

# UART

- UART와 RS232 개요
- ATmega128의 USART 포트
- UART로 Hello 보내기
- UART로 스위치 입력 값 보내기
- UART로 PC와 데이터 주고받기
- UART로 PC에서 FND 점멸시키기



엣지아이랩

# UART와 RS232 개요

- UART(Universal Asynchronous Receiver/Transmitter)
  - 시리얼 기반의 통신 방식으로 일반적으로 RS232 프로토콜을 통해 원격지와 통신을 지원하는 방식
  - UART는 컴퓨터에게 RS-232C DTE 인터페이스를 제공함으로써, 모뎀이나 기타 다른 직렬장치들과 통신하거나 데이터를 주고받을 수 있게 함
  - UART의 동작
    - 병렬 데이터를 하나의 단일 직렬 비트 스트림으로 변환
      - 직렬 비트 스트림을 컴퓨터가 처리할 수 있도록 바이트로 변환
      - 패리티 비트 처리
      - 시작 비트와 정지 비트 처리
      - 키보드나 마우스로부터 들어오는 인터럽트 처리

# UART와 RS232 개요

- RS-232C
  - 직렬전송을 위한 규격
  - 1969년 미국의 EIA (Electric Industries Association)에 의해서 정해진 표준 인터페이스
  - "직렬 2진 데이터의 교환을 하는 데이터 터미널 장비(DTE)와 데이터 통신장비(DCE)간의 인터페이스의 제반을 규정하는 것"
  - RS-232C의 동작
    - 병렬을 직렬로 직렬을 병렬로 바꾸어 주는 작업
    - 스타트 비트와 스톱비트 포함하여 10비트를 1바이트로 보냄
    - RXD, TXD 라인을 통해 신호를 송수신
    - RS232 Transceiver를 통해 전송 전압을 끌어 올려 보다 먼 거리까지 전송

# UART와 RS232 개요

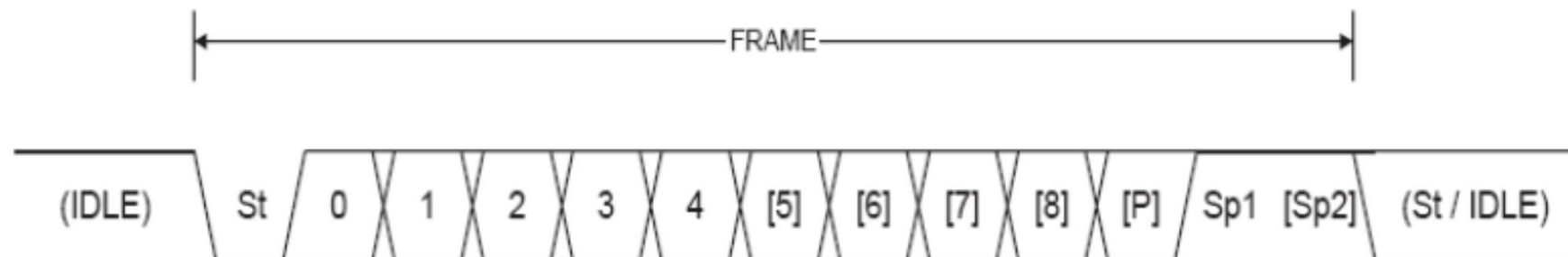
- RS-232C를 이용한 비동기식 전송시 규약
  - 통신속도
    - 시간당 데이터를 전송할 수 있는 양
    - baud rate : 1초당 전송되는 변조된 신호의 수
  - 스톱비트
    - 데이터의 시작과 끝을 알리는 스타트와 스톱 비트를 사용
    - 전송을 시작할 경우 1을 내보내고 8비트를 전송한 후 스톱비트를 전송
    - 스타트 비트는 고정/ 스톱 비트는 1과 1.5, 2비트중 하나를 선택
  - 패리티
    - 오류 검출을 위해 사용
    - 패리티의 종류는 짝수 및 홀수 방식과 사용하지 않는 경우
    - 데이터 길이가 7인 경우에 8번째 비트를 패리티 비트로 이용
  - 자료 길이
    - 하나의 데이터를 전송하는데 필요한 데이터 길이(비트 수)
    - 보통 7과 8 비트

# ATMega128A의 USART 포트

- ATMega128A의 직렬통신 포트
  - 직렬통신포트 USART(Universal Synchronous and Asynchronous Receive and Transmitter) 2개 내장
    - USART0
    - USART1
  - 완전 이중방식(Full-Duplex)
  - 동기 및 비동기 전송 가능
  - 멀티 프로세서 통신 모드로 동작 가능
  - 높은 정밀도의 보레이트 발생기 내장
  - 인터럽트
    - 송신 완료(TX Complete)
    - 송신 데이터 레지스터 준비완료(TX Data Register Empty)
    - 수신완료(RX Complete)

# ATMega128A의 USART 포트

- ATMega128 USART 데이터 프레임 포맷
  - 최소 7비트 최대 13비트로 구성
    - (1 비트의 스타트 비트) + (5,6,7,8,9 비트의 데이터 비트) + (0, 1 비트의 패리티비트) + (1,2 비트의 스탑비트) 프레임



USART 통신의 데이터 프레임

# ATMega128A의 USART 포트

- ATMega128A USART 데이터 프레임 포맷
  - 스타트 비트
    - 1비트로 이루어 졌으며 항상 0레벨이다. 송신시에 자동적으로 생성
  - 데이터 비트
    - 5, 6, 7, 8, 9비트가 가능
  - 패리티 비트
    - 패리티를 사용하지 않을 수도 있고, 사용하는 경우 홀수 혹은 짝수 패리티 1비트를 사용
  - 스톱 비트
    - 1, 2개의 비트가 가능하며 항상 1레벨, 송신시에 자동적으로 생성

# ATMega128A의 USART 포트

- ATMega128 USART 레지스터
  - UDRn(Usart i/o Data Register n)
    - USART I/O 데이터 레지스터 (UDR0, UDR1)
  - UCSRnA(Usart Control and Status Register n A)
    - USART 제어 및 상태 레지스터 A
  - UCSRnB(Usart Control and Status Register n B)
    - USART 제어 및 상태 레지스터 B
  - UCSRnC(Usart Control and Status Register n C)
    - USART 제어 및 상태 레지스터 C
  - UBRRnH/L (USART BAUD RATE REGISTER)
    - USART baud Rate 레지스터



# ATMega128A의 USART 포트

- UDRn(Usart i/o Data Register n)
  - USART I/O 데이터 레지스터 (UDR0, UDR1)
  - USARTn 모듈의 송수신 데이터 버퍼의 기능을 수행하는 8비트 레지스터(n= 0, 1)
    - 송신 데이터를 UDRn에 write하면, 송신 데이터 버퍼 TXB에 저장
    - 수신 데이터를 DRn에서 읽으면 수신 데이터 버퍼 RXB에 수신되어 있는 값이 읽힘

7	6	5	4	3	2	1	0
RXB7	RXB6	RXB5	RXB4	RXB3	RXB2	RXB1	RXB0
TXB7	TXB6	TXB5	TXB4	TXB3	TXB2	TXB1	TXB0

# ATMega128A의 USART 포트

- UCSRnA(Usart Control and Status Register n A)
  - USART 제어 및 상태 레지스터 A
  - USARTn모듈의 송수신 동작을 제어하거나 송수신 상태를 저장하는 기능을 수행하는 8비트 레지스터
  - 비트 7 : RXCn (USARTn Receiver Complete)
    - 수신버퍼의 상태 플래그
    - 수신버퍼에 수신문자가 있으면 "1"로 세트
    - 수신 버퍼가 비어있는 상태라면 "0"으로 클리어

7	6	5	4	3	2	1	0
RXCn	TXCn	UDREN	FEn	DORn	PEn	U2Xn	MPCMn

# ATMega128A의 USART 포트

- UCSRnA(Usart Control and Status Register n A)
  - 비트 6 : TxCn (USARTn Transmit Complete)
    - 송신버퍼의 상태 플래그
    - 송신 시프트 레지스터에 있는 송신 데이터가 모두 송신되고 UDRn의 송신 버퍼에 아직 새로운 데이터가 저장되지 않은 상태이면 "1"로 세트
  - 비트 5 : UDREn (USARTn Data Register Empty)
    - 새로운 송신 데이터를 받기 위한 상태 플래그
    - UDRn의 송신 버퍼에 새로운 송신 데이터를 받을 준비가 되어 있으면 "1"로 세트
  - 비트 4 : FEn (Frame Error)
    - 수신 프레임 에러 상태 플래그
    - UDRn의 수신 버퍼에 현재 저장되어있는 데이터를 수신하는 동안에 프레임 에러가 발생했음을 나타냄

# ATMega128A의 USART 포트

- UCSRnA(Usart Control and Status Register n A)
  - 비트 3 : DORn (Data Overrun Error)
    - 수신동작 오버런 에러 상태 플래그
    - 수신동작에서 오버런에러가 발생했음을 나타내는 상태 플래그
  - 비트 2 : UPEn (USARTn Parity Error)
    - UDR의 수신버퍼에 현재 저장되어 있는 데이터를 수신하는 동안에 패리티 에러가 발생하였음을 나타내는 상태 플래그
  - 비트 1 : U2Xn (Double the USARTn Transmission Speed)
    - 비동기 모드에서만 사용가능,
    - 클록의 n분주비를 16에서 8로 1/2만큼 낮추어 전송속도를 2배 높이는 기능
  - 비트 0 : MPCMn (USARTn Multi-Processor Communication Mode)
    - USARTn을 멀티프로세서 통신모드 설정

# ATMega128A의 USART 포트

- UCSRnB(Usart Control and Status Register n B)
  - USART 제어 및 상태 레지스터 B
  - USART 모듈의 송수신 동작 제어/송수신 상태 저장
  - USART0, USART1 포트의 송수신 동작제어, 전송 데이터의 9번째 비트값 저장
  - 비트 7 : RXCIEn (USARTn RX Complete Interrupt Enable)
    - 수신완료 인터럽트를 개별적으로 enable
    - 이 비트를 "1"로 설정하고 SREG레지스터의 I비트가 "1"이고, UCSRnA 레지스터의 RXCn비트가 "1"로 설정되어 있으면 수신완료 인터럽트가 발생

