Microcontroladores

Sistemas Digitais Microprocessados (SDM)

E/S digitais

Profa. Ana T. Y. Watanabe

atywata@gmail.com.br

Microcontroladores

“Mas os que esperam no SENHOR renovarão as forças, subirão com asas como águias; correrão, e não se cansarão; caminharão, e não se fatigarão.” Isaías 40:31

AGENDA DO DIA

- **ATMEGA328-P – Portas Digitais**
- **Exercício de E/S digitais**

INSTRUÇÕES

PORTAS DE ENTRADAS E SAIDAS (I/Os)

*** Possui 3 conjuntos de pinos de I/Os:**

- **PORTB (PB7...PB0)**
- **PORTC (PC6...PC0)**
- **PORTD (PD7..PD0)**

*** Todos os pinos tem a função Lê – Modifica – Escreve, ou seja, cada bit pode ser alterado individualmente.**

*** A capacidade para drenar ou suprir corrente por pino é 20mA. Cada PORT 100mA e o componente 200mA.**

ATMEGA 328-P – PORTAS DIGITAIS

- ATMEGA 328-P tem pinos de I/O digitais:
- **PORTx:** Registrador de dados, usado para escrever nos pinos de PORTx;
- **DDRx:** Registrador de direção, usado para definir se os pinos do PORTx são de entradas ou saídas;
- **PINx:** Registrador de entrada, usado para ler o conteúdo dos pinos do PORTx.
-

ATMEGA 328-P – PORTAS DIGITAIS

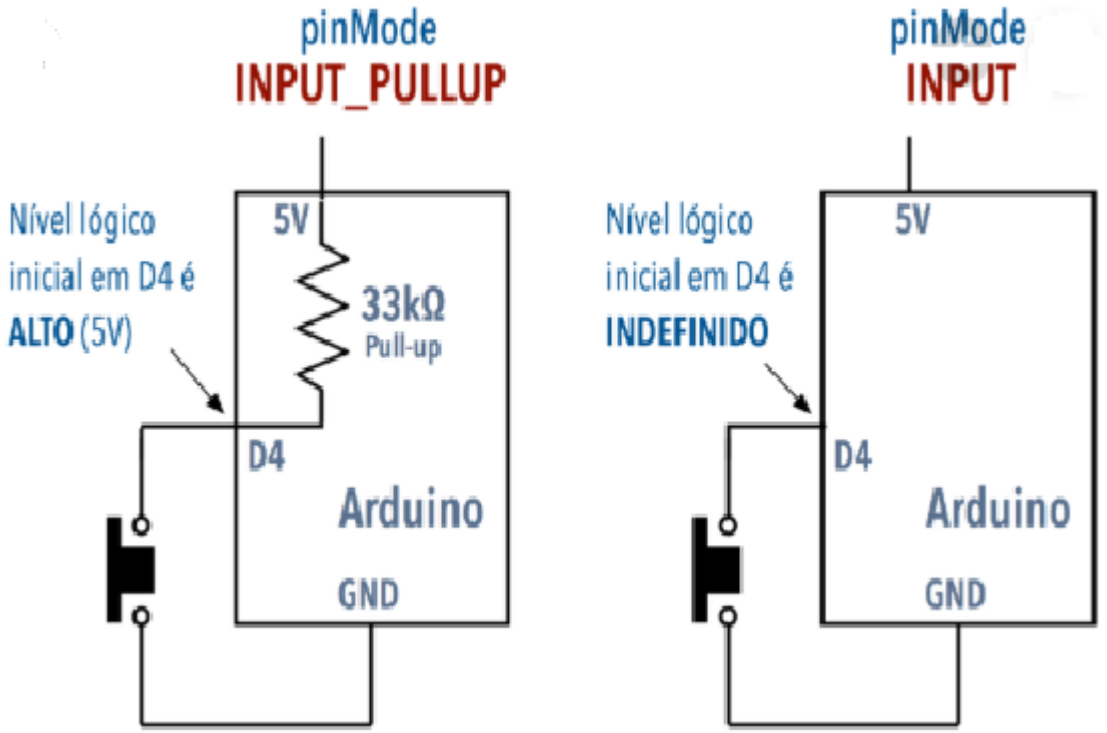
Primeiramente, deve-se definir se é entrada ou saída, escrevendo no registrador DDRx;

