SISTEMAS MICROPROCESSADOS I

Prof.: João Castelo

CENTRO UNIVERSITÁRIO
SENAI CIMATEC
SENAI CIMATEC SISTEMA FIEB
Federação das Indústrias do Estado da Bahia

Portas de E/S - IO

- São os pinos de entrada e saída do microcontrolador;
- São responsáveis pela comunicação do microcontrolador com o mundo externo;
- No MSP430 cada pino possui mais de uma função, isto é, o pino fornece acesso a diversos periféricos do microcontrolador.
- O acesso a estas funções se dá através da multiplexação dos pino;
- A família MSP430 possui até 8 portas de entrada e saída;
- Cada porta possui até 8 pinos de entrada/saída;
- Cada pino pode ser individualmente configurado como entrada ou saída;

Portas de E/S - IO

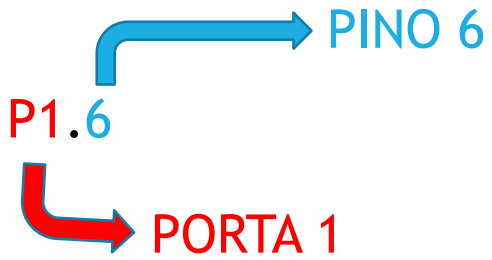
- O padrão de nomenclatura deste registradores é:
  - PxYYY, onde x indica a porta e YYY indicam a função do registrador;
- Exemplo:

P1OUT→ Registrador de saída

└▶PORT 1

Portas de E/S - IO

- A nomenclatura dos pinos de E/S do MSP430 segue o seguinte padrão:
  PX.Y: onde X indica qual é a porta e Y qual é o pino desta porta.



- Cada porta contém um conjunto básico de registradores os quais controlam sua operação

Registradores básicos de E/S

- PxDIR (registrador de direção): Define se é pino de entrada ou se é pino de saída;
  Bit = 0 → o pino é uma entrada;
  Bit = 1 → o pino é uma saída;

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Px.7	Px.6	Px.5	Px.4	Px.3	Px.2	Px.1	Px.0

Registradores básicos de E/S

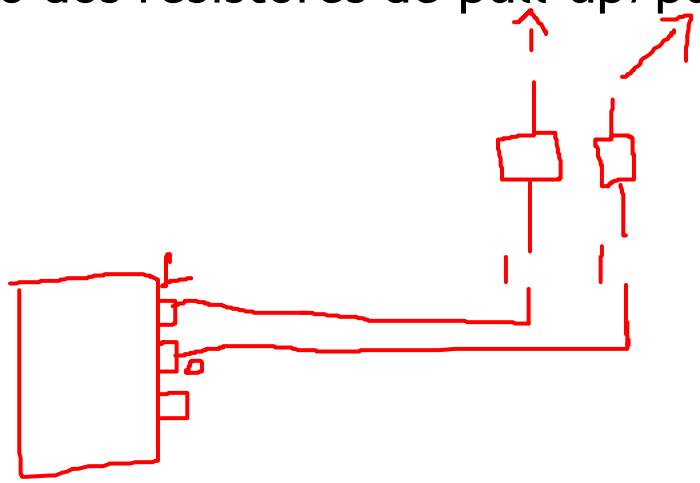
- PxIN: registrador de entrada para a leitura do estado dos pinos da porta
- PxOUT (registrador de saída): contém o valor do pino correspondente, quando o pino está configurado como saída;
- PxSEL:registrador para a seleção do pino, que seleciona entre a função de E/S normal e a função eventualmente multiplexada ao pino
  - PxSEL = 1 => função alternativa para o respectivo pino;
  - PxSEL = 0 => seleciona a função normal

