SISTEMAS MICROPROCESSADOS I

Prof.: João Castelo

CENTRO UNIVERSITÁRIO
SENAI CIMATEC
SENAI CIMATEC SISTEMA FIEB
Federação das Indústrias do Estado da Bahia

Linguagem de Programação para Microcontroladores

- Num computador de uso geral existem dispositivos padronizados de entrada e saída (teclado, monitor de vídeo, impressora, mouse, etc.) facilitando a criação de rotinas também padronizadas para tratar estes dispositivos;
- Existe um Sistema Operacional para “gerenciar” os recursos do computados. O Sistema Operacional aloca e controla todos os recursos disponíveis e escalona a execução dos programas;
- Um sistema microcontrolado é, geralmente, um sistema com função específica;

Linguagem de Programação para Microcontroladores

- Normalmente não existem dispositivos e rotinas padronizadas para os periféricos. Na verdade, cada projeto deverá definir quais periféricos farão parte e como eles serão tratados;
- O projetista deverá definir e construir as rotinas necessárias de acordo com as necessidades do projeto;
- Num sistema microcontrolado não existe um Sistema Operacional para “gerenciar” os recursos;
- O controle e alocação dos recursos deve ser feito pelo programa (firmware) desenvolvido para o projeto em questão;
- O programa deve ficar em execução o tempo todo;

Linguagem de Programação

Num computador genérico

- Existe um dispositivo padrão de saída (monitor de vídeo);
- O sistema operacional carrega o programa na memória;
- Após a execução do programa, o controle retorna para o sistema operacional;

Num sistema microcontrolado

- Dispositivo de saída depende do projeto. Pode ser um display LCD, pode ser um conjunto de LEDS, pode ser que não precise de nada disso;
- O programa é executado assim que o sistema é ligado;
- O programa nunca termina;

Linguagem de Programação C

- C é uma linguagem imperativa e procedural, para implementação de sistemas.
- Seus pontos de design foram para ele ser compilado, fornecendo acesso de baixo nível à memória e baixos requerimentos do hardware.
- Foi desenvolvida para ser uma linguagem de alto nível, para maior reaproveitamento do código.
- C foi útil para muitas aplicações que foram codificadas originalmente em Assembly.

Linguagem de Programação C

- A focalização no paradigma de programação procedimental;
- Um sistema de tipos simples que evita várias operações que não fazem sentido
- Uso de uma linguagem de pré-processamento, o pré-processador de C, para tarefas tais como a definição de macros e a inclusão de múltiplos ficheiros de código fonte;
- Ponteiros dão maior flexibilidade à linguagem;

Palavras reservadas

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

Tipos de dados

Tipo de dados em C	Significado	Código printf	Bits
void	Vazio		0
Char	Caracter	%c	8
String	Cadeia de caracteres	%s	
Int	Números Inteiros	%i ou %d	16 ou 32
Float	Números Reais	%f	32
Double	Números Reais maior nº de casa decimais	%f	64

Modificadores de tipos

- signed:
  - utilizará o último bit para sinalizar se um dado é positivo ou negativo.
  - Com isto, o dado sempre estará com um tamanho menor em um bit.
- unsigned:
  - não informa se o dado tem valor positivo ou negativo.
  - Por consequência todos os bits do dado podem ser utilizados como informação;
- short:
  - faz com que o dado passe a ter um tamanho menor do que especificado em sua definição.
  - Por exemplo: utilizar o short int pode fazê-lo assumir o valor de apenas um bit, dependendo do compilador utilizado;
- long:
  - faz com que o dado passe a ter um tamanho maior do que especificado em sua definição.
  - Por exemplo: utilizar o long int pode fazê-lo assumir o valor de 65536 bits, dependendo do compilador utilizado

Modificadores de tipos

Keyword	*Variable Type*	*Range*
char	Character (or string)	–128 to 127
int	Integer	–32,768 to 32,767
short	Short integer	–32,768 to 32,767
short int	Short integer	–32,768 to 32,767
long	Long integer	–2,147,483,648 to 2,147,483,647
unsigned char	Unsigned character	0 to 255
unsigned int	Unsigned integer	0 to 65,535
unsigned short	Unsigned short integer	0 to 65,535
unsigned long	Unsigned long integer	0 to 4,294,967,295
float	Single-precision floating-point (accurate to 7 digits)	$\pm 3.4 \infty 10^{38}$ to $\pm 3.4 \infty 10^{-38}$
double	Double-precision floating-point (accurate to 15 digits)	$\pm 1.7 \infty 10^{-308}$ to $\pm 1.7 \infty 10^{308}$

Declaração de variáveis

```
TIPO nome_da_variável {,outras_variáveis};
unsigned int tempo;
```

Inicialização de variáveis

```
unsigned int tempo = 100;
```

Variáveis globais e locais

- Variáveis globais: são acessíveis de qualquer ponto do programa, por qualquer função, e devem ser declaradas no corpo principal do programa, fora de qualquer função, inclusive da main;
- Variáveis locais: só podem ser acessadas dentro das funções onde foram criadas. Ao sair desta função, a linguagem C destrói uma variável local e, portanto ela não será acessível por outras funções.

Exemplo

```
#include <msp430xG46x.h>
#include <stdio.h>

int somatorio; // VARIÁVEL GLOBAL! SERÁ ACESSADA POR TODAS AS FUNÇÕES

void soma (int valor) //AO ACESSAR ESTA FUNÇÃO, O PARÂMETRO VALOR RECEBE DADOS DE QUEM O CHAMOU
{
 int conta; // VARIÁVEL LOCAL! SERÁ ACESSADA APENAS PELA FUNÇÃO SOMA
 somatorio = somatorio + valor;
 printf("0");
 for (conta = 1;(conta<(valor+1));conta++)
 {
 printf("+%u",conta);
 }
 printf(" = %u\r\n",somatorio);
}

void main()
{
 WDTCTL = WDTPW+WDTHOLD; // Stop WDT
 int conta; // VARIÁVEL LOCAL! SERÁ ACESSADA APENAS PELA FUNÇÃO MAIN
 somatorio = 0; // A VARIÁVEL GLOBAL É INICIALIZADA
 for (conta=1;conta<20;conta++)
 {
 soma(conta); // É CHAMADA A FUNÇÃO SOMA, ONDE É PASSADO O VALOR DE CONTA
 }
}
```

Interrupção

- A rotina de interrupção pode ser escrita da seguinte forma:

```
interrupt void intproc(void) {
      ...
}
```

- Ou assim:

```
#pragma interrupt
void intproc(void) {
      ...
}
```

Exercícios

1. Faça um programa, em linguagem C, para piscar um LED.
2. Assumindo que o PORT_1 do MSP430 está configurado como saída digital e sabendo-se que o valor presente em cada pino e dado pelo conteúdo do registrador P1OUT, escreva um programa eficiente em linguagem C que faça com que os pinos do PORT_1 sejam ativados (em nível 1) sequencialmente.
   - Observação: no MSP430 não é possível escrever valores individuais no bits dos registradores. Para se fazer isto é necessário usar máscaras. Por exemplo: para escrever um valor “1” no pino P1.3, sem alterar o valor dos demais pinos usa-se a seguinte linha de código: P1OUT |= 0x08;ou P1OUT |= BIT3; onde BIT3 é uma constante definida em “io430.h” e em “msp430g2553.h