ENG
Departamento de Engenharia Eletrônica
UFMG

ELT134 – Laboratório de Arquitetura e Organização de Computadores

Método de Projeto de Periféricos

Autor: Prof. Ricardo de Oliveira Duarte

As figuras usadas nesse conjunto de slides para fins de ensino, foram obtidas a partir *PIC32MX_Datasheet_v2_61143B.pdf* encontrado em: https://ww1.microchip.com/downloads/en/DeviceDoc/PIC32MX_Datasheet_v2_61143B.pdf

creative commons

ENG Departamento de Engenharia Eletrônica

UFMG

Atribuição-NãoComercial-CompartilhaIgual 4.0 Internacional

Atribuição-NãoComercial-CompartilhaIgual
CC BY-NC-SA

Esta licença permite que outros remixem, adaptem e criem a partir do seu trabalho para fins não comerciais, desde que atribuam a você o devido crédito e que licenciem as novas criações sob termos idênticos.

Texto completo da licença (em Português): https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode.pt

Assuntos

- Considerações iniciais
- Método de Projeto e Implementação de um Periférico
- Integração a CPU
- Exemplos

Considerações iniciais

- Consideraremos periférico todo e qualquer componente externo a CPU que possa realizar alguma tarefa auxiliar à CPU.
- Essa tarefa pode ser:
  - Transação de entrada de dados;
  - Transação de saída de dados;
  - Transação de ocorrência de interrupções e/ou exceções;
  - Transações de controle de fluxo de dados.
- Cada periférico pode funcionar em frequência igual e/ou diferente da frequência de funcionamento da CPU.

Considerações iniciais

- Todo microcontrolador apresenta uma diversidade de periféricos.
- Cada periférico tem uma diversidade de modos de funcionamento (configuração).
- Cada microcontrolador tem o seu próprio conjunto de periféricos projetado para funcionar com uma diversidade finita de modos de operação.
- Usaremos como referência [5] o *datasheet* do documento juntamente com roteiros de tarefas e o planejamento de aulas dessa disciplina para projetar e implementar cada periférico tratado nesse curso.

Método de Projeto e Implementação de um Periférico

- O método de projeto de qualquer periférico segue o método RTL da referência [3] do documento com o planejamento de aulas dessa disciplina.

TABELA 5.1 Método de projeto RTL

	Passo	Descrição
Passo 1	Obtenha uma máquina de estados de alto nível	Descreva o comportamento desejado do sistema na forma de uma máquina de estados de alto nível. Essa máquina consiste em estados e transições. A máquina de estados é de "alto nível" porque as condições para as transições e as ações dos estados são mais do que simplesmente operações booleanas envolvendo os bits de entrada e de saída.
Passo 2	Crie um bloco operacional	Partindo da máquina de estados de alto nível do passo anterior, crie um bloco operacional capaz de realizar as operações que envolvem dados.
Passo 3	Conecte o bloco operacional a um bloco de controle	Conecte o bloco operacional a um bloco de controle. Conecte também as entradas e saídas booleanas que são externas ao bloco de controle.
Passo 4	Obtenha a FSM do bloco de controle	Converta a máquina de estados de alto nível na máquina de estados finitos do bloco de controle (FSM). Para isso, substitua as operações que envolvem dados por sinais de controle, que são ativados ou lidos pelo bloco de controle.

A figura usada nesse slide para fins de ensino, foi obtida a partir da pág. 244 do livro de Frank Vahid, Sistemas Digitais: projeto, otimizações e HDLs. 2008 – Ed. Bookman

Integração a *CPU*

- A integração de um periférico à *CPU* se dá para a construção de Microcontroladores (*MCUs*) e/ou Sistemas integrados (*SoCs*).
- Cada periférico tem à ele associado um conjunto finito de registradores específicos para um determinado fim. São exemplos de registradores:
  - Registrador(es) para armazenar a configuração ou modo de funcionamento;
  - Registrador para armazenar o endereço da rotina de tratamento de interrupção associada à esse tipo de periférico;
  - Registrador para armazenar entrada de dados para o periférico;
  - Registrador para armazenar saída de dados do periférico;
  - Registrador para armazenar situações de controle do periférico;
  - Registrador para armazenar *status* de funcionamento do periférico.

