Vitor Amadeu Souza
vitor@cerne-tec.com.br

Programação em BASIC para o PIC

Introdução

Os microcontroladores da família PIC estão a cada dia mais presentes nos projetos eletrônicos. Em média, cada um de nós têm acesso a 9 chips desses todos os dias e aprender a trabalhar e programar estes *computadores em um só chip* se torna praticamente imperativo para o profissional do campo tecnológico. Aplicações como robótica, domótica, automação, entretenimento e etc são alguns dos campos de utilização destes chips.

Existem diversas linguagens no qual o PIC pode ser programado. Dentro deste leque, irei abordar nesta série de artigos a linguagem BASIC com base no compilador mikroBASIC desenvolvido pela mikroElektronika. Este artigo será baseado no livro recém escrito de minha própria autoria, chamado *Programação em BASIC para o microcontrolador PIC18F1220 (Editora Érica 2006)* que pode ser verificado na figura 1.

Figura 1 – Livro Programação em BASIC para o microcontrolador PIC18F1220

Este livro pode ser adquirido nas melhores livrarias técnicas ou através do site www.cerne-tec.com.br . Recomendo a todos que querem ter um conhecimento mais aprofundado da linguagem e do PIC que façam a leitura deste livro. Informo aos leitores desta revista que acompanharão esta série que baixem o compilador no site do fabricante, no endereço *www.mikroe.com*. A versão disponível para download é chamada de versão DEMO. Nesta versão, o código máximo que o

compilador irá gerar é de 2kW e em todos os projetos desta revista, em nenhum momento este limite será ultrapassado.

O kit de desenvolvimento utilizado para testes é placa PICLAB18F1220 que utiliza o microcontrolador PIC18F1220 e pode ser verificado na Figura 2. Este kit foi desenvolvida pela Cerne Tecnologia e Treinamento (www.cerne-tec.com.br), empresa no qual eu represento.

Figura 2 – Placa de Desenvolvimento PICLAB18F1220

O PIC

Os microcontroladores da família PIC18 apresentam-se com grandes vantagens frente a família PIC16. Aspectos como memória de programa e dados são apenas alguns dos vários itens que fazem da família PIC18 superior a PIC16. Observe a tabela 1, onde é apresentado um quadro comparativo entre estas duas famílias.

Descrição	PIC16	PIC18
Memória de programa	8kW *	2 MB *
Memória de dados	512 B *	4 kB *
Interrupção	Único vetor sem prioridades	Dois vetores sendo um de alta prioridade e outro de baixa prioridade
USB	Versão 1.1	Versão 2.0
Módulo LVD (Low Voltage Detect)	Não	Sim
Memória de programa linear	Não, é Segmentada	Sim

Timers	3 *	4 *
Processamento	5 MIPS *	10 MIPS *

* Valores máximos

Tabela 1 – Comparativo entre a família PIC16 e PIC18

Note neste pequeno quadro que a família PIC18 é bem superior a família PIC16. Outro detalhe importante é que mesmo sendo melhor, o custo de ambas as famílias está muito próxima.

O microcontrolador utilizado neste artigo para demonstrar os aspectos da família PIC será o PIC18F1220 onde a pinagem deste componente pode ser observada na figura 2.

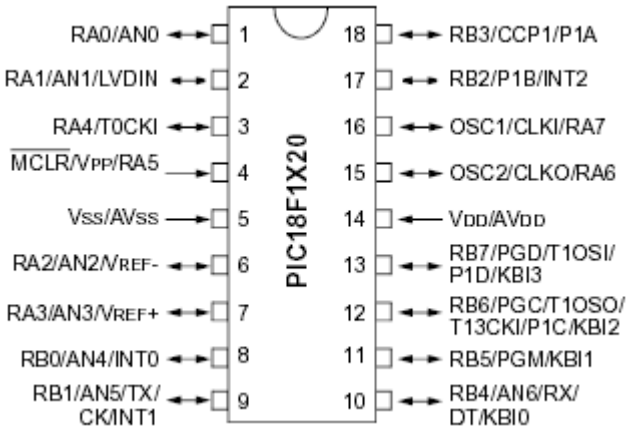


Figura 2 – Microcontrolador PIC18F1220

Observe que como nas famílias tradicionais de 18 pinos, este microcontrolador possui dois PORTS denominados PORTA e PORTB. O PORTA está ligado aos pinos 1 (RA0), 2 (RA1), 3 (RA4), 4 (RA5), 6 (RA2), 7 (RA3), 15 (RA6) e 16 (RA7). Já os pinos do PORTB estão ligados aos pinos 8 (RB0), 9 (RB1), 17 (RB2), 18 (RB3), 10 (RB4), 11 (RB5), 12 (RB6) e 13 (RB7). Este microcontrolador funciona com uma tensão de 2 V até 5,5 V. Em todos os exemplos a alimentação será de 5 V.

Vejamos algumas características importantes deste microcontrolador na tabela 2.

Característica
Memória de Programa Flash de 2 kW
Memória de dados RAM de 256 bytes
Memória de dados EEPROM de 256 bytes
16 pinos de I/O
Fonte de clock interna de até 8 MHz
Módulo LVD
15 Fontes de interrupção
USART
Módulo CCP

Tabela 2 – Características do PIC18F1220

Este microcontrolador está instalado na placa didática PICLAB18F1220 e nesta placa poderemos testar praticamente todos os seus recursos.

O mikroBASIC

Para que possamos criar qualquer programa neste ambiente, é necessário primeiramente criar um projeto. Para isso, após o download e instalação do mikroBASIC, inicialize o mesmo e vá no menu *Project -> New Project*. A tela apresentada na figura 3 surgirá.

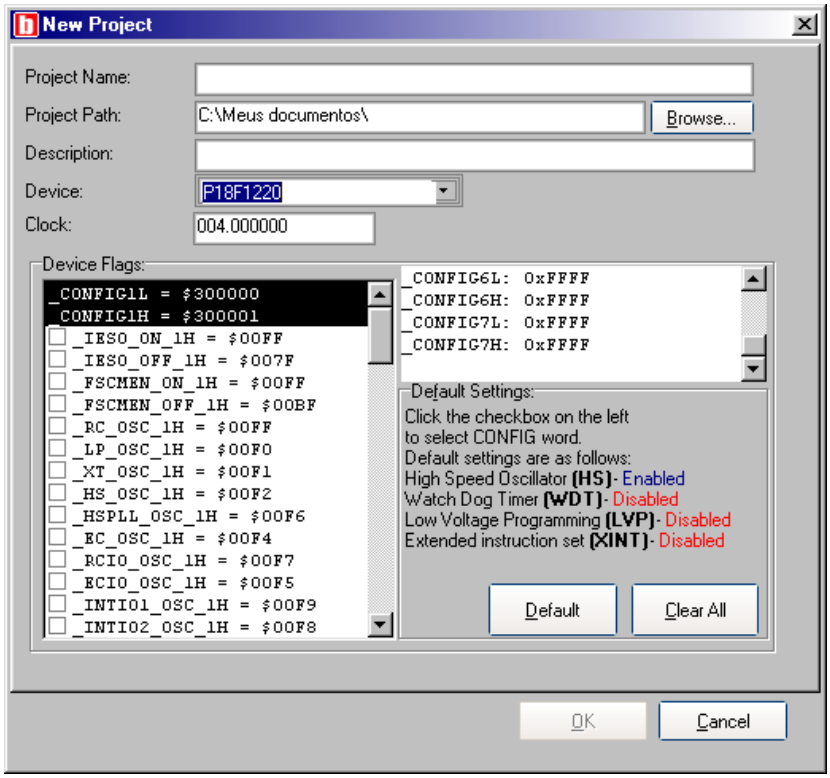


Figura 3 – Criando um novo projeto no mikroBASIC

No campo *Project Name* é informado o nome do seu projeto. Neste campo, digite por exemplo *experimento1.* No campo *Project Path* definimos a pasta onde os arquivos referentes ao projeto ficarão salvos. Escolha neste caso, a pasta de sua preferência. O campo *description* é opcional e serve para fazermos alguma descrição referente ao projeto, sobre o que o mesmo faz por exemplo. Em *device* é escolhido o microcontrolador utilizado no projeto. Clique neste campo e note quantos microcontroladores o mikroBASIC suporta em sua lista. Como estamos estudando o PIC18F1220 escolha neste caso este microcontrolador. Em *device*

flags são ajustados os configurations bits do microcontrolador. Não vamos nos preocupar sobre estes agora, pressione o botão *default* desta janela para que o mesmo seja configurado desta forma. Agora podemos clicar sobre o botão *ok* e prosseguir com o projeto. Assim que você pressionar o *ok*, a tela da figura 4 surgirá.

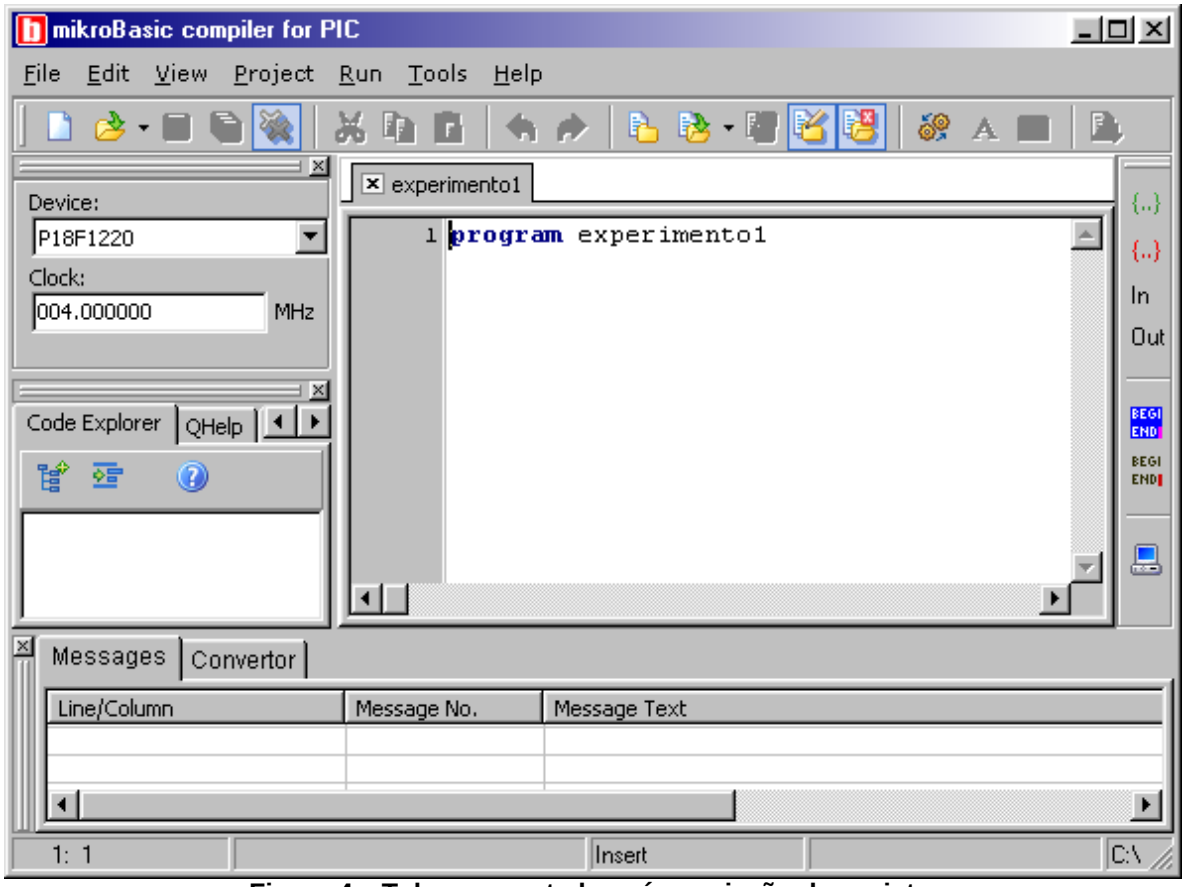


Figura 4 – Tela apresentada após a criação do projeto

Note que a parte branca que está apresentada é onde você deverá criar o seu programa. Um programa em BASIC segue a forma apresentada na tabela 3.

program *nome_do_projeto* *definição de entradas* *definição de saídas* *definição de variáveis* main: end.

Tabela 3 – Organização de um programa em BASIC

Em *definições de entrada* e *definições de saída* são informados quais pinos do microcontrolador são utilizados para funções de I/O (entrada e saída). Já em *definições de variáveis* podemos declarar as variáveis que venham a ser utilizados pelo programa. A partir do *main:* começamos a escrever o programa que irá "rodar" no microcontrolador. Finalmente, o programa é finalizado através do *end.*

Após desenvolver todo o programa da forma apresentada, será necessário *compilar* o seu projeto. Compilar significa traduzir o código que está em BASIC para um código que o microcontrolador consiga entender. Para iniciar a compilação, pressione junto CTRL+ F9 ou vá no menu *Project -> Build.* Neste momento inicia-se a compilação e caso haja algum erro, o mikroBASIC solicitará o ajuste do mesmo para iniciar uma nova etapa de compilação.

