Tecnológico de Monterrey

Microcontrollers

Professor: Dr. Gilberto Ochoa Ruiz

Topic 8: **PWM and DC Motor Control**

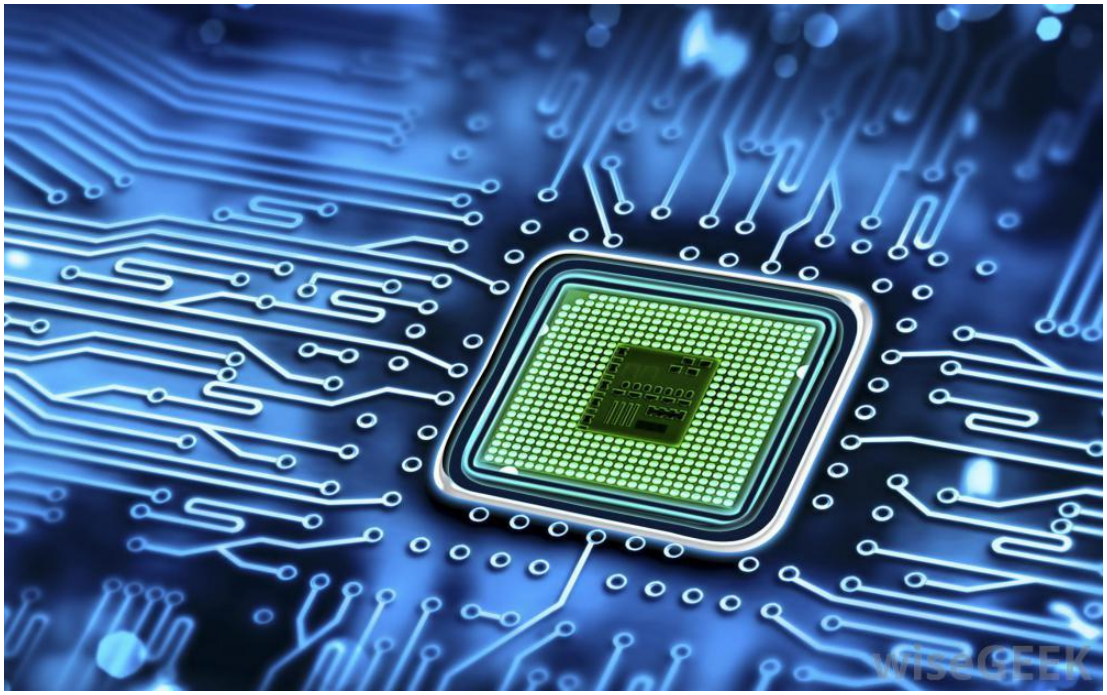
EIAD-204

Wednesday

May 20th 2020

6h30 – 9h30 PM

Guadalajara, Mexico



Basic Programming Techniques – PWM

DC Motor Interfacing and PWM

This section begins with an overview of the basic operation of the DC motors.

Then we describe how to interface a DC motor to the ARM.

Finally, we use C language programs to demonstrate the concept of pulse width modulation (PWM) and show how to control the speed and direction of a DC motor.

DC Motor Interfacing and PWM

A direct current (DC) motor is a widely used device that translates electrical current into mechanical movement.

In the DC motor we have only + and – leads.

Connecting them to a DC voltage source moves the motor in one direction.

By reversing the polarity, the DC motor will rotate in the opposite direction. One can easily experiment with the DC motor.

DC Motor Interfacing and PWM

For example, some small fans used in many motherboards to cool the CPU are run by DC motors.

While a stepper motor moves in discrete steps of 1 to 15 degrees, the DC motor moves continuously.

In a stepper motor, if we know the starting position we can easily count the number of steps the motor has moved and calculate the final position of the motor.

This is not possible in a DC motor. The maximum speed of a DC motor is indicated in RPM and is given in the data sheet.

DC Motor Interfacing and PWM

The DC motor has two types of RPM: no-load and loaded.

The manufacturer's data sheet gives the no-load RPM.

The no-load RPM can be from a few thousand to tens of thousands.

The RPM is reduced when moving a load and it decreases as the load is increased.

For example, a drill turning a screw has a much lower RPM speed than when it is in the no-load situation

DC Motor Interfacing and PWM

DC motors also have voltage and current ratings. The nominal voltage is the voltage for that motor under normal conditions, and can be from 1 to 150 V, depending on the motor.

As we increase the voltage, the RPM goes up. The current rating is the current consumption when the nominal voltage is applied with no load, and can be from 25 mA to a few amps.

As the load increases, the RPM is decreased, unless the current or voltage provided to the motor is increased, which in turn increases the torque.

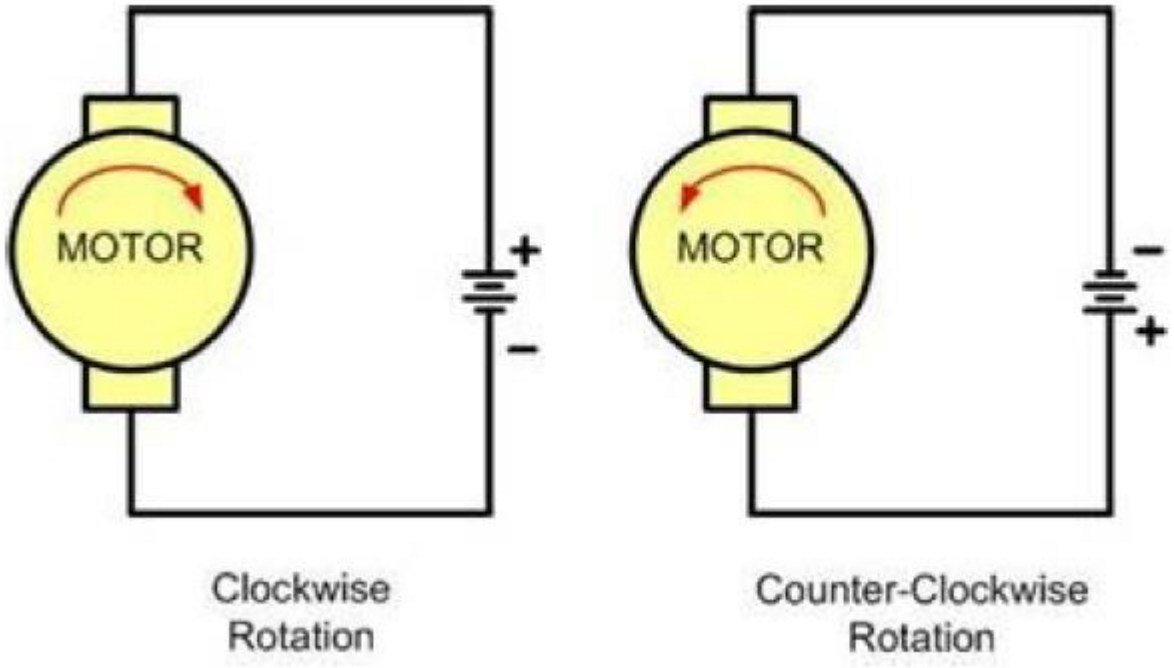
DC Motor Interfacing and PWM

Part No.	Nominal Volts	Volt Range	Current	RPM	Torque
154915CP	3 V	1.5–3 V	0.070 A	5,200	4.0 g-cm
154923CP	3 V	1.5–3 V	0.240 A	16,000	8.3 g-cm
177498CP	4.5 V	3–14 V	0.150 A	10,300	33.3 g-cm
181411CP	5 V	3–14 V	0.470 A	10,000	18.8 g-cm

Selected DC Motor Characteristics (http://www.Jameco.com)

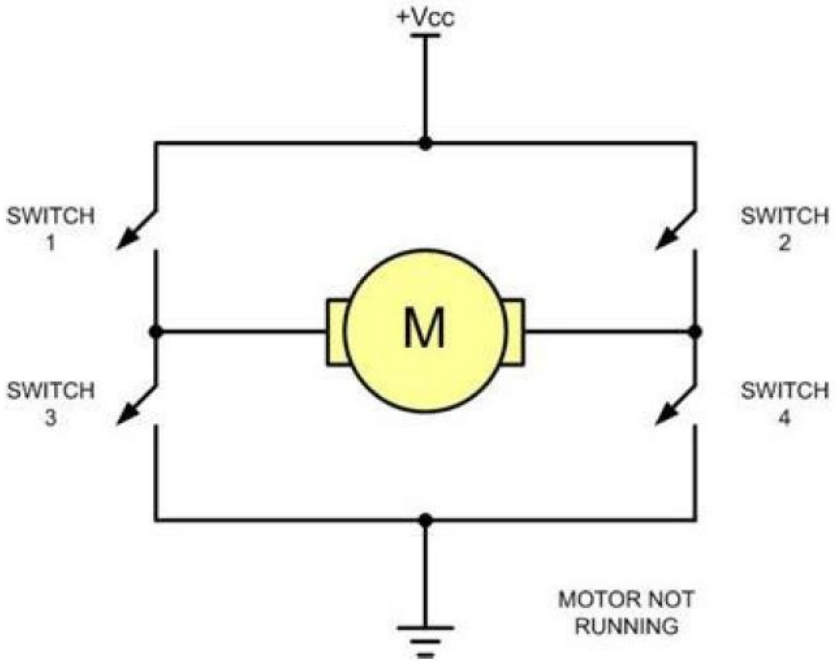
Unidirectional control

The figure below shows the DC motor clockwise (CW) and counterclockwise (CCW) rotations.



