UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

Leonardo Winter Pereira
Rodrigo Yudi Endo

Projeto 01 - Fechadura Eletrônica

CURITIBA

2017

Introdução

Relatório para o trabalho prático apresentado para a disciplina de microcontroladores, turma S12, da Universidade Tecnológica Federal do Paraná.

Objetivos

Projetar um sistema acionador de trava eletromecânica, utilizando um microcontrolador da família 8051, contendo os seguintes requisitos:

- Possuir um teclado matricial contendo dígitos de 0-9, ENTER e CLR. Além disso, o projeto deve envolver um display LCD, um motor-de-passo e um buzzer sonoro.
- Uma mensagem de texto personalizada deve aparecer ao início do programa.
- Ao digitar a senha, os dígitos devem ser substituídos por um “*” e um beep sonoro deve ser tocado a cada tecla pressionada. Após a senha ser digitada, a tecla ENTER deve ser pressionada.
- Caso o usuário desista da operação ou queira corrigir a senha digitada, ele deve pressionar a tecla CLR.
- A validação da senha deve ser feita no display LCD, que deve mostrar uma mensagem de senha correta ou incorreta. No caso de a senha estar correta, o buzzer deve soar por 1 segundo e o motor-de-passo mover uma fechadura para a posição de destrancamento.
- Após 3 segundos do destravamento da fechadura, o motor-de-passo deve retornar a sua posição original.
- No caso de senha incorreta, o buzzer deve tocar por 2 segundos e o sistema ficar inativo por 3 segundos.
- O teclado não deve possuir bounce e o LCD não deve mostrar caracteres estranhos.
- O buzzer pode ser ativado por um oscilador externo

Esquemático do projeto

O esquemático utilizado para a montagem do projeto pode ser visto na figura 1 abaixo:

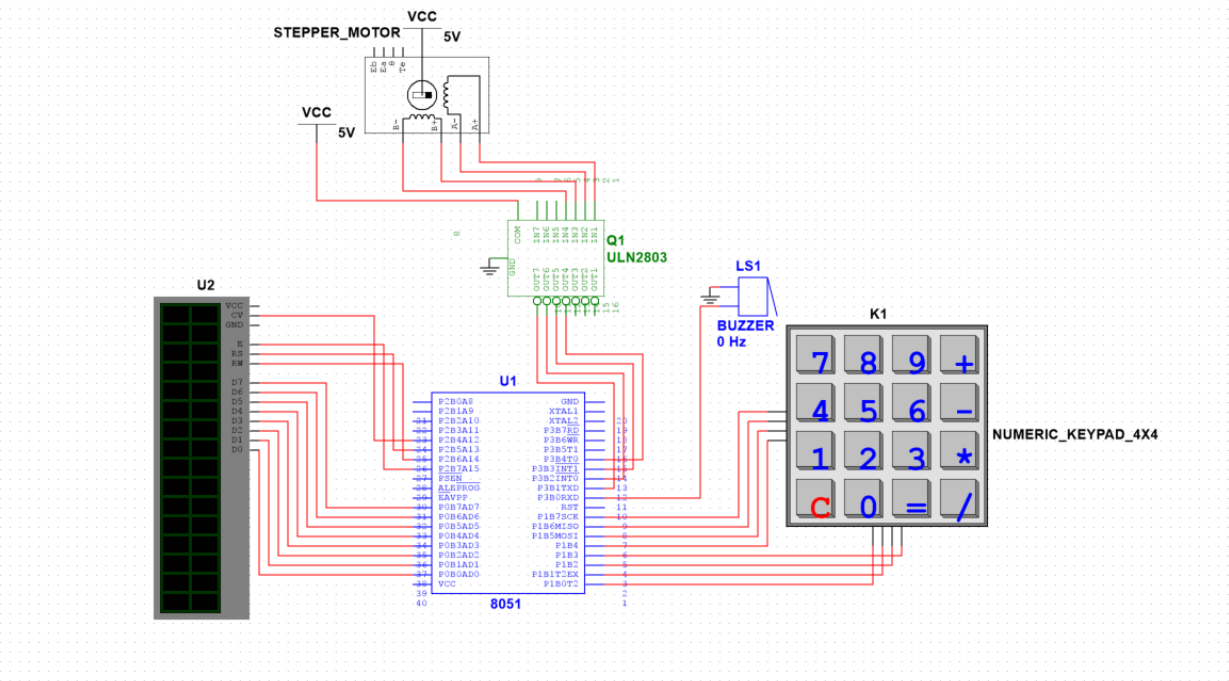

Figura *1 – Esquemático*

Métodos e funções implementadas

Timers e interrupções

Inicialmente, as interrupções foram setadas, porem somente as interrupções internas TIMER/COUNTER 0 e TIMER/COUNTER 1 foram utilizadas para lidar com os delays e as interrupções, respectivamente.

O TIMER0 foi configurado em modo 1(16 bits), com os bytes de TH0 e TL0 configurados de forma a causar um delay de 20ms. A função TIMER_DELAY_20_MS possui um parâmetro de entrada, R0, que deve ser configurado antes da função ser chamada, gerando um delay de R0*20ms, sendo que R0 varia entre 0x1 e0xFF.

Outra função, TIMER_DELAY_1_S , foi criada para gerar um delay de 1s, para facilitar os delays utilizados durante o programa. Ela funciona utilizando um registrador de entrada R1 e causando um delay de R1*1s.

A função CONFIGURA_VALORES_TIMER_1, configura TIMER1, responsável pela interrupção de TIMEOUT, que reinicia o programa quando o usuário leva muito tempo para executar ações. Ela utiliza 3 registradores:

- TIMEOUT_LOW e TIMEOUT_HIGH: Registradores pré-configurados para que o tempo de TIMEOUT seja de TIMEOUT_X1S*1s.

- TIMEOUT_X1S: Representa a quantidade em segundos para o TIMEOUT do sistema.

Display LCD

O display é primeiramente limpo e as strings pré-definidas para a mensagem de boas-vindas são escritas nele. As funções CLR1L e CLR2L limpam, respectivamente, a primeira e segunda linha do display e posicionam o cursor no início. Elas funcionam retornando o cursor para a primeira posição da linha e escrevendo um espaço em branco em cada uma das 16 células.

Já as funções ESC_STR1 e ESC_STR2 escrevem caracteres na primeira e na segunda linha, movendo o cursor uma posição para frente. Quando elas são chamadas, o cursor é posicionado no local desejado utilizando a função GOTOXY e um caractere escrito utilizando a função ESCDADO, que habilita o LCD e seleciona o registrador de dados, movendo o caractere desejado utilizando a instrução:

```
MOV          P0,A //escreve no bus de dados
```

Motor-de-passo

Para controlar o motor-de-passo, o driver de motor LN2803 foi utilizado. As saídas do CI são utilizadas para controlar o componente e foram ligadas aos ports P3.1 ao P3.4. Após o usuário inserir a senha correta, a função ABRE_FECHADURA é chamada. Esse método funciona setando o port P3.1 em um e os demais em zero, passando em seguida para o P3.2 e zerando o P3.1, até chegar ao P3.4. Entre cada uma dessa mudanças, um delay de 20ms*R0 é utilizado. Sendo assim, o tempo de cada passo pode ser ajustado alterando o valor do registrador R0. Além disso, a quantidade de passos é determinada pelo registrador R5, que é decrementado por uma instrução DJNZ na função de abertura da fechadura.

