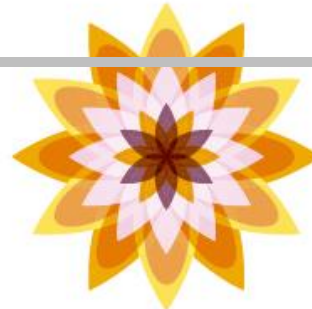
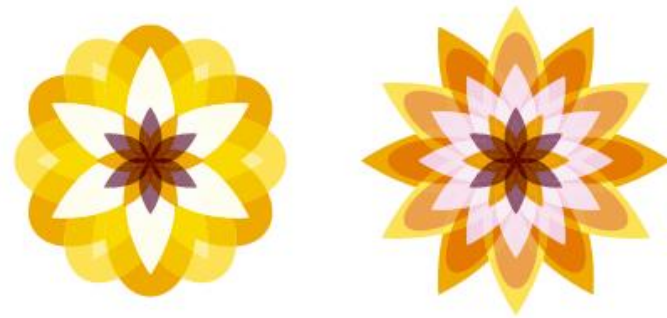


*Chapter 01*

# 시리얼 통신 개요

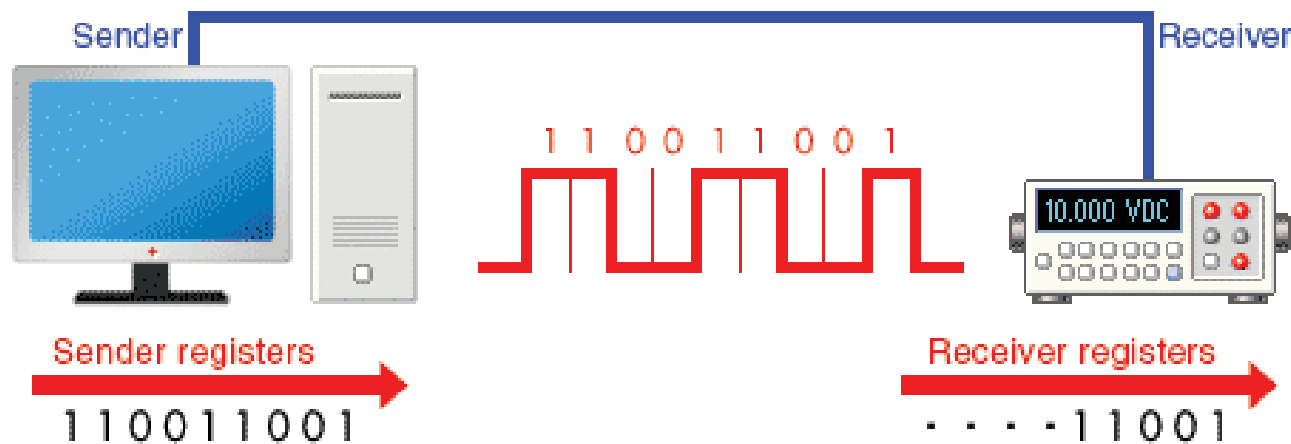


# 목차

1. 시리얼 통신이란 무엇입니까?
2. 시리얼 통신 규격
3. 신호 배치 및 커넥터
4. 연결 방법
5. 비동기 통신 및 동기식 통신

# 1. 시리얼 통신이란 무엇입니까?

- 직렬 통신은 하나 또는 두 개의 전송 라인을 사용하여 데이터를 송수신하는 통신 방법으로, 한 번에 한 비트 씩 데이터를 지속적으로 주고 받습니다.
- 적은 신호선으로 연결이 가능하기 때문에 선재와 중계 장치의 비용이 억제되는 등의 장점이 있습니다.



## 2. 시리얼 통신 규격

- RS-232C / RS-422A / RS-485는 EIA (전자 산업 협회) 통신 표준입니다.
- 이러한 통신 표준 중 RS-232C는 다양한 응용 분야에서 널리 채택되어 왔으며 컴퓨터의 표준 장비이기도 합니다.
- 센서 및 액추에이터에는 이러한 인터페이스가 포함되어 있으며 대부분이 직렬 통신을 통해 제어 할 수 있습니다.
- RS-232C
  - 이 직렬 통신 표준은 널리 사용되며 종종 표준으로 컴퓨터에 장착됩니다."EIA-232"라고도 합니다.
  - 신호선과 커넥터의 목적과 타이밍이 정의되었습니다 (D-sub 25 핀 또는 D- sub 9 핀).
  - 현재 표준은 신호선을 추가하여 개정되었으며 정식으로 "ANSI / EIA-232-E"라고 합니다. 그러나 지금은 일반적으로 "RS-232C"라고도 합니다.

## 2. 시리얼 통신 규격

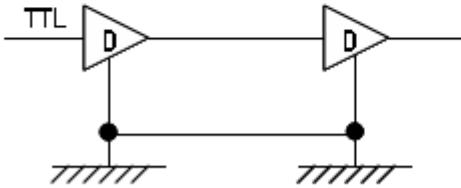
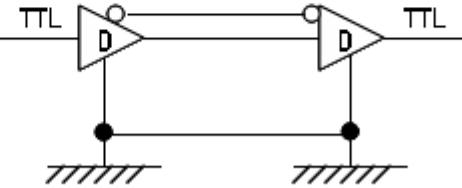
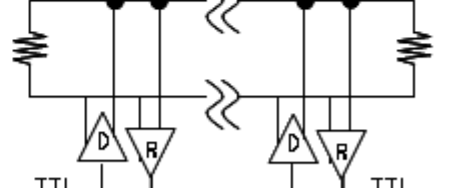
### ■ RS-422A

- 이 표준은 짧은 전송 거리 및 느린 전송 속도와 같은 RS-232C 문제를 수정합니다. "EIA-422A"라고도 합니다.
- 신호선의 목적과 타이밍은 정의되었지만 커넥터는 아닙니다. 많은 호환 제품 주로 D-sub 25 핀 및 D-sub 9 핀 커넥터를 사용합니다.

### ■ RS-485

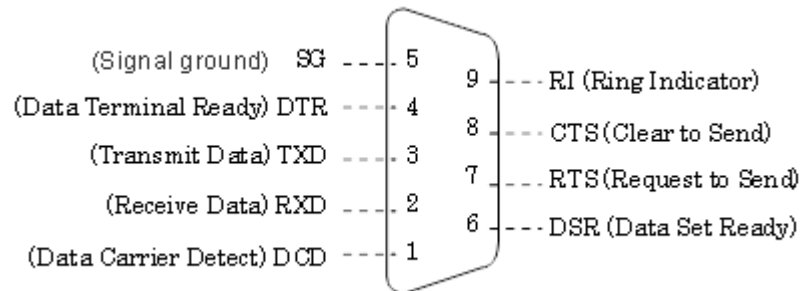
- 이 표준은 RS-422A에서 몇 가지 연결 장치의 문제점을 수정합니다. "EIA-485"라고도 합니다.
- RS-485는 RS-422A와 호환 가능한 표준입니다.
- 신호선의 목적과 타이밍이 정의되지만 커넥터 대부분의 호환 제품은 주로 D-sub 25 핀 및 D-sub 9 핀 커넥터를 채택합니다.

## 2. 시리얼 통신 규격

매개변수	RS-232C	RS-422A	RS-485
전송 모드	단면	멀티포인트, 심플렉스	멀티포인트, 멀티플렉스
최대 접속 대수	1 드라이버, 1 수신기	1개의 드라이버, 10개의 수신기	32 드라이버, 32 수신기
최대 전송 속도	20kbps	10Mbps	10Mbps
최대 케이블 길이	15m	1,200m	1,200m
동작 모드	단일 종단 형 (불균형)	차동 (균형 유형)	차동 (균형 유형)
연결 이미지			
특징	단거리, 전이중, 1:1 연결	장거리, 전이중, 반이중, 1:N 연결	장거리, 전이중, 반이중, N:N 연결

### 3. 신호 배치 및 커넥터

- RS-232C에서는 사용할 커넥터와 신호 할당이 정의되고 표준화되었습니다.
- 그림은 D-sub 9 핀 신호 지정과 신호 라인을 나타냅니다.



- DCD : Data Carrier Detect, 반송파 감지
- RXD : Receive Data, 수신 데이터
- TXD : Transmit Data, 전송 데이터
- DTR : Data Terminal Ready, 데이터 터미널 준비
- SG : Signal Ground, 신호 접지 또는 공통 리턴
- DSR : Data Set Ready, 데이터 세트 준비
- RTS : Request To Send, 송신 요구
- CTS : Clear to Send, 송신 허가
- RI : Ring Indicator, 착신 표시
- CASE FG : Frame Ground, 접지

## 4. 연결방법

- RS-232C에서는 커넥터와 신호 할당이 표준화되어 많은 표준 호환 케이블이 상업적으로 제공됩니다.
- 그러나 장비는 다음 유형으로 제공되며 연결되는 장비에 따라 직선 케이블 또는 크로스 오버 케이블 필요합니다.
- DCE
  - 데이터 통신장비. 이 용어는 모뎀, 프린터 및 플로터와 같이 수동적으로 작동하는 장비를 나타냅니다.
- DTE
  - 데이터 터미널 장비. 이 용어는 컴퓨터와 같이 능동적으로 작동하는 장비를 나타냅니다.



## 5. 비동기 통신 및 동기식 통신

- 직렬 통신에서 데이터는 하나의 신호선을 사용하여 한번에 한비트 씩 전송되므로, 수신측에서 데이터를 정확하게 수신하려면 송신측에서 각 비트를 전송할 속도를 알아야 합니다.
- RS-232C에서 동기식 통신 및 비동기 통신 표준이 정의되었습니다.
- 측정 또는 제어에 사용되는 주변 장치의 경우, 일반적으로 앞서 언급한 전이중 통신과 비동기 통신이 일반적으로 사용됩니다.
- 동기식 통신
  - 이 방법은 다른 장비에서 생성된 클럭 또는 자체 생성된 클럭에 동기된 데이터를 송수신 합니다.
  - 송신은 송신측에서 각 비트에 추가된 동기 신호를 기반으로 수행됩니다.
  - 이는 데이터 전송 효율이 좋지만 전송 절차가 복잡해진다는 단점이 있습니다.

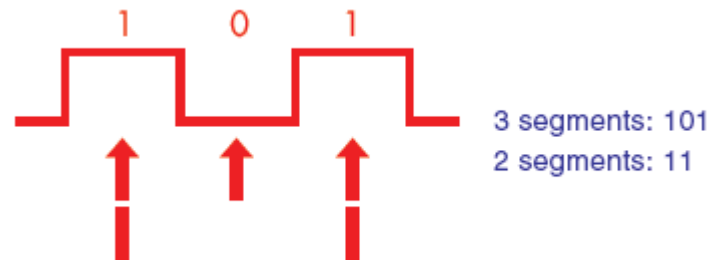
## 5. 비동기 통신 및 동기식 통신

### ■ 비동기 통신

- 이 방법은 각 측의 자체 생성 클록에 동기화 된 데이터를 송수신합니다.
- 전송 속도 설정이 일치하지 않으면 정상적인 통신이 불가능합니다.
- 비동기 통신의 경우 한 번에 한 비트 씩 데이터를 송수신하므로 각 측의 통신 조건이 초기에 일치하지 않으면 정상적인 통신이 불가능합니다.
- 컴퓨터 (컨트롤러) 측 설정을 주변 기기 측에 설정을 일치 시키는 것이 일반적인 설정 방법입니다.

### ■ 전송 속도

- 초당 보낼 비트 수를 지정합니다.
- 단위는 bps (초당 비트 수)이며 300, 600, 1200, 2400, 4800, 9600, 19200 등에서 선택됩니다.
- 설정과 타이밍이 일치하면 데이터 구분 기호가 일치하고 데이터를 정상적으로 송수신 할 수 있습니다.
- 이 때문에 올바른 타이밍을 얻기 위해 각 데이터 항목 (1 바이트)에 시작 비트가 추가됩니다.



## 5. 비동기 통신 및 동기식 통신

### ■ 정지 비트 길이

- 이 값은 데이터의 끝을 나타내는 비트의 길이를 설정합니다.
- 이것은 일반적으로 1 비트, 1.5 비트 또는 2 비트로 선택됩니다.
- 시작 비트 길이는 1 비트로 고정되어 있으므로 이 설정이 필요하지 않습니다.

### ■ 데이터 비트 길이

- 이것은 각 데이터 항목이 구성되는 비트 수를 지정합니다.
- 사용되는 장치에 따라 다르지만 일반적으로 영숫자 및 기호에는 7 비트를 지정하고 1 바이트 이진 데이터에는 8 비트를 지정합니다.

### ■ 패리티 체크 설정

- 데이터에서 오류를 찾는 기능이며 "짝수 패리티 검사 (EVEN)", "홀수 패리티 검사 (ODD)" 또는 "패리티 없음 검사 (없음)" 중에서 선택됩니다.

## 5. 비동기 통신 및 동기식 통신

### ■ 패리티 검사 세부 정보

- 송신 측에서는 EVEN에 대해서도 "1"데이터 비트의 수를, ODD에 대해서는 홀수가되도록 데이터에 "1"또는 "0"의 패리티 비트를 추가합니다.
- 수신 측에서는 " 1 "데이터 비트가 카운트되고 EVEN 일 때도 숫자가 ODD 인 경우 데이터가 올바른 것으로 판단됩니다.
- 예 : 짝수 패리티 검사



## 5. 비동기 통신 및 동기식 통신

### ■ 핸드 셰이크 (흐름 제어)

- 장치간에 데이터를 송수신 할 때 수신 측이 수신 상태가 아닌 경우 데이터를 전송할 때 데이터가 손실 될 수 있으므로 상대방의 상태를 확인하는 것이 통신에서 중요합니다.
- 핸드 셰이크 (흐름 제어)는 통신의 신뢰성을 유지합니다.
- 송신 측에서 "데이터를 송신 중"이라고하는 수신 측으로 신호가 보내지고 수신 측에서 그 신호를 수신하여 신호선에서 데이터를 읽습니다.
- 그러면 송신 측에 응답을 보냅니다. "데이터가 수신되었습니다."
- 즉, 각 측면에서 데이터 송수신을 확인하면서 데이터를 전송할 수 있습니다.

### ■ 소프트웨어 핸드 셰이크 (XON / XOFF 흐름 제어)

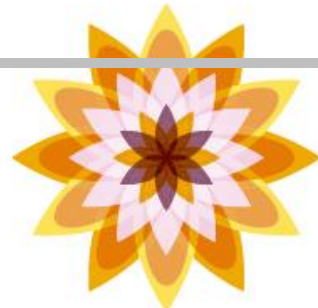
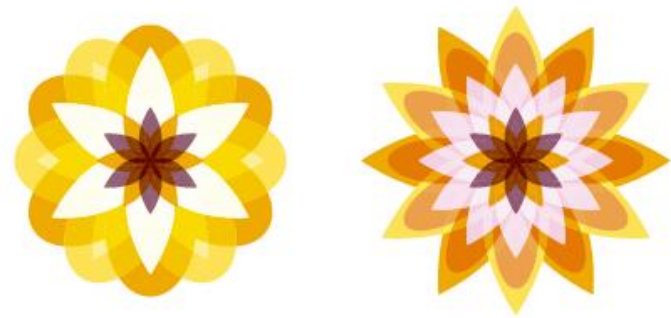
- 이는 "XOFF 코드"가 수신 측에서 수신 버퍼의 남아있는 여유 공간이 적어지면 송신이 일시적으로 중단되도록 요청하기 위해 송신 측으로 전송되는 제어 방법입니다.
- 충분한 여유 공간이 있는 경우, "XON 코드"는 송신 측이 송신을 다시 시작하도록 요청하기 위해 전송됩니다.

### ■ 하드웨어 핸드 셰이크

- 제어 흐름 (RTS 또는 DTR)은 소프트웨어 흐름 제어에서 XON / XOFF 코드를 전송하는 대신 자동으로 켜지거나 꺼집니다.
- RTS 신호와 CTS 신호 또는 DTR 신호와 DSR 신호는 서로 연결되어야 합니다.

*Chapter 02*

# 시리얼 통신 프로그램



# 목차

1. Visual Basic에서 직렬 포트 제어
2. .NET Framework SerialPort (SerialPort 클래스)
3. .NET Framework SerialPort 프로그래밍 포인트
4. Win32 API를 사용하여 직렬 포트 제어

# 1. Visual Basic에서 직렬 포트 제어

---

- Visual Basic에서 직렬 포트를 제어하려면 .NET Framework SerialPort 클래스를 사용하는 메서드와 Win32 API를 사용하는 메서드가 있습니다.
- 여기에서는 상대적으로 간단한 SerialPort 클래스를 사용하여 데이터를 보내고 받는 프로그래밍 예제를 소개합니다.



## 2. .NET Framework SerialPort (SerialPort 클래스)

- COM 포트 클래스 (SerialPort 구성 요소)가 .NET Framework에 Ver. 2.0. Visual Basic 2005 이상에서는 COM 포트를 비교적 간단하게 제어 할 수 있는 .NET Framework SerialPort 구성 요소를 사용할 수 있습니다.
- Visual Basic 6과 같은 이전 버전에서 자주 사용되었던 MSComm 컨트롤을 더 이상 사용할 수 없습니다.
- .NET Framework SerialPort 기능
  - 직렬 포트에 연결하기위한 설정
  - 명령 전송 (RTS 및 CTS와 같은 다양한 직렬 인터페이스 제어 신호 및 상태 입력 제어)
  - 데이터 전송
  - 직렬 연결 및 오류 모니터링 및 처리 중 다양한 이벤트 (제어 신호에 상태 변화가있을 때 및 통신 중 오류가 발생할 때 이벤트가 생성 될 수 있음)
  - 직렬 포트에 연결하기위한 설정

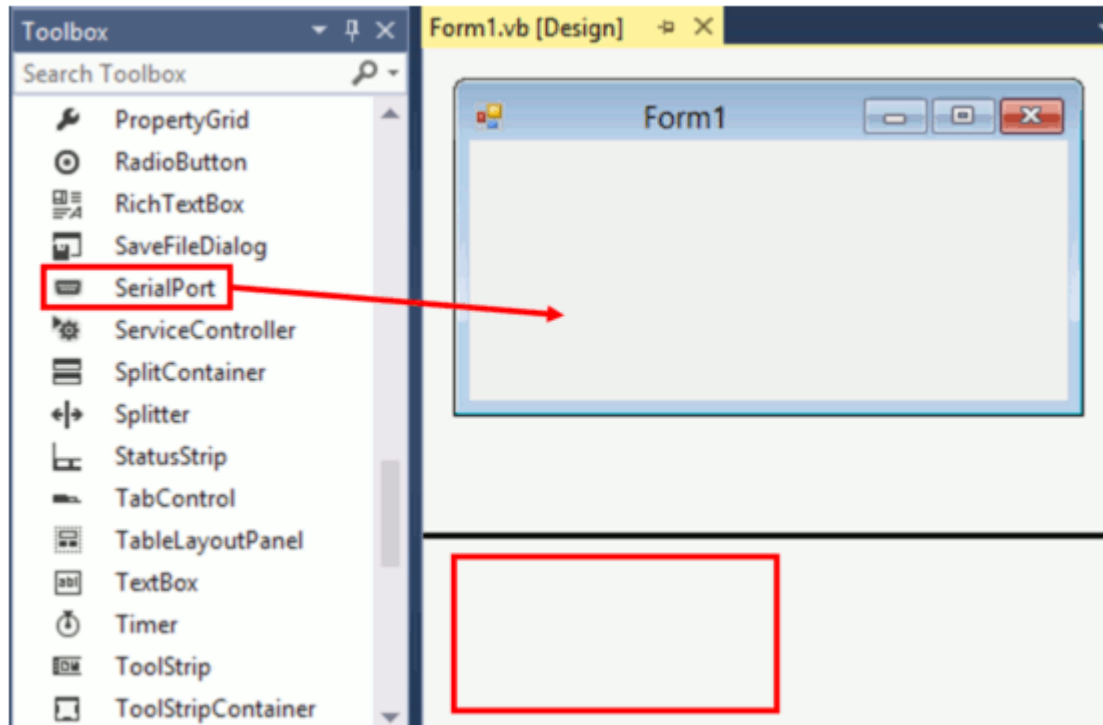
### 3. .NET Framework SerialPort 프로그래밍 포인트

- NewLine (편의상 "구분자 코드"(종결 자, 구분 기호)로 표시됨)를 포함하는 데이터를 쓰고 읽을 수 있습니다.
- 구분 기호 코드를 추가하면 연속 데이터의 구분 기호로 자동 판단됩니다.
- 송신시 "WriteLine"을 사용하면 지정된 구분 기호 코드가 자동으로 데이터 문자열에 추가됩니다.
- 양방향 처리는 인터럽트 (이벤트 구동) 처리로 가능합니다. "DataReceived" 및 "PinChanged" 이벤트를 사용하면 제어 신호선의 변경 및 데이터 수신과 같은 이벤트가 발생할 때 알림을 수신 할 수 있습니다.
- 인터럽트는 이벤트가 발생할 때 발생하므로 필요한 처리를 즉시 수행 할 수 있습니다.

