【ZBM v1.2 활용】

무선 제어 예제 프로젝트

OL마이크로웨이브

http://olmicrowaves.com

목 차

1. 개요

본 예제 프로젝트에서는 **IEEE 802.15.4** 기술에 기반한 무선 제어 시스템을 구현해 본다. 무선 제어 시스템은 하나의 마스터와 여러 개의 슬래이브로 구성되어, 마스터에서 특정한 슬래이브로 제어 신호를 보내거나 정보를 수집하는 기능을 수행한다. 따라서 본 예제 프로젝트를 통하여 다음과 같은 기술에 용이하게 접근할 수 있다.

- **IEEE 802.15.4** 및 지그비 기술
- **AVR(Atmega128)** 설계 및 펌웨어 제작
- **1:1** 혹은 **1:N** 무선 제어 및 센싱 기술
- **RS232** 시리얼통신 기술
- **하이퍼터미널** 사용

본 자료를 통해 지그비 무선 제어 기술을 기반으로 프로젝트를 준비중이거나 응용시스템을 제작하고자 하는 사용자들에게 “Hello!” 역할과 더불어 지그비 활성화를 위한 다양한 아이디어가 제공되기를 기대한다.

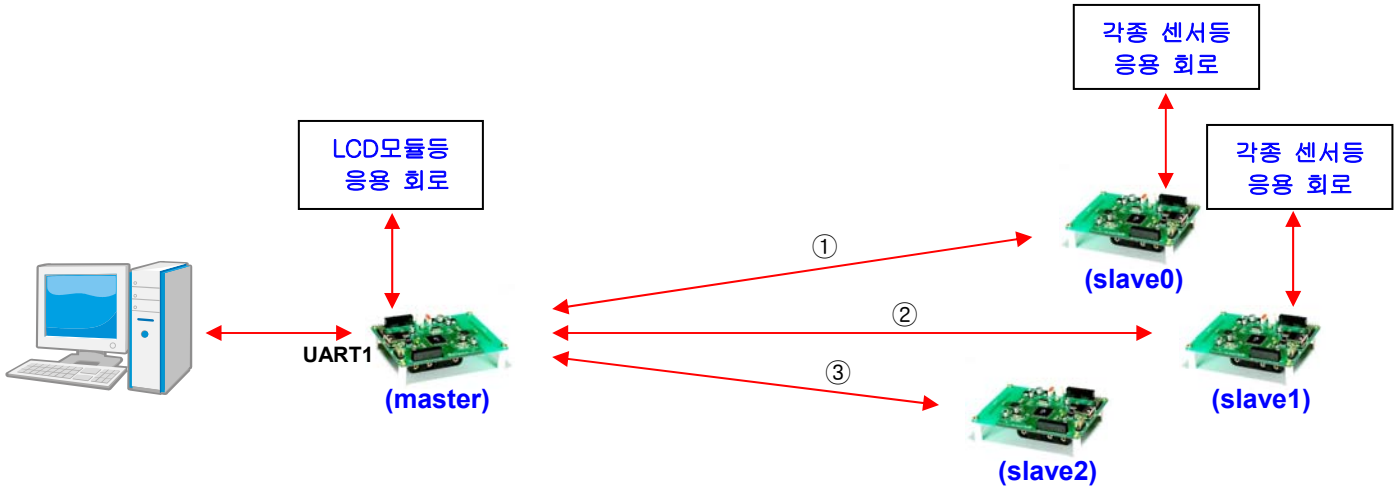
2. 활용 분야

IEEE 802.15.4 및 지그비 기반의 무선제어 및 센싱 기술은 다양한 분야에서 기존의 기술에 비해 보다 효과적으로 적용될 수 있다. 대표적인 응용 사례는 다음과 같다.

- **재해/안전** : 산불감시, 건물 화재감시, 재난/재해 조기 경보, 교각/터널 안전관리
- **홈네트워크** : 가전제품 제어, 리모콘
- **IBS(지능형 빌딩 시스템)** : 조명제어, 설비관리, 보안시스템
- **ITS(지능형 교통 시스템)** : 실시간 교통 제어, 물류/유통 관리
- **기상/환경** : 실시간 기상정보, 공해/폐수 감시
- **의료** : 환자 상태 모니터링, 재활시스템
- **텔레매틱스** : 원격검침, 원격 데이터 수집
- **사회복지** : 노약자 및 장애인 보호 시스템
- **기타** : 공장자동화, 농작물 관리

3. 무선 제어 예제 시스템 구조

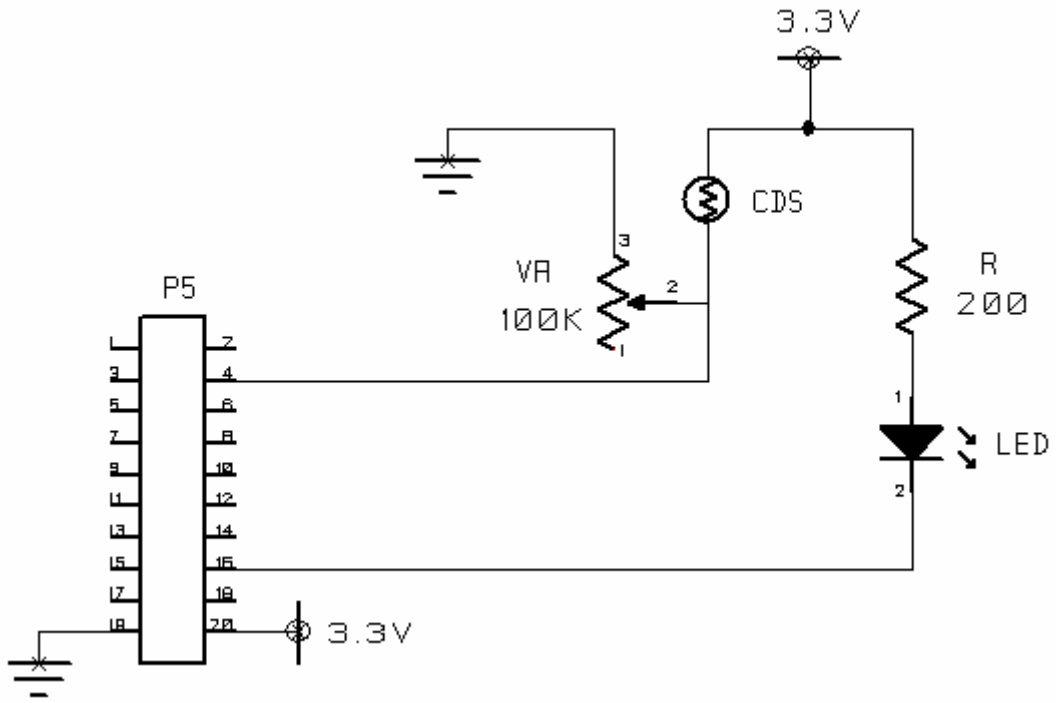
본 무선 제어 예제 시스템은 하나의 마스터와 여러 개의 슬레이브로 구성되어 마스터가 각 슬레이브에게 주기적으로 명령을 하달하고 응답을 수집하는 구조이다. 무선 제어 동작과 센싱 동작을 가장 간단하게 구현하기 위하여, 마스터는 PC의 **하이퍼터미널**을 통해 제어 명령을 수신하여 슬레이브로 전달하고, 슬레이브는 해당 명령에 상응하는 동작을 수행하고 응답을 마스터로 리턴하고, 마스터는 그 응답을 PC의 **하이퍼터미널**로 출력하도록 하는 형태로 프로그램을 제작해보도록 한다. 슬레이브의 개수는 사용자의 필요에 따라 자유롭게 선정하면 되며, **1:1 무선제어** 혹은 **1:N 무선제어**를 구현할 수 있다. 구체적으로, 제어의 종류는 LED를 사용한 **조명제어**(ON/OFF)로 설정하고, 센싱의 종류는 조도센서를 사용한 **조도값 센싱**으로 설정한다. 따라서 LED회로와 조도센서 회로를 구현한 도터보드가 별도로 제작되어 슬레이브 **ZBM v1.2**의 확장커넥터에 장착될 수 있도록 해야 한다. 또한, RSSI(전파수신세기) 측정 명령을 추가하여 슬레이브의 RSSI 값을 측정할 수 있도록하여, 도터보드가 없더라도 마스터와 슬레이브 간의 무선통신을 수행해볼 수 있도록 한다.

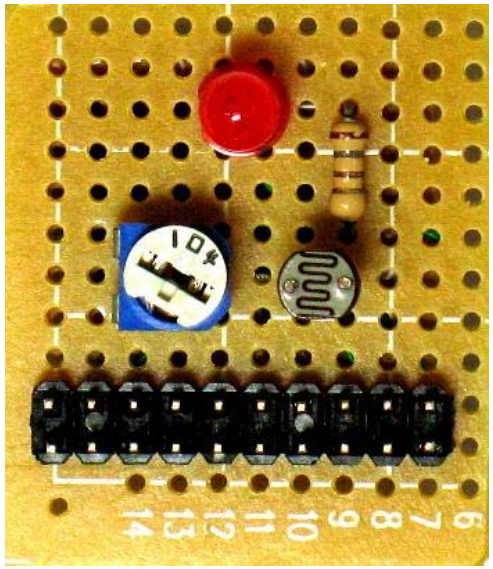


4. 도터보드 제작

도터보드는 **ZBM v1.2**의 확장커넥터에 장착되는 형태로 제작한다.

