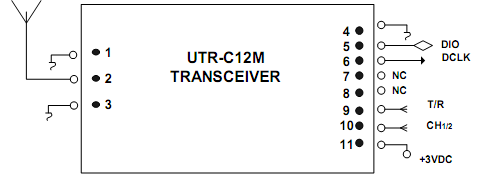
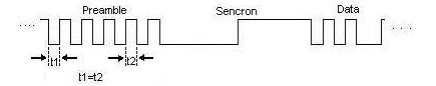


UMD-B12 – RF MODEM BOARD



Sistem açıldıktan sonra işlemci RS232’ye <READY> sinyali yollar ve Alıcı(Reciever) modunda bekler. Modül açıldığından itibaren Alıcı modunda olduğu zaman sürekli gürültü alır.Istenilen data ile gürültüyü ayırmak için preamble ve sencron verileri kullanılır. Preamble veri olarak ardışık 1 ve 0 lardan oluşan (01010101...) bir bit dizinidir. 5 byte 0x55 veya 0xAA olabilir. Gönderilen 1 ve 0’ların süreleri eşit olmalıdır. Sencron ise yazılımın senkronizasyonuna yardımcı olur. Bit senkronizasyonunun sağlanması ve mesaj başlangıcının doğru tayini için kullanılması gereklidir. Bu bit dizininin boyu uygulama gereksinimleri veya kısıtlamalarına göre değişebilmekle birlikte 5 byte 0x00 + 5 byte 0xFF olabilir veya bunun ne olacağına kulanıcı kendisi karar verebilir.Data gönderirken araya boşluk girmemeli, girer ise tekrar preamble ve sencron gönderilmeli. Alıcı(Reciever) tarafında preamble’a bakılmaz. Sadece sencron aranır, sonrasında data okunur.



Sencron alındıktan data alınmaya başlanır, alınan data mutlaka “<” ile başlayıp “>” ile bitmelidir. Data “<” ile başlamazsa sencron alınmışsa bile gelen datayı almaz ve tekrar sencron bakmaya devam eder. Data “>” ile bitmez ise gelen serinin bitmediğini düşünür, ve <TIME OUT> hatası verir. Düzgün şekilde modülden datayı aldıktan sonra işlemci bu datayı RS-232’ye yollar. RS-232’den data geldiğinde aynı şekilde gelen datanın “<” ile başlayıp “>” ile bitmesi gerekmektedir. RS-232’den data geldiğinde işlemci gelen datayı almaya başlar. Data alımı tamamlandığında işlemci modülü transmitter moda sokar ve veri yollamaya başlar. Yollanan veri öncelikle sencron, daha sonra gönderilecek toplam byte sayısı, daha sonra “<xxxxxx>” şeklinde data ve en son olarak da checksum’dır. Alıcı(Reciever) taraf bu şekilde datayı aldığı zaman kendisi aldığı byteları ve checksum’ı ayrıca hesaplar ve sonra karşı taraftan alınan byte sayısı ile checksum değerlerini kendi hesapladıkları ile karşılaştırır. Eğer sayılar birbirine eşit ise, RS-232’ye aldığı verileri yollar ve ardından ready yollar.

