EA871

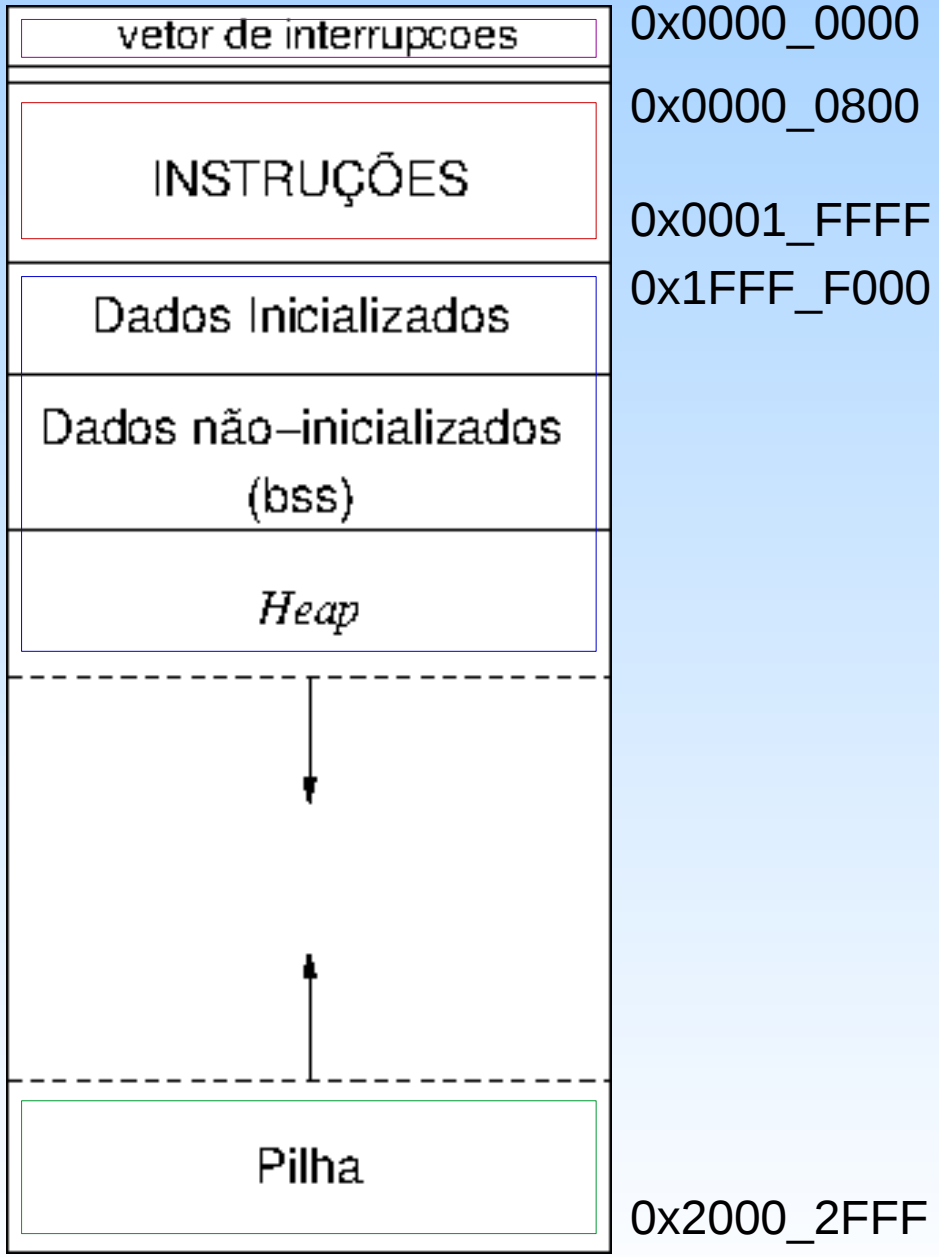
Hello FRDM-KL25Z!

Declaração e Definição de Dados em C

Wu Shin – Ting
DCA – FEEC - Unicamp
Segundo Semestre de 2020

Organização da Memória

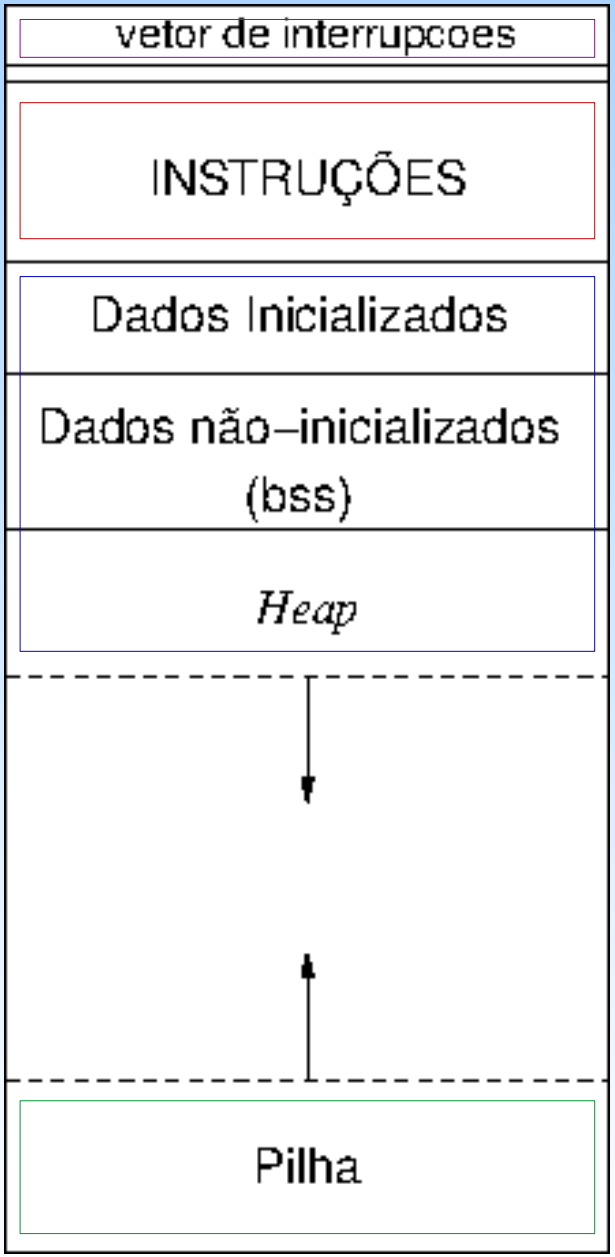
- Segmento de instruções (FLASH): contém os códigos **binários** de operação que geram sinais de controle ao longo de um fluxo de execução.
- Segmento de dados (RAM): contém os operandos **binários** sobre os quais os sinais de controle atuam.
- Pilha (RAM): contém os dados de trabalho, temporários, ao longo da execução de um programa.
- Vetor de interrupções (FLASH): contém os endereços das rotinas de serviço às interrupções.



Conceitos

- **Declaração numa linguagem de alto nível**: informar ao compilador as propriedades de um objeto que requer um espaço de memória.
- **Definição numa linguagem de alto nível**: atribuir um espaço físico na memória para acomodar efetivamente o objeto.
- **Variáveis**: nomes dos espaços de memória ocupados pelos dados.

Conceitos



0x0000_0000

0x0000_0800

0x0001_FFFF

0x1FFF_F000

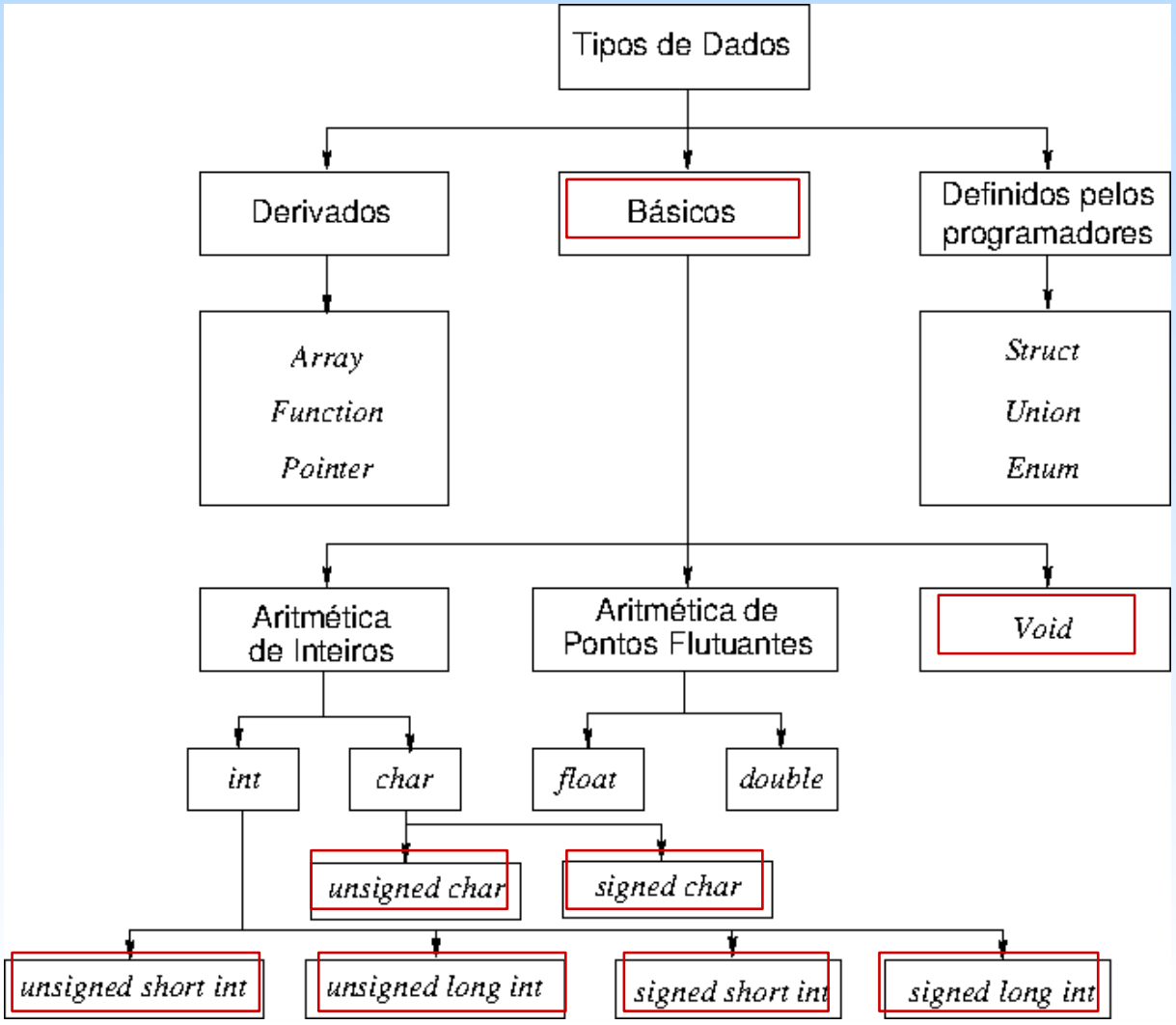
0x2000_2FFF

Declaração de variáveis: informar o compilador sobre o nome, o **tipo de dado** e o seu valor inicial, a fim de que seja alocado apropriadamente um espaço na memória .

```
char c;
int counter=0;
float x;
double xx;
```

Conceitos

- **Tipos de dados**: especificam o tamanho de memória requerido, a forma de interpretação dos padrões de *bits* que ocupam este espaço e as operações válidas sobre estes *bits*.