Integração a *CPU*

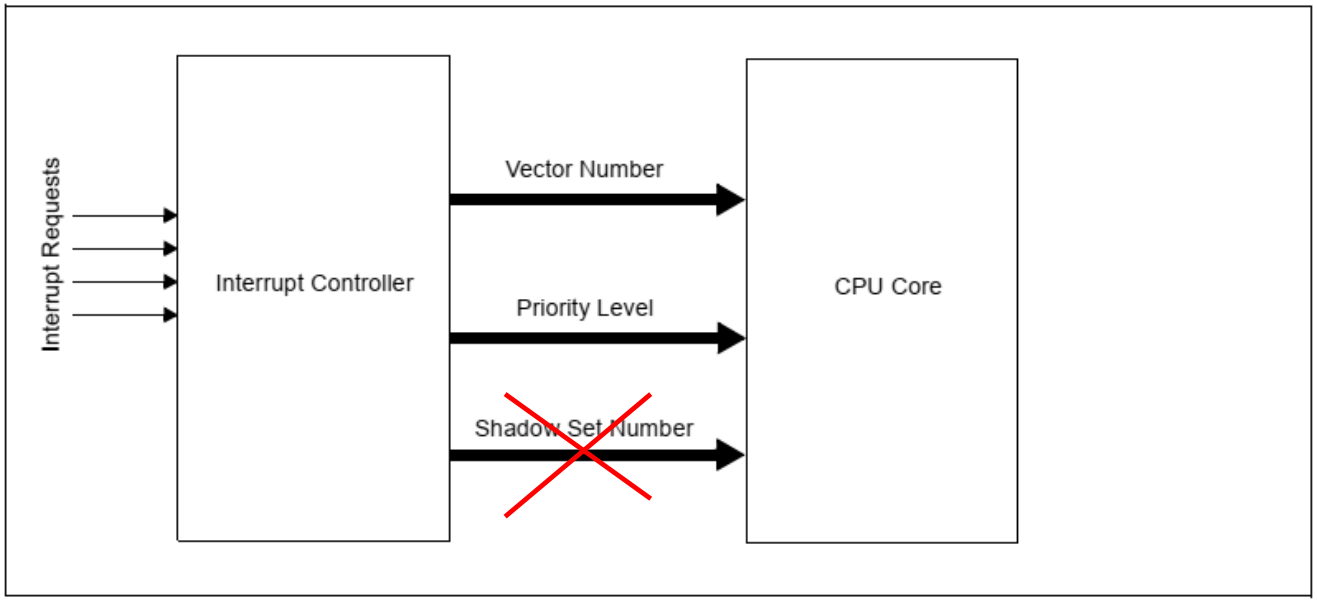
- A *CPU* tem que prever acesso a esses registradores.
- Em uma modelo de *CPU load-store*, o acesso a leitura e escrita à esses registradores é realizado com instruções *load* and *store* respectivamente.
- O acesso aos registradores dos periféricos são realizados com instruções *load and store,* acessando endereços específicos de memória, onde se encontram mapeados os registradores reservados à cada periférico.
- Portanto uma região específica da memória, acessível pela CPU, deve ser reservada para os registradores dedicados a cada periférico.
- Algumas modificações no caminho de dados e na unidade de controle da *CPU* podem ser necessárias. Fique atento a essa observação!

Exemplos: A controladora de interrupções

- A controladora de interrupções é um periférico existente em qualquer sistema composto por uma *CPU*.
- A controladora de interrupções é responsável pelo pré-processamento de solicitações de interrupção (*IRQ*) de um número de periféricos integrados no sistema e apresentá-las no ordem apropriada para o processador tratar uma a uma por vez.
- A controladora de interrupções tem por objetivo, portanto:
  - Gerenciar qual recurso do sistema (periférico) demanda a atenção da *CPU;*
  - Gerenciar a ordem de atendimento de interrupções por parte da *CPU*;
  - Gerenciar se a *CPU* poderá atender ou não interrupções.