4-1. 기본 회로 구성





4-2. 소프트웨어 처리

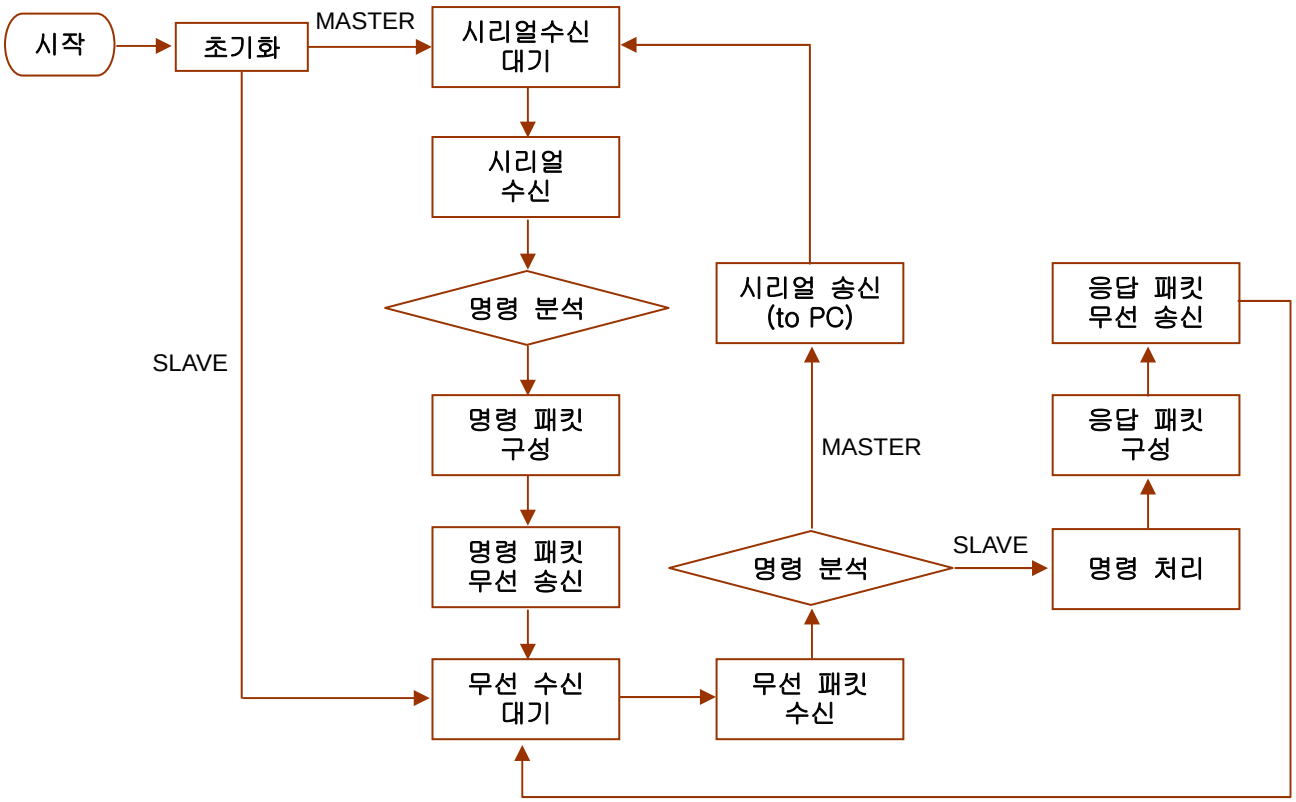
예제 프로젝트 파일 중에서 **hal_zbm.h**의 '**I/O 포트 정의**' 부분에 PORTB의 PB6을 조명 제어신호 출력 핀으로 사용하기 위해 LED라고 지정하고, PORTF의 PF2를 ADC2 입력으로 사용하기 위해 ADC_INPUT_2로 지정한다.

```
//-------------------------------------------------------------------------------------------------
// Port B
//-------------------------------------------------------------------------------------------------
…
…
#define LED                 6 // PB.6 - Output: Light control
…

//-------------------------------------------------------------------------------------------------
// Port F
//-------------------------------------------------------------------------------------------------
…
…
#define ADC_INPUT_2     2 // PF.2 - ADC2
…
```

프로그램 구현이 편리하도록 **hal_zbm.h**의 '**사용자 추가 매크로 - 외부장치 접속 관련**' 부분에 조명제어 명령 매크로를 정의한다.

```
#define LIGHT_ON()      PORTB &= 0xBF

#define LIGHT_OFF()     PORTB |= 0x40
```

5. 펌웨어 프로그램 순서도



6. 펌웨어 프로그램 제작

지금까지 설명한 내용을 바탕으로 펌웨어 프로그램을 제작해보기로 한다.
프로그램은 마스터와 슬레이브로 구분하여 제작하지 않고, 하나의 프로그램 안에서 마스트인 경우와 슬레이브인 경우에 대해 선택적으로 처리하도록 구현한다.
펌웨어 제작 방법은, **ZBM v1.2** 응용 프로그램 개발을 용이하게 할 수 있도록 미리 제작된 범용 소스인 **PROTO**를 활용한다.

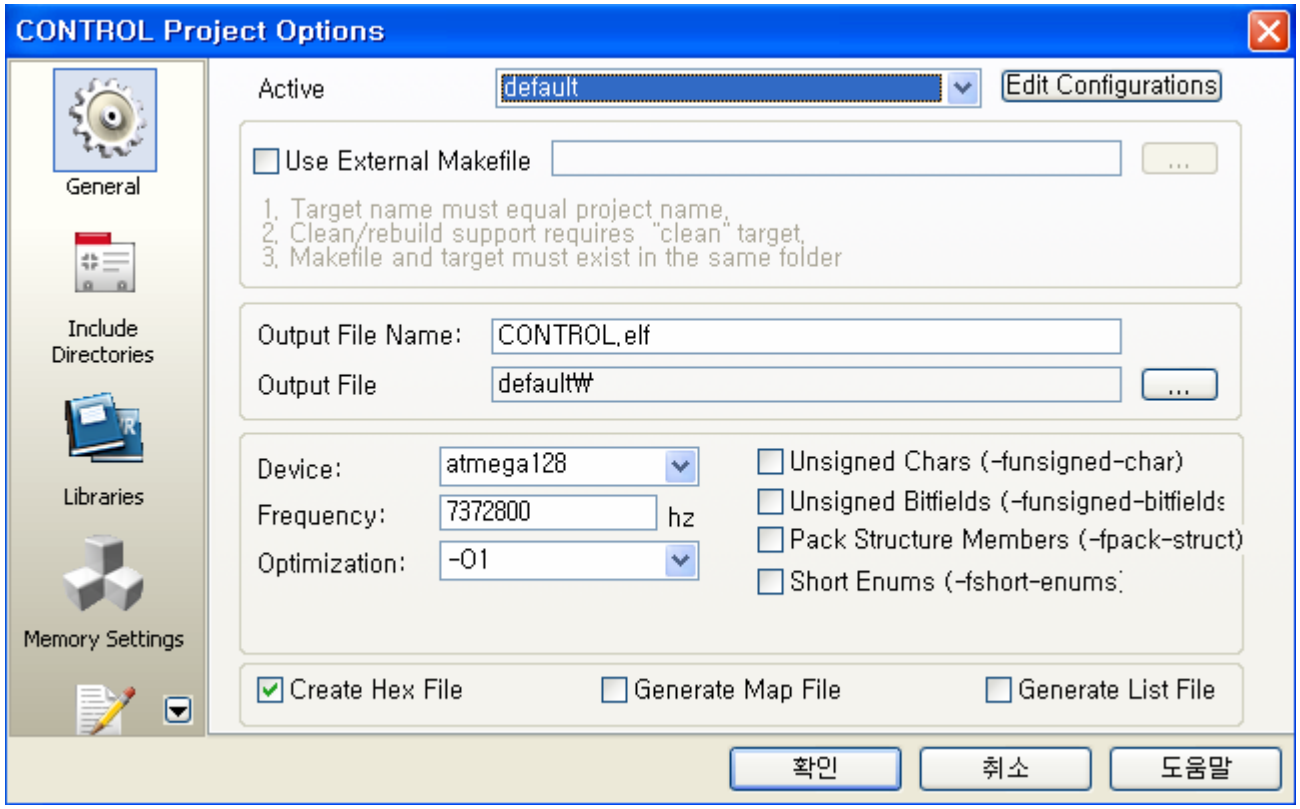
[Step.1] 프로젝트 생성

① 프로젝트를 저장할 디렉토리를 하나 만들고(예, **CONTROL**) **AVR Studio 4**를 실행한 후 **Project** 메뉴에 있는 **New Project**를 클릭한다.
(※ 디렉토리와 프로젝트 이름에는 한글을 사용하지 않도록 한다.)