- DDRx = 1(saída): A escrita no bit do registrador PORTx alterará o estado do pino;

- DDRx = 0(entrada): A escrita de 1(set_bit) no bit registrador PORTx habilitará o pull-up interno e a leitura do estado lógico do pino deve ser lido do registrador PINx.

Qual a necessidade de um pull-up??

Resistores de pull-up (Pull-down) garantem estado inicial definido quando a chave estiver aberta.



ATMEGA 328-P – PORTAS DIGITAIS

- Os **módulos periféricos têm prioridade sobre o E/S digitais** de modo que quando um periférico está habilitado, as funções E/S digitais associadas são desabilitadas;

- Após um *reset* , as funções periféricas são desabilitadas, tendo os pinos de E/S digitais habilitadas;

ATMEGA 328-P – PORTAS DIGITAIS

Exemplo:

LIGANDO/PISCANDO UM LED

```
//                                        pisca_led.c

//-------------------------------------------------------------------
```

```
#include <avr/io.h> //definições do componente especificado

#include <util/delay.h>   //para incluir rotina _delay_ms()
*define LED  PB5

//Definições de macros - empregadas para o trabalho com os bits
#define set_bit(Y,bit_x) (Y|=(1<<bit_x))  //ativa o bit x da variável Y  (coloca em 1)
 #define clr_bit(Y,bit_x) (Y&=~(1<<bit_x))   // limpa o bit x da variável Y (coloca em 0)

#define tst_bit(Y,bit_x) (Y&(1<<bit_x))      // testa bit x da variável Y (retorna 0 ou 1)

# define cpl_bit(Y,bit_x) ) (Y^=(1<<bit_x)) // troca o  bit x da variável Y (complementa)
```

// pisca_led.c (continuação)

```
int main(void)
{
// configuração
DDRB = 0xff;  // Configura todos os pinos do PORTB como saídas
//inicialização
PORTB =0xff;  // Apaga todos os leds
while(1) //laço infinito
  {
   set_bit(PORTB,LED); //desliga LED
   _delay_ms(500);
  clr_bit(PORTB,LED); //liga LED
  _delay_ms(500);
  }
}
```

#define set_bit(Y,bit_x) (Y|=(1<<bit_x))
//ativa o bit x da variável Y (coloca em 1)

Página 98 do livro AVR

Ex.: set_bit(PORTD,5)

Y = PORTD e bit_x = 5

Y |= (1<<bit_x) ou Y = Y | (1<<bit_x)

PORTD = PORTD | (1<<5)

0b xxxx xxxx | PORTD x pode ser 0 ou 1
0b 0010 0000 (1 << 5) é a mascara

0b xx1x xxxx o bit 5 com certeza será 1

#define clr_bit(Y,bit_x) (Y&=~(1<<bit_x))
// limpa o bit x da variável Y (coloca em 0)

Página 98 do livro AVR

Ex.: clr_bit(PORTB,2)
Y = PORTB e bit_x = 2

Y&=~(1<<bit_x) ou Y = Y &~(1<<bit_x)
PORTB = PORTB & ~ (1<<2)

0b xxxx xxxx & PORTB x pode ser 0 ou 1
0b 1111 1011 ~ (1 << 2) é a mascara

0b xxxx x0xx o bit 2 com certeza será 0

#define tst_bit(Y,bit_x) (Y&(1<<bit_x))
// testa bit x da variável Y (retorna 0 ou diferente de 0)

