Universidade Federal do ABC

Projeto 1

Alarme Digital com o Kit EXSTO XM852

Docente:

Germán Carlos Santos Quispe.

Discentes:

Gilmar Correia Jeronimo

Lucas Barboza Moreira Pinheiro

16 de Julho de 2019

Universidade Federal do ABC

Projeto 1

Alarme Digital com o Kit EXSTO XM852

Trabalho para Avaliação na Disciplina de Aplicações de Microcontroladores da Universidade Federal do ABC.

Docente:

Germán Carlos Santos Quispe.

Discentes:

Gilmar Correia Jeronimo - 11014515

Lucas Barboza Moreira Pinheiro - 11017015

16 de Julho de 2019

Sumário

1 Introdução

O projeto consiste na programação de um relógio-despertador utilizando o kit MX 852, capaz de exibir em um conjunto de 4 displays a contagem em tempo real do horário ou da data, receber as informações de horário e data do usuário por meio do teclado matricial e acionar um despertador (neste caso, o conjunto de quatro LEDs do kit foi utilizado para representar o acionamento do despertador), configurado de acordo com as informações entradas pelo usuário. Foram adicionadas como funcionalidades extra a validação do horário e da data e a exibição do dia da semana.

O procedimento para a configuração do relógio despertador deve seguir os seguintes passos:

- Ligar o kit XM 852
- Digitar o horário na forma hh:mm, onde h é o dígito da hora e m é o dígito do minuto
- Aguardar o display zerar
- Digitar o dia e o mês na forma dd:MM, onde d é o dígito do dia e M é o dígito do mês
- Aguardar o display zerar
- Digitar o ano na forma aaaa, onde a é o dígito do ano
- Aguardar o display zerar
- Caso o usuário queira verificar o dia e o mês, pressionar o botão D. Para retornar ao horário, pressionar D novamente
- Caso o usuário queira verificar o ano, pressionar o botão A. Para retornar ao horário, pressionar A novamente
- Caso o usuário queira verificar o dia da semana, pressionar o botão E. Para retornar ao horário, pressionar E novamente
- Para configurar o alarme:
  - Pressionar o botão C
  - Digitar o horário em que o despertador acionará
  - Aguardar o display zerar
  - Digitar o dia e o mês em que o despertador acionará

A sequência de passos para configurar o relógio-despertador está representada pelo fluxograma da figura Figura 1.

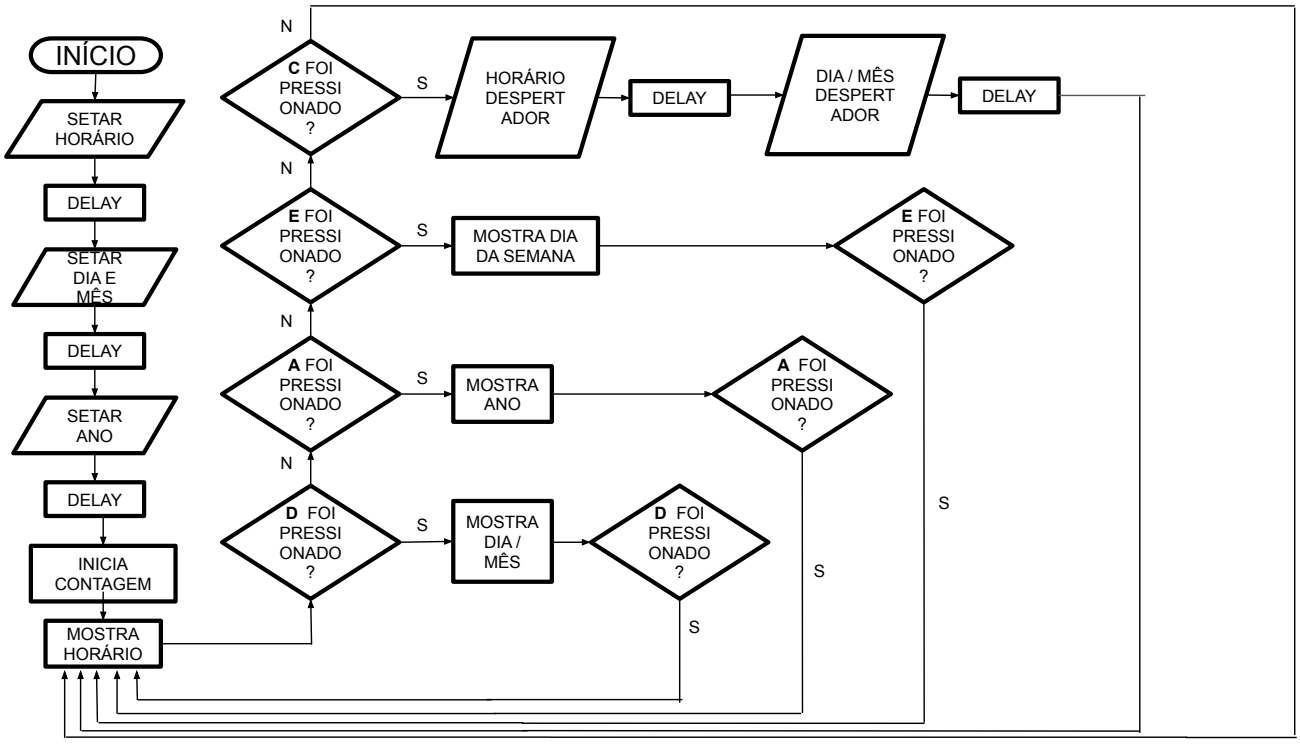

Figura 1: Fluxograma da lógica do programa

2 Materiais, métodos e projeto

Os equipamentos e softwares utilizados para este projeto estão descritos nesta seção.

2.1 Kit EXSTO XM 852

O kit EXSTO XM 852 é um kit educacional desenvolvido pela Exsto Tecnologias para disciplinas de arquitetura de computadores e tem como principal foco o uso da arquitetura 8051 em modo processador. O kit possui memórias externas EEPROM e RAM e diversas aplicações em portais mapeados em memória, além de alguns recursos conectados diretamente aos terminais do microcontrolador, como aplicações temos conversor A/D e D/A, motor de passo, display de 7 segmentos e display LCD.

O microcontrolador 8051 teve sua produção iniciada pela Intel em 1981 e atualmente é um dos mais populares no mercado, possuindo forte apelo didático devido à sua estrutura interna de fácil compreensão. Dentre suas principais características pode-se citar: frequência de clock de 12 MHz, dois temporizadores/ contadores de 16 bits, um canal de comunicação serial, cinco fontes de interrupção (dois timers, dois pinos externos e o canal de comunicação serial) e um oscilador de clock interno.

2.2 MIDE 51

MIDE-51 é um ambiente de desenvolvimento integrado (IDE) para o microcontrolador MCS-51. Seu pacote completo inclui o compilador Assembler, o Compilador C para Dispositivos Pequenos (SDCC), o Emulador TS Controls 8051 e o Simulador JSIM-51.

É por meio desta IDE que os programas que serão gravados na placa do kit são escritos, compilados e construídos (operação "build", responsável por gerar as extensões de arquivos que serão gravadas na memória do microcontrolador). Utiliza linguagem C.

2.3 SDCC

O Compilador C para Dispositivos Pequenos (Small Device C Compiler ou SDCC) é uma suíte de compiladores padronizada em C, reprogramável e otimizada direcionada para os processadores baseados no MCS-51 da Intel, originalmente desenvolvida pela *Sandeep Dutta*. Dentre os recursos oferecidos pelo

compilador SDCC estão: uma gama completa de tipos de variáveis (*bool*,*int*,*char*,*short*,*long*,*long long* e *float*), um conjunto de otimizações padrão (eliminação de sub-expressões globais, loop invariante, eliminação de código "morto", entre outras) e testes de regressão automatizados.