Primeiro exemplo prático no mikroBASIC

Muito bem, neste ponto já temos subsídios suficientes para desenvolver o nosso primeiro projeto no PIC. Para isso, a placa didática PICLAB18F1220 deve estar conectada ao gravador GPPIC PRO (Desenvolvido pela Cerne Tecnologia) e este deve estar conectado a porta paralela do PC. Este primeiro exemplo consistirá em ler o estado de um botão e dependendo do seu estado, acionar ou não um led conectado a uma saída do microcontrolador. O esquema elétrico deste projeto pode ser verificado na figura 5.

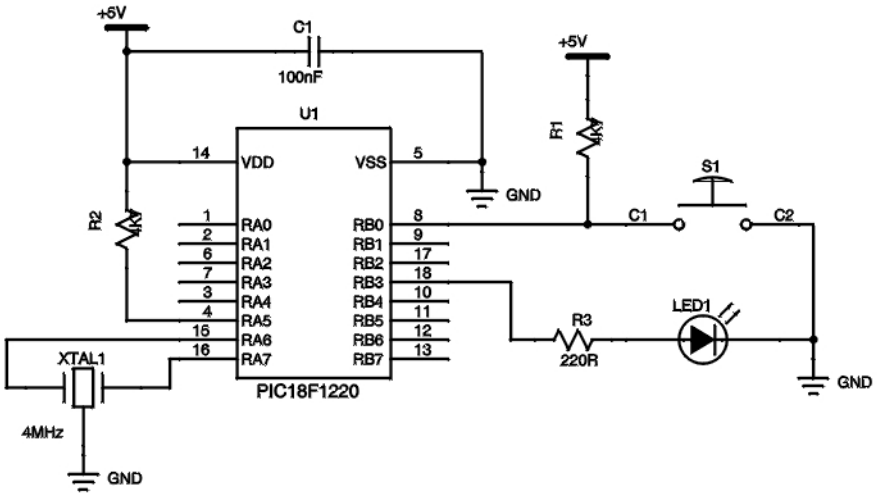


Figura 5 – Esquema Elétrico do exemplo

Observe que o microcontrolador é alimentado em 5 V através dos pinos 5 e 14. O botão está conectado em lógica negativa ao pino RB0. Isto quer dizer que quando o botão está solto, o nível presente na entrada RB0 é 1 e quando o mesmo fica pressionado, este nível vai a 0. O led está conectado ao pino RB3 e este já é acionado em lógica positiva. O ressonador conectado aos pinos RA6 e RA7 geram a freqüência para funcionamento interno do microcontrolador. Note que no pino RA5 existe um resistor ligado a VCC. Este resistor é chamado de pull up e sua função é garantir o nível alto na entrada RA5, evitando desta forma que o microcontrolador venha a resetar. Como nos PCs, o PIC também a sua entrada de reset que fica no pino RA5. Quando este pino é levado a nível lógico 0, o microcontrolador é resetado e quando fica em 1 o mesmo opera normalmente. Neste exemplo onde a função de reset não é utilizada, este pino foi mantido em nível alto.

O programa que fará o efeito desejado está apresentado na tabela 4.

```
program experimento1

main:
     trisb.0=1
     trisb.3=0

     if portb.0 = 0 then
        portb.3=1
     else
         portb.3=0
     end if

     goto main

end.
```

Tabela 4 – Programa do exemplo

Agora vamos entender melhor como o programa funciona. A linha *program experimento1* informa o nome do projeto definido. Logo em seguida temos o label *main:* onde a partir deste ponto poderemos criar o programa. O PORTB do microcontrolador possui 8 pinos que são chamados de pinos de I/O (entrada e saída). A definição destes pinos, ou seja se eles serão de entrada e saída é feita no registrador TRISB. O bit 0 do registrador TRISB é referente ao RB0, enquanto o bit 1 ao RB1 e assim sucessivamente. Como no RB0 está ligado um botão que para o sistema é uma entrada, este pino foi configurado como entrada através do comando trisb.0=1. Quando algum destes bits recebe 1, significa que ele está configurado como entrada e quando recebe 0, como saída. Note que na linha abaixo está definido o trisb.3 como saída, pois ele recebe 0. Definido a direção dos pinos, é necessário testar o PORTB para saber o estado atual deles. Isto é feito através da declaração *if*. Observe o trecho *if portb.0 = 0 then*. A tradução literal ficaria *se o portb.0 está em nível baixo então.* Ele testa o nível lógico presente na entrada RB0 e se o botão estiver pressionado, este teste será verdadeiro e o comando associado ao *if* será executado que neste caso é para ligar o led através de *portb.3=1.* Caso o botão não esteja pressionado, o *else (Senão)* será tratado, deixando neste caso o led desligado através do comando *portb.3=0.* Finalmente o *if* é encerrado através do *end if (fim do se).* Este bloco fica se repetindo continuamente, pois logo em seguida observamos o uso do *goto main* que faz com que o programa volte para o label main e fique testando o estado do botão e tomando determinada ação caso o botão esteja pressionado ou não. O programa agora é encerrado através de end.

Após a digitação deste código no mikroBASIC, compile o mesmo e neste ponto poderemos transferir o programa para a placa didática. Existem vários softwares que podem ser utilizados para este fim. Neste exemplo, o software ic-prog foi utilizado. Este software pode ser baixado gratuitamente no site do desenvolvedor, que é www.ic-prog.com. Após o download deste software, inicialize o mesmo. Surgirá uma janela solicitando que você escolha o tipo de gravador utilizado. Deixe esta janela conforme apresentado na figura 6.

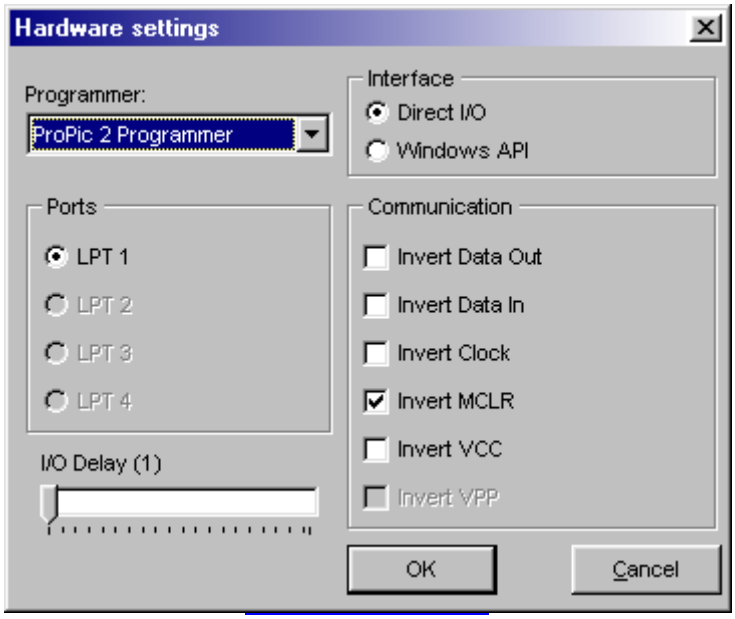


Figura 6 – Configurando o ProPic II

*obs: Caso o seu Windows seja o Xp, 2000 ou NT, será necessário liberar o acesso a porta paralela do seu PC. Vá no site da Cerne Tecnologia (**www.cerne-tec.com.br**) e veja na seção tutoriais o guia Liberando o acesso da porta paralela.*

Após esta configuração, pressione o botão *ok.* Será aberto agora a janela normal do Ic-Prog, como apresentado na figura 7.

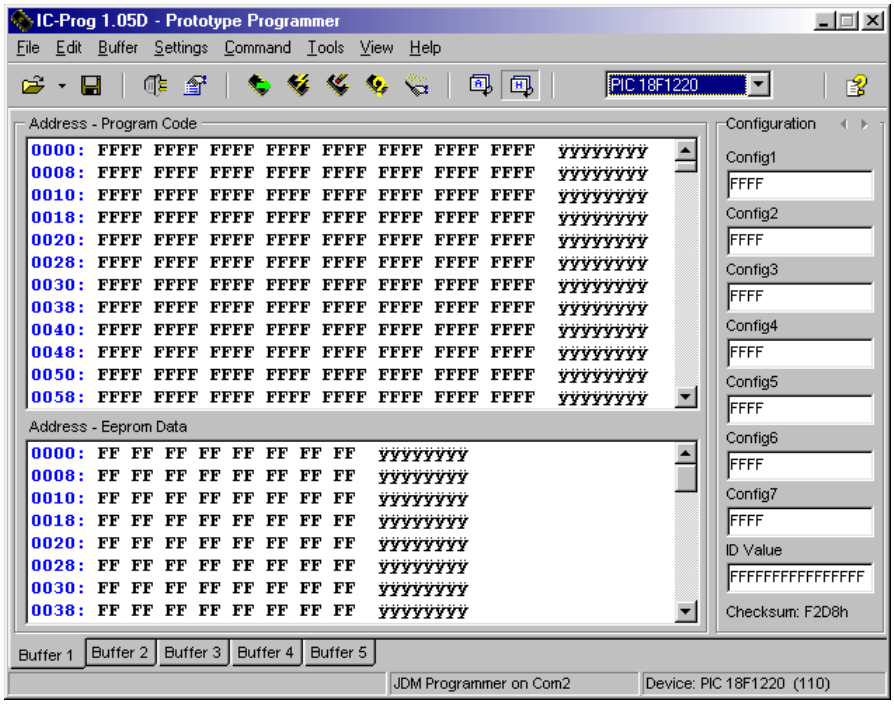


Figura 7 – Janela do Ic-Prog

Será necessário agora escolher o microcontrolador utilizado para gravação, neste caso o PIC18F1220. Para isso vá em *Settings -> Device -> Microchip PIC* e procure na lista apresentada o microcontrolador PIC18F1220.

Podemos também trocar o idioma utilizado pelo Ic-Prog para o Português. Vá em *Settings -> Options* e abra a aba *Language.* Escolha no combo presente neste aba, o idioma *Portuguese* e pressione *ok.*

Agora podemos abrir o arquivo a ser gravado no microcontrolador. Sempre que o mikroBASIC compila um arquivo e não houve nenhum erro de compilação, o compilador gera um arquivo com o mesmo nome do projeto porém com a extensão hex. É este arquivo que deve ser gravado no microcontrolador. Para abri-lo, vá em *Arquivo -> Abrir* e veja na pasta onde você criou o seu projeto o arquivo com a extensão hex. Clique duas vezes sobre o mesmo e pressione a tecla F5. Neste momento uma janela como a apresentada na figura 8 surgirá.

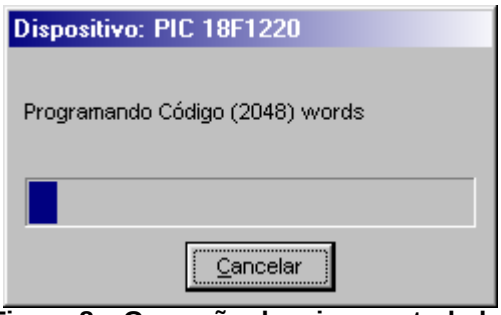


Figura 8 – Gravação do microcontrolador

Após a gravação, teste o seu programa recém gravado na placa didática. Pressione o botão que fica ligado ao RB0 e veja se o led acende de acordo com o pressionar dele.

Exemplo II

Continuando com a série do último artigo, hoje veremos os passos para o desenvolvimento de uma minuteria com o microcontrolador PIC18F1220 e o mikroBASIC. Para isso, aprenderemos a utilizar as funções de retardo disponíveis na linguagem.

A função de uma minuteria é manter ligada uma saída durante um intervalo de tempo. Nos dias de hoje em que a demanda por energia elétrica é alta, ter formas de economizá-la é uma das aplicações em que um microcontrolador pode ser utilizado. Normalmente a minuteria é utilizada em corredores ou garagens por exemplo e que a ao se detectar a presença de uma pessoa, a iluminação é acionada por um período de tempo e logo em seguida ela é desligada automaticamente. Os sensores que informam ao circuito de controle a presença de uma pessoa são chamados de sensores de infravermelho PIR e estes podem ser observados na figura 1.

Figura 1 – Sensor de Infravermelho

Internamente destes sensores, já fica instalado o circuito de minuteria e o tempo em que o mesmo ficará acionado após receber o sinal de presença é ajustado através de um trimpot. Apesar deste projeto poder ser desenvolvido por um microcontrolador, nada impede também que utilizemos lógica discreta para o desenvolvimento deste circuito. Esta é mais uma aplicação onde o microcontrolador pode ser utilizado.

Recursos de Hardware

Para o desenvolvimento deste projeto, utilizaremos a placa didática PICLAB18F1220 desenvolvido pela Cerne Tecnologia (**www.cerne-tec.com.br**). Para simular o sensor de infravermelho, utilizaremos um botão e para simular a saída de um relé o led que fica instalado na própria placa. O circuito elétrico pode ser observado na figura 2.