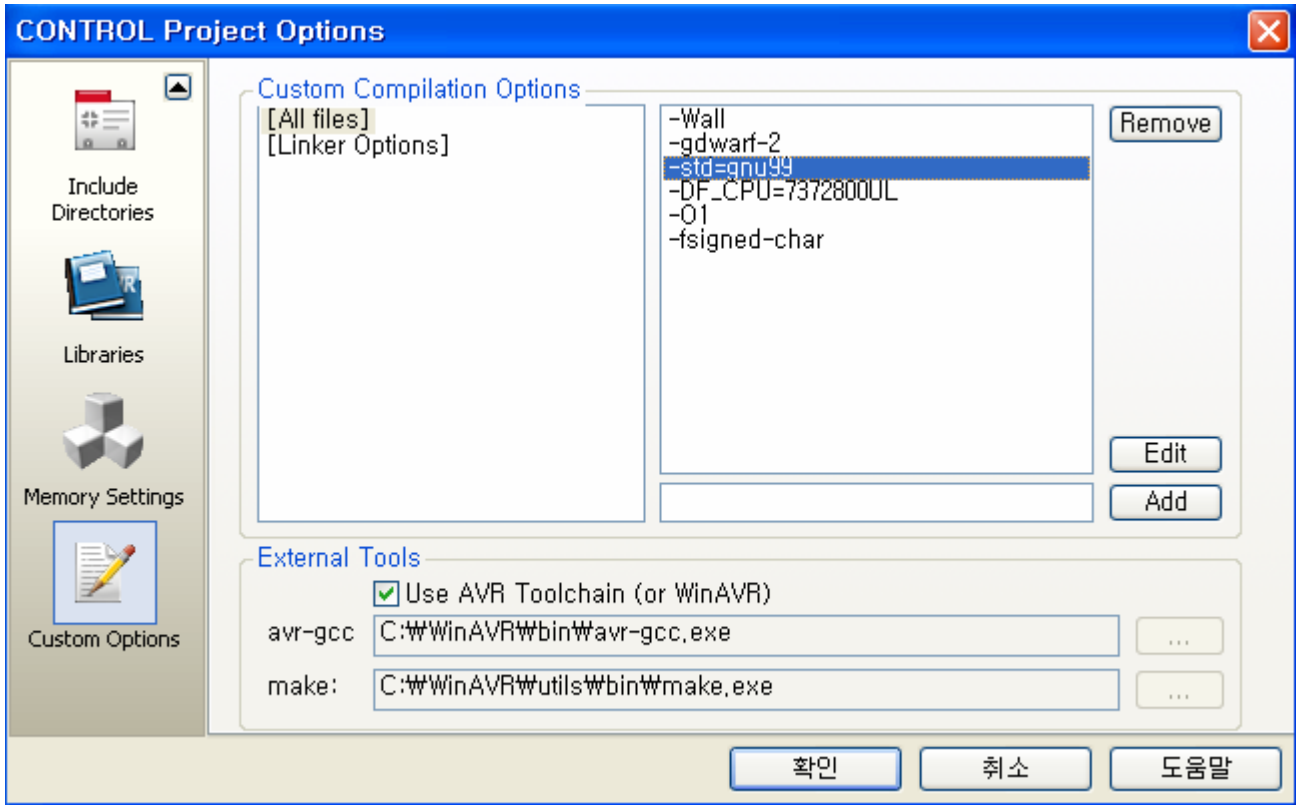
② **Project type**에서 **AVR GCC**를 선택하고, **Project name**에 적당한 이름을 넣고(예, **CONTROL**), **Create initial file**을 체크 해제하고, **Location**에 앞에서 만든 프로젝트 디렉토리를 지정해주고, **Next**를 클릭한다.

③ **Debug platform**을 **JTAG ICE**로 선택하고, **Device**를 **ATmega128**로 선택하고 **Finish**를 클릭한다. 그러면 새로운 프로젝트가 생성된다.

④ **Project** 메뉴에 있는 **Configuration Options**를 선택하여 열고, 좌측의 아이콘 창에서 **General**을 선택하여 열고 아래 화면과 같이 입력 및 체크한다.



아이콘 창을 아래로 스크롤하여 **Custom Options**를 선택하여 열고 **-std=gnu99**가 등록되어 있지 않으면 입력하고 **ADD**를 클릭하여 등록한 후 **확인**을 클릭한다.



⑤ 프로젝트 디렉토리 아래에, **PROTO** 디렉토리에 있는 3개의 디렉토리를 복사해 넣는다.(**apps**, **include**, **lib**)

⑥ **Project** 메뉴의 **Configuration Options**를 다시 선택하여 열고, 좌측의 아이콘 창에서 **Include Directories** 를 선택하여 열고, 프로젝트 디렉토리를 선택하여 등록한다.

⑦ 화면 좌측의 프로젝트매니저 창에서 **Source Files**를 선택하고, 마우스 우측 버튼을 클릭하고, **Add Existing File(s)...**를 클릭한 후 **lib** 디렉토리 및 그 하위의 디렉토리에 있는 모든 **.c** 파일을 선택하여 등록한다.

⑧ 화면 좌측의 프로젝트매니저 창에서 **Source Files**를 선택하고, 마우스 우측 버튼을 클릭하고, **Add Existing File(s)...**를 클릭한 후 **apps** 디렉토리에 있는 **control.c** 파일을 선택하여 등록한다.
화면 좌측의 프로젝트매니저 창에서 **Source Files**를 선택하고, 마우스 우측 버튼을 클릭하고, **Add Existing File(s)...**를 클릭한 후 **apps** 디렉토리에 있는 **master.c** 파일을 선택하여 등록한 후 **Rename File...**하여 파일명을 바꾼다.(예, **control.c**)

⑨ **Build** 메뉴에서 **Rebuild All**을 클릭하여 컴파일 및 **hex**파일을 생성한다. (※ 오류가 발생하지 않아야 한다.)

[Step.2] hal_zbm.h 수정

hal_zbm.h는 **ATmega128(A,L)**의 I/O핀과 외부장치와의 핀 연결에 관한 초기화 매크로나 간단한 명령 매크로를 정의한다.
외부장치는 외부메모리, LCD, 각종 센서등, **ZBM v1.2**의 확장커넥터와 접속되는 모든 장치를 말한다.
따라서 사용자는 외부확장 커넥터(P4, P5)를 통해서 응용보드를 제어하려면 해당 I/O핀을 **hal_zbm.h**에 등록하고 초기화할 수 있도록 해야한다.
본 예제에서는 마스터와 슬래이브에 동일한 프로그램을 제작하여 사용한다. **ZBM v1.2**가 스래이브로 사용되는 경우에는 도터보드가 장착되기 때문에 **4-2** 절에서 설명한 바와 같이 외부 도터보드와의 접속을 위한 사항을 적용한다.

① 프로젝트매니저 창에서 **External Dependancies**를 열고 **hal_zbm.h**를 더블클릭하여 편집창에 띄운다.

② **I/O 포트 정의** 부분에 사용하고자 하는 핀을 추가로 정의한다.

```
//-------------------------------------------------------------------------------------------------
// Port B
//-------------------------------------------------------------------------------------------------
#define CSN             0   // PB.0 - Output: SPI Chip Select (CS_N)
#define SCK             1   // PB.1 - Output: SPI Serial Clock (SCLK)
#define MOSI            2   // PB.2 - Output: SPI Master out - slave in (MOSI)
#define MISO            3   // PB.3 - Input:   SPI Master in - slave out (MISO)
//#define               4   // PB.4 - 사용가능
//#define               5   // PB.5 - 사용가능
#define LED             6   // PB.6 - Output: Light control
//#define               7   // PB.7 - 사용가능

…
…

//-------------------------------------------------------------------------------------------------
// Port F
//-------------------------------------------------------------------------------------------------
//#define               0   // PF.0 - 사용가능
//#define               1   // PF.1 - 사용가능
#define ADC_INPUT_2     2   // PF.2 - ADC2
//#define               3   // PF.3 - 사용가능

…
```

③ **초기화 매크로 정의** 부분의 **I/O 포트 초기화 매크로 정의** 부분에 위에서 추가로 정의한 LED 핀(PB6)을 출력핀으로 설정하고, HIGH 출력으로 초기화한다. LED 핀의 출력이 HIGH이면, 도터보드의 회로에 의해 LED가 OFF 상태로 된다.

```
//------------------------------------------------------------------------------------------
// I/O 포트 초기화 매크로 정의
//------------------------------------------------------------------------------------------
#define PORT_INIT() \
    do { \
        SFIOR |= BM(PUD); \
        DDRB  = BM(MOSI) | BM(SCK) | BM(CSN) | BM(LED); \
        PORTB = BM(MOSI) | BM(SCK) | BM(CSN) | BM(LED); \
        DDRD  = BM(UART1_RTS); \
        PORTD = BM(UART1_RTS) | BM(UART1_CTS); \
        DDRG  = BM(RESET_N) | BM(VREG_EN); \
        PORTG = BM(RESET_N); \
    } while (0)
```

④ 사용자가 장착하는 응용보드에 대한 초기화나 명령 매크로가 필요한 경우 **사용자 추가 매크로 - 외부장치 접속 관련** 부분에 추가한다.

