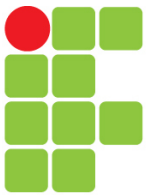
Microcontroladores

Interrupções: externas e temporizadas

Prof. Renan Augusto Starke

Instituto Federal de Santa Catarina – IFSC
Campus Florianópolis
renan.starke@ifsc.edu.br

12 de março de 2020



INSTITUTO FEDERAL
SANTA CATARINA

Ministério da Educação
Secretaria de Educação Profissional e Tecnológica
INSTITUTO FEDERAL DE SANTA CATARINA

Tópicos

Tópico

Interrupções

- Alteração do fluxo de execução (*software*):
  - Instruções de *branch* – laços e desvio condicionais.
  - Instruções de saltos incondicionais – *jmp*
  - Chamada de funções – *call*
  - Retorno de funções – *ret*.
- Alteração do fluxo de execução (*hardware*)
  - Exceção: divisão por 0, instrução inválida, falha de hardware.
  - *Reset* externo: pino ou energização.
  - **Interrupções: periférico solicitando tempo de CPU.**

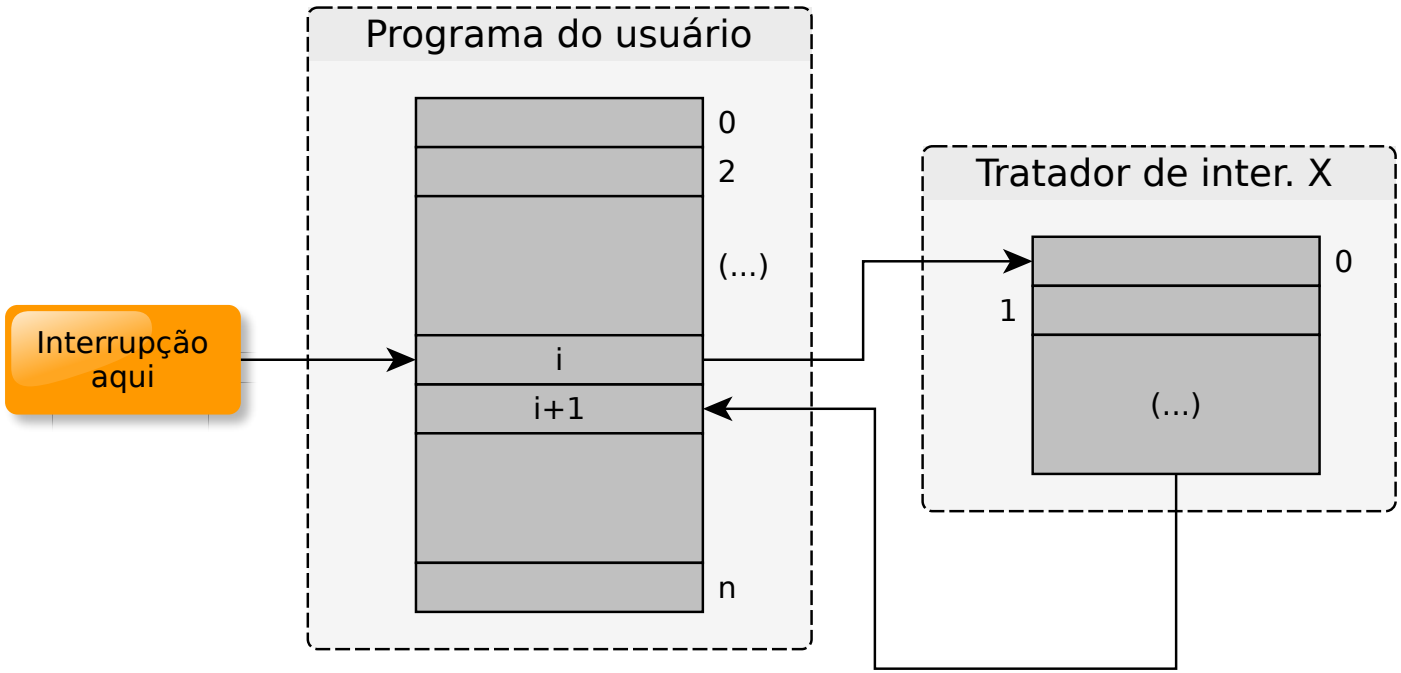
Interrupções

- Exceções e interrupções:
  1. O periférico faz um pedido de interrupção ao processador ou acontece uma exceção.
  2. Suspende-se a execução do código atual em execução.
  3. Executa-se a rotina do serviço de interrupção (ISR) ou recuperação da exceção (se for recuperável).
  4. Retorna-se a executar o código suspenso previamente.

