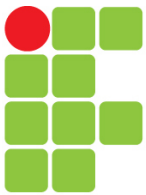
Microcontroladores

Configuração de pinos E/S

Prof. Renan Augusto Starke

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

20 de fevereiro de 2020



INSTITUTO FEDERAL
SANTA CATARINA

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

Linguagem C em microcontroladores

- Programação usando assembly:
  - Controle maior sobre o desempenho.
  - Útil na construção de rotinas críticas (otimização).
  - Aplicação **não** é portável.
  - Grande esforço de programação, exige conhecimento do microcontrolador e do núcleo.
- Programação usando Linguagem C:
  - Redução do tempo de desenvolvimento.
  - O reuso do código é facilitado.
  - Facilidade de manutenção.
  - Aplicação **mais** portável.

Linguagem C em microcontroladores

- Programação usando Linguagem C:
  - Redução do tempo de desenvolvimento.
    - Foco maior no microcontrolador e na aplicação.
  - O reuso do código é facilitado.
    - Algoritmos são portáveis.
- Porém:
  - Desempenho geral da aplicação depende do compilador e otimizações.
  - Fabricante deve disponibilizar arquivos de cabeçalhos para configuração dos periféricos (endereços de registradores e funções otimizadas).
  - Alguns microcontroladores são tão complexos que o fabricante disponibiliza uma **biblioteca** de acesso e configuração de periféricos.

Linguagem C em microcontroladores

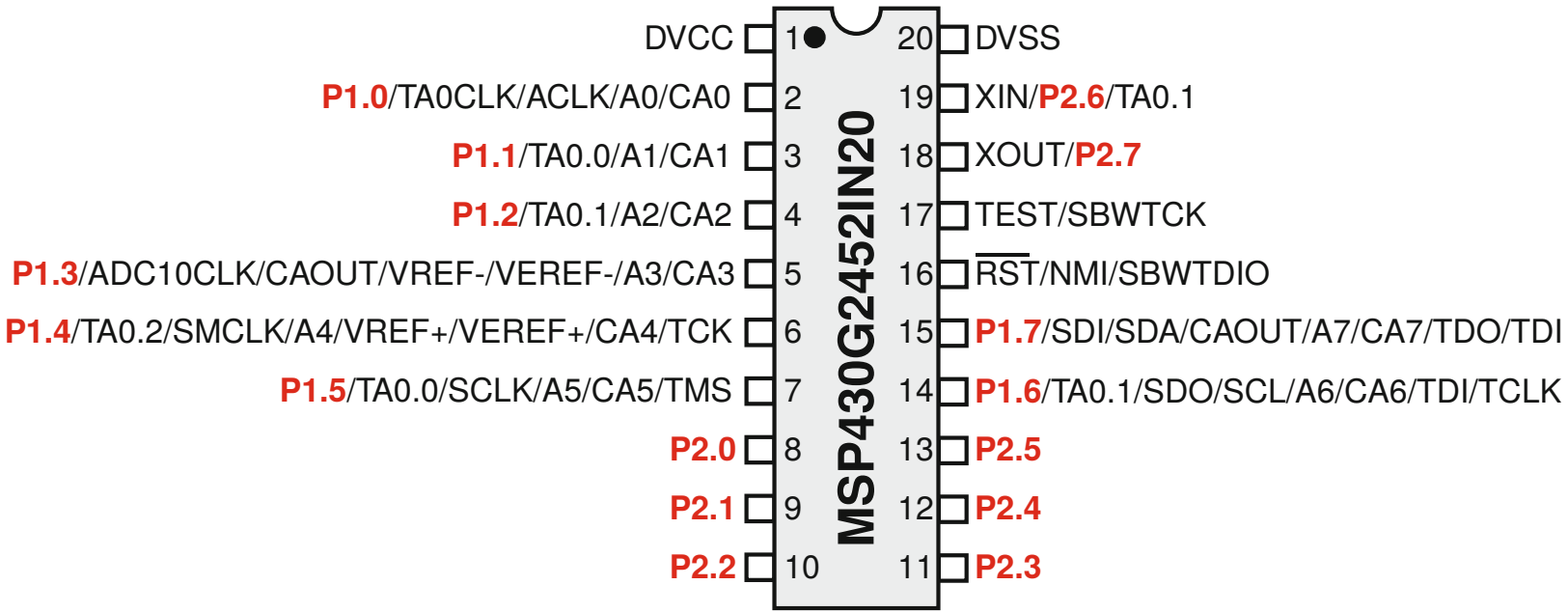
- Comparação código:

```c
#include <msp430.h>

#define LED   BIT0
#define DELAY 5000

int main(void)
{
    int i;

    /* Configuração de hardware */
    WDTCTL = WDTPW | WDTHOLD;
    P1DIR |= BIT0;

    /* main não pode retornar */
    while(1){
        /* Liga/Desliga LED */
        P1OUT = P1OUT ^ LED;

        /* Atraso */
        for (i=DELAY;i--; i > 0);
    }

    return 0;
}
```

```
#include "msp430g2231.h"

LED   EQU 01h
DELAY EQU 5000

#define COUNTER R15

ORG 0f800h

RESET:
        mov.w #0300h, SP
        mov.w #WDTPW+WDTHOLD, &WDTCTL
        bis.b #001h, &P1DIR

main:
        xor.b #LED, &P1OUT
        mov.w #DELAY, COUNTER
L1:
        dec.w COUNTER
        jnz L1
        jmp main


ORG 0FFFEh
DW RESET
END
```

