MYHEADER.H:

#ifndef \_\_MYHEADER\_H\_\_

#define \_\_MYHEADER\_H\_\_

#include<lpc214x.h>

#include<string.h>

#include<stdlib.h>

#include "types.h"

#include "delay.h"

#include "defines.h"

#include "i2c.h"

#include "i2c\_define.h"

#include "eeprom.h"

#include "lcd\_defines.h"

#include "lcd.h"

#include "keypad.h"

#include "keypad\_defines.h"

#include "interrupt.h"

#include "uart0.h"

#include "gsm.h"

#include "adc.h"

#include "adc\_defines.h"

#include "edit.h"

#define BUZZER 4 // P0.4

#define MQ2 27 // P1.27

#define B\_SW 6 // P0.6

#endif

TYPES.H:

#ifndef \_\_TYPES\_H\_

#define \_\_TYPES\_H\_

typedef unsigned char u8;

typedef signed char s8;

typedef unsigned short int u16;

typedef signed short int s16;

typedef unsigned int u32;

typedef unsigned long int u64;

typedef signed int s32;

typedef float f32;

typedef double f64;

#endif

DELAY.H:

#ifndef \_DELAY\_H\_

#define \_DELAY\_H\_

void delay\_us(u32);

void delay\_ms(u32);

void delay\_s(u32);

#endif

DEFINES.H:

#define FUNC1 0

#define FUNC2 1

#define FUNC3 2

#define FUNC4 3

#define GPIO\_0\_0 FUNC1

#define TXD0\_0\_0 FUNC2

#define PWM1\_0\_0 FUNC3

#define RSEV\_0\_0 FUNC4

#define GETPINFUNC(WORD,PIN) \

(PIN<16) ? ((WORD>>(PIN\*2))&3) : ((WORD>>((PIN-16)\*2))&3);

#define CFGPIN(WORD,PIN,FUNC) \

(WORD=(PIN<16) ? \

((WORD & ~(3<<(PIN\*2))) | (FUNC<<(PIN\*2))) : \

((WORD & ~(3<<((PIN-16)\*2))) | (FUNC<<((PIN-16)\*2))));

#define SETBIT(WORD,BITPOS) (WORD|=(1<<BITPOS))

#define CLRBIT(WORD,BITPOS) (WORD&=~(1<<BITPOS))

#define CPLBIT(WORD,BITPOS) (WORD^=(1<<BITPOS))

#define WRITEBIT(WORD,BITPOS,BIT) \

(WORD=((WORD&~(1<<BITPOS)) | (BIT<<BITPOS)))

#define READBIT(WORD,BITPOS) ((WORD>>BITPOS)&1)

#define GPIO\_IN 0x00000000

#define GPIO\_OUT 0xFFFFFFFF

#define WRITENIBBLE(WORD,STARTBITPOS,NIBBLE) \

(WORD=(WORD&~(15<<STARTBITPOS)) | (NIBBLE<<STARTBITPOS))

#define READNIBBLE(WORD,STARTBITPOS) ((WORD>>STARTBITPOS)&15)

#define WRITEBYTE(WORD,STARTBITPOS,BYTE) \

(WORD=(WORD&~(255<<STARTBITPOS)) | (BYTE<<STARTBITPOS))

#define READBYTE(WORD,STARTBITPOS) ((WORD>>STARTBITPOS)&255)

#define SSETBIT(WORD,BITPOS) (WORD=1<<BITPOS)

#define SCLRBIT SSETBIT

#define WRITEHWORD(WORD,STARTBITPOS,HWORD) \

(WORD=(WORD&~(0xFFFF<<STARTBITPOS)) | (HWORD<<STARTBITPOS))

#define READWRITEBIT(WORD,WBIT,RBIT) \

(WORD=(WORD&~(1<<WBIT))|(((WORD>>RBIT)&1)<<WBIT));

#define READWRITEBIT2(DWORD,WBIT,SWORD,RBIT) \

(DWORD=(((DWORD&~(1<<WBIT))|((SWORD>>RBIT)&1)<<WBIT)))

i2c.h:

#ifndef \_\_I2C\_H\_\_

#define \_\_I2C\_H\_\_

#include "myheader.h"

void init\_i2c(void);

void i2c\_start(void);

void i2c\_stop(void);

void i2c\_restart(void);

void i2c\_write(u8);

u8 i2c\_read(void);

u8 i2c\_ack(void);

u8 i2c\_nack(void);

u8 i2c\_masterack(void);

#endif

I2C\_DEFINES.H:

#ifndef \_\_I2C\_DEFINES\_H\_\_

#define \_\_I2C\_DEFINES\_H\_\_

//defines for pin function selection

#define SCL\_EN 0x00000010

#define SDA\_EN 0x00000040

//defines for I2C\_SPEED Configuration

#define CCLK 60000000 //Hz

#define PCLK CCLK/4 //Hz

#define I2C\_SPEED 100000 //Hz

#define LOADVAL ((PCLK/I2C\_SPEED)/2)