7	6	5	4	3	2	1	0
RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n

# ATMega128A의 USART 포트

- UCSRnB(Usart Control and Status Register n B)
  - 비트 6 : TXCIEn (USARTn TX Complete Interrupt Enable)
    - 송신완료 인터럽트를 개별적으로 enable
    - 이 비트를 "1"로 설정하고 SREG레지스터의 I비트가 "1"이고, UCSRnA 레지스터의 RXCn비트가 "1"로 설정되어 있으면 송신완료 인터럽트가 발생
  - 비트 5 : UDRIEn (USARTn Data Register Empty Interrupt Enable)
    - 송신 데이터 레지스터 준비완료 인터럽트(Data Register Empty)를 개별적으로 Enable
    - "1"로 설정하고 SREG레지스터의 I비트가 "1"이고, UCSRnA 레지스터의 UDREn비트가 "1"로 되면, USARTn Data Register Empty 인터럽트가 발생
  - 비트 4 : RXENn (USARTN Receiver Enable)
    - USARTn 모듈의 수신부가 동작하도록 enable
    - RXDn 핀이 병렬 I/O포트가 아니라 직렬 데이터 수신단자로 동작하도록 설정

# ATMega128A의 USART 포트

- UCSRnB(Usart Control and Status Register n B)
  - 비트 3 : TXENn (USARTn Transmitter Enable)
    - USARTn 모듈의 송신부가 동작하도록 enable
    - TXDn 핀이 병렬 I/O포트가 아니라 직렬 데이터 송신단자로 동작하도록 설정
  - 비트 2 : UCSZn2 (USARTn Character Size)
    - UCSRnC레지스터의 UCSZn1~0비트와 함께 전송문자의 데이터 비트수를 설정
  - 비트 1 : RXB8n (USARTn Receiver Data 8Bit)
    - 수신문자가 9비트로 설정된 경우에 수신된 문자의 9번째 비트를 저장
  - 비트 0 : TXB8n (USARTn Transmit Data 8Bit)
    - 송신문자가 9비트로 설정된 경우에 송신된 문자의 9번째 비트를 저장

# ATMega128A의 USART 포트

- UCSRnC(Usart Control and Status Register n C)
  - USART 제어 및 상태 레지스터 C
  - USARTn 모듈의 송수신 동작을 제어하거나 송수신 상태를 저장
- 비트 6 : UMSELn(USARTn Mode Select)
  - USART 모드 설정
  - "1"이면 USARTn 모듈을 동기 전송모드로 설정하고, "0"이면 비동기 전송모드로 설정

7	6	5	4	3	2	1	0
-	UMSELn	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn



# ATMega128A의 USART 포트

- UCSRnC(Usart Control and Status Register n C)
  - 비트 5,4 : UPMn1,0 (USARTn Parity Mode)
    - 패리티 모드 설정
    - UPMn1비트를 "1"로 설정하면 패리티를 발생
    - 오류가 발생하면 UCSRnA 레지스터의 PE플래그가 "1"로 세트
  - 페리티모드 설정표

UPMn1	UPMn0	Parity모드
0	0	Disable
0	1	예약
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

# ATMega128A의 USART 포트

- UCSRnC(Usart Control and Status Register n C)
  - 비트 2,1 : UCSZn1,0(USARTn Character Size)
    - UCSRnB 레지스터의 UCSZn2 비트와 함께 전송문자의 데이터 비트 수를 설정
  - UCSZn에 의한 character Size 설정표

UCSZn2	UCSZn1	UCSZn0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	예약
1	0	1	예약
1	1	0	예약
1	1	1	9-bit

# ATMega128A의 USART 포트

- UCSRnC(Usart Control and Status Register n C)
  - 비트 0 : UCPOLn (Usart Clock POLarity n)
    - 동기 전송 모드의 슬레이브 동작에서만 유효
    - '1'로 설정하면 송신 데이터는 클럭의 하강에지에서 새로운 XCKn 값이 출력되고, 수신 문자는 XCKn의 상승에지에서 얻어짐
    - '0'로 설정하면 반대

# ATMega128A의 USART 포트

- UBRRnH/L (USART BAUD RATE REGISTER)
  - USART baud Rate 레지스터
  - USARTn 모듈의 송수신 속도를 설정
  - 16비트중에서 12비트만 사용
  - 비트 11~0 : UBRRn11~0
    - 12비트를 이용하여 USARTn의 Baud Rate을 결정
    - UBRRnH의 4비트와 UBRRnL의 8비트가 조합을 이룸.

15	14	13	12	11	10	9	8
-	-	-	-	UBRRn 11	UBRRn 10	UBRRn9	UBRRn8
7	6	5	4	3	2	1	0
UBRRn7	UBRRn6	UBRRn5	UBRRn4	UBRRn3	UBRRn2	UBRRn1	UBRRn0

# ATMega128A의 USART 포트

- UBRRnH/L (USART BAUD RATE REGISTER)
  - UBRR에 의한 Baud Rate 설정표

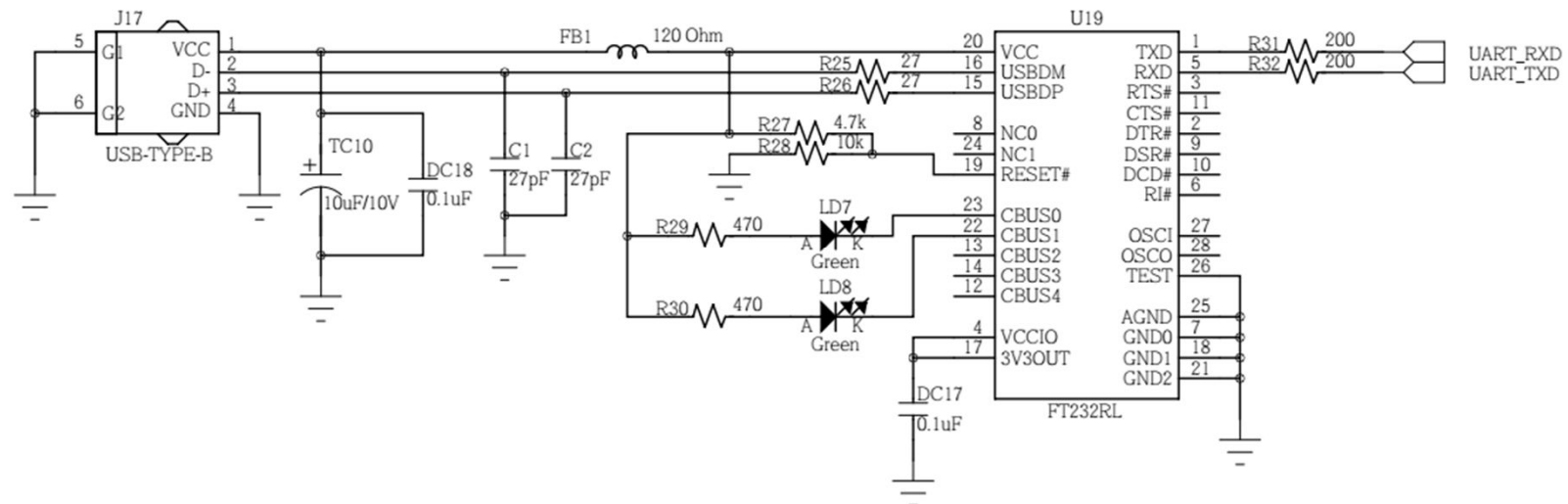
Baud Rate (bps)/14.7456MHz	비동기 일반모드 (U2Xn = 0)		비동기 2배속 모드 (U2Xn=1)	
	UBRRn	Error	UBRRn	Error
2400	383	0.0%	767	0.0%
4800	191	0.0%	383	0.0%
9600	95	0.0%	191	0.0%
14,400	63	0.0%	127	0.0%
19,200	47	0.0%	95	0.0%
28,800	31	0.0%	63	0.0%
38,400	23	0.0%	47	0.0%
57,600	15	0.0%	31	0.0%
76,800	11	0.0%	23	0.0%
115,200	7	0.0%	15	0.0%
230,400	3	0.0%	7	0.0%
250,000	3	-7.8%	6	5.3%
500,000	1	-7.8%	3	-7.8%
1,000,000	0	-7.8%	1	-7.8%

# 실습 1 : UART로 Hello 보내기

- 실습 개요
  - UART를 이용하여 미리 작성된 문장("Hello World")을 PC로 전송하는 실습
  - ATmega128A의 USART 포트를 입력과 출력으로 선언하고 이 포트를 UART 모듈에 연결
  - USB 케이블을 이용하여 PC와 연결
- 실습 목표
  - UART 기능 동작원리 이해
  - ATmega128A의 USART 제어 방법의 습득(관련 레지스터 이해)
  - UART를 통해 PC와 통신하는 방법 습득

# 실습 1 : UART로 Hello 보내기

- 사용 모듈 : UART 모듈 회로



## 실습 1 : UART로 Hello 보내기

- 구동 프로그램 : 사전 지식
  - 미리 정해 주어야 하는 통신 규약을 결정

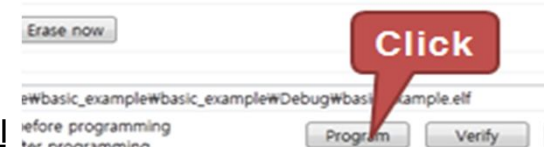
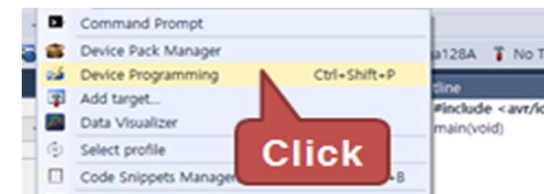
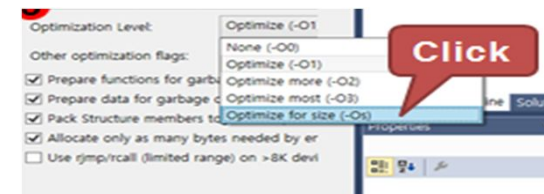
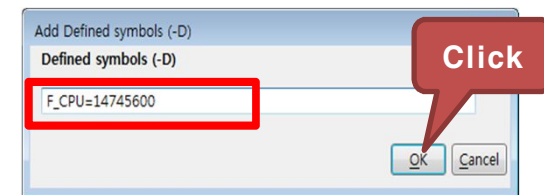
Baud Rate	115200
패리티	No Parity
Stop Bit	1
전송문자 데이터 비트수	8
흐름제어	없음

