Code

#define F\_CPU 8000000ul

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

/\*\*MACROS/

#define USART\_BAUDRATE 9600

#define BAUD\_PRESCALE (((F\_CPU / (USART\_BAUDRATE \* 16UL))) - 1)

#define uchar unsigned char

#define uint unsigned int

#define LCDPORTDIR DDRB

#define LCDPORT PORTB

#define rs 0

#define rw 1

#define en 2

#define RSLow (LCDPORT&=~(1<<rs))

#define RSHigh (LCDPORT|=(1<<rs))

#define RWLow (LCDPORT&=~(1<<rw))

#define ENLow (LCDPORT&=~(1<<en))

#define ENHigh (LCDPORT|=(1<<en))

#define KeyPORTdir DDRA

#define key PINA

#define KeyPORT PORTA

#define OK 3

#define UP 0

#define DOWN 1

#define DEL 3

#define MATCH 1

#define ENROL 2

#define enrol (key & (1<<ENROL)) // key 1

#define match (key & (1<<MATCH)) // key 4

#define delet (key & (1<<DEL)) // key 2

#define up (key & (1<<UP)) // key 3

#define down (key & (1<<DOWN)) // key 4

#define ok (key & (1<<OK)) // key 2

#define LEDdir DDRC

#define LEDPort PORTC

#define LED 3

#define BUZ 2

#define LEDHigh (LEDPort += (1<<LED))

#define LEDLow (LEDPort &= ~(1<<LED))

#define BUZHigh (LEDPort += (1<<BUZ))

#define BUZLow (LEDPort &= ~(1<<BUZ))

#define HIGH 1

#define LOW 0

#define PASS 0

#define ERROR 1

#define check(id) id=up<down?++id:down<up?--id:id;

#define maxId 5

#define dataLenth 6

#define eepStartAdd 10

/variable/

uchar buf[20];

uchar buf1[20];

volatile uint ind;

volatile uint flag;

uint msCount=0;

uint g\_timerflag=1;

volatile uint count=0;

uchar data[10];

uint id=1;

int s,a,b,c;

const char passPack[]={0xEF, 0x1, 0xFF, 0xFF, 0xFF, 0xFF, 0x1, 0x0, 0x7, 0x13, 0x0, 0x0, 0x0, 0x0, 0x0, 0x1B};

const char f\_detect[]={0xEF, 0x1, 0xFF, 0xFF, 0xFF, 0xFF, 0x1, 0x0, 0x3, 0x1, 0x0, 0x5};

const char f\_imz2ch1[]={0xEF, 0x1, 0xFF, 0xFF, 0xFF, 0xFF, 0x1, 0x0, 0x4, 0x2, 0x1, 0x0, 0x8};

const char f\_imz2ch2[]={0xEF, 0x1, 0xFF, 0xFF, 0xFF, 0xFF, 0x1, 0x0, 0x4, 0x2, 0x2, 0x0, 0x9};

const char f\_createModel[]={0xEF,0x1,0xFF,0xFF,0xFF,0xFF,0x1,0x0,0x3,0x5,0x0,0x9};

char f\_storeModel[]={0xEF,0x1,0xFF,0xFF,0xFF,0xFF,0x1,0x0,0x6,0x6,0x1,0x0,0x1,0x0,0xE};

const char f\_search[]={0xEF, 0x1, 0xFF, 0xFF, 0xFF, 0xFF, 0x1, 0x0, 0x8, 0x1B, 0x1, 0x0, 0x0, 0x0, 0xA3, 0x0, 0xC8};

char f\_delete[]={0xEF,0x1,0xFF,0xFF,0xFF,0xFF,0x1,0x0,0x7,0xC,0x0,0x0,0x0,0x1,0x0,0x15};

//const char f\_readNotepad[]={0xEF,0x1,0xFF,0xFF,0xFF,0xFF,0x1,0x0,0x4,0x19,0x0,0x0,0x1E};

//char f\_writeNotepad[]={0xEF,0x1,0xFF,0xFF,0xFF,0xFF,0x1,0x0,0x24};

int timeStamp[7],day;

enum

{

CMD=0,

DATA,

};

void buzzer(uint);

void lcdwrite(char ch,char r)

{

LCDPORT=ch & 0xF0;

RWLow;

if(r == 1)

RSHigh;

else

RSLow;

ENHigh;

\_delay\_ms(5);

ENLow;

\_delay\_ms(10);

LCDPORT=ch<<4 & 0xF0;

RWLow;

if(r == 1)

RSHigh;

else

RSLow;