Após a abertura completa da fechadura, um delay de 3s é aguardado e a função TRANCA_FECHADURA é chamada. Esta funciona de maneira muito semelhante a ABRE_FECHADURA, porem de forma contraria.

Buzzer

Para o acionamento do buzzer, inicialmente o port P3.0 foi setado como o pino do buzzer. Em seguida, a função ACIONA_BUZZER foi criada, setando o bit do buzzer por 20ms e em seguida realizando um clear na port utilizada. A função é simples e pode ser vista abaixo:

```
ACIONA_BUZZER:
                SETB BUZZER
                ACALL       TIMER_DELAY_20_MS
                CLR   BUZZER
```

RET

Dessa forma, cada vez que o buzzer precisou ser acionado, essa função foi chamada. Ela recebe R0 como parâmetro de entrada e o buzzer fica ativo por R0*20ms.

Teclado matricial

Inicialmente, o teclado matricial foi ligado nos ports P1.0 a P1.7. As funções relativas a leitura dos dados vindo do teclado começam a ser chamadas após as mensagens iniciais do display. Para realizar o debounce do teclado, um delay de 200ms foi utilizado entre as leituras de teclas.

Para ler a tecla pressionada, o programa lê as entradas no port P1 e as relaciona com as linhas e as colunas do teclado matricial, utilizando a label LE_4_DIGITOS, um loop que chama a função VARREDURA_TECLADO. Por exemplo, se ele lê 1101 1011, os zeros indicam que a tecla na linha 3 com a coluna 2 foi pressionada, correspondendo ao digito 6. Dessa forma, o programa é direcionado a função da tecla correspondente, salvando-a em um registrador e pulando para o próximo endereço de registrador, que salvara o próximo digito da senha. Além disso, a função ESCREVE_ASTERISCO é chamada para escrever um asterisco na tela e manter a privacidade da senha.

Quando o usuário terminar de digitar os 4 dígitos da senha, ele deve pressionar ENTER para o programa continuar. Além disso, ao pressionar a tecla CLR, o programa retorna ao ponto em que ele pede para o usuário inserir a senha, apagando os dígitos inseridos até o momento.

Validação da senha

Duas senhas foram inicialmente pré-programadas na memória ROM. A validação da senha digitada ocorre depois do pressionamento da tecla ENTER, após a senha ter sido digitada.

O código compara o primeiro digito da senha digitada com o primeiro digito da primeira senha e assim em diante, utilizando a função TESTA_SENHA1. Caso um digito esteja errado, ele pula para a função TESTA_SENHA2 e, caso um digito da segunda senha esteja errado novamente, ele chama a função LIMPA_LCD_E_MOSTRA_SENHA_INVALIDA, mostrando “SENHA INCORRETA” no display e soando o buzzer por 2 segundos.

Caso uma das senhas esteja correta, o programa é direcionado para LIMPA_LCD_E_MOSTRA_SENHA_VALIDA, escrevendo “SENHA CORRETA” no display, acionando a rotina do motor-de-passo e soando o buzzer por 1 segundo.

Adicionalmente, uma função que impede o usuário de realizar muitas tentativas foi também implementada. Ela utiliza um desvio DJNZ, que recebe como registrador de entrada o registrador definido como TENTATIVAS e pula para a função LIMPA_LCD_E_MOSTRA_SENHA_INVALIDA enquanto ainda restarem tentativas.

Caso o registrador TENTATIVAS alcance o valor zero, ocorre um JMP para o fim do programa e o mesmo deve ser resetado fisicamente pelo botão RST da placa.

Conclusão

Durante esse experimento, foi possível aplicar vários dos conhecimentos adquiridos em sala de aula. Foram utilizadas diversas instruções logicas, aritméticas, de transferência de dados, de desvio e booleanas durante o projeto. Além disso, foi necessária a utilização dos ports físicos como dispositivos de I/O. Dessa forma o entendimento da equipe sobre a utilização destes para a leitura e escrita de dados, vindos de dispositivos externos, aumentou consideravelmente. Complementarmente, após a parte básica do trabalho ser finalizada, a equipe ainda utilizou ferramentas como interrupções e timers no código para expandir seus conhecimentos.

Adicionalmente, ainda foi possível notar a importância da organização e dos comentários no código, durante o desenvolvimento. Essa pratica foi fundamental para a equipe se localizar no código e poder reutilizar funções diversas vezes em partes diferentes do programa. Um exemplo disso ocorreu com os timers, que foram aproveitados em várias outras funções.

O resultado final foi muito satisfatório e realizou todas as funções e requisitos exigidos, mostrando que os membros da equipe entenderam o propósito da atividade, bem como os fundamentos teóricos aprendidos em sala de aula.

Referências

1. NICOLASSI, Denys E. C. Microcontrolador 8051 Detalhado. São Paulo: Editora Erica, 6ª edição.

2. Interfacing 16×2 LCD with 8051. Disponível em: <http://www.circuitstoday.com/interfacing-16x2-lcd-with-8051>. Acesso em 17 de março de 2017.

3. Driving a Unipolar stepper motor using ULN2803 + PIC16F887. Disponível em: <http://www.incrediblediy.com/2013/02/driving-unipolar-stepper-motor-using.html>. Acesso em 22 de março de 2017.

4. 2x16 LCD And 4x4 Keypad Interfacing With 8051 in Assembly Language Disponível em: <http://www.botskool.com/user-pages/tutorials/electronics/2x16-lcd-and-4x4-keypad-interfacing-8051-assembly-language>. Acesso em 18 de março de 2017.

