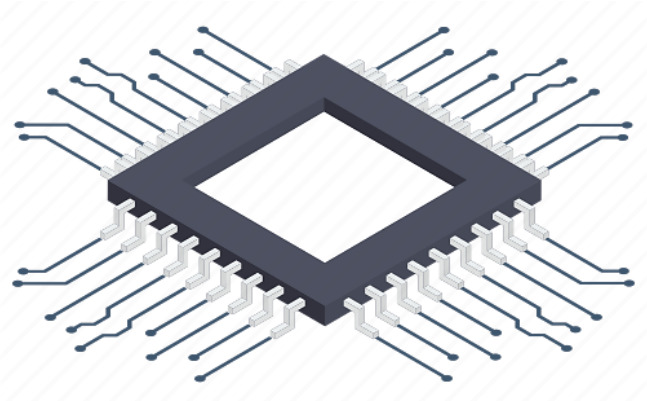
E209 – Microcontroladores

Linguagem C para microcontroladores



Prof. Yvo Marcelo Chiaradia Masselli

- O emprego da linguagem C na programação de microcontroladores exige o conhecimento mínimo das estruturas de programação, além de conhecimentos específicos sobre as estruturas internas do microcontrolador em questão.

- Este documento apresentará as estruturas de programação mais comuns, necessárias para se programar microcontroladores.

- Para os conhecimentos específicos de cada microcontrolador, uma leitura do *datasheet* do componente se faz necessária.

A estrutura básica de um programa em C para microcontroladores é a seguinte:

```
void main()
{
    // inicializações de hardware e software

    while(1)            // loop infinito
    {
        /* tarefas a serem executadas pelo microcontrolador
           durante o funcionamento */
    }
}
```

O uso inteligente dos comentários e funções facilita a compreensão do código e em muitos casos, reduz a utilização da memória do microcontrolador deixando o processamento mais rápido.

Definições Gerais em Linguagem C

- A Linguagem C é *Case Sensitive*

- Palavras Reservadas

auto	break	case	char	const
continue	default	do	double	else
enum	extern	float	for	goto
if	int	long	register	return
short	signed	sizeof	static	struct
switch	typedef	union	unsigned	void
volatile	while	main		

- Comentários

Os comentários de linha simples sempre são iniciados com "//" (dupla barra)

```
// Este é um comentário de linha simples.
/* Este é um comentário de
múltiplas linhas.
*/
```

Representação Numérica

- Decimal:

```
valor_decimal = 125;
```

- Binário:

```
PORTA = 0b11110001;
```

- Hexadecimal:

```
valor_hexa = 0xFF;
```

- Octal:

```
valor_octal = 075;
```

- Caractere:

```
caracter = 'a';
```

Tipos de Dados

Tipo	Bytes	Mínimo	Máximo	Representa
void	Zero	Sem valor	Sem valor	Valores nulos
char	1	-128	127	Caracteres
int	2	-32768	32767	Inteiros
float	4	$\pm 1,5 \times 10^{-45}$	$\pm 3,4 \times 10^{38}$	Valores com casas decimais (ponto flutuante)
double	8	$\pm 4,9 \times 10^{-324}$	$\pm 1,8 \times 10^{308}$	Valores com casas decimais (ponto flutuante)

Modificadores de Tipos de Dados

Tipo	Modificação
signed	Com sinal
unsigned	Sem sinal
short	Inteiro menor que o padrão
long	Inteiro maior que o padrão

Tipos de Dados Inteiros

Tipo	Bytes	Mínimo	Máximo
char	1	-128	127
signed char	1	-128	127
unsigned char	1	0	255
short int	2	-128	127
signed short int	2	-128	127
unsigned short int	2	0	255
int	2	-32768	32767
signed int	2	-32768	32767
unsigned int	2	0	65535
short	2	-32768	32767
signed short	2	-32768	32767
unsigned short	2	0	65535
long int	4	-2^{31}	$2^{31}-1$
signed long int	4	-2^{31}	$2^{31}-1$
unsigned long int	4	0	$2^{32}-1$

