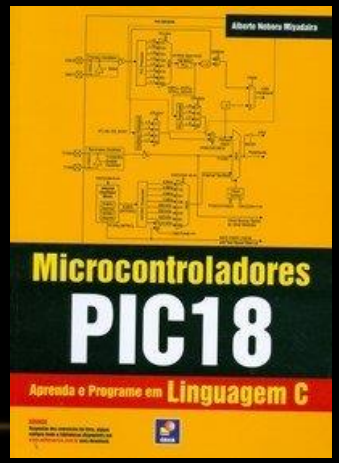
MICROPROCESSADORES II

Professor: Patric Janner Marques

Aula : Utilização e configuração de Interrupções

Referência bibliográfica

- Para aula de hoje:
  - Salvador: Cap. 7.
  - Miyadaira: Cap. 9.

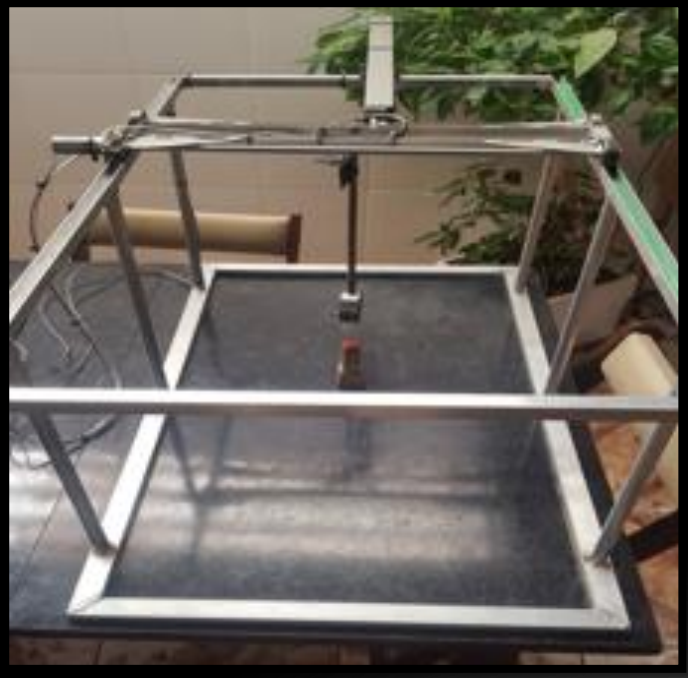


Interrupção geral

- Em sistemas microcontrolados, muitas vezes um projeto exige que providências sejam tomadas imediatamente no momento em que um determinado evento ocorre, porém não se pode ficar o tempo todo verificando as entradas associadas a esse evento, seria um desperdício de processamento.

Interrupção geral

- Exemplificando com um projeto de ponte rolante:

