Comunicação de LCD em 4 vias

Eduardo Souza Ramos
22 de setembro de 2005

ÍNDICE

INTRODUÇÃO

A idéia deste material é realizar a conexão de um LCD paralelo que utiliza o processador Hitachi HD44780 ou KS0066U a um microcontrolador PIC, utilizando apenas 4 vias de dados. Devido à sua simplicidade, estes LCDs não são avançados e não possuem recursos como gerenciamento automático de pixels, não são coloridos (full color), não possuem iluminação ativa entre outras limitações, mas ainda são largamente utilizados na indústria. Basta ver que muitas registradoras, equipamentos hand-held, sistemas de informação de computadores servidores entre outros, ainda utilizam largamente este dispositivo.

Os testes foram executados em uma placa Mosaico McLAB2 e em uma placa confeccionada para esta finalidade. Os diagramas para esta placa encontram-se no capítulo referente ao hardware.

Este material foi desenvolvido utilizando o MPLAB 7.21 e o compilador PCWH 3.168 (versão gratuita).

Estes procedimentos também foram testados em Assembly, mas está fora do escopo deste trabalho. Para programação e gravação do microcontrolador existente nas placas utilizei o McPLUS (também da Mosaico). Isto garantiu uma perfeita integração ao MPLAB e às características de gravação “in circuit” (ICSP).

Os PICs testados foram o PIC16F628A, PIC16F877A e o PIC18F452.

Capítulo 1 – O LCD Alfa-numérico

Capítulo 1 - O LCD ALFA-NUMÉRICO

Existem, comercialmente, diversos tipos e tamanhos de LCD alfanuméricos que podem ser incorporados a um microcontrolador PIC. Os mais comuns são os gerenciados por um processador Hitachi HD44780. Apesar de ter sido descontinuado, ainda é o mais comumente encontrado, assim como os equivalentes de outros fabricantes. Maiores detalhes sobre este processador podem ser encontrados no data sheet do fabricante. Veja no link http://cn.renesas.com/media/products/lcd/discontinued_products/d_e780u.pdf .

Os displays são fornecidos em diversos formatos e tamanhos e são sempre especificados em número de caracteres exibidos, no formato de colunas e linhas, sendo os mais comuns os 08x02 (oito colunas por duas linhas), 16X01 (16 colunas por 1 linha), 16X02 (16 colunas por 2 linhas), 16X04 (16 colunas por 4 linhas), 20X01 (20 colunas por 1 linha), 20X02 (20 colunas por 2 linhas) e 20X04 (20 colunas por 4 linhas).

Estes são alguns exemplos:

Figura 1: Display 08X02 sem back-light

Figura 2: Display 16X01 sem back-light

Figura 3: Display 16X02 com back-light

Figura 4: Display 16X04 com back-light

Figura 5: Display 20X01 sem back-light

Figura 6: Display 20X02 sem back-light

Figura 7: Display 20X04 com back-light

Alguns módulos possuem iluminação em "back-light" e estão disponíveis nas cores (monocromáticas) azul, âmbar, verde entre outras. Mas, no fundo, todos eles são compatíveis e podem ser controlados pelo PIC exatamente da mesma maneira. Isto é importante para que, no futuro, uma rotina desenvolvida para um LCD 16X02 possa ser utilizada para controlar um LCD 20X04 sem a necessidade de alteração de código.

Parte 1 - Os caracteres

Basicamente, cada "célula" (caractere) do LCD é composto de 8 pixels na horizontal e de 5 pixels na vertical, ou seja, cada "célula" é representada da seguinte forma (todas as medidas são em mm):

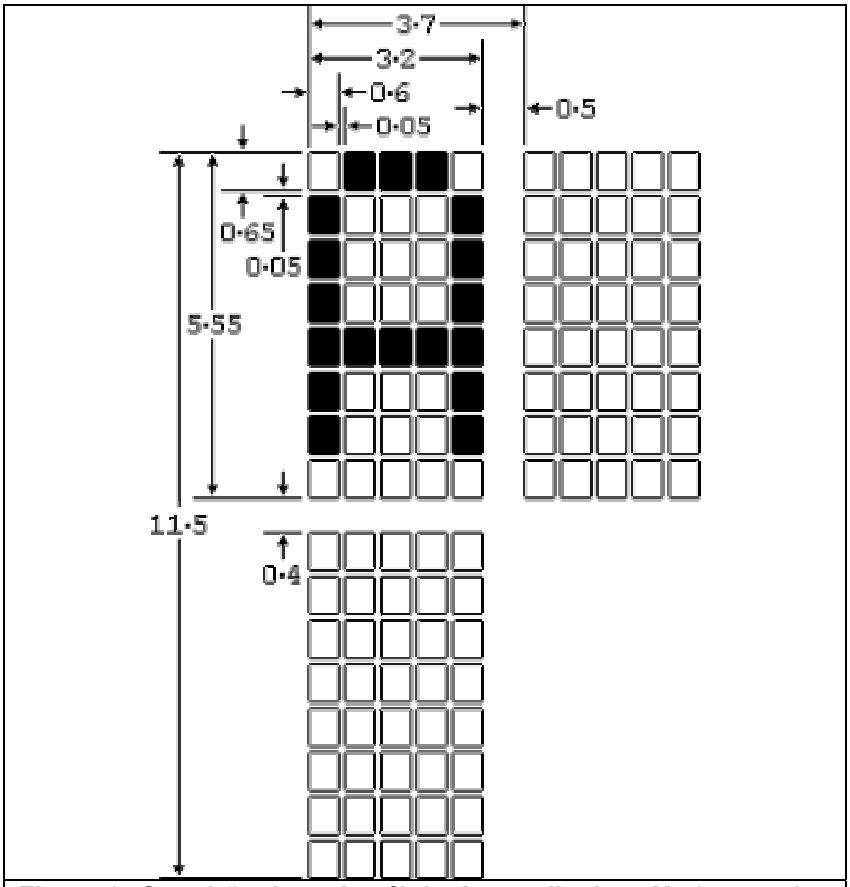


Figura 8: O padrão de cada célula de um display 5X7 (em mm)

Estes caracteres são conhecidos como 5X7, uma vez que a linha inferior é normalmente reservada para o cursor. Existem também displays que possuem 11 pixels de altura. Os caracteres deste LCD são conhecidos como 5X10 e tem a vantagem de que não existe a distância entre a primeira e a segunda linha. Os caracteres podem parecer contíguos.

Parte 2 - Conexões

Basicamente os LCDs alfa numéricos seguem um padrão de especificação de interface, onde estão presentes 14 pinos de acesso (para os LCDs sem iluminação em "back-light") ou 16 pinos (para os que possuem iluminação em "back-light"). Na verdade, estes pinos são “pads” soldáveis para a inserção de conectores IDC e são classificados em 8 pinos para controle de dados, três linhas de controle e três linhas de alimentação. Iremos detalhar cada um a seguir. As conexões estão dispostas em 1 linha de 14 (ou 16) pinos lado a lado, onde podemos utilizar uma barra de pinos SIL ou em duas linhas de 7 (ou 8) pinos, onde nos valeremos de um conector DIL. Na figura 1 podemos ver um exemplo de conector DIL e na figura 2 o exemplo mostra um display com conexão SIL.

Na maioria dos displays, a pinagem está impressa e torna mais fácil a identificação de cada um. Mas, nos casos onde não há identificação impressa, lembre-se que o pino 1 é sempre o terra e deve possuir uma trilha mais larga ou estar aterrado na carcaça em algum ponto da trilha.

A função de cada pino é resumida de acordo com a tabela abaixo:

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

Apesar dos LCDs serem projetados para trabalharem com 5V, consumindo apenas alguns miliampéres, tensões entre 4,5V e 6V funcionam perfeitamente. Alguns módulos também funcionam com 3V. Por isso, o mais indicado é sempre ler o data sheet do LCD que você possui para saber melhor seus recursos e limitações. Por estas razões é que o LCD pode ser utilizado, sem problemas, em equipamentos alimentados por pilhas e baterias. O consumo deles é bem menor que o de um simples LED!

Como já dito antes, o pino 1 (Vss) é o terra do módulo LCD e pode ser ligado juntamente com o pino Vss do PIC.

O Pino 2 (Vdd) é o positivo e deve ser ligado juntamente com o Vdd do PIC

O pino 3 (Vo) é onde controlamos o contraste do LCD, ou seja, onde controlamos se o que irá aparecer no LCD será "escuro" ou "claro". Para isso devemos ligá-lo a um potenciômetro ou trim-pot de 10KR conectado ao Vss e Vdd (atuando como um divisor de tensão) de acordo com o esquemático a seguir:

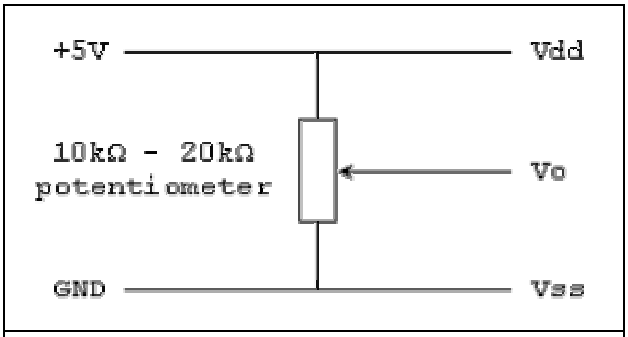


Figura 9: Controle de contraste de um LCD

O pino 4 (RS) é o Seletor de Registros. Resumindo, quando este pino está em nível lógico baixo (0), os dados enviados para o LCD são tratados como comandos e os dados lidos do LCD indicam o seu estado atual (status). Quando este pino está em nível lógico alto (1), os dados são tratados como caracteres, tanto para leitura como para escrita.

Para indicar se os dados serão lidos ou escritos no LCD, existe o pino 5 (R/W) que controla exatamente se a operação em andamento será de leitura (quando estivem em nível lógico alto - 1) ou gravação (quando estiver em nível lógico baixo - 0).

O pino 6 (E) é a linha de habilitação para os comandos do LCD. É utilizado para iniciar a transferência de comandos ou caracteres entre o módulo e as linhas de dados. Quando estiver escrevendo para o display, os dados serão transmitidos apenas a partir de uma transição de high para low (H -> L) deste sinal. No entanto, para ler informações do display, as informações estarão disponíveis imediatamente após uma transição L -> H e permanecerá lá até que o sinal volte para o nível lógico baixo (0) novamente.

Os pinos 7 a 14 fazem parte do barramentos de dados. Ele trabalha com os oito sinais em paralelo ou ainda pode trabalhar com um barramento de 4 vias (normalmente D4 a D7), mas os dados devem ser transmitidos em dois pacotes. Cada pacote de quatro bits é conhecido como "nibble". Este é um excelente recurso para minimizar o uso de pinos de I/O do microcontrolador, mas ocupa um pouco mais de memória. A decisão de utilizar 8 vias ou 4 vias é exclusiva do desenvolvedor do projeto.

Parte 3 - Comandos do LCD

Ao ligar o LCD na alimentação, a primeira linha fica toda preenchida (veja foto abaixo). Para que ele fique operacional, precisamos inicializá-lo, passando diversas informações de configuração. Para isso, existem alguns parâmetros que precisam ser definidos. A tabela 2 a seguir mostra quais são estes comandos.

Figura 10: LCD 16X02 energizado, mas não inicializado.

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.	3
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	3
Read Data	1	1	Dado								Lê dados da DDRAM ou da CGRAM	3

x : Sem importância						
	I/D	1 0	Incrementa Decrementa	R/L	1 0	Deslocamento para a direita Deslocamento para a esquerda
	S	1 0	Deslocamento automático do display	DL	1 0	Interface de 8 bits Interface de 4 bits
	D	1 0	Display ON (ligado) Display OFF (desligado)	N	1 0	2 linhas 1 linha
	C	1 0	Cursor ON (ligado) Cursor OFF (desligado)	F	1 0	Caracteres 5x10 Caracteres 5x7
	B	1 0	Cursor piscando	DDRAM : Display Data RAM CGRAM : Character Generator RAM		
	S/C	1 0	Deslocamento do display Movimento do Cursor			

Tabela 2: Instruções para inicialização e configuração do LCD

Baseados na tabela anterior verificamos que existem diversas configurações que podem ser atribuídas ao LCD. A tabela a seguir mostra as opções disponíveis.

RS	EN	D7	D6	D5	D4	D3	D2	D1	D0	Descrição	Binário	Decimal	Hexa
0	0	0	0	0	0	0	0	0	0	NOP (Sem operação)	00000000	0	0x00
0	0	0	0	0	0	0	0	0	1	Limpa o display, retorna para a posição inicial	00000001	1	0x01
0	0	0	0	0	0	0	0	1	X	CURSOR HOME - contador de endereço = 0, retorno para a posição inicial. Não altera a DDRAM	00000010	2	0x02
0	0	0	0	0	0	0	1	0	0	ENTRY MODE SET - A cada caracter, decrementa uma posição	00000100	4	0x04
0	0	0	0	0	0	0	1	0	1	ENTRY MODE SET - A cada caracter, decrementa uma posição e faça um shift automático	00000101	5	0x05
0	0	0	0	0	0	0	1	1	0	ENTRY MODE SET - A cada caracter, incrementa uma posição	00000110	6	0x06
0	0	0	0	0	0	0	1	1	1	ENTRY MODE SET - A cada caracter, incrementa uma posição e faça um shift automático	00000111	7	0x07
0	0	0	0	0	0	1	0	0	0	DISPLAY CONTROL - Display desligado.	00001000	8	0x08
0	0	0	0	0	0	1	0	0	1	DISPLAY CONTROL - Display desligado.	00001001	9	0x09
0	0	0	0	0	0	1	0	1	0	DISPLAY CONTROL - Display desligado.	00001010	10	0x0A
0	0	0	0	0	0	1	0	1	1	DISPLAY CONTROL - Display desligado.	00001011	11	0x0B
0	0	0	0	0	0	1	1	0	0	DISPLAY CONTROL - Display ligado, sem cursor	00001100	12	0x0C
0	0	0	0	0	0	1	1	0	1	DISPLAY CONTROL - Display ligado, sem cursor, piscando posição.	00001101	13	0x0D
0	0	0	0	0	0	1	1	1	0	DISPLAY CONTROL - Display ligado, cursor ligado..	00001110	14	0x0E
0	0	0	0	0	0	1	1	1	1	DISPLAY CONTROL - Display ligado, cursor ligado, cursor piscando	00001111	15	0x0F
0	0	0	0	0	1	0	0	X	X	CURSOR/DISPLAY SHIFT - Cursor move para a esquerda	00010000	16	0x10
0	0	0	0	0	1	0	1	X	X	CURSOR/DISPLAY SHIFT - Cursor move para a direita	00010100	20	0x14
0	0	0	0	0	1	1	0	X	X	CURSOR/DISPLAY SHIFT - Display move para a esquerda	00011000	24	0x18
0	0	0	0	0	1	1	1	X	X	CURSOR/DISPLAY SHIFT - Display move para a direita	00011100	28	0x1C
0	0	0	0	1	0	0	0	X	X	FUNCTION SET - Interface de 4 bits, 1 linha e fontes 5X7	00100000	32	0x20
0	0	0	0	1	0	0	1	X	X	FUNCTION SET - Interface de 4 bits, 1 linha e fontes 5X10	00100100	36	0x24
0	0	0	0	1	0	1	0	X	X	FUNCTION SET - Interface de 4 bits, duas linhas e fontes 5X7	00101000	40	0x28
0	0	0	0	1	0	1	1	X	X	FUNCTION SET - Interface de 4 bits, duas linhas e fontes 5X10	00101100	44	0x2C
0	0	0	0	1	1	0	0	X	X	FUNCTION SET - Interface de 8 bits, 1 linha e fontes 5X7	00110000	48	0x30
0	0	0	0	1	1	0	1	X	X	FUNCTION SET - Interface de 8 bits, 1 linha e fontes 5X10	00110100	52	0x34
0	0	0	0	1	1	1	0	X	X	FUNCTION SET - Interface de 8 bits, duas linhas e fontes 5X7	00111000	56	0x38
0	0	0	0	1	1	1	1	X	X	FUNCTION SET - Interface de 8 bits, duas linhas e fontes 5X10	00111100	60	0x3C

Tabela 3: Configurações possíveis para o LCD

Lembre-se que, antes de qualquer operação com o LCD ele precisa ser inicializado utilizando estas informações da ***Tabela 3***. É importante salientar que não existe uma ordem específica para os itens de configuração especificados acima. Eles podem ser enviados em qualquer ordem, uma vez que o bit mais significativo de cada categoria indica o seu grupo de configuração. No entanto para o processo de inicialização, é importante que antes de entrarmos nos grupos acima, existe uma ordem que deve ser respeitada. Isso será visto em um capítulo específico que irá abordar a rotina de inicialização do display.

Parte 4 - Tabelas de Caracteres

Como parte integrante do controlador que o LCD utiliza, há uma tabela de caracteres pré-programados que estão prontos para uso imediato. Com isso, todo o trabalho de definir pixel por pixel para os caracteres foi eliminado. Mas há uma desvantagem. Como a maioria dos LCDs são produzidos na Ásia, ele vem com um conjunto de caracteres específicos. A figura a seguir mostra a tabela existente na maioria dos LCDs. Ela é conhecida como **ROM Code A00.**

Tabela 4: Os caracteres do ROM Code A00

Existe outra tabela que também é encontrada nos LCDs, mas são mais raras. Geralmente os LCDs mais caros possuem esta opção. Novamente, é sempre muito importante

ler o data sheet do seu módulo de LCD antes de prosseguir com o desenvolvimento do projeto. A tabela a seguir ilustra os caracteres pertencentes ao **ROM Code A02:**

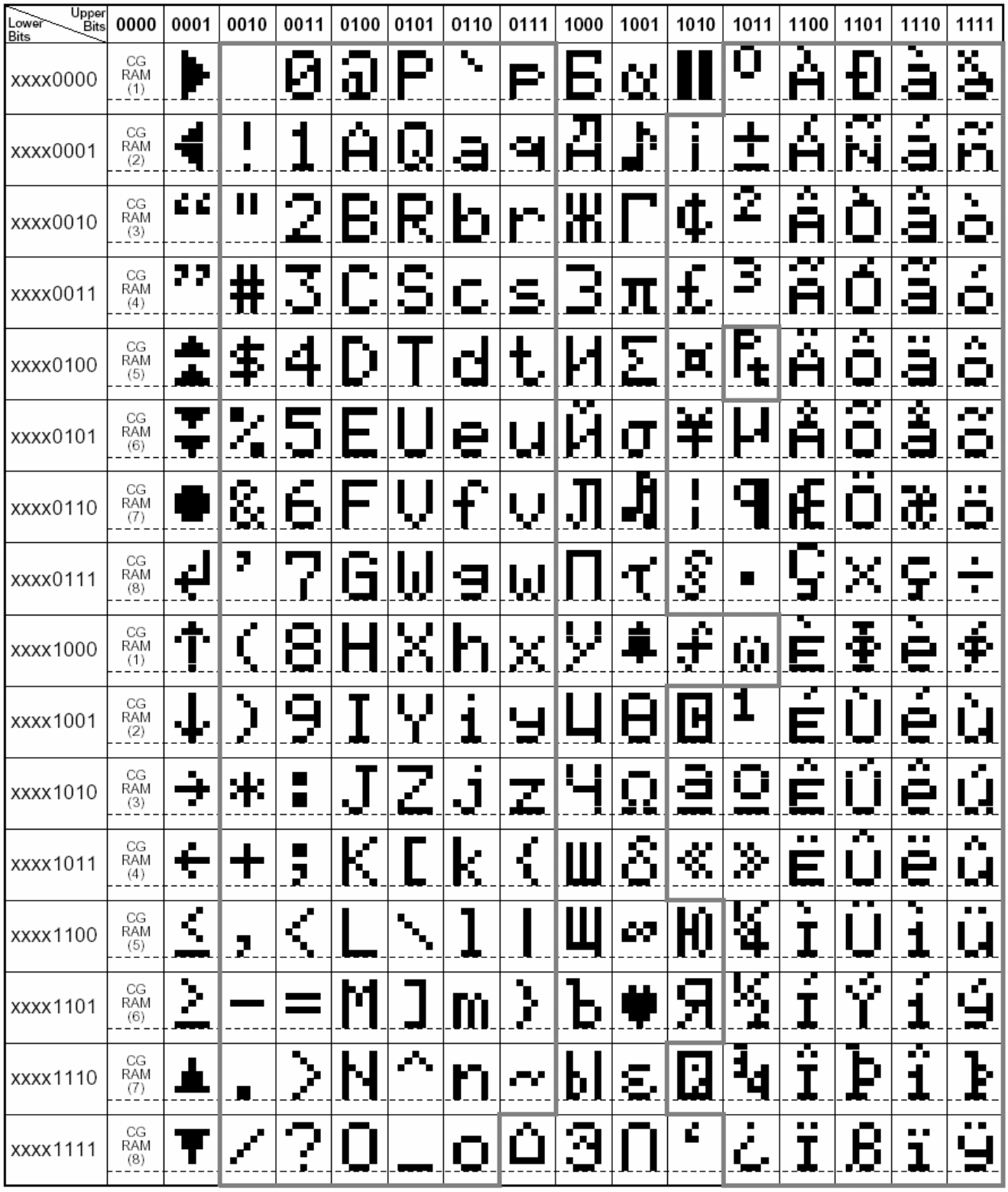


Tabela 5: Os caracteres do ROM Code A02

Como você pode reparar, os caracteres acentuados que temos em português somente estarão disponíveis nesta versão de ROM. Os LCDs são fornecidos com apenas um tabela de caracteres.

Para evitar que os módulos sejam “rígidos” no que diz respeito a caracteres exibidos, todos eles possuem uma área específica para caracteres criados pelo usuário. Esta área

chama-se CG RAM e é volátil, ou seja, se desligarmos o LCD ou o reiniciarmos, todas estas informações serão perdidas. Termos um capítulo específico sobre a CGRAM.

Parte 5 - Endereçamento

Após ligarmos o LCD e o iniciarmos, o cursor irá para a primeira posição que é a 0x00 (primeira linha x primeira coluna). De acordo com que vamos inputando dados nele, o cursor irá deslocar para as posições seguintes. Este auto-incremento é uma facilidade muito interessante, pois dispensa especificar cada posição para cada caractere em separado, economizando (e muito) em linhas de códigos necessárias.

fMas, e se quisermos escrever em um ponto específico do LCD?

Neste caso, podemos especificar exatamente qual é o endereço que o cursor deverá estar para exibir o caractere desejado. Este dado será passado para o LCD pelas mesmas vias de dados que passamos um caractere, só que dessa vez será um comando. De acordo com a Tabela 2, para entrarmos com um comando para setar um endereço na DDRAM do LCD precisaremos ter o pino RS e RW em nível lógico baixo e o bit mais significativo precisa estar obrigatoriamente em nível lógico alto. Com isso podemos endereçar até 128 posições e, ao passar o endereço, deveremos somar este bit, ou seja, precisaremos somar o valor 0x80 ao endereço desejado.