- UART 제어 레지스터 세팅
  - 비동기 전송 모드
  - 멀티 프로세서 통신 모드
  - UART의 RX와 TX를 Enable
- UCSR0A 레지스터의 플래그를 보면서 데이터를 보낼 수 있는 상태를 기다렸다가 UDR0 레지스터에 데이터를 넣어주면 UART로 데이터가 출력



# 실습 1 : UART로 Hello 보내기

- 예제 프로그램 작성 및 구동
  - Atmel Studio 실행
  - New Project 생성
    - Name : UartHello, Location : D:\AVR\_Example
    - Device Selection : ATmega128A
  - 프로젝트 설정
    - Project 탭에서 "UartHello Properties..." 선택
    - Toolchain -> AVR/GNU C Compiler에서
    - Symbols -> F\_CPU=14745600 추가
    - Optimization -> Optimize for size (-OS) 선택
    - 저장 (Ctrl+S)
  - 소스코드 작성
  - 프로젝트 빌드
    - Build 탭에서 "Build Solution" 클릭
  - 프로그래밍
    - Tool 탭에서 "Device Programming" 클릭
    - AVRISP mkII, ATmega128A 선택 후 "Apply" 클릭
    - 인식 완료되면, Memories 탭 선택, "Program" 클릭



# 실습 1 : UART로 Hello 보내기

- 구동 프로그램
  - main.c 코드 작성

```
#include <avr/io.h>                // AVR 입출력에 대한 헤더 파일

void putch(unsigned char data)
{
    while((UCSR0A & 0x20) == 0);    // 전송준비가 될때까지 대기
    UDR0 = data;                    // 데이터를 UDR0에 쓰면 전송
    UCSR0A |= 0x20;
}

int main(void)
{
    unsigned char text[] = "Hello! World!! \r\n";
    unsigned char i=0;

    DDRE = 0xfe;                    // Rx(입력 0), Tx(출력, 1)
```

# 실습 1 : UART로 Hello 보내기

- 구동 프로그램
  - main.c 코드 작성

```
int main(void) {  
    //...  
    UCSR0A = 0x00;  
    UCSR0B = 0x18;           // Rx, Tx enable  
    UCSR0C = 0x06;           // 비동기 방식, No Parity bit, 1 Stop bit  
  
    UBRR0H = 0x00;  
    UBRR0L = 0x07;           // 115200 bps  
  
    while(text[i]!='\0') // 문자열 데이터가 '\0'인 경우, 문자열 끝임  
    {  
        putchar(text[i++]); // 저장된 문자열 출력  
    }  
}
```

# 실습 1 : UART로 Hello 보내기

- VCP 설치
  - UART 실습을 하려면 PC와 장비를 USB로 연결해야 하며, 장비의 USB-to-Serial 장치의 드라이버를 설치해야 함
  - VCP 설치하기(다운로드) : <http://www.ftdichip.com/Drivers/VCP.htm>

Virtual COM Port Drivers - Windows Internet Explorer

http://www.ftdichip.com/Drivers/VCP.htm

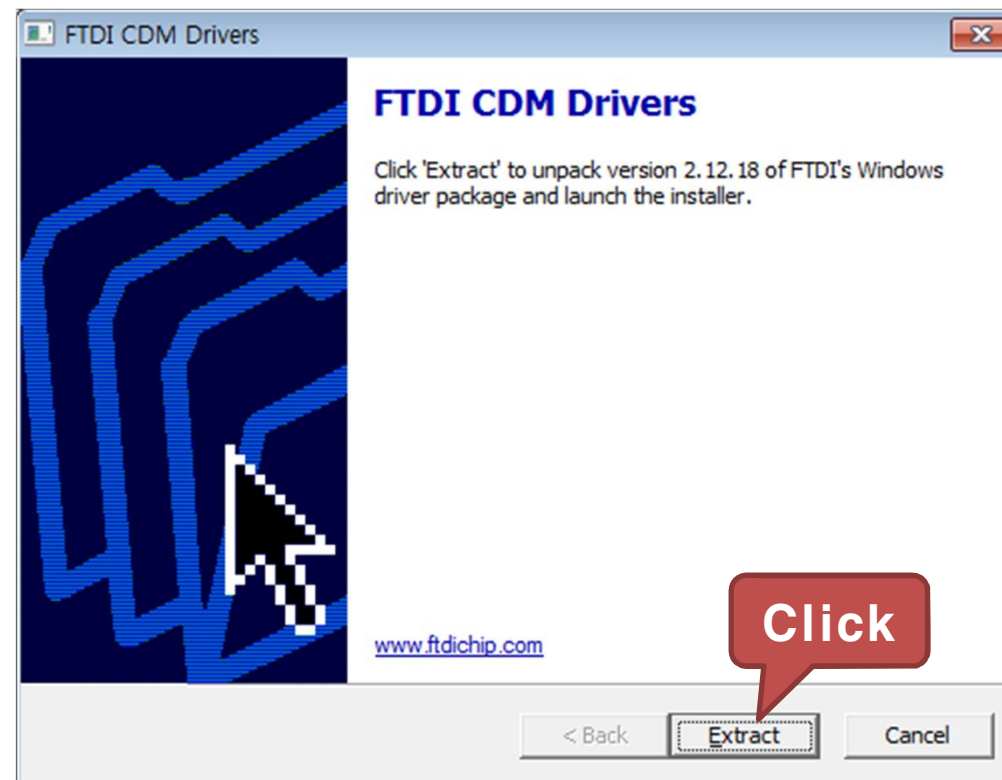
★ 즐겨찾기 Virtual COM Port Drivers

Currently Supported VCP Drivers:

Operating System	Release Date	Processor Architecture							Comments
		x86 (32-bit)	x64 (64-bit)	PPC	ARM	MIPSII	MIPSIV	SH4	
Windows*	2016-06-23	2.12.18	2.12.18	-	-	-	-	-	WHQL Certified. Includes VCP and DXX. Available as <b>setup executable</b> . Please read the Release Notes and Installation Guides.
Linux	2009-05-14	1.5.0	1.5.0	-	-	-	-	-	All FTDI devices now supported in Ubuntu 11.10, kernel 3.0.0-19. Refer to TN-101 if you need a custom VCP VID/PID in Linux.
Mac OS X 10.3 to 10.8	2012-08-10	2.2.18	2.2.18	2.2.18	-	-	-	-	Refer to TN-105 if you need a custom VCP VID/PID in MAC OS