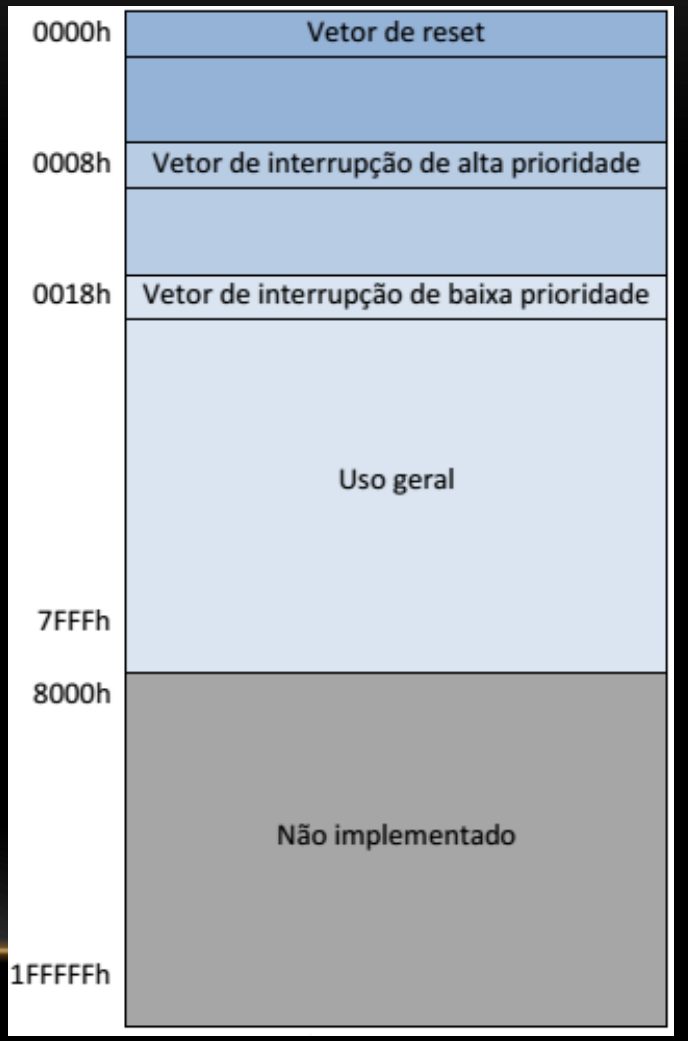


Interrupção geral

- Uma interrupção basicamente interrompe a execução normal do programa para executar tarefas de maior prioridade no momento;
- Ela pode ser chamada por determinados eventos, parando a execução normal no momento em que ocorre o evento, executando o código armazenado em uma posição de memória vinculada a interrupção e depois retornam a execução normal do programa;
- As interrupções podem ser consideradas chamadas de sub-rotinas realizadas pelo *hardware*.

Interrupção geral

- Quando ocorre um pedido de interrupção para a CPU tipo botão de *reset*, o programa desviado para um endereço pré-estabelecido, vetor de *reset*;
- De forma semelhante ocorre quando ocorre um pedido de interrupção para a CPU por outro evento, desviando para um endereço;
- No PIC18 existem dois vetores de interrupção, e a partir deste endereço é feito o chamado tratamento de interrupção, isto é, são executadas as rotinas que se deseja associar ao evento causador da interrupção.



Interrupção geral

- O PIC18F4550 possui 20 fontes de interrupções, oriundas de eventos externos como de sinais digitais externos e comparadores analógicos, e de eventos internos como estouro de *timers* e *watchdog*.
- A cada interrupção são associados três bits essenciais:
  - Bit de habilitação: indica se a interrupção está ativa (0 = interrupção desativada, 1 = interrupção ativa);
  - Bit de flag: indica se ocorreu o evento associado à interrupção (0 = não ocorreu, 1 = ocorreu);
  - Bit de prioridade: indica a prioridade da interrupção (0 = baixa prioridade, 1 = alta prioridade)

Interrupção geral

- A ocorrência de um evento faz com que um bit de *flag* vá para 1, caso contrário ele permanecerá sempre em 0.
- Contudo, para que **haja um pedido de interrupção** para a CPU é necessário que a **interrupção associada a esse evento esteja ativa**, isto é, seu bit de habilitação seja 1.
- Em outras palavras, é possível ativar as interrupções desejadas e deixar as demais inativas.

Interrupção geral

- Existem dois níveis de prioridade de interrupções, alto e baixo.
- Uma interrupção de baixa prioridade interrompe a execução do programa e causa o desvio para o vetor de interrupção de baixa prioridade.
- Já a interrupção de alta prioridade interrompe não só a execução normal do programa como também interrompe a rotina de tratamento de interrupção de baixa prioridade (se estiver sendo executada).
- Cada fonte de interrupção pode ser configurada para atuar com baixa ou alta prioridade, sendo que para isso basta definir se o bit de prioridade é 0 (baixa) ou 1 (alta).

Interrupção geral

- Para habilitar o sistema de prioridade de interrupção o bit IPEN deve estar em 1 (pg. 46 do *datasheet*);
- Caso não queira utilizar prioridade IPEN = 0 (sistema padrão);
  - Neste caso, todos as interrupções convergem para o endereço 0008h (pg. 59 do *datasheet*).