Para ilustrar, o endereçamento de um display LCD 16X02 é o seguinte:

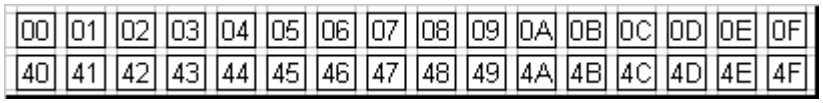


Figura 11: Endereçamento direto para um LCD 16X02

Como devemos, obrigatoriamente fazer com que o bit mais significativo do endereço seja 1, o valor que devemos passar para o LCD obedece à figura abaixo:

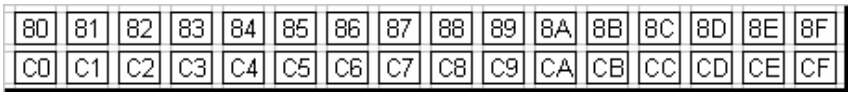
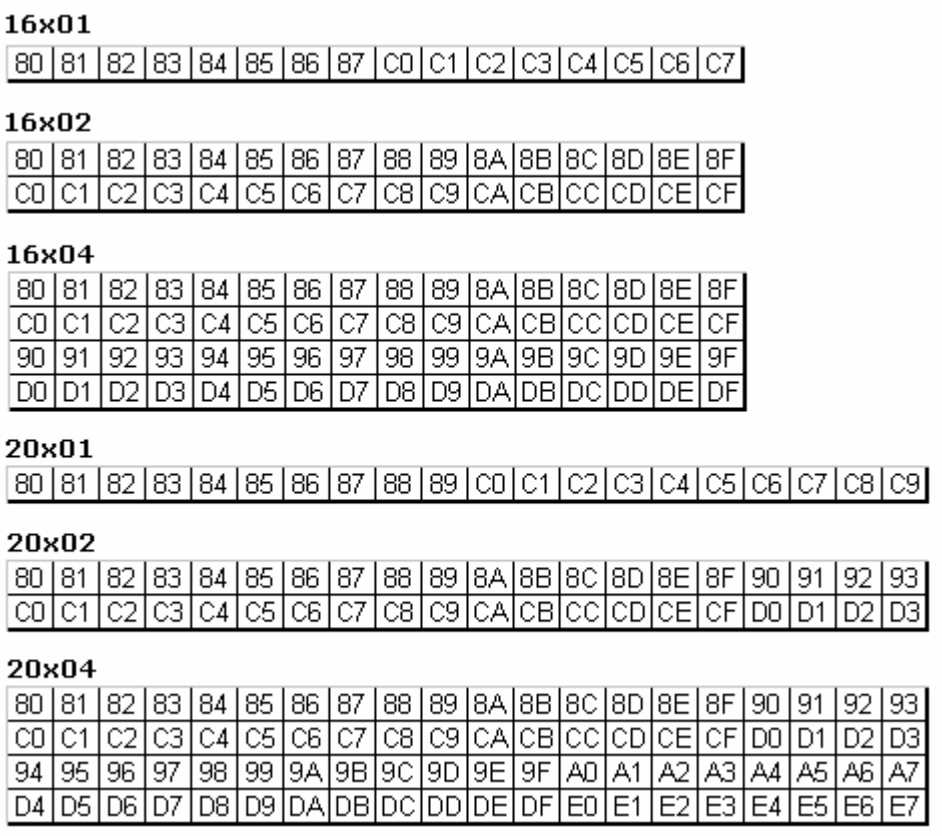


Figura 12: Endereçamento com deslocamento para um LCD 16X02

Para facilitar as contas, segue abaixo o endereçamento para a maioria dos LCDs comerciais:



16x01

80	81	82	83	84	85	86	87	C0	C1	C2	C3	C4	C5	C6	C7

16x02

80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF

16x04

80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF

20x01

80	81	82	83	84	85	86	87	88	89	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9

20x02

80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	90	91	92	93
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	D0	D1	D2	D3

20x04

80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	90	91	92	93
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	D0	D1	D2	D3
94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	A0	A1	A2	A3	A4	A5	A6	A7
D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5	E6	E7

Figura 13: Endereçamentos com deslocamento para diversos LCDs

Parte 6 - Rotacionando o LCD

Independente do tamanho do LCD existem sempre 80 posições por linha que podem ser usadas. Como não existem posições suficientes no mostrador do LCD, o texto é rotacionado ou deslocado, tanto para a direita como para a esquerda. Portanto este processo deve ser feito cuidadosamente para que não haja confusões durante o endereçamento.

Este modo também será estudado durante os exercícios.

Parte 7 - A CGRAM – Caracteres definidos pelo usuário

Os endereços 00000000 a 00000111 (0x00 a 0x07) são reservados para os caracteres criados pelo usuário. Temos, portanto, oito caracteres programáveis. Esta programação é realizada apontando para o endereço da CG RAM (character Generator RAM) onde desejamos armazenar o caractere. Os próximos 8 bytes enviados para a RAM representam cada linha do caractere, iniciando pelo top.

Ficou confuso? Então vamos explicar melhor. Como já vimos anteriormente, cada caractere é formado por uma matriz de 8 linhas por 5 colunas, de acordo com a figura abaixo.

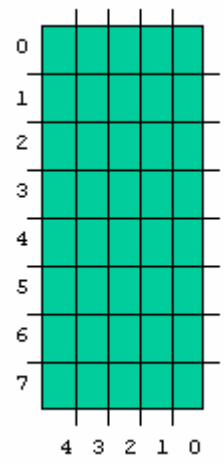


Figura 14: O modelo do caractere na CG RAM

Se desejarmos criar um “robozinho”, devemos preencher esta matriz da seguinte forma:

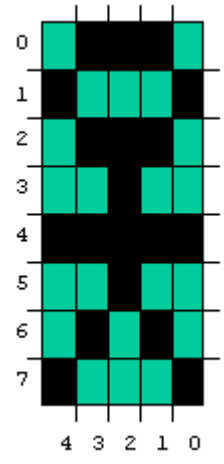


Figura 15: Criando um robô na CG RAM

Durante a programação, cada linha é tratada como um byte em separado. Portanto, imaginemos que cada linha seja tratada como um arranjo de bits, onde cada ponto preenchido é representado por 1 e cada ponto “apagado” seja representado por 0. Teremos então a seguinte configuração: Teremos então a seguinte configuração:

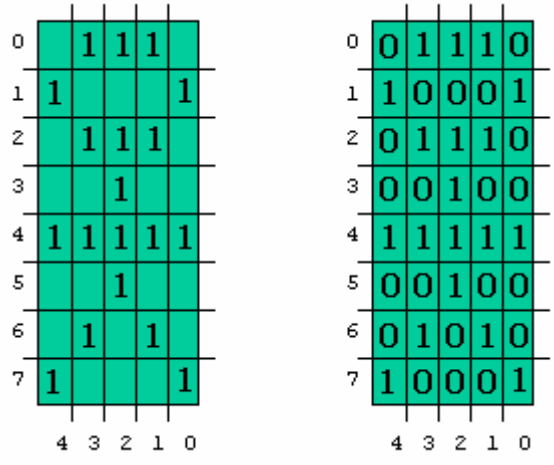


Figura 16: O robô, traduzido em bits

Lembrando que, pelo fato do byte ter 8 bits e o caractere ter apenas 5 bits, os 3 bits mais significativos serão sempre 0. Agora fica fácil para entendermos que o robô pode ser traduzido nos seguintes bytes (sempre de cima para baixo): 00001110, 00010001, 00001110, 00000100, 00011111, 00000100, 00001010 e 00010001. Passando para hexadecimal, teremos que o robozinho é decomposto nos bytes 0x0E, 0X11, 0X0E, 0X04, 0X1F, 0X04, 0X0A e 0X11.

Agora que já vimos como decompor um caractere, vamos aprender a programá-lo na CGRAM. De acordo com a Tabela 2, precisamos enviar o comando SET CGRAM ADRESS juntamente com o endereço do caractere inicial que desejamos gravar. Então, para iniciar a programação pelo primeiro caractere disponível, precisamos enviar o comando 00010000 para o LCD dizendo para posicionar o ponteiro no primeiro endereço da CGRAM. A seguir precisaremos enviar os 8 bytes que compões o caractere. Voilá! O nosso robozinho começa a aparecer na tela e pode ser acessado normalmente pelo endereço 0x00!

Agora, se continuarmos inserindo informações de bytes, automaticamente o ponteiro passará para o segundo caractere na CG RAM e assim sucessivamente. Então para programarmos os oitos caracteres, bastaria posicionar na posição inicial (0x00) e passarmos 64 bytes diretamente para o LCD.

Maiores detalhes serão vistos durante os exercícios onde criaremos diversos caracteres para uso.

Parte 8 - Transferência em 8 bits

O LCD baseado no controlador Hitachi HD44780 ou no controlador KS0066U possuem 8 vias de dados e são conhecidos como LCDs paralelos, pois os dados são enviados paralelamente ao LCD. Em nosso estudo não detalharemos este processo, visto que o nosso foco é a comunicação em 4 vias de acesso. Todavia, o código necessário para se comunicar com o LCD usando todas as vias de dados também é apresentado na seção SOFTWARE.

Para iniciarmos o LCD neste modo, devemos adotar a seguinte seqüência de comandos:

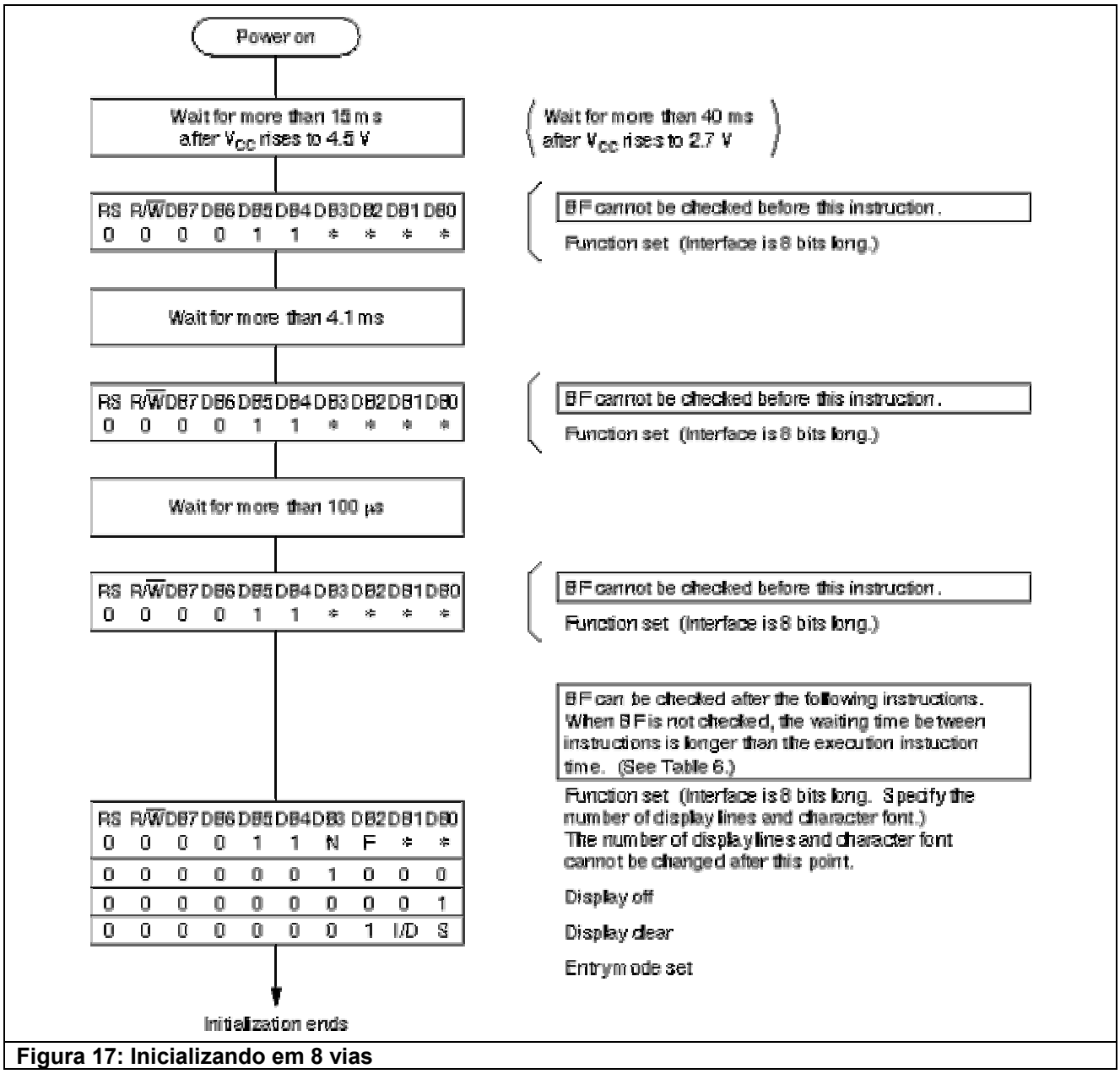


Figura 17: Inicializando em 8 vias

Parte 9 - Transferência em 4 bits

O controlador HD44780 ou o KS0066U, encontrados na maioria dos módulos de LCDs alfa-numéricos existentes atualmente foram desenvolvidos para serem 100% compatíveis com microprocessadores antigos de 4 bits. No entanto esta característica é muito útil quando precisamos interfacear um microcontrolador.

Muitas vezes, durante o desenvolvimento de um projeto, precisamos ter muito cuidado com o número de pinos de I/O utilizados, pois normalmente, este é o componente mais caro do projeto. Por isso precisamos racionar o uso destes pinos. Já pensou você desenvolvendo um equipamento com um PIC16F628A e ter que migrar para o, por exemplo, PIC16F873A apenas por que faltaram alguns pinos para ligar um LCD?

Outro ponto diz respeito à miniaturização, pois atualmente um grande diferencial de cada projeto está neste ponto. Cada vez mais precisamos ter equipamentos que exerçam mais funções e ocupem cada vez menos espaço. É aqui que reside a grande vantagem de se comunicar em quatro vias de dados.

Mas como eu vou mandar 1 byte (8 bits) se eu só tenho 4 vias de dados? Uma vez que o display é posto no modo de 4 vias, basta mandarmos 2 pacotes de 4 bits cada que o display se encarrega de fazer o resto. Cada pacote de 4 bits é conhecido como “nibble”.

Neste modo, apenas as vias de dados D4 a D7 são utilizadas, e as vias D0 a D3 devem ser deixadas flutuando (sem conexão alguma) ou conectadas ao positivo da alimentação, via resistor limitador que tem seu valor entre 4K7 e 47K. Não é aconselhável aterrá-los, a não ser que o pino R/W também esteja aterrado, prevenindo que os pinos sejam configurados como saída. Se o pino for configurado erroneamente como saída e os pinos estiverem diretamente aterrados, os pinos que estiverem com valor lógico 1 poderão se queimar.

Ao energizar o LCD, ele é automaticamente posto no mode de comunicação em 8 vias. Por isso, precisamos inicializá-lo para que possa operar corretamente. Neste ponto há uma pegadinha… Como eu faço para inicializar o LCD se ele só possui as vias D4 a D7 e ele está no modo 8 vias?

Foi pensando neste problema que os projetistas dos controladores determinaram que o bit que irá configurar o modo 4 vias seria o D4, por isso não precisamos nos preocupar. Basta enviar um comando com o valor 0b001000000 (0x20) que o LCD entra no modo de 4 vias.

A partir deste momento, todas as informações que serão passadas para o LCD deverão ser divididas em 2 “nibbles”, sendo que o “nibble” mais significativo deve ser enviado primeiramente, e o “nibble” menos significativo deverá ser enviado logo em seguida.

Para inicializarmos em 4 bits, devemos adotar a seguinte seqüência de comandos:

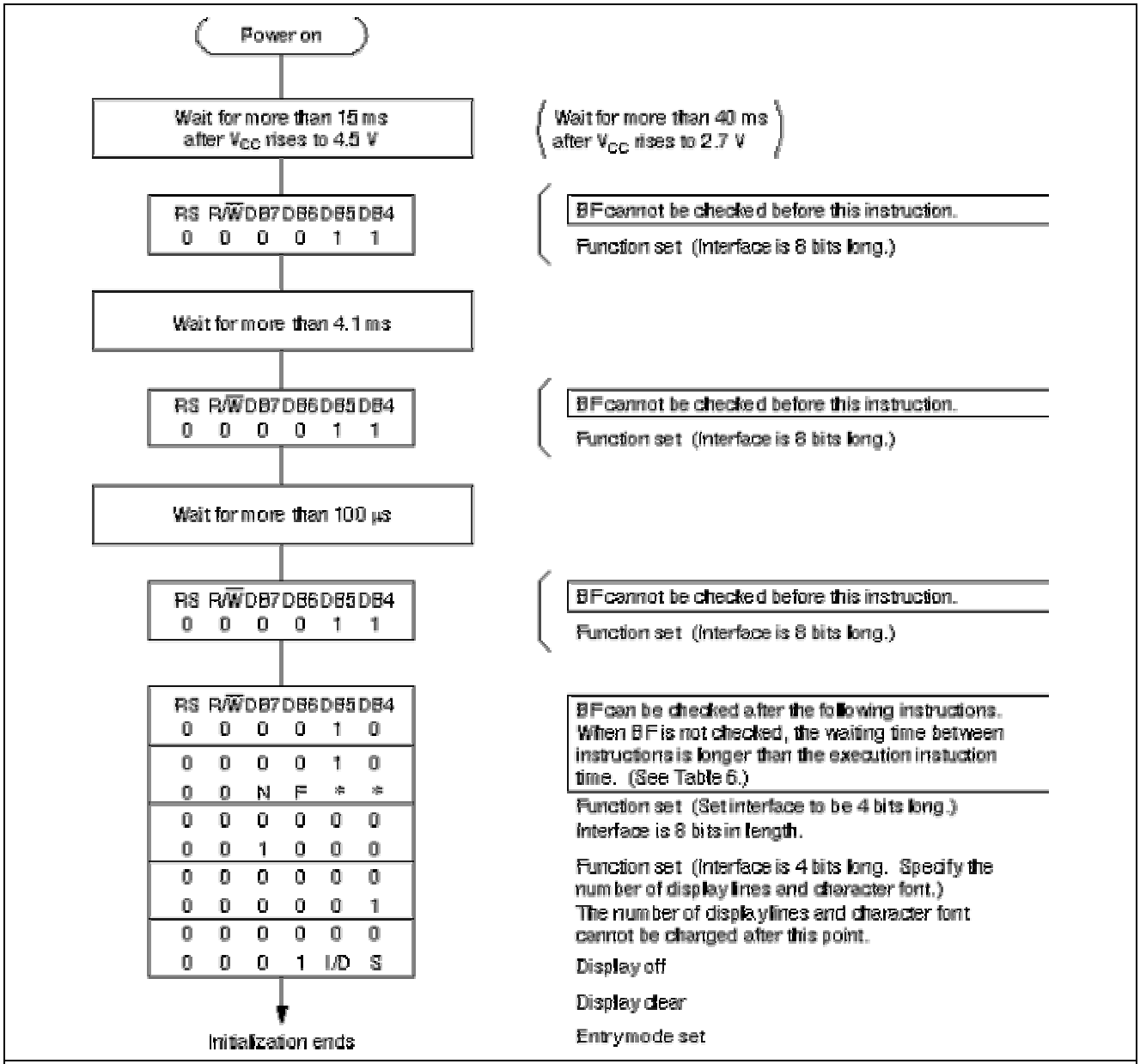
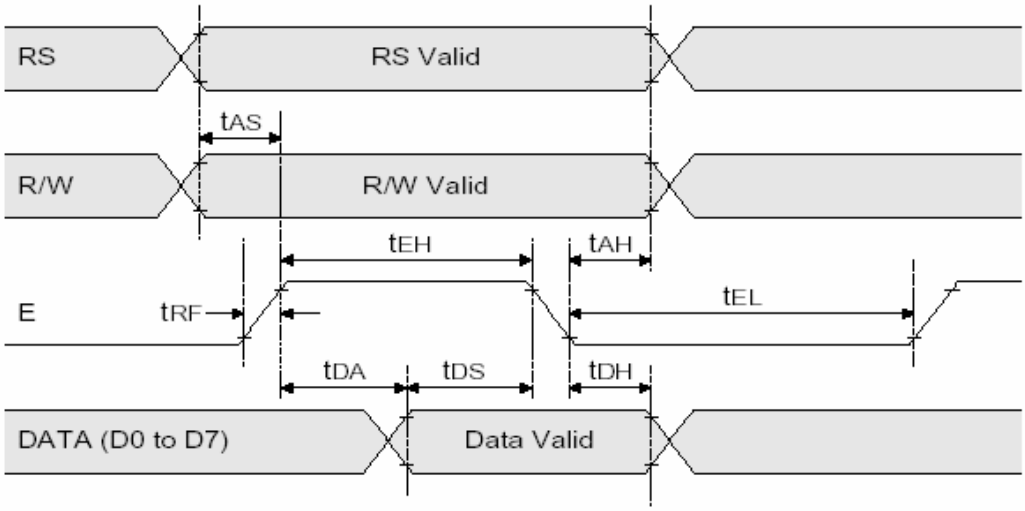


Figura 18: Inicializando em 4 vias

Parte 10 - Timing

Antes de passarmos definitivamente para a parte prática deste tutorial, precisamos conhecer uma limitação dos LCDs: o tempo de sincronização, também conhecido como TIMING. Independentemente da velocidade em que seu projeto/protótipo opera, existe um tempo mínimo que o LCD precisa para processar as informações. Se este tempo não for respeitado, o LCD não funcionará.

A figura abaixo ilustra o gráfico de tempo que o LCD opera



Parâmetro	Descrição	Tempo
tAS	Tempo de set-up do endereço (Address set-up time).	Mínimo 140ns
tAH	Tempo de retenção do endereço (Address hold time).	Mínimo 10ns
tDS	Tempo de ajuste do dado (Data set-up time).	Mínimo 200ns
tDH	Tempo de retenção do dado (Data hold time).	Mínimo 20ns
tDA	Tempo de acesso ao dado (Data access time).	Mínimo 320ns
tEH	Tempo para habilitar nível 1(Enable high time time).	Mínimo 450ns
tEL	Tempo para habilitar nível 0(Enable low time).	Mínimo 500ns
tRF	Tempo de transição (Rise/Fall time).	Mínimo 25ns

Figura 19: Diagrama de temporização para o LCD

Mas cuidado… Este diagrama não conta toda a história. Apesar de o LCD precisar apenas destes tempos para a execução, diversos flags internos precisam ser setados que, aliados a outros fatores, tornam o processo de temporização muito maior. A maioria dos comandos ocupam o LCD por até 40us e outros podem demandar até 3ms para serem liberados. É exatamente por isso que existe um flag chamado “BUSY” que indica se o display está pronto para executar outra instrução ou não. A função para esta operação é a READ STATUS.

Em caso onde não utilizamos o modo de leitura do LCD, podemos nos basear em um tempo mínimo que cada operação precisa para ser completada. A tabela a seguir ilustra os tempos necessários.

Instrução	Tempo
Limpar ao Display (Clear Display)	82us a 4,1ms
Ir para a posição inicial (display & Cursor Home)	40us a 4,1ms
Entrada de caracteres (Character Entry Mode)	40us
Controles de Display e Cursor (Display On/Off & Cursor)	40us
Deslocamento do display e do cursor (Display/Cursor Shift)	40us
Ajustes de funções (Function Set)	40us
Setar endereço da CGRAM (Set CGRAM Address)	40us
Setar endereço do display (Set Display Address)	40us
Escrever dados (Write Data)	40us
Ler dados (Read Data)	40us
Led o estado (Read Status)	1us
6: Diagrama de temporização real para o LCD	

Novamente, apesar da tabela acima funcionar em qualquer LCD alfa-numérico, consulte o data sheet de seu LCD para determinar exatamente quais são os tempos mínimos necessários.

Capítulo 2 – O Hardware

Capítulo 2 - O HARDWARE

Apesar de este tutorial ter sido desenvolvido para utilização na placa Mosaico McLAB2, quem não possuir esta placa poderá fazer uma placa para testes. O esquemático para tal placa de testes é mostrado na figura abaixo. Para testes realizados na placa McLAB2 nenhuma alteração é necessária.