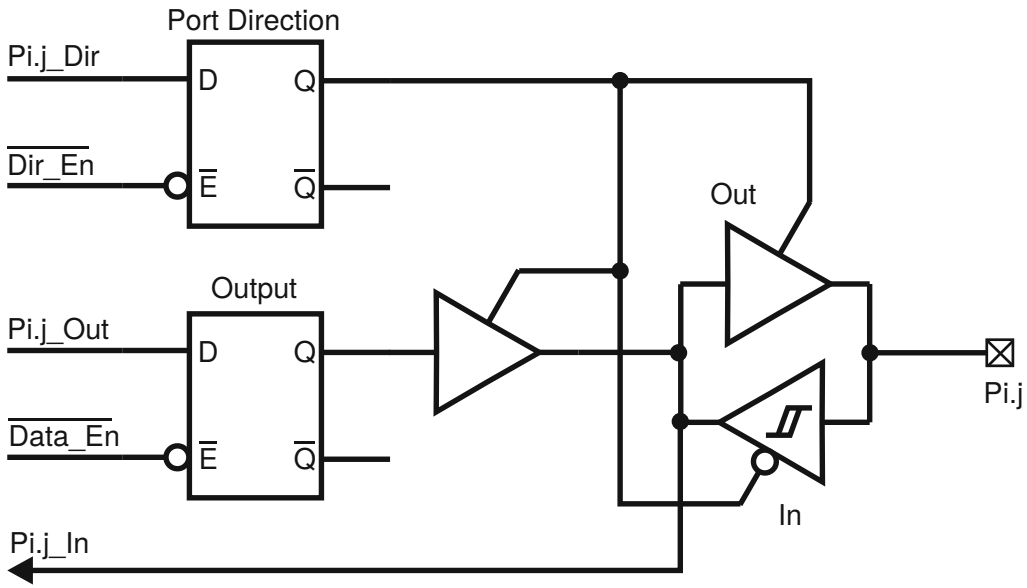
Pinos de entrada e saída

- Pinos de entrada e saída são responsáveis pela interação do microcontrolador com o mundo externo:
  - Acionar um LED.
  - Acionar um motor.
  - Ler um botão.
  - Acionar displays.
  - ...