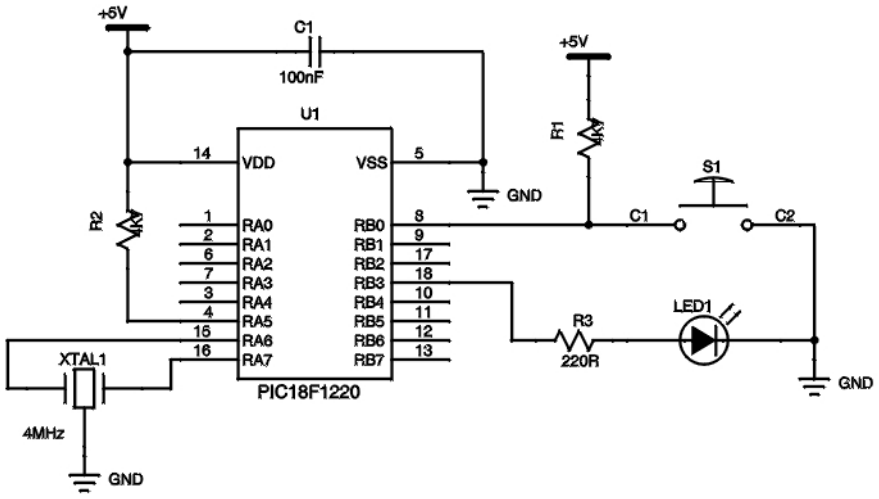


Figura 2 – Circuito Elétrico

Carta de Tempos

O exemplo funcionará da seguinte forma: Quando for detectado a presença de uma pessoa (ao pressionar o botão, no nosso caso) a saída ficará acionada durante o intervalo de 10 s. Este tempo pode ser alterado através de parâmetros de software como será visto em breve. Desta forma, a carta de tempo do nosso exemplo será como a apresentada na figura 3.

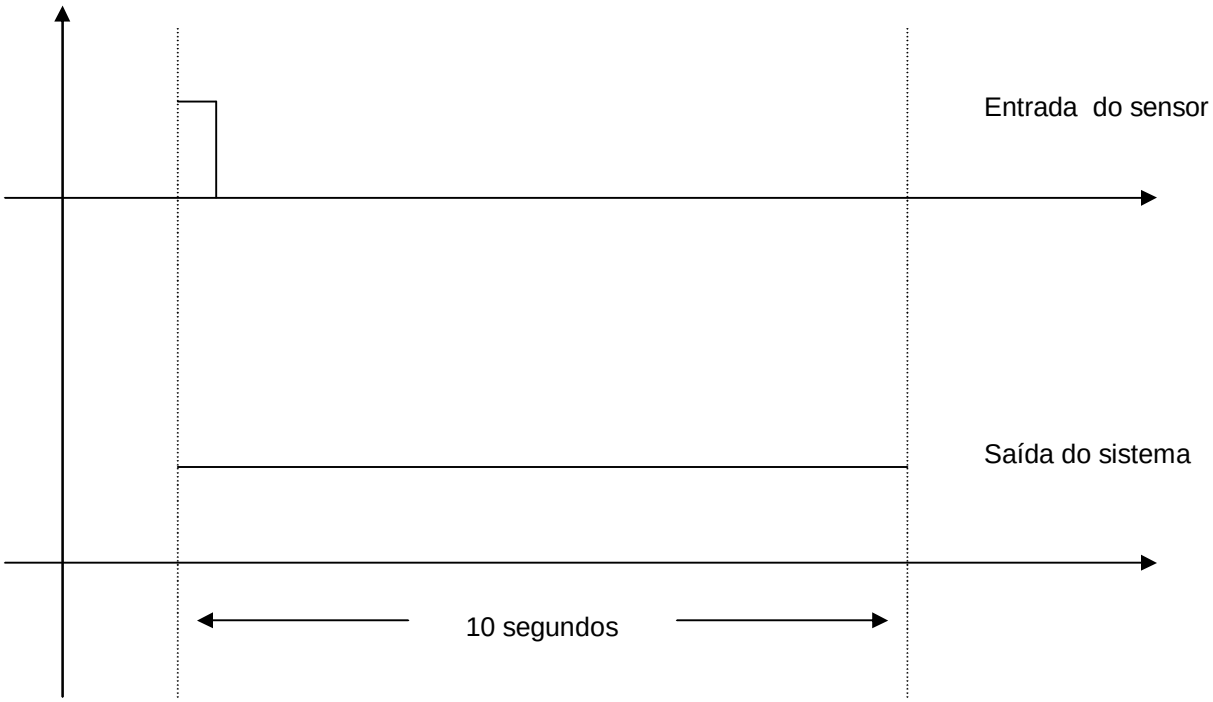
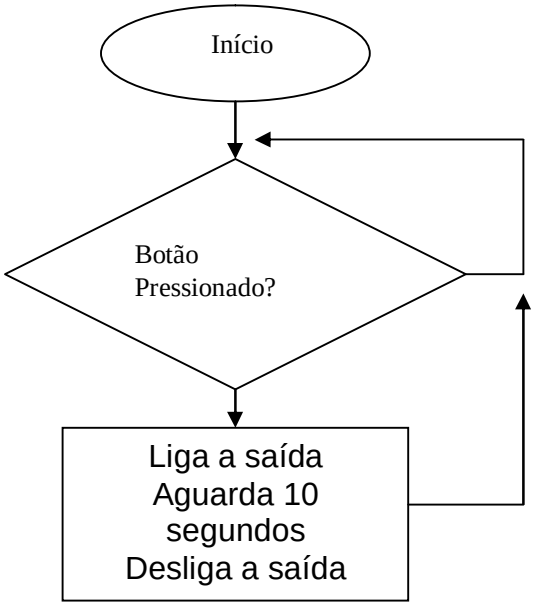


Figura 3 – Carta de Tempos

Fluxograma

O fluxograma que irá reger o funcionamento deste software pode ser apreciado na figura 4. Note que enquanto não é verificado a presença de nenhuma pessoa, o sistema fica preso em loop, testando ciclicamente a entrada do sensor. No entanto assim que este estado se altera, a saída é ligada e logo em seguida entra em uma rotina que faz com fique durante o intervalo de 10 s neste estado. Logo em seguida a mesma é desligada e volta-se a verificar a entrada do sensor.



Recursos de Software

Para execução deste exemplo, precisaremos de uma função que permita um retardo no programa. Para isso, utilizaremos a função da linguagem BASIC chamada *delay_ms(parâmetro).* Esta função faz com que o programa fique parado durante um intervalo de tempo em função de *parâmetro.* Desta forma, se chamarmos *delay_ms(1000)* o programa ficará durante o intervalo de 1 segundo "preso" nesta rotina causando desta forma o retardo necessário que precisamos. É importante salientar que o *parâmetro* máximo que pode ser passado para esta função é de 65535.

Software

O software final, que permitirá que construamos a nossa minuteria pode ser observado no box1. Observe que o leitor deve criar antes um projeto, de acordo com o primeiro artigo desta série e desta forma digitar o código abaixo.

```
program Minuteria

main:
  adcon1=%01111111      'Configura os pinos para funcionar em modo digital
  trisb=%00000001       'Configura o RB0 como entrada e restante como saída

      if portb.0 = 0 then   'Se o botão estiver pressionado...
            portb.3=1       'Liga a saída
            delay_ms(10000) 'Aguarda 10 segundos
            portb.3=0       'Desliga a saída
      end if

      goto main             'Salta para main

end.
```

Box 1 – Programa do Exemplo

Vamos agora analisar melhor o código exemplo. Primeiramente, o nome do programa é informado através de *program Minuteria.* Note que neste caso o projeto foi chamado de minuteria. Logo em seguida o programa começa através do label *main.* Na próxima linha encontramos a declaração *adcon1=%01111111.* Isto foi feito pois os pinos RA0, RA1, RA2, RA3, RB0, RB1e RB4 ficam configurados automaticamente como pinos de função analógica e como no exemplo proposto esta função não é utilizada, foi necessário desligá-la através deste comando. Mais detalhes sobre os pinos de função analógica serão explorados nos próximos artigos. O comando *trisb=%00000001* faz com que o pino RB0 fique configurado como entrada e o restante como saída (observe o esquema elétrico na figura 2). Após estas configurações, inicia-se o teste do pino RB0 e caso o mesmo fique ativo (neste caso em nível lógico 0) o led conectado na saída RB3 será acionado e ficará neste estado durante o intervalo de 10 s através da função *delay_ms(10000)*. Passado este intervalo, o led é desligado e o sistema volta a testar o sensor (botão) afim de detectar algum evento e acionar novamente a saída. Observe que ao lado de alguns comandos do programa, existe um texto explicativo ao lado que inicia-se por ‘ (apóstrofo). Todo o texto que é iniciado por este caracter é chamado de comentário e o compilador no momento da compilação não trata este texto, tornando o seu uso livre.

Após o desenvolvimento deste programa, compile o mesmo e transfira o arquivo hex para a placa didática afim de validar o exercício.

Exemplo III

Nesta terceira série de artigos que aborda a programação na linguagem BASIC para o microcontrolador PIC, iremos utilizar outro microcontrolador mais poderoso da família PIC18, que é o PIC18F442. A intenção deste projeto é poder mostrar uma mensagem em um display do tipo LCD. Estes displays são hoje largamente utilizados em nosso dia-a-dia, pois permitem mostrar o estado de um

dispositivo qualquer e além disso consomem pouca energia e utilizam um pequeno espaço para isso, comparado com outros periféricos. Estes displays são comumente chamados de interfaces *IHM (Interface Homem-Máquina)* pois permitem que o usuário de um sistema por exemplo saiba o funcionamento atual de alguma máquina através dele. Na figura 1 está apresentado o display utilizado no projeto, que é chamado de alfanumérico (pois mostra caracteres e números).

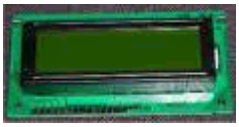


Figura 1 – Display LCD alfanumérico

Apesar do display do tipo alfanumérico também suporta o modo gráfico, sua maior utilização é para o modo alfanumérico. Outro display também muito utilizado é o chamado de display gráfico que pode ser observado na figura 2.

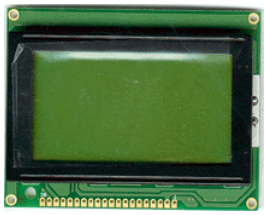


Figura 2 – Display LCD Gráfico

A pinagem do display lcd alfanumérico obedece normalmente a configuração da figura 3.

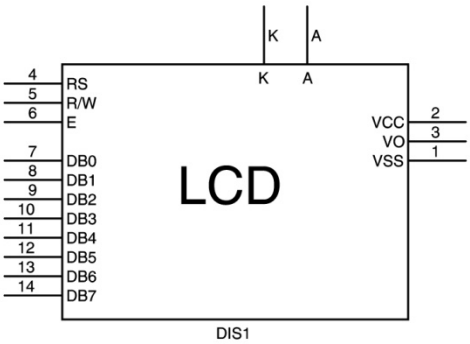


Figura 3 – Pinagem do display LCD 16x2

Note que existem oito pinos que vão do DB0 até o DB7. Estes pinos são os pinos de dados do LCD. É através deles que o microcontrolador irá comunicar com este dispositivo e mostrar uma mensagem em uma dada posição. A função dos pinos RS, R/W e E são controlar a comunicação que é feita através do barramento de dados entre o microcontrolador e o display. Os pinos VCC e VSS são utilizados para alimentar o display. A alimentação típica é de 5 V. Através do pino VO podemos controlar o contraste do display e finalmente através dos pinos K e A ligar ou não o backlight (luz de fundo) do display.

Para esta experiência, faremos uso da placa didática PICLAB18F442 desenvolvida pela Cerne Tecnologia ***(www.cerne-tec.com.br)***. Esta placa pode ser observada na figura 4.

Figura 4 – Placa PICLAB18F442 desenvolvida pela Cerne Tecnologia

Alguns recursos que esta placa disponibiliza para o desenvolvedor podem ser verificados na tabela 1.

LCD alfanumérico
Displays de leds de 7 segmentos
Varredura de leds
Memória serial EEPROM 24C04 (protocolo I²C)
Varredura de Teclas
Acionamento de cargas externas
Gravação in-circuit
Botão de reset manual
Tacômetro
Ventilador

Sensor de temperatura
Aquecedor
Conversão A/D
Comunicação serial RS232

Tabela 1 – Características da Placa PICLAB18F442

Vamos utilizar o recurso Display LCD disponibilizado por esta placa para realizarmos nossa experiência. Os leitores interessados em adquirir esta placa podem entrar em contato pelo telefone (21)3062-1711 ou através de nossa página na internet, que é **www.cerne-tec.com.br**.