As interrupções são muito importantes porque permitem que diferentes serviços (ou periféricos) executem "paralelamente", interrompendo o processador somente quando necessário.

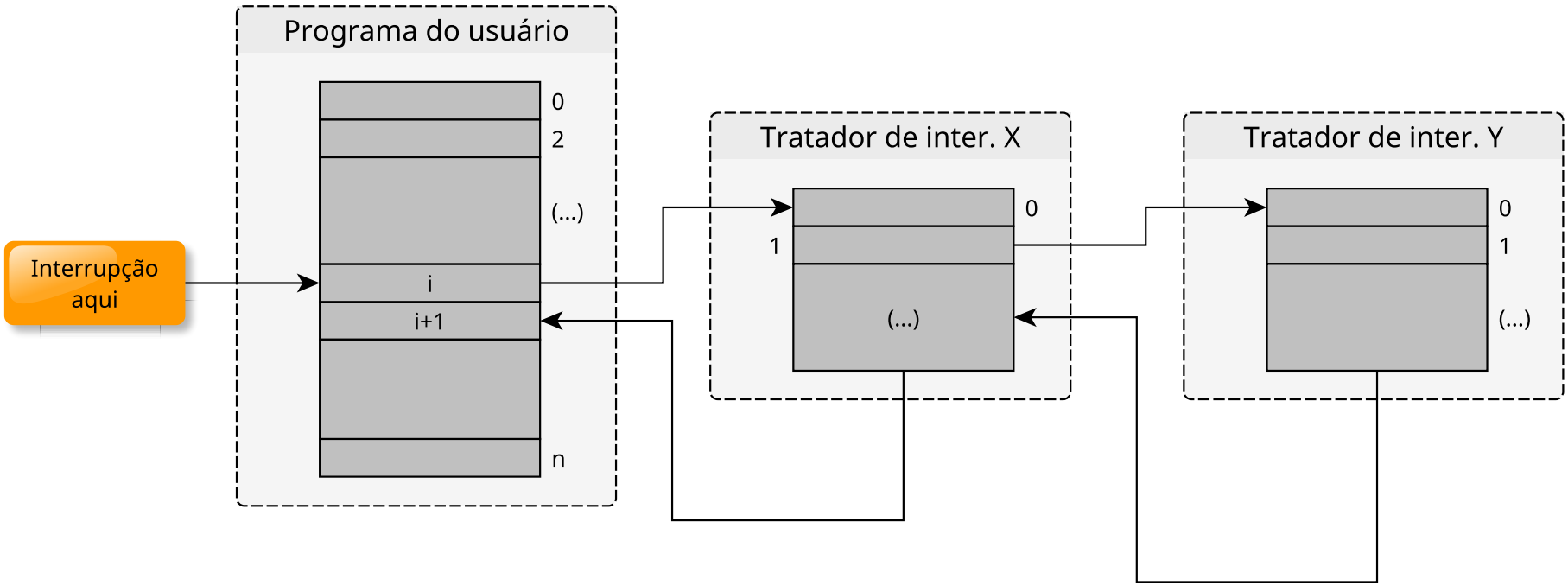
Interrupções

- Interrupções são desvios assíncronos e aperiódicos.
- Tratadores de interrupções são funções "chamadas" por um hardware externo.
- Dependendo do processador, existe um vetor de interrupção que mapeia o endereço de transferência de controle para cada tipo de interrupção.