Exercícios:

Escreva a linha de código em C para as situações a seguir:

A. Declaração de uma variável que varie de 0 a 255 de nome “contador”

B. Declaração de uma variável que varie de -128 a 127 de nome “entrada”

C. Declaração de uma variável que varie de 0 a 65535 de nome “tempo”

D. Declaração de uma variável que varie de -32768 a 32767 de nome “temperatura”

E. Declaração da inclusão de uma biblioteca de definição de registros do microcontrolador de nome “avr.h”

F. Definição de uma constante de nome “IN” com o valor 16d (em decimal)

Constantes

```
const LIGADO = 1;
const DESLIGADO = 0;
const TMAX = 100;
```

Após definir as constantes, é possível utilizar estas palavras como tendo o mesmo significado do valor numérico atribuído.

Definições

Além de dar nome a valores constantes, também podemos atribuir outros nomes a símbolos já existentes no microcontrolador.
Por exemplo, imagine que você tenha um LED ligado ao bit 0 da porta A do seu microcontrolador, além de um motor DC ligado ao bit 1 dessa mesma porta. Para facilitar a programação, pode-se fazer:

```
#define LED   PORTA.B0
#define MOTOR PORTA.B1
```

Na Linguagem C os comandos iniciados pelo caractere "#" são os únicos que não recebem ponto-e-vírgula no final da sentença.

Os comandos de definição “#define” e “const” servem para tornar o programa mais compreensível.

Além disso, estes comandos são sempre eliminados pelo compilador, ou seja, não consomem memória extra do microcontrolador.

```
#define LED   PORTA.B0
#define MOTOR PORTA.B1

const LIGADO = 1;
const DESLIGADO = 0;

void main()
{
     LED = LIGADO;
     MOTOR = LIGADO;
}
```

Variáveis

Uma variável sempre deve ser declarada da seguinte forma:
(<modificador>) <tipo de dado> <nome da variável> (= <valor>);
Note que a utilização do modificador e da inicialização da variável são opcionais.

```
unsigned int valor = 123;
char letra = 'a';
int contador;
```

Variáveis Locais

São variáveis declaradas dentro de um bloco de instruções (ou função) e só podem ser acessadas dentro desse bloco. Exemplo:

```
int media()
{
    int valor1 = 5;
    int valor2 = 7;

    return (valor1+valor2)/2;
}
```

Variáveis Globais

São declaradas fora dos blocos de função e podem ser acessadas por todas as funções e blocos de instruções. Exemplo:

```
int valor1 = 5;
int valor2 = 7;

int media()
{
    return (valor1+valor2)/2;
}

int soma()
{
    return valor1+valor2;
}
```

Declarar variáveis globais não é considerada uma boa prática de programação, devendo ser utilizadas apenas quando estritamente necessárias, e de forma bastante cuidadosa.

Funções

Quando temos trechos de códigos que são repetidos várias vezes, podemos isolar estes trechos em funções.

- Uma função deve ser declarada antes de ser chamada.
Por isso é importante escrever todas as novas funções **antes** da função principal (main).

Exemplo:

```
void piscaLED()                 // declara o início da função piscaLED
{
    PORTA.B0 = 1;               // liga o pino A0
    delay_ms(1000);             // aguarda 1000ms (1 segundo)
    PORTA.B0 = 0;               // desliga o pino A0
    delay_ms(1000);             // aguarda 1 segundo
}

void main(void)                 // início do programa principal
{
    while(1)
    {
        piscaLED();             // chama a função piscaLED
    }
}
```

O **void**, que traduzido para português quer dizer vazio, é um tipo de dado utilizado em funções que não retornam nenhum parâmetro.

A função **main** é sempre do tipo void, pois não retorna nem recebe nenhum valor.

Entretanto, as funções podem muitas vezes conter parâmetros de entrada que carregam informações para dentro do bloco da função, além de parâmetros de saída, devolvendo informações ao programa principal.