ENHigh;

\_delay\_ms(5);

ENLow;

\_delay\_ms(10);

}

void lcdprint(char \*str)

{

while(\*str)

{

lcdwrite(\*str++,DATA);

//\_\_delay\_ms(20);

}

}

void lcdbegin()

{

uchar lcdcmd[5]={0x02,0x28,0x0E,0x06,0x01};

uint i=0;

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

lcdwrite(lcdcmd[i], CMD);

}

void serialbegin()

{

UCSRC = (1 << URSEL) | (1 << UCSZ0) | (1 << UCSZ1);

UBRRH = (BAUD\_PRESCALE >> 8);

UBRRL = BAUD\_PRESCALE;

UCSRB=(1<<RXEN)|(1<<TXEN)|(1<<RXCIE);

sei();

}

ISR(USART\_RXC\_vect)

{

char ch=UDR;

buf[ind++]=ch;

if(ind>0)

flag=1;

[11:52 AM, 4/24/2022] Sindhu M T: serial1Write(ch);

}

void serialwrite(char ch)

{

while ((UCSRA & (1 << UDRE)) == 0);

UDR = ch;

}

void serialprint(char \*str)

{

while(\*str)

{

serialwrite(\*str++);

}

}

void serialprintln(char \*str)

{

serialprint(str);

serialwrite(0x0d);

serialwrite(0x0a);

}

void serialFlush()

{

for(int i=0;i<sizeof(buf);i++)

{

buf[i]=0;

}

}

void SerialSoftWrite(char ch)

{

PORTD&=~(1<<7);

\_delay\_us(104);

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

{

if(ch & 1)

PORTD|=(1<<7);

else

PORTD&=~(1<<7);

\_delay\_us(104);

ch>>=1;

}

PORTD|=(1<<7);

\_delay\_us(104);

}

void SerialSoftPrint(char \*str)

{

while(\*str)

{

SerialSoftWrite(\*str);

str++;

}

}

void SerialSoftPrintln(char \*str)

{

SerialSoftPrint(str);

SerialSoftWrite(0x0D);

SerialSoftWrite(0x0A);

}

int bcdtochar(char num)

{

return ((num/16 \* 10) + (num % 16));

}

void RTC\_start()

{

TWCR=(1<<TWINT)|(1<<TWSTA)|(1<<TWEN);

while((TWCR&0x80)==0x00);

}

void RTC\_stp()

{

TWCR=(1<<TWINT)|(1<<TWEN)|(1<<TWSTO); //stop communication

}

void RTC\_read()

{

TWCR=(1<<TWINT)|(1<<TWSTA)|(1<<TWEN);

while((TWCR&0x80)==0x00);

TWDR=0xD0; //RTC write (slave address)

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR&(1<<TWINT)));

TWDR=0x00; //RTC write (word address)

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR&(1<<TWINT)));

TWCR=(1<<TWINT)|(1<<TWSTA)|(1<<TWEN); //start RTC communication again

while ((TWCR&0x80)==0x00);

TWDR=0xD1; // RTC command to read

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR&(1<<TWINT)));

}

void sec\_init(unsigned char d)

{

TWDR=d; //second init

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR&(1<<TWINT)));

}

void min\_init(unsigned char d)

{

TWDR=d; //minute init

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR&(1<<TWINT)));

}

void hr\_init(unsigned char d)

{

TWDR=d; //hour init

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR&(1<<TWINT)));

}

void day\_init(unsigned char d)

{

TWDR=d; //days init

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR&(1<<TWINT)));

}

void date\_init(unsigned char d)

{

TWDR=d; //date init

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR&(1<<TWINT)));

}

void month\_init(unsigned char d)

{

TWDR=d; //month init

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR&(1<<TWINT)));

}

void yr\_init(unsigned char d)

{

TWDR=d; //year init

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR&(1<<TWINT)));

}

int sec\_rw()

{

TWCR|=(1<<TWINT)|(1<<TWEA); //RTC second read

while((TWCR & 0x80)==0x00);

return bcdtochar(TWDR);

}

int min\_rw()

{

TWCR|=(1<<TWINT); //RTC minute read

TWCR|=(1<<TWEA);

while((TWCR & 0x80)==0x00);

return bcdtochar(TWDR);

}[11:59 AM, 4/24/2022] Sindhu M T: void yr\_init(unsigned char d)

{

TWDR=d; //year init

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR&(1<<TWINT)));

}

int sec\_rw()

