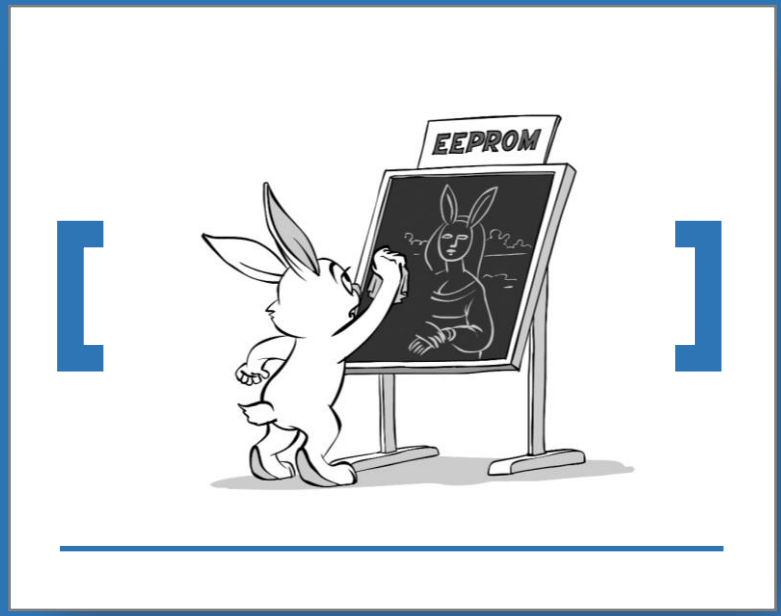
AVR EEPROM에 관한고찰

이 나 겸



AVR EEPROM에 관한고찰

이 나 겸

EEPROM의 개요

- EEPROM의 특징
  - Electrically Erasable Programmable Read Only Memory
  - 고전압으로 데이터를 지우고 다시 기록할 수 있는 메모리이다.
  - 전원이 공급되지 않아도 저장된 값이 유지된다.
  - 새로운 프로그램이 LOAD 되어도 데이터가 유지된다.
  - ATmega128은 4KByte의 EEPROM을 내장하고 있다.
  - 10만 번 쓰고 지울 수 있다.
  - 프로그램 실행 중 생성된 데이터를 유지시키기 위해 사용한다.

- EEPROM의 특성

사용자가 내용을 Byte 단위로 Read하거나 Write 할 수 있으므로 사실상 SRAM처럼 사용할 수 있는 비 휘발성 메모리이다.
Read 동작은 Access 동작이 다소 느릴지라도 SRAM(주기억장치)과 유사하므로 별 문제가 없는데 비하여
Write 동작을 수행하는 경우에는 1byte를 Write 할 때마다 수 ms이상의 시간 지연이 필요하므로 SRAM과 동일하게 사용할 수 없다.
따라서 EEPROM은 실시간으로 사용되는 변수를 저장하는 메모리나 스택 메모리로는 사용될 수 없으며,
한번 내용을 저장하면 비교적 오랫동안 이를 기억하고 있으면서 주로 이를 읽어 사용하기만 하거나 전원을 꺼도 지워져서는 안되는
중요한 데이터를 백업해 두어야 하는 설정 값 저장용 메모리로 적합하다.

AVR EEPROM에 관한고찰

이 나 겸

EEPROM과 Flash Memory의 비교

- EEPROM의 특징
  - 전기적으로만 지울 수 있는 PROM 으로 칩의 한 핀에 전기적 신호를 가하면 내부 데이터가 지워지는 롬
  - 데이터를 삭제하기 위한 이레이저가 따로 필요하지 않고 하나의 롬 라이터를 사용해서 쓰고 지울 수 있다.
  - EEPROM은 전기를 노출시켜서 한번에 1byte씩 지울 수 있기 때문에 플래시 메모리와 비교하면 매우 느리다.
  - 반복 기록 횟수에 약 10만번 정도로 횟수 제한이 있다.

- Flash Memory의 특징
  - 전기적으로 데이터를 지우고 다시 기록할 수 있는 비휘발성 컴퓨터 기억 장치
  - EEPROM에 컴퓨터 기억 장치
  - EEPROM에 비하여 메모리 셀의 구조가 매우 단순하여 대용량에 적합하고 소비전력이 적다는 장점을 가진다.
  - EEPROM과 다르게 여러 구역으로 구성된 블록 안에서 지우고 쓸 수 있다.
  - 현재는 플래시 메모리 가격이 EEPROM보다 훨씬 싸기 때문에 비휘발성 고체 상태 저장 매체가 상당량 필요한 곳에서 많이 사용된다.

차이점 EEPROM과 플래시 메모리의 가장 큰 차이는
바이트 단위로 수정이 가능한가 (EEPROM)
블록 단위로만 수정이 가능한가 (Flash Memory)에 따라 결정된다.