RS-232

RS-232

İşlemci

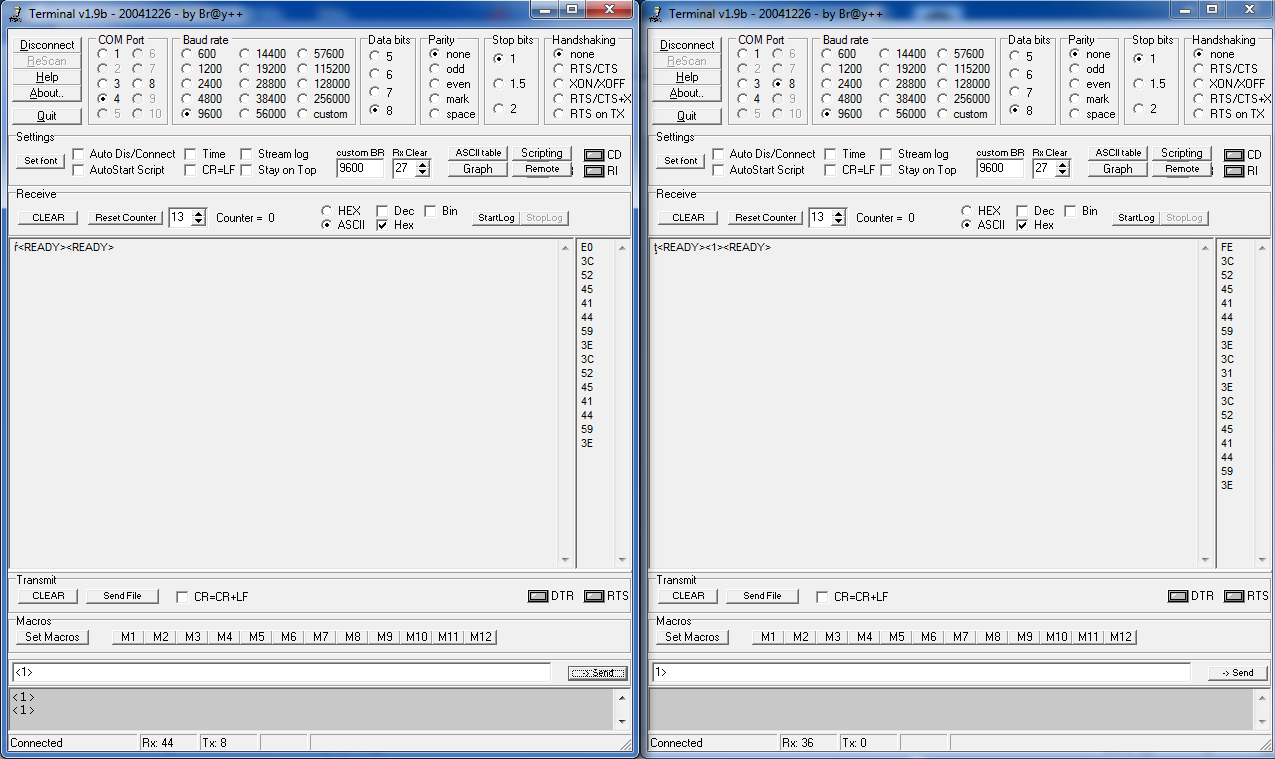
İşlemci

UTR-C12

UTR-C12

Basitçe iki modem arasındaki işlem yukardaki gibidir.

RS-232’ye gelen datayı görmek için “Terminal” adlı program kullanılır. Bilgisayara bağlı olan port seçilir ve Baud Rate, Data Bits, Parity ve Stop Bits ayarları yapıldıktan sonra “Connect”Aşağıdaki resimde iki modemde aynı bilgisayara bağlanmış olup soldakinden <1> yollanmıştır ve sağda alınan tarafta görünmektedir.



UTR-C12M modül 2 farklı frekans kanalı seçeneği sunmaktadır. Bu kanallara geçebilmek için <C0> veya <C1> komutları gönderilmelidir. 0 veya 1 dışındaki herhangi bir rakamı işlemci kabul etmeyecektir ve <ERROR> hata mesajı verecektir. İşlemcinin buffer’ı 40 byte olduğundan, 40dan fazla karakter yollanmaya çalışıldığında işlemci <OVERFLOW> hatası verecektir.

Aşağıdaki yazılım yukarda anlatılan işlemleri gerçekleştirmektedir. Yazılımın kullandığı protokol aşşağıdaki gibidir.