본 예제에서는 프로그램 구현이 편리하도록 조명제어 명령 매크로를 정의하여 사용한다.

```
//------------------------------------------------------------------------------------------
// LCD, 센서, 외부인터럽트 사용 등, 외부장치 관련 초기화 및 명령 매크로
//------------------------------------------------------------------------------------------
#define LIGHT_ON()      PORTB &= 0xBF
#define LIGHT_OFF()     PORTB |= 0x40
```

[Step.3] hal.h 수정

hal.h는 **ATmega128(A,L)**의 내부 리소스 사용에 관한 초기화 매크로나 간단한 명령 매크로를 정의한다.
현재 기본적으로 정의되어 있는 매크로는 다음과 같다. **(hal.h** 참조)

- SPI_INIT() 매크로
- SPI를 통한 **CC2420** 제어 매크로
- 1 usec 딜레이 함수 선언 (halWait)
- Global interrupt 제어 매크로 (sei, cli)
- UART0, UART1 초기화 및 사용과 관련된 매크로
- ADC 초기화 및 사용과 관련된 매크로

SPI는 **CC2420** 제어 전용으로 사용되므로 사용자가 SPI를 추가적으로 사용할 경우에는 세심한 주의를 요한다.
위에서 SPI 관련 항목을 제외하고 4가지 항목이 정의되어 있으므로 사용자는 내용을 충분히 파악해서 프로그램 개발에 활용하도록 한다.
위에서 언급되지 않은 항목을 사용하려면 **hal.h**에 관련된 매크로를 추가해준다. 예를 들어 다음과 같은 내부 리소스를 사용하려면 사용자가 초기화 및 사용과 관련된 매크로를 추가해 주어야 한다.

- 타이머/카운터
- 외부인터럽트
- 내부EEPROM

① 프로젝트매니저 창에서 **External Dependancies**를 열고 **hal.h**를 더블클릭하여 편집창에 띄운다.

② 기존에 등록되어 있지 않은 **ATmega128(A,L)** 내부 리소스를 사용하려면 관련된 초기화 및 명령 매크로를 **사용자 추가 매크로** 부분에 추가한다.

※ 본 예제의 프로그램에서는 내부 리소스 사용에 관한 사용자 추가 매크로가 필요하지 않음

[Step.4] control.c 작성

① **control.c**를 더블클릭하여 편집창에 띄우고 고정노드에 대한 소스를 작성한다. (소스에 표시된 (1) ~ (9) 순)

② 작성이 완료되면 **Build** 메뉴에서 **Rebuild All**을 클릭하여 컴파일 및 hex파일을 생성한다.
(※ 오류가 발생하면 디버깅한다.)