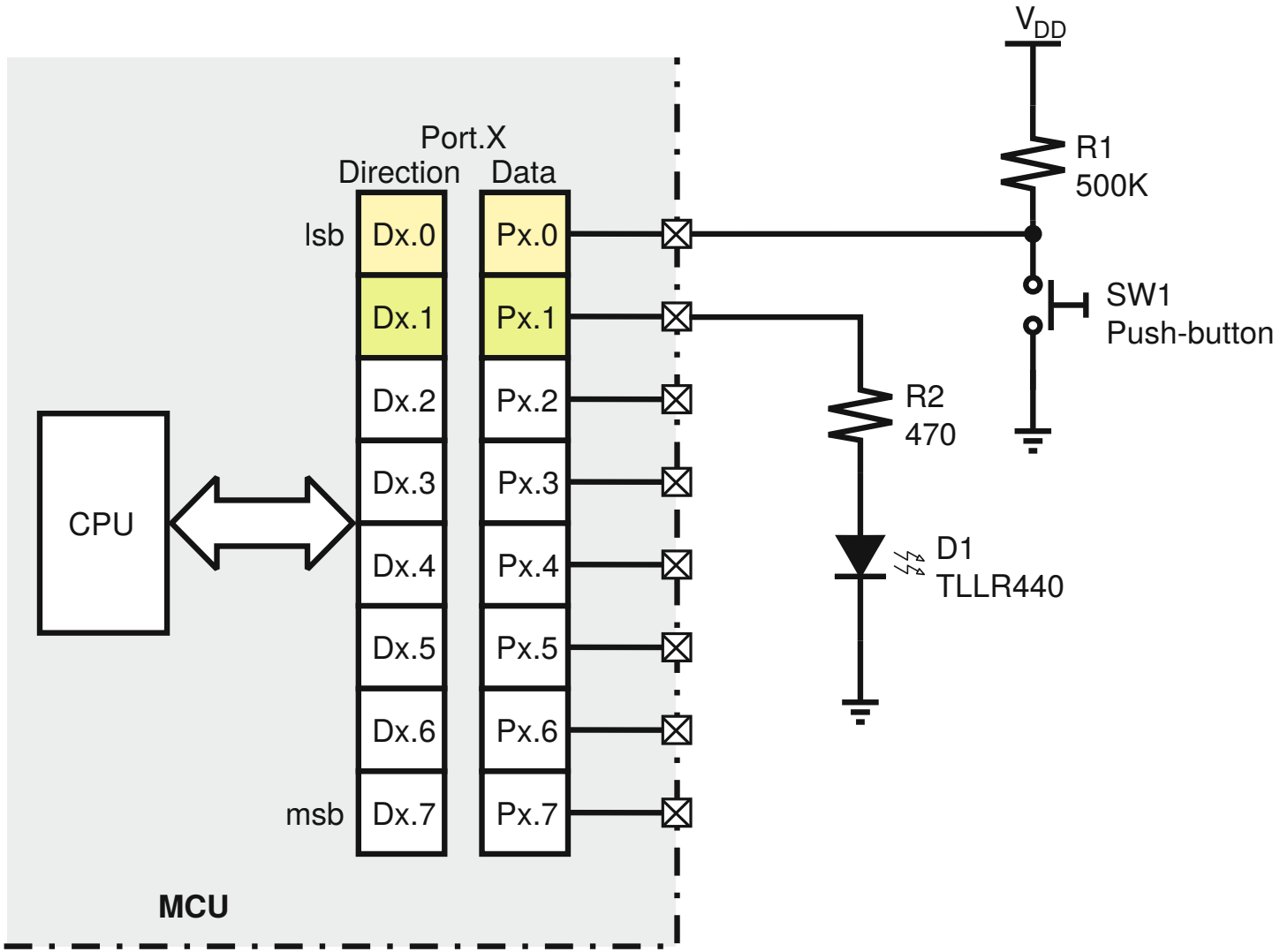


Entradas e saídas digitais: MSP430

- Todas os pinos são "Read-Modify-Write": mudança de configuração de um pino não altera outros.
- Os pinos podem ser configurados como entrada ou saída.
- É possível habilitar resistores internos de pull-up ou pull-down.
- Cada pino pode fornecer/drenar **até 6mA**. A **porta** pode fornecer no máximo **48mA**.

