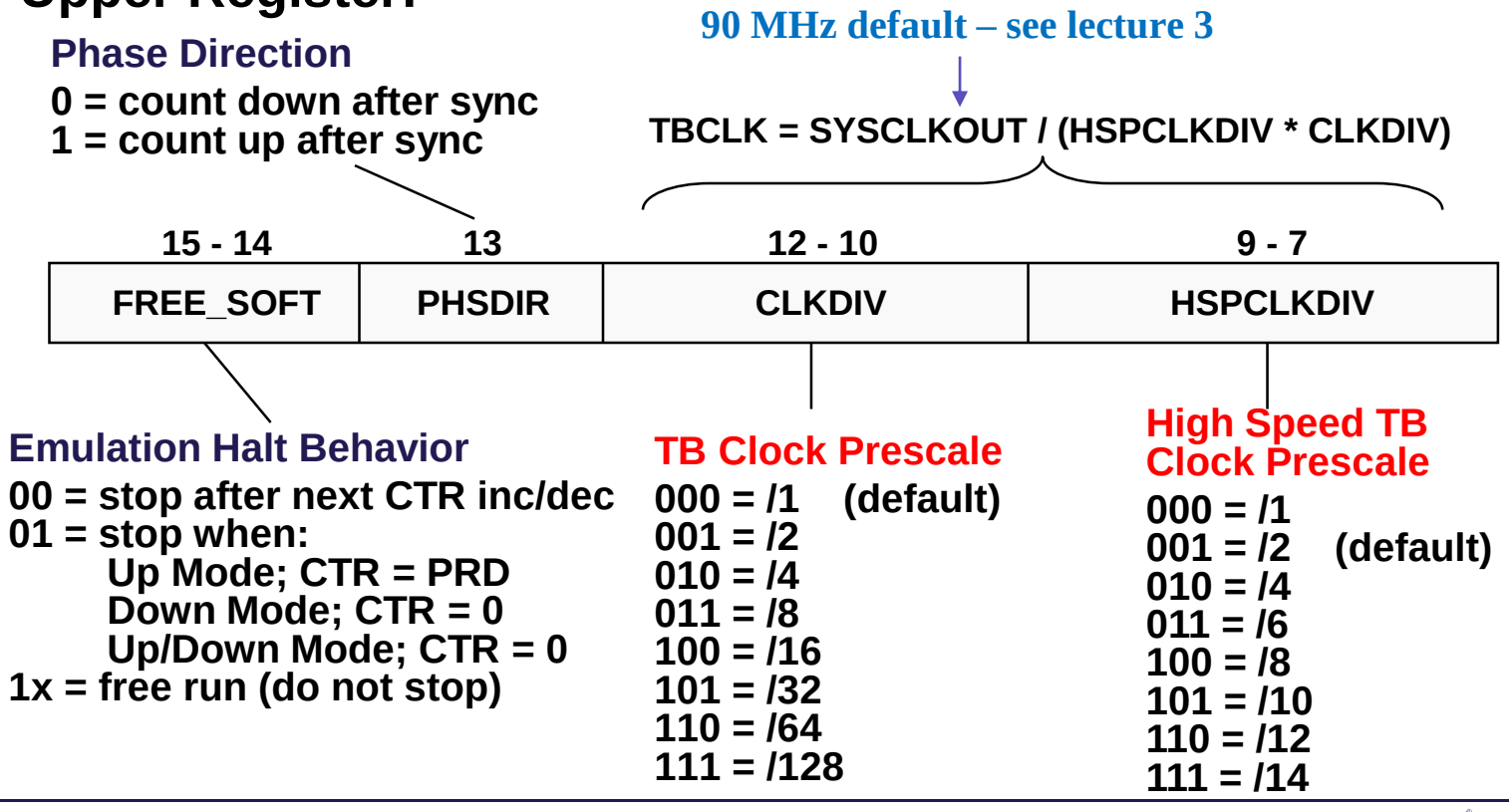
Register definition file **EPwm**x**Regs (**x **from 1 to 8)**

Name	Description	Structure
TBCTL	**Time-Base Control**	**EPwm*x*Regs.TBCTL.all =**
TBSTS	**Time-Base Status**	**EPwm*x*Regs.TBSTS.all =**
TBPHS	**Time-Base Phase**	**EPwm*x*Regs.TBPHS =**
TBCTR	**Time-Base Counter**	**EPwm*x*Regs.TBCTR =**
TBPRD	**Time-Base Period**	**EPwm*x*Regs.TBPRD =**