Anexo A - Código em assembly comentado

A partir deste ponto, segue o código completo em ASSEMBLY comentado.

```
1   ////////////////////////////////////////////////////////
2   //                                                    //
3   //           PROJETO 01 - FECHADURA ELETRONICA        //
4   //                                                    //
5   // Requisitos:                                        //
6   // - LCD, teclado matricial, buzzer e motor de passos //
7   // - 02 senhas padroes na memoria de programa         //
8   // - Mensagem introdutoria personalizada              //
9   // - Validacao visual no LCD                          //
10  // - O teclado nao deve possuir bounce                //
11  //                                                    //
12  // @author: Leonardo Winter Pereira                   //
13  // @author: Rodrigo Yudi Endo                         //
14  //                                                    //
15  ////////////////////////////////////////////////////////
16
17  org 00h//2000h // Origem do codigo
18  ljmp __STARTUP__
19
20  org 03h//2003h // Inicio do codigo da interrupcao externa INT0
21  ljmp INT_INT0
22
23  org 0Bh//200Bh // Inicio do codigo da interrupcao interna gerada pelo TIMER/COUNTER 0
24  ljmp INT_TIMER0
25
26  org 013h//2013h // Inicio do codigo da interrupcao externa INT1
27  ljmp INT_INT1
28
29  org 01Bh//201Bh // Inicio do codigo da interrupcao interna gerada pelo TIMER/COUNTER 1
30  ljmp INT_TIMER1
31
32  org 023h//2023h // Inicio do codigo da interrupcao SERIAL
33  ljmp INT_SERIAL
34
35  ////////////////////////////////////////////
36  //       TABELA DE EQUATES DO PROGRAMA        //
37  ////////////////////////////////////////////
38
39  // PARA PLACA USB VERMELHA
40  RS              EQU P2.5            // COMANDO RS LCD
41  RW              EQU P2.6            // READ/WRITE
42  E_LCD           EQU P2.7            // COMANDO E (ENABLE) LCD
43
44  BUSYF           EQU P0.7            // BUSY FLAG
45
46  // LEDS DA PLACA
47  LED_SEG         EQU P3.6
48  LED1            EQU P3.7
49
50  // LINHAS E COLUNAS DO TECLADO MATRICIAL
51  TECLADO         EQU P1
52
53  COL1            EQU P1.0
54  COL2            EQU P1.1
55  COL3            EQU P1.2
56  COL4            EQU P1.3
57
58  LIN1            EQU P1.4
59  LIN2            EQU P1.5
60  LIN3            EQU P1.6
61  LIN4            EQU P1.7
62
63  // BUZZER
64  BUZZER          EQU P3.0
65
66  // ESTADOS DO MOTOR DE PASSO
67  ESTADO_UM       EQU P3.1
68  ESTADO_DOIS     EQU P3.2
69  ESTADO_TRES     EQU P3.3
70  ESTADO_QUATRO   EQU P3.4
71
72  // ENDEREÇOS DOS DIGITOS DOS DADOS DE ENTRADA
73  TECLADO_1       EQU 31h
```

```
  74   TECLADO_2       EQU 32h
  75   TECLADO_3       EQU 33h
  76   TECLADO_4       EQU 34h
  77
  78   // ENDEREÇOS DOS DIGITOS DE CADA SENHA
  79   SENHA1_1        EQU 35h
  80   SENHA1_2        EQU 36h
  81   SENHA1_3        EQU 37h
  82   SENHA1_4        EQU 38h
  83
  84   SENHA2_1        EQU 45h
  85   SENHA2_2        EQU 46h
  86   SENHA2_3        EQU 47h
  87   SENHA2_4        EQU 48h
  88
  89   TENTATIVAS      EQU 50h
  90
  91   TIMEOUT_LOW     EQU 51h
  92   TIMEOUT_HIGH    EQU 52h
  93   TIMEOUT_X1S     EQU 53h
  94
  95   ////////////////////////////////////////////////
  96   // REGIAO DA MEMORIA DE PROGRAMA COM AS STRINGS //
  97   ////////////////////////////////////////////////
  98
  99   org 30h//2030h
 100   SENHA_PADRAO_1:
 101           db      '2812', 00H
 102   SENHA_PADRAO_2:
 103           db      '0912', 00H
 104   STR_LOGO:
 105           DB      ' PROFS. LINDOS! ',00H
 106   STR_LEONARDO_PEREIRA:
 107           DB      'Leonardo Pereira',00H
 108   STR_RODRIGO_ENDO:
 109           DB      '  Rodrigo Endo  ',00H
 110   STR_SENHA:
 111           DB      'SENHA: ',00H
 112   STR_PRESSIONE_ENTER:
 113           DB      'PRESSIONE ENTER ',00H
 114   STR_SENHA_VALIDA:
 115           DB      '  SENHA VALIDA  ',00H
 116   STR_SENHA_INVALIDA:
 117           DB      ' SENHA INVALIDA ',00H
 118   STR_LIBERANDO_PORTA:
 119           DB      'LIBERANDO PORTA ',00H
 120   STR_TRANCANDO_PORTA:
 121           DB      'TRANCANDO PORTA ',00H
 122   STR_SISTEMA_BLOQUEADO:
 123           DB      'PORTA BLOQUEADA ',00H
 124   STR_TIMEOUT:
 125           DB      '    TIMEOUT     ',00H
 126
 127   ////////////////////////////////////////////
 128   //              INICIO DO PROGRAMA          //
 129   ////////////////////////////////////////////
 130
 131   __STARTUP__:
 132           CALL    TIMER_CONFIGURA_TIMER
 133           CALL    INT_CONFIGURA_INTERRUPCOES
 134
 135           MOV     R0, #0Fh        // R0 x 20 ms com o buzzer acionado - apenas para mostrar que o sistema
       esta inicializando
 136           LCALL   ACIONA_BUZZER
 137
 138           MOV     DPTR, #SENHA_PADRAO_1   // inicializa o DPTR com o endereco da senha 01
 139           MOV     R0, #00h                // R0 representa o caracter atual do texto a ser lido
 140           MOV     R1, #SENHA1_1           // R1 possui o endereco do registrador do primeiro numero da
       senha 01
 141           MOV     R2, #04h                // quantidade de digitos da senha 01
 142           CALL    LE_SENHA
 143
 144           MOV     DPTR, #SENHA_PADRAO_2   // inicializa o DPTR com o endereco da senha 02
```

```
145             MOV     R0, #00h                // R0 representa o caracter atual do texto a ser lido
146             MOV     R1, #SENHA2_1           // R1 possui o endereco do registrador do primeiro numero da
        senha 02
147             MOV     R2, #04h                // quantidade de digitos da senha 02
148             CALL    LE_SENHA
149
150             MOV     TENTATIVAS, #3h // Igual ao sistema de cartao de credito - Caso erre 3x seguidas a
        senha, bloqueia o sistema, tendo que reiniciar o sistema pelo botao fisico de RESET
151
152     MAIN:
153             CALL    INIDISP         // chama rotina de inicializacao do display 16x2
154             MOV     DPTR,#STR_LOGO  // seta o DPTR com o endereco da string LOGO
155             CALL    ESC_STR1        // escreve na primeira linha do display
156
157             // Atrasa 1s para escrever outra string
158             MOV     R1, #01h
159             CALL    TIMER_DELAY_1_S
160
161             CALL    CLR1L   // limpa primeira linha do display
162
163             MOV     DPTR, #STR_LEONARDO_PEREIRA // seta o DPTR com o endereco da string LEONARDO_PEREIRA
164             CALL    ESC_STR1                    // escreve na primeira linha do display
165
166             MOV     DPTR, #STR_RODRIGO_ENDO     // seta o DPTR com o endereco da string RODRIGO_ENDO
167             CALL    ESC_STR2                    // escreve na segunda linha do display