2.4 Gravador XM85X

O software XM85X é responsável pela gravação do arquivo construído pelo compilador na memória no microcontrolador, para a sua execução no kit. Para tanto, o arquivo com extensão .hex gerado pelo compilador deve ser carregado no gravador,deve-se configurar a porta do computador que será conectada ao kit, estabelecer a conexão entre ambos, pressionar o botão reset para apagar o programa que estiver em execução no dispositivo e então gravar o programa desejado.

3 Código e Lógica Desenvolvidas

O projeto desenvolvido utiliza do Kit EXSTO XM852 os seus Temporizadores, Módulo de 4 *Displays* de 7 Segmentos, Teclado Matricial de 16 Teclas e os LEDS.

As configurações do teclado matricial e do *display* 7S determinam o funcionamento do programa, já que precisam ser atualizados a todo o instante, enquanto processos paralelos ocorrem para atualizar a lógica do alarme.

O código do programa se divide em treze funções - a saber: *main*, *add7s*, *delayMicroseconds*, *timerMicroseconds*, *varreduraD7S*, *TM2HEX*, *varreduraTecladoMat*, *atualizaHora*, *delayTime*, *timerMiliseconds*, *configurations*, *varredura* e *resetTM* -, três *structs* (*timer*, *dataHora* e *tecladoMatricial*) e uma enumeração (*control*), que serão explicados adiante.

3.1 Configuração do *Display* de 7 Segmentos

O *Display* 7S utiliza uma ativação através do decodificador de endereços do KIT, para isso são utilizados dois endereços. Um denominado de **sel7s** para atribuir as configurações necessárias para qual *display* do módulo será acesso e outro denominado **dado7s**, que envia os dados de quais dos 7 LEDS do *display* devem ser acesos. Essas configurações podem ser analisadas abaixo:

```
1 #define dado7s 0xFFC0 // Ativação do decodificador de endereços (F - A15 até A12, F - A11 até A8, C - A7 e A6, 0 - é o
   ↪  endereço CS_DISP7S)
2 #define sel7s 0xFFC1  // Ativação do decodificador de endereços (F - A15 até A12, F - A11 até A8, C - A7 e A6, 1 - é o
   ↪  endereço CS_DISP7S_SELECT)
3
4 static unsigned char __far __at dado7s dado;                 //  atribui valor ao endereço dado7s
5 static unsigned char __far __at sel7s sel;                        //  seleciona o chip ao endereço sel7s
```

Após configurado, as variáveis **dado** e **sel** recebem os valores necessários para atribuir a cada *display* o respectivo valor desejado. Uma função de *varreduraD7S()* foi configurada para realizar essa atribuição, como analisado em:

```
1 /* FUNÇÃO varreduraD7S:
2 *                Função que realiza a varredura do DISPLAY de 7 Segmentos, recebe como parâmetro:
3 *                        D1   -> Digito/letra do primeiro display
4 *                        D2   -> Digito/letra do segundo display
```

```
*                       D3   -> Digito/letra do terceiro display
*                       D4          -> Digito/letra do quarto display
*                       time -> Tempo de varredura em microsegundos
*                       dot  -> Se o segundo display irá ter ponto ou não
*/
void varreduraD7S(unsigned char D1, unsigned char D2, unsigned char D3, unsigned char D4, unsigned int time, unsigned
 ↪   char dot){
        sel = 1;            // seleciona o display 1 -> Mais a esquerda (possíveis valores: 1, 2, 4, 8)
        dado = add7s(D1,0);                 // atribuí o endereço hex ao dado7s
        delayMicroseconds(time);        // Espera um tempo para mostrar na tela

        sel = 2;                                        // seleciona o display 2
        dado = add7s(D2,dot);                 // atribuí o endereço hex ao dado7s
        delayMicroseconds(time);        // Espera um tempo para mostrar na tela

        sel = 4;                                        // seleciona o display 3
        dado = add7s(D3,0);                 // atribuí o endereço hex ao dado7s
        delayMicroseconds(time);        // Espera um tempo para mostrar na tela

        sel = 8;                                        // seleciona o display 4
        dado = add7s(D4,0);                 // atribuí o endereço hex ao dado7s
        delayMicroseconds(time);        // Espera um tempo para mostrar na tela
}
```

Assim essa função atribui para cada *display* um respectivo valor, aguardando um tempo de $125\mu s$ para mostrá-lo e atualizá-lo. Assim se $\boldsymbol{sel} = 1$ é selecionado o *display* mais a esquerda do KIT, enquanto que o $\boldsymbol{sel} = 8$ é o *display* mais a direita.

O parâmetro **dado** é atualizado através de uma função denominada *add7s* que retorna o endereço necessário para a ativação dos LEDS, a fim de mostrar o digito/letra selecionado. A tabela de conversão pode ser vista na Figura 2.

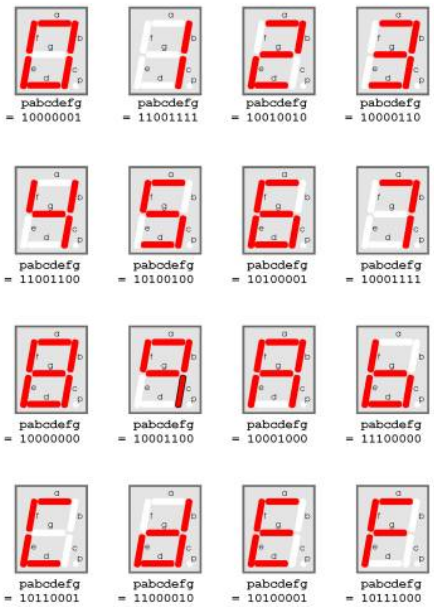

Figura 2: Configurações de Endereços para Display de 7 Segmentos de (Fronteira TEC, 2013).

Outras letras foram configuradas também, como o G, T, R, U, X e O, além do ponto.

3.2 Configuração do Teclado Matricial 4x4

O Teclado Matricial (TM) utiliza uma ativação através do decodificador de endereços do KIT, para isso é utilizado somente um endereço. Um denominado de **selTM** para atribuir as configurações necessárias para selecionar a linha dos. Essas configurações podem ser analisadas abaixo:

```
1  #define selTM 0xFFC3  // Ativação do decodificador de endereços para teclado matricial (F - A15 até A12, F - A11 até
   ↪  A8, C - A7 e A6, 3 - é o endereço CS_TECLADO)
2
3  static unsigned char __far __at selTM selTeclado;         //  seleciona o chip ao endereço selTM
```

Da mesma forma que ocorre a varredura para o *display* ocorre para o teclado matricial. Porém, o teclado é subdividido em 2 partes, as primeiras duas linhas contendo 1, 2, 3, A, 4, 5, 6 e B e as duas últimas linhas, contendo 7, 8, 9, C, F, 0, E e D. A seguinte ilustração representa a configuração do teclado:

```
/*
*                TECLADO                      VALOR CORRESPONDENTE
*                |---------------|            |----------------------|
*                | 1 | 2 | 3 | A |            |  1  |  2  |  4  |  8  |
*                |---------------|            |----------------------|
*                | 4 | 5 | 6 | B |            | 16  | 32  | 64  | 128 |
*                |---------------|            |----------------------|
*                | 7 | 8 | 9 | C |            |  1  |  2  |  4  |  8  |
*                |---------------|            |----------------------|
*                | F | 0 | E | D |            | 16  | 32  | 64  | 128 |
*                |---------------|            |----------------------|
*/
```

Para diferenciar os valores correspondentes das primeiras duas linhas com as duas últimas linhas, soma-se 128 nas duas últimas linhas, reconhecendo o valor digitado.

A função de varredura do teclado matricial é dado pelo código abaixo. Neste as duas primeiras linhas do teclado são selecionados quando $\boldsymbol{selTeclado} = 0x40$, e as duas últimas são selecionadas quando $\boldsymbol{selTeclado} = 0x80$.