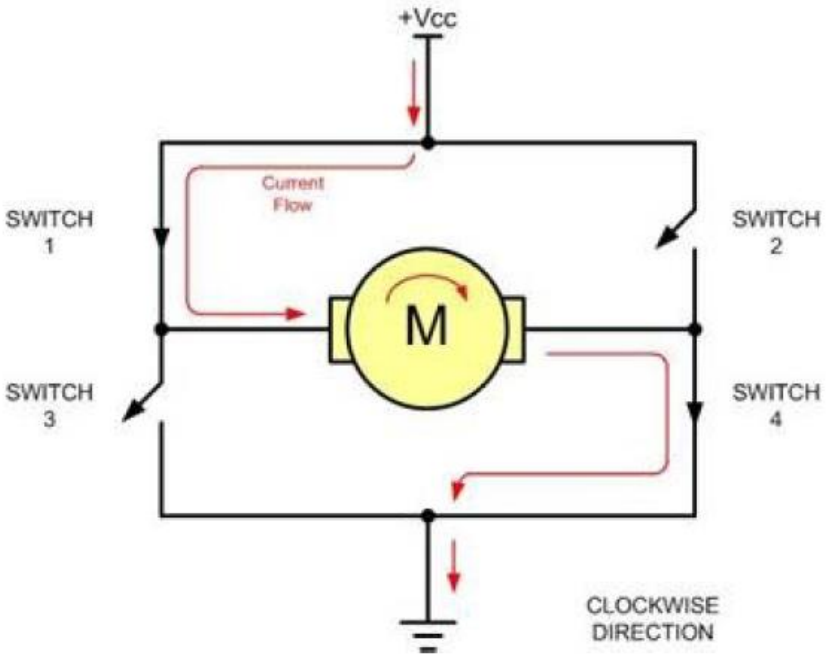
Bidirectional control

With the help of relays, transistor circuit or some specially designed chips (H Bridge) we can change the direction of the DC motor rotation

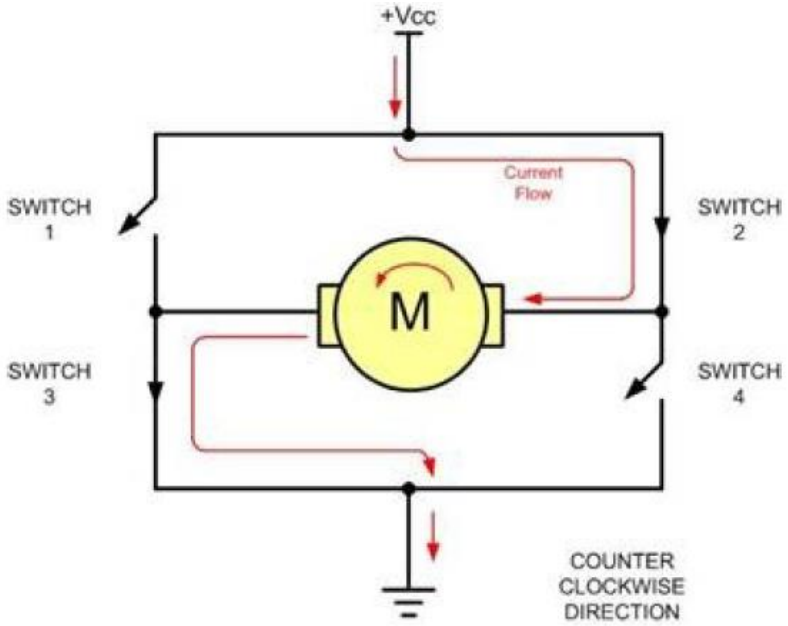


Tecnológico de Monterrey

Bidirectional control



H-Bridge Motor Clockwise Configuration



H-Bridge Motor Counterclockwise Configuration

Tecnológico de Monterrey

Bidirectional control

Motor Operation	SW1	SW2	SW3	SW4
Off	Open	Open	Open	Open
Clockwise	Closed	Open	Open	Closed
Counterclockwise	Open	Closed	Closed	Open
Invalid	Closed	Closed	Closed	Closed

H-Bridge control can be created using relays, transistors, or a single IC solution such as the L298. When using relays and transistors, you must ensure that invalid configurations do not occur.

Example

A switch is connected to pin PTD7. Using relays make the H-Bridge in previous table and write the proper program.

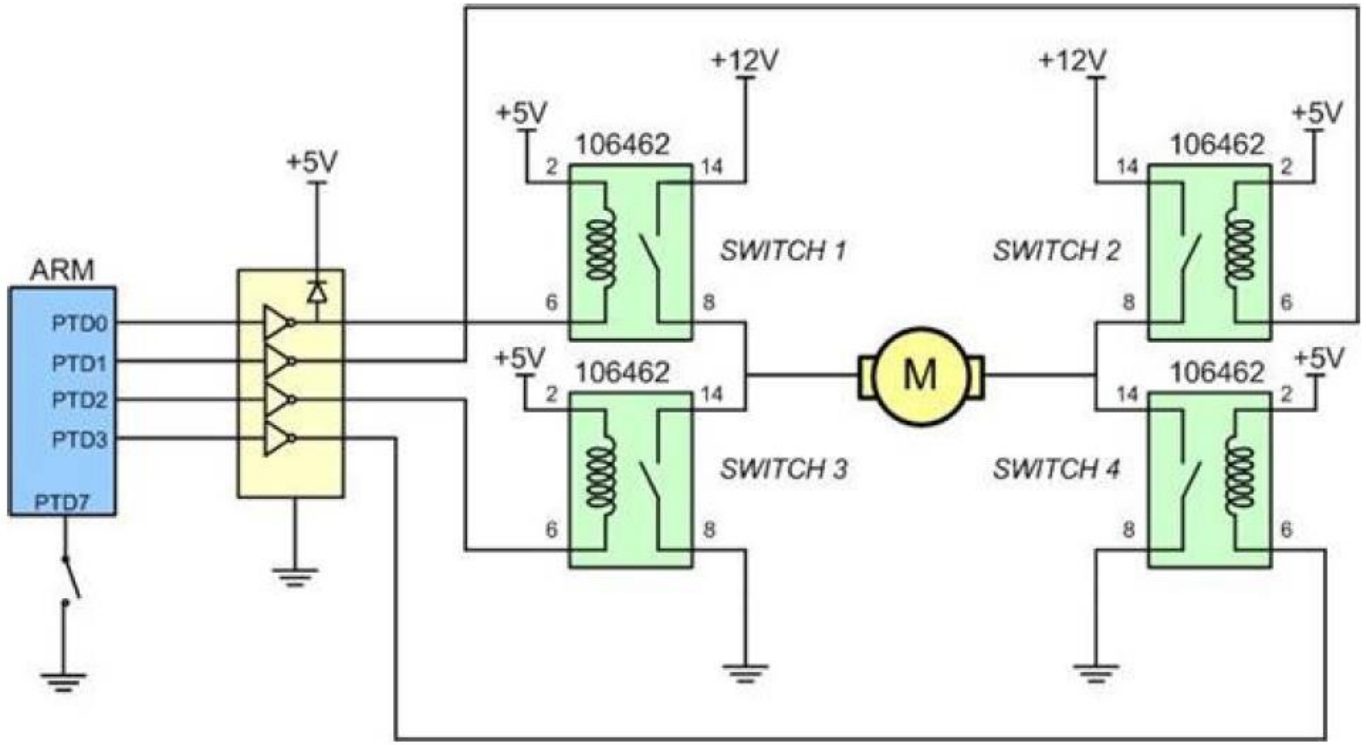
We must perform the following:

(a) If PTD7 = 0, the DC motor moves clockwise.
(b) If PTD7 = 1, the DC motor moves counterclockwise.

Tecnológico de Monterrey

Example 1

Using SPST Relays



Example 1

```
int main (void) {

void delayMs(int n);

PORTD->PCR[0] = 0x100; /* make PTD0 pin as GPIO */
PORTD->PCR[1] = 0x100; /* make PTD1 pin as GPIO */
PORTD->PCR[2] = 0x100; /* make PTD2 pin as GPIO */
PORTD->PCR[3] = 0x100; /* make PTD3 pin as GPIO */
PORTD->PCR[7] = 0x103; /* make PTD7 pin as GPIO and PE */

PTD->PDDR |= 0x0F; /* make PTD0-3 as output pin */
PTD->PDDR &= ~0x80; /* make PTD7 as input pin */
```