//bit defines for I2CONSET sfr

#define AA\_BIT 2

#define SI\_BIT 3

#define STO\_BIT 4

#define STA\_BIT 5

#define I2EN\_BIT 6

#define I2C\_SLAVE\_ADDR 0X50

#define SP\_ADDR 0x10

#define PASS\_ADDR 0X00

#define MOBILE\_ADDR 0x40

#endif

EEPROM.h:

#ifndef \_\_I2C\_EEPROM\_H\_\_

#define \_\_I2C\_EEPROM\_H\_\_

void i2c\_eeprom\_byte\_write(u8,u16,u8);

u8 i2c\_eeprom\_randomread(u8,u16);

void i2c\_eeprom\_page\_write(u8,u16,u8 \*p);

void i2c\_eeprom\_seq\_read(u8,u16,u8 \*p);

#endif

LCD\_DEFINES.H:

#ifndef \_\_LCD\_DEFINES\_H\_\_

#define \_\_LCD\_DEFINES\_H\_\_

#define LCD\_DATA 8

#define LCD\_RS 17

#define LCD\_EN 18

#define CLEARLCD 0x01

#define RET\_CUR\_HOME 0x02

#define SHIFT\_DISP\_RIGHT 0x06

#define SHIFT\_DISP\_LEFT 0x07

#define DISP\_OFF 0x08

#define DISP\_ON\_CUR\_OFF 0x0c

#define DISP\_ON\_CUR\_ON 0x0E

#define DISP\_ON\_CUR\_BLK 0x0F

#define MODE\_8BIT\_1LINE 0x30

#define MODE\_4BIT\_1LINE 0x20

#define MODE\_8BIT\_2LINE 0x38

#define MODE\_4BIT\_2LINE 0x20

#define GOTO\_LINE1\_POS0 0x80

#define GOTO\_LINE2\_POS0 0xC0

#define GOTO\_LINE3\_POS0 0x94

#define GOTO\_LINE4\_POS0 0xD4

#define GOTO\_CGRAM\_START 0x40

#define SHIFT\_CUR\_LEFT 0x10

#define SHIFT\_CUR\_RIGH 0x14

#endif

Lcd.h:

#ifndef \_\_LCD\_H\_\_

#define \_\_LCD\_H\_\_

void Init\_LCD(void);

void Char\_LCD(u8);

void Write\_LCD(u8);

void Cmd\_LCD(u8);

void U32\_LCD(u32);

void S32\_LCD(s32);

void F32\_LCD(f32,u32);

void Bin\_LCD(u8,u32);

void Hex\_LCD(u32);

void BuildCGRAM\_LCD(u8 \*,u32);

void SetCursor(u8,u8);

void Str\_LCD(u8 \*);

void Octa\_LCD(u32);

void c\_scroll(s8 \*s1);

#endif

Keypad.h:

#ifndef \_\_KEYPAD\_H\_\_

#define \_\_KEYPAD\_H\_\_

void initkpm(void);

u8 Rowcheck(void);

u8 colcheck(void);

u8 colscan(void);

u8 keyscan(void);

u32 Readnum(void);

void password(u8 \*,u8);

void ReadMobilenum(u8 \*);

#endif

Keypad\_defines.h:

#ifndef \_\_KEYPAD\_DEFINES\_H\_\_

#define \_\_KEYPAD\_DEFINES\_H\_\_

#define ROW0 16 //p1.16

#define ROW1 17

#define ROW2 18

#define ROW3 19

#define COL0 20

#define COL1 21

#define COL2 22

#define COL3 23

#endif

Interrupt.h:

#ifndef \_INTERRUPT\_H

#define \_INTERRUPT\_H

void Enable\_eint0(void);

void eint0\_isr(void) \_\_irq;

#endif

Uart0.h:

#ifndef \_UART0\_H\_

#define \_UART0\_H\_

void InitUART0 (void);

void UART0\_Tx(char ch);

char UART0\_Rx(void);

void UART0\_Str(char \*);

void UART0\_Int(unsigned int);

void UART0\_Float(float);

#endif

Gsm.h:

#ifndef \_GSM\_H\_

#define \_GSM\_H\_

void GSM\_init(void);

void send\_sms(char \*num,char \*msg);

#endif

#ifndef \_ADC\_H\_

#define \_ADC\_H\_

void Init\_ADC(void);

f32 Read\_ADC(u8 chNo);

#endif

Adc\_defines.h:

#ifndef \_ADC\_DEFINES\_H\_

#define \_ADC\_DEFINES\_H\_

//defines for ADCR

#define CH0 0x01

#define CH1 0x02

#define CH2 0x04

#define CH3 0x08

#define CHANNEL\_SEL CH1

#define FOSC 12000000

#define ADCLK 3750000

#define CLKDIV (((PCLK/ADCLK)-1)<<8)

#define PDN\_BIT (1<<21)

#define ADC\_START\_BIT 24

//defines for ADDR

#define DONE\_BIT 31

#endif

Edit.h:

#ifndef \_EDIT\_H

#define \_EDIT\_H

void Menu(void);

void edit\_setpoint(void);

void edit\_pass(void);

void edit(void);

void buzzer(void);