Quando algum dos botões é pressionado o valor será atribuído para os *display*, verificado e executado o controle de estados do alarme.

```
void varreduraTecladoMat(){
        unsigned int dadoTM = 0, dadoTM1, dadoTM2;
        unsigned char boolHour1, boolHour2, boolMin1, boolMin2, boolControl;

        selTeclado = 0x40; // Habilitando as duas primeiras linhas do TM
        dadoTM1 = selTeclado; // Lendo o valor pressionado

        selTeclado = 0x80; // Habilitando as duas ultimas linhas do TM
        dadoTM2 = selTeclado; // Lendo o valor pressionado

        if(dadoTM1 != 0 || dadoTM2 != 0){        // Se estou apertando algum botão
                if (dadoTM1 !=0) // Se estou apertando as primeiras linhas
                        dadoTM = dadoTM1; // Atribuí o valor pressionado ao dadoTM
                else if(dadoTM2 != 0) // Se estou apertando as duas últimas linhas
                        dadoTM = dadoTM2 + 128;        // Atribuí o valor pressionado ao dadoTM

                TM.value[TM.vectorControl] = TM2HEX(dadoTM); // Coloca no vetor do Display o valor correspondente ao
                 ↪  pressionado.
                do{
                        selTeclado = 0x40;
                        dadoTM1 = selTeclado;
```

```
                        selTeclado = 0x80;
                        dadoTM2 = selTeclado;
                } while((dadoTM1 == dadoTM) || (dadoTM2 +128 == dadoTM ));// Espera até soltar o botão (DEBOUCER)

                /*
                * PARTE OCULTA: VERIFICAÇÃO DO DIGITOS DE HORÁRIO
                */

                if( !(boolHour1 || boolHour2 || boolMin1 || boolMin2 || boolControl) )
                        TM.value[TM.vectorControl] = 16;
                else{
                    /*
                    * PARTE OCULTA: FUNÇÃO PARA CONTROLE DOS ESTADOS DO ALARME
                    */
                }
        }
}
```

Os estados do alarme são controlados para uma enumeração definida no começo do código. Sendo a seguinte:

```
/* Control foi definida como flags do projeto. Conforme as etapas do projeto vão evoluindo control atualiza suas flags
 ↪  para perceber em qual parte *do programa se encontra. Se a variável for atribuída como:
* hour1      -> Ativa a atribuição da dezena da hora
* hour2      -> Ativa a atribuição da unidade da hora
* min1       -> Ativa a atribuição da dezena do minuto
* min2       -> Ativa a atribuição da unidade do minuto
* reset1     -> Ativa o delay para mostrar as os números digitados de hora e minuto
* day1       -> Ativa a atribuição da dezena do dia
* day2       -> Ativa a atribuição da unidade do dia
* month1     -> Ativa a atribuição da dezena do mês
* month2     -> Ativa a atribuição da unidade do mês
* reset2     -> Ativa o delay para mostrar as os números digitados da data
* year1      -> Ativa a atribuição do milhar do ano
* year2      -> Ativa a atribuição da centena do ano
* year3      -> Ativa a atribuição da dezena do ano
* year4      -> Ativa a atribuição da unidade do ano
* reset3     -> Ativa o delay para mostrar as os números digitados do ano
* clear      -> Ativa a condição padrão do relório, mostrar o horário
* showDay    -> Mostra o dia que foi configurado.
* showYear   -> Mostra o ano que foi configurado.
* showWeek   -> Mostra o dia da semana (SEG,TER,QUA,QUI,SEX,SAB,DOM)
* configHour -> Habilita a configuração do alarme para hora
* configDay  -> Habilita a configuração do alarme para dia e mês
*/
enum control
 ↪  {hour1,hour2,min1,min2,reset1,day1,day2,month1,month2,reset2,year1,year2,year3,year4,reset3,clear,showDay,showYear,showWeek
 ↪  configHour, configDay};
```

3.3 Temporizadores

A temporização do programa foi implementada por meio dos dois timers do microcontrolador (timer 0 e timer 1).

O timer 0 está associado à base de tempo do relógio e é controlado pelas funções *delayMiliseconds* e *timerDelay*. A função *timerDelay* é chamada na função *Main*, calcula a quantidade de ciclos do timer e a quantidade de contagens do último ciclo correspondentes ao tempo passado como parâmetro pela função *Main*, que são armazenados nas flags *Timer0.cycles* e *Timer0.lastClock* respectivamente, e inicializa o timer 0. Quando ocorre a interrupção, a função *delayMiliseconds* é responsável por reinicializar o timer e implementar a lógica responsável por verificar se o tempo decorrido desde sua inicialização é igual ao tempo recebido como parâmetro pela função *timerDelay*.

O timer 1 está associado ao controle do tempo de varredura dos displays e é controlado pelas funções *delayMicroseconds* e *timerMicroseconds*. A função *delayMicroseconds* é chamada na função que realiza a varredura do display (*varreduraD7S*), calcula a quantidade de contagens correspondente ao tempo recebido como entrada (com limite de 65535 contagens) e configura e inicializa o timer 1. Diferente da função *delayMiliseconds*, esta função espera que o timer tenha finalizado para retornar à função *Main*, o que é controlado pela flag *Timer1.finish*. Quando ocorre a interrupção, o programa entra na função *timerMicroseconds* que reinicializa o timer.

3.4 Função Main

Ao iniciar a função *main*, é chamada a função *configurations*, responsável por configurar os timers e interrupções e a prioridade de interrupções, inicializar os timers, habilitar o modo de contagem de 16 bits, zerar os atributos da *struct dh*, zerar a variável de controle *control* e setar como nulo o valor mostrado pelos displays (para este projeto, o valor nulo no display é representado pela exibição de um ponto apenas).

Os valores são entrados por meio do teclado matricial seguindo a seguinte sequência: deve-se digitar o horário (hh.mm), então a data (dd.MM) e depois o ano (aaaa). Após o display voltar a mostrar o horário, os botões D, A e E, quando pressionados uma única vez, serão responsáveis por exibir: o dia (D), o ano (A) e o dia da semana (E), voltando a mostrar o horário caso pressionados novamente.

O programa contem um loop infinito, que será responsável por controlar a inserção de entradas no relógio e no alarme e o término dos delays. Cada etapa de inserção (isto é, a cada quatro dígitos inseridos) envolve um delay representado pelas flags *reset1*, *reset2* e *reset3*, associadas respectivamente ao horário, data e ano. Primeiramente, verifica-se se o delay associado a cada uma das etapas de inserção das entradas já terminou e, em caso afirmativo, habilita a etapa seguinte. Quando todos os valores tiverem sido entrados pelo usuário, estes são validados pela função *verificaData* e, caso sejam inválidos, a data, o horário e o ano são resetados por meio das flags *TM.control* e *TM.vectorControl* para que o usuário possa redigitar a data.

O horário é mostrado por default enquanto o dia, o ano e o dia da semana são mostrados ao se pressionar uma única vez respectivamente os botões D, A e E do teclado matricial, associados respectivamente às flags *showDay*, *showYear* e *showWeek*, voltando a mostrar o horário se novamente pressionados. Os valores entrados são passados para o display por meio da função varredura, que recebe os valores (em decimal) dos dígitos que devem aparecer no display, o tempo de atualização dos displays e a exibição ou não do ponto. A função *delayTime* é chamada e executada a cada loop, servindo como base de tempo para os segundos.

Ao se pressionar o botão C, habilita-se a configuração do horário e data de disparo do alarme pelo usuário, voltando a mostrar o horário do relógio após a inserção do horário e da data pelo usuário. Finalmente, caso o horário do relógio se iguale ao horário entrado pelo usuário, o alarme é disparado, fazendo acender os LEDS pelo tempo em que o horário do relógio for igual ao horário configurado no despertador.

