Livro sobre micro controladores PIC em linguagem C – Compilador CCS – Volume 1

Relógio com GPS

GPS com PIC

TX de 433MHz com PIC e ASCII

RX de 433MHz com PIC, RS232, Rede e Eeprom

Luiz Bertini

Este livro não é um livro necessariamente para iniciantes em linguagem C e utiliza o compilador da CCS. Mas poderá ajuda-lo muito e até ser o início de seu aprendizado, pois já vi, na pratica, alguns alunos, de cursos extracurriculares que eu ministrava, pegarem estes e outros exemplos e "debulharem", para a minha felicidade, o código fonte.

Terei um livro que começara do zero, bem como outros que darão continuidade a esta série.

Ele traz alguns circuitos e códigos fontes, todos testados e funcionando e tem a intenção de despertar novas ideias em usuários e de micro controladores PIC. Mas tudo o que tem teoria tem pratica, você deve montar os circuitos, soldar, ligar, e realmente aprender.

Nele você aprenderá:

Como ler um protocolo recebido pelos módulos de GPS chamado de RMC e como escrever as informações necessárias em um display LCD, fazendo um relógio controlado por GPS, porem em horário GMT. Deixou um pouco para você leitor.

Como ler este mesmo protocolo RMC e apresentar em um display as coordenadas do local onde você está. Fazendo um equipamento de GPS.

Como transmitir caracteres no código ASCII, que consistem, basicamente, nos mesmos caracteres de seu teclado, através de um circuito que pode funcionar com uma bateria ou fonte e usara ainda um modulo de transmissão de 433MHz.

Como receber uma informação via RF, com um modulo receptor de 433MHz, e se comunicar com a internet através de um modulo de rede. Além de conseguir grava-la em uma memória eeprom externa.

Para fazer estes projetos você precisara de um laboratório para trabalhar com o PIC usado, ou montar todo o circuito em uma placa universal (eu aconselho em placas universais PP18 ou PP20), um compilador da CCS (você consegue uma versão de avaliação gratuita em):

http://www.ccsinfo.com/ccsfreedemo.php

Ou pode comprar em:

http://www.acepiccamp.com.br/compilador-linguagem-c-ccs-pcwhd-p65?gclid=Cj0KEQjw_YKtBRC7zZjFp8bF_foBEiQAfyigc6Cdvz-jzsKb0xWldYG8LDgKniHZTSJ_jciuLmvd9P8aAk1_8P8HAQ

Ou

http://www.ccsinfo.com/content.php?page=compilers

Você pode também usar o MPLAB para gravar os PICs, eu prefiro fazer assim...

Baixe de raça em:

http://www.microchip.com

De uma procurada por lá.

Também obterá informações na minha página:

http://www.luizbertini.net/microcontroladores/microcontroladores.html

Os comentários entre parênteses em negrito e sublinhados, dentro dos códigos fontes fazem referência a alguma informação que eu acho importante ressaltar. Mas não fazem parte do código fonte e você deve retirá-los ao copiar ou escrever estes códigos, ou deixá-los como comentários.

Fique atento ao que é comando e ao que é comentário.

Bons estudos e boa sorte.

A linguagem C para micro controladores PIC, necessita deum compilador para ser escrita e revisada. Eu aconselho o uso do compilador da CCS, mas isto é uma questão de gosto. Tudo dito aqui se refere a micro controladores PIC e a este compilador.

Você deve seguir uma coisa, chamada de regra de escopo, ou seja, deve haver uma ordem em seu programa na construção das suas funções.

Uma função é um conjunto de instruções que pode ser chamada por outra função, que deve estar abaixo dela e isto é importante e faz parte do que chamamos de regra de escopo.

O ideal é você separar o seu programa de uma forma que fique bem claro de entender, com muitos comentários e muitas informações adicionais.

Ao usar pinos ou ports inteiros do PIC, deixe como comentário os pinos correspondentes a eles. Fica mais fácil para montar o hardware e consertar o software.

Tente fazer ambos, hardware e software, juntos, começando pelo hardware, assim você se transformará em um desenvolvedor de circuitos práticos e uteis.

Relógio com GPS e micro controlador PIC em linguagem C com uso do compilador da CCS

Através da leitura de diversos dados, que chegam em uma stream (trem de bits) de um modulo de GPS e salvando estes dados em diversas variáveis, fazer as comparações necessárias e apresentar as informações da hora GMT em um display, com precisão de um modulo de GPS. Você poderá alterar o código fonte para corrigir a hora para o seu local. A finalidade deste circuito e código fonte é mostrar uma maneira de como obter informações de um trem de bits que sempre está enviando dados. E como separar os dados que lhe interessam, neste caso a hora, e apresentar em um display LCD.

SE você pretende interligar um modulo GPS com um micro controlador PIC eu aconselho a ficar atento a algumas coisas:

O protocolo de comunicação usados pelo GPS e o NMEA-0183. Para ter mais informações sobre ele baixe um datasheet de um modulo de GPS de um site, pode até ser do meu:

http://www.luizbertini.net/microcontroladores/microcontroladores.html

Isto é o ideal para você adquirir mais conhecimentos sobre este protocolo.

Este protocolo consiste de uma stream de bits com diversas informações. Este stream está dividido em

mensagens diferentes de saída, veja o nome de algumas delas:

GGA, GLL, GSA, GSV, MSS, RMC, VTG, ZDA, ETC.

Nestas mensagens você encontrara diversas informações como: hora UTC, latitude, longitude, quantidade de satélites recebidos, altura, velocidade, etc.

Estas mensagens estarão disponíveis nos pinos de TX do modulo de GPS, que normalmente se comunica no padrão RS232, porem as vezes o nível de tensão correspondente a 1 é igual a 3,3 volts.

Os dados serão apresentados assim:

4800 BPS

8 bits de dados

0 bits de paridade

1 stop bit

Tudo em ASCII. Lembre-se que a comunicação elétrica é feita em valores de tensão que serão considerados como 0 e 1.

0 = sem tensão.

1 = com tensão.

Mas estes dados serão lidos assim:

48 em decimal será igual a 0x30 em hexadecimal que corresponde a 0 em ASCII.

Outro exemplo:

57 em decimal será igual a 0x39 em hexadecimal que corresponde a 9 em ASCII.

Perceba que, se você só ler a hora com o micro controlador e enviá-la para o display de LCD, sem converter para ASCII, acontecerá o seguinte:

Hora correta recebida = 10h25min

Enviando sem converter e lendo no LCD:

49 48:50 53 ou coisa pior...

É preciso converter para ASCII que é a representação do seu teclado, por exemplo.

Mas como fazer esta conversão? No nosso compilador e em C você pode usar o comando:

PRINTF (ESCREVE_DISPLAY, ”%2c”, VARIAVEL);

O C que vem depois do 2 converterá a informação recebida para ASCII, sacou?

Agora, voltando as mensagens do GPS, saiba que:

Todas começam com $

Todas acabam com <CR><LF>

CR = 13 em decimal e significa retorno de carro (desde as maquinas de escrever...)

LF = 10 em decimal e significa nova linha

< = 60 em decimal

>= 62 em decimal

Voltando a parte elétrica da saída do modulo saiba que:

O nível de tensão em sua saída de TX é igual a LOW TTL, ou seja, varia entre 0 e 2,85 volts. Onde:

Nível 0 = 0 volts.

Nível 1 = 2,85 volts.