Exemplos: A controladora de interrupções

FIGURE 8-1: INTERRUPT CONTROLLER MODULE

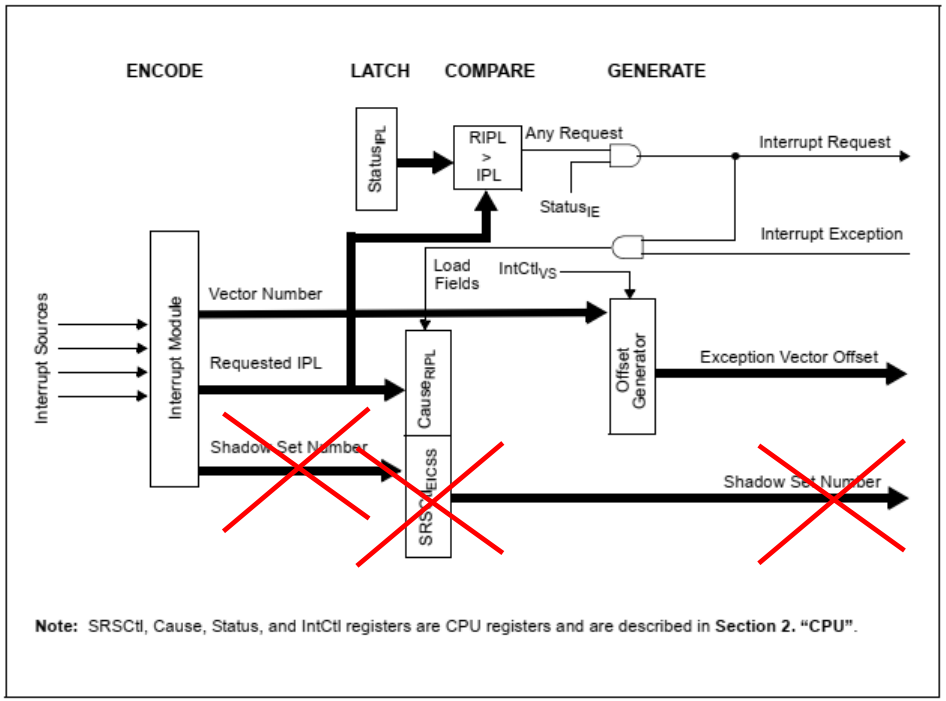


As figuras usadas nesse conjunto de slides para fins de ensino, foram obtidas a partir *PIC32MX_Datasheet_v2_61143B.pdf* encontrado em: https://ww1.microchip.com/downloads/en/DeviceDoc/PIC32MX_Datasheet_v2_61143B.pdf

Exemplos: A controladora de interrupções

- Registradores específicos:
  - *IER (Interrupt Enable Register)*
  - *IFR (Interrupt Flag Register)*
  - *IPR (Interrupt Priority Register)*
  - *ISR (Interrupt Service Routine Reg)*
  - *CAUSE (Interrupt Cause Register)*
  - *EPC (Exception Program Counter)*

FIGURE 8-2: INTERRUPT PROCESS



Note: SRSCtl, Cause, Status, and IntCtl registers are CPU registers and are described in **Section 2. "CPU"**.

As figuras usadas nesse conjunto de slides para fins de ensino, foram obtidas a partir *PIC32MX_Datasheet_v2_61143B.pdf* encontrado em: https://ww1.microchip.com/downloads/en/DeviceDoc/PIC32MX_Datasheet_v2_61143B.pdf

Exemplos: A controladora de interrupções

- Registradores específicos:
  - *IER (Interrupt Enable Register)* – registrador responsável por guardar estado de habilitar ou desabilitar a atenção da CPU às exceções (eventos internos). Responsável também por guardar estado de habilitar ou desabilitar individualmente, a atenção da CPU aos eventos de periféricos (eventos externos).
  - *IFR (Interrupt Flag Register)* – registrador responsável por registrar para a CPU a ocorrência de alguma exceção.
  - *IPR (Interrupt Priority Register)* – registrador responsável por armazenar a ordem de prioridade do atendimento de uma determinada exceção à outra.
  - *CAUSE (Interrupt Cause Register)* – registrador responsável por guradar o código do periférico e/ou da exceção que causou o pedido de atenção da CPU.
  - *ISR (Interrupt Service Routine Reg)* – registrador responsável por armazenar o endereço da primeira instrução da rotina que tratará a exceção em questão.
  - *EPC (Exception Program Counter)* – registrador responsável por guardar o endereço da primeira instrução a ser executada pela CPU após o término da execução da rotina de tratamento de exceção.