O microcontrolador PIC18F442, que será o tema deste artigo possui as características apresentadas na tabela 2.

16kB de memória de programa
768 Bytes de memória de dados
256 Bytes de memória EEPROM
Processamento de até 10 MIPS
Conversor AD
USART
Comunicação I^2C
Módulo CCP
4 Timers internos
Watchdog programável

T abela 2 – Características do microcontrolador PIC18F442

A pinagem deste microcontrolador pode ser observada na figura 5.

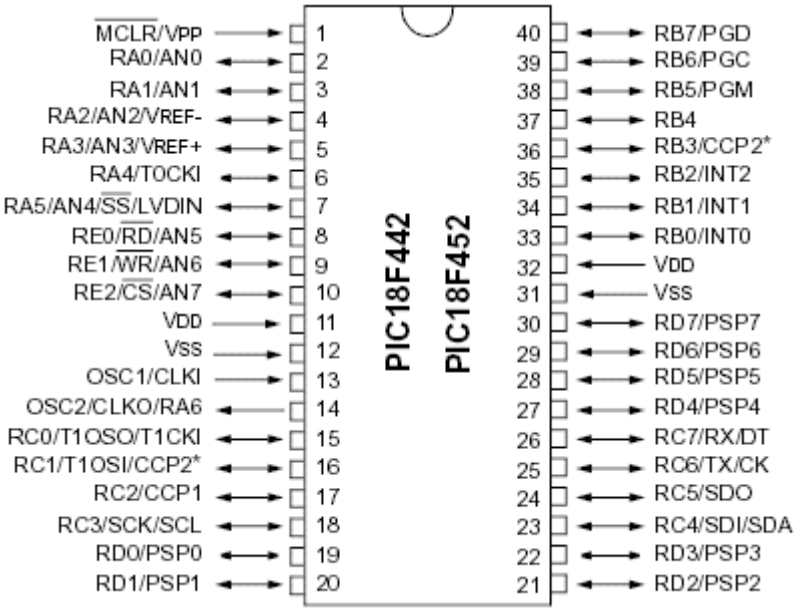


Figura 5 – Pinagem do microcontrolador PIC18F442

Recursos de Hardware

O esquema elétrico para execução deste exemplo está apresentado na figura 6. Observe que neste exemplo, todo o PORTD do microcontrolador é utilizado como via de dados para o display LCD e os três pinos do PORTE são utilizados como pinos de controle. A alimentação tanto do microcontrolador quanto do display é de 5 V. A fonte de clock para o microcontrolador é um ressonador de 4 MHz que fica conectado aos pinos OSC1 e OSC2.

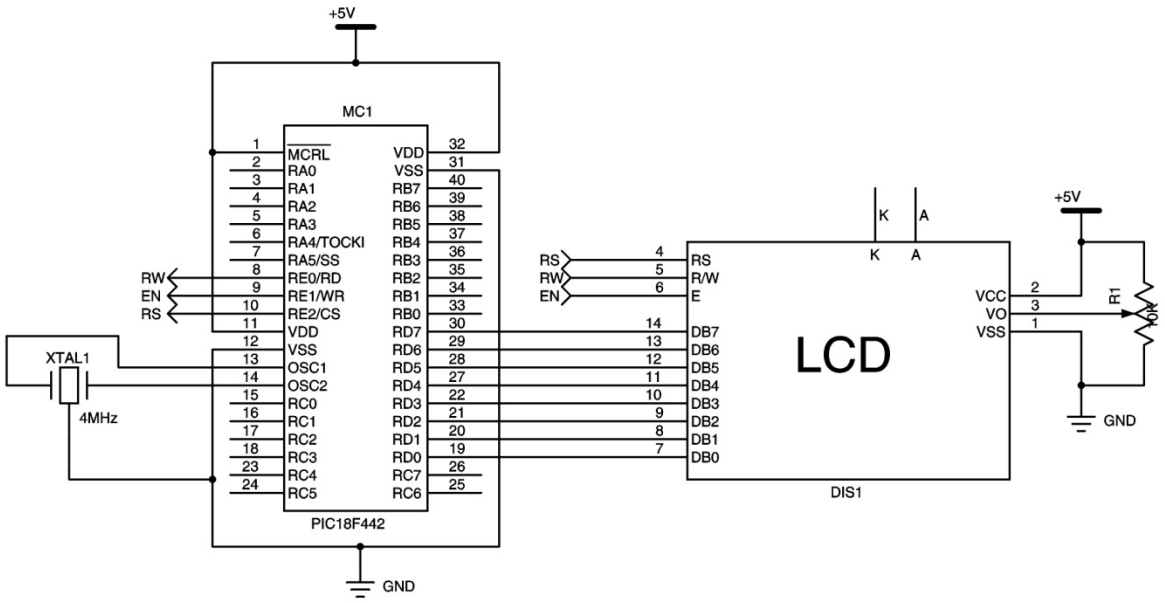


Figura 6 – Esquema Elétrico do exemplo

Fluxograma

O fluxograma de funcionamento deste exemplo pode ser observado na figura 7.

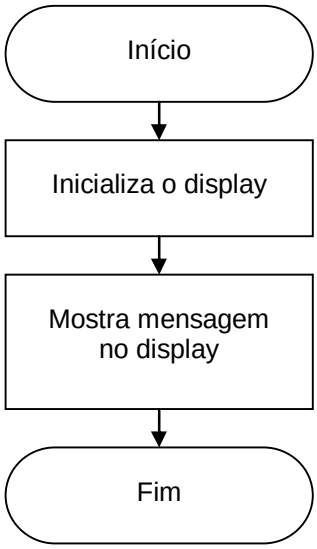


Figura 7 – Fluxograma

Recursos de Software

Para configurarmos e acessarmos o LCD existem três funções básicas que permitem isto no mikroBASIC. Estas funções estão apresentadas na tabela 3.

lcd8_config(porta de controle, porta de dados, RS, EN, RW, D7, D6, D5, D4, D3, D2, D1, D0)
lcd8_init(porta de controle, porta de dados)
lcd8_out(linha, coluna, dados)

Tabela 3 – Funções de acesso ao LCD

A primeira função configura em que pinos estão ligados os pinos de controle e dados do display LCD no microcontrolador. No nosso caso, de acordo com o esquema elétrico apresentado os pinos de controle RS, EN e RW estão ligados no PORTE nos pinos 2, 1 e 0 respectivamente. Já os pinos de dados estão ligados ordenadamente nos pinos do PORTD. Desta forma, esta função ficaria da forma apresentada no box 1.

```
lcd8_config(porte, portd, 2, 1, 0, 7, 6, 5, 4, 3, 2, 1, 0)
```

Box 1 – Configurando os pinos de I/O

A segunda função tem a incumbência de inicializar o LCD de acordo com os pinos recém configurados através da função lcd8_config. Os parâmetros passados para esta função são a porta de controle e dados. Verifique no box2 como ficaria a chamada desta função:

```
lcd8_init(porte, portd)
```

Box 2 – Inicializando o LCD

Após este processo de inicialização do display, finalmente podemos mostrar um dado no mesmo. Para isso, utilizaremos a função lcd8_out que escreve na linha e coluna informadas como parâmetro o dado a ser escrito no lcd. Neste exemplo iremos mostrar a mensagem “Cerne Tecnologia” na primeira linha e “cerne-tec.com.br” na segunda. Observe no box 3 como ficaria a escrita de dados no LCD.

```
lcd8_out(1,1,”Cerne Tecnologia”)
lcd8_out(2,1,”cerne-tec.com.br”)
```

Box 3 – Escrevendo dados no LCD

Software

O software completo que possibilitará apresentar os dados no display está apresentado no box 4.

```
program display
        adcon1=7
        trise=0
        trisd=0
        lcd8_config(porte,portd,2,1,0,7,6,5,4,3,2,1,0)
        lcd8_init(porte,portd)
        lcd8_out(1,1,"Cerne Tecnologia")
        lcd8_out(2,1,"cerne-tec.com.br")
end.
```

Box 4 – Programa de controle do LCD

Não esqueça de criar um projeto novo para o microcontrolador PIC18F442 no mikroBASIC e configure os configurations bits de forma com que o oscilador seja XT e deixe o Watchdog e LVP desativados.

Após compilar este arquivo, transfira o arquivo hex gerado pelo mikroBASIC para a placa Cerne PICLAB18F442 e veja o resultado!

Exemplo IV

Nesta quarta série relatando o uso do mikroBASIC para programar o PIC, veremos como realizar a comunicação serial entre o PC e o microcontrolador. A idéia será ligar ou desligar um relé instalado na placa didática, de acordo com o caracter recebido via comunicação serial. Esta aplicação seria bem interessante para no caso de alguma automação residencial, onde em determinados momentos alguma lâmpada poderia ligar ou desligar de acordo com o horário ajustado no PC. A comunicação utilizada é do tipo RS232, e os níveis lógicos associados ao nível alto e baixo deste tipo de comunicação e do TTL podem ser observados na tabela 1.

Níveis / Tipos	TTL	RS232
1	5 Vcc	-3 a –18V
0	0 Vcc	3 a 18 V

Tabela 1 – Níveis de comunicação

Note que os níveis utilizados pelo PIC são os níveis TTL enquanto o utilizado pelo PC, é o RS232. Para compatibilizar um nível ao outro, é necessário um conversor de níveis de TTL para RS232 e vice-versa. Um conversor muito popular hoje no mercado é o MAX232 desenvolvido pela Maxim. Este conversor pode ser observado na figura 1.

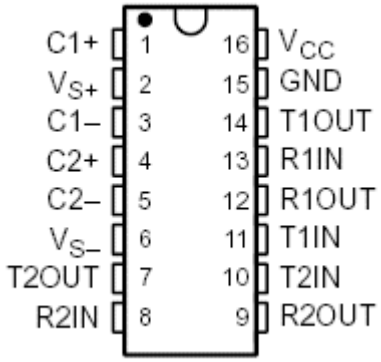


Figura 1 – Conversor TTL – RS232 e vice-versa

O esquema básico de ligação entre o microcontrolador e o PC, de forma com que os mesmos possam se comunicar está apresentado na figura 2.

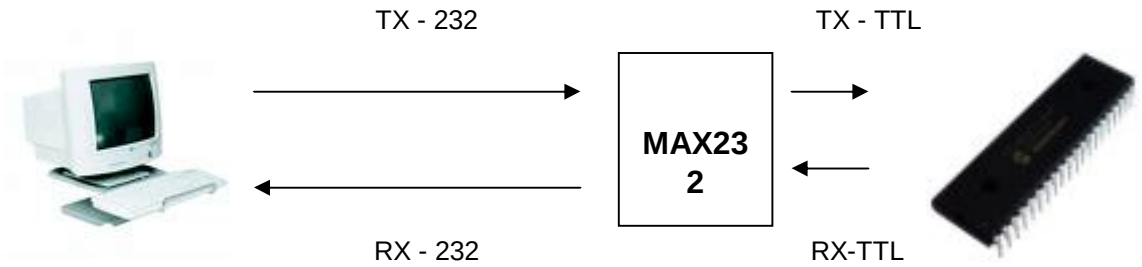


Figura 2 – Esquema de ligação básico para comunicação

Observe que o circuito de conversão, que é o CI MAX232 e o microcontrolador ficam ambos colocados na placa PICLAB18F442.

Para entendermos melhor o funcionamento deste tipo de comunicação, vamos plotar um gráfico que irá apresentar a comunicação no nível TTL e RS232. Verifique a transmissão do byte 01101010 na figura 3.

