



LCD (*Liquid Crystal Display*)

Alex Vidigal Bastos
www.decom.ufop.br/alex/
alexvbh@gmail.com

Sumário

- Introdução
- Displays LCD
- Características dos Displays LCD
- Sobre o Hardware (LCD)
- Funções dos pinos do LCD
- Comunicação
- Inicialização do Display
- Comandos do LCD
- Escrita no LCD
- Tabela ASCII
- Programação

Introdução

- Imagem é formado pela polarização da luz;
- Pixel formado pela retenção da luz;
- Geralmente tem um controlador integrado;
 - Hitachi HD44780
- Formato de 7 segmentos ou matricial;
- No formato matricial é possível a representação de símbolos;

Displays LCD

Existem vários tipos de LCD no mercado atualmente e são descritos por AxB onde A é o número de colunas e B o número de linhas;

Exemplo:

08x02 – oito colunas e duas linhas;

16x01 – 16 colunas e uma linha;

16x02 – 16 colunas e duas linhas (kit)

Displays LCD

Modelos com **back-light** e sem **back-light**.

Back-light – é um LED ou conjunto de LEDs no fundo do display que permite que o escrito possa ser lido em ambientes de baixa ou nenhuma luminosidade.

Displays LCD



Figura 6: Display 20X02 sem back-light



Figura 7: Display 20X04 com back-light

Características dos Displays LCD

Os modelos mais comuns se comunicam através de pinos de I/O de seu microcontrolador chamados de LCDs paralelos.

Outros exemplos:

- I2c;
- SPI;
- outro protocolo;

Características dos Displays LCD

- Diversas cores de escrito e de fundo;
- Pinos de conexão na parte de cima , ao lado e embaixo;

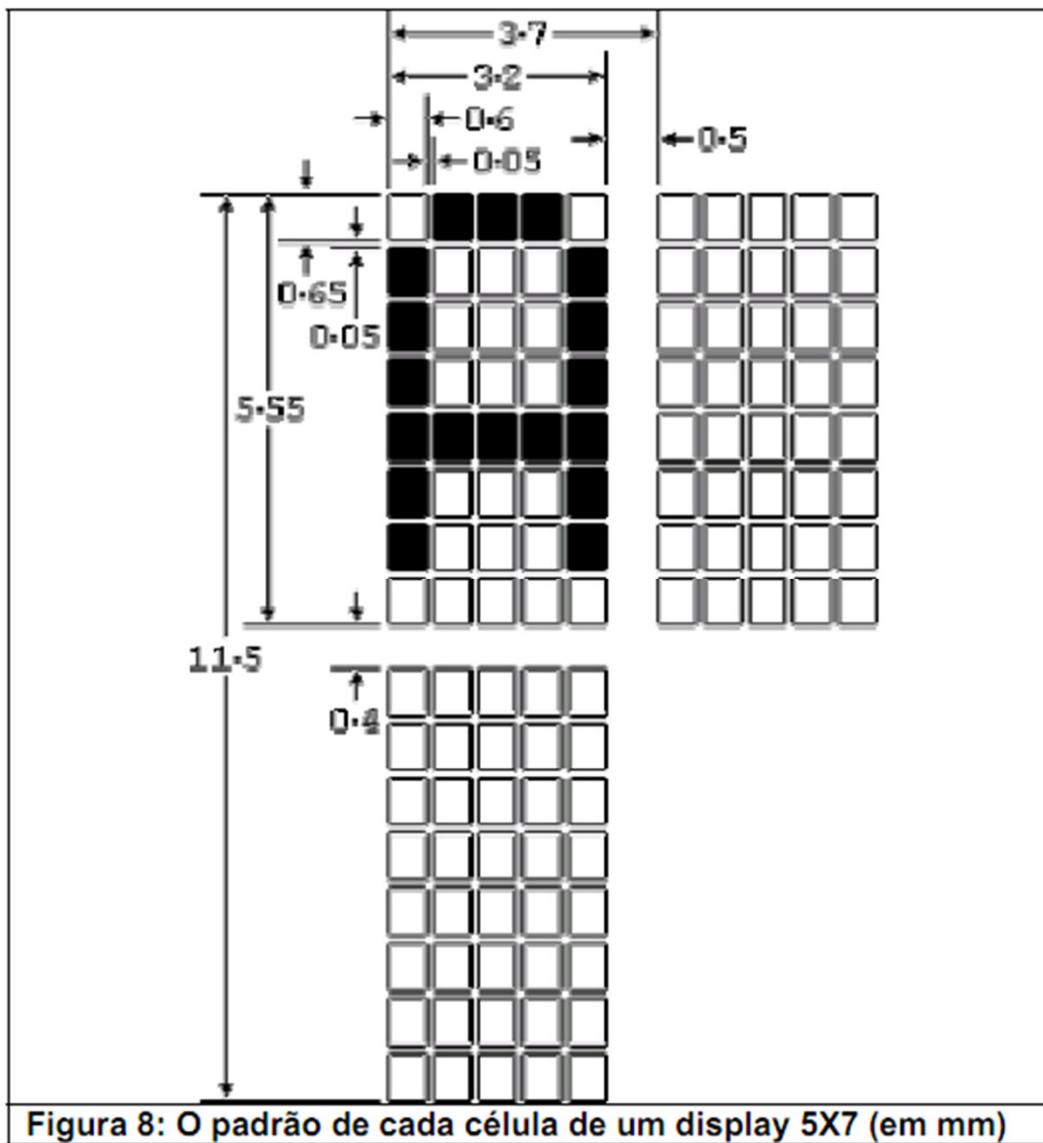
Sobre o Hardware (LCD)