168
169             // Atrasa 1s para escrever outra string
170             MOV     R1, #01h
171             CALL    TIMER_DELAY_1_S
172
173     LIMPA_LCD_E_INICIA_SISTEMA:
174             // Limpa ambas as linhas do display
175             CALL    CLR1L
176             CALL    CLR2L
177
178             // Inicialmente, sera necessario o usuario apertar ENTER (simulando um sistema de bancos, por
        exemplo, onde voce precisa primeiro inserir o cartao para depois entrar com a senha)
179             MOV     DPTR,#STR_PRESSIONE_ENTER   // seta o DPTR com o endereco da string PRESSIONE_ENTER
180             CALL    ESC_STR1                    // escreve na primeira linha do display
181
182             ACALL   PRESS_ENT       // espera o pressionamento da tecla ENTER
183
184             MOV     R0, #05h
185             LCALL   ACIONA_BUZZER
186
187     LIMPA_LCD_E_REINICIA_SISTEMA:
188             CALL    CONFIGURA_VALORES_TIMER_1 // Temos 3 registradores separados para o TIMER_1 (sendo o
        TIMEOUT_X1S o mais importante -> TIMEOUT_X1S x 1s)
189             SETB    TR1 // Como queremos ter a opcao de TIMEOUT no projeto, aciona o TR1, que comeca a
        contagem do TIMER_1
190
191     LIMPA_LCD_E_PEDE_SENHA:
192             // Limpa ambas as linhas do display
193             CALL    CLR1L
194             CALL    CLR2L
195
196             // Apos apertar ENTER, pede-se a senha (que precisa bater com uma das 2 senhas pre-definidas)
197             MOV     DPTR,#STR_SENHA // seta o DPTR com o endereco da string SENHA
198             CALL    ESC_STR1        // escreve na primeira linha do display
199
200             // Move para a coluna 07 da primeira linha do display
201             MOV     R0, #00h    // linha
202             MOV     R1, #07h    // coluna
203             CALL    GOTOXY
204
205             MOV     R1, #TECLADO_1  // ponteiro para a primeira leitura do teclado
206             MOV     R3, #1h         // primeiro digito a ser lido, esse registrador e incrementado
        automaticamente ao terminar de varrer o teclado
207             MOV     R7, #04h        // quantidade de digitos a serem lidos consecutivamente
208     LE_4_DIGITOS:
209             MOV     R0, #0Ah        // R0 x 20 ms de delay - para nao sentir o efeito de bounce no teclado
        matricial
210             ACALL   TIMER_DELAY_20_MS
```

```
211
212             ACALL   VARREDURA_TECLADO   // permanece varrendo o teclado ate que alguma tecla seja pressionada
213             CALL    ESCREVE_ASTERISCO   // escreve apenas um * na tela, semelhante a como funciona um
        sistema de cartao / banco
214
215             INC R1  // incrementa o ponteiro R1, que aponta agora para o proximo digito a ser lido
216             INC R3
217
218             DJNZ R7, LE_4_DIGITOS // enquanto nao ler os 4 digitos, se mantem no loop
219
220             // Apos ler os 4 digitos da senha, e necessario testar a senha com as duas senhas gravadas na ROM
221             MOV     R0, #0Ah        // R0 x 20 ms de delay - para nao sentir o efeito de bounce no teclado
        matricial
222             ACALL   TIMER_DELAY_20_MS
223
224             ACALL   PRESS_ENT_OU_CLR        // espera o pressionamento da tecla ENTER ou CLR
225
226             JMP     LIMPA_LCD_E_INICIA_SISTEMA
227
228     ////////////////////////////////////////////////
229     // NOME: LE_SENHA e LE_SENHA_ROM                //
230     // DESCRICAO: ROTINA QUE LE A SENHA CODIFICADA NA   //
231     // ROM E ARMAZENA NO ENDERECO APONTADO POR R1       //
232     // ENTRADA: R1 -> Ponteiro para a variavel          //
233     //          DPTR -> Endereco da posicao da senha    //
234     //          R0 -> Caracter atual da senha           //
235     // SAIDA: -                                         //
236     // DESTROI: R1, R0, R2, A                           //
237     ////////////////////////////////////////////////
238     LE_SENHA:
239             CALL    LE_SENHA_ROM
240
241             INC     R1
242
243             DJNZ    R2, LE_SENHA
244
245             RET
246
247     LE_SENHA_ROM:
248             MOV     A, R0
249             MOVC    A, @A + DPTR
250             MOV     @R1, A
251
252             INC     R0
253
254             CALL    CONV_ASCII_TO_NUMBER
255
256             RET
257
258     ////////////////////////////////////////////////
259     // NOME: CONV_ASCII_TO_NUMBER                   //
260     // DESCRICAO: ROTINA QUE CONVERTE UM ASCII LIDO     //
261     // PARA UM VALOR NUMERICO QUE PODE SER TESTADO      //
262     // ENTRADA: R1 -> Ponteiro para a variavel          //
263     // SAIDA: -                                         //
264     // DESTROI: A                                       //
265     ////////////////////////////////////////////////
266     CONV_ASCII_TO_NUMBER:
267             MOV     A, @R1
268
269             ADD     A, #-30h
270
271             MOV     @R1, A
272
273             RET
274
275     ////////////////////////////////////////////////
276     // NOME: ESCREVE_ASTERISCO                      //
277     // DESCRICAO: ROTINA QUE ESCREVE UM * NO LCD        //
278     // ENTRADA: -                                       //
279     // SAIDA: -                                         //
280     // DESTROI: A                                       //
281     ////////////////////////////////////////////////
```

```
282   ESCREVE_ASTERISCO:
283           MOV     R0, #05h
284           LCALL   ACIONA_BUZZER
285
286           MOV     A, #2Ah // valor em hexa para '*'
287           CALL    ESCDADO
288
289           RET
290
291   ////////////////////////////////////////////////
292   // NOME: PRESS_ENT                            //
293   // DESCRICAO: Ao digitar os 4 digitos da senha, e   //
294   // necessario apertar ENTER                   //
295   // ENTRADA: -                                 //
296   // SAIDA: -                                   //
297   // DESTROI: A                                 //
298   ////////////////////////////////////////////////
299   PRESS_ENT:
300           // Apenas LIN4 esta em 0
301           MOV     TECLADO, #01111111b
302
303   VARRE_ENT:
304           JNB     COL4, CONTINUA_PROG     // se apertou ENTER, testa senha
305
306           JMP     VARRE_ENT
307   CONTINUA_PROG:
308           RET
309
310   ////////////////////////////////////////////////
311   // NOME: PRESS_ENT_OU_CLR                     //
312   // DESCRICAO: Ao digitar os 4 digitos da senha, e   //
313   // necessario apertar ENTER                   //
314   // ENTRADA: -                                 //
315   // SAIDA: -                                   //
316   // DESTROI: A                                 //