### 3. .NET Framework SerialPort 프로그래밍 포인트

#### ■ SerialPort 구성 요소 붙여 넣기

- Visual Basic "도구 상자"에서 SerialPort 구성 요소를 선택하고 구성 요소를 마우스 왼쪽 단추로 끌어서 끌어다 놓아 폼에 붙여 넣습니다.
- 구성 요소의 붙여 넣기가 끝나면 구성 요소가 양식 아래에 붙여 넣어집니다.



### 3. .NET Framework SerialPort 프로그래밍 포인트

#### ■ 속성 구성

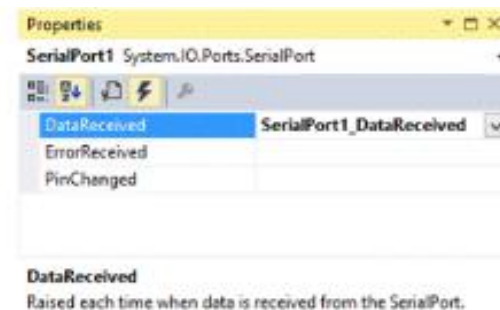
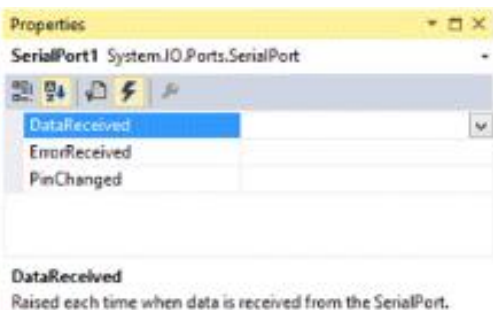
- 구성 요소의 등록 정보 창으로 전환 한 것으로 보이는 구성 요소를 클릭합니다.
- "PortName"속성으로 통신에 사용될 포트 번호를 설정합니다. 초기 값은 "COM1"입니다.
- "BaudRate"속성으로 전송 속도를 설정합니다. 초기 값은 "9600"입니다.
- RTS 사용 여부와 같은 기타 속성을 구성 할 수도 있습니다.

The screenshot shows the Visual Studio IDE with a form named 'Form1' in the design view. In the bottom-left corner of the form, a 'SerialPort1' component is added and highlighted with a red rectangle. To the right, the 'Properties' window is open, displaying the properties for 'SerialPort1' of type 'System.IO.Ports.SerialPort'. The 'BaudRate' and 'PortName' properties are highlighted with red rectangles, showing values of '9600' and 'COM1' respectively. Other visible properties include 'DataBits' (8), 'DiscardNull' (False), 'DtrEnable' (False), 'GenerateMember' (True), 'Handshake' (None), 'Modifiers' (Friend), 'Parity' (None), 'ParityReplace' (63), 'ReadBufferSize' (4096), and 'ReadTimeout' (-1).

(ApplicationSettings)	
(Name)	SerialPort1
BaudRate	9600
DataBits	8
DiscardNull	False
DtrEnable	False
GenerateMember	True
Handshake	None
Modifiers	Friend
Parity	None
ParityReplace	63
PortName	COM1
ReadBufferSize	4096
ReadTimeout	-1

### 3. .NET Framework SerialPort 프로그래밍 포인트

#### ■ 이벤트 설정



- 속성 창의 이벤트 버튼을 클릭하여 구성 요소의 이벤트 목록을 표시합니다.
- 사용할 이벤트를 선택하고 두 번 클릭하면 해당 이벤트 루틴이 추가됩니다.
- 이 위치에서 이벤트가 발생했을 때 처리 코드를 작성하십시오.

#### ■ 이벤트 유형

- DataReceived 이벤트
  - 문자가 수신되고 데이터가 수신 버퍼에 저장되면 발생.

### 3. .NET Framework SerialPort 프로그래밍 포인트

#### ■ 이벤트 설정

##### ▪ ErrorReceived 이벤트

- Frame : 프레임 오류가 감지될 경우
- Overrun : 오버런 오류가 감지될 경우
- RxOver : 버퍼 오버플로가 감지될 경우
- RxParity : 패리티 오류가 감지될 경우
- TxFull : 송신 버퍼가 가득 차서 버퍼에 데이터를 저장할 수 없는 경우

## 4. Win32 API를 사용하여 직렬 포트 제어

- 직렬 포트 프로그래밍의 경우 Win32 API (응용 프로그래밍 인터페이스)를 호출하는 메소드도 있습니다.
- Win32 API는 Windows가 표준으로 제공하는 기능 (함수)입니다. DLL로 제공되며 Windows에서 실행되는 모든 응용 프로그램은 이를 공유하고 사용할 수 있습니다.
- Windows는 약 1000 가지 유형의 API 기능으로 기본 기능을 제공하며 응용 프로그램 소프트웨어는 이 기능을 결합하여 빌드됩니다.
- Visual Basic 및 Visual C에는 많은 명령이 있지만 이는 모든 Windows 기능을 사용할 수 있다는 것을 의미하지 않으므로 필요에 따라 API 함수를 호출하고 사용해야 합니다.

## 4. Win32 API를 사용하여 직렬 포트 제어

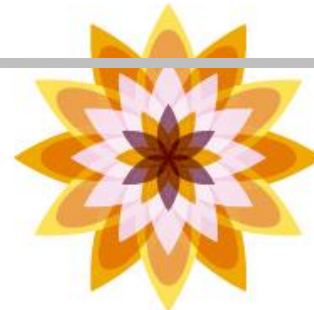
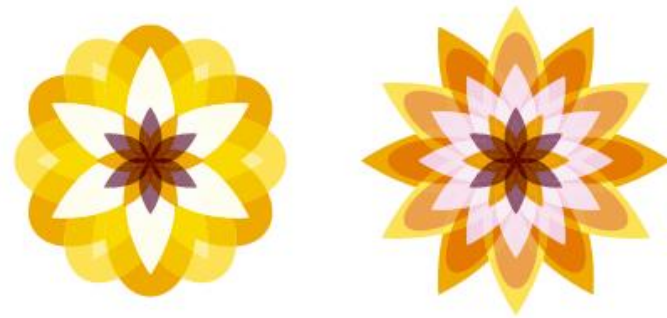
---

- 직렬 포트를 제어하려면 다음 API 함수를 사용하십시오.
- 직렬 포트를 제어하는 데 사용되는 Win32 API
  - 포트 열기 : CreateFile
  - 포트 구성 : SetCommState, SetCommTimeouts
  - 데이터 전송 : WriteFile
  - 데이터 수신 : ReadFile



*Chapter 03*

# 시리얼 통신 프로그램



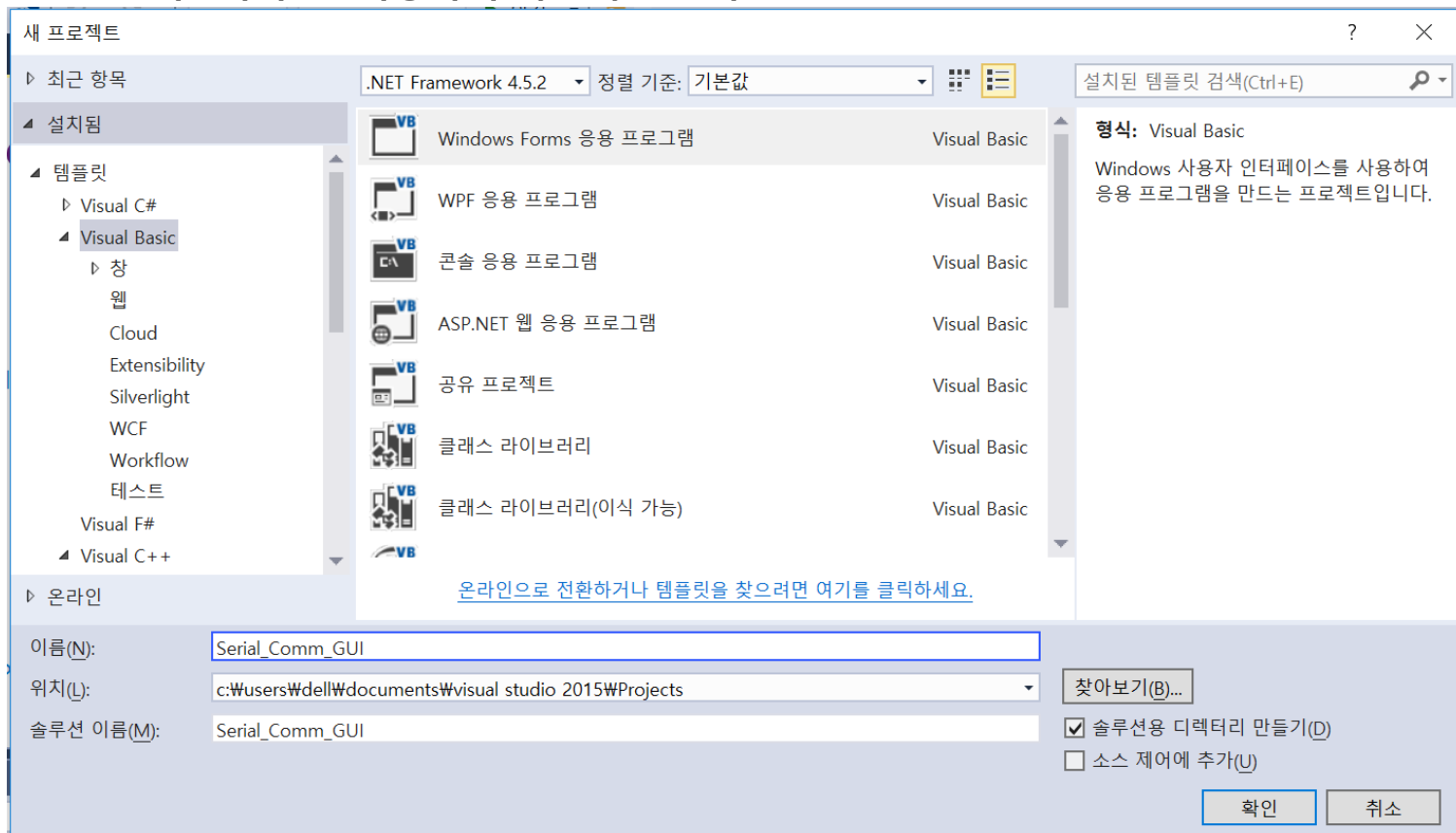
# 목차

1. "시리얼 통신 설정" 그룹박스에 컨트롤 배치
2. 시리얼 통신 컨트롤 추가
3. 이벤트 추가
4. 소스 코드 추가
5. 프로그램 동작확인

# 1. Visual Basic에서 신규 프로젝트 생성

■ Visual Studio를 실행한 후 [파일] - [새로 만들기] - [프로젝트]를 클릭하여 아래의 그림과 같이 나타나는 [새 프로젝트] 창에서 다음과 같이 설정해 줍니다.

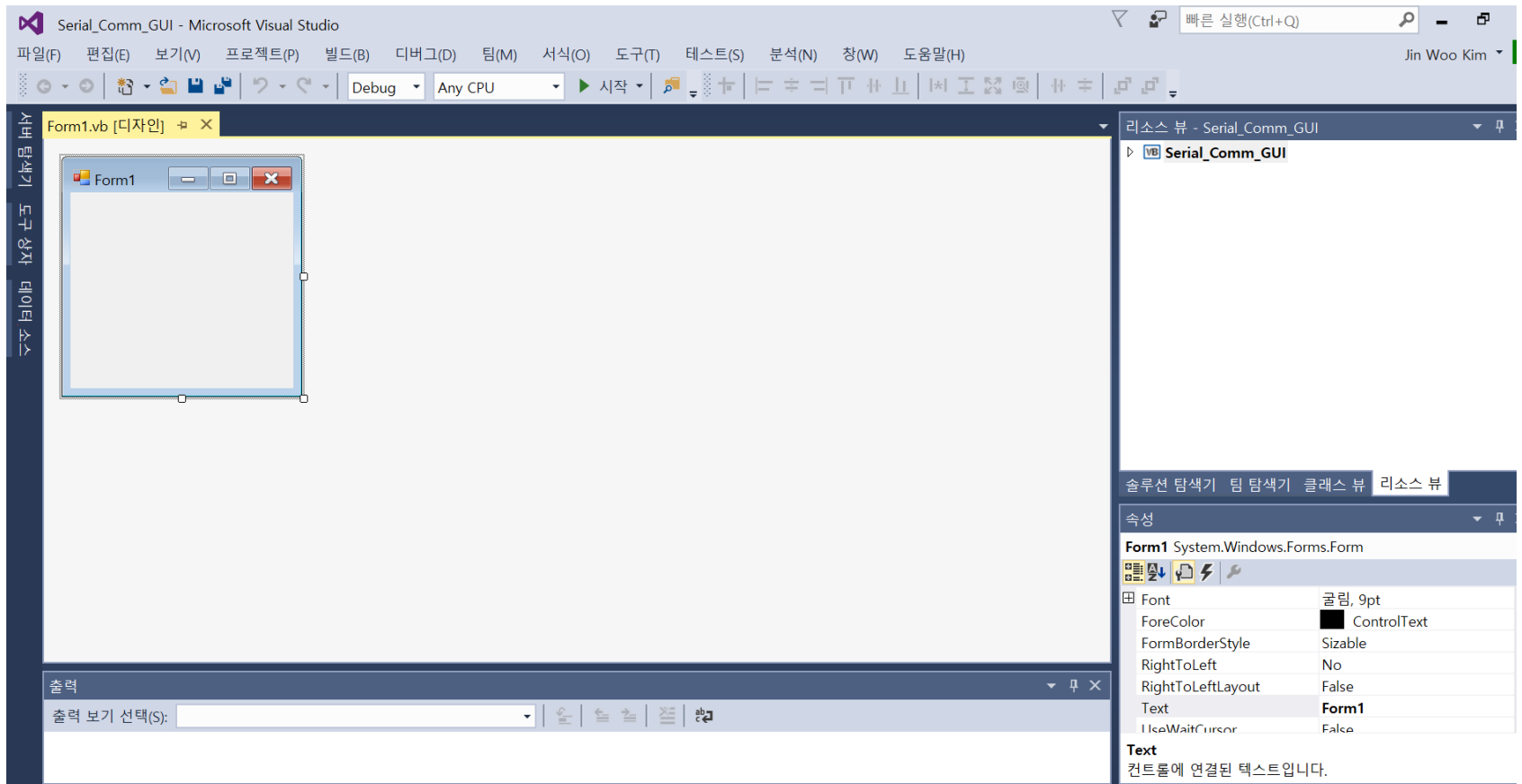
- 먼저 좌측의 "프로젝트 형식"에서 "다른 언어" 카테고리에 있는 "Visual Basic"을 클릭한 후 우측 창의 "템플릿"에 나열되는 항목 중 "Windows Forms 응용 프로그램"을 클릭하고, 새 프로젝트 하단에 있는 "이름"과 "위치"를 적당하게 지정해 줍니다.



# 1. Visual Basic에서 신규 프로젝트 생성

■ 그림 아래의 그림과 같이 Visual Studio의 프로그램 화면이 변경되는 것을 알 수 있습니다.

- 기본적인 Visual Basic의 프로그래밍을 할 수 있는 프로젝트가 생성된 것으로, 빈 Form과 도구상자, 솔루션 탐색기, Form의 속성을 볼 수 있는 속성창 등으로 구성되어 있는 것을 알 수 있습니다.



