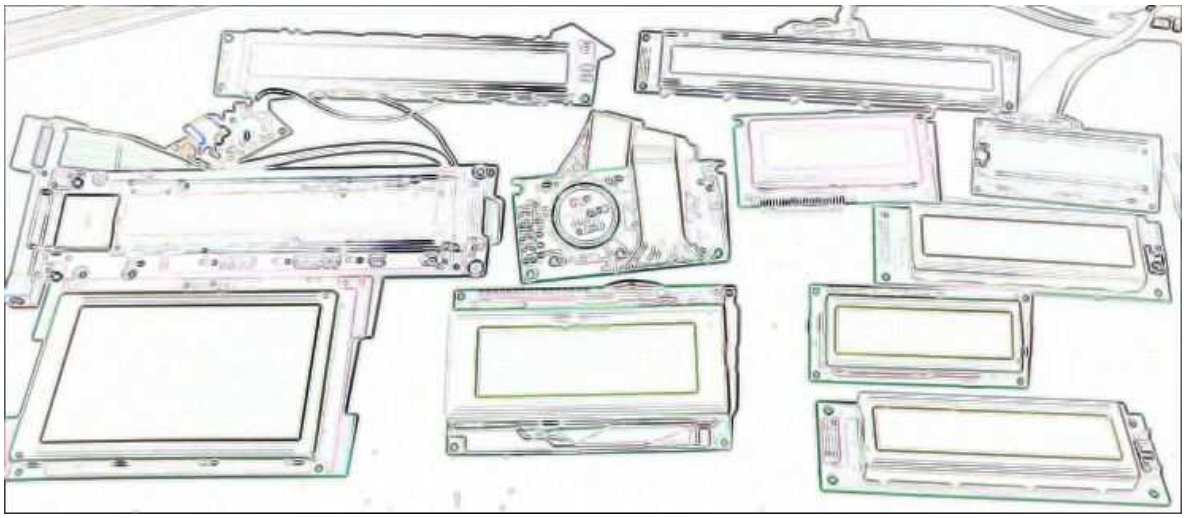


LCD de caracteres (com controlador HD44780) revisão 19.01.2005

Por Renie S. Marquet (reniemarquet@uol.com.br)

Os LCD (Liquid Cristal Display, em português: Display de Cristal Líquido) de caracteres são componentes destinados a transmitir informações ao usuário de forma mais clara e descritiva, podendo até conter mensagens inteiras.

Os LCD apresentam diversas vantagens sobre os display de LED (Light Emiter Diode, em português: Diodo Emissor de Luz):

- Menor consumo de energia.
- Capacidade de informação por centímetro muito maior.
- Menor quantidade de linhas de controle.
- Grande flexibilidade para montagem de mensagens, inclusive possibilitando pequenos desenhos/símbolos.

Para sorte dos usuários e desenvolvedores as industrias de LCD praticamente adotaram o controlador HD44780 como padrão, a maioria das vezes em que o display não possui um HD44780 é utilizado um controlador compatível (ex. KS0066, KS0076, etc.).

Um outro ponto de grande valia é que a utilização dos pinos que interagem com o exterior também está praticamente padronizada como o indicado na figura 1. Deve-se levar em consideração que a padronização das linhas não significa que todos os conectores de LCD são exatamente iguais, pelo contrário, existem muitas variações (figura 3), por exemplo:

- Pinos dispostos em uma única fileira na horizontal cuja numeração começa da esquerda para a direita.
- Pinos dispostos em uma única fileira na horizontal cuja numeração começa da direita para esquerda.
- Pinos dispostos em 2 fileiras na vertical cuja numeração começa de cima para baixo.
- Pinos dispostos em 2 fileiras na vertical cuja numeração começa de baixo para cima.

Os modelos com backlight (luz de fundo, geralmente um conjunto de LED's especiais) possuem ainda mais 2 pinos para a ligação do mesmo. Estes 2 pinos adicionais (15 e 16) podem estar dispostos fisicamente no conector antes do pino 1 ou depois do pino 14.

Figura 1.

Pino	Nome	Função
1	GND	Alimentação (terra)
2	VCC	Alimentação (+ 5V)
3	V0	Tensão de ajuste do contraste
4	RS	1 = Dado 0 =Instrução
5	R/W	1 = Leitura 0 = Escrita
6	CS	Chip Select ; sinal Enable
7	D0	Barramento de Dados
8	D1	
9	D2	
10	D3	
11	D4	
12	D5	
13	D6	
14	D7	

Como identificar os pinos do conector:

A melhor fonte de informação a respeito de um LCD é o datasheet (folha de dados) do mesmo, não só para identificar o posicionamento dos pinos, como também informações importantes como consumo, alimentação, tipo de backlight , tempos, voltagens, etc.

No caso de não dispor do datasheet, identificar os pinos do LCD (com controlador HD44780 ou compatível) não é muito difícil, bastando seguir os passos abaixo:

1 – Para descobrir quais são os pinos do backlight basta seguir as vias mais extremas do conector para identificar quais vão para o mesmo. Alguns LCD apesar de não terem backlight, suas placas foram projetadas com opção para o mesmo, neste caso existem os pinos para o backlight no conector apesar dele não estar presente (veja figura 2).

2 – Para identificar o pino 1 (terra), utilize um multímetro (quando não for possível a identificação visual) e localize o pino que está ligado a(s) ilha(s) onde as orelhas de fixação da carcaça metálica do display estão dobradas. Deve-se prestar atenção pois algumas ilhas podem não estar conectadas a nada (normalmente, no mínimo uma ilha estará conectada ao terra). O teste deve ser feito com as ilhas visto que alguns LCD tem a carcaça metálica pintada, deixando-as isoladas .

3 – Tendo-se identificado o pino 1, os demais virão na seqüência um ao lado do outro no caso de conectores de uma única fileira de pinos, e no caso de conectores de 2 fileiras, os demais também virão na seqüência porém com os pinos pares em uma fileira e os impares na outra (veja figura 3) .

Figura 2

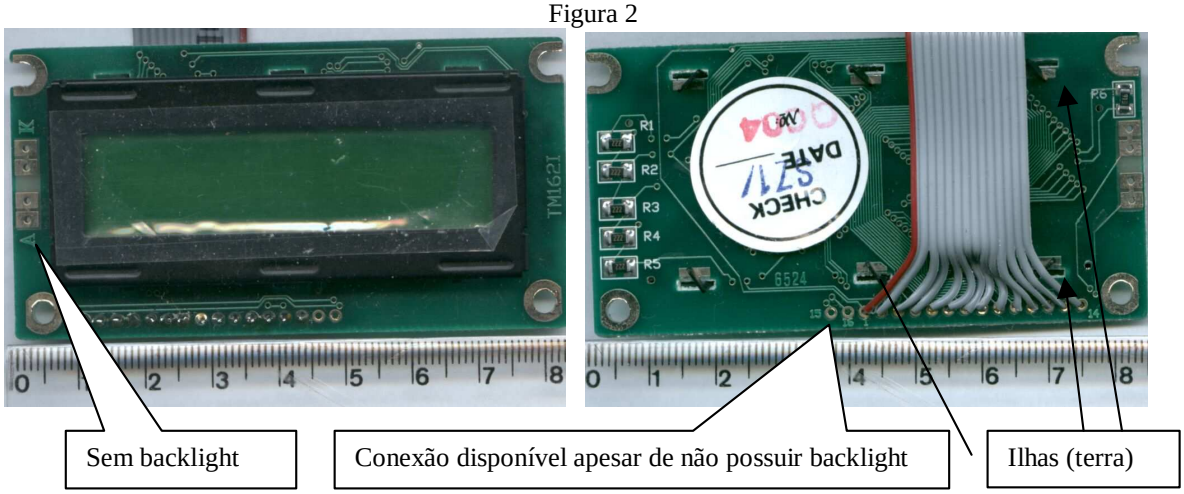
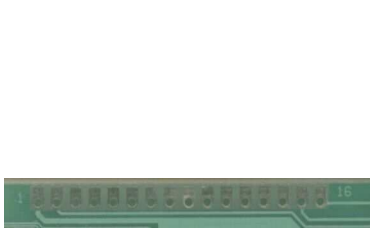
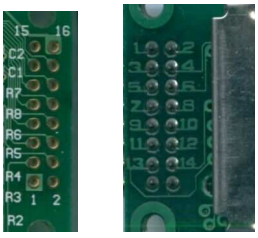