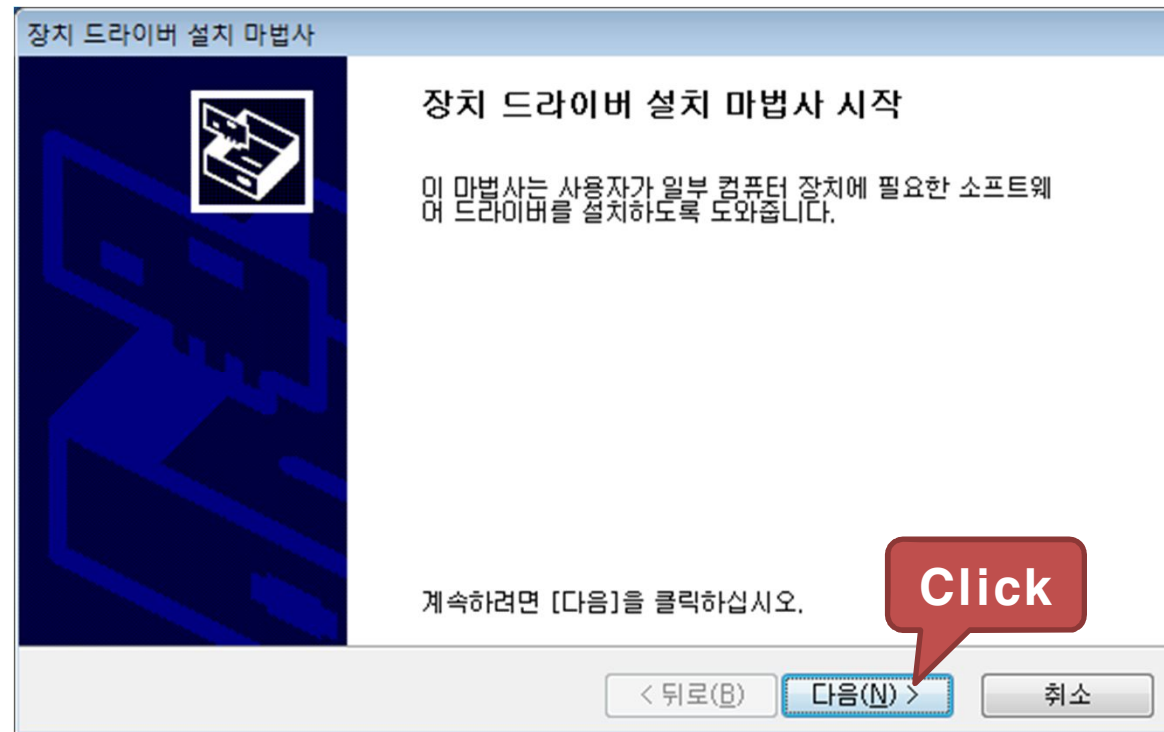
## 실습 1 : UART로 Hello 보내기

- VCP 설치
  - CDM21218\_Setup.exe 파일을 실행



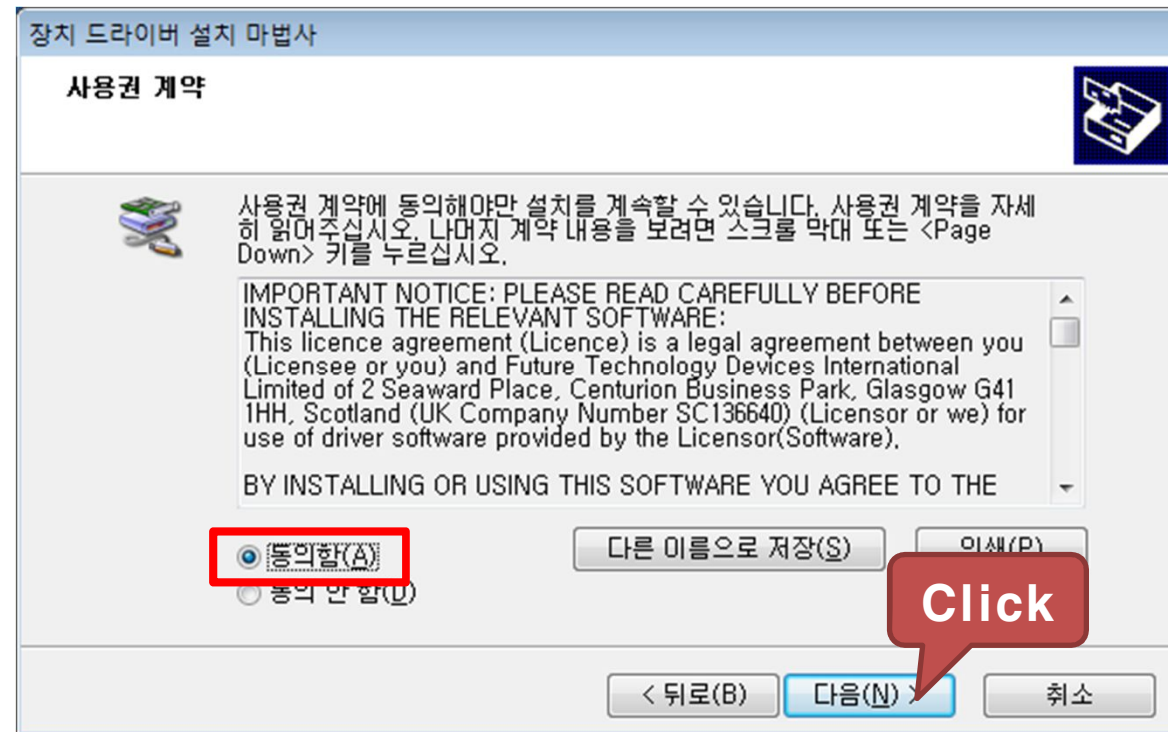
## 실습 1 : UART로 Hello 보내기

- VCP 설치



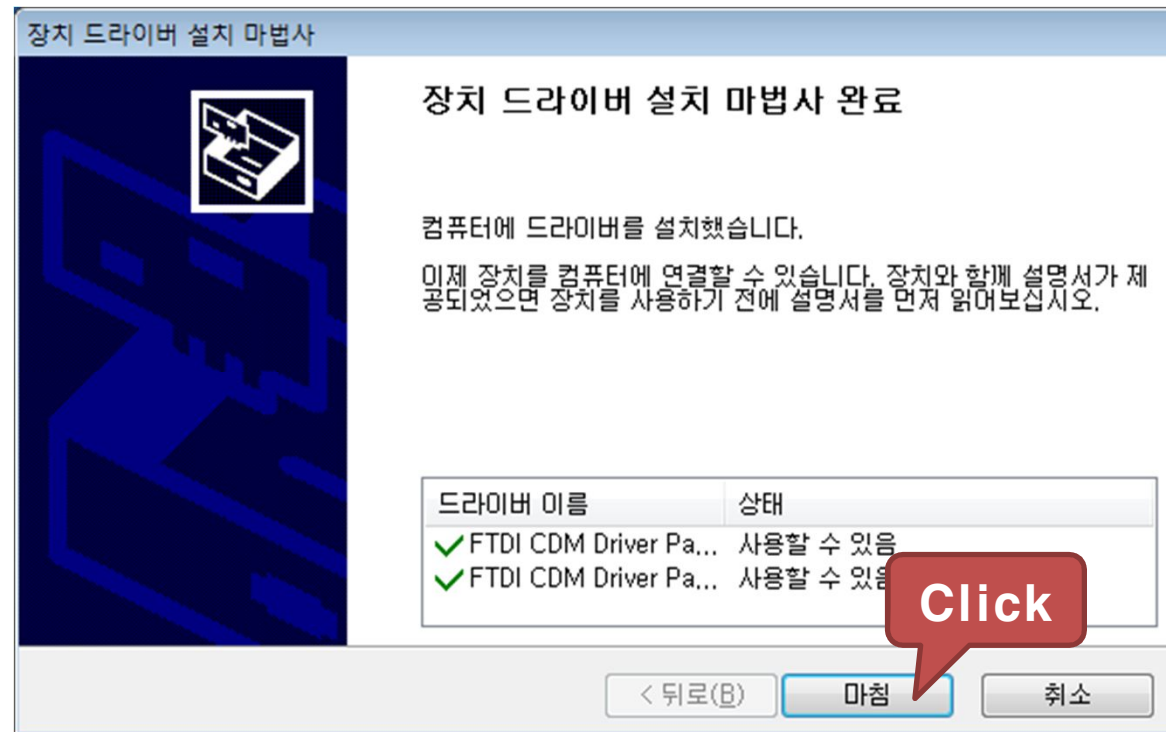
# 실습 1 : UART로 Hello 보내기

- VCP 설치



# 실습 1 : UART로 Hello 보내기

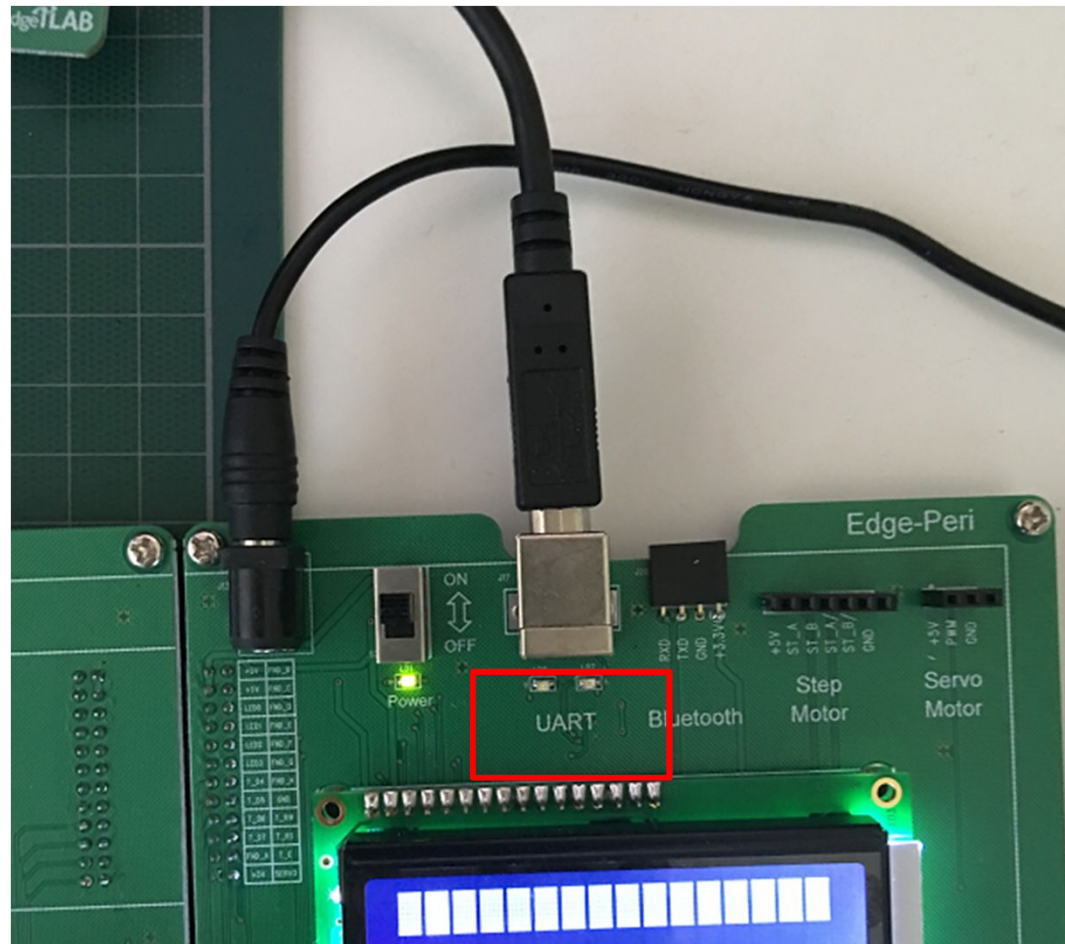
- VCP 설치





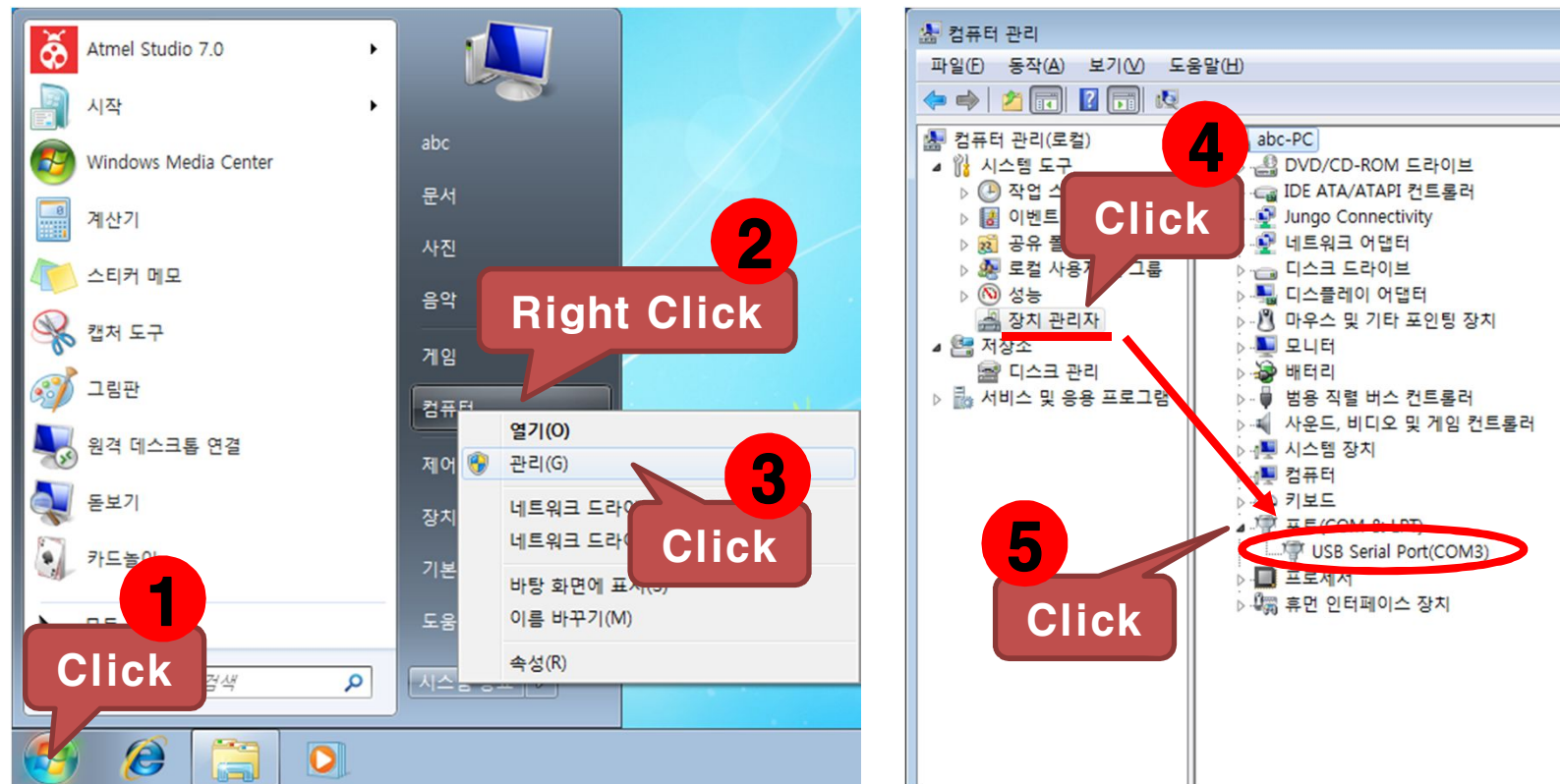
## 실습 1 : UART로 Hello 보내기

- Edge Peri 보드의 UART 커넥터에 USB 케이블을 꽂은 뒤 PC와 연결



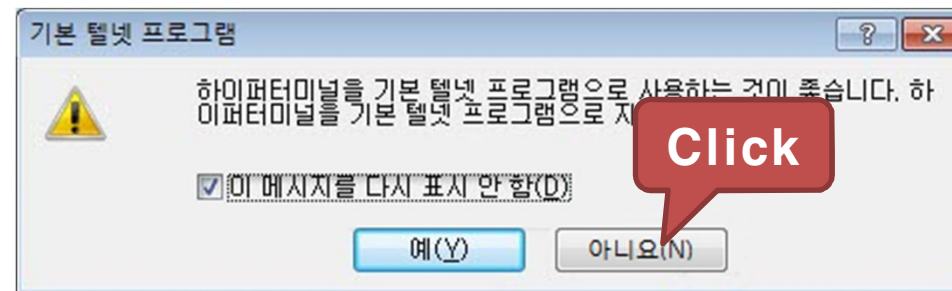
# 실습 1 : UART로 Hello 보내기

- VCP 포트 확인
  - 장치관리자를 실행해서 USB Serial Port 번호를 확인하고 기억해둬م



## 실습 1 : UART로 Hello 보내기

- 하이퍼터미널 실행
  - 예제 확인은 하이퍼터미널을 통해서 하게 되며, 인터넷에서 하이퍼터미널을 임의의 폴더에 다운로드 후, "hypertrm.exe" 파일을 실행