#endif

//FUNCTION DEFNITION & MAIN

/\*CONNECTIONS INFO

LCD ----> P0.8(D0) - P0.15(D7)

P0.17(RS), P0.18(EN)

LM35----> P0.28(DATA PIN)

5V(VCC) , GND

MQ2 ----> P1.27(D0),5V,GND

GSM(M660A) ---> P0.1(TX),P0.0(RX),GND

KEYPAD ----> P1.16 - P1.23 (ROWS 16) & (COLS 20)

EEPROM(AT24C256) ----> P0.2(SCL),P0.3(SDA),3.3V,GND

BUZZER ---> P0.4

EXTERNAL INTERRUPT ---> P0.16

BUZZER OFF SWITCH -----> P0.6

\*/

/\* MAIN.C \*/

#include "myheader.h"

u8 temp\_set,cur\_temp,r\_pass[5],m\_no[11]="7887762200",rm\_no[11],s\_flag=0;

f32 temp;

extern u8 i\_flag;

u32 k=0;

int main()

{

Enable\_eint0(); // INTERRUPT INITIALIZATION

Init\_LCD( ); // LCD INITIALIZATION

Str\_LCD("RAJIV GANDHI COLLEGE"); // DISPLAY COLLEGE NAME

SetCursor(2,0);

Str\_LCD(" OF ENGINEERING ");

SetCursor(3,0);

Str\_LCD("\*\*\*RESEARCH & TECH\*\*");

delay\_ms(5000);

Cmd\_LCD(CLEARLCD);

Str\_LCD("\*\*INDUSTRIAL FAULT\*\*"); // DISPLAY PROJECT NAME

SetCursor(2,0);

Str\_LCD(" INDICATION SYSTEM ");

SetCursor(3,0);

Str\_LCD("\*\*\*WITH SMS ALERT\*\*\*");

delay\_ms(3000);

Cmd\_LCD(CLEARLCD);

InitUART0(); // UART INITIALIZATION

GSM\_init(); // GSM INITIALIZATION

init\_i2c(); // I2C INITIALIZATION

Init\_ADC(); // ADC INITIALIZATION

initkpm(); // KPM INITIALIZATION

SETBIT(IODIR0,BUZZER); // CONFIGURE THE BUZZER PIN AS OUTPUT

i2c\_eeprom\_byte\_write(I2C\_SLAVE\_ADDR,SP\_ADDR,45); // WRITE THE SETPOINT TO THE EEPROM (AT24C256)

i2c\_eeprom\_page\_write(I2C\_SLAVE\_ADDR,PASS\_ADDR,"1234"); // WRITE THE PASSWORD TO THE EEPROM (AT24C256)

temp\_set=i2c\_eeprom\_randomread(I2C\_SLAVE\_ADDR,SP\_ADDR); // READ THE SET POINT FROM EEPROM (AT24C256)

i2c\_eeprom\_seq\_read(I2C\_SLAVE\_ADDR,PASS\_ADDR,r\_pass); // READ THE PASSWORD FROM EEPROM (AT24C256)

i2c\_eeprom\_page\_write(I2C\_SLAVE\_ADDR,MOBILE\_ADDR,m\_no); // WRITE THE MOBILE NUMBER INTO THE EEPROM (AT24C256)

i2c\_eeprom\_seq\_read(I2C\_SLAVE\_ADDR,MOBILE\_ADDR,rm\_no); // READ THE MOBILE NUMBER FROM EEPROM (AT24C256)

Cmd\_LCD(CLEARLCD);

while(1)

{

do

{

SetCursor(1,0);// SET THE CURSOR POSITION 1ST LINE 0TH POSITION ON LCD

Str\_LCD("C.T: "); // C.T MEANS CURRENT TEMPERATURE

temp=Read\_ADC(CH1); // RERAD THE ADC VALUES FROM ON-CHIP ADC IN LPC2148 CHANNEL 1

cur\_temp=(int)(temp\*100); // COVERTING THE FLOAT INTO INTEGER

U32\_LCD(cur\_temp); // DISPLAY CURRENT TEMPERATURE FROM LM35

Char\_LCD(223);

Char\_LCD('C');

SetCursor(2,0); // SET CURSOR POSITION TO 2ND LINE 0TH POSITION ON LCD

Str\_LCD("S.P: "); // S.P MEANS SET POINT

U32\_LCD(temp\_set); // DISPLAY THE SET POINT WHICH IS READ FROM EEPROM

Char\_LCD(223);

Char\_LCD('C');

if(s\_flag==0)

{

SetCursor(3,0);

Str\_LCD("SMOKE: NOT DETECTED");

}

else

{

SetCursor(3,0);

Str\_LCD("SMOKE: DETECTED ");

}

if(k==0)

{

if(cur\_temp>temp\_set) // COMPARE THE CURRENT TEMPERATURE AND SET POINT

{

//IF CURRENT TEMPERATURE IS GRATER THAN SET POINT THEN SEND THE ALERT MESSAGE BY USING SEND\_SMS FUNCTION

send\_sms((char \*)rm\_no, "ALERT! TEMPERATURE CROSSED LIMIT");

buzzer();// BUZZER ON FOR ALERT

k++;