Figura 3 . Exemplos de conectores



1 fileira, pinos de 1 a16

1 fileira, pinos de 14 a 1, 15 e 16 (não possui numeração na placa)



2 fileiras de baixo p/cima e 2 fileiras de cima p/baixo

Notas: Apesar de algumas fotos parecerem estar de cabeça para baixo, todas as figuras e fotos apresentadas neste artigo estão na posição correta de utilização do display, ou seja, o display está de cabeça para cima.

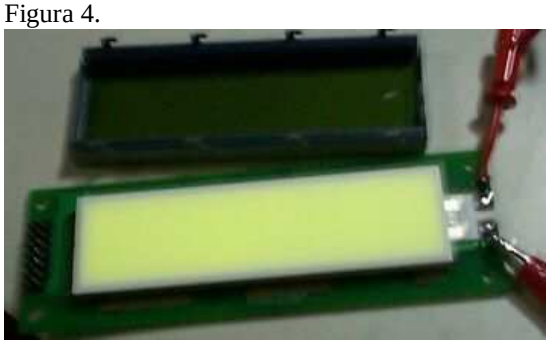
Ao visualizar este artigo no computador, o leitor pode utilizar o Zoom para ampliar as fotos e observar maiores detalhes dos LCD.

As fotos dos LCD foram tiradas com uma régua junto, proporcionando identificar as dimensões dos LCD.

Backlight

Existem no mercado diversos modelos de LCD com backlight incorporado. O backlight (luz de fundo) é uma fonte de luz posicionada atrás do cristal do display de modo a facilitar a visualização das informações apresentadas em locais de pouca ou mesmo nenhuma iluminação. Apesar de existirem vários tipos de backlight (EL,CFL, LED, etc.) , nos LCD de caracteres o tipo mais comum encontrado é o backlight de LED (veja figura 4) assim, este tópico abrangerá somente este tipo.

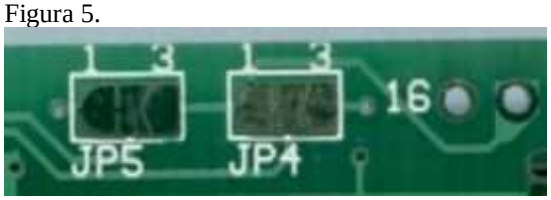
Figura 4.



A maioria dos LCD com backlight apresentam em seu conector mais 2 pinos destinados a alimentação do mesmo. Estes pinos podem estar posicionados antes ou depois dos pinos que compõem as linhas de controle e dados normais do display.

Algumas placas disponibilizam ao usuário ilhas de conexão entre os pinos do conector e o LED do backlight (veja figura 5), deste modo o usuário tem a opção de escolher em qual pino (15 ou 16) será ligado ao Anodo (positivo) e ao Katodo (negativo) do LED. Estas ilhas de conexão podem vir pré-configuradas de fábrica ou não, uma aferição visual é o bastante para identificar se existe pré-configuração.

Figura 5.

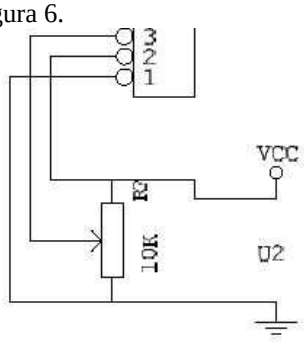


No caso do LCD ou seu LED de backlight não apresentar inscrição identificando o Anodo (positivo) nem o Katodo (negativo) e não se dispor do datasheet, é possível identificá-los por meio de experimento. Com uma fonte de 5V através de um resistor limitador de 68 a 100 ohms, se ao conectar o LED, este não acender, deve-se inverter as ligações. Após identificar a polaridade do backlight, caso a luminosidade não seja satisfatória, valores menores do resistor limitador podem ser experimentados com cuidado para não sobrecarregar o LED e queimá-lo.

Teste básico

É possível efetuar um teste simples para verificar se o LCD está funcionando. Para o teste básico é necessário apenas uma fonte de 5V, um trimpot (ou potenciômetro) de 5K ohm a 20K ohm e fios para as ligações (figura 6).

Figura 6.



Ao ligar-se a fonte de alimentação, o display começa um processo de auto teste preenchendo a linha 1 com blocos cheios ( ) permitindo assim identificar o bom funcionamento do LCD (figura7). Para alterar o contraste o usuário dever variar a posição do trimpot.

Figura 7.

Entendendo as linhas de controle

Para controlar os LCD são utilizadas 3 linhas descritas abaixo.

RS : Seleciona o tipo de informação que será passada ou recebida do LCD, 0 = Instrução, 1 = Dado.

R/W : Informa ao LCD o tipo da próxima ação a ser executada, 0 = Write (escrita), 1 = Read (leitura).

E : (Enable) habilita o LCD a executar a ação desejada (leitura ou escrita).

Passos para enviar um comando (veja o tópico Comandos mais a frente) para o display:

- Colocar o comando desejado nas linhas **D0** a **D7** (no modo 4 bits são usadas somente as linhas **D4** a **D7**, assim, o comando precisa ser enviado em 2 partes) .
- Colocar a linha **RS** em nível baixo.
- Colocar a linha **R/W** em nível baixo.
- Gerar um pulso na linha **E** (Enable). A linha **E** deve normalmente ser mantida em nível baixo, para gerar um pulso, coloca-se a linha em nível alto e retorna-se a nível baixo. O controlador receberá o conteúdo das linhas **D0** a **D7** na transição da linha **E** para nível baixo.

Passos para enviar um dado para o display:

- Colocar o dado desejado nas linhas **D0** a **D7** (no modo 4 bits são usadas somente as linhas **D4** a **D7**, assim, o dado precisa ser enviado em 2 partes) .
- Colocar a linha **RS** em nível alto.
- Colocar a linha **R/W** em nível baixo.
- Gerar um pulso na linha **E** (Enable). A linha **E** deve normalmente ser mantida em nível baixo, para gerar um pulso, coloca-se a linha em nível alto e retorna-se a nível baixo. O controlador receberá o conteúdo das linhas **D0** a **D7** na transição da linha **E** para nível baixo.

Passos para ler o flag Busy (ocupado) e o conteúdo atual do contador de endereço :

- Colocar a linha **RS** em nível baixo.
- Colocar a linha **R/W** em nível alto.
- Colocar a linha **E** em nível alto.
- Ler o conteúdo das linhas **D0** a **D7** (o bit **D7** conterá o flag BUSY, os bits **D0** a **D6** conterão o endereço atual em que o cursor está posicionado.