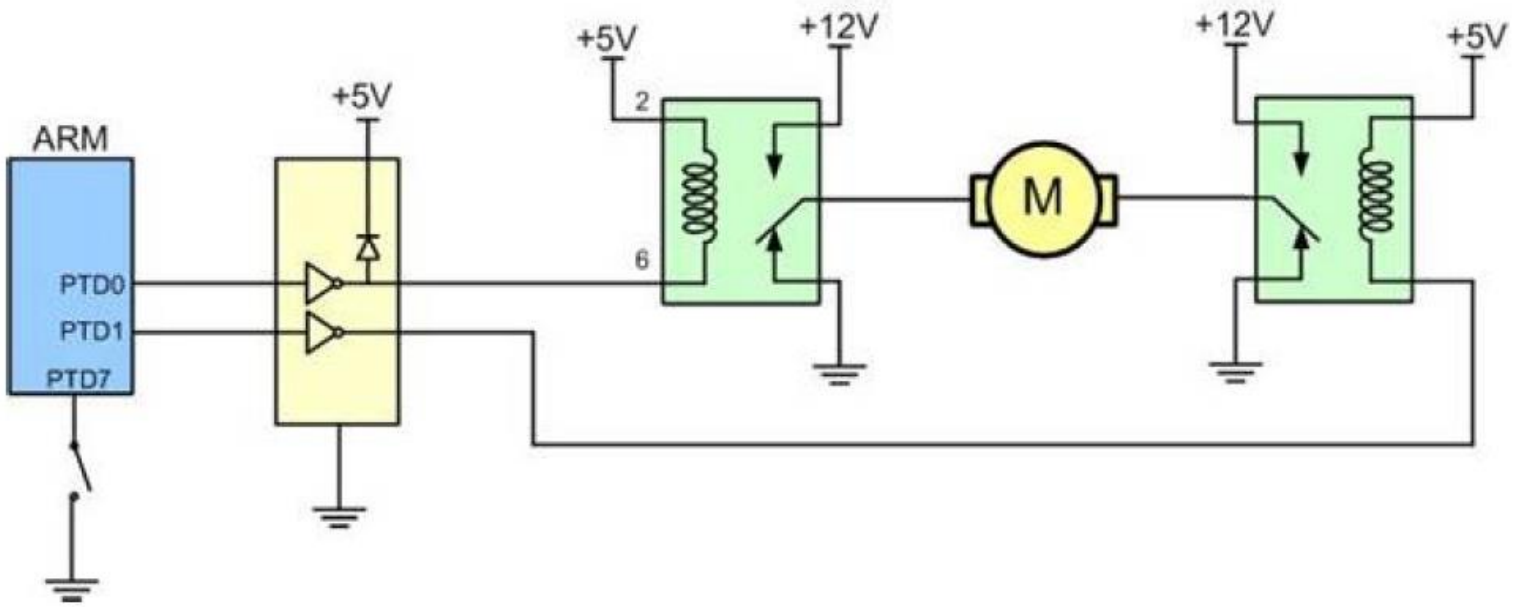
Example 1

```
if((PTD->PDIR & 0x80) == 0) /* PTD7 == 0 */
{
PTD->PDOR &= ~0x0F; /* open all switches */
delayMs(100); /* wait 0.1 second */
PTD->PDOR |= 0x09; /* close SW1 & SW4 */
while((PTD->PDIR & 0x80) == 0) ; /*PTD7 == 0 */ }

else /* PTD7 == 1 */
{
PTD->PDOR &= ~0x0F; /* open all switches */
delayMs(100); /* wait 0.1 second */
PTD->PDOR |= 0x06; /* close SW2 & SW3 */
while((PTD->PDIR & 0x80) != 0) ; /*PTD7 == 0 */ } }
```

Example 2

Using SPDT Relays

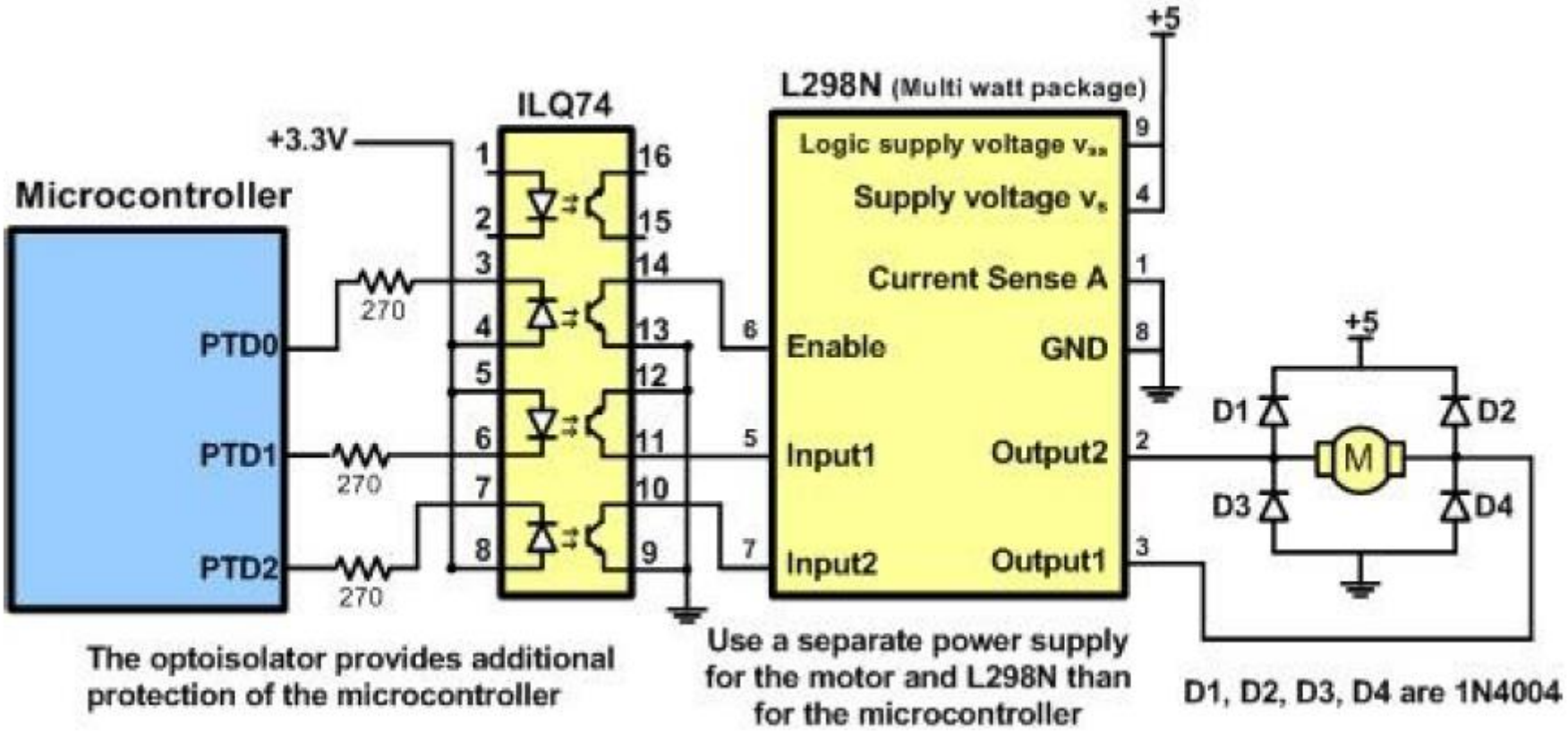


Example 2

```
int main (void) {
PORTD->PCR[0] = 0x100; /* make PTD0 pin as GPIO */
PORTD->PCR[1] = 0x100; /* make PTD1 pin as GPIO */
PORTD->PCR[7] = 0x103; /* make PTD7 pin as GPIO and enable pullup */
PTD->PDDR |= 0x03; /* make PTD0-1 as output pin */
PTD->PDDR &= ~0x80; /* make PTD7 as input pin */
if((PTD->PDIR & 0x80) == 0)
{ /* PTD7 == 0 */
PTD->PDOR &= ~0x02; /* Relay 2 = Off */
PTD->PDOR |= 0x01; /* Relay 1 = On */
}
else
{ /* PTD7 == 1 */
PTD->PDOR &= ~0x01; /* Relay 1 = 0ff */
PTD->PDOR |= 0x02; /* Relay 2 = 0n */ } }
```

Tecnológico de Monterrey

Example 3



Pulse width modulation (PWM)

The speed of the motor depends on three factors: (a) load, (b) voltage, and (c) current.

For a given fixed load we can maintain a steady speed by using a method called pulse width modulation (PWM).

By changing (modulating) the width of the pulse applied to the DC motor we can increase or decrease the amount of power provided to the motor, thereby increasing or decreasing the motor speed.

Notice that, although the voltage has a fixed amplitude, it has a variable duty cycle. That means the wider the pulse, the higher the speed.

Pulse width modulation (PWM)

PWM is so widely used in DC motor control that many microcontrollers come with an on-chip PWM circuitry.

In such microcontrollers all we have to do is load the proper registers with the values of the high and low portions of the desired pulse, and the rest is taken care of by the microcontroller.

This allows the microcontroller to do other things. For microcontrollers without on-chip PWM circuitry, we must create the various duty cycle pulses using software, which prevents the microcontroller from doing other things.

Pulse width modulation (PWM)

The ability to control the speed of the DC motor using PWM is one reason that DC motors are preferable over AC motors.

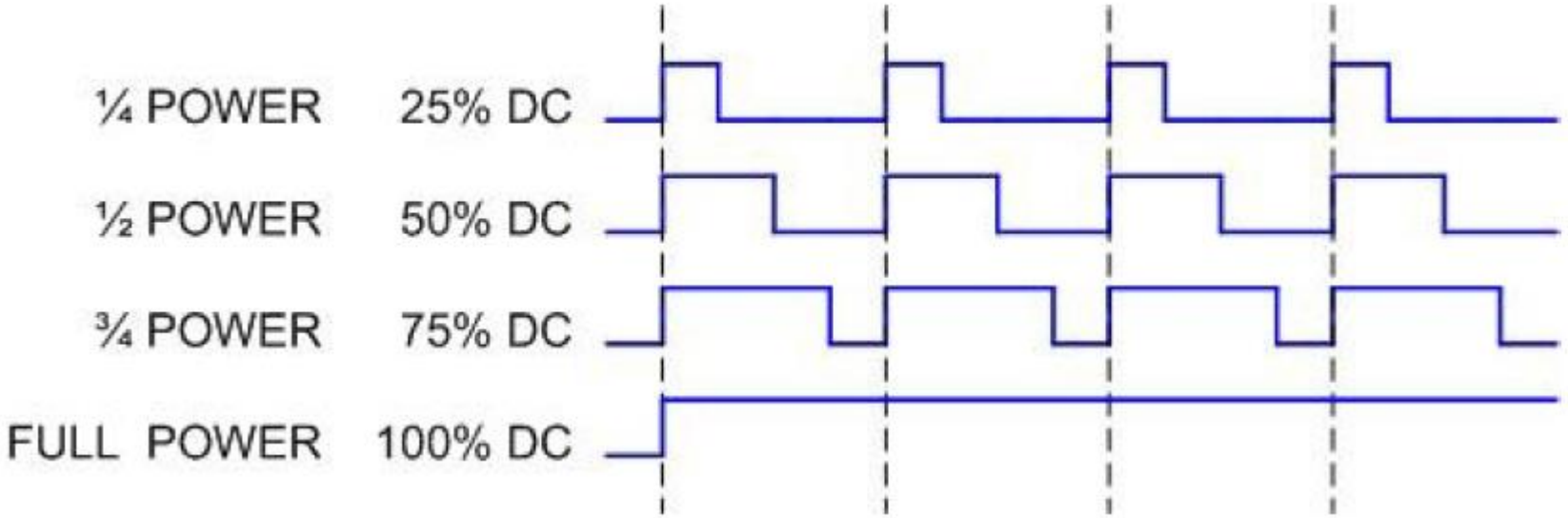
AC motor speed is dictated by the AC frequency of the voltage applied to the motor and the frequency is generally fixed.

