/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Nome do Arquivo: displayLCD.c

\* Dependencias: Veja arquivo "displayLCD.h"

\* Processador: PIC18F4550

\* Opção de Clock: HS 20MHz (externo) - CPU: 48MHz

\* Compilador: C18

\*

\* Autor:

\*

\* Data: v1 - 08/Set/2014 -

\*

\* Descrição: Modificação da biblioteca xlcd.h fornecida com o compilador

\* C18 da Microchip. A biblioteca xlcd.h trata da comunicação

\* com display de LCD por meio de 4-bits (length), utilizando o

\* PORTB tanto para os 4 bits de dados quanto para os 3 bits de

\* controle (RS, RW e E).

\* No projeto da placa o display LCD é ligado às portas

\* digitais diversas do PIC 18F4550. A placa utiliza

\* os seguintes pinos para comunicação com o display LCD:

\* PIC18F4550(pino):: Display LCD

\* RD4 (27) :: D4

\* RD5 (28) :: D5

\* RD6 (29) :: D6

\* RD7 (30) :: D7

\* RD1 (20) :: RS

\* RD0 (19) :: E

\* RD2 (21) :: R/W

\* Notas:

\* - Essa bliblioteca de funçoes pode ser utilizado com qualquer LCD

\* 16x2 com controladores equivalentes como, o controlador Hitachi

\* HD44780 LCD controller.

\*

\* - O usuário deve definir os seguintes itens:

\* -- O tipo de interface de comunicação do LDC com o PIC (4 ou 8-bits):

\* Se for o modo 4-bit, utilizar o upper nibble (nibble é metade de um Byte)

\* , ou seja, enviar primeiro os 4 bits mais significativos (upper nibble)

\* e depois os 4 bits menos sgnificativos (lower nibble).

\*

\* -- Cursor e/ou Blink:

\* O usuário poderá definir se quer visualizar cursor, blink

\* ou nenhum dos dois.

\*

\* -- O sincronismo da comunicação com o LCD:

\* As temporizações entre o envio de comandos, não foi utilizado

\* o bit de Busy do LCD, mas sim, através de funções de atraso.

\* Os tempos de delay escolhidos estão colocados nos defines:

\* DELAY\_SINC e DELAY\_INICIO, que foram ecolhidos bem acima do

\* tempo mínimo requerido pelo display de LCD (5ms o de sincronismo e

\* 15ms o inicial).

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\* I N C L U D E S \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "pin\_manager.h"

#include "displayLCD.h"

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

/\*\* F U N C O E S \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void delay\_ms(unsigned char Cnt2)

{

waitTime(IfxStm\_getTicksFromMilliseconds(BSP\_DEFAULT\_TIMER, Cnt2)); /\*! Wait in milliseconds \*/

}

void delay\_us(unsigned char Cnt2)

{

waitTime(IfxStm\_getTicksFromMicroseconds(BSP\_DEFAULT\_TIMER, Cnt2)); /\*! Wait in microseconds \*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Funcao: void ConfiguraLCD(void)

\* Entrada: Nenhuma (void)

\* Saída: Nenhuma (void)

\* Descrição: Configura portas do Microcontrolador que estão conectadas ao

\* display LCD como saida. Essa rotina é baseada no controlador de LCD

\* Hitachi HD44780. Configura os parametros de comunicação com o display,

\* tais como:

\* - modo de operação (4 ou 8 data bits)

\* - tipo de cursor

\* - tamanho da fonte

\* Ao final da configuração limpa o display.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ConfiguraLCD(void)

{

// Configura os pinos de controle (RW,RS,E) como saída.

LCD\_RW\_OUTPUT();

LCD\_RS\_OUTPUT();

LCD\_E\_OUTPUT();

// Configura os pinos de dados (D4,D5,D6,D7) como saída.