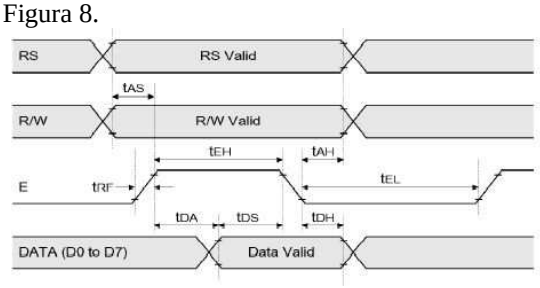
Passos para ler o dado contido no endereço atual apontado pelo contador de endereço:

- Se necessário, enviar comando posicionando o contador de endereço para o local da RAM do display a ser lido.
- Colocar a linha **RS** em nível alto.
- Colocar a linha **R/W** em nível alto.
- Colocar a linha **E** (Enable) em nível alto.
- Ler o conteúdo das linhas **D0** a **D7** (no modo 4 bits, só serão usadas as linhas **D4** a **D7** e duas leituras serão necessárias para receber os 2 nibles (1/2 byte) que compõem o dado a ser lido).

Considerações sobre tempos

A ativação das linhas de controle do LCD deve obedecer tempos mínimos para que os circuitos internos se estabilizem (figura 8).

Figura 8.



Com base no diagrama da figura 8 temos:

tAS = tempo necessário para se definir as linhas RS e R/W antes do sinal Enable ser colocado em nível alto, mínimo 140 ns.
tAH= tempo necessário após a linha Enable passar para nível baixo em que as linhas RS e R/W devem ser mantidas estáveis, mínimo 10 ns.
tDS = tempo mínimo para que o conteúdo das linhas D0 a D7 estabilizem, mínimo 200 ns.
tDA = tempo máximo necessário para que o conteúdo das linhas D0 a D7 seja válido para o controlador, máximo 320 ns.
tDH = tempo mínimo para que o conteúdo das linhas D0 a D7 seja mantido estável antes da linha Enable ser passada para nível baixo, mínimo 20 ns.
tEH = tempo mínimo que o sinal Enable deve permanecer em nível alto, mínimo 450 ns.

tEL = tempo mínimo que o sinal Enable deve permanecer em nível baixo, mínimo 500 ns.
tRF = tempo máximo para que o sinal Enable estando em nível baixo atinja o nível alto e vice-versa.

A execução dos comandos solicitados ao LCD também consomem tempo. Qualquer comando enviado ao LCD antes da finalização do comando anterior será ignorado, o que pode causar resultados totalmente adversos aos desejados.
Abaixo é apresentada uma relação de tempos máximos para que o controlador execute os comandos, porém o desenvolvedor deve levar em consideração que esses tempos podem variar em função da temperatura, voltagem de operação e principalmente se o controlador não for um HD44780 e sim um compatível.

Limpar display	82 µs a 1,64 ms
Mover display e cursor para posição Home	40 µs a 1,64 ms
Ligar/Desligar display e cursor	40 µs
Deslocar display/cursor	40 µs
Funções de definição	40 µs
Def. endereço CGRAM	40 µs
Def. endereço DDRAM	40 µs
Escrever dados	40 µs
Ler dados	40 µs
Def. modo de entrada de caracteres	40 µs
Ler status	1 µs

O desenvolvedor deve prestar bastante atenção quanto aos tempos tanto nas definições das linhas de controle quanto os consumidos nos comandos enviados ao display. No caso dos tempos mínimos não serem respeitados o display pode até mesmo não inicializar.

Comandos

Para que o controlador interprete o conteúdo das linhas D0 a D7 como um comando a linha RS tem que estar em nível baixo.
Os marcas (*) identificam os valores padrão assumidos pelo controlador ao inicializar.

Limpar display

D7	D6	D5	D4	D3	D2	D1	D0	hex
0	0	0	0	0	0	0	1	01

Limpa a tela do display e posiciona o cursor na primeira posição da tela.

Display e Cursor Home

D7	D6	D5	D4	D3	D2	D1	D0	hex
0	0	0	0	0	0	1	X	01 a 03

x = não importa

Posiciona o cursor e o display (caso tenha sido deslocado) na posição inicial.

Modo de entrada de caracteres

D7	D6	D5	D4	D3	D2	D1	D0	hex
0	0	0	0	0	1	I/D	S	04 a 07

Define o comportamento do display a cada entrada de caracter.
I/D : 1 = incrementa o contador de endereço após a entrada (*).
0 = decrementa o contador de endereço após a entrada.
S : 1 = desloca o conteúdo anterior (os novos caracteres são inceridos).
0 = sobrepõe o conteúdo anterior (*).

Liga/Desliga Cursor e Display

D7	D6	D5	D4	D3	D2	D1	D0	hex
0	0	0	0	1	D	U	B	08 a 0F

Define as características do cursor e liga/desliga a tela do display.

D : 1 = Display ligado.
0 = Display desligado, o conteúdo da DDRAM não é alterado, apenas a tela é desligada.
U : 1 = cursor ligado.
0 = cursor desligado (*)
B : 1 = cursor intermitente.
0 = intermitência do cursor desligada (*).

Deslocamento Display/Cursor

D7	D6	D5	D4	D3	D2	D1	D0	hex
0	0	0	1	D/C	D/E	x	x	10 a 1F

x = não importa
Desloca o cursor ou o display. Deslocar o display tem o efeito de deslocar a área de visão da tela em relação a DDRAM. Exemplo: Normalmente a primeira coluna da primeira linha apresenta o conteúdo do endereço 0x00 da DDRAM e as demais colunas os endereços seguintes, após um comando de deslocamento do display para a direita a primeira coluna passará a mostrar o conteúdo do endereço 0x01, a segunda coluna o endereço 0x02 e daí por diante.

D/C : 1 = desloca a área de visão do display. O conteúdo da DDRAM não é alterado (figura xx). Como o cursor permanece no endereço apontado

pelo contador de endereço, ocorre a ilusão de deslocamento do cursor também.

0 = desloca somente o cursor.

D/E : 1 = deslocamento para a direita.

0 = deslocamento para a esquerda .

Figura xx.
Display de 16 colunas por 2 linhas após 1 deslocamento do display para a esquerda.

Posição	1	2	3	4	5	6	...	16	17	...	40
Endereço	01	02	03	04	05	06	...	10	11	...	00
DDRAM	41	42	43	44	45	46	...	50	51	...	40
(hex.)	Área visível								Não visível		

Display de 16 colunas por 2 linhas após 1 deslocamento do display para a direita.

Posição	1	2	3	4	5	6	...	16	17	..	40
Endereço	27	00	01	02	03	04	...	0E	0F	..	26
DDRAM	67	40	41	42	43	44	...	4E	4F	..	66
(hex.)	Área visível								Não visível		

Definição de funcionamento

D7	D6	D5	D4	D3	D2	D1	D0	hex
0	0	1	8/4	2/1	10/7	X	x	20 a 3F

Define o modo de comunicação, quantidade de linhas e padrão de caracteres.

8/4 : 1 = modo de comunicação por 8 bits (*).

0 = modo de comunicação por 4 bits.

2/1 : 1 = modo 2 linhas.

0 = modo 1 linha (*).

10/7 : 1 = padrão de caracteres no formato 5 x 10 pontos (o bit 2/1 precisa estar com 0 – modo 1 linha , nos displays de uma única linha física a parte de baixo dos caracteres não serão visíveis).

