

EEL7030 Microprocessadores – Laboratório 8

Prof. Raimes Moraes

Linguagem C para o 8051

O programa a seguir mostra os caracteres de 0 a F no *display3* do EDSIM51 (<http://www.edsim51.com>). Para permitir a escrita no *display3* do EDSIM51, deve-se fazer para CS='1', END0=END1='1'. A função **converte_7seg** recebe como parâmetro, o valor contido na variável **j** e retorna o valor a ser escrito no *display* de 7 segmentos (através da porta P1) para mostrar o dígito. Compile o programa no Keil, gere .HEX (pressione Alt+F7; selecione o tab Output e selecione Create HEX file) e o carregue no EDSIM51 para observar a sua execução.

//Programa que escreve números hexadecimais (0 a F) no display 3 do EDSIM51

#include <reg51.h> // declaração de *specialfunctionregisters* para 8051

unsigned char converte_7seg (unsigned char);

void main (void)

{

short i;

unsigned char j=0;

P0=0x80;

P3=0xFF;

while (1) {

if (j == 16) j=0;

for (i = 0; i <15000; i++); // atraso para Edsim 2.1.15 – Update freq=50000

P1=converte_7seg(j);

j++;

} // end of while

} // end of main

```

unsigned char converte_7seg (unsigned char dado) // função que retorna valor a ser escrito...
{
    //... nos displays para formar o caractere

    unsigned char led;

    switch (dado)
        // GFEDCBA
        {
            case 0: led = 0x40; break; // "1000000";
            case 1: led = 0x79; break; // "1111001";
            case 2: led = 0x24; break; // "0100100";
            case 3: led = 0x30; break; // "0110000";
            case 4: led = 0x19; break; // "0011001";
            case 5: led = 0x12; break; // "0010010";
            case 6: led = 0x02; break; // "0000010";
            case 7: led = 0x78; break; // "1111000";
            case 8: led = 0x00; break; // "0000000";
            case 9: led = 0x10; break; // "0010000";
            case 10: led = 0x08; break; // "0001000";
            case 11: led = 0x03; break; // "0000011";
            case 12: led = 0x46; break; // "1000110";
            case 13: led = 0x21; break; // "0100001";
            case 14: led = 0x06; break; // "0000110";
            case 15: led = 0x0E; break; // "0001110";
            default: led = 0x80; // "0000000";
        }

    return led;
} // end converte

```

Exercícios:

- 1) Modifique o programa anterior para mostrar uma contagem decrescente (9 a 0) no *display* 3.
- 2) Modifique o programa anterior para mostrar valor lido das chaves (0 a FH) conectadas à porta P2 (*Switch Bank*) no *display* 0. Valor lido de chave aberta equivale a '1'; de chave fechada, '0'. Portanto, complemente o valor lido de P2 antes de apresentá-lo à função de

conversão. Mascarar valor lido referente às 4 chaves mais significativas para evitar influência das mesmas.

- 3) Modifique o programa anterior para mostrar no *display* 1, o nro de chaves fechadas.
- 4) Teste a escrita de caracteres no LCD do EDSIM51 com o código disponibilizado nos últimos 3 slides da Aula 5.
- 5) Informar no LCD, o dado apresentado às chaves conectadas à porta P2; por exemplo, quando as chaves 0 e 1 estiverem pressionadas, mostrar a mensagem “Valor: 3”. OBS: subdivida a função WRITE (contida na aula sobre cristal líquido – Aula 5) para criar uma segunda função que envie apenas um caractere de dados ou comando para o LCD. Utilizar comando para posicionar o cursor (*Set DDRAM address*) sobre a coluna com valor a ser sobrescrito. O valor lido das chaves deve ser convertido no ASCII correspondente para ser enviado para o LCD.
- 6) No EDSIM51, há leds conectados à Porta P1. Faça um programa que rotacione um led aceso da esquerda para a direita inserindo atraso entre rotações. Fazer que o procedimento seja cíclico (repita-se). Utilize o TIMER0 no modo 2 para inserir atraso (em vez do uso de for). OBS: Inibir CS do *display*.
- 7) Faça um programa que acenda todos os 8 leds, um a um (da esquerda para a direita) com inserção de atraso; quando todos os leds estiverem acesos, apague-os um a um (da direita para a esquerda). Fazer que o procedimento seja cíclico.
- 8) Faça um programa que teste a chave conectada a P2.7. Se a mesma estiver fechada (P2.7='0'), rotacione um led aceso para a esquerda (se P2.6='1') ou para a direita (se P2.6='0') pelo número de vezes especificado pelo complemento do valor das 4 chaves

menos significativas (P2.3 a P2.0). Se a chave P2.7 estiver aberta, fique aguardando ser alterada.