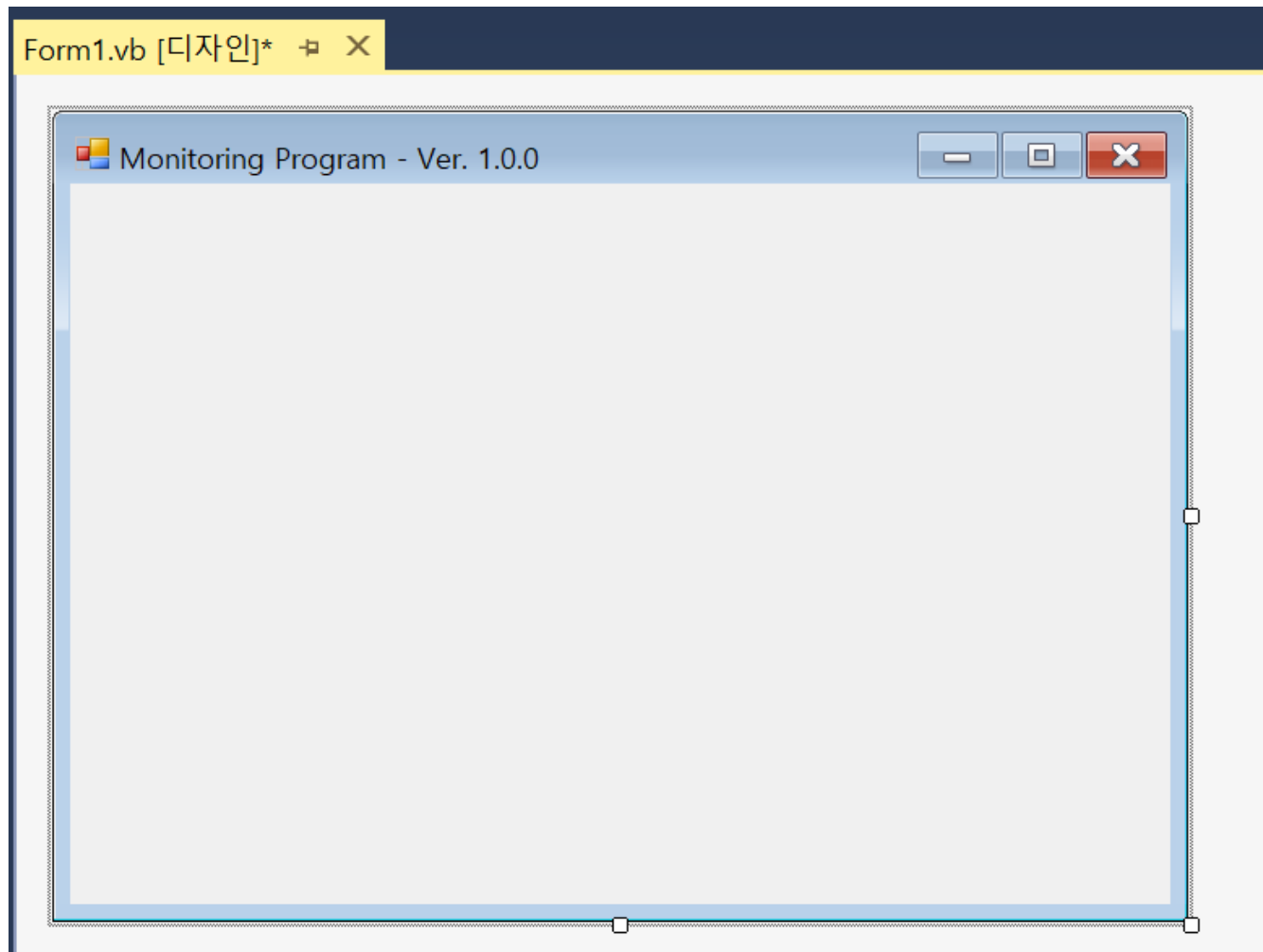
## 2. Form의 속성 변경

- 기본적인 프로젝트 생성이 완료가 되었으면, Form의 크기 등의 각종 속성을 설정해 주도록 합니다.
  - 먼저 아래의 그림과 같이 폼을 클릭해 포커스를 잡은 다음 Visual Studio 2008 프로그램 우측 하단에 나타나는 속성창에서 속성을 변경해 줍니다. Form의 기본적인 속성들은 그대로 둔 다음 아래의 속성들만 변경해줍니다.

(Name) : Form\_Monitoring\_Program\_Main  
SizeGripStyle : Hide  
StartPosition : CenterScreen  
Text : Monitoring Program - Ver. 1.0.0

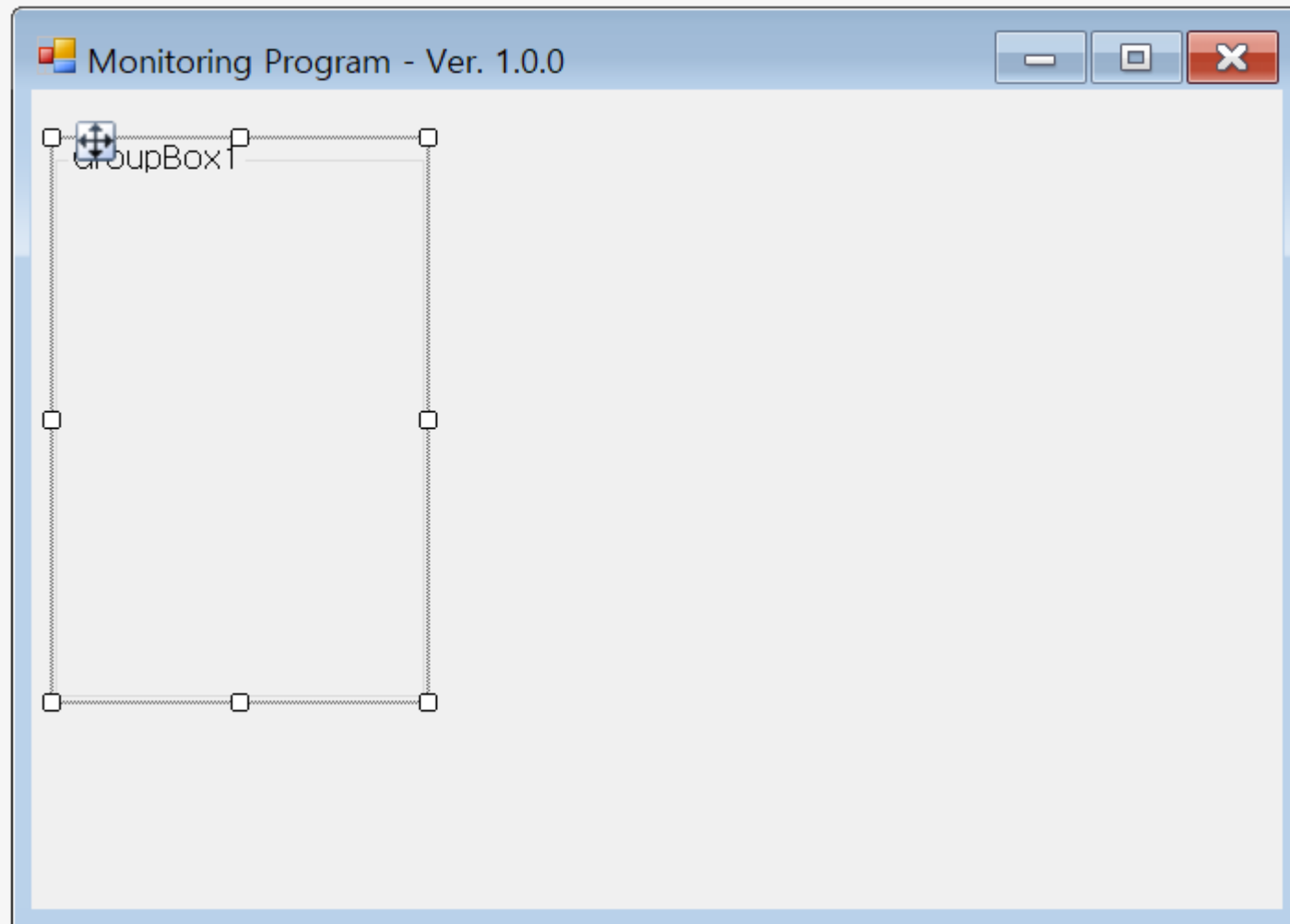
## 2. Form의 속성 변경

- Form의 속성 변경을 완료하면, 아래의 그림과 같이 Form의 크기와 Form 상단의 텍스트가 변경이 완료된 것을 확인할 수 있습니다.



### 3. Form에 Component 배치를 위한 기본 레이아웃

- 다음으로 Component를 배치하기 위하여 Form에 그룹박스를 아래의 그림과 같이 배치하도록 합니다.



### 3. Form에 Component 배치를 위한 기본 레이아웃

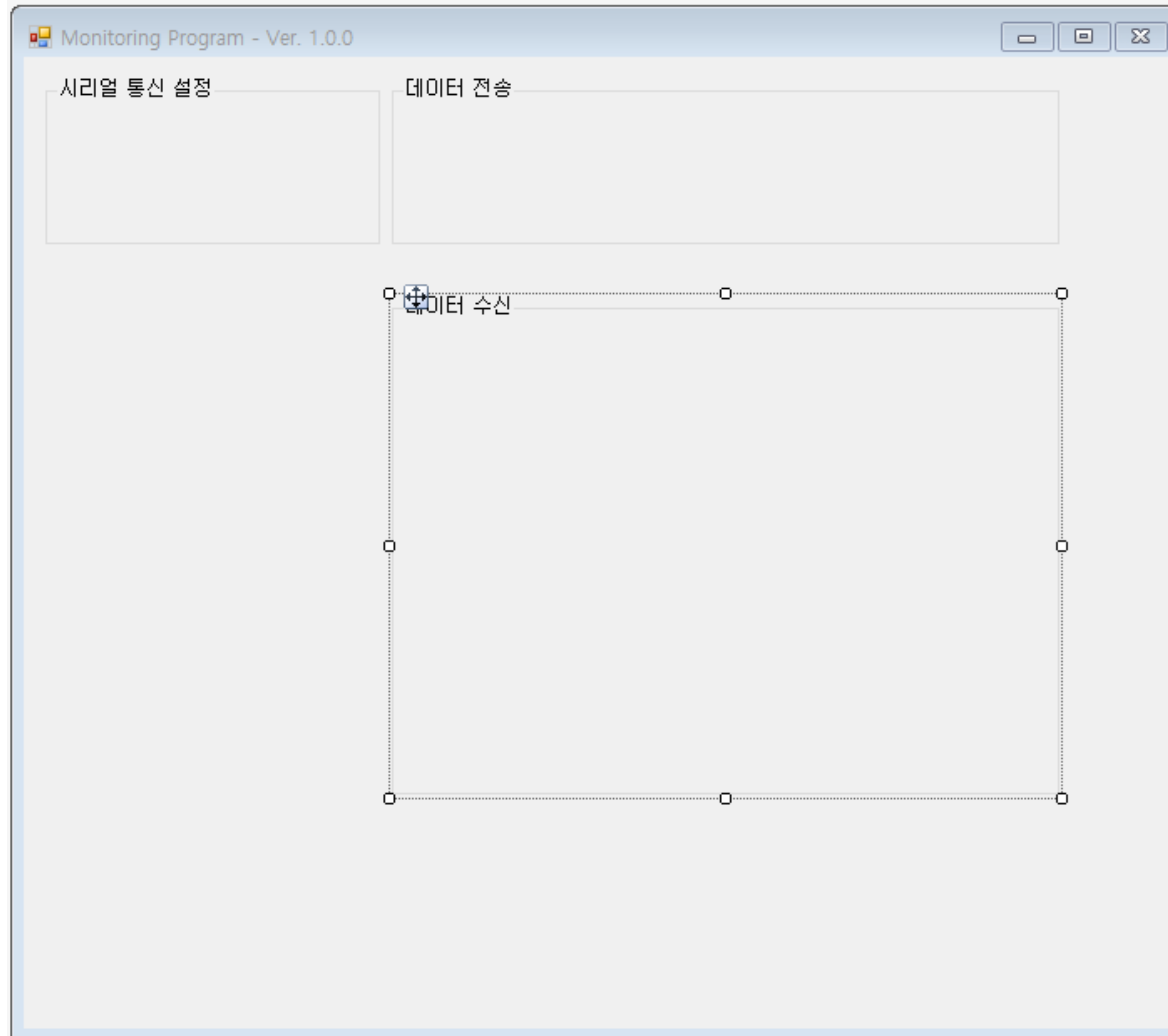
- Form에 배치된 그룹박스는 기본적으로 설정된 속성을 제외하고, 다음과 같은 속성으로 지정합니다.

(Name) : GroupBox\_Serial\_Port\_Setting  
Text : 시리얼 통신 설정



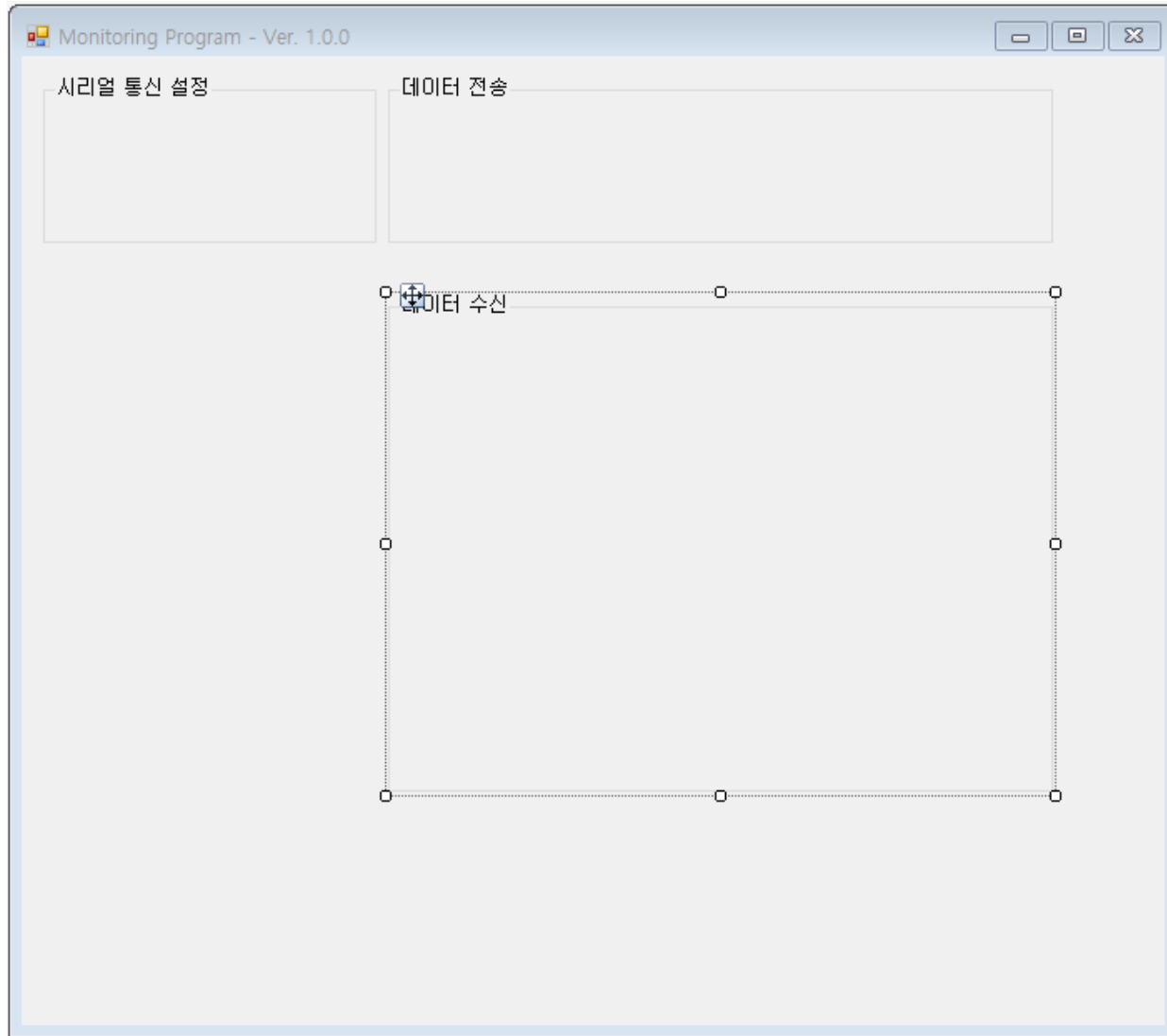
### 3. Form에 Component 배치를 위한 기본 레이아웃

- 그런 다음 컨테이너에서 두 그룹 상자를 추가하고 텍스트를 "데이터 전송"및 "수신 데이터"로 변경합니다.



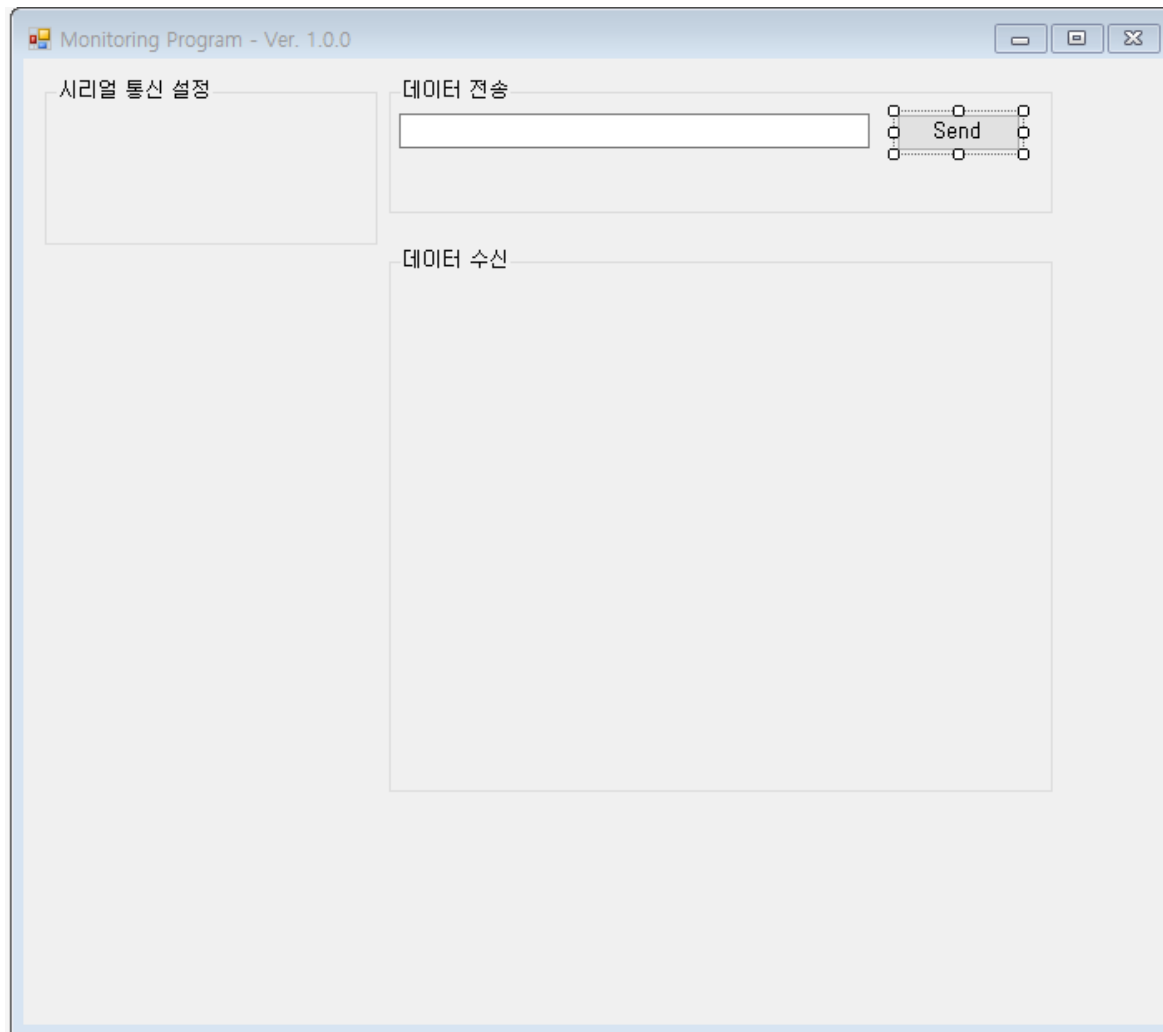
### 3. Form에 Component 배치를 위한 기본 레이아웃

- 데이터 전송 그룹 상자에서 TextBox와 Button을 추가합니다.



### 3. Form에 Component 배치를 위한 기본 레이아웃

- TextBox의 이름을 txtTransmit으로 변경하고 Button를 btnSend로 변경하고 텍스트를 Send로 변경합니다.