LCD\_D4\_OUTPUT();

LCD\_D5\_OUTPUT();

LCD\_D6\_OUTPUT();

LCD\_D7\_OUTPUT();

// Inicio da rotina de inicialização do display LCD

// Aguardar tempos e enviar 3 vezes o comando 0b0011XXXX como descrito no datasheet

// do controlador HD44780 ou SPLC780D

//Seleciona pinos de Controle para escrita do comando

RW\_PIN\_LOW(); //escrita=0, leitura=1

RS\_PIN\_LOW(); // comando=0, dados=1

E\_PIN\_LOW(); // desabilita=0, habilita=1

//selecioan pinos de dados com comando de inicialização 0b0011 no Nibble-High

//(os 4 bits de dados mais significativos), os bits do Nibble-Low são Don't Care

D7\_PIN\_LOW();

D6\_PIN\_LOW();

D5\_PIN\_HIGH();

D4\_PIN\_HIGH();

//Passo 1: Envio do comando de inicialização 0b0011

// Delay inicial de 15ms após alimentar o LCD: Vdd > 4,5V

delay\_ms(15);

//Habilita leitura do comando pelo LCD, através do pino de enable, para tanto,

//um pulso de 1ms com borda de descida é efetuado

E\_PIN\_HIGH();

delay\_us(20);

E\_PIN\_LOW();

delay\_us(20);

//Passo 2: Envio do comando de inicialização 0b0011

// Delay de pelo menos 4.1ms

delay\_ms(5);

//Pulso no pino de Enable para habilitar leitura do LCD na borda de descida

E\_PIN\_HIGH();

delay\_us(20);

E\_PIN\_LOW();

delay\_us(20);

//Passo 3: Envio do comando de inicialização 0b0011

// Delay de pelo menos 100us

delay\_us(50);

//Pulso no pino de Enable para habilitar leitura do LCD na borda de descida

E\_PIN\_HIGH();

delay\_us(20);

E\_PIN\_LOW();

delay\_us(20);

//Passo 4: Envio da Configuração 0b0010 no Nibble-High

D7\_PIN\_LOW();

D6\_PIN\_LOW();

D5\_PIN\_HIGH();

D4\_PIN\_LOW();

//Pulso no pino de Enable para habilitar leitura do LCD na borda de descida

E\_PIN\_HIGH();

delay\_us(20);

E\_PIN\_LOW();

delay\_us(20);

//Envio dos comandos de Configuração do LCD

EscreveComandoLCD(0x28); // Function Set: 4bit interface, 2 lines, Font 5x7

EscreveComandoLCD(0x04); // Entry mode set: Increment, Shift OFF

EscreveComandoLCD(0x0C); // Display Control: Display ON, Cursor ON, Blink OFF

EscreveComandoLCD(0x01); // Clear display

/\*

Delay(); //aguarda Reset do Display LCD

EscreveComandoLCD(0x28);

Delay();

EscreveComandoLCD(0x28);

Delay();

EscreveComandoLCD(0x06);

Delay();

EscreveComandoLCD(0x0C);

Delay();

EscreveComandoLCD(0x01);

Delay(); \*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Funcao: void EscreveComandoLCD(unsigned char cmd)

\* Entrada: cmd: comando que será enviado para o LCD

\* Saída: Nenhuma (void)

\* Descrição: (Write a command to the LCD)

\* Escreve um comando para o display de LCD.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void EscreveComandoLCD(unsigned char cmd)

{

//Seleciona pinos de Controle para escrita do comando

RW\_PIN\_LOW(); //escrita=0, leitura=1

RS\_PIN\_LOW(); // comando=0, dados=1

E\_PIN\_LOW(); // desabilita=0, habilita=1

//delay\_us(40);

//Seleciona pinos de Dados: Nibble-High

if(cmd&0x10){D4\_PIN\_HIGH();}

else {D4\_PIN\_LOW();}

if(cmd&0x20){D5\_PIN\_HIGH();}

else {D5\_PIN\_LOW();}

if(cmd&0x40){D6\_PIN\_HIGH();}

else {D6\_PIN\_LOW();}

if(cmd&0x80){D7\_PIN\_HIGH();}

else {D7\_PIN\_LOW();}

//Pulso no pino de Enable para habilitar leitura do LCD na borda de descida

E\_PIN\_HIGH();

delay\_us(20);