4 Conclusão

O projeto do alarme digital foi concluído para a entrega final, de forma que todas as funções exigidas funcionaram corretamente, e ainda foram acrescentadas as funções de validação de data/horário e de indicação de dia da semana, que funcionaram conforme o esperado.

Um dos maiores problemas verificados durante o desenvolvimento do projeto foi determinar a inicialização da função de *timer* uma única vez até que sua temporização fosse finalizada, sendo necessária a inclusão de uma *flag* na *struct* do *timer*.

Como trabalhos futuros, vê a necessidade da otimização da função de *delayTime()* para contagem correta do tempo, devendo ocorrer em um processo paralelo no sistema. Com o código atual, o tempo contado pelo temporizador tem um erro de alguns segundos em comparação com o computador, devido ao tempo computacional gasto pelo microcontrolador para processar as linhas de código dentro do *while* da função *main*, assim como o *delay* de varredura dos *display* 7S.

Referências

Fronteira TEC. **Projeto Fliperama: Utilização do display 7 segmentos.** 2013. <https://fronteiratec.com/blog/projeto-fliperama-utilizacao-do-display-7-segmentos/>. Accessed: 2019-07-14.

5 Apêndice A - Códigos

```
#include<8051.h>

/*
=============================================================================================
====================================PROJETO 1 - ALARME DIGITAL=====================================
=============================================================================================
Nomes:
        Gilmar Correia Jeronimo         - 11014515
        Lucas Barboza Moreira Pinheiro  - 11017015
*/

#define dado7s 0xFFC0 // Ativação do decodificador de endereços (F - A15 até A12, F - A11 até A8, C - A7 e A6, 0 - é o endereço CS_DISP7S)
#define sel7s 0xFFC1  // Ativação do decodificador de endereços (F - A15 até A12, F - A11 até A8, C - A7 e A6, 1 - é o endereço CS_DISP7S_SELECT)
#define selTM 0xFFC3  // Ativação do decodificador de endereços para teclado matricial (F - A15 até A12, F - A11 até A8, C - A7 e A6, 3 - é o endereço
 ↪    CS_TECLADO)

static unsigned char __far __at dado7s dado;                //  atribui valor ao endereço dado7s
static unsigned char __far __at sel7s sel;                        //  seleciona o chip ao endereço sel7s
static unsigned char __far __at selTM selTeclado;         //  seleciona o chip ao endereço selTM


/* Control foi definida como flags do projeto. Conforme as etapas do projeto vão evoluindo control atualiza suas flags para perceber em qual parte *do
 ↪    programa se encontra. Se a variável for atribuída como:
* hour1      -> Ativa a atribuição da dezena da hora
* hour2      -> Ativa a atribuição da unidade da hora
* min1       -> Ativa a atribuição da dezena do minuto
* min2       -> Ativa a atribuição da unidade do minuto
* reset1     -> Ativa o delay para mostrar as os números digitados de hora e minuto
* day1       -> Ativa a atribuição da dezena do dia
* day2       -> Ativa a atribuição da unidade do dia
* month1     -> Ativa a atribuição da dezena do mês
* month2     -> Ativa a atribuição da unidade do mês
* reset2     -> Ativa o delay para mostrar as os números digitados da data
* year1      -> Ativa a atribuição do milhar do ano
* year2      -> Ativa a atribuição da centena do ano
* year3      -> Ativa a atribuição da dezena do ano
* year4      -> Ativa a atribuição da unidade do ano
* reset3     -> Ativa o delay para mostrar as os números digitados do ano
* clear      -> Ativa a condição padrão do relório, mostrar o horário
* showDay    -> Mostra o dia que foi configurado.
* showYear   -> Mostra o ano que foi configurado.
* showWeek   -> Mostra o dia da semana (SEG,TER,QUA,QUI,SEX,SAB,DOM)
* configHour -> Habilita a configuração do alarme para hora
* configDay  -> Habilita a configuração do alarme para dia e mês
*/
enum control
 ↪    {hour1,hour2,min1,min2,reset1,day1,day2,month1,month2,reset2,year1,year2,year3,year4,reset3,clear,showDay,showYear,showWeek,configHour,configDay};
//enum control
 ↪    {min1,min2,hour1,hour2,reset1,day1,day2,month1,month2,reset2,year1,year2,year3,year4,reset3,clear,showDay,showYear,showWeek,configHour,configDay};


/* A struct timer guarda as variáveis relacionada aos timers usados
*        TIMER0 -> Delay Milisseconds
*        TIMER1 -> Delay Microsseconds
*
*         Como a memória do timer interno dá overflow em 65535 e cada contagem acontece em 1 microssegundo, o timer dá um overflow a cada 65.535 milisegundos,
 ↪    portanto:
*        Cicles    = (Milisseconds * 1000)/ 65535 (DIVISÃO INTEIRA)
*         lastClock = (Milisseconds * 1000)% 65535 (RESTO DA DIVISÃO)
*   finish   -> flag para informar que a contagem acabou
*   flag     -> Controla se o timer está temporizando ou não
*/
struct timer{
```

```
59          unsigned int cycles;
60          unsigned int lastClock;
61          unsigned char finish;
62          unsigned char flag;
63  };
64
65
66  /* A struct dataHora guarda as variáveis relacionada as configurações iniciais do alarme
67  *        dia    -> Guarda os valores de dia (dezena + unidade)
68  *        mes    -> Guarda os valores do mes (dezena + unidade)
69  *        ano    -> Guarda os valores do ano (milhar + centena + dezena + unidade)
70  *        hora   -> Guarda os valores de hora (dezena + unidade)
71  *        minuto -> Guarda os valores de minuto (dezena + unidade)
72  */
73  struct dataHora{
74          unsigned char dia;
75          unsigned char mes;
76          unsigned int ano;
77          unsigned char hora;
78          unsigned char min;
79  };
80
81
82  /* A struct tecladoMatricial guarda as variáveis relacionada as configurações do teclado matricial
83  *        control       -> Variável que utilizará como parâmetro os ENUM CONTROL
84  *   value[4]      -> Cada posição do vetor value guarda os parâmetros atribuídos ao DISPLAY7S correspondente, sendo a posição 0 relativa ao display da mais a
     ↪    esquerda e a posição 4 relativa ao display mais a direita.
85  *        vectorControl -> Controla a qual display será atribuído o valor pressionado.
86  */
87  struct tecladoMatricial{
88          unsigned char control;
89          unsigned char value[4];
90          unsigned char vectorControl;
91  };
92
93  struct tecladoMatricial TM;                // Atribuição de um tecladoMatricial nomeado de TM.
94
95  struct dataHora dh;                        // Atribuição de uma dataHora nomeada de dh.
96  struct dataHora alarm;                        // Atribuição de uma dataHora nomeada de alarm.
97
98  struct timer Timer0;                         // Atribuição de um timer nomeado de Timer0.
99  struct timer Timer1;                         // Atribuição de um timer nomeado de Timer0.
100
101
102 /*
103 * FUNÇÃO RELACIONADAS AO DISPLAY7S
104 */
105
106 /* FUNÇÃO add7s:
107 *          - Função para retornar o endereço hexadecimal do display 7s correspondente ao decimal ou letra entrada
108 *                decimal -> Variável do número/letra desejado
109 *                dot         -> Se o número apresentará ponto ou não
110 *                se decimal = 16 só acenderá o ponto.
111 */
112 unsigned char add7s(unsigned char decimal, unsigned char dot){
113
114         char letter = decimal;
115
116         unsigned char address[16] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x67, 0x77, 0x7C, 0x39, 0x5E, 0x79, 0x71};
117
118         if (decimal <= 15)
119                 return address[decimal]+(dot*(128));
120         else if (decimal == 16)
121                 return 128; // Só acende o ponto
122         else if(letter == 'S')
123                 return 0x6D;
124         else if(letter == 'E')
```

```
                return 0x79;
        else if(letter == 'G')
                return 0x3D;
        else if(letter == 'T')
                return 0x78;
        else if(letter == 'R')
                return 0x50;
        else if(letter == 'Q')
                return 0x67;
        else if(letter == 'U')
                return 0x1C;
        else if(letter == 'A')
                return 0x77;
        else if(letter == 'I')
                return 0x06;
        else if(letter == 'X')
                return 0x76;
        else if(letter == 'B')
                return 0x7C;
        else if(letter == 'D')
                return 0x5E;
        else if(letter == 'O')
                return 0x5C;
        else if(letter == 'N')
                return 0x54;

        return 0;
}


/* FUNÇÃO delayMicroseconds:
*         Funcao para delay em microsegundos que recebe como parâmetro a quantidade de microsegundos desejados
*        OBS: Diferente do delayMiliseconds, esse delay ele ESPERA ATÉ TERMINAR A TEMPORIZAÇÃO, para desativar comentar as linhas com θ
*
*        Limite da função microseconds = 65535;
*/
void delayMicroseconds(unsigned int microseconds){

        Timer1.lastClock = 65535 - microseconds;         // Cálculo de quantos microsecundos o temporizador deve começar
        TR1 = 1;                                        // Inicializar o timer

        while(!Timer1.finish);                                  //θ Espera a flag retornar o valor desejado

        Timer1.finish = 0;                                    //θ        Zera a flag

        // SE EU QUISER CONTROLAR O RELÓGIO COM MICROSSEGUNDOS DESCOMENTAR ESSA LINHA ABAIXO:
        /*
        if(TM.control>=clear)
                atualizaHora();
        */
}


/* FUNÇÃO timerMicroseconds:
*         Função que controla a interrupção do TIMER1
*/
void timerMicroseconds() __interrupt 3{ //(slide aplicmicro 11_C.pdf pg.13)

        TH1 = 0;                                  // Zerar os bits mais significativos do temporizador
        TL1 = 0;                                  // Zerar os bits menos significativos do temporizador
        TF1 = 0;                                 // Zero a flag do contador
        TR1 = 0;                                 // Paro timer1

        TH1 = Timer1.lastClock & 0xFF ;                // Atribui ao ultimo timer o valor dos bits mais significativos
        TL1 = Timer1.lastClock >> 8;                // Atribui ao ultimo timer o valor dos bits menos significativos
    //Timer1.cycles--;                                  // decrementa cycles