As a result, we cannot control the speed of the AC motor when the load is increased.

As will be shown later, we can also change the DC motor's direction and torque.

Tecnológico de Monterrey

Pulse width modulation (PWM)



DC motor control with optoisolator

The optoisolator is indispensable in many motor control applications.

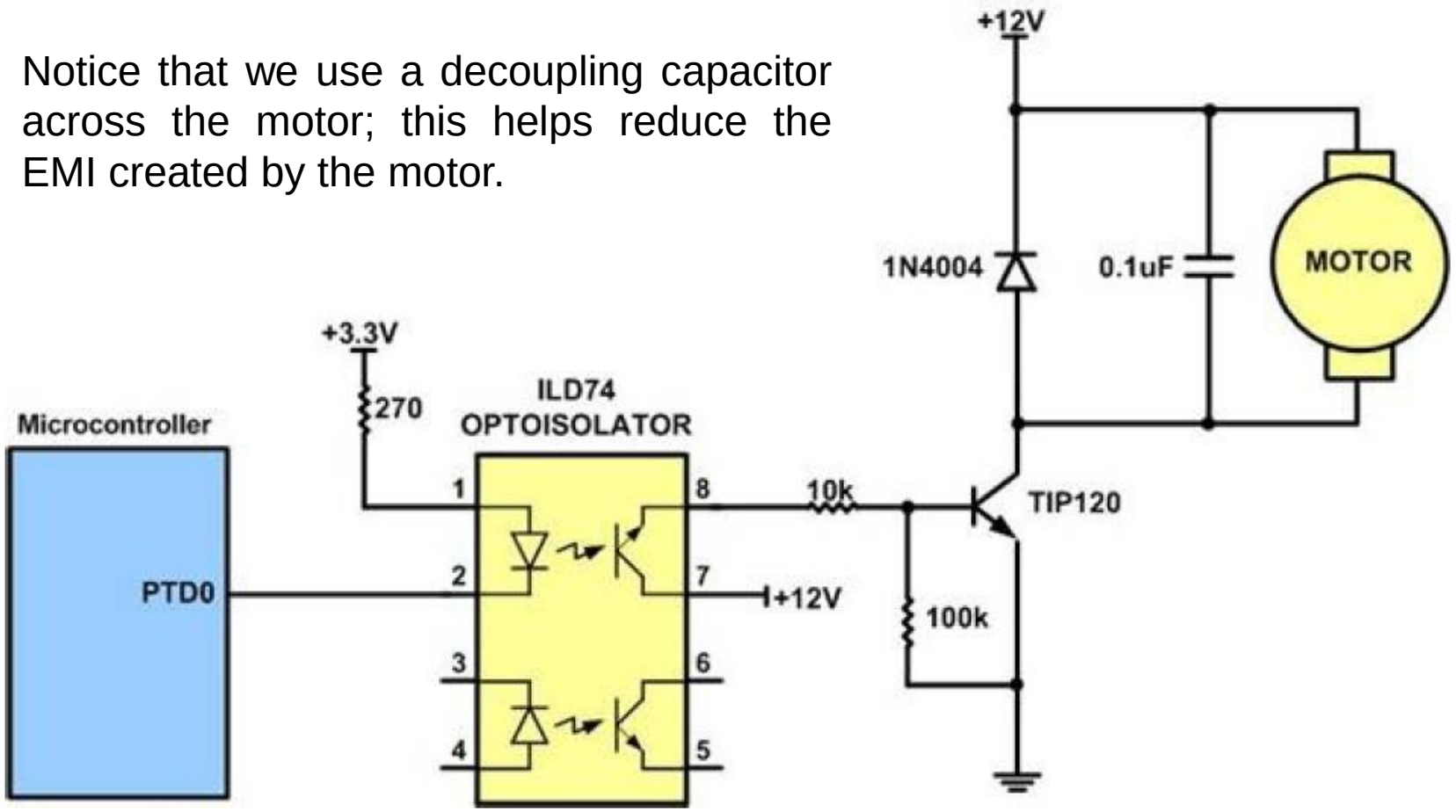
In the next two slides notice that the microcontroller is protected from EMI created by motor brushes by using an optoisolator and a separate power supply.

Separating the power supplies of the motor and logic will reduce the possibility of damage to the control circuit.

The separation of power supplies also allows the use of high-voltage motors.

DC motor control with optoisolator

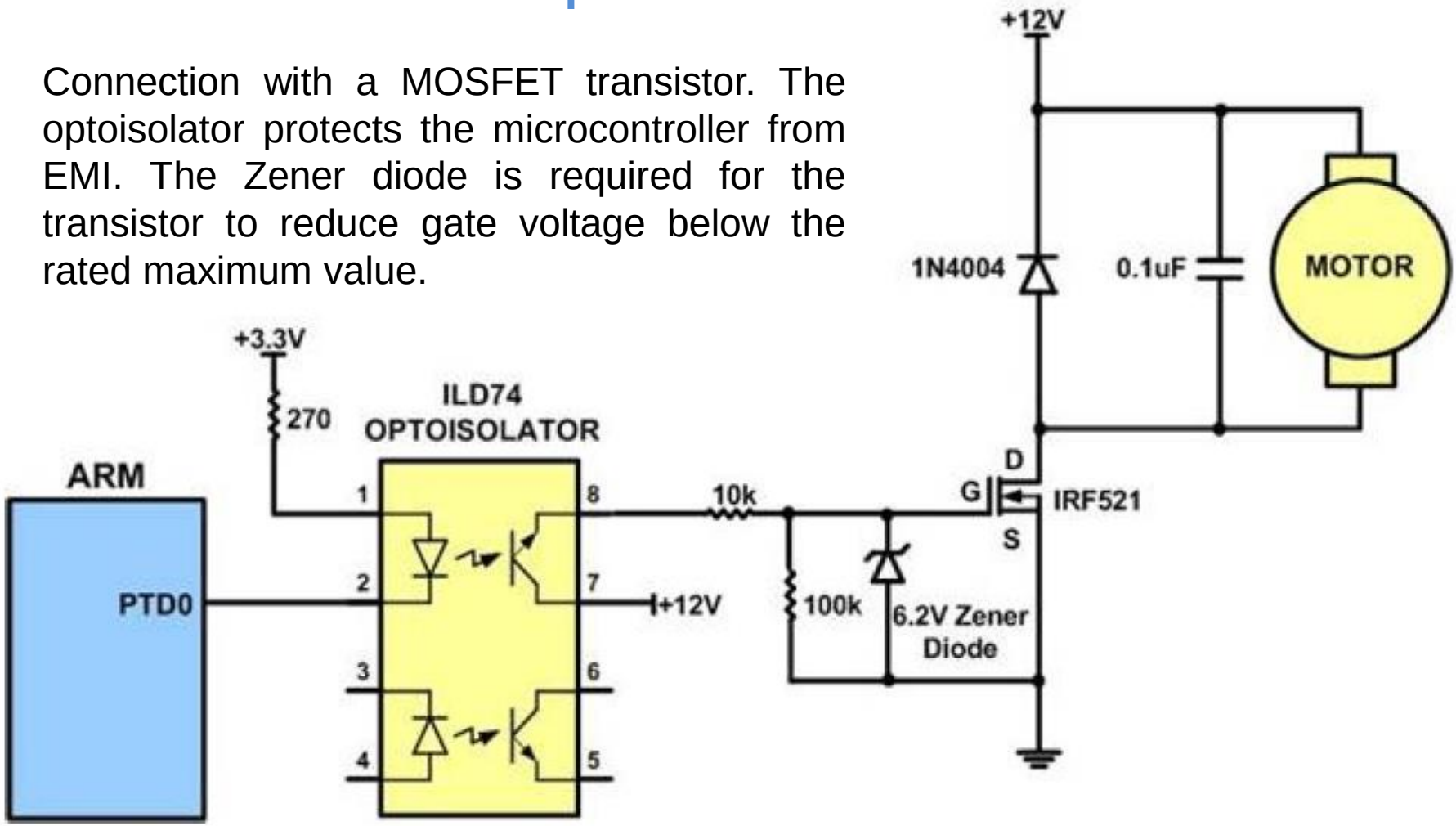
Notice that we use a decoupling capacitor across the motor; this helps reduce the EMI created by the motor.



Tecnológico de Monterrey

DC motor control with optoisolator

Connection with a MOSFET transistor. The optoisolator protects the microcontroller from EMI. The Zener diode is required for the transistor to reduce gate voltage below the rated maximum value.



PWM in Freescale ARM KL25Z

In Freescale ARM KL25Z, the PWM (Pulse Width Modulation) is incorporated into the Timer.

As we saw before, the Timer in KL25Z is called TPM (Timer/PWM Module).