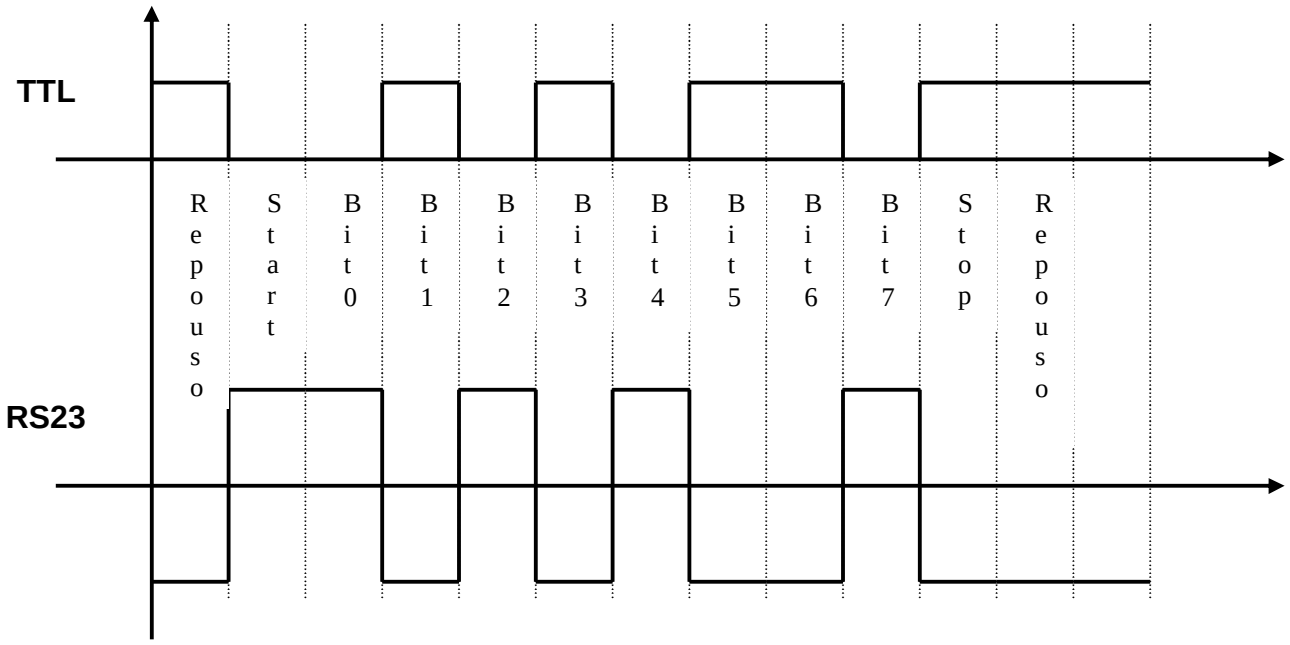


Figura 3 – Plotagem do gráfico na linha TTL e RS232

Observe que no momento em que não há comunicação, a linha TTL permanece em nível alto. Este é o conhecido estado de repouso, ou seja, o momento em que não há comunicação na linha e a mesma está pronta para começar. Toda vez em que um byte será enviado, primeiramente temos o bit de start (início) e logo em seguida, os oito bits do byte a ser transmitido, começando sempre pelo bit menos significativo. Para finalizar a comunicação, temos o bit de stop (parada) em que neste momento, a linha de dados volta ao seu estado de repouso.

Toda comunicação serial, seja ela USB, RS485 ou RS232 utiliza uma taxa de comunicação (baud rate). Com o baud rate, podemos saber quantos bits podem trafegar pela linha em um intervalo de 1 segundo. Comumente os baud rates são múltiplos de 300 bps (bits por segundo), encontrando desta forma taxas de comunicação como por exemplo 2400 bps, 4800 bps e 9600 bps. Em nosso exemplo, iremos utilizar a taxa de 9600 bps e para acharmos o tempo que 1 bit demora para ser transmitido nesta taxa, basta dividirmos 1 pelo número de bits por segundo. No gráfico plotado na figura 3, considerando o baud rate de 9600 bps, achamos o tempo de aproximadamente de 104 us, pois 1/9600 é igual a este valor.

Existem basicamente três modos de comunicação utilizados, sendo estes o simplex, half duplex e full duplex. A comunicação simplex é aquela em que se dá em somente 1 sentido. Um exemplo seria o caso da televisão, em que o usuário somente consegue receber dados oriundos da central de transmissão e nunca consegue fazer o processo inverso. Poderíamos representar a comunicação simplex como uma seta de uma só direção, como apresentado na figura 4.

Figura 4 – Representação da comunicação Simplex

Já a comunicação half duplex se dá nos dois sentidos, porém neste caso a linha de comunicação é compartilhada, sendo em instante somente de transmissão e em outro somente de recepção. Existem aparelhos telefônicos, como o da operadora NEXTEL que é um exemplo deste tipo de comunicação. Neste caso, poderíamos representar este tipo de comunicação como apresentado na figura 5, com uma seta única com comunicação bidirecional.

Figura 5 – Representação da comunicação Half Duplex

Já a comunicação full duplex permite a troca de dados simultaneamente, pois temos duas linhas, sendo uma de transmissão e outra de recepção. Um exemplo corriqueiro seria o uso de um celular, em que duas pessoas conseguem falar no mesmo momento e trocar informações. A representação desta comunicação seria como apresentado na figura 6, com uma seta somente de transmissão e outra somente de recepção.

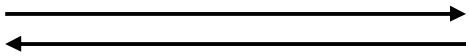


Figura 6 – Representação da comunicação Full Duplex

A comunicação do tipo RS232 é do tipo full duplex, pois temos uma linha somente de transmissão e outra somente de recepção. Desta forma, enquanto estamos transmitindo um byte pela linha de TX, podemos perfeitamente estar recebendo outro pela linha de RX.

Recursos de Hardware

O esquema elétrico deste exemplo pode ser observado na figura 7. Observe que as linhas de comunicação do PIC ficam ligadas ao MAX232 e no pino RB0 está conectado o relé do nosso exemplo. Além disso, o microcontrolador e o MAX232 ficam ambos alimentados através de uma fonte de 5 Vcc.

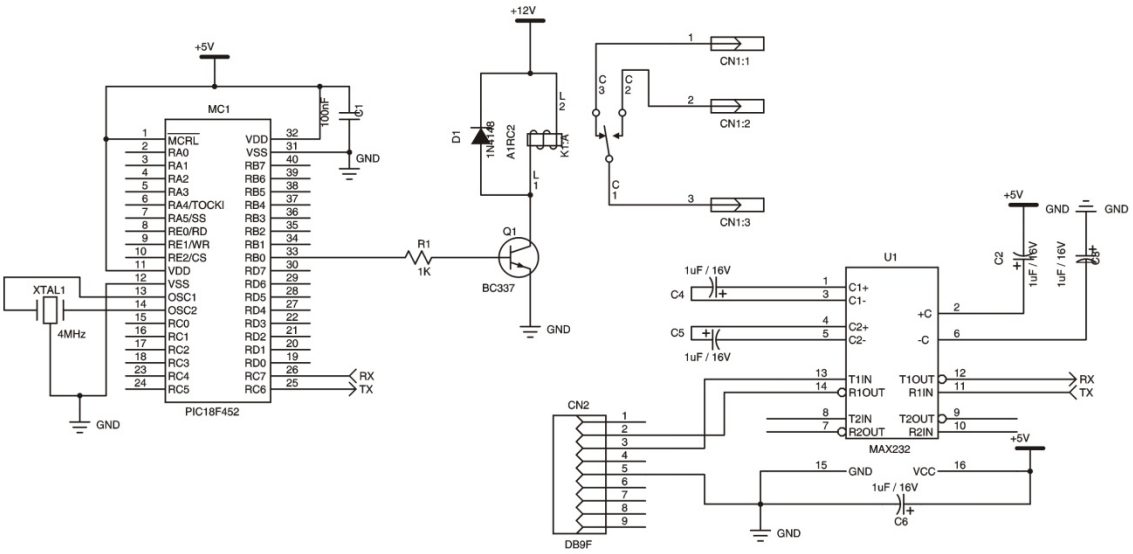


Figura 7 – Esquema Elétrico do Exemplo

Fluxograma

O fluxograma que irá reger este exemplo está apresentado na figura 8. Note que primeiramente é feita a configuração do microcontrolador, como a configuração dos pinos e da USART e logo em seguida o sistema entra em loop infinito verificando a existência de algum byte no canal serial. Ao ser verificado que algum byte foi recebido, é checado se o mesmo é o caracter “A” e caso seja, o relé é acionado. Caso o caracter não seja o “A”, o buffer é novamente testado e caso seja o “B”, o relé é desligado. Quaisquer caracteres diferentes destes dois não terão efeito sobre o sistema.

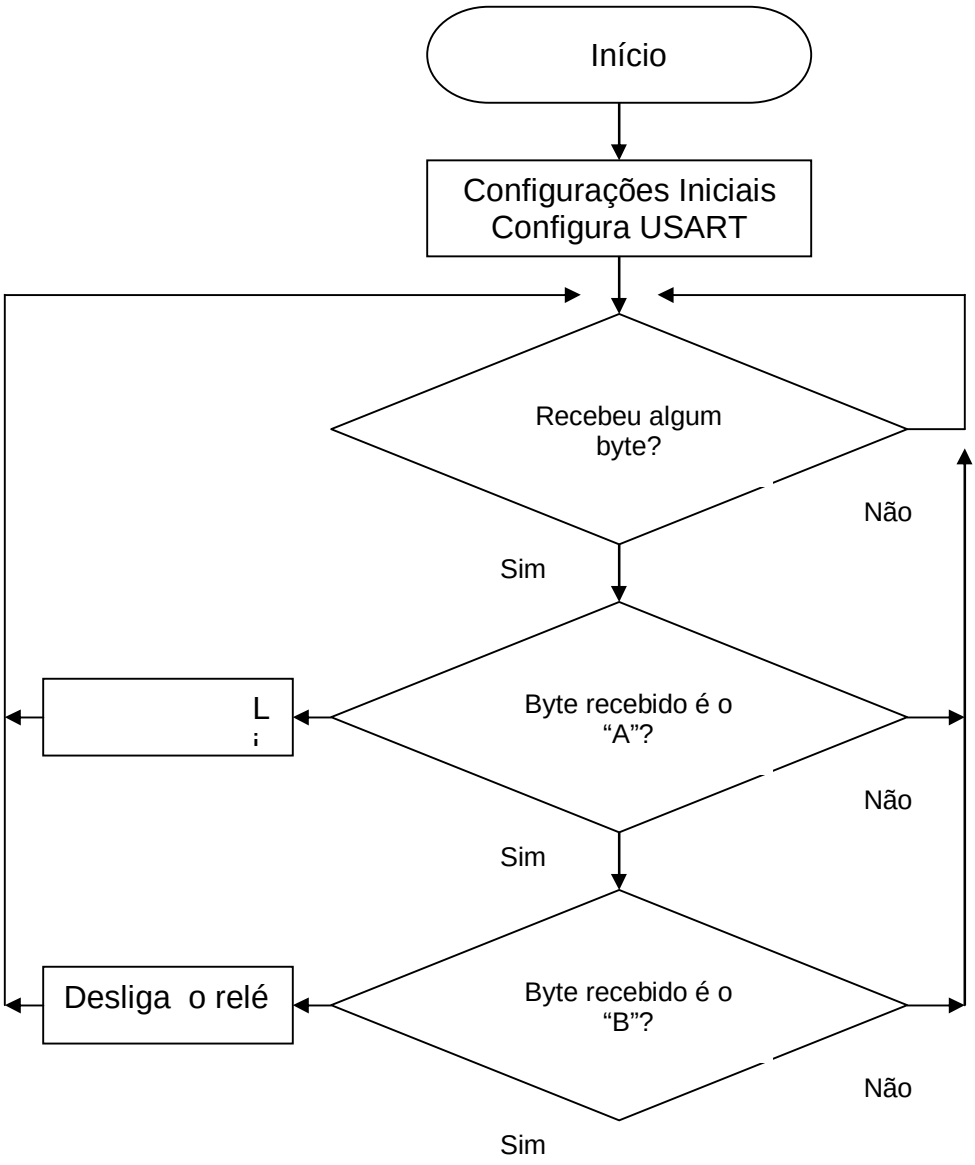


Figura 8 – Fluxograma do exemplo

Recursos de Software

Existem basicamente quatro funções disponíveis para o acesso a porta de comunicação serial do microcontrolador. Estas estão apresentadas na tabela 2.

usart_init (baud rate)
usart_read
usart_data_ready
usart_write_text (texto a escrever)

Tabela 2 – Funções de acesso a comunicação serial

Através da função usart_init podemos inicializar a máquina de comunicação serial com um baud rate definido pelo programa. Como no nosso caso a taxa será de 9600 bps, este comando será definido como usart_init (9600). Utilizando a função usart_read, podemos ler o byte que foi recebido e está armazenado no buffer do microcontrolador. Já a função usart_data_ready permite com que saibamos se existe ou não um byte para ser feita a leitura. Sempre antes de lermos o buffer de comunicação através da função usart_read, precisamos saber se existe algum byte lá e através do teste da usart_data_ready podemos obter esta informação. Com a função usart_write_text podemos enviar um dado para o PC.

Software

O software completo que permitirá observar o funcionamento deste exemplo está apresentado no box 1.

```
program comunicacao
main:

     trisb.0=0
     usart_init(9600)

repete:

       if usart_data_ready=1 then
           if usart_read="A" then
                portb.0=1
           end if
           if usart_read="B" then
                portb.0=0
           end if
       end if
       goto repete

end.
```

Box 1 – Programa do exemplo

Vamos entender melhor o código apresentado. Logo destarte observamos o pino RB0 configurado como saída, através do comando trisb.0=0. Esta configuração é feita pois neste pino está ligado o relé. Logo na linha abaixo a usart é configurada para funcionar a 9600 bps através da função usart_init. No ponto

seguinte, o programa fica preso em loop testando continuamente se o existe ou não um dado no buffer de comunicação. Ao ser detectado a presença de um byte, é verificado se o caracter é o “A” ou o “B” e caso seja algum deles será tomada a ação de acordo com o caracter.

Para que possamos enviar os dados do PC para o microcontrolador, utilizaremos o software chamado Communication Terminal que já vem no próprio mikroBASIC. Para acessar este software, vá em Tools -> Usart Terminal. Será aberta uma tela como apresentado na figura 9.