◆ **control.c**의 내용 (청색 글씨의 내용만 필요에 따라 수정) ◆

```
/*****************************************************************************************************
*           - PAN ID: 0x2176 for example                                                             *
*           - Short address: 0x0000 ~ 0xFFFF changeable                                              *
*****************************************************************************************************/

#include "include\include.h"
#include <avr/eeprom.h>

//-----------------------------------------------------------------------------------------------------
// Basic RF transmission and reception structures
//-----------------------------------------------------------------------------------------------------
BASIC_RF_RX_INFO rfRxInfo;
BASIC_RF_TX_INFO rfTxInfo;
BYTE pTxBuffer[BASIC_RF_MAX_PAYLOAD_SIZE];
```

```
BYTE pRxBuffer[BASIC_RF_MAX_PAYLOAD_SIZE];

//-----------------------------------------------------------------------------------------------
// (7) 사용자가 필요에 따라 만든 외부변수
//-----------------------------------------------------------------------------------------------
//
// 사용자 외부변수 추가…
//

//-----------------------------------------------------------------------------------------------
// (8) 사용자가 필요에 따라 만든 함수는 여기에 정의하거나 별도의 .c 파일에 기술한다.
//-----------------------------------------------------------------------------------------------
//
// 사용자 함수 추가…
//
void UART1_PUTSTRING(char *str_add) // 문자 스트링 출력
{
        UINT8 count_str=0,lenth=0;
        lenth=strlen(str_add);

        for (count_str=0;count_str<lenth;count_str++)
        {
                UART1_WAIT_AND_SEND(str_add[count_str]);
        }
}

//-----------------------------------------------------------------------------------------------
// (9) 마스터 혹은 슬래이브로부터 무선데이타 수신후 처리 함수(인터럽트 서비스 루틴)
//-----------------------------------------------------------------------------------------------
BASIC_RF_RX_INFO* basicRfReceivePacket(BASIC_RF_RX_INFO *pRRI)
{
        unsigned char Command, tmp;
        unsigned int Fromaddr;

        // 패킷을 보낸 노드의 주소
```

```
        Fromaddr = pRRI->srcAddr;

        // 명령 코드
        Command = pRRI->pPayload[0];

        // 명령 분석
        switch(Command)
        {
//-------------------------------------------------------------------------------------------------
// (9-1) 마스터로부터 슬래이브가 무선데이타 수신한 경우: 처리후 결과를 다시 마스터로 보고
//-------------------------------------------------------------------------------------------------
                case 0x01: // [1] Read RSSI
                {
                        pTxBuffer[0] = 0x11; // 슬래이브가 마스터로 RSSI 보고
                        pTxBuffer[1] = (pRRI->rssi);

                        rfTxInfo.destAddr = Fromaddr;
                        basicRfSendPacket(&rfTxInfo);

                        break; // switch문 탈출
                } // case 0x01
                case 0x02: // [2] Read Light ADC
                {
                        pTxBuffer[0] = 0x12; // 슬래이브가 마스터로 조도센서 ADC값 보고

                        ADC_SAMPLE_SINGLE();
                        ADC_GET_SAMPLE_8(tmp);

                        pTxBuffer[1] = tmp;

                        rfTxInfo.destAddr = Fromaddr;
                        basicRfSendPacket(&rfTxInfo);

                        break; // switch문 탈출
                } // case 0x02
```

```
                case 0x03: // [3] Set Light ON/OFF
                {
                        if (pRRI->pPayload[1] == 1)
                                LIGHT_ON(); // Light ON
                        else if (pRRI->pPayload[1] == 0)
                                LIGHT_OFF(); // Light OFF

                        pTxBuffer[0] = 0x13; // 슬래이브가 마스터로 조명 ON/OFF 결과 보고
                        pTxBuffer[1] = pRRI->pPayload[1];

                        rfTxInfo.destAddr = Fromaddr;
                        basicRfSendPacket(&rfTxInfo);

                        break; // switch문 탈출
                } // case 0x03
//-------------------------------------------------------------------------------------------------
// (9-2) 슬래이브로부터 마스터가 무선데이타 수신한 경우: 보고내용을 하이퍼터미널로 출력
//-------------------------------------------------------------------------------------------------
                case 0x11: // 슬래이브가 마스터로 RSSI 보고
                {
                        tmp = pRRI->pPayload[1] - 45;

                        UART1_LINEFEED();
                        UART1_LINEFEED();
                        UART1_LINEFEED();
                        UART1_LINEFEED();
                        UART1_LINEFEED();

                        UART1_PUTSTRING("> RSSI : ");

                        if((tmp & 0x80) == 0)
                        {
                                UART1_WAIT_AND_SEND(tmp / 100 + 0x30);
                                UART1_WAIT_AND_SEND(tmp / 10 - (tmp / 100) * 10 + 0x30);
                                UART1_WAIT_AND_SEND((tmp - (tmp / 100) * 100) - ((tmp - (tmp / 100) * 100) / 10) * 10 + 0x30);
```

```
		}
		else
		{
			tmp = ~(tmp - 1);
			UART1_WAIT_AND_SEND('-');
			UART1_WAIT_AND_SEND(tmp / 100 + 0x30);
			UART1_WAIT_AND_SEND(tmp / 10 - (tmp / 100) * 10 + 0x30);
			UART1_WAIT_AND_SEND((tmp - (tmp / 100) * 100) - ((tmp - (tmp / 100) * 100) / 10) * 10 + 0x30);
		}

		UART1_PUTSTRING("[dBm]");
		UART1_LINEFEED();

		break; // switch문 탈출
	} // case 0x11
	case 0x12: // 슬래이브가 마스터로 조도센서 ADC값 보고
	{
		tmp = pRRI->pPayload[1];

		UART1_LINEFEED();
		UART1_LINEFEED();
		UART1_LINEFEED();
		UART1_LINEFEED();
		UART1_LINEFEED();

		UART1_PUTSTRING("> Light ADC : ");

		UART1_WAIT_AND_SEND(tmp / 100 + 0x30);
		UART1_WAIT_AND_SEND(tmp / 10 - (tmp / 100) * 10 + 0x30);
		UART1_WAIT_AND_SEND((tmp - (tmp / 100) * 100) - ((tmp - (tmp / 100) * 100) / 10) * 10 + 0x30);

		UART1_LINEFEED();

		break; // switch문 탈출
	} // case 0x12
```

```
                case 0x13: // 슬래이브가 마스터로 조명 ON/OFF 결과 보고
                {
                        UART1_LINEFEED();
                        UART1_LINEFEED();
                        UART1_LINEFEED();
                        UART1_LINEFEED();
                        UART1_LINEFEED();

                        UART1_PUTSTRING("> Light Status : ");
                        if (pRRI->pPayload[1] == 1) UART1_PUTSTRING("ON");
                        else if(pRRI->pPayload[1] == 0) UART1_PUTSTRING("OFF");
                        else UART1_PUTSTRING("WRIGHT ERROR!!");

                        UART1_LINEFEED();

                        break; // switch문 탈출
                } // case 0x13
                default: // 수신 패킷 오류
                {
                        break; // switch(Command)문 탈출
                }

        } // switch(Command)

        return pRRI; // 수신대기 상태로 복귀

} // basicRfReceivePacket

//-------------------------------------------------------------------------------------------------
// main 함수
//-------------------------------------------------------------------------------------------------
int main(void)
{
//-------------------------------------------------------------------------------------------------
// (1) I/O포트, 변수, ATmega128 내부장치 및 주변장치 초기화
```

```
//-------------------------------------------------------------------------------------------------
        unsigned int srcaddr;
        unsigned char i, x, eeprom[1];

        PORT_INIT();
        SPI_INIT();
        INIT_UART1(UART_BAUDRATE_115K2,UART_OPT_8_BITS_PER_CHAR); // UART1 초기화(115.2 Kpbs)
        ENABLE_UART1();
        ADC_INIT();
        ADC_SET_CHANNEL(ADC_INPUT_2); // ADC2 사용
        ADC_ENABLE();

//-------------------------------------------------------------------------------------------------
// (2) source address 설정
//-------------------------------------------------------------------------------------------------
        *eeprom = 0; // EEPROM 0번지를 source address 저장용으로 사용한다
        srcaddr = (UINT16)eeprom_read_byte(eeprom); // EEPROM 0번지에서 읽어낸다

//-------------------------------------------------------------------------------------------------
// (3) 채널 및 Pan ID 설정
//-------------------------------------------------------------------------------------------------
        basicRfInit(&rfRxInfo, 26, 0x2176, srcaddr); // 26번 채널 설정, Pan ID = 0x2176(예를 들어..)

//-------------------------------------------------------------------------------------------------
// (4) 전송데이타 크기 설정
//-------------------------------------------------------------------------------------------------
        rfTxInfo.length = 10; // 10 바이트 전송(예를 들어..)

//-------------------------------------------------------------------------------------------------
// ACK 수신 여부 설정
//-------------------------------------------------------------------------------------------------
        rfTxInfo.ackRequest = TRUE;

        rfTxInfo.pPayload = pTxBuffer;
        rfRxInfo.pPayload = pRxBuffer;
```

```
//-------------------------------------------------------------------------------------------------
// (5) 전송데이타 초기화
//-------------------------------------------------------------------------------------------------
        for (unsigned char n = 0; n < 10; n++)
        {
                pTxBuffer[n] = 0; // 송신 버퍼 클리어
        }

        // Turn on RX mode
        basicRfReceiveOn();

//-------------------------------------------------------------------------------------------------
// (6) 메인 루틴
//-------------------------------------------------------------------------------------------------
        while(1) // 무한 반복
        {
//-------------------------------------------------------------------------------------------------
// (6-1) 마스터가 하이퍼터미널(PC)과 입출력
//-------------------------------------------------------------------------------------------------

                // UART1 sends menu to Terminal
                UART1_LINEFEED();
                UART1_LINEFEED();
                UART1_LINEFEED();
                UART1_PUTSTRING("**********<< ZBM Monitor V1.0 >>**********");
                UART1_LINEFEED();
                UART1_PUTSTRING("* [1] Read RSSI      [2] Read Light ADC       [3] Set Light ON/OFF *");
                UART1_LINEFEED();
                UART1_PUTSTRING("* [4] Read Address [5] Write Address                                  *");
                UART1_LINEFEED();
                UART1_PUTSTRING("*************************************************************");
                UART1_LINEFEED();
                UART1_PUTSTRING("> Select Command Number : ");
```

```
                // UART1 receives Command Number from Terminal
                UART1_WAIT_AND_RECEIVE(x);

//-------------------------------------------------------------------------------------------------
// (6-2) 마스터에서 슬레이브로 보낼 데이타, destination address 설정 및 무선 송신
//-------------------------------------------------------------------------------------------------

                // Command Number 분석
                switch(x)
                {
                        case 0x31: // [1] Read RSSI
                        {
                                UART1_WAIT_AND_SEND(x);
                                UART1_PUTSTRING(" Read RSSI");
                                UART1_LINEFEED();
                                pTxBuffer[0] = 0x01; // 슬래이브로 보낼 명령: Read RSSI

                                // UART1 sends sub-menu to Terminal for destination address
                                i = 2;
                                UART1_PUTSTRING("> Type Destination Address[Decimal 00~99] : ");
                                while(i)
                                {
                                        UART1_WAIT_AND_RECEIVE(x);
                                        if((x >= 0x30) && (x <= 0x39))
                                        {
                                                UART1_WAIT_AND_SEND(x);
                                                if(i == 2) rfTxInfo.destAddr = (x - 0x30) * 10;
                                                else if(i == 1) rfTxInfo.destAddr += x - 0x30;
                                                i--;
                                        }
                                } // while(i)

                                // Master sends packet to Slave
                                basicRfSendPacket(&rfTxInfo);
                                halWait(50000);
```

```
            break; // switch문 탈출
} // case 0x31
case 0x32: // [2] Read Light ADC
{
            UART1_WAIT_AND_SEND(x);
            UART1_PUTSTRING(" Read Light ADC");
            UART1_LINEFEED();
            pTxBuffer[0] = 0x02; // 슬래이브로 보낼 명령: Read Light ADC

            // UART1 sends sub-menu to Terminal for destination address
            i = 2;
            UART1_PUTSTRING("> Type Destination Address[Decimal 00~99] : ");
            while(i)
            {
                        UART1_WAIT_AND_RECEIVE(x);
                        if((x >= 0x30) && (x <= 0x39))
                        {
                                    UART1_WAIT_AND_SEND(x);
                                    if(i == 2) rfTxInfo.destAddr = (x - 0x30) * 10;
                                    else if(i == 1) rfTxInfo.destAddr += x - 0x30;
                                    i--;
                        }
            } // while(i)

            // Master sends packet to Slave
            basicRfSendPacket(&rfTxInfo);
            halWait(50000);

            break; // switch문 탈출
} // case 0x32
case 0x33: // [3] Set Light ON/OFF
{
            UART1_WAIT_AND_SEND(x);
            UART1_PUTSTRING(" Set Light ON/OFF");
```

```
UART1_LINEFEED();
pTxBuffer[0] = 0x03; // 슬래이브로 보낼 명령: Set Light ON/OF

// UART1 sends sub-menu to Terminal for destination address
i = 2;
UART1_PUTSTRING("> Type Destination Address[Decimal 00~99] : ");
while(i)
{
        UART1_WAIT_AND_RECEIVE(x);
        if((x >= 0x30) && (x <= 0x39))
        {
                UART1_WAIT_AND_SEND(x);
                if(i == 2) rfTxInfo.destAddr = (x - 0x30) * 10;
                else if(i == 1) rfTxInfo.destAddr += x - 0x30;
                i--;
        }
} // while(i)

// UART1 sends sub-menu to Terminal for light on or off
UART1_LINEFEED();
i = 1;
UART1_PUTSTRING("> Select Number[0:off, 1:on] : ");
while(i)
{
        UART1_WAIT_AND_RECEIVE(x);
        if((x == 0x30) || (x == 0x31))
        {
                pTxBuffer[1] = x - 0x30;
                i--;
        }
} // while(i)

// Master sends packet to Slave
basicRfSendPacket(&rfTxInfo);
halWait(50000);
```

```
            break; // switch문 탈출
} // case 0x33
case 0x34: // [4] Read Address
{
            UART1_WAIT_AND_SEND(x);
            UART1_PUTSTRING(" Read Address");
            UART1_LINEFEED();
            UART1_LINEFEED();
            UART1_LINEFEED();
            UART1_LINEFEED();
            UART1_LINEFEED();

            UART1_PUTSTRING("> Address : ");

            // EEPROM 0번지에서 source address를 읽어내어 출력한다
            *eeprom = 0;
            UART1_WAIT_AND_SEND(eeprom_read_byte(eeprom)/10 + 0x30);
            UART1_WAIT_AND_SEND(eeprom_read_byte(eeprom)%10 + 0x30);
            UART1_LINEFEED();

            break; // switch문 탈출
} // case 0x34
case 0x35: // [5] Write Address
{
            UART1_WAIT_AND_SEND(x);
            UART1_PUTSTRING(" Write Address");
            UART1_LINEFEED();

            // UART1 sends sub-menu to Terminal
            i = 2;
            UART1_PUTSTRING("> Type Address[Decimal 00~99] : ");
            while(i)
            {
                        UART1_WAIT_AND_RECEIVE(x);
```

```
                        if((x >= 0x30) && (x <= 0x39))
                        {
                                UART1_WAIT_AND_SEND(x);
                                if(i == 2) rfSettings.myAddr = (x - 0x30) * 10;
                                else if(i == 1) rfSettings.myAddr += x - 0x30;
                                i--;
                        }
                } // while(i)

                // EEPROM 0번지에 source address를 저장한다
                *eeprom = 0;
                eeprom_write_byte (eeprom, (UINT8)rfSettings.myAddr);

                // Revise address on CC2420 tranceiver
                DISABLE_GLOBAL_INT();
                FASTSPI_WRITE_RAM_LE(&rfSettings.myAddr, CC2420RAM_SHORTADDR, 2, i);
                ENABLE_GLOBAL_INT();

                UART1_LINEFEED();
                UART1_PUTSTRING("> Address Writing Completed!");
                UART1_LINEFEED();

                break; // switch문 탈출
        } // case 0x35
        default: // Incorrect Number
        {
                UART1_LINEFEED();
                UART1_PUTSTRING("Incorrect! Select Number Again..");
                UART1_LINEFEED();
                break; // switch문 탈출
        }

        } // switch(x)
    } // while(1)
} // main()
```

