# C FILE

#include <RECIEVERREAL.h>

#include <math.h>

unsigned char ItStatus1,ItStatus2;

unsigned char rx\_buf[9];

void SI4432\_RX\_init\_parameter(void);

void rx\_data();

unsigned char spi\_writex(unsigned char addr, unsigned char data);

unsigned char spi\_readx(unsigned char addr );

unsigned char spi\_R(unsigned char data);

void main()

{

unsigned char i;

set\_tris\_d(0b00000001);

set\_tris\_b(0b00000000);

set\_tris\_c(0b00010000);

SI4432\_RX\_init\_parameter();

spi\_writex(0x0e|0x80, 0x00); // TX0\_RX0 Antenna is not in Tx/Rx mode

while(TRUE)

{

delay\_ms(100);

rx\_data();

while(input(nIRQ));

if(!input(nIRQ))

{

for(i=0;i<5;i++)

{

output\_high(LEDB5);

delay\_ms(200);

output\_low(LEDB5);

delay\_ms(200);

}

ItStatus1 = spi\_readx(0x03); // read status, clear interrupt

ItStatus2 = spi\_readx(0x04);

output\_low(SCLK);

output\_low(nSEL); // must be low during the addr write

spi\_R(0x7f);

for( i=0;i<9;i++)

{

rx\_buf[i]=spi\_R(0x00);

}

output\_high(nSEL);

spi\_writex(0x07, SI4432\_PWRSTATE\_READY); // Exit Rx mode after all the data read from the FiFo

// chksum = 0;

// for(i=0;i<3;i++) // calculate the checksum for the received data

//{chksum += rx\_buf[i];}

//if(( chksum== rx\_buf[3] )&&( rx\_buf[0] == 0x41 ))

//! if(rx\_buf[0] == 0x41 )

//! {

output\_high(LEDB5);

delay\_ms(30);

for(i=0;i<9;i++)

{

printf("data is= %c \n ",rx\_buf[i]);// data verified OK

}

// }

//! else

//! {

//! printf("Data Verified error, then restart to Rx");

//! for(i=0;i<7;i++)

//! {

//! output\_high(LEDB5);

//! delay\_ms(50);

//! output\_low(LEDB5);

//! delay\_ms(50);

//! }

//! rx\_data(); // Data Verified error, then restart to Rx

//! }

}

}

}

void SI4432\_RX\_init\_parameter(void)

{

unsigned char fb=0x53,fc0=0x64,fc1=0x00; // //434 mhz

ItStatus1 = spi\_readx(0x03); // read status, clear interrupt

ItStatus2 = spi\_readx(0x04);

spi\_writex(0x06, 0x00); // no wakeup up, lbd,

spi\_writex(0x07, SI4432\_PWRSTATE\_READY); // disable lbd, wakeup timer, use internal 32768,xton = 1; in ready mode

spi\_writex(0x09, 0x7f); // c = 12.5p

spi\_writex(0x0a, 0x05);

spi\_writex(0x0b, 0xf4); // gpio0 rx data output

spi\_writex(0x0c, 0xfd); // gpio vdd

spi\_writex(0x0d, 0xfd); // gpio vdd

spi\_writex(0x0e, 0x00); // gpio 0, 1,2 NO OTHER FUNCTION.

spi\_writex(0x70, 0x20); // disable manchest

spi\_writex(0x1d, 0x00); // enable afc

spi\_writex(0x1c, 0x1d); // RATE\_24K: // 2.4k

spi\_writex(0x20,0x41);//0x20 calculate from the datasheet= 500\*(1+2\*down3\_bypass)/(2^ndec\*RB\*(1+enmanch))

spi\_writex(0x21, 0x60); // 0x21 , rxosr[10--8] = 0; stalltr = (default), ccoff[19:16] = 0;

spi\_writex(0x22, 0x27); // 0x22 ncoff =5033 = 0x13a9

spi\_writex(0x23, 0x52); // 0x23

spi\_writex(0x24, 0x00); // 0x24

spi\_writex(0x25, 0x06); // 0x25

spi\_writex(0x2a, 0x1e);

spi\_writex(0x30, 0x8c); // enable packet handler, msb first, enable crc,

// 0x31 only readable

spi\_writex(0x32, 0xff); // 0x32address enable for headere byte 0, 1,2,3, receive header check for byte 0, 1,2,3

spi\_writex(0x33, 0x42); // header 3, 2, 1,0 used for head length, fixed packet length,synchronize word length 3, 2,

spi\_writex(0x34, 64); // 64 nibble = 32byte preamble

spi\_writex(0x35, 0x20); //0x35 need to detect 20bit preamble

spi\_writex(0x36, 0x2d); // synchronize word

spi\_writex(0x37, 0xd4);

spi\_writex(0x38, 0x00);

spi\_writex(0x39, 0x00);

spi\_writex(0x3f, 's'); // check hearder

spi\_writex(0x40, 'o');

spi\_writex(0x41, 'n');

spi\_writex(0x42, 'g');

spi\_writex(0x43, 0xff); // all the bit to be checked

spi\_writex(0x44, 0xff); // all the bit to be checked

spi\_writex(0x45, 0xff); // all the bit to be checked

spi\_writex(0x46, 0xff); // all the bit to be checked

// 0x56 ---------0x6c ??????????????????????????

spi\_writex(0x79, 0x0); // no hopping

spi\_writex(0x7a, 0x0); // no hopping

spi\_writex(0x71, 0x22); // Gfsk, fd[8] =0, no invert for Tx/Rx data, fifo mode, txclk -->gpio

spi\_writex(0x73, 0x0);

spi\_writex(0x74, 0x0); // no offset

//band 434:

spi\_writex(0x75, fb); // hbsel = 0, sbsel =1 ???, fb = 19

spi\_writex(0x76, fc0); // 25600= 0x6400 for 434Mhz

spi\_writex(0x77, fc1);