  - 기본 텔넷 프로그램 설정
    - "이 메시지를 다시 표시 안함(D)"를 체크하고 "아니오"를 클릭



# 실습 1 : UART로 Hello 보내기

- 하이퍼터미널 실행
  - 위치 정보 입력창에서 아무 지역번호나 입력하고 "확인"을 클릭
  - 전화 및 모뎀창에서는 "확인"을 클릭

위치 정보

전화나 모뎀을 연결하기 전에 현재 위치에 대한 다음과 같은 정보가 필요합니다.

국가/지역(W)

대한민국

지역 번호(C)

042

통신 회사 번호를 지정할 필요가 있으면 입력하십시오(R).

외부 회선을 사용할 때 누를 번호(Q)

전화 방식:

☒ 톤 방식(T)    ☐ 펄스 방식(P)

Click

확인 취소

전화 및 모뎀

전화 걸기 규칙

아래 목록은 사용자가 지정한 위치를 표시합니다. 전화 거는 위치를 선택하십시오.

위치(L):

위치	지역 번호
☉ 내 위치	042

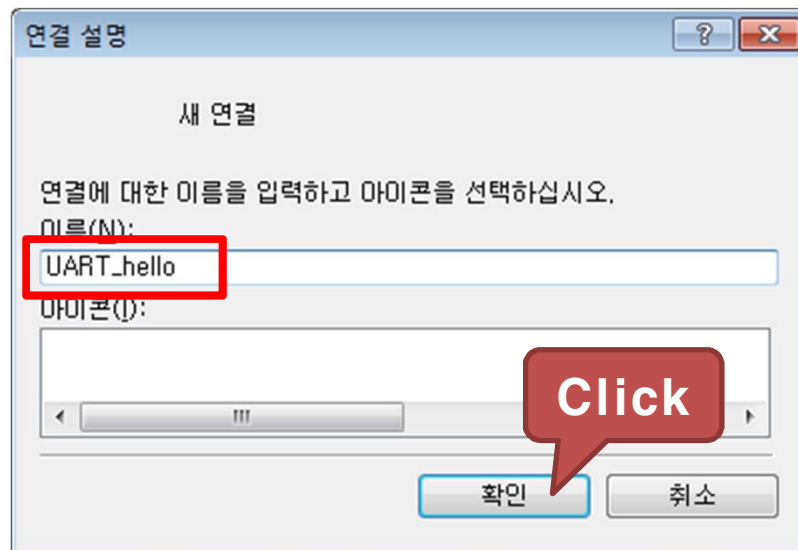
새로 만들기(N)...    편집(E)...    삭제(D)

Click

확인 취소 적용(A)

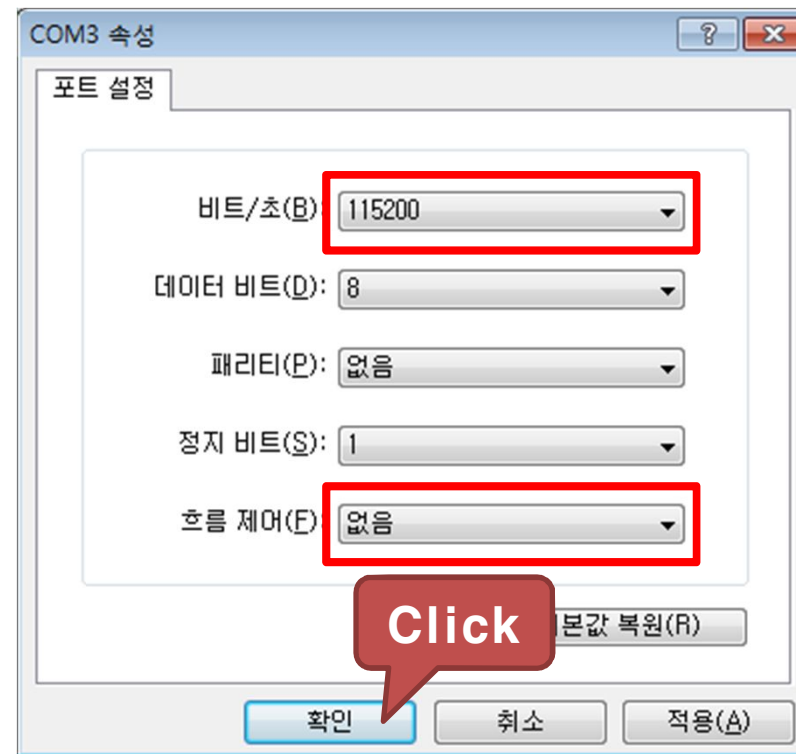
## 실습 1 : UART로 Hello 보내기

- 하이퍼터미널 실행
  - 연결 설명 창의 이름 란에 "UART\_hello" 라고 확인을 클릭
  - 연결 대상 창의 연결에 사용할 모뎀 번호를 설정하고 확인을 클릭



