PDF-XChange
Click to buy NOW!
www.docu-track.com

TUTORIAL

Ligando um LCD no 8051

PDF-XChange
Click to buy NOW!
www.docu-track.com

Índice

PDF-XChange
Click to buy NOW!
www.docu-track.com

Introdução

Esse tutorial tem como objetivo apresentar de forma simples e direta como fazer funcionar um LCD de 16 caracteres por 2 linhas em um Micro Controlador (MC) da família 8051.

Ainda vou produzir outro tutorial para explicar como simular, programar, gravar memória flash e executar programas no 8051. Mas neste tutorial vou supor que você já sabe fazer tudo isso.

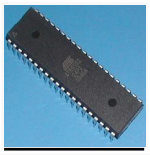
Material Necessário

Para fazer esse tutorial, você precisará dos seguintes componentes:

Tabela de componentes necessários para a montagem do tutorial. Informações retiradas do site http://www.SoldaFria.com.br em 04/07/2008. Os preços podem variar e são de total responsabilidade do site mencionado.

	Display LCD 16x2 Led Lateral P/ Back Fundo Verde	1	R$16,00
	Trimpot Multivoltas 3296W 10K	1	R$1,00
	Led Verde 8mm	1	R$0,45
	Resistor de 220R Carbono 5% 1/4W	1	R$0,10
	Resistor de 8K2 Carbono 5% 1/4W	1	R$0,10
	Capacitor Eletrolítico 10uF x 25V	1	R$0,15
	Cristal 12Mhz	1	R$0,90
	Capacitor Disco Cerâmico 33pF x 50V	2	R$0,20
	Chave Tactil de 7mm	1	R$0,20

PDF-XChange Click to buy NOW! www.docu-track.com

	Resistor de 100R Carbono 5% 1/4W	1	R$0,10
	Circuito Integrado Microcontrolador AT89S52	1	R$7,50
	Soquete 40 Pinos Estampado	1	R$0,60
	Protoboard S/ Base BB-01 (840 Pontos)	1	R$14,50

Sub-Total: R$41,80

Você pode encontrar todos esses componentes facilmente na Santa Ifigênia, na loja Mult Comercial (http://www.multcomercial.com.br/), ou ainda no site Solda Fria (http://soldafria.com.br/loja/).

Você também vai precisar dos seguintes datasheets:

- KS0066U – Driver de matriz de pontos e controlador de LCD; http://www.alldatasheet.com/datasheet-pdf/pdf/37317/SAMSUNG/KS0066U.html
- AT89S52 – Micro controlador de 8 bits; http://www.alldatasheet.com/datasheet-pdf/pdf/82390/ATMEL/AT89S52.html
- WinStar - WH1602 http://www.alldatasheet.com/datasheet-pdf/pdf/89421/ETC/WH1602.html

PDF-XChange
Click to buy NOW!
www.docu-track.com

Montagem

Como você pôde perceber o LCD não é apenas uma tela de cristal líquido. Ele é controlado por um MC próprio. O 8051 não conversa diretamente com o LCD, mas sim com esse MC, que no caso do LCD que vamos utilizar nesse tutorial, é o KS0066U.

Para conversar com esse MC é simples, basta trabalhar com 3 bits de controle e enviar os dados por uma porta de 8 bits.

Mas antes de entrar nos detalhes, vamos fazer a montagem do hardware, que é bem mais simples. Veja o esquema abaixo e siga as seguintes dicas:

PDF-XChange
Click to buy NOW!
www.docu-track.com

Montagem do Hardware

Para fazer esses laboratórios eu prefiro usar uma protoboard, pois dá muito trabalho fabricar uma placa de cobre e soldar tudo somente para fazer uma experiência. Então acho que o melhor é cortar vários cabinhos de cobre duro – ou soldar a ponta dos que são de feixe de fios – e fazer as ligações na protoboard.

Primeiro solde os cabinhos nos conectores do LCD, conforme figura abaixo.

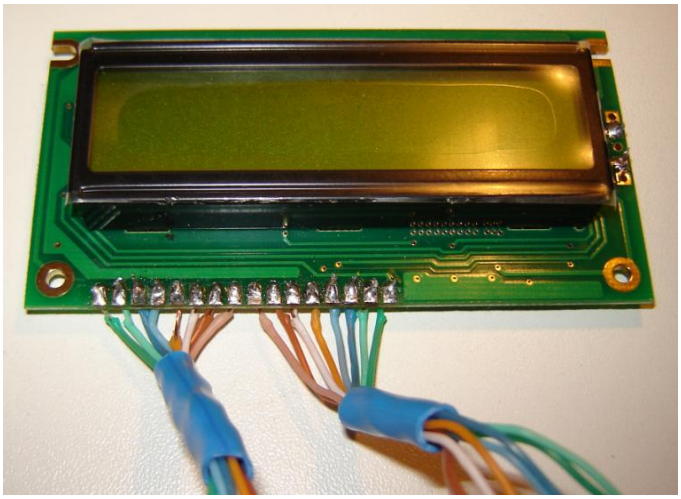
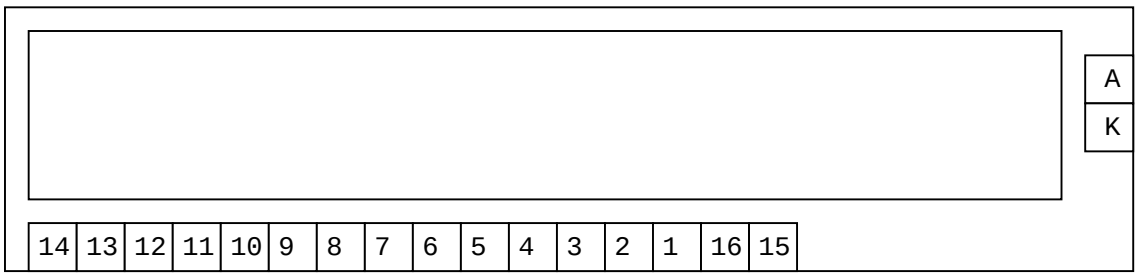


Figura 1 – Solde os cabinhos na placa do LCD.

Nota:

Este modelo de LCD WinStar WH1602A possui 18 terminais, dispostos da seguinte maneira:



- Onde:
  - 1: VCC (+5V)
  - 2: GND (Terra 0V)
  - 3: Controle de intensidade do display
  - 4: RS (Controla o tipo de comunicação: 0 para comandos e 1 para dados);
  - 5: RW (Controla o tipo de ação: 0 para escrita e 1 para leitura de dados);
  - 6: EN (Controle de envio: 0 para limpar status e 1 para executar o comando);
  - 7 até 14: Bus de dados com largura de 8 bits.
  - 15: Anodo (+) do LED de iluminação de fundo;
  - 16: Catodo (-) do LED de iluminação de fundo;
  - A: Anodo (+) do LED de iluminação de fundo;
  - K: Catodo (-) do LED de iluminação de fundo;
- Repare que a função dos pinos 15 e 16 e do A e K é a mesma, ou seja, ligar a luz de fundo do LCD.

Conecte agora o resitor de 8k2. Repare na foto que eu estava sem esse resistor e tive que improvisar.

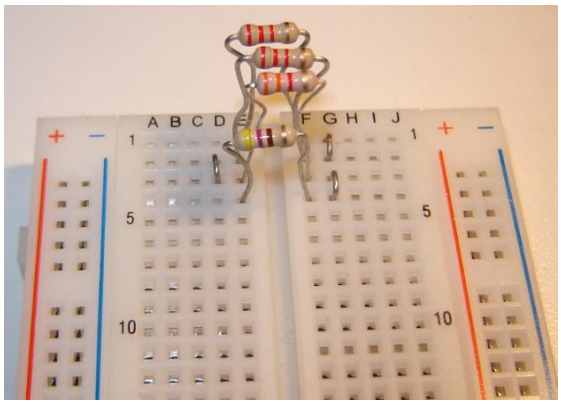


Figura 2 – Instalação do resistor de 8k2 (improvisado).

Instale então o micro controlador AT89S52 na protoboard.