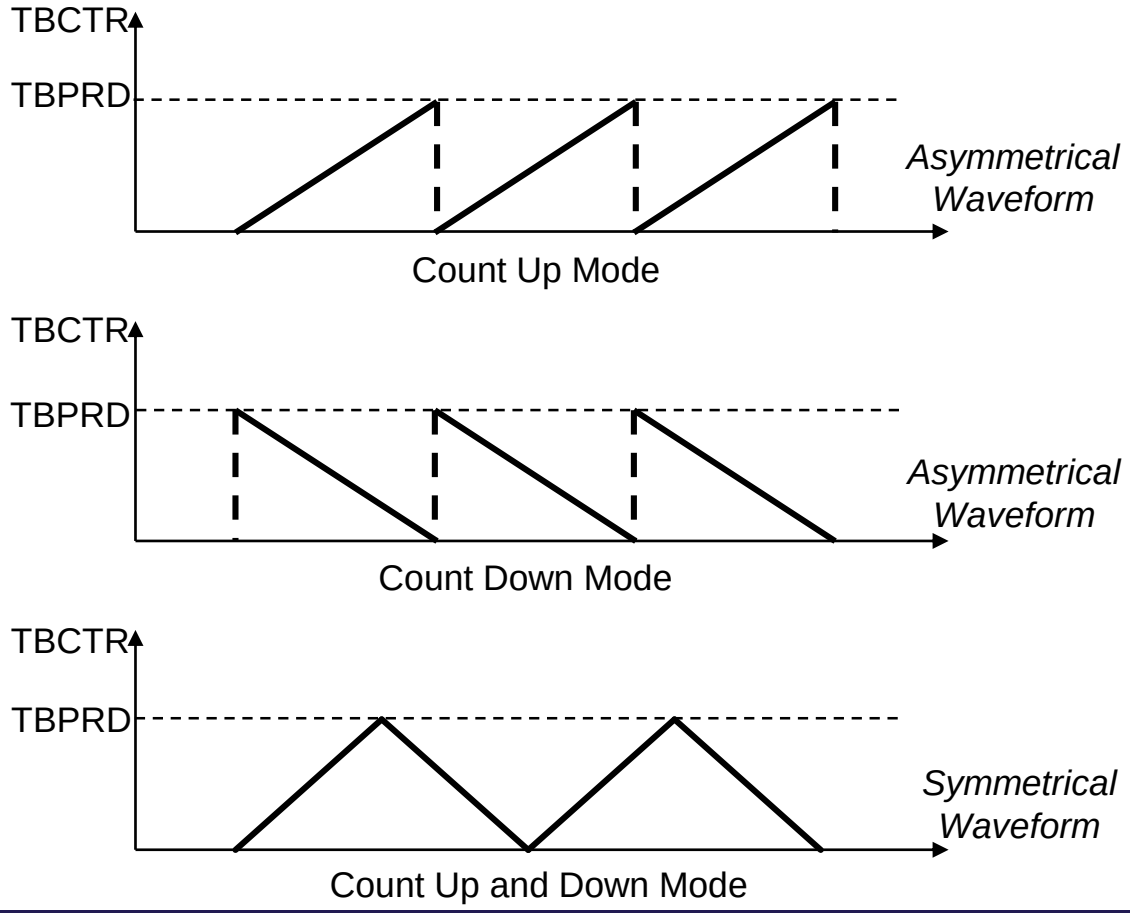
ePWM Time-Base Control Register

EPwmxRegs.TBCTL (x from 1 to 8)

Upper Register:



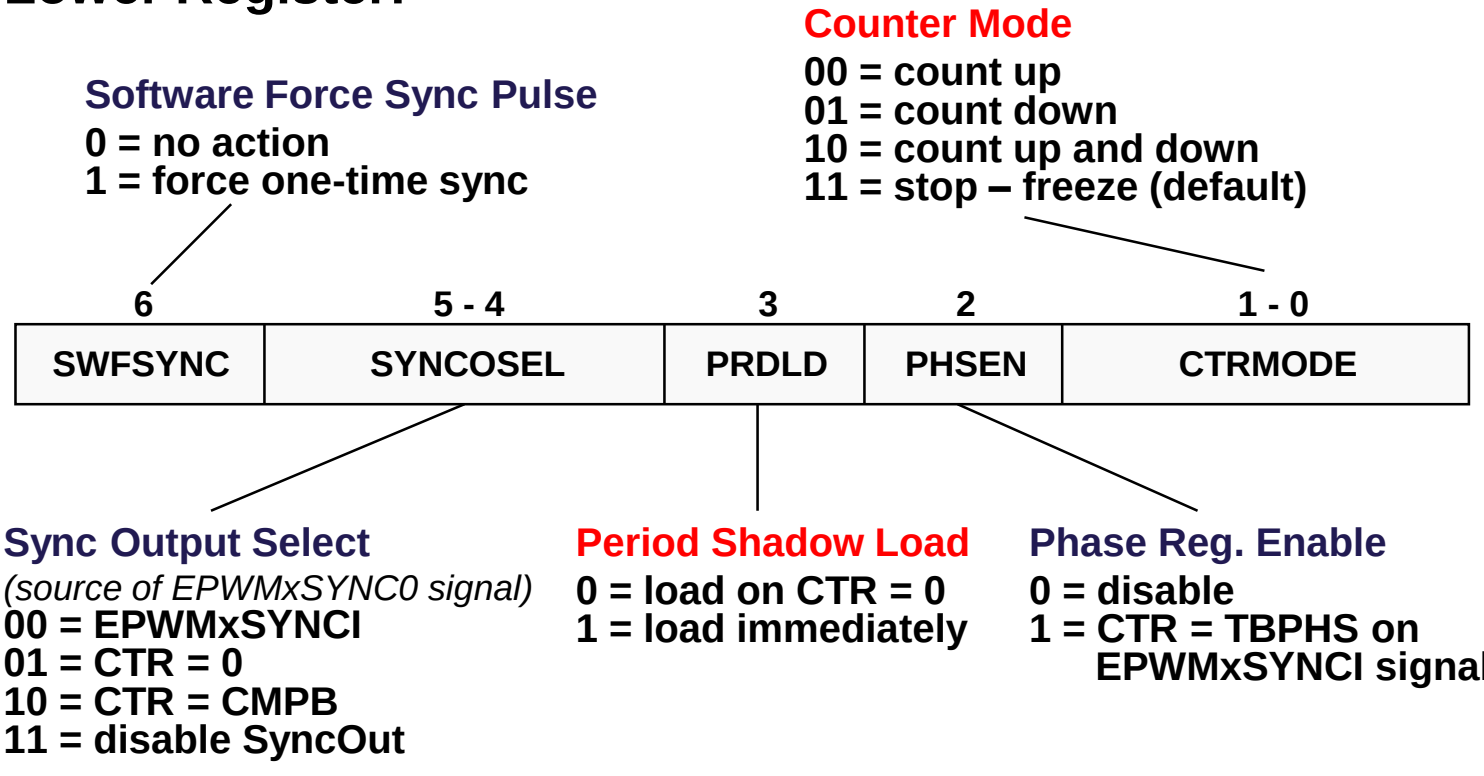
ePWM Time-base Count Modes



ePWM Time-Base Control Register

EPwmxRegs.TBCTL (x from 1 to 8)

Lower Register:



What is Register Shadowing?

- To avoid randomly changing the content of a register which contains hardware related information

