

# ELF52 - Sistemas Microcontrolados

## Pinos de Entrada/Saída - Parte 1

**Professor:**

Prof. Marcos Eduardo

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

# GPIO

# GPIO

- General Purpose Input/Output;
- Para que servem?

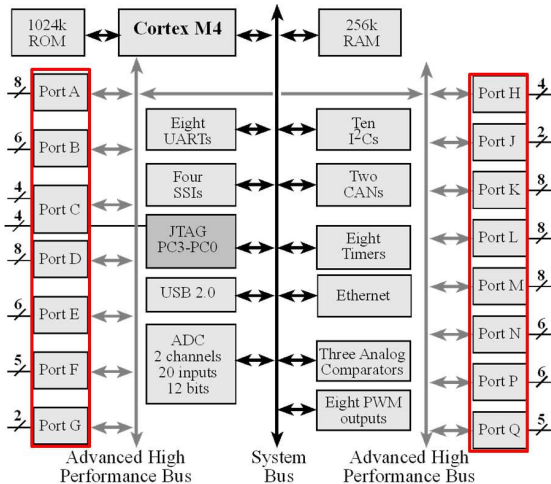
# GPIO

- Utilizados para para trocar informação digital com o mundo externo;
- Exemplo:
  - Controlar LEDs;
  - Controlar chaves.

## Pinos de I/O do TM4C

- Podemos utilizar os pinos para operações de I/O em paralelo;
- Entretanto, a maioria dos pinos têm uma ou mais funções alternativas:
  - UART;
  - SSI (SPI);
  - $I^2C$ ;
  - *Timer*;
  - PWM;
  - ADC;
  - Comparador Analógico;
  - USB;
  - Ethernet;
  - CAN.

## Pinos de I/O do TM4C



(Adaptado de VALVANO, J.)

## Pinos de I/O do TM4C1294

- Nota sobre o JTAG/SWD, JLINK e ICDI:
  - JTAG: norma projetada originalmente para realizar testes elétricos em componentes e placas saindo das fábricas, assumindo o controle de seus pinos de entrada / saída. Presente no microcontrolador que utilizamos;
  - SWD: protocolo alternativo específico para chips ARM, que é compatível com os pinos JTAG, mas usa menos fios. Presente no microcontrolador que utilizamos;
  - J-LINK: programador da Segger externo ao microcontrolador capaz de programar CIs compatíveis com JTAG/SWD através de uma interface USB conectada ao computador;
  - In-Circuit Debug Interface (ICDI): programador presente no nosso kit de desenvolvimento capaz de realizar a mesma operação feita pelo J-LINK, mas sem a necessidade de comprar um equipamento a mais.

# Pinos de I/O do TM4C1294

- Os pinos de I/O podem ser associados a até sete funções alternativas;
- Exemplo: PA0
  - I/O Digital;
  - Entrada Serial;
  - Clock I2C;
  - Timer I/O;
  - Receptor CAN.



- Pinos PC3 - PC0 devem ser reservados para o depurador JTAG;
- Pinos PA1 - PA0 já são usados para comunicação serial;
- Há funções que podem ser mapeadas em mais de um pino, por exemplo:
  - T0CCP0 pode ser mapeado em PA0, PD0 ou PL4.
- Há funções que só existem em um pino, por exemplo:
  - U0Rx só existe em PA0.

## Pinos de I/O do TM4C1294

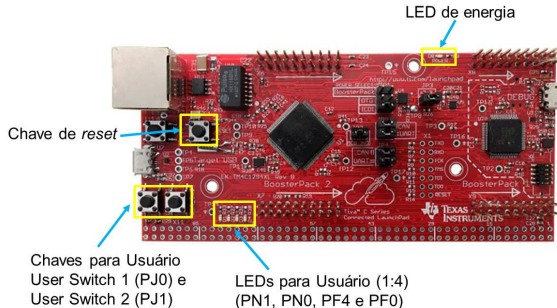
- A tabela 10-2 (páginas 743-746) do Datasheet do microcontrolador demonstra todas as funções dos pinos;
- Exemplo (PARTE da tabela extraída do Datasheet):
  - Coluna indica a posição os bits no registrador PCTL (4 bits);
  - Exemplo: Coluna 3 → PCTL = 0011.