7. 펌웨어 다운로드

[Step.1] 하드웨어 준비

① **ZBM v1.2** 모듈에 **AVR**용 JTAG장치 커넥터와 전원을 장착한다. (※ 전원 OFF상태, JTAG장치와 PC간 케이블은 장착하지 않은 상태)

※ JTAG장치 주의사항

- JTAG장치에 전원 선택 스위치가 있는 경우, 타겟(**ZBM v1.2** 모듈)으로부터 전원을 공급받도록 스위치를 설정한다. 만약 PC로부터 전원을 공급받도록 설정하게 되면 JTAG장치의 종류에 따라서 타겟이나 JTAG장치 자체에 손상을 야기할 수 있다.
- **ZBM v1.2** 모듈은 JTAG장치에 +3.3V의 전원을 공급하므로, JTAG장치는 3.3V의 전원으로 구동이 가능해야 한다.

② **ZBM v1.2** 모듈의 전원을 ON한 후, JTAG장치와 PC간 케이블을 장착한다. 이것은 JTAG장치가 정상적으로 셋업된 후에 PC에서 정상적으로 JTAG장치를 인식할 수 있기 때문이다.

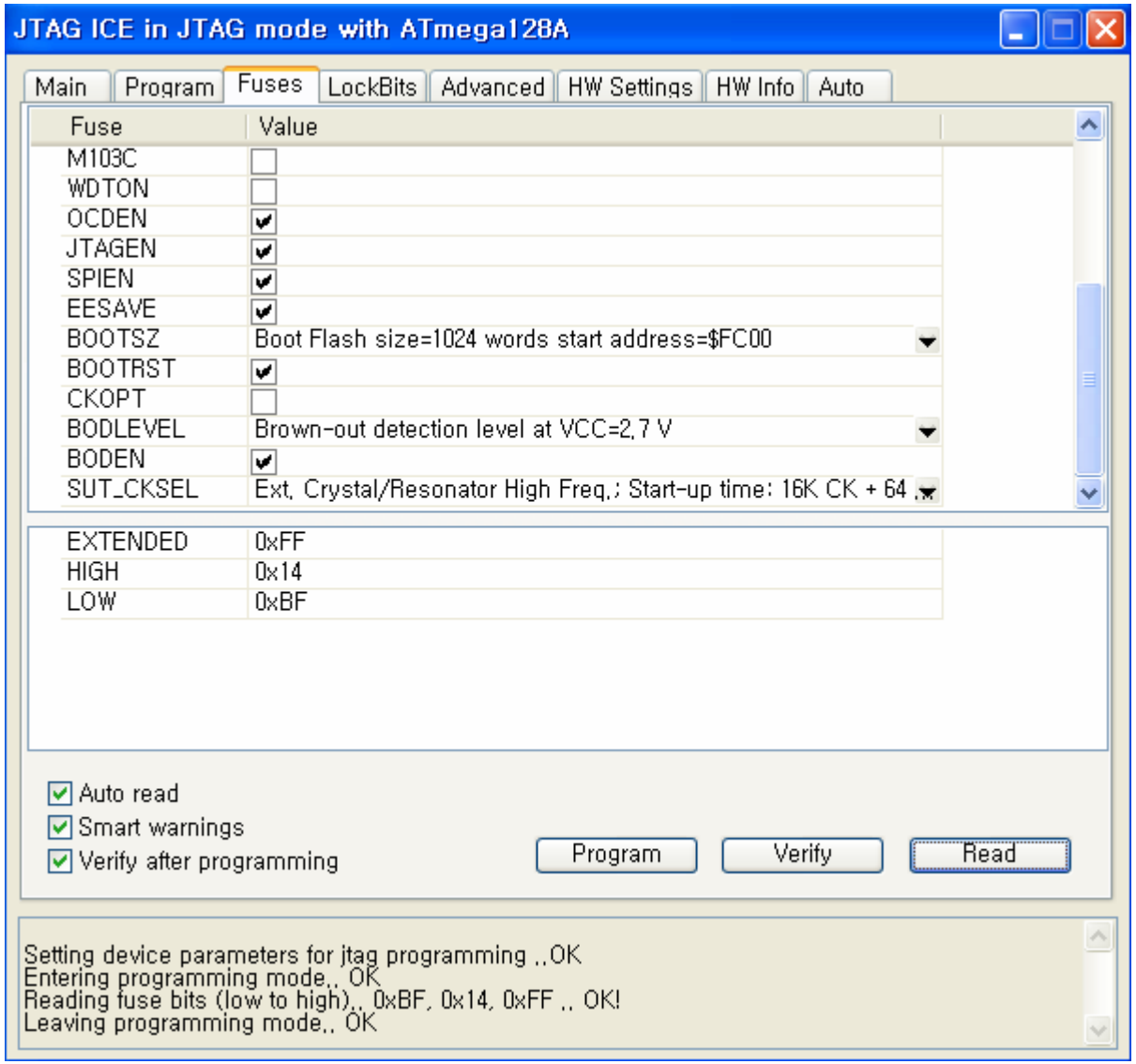
[Step.2] AVR Studio에서 다운로드

① **AVR Studio**에서 **Tools** 메뉴 아래에 **Program AVR -> Connect...**를 클릭하여 **Select AVR Programmer** 창을 띄운다.

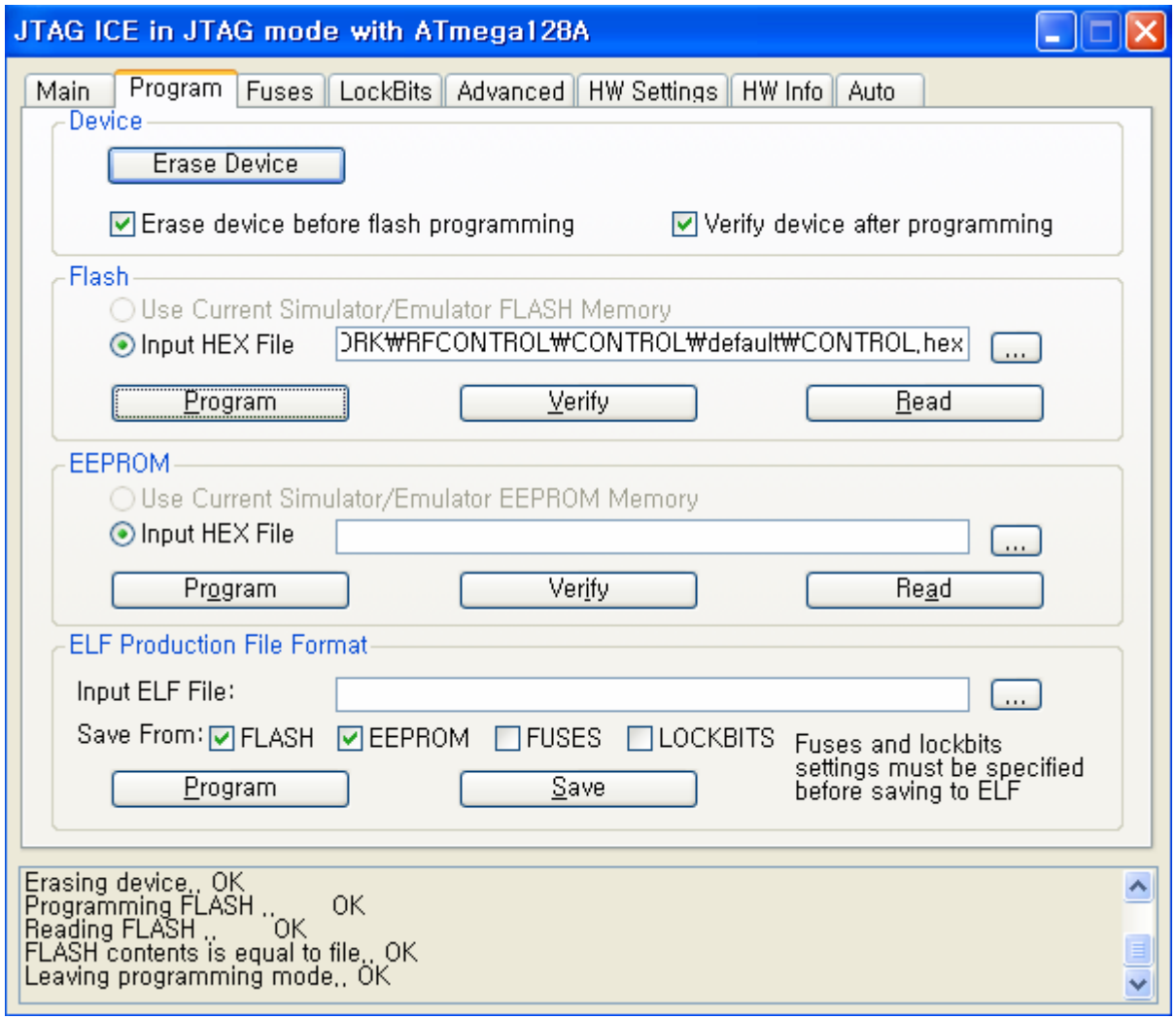
② **Select AVR Programmer** 창에서, **Platform: JTAG ICE**, **Port: COM3**(예, COM Port를 모르면 **Auto** 선택)을 선택하고 **Connect**를 클릭하여 **JTAG ICE** 창을 띄운다.