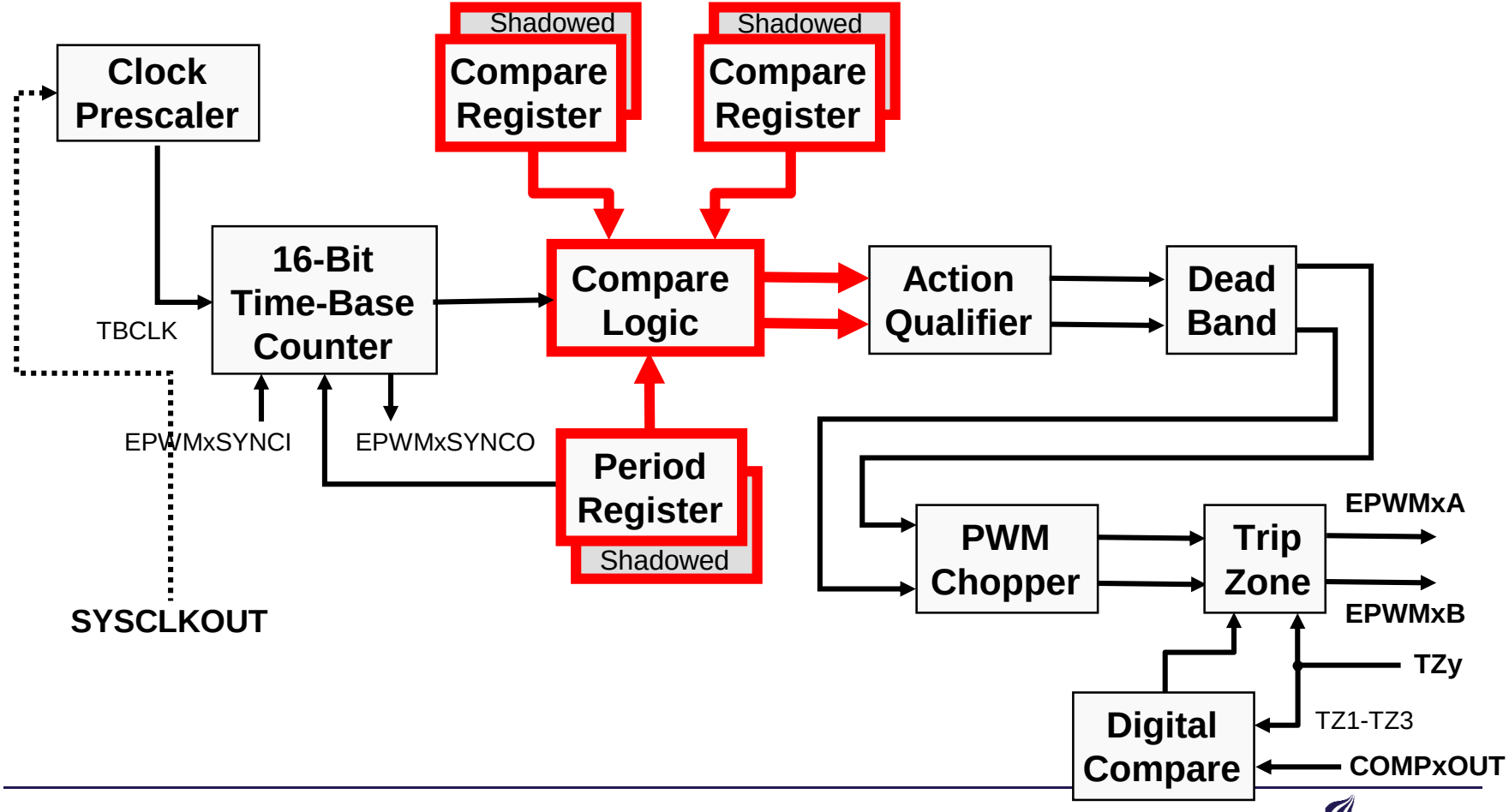
- Example:
  - Randomly changing the Period Register can cause unexpected long delay:
    - At the time when the counter value is 100 and we load to the PR register a value smaller 100 the counter will match the PR only after overflow which will take a long time
    - With shadow mechanism the new value of the PR is loaded only when the counter is 0 or PR

ePWM Time-base programming example

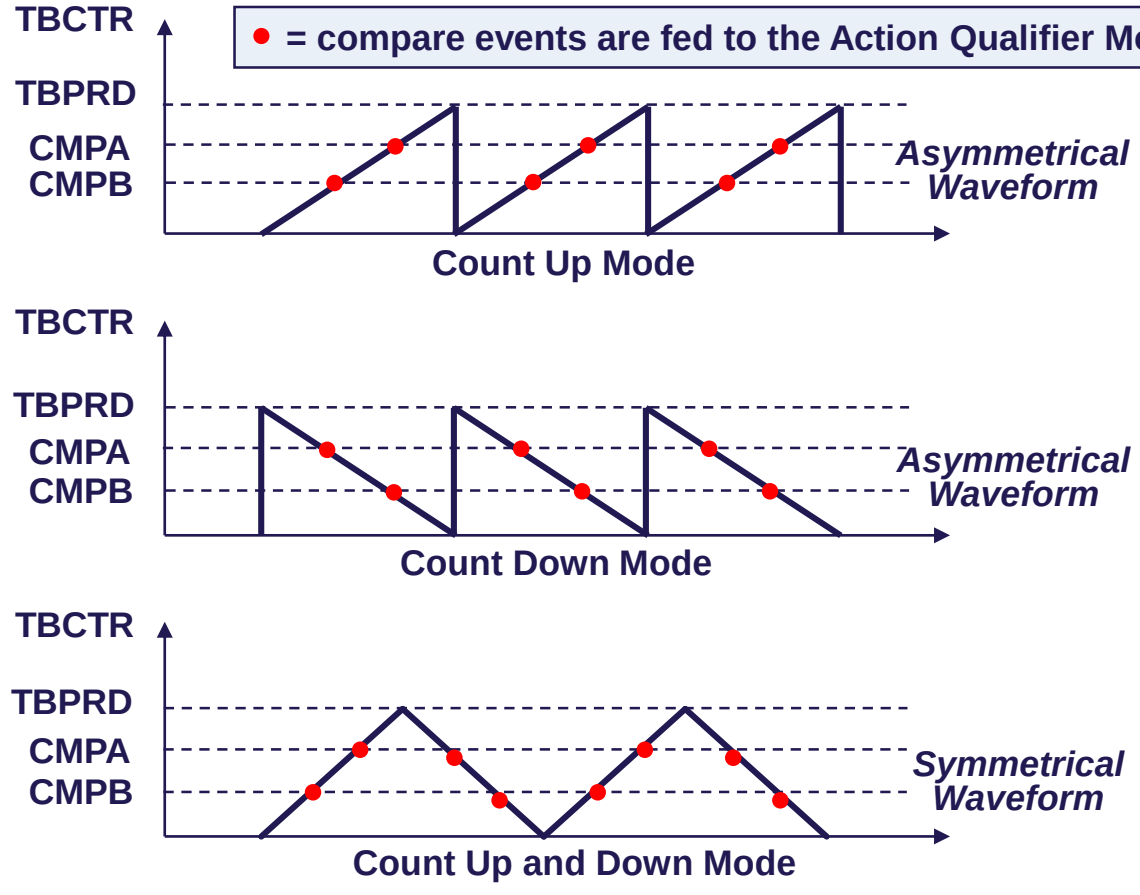
```
 // Setup time base counter for ePWM1
   EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;// Enable shadow
register
   EPwm1Regs.TBCTL.bit.CLKDIV = 0; // /1
   EPwm1Regs.TBCTL.bit.HSPCLKDIV = 0;// /1
   EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN; // up-down
mode
   EPwm1Regs.TBPRD = 15000;

   // Disable phase synchronization for ePWM1
   EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;
```