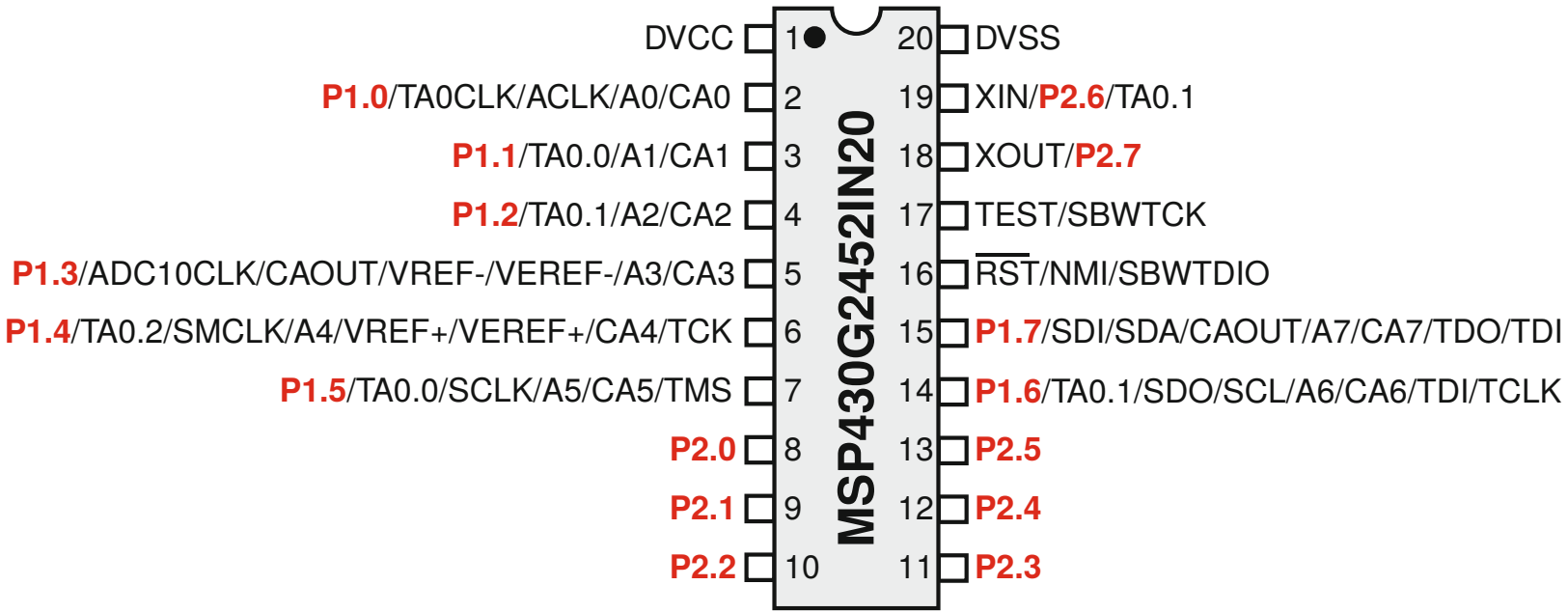


Entradas e saídas digitais: MSP430



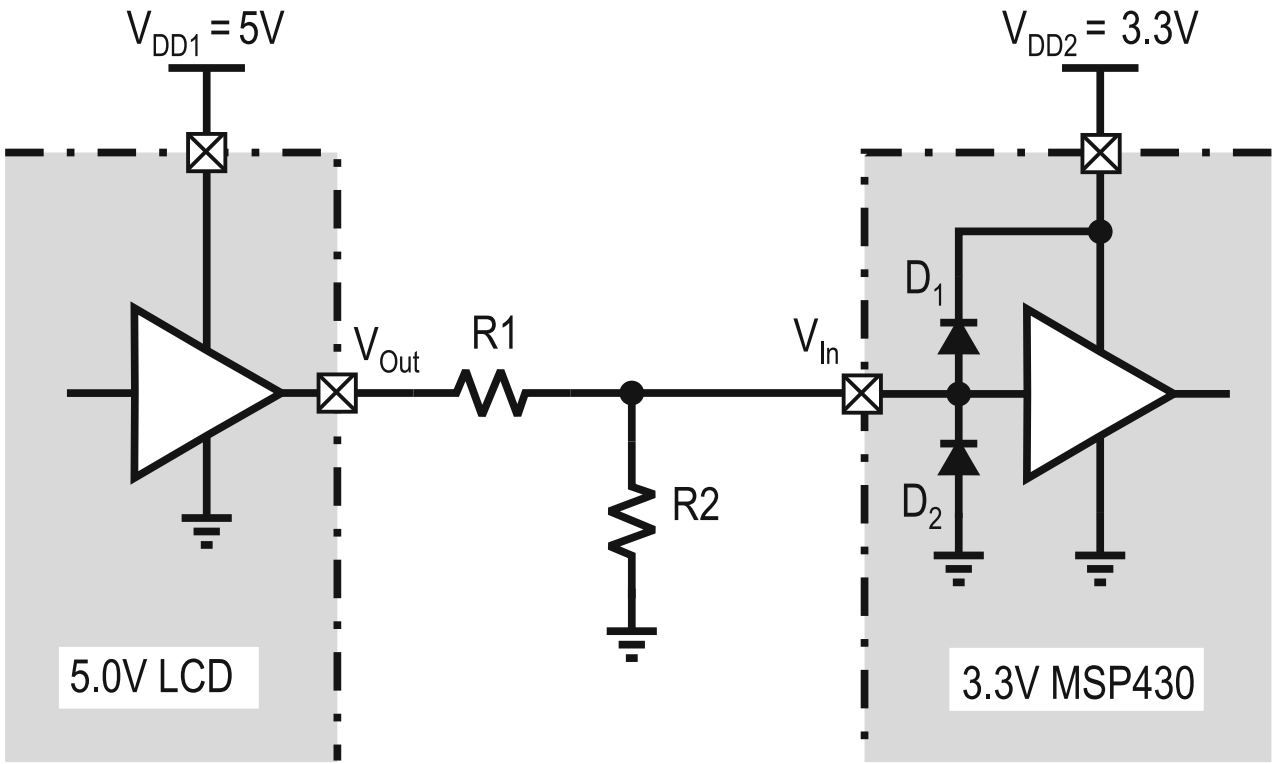
Configuração de Entradas e saídas digitais: MSP430

- Pinos são divididos em portas de 8 pinos (registradores de 8-bits):
  - P1: P1.0, P1.1 ..., P1.7
  - P2: P2.0, P2.1 ..., P2.7
  - Pi.j: número de portas depende do microcontrolador



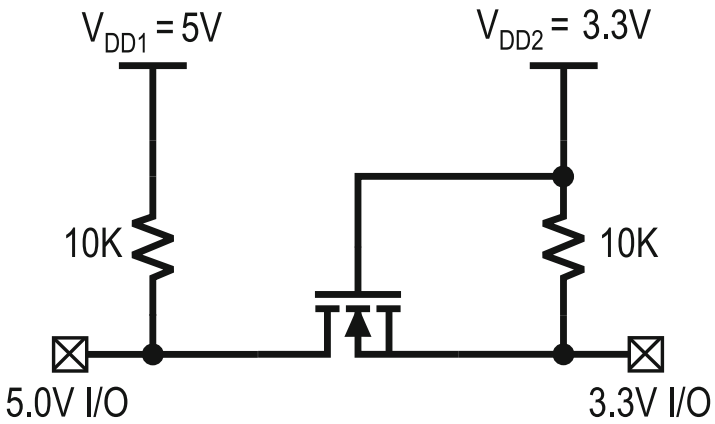
Entradas e saídas digitais: MSP430

- Conversor de nível: divisor resistivo.
- Sistemas unidirecionais e lentos.



