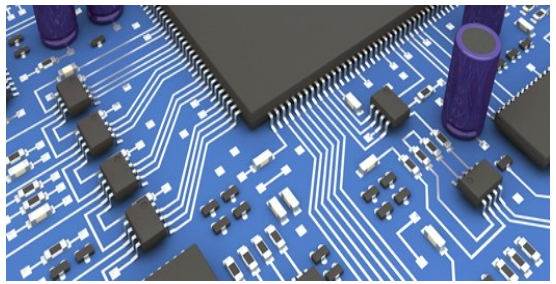
Aplikace Embedded systémů v Mechatronice



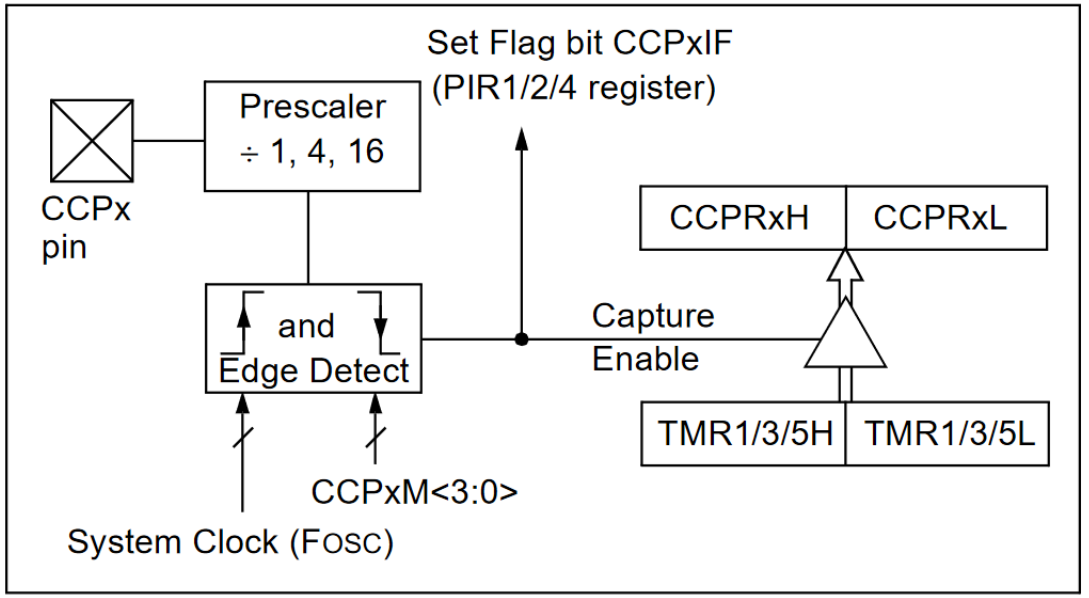
MPLAB® X IDE

Michal Bastl

Capture

Capture:

- Periferie spolupracuje s Timerem
- Při detekci hrany na pinu (dle nastaveni) dojde k přesunu stavu TMR registru
- Tuto hodnotu mohu použít např. k určení periody těchto událostí
- Periferie produkuje i interrupt

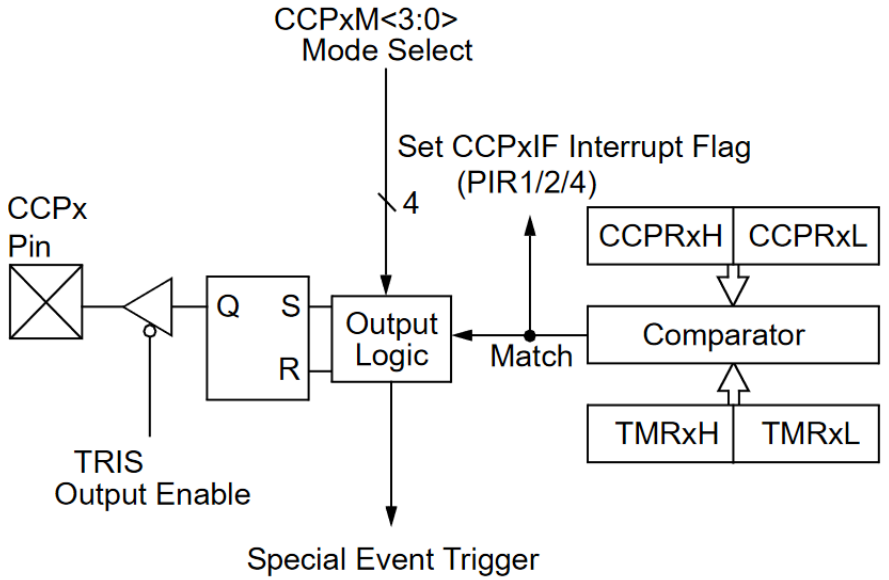


- Lze zachytit nástupní, nebo sestupnou hranu
- Lze nakonfigurovat před-děličku

Compare

Compare:

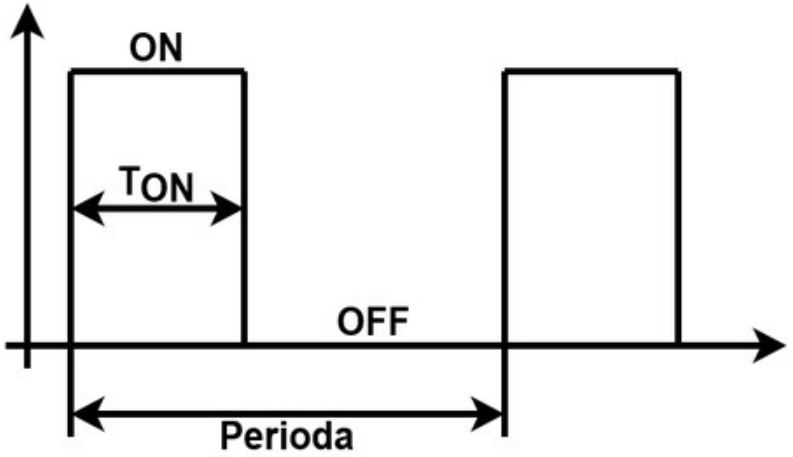
- Tato periferie spolupracuje s Timerem
- Pokud se hodnota v TMR registru rovná s registrem periferie je vyvolána událost
- Převrácení
- Pin logická 0
- Pin logická 1
- Periferie může produkovat interrupt



PWM

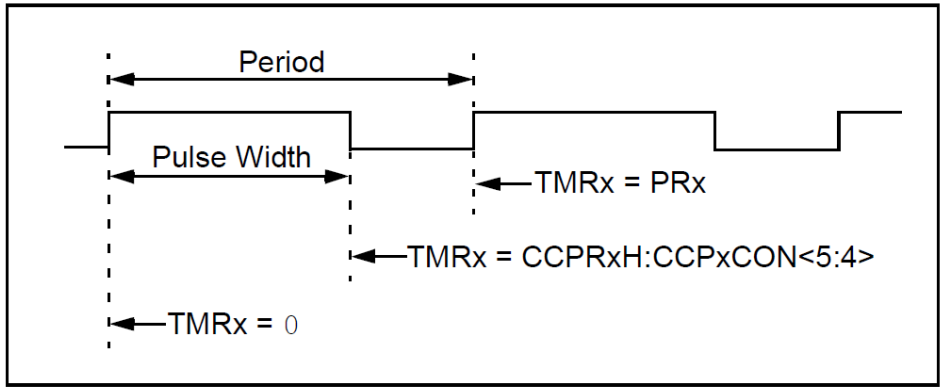
Důležité parametry PWM:

- Frekvence-jedná se o frekvenci celého cyklu, tak je většinou fixní.
- V praxi se používají různé frekvence.
- V elektrických pohonech se nejčastěji setkáváme s frekvencemi 10-20 kHz.
- Jinde to může být různé-v audio aplikacích více jak 100 kHz.
- V topných systémech i jednotky Hz.
- Dalším parametrem je střída (duty cycle).
- Je to poměr zapnutého času k celkové periodě PWM.
- Právě střída se u PWM nastavuje a tím se reguluje výkon.
- Střída tedy může nabývat hodnot 0-1, respektive 0-100%.
- Střída 1 tedy znamená plný výkon, 0,5 pak poloviční.