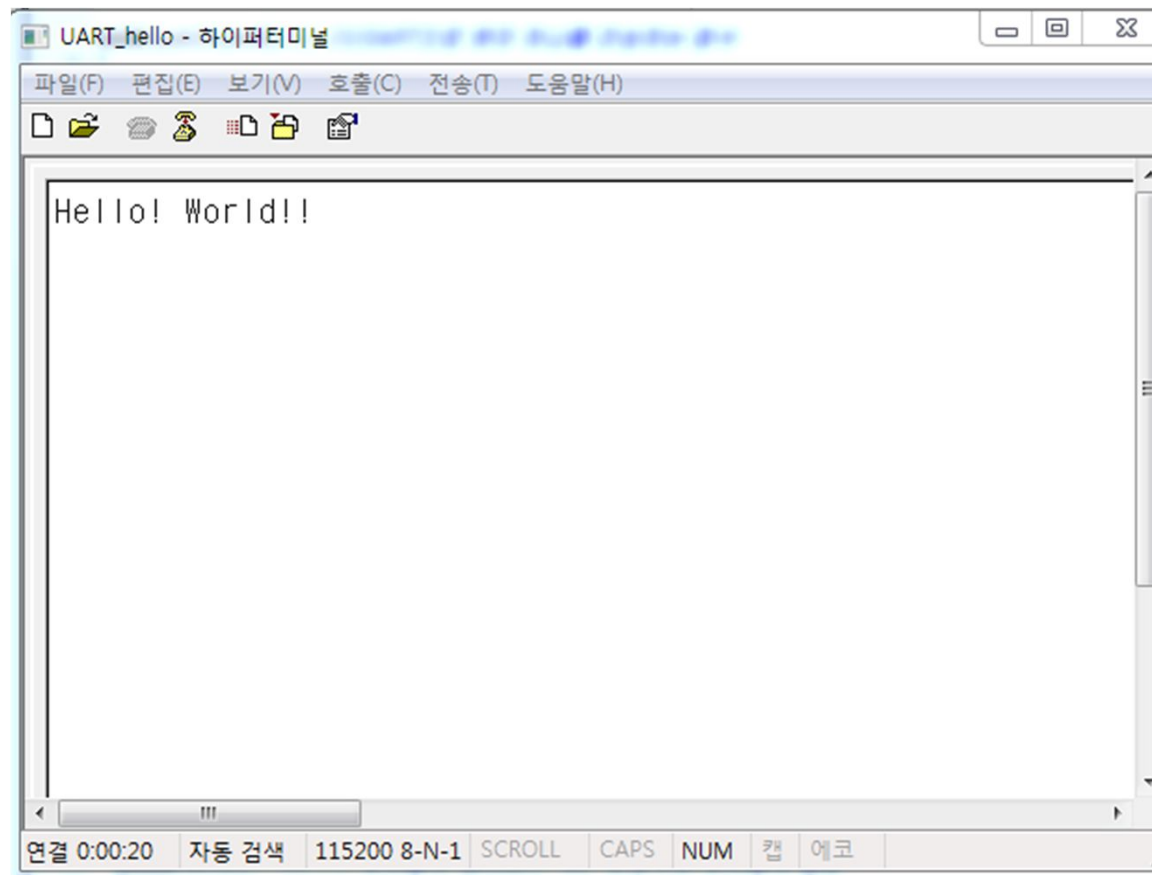
## 실습 1 : UART로 Hello 보내기

- 하이퍼터미널 실행
  - 통신 설정을 그림과 같이하고, 확인을 클릭



## 실습 1 : UART로 Hello 보내기

- 실행 결과 확인
  - 이제 MCU 모듈의 리셋 버튼을 누르면 다음 그림과 같은 결과를 확인



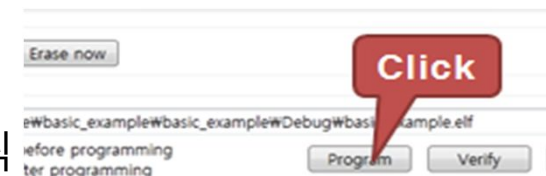
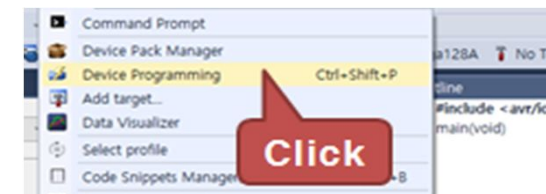
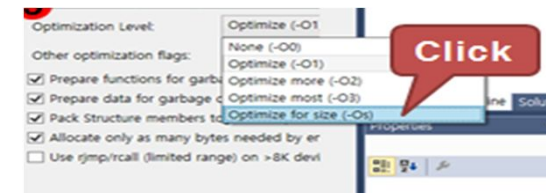
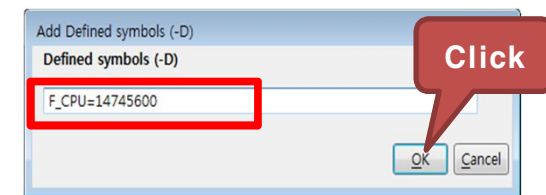
## 응용 1 : UART로 스위치 입력 값 보내기

- 실습 개요
  - UART를 이용하여 스위치 값을 PC로 전송하는 실습
  - ATmega128A의 USART 포트를 입력과 출력으로 선언하고 이 포트를 UART 모듈에 연결
  - USB 케이블을 이용하여 PC와 연결
- 실습 목표
  - UART 기능 동작원리 이해
  - ATmega128A의 USART 제어 방법의 습득(관련 레지스터 이해)
  - UART를 통해 PC와 통신하는 방법 습득



# 응용 1 : UART로 스위치 입력 값 보내기

- 예제 프로그램 작성 및 구동
  - Atmel Studio 실행
  - New Project 생성
    - Name : 08\_UartSwitch\_Application, Location : D:\WAVR\_Example
    - Device Selection : ATmega128A
  - 프로젝트 설정
    - Project 탭에서 "... Properties..." 선택
    - Toolchain -> AVR/GNU C Compiler에서
    - Symbols -> F\_CPU=14745600 추가
    - Optimization -> Optimize for size (-OS) 선택
    - 저장 (Ctrl+S)
  - 소스코드 작성
  - 프로젝트 빌드
    - Build 탭에서 "Build Solution" 클릭
  - 프로그래밍
    - Tool 탭에서 "Device Programming" 클릭
    - AVRISP mkII, ATmega128A 선택 후 "Apply" 클릭
    - 인식 완료되면, Memories 탭 선택, "Program" 클릭



## 응용 1 : UART로 스위치 입력 값 보내기

- 구동 프로그램
  - main.c 코드 작성

```
#include <avr/io.h>           // AVR 입출력에 대한 헤더 파일
#include <avr/interrupt.h>     // AVR 인터럽트에 대한 헤더파일

volatile unsigned char TX_flag = 0;
volatile char TX_data = 0;     // 스위치 전송 값 저장 변수

void putch(unsigned char data)
{
    while((UCSR0A & 0x20) == 0); // 전송준비가 될때까지 대기
    UDR0 = data;                 // 데이터를 UDR0에 쓰면 전송된다
    UCSR0A |= 0x20;
}
```

## 응용 1 : UART로 스위치 입력 값 보내기

- 구동 프로그램
  - main.c 코드 작성

```
// 문자열을 출력하는 함수
void putch_Str(char *str)
{
    unsigned char i=0;
    while(str[i]!='\0')
        putch(str[i++]); //문자열을 출력
}

int main(void)
{
    DDRE = 0x02;          // Rx(입력 0), Tx(출력, 1), SW0 ~ 3 입력
    UCSRA = 0x00;
    UCSRB = 0x18;         // Rx, Tx enable
    UCSR0C = 0x06;        // 비동기 방식, No Parity bit, 1 Stop bit
    UBRR0H = 0x00;
    UBRR0L = 0x07;        // 115200 bps
}
```

## 응용 1 : UART로 스위치 입력 값 보내기

- 구동 프로그램
  - main.c 코드 작성

```
EICRB = 0xFF; // 인터럽트 4, 5, 6, 7을 상승엣지에서 동작하도록 설정
한다.
EIMSK = 0xF0; // 인터럽트 4, 5, 6, 7을 허용
EIFR = 0xF0; // 인터럽트 4, 5, 6, 7 플래그를 클리어
sei();      // 전체 인터럽트를 허용

while(1)
{
    if(TX_flag == 1)
    {
        putch_Str("\n\r Input Switch : ");
        putch(TX_data);
        TX_flag = 0;
    }
}
```

## 응용 1 : UART로 스위치 입력 값 보내기

- 구동 프로그램
  - main.c 코드 작성

```
SIGNAL(INT4_vect)      // 인터럽트 서비스 루틴
{
    cli();              // 전체 인터럽트를 금지
    TX_data = '0';      // 문자 0을 저장
    TX_flag = 1;
    sei();              // 전체 인터럽트를 허용
}

SIGNAL(INT5_vect)      // 인터럽트 서비스 루틴
{
    cli();              // 전체 인터럽트를 금지
    TX_data = '1';      // 문자 1을 저장
    TX_flag = 1;
    sei();              // 전체 인터럽트를 허용
}
```

## 응용 1 : UART로 스위치 입력 값 보내기

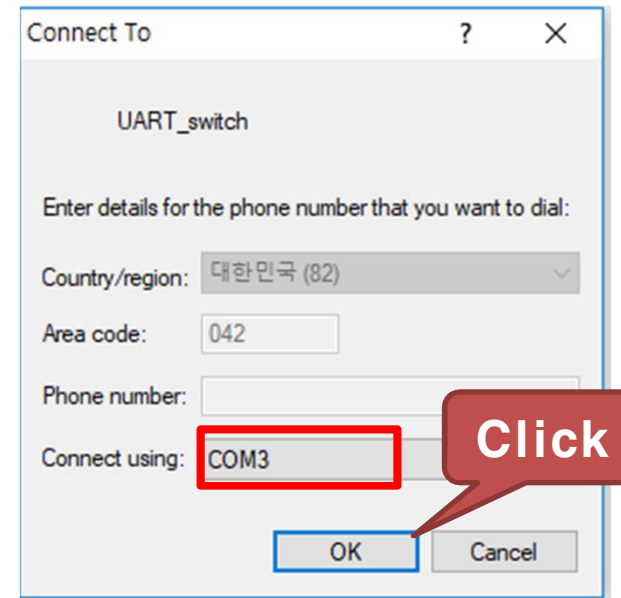
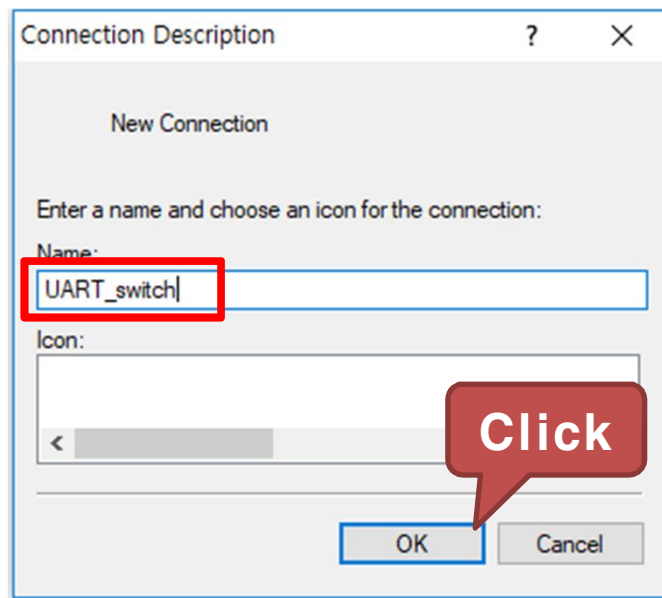
- 구동 프로그램
  - main.c 코드 작성

```
SIGNAL(INT6_vect)      // 인터럽트 서비스 루틴
{
    cli();              // 전체 인터럽트를 금지
    TX_data = '2';      // 문자 2을 저장
    TX_flag = 1;
    sei();              // 전체 인터럽트를 허용
}

SIGNAL(INT7_vect)      // 인터럽트 서비스 루틴
{
    cli();              // 전체 인터럽트를 금지
    TX_data = '3';      // 문자 3을 저장
    TX_flag = 1;
    sei();              // 전체 인터럽트를 허용
}
```