ePWM Counter Compare (CC) Submodule



ePWM Counter Compare (CC) Submodule



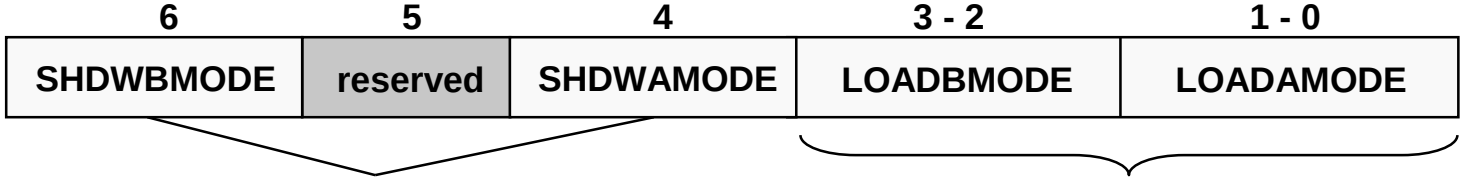
ePWM Counter Compare (CC) Submodule

- Used to configure the **PWM duty cycle** for output **EPWMxA** and/or output **EPWMxB**
- The **time-base counter value** is continuously compared to the **counter-compare A (CMPA)** and **counter-compare B (CMPB)** registers
- When the time-base counter is equal to one of the compare registers, the counter-compare unit generates an appropriate event.
- **Generates events** based on programmable time stamps using the CMPA and CMPB registers
  - CTR = CMPA: Time-base counter equals counter-compare A register (TBCTR = CMPA).
  - CTR = CMPB: Time-base counter equals counter-compare B register (TBCTR = CMPB)
- **Shadows new compare values** to prevent corruption or glitches during the active PWM cycle

ePWM Counter Compare (CC) Registers

Name	Description	Structure
CMPCTL	Compare Control	EPwm*x*Regs.CMPCTL.all =
CMPA	Compare A	EPwm*x*Regs.CMPA =
CMPB	Compare B	EPwm*x*Regs.CMPB =

CMPCTL



CMPA and CMPB Operating Mode

0 = shadow mode;
double buffer w/ shadow register
1 = immediate mode;
shadow register not used

CMPA and CMPB Shadow Load Mode

00 = load on CTR = 0
01 = load on CTR = PRD
10 = load on CTR = 0 or PRD
11 = freeze (no load possible)

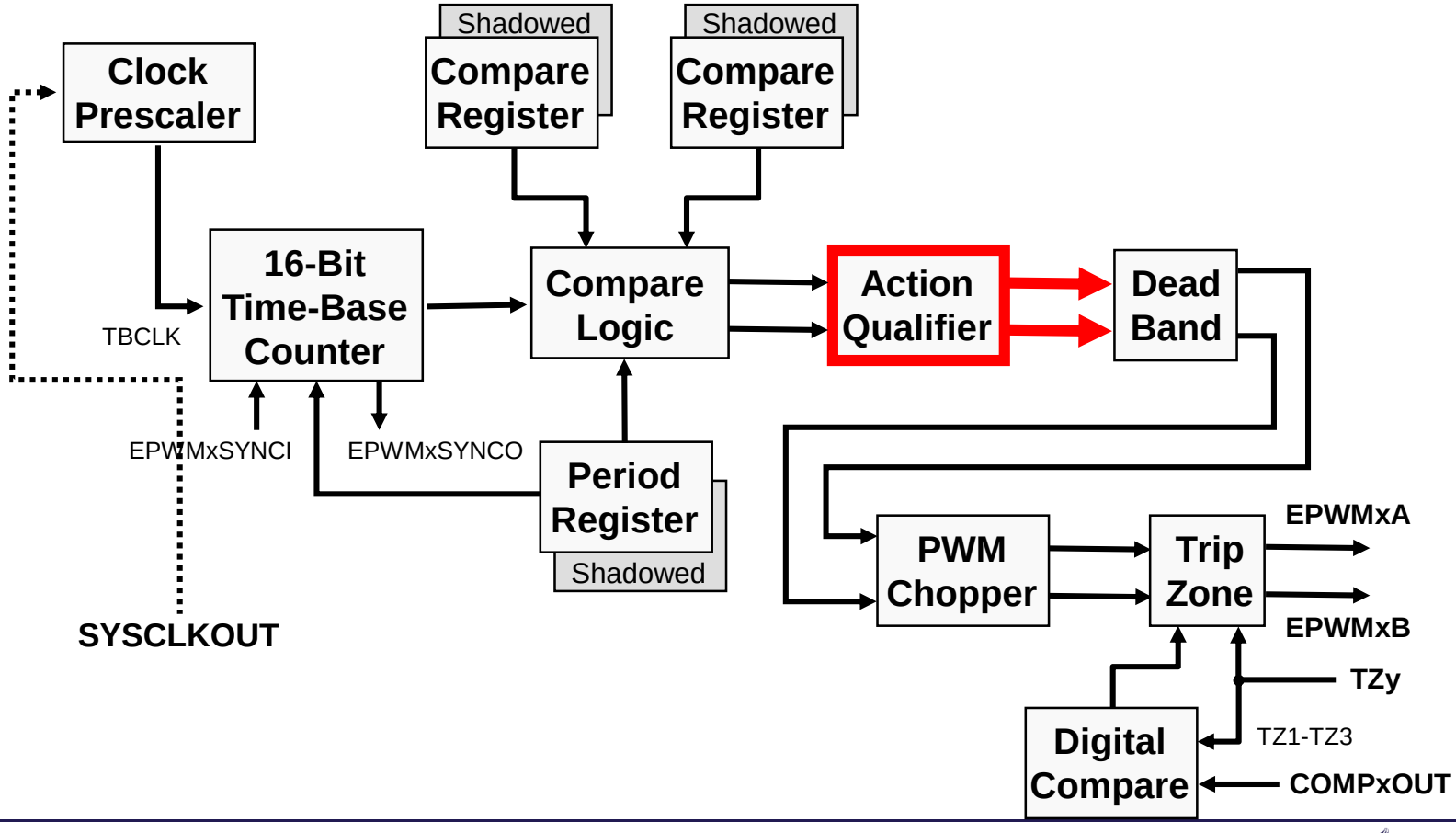
Counter Compare (CC) Programming Example

```
 // Configure the PWM compare units
   EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO_PRD; //load the
compare value in on CTR=0 and CTR=PRD

   EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW; //CMPR registers are
shadowed

   EPwm1Regs.CMPA.half.CMPA = EPwm1Regs.TBPRD / 4;// set to 50% in
up-down counter mode
```

