175480_06

EA871

Atividade 1

1. Através dos terminais D0 à D7 do display, respectivamente, PTC0 à PTC7 do microcontrolador são passados os dados. É importante ressaltar que o terminal enable deve estar ativo em nível alto durante o tempo adequado para que os dados sejam passados e, em seguida, voltar ao nível baixo. Dentre os três pinos de controle , é importante que o R/W esteja em nível baixo, indicando leitura de dados.

2. O sinal alto em “E” deve durar 230ns ou mais. Nesta instrução, um pulso maior que o tempo mínimo não interfere na instrução, mas há outras instruções que não são executadas normalmente caso seu tempo requisitado seja excedido.

3. Os comandos indisponíveis são todos aqueles de leitura, pois o pino R/W está ligado ao GND, ou seja, setado em nível baixo permanentemente. Logo, não há como colocá-lo em nível alto, de forma a executar os comandos de leitura.

4. Primeiramente, deve-se ativar o clock da placa, isto é, o bit de SIM_SCGC5 correspondente ao da porta C (pino 11, 0x00000400). Devemos setar os 9 primeiros bits da PORTC como saídas, ou seja, passar para GPIOC_PDDR o valor 0x000003FF. Por último, o pino PTC8 deve estar em nível baixo para podermos habilitar a leitura de comandos no LCD, assim, fazemos com que GPIOC_PCOR receba 0x00000100.

5. Deve-se ligar o display e configurá-lo para uma ou duas linhas. esperar 39us, configurar o controle On/Off do Display, esperar 39us, executar o clear do display, esperar 1.53ms, configurar o modo de entrada de dados e aguardar novamente 39us. Para fazer isto em C, podemos utilizar comandos que setam os bits a partir de acessos de posições da memória, sendo que para saber quais bits devemos setar, deve-se verificar os pinos conectados no display.

6. No manual do display LCD, na tabela da página 13, encontram-se os valores das respectivas posições no display. Assim, somando-se 0x80 (devido ao sinal de controle nível alto no sexto terminal) ao valor de posição desejado, deve-se passar aos terminais do display (de D0 a D7) o resultado desta soma. Ressalta-se que os terminais RS e R/W devem estar em nível baixo e que, ao final desta operação, é necessário aguardar 39us para o display terminar a execução desta.

7. Para posicionar o cursor na posição inicial desejada, segue-se os passos do item anterior. Em seguida, envia-se o caractere desejado e passa-se para a nova posição, repetindo este procedimento a cada caractere desejado. Para passar um caractere, ativa-se os sinais de controle necessários para gravação de dados no display e, após zerar os oito primeiros bits de GPIOC_PDOR, escreve-se nestes

primeiros bits o valor do caractere desejado. Imediatamente antes de se passar os bits do caractere desejado ao GPIOC_PDOR, ativa-se o sinal de enable pelo tempo mínimo especificado pelo display e, após transferir o caractere, o nível deste é levado de volta ao nível baixo, aguardando os 39us para o display ser desocupados e poder receber as instruções seguintes.

8. Para programar um caractere que não está na tabela original do display LCD , habilita-se o CG RAM para endereçar uma região da memória interna do LCD própria para isto. Assim, dá-se um comando para o local onde o caractere será endereçado e então se passam os dados que representarão cada um uma das 8 linhas disponíveis e seus pontos respectivos às 5 colunas. Bits em nível alto representam pixels pretos no LCD, e os em nível baixo, pixels apagados (possuem a cor do fundo do display). Para executar esta operação, também iremos utilizar os registradores da memória sendo acessados por comandos de mapeamento na memória.

9. Tendo executado o passo anterior para criar o caractere, podemos acessá-lo passando como um dado o endereço onde o caractere está salvo, escrevendo assim o caractere novo no display.

Atividade 2

Os comandos, funções e parâmetros estão devidamente comentados e explicados ao longo dos códigos.

