중간 프로젝트 결과보고서

2024-50924 우상욱

주제: ATtiny2313A를 활용한 타이머

목차

1. 개요 및 기능 설명

수업에서 지금까지 배웠던 것을 응용하여 HD44780, MAX7219, SH1106을 동시에 실행시킨다. 타이머를 구현하기 위해 while문의 clock을 사용하여 시간을 측정하여 숫자를 계산하여 display한다.

전체적인 기능은 다음과 같습니다.

1) 스위치1을 누르면 타이머가 시작한다.
2) 스위치2를 누르면 타이머가 초기화되고 멈춘다.
3) 타이머가 99시간을 초과할 경우 동작을 정지한다.

2. 코드 설명

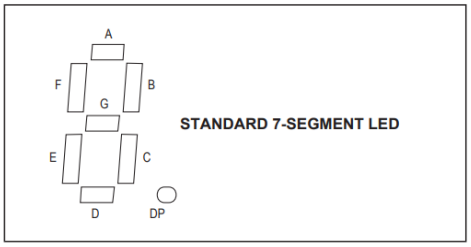
```
void max_print_init(){
    MAX_CS_PORT &= ~(1<<MAX_CS_BIT);
    SpiUSITx(0x09);
    SpiUSITx(0b11011110);

    MAX_CS_PORT |= (1<<MAX_CS_BIT);

    /// CS 를 0 으로 설정.
    MAX_CS_PORT &= ~(1<<MAX_CS_BIT);
    SpiUSITx(6);
    SpiUSITx(0b10110111);//H
    /// CS 를 1 로 설정.
    MAX_CS_PORT |= (1<<MAX_CS_BIT);

    /// CS 를 0 으로 설정.
    MAX_CS_PORT &= ~(1<<MAX_CS_BIT);
    SpiUSITx(1);
    SpiUSITx(0b11011011);
    /// CS 를 1 로 설정.
    MAX_CS_PORT |= (1<<MAX_CS_BIT);
}
```

MAX7219 에 H 와 S 를 display 하기 위해 6 과 1 번지 7-Segment 만 non-decode 모드로 정보를 썼습니다. H 와 S 는 바뀌지 않으므로 init 함수로 한 번만 실행한다.



	REGISTER DATA							
	D7	D6	D5	D4	D3	D2	D1	D0
Corresponding Segment Line	DP	A	B	C	D	E	F	G

```
static void max_print(uint8_t hour, uint8_t minute, uint8_t second){
```

```
        /// CS 를 0 으로 설정.
        MAX_CS_PORT &= ~(1<<MAX_CS_BIT);
        SpiUSITx(8);
        SpiUSITx(hour/10);
        /// CS 를 1 로 설정.
        MAX_CS_PORT |= (1<<MAX_CS_BIT);

        MAX_CS_PORT &= ~(1<<MAX_CS_BIT);
        SpiUSITx(7);
        SpiUSITx(hour%10);
        MAX_CS_PORT |= (1<<MAX_CS_BIT);

        MAX_CS_PORT &= ~(1<<MAX_CS_BIT);
        SpiUSITx(5);
        SpiUSITx(minute/10);
        MAX_CS_PORT |= (1<<MAX_CS_BIT);

        MAX_CS_PORT &= ~(1<<MAX_CS_BIT);
        SpiUSITx(4);
        SpiUSITx(minute%10);
        MAX_CS_PORT |= (1<<MAX_CS_BIT);

        MAX_CS_PORT &= ~(1<<MAX_CS_BIT);
        SpiUSITx(3);
        SpiUSITx(second/10);
        MAX_CS_PORT |= (1<<MAX_CS_BIT);

        MAX_CS_PORT &= ~(1<<MAX_CS_BIT);
        SpiUSITx(2);
        SpiUSITx(second%10);
        MAX_CS_PORT |= (1<<MAX_CS_BIT);
}
```

decode 모드로 숫자를 출력한다.

```
static void sh_print(uint8_t hour, uint8_t minute, uint8_t second){
     //test font
     uint8_t k = 16;
     sh1106_text_font24(char_addr[hour/10],char_width[hour/10],4,10);
     sh1106_text_font24(char_addr[hour%10],char_width[hour%10],4,10+k*1);
     //// CS <- 0
     SIM_CS_PORT &= ~(1<<SIM_CS_BIT);
     for(uint8_t i =4;i<7;i++){
               sh1106_set_location(i,45);//40 , 26
               //// DC <- 1
               SIM_CS_PORT |= (1<<(SIM_CS_BIT-1));
               SpiUSITx(0xFF);
     }
     //// CS <- 1
     SIM_CS_PORT |= (1<<SIM_CS_BIT);
     sh1106_text_font24(char_addr[minute/10],char_width[minute/10],4,10+k*3-10);
     sh1106_text_font24(char_addr[minute%10],char_width[minute%10],4,10+k*4-10);
     //// CS <- 0
```

```
	SIM_CS_PORT &= ~(1<<SIM_CS_BIT);
	for(uint8_t i =4;i<7;i++){
		sh1106_set_location(i,84);
		//// DC <- 1
		SIM_CS_PORT |= (1<<(SIM_CS_BIT-1));
		SpiUSITx(0xFF);
	}
	//// CS <- 1
	SIM_CS_PORT |= (1<<SIM_CS_BIT);
	sh1106_text_font24(char_addr[second/10],char_width[second/10],4,10+k*5);
	sh1106_text_font24(char_addr[second%10],char_width[second%10],4, 10+k*6);
}
```

