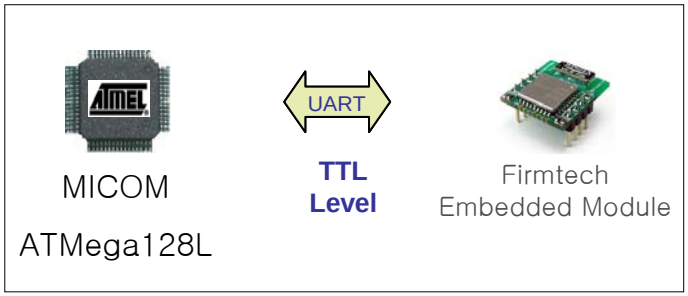
FBDx5xMC

< FBD155MC_gcc V0.1 >

1. 개요

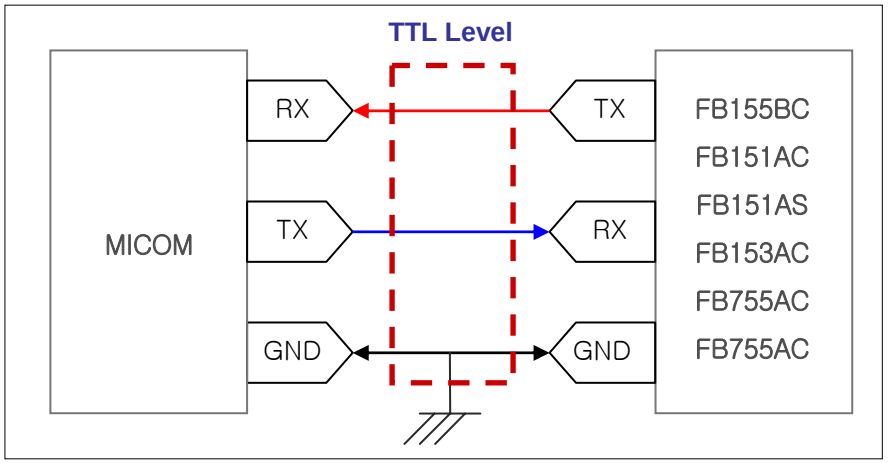
- 본 문서는 MICOM으로 ㈜펌테크 제품인 Bluetooth Embedded Module을 제어하기 위한 하드웨어 및 소프트웨어 사용 방법과 요령을 소개하는 문서로서 사용자들이 구현하고자 하는 Target Application 구성을 보다 빠르고 쉽게 구성할 수 있도록 도움을 주는데 목적을 둡니다.

<1> 하드웨어 구성의 이해



참고 : UART : Universal Asynchronous Receiver Transmitter
MICOM의 경우 현재 시중에 널리 사용되는 ATMega128L 사용

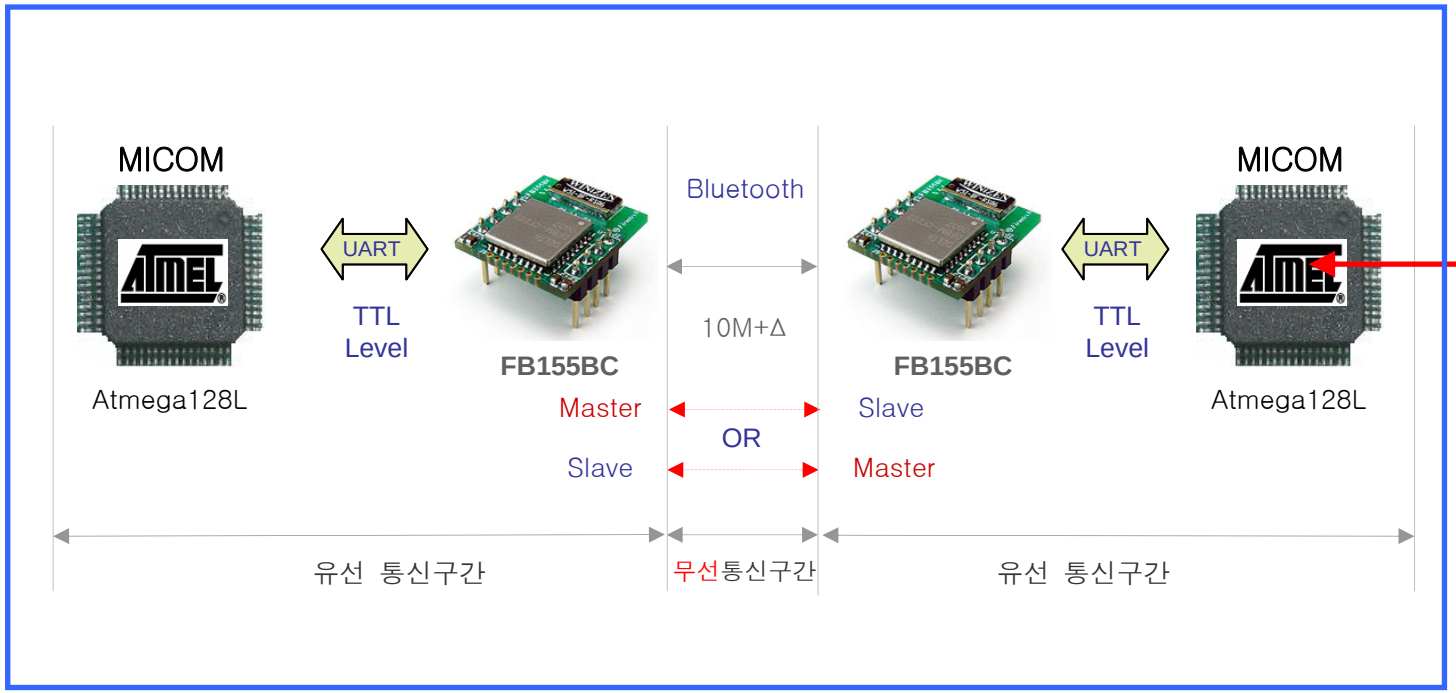
<2> MICOM 과 Firmtech Bluetooth Embedded Module 과의 인터페이스 방법



2. 전체 구성도 (실제 구성도)

< 1th FBDx5xMC >

< 2th FBDx5xMC >



소프트웨어의 구성

사용 언어 : ANSI C

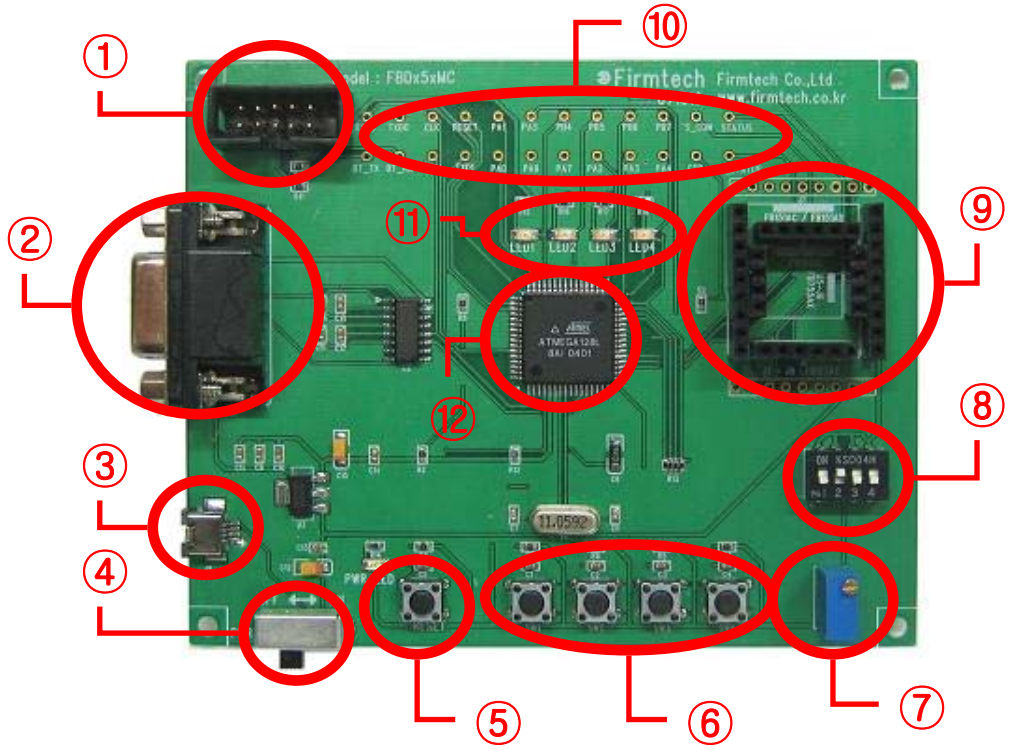
컴파일러 : GCC

사용자가 개발한 최종 소프트웨어를 Atmega128L 내부 플래시 메모리에 다운로딩 하여 사용함

3. FBD155MC_gcc 소개

- FBD155MC_gcc는 ㈜펌테크의 FBDx5xMC 보드에 ㈜펌테크의 임베디드 모듈인 FB155BC를 장착하여, FB155BC에서 제공되는 AT-Command를 이용하여 Bluetooth 연결 / 데이터 송-수신을 MICOM으로 제어하는 프로그램 소스 명
- 사용하는 MICOM으로는 ATMEL사의 ATMega128L 사용
- 사용하는 컴파일러는 WINAVR 사용
- 실행 파일을 다운로드하기 위해 사용하는 프로그램은 PonyProg 사용
- 시리얼 데이터 송/수신 및 디버깅 포트로 DB-9 사용
- 디버깅을 위한 테스트 포인트 사용
- 전원 입력은 PC의 USB 포트 사용
- 데이터의 입력으로 RS-232 포트 사용
- 데이터의 출력으로 RS-232 포트 사용
- FB155BC 2개와 FBDx5xMC 보드 2개 사용
- FBD155MC_gcc는 FB155BC의 Slave Bluetooth address를 직접 소스에 입력하여 "소스 컴파일->프로그램 다운로드->Bluetooth 연결->시리얼 데이터 송/수신" 진행

4. FBDx5xMC 제품 외형



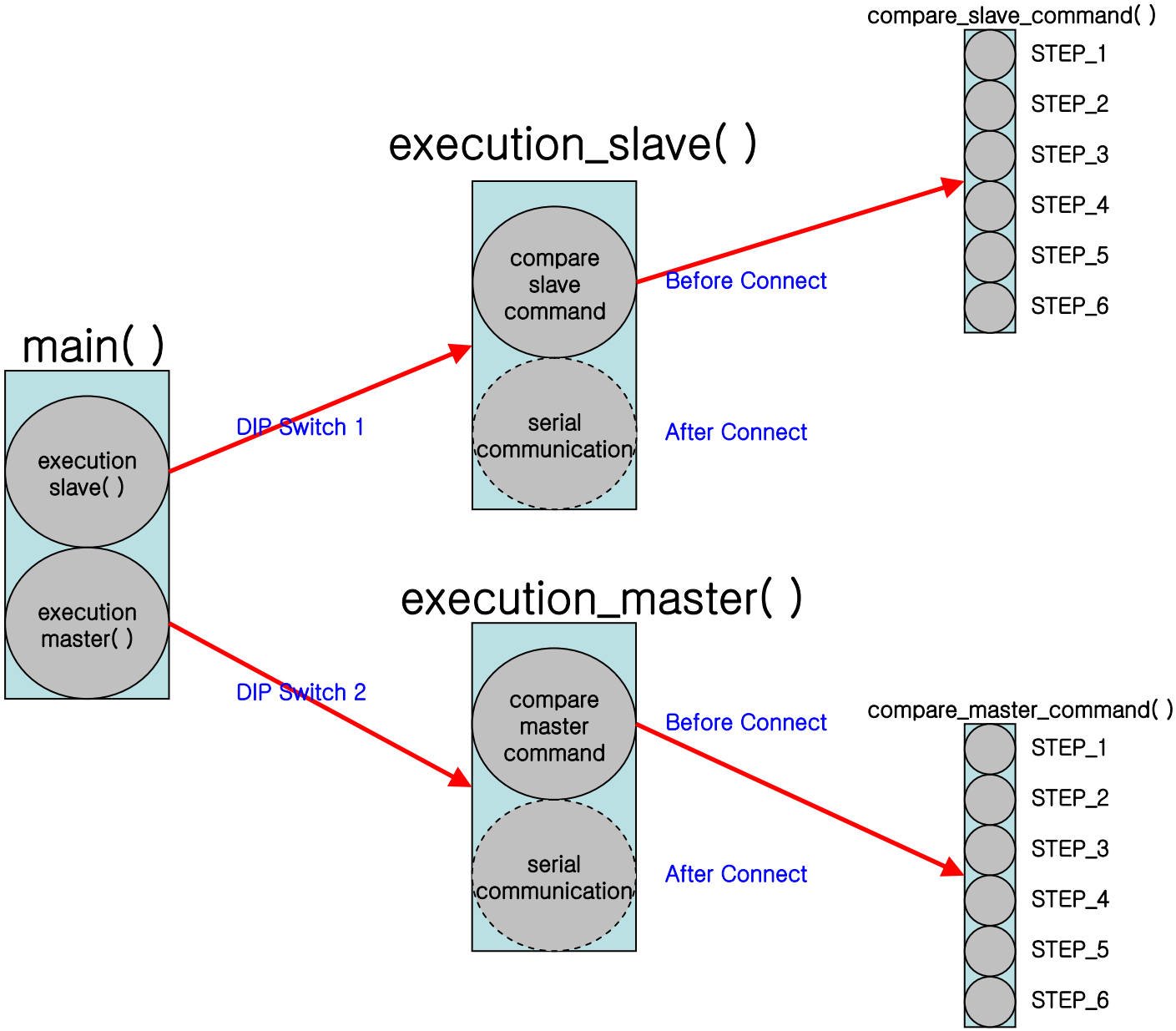
< FBDx5xMC 제품 외형 >

1. AVR Loader 연결 커넥터
2. RS-232 시리얼 연결 커넥터
3. USB 전원 연결 커넥터
4. 전원 ON/OFF 스위치
5. 리셋 스위치 (ATMega128L 리셋 용)
6. 스위치 (데이터 입력 용)
7. 가변 저항 (데이터 입력 용)
8. 딥 스위치 (데이터 입력 용)
9. 임베디드 제품 연결 커넥터
10. 테스트 포인트 (디버거 용)
11. LED (데이터 출력 용)
12. ATMega128L (프로그램용 MICOM)