③ **JTAG ICE** 창에서 **Main** 탭을 선택하고, **Device**를 **ATmega128**로 선택한다.

④ **JTAG ICE** 창에서 **Fuses** 탭을 선택하고, 다음과 같이 퓨즈 비트를 체크해제 및 체크하고 **Program**을 클릭하여 퓨즈 비트를 설정한다.



⑤ **JTAG ICE** 창에서 **Program** 탭을 선택하고, 다운로드할 hex 파일의 위치를 지정하고 **Program**을 클릭하여 펌웨어를 다운로드한다.



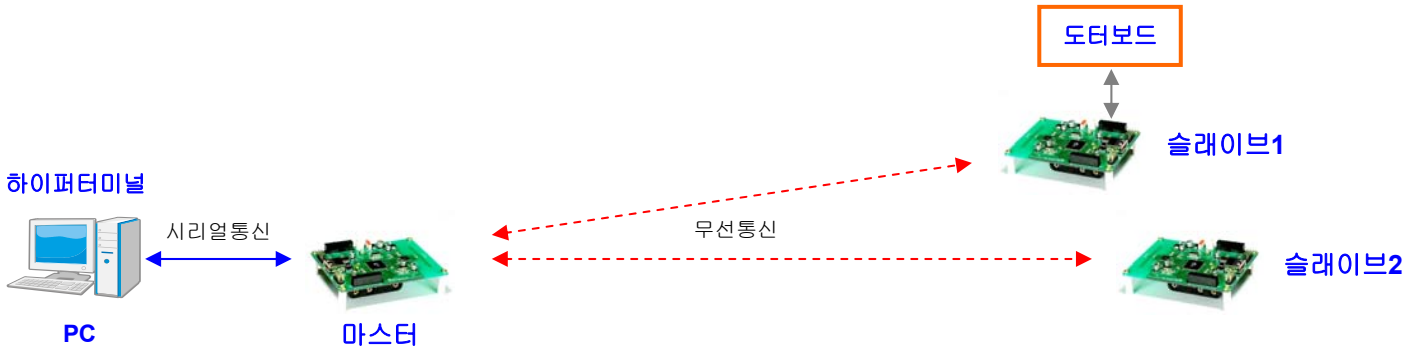
8. 무선 제어 예제 시스템 종합 실습

앞에서 제작한 펌웨어 프로그램을 사용하여 무선 제어 예제 시스템을 구축하고, PC에서 **하이퍼터미널**을 통하여 RSSI 측정, 조명(LED) 제어, 조도 측정 과정을 실습해보도록 한다.

기본적인 구성은 두 개의 **ZBM v1.2**을 사용하여 **1:1 무선 제어 시스템**을 구성하는 것으로서, 한 개를 마스터로하고 나머지를 슬래이브로 사용하는 것이다. **1:N 무선 제어 시스템**은 한 개를 마스터로하고 여러 개의 슬래이브를 사용하여 구성할 수 있다.

마스터와 슬래이브에는 주소(source address)를 할당해주어야 한다. 제품이 출고될 때 모든 **ZBM v1.2**는 주소가 00 번지로 설정된다. **1:1 무선 제어**는 동일한 주소라도 사용이 가능하지만, **1:N 무선 제어**는 마스터와 각각의 슬래이브에 서로 다른 주소를 설정해주어야 한다. 본 예제 프로그램에서는 00 ~ 99 까지 100개의 주소를 사용할 수 있다.(예제 프로그램을 변경하면 **IEEE 802.15.4** 규격인 16비트 용량의 주소도 사용 가능)

7-1. 시스템 구성



[예제 프로젝트의 시스템구성도]

7-2. 실습 절차

[Step.1] 시스템 설치

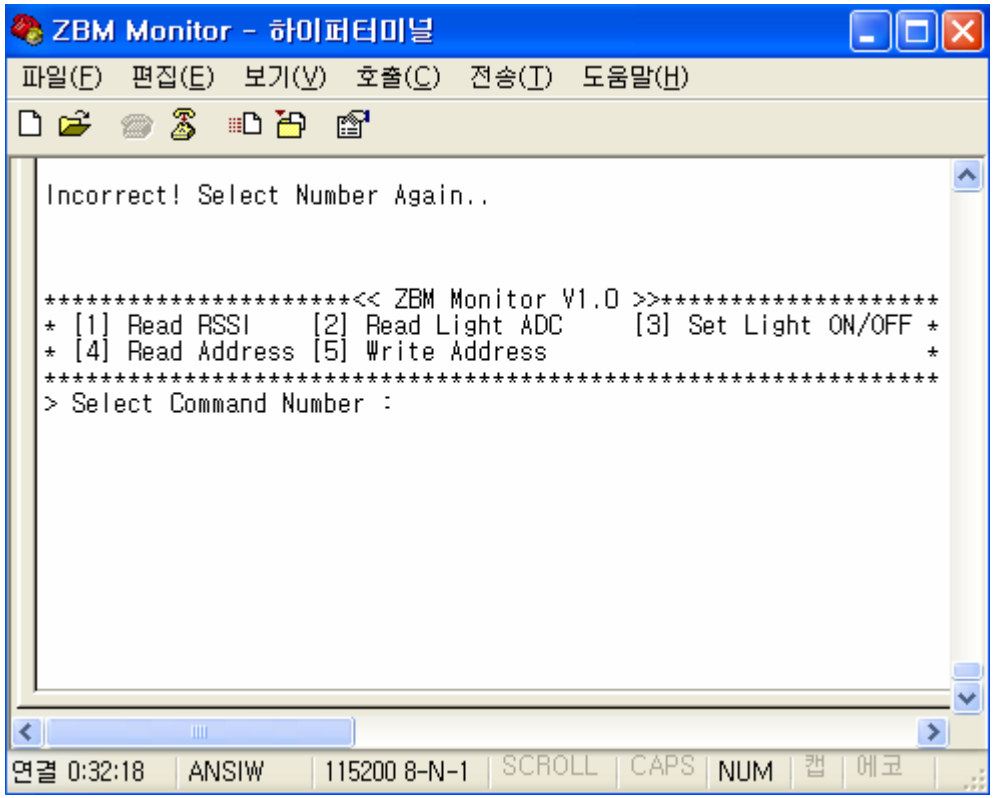
① PC를 기동하고 **하이퍼터미널**을 실행한다.(**시작->모든프로그램->보조프로그램->통신**/)
② 마스터(**ZBM v1.2**)를 설치하고 PC와 USB케이블로 접속한 후 전원을 켠다.

⇨ PC에서 자동으로 2개의 가상 시리얼포트를 할당한다.
⇨ **시작->제어판->시스템->하드웨어->장치관리자**에서 추가된 COM포트를 확인한다.
(※ **하이퍼터미널**에서도 추가된 COM포트가 확인됨)

③ **하이퍼터미널**에서 **파일->새연결**을 클릭하여 다음과 같이 설정한다.

이름 : **ZBM Monitor** (예)
연결에 사용할 모뎀 : **COM3** (예,장치관리자에서 확인된 포트)
비트/초 : **115200**
데이터 비트 : **8**
패리티 : **없음**
정지 비트 : **1**
흐름 제어 : **없음**

④ **하이퍼터미널**의 **호출** 버튼을 클릭하여 연결 상태로 한 후 엔터 키를 눌러 **ZBM v1.2**의 메뉴 화면이 **하이퍼터미널**에 출력되는 것을 확인한다.