}

// READING THE STATUS OF THE MQ2 SENSOR

// IF IT IS '0' SMOKE DETEDTED

// IF IT IS '1' SMOKE NOT DETECTED

if(((IOPIN1>>MQ2)&1)==0)

{

SetCursor(3,0);

s\_flag=1;

Str\_LCD("SMOKE: DETECTED ");

send\_sms((char \*)rm\_no, "ALERT! SMOKEDETECTED");

buzzer();

k++;

}

}

if(((IOPIN1>>MQ2)&1)==1)

{

s\_flag=0;

}

/\* IF THE TEMOERATURE OR SMOKE DETECTED THEN SEND THE MSG TO OWNER

AND IT SENDS THE MSG CONTINUOUSLY THAT Y WE ARE WRITING THIS CONDITION

BECAUSE OF THID CONDITION THE MSG WILL SEND EVERY 50 ITTERATIONS \*/

if(k>=1)

{

if(k==30)

k=0;

k++;

}

delay\_ms(1000);

}while(i\_flag==0); // CHECKING THE CONDITION THE INTERRUPT IS RAISED OR NOT (i\_flag==1) THEN THE WHILE TERMINATED

i\_flag=0; // AGAIN RESET THE i\_flag VALUE INTO '0'

edit(); // GOTO EDIT FUNCTION TO EDIT PASSWORD AND SET POINT

delay\_ms(1000);

}

}

/\* UART0.C \*/

#include "myheader.h"

#define UART\_INT\_ENABLE 1

char buff[30],dummy;

unsigned char i=0,c,r\_flag;

void UART0\_isr(void) \_\_irq

{

if((U0IIR & 0x04)) //check if receive interrupt

{

c = U0RBR; /\* Read to Clear Receive Interrupt \*/

if(i<30)

buff[i++] = c;

}

else

{

dummy=U0IIR; //Read to Clear transmit interrupt

}

VICVectAddr = 0; /\* dummy write \*/

}

void InitUART0 (void) /\* Initialize Serial Interface \*/

{

PINSEL0 |= 0x00000005; /\* Enable RxD0 and TxD0 \*/

U0LCR = 0x83; /\* 8 bits, no Parity, 1 Stop bit \*/

U0DLL = 97; /\* 9600 Baud Rate @ CCLK/4 VPB Clock \*/

U0LCR = 0x03; /\* DLAB = 0 \*/

#if UART\_INT\_ENABLE > 0

VICIntSelect = 0x00000000; // IRQ

VICVectAddr0 = (unsigned)UART0\_isr;

VICVectCntl0 = 0x20 | 6; /\* UART0 Interrupt \*/

VICIntEnable = 1 << 6; /\* Enable UART0 Interrupt \*/

// U0IIR = 0xc0;

// U0FCR = 0xc7;

U0IER = 0x03; /\* Enable UART0 RX and THRE Interrupts \*/

#endif

}

void UART0\_Tx(char ch) /\* Write character to Serial Port \*/

{

while (!(U0LSR & 0x20));

U0THR = ch;

}

char UART0\_Rx(void) /\* Read character from Serial Port \*/

{

while (!(U0LSR & 0x01));

return (U0RBR);

}

void UART0\_Str(char \*s)

{

while(\*s)

UART0\_Tx(\*s++);

}

void UART0\_Int(unsigned int n)

{

unsigned char a[10]={0,0,0,0,0,0,0,0,0,0};

int i=0;

if(n==0)

{

UART0\_Tx('0');

return;

}

else

{

while(n>0)

{

a[i++]=(n%10)+48;

n=n/10;

}

--i;

for(;i>=0;i--)

{

UART0\_Tx(a[i]);

}

}

}

void UART0\_Float(float f)

{

int x;

float temp;

x=f;

UART0\_Int(x);

UART0\_Tx('.');

temp=(f-x)\*100;

x=temp;

UART0\_Int(x);

}

/\* ADC.C \*/

#include "myheader.h"

#define ADC\_FUNC 0x01

void Init\_ADC(void)

{

PINSEL1 |= (ADC\_FUNC<<24); //configure P0.28 as ADC input

AD0CR=PDN\_BIT|CLKDIV|CHANNEL\_SEL;

}

f32 Read\_ADC(u8 chNo)

{

u16 adcVal=0;

f32 eAR;

WRITEBYTE(AD0CR,0,chNo);

SETBIT(AD0CR,ADC\_START\_BIT);

delay\_us(3);

while(!READBIT(AD0GDR,DONE\_BIT));

CLRBIT(AD0CR,ADC\_START\_BIT);

adcVal=(AD0GDR>>6)&0x3FF;

eAR=((adcVal\*3.3)/1023);

return eAR;

}

/\* EEPROM.C \*/

#include "myheader.h"

void i2c\_eeprom\_byte\_write(u8 slaveAddr,u16 wBuffStartAddr,u8 dat)