O LCD é formado por “células” onde ficam os caracteres.

Exemplo: display 16x2 – 32 células

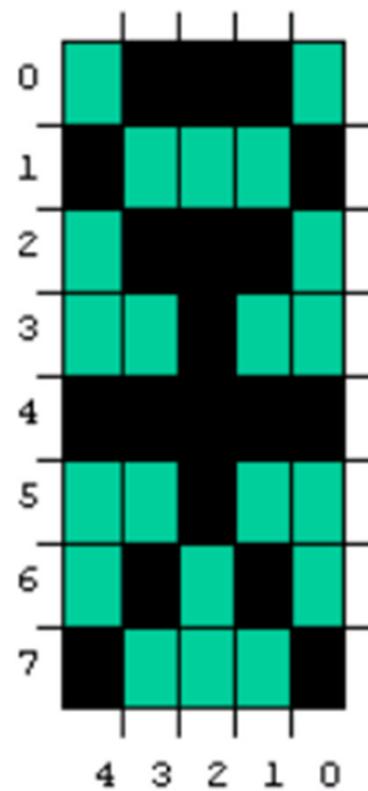
Cada “célula” é composta por uma matriz de 8x5 pontos (pixels).

Sobre o Hardware (LCD)



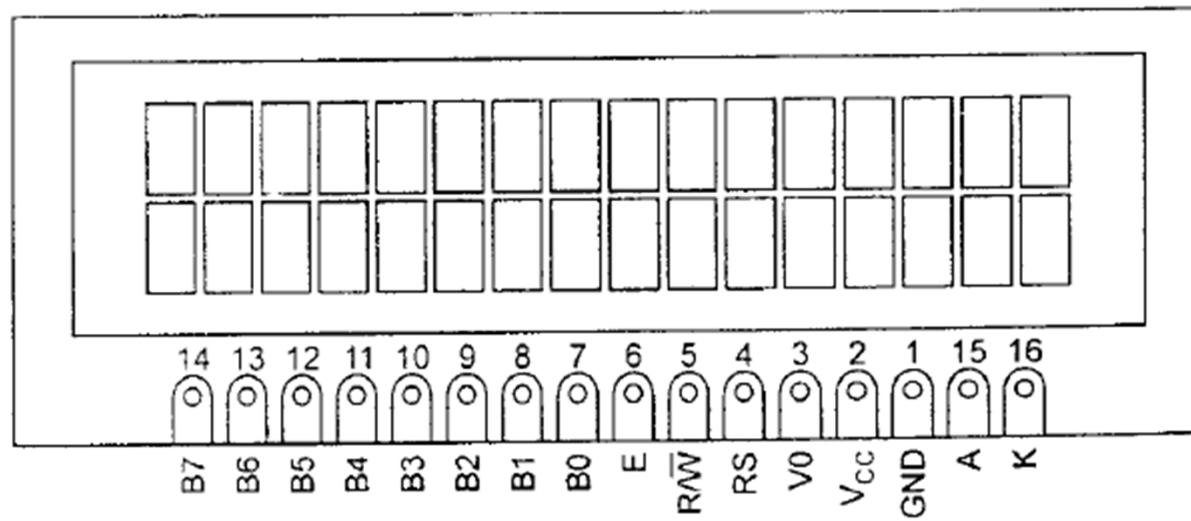
Sobre o Hardware (LCD)

Caractere formado:



Sobre o Hardware (LCD)

Existe uma interface padrão de hardware que todos os fabricantes utilizam. Em geral um LCD possui 14 pinos (quando não tem back-light) e 16 pinos (quando tem back-light).