Portanto para ligar este modulo com um PIC eu aconselho usar uma interface feita com transistores conforme o esquema.

Já que falamos do PIC e do compilador da CCS, vamos ver mais algumas coisas:

Você irá usar a instrução:

FGETC (dados);

Para capturar as informações da stream. Com esta função você deverá usar pinos não associados ao hardware interno de TX e RX do PIC, vai por mim.

O hardware interno está associado aos pinos, no caso

26 = RC7 = RX

25 = RC6 = TX

Escolha outros pinos e use esta função abaixo:

X = FGETC (DADOS);// ONDE X É UMA VARIÁVEL E
//DADOS E O NOME DE SUA STREAM PARA PEGAR
//ESTES DADOS.

Você pode ler uma mensagem de saída desejada ou mais de uma, mas deve salvar cada byte dela, inclusive as virgulas, em uma variável diferente. Desta forma não há confusão, mas você gastara bastante memória. Para economizar memoria use uma variável comum para ler os dados que não precisam ser guardados e usados depois (estude o código fonte e isto ficará mais claro).

“Todos os bytes do pino TX do GPS devem ser lidos de uma vez só, ou melhor falando, de forma continua, e salvos em diferentes variáveis e só depois tratados. Faça isto. ”

De todas as mensagens de saída do modulo GPS eu aconselho você ler a RMC ou a GGA. Nosso código fonte foi montado para ler a RMC.

Vamos ter uma ideia de como é esta mensagem RMC:

$GPRMC,161219.487,A,3723.2475,N,12158.3416,W,O.13,309.62,120598,,*10

161219 = HORA, MINUTOS E SEGUNDOS

Depois tem longitude, latitude entre outras coisas.

Agora vamos ao programa ou código fonte ou algoritmo:

Na parte do programa chamada de: leitura da stream do RMC nós temos diversas variáveis que correspondem as letras $GPRMC e serão usadas para identificar esta mensagem no meio de todas as outras. Parte do segredo do código está aqui.

Na parte do programa chamada de: leitura da hora nós temos diversas variáveis que receberão os dados correspondentes a hora.

Na parte do programa chamado de: escreve no display, você tem uma linha de comando que lê os dados recebidos e escreve a hora no display caso eles correspondam a mensagem RMC. Caso você queira ler a mensagem GGA você pode usar a seguinte linha de comando:

IF(LE1==71 && LE2==80 && LE3==71 && LE4==71 && LE5==65){INSTRUÇÕES ETC E TAL}

Mas perceba que:

G = 71

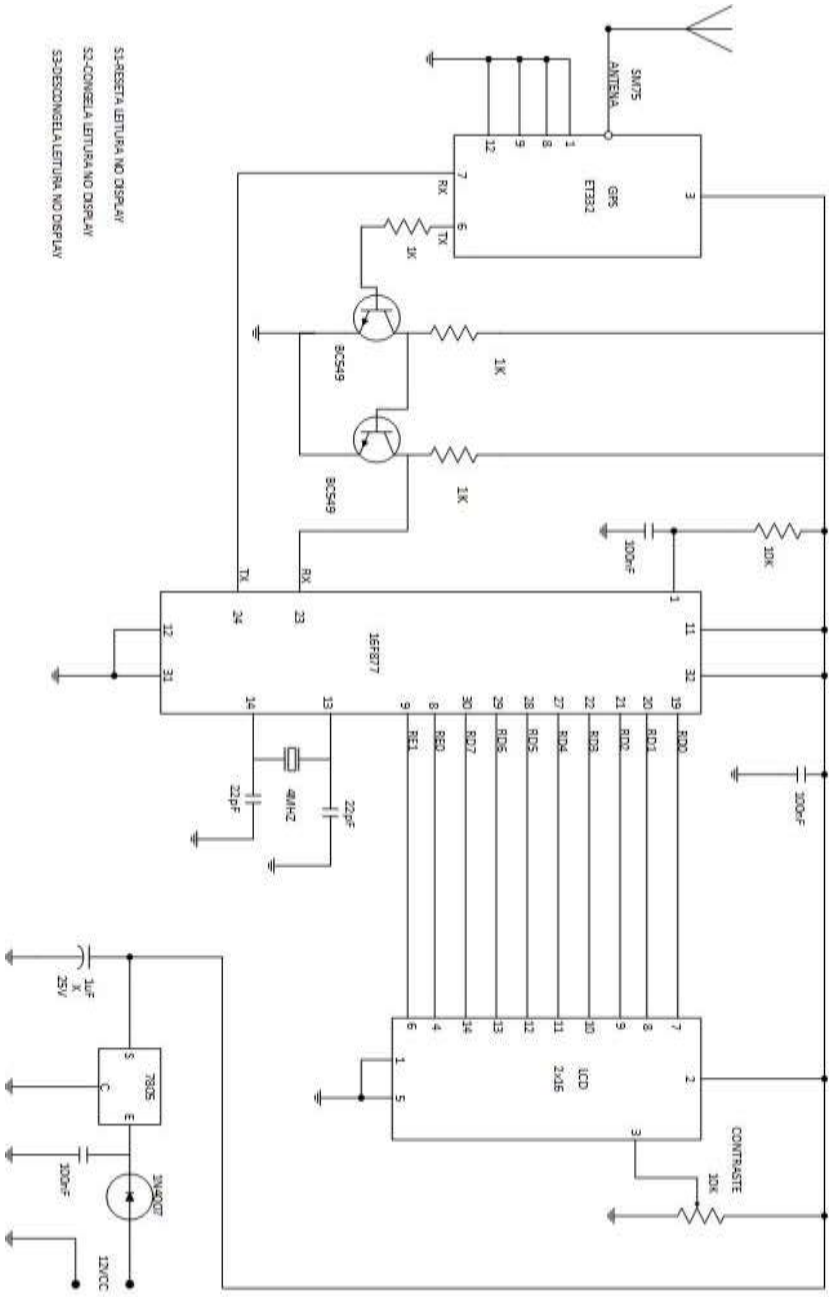
P= 80

G = 71

G = 71

A = 65

O resto do programa é o mesmo, mas cuidado ao altera-lo. Sempre salve uma versão anterior.


S1-RESETA LEITURA NO DISPLAY
S2-CONGELA LEITURA NO DISPLAY
S3-DESCONGELA LEITURA NO DISPLAY
SMTS
ANTENA
GPS
ET332
RX
TX
1K
BC549
BC549
1K
1K
100nF
10K
16F877
4MHZ
22pF
22pF
100nF
RE1
RE0
RD7
RD6
RD5
RD4
RD3
RD2
RD1
RD0
LCD
2x16
CONTRASTE
10K
1uF
25V
7805
100nF
2N6027
12VCC