{

i2c\_start();

i2c\_write(slaveAddr<<1);

i2c\_write(wBuffStartAddr>>8);

i2c\_write(wBuffStartAddr);

i2c\_write(dat);

i2c\_stop();

delay\_ms(10);

}

u8 i2c\_eeprom\_randomread(u8 slaveAddr,u16 rBuffStartAddr)

{

u8 dat;

i2c\_start();

i2c\_write(slaveAddr<<1);

i2c\_write(rBuffStartAddr>>8);

i2c\_write(rBuffStartAddr);

i2c\_restart();

i2c\_write(slaveAddr<<1|1);

dat=i2c\_nack();

i2c\_stop();

delay\_ms(10);

return dat;

}

void i2c\_eeprom\_page\_write(u8 slaveAddr,u16 wBuffStartAddr,u8 \*p)

{

u8 i;

i2c\_start();

i2c\_write(slaveAddr<<1); //slaveAddr + w

i2c\_write(wBuffStartAddr>>8); //wBuffStartAddr

i2c\_write(wBuffStartAddr);

for(i=0;p[i]!='\0';i++)

{

i2c\_write(p[i]); //wBuffAddr

}

i2c\_write(p[i]);

i2c\_stop();

delay\_ms(10);

}

void i2c\_eeprom\_seq\_read(u8 slaveAddr,u16 rBuffStartAddr,u8 \*p)

{

u8 i=0;

i2c\_start();

i2c\_write(slaveAddr<<1); //slaveAddr + w

i2c\_write(rBuffStartAddr>>8); //rBuffAddr

i2c\_write(rBuffStartAddr);

i2c\_restart();

i2c\_write(slaveAddr<<1|1); //slaveAddr + r

do

{

p[i++]=i2c\_masterack();

}while(p[i-1]!='\0');

p[i]=i2c\_nack();

i2c\_stop();

}

/\* I2C.C \*/

#include "myheader.h"

void init\_i2c(void)

{

//Configure I/O pin for SCL & SDA functions using PINSEL0

PINSEL0|=SCL\_EN|SDA\_EN;

//Configure Speed for I2C Serial Communication

//Using I2CSCLL

I2C0SCLL=LOADVAL;

//& I2CSCLH

I2C0SCLH=LOADVAL;

//I2C Peripheral Enable for Communication

I2C0CONSET=1<<I2EN\_BIT;

}

void i2c\_start(void)

{

// start condition

I2C0CONSET=1<<STA\_BIT;

//wait for start bit status

while(((I2C0CONSET>>SI\_BIT)&1)==0);

// clear start condition

I2C0CONCLR=1<<STA\_BIT;

}

void i2c\_restart(void)

{

// start condition

I2C0CONSET=1<<STA\_BIT;

//clr SI\_BIT

I2C0CONCLR=1<<SI\_BIT;

//wait for SI bit status

while(((I2C0CONSET>>SI\_BIT)&1)==0);

// clear start condition

I2C0CONCLR=1<<STA\_BIT;

}

void i2c\_write(u8 dat)

{

//put data into I2DAT

I2C0DAT=dat;

//clr SI\_BIT

I2C0CONCLR = 1<<SI\_BIT;

//wait for SI bit status

while(((I2C0CONSET>>SI\_BIT)&1)==0);

}

void i2c\_stop(void)

{

// issue stop condition

I2C0CONSET=1<<STO\_BIT;

// clr SI bit status

I2C0CONCLR = 1<<SI\_BIT;

//stop will cleared automatically

//while(((I2CONSET>>STO\_BIT)&1));

}

u8 i2c\_nack(void)

{

I2C0CONSET = 0x00; //Assert Not of Ack

I2C0CONCLR = 1<<SI\_BIT;

while(((I2C0CONSET>>SI\_BIT)&1)==0);

return I2C0DAT;

}

u8 i2c\_masterack(void)

{

I2C0CONSET = 0x04; //Assert Ack

I2C0CONCLR = 1<<SI\_BIT;

while(((I2C0CONSET>>SI\_BIT)&1)==0);

I2C0CONCLR = 0x04; //Clear Assert Ack

return I2C0DAT;

}

/\*INTERRUPT.C\*/

#include "myheader.h"

u8 i\_flag=0;

void eint0\_isr(void) \_\_irq

{

i\_flag=1;

EXTINT=1<<0;

VICVectAddr=0;

}

void Enable\_eint0(void)

{

//CFGPIN(PINSEL0,1,FUNC4);

PINSEL1|=0x15400001;

VICIntSelect=0;

VICIntEnable=1<<14;

VICVectCntl1=(1<<5)|14;

VICVectAddr1=(u32)eint0\_isr;

EXTINT=1<<0;

EXTMODE=1<<0;

EXTPOLAR=0;

}

/\* KEYPAD.C \*/

#include "myheader.h"

u8 kpmlut[4][4]=

{

'1','2','3','A',

'4','5','6','B',

'7','8','9','C',

'\*','0','#','D'

};

u8 codes[4]={0xe,0xd,0xb,0x7};

u32 num;

void initkpm(void)