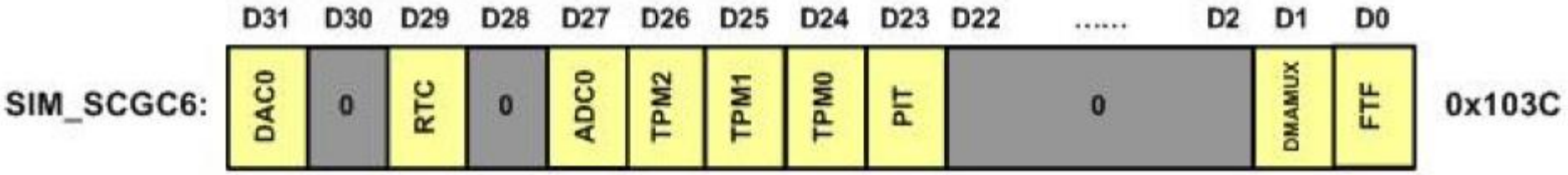
To program the PWM features of the ARM KL25Z chip, we must understand the Timer topics covered in before since PWM is subset of the Timer.

In this section, we examine the PWM features and show how to program them.

Tecnológico de Monterrey

PWM Clock source

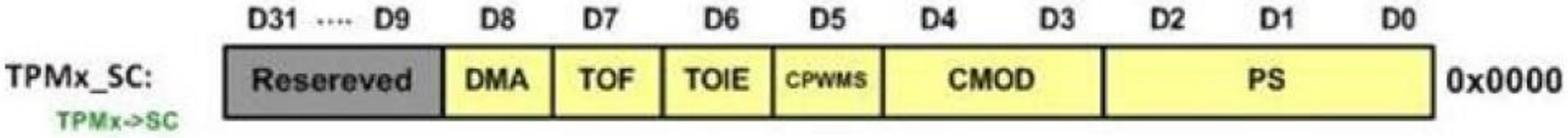
The Clock source to the TPM module is enabled using the SIM_SCGC6 register



bit	Name	Description
24	TPM0	TPM0 clock gate control (0: clock disabled, 1: clock enabled)
25	TPM1	TPM1 clock gate control (0: clock disabled, 1: clock enabled)
26	TPM2	TPM2 clock gate control (0: clock disabled, 1: clock enabled)

CPWMS bit and the TPM counting

As discussed before, the TPMx_SC register has control on the counting of the timer.



Field	Bits	Description
PS	0–2	In the prescaler, the clock is divided by 2^{PS}.

PS value	000	001	010	011	100	101	110	111
Division	1	2	4	8	16	32	64	128

Tecnológico de Monterrey

CPWMS bit and the TPM counting

CMOD	3–4	Clock Mode Selection

CMOD value	Selected clock
00	Timer stopped (No clock selected): In the mode, the TPM_CNT register receives no clock and it is stopped.
01	Timer mode (clock selected at SIM_SOPT2): This mode can be used to generate delays, periodic interrupts, or PWM.
10	Counter mode (clocked by LPTPM_EXTCLK pin): This mode is used to count an external event.
11	Reserved

CPWMS bit and the TPM counting

CPWMS	5	Center-aligned PWM select (0: Up counter mode, 1: up-down counter mode).
TOIE	6	Time Overflow Interrupt Enable (0: Disabled, 1: Enabled).
TOF	7	Timer Overflow Flag
DMA	8	DMA Enable (0: Disabled, 1: Enabled)

The CPWMS bit can be set as up-counter (CPWMS=0) or up-down counter (CPWMS=1).

The counter has two modes:

CPWMS bit and the TPM counting

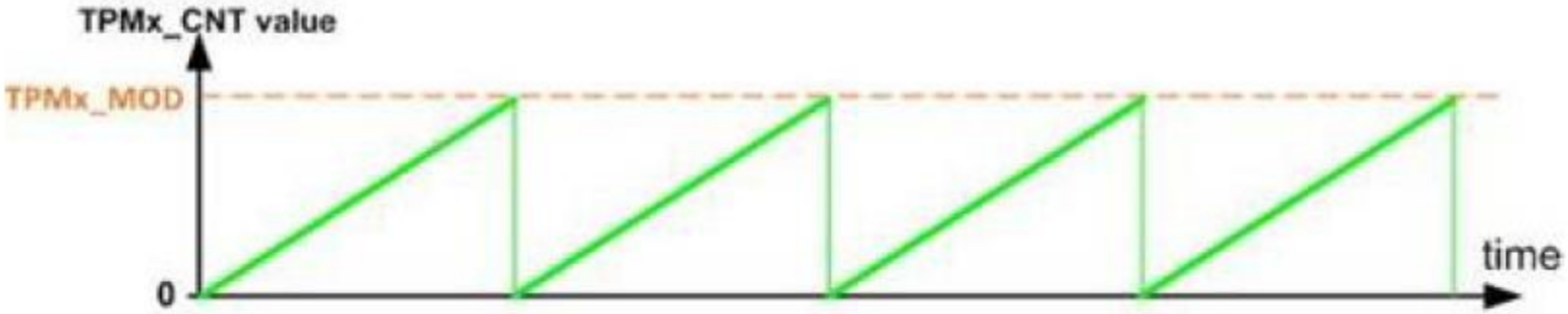
1) ***Count Up:*** The TPMx_CNT counts up from the 0 value until it reaches the value of MOD register.

Upon matching the MOD, the CNT is cleared to zero and count-up starts again. This is the default option for CPWMS=0.

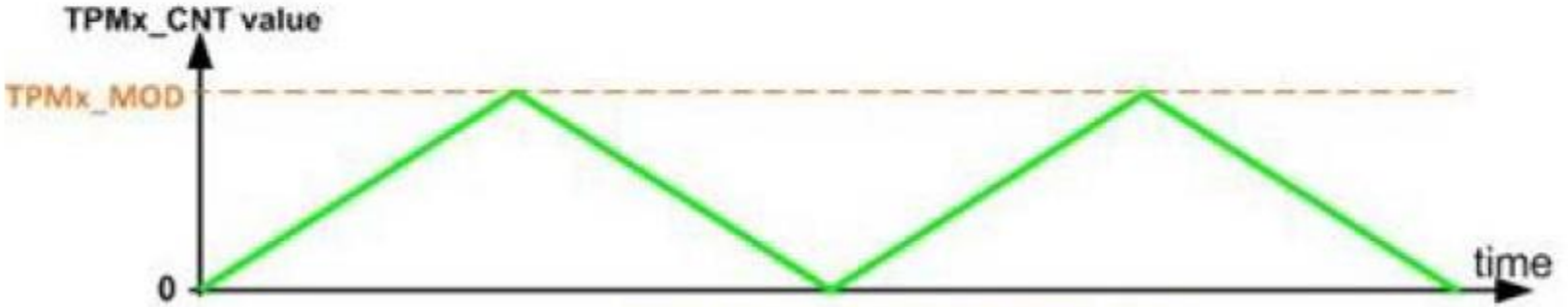
2) ***Count Up-Down:*** counts up from 0 until it reaches the MOD value. After reaching the MOD value, it turns around and counts down to 0.

And upon reaching 0, it repeats the process. We must make CPWMS=1 to get this option.

CPWMS bit and the TPM counting



(a) Up Counting (CPWMS = 0)



(b) Up-down Counting (CPWMS = 1)

CPWMS bit and the TPM counting

We saw that TPMx_CnSC (TPMx Channel n Status and Control) register to program the Output Control or Input Capture features of the KL25Z Timer.

As it was shown, the MSnB:MSnA bits along with the ELSnB:ELSnA bits of TPMx_CnSC register gave us the choices of Input Capture and Output Compare.

For PWM, we can use the options of Center-aligned (up-down counting) or Edge-aligned (up counting).

CPWMS bit and the TPM counting

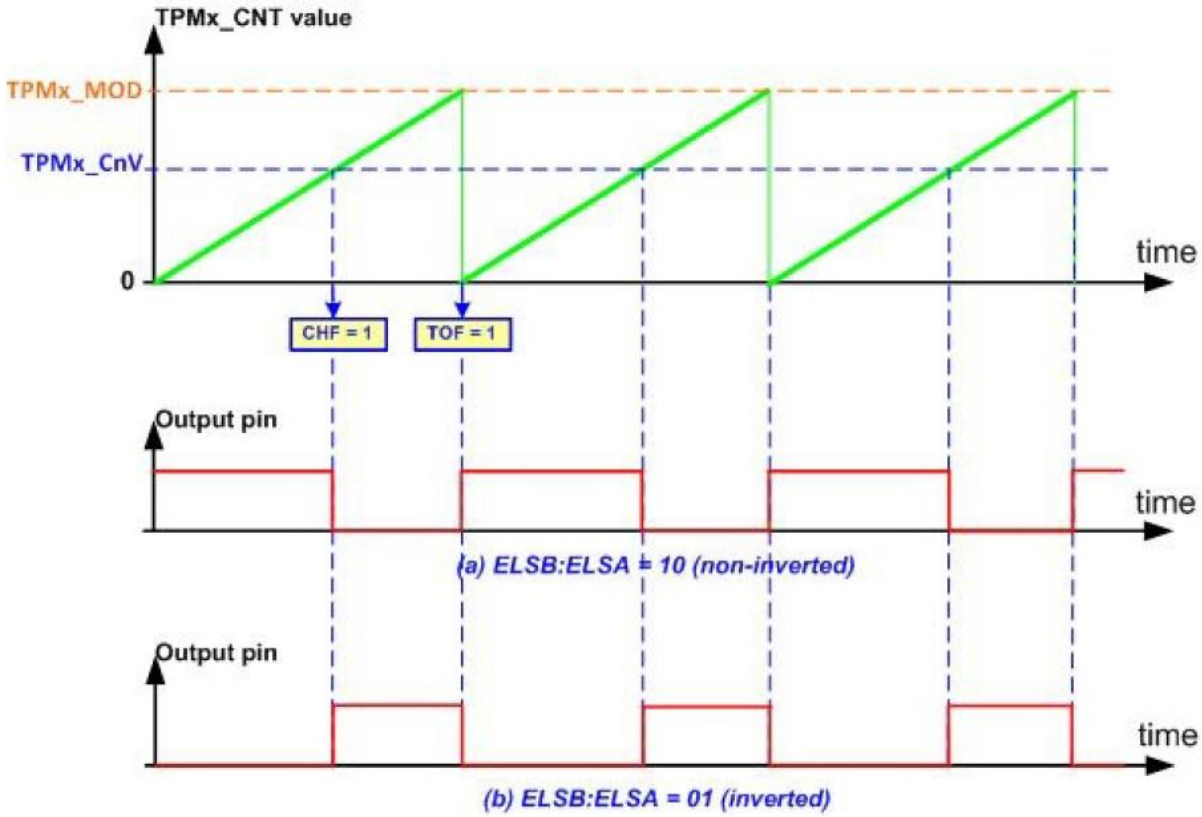
Field	Bit	Description
CHF	7	Channel Flag
CHIE	6	Channel interrupt enable
MSB and MSA	5-4	Channel mode select

D5:D4 (MSB:MSA)	Output mode
00	Channel disabled
01	Output compare
10	PWM
11	Output compare

ELSB and ELSA	3-2	Edge or Level Select
DMA	0	DMA enable (0: Disabled, 1: Enabled)

Edge-Aligned PWM

In the Edge-aligned PWM, the leading edge of the pulse starts at the beginning of the period.