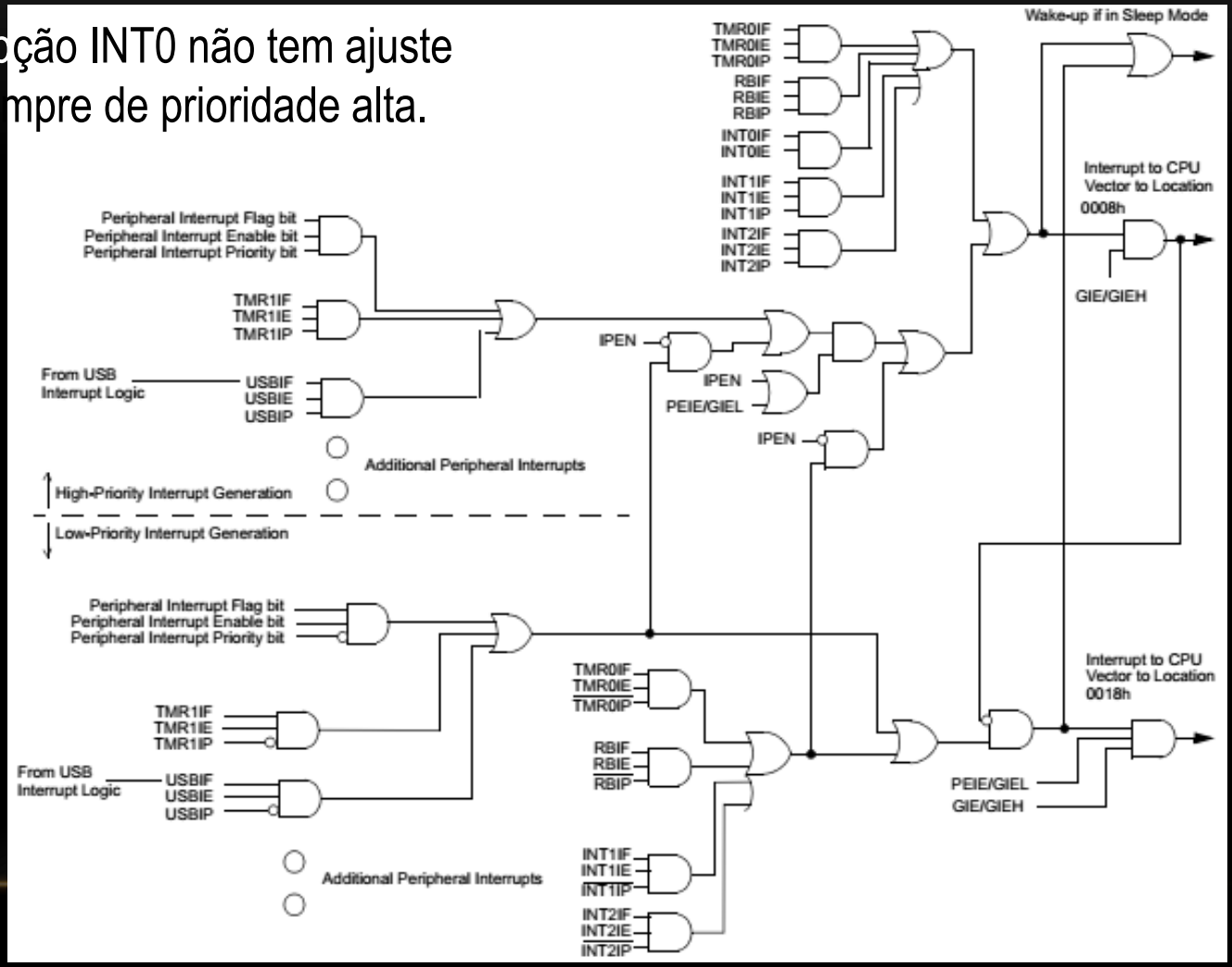
Interrupção geral

- Se for utilizar a prioridade existem duas habilitações globais (que habilitam todas as interrupções em função da sua prioridade), uma para prioridade alta (GIEH) e outra para baixa (GIEL);

- Caso não queira utilizar prioridade, existe outra forma de habilitar as interrupções: (GIE) para habilitar todas as interrupções e (PEIE) para habilitar as interrupções de periféricos (ADC, EUSART, CCP, USB, pg. 104 do *datasheet*).

Interrupção geral

Observação: A interrupção INT0 não tem ajuste de prioridade, ela é sempre de prioridade alta.
Pg. 100 do *datasheet*



Interrupção geral

- IMPORTANTE:
  - **O bit de *flag* vai para 1 quando ocorre o evento mesmo que a interrupção associada a ele não esteja habilitada**.
  - Além disso, o *hardware* força o bit de *flag* para 1, **mas o mesmo deve ser zerado pelo *software*** para indicar que a interrupção em questão foi tratada.

Interrupção geral

- Quando ocorre uma interrupção, o fluxo do programa desvia para um vetor de interrupção (conforme a prioridade da interrupção);
- Consequentemente, a rotina de tratamento de interrupção deve obrigatoriamente começar nesse endereço.
- IMPORTANTE:
  - Para determinar qual interrupção ocorreu, testa-se os bits de *flags* de todas as interrupções ativas com a mesma prioridade.