E\_PIN\_LOW();

delay\_us(20);

//Seleciona pinos de Dados: Nibble-Low

if(cmd&0x01){D4\_PIN\_HIGH();}

else {D4\_PIN\_LOW();}

if(cmd&0x02){D5\_PIN\_HIGH();}

else {D5\_PIN\_LOW();}

if(cmd&0x04){D6\_PIN\_HIGH();}

else {D6\_PIN\_LOW();}

if(cmd&0x08){D7\_PIN\_HIGH();}

else {D7\_PIN\_LOW();}

//Pulso no pino de Enable para habilitar leitura do LCD na borda de descida

E\_PIN\_HIGH();

delay\_us(20);

E\_PIN\_LOW();

delay\_us(20);

//Como os comandos Clear Display(0x01) e Cursor Home(0b0000001x),

//demoram mais tempo para serem executados,cerca de cerca de 1,5ms.

//Garante-se um atraso maior caso os mesmos sejam efetuados

if (cmd==0x01 || cmd==0x02 || cmd==0x03)

{ delay\_ms(2); }

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Funcao: void EnderecoCursor(unsigned char endereco)

\* Entrada: endereco: endereço da memoria de dados do display que se

\* colocar o cursor

\* Saída: Nenhuma (void)

\* Descrição: (Set the Display Data RAM Address)

\* Essa rotina posiciona o cursor no endereço desejado da memoria RAM do

\* display LCD. Os comandos de escrita seguintes vão começar a partir da

\* posição escolhida por esta função

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void EnderecoCursor(unsigned char endereco)

{

//Seleciona pinos de Controle para escrita do comando

RW\_PIN\_LOW(); //escrita=0, leitura=1

RS\_PIN\_LOW(); // comando=0, dados=1

E\_PIN\_LOW(); // desabilita=0, habilita=1

//delay\_us(40);

//Seleciona pinos de Dados: Nibble-High

D7\_PIN\_HIGH();

if(endereco&0x10){D4\_PIN\_HIGH();}

else {D4\_PIN\_LOW();}

if(endereco&0x20){D5\_PIN\_HIGH();}

else {D5\_PIN\_LOW();}

if(endereco&0x40){D6\_PIN\_HIGH();}

else {D6\_PIN\_LOW();}

//Pulso no pino de Enable para habilitar leitura do LCD na borda de descida

E\_PIN\_HIGH();

delay\_us(20);

E\_PIN\_LOW();

delay\_us(20);

//Seleciona pinos de Dados: Nibble-Low

if(endereco&0x01){D4\_PIN\_HIGH();}

else {D4\_PIN\_LOW();}

if(endereco&0x02){D5\_PIN\_HIGH();}

else {D5\_PIN\_LOW();}

if(endereco&0x04){D6\_PIN\_HIGH();}

else {D6\_PIN\_LOW();}

if(endereco&0x08){D7\_PIN\_HIGH();}

else {D7\_PIN\_LOW();}

//Pulso no pino de Enable para habilitar leitura do LCD na borda de descida

E\_PIN\_HIGH();

delay\_us(20);

E\_PIN\_LOW();

delay\_us(20);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Funcao: void PosicaoCursorLCD(unsigned char linha, unsigned char coluna)

\* Entrada: linha: número de 1 à 4 da linha em que se deseja deixar o cursor

\*

\* coluna: número de 1 à 16 coluna linha em que se deseja deixar o cursor

\*

\* Saída: Nenhuma (void)

\* Descrição: Essa rotina posiciona o cursor na tela do display, na linha

\* e coluna desejados. Os comandos de escrita seguintes vão começar a partir

\* da posição escolhida por esta função

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void PosicaoCursorLCD(unsigned char linha, unsigned char coluna)

