msg id Selection bits (HS oscillator)

#pragma config OSCS = OFF // Oscillator System Clock Switch Enable bit (Oscillator system clock switch option is disabled (main oscillator is source))

// CONFIG2L

#pragma config PWRT = OFF // Power-up Timer Enable bit (PWRT disabled)

#pragma config BOR = OFF // Brown-out Reset Enable bit (Brown-out Reset disabled)

#pragma config BORV = 25 // Brown-out Reset Voltage bits (VBOR set to 2.5V)

// CONFIG2H

#pragma config WDT = OFF // Watchdog Timer Enable bit (WDT disabled (control is placed on the SWDTEN bit))

#pragma config WDTPS = 128 // Watchdog Timer Postscale Select bits (1:128)

// CONFIG4L

#pragma config STVR = OFF // Stack Full/Underflow Reset Enable bit (Stack Full/Underflow will not cause Reset)

#pragma config LVP = OFF // Low-Voltage ICSP Enable bit (Low-Voltage ICSP disabled)

// CONFIG5L

#pragma config CP0 = OFF // Code Protection bit (Block 0 (000200-001FFFh) not code protected)

#pragma config CP1 = OFF // Code Protection bit (Block 1 (002000-003FFFh) not code protected)

#pragma config CP2 = OFF // Code Protection bit (Block 2 (004000-005FFFh) not code protected)

#pragma config CP3 = OFF // Code Protection bit (Block 3 (006000-007FFFh) not code protected)

// CONFIG5H

#pragma config CPB = OFF // Boot Block Code Protection bit (Boot Block (000000-0001FFh) not code protected)

#pragma config CPD = OFF // Data EEPROM Code Protection bit (Data EEPROM not code protected)

// CONFIG6L

#pragma config WRT0 = OFF // Write Protection bit (Block 0 (000200-001FFFh) not write protected)

#pragma config WRT1 = OFF // Write Protection bit (Block 1 (002000-003FFFh) not write protected)

#pragma config WRT2 = OFF // Write Protection bit (Block 2 (004000-005FFFh) not write protected)

#pragma config WRT3 = OFF // Write Protection bit (Block 3 (006000-007FFFh) not write protected)

// CONFIG6H

#pragma config WRTC = OFF // Configuration Register Write Protection bit (Configuration registers (300000-3000FFh) not write protected)

#pragma config WRTB = OFF // Boot Block Write Protection bit (Boot Block (000000-0001FFh) not write protected)

#pragma config WRTD = OFF // Data EEPROM Write Protection bit (Data EEPROM not write protected)

// CONFIG7L

#pragma config EBTR0 = OFF // Table Read Protection bit (Block 0 (000200-001FFFh) not protected from Table Reads executed in other blocks)

#pragma config EBTR1 = OFF // Table Read Protection bit (Block 1 (002000-003FFFh) not protected from Table Reads executed in other blocks)

#pragma config EBTR2 = OFF // Table Read Protection bit (Block 2 (004000-005FFFh) not protected from Table Reads executed in other blocks)

#pragma config EBTR3 = OFF // Table Read Protection bit (Block 3 (006000-007FFFh) not protected from Table Reads executed in other blocks)

// CONFIG7H

#pragma config EBTRB = OFF // Boot Block Table Read Protection bit (Boot Block (000000-0001FFh) not protected from Table Reads executed in other blocks)

// #pragma config statements should precede project file includes.

// Use project enums instead of #define for ON and OFF.

#include <xc.h>

#define \_XTAL\_FREQ 20000000

#define ConfigurationMode\_0x80 0x80

#define ListenOnlyMode\_0x60 0x60

#define LoopbackMode\_0x40 0x40

#define DisableMode\_0x20 0x20

#define NormalMode\_0x00 0x00

#define CANCON\_TransmitBuffer\_0 0x08

#define CANCON\_TransmitBuffer\_1 0x06

#define CANCON\_TransmitBuffer\_2 0x04

#define TransmitAtHighPriority 0x0B

#define SWITCH PORTBbits.RB0

enum Tx\_Buffer{Transfer\_Buffer\_0, Transfer\_Buffer\_1, Transfer\_Buffer\_2};