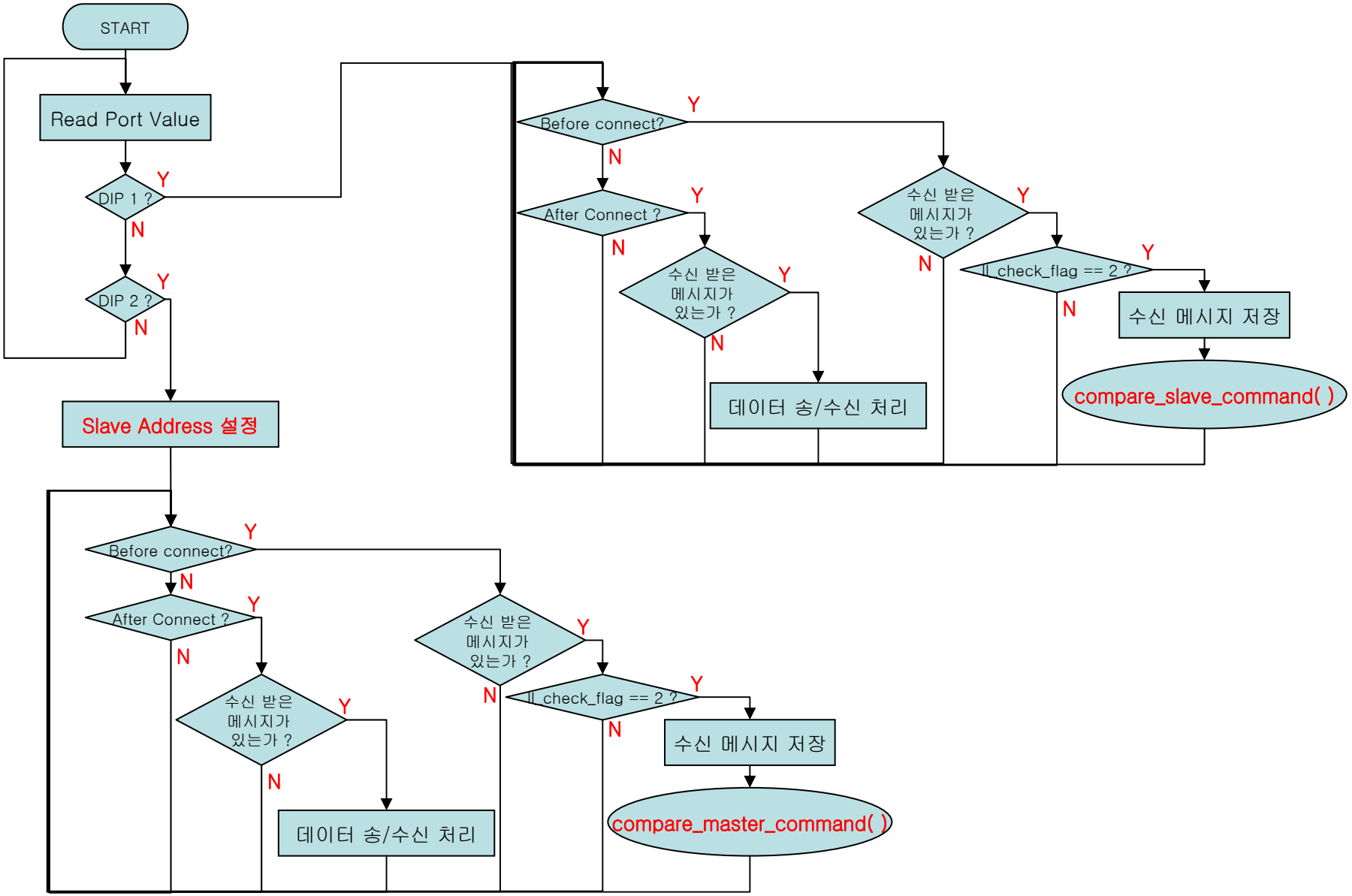
5. FBDx5xMC (FBD155MC_gcc) 제품의 기본 구성 품

Model NO.	Pictures	Q'ty	Description
FBDx5xMC		1	임베디드 블루투스 컨트롤 보드
FBA180SC		1	RS-232 시리얼 연결 케이블
FBA002PO		1	USB 전원 공급 케이블
CD		1	Source, Datasheet, Manual, Utility CD
AVR Loader		1	프로그램 다운로드 케이블
FB155BC		1	임베디드 블루투스 모듈

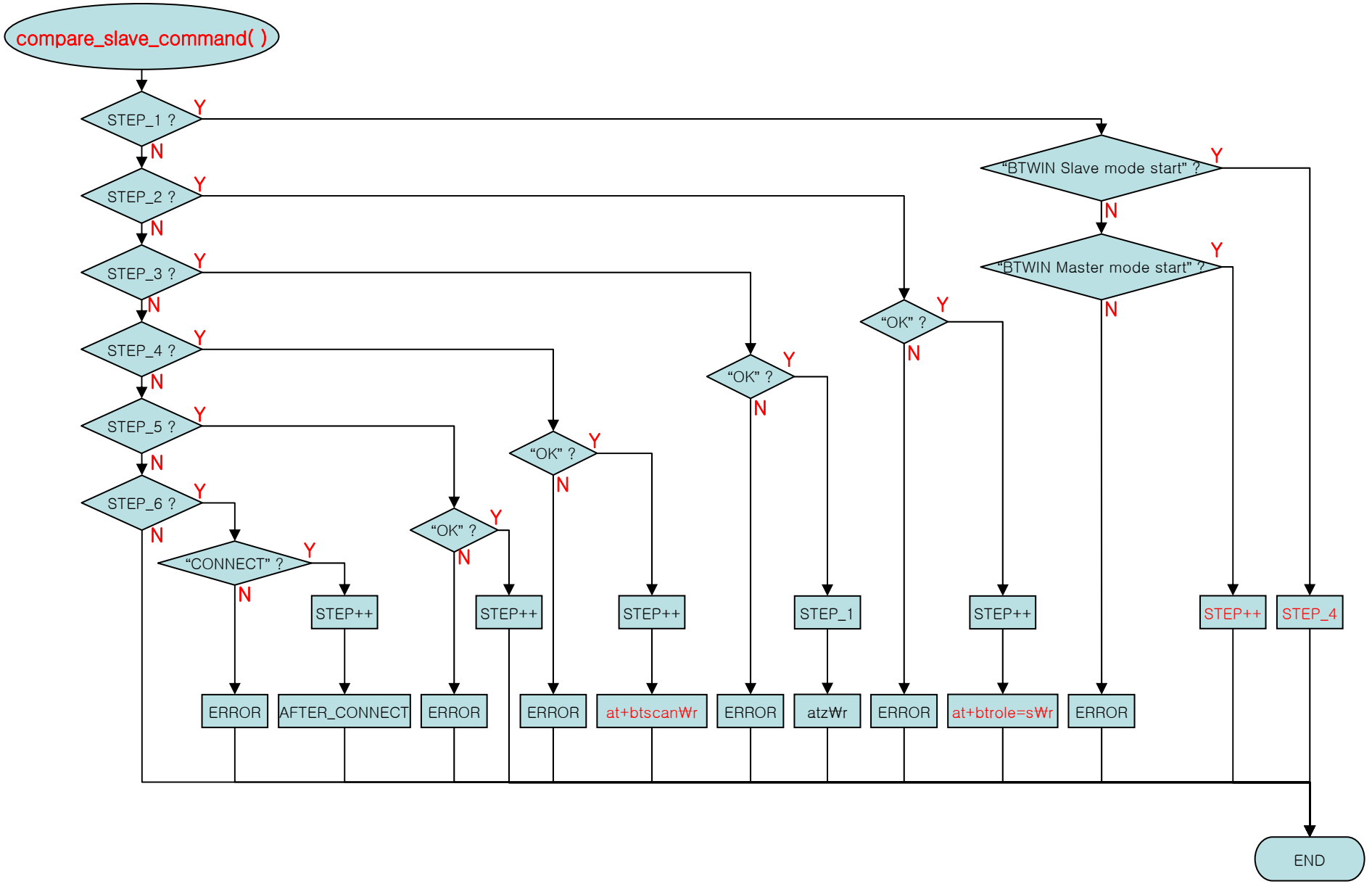
6. FBD155MC_gcc Block Diagram



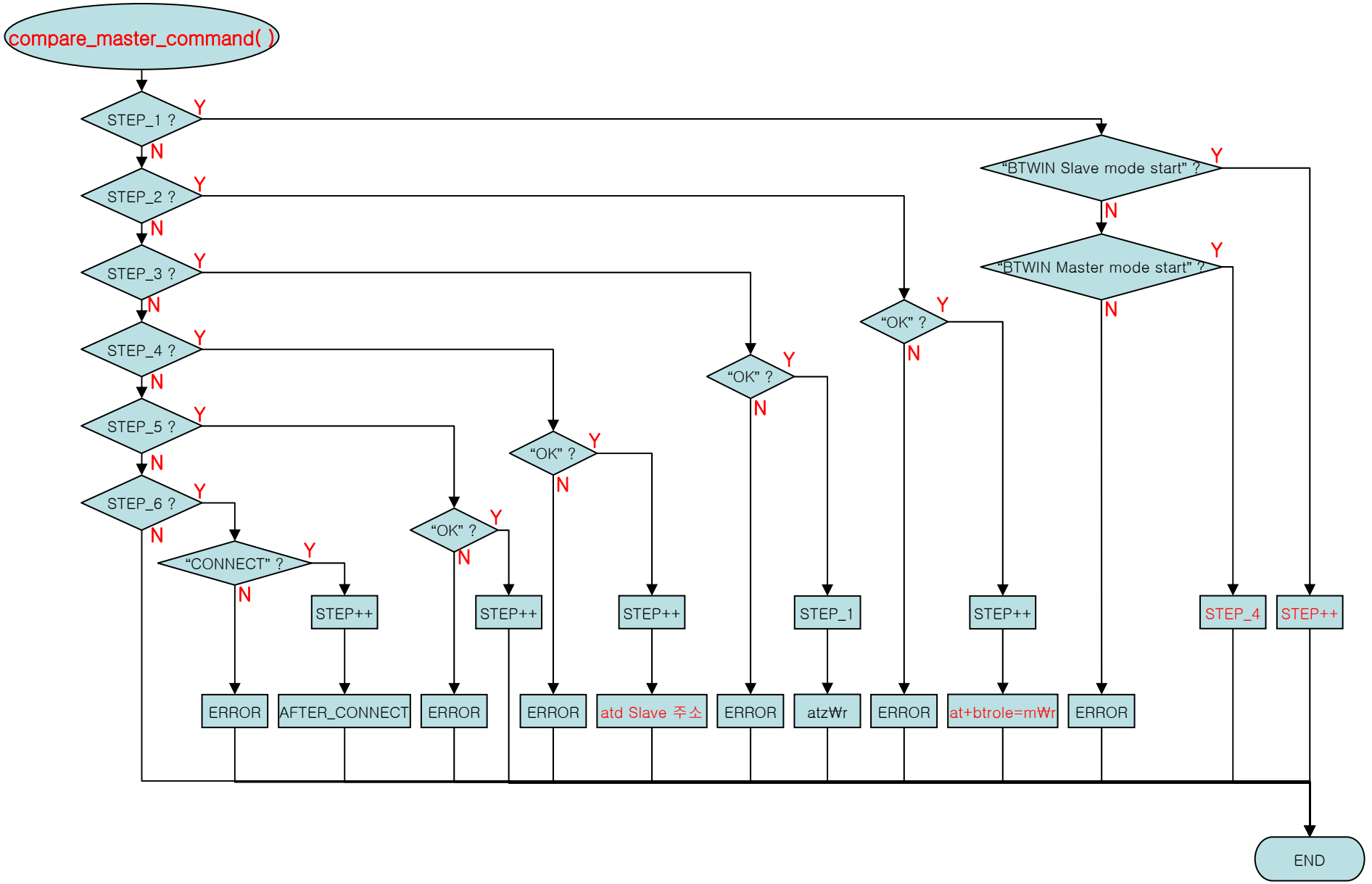
7. FBD155MC_gcc Flow Chart - 1



8. FBD151MC_gcc Flow Chart – 2 < compare_slave_compare( ) >



9. FBD155MC_gcc Flow Chart – 3 < compare_master_command( ) >



10. FBD155MC_gcc compare_slave_command( ) STEP Description

step_slave_bt	Description 1th	Description
STEP_1	Bluetooth Device로부터 최초 수신한 메시지 비교 루틴	• Bluetooth Device로부터 “BTWIN Slave mode start”메시지가 수신되었는지 검사 – Bluetooth Device가 Slave로 시작된 경우, 다음 스텝 STEP_4 설정 • Bluetooth Device로부터 “BTWIN Master mode start”메시지가 수신되었는지 검사 – Bluetooth Device가 Master로 시작된 경우, 다음 스텝 STEP_2 설정 • 비교 데이터가 틀린 경우 error루틴 처리
STEP_2	Bluetooth Device Role 변경 루틴 1번째	• Bluetooth Device로부터 “OK”메시지가 수신되었는지 검사 – Bluetooth Device가 “Master”로 시작된 후 “OK” 메시지를 수신 받으면, “at+btrole=s₩r” Command를 이용하여 Bluetooth Device를 Slave로 설정 – 다음 스텝 STEP_3 설정 • 비교 데이터가 틀린 경우 error루틴 처리
STEP_3	Bluetooth Device Role 변경 루틴 2번째	• Bluetooth Device로부터 “OK”메시지가 수신되었는지 검사 – “at+btrole=s₩r” Command를 이용하여 Bluetooth Device가 Slave로 설정된 경우 “OK” 메시지 수신 – 변경된 Role을 적용하기 위해 “atz₩r” Command를 이용하여 Bluetooth Device Soft Reset 진행 – 다음 스텝 STEP_1 설정 • 비교 데이터가 틀린 경우 error루틴 처리
STEP_4	Bluetooth Device 검색/연결 대기 설정 루틴 1번째	• Bluetooth Device로부터 “OK”메시지가 수신되었는지 검사 – Bluetooth Device가 “Slave”로 시작된 후 “OK”메시지를 수신 받으면, “at+btscan₩r” Command를 이용하여 Bluetooth Device를 검색/연결 대기 상태로 설정 – 다음 스텝 STEP_5 설정 • 비교 데이터가 틀린 경우 error루틴 처리
STEP_5	Bluetooth Device 검색/연결 대기 설정 루틴 2번째	• Bluetooth Device로부터 “OK”메시지가 수신되었는지 검사 – “at+btscan₩r” Command를 이용하여 Bluetooth Device를 검색/연결 대기 상태로 설정한 경우 “OK”메시지 수신 – 다음 스텝 STEP_6 설정 • 비교 데이터가 틀린 경우 error루틴 처리
STEP_6	Bluetooth Device 연결 상태 체크 루틴	• Bluetooth Device로부터 “CONNECT”메시지가 수신되었는지 검사 – Slave가 검색/연결 대기를 하고 있는 경우, Master에서 검색 및 연결 가능 – Slave가 Master와 연결된 경우 “CONNECT”메시지 수신 – Slave와 Master가 연결된 경우, 데이터 송/수신 루틴 진행(루틴상 다음 스텝 STEP_7 설정) • 비교 데이터가 틀린 경우 error루틴 처리