{

//endereço de inicio de cada linha

#define linha1 0x00

#define linha2 0x40

#define linha3 0x14

#define linha4 0x54

unsigned char EnderecoLCD = 0x00;

coluna--;

if (coluna >= 0)

{

//Calcula o endereço da memória RAM do display que deve ser usado

//para posicionar o cursor na linha e coluna desejados

switch (linha)

{

case 1:

EnderecoLCD = linha1 + coluna;

EnderecoCursor(EnderecoLCD);

break;

case 2:

EnderecoLCD = linha2 + coluna;

EnderecoCursor(EnderecoLCD);

break;

case 3:

EnderecoLCD = linha3 + coluna;

EnderecoCursor(EnderecoLCD);

break;

case 4:

EnderecoLCD = linha4 + coluna;

EnderecoCursor(EnderecoLCD);

break;

default:

EnderecoLCD = linha1;

break;

}

}

else

{

EnderecoLCD = linha1;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Funcao: void EscreveCaractereLCD(unsigned char data)

\* Entrada: data: Byte de dados para ser escrito na memoria RAM do LCD

\* Saída: Nenhuma (void)

\* Descrição: (Write a Data byte to the LCD)

\* Essa rotina escreve o byte de caracteres no display a partir da posição

\* de memoria atual do cursor ou da posição de memória escolhida com as

\* funções PosicaoCursorLCD() ou EnderecoCursor().

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void EscreveCaractereLCD(char data)

{

//Seleciona pinos de Controle para escrita do comando

RW\_PIN\_LOW(); //escrita=0, leitura=1

RS\_PIN\_HIGH(); // comando=0, dados=1

E\_PIN\_LOW(); // desabilita=0, habilita=1

//delay\_us(40);

//Seleciona pinos de Dados: Nibble-High

if(data&0x10){D4\_PIN\_HIGH();}

else {D4\_PIN\_LOW();}

if(data&0x20){D5\_PIN\_HIGH();}

else {D5\_PIN\_LOW();}

if(data&0x40){D6\_PIN\_HIGH();}

else {D6\_PIN\_LOW();}

if(data&0x80){D7\_PIN\_HIGH();}

else {D7\_PIN\_LOW();}

//Pulso no pino de Enable para habilitar leitura do LCD na borda de descida

E\_PIN\_HIGH();

delay\_us(20);

E\_PIN\_LOW();

delay\_us(20);

//Seleciona pinos de Dados: Nibble-Low

if(data&0x01){D4\_PIN\_HIGH();}

else {D4\_PIN\_LOW();}

if(data&0x02){D5\_PIN\_HIGH();}

else {D5\_PIN\_LOW();}

if(data&0x04){D6\_PIN\_HIGH();}

else {D6\_PIN\_LOW();}

if(data&0x08){D7\_PIN\_HIGH();}

else {D7\_PIN\_LOW();}

//Pulso no pino de Enable para habilitar leitura do LCD na borda de descida

E\_PIN\_HIGH();

delay\_us(20);

E\_PIN\_LOW();

delay\_us(20);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Funcao: void EscreveInteiroLCD(int valor)

\* Entrada: valor: numero inteiro com sinal (de -32768 a 32767) a ser escrito no

\* display.

\*

\* Saída: Nenhuma (void)

\* Descrição: Essa rotina escreve o numero inteiro no display a partir da posição

\* de memoria atual do cursor ou da posição de memória escolhida com as

\* funções PosicaoCursorLCD() ou EnderecoCursor().

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void EscreveInteiroLCD(int valor)

{

char texto[7] = "000000";

(void)itoa(valor,texto, 10);

EscreveFraseRamLCD(texto);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Funcao: inline void swap(char \*x, char \*y)

\* Entrada: valor: dois endereço de caracter.