Funções dos pinos do LCD

Pino	Nome	Função
1	Vss	Terra
2	Vdd	Positivo (normalmente 5V)
3	Vo	Contraste do LCD. Às vezes também é chamado de Vee
4	RS	Register Select
5	R/W	Read/Write
6	E	Enable
7	D0	Bit 0 do dado a ser escrito no LCD (ou lido dele).
8	D1	Bit 1 do dado a ser escrito no LCD (ou lido dele).
9	D2	Bit 2 do dado a ser escrito no LCD (ou lido dele).
10	D3	Bit 3 do dado a ser escrito no LCD (ou lido dele).
11	D4	Bit 4 do dado a ser escrito no LCD (ou lido dele).
12	D5	Bit 5 do dado a ser escrito no LCD (ou lido dele).
13	D6	Bit 6 do dado a ser escrito no LCD (ou lido dele).
14	D7	Bit 7 do dado a ser escrito no LCD (ou lido dele).
15	A	Anodo do back-light (se existir back-light).
16	K	Catodo do back-light (se existir back-light).

Tabela 1: Descrição das funções dos pinos do LCD

Funções dos pinos do LCD

- O pino 3 normalmente não é ligado ao microcontrolador e sim a um potenciômetro;
- O pino 5 possui a função de selecionar o modo de escrita e leitura;

Comunicação

O LCD possui um microcontrolador soldado a sua placa. A comunicação com o LCD é feita através dos pinos de I/O digitais do seu microcontrolador.



Comunicação

Para haver compatibilidade com firmwares que usavam LCDs antigos, existem duas formas de comunicação:

- 1- Enviar um byte (8 bits) por vez com a configuração ou caractere a ser escrito.
- 2- Enviar dois nibbles (4 bits) com a configuração ou caractere a ser escrito.

Comunicação

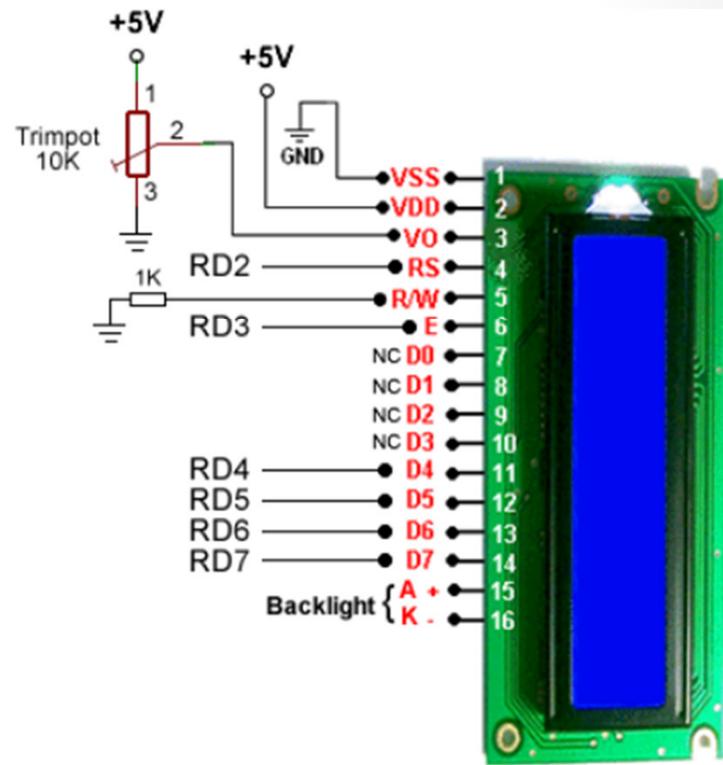
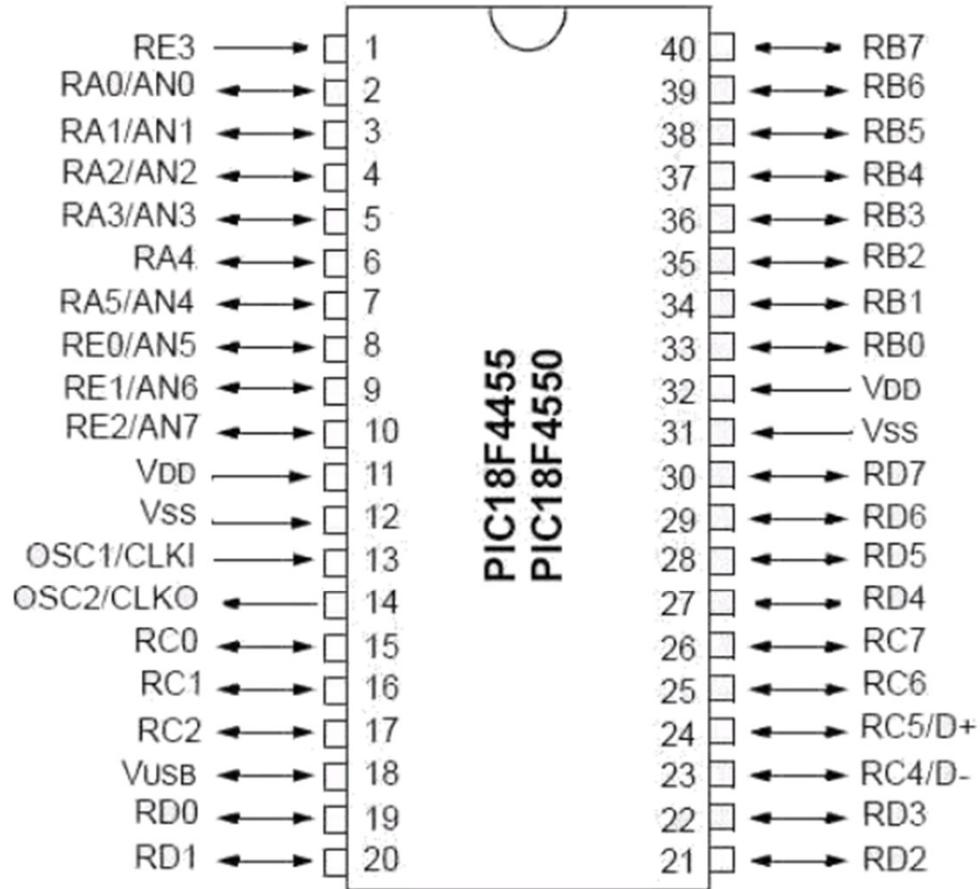
O modo de transferência por 2 nibbles foi feito para diminuir o número de portas usadas do microcontrolador.