Interrupções

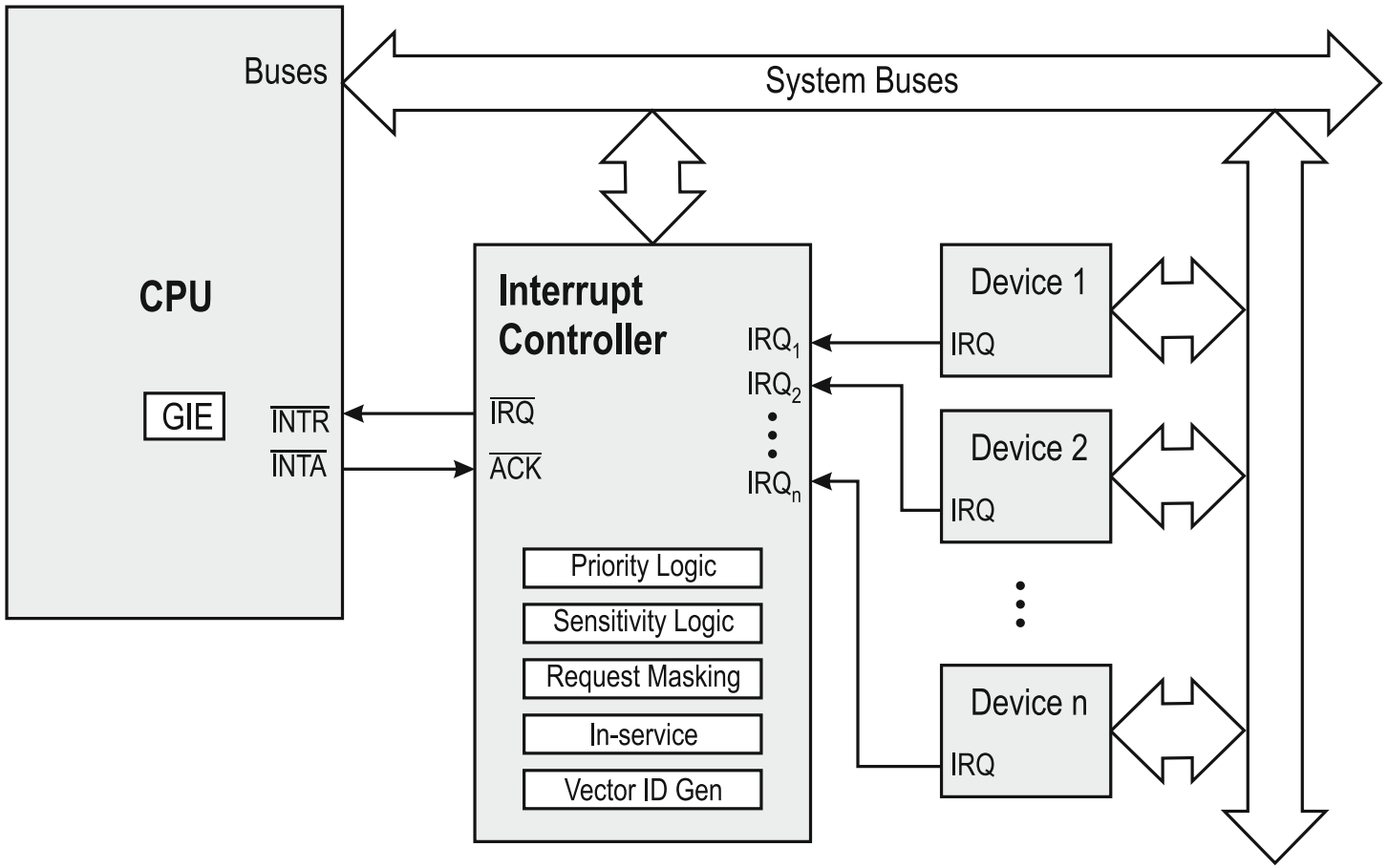
- Interrupções/exceções podem acontecer enquanto está lidando-se com outra interrupção.