0 = padrão de caracteres no formato 5 x 7 pontos (*).

Nota: os formatos 5x7 e 5x10, na realidade são 5x8 e 5x11, a última linha do caracter é utilizada para apresentar o cursor.

Define endereço da CGRAM

D7	D6	D5	D4	D3	D2	D1	D0	hex
0	1	A	A	A	A	A	A	40 a 7F

Carrega o contador de endereço para um endereço na CGRAM (memória do gerador de caracteres).

AAAAAA : endereço da CGRAM. A CGRAM dispõem de 64 bytes possibilitando ao desenvolvedor definir o formato de até 8 caracteres quando no padrão 5x7 ou 4 caracteres quando definido o padrão 5x10.

Define endereço da DDRAM

D7	D6	D5	D4	D3	D2	D1	D0	hex
1	A	A	A	A	A	A	A	80 a FF

Carrega o contador de endereço para um endereço na DDRAM (memória de dados do display)
AAAAAAA : endereço da DDRAM. A DDRAM dispõem de 80 bytes que contém os dados que são apresentados na tela do display. Dependendo do tamanho físico do display (linhas/colunas), apenas uma parte dos dados da DDRAM ficam visíveis.

CGRAM (RAM do gerador de caracteres)

O HD44780 possui 64 bytes de memória RAM localizados no início do mapa de caracteres possibilitando o desenvolvedor criar caracteres próprios (ex. caracter acentuado), símbolos ou pequenos gráficos.

No formato de caracteres 5x7 cada caracter ocupa 8 bytes consecutivos, assim, é possível criar-se até 8 caracteres/gráficos próprios. Cada linha do novo caracter/gráfico é composto por um byte e somente os 5 bits menos significativos (**D4** a **D0**) são visíveis (exemplo na figura 9).

Para programar o caracter especial é necessário enviar um comando de escrita para posicionar o ponteiro na CGRAM, em seguida escrever 8 bytes de dados que irão compor o caracter/gráfico. Seqüência Para montar o boneco do exemplo:

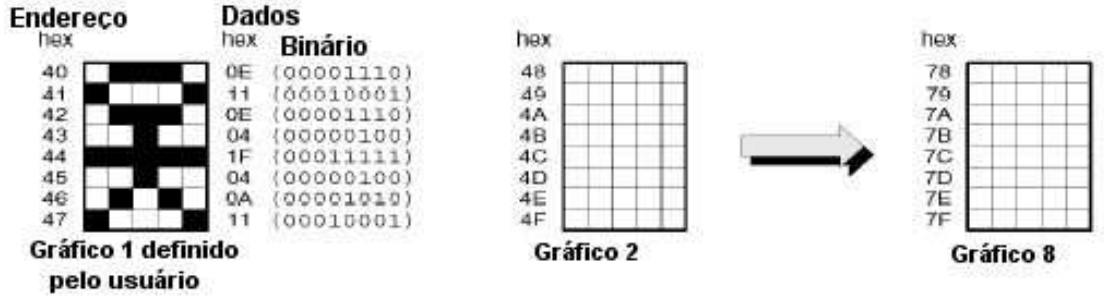
RS=0 (comando) , **R/W**=0 (escrita)
D7 a **D0** = 0x40, pulso em **E**nable
RS=1 (dado) , **R/W**=0 (escrita)
D7 a **D0** = 0x0E, pulso em **E**nable
D7 a **D0** = 0x11, pulso em **E**nable
D7 a **D0** = 0x0E, pulso em **E**nable
D7 a **D0** = 0x04, pulso em **E**nable
D7 a **D0** = 0x1F, pulso em **E**nable
D7 a **D0** = 0x04, pulso em **E**nable
D7 a **D0** = 0x04, pulso em **E**nable
D7 a **D0** = 0x11, pulso em **E**nable

Seqüência para apresentar o caracter criado:

RS=0 (comando) , **R/W**=0 (escrita)
D7 a **D0** = 0x80, pulso em **E**nable
RS=1 (dado) , **R/W**=0 (escrita)
D7 a **D0** = 0x00, pulso em **E**nable

No formato 5x10 apesar de cada caracter ocupar 16 bytes consecutivos, apenas os primeiros 11 serão utilizados para apresentação na tela.

Figura 9.



DDRAM (RAM de dados do display)

O HD44780 possui uma memória de 80 bytes destinados aos dados que serão apresentados na tela do display. Estes 80 bytes estão divididos fisicamente em 2 blocos de 40 bytes cada, o primeiro começando na posição 0x00 e terminando em 0x27, o segundo bloco tem início no endereço 0x40 e termina em 0x67. Esta memória é disponibilizada de forma circular, ou seja, escrevendo-se seguidamente sem alterar o contador de endereço (o contador é incrementado/decrementado automaticamente, dependendo de como foi configurado na inicialização), após escrever um dado na última posição do bloco atual, o contador automaticamente passará a apontar para a primeira posição do bloco seguinte e vice-versa.

Como o comando para carregar o contador de endereços com um endereço da DDRAM é 0x80, para obter-se o valor a ser passado nas linhas D7 a D0 no comando de escrita, soma-se 0x80 com a posição desejada (lembrando que a primeira posição da linha 1 é 0 e não 1), assim, para carregar o contador de endereços para apontar para a quarta coluna da primeira linha, coloca-se o valor 0x83 nas linhas D7 a D0 (0x80 + 0x03).

Caso não seja utilizado o comando de deslocamento do display, os endereços da DDRAM que não sejam visíveis na tela podem ser aproveitados como memória auxiliar para guardar dados temporários.

A primeira linha física do display é relacionada ao primeiro bloco da DDRAM e a segunda linha ao segundo bloco (será explicado mais adiante). Quando o display dispor de uma tela de mais de 2 linhas físicas , o endereço da DDRAM seguinte a última posição visível da primeira linha é utilizado como primeiro endereço da terceira linha, o mesmo ocorrendo com a segunda linha em relação a quarta linha (veja figura 10).

Figura 10.
Display de 16 colunas por 2 linhas

Posição	1	2	3	4	5	6	...	16	17	...	40
Endereço	00	01	02	03	04	05	...	0F	10	...	27
DDRAM	40	41	42	43	44	45	...	4F	50	...	67
(hex.)	Área visível								Não visível		

Display de 16 colunas por 4 linhas

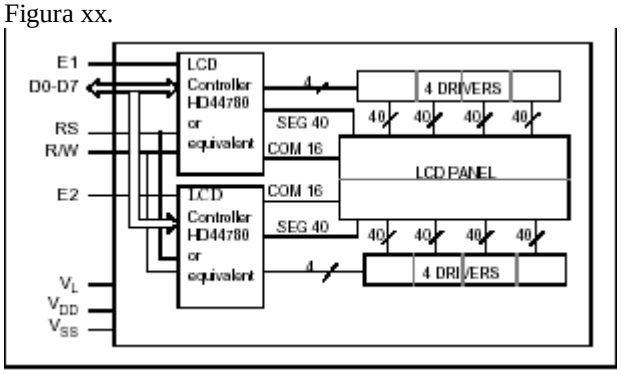
Posição	1	2	3	4	5	6	...	16	33	...	40
	00	01	02	03	04	05	...	0F			
Endereço	40	41	42	43	44	45	...	4F			
DDRAM	10	11	12	13	14	15	...	1F	20	...	27
(hex.)	50	51	52	53	54	55	...	5F	60	...	67
	Área visível								Não visível		

A razão da terceira e quarta linha na maioria dos displays de 4 linhas serem as continuações dos endereços da primeira e da segunda linha deve-se ao fato de que o controlador HD44780 possuir 16 vias para controle dos pontos na horizontal (comum) e 40 vias para controle dos pontos na vertical (segmento), como os caracteres são apresentados como uma matriz de 5x8 pontos, um HD44780 pode controlar sozinho uma tela de até 8 colunas por 2 linhas (40/5 = 8 colunas, 16/8 = 2 linhas) . Como o HD44780 possui 80 bytes para a DDRAM, com o acréscimo de circuitos auxiliares conhecidos como drivers de segmentos (colunas de pontos da tela), por exemplo o HD44100, é possível aos fabricantes produzirem displays com qualquer combinação linhas x colunas contanto que não ultrapasse 80 caracteres (ex: 16x4, 40x2, 20x4, etc.) .