Código fonte:

```
//-------------------------------------------------------------------------

/*Relógio com GPS

Luiz bertini/Luciana Petraites

Todos os dados da stream desejada do GPS devem ser lidas sequencialmente e

salvas em variáveis. Depois as variaveis devem ser enviadas para o display.

Lembre-se de que tudo o que está escrito na stream do GPS está em

ASCII, inclusive os números, portanto use, em printf, a letra u.

Você deve criar uma variável única para salvar nela o que não lhe importa

colocar no display. Isto economiza memória. */

//------------------------variáveis internas do pic----------

#include <16f877.h>// características do 16F877

//--------------------configurações dos fusíveis para gravar--

#fuses xt,wdt,noprotect,put,brownout,nolvp,nocpd,nowrt

//--------------------------definição da rotina de delay-------

#use delay(clock=4000000, restart_wdt)
```

```
//---------------------------definição de comunicação serial-

#use
rs232(baud=4800,bits=8,xmit=pin_c5,rcv=pin_c4,strea
m=txrx,parity=n,restart_wdt)

//o comando acima faz a comunicação com o ET332

//====definição dos ports====================

#use  standard_io(a)

#use  standard_io(b)

#use  standard_io(c)

#use  standard_io(d)

#use  standard_io(e)

#byte porta = 0x05

#byte portb = 0x06

#byte portc = 0x07

#byte portd = 0x08

#byte porte = 0x09

//========definição dos bits====================

#bit ra0 = porta.0

#bit ra1 = porta.1

#bit ra2 = porta.2

#bit rc4 = portc.4
```

```
#bit rc5 = portc.5

#bit rd0 = portd.0

#bit rd1 = portd.1

#bit rd2 = portd.2

#bit rd3 = portd.3

#bit rd4 = portd.4

#bit rd5 = portd.5

#bit rd6 = portd.6

#bit rd7 = portd.7

//============variáveis======================-

int le0=0;

int le1=0;

int le2=0;

int le3=0;

int le4=0;

int le5=0;

int le6=0;

int le7=0;

int le8=0;

int le9=0;
```

```
int le10=0;

int le11=0;

int le12=0;

int leX=0;

//===definição de saídas do porte para LCD==========

#bit rs = porte.0// pino do lcd que define recepção de comandos ou dados.

#bit enable = porte.1//pino do lcd que o habilita

//==========funções==========================

//

//=====rotina para enviar comando para o LCD======

void COMANDO_LCD(int texto)

{

rs = 0;//seleciona para enviar um comando

portd = texto;//carrega o portd com o carcter

enable = 1;//gera pulso para acionar lcd

delay_us(1);//gera um delay de 1 microssegundo, necessário para o display.

enable = 0;//desabilita o lcd

delay_us(40);//espera 40 microssegundos, necessário para o display
```

```
return;//retorna
}
//==============escreve_lcd===================
void ESCREVE_LCD(int texto)
{
rs = 1;//habilita o envio de um comando
portd = texto;//carrega portb com o caracter
enable = 1;//cria pulso no enable
delay_us(1);//espera 1uS
enable = 0;//desabilita LCD
delay_us(40);//espera 40uS
return;
}
//===============limpando o LCD===============
void LIMPA_LCD()
{
comando_lcd(0x01);//limpa lcd
delay_ms (2);// espera 2 mS
return;
}
```

```
//===========inicialização do display=============

void INICIALIZA_LCD()

{

comando_lcd(0x30);//comando para inicializar display

delay_ms(4); //espera 4 mS

comando_lcd(0x30);

delay_us(100);

comando_lcd(0x30);

comando_lcd(0x38);

limpa_lcd();//limpa o lcd

comando_lcd(0x0c);//comando para display sem
cursor/ver apostilas sobre display

comando_lcd (0x06);

return;//retorna para a função principal (main) - fora
do while

}

//==============tela principal do display=========

void TELA_PRINCIPAL()

{

comando_lcd(0x80);//posiciona o cursor na linha 0 e na
coluna 1
```

printf (escreve_lcd, "HORA VIA GPS-UTC");//escreve HORA LONG LAT no LCD

return;//retorna para a função principal (main) - fora do while

}

//=========lendo dados RMC do GPS==========

void RMC()

{

restart_wdt();

(Este loop faz a leitura ser continua porem aleatória, mas permite obter os dados)

//===============leitura da stream do RMC========

//todos os comandos devem ser em sequencia

le0=fgetc(txrx);//$ = 36

le1=fgetc(txrx);//G = 71

le2=fgetc(txrx);//P = 80

le3=fgetc(txrx);//R = 82

le4=fgetc(txrx);//M = 77

le5=fgetc(txrx);//C = 67

le6=fgetc(txrx);//, = 44

//==============leitura da hora===============

```
le7=fgetc(txrx);//0

le8=fgetc(txrx);//0

le9=fgetc(txrx);//0

le10=fgetc(txrx);//0

le11=fgetc(txrx);//0

le12=fgetc(txrx);//0

//=============escreve no display===========

if(le1==71 && le2==80 && le3==82 && le4==77 &&
le5==67)

{

//================HORA=======================

LIMPA_LCD();

comando_lcd(0x80);

printf (ESCREVE_LCD, "    HORA UTC    ");//escreve hora

comando_lcd(0xC4);//define posição para escrever a
//hora

printf(ESCREVE_LCD,"%1c%1c",le7,le8);//escreve a
//hora com o

comando_lcd(0xC6);

printf (ESCREVE_LCD, ":");//escreve : (dois pontos)
```

```
comando_lcd(0xC7);//define posição para escrever a
//hora

printf(ESCREVE_LCD,"%1c%1c",le9,le10);//escreve a
//hora com o

comando_lcd(0xC9);

printf (ESCREVE_LCD, ":");//escreve :

printf(ESCREVE_LCD,"%1c%1c",le11,le12);//escreve a
//hora com o

//formato: 00:00 ou hora e minuto

}

return;

}

//===============principal====================

void main()

{

//=======setando as portas=====================

set_tris_a(0b11111111);//todas entradas

set_tris_b(0b11111111);//todas entradas

set_tris_c(0b11111111);//todas entradas

set_tris_d(0b00000000);//todas saidas. Comandará o
display LCD.
```

```
set_tris_e(0b00000100);//o 1 corresponde ao comando CS do LCD que deve ser uma

//entrada e ter o valor igual a zero para que possa ser //escrito no lcd. Quando

//colocamos um pino do pic como entrda ele fica em //zero. Caso CS tivesse um

//potencial igual a 1 estaríamos lendo o LCD.

//========limpando as portas=================

porta=0x00;

portb=0x00;

portc=0x00;

portd=0x00;

porte=0x00;

//===============resetando o watch dog=========

restart_wdt();

//=========programa principal==================

INICIALIZA_LCD();

TELA_PRINCIPAL();

//===========loop principal====================

while(true)

{
```

```
restart_wdt();

leX=fgetc(txrx);//aguarda LF=10. Isto sincroniza com as
mensagens de

//saida do GPS, pois todas terminam em <LF>. O > dá o
tempo para chamar

//a função.

if(leX==10) RMC();//chama função RMC para coleta de
dados do GPS.
```

(esta parte acima pode ser alterada, pense nisto)

```
}

}

//

//Sanctify Yourself

//

//======================The End==============
```

GPS com micro controlador PIC em C da CCS.

Agora você vai aperfeiçoar o seu projeto anterior e fazer um GPS completo, ou quase... mas:

É bom lembrar que tudo o que foi dito no relógio dom GPS serve aqui.

Nele nós iremos mostrar a latitude, a longitude, a posição no hemisfério, etc.

O código fonte lera a mensagem RMC. No esquema a chave S1 reseta o display e mostra a tela inicial.

A chave S2 trava a leitura para podermos visualizar melhor os dados obtidos.