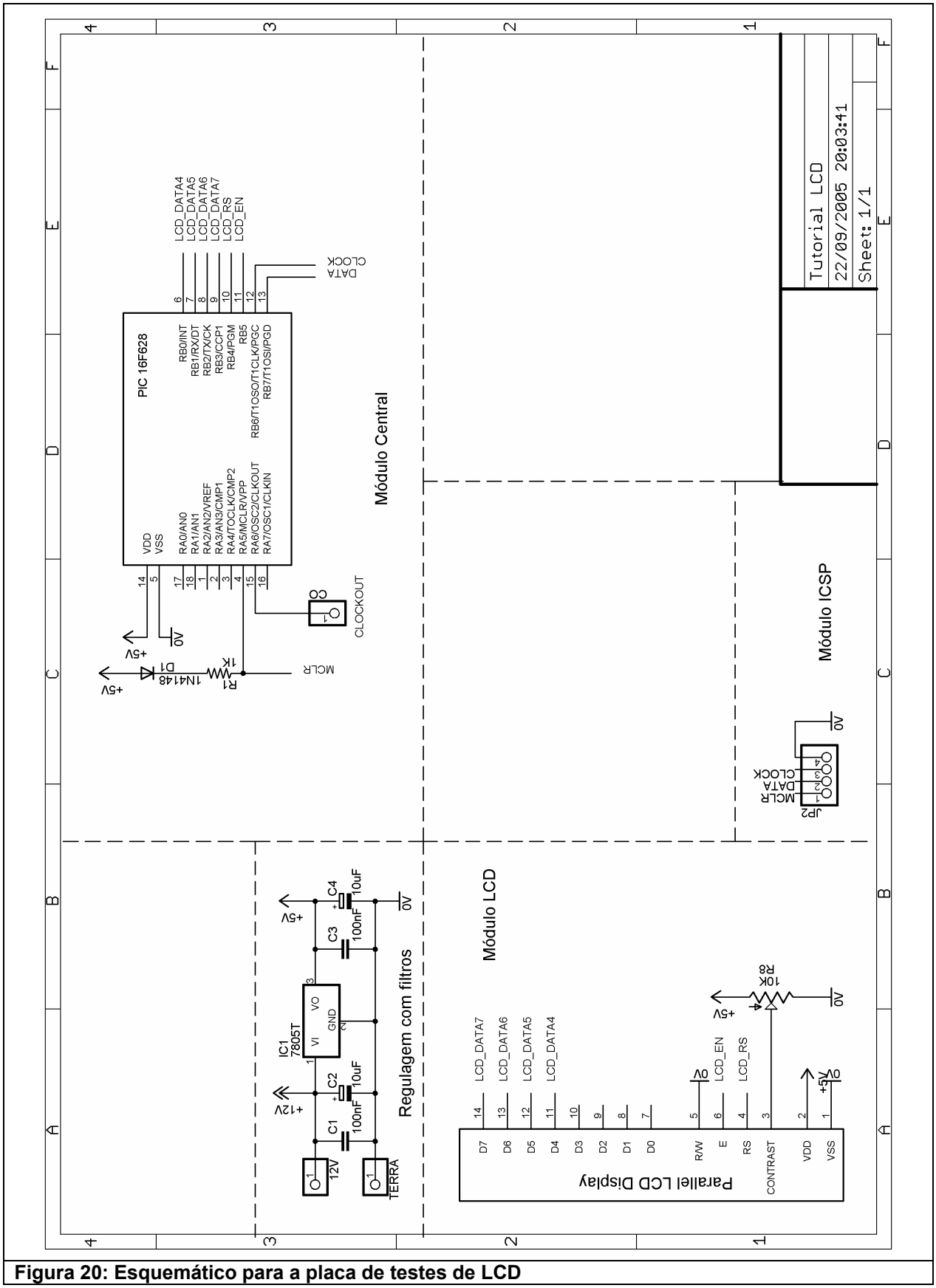


Figura 20: Esquemático para a placa de testes de LCD

A placa de circuito impresso sugerida é mostrada nas figuras abaixo e tem 2.225” X 1.600” (aproximadamente 56,5mm X 40.6mm).

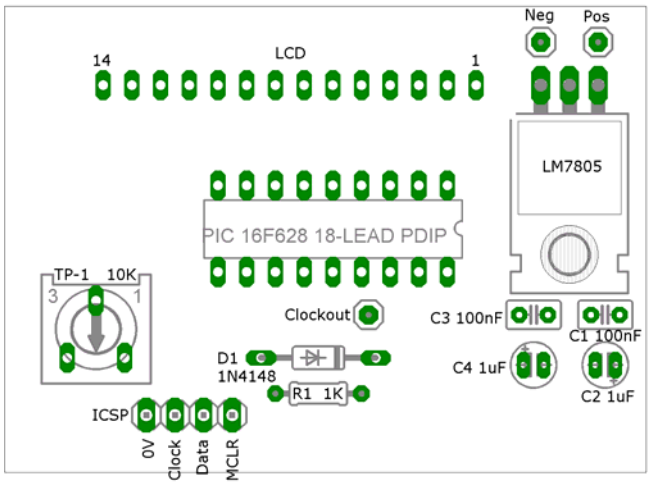


Figura 21: PCI lado dos componentes

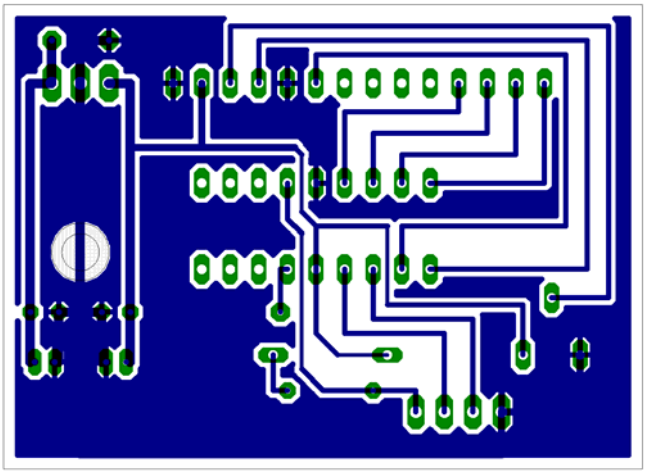


Figura 22: PCI lado da solda (trilhas)

O conector ICSP foi dimensionado para utilizar diretamente com os programadores McFlash e McPlus, evitando assim que o PIC tenha que ser retirado do circuito para ser programado.

Outro detalhe diz respeito ao regulador de tensão LM7805 que pode ser substituído pelo 78L05, devido ao baixo consumo do circuito.

Todos os componentes estão detalhados tanto no esquemático quanto no desenho da PCI – Lado dos componentes, são facilmente encontrados no mercado e são de baixo custo.

A foto abaixo mostra a placa conectada diretamente no McPlus.

Como este protótipo não utiliza oscilador ou ressonador externo, apenas o interno calibrado à 4MHz, utilizou-se o pino CLOCKOUT para realizar aferições para calibragem.

O PIC aqui utilizado foi o 16F628A.

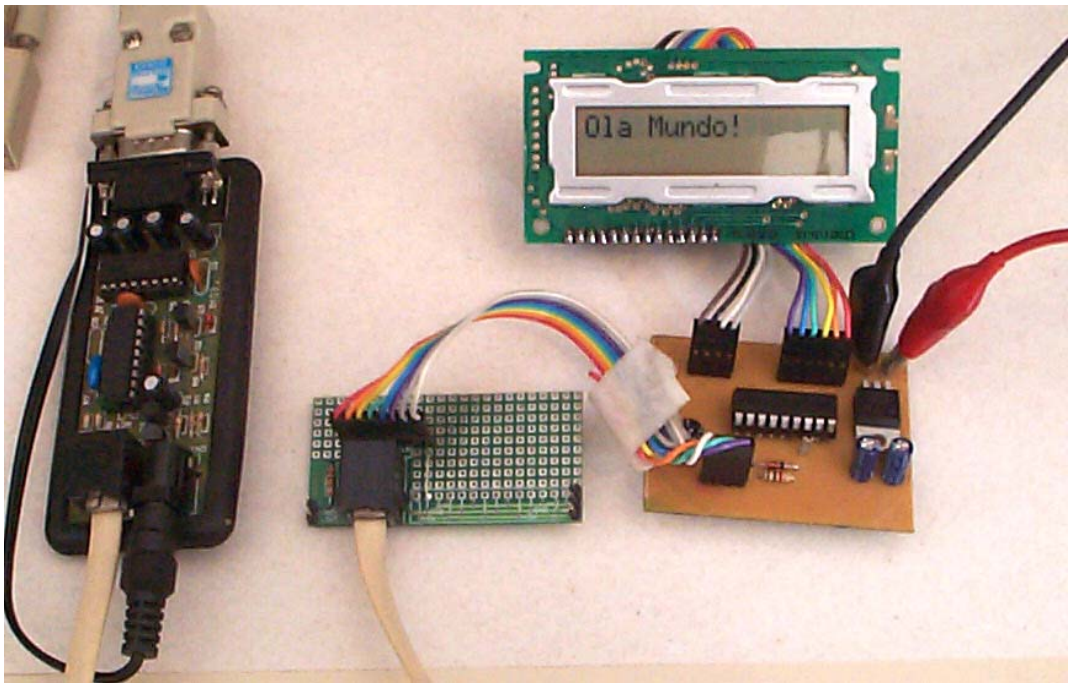


Figura 23: Placa de testes conectada ao McPlus via conector ICSP

Capítulo 3 – O Software

Capítulo 3 - O SOFTWARE

Esta seção será dividida em três partes. A primeira e a segunda serão inteiramente focadas na McLAB2, onde apresentaremos, na parte 1, o código para comunicação em 8 vias e em seguida o código necessário para estabelecer a comunicação em 4 bits na parte 2. Todo o detalhamento é explicitado no próprio código.

Na terceira parte iremos demonstrar a comunicação em 4 bits na placa de testes apresentada no capítulo anterior. Como não há ligação física das 8 vias de dados nela, obviamente não iremos demonstrar esta comunicação. Mas será facilmente verificado que o código detalhado para a McLAB2 será perfeitamente aproveitado com pequenas mudanças.

Parte 1 – Comunicação em 8 vias pela McLAB2

Este é o código necessário para exibir uma mensagem no LCD, utilizando o PORTD do PIC16F877A da placa McLAB2

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                                                                             *
 *                                                                             *
 *                                                                             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                              *
 *     DATA : 23/09/2005                                                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Descrição geral                                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Este exemplo foi elaborado para explicar o funcionamento do módulo de LCD.
// Foi utilizado um barramento de 8 vias de comunicação pelo PORTD

#include    <16f877A.h> // microcontrolador utilizado
#fuses      xt,wdt,noprotect,put,brownout,nolvp,nocpd,nowrt   //   configuração   dos
fusíveis
#use  delay(clock=4000000, RESTART_WDT)
#use  fast_io(a)
#use  fast_io(b)
#use  fast_io(c)
#use  fast_io(d)
#use  fast_io(e)
#byte porta = 0x05
#byte portb = 0x06
#byte portc = 0x07
#byte portd = 0x08
#byte porte = 0x09
#bit rs = porte.0       // via do LCD que sinaliza recepção de dados ou comando
#bit enable = porte.1   // enable do lcd

/******************  Rotina que envia um COMANDO para o LCD  ******************/
void comando_lcd(int caracter)
{
      rs = 0;                 // seleciona o envio de um comando
      portd = caracter;       // carrega o portd com o caracter
      enable = 1 ;            // gera pulso no enable
      delay_us(1);            // espera 3 microssegundos
      enable = 0;             // desce o pino de enable
      delay_us(40);           // espera mínimo 40 microssegundos
      return;                 // retorna
}
```

```
/**************  Rotina que envia um DADO a ser escrito no LCD  **************/
void escreve_lcd(int caracter)
{
      rs = 1;                 // seleciona o envio de um comando
      portd = caracter;       // carrega o portd com o caracter
      enable = 1;             // gera pulso no enable
      delay_us(1);            // espera 3 microssegundos
      enable = 0;             // desce o pino de enable
      delay_us(40);           // espera mínimo 40 microssegundos
      return;                 // retorna
}

/********************  Função para limpar o LCD  ********************/
void limpa_lcd()
{
      comando_lcd(0x01);      // limpa LCD
      delay_ms (2);
      return;
}

/********************  Inicialização do Display de LCD  ********************/
void inicializa_lcd()
{
      comando_lcd(0x30);      // envia comando para inicializar display
      delay_ms(4);            // espera 4 milissegundos
      comando_lcd(0x30);      // envia comando para inicializar display
      delay_us(100);          // espera 100 microssegundos
      comando_lcd(0x30);      // envia comando para inicializar display
      comando_lcd(0x38);      // configura LCD, 8 bits, matriz de 7x5, 2 linhas
      limpa_lcd();            // limpa LCD
      comando_lcd(0x0c);      // display sem cursor
      comando_lcd(0x06);      // desloca cursor para a direita
      return;           // retorna
}

/********************  Tela Principal  ********************/
void tela_principal()
{
      // posiciona o cursor na linha 0, coluna 0
      comando_lcd(0x80);
      // imprime mensagem no LCD
      printf (escreve_lcd, "Ola, mundo!");
      // posiciona o cursor na linha 1, coluna 2
      comando_lcd(0xC0);
      // imprime mensagem no LCD
      printf (escreve_lcd, "Oi, mundo");
      // retorna da função
      return;
}

/******************** Configurações do PIC  ********************/
void main()
{
            // configura microcontrolador
            setup_adc_ports (no_analogs);
            setup_counters (rtcc_internal, WDT_2304MS);
            // configura os tris
            set_tris_a(0b11011111); // configuração da direção dos pinos de I/O
            set_tris_b(0b00000011);
            set_tris_c(0b11111101);
            set_tris_d(0b00000000);
            set_tris_e(0b00000100);
            // inicializa os ports
            porta=0x00;             // limpa porta
            portb=0x00;             // limpa portb
```

```
            portc=0x00;             // limpa portc
            portd=0x00;             // limpa portd
            porte=0x00;             // limpa porte

            tela_principal();       // imprime a tela principal no LCD

/********************  Rotina principal  ********************/
loop:
      while(TRUE)                   // rotina principal
      {
                  RESTART_WDT();    // incia o watch-dog timer
      }
}
```

Parte 2 – Comunicação em 4 vias pela McLAB2

Como parte didática, a codificação foi dividida em várias partes que serão detalhadas a seguir. Observe que todas estas rotinas partem do princípio que o pino R/W do LCD está aterrado, garantindo sempre nível lógico “0”, ou seja, poderemos apenas enviar informações (apenas escrita). As características de leitura do LCD ficam bloqueadas. Por isso, não poderemos verificar o STATUS BIT para saber quando ele estará pronto para processar outra informação. Devido a estes fatores, devemos nos basear no tempo mínimo par cada operação, estabelecida na Tabela 6.

O fato de termos aterrado o pino R/W não é uma deficiência no protótipo e sim uma característica da maioria dos projetos comerciais.

Definição de constantes

Antes de começarmos a programação propriamente dita, precisamos informar quais são os pinos de I/O que serão utilizados pelo nosso LCD. Este procedimento faz com que a portabilidade entre os códigos e entre LCDs seja maximizada. Estas são as linhas que devemos incluir em nosso programa principal, juntamente com outras constantes internas:

```
#define lcd_enable  pin_e1         // pino enable do LCD
#define lcd_rs      pin_e0         // pino rs do LCD
#define lcd_db4          pin_d4    // pino de dados d4 do LCD
#define lcd_db5          pin_d5    // pino de dados d5 do LCD
#define lcd_db6          pin_d6    // pino de dados d6 do LCD
#define lcd_db7          pin_d7    // pino de dados d7 do LCD
```

Rotina de envio de um “nibble”

Esta é a principal rotina para comunicação em 4 vias (depois da inicialização do LCD em si). É justamente através dela que passaremos um “nibble” para o LCD. Esta rotina basicamente assume que um comando ou caractere especificado pela variável DADO será enviado para o LCD. Ela então pega o “nibble” mais baixo da variável e envia para o LCD. Outro ponto diz respeito ao tempo necessário para a duração do pulso de enable. Como boa prática, utilize sempre 1us para evitar instabilidades.

```
/************************************************************************/
/*                     Envio de "Nibble" para o LCD                     */
/************************************************************************/
//Esta rotina lê o "Nibble" inferior de uma variável e envia para o LCD.
void envia_nibble_lcd(int dado)
{
  //Carrega as vias de dados (pinos) do LCD de acordo com o nibble lido
  output_bit(lcd_db4, bit_test(dado,0)); //Carrega DB4 do LCD com o bit DADO<0>
  output_bit(lcd_db5, bit_test(dado,1)); //Carrega DB5 do LCD com o bit DADO<1>
  output_bit(lcd_db6, bit_test(dado,2)); //Carrega DB6 do LCD com o bit DADO<2>
  output_bit(lcd_db7, bit_test(dado,3)); //Carrega DB7 do LCD com o bit DADO<3>

      //Gera um pulso de enable
      output_high(lcd_enable);      // ENABLE = 1
      delay_us(1);                  // Recomendado para estabilizar o LCD
      output_low(lcd_enable); // ENABLE = 0

      return;                 // Retorna ao ponto de chamada da função
}
```

Rotina de envio de um Byte

Como o LCD precisa receber um byte inteiro antes de exibi-lo, precisamos montar esta rotina para que os 8 bits sejam enviados para ele. Outro destaque é que esta rotina ainda diferencia se a variável irá passar um comando ou um caractere a ser exibido. O grande truque desta rotina consiste em pegar a variável DADO, deslocar os 4 bits mais significativos (DADOS<7:4>) e deslocá-los para a direita para que ocupem as posições menos significativas (DADOS<3:0>), ou seja, passaremos o "nibble" superior (mais significativo) para a posição do "nibble" inferior (menos significativo). E por que isso é feito? Lembre-se que a nossa rotina para enviar "nibbles" para o LCD só pega o nível inferior da variável. Então a rotina envia_nibble_lcd() é chamada, transferindo o conteúdo do "nibble" superior para o LCD. Lembre-se, também, que **sempre** devemos transferir primeiramente o "nibble" superior. Atente para o fato que na instrução *envia_nibble_lcd(dado>>4);* o valor da variável passada é deslocada, mas o conteúdo dela não é alterado. É como se utilizássemos uma variável auxiliar para este procedimento.

Como a variável DADO permanece inalterada, podemos transferir agora o "nibble" inferior. Teoricamente, bastaríamos chamar novamente a rotina *envia_nibble_lcd(dado>>4);* mas, como uma norma de "boa prática", nós primeiramente limpamos a parte alta do byte através da instrução "dado & 0x0f". Isso faz com que os 4 bits mais significativos sejam forçados para "0". Observe que aqui também a manipulação e alteração do valor passado como parâmetro não altera o valor original da variável DADO.

Observe que aqui precisamos de, pelo menos 40us para que o LCD processe o byte (na verdade são dois "nibbles") transmitido e exiba-o na tela. Para resumir, dizemos que este é o tempo mínimo para estabilizar o LCD. Repare, também, que este é o tempo mínimo estabelecido pela Tabela 6.

```
/************************************************************************/
/*                        Envio de Byte para o LCD                      */
/************************************************************************/
//Esta rotina irá enviar um dado ou um comando para o LCD conforme abaixo:
//     ENDEREÇO = 0 -> a variável DADO será uma instrução
//     ENDEREÇO = 1 -> a variável DADO será um caractere
void envia_byte_lcd(boolean endereco, int dado)
{
  output_bit(lcd_rs,endereco);  // Seta o bit RS para instrução ou caractere
  delay_us(100);                // Aguarda 100 us para estabilizar o pino do LCD
  output_low(lcd_enable);       // Desativa a linha ENABLE
  envia_nibble_lcd(dado>>4);    // Envia a parte ALTA do dado/comando
  envia_nibble_lcd(dado & 0x0f);// Limpa a parte ALTA e envia a parte BAIXA do
                                // dado/comando
  delay_us(40);                 // Aguarda 40us para estabilizar o LCD
  return;                       // Retorna ao ponto de chamada da função
}
```

Função para Limpar o LCD.

Como este é um procedimento muito comum e pode ocorrer diversa vezes dentro de um programa, foi criada uma rotina simples especificamente para este fim. Este procedimento faz com que durante a compilação, seja gerada um rotina específica e faz com que o código fique mais otimizado após a compilação.

Atente também para o fato de precisarmos aguardar cerca de 2ms antes de retornarmos ao programa. Esta valor também está presente na Tabela 6. Caso esta rotina apresente

instabilidade para o seu LCD em particular, experimente alterar o valor deste delay, lembrando que, como boa prática e de acordo com a Tabela 6, este valor pode variar entre 82us e 4,1ms.

```
/*************************************************************************/
/*                          Função para limpar o LCD                     */
/*************************************************************************/
//
void limpa_lcd()
{
      envia_byte_lcd(0,0x01);       // Envia instrução para limpar o LCD
      delay_ms(2);                  // Aguarda 2ms para estabilizar o LCD
      return;                       // Retorna ao ponto de chamada da função
}
```

Inicialização do LCD

A seguir, temos o código para inicializar um LCD em 4 vias. A primeira ação da rotina é forçar todas as linhas para o nível lógico “0”. O passo seguinte é aguardar 15ms para estabilizar o LCD, após ser energizado. Até este ponto, é o mesmo procedimento para os modos de 8 e 4 vias. Após este ponto, precisamos mandar as informações para inicialização básica do LCD que consiste em enviar três vezes o “nibble” 0x03, ou enviar o byte 0x30. Para efeito de economia de código e utilizando a característica de que a rotina envia_nibble_lcd() trabalha com o “nibble” inferior, será melhor enviarmos o comando 0x03. Repare que devemos aguardar um tempo para estabilizar o LCD. Como este é um procedimento executado apenas 1 vez em todo o ciclo do PIC e verificando que o tempo determinado na maioria dos data sheets são inferiores ou muito próximos a 5ms, adotou-se este valor como intervalo mínimo entre as fases iniciais de configuração, onde devemos enviar 3 comando 0x03. Após isso, vamos enviar o “nibble” para posicionar o cursor para a posição inicial e, ainda seguindo a Tabela 6, devemos aguardar de 40us a 4,1ms . Por padrão foi adotado 1ms. Aqui também cabe a observação de que, se o display apresentar instabilidades, este tempo deve ser alterado. Em seguida é enviado o “nibble” 0x20 que indica que o LCD será configurado para 4 vias de dados, e 1 linha com matriz de 7X5 com cursor. Este valor pode ser alterado a fim de realizar testes, conforme descrito nas Tabelas 2 e 3. Repare que para este último comando, é necessário enviar um byte completo e não apenas um “nibble”.

Estes são os procedimentos iniciais e obrigatoriamente deve seguir esta seqüência. A partir deste ponto podemos entrar com as configurações constantes nas Tabelas 2 e 3 em qualquer ordem. É importante salientar também que as instruções constantes a partir deste ponto, assim como a de posicionamento do cursor (CURSOR HOME) podem ser utilizadas em qualquer ponto do programa e não apenas na inicialização do LCD. Com isso fica fácil implementar, por exemplo, uma rotina do tipo “backspace”.

Os comando seguintes enviam um byte completo (e não apenas um “nibble”) para as configurações de controle do display (Display control) e para os deslocamentos desejados (Entry Mode Set).

Sempre use as Tabelas 2 e 3 como referência.

E para finalizar, existe o comando RETURN para retornar ao ponto onde a rotina foi chamada. Novamente esta é uma boa prática, mas este comando pode ser retirado, uma vez a função sempre retorna ao ponto onde foi chamada.

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Inicializa o LCD                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void inicializa_lcd()
{
    output_low(lcd_db4);       // Garante que o pino DB4 estão em 0 (low)
    output_low(lcd_db5);       // Garante que o pino DB5 estão em 0 (low)
    output_low(lcd_db6);       // Garante que o pino DB6 estão em 0 (low)
    output_low(lcd_db7);       // Garante que o pino DB7 estão em 0 (low)
    output_low(lcd_rs);        // Garante que o pino RS estão em 0 (low)
    output_low(lcd_enable);    // Garante que o pino ENABLE estão em 0 (low)
    delay_ms(15);              // Aguarda 15ms para estabilizar o LCD

    envia_nibble_lcd(0x03);    // Envia comando para inicializar o display
    delay_ms(5);               // Aguarda 5ms para estabilizar o LCD
    envia_nibble_lcd(0x03);    // Envia comando para inicializar o display
    delay_ms(5);               // Aguarda 5ms para estabilizar o LCD
    envia_nibble_lcd(0x03);    // Envia comando para inicializar o display
    delay_ms(5);               // Aguarda 5ms para estabilizar o LCD
    envia_nibble_lcd(0x02);// CURSOR HOME - Envia comando para zerar o contador de
                               // caracteres e retornar à posição inicial (0x80).
    delay_ms(1);               // Aguarda 1ms para estabilizar o LCD
    envia_byte_lcd(0,0x28);    // FUNCTION SET - Configura o LCD para 4 bits,
                               // 2 linhas, fonte 5X7.
    envia_byte_lcd(0,0x0c);    // DISPLAY CONTROL - Display ligado, sem cursor
    limpa_lcd();               // Limpa o LCD
    envia_byte_lcd(0,0x06);    // ENTRY MODE SET - Desloca o cursor para a direita

    return;              // Retorna ao ponto de chamada da função
}
```

Parte 3 – Comunicação em 4 vias pela placa de testes

Para configurar a placa de testes sugerida neste documento, basta alterar as definições das constantes declaradas no início do programa. Com isto fica muito fácil visualizar quão poderosa é esta rotina. Ela pode ser implementada em qualquer projeto que você estiver desenvolvendo.