Interrupções

- Interrupções são usadas para:
  - Tarefas urgentes de prioridade mais alta que o código principal.
  - Tarefas infrequentes evitando-se o *polling overhead*.
  - Chamada para o sistema operacional (*software interrupt*).
- Programação orientada a eventos:
  - Fluxo do programa é determinado por eventos - sensores ou ações do usuário (clicks, teclas) ou menssagens entre programas/*threads*.
  - Aplicação tem um laço principal detectando e manipulando eventos

Interrupções



Tópico

Interrupções no MSP430

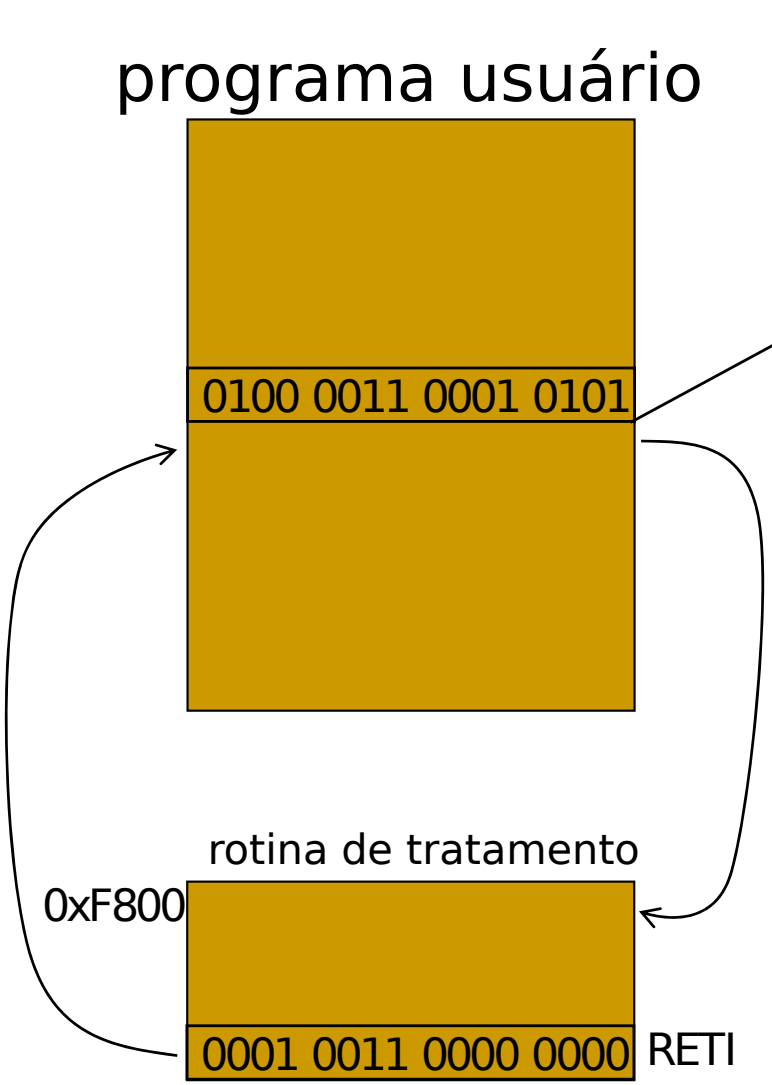
- As interrupções são vetorizadas.
- Vetor possue endereço fixo, mas a implementação pode ficar em qualquer lugar.
- Possuem prioridades.
- São habilitadas e desabilitadas através de registradores.
- Interupções podem ser mascaradas: habilitadas e/ou desabilitadas.

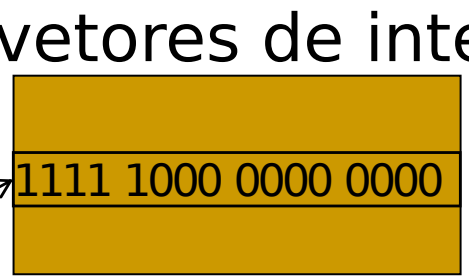
Reset possui a maior prioridade.

