Módulo 3 - Displays LCD

Agora que já entendemos bem como os pinos de I/O do microcontrolador funcionam, podemos utilizar uma forma mais profissional de se mostrar dados que vêm de dentro do microcontrolador. Por isso, utilizaremos o display LCD.

Os displays LCD recebem comandos em 8 bits ou 4 bits e os reproduzem na forma de caracteres. Esses comandos podem alterar algumas configurações também, como posição do cursor ou conteúdo da memória.

O display LCD

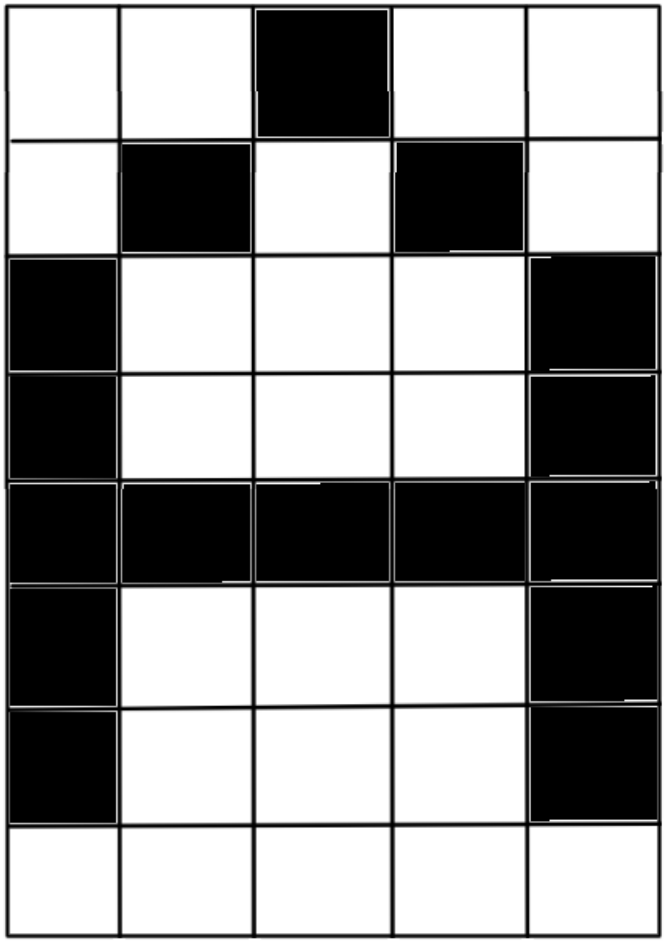
Existem vários tipos de display. O modelo que utilizaremos é um 16x2 com backlight azul, mas também é comum encontrar displays com blacklight amarelo ou com formato 16x4. Todos possuem modo de utilização idênticos

Pino	Função	Descrição
1	Alimentação	GND
2	Alimentação	+5V

Pino	Função	Descrição
3	Contraste	Liga com um resistor variável ao GND para controlar o contraste do display
4	Modo de comando	1 - Caractere, 0 - Comando. Liga no microcontrolador e define o estado do pino via software.
5	Leitura ou Escrita	1 - Leitura da memória, 0 - Escrita no display. Geralmente mantido em 0.
6	Clock	Executa o comando enviado ao display. Ligado ao microcontrolador.
7 → 14	Barramento de Dados	8 bits de dados
15	Anodo	
16	Catodo	

Dizer que o display é 16x2 ou 16x4 está relacionado a quantidade de caracteres que ele pode mostrar de uma só vez. Mas cada um desses caracteres são compostos de uma matriz de pixels 8x5 que criam estes caracteres.

Para mostrar um caractere, é necessário acender cada um dos pixels afim de formar o caracter em sim, como é o caso do caracter A:



A ultima linha seria reservada ao cursor, por isso nosso caractere fica com um tamanho de 7x5. Então, assim como no acendimento dos leds, teríamos que nos preocupar em criar código que gerenciasse eles pixels 1 a 1 para trabalhar com caracteres. Felizmente, isso não é necessário.

Displays LCD possuem um controlador interno com uma memória com todas as matrizes de caracteres definidas, além de comandos internos de endereçamento e manipulação de caracteres. Para acessá-los, basta utilizar o barramento de dados do display D<0:7>.