Exemplos: A controladora de interrupções

- Tarefa prática a ser executada:
  - Leia e execute os passos do roteiro *modulo controladora de interrupções.pdf* que eu preparei para vocês.

Exemplos: O modulo GPIO programável

- Periférico comum encontrado em qualquer Microcontrolador.
- Esse periférico registra qual pino do MCU será configurado como entrada, qual será configurado como saída. Se o pino for configurado como entrada, se essa entrada vai ser uma entrada analógica ou digital.
- Se um pino for configurado como entrada, se essa entrada estará associada a um tipo de interrupção. Exemplos:
  - Mudança de estado. 1→0 ou de 0 →1
  - Um único tipo de transição, por exemplo: de 0 →1

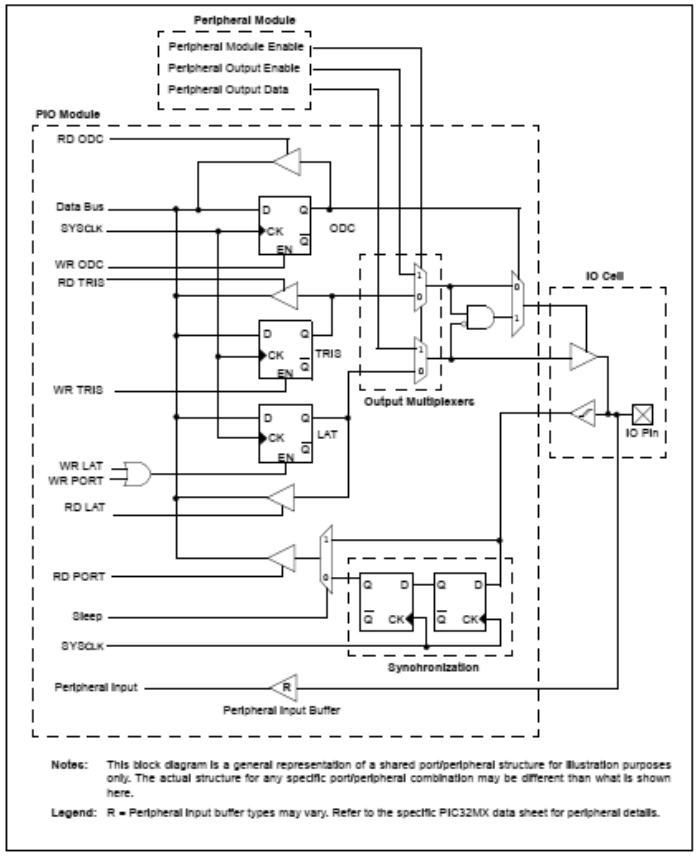
Exemplos: O modulo GPIO programável

A figura ao lado mostra o modulo de GPIO programável da família PIC32MX de Microcontroladores da Microchip

As figuras usadas nesse conjunto de slides para fins de ensino, foram obtidas a partir *PIC32MX_Datasheet_v2_61143B.pdf* encontrado em:

https://ww1.microchip.com/downloads/en/DeviceDoc/PIC32MX_Datasheet_v2_61143B.pdf

FIGURE 12-2: BLOCK DIAGRAM OF A TYPICAL MULTIPLEXED PORT STRUCTURE



Exemplos: O módulo GPIO programável

- Tarefa prática a ser executada:
  - Leia e execute os passos do roteiro *modulo GPIO programavel.pdf* que eu preparei para vocês.
- Referência complementar para consulta se necessário:
  - *GPIO paper.pdf* e *mb_gpio_mod_act.pdf*