11. FBD155MC_gcc compare_master_command( ) STEP Description

step_slave_bt	Description 1th	Description 2th
STEP_1	Bluetooth Device로부터 최초 수신한 메시지 비교 루틴	• Bluetooth Device로부터 “BTWIN Slave mode start”메시지가 수신되었는지 검사 – Bluetooth Device가 Slave로 시작된 경우, 다음 스텝 STEP_2 설정 • Bluetooth Device로부터 “BTWIN Master mode start”메시지가 수신되었는지 검사 – Bluetooth Device가 Master로 시작된 경우, 다음 스텝 STEP_4 설정 • 비교 데이터가 틀린 경우 error루틴 처리
STEP_2	Bluetooth Device Role 변경 루틴 1번째	• Bluetooth Device로부터 “OK”메시지가 수신되었는지 검사 – Bluetooth Device가 “Slave”로 시작된 후 “OK” 메시지를 수신 받으면, “at+btrole=m₩r” Command를 이용하여 Bluetooth Device를 Master로 설정 – 다음 스텝 STEP_3 설정 • 비교 데이터가 틀린 경우 error루틴 처리
STEP_3	Bluetooth Device Role 변경 루틴 2번째	• Bluetooth Device로부터 “OK”메시지가 수신되었는지 검사 – “at+btrole=m₩r” Command를 이용하여 Bluetooth Device가 Master로 설정된 경우 “OK” 메시지 수신 – 변경된 Role을 적용하기 위해 “atz₩r” Command를 이용하여 Bluetooth Device Soft Reset 진행 – 다음 스텝 STEP_1 설정 • 비교 데이터가 틀린 경우 error루틴 처리
STEP_4	Bluetooth Device 연결 요청 루틴 1번째	• Bluetooth Device로부터 “OK”메시지가 수신되었는지 검사 – Bluetooth Device가 “Master”로 시작된 후 “OK”메시지를 수신 받으면, “atd” Command와 저장되어 있는 Slave의 어드레스를 이용하여 연결 요청 진행 – 다음 스텝 STEP_5 설정 • 비교 데이터가 틀린 경우 error루틴 처리
STEP_5	Bluetooth Device 연결 요청 루틴 2번째	• Bluetooth Device로부터 “OK”메시지가 수신되었는지 검사 – “atd” Command와 Slave의 어드레스를 이용하여 연결 요청이 진행되는 경우 “OK”메시지 수신 – 다음 스텝 STEP_6 설정 • 비교 데이터가 틀린 경우 error루틴 처리
STEP_6	Bluetooth Device 연결 상태 체크 루틴	• Bluetooth Device로부터 “CONNECT”메시지가 수신되었는지 검사 – Master가 Slave와 연결을 하기 위해서 Slave는 검색/연결 대기 상태이어야 함 – Master가 Slave와 연결된 경우 “CONNECT”메시지 수신 – Master와 Slave가 연결된 경우, 데이터 송/수신 루틴 진행(루틴상 다음 스텝 STEP_7 설정) • 비교 데이터가 틀린 경우 error루틴 처리

12. FBD155MC_gcc main( ) Source

```
int main(void)
{
    unsigned int i;

    init_port();
    Init_UART1(BAUD_9600);
    Init_UART0(BAUD_9600);
    init_timer0();
    sei();

    read_avr_port = 0;
    lf_check_flag = LF_CHECK_NON;
    status_target = TARGET_NOT_DETECT;
    wait_1ms(10);
    DispStr1("\r\n\r\nFBD155MC_gcc START");
    while(1)
    {
        read_avr_port = read_port_value(READ_DIP);
        if(read_avr_port == DIP_1)
            execution_slave();
        else if(read_avr_port == DIP_2)
            execution_master();
    }
} ? end main ?
```

< 시스템 초기화 >

1. 포트 초기화
2. 통신 속도 초기화
3. 타이머 초기화

< Flag 초기화 >

1. 포트 저장 변수 초기화
2. LF Flag 초기화
3. 타겟 검색 Flag 초기화

< Master/Slave 설정 및 수행 >

1. 딥 스위치 Read
2. 1 번 딥 스위치가 ON 이면 슬레이브 루틴 진행
3. 2 번 딥 스위치가 ON 이면 마스터 루틴 진행

13. FBD155MC_gcc execution_slave( ) Source

```
void execution_slave(void)
{
    unsigned int i;

    status_sequence = BEFORE_CONNECT;
    step_slave_bt = STEP_1;
    init_uart0_read_data();
    while(1)
    {
        if(status_sequence == BEFORE_CONNECT)
        {
            uart0_length = Check_Rx_Buf0();
            if(uart0_length>0)
            {
                if(lf_check_flag == 2)
                {
                    uart0_length = Check_Rx_Buf0();
                    for(i = 0;i<uart0_length;i++)
                        uart0_read_data[i] = GetChar0();
                    compare_slave_command();
                }
            }
        }
        else if(status_sequence == AFTER_CONNECT)
        {
            lf_check_flag = LF_CHECK_NON;
            uart0_length = Check_Rx_Buf0();
            if(uart0_length)
            {
                for(i = 0;i < uart0_length;i++)
                    PutChar1(GetChar0());
            }
            uart1_length = Check_Rx_Buf1();
            if(uart1_length)
            {
                for(i = 0;i < uart1_length;i++)
                    PutChar0(GetChar1());
            }
        }
    } ? end while 1 ?
} ? end execution_slave ?
```

< 슬레이브 루틴 진행 >

1. Bluetooth 연결 전 상태 설정
2. STEP_1 설정
3. 비교 데이터 저장 버퍼 초기화

< Bluetooth 연결 전 진행 루틴 >

1. Bluetooth로부터 데이터 수신 시 동작
2. Bluetooth로부터 수신 받은 데이터에 0x0a가 2개인 경우 동작
3. Bluetooth로부터 수신 받은 데이터를 비교 데이터 저장 버퍼에 저장
4. Bluetooth 슬레이브 메시지 비교 루틴 진행

< Bluetooth 연결 후 진행 루틴 >

1. Bluetooth로부터 수신 받은 데이터가 있는 경우 호스트(PC)로 전달
2. 호스트(PC)로부터 수신 받은 데이터가 있는 경우 Bluetooth로 전달

14. FBD155MC_gcc execution_master( ) Source

```
void execution_master(void)
{
    unsigned int i;

    //
    memcpy(target_data,"00066e150c68",12);
    //
    status_sequence = BEFORE_CONNECT;
    step_master_bt = STEP_1;
    init_uart0_read_data();
    while(1)
    {
        if(status_sequence == BEFORE_CONNECT)
        {
            uart0_length = Check_Rx_Buf0();
            if(uart0_length>0)
            {
                if(lf_check_flag == 2)
                {
                    uart0_length = Check_Rx_Buf0();
                    for(i = 0;i<uart0_length;i++)
                        uart0_read_data[i] = GetChar0();
                    compare_master_command();
                }
            }
        }
        else if(status_sequence == AFTER_CONNECT)
        {
            lf_check_flag = LF_CHECK_NON;
            uart0_length = Check_Rx_Buf0();
            if(uart0_length)
            {
                for(i = 0;i < uart0_length;i++)
                    PutChar1(GetChar0());
            }
            uart1_length = Check_Rx_Buf1();
            if(uart1_length)
            {
                for(i = 0;i < uart1_length;i++)
                    PutChar0(GetChar1());
            }
        }
    } ? end while 1 ?
} ? end execution_master ?
```

< 슬레이브 어드레스 저장 >

1. 연결할 Bluetooth Slave Device의 어드레스를 입력

< 마스터 루틴 진행 >

1. Bluetooth 연결 전 상태 설정
2. STEP_1 설정
3. 비교 데이터 저장 버퍼 초기화

< Bluetooth 연결 전 진행 루틴 >

1. Bluetooth로부터 데이터 수신 시 동작
2. Bluetooth로부터 수신 받은 데이터에 0x0a가 2개인 경우 동작
3. Bluetooth로부터 수신 받은 데이터를 비교 데이터 저장 버퍼에 저장
4. Bluetooth 마스터 메시지 비교 루틴 진행

< Bluetooth 연결 후 진행 루틴 >

1. Bluetooth로부터 수신 받은 데이터가 있는 경우 호스트(PC)로 전달
2. 호스트(PC)로부터 수신 받은 데이터가 있는 경우 Bluetooth로 전달

15. FBD155MC_gcc compare_slave_command( ) Source – 1th

```
void compare_slave_command(void)
{
    switch(step_slave_bt)
    {
        case STEP_1:
            lf_check_flag = LF_CHECK_NON;
            if(!memcmp("\r\nBTWIN Slave mode start\r\n",&uart0_read_data[uart0_length-26],26))
            {
                step_slave_bt = STEP_4;
                DispStr1("[STEP_1 : MESSAGE RECEIVE OK]");
            }
            else if(!memcmp("\r\nBTWIN Master mode start\r\n",&uart0_read_data[uart0_length-27],27))
            {
                step_slave_bt++;
                DispStr1("[STEP_1 : MESSAGE RECEIVE OK]");
            }
            else
                execution_error();
        break;
        case STEP_2:
            lf_check_flag = LF_CHECK_NON;
            if(!memcmp("\r\nOK\r\n",&uart0_read_data[uart0_length-6],6))
            {
                step_slave_bt++;
                DispStr1("[STEP_2 : MESSAGE RECEIVE OK]");
                DispStr1("[       CHANGE ROLE]");
                DispStr0("at+btrole=s\r");
            }
            else
                execution_error();
        break;
        case STEP_3:
            lf_check_flag = LF_CHECK_NON;
            if(!memcmp("\r\nOK\r\n",&uart0_read_data[uart0_length-6],6))
            {
                step_slave_bt = STEP_1;
                DispStr1("[STEP_3 : MESSAGE RECEIVE OK]");
                DispStr1("[       RESET BLUETOOTH]");
                DispStr0("atz\r");
            }
            else
                execution_error();
        break;
```

< STEP_1 >

1. Bluetooth로부터 “₩r₩nBTWIN Slave mode start₩r₩n” 메시지를 수신한 경우 STEP_4 진행
2. Bluetooth로부터 “₩r₩nBTWIN Mater mode start₩r₩n” 메시지를 수신한 경우 STEP_2 진행
3. 비교 데이터가 틀린 경우 execution_error()루틴 진행

16. FBD155MC_gcc compare_slave_command( ) Source – 2th

```
void compare_slave_command(void)
{
    switch(step_slave_bt)
    {
        case STEP_1:
            lf_check_flag = LF_CHECK_NON;
            if(!memcmp("\r\nBTWIN Slave mode start\r\n",&uart0_read_data[uart0_length-26],26))
            {
                step_slave_bt = STEP_4;
                DispStr1("[STEP_1 : MESSAGE RECEIVE OK]");
            }
            else if(!memcmp("\r\nBTWIN Master mode start\r\n",&uart0_read_data[uart0_length-27],27))
            {
                step_slave_bt++;
                DispStr1("[STEP_1 : MESSAGE RECEIVE OK]");
            }
            else
                execution_error();
        break;
        case STEP_2:
            lf_check_flag = LF_CHECK_NON;
            if(!memcmp("\r\nOK\r\n",&uart0_read_data[uart0_length-6],6))
            {
                step_slave_bt++;
                DispStr1("[STEP_2 : MESSAGE RECEIVE OK]");
                DispStr1("[       CHANGE ROLE]");
                DispStr0("at+btrole=s\r");
            }
            else
                execution_error();
        break;
        case STEP_3:
            lf_check_flag = LF_CHECK_NON;
            if(!memcmp("\r\nOK\r\n",&uart0_read_data[uart0_length-6],6))
            {
                step_slave_bt = STEP_1;
                DispStr1("[STEP_3 : MESSAGE RECEIVE OK]");
                DispStr1("[       RESET BLUETOOTH]");
                DispStr0("atz\r");
            }
            else
                execution_error();
        break;
```

< STEP_2 : Bluetooth가 마스터로 시작된 경우 >

1. Bluetooth로부터 “₩r₩nOK₩r₩n”메시지를 수신한 경우 STEP_3 진행
2. Bluetooth Device를 슬레이브로 변경
3. 비교 데이터가 틀린 경우 execution_error()루틴 진행

< STEP_3 : Bluetooth가 마스터로 시작된 경우 >

1. Bluetooth로부터 “₩r₩nOK₩r₩n”메시지를 수신한 경우 STEP_1 진행
2. Bluetooth Device Soft Reset 진행
3. 비교 데이터가 틀린 경우 execution_error()루틴 진행

17. FBD155MC_gcc compare_slave_command( ) Source – 3th

```
        case STEP_4:
            lf_check_flag = LF_CHECK_NON;
            if(!memcmp("\r\nOK\r\n",&uart0_read_data[uart0_length-6],6))
            {
                step_slave_bt++;
                DispStr1("[STEP_4 : MESSAGE RECEIVE OK]");
                DispStr1("[       EXECUTON SCAN]");
                wait_1ms(100);
                DispStr0("at+btscan\r");
            }
            else
                execution_error();
        break;
        case STEP_5:
            lf_check_flag = LF_CHECK_NON;
            if(!memcmp("\r\nOK\r\n",&uart0_read_data[uart0_length-6],6))
            {
                step_slave_bt++;
                DispStr1("[STEP_5 : MESSAGE RECEIVE OK]");
            }
            else
                execution_error();
        break;
        case STEP_6:
            lf_check_flag = LF_CHECK_NON;
            if(!memcmp("\r\nCONNECT",uart0_read_data,9))
            {
                step_master_bt++;
                status_sequence = AFTER_CONNECT;
                DispStr1("[STEP_6 : COMMUNICATION START]");
            }
            else
                execution_error();
        break;
    } ? end switch step_slave_bt ?
    init_uart0_read_data();
} ? end compare_slave_command ?
```