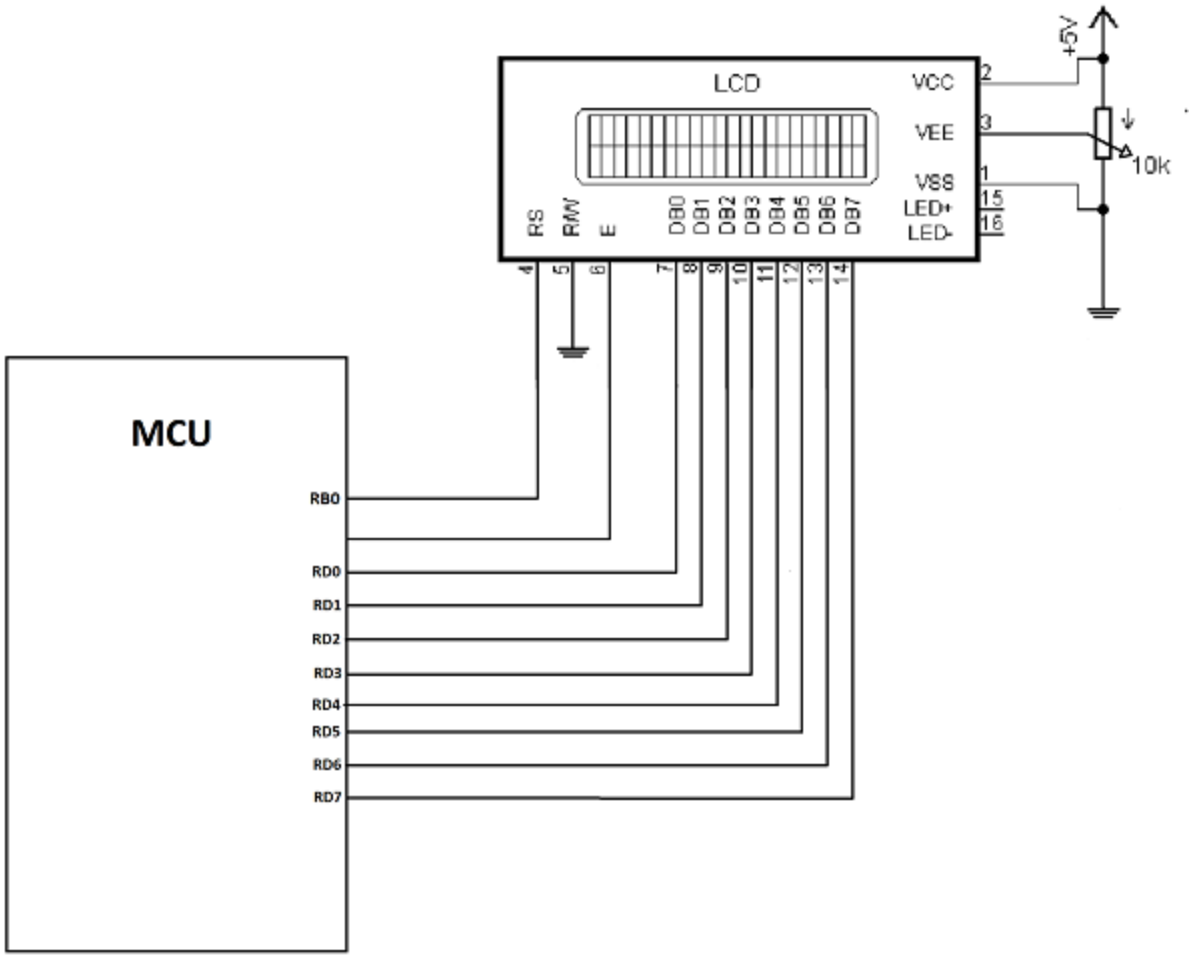

Upper 4 Bits
Lower 4 Bits
0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
xxxx0000 CG RAM (1)
xxxx0001 (2)
xxxx0010 (3)
xxxx0011 (4)
xxxx0100 (5)
xxxx0101 (6)
xxxx0110 (7)
xxxx0111 (8)
xxxx1000 (1)
xxxx1001 (2)
xxxx1010 (3)
xxxx1011 (4)
xxxx1100 (5)
xxxx1101 (6)
xxxx1110 (7)
xxxx1111 (8)

Os caracteres do display seguem o mesmo código da tabela ASCII, o que torna possível enviar os caracteres diretamente pelo PORT do microcontrolador. Isso significa que, no seu código, você poderá utilizar os caracteres na forma como você está acostumado a fazer, facilitando seu trabalho.