main:

```
#include "derivative.h" /* include peripheral declarations */
#include "LiquidCrystal.h"

int main(void){

 int i=0;

 inicGPIO();
 inicLCD(2);

 passaComando(0x40);
 delay10us(4);


 setChar( 0b00000); // Criando primeiro caractere especial.
 setChar( 0b00000); //Parte de trás do navio.
```

```
 setChar( 0b11111);
 setChar( 0b01000);
 setChar( 0b00111);
 setChar( 0b00000);
 setChar( 0b00000);
 setChar( 0b00000);

  passaComando(0x60);
  delay10us(4);

  setChar( 0b00000); // Criando segundo caractere especial.
  setChar( 0b00000); //Parte da frente do navio.
  setChar( 0b11111);
  setChar( 0b00010);
  setChar( 0b11100);
  setChar( 0b00000);
  setChar( 0b00000);
  setChar( 0b00000);

//====ATIVIDADE_1============================================================
==========================================================================
====
 setString("175480",0,0);
 setPos(1,11);
 setString("EA871",1,11);
 delay10us(500000);
 clear();
//====ATIVIDADE_1============================================================
==========================================================================
====

//====ATIVIDADE_2============================================================
==========================================================================
====

//A atividade 2 tenta representar um navio soltando fumça pela chaminé e indo para o lado.

 while(1)
  for(i=0; i<11; i++){

   setPos(0,i); //Fumaça da chaminé
   setString("Oo",0,i+1);


   setPos(1,i);
   setChar( 0x00); //Parte debaixo do NAVIO.
```

```
  setString("=",1,i+1);
  setPos(1,i+5);
  setChar( 0x04);

  delay10us(50000);
  clear();

 };
//====ATIVIDADE_2=============================================================
=============================================================================
====

 return 0;
}
```

LiquidCrystal:

```
/*
 * LiquidCrystal.c
 *
 *  Created on: Sep 22, 2016
 *          Author: ea871
 */

#include "LiquidCrystal.h"
#include "derivative.h" /* include peripheral declarations */

#define clearCMD 0b0000000001
#define posInicial 0b0000000010
#define oneLineMode 0b0000110111
#define twoLineMode 0b0000111111


void inicGPIO(){
 SIM_SCGC5 |= 0x00000800; // Ativa clock da PORTC.

 PORTC_PCR0  = 0x00000100; // Configura pin MUX control what.
 PORTC_PCR1  = 0x00000100;
 PORTC_PCR2  = 0x00000100;
 PORTC_PCR3  = 0x00000100;
 PORTC_PCR4  = 0x00000100;
 PORTC_PCR5  = 0x00000100;
 PORTC_PCR6  = 0x00000100;
 PORTC_PCR7  = 0x00000100;
 PORTC_PCR8  = 0x00000100;
```

```
 PORTC_PCR9  = 0x00000100;

 GPIOC_PDDR  = 0x000003FF; // Dez primeiros terminais de PDDR são saídas.
 GPIOC_PDOR  &= 0xFFFFFC00; // Zerando os terminais que serão utilizados aqui.
}

void delay10us(int n){
 n *= 21; // 10*n em microsegundos, aproximadamente

 while(n>0) n--;
}

void inicLCD(int qtsLinhas){

 delay10us(3001); // Tempo para o display ser energizado.

 if(qtsLinhas==1){ // Se for desejado somente uma linha...
  passaComando(oneLineMode);
 }else { // Se forem duas entao...
  passaComando(twoLineMode);
 }
 delay10us(4); // 39ms para garantir que o display recebeu e se desocupou da operação.

 passaComando(0b0000001100);// Comando para deixar o display no modo ON e o cursor
no modo OFF.
 delay10us(4);

 clear(); //Limpa o display conforme requisitado.
 passaComando(posInicial); //Retorna o cursor ao início da tela.
 delay10us(154); // Delay de 1,53ms.
}

void passaComando(unsigned int comando){
 GPIOC_PDOR |= (1<<9)|(1<<8); // Enable em nível alto.
 GPIOC_PCOR = (1<<8); // PTC8 valendo zero para utilizar o modo comando.
 GPIOC_PDOR  &= 0xFFFFFF00; // Zera os bits nos quais será salvo o comando
requisitado.
 GPIOC_PDOR |= comando; //Guarda o comando nos oito primeiros bits.
 GPIOC_PCOR = (1<<9); //Reseta o terminal de Enable para podermos prosseguir a
execução do LCD.
}

void clear(){ //clear display
 passaComando(clearCMD);
 delay10us(154); //aguarda 1,53ms.
}
```

```
void setString(char frase[], int linha, int coluna){

 setPos(linha, coluna); //Move o cursos à posição desejada.

 int i=0;
 while(frase[i]){
  setChar(frase[i++]); // Utiliza o caractere e já passa ao próximo.
   }
}

void setChar(char caractere){
 GPIOC_PDOR |= (1<<9)|(1<<8); // Enable em nível alto e PTC8 valendo 1 para utilizar o
modo comando.
 GPIOC_PDOR  = (GPIOC_PDOR & 0xFFFFFF00) | caractere; // Passa o caractere.
 GPIOC_PCOR = (1<<9); // Retorna enable para o nível baixo.
 delay10us(4); // 39ms.
}

void setPos(int linha, int coluna){

 int aux=0;

 if (linha == 1) aux=0x40;//0x40 para escrever na segunda linha.

 aux += coluna; // Define a coluna desejada.
 aux += 0b0010000000; // SET DD.

 passaComando(aux);

 delay10us(4);// aguarda 39ms.
}
```