< STEP_4 : Bluetooth가 슬레이브로 시작된 경우 >

1. Bluetooth로부터 “₩r₩nOK₩r₩n”메시지를 수신한 경우 STEP_5 진행
2. Bluetooth Device를 검색 대기 상태로 동작
3. 비교 데이터가 틀린 경우 execution_error()루틴 진행

< STEP_5 : Bluetooth가 슬레이브로 시작된 경우 >

1. Bluetooth로부터 “₩r₩nOK₩r₩n”메시지를 수신한 경우 STEP_6 진행
2. 비교 데이터가 틀린 경우 execution_error()루틴 진행

< STEP_6 : Bluetooth가 슬레이브로 시작된 경우 >

1. Bluetooth로부터 “₩r₩nCONNECT”메시지를 수신한 경우 STEP_7 진행(STEP_7은 아무런 동작 없고, 비교 루틴을 나가는 동작 진행)
2. Bluetooth Device를 연결 상태로 설정
3. 비교 데이터가 틀린 경우 execution_error()루틴 진행

18. FBD155MC_gcc compare_master_command( ) Source – 1th

```
void compare_master_command(void)
{
    unsigned int i;

    switch(step_master_bt)
    {
        case STEP_1:
            lf_check_flag = LF_CHECK_NON;
            if(!memcmp("\r\nBTWIN Slave mode start\r\n",&uart0_read_data[uart0_length-26],26))
            {
                step_master_bt++;
                DispStr1("[STEP_1 : MESSAGE RECEIVE OK]");
            }
            else if(!memcmp("\r\nBTWIN Master mode start\r\n",&uart0_read_data[uart0_length-27],27))
            {
                step_master_bt = STEP_4;
                DispStr1("[STEP_1 : MESSAGE RECEIVE OK]");
            }
            else
                execution_error();
        break;
        case STEP_2:
            lf_check_flag = LF_CHECK_NON;
            if(!memcmp("\r\nOK\r\n",&uart0_read_data[uart0_length-6],6))
            {
                step_master_bt++;
                DispStr1("[STEP_2 : MESSAGE RECEIVE OK]");
                DispStr1("[       CHANGE ROLE]");
                DispStr0("at+btrole=m\r");
            }
            else
                execution_error();
        break;
        case STEP_3:
            lf_check_flag = LF_CHECK_NON;
            if(!memcmp("\r\nOK\r\n",&uart0_read_data[uart0_length-6],6))
            {
                step_master_bt = STEP_1;
                DispStr1("[STEP_3 : MESSAGE RECEIVE OK]");
                DispStr1("[       RESET BLUETOOTH]");
                DispStr0("atz\r");
            }
            else
                execution_error();
        break;
```

< STEP_1 >

1. Bluetooth로부터 “₩r₩nBTWIN Slave mode start₩r₩n” 메시지를 수신한 경우 STEP_2 진행
2. Bluetooth로부터 “₩r₩nBTWIN Mater mode start₩r₩n” 메시지를 수신한 경우 STEP_4 진행
3. 비교 데이터가 틀린 경우 execution_error()루틴 진행

19. FBD155MC_gcc compare_master_command( ) Source – 2th

```
void compare_master_command(void)
{
    unsigned int i;

    switch(step_master_bt)
    {
        case STEP_1:
            lf_check_flag = LF_CHECK_NON;
            if(!memcmp("\r\nBTWIN Slave mode start\r\n",&uart0_read_data[uart0_length-26],26))
            {
                step_master_bt++;
                DispStr1("[STEP_1 : MESSAGE RECEIVE OK]");
            }
            else if(!memcmp("\r\nBTWIN Master mode start\r\n",&uart0_read_data[uart0_length-27],27))
            {
                step_master_bt = STEP_4;
                DispStr1("[STEP_1 : MESSAGE RECEIVE OK]");
            }
            else
                execution_error();
        break;
        case STEP_2:
            lf_check_flag = LF_CHECK_NON;
            if(!memcmp("\r\nOK\r\n",&uart0_read_data[uart0_length-6],6))
            {
                step_master_bt++;
                DispStr1("[STEP_2 : MESSAGE RECEIVE OK]");
                DispStr1("[       CHANGE ROLE]");
                DispStr0("at+btrole=m\r");
            }
            else
                execution_error();
        break;
        case STEP_3:
            lf_check_flag = LF_CHECK_NON;
            if(!memcmp("\r\nOK\r\n",&uart0_read_data[uart0_length-6],6))
            {
                step_master_bt = STEP_1;
                DispStr1("[STEP_3 : MESSAGE RECEIVE OK]");
                DispStr1("[       RESET BLUETOOTH]");
                DispStr0("atz\r");
            }
            else
                execution_error();
        break;
```

< STEP_2 : Bluetooth가 슬레이브로 시작된 경우 >

1. Bluetooth로부터 “₩r₩nOK₩r₩n”메시지를 수신한 경우 STEP_3 진행
2. Bluetooth Device를 마스터로 변경
3. 비교 데이터가 틀린 경우 execution_error()루틴 진행

< STEP_3 : Bluetooth가 슬레이브로 시작된 경우 >

1. Bluetooth로부터 “₩r₩nOK₩r₩n”메시지를 수신한 경우 STEP_1 진행
2. Bluetooth Device Soft Reset 진행
3. 비교 데이터가 틀린 경우 execution_error()루틴 진행

20. FBD155MC_gcc compare_master_command( ) Source – 3th

```
        case STEP_4:
            lf_check_flag = LF_CHECK_NON;
            if(!memcmp("\r\nOK\r\n",&uart0_read_data[uart0_length-6],6))
            {
                step_master_bt++;
                status_target = TARGET_DETECT;
                DispStr1("[STEP_4 : MESSAGE RECEIVE OK]");
                DispStr1("[       TRY TO CONNECT]");
                DispStr0("atd");
                for(i = 0;i<12;i++)
                    PutChar0(target_data[i]);
                PutChar0(0x0d);
            }
            else
                execution_error();
        break;
        case STEP_5:
            lf_check_flag = LF_CHECK_NON;
            if(!memcmp("\r\nOK\r\n",&uart0_read_data[uart0_length-6],6))
            {
                step_master_bt++;
                DispStr1("[STEP_5 : MESSAGE RECEIVE OK]");
            }
            else
                execution_error();
        break;
        case STEP_6:
            lf_check_flag = LF_CHECK_NON;
            if(!memcmp("\r\nCONNECT",&uart0_read_data[0],9))
            {
                step_master_bt++;
                status_sequence = AFTER_CONNECT;
                DispStr1("[STEP_6 : COMMUNICATION START]");
            }
            else
                execution_error();
        break;
    } ? end switch step_master_bt ?
    init_uart0_read_data();
} ? end compare_master_command ?
```

< STEP_4 : Bluetooth가 마스터로 시작된 경우 >

1. Bluetooth로부터 “₩r₩nOK₩r₩n”메시지를 수신한 경우 STEP_5 진행
2. Bluetooth Slave Device와 연결 시도
3. 비교 데이터가 틀린 경우 execution_error()루틴 진행

< STEP_5 : Bluetooth가 마스터로 시작된 경우 >

1. Bluetooth로부터 “₩r₩nOK₩r₩n”메시지를 수신한 경우 STEP_6 진행
2. 비교 데이터가 틀린 경우 execution_error()루틴 진행

< STEP_6 : Bluetooth가 마스터로 시작된 경우 >

1. Bluetooth로부터 “₩r₩nCONNECT”메시지를 수신한 경우 STEP_7 진행(STEP_7은 아무런 동작 없고, 비교 루틴을 나가는 동작 진행)
2. Bluetooth Device를 연결 상태로 설정
3. 비교 데이터가 틀린 경우 execution_error()루틴 진행

21. FBD155MC_gcc Initialize Function – port, timer0

```
void init_port (void)
{
    DDRA  = 0x00;
    PORTA = 0x0f;

    DDRB  = 0xff;
    PORTB = 0xf0;

    DDRC  = 0x03;
    PORTC = 0x00;

    DDRD  = 0xff;
    PORTD = 0x00;

    DDRE  = 0xff;
    PORTE = 0x00;

    DDRF  = 0xff;
    PORTF = 0x00;
} ? end init_port ?
```

< ATMega128 포트 초기화 >

1. 사용하는 포트는 LED로 할당된 PORTB의 4~7 비트, 딥 스위치로 할당된 PORTA의 4~7 비트
2. 딥 스위치 1번이 ON 되어 있으면 Slave로 동작, 딥 스위치 2번이 ON 되어 있으면 Master로 동작
3. Slave 동작 시 LED 1번 동작, Master 동작 시 LED 2번 동작
4. PORTB를 출력 포트로 설정 (DDRB = 0xFF)
5. LED는 Active Low (ATMega128에서 0 V 출력할 때 LED가 ON됨)
6. 초기값으로 PORTB의 상위 비트 (PORTB의 4~7 비트) 1로 설정 (PORTB = 0xF0 = 1111 0000)
7. PORTA를 입력 포트로 설정 (DDRA = 0x00)
8. 딥 스위치는 Active High (ATMega128에 3.3 V 또는 5 V 입력할 때 딥 스위치 인식)
9. 초기값으로 PORTA의 상위 비트 (PORTA의 4~7 비트) 0으로 설정 (PORTA = 0x0F = 0000 1111)

```
void init_timer0(void)
{

    TIMSK |= 1 << TOIE0;
    TCNT0 = 0;
    TCCR0 = 5;
    timer0_counter = 0;
}
```

< ATMega128 Timer0 초기화 >

1. ATMega128의 Timer0를 LED의 상태를 나타내는 간격으로 사용
2. Timer0의 인터럽트 허용 설정 (TIMSK |=1<<TOIE0)
3. Timer0의 카운터는 0부터 시작 (TCNT0 = 0)
4. Timer0의 분주 비 128 설정 (TCCR0 = 5 = 0x05 = 0000 0101)

22. FBD155MC_gcc Initialize Function – uart0, uart1

```
void Init_UART0(unsigned char baudrate)
{
    unsigned int i;

    UBRR0H    = 0;
    if(baudrate == BAUD_9600){UBRR0L = 71;}/* 9600 BAUD at 11.0592MHz*/
    else if(baudrate == BAUD_19200){UBRR0L = 35;}/* 19200 BAUD at 11.0592MHz*/
    else if(baudrate == BAUD_38400){UBRR0L = 17;}/* 38400 BAUD at 11.0592MHz*/
    else if(baudrate == BAUD_57600){UBRR0L = 11;}/* 57600 BAUD at 11.0592MHz*/
    else if(baudrate == BAUD_115200){UBRR0L = 5;} /* 115200 BAUD at 11.0592MHz*/
    else {UBRR0L = 71;};/* default 9600 BAUD at 11.0592MHz*/
    UCSR0B = (1<<RXCIE)|(1<<RXEN)|(1<<TXEN);
    p_rx0_wr = 0;
    p_rx0_rd = 0;
    for (i=0; i<UART0_BUF_SIZE; i++) rx0_buf[i] = 0;
}
```

< ATMega128 uart0 초기화 >

1. UBBR0L 설정에 따른 시리얼 데이터 통신 속도 설정
2. 수신 완료 인터럽트 허용 (RXCIE)
3. UART0 수신 부 동작 허용 (RXEN)
4. UART0 송신 부 동작 허용 (TXEN)
5. 수신 데이터 저장 버퍼 및 수신 데이터 포인터 초기값 설정

```
void Init_UART1(unsigned char baudrate)
{
    unsigned int i;

    UBRR1H    = 0;
    if(baudrate == BAUD_9600){UBRR1L = 71;}/* 9600 BAUD at 11.0592MHz*/
    else if(baudrate == BAUD_19200){UBRR1L = 35;}/* 19200 BAUD at 11.0592MHz*/
    else if(baudrate == BAUD_38400){UBRR1L = 17;}/* 38400 BAUD at 11.0592MHz*/
    else if(baudrate == BAUD_57600){UBRR1L = 11;}/* 57600 BAUD at 11.0592MHz*/
    else if(baudrate == BAUD_115200){UBRR1L = 5;} /* 115200 BAUD at 11.0592MHz*/
    else {UBRR1L = 71;};/* default 9600 BAUD at 11.0592MHz*/
    UCSR1B = (1<<RXCIE)|(1<<RXEN)|(1<<TXEN);
    p_rx1_wr = 0;
    p_rx1_rd = 0;
    for (i=0; i<UART1_BUF_SIZE; i++) rx1_buf[i] = 0;
}
```

< ATMega128 uart1 초기화 >

1. UBBR1L 설정에 따른 시리얼 데이터 통신 속도 설정
2. 수신 완료 인터럽트 허용 (RXCIE)
3. UART0 수신 부 동작 허용 (RXEN)
4. UART0 송신 부 동작 허용 (TXEN)
5. 수신 데이터 저장 버퍼 및 수신 데이터 포인터 초기값 설정

23. FBD155MC_gcc Uart0 Function

```
unsigned char PutChar0 (unsigned char c)
{
    while (!(UCSR0A & 0x20)) ;
    UDR0 = c;
    return 0;
}

void DispStr0 (unsigned char *s)
{
    while (*s != '\0') {
        PutChar0(*s++);
    }
}

unsigned int Check_Rx_Buf0(void)
{
    unsigned int len;

    if (p_rx0_wr == p_rx0_rd) {
        len = 0;
    }
    else {
        if (p_rx0_wr > p_rx0_rd) len = p_rx0_wr - p_rx0_rd;
        else len = UART0_BUF_SIZE + p_rx0_wr - p_rx0_rd;
    }
    return len;
}

unsigned char GetChar0 (void)
{
    unsigned char ch;

    ch = rx0_buf[p_rx0_rd];
    p_rx0_rd++;
    if (p_rx0_rd > UART0_BUF_SIZE-1) p_rx0_rd = 0;
    return ch;
}
```