{

WRITENIBBLE(IODIR1,ROW0,15);

}

u8 Rowcheck(void)

{

u8 rno,i;

for(i=0;i<4;i++)

{

WRITENIBBLE(IOPIN1,ROW0,codes[i]);

if(!colscan())

{

rno=i;

break;

}

}

WRITENIBBLE(IOPIN1,ROW0,0);

return rno;

}

u8 colcheck(void)

{

u8 cno,i;

for(i=0;i<4;i++)

{

if(READNIBBLE(IOPIN1,COL0)==codes[i])

{

cno=i;

break;

}

}

return cno;

}

u8 colscan(void)

{

u32 t;

t=READNIBBLE(IOPIN1,COL0);

t=(t<15)?0:1;

return t;

}

u8 keyscan(void)

{

u8 rno,cno;

while(colscan());

rno=Rowcheck();

cno=colcheck();

while(!colscan());

return kpmlut[rno][cno];

}

u32 Readnum(void)

{

u32 i=0;

static u8 arr[20];

u8 keyval;

Cmd\_LCD(0xc0);

num=0;

while(1)

{

keyval=keyscan();

delay\_ms(100);

if((keyval>='0')&&(keyval<='9'))

{

arr[i]=keyval;

Char\_LCD(arr[i]);

i++;

}

else if(keyval=='D')

{

arr[i]='\0';

break;

}

else if(keyval=='B')

{

i--;

Cmd\_LCD(0xc0+i);

Char\_LCD(' ');

Cmd\_LCD(0xc0+i);

}

}

num=atoi((const char \*)arr);

return (num);

}

void ReadMobilenum(u8 \*arr)

{

u32 i=0;

u8 keyval;

Cmd\_LCD(0xc0);

num=0;

while(1)

{

keyval=keyscan();

delay\_ms(100);

if((keyval>='0')&&(keyval<='9'))

{

arr[i]=keyval;

Char\_LCD(arr[i]);

i++;

}

else if(keyval=='D')

{

arr[i]='\0';

break;

}

else if(keyval=='B')

{

i--;

Cmd\_LCD(0xc0+i);

Char\_LCD(' ');

Cmd\_LCD(0xc0+i);

}

}

}

void password(u8 \*p,u8 k)

{

u8 ch,i=0;

Cmd\_LCD(CLEARLCD);

if(k==1)

{

Str\_LCD("\*ENTER THE PASSWORD\*");

SetCursor(2,0);

Str\_LCD("\*\*\*\*\*\*TO EDIT\*\*\*\*\*\*");

}

else if(k==2)

{

Str\_LCD("\*\*\*ENTER THE OLD \*\*\*");

SetCursor(2,0);

Str\_LCD("\*\*\*\*\*\*PASSWORD\*\*\*\*\*\*");

}

else if(k==3)

{

Str\_LCD("\*\*\*ENTER THE NEW \*\*\*");

SetCursor(2,0);

Str\_LCD("\*\*\*\*\*\*PASSWORD\*\*\*\*\*\*");

}

else if(k==4)

{

Str\_LCD("\*\*CONFIRM THE NEW\*\*");

SetCursor(2,0);

Str\_LCD("\*\*\*\*\*\*PASSWORD\*\*\*\*\*\*");

}

SetCursor(3,0);

while(1)

{

ch=keyscan();

delay\_ms(200);

if((ch>='0')&&(ch<='9'))

{

p[i]=ch;

Char\_LCD(p[i]);

delay\_ms(50);

SetCursor(3,i);

Char\_LCD('\*');

i++;

}

else if(ch=='D')

{

p[i]='\0';

break;

}

else if(ch=='B')

{

i--;

SetCursor(3,i);

Char\_LCD(' ');

SetCursor(3,i);

}

}

}

/\* LCD.C \*/

//#include "pro\_header.h"

//extern char r\_flag;

#include "myheader.h"

void Write\_LCD(u8 data)

{

//SCLRBIT(IOCLR0,LCD\_RW);

WRITEBYTE(IOPIN0,LCD\_DATA,data);

SSETBIT(IOSET0,LCD\_EN);

delay\_us(1);

SCLRBIT(IOCLR0,LCD\_EN);

delay\_ms(2);

}

void Cmd\_LCD(u8 cmd)

{

SCLRBIT(IOCLR0,LCD\_RS);

Write\_LCD(cmd);

}

void Char\_LCD(u8 Sdata)

{

SSETBIT(IOSET0,LCD\_RS);

Write\_LCD(Sdata);

}

void Init\_LCD(void)

{

WRITEBYTE(IODIR0,LCD\_DATA,255);

SETBIT(IODIR0,LCD\_RS);

//SETBIT(IODIR0,LCD\_RW);

SETBIT(IODIR0,LCD\_EN);

delay\_ms(15);

Cmd\_LCD(0x30);

delay\_ms(4);

Cmd\_LCD(0x30);

delay\_us(100);

Cmd\_LCD(0x30);

Cmd\_LCD(RET\_CUR\_HOME);

Cmd\_LCD(MODE\_8BIT\_2LINE);