Interrupções no MSP430

Memory Address		Description	Access
End:	0FFFFh	**Interrupt Vector Table**	Word/Byte
Start:	0FFE0h		
End:	0FFDFh	**Flash/ROM**	Word/Byte
Start *:	0F800h		
	01100h		
End *:	010FFh	**Information Memory** (Flash devices only)	Word/Byte
	0107Fh		
Start:	01000h		
End:	0FFFh	**Boot Memory** (Flash devices only)	Word/Byte
Start:	0C00h		
End *:	09FFh	**RAM**	Word/Byte
	027Fh		
Start:	0200h		
End:	01FFh	**16-bit Peripheral modules**	Word
Start:	0100h		
End:	00FFh	**8-bit Peripheral modules**	Byte
Start:	0010h		
End:	000Fh	**Special Function Registers**	Byte
Start:	0000h		

Interrupções no MSP430





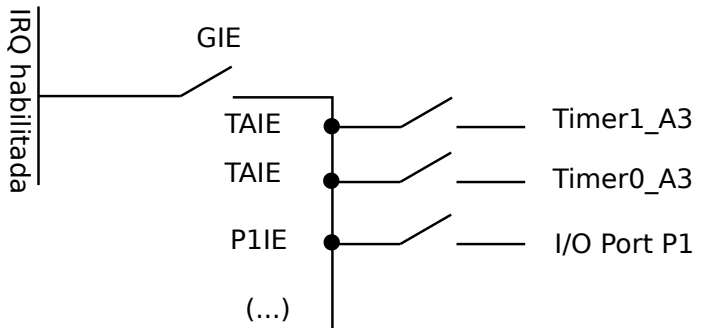
1. Busca do endereço inicial da rotina no vetor.
2. Salva contexto (PC e stack pointer na pilha)
3. Salta para a rotina de tratamento da interrupção
4. Restaura o contexto anterior.
5. Retorna ao programa do usuário **(RETI: obtem o endereço anterior da pilha)**

Interrupções no MSP430

INTERRUPT SOURCE	INTERRUPT FLAG	SYSTEM INTERRUPT	WORD ADDRESS	PRIORITY
Power-Up External Reset Watchdog Timer+ Flash key violation PC out-of-range(1)	PORIFG RSTIFG WDTIFG KEYV(2)	Reset	0FFFEh	31, highest
NMI Oscillator fault Flash memory access violation	NMIIFG OFIFG ACCVIFG(2)(3)	(non)-maskable (non)-maskable (non)-maskable	0FFFCh	30
Timer1_A3	TA1CCR0 CCIFG(4)	maskable	0FFFAh	29
Timer1_A3	TA1CCR2 TA1CCR1 CCIFG, TAIFG(2)(4)	maskable	0FFF8h	28
Comparator_A+	CAIFG(4)	maskable	0FFF6h	27
Watchdog Timer+	WDTIFG	maskable	0FFF4h	26
Timer0_A3	TA0CCR0 CCIFG(4)	maskable	0FFF2h	25
Timer0_A3	TA0CCR2 TA0CCR1 CCIFG, TAIFG (5)(4)	maskable	0FFF0h	24
USCI_A0/USCI_B0 receive USCI_B0 I2C status	UCA0RXIFG, UCB0RXIFG(2)(5)	maskable	0FFEEh	23
USCI_A0/USCI_B0 transmit USCI_B0 I2C receive/transmit	UCA0TXIFG, UCB0TXIFG(2)(6)	maskable	0FFECh	22
ADC10 (MSP430G2x53 only)	ADC10IFG(4)	maskable	0FFEAh	21
			0FFE8h	20
I/O Port P2 (up to eight flags)	P2IFG.0 to P2IFG.7(2)(4)	maskable	0FFE6h	19
I/O Port P1 (up to eight flags)	P1IFG.0 to P1IFG.7(2)(4)	maskable	0FFE4h	18
			0FFE2h	17
			0FFE0h	16
See (7)			0FFDEh	15
See (8)			0FFDEh to 0FFC0h	14 to 0, lowest

Interrupções no MSP430

- Habilitação/desabilitação geral:
  - **__bis_SR_register(GIE)**;
  - **__bic_SR_register(GIE)**;
- Habilitação individual
  - Por periférico.