EEPROM 사용 레지스터

- EEAR (EEPROM Address Register)
  - 접근할 EEPROM의 주소를 지정하는 역할

EEARH/EEARL (EEPROM Address Register) 0x1F(0x3F), 0x1E(0x3E)

bit	7	6	5	4	3	2	1	0
	-	-	-	-	EEARH[11:8]			
	EEARL[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
초기값	0	0	0	0	x	x	x	x
	x	x	x	x	x	x	x	x

  - (-)로 되어있는 부분은 사용하지 않고 항상 0으로 판독한다.
  - EEAR = EEARH (상위 4비트) + EEARL (하위 4비트)
  - 4Kbyte 내의 EEPROM Address를 규정한다 (0 ~ 4095)
  - 초기화 되어 있지 않으므로 사용 전에 반드시 값을 써야한다.

AVR
EEPROM에
관한고찰

이 나 겸

EEPROM 사용 레지스터

- EEDR (EEPROM Data Register)
  - EEPROM 기록 ▶ EEAR에 지정된 번지에 write할 데이터를 보관
  - EEPROM 판독 ▶ EEAR에 지정된 번지에 read할 데이터를 보관

EEDR (EEPROM Data Register) 0x1F(0x3F), 0x1E(0x3E)

bit	7	6	5	4	3	2	1	0
	MSB							LSB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
초기값	0	0	0	0	0	0	0	0

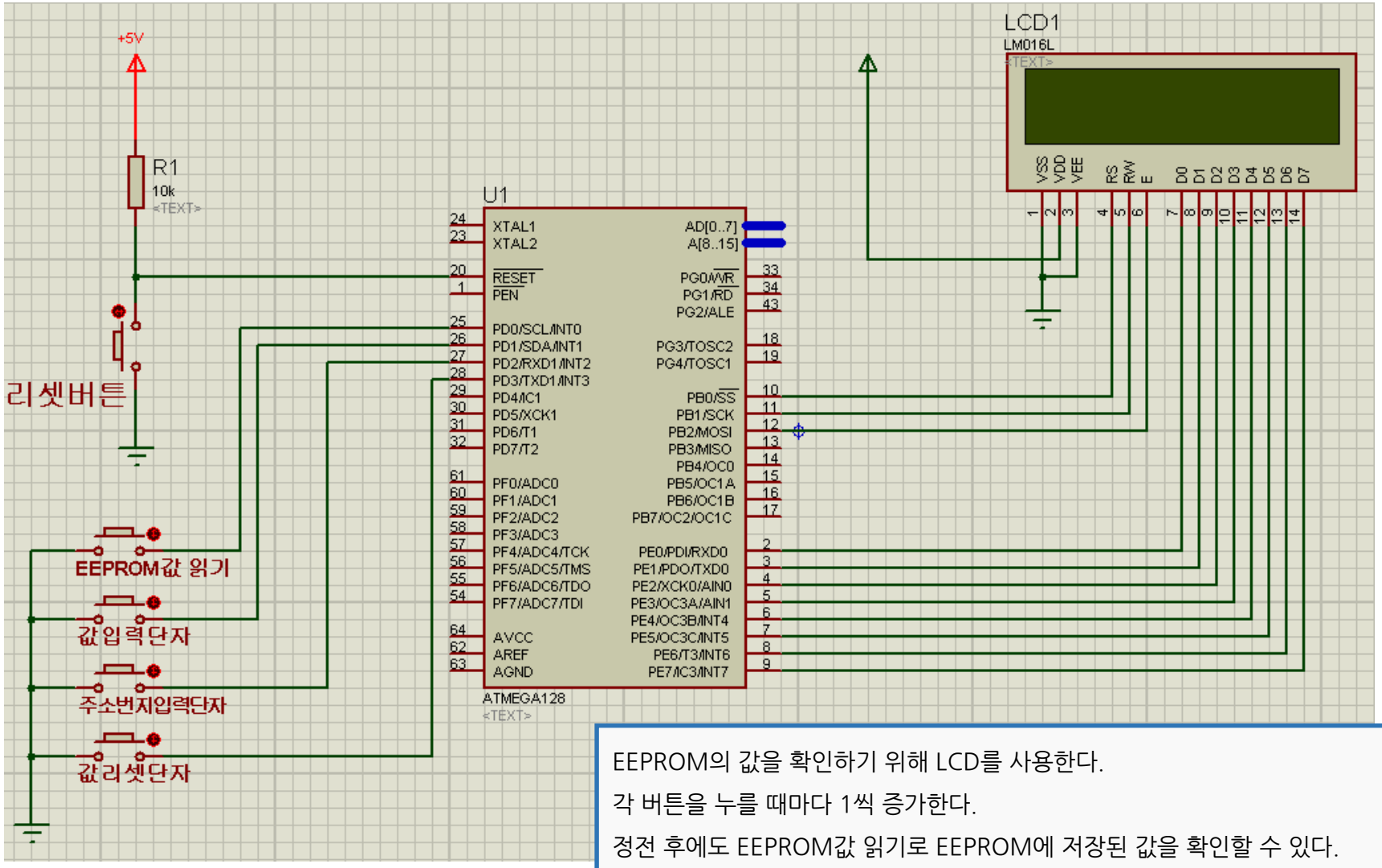
AVR EEPROM에 관한고찰

이 나 겸

AVR EEPROM에 관한고찰

이 나 겸

회로구성 - 시뮬레이션



EEPROM의 값을 확인하기 위해 LCD를 사용한다.

