Programação de Sistemas Embarcados Baseados em Microcontroladores PIC

2º dia - Programação de Módulos e Periféricos do microcontrolador PIC

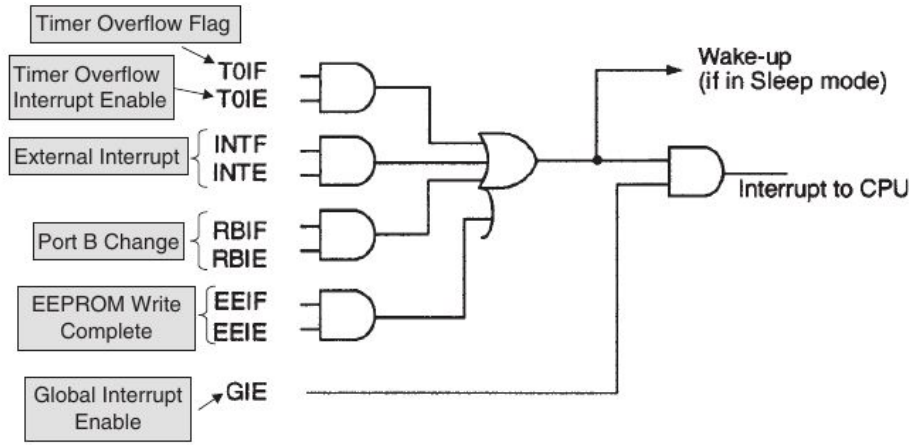
Blink Tradicional

- Como fazer multitasking dessa forma?

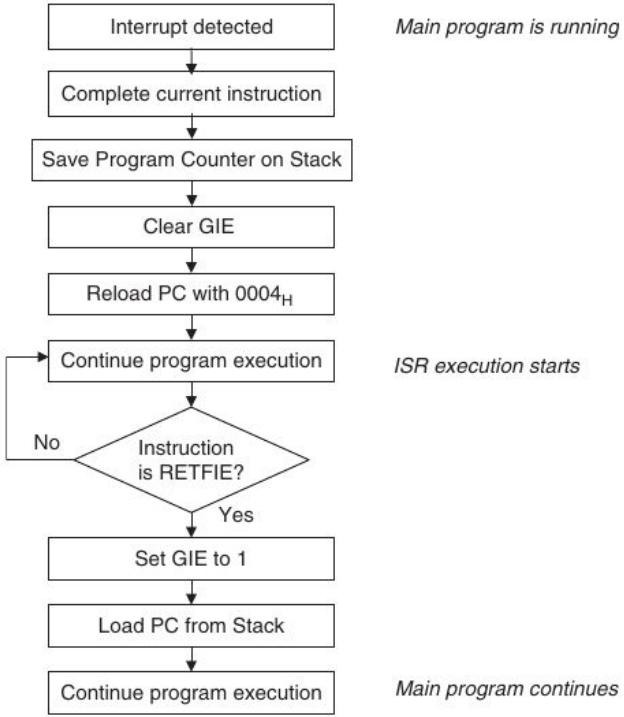
```
11  void main(void) {
12      TRISB0=0;
13      while(1){
14      RB0=0;
15      __delay_ms(200);
16      RB0=1;
17      __delay_ms(200);
18      }
19      return;
20  }
```

