

ELF52 - Sistemas Microcontrolados

Pinos de Entrada/Saída

Professor:

Prof. Marcos Eduardo

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

GPIO

GPIO

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

GPIO

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



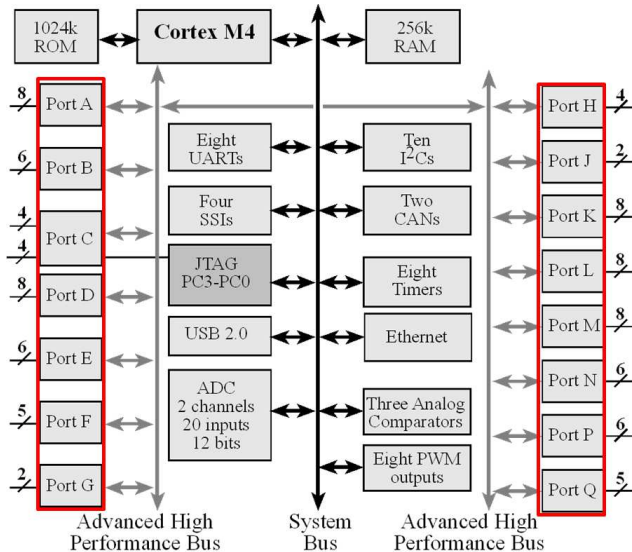
GPIO

- 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



Pinos de I/O do TM4C1294

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

Pinos de I/O do TM4C1294

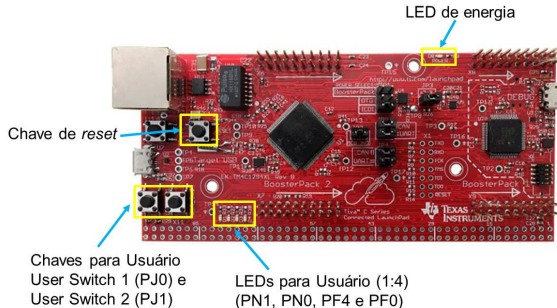
- A tabela 10-2 (páginas 743-746) do Datasheet do microcontrolador demonstra todas as funções dos pinos;
- Exemplo (PARTE da tabela extraída do Datasheet):
 - Coluna indica a posição os bits no registrador PCTL (4 bits);
 - Exemplo: Coluna 3 → PCTL = 0011.

IO	Pin	Analog or Special Function ^a	Digital Function (GPIO PCTL PMCx Bit Field Encoding) ^b											
			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	•	•	•	•	•	•	•	•	SSI0Fas
PA4	37	•	U3Rx	I2C7SCL	T2CCP0	•	•	•	•	•	•	•	•	SSI0XDATO

- 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.

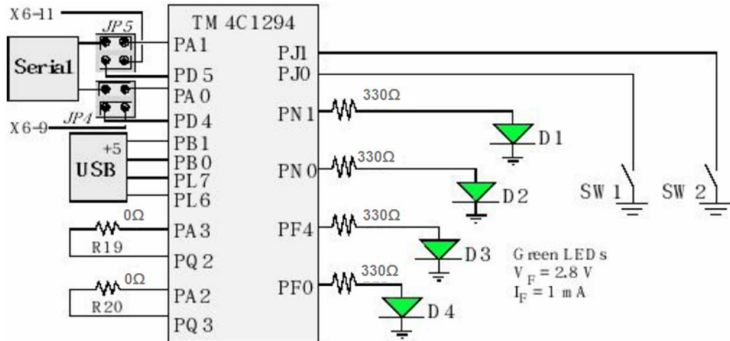
Pinos de I/O do TM4C1294

- A 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.



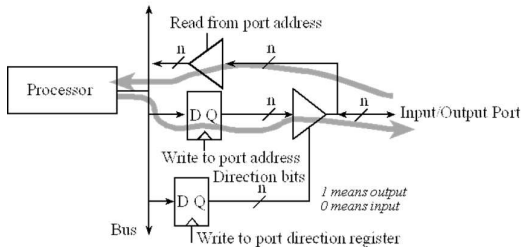
Pinos de I/O do TM4C1294

- EK-TM4C1294XL



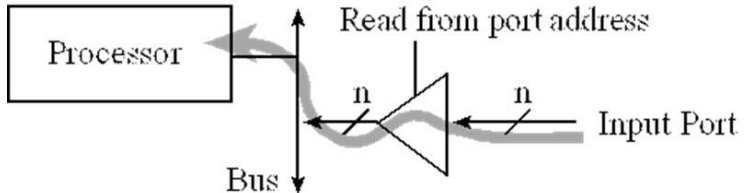
Pinos de I/O

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



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:

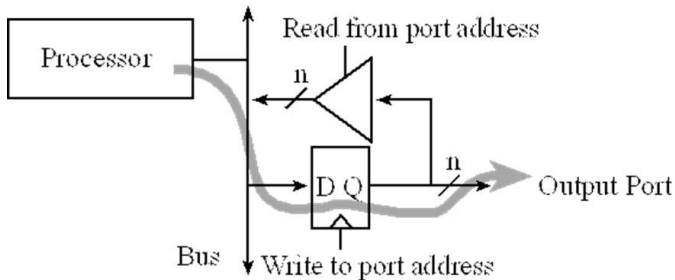


Pinos de I/O - Entrada

- 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:



Programação de I/O

- Mas, 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.