IO	Pin	Analog or Special Function <sup>a</sup>	Digital Function (GPIO PCTL PMCx Bit Field Encoding) <sup>b</sup>											
			1	2	3	4	5	6	7	8	11	13	14	15
PA0	33	*	U0Rx	I2C9SCL	T0CCP0	*	*	*	CAN0Rx	*	*	*	*	*
PA1	34	*	U0Tx	I2C9SDA	T0CCP1	*	*	*	CAN0Tx	*	*	*	*	*
PA2	35	*	U4Rx	I2C8SCL	T1CCP0	*	*	*	*	*	*	*	*	SSI0C1k
PA3	36	*	U4Tx	I2C8SDA	T1CCP1	*	*	*	*	*	*	*	*	SSI0Fss
PA4	37	*	U3Rx	I2C7SCL	T2CCP0	*	*	*	*	*	*	*	*	SSI0XDAT0

(Adaptado de datasheet do microcontrolador)

# Pinos de I/O do TM4C1294

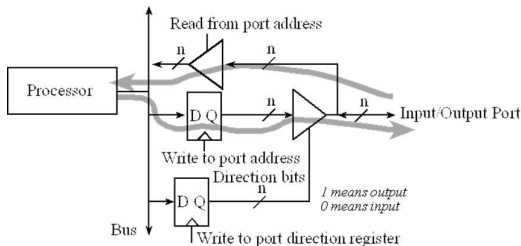
- A placa EK-TM4C1294XL tem duas chaves e quatro LEDs:
  - Chaves de usuário → Lógica negativa e necessitam habilitar um resistor de *pull-up* (PUR);
  - LEDs de usuário → Lógica positiva;
  - Chave de reset;
  - LED de energia.



(Adaptado do *user guide* da placa EK-TM4C1294XL)

## Pinos de I/O

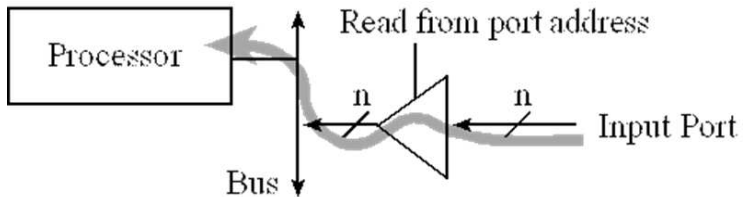
- A porta de I/O mais simples em um  $\mu$ Controlador é a porta paralela:
  - Múltiplos sinais podem ser acessados ao mesmo tempo;
  - Mecanismo simples que permite ao SW interagir com dispositivos externos;



(Adaptado de VALVANO, J.)

## Pinos de I/O - Entrada

- Porta de entrada permite o SW ler sinais digitais externos;
- Um ciclo de leitura ao endereço da porta retorna o valor de todas as entradas naquele momento;
- O *driver tristate* direciona o sinal de entrada para o barramento de dados:



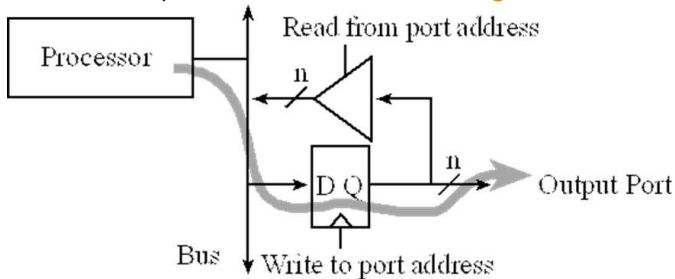
(Adaptado de VALVANO, J.)

## Pinos de I/O - Entrada

- Para fazer um pino de entrada escrever 0 no registrador de direção;
- Desta forma um acesso de escrita não tem efeito nenhum;
- A maioria dos pinos são tolerantes a 5V de entrada:
  - Valores entre 2V e 5V serão considerados ALTOS;
  - Valores entre 0V e 1,3V serão considerados BAIXOS.

## Pinos de I/O - Saída

- Porta de saída permite o SW escrever sinais digitais externos, mas também permite ler o que foi escrito;
- Um ciclo de escrita no endereço porta escreve os valores nos pinos de saída;
- Para fazer um pino de saída escrever 1 no **registrador de direção**:



(Adaptado de VALVANO, J.)