* Baud Rate: 9600
* No Parity
* 8 Bit Data
* Stop Bits: 1

#include <pic.h>

#include "delay.c"

void InitBoardReciever();

void InitBoardTransmitter();

void InitBoard();

void SendData(char);

void SendSencron();

void EepromWrite(char);

char EepromRead();

char DataRead();

int Sencron();

void SendReady();

void Error();

void TimeOut();

void Overflow();

void Reset();

/\*Hata mesajları\*/

char Ready[]= "<READY>";

char Ovf[]= "<OVERFLOW>";

char TimeO[]= "<TIME OUT>";

char Err[]= "<ERROR>";

char RecievedChar[50]; //buffer

int i=0,TransmitDataReady=0,k;

char CheckSum;

char ByteCount; //Alınan toplam byte sayısı

int StartRecieve=0;

int main()

{

CLRWDT();

int j;

char temp;

InitBoard();

SendReady();

while(1)

{

while(!Sencron()) //Senkron bakar

{

CLRWDT();

if(TransmitDataReady) //Eğer data yollanmaya hazırsa senkron

//bakmayı keser,senkron bakması recieve //için gereklidir

break;

}

if(TransmitDataReady) //Data hazır mı diye bakar, hazırsa yollar.

{

CLRWDT();

InitBoardTransmitter(); //kart sürekli reciever modda olduğu için //yollamadan önce transmittera alır.

CheckSum=0;

TransmitDataReady=0;

SendSencron(); //Preamble+Senkron yollar

SendData(i); //Toplam Byte sayısını yollar

for(j=0;j<i;j++) //Sonra Datayı yollamaya başlar

{

if(RecievedChar[j]!='>') //Sona gelmediyse normal yollama //işlemini yapar,her yolladığı karakteri //Checksum ile toplar

{

SendData(RecievedChar[j]);

CheckSum=CheckSum+RecievedChar[j];

}

else if(RecievedChar[j]=='>') //Sona geldiyse sonuncu karakteri yollar,

//sonrada Checksum modüle yollar, seri

//portada ready yollar

{

CheckSum=CheckSum+RecievedChar[j];

SendData(RecievedChar[j]);

SendData(CheckSum);

SendReady();

}

}

CLRWDT();

i=0;

InitBoardReciever(); //Sadece Transmit Data hazırsa tx moduna

//girer,normalde hep rx modunda bekler.

}

else

{

RCIE=0;

CheckSum=0;

CLRWDT();

do

{

temp=DataRead();

if(i==0)

ByteCount=temp; //ilk aldığı karakter toplam byte sayısı

//olduğu için

//checksum a dahil etmez.

else //Son karakteri alana kadar data almaya

//devam eder.

{

CheckSum=CheckSum+temp; RecievedChar[i]=temp;

}

i++;

}while(temp!='>' || i==1); //">" işaretinin ascii kodu 62 olduğundan,

// byte sayısı olarak 62 aldığında

//durmaması için i==1 de loopu kesmez.

RecievedChar[i]=DataRead(); //Son karakteri de aldıktan sonra

//Checksum almak için bir karakter daha

//alır.

CLRWDT();

if(CheckSum==RecievedChar[i]) //Alınan checksum ile hesaplanan

//Checksum eşit mi diye bakar.

{

if(ByteCount==(i-1)) //Karşı taraftan yollanan toplam byte

//sayısı ile kendi hesapladığı byte sayısını //karşılaştırır.

{

//eğer checksum ve byte sayıları

//tutarsa,rs-232’ye transmit

//işlemine başlar.

for(j=1;j<i;j++)

{

while(!TXIF && !TRMT); //Transmit işlemi bitene kadar, yani tx

//ready flagi 1 olana kadar bekler.

TXREG=RecievedChar[j]; //seri porta tektek aldığı karakterleri basar.

}

SendReady(); //gönderme işlemi bittikten sonra seri

//porta ready basar.

ByteCount=0; //bir sonraki işlem için byte sayısını 0

//yapar.