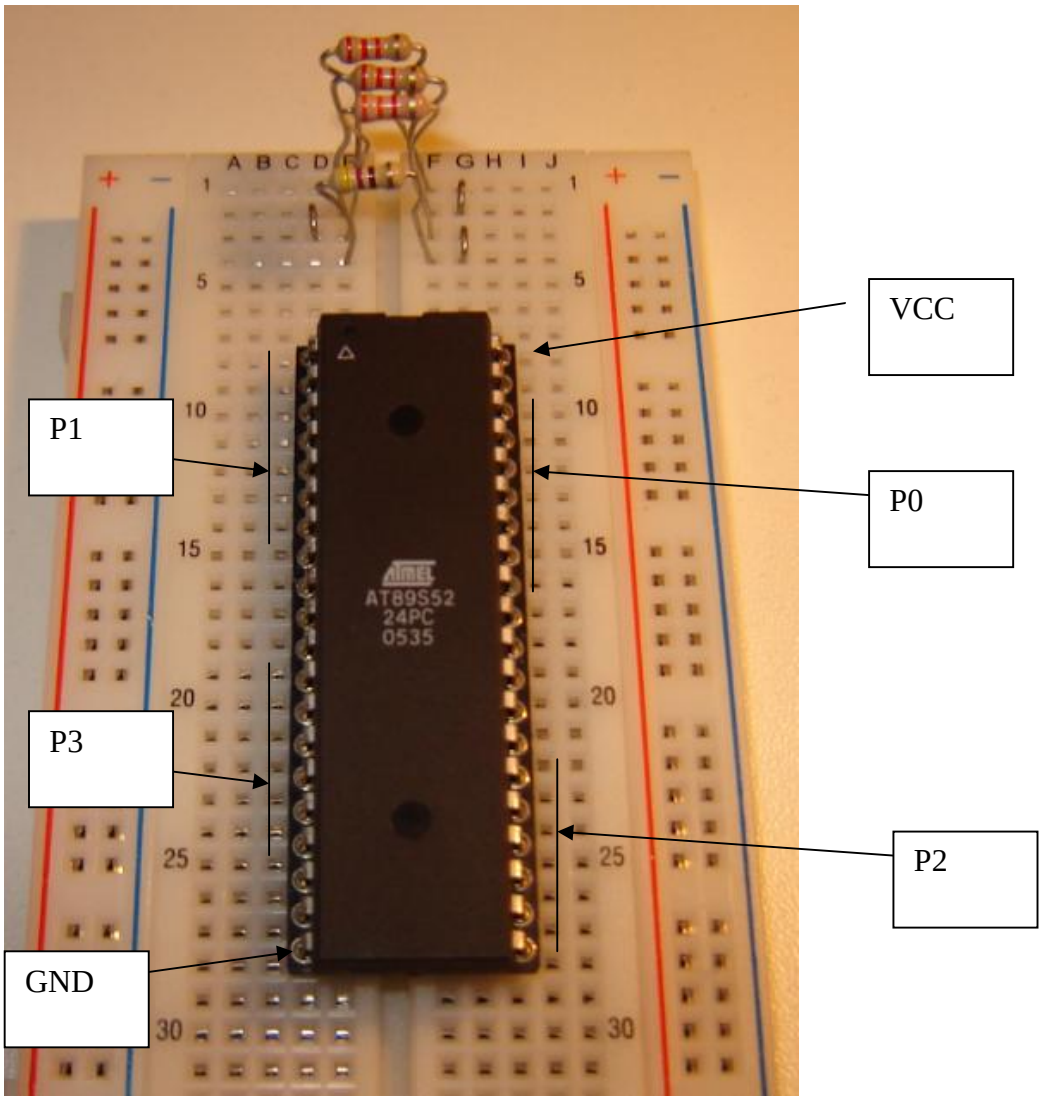


Figura 3 – Montagem do MC na protoboard.

Organize os fios do LCD e conecte-os na porta P2 do MC, conforme figura abaixo. E também ligue os outros fios nos furos logo abaixo do MC, pois mais para frente vamos jumpea-los na própria protoboard.

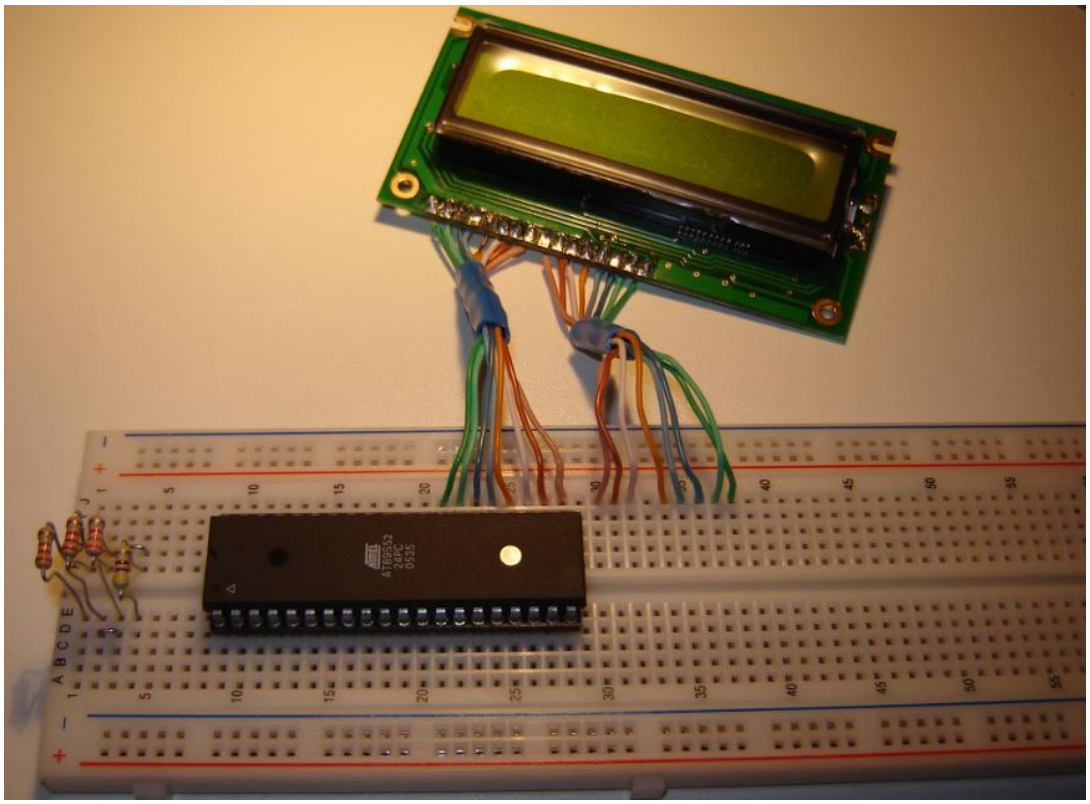


Figura 4 – Conexão dos fios do LCD na porta 2 do MC e nas vias do protoboard.

Nota:

- Para conectar o LCD, ligue os fios correspondentes aos conectores 7 até 14 na porta 2 do MC; Ou seja:
    - Fio do conector 7 do LCD no pino 1 da porta 2 do MC (P2.0);
    - Fio do conector 8 do LCD no pino 2 da porta 2 do MC (P2.1);
    - Fio do conector 9 do LCD no pino 3 da porta 2 do MC (P2.2);
    - Fio do conector 10 do LCD no pino 4 da porta 2 do MC (P2.3);
    - Fio do conector 11 do LCD no pino 5 da porta 2 do MC (P2.4);
    - Fio do conector 12 do LCD no pino 6 da porta 2 do MC (P2.5);
    - Fio do conector 13 do LCD no pino 7 da porta 2 do MC (P2.6);
    - Fio do conector 14 do LCD no pino 8 da porta 2 do MC (P2.7);

Agora faça os jumpers dos fios de controle do LCD.