Página 100

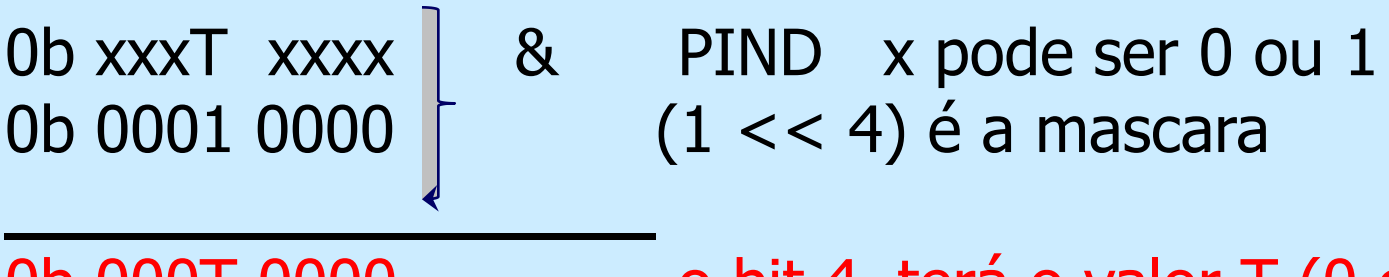
Ex.:
tst_bit(PIND,4)

#define tst_bit(Y,bit_x) (Y&(1<<bit_x))
// testa bit x da variável Y (retorna 0 ou diferente de 0)

Página 100 do livro AVR

Ex.: tst_bit(PIND,4)
Y = PIND e bit_x = 4

Y & (1<<bit_x)
PIND & (1<<4)



```
# define cpl_bit(Y,bit_x) ) (Y^=(1<<bit_x))
// troca o  bit x da variável Y (complementa)
```

Página 99

Ex.:
cpl_bit(PORTC,3)

 # define cpl_bit(Y,bit_x)) (Y^=(1<<bit_x))
// troca o bit x da variável Y (complementa)

Página 99 do livro AVR

Ex.: cpl_bit(PORTC,3)
Y = PORTC e bit_x = 3

Y^=(1<<bit_x) ou Y = Y ^(1<<bit_x)
PORTC = PORTC ^ (1<<3) **OU exclusivo**

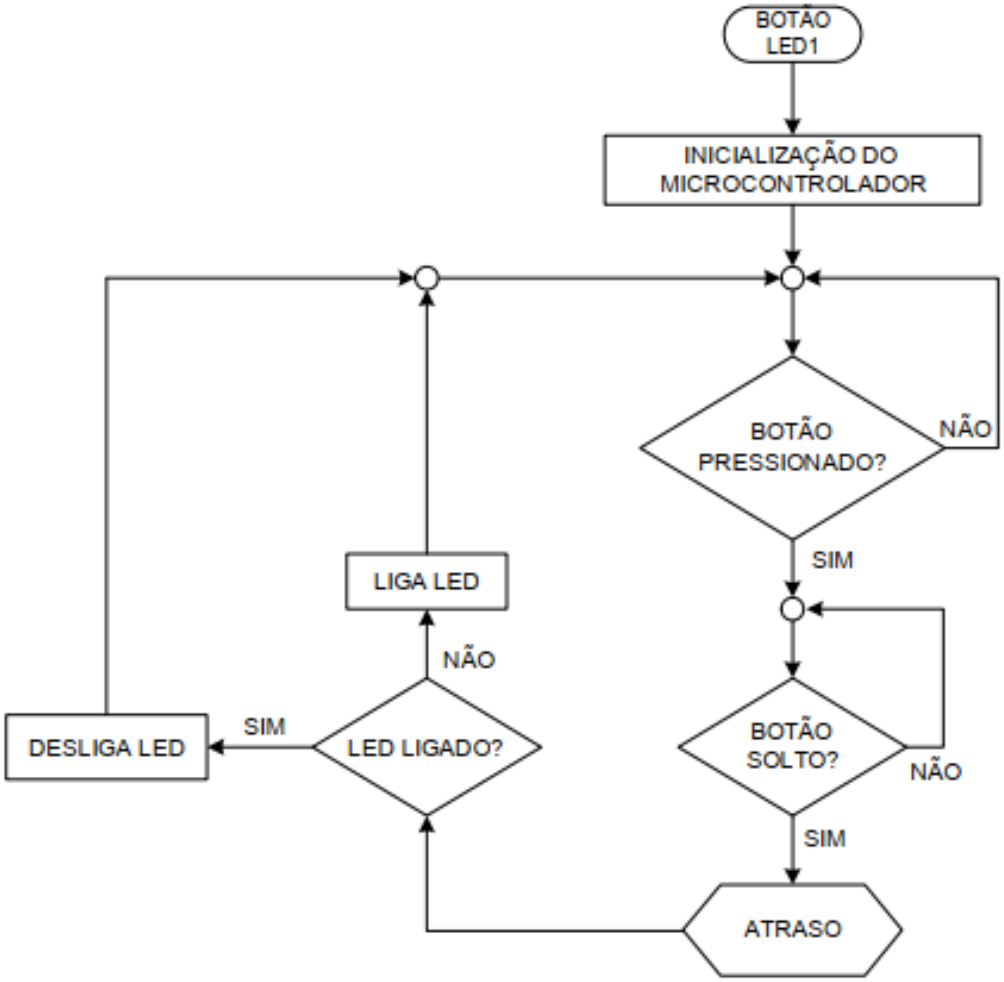
0b xxxx 1xxx ^ PORTC x pode ser 0 ou 1
0b 0000 1000 (1 << 3) é a mascara