Exemplo:

```
int media(int a, int b)     /* declara o início da função media.
                               Essa função recebe dois valores
                               do tipo int (a e b) */
{
    return (a + b)/2;       // retorna um valor do tipo int
}

void main(void)             // início do programa principal
{
    int valor1 = 10;
    int valor2 = 20;
    int result;

    result = media(valor1, valor2);    /* A função média retorna
                                          seu valor do tipo int
                                          na variável result */
}
```

Header

- Em muitos compiladores, é necessário incluir um arquivo que contém todas as definições do microcontrolador que está sendo programado.
Isto é feito através do comando “#include”.
Esta ser incluída logo no início de qualquer programa:

#include<microcontrolador.h>

Pode-se utilizar a instrução “#include” para incluir algum arquivo denominado cabeçalho, contendo funções auxiliares a serem utilizadas. Por exemplo:

#include<nome_do_arquivo.h>

Operadores

A linguagem C possui operadores para atribuir, comparar ou modificar dados e variáveis.
Podemos dividir estes operadores em quatro categorias principais:

- **Aritméticos**: para cálculos;
- **Relacionais**: para comparar valores numéricos;
- **Lógicos**: para testar condições ou manipular bits;
- **Booleanos**: para realizar operações lógicas com mais de um bit.

Operadores Aritméticos

Símbolo	Operação
+	Soma
-	Subtração
*	Multiplicação
/	Divisão
%	Resto inteiro da divisão
++	Incremento em um
–	Decremento em um

Operadores Relacionais

Realizam operações de comparação, retornando os valores FALSO (0) ou VERDADEIRO (1).
Esses operadores não afetam o conteúdo das variáveis sendo comparadas.

Símbolo	Descrição
==	Igual a
!=	Diferente de
<	Menor que
>	Maior que
<=	Menor ou igual
>=	Maior ou igual

Exemplo:

```
void main()
{
    unsigned char cont = 0;
    while(1)
    {
        if (PORTA.B0 == 1)      // Se o pino A0 estiver ligado
            if (cont < 9)
                cont++;         /* Se contagem for maior que 9,
                                   incrementa */
            else
                cont = 0;       // Caso contrário, zera
    }
}
```

Operadores Booleanos

São usados para realizar testes booleanos entre elementos em um teste condicional.

Símbolo	Descrição
&&	E (AND)
||	OU (OR)
!	NÃO (NOT)

Exemplo:

```
int A = 0;
int B = 0;
int C = 0;
if ((A != B) || (A != C))
    A--;
```

Operadores Lógicos

São usados para realizar testes lógicos bit a bit. Os operadores lógicos são:

Símbolo	Descrição
&	E (AND)
|	OU (OR)
^	OU Exclusivo (XOR)
~	NÃO (NOT)
»	Deslocamento para a direita
«	Deslocamento para a esquerda

Exemplo 1:

```
int cont = 0;
if (PORTA.B0 & PORTA.B1)
    cont++;
```

Exemplo 2:

```
int cont = 0b00000001;
if (PORTA.B0 | PORTA.B1)
    cont = cont<<1;
```

Exemplo 3:

```
int cont = 10;
if (PORTA.B0 & ~PORTA.B1)
    cont--;
```

Associação de Operadores

Forma Expandida	Forma Reduzida
x = x + y	x += y
x = x - y	x -= y
x = x * y	x *= y
x = x / y	x /= y
x = x % y	x %= y
x = x & y	x &= y
x = x | y	x |= y
x = x ∧ y	x ∧= y
x = x << y	x <<= y
x = x >> y	x >>= y

Exercícios:

Dado que A=55h e B=F0h. Considerando que todas as variáveis são do tipo unsigned char, qual o valor resultante das expressões a seguir:

a) X = A & B;

b) X = A | B;

c) X = A && B;

d) X = A || B;

e) X = B >> 4;

f) X = A << 1;