Para entender o funcionamento de uma forma didática, podemos utilizar um software simulador de LCD, disponível em: http://paginapessoal.utfpr.edu.br/rosangela/lcd-e-adc

Ligação do display no microcontrolador

A ligação abaixo é para o modo 8 bits:



Características do display

CGRAM → memória interna para desenhar no display LCD

ROM → É aqui que fica salvo as informações do display

O clock (E) é processado na queda do sinal

O pino R/W define as funcionalidades de Read e Write.

- O modo Write (0) é o usado para escrever dados no display
- O modo Read (1) do display serve para ler o status do display. O busy flag, por exemplo, é um sinal que o display possui para indicar se ele está ocupado com algum dado ou se está livre para receber um novo dado. Nesta funcionalidade, o display sinaliza sua situação no pino D7. Geralmente, não é necessário utilizar esta opção pois nós fazemos um código cheio de pausas em locais estratégicos para dar o tempo do display processar os dados.

Existem vários modelos de displays no mercado. Geralmente, estes displays já contém um controlador e uma memória interna que controlam a matriz de pixels presente no LCD. Ou seja, basta enviar o comando referente a algum caratere pelo barramento do display e ele cuidará do resto. O tipo

Mesmo com a tabela ASCII contendo todos estes símbolos, nem todos os sitemas suportam eles. Mas existe uma faixa de caracteres que é quase garantido que seja compatível com todos os sistemas, esta faixa começa do comando SPACE (32 em decimal) e vai até o comando DEL (127 em decimal).

Modo 8 bits e modo 4 bits

É muito mais comum encontrar configurações de 4 bits pois assim você economiza pinos do microcontrolador. Em relação à desempenho, não é perdido nada relevante para o usuário final.

O modo de operação é indicado ao display no processo de inicialização.

Inicialização do display

Comandos gerais do display

```
// Inicialização
// Geralmente enviado 5 vezes
```

```
enviaComando(0x30);
__delay_ms(5);
```

Modo de utilização do LCD

0b001ABC00

- A = 1: Estabelece comunicação em 8 bits
- A = 0: Estabelece comunicação em 4 bits
- B = 1: Estabelece 2 ou mais linhas de escrita
- B = 0: Estabelece 1 linha de escrita
- C = 1: Fixa o tipo de matriz de 10x5 (somente quando B = 0)
- C = 0: Fixa o tipo de matriz em 7x5 ou 8x5

Tempo de espera 40 us

Deslocamento do cursor ou da mensagem

0b0001AB00

A	B	**Funcao**
0	0	Desloca o cursor para a esquerda e decrementa o contador de endereco.
0	1	Desloca o cursor para a direita e incrementa o contador de endereco
1	0	Desloca a mensagem e o cursor para a esquerda
1	1	Desloca a mensagem e o cursor para a direita

Tempo de operação 40us

Controle do Display

0b00001ABC

- A = 1: Liga display sem cursor
- A = 0: Desliga display

- B = 1: Liga cursor
- B = 0: Desliga cursor
- C = 1: Cursor piscante se B = 1
- C = 0: Desliga cursor piscante

Tempo de operação 40us

Fixa o modo de operação

0b000001AB

Descolamento do cursor :

- A = 1 Para a direita
- A = 0 Para a esquerda

Estabelece se a mensagem deve ou não ser deslocada com a entrada de um novo caractere

- B = 1: Sim
- B = 0: Não

Tempo de operação 40us

Cursor Home

0b00000010

Limpa o display e retorna o cursor para linha 1 coluna 1. Também retorna a mensagem previamente deslocada para sua posição original.