ePWM Action Qualifier (AQ) Submodule



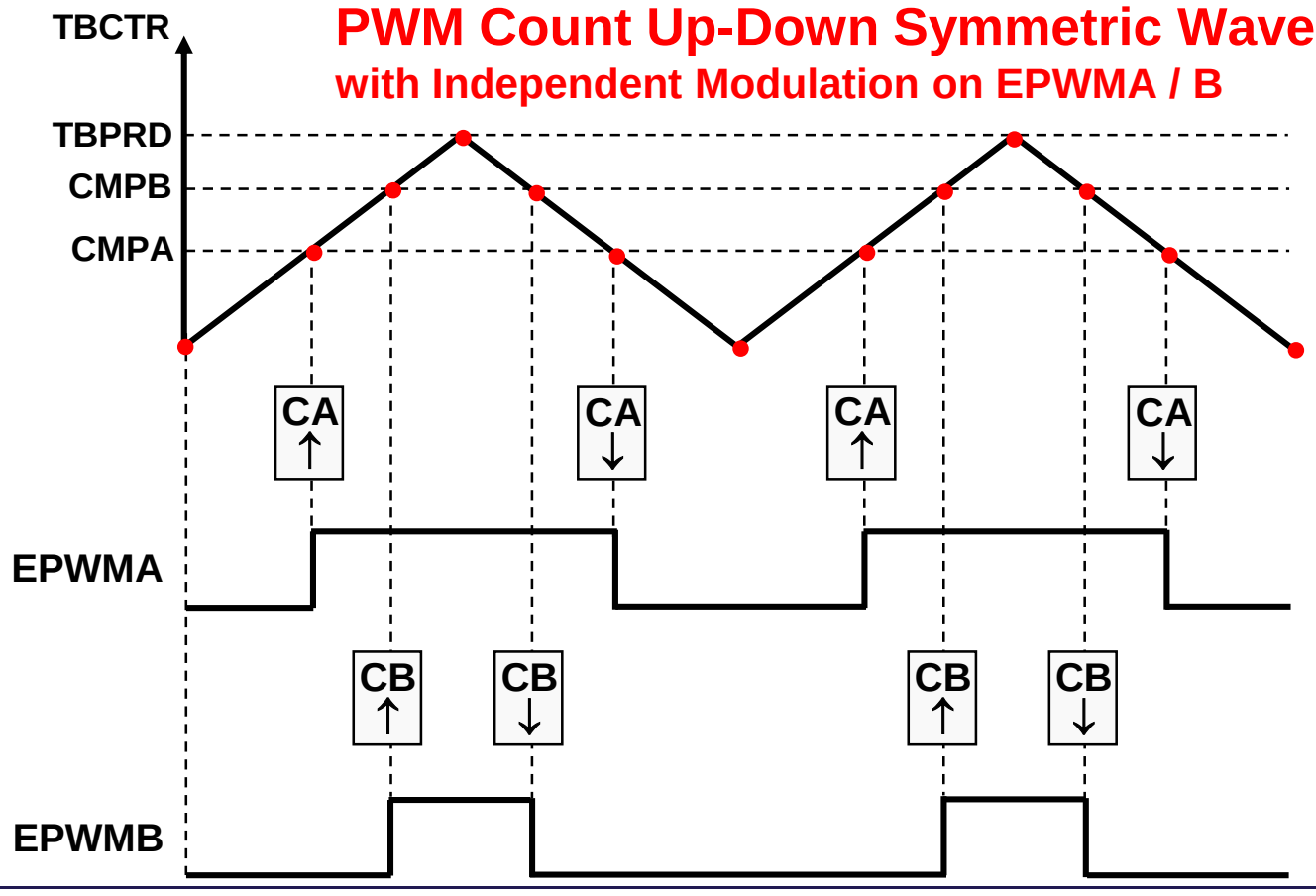
ePWM Action Qualifier (AQ) Submodule

- The **action-qualifier** submodule controls how the two outputs EPWMxA and EPWMxB behave when a particular event occurs.
- **Events** can be CTR=0, CTR=PRD, CTR=CMPA (on counter up), CTR=CMPA (on counter down), CTR=CMPB (on counter up), CTR=CMPB (on counter down)
- Type of **action** taken **when** a **time-base** or **counter-compare** submodule **event** occurs:
  - No action taken
  - Output EPWMxA and/or EPWMxB switched high
  - Output EPWMxA and/or EPWMxB switched low
  - Output EPWMxA and/or EPWMxB toggled
- **Force** the **PWM output** state through **software control**
- Actions are specified independently for either output (EPWMxA or EPWMxB).