```

```
192         Timer1.finish = 1;                      // finish é uma flag que indica que o tempo acabou
193 }
194
195
196 /* FUNÇÃO varreduraD7S:
197 *        Função que realiza a varredura do DISPLAY de 7 Segmentos, recebe como parâmetro:
198 *                D1   -> Digito/letra do primeiro display
199 *                D2   -> Digito/letra do segundo display
200 *                D3   -> Digito/letra do terceiro display
201 *                D4          -> Digito/letra do quarto display
202 *                time -> Tempo de varredura em microsegundos
203 *                dot  -> Se o segundo display irá ter ponto ou não
204 */
205 void varreduraD7S(unsigned char D1, unsigned char D2, unsigned char D3, unsigned char D4, unsigned int time, unsigned char dot){
206         sel = 1;                         // seleciona o display 1 -> Mais a esquerda (possíveis valores: 1, 2, 4, 8)
207         dado = add7s(D1,0);                 // atribuí o endereço hex ao dado7s
208         delayMicroseconds(time);        // Espera um tempo para mostrar na tela
209
210         sel = 2;                        // seleciona o display 2
211         dado = add7s(D2,dot);                 // atribuí o endereço hex ao dado7s
212         delayMicroseconds(time);        // Espera um tempo para mostrar na tela
213
214         sel = 4;                        // seleciona o display 3
215         dado = add7s(D3,0);                 // atribuí o endereço hex ao dado7s
216         delayMicroseconds(time);        // Espera um tempo para mostrar na tela
217
218         sel = 8;                        // seleciona o display 4
219         dado = add7s(D4,0);                 // atribuí o endereço hex ao dado7s
220         delayMicroseconds(time);        // Espera um tempo para mostrar na tela
221 }
222
223 /* FUNÇÃO digitosLetter7S:
224 *        Função que retorna os digitos que devem ser colocados na tela dado um dia da semana, se o parâmetro for:
225 *                0 -> Retorna as letras relacionadas a SEGUNDA
226 *                1 -> Retorna as letras relacionadas a TERCA
227 *                2 -> Retorna as letras relacionadas a QUARTA
228 *                3 -> Retorna as letras relacionadas a QUINTA
229 *                4 -> Retorna as letras relacionadas a SEXTA
230 *                5 -> Retorna as letras relacionadas a SABADO
231 *                6 -> Retorna as letras relacionadas a DOMINGO
232 */
233 unsigned char * digitosLetter7S(unsigned char weekDay){
234         unsigned char Digits[4] = {0,0,0,0};
235
236         if(weekDay == 0){
237                 Digits[0] = 'S';
238                 Digits[1] = 'E';
239                 Digits[2] = 'G';
240         }
241         else if(weekDay == 1){
242                 Digits[0] = 'T';
243                 Digits[1] = 'E';
244                 Digits[2] = 'R';
245         }
246         else if(weekDay == 2){
247                 Digits[0] = 'Q';
248                 Digits[1] = 'U';
249                 Digits[2] = 'A';
250         }
251         else if(weekDay == 3){
252                 Digits[0] = 'Q';
253                 Digits[1] = 'U';
254                 Digits[2] = 'I';
255         }
256         else if(weekDay == 4){
257                 Digits[0] = 'S';
258                 Digits[1] = 'E';
```

```
259                 Digits[2] = 'X';
260         }
261         else if(weekDay == 5){
262                 Digits[0] = 'S';
263                 Digits[1] = 'A';
264                 Digits[2] = 'B';
265         }
266         else if(weekDay == 6){
267                 Digits[0] = 'D';
268                 Digits[1] = 'O';
269                 Digits[2] = 'N';
270                 Digits[3] = 'N';
271         }
272
273         return Digits;
274 }
275 /*
276 * FUNÇÕES RELACIONADAS AO DISPLAY7S
277 */
278
279 /*
280 * FUNÇÕES RELACIONADAS AO TECLADO MATRICIAL
281 */
282
283 /* FUNÇÃO TM2HEX:
284 *                  Função usado para retornar qual o correspondente valor NOMINAL foi apertado do teclado matricial, recebe como parâmetro:
285 *                        value -> Valor dado ao pressionar o teclado matricial (TM).
286 *
287 *                TECLADO                         VALOR CORRESPONDENTE DADO PELO DISPOSITIVO A PRESSIONAR A TECLA CORRESPONDENTE
288 *                |---------------|       |-----------------------|
289 *                | 1 | 2 | 3 | A |       |  1  |  2  |  4  |  8  |
290 *                |---------------|       |-----------------------|
291 *                | 4 | 5 | 6 | B |        | 16  | 32  | 64  | 128 |
292 *                |---------------|       |-----------------------|
293 *                | 7 | 8 | 9 | C |        |  1  |  2  |  4  |  8  | -> PARA DIFERENCIAR AS PRIMEIRA 2 LINHAS DAS SEGUNDAS SOMA-SE 128 QUANDO FEITA A VARREDURA
294 *                |---------------|       |-----------------------|
295 *                | F | 0 | E | D |        | 16  | 32  | 64  | 128 | -> PARA DIFERENCIAR A PRIMEIRAS 2 LINHAS DAS SEGUNDAS SOMA-SE 128 QUANDO FEITA A VARREDURA
296 *                |---------------|       |-----------------------|
297 */
298 unsigned char TM2HEX(unsigned int value){
299
300         switch(value){
301                 case 1:                                         // Se o resultado for 1 do TM
302                         return 1;                       // retorna 1
303                 case 2:
304                         return 2;                       // retorna 2
305                 case 4:
306                         return 3;                       // retorna 3
307                 case 8:
308                         return 10;                       // retorna A
309                 case 16:
310                         return 4;                       // retorna 4
311                 case 32:
312                         return 5;                       // retorna 5
313                 case 64:
314                         return 6;                       // retorna 6
315                 case 128:
316                         return 11;                       // retorna B
317                 case 129:
318                         return 7;                       // retorna 7
319                 case 130:
320                         return 8;                       // retorna 8
321                 case 132:
322                         return 9;                       // retorna 9
323                 case 136:
324                         return 12;                       // retorna C
325                 case 144:
```

```
                return 15;                      // retorna F
        case 160:
                return 0;                      // retorna 0
        case 192:
                return 14;                      // retorna E
        case 256:
                return 13;                      // retorna D
    }

    return 0;
}

/* FUNÇÃO verreduraTecladoMat:
*       Função que executa a varredura do teclado matricial e controla em qual parte o programa se encontra, atribuindo os valores de acordo com o ENUM CONTROL.
*/
void varreduraTecladoMat(){
    unsigned int dadoTM = 0, dadoTM1, dadoTM2;
    unsigned char boolHour1, boolHour2, boolMin1, boolMin2, boolControl;

    selTeclado = 0x40;                                                  // Habilitando as duas primeiras linhas do TM
    dadoTM1 = selTeclado;                                                  // Lendo o valor pressionado

    selTeclado = 0x80;                                                  // Habilitando as duas ultimas linhas do TM
    dadoTM2 = selTeclado;                                                  // Lendo o valor pressionado


    if(dadoTM1 != 0 || dadoTM2 != 0){                                   // Se estou apertando algum botão

        if (dadoTM1 !=0)                                               // Se estou apertando as primeiras linhas
            dadoTM = dadoTM1;                                           // Atribuí o valor pressionado ao dadoTM
        else if(dadoTM2 !=0)                                              // Se estou apertando as duas últimas linhas
            dadoTM = dadoTM2 + 128;                                        // Atribuí o valor pressionado ao dadoTM


        TM.value[TM.vectorControl] = TM2HEX(dadoTM);                          // Coloca no vetor do Display o valor correspondente ao pressionado.

        do{
            selTeclado = 0x40;
            dadoTM1 = selTeclado;
            selTeclado = 0x80;
            dadoTM2 = selTeclado;
        } while((dadoTM1 == dadoTM) || (dadoTM2 +128 == dadoTM ));          // Espera até soltar o botão (DEBOUCER)


        boolHour1 = ((TM.control == hour1 || (TM.control == configHour && TM.vectorControl == 0 )) && TM2HEX(dadoTM) <=2); // Verifica se o digito da dezena da hora que está sendo pressionado é o correto

        boolHour2 = ((TM.control == hour2 || (TM.control == configHour && TM.vectorControl == 1 )) && ((TM2HEX(dadoTM) <=9 && TM.value[0] <= 1) || (TM2HEX(dadoTM) <=3 && TM.value[0] == 2)) ); // Verifica se o digito da unidade da hora que está sendo pressionado é o correto