### 3. Form에 Component 배치를 위한 기본 레이아웃

- 받은 데이터 그룹 상자에서 RichTextBox를 추가하고 이름을 rtbReceived로 변경합니다.
- 마지막으로 Form에 레이블 컨트롤을 1개를 아래의 그림과 같이 배치하도록 합니다.
- 레이블 컨트롤의 기본적인 속성들은 그대로 사용하고, 아래와 같이 속성을 변경해 줍니다.

(Name) : Label\_Credit

Text : Programmed By Jin-Woo Kim, Date : 2016.08.21

### 3. Form에 Component 배치를 위한 기본 레이아웃

Monitoring Program - Ver. 1.0.0

시리얼 통신 설정

데이터 전송

Send

데이터 수신

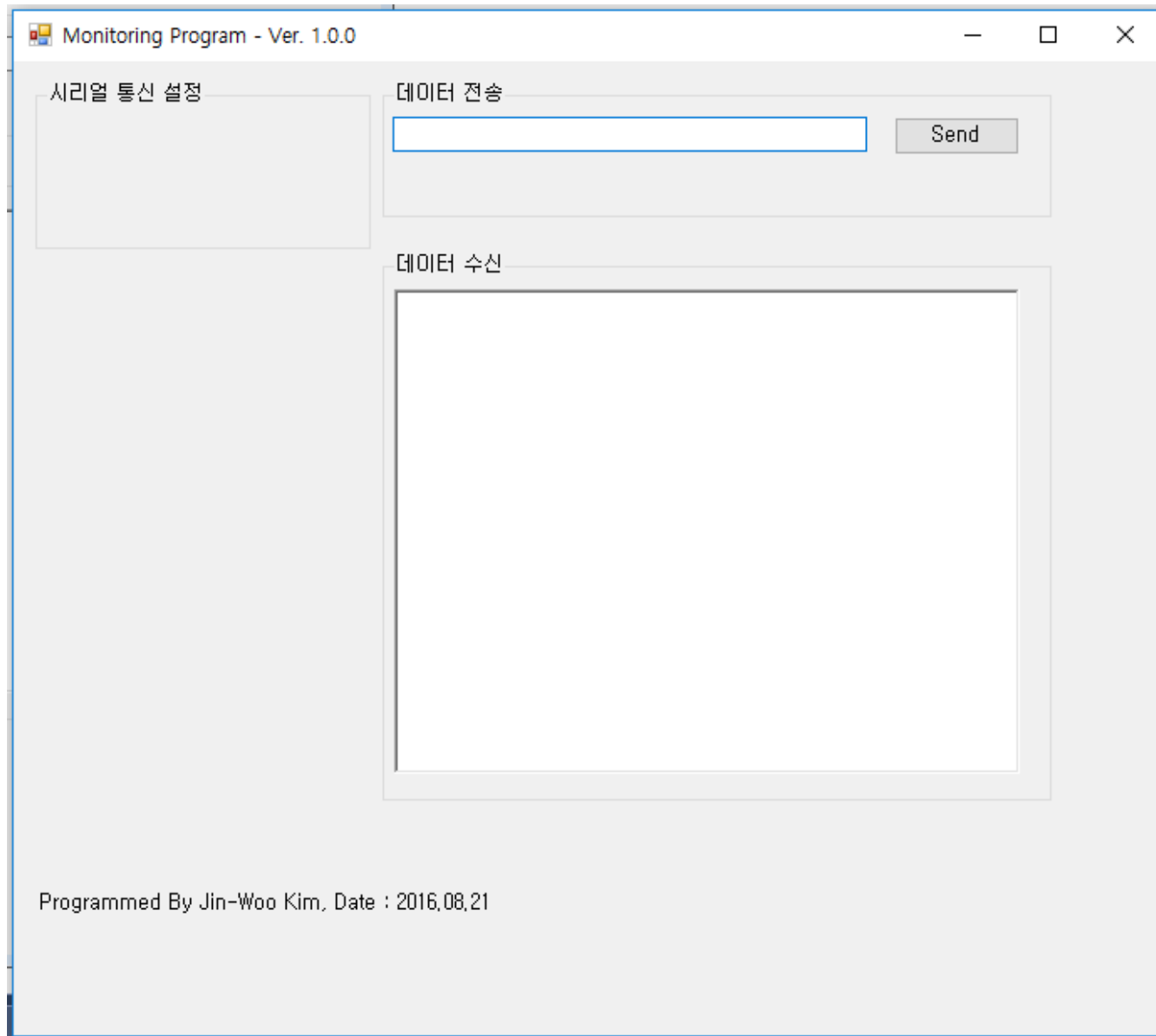
Programmed By Jin-Woo Kim, Date : 2016,08,21

## 4. 프로그램 동작확인

---

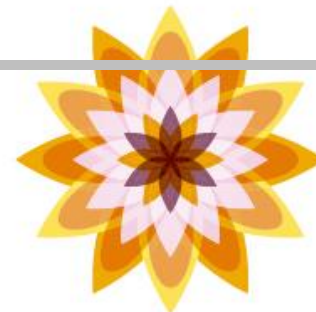
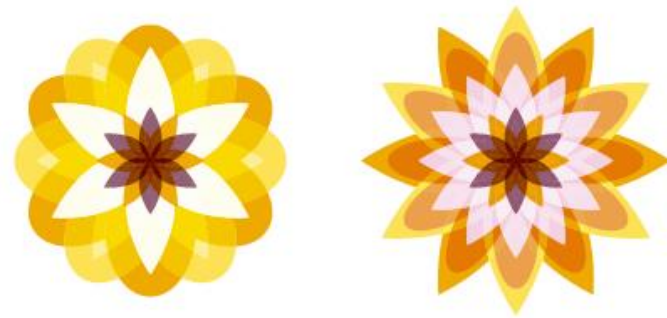
- 그룹박스 컨트롤과 레이블 컨트롤의 배치 및 속성의 변경을 완료하였으면, 저장하기를 누른 다음 Visual Studio 프로그램 상단에 있는 "디버깅 시작(F5)" 버튼을 눌러 프로그램을 실행해 정상적으로 그룹박스 컨트롤과 레이블 컨트롤이 나타나는지 확인합니다.

## 4. 프로그램 동작확인



*Chapter 04*

# 시리얼 통신 설정 기 능추가



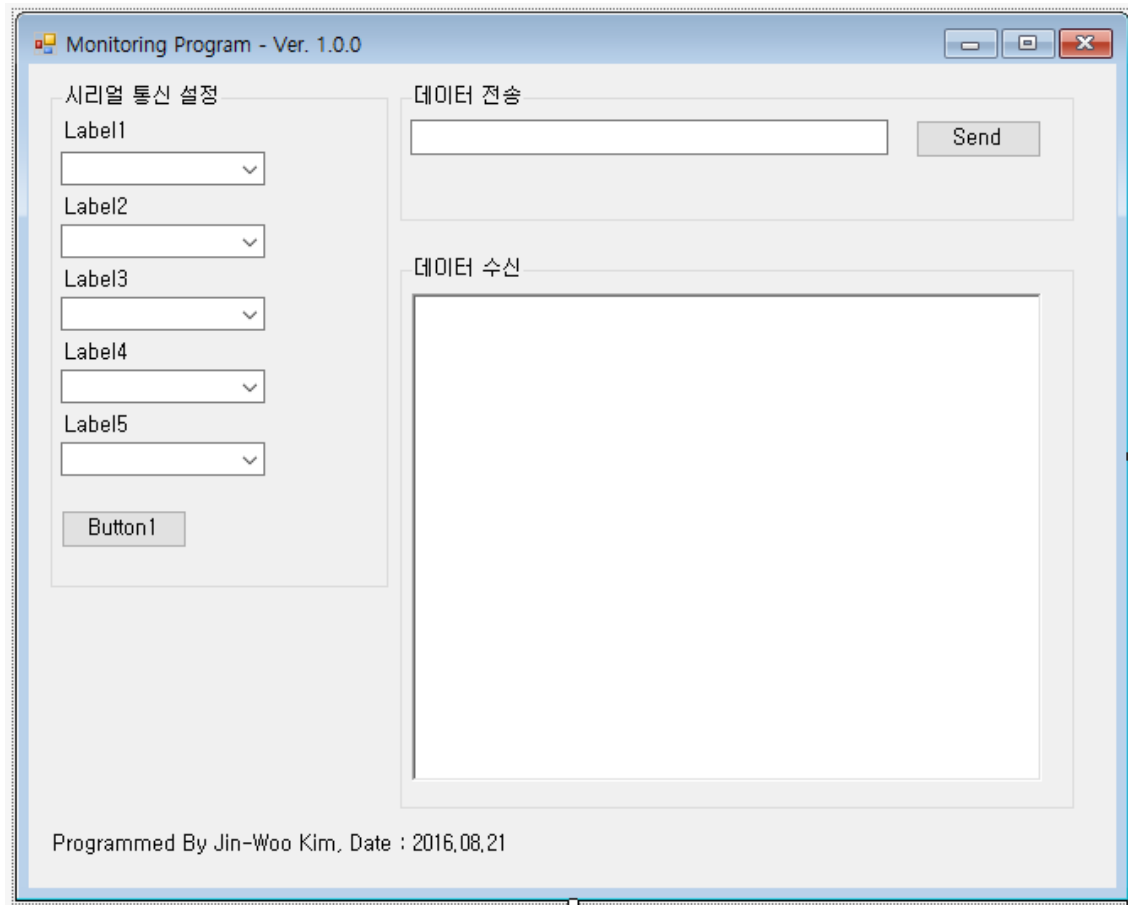


# 목차

1. "시리얼 통신 설정" 그룹박스에 컨트롤 배치
2. 시리얼 통신 컨트롤 추가
3. 이벤트 추가
4. 소스 코드 추가
5. 프로그램 동작확인

# 1. "시리얼 통신 설정" 그룹박스에 컨트롤 배치

- 아래의 그림과 같이 레이블 컨트롤 5개, 콤보박스 컨트롤 5개, 버튼 컨트롤 1개를 배치하도록 한 다음 기본적인 속성은 그대로 두고, 아래와 같이 나타낸 속성들을 동일하게 변경해 줍니다.



# 1. "시리얼 통신 설정" 그룹박스에 컨트롤 배치

- 아래의 그림과 같이 레이블 컨트롤 5개, 콤보박스 컨트롤 5개, 버튼 컨트롤 1개를 배치하도록 한 다음 기본적인 속성은 그대로 두고, 아래와 같이 나타낸 속성들을 동일하게 변경해 줍니다.

- "COM 포트 설정 :" 레이블 속성

(Name) : Label\_COM\_Port\_Setting  
Text : COM 포트 설정 :

- "통신 속도 설정 :" 레이블 속성

(Name) : Label\_Baudrate\_Setting  
Text : 통신 속도 설정 :

# 1. "시리얼 통신 설정" 그룹박스에 컨트롤 배치

- 아래의 그림과 같이 레이블 컨트롤 5개, 콤보박스 컨트롤 5개, 버튼 컨트롤 1개를 배치하도록 한 다음 기본적인 속성은 그대로 두고, 아래와 같이 나타낸 속성들을 동일하게 변경해 줍니다.

- "데이터 Bit 길이 설정 : " 레이블 속성

(Name) : Label\_Data\_Bit\_Setting  
Text : 데이터 Bit 길이 설정 :

- "정지 Bit 길이 설정 : " 레이블 속성

(Name) : Label\_Stop\_Bit\_Setting  
Text : 정지 Bit 길이 설정 :

# 1. "시리얼 통신 설정" 그룹박스에 컨트롤 배치

- 아래의 그림과 같이 레이블 컨트롤 5개, 콤보박스 컨트롤 5개, 버튼 컨트롤 1개를 배치하도록 한 다음 기본적인 속성은 그대로 두고, 아래와 같이 나타낸 속성들을 동일하게 변경해 줍니다.

- "패리티 Bit 설정 : " 레이블 속성

(Name) : Label\_Parity\_Bit\_Setting  
Text : 패리티 Bit 설정 :

- "COM 포트 설정 : " 레이블 하단 콤보박스 속성

(Name) : ComboBox\_COM\_Port\_Setting  
DropDownStyle : DropDownList

# 1. "시리얼 통신 설정" 그룹박스에 컨트롤 배치

- 아래의 그림과 같이 레이블 컨트롤 5개, 콤보박스 컨트롤 5개, 버튼 컨트롤 1개를 배치하도록 한 다음 기본적인 속성은 그대로 두고, 아래와 같이 나타낸 속성들을 동일하게 변경해 줍니다.

- "통신 속도 설정 : " 레이블 하단 콤보박스 속성

(Name) : ComboBox\_Baudrate\_Setting  
DropDownStyle : DropDownList

- "데이터 Bit 길이 설정 : " 레이블 하단 콤보박스 속성

(Name) : ComboBox\_Data\_Bit\_Setting  
DropDownStyle : DropDownList  
Size : 125, 26

# 1. "시리얼 통신 설정" 그룹박스에 컨트롤 배치

- 아래의 그림과 같이 레이블 컨트롤 5개, 콤보박스 컨트롤 5개, 버튼 컨트롤 1개를 배치하도록 한 다음 기본적인 속성은 그대로 두고, 아래와 같이 나타낸 속성들을 동일하게 변경해 줍니다.

- "정지 Bit 길이 설정 : " 레이블 하단 콤보박스 속성

(Name) : ComboBox\_Stop\_Bit\_Setting  
DropDownStyle : DropDownList  
Size : 125, 26

- "패리티 Bit 설정 : " 레이블 하단 콤보박스 속성

(Name) : ComboBox\_Parity\_Bit\_Setting  
DropDownStyle : DropDownList  
Size : 125, 26

# 1. "시리얼 통신 설정" 그룹박스에 컨트롤 배치

- 아래의 그림과 같이 레이블 컨트롤 5개, 콤보박스 컨트롤 5개, 버튼 컨트롤 1개를 배치하도록 한 다음 기본적인 속성은 그대로 두고, 아래와 같이 나타낸 속성들을 동일하게 변경해 줍니다.