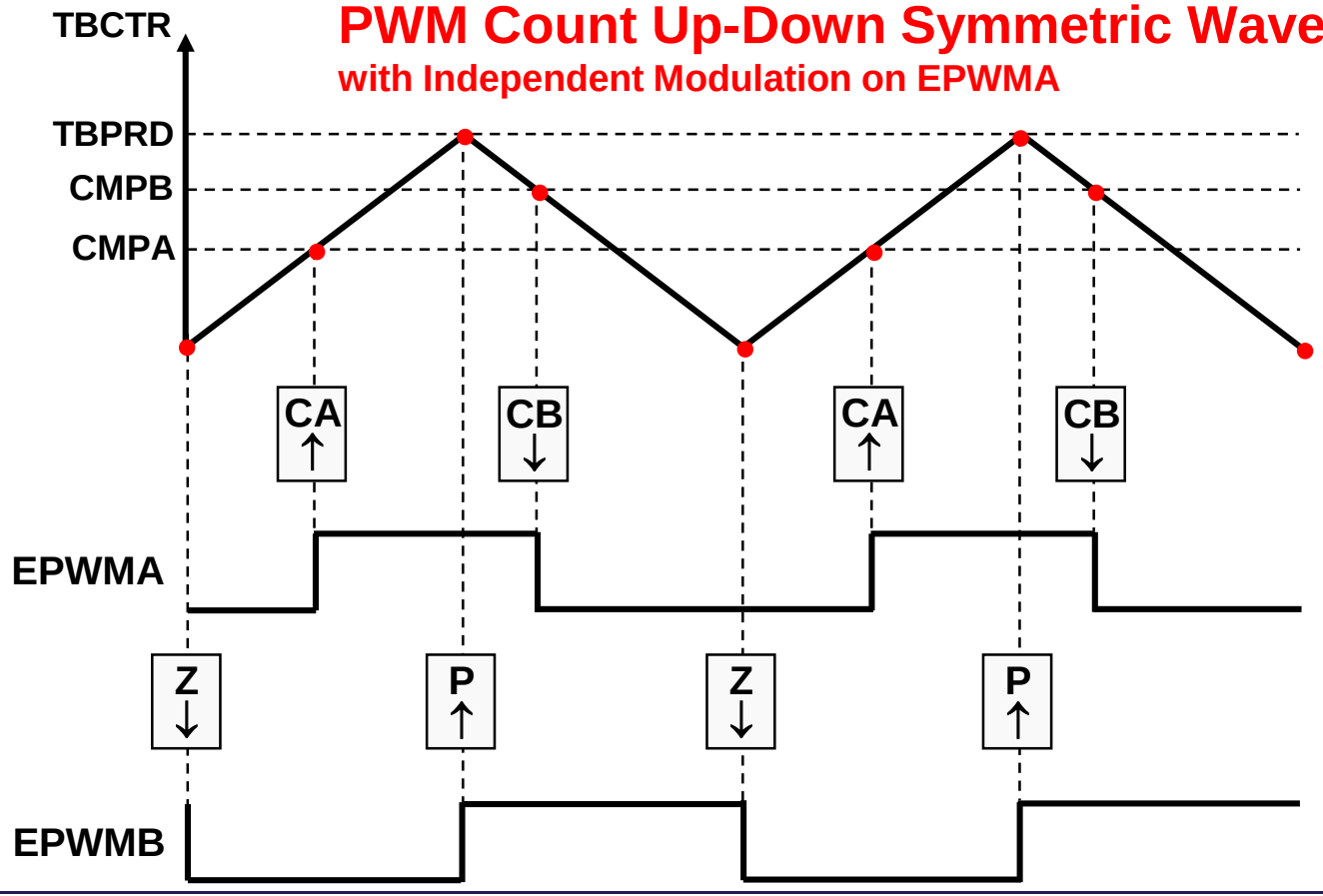
PWM Action Qualifier (AQ) Actions

S/W Force	Time-Base Counter equals:				EPWM Output Actions
	Zero	CMPA	CMPB	TBPRD	
SW X	Z X	CA X	CB X	P X	Do Nothing
SW ↓	Z ↓	CA ↓	CB ↓	P ↓	Clear Low
SW ↑	Z ↑	CA ↑	CB ↑	P ↑	Set High
SW T	Z T	CA T	CB T	P T	Toggle

ePWM Action Qualifier (AQ) Example 1

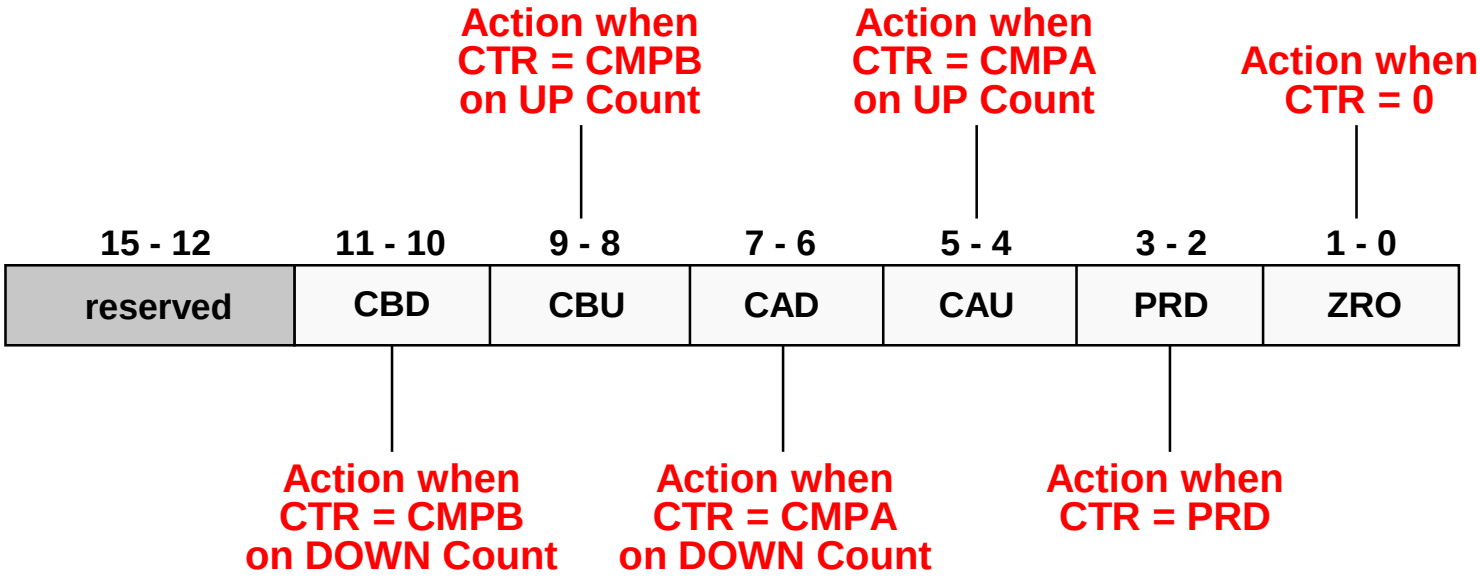


ePWM Action Qualifier (AQ) Example 2



ePWM Action Qualifier (AQ) Registers

EPwmxRegs.AQCTLy (y = A or B)



00 = do nothing (action disabled)
01 = clear (low)
10 = set (high)
11 = toggle (low → high; high → low)