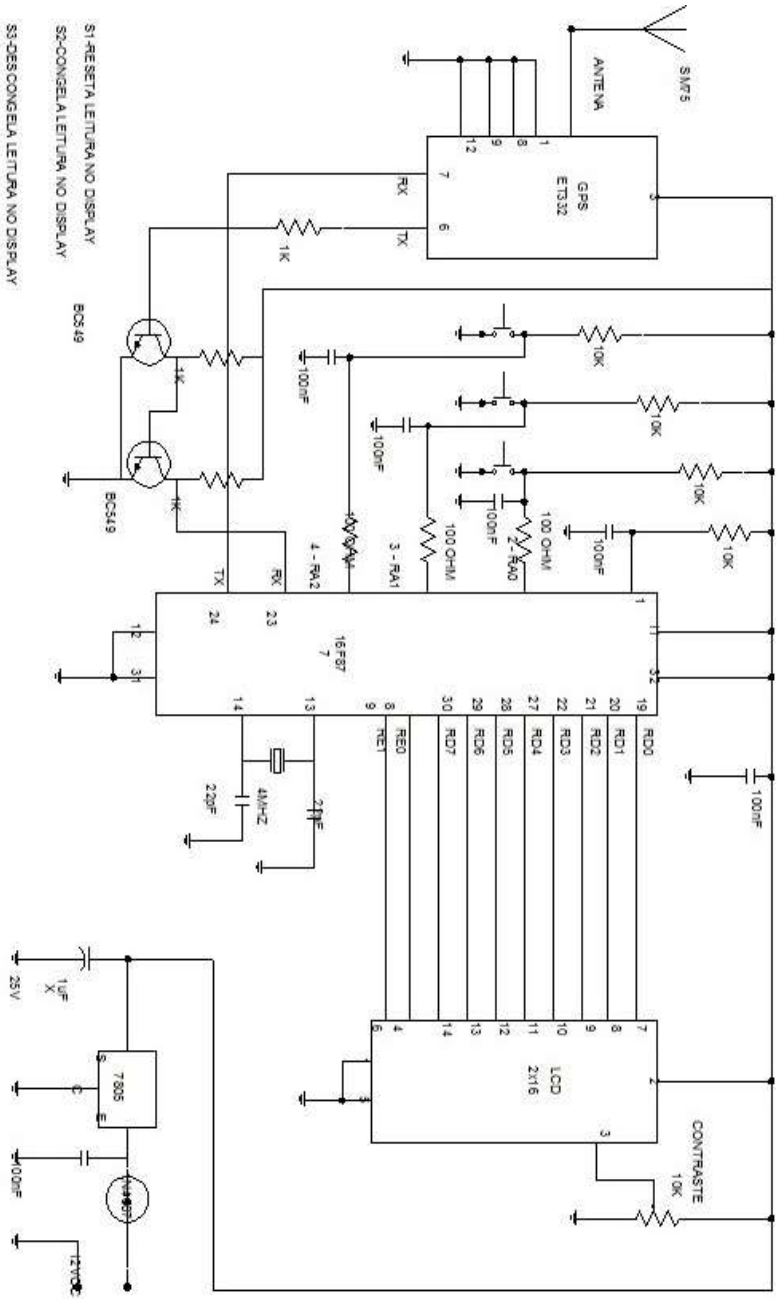
A chave S3 destrava a leitura.

As informações obtidas pelo GPS serão mostradas em duas telas diferentes, cada tela ficará aparecendo por dois segundos, por isto existem as chaves para travar em uma ou outra tela.

É importante ressaltar que o GPS demora um pouco para receber os sinais corretamente, depois de ligado, e isto faz com que o display demore um pouco para apresentar estas informações. Além do processamento do programa que demora um pouco para associar as variáveis com o RMC.

Antes de mais nada que me desculpem os puritanos, mas eventualmente eu uso GOTO em C, apenas para acelerar um processo. Mas o ideal é, eu assumo e sei disto, montar uma estrutura com diversas funções que faça o mesmo.

No loop principal deste programa está o segredo da “velocidade da leitura”. Caso você o altere para poderá diminuir a velocidade e o tempo de resposta de seu GPS. Mas fique à vontade, você deve fazer o que achar necessário.


S1-RESETA LEITURA NO DISPLAY
S2-CONGELA LEITURA NO DISPLAY
S3-DESCONGELA LEITURA NO DISPLAY
ANTENA
SMA5
GPS
ET332
RX
TX
1K
BC549
10K
100nF
100 OHM
4 - RA2
3 - RA1
2 - RA0
16F87
7
RB0
RB1
RB2
RB3
RB4
RB5
RB6
RB7
RE0
RE1
4MHZ
22pF
LCD
2x16
CONTRASTE
10K
7805
10uF
25V
100nF

Código fonte:

```
//=========================================

/*Leitor de dados GPS usando a mensagem de saída
RMC Versão 1.1

Apresenta uma leitura com tempos preciso e rápidos e
pode ser usado como GPS

Luiz bertini/Luciana Petraites - 25/02/2010 again and
again...

13/09/2013*/

========variáveis internas do pic============

#include <16f877.h>// características do 16F877

//=====configurações dos fusíveis para gravar======

#fuses
xt,wdt,noprotect,put,brownout,nolvp,nocpd,nowrt

//==========definição da rotina de delay==========

#use delay(clock=4000000, restart_wdt)

//===========definição de comunicação serial=====

#use
rs232(baud=4800,bits=8,xmit=pin_c5,rcv=pin_c4,strea
m=txrx,parity=n,restart_wdt)

//o comando acima faz a comunicação com o ET332

//====definicao dos ports e inicialização==========
```

```
#use  standard_io(a)

#use  standard_io(b)

#use  standard_io(c)

#use  standard_io(d)

#use  standard_io(e)

#byte porta = 0x05

#byte portb = 0x06

#byte portc = 0x07

#byte portd = 0x08

#byte porte = 0x09

//======definição dos bits=====================

#bit ra0 = porta.0

#bit ra1 = porta.1

#bit ra2 = porta.2

#bit rc6 = portc.4

#bit rc7 = portc.5

#bit rd0 = portd.0

#bit rd1 = portd.1

#bit rd2 = portd.2

#bit rd3 = portd.3
```

```
#bit rd4 = portd.4

#bit rd5 = portd.5

#bit rd6 = portd.6

#bit rd7 = portd.7

//================variaveis====================

int le0=0;

int le1=0;

int le2=0;

int le3=0;

int le4=0;

int le5=0;

int le6=0;

int le7=0;

int le8=0;

int le9=0;

int le10=0;

int le11=0;

int le12=0;

int le13=0;

int le14=0;
```

```
int le15=0;

int le16=0;

int le17=0;

int le18=0;

int le19=0;

int le20=0;

int le21=0;

int le22=0;

int le23=0;

int le24=0;

int le25=0;

int le26=0;

int le27=0;

int leX=0;

//==definição de saídas do porte para comandar LCD==

#bit rs = porte.0// pino do lcd que define recepção de
comandos ou dados.

#bit enable = porte.1//pino do lcd que o habilita

//==========funções==========================

//
```

```
//======rotina para enviar comando para o LCD======

void COMANDO_LCD(int texto)

{

rs = 0;//seleciona para enviar um comando

portd = texto;//carrega o portd com o caracter

enable = 1;//gera pulso para acionar lcd

delay_us(1);//gera um delay de 1 microssegundo,
//necessário para o display.

enable = 0;//desabilita o lcd

delay_us(40);//espera 40 microssegundos, necessário
//para o display

return;//retorna

}

//=====escreve_lcd===========================

void ESCREVE_LCD(int texto)

{

rs = 1;//habilita o envio de um comando

portd = texto;//carrega portb com o caracter

enable = 1;//cria pulso no enable

delay_us(1);//espera 1uS

enable = 0;//desabilita LCD
```

```
delay_us(40);//espera 40uS
return;
}
//=======limpando o LCD====================
void LIMPA_LCD()
{
comando_lcd(0x01);//limpa lcd
delay_ms (2);// espera 2 mS
return;
}
//==========inicialização do display========
void INICIALIZA_LCD()
{
comando_lcd(0x30);//comando para inicializar display
delay_ms(4); //espera 4 mS
comando_lcd(0x30);
delay_us(100);
comando_lcd(0x30);
comando_lcd(0x38);
limpa_lcd();//limpa o lcd
```

```
comando_lcd(0x0c);//comando para display sem
//cursor/ver apostilas sobre display

comando_lcd (0x06);

return;//retorna para a função principal (main) - fora
//do while

}

//=======tela principal do display=================

void TELA_PRINCIPAL()

{

LIMPA_LCD();

comando_lcd(0x80);//posiciona o cursor na linha 0 e na
//coluna 1

printf (escreve_lcd, "    GPS-AFTR    ");//escreve HORA
//LONG LAT no LCD

return;//retorna para a função principal (main) - fora
//do while

}

//=============CONGELA TELA==================

void CONGELA()

{

espera:

restart_wdt();
```

```
if(ra2==0) return;
goto espera;
}
//==========VARIAVEIS=======================
void RMC()
{
restart_wdt();
//==========CABEÇALHO=======================
```