Cmd\_LCD(DISP\_ON\_CUR\_OFF);

Cmd\_LCD(CLEARLCD);

Cmd\_LCD(SHIFT\_DISP\_RIGHT);

}

void SetCursor(u8 lineno,u8 pos)

{

if(lineno==1)

{

Cmd\_LCD(GOTO\_LINE1\_POS0+pos);

}

else if(lineno==2)

{

Cmd\_LCD(GOTO\_LINE2\_POS0+pos);

}

else if(lineno==3)

{

Cmd\_LCD(GOTO\_LINE3\_POS0+pos);

}

else if(lineno==4)

{

Cmd\_LCD(GOTO\_LINE4\_POS0+pos);

}

}

void Str\_LCD(u8 \*p)

{

while(\*p)

Char\_LCD(\*p++);

}

void U32\_LCD(u32 n)

{

u8 a[10];

s32 i=0;

if(n==0)

{

Char\_LCD('0');

}

else

{

while(n)

{

a[i++]=n%10+48;

n=n/10;

}

for(--i;i>=0;i--)

{

Char\_LCD(a[i]);

}

}

}

void S32\_LCD(s32 num)

{

if(num<0)

{

Char\_LCD('-');

num=-num;

}

U32\_LCD(num);

}

/\* EDIT.C \*/

#include "myheader.h"

u8 c\_pass[5],n\_pass[5],cn\_pass[5];

extern u8 r\_pass[5],temp\_set,m\_no[11],rm\_no[11];

extern u32 k;

void Menu(void)

{

Cmd\_LCD(CLEARLCD);

Str\_LCD("1. EDIT SET POINT");

SetCursor(2,0);

Str\_LCD("2. EDIT PASSWORD");

SetCursor(3,0);

Str\_LCD("3. MODIFY MOBILE NUM");

SetCursor(4,0);

Str\_LCD("4. EXIT");

}

void edit\_setpoint(void)

{

u8 i,val,j=0;

l: password(c\_pass,1);

Cmd\_LCD(CLEARLCD);

Str\_LCD("VALIDATING");

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

{

Char\_LCD('.');

delay\_ms(100);

}

if(strcmp((const char \*)c\_pass,(const char \*)r\_pass)==0)

{

Cmd\_LCD(CLEARLCD);

Str\_LCD("VALID PASSWORD");

delay\_ms(2000);

Cmd\_LCD(CLEARLCD);

Str\_LCD("ENTER THE SET POINT");

val=Readnum();

Cmd\_LCD(CLEARLCD);

Str\_LCD("SET POINT CHANGED");

delay\_ms(1000);

i2c\_eeprom\_byte\_write(I2C\_SLAVE\_ADDR,SP\_ADDR,val); // WRITE THE NEW SETPOINT TO THE EEPROM (AT24C256)

temp\_set=i2c\_eeprom\_randomread(I2C\_SLAVE\_ADDR,SP\_ADDR); // READ THE NEW SET POINT FROM EEPROM (AT24C256)

}

else

{

Cmd\_LCD(CLEARLCD);

Str\_LCD("INVALID PASSWORD");

delay\_ms(1000);

j++;

if(j==3)

{

send\_sms((char \*)rm\_no, "ALERT! UNAUTHORIZED PERSON");

Cmd\_LCD(CLEARLCD);

Str\_LCD("U R SYSTEM BLOCKED");

delay\_ms(5000);

j=0;

return;

}

goto l;

}

Cmd\_LCD(CLEARLCD);

}

void edit\_pass(void)

{

u8 i;

password(c\_pass,2);

Cmd\_LCD(CLEARLCD);

Str\_LCD("VALIDATING");

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

{

Char\_LCD('.');

delay\_ms(100);

}

if(strcmp((const char \*)c\_pass,(const char \*)r\_pass)==0)

{

Cmd\_LCD(CLEARLCD);

Str\_LCD("VALID PASSWORD");

delay\_ms(1000);

Cmd\_LCD(CLEARLCD);

password(n\_pass,3);

if(strcmp((const char \*)n\_pass,(const char \*)r\_pass)!=0)

{

password(cn\_pass,4);

if(strcmp((const char \*)n\_pass,(const char \*)cn\_pass)==0)

{

Cmd\_LCD(CLEARLCD);

Str\_LCD("PASSWORD CHANGED");

delay\_ms(1000);

i2c\_eeprom\_page\_write(I2C\_SLAVE\_ADDR,PASS\_ADDR,cn\_pass); // WRITE THE PASSWORD TO THE EEPROM (AT24C256)

i2c\_eeprom\_seq\_read(I2C\_SLAVE\_ADDR,PASS\_ADDR,r\_pass); // READ THE PASSWORD FROM EEPROM (AT24C256)

}

}

else

{

Cmd\_LCD(CLEARLCD);

Str\_LCD("OLD PASS & NEW PASS");

SetCursor(2,0);

Str\_LCD("\*\*\*\*\* ARE SAME \*\*\*\*\*");

delay\_ms(1000);

}

}

Cmd\_LCD(CLEARLCD);

