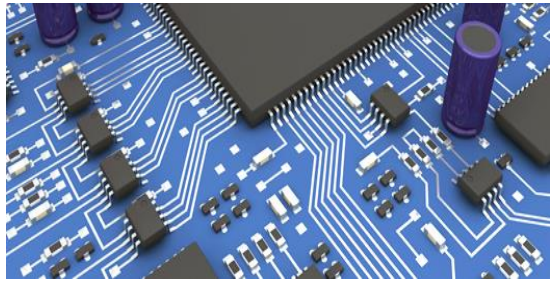
Aplikace Embedded systémů v Mechatronice



MPLAB® X IDE

Michal Bastl
A2/713a

Aplikace Embedded systémů v Mechatronice

Obsah přednášky:

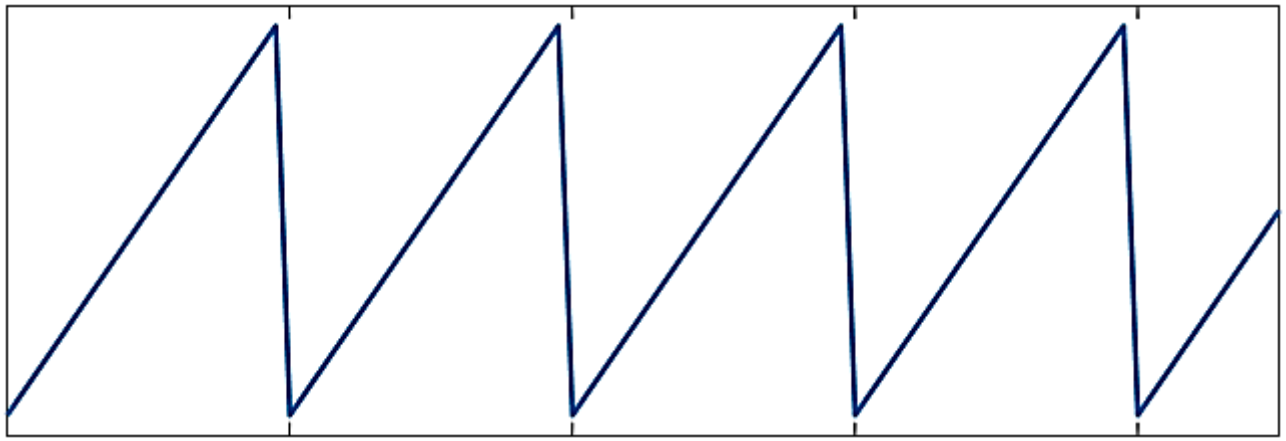
- Opakování
- Timer
- Přerušení
- LCD
- Ukázky použití
- Hardware poznámky

Opakování

Co je GPIO pin?
Jaké registry používáme k ovládání GPIO?
Proč je výhodné používat uživatelská makra?

Timer

- Timer je další periferie MCU, jak napovídá název jedná se o obvod k odměřování času. Jde v podstatě o čítač, který načítá počet pulzů hodinového signálu.
- Zdroj signálu může být upravován před-děličkou
- Aktuální stav Timeru je možné vyčíst z příslušných registrů
- PIC18: 16-bit 1/3/5, 8-bit 2/4/6
- čas se odvozuje od frekvence zdrojového signálu