(Aqui você está associando as variáveis aos dados do GPS)

```
le0=fgetc(txrx);//$
le1=fgetc(txrx);//G=71
le2=fgetc(txrx);//P=80
le3=fgetc(txrx);//R=82
le4=fgetc(txrx);//M=77
le5=fgetc(txrx);//C=67
leX=fgetc(txrx);//,
//========HORA==============================
le6=fgetc(txrx);//hora
le7=fgetc(txrx);//hora
```

```
le8=fgetc(txrx);//minutos
le9=fgetc(txrx);//minutos
leX=fgetc(txrx);//0
leX=fgetc(txrx);//0
leX=fgetc(txrx);//.
leX=fgetc(txrx);//0
leX=fgetc(txrx);//0
leX=fgetc(txrx);//0
leX=fgetc(txrx);//,
//===============STATUS=====================
le10=fgetc(txrx);//A valido ou V = invalido
leX=fgetc(txrx);//,
//============LATITUDE=======================
le11=fgetc(txrx);//d
le12=fgetc(txrx);//d
le13=fgetc(txrx);//m
le14=fgetc(txrx);//m
le15=fgetc(txrx);//.
le16=fgetc(txrx);//m
le17=fgetc(txrx);//m
```

```
leX=fgetc(txrx);//0
leX=fgetc(txrx);//0
leX=fgetc(txrx);//,
//===========NORTE/SUL=====================
le18=fgetc(txrx);// N = norte ou S = sul
leX=fgetc(txrx);//,
//==============LONGITUDE====================
le19=fgetc(txrx);//d
le20=fgetc(txrx);//d
le21=fgetc(txrx);//d
le22=fgetc(txrx);//m
le23=fgetc(txrx);//m
le24=fgetc(txrx);//.
le25=fgetc(txrx);//m
le26=fgetc(txrx);//m
leX=fgetc(txrx);//0
leX=fgetc(txrx);//0
leX=fgetc(txrx);//,
//===========LESTE/OESTE=====================
le27=fgetc(txrx);//E = leste ou W = oeste
```

```
leX=fgetc(txrx);//0

//==================RMC=====================

if(le1==71 && le2==80 && le3==82 && le4==77 &&
le5==67)

{

//======primeira tela do LCD de dados=============

//===========HORA============================

LIMPA_LCD();

comando_lcd(0x80);

printf (ESCREVE_LCD, "HORA UTC=");//escreve hora

comando_lcd(0x89);//define posição para escrever a
//hora

printf(ESCREVE_LCD,"%1c%1c",le6,le7);//escreve a
//hora com o

comando_lcd(0x8B);

printf (ESCREVE_LCD, ":");//escreve :

comando_lcd(0x8C);//define posição para escrever a
//hora

printf(ESCREVE_LCD,"%1c%1c",le8,le9);//escreve a
//hora com o

//formato: 00:00 ou hora e minuto

//=======STATUS============================
```

```
comando_lcd(0xC0);//define posição do ST=

printf (ESCREVE_LCD, "ST=");//escreve ST=

comando_lcd(0xC3);//define posição para escrever
//estado do status

printf(ESCREVE_LCD,"%1c",le10);//escreve o status

//========L/O================================

comando_lcd(0xC5);//define posição do L/O=

printf (ESCREVE_LCD, "L/O=");//escreve L/O=

comando_lcd(0xC9);//define posição para escrever
//valor do L/O

printf(ESCREVE_LCD,"%1c",le27);//escreve o status

//=========N/S==============================

comando_lcd(0xCB);//define posição do N/S=

printf (ESCREVE_LCD, "N/S=");//escreve N/S=

comando_lcd(0xCF);//define posição para escrever
//valor do N/S

printf(ESCREVE_LCD,"%1c",le18);//escreve o N/S

//=========================================

delay_ms(2000);//intervalo entre uma tela e a outra

if(ra1==0) CONGELA();

LIMPA_LCD();
```

```
//=====segunda tela do LCD de dados=============

//=============LONGITUDE====================

comando_lcd(0x80);//define posição para escrever
LONG=

printf (ESCREVE_LCD, "LONG=");//escreve LONG=

comando_lcd(0x85);//define posição para escrever a
LONGITUDE

printf(ESCREVE_LCD,"%1c%1c%1c%1c%1c%1c%1c%1c",
le19,le20,le21,le22,le23,le24,le25,le26);

//escreve o valor da longitude

//============LATITUDE=======================

comando_lcd(0xC0);//define a posição para escrever
LAT=

printf (ESCREVE_LCD, "LAT=");

comando_lcd(0xC4);//define posição para escrever a
//LATITUDE

printf(ESCREVE_LCD,"%1c%1c%1c%1c%1c%1c%1c",le1
1,le12,le13,le14,le15,le16,le17);

//escreve o valor da latitude

//==========================================

delay_ms(2000);//intervalo entre uma tela e a outra

if(ra1==0) CONGELA();
```

```
}
return;
}
//=========================================
//
//======principal=============================
void main()
{
//======setando as portas=====================
set_tris_a(0b11111111);//todas entradas
set_tris_b(0b11111111);//todas entradas
set_tris_c(0b11111111);//todas entradas
set_tris_d(0b00000000);//todas saídas. Comandará o
//display LCD.
set_tris_e(0b00000100);//o 1 corresponde ao comando
//CS do LCD que deve ser uma
//entrada e ter o valor igual a zero para que possa ser
//escrito no lcd. Quando
//colocamos um pino do pic como entrada ele fica em
//zero. Caso CS tivesse um
//potencial igual a 1 estaríamos lendo o LCD.
```

```
//=====limpando as portas==================

porta=0x00;

portb=0x00;

portc=0x00;

portd=0x00;

porte=0x00;

//========resetando o watch dog================

restart_wdt();

//========programa principal====================

INICIALIZA_LCD();

TELA_PRINCIPAL();//escreve GPS-AFTR no display

//==========loop principal=====================

while(true)

{

restart_wdt();

leX=fgetc(txrx);//aguarda LF=10. Isto sincroniza com as
//mensagens de

//saída do GPS, pois todas terminam em <LF>. O > dá o
//tempo para chamar

//a função.
```

```
if(leX==10) RMC();//chama função RMC para coleta de
//dados do GPS.

if(ra0==0) TELA_PRINCIPAL();//escreve GPS-AFTR no
//display

}

}

//

//Sanctify Yourself

//=========The End==========================
```