구분선을 위해 시와 분, 분과 초 사이에 일자선을 추가했습니다. SH1106은 column 0부터 131, row 0부터 63이고 page 0부터 7까지로 구분되므로 위와 같은 방식으로 구분선을 추가한다.

```
char message[] = "00 : 00 : 00";

static void display(uint8_t hour, uint8_t minute, uint8_t second){
	max_print(hour,minute,second);
	sh_print(hour,minute,second);
	sprintf(message,"%02d : %02d : %02d",hour,minute,second);
	lcd_puts(message);//delay == 240ms
}
```

코드의 간편화를 위해 display를 위한 함수를 한 함수에 다시 모아서 실행한다.

```
int main(void){
	uint8_t hour=0,minute=0,second=0;
	uint32_t timer=0;
	PINB &= 0;//read mode
	PORTB |= (1<<PORTB1);
	PORTB |= (1<<PORTB2);//pullup necessary
	_delay_ms(10);
	sh1106_init();
	max7219_init();
	max_print_init();
	lcd_init();
	lcd_write_byte(0, 0x01); // display clear
	sh1106_clear();
	sh1106_border();
	display(hour,minute,second);
	uint8_t reset = 0;//when 1 all reset
	uint8_t start = 0;//0 stop 1 start


	while (1) {

		if(((PINB>>1) & 1)==0){
```

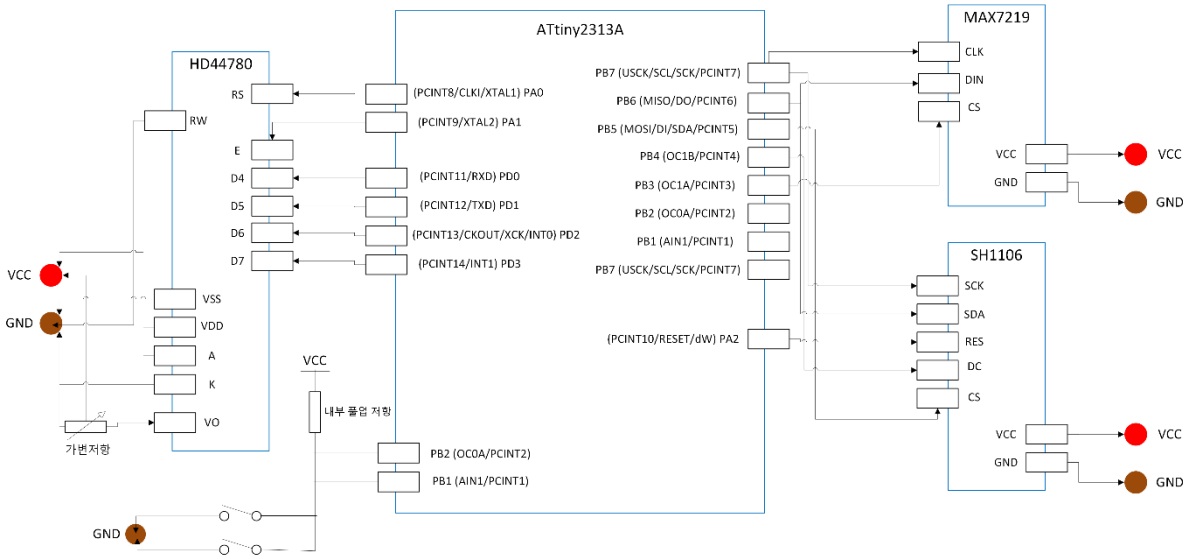
```
                start=1;
        }
        if(((PINB>>2) & 1)==0){
                reset = 1;
                start=0;
        }
        if(start == 1){
                if(timer == 350000){
                        second++;
                        timer=0;
                        display(hour,minute,second);
                        timer = timer + 10;
                        timer = timer + 96000;
                }
                if(second == 60){
                        second = 0;
                        minute++;
                }
                if(minute == 60){
                        minute=0;
                        hour++;
                }
                if(hour>=100){
                        start = 0;
                }
                timer++;
        }

        if(reset==1){
                hour = 0;
                timer = 0;
                minute = 0;
                second = 0;
                display(hour,minute,second);
                timer = timer + 96000;
                reset = 0;
        }

    }
}
```

display시에 lcd_puts 함수에서 240ms의 delay가 발생하므로 timer에 clock회수로 치환하여 값을 더해 보정한다.

3. 회로도



4. 한계점 및 개선 사항

timer 변수를 제어하여 초를 측정할 때 초기에는 clock 주파수인 8Mhz의 값을 기준으로 하고 lcd제어시 발생하는 delay를 보정해주는 작업도 하였으나 부정확하였다. 이번 프로젝트에서는 값을 실험적으로 수정하여 최대한 비슷하게 수행하려고 하였다. 값을 정확히 하기 위해서는 interrupt timer와 같은 방식을 활용하여 보다 정확히 시간을 측정해야 한다.

스위치를 사용하여 timer를 제어할 때 스위치 바운싱이 발생할 수 있다. 따라서 스위치에 하드웨어적으로 디바운싱을 회로를 사용하는 등 개선할 수 있다.

현재 코드에서는 스위치의 읽기를 polling 방식으로 하고 있지만 <interrupt.h>의 헤더 파일을 활용하여 interrupt 방식으로 하여 보다 효율적인 방법으로 제어할 수 있다.