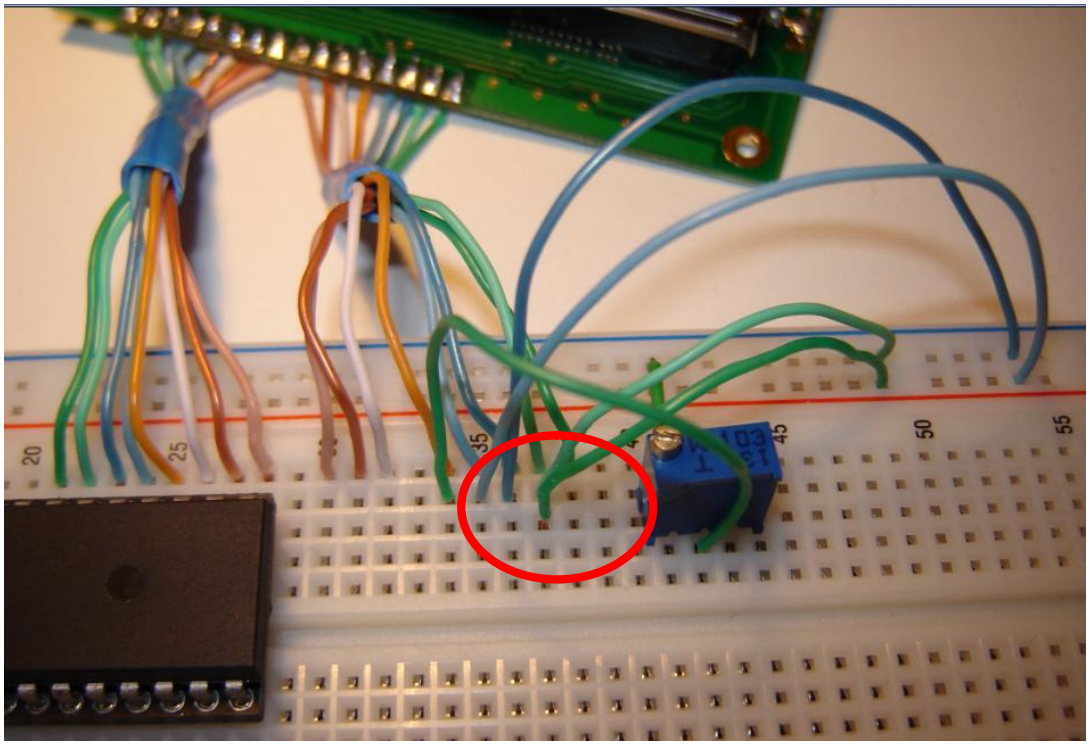


Figura 5 – Ligação da energia no LCD. (Em detalhe: fios ligados errados)

Nota:

Muita atenção na hora da montagem. Veja que eu montei errados os cabinhos de alimentação do pino 15 e 16 do LCD (que são responsáveis pela luz de fundo) e coloquei o sistema em curto. A sorte é que foi logo na entrada de energia e a fonte que eu estou usando é boa, de computador. Agüentou. Mesmo assim derreteu os fios e um pouco da protoboard.

- Para energizar o LCD ligue os terminais assim:
  - Fio do conector 1 do LCD no GND;
  - Fio do conector 2 do LCD no VCC;
  - Fio do conector 15 do LCD no VCC;
  - Fio do conector 16 do LCD no GND;
- Para o controle de intensidade do LCD, faça a seguinte ligação:
  - Fio do conector 3 do LCD em uma das pernas laterais do resistor variável e a perna central do resistor, ligue no terra;

Os outros conectores de controle que faltam são 4, 5 e 6, respectivamente:

- RS – Registrador de seleção de entrada. Quando RS = “Alto” (1), o modo de dados está selecionado. Quando RS = “Baixo” (0), o modo de instrução está selecionado. Através desse registrador o LCD sabe se o que será enviado é dado ou instrução.
- RW – Registrador de Leitura/Escrita. Quando RW = “Alto” (1), modo de leitura. Quando RW = “Baixo” (0), modo de escrita.
- EN – Registrador de ativação de Leitura/Escrita. Ou seja, esse é o registrador que executa os comandos de instrução ou de dado, tanto para leitura quanto para escrita.

Conecte os terminais dos pinos comentados acima conforme figura abaixo.

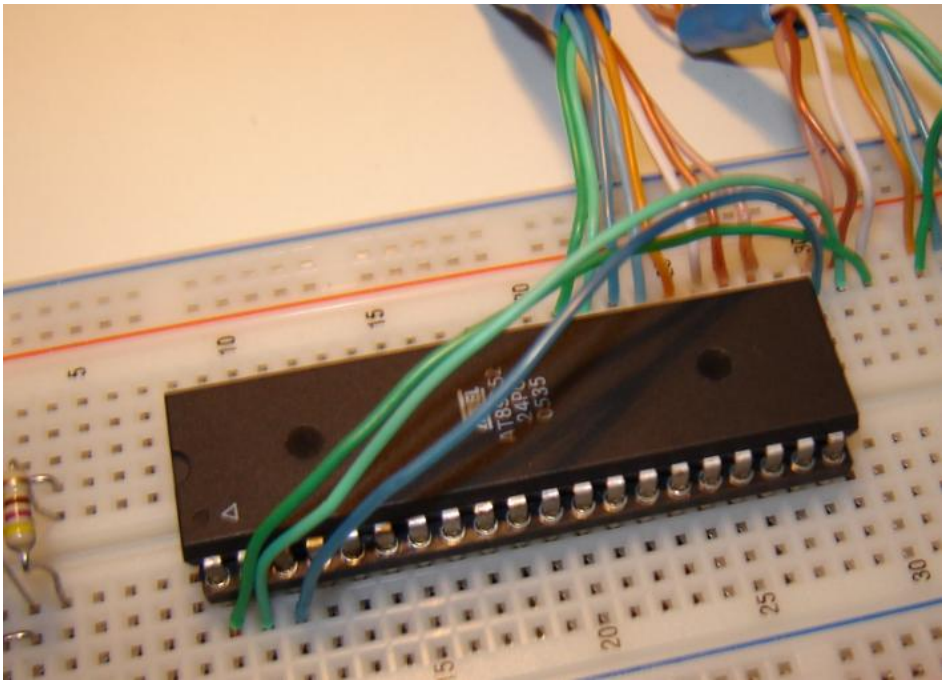


Figura 6 – Ligação dos fios dos registradores de controle do LCD na porta 1 do MC.

Nota:

- Mapeamento dos fios RS, RW, EN:
  - Fio do conector 4 do LCD no pino 1 da porta 1 do MC (P1.0);
  - Fio do conector 5 do LCD no pino 2 da porta 1 do MC (P1.1);
  - Fio do conector 6 do LCD no pino 3 da porta 1 do MC (P1.2);

Voltando ao resitor de 8k2 que colocamos na primeira etapa do tutorial, ligue uma das pontas desse resistor no GND e a outra ponta, no negativo do capacitor eletrolítico de 10uF. Ligue o positivo do capacitor eletrolítico no VCC. Com isso estamos montando o circuito de reset. Para finalizar, ligue o pino 9 do MC também no negativo do capacitor.

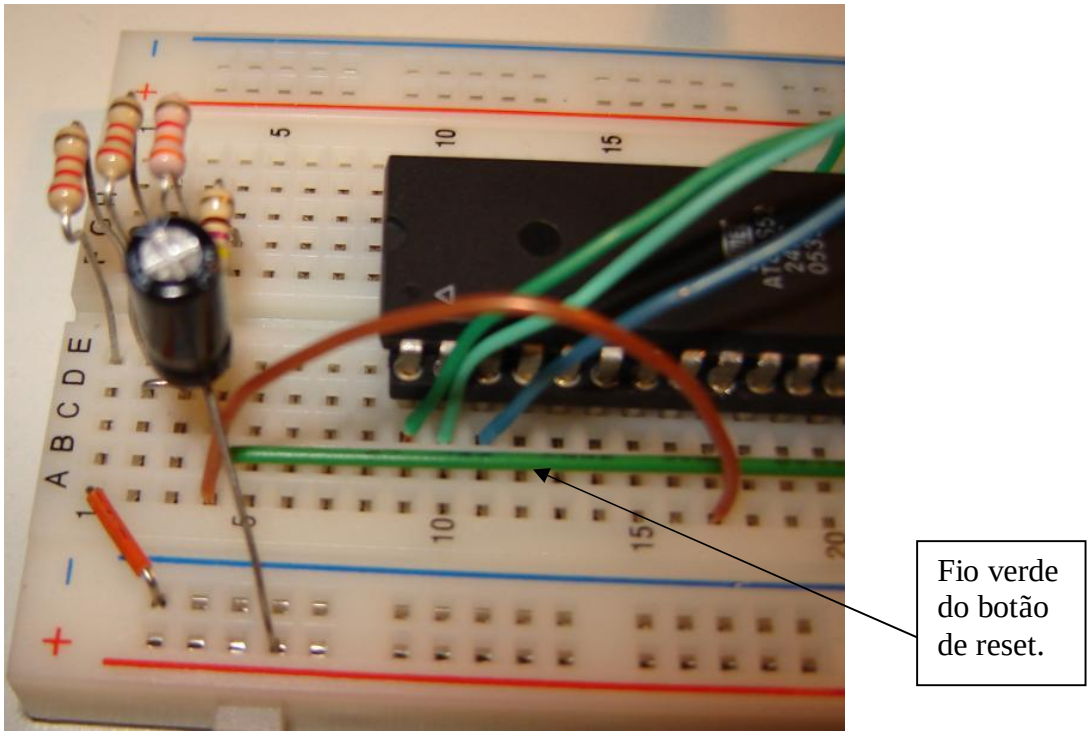


Figura 7 – Montagem do circuito de reset.

