



AVR – 내부 EEPROM

뉴테크놀로지 컴패니
대표 류대우

[EEPROM이란?



- Electrically Erasable Programmable Read Only Memory
- 사용자가 메모리 내의 내용을 수정할 수 있는 **롬**으로서, 정상보다 더 높은 전압을 이용하여 반복적으로 지우거나, 다시 프로그램 (기록)할 수 있다.
- Controlled by 2-wire, 3-wire, and SPI protocols : as I2C, SPI, TWI

[EEPROM in ATMEGA128]

- 4 k bytes data EEPROM memory
- Single byte can be read and written
- At least 100,000 write/erase cycles
- Three Registers
 - EEPROM Address Register
 - EEPROM Data Register
 - EEPROM Control Register
- Programming mode
 - SPI, JTAG, Parallel programming mode

[EEPROM Address Register]

■ EEARH & EEARL

- 4k byte 내의 EEPROM Address를 규정한다.(0~4096)
- 초기화 되어 있지 않으므로, 사용 전에 반드시 값을 써야한다.

15	14	13	12	11	10	9	8	
-	-	-	-	EEAR11	EEAR10	EEAR9	EEAR8	EEARH
EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
7	6	5	4	3	2	1	0	

[EEPROM 프로그래밍 시간]

Table 2. EEPROM Programming Time

Symbol	Number of Calibrated RC Oscillator Cycles ⁽¹⁾	Typ Programming Time
EEPROM Write (from CPU)	8448	8.5 ms

Note: 1. Uses 1 MHz clock, independent of CKSEL-fuse settings.

[EEPROM Control Register]

■ EECR

○ EERIE

- 1을 쓰면, SREG의 I-bit가 set인 경우 EEPROM 준비 인터럽트가 enable됨
- 1이면 EEWEIF가 클리어 될 때, 항상 인터럽트를 발생시킴

○ EEMWE

- EEPROM MASTER WRITE ENABLE

○ EEWEIF

- EEPROM WRITE ENABLE

○ EERE

- EEPROM READ ENABLE

[퓨즈비트 EESAVE 셋팅 이유]

- EESAVE를 ON 시켜서, 프로그램 할 때마다 EEPROM 데이터가 바뀌는 불편을 없앤 후에 프로그램 하면, Flash가 써질 때마다 EEPROM도 다시 써야 하는 불편함이 사라진다.
- 즉, EEPROM 쓰는 시간이 무척 길데, 이 EEPROM 프로그램 쓰는 시간이 단축되며, 기본 값을 유지한 채로 테스트가 가능해 진다.

[Fuse bit of ATMEGA128]

- Extended fuse byte(→0xff로, 128모드로 사용)

Extended Fuse Byte	Bit No.	Description	Default Value
—	7	—	1
—	6	—	1
—	5	—	1
—	4	—	1
—	3	—	1
—	2	—	1
M103C ⁽¹⁾	1	ATmega103 compatibility mode	0 (programmed)
WDTON ⁽²⁾	0	Watchdog Timer always on	1 (unprogrammed)

[Fuse bit of ATMEGA128]

- Fuse high byte(0xD1로) → EESAVE ON 시킴

Fuse High Byte	Bit No.	Description	Default Value
OCDEN ⁽⁴⁾	7	Enable OCD	1 (unprogrammed, OCD disabled)
JTAGEN ⁽⁵⁾	6	Enable JTAG	0 (programmed, JTAG enabled)
SPIEN ⁽¹⁾	5	Enable Serial Program and Data Downloading	0 (programmed, SPI prog. enabled)
CKOPT ⁽²⁾	4	Oscillator options	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the Chip Erase	1 (unprogrammed, EEPROM not preserved)
BOOTSZ1	2	Select Boot Size (see Table 112 for details)	0 (programmed) ⁽³⁾
BOOTSZ0	1	Select Boot Size (see Table 112 for details)	0 (programmed) ⁽³⁾
BOTRST	0	Select Reset Vector	1 (unprogrammed)

[Fuse bit of ATMEGA128]

- Fuse low byte(0xEF)로

Fuse Low Byte	Bit No.	Description	Default Value
BODLEVEL	7	Brown out detector trigger level	1 (unprogrammed)
BODEN	6	Brown out detector enable	1 (unprogrammed, BOD disabled)
SUT1	5	Select start-up time	1 (unprogrammed) ⁽¹⁾
SUT0	4	Select start-up time	0 (programmed) ⁽¹⁾
CKSEL3	3	Select Clock source	0 (programmed) ⁽²⁾
CKSEL2	2	Select Clock source	0 (programmed) ⁽²⁾
CKSEL1	1	Select Clock source	0 (programmed) ⁽²⁾
CKSEL0	0	Select Clock source	1 (unprogrammed) ⁽²⁾

[ICCAVR에서의 사용법 I]

- EESAVE를 1로 두어 unprogram 한 후에,
- `#pragma data:eeeprom`
`int foo = 0x1234;`
`char table[] = { 0, 1, 2, 3, 4, 5 };`
`#pragma data:data`
- `... int i;`
- `EEPROM_READ((int)&foo, i);` // i now has 0x1234
- 위와 같이 사용하면, ISP 다운로드 시 1번지부터 순서대로 EEPROM에 써지게 된다.(0번지는 사용하지 않는다.)