Status Register (SR)

15	9	8	7							0
Reserved		V	SCG1	SCG0	OSC OFF	CPU OFF	GIE	N	Z	C
				rw-0						

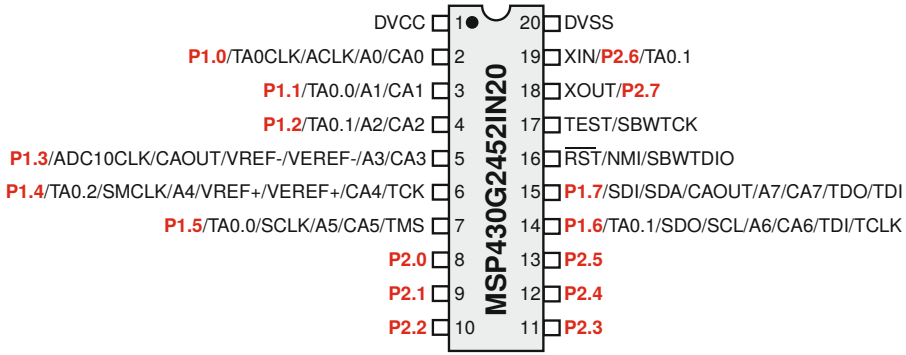
GIE	General interrupt enable. This bit, when set, enables maskable interrupts. When reset, all maskable interrupts are disabled.

Tópico

Interrupções externas

Desviam o fluxo de execução devido à uma **mudança de estado** em um pino específico.

- **Quase** todo pino de uma porta **Px** pode ser habilitado para interrupção externa (**sempre observar o datasheet específico**).
- Interrupções são configuradas pelos registradores PxIE e PxIES.
- Há um vetor para cada porta. Todos as IRQ de **P1** desviam para **um vetor específico de P1**.
- Caso há mais de uma IRQ externa na mesma porta, deve-se testar a fonte.



Interrupções externas – função main e energia

Técnica de polling

```
int main(void)
{
    (...)

    /* Laço infinito */
    while(1){

        if (TST_BIT(P1IN, BUTTON))
            CLR_BIT(P1OUT, LED_1 | LED_2);
        else
            SET_BIT(P1OUT, LED_1 | LED_2);
    }

    return 0;
}
```

Orientado a Interrupções

```
void main(){
    /* Configuração de hardware */
    WDTCTL = WDTPW | WDTHOLD;

    /* Configura interupções */
    config_ext_irq();

    /* Habilita IRQs e desliga CPU */
    __bis_SR_register(LPM4_bits | GIE);
}

/* Port 1 ISR (interrupt service routine) */
#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void)
{
    /* Código da ISR aqui */


    /* Limpa sinal de IRQ do bit xyz  */
    P1IFG &= ~BIT_xyz
}
```