}

void edit\_mobileno(void)

{

Cmd\_LCD(CLEARLCD);

Str\_LCD("ENTER THE MOBILE NUM");

ReadMobilenum(m\_no);

i2c\_eeprom\_page\_write(I2C\_SLAVE\_ADDR,MOBILE\_ADDR,m\_no); // WRITE THE MOBILE NUMBER INTO THE EEPROM (AT24C256)

i2c\_eeprom\_seq\_read(I2C\_SLAVE\_ADDR,MOBILE\_ADDR,rm\_no); // READ THE MOBILE NUMBER FROM EEPROM (AT24C256)

Cmd\_LCD(CLEARLCD);

Str\_LCD("MOBILE NUM MODIFIED");

delay\_ms(2000);

k=0;

Cmd\_LCD(CLEARLCD);

}

void edit(void)

{

u8 ch;

Menu();

delay\_ms(2000);

Cmd\_LCD(CLEARLCD);

Str\_LCD("ENTER THE CHOICE");

SetCursor(2,0);

ch=keyscan();

delay\_ms(100);

Char\_LCD(ch);

delay\_ms(1000);

switch(ch)

{

case '1':edit\_setpoint();

break;

case '2':edit\_pass();

break;

case '3':edit\_mobileno();

break;

case '4':Cmd\_LCD(CLEARLCD);

return;

}

}

void buzzer(void)

{

while(1)

{

SETBIT(IOPIN0,BUZZER);

delay\_ms(100);

CLRBIT(IOPIN0,BUZZER);

delay\_ms(50);

Cmd\_LCD(CLEARLCD);

Str\_LCD("ALERT! ");

if(((IOPIN0>>B\_SW)&1)==0)

{

CLRBIT(IOPIN0,BUZZER);

Cmd\_LCD(CLEARLCD);

break;

}

}

}

/\* DELAY.C \*/

#include "myheader.h"

void delay\_us(u32 dlyUS)

{

dlyUS\*=12;//@CPU clock 60MHz

while(dlyUS--);

}

void delay\_ms(u32 dlyMS)

{

dlyMS\*=12000;//@CPU clock 60MHz

while(dlyMS--);

}

void delay\_s(u32 dlyS)

{

dlyS\*=12000000;//@CPU clock 60MHz

while(dlyS--);

}

/\* GSM.C \*/

#include "myheader.h"

extern char buff[30],i,ch;

void GSM\_init(void)

{

Cmd\_LCD(0x80);

Str\_LCD("GSM init....");

i=0;memset(buff,'\0',200);

UART0\_Str("AT\r\n");

while(i<2);

delay\_ms(500);

buff[i] = '\0';

if(strstr(buff,"OK"))

{

Cmd\_LCD(0xC0);

Str\_LCD("OK");

delay\_ms(500);

Cmd\_LCD(0xC0);

Str\_LCD(" ");

i=0;memset(buff,'\0',200);

UART0\_Str("ATE0\r\n");

while(i<2);

delay\_ms(500);

buff[i] = '\0';

if(strstr(buff,"OK"))

{

Cmd\_LCD(0xC0);

Str\_LCD("OK");

delay\_ms(500);

Cmd\_LCD(0xC0);

Str\_LCD(" ");

i=0;memset(buff,'\0',200);

UART0\_Str("AT+CMGF=1\r\n");

while(i<2);

delay\_ms(500);

buff[i] = '\0';

if(strstr(buff,"OK"))

{

Cmd\_LCD(0xC0);

Str\_LCD("OK");

delay\_ms(500);

Cmd\_LCD(0xC0);

Str\_LCD(" ");

}

else

{

Cmd\_LCD(0xC0);

Str\_LCD("ERROR");

delay\_ms(500);

Cmd\_LCD(0xC0);

Str\_LCD(" ");

}

}

else

{

Cmd\_LCD(0xC0);

Str\_LCD("ERROR");

delay\_ms(500);

Cmd\_LCD(0xC0);

Str\_LCD(" ");

}

}

else

{

Cmd\_LCD(0xC0);

Str\_LCD("ERROR");

delay\_ms(500);

Cmd\_LCD(0xC0);

Str\_LCD(" ");

}

}

void send\_sms(char \*num,char \*msg)

{

Cmd\_LCD(0x01);

Str\_LCD("Sending sms....");

delay\_ms(500);

i=0;memset(buff,'\0',200);

UART0\_Str("AT+CMGS=");

UART0\_Tx('"');

UART0\_Str((char \*)num);

UART0\_Tx('"');

UART0\_Str("\r\n");

delay\_ms(500);

i=0;

UART0\_Str((char \*)msg);

UART0\_Tx(0x1A);

delay\_ms(5000);

while(i<3);

buff[i] = '\0';

i=0;

if(strstr(buff,"OK"))

{

Cmd\_LCD(0x01);

Str\_LCD("Message Sent..");

delay\_ms(500);

}

else

{

Cmd\_LCD(0x01);

Str\_LCD("Message failed..");

delay\_ms(1000);

}

Cmd\_LCD(CLEARLCD);

}