Edge-Aligned PWM

The pulse period is set by the MOD register value (actually MOD+1) and the pulse width value is set by the CnV register.

When ELSnB:ELSnA = 10, it produces high-true pulses. The output is high at the beginning of the pulse when the counter is reloaded and it goes low when the counter value matches CnV register.

When ELSnB:ELSnA = x1, it produces low-true pulses. The output is low at the beginning of the pulse when the counter is reloaded and it goes high when the counter value matches CnV register.

Tecnológico de Monterrey

Edge-Aligned PWM

CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
0	10	10	Edge-Aligned PWM (non-inverted)	Set output on reload, clear output on match
0	10	01 or 11	Edge-Aligned PWM (inverted)	Clear output on reload, set output on match

The PWM output duty cycle and frequency

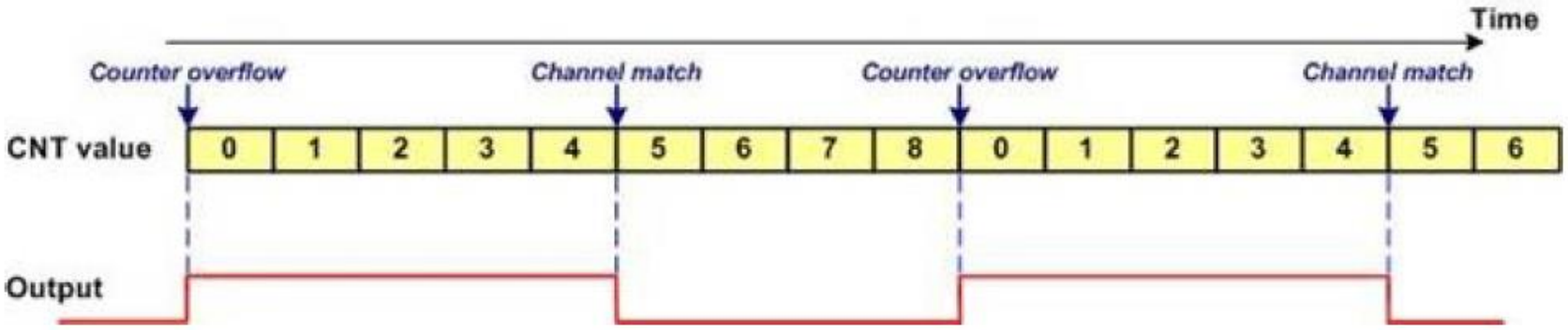
The pulse period is set by the MOD register.

Using the CnV register we set the pulse width (duty cycle) Now, if CnV = 0, then Channel output has 0% duty cycle.

The same way, if CnV greater than or equal to MOD, the duty cycle is 100% since there is never a match.

The next slides shows an example where the output waveform when ELSB:ELSA = 10 (noninverted), MOD = 8, and CnV = 5.

The PWM output duty cycle and frequency



The output is set on counter overflow (reload) and it is cleared on compare match. The CNT is reloaded with 0 after MOD + 1 clocks and the output is set to HIGH for CnV clocks. So, the duty cycle can be calculated using the following formula →

$$\text{Duty Cycle} = \frac{\text{CnV}}{\text{MOD} + 1} \times 100$$

The PWM output duty cycle and frequency

When ELSB:ELSA = 01, the output is inverted and the duty cycle is:

$$\text{Duty Cycle} = 100 - \left(\frac{\text{CnV}}{\text{MOD} + 1} \times 100\right)$$

In edge-aligned PWM mode, the timer counts from 0 to MOD and then rolls over. So, the frequency of the output is 1 / (MOD+1) of the frequency of timer clock. The frequency of the timer clock can be selected using the prescaler. So,

$$F_{generated\ wave} = \frac{F_{timer\ clock} / \text{precaler}}{\text{MOD+1}} = \frac{F_{timer\ clock}}{(\text{MOD} + 1) \times 2^{PS}}$$

Example 1

Find the period (T), frequency (F) and pulse width (DC, duty cycle) of a PWM if TPMx_MOD=999 and TMPx_CnV=250. Assume ELSB:ELSA = 10, no prescaler, and TPMx Module freqs of (a) 8MHz and (b) 1MHz.

Solution:

(a) 1/8MHz = 125ns.
T= (MOD+1) × 125ns = (999+1) x 125ns = 125ms.
Frequency=1/125ms=8000Hz.
The Duty Cycle is [TPMx_CnV/(TPMx_MOD+1)] × 100 = (250/1000) × 100 = 25%.

b) 1/1MHz= 1000ns.
T= (MOD+1) x 1000ns = (999+1) x 1000ns = 1ms.
Frequency = 1/1ms=1000Hz.
The Duty Cycle is [TPMx_CnV/(TPMx_MOD+1)] × 100 = (250/1000) × 100 = 25%.

Example 2

Assume the TPMx Module CF is 8MHz. Find the value of the MOD register for PWM output freq of (a) 5KHz, (b) 10KHz, and (c) 25KHz.

Solution:
The clock period for TPM Module is 1/8MHz=0.125μs (micro second).

(a) The PWM output period is 1/5KHz = 200μs.
Now, TPMx_MOD = (200μs/0.125μs) – 1 = 1600 – 1 = 1599.

(b) The PWM output period is 1/10KHz = 100μs.
Now, TPWM_MOD = (100μs/0.125μs) – 1 = 800 – 1 = 799.

(c) The PWM output period is 1/25KHz = 40μs.
TPMx_MOD = (40μs/0.125μs) – 1 = 320 – 1 = 319.

Example 3

In a given PWM application, we need the PWM output frequency of 60Hz. Using the TPMx Module frequency of 41.98MHz, find out the value of the TPMx_MOD register.

Solution:
TPMx_MOD = (41.98MHz / 60Hz) – 1 = 699,666 – 1 = 699,665. This is not acceptable since it is larger than 65,535, the maximum value the TPMx_MOD register can hold.

Now, 699,666/128 – 1 = 5,465 is acceptable if we use prescaler of 128. The lowest prescaler value we can use is 16 since 699,666/16 – 1 = 43,728. Notice, the prescaler of 8 is not acceptable since 699,666 / 8 – 1 = 87,457.

Configuring GPIO pin for PWM

When using PWM, we must configure the GPIO pins for TPMx output. In this regard, it is same as all other peripherals. The steps are as follows:

1. Enable the clock to GPIO pin.
2. Assign the TPMx signals to specific pins using PORTx_PCRn register.

Configuring PWM generator to create pulses

After the GPIO configuration, we need to take the following steps to configure the PWM:

1. Enable clock to TPMx module in SIM_SCGC6 register

2. Select counter clock source in SIM_SOPT2 register

3. Disable timer while the configuration is being done.

4. Set the mode for Edge-Aligned PWM with TPMx_SC.

5. Load the value into TPMx_MOD register to set the desired output frequency.

6. Load the value into TPMx_CnV register to set the desired duty cycle.

7. Enable timer.

Example 1

```
/* Generate 60Hz 33% PWM output
* TPM0 uses MCGFLLCLK which is 41.94 MHz.

* The prescaler is set to divide by 16.

* The modulo register is set to 43702 and the CnV

* register is set to 14568. See Example 3 for

* the calculations of these values.*/

#include <MKL25Z4.H>
```

```
int main (void) {
SIM->SCGC5 |= 0x1000; /* enable clock to Port D */
PORTD->PCR[1] = 0x0400; /* PTD1 used by TPM0 */
SIM->SCGC6 |= 0x01000000; /* enable clock to TPM0 */
SIM->SOPT2 |= 0x01000000; /* use MCGFLLCLK as timer counter
clock */
TPM0->SC = 0; /* disable timer */
TPM0->CONTROLS[1].CnSC = 0x20 | 0x08;
/* edge-aligned, pulse high */
TPM0->MOD = 43702; /* Set up modulo register for 60 kHz */
TPM0->CONTROLS[1].CnV = 14568;
/* Set up channel value for 33% dutycycle */
TPM0->SC = 0x0C; /* enable TPM0 with prescaler /16 */
while (1) {
}
}
```

Tecnológico de Monterrey

Example2

```
/* Generate 60Hz with varying duty cycle PWM output

*This program is setup identical to the previous. But
* in the infinite loop, the CnV register value is
*  Incremented by 437 (1%) every 20ms.
*Because the LED is low active,
* the longer the duty cycle results in lower light
*intensity.
*/

#include <MKL25Z4.H>
void delayMs(int n);
```

```
int main (void) {
int pulseWidth = 0;
SIM->SCGC5 |= 0x1000; /* enable clock to Port D */
PORTD->PCR[1] = 0x0400; /* PTD1 used by TPM0 */
SIM->SCGC6 |= 0x01000000; /* enable clock to TPM0 */
SIM->SOPT2 |= 0x01000000;
/* use MCGFLLCLK as timer counter clock */
TPM0->SC = 0; /* disable timer */
TPM0->CONTROLS[1].CnSC = 0x20 | 0x08; /* edge-aligned,
pulse high */
TPM0->MOD = 43702; /* Set up modulo register for 60 kHz
*/
TPM0->CONTROLS[1].CnV = 14568; /* Set up channel value
for 33% dutycycle */
TPM0->SC = 0x0C; /* enable TPM0 with prescaler /16 */
```

```
while (1) {
pulseWidth += 437;
if (pulseWidth > 43702)
pulseWidth = 0;
TPM0->CONTROLS[1].CnV = pulseWidth;
delayMs(20);
}
}
/* Delay n milliseconds
* The CPU core clock is set to MCGFLLCLK at 41.94 MHz
in SystemInit().
*/
void delayMs(int n) {
int i;
int j;
for(i = 0 ; i < n; i++)
for (j = 0; j < 7000; j++) {}
}
```

Center-Aligned PWM

If we set the CPWMS bit in TPMx_SC register to High, then the output is Center-Aligned PWM.