각 버튼을 누를 때마다 1씩 증가한다.

정전 후에도 EEPROM값 읽기로 EEPROM에 저장된 값을 확인할 수 있다.

AVR EEPROM에 관한고찰

이 나 겸

EEPROM 소스코드

- 버튼 구성 (외부 인터럽트 사용)

버튼 구성을 외부 인터럽트를 사용한 이유는 그 당시 수업시간에 폴링을 이용한 EEPROM 기본 코드를 실습하고 외부 인터럽트에 대해서 배울 수 있었다.
폴링방식은 주 프로그램 안에서 CPU 명령어를 이용하여 알고리즘에서 반복적으로 상태를 관찰해야 하지만,
인터럽트는 외부나 내부에서 발생하는 사건에 대해서 CPU 하드웨어의 도움을 받아 자동으로 감지되 ISR이 자동으로 실행되는 매커니즘을 갖는다.
버튼이 동작하지 않을때는 상태를 관찰하지 않는 인터럽트가 좀 더 효율적이라 느껴졌기 때문에 인터럽트를 사용해서 소스코드를 만들게 되었다.

```
ISR(INT0_vect)      // EEPROM 값 읽기 버튼
{
    //ee_addr = eeprom_read_byte(0);
    ee_rdata = eeprom_read_byte(ee_addr);   // 주소에 저장된 값 불러오기
    ee_wdata = ee_rdata;                    // 불러온 값을 쓰기단자에도 넣기
    flag = 1;
}
```

```
ISR(INT1_vect)      // 값 입력 버튼
{
    ee_wdata++;     // 값 입력은 버튼을 누를때마다 1씩 커짐
    _delay_ms(500);
    eeprom_write_byte(ee_addr, ee_wdata);   // 주소에 값을 저장
    //eeprom_write_byte(0, ee_addr);
    flag = 1;
}
```

```
ISR(INT2_vect)      // 주소 번지 입력 버튼
{
    ee_addr++;          // 주소는 버튼을 누를때마다 1씩 커짐
    _delay_ms(500);
    if(ee_addr>110)     // 주소의 범위는 100부터 110까지 사용
        ee_addr = 100;
    flag = 1;
}
```

```
ISR(INT3_vect)      // 값 리셋 버튼
{
    ee_wdata = 0;   // 값 리셋단자로 0부터 입력할 수 있게함
    flag = 1;
    _delay_ms(500);
}
```

코드 해석

- 주석처리가 된 상태로 코드를 실행하면 100부터 110까지의 주소 번지에 값을 넣고 리셋한 후에 각 주소에 대해 저장된 값을 불러올 수 있다.
- 주석처리 된 부분을 활성화 시키면 마지막에 값을 저장한 EEPROM 주소와 값을 확인할 수 있다.

AVR EEPROM에 관한고찰

이 나 겸

EEPROM 소스코드

- EEPROM 동작 원리

```
void    eeprom_write_byte(unsigned int address, unsigned char data)
{
    while (EECR & (1 << EEWE)); // 앞의 기록이 완료될 때 까지 기다린다.

    cli();
    EEAR = address; // 기록할 주소를 EEAR에 지정한다.
    EEDR = data;    // 데이터를 EEDR에 기록한다

    EECR |= (1 << EEMWE);   // EEMWE에 1을 기록한다,
    EECR |= (1 << EEWE);    // EEWE를 세트함으로써 EEPROM 기록을 시작한다.
    sei();
}
unsigned char   eeprom_read_byte(unsigned int address)
{
    while (EECR & (1 << EEWE)); // 이전의 write가 완료될 때까지 기다린다.
    EEAR = address;             // 읽어들일 주소를 지정한다.
    EECR |= (1 << EERE);        // EERE에 1을 기록함으로서 판독을 시작한다.

    return EEDR;                // EEDR의 데이터를 리턴한다.
}
```

- EEMWE 비트를 1로 set하면 EEPROM에 기록이 가능하다.
- EEWE 비트를 1로 set하면 EEPROM에 기록이 이루어진다.
- EECR
  EEPROM 데이터 메모리에 write나 read할 때 제어기능을 하는 레지스터

EEPROM 소스코드

- LCD 동작원리

```
LCD_init();                                 // LCD 초기화하기

sprintf(lcd_string[0], "ATMEGA128 EEPROM"); // LCD 첫 화면에 출력할 값 저장
sprintf(lcd_string[1], "  NAGYEOM:)");      // LCD 첫 화면에 출력할 값 저장

LCD_str_write(0, 0, lcd_string[0]);         // LCD 첫 화면 출력
LCD_str_write(1, 0, lcd_string[1]);         // LCD 첫 화면 출력

while (1)
{
    if (flag == 1)
    {
        sprintf(lcd_string[0], "[EEPROM ADDR]%3d", ee_addr);    // 결과를 lcd_string[]에 기록
        sprintf(lcd_string[1], "[WR]%2d [RD]%2d", ee_wdata, ee_rdata);

        LCD_str_write(0, 0, lcd_string[0]); // 기록한 lcd_string 문자열을 LCD에 디스플레이
        LCD_str_write(1, 0, lcd_string[1]);
        _delay_ms(100);                     // LCD는 계속 변화하기 때문에 delay를 걸어줘야한다.
    }

}
return 0;
```