- "통신 포트 열기" 버튼 컨트롤 속성

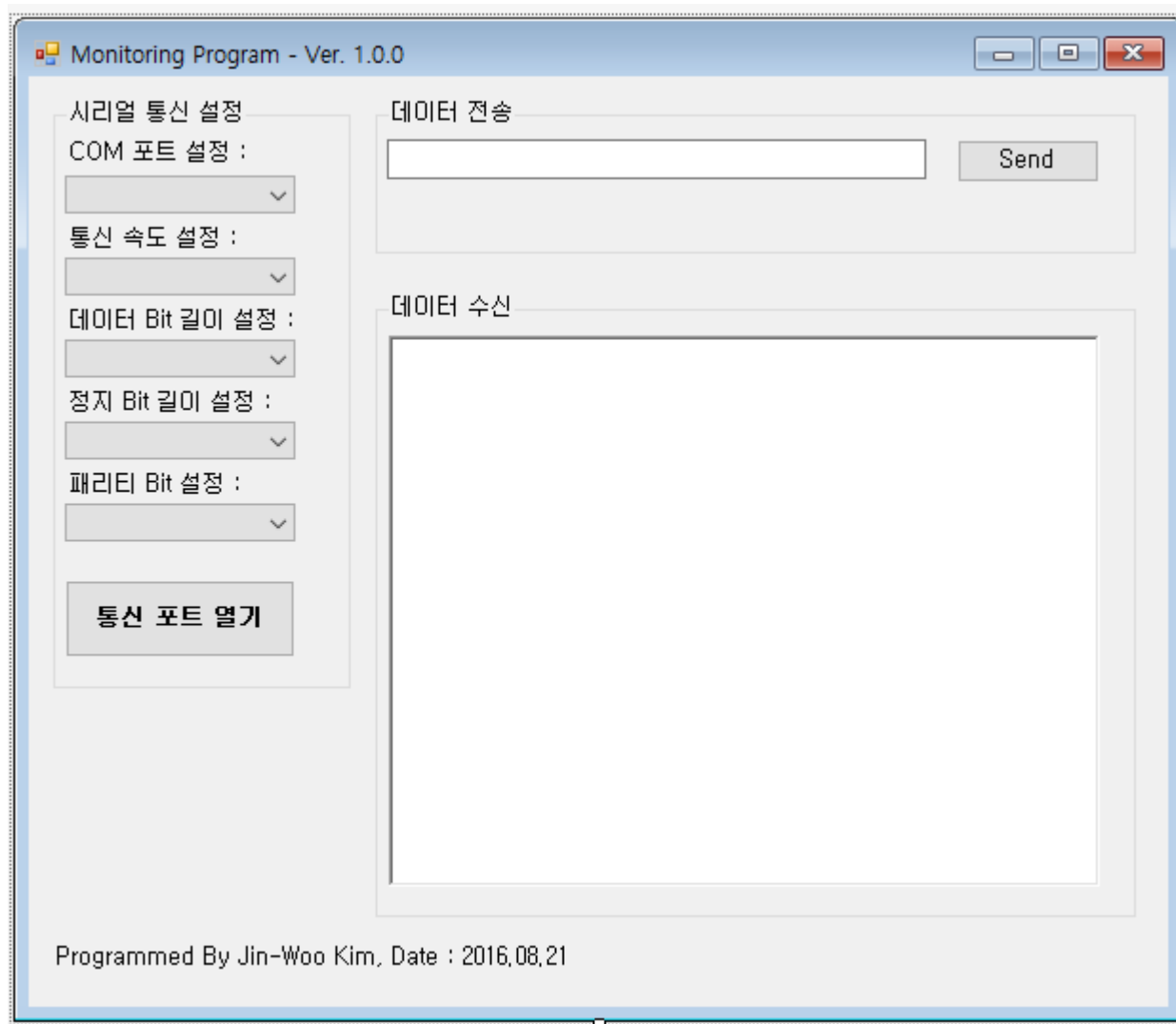
(Name) : Button\_Serial\_Port\_On\_Off

Font : 굴림, 9pt, style=Bold

Text : 통신 포트 열기

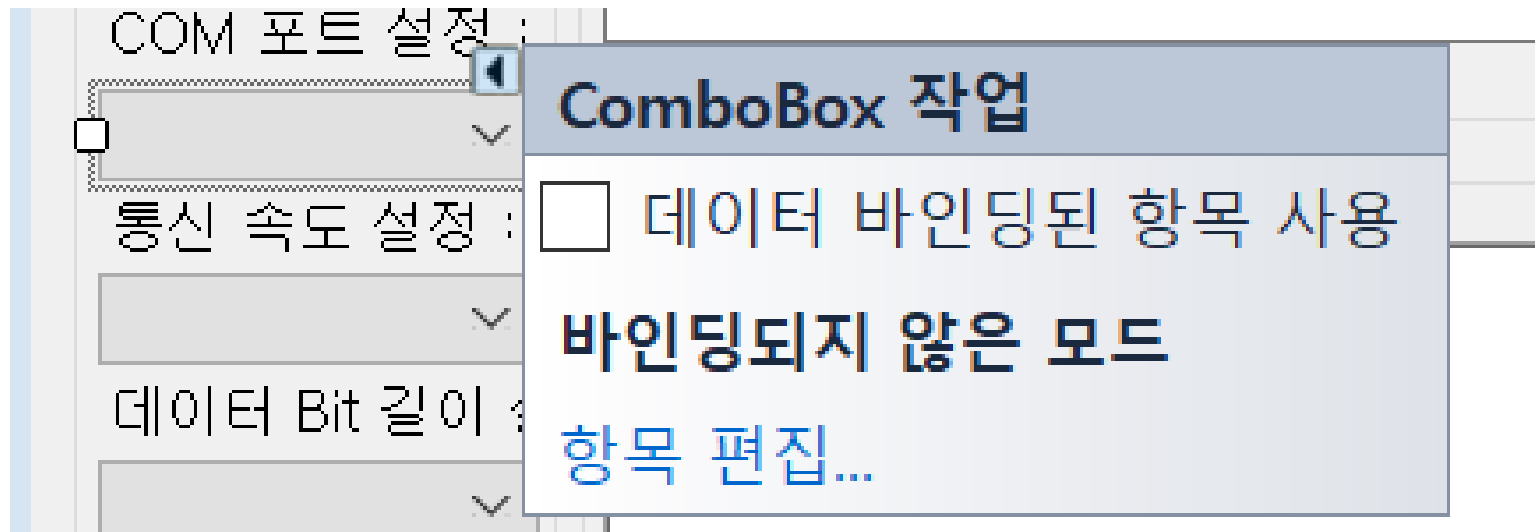


# 1. "시리얼 통신 설정" 그룹박스에 컨트롤 배치

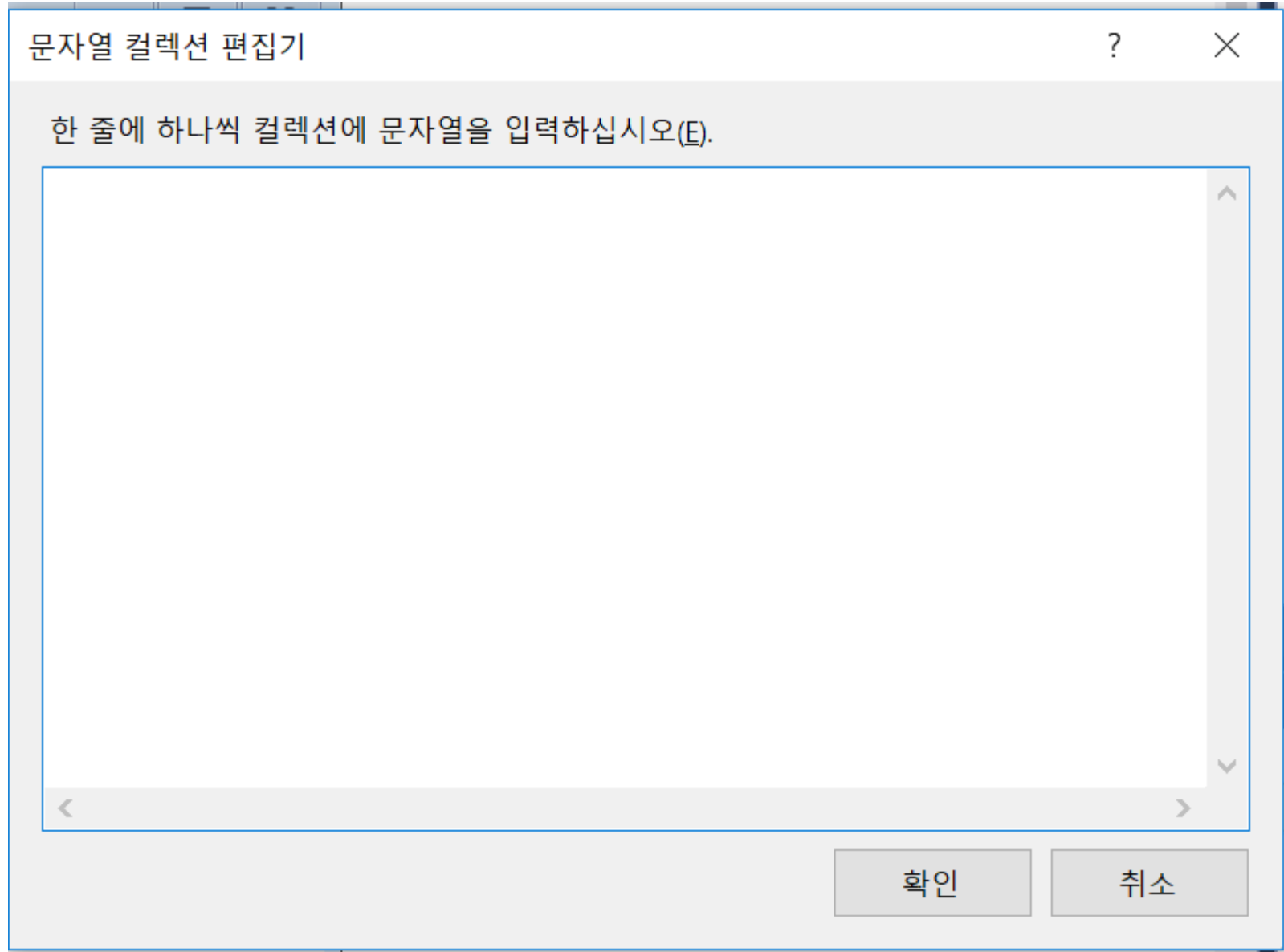


# 1. "시리얼 통신 설정" 그룹박스에 컨트롤 배치

- 레이블 컨트롤 5개, 콤보박스 컨트롤 5개, 버튼 컨트롤 1개의 배치 및 속성 변경이 완료된 다음 콤보박스의 항목을 아래와 같이 추가해 줍니다.
- 항목 추가는 추가한 각각의 콤보박스를 마우스로 클릭하여 포커스를 잡은 뒤, 추가로 나타나는 "▶" 모양의 버튼을 클릭하면 "ComboBox 작업"이라는 팝업 메뉴가 나타나고, 여기서 "항목 편집"을 클릭하면 "문자열 컬렉션 편집기"가 나타나는데 이 문자열 컬렉션 편집기에서 목록을 추가해주면 됩니다.