# Programação de I/O

- Mas, como acessar os GPIOs na Tiva?
  - Capítulo 10 do Datasheet;



# Programação de I/O

- Mas, como acessar os GPIOs na Tiva?
  - Capítulo 10 do Datasheet;



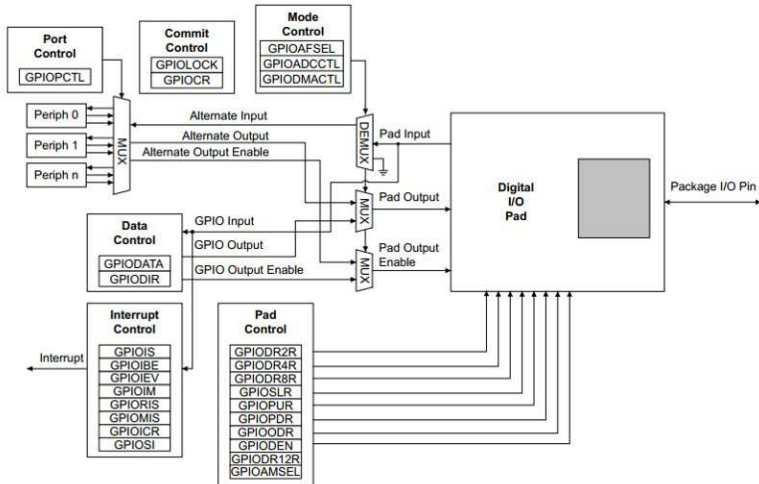
# Programação de I/O

- Como acessar os GPIOs na Tiva?
  - Na Tiva e em vários outros microcontroladores as portas de I/O **são mapeadas em memória**;
  - Cada porta deve seguir uma série de configurações na memória (em registradores) antes de ser utilizada;
  - Para escrever e ler nos pinos de cada porta também deve-se escrever ou ler em endereços específicos da memória.
    - Realizar operações de **LDR** e **STR**;

# Programação dos GPIO

- As operações com I/O mapeado em memória se parecem com operações com memória, mas não agem igual memória:
  - Alguns bits são *read-only*;
  - Alguns bits são *write-only*;
  - Alguns bits só podem ser setados (1);
  - Alguns bits só podem ser limpos (0).

# Registradores dos GPIO



(Adaptado de datasheet do microcontrolador)

## Registadores dos GPIO

- Cada registrador segue o endereço base de uma porta + o endereço de configuração;
- Endereços base de cada porta (Datasheet Seção 10.5 - pag 759):

GPIO Port	Endereço Base
GPIO Port A	0x4005.8000
GPIO Port B	0x4005.9000
GPIO Port C	0x4005.A000
GPIO Port D	0x4005.B000
GPIO Port E	0x4005.C000
GPIO Port F	0x4005.D000
GPIO Port G	0x4005.E000
GPIO Port H	0x4005.F000

GPIO Port	Endereço Base
GPIO Port J	0x4006.0000
GPIO Port K	0x4006.1000
GPIO Port L	0x4006.2000
GPIO Port M	0x4006.3000
GPIO Port N	0x4006.4000
GPIO Port P	0x4006.5000
GPIO Port Q	0x4006.6000

# Registadores dos GPIO

- *Direction Register* (**GPIODIR**)
  - Especifica se os pinos são de entrada ou saída. 1 bit por pino.
- *Digital Enable Register* (**GPIODEN**)
  - Se o pino deve ser utilizado como digital (entrada ou saída). 1 bit por pino.
- *Analog Mode Select Register* (**GPIOAMSEL**)
  - Especifica se o pino será usado como entrada analógica. 1 bit por porta.
- *Alternate Function Register* (**GPIOAFSEL**)
  - Especifica se alguma função alternativa será utilizada. 1 bit por pino.
- *Port Control Register* (**GPIOCTL**)
  - Especifica qual a função alternativa (tabela 10-2 do *datasheet*) será utilizada. 4 bits por pino.

# Registradores dos GPIO

- *Data Register* (**GPIODATA**)
  - Realiza entrada e saída na porta. 1 bit por pino.
- *Run Mode Clock Gating* (**RCGCGPIO**) pag 382
  - Habilita o *clock* de cada porta. Obrigatório para habilitar uma porta. 1 bit por porta.
- *Peripheral Ready* (**PRGPIO**) pag 499
  - Indica se a porta de GPIO já está pronta para o uso. 1 bit por porta.