Programação de I/O

- Mas, 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.

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

- 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

Registradores dos GPIO

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

Registradores dos GPIO

- *Data Register* (GPIO_PORTx_DATA_R)
 - Realiza entrada e saída na porta. 1 bit por pino.
- *Run Mode Clock Gating* (RCGCGPIO) (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 `RCCGPIO` e, após isso, verifique no `PRGPIO` se a porta está pronta para uso;
 - 2 Desabilite a funcionalidade analógica, limpando os bits no registrador `GPIO_PORTx_AMSEL_R`;
 - 3 Selecione a funcionalidade de GPIO limpando os bits no registrador `GPIO_PORTx_PCTL_R`;
 - 4 Especifique se o pino é de entrada ou saída limpando ou setando, respectivamente, os bits no registrador `GPIO_PORTx_DIR_R`.

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 `GPIO_PORTx_AFSEL_R`;
 - 6 Habilite a funcionalidade de entrada e saída digital no registrador `GPIO_PORTx_DEN_R`.
- **(Opcional)** Habilite um resistor de *pull-up* para entrada: importante para operação com chaves no registrador `GPIO_PORTx_PUR_R`.

Leitura e Escrita dos GPIO

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

Leitura e Escrita dos GPIO

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



Leitura e Escrita dos GPIO

- Data Register (GPIO_PORTx_DATA_R):
 - 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_PORTx_DATA_R):

GPIO Port	Endereço
GPIO Port A	0x4005.83FC
GPIO Port B	0x4005.93FC
GPIO Port C	0x4005.A3FC
GPIO Port D	0x4005.B3FC
GPIO Port E	0x4005.C3FC
GPIO Port F	0x4005.D3FC
GPIO Port G	0x4005.E3FC
GPIO Port H	0x4005.F3FC

GPIO Port	Endereço
GPIO Port J	0x4006.03FC
GPIO Port K	0x4006.13FC
GPIO Port L	0x4006.23FC
GPIO Port M	0x4006.33FC
GPIO Port N	0x4006.43FC
GPIO Port P	0x4006.53FC
GPIO Port Q	0x4006.63FC

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  
   ; anterior da porta inteira  
3  
4 ORR R0, R0, #0x80 ; Faz o OR bit a bit para  
   manter os valores  
   ; anteriores e setar somente o bit  
5  
6 STR R0, [R1] ; Escreve o novo valor da porta
```

Escrita Amigável nos GPIO

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

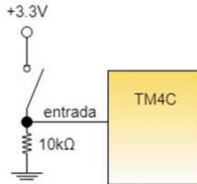
```
1 LDR R1, =GPIO_PORTK_DATA_R ;Carrega-se o  
   endereço  
2 LDR R0, [R1] ; Lê para carregar o valor  
3               ; anterior da porta inteira  
4 BIC R0, R0, #0x80 ; Faz o AND negado bit a bit  
   para manter os  
5                   ; valores anteriores e limpar  
                   somente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

Chaves e LEDs

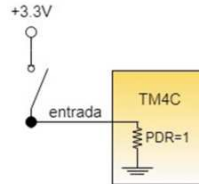
Entrada com Chaves

- Há as seguintes formas de interfacear com uma chave:

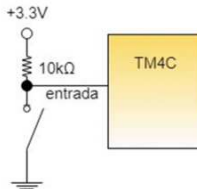
Lógica positiva, resistor externo



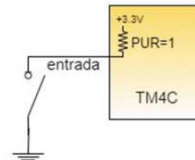
Lógica positiva, resistor interno



Lógica negativa, resistor externo



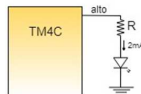
Lógica negativa, resistor interno



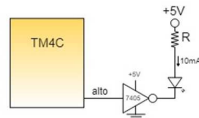
Saída com LEDs

- Há as seguintes formas de interfacear com LEDs:
- Uma porta no TM4C1294 suporta no máximo 12mA, mas o microcontrolador não suporta todas as portas drenando/suprindo este máximo de corrente para todas as portas. (Ver Datasheet seção 27.3.2.1);
- O valor **default** é cada porta drenar/suprir 2mA por porta;
- Se precisar que uma porta forneça mais corrente que ela suporta, deve-se utilizar um driver com circuito integrado ou transistor.