Definição de constantes

Antes de começarmos a programação propriamente dita, precisamos informar quais são os pinos de I/O que serão utilizados pelo nosso LCD. Este procedimento faz com que a portabilidade entre os códigos e entre LCDs seja maximizada. Estas são as linhas que devemos incluir em nosso programa principal, juntamente com outras constantes internas:

```
#define lcd_enable      pin_b5            // pino enable do LCD
#define lcd_rs          pin_b4      // pino rs do LCD
#define lcd_db4         pin_b0      // pino de dados d4 do LCD
#define lcd_db5         pin_b1      // pino de dados d5 do LCD
#define lcd_db6         pin_b2      // pino de dados d6 do LCD
#define lcd_db7         pin_b3      // pino de dados d7 do LCD
```

Outras alterações

É obvio que, além destas configurações, o tipo de processador, fuses, declaração e inicialização de PORTs e TRIS, assim como as configurações básicas do microcontrolador deverão ser adequados, uma vez que utilizamos o PIC16F628A ao invés do PIC16F877A existente na McLAB2.

Isso ficará bem claro ao comparar os dois códigos referentes a um mesmo exercício.

Exercício 1 – Inicializando o LCD e “Olá Mundo!”

Exercício 1 – Inicializando o LCD e “Olá Mundo”

Como primeiro exercício vamos apenas inicializar o LCD e escrever a frase “Olá Mundo” no display da McLAB2 e da placa de testes. A seguir temos detalhado os testes para cada placa.

McLAB2

O fluxograma necessário para realizar este primeiro exercício é exibido na figura abaixo:

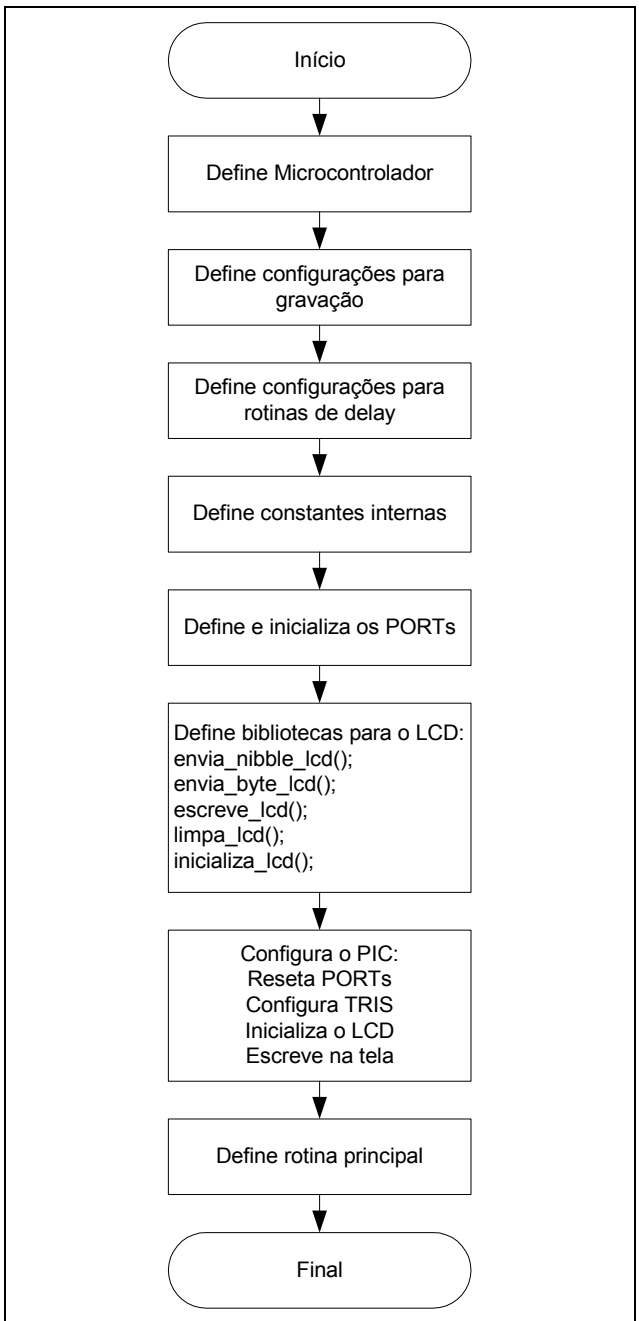


Figura 24: Fluxograma para a placa McLAB2

O código que deverá ser gravada na McLAB2 é o seguinte:

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Tutorial LCD em 4 vias                           *
 *                     Para uso na placa de testes                         *
 *                                                                         *
 *                       Exercício 1 - Olá Mundo!                          *
 *                                                                         *
 *                   Criado por Eduardo de Souza Ramos                     *
 *                                                                         *
 *                  Memory usage:   ROM=4%      RAM=3% - 6%                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                          *
 *   DATA : 31/08/2005                                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Descrição geral                              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//  Primeiro teste de LCD em 4 vias - Olá Mundo

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Definição do PIC utilizado                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#include <16f877a.h>

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                         Configurações para gravação                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#fuses XT,NOWDT,NOPROTECT,PUT,BROWNOUT,NOLVP,NOCPD,NOWRT

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                Definição para uso de Rotinas de Delay                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#use delay(clock=4000000)

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Constantes internas                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Estas são as definições dos pinos que o LCD utiliza.
// Definem quais pinos do PIC controlarão os pinos do LCD
#define lcd_enable      pin_e1      // pino enable do LCD
#define lcd_rs    pin_e0            // pino rs do LCD
#define lcd_db4   pin_d4            // pino de dados d4 do LCD
#define lcd_db5   pin_d5            // pino de dados d5 do LCD
#define lcd_db6   pin_d6            // pino de dados d6 do LCD
#define lcd_db7   pin_d7            // pino de dados d7 do LCD

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Definição e inicialização dos port's                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#use  fast_io(a)   // Inicialização rápida dos Pinos de I/O
#use  fast_io(b)
#use  fast_io(c)
#use  fast_io(d)
#use  fast_io(e)

#byte porta = 0x05
#byte portb = 0x06
#byte portc = 0x07
#byte portd = 0x08
#byte porte = 0x09

/*************************************************************************/
/*                            Rotinas para o LCD                         */
/*************************************************************************/
```

```
//Este é o bloco com as rotinas necessárias para manipular o LCD

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Envio de "Nibble" para o LCD                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Esta rotina lê o "Nibble" inferior de uma variável e envia para o LCD.
void envia_nibble_lcd(int dado)
{
  //Carrega as vias de dados (pinos) do LCD de acordo com o nibble lido
  output_bit(lcd_db4, bit_test(dado,0)); //Carrega DB4 do LCD com o bit DADO<0>
  output_bit(lcd_db5, bit_test(dado,1)); //Carrega DB5 do LCD com o bit DADO<1>
  output_bit(lcd_db6, bit_test(dado,2)); //Carrega DB6 do LCD com o bit DADO<2>
  output_bit(lcd_db7, bit_test(dado,3)); //Carrega DB7 do LCD com o bit DADO<3>

  //Gera um pulso de enable
  output_high(lcd_enable);    // ENABLE = 1
  delay_us(1);                // Recomendado para estabilizar o LCD
  output_low(lcd_enable);     // ENABLE = 0

  return;                     // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Envio de Byte para o LCD                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Esta rotina irá enviar um dado ou um comando para o LCD conforme abaixo:
//     ENDEREÇO = 0 -> a variável DADO será uma instrução
//     ENDEREÇO = 1 -> a variável DADO será um caractere
void envia_byte_lcd(boolean endereco, int dado)
{
  output_bit(lcd_rs,endereco);// Seta o bit RS para instrução ou caractere
  delay_us(100);              // Aguarda 100 us para estabilizar o pino do LCD
  output_low(lcd_enable);     // Desativa a linha ENABLE
  envia_nibble_lcd(dado>>4);  // Envia a parte ALTA do dado/coamndo
  envia_nibble_lcd(dado & 0x0f);// Limpa a parte ALTA e envia a parte BAIXA do

                              // dado/comando
  delay_us(40);               // Aguarda 40us para estabilizar o LCD
  return;                     // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Envio de caractere para o LCD                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Esta rotina serve apenas como uma forma mais fácil de escrever um caractere
// no display. Ela pode ser eliminada e ao invés dela usaremos diretamente a
// função envia_byte_lcd(1,"<caractere a ser mostrado no lcd>"); ou
// envia_byte_lcd(1,<código do caractere a ser mostrado no lcd>);

void escreve_lcd(char c)
// envia caractere para o display
{
  envia_byte_lcd(1,c);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função para limpar o LCD                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Como esta operação pode ser muito utilizada, transformando-a em função
// faz cvom que o código compilado seja menor.
void limpa_lcd()
{
      envia_byte_lcd(0,0x01); // Envia instrução para limpar o LCD
      delay_ms(2);            // Aguarda 2ms para estabilizar o LCD
      return;                 // Retorna ao ponto de chamada da função
}
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Inicializa o LCD                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void inicializa_lcd()
{
  output_low(lcd_db4);  // Garante que o pino DB4 estão em 0 (low)
  output_low(lcd_db5);  // Garante que o pino DB5 estão em 0 (low)
  output_low(lcd_db6);  // Garante que o pino DB6 estão em 0 (low)
  output_low(lcd_db7);  // Garante que o pino DB7 estão em 0 (low)
  output_low(lcd_rs);   // Garante que o pino RS estão em 0 (low)
  output_low(lcd_enable);// Garante que o pino ENABLE estão em 0 (low)
  delay_ms(15);         // Aguarda 15ms para estabilizar o LCD

  envia_nibble_lcd(0x03); // Envia comando para inicializar o display
  delay_ms(5);            // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x03); // Envia comando para inicializar o display
  delay_ms(5);            // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x03); // Envia comando para inicializar o display
  delay_ms(5);            // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x02); // CURSOR HOME - Envia comando para zerar o contador de
                          // caracteres e retornar à posição inicial (0x80).
  delay_ms(1);            // Aguarda 1ms para estabilizar o LCD
  envia_byte_lcd(0,0x28); // FUNCTION SET - Configura o LCD para 4 bits,
                          // 2 linhas, fonte 5X7.
  envia_byte_lcd(0,0x0c); // DISPLAY CONTROL - Display ligado, sem cursor
  limpa_lcd();            // Limpa o LCD
  envia_byte_lcd(0,0x06); // ENTRY MODE SET - Desloca o cursor para a direita

  return;              // Retorna ao ponto de chamada da função
}

/************************************************************************/
/*                      Final das rotinas para o LCD                    */
/************************************************************************/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Configurações do Pic                             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
main()
{
  // Configura o PIC
  setup_adc_ports(no_analogs);

  // Reseta portas
  porta = 0;
  portb = 0;
  portc = 0;
  portd = 0;
  porte = 0;

  // configura os tris
  set_tris_a(0b00111111);  // configuração da direção dos pinos de I/O
  set_tris_b(0b11111111);
  set_tris_c(0b11111111);
  set_tris_d(0b00001111);
  set_tris_e(0b11111100);

  // Inicializa o LCD
  inicializa_lcd();

  //Escreve tela
  escreve_lcd("O");
  escreve_lcd("l");
  escreve_lcd("a");
  escreve_lcd(" ");
```

```
  escreve_lcd("M");
  escreve_lcd("u");
  escreve_lcd("n");
  escreve_lcd("d");
  escreve_lcd("o");
  escreve_lcd("!");

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Rotina principal                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Como não há outra execução, a rotina principal fica vazia
  while (true)
  {

  }

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              Fim do Programa                              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
}
```

O fluxograma necessário para realizar este primeiro exercício é exibido na figura abaixo:

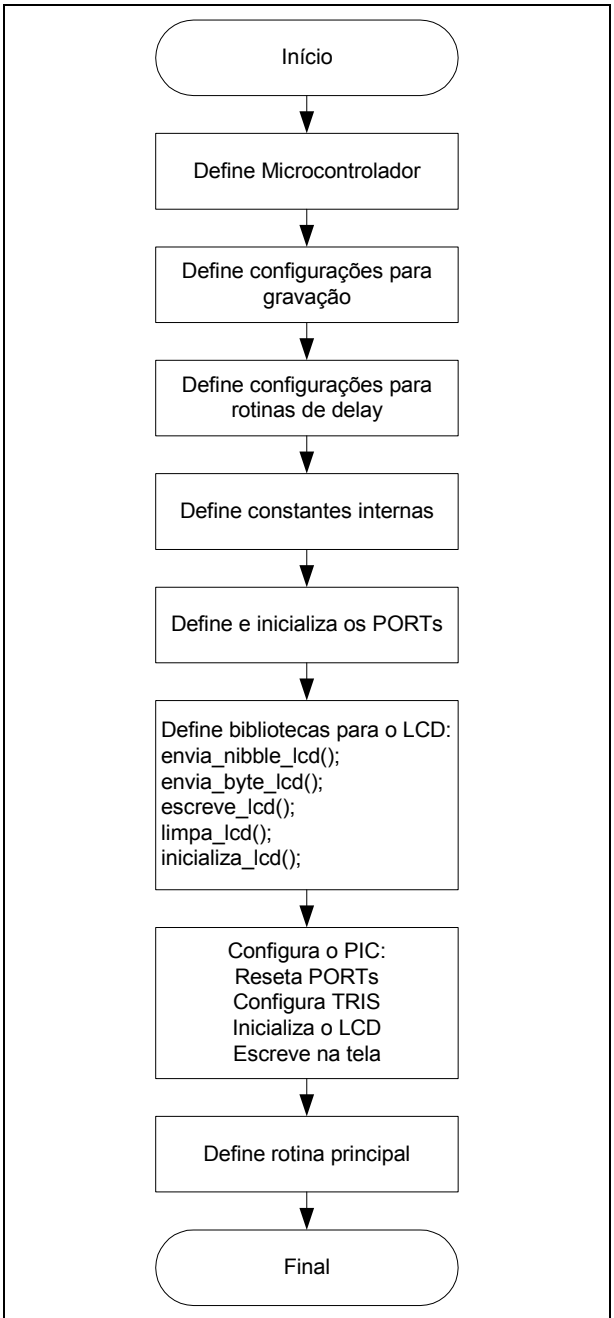


Figura 25: Fluxograma para a placa de testes

Este é o código a ser programado em nossa placa de testes:

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Tutorial LCD em 4 vias                           *
 *                     Para uso na placa de testes                         *
 *                                                                         *
 *                       Exercício 1 - Olá Mundo!                          *
 *                                                                         *
 *                   Criado por Eduardo de Souza Ramos                     *
 *                                                                         *
 *                 Memory usage:   ROM=15%      RAM=3% - 6%                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                          *
 *   DATA : 31/08/2005                                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Descrição geral                              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//  Primeiro teste de LCD em 4 vias - Olá Mundo
//
//
//

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      Definição do PIC utilizado                         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#include <16f628a.h>

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                         Configurações para gravação                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#fuses INTRC,NOWDT,PUT,MCLR,NOBROWNOUT,NOLVP
#ROM 0x07ff = {0} //Calibragem do oscilador interno

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                 Definição para uso de Rotinas de Delay                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#use delay(clock=4000000)

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Constantes internas                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Estas são as definições dos pinos que o LCD utiliza.
// Definem quais pinos do PIC controlarão os pinos do LCD
#define lcd_enable      pin_b5      // pino enable do LCD
#define lcd_rs          pin_b4      // pino rs do LCD
#define lcd_db4         pin_b0      // pino de dados d4 do LCD
#define lcd_db5         pin_b1      // pino de dados d5 do LCD
#define lcd_db6         pin_b2      // pino de dados d6 do LCD
#define lcd_db7         pin_b3      // pino de dados d7 do LCD


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Definição e inicialização dos port's                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#use  fast_io(a)   // Inicialização rápida dos Pinos de I/O
#use  fast_io(b)

#byte porta = 0x05
#byte portb = 0x06


/*************************************************************************/
/*                            Rotinas para o LCD                         */
/*************************************************************************/
```

```
//Este é o bloco com as rotinas necessárias para manipular o LCD

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Envio de "Nibble" para o LCD                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Esta rotina lê o "Nibble" inferior de uma variável e envia para o LCD.
void envia_nibble_lcd(int dado)
{
  //Carrega as vias de dados (pinos) do LCD de acordo com o nibble lido
  output_bit(lcd_db4, bit_test(dado,0)); //Carrega DB4 do LCD com o bit DADO<0>
  output_bit(lcd_db5, bit_test(dado,1)); //Carrega DB5 do LCD com o bit DADO<1>
  output_bit(lcd_db6, bit_test(dado,2)); //Carrega DB6 do LCD com o bit DADO<2>
  output_bit(lcd_db7, bit_test(dado,3)); //Carrega DB7 do LCD com o bit DADO<3>

  //Gera um pulso de enable
  output_high(lcd_enable);    // ENABLE = 1
  delay_us(1);                // Recomendado para estabilizar o LCD
  output_low(lcd_enable);     // ENABLE = 0

  return;                     // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Envio de Byte para o LCD                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Esta rotina irá enviar um dado ou um comando para o LCD conforme abaixo:
//     ENDEREÇO = 0 -> a variável DADO será uma instrução
//     ENDEREÇO = 1 -> a variável DADO será um caractere
void envia_byte_lcd(boolean endereco, int dado)
{
  output_bit(lcd_rs,endereco); // Seta o bit RS para instrução ou caractere
  delay_us(100);               // Aguarda 100 us para estabilizar o pino do LCD
  output_low(lcd_enable);      // Desativa a linha ENABLE
  envia_nibble_lcd(dado>>4);   // Envia a parte ALTA do dado/comando
  envia_nibble_lcd(dado & 0x0f);// Limpa a parte ALTA e envia a parte BAIXA do
                               // dado/comando
  delay_us(40);                // Aguarda 40us para estabilizar o LCD
  return;                      // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Envio de caractere para o LCD                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Esta rotina serve apenas como uma forma mais fácil de escrever um caractere
// no display. Ela pode ser eliminada e ao invés dela usaremos diretamente a
// função envia_byte_lcd(1,"<caractere a ser mostrado no LCD>"); ou
// envia_byte_lcd(1,<código do caractere a ser mostrado no LCD>);

void escreve_lcd(char c)
// envia caractere para o display
{
  envia_byte_lcd(1,c);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                         Função para limpar o LCD                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Como esta operação pode ser muito utilizada, transformando-a em função
// faz com que o código compilado seja menor.
void limpa_lcd()
{
      envia_byte_lcd(0,0x01); // Envia instrução para limpar o LCD
      delay_ms(2);            // Aguarda 2ms para estabilizar o LCD
      return;                 // Retorna ao ponto de chamada da função
}
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Inicializa o LCD                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void inicializa_lcd()
{
  output_low(lcd_db4);        // Garante que o pino DB4 estão em 0 (low)
  output_low(lcd_db5);        // Garante que o pino DB5 estão em 0 (low)
  output_low(lcd_db6);        // Garante que o pino DB6 estão em 0 (low)
  output_low(lcd_db7);        // Garante que o pino DB7 estão em 0 (low)
  output_low(lcd_rs);         // Garante que o pino RS estão em 0 (low)
  output_low(lcd_enable);     // Garante que o pino ENABLE estão em 0 (low)
  delay_ms(15);               // Aguarda 15ms para estabilizar o LCD

  envia_nibble_lcd(0x03);     // Envia comando para inicializar o display
  delay_ms(5);                // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x03);     // Envia comando para inicializar o display
  delay_ms(5);                // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x03);     // Envia comando para inicializar o display
  delay_ms(5);                // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x02);  // CURSOR HOME - Envia comando para zerar o contador de
                              // caracteres e retornar à posição inicial (0x80).
  delay_ms(1);                // Aguarda 1ms para estabilizar o LCD
  envia_byte_lcd(0,0x28);     // FUNCTION SET - Configura o LCD para 4 bits,
                              // 2 linhas, fonte 5X7.
  envia_byte_lcd(0,0x0c);     // DISPLAY CONTROL - Display ligado, sem cursor
  limpa_lcd();                // Limpa o LCD
  envia_byte_lcd(0,0x06);     // ENTRY MODE SET - Desloca o cursor para a direita

  return;                     // Retorna ao ponto de chamada da função
}

/*************************************************************************/
/*                      Final das rotinas para o LCD                     */
/*************************************************************************/


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Configurações do PIC                             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
main()
{
  // Reseta portas
  porta = 0;
  portb = 0;

  // configura os tris
  set_tris_a(0b00111111);  // configuração da direção dos pinos de I/O
  set_tris_b(0b11000000);

  // Inicializa o LCD
  inicializa_lcd();

  //Escreve tela
  escreve_lcd("O");
  escreve_lcd("l");
  escreve_lcd("a");
  escreve_lcd(" ");
  escreve_lcd("M");
  escreve_lcd("u");
  escreve_lcd("n");
  escreve_lcd("d");
  escreve_lcd("o");
  escreve_lcd("!");

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Rotina principal                              *
```

```
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Como não há outra execução, a rotina principal fica vazia
  while (true)
  {

  }


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Fim do Programa                                                      *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
}
```

Repare que neste programa exemplo, o uso de memória foi ROM=15% e RAM=3% - 6%. Esta observação é importante para que possamos comparar com os outros exercícios.

Exercício 2 – A função PRINTF();

Exercício 2 – A função PRINTF()

Depois de termos realizado o Exercício 1, é muito comum termos a seguinte dúvida: “Ok, mas não tem uma forma melhor de escrever no LCD sem ter que declarar caractere por caractere? Isso é muito chato!”.

Sim, existe uma forma onde, além de deixarmos o programa mais enxuto, ainda economizamos espaço de memória! E qual é a mágica? É simples... vamos usar uma função disponível no compilador (neste caso o CCS) chamada PRINTF().

Para que tenhamos uma visão mais clara das vantagens que esta função oferece, vamos manter o mesmo corpo do exercício anterior, mudando apenas a forma de exibir as informações no LCD.

Como no Exercício anterior, vamos abordar o uso desta rotina tanto na McLAB2 quanto na placa de testes.