317   ////////////////////////////////////////////////
318   PRESS_ENT_OU_CLR:
319           // Apenas LIN4 esta em 0
320           MOV     TECLADO, #01111111b
321
322   VARRE_ENT_OU_CLR:
323           JNB     COL2, CLEAR         // se apertou CLR, limpa senha de entrada
324           JNB     COL4, TESTA_SENHA1  // se apertou ENTER, testa senha
325
326           JNB     TR1, FINALIZA_VARREDURA_POR_TIMEOUT
327
328           JMP     VARRE_ENT_OU_CLR
329
330   ////////////////////////////////////////////////
331   // NOME: CLEAR                                //
332   // DESCRICAO: Reseta os digitos do teclado para as  //
333   // senhas normais. Os POPs sao necessarios por      //
334   // causa do ACALL feito no main               //
335   // ENTRADA: -                                 //
336   // SAIDA: -                                   //
337   // DESTROI: A                                 //
338   ////////////////////////////////////////////////
339   CLEAR:
340           MOV     R0, #05h
341           LCALL   ACIONA_BUZZER
342
343           POP     ACC
344           POP     ACC
345           JMP     LIMPA_LCD_E_PEDE_SENHA
346
347   ////////////////////////////////////////////////
348   // NOME: VARREDURA_TECLADO                    //
349   // DESCRICAO: Varredura de teclado            //
350   // ENTRADA:                                   //
351   // SAIDA:                                     //
352   // DESTROI:                                   //
353   ////////////////////////////////////////////////
354   VARREDURA_TECLADO:
```

```
355             CLR     LIN1
356             SETB    LIN2
357             SETB    LIN3
358             SETB    LIN4
359             JNB     COL2, DIGITO1
360             JNB     COL3, DIGITO2
361             JNB     COL4, DIGITO3
362
363             CLR     LIN2
364             SETB    LIN1
365             JNB     COL2, DIGITO4
366             JNB     COL3, DIGITO5
367             JNB     COL4, DIGITO6
368
369             CLR     LIN3
370             SETB    LIN2
371             JNB     COL2, DIGITO7
372             JNB     COL3, DIGITO8
373             JNB     COL4, DIGITO9
374
375             CLR     LIN4
376             SETB    LIN3
377             JNB     COL2, CLEAR
378             JNB     COL3, DIGITO0
379
380             JNB     TR1, FINALIZA_VARREDURA_POR_TIMEOUT
381
382             JMP     VARREDURA_TECLADO
383
384     FINALIZA_VARREDURA_POR_TIMEOUT:
385             CALL    ESCREVE_TIMEOUT
386
387             POP     ACC
388             JMP     LIMPA_LCD_E_INICIA_SISTEMA
389
390     ////////////////////////////////////////////////////////////////////////
391     // NOME: DIGITOX (X = [0,9])                                          //
392     // DESCRICAO: Coloca o valor X em R1 e grava no dígito correspondente //
393     // ao valor atual de R2                                               //
394     // ENTRADA: R1 -> ponteiro para o caracter da senha atual a ser lido  //
395     // SAIDA:                                                             //
396     // DESTROI: A                                                         //
397     ////////////////////////////////////////////////////////////////////////
398     DIGITO1: MOV A, #1h
399              AJMP GRAVA_DIGITO
400     DIGITO2: MOV A, #2h
401              AJMP GRAVA_DIGITO
402     DIGITO3: MOV A, #3h
403              AJMP GRAVA_DIGITO
404     DIGITO4: MOV A, #4h
405              AJMP GRAVA_DIGITO
406     DIGITO5: MOV A, #5h
407              AJMP GRAVA_DIGITO
408     DIGITO6: MOV A, #6h
409              AJMP GRAVA_DIGITO
410     DIGITO7: MOV A, #7h
411              AJMP GRAVA_DIGITO
412     DIGITO8: MOV A, #8h
413              AJMP GRAVA_DIGITO
414     DIGITO9: MOV A, #9h
415              AJMP GRAVA_DIGITO
416     DIGITO0: MOV A, #0h
417              AJMP GRAVA_DIGITO
418
419     GRAVA_DIGITO:
420             CJNE R3, #1h, GRAVA_TECLADO_2
421             MOV @R1, A
422
423             RET
424
425     ////////////////////////////////////////////////
426     // NOME: TESTA_SENHAX [1,2]                   //
427     // DESCRICAO: Compara os dígitos fornecidos pelo    //
```

```
428   // usuário com a senha X                                //
429   // ENTRADA: -                                           //
430   // SAIDA: -                                             //
431   // DESTROI: A                                           //
432   ////////////////////////////////////////////////////////
433   TESTA_SENHA1:
434           MOV     R0, #05h
435           LCALL   ACIONA_BUZZER
436
437           CLR     TR1 // Se chegou ate aqui é porque o usuario apertou ENTER, entao nao precisa mais dar
      TIMEOUT
438
439           MOV     A, SENHA1_1
440           CJNE    A, TECLADO_1, TESTA_SENHA2
441           MOV     A, SENHA1_2
442           CJNE    A, TECLADO_2, TESTA_SENHA2
443           MOV     A, SENHA1_3
444           CJNE    A, TECLADO_3, TESTA_SENHA2
445           MOV     A, SENHA1_4
446           CJNE    A, TECLADO_4, TESTA_SENHA2
447           JMP     LIMPA_LCD_E_MOSTRA_SENHA_VALIDA
448
449   TESTA_SENHA2:
450           MOV     A, SENHA2_1
451           CJNE    A, TECLADO_1, LIMPA_LCD_E_MOSTRA_SENHA_INVALIDA
452           MOV     A, SENHA2_2
453           CJNE    A, TECLADO_2, LIMPA_LCD_E_MOSTRA_SENHA_INVALIDA
454           MOV     A, SENHA2_3
455           CJNE    A, TECLADO_3, LIMPA_LCD_E_MOSTRA_SENHA_INVALIDA
456           MOV     A, SENHA2_4
457           CJNE    A, TECLADO_4, LIMPA_LCD_E_MOSTRA_SENHA_INVALIDA
458           JMP     LIMPA_LCD_E_MOSTRA_SENHA_VALIDA
459
460   ////////////////////////////////////////////////////////////////////////
461   // NOME: GRAVA_TECLADO_X [2,4]                                        //
462   // DESCRICAO: Grava o valor de R1 e no dígito X do teclado            //
463   // ENTRADA: R1                                                        //
464   // SAIDA: TECLADO_X                                                   //
465   // DESTROI:                                                           //
466   ////////////////////////////////////////////////////////////////////////
467   GRAVA_TECLADO_2: CJNE R3, #2h, GRAVA_TECLADO_3
468                    AJMP GRAVA_TECLADO_X
469   GRAVA_TECLADO_3: CJNE R3, #3h, GRAVA_TECLADO_4
470   GRAVA_TECLADO_4: AJMP GRAVA_TECLADO_X
471
472   GRAVA_TECLADO_X:
473            MOV @R1, A
474
475            RET
476
477   LIMPA_LCD_E_MOSTRA_SENHA_INVALIDA:
478           // Limpa ambas as linhas do display
479           CALL    CLR1L
480           CALL    CLR2L
481
482           MOV     DPTR,#STR_SENHA_INVALIDA    // string da primeira linha
483           CALL    ESC_STR1                    // escreve na primeira linha