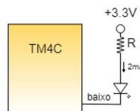
Lógica positiva, corrente baixa



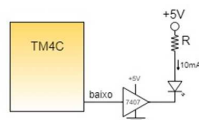
Lógica positiva, corrente alta



Lógica negativa, corrente baixa



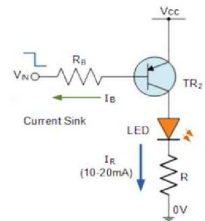
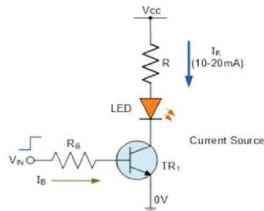
Lógica negativa, corrente alta



Saída com LEDs

- Para drivers com CIs:
 - Para lógica positiva: Exemplo → 74xx05 ou 74xx06;
 - Para lógica negativa: Exemplo → 74xx07.

- Para transistores, usar operação como chave:
 - Calcular R_C e R_B ;
 - Região corte-saturado;
 - Como calcular?



Saída com LEDs

- Para transistores, operação como chave:

- Calcular R_C e R_B ;
- Região corte-saturado;

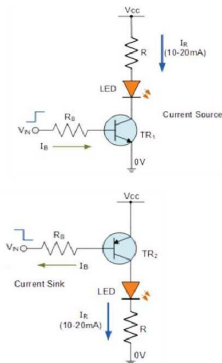
$$I_C = \frac{V_{CC} - V_{LED} - V_{CE}}{R_C}$$

$$I_B = \frac{I_C}{\beta}$$

$$I'_B = 5I_B \quad (\text{para garantir a saturação})$$

$$R_B = \frac{V_{IN} - V_{BE}}{I'_B}$$

(1)



Exercício: Instruções de Memória/Transfer

1 Exemplo de inicialização do GPIO.

Verificar a inicialização da porta J e porta F. Os pinos J0 e J1 estão ligados às chaves tácteis USR_SW1 e USR_SW2, respectivamente e os pinos F4 e F0 estão ligados aos LEDs 3 e 4, respectivamente.

- a) Baixe e abra o projeto GPIO1 no moodle;
- b) Faça o *build* e execute passo-a-passo para verificar a inicialização das portas;
- c) Execute e teste pressionando os dois botões e verificar se os LEDs acendem;
- d) Faça um fluxograma do código.

Clock e SysTick

Clock

- Microcontroladores podem ter *clock*:
 - Externo: provido geralmente por um cristal;
 - Interno: geralmente um oscilador R-C mais impreciso e de menor frequência.

PPL

- Normalmente a velocidade de execução de um microcontrolador é determinada pelo cristal externo, no caso do TM4C1294 é 25MHz;
- Muitos microcontroladores possuem um **PLL** que possibilita **ajustar a velocidade** de execução por software;
- Normalmente a frequência é um *tradeoff* entre velocidade de execução e potência elétrica;
- Para informações avançadas sobre o gerenciamento de *clock* (Seção 5.2.5 do *Datasheet*).

SysTick

- Contador simples para gerar atrasos e gerar interrupções periódicas em todos os Cortex-M;
- Fácil de portar para outros microcontroladores;
- 4 passos para ativar:
 - 1 Limpr o bit **ENABLE** no registrador **NVIC_ST_CTRL_R** para desligar o SysTick durante a inicialização;
 - 2 Sete o registrador **NVIC_ST_RELOAD_R**;
 - 3 Escreva qualquer valor no registrador **NVIC_ST_CURRENT_R** para limpar o contador.
 - 4 Sete os bits **CLK_SRC** e **ENABLE** no registrador **NVIC_ST_CTRL_R**. Como interrupções ainda não serão tratadas, não é necessário setar o bit **INTEN**.

SysTick

- Quando a contagem no registrador **CURRENT** mudar de 1 para 0, o *flag* **COUNT** será setado;
- Se ativar o PLL para rodar o microcontrolador em 80MHz então o contador do SysTick decrementa a cada 12,5 ns;
- Em geral se o período do *clock* de barramento é t o *flag* de contagem será setado a cada $(n + 1)t$, tal que n é o valor do registrador **RELOAD**;
- Se escrever no registrador **CURRENT**, o contador será zerado e o *flag* de contagem no registrador CTRL será limpo.

Exercício: Instruções de Memória/Transfer

2. Piscar um LED.

Baseando-se no exemplo anterior, criar um projeto que pisque um LED a cada intervalo, quando pressionado um botão.

- a) Baixe o projeto gpio2 do moodle;
- b) Abra o projeto no *Keil MDK*;
- c) Faça um fluxograma do problema proposto;
- d) Modifique o arquivo gpio.s para inicializar os GPIO para uma chave e um LED;
- e) Modifique o arquivo main.s para fazer o que foi pedido no enunciado;
- f) Primeiramente, faça apenas o LED acender;
- g) Depois que esta parte estiver pronta, faça o LED piscar;
- h) Para fazer o LED piscar utilize a rotina SysTick_Wait1ms.

Dúvidas?