Nota:

Agora repare na foto, um fio verde rente a placa da protoboard conectado também no negativo do capacitor. Esse fio vai até o outro lado para a conexão do botão de reset. Veja a próxima foto.

Monte a chave táctil no final do fio verde. No terminal oposto da chave, ligue o resitor de 100 ohms e a outra ponta do resistor, ligue no VCC.

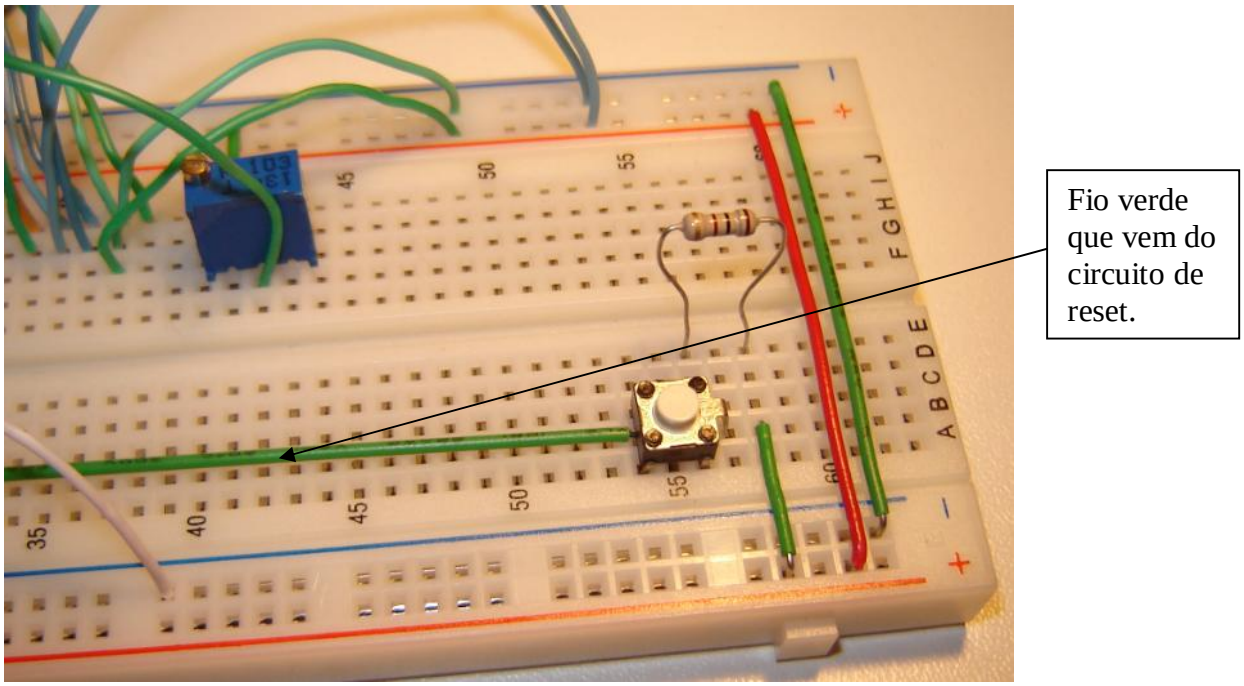


Figura 8 – Montagem do circuito de reset (parte 2).

Nota:

Para saber quais são os contatos que formam o interruptor em uma chave táctil de 4 patas, utilize um multímetro em modo de aferição de resistência. Vá testando as patas e apertando o botão para saber se dá contato.

Para verificar se, quando ligarmos a força, pelo menos a alimentação de energia estará funcionando, eu coloquei um led verde na entrada de energia. Conecte o catodo (K), que é a perna mais curta do led, no GND e a outra perna, o anodo (A), no resistor. Finalizando, ligue a outra perna do resistor no VCC.

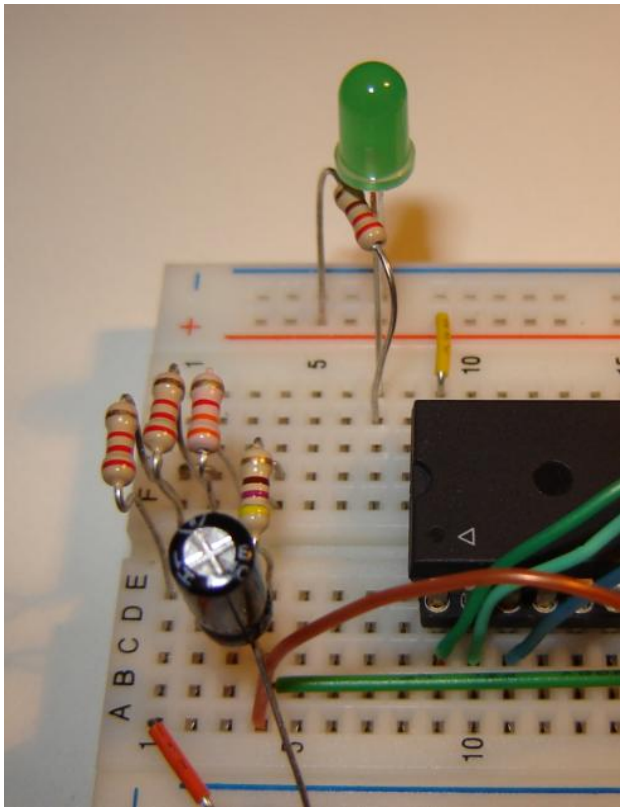


Figura 9 – Led indicador de funcionamento.

Conecte o cristal nas patas 18 e 19 do MC. Conecte também os capacitores de 33pF em cada perna do cristal e o outro lado no GND.

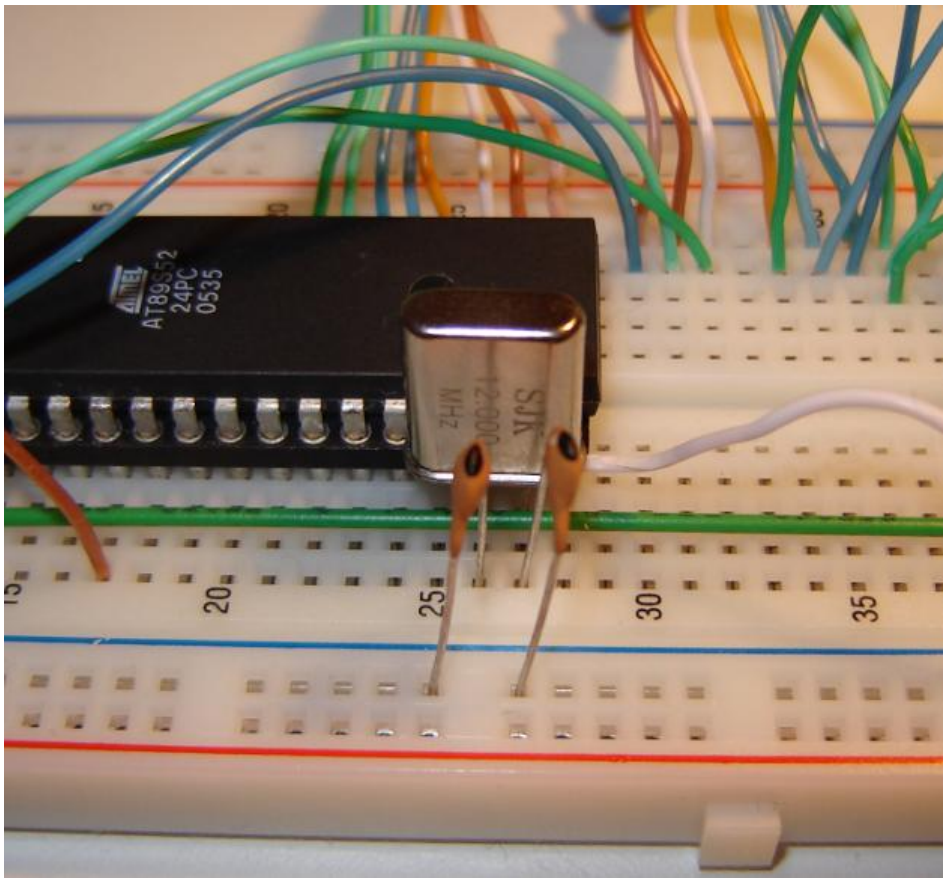


Figura 10 – Montagem do cristal.

Nota:

Não convém colocar o cristal muito longe do MC. Ele serve como fonte estabilizadora do oscilador interno do controlador. Quanto menos interferência externa houver nesse circuito, melhor. Diminuindo a distância, ajuda em muito a diminuir ruídos.