[ICCAVR에서의 사용법 II]

■ [EEPROM에 바이트 단위로 쓸 때]

- ```
static char MEMORY[3] = {0x81, 0x01, 0x06};
EEPROMwrite(i, MEMORY[0]);
EEPROMwrite(i+1, MEMORY[1]);
EEPROMwrite(i+2, MEMORY[2]);
```

## ■ [EEPROM에 바이트 단위로 읽을 때]

- ```
char ABSOL[3];  
ABSOL[0] = EEPROMread(i);  
ABSOL[1] = EEPROMread(i+1);  
ABSOL[2] = EEPROMread(i+2);
```

■ 사용법

- `#include <eeprom.h>`를 하신 후에,
- `presetnum` 을 정한 뒤(예를 들어 0 또는 1 또는 2 등등)
예를 들어서, `i=presetnum*10;` 이라고 하고, 주소를 정한 뒤 사용한다.

[두 바이트 씩 읽고 쓰기]

- 1번지에서 두 바이트 읽어서 변수 i에 넣기
 - `int i;`
 - `EEPROM_READ(0x1, i);`
- 1번지부터 두 바이트 쓰기(변수 i값)
 - `int i;`
 - `EEPROM_WRITE(0x1, i);`

[EEPROM_write 함수

```
void EEPROM_write(unsigned int uiAddress, unsigned char ucData)
{
    /* Wait for completion of previous write */
    while(EECR & (1<<EWE))
        ;
    /* Set up address and data registers */
    EEAR = uiAddress;
    EEDR = ucData;
    /* Write logical one to EEMWE */
    EECR |= (1<<EEMWE);
    /* Start eeprom write by setting EWE */
    EECR |= (1<<EWE);
}
```

[EEPROM_read 함수



```
unsigned char EEPROM_read(unsigned int uiAddress)
{
    /* Wait for completion of previous write */
    while(EECR & (1<<EWE))
        ;
    /* Set up address register */
    EEAR = uiAddress;
    /* Start eeprom read by writing EERE */
    EECR |= (1<<EERE);
    /* Return data from data register */
    return EEDR;
}
```

[내부함수]

- **unsigned char EEPROMread(int location)**
 - 규정된 EEPROM 위치에서 1바이트를 읽음.
- **int EEPROMwrite(int location, unsigned char byte)**
 - 규정된 EEPROM 위치에 1바이트를 씀. 성공 시 리턴값 : 0
- **void EEPROMReadBytes(int location, void *ptr, int size)**
 - Ptr에 의해 지정된 버퍼로, EEPROM location으로부터 size만큼 읽는다.
- **void EEPROMWriteBytes(int location, void *ptr, int size)**
 - Ptr에 의해 지정된 버퍼의 내용과 location로부터 시작하여, size만큼 바이트들을 쓴다.

[실시간 시스템에서 EEPROM]

- `#include "rteeprom.h"`
- `// EEPROM을 읽고 쓰는 데 준비 되었는 지 확인함`
- `// 리턴값`
 - 0이 아닌 값 : 준비되면,
 - 0 : 준비되지 않음
- `unsigned char RTEEPROMReady(void)`
- `{`
- `return !(EECR & 0x02);`
- `}`

]

```
// EEPROM Write를 초기화 하는 함수
// EEPROM이 준비 되었을 때만 특정 데이터 바이트를 특정 위치에 쓴다. (준비 안되면, 실패할 수 있음)
void RTEEPROMwrite(int location, unsigned char databyte)
{
    unsigned char savedSREG;
    EEAR = location;
        // set address
    EEDR = databyte;
        // set data
    savedSREG = SREG;
        // keep setting so it can be restored
    CLI();
                                                                    // disable interrupts

    EECR |= BIT(EEMWE);
    // set "write enable" bit
    EECR |= BIT(EWE);
    // set "write" bit
    SREG = savedSREG;
        // restore SREG
    EEAR = 0;
}
```

[RTEEPROMread

- // EEPROM에서 데이터를 읽는 함수
- // EEPROM이 준비 되었을 때만 특정 데이터 바이트를 특정 위치에 읽는다. (준비 안되면, 실패할 수 있음)
- unsigned char RTEEPROMread(int location)
- {
- EEAR = location;
 // set address
- EECR |= BIT(EERE);
 // set "read enable" bit
- EEAR = 0;
- return (EEDR);
- }