Timer

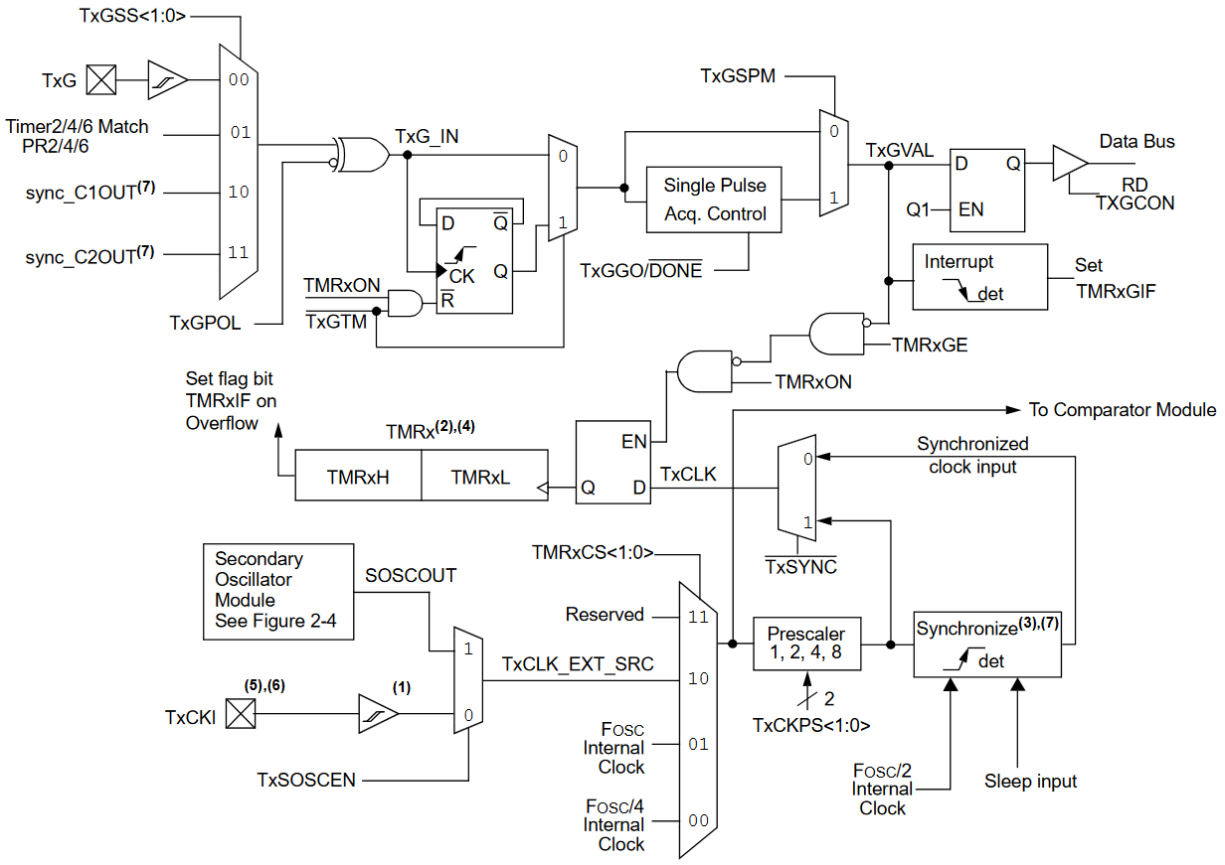
12.13 Register Definitions: Timer1/3/5 Control

REGISTER 12-1: TXCON: TIMER1/3/5 CONTROL REGISTER

R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/0	R/W-0/u
TMRxCS<1:0>		TxCKPS<1:0>		TxSOSCEN	$\overline{\text{TxSYNC}}$	TxRD16	TMRxON
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	

- TMRxCS nastavuje zdroj signalu
- TxCKPS Nastavuje před-děličku
- TMRxON Spouští timer



Použití timeru

Nastavení timeru (inicializace periferie):

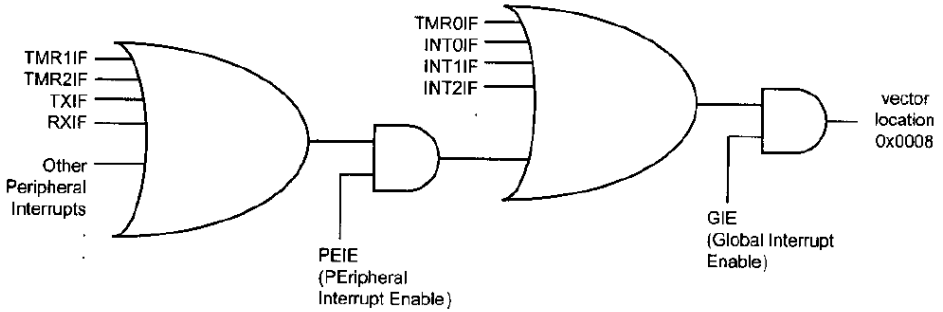
```
void init(void){

   // set pins as outputs
   TRISDbits.TRISD2 = 0;

   // Timer
   T1CONbits.TMR1CS = 0b00;   // TMR1 source (FOSC/4)
   T1CONbits.T1CKPS = 0b11;    // TMR1 prescaler (1:8)
   T1CONbits.TMR1ON = 1;      // TMR1 on
}
```