Displays com mais de 80 caracteres (exemplo: 4x40) são construídos com controladores especiais as vezes compatíveis com os comandos do HD44780 ou mesmo com 2 controladores HD44780 e alguns componentes auxiliares

(responsáveis pelo chaveamento entre um controlador e o outro quando necessário), em ambos os casos o LCD possuirá linhas de controles adicionais para ter acesso as áreas extras (figura xx).

Figura xx.



Enquanto na CGRAM e CGROM é representado o mapa de caracteres em conjuntos de 8 bytes que compõem um caracter, e que cada bit dos bytes representam um ponto no desenho do caracter (somente os 5 bits de menor ordem são visíveis), na DDRAM cada endereço (que contém um byte) guarda um índice para a imagem do caracter (normalmente chamado de código do caracter), assim, para apresentar um determinado caracter é necessário gravar o código no endereço da tela (DDRAM). Por exemplo: para mostrar a letra "A" na primeira posição da tela grava-se o código 0x41 (65 em decimal, 0100001 em binário) no endereço 0x00 da DDRAM.

O mapa de caracteres é composto da CGRAM + CGROM. Os códigos 0x00 até 0x0F apontam para os caracteres contidos na CGRAM enquanto os códigos 0x10 a 0xFF apontam para a CGROM. Apesar dos códigos 0x00 a 0x0F corresponderem aos caracteres da CGRAM, apenas os 8 primeiros estão fisicamente implementados (0x00 a 0x07), os códigos 0x08 a 0x0F apontam para os mesmos endereços que os 8 primeiros (0x08=0x00, 0x09=0x02, etc.).

O HD44780 pode ser encontrado em 3 versões de mapa de caracter da CGROM:

- Fonte de caracteres padrão Europeu.
- Fonte de caracteres padrão Japonês , que são os mais comuns de serem encontrados no comércio brasileiro (figura 11).
- Fonte customizada, ou seja, no formato encomendado pelo cliente (fabricante do módulo do LCD).

Figura 11.

Inicialização do LCD

O controlador HD44780 ao ser alimentado efetua um processo interno de inicialização, sua configuração inicial está assinalada no tópico Comandos com (*).

A inicialização do HD44780 pode falhar em virtude das linhas de controle internas e externas atingirem seus níveis de voltagens em tempos fora das especificações do fabricante por motivos diversos, como fonte, temperatura ambiente etc.

Para garantir o funcionamento adequado do LCD é aconselhável ao desenvolvedor efetuar os procedimentos de inicialização mesmo que vá utilizar as configurações default (comunicação 8 bits, caracteres 5 x 7, etc.).

A seguir serão apresentados os fluxos de inicialização tanto no modo de comunicação por 8 bits quanto por 4 bits sugeridos pela Hitachi (fabricante do HD44780).

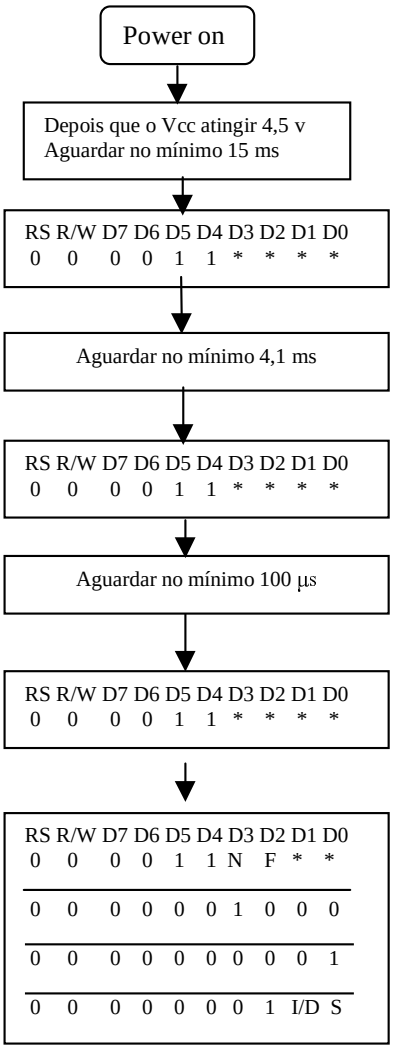
Mesmo que o processo de inicialização interna do controlador não tenha sido completado com sucesso, o modo de comunicação inicial é sempre por 8 bits.

Em virtude do tempo necessário para estabilizar a alimentação do circuito onde o LCD será utilizado depender muito do tipo e da qualidade da fonte, é aconselhável aguardar um tempo entre 100 ms a 200 ms após o circuito ser ligado para começar o processo de inicialização do LCD.

Vale lembrar que para cada envio de dados ou comandos para o LCD, é necessário um pulso na linha **E**nable.

O flag BUSY, o qual indica que o controlador está ocupado efetuando alguma tarefa internamente, não está disponível durante os primeiros passos da inicialização do LCD, assim, é de suma importância respeitar os tempos mínimos indicados pelo fabricante para cada passo.

Acesso no modo 8 bits



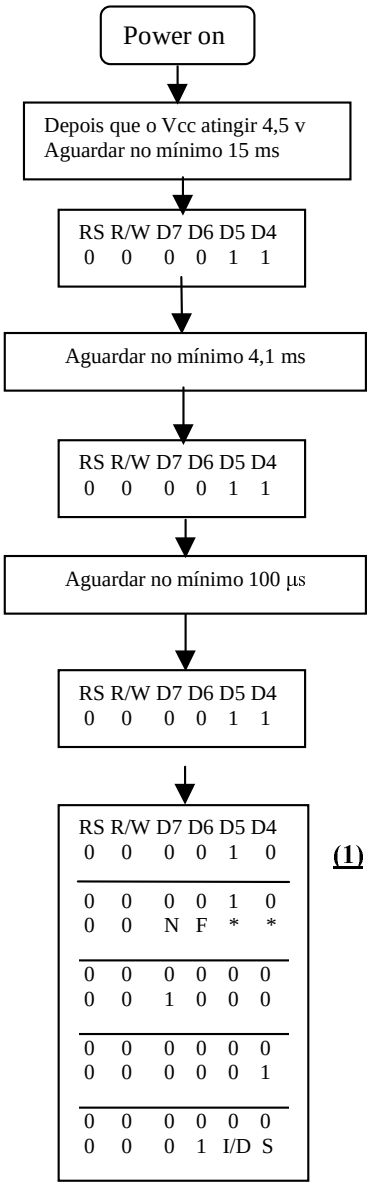
(1)

* = Não importa
N = Nr. de linhas (0=1linha, 1=2linhas)
F = Formato da fonte (0=5x7, 1=5x10).
I/D= Contador de endereço após entrada de caracter (0=decrementa, 1= incrementa).
S = Modo de entrada (0=sobrepõe dado anterior, 1=desloca dado anterior).