AVR
EEPROM에
관한고찰

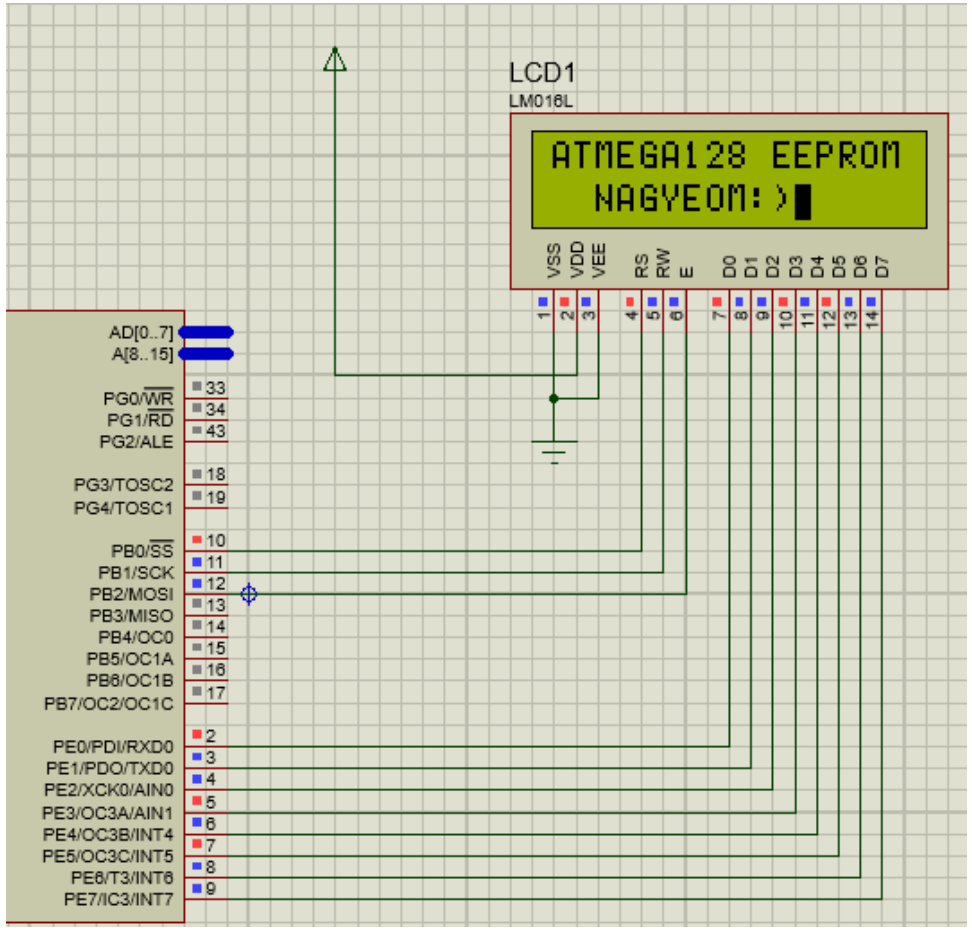
이 나 겸

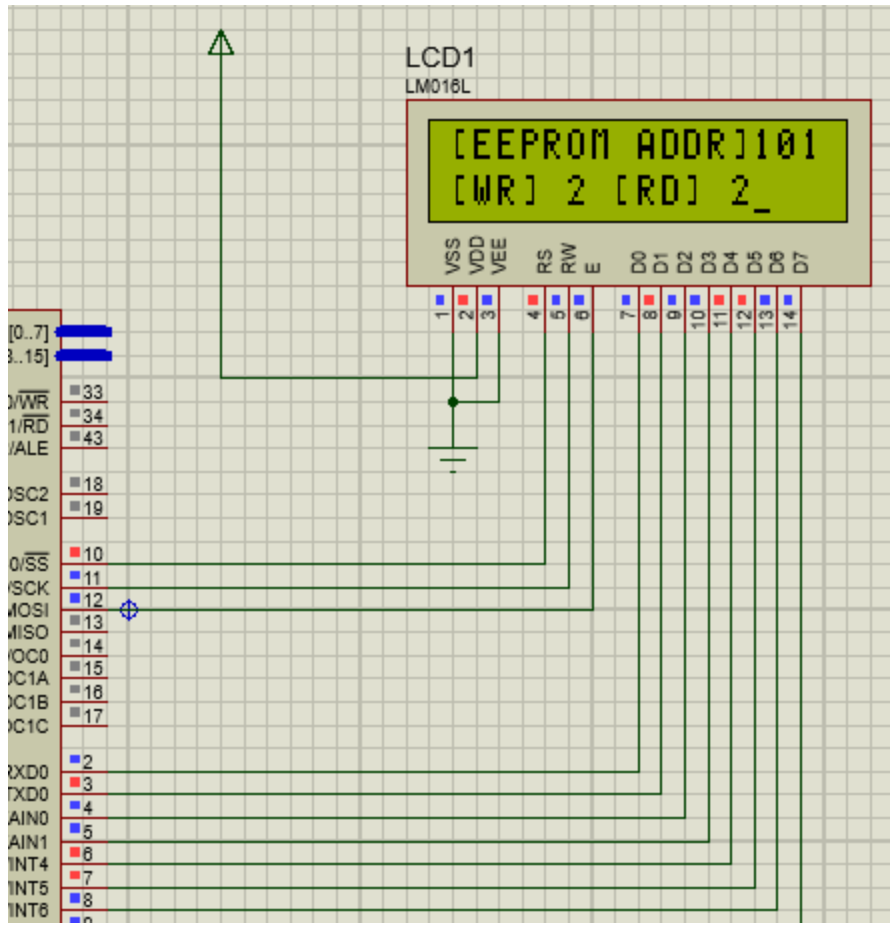
AVR EEPROM에 관한고찰

이 나 겸

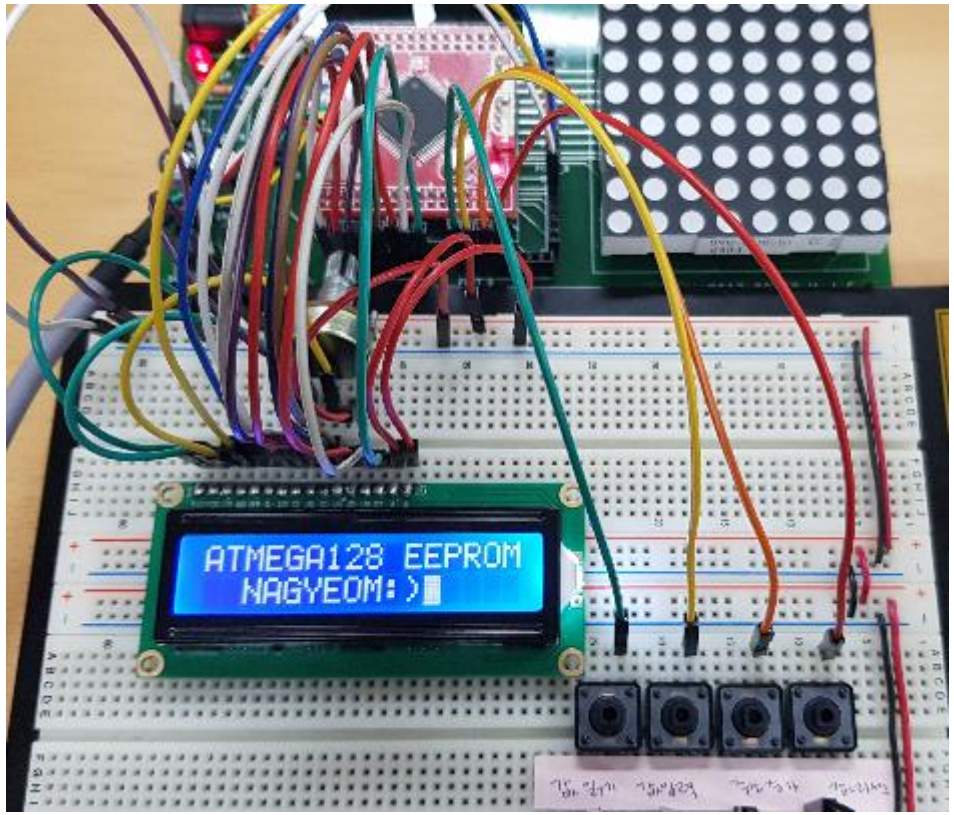
시뮬레이션 동작

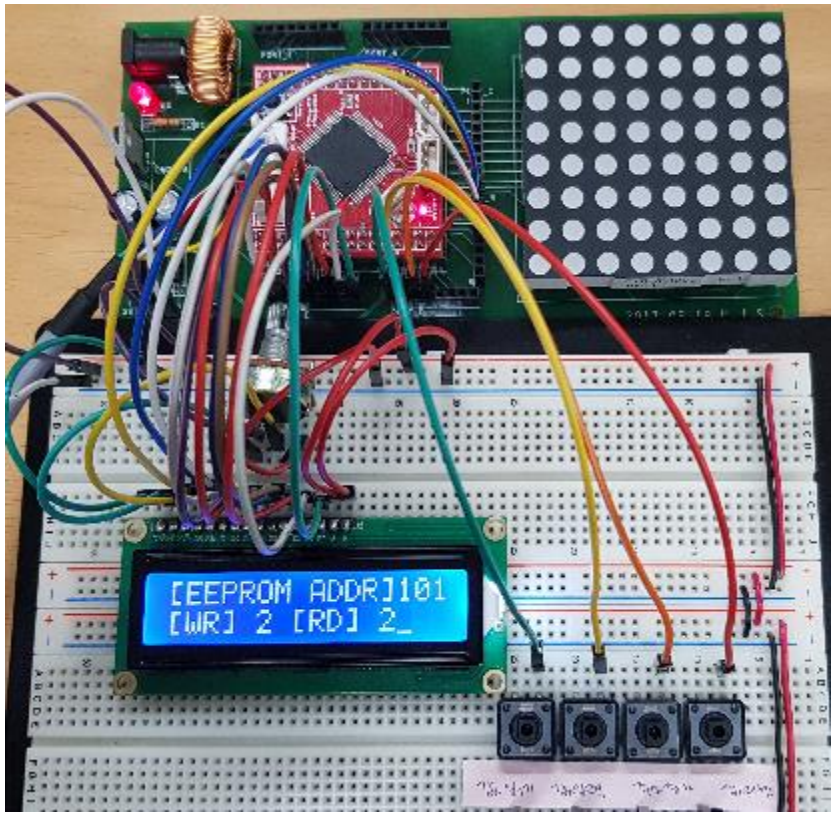
MCU : ATmega128 / Program : Proteus 7





실제 MCU 동작





AVR
EEPROM에
관한고찰

이 나 겸

영상QR

앞쪽의 영상이 제대로 안 나온다면 QR코드를 인식해주세요.

GITLAB

프로젝트에 대한 형상관리는 GITLAB을 사용하였습니다.

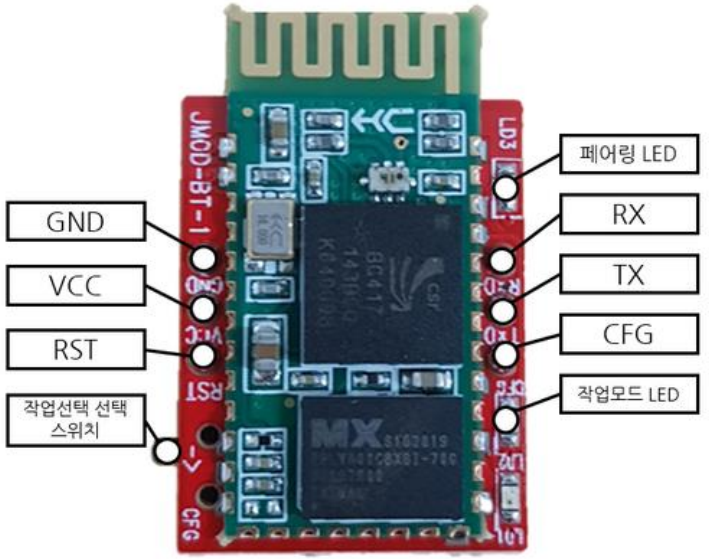
EEPROM 블루투스 통신하기

- UART 통신
  - Universal Asynchronous Reciever Transmitter
  - ATmega128에는 2개의 UART가 있다.
  - 송신과 수신을 동시에 수행하는 전이중 방식의 통신을 한다.