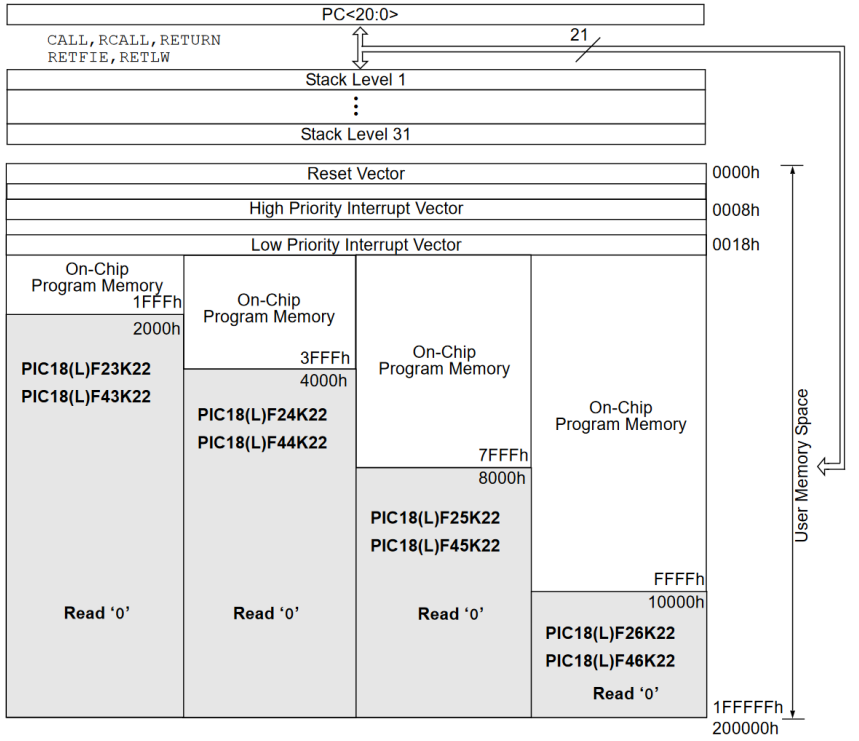
Použití v kódu:

```
void main(void)
{
   init();
   while(1){
      if(TMR1 >= 50000){
         LATDbits.LATD2 ^= 1;;
         TMR1 = 0;
      }
   }
}
```

Interrupt(přerušení)

- Jedná se o techniku, která umožňuje procesoru zpracovávat Asynchronní(nesoučasné) události.
- Kontrolér obsahuje řadič přerušení, který po vyvolání události přeruší program a obslouží přerušení. Poté se vrací zpět.
- Poslední instrukce je dokončena. Adresa následující je uložena do zásobníku a po návratu z obsluhy přerušení se pokračuje následující instrukcí.
- PIC18 má pouze dvě lokace paměti pro vektor přerušení 0x0008 a 0x0018
- Má jen dvě priority přerušení





Interrupt(přerušení)

- Zdrojem přerušení můžou být různé události
- Kompletní seznam nalezne uživatel v datasheetu
- Lze zmínit např. přerušení od timeru, ADC převodníků, změna stavu input pinu, komunikační sběrnice (příjem dat) apod.
- Ve cvičeních budeme pracovat s přerušením, které je vyvolané přetečením registru Timeru.
- Budeme tedy v přesných časových intervalech vykonávat program, který naprogramujeme do tzv. obsluhy přerušení.
- Tato obsluha se nazývá ISR tedy interrupt service rutine.