\*

\* Saída: Nenhuma (void)

\* Descrição: essa função coloca o valor de x em y e de y em x.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

inline void swap(char \*x, char \*y) {

char t = \*x; \*x = \*y; \*y = t;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Funcao: Reverse(char \*buffer, int i, int j)

\* Entrada: valor: sequencia de caracter.

\*

\* Saída: sequencia de caracter inversa

\* Descrição: essa função inverte a escrita do buffer recebido.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

char\* Reverse(char \*buffer, int i, int j)

{

while (i < j)

swap(&buffer[i++], &buffer[j--]);

return buffer;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Funcao: char\* itoa(int value, char\* buffer, int base);

\* Entrada: valor: converte um inteiro em caracter.

\*

\* Saída: sequencia de caracter

\* Descrição: essa função converte um inteiro em caracter.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

char\* itoa(int value, char\* buffer, int base)

{

int n, i;

// invalid input

if (base < 2 || base > 32)

return buffer;

// consider absolute value of number

n = abs(value);

i = 0;

while (n)

{

int r = n % base;

if (r >= 10)

buffer[i++] = (char)(65 + (r - 10));

else

buffer[i++] = (char)(48 + r);

n = n / base;

}

// if number is 0

if (i == 0)

buffer[i++] = '0';

// If base is 10 and value is negative, the resulting string

// is preceded with a minus sign (-)

// With any other base, value is always considered unsigned

if (value < 0 && base == 10)

buffer[i++] = '-';

buffer[i] = '\0'; // null terminate string

// reverse the string and return it

return Reverse(buffer, 0, i - 1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Funcao: void EscreveFloatLCD(float valor, char formatacao)

\* Entrada: valor: numero float com sinal (de -9999.9999 a 9999.9999) que

\* será escrito no display

\*

\* formatacao: numero de casas decimais desejadas após a

\* virgula (de 0 à 4 casas decimais).

\*

\* Saída: Nenhuma (void)

\* Descrição: Essa rotina escreve o numero float no display a partir da posição

\* atual do cursor com quantas casas decimais forem definidas no

\* parâmetro formatacao.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void EscreveFloatLCD(float valor, char formatacao)

{

int numInt = 0; // Guarda parte inteira

float numFloat = 0; // Guarda parte fracionária

char n; // contador do comando for

// Escreve parte inteira

numInt = (int)valor;

EscreveInteiroLCD(numInt);

// Escreve as casas decimais indicada na formatação

if( formatacao > 0 )

{

EscreveCaractereLCD('.');

// Separa parte fracionária

numFloat = (valor - numInt);

// Se NumFloat é negativo, corrige o sinal

numFloat = (float)(abs(numFloat));

// Garante que so escreve ate 5 casas decimais

if(formatacao>4)

{formatacao=4;}

// Calcula as casas decimais

for (n = 0; n<formatacao; n++)

{

numFloat \*= 10;

// Escreve zeros após a virgula

if(numFloat<1)

{

EscreveCaractereLCD('0');

}

}

// Escreve parte fracionária

EscreveInteiroLCD((int)numFloat);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Funcao: void EscreveFraseRamLCD(char \*buffer)

\* Entrada: \*buffer: apontador para string na memoria de dados(RAM)

\* Saída: Nenhuma (void)

\* Descrição: Essa rotina copia uma string terminada com caracter nulo

\* da memoria de dados do microcontrolador para o display de LCD.

\* A string apontada por \*buffer é copiada no display a partir da posição

\* de memoria atual do cursor ou da posição de memória

\* escolhida com as funções PosicaoCursorLCD() ou EnderecoCursor().

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void EscreveFraseRamLCD(char \*buffer)

{

while(\*buffer) // escreve dados no LCD ate achar caracter nulo

{

EscreveCaractereLCD(\*buffer); // escreve caracter no LCD

buffer++; // incrementa apontador

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Funcao: void EscreveFraseLCD(const char \*buffer)

\* Entrada: \*buffer: apontador para string na memoria de programa(ROM)

\* Saída: Nenhuma (void)

\* Descrição: Essa rotina copia uma string terminada com caracter nulo

\* da memoria de programa do microcontrolador para o display de LCD.