< PutChar0( ) >

1. UCSR0A 레지스터의 5번째 비트가 1 이 되면 1 바이트 시리얼 데이터를 UART0 포트로 출력

< DispStr0( ) >

1. 문자열로 된 시리얼 데이터를 UART0 포트로 출력

< Check_Rx_Buf0( ) >

1. UART0로 입력된 데이터는 rx0_buf[] 버퍼에 저장
2. Check_Rx _Buf0() 함수는 rx0_buf[] 버퍼에 저장되어 있는 데이터의 개수를 체크하는 함수

< GetChar0( ) >

1. rx0_buf[]에 저장되어 있는 데이터를 1바이트씩 꺼내오는 함수

24. FBD155MC_gcc Uart1 Function

```
unsigned char PutChar1 (unsigned char c)
{
    while (!(UCSR1A & 0x20)) ;
    UDR1 = c;
    return 0;
}

void DispStr1 (unsigned char *s)
{
    while (*s != '\0') {
        PutChar1(*s++);
    }
    PutChar1(0x0a);
    PutChar1(0x0d);
}

void DispStr1_line (unsigned char *s)
{
    while (*s != '\0') {
        PutChar1(*s++);
    }
}

unsigned int Check_Rx_Buf1(void)
{
    unsigned int len;

    if (p_rx1_wr == p_rx1_rd) {
        len = 0;
    }
    else {
        if (p_rx1_wr > p_rx1_rd) len = p_rx1_wr - p_rx1_rd;
        else len = UART1_BUF_SIZE + p_rx1_wr - p_rx1_rd;
    }
    return len;
}

unsigned char GetChar1 (void)
{
    unsigned char ch;

    ch = rx1_buf[p_rx1_rd];
    p_rx1_rd++;
    if (p_rx1_rd > UART1_BUF_SIZE-1) p_rx1_rd = 0;
    return ch;
}
```

< PutChar1( ) >

1. UCSR1A 레지스터의 5번째 비트가 1 이 되면 1 바이트 시리얼 데이터를 UART1 포트로 출력

< DispStr1( ) >

1. 문자열로 된 시리얼 데이터를 UART1 포트로 출력
2. 마지막에 0x0a(Line Feed) 0x0d(Carriage return)를 출력

< DispStr1_line( ) >

1. 문자열로 된 시리얼 데이터를 UART1 포트로 출력
2. 마지막에 0x0a(Line Feed) 0x0d(Carriage return)를 출력하지 않음

< Check_Rx_Buf1( ) >

1. UART1로 입력된 데이터는 rx1_buf[] 버퍼에 저장
2. Check_Rx _Buf1() 함수는 rx1_buf[] 버퍼에 저장되어 있는 데이터의 개수를 체크하는 함수

< GetChar1( ) >

1. rx1_buf[]에 저장되어 있는 데이터를 1바이트씩 꺼내오는 함수

25. FBD155MC_gcc LED 처리 함수

```
void status_led(void)
{
    if(status_sequence == BEFORE_CONNECT)
    {
        if (timer0_counter < 100 )
        {
            if(read_avr_port == DIP_1)
                write_port_value(WRITE_LED_1_ON);
            else if(read_avr_port == DIP_2)
                write_port_value(WRITE_LED_2_ON);
        }
        else if (timer0_counter < 200 )
        {
            if(read_avr_port == DIP_1)
                write_port_value(WRITE_LED_1_OFF);
            else if(read_avr_port == DIP_2)
                write_port_value(WRITE_LED_2_OFF);
        }
    }
    else
    {
        if(read_avr_port == DIP_1)
            write_port_value(WRITE_LED_1_ON);
        else if(read_avr_port == DIP_2)
            write_port_value(WRITE_LED_2_ON);
    }
} ? end status_led ?
```

< status_led( ) >

1. LED의 상태를 나타내는 함수
2. timer0_counter는 timer0 인터럽트 벡터 상에서 카운트 됨
3. Bluetooth가 연결되기 전에는 LED 가 깜빡임
4. Bluetooth가 연결되면 LED가 ON 된 상태 유지
5. 딥 스위치 1번이 ON 된 상태에서는 1번 LED가 동작
6. 딥 스위치 2번이 ON 된 상태에서는 2번 LED가 동작
7. LED의 ON/OFF는 write_port_value( ) 함수 사용

26. FBD155MC_gcc Hex To Char 변경 / 시간 지연 함수

```
/*
*****************************************
CONVERT HEX INTO CHAR
*****************************************
*/
unsigned char Hex2Char(unsigned char c)
{
     if (c >= 0 && c <= 9)
          return '0' + c;
     if (c >= 10 && c <= 15)
          return 'A' + c - 10;

     return c;
}

/*
*****************************************
Function For Delay
*****************************************
*/
void wait_1ms(unsigned int cnt)
{
     unsigned int i;

     for (i = 0; i < cnt; i++) wait_1us(1000);
}

void wait_1us(unsigned int cnt)
{
     unsigned int i;

     for (i = 0; i < cnt; i++) ;
}
```

< Hex2Char( ) >

1. 1 바이트의 HEX 값을 1 바이트의 Char로 변경
2. 하이퍼 터미널과 같은 시리얼 통신 프로그램에서 정상적으로 데이터가 표시 되기 위해서는 HEX 값을 Char 형태로 변경해야 함
3. HEX값 0x1은 Char 타입의 1(HEX값으로 0x31)로 변경됨

27. FBD155MC_gcc 포트 읽기 / 포트 쓰기 함수

```
/*
*********************************************************************
Function For PORT Read
*********************************************************************
*/
unsigned char read_port_value(unsigned char avr_port_num)
{
    unsigned char return_avr_port_value;

    switch(avr_port_num)
    {
        case READ_DIP:
            return_avr_port_value = PINA;
            return_avr_port_value = return_avr_port_value & 0xf0;
            break;
        case READ_SWITCH:
            return_avr_port_value = PINA;
            return_avr_port_value = return_avr_port_value & 0x0f;
            break;
        default:
            break;
    }
    return return_avr_port_value;
}
/*
*********************************************************************
Function For PORT Write
*********************************************************************
*/
void write_port_value(unsigned char avr_port_value)
{

    switch(avr_port_value)
    {
        case WRITE_LED_1_ON:
            cbi(PORTB,PB4);
            break;
        case WRITE_LED_1_OFF:
            sbi(PORTB,PB4);
            break;
        case WRITE_LED_2_ON:
            cbi(PORTB,PB5);
            break;
        case WRITE_LED_2_OFF:
            sbi(PORTB,PB5);
            break;
        default:
            break;
    }
} ? end write_port_value ?
```

< read_port_value( ) >

1. ATMega128의 포트를 읽어오는 함수
2. 딥 스위치 값으로 할당된 PORTA의 상위 비트 를 읽음 (...&0xF0)
3. 읽은 포트 값 리턴

< write_port_value( ) >

1. ATMega128의 포트 값을 설정하는 함수
2. LED로 할당된 PORTB의 값을 설정
3. cbi( ) (Low 출력) / sbi( ) (High 출력)를 이용하여 비트 별로 설정

28. FBD155MC_gcc 비교버퍼 초기화 / 에러 처리 함수

```
/*
*********************************************
* Function For Init uart0_read_data
*********************************************
*/
void init_uart0_read_data(void)
{
     unsigned int i;

     for(i = 0;i<50;i++)
          uart0_read_data[i] = 0;
}
/*
*********************************************
* Function For error
*********************************************
*/
void execution_error(void)
{
     unsigned int i;

     DispStr1_line("[ERROR ");
     DispStr1_line(" STEP : ");
     if(read_avr_port == DIP_1)
          PutChar1(Hex2Char(step_slave_bt));
     else if(read_avr_port == DIP_2)
          PutChar1(Hex2Char(step_master_bt));
     DispStr1_line(" UART_LEN : ");
     PutChar1(Hex2Char(uart0_length));
     DispStr1_line(" DATA : ");
     for(i = 0;i < uart0_length;i++)
          PutChar1(uart0_read_data[i]);
     if(read_avr_port == DIP_1)
          step_slave_bt = 50;
     else if(read_avr_port == DIP_2)
          step_master_bt = 50;

} ? end execution_error ?
```

< init_uart0_read_data( ) >

1. uart0_read_data[] 버퍼는 수신된 데이터의 비교 버퍼로 사용
2. uart0_read_data[] 버퍼의 데이터를 초기화 하는 함수

< execution_error( ) >

1. 수신 데이터 비교 중에 비교 데이터와 다른 경우 수행되는 함수
2. 시리얼로 “ERROR”를 출력하고, ERROR가 발생된 현재의 STEP 출력, Bluetooth로부터 수신 받은 데이터의 길이 출력, Bluetooth 로부터 수신 받은 데이터 출력
3. ERROR 처리 후, 스텝을 50으로 설정하여 더 이상 다른 스텝이 진행되지 않게 설정

29. FBD155MC_gcc Interrupt Vector

```
SIGNAL(SIG_UART0_RECV)
{
    rx0_buf[p_rx0_wr++] = UDR0;
    if (p_rx0_wr > UART0_BUF_SIZE-1)p_rx0_wr = 0;
    //
    if(rx0_buf[p_rx0_wr-1] == LF_VALUE)lf_check_flag++;
}
```

< uart0 interrupt vector >

1. Uart0로 데이터가 입력된 경우 수행되는 부분
2. Uart0으로 입력된 데이터는 UDR0 에 저장되어 있음
3. UDR0 에 저장되어 있는 1바이트의 데이터를 rx0_buf [] 버퍼에 저장
4. rx0_buf[]의 저장 공간이 더 이상 없는 경우, rx0_buf[]의 처음부터 저장 (링 버퍼 구성)
5. 입력된 데이터가 0x0a (Line Feed)인지 검사
6. Bluetooth로부터 수신된 데이터의 비교 시작을 0x0a가 2개 검출된 시점부터 적용

```
SIGNAL(SIG_UART1_RECV)
{
    rx1_buf[p_rx1_wr++] = UDR1;
    if (p_rx1_wr > UART1_BUF_SIZE-1)p_rx1_wr = 0;
}
```

< uart1 interrupt vector >

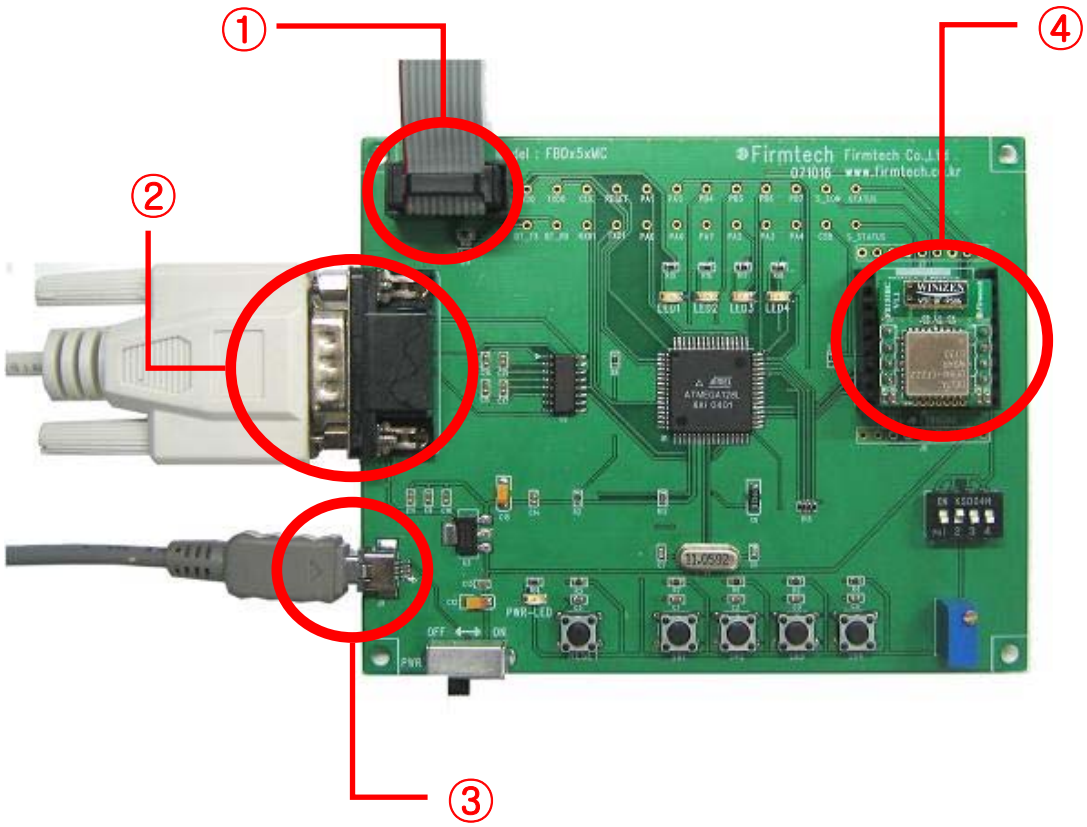
1. Uart1로 데이터가 입력된 경우 수행되는 부분
2. Uart1로 입력된 데이터는 UDR1 에 저장되어 있음
3. UDR1 에 저장되어 있는 1바이트의 데이터를 rx1_buf [] 버퍼에 저장
4. rx1_buf[]의 저장 공간이 더 이상 없는 경우, rx1_buf[]의 처음부터 저장 (링 버퍼 구성)

```
SIGNAL(SIG_OVERFLOW0)
{
    timer0_counter++;
    if (timer0_counter > 200 )
    {
        timer0_counter = 0;
    }
    status_led();
}
```

< timer0 interrupt vector >

1. timer0으로 설정된 시간이 되면 수행되는 부분
2. 설정된 시간마다 timer0_counter를 1씩 증가
3. timer0_counter가 200보다 커지면 timer0_counter를 0으로 설정
4. 설정된 시간마다 status_led( )함수를 이용하여 LED 상태 설정

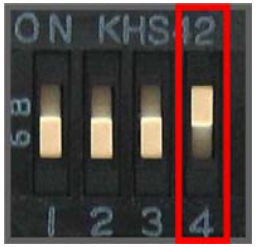
30. FBDx5xMC 보드의 올바른 연결



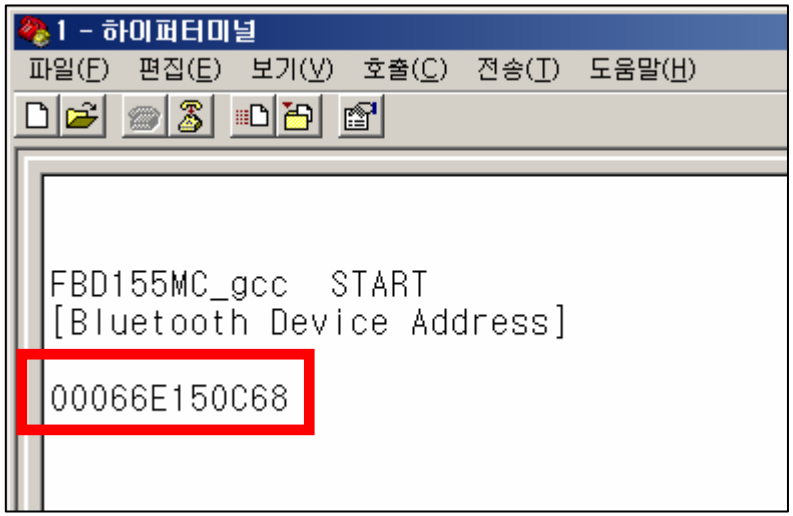
< 제품의 올바른 연결 >

1. PC와 FBDx5xMC를 AVR Loader로 연결
2. PC와 FBDx5xMC를 RS-232 Cable로 연결
3. PC와 FBDx5xMC를 USB Power Cable로 연결
4. FB155BC와 FBDx5xMC를 연결