$$DUTY = \frac{T_{ON}}{PERIOD}$$

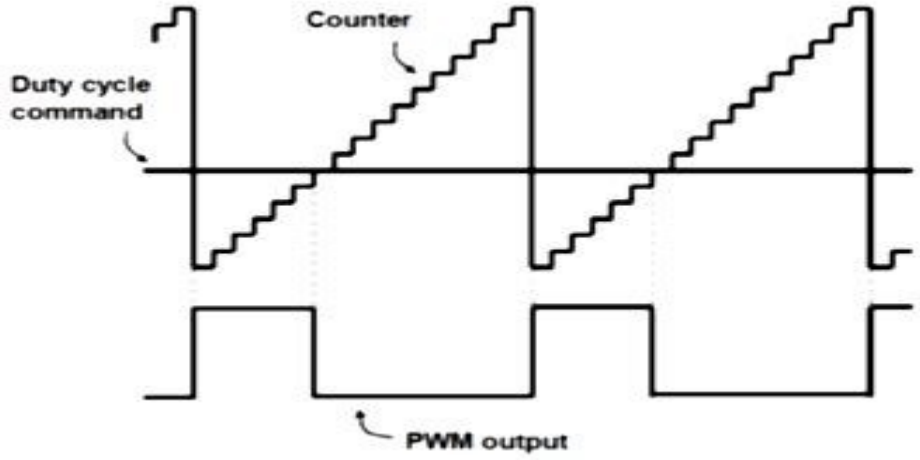
FIGURE 14-3: CCP PWM OUTPUT SIGNAL

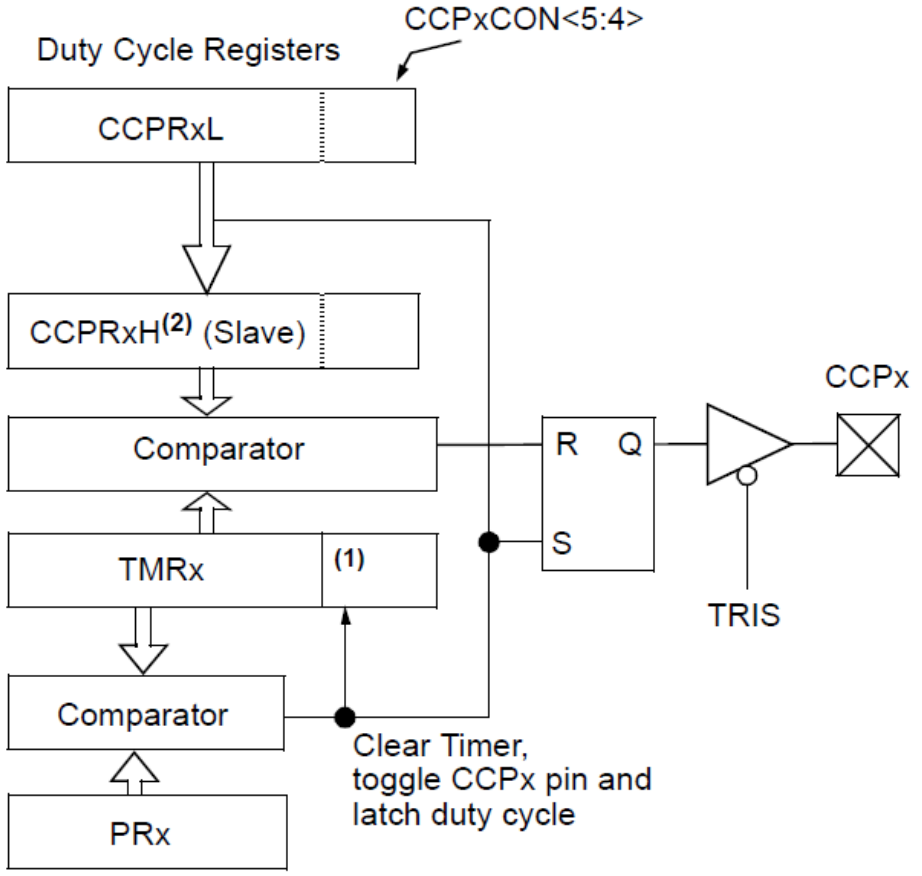


PWM PIC18

Generátor PWM lze digitálně realizovat následovně:

- Mám čítač impulsů, což může být například TIMER na PIC18.
- Mám zdroj hodinových pulzů.
- Čítač načítá až do své maximální hodnota poté přeteče.
- Tento signál by měl v podstatě trojúhelníkový průběh.
- Pomocí binárního komparátoru mohu porovnávat aktuální hodnotu čítače s dalším registrem.
- Výsledek komparace zobrazím na pin kontroléru.
- Pokud mám 8-bit čítač, který načítá až do 255 a nastavím hodnotu v registru 128 mám právě poloviční střídu na pinu.





EQUATION 14-1: PWM PERIOD

$$PWM\ Period = [(PRx) + 1] \bullet 4 \bullet TOSC \bullet (TMRx\ Prescale\ Value)$$

Note 1: Tosc = 1/Fosc

PWM PIC18 postup

1)Nastavení pinů jako input
2)Zvolení Timeru CxTSEL v CCPTMRSx
3)Nastavení period registru Prx
4)Nastaveni CCPxCON podle modu který chci
5)Vynulovaní TMRxIF
6)Zapnuti timeru
7)Počkám na první přetečení pomoci while
8)V PSTRxCON mohu nastavit steering (zapnu stejnou PWM na různé piny)
9)Nastavim piny jako output

```
//init - PWM
TRISDbits.RD5 = 1;              // vypnu pin P1B
TRISCbits.RC2 = 1;              // vypnu pin P1A
CCPTMRS0bits.C1TSEL = 0b00;     // Timer 2
PR2 = 199;                      // f = 10kHz
CCP1CONbits.P1M = 0b00;         // PWM single
CCP1CONbits.CCP1M = 0b1100;     // PWM single
CCPR1L = 0;                     // strida 0%
TMR2IF = 0;                     // az pretece timer
TMR2ON = 1;                     // staci zapnout defaultne
while(!TMR2IF){};               // cekam az jednou pretece
PSTR1CON |= 0b11;               // P1B a P1A

TRISDbits.RD5 = 0;              // zapnu pin P1B
TRISCbits.RC2 = 0;              // zapnu pin P1A
```