Para terminar a parte de hardware, ligue o DB9 fêmea no circuito de gravação do MC.

Figura 11 – Conector do circuito de gravação do MC.

Nota:

Se você não conhece essa parte, não se preocupe, constará no primeiro tutorial da série 8051.

Conecte o cabo de dados com o PC.

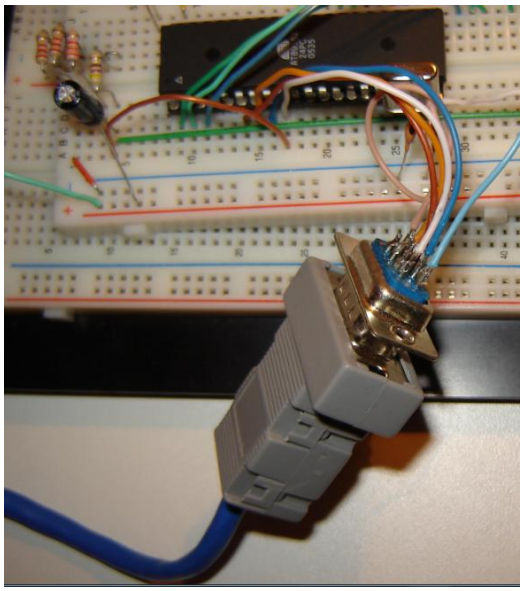


Figura 12 – Conectando o cabo com o PC.

Inicie o ISP Flash Programmer e grave o programa que vamos desenvolver na parte de codificação desse tutorial.

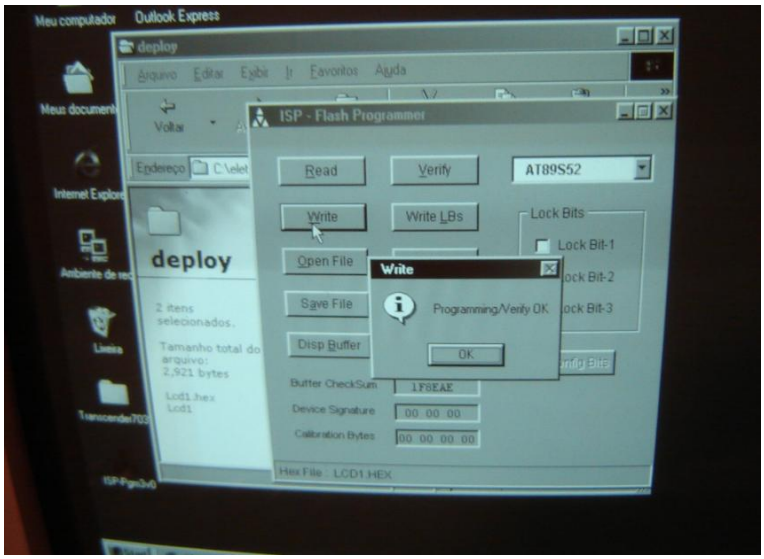


Figura 13 – Tela do programa de gravação da memória flash do MC.

Nota:

Tudo isso estará explicado no primeiro tutorial da série, não se preocupe.

A montagem final deve ficar como a da foto abaixo.

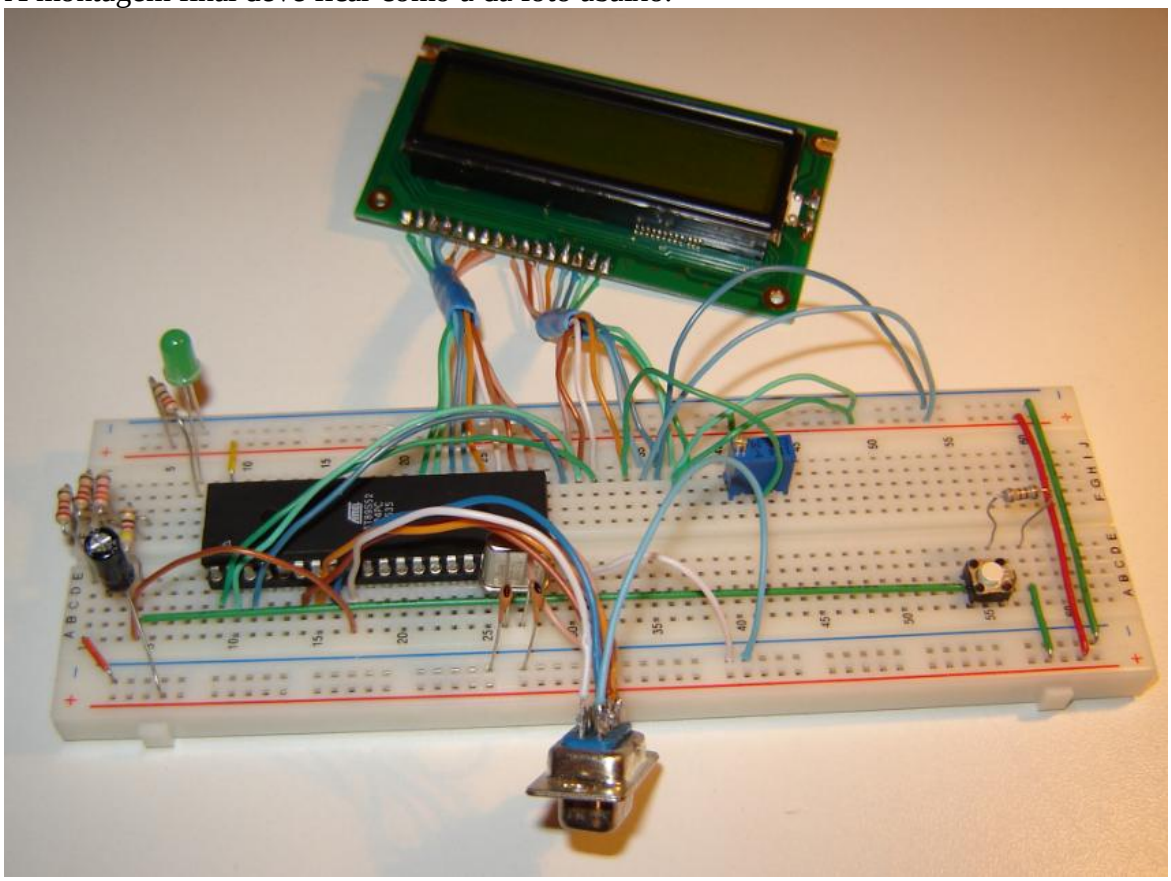


Figura 14 – Aparência final da montagem.

O resultado final da execução do programa é o aparecimento da frase “Hello World” na tela do LCD.

Figura 15 – Resultado final do tutorial.