O fluxograma necessário para realizar este exercício é exibido na figura abaixo:

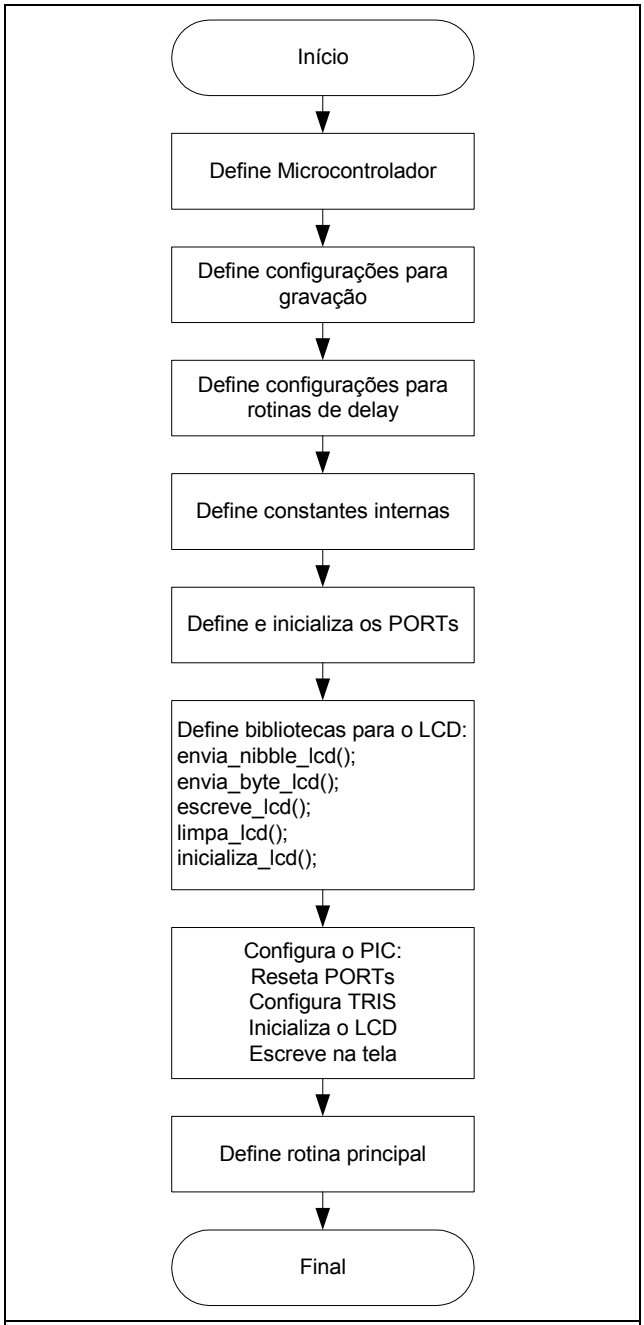


Figura 26: Fluxograma para a placa McLAB2

Este é o código:

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Tutorial LCD em 4 vias                           *
 *                     Para uso na placa de testes                         *
 *                                                                         *
 *                    Exercício 2 - A função PRINTF()                      *
 *                                                                         *
 *                   Criado por Eduardo de Souza Ramos                     *
 *                                                                         *
 *               Memory usage:   ROM=3%      RAM=3% - 6%                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                          *
 *   DATA : 31/08/2005                                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Descrição geral                              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//  Segundo teste de LCD em 4 vias - Olá Mundo utilizando a função PRINTF()
//
//
//

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      Definição do PIC utilizado                         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#include <16f877a.h>

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                         Configurações para gravação                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#fuses XT,NOWDT,NOPROTECT,PUT,BROWNOUT,NOLVP,NOCPD,NOWRT

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                 Definição para uso de Rotinas de Delay                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#use delay(clock=4000000)

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Constantes internas                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Estas são as definições dos pinos que o LCD utiliza.
// Definem quais pinos do PIC controlarão os pinos do LCD
#define lcd_enable      pin_e1      // pino enable do LCD
#define lcd_rs    pin_e0            // pino rs do LCD
#define lcd_db4   pin_d4            // pino de dados d4 do LCD
#define lcd_db5   pin_d5            // pino de dados d5 do LCD
#define lcd_db6   pin_d6            // pino de dados d6 do LCD
#define lcd_db7   pin_d7            // pino de dados d7 do LCD



/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Definição e inicialização dos port's                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#use  fast_io(a)   // Inicialização rápida dos Pinos de I/O
#use  fast_io(b)
#use  fast_io(c)
#use  fast_io(d)
#use  fast_io(e)

#byte porta = 0x05
#byte portb = 0x06
#byte portc = 0x07
#byte portd = 0x08
```

```
#byte porte = 0x09

/************************************************************************/
/*                            Rotinas para o LCD                        */
/************************************************************************/
//Este é o bloco com as rotinas necessárias para manipular o LCD

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      Envio de "Nibble" para o LCD                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Esta rotina lê o "Nibble" inferior de uma variável e envia para o LCD.
void envia_nibble_lcd(int dado)
{
  //Carrega as vias de dados (pinos) do LCD de acordo com o nibble lido
  output_bit(lcd_db4, bit_test(dado,0)); //Carrega DB4 do LCD com o bit DADO<0>
  output_bit(lcd_db5, bit_test(dado,1)); //Carrega DB5 do LCD com o bit DADO<1>
  output_bit(lcd_db6, bit_test(dado,2)); //Carrega DB6 do LCD com o bit DADO<2>
  output_bit(lcd_db7, bit_test(dado,3)); //Carrega DB7 do LCD com o bit DADO<3>

  //Gera um pulso de enable
  output_high(lcd_enable);    // ENABLE = 1
  delay_us(1);                // Recomendado para estabilizar o LCD
  output_low(lcd_enable);     // ENABLE = 0

  return;                     // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Envio de Byte para o LCD                      *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Esta rotina irá enviar um dado ou um comando para o LCD conforme abaixo:
//     ENDEREÇO = 0 -> a variável DADO será uma instrução
//     ENDEREÇO = 1 -> a variável DADO será um caractere
void envia_byte_lcd(boolean endereco, int dado)
{
  output_bit(lcd_rs,endereco);  // Seta o bit RS para instrução ou caractere
  delay_us(100);                // Aguarda 100 us para estabilizar o pino do LCD
  output_low(lcd_enable);       // Desativa a linha ENABLE
  envia_nibble_lcd(dado>>4);    // Envia a parte ALTA do dado/coamndo
  envia_nibble_lcd(dado & 0x0f);// Limpa a parte ALTA e envia a parte BAIXA do
                                // dado/comando
  delay_us(40);                 // Aguarda 40us para estabilizar o LCD
  return;                       // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

 *                      Envio de caractere para o LCD                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Esta rotina serve apenas como uma forma mais fácil de escrever um caractere
// no display. Ela pode ser eliminada e ao invés dela usaremos diretamente a
// função envia_byte_lcd(1,"<caractere a ser mostrado no lcd>"); ou
// envia_byte_lcd(1,<código do caractere a ser mostrado no lcd>);

void escreve_lcd(char c)
// envia caractere para o display
{
  envia_byte_lcd(1,c);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Função para limpar o LCD                      *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Como esta operação pode ser muito utilizada, transformando-a em função
// faz cvom que o código compilado seja menor.
void limpa_lcd()
```

```
{
      envia_byte_lcd(0,0x01); // Envia instrução para limpar o LCD
      delay_ms(2);            // Aguarda 2ms para estabilizar o LCD
      return;                 // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Inicializa o LCD                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void inicializa_lcd()
{
  output_low(lcd_db4);   // Garante que o pino DB4 estão em 0 (low)
  output_low(lcd_db5);   // Garante que o pino DB5 estão em 0 (low)
  output_low(lcd_db6);   // Garante que o pino DB6 estão em 0 (low)
  output_low(lcd_db7);   // Garante que o pino DB7 estão em 0 (low)
  output_low(lcd_rs);    // Garante que o pino RS estão em 0 (low)
  output_low(lcd_enable);// Garante que o pino ENABLE estão em 0 (low)
  delay_ms(15);          // Aguarda 15ms para estabilizar o LCD

  envia_nibble_lcd(0x03);// Envia comando para inicializar o display
  delay_ms(5);           // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x03);// Envia comando para inicializar o display
  delay_ms(5);           // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x03);// Envia comando para inicializar o display
  delay_ms(5);           // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x02);// CURSOR HOME - Envia comando para zerar o contador de
                         // caracteres e retornar à posição inicial (0x80).
  delay_ms(1);           // Aguarda 1ms para estabilizar o LCD
  envia_byte_lcd(0,0x28);// FUNCTION SET - Configura o LCD para 4 bits,
                         // 2 linhas, fonte 5X7.
  envia_byte_lcd(0,0x0c);// DISPLAY CONTROL - Display ligado, sem cursor
  limpa_lcd();           // Limpa o LCD
  envia_byte_lcd(0,0x06);// ENTRY MODE SET - Desloca o cursor para a direita

  return;               // Retorna ao ponto de chamada da função
}

/*****************************************************************************/
/*                      Final das rotinas para o LCD                         */
/*****************************************************************************/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                         Configurações do Pic                            *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
main()
{
  // Configura o PIC
  setup_adc_ports(no_analogs);

  // Reseta portas
  porta = 0;
  portb = 0;
  portc = 0;
  portd = 0;
  porte = 0;

  // configura os tris
  set_tris_a(0b00111111);  // configuração da direção dos pinos de I/O
  set_tris_b(0b11111111);
  set_tris_c(0b11111111);
  set_tris_d(0b00001111);
  set_tris_e(0b11111100);

  // Inicializa o LCD
  inicializa_lcd();
```

```
  //Escreve tela
  printf(escreve_lcd,"Ola Mundo!");

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              Rotina principal                             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Como não há outra execução, a rotina principal fica vazia
  while (true)
  {

  }

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              Fim do Programa                              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
}
```

O fluxograma necessário para realizar este exercício é exibido na figura abaixo:

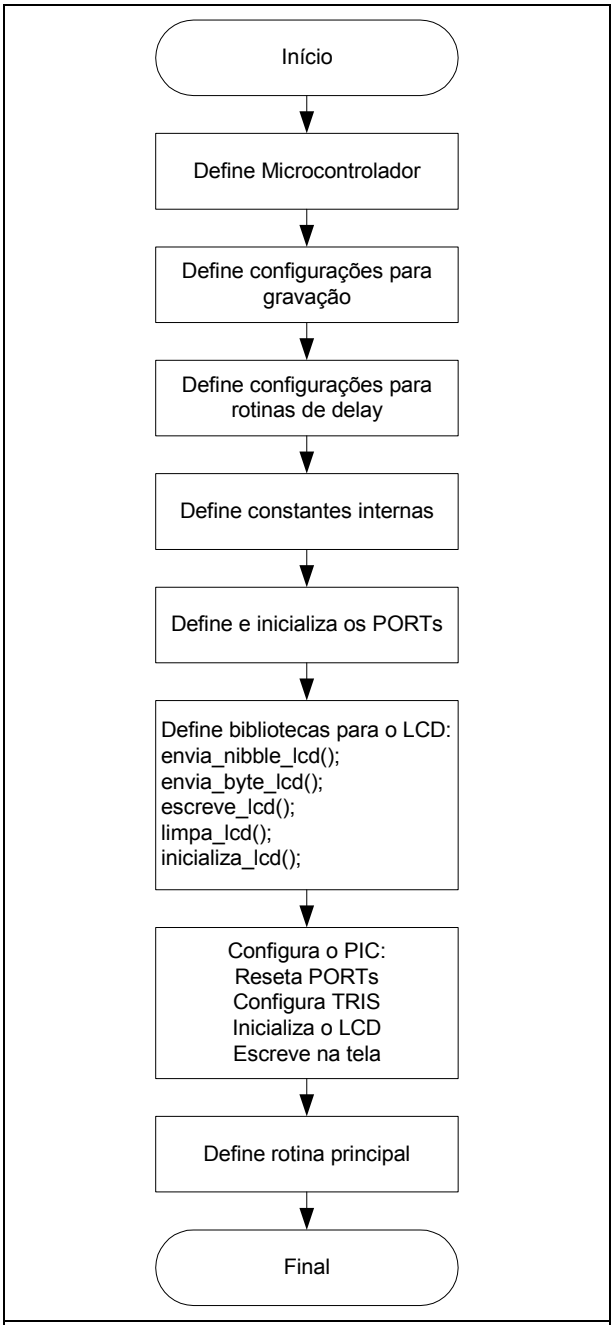


Figura 27: Fluxograma para a placa de testes

Este é o código para a nossa placa de testes:

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Tutorial LCD em 4 vias                           *
 *                     Para uso na placa de testes                         *
 *                                                                         *
 *                    Exercício 2 - A função PRINTF()                      *
 *                                                                         *
*                    Criado por Eduardo de Souza Ramos                     *
 *                                                                         *
 *                Memory usage:   ROM=9%      RAM=3% - 6%                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                          *
 *   DATA : 31/08/2005                                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                             Descrição geral                             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//  Segundo teste de LCD em 4 vias - Olá Mundo utilizando a função PRINTF()
//

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Definição do PIC utilizado                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#include <16f628a.h>

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                         Configurações para gravação                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#fuses INTRC,NOWDT,PUT,MCLR,NOBROWNOUT,NOLVP
#ROM 0x07ff = {0} //Calibragem do oscilador interno

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                Definição para uso de Rotinas de Delay                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#use delay(clock=4000000)

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Constantes internas                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Estas são as definições dos pinos que o LCD utiliza.
// Definem quais pinos do PIC controlarão os pinos do LCD
#define lcd_enable      pin_b5      // pino enable do LCD
#define lcd_rs    pin_b4      // pino rs do LCD
#define lcd_db4   pin_b0      // pino de dados d4 do LCD
#define lcd_db5   pin_b1      // pino de dados d5 do LCD
#define lcd_db6   pin_b2      // pino de dados d6 do LCD
#define lcd_db7   pin_b3      // pino de dados d7 do LCD

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Definição e inicialização dos port's                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#use  fast_io(a)   // Inicialização rápida dos Pinos de I/O
#use  fast_io(b)

#byte porta = 0x05
#byte portb = 0x06

/*************************************************************************/
/*                           Rotinas para o LCD                          */
/*************************************************************************/
//Este é o bloco com as rotinas necessárias para manipular o LCD

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Envio de "Nibble" para o LCD                      *
```

```
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Esta rotina lê o "Nibble" inferior de uma variável e envia para o LCD.
void envia_nibble_lcd(int dado)
{
  //Carrega as vias de dados (pinos) do LCD de acordo com o nibble lido
  output_bit(lcd_db4, bit_test(dado,0)); //Carrega DB4 do LCD com o bit DADO<0>
  output_bit(lcd_db5, bit_test(dado,1)); //Carrega DB5 do LCD com o bit DADO<1>
  output_bit(lcd_db6, bit_test(dado,2)); //Carrega DB6 do LCD com o bit DADO<2>
  output_bit(lcd_db7, bit_test(dado,3)); //Carrega DB7 do LCD com o bit DADO<3>

  //Gera um pulso de enable
  output_high(lcd_enable);    // ENABLE = 1
  delay_us(1);                // Recomendado para estabilizar o LCD
  output_low(lcd_enable);     // ENABLE = 0

  return;                     // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Envio de Byte para o LCD                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Esta rotina irá enviar um dado ou um comando para o LCD conforme abaixo:
//     ENDEREÇO = 0 -> a variável DADO será uma instrução
//     ENDEREÇO = 1 -> a variável DADO será um caractere
void envia_byte_lcd(boolean endereco, int dado)
{
  output_bit(lcd_rs,endereco);  // Seta o bit RS para instrução ou caractere
  delay_us(100);                // Aguarda 100 us para estabilizar o pino do LCD
  output_low(lcd_enable);       // Desativa a linha ENABLE
  envia_nibble_lcd(dado>>4);    // Envia a parte ALTA do dado/comando
  envia_nibble_lcd(dado & 0x0f);// Limpa a parte ALTA e envia a parte BAIXA do
                                // dado/comando
  delay_us(40);                 // Aguarda 40us para estabilizar o LCD
  return;                       // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Envio de caractere para o LCD                                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Esta rotina serve apenas como uma forma mais fácil de escrever um caractere
// no display. Ela pode ser eliminada e ao invés dela usaremos diretamente a
// função envia_byte_lcd(1,"<caractere a ser mostrado no LCD>"); ou
// envia_byte_lcd(1,<código do caractere a ser mostrado no LCD>);

void escreve_lcd(char c)
// envia caractere para o display
{
  envia_byte_lcd(1,c);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                         Função para limpar o LCD                                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Como esta operação pode ser muito utilizada, transformando-a em função
// faz com que o código compilado seja menor.
void limpa_lcd()
{
      envia_byte_lcd(0,0x01); // Envia instrução para limpar o LCD
      delay_ms(2);            // Aguarda 2ms para estabilizar o LCD
      return;                 // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Inicializa o LCD                                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void inicializa_lcd()
```

```
{
  output_low(lcd_db4);  // Garante que o pino DB4 estão em 0 (low)
  output_low(lcd_db5);        // Garante que o pino DB5 estão em 0 (low)
  output_low(lcd_db6);        // Garante que o pino DB6 estão em 0 (low)
  output_low(lcd_db7);        // Garante que o pino DB7 estão em 0 (low)
  output_low(lcd_rs);         // Garante que o pino RS estão em 0 (low)
  output_low(lcd_enable);     // Garante que o pino ENABLE estão em 0 (low)
  delay_ms(15);               // Aguarda 15ms para estabilizar o LCD

  envia_nibble_lcd(0x03);     // Envia comando para inicializar o display
  delay_ms(5);                // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x03);     // Envia comando para inicializar o display
  delay_ms(5);                // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x03);     // Envia comando para inicializar o display
  delay_ms(5);                // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x02);  // CURSOR HOME - Envia comando para zerar o contador de
                              // caracteres e retornar à posição inicial (0x80).
  delay_ms(1);                // Aguarda 1ms para estabilizar o LCD
  envia_byte_lcd(0,0x28);     // FUNCTION SET - Configura o LCD para 4 bits,
                              // 2 linhas, fonte 5X7.
  envia_byte_lcd(0,0x0c);     // DISPLAY CONTROL - Display ligado, sem cursor
  limpa_lcd();                // Limpa o LCD
  envia_byte_lcd(0,0x06);     // ENTRY MODE SET - Desloca o cursor para a direita

  return;                     // Retorna ao ponto de chamada da função
}

/************************************************************************/
/*                      Final das rotinas para o LCD                    */
/************************************************************************/


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Configurações do PIC                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
main()
{
  // Reseta portas
  porta = 0;
  portb = 0;

  // configura os tris
  set_tris_a(0b00111111);  // configuração da direção dos pinos de I/O
  set_tris_b(0b11000000);

  // Inicializa o LCD
  inicializa_lcd();

  //Escreve tela
  printf(escreve_lcd,"Ola Mundo!");

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Rotina principal                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Como não há outra execução, a rotina principal fica vazia
  while (true)
  {

  }


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Fim do Programa                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
}
```

Repare que neste programa exemplo, o uso de memória foi ROM=9% e RAM=3% - 6%. Se compararmos com o exercício anterior, veremos que houve um ganho substancial de memória (6 pontos percentuais) que em alguns casos pode representar o sucesso de seu projeto! Portanto, use esta função e seja feliz!!!

Maiores detalhes sobre esta função e como imprimir o conteúdo de variáveis pode ser encontrada no help do compilador CCS

Exercício 3 – Rotacionando o display

Exercício 3 – Rotacionando o display

Conforme dito antes, independentemente do tamanho do display, há sempre 80 posições em cada linha do display para armazenar caracteres. Mas preste atenção que, nos displays de 4 linhas, a terceira linha é continuação da primeira e a quarta e a continuação da segunda. Verifique isto na Figura 13.

Mas, como não há espaço disponível para visualizar todos estes espaços, há um modo de deslocar a telado LCD de forma que temos acesso a todo o conteúdo armazenado em cada linha. Este é um procedimento muito simples, mas há uma pegadinha... ele altera a relação dos endereços com as posições na tela. Portanto, se você realizar diversas rotações, é uma boa prática utilizar o comando envia_byte_lcd(0,0x02) para que o comando CURSOR HOME seja executado, direcionando o cursor para a posição 0x00 (ou 0x80 se levarmos em conta que o bit 7 deve ser sempre “1”, ao nos referenciarmos a um endereço) e deslocando o display para o estado original. Não se esqueça que este comando não altera a memória RAM do LCD. Não use o comando CLEAR DISPLAY (0x01), pois neste caso, todo o conteúdo da memória RAM poderá será perdido.

Se entrarmos com mais de 16 caracteres por linha (no nosso caso, uma vez que usamos um LCD de 16 posições), os caracteres excedentes ficarão fora da linha de visão, no lado “direito” da tela.

Então, poderemos usar a função envia_byte_lcd(0,0x18) para que todos os endereços da tela sejam deslocadas para a esquerda. Isto dá a sensação de que a tela deslocou para a direita. É como se tivéssemos uma lupa em cima do texto.

Da mesma forma, poderemos utilizar a função envia_byte_lcd(0,0x1c) para realizar a operação inversa.

Neste exemplo mostramos como podemos criar um mostrador onde o texto apresenta o clássico movimento de “vai-e-vem”.

Como vocês já devem ter reparado nos dois exemplos anteriores, o fluxograma é o mesmo, portanto, para este exemplo, não vamos fugir à regra. Devido a isto, não iremos repetir o mesmo desenho, passando diretamente para a codificação para a placa McLAB2 e para a nossa placa de testes.

Este é o código para a McLAB2

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Tutorial LCD em 4 vias                           *
 *                     Para uso na placa de testes                         *
 *                                                                         *
 *          Exercício 3 - Deslocando e rotacionando o display              *
 *                                                                         *
 *                   Criado por Eduardo de Souza Ramos                     *
 *                                                                         *
 *                Memory usage:   ROM=4%      RAM=3% - 7%                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                          *
 *   DATA : 31/08/2005                                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Descrição geral                              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//  Neste exercício, o display será deslocado para a esquerda e, ao final
//  da mensagem ele será deslocado para a direita, criando um efeito de
//  vai-e-vem. Serão utilizadas as duas linhas do LCD para ter uma melhor
//  visualização.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Definição do PIC utilizado                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#include <16f877a.h>
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                         Configurações para gravação                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#fuses XT,NOWDT,NOPROTECT,PUT,BROWNOUT,NOLVP,NOCPD,NOWRT
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                 Definição para uso de Rotinas de Delay                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#use delay(clock=4000000)

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Constantes internas                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Estas são as definições dos pinos que o LCD utiliza.
#define lcd_enable      pin_e1            // pino enable do LCD
#define lcd_rs    pin_e0            // pino rs do LCD
#define lcd_db4   pin_d4            // pino de dados d4 do LCD
#define lcd_db5   pin_d5            // pino de dados d5 do LCD
#define lcd_db6   pin_d6            // pino de dados d6 do LCD
#define lcd_db7   pin_d7            // pino de dados d7 do LCD

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Definição e inicialização dos port's                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#use  fast_io(a)   // Inicialização rápida dos Pinos de I/O
#use  fast_io(b)
#use  fast_io(c)
#use  fast_io(d)
#use  fast_io(e)

#byte porta = 0x05
#byte portb = 0x06
#byte portc = 0x07
#byte portd = 0x08
#byte porte = 0x09
```