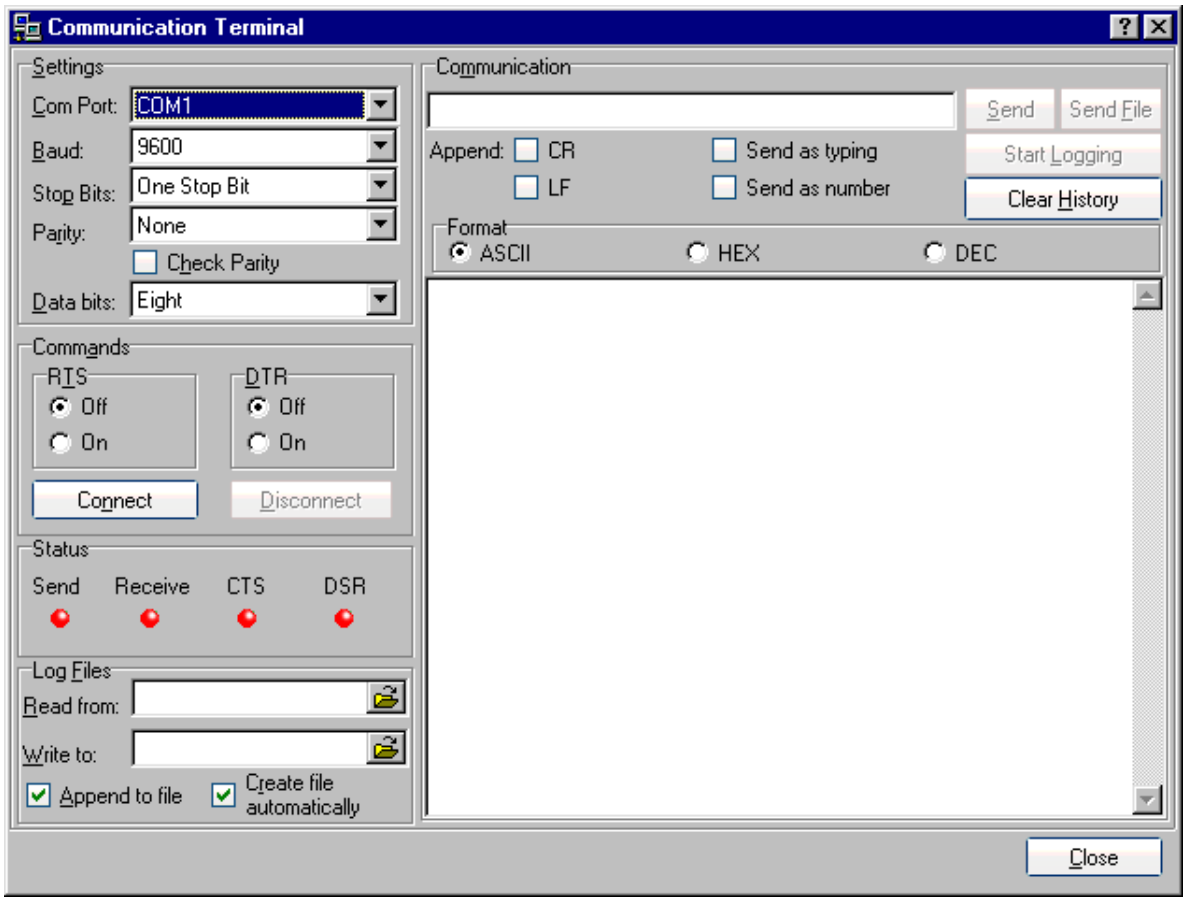


Figura 9 – Utilizando o Communication Terminal

Escolha em Com Port a porta que esteja livre em seu PC para comunicação. Em seguida, em baud mantenha a taxa de 9600 bps. Configure em Stop Bit a opção One Stop Bit e em paridade None. Utilizaremos a comunicação de 8 bits de dados, escolha desta forma esta opção em Data Bits. Muito bem, com estes parâmetros ajustados podemos iniciar a comunicação. Para isso, clique em Connect. Neste momento a porta estará aberta e pronta para enviar dados. Na caixa Communication, digite o caracter “A” e clique em send. Neste momento, o relé deverá ser acionado. Para desativar o mesmo o processo é o mesmo, bastando apenas trocar o caracter para “B”.

Exemplo V

Nesta série de artigos apresentados, veremos neste capítulo como medir uma tensão analógica na entrada do microcontrolador e transmitir este resultado via canal serial de comunicação RS232 o resultado da conversão para o PC. O microcontrolador PIC18F442 possui internamente um *conversor analógico para digital* de 10 bits. A idéia deste exemplo será a de transmitir de 1 em 1 segundo, o valor presente na entrada RA0 e através do Communication Terminal do mikroBASIC poder visualizar o resultado da conversão AD. Como o AD do PIC18F442 é de 10 bits e 2^{10} é igual a 1024, podemos definir a resolução de cada bit nesta conversão. Digamos que o nosso sistema irá variar a tensão de 0 a 5 Vcc. Para acharmos a tensão referente a alteração de uma unidade de AD no microcontrolador, devemos dividir a diferença entre as tensões de referência positiva e negativa pela resolução do AD. Observe o box 1 para verificar melhor esta fórmula.

Rbit = (Vref+ - Vref-) / Resolução do AD

Box 1 – Cálculo da Resolução de AD

Desta forma, conforme a tensão analógica presente na entrada RA0 variar 4,88 mV, o resultado da conversão AD variará 1 bit do seu resultado. Fique claro que quando o resultado da conversão for 1023, teremos na entrada o equivalente a 5 V, pois este é o resultado máximo de conversão e quando o resultado da conversão for 0, teremos 0 Vcc na entrada.

As entradas dos pinos analógicos estão presentes nos pinos RA0, RA1, RA2, RA3, RA5, RE0, RE1 e RE2 e quando eles são usados na forma analógica, nos referimos aos mesmos como AN0, AN1, AN2, AN3, AN4, AN5, AN6 e AN7. Na placa didática PICLAB18F442 temos dois trimpots, sendo o primeiro ligado ao RA0 e o segundo no RA1. De acordo com a tensão presente no pino RA0, este resultado será convertido para digital e enviado em seqüência para o PC através da RS232. Com o resultado que surgir no PC, poderemos saber perfeitamente a tensão que está presente na entrada analógica. Digamos que seja observado o resultado 500 no Communication Terminal. Como sabemos através da fórmula anterior que cada bit eqüivale a tensão de 4,88 mV, basta multiplicarmos o resultado da conversão por este valor e encontrarmos a tensão real na entrada do microcontrolador. Desta forma, encontraremos a tensão de 2,44 V que é a tensão encontrada neste caso. Apesar do AD do PIC ser de 10 bits, neste exemplo utilizaremos somente 8, de forma com que este valor seja enviado em um único byte para o PC. A lógica de funcionamento permanece a mesma, porém agora a resolução será 256 passos em função da resolução adotada.

Recursos de Hardware

O esquema elétrico para execução deste experimento pode ser observado na figura 1. Observe que no pino RA0 do microcontrolador está conectado o trimpot que será utilizado para medirmos a tensão analógica ajustada. A parte da comunicação RS232 permanece a mesma do artigo passado, mantendo desta forma a utilização do MAX232 como circuito de conversão.

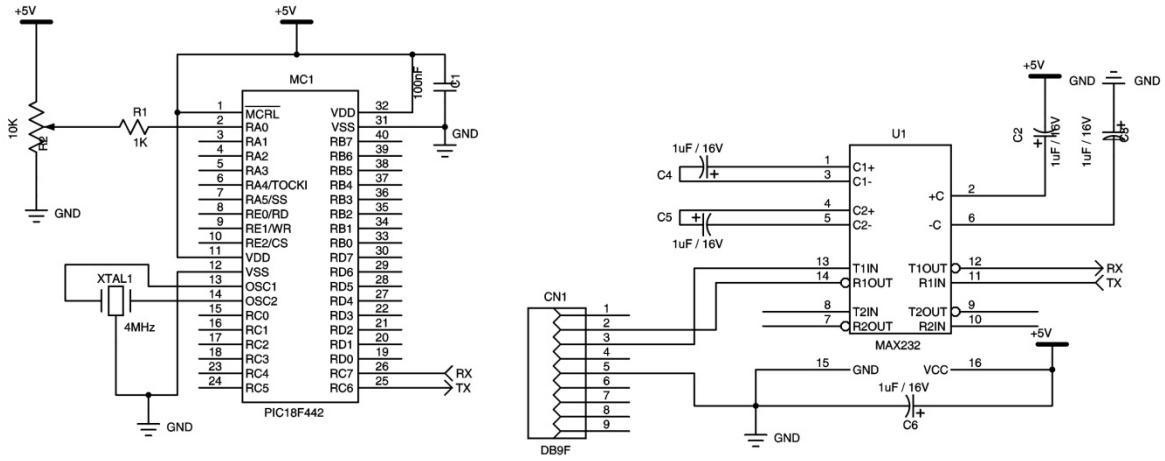


Figura 1 – Esquema Elétrico do Exemplo

Fluxograma

A lógica de funcionamento deste exemplo está apresentado na figura 2. Note que primeiramente a USART é configurada e após este processo, de 1 em 1 segundo o programa fica enviando pelo canal serial o resultado da conversão AD.

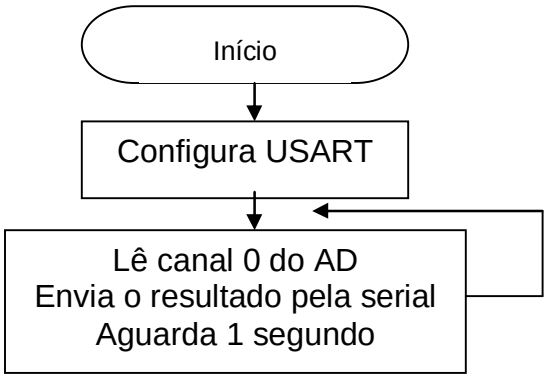


Figura 2 – Fluxo de funcionamento do exemplo

Recursos de Software

Neste exemplo veremos uma nova função disponibilizada pelo mikroBASIC que permite a medição de uma entrada analógica. Esta função está apresentada na tabela 1.

adc_read (*canal a ser lido*)

Tabela 1 – Função de leitura de AD

Quando esta função é chamada, devemos passar como parâmetro o canal de AD a ser lido. No PIC18F442 existem 8 canais que variam de 0 a 7 referentes aos pinos AN0 a AN7. No nosso caso, esta função deverá ser chamada da forma apresentada na tabela 2 permitindo assim a leitura do canal 0.

adc_read (0)

Tabela 2 – Chamando a função de leitura de AD

Software

O software completo que permitirá o teste deste exemplo está apresentado no box 2.

```
program ad

main:

     usart_init(9600)

repete:

     usart_write(adc_read(0)/4)
     delay_ms(1000)
     goto repete

end.
```

Box 2 – Listagem do Programa

Logo no início do programa, é configurada a taxa de comunicação utilizada pelo exemplo que neste caso é de 9600 bps. Em seguida, podemos visualizar que o resultado da conversão AD é enviado pelo canal serial através da função usart_write. Veja que a leitura do canal de AD é dividida por 4 antes do valor ser enviado. Isto é necessário pelo fato do resultado retornado pela função de leitura ser em 10 bits e como somente os oito bits mais significativos serão enviados pelo canal serial, houve a necessidade de retirar os dois bits menos significativos do resultado da conversão. Isto foi feito dividindo o valor retornado por esta função por 4 e logo em seguida enviando o mesmo via RS232. Na próxima linha de programa encontramos um retardo de programa através da função delay_ms.

Finalmente o programa entra em loop, repetindo este processo infindavelmente através do comando goto.

Após escrever este programa e compilar o mesmo, transfira o arquivo hex gerado pelo compilador para a placa didática PICLAB18F442. Abra neste momento o Communication Terminal e veja o resultado da conversão analógica chegar via RS232.

Exemplo VI

No artigo desta edição, veremos os passos para controlar um dispaly gráfico de 128 x 64 pixels. A idéia será apresentar uma imagem neste display. Estes displays são utilizados hoje por exemplo em celulares e uma das grandes vantagens da utilização deles é o fato dos mesmos funcionarem por pixels ao invés de caracteres definidos. Desta forma, podemos imaginar que podemos formar qualquer caractere de qualquer fonte assim como tamanho. O display gráfico utilizado está apresentado na figura 1. Observe que o mesmo é composto de 128 x 64 pixels, ou seja, ele tem o comprimento de 128 pixels e a altura de 64 pixels.

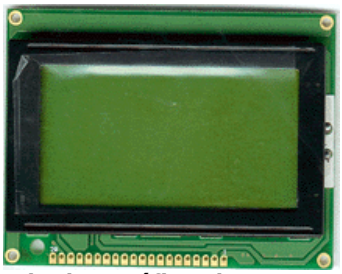


Figura 1 – Display gráfico de 128 x 64 pixels

Este display apresenta a pinagem apresentada na figura 2. Observe que temos 8 vias de dados e 6 de controle, além da parte de alimentação do display assim como a de controle do backlight (luz de fundo).