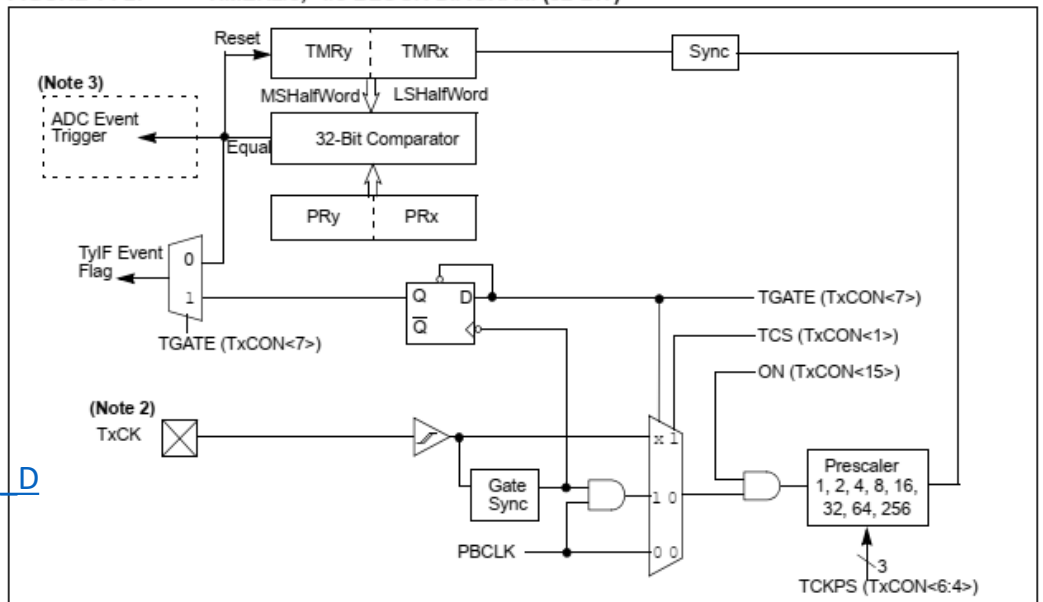
Exemplos: O modulo de contadores e *timers*

A figura ao lado mostra o modulo Timer 2/3, 4/5 da família PIC32MX de Microcontroladores da Microchip

As figuras usadas nesse conjunto de slides para fins de ensino, foram obtidas a partir *PIC32MX_Datasheet_v2_61143B.pdf* encontrado em:

https://ww1.microchip.com/downloads/en/DeviceDoc/PIC32MX_Datasheet_v2_61143B.pdf

FIGURE 14-2: TIMER2/3, 4/5 BLOCK DIAGRAM (32-BIT)



Note 1: In this diagram, the use of of "x' in registers TxCON, TMRx, PRx, TxCK refers to either Timer2 or Timer4; the use of 'y' in registers TyCON, TMRy, PRy, TyIF refers to either Timer3 or Timer5.

2: TxCK pins not available on 64-pin devices.

3: ADC event trigger is available only on Timer2/3 pair.

Exemplos: O modulo de contadores e *timers*

- Tarefa prática a ser executada:
  - Leia e execute os passos do roteiro *modulo contador e timer programavel.pdf* que eu preparei para vocês.