```
/************************************************************************/
/*                            Rotinas para o LCD                        */
/************************************************************************/
//Este é o bloco com as rotinas necessárias para manipular o LCD
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Envio de "Nibble" para o LCD                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Esta rotina lê o "Nibble" inferior de uma variável e envia para o LCD.
void envia_nibble_lcd(int dado)
{
  //Carrega as vias de dados (pinos) do LCD de acordo com o nibble lido
  output_bit(lcd_db4, bit_test(dado,0)); //Carrega DB4 do LCD com o bit DADO<0>
  output_bit(lcd_db5, bit_test(dado,1)); //Carrega DB5 do LCD com o bit DADO<1>
  output_bit(lcd_db6, bit_test(dado,2)); //Carrega DB6 do LCD com o bit DADO<2>
  output_bit(lcd_db7, bit_test(dado,3)); //Carrega DB7 do LCD com o bit DADO<3>
  //Gera um pulso de enable
  output_high(lcd_enable);               // ENABLE = 1
  delay_us(1);                           // Recomendado para estabilizar o LCD
  output_low(lcd_enable);                // ENABLE = 0
  return;                                // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Envio de Byte para o LCD                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Esta rotina irá enviar um dado ou um comando para o LCD conforme abaixo:
//     ENDEREÇO = 0 -> a variável DADO será uma instrução
//     ENDEREÇO = 1 -> a variável DADO será um caractere
void envia_byte_lcd(boolean endereco, int dado)
{
  output_bit(lcd_rs,endereco);  // Seta o bit RS para instrução ou caractere
  delay_us(100);                // Aguarda 100 us para estabilizar o pino do LCD
  output_low(lcd_enable);       // Desativa a linha ENABLE
  envia_nibble_lcd(dado>>4);    // Envia a parte ALTA do dado/coamndo
  envia_nibble_lcd(dado & 0x0f);// Limpa a parte ALTA e envia a parte BAIXA do
                                // dado/comando
  delay_us(40);                 // Aguarda 40us para estabilizar o LCD
  return;                       // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Envio de caractere para o LCD                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Esta rotina serve apenas como uma forma mais fácil de escrever um caractere
// no display. Ela pode ser eliminada e ao invés dela usaremos diretamente a
// função envia_byte_lcd(1,"<caractere a ser mostrado no lcd>"); ou
// envia_byte_lcd(1,<código do caractere a ser mostrado no lcd>);

void escreve_lcd(char c)
// envia caractere para o display
{
  envia_byte_lcd(1,c);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função para limpar o LCD                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Como esta operação pode ser muito utilizada, transformando-a em função
// faz cvom que o código compilado seja menor.
void limpa_lcd()
{
      envia_byte_lcd(0,0x01); // Envia instrução para limpar o LCD
      delay_ms(2);            // Aguarda 2ms para estabilizar o LCD
      return;                 // Retorna ao ponto de chamada da função
}
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Inicializa o LCD                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void inicializa_lcd()
{
  output_low(lcd_db4);  // Garante que o pino DB4 estão em 0 (low)
  output_low(lcd_db5);  // Garante que o pino DB5 estão em 0 (low)
  output_low(lcd_db6);  // Garante que o pino DB6 estão em 0 (low)
  output_low(lcd_db7);  // Garante que o pino DB7 estão em 0 (low)
  output_low(lcd_rs);   // Garante que o pino RS estão em 0 (low)
  output_low(lcd_enable);// Garante que o pino ENABLE estão em 0 (low)
  delay_ms(15);         // Aguarda 15ms para estabilizar o LCD
  envia_nibble_lcd(0x03);// Envia comando para inicializar o display
  delay_ms(5);          // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x03);// Envia comando para inicializar o display
  delay_ms(5);          // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x03);// Envia comando para inicializar o display
  delay_ms(5);          // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x02);// CURSOR HOME - Envia comando para zerar o contador de
                        // caracteres e retornar à posição inicial (0x80).
  delay_ms(1);          // Aguarda 1ms para estabilizar o LCD
  envia_byte_lcd(0,0x28);// FUNCTION SET - Configura o LCD para 4 bits,
                        // 2 linhas, fonte 5X7.
  envia_byte_lcd(0,0x0c);// DISPLAY CONTROL - Display ligado, sem cursor
  limpa_lcd();          // Limpa o LCD
  envia_byte_lcd(0,0x06);// ENTRY MODE SET - Desloca o cursor para a direita
  return;               // Retorna ao ponto de chamada da função
}

/*************************************************************************/
/*                      Final das rotinas para o LCD                     */
/*************************************************************************/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Configurações do Pic                             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
main()
{
  // Configura o PIC
  setup_adc_ports(no_analogs);
  // Reseta portas
  porta = 0;
  portb = 0;
  portc = 0;
  portd = 0;
  porte = 0;
  // configura os tris
  set_tris_a(0b00111111);  // configuração da direção dos pinos de I/O
  set_tris_b(0b11111111);
  set_tris_c(0b11111111);
  set_tris_d(0b00001111);
  set_tris_e(0b11111100);
  // Inicializa o LCD
  inicializa_lcd();
  //Escreve tela
  printf(escreve_lcd,"Teste 1-linha com mais de 16 caracteres");
  envia_byte_lcd(0,0xc0); //Desloca o cursor para a segunda linha do LCD
  printf(escreve_lcd,"Teste 2-linha comum, + de 16 caracteres");

 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Rotina principal                              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Esta rotina é a responsável pelo movimento de vai-e-vem do display.
// Ela fica em loop infinito.;
  while (true)
```

```
  {
      int i;
      //Desloca os caracteres do LCD para a esquerda
      for(i=0;i<=22;i++)
      {
              envia_byte_lcd(0,0x18); //0x18 = 0b00011000
              delay_ms(250);
      }
      //Desloca os caracteres do LCD para a direita
      for(i=0;i<=22;i++)
      {
              envia_byte_lcd(0,0x1c); //0x1c = 0b00011100
              delay_ms(250);
      }
  }

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              Fim do Programa                                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
}
```

O seguinte código é o que deverá ser executado em nossa placa de testes:

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Tutorial LCD em 4 vias                           *
 *                     Para uso na placa de testes                         *
 *                                                                         *
 *          Exercício 3 - Deslocando e rotacionando o display              *
 *                                                                         *
*                    Criado por Eduardo de Souza Ramos                     *
 *                                                                         *
 *               Memory usage:   ROM=15%      RAM=3% - 7%                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                          *
 *   DATA : 31/08/2005                                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Descrição geral                              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//  Neste exercício, o display será deslocado para a esquerda e, ao final
//  da mensagem ele será deslocado para a direita, criando um efeito de
//  vai-e-vem. Serão utilizadas as duas linhas do LCD para ter uma melhor
//  visualização.

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Definição do PIC utilizado                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#include <16f628a.h>

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Configurações para gravação                      *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#fuses INTRC,NOWDT,PUT,MCLR,NOBROWNOUT,NOLVP
#ROM 0x07ff = {0} //Calibragem do oscilador interno

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                 Definição para uso de Rotinas de Delay                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#use delay(clock=4000000)

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Constantes internas                            *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Estas são as definições dos pinos que o LCD utiliza.
// Definem quais pinos do PIC controlarão os pinos do LCD
#define lcd_enable      pin_b5      // pino enable do LCD
#define lcd_rs    pin_b4      // pino rs do LCD
#define lcd_db4   pin_b0      // pino de dados d4 do LCD
#define lcd_db5   pin_b1      // pino de dados d5 do LCD
#define lcd_db6   pin_b2      // pino de dados d6 do LCD
#define lcd_db7   pin_b3      // pino de dados d7 do LCD

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Definição e inicialização dos port's                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#use  fast_io(a)   // Inicialização rápida dos Pinos de I/O
#use  fast_io(b)

#byte porta = 0x05
#byte portb = 0x06

/*************************************************************************/
```

```
/*                            Rotinas para o LCD                              */
/****************************************************************************/
//Este é o bloco com as rotinas necessárias para manipular o LCD

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Envio de "Nibble" para o LCD                      *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Esta rotina lê o "Nibble" inferior de uma variável e envia para o LCD.
void envia_nibble_lcd(int dado)
{
  //Carrega as vias de dados (pinos) do LCD de acordo com o nibble lido
  output_bit(lcd_db4, bit_test(dado,0)); //Carrega DB4 do LCD com o bit DADO<0>
  output_bit(lcd_db5, bit_test(dado,1)); //Carrega DB5 do LCD com o bit DADO<1>
  output_bit(lcd_db6, bit_test(dado,2)); //Carrega DB6 do LCD com o bit DADO<2>
  output_bit(lcd_db7, bit_test(dado,3)); //Carrega DB7 do LCD com o bit DADO<3>

  //Gera um pulso de enable
  output_high(lcd_enable);    // ENABLE = 1
  delay_us(1);                // Recomendado para estabilizar o LCD
  output_low(lcd_enable);     // ENABLE = 0
  return;                     // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Envio de Byte para o LCD                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Esta rotina irá enviar um dado ou um comando para o LCD conforme abaixo:
//     ENDEREÇO = 0 -> a variável DADO será uma instrução
//     ENDEREÇO = 1 -> a variável DADO será um caractere
void envia_byte_lcd(boolean endereco, int dado)
{
  output_bit(lcd_rs,endereco);  // Seta o bit RS para instrução ou caractere
  delay_us(100);                // Aguarda 100 us para estabilizar o pino do LCD
  output_low(lcd_enable);       // Desativa a linha ENABLE
  envia_nibble_lcd(dado>>4);    // Envia a parte ALTA do dado/comando
  envia_nibble_lcd(dado & 0x0f);// Limpa a parte ALTA e envia a parte BAIXA do
                                // dado/comando
  delay_us(40);                 // Aguarda 40us para estabilizar o LCD
  return;                       // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Envio de caractere para o LCD                      *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Esta rotina serve apenas como uma forma mais fácil de escrever um caractere
// no display. Ela pode ser eliminada e ao invés dela usaremos diretamente a
// função envia_byte_lcd(1,"<caractere a ser mostrado no LCD>"); ou
// envia_byte_lcd(1,<código do caractere a ser mostrado no LCD>);

void escreve_lcd(char c)
// envia caractere para o display
{
  envia_byte_lcd(1,c);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função para limpar o LCD                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Como esta operação pode ser muito utilizada, transformando-a em função
// faz com que o código compilado seja menor.
void limpa_lcd()
{
      envia_byte_lcd(0,0x01); // Envia instrução para limpar o LCD
      delay_ms(2);            // Aguarda 2ms para estabilizar o LCD
      return;                 // Retorna ao ponto de chamada da função
}
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Inicializa o LCD                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void inicializa_lcd()
{
  output_low(lcd_db4);        // Garante que o pino DB4 estão em 0 (low)
  output_low(lcd_db5);        // Garante que o pino DB5 estão em 0 (low)
  output_low(lcd_db6);        // Garante que o pino DB6 estão em 0 (low)
  output_low(lcd_db7);        // Garante que o pino DB7 estão em 0 (low)
  output_low(lcd_rs);         // Garante que o pino RS estão em 0 (low)
  output_low(lcd_enable);     // Garante que o pino ENABLE estão em 0 (low)
  delay_ms(15);               // Aguarda 15ms para estabilizar o LCD

  envia_nibble_lcd(0x03);  // Envia comando para inicializar o display
  delay_ms(5);             // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x03);  // Envia comando para inicializar o display
  delay_ms(5);             // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x03);  // Envia comando para inicializar o display
  delay_ms(5);             // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x02);  // CURSOR HOME - Envia comando para zerar o contador de
                           // caracteres e retornar à posição inicial (0x80).
  delay_ms(1);             // Aguarda 1ms para estabilizar o LCD
  envia_byte_lcd(0,0x28);  // FUNCTION SET - Configura o LCD para 4 bits,
                           // 2 linhas, fonte 5X7.
  envia_byte_lcd(0,0x0c);  // DISPLAY CONTROL - Display ligado, sem cursor
  limpa_lcd();             // Limpa o LCD
  envia_byte_lcd(0,0x06);  // ENTRY MODE SET - Desloca o cursor para a direita

  return;                  // Retorna ao ponto de chamada da função
}

/************************************************************************/
/*                     Final das rotinas para o LCD                     */
/************************************************************************/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                         Configurações do PIC                            *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
main()
{
  // Reseta portas
  porta = 0;
  portb = 0;

  // configura os tris
  set_tris_a(0b00111111);  // configuração da direção dos pinos de I/O
  set_tris_b(0b11000000);

  // Inicializa o LCD
  inicializa_lcd();

  //Escreve tela
  printf(escreve_lcd,"Teste 1-linha com mais de 16 caracteres");
  envia_byte_lcd(0,0xc0); //Desloca o cursor para a segunda linha do LCD
  printf(escreve_lcd,"Teste 2-linha comum, + de 16 caracteres");

 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Rotina principal                              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Esta rotina é a responsável pelo movimento de vai-e-vem do display.
// Ela fica em loop infinito.;
  while (true)
  {
      int i;
      //Desloca os caracteres do LCD para a esquerda
```

```
      for(i=0;i<=22;i++)
      {
              envia_byte_lcd(0,0x18); //0x18 = 0b00011000
              delay_ms(250);
      }
      //Desloca os caracteres do LCD para a direita
      for(i=0;i<=22;i++)
      {
              envia_byte_lcd(0,0x1c); //0x1c = 0b00011100
              delay_ms(250);
      }
  }

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              Fim do Programa                              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
}
```

Faça um teste... Comentes as linhas que direcionam os caracteres do LCD para a direita e veja o que acontece...

Exercício 4 – Criando caracteres

Exercício 4 – Criando caracteres

Os endereços da CGRAM de 0x40 até 0x7F são usados para que armazenemos os caracteres desenhados por nós.

Para isso, precisaremos, logo no início do programa, definir quais são os valores dos caracteres, conforme explicado no Capítulo 1 – Parte 7.

Este exemplo, para as placas McLAB2 e a nossa placa de testes, utiliza o mesmo fluxograma, que é exibido a seguir.

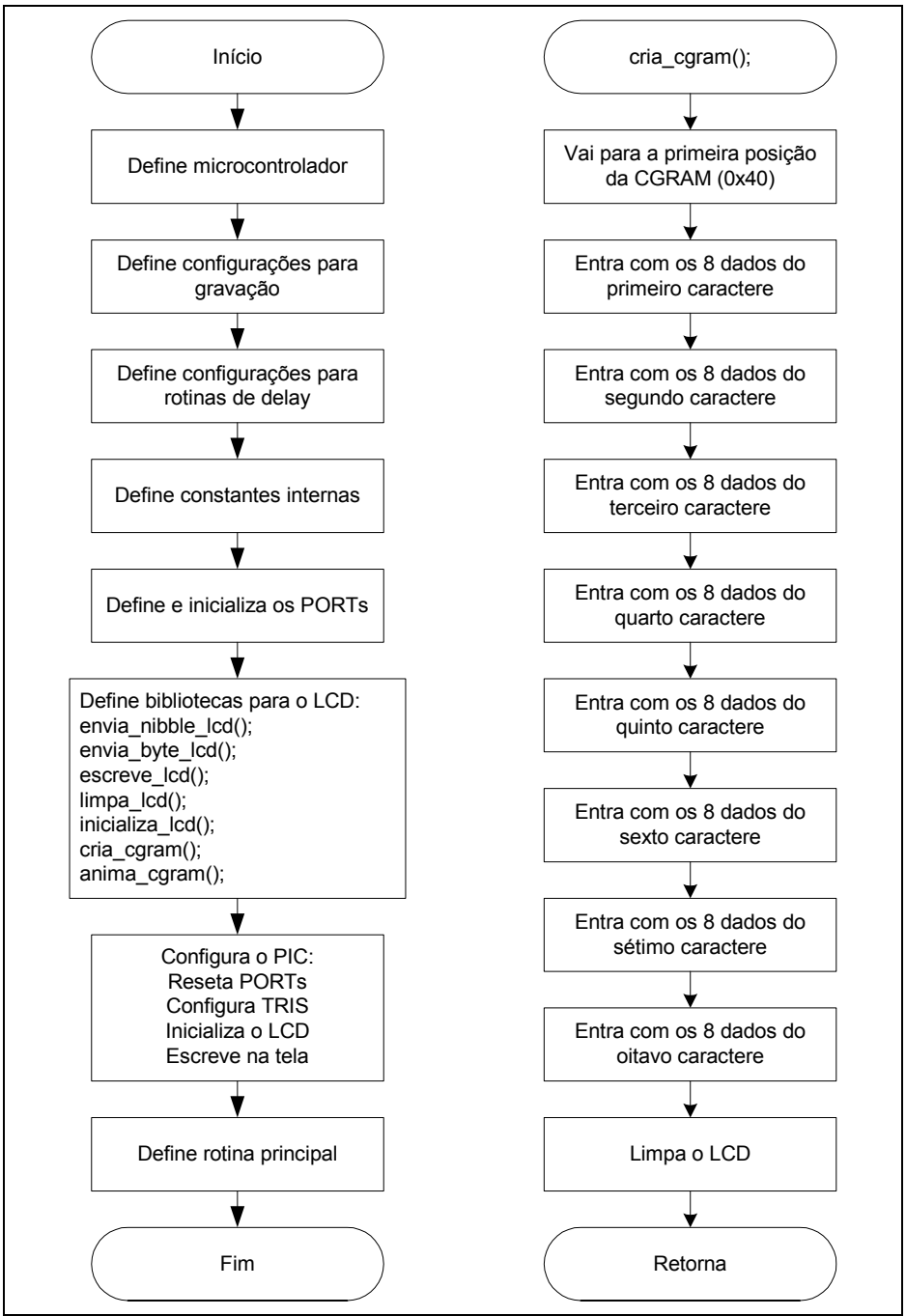


Figura 28: Fluxograma para o exemplo 4

Este é o código que deverá ser implementado em nossa placa de testes:

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Tutorial LCD em 4 vias                         *
 *                     Para uso na placa de testes                       *
 *                                                                       *
 *               Exercício 4 - Criando caracteres na CGRAM               *
 *                                                                       *
 *                    Criado por Eduardo de Souza Ramos                  *
 *                                                                       *
 *                  Memory usage:   ROM=7%      RAM=3% - 6%              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                        *
 *   DATA : 31/08/2005                                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Descrição geral                             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//     Neste exercício, criaremos 8 caracteres especiais (desenhos) na CGRAM
//  para utilizarmos em qualquer ponto do LCD
//     A priori eles serão exibidos nas 8 primeiras posições da primeira
//  linha do LCD. Maseles poderão ser utilizados em qualquer ponto.
//     Como estes caracteres ficam na CGRAM, são voláteis, ou seja, ao
//  desligarmos o módulo e o ligarmos novamente, precisarmos reintroduzir
//  o código para a criação de caracteres.
//     Reparem que, apesar de termos 16 posições disponíveis na CGRAM,
//  somente poderemos criar 8 caracteres. As 8 posições adjacentes (posições
//  0x08 a 0x0f) são espelho da primeira (0x00 a 0x07)

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Definição do PIC utilizado                      *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#include <16f877a.h>

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                         Configurações para gravação                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#fuses XT,NOWDT,NOPROTECT,PUT,BROWNOUT,NOLVP,NOCPD,NOWRT

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                 Definição para uso de Rotinas de Delay                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#use delay(clock=4000000)

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

 *                           Constantes internas                         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Estas são as definições dos pinos que o LCD utiliza.
// Definem quais pinos do PIC controlarão os pinos do LCD
#define lcd_enable      pin_e1            // pino enable do LCD
#define lcd_rs    pin_e0            // pino rs do LCD
#define lcd_db4   pin_d4            // pino de dados d4 do LCD
#define lcd_db5   pin_d5            // pino de dados d5 do LCD
#define lcd_db6   pin_d6            // pino de dados d6 do LCD
#define lcd_db7   pin_d7            // pino de dados d7 do LCD

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Definição e inicialização dos port's                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
```

```
#use  fast_io(a)   // Inicialização rápida dos Pinos de I/O
#use  fast_io(b)
#use  fast_io(c)
#use  fast_io(d)
#use  fast_io(e)

#byte porta = 0x05
#byte portb = 0x06
#byte portc = 0x07
#byte portd = 0x08
#byte porte = 0x09

/************************************************************************/
/*                            Rotinas para o LCD                        */
/************************************************************************/
//Este é o bloco com as rotinas necessárias para manipular o LCD

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Envio de "Nibble" para o LCD                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Esta rotina lê o "Nibble" inferior de uma variável e envia para o LCD.
void envia_nibble_lcd(int dado)
{
  //Carrega as vias de dados (pinos) do LCD de acordo com o nibble lido
  output_bit(lcd_db4, bit_test(dado,0)); //Carrega DB4 do LCD com o bit DADO<0>
  output_bit(lcd_db5, bit_test(dado,1)); //Carrega DB5 do LCD com o bit DADO<1>
  output_bit(lcd_db6, bit_test(dado,2)); //Carrega DB6 do LCD com o bit DADO<2>
  output_bit(lcd_db7, bit_test(dado,3)); //Carrega DB7 do LCD com o bit DADO<3>

  //Gera um pulso de enable
  output_high(lcd_enable);    // ENABLE = 1
  delay_us(1);                // Recomendado para estabilizar o LCD
  output_low(lcd_enable);     // ENABLE = 0

  return;                     // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                         Envio de Byte para o LCD                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Esta rotina irá enviar um dado ou um comando para o LCD conforme abaixo:
//     ENDEREÇO = 0 -> a variável DADO será uma instrução
//     ENDEREÇO = 1 -> a variável DADO será um caractere
void envia_byte_lcd(boolean endereco, int dado)
{
  output_bit(lcd_rs,endereco);  // Seta o bit RS para instrução ou caractere
  delay_us(100);                // Aguarda 100 us para estabilizar o pino do LCD
  output_low(lcd_enable);       // Desativa a linha ENABLE
  envia_nibble_lcd(dado>>4);    // Envia a parte ALTA do dado/coamndo
  envia_nibble_lcd(dado & 0x0f);// Limpa a parte ALTA e envia a parte BAIXA do
                                // dado/comando
  delay_us(40);                 // Aguarda 40us para estabilizar o LCD
  return;                       // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Envio de caractere para o LCD                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Esta rotina serve apenas como uma forma mais fácil de escrever um caractere
// no display. Ela pode ser eliminada e ao invés dela usaremos diretamente a
// função envia_byte_lcd(1,"<caractere a ser mostrado no lcd>"); ou
// envia_byte_lcd(1,<código do caractere a ser mostrado no lcd>);

void escreve_lcd(char c)
// envia caractere para o display
{
```

```
  envia_byte_lcd(1,c);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função para limpar o LCD                         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Como esta operação pode ser muito utilizada, transformando-a em função
// faz cvom que o código compilado seja menor.
void limpa_lcd()
{
      envia_byte_lcd(0,0x01); // Envia instrução para limpar o LCD
      delay_ms(2);            // Aguarda 2ms para estabilizar o LCD
      return;                 // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Inicializa o LCD                                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void inicializa_lcd()
{
  output_low(lcd_db4);    // Garante que o pino DB4 estão em 0 (low)
  output_low(lcd_db5);    // Garante que o pino DB5 estão em 0 (low)
  output_low(lcd_db6);    // Garante que o pino DB6 estão em 0 (low)
  output_low(lcd_db7);    // Garante que o pino DB7 estão em 0 (low)
  output_low(lcd_rs);     // Garante que o pino RS estão em 0 (low)
  output_low(lcd_enable); // Garante que o pino ENABLE estão em 0 (low)
  delay_ms(15);           // Aguarda 15ms para estabilizar o LCD

  envia_nibble_lcd(0x03); // Envia comando para inicializar o display
  delay_ms(5);            // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x03); // Envia comando para inicializar o display
  delay_ms(5);            // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x03); // Envia comando para inicializar o display
  delay_ms(5);            // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x02); // CURSOR HOME - Envia comando para zerar o contador de
                          // caracteres e retornar à posição inicial (0x80).
  delay_ms(1);            // Aguarda 1ms para estabilizar o LCD
  envia_byte_lcd(0,0x28); // FUNCTION SET - Configura o LCD para 4 bits,
                          // 2 linhas, fonte 5X7.
  envia_byte_lcd(0,0x0c); // DISPLAY CONTROL - Display ligado, sem cursor
  limpa_lcd();            // Limpa o LCD
  envia_byte_lcd(0,0x06); // ENTRY MODE SET - Desloca o cursor para a direita

  return;                 // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      Cria caracteres na CG RAM                            *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void cria_cgram()
{
  envia_byte_lcd(0,0b01000000); // Vai para primeira posição da CGRAM

  envia_byte_lcd(1,0b00001110); // Cria a primeira linha de nosso robozinho
  envia_byte_lcd(1,0b00010001); // Cria a segunda linha de nosso robozinho
  envia_byte_lcd(1,0b00001110); // Cria a terceira linha de nosso robozinho
  envia_byte_lcd(1,0b00000100); // Cria a quarta linha de nosso robozinho
  envia_byte_lcd(1,0b00011111); // Cria a quinta linha de nosso robozinho
  envia_byte_lcd(1,0b00000100); // Cria a sexta linha de nosso robozinho
  envia_byte_lcd(1,0b00001010); // Cria a sétima linha de nosso robozinho
  envia_byte_lcd(1,0b00010001); // Cria a oitava linha de nosso robozinho

  envia_byte_lcd(1,0b00001110); // Cria a primeira linha de nosso robozinho 2
  envia_byte_lcd(1,0b00010001); // Cria a segunda linha de nosso robozinho 2
  envia_byte_lcd(1,0b00001110); // Cria a terceira linha de nosso robozinho 2
  envia_byte_lcd(1,0b00010100); // Cria a quarta linha de nosso robozinho 2
```