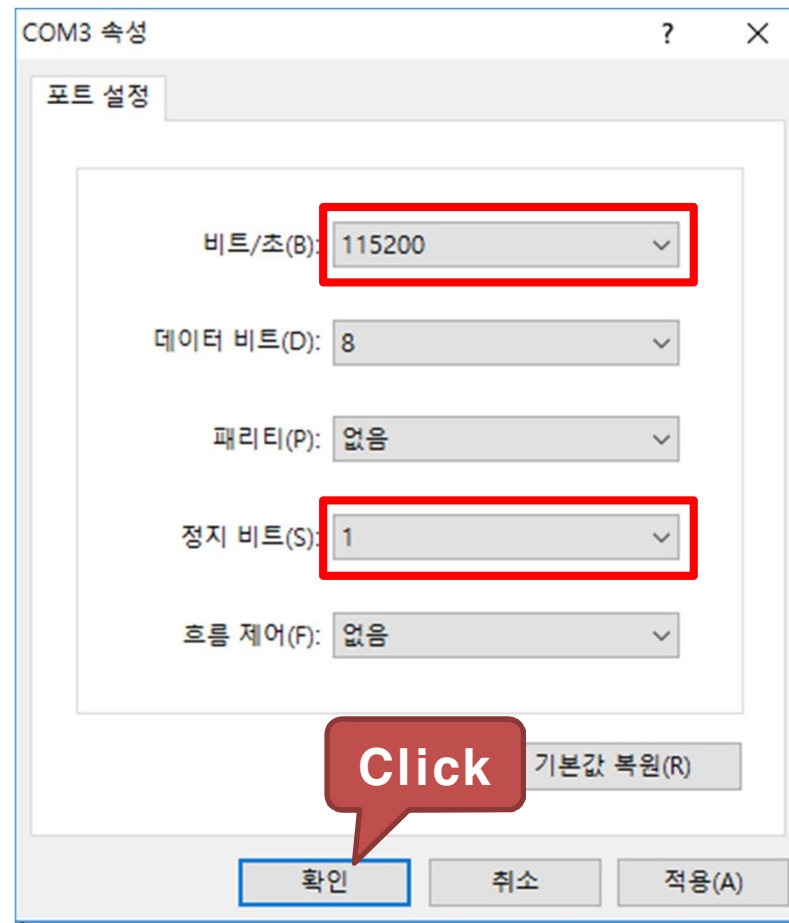
## 응용 1 : UART로 스위치 입력 값 보내기

- 하이퍼터미널 실행
  - 연결 설명 창의 이름 란에 "UART\_hello" 라고 확인을 클릭
  - 연결 대상 창의 연결에 사용할 모뎀 번호를 설정하고 확인을 클릭



## 응용 1 : UART로 스위치 입력 값 보내기

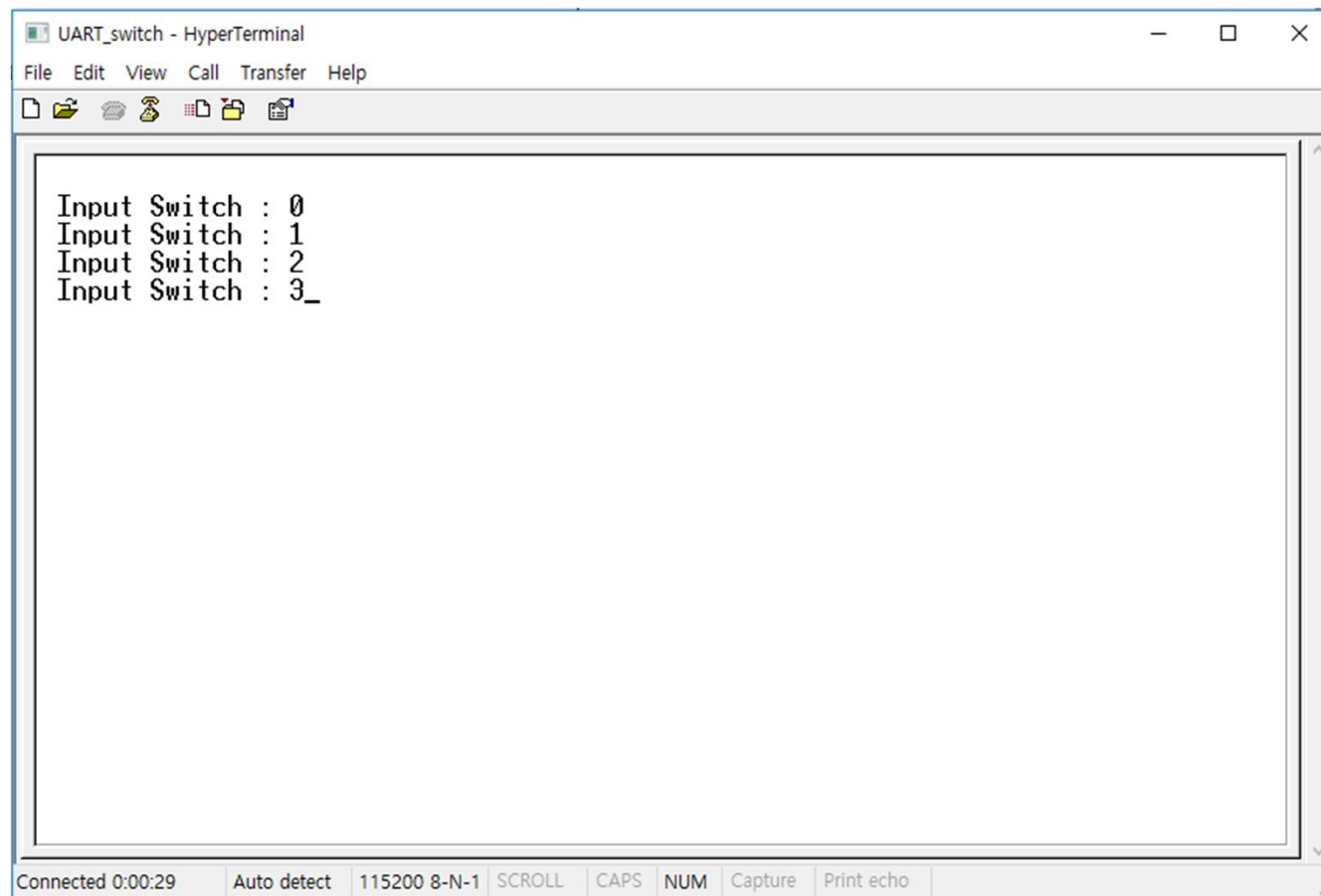
- 하이퍼터미널 실행
  - 통신 설정을 그림과 같이하고, 확인을 클릭





## 응용 1 : UART로 스위치 입력 값 보내기

- 실행 결과 확인
  - MCU 모듈의 리셋 버튼을 누르고, 버튼을 누르면 다음 그림과 같은 결과를 확인



```
UART_switch - HyperTerminal
File Edit View Call Transfer Help

Input Switch : 0
Input Switch : 1
Input Switch : 2
Input Switch : 3_

Connected 0:00:29 Auto detect 115200 8-N-1 SCROLL CAPS NUM Capture Print echo
```

## 실습 2 : UART로 PC와 데이터 주고받기

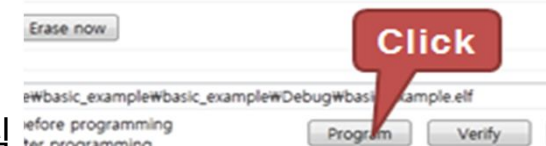
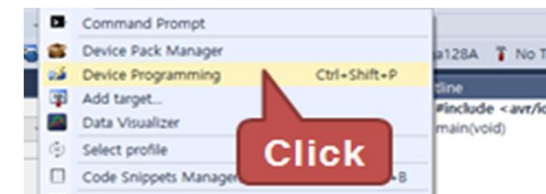
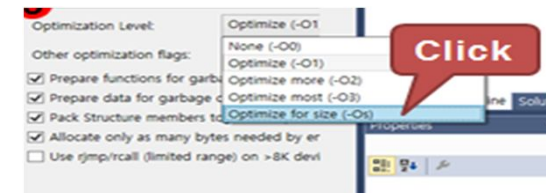
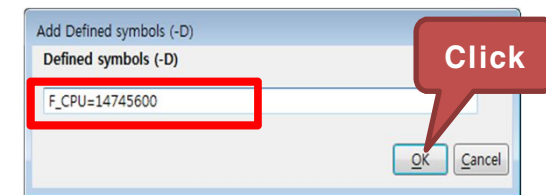
- 실습 개요
  - PC로부터 전송되는 문자열을 받아 다시 PC로 되돌려 전송하도록 함
  - 설정은 앞의 예제와 유사
  - 프로그램의 작성된 문장을 PC 화면에 뿌려주는 역할과 키보드를 통해 입력 받은 ASCII값을 화면상에 보여주는 기능을 함
- 실습 목표
  - UART 기능 동작원리 이해
  - ATmega128A의 USART 제어 방법의 습득(관련 레지스터 이해)
  - UART를 통해 PC와 통신하는 방법 습득

## 실습 2 : UART로 PC와 데이터 주고받기

- 구동 프로그램 : 사전 지식
  - ATmega128A의 USART 포트를 통해서 PC와 UART 통신을 연결하고, PC로 받은 문자열을 그대로 되돌려 전송
  - 설정 : 기본적인 UART 설정은 이전 예제와 동일하게 설정
  - 데이터 수신
    - 모든 설정이 끝나면 UCSR0A 레지스터의 플래그를 보면서 PC로부터 데이터가 도착했는지 살펴보고 있다가 데이터가 도착하면 UDR0 레지스터로부터 데이터를 가져오면 됨
  - 데이터 송신
    - UCSR0A 레지스터의 플래그를 보면서 데이터를 보낼 수 있는 상태를 기다렸다가 UDR0 레지스터에 데이터를 넣어주면 UART로 데이터가 출력

## 실습 2 : UART로 PC와 데이터 주고받기

- 예제 프로그램 작성 및 구동
  - Atmel Studio 실행
  - New Project 생성
    - Name : 08\_UartEcho\_Example, Location : D:\AVR\_Example
    - Device Selection : ATmega128A
  - 프로젝트 설정
    - Project 탭에서 "... Properties..." 선택
    - Toolchain -> AVR/GNU C Compiler에서
    - Symbols -> F\_CPU=14745600 추가
    - Optimization -> Optimize for size (-OS) 선택
    - 저장 (Ctrl+S)
  - 소스코드 작성
  - 프로젝트 빌드
    - Build 탭에서 "Build Solution" 클릭
  - 프로그래밍
    - Tool 탭에서 "Device Programming" 클릭
    - AVRISP mkII, ATmega128A 선택 후 "Apply" 클릭
    - 인식 완료되면, Memories 탭 선택, "Program" 클릭