Comunicação

Em ambos os casos a comunicação é feita setando os pinos de dados (DB0 a DB7 com byte e DB4 a DB7 com 2 nibbles), o RS e o R/W com 0 e 1 e informando ao controlador que os pinos devem ser lidos pois existe uma configuração ou comando para ser executado.

Este informe é feito elevando o nível do pino de enable de 0 para 1 e retornando-o para 0.

Comunicação



Comunicação

Quando trabalhamos com a comunicação através de 2 nibbles , dividimos o byte que desejamos em 2 nibbles. Então enviamos o nibble mais significativo (fazendo enable = 1 e enable = 0) e então enviamos o nibble menos significativo.

Comunicação

Cada “célula” do display é associada a um endereçamento e a um espaço de memória (de 8 bits). Este endereçamento, permite que escolhamos um lugar específico no display para escrita. O endereçamento começa na primeira linha primeira coluna como 0x00 e vai incrementando a cada caractere. A segunda linha primeira coluna é o 0x40 e cada célula subsequente é incrementado 1.

Comunicação

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

Endereçamento direto para um LCD 16X02

Inicialização do LCD

O LCD precisa ser inicializado. Esta inicialização irá configurá-lo para funcionar com um byte ou dois nibbles e deixá-lo pronto para receber um comando ou configuração.

Inicialização do LCD

A - Comunicação em dois Nibbles:

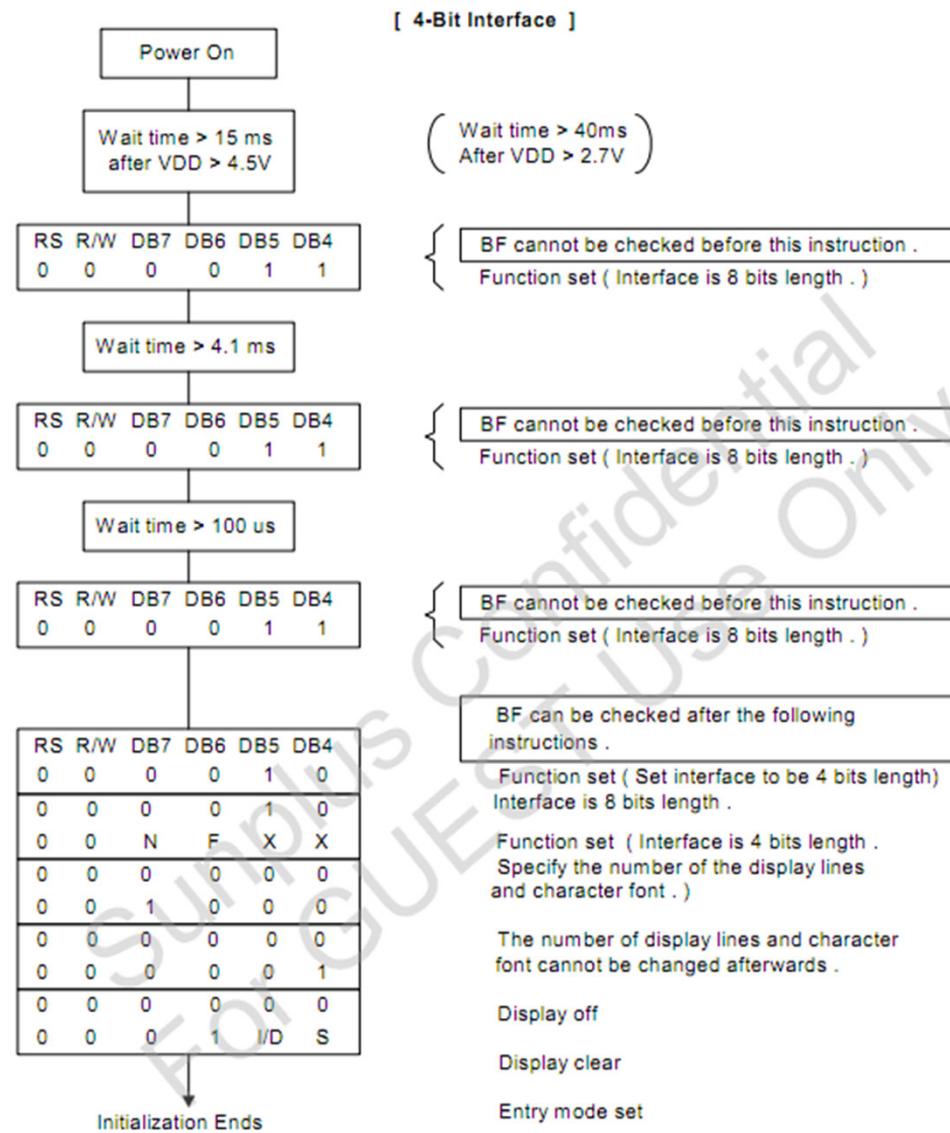
- 1- Após o LCD ser energizado deve esperar um tempo maior que 15 ms.
- 2- Enviar para o LCD: Rs = 0, R/W = 0, DB7 = 0, DB6 = 0, DB5 = 1, DB4 = 1
- 3- Aguardar um tempo maior do que 4,1 milissegundos.
- 4- Enviar para o LCD: Rs = 0, R/W = 0, DB7 = 0, DB6 = 0, DB5 = 1, DB4 = 1
- 5- Aguardar um tempo maior do que 100 microssegundos.
- 6- Enviar para o LCD: Rs = 0, R/W = 0, DB7 = 0, DB6 = 0, DB5 = 1, DB4 = 1
- 7- Enviar para o LCD: Rs = 0, R/W = 0, DB7 = 0, DB6 = 0, DB5 = 1, DB4 = 0
- 8- Enviar configurações de tamanho de caractere.
- 9- Enviar Display off
- 10- Limpar o LCD

Inicialização do LCD

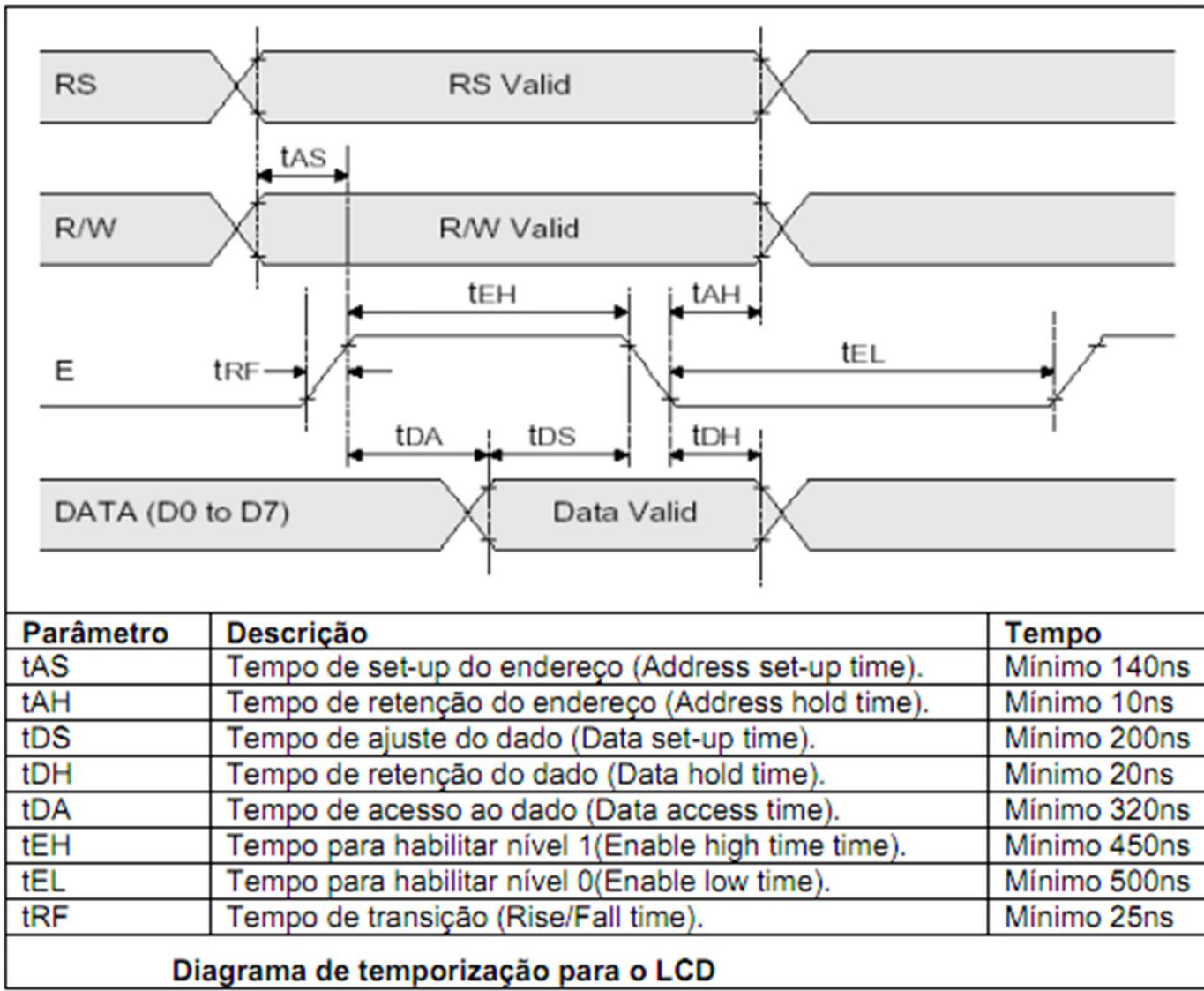
B - Comunicação em um Byte:

- 1- Após o LCD ser energizado deve esperar um tempo maior que 15 ms.
- 2- Enviar para o LCD: Rs = 0, R/W = 0, DB7 = 0, DB6 = 0, DB5 = 1, DB4 = 1, DB3 = X, DB2 = X, DB1 = X, DB0 = X, onde X pode ser tanto 1 quanto 0.
- 3- Aguardar um tempo maior do que 4,1 milissegundos.
- 4- Enviar para o LCD: Rs = 0, R/W = 0, DB7 = 0, DB6 = 0, DB5 = 1, DB4 = 1, DB3 = X, DB2 = X, DB1 = X, DB0 = X, onde X pode ser tanto 1 quanto 0.
- 5- Aguardar um tempo maior do que 100 microssegundos.
- 6- Enviar para o LCD: Rs = 0, R/W = 0, DB7 = 0, DB6 = 0, DB5 = 1, DB4 = 1, DB3 = X, DB2 = X, DB1 = X, DB0 = X, onde X pode ser tanto 1 quanto 0.
- 7- Enviar para o LCD: Rs = 0, R/W = 0, DB7 = 0, DB6 = 0, DB5 = 1, DB4 = 1, DB3 = N, DB2 = F, DB1 = X, DB0 = X. (N e F serão explicados abaixo.)
- 8- Enviar Display off
- 9- Limpar o LCD

Inicialização e transferência em 4 vias



Inicialização - Temporização



Comandos do LCD

O pino RS é quem define se os bits a serem enviados para o LCD serão configurações ou dados. Quando RS = 0 o controlador do LCD interpreta os bits enviados para ele como configurações. Quando RS = 1 o controlador do LCD interpreta os bits como dados e os escreve no display.

Comandos do LCD

Instrução	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0	Descrição	Ciclos de Clock						
NOP	0	0	0	0	0	0	0	0	0	0	Sem efeito	0						
Clear Display	0	0	0	0	0	0	0	0	0	1	Limpa o display e seta contador de endereço para zero.	165						
Cursor Home	0	0	0	0	0	0	0	0	1	x	Seta contador de endereço para zero, retorna o cursor para a posição original. O conteúdo da DD RAM permanece inalterado.	3						
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Seta a direção do movimento do cursor (I/D) e especifica deslocamento automático (S).	3						
Display Control	0	0	0	0	0	0	1	D	C	B	Liga e desliga o Display (D), cursor (C) e ativa cursor piscante, (B).	3						
Cursor / Display shift	0	0	0	0	0	1	S/C	R/L	x	x	Desloca o display ou move o cursor (S/C) e especifica a direção (R/L).	3						
Function Set	0	0	0	0	1	DL	N	F	x	x	Seta número de bits para comunicação (DL), número de linhas (N) e tamanho da fonte (F).	3						
Set CGRAM Address	0	0	0	1	Endereço CGRAM				Seta o endereço da CGRAM. O dado (endereço) é enviado junto.									
Set DDRAM Address	0	0	1	Endereço DDRAM				Seta o endereço da DDRAM. O dado (endereço) é enviado junto.				3						
Busy Flag & Address	0	1	BF	Address Counter (Contador de Endereço)				Flag do Read busy (BF) e contador de endereços.				0						
Write Data	1	0	Dado				Escreve dados na DDRAM ou na CGRAM											
Read Data	1	1	Dado				Lê dados da DDRAM ou da CGRAM											
x : Sem importância	I/D	1	Incrementa Decrementa				R/L	1	Deslocamento para a direita Deslocamento para a esquerda									
	S	1	Deslocamento automático do display				DL	1	Interface de 8 bits Interface de 4 bits									
	D	1	Display ON (ligado) Display OFF (desligado)				N	1	2 linhas 1 linha									
	C	1	Cursor ON (ligado) Cursor OFF (desligado)				F	1	Caracteres 5x10 Caracteres 5x7									
	B	1	Cursor piscando				DDRAM : Display Data RAM CGRAM : Character Generator RAM											
	S/C	1	Deslocamento do display Movimento do Cursor															