```
  envia_byte_lcd(1,0b00001110); // Cria a quinta linha de nosso robozinho 2
  envia_byte_lcd(1,0b00000101); // Cria a sexta linha de nosso robozinho 2
  envia_byte_lcd(1,0b00001010); // Cria a sétima linha de nosso robozinho 2
  envia_byte_lcd(1,0b00010001); // Cria a oitava linha de nosso robozinho 2

  envia_byte_lcd(1,0b00001110); // Cria a primeira linha de nosso robozinho 3
  envia_byte_lcd(1,0b00010001); // Cria a segunda linha de nosso robozinho 3
  envia_byte_lcd(1,0b00001110); // Cria a terceira linha de nosso robozinho 3
  envia_byte_lcd(1,0b00000101); // Cria a quarta linha de nosso robozinho 3
  envia_byte_lcd(1,0b00001110); // Cria a quinta linha de nosso robozinho 3
  envia_byte_lcd(1,0b00010100); // Cria a sexta linha de nosso robozinho 3
  envia_byte_lcd(1,0b00001010); // Cria a sétima linha de nosso robozinho 3
  envia_byte_lcd(1,0b00010001); // Cria a oitava linha de nosso robozinho 3

  envia_byte_lcd(1,0b00001110); // Cria a primeira linha de nosso robozinho 4
  envia_byte_lcd(1,0b00010001); // Cria a segunda linha de nosso robozinho 4
  envia_byte_lcd(1,0b00001110); // Cria a terceira linha de nosso robozinho 4
  envia_byte_lcd(1,0b00010101); // Cria a quarta linha de nosso robozinho 4
  envia_byte_lcd(1,0b00001110); // Cria a quinta linha de nosso robozinho 4
  envia_byte_lcd(1,0b00000100); // Cria a sexta linha de nosso robozinho 4
  envia_byte_lcd(1,0b00001010); // Cria a sétima linha de nosso robozinho 4
  envia_byte_lcd(1,0b00001010); // Cria a oitava linha de nosso robozinho 4

  envia_byte_lcd(1,0b00011111); // Cria a primeira linha de nosso janelinha
  envia_byte_lcd(1,0b00010101); // Cria a segunda linha de nosso janelinha
  envia_byte_lcd(1,0b00011111); // Cria a terceira linha de nosso janelinha
  envia_byte_lcd(1,0b00010101); // Cria a quarta linha de nosso janelinha
  envia_byte_lcd(1,0b00010101); // Cria a quinta linha de nosso janelinha
  envia_byte_lcd(1,0b00011111); // Cria a sexta linha de nosso janelinha
  envia_byte_lcd(1,0b00010101); // Cria a sétima linha de nosso janelinha
  envia_byte_lcd(1,0b00011111); // Cria a oitava linha de nosso janelinha

  envia_byte_lcd(1,0b00011111); // Cria a primeira linha de nossa ampulheta 1
  envia_byte_lcd(1,0b00011111); // Cria a segunda linha de nossa ampulheta 1
  envia_byte_lcd(1,0b00001110); // Cria a terceira linha de nossa ampulheta 1
  envia_byte_lcd(1,0b00000100); // Cria a quarta linha de nossa ampulheta 1
  envia_byte_lcd(1,0b00001010); // Cria a quinta linha de nossa ampulheta 1
  envia_byte_lcd(1,0b00010001); // Cria a sexta linha de nossa ampulheta 1
  envia_byte_lcd(1,0b00010001); // Cria a sétima linha de nossa ampulheta 1
  envia_byte_lcd(1,0b00011111); // Cria a oitava linha de nossa ampulheta 1

  envia_byte_lcd(1,0b00011111); // Cria a primeira linha de nossa ampulheta 2
  envia_byte_lcd(1,0b00010101); // Cria a segunda linha de nossa ampulheta 2
  envia_byte_lcd(1,0b00001110); // Cria a terceira linha de nossa ampulheta 2
  envia_byte_lcd(1,0b00000100); // Cria a quarta linha de nossa ampulheta 2
  envia_byte_lcd(1,0b00001010); // Cria a quinta linha de nossa ampulheta 2
  envia_byte_lcd(1,0b00010101); // Cria a sexta linha de nossa ampulheta 2
  envia_byte_lcd(1,0b00011111); // Cria a sétima linha de nossa ampulheta 2
  envia_byte_lcd(1,0b00011111); // Cria a oitava linha de nossa ampulheta 2

  envia_byte_lcd(1,0b00011111); // Cria a primeira linha de nossa ampulheta 3
  envia_byte_lcd(1,0b00010001); // Cria a segunda linha de nossa ampulheta 3
  envia_byte_lcd(1,0b00001010); // Cria a terceira linha de nossa ampulheta 3
  envia_byte_lcd(1,0b00000100); // Cria a quarta linha de nossa ampulheta 3
  envia_byte_lcd(1,0b00001110); // Cria a quinta linha de nossa ampulheta 3
  envia_byte_lcd(1,0b00011111); // Cria a sexta linha de nossa ampulheta 3
  envia_byte_lcd(1,0b00011111); // Cria a sétima linha de nossa ampulheta 3
  envia_byte_lcd(1,0b00011111); // Cria a oitava linha de nossa ampulheta 3

  limpa_lcd();


  return;                       // Retorna ao ponto de chamada da função
}

/**************************************************************************/
```

```
/*                      Final das rotinas para o LCD                        */
/**************************************************************************/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Configurações do Pic                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
main()
{
  // Configura o PIC
  setup_adc_ports(no_analogs);

  // Reseta portas
  porta = 0;
  portb = 0;
  portc = 0;
  portd = 0;
  porte = 0;

  // configura os tris
  set_tris_a(0b00111111);  // configuração da direção dos pinos de I/O
  set_tris_b(0b11111111);
  set_tris_c(0b11111111);
  set_tris_d(0b00001111);
  set_tris_e(0b11111100);

  // Inicializa o LCD
  inicializa_lcd();

  //Escreve tela
  limpa_lcd();
  cria_cgram();

  //Escreve os caracteres da CGRAM nas duas linhas do LCD
  escreve_lcd(0b00000000);
  escreve_lcd(0b00000001);
  escreve_lcd(0b00000010);
  escreve_lcd(0b00000011);
  escreve_lcd(0b00000100);
  escreve_lcd(0b00000101);
  escreve_lcd(0b00000110);
  escreve_lcd(0b00000111);

  envia_byte_lcd(0,0xc0);
  escreve_lcd(0b00001000);
  escreve_lcd(0b00001001);
  escreve_lcd(0b00001010);
  escreve_lcd(0b00001011);
  escreve_lcd(0b00001100);
  escreve_lcd(0b00001101);
  escreve_lcd(0b00001110);
  escreve_lcd(0b00001111);


 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Rotina principal                            *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
 //
  while (true)
  {

  }

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Fim do Programa                              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
}
```

Placa de testes

Este é o código que deverá ser implementado em nossa placa de testes:

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Tutorial LCD em 4 vias                         *
 *                     Para uso na placa de testes                       *
 *                                                                       *
 *               Exercício 4 - Criando caracteres na CGRAM               *
 *                                                                       *
 *                    Criado por Eduardo de Souza Ramos                  *
 *                                                                       *
 *               Memory usage:   ROM=26%      RAM=3% - 6%               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                        *
 *   DATA : 31/08/2005                                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Descrição geral                             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//     Neste exercício, criaremos 8 caracteres especiais (desenhos) na CGRAM
//  para utilizarmos em qualquer ponto do LCD
//     A priori eles serão exibidos nas 8 primeiras posições da primeira
//  linha do LCD. Mas eles poderão ser utilizados em qualquer ponto.
//     Como estes caracteres ficam na CGRAM, são voláteis, ou seja, ao
//  desligarmos o módulo e o ligarmos novamente, precisarmos reintroduzir
//  o código para a criação de caracteres.
//     Reparem que, apesar de termos 16 posições disponíveis na CGRAM,
//  somente poderemos criar 8 caracteres. As 8 posições adjacentes (posições
//  0x08 a 0x0f) são espelho da primeira (0x00 a 0x07)

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      Definição do PIC utilizado                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#include <16f628a.h>

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                         Configurações para gravação                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#fuses INTRC,NOWDT,PUT,MCLR,NOBROWNOUT,NOLVP
#ROM 0x07ff = {0} //Calibragem do oscilador interno

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                 Definição para uso de Rotinas de Delay                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#use delay(clock=4000000)

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Constantes internas                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Estas são as definições dos pinos que o LCD utiliza.
// Definem quais pinos do PIC controlarão os pinos do LCD
#define lcd_enable      pin_b5      // pino enable do LCD
#define lcd_rs    pin_b4      // pino rs do LCD
#define lcd_db4   pin_b0      // pino de dados d4 do LCD
#define lcd_db5   pin_b1      // pino de dados d5 do LCD
#define lcd_db6   pin_b2      // pino de dados d6 do LCD
#define lcd_db7   pin_b3      // pino de dados d7 do LCD

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Definição e inicialização dos port's                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#use  fast_io(a)   // Inicialização rápida dos Pinos de I/O
```

```
#use  fast_io(b)

#byte porta = 0x05
#byte portb = 0x06

/*************************************************************************/
/*                            Rotinas para o LCD                         */
/*************************************************************************/
//Este é o bloco com as rotinas necessárias para manipular o LCD

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      Envio de "Nibble" para o LCD                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Esta rotina lê o "Nibble" inferior de uma variável e envia para o LCD.
void envia_nibble_lcd(int dado)
{
  //Carrega as vias de dados (pinos) do LCD de acordo com o nibble lido
  output_bit(lcd_db4, bit_test(dado,0)); //Carrega DB4 do LCD com o bit DADO<0>
  output_bit(lcd_db5, bit_test(dado,1)); //Carrega DB5 do LCD com o bit DADO<1>
  output_bit(lcd_db6, bit_test(dado,2)); //Carrega DB6 do LCD com o bit DADO<2>
  output_bit(lcd_db7, bit_test(dado,3)); //Carrega DB7 do LCD com o bit DADO<3>

  //Gera um pulso de enable
  output_high(lcd_enable);    // ENABLE = 1
  delay_us(1);                // Recomendado para estabilizar o LCD
  output_low(lcd_enable);     // ENABLE = 0

  return;                     // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Envio de Byte para o LCD                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Esta rotina irá enviar um dado ou um comando para o LCD conforme abaixo:
//     ENDEREÇO = 0 -> a variável DADO será uma instrução
//     ENDEREÇO = 1 -> a variável DADO será um caractere
void envia_byte_lcd(boolean endereco, int dado)
{
  output_bit(lcd_rs,endereco);  // Seta o bit RS para instrução ou caractere
  delay_us(100);                // Aguarda 100 us para estabilizar o pino do LCD
  output_low(lcd_enable);       // Desativa a linha ENABLE
  envia_nibble_lcd(dado>>4);    // Envia a parte ALTA do dado/comando
  envia_nibble_lcd(dado & 0x0f);// Limpa a parte ALTA e envia a parte BAIXA do
                                // dado/comando
  delay_us(40);                 // Aguarda 40us para estabilizar o LCD
  return;                       // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                     Envio de caractere para o LCD                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Esta rotina serve apenas como uma forma mais fácil de escrever um caractere
// no display. Ela pode ser eliminada e ao invés dela usaremos diretamente a
// função envia_byte_lcd(1,"<caractere a ser mostrado no LCD>"); ou
// envia_byte_lcd(1,<código do caractere a ser mostrado no LCD>);

void escreve_lcd(char c)
// envia caractere para o display
{
  envia_byte_lcd(1,c);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Função para limpar o LCD                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Como esta operação pode ser muito utilizada, transformando-a em função
```

```
// faz com que o código compilado seja menor.
void limpa_lcd()
{
      envia_byte_lcd(0,0x01); // Envia instrução para limpar o LCD
      delay_ms(2);            // Aguarda 2ms para estabilizar o LCD
      return;                 // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Inicializa o LCD                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void inicializa_lcd()
{
  output_low(lcd_db4);  // Garante que o pino DB4 estão em 0 (low)
  output_low(lcd_db5);        // Garante que o pino DB5 estão em 0 (low)
  output_low(lcd_db6);        // Garante que o pino DB6 estão em 0 (low)
  output_low(lcd_db7);        // Garante que o pino DB7 estão em 0 (low)
  output_low(lcd_rs);         // Garante que o pino RS estão em 0 (low)
  output_low(lcd_enable);     // Garante que o pino ENABLE estão em 0 (low)
  delay_ms(15);         // Aguarda 15ms para estabilizar o LCD

  envia_nibble_lcd(0x03);  // Envia comando para inicializar o display
  delay_ms(5);             // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x03);  // Envia comando para inicializar o display
  delay_ms(5);             // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x03);  // Envia comando para inicializar o display
  delay_ms(5);             // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x02);  // CURSOR HOME - Envia comando para zerar o contador de
                           // caracteres e retornar à posição inicial (0x80).
  delay_ms(1);             // Aguarda 1ms para estabilizar o LCD
  envia_byte_lcd(0,0x28);  // FUNCTION SET - Configura o LCD para 4 bits,
                           // 2 linhas, fonte 5X7.
  envia_byte_lcd(0,0x0c);  // DISPLAY CONTROL - Display ligado, sem cursor
  limpa_lcd();             // Limpa o LCD
  envia_byte_lcd(0,0x06);  // ENTRY MODE SET - Desloca o cursor para a direita

  return;                  // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      Cria caracteres na CG RAM                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void cria_cgram()
{
  envia_byte_lcd(0,0b01000000); // Vai para primeira posição da CGRAM

  envia_byte_lcd(1,0b00001110); // Cria a primeira linha de nosso robozinho
  envia_byte_lcd(1,0b00010001); // Cria a segunda linha de nosso robozinho
  envia_byte_lcd(1,0b00001110); // Cria a terceira linha de nosso robozinho
  envia_byte_lcd(1,0b00000100); // Cria a quarta linha de nosso robozinho
  envia_byte_lcd(1,0b00011111); // Cria a quinta linha de nosso robozinho
  envia_byte_lcd(1,0b00000100); // Cria a sexta linha de nosso robozinho
  envia_byte_lcd(1,0b00001010); // Cria a sétima linha de nosso robozinho
  envia_byte_lcd(1,0b00010001); // Cria a oitava linha de nosso robozinho

  envia_byte_lcd(1,0b00001110); // Cria a primeira linha de nosso robozinho 2
  envia_byte_lcd(1,0b00010001); // Cria a segunda linha de nosso robozinho 2
  envia_byte_lcd(1,0b00001110); // Cria a terceira linha de nosso robozinho 2
  envia_byte_lcd(1,0b00010100); // Cria a quarta linha de nosso robozinho 2
  envia_byte_lcd(1,0b00001110); // Cria a quinta linha de nosso robozinho 2
  envia_byte_lcd(1,0b00000101); // Cria a sexta linha de nosso robozinho 2
  envia_byte_lcd(1,0b00001010); // Cria a sétima linha de nosso robozinho 2
  envia_byte_lcd(1,0b00010001); // Cria a oitava linha de nosso robozinho 2

  envia_byte_lcd(1,0b00001110); // Cria a primeira linha de nosso robozinho 3
  envia_byte_lcd(1,0b00010001); // Cria a segunda linha de nosso robozinho 3
```

```
  envia_byte_lcd(1,0b00001110); // Cria a terceira linha de nosso robozinho 3
  envia_byte_lcd(1,0b00000101); // Cria a quarta linha de nosso robozinho 3
  envia_byte_lcd(1,0b00001110); // Cria a quinta linha de nosso robozinho 3
  envia_byte_lcd(1,0b00010100); // Cria a sexta linha de nosso robozinho 3
  envia_byte_lcd(1,0b00001010); // Cria a sétima linha de nosso robozinho 3
  envia_byte_lcd(1,0b00010001); // Cria a oitava linha de nosso robozinho 3

  envia_byte_lcd(1,0b00001110); // Cria a primeira linha de nosso robozinho 4
  envia_byte_lcd(1,0b00010001); // Cria a segunda linha de nosso robozinho 4
  envia_byte_lcd(1,0b00001110); // Cria a terceira linha de nosso robozinho 4
  envia_byte_lcd(1,0b00010101); // Cria a quarta linha de nosso robozinho 4
  envia_byte_lcd(1,0b00001110); // Cria a quinta linha de nosso robozinho 4
  envia_byte_lcd(1,0b00000100); // Cria a sexta linha de nosso robozinho 4
  envia_byte_lcd(1,0b00001010); // Cria a sétima linha de nosso robozinho 4
  envia_byte_lcd(1,0b00001010); // Cria a oitava linha de nosso robozinho 4

  envia_byte_lcd(1,0b00011111); // Cria a primeira linha de nosso janelinha
  envia_byte_lcd(1,0b00010101); // Cria a segunda linha de nosso janelinha
  envia_byte_lcd(1,0b00011111); // Cria a terceira linha de nosso janelinha
  envia_byte_lcd(1,0b00010101); // Cria a quarta linha de nosso janelinha
  envia_byte_lcd(1,0b00010101); // Cria a quinta linha de nosso janelinha
  envia_byte_lcd(1,0b00011111); // Cria a sexta linha de nosso janelinha
  envia_byte_lcd(1,0b00010101); // Cria a sétima linha de nosso janelinha
  envia_byte_lcd(1,0b00011111); // Cria a oitava linha de nosso janelinha

  envia_byte_lcd(1,0b00011111); // Cria a primeira linha de nossa ampulheta 1
  envia_byte_lcd(1,0b00011111); // Cria a segunda linha de nossa ampulheta 1
  envia_byte_lcd(1,0b00001110); // Cria a terceira linha de nossa ampulheta 1
  envia_byte_lcd(1,0b00000100); // Cria a quarta linha de nossa ampulheta 1
  envia_byte_lcd(1,0b00001010); // Cria a quinta linha de nossa ampulheta 1
  envia_byte_lcd(1,0b00010001); // Cria a sexta linha de nossa ampulheta 1
  envia_byte_lcd(1,0b00010001); // Cria a sétima linha de nossa ampulheta 1
  envia_byte_lcd(1,0b00011111); // Cria a oitava linha de nossa ampulheta 1

  envia_byte_lcd(1,0b00011111); // Cria a primeira linha de nossa ampulheta 2
  envia_byte_lcd(1,0b00010101); // Cria a segunda linha de nossa ampulheta 2
  envia_byte_lcd(1,0b00001110); // Cria a terceira linha de nossa ampulheta 2
  envia_byte_lcd(1,0b00000100); // Cria a quarta linha de nossa ampulheta 2
  envia_byte_lcd(1,0b00001010); // Cria a quinta linha de nossa ampulheta 2
  envia_byte_lcd(1,0b00010101); // Cria a sexta linha de nossa ampulheta 2
  envia_byte_lcd(1,0b00011111); // Cria a sétima linha de nossa ampulheta 2
  envia_byte_lcd(1,0b00011111); // Cria a oitava linha de nossa ampulheta 2

  envia_byte_lcd(1,0b00011111); // Cria a primeira linha de nossa ampulheta 3
  envia_byte_lcd(1,0b00010001); // Cria a segunda linha de nossa ampulheta 3
  envia_byte_lcd(1,0b00001010); // Cria a terceira linha de nossa ampulheta 3
  envia_byte_lcd(1,0b00000100); // Cria a quarta linha de nossa ampulheta 3
  envia_byte_lcd(1,0b00001110); // Cria a quinta linha de nossa ampulheta 3
  envia_byte_lcd(1,0b00011111); // Cria a sexta linha de nossa ampulheta 3
  envia_byte_lcd(1,0b00011111); // Cria a sétima linha de nossa ampulheta 3
  envia_byte_lcd(1,0b00011111); // Cria a oitava linha de nossa ampulheta 3

  limpa_lcd();

  return;               // Retorna ao ponto de chamada da função
}

/*************************************************************************/
/*                      Final das rotinas para o LCD                     */
/*************************************************************************/


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                         Configurações do PIC                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
main()
```

```
{
  // Reseta portas
  porta = 0;
  portb = 0;

  // configura os tris
  set_tris_a(0b00111111);  // configuração da direção dos pinos de I/O
  set_tris_b(0b11000000);

  // Inicializa o LCD
  inicializa_lcd();

  //Escreve tela
  limpa_lcd();
  cria_cgram();

  //envia_byte_lcd(0,0x80);
  escreve_lcd(0b00000000);
  escreve_lcd(0b00000001);
  escreve_lcd(0b00000010);
  escreve_lcd(0b00000011);
  escreve_lcd(0b00000100);
  escreve_lcd(0b00000101);
  escreve_lcd(0b00000110);
  escreve_lcd(0b00000111);

  envia_byte_lcd(0,0xc0);
  escreve_lcd(0b00001000);
  escreve_lcd(0b00001001);
  escreve_lcd(0b00001010);
  escreve_lcd(0b00001011);
  escreve_lcd(0b00001100);
  escreve_lcd(0b00001101);
  escreve_lcd(0b00001110);
  escreve_lcd(0b00001111);


 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Rotina principal                              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
 //
  while (true)
  {

  }

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Fim do Programa                                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
}
```

Exercício 5 – Uma pequena animação

Exercício 5 – Uma pequena animação

E agora, como exercício final, vamos fazer uma pequena animação com os caracteres criados no exercício anterior e vamos introduzir o conceito de “backspace” automaticamente pelo LCD.