O flag BUSY só será válido após o comando marcado com **(1)** no fluxo.

Acesso no modo 4 bits

No modo de comunicação de 4 bits só serão utilizadas as linhas **D7** a **D4**, o HD44780 possui resistores pull-down nas linhas **D7** a **D0**, assim as linhas **D3** a **D0** podem ser deixadas desconectadas .



(1)

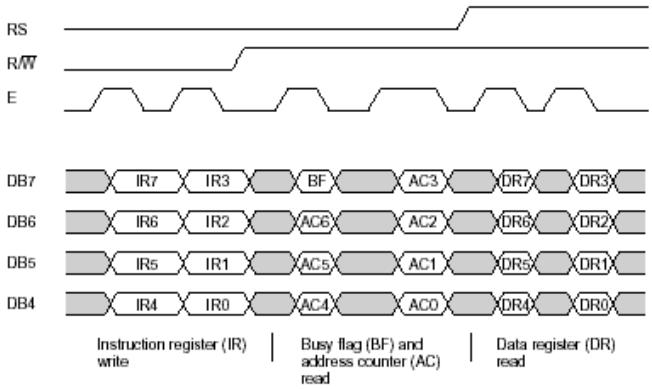
* = Não importa
N = Nr. de linhas (0=1linha, 1=2linhas)
F = Formato da fonte (0=5x7, 1=5x10).
I/D= Contador de endereço após entrada de caracter (0=decrementa, 1= incrementa).
S = Modo de entrada (0=sobrepõe dado anterior, 1=desloca dado anterior).

O flag BUSY só será válido após o comando marcado com **(1)** no fluxo.

Até o comando marcado com **(1)** o LCD ainda está no modo de comunicação de 8 bits.

Como no modo 4 bits só são utilizadas 4 linhas de dados, os comandos e dados a serem passados para o LCD precisam ser divididos em 2 nibles (1/2 byte = nible), sendo enviado o nible alto primeiro e depois o nible baixo. O mesmo também é válido para a leitura, o controlador disponibiliza primeiro o nible alto e na leitura seguinte o nible baixo. (figura 12)

Figura 12.



O modo de comunicação de 4 bits traz algumas desvantagens, como programação mais trabalhosa e comunicação mais lenta devido a necessidade de 2 acessos para cada operação , mas, uma grande vantagem é a necessidade de menos linhas de I/O para comunicação entre o processador (microprocessador, microcontrolador, etc.) e o LCD.

Uma alternativa para economizar mais uma linha de controle é ligar a linha R/W do LCD direto ao terra, deste modo o LCD só receberá instruções de escrita e assim não será possível utilizar o flag Busy para verificar se o controlador encontra-se ocupado. Para lançar mão desta facilidade o desenvolvedor precisa utilizar pausas entre os comandos e os tempos destas pausas precisam respeitar as especificações já mencionadas anteriormente. Este artifício pode ser utilizado tanto no modo de comunicação de 4 bits quanto de 8 bits.

Exemplos de rotinas básicas

A seguir são apresentados exemplos de rotinas para acesso a LCD em assembly e C (CCS), tanto no modo 8 bits quanto no modo 4 bits.

Caso o usuário não queira utilizar a linha R/W para economizar um pino de I/0, a linha R/W deve ser ligada ao terra ,e como o flag Busy não estará displonível, será necessário utilizar as rotinas de pausas.

Rotinas em ASM para PIC

Modo 8 bits

```
;; as rotinas de pausas devem trabalhar com
;; folga para clock de até 8 MHz.
;; as seguintes váriaveis devem ser declaradas
; LCD_dados = port que será ligado em D7 a D0
; LCD_Busy =  7  (bit  7 de LCD_dados)
; LCD_TRIS_dados = TRIS do port dos dados
;;
; LCD_ctrl = port que será ligado as linhas de
;;controle do LCD.
;;LCD_RS = bit de LCD_ctrl da linha RS
;;LCD_RW = bit de LCD_ctrl da linha RW
;;LCD_Enable = bit de LCD_ctrl da linha Enable
;;LCD_TRIS_ctrl = TRIS do port dos controles
;;
;; Temp1 e Temp2 = variáveis temporárias

;;; Enviar_Comando/Enviar_dado
;;; o comando ou dado
;;;   deve ser passado no registro W.
;;;
Enviar_Comando
   bcf      LCD_ctrl, LCD_RS
   goto     Comum_enviar
Enviar_Dado
   bsf      LCD_ctrl, LCD_RS
Comum_enviar
  movwf  LCD_dados
   bcf      LCD_ctrl, LCD_RW
   bsf      LCD_ctrl, LCD_Enable
   nop
   nop
   bcf      LCD_ctrl, LCD_Enable
   return

;;; Checar_Busy
;;; só retorna da rotina quando o
;;;  flag Busy = 0  (livre)
;;;
Checar_Busy
   movlw  0xFF
   bank1
   movwf  LCD_TRIS_dados
   bank0
   bcf      LCD_ctrl, LCD_RS
   bsf      LCD_ctrl, LCD_RW
Cont_Busy
   bsf      LCD_ctrl, LCD_Enable
   nop
   nop
   rlf      LCD_dados
   bcf      LCD_ctrl, LCD_Enable
   nop
   nop
   skipnc
   goto     Cont_Busy
   bank1
   clrf     LCD_TRIS_dado
   bank0
   return

;;** Inicializar Display
;;  inicializa display modo 8 bits, 2 linhas
;;  caracteres 5x7, escrever sobrepondo
;;  cursor  ligado e piscando
Inicializar_Display
  bank1
  clrf    LCD_TRIS_dados
  clrf    LCD_TRIS_ctrl
  bank0
  movlw   0x30
  call    Enviar_comando
  call    Pausa_longa
  call    Pausa_longa
  movlw   0x30
  call    Enviar_comando
  call    Pausa_longa
  movlw   0x30
  call    Enviar_comando
  call    Pausa_curta
  movlw   0x38
  call    Enviar_comando
  call    Pausa_curta
  movlw   0x0B
  call    Enviar_comando
  ;  a partir deste ponto pode-se usar a rotina
  ;  Checar_Busy ao invés das rotinas de pausas,
  ; o que pode evitar perdas de tempo desnecessárias
  call    Pausa_curta
  movlw   0x01
  call    Enviar_comando
  call    Pausa_curta
  movlw   0x06
  call    Enviar_comando
  call    Pausa_curta
  clrf    Temp1
  clrf    Temp2
  return

Pausa_longa
  clrf    Temp1
  goto    Comum_pausa
Pausa_curta
  clrf    Temp1
  incf    Temp1,f
Comum_pausa
  decfsz  Temp2,f
  goto    Comum_pausa
```

```
   decfsz    Temp1,f
   goto      Comum_pausa
   return
```