  - 보레이트(=전송속도)를 프로그래머가 결정할 수 있다.
  - 유선, 무선 통신이 가능하다.

    유선 : CP2192같은 모듈로 컴퓨터 터미널 프로그램을 통해 아두이노의 시리얼 모니터처럼 값을 확인할 수 있다.

    무선 : 블루투스 모듈 앱으로 AVR을 제어할 수 있다.

- 블루투스 모듈 : JMOD BT-1
  - RX, TX, GND, VCC를 사용한다.
  - UART 통신의 Baud Rate는 115200bps
  - 데이터 비트 8
  - 정지 비트 1
  - 패리티 비트 없음
  - 흐름 제어 없음



AVR EEPROM에 관한고찰

이 나 겸

UART 통신 관련 레지스터

- USCRA 레지스터

Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

USART 제어 및 상태 레지스터 A

통신 데이터 체크에 필요한 레지스터이다.
체크에는 폴링 방식과 인터럽트 방식이 있다.

- USCRB 레지스터

Bit	7	6	5	4	3	2	1	0	
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

USART 포트의 송수신 기능 활성화 레지스터

송수신에 대해서 폴링방식과 인터럽트 방식에 대해서 설정한다.

AVR
EEPROM에
관한고찰

이 나 겸

UART 통신 관련 레지스터

- USCRC 레지스터

Bit	7	6	5	4	3	2	1	0	
	–	UMSELn	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

USART 포트의 송수신 동작을 제어하는 기능을 수행

동기모드와 비동기 모드를 선택한다.
USART가 사용하는 패리티모드를 설정할 수 있다.

- UDR 레지스터

Bit	7	6	5	4	3	2	1	0	
	RXBn[7:0]								UDRn (Read)
	TXBn[7:0]								UDRn (Write)
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

USART 데이터 레지스터

송신용, 수신용 모두 사용한다.
UDR에 데이터를 넣으면 송신이 시작되며, 수신이 완료되었을 경우 UDR에서 읽어내면 된다.