\* A string apontada por \*buffer é copiada no display a partir da

\* posição de memoria atual do cursor ou da posição de memória escolhida

\* com as funções PosicaoCursorLCD() ou EnderecoCursor().

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void EscreveFraseLCD(const char \*buffer)

{

while(\*buffer) // escreve dados no LCD ate achar caracter nulo

{

EscreveCaractereLCD(\*buffer); // escreve caracter no LCD

buffer++; // incrementa apontador

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Funcao: void CriaCaractereLCD(unsigned char endereco, const char \*buffer)

\* Entrada: endereco: Numero de 0 à 7 que indica a posição da CGRAM que o

\* caractere será criado.

\*

\* \*buffer: apontador para o vetor de char com tamanho 8 que

\* guarda cada linha de desenho do caractere que será criado.

\*

\* Saída: Nenhuma (void)

\* Descrição: Essa rotina cria um caractere de usuário em uma das 7 posições

\* disponiveis na CGRAM do display. O caractere criado tem tamanho 8x5 pixel

\* (8 linhas e 5 colunas), portanto cada valor do vetor buffer representa

\* uma linha, onde os 3 bits mais significativos de cada valor é desprezado.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void CriaCaractereLCD(unsigned char endereco, const char \*buffer)

{

unsigned char cmd; //guarda comando a ser enviado para LCD

unsigned char n = 0; //contador do comando for

//Dependendo do valor de endereco, seleciona o comando correto

//Garante que enderco só tenha valores de 0 à 7

endereco = endereco & 0x07;

switch (endereco)

{ case 0: cmd = 0x40; break;

case 1: cmd = 0x48; break;

case 2: cmd = 0x50; break;

case 3: cmd = 0x58; break;

case 4: cmd = 0x60; break;

case 5: cmd = 0x68; break;

case 6: cmd = 0x70; break;

case 7: cmd = 0x78; break;

default:cmd = 0x40; break;

}

//Comando de escrita na CGRAM (criação de caratere personalizado)

EscreveComandoLCD(cmd);

//Escreve cada linha do caractere com a sequencia de valores do buffer

for(n=0; n<8; n++)

{

EscreveCaractereLCD(\*buffer); // escreve caracter no LCD

buffer++; // incrementa apontador

}

//Comando para sair do modo criação de caratere e posicionar cursor

//na linha 1, coluna 1 da tela.

EnderecoCursor(0x00);

}

void EscreveShortLongLCD(unsigned long valor)

{

char texto[9] = "00000000";

//ultoa(texto,valor, 10);

snprintf(texto, 10, " %ld", valor);

EscreveFraseRamLCD(texto);

}

void PrintNUM\_LCD(unsigned long number)

{

/\*\*\*\*\*\*\*\*\* Maximum length of Number (Decimal) = 10 \*\*\*\*\*\*\*\*\*\*/

unsigned int unit=0;

unsigned char numbers[]={'0','1','2','3','4','5','6','7','8','9'};

unsigned char strNum[]={0,0,0,0,0,0,0,0,0,0};

unsigned long i=0, j=0;

while(number > 9)

{

unit = (unsigned int)(number % 10);

strNum[i] = (unsigned char)(unit);

number = number/10;

i++;

}

strNum[i] = (unsigned int) number;

for(j=i;j>-1;j--)

{

//EscreveFraseRamLCD(numbers[strNum[j]]);

EscreveCaractereLCD(numbers[strNum[j]]);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Funcao: void EscreveByteLCD(unsigned char data, unsigned char linha, unsigned char coluna)

\* Entrada: valor: numero inteiro (de 0 a 999) a ser escrito no

\* display.