31. Bluetooth Device Address (Slave Address) 확인 하는 방법



< 딥 스위치 4번 ON 상태 >



< 하이퍼 터미널에 출력된 Bluetooth Device Address >

< Bluetooth Device Local Address 확인 하는 방법>

1. FBDx5xMC 보드에 Bluetooth Device를 장착
   - 장착 방향에 유의
2. FBDx5xMC 보드와 PC를 RS-232 Cable로 연결
3. FBDx5xMC 보드에 Power Cable 연결
4. PC에서 하이퍼 터미널 실행
   - 통신속도 : 9600bps
   - 데이터 비트 : 8 비트
   - 패리티 : 없음
   - 정지 비트 : 1
   - 흐름제어 : 없음
5. FBDx5xMC 보드의 딥 스위치 4번 ON
6. FBDx5xMC 보드의 전원 ON
7. Bluetooth Device Address 하이퍼 터미널로 출력
   - 기본적으로 제공되는 소스를 이용하여 Bluetooth Device Address 확인 가능
   - HEX 파일 다운로드 없이 기본적으로 동작되는 FBDx5xMC보드에 Bluetooth Device 장착하여 Bluetooth Device Address 확인 가능

32. FBD155MC_gcc 프로그램 수정 (Slave Address 수정)

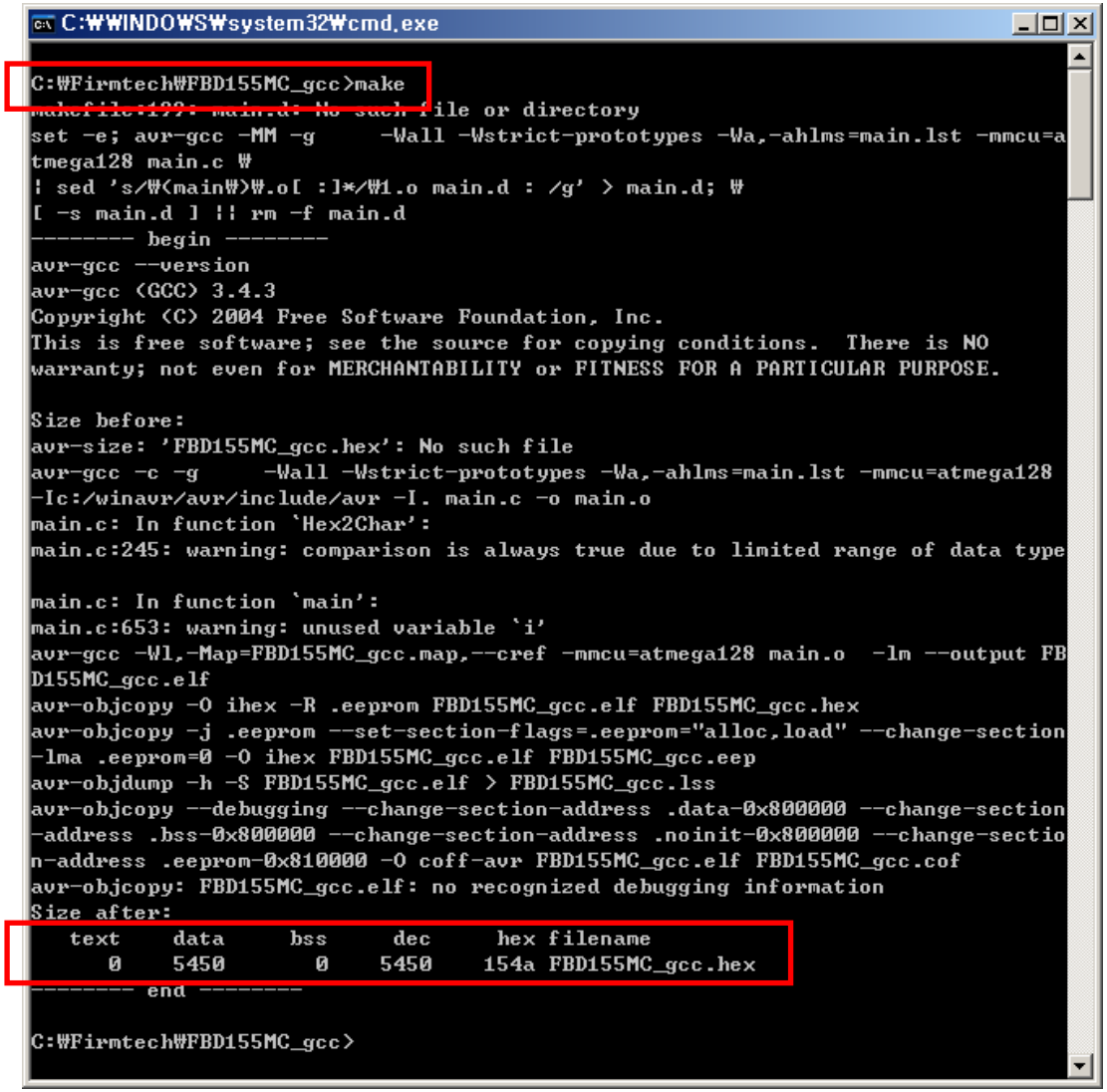
```
void execution_master(void)
{
    unsigned int i;

    //
    memcpy(target_data,"00066e150c68",12);
    //
    status_sequence = BEFORE_CONNECT;
    step_master_bt = STEP_1;
    init_uart0_read_data();
    while(1)
    {
        if(status_sequence == BEFORE_CONNECT)
        {
            uart0_length = Check_Rx_Buf0();
            if(uart0_length>0)
            {
                if(lf_check_flag == 2)
                {
                    uart0_length = Check_Rx_Buf0();
                    for(i = 0;i<uart0_length;i++)
                        uart0_read_data[i] = GetChar0();
                    compare_master_command();
                }
            }
        }
        else if(status_sequence == AFTER_CONNECT)
        {
            lf_check_flag = LF_CHECK_NON;
            uart0_length = Check_Rx_Buf0();
            if(uart0_length)
            {
                for(i = 0;i < uart0_length;i++)
                    PutChar1(GetChar0());
            }
            uart1_length = Check_Rx_Buf1();
            if(uart1_length)
            {
                for(i = 0;i < uart1_length;i++)
                    PutChar0(GetChar1());
            }
        }
    } ? end while 1 ?
} ? end execution_master ?
```

< 슬레이브 어드레스 저장 >

1. execution_master( ) 소스에 확인된 Slave Bluetooth Device의 Address를 입력
2. Slave의 Address를 정확히 입력하지 않으면 Master / Slave의 연결이 원활하지 않음
3. 예제 소스의 00066e150c68 부분에 확인된 Slave Bluetooth Device의 Address 12바이트를 정확하게 입력
4. 수정 완료된 프로그램 컴파일 진행
5. 컴파일 진행 후 생성된 HEX 파일 FBDx5xMC 보드에 다운로드

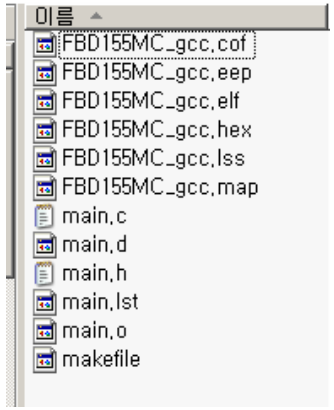
33. FBD155MC_gcc 프로그램 컴파일 (WINAVR 컴파일러 사용)



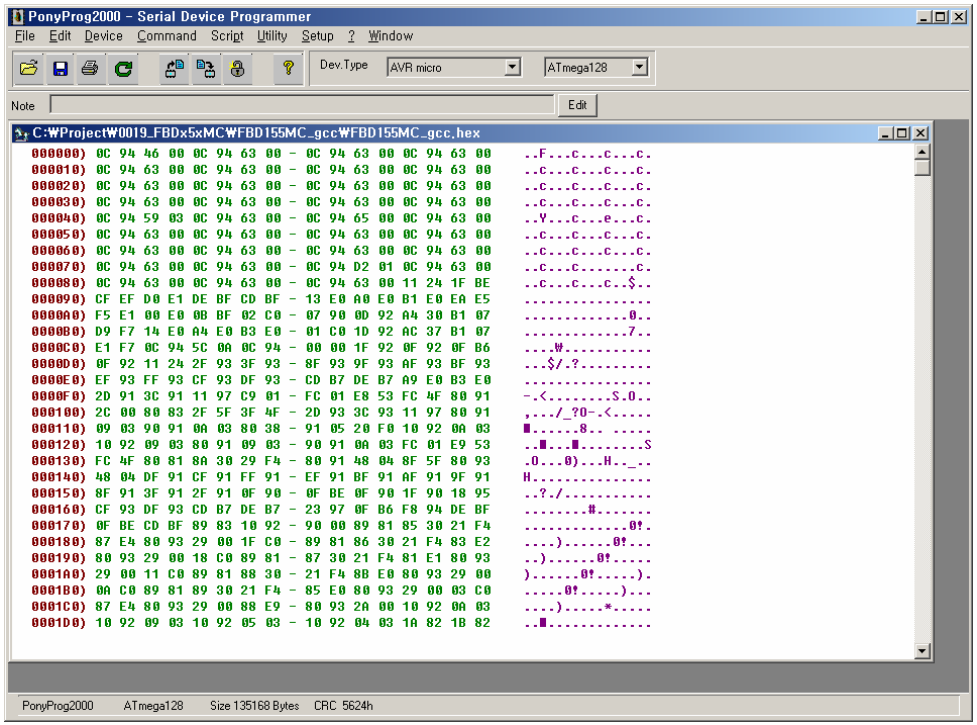
1. 사용 컴파일러
   - 사용하는 컴파일러는 WINAVR 사용
   - http://winavr.sourceforge.net 에서 다운로드 하여 설치
2. 컴파일에 사용하는 파일
   - main.c
   - main.h
   - makefile
3. 컴파일 방법
   - Window Command 창을 이용하여 컴파일에 사용할 파일이 있는 위치로 이동
   - Make 입력 후 Enter Key 입력
   - ERROR 없이 진행된 경우, HEX 파일과 기타 파일 생성

< make 실행 이전 파일>

< make 실행 이후 파일>



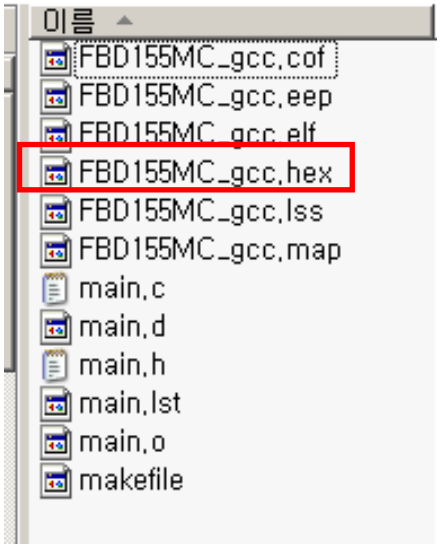
34. FBDx5xMC 보드에 프로그램 다운로드



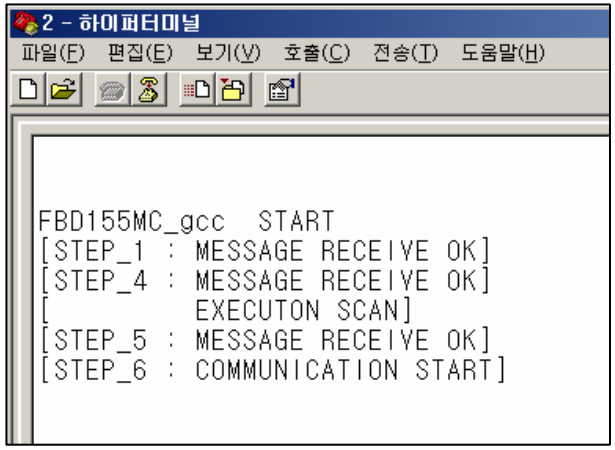
< HEX 파일 다운로드 프로그램 PonyProg>

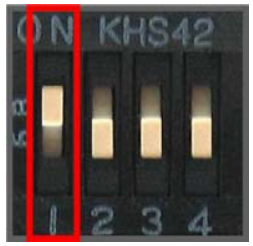
1. Slave Bluetooth Device Address를 확인하고, 확인한 Slave Bluetooth Device Address를 소스에 입력 후 컴파일 진행
   - WINAVR을 이용하여 컴파일 진행
2. 컴파일에 의해 생성된 HEX 파일을 FBDx5xMC 보드에 다운로드
   - 다운로드 프로그램은 PonyProg 사용
   - http://www.lancos.com/ppwin95.html 에서 다운로드 하여 설치
   - AVR Loader로 PC와 FBDx5xMC 보드 연결
   - FBDx5xMC 보드 2개에 생성된 HEX 파일을 다운로드