Action Qualifier Programming Example

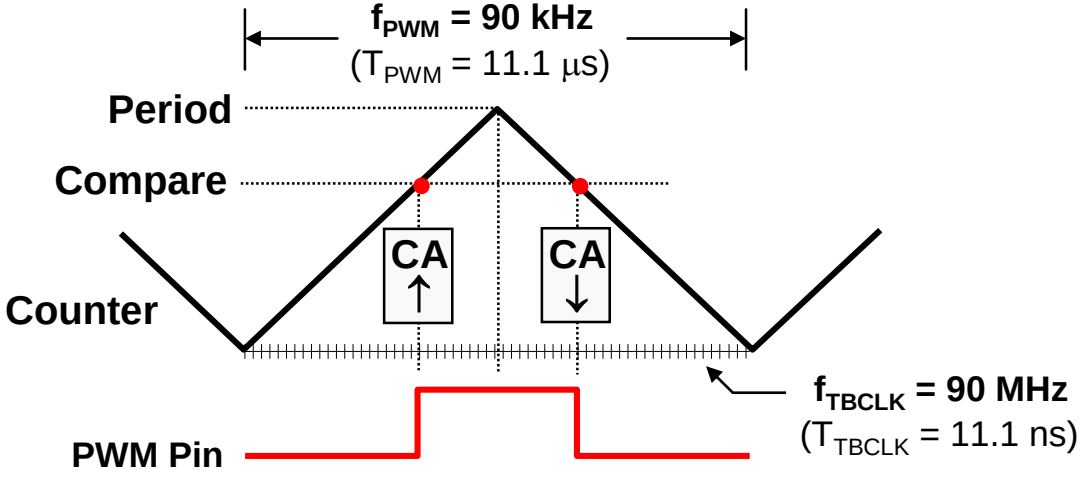
```
// Configure the action qualifier for output EPWM1A
   EPwm1Regs.AQCTLA.all = 0;
   EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;
   EPwm1Regs.AQCTLA.bit.CAD = AQ_CLEAR;

// Configure the action qualifier for output EPWM1B
   EPwm1Regs.AQCTLB.all = 0;
   EPwm1Regs.AQCTLB.bit.CBU = AQ_SET;
   EPwm1Regs.AQCTLB.bit.CBD = AQ_CLEAR;
```

See ePWM Action Qualifier (AQ) Example 1

Symmetric PWM Computation Example

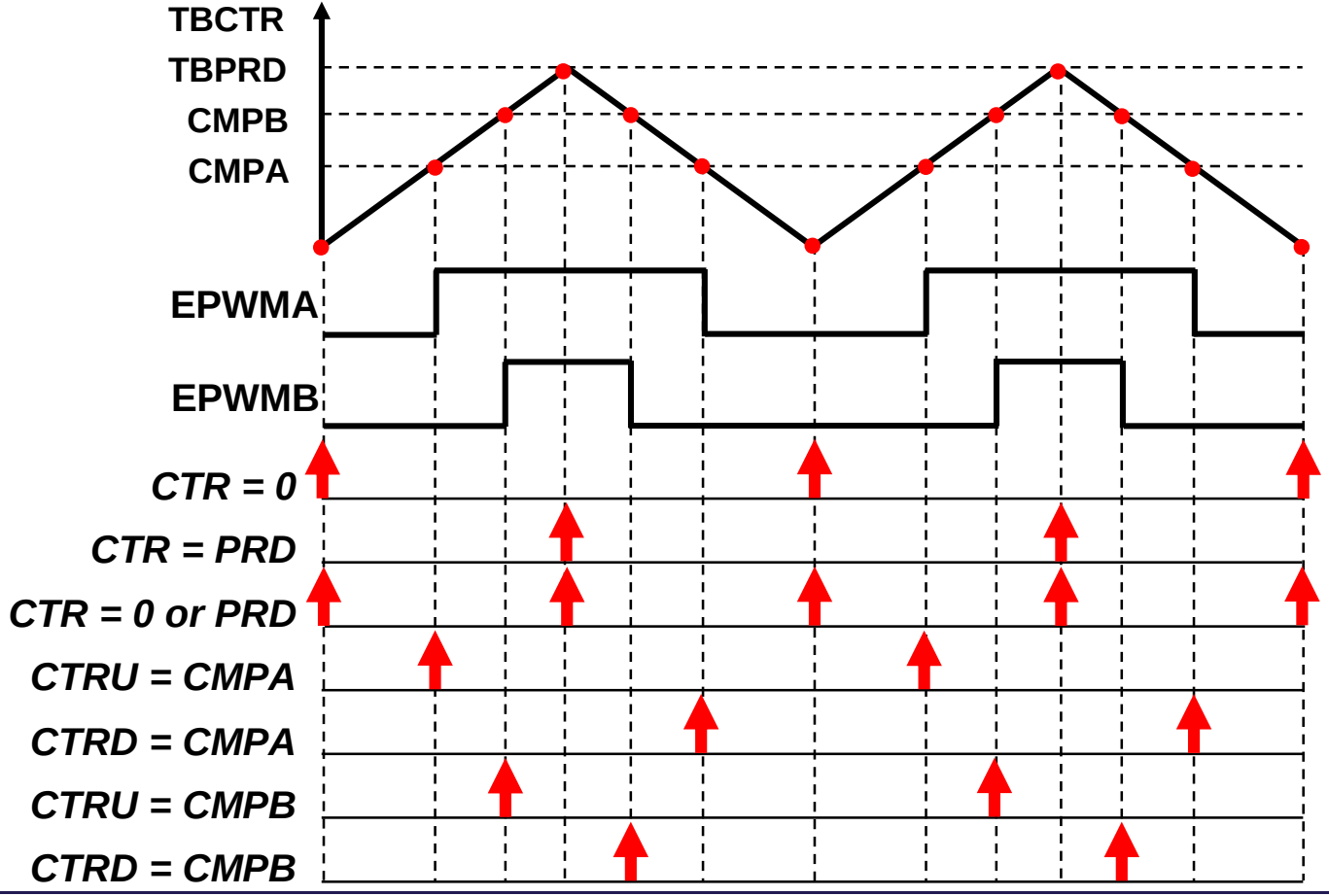
- **Determine TBPRD and CMPA for 90 kHz, 25% duty symmetric PWM from a 90 MHz time base clock**



$$TBPRD = \frac{1}{2} \cdot \frac{f_{TBCLK}}{f_{PWM}} = \frac{1}{2} \cdot \frac{90\ MHz}{90\ kHz} = 500$$

CMPA = (100% - duty cycle)*TBPRD = 0.75*500 = 375

ePWM **Events** that can **trigger Interrupts** and/or an **ADC** Start-of-Conversion (SOC)



ePWM Event trigger (ET) submodule

- **Each ePWM** module has **one interrupt request line** connected to the **PIE (EPWMx_INT)** and **two start of conversion signals** (one for each sequencer **EPWMx_SOCA** and **EPWMx_SOCB**) connected to the **ADC module**.
- **ET submodule monitors** various **event conditions**, such as CTR = 0, CTR = PRD, CTR = CMPA, CTR = CMPB