modo baixa energia

Global interrupt enabble

config. ex P2.4

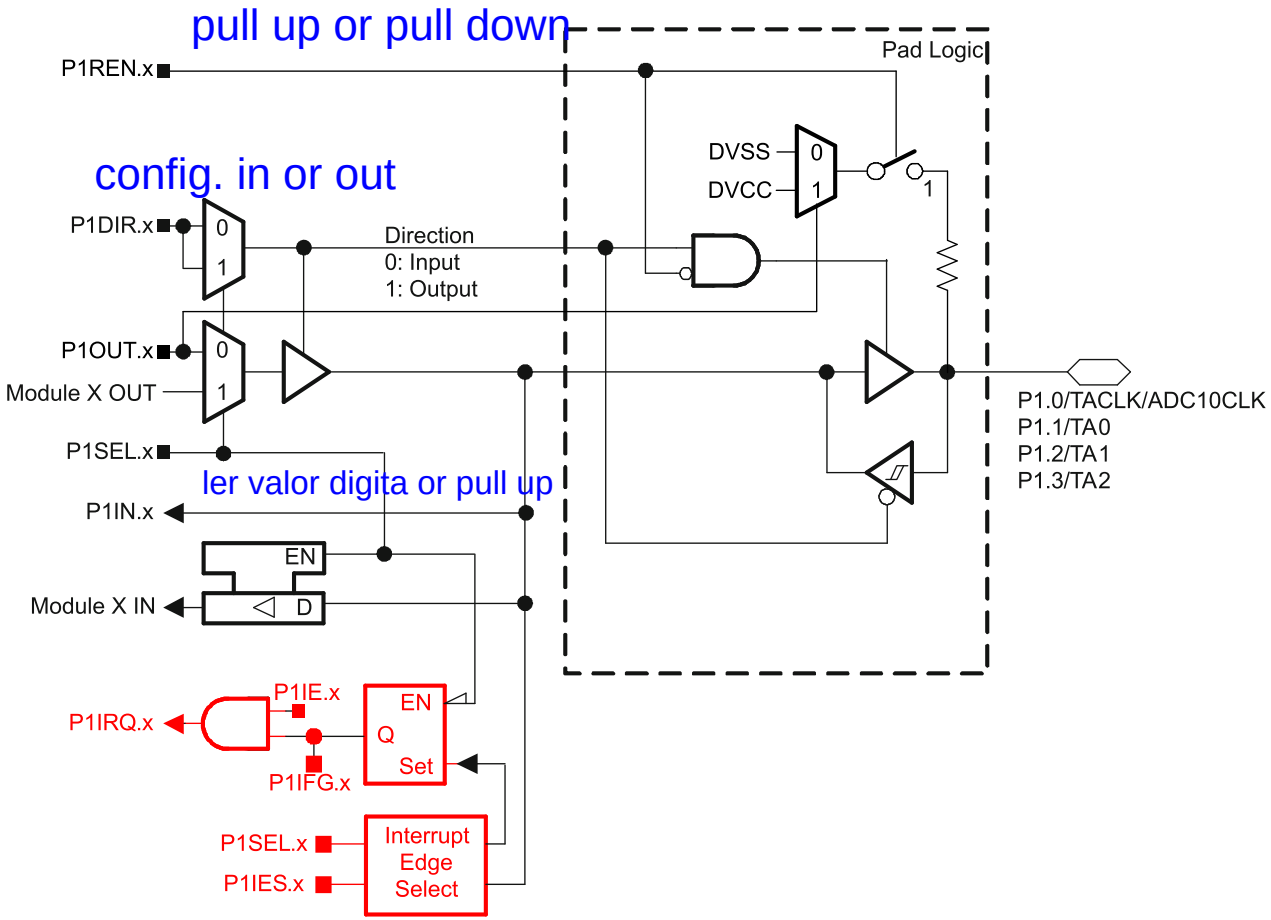
P2IE = bit 4;

ou

config. ex P2.4/p2.5

P2IE = bit 4 | bit 5;

Interrupções externas – configuração



Interrupções externas – configuração

8.4.12 PxIES Register

Port x Interrupt Edge Select Register

Figure 8-12. PxIES Register

7	6	5	4	3	2	1	0
PxIES							
rw	rw	rw	rw	rw	rw	rw	rw

Table 8-16. PxIES Register Description

Bit	Field	Type	Reset	Description
7-0	PxIES	RW	Undefined	Port x interrupt edge select 0b = PxIFG flag is set with a low-to-high transition 1b = PxIFG flag is set with a high-to-low transition