registry přerušení

9.8 Register Definitions: Interrupt Control

REGISTER 9-1: INTCON: INTERRUPT CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **GIE/GIEH:** Global Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked interrupts
0 = Disables all interrupts including peripherals
When IPEN = 1:
1 = Enables all high priority interrupts
0 = Disables all interrupts including low priority

bit 6 **PEIE/GIEL:** Peripheral Interrupt Enable bit

```
// interupts
  T1CONbits.TMR1CS = 0b00;
  T1CONbits.T1CKPS = 0b11;   // TMR1 prescaler
  T1CONbits.TMR1ON = 1;      // TMR1 on

  /* init - interrupts */
  PEIE = 1;          // global interrupt enable
  GIE = 1;           // peripheral interrupt enable
  TMR1IE = 1;        // enable TMR1 interrupt
```

registry přerušení

REGISTER 9-9: PIE1: PERIPHERAL INTERRUPT ENABLE (FLAG) REGISTER 1

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

```
// interupts
  T1CONbits.TMR1CS = 0b00;
  T1CONbits.T1CKPS = 0b11;   // TMR1 prescaler
  T1CONbits.TMR1ON = 1;      // TMR1 on

  /* init - interrupts */
  PEIE = 1;          // global interrupt enable
  GIE = 1;           // peripheral interrupt enable
  TMR1IE = 1;        // enable TMR1 interrupt
```

Zápis ISR

- Zápis se provádí vytvořením funkce
- Používá se klíčové slovo interrupt
- Pokud není dále specifikováno jedná se o vysokou prioritu
- proměnné, které používám v ISR by měly být deklarováný s užitím klíčového slova volatile (nebudou provedeny žádné optimalizace)
- Používají se globální proměnné, před funkcí main.

```
//global variable
volatile char flag = 0;

void interrupt ISR(void){
   if(TMR1IE & TMR1IF){
      TMR1 -= ISR_PERIOD;
      flag = 1;
      TMR1IF = 0;
  }
}
```

Tuto funkci ISR nelze volat z kódu.
Provádí se s příznakem přerušení
Interrupt flag!

```
while(1){
  if(flag){
     LED1 = ~LED1;
     flag = 0;
  }
```

Zápis ISR

REGISTER 9-4: PIR1: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 1

U-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
—	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Příznak přerušení je vyvolán přetečením TMR1 → začne se vykonávát ISR:

- kontrola zda je interrupt TIMER1 zapnutý a zda došlo ke změně příznaku (interrupt flag)
- Nastavení hodnoty do TMR1 registru
- přepsání proměně, která se používá v hlavní smyčce
- Smazání příznaku přerušení (interrupt flag)

Množství kódu v ISR se snažím omezovat

Pokud nedojde ke smazaní příznaku, tento již se nevyvolá (je zamaskován)

```
//global variable
volatile char flag = 0;

void interrupt ISR(void){
   if(TMR1IE & TMR1IF){
      TMR1 -= ISR_PERIOD;
      flag = 1;
      TMR1IF = 0;
   }
}


 while(1){
   if(flag){
      LED1 = ~LED1;
      flag = 0;
   }
```

Více zdrojů přerušení

```
T1CONbits.TMR1CS = 0b00;
T1CONbits.T1CKPS = 0b11;   // TMR1 prescaler
T1CONbits.TMR1ON = 1;      // TMR1 on

T5CONbits.TMR5CS = 0b00;
T5CONbits.T5CKPS = 0b11;   // TMR1 prescaler
T5CONbits.TMR5ON = 1;      // TMR1 on

/* init - interrupts */
PEIE = 1;          // global interrupt enable
GIE = 1;           // peripheral interrupt enable
TMR1IE = 1;        // enable TMR1 interrupt
TMR5IE = 1;        // enable TMR1 interrupt
```

```
void interrupt ISR(void){
  if(TMR1IE & TMR1IF){
      TMR1 = 0x8000;
      LED1 ^= 1;
      TMR1IF = 0;
  }

  if(TMR5IE & TMR5IF){
      TMR5 = 0;
      LED2 ^= 1;
      TMR5IF = 0;
  }
}


 while(1){
      asm("NOP");
  }
```

LCD

RAYSTAR OPTRONICS RX1602A3-BIW-TS:

řadič: ST7032
sběrnice: I2C

Použití:

```
#include <stdio.h>
#include "lcd.h„

 LCD_Init();
 char text[17];
 sprintf(text,"Mechlab je bozi!"); //funkce z stdio.h
 LCD_ShowString(1,text);

 if(BTN1){
      LCD_Clear();    //smazání
```