- In the ET submodule
  - **Enable** the **ePWM events** that will **trigger** an **interrupt**
  - **Enable ePWM events** that will **trigge**r an **ADC start-of-conversion event.**
    - *In motor control a current sampling is done at CTR = 0 or CTR = PRD because it is happening when the current is close to the average value.*
  - **Specify** the **rate** at which **events** cause **triggers** (every occurrence or every second or third occurrence)

ePWM Event trigger (ET) Registers

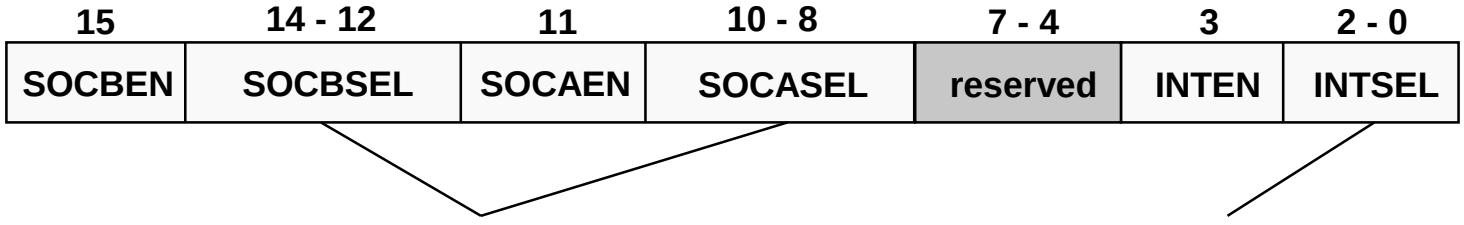
Name	Description	Structure
ETSEL	Event-Trigger Selection	EPwm*x*Regs.ETSEL.all =
ETPS	Event-Trigger Pre-Scale	EPwm*x*Regs.ETPS.all =
ETFLG	Event-Trigger Flag	EPwm*x*Regs.ETFLG.all =
ETCLR	Event-Trigger Clear	EPwm*x*Regs.ETCLR.all =
ETFRC	Event-Trigger Force	EPwm*x*Regs.ETFRC.all =

ePWM Event trigger (ET) Selection Registers

EPwm*x*Regs.ETSEL

Enable SOCB / A
0 = disable
1 = enable

Enable EPWMxINT
0 = disable
1 = enable



EPWMxSOCB / A Select
000 = DCBEVT1 / DCAEVT1
001 = CTR = 0
010 = CTR = PRD
011 = CTR = 0 or PRD
100 = CTRU = CMPA
101 = CTRD = CMPA
110 = CTRU = CMPB
111 = CTRD = CMPB

EPWMxINT Select
000 = reserved
001 = CTR = 0
010 = CTR = PRD
011 = CTR = 0 or PRD
100 = CTRU = CMPA
101 = CTRD = CMPA
110 = CTRU = CMPB
111 = CTRD = CMPB

PWM Event trigger (ET) Prescale Registers

EPwm*x*Regs.ETPS

EPWMxSOCB / A Counter

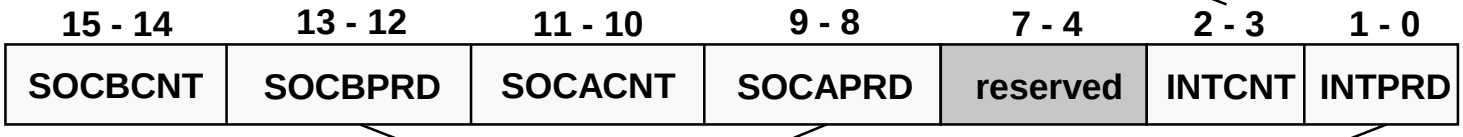
(number of events have occurred)

00 = no events
01 = 1 event
10 = 2 events
11 = 3 events

EPWMxINT Counter

(number of events have occurred)

00 = no events
01 = 1 event
10 = 2 events
11 = 3 events



EPWMxSOCB / A Period

(number of events before SOC)

00 = disabled
01 = SOC on first event
10 = SOC on second event
11 = SOC on third event

EPWMxINT Period

(number of events before INT)

00 = disabled
01 = INT on first event
10 = INT on second event
11 = INT on third event

Event trigger Programming Example

```
  // Configure interrupt generation
   EPwm1Regs.ETSEL.bit.INTEN = 1;  // Enable INT
   EPwm1Regs.ETSEL.bit.INTSEL = ET_CTR_ZERO;    // Generate INT
on Zero event
EPwm1Regs.ETPS.bit.INTPRD = ET_1ST;          // Generate INT on
1st event


   // Configure ADC start of conversion trigger generation
   EPwm1Regs.ETSEL.bit.SOCASEL = ET_CTR_PRD;// Trigger ADC on
CTR = PRD
   EPwm1Regs.ETSEL.bit.SOCAEN = 1;// Enable SOCA trigger
   EPwm1Regs.ETPS.bit.SOCAPRD = ET_1ST;// SOCA trigger on each
event
```

Event trigger Programming Example

```
void main{
…
EALLOW;  // This is needed to write to EALLOW protected register
    PieVectTable.EPWM1_INT = &ePWM1_isr;
EDIS;    // This is needed to disable write to EALLOW protected registers
PieCtrlRegs.PIEIER3.bit.INTx1 = 1;
…
}

interrupt void ePWM1_isr(void)
{
//User code

// To receive more interrupts ePWM module
EPwm1Regs.ETCLR.bit.INT = 1;

  // To receive more interrupts from this PIE group, acknowledge this interrupt
PieCtrlRegs.PIEACK.bit.ACK3 = 1;
}
```

Setting up the ePWM module

1. Configure the GPIO output:
    1. Disable the pull up resistors for the I/O pin
    2. Select the PWM unit to control the I/O pin
2. Configure the PWM Time-base (TB) module
    1. Configure TBCLK
    2. Configure the period register (switching frequency)
    3. Enable Shadow registers
    4. Configure timer counting mode (carrier wave type)
    5. Synchronize the PWM units (if more than one is used and sync is necessary)
3. Configure the Counter-Compare (CC) module
    1. Enable shadow registers
    2. Configure counter load mode (event)
    3. Setup initial CMPA and CMPB value (depending on which you need to use)
4. Configure the Action-Qualifier (AQ) module
5. Configure the Event-Trigger (ET) module
    1. Enable/Disable and select the events that trigger CPU interrupts
    2. Enable/Disable and select the events that trigger ADC SOCA and/or SOCB

Summary Digital to Analog converters

- What is a DAC?
- Main parameters of a DAC
- Theory behind PWM
- PWM module of F28069