Esquema de Ligação dos Componentes

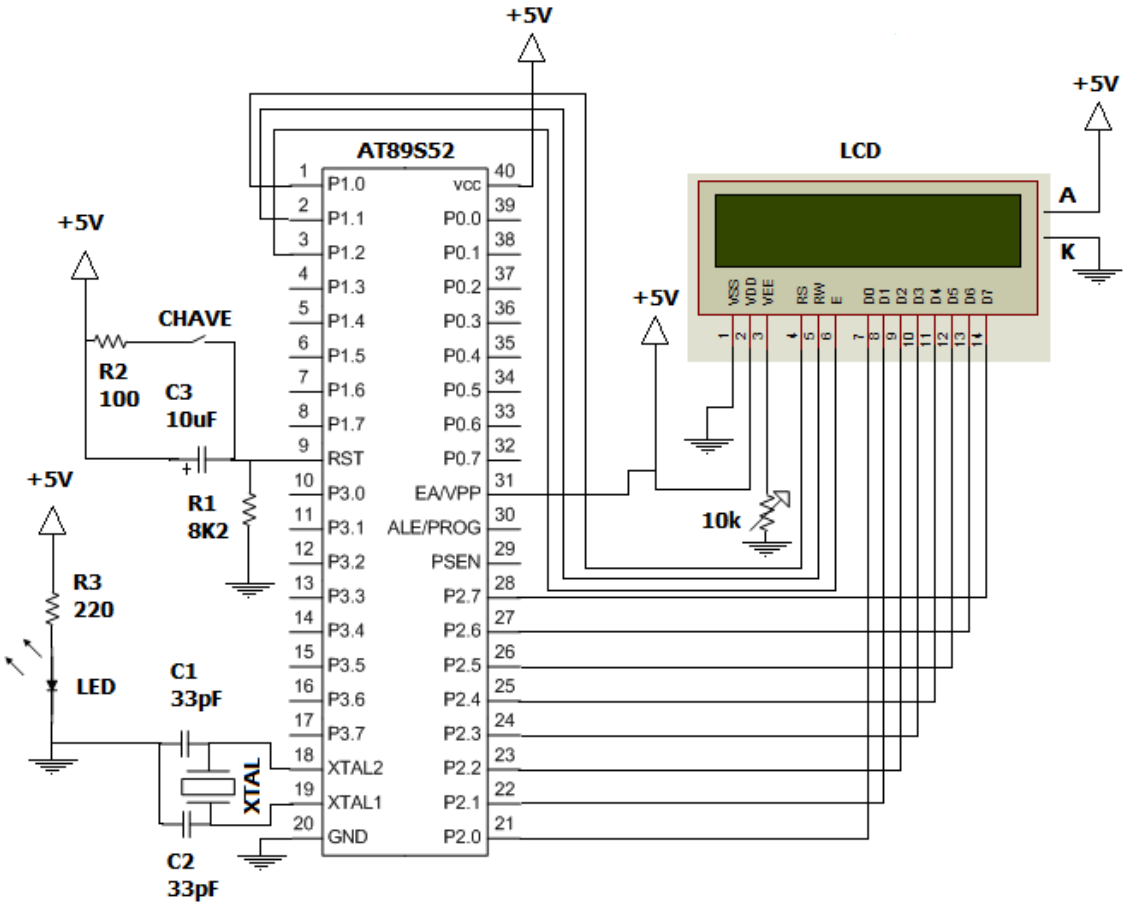


Figura 16 – Esquema de ligação dos componentes.

Nota:

Comentários sobre as ligações:

- Não se esqueça de colocar o EA\ em “Alto”, senão o programa interno do MC não rodará;
- Repare que no esquema acima, a ligação da luz de fundo é feita pelos conectores da lateral direita do LCD, mas na protoboard fizemos a ligação pelos conectores 15 e 16. Ambos conectores tem a mesma funcionalidade;
- Deixe o trimpot com mais ou menos 7k ohms, depois que o circuito funcionar, faça o ajuste fino com uma pequena chave de fenda.

Agora acompanhe a parte de criação do programa que fará escrever a frase “Hello World” na tela do LCD.

PDF-XChange
Click to buy NOW!
www.docu-track.com

Montagem do Software

Antes de entramos nos detalhes da programação, devo falar um pouco de teoria. O LCD é um componente complexo e, ao contrário do que desejaríamos, não é só “sair escrevendo”. Precisamos primeiro entender um pouco do funcionamento dessa peça que é composta de duas partes: um display de cristal líquido e um micro controlador de comunicação.

Esses fios que estamos ligando entre o LCD e o MC são internamente como se fossem pernas de outro MC. Estamos então ligando um MC no outro. Essa conexão direta nos permite “falar” diretamente com o LCD. Alterando os estados das portas no MC conseguimos diretamente acionar portas no LCD.

Trocando em miúdos, precisamos configurar os estados das portas de controle, (RS, RW e EN) e configurar as informações que desejamos escrever ou ler na porta P2. Por isso logo na parte de montagem do hardware eu já citei os PDFs que você precisaria. No datasheet do KS0066U você encontra informações sobre todos os comandos de comunicação com o LCD.

Mas para deixar esse tutorial simples, vamos fazer o básico. Para escrever a frase “Hello World” na tela do LCD, precisamos seguir o seguinte fluxograma:

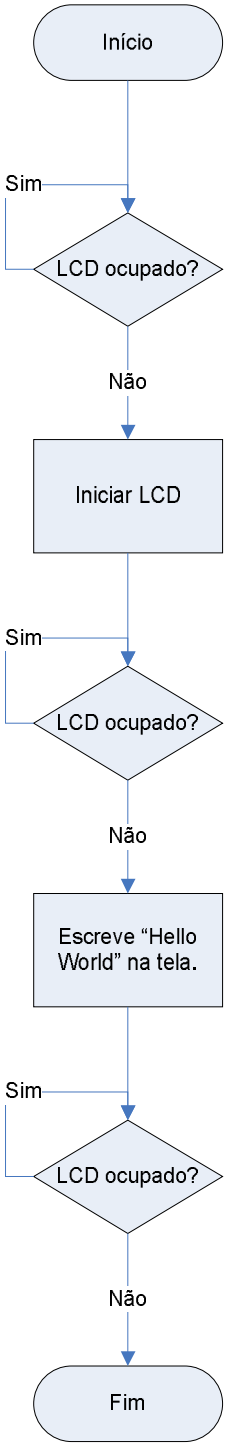


Figura 17 – Fluxograma da aplicação que escreve “Hello World” na tela do LCD.

Nota:

Repare no fluxo, a preocupação da aplicação com o estado de ocupado do LCD. É preciso sincronizar os comandos para que não esbarrem em um estado de ocupado do LCD.

Então a primeira coisa que devemos ver é a declaração de variáveis e atribuições:

```
TEMPO   EQU     60535
LCD     EQU     P2
D0      EQU     P2.0
D1      EQU     P2.1
D2      EQU     P2.2
D3      EQU     P2.3
D4      EQU     P2.4
D5      EQU     P2.5
D6      EQU     P2.6
D7      EQU     P2.7
EN      EQU     P1.2
RW      EQU     P1.1
RS      EQU     P1.0
```

Nota:

- A primeira declaração de variável é a de tempo, para usar na configuração do timer com 5 ms de atraso;
- A variável LCD aponta para o port 2 como um todo, ou seja, todos os 8 bits;
- E as variáveis D0..D7 também apontam para os bits da porta 2, só que individualmente;
- Finalmente, EN, RW e RS apontando para os bits da porta 1 do MC.

Na sequência, precisamos de um jump para pular as áreas da memória reservadas para as interrupções. Vamos colocar no endereço 0h de memória uma instrução de long jump para ir buscar o ponto real de inicio do programa, no endereço 040h.

```
; Iniciando programa
PRE_START:
        ORG     0000H
        LJMP    START
```

Nota:

O ponto de long jump chamado `START` é configurado logo abaixo. Veja próxima parte.

Para iniciar o programa, primeiro configuramos o ponto de memória de origem, que deve ter o valor 040h para funcionar com a parte acima. Depois iniciamos a programação do fluxograma.

```
; Pulando as interrupções
START:
        ORG     0040H
```

Nota:

A partir desse ponto começa realmente a programação do que foi definido no tutorial.

Agora vamos ao fluxo do programa. A primeira ação após o início é verificar se o LCD não está ocupado.

```
        ; Verificando primeiro se o LCD está desocupado
        LCALL   CHECK
```

Nota:

Coloquei essa verificação de ocupado em uma função chamada `CHECK`. Mais adiante veremos como ela funciona.

Na sequência devemos iniciar o LCD para, fazendo isto, configurá-lo. Precisamos dizer a ele que vamos conversar em oito bits, que o display tem duas linhas, retornar o cursor a posição inicial, escolher se queremos ver o cursor, essas coisas.

```
; Iniciando o LCD
    CLR     EN                  ; Limpando EN
    CLR     RW                  ; Limpando RW
    CLR     RS                  ; Limpando RS
    MOV     LCD, #38H           ; Alimentando porta de dados com
                                ; instrução 38h.
    SETB    EN                  ; Executa!

    ; Verificando se o LCD está desocupado
    LCALL   CHECK

    CLR     EN
    CLR     RW
    CLR     RS
    MOV     LCD, #38H           ; De novo.
                                ; Ainda não entendi porque.
    SETB    EN

    ; Verificando se o LCD está desocupado
    LCALL   CHECK

    CLR     EN
    CLR     RW
    CLR     RS
    MOV     LCD, #06H           ; Alimentando porta de dados com
                                ; instrução 06h.


    SETB    EN

    ; Verificando se o LCD está desocupado
    LCALL   CHECK

    CLR     EN
    CLR     RW
    CLR     RS
    MOV     LCD, #0FH           ; Alimentando porta de dados com
                                ; instrução 0Fh.

    SETB    EN

    ; Verificando se o LCD está desocupado
    LCALL   CHECK
```

```
CLR     EN
CLR     RW
CLR     RS
MOV     LCD, #01H           ; Alimentando porta de dados com
                            ; instrução 01h.

SETB    EN

; Verificando se o LCD está desocupado
LCALL   CHECK
```

Nota:

Repare que a rotina de iniciação do LCD é composta de cinco comandos, na seguinte ordem:

- Executar duas vezes o `38h`.
- Executar uma vez o `06h`.
- Executar uma vez o `0Fh`.
- Executar uma vez o `01h`.