AVR
EEPROM에
관한고찰

이 나 겸

UART 통신 관련 레지스터

- UBRR 레지스터

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRRn[11:8]				UBRRnH
	UBRRn[7:0]								UBRRnL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

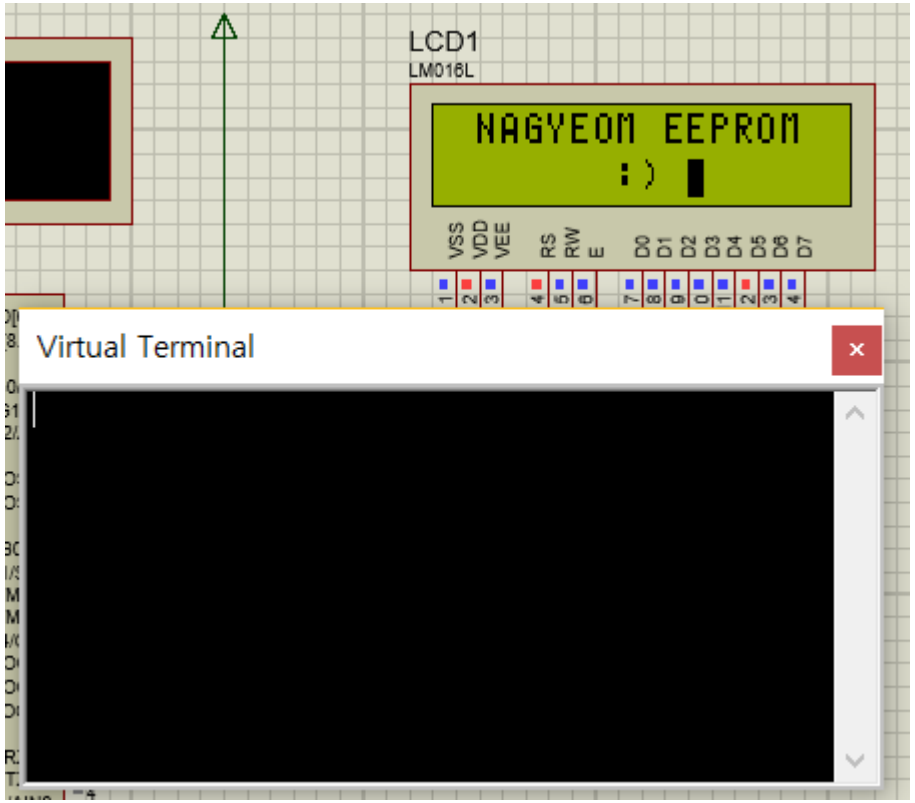
USART Baud Rate 레지스터

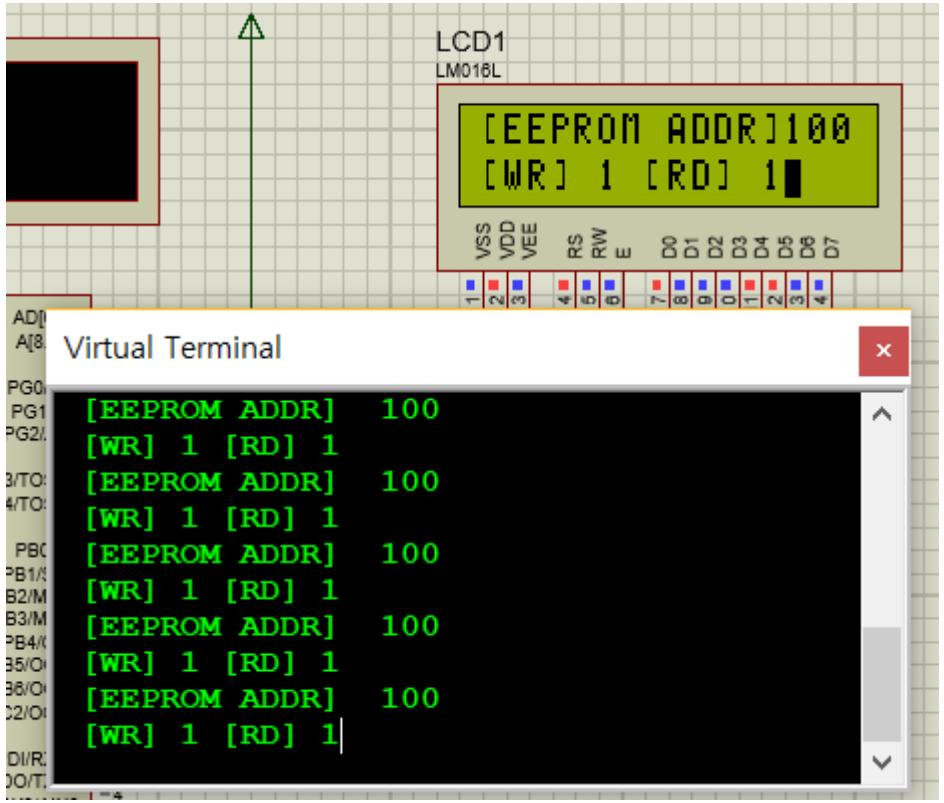
AVR
EEPROM에
관한고찰

이 나 겸

시뮬레이션 동작

MCU : ATmega128 / Program : Proteus 8

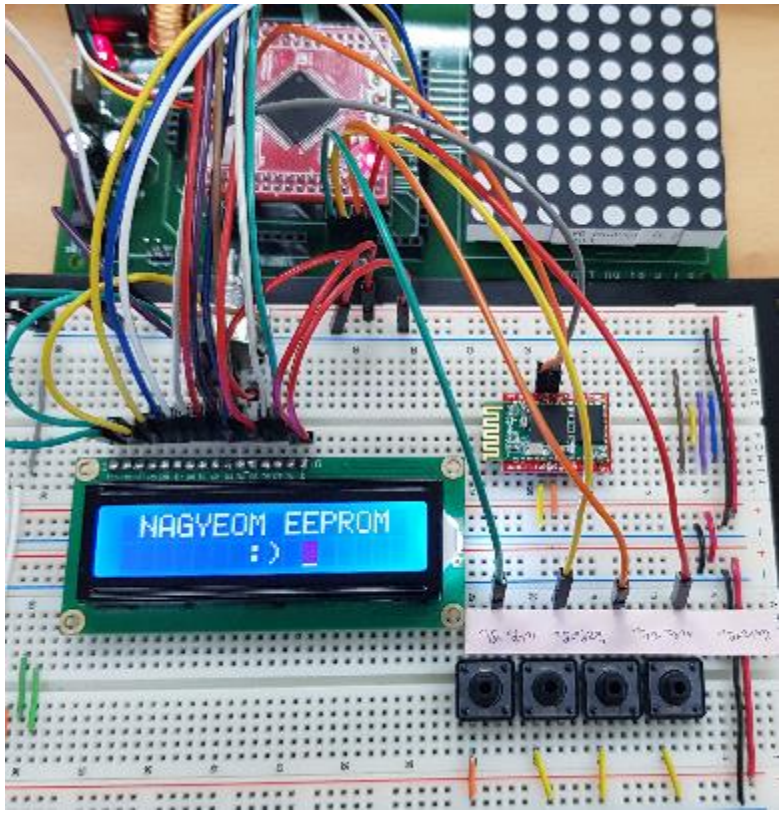




AVR EEPROM에 관한고찰

이 나 겸

실제 MCU 동작




```
[EEPROM ADDR]  100
[WR] 8 [RD] 7
[EEPROM ADDR]  100
[WR] 8 [RD] 7
[EEPROM ADDR]  100
[WR] 8 [RD] 7
[EEPROM ADDR]  100
[WR] 8 [RD] 7
[EEPROM ADDR]  100
[WR] 8 [RD] 7
[EEPROM ADDR]  100
[WR] 0 [RD] 7
[EEPROM ADDR]  100
[WR] 0 [RD] 7
[EEPROM ADDR]  100
[WR] 8 [RD] 8
[EEPROM ADDR]  100
[WR] 0 [RD] 8
[EEPROM ADDR]  100
[WR] 0 [RD] 8
[EEPROM ADDR]  100
[WR] 0 [RD] 8
[EEPROM ADDR]  102
[WR] 0 [RD] 8
[EEPROM ADDR]  102
[WR] 0 [RD] 8
[EEPROM ADDR]  103
[WR] 0 [RD] 8
[EEPROM ADDR]  103
[WR] 0 [RD] 8
[EEPROM ADDR]  103
[WR] 0 [RD] 8
```

```
[WR]22 [RD]22