< 다운로드 할 HEX 파일>



35. FBDx5xMC 보드의 Master / Slave 설정 & 연결

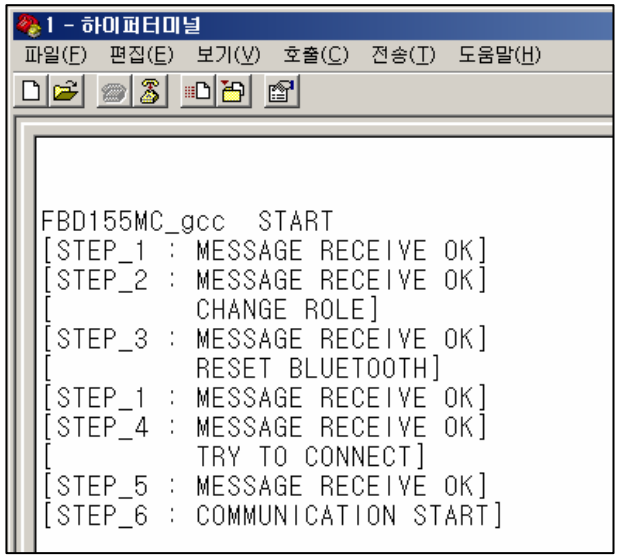


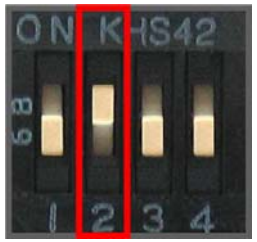


< 딥 스위치 1번 ON 상태 >

< Slave 설정 & 연결 >

1. FBDx5xMC 보드를 Slave로 동작 시키기 위해 딥 스위치 1번 ON
2. FBDx5xMC 보드에 Bluetooth Device를 장착 후 전원 ON
3. Slave로 동작되는 경우, 1번 LED가 깜빡임
4. 하이퍼 터미널에 각 스텝 별 메시지 출력
   - STEP_1 : BTWIN Slave mode start 메시지 수신
   - STEP_4 : OK 메시지 수신, at+btscan 명령 전달
   - STEP_5 : OK 메시지 수신
   - STEP_6 : CONNECT 메시지 수신 (마스터와 연결된 경우 1번 LED가 ON 된 상태 유지)





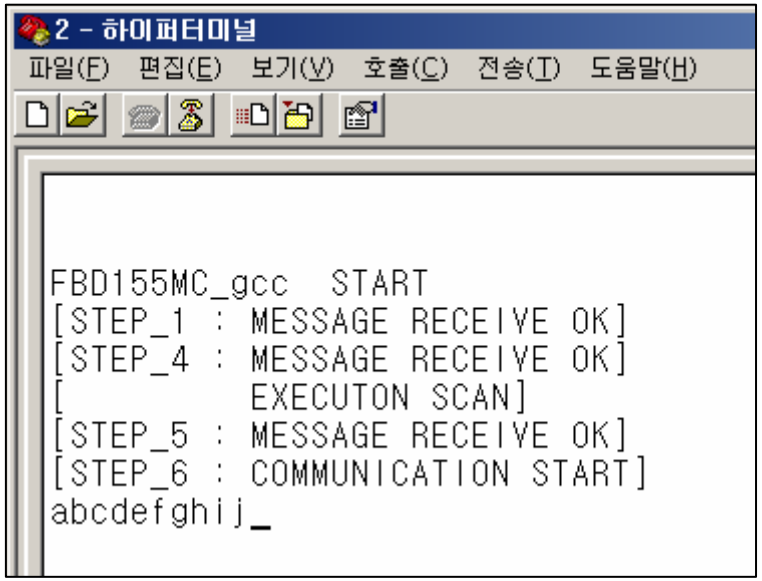
< 딥 스위치 2번 ON 상태 >

< Master 설정 & 연결>

* Master가 실행되기 이전에 Slave가 SCAN 동작을 진행하고 있어야 함

1. FBDx5xMC 보드를 Master로 동작 시키기 위해 딥 스위치 2번 ON
2. FBDx5xMC 보드에 Bluetooth Device를 장착 후 전원 ON
3. Master로 동작되는 경우, 2번 LED가 깜빡임
4. 하이퍼 터미널에 각 스텝 별 메시지 출력
   - STEP_1 : BTWIN Slave mode start 메시지 수신
   - STEP_2 : OK 메시지 수신, at+btrole=m 명령 전달
   - STEP_3 : OK 메시지 수신, atz 명령 전달
   - STEP_1 : BTWIN Master mode start 메시지 수신
   - STEP_4 : OK 메시지 수신, atd001122334455 명령 전달
   - STEP_6 : CONNECT 메시지 수신 (슬레이브와 연결된 경우 2번 LED가 ON 된 상태 유지)

36. FBD155MC_gcc 시리얼 데이터 통신

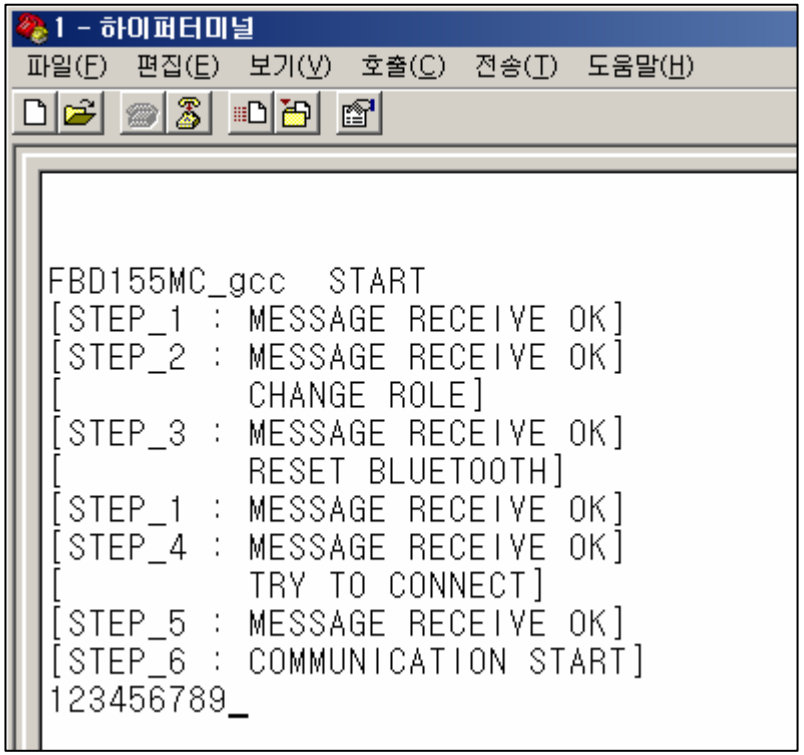


< Slave 확인용 하이퍼 터미널 창>

각 스텝 별 진행

연결된 후 123456789 입력

상대편 Bluetooth Device로부터 abcdefghij 수신



< Master 확인용 하이퍼 터미널 창>

각 스텝 별 진행

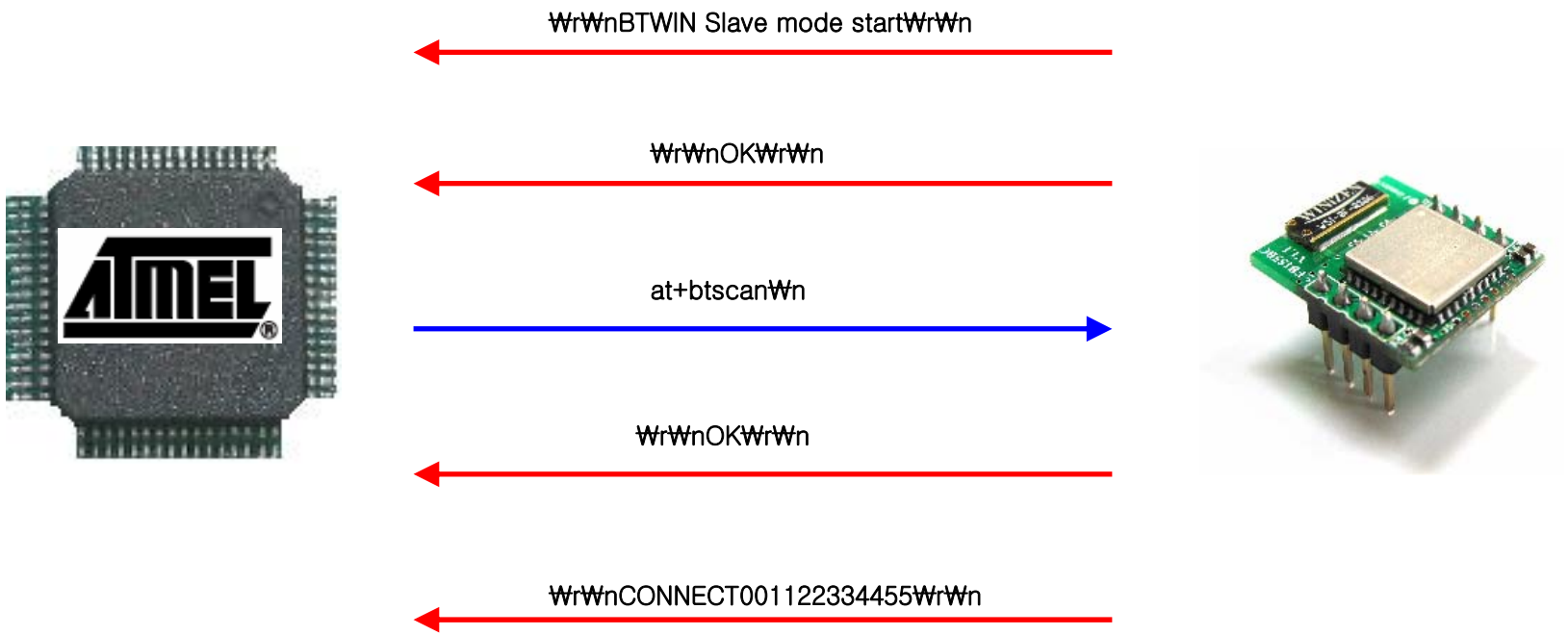
연결된 후 abcdefghij 입력

상대편 Bluetooth Device로부터 123456789 수신

37. 사용되는 상태 메시지 및 AT-Command 설명

1. “BTWIN Slave mode start”
   - 최초 스타트 메시지
   - Bluetooth Module 이 Slave로 설정되어 있는 경우, 시작되면서 출력되는 메시지
   - FBDx5xMC 보드에서 최초에 이 메시지를 검출
2. “BTWIN Master mode start”
   - 최초 스타트 메시지
   - Bluetooth Module 이 Master로 설정되어 있는 경우, 시작되면서 출력되는 메시지
   - FBDx5xMC 보드에서 최초에 이 메시지를 검출
3. “OK”
   - 최초 스타트 메시지 출력이 후 AT-Command를 인식할 수 있는 상태를 나타내는 메시지
   - Bluetooth Module에 AT-Command를 입력한 경우, 입력된 AT-Command를 바르게 인식한 경우 출력되는 메시지
   - 모든 AT-Command에 대해서 “OK” 메시지를 출력하지는 않음
4. “ERROR”
   - Bluetooth Module에 AT-Command를 입력한 경우, 입력된 AT-Command를 바르게 인식하지 못한 경우 출력되는 메시지
5. “at+btrole=s”
   - Bluetooth Module의 Role을 Slave로 변경하기 위한 AT-Command
6. “at+btrole=m”
   - Bluetooth Module의 Role을 Master로 변경하기 위한 AT-Command
7. “atz”
   - Bluetooth Module을 Soft Reset (전원을 껐다가 켜는 동작과 같음) 시키는 AT-Command
   - 설정을 변경하는 AT-Command중에 Soft Reset을 시켜야 정상적으로 설정되는 AT-Command가 있음
8. “at+btscan”
   - Slave로 설정된 Bluetooth Module 이 Master에서 검색/연결이 가능하도록 검색/연결 대기 상태로 진입하는 AT-Command
9. “atd”
   - 검색/연결 대기 중인 Slave Bluetooth Module과 연결을 하기 위한 AT-Command
   - “atd “ Command 이후에 연결하고자 하는 Slave Bluetooth Module의 Address 입력
10. “CONNECT”
    - Master와 Slave가 연결된 경우 출력되는 메시지