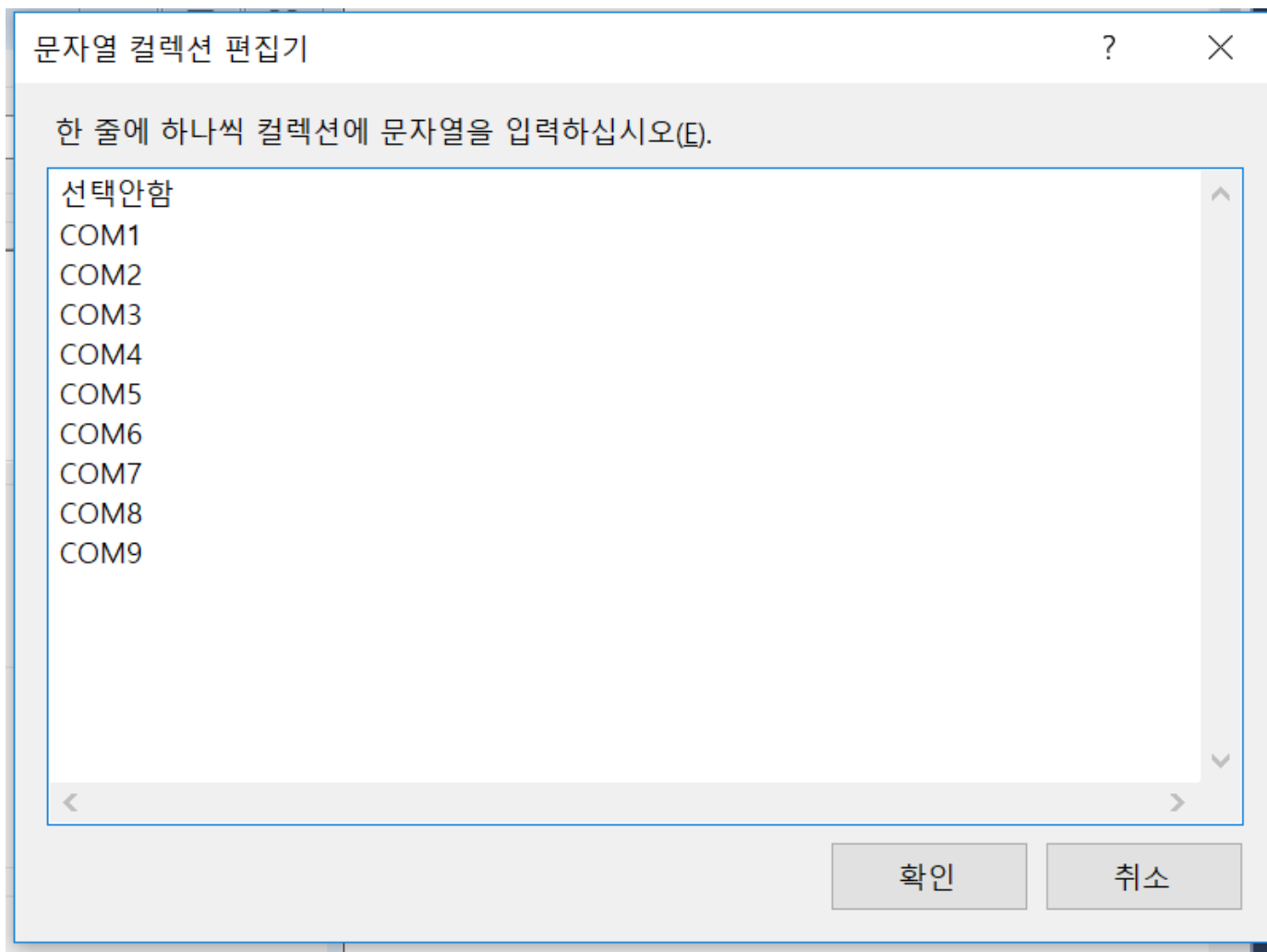


# 1. "시리얼 통신 설정" 그룹박스에 컨트롤 배치



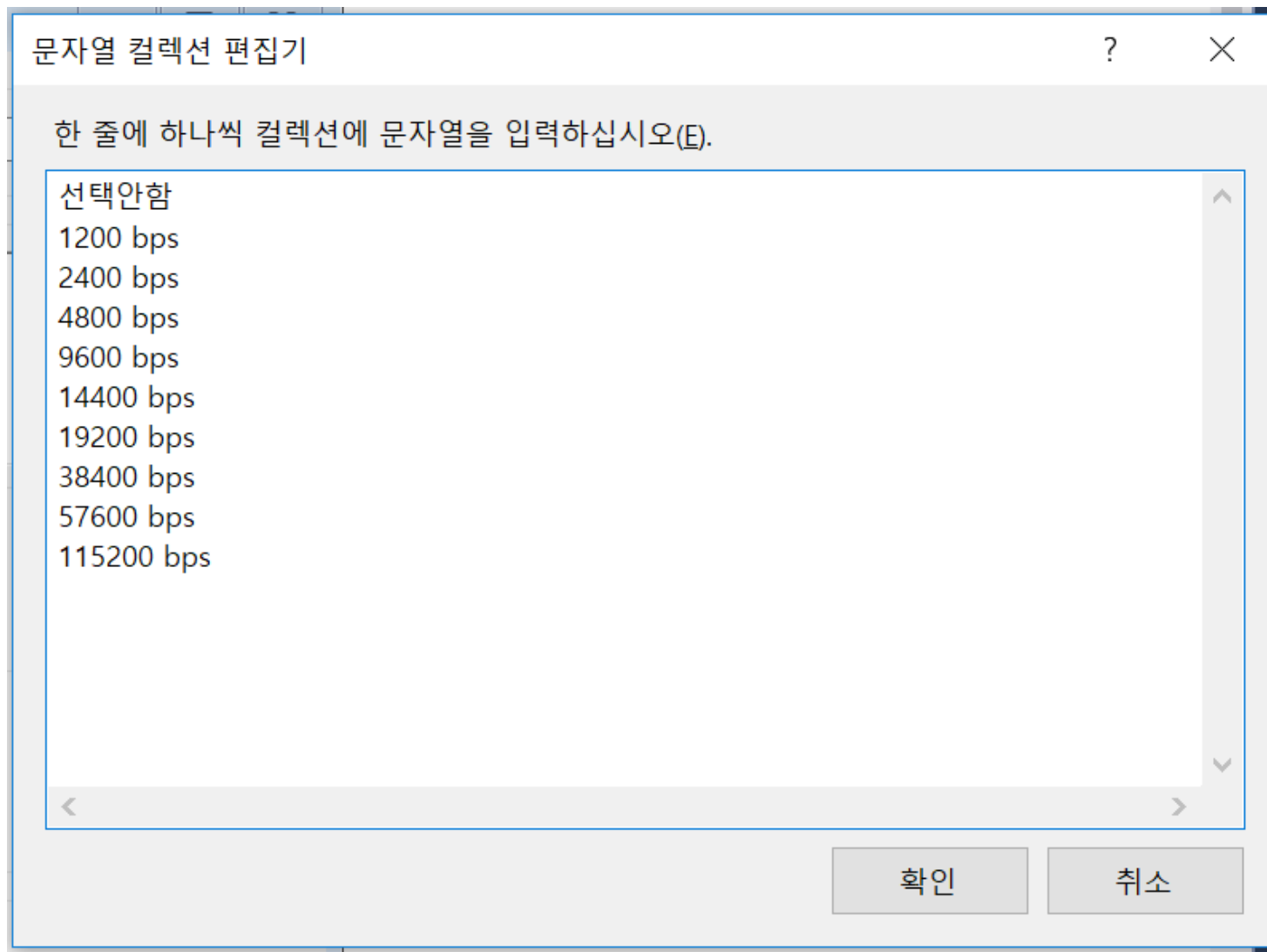
# 1. "시리얼 통신 설정" 그룹박스에 컨트롤 배치

## ■ "COM 포트 설정 : " 레이블 하단 콤보박스 항목 추가



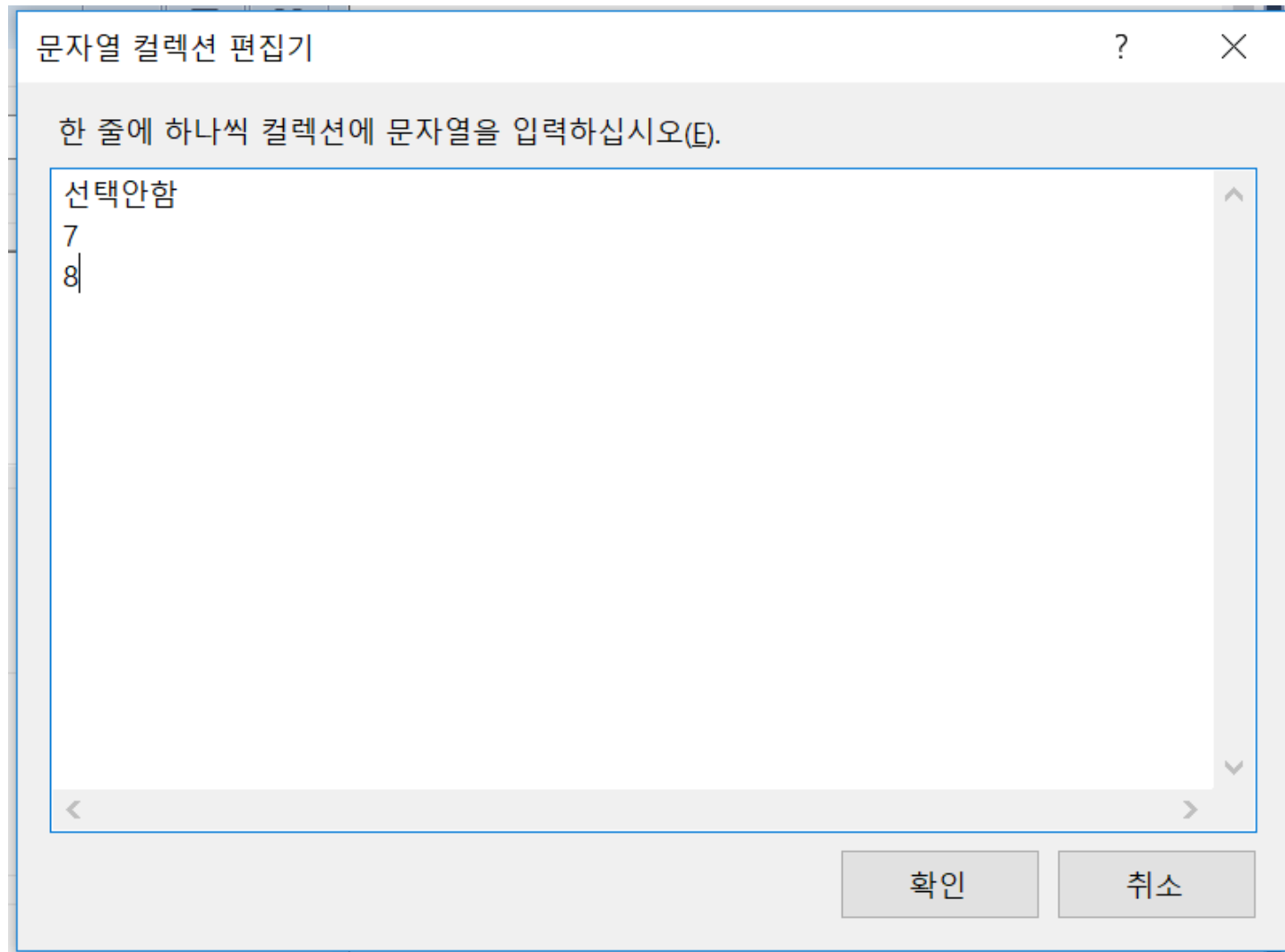
# 1. "시리얼 통신 설정" 그룹박스에 컨트롤 배치

## ■ "통신 속도 설정 : " 레이블 하단 콤보박스 항목 추가



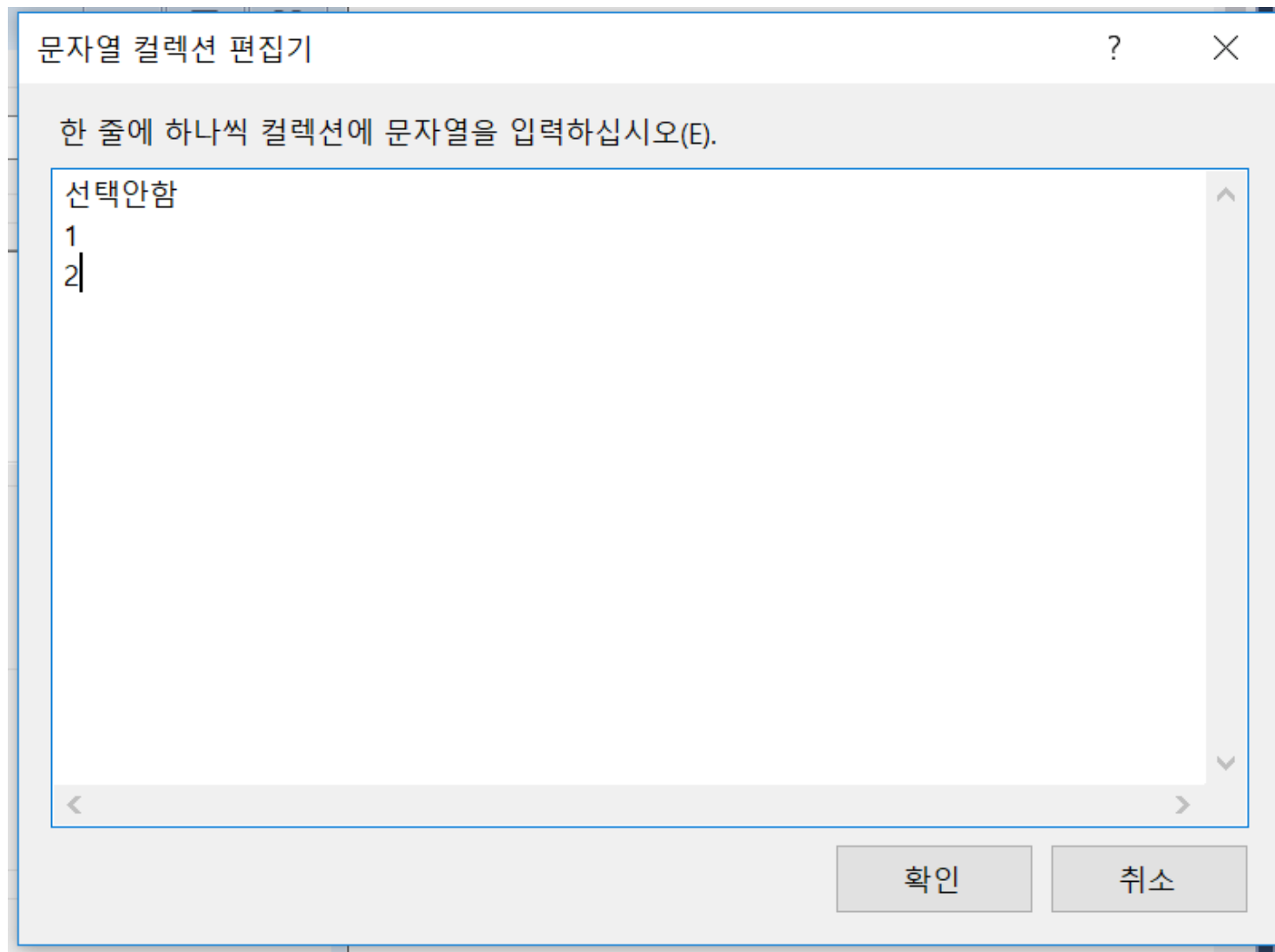
# 1. "시리얼 통신 설정" 그룹박스에 컨트롤 배치

## ■ "데이터 Bit 길이 설정 :" 레이블 하단 콤보박스 항목 추가



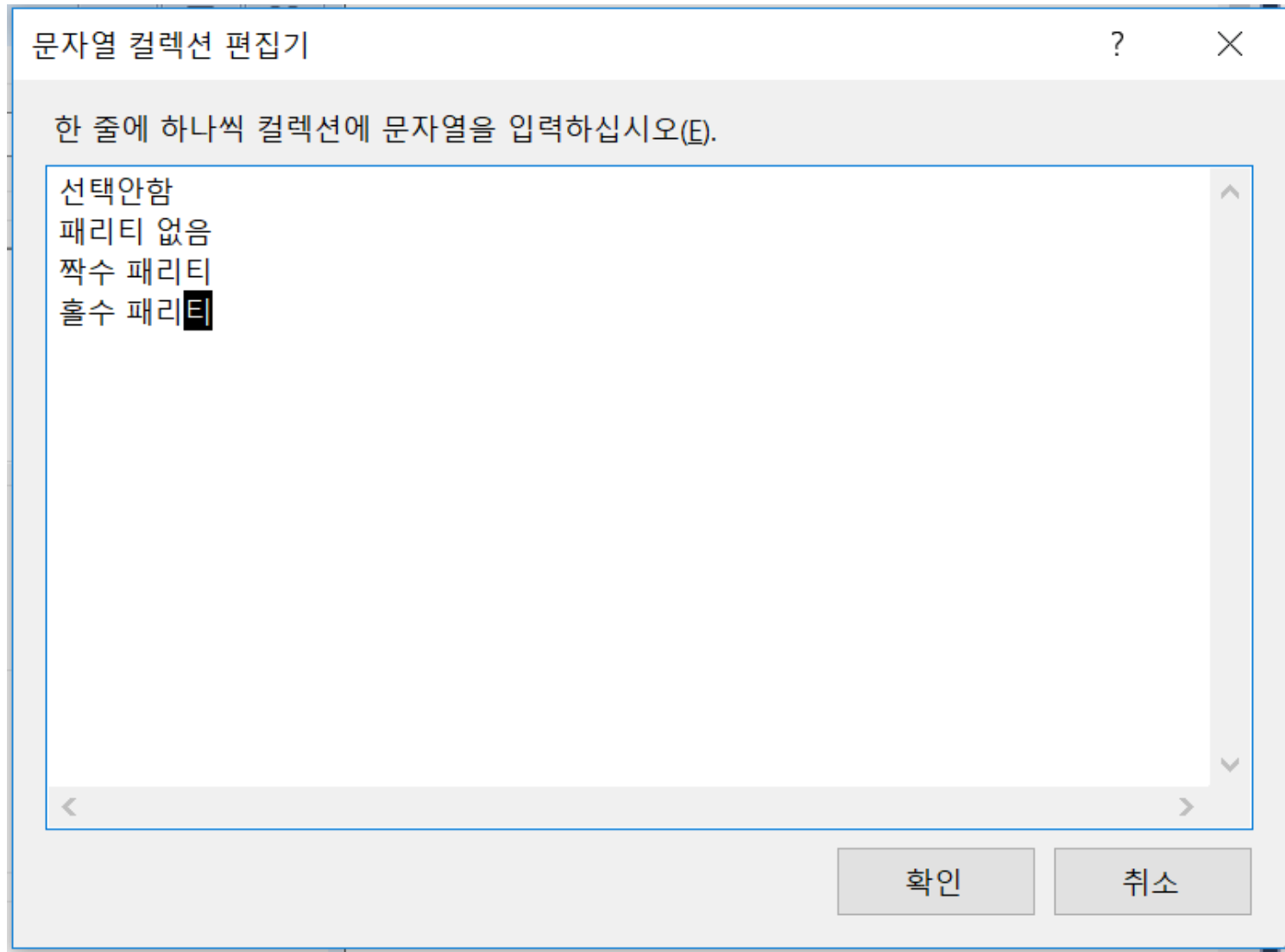
# 1. "시리얼 통신 설정" 그룹박스에 컨트롤 배치

## ■ "정지 Bit 길이 설정 :" 레이블 하단 콤보박스 항목 추가



# 1. "시리얼 통신 설정" 그룹박스에 컨트롤 배치

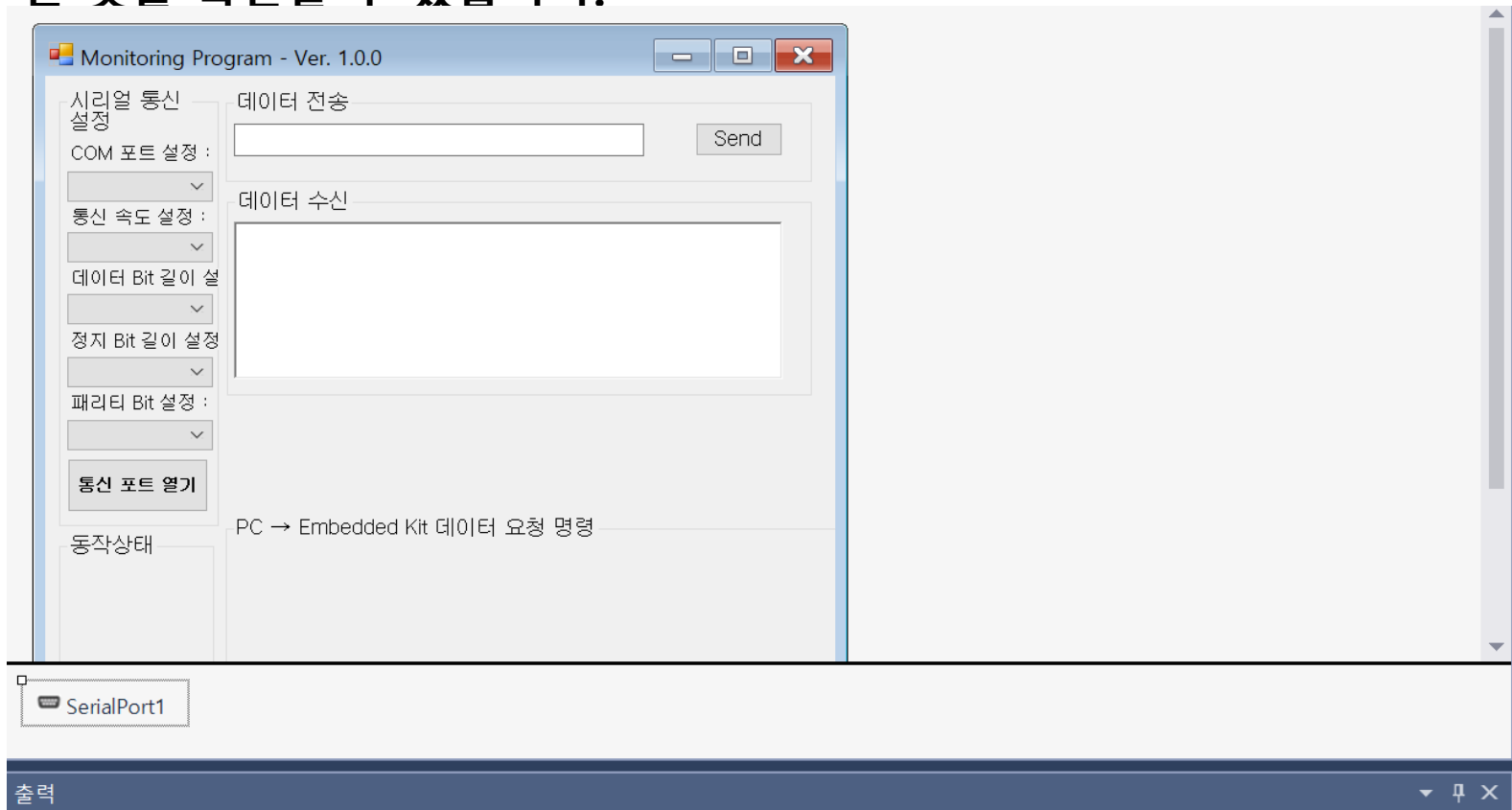
## ■ "패리티 Bit 설정 :" 레이블 하단 콤보박스 항목 추가





## 2. 시리얼 통신 컨트롤 추가

- 이제 폼에 시리얼 통신용 컨트롤을 추가해줍니다.
- Visual Studio 프로그램 좌측에 있는 도구 상자에서 "SerialPort"를 드래그하여 폼에 가져다 놓으면 아래의 그림과 같이 폼 하단에 SerialPort 컨트롤이 배치되는 것을 확인할 수 있습니다.



## 2. 시리얼 통신 컨트롤 추가

- 추가된 SerialPort 컨트롤의 속성을 아래와 같이 변경해 줍니다. 기본값으로 설정된 속성은 그대로 두고, 아래의 속성만 변경하도록 합니다.
  - SerialPort 컨트롤 속성

(Name) : SerialPort1

### 3. 이벤트 추가

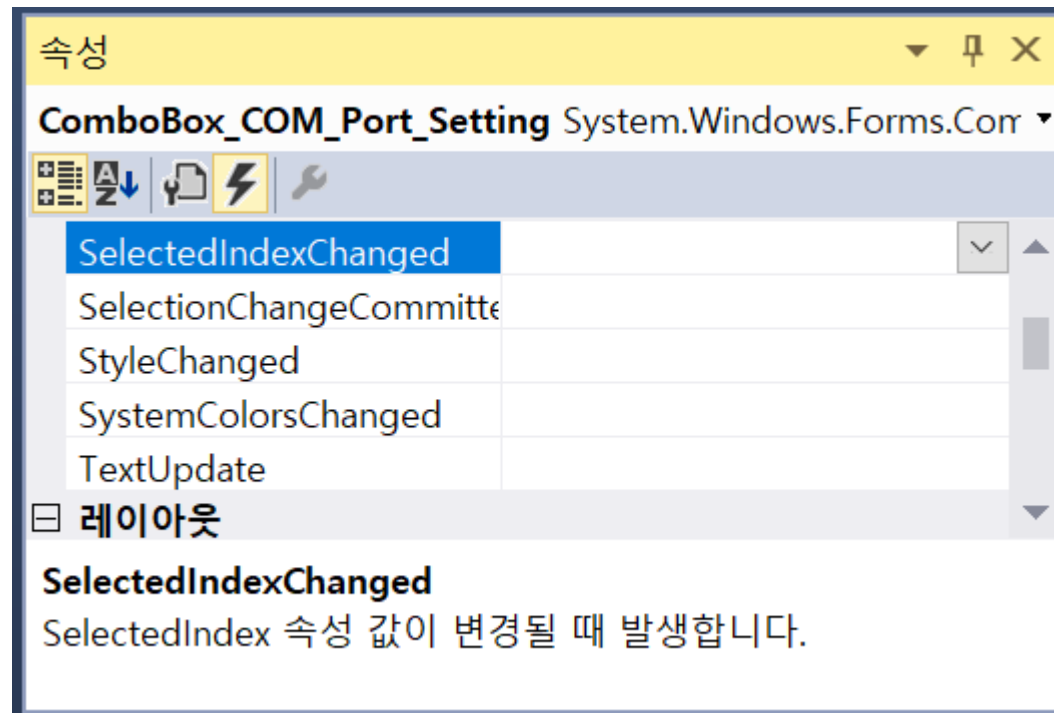
- Form을 더블클릭해 폼이 최초에 로드될 때의 이벤트를 추가해주도록 합니다.
- Form을 더블클릭하여 추가된 이벤트는 아래와 같이 소스 코드 창으로 전환되면서 추가되는 것을 확인할 수 있습니다.

```
Public Class Form_Monitoring_Program_Main
    Private Sub Form_Monitoring_Program_Main_Load
(sender As Object, e As EventArgs) Handles MyBase.Load

        End Sub
End Class
```

### 3. 이벤트 추가

- 다음으로 이전에 추가했던 콤보박스에 이벤트를 추가해줍니다.
- 아래의 그림과 같이 콤보박스를 마우스로 클릭하여 포커스를 잡은 다음 번개 모양의 아이콘을 눌러 이벤트 창으로 전환하여, "SelectedIndexChanged" 이벤트를 찾은 다음 해당 항목을 더블클릭해 이벤트를 추가해 줍니다.
- 추가했던 다섯개의 콤보박스에 동일하게 적용합니다.



### 3. 이벤트 추가

- 이벤트가 추가되면 아래와 같이 소스 코드가 추가되어 있는 것을 확인할 수 있습니다.

```
Public Class Form_Monitoring_Program_Main
    Private Sub Form_Monitoring_Program_Main_Load
(sender As Object, e As EventArgs) Handles MyBase.Load

        End Sub

        Private Sub ComboBox_COM_Port_Setting_SelectedIndexChanged
(sender As Object, e As EventArgs) Handles
ComboBox_COM_Port_Setting.SelectedIndexChanged

            End Sub
End Class
```

### 3. 이벤트 추가

- 마지막으로 시리얼 통신 컨트롤에도 이벤트를 추가해줍니다.
- 이벤트 창에서 "DataReceived" 이벤트를 확인한 다음 해당 항목을 더블클릭해 이벤트를 추가해줍니다.

```
Public Class Form_Monitoring_Program_Main
    Private Sub SerialPort1_DataReceived
(sender As Object, e As IO.Ports.SerialDataReceivedEventArgs) Handles
SerialPort1.DataReceived

    End SubEnd Class
```

## 4. 소스 코드 추가

---

- 이제 본격적으로 소스 코드를 추가해 줍니다.
- 아래의 소스 코드는 프로그램이 최초에 로드될 때 콤보박스의 항목을 지정해주고, 콤보박스의 항목을 선택함에 따라서 각각의 콤보박스가 단계적으로 활성화되는 기능을 담고 있습니다.
- 또한 콤보박스의 항목을 선택해줌에 따라서 Form에 추가했던 시리얼 통신 컨트롤의 통신 속성들을 변경해주는 기능을 합니다.

## 4. 소스 코드 추가

```
Private Sub Form_Monitoring_Program_Main_Load  
(sender As Object, e As EventArgs) Handles MyBase.Load  
    ' 최초 Form 로드시 발생 이벤트  
    ' 최초 Form 로드시 콤보박스의 기본 선택 항목 지정  
    ComboBox_COM_Port_Setting.Text  
        = ComboBox_COM_Port_Setting.Items(0)  
    ComboBox_Baudrate_Setting.Text  
        = ComboBox_Baudrate_Setting.Items(0)  
    ComboBox_Data_Bit_Setting.Text  
        = ComboBox_Data_Bit_Setting.Items(0)  
    ComboBox_Stop_Bit_Setting.Text  
        = ComboBox_Stop_Bit_Setting.Items(0)  
    ComboBox_Parity_Bit_Setting.Text  
        = ComboBox_Parity_Bit_Setting.Items(0)
```



## 4. 소스 코드 추가

- 그 밑에 다음 코드도 추가합니다.

```
'최초 Form 로드 시 콤보박스 항목값 선택 불가  
    ComboBox_Baudrate_Setting.Enabled = False  
    ComboBox_Data_Bit_Setting.Enabled = False  
    ComboBox_Data_Bit_Setting.Enabled = False  
    ComboBox_Parity_Bit_Setting.Enabled = False  
    Button_Serial_Port_On_Off.Enabled = False  
End Sub
```

## 4. 소스 코드 추가

### ■ 다음 코드도 추가합니다.

```
Private Sub ComboBox_COM_Port_Setting_SelectedIndexChanged  
(sender As Object, e As EventArgs) Handles  
    ComboBox_COM_Port_Setting.SelectedIndexChanged  
        'COM 포트 설정 콤보박스 항목 변경 시 발생 이벤트  
        'COM 포트 설정 콤보박스 항목 변경 시 시리얼 포트의  
COM 포트값 지정  
        If (ComboBox_COM_Port_Setting.SelectedIndex <> 0) Then  
            ComboBox_Baudrate_Setting.Enabled = True  
  
            If (ComboBox_COM_Port_Setting.SelectedIndex = 1) Then  
                SerialPort1.PortName = "COM1"  
            ElseIf (ComboBox_COM_Port_Setting.SelectedIndex = 2) Then  
                SerialPort1.PortName = "COM2"  
            ElseIf (ComboBox_COM_Port_Setting.SelectedIndex = 3) Then  
                SerialPort1.PortName = "COM3"  
            ElseIf (ComboBox_COM_Port_Setting.SelectedIndex = 4) Then  
                SerialPort1.PortName = "COM4"
```