Entre cada um desses comandos sempre verificamos se o LCD não está ocupado com tarefas internas.

Cada comando, conforme o datasheet do produto equivale a uma instrução. Primeiro nós configuramos os registradores de controle, conforme a tabela abaixo:

Tabela 1 – Configuração dos registradores RS e R/W e os modos de operação.

RS	R/W	Modo
L	L	Modo de escrita de instrução.
L	H	Ler o registrador indicador de ocupado.
H	L	Modo de escrita de dados.
H	H	Modo de leitura de dados.

L = Low (nível lógico baixo); H = High (nível lógico alto).

Depois podemos configurar os registradores de dados para escrever o comando.

É simples, veja primeiro limpamos (ou seja, colocamos em nível lógico baixo) `EN`, apenas para garantir:

```
CLR   EN
```

Depois configuramos `RS` e `R/W` conforme tabela acima. Para registrar as configurações de iniciação do LCD, precisamos de `RS` e `R/W` em nível lógico baixo, pois precisamos dizer ao LCD que vamos conversar em modo de escrita de instrução.

```
CLR   RS
CLR   RW
```

Depois de configurado o que será conversado, precisamos dizer uma frase. Essa frase é o comando. Conforme o manual do produto, vamos utilizar `38h`, que equivale ao comando: Function Set com parâmetros.

Funciona assim: as frases são montadas em dois nibbles, ou seja, em oito bits. Transformando esses nibbles em hexadecimal, temos dois dígitos, como por exemplo: `38h`. Veja tabela abaixo.

Tabela 2 – Frase de dados do comando Funcion Set com parâmetros.

Nibble 1					Nibble 2				Hex.
0	0	1	1		1	1	0	0	38h

Se fizermos as contas, fica fácil.

Tabela 3 – Frase de dados desmontada para calcular os parâmetros.

Nibble 1					Nibble 2				Hex.
0	0	1	0	+	0	0	0	0	20h
0	0	0	1	+	0	0	0	0	10h
0	0	0	0	+	1	0	0	0	08h
0	0	0	0	+	0	0	0	0	00h
0	0	1	1	=	1	0	0	0	38h

Onde:

- A frase `20h` é a chamada do comando Function Set em si;
- A frase `10h` é a configuração do comprimento da interface de dados, ou seja, quatro bits ou oito bits. No nosso caso, temos uma via de dados de oito bits através da porta P2 do MC, então vamos colocar o nível desse registrador em alto;
- A frase `08h` configura o registrador do controle de números de linhas de texto do display. Podemos configurar em alto para o modo de duas linhas de texto, ou baixo para o modo de apenas uma linha de texto. Como nosso display tem duas linhas, vamos configurar esse registrador em alto.
- A frase `00h` (veja o bit em vermelho na tabela acima) configura o tipo de fonte (tipologia) do display. Como foi colocada em nível baixo, o tipo da fonte é 5 x 8 pontos. Se essa instrução tivesse o valor `04h`, mudaria a frase toda para `3Ch` (faça as contas `38h + 04h = 3Ch`) e alteraria a fonte para 5 x 11 pontos.

Nota:

O comando `38h` é executado duas vezes. Não sei exatamente a razão, mas o exemplo que eu peguei estava assim e também não explicou o porquê.

Continuando, movemos então o valor `38h` para a porta P2 do MC, que está mapeada para a variável LCD do programa.

```
MOV   LCD, #38H
```

E finalmente para executar o comando, precisamos levantar o valor lógico de `EN`. Assim estamos dizendo ao controlador do LCD que já configuramos os registradores (de controle e de dados) e que vamos executar a requisição.

```
SETB    EN
```

Pelo código postado mais acima, você perceberá que é um processo cíclico bem rústico e que pode ser melhorado depois, mas basicamente funciona assim:

1. Limpa o `EN` para resetar a operação;
2. Configura `RS` e `RW` para “dizer” ao LCD qual modo será adotado;
3. Move os dados do comando para a porta P2 do MC, que está diretamente ligada a via de dados do LCD;
4. Coloca `EN` em nível lógico alto para indicar ao controlador do LCD que pode executar a tarefa;
5. Verifica o registrador de busy do LCD. Ou seja, fica em loop enquanto o LCD está ocupado.

Essa rotina acima é feita cinco vezes para executar os comandos. O comando `38h` já foi citado acima. Segue abaixo a tabela dos outros três comandos: `06h`, `0Fh` e `01h`.

PDF-XChange
Click to buy NOW!
www.docu-track.com

O comando `06h` equivale a função: Entry Mode Set. Esta instrução configura o comportamento do cursor e do display.

Instrução completa com parâmetros:

Tabela 4 - Frase de dados do comando Entry Mode Set com parâmetros.

Nibble 1					Nibble 2				Hex.
0	0	0	0		0	1	1	0	06h

Detalhamento dos parâmetros:

Tabela 5 - Frase de dados desmontada para calcular os parâmetros.

Nibble 1					Nibble 2				Hex.
0	0	0	0	+	0	1	0	0	04h
0	0	0	0	+	0	0	1	0	02h
0	0	0	0	+	0	0	0	0	00h
0	0	0	0	=	0	1	1	0	06h

Onde:

- `04h` é o comando em si;
- `02h` diz ao LCD que o cursor se deslocará para a direita após a inclusão de um caractere no display. Se colocarmos esse bit em nível baixo, teremos `00h` nesse parâmetro e o comando mudaria para `04h`, e nessa condição, o cursor se desloca para a esquerda;
- `00h` poderia ser `01h` e mudar o comando para `07h`, dizendo para o LCD inverter todo o display. Ainda não estudei essa funcionalidade e não sei como funciona.

O comando `0Fh` equivale a função: Display ON/OFF Control. Esta instrução efetivamente liga ou desliga o display, mas também controla a exibição do cursor e seu modo de piscar.

Instrução completa com parâmetros:

Tabela 6 - Frase de dados do comando Display ON/OFF Control com parâmetros.

Nibble 1					Nibble 2				Hex.
0	0	0	0		1	1	1	1	0Fh

Detalhamento dos parâmetros:

Tabela 7 - Frase de dados desmontada para calcular os parâmetros.

Nibble 1					Nibble 2				Hex.
0	0	0	0	+	1	0	0	0	08h
0	0	0	0	+	0	1	0	0	04h
0	0	0	0	+	0	0	1	0	02h
0	0	0	0	+	0	0	0	1	01h
0	0	0	0	=	1	1	1	1	0Fh

Onde:

- `08h` é o comando em si;
- `04h` efetivamente liga ou desliga o visor do LCD;
- `02h` indica que o cursor será visível. Para desligar o cursor, coloque `00h` nesse parâmetro para ter o comando `0Dh`;
- `01h` indica que o cursor ficará em modo piscante. Para configurar o modo estático, coloque esse bit em nível baixo para ter o comando `0Eh`.

O comando 01h equivale a função: Clear Display. Esta instrução limpa o display, escrevendo o caractere “em branco” em todas as posições da memória de exibição. Coloca também o cursor na posição inicial, totalmente a esquerda na primeira linha. Configura o modo de avanço do cursor para o padrão incremental a direita.

Instrução completa com parâmetros:

Tabela 8 - Frase de dados do comando Clear Display.

Nibble 1					Nibble 2				Hex.
0	0	0	0		0	0	0	1	01h

Detalhamento dos parâmetros:

Tabela 9 - A função Clear Display não tem parâmetros.

Nibble 1					Nibble 2				Hex.
0	0	0	0	+	0	0	0	1	01h
0	0	0	0	+	0	0	0	0	00h
0	0	0	0	=	0	0	0	1	01h

Onde:

- 01h é o comando em si;
- 00h é só pra indicar que a função clear display não tem parâmetros.

PDF-XChange
Click to buy NOW!
www.docu-track.com

Agora vamos escrever a frase “Hello World” no LCD. Para fazer isso utilizei um truque do compilador. Utilizei uma instrução especial chamada `DB`. Veremos sobre essa instrução mais abaixo, mas adianto que é um ponteiro de memória para guardar dados. Então, no código abaixo veremos que basta eu pegar o endereço inicial e, através de um loop, ir lendo os endereços de memória seguintes até o final.

```
       ; Escrevendo alguma coisa
WRITE_STR:
       MOV     R0, #11D
       MOV     DPTR, #0500H
WRITE_NOW:
       CLR     A
       MOVC    A, @A + DPTR
       LCALL   WRITE_LCD
       INC     DPTR
       DJNZ    R0, WRITE_NOW
```

Nota:

Veja que na parte inicial, em `WRITE_STR`, gravamos o valor onze em decimal no banco `R0` de memória. Esse valor nos servirá como contador invertido.