Transmissor em 433MHz, usando RS232

A finalidade deste projeto é transmitir um código predeterminado de um PIC, através de um transmissor de 433Mhz. Este código será como um identificador ou uma TAG deste PIC, e se este PIC, com o circuito for instalado em algum lugar, conseguiremos encontrar sempre este lugar ou equipamento. Devido a isto este circuito pode ser alimentado com uma fonte ou com uma pilha de 3,7 volts e alta capacidade de corrente. Tudo nele foi feito de forma a diminuir o preço do circuito, mas o mais importante é o seu aprendizado. Você pode ver que a finalidade deste livro, bem como dos nossos outros de micro controladores, é associar o hardware com o software e mostrar diversos componentes. Veja que neste circuito

você tem um IC LM317 trabalhando como fonte e carregador da pilha de 3,7 volts. Se tiver dúvidas pode comprar este meu livro ou me mandar um e-mail:

https://www.clubedeautores.com.br/backstage/my_books/160588

Observe que com esta tensão o PIC está trabalhando de uma forma “especial”. Veja o código fonte para obter as suas respostas, mas se tiver dúvidas veja:

http://www.luizbertini.net/microcontroladores/microcontroladores.html

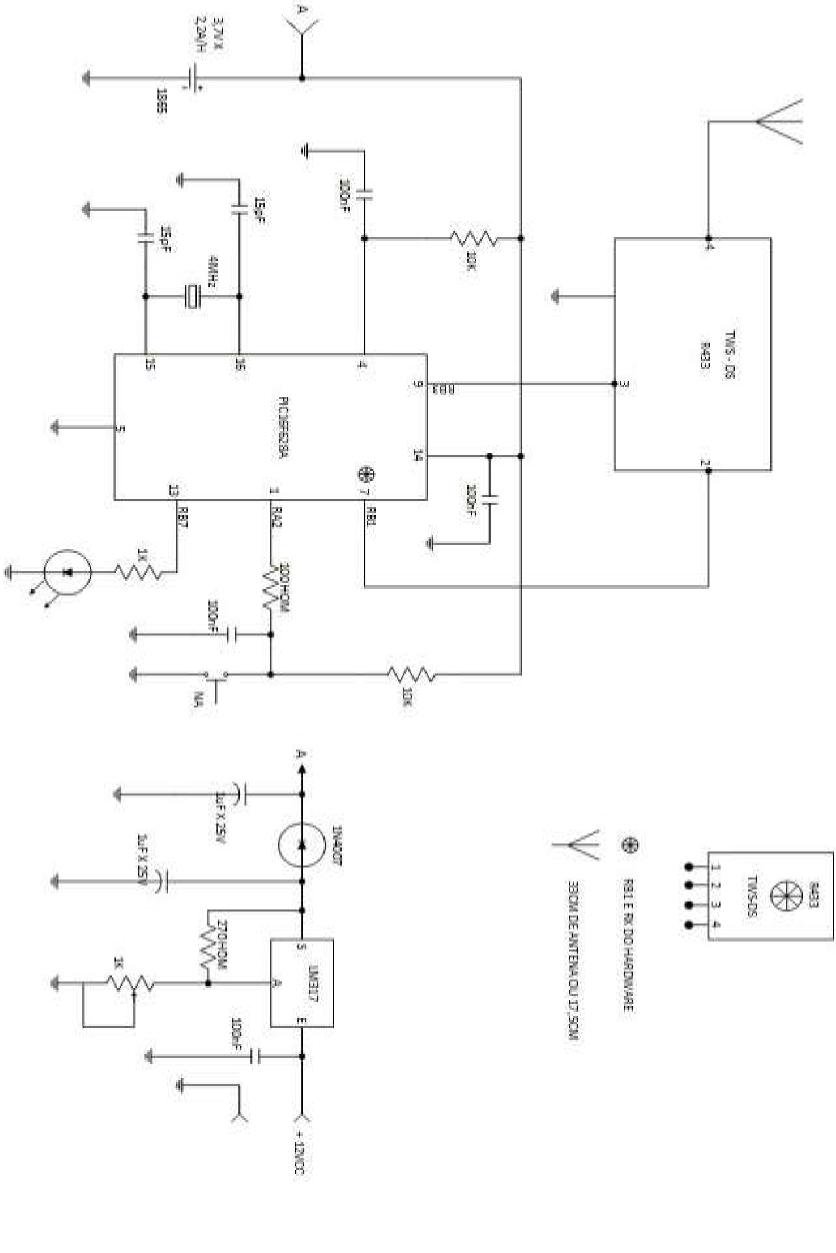
Ou me envie um e-mail:

luizbertini@terra1.com.br

Mas não seja muito apressado com a resposta...

O PIC usa um ressonador de 500KHz, mas você pode usar um cristal de 4MHz ou 20Mhz, mas altere o programa e conserve a “velocidade” de transmissão baixa, pois nestas condições tanto o RX como TX, que fica em outro circuito, apresentaram uma melhor performance na pratica com um maior alcance e menor taxa de erro.

Quando a chave NA1 é pressionada o circuito começa a transmitir e você irá receber estes dados no receptor que é a segunda parte deste projeto está a seguir.


PIC16F628A
TWS - DS
R433
LM317
1N4007
+ 12VDC
10K
1K
100nF
15pF
4MHz
NA
RB1 E RX DO HARDWARE
33CM DE ANTENA OU 17,5CM

Código fonte:

```
//=======================================

/*

Tx para enviar dados via RF.

Luiz Bertini

Luciana Petraites

25/03/2010 - 01/04/2010

*/

//=======================================

//

//============================PIC========

#include <16f628A.h>

//========================FUSÍVEIS========

#fuses
NOPROTECT,NOBROWNOUT,NOCPD,PUT,WDT,NO
LVP,MCLR,XT   //sem wdt diminui corrente

//=====DEFINIÇÃO DE ROTINA DE DELAY=======

#use delay (clock=500000,restart_wdt)
```

```
#use
rs232(baud=300,bits=8,xmit=pin_b1,stream=txrx,p
arity=n,restart_wdt)//300 saida analógica

//=============PORTAS====================

#USE standard_io(a)

#USE standard_io(b)

#byte porta = 0x20

#byte portb = 0x21

//====VARIAVEIS GLOBAIS===============

int chave=0;

int16 vezdia=0;

int16 tempo=4;//ajusta o tempo para enviar
pulsos rápidos

int transmite=0;

//====PISCA LED INDICADOR==================

void LED()

{

output_high(pin_b7);

delay_ms(250);

output_low(pin_b7);
```

```
return;
}
/============FUNÇÃO DE ENVIO=============
void ENVIA()
{
transmite=0;//0
while(transmite<1)//o número depois de
//transmite define precisão=1
{
restart_wdt();
transmite++;
fputc(95,txrx);//início da leitura no RX
fputc(75,txrx);//K - definem dispositivo
fputc(65,txrx);//A - definem dispositivo
fputc(90,txrx);//Z - definem dispositivo
fputc(85,txrx);//U - definem dispositivo
}
return;
}
```

```
//==========PRINCIPAL====================

void main()

{

porta=0x00;//inicialização das portas

portb=0x00;//inicialização das portas

//======LOOP PRINCIPAL===============

while (true)

{

restart_wdt();

//sleep();//faz o circuito dormir e ser mais
economico - importante

//==CONTADOR DE TEMPO==================

vezdia++;

delay_ms(1000);/*define tempo para enviar pulso
- importante

1 segundo = 18 horas, para um valor de vezdia =
65000, ou seja, pode definir

quantos vezes envia código por dia. Para 24 horas
o valor do delay deve se de

1329 */
```