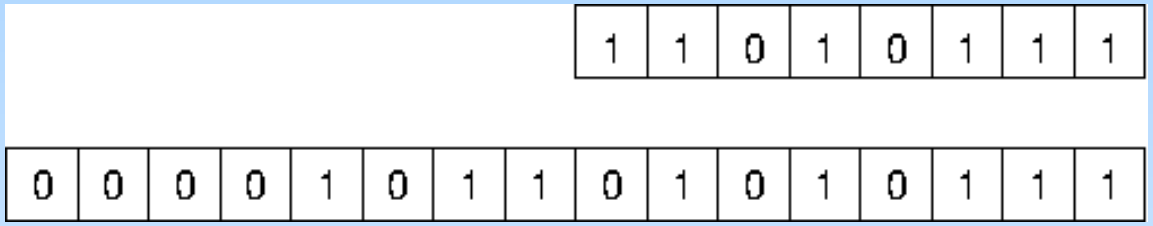


Conceitos

Tipo de dado	Tamanho típico	Interpretação	Intervalo típico
char	1 *byte*	Código ASCII	0 – 255
unsigned **char**	1 *byte*	Código ASCII	0 - 255
signed char	1 *byte*	Complemento de 2	-128 - 127
signed **short** int	2 *bytes*	Complemento de 2	-32768 - 32767
unsigned **short** int	2 *bytes*	Código binário	0 - 65535
int	4 *bytes*	Complemento de 2	-2147483648 - 2147483647
unsigned int	4 *bytes*	Código binário	0 - 4294967295
float	4 *bytes*	IEEE 754	- .1*2^{-128} - .1*2^{127}
double	8 *bytes*	IEEE 754	-.1*2^{-1024} - .1*2^{1023}
void	sem valor associado		

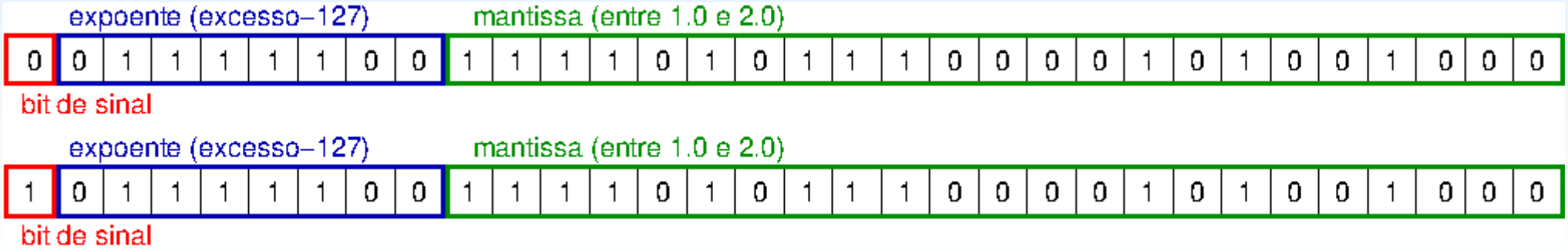
Conceitos

- **Código Binário**: representação dos valores na base binária
  - 215
  - 2895



- **Complemento de 2**
  - **Positivo**: *bit* de sinal + código binário
  - **Negativo**: *bit* de sinal + complemento de 2

- **IEEE754**
  - $0.245 = 1.96000003814697 \times 2^{-3} = (1+2^{-1}+2^{-2}+2^{-3}+2^{-4}+2^{-6}+2^{-8}+2^{-9}+2^{-10}+2^{-15}+2^{-17}+2^{-20}) \times 2^{-3}$



Conceitos

- **Classe de armazenamento:** especifica o escopo de validade/visibilidade e tempo de vida de uma variável ao longo de uma execução.

Especificadores de classe	Local de armazenamento	Valor inicial	Escopo	Tempo de vida
register	Registradores	Não-inicializado	Função/Rotina	Até final da rotina
auto	pilha	Não-inicializado	Função/Rotina	Até final da rotina
extern	Segmento de dados	zero	Programa	Até final do programa
static	Segmento de dados	zero	Função/Rotina	Até final do programa

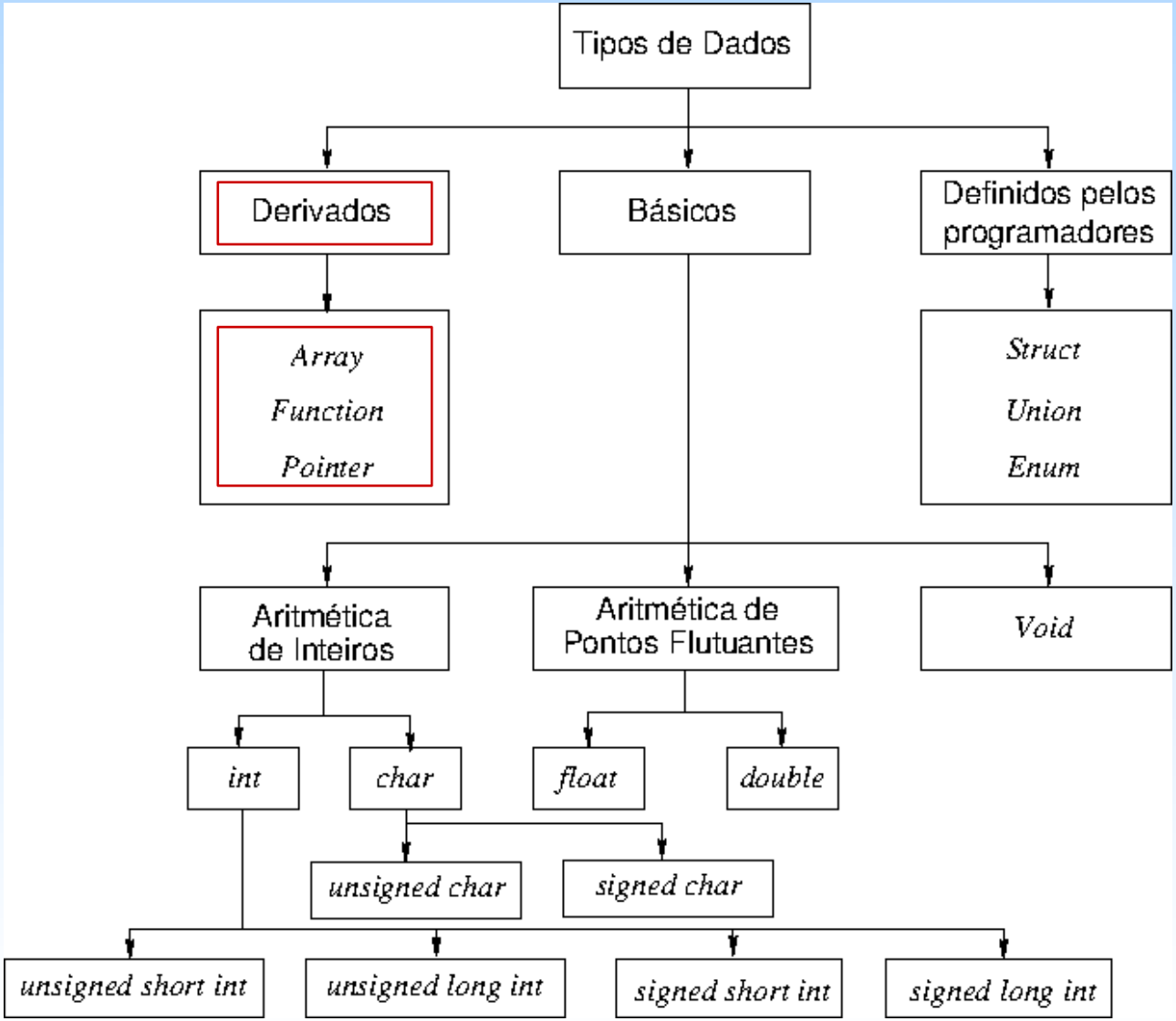
  - **Variáveis locais**: conteúdos preservados no escopo de rotina.
  - **Variáveis globais**: conteúdos preservados no escopo de aplicativo.

Conceitos

- **Modificadores de acesso às variáveis**: mudam a maneira como uma variável é acessada e modificada.
  - **const**: o conteúdo da variável não é alterado ao longo da execução de um programa.
  - **volatile**: o conteúdo da variável pode ser alterado por ações não programadas.

Conceitos

- **Tipos de dados derivados**: são os tipos de dados construídos a partir dos tipos básicos, integrados em C.