enum Rx\_Buffer{Receiver\_Buffer\_0, Receiver\_Buffer\_1};

char Rx\_data\_11bytes\_Buffer\_0[11]={0};

char Rx\_data\_11bytes\_Buffer\_1[11]={0};

char Rx\_data\_11bytes\_Buffer\_2[11]={0};

char Tx\_data\_8bytes\_Buffer\_0[8]= {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};

char Tx\_data\_8bytes\_Buffer\_1[8]= {0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08};

char Tx\_data\_8bytes\_Buffer\_2[8]= {0x10,0x20,0x30,0x40,0x50,0x60,0x70,0x80};

int flag=0;

int can\_flag=0;

unsigned int Rx\_data\_11bytes\_msgid;

int Sys\_Init()

{

GIE=1;//Global Interrupt Enable

PEIE=1;//Peripheral Interrupt Enable

}

int CANSTAT\_Get\_Mode(void)

{

if((CANSTAT == NormalMode\_0x00))

{

return 1;

}

else if((CANSTAT == DisableMode\_0x20))

{

return 1;

}

else if ((CANSTAT == LoopbackMode\_0x40))

{

return 1;

}

else if ((CANSTAT == ListenOnlyMode\_0x60))

{

return 1;

}

else if ((CANSTAT == ConfigurationMode\_0x80))

{

return 1;

}

else

{

return 0;

}

}

void CANCON\_Set\_Mode(int mode)

{

switch(mode)

{

case 0:

CANCON = ConfigurationMode\_0x80;// CANCON Register set to configure mode

break;

case 1:

CANCON = ListenOnlyMode\_0x60; // CANCON Register set to listen mode

break;

case 2:

CANCON = LoopbackMode\_0x40;// CANCON Register set to loop back mode

case 3:

CANCON = DisableMode\_0x20;// CANCON Register set to disable mode

case 4:

CANCON = NormalMode\_0x00;// CANCON Register set to normal mode

}

}

void Can\_Init(){

// TRISA=0X00; //Trisa bits are set to high for LED as an Output

TRISBbits.RB2=0; //Rb2 bit is set for output

TRISBbits.RB3=1; //Rb3 bit is set for input

//PIE3=0x02; //PIE3 register is set to enable Receiver Buffer 1

//IPR3=0X02; //IPR3 register is set to enable

PIE3bits.RXB0IE = 1; /\*Enable receiver buffer 0 interrupt\*/

IPR3bits.RXB0IP = 1; /\*Enable receiver buffer 0 as high priority\*/

}

void Set\_BaudRate() // baud rate function

{

BRGCON1=0xC1; //BRGCON1 register set to be 0xc1 for 250kbps.

BRGCON2=0xAE; //BRGCON2 register set to be 0xAE for 250kbps.

BRGCON3=0x45; //BRGCON3 register set to be 0x45 for 250kbps.

}

void Set\_Filter() //acceptance filter function

{

RXF0SIDH=0x00; // RXF0SIDH register set to 0x00.

RXF0SIDL=0x00; // RXF0SIDL register set to 0x00.

}

void Set\_Mask() //acceptance mack function

{

RXM0SIDH=0x00; //RXM0SIDH register set to 0x00;

RXM0SIDL=0x00; //RXM0SIDL register set to 0x00;

}

int Tx\_Buffer(char Transfer\_buffer, char byte\_0,char byte\_2)

{

switch(Transfer\_buffer)

{

case 0:

CANCON = CANCON\_TransmitBuffer\_0;

TXB0SIDH=0x10;

TXB0SIDL=0x00;

TXB0DLC = 0x08;

TXB0D0 = byte\_0;

TXB0D1 = Tx\_data\_8bytes\_Buffer\_0[1];

TXB0D2 = byte\_2;

TXB0D3 = Tx\_data\_8bytes\_Buffer\_0[3];

TXB0D4 = Tx\_data\_8bytes\_Buffer\_0[4];

TXB0D5 = byte\_0;

TXB0D6 = byte\_2;

TXB0D7 = Tx\_data\_8bytes\_Buffer\_0[7];

//enable the TXBnCON such that the transmitter buffer and assign the highest priority to the chosen buffer.

