Projecto FST Novabase – Electrónica

Relatório Semanal - 14/11/2014

Por Laura Gouveia

Programação para dsPIC

Colocar pin a HIGH

- Declarar pin como output (TRISx: 0 – output; 1-input)
- Declarar valor do pin (LATx)

```
//coloca pin a HIGH
void bitB0_high(){
      TRISBbits.TRISB0 = 0; //set pin 0 port B as output
      LATBbits.LATB0 = 1; // set pin 0 port B a high
}
```

Inicializar Timer1

- Declarar funcionamento em idle, uso de gaterd time, relógio a usar para sincronizar (interior ou externo) e prescale de frequência a usar na frequência de relógio.
- Declarar parâmetros de interrupção.
- Declarar valor final do timer (PRx) e valor inicial (TMRx).
- Iniciar timer.

```
//inicializar timer 1
void timer1_init(){
      T1CONbits.TSIDL = 0; //continue in idle mode
      T1CONbits.TGATE = 0; //disable gated time accumulation
      T1CONbits.TCS = 0; //use internal clock (TSYNC is ignored)
      T1CONbits.TCKPS = 3; //prescale 1:256

      IEC0bits.T1IE = 1; //interrupt enable timer1
      IFS0bits.T1IF = 0; //clear interrupt flag
      IPC0bits.T1IP = 4; // prioridade 4

      PR1 = 29297; //valor final do timer 1 (FCY/PRESCALE) Freq = 1 Hz
      TMR1 = 0; //valor inicial do timer1
```

```
        T1CONbits.TON = 1; //starts timer1
}
```

Tratamento de interrupção do Timer1 – Inverter valor do pin

- Tratamento de interrupção.
- Limpar flag de interrupção.

```
//tratamento interrupçao timer1
void __attribute__((interrupt, auto_psv, shadow)) _T1Interrupt(void){
        LATBbits.LATB0 =!LATBbits.LATB0;//inverte valor do pin 0 do port B
        IFS0bits.T1IF = 0;              //clear interrupt flag
}
```

Pulse Width Modulation

- Inicializar Output Compare pin como output.
- Declarar modo em idle, timer a ser utilizado e modo de Output Compare
- Declarar valor inicial, em registo (OCxR) e registo auxiliar(OCxRS).
- Inicializar timer.

```
//incializa pulse width modulation
void init_pwm(){
        TRISDbits.TRISD0 = 0;     //output OC1
        OC1CONbits.OCSIDL = 1;    //disable output compare in idle
        OC1CONbits.OCTSEL = 0;    //use timer2
        OC1CONbits.OCM = 6;       //pwm without fault mode

        OC1R = 0; //valor inicial
        OC1RS = 0; //duty cycle
}

//incializa timer2 (pwm)
void timer2_init(){
        T2CONbits.TSIDL = 0; //continue in idle mode
        T2CONbits.TGATE = 0; //disable gated time accumulation
        T2CONbits.TCS = 0; //use internal clock (TSYNC is ignored)
        T2CONbits.TCKPS = 0; //prescale 1:1
```

```
	IEC0bits.T2IE = 0; //interrupt disable timer2
	IFS0bits.T2IF = 0; //clear interrupt flag
	IPC1bits.T2IP = 4; // prioridade 4

	PR2 = 7500; //valor final do timer2 FCY/PRESCALE fpwm=1kHz
	TMR2 = 0; //valor inicial do timer2

	T2CONbits.TON = 1; //starts timer2
}
```

Aumentar duty cycle de PWM

```
//tratamento interrupçao timer1
void __attribute__((interrupt, auto_psv, shadow)) _T1Interrupt(void){
	OC1RS++; //incrementa duty cycle
	if(OC1RS==7500) //duty cycle 100%?
		OC1RS=0;  //reset duty cycle

	IFS0bits.T1IF = 0; //clear interrupt flag
}
```

Cálculo do período e resolução de PWM

Equation 14-1: Calculating the PWM Period

$$\text{PWM Period} = [(PRy) + 1] \cdot T_{CY} \cdot (\text{TMRy Prescale Value})$$

$$\text{PWM Frequency} = 1/[\text{PWM Period}]$$

Equation 14-2: Calculation for Maximum PWM Resolution

$$\text{Maximum PWM Resolution (bits)} = \frac{\log_{10}\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log_{10}(2)} \text{ bits}$$