\*

\* Saída: Nenhuma (void)

\* Descrição: Essa rotina escreve o numero inteiro no display a partir da posição

\* de memoria atual do cursor ou da posição de memória escolhida com as

\* funções PosicaoCursorLCD() ou EnderecoCursor().

\* Ex: EscreveByteLCD(256, 1,2)

\* 1

\* Coluna -> 1234567890123456

\* linha 1-> 256

\* linha 2->

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void EscreveByteLCD(unsigned char data, unsigned char linha, unsigned char coluna)

{

unsigned char \_i = 2;

unsigned char \_j = 0;

unsigned char ASCII[3];

ASCII[0] = 0x20;

ASCII[1] = 0x20;

ASCII[2] = 0x20;

while (data>=10)

{

ASCII[\_i] = data%10;

data = data/10;

ASCII[\_i] = ASCII[\_i]+0x30;

--\_i;

}

ASCII[\_i] = data+0x30;

PosicaoCursorLCD (linha,coluna);

while(\_j<3)

{

EscreveCaractereLCD(ASCII[\_j]);

++\_j;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Funcao: void EscreveWordLCD(unsigned char data, unsigned char linha, unsigned char coluna)

\* Entrada: valor: numero inteiro (de 0 a 999) a ser escrito no

\* display.

\*

\* Saída: Nenhuma (void)

\* Descrição: Essa rotina escreve o numero inteiro no display a partir da posição

\* de memoria atual do cursor ou da posição de memória escolhida com as

\* funções PosicaoCursorLCD() ou EnderecoCursor().

\* Ex: EscreveWordLCD(256, 1,2)

\* 1

\* Coluna -> 1234567890123456

\* linha 1-> 00256

\* linha 2->

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void EscreveWordLCD(unsigned int data, unsigned char linha,unsigned char coluna)

{

char \_i = 4;

char \_j = 0;

unsigned int ASCII[5];

ASCII[0] = 0x20;

ASCII[1] = 0x20;

ASCII[2] = 0x20;

ASCII[3] = 0x20;

ASCII[4] = 0x20;

while (data>=10)

{

ASCII[\_i] = (unsigned int)data%10;

data = data/10;

ASCII[\_i] = ASCII[\_i]+0x30;

--\_i;

}

ASCII[\_i] = data+0x30;

PosicaoCursorLCD (linha,coluna);

while(\_j<5)

{

EscreveCaractereLCD((char)(ASCII[\_j]));

++\_j;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Funcao: void EscreveLongLCD(unsigned char data, unsigned char linha, unsigned char coluna)

\* Entrada: valor: numero inteiro (de 0 a 999999999) a ser escrito no

\* display.

\*

\* Saída: Nenhuma (void)

\* Descrição: Essa rotina escreve o numero inteiro no display a partir da posição

\* de memoria atual do cursor ou da posição de memória escolhida com as

\* funções PosicaoCursorLCD() ou EnderecoCursor().

\* Ex: EscreveLongLCD(256, 1,2)

\* 1

\* Coluna -> 1234567890123456

\* linha 1-> 000000256

\* linha 2->

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void EscreveLongLCD(unsigned long data, unsigned char linha,unsigned char coluna)

{

char \_i = 8;

char \_j = 0;

unsigned char ASCII[9];

ASCII[0] = 0x20;

ASCII[1] = 0x20;

ASCII[2] = 0x20;

ASCII[3] = 0x20;

ASCII[4] = 0x20;

ASCII[5] = 0x20;

ASCII[6] = 0x20;

ASCII[7] = 0x20;

ASCII[8] = 0x20;

while (data>=10)

{

ASCII[\_i] = (unsigned char)data%10;

data = data/10;

ASCII[\_i] = ASCII[\_i]+0x30;

--\_i;

}

ASCII[\_i] = (unsigned char)(data+0x30);

PosicaoCursorLCD (linha,coluna);

while(\_j<9)

{

EscreveCaractereLCD(ASCII[\_j]);

++\_j;

}

}