0b xxxx 0xxx o bit 3 será 0 se o bit a ser complementado for 1, e 1 se for 0

EXERCÍCIO:

Utilizando a macro para complementar um bit (cpl_bit), faça um programa para piscar um LED (PORTD6) três vezes rapidamente (500ms) e três vezes lentamente (1000ms) num loop eterno.

Leitura de um botão



```
// LIGANDO E DESLIGANDO UM LED QUANDO UM BOTÃO É PRESSIONADO

#define F_CPU 16000000UL

#include <avr/io.h>
#include <util/delay.h>
//Definições de macros - para o trabalho com os bits de uma variável
#define set_bit(Y,bit_x)(Y|=(1<<bit_x))
#define clr_bit(Y,bit_x)(Y&=~(1<<bit_x))
#define cpl_bit(Y,bit_x)(Y^=(1<<bit_x))
#define tst_bit(Y,bit_x)(Y&(1<<bit_x))

#define LED PB3          //LED é o substituto de PB3 na programação
#define BOTAO PC2        //BOTAO é o substituto de PC2 na programação
//-----------------------------------------------------------------
```

```
int main()
{
// configuração
DDRB = 0b00001000; /*Configura o PORTB, PB3(arduino 11) saída, os demais pinos entradas */
DDRC = 0b00000000; /*Configura o PORTC, PC2 (arduino A2) todos pinos são entradas */
PORTC= 0b00000100; //Habilita o pull-up para o BOTAO

// inicialização
PORTB = 0XFF;   // Escreve 1s em todos os pinos – apaga PB3
```

if (x) => se x diferente de 0

if(!x) => se x igual a 0

```
while(1) //laço infinito
  {
  if(!tst_bit(PINC,BOTAO))//se o botão for pressionado executa o if
    {
     while(!tst_bit(PINC,BOTAO))
        ; //fica preso até soltar o botão
     _delay_ms(10); //atraso de 10 ms para eliminar o ruído do botão
     if(tst_bit(PORTB,LED)) //se o LED estiver apagado, liga o LED
        clr_bit(PORTB,LED);
     else //se não apaga o LED
        set_bit(PORTB,LED);
    //o comando cpl_bit(PORTB,LED) pode substituir este laço if-else

    }   //if do botão pressionado
   }    //laço infinito
}
```

Exercício:

Elaborar um programa que um led fique piscando se o botão estiver pressionado. Se não, apague o led. Utilize uma frequência que torne agradável o piscar do LED.