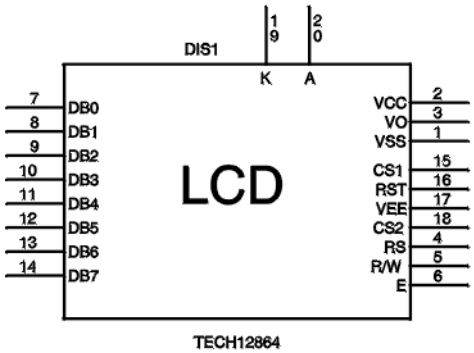


Figura 2 – Pinagem do display gráfico

A função destes pinos podem ser vistos com mais detalhes na tabela 1.

Pino	Descrição
1 (Vss)	Terra da alimentação do display
2 (Vdd)	Alimentação positiva de 5 V
3 (Vo)	Ajuste do contraste do display
4 (RS)	Controle de envio de dados ou programa para o display
5 (RW)	Controle de escrita ou leitura no display
6 (E)	Envio de pulso de habilitação do display
7 – 14 (DB0 a DB7)	Barramento de dados do display
15 (CS1)	Seleção do circuito de varredura da coluna 0 a 63
16 (RST)	Reset do display
17 (Vee)	Saída de tensão para ajuste do contraste
18 (CS2)	Seleção do circuito de varredura da coluna 64 a 127
19 (K)	Cátodo do backlight
20 (A)	Ânodo do backlight

Tabela 1 – Função dos pinos do display

Para executarmos esta experiência, faremos uso da placa PIC MASTER desenvolvida pela Cerne Tecnologia (**www.cerne-tec.com.br**). Observe esta placa na figura 3.

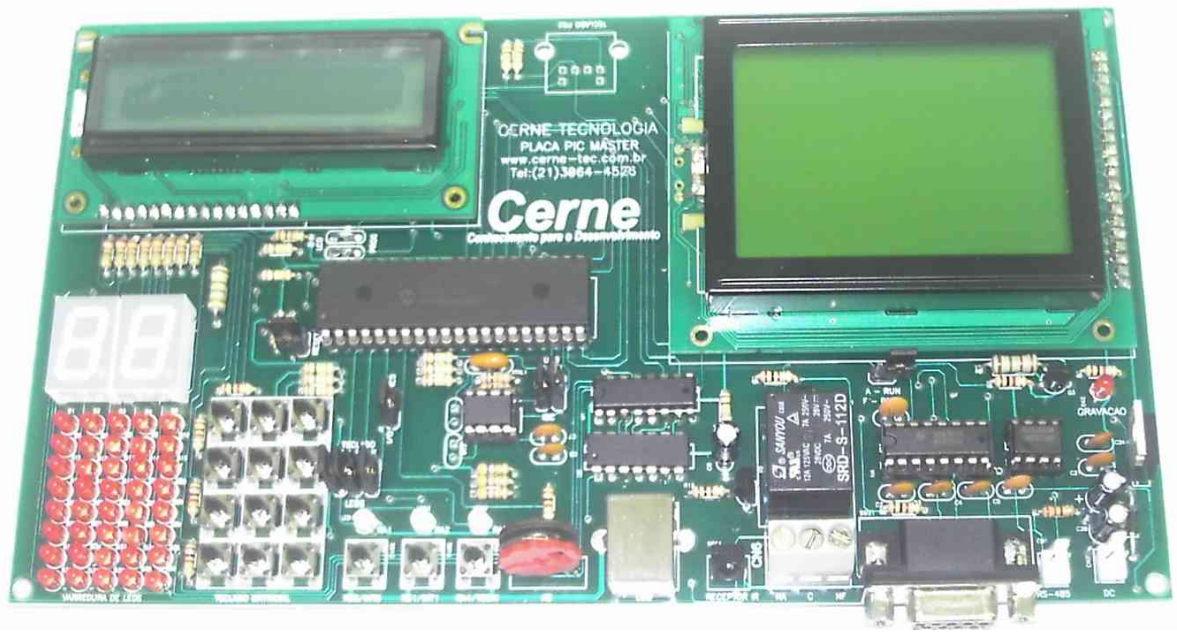


Figura 3 – Placa PIC MASTER desenvolvida pela Cerne Tecnologia

As características desta placa podem ser observadas na tabela 2.

Display Gráfico de 128 x 64 pixels
Display LCD 16x2
Display de 7 segmentos
Comunicação RS232
Comunicação RS485
Varredura de Leds
Comunicação USB 2.0
Comunicação com teclado PS2
Acionamento de Relé
Gravador On-Board

Tabela 2 – Características da placa PIC MASTER

Iremos utilizar no exemplo de hoje, o recurso display gráfico da placa PIC MASTER. Esta placa pode ser adquirida pela página da Cerne Tecnologia, no endereço **www.cerne-tec.com.br** ou através de nossa central de atendimento, no telefone (21)3064-4526.

Recursos de Hardware

O esquema elétrico de funcionamento deste exemplo está apresentado na figura 4. Note que o barramento de dados do display está integralmente conectado ao PORTD do microcontrolador PIC18F442. Já as linhas de controle RS, RW, EN, CS2, RST e CS1 estão conectadas ao PORTB, nos pinos 2, 3, 4, 5, 6 e 7 deste PORT respectivamente.

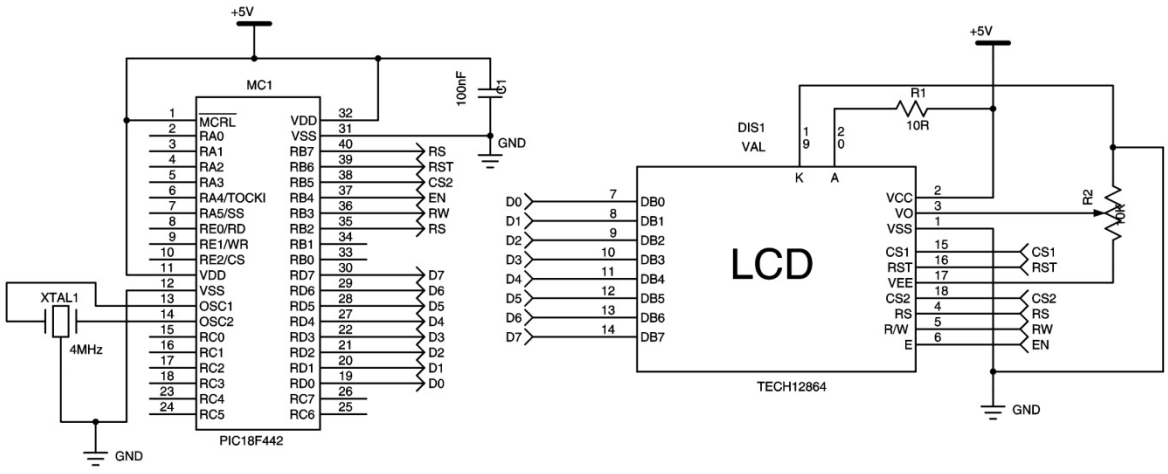


Figura 4 – Esquema Elétrico

Fluxograma

O fluxo de funcionamento deste exemplo está apresentado na figura 5. Note que logo no início do programa, o display gráfico é configurado e logo em seguida é carregada uma imagem neste dispositivo.

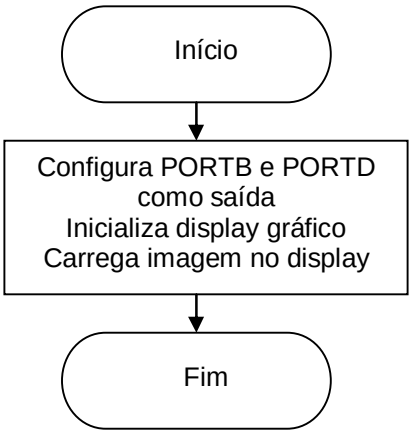


Figura 5 – Fluxograma do Exemplo

Recursos de Software

A intenção deste projeto é poder carregar uma imagem no display gráfico. Existem basicamente duas funções que permitem com que esta operação seja feita. Observe a tabela 3.

glcd_init (porta de controle, CS2, CS1, RS, RW, RST, EN, porta de dados)
glcd_Image (imagem a ser carregada)

Tabela 3 – Funções de acesso ao LCD

A primeira função configura os pinos onde estão ligados os pinos de controle e de dados do display. Conforme o esquema elétrico apresentado na figura 4, este comando ficaria da forma apresentada na tabela 4.

glcd_init(portb,5,7,2,3,6,4,portd)

Tabela 4 – Inicializando o display

Já a segunda função tem a incumbência de carregar uma imagem no display. Desta forma, devemos informar o vetor que mantêm a imagem para esta função.

Software

O mikroBASIC disponibiliza uma ferramenta muito importante que será de suma importância para execução deste exemplo. Após criar o projeto para o microcontrolador PIC18F442, vá no menu Tools e abra a opção Glcd Bitmap Editor. A tela da figura 6 surgirá.

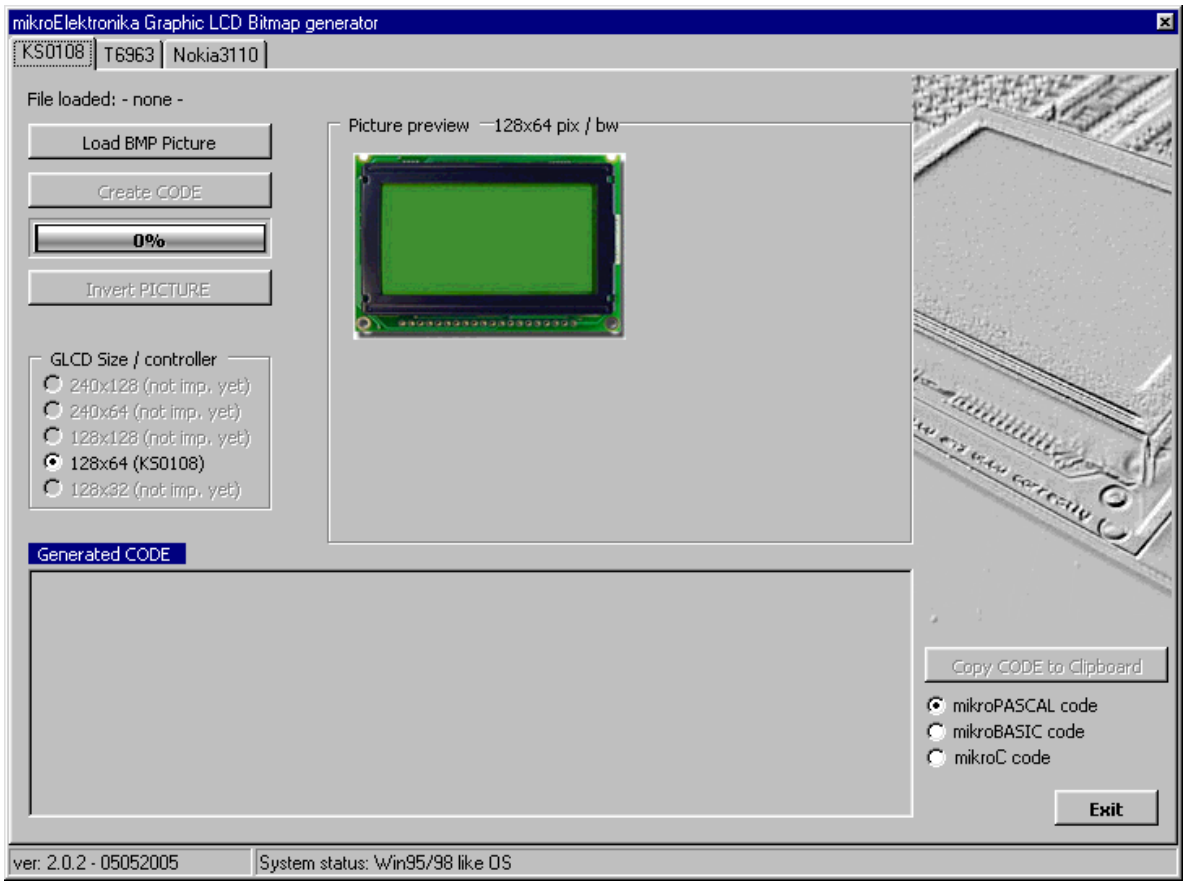


Figura 6 – Abrindo o GLCD Image

Note que destarte, o programa mostra o display utilizado em nosso projeto. Existem outros tamanhos de display, porém por enquanto vamos nos ater ao display de 128 x 64 pixels. Para carregar uma imagem a ser apresentada no display, clique em *Load BMP Picture.* Surgirá a tela da figura 7.