Tabela 2: Instruções para inicialização e configuração do LCD

Escrita no LCD

- A escrita no LCD é feita setando RS como 1 e R/W como 0 e o código ASCII do caractere nos dados.

Write Data	1	0	Dado	Escreve dados na DDRAM ou na CGRAM	3
Read Data	1	1	Dado	Lê dados da DDRAM ou da CGRAM	3

Escrita no LCD

Muitas vezes utilizamos o comando *ST CGRAM Address* para informá-lo a posição onde deve ser feita a próxima escrita e, em seguida, enviamos o dado para ser escrito. Os dados são passados de acordo com a tabela abaixo que coincide em grande parte com a tabela ASCII:

Escrita no LCD

	Higher 4-bit (D4 to D7) of Character Code (Hexadecimal)															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	CG RAM (1)															
1	CG RAM (2)															
2	CG RAM (3)															
3	CG RAM (4)															
4	CG RAM (5)															
5	CG RAM (6)															
6	CG RAM (7)															
7	CG RAM (8)															
8	CG RAM (9)															
9	CG RAM (0)															
A	CG RAM (1)															
B	CG RAM (2)															
C	CG RAM (3)															
D	CG RAM (4)															
E	CG RAM (5)															
F	CG RAM (6)															

Escrita no LCD

Como os principais caracteres estão de acordo com a tabela ASCII podemos usar a biblioteca string.h de C para manipular os strings de texto e mandá-los para o LCD sem precisar de nenhum tipo de conversão.

Existem espaços de memória reservados para o usuário desenhar um símbolo que desejar.

Tabela ASCII

Tabela ASCII (*American Standard Code for Information Interchange*)

- Codificação de caracteres de oito bits
- Baseada no alfabeto inglês
- Desenvovida em 1960
- Grande parte da codificação moderna a tem como base.

Tabla ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0 000		NUL (null)	32	20	040	 	Space	64	40	100	@	Ø	96	60	140	`	`
1	1 001		SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2 002		STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3 003		ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4 004		EOT (end of transmission)	36	24	044	$	\$	68	44	104	D	D	100	64	144	d	d
5	5 005		ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6 006		ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7 007		BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8 010		BS (backspace)	40	28	050	((72	48	110	H	H	104	68	150	h	h
9	9 011		TAB (horizontal tab)	41	29	051))	73	49	111	I	I	105	69	151	i	i
10	A 012		LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B 013		VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C 014		FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D 015		CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E 016		SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F 017		SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10 020		DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11 021		DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12 022		DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13 023		DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14 024		DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15 025		NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16 026		SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17 027		ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18 030		CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19 031		EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A 032		SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B 033		ESC (escape)	59	3B	073	;	;	91	5B	133	[[123	7B	173	{	{
28	1C 034		FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|	
29	1D 035		GS (group separator)	61	3D	075	=	=	93	5D	135]]	125	7D	175	}	}
30	1E 036		RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F 037		US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177		DEL

Tabela ASCII

Gerando caracteres em código ascii

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    char c = 32;
    do printf("[%d] %c\n", c, c);
    while (++c <= 126);
    system("Pause");
}
```

Programação

Biblioteca: *Lcd.c* e *Lcd.h*

- Criação de rotinas de delay.
- Função para enviar comandos.
- Função para enviar dados.
- Função de inicialização.
- Função para posicionamento do cursor.

Programação

Icd.h

```
#ifndef _LCD_H_
#define _LCD_H_
//*****  
// RD      7   6   5   4   -   -   -   *  
// Data Bus D7  D6  D5  D4  D3  D2  D1  D0 *  
//-----*  
// RD      2   gnd  3   *  
// Controls RS  RW   E   *  
//*****  
  