{

TWCR|=(1<<TWINT)|(1<<TWEA); //RTC second read

while((TWCR & 0x80)==0x00);

return bcdtochar(TWDR);

}

int min\_rw()

{

TWCR|=(1<<TWINT); //RTC minute read

TWCR|=(1<<TWEA);

while((TWCR & 0x80)==0x00);

return bcdtochar(TWDR);

}

int hr\_rw()

{

TWCR|=(1<<TWINT)|(1<<TWEA); //RTC hour read

while((TWCR & 0x80)==0x00);

return bcdtochar(TWDR);

}

int day\_rd()

{

TWCR|=(1<<TWINT)|(1<<TWEA); //RTC day read

while((TWCR&0x80)==0x00);

return bcdtochar(TWDR);

}

int date\_rw()

{

TWCR|=(1<<TWINT)|(1<<TWEA); //RTC date read

while((TWCR & 0x80)==0x00);

return bcdtochar(TWDR);

}

int month\_rw()

{

TWCR|=(1<<TWINT)|(1<<TWEA); //RTC month read

while((TWCR & 0x80)==0x00);

return bcdtochar(TWDR);

}

int yr\_rw()

{

TWCR|=(1<<TWINT); //RTC year read

TWCR&=(~(1<<TWEA));

while((TWCR & 0x80)==0x00);

return bcdtochar(TWDR);

}

void device()

{

TWDR=0xD0; //RTC write (slave address)

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR&(1<<TWINT)));

TWDR=0x00; // word address write

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR&(1<<TWINT)));

}

void RTCTimeSet()

{

RTC\_start();

device();

sec\_init(0);

min\_init(0x47);

hr\_init(0x22);

day\_init(0x03);

date\_init(0x23);

month\_init(0x08);

yr\_init(0x19);

RTC\_stp();

}

void show()

{

char tem[20];

sprintf(tem,"%d",timeStamp[0]);

lcdwrite(0x80,CMD);

lcdprint("Time:");

lcdprint(tem);

lcdwrite(':',DATA);

sprintf(tem,"%d",timeStamp[1]);

lcdprint(tem);

lcdwrite(':',DATA);

sprintf(tem,"%d",timeStamp[2]);

lcdprint(tem);

lcdprint(" ");

lcdwrite(0xc0,CMD);

lcdprint("Date:");

sprintf(tem,"%d",timeStamp[3]);

lcdprint(tem);

lcdwrite('/',DATA);

sprintf(tem,"%d",timeStamp[4]);

lcdprint(tem);

lcdwrite('/',DATA);

sprintf(tem,"%d",timeStamp[5]);

lcdprint("20");

if(timeStamp[5]<10)

lcdwrite('0',DATA);

lcdprint(tem);

lcdprint(" ");

}

void RTC()

{

RTC\_read();

timeStamp[2]=sec\_rw();

timeStamp[1]=min\_rw();

timeStamp[0]=hr\_rw();

day=day\_rd();

timeStamp[3]=date\_rw();

timeStamp[4]=month\_rw();

timeStamp[5]=yr\_rw();

RTC\_stp();

show();

}

int eeprom\_write(unsigned int add,unsigned char data)

{

while(EECR&(1<<EEWE));

EEAR=add;

EEDR=data;

EECR|=(1<<EEMWE);

EECR|=(1<<EEWE);

return 0;

}

char eeprom\_read(unsigned int add)

{

while(EECR & (1<<EEWE));

EEAR=add;

EECR|=(1<<EERE);

return EEDR;

}

void saveData(int id)

{

uint cIndex= eeprom\_read(id);

if(cIndex == 0)

cIndex=1;

uint cAddress= (cIndex\*6) + (id-1)\*48;

for(int i=0;i<6;i++)

eeprom\_write(cAddress+i,timeStamp[i]);

eeprom\_write(id,cIndex+1);

}

int sendcmd2fp(char \*pack, int len)

{

int res=ERROR;

serialFlush();

ind=0;

\_delay\_ms(100);

for(int i=0;i<len;i++)

{

serialwrite(\*(pack+i));

}

\_delay\_ms(1000);

if(flag == 1)

{

if(buf[0] == 0xEF && buf[1] == 0x01)

{

if(buf[6] == 0x07) // ack

{

if(buf[9] == 0)

{

uint data\_len= buf[7];

data\_len<<=8;

data\_len|=buf[8];

for(int i=0;i<data\_len;i++)

data[i]=0;

//data=(char \*)calloc(data\_len, sizeof(data));

for(int i=0;i<data\_len-2;i++)

{

data[i]=buf[