## 4. 소스 코드 추가

- 그 밑에 다음 코드도 추가합니다.

```
Elseif (ComboBox_COM_Port_Setting.SelectedIndex = 5) Then  
    SerialPort1.PortName = "COM5"  
Elseif (ComboBox_COM_Port_Setting.SelectedIndex = 6) Then  
    SerialPort1.PortName = "COM6"  
Elseif (ComboBox_COM_Port_Setting.SelectedIndex = 7) Then  
    SerialPort1.PortName = "COM7"  
Elseif (ComboBox_COM_Port_Setting.SelectedIndex = 8) Then  
    SerialPort1.PortName = "COM8"  
Elseif (ComboBox_COM_Port_Setting.SelectedIndex = 9) Then  
    SerialPort1.PortName = "COM9"
```

```
End If
```

```
End Sub
```

## 4. 소스 코드 추가

### ■ 다음 코드도 추가합니다.

```
Private Sub ComboBox_Baudrate_Setting_SelectedIndexChanged  
(sender As Object, e As EventArgs) Handles  
    ComboBox_Baudrate_Setting.SelectedIndexChanged  
  
    '통신 속도 설정 콤보박스 항목 변경 시 발생 이벤트  
    '통신 속도 설정 콤보박스 항목 변경 시 시리얼 포트의 통신 속도값 지정  
  
    If (ComboBox_Baudrate_Setting.SelectedIndex <> 0) Then  
        ComboBox_Data_Bit_Setting.Enabled = True  
  
        If (ComboBox_Baudrate_Setting.SelectedIndex = 1) Then  
            SerialPort1.BaudRate = 1200  
        ElseIf (ComboBox_Baudrate_Setting.SelectedIndex = 2) Then  
            SerialPort1.BaudRate = 2400  
        ElseIf (ComboBox_Baudrate_Setting.SelectedIndex = 3) Then  
            SerialPort1.BaudRate = 4800  
        ElseIf (ComboBox_Baudrate_Setting.SelectedIndex = 4) Then  
            SerialPort1.BaudRate = 9600
```

## 4. 소스 코드 추가

- 그 밑에 다음 코드도 추가합니다.

```
Elseif (ComboBox_Baudrate_Setting.SelectedIndex = 5) Then  
    SerialPort1.BaudRate = 14400  
Elseif (ComboBox_Baudrate_Setting.SelectedIndex = 6) Then  
    SerialPort1.BaudRate = 19200  
Elseif (ComboBox_Baudrate_Setting.SelectedIndex = 7) Then  
    SerialPort1.BaudRate = 38400  
Elseif (ComboBox_Baudrate_Setting.SelectedIndex = 8) Then  
    SerialPort1.BaudRate = 57600  
Elseif (ComboBox_Baudrate_Setting.SelectedIndex = 9) Then  
    SerialPort1.BaudRate = 115200  
End If
```

## 4. 소스 코드 추가

- 그 밑에 다음 코드도 추가합니다.

```
Else
```

```
ComboBox_Data_Bit_Setting.Enabled = False
```

```
ComboBox_Stop_Bit_Setting.Enabled = False
```

```
ComboBox_Parity_Bit_Setting.Enabled = False
```

```
Button_Serial_Port_On_Off.Enabled = False
```

```
ComboBox_Data_Bit_Setting.Text = ComboBox_Data_Bit_Setting.Items(0)
```

```
ComboBox_Stop_Bit_Setting.Text = ComboBox_Stop_Bit_Setting.Items(0)
```

```
ComboBox_Parity_Bit_Setting.Text = ComboBox_Parity_Bit_Setting.Items(0)
```

```
End If
```

```
End Sub
```

## 4. 소스 코드 추가

### ■ 다음 코드도 추가합니다.

```
Private Sub ComboBox_Data_Bit_Setting_SelectedIndexChanged  
(sender As Object, e As EventArgs) Handles  
    ComboBox_Data_Bit_Setting.SelectedIndexChanged  
        '데이터 Bit 설정 콤보박스 항목 변경 시 발생 이벤트  
        '데이터 Bit 설정 콤보박스 항목 변경 시 시리얼 포트의 데이터 Bit 길이값 지정  
        If (ComboBox_Data_Bit_Setting.SelectedIndex <> 0) Then  
            ComboBox_Stop_Bit_Setting.Enabled = True  
  
            If (ComboBox_Data_Bit_Setting.SelectedIndex = 1) Then  
                SerialPort1.DataBits = 7  
            ElseIf (ComboBox_Data_Bit_Setting.SelectedIndex = 2) Then  
                SerialPort1.DataBits = 8  
            End If  
        End If
```

## 4. 소스 코드 추가

### ■ 다음 코드도 추가합니다.

```
Private Sub ComboBox_Stop_Bit_Setting_SelectedIndexChanged  
(sender As Object, e As EventArgs) Handles  
    ComboBox_Stop_Bit_Setting.SelectedIndexChanged  
        '정지 Bit 설정 콤보박스 항목 변경 시 발생 이벤트  
        '정지 Bit 설정 콤보박스 항목 변경 시 시리얼 포트의 정지 Bit 길이값 지정  
        If (ComboBox_Stop_Bit_Setting.SelectedIndex <> 0) Then  
            ComboBox_Parity_Bit_Setting.Enabled = True  
  
            If (ComboBox_Stop_Bit_Setting.SelectedIndex = 1) Then  
                SerialPort1.StopBits = 1  
            ElseIf (ComboBox_Stop_Bit_Setting.SelectedIndex = 2) Then  
                SerialPort1.StopBits = 2  
            End If  
        End If  
    End Sub
```



## 4. 소스 코드 추가

- 그 밑에 다음 코드도 추가합니다.

```
Else
```

```
    ComboBox_Parity_Bit_Setting.Enabled = False
```

```
    Button_Serial_Port_On_Off.Enabled = False
```

```
    ComboBox_Parity_Bit_Setting.Text = ComboBox_Parity_Bit_Setting.Items(0)
```

```
End If
```

```
End Sub
```

## 4. 소스 코드 추가

### ■ 다음 코드도 추가합니다.

```
Private Sub ComboBox_Parity_Bit_Setting_SelectedIndexChanged  
(sender As Object, e As EventArgs) Handles  
    ComboBox_Parity_Bit_Setting.SelectedIndexChanged  
        '패리티 Bit 설정 콤보박스 항목 변경 시 발생 이벤트  
        '패리티 Bit 설정 콤보박스 항목 변경 시 시리얼 포트의 패리티 Bit 설정값 지정  
        If (ComboBox_Parity_Bit_Setting.SelectedIndex <> 0) Then  
            Button_Serial_Port_On_Off.Enabled = True  
  
            If (ComboBox_Parity_Bit_Setting.SelectedIndex = 1) Then  
                SerialPort1.Parity = IO.Ports.Parity.None  
            ElseIf (ComboBox_Parity_Bit_Setting.SelectedIndex = 2) Then  
                SerialPort1.Parity = IO.Ports.Parity.Odd  
            ElseIf (ComboBox_Parity_Bit_Setting.SelectedIndex = 3) Then  
                SerialPort1.Parity = IO.Ports.Parity.Even  
            End If  
        End If
```

## 4. 소스 코드 추가

- 그 밑에 다음 코드도 추가합니다.

```
Else  
    Button_Serial_Port_On_Off.Enabled = False  
End If  
End Sub
```

## 4. 소스 코드 추가

---

```
Public Class Form_Monitoring_Program_Main
```

```
    Dim Val_Button_Serial_Port_On_Off_Toggle As Integer = 0
```

## 4. 소스 코드 추가

```
Private Sub Button_Serial_Port_On_Off_Click  
(sender As Object, e As EventArgs) Handles Button_Serial_Port_On_Off.Click  
    '통신 포트 열기 버튼을 클릭할 때, 실제 통신 포트가 열리도록 하며,  
    다시 한 번 더 누르게 되면,  
    '통신 포트를 닫도록 기능 전환(통신 포트 닫기 문자가 나타남과  
    동시 실제 시리얼 포트 닫힘)  
    '통신 포트 열림/닫힘에 따라서 프로그램 로그에 관련 로그가 남도록 하는  
    기능도 추가되어 있음  
    Val_Button_Serial_Port_On_Off_Toggle  
= Val_Button_Serial_Port_On_Off_Toggle Xor 1  
  
    If (Val_Button_Serial_Port_On_Off_Toggle = 1) Then  
        Try  
            SerialPort1.Open()  
            Button_Serial_Port_On_Off.Text = "통신 포트 닫기"
```

## 4. 소스 코드 추가

```
ComboBox_COM_Port_Setting.Enabled = False  
ComboBox_Baudrate_Setting.Enabled = False  
ComboBox_Data_Bit_Setting.Enabled = False  
ComboBox_Stop_Bit_Setting.Enabled = False  
ComboBox_Parity_Bit_Setting.Enabled = False
```

## 4. 소스 코드 추가

Catch ex As Exception

Val\_Button\_Serial\_Port\_On\_Off\_Toggle  
= Val\_Button\_Serial\_Port\_On\_Off\_Toggle Xor 1

Button\_Serial\_Port\_On\_Off.Text = "통신 포트 열기"

ComboBox\_COM\_Port\_Setting.Enabled = True

ComboBox\_Baudrate\_Setting.Enabled = True

ComboBox\_Data\_Bit\_Setting.Enabled = True

ComboBox\_Stop\_Bit\_Setting.Enabled = True

ComboBox\_Parity\_Bit\_Setting.Enabled = True

End Try

## 4. 소스 코드 추가

Else

SerialPort1.Close()

Button\_Serial\_Port\_On\_Off.Text = "통신 포트 열기"

ComboBox\_COM\_Port\_Setting.Enabled = True

ComboBox\_Baudrate\_Setting.Enabled = True

ComboBox\_Data\_Bit\_Setting.Enabled = True

ComboBox\_Stop\_Bit\_Setting.Enabled = True

ComboBox\_Parity\_Bit\_Setting.Enabled = True

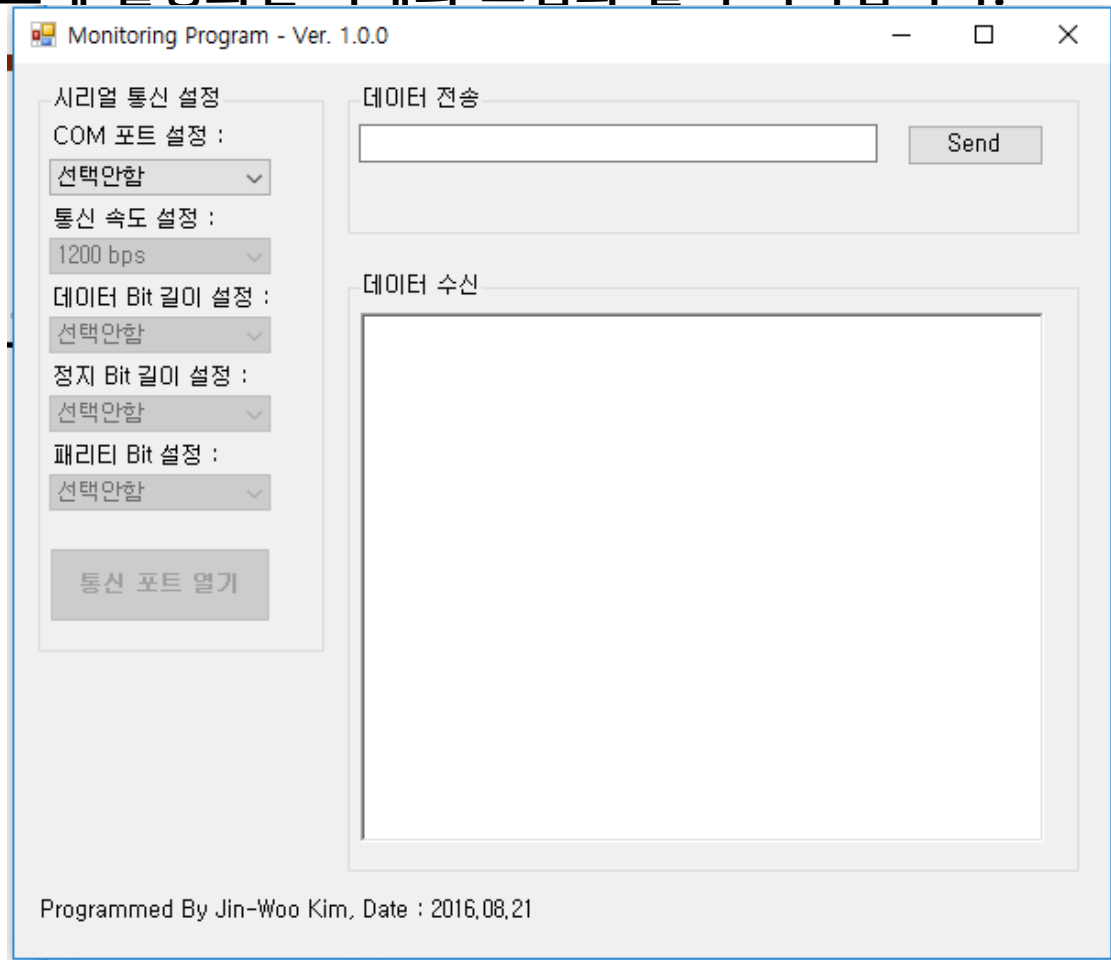
End If

End Sub



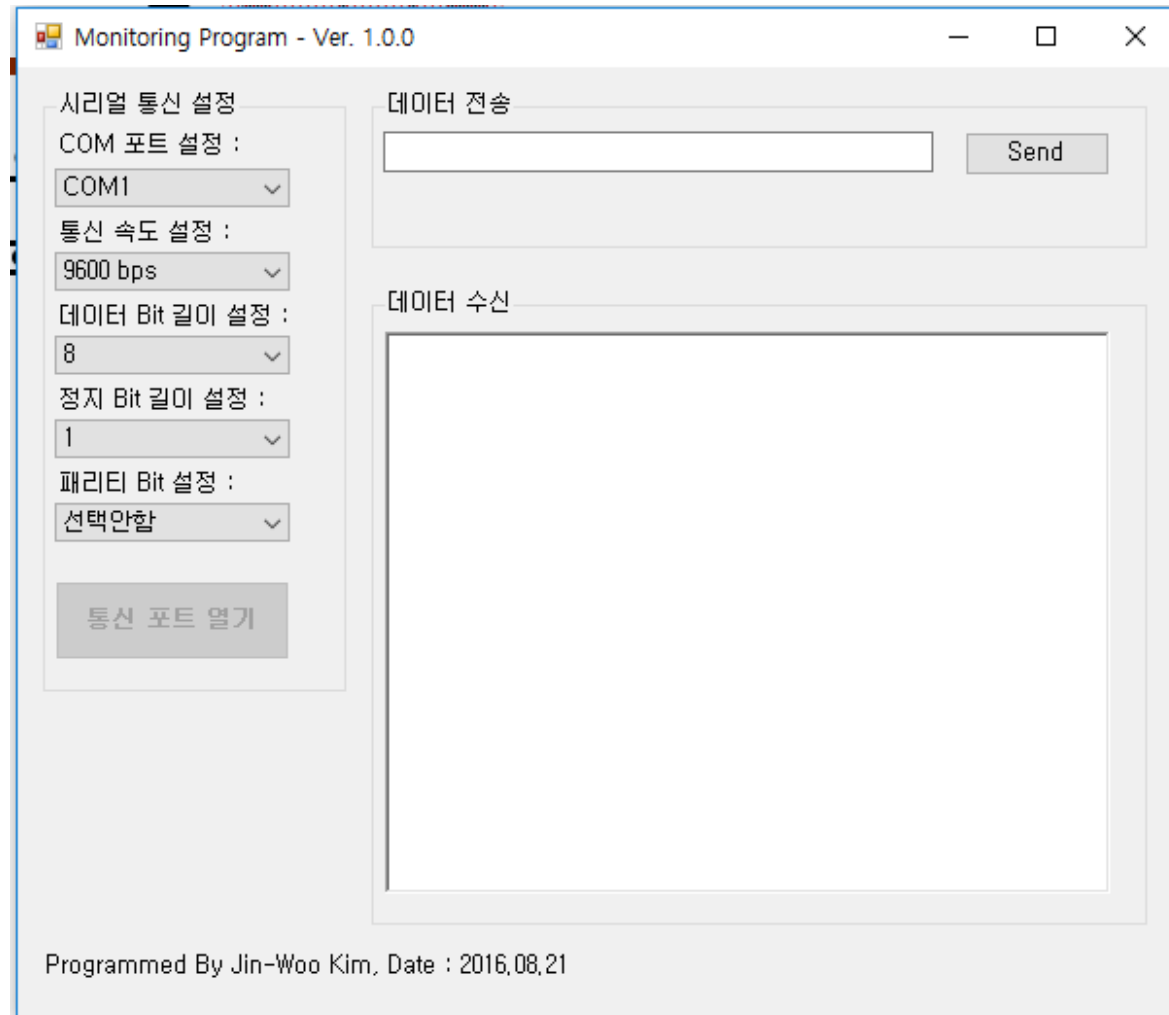
## 5. 프로그램 동작확인

- 그림 소스 코드의 추가가 완료되면, Visual Studio의 프로그램 상단 중앙에 있는 "디버깅 시작(F5)"를 클릭해 프로그램이 동작하는 것을 확인합니다.
- 프로그램이 최초에 실행되면 아래의 그림과 같이 나타납니다.



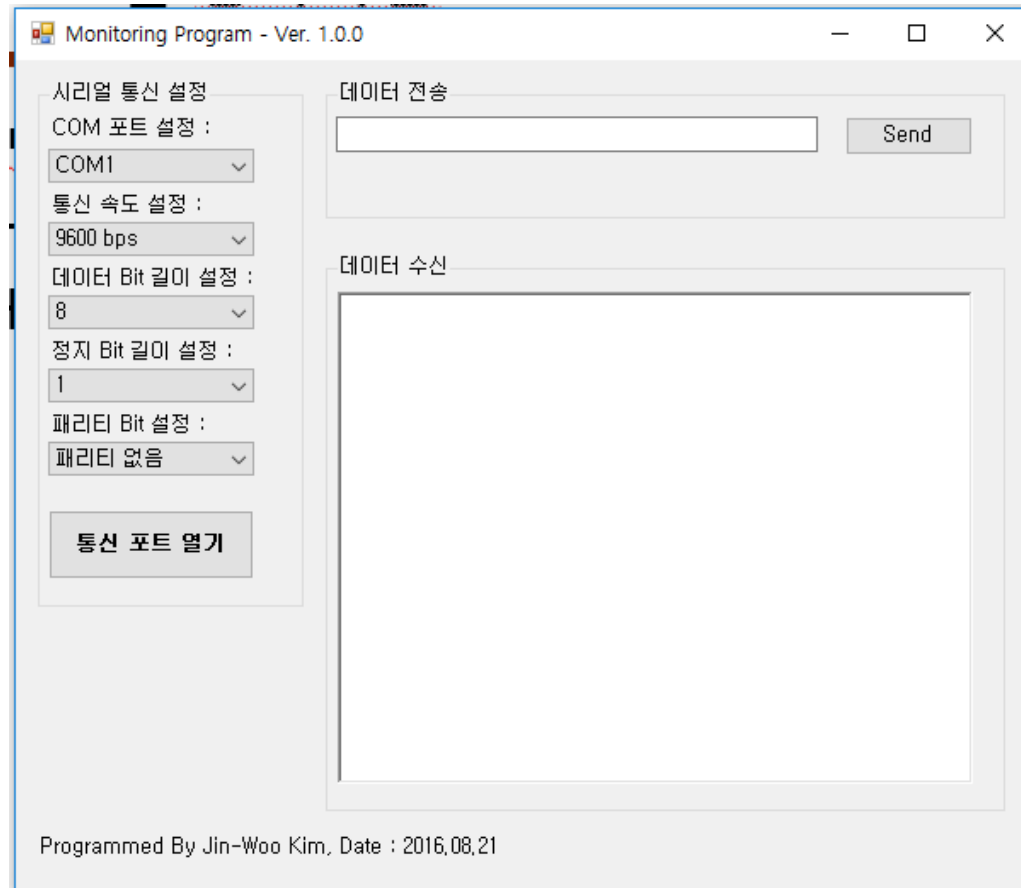
## 5. 프로그램 동작확인

- 다음으로 콤보박스 하나하나의 항목을 지정함에 따라서 단계적으로 콤보박스가 활성화되는 것을 알 수 있습니다.