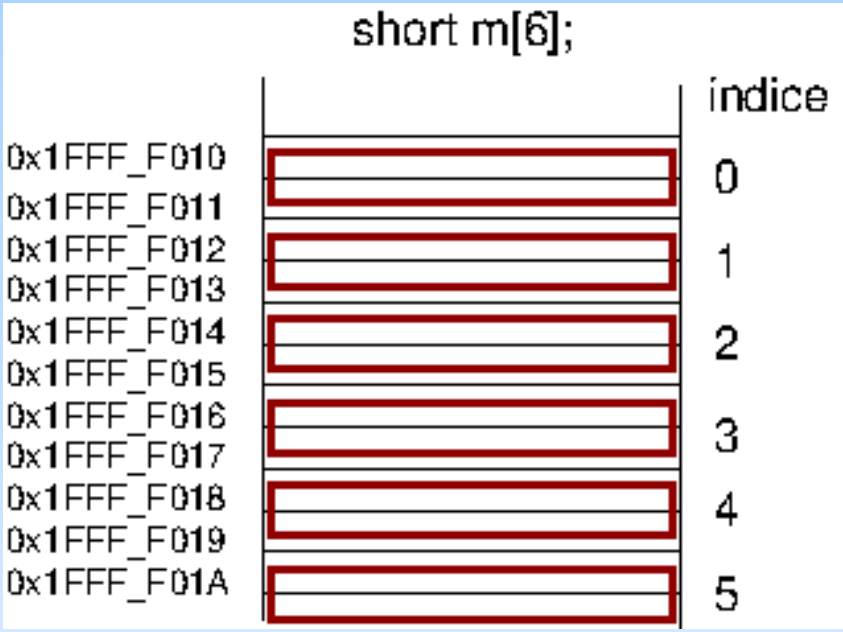
Conceitos

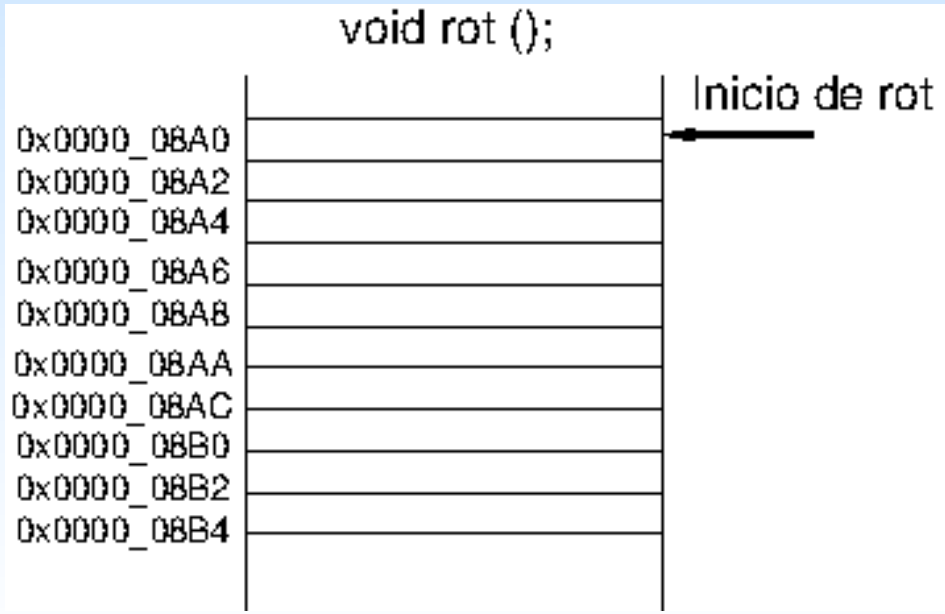
- ***Array***: consiste de uma lista de elementos de um mesmo tipo de dados. Cada elemento é acessível pelo índice

  <tipo> nome[<elementos>];

- **Função**: consiste de uma sequência de instruções. No nosso caso, instrução (Thumb) ocupa 2 bytes.

  <tipo> nome (<argumentos>);





Conceitos

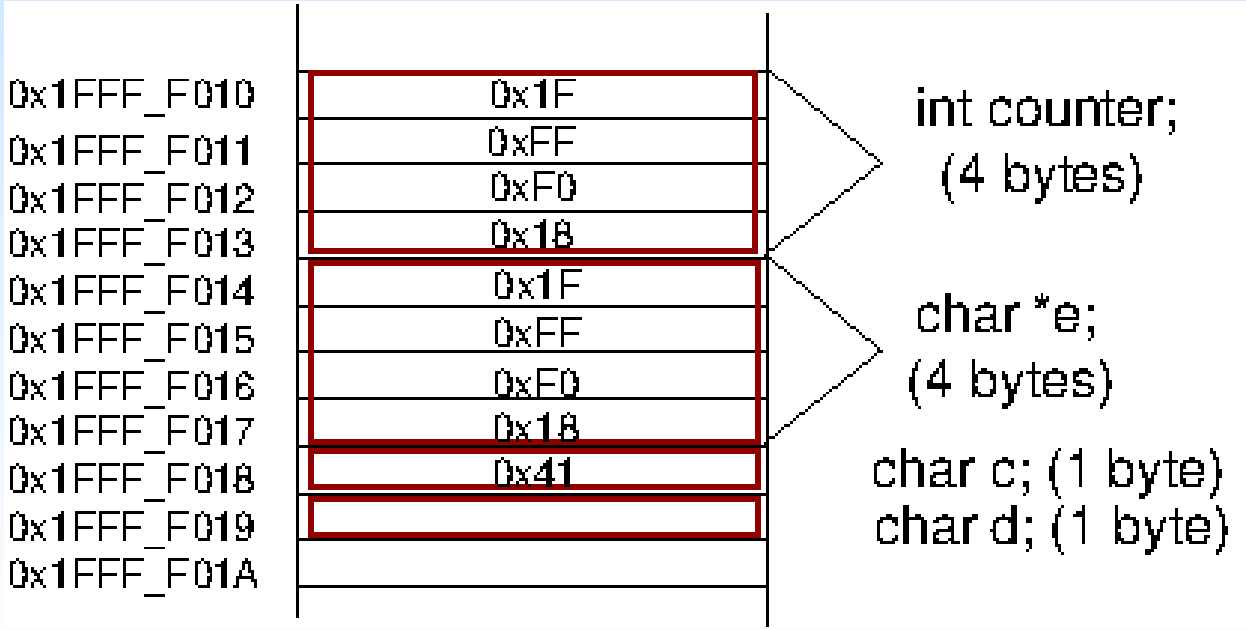
- **Operador Unário &**: é o operador de acesso ao endereço de uma variável.

  <endereço> = &<var_name>;

- **Ponteiro**: declara o endereço de um espaço de memória ocupado por uma variável de qualquer tipo de dado.

  <tipo> * nome;

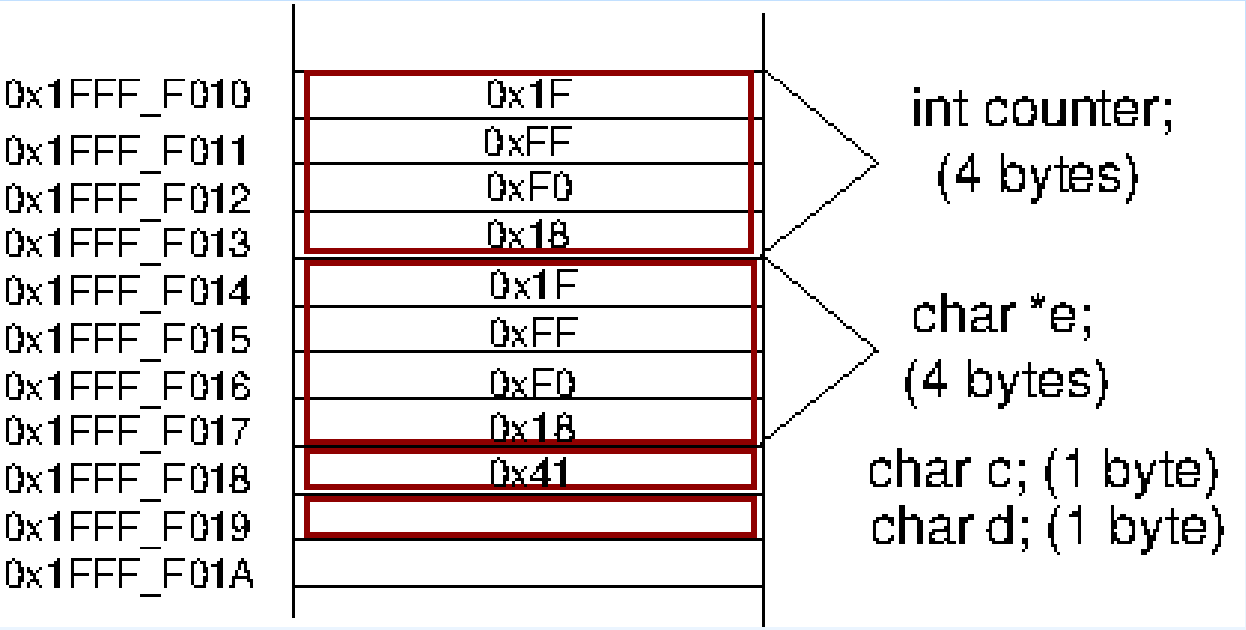
Espaço de memória endereçável por 32 bits → ponteiros ocupam 4 *bytes*.



Conceitos

- **Operador Unário ***: é o operador de acesso ao conteúdo do endereço que está armazenado num ponteiro.