PWM PIC18 registry

REGISTER 14-2: CCPxCON: ENHANCED CCPx CONTROL REGISTER

R/x-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PxM<1:0>		DCxB<1:0>		CCPxM<3:0>			
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Reset
'1' = Bit is set	'0' = Bit is cleared	

bit 7-6 **PxM<1:0>**: Enhanced PWM Output Configuration bits
If CCPxM<3:2> = 00, 01, 10: (Capture/Compare modes)
xx = PxA assigned as Capture/Compare input; PxB, PxC, PxD assigned as port pins

Half-Bridge ECCP Modules(1):
If CCPxM<3:2> = 11: (PWM modes)
0x = Single output; PxA modulated; PxB assigned as port pin
1x = Half-Bridge output; PxA, PxB modulated with dead-band control

Full-Bridge ECCP Modules(1):
If CCPxM<3:2> = 11: (PWM modes)
00 = Single output; PxA modulated; PxB, PxC, PxD assigned as port pins
01 = Full-Bridge output forward; PxD modulated; PxA active; PxB, PxC inactive
10 = Half-Bridge output; PxA, PxB modulated with dead-band control; PxC, PxD assigned as port pins
11 = Full-Bridge output reverse; PxB modulated; PxC active; PxA, PxD inactive

bit 5-4 **DCxB<1:0>:** PWM Duty Cycle Least Significant bits
Capture mode:
Unused
Compare mode:
Unused
PWM mode:
These bits are the two LSbs of the PWM duty cycle. The eight MSbs are found in CCPRxL.

Note 1: See Table 14-1 to determine full-bridge and half-bridge ECCPs for the device being used.

```
//init – PWM
PSTR1CON = 0b11;                    //P1A PWM
CCPTMRS0bits.C1TSEL = 0b00;         //timer2 will be used
PR2 = 200;                          //period
CCP1CON = 0b00001100;               //enable PWM
CCPR1L = 200;                       //duty cycle

//init - timer2
T2CON = 0b00111101;
```

PWM PIC18 registry

REGISTER 14-3: CCPTMRS0: PWM TIMER SELECTION CONTROL REGISTER 0

R/W-0	R/W-0	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
C3TSEL<1:0>		—	C2TSEL<1:0>		—	C1TSEL<1:0>	
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7-6 **C3TSEL<1:0>:** CCP3 Timer Selection bits
00 = CCP3 – Capture/Compare modes use Timer1, PWM modes use Timer2
01 = CCP3 – Capture/Compare modes use Timer3, PWM modes use Timer4
10 = CCP3 – Capture/Compare modes use Timer5, PWM modes use Timer6
11 = Reserved

bit 5 **Unused**

bit 4-3 **C2TSEL<1:0>:** CCP2 Timer Selection bits
00 = CCP2 – Capture/Compare modes use Timer1, PWM modes use Timer2
01 = CCP2 – Capture/Compare modes use Timer3, PWM modes use Timer4
10 = CCP2 – Capture/Compare modes use Timer5, PWM modes use Timer6
11 = Reserved

bit 2 **Unused**

bit 1-0 **C1TSEL<1:0>:** CCP1 Timer Selection bits
00 = CCP1 – Capture/Compare modes use Timer1, PWM modes use Timer2
01 = CCP1 – Capture/Compare modes use Timer3, PWM modes use Timer4
10 = CCP1 – Capture/Compare modes use Timer5, PWM modes use Timer6
11 = Reserved

```
//init – PWM
PSTR1CON = 0b11;                    //P1A PWM
CCPTMRS0bits.C1TSEL = 0b00;         //timer2 will be used
PR2 = 200;                          //period
CCP1CON = 0b00001100;               //enable PWM
CCPR1L = 200;                       //duty cycle

//init - timer2
T2CON = 0b00111101;
```

PWM PIC18 registry

REGISTER 14-7: PSTRxCON: PWM STEERING CONTROL REGISTER(1)

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1
—	—	—	STRxSYNC	STRxD	STRxC	STRxB	STRxA
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7-5 **Unimplemented:** Read as '0'

bit 4 **STRxSYNC:** Steering Sync bit
1 = Output steering update occurs on next PWM period
0 = Output steering update occurs at the beginning of the instruction cycle boundary

bit 3 **STRxD:** Steering Enable bit D
1 = PxD pin has the PWM waveform with polarity control from CCPxM<1:0>
0 = PxD pin is assigned to port pin

bit 2 **STRxC:** Steering Enable bit C
1 = PxC pin has the PWM waveform with polarity control from CCPxM<1:0>
0 = PxC pin is assigned to port pin

bit 1 **STRxB:** Steering Enable bit B
1 = PxB pin has the PWM waveform with polarity control from CCPxM<1:0>
0 = PxB pin is assigned to port pin