Interrupção geral

- Um outro cuidado que deve ser tomado quando se trabalha com interrupções é não executar rotinas muito grandes, isso porque o tratamento de interrupção deve tomar medidas emergenciais;
- Nada que não seja estritamente necessário deve ser trabalhado na interrupção, e isso principalmente porque podem ocorrer várias interrupções em seguida, e só se pode tratar uma por vez;
- Quando uma interrupção é chamada, a habilitação das demais fica desativada, e só é reativada após a execução do RETFIE (comando interno de retorno da interrupção);
- Se outra interrupção de mesma prioridade ocorrer nesse intervalo de tempo, ela é tratada após RETFIE;

Interrupção geral

- **Deve-se notar que a ativação da chamada de interrupção se dá pela existência de um *flag* em 1 de uma interrupção habilitada**. Por isso, depois de tratar uma determinada interrupção **deve-se zerar o *flag*** associado a ela;
- Caso contrário, o programa fica travado, pois cada vez que há o retorno de interrupção, a interrupção é chamada novamente.

Interrupção geral

- As interrupções ditas externas ao microcontrolador, são a interrupção INT0 (pino RB0), INT1 (pino RB1), INT2 (pino RB2) e a interrupção por mudança de estado da porta B;
- As demais interrupções são internas, causadas pelos periféricos para indicar determinados eventos (chegada de dado pelo portal serial, “estouro” de *timer*, etc).

Interrupção – Configuração e utilização

- Para trabalhar com uma interrupção deve-se primeiramente configurar o periférico associado a ela, o que é feito normalmente uma única vez, na inicialização programa (além do TRIS).

- Mas isso não impede que um periférico seja configurado, ou reconfigurado, em qualquer outro ponto do programa.

Interrupção – Configuração e utilização

- O próximo passo é habilitar as interrupções a serem utilizadas, e isso pode ser feito em dois momentos;
- Pode-se determinar que uma interrupção esteja ativa durante toda a execução do programa, e nesse caso sua habilitação deve ser feita na inicialização;
- Por outro lado, em algumas situações deseja-se que a interrupção seja ativada somente a partir de um dado momento ou ainda que seja ativada e, posteriormente, desativada;
- Nessa situação basta habilitar, ou desabilitar, a interrupção no momento oportuno.

Interrupção – Configuração e utilização

- IMPORTANTE:
  - É importante lembrar que as interrupções habilitadas somente causam chamadas de interrupção se o bit de **habilitação global estiver habilitado** (GIEH e GIEL quando houver prioridade, ou GIE e PEIE para sem prioridade).

Interrupção – Configuração e utilização

- Finalmente, deve-se fazer o tratamento de interrupção, isto é, construir as rotinas que serão chamadas quando ocorrer uma determinada interrupção.

Mas primeiro, vamos ver quais os registrados precisamos setar para habilitar e configurar cada interrupção, onde podemos ver na pg. 99 do *datasheet*, e como faremos isso.....

Interrupção – Acesso aos registradores

R/W – 0	R/W-1	U – 0	R/W – 1	R – 1	R – 1	R/W – 0	R/W – 0
IPEN	**SBOREN**	**-**	**RI**	**TO**	**PD**	**POR**	**BOR**
Bit 7							Bit 0

RCON

- **IPEN** : habilitação de prioridade de interruipção
  - 1 = Prioridade Habilitada
  - 0 = Prioridade desabilitada

Interrupção – Acesso aos registradores

R/W – 0	R/W – 0	R/W – 0	R/W – 0	R/W – 0	R/W – 0	R/W – 0	R/W – 0
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
Bit 7							Bit 0

INTCON

- **GIE/GIEH** : Habilitação Global
  - Se IPEN = 0
    - 1 = Habilita as interrupções
    - 0 = Desabilita as interrupções
  - Se IPEN = 1
    - 1 = Habilita as interrupções de alta prioridade
    - 0 = Desabilita interrupções de alta prioridade
- **PEIE**: Habilitação de interrupções de periféricos
  - Se IPEN = 0
    - 1 = Habilita as interrupções de periféricos
    - 0 = Desabilita as interrupções de periféricos
  - Se IPEN = 1
    - 1 = Habilita as interrupções de baixa prioridade
    - 0 = Desabilita interrupções de baixa prioridade

Interrupção – Acesso aos registradores

R/W – 0	R/W – 0	R/W – 0	R/W – 0	R/W – 0	R/W – 0	R/W – 0	R/W – 0
GIE/GIEH	**PEIE/GIEL**	**TMR0IE**	**INT0IE**	**RBIE**	**TMR0IF**	**INT0IF**	**RBIF**
Bit 7							Bit 0