  (*e) equivale ao conteúdo de (0x1FFFF_F018): 0x41



CodeWarrior IDE Development Suite

Informações Adicionais

- Curso C – Tipos de dados avançados:
  http://mtm.ufsc.br/~azeredo/cursoC/aulas/ca00.html
- Difference between Definition and Declaration
  https://www.geeksforgeeks.org/difference-between-definition-and-declaration/
- C – Data Types:
  https://www.tutorialspoint.com/cprogramming/c_data_types.htm
- IEEE754 Floating Point Converter
  https://www.h-schmidt.net/FloatConverter/IEEE754.html
- C – Variables:
  https://www.tutorialspoint.com/cprogramming/c_variables.htm
- C – Constants and Literals:
  https://www.tutorialspoint.com/cprogramming/c_constants.htm

Informações Adicionais

- C- Storage Classes

  https://www.tutorialspoint.com/cprogramming/c_storage_classes.htm
- C – Arrays:

  https://www.tutorialspoint.com/cprogramming/c_arrays.htm
- C – Functions:

  https://www.tutorialspoint.com/cprogramming/c_functions.htm
- C – Pointers:

  https://www.tutorialspoint.com/cprogramming/c_pointers.htm
- Linguagem C: Representação de Dados

  ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/apostila_C/RepresentacaoDados.pdf

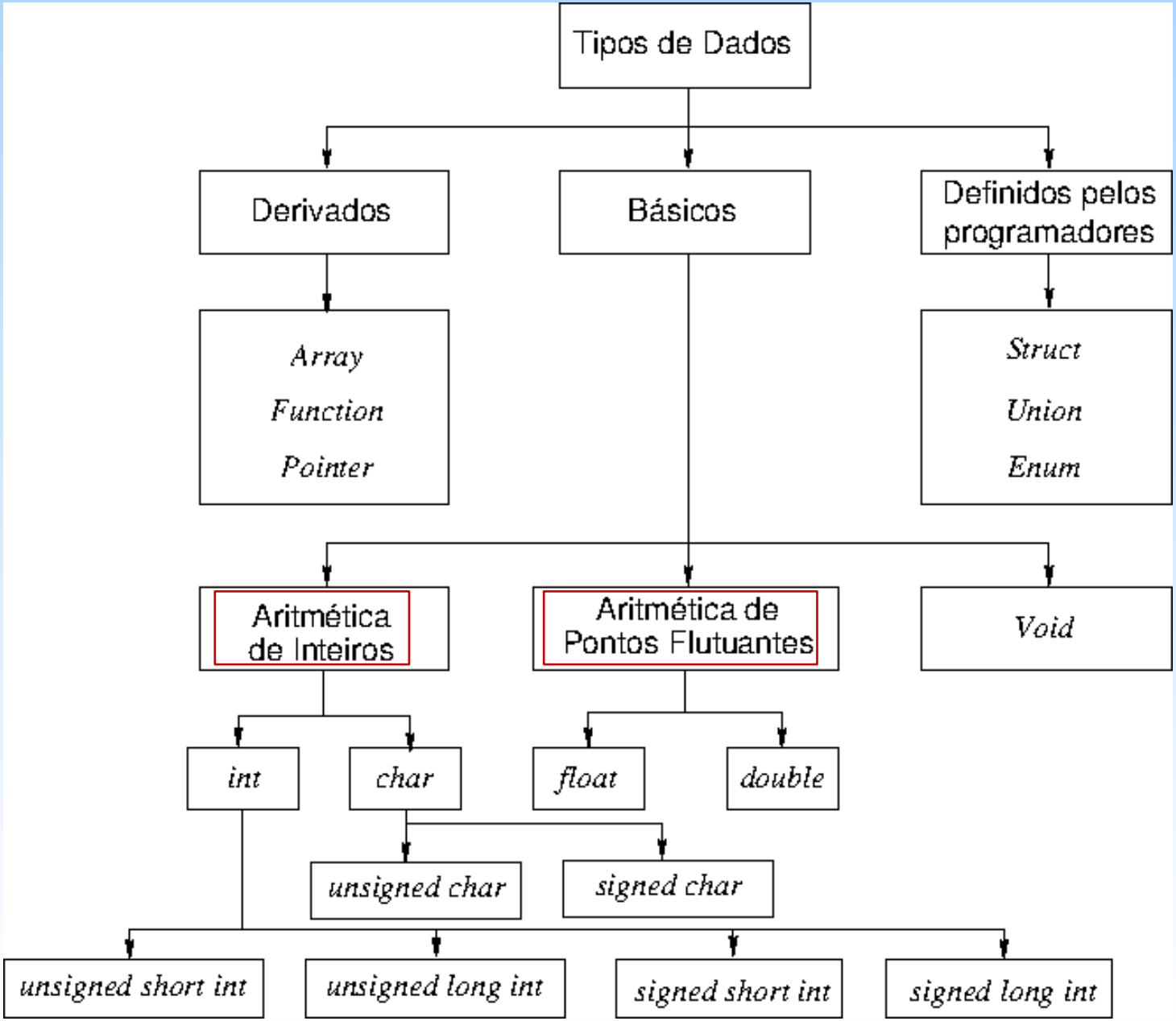
EA871

Hello FRDM-KL25Z!

Conversão de Tipos de Dados

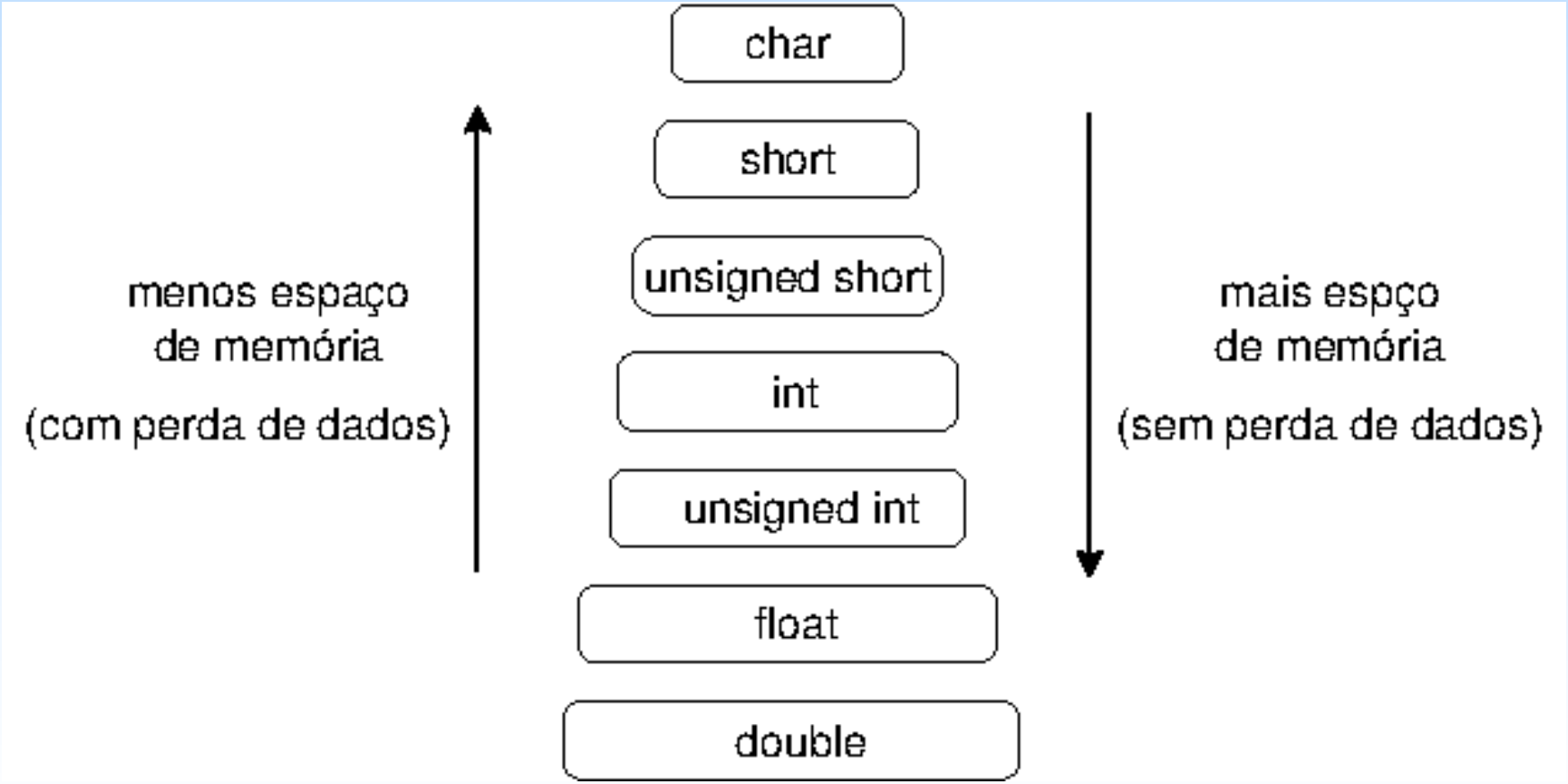
Wu Shin – Ting
DCA – FEEC - Unicamp
Segundo Semestre de 2020

Inteiros e Pontos Flutuantes



Conceitos

- **Conversão Implícita**: é uma conversão pré-definida no processador, não requerendo nenhum operador especial. Consiste em converter todos os dados para um mesmo tipo antes de processar uma expressão aritmética.



Conceitos

- **Conversão explícita (***casting*) é uma conversão em que o programador explicitamente redefine o tipo de dado de um objeto. A sintaxe de conversão é

  <novoTipo> <variável>

  - Atenção com perda de dados quando se passa de um tipo de dado que ocupa mais espaço de memória para um que ocupa menos espaço.

Exemplos

- **Atribuição de um valor**

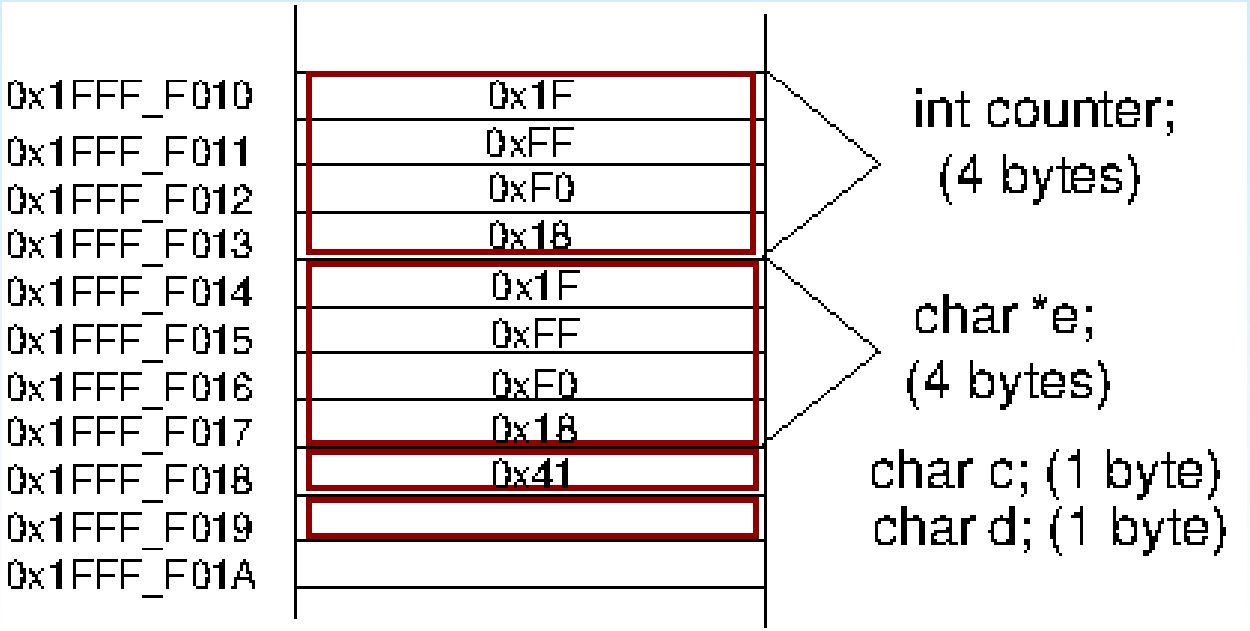
  e = &c;

  e = (char *)(0x1FFFF018);

- **Acesso ao conteúdo do endereço num ponteiro**

  d = *e;

  d = (*((char *)0x1FFFF018));



CodeWarrior IDE Development Suite

Informações Adicionais

- Type Conversion in C

  https://www.geeksforgeeks.org/type-conversion-c/

- Linguagem C: Representação de Dados

  ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/apostila_C/RepresentacaoDados.pdf

EA871

Hello FRDM-KL25Z!