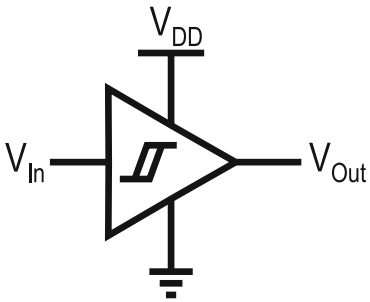
Entradas e saídas digitais: MSP430

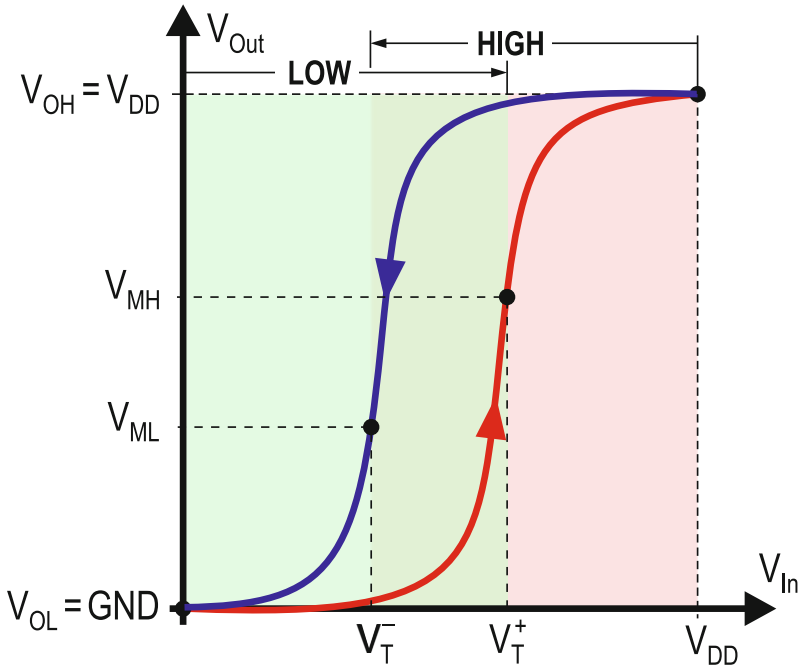
- Conversor de nível bidirecional.
- Sistemas rápidos.



Entradas e saídas digitais: MSP430

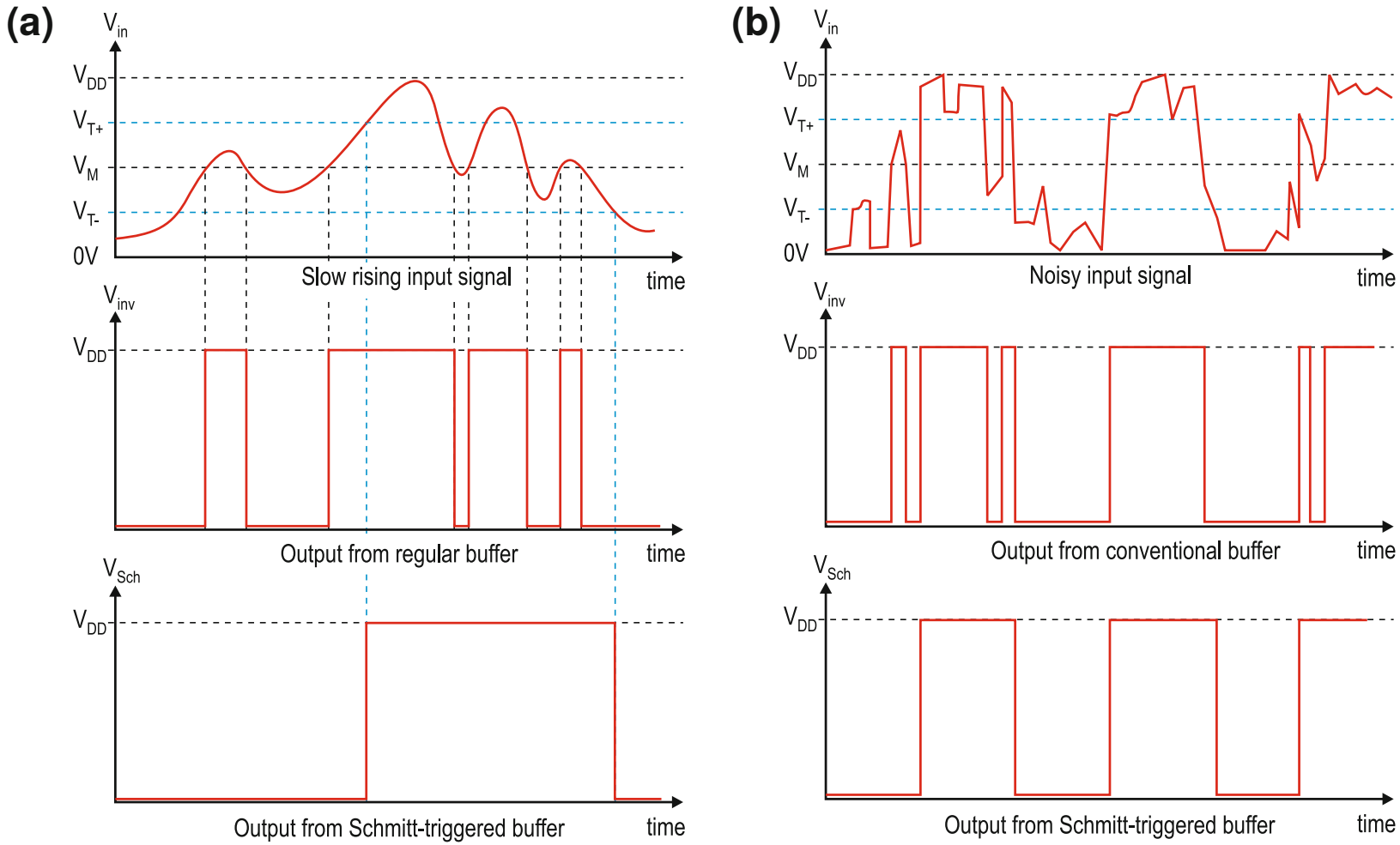
- Buffer com histerese: Schmitt-trigger.