A próxima instrução aponta `DPTR` para o endereço `500H` que é onde começa nosso `DB`.

Agora na parte do `WRITE_NOW` temos primeiro a limpeza de `A`, colocando-o em nível baixo.

Na seqüência utilizamos o comando `MOVC` para indicar que trabalharemos com leitura de dados da ROM, que é a memória destinada ao código. O parâmetro `@A + DPTR` soma o endereço de `A` com o de `DPTR` o resultado é usado na instrução. Essa instrução pega o valor nesse endereço e coloca em `A`. No nosso tutorial, o valor que a instrução vai pegar e o que está no endereço `500h` da memória ROM. Este é o endereço onde colocamos a nossa frase “Hello World”.

Vamos esclarecer algumas coisas, primeiro: O comando `MOVC` move apenas oito bits por vez. Então `A` terá a cada ciclo o valor de uma letra por vez, pois pela tabela do ASCII, cada caractere ocupa oito bits. Na verdade são sete, mas não vem ao caso agora.

No primeiro ciclo teremos o valor dos oito primeiros bits do endereço `500h`, que é justamente a letra “H”.

Logo após movermos a primeira letra para `A`, chamamos a função `WRITE_NOW` para escrever a letra no LCD.

Agora precisamos incrementar `DPTR` para que no próximo loop, peguemos o endereço seguinte e conseqüentemente a letra seguinte.

O comando `DJNZ` fica em loop enquanto `R0` for diferente de zero. E se o valor for maior que zero, decrementa e pula para o início do laço. Dessa forma, cada passada por esse ponto decrementa o valor de onze (que é a quantidade de caracteres) e fica nesse loop até esse valor atingir zero. Com isso, o loop rodará até que todos os onze caracteres da frase “Hello World” sejam escritos no LCD.

PDF-XChange
Click to buy NOW!
www.docu-track.com

Para finalizar a parte principal de escrita da frase no LCD, temos um jump para o final do programa. Para pular os endereços das funções.

```
        ; Vai para o fim do programa
        LJMP    FIM
```

E agora, a parte das funções. Coloquei no endereço `200h` para não dar overlap no código principal.

```
; Funcoes auxiliares
ORG     0200H
```

Primeiro temos o temporizador de 5ms. Não tem nada de mais, configuramos o valor de seus registradores e o iniciamos. Este então fica em loop até que o tempo de 5ms tenha passado.

```
; Temporizador de 5 ms
TIMER:
        MOV     TL0, #LOW(TEMPO)                    ; Configura o timer
        MOV     TH0, #HIGH(TEMPO)
        SETB    TR0
        JNB     TF0, $
        CLR     TR0
        CLR     TF0
        RET
```

A função CHECK já é mais interessante. Também se trata de uma rotina temporizadora, mas baseada no ciclo de tarefas do LCD. O segredo aqui é enviar um comando específico para o LCD e verificar o bit sete da resposta. Esse bit sete tem a informação de se o dispositivo está ocupado, ou não.

```
CHECK:
        ; Verificando se o LCD está disponível
        CLR     EN
        CLR     RS
        SETB    RW
        MOV     LCD, #0FFH
        SETB    EN
        JB      D7, CHECK
        CLR     EN
        CLR     RW
        RET
```

Nota:

Vamos estudar esse código acima. Primeiro como de costume zeramos EN.

```
        CLR     EN
```

Depois vamos configurar RS e RW. Conforme a Tabela 1, vamos utilizar o modo “Ler o registrador indicador de ocupado”. Para isso precisamos do RS em nível baixo, indicando que é uma instrução, e o RW em nível alto, indicando que é leitura.

```
        CLR     RS
        SETB    RW
```

A instrução abaixo limpa a porta P2 configurando todos os bits com o nível lógico alto. Essas instruções são importantes para garantir um nível lógico. Quando o MC é iniciado, muitas portas são deixadas em um estado de “nível lógico fraco”. Ou seja, garantir que elas vão estar com um nível lógico esperado é uma boa prática.

```
        MOV     LCD, #0FFH
```

Chegou o momento de executar a tarefa. Por isso configuramos EN em nível alto.

```
        SETB    EN
```

Se o LCD estiver ocupado, ele vai responder colocando o bit 7 da porta P2 em nível alto. Por isso, na seqüência utilizamos o JB para verificar o estado do bit sete e fazer o jump se ele estiver em nível alto.

```
        JB      D7, CHECK
```

Para finalizar, limpamos EN, RW e retornamos.

```
        CLR     EN
        CLR     RW
        RET
```

PDF-XChange
Click to buy NOW!
www.docu-track.com

A última função é a que escreve dados no LCD. Nada de muito complicado, configuramos o RW com nível baixo, para indicar escrita e RS em nível alto, para indicar dados. Por fim colocamos o valor de A na porta de dados do LCD e executamos.

```
WRITE_LCD:
       CLR     EN
       CLR     RW
       SETB    RS
       MOV     LCD, A
       SETB    EN
       LCALL   CHECK
       RET
```

Abaixo temos os dados gravados diretamente na ROM, junto com o programa. O código abaixo apenas define um ponto de memória para iniciar o banco que armazena a frase “Hello World” neste endereço.

```
; Dados internos do programa
INTERNAL_DATA:
       ORG     0500H
       DB      'HELLO WORLD'
```

Para finalizar, temos o ponto de saída do programa. Esta instrução está gravada na penúltima posição da memória ROM.

```
; Fim do programa
ORG    1FFEH
FIM:
       END
```

PDF-XChange
Click to buy NOW!
www.docu-track.com

Código Completo

```
TEMPO  EQU     60535
LCD    EQU     P2
D0     EQU     P2.0
D1     EQU     P2.1
D2     EQU     P2.2
D3     EQU     P2.3
D4     EQU     P2.4
D5     EQU     P2.5
D6     EQU     P2.6
D7     EQU     P2.7
EN     EQU     P1.2
RW     EQU     P1.1
RS     EQU     P1.0


; Iniciando programa
PRE_START:
       ORG     0000H
       LJMP    START


; Pulando as interrupções
START:
       ORG     0040H

       ; Verificando primeiro se o LCD está desocupado
       LCALL   CHECK

       ; Iniciando o LCD
       CLR     EN
       CLR     RW
       CLR     RS
       MOV     LCD, #38H
       SETB    EN
       LCALL   CHECK
       CLR     EN
       CLR     RW
       CLR     RS
       MOV     LCD, #38H
       SETB    EN
       LCALL   CHECK
       CLR     EN
       CLR     RW
       CLR     RS
       MOV     LCD, #06H
       SETB    EN
       LCALL   CHECK
       CLR     EN
       CLR     RW
       CLR     RS
       MOV     LCD, #0FH
       SETB    EN
       LCALL   CHECK
       CLR     EN
       CLR     RW
```

PDF-XChange
Click to buy NOW!
www.docu-track.com

```
        CLR     RS
        MOV     LCD, #01H
        SETB    EN
        LCALL   CHECK


        ; Escrevendo alguma coisa
WRITE_STR:
        MOV     R0, #11D
        MOV     DPTR, #0500H
WRITE_NOW:
        CLR     A
        MOVC    A, @A + DPTR
        LCALL   WRITE_LCD
        INC     DPTR
        DJNZ    R0, WRITE_NOW


        ; Fica num loop infinito
        LOOP:
                NOP
                SJMP    LOOP

        ; Vai para o fim do programa
        LJMP    FIM


; Funcoes auxiliares
ORG     0200H

; Temporizador de 5 ms
TIMER:
        MOV     TL0, #LOW(TEMPO)                        ; Configura o
timer
        MOV     TH0, #HIGH(TEMPO)
        SETB    TR0
        JNB     TF0, $
        CLR     TR0
        CLR     TF0
        RET

CHECK:
        ; Verificando se o LCD está disponível
        CLR     EN
        CLR     RS
        SETB    RW
        MOV     LCD, #0FFH
        SETB    EN
        ;MOV    A, LCD
        ;JB     ACC.7, CHECK
        JB      D7, CHECK
        CLR     EN
        CLR     RW
        RET

WRITE_LCD:
        CLR     EN
        CLR     RW
```

PDF-XChange
Click to buy NOW!
www.docu-track.com

```
        SETB    RS
        MOV     LCD, A
        SETB    EN
        LCALL   CHECK
        RET


; Dados internos do programa
INTERNAL_DATA:
        ORG     0500H
        DB      'HELLO WORLD'


; Fim do programa
ORG     1FFEH
FIM:
        END
```