- 9) Teste o exemplo para transmissão e recepção de dados pela serial à taxa de 2400 bps (OBS: colocar cristal do simulador em 11,059 MHz – *Options for Target*). O mesmo interrompe comunicação ao receber a letra x.

```
#include <reg51.h>          /* endereços dos special function registers */
#include <stdlib.h>

void serial(void);

unsigned char xdataserial_input_buffer [] = "Teste para verificar se x interrompe serial";
unsigned char xdatamempool [100]; // aloca 100 bytes no pool de memória
unsigned char xdata *regPtr;
unsigned char xdata *Reg1;

void main (void) {

    init_mempool (&mempool, sizeof(mempool)); //inicializar rotinas de memory management

    regPtr = calloc (32, sizeof (unsigned char)); // aloca 32 dados unsigned char na memória
    Reg1=regPtr;

    SCON = 0x50;           // SCON: modo 1, 8-bit, recepcao habilitada
    TMOD |= 0x20;          // TMOD: timer 1, modo 2
    TH1 = 0xf4;            // TH1: valor de recarga para 2400 baud; clk = 11,059 MHz
    TR1 = 1;               // TR1: dispara timer

    ES = 1;
    EA = 1;                // habilita interrupcao serial
    SBUF = 0x41;           // envia 'A'

    while (SBUF != 'x');   //enquanto não detecta um 'x' na porta serial, executa...
    ES = 0;                // inibe interrupcao pela porta serial
    free (regPtr);
    while (1);             // necessário para evitar travamento

}

void serial(void) interrupt 4 { // especifica interrupção serial (4)
    static unsigned char tm=0x42; // inicializa tm com o ASCII = 'B'
    static unsigned char count = 0;

    if (RI) { // testa se dado recebido - buffer de recepção cheio

        RI=0;
        *regPtr++ = SBUF;
        count++;
        if(count==32) {regPtr = Reg1; count=0; };
    }
}
```

```

if (TI) { // testa se buffer de transmissão vazio

    TI=0;
    SBUF = tm;
    tm++;
    if (tm==0x62) tm=0x41;

}

} // endof serial

```

Para melhor avaliar o funcionamento do código, compile o script que se segue a este enunciado na ferramenta de depuração do Keil (ver: http://www.keil.com/support/man/docs/uv4/uv4_df_createfunct.htm)

```

define int seri_index;

signal void seri_start (float baudrate) {
    seri_index = 0;

    for (seri_index = 0; serial_input_buffer[seri_index] != '\0'; seri_index++) {
        SIN = serial_input_buffer[seri_index];
        // Wait for a little longer than 1 character time between each character
        swatch (1.0 / ((baudrate/10.0) + 20.0));
    }
}

define button "Send Data" , "seri_start(2400)"

```

Tal procedimento gera um botão “Send Data” que deve ser pressionado para enviar a cadeia de caracteres: “”Teste para verificar se x interrompe serial”;

- 10) Faça um programa que identifique a tecla pressionada no *keypad* conectado à porta P0 e a apresente no *display* 0. A rotina para identificar a tecla pressionada em Assembly é apresentada a seguir; este código coloca uma das linhas do teclado (P0.3 a P0.0) em nível lógico baixo, uma após a outra. Quando uma das linhas é colocada em nível lógico baixo, as colunas são testadas (P0.6 a

P0.4) para verificar se em alguma delas, '0' é lido, haja vista que a tecla pressionada fecha contato da linha com a coluna. Se sim, a tecla pressionada corresponde àquela combinação de linha e coluna. Rescrever a rotina KEYPAD como uma função em C para executar o programa.

; Subrotina que retorna em R0 o valor do dígito pressionado no teclado do EDSIM51
 ; (Obs: retorna A para * e C para #)

```
; teclado
;
;                               linhas
;       +---+---+---+
;       | 1 | 2 | 3 |   P0.3
;       +---+---+---+
;       | 4 | 5 | 6 |   P0.2
;       +---+---+---+
;       | 7 | 8 | 9 |   P0.1
;       +---+---+---+
;       | A | 0 | C |   P0.0
;       +---+---+---+
;   colunas  P0.6 P0.5 P0.4
```

KEYPAD:

```
ORL   P0,#7Fh      ; escreve '1' em 7 pinos da porta P0
CLR   F0            ; limpa flag que identifica tecla pressionada
MOV   R0, #0        ; limpa R0 – retorna o número da tecla foi pressionada

; varre primeira linha
CLR   P0.3          ; coloca '0' na linha P0.3
CALL  colScan        ; chama rotina para varredura de coluna
JB    F0, finish     ; se flag F0 = '1', tecla identificada => retorna

; varre segunda linha
SETB  P0.3          ; seta linha P0.3
CLR   P0.2          ; coloca '0' na linha P0.2
CALL  colScan        ; chama rotina para varredura de coluna
JB    F0, finish     ; se flag F0 = '1', tecla identificada => retorna

; varre terceira linha
SETB  P0.2          ; seta linha P0.2
CLR   P0.1          ; coloca '0' na linha P0.1
CALL  colScan        ; chama rotina para varredura de coluna
JB    F0, finish     ; se flag F0 = '1', tecla identificada => retorna

; varre quarta linha
SETB  P0.1          ; seta linha P0.1
CLR   P0.0          ; coloca '0' na linha P0.0
CALL  colScan        ; chama rotina para varredura de coluna
JB    F0, finish     ; se flag F0 = '1', tecla identificada => retorna

JMP   KEYPAD        ; se flag F0 = '0', tecla não identificada => repete varredura

finish:
RET
```

; Subrotina que varre as colunas para identificar a qual pertence a tecla pressionada
; o registrador R0 é incrementado a cada insucesso de forma a conter o nro. da tecla
; pressionada

colScan:

JNB P0.6, gotKey ; tecla pressionada pertence a esta coluna – retornar

INC R0

JNB P0.5, gotKey ; tecla pressionada pertence a esta coluna – retornar

INC R0

JNB P0.4, gotKey ; tecla pressionada pertence a esta coluna – retornar

INC R0

RET ; tecla pressionada não pertence a esta linha – retornar

gotKey:

SETB F0 ; faz flag F0 = '1' => tecla identificada

RET

; Subrotina converte com tabela modificada para o exercício solicitado

CONVERTE: INC A

MOVC A,@A+PC

RET

TABELA: DB 79H, 24H, 30H, 19H, 12H, 02H, 78H, 00H, 10H,08H,40H, 46H