Entradas e saídas digitais: MSP430

- Buffer com histerese: a Schmitt-trigger.

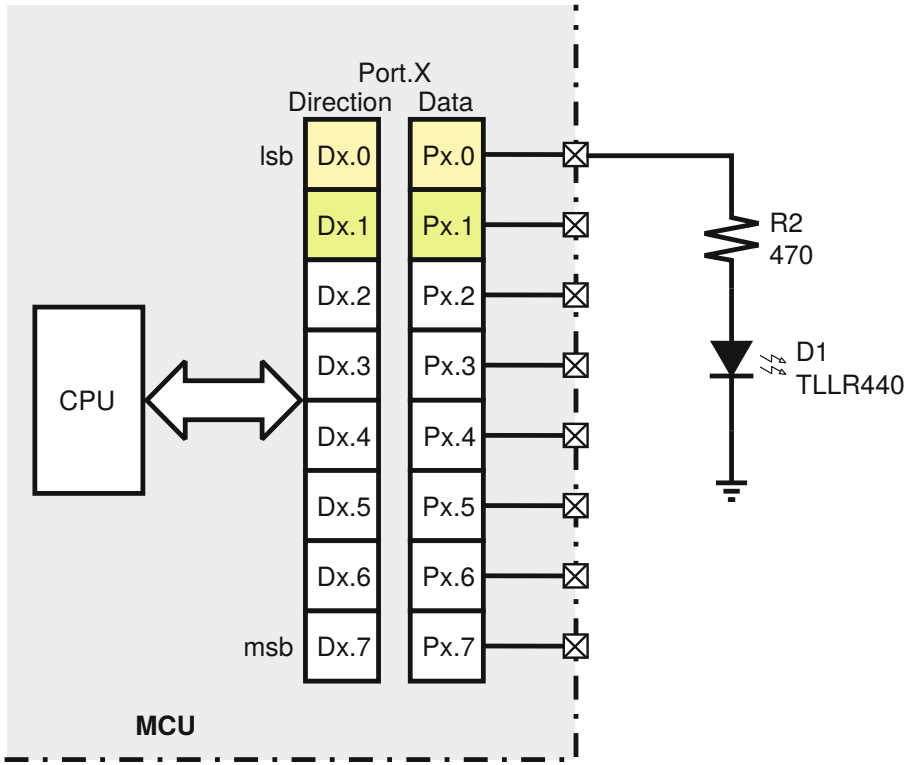


Configuração de entradas e saídas digitais

- Registradores básicos de configuração:
  - PxDIR com $x = (1, 2, 3, ...)$: direção. Cada bit em nível alto configura um pino como saída.
  - PxREN com $x = (1, 2, 3, ...)$: caba bit em nível alto habilita resistor de *pull-up/down* de um pino.
  - PxOUT com $x = (1, 2, 3, ...)$: se pino é saída, altera o nível lógic. Se entrada e pull-up/down habilitado, nível alto habilita pull-up, nível baixo habilita pull-down.
  - PxIN com $x = (1, 2, 3, ...)$: leitura de dados quando configurado como entrada.