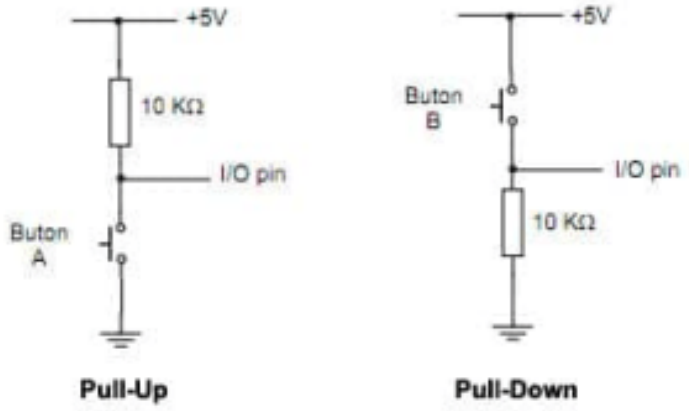
Registradores específicos de E/S

- São registradores paras as portas 1 e 2:
  - PxIFG: registrador sinalizador de interrupção do pino – responsável pela sinalização de uma mudança de estado em um dos pinos da porta.
  - PxIES: registrador de seleção de borda de sensibilidade de cada pino – utilizado para controlar a borda de sensibilidade à interrupção da porta.
    - PxIES = 0 => borda de subida
    - PxIES = 1 => borda de descida
  - PxIE: registrador de habilitação individual de interrupção para cada pino da porta – responsável pela habilitação das interrupções nas portas.

Registrador – Família 2xx

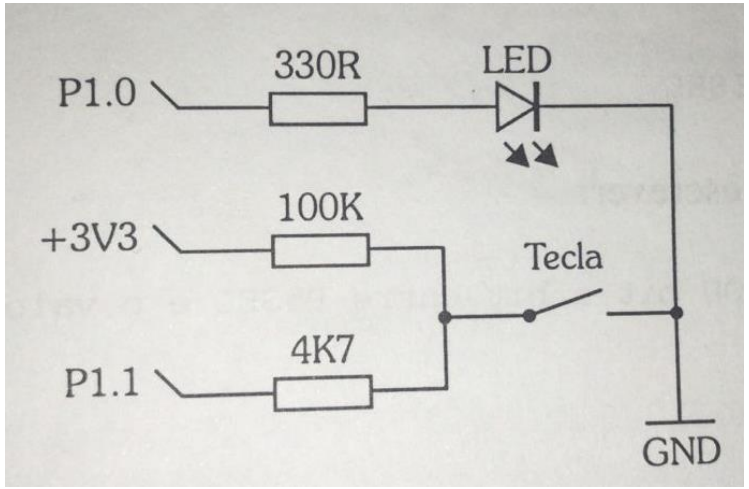
- PxREN: utilizado para habilitação dos resistores de pull-up/pull-down





Exemplo

- Acionar um LED conectado ao pino PI.0 em função de um chave conectada ao pino PI.1.



Exemplo

```
#include <msp430x14x.h>
        NAME    main
        PUBLIC  main
        ORG     0FFFEh
        DC16    main
        RSEG    CODE
main    MOV       #0x3FF,SP    ; inicializa o apontador da pilha
        MOV       #WDTPW+WDTHOLD,WDTCTL ; desliga o watchdog
        BIS.B     #1,P1DIR     ; configura o pino P1.0 como saída
loop    BIT.B     #2,P1IN      ; testa o pino p1.1
        JZ        acende       ; desvia se P1.1 = 0
        BIC.B     #1,P1OUT     ; limpa a saída P1.0
        BR        #loop        ; desvia para o loop
acende  BIS.B     #1,P1OUT     ; seta a saída P1.0
        BR        #loop        ; desvia para o loop
        END       main
```

Acesso aos pinos

- Na linguagem C o acesso a cada pino é feito através de máscaras de bits nos registradores.
- Para colocar em nível lógico 1 o bit fazemos uma operação lógica OR do registrador com um valor que possua bits iguais a 1 nas posições que se deseja setar no registrador e zeros nas demais posições;
- Exemplo: para setar os bit 2, 5 e 6 do registrador P1OUT, fazemos um OR com o valor "01100100b", ou seja, 0x64:

```
P1OUT = P1OUT | 0x64;
```

ou

```
P1OUT |= 0x64;
```

Acesso aos pinos

- Assim, supondo que P1OUT esteja com os bits 0, 4, e 6 setados e os demais em zero:

P1OUT	0	1	0	1	0	0	0	1

- Fazendo um OR com 0x64, temos:

P1OUT	0	1	0	1	0	0	0	1
0x64	0	1	1	0	0	1	0	0
Novo P1OUT	0	1	1	1	0	1	0	1

- No fim, os bits 2 e 5 foram setados e os demais permaneceram inalterados, inclusive o bit 6 que já estava setado.