Modo 4 bits:

```
; **implementação das rotinas anteriores
;  utilizando o modo de comunicação de 4 bits
; fazendo uso de apenas um port e ainda sobrando
; 1 bit (bit 0) livre para o usuário
; D7 a D4 = bit 7 a bit 4,  Enable = bit 3,
; R/W = bit2 , RS= bit 1, bit 0 livre
;; as seguintes váriaveis devem ser declaradas
; LCD_port = port que será usado para comunicação
;  com o LCD
; LCD_Busy =  7  (bit  7 de na leitura)
; LCD_TRIS_port = TRIS do port do LCD
;
;;LCD_Enable = bit 3 de LCD_port
;;LCD_RW = bit 2 de LCD_port
;;LCD_RS = bit 1 de LCD_port
;;
;; Temp1,Temp2  e Temp3= variáveis temporárias

;; o comando  ou dado é passado em W
Enviar_cmd
   bcf       LCD_port, LCD_RS
   goto      Comum_enviar
Enviar_dado
   bsf       LCD_port, LCD_RS
Comum_enviar
   movwf   Temp1
   movlw   0x0F
   andwf    LCD_port, w
   addwf    Temp1, w
   movwf   LCD_port
   bcf       LCD_port, LCD_RW
   bsf       LCD_port, LCD_Enable
   nop
   nop
   bcf       LCD_port, LCD_Enable
   return

;;; Checar_Busy
;;; só retorna da rotina quando o
;;;  flag Busy = 0  (livre)
;;;
Checar_Busy
    movlw  0x0F ; alterar se usar bit 0
    bank1
    andwf  LCD_TRIS_port,f
    bank0
    bcf       LCD_port, LCD_RS
    bsf       LCD_port, LCD_RW
Cont_Busy
    bsf       LCD_port, LCD_Enable
    nop
    nop
    movlw    0x80
    andwf     LCD_port,w
    bcf       LCD_port, LCD_Enable
    nop
    nop
    skipnc
    goto      Cont_Busy
    bank1
    clrf      LCD_TRIS_dado
    bank0
    return

;;** Inicializar Display
;;  inicializa display modo 4 bits, 2 linhas
;;  caracteres 5x7, escrever sobrepondo
;;  cursor  ligado e piscando
Inicializar Display
   bank1
   clrf     LCD_TRIS_port
   bank0
   movlw    0x30
   call       Enviar_cmd
   call       Pausa_longa
   call       Pausa_longa
   movlw    0x30
   call       Enviar_cmd
   call       Pausa_longa
   movlw    0x30
   call       Enviar_cmd
   call       Pausa_curta
   movlw    0x28
   call       Enviar_cmd
   call       Pausa_curta
   movlw    0x0B
   call       Enviar_cmd
   ;  a partir deste ponto pode-se usar a rotina
   ;  Checar_Busy ao invés das rotinas de pausas,
   ; o que pode evitar perdas de tempo desnecessárias
   call       Pausa_curta
   movlw    0x01
   call       Enviar_cmd
   call       Pausa_curta
   movlw    0x06
   call       Enviar_cmd
   call       Pausa_curta
   clrf       Temp1
   clrf       Temp2
   return

 Pausa_longa
   clrf       Temp1
   goto       Comum_pausa
Pausa_curta
   clrf       Temp1
   incf       Temp1,f
Comum_pausa
   decfsz    Temp2,f
   goto       Comum_pausa
   decfsz    Temp1,f
   goto       Comum_pausa
   return
```

Rotinas em C (CCS) para PIC

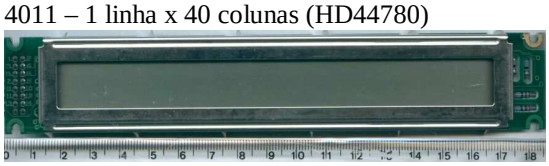
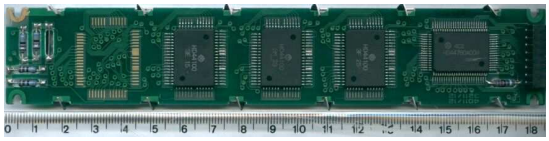
Como o compilador CCS já possui suas próprias rotinas para manipulação de LCD no modo de comunicação de 4 bits, aqui só será apresentada uma alternativa para o modo 8 bits.

Modo 8 bits

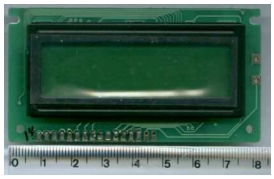
```
// Será utilizado o PORTB para linha de dados
//e PORTD para linhas de controle , que pode
// ser alterado pelo usuário conforme necessidade
// ou modelo de PIC utilizado.
//
//atualizado em 06/07/2004, os valores em hexa
// estavam no formato do compilador Hitech, a
//rotina Busy continha problema e estava
//incompleta
#DEFINE lcd_ctr PORTD
#DEFINE TRIS_ctr TRISD
#DEFINE lcd_dados PORTB
#DEFINE TRIS_dados TRISD
#DEFINE lcd_RS   0
#DEFINE lcd_RW  1
#DEFINE lcd_Enable 2

 void  lcd_cmd8 (int8 cmd)
{
   output_B (cmd);
   bit_clear  (lcd_ctr, lcd_RS);
   bit_clear  (lcd_ctr, lcd_RW);
   lcd_pulso (  );
}

 void lcd_dat8 (int8 dat )
{
   output_B (dat);
   bit_set     (lcd_ctr, lcd_RS);
   bit_clear  (lcd_ctr, lcd_RW);
   lcd_pulso (  );
}

void lcd_pulso (void )
{
  bit_set  (lcd_ctr, lcd_Enable);
  asm (“nop”);
  asm (“nop”);
  bit_clear (lcd_ctr, lcd_Enable);
}

void lcd_init8 (void )
{
//  só são utilizados os 3 bits de menor
// ordem de lcd_ctr (PORTD)
TRIS_ctr  =  TRIS_ctr  & 0xF8;
TRIS_dados =  0x00;
lcd_cmd8  (0x30) ;
delay_ms (30);
lcd_cmd8  (0x30);
delay_ms (10);
lcd_cmd8  (0x30);
delay_ms (02);
// modo 8 bits, 2 linhas, car 5x7
// a partir deste ponto pode-se verificar o flag
// busy  ao invés de usar delay
lcd_cmd8  (0x38);
delay_us (40);   // ou  busy (  );
// desliga display, cursor desligado
// não piscar cursor
lcd_cmd8  (0x08);
delay_us (40);
// limpar tela do display
 lcd_cmd8  (0x01);
delay_us (40);
// sobrepor dados  antigos ao escrever
lcd_cmd8  (0x06);
delay_us (40);
// ligar display
lcd_cmd8 (0x0A);
}

int busy  (void ){
  TRIS_dados = 0xFF;
  bit_clear  (lcd_crt, lcd_RS);
  bit_set     (lcd_ctr, lcd_RW);
  bit_set     (lcd_ctr, lcd_Enable);
  asm(“nop”);  asm(“nop”);
 while  (bit_test(lcd_dados,7))
    { asm(“nop”);
    asm(“nop”);
    bit_clear  (lcd_ctr, lcd_Enable );
    asm(“nop”);
    bit_set (lcd_ctr, lcd_Enable);
  }
 TRIS_dados = 0x00;
}
void lcd_gotoxy (int8  linha, coluna)
// linha 1 a ??, coluna 1 a 80
{
int8  col, offset;
if (linha == 1)  offset = 0x80;
if (linha == 2)  offset = 0xC0;
// os IF abaixo são válidos para LCD de 4
// linhas por 16 colunas
if (linha == 3)  offset = 0x90; //para 20 col ,0x95
if (linha == 4)  offset = 0xD0;//para 20 col ,0xD5

col = coluna;
if ((col < 1) || (col>80)) col=1;
lcd_cmd8  (offset + (col –1));
}
```

