LCD COM MPLABX C18

O LCD, ou seja, display de cristal líquido, é um dos periféricos mais utilizados como dispositivo de saída em sistemas eletrônicos. Ele contém um microprocessador de controle, uma RAM interna que mantém *escritos* no display (DDRAM) os dados enviados pelo microcontrolador e uma RAM de construção de caracteres especiais (CGRAM). Os LCDs são encontrados nas configurações previstas na Tabela abaixo.

Número de Colunas	Número de Linhas	Quantidade de pinos
8	2	14
12	2	14/15
16	1	14/16
16	**2**	**14/16**
16	**4**	**14/16**
20	1	14/16
20	2	14/16
20	4	14/16
24	2	14/16
24	4	14/16
40	2	16
40	4	16

Os displays mais comuns apresentam 16 colunas e duas linhas. Eles têm normalmente 14 pinos ou 16 pinos. Destes, oito pinos são destinados para dados ou instrução, seis são para controle e alimentação do periférico e dois para *backlight*. O *LED backlight* (iluminação de fundo) serve para facilitar as leituras durante a noite. Neste caso, a alimentação deste led faz-se normalmente pelos pinos 15 e 16, sendo o pino 15 para ligação ao anodo e o pino 16 para o catodo. A ferramenta SanUSB tem uma biblioteca em C para MPLABX C18 que suporta LCDs **16x2** e **16x4** e utiliza **somente o *nibble* superior do barramento de dados (D7, D6, D5 e D4)**, como é o caso da biblioteca LCD.h com a seguinte configuração:

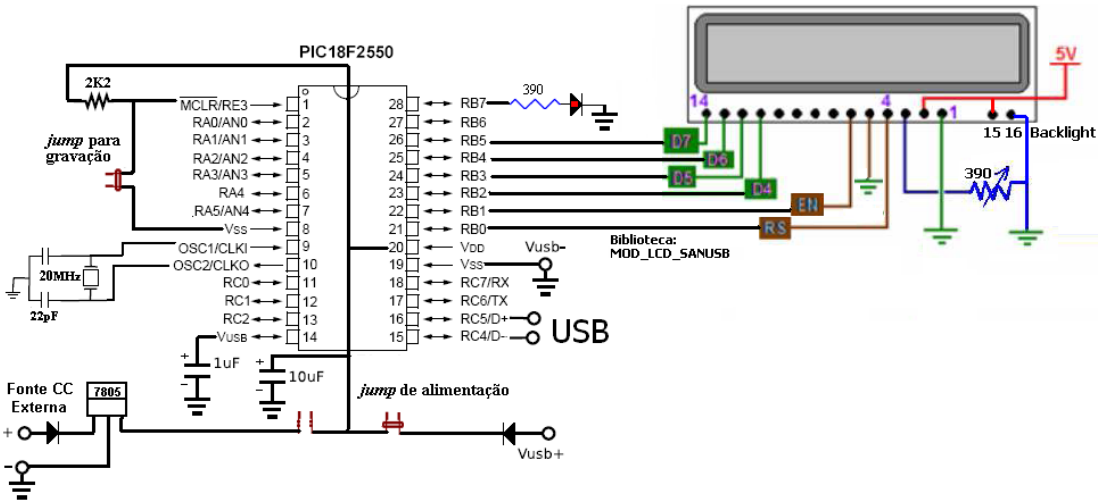

Figura 9. 1: Conexão do LCD no PIC.

A Tabela abaixo traz um resumo das instruções mais usadas na comunicação com os módulos LCD.

Tabela - Instruções mais comuns

DESCRIÇÃO	**MODO**	**RS**	**R/W**	**Código (Hex)**
Display	Liga (sem cursor)	0	0	**0C**
	Desliga	0	0	**0A/ 08**
Limpa Display com Home cursor		0	0	**01**
Controle do Cursor	Liga	0	0	**0E**
	Desliga	0	0	**0C**
	Desloca para Esquerda	0	0	**10**
	Desloca para Direita	0	0	**14**
	Cursor Home	0	0	**02**
	Cursor Piscante	0	0	**0D**
	Cursor com Alternância	0	0	**0F**
Sentido de deslocamento cursor ao entrar com caractere	Para a esquerda	0	0	**04**
	Para a direita	0	0	**06**
Deslocamento da mensagem ao entrar com caractere	Para a esquerda	0	0	**07**
	Para a direita	0	0	**05**
Deslocamento da mensagem sem entrada de caractere	**Para a esquerda**	0	0	**18**
	Para a direita	0	0	**1C**
End. da primeira posição	primeira linha	0	0	**80**
	segunda linha	0	0	**C0**

Utilizando as instruções do LCD:

Para rolar o conteúdo do LCD um caractere para a direita, utilize o comando **lcd_comando(instrução),** por exemplo, **lcd_comando (0x1C**) e para rolar o conteúdo do LCD um caractere para a esquerda, utilize o comando **lcd_comando (0x18).** Abaixo algumas instruções de comando da biblioteca LCD.h e o respectivo valor em decimal da instrução configurada em **lcd_comando()**:

LCD_FIRST_ROW	128
LCD_SECOND_ROW	192
LCD_THIRD_ROW	148
LCD_FOURTH_ROW	212
LCD_CLEAR	1
LCD_RETURN_HOME	12
LCD_UNDERLINE_ON	14
LCD_MOVE_CURSOR_LEFT	16
LCD_MOVE_CURSOR_RIGHT	20
LCD_TURN_OFF	0
LCD_TURN_ON	8
LCD_BLINK_CURSOR_ON	15
LCD_SHIFT_LEFT	24
LCD_SHIFT_RIGHT	28

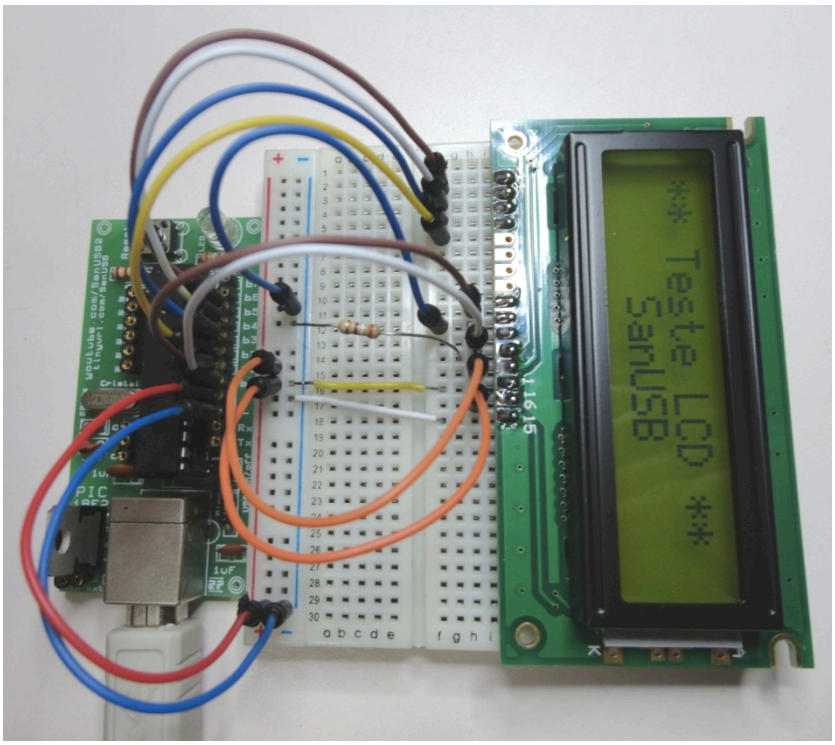

Figura 5. 1: Prática 6 – Display LCD, montada em protoboard.