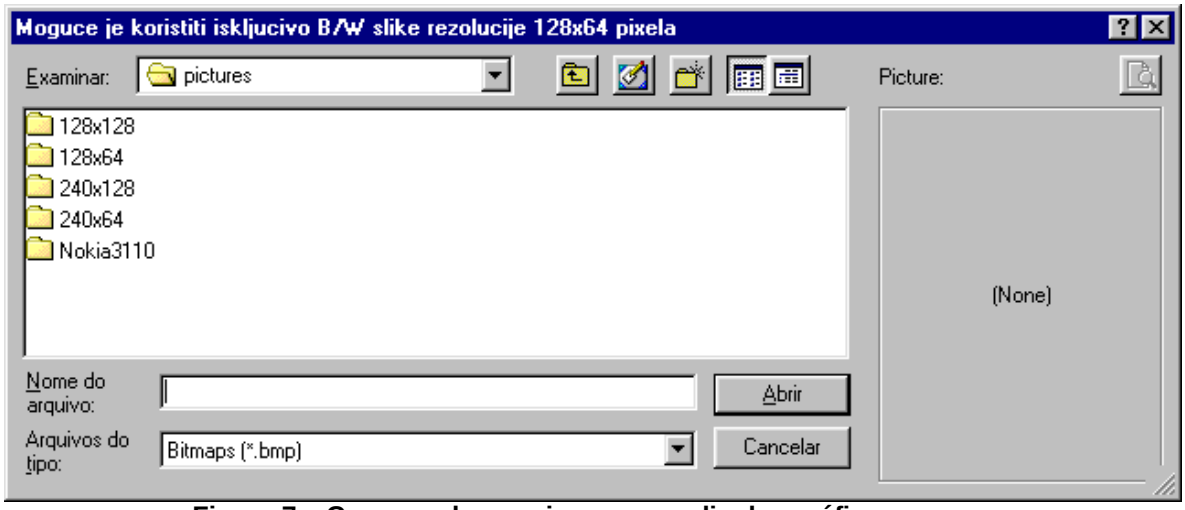


Figura 7 – Carregando uma imagem no display gráfico

Repare que existem algumas pastas relativas a displays de diversas dimensões. No nosso caso, clique na pasta de 128x64. Abrindo esta pasta, você notará uma série de figura já prontas que podem ser utilizadas para carregar uma imagem no display. Vamos utilizar uma delas, clique por exemplo no arquivo truck.bmp. Nada impede que você utilize outras imagens. A única exigência é que as mesmas tenham a dimensão adotada pelo display. O resultado será agora o que está apresentado na figura 8.

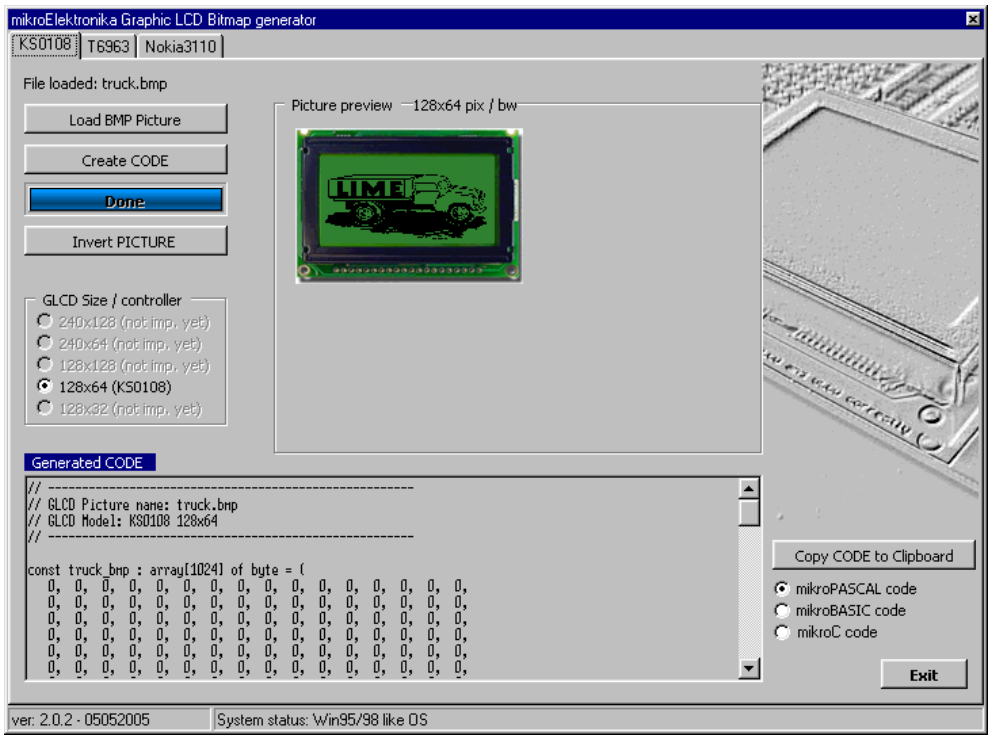


Figura 8 – Resultado após carregar a imagem

Observe que a imagem apresentada já é uma amostra do que será visto assim que o programa for gravado no microcontrolador. Você também pode inverter o estado dos pixels, deixando o que está aceso apagado e vice-versa. Para isso, clique no botão *Invert Picture.* O resultado será como o que está apresentado na figura 9.

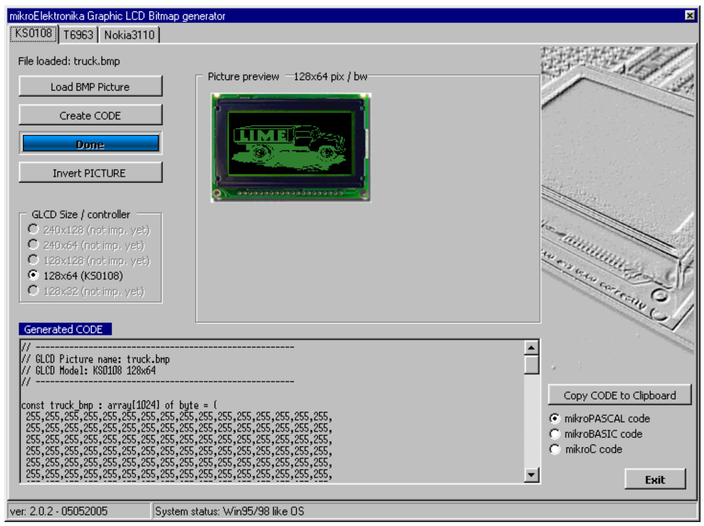


Figura 9 – Resultado após inverter o estado dos pixels

De acordo com a figura carregada, o mikroBASIC gera um vetor que é a imagem que será carregada pela função glcd_image. Vamos copiar este vetor para o nosso programa. Para isso, clique no option button chamado *mikroBASIC code* e logo em seguida em *Copy CODE to Clipboard.* Feito esta operação, esta janela pode ser fechada. Agora voltando ao mikroBASIC, cole este vetor que está salvo na memória logo abaixo da linha *program.* O código completo que permitirá carregar a imagem está apresentado no box 1.

```
program display

' ------------------------------------------------------
' GLCD Picture name: truck.bmp
' GLCD Model: KS0108 128x64
' ------------------------------------------------------

const truck_bmp as byte[1024] = (
   0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
   0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
   0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
   0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
   0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
   0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
   0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
   0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
   0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,128,128,128,128,
```

```
128, 64, 64, 64, 64, 64, 64, 32, 32, 32, 32, 32, 32, 32, 32, 32,
  32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32,
  32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32,
  32, 32,160,160,160, 96,224,224, 96, 96, 96, 32,  0, 32, 32, 32,
  32, 32, 32, 32, 32, 32,  0,  0, 64, 64, 64,  0,128,  0,  0,  0,
   0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
   0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
   0,  0,  0,  0,  0,254,  0,  1,  1,  1,  1,  1,253,253,253,253,
   1,  1,  1,  1,  1,  1,253,253,253,253,  1,  1,  1,  1,253,253,
 253,253,249,241,225,193,129,193,225,249,253,253,253,253,253,  1,
   1,  1,  1,253,253,253,253,221,221,221, 29, 17,  1,255,  1,  1,
   1,  1,  0,254,  1,168,  8,  8,  8,  8,  8,  8,  8,  8,  8,  8,
   8,  8,  8,  8, 16,224, 24, 36,132,  0,  2,130,  5, 81, 68,112,
 160,192, 96, 96, 32,  0,128,128,192,192,192, 64,  0,  0,  0,  0,
   0,128,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
   0,  0,  0,  0,  0, 63, 96, 96, 96,224, 96, 96,127,127,127,127,
 124,124,124,124, 96, 96,127,127,127,127, 96, 96, 96, 96,127,127,
 127,127, 97,103,111,127,127,111,103, 99, 97,127,127,127,127, 96,
  96, 96, 96,127,127,127,127,125,125,125,124,116, 96,127, 64, 64,
  64,160, 96,252,246,  1, 14,  6,  6,  2,  2,  2,  2,  2,  2,  2,
   2,  2,  2,  2,  2, 66,114, 60, 32, 16, 16,  0,  4,  2,  3,  2,
   0,  0,  0,  4,  4,  4,  4, 20, 16, 16, 16, 16, 16,  9,  9, 41,
 112, 32, 67,  5,240,126,174,128, 56,  0,  0,  0,  0,  0,  0,  0,
   0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  1,  1,  1,  1,  1,  1,
   1,  1,127,127,127,127,255,255,247,251,123,191, 95, 93,125,189,
 189, 63, 93, 89,177,115,243,229,207, 27, 63,119,255,207,191,255,
 255,255,255,255,255,255,255,127,127,127,127,127,127,127,127,255,
 255,255,127,127,125,120,120,120,120,120,248,120,120,120,120,120,
 120,248,248,224,135,  0,  0,  0,  0,  0,  0,  0,  0,128,240,248,
 120,188,220, 92,252, 28, 28, 60, 92, 92, 60,120,248,248, 96,192,
 143,168,216,136, 49, 68, 72,  2,160, 96,  0,  0,  0,  0,  0,  0,
   0,  0,  0,128,192,248,248,248,248,252,254,254,254,254,254,254,
 254,254,254,254,254,255,255,255,255,255,246,239,208,246,174,173,
 169,128,209,208,224,247,249,255,255,252,220,240,127,255,223,255,
 255,255,255,255,255,254,254,255,255,255,255,255,255,255,254,255,
 255,255,255,255,255,255,254,254,254,254,254,254,254,254,254,254,
 254,254,254,254,255,255,255,255,255,255,254,255,190,255,255,253,
 240,239,221,223,254,168,136,170,196,208,228,230,248,127,126,156,
 216,224,240,240,242,242,240,177, 32,  0,  0,  0,  0,  0,  0,  0,
   0,  0,  0,  1,  1,  1,  1,  3,  3,  3,  7,  7,  7,  7,  7, 15,
  15, 15,  7, 15, 15, 15,  7,  7, 15, 14, 15, 13, 15, 47, 43, 43,
  43, 43, 43, 47,111,239,255,253,253,255,254,255,255,255,255,255,
 191,191,239,239,239,191,255,191,255,255,255,255,255,255,255,255,
 255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
 255,255,255,255,127,127,127,127,255,255,191,191,191,191,255,254,
 255,253,255,255,255,251,255,255,255,127,125, 63, 31, 31, 31, 31,
  31, 31, 63, 15, 15,  7,  7,  3,  3,  3,  0,  0,  0,  0,  0,  0,
   0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
   0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
   0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  1,  0,
   1,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  1,  1,  1,
   1,  1,  1,  1,  3,  3,  3, 11, 11, 11, 11,  7,  3, 14,  6,  6,
   6,  2, 18, 19, 19,  3, 23, 21, 21, 17,  1, 19, 19,  3,  6,  6,
  14, 15, 15,  7, 15, 15, 15, 11,  2,  0,  0,  0,  0,  0,  0,  0,
   0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0
)

main:
     trisb=0
     trisd=0
     glcd_init(portb,5,7,2,3,6,4,portd)
     glcd_image(truck_bmp)

end.
```

Box 1 – Programa Completo do Exemplo

Vamos desmistificar melhor este código. Logo após a declaração de *program display* está declarado o vetor que é a imagem propriamente dita que será carregada no display. Em seguida os pinos onde está ligado o display são configurados como saída e o display é inicializado pela função glcd_init. Finalmente a imagem é carregada pela função glcd_image, que carrega o vetor de constantes criado pela ferramenta Glcd Image e copiado logo abaixo da linha program.

Após o desenvolvimento e compilação deste programa, grave o arquivo hex no microcontrolador que está na placa didática PIC MASTER da Cerne Tecnologia e comprove o funcionamento.

Conclusão

Os displays gráficos estão cada dia mais presentes nos projetos eletrônicos. Nos equipamentos de suporte à vida, eles são utilizados por exemplo para mostrar a pulsação de um paciente. Saber utilizar estas IHMs é importante para o desenvolvimento de projetos eletrônicos que venham a necessitar de sua utilização.