}

}

i=0;

RCIE=1; //seri porta transmit işlemi bittiği için

//recieve interruptu aktif eder

}

}

}

void InitBoard()

{

CLRWDT();

OPTION=0x8E;

TRISA=0x03;

PORTA=0x00;

RA5=EepromRead(); //Kanal Seçmek için eeprom u okur.

ADCON1=0x07;

TRISC6=1;

TRISC7=1;

GIE=1;

PEIE=1;

BRGH=1;

RCIE=1;

TXIE=0;

SPBRG=25;

SYNC=0;

TXEN=1;

CREN=1;

SPEN=1;

T1CON = 0b00110000;

TMR1IE=0;

}

void InitBoardTransmitter()

{

CLRWDT();

TRISA=0x02;

RA4=1; //4. bacak modülün tx/rx mod seçim

//bacağı olduğundan, modülü transmitter

//moda alır.

RCIE=0;

}

void InitBoardReciever()

{

CLRWDT();

TRISA=0x03; //4. bacağı 0 yaparak modülü reciever

//moduna alır.

PORTA=0x00;

RCIE=1;

}

void SendData(char RF1)

{

int Databit=8;

char TestBit;

CLRWDT();

while (Databit!=0) //8 bit(1 byte) yollar

{

while (RA1);

TestBit = RF1 & 0x80;

RF1 = RF1 << 1;

if (TestBit == 0x80)

RA0 = 1;

else

RA0 = 0;

while(!RA1);

Databit--;

}

}

void SendSencron() //senkron+preamble yollar

{

CLRWDT();

SendData(0x55);

SendData(0x55);

SendData(0x55);

SendData(0x55);

SendData(0x55);

SendData(0x00);

SendData(0x00);

SendData(0x00);

SendData(0x00);

SendData(0x00);

SendData(0xFF);

SendData(0xFF);

SendData(0xFF);

SendData(0xFF);

SendData(0xFE);

CLRWDT();

}

void EepromWrite(char data) //eeproma bilgi yazar.

{

CLRWDT();

while(WR);

EEADR=0xFF;

EEDATA = data;

EEPGD = 0;

WREN = 1;

GIE = 0;

EECON2 = 0x55;

EECON2 = 0xaa;

WR = 1;

WREN = 0;

GIE = 1;

EEIF = 0;

}

char EepromRead() //eeprom u okur

{

CLRWDT();

EEADR = 0xFF;

EEPGD = 0;

RD = 1;

return EEDATA;

}

char DataRead()

{

int k=0;

char RF0=0;

while(k!=8)

{

while(!RA1);

RF0=RF0<<1;

if (RA0)

RF0 = RF0 | 0x01;

else

RF0 = RF0 & 0xFE;

while(RA1);

k++;

}

CLRWDT();

return RF0;

}

int Sencron() //gelen datanın senkron olup olmadığına

//bakar.

{

int high=0, low=0, i ,j;

for(j =0; j < 2; j++)

{

i = 0;

while(i != 250)

{

while(!RA1);

i++;

if (!RA0)

{

if (high == 39)

{

while(RA1);

return 1;

}

high = 0;

low++;

}

else

{

if (low != 40)

{

low = 0;

}

high++;

}

while(RA1);

}

}

CLRWDT();

return 0;

}

void Reset() //watchdog timer'ı clear etmeyerek

//işlemciyi resetler.

{

PSA=0;

PS2=0;

PS1=0;

while(1);

}

void SendReady() //RS232'ye <READY> mesajı yollar.

{

int x;

x=RCIE;

RCIE=0;

for(k=0;k<7;k++)

{

CLRWDT();

while(!TXIF && !TRMT);

TXREG=Ready[k];

}

RCIE=x;

}

void Error() //Seri porta Genel Hata

//mesajı(<ERROR>) yollar.