INTCON

- **TMR0IE**: Habilitação de interrupção do Timer 0
  - 1 = interrupção habilitada
  - 0 = interrupção desabilitada
- **INT0IE**: Habilitação de interrupção Externa 0 (RB0)
  - 1 = interrupção habilitada
  - 0 = interrupção desabilitada
- **RBIE**: Habilitação de interrupção por mudança de estado no PORTB
  - 1 = interrupção habilitada
  - 0 = interrupção desabilitada
- **TMR0IF**: Flag da interrupção do Timer 0
  - 1 = Houve *overflow* do timer 0
  - 0 = Não houve *overflow* do timer 0
- **INT0IF**: Flag da interrupção Externa 0 (RB0)
  - 1 = Houve transição no pino RB0
  - 0 = Não houve transição no pino RB0
- **RBIF**: Flag da interrupção por mudança de estado no PORTB
  - 1 = houve alteração em um dos bits de RB7:RB4
  - 0 = não houve alteração em um dos bits de RB7:RB4

Interrupção – Acesso aos registradores

R/W – 0	R/W – 0	R/W – 0	R/W – 0	U – 0	R/W – 0	U – 0	R/W – 0
RBPU	**INTEDG0**	**INTEDG1**	**INTEDG2**	**-**	**TMR0IP**		**RBIP**
Bit 7							Bit 0

INTCON2

- **RBPU** : Habilitação dos Pull-ups de PORTB
  - 1 = desabilita Pull-ups
  - 0 = Habilita Pull-ups
- **INTEDG0**: Seleção da transição da interrupção INT0
  - 1 = Transição de subida
  - 0 = Transição de descida
- **INTEDG1**: Seleção da transição da interrupção INT1
  - 1 = Transição de subida
  - 0 = Transição de descida
- **INTEDG2**: Seleção da transição da interrupção INT2
  - 1 = Transição de subida
  - 0 = Transição de descida

Interrupção – Acesso aos registradores

R/W – 0	R/W – 0	R/W – 0	R/W – 0	U – 0	R/W – 0	U – 0	R/W – 0
RBPU	**INTEDG0**	**INTEDG1**	**INTEDG2**	**-**	**TMR0IP**		**RBIP**
Bit 7							Bit 0

INTCON2

- **TMR0IP**: Prioridade da Interrupção do Timer 0
  - 1 = Alta prioridade
  - 0 = Baixa prioridade
- **RBIP**: Prioridade da Interrupção de PORTB
  - 1 = Alta prioridade
  - 0 = Baixa prioridade

Interrupção – Acesso aos registradores

R/W – 0	R/W – 0	U – 0	R/W – 0	R/W – 0	U – 0	R/W – 0	R/W – 0
INT2IP	**INT1IP**	**-**	**INT2IE**	**INT1IE**	**-**	**INT2IF**	**INT1IF**
Bit 7							Bit 0

INTCON3

- **INT2IP**: Prioridade da Interrupção INT2
  - 1 = Alta prioridade
  - 0 = Baixa prioridade
- **INT1IP**: Prioridade da Interrupção INT1
  - 1 = Alta prioridade
  - 0 = Baixa prioridade
- **INT2IE**: Habilitação de interrupção Externa 2 (RB2)
  - 1 = interrupção habilitada
  - 0 = interrupção desabilitada
- **INT1IE**: Habilitação de interrupção Externa 1 (RB1)
  - 1 = interrupção habilitada
  - 0 = interrupção desabilitada
- **INT2IF**: Flag da interrupção Externa 2 (RB2)
  - 1 = Houve transição no pino RB2
  - 0 = Não houve transição no pino RB2
- **INT1IF**: Flag da interrupção Externa 1 (RB1)
  - 1 = Houve transição no pino RB1
  - 0 = Não houve transição no pino RB1