TXB0CON = TransmitAtHighPriority;

break;

case 1:

CANCON = CANCON\_TransmitBuffer\_1;

TXB1SIDL = 0x00 ;

TXB1SIDH = 0x00;

TXB1DLC = 0x08;

TXB1D0 = Tx\_data\_8bytes\_Buffer\_1[0];

TXB1D1 = Tx\_data\_8bytes\_Buffer\_1[1];

TXB1D2 = Tx\_data\_8bytes\_Buffer\_1[2];

TXB1D3 = Tx\_data\_8bytes\_Buffer\_1[3];

TXB1D4 = Tx\_data\_8bytes\_Buffer\_1[4];

TXB1D5 = Tx\_data\_8bytes\_Buffer\_1[5];

TXB1D6 = Tx\_data\_8bytes\_Buffer\_1[6];

TXB1D7 = Tx\_data\_8bytes\_Buffer\_1[7];

//enable the TXBnCON such that the transmitter buffer and assign the highest priority to the chosen buffer.

TXB1CON = TransmitAtHighPriority;

break;

// else if(Transfer\_buffer == 2)//010 = Transmit Buffer 2

case 2:

CANCON = CANCON\_TransmitBuffer\_2;

TXB2SIDL = 0x00;//for LSB 3 bits

TXB2SIDH = 0x01;//for MSB 8 bits

TXB2DLC = 0x08;

TXB2D0 = Tx\_data\_8bytes\_Buffer\_2[0];

TXB2D1 = Tx\_data\_8bytes\_Buffer\_2[1];

TXB2D2 = Tx\_data\_8bytes\_Buffer\_2[2];

TXB2D3 = Tx\_data\_8bytes\_Buffer\_2[3];

TXB2D4 = Tx\_data\_8bytes\_Buffer\_2[4];

TXB2D5 = Tx\_data\_8bytes\_Buffer\_2[5];

TXB2D6 = Tx\_data\_8bytes\_Buffer\_2[6];

TXB2D7 = Tx\_data\_8bytes\_Buffer\_2[7];

//enable the TXBnCON such that the transmitter buffer and assign the highest priority to the chosen buffer.

TXB2CON = TransmitAtHighPriority;

default://No code line statement

;

}//switch ends

CANCON=NormalMode\_0x00;

}

void CAN\_Read\_fun()

{

CANCON=0x0C;//Select Normal Mode and Select Receive Buffer\_0.

Rx\_data\_11bytes\_Buffer\_0[0] = RXB0D0;//data\_byte\_0

Rx\_data\_11bytes\_Buffer\_0[1] = RXB0D1;//data\_byte\_1

Rx\_data\_11bytes\_Buffer\_0[2] = RXB0D2;//data\_byte\_2

Rx\_data\_11bytes\_Buffer\_0[3] = RXB0D3;//data\_byte\_3

Rx\_data\_11bytes\_Buffer\_0[4] = RXB0D4;//data\_byte\_4

Rx\_data\_11bytes\_Buffer\_0[5] = RXB0D5;//data\_byte\_5

Rx\_data\_11bytes\_Buffer\_0[6] = RXB0D6;//data\_byte\_6

Rx\_data\_11bytes\_Buffer\_0[7] = RXB0D7;//data\_byte\_7

Rx\_data\_11bytes\_Buffer\_0[8] = RXB0SIDL;//LowByte\_3bits

Rx\_data\_11bytes\_Buffer\_0[9] = RXB0SIDH ;//HigherByte\_8bits

Rx\_data\_11bytes\_Buffer\_0[10] = RXB0DLC;//DLC

Rx\_data\_11bytes\_msgid=Rx\_data\_11bytes\_Buffer\_0[9]<<3|Rx\_data\_11bytes\_Buffer\_0[8]>>5;

RXB0CONbits.RXB0FUL = 0;

CANCON=NormalMode\_0x00;//Resetting the Receiver Buffer\_0 000 and set mode to Normal Mode to get next data

}

\_\_interrupt () ISR1()

{

CANCON=NormalMode\_0x00;

if(RXB0IF)

{

RXB0IF=0;//Clear The Interrupt enable flag

CAN\_Read\_fun();//CAN Rx is called

}

can\_flag=1;