Operadores *bit* a *bit*

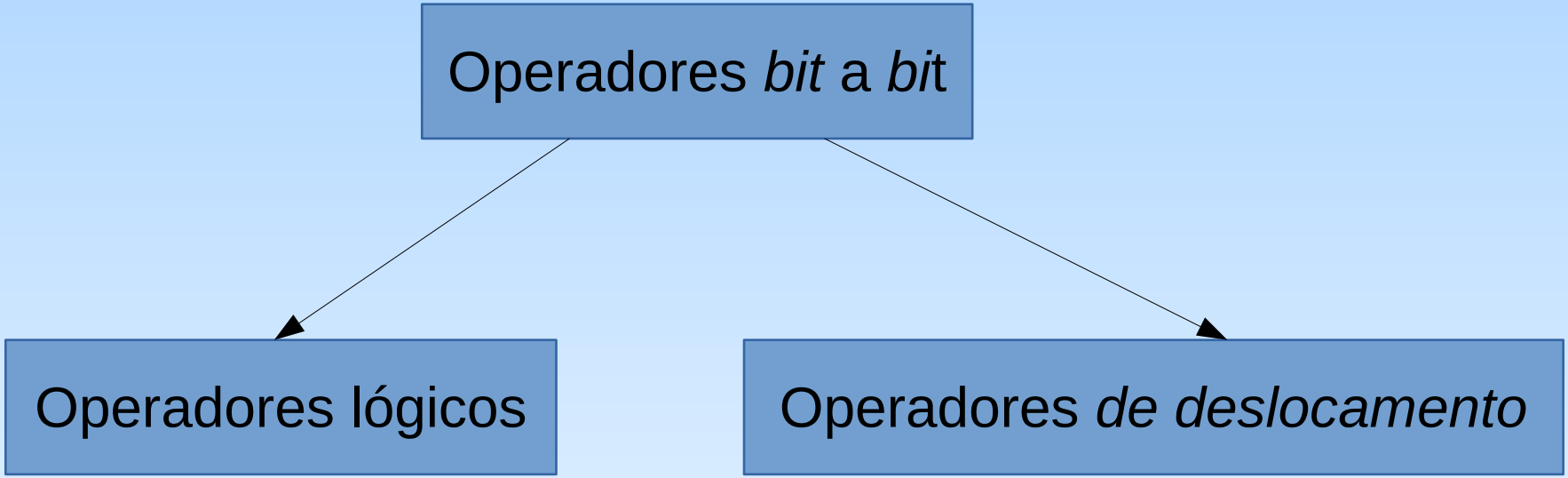
Wu Shin – Ting
DCA – FEEC - Unicamp
Segundo Semestre de 2020

Conceitos

- **Operações lógicas** sobre dois elementos 0 e 1 da Álgebra Booleana.

p	q	&	|	^	~p	~q
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	0	0	1	1	0	1
1	1	1	1	0	0	0

Conceitos


Operadores *bit* a *bi*t
Operadores lógicos
Operadores *de deslocamento*

Conceitos

- **Operadores lógicos *bit* a *bit***: são operadores binários que aplicam as operações lógicas a nível de *bit*.

Operadores *bit* a *bit*	Operação lógica
&	E
|	OU
~	NEGAÇÃO
^	OU Exclusivo

Exemplos

- Operador E (AND)

```
  1 0 1 0 1 1 0 0
& 0 0 0 1 1 1 1 1
-----------------
  0 0 0 0 1 1 0 0
```

- Operador OU (OR)

```
  1 0 0 0 0 0 1 0
| 0 0 0 1 1 1 0 0
-----------------
  1 0 0 1 1 1 1 0
```

- Operador Ou Exclusivo (XOR)

```
    1 0 0 1 0 1 0 1
^ 0 0 0 1 1 1 1 0
-----------------
  1 0 0 0 1 0 1 1
```

- Operador Negação (NOT)

```
~ 1 0 0 1 0 1 0 1
-----------------
  0 1 1 0 1 0 1 0
```

Conceitos

- **Máscara**: é um conjunto de *bits* que modificam um campo de *bits* de um operando.
- **Mascaramento**: é aplicação de uma máscara sobre um dado de forma que um campo específico de *bits* seja alterado.

- Mascarando *bits* para 1

```
  1 0 1 0 0 1 0 1
| 0 0 1 1 1 0 0 0
-----------------
  1 0 1 1 1 1 0 1
```

- Mascarando *bits* para 0

```
  1 0 1 0 0 1 0 1
& 1 1 0 0 0 1 1 1
-----------------
  1 0 0 0 0 1 0 1
```

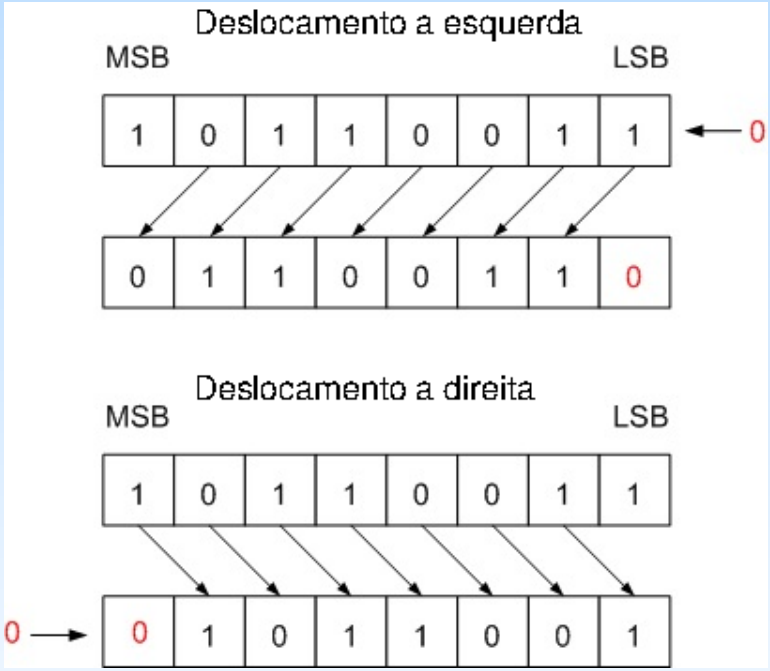
Conceitos

- **Operadores de deslocamento *bit* a *bit***: são operadores binários que deslocam os *bits* de um operando.

Operadores *bit* a *bit*	**Operação**
<<	Deslocamento para esquerda
>>	Deslocamento para direita

Conceitos

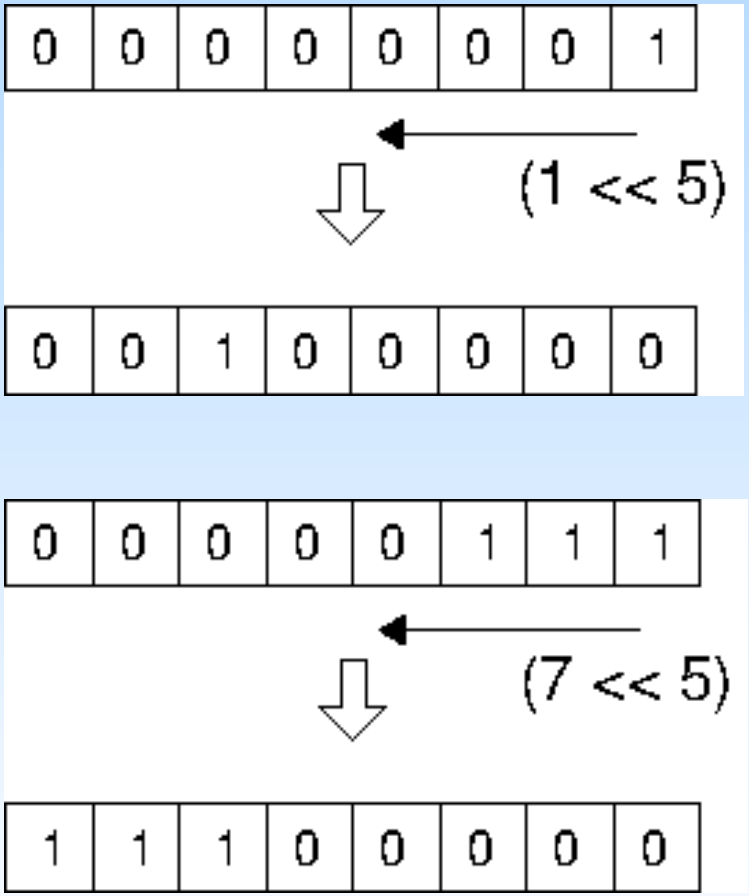
- **Deslocamentos lineares:** *bit* que sai do espaço da variável é discartado.