}

void rx\_data()

{

spi\_writex(0x07|0x80, SI4432\_PWRSTATE\_READY); // enter Ready mode

delay\_ms(5); // stablize the OSC; not needed if OSC is on

spi\_writex(0x0e|0x80, 0x02);//TX0\_RX1 antenna switch = Rx mode

spi\_writex(0x08, 0x03); //clear Tx/Rx fifo

spi\_writex(0x08, 0x00); //clear Tx/Rx fifo WE MUST SET TO THE DEFAUT VALUES=0

spi\_writex(0x07,SI4432\_PWRSTATE\_RX ); // enter Rx mode //AUTOMATICALLY CLEARED WHEN A VALID PACKET RECıEVED

spi\_writex(0x05, SI4432\_Rx\_packet\_received\_interrupt); // interrupt for packet received

//when we write interrupt in 0x05 register we must read status registers(03h,04h) in order to enable nIRQ=0 pin.

//after that we must read status registers for disable the interrupts.

ItStatus1 = spi\_readx(0x03); // read status, clear interrupt

ItStatus2 = spi\_readx(0x04);

}

unsigned char spi\_writex(unsigned char addr , unsigned char data )

{

unsigned char i;

//make sure to use addr|0x80 in the argument of addr for writing mode

addr|=0x80;

output\_low(SCLK);

output\_low(nSEL); // must be low during the data read write

for(i=0;i<8;i++)

{

if(addr&0x80)

output\_high(SDO);

else

output\_low(SDO);

addr<<=1;

output\_high(SCLK);

delay\_cycles(1);

output\_low(SCLK);

}

for(i=0;i<8;i++)

{

if(data&0x80) //if bit of data is 1

output\_high(SDO);

else //if bit of data is 0

output\_low(SDO);

data<<=1;

output\_high(SCLK);

delay\_cycles(1);

output\_low(SCLK);

}

output\_high(nSEL); // must be high after finish Writing addr and data

output\_high(SCLK);

return data;

}

unsigned char spi\_readx(unsigned char addr)

{

unsigned char i;

output\_low(SCLK);

output\_low(nSEL); // must be low during the addr write

for(i=0;i<8;i++)

{

if(addr&0x80)

output\_high(SDO);

else

output\_low(SDO);

addr<<=1;

output\_high(SCLK);

delay\_cycles(1);

output\_low(SCLK);

}

output\_low(SDO);

unsigned data;

for(i=0;i<8;i++)

{

//! if(data&0x80)

//! output\_high(SDO);

//! else

//! output\_low(SDO);

data <<= 1;

output\_high(SCLK);

if(input(SDI)) // reading the actual data from rf module

{ data|=0x01;}

//! else

//! { data&=0xfe;}

//data<<=1; //else kullanmazsak bunu alta da yazabiliriz.

output\_low(SCLK);

}

output\_high(nSEL);

output\_high(SCLK);

return data;

}

unsigned char spi\_R(unsigned char data)

{

unsigned char i;

for(i=0;i<8;i++)

{

if(data&0x80) //if bit of data is 1

output\_high(SDO);

else //if bit of data is 0

output\_low(SDO);

data<<=1;

output\_high(SCLK);

if(input(SDI))

data|=0x01;

else

data&=0xfe;

output\_low(SCLK);

}

return data;

}

# H FILE

#include <18LF46K22.h>

#device ADC=16

#FUSES NOMCLR

#FUSES NOWDT //No Watch Dog Timer

#FUSES WDT128 //Watch Dog Timer uses 1:128 Postscale

#FUSES NOBROWNOUT //No brownout reset

#FUSES NOLVP //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O

#FUSES NOXINST //Extended set extension and Indexed Addressing mode disabled (Legacy mode)

unsigned char spi\_R(unsigned char data);

void SI4432\_RX\_init\_parameter(void);

void rx\_data();

unsigned char spi\_writex(unsigned char addr, unsigned char data);

unsigned char spi\_readx(unsigned char addr );

#define SI4432\_PWRSTATE\_READY 01 // Si4432 ready mode define

#define SI4432\_PWRSTATE\_TX 0x09 // Si4432 Tx mode define

#define SI4432\_PWRSTATE\_RX 05 // Si4432 Rx mode define

#define SI4432\_PACKET\_SENT\_INTERRUPT 04 // Si4432 packet sent interrupt define

#define SI4432\_Rx\_packet\_received\_interrupt 0x02 // Si4432 packet received interrupt define

#define TX1\_RX0 spi\_writex(0x0e|0x80, 0x01) // Antenna switch to tx mode

#define TX0\_RX1 spi\_writex(0x0e|0x80, 0x02) // Antenna switch to Rx mode

#define TX0\_RX0 spi\_writex(0x0e|0x80, 0x00) // Antenna is not in Tx/Rx mode

#define LEDB5 PIN\_B5 //(output)

#define SDO PIN\_D3 // (OUTPUT) PIC E GÖRE

#define SDI PIN\_C4 // (INPUT)

#define SCLK PIN\_D2 // (OUTPUT)

#define nSEL PIN\_D1 // (OUTPUT)

#define nIRQ PIN\_D0 // (INPUT)

#use delay(clock=64MHz,crystal=16MHz)

#use rs232(baud=9600,parity=N,xmit=PIN\_A1,rcv=PIN\_A2,bits=8,stream=PORT1)