        boolMin1 = ((TM.control == min1 || (TM.control == configHour && TM.vectorControl == 2 )) && TM2HEX(dadoTM) <=5); // Verifica se o digito da dezena do minuto que está sendo pressionado é o correto

        boolMin2 = ((TM.control == min2 || (TM.control == configHour && TM.vectorControl ==3 )) && TM2HEX(dadoTM) <=9); // Verifica se o digito da unidade do minuto que está sendo pressionado é o correto

        boolControl = ((TM.control> min2 && TM.control<configHour) || TM.control == configDay); // Como apresenta duas vezes que se configura o horário, no alarme e no início do display, a boolControl atribuí as correções para esses momentos, para os outros, deixa apenas configurar o dia e o ano.

        if( !(boolHour1 || boolHour2 || boolMin1 || boolMin2 || boolControl) )
            TM.value[TM.vectorControl] = 16;
        else{

            if(TM.control == hour1)
```

```
385                         dh.hora = 10*TM.value[TM.vectorControl];                    // Atribuí o valor para a dezena da hora de
                             ↪  dh
386                     else if(TM.control == hour2)
387                         dh.hora += TM.value[TM.vectorControl];                      // Atribuí o valor para a unidade da
                             ↪  hora de dh
388                     else if(TM.control == min1)
389                         dh.min = 10*TM.value[TM.vectorControl];                     // Atribuí o valor para a dezena do
                             ↪  minuto de dh
390                     else if(TM.control == min2)
391                         dh.min += TM.value[TM.vectorControl];                       // Atribuí o valor para a unidade do
                             ↪  minuto de dh
392
393                     else if(TM.control == day1)
394                         dh.dia = 10*TM.value[TM.vectorControl];                     // Atribuí o valor para a dezena do dia
                             ↪  de dh
395                     else if(TM.control == day2)
396                         dh.dia += TM.value[TM.vectorControl];                       // Atribuí o valor para a unidade do dia
                             ↪  de dh
397                     else if(TM.control == month1)
398                         dh.mes = 10*TM.value[TM.vectorControl];                     // Atribuí o valor para a dezena do mes
                             ↪  de dh
399                     else if(TM.control == month2)
400                         dh.mes += TM.value[TM.vectorControl];                       // Atribuí o valor para a unidade do mes
                             ↪  de dh
401
402                     else if(TM.control == year1)
403                         dh.ano = 1000*TM.value[TM.vectorControl];                   // Atribuí o valor para o milhar do ano de dh
404                     else if(TM.control == year2)
405                         dh.ano += 100*TM.value[TM.vectorControl];                   // Atribuí o valor para a centena do ano de
                             ↪  dh
406                     else if(TM.control == year3)
407                         dh.ano += 10*TM.value[TM.vectorControl];                    // Atribuí o valor para a dezena do ano de dh
408                     else if(TM.control == year4){
409                         dh.ano += TM.value[TM.vectorControl];                       // Atribuí o valor para a unidade do ano
                             ↪  de dh
410                         TM.vectorControl = 0;
411                     }
412
413                     else if(TM.control == configHour && TM.vectorControl == 0){     // Se estiver no modo de configuração de
                         ↪  hora
414                         alarm.hora = 10*TM.value[TM.vectorControl];                 // Atribuí o valor para a dezena da hora do
                             ↪  alarme
415                         TM.vectorControl++;
416                     }
417                     else if(TM.control == configHour && TM.vectorControl == 1){
418                         alarm.hora += TM.value[TM.vectorControl];                   // Atribuí o valor para a unidade da hora do
                             ↪  alarme
419                         TM.vectorControl++;
420                     }
421                     else if(TM.control == configHour && TM.vectorControl == 2){
422                         alarm.min = 10*TM.value[TM.vectorControl];                  // Atribuí o valor para a dezena do minuto
                             ↪  do alarme
423                         TM.vectorControl++;
424                     }
425                     else if(TM.control == configHour && TM.vectorControl == 3){
426                         alarm.min += TM.value[TM.vectorControl];                    // Atribuí o valor para a unidade do minuto do
                             ↪  alarme
427                         TM.control = configDay;                                     // Atribuí a configuração do dia
428                         TM.value[0] = 16;                                           // Coloca ponto no display 1
429                         TM.value[1] = 16;                                           // Coloca ponto no display 2
430                         TM.value[2] = 16;                                           // Coloca ponto no display 3
431                         TM.value[3] = 16;                                           // Coloca ponto no display 4
432                         TM.vectorControl = 0;                                       // Reinicializa a variável
433                     }
434
435
436                     else if(TM.control == configDay && TM.vectorControl == 0){
```

```
437                             alarm.dia = 10*TM.value[TM.vectorControl];                                      // Atribuí o valor para a dezena do dia do
                                 ↪    alarme
438                             TM.vectorControl++;
439                         }
440                         else if(TM.control == configDay && TM.vectorControl == 1){
441                             alarm.dia += TM.value[TM.vectorControl];                                    // Atribuí o valor para a unidade do dia do
                                 ↪    alarme
442                             TM.vectorControl++;
443                         }
444                         else if(TM.control == configDay && TM.vectorControl == 2){
445                             alarm.mes = 10*TM.value[TM.vectorControl];                                      // Atribuí o valor para a dezena do mes do
                                 ↪    alarme
446                             TM.vectorControl++;
447                         }
448                         else if(TM.control == configDay && TM.vectorControl == 3){
449                             alarm.mes += TM.value[TM.vectorControl];                                    // Atribuí o valor para a unidade do mes do
                                 ↪    alarme
450                             TM.control = clear;                                                           // Volta para mostrar o horário
451                             TM.vectorControl = 0;                                                           // Reinicializa a variável
452                         }
453
454
455                         if(TM.control<clear){
456                             TM.vectorControl++;
457                             TM.control++;
458                         }
459                         else{
460                             if (TM.value[TM.vectorControl] == 13 && TM.control == clear)                // Se o valor digitado for 'D' e o botão não tiver
                                 ↪    sido posteriormente ativado
461                                 TM.control = showDay;                                                   // Ativa leitura da Data
462                             else if(TM.value[TM.vectorControl] == 13 && TM.control == showDay)          // Se o valor digitado for 'D' e a leitura da data
                                 ↪    estiver ativada
463                                 TM.control = clear;                                                                                                 //
                                     ↪    Desativa leitura da Data
464                             else if(TM.value[TM.vectorControl] == 10 && TM.control == clear)            // Se o valor digitado for 'A' e o botao não tiver
                                 ↪    sido posteriormente ativado
465                                 TM.control = showYear;                                                  // Ativa leitura do Ano
466                             else if(TM.value[TM.vectorControl] == 10 && TM.control == showYear)         // Se o valor digitado for 'A' e a leitura do Ano
                                 ↪    estiver ativada
467                                 TM.control = clear;                                                                                                 //