O organograma é o mesmo para a McLAB2 e para a nossa placa de testes:

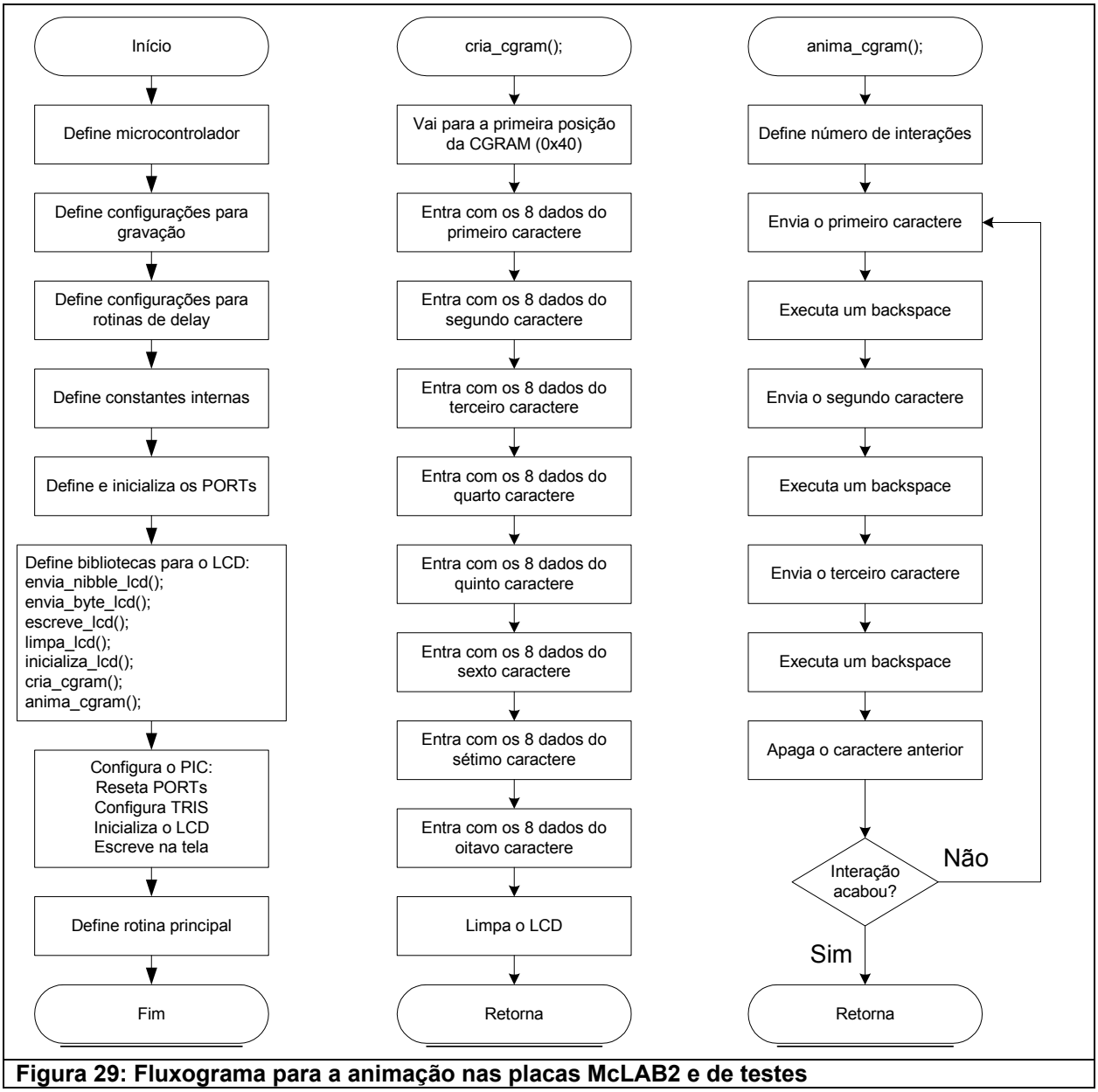


Figura 29: Fluxograma para a animação nas placas McLAB2 e de testes

O código a seguir deve ser programado na McLAB2:

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Tutorial LCD em 4 vias                           *
 *                     Para uso na placa de testes                         *
 *                                                                         *
 *                   Exercício 5 - Uma pequena animação                    *
 *                                                                         *
 *                 Memory usage:   ROM=5%      RAM=3% - 7%                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                          *
 *   DATA : 31/08/2005                                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Descrição geral                              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//  Pequena animação para exemplificar o uso dos caracteres gerados na CGRAM
//  e testar a função de backspace.
//  O tempo de animação é controlado pela constante tempo

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Definição do PIC utilizado                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#include <16f877a.h>

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                         Configurações para gravação                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#fuses XT,NOWDT,NOPROTECT,PUT,BROWNOUT,NOLVP,NOCPD,NOWRT

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                 Definição para uso de Rotinas de Delay                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#use delay(clock=4000000)

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Constantes internas                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Estas são as definições dos pinos que o LCD utiliza.
// Definem quais pinos do PIC controlarão os pinos do LCD
#define lcd_enable      pin_e1            // pino enable do LCD
#define lcd_rs    pin_e0            // pino rs do LCD
#define lcd_db4   pin_d4            // pino de dados d4 do LCD
#define lcd_db5   pin_d5            // pino de dados d5 do LCD
#define lcd_db6   pin_d6            // pino de dados d6 do LCD
#define lcd_db7   pin_d7            // pino de dados d7 do LCD
#define tempo 150

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                   Definição e inicialização dos port's                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#use  fast_io(a)   // Inicialização rápida dos Pinos de I/O
#use  fast_io(b)
#use  fast_io(c)
#use  fast_io(d)
#use  fast_io(e)

#byte porta = 0x05
#byte portb = 0x06
#byte portc = 0x07
#byte portd = 0x08
```

```
#byte porte = 0x09

/************************************************************************/
/*                            Rotinas para o LCD                        */
/************************************************************************/
//Este é o bloco com as rotinas necessárias para manipular o LCD

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Envio de "Nibble" para o LCD                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Esta rotina lê o "Nibble" inferior de uma variável e envia para o LCD.
void envia_nibble_lcd(int dado)
{
  //Carrega as vias de dados (pinos) do LCD de acordo com o nibble lido
  output_bit(lcd_db4, bit_test(dado,0)); //Carrega DB4 do LCD com o bit DADO<0>
  output_bit(lcd_db5, bit_test(dado,1)); //Carrega DB5 do LCD com o bit DADO<1>
  output_bit(lcd_db6, bit_test(dado,2)); //Carrega DB6 do LCD com o bit DADO<2>
  output_bit(lcd_db7, bit_test(dado,3)); //Carrega DB7 do LCD com o bit DADO<3>
  //Gera um pulso de enable
  output_high(lcd_enable);    // ENABLE = 1
  delay_us(1);                // Recomendado para estabilizar o LCD
  output_low(lcd_enable);     // ENABLE = 0
  return;                     // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                         Envio de Byte para o LCD                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Esta rotina irá enviar um dado ou um comando para o LCD conforme abaixo:
//     ENDEREÇO = 0 -> a variável DADO será uma instrução
//     ENDEREÇO = 1 -> a variável DADO será um caractere
void envia_byte_lcd(boolean endereco, int dado)
{
  output_bit(lcd_rs,endereco);  // Seta o bit RS para instrução ou caractere
  delay_us(100);                // Aguarda 100 us para estabilizar o pino do LCD
  output_low(lcd_enable);       // Desativa a linha ENABLE
  envia_nibble_lcd(dado>>4);    // Envia a parte ALTA do dado/coamndo
  envia_nibble_lcd(dado & 0x0f);// Limpa a parte ALTA e envia a parte BAIXA do
                                // dado/comando
  delay_us(40);                 // Aguarda 40us para estabilizar o LCD
  return;                       // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                      Envio de caractere para o LCD                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Esta rotina serve apenas como uma forma mais fácil de escrever um caractere
// no display. Ela pode ser eliminada e ao invés dela usaremos diretamente a
// função envia_byte_lcd(1,"<caractere a ser mostrado no lcd>"); ou
// envia_byte_lcd(1,<código do caractere a ser mostrado no lcd>);

void escreve_lcd(char c)
// envia caractere para o display
{
  envia_byte_lcd(1,c);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Função para limpar o LCD                      *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Como esta operação pode ser muito utilizada, transformando-a em função
// faz cvom que o código compilado seja menor.
void limpa_lcd()
{
      envia_byte_lcd(0,0x01); // Envia instrução para limpar o LCD
      delay_ms(2);            // Aguarda 2ms para estabilizar o LCD
```

```
      return;                 // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Inicializa o LCD                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void inicializa_lcd()
{
  output_low(lcd_db4);    // Garante que o pino DB4 estão em 0 (low)
  output_low(lcd_db5);    // Garante que o pino DB5 estão em 0 (low)
  output_low(lcd_db6);    // Garante que o pino DB6 estão em 0 (low)
  output_low(lcd_db7);    // Garante que o pino DB7 estão em 0 (low)
  output_low(lcd_rs);     // Garante que o pino RS estão em 0 (low)
  output_low(lcd_enable); // Garante que o pino ENABLE estão em 0 (low)
  delay_ms(15);           // Aguarda 15ms para estabilizar o LCD
  envia_nibble_lcd(0x03); // Envia comando para inicializar o display
  delay_ms(5);            // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x03); // Envia comando para inicializar o display
  delay_ms(5);            // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x03); // Envia comando para inicializar o display
  delay_ms(5);            // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x02); // CURSOR HOME - Envia comando para zerar o contador de
                          // caracteres e retornar à posição inicial (0x80).
  delay_ms(1);            // Aguarda 1ms para estabilizar o LCD
  envia_byte_lcd(0,0x28); // FUNCTION SET - Configura o LCD para 4 bits,
                          // 2 linhas, fonte 5X7.
  envia_byte_lcd(0,0x0c); // DISPLAY CONTROL - Display ligado, sem cursor
  limpa_lcd();            // Limpa o LCD
  envia_byte_lcd(0,0x06); // ENTRY MODE SET - Desloca o cursor para a direita
  return;                 // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                     Cria caracteres na CG RAM                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void cria_cgram()
{
  //escreve_lcd(0b00000000);
  //escreve_lcd(0b00000001);
  //escreve_lcd(0b00000010);
  //escreve_lcd(0b00000011);
  //escreve_lcd(0b00000100);
  //escreve_lcd(0b00000101);
  //escreve_lcd(0b00000110);
  //escreve_lcd(0b00000111);

  envia_byte_lcd(0,0b01000000); // Vai para primeira posição da CGRAM
  envia_byte_lcd(1,0b00001110); // Cria a primeira linha de nosso robozinho 1
  envia_byte_lcd(1,0b00010001); // Cria a segunda linha de nosso robozinho 1
  envia_byte_lcd(1,0b00001110); // Cria a terceira linha de nosso robozinho 1
  envia_byte_lcd(1,0b00000100); // Cria a quarta linha de nosso robozinho 1
  envia_byte_lcd(1,0b00011111); // Cria a quinta linha de nosso robozinho 1
  envia_byte_lcd(1,0b00000100); // Cria a sexta linha de nosso robozinho 1
  envia_byte_lcd(1,0b00001010); // Cria a sétima linha de nosso robozinho 1
  envia_byte_lcd(1,0b00010001); // Cria a oitava linha de nosso robozinho 1

  envia_byte_lcd(1,0b00001110); // Cria a primeira linha de nosso robozinho 2
  envia_byte_lcd(1,0b00010001); // Cria a segunda linha de nosso robozinho 2
  envia_byte_lcd(1,0b00001110); // Cria a terceira linha de nosso robozinho 2
  envia_byte_lcd(1,0b00010100); // Cria a quarta linha de nosso robozinho 2
  envia_byte_lcd(1,0b00001110); // Cria a quinta linha de nosso robozinho 2
  envia_byte_lcd(1,0b00000101); // Cria a sexta linha de nosso robozinho 2
  envia_byte_lcd(1,0b00001010); // Cria a sétima linha de nosso robozinho 2
  envia_byte_lcd(1,0b00010001); // Cria a oitava linha de nosso robozinho 2

  envia_byte_lcd(1,0b00001110); // Cria a primeira linha de nosso robozinho 3
```

```
  envia_byte_lcd(1,0b00010001); // Cria a segunda linha de nosso robozinho 3
  envia_byte_lcd(1,0b00001110); // Cria a terceira linha de nosso robozinho 3
  envia_byte_lcd(1,0b00000101); // Cria a quarta linha de nosso robozinho 3
  envia_byte_lcd(1,0b00001110); // Cria a quinta linha de nosso robozinho 3
  envia_byte_lcd(1,0b00010100); // Cria a sexta linha de nosso robozinho 3
  envia_byte_lcd(1,0b00001010); // Cria a sétima linha de nosso robozinho 3
  envia_byte_lcd(1,0b00010001); // Cria a oitava linha de nosso robozinho 3

  envia_byte_lcd(1,0b00001110); // Cria a primeira linha de nosso robozinho 4
  envia_byte_lcd(1,0b00010001); // Cria a segunda linha de nosso robozinho 4
  envia_byte_lcd(1,0b00001110); // Cria a terceira linha de nosso robozinho 4
  envia_byte_lcd(1,0b00010101); // Cria a quarta linha de nosso robozinho 4
  envia_byte_lcd(1,0b00001110); // Cria a quinta linha de nosso robozinho 4
  envia_byte_lcd(1,0b00000100); // Cria a sexta linha de nosso robozinho 4
  envia_byte_lcd(1,0b00001010); // Cria a sétima linha de nosso robozinho 4
  envia_byte_lcd(1,0b00001010); // Cria a oitava linha de nosso robozinho 4

  limpa_lcd();
  return;                     // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                     Cria uma animação na CG RAM                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void anima_cgram()
{
      int i;
      for (i=0;i<=15;i++)
      {
            escreve_lcd(0b00000000);
            delay_ms(tempo);
            envia_byte_lcd(0,0b00010000); //backspace
            escreve_lcd(0b00000001);
            delay_ms(tempo);
            envia_byte_lcd(0,0b00010000); //backspace
            escreve_lcd(0b00000010);
            delay_ms(tempo);
            envia_byte_lcd(0,0b00010000); //backspace
            escreve_lcd(0b00000011);
            delay_ms(tempo);
            envia_byte_lcd(0,0b00010000); //backspace
escreve_lcd(" ");
}
return;
}

/*************************************************************************/
/*                     Final das rotinas para o LCD                      */
/*************************************************************************/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Configurações do Pic                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
main()
{
  // Configura o PIC
  setup_adc_ports(no_analogs);

  // Reseta portas
  porta = 0;
  portb = 0;
  portc = 0;
  portd = 0;
  porte = 0;

  // configura os tris
```

```
  set_tris_a(0b00111111);  // configuração da direção dos pinos de I/O
  set_tris_b(0b11111111);
  set_tris_c(0b11111111);
  set_tris_d(0b00001111);
  set_tris_e(0b11111100);

  // Inicializa o LCD
  inicializa_lcd();

  //Escreve tela
  limpa_lcd();
  cria_cgram();

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Rotina principal                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Esta rotina é a responsável pelo movimento de vai-e-vem do display.
  while (true)
  {
      //Faz uma animação do robozinho
      envia_byte_lcd(0,0x80);
      anima_cgram();
  }

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Fim do Programa                                 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
}
```

Placa de testes

O código a seguir deve ser programado na nossa placa de testes:

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Tutorial LCD em 4 vias                           *
 *                     Para uso na placa de testes                         *
 *                                                                         *
 *                   Exercício 5 - Uma pequena animação                    *
 *                                                                         *
 *               Memory usage:   ROM=19%      RAM=3% - 7%                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *   VERSÃO : 1.0                                                          *
 *   DATA : 31/08/2005                                                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                            Descrição geral                              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//  Pequena animação para exemplificar o uso dos caracteres gerados na CGRAM
// e testar a função de backspace.
//  O tempo de animação é controlado pela constante tempo

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Definição do PIC utilizado                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#include <16f628a.h>

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Configurações para gravação                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#fuses INTRC,NOWDT,PUT,MCLR,NOBROWNOUT,NOLVP
#ROM 0x07ff = {0} //Calibragem do oscilador interno

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                Definição para uso de Rotinas de Delay                     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#use delay(clock=4000000)

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                           Constantes internas                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Estas são as definições dos pinos que o LCD utiliza.
// Definem quais pinos do PIC controlarão os pinos do LCD
#define lcd_enable      pin_b5      // pino enable do LCD
#define lcd_rs    pin_b4      // pino rs do LCD
#define lcd_db4   pin_b0      // pino de dados d4 do LCD
#define lcd_db5   pin_b1      // pino de dados d5 do LCD
#define lcd_db6   pin_b2      // pino de dados d6 do LCD
#define lcd_db7   pin_b3      // pino de dados d7 do LCD

#define tempo 150

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Definição e inicialização dos port's                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#use  fast_io(a)   // Inicialização rápida dos Pinos de I/O
#use  fast_io(b)

#byte porta = 0x05
#byte portb = 0x06

/*************************************************************************/
/*                            Rotinas para o LCD                         */
```

```
/************************************************************************/
//Este é o bloco com as rotinas necessárias para manipular o LCD

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                        Envio de "Nibble" para o LCD                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Esta rotina lê o "Nibble" inferior de uma variável e envia para o LCD.
void envia_nibble_lcd(int dado)
{
  //Carrega as vias de dados (pinos) do LCD de acordo com o nibble lido
  output_bit(lcd_db4, bit_test(dado,0)); //Carrega DB4 do LCD com o bit DADO<0>
  output_bit(lcd_db5, bit_test(dado,1)); //Carrega DB5 do LCD com o bit DADO<1>
  output_bit(lcd_db6, bit_test(dado,2)); //Carrega DB6 do LCD com o bit DADO<2>
  output_bit(lcd_db7, bit_test(dado,3)); //Carrega DB7 do LCD com o bit DADO<3>

  //Gera um pulso de enable
  output_high(lcd_enable);    // ENABLE = 1
  delay_us(1);                // Recomendado para estabilizar o LCD
  output_low(lcd_enable);     // ENABLE = 0

  return;                     // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Envio de Byte para o LCD                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
//Esta rotina irá enviar um dado ou um comando para o LCD conforme abaixo:
//     ENDEREÇO = 0 -> a variável DADO será uma instrução
//     ENDEREÇO = 1 -> a variável DADO será um caractere
void envia_byte_lcd(boolean endereco, int dado)
{
  output_bit(lcd_rs,endereco);  // Seta o bit RS para instrução ou caractere
  delay_us(100);                // Aguarda 100 us para estabilizar o pino do LCD
  output_low(lcd_enable);       // Desativa a linha ENABLE
  envia_nibble_lcd(dado>>4);    // Envia a parte ALTA do dado/comando
  envia_nibble_lcd(dado & 0x0f);// Limpa a parte ALTA e envia a parte BAIXA do
                                // dado/comando
  delay_us(40);                 // Aguarda 40us para estabilizar o LCD
  return;                       // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                       Envio de caractere para o LCD                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Esta rotina serve apenas como uma forma mais fácil de escrever um caractere
// no display. Ela pode ser eliminada e ao invés dela usaremos diretamente a
// função envia_byte_lcd(1,"<caractere a ser mostrado no LCD>"); ou
// envia_byte_lcd(1,<código do caractere a ser mostrado no LCD>);

void escreve_lcd(char c)
// envia caractere para o display
{
  envia_byte_lcd(1,c);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Função para limpar o LCD                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Como esta operação pode ser muito utilizada, transformando-a em função
// faz com que o código compilado seja menor.
void limpa_lcd()
{
      envia_byte_lcd(0,0x01); // Envia instrução para limpar o LCD
      delay_ms(2);            // Aguarda 2ms para estabilizar o LCD
      return;                 // Retorna ao ponto de chamada da função
}
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                          Inicializa o LCD                                   *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void inicializa_lcd()
{
  output_low(lcd_db4);     // Garante que o pino DB4 estão em 0 (low)
  output_low(lcd_db5);     // Garante que o pino DB5 estão em 0 (low)
  output_low(lcd_db6);     // Garante que o pino DB6 estão em 0 (low)
  output_low(lcd_db7);     // Garante que o pino DB7 estão em 0 (low)
  output_low(lcd_rs);      // Garante que o pino RS estão em 0 (low)
  output_low(lcd_enable);  // Garante que o pino ENABLE estão em 0 (low)
  delay_ms(15);            // Aguarda 15ms para estabilizar o LCD

  envia_nibble_lcd(0x03);  // Envia comando para inicializar o display
  delay_ms(5);             // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x03);  // Envia comando para inicializar o display
  delay_ms(5);             // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x03);  // Envia comando para inicializar o display
  delay_ms(5);             // Aguarda 5ms para estabilizar o LCD
  envia_nibble_lcd(0x02);  // CURSOR HOME - Envia comando para zerar o contador de
                           // caracteres e retornar à posição inicial (0x80).
  delay_ms(1);             // Aguarda 1ms para estabilizar o LCD
  envia_byte_lcd(0,0x28);  // FUNCTION SET - Configura o LCD para 4 bits,
                           // 2 linhas, fonte 5X7.
  envia_byte_lcd(0,0x0c);  // DISPLAY CONTROL - Display ligado, sem cursor
  limpa_lcd();             // Limpa o LCD
  envia_byte_lcd(0,0x06);  // ENTRY MODE SET - Desloca o cursor para a direita

  return;                  // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                     Cria caracteres na CG RAM                               *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void cria_cgram()
{
  //escreve_lcd(0b00000000);
  //escreve_lcd(0b00000001);
  //escreve_lcd(0b00000010);
  //escreve_lcd(0b00000011);
  //escreve_lcd(0b00000100);
  //escreve_lcd(0b00000101);
  //escreve_lcd(0b00000110);
  //escreve_lcd(0b00000111);

  envia_byte_lcd(0,0b01000000); // Vai para primeira posição da CGRAM

  envia_byte_lcd(1,0b00001110); // Cria a primeira linha de nosso robozinho 1
  envia_byte_lcd(1,0b00010001); // Cria a segunda linha de nosso robozinho 1
  envia_byte_lcd(1,0b00001110); // Cria a terceira linha de nosso robozinho 1
  envia_byte_lcd(1,0b00000100); // Cria a quarta linha de nosso robozinho 1
  envia_byte_lcd(1,0b00011111); // Cria a quinta linha de nosso robozinho 1
  envia_byte_lcd(1,0b00000100); // Cria a sexta linha de nosso robozinho 1
  envia_byte_lcd(1,0b00001010); // Cria a sétima linha de nosso robozinho 1
  envia_byte_lcd(1,0b00010001); // Cria a oitava linha de nosso robozinho 1

  envia_byte_lcd(1,0b00001110); // Cria a primeira linha de nosso robozinho 2
  envia_byte_lcd(1,0b00010001); // Cria a segunda linha de nosso robozinho 2
  envia_byte_lcd(1,0b00001110); // Cria a terceira linha de nosso robozinho 2
  envia_byte_lcd(1,0b00010100); // Cria a quarta linha de nosso robozinho 2
  envia_byte_lcd(1,0b00001110); // Cria a quinta linha de nosso robozinho 2
  envia_byte_lcd(1,0b00000101); // Cria a sexta linha de nosso robozinho 2
  envia_byte_lcd(1,0b00001010); // Cria a sétima linha de nosso robozinho 2
  envia_byte_lcd(1,0b00010001); // Cria a oitava linha de nosso robozinho 2
```

```
  envia_byte_lcd(1,0b00001110); // Cria a primeira linha de nosso robozinho 3
  envia_byte_lcd(1,0b00010001); // Cria a segunda linha de nosso robozinho 3
  envia_byte_lcd(1,0b00001110); // Cria a terceira linha de nosso robozinho 3
  envia_byte_lcd(1,0b00000101); // Cria a quarta linha de nosso robozinho 3
  envia_byte_lcd(1,0b00001110); // Cria a quinta linha de nosso robozinho 3
  envia_byte_lcd(1,0b00010100); // Cria a sexta linha de nosso robozinho 3
  envia_byte_lcd(1,0b00001010); // Cria a sétima linha de nosso robozinho 3
  envia_byte_lcd(1,0b00010001); // Cria a oitava linha de nosso robozinho 3

  envia_byte_lcd(1,0b00001110); // Cria a primeira linha de nosso robozinho 4
  envia_byte_lcd(1,0b00010001); // Cria a segunda linha de nosso robozinho 4
  envia_byte_lcd(1,0b00001110); // Cria a terceira linha de nosso robozinho 4
  envia_byte_lcd(1,0b00010101); // Cria a quarta linha de nosso robozinho 4
  envia_byte_lcd(1,0b00001110); // Cria a quinta linha de nosso robozinho 4
  envia_byte_lcd(1,0b00000100); // Cria a sexta linha de nosso robozinho 4
  envia_byte_lcd(1,0b00001010); // Cria a sétima linha de nosso robozinho 4
  envia_byte_lcd(1,0b00001010); // Cria a oitava linha de nosso robozinho 4

  limpa_lcd();

  return;               // Retorna ao ponto de chamada da função
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                     Cria uma animação na CG RAM                         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void anima_cgram()
{
      int i;
      for (i=0;i<=15;i++)
      {
            escreve_lcd(0b00000000);
            delay_ms(tempo);
            envia_byte_lcd(0,0b00010000); //backspace
            escreve_lcd(0b00000001);
            delay_ms(tempo);
            envia_byte_lcd(0,0b00010000); //backspace
            escreve_lcd(0b00000010);
            delay_ms(tempo);
            envia_byte_lcd(0,0b00010000); //backspace
            escreve_lcd(0b00000011);
            delay_ms(tempo);
            envia_byte_lcd(0,0b00010000); //backspace
escreve_lcd(" ");
}

return;
}

/*****************************************************************************/
/*                     Final das rotinas para o LCD                          */
/*****************************************************************************/


/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                         Configurações do PIC                            *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
main()
{
  // Reseta portas
  porta = 0;
  portb = 0;

  // configura os tris
  set_tris_a(0b00111111);  // configuração da direção dos pinos de I/O
  set_tris_b(0b11000000);
```

```
  // Inicializa o LCD
  inicializa_lcd();

  //Escreve tela
  limpa_lcd();
  cria_cgram();

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                              Rotina principal                           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
// Esta rotina é a responsável pelo movimento de vai-e-vem do display.
// Ela fica em loop infinito.;
  while (true)
  {
      //Faz uma animação do robozinho
      envia_byte_lcd(0,0x80);
anima_cgram();

  }

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                             Fim do Programa                             *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
}
```

Considerações finais

CONSIDERAÇÕES FINAIS

Acreditamos que este material cobriu todos os tópicos que abordam a configuração de um módulo LCD baseado nos processadores HD44780, KS0066U ou equivalentes.

Com este material, você será capaz de introduzir ou modificar um LCD em seus projetos.

Portanto, mãos à obra e boa sorte!

Eduardo Souza Ramos