38. 상태 메시지와 AT-Command Flow – Slave를 Slave로 동작



39. 상태 메시지와 AT-Command Flow – Master를 Slave로 동작

₩r₩nBTWIN Master mode start₩r₩n

₩r₩nOK₩r₩n

at+btrole=s₩n

₩r₩nOK₩r₩n

atz₩n

₩r₩nBTWIN Slave mode start₩r₩n

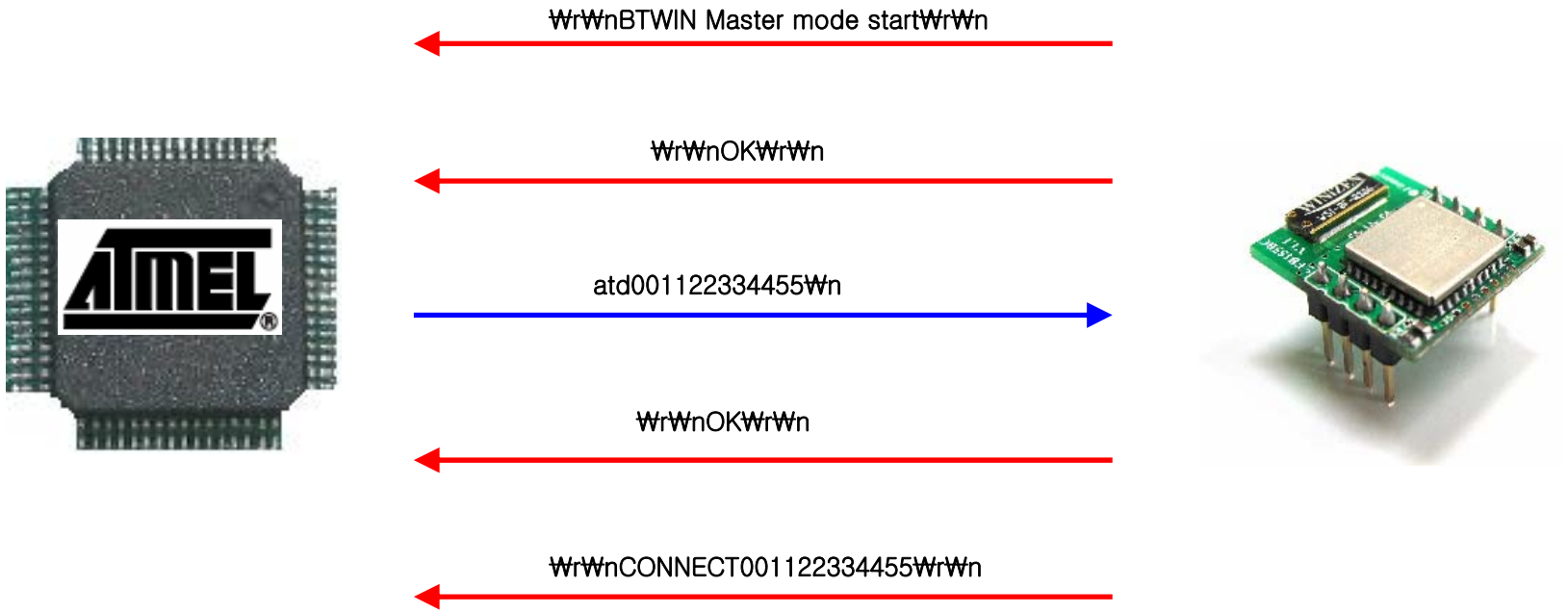
₩r₩nOK₩r₩n

at+btscan₩n

₩r₩nOK₩r₩n

₩r₩nCONNECT001122334455₩r₩n

40. 상태 메시지와 AT-Command Flow – Master를 Master로 동작



41. 상태 메시지와 AT-Command Flow – Slave를 Master로 동작

₩r₩nBTWIN Slave mode start₩r₩n

₩r₩nOK₩r₩n

at+btrole=m₩n

₩r₩nOK₩r₩n

atz₩n

₩r₩nBTWIN Master mode start₩r₩n

₩r₩nOK₩r₩n

atd001122334455₩n

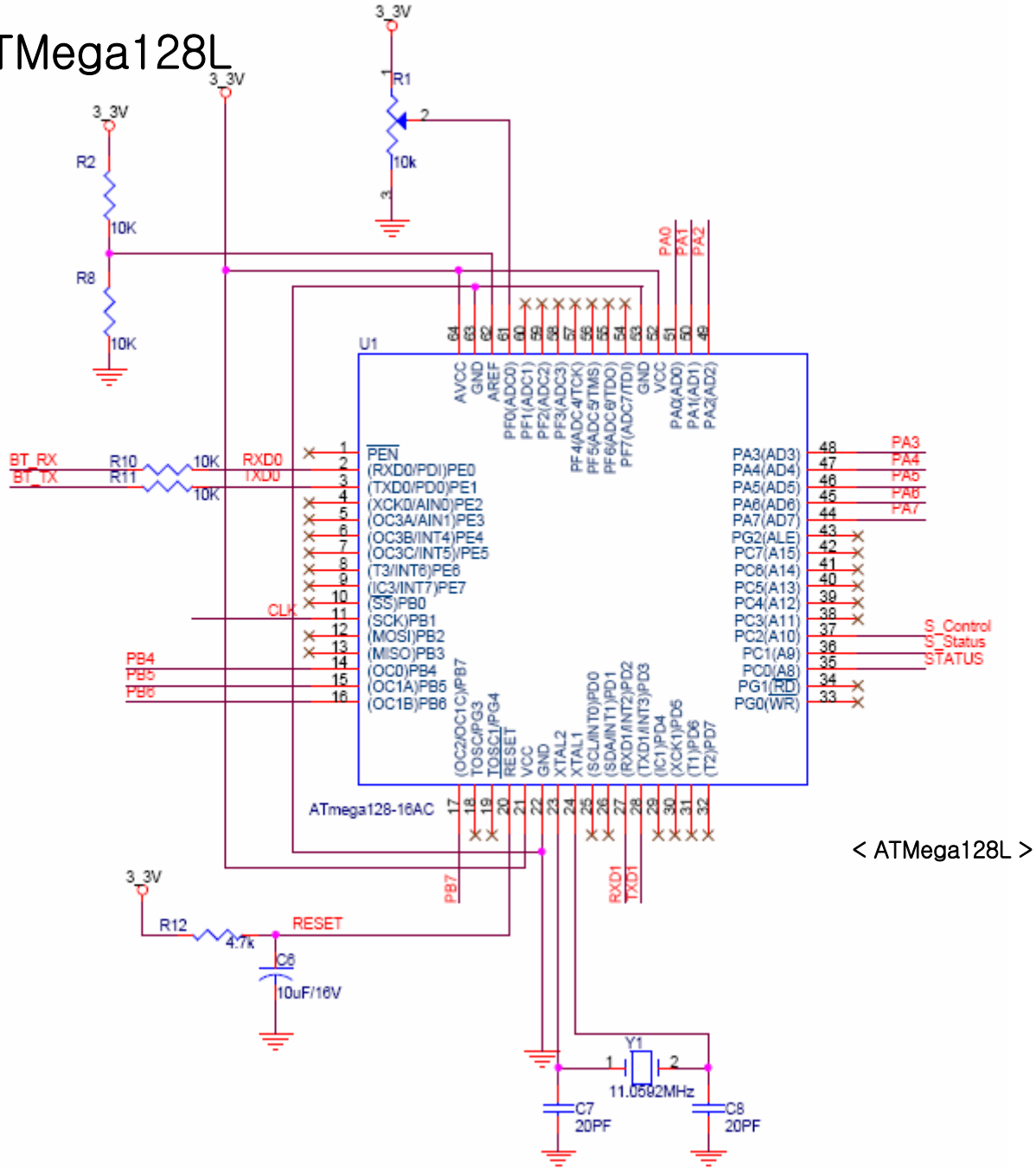
₩r₩nOK₩r₩n

₩r₩nCONNECT001122334455₩r₩n

※ FBDx5xMC(FBD155MC_gcc) 사용시 주의 사항

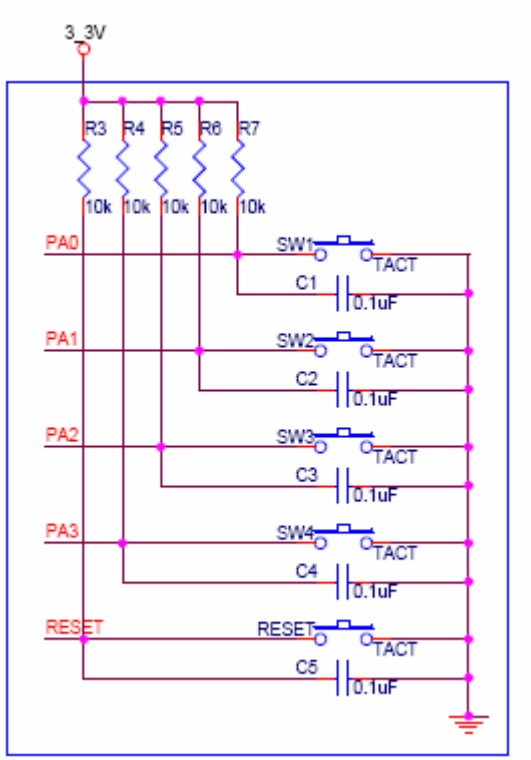
1. Slave Bluetooth Device Address의 정확한 입력
   - Master Bluetooth Device가 Slave Bluetooth Device와 연결하기 위해서 Slave Bluetooth Device의 Address 필요
   - Slave Bluetooth Device의 정확한 Address를 execution_master() Source에 정확히 입력 후 컴파일 진행
   - Slave Bluetooth의 Address가 정확하지 않으면 연결이 되지 않음
2. AVR Loader의 연결
   - 사용자의 PC를 최초에 ON 하고, PonyProg 프로그램을 실행시키지 않은 상태에서, PC와 FBDx5xMC 보드 사이에 AVR Loader를 연결한여 FBDx5xMC 보드를 동작시키면 정상적으로 동작 되지 않음
   - FBDx5xMC 보드를 정상 동작 시키기 위해서, PC와 FBDx5xMC 보드 사이에 AVR Loader가 연결되어 있는 경우에는 PonyProg를 실행시켜야 함
   - FBDx5xMC 보드를 정상 동작 시키기 위해서, HEX 파일을 다운로드 시키는 경우를 제외하고는 PC와 FBDx5xMC 보드 사이에 AVR Loader를 연결하지 말아야 함
3. Bluetooth Device의 올바른 방향 장착
   - FBDx5xMC 보드에 장착하는 Bluetooth Device의 장착 방향 유의
   - Bluetooth Device를 잘못 장착 한 경우 FBDx5xMC 보드 뿐만 아니라 Bluetooth Device에도 무리가 발생할 수 있음
4. Bluetooth Device의 설정 상태
   - FBDx5xMC 보드는 장착되는 Bluetooth Device의 설정 상태가 공장 초기 설정 값을 기반으로 동작됨
   - 기본 설정 값 : Connect Mode 4 (AT-Command Mode), Status Message Enable, Baud rate 9600bps
5. Bluetooth Device 동작 순서
   - Slave Bluetooth Device가 Scan 동작을 진행한 이후에, Master Bluetooth Device가 연결을 진행 해야 함

회로도 – ATMega128L

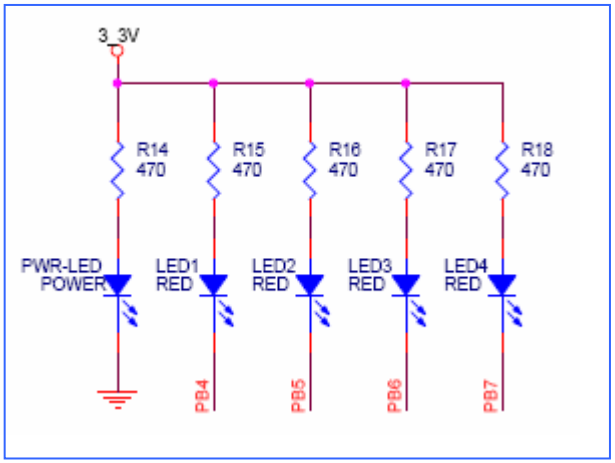


< ATMega128L >

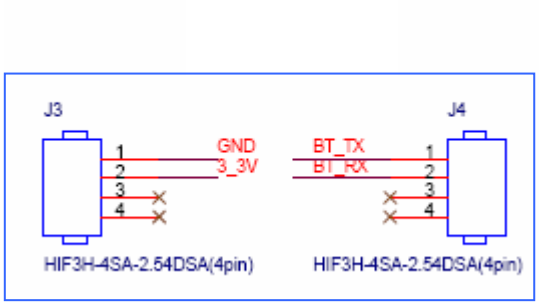
회로도 – 스위치 & LED & FB155BC 커넥터 & 딥 스위치



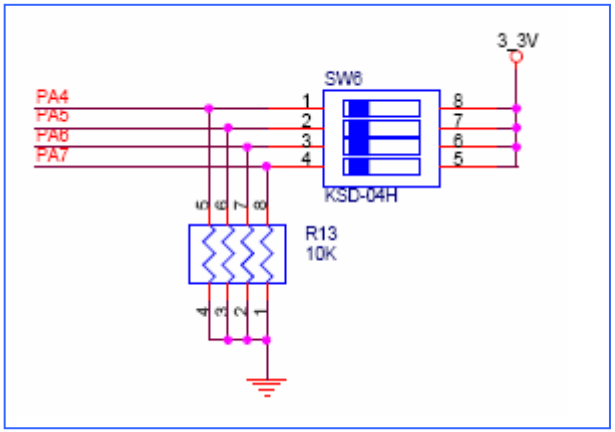
< Switch >



< LED >

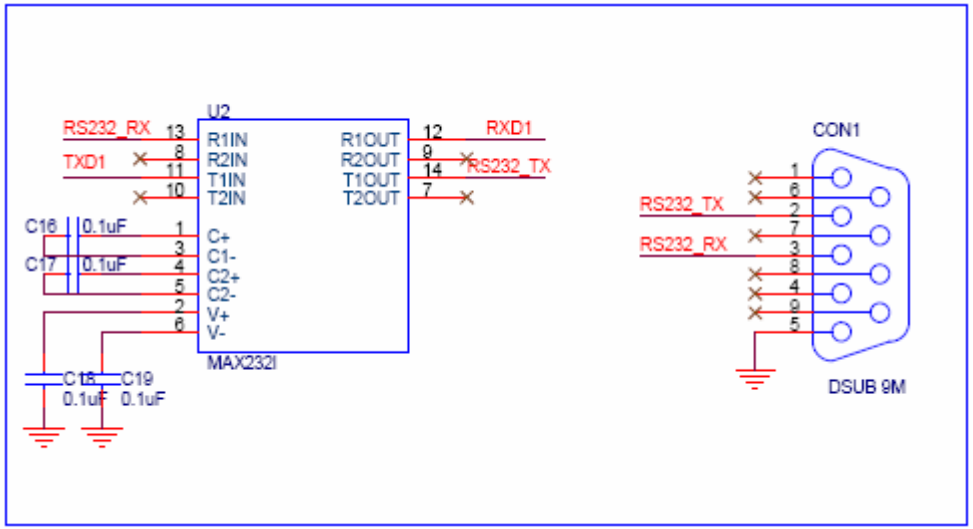


< FB155BC Connector >

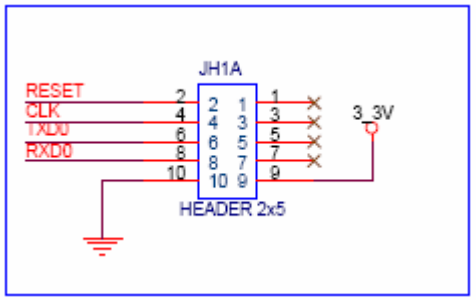


< Dip Switch >

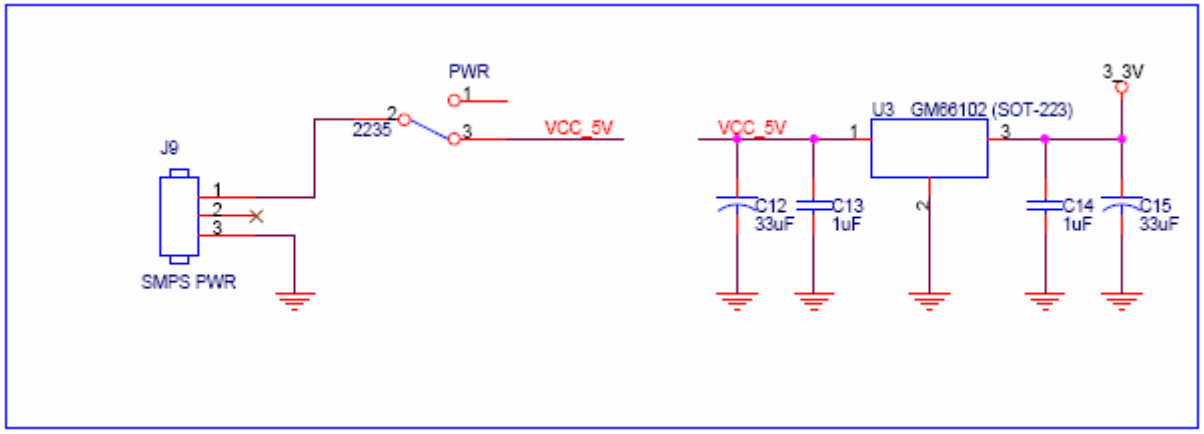
회로도 – RS-232 & AVR Loader & Power



< RS – 232 >



< AVR Loader >



< Power >