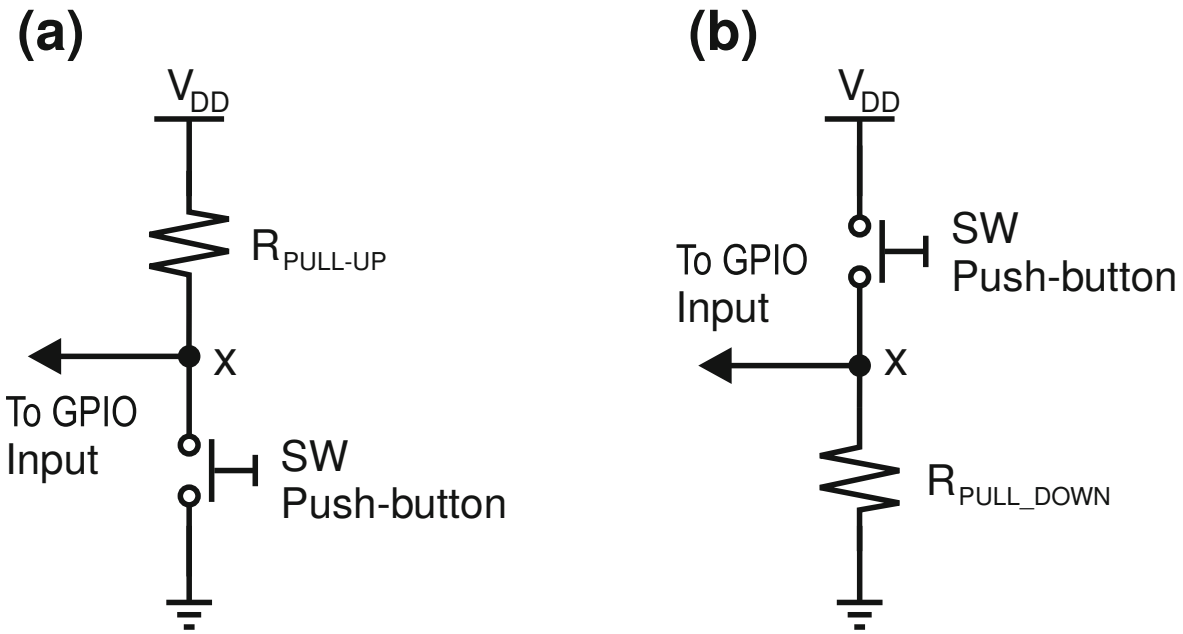
Configuração de saídas digitais

```c
#include <msp430.h>

#define LED   BIT0
#define DELAY 5000

int main(void)
{
    int i;

    /* Configuração de hardware */
    WDTCTL = WDTPW | WDTHOLD;

    /* Pino P1.0 como saída */
    P1DIR |= LED;

    /* main não pode retornar */
    while(1){
        /* Liga/Desliga LED */
        P1OUT = P1OUT ^ LED;

        /* Atraso */
        for (i=DELAY;i--; i > 0);
    }

    return 0;
}
```



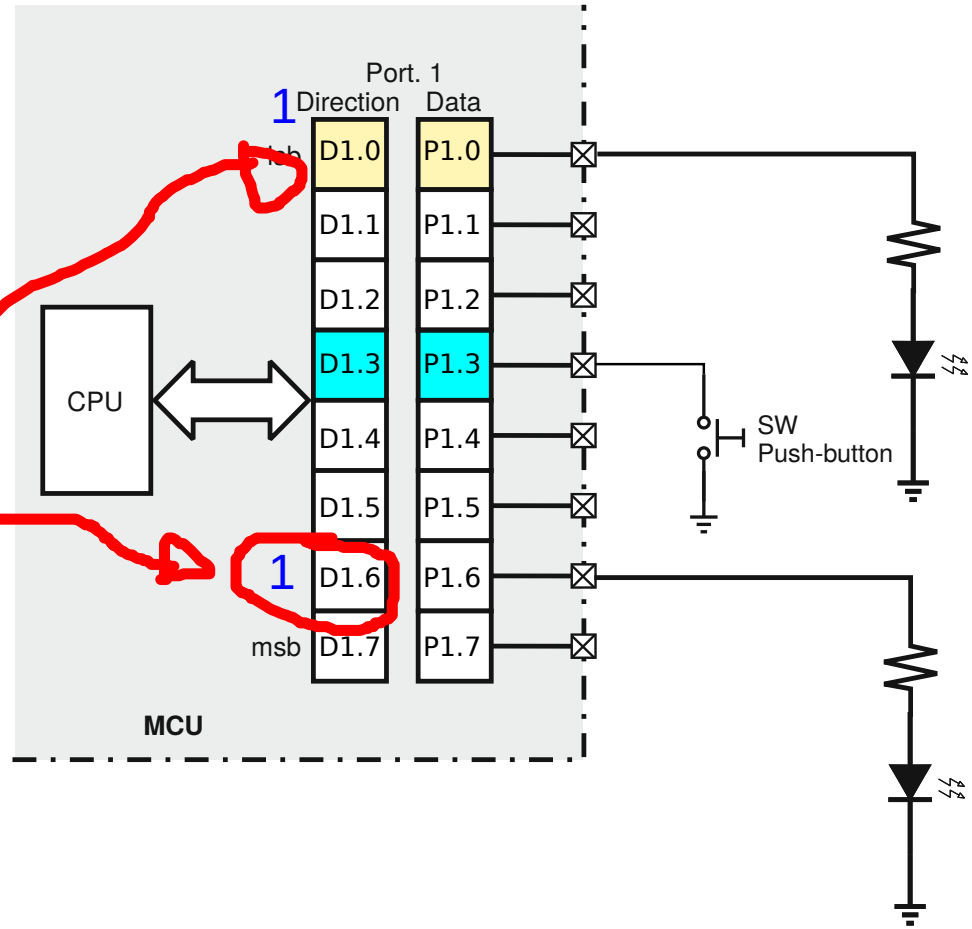
Configuração de entradas digitais: *push buttons*

- Entradas digitais possuem alta impedância.
- Inadequado deixá-las flutuando.