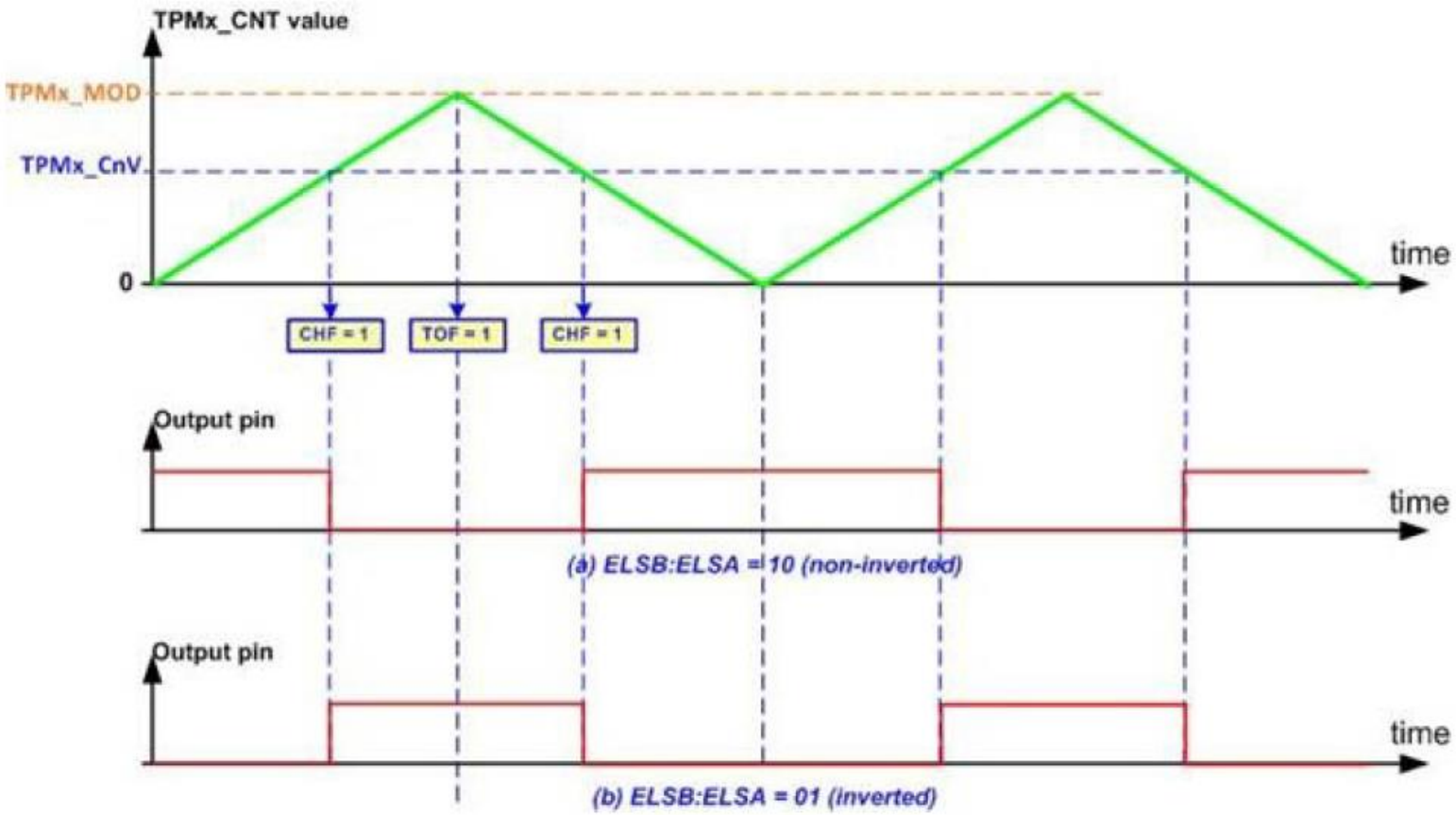
The counter will count up from 0 to the value in MOD register then turn around and count down to 0.

That means, the period of the pulse is 2 × MOD. The same way, the pulse width = 2 × CnV.

At the same time whenever the CnV=MOD, the output pin is forced High or Low depending on the ELSnB:ELSnA bits and whether the counter is counting up or down.

Tecnológico de Monterrey

Center-Aligned PWM



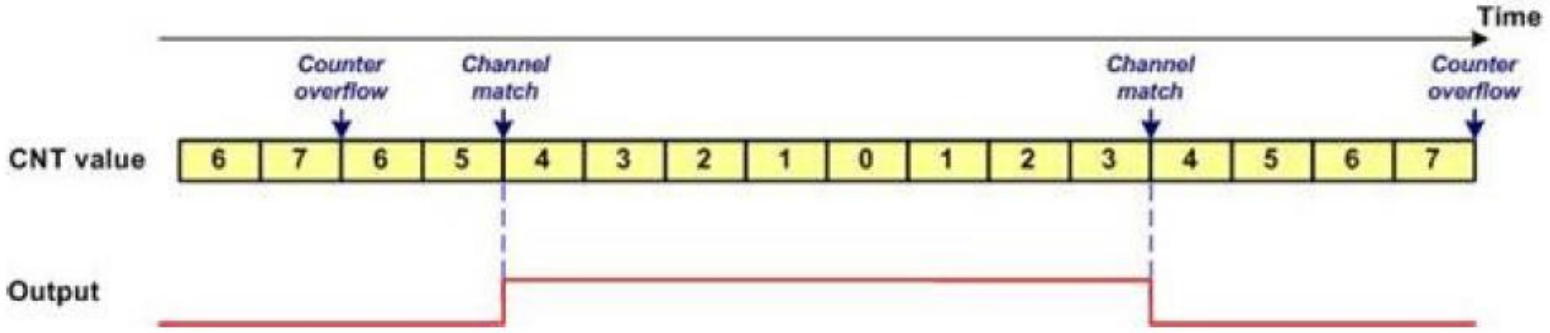
Center-Aligned PWM

CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
1	10	10	Center-Aligned PWM	Clear output on match-up, set output on match-down
1	10	X1	Center-Aligned PWM	Set output on match-up, clear output on match-down

Center-Aligned PWM (notice CPWMMS=1)

The PWM output duty cycle and frequency

The figure shows the output when MOD = 7 and CnV = 4. The output is set on compare match when counting down, and is cleared on compare match when counting up.



The output is HIGH for CnV×2 clocks and each cycle takes MOD × 2 clocks. As a result, the duty cycle is:

$$\text{Duty Cycle} = \frac{\text{CnV} \times 2}{\text{MOD} \times 2} \times 100 = \frac{\text{CnV}}{\text{MOD}} \times 100$$

The PWM output duty cycle and frequency

When ELSB:ELSA = 01, the output is inverted and the duty cycle is:

$$\text{Duty Cycle} = 100 - \left(\frac{\text{CnV}}{\text{MOD}} \times 100\right)$$

The frequency of the generated wave is:

$$F_{generated\ wave} = \frac{F_{timer\ clock} / \text{precaler}}{\text{MOD} \times 2} = \frac{F_{timer\ clock}}{\text{MOD} \times 2^{PS+1}}$$

Example

Find the period (T), frequency (F) and pulse width (DC, duty cycle) of a PWM if TPMx_MOD=400 and TMPx_CnV=250. Assume ELSB:ELSA = 10 (non-inverted), no prescaler, and TPWx Module clock frequencies of (a) 8MHz, (b) 2MHz

Solution:

(a) 1/8MHz = 125ns. Now T = MOD × 2 × 125ns = 400 × 2 × 125ns = 100µs.
Frequency = 1 / 100µs = 10 kHz.
Duty Cycle is (TPMx_CnV / TPMx_MOD) × 100 = (250 / 400) × 100 = 62.5%.