                                     ↪    Desativa leitura do Ano
468                             else if(TM.value[TM.vectorControl] == 14 && TM.control == clear)            // Se o valor digitado for 'E' e o botao não tiver
                                 ↪    sido posteriormente ativado
469                                 TM.control = showWeek;                                                  // Ativa leitura do dia da semana
470                             else if(TM.value[TM.vectorControl] == 14 && TM.control == showWeek)         // Se o valor digitado for 'E' e a leitura do Ano
                                 ↪    estiver ativada
471                                 TM.control = clear;                                                                                                 //
                                     ↪    Desativa leitura do dia da semana
472                             else if(TM.value[TM.vectorControl] == 12 && TM.control == clear){           // Se o valor digitado for 'C' e a leitura do horário
                                 ↪    para o alarme é ativado.
473                                 TM.control = configHour;                                                // Atribui a configuração do horário do alarme
474                                 TM.vectorControl = 0;                                                   // Reinicializa a variável
475                                 TM.value[0] = 16;                                                                                                   //
                                     ↪    Coloca ponto no display 1
476                                 TM.value[1] = 16;                                                                                                   //
                                     ↪    Coloca ponto no display 2
477                                 TM.value[2] = 16;                                                                                                   //
                                     ↪    Coloca ponto no display 3
478                                 TM.value[3] = 16;                                                                                                   //
                                     ↪    Coloca ponto no display 4
479                             }
480
481                         }
482                 }
483         }
484 }
485
```

```
/*
* FUNÇÕES RELACIONADAS AO TECLADO MATRICIAL
*/


/* FUNÇÃO getDiaDaSemana:
*                 Função que dado um DIA, MES e ANO, retorna qual o dia da semana sendo:
*                        0 -> SEGUNDA
*                        1 -> TERCA
*                        2 -> QUARTA
*                        3 -> QUINTA
*                        4 -> SEXTA
*                        5 -> SABADO
*                        6 -> DOMINGO
*                ALGORITMO BASEADO NO SITE:
*/
unsigned char getDiaDaSemana(unsigned char dia,unsigned char mes,unsigned char ano){

        unsigned char JND =
          dia
        + ((153 * (mes + 12 * ((14 - mes) / 12) - 3) + 2) / 5)
        + (365 * (ano + 4800 - ((14 - mes) / 12)))
        + ((ano + 4800 - ((14 - mes) / 12)) / 4)
        - ((ano + 4800 - ((14 - mes) / 12)) / 100)
        + ((ano + 4800 - ((14 - mes) / 12)) / 400)
        - 32045;

        return JND % 7;
}

/* FUNÇÃO verificaData:
*                  Função que verifica se a data é válida ou não, dado um DAY, MONTH e YEAR retorna:
*                        0 -> se a data não existir
*                        1 -> se a data existir
*/
unsigned char verificaData(unsigned char day, unsigned char month, unsigned int year){
        unsigned char bissexto = 0;

        if(year % 400 == 0 || (year % 100 != 0 && year % 4 == 0))
                bissexto = 1;

        if((day > 0 && day<=31) && (month>0 && month<=12)){
                if(day == 31 && (month != 1 && month != 3 && month != 5 && month != 7 && month != 8 && month != 10 && month != 12))
                        return 0;
                else if(day == 30 && (month != 4 && month != 6 && month != 9 && month != 11))
                        return 0;
                else if(day == 29 && month == 2 && bissexto == 0)
                        return 0;
                else if(month == 2 && day>29 )
                        return 0;


                return 1;
        }

        return 0;
}


/* FUNÇÃO atualizaHora:
*                  Função que atribuí os valores de minuto, somando em horas, dias, meses e anos.
*/
void volatile atualizaHora(){

        unsigned char mes[] = {31,28,31,30,31,30,31,31,30,31,30,31};

        if(dh.ano % 400 == 0 || (dh.ano % 100 != 0 && dh.ano % 4 == 0))
```

```
553                 mes[1] = 29;
554
555         dh.min++;
556
557         if(dh.min == 60){
558                 dh.hora++;
559                 dh.min = 0;
560                 if(dh.hora == 24){
561                         dh.dia++;
562                         dh.hora = 0;
563                         dh.min = 0;
564                         if(dh.dia > mes[dh.mes - 1]){
565                                 dh.mes++;
566                                 dh.dia = 1;
567                                 dh.hora = 0;
568                                 dh.min = 0;
569                                 if(dh.mes == 13){
570                                         dh.ano++;
571                                         dh.mes = 1;
572                                         dh.dia = 1;
573                                         dh.hora = 0;
574                                         dh.min = 0;
575                                 }
576                         }
577                 }
578         }
579 }
580
581 /* FUNÇÃO delayTime:
582 *                 Funcao para delay em miliegundos que recebe como parâmetro a quantidade de milisegundos desejados
583 *                OBS: Diferente do delayMicroseconds, esse delay ele NÃO ESPERA ATÉ TERMINAR A TEMPORIZAÇÃO
584 *
585 */
586 void delayTime(unsigned int miliseconds){
587         /*
588         * Funcao para delay em milisegundos
589         */
590         Timer0.flag=1;
591
592         if(!Timer0.finish){
593                 Timer0.cycles = miliseconds/65;
594                 Timer0.lastClock = 65535 - ((miliseconds % 65)*1000);
595
596                 TR0 = 1;
597                 Timer0.finish = 1;
598         }
599 }
600
601 /* FUNÇÃO timerMiliseconds:
602 *                 Função que controla a interrupção do TIMER0
603 */
604 void volatile timerMiliseconds() __interrupt 1{ //(slide aplicmicro 11_C.pdf pg.13)
605
606         TH0 = 0;                                                                        // Zerar os bits mais significativos do contador
607         TL0 = 0;                                                                        // Zerar os bits menos significativos do contador
608         TF0 = 0;                                                                       // Zero a flag do contador
609         TR0 = 0;                                                                       // Paro timer0
610
611         if(Timer0.cycles > 1){
612         Timer0.cycles--;                                                  // decrementa cycles
613                 TR0 = 1;                                                                // inicia contador
614         }
615         else if(Timer0.cycles == 1){
616                 TH0 = Timer0.lastClock & 0xFF ;                        // Atribui ao ultimo timer o valor dos bits mais significativos
617                 TL0 = Timer0.lastClock >> 8;                          // Atribui ao ultimo timer o valor dos bits menos significativos
618         Timer0.cycles--;                                                   // decrementa cycles
619                 TR0 = 1;                                                                // inicia contador
```

```
620         }
621         else if(Timer0.cycles == 0){
622         Timer0.finish = 0;                                          // finish é uma flag que indica que o tempo acabou
623         if(TM.control>=clear)
624                 atualizaHora();
625         }
626 }
627
628 /* FUNÇÃO configurations:
629 *               Função atribuí as configurações para o KIT funcionar
630 */
631 void configurations(){
632         IE = 0x8A;                                  // Habilitando interrupções, timer0 e timer1 (slide sistmicro 03_Interrupções.pdf pg.9)
633         IP = 0x02;                                  // Habilitando prioridade de interrupções para timer0 (slide aplicmicro 11_C.pdf pg.13)
634
635         TH0 = 0;                                   // Zerar os bits mais significativos do temporizador TIMER0
636         TL0 = 0;                                   // Zerar os bits menos significativos do temporizador TIMER0
637
638         TH1 = 0;                                   // Zerar os bits mais significativos do contador TIMER1