[Step.2] 주소 설정

① 키보드에서 숫자 **5**를 입력하여 '**Write Address**' 명령을 실행한다.
② '**Type Address:**' 에 두자리의 숫자를 입력하여 마스터에 대한 주소를 설정한다.(예, 00)
③ 키보드에서 숫자 **4**를 입력하여 '**Read Address**' 명령을 실행하고 앞에서 설정한 주소를 확인한다.
④ 슬래이브에 대해서도 앞의 과정을 참조하여 각각 주소를 설정한다.(예, 01, 02)

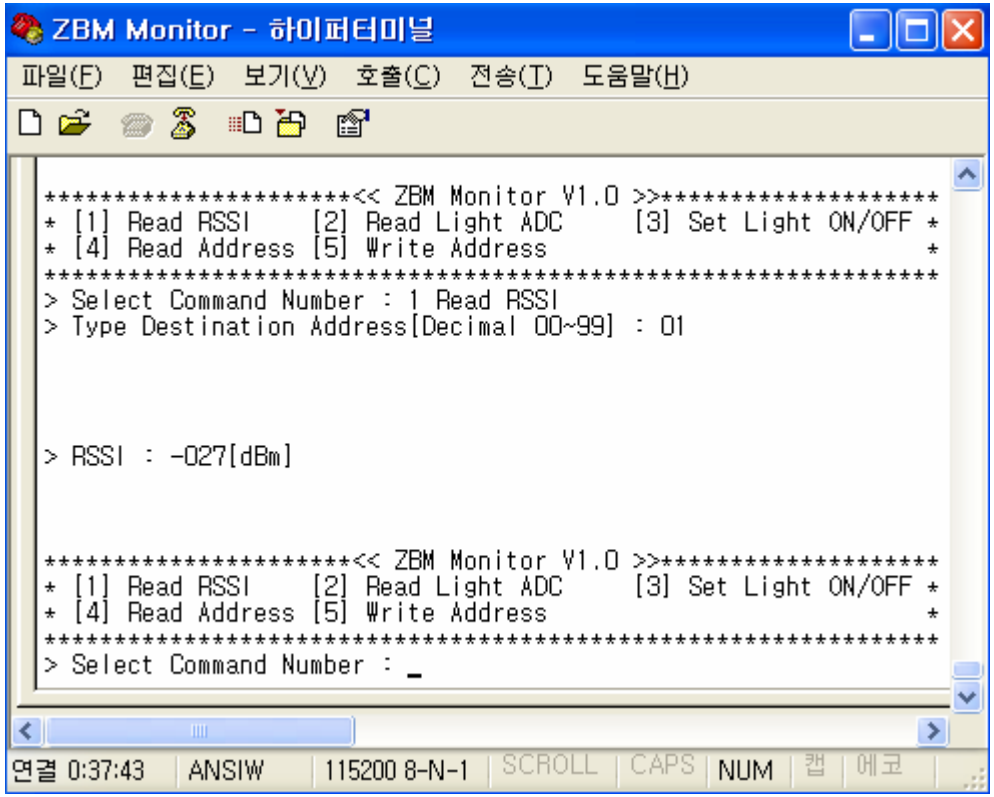
[Step.3] 슬래이브 설치

① 슬래이브1에는 도터보드를 장착한다. (※ 도터보드가 없으면 **RSSI 값 읽기** 실습만 가능)
② **예제 프로젝트의 시스템구성도**에 따라, 슬래이브1과 스래이브2를 적당한 위치에 각각 설치하고 전원을 켠다.

[Step.4] RSSI 값 읽기

① 하이퍼터미널에서 숫자 **1**를 입력하여 '**Read RSSI**' 명령을 실행한다.

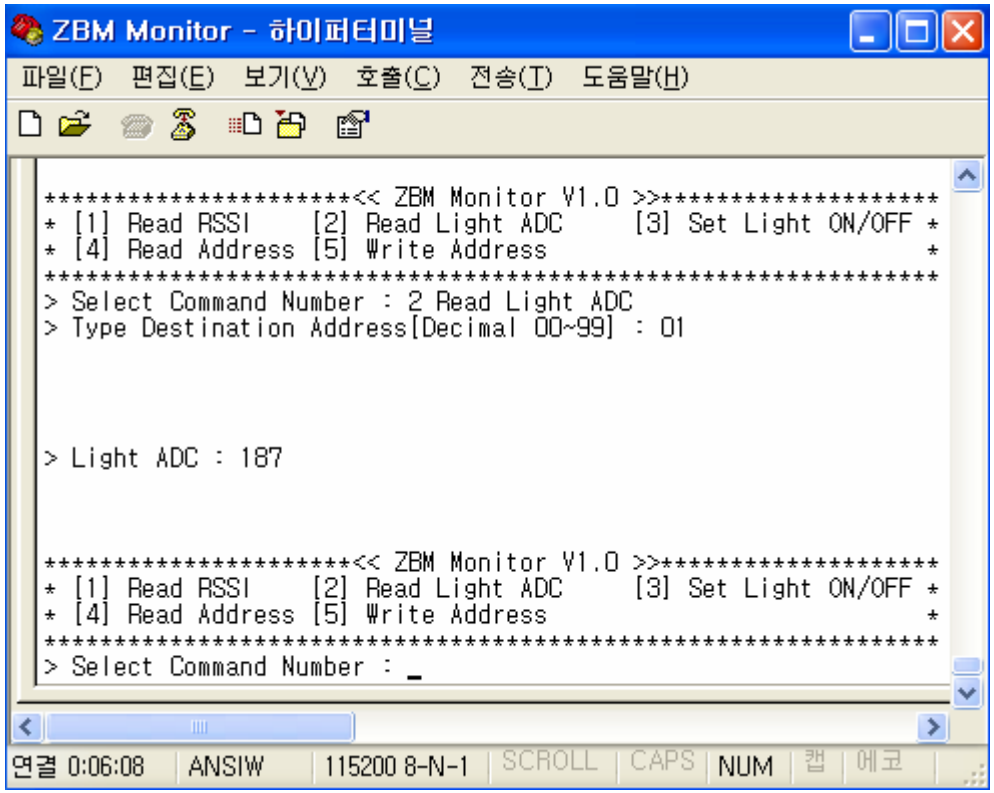
② '**Type Destination Address:**' 에 측정하고자 하는 슬레이브의 주소(두자리 숫자)를 입력한다.(예, 01)



[Step.5] 조도센서 ADC 값 읽기

① 하이퍼터미널에서 숫자 **2**를 입력하여 '**Read Light ADC**' 명령을 실행한다.

② '**Type Destination Address:**' 에 측정하고자 하는 슬레이브의 주소(두자리 숫자)를 입력한다.(예, 01)

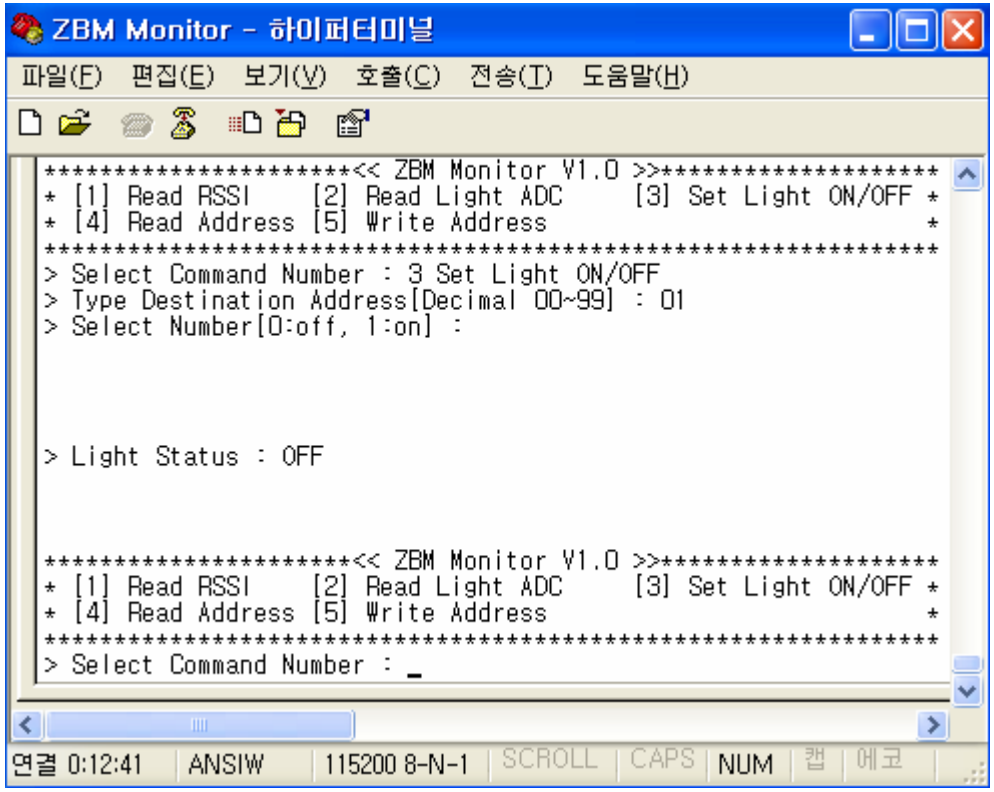


[Step.6] 조명 ON/OFF 제어

① 하이퍼터미널에서 숫자 **3**를 입력하여 '**Read Light ADC**' 명령을 실행한다.

② '**Type Destination Address:**' 에 측정하고자 하는 슬래이브의 주소(두자리 숫자)를 입력한다.(예, 01)

③ '**Select Number:**' 에 숫자 **1**을 입력하여 조명(LED)이 켜지도록 해보고, 명령을 반복 수행하여 꺼지도록 해본다.



[Step.7] 정리

① 실습이 완료되면 마스터와 슬레이브의 전원을 끄고 시스템을 정리한다.

감사합니다

(제품문의, 기술지원)
OL마이크로웨이브 Co.
●홈페이지 http://www.olmicrowaves.com
●E-mail webmaster@olmicrowaves.com , imaman@hitel.net