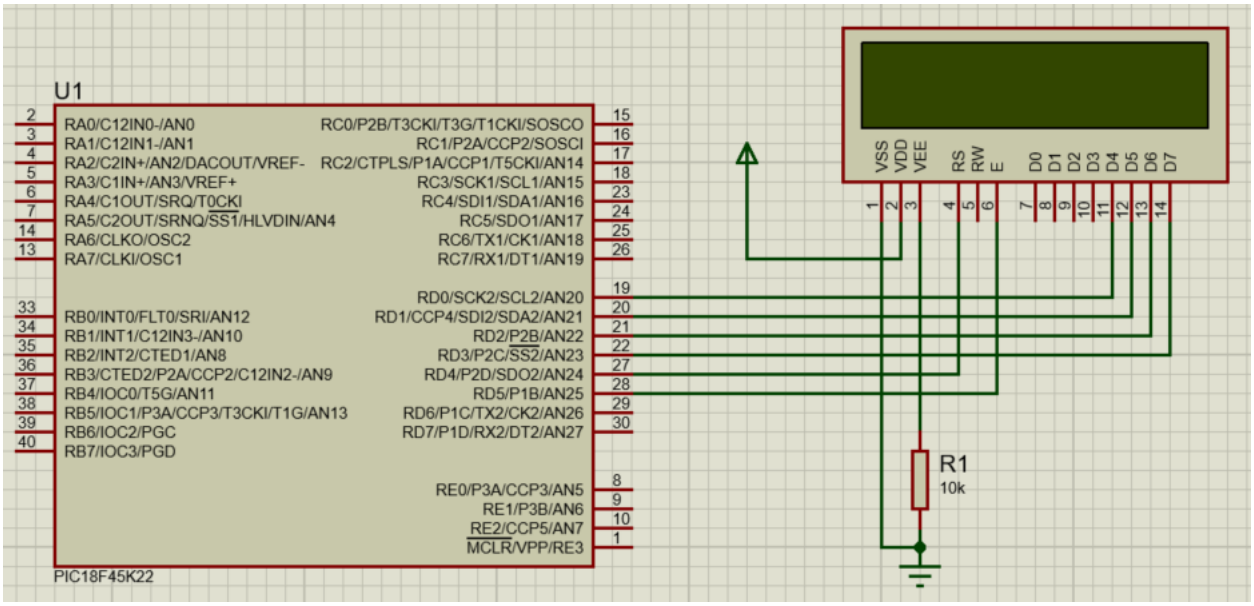
Tempo de operação 1,6ms

Limpa Display e memória

0b00000001

Tempo de operação 1,6ms

Configuração elétrica



Código inicialização 4 bits

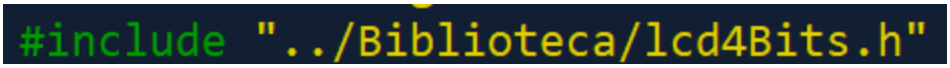
Displays LCDs recebem:

- Um comando em binário pelo seu barramento (8 bits);
- Um pulso de clock pelo enable para executar a função;
- Um sinal pelo RS para indicar ao display se ele vai receber um comando ou um caracetere
- Um sinal pelo R/W para indicar se é para escrever no display ou ler o display

Criando biblioteca

- Em seu projeto, crie um arquivo .c e outro arquivo .h, ambos com o nome LCD4Bits;
- No arquivo LCD4Bits.c, copie todas as funções de LCD e inclua o cabeçalho LCD4bits.h criado;
- No arquivo LCD4Bits.h, coloque todos os define, protótipos, defina o cristal e inclua o xc.h. Não se esqueça de criar o encapsulamento.
- No arquivo main.c, você pode apagar todas as funções destinadas ao display lcd e apenas incluir o cabeçalho LCD4Bits.h.

- Agora você pode pegar os arquivos de lcd e colocá-los em uma pasta separada de bibliotecas
- Remova os arquivos do projeto principal
- Altere o cabeçalho em main.c para encontrar o novo arquivo



- Inclua novamente o arquivo LCD4bits.c em seu projeto.

CGRAM

O comando para acessar a CGRAM inicia com o pino DB6 em nível lógico alto e os outros servem como endereçamento. Então, envie o comando 0x40 para acessar o primeiro endereço da CGRAM.

Agora, seu cursor está posicionado na CGRAM, você pode enviar no barramento cada pixel que formará o caractere desejado

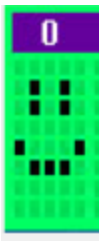
```
RS = 0; // LCD recebe comandos

0x40 // Comando para transferir o cursor para o primeiro endereço da CGRAM
E = 1;
E = 0; // clock

RS = 1; // LCD recebe caracteres

0b00000000
clock
0b00001010
clock
0b00001010
clock
0b00000000
clock
0b00010001
clock
0b00001110
```

Saída:



Agora, posicione novamente o cursor

```
lcdSetCursor(7, 1); // Posiciona o cursor na primeira coluna da primeira linha

// Mande um caracter com o endereço da CGRAM que você acabou de desenhar
lcdComando(1, 0x00);
```