#define RS          PORTDbits.RD2  
//#define RW         no kit picminas esta em GND, ou seja Rw=0 => SEMPRE ESCRITA  
#define E          PORTDbits.RD3  
#define Data4       PORTDbits.RD4  
#define Data5       PORTDbits.RD5  
#define Data6       PORTDbits.RD6  
#define Data7       PORTDbits.RD7  
  
void delay_100uS(void) ;  
void delay_500uS(void) ;  
void delay_5mS(void);  
void delay_40mS(void);  
void sendNibble(unsigned char data);  
void sendCmd(unsigned char Cmd);  
void sendData(unsigned char data);  
void initialiseLCD(void);  
void setLCDPosition(unsigned char posAddres);  
#endif // _LCD_H_
```

Programação

Função principal

```
#include "lcd.h"
void main(void)
{
    ConfiguraSistema();           // Configura as portas e periféricos do PIC.
    initialiseLCD();
    sendData('P');
    sendData('I');
    sendData('C');

    while(1);
}
```

Programação

```
void initialiseLCD(void)
{
    //A – Comunicação em 4 vias (usando apenas um Nibble)
    //1– Após o LCD ser energizado deve esperar um tempo maior que 40 ms.
    //2– Enviar para o LCD: Rs = 0, R/W = 0, DB7 = 0, DB6 = 0, DB5 = 1, DB4 = 1
    //3– Aguardar um tempo maior do que 4,1 milissegundos.
    //4– Enviar para o LCD: Rs = 0, R/W = 0, DB7 = 0, DB6 = 0, DB5 = 1, DB4 = 1
    //5– Aguardar um tempo maior do que 100 microssegundos.
    //6– Enviar para o LCD: Rs = 0, R/W = 0, DB7 = 0, DB6 = 0, DB5 = 1, DB4 = 1
    //7– Enviar para o LCD: Rs = 0, R/W = 0, DB7 = 0, DB6 = 0, DB5 = 1, DB4 = 0
    //8– Enviar configurações de tamanho de caractere.
    //9– Enviar Display on
    //10– Limpar o LCD

    delay_mS(40);

    RS = 0;                                // Command mode
    E = 0;
    Data4=0;
    Data5=0;
    Data6=0;
    Data7=0;

    sendNibble(0x03);
    delay_mS(5);
    sendNibble(0x03);
    delay_300uS();
    sendNibble(0x02);
    .
}
```

Programação

```
void sendNibble(unsigned char data)
{
    // Pega apenas os bits de interesse (do nibble menos significativo)
    if(data&0x01) Data4 =1;
    else Data4 = 0;
    if(data&0x02) Data5 =1;
    else Data5 = 0;
    if(data&0x04) Data6 =1;
    else Data6 = 0;
    if(data&0x08) Data7 =1;
    else Data7 = 0;

    delay_200uS();

    // Gera um pulso em enable
    // após colocar os dados na porta, é preciso que o pino E sofra uma transição de high para low
    E = 1;
    delay_500uS();           // tempo para estabilização, conforma datasheet do fabricante
    E = 0;                  // send data to LCD
    delay_500uS();

    return;
}
```

Programação

```
void sendCmd(unsigned char cmd)
{
    RS = 0;          // modo comando => RS=0
    delay_500uS ();
    E = 0;
    sendNibble((cmd>>4)&0x0F);      // envia primeiro o nibble mais significativo
    delay_100uS ();
    sendNibble(cmd&0x0F);           // envia o nibble menos significativo
    delay_100uS ();
    return;
}
```

Programação

```
void sendCmd(unsigned char cmd)
{
    RS = 0;           // modo comando => RS=0
    delay_500uS();
    E = 0;
    sendNibble((cmd>>4)&0x0F);      // envia primeiro o nibble mais significativo
    delay_100uS();
    sendNibble(cmd&0x0F);            // envia o nibble menos significativo
    delay_100uS();
    return;
}
```

Função SendData()

Diferença no estado de RS!

Programação

```
*****
 * Funcao:
 * Entrada:      Nenhuma (void)
 * Saída:        Nenhuma (void)
 * Descrição:    Seta a posicao do cursor do LCD
 *
 * O LCD foi configurado para ter 2 linhas por 16 colunas e cada caractere tem 5x7 pixels.
 * Cada posição possui um endereço equivalente, como abaixo:
 *
 * linha 1 | 00 | 01 | 02 | ... | 0F |
 * linha 2 | 40 | 41 | 42 | ... | 4F |
 *
 * para tal, basta enviar um comando com o bit D7 setado.
 ****/
void setLCDPosition(unsigned char posAddres)
{
    sendCmd(0x80|posAddres);
    return;
}
```

Exercício

(45)

Perguntas