484
485           // Aciona o buzzer por 2s
486           MOV     R0, #064h
487           LCALL   ACIONA_BUZZER
488
489           // Atrasa 3s (contando o tempo do buzzer) para reiniciar o sistema (caso nao tenha errado a
      senha 3x em sequencia)
490           MOV     R1, #01h
491           CALL    TIMER_DELAY_1_S
492
493           DJNZ    TENTATIVAS, REINICIA_SISTEMA
494
495           CALL    CLR1L
496           MOV     DPTR,#STR_SISTEMA_BLOQUEADO // string da primeira linha
497           CALL    ESC_STR1                    // escreve na primeira linha
498
```

```
499             JMP     FIM
500
501     REINICIA_SISTEMA:
502             RET
503
504     LIMPA_LCD_E_MOSTRA_SENHA_VALIDA:
505             MOV     TENTATIVAS, #3h // reinicia numero de tentativas
506
507             // Limpa ambas as linhas do display
508             CALL    CLR1L
509             CALL    CLR2L
510
511             MOV     DPTR,#STR_SENHA_VALIDA  // string da primeira linha
512             CALL    ESC_STR1                // escreve na primeira linha
513
514             // Aciona por 1s o buzzer
515             MOV     R0, #032h
516             LCALL   ACIONA_BUZZER
517
518             MOV     DPTR, #STR_LIBERANDO_PORTA  // string da segunda linha
519             CALL    ESC_STR2                    // escreve na segunda linha
520
521             // Aciona o mecanismo (motor de passos) para abrir a fechadura
522             MOV     R5, #0Dh
523             LCALL   ABRE_FECHADURA
524
525             // Atrasa 3s para escrever outra string
526             MOV     R1, #03h
527             CALL    TIMER_DELAY_1_S
528
529             CALL    CLR2L                       // limpa a segunda linha do LCD
530             MOV     DPTR, #STR_TRANCANDO_PORTA  // string da segunda linha
531             CALL    ESC_STR2                    // escreve na segunda linha
532
533             // Aciona o mecanismo (motor de passos) para trancar a fechadura (basicamente faz o caminho
        oposto ao ABRE_FECHADURA)
534             MOV     R5, #0Dh
535             LCALL   TRANCA_FECHADURA
536
537             RET
538
539     ////////////////////
540     // ACIONA BUZZER //
541     //  R0 x 20 MS   //
542     ////////////////////
543     ACIONA_BUZZER:
544             SETB    BUZZER
545             ACALL   TIMER_DELAY_20_MS
546             CLR     BUZZER
547
548             RET
549
550     ////////////////////////////////////////////
551     // CODIGOS RELACIONADOS AO MOTOR DE PASSOS  //
552     ////////////////////////////////////////////
553     ABRE_FECHADURA:
554             SETB    ESTADO_UM
555             CLR     ESTADO_DOIS
556             CLR     ESTADO_TRES
557             CLR     ESTADO_QUATRO
558
559             MOV     R0, #05h
560             ACALL   TIMER_DELAY_20_MS
561
562             CLR     ESTADO_UM
563             SETB    ESTADO_DOIS
564
565             MOV     R0, #05h
566             ACALL   TIMER_DELAY_20_MS
567
568             CLR     ESTADO_DOIS
569             SETB    ESTADO_TRES
570
```

```
571             MOV     R0, #05h
572             ACALL   TIMER_DELAY_20_MS
573
574             CLR     ESTADO_TRES
575             SETB    ESTADO_QUATRO
576
577             MOV     R0, #05h
578             ACALL   TIMER_DELAY_20_MS
579
580             DJNZ    R5, ABRE_FECHADURA
581
582             RET
583
584     TRANCA_FECHADURA:
585             CLR     ESTADO_UM
586             CLR     ESTADO_DOIS
587             CLR     ESTADO_TRES
588             SETB    ESTADO_QUATRO
589
590             MOV     R0, #01h
591             ACALL   TIMER_DELAY_20_MS
592
593             SETB    ESTADO_TRES
594             CLR     ESTADO_QUATRO
595
596             MOV     R0, #01h
597             ACALL   TIMER_DELAY_20_MS
598
599             SETB    ESTADO_DOIS
600             CLR     ESTADO_TRES
601
602             MOV     R0, #01h
603             ACALL   TIMER_DELAY_20_MS
604
605             SETB    ESTADO_UM
606             CLR     ESTADO_DOIS
607
608             MOV     R0, #01h
609             ACALL   TIMER_DELAY_20_MS
610
611             DJNZ    R5, TRANCA_FECHADURA
612
613             RET
614
615     ////////////////////////////////////////////////////
616     //                                                //
617     ////////////////////////////////////////////////////
618     ESCREVE_TIMEOUT:
619             CALL    CLR1L
620             CALL    CLR2L
621
622             // Apos apertar ENTER, pede-se a senha (que precisa bater com uma das 2 senhas pre-definidas)
623             MOV     DPTR,#STR_TIMEOUT   // seta o DPTR com o endereco da string SENHA
624             CALL    ESC_STR1            // escreve na primeira linha do display
625
626             // Atrasa 2s para escrever outra string
627             MOV     R1, #02h
628             CALL    TIMER_DELAY_1_S
629
630             RET
631
632     ////////////////////////////////////////////////
633     //        INICIO DOS CODIGOS PARA LCD         //
634     ////////////////////////////////////////////////
635
636     ////////////////////////////////////////////////////
637     // NOME: INIDISP                                  //
638     // DESCRICAO: ROTINA DE INICIALIZACAO DO DISPLAY  //
639     // LCD 16x2 --- PROGRAMA CARACTER 5x7, LIMPA      //
640     // DISPLAY E POSICIONA (0,0)                      //
641     // ENTRADA: -                                     //
642     // SAIDA: -                                       //
643     // DESTROI: R0, R2                                //
```

```
644   ////////////////////////////////////////////////
645   INIDISP:
646           MOV     R0,#38H         // UTILIZACAO: 8 BITS, 2 LINHAS, 5x7
647           MOV     R2,#05          // ESPERA 5ms
648           CALL    ESCINST         // ENVIA A INSTRUCAO
649
650           MOV     R0,#38H         // UTILIZACAO: 8 BITS, 2 LINHAS, 5x7
651           MOV     R2,#01          // ESPERA 1ms
652           CALL    ESCINST         // ENVIA A INSTRUCAO
653
654           MOV     R0,#06H         // INSTRUCAO DE MODO DE OPERACAO
655           MOV     R2,#01          // ESPERA 1ms
656           CALL    ESCINST         // ENVIA A INSTRUCAO
657
658           MOV     R0,#0CH         // INSTRUCAO DE CONTROLE ATIVO/INATIVO
659           MOV     R2,#01          // ESPERA 1ms
660           CALL    ESCINST         // ENVIA A INSTRUCAO
661
662           MOV     R0,#01H         // INSTRUCAO DE LIMPEZA DO DISPLAY
663           MOV     R2,#02          // ESPERA 2ms
664           CALL    ESCINST         // ENVIA A INSTRUCAO
665
666           RET
667
668   ////////////////////////////////////////////////
669   // NOME: ESCINST                               //