}

void main(void) {

RBPU=0; //PORTB pull-ups are enabled by individual port latch values

TRISCbits.RC1=0;//LED OUTPUT

TRISBbits.RB0=1;//SWITCH1 DOOR INPUT FOR UNLOCK

TRISBbits.RB1=1;//SWITCH1 DOOR INPUT FOR LOCK

TRISCbits.RC2=0;//BUZZER OUTPUT

Sys\_Init();

Can\_Init();

CANCON\_Set\_Mode(0);//Set Configuration Mode the CANCON Register

Set\_Mask();

Set\_Filter();

Set\_BaudRate();

CANCON\_Set\_Mode(4);//Set Normal Mode

PORTCbits.RC1=0; //LED OUTPUT

PORTCbits.RC2=0; //BUZZER OUTPUT

while(1)

{

if(PORTBbits.RB0==0) //PUSH BUTTON (DOOR INPUT)

{

Tx\_Buffer(0,0x0C,0x80);

PORTCbits.RC2=1;

PORTCbits.RC1=1;

\_\_delay\_ms(10000);

PORTCbits.RC2=0;

PORTCbits.RC1=0;

}

if(PORTBbits.RB1==0) //PUSH BUTTON (DOOR INPUT)

{

Tx\_Buffer(0,0x0A,0x70);//LOCK PURPOSE

for(int i=0;i<3;i++) //LED WILL BLINK 6 TIMES

{

PORTCbits.RC1=1;

\_\_delay\_ms(1000);

PORTCbits.RC1=0;

\_\_delay\_ms(1000);

}

for(int i=0;i<2;i++) //LED WILL BLINK 6 TIMES

{

PORTCbits.RC2=1;

\_\_delay\_ms(1000);

PORTCbits.RC2=0;

\_\_delay\_ms(1000);

}

}

// CAN\_Read\_fun();

//\_\_delay\_ms(3000);

if(can\_flag==1)

{

if(Rx\_data\_11bytes\_msgid==0x03) //

{

flag++;

can\_flag=0;

if(Rx\_data\_11bytes\_Buffer\_0[5]==0x0A && Rx\_data\_11bytes\_Buffer\_0[6]==0x70)

{

flag=0;

//PORTCbits.RC1=1;

PORTCbits.RC2=1;

\_\_delay\_ms(3000);

//PORTCbits.RC1=0;

PORTCbits.RC2=0;

\_\_delay\_ms(3000);

}

else if(Rx\_data\_11bytes\_Buffer\_0[0]==0x0C && Rx\_data\_11bytes\_Buffer\_0[2]==0x80)//UNLOCKING

{

flag=0;

PORTCbits.RC1=1;

PORTCbits.RC2=1;

\_\_delay\_ms(3000);

PORTCbits.RC1=0;

PORTCbits.RC2=0;

\_\_delay\_ms(3000);

}

else

{

if(flag==1) //FIRST INVALID ATTEMPTS

{

for(int i=0;i<3;i++) //BUZZER ON/OFF FOR 3 TIMES

{

PORTCbits.RC2=1;

\_\_delay\_ms(1000);

PORTCbits.RC2=0;

\_\_delay\_ms(1000);

}

}

else if(flag==2) //SEDOND INVALID ATTEMPTS

{

for(int i=0;i<6;i++) //BUZZER ON/OFF FOR 6 TIMES

{

PORTCbits.RC2=1;

\_\_delay\_ms(1000);

PORTCbits.RC2=0;

\_\_delay\_ms(1000);

}

}

else if(flag==3) //THIRD INVALID ATTEMPTS

{

PORTCbits.RC2=1; //BUZZER WILL CONTINOUSLY ON

\_\_delay\_ms(10000);

}

}

}

}

}

}

MM

Data

LED1&ALARAM ON

Update 0x80 in byte 0 and byte 1 of transmit data

LED1&ALARAM ON

Update 0x80 in byte 0 and byte 1 of transmit data

LED1 and ALARAM should be OFF

if switch 2

Is ON

(

Setting output for PORTC

for 1th and 2rd pin. PORTB as input switch in RB0

Enable RBPU

Calling following functions

Sys\_Init()

Can\_Init()

CANCON\_Set\_Mode(0);//Set Configuration Set\_Mask()

Set\_Filter()

Set\_BaudRate()