{

for(k=0;k<7;k++)

{

CLRWDT();

while(!TXIF && !TRMT);

TXREG=Err[k];

}

TMR1IF=0;

Reset();

}

void Overflow() //40 den fazla byte girildiğinde seri porta

//hata(<OVERFLOW>) yollar

{

for(k=0;k<10;k++)

{

CLRWDT();

while(!TXIF && !TRMT);

TXREG=Ovf[k];

}

TMR1IF=0;

Reset();

}

void TimeOut() //iki byte arasında 500ms den fazla süre

//geçerse seri porta hata(<TIME OUT>) //basar.

{

for(k=0;k<10;k++)

{

CLRWDT();

while(!TXIF && !TRMT);

TXREG=TimeO[k];

}

TMR1IF=0;

Reset();

}

void interrupt recieve()

{

char temp;

if(i>40)

Overflow();

CLRWDT();

if(RCIE && RCIF) //Yeni data için mi interrupta girildiğini

//kontrol eder.

{

RCIE=0;

temp=RCREG;

TMR1IE=1; //seri porttan datayı aldığı için timer

//interruptları aktif eder.

if(temp=='<' && StartRecieve==0) //Data "<" ile başlıyor mu diye kontrol

//eder.

StartRecieve=1;

if(!StartRecieve) //Başlamıyorsa hata verir.

Error();

else if(StartRecieve) //Başlıyorsa datayı almaya başlar.

{

CLRWDT();

TMR1H=0; //Data almaya başladığı için timer'ı sıfırlar

//ve zaman tutmaya başlar.

TMR1L=0;

TMR1ON=1;

if(RecievedChar[i-1]=='C') //gelen datanın kanal değiştirme komutu

//olup olmadığına bakar.

{

TMR1ON=0;

if(temp != 0x30 && temp != 0x31) //kanal değiştirme komutundan

//bir sonraki komuta bakar, alınan değer 0 //veya 1 değilse,

Error(); //istenilen kanal numarası mevcut

//olmadığı için hata verir.

else //Hata yoksa kanalı değiştirir, yeni kanal

//numarasını eeproma kaydeder.

{

if(temp==0x30) //eğer gelen sayı 0 ise, eeproma 0 yazar,

{

TXREG='0';

EepromWrite(0x00);

}

else

{

TXREG='1'; //eğer gelen sayı 1 ise, eeproma 1 yazar.

EepromWrite(0xFF);

}

RA5=EepromRead(); //5. bacak kanal seçme bacağı olduğundan

//eepromdaki değeri 5. bacağa atar.

CheckSum=CheckSum-RecievedChar[i-1];

//Checksum a kanal değiştirme komutunu //dahil etmemk için kanal değiş komutunu //checksumdan çıkartır.

i=i-1; //kanal değiştirme komutunu toplam byte

//sayısına dahil etmemek için 1 çıkartır.

}

}

else //kanal değiştirme komutu değilse, normal

//aldığı datayı buffera yazar.

{ //aynı zamanda iki data alışı arasındaki

//süreyi ölçmek için zaman tutar.

TMR1H=0;

TMR1L=0;

RecievedChar[i]=temp;;

if(temp == '>') //eğer son karakter ise data gönderilmeye

//hazır olduğu için TransmitDataReady'i 1 //yapar.

{

TMR1IE=0; //Son karakteri aldığı için timer

//interruptları kapatır.

TMR1ON=0;

TransmitDataReady=1; //Datanın transmite hazır olduğunu

//belirtir.

StartRecieve=0; //Bir sonraki data almaya başladığında

//tekrardan ilk karakterin < olmasını //düzgün kontrol edebilmek için sıfırlar.

}

i++;

}

}

CLRWDT();

RCIE=1; //recieve interruptu tekrar aktif eder.

RCIF=0; //interrupt flagini siler.

}

else if(TMR1IF) //eğer interrupta zaman aşımından dolayı

//girdiyse, time out hatası verir.

{

TMR1ON = 0;

TimeOut();

}

}