Alguns modelos de LCD

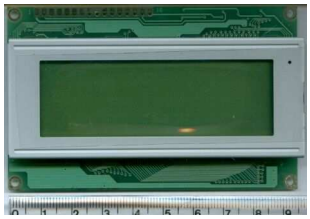
4011 – 1 linha x 40 colunas (HD44780)

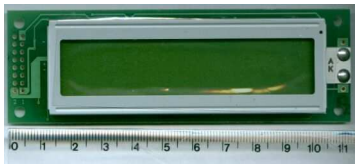
A1602K – 2 linhas x 16 colunas (HD44780)



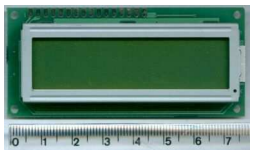

AC204A – 4 linhas x 20 colunas (HD44780)



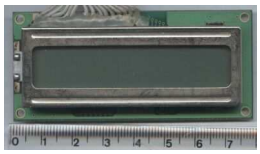

AC202A – 2 linhas x 20 colunas (HD44780)




162B – 2 linhas x 16 colunas (HD44780)

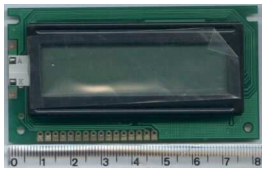



PVC1602 – 2 linhas x 16 colunas (KS0076)

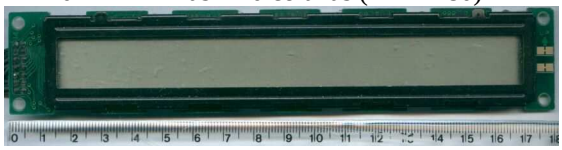



PWB16433 – 4 linhas x 16 colunas (HD44780)

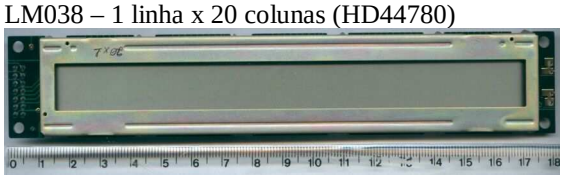
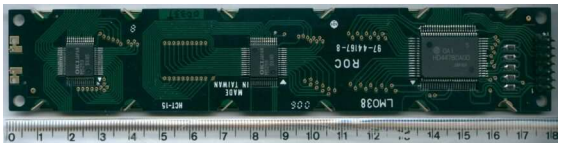

LCM1602 – 2 linhas x 16 colunas (KS0066)




DM4021 – 2 linhas x 40 colunas (HD44780)




LM038 – 1 linha x 20 colunas (HD44780)

Considerações finais e curiosidades

Quando a quantidade de linhas de controle é critica, pode-se economizar uma linha colocando a linha R/W direto ao terra, neste caso não é possível ler-se os dados do LCD e portanto não pode-se utilizar o flag busy.

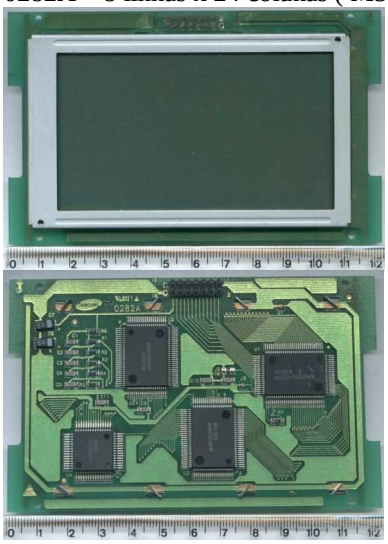
A verificação do flag busy antes de enviar um comando/dado ao LCD ao invés de utilizar rotinas de delay pode melhorar muito a performance dos programas visto que os mesmos só irão esperar o tempo que o LCD esteja realmente ocupado, porém existem aplicações em esta prática pode ser catastrófica pois se ocorrer algum problema na comunicação com o LCD (ex. o LCD queimar) o programa ficará travado. Um exemplo de aplicação onde é mais indicado utilizar delay ao invés de verificar o flag busy, seria uma central de alarme visto que se ocorrer algum problema com o LCD, o programa travaria e toda a central de alarme ficaria inoperante.

Alguns LCD (principalmente os construídos com controladores compatíveis) podem não ter internamente o mapa de caracteres 5 x 10.

Alguns displays apresentam uma separação entre as 2 linhas de caracteres acentuada, o que inviabiliza a utilização de caracteres 5 x 10 (mesmo que exista a fonte na CGROM) pois a visualização daria a sensação de que os caracteres estivessem partidos.

Apesar dos LCD com o controlador HD44780 serem os mais fáceis de se encontrar, existem modelos com outros controladores também interessantes e até mais poderosos como o M50530 (figura 13) que possui 12 bytes de CGRAM, 256 bytes de DDRAM, o dobro de vias horizontais que o HD44780, permitindo compor displays de até 64 colunas por 4 linhas.

Figura 13.
0282A – 8 linhas x 24 colunas (M50530)



Existem também LCD caracteres que não tem controlador, possuindo apenas drivers de linhas e segmentos, alguns até possuem RAM (semelhante a DDRAM). Estes displays apesar de serem semelhantes aos displays gráficos , pois todos os caracteres precisam ser desenhados ponto a ponto (byte a byte) na tela, não podem ser considerados gráficos devido a construção física de suas linhas de pontos, as quais são separadas por um espaço maior a cada grupo de 8, inviabilizando assim uma utilização realmente gráfica (figuras 14 e 15).

Figura 14.

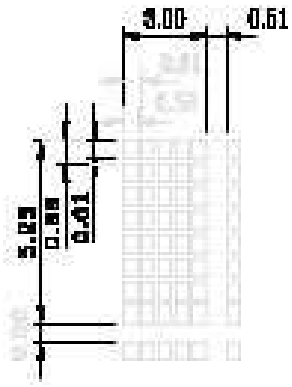
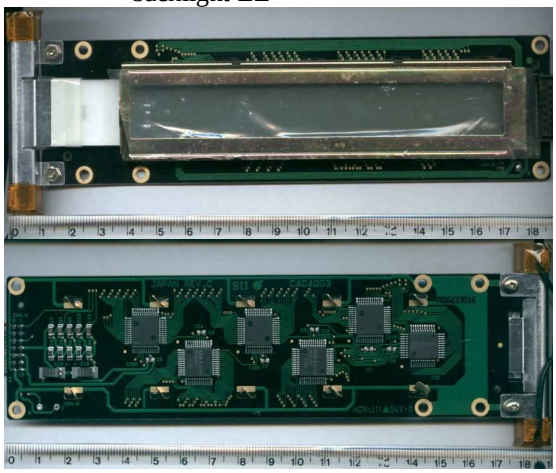


Figura 15.
C404003 – 4 linhas x 40 colunas (só drivers)
backlight EL



Autor, fontes de pesquisa, etc.

O autor é Analista de Sistemas trabalhando há 14 anos com Mainframes. Hobbytista de Eletrônica.

No site do autor http://reniemarquet.sites.uol.com.br , na página Projeto Remoto Serial, podem ser encontrados arquivos para teste de alguns modelos de LCD caracter através do projeto exposto na página, inclusive um vídeo da execução de teste com um LCD de 4 x 16.

Links interessantes:
http://www.eio.com/lcdintro.htm
http://www.skippari.net/phpBB2/viewforum.php?f=6
http://www.shellyinc.com/index.htm
http://www.lcd-module.com/eng/dbl/dbl.htm