(Você pode mudar estes valores para mais de uma transmissão por dia)

```
//=======================================

chave=input(pin_a2);

if(chave==0) vezdia=tempo;//para manter os dois
//números iguais

if(vezdia==tempo)//para manter os dois números
//iguais

{

vezdia=0;

output_high(pin_b3);

delay_ms(100);//levar este tempo em
consideração

ENVIA();

output_low(pin_b3);

LED();

transmite=0;

//vezdia é uma variavel que define 65000 este
//número deve ser diferente para

//cada TX. O valor deste número junto com o valor
//da variável dia define de
```

```
//quantas em quantas horas é enviado o pulso
}
}
}
/*Thanks God.
Santicfy Yourself.*/
//====THE END====================
```

Receptor de 433MHz usando RS232

A proposta deste circuito é ler um sinal recebido na frequência de 433MHz e decodifica-lo, salvando-o em uma memória eeprom externa, e depois, quando a chave NA for pressionada, ele enviara estes dados via RS232, para um modulo da wiznet e este modulo poderá enviar estes dados para uma rede ethernet.

O modulo utilizado foi o WIZ100SR, mas você pode usar outros. Nos testes usamos uma conexa0 de rede com o Windows XP e Windows & (marcas registradas), junto com o hiperterminal (marca registrada) para receber os dados gravados. Este modulo necessita de uma certa configuração que é muito fácil de fazer, principalmente se você tiver o datasheet dele.

Esta unidade de recepção, pode receber dados de diversos transmissores, conforme o explicado anteriormente.

A nossa idéia básica é mostrar para você como gravar em um eeprom externa, como usar a comunicação RS232 ou EIA232 do PIC e como converter isto para Rede através de um modulo.

Se você tiver dúvidas mande um e-mail (ele já está aí pelo livro, mas colocarei tudo no final novamente).

Use o código fonte pronto ou faça alterações e aprenda mais com isto.

Datasheets destes componentes você poderá encontrar em:

http://www.luizbertini.net/microcontroladores/microcontroladores.html

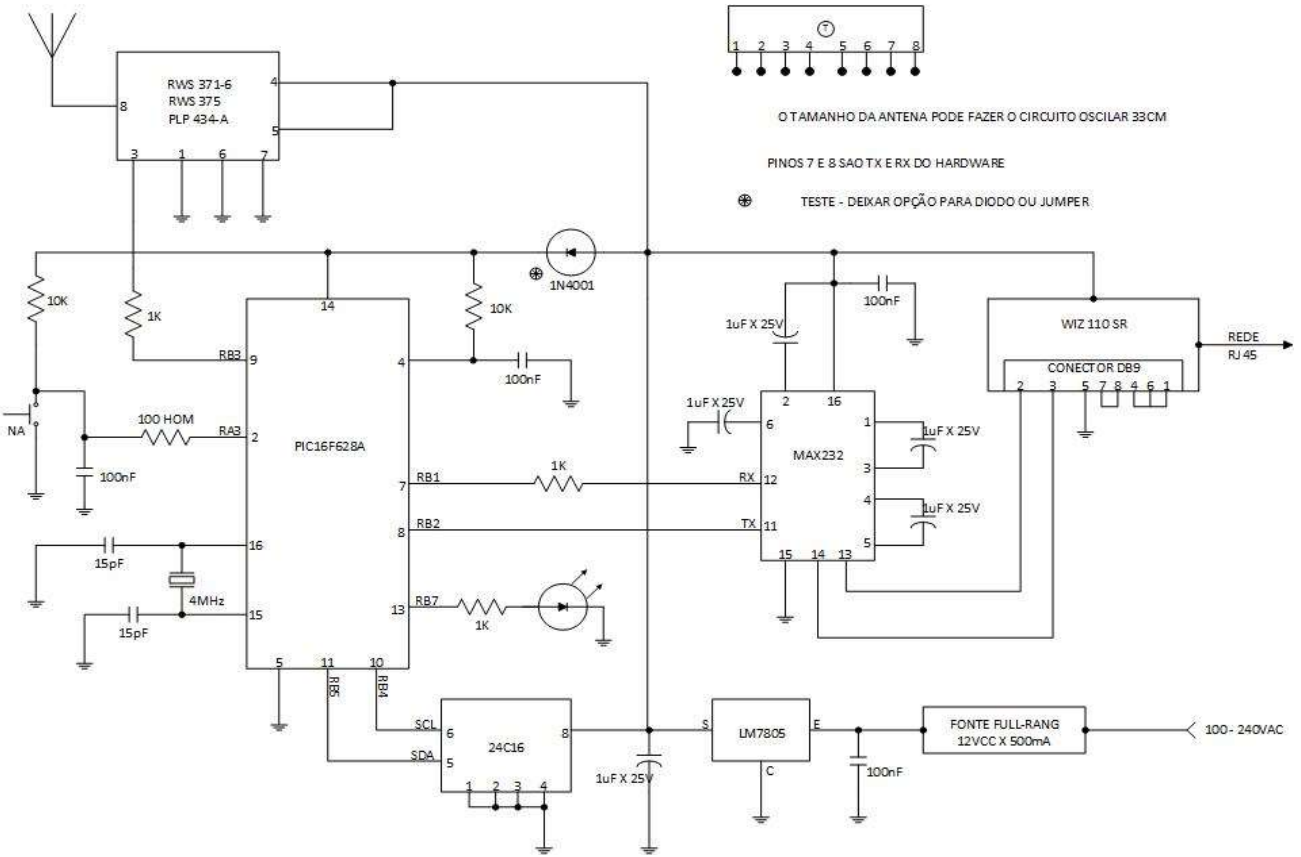
Ou no próprio sites deles.

Você pode perguntar para que serve este circuito???

Ele serve para você controlar equipamentos dentro de uma área, você coloca vinte transmissores em um local, dentro de equipamentos, e monitora com esta central, através da rede ethernet. Você faz uma leitura no início do dia e outra no final, por exemplo, e com isto você pode gerar um relatório se os equipamentos ainda estão na sala ou “fugiram”...

Caso você tenha diversos setores com RXs separados, você saber aonde ele “apareceu”...

Isto é apenas uma aplicação e uma idéia, desenvolva as suas.