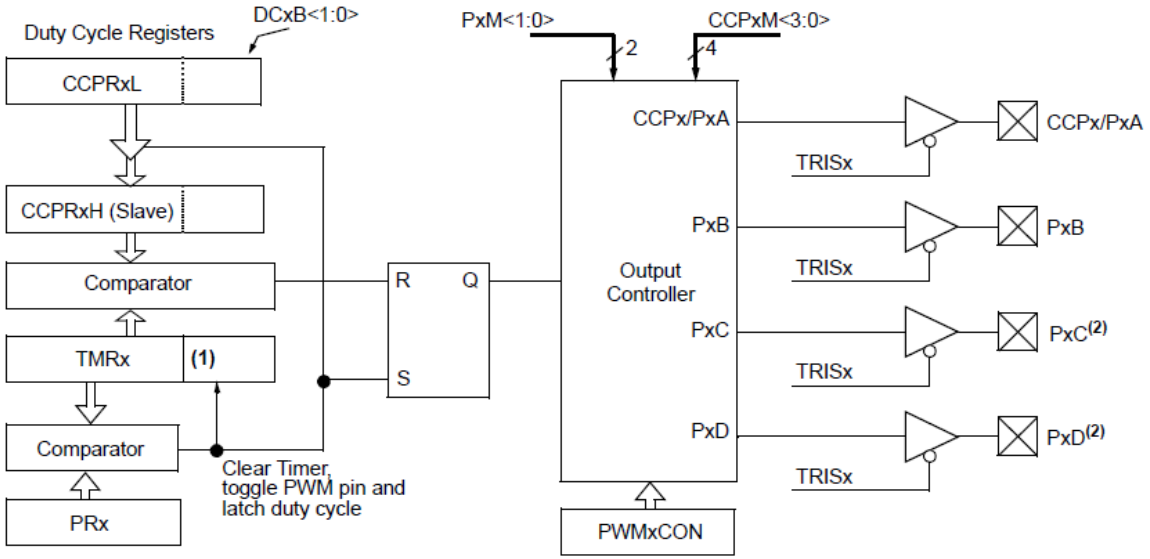
bit 0 **STRxA:** Steering Enable bit A
1 = PxA pin has the PWM waveform with polarity control from CCPxM<1:0>
0 = PxA pin is assigned to port pin

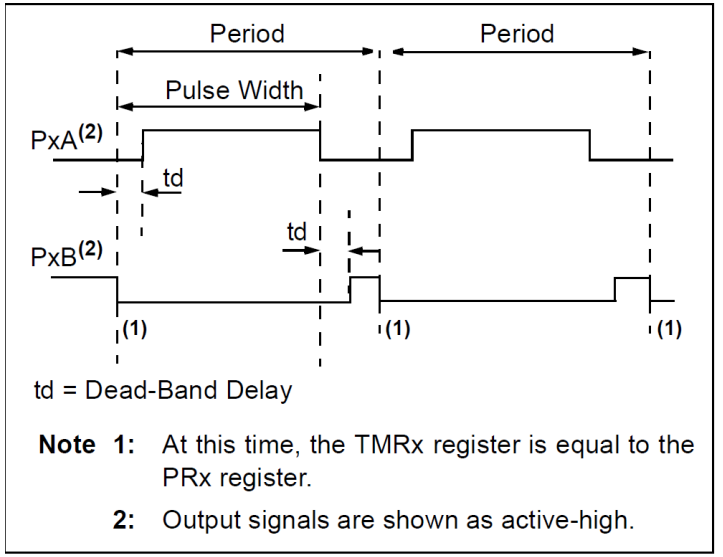
```
//init – PWM
PSTR1CON = 0b11;                    //P1A PWM
CCPTMRS0bits.C1TSEL = 0b00;         //timer2 will be used
PR2 = 200;                          //period
CCP1CON = 0b00001100;               //enable PWM
CCPR1L = 200;                       //duty cycle

//init - timer2
T2CON = 0b00111101;
```