## 실습 2 : UART로 PC와 데이터 주고받기

- 구동 프로그램
  - main.c 코드 작성

```
#include <avr/io.h>                // AVR 입출력에 대한 헤더 파일

void putch(unsigned char data) {
    while((UCSR0A & 0x20) == 0);    // 전송 준비가 될 때까지 대기
    UDR0 = data;                    // 데이터를 UDR0에 쓰면 전송
    UCSR0A |= 0x20;
}

unsigned char getch(){
    unsigned char data;
    while((UCSR0A & 0x80)==0);      // 데이터를 받을 때까지 대기
    data=UDR0;                      // 수신된 데이터는 UDR0에 저장
    UCSR0A |= 0x80;
    return data;                    // 읽어온 문자를 반환
}
```

## 실습 2 : UART로 PC와 데이터 주고받기

- 구동 프로그램
  - main.c 코드 작성

```
int main(void) {
    unsigned char text[] =
        "\r\nWelcome! edgeiLAB\r\n USART 0 Test Program.\r\n";

    unsigned char echo[] = "ECHO >> ";
    unsigned char i = 0;

    DDRE = 0xfe;           // Rx(입력 0), Tx(출력, 1)

    UCSRA = 0x00;
    UCSRB = 0x18;          // Rx, Tx enable
    UCSR0C = 0x06;         // 비동기 방식, No Parity bit, 1 Stop bit

    UBRR0H = 0x00;
    UBRR0L = 0x07;         // 115200 bps
```

## 실습 2 : UART로 PC와 데이터 주고받기

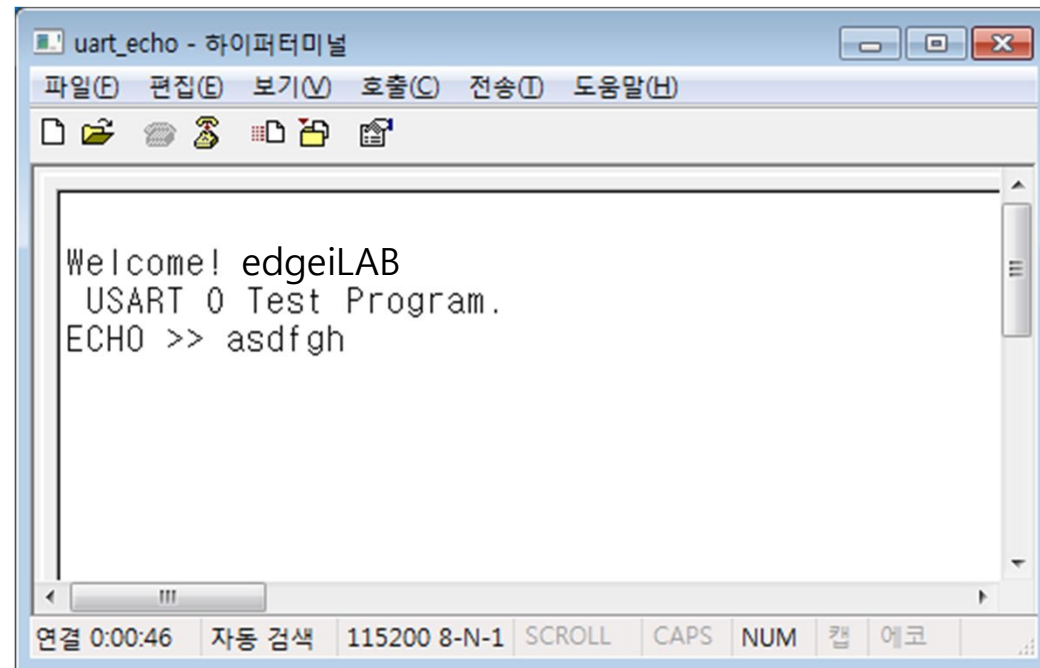
- 구동 프로그램
  - main.c 코드 작성

```
int main(void) {  
    //...  
    while(text[i]!='\0') {// 문자열 데이터가 '\0'인 경우, 문자열 끝임  
        putchar(text[i++]);    // 저장된 text 문자열 출력  
    }  
    i=0;                // 카운터변수 초기화  
    while(echo[i]!='\0') {// 문자열 데이터가 '\0'인 경우, 문자열 끝임  
        putchar(echo[i++]);    // 저장된 echo 문자열을 출력  
    }  
    while(1) {  
        putchar(getch());  
        // getch()로 입력받은 문자를 다시 putchar()로 문자열 출력  
    }  
}
```

## 실습 2 : UART로 PC와 데이터 주고받기

- 실행 결과
  - Edge Peri 보드의 UART 커넥터에 USB 케이블을 꽂은 뒤 PC와 연결
  - 하이퍼터미널을 다음과 같이 설정한 다음 MCU를 리셋하여 문자열을 확인

Baud Rate	115200
패리티	No Parity
Stop Bit	1
데이터 비트수	8
흐름제어	없음



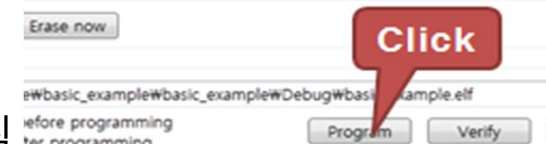
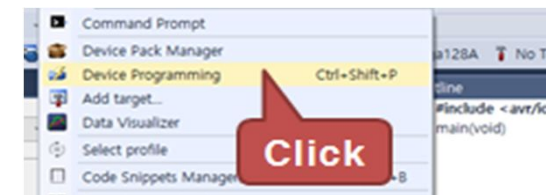
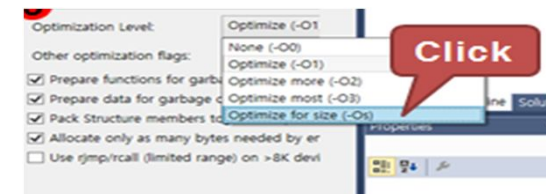


## 응용 2 : UART로 PC에서 FND 점멸시키기

- 실습 개요
  - PC로부터 전송되는 숫자를 받아 FND에 해당 숫자를 보여줌
  - 설정은 앞의 예제와 유사
  
- 실습 목표
  - UART 기능 동작원리 이해
  - ATmega128A의 USART 제어 방법의 습득(관련 레지스터 이해)
  - UART를 통해 PC와 통신하는 방법 습득

## 응용 2 : UART로 PC에서 FND 점멸시키기

- 예제 프로그램 작성 및 구동
  - Atmel Studio 실행
  - New Project 생성
    - Name : 08\_UartFnd\_Application, Location : D:\WAVR\_Example
    - Device Selection : ATmega128A
  - 프로젝트 설정
    - Project 탭에서 "... Properties..." 선택
    - Toolchain -> AVR/GNU C Compiler에서
    - Symbols -> F\_CPU=14745600 추가
    - Optimization -> Optimize for size (-OS) 선택
    - 저장 (Ctrl+S)
  - 소스코드 작성
  - 프로젝트 빌드
    - Build 탭에서 "Build Solution" 클릭
  - 프로그래밍
    - Tool 탭에서 "Device Programming" 클릭
    - AVRISP mkII, ATmega128A 선택 후 "Apply" 클릭
    - 인식 완료되면, Memories 탭 선택, "Program" 클릭



## 응용 2 : UART로 PC에서 FND 점멸시키기

- 구동 프로그램
  - main.c 코드 작성

```
#include <avr/io.h>           // AVR 입출력에 대한 헤더 파일
#include <avr/interrupt.h>     // AVR 인터럽트에 대한 헤더파일

volatile unsigned char TX_flag = 0;
volatile char TX_data = 0;     // 스위치 전송 값 저장 변수

// 7-Segment에 표시할 글자의 입력 데이터를 저장
unsigned char FND_DATA_TBL[]={0x3F,0X06,0X5B,0X4F,0X66,0X6D,0X7C,
                              0X07,0X7F,0X67,0X77,0X7C,0X39,0X5E,0X79,0X71,0X08,0X80};

void putch(unsigned char data)
{
    while((UCSR0A & 0x20) == 0); // 전송준비가 될때까지 대기
    UDR0 = data;                 // 데이터를 UDR0에 쓰면 전송된다
    UCSR0A |= 0x20;
}
```

## 응용 2 : UART로 PC에서 FND 점멸시키기

- 구동 프로그램
  - main.c 코드 작성

```
unsigned char getch(void)
{
    unsigned char data;
    while((UCSR0A & 0x80)==0); // 데이터를 받을때까지 대기
    data=UDR0;                // 수신된 데이터는 UDR0에 저장되어 있다.
    UCSR0A |= 0x80;
    return data;              // 읽어온 문자를 반환한다.
}

int main(void)
{
    unsigned char RX_data = 0;
    DDRE = 0x0e; // Rx(입력 0), Tx(출력, 1)
    DDRA = 0xFF; // 포트A 를 출력포트로 설정한다.
```

## 응용 2 : UART로 PC에서 FND 점멸시키기

- 구동 프로그램
  - main.c 코드 작성

```
UCSR0A = 0x00;
UCSR0B = 0x18; // Rx, Tx enable
UCSR0C = 0x06; // 비동기 방식, No Parity bit, 1 Stop bit
UBRR0H = 0x00;
UBRR0L = 0x07; // 115200 bps
while(1)
{
    RX_data = getch(); //PC로 부터 입력 받은 데이터를 변수 RX_data에
저장
    if((RX_data >= 0x30) && (RX_data <= 0x39))
    {
        PORTA = FND_DATA_TBL[RX_data - 0x30]; // 포트A에 입력된
값의 FND Table 값을 출력한다.
        // 아스키코드 '0'은 0x30 임.
    }
}
```

## 응용 2 : UART로 PC에서 FND 점멸시키기

- 실행 결과
  - 키보드에서 숫자키를 누르면 FND에 해당하는 숫자가 출력됨을 확인

Baud Rate	115200
패리티	No Parity
Stop Bit	1
데이터 비트수	8
흐름제어	없음