Interrupções: O distúrbio da ordem

- Um Alerta a CPU de que um evento significativamente importante ocorreu



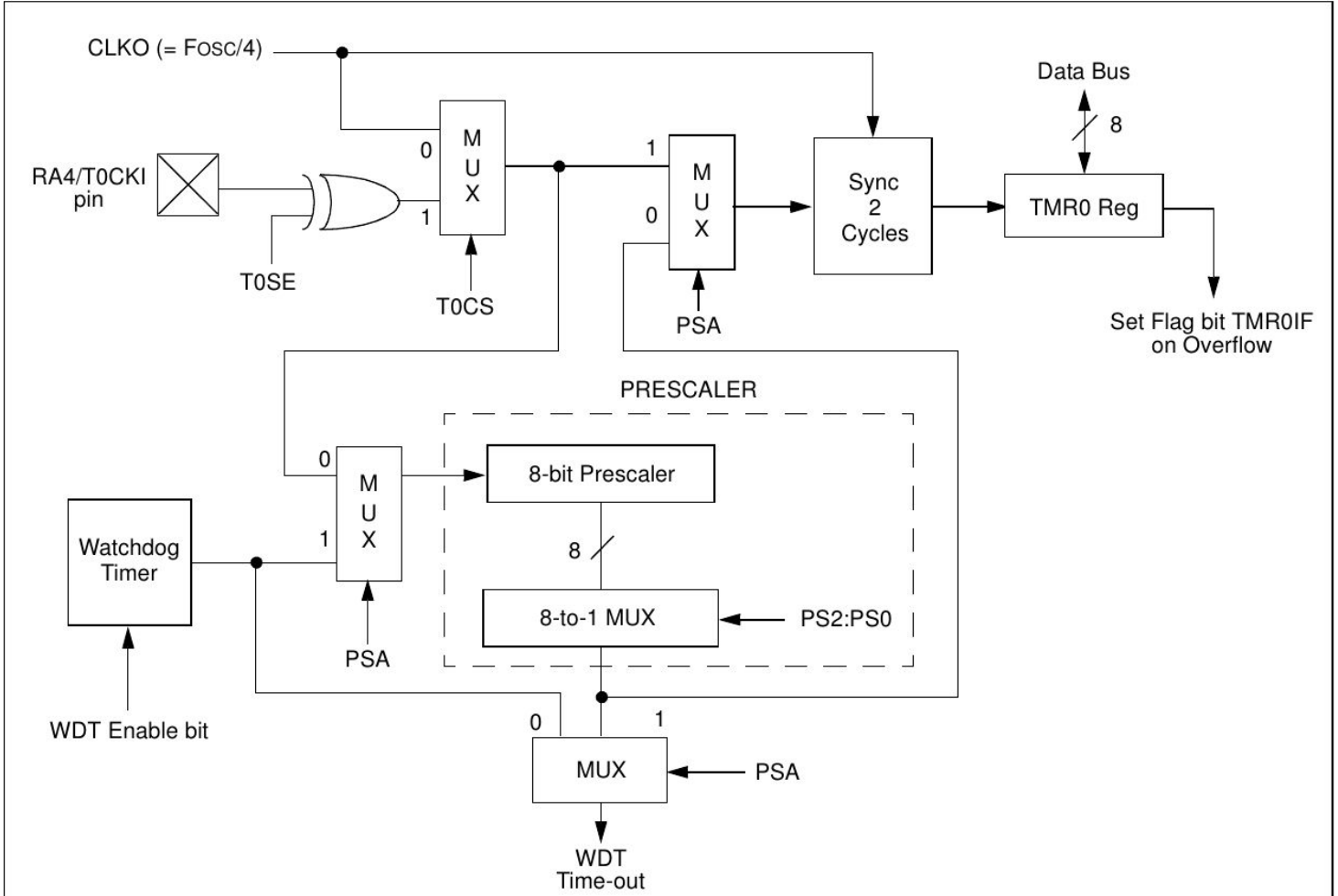
Fluxograma de uma interrupção



Timers

- Nosso modelo de PIC (PIC16F877A) possui três Timers: Timer 0, Timer 1, Timer2

- Contador e temporizador
  - Temporização do Sistema
  - Utilização no módulo de captura e comparação de sinais (Timer1)
  - Utilização no módulo PWM (Timer2)



Note: T0CS, T0SE, PSA, PS2:PS0 are (OPTION_REG<5:0>).

Registradores associados ao Timer 0

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
01h,101h	TMR0	Timer0 Module Register								xxxx xxxx	uuuu uuuu
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
81h,181h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

timer0lib.h

```
7
8   #ifndef TIMER0LIB_H
9   #define TIMER0LIB_H
10
11
12  #define PRESCALER2      0b00000000
13  #define PRESCALER4      0b00000001
14  #define PRESCALER8      0b00000010
15  #define PRESCALER16     0b00000011
16  #define PRESCALER32     0b00000100
17  #define PRESCALER64     0b00000101
18  #define PRESCALER128    0b00000110
19  #define PRESCALER256    0b00000111
20
21  void configTimer0(unsigned char prescalerValue);
22
23  void setTimer0Value(unsigned char value);
24
25  unsigned char getTimer0Value(void);
26
27  void resetTimer0(void);
28
29  void isrTimer0(void);
30
31
32  #ifdef  __cplusplus
33  extern "C" {
34  #endif
35
36
37
```

Implementação dos source files

- timer0lib.c
  - configTimer0()
  - setTimer0Value()
  - getTimer0Value()

- main.c
  - Desenvolver um BLINK (simular no Proteus) utilizando interrupção do Timer 0
  - Na frequência de 1 Hz