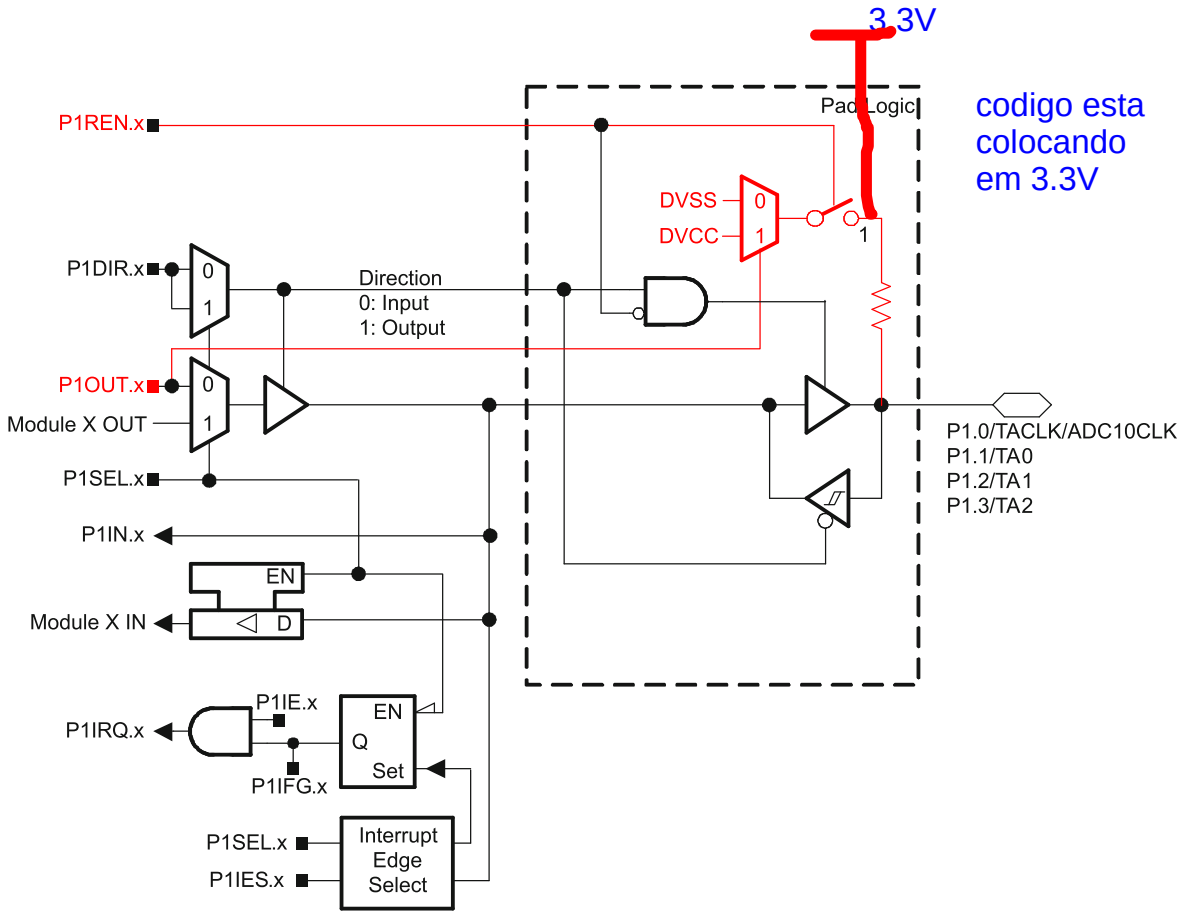
Configuração de saídas digitais

```c
#include <msp430.h>

#define LED_1   BIT0
#define LED_2   BIT6
#define BUTTON  BIT3

void hardware_init()
{
    /* Acesso direto: P1.0 e P1.6 como
     saídas. Demais como entrada      */
    P1DIR |= LED_1 | LED_2;

    /* Habilita resistor de pull up
     * ou pull down */
    P1REN |= BUTTON;
    /* Habilita resitor como pull up */
    P1OUT |= BUTTON;
}
```



Configuração de saídas digitais

```c
#include <msp430.h>

#define LED_1   BIT0
#define LED_2   BIT6
#define BUTTON  BIT3

void hardware_init()
{
    /* Acesso direto: P1.0 e P1.6 como
     saídas. Demais como entrada      */
    P1DIR |= LED_1 | LED_2;

    /* Habilita resistor de pull up
     * ou pull down */
    P1REN |= BUTTON;
    /* Habilita resitor como pull up */
    P1OUT |= BUTTON;
}
```




DANGER
MAN AT WORK