PWM PIC18 módy

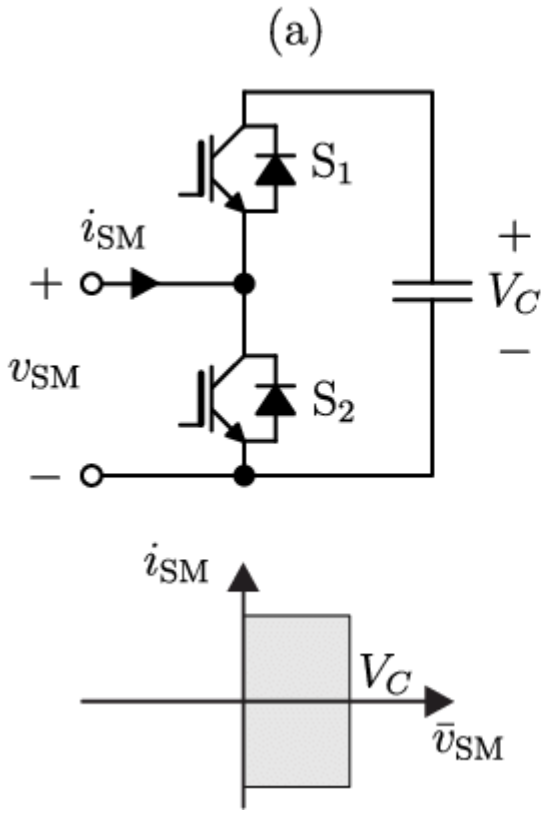
- Kromě jednoduchého single modů umožňuje PWM periferie další specializované konfigurace.
- Half-bridge.
- Full-bridge.
- Dále mohu v těchto konfiguracích generovat dead-time, což umožňuje spínat dva tranzistory nad sebou.
- Mezi sepnutím musí existovat určitý čas, než je předchozí tranzistor spolehlivě vypnut.
- Mezi další možnosti patří konfigurace shut-down modu, kde mohu definovat stav pinu P1A, P1B, P1C a P1D po vyvolání události.
- Událostí může být změna logické hodnoty na definovaném pinu MCU.

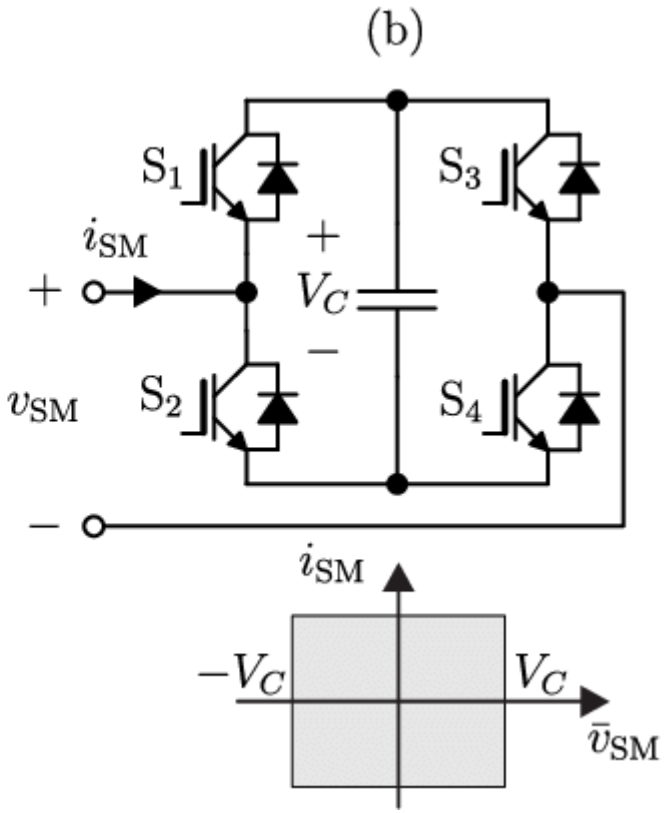




PWM PIC18 módy

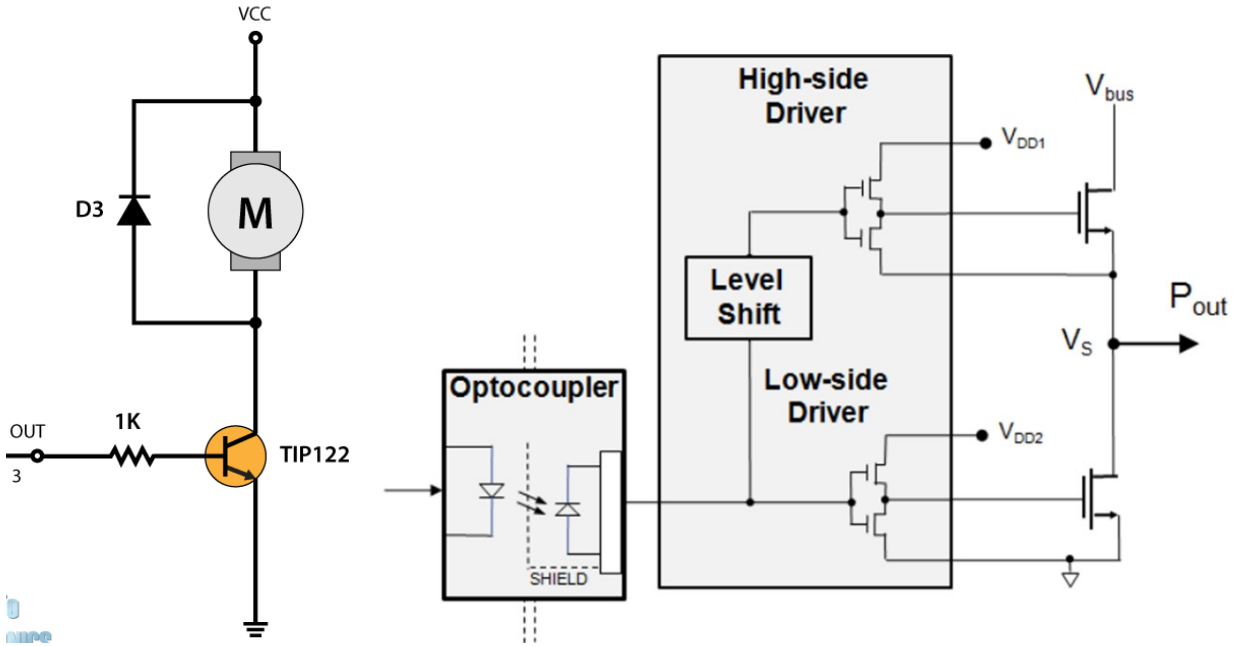
- Enhanced PWM modul obsahuje navíc módy half a full bridge
- Jedná se o dobře známá zapojení z výkonové elektroniky
- Ovládáná DC motoru, power audio a nebo spínaní výkonových transformátorů
- Half-bridge dokáže ovládat oba směry proudu
- Zatímco full-bridge již umožňuje operovat ve všech kvadrantech
- U motoru to například znamená, že je možné provozovat motorické a generátorické režimy v obou směrech