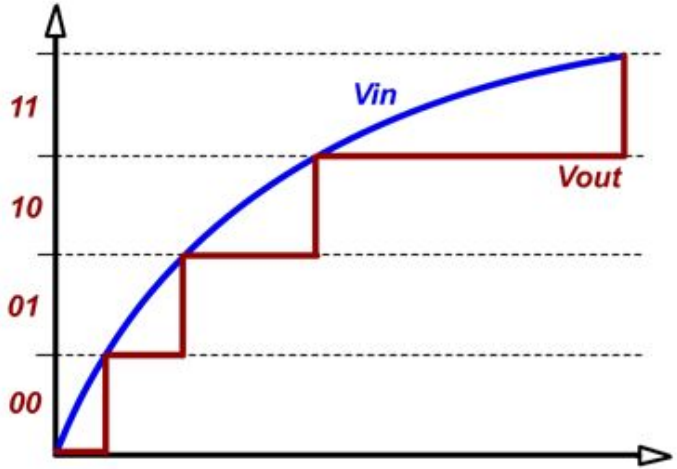
Desafio

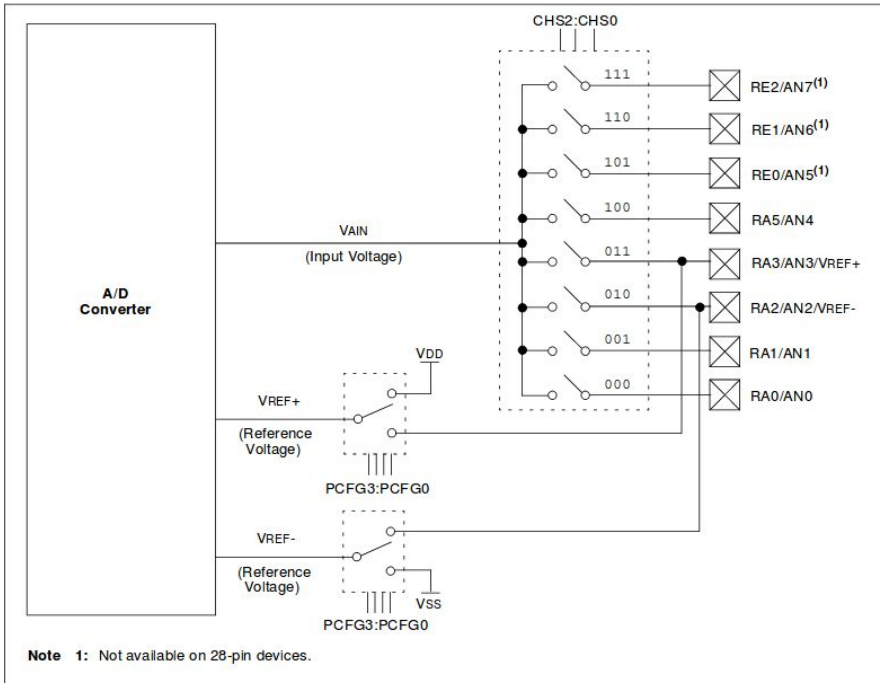
A partir da biblioteca desenvolvida, implementar um projeto de blink com dois leds em frequências diferentes

- Utilize a interrupção do Timer0

Conversor Analógico Digital (ADC)

- Conversão de sinais analógicos para binários de 10 bits





Que taxa de conversão escolher?

AD Clock Source (TAD)		Maximum Device Frequency
Operation	ADCS2:ADCS1:ADCS0	
2 TOSC	000	1.25 MHz
4 TOSC	100	2.5 MHz
8 TOSC	001	5 MHz
16 TOSC	101	10 MHz
32 TOSC	010	20 MHz
64 TOSC	110	20 MHz
RC(1, 2, 3)	x11	(Note 1)

Registradores associados ao ADC

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on MCLR, WDT
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
1Eh	ADRESH	A/D Result Register High Byte								xxxx xxxx	uuuu uuuu
9Eh	ADRESL	A/D Result Register Low Byte								xxxx xxxx	uuuu uuuu
1Fh	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON	0000 00-0	0000 00-0
9Fh	ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	00-- 0000	00-- 0000
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	--11 1111
05h	PORTA	—	—	PORTA Data Latch when written: PORTA pins when read						--0x 0000	--0u 0000
89h(1)	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction bits			0000 -111	0000 -111
09h(1)	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	---- -uuu

adclib.h

```
10
11  #define ADC_CLOCK_2     0b00000000
12  #define ADC_CLOCK_8     0b00100000
13  #define ADC_CLOCK_32    0b01000000
14  #define ADC_CLOCK_RC    0b01100000
15  #define ADC_CLOCK_4     0b10000000
16  #define ADC_CLOCK_16    0b10100000
17  #define ADC_CLOCK_64    0b11000000
18
19  #define ADC_CONFIG_0     0b00000000
20  #define ADC_CONFIG_1     0b00000001
21  #define ADC_CONFIG_2     0b00000010
22  #define ADC_CONFIG_3     0b00000011
23  #define ADC_CONFIG_4     0b00000100
24  #define ADC_CONFIG_5     0b00000101
25  #define ADC_CONFIG_6     0b00000110
26  #define ADC_CONFIG_7     0b00001000
27  #define ADC_CONFIG_8     0b00001001
28  #define ADC_CONFIG_9     0b00001010
29  #define ADC_CONFIG_10    0b00001011
30  #define ADC_CONFIG_11    0b00001100
31  #define ADC_CONFIG_12    0b00001101
32  #define ADC_CONFIG_13    0b00001110
33  #define ADC_CONFIG_14    0b00001111
34
35  #define RIGHT_JUST      0b10000000
36  #define LEFT_JUST       0b00000000
37
38  void adcConfig(unsigned char conversionRate,unsigned char portConfig);
39  unsigned int adcRead(unsigned char channel,unsigned char resultFormat);
40
```

Implementação dos Source Files

- adclib.c
  - adcConfig();
  - adcRead();
- main.c
  - Desenvolver uma aplicação que leia um sinal analógico (a partir de um potenciômetro) e refletir seu valor digitais em LEDS ligados ao PORTB e ao PORTC (RC0,RC1)
  - Simular no Proteus

Desafio

- Implementar um Blink de alarme quando o valor lido pelo ADC ultrapassar um valor limite
  - Usar o timer 0 para o blink