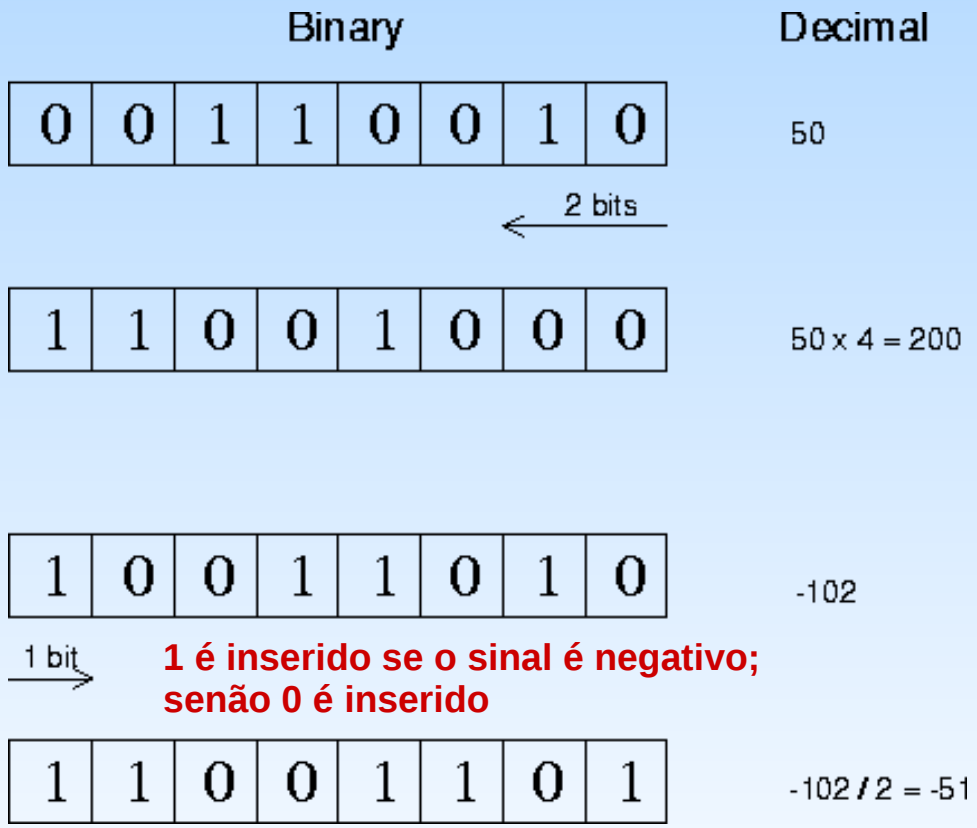
- **Deslocamentos rotativos:** *bit* que sai do espaço da variável é inserido no outro lado.

Conceitos

- **Construção de máscara**



- **Operações em potência de 2**



CodeWarrior IDE Development Suite

Informações Adicionais

- Bitwise Operators in C

  https://www.tutorialspoint.com/cprogramming/c_bitwise_operators.htmC
- Bitwise Operator

  http://www.c4learn.com/c-programming/c-bitwise-masking/
- Linguagem C: Representação de Dados

  ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/apostila_C/RepresentacaoDados.pdf

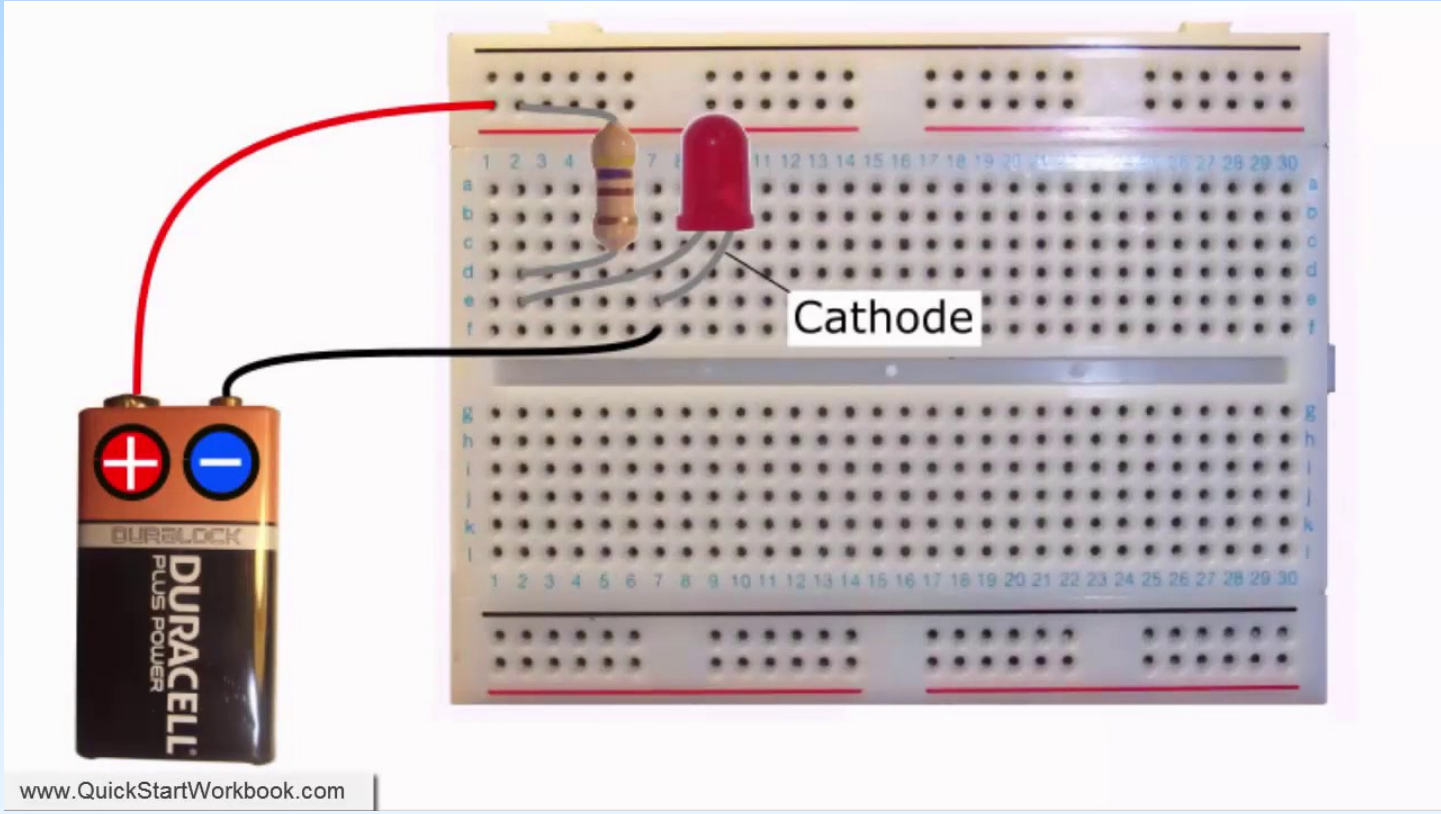
EA871

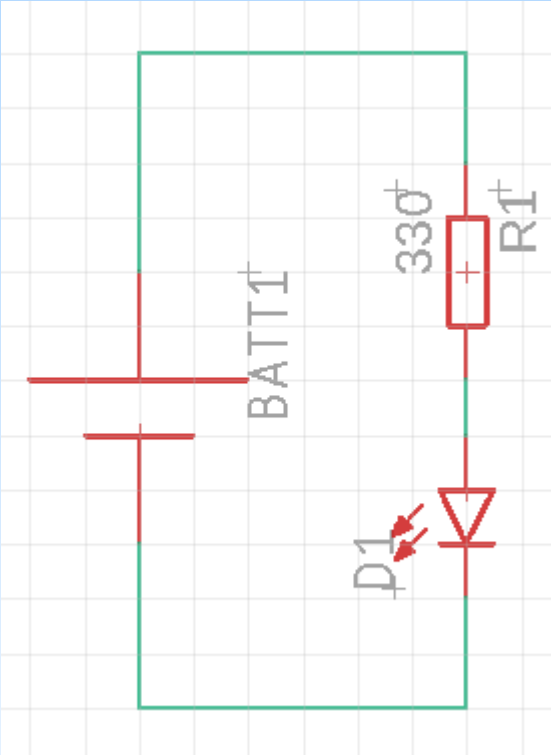
Hello FRDM-KL25Z!

Módulos SIM, PORT e GPIO no controle de um *led* piscante

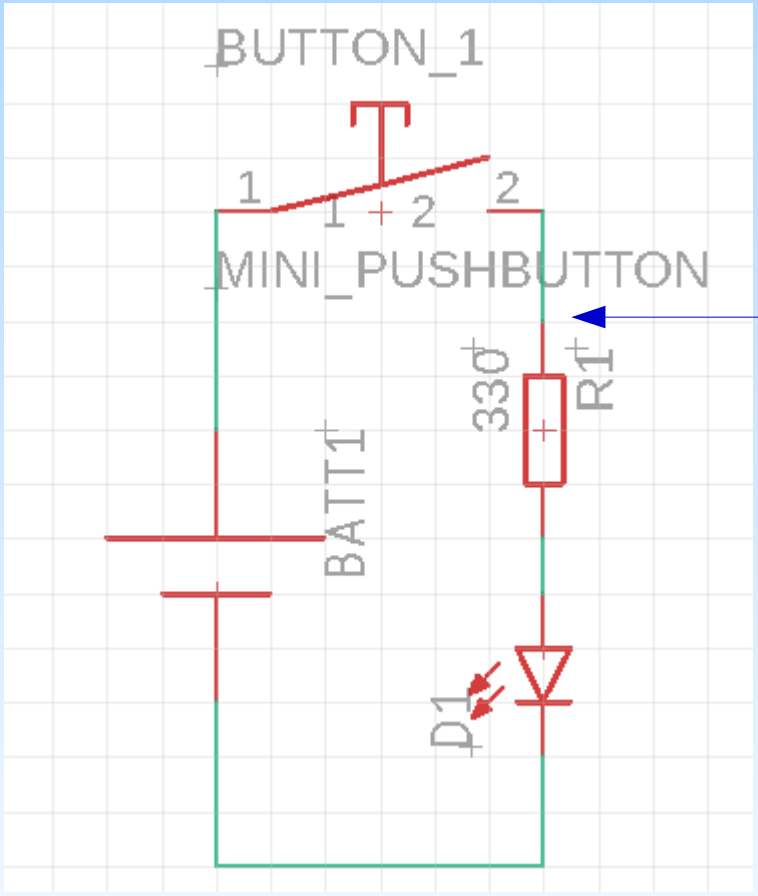
Wu Shin – Ting
DCA – FEEC - Unicamp
Segundo Semestre de 2020