(b) 1/2MHz= 500ns. Now T=400 × 2 × 500ns = 400µs.
Frequency = 1 / 400µs = 2500Hz.
Duty Cycle is (TPMx_CnV / TPMx_MOD) × 100 = (250/400) × 100 = 62.5%.

```
/* Generate 30Hz 40% center-aligned PWM

* TPM0 uses MCGFLLCLK which is 41.94 MHz.

* The prescaler is set to divide by 16.

* The modulo register is set to 43703 and the CnV
* register is set to 17481. TPM0 channel 1 is
* configured to be center-aligned pulse high.

*/

#include <MKL25Z4.H>
```

```
int main (void) {
SIM->SCGC5 |= 0x1000; /* enable clock to Port D */
PORTD->PCR[1] = 0x0400; /* PTD1 used by TPM0 */
SIM->SCGC6 |= 0x01000000; /* enable clock to TPM0 */
SIM->SOPT2 |= 0x01000000;
/* use MCGFLLCLK as timer counter clock */
TPM0->SC = 0; /* disable timer */
TPM0->CONTROLS[1].CnSC = 0x20 | 0x08;
/* center-aligned, pulse high */
TPM0->MOD = 43703; /* Set up modulo register for 1 kHz */
TPM0->CONTROLS[1].CnV = 17481;
/* Set up channel value for 40% duty cycle */
TPM0->SC = 0x0C | 0x20; /* enable TPM0 with prescaler /16,
center-aligned
*/
while (1) { } }
```

Edge-aligned vs. center-aligned mode

See figures in next slides. In both figures the bold vertical blue lines are repeated periodically.

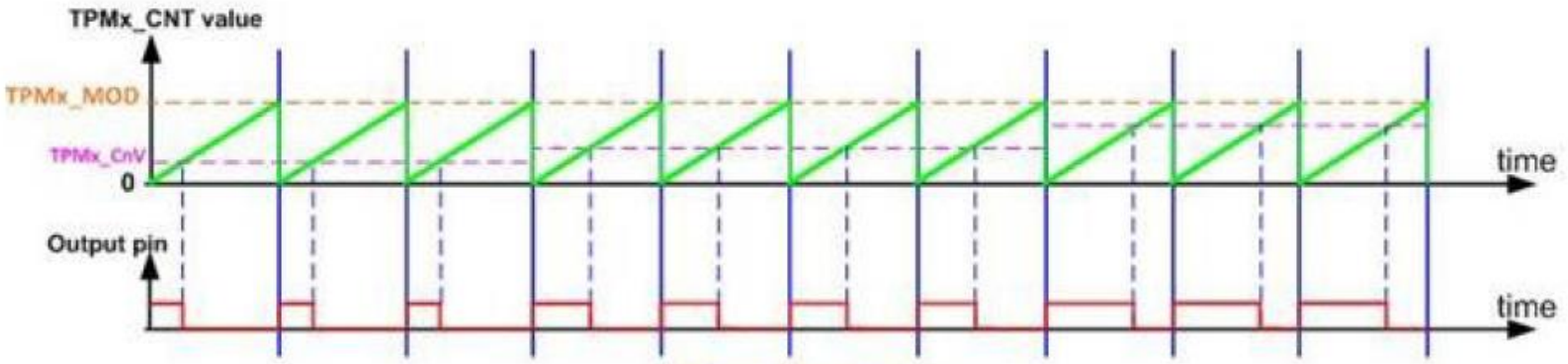
In the edge-aligned mode, the left edge of the pulse is always on the bold blue line while in center-aligned mode, the center of the pulse is always fixed on the bold line.

In other words, in edge-aligned mode, the phase of the wave is different for different duty cycles, while it remains unchanged in the center-aligned mode.

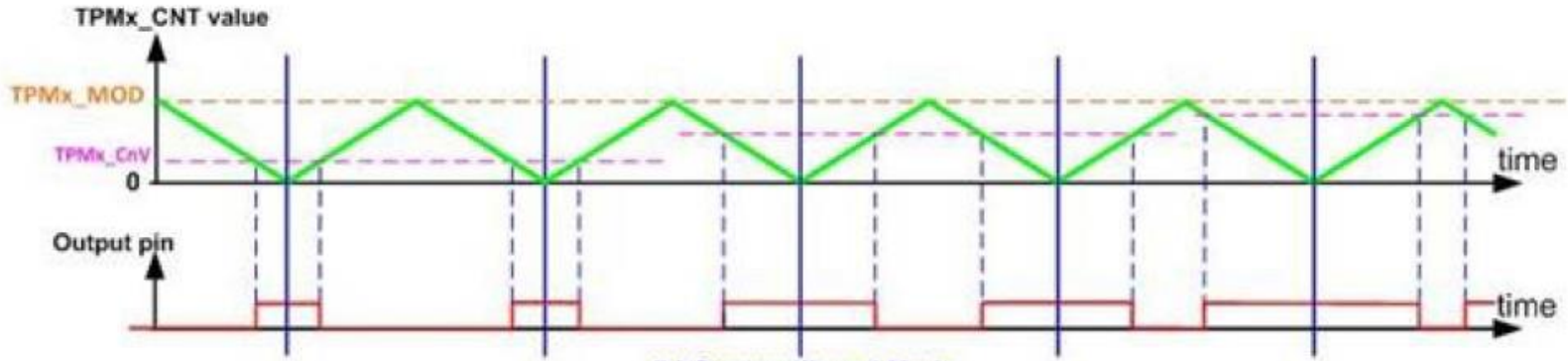
For driving motors, it is preferable to use center-aligned rather than edge-aligned.

Tecnológico de Monterrey

Edge-aligned vs. center-aligned mode



(a) Edge-aligned Mode



(b) Center-aligned Mode

Dead-band generation (Case Study)

One application of center-aligned PWM is to generate outputs with deadband.

Review example in slide 15, when we switched the direction of the H-bridge circuit, we opened all switches and delayed for a period of time.

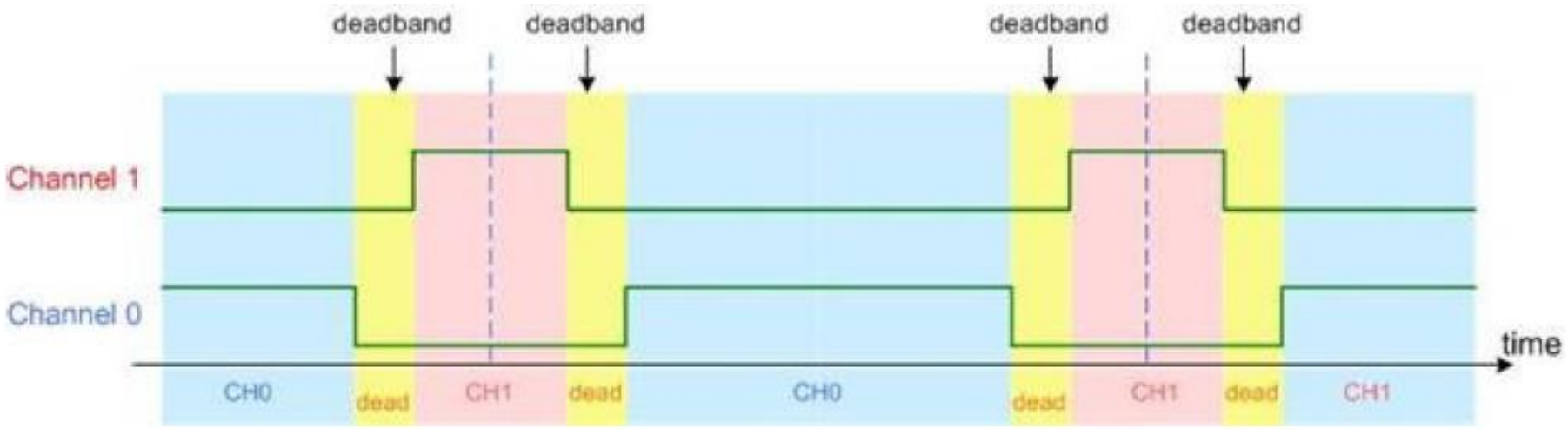
That was deadband, a period of time when all switches are open to avoid the possibility of overlapping time →

“hen both switches on the same leg of the H-bridge are one which may cause a short circuit.

Dead-band generation (Case Study)

Transistors are faster to turn on and slower to turn off. If we turn one on and the other off, there will be a short time that both transistors are on.

To generate deadband, we use two center-aligned channels (one with positive pulse and the other negative pulse.



Dead-band generation (Case Study)

Assuming the circuit is active high

Now we have one channel centered at the time when the timer counter reaches the value in MOD register

And the other channel centered at the time when the timer counter reaches 0 so they will be 180 degree out of phase.

For each channel we program them to have less than 50% duty cycle therefore dead-bands are created between the two channels.

Tecnológico de Monterrey

Dead-band generation (Case Study)

```
/* Deadband generation with center-aligned PWM

* TPM0 uses MCGFLLCLK which is 41.94 MHz. The prescaler
* is set to divide by 16. The modulo register is set to
* 43703.
*
 The timer is configured for center-aligned PWM.

* channel 0 is configured for 60% duty cycle pulse low.

* channel 1 is configured for 40% duty cycle pulse high.

* This creates a 10% deadband between channel 0 high and
* channel 1 high.
*/
#include <MKL25Z4.H>
```

Dead-band generation (Case Study)

```
int main (void) {

SIM->SCGC5 |= 0x1000; /* enable clock to Port D */
PORTD->PCR[0] = 0x0400; /* PTD0 used by TPM0 */
PORTD->PCR[1] = 0x0400; /* PTD1 used by TPM0 */
SIM->SCGC6 |= 0x01000000; /* enable clock to TPM0 */
SIM->SOPT2 |= 0x01000000;

/* use MCGFLLCLK as timer counter clock */
TPM0->SC = 0; /* disable timer */
TPM0->CONTROLS[0].CnSC = 0x20 | 0x04;
/* center-aligned, pulse low */
TPM0->CONTROLS[1].CnSC = 0x20 | 0x08;
/* center-aligned, pulse high */
```

Dead-band generation (Case Study)

```
TPM0->MOD = 43703;
/* Set up modulo register for 30Hz */

TPM0->CONTROLS[0].CnV = 26221;
/* Set up channel value for 60% duty cycle */
TPM0->CONTROLS[1].CnV = 17481;
/* Set up channel value for 40% duty cycle*/
TPM0->SC = 0x0C | 0x20;
/* enable TPM0 with prescaler /16, center-aligned */

while (1) {
}
}
```