8.4.13 PxIE Register

Port x Interrupt Enable Register

Figure 8-13. PxIE Register

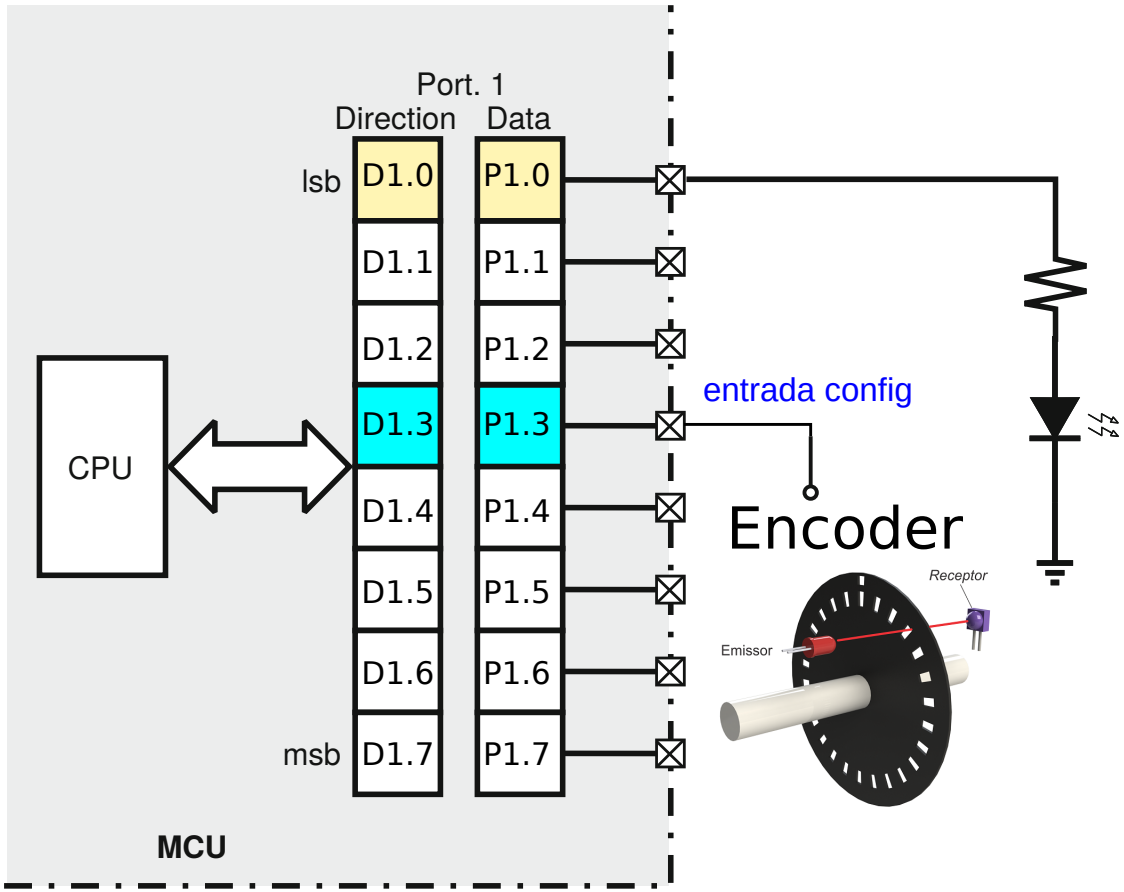
7	6	5	4	3	2	1	0
PxIE							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 8-17. PxIE Register Description

Bit	Field	Type	Reset	Description
7-0	PxIE	RW	0h	Port x interrupt enable 0b = Corresponding port interrupt disabled 1b = Corresponding port interrupt enabled

Interrupções externas – Exemplo

- Contador de pulsos de um encoder para medir deslocamento de um motor.



Interrupções externas – Exemplo

```c
#define LED_DEBUG   BIT0
#define ENCODER_INPUT  BIT3

volatile uint16_t pulses = 0;

void config_ext_irq(){
    /* Primeiramente configura porta: LED como saída.
     * Demais pinos como entraada  */
    P1DIR = LED_DEBUG;

    /* Pull up/down */  habilita pra pull up or down
    P1REN = ENCODER_INPUT;

    /* Pull up */
    P1OUT = ENCODER_INPUT;

    /* Habilitação da IRQ apenas para encoder */
    P1IE =  ENCODER_INPUT;

    /* Transição de nível alto para baixo */
    P1IES = ENCODER_INPUT;

    /* Limpa alguma IRQ pendente */
    P1IFG &= ~ENCODER_INPUT;
}
```

```c
void main(){
    /* Configuração de hardware */
    WDTCTL = WDTPW | WDTHOLD;

    /* Configura interupções */
    config_ext_irq();

    /* Habilita IRQs e desliga CPU */
    __bis_SR_register(LPM4_bits | GIE);
}

/* Port 1 ISR (interrupt service routine) */
#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void)
{
    /* Liga/desliga LED quando detectado
     * borda no encoder */
    P1OUT ^= LED_DEBUG;

    /* Conta o número de pulsos */
    pulses++;

    /* Limpa sinal de IRQ do botão 0 */
    P1IFG &= ~ENCODER_INPUT;
}
```