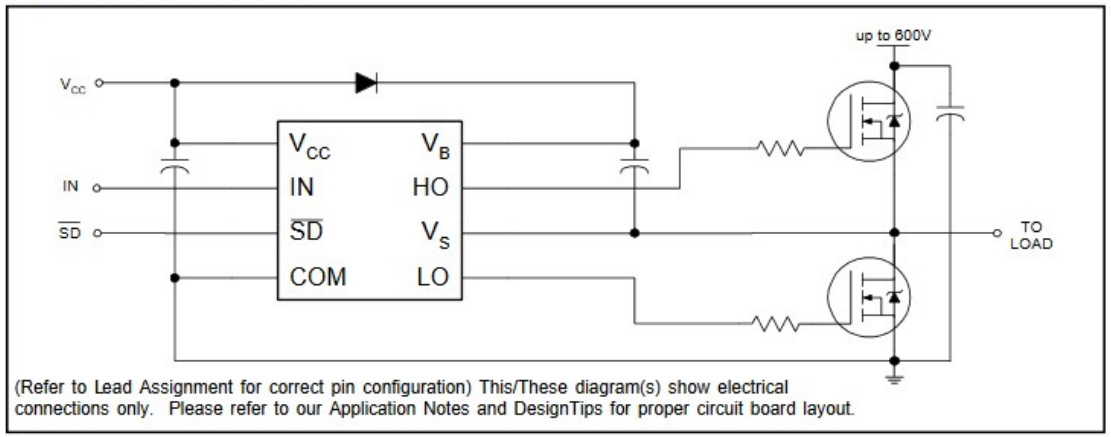


Regulace výkonu

- Při použití N-MOS nastává problém se spínáním horního tranzistoru
- Řeší to použití speciálního zákaznického obvodu tzv. driveru
- Obvod řeší i dead-time, tedy ochranu proti sepnutí obou tranzistoru nad sebou do zkratu
- Infineon IR2104 (vpravo dole)