Circuito de *Led*



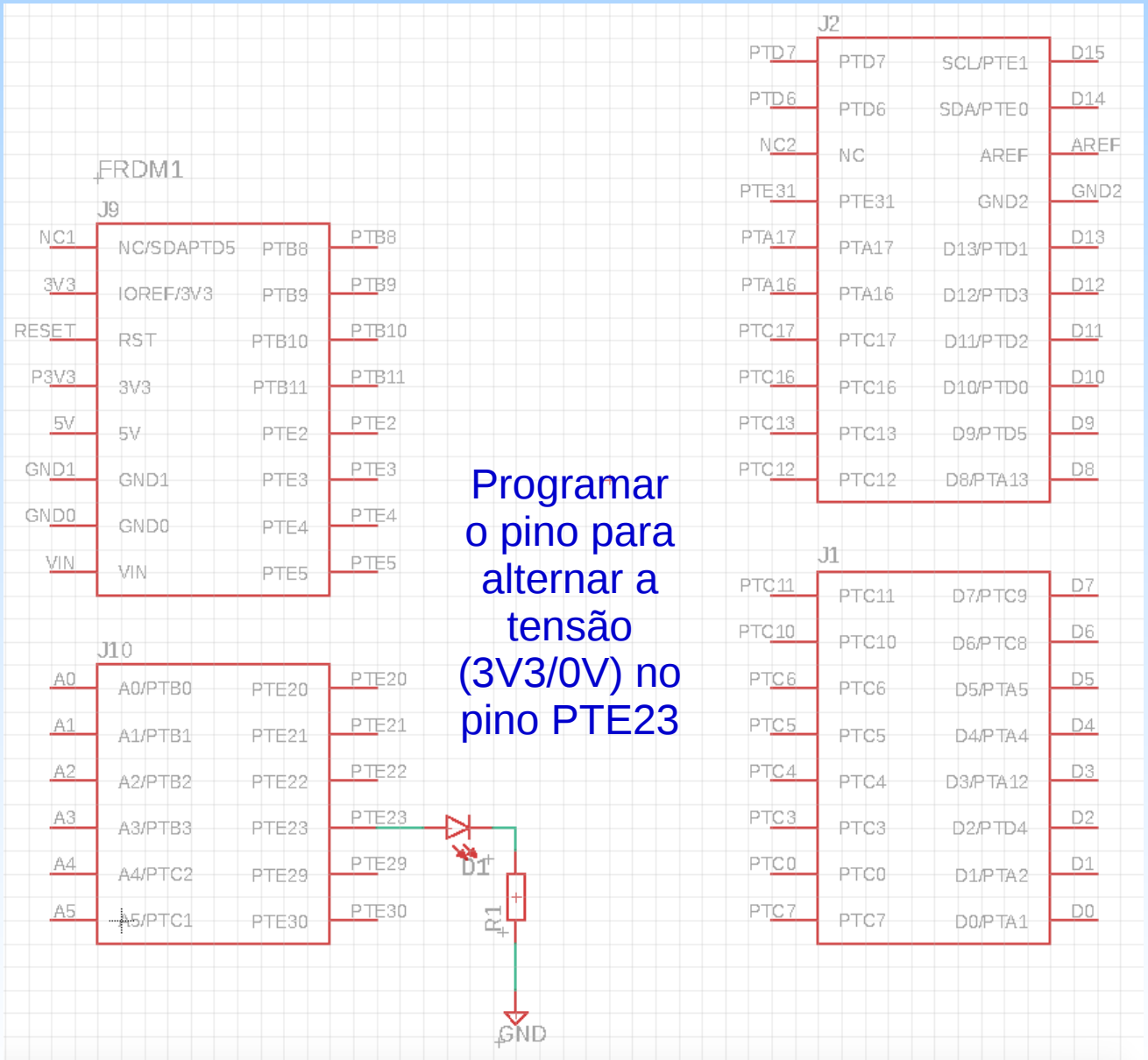


Circuito de *Led* Piscante controlado pela chave

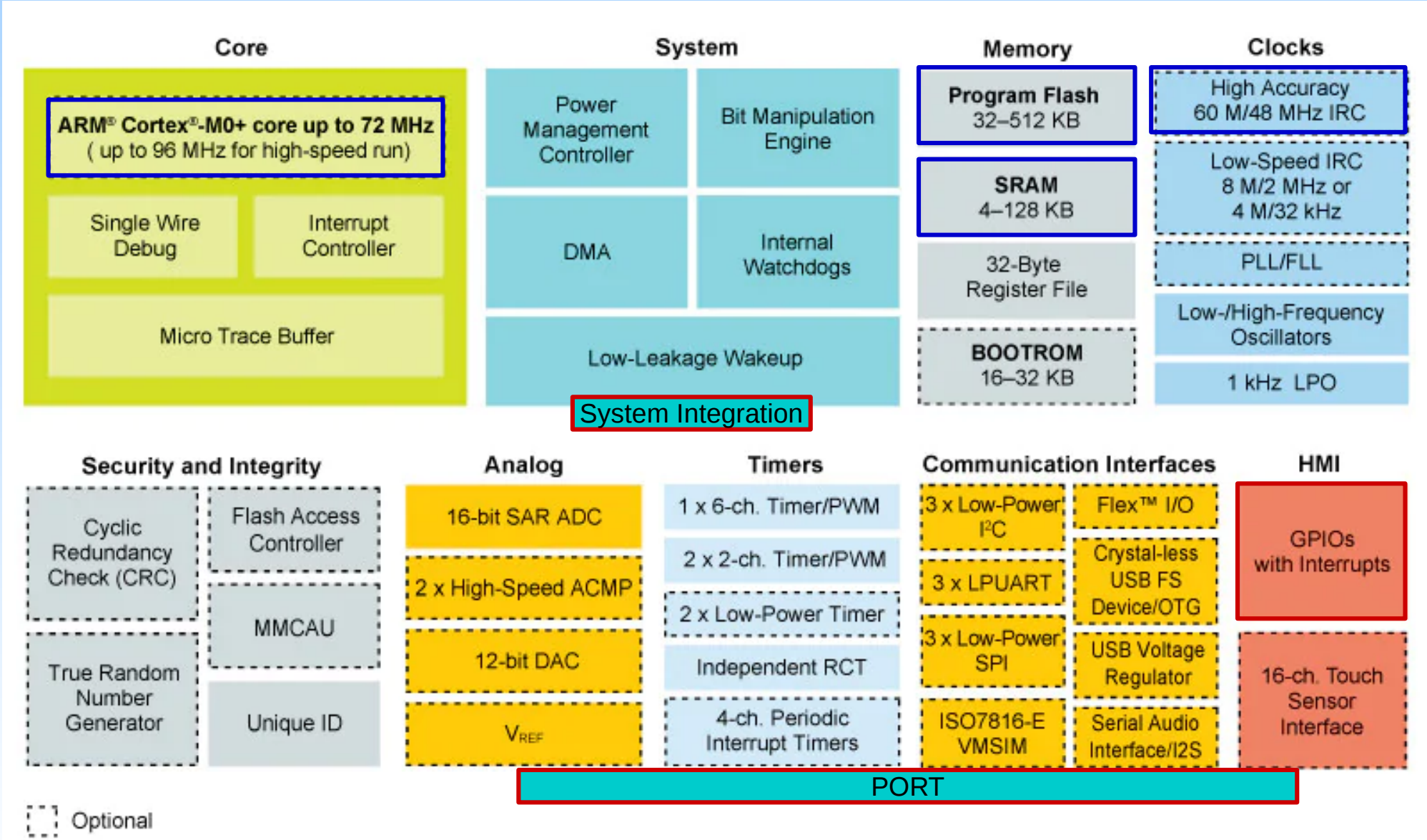


Fecha chave: 3V3
Led acende
Abre chave: 0V
Led apaga

Circuito de *Led* Piscante microcontrolado

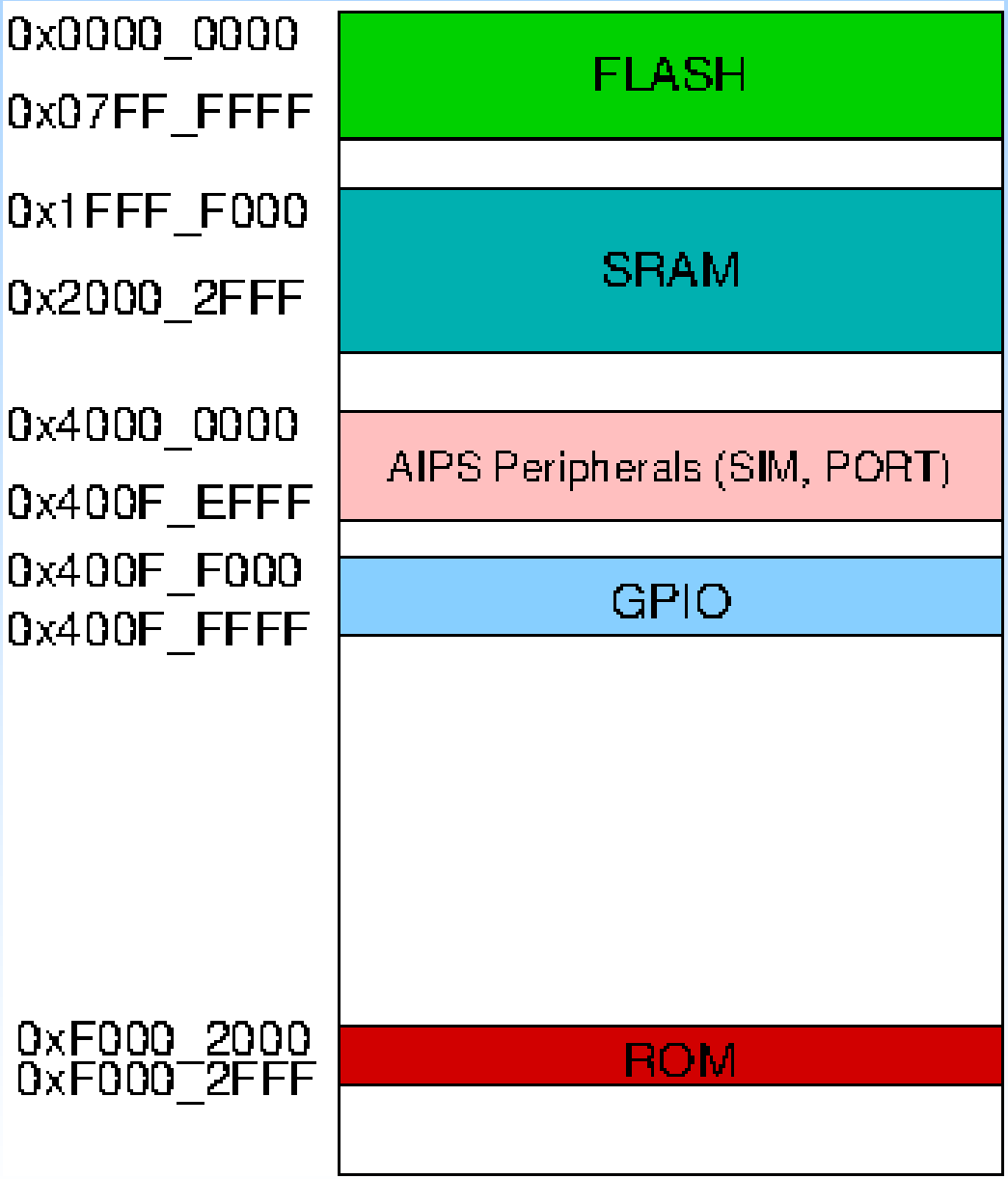


Microcontrolador



Conceitos

- **Mapeamento no espaço de memória**: os registradores dos módulos são acessados usando os endereços do espaço de memória do microcontrolador.



Conceitos

- **Módulo SIM** (*System Integration Module*): inclui a lógica de integração dos módulos do microcontrolador, responsável pela configuração do *chip* e seleção do *clock* de cada módulo.

Address: 4004_7000h base + 1038h offset = 4004_8038h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												0		0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		PORTE	PORTD	PORTC	PORTB	PORTA	1		0	TSI		0		0	LPTMR
W																
Reset	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0

SIM_SCGC5 field descriptions

Endereços na página 192 do *Reference Manual*

Conceitos

- **Módulo PORT**: suporta o controle dos 32 pinos de uma porta, incluindo as funções do pino, as interrupções externas associadas a ele e o seu estado.

Address: Base address + 0h offset + (4d × i), where i=0d to 31d = 4004_9000h + 4000h + (4dx21d) = 4004_D054h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0							ISF	0				IRQC			
W								w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0					MUX			0	DSE	0	PFE	0	SRE	PE	PS
W																
Reset	0	0	0	0	0	x*	x*	x*	0	x*	0	x*	0	x*	x*	x*

* Notes:
- x = Undefined at reset.

PORTE_PCR23

PORTx_PCR*n* field descriptions

Endereços na página 177 do *Reference Manual*

Conceitos

- **Multiplexação dos sinais**: cada pino físico pode servir até 8 diferentes módulos para comunicação do microcontrolador com o mundo externo.

80 LQFP	64 LQFP	48 QFN	32 QFN	Pin Name	Default	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7
3	—	—	—	PTE2	DISABLED		PTE2	SPI1_SCK					
4	—	—	—	PTE3	DISABLED		PTE3	SPI1_MISO			SPI1_MOSI		
5	—	—	—	PTE4	DISABLED		PTE4	SPI1_PCS0					
6	—	—	—	PTE5	DISABLED		PTE5						
7	3	1	—	VDD	VDD	VDD							
8	4	2	2	VSS	VSS	VSS							
9	5	3	3	USB0_DP	USB0_DP	USB0_DP							
10	6	4	4	USB0_DM	USB0_DM	USB0_DM							
11	7	5	5	VOUT33	VOUT33	VOUT33							
12	8	6	6	VREGIN	VREGIN	VREGIN							
13	9	7	—	PTE20	ADC0_DP0/ ADC0_SE0	ADC0_DP0/ ADC0_SE0	PTE20		TPM1_CH0	UART0_TX			
14	10	8	—	PTE21	ADC0_DM0/ ADC0_SE4a	ADC0_DM0/ ADC0_SE4a	PTE21		TPM1_CH1	UART0_RX			
15	11	—	—	PTE22	ADC0_DP3/ ADC0_SE3	ADC0_DP3/ ADC0_SE3	PTE22		TPM2_CH0	UART2_TX			
16	12	—	—	PTE23	ADC0_DM3/ ADC0_SE7a	ADC0_DM3/ ADC0_SE7a	PTE23		TPM2_CH1	UART2_RX			

PORTE_PCR23[MUX] = 001b

Endereços na página 162 do *Reference Manual*

Conceitos

- **Módulo GPIO** (*General Purpose Input/Output*): permite que o processador comunique com o mundo externo através de uma interface de 0 estado de espera por meio de sinais digitais.