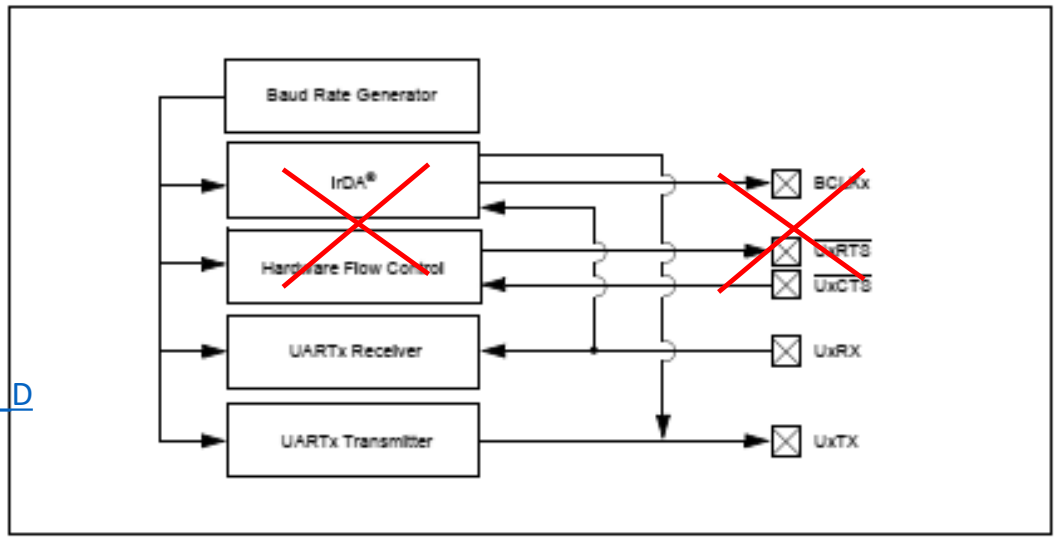
Exemplos: *UART*

A figura ao lado mostra o modulo UART
da família PIC32MX de Microcontroladores da Microchip

As figuras usadas nesse conjunto de slides para fins de ensino, foram obtidas a partir *PIC32MX_Datasheet_v2_61143B.pdf* encontrado em:

https://ww1.microchip.com/downloads/en/DeviceDoc/PIC32MX_Datasheet_v2_61143B.pdf

FIGURE 19-1: UART SIMPLIFIED BLOCK DIAGRAM



Exemplos: *UART*

As figuras usadas nesse conjunto de slides para fins de ensino, foram obtidas a partir *PIC32MX_Datasheet_v2_61143B.pdf* encontrado em:

https://ww1.microchip.com/downloads/en/DeviceDoc/PIC32MX_Datasheet_v2_61143B.pdf

19.2 UART Baud Rate Generator (BRG)

The UART module includes a dedicated 16-bit Baud Rate Generator. The BRGx register controls the period of a free-running 16-bit timer. Equation 19-1 shows the formula for computation of the baud rate with BRGH = 0.

EQUATION 19-1: UART BAUD RATE WITH BRGH = 0(1)

$$Baud\ Rate = \frac{F_{PB}}{16 \cdot (UxBRG + 1)}$$

$$UxBRG = \frac{F_{PB}}{16 \cdot Baud\ Rate} - 1$$

Note 1: FPB denotes the peripheral bus clock frequency.

Example 19-1 shows the calculation of the baud rate error for the following conditions:

- FPB = 4 MHz
- Desired Baud Rate = 9600

The maximum possible baud rate with BRGH = 0 is FPB/16.

The minimum possible baud rate is FPB/(16 * 65536).

Equation 19-2 shows the formula for computation of the baud rate with BRGH = 1.

EQUATION 19-2: UART BAUD RATE WITH BRGH = 1(1)

$$Baud\ Rate = \frac{F_{PB}}{4 \cdot (UxBRG + 1)}$$

$$UxBRG = \frac{F_{PB}}{4 \cdot Baud\ Rate} - 1$$

Note 1: FPB denotes the instruction cycle clock frequency.

The maximum possible baud rate with BRGH = 1 is FPB/4 for UxBRG = 0, and the minimum possible baud rate is FPB/(4 * 65536).

Writing a new value to the UxBRG register causes the baud rate counter to be cleared. This ensures that the BRG does not wait for a timer overflow before it generates the new baud rate.

EXAMPLE 19-1: BAUD RATE ERROR CALCULATION (BRGH = 0)

```
Desired Baud Rate      =   Fpb/(16 (UxBRG + 1))
Solving for UxBRG value:
    UxBRG              =   ( (Fpb/Desired Baud Rate)/16) - 1
    UxBRG              =   ((4000000/9600)/16) - 1
    UxBRG              =   [25.042] = 25
Calculated Baud Rate   =   4000000/(16 (25 + 1))
                       =   9615
Error                  =   (Calculated Baud Rate - Desired Baud Rate)
           Desired Baud Rate
                       =   (9615 - 9600)/9600
                       =   0.16%
```

Exemplos: *UART*

- Tarefa prática a ser executada:
  - Leia e execute os passos do roteiro *modulo UART.pdf* que eu preparei para vocês.