Interrupção – Acesso aos registradores

Evento de Interrupção	*Flag* *1 = ocorreu* *0 = não ocorreu*	*Habilitação* *1 = Habilitada* *0 = Desabilitada*	*Prioridade* *1 = Alta* *0 = Baixa*
	INTCON	**INTCON**	**INTCON2**
Interrupção externa INT0	INT0IF	INT0IE	Sempre alta
Estouro do timer 0	TMR0IF	TMR0IE	TMR0IP
Mudança de estado na porta B	RBIF	RBIE	RBIP
	INTCON3	**INTCON3**	**INTCON3**
Interrupção externa INT1	INT1IF	INT1IE	INT1IP
Interrupção externa INT2	INT2IF	INT2IE	INT2IP
	PIR1	**PIE1**	**IPR1**
Leitura ou escrita da porta paralela escrava	PSPIF	PSPIE	PSPIP
Conversão do AD completa	ADIF	ADIE	ADIP
Recepção serial USART	RCIF	RCIE	RCIP
Transmissão serial	TXIF	TXIE	TXIP
Transmissão/recepção completa pelo MSSP	SSPIF	SSPIE	SSPIP
Interrupção do módulo CCP1	CCP1IF	CCP1IE	CCP1IP
Estouro do timer 2	TMR2IF	TMR2IE	TMR2IP
Estouro do timer 1	TMR1IF	TMR1IE	TMR1IP
	PIR2	**PIE2**	**IPR2**
Falha no oscilador	OSCFIF	OSCFIE	OSCFIP
Alteração no estado do comparador	CMIF	CMIE	CMIP
Término de escrita na EEPROM	EEIF	EEIE	EEIP
Colisão no barramento SPI	BCLIF	BCLIE	BCLIP
Detecção de baixa/alta tensão	HLVDIF	HLVDIE	HLVDIP
Estouro do timer 3	TMR3IF	TMR3IE	TMR3IP
Interrupção do módulo CCP2	CCP2IF	CCP2IE	CCP2IP

Interrupção – Programação

- O compilador XC8 permite o trabalho com interrupções através da diretiva ***__interrupt()***;
- Através dessa diretiva é possível informar ao compilador que uma determinada função é uma RTI (Rotina de Tratamento de Interrupção) e associá-la a interrupção de alta ou baixa prioridade;
- Uma rotina de tratamento de interrupção não pode ter argumentos nem retornar valores;
- Para realizar a troca de valores entre as RTI e o programa principal (ou mesmo entre as RTIs de alta e baixa prioridade) deve-se declarar uma variável global com o qualificador *volatile*.

Interrupção – Programação

- IMPORTANTE:
  - A diretiva ***__interrupt()*** só informa qual função é uma RTI. O processo de identificação da interrupção e zerar o *flag* relativo a cada uma deve ser implementado no código;
  - E a diretiva *__interrupt(low_priority)* se refere as interrupções configuradas com baixa prioridade.

Interrupção – Exemplo

```
#include <xc.h>
#define _XTAL_FREQ   20000000
// CONFIG1L
#pragma config PLLDIV = 1
#pragma config CPUDIV = OSC1_PLL2
#pragma config USBDIV = 1
//// E demais configurações padrão
// *** Protótipos
void __interrupt(low_priority)baixaPrioridade(void);
void __interrupt() altaPrioridade(void);
void Interr_0(void);
void Interr_1(void);
void Interr_2(void);
```

Interrupção – Exemplo

```
// *** Funções
void __interrupt() altaPrioridade(void)
{
   if (INT0IF && INT0IE)
   {
       Interr_0();
   }
}
void  __interrupt(low_priority)baixaPrioridade(void)
{
   if (INT1IF && INT1IE)
   {
      Interr_1();
   }else if (INT2IF && INT2IE)
   {
      Interr_2();
   }
}
```

Interrupção – Exemplo

```
void Interr_0(void)
{
   INT0IF = 0;
   LATD = LATD^0x01; // inverte o bit 0
}

void Interr_1(void)
{
   INT1IF = 0;
   LATD = LATD^0x02; // inverte o bit 1
}

void Interr_2(void)
{
   INT2IF = 0;
   LATD = LATD^0x04; // inverte o bit 2
}
```

Interrupção – Exemplo

```
void main (void)
{
   TRISB= 0b00000111;
   TRISD = 0;
   LATD = 255;

   INT0IF = 0; // flag da interrupção externa 0
   INT1IF = 0; // flag da interrupção externa 1
   INT2IF = 0; // flag da interrupção externa 2

   INTEDG0 = 0; // transição na descida
   INTEDG1 = 1; // transição na subida
   INTEDG2 = 0; // transição na descida

   INT0IE = 1; // interrupção externa 0 ativada
   INT1IE = 1; // interrupção externa 1 ativada
   INT2IE = 1; // interrupção externa 2 ativada

   /// Prioridade de INT0 sempre em alta
   INT1IP = 0; // prioridade de INT1 baixa
   INT2IP = 0; // prioridade de INT2 baixa

   IPEN = 1; // habilita prioridade
   GIEH = 1; // habilita interrupções de alta prioridade
   GIEL = 1; // habilita interrupções de baixa prioridade

   while (1); // fica em loop sem fazer nada
}
```