# Programação dos GPIO

- Passo-a-passo para ativar uma porta como entrada e saída (Resumo da seção 10.4 do *Datasheet*):
  - 1 Ative o *clock* para a porta setando o bit correspondente no registrador **RCCGCGPIO** e, após isso, verifique no **PRGPIO** se a porta está pronta para uso;
  - 2 Desabilite a funcionalidade analógica, limpando os bits no registrador **GPIOAMSEL**;
  - 3 Selecione a funcionalidade de GPIO limpando os bits no registrador **GPIOCTL**;
  - 4 Especifique se o pino é de entrada ou saída limpando ou setando, respectivamente, os bits no registrador **GPIODIR**.



# Programação dos GPIO

- Passo-a-passo para ativar uma porta como entrada e saída (continuação):
  - 5 Como o objetivo é utilizar os pinos como GPIO, e não a função alternativa, limpe os bits correspondentes no registrador **GPIOAFSEL**;
  - 6 Habilite a funcionalidade de entrada e saída digital no registrador **PIODEN**.
- **(Opcional)** Habilite um resistor de *pull-up* para entrada: importante para operação com chaves no registrador **GPIOPUR**.

# Leitura e Escrita dos GPIO

- Iniciamos uma GPIO;
- E agora, como ler/escrever na GPIO?

# Leitura e Escrita dos GPIO

- Data Register (**GPIODATA**):
  - Através do *Data Register* realiza-se a leitura e escrita do valor desejado dos pinos de dada porta;
  - Um **STR** para o endereço do *Data Register* fará com que os pinos sejam modificados, ou seja, é realizada uma **ESCRITA** nos pinos;
  - Um **LDR** do endereço do DATA Register fará com que os pinos sejam lidos, ou seja, é realizada uma operação de **LEITURA**.

# Leitura e Escrita dos GPIO

- Data Register (**GPIO DATA**):

GPIO Port	Endereço
GPIO Port A	0x4005.8 <b>3FC</b>
GPIO Port B	0x4005.9 <b>3FC</b>
GPIO Port C	0x4005.A <b>3FC</b>
GPIO Port D	0x4005.B <b>3FC</b>
GPIO Port E	0x4005.C <b>3FC</b>
GPIO Port F	0x4005.D <b>3FC</b>
GPIO Port G	0x4005.E <b>3FC</b>
GPIO Port H	0x4005.F <b>3FC</b>

GPIO Port	Endereço
GPIO Port J	0x4006.0 <b>3FC</b>
GPIO Port K	0x4006.1 <b>3FC</b>
GPIO Port L	0x4006.2 <b>3FC</b>
GPIO Port M	0x4006.3 <b>3FC</b>
GPIO Port N	0x4006.4 <b>3FC</b>
GPIO Port P	0x4006.5 <b>3FC</b>
GPIO Port Q	0x4006.6 <b>3FC</b>

# Leitura e Escrita dos GPIO

- Entretanto, se uma escrita é feita modificando todos os bits de uma porta, corre-se o risco de sobrescrever outros pinos **indesejadamente**;
- Para evitar alterações em pinos indesejados há duas formas (escrita "amigável"):
  - Usar o trio: *read-modify-write*;
  - Usar **endereçamento de bit específico** (disponível em alguns microcontroladores):
    - O *Data Register* apresenta uma estrutura complexa, permitindo o acesso individualmente de cada um dos bits ou de todos os bits da porta apenas modificando os endereços de acesso.

# Escrita Amigável nos GPIO

- *Read-modify-write:*
  - Se desejar setar o pino PK7 para 1:

```
1 LDR R1, =GPIO_PORTK_DATA_R ; Carrega-se o  
   endereço  
2 LDR R0, [R1] ; Lê para carregar o valor  
3 ; anterior da porta inteira  
4 ORR R0, R0, #0x80 ; Faz o OR bit a bit para  
   manter os valores  
5 ; anteriores e setar somente o bit  
6 STR R0, [R1] ; Escreve o novo valor da porta
```

## Escrita Amigável nos GPIO

- *Read-modify-write:*
  - Se desejar limpar o pino PK7.

```
1 LDR R1, =GPIO_PORTK_DATA_R ;Carrega-se o  
   endereço  
2 LDR R0, [R1] ; Lê para carregar o valor  
3 ; anterior da porta inteira  
4 BIC R0, R0, #0x80 ; Faz o AND negado bit a bit  
   para manter os  
5 ; valores anteriores e limpar somente o bit  
6 STR R0, [R1] ; Escreve o novo valor da  
   porta
```

# Dúvidas?