Acesso aos pinos

- Para zerar um bit fazemos uma operação lógica AND do registrador com um valor que possua bits iguais a 0 nas posições que se deseja zerar no registrador e uns nas demais posições;
- Exemplo: para zerar os bit 2, 5 e 6 do registrador P1OUT fazemos um AND de

```
P1OUT = P1OUT & 0x9B;
```

ou

```
P1OUT &= 0x9B;
```

Acesso aos pinos

- Assim, supondo que P1OUT esteja com os bits 0, 4, e 6 setados e os demais em zero:

P1OUT	1	0	0	1	1	1	1	1

- Fazendo um AND com 0x9B, temos:

P1OUT	0	1	0	1	0	0	1	1
0x9B	1	0	0	1	1	1	0	1
Novo P1OUT	0	**0**	**0**	1	0	0	**0**	1

- No fim, os bits 1 e 6 foram zerados e os demais permaneceram inalterados, inclusive o bit 5 que já estava em zero.

Acesso aos pinos

- Os arquivos .h dos diversos MSP430 possuem constantes definidas para cada bit individualmente:

```
#define BIT0 (0x0001u)
#define BIT1 (0x0002u)
#define BIT2 (0x0004u)
#define BIT3 (0x0008u)
#define BIT4 (0x0010u)
#define BIT5 (0x0020u)
#define BIT6 (0x0040u)
#define BIT7 (0x0080u)
```

```
#define BIT8 (0x0100u)
#define BIT9 (0x0200u)
#define BITA (0x0400u)
#define BITB (0x0800u)
#define BITC (0x1000u)
#define BITD (0x2000u)
#define BITE (0x4000u)
#define BITF (0x8000u)
```

Acesso aos pinos

- Assim, para ficar mais bem documentado, podemos usar BIT6 + BIT5 + BIT2 no lugar de 0x64
- Como 0x9B é 0x64, para zerar os bits com AND usamos ~(BIT6 + BIT5 + BIT2) no lugar de 0x9B
  Ficando assim:

```
P1OUT |= BIT6 + BIT5 + BIT2;
P1OUT &= ~(BIT6 + BIT5 + BIT2);
```

PxDIR

Exemplo:

- Definindo os pinos P2.0 e P2.1 como saída:
  P2DIR |= BIT0 + BIT1; //Define P2.0 e P2.1 como saída
- Definindo o pino P1.7 como entrada:
  P1DIR &= ~BIT7; // define o pino P1.7 como entrada

PxOUT

Exemplo:

- Colocando o pino P1.0 em '0':
  `P1OUT &= ~BIT0;` // coloca o pino P1.0 em ‘0’
- Colocando o pino P1.6 em '1':
  `P1OUT |= BIT6;` // coloca o pino P1.6 em '1'

PxIN

Exemplo 1: testando se o valor do pino P1.3 é zero:

```
if((P1IN & BIT3) == 0)
{
}
```

Se o resultado for zero, P1.3 é zero.

Faz uma máscara com 00001000b.

Exemplo 2: testando se o valor do pino P1.3 é um:

```
if((P1IN & BIT3) == BIT3)
{
}
```

Se o resultado for 00001000b, P1.3 é um.

Faz uma máscara com 00001000b.

Recomendação Texas

- Os pinos não utilizados devem ser configurados como saída e deixados em aberto, dessa forma reduz o consumo e aumenta a imunidade a ruídos do circuito.
- Após o reset, os pinos são sempre configurados como entradas e com a função alternativa desativadas.

Exemplo

```
#include <io430x14x.h>

#define led P1OUT_bit.P1OUT_0

int main( void )
{
  WDTCTL = WDTPW + WDTHOLD; // desativa o watchdog
  P1DIR_bit.P1DIR_0 = 1;    // configura o pino P1.0 como saída
  while (1)
  {
    // lê o estado do pino P1.1
    // caso esteja setado, apaga o LED, caso esteja em zero, liga o LED
    if (P1IN_bit.P1IN_1) LED =0; else LED =1;
  }
}
```

Exercícios

1. Implemente o programa exemplo no IAR.
2. Faça o circuito do programa exemplo no Proteus. (OPCIONAL)
3. Gere o arquivo hexadecimal do programa exemplo e o simule no Proteus.
4. Faça um programa em C para que o LED vermelho, ligado ao pino P1.0, e o LED verde, ligado ao pino P1.6, mudem de estado, alternadamente, cada vez que o botão de uso geral (ligado ao pino P1.3) for pressionado usando interrupção.

Prática para entregar ao professor: Questão 4

Forma de entrega: Ao final da aula os alunos devem mostrar ao professor o circuito funcionando no MSP430

DATA DE ENTREGA: 26/04/2022

Referências

- Notas de Aula. Prof. Gabriel Kovalhuk. UTFPR.