639         TL1 = 0;                                   // Zerar os bits menos significativos do contador TIMER1
640
641         TMOD = 0x01;                           // Habilitando contagem modo 16-bits (slide sistmicro 04_Timer.pdf pg.8)
642
643         P1 = 0x00;                                  // Desligando os LEDS
644
645         dh.dia = 0;
646         dh.mes = 0;
647         dh.ano = 0;
648         dh.hora = 0;
649         dh.min = 0;
650
651         TM.control = 0;
652         TM.value[0] = 16;               // Coloca ponto no display 1
653         TM.value[1] = 16;               // Coloca ponto no display 2
654         TM.value[2] = 16;               // Coloca ponto no display 3
655         TM.value[3] = 16;               // Coloca ponto no display 4
656 }
657
658 /* FUNÇÃO varredura:
659 *               Função executa a varredura do DISPLAY7S e do Teclado Matricial
660 */
661 void varredura(unsigned char D1, unsigned char D2, unsigned char D3, unsigned char D4, unsigned int time, unsigned char dot){
662         varreduraD7S(D1,D2,D3,D4,time,dot);
663         varreduraTecladoMat();
664 }
665
666 /* FUNÇÃO resetTM:
667 *               Espera um tempo depois de digitado o último display para mudar as opções
668 */
669 void resetTM(){
670         varredura(TM.value[0], TM.value[1], TM.value[2], TM.value[3],125,0);       // Mostra os valores selecionados nos Displays
671         if(!Timer0.flag)                                                                                                    // Se o timer não
          ↪    estiver funcionando
672                 delayTime(1000);                                                                                            // Ativa o timer
673         if(!Timer0.finish){                                                                                                    // Quando o timer
          ↪    acabar sua temporização
674                 Timer0.flag = 0;                                                                                            // o timer não está
                  ↪    mais funcionando
675                 TM.value[0] = 16;                                                                                            // Coloca ponto no
                  ↪    display 1
676                 TM.value[1] = 16;                                                                                            // Coloca ponto no
                  ↪    display 2
677                 TM.value[2] = 16;                                                                                            // Coloca ponto no
                  ↪    display 3
678                 TM.value[3] = 16;                                                                                            // Coloca ponto no
                  ↪    display 4
679                 TM.vectorControl = 0;                                                                                    // Reinicializa variável
```

```
680                 TM.control++;
681         }
682 }
683
684 /* FUNÇÃO main:
685 *               Função que é a primeira a ser executada
686 */
687 void main(){
688         unsigned char *digits;
689
690         configurations();
            ↪                                                                                                                                    //
            ↪   Atribuí as configurações
691
692         while(1){
693
694                 if(TM.control == reset1 || TM.control == reset2 || TM.control == reset3){
695                         resetTM();
696                         if(TM.control == reset3){
697                                 if(!verificaData(dh.dia,dh.mes,dh.ano)){                                                                    // Se a data
                                    ↪   colocada estiver errada, habilita para configurar novamente
698                                         TM.value[0] =
                                            ↪   16;                                                                                          //
                                            ↪   Coloca ponto no display 1
699                                         TM.value[1] =
                                            ↪   16;                                                                                          //
                                            ↪   Coloca ponto no display 2
700                                         TM.value[2] =
                                            ↪   16;                                                                                          //
                                            ↪   Coloca ponto no display 3
701                                         TM.value[3] =
                                            ↪   16;                                                                                          //
                                            ↪   Coloca ponto no display 4
702                                         TM.vectorControl = 0;
703                                         TM.control =
                                            ↪   day1;                                                                                         //
                                            ↪   Habilita colocar a data novamente
704                                 }
705                         }
706                 }
707                 else if(TM.control == clear)
                    ↪                                                                                                                   // Se estiver em
                    ↪   CONTROL = CLEAR
708                         varredura(dh.hora / 10,dh.hora % 10,dh.min / 10,dh.min%10,125,1);               // Mostra o horário
709                         //varredura(dh.min / 10,dh.min % 10,dh.hora / 10,dh.hora % 10,125,1);
710                 else if(TM.control == showDay)
711                         varredura(dh.dia / 10,dh.dia % 10,dh.mes / 10,dh.mes%10,125,1);                     // Mostra o dia e mes
712                 else if(TM.control == showYear)
713                         varredura(dh.ano/1000,(dh.ano%1000) / 100,(dh.ano%100) / 10,dh.ano%10,125,0);       // Mostra o ano
714                 else if(TM.control == showWeek){
715                         digits = digitosLetter7S(getDiaDaSemana(dh.dia,dh.mes,dh.ano));                     // Pega os digitos do dia da semana para
                            ↪   atribuir ao display
716                         if(digits[3] == 0)
717                                 digits[3] = 127;
718
719                         varredura(digits[0],digits[1],digits[2],digits[3],125,0);                       // Mostra o dia da semana
720                 }
721
722                 else if(TM.control < clear || TM.control == configHour || TM.control == configDay)
723                         varredura(TM.value[0], TM.value[1], TM.value[2], TM.value[3],125,0);            // Mostra o número que está sendo pressionado
724
725                 if(TM.control>=
                    ↪   clear){                                                                                                                 //
                    ↪   O tempo começa a se contado depois que configura todas as opção de horário, dia, mes e ano. Para começar depois de setar o horário,
                    ↪   trocar para TM.control >=reset1 aqui e na linha 620 do código.
726                         if(!Timer0.flag)                                                                                                    //
                            ↪   Se o timer não estiver ativo
```

```
727                             delayTime(2000);                                                                                        //
                                 ↪   Inicializa a contagem de 1 minuto
728                         if(!Timer0.finish)                                                                                          /
                             ↪   Quando a contagem acabar
729                             Timer0.flag =
                                 ↪   0;                                                                                                 //
                                 ↪   0 timer não estará mais ativo
730                 }
731
732                 if(TM.control >= clear && TM.control != configHour && TM.control != configDay &&
733                         (alarm.hora == dh.hora && alarm.min == dh.min && alarm.dia == dh.dia && dh.mes == dh.mes)){        // Se o dia e o horário do alarme
                             ↪   corresponder ao dia e horário atual de dh, acende os LEDS
734                         P1 += 1;
735                         if(P1 > 15)
736                             P1 = 1;
737                 }
738                 else if(alarm.min != dh.min)        // Quando os horários de alarme e dh não coincidirem os LEDS apagam
739                         P1 = 0;
740
741         }
742 }
```