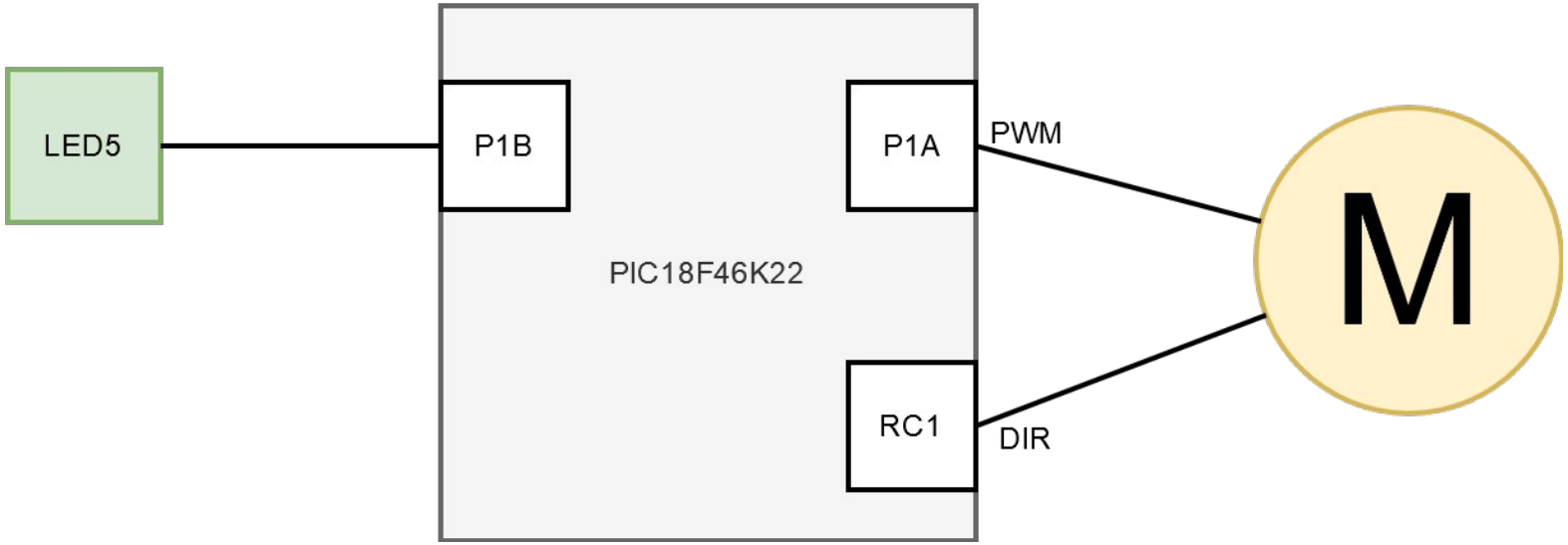
Typical Connection

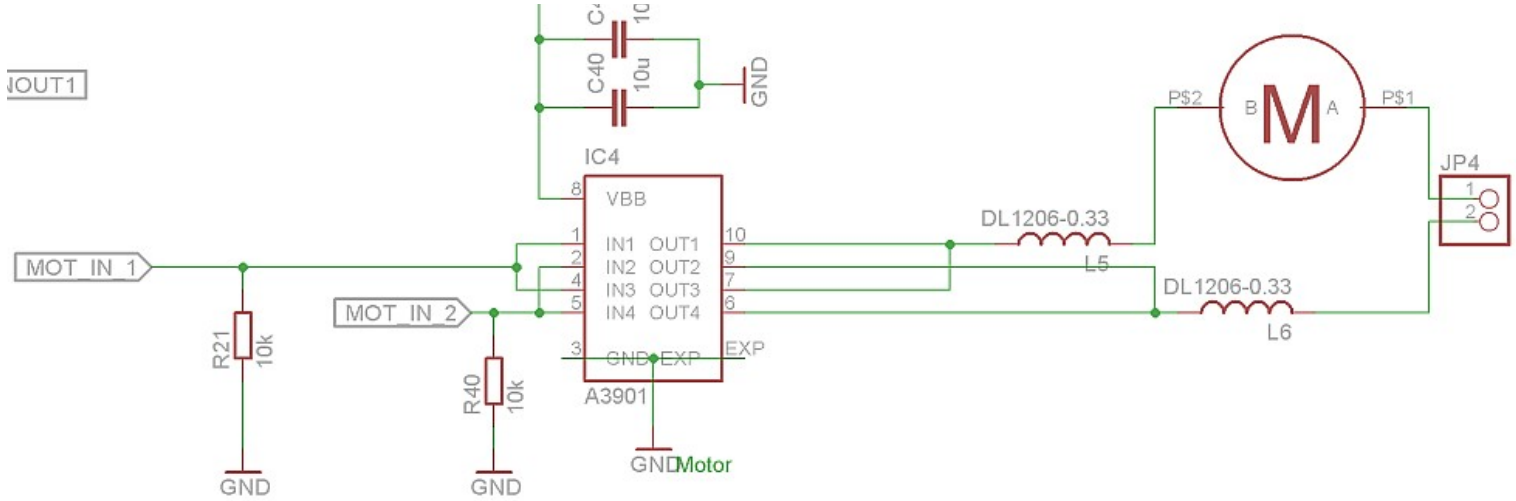


(Refer to Lead Assignment for correct pin configuration) This/These diagram(s) show electrical connections only. Please refer to our Application Notes and DesignTips for proper circuit board layout.

PWM PIC18 módy

- Motor lze ovládat jedním PWM MOT IN 1 Pin P1A.
- MOT IN 2 pak pomocí běžného GPIO pinu volí DIR (směr motoru).
- Invertuje se i PWM!!!
- Tedy pokud mate hodnotu střídy 200, tak bude motor reagovat jako na 55.
- V jednom směru tedy platí střída **s** a v druhém **1-s.**





Úkol:

1. Ovládejte rychlost motoru potenciometrem
   - V polovině rozsahu motor stojí
   - na každou stranu nastavuji rychlost motoru vpravo/vlevo
2. Aktuální rychlost(střídu) a směr otáčení zobrazujte na displeji

Do e-learningu:
Přijmeni_jmeno_body.zip