[EEPROM ADDR]  105
[WR]22 [RD]22
[EEPROM ADDR]  105
[WR]22 [RD]22
[EEPROM ADDR]  105
[WR]22 [RD]22
[EEPROM ADDR]  105
[WR]22 [RD]22
[EEPROM ADDR]  105
[WR]22 [RD]22
[EEPROM ADDR]  105
[WR]22 [RD]22
[EEPROM ADDR]  105
[WR]22 [RD]22
[EEPROM ADDR]  105
[WR]22 [RD]22
[EEPROM ADDR]  105
[WR]22 [RD]22
[EEPROM ADDR]  105
[WR]22 [RD]22
[EEPROM ADDR]  105
[WR]22 [RD]22
[EEPROM ADDR]  105
[WR]22 [RD]22
[EEPROM ADDR]  105
[WR]22 [RD]22
[EEPROM ADDR]  105
[WR]22 [RD]22
```

AVR EEPROM에 관한고찰

이 나 겸

영상QR

앞쪽의 영상이 제대로 안 나온다면 QR코드를 인식해주세요.

GITLAB

프로젝트에 대한 형상관리는 GITLAB을 사용하였습니다.

EEPROM에 관한 고찰

- EEPROM을 공장의 생산라인과 같은 곳에서
  단순히 물건 카운트를 하거나 단순한 작업에 대해서 쓰인다면
  정전과 같은 전기가 끊기는 상황이 오더라도
  이전의 데이터가 지워지지 않아서 작업을 계속 진행할 수 있다.

- 현재는 100번부터 110번까지의 주소를 사용하고 있다.
  하지만 EEPROM은 4KByte의 용량을 가지고 있기 때문에
  약 4000개의 주소번지를 가질 수 있다.
  바이트 단위로 데이터를 저장할 수 있으므로
  다수의 생산 라인의 작업을 하나의 AVR로 처리할 수 있을 것이다

- 공장 생산라인과 비슷한 환경의 실험 조건을 만들려면
  버튼 입력 단자를 적외선 센서 버튼으로 만들고
  LCD와 블루투스 통신으로 확인 뿐만 아니라
  제어까지 할 수 있다면 좀 더 밀접한 실험 환경이 될 것이다.

AVR
EEPROM에
관한고찰

이 나 겸