670   // DESCRICAO: ROTINA QUE ESCREVE INSTRUCAO PARA O   //
671   // DISPLAY E ESPERA DESOCUPAR                  //
672   // ENTRADA: R0 = INSTRUCAO A SER ESCRITA NO MODULO  //
673   //          R2 = TEMPO DE ESPERA EM ms         //
674   // SAIDA: -                                    //
675   // DESTROI: R0, R2                             //
676   ////////////////////////////////////////////////
677   ESCINST:
678           CLR     RW              // MODO ESCRITA NO LCD
679           CLR     RS              // RS  = 0 (SELECIONA REG. DE INSTRUCOES)
680           SETB    E_LCD           // E = 1 (HABILITA LCD)
681
682           MOV     P0,R0           // INSTRUCAO A SER ESCRITA
683
684           CLR     E_LCD           // E = 0 (DESABILITA LCD)
685
686           MOV     P0,#0xFF        // PORTA 0 COMO ENTRADA
687
688           SETB    RW              // MODO LEITURA NO LCD
689           SETB    E_LCD           // E = 1 (HABILITA LCD)
690
691   ESCI1:  JB  BUSYF,ESCI1         // ESPERA BUSY FLAG = 0
692
693           CLR     E_LCD           // E = 0 (DESABILITA LCD)
694
695           RET
696
697   ////////////////////////////////////////////////
698   // NOME: GOTOXY                                //
699   // DESCRICAO: ROTINA QUE POSICIONA O CURSOR    //
700   // ENTRADA: R0 = LINHA (0 A 1)                 //
701   //          R1 = COLUNA (0 A 15)               //
702   // SAIDA: -                                    //
703   // DESTROI: R0,R2                              //
704   ////////////////////////////////////////////////
705   GOTOXY:
706           PUSH    ACC
707
708           MOV     A,#80H
709           CJNE    R0,#01,GT1      // SALTA SE COLUNA 0
710
711           MOV     A,#0C0H
712
713   GT1:    ORL     A,R1            // CALCULA O ENDERECO DA MEMORIA DD RAM
714           MOV     R0,A
715           MOV     R2,#01          // ESPERA 1ms
716
```

```
717           CALL    ESCINST         // ENVIA PARA O MODULO DISPLAY
718
719           POP     ACC
720
721           RET
722
723   ////////////////////////////////////////////////
724   // NOME: CLR1L                                //
725   // DESCRICAO: ROTINA QUE APAGA PRIMEIRA LINHA DO    //
726   // DISPLAY LCD E POSICIONA NO INICIO          //
727   // ENTRADA: -                                 //
728   // SAIDA: -                                   //
729   // DESTROI: R0,R1                             //
730   ////////////////////////////////////////////////
731   CLR1L:
732           PUSH    ACC
733
734           MOV     R0,#00              // LINHA
735           MOV     R1,#00
736
737           CALL    GOTOXY
738
739           MOV     R1,#16              // CONTADOR
740
741   CLR1L1: MOV     A,#' '              // ESPACO
742
743           CALL    ESCDADO
744
745           DJNZ    R1,CLR1L1
746           MOV     R0,#00              // LINHA
747           MOV     R1,#00
748
749           CALL    GOTOXY
750
751           POP     ACC
752
753           RET
754
755   ////////////////////////////////////////////////
756   // NOME: CLR2L                                //
757   // DESCRICAO: ROTINA QUE APAGA SEGUNDA LINHA DO     //
758   // DISPLAY LCD E POSICIONA NO INICIO          //
759   // ENTRADA: -                                 //
760   // SAIDA: -                                   //
761   // DESTROI: R0,R1                             //
762   ////////////////////////////////////////////////
763   CLR2L:
764           PUSH    ACC
765
766           MOV     R0,#01              // LINHA
767           MOV     R1,#00
768
769           CALL    GOTOXY
770
771           MOV     R1,#16              // CONTADOR
772
773   CLR2L1: MOV     A,#' '              // ESPACO
774
775           CALL    ESCDADO
776
777           DJNZ    R1,CLR2L1
778           MOV     R0,#01              // LINHA
779           MOV     R1,#00
780
781           CALL    GOTOXY
782
783           POP     ACC
784
785           RET
786
787   ////////////////////////////////////////////////
788   // NOME: ESCDADO                              //
789   // DESCRICAO: ROTINA QUE ESCREVE DADO NO DISPLAY    //
```

```
790   // ENTRADA: A = DADO A SER ESCRITO NO DISPLAY       //
791   // SAIDA: -                                        //
792   // DESTROI: R0                                     //
793   /////////////////////////////////////////////////////
794   ESCDADO:
795           CLR     RW              // MODO ESCRITA NO LCD
796           SETB    RS              // RS  = 1 (SELECIONA REG. DE DADOS)
797           SETB    E_LCD           // LCD = 1 (HABILITA LCD)
798
799           MOV     P0,A            // ESCREVE NO BUS DE DADOS
800
801           CLR     E_LCD           // LCD = 0 (DESABILITA LCD)
802
803           MOV     P0,#0xFF        // PORTA 0 COMO ENTRADA
804
805           SETB    RW              // MODO LEITURA NO LCD
806           CLR     RS              // RS = 0 (SELECIONA INSTRUÇÃO)
807           SETB    E_LCD           // E = 1 (HABILITA LCD)
808
809   ESCD1:  JB      BUSYF,ESCD1     // ESPERA BUSY FLAG = 0
810
811           CLR     E_LCD           // E = 0 (DESABILITA LCD)
812
813           RET
814
815   /////////////////////////////////////////////////////
816   // NOME: MSTRING                                   //
817   // DESCRICAO: ROTINA QUE ESCREVE UMA STRING DA ROM  //
818   // NO DISPLAY A PARTIR DA POSICAO DO CURSOR         //
819   // ENTRADA: DPTR = ENDERECO INICIAL DA STRING NA    //
820   // MEMORIA ROM FINALIZADA POR 00H                   //
821   // SAIDA: -                                        //
822   // DESTROI: A,DPTR,R0                              //
823   /////////////////////////////////////////////////////
824   MSTRING:
825             CLR     A
826             MOVC    A,@A+DPTR       // CARACTER DA MENSAGEM EM A
827
828             JZ      MSTR1
829
830             LCALL   ESCDADO         // ESCREVE O DADO NO DISPLAY
831
832             INC     DPTR
833
834             SJMP    MSTRING
835
836   MSTR1:    RET
837
838   /////////////////////////////////////////////////////
839   // NOME: MSTRINGX                                  //
840   // DESCRICAO: ROTINA QUE ESCREVE UMA STRING DA RAM  //
841   // NO DISPLAY A PARTIR DA POSICAO DO CURSOR         //
842   // ENTRADA: DPTR = ENDERECO INICIAL DA STRING NA    //
843   // MEMORIA RAM FINALIZADA POR 00H                   //
844   // SAIDA: -                                        //
845   // DESTROI: A,DPTR,R0                              //
846   /////////////////////////////////////////////////////
847   MSTRINGX:
848             MOVX    A,@DPTR         // CARACTER DA MENSAGEM EM A
849
850             JZ      MSTR21
851
852             LCALL   ESCDADO         //ESCREVE O DADO NO DISPLAY
853