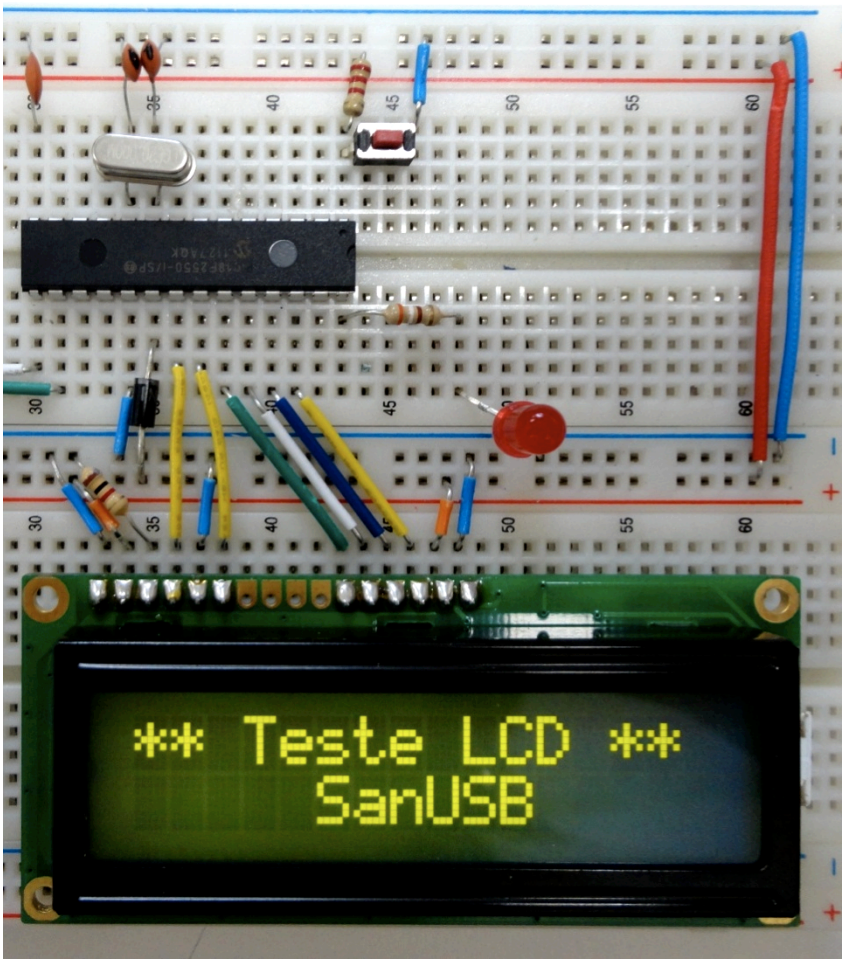

Figura 5. 2: Prática 6 – Display LCD, montada em protoboard.

Exemplo de uso do recurso de rolagem do display.

A seguinte seqüência de comandos, gera o efeito de uma mensagem rolando no display. Para isso, será necessário declarar uma variável do tipo INT x.

Código em C para MPLABX: Ler conversor AD e escrever em LCD:

```
#include "SanUSB1.h"
#include "lcd.h"

unsigned int i;
unsigned char buffer1[20];

#pragma interrupt interrupcao
void interrupcao(){}

void main(void) {
clock_int_4MHz();
habilita_canal_AD(AN0);

lcd_ini();
lcd_comando(LCD_CLEAR);
lcd_comando(LCD_CURSOR_OFF);
tempo_ms(100);

lcd_escreve(1, 1, " **  Teste LCD  **");
tempo_ms(600);

lcd_escreve(2, 1, "SanUSB");
tempo_ms(500);

   while(1)
   {
   i= le_AD10bits(0);

   sprintf(buffer1,"%d  ",i); //Imprime valor do potenciômetro de 0 a 1023
   lcd_escreve2(2, 12, buffer1); //com buffer
   tempo_ms(100);
}
}
```

A posicição o cursor no LCD é configurada dentro da função **lcd_escreve(x, y, "nome");** , onde x e y são, respectivamente, a linha e a coluna onde o cursor deve ser reposicionado.

Desta forma, caso deseje escrever, por exemplo, a frase **Teste LCD** na primeira linha do display, sem apagar a segunda linha, basta inserir o comando **lcd_escreve(1, 1, " Teste LCD"); .** Isto irá posicionar o cursor na primeira linha, e primeira coluna.

STRING : É o trecho de caracteres delimitado por aspas duplas, que irá definir como será a seqüência de caracteres a ser gerada. Dentro das aspas, podem ser inseridos caracteres de texto, caracteres especiais e especificadores de formato.

No caso dos **caracteres especiais**, por não possuírem uma representação impressa, são compostos por uma barra invertida seguida de um símbolo, geralmente uma letra.

Exemplo de caracteres especiais : **\f** (limpar display), **\n** (nova linha), **\b** (voltar um caractere), **\r** (retorno de carro), **\g** (beep), etc...

Obs: alguns caracteres especiais somente resultarão efeito em terminais seriais.

Já os **especificadores de formato** são os locais, em meio ao texto, onde serão inseridas as variáveis que aparecerão após a STRING. Desta forma, estes especificadores devem obedecer algumas regras, de acordo com o tipo da variável a ser impressa.

Observe a seguinte tabela :

Tipo de variável	**Especificador de formato e exemplos de uso**
int	**%u** à valor decimal (ex: 30) **%x** à valor em hexadecimal (ex: 1D) **%3u** à valor decimal alinhado com três dígitos (ex: _30) **%03u** à valor decimal alinhado 3 dígitos c/ zero (ex: 030)
signed int	**%i** à valor decimal com sinal. (ex: -2) **%02i** à decimal com sinal, 2 casas e zeros a esq. (ex: -02)
long int32	**%lu** à valor decimal (ex: 32345675); **%05lu** à valor decimal 5 casas c/ zeros a esquerda. (ex: 01000)
signed long signed int32	**%li** à valor decimal c/ sinal (ex: -500) **%4li** à valor decimal c/ sinal alinhado a esquerda (ex: -_500)
float	**%f** à valor real. Ex: (23.313451) **%2.3f** à valor real c/ 2 casas inteiras, 3 decimais. Ex: (23.313)
char	**%c** à caractere. Ex: (A)