Estruturas Condicionais

Estrutura *if*

```
if (condição)
{
    comando 1;
    comando 2;
    ...
    comando n;
}
```

Exemplo

```
void main()
{
    int contagem = 0;
    if (PORTA.B0)
        contador++;        /* Apenas se o valor da porta A0 for igual a
                              um, o contador será incrementado */
}
```

Estrutura *if-else*

```
int i = 10;
int j = 5;

if (i == 10)     // Se 'i' for igual a 10, incrementa j
    j++;
else             // Senão, decrementa j
    j--;
```

Estrutura *if-else if*

```
if (condição 1)
{
   comandos;
}
else if (condição 2)
{
    comandos;
}
else if (condição 3)
{
    comandos;
}
```

Estrutura *switch*

```
switch(variável)
{
    case constante1:
        comandos;
        break;
    case constante2:
        comandos;
        break;
    ...
    case constante n:
        comandos;
        break;
    default:
        comandos;
        break;
```

Exemplo

```
int contador = read_ADC();     /* lê o valor do conversor AD e armazena
                                  na variável 'contador' */

switch(contador)
{
    case 0:
        x++;
        PORTA.B0 = 1;
        break;
    case 1:
        PORTA.B1 = 1;
        break;
    case 2:
        PORTA.B0 = 0;
        PORTA.B1 = 0;
        break;
    default:
        break;
```

Estruturas de Repetição

Estrutura *for*

```
for(inicialização; condição; incremento)
{
    comandos;
}
```

Exemplo:

```
for(i = 0; i < 10; i++)
{
    PORTA.B0 = 1;          // liga o pino A0
    delay_ms(1000);        // aguarda 1 segundo
    PORTA.B0 = 0;          // desliga o pino A0
    delay_ms(1000);        // aguarda 1 segundo
}
```

O Laço *while*

O laço *while* tem a seguinte estrutura:

```
while(condição)
{
    comandos;
}
```

Exemplo:

```
while(x<50)    // Enquanto x for menor do que 50
{
    x++;       // incrementa x
}
```

```
void main()
{
    // inicializações de hardware e software

    while(1)              // loop infinito
    {
        /* tarefas a serem executadas pelo microcontrolador
           durante o funcionamento */
    }
}
```

Laço *do-while*

```
do
{
    comandos;
} while(condição);
```

Exercícios:

1 - Analise o programa a seguir e responda:

```
unsigned char cont=0;

void main (void)
{
   for(;;)
   {
      P1 = ~cont;
      cont++;
      delayms(1000);
   }
}
```

a) Indique quais partes são:
- Programa principal
- Declaração de variável global
- Chamada de função
- Loop Infinito

b) Qual o limite de armazenamento da variável cont?

c) Qual o valor carregado em cont a cada vez que o loop é executado?

d) Qual a relação que existe entre os valores de cont e P1?

2 - Analise as estruturas condicionais abaixo e responda:

Estrutura 1

```
if (a>10)
{
     j++;
}
else
{
     j=0;
}
```

Estrutura 2

```
if(a>0 && b>0)
{
     j++;
}
else
{
     j=0;
}
```

Obs: considere que todas as variáveis são do tipo unsigned int e que o valor inicial do "j" é 255.

A. Para quais valores de "a" a variável j será igualada a zero?

B. Para quais valores de "a" a variável j será incrementada?

C. Seria viável substituir essas estruturas "if" por "switch"?

Exercícios: Analise as estruturas de repetição abaixo e responda:

Estrutura 1

```
for (i=0;i<10;i++)
{
   j++;
}
```

Estrutura 2

```
for (i=5;i>0;i--)
{
   j++;
}
```

Estrutura 3

```
for (;;)
{
   j++;
}
```

a) Qual o valor final de "j" quando as estruturas 1,2 e 3 chegarem ao fim?

b) Qual das estruturas poderia ser substituída por uma estrutura do tipo:

```
while(1)
{
    j++;
}
```

Obs: considere que todas as variáveis são do tipo unsigned int e que o valor inicial de j é 0.