854             INC     DPTR
855
856             SJMP    MSTRINGX
857
858   MSTR21:   RET
859
860   /////////////////////////////////////////////////////
861   // NOME: ESC_STR1                                  //
862   // DESCRICAO: ROTINA QUE ESCREVE UMA STRING NO      //
```

```
863   // DISPLAY A PARTIR DO INICIO DA PRIMEIRA LINHA     //
864   // ENTRADA: DPTR = ENDERECO INICIAL DA STRING NA    //
865   // MEMORIA ROM FINALIZADA POR 00H                   //
866   // SAIDA: -                                         //
867   // DESTROI: R0,A,DPTR                               //
868   ////////////////////////////////////////////////////
869   ESC_STR1:
870             // PRIMEIRA LINHA E PRIMEIRA COLUNA
871             MOV     R0,#00
872             MOV     R1,#00
873
874             JMP     ESC_S
875
876   ////////////////////////////////////////////////////
877   // NOME: ESC_STR2                                   //
878   // DESCRICAO: ROTINA QUE ESCREVE UMA STRING NO      //
879   // DISPLAY A PARTIR DO INICIO DA SEGUNDA LINHA      //
880   // ENTRADA: DPTR = ENDERECO INICIAL DA STRING NA    //
881   // MEMORIA ROM FINALIZADA POR 00H                   //
882   // SAIDA: -                                         //
883   // DESTROI: R0,A,DPTR                               //
884   ////////////////////////////////////////////////////
885   ESC_STR2:
886             // SEGUNDA LINHA E PRIMEIRA COLUNA
887             MOV     R0,#01
888             MOV     R1,#00
889
890   ESC_S:    LCALL   GOTOXY          // POSICIONA O CURSOR
891
892             LCALL   MSTRING
893
894             RET
895
896   ////////////////////////////////////////////////////
897   // NOME: CUR_ON E CUR_OFF                           //
898   // DESCRICAO: ROTINA CUR_ON => LIGA CURSOR DO LCD   //
899   //         ROTINA CUR_OFF => DESLIGA CURSOR DO LCD  //
900   // ENTRADA: -                                       //
901   // SAIDA: -                                         //
902   ; DESTROI: R0,R2                                   //
903   ////////////////////////////////////////////////////
904   CUR_ON:
905             MOV     R0,#0FH             // INST.CONTROLE ATIVO (CUR ON)
906             SJMP    CUR1
907
908   CUR_OFF:
909             MOV     R0,#0CH             // INST. CONTROLE INATIVO (CUR OFF)
910
911   CUR1:     MOV     R2,#01
912
913             CALL    ESCINST             // ENVIA A INSTRUCAO
914
915             RET
916
917   ////////////////////////////////////////////
918   //       CODIGOS RELACIONADOS AO TIMER        //
919   ////////////////////////////////////////////
920
921   TIMER_CONFIGURA_TIMER:
922           MOV     TMOD, #00100001b // Seta o TIMER_0 para o modo 01 (16 bits) e o TIMER_1 para o modo 02
      (8 bits com reset)
923
924           // Para o TIMER_0, TH0 e TL0 representam o necessario para um delay de 20ms
925           MOV     TH0, #HIGH(65535 - 43350)
926           MOV     TL0, #LOW(65535 - 43350)
927
928           ////////////////////////////////////////////////
929           // Aqui configuramos o TIMER_1 (para o TIMEOUT) //
930           //    Interrupcao a ser chamada a cada 200 us   //
931           ////////////////////////////////////////////////
932
933           MOV     TH1, #0FFh
934           MOV     TL1, #05h
```

```
935
936             RET
937
938     CONFIGURA_VALORES_TIMER_1:
939             MOV     TIMEOUT_LOW,    #0FFh
940             MOV     TIMEOUT_HIGH,   #01Eh
941             MOV     TIMEOUT_X1S,    #014h
942
943             RET
944
945     ////////////////////////////////////////////////
946     // NOME: TIMER_DELAY_20_MS                    //
947     // DESCRICAO: INTRODUZ UM ATRASO DE 20 MS     //
948     // P.ENTRADA: R0 => (R0 x 20) ms              //
949     // P.SAIDA: -                                 //
950     // ALTERA: R0                                 //
951     ////////////////////////////////////////////////
952     TIMER_DELAY_20_MS:
953             CLR TF0
954             SETB TR0
955
956             JNB TF0, $
957
958             CLR TF0
959             CLR TR0
960
961             DJNZ R0, TIMER_DELAY_20_MS
962
963             RET
964
965     ////////////////////////////////////////////////
966     // NOME: TIMER_DELAY_1_S                      //
967     // DESCRICAO: INTRODUZ UM ATRASO DE 1 S       //
968     // P.ENTRADA: R1 = y => (y x 1) s             //
969     // P.SAIDA: -                                 //
970     // ALTERA: R1                                 //
971     ////////////////////////////////////////////////
972     TIMER_DELAY_1_S:
973             MOV     R0, #50d
974             CALL    TIMER_DELAY_20_MS
975
976             DJNZ    R1, TIMER_DELAY_1_S
977
978             RET
979
980     ////////////////////////////////////////////
981     // INICIO DOS CODIGOS GERADOS POR INTERRUPCAO //
982     ////////////////////////////////////////////
983
984     INT_CONFIGURA_INTERRUPCOES:
985             MOV     IE, #10001000b // Configura interrupcao apenas para o TIMER_1
986             MOV     IP, #00001000b // da prioridade alta para o TIMER_1
987
988             RET
989
990     /*
991     *
992     */
993     INT_INT0:
994             RETI
995
996     /*
997     *
998     */
999     INT_TIMER0:
1000            RETI
1001
1002    /*
1003    *
1004    */
1005    INT_INT1:
1006            RETI
1007
```

```
1008    /*
1009    * Para podermos utilizar valores elevados para nosso TIMEOUT, decidimos
1010    * utilizar até 3 bytes (TIMEOUT_LOW, TIMEOUT_HIGH, TIMEOUT_X1S) para definir
1011    * o tempo do nosso TIMEOUT
1012    *
1013    * Mantendo TIMEOUT_LOW em 0xFF e TIMEOUT_HIGH em 0x1E, juntamente com os 200 us
1014    * que a interrupcao e chamada, temos aproximadamente 1 s.
1015    * Desta forma, basta configurar o TIMEOUT_X1S para a quantidade de segundos desejado
1016    * Da forma como esta configurado, e possivel configurar o sistema para trabalhar com
1017    * um TIMEOUT de ate 255 s ( ~4.25 min)
1018    */
1019    INT_TIMER1:
1020            PUSH    ACC
1021            PUSH    PSW
1022
1023            MOV     TL1, #05h
1024            CLR     TF1
1025
1026            DJNZ    TIMEOUT_LOW, FINALIZA_TIMER_2
1027            MOV     TIMEOUT_LOW, #0FFh
1028
1029            DJNZ    TIMEOUT_HIGH, FINALIZA_TIMER_2
1030            MOV     TIMEOUT_HIGH, #01Eh
1031
1032            DJNZ    TIMEOUT_X1S, FINALIZA_TIMER_2
1033            MOV     TIMEOUT_X1S, #014h
1034
1035            CLR     TR1
1036
1037    FINALIZA_TIMER_2:
1038            POP     PSW
1039            POP     ACC
1040
1041            RETI
1042
1043    /*
1044    *
1045    */
1046    INT_SERIAL:
1047            RETI
1048
1049    FIM:
1050            JMP $
1051            END
```