RWS 371-6
RWS 375
PLP 434-A
O TAMANHO DA ANTENA PODE FAZER O CIRCUITO OSCILAR 33CM
PINOS 7 E 8 SAO TX E RX DO HARDWARE
TESTE - DEIXAR OPÇÃO PARA DIODO OU JUMPER
1N4001
10K
1K
100nF
1uF X 25V
WIZ 110 SR
REDE
RJ 45
CONECTOR DB9
NA
100 HOM
PIC16F628A
MAX232
RB1
RB2
RB7
RB5
RB4
RX
TX
15pF
4MHz
24C16
SCL
SDA
LM7805
FONTE FULL-RANG
12VCC X 500mA
100 - 240VAC

Código fonte:

```
//====================================

/*

Receptor de RF para código ASCII

Luiz Bertini

Luciana Petraites

04/03/2010

*/

//=======================================

//

//==================PIC==================

#include <16f628A.h>

//==================FUSÍVEIS==============

#fuses
NOPROTECT,BROWNOUT,NOCPD,PUT,WDT,NOLVP
,MCLR,XT

//=====DEFINIÇÃO DE ROTINA DE DELAY=======

#USE DELAY (CLOCK=4000000, RESTART_WDT)
```

```
#use
rs232(baud=300,bits=8,rcv=pin_b3,stream=txrx,pa
rity=n,restart_wdt)/*300 bps

Devido a saída analógica. Entrada de dados via
RF*/

#use
rs232(baud=9600,bits=8,xmit=pin_b2,rcv=pin_b1,s
tream=saida,parity=n,restart_wdt)/*

Conversa com o PC ou a rede*/

//=================PORTAS================

#USE standard_io(a)

#USE standard_io(b)

#byte porta = 0x20

#byte portb = 0x21

//========I2C============================

#use i2c (master, sda=pin_b5, scl=pin_b4, slow,
restart_wdt, force_hw)

//============CONSTANTES================

#define le                0b10100001//byte de leitura
//para a memória eeprom

#define esc               0b10100000//byte de
//escrita na memória eeprom
```

```
//=============VARIAVEIS GLOBAIS==========
int le8=0;
int le9=0;
int le10=0;
int le11=0;
int leX=0;
int leT=0;
int RTS=0;
int CRTS=0;
int final=0;
//=======================================
int end=0;
int dado=0;
//=================FUNÇÕES==============
//
//======PISCA LED INDICADOR===============
void LED()
{
output_high(pin_b7);
```

```
delay_ms(250);

output_low(pin_b7);

return;

}

//======ESCREVE EEPROM EXTERNA==========

ESCREVE_EEPROM_EXT()

{

i2c_start();//comando para iniciar comunicação com eeprom externa

i2c_write(esc);//envia constante que corresponde a escrita

i2c_write(end);//envia o endereço onde deve ser escrito

i2c_write(dado);//envia o dado a ser escrito/verificar o nome deste dado

i2c_stop();//termina a comunicação de escrita

delay_ms(10);//aguarda 10 milisegundos para completar a gravação

return;//volta para o programa principal

}

//======LEITURA DA EEPROM EXTERNA========
```

```
LEITURA_EEPROM_EX()

{

i2c_start();//comando para iniciar comunicação com eeprom externa

i2c_write(esc);//envia constante que corresponde a escrita

i2c_write(end);//envia o endereço onde deve ser escrito

i2c_start();//começa novamente

i2c_write(le);//envia constante que corresponde a leitura

dado = i2c_read(0);//lê o dado da eeprom

i2c_stop();//termina leitura

return;//volta para o programa principal

}

//=======CONVERSÃO=====================

void GRAVA_EEPROM()//fazer com que a gravação seja continua nos endereços

{

end++;

dado=le8;
```

```
ESCREVE_EEPROM_EXT();
//=======================================
end++;
dado=le9;
ESCREVE_EEPROM_EXT();
//=======================================
end++;
dado=le10;
ESCREVE_EEPROM_EXT();
//=======================================
end++;
dado=le11;
ESCREVE_EEPROM_EXT();
//===========FINAL DE 4 BYTES=============
end++;
dado=95;//divisor de stream de 4 bytes
ESCREVE_EEPROM_EXT();
//=======================================
LED();
```

```
final++;
return;
}
//=========APAGA EEPROM================
void APAGA()
{
end=0;
while(true)
{
restart_wdt();
end++;
dado=0xFF;
ESCREVE_EEPROM_EXT();
LED();
if(end==32) {final=0; end=0; return;}//escopo
}
return;
}
//=======================================
```

```
void ENVIO_EEPROM()//envia dados da eeprom

{

end=0;//o endereço deve ser zerado para fazer a leitura da eeprom do começo

while(end<32)//quantidade de dispositivos armazenados

{

restart_wdt();

end++;//incrementa endereço

LEITURA_EEPROM_EX();//faz leitura da eeprom

fputc(dado,saida);//envia dados para placa wiznet mandar para rede

LED();

}

APAGA();//só pode ir para o apaga se tiver dados validos gravados-tem que ser aqui

return;

}

//====================PARA PC=============

void PC()//testar com PC e wiznet na bancada, depois pode tirar. Funcionou
```

```
//perfeitamente. O sinal chega limpo, sem
interferências ou bytes errados.

//para tirar esta função basta ir na função principal
e fazer //PC();

{

fputc(le8,saida);//K

fputc(le9,saida);//A

fputc(le10,saida);//Z

fputc(le11,saida);//U

LED();

return;

}

//================LEITURA================

void LEITURA()

{

restart_wdt();

le8=fgetc(txrx);//K = 75

le9=fgetc(txrx);//A = 65

le10=fgetc(txrx);//Z = 90

le11=fgetc(txrx);//U = 85
```

```
if(le8==75 && le9==65 && le10==90 && le11==85)
LED();//Desta forma voce tem

//certeza do que recebe. Serve apenas para teste.
OK-20/04/10

return;

}

//===============PRINCIPAL===============

void main()

{

porta=0x00;

portb=0x00;

while (true)

{

output_low(pin_b0);

restart_wdt();

RTS=input(pin_b6);//pino12 do PIC

//=====================================

if(final<=5)//define número de dispositivos
gravados tem que colocar final=0 em algum lugar

{
```

```
leX=fgetc(txrx);

if(leX==95) LEITURA();//95 é igual a underline = _

delay_ms(300);//teste para visualizar led

PC();//Apenas para teste - Funcionou
perfeitamente - 19/04/10

delay_ms(300);//teste para visualizar led

GRAVA_EEPROM();//manda escrever os dados na
eeprom

}

//====================================

if(RTS==1) CRTS=1;

if(CRTS==1)

{

leT=fgetc(saida);

if(leT==89) ENVIO_EEPROM();//89 = Y é o
comando para chamar a base, ela já é

//identificada pelo IP da placa, etc e tal. Também
lê os dados da eeprom

CRTS=0;

}
```

```
}
}
/*Thanks God.
Santicfy Yourself.*/
//=============THE END====================
```

Sites interessantes para obter informações:

Informações minhas:

http://www.luizbertini.net/microcontroladores/microcontroladores.html

http://www.clubedeautores.com.br/book/158866--Microcontroladores_PIC_16F62816F628A_em_Assembly?topic=eletronica#.VaE1jE3bLIU

Compilador da CCS:

http://www.acepiccamp.com.br/compilador-linguagem-c-ccs-pcwhd-p65?gclid=Cj0KEQjw_YKtBRC7zZjFp8bF_foBEiQAfyigc10LWgWHXfTjs_wlZW1OdfHiLCei3hWNHl8ASIS7f2AaAmi_8P8HAQ

http://www.ccsinfo.com/

MPLAB:

http://www.microchip.com

Gravadores:

http://www.mosaico.com.br/

http://newportcom.com.br/

Componentes em geral:

http://luizbertini2007.mercadoshops.com.br/

Procure ou pergunte.

Contato comigo para sugestões, duvidas ou críticas:

luizbertini1@terra.com.br

Todas as marcas e fornecedores citados nos textos e códigos fontes tem direitos autorais e sua citação aqui é apenas para uso didático.

Luiz Bertini

Sanctify Yourself.

The End.