Address: Base address + 14h offset = 4004_F000h + (40hx 4d) + 14h= 4004_F114h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PDD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_PDDR field descriptions **Sentido: Entrada (0)/Saída(1)**

Address: Base address + 0h offset = 4004_F000h + (40hx 4d) + 0h= 4004_F100h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PDO																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_PDOR field descriptions **Saída**

Address: Base address + 10h offset = 4004_F000h + (40hx 4d) + 10h= 4004_F110h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PDI																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_PDIR field descriptions **Entrada**

Conceitos

Address: Base address + 4h offset = 4004_F000h + (40hx 4d) + 4h= 4004_F104h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																															
W	PTSO																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_PSOR field descriptions ***Set***

Address: Base address + 8h offset = 4004_F000h + (40hx 4d) + 8h= 4004_F108h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																															
W	PTCO																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_PCOR field descriptions ***Clear***

Address: Base address + Ch offset = 4004_F000h + (40hx 4d) + Ch= 4004_F10Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																															
W	PTTO																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_PTOR field descriptions ***Toggle***

Endereços na página 773 do *Reference Manual*

Pseudocódigo

(1)Configurar os módulos para que o pino PTE23 seja um pino digital de saída.

(2)Início do laço infinito

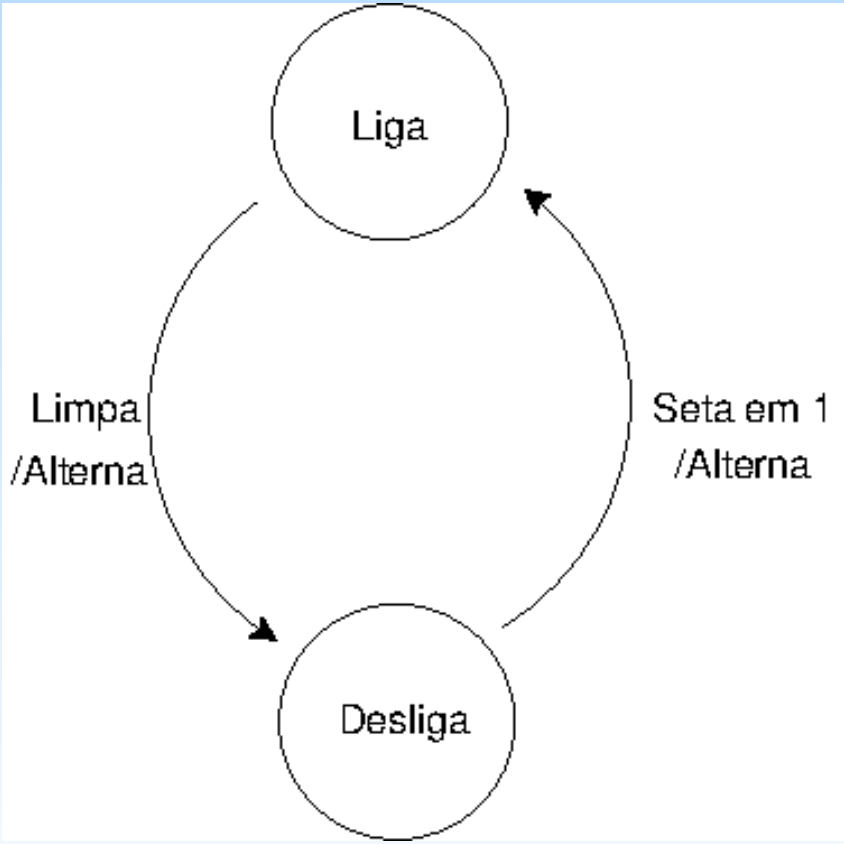
(3) Ligar o led

(4) Laço de espera

(5) Apagar led

(6) Laço de espera

(7)Fim do laço



Endereços dos Registradores

Conteúdos dos registradores:

- 32 *bits*, códigos binários → unsigned int
- Campos de *bits* modificáveis pelos circuitos do microcontrolador → volatile

- **SIM_SCGC5**
  - (*(unsigned int volatile *) 0x40048038u)
  - Setar *bit* 13 para habilitar o *clock* da porta E
- **PORTE_PCR23**
  - (*(unsigned int volatile *) 0x4004D05Cu)
  - Setar *bits* 8, 9 e 10 para definir o tipo de sinal no pino 23
- **GPIOE**
  - **GPIOE_PDDR** (Sentido): (*(unsigned int volatile *) 0x400FF114u)
  - **GPIOE_PSOR** (1): (*(unsigned int volatile *) 0x400FF104u)
  - **GPIOE_PCOR** (0): (*(unsigned int volatile *) 0x400FF108u)

CodeWarrior IDE Development Suite

Informações Adicionais

- KL25 Sub-Family Reference Manual

  ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/KL25P80M48SF0RM.pdf

  - Mapa de memória: Seção 4.2 (página 105)
  - SIM: Capítulo 12 (página 192)
  - PORT: Capítulo 11 (página 175)
  - GPIO: Capítulo 41 (página 771)