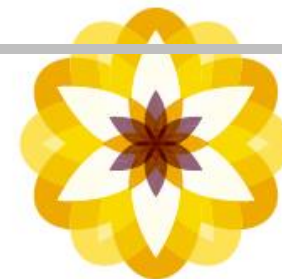
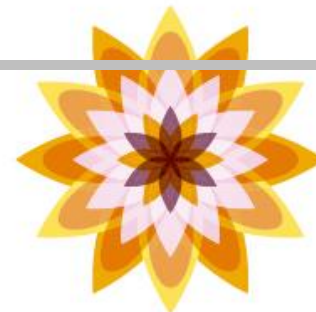
## 5. 프로그램 동작확인

- 콤보박스의 모든 항목을 설정하면, 제일 마지막의 "통신 포트 열기" 버튼이 활성화되는 것을 확인할 수 있고, 반대로 콤보박스의 항목을 "선택안함"으로 설정하면 해당 콤보박스 이후의 콤보박스 및 버튼이 비활성화되는 것을 알 수 있습니다



*Chapter 05*

# 시리얼 통신 명령 송 신기능 추가



# 1. 소스추가

- 명령 송신을 위한 Send버튼을 각각 더블 클릭하여 아래와 같이 소스 코드를 추가해줍니다.

```
Private Sub btnSend_Click(sender As Object, e As EventArgs) Handles btnSend.Click
    SerialPort1.Write(txtTransmit.Text & vbCr)
    '텍스트 박스안의 문자는 아스키 코드로 시리얼 포트에 전송
    ' the carriage return (Enter Key) 가 추가

End Sub
```

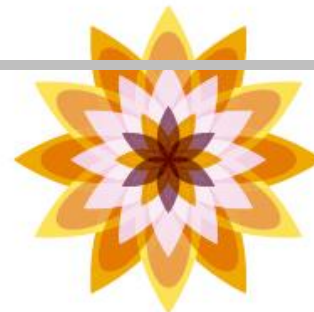
## 2. 프로그램 동작확인

---

- 아직 수신관련 코드가 작성되지 않았기 때문에 동작을 확인할 수는 없다.

*Chapter 05*

# 시리얼 데이터 수신 설정



# 목차

1. 소스 추가
2. 프로그램 동작 확인
3. 배포용 실행파일 만들기



# 1. 소스추가

- 앞장에서 추가했던 “SerialPort1” 컨트롤의 수신 이벤트 함수를 찾아 아래와 같이 소스코드를 추가한다.

```
Private Sub SerialPort1_DataReceived(sender As Object, e As IO.Ports.SerialDataReceivedEventArgs)  
Handles SerialPort1.DataReceived  
    '시리얼 포트 데이터 수신 시 발생 이벤트  
    ReceivedText(SerialPort1.ReadExisting()) '시리얼 데이터가 수신될 때마다 자동으로 호출된다.  
  
End Sub
```

# 1. 소스추가

- 컴파일을 하게 되면 오류가 발생할 것입니다.
- 먼저 위에서 추가한 소스코드에서 사용한 함수의 실제 코드를 정의한다.

```
Private Sub ReceivedText(ByVal [text] As String)
    'compares the ID of the creating Thread to the ID of the calling Thread
    If Me.rtbReceived.InvokeRequired Then
        Dim x As New SetTextCallback(AddressOf ReceivedText)
        Me.Invoke(x, New Object() {(text)})
    Else
        Me.rtbReceived.Text &= [text]
    End If
End Sub
```

# 1. 소스추가

- 위에서 정의한 함수 내의 **SetTextCallback** 함수를 최상위 클래스에서 선언한다.

```
Public Class Form_Monitoring_Program_Main
```

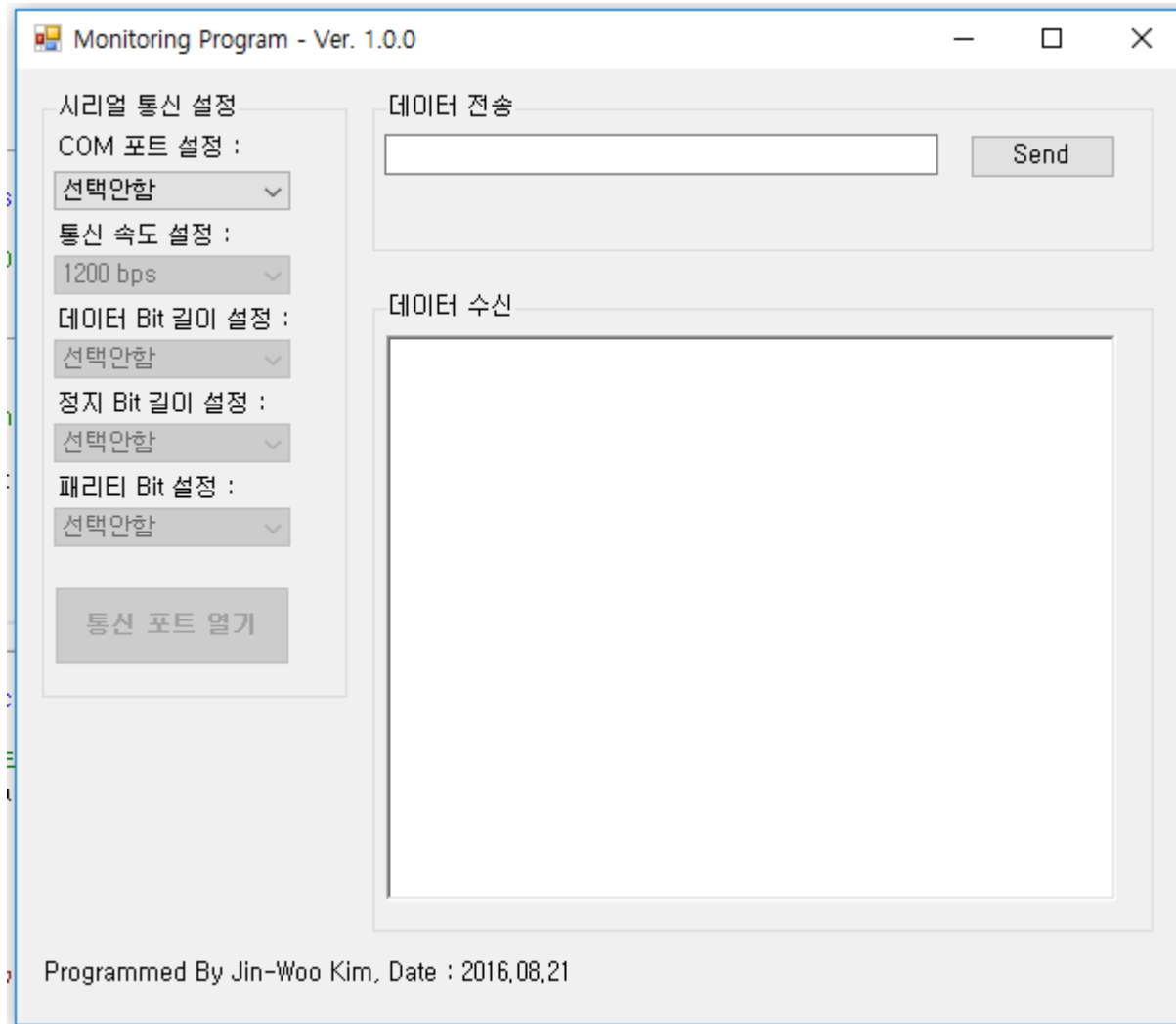
```
Dim Val_Button_Serial_Port_On_Off_Toggle As Integer = 0
```

```
Delegate Sub SetTextCallback(ByVal [text] As String) '데이터 수신동안 쓰레드 에러를 방지
```

- 델리게이트를 사용하는 이유가 시리얼 통신에 있어서 데이터수신이벤트(DataReceived)가 다른 스레드에서 일어나는 일이기 때문에 메인에서 만들어진 텍스트박스의 속성, 즉 text에 접근하지 못하기 때문입니다.
- 여기서의 델리게이트는 이벤트처리자로서 기능 즉 데이터수신이벤트처리자로 기능하는 것은 아닙니다.
- 여기서의 기능은 리치텍스트 박스의 Text를 업데이트시키는 서브루틴인 ReceivedText를 대리하는 겁니다.

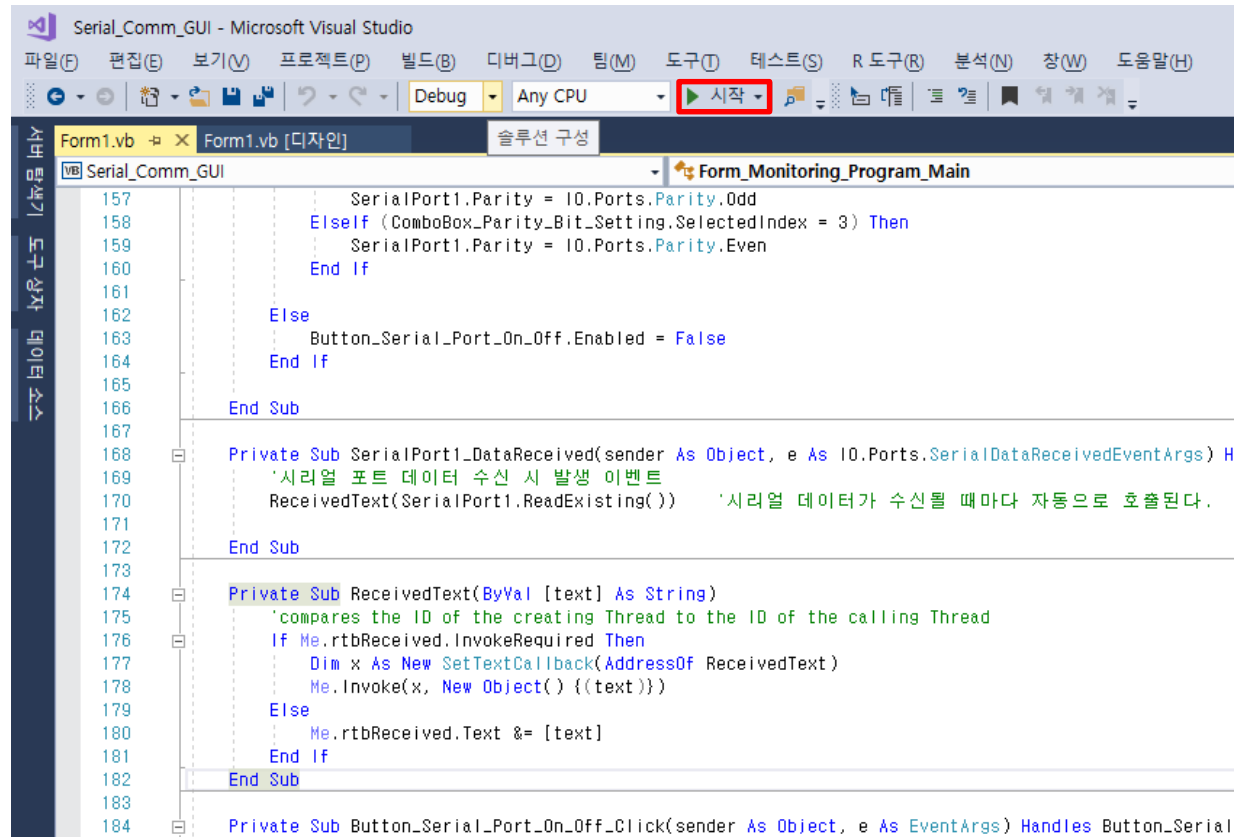
## 2. 프로그램 동작확인

- 컴파일하면 다음과 같은 화면이 출력됩니다.



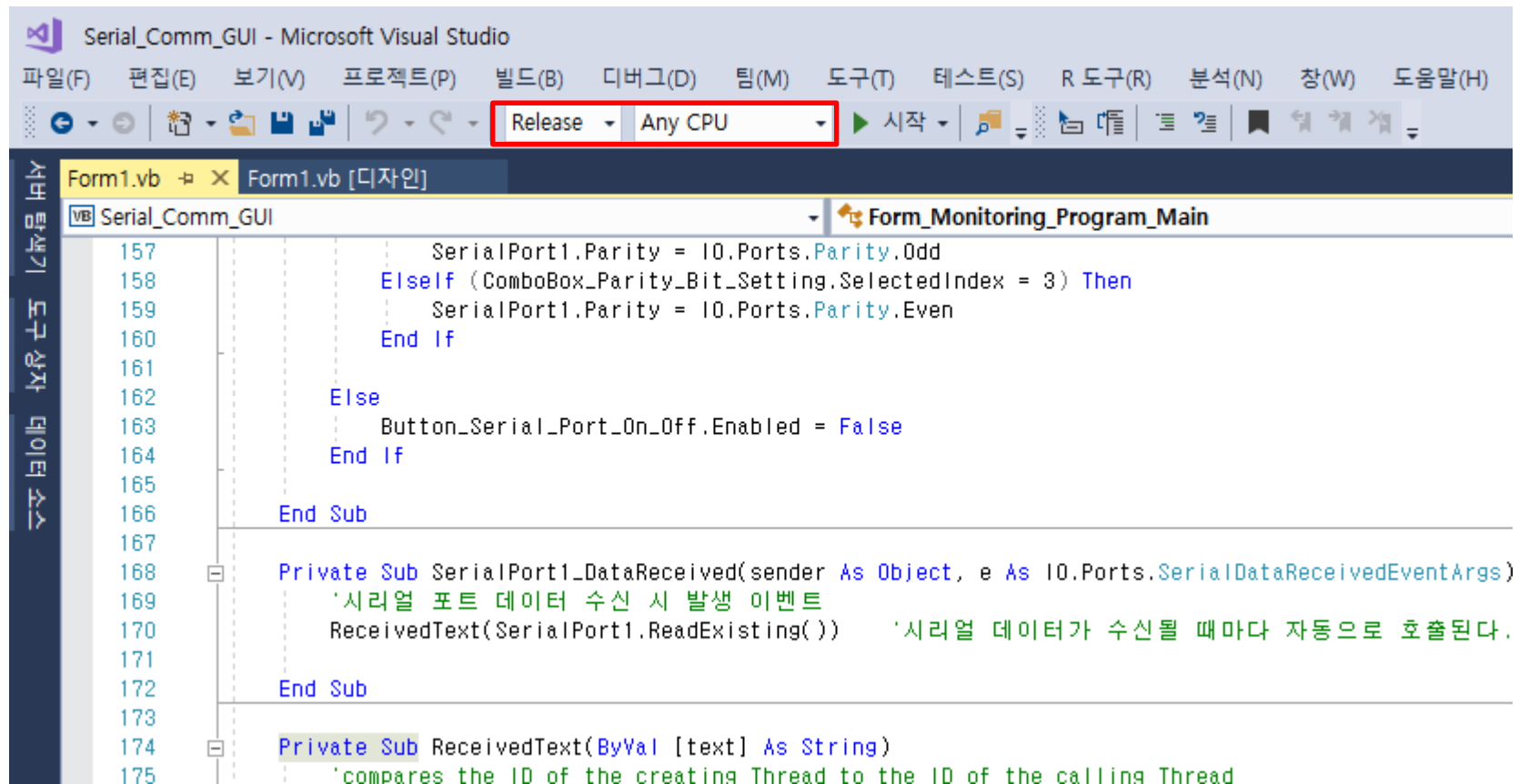
### 3. 배포용 실행파일 만들기

- 이제 최종적으로 배포할 수 있는 형태의 실행파일인 EXE 파일을 만들어보도록 하겠습니다.
- 방법은 매우 간단하기 때문에 굳이 설명이라고 할 것도 없습니다.
- 우선 Visual Studio 프로그램 상단 중앙에 있는 시작 버튼을 클릭하여 프로그램을 실행합니다.



### 3. 배포용 실행파일 만들기

- 프로그램이 실행되면 실행된 프로그램을 종료하고 아래의 그림과 같이 디버깅 시작 버튼 옆의 "Debug"로 되어 있던 콤보박스를 "Release"로 변경하고, 바로 옆의 콤보박스에 "Any CPU"로 변경되어 있는지 확인합니다.



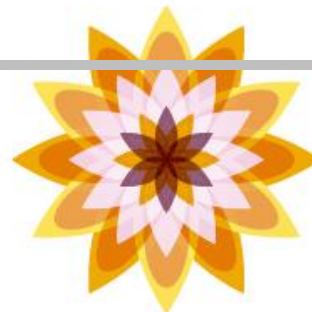
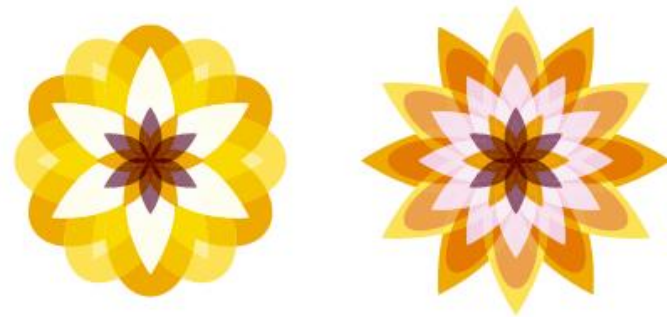
### 3. 배포용 실행파일 만들기

---

- 그런 다음 다시 Visual Studio 프로그램 상단 중앙에 있는 "시작" 버튼을 클릭하여 프로그램을 실행한 다음 종료합니다.
- 상기 절차를 모두 완료하였으면, 프로젝트가 저장되어 있는 경로의 bin 디렉토리 내의 Release 디렉토리에 실행파일이 생성되어 있는 것을 확인할 수 있습니다.
- 이제 실행파일만 따로 복사하여 배포를 하면 됩니다.

*Chapter 07*

# 소켓 통신프로그램





*Chapter 13*

# 채팅 프로그램



# 목차

1. 네트워크의 개요
2. 채팅 프로그램 설계하기
3. 채팅 프로그램 화면 디자인 및 소켓 클래스 추가하기
4. 채팅 프로그램 코드 작성하기
5. 채팅 프로그램 실행하기

# 1. 네트워크의 개요

## ■ 네트워크의 개념과 통신 방법

### ■ 네트워크

- 컴퓨터와 컴퓨터가 서로 데이터를 주고받을 수 있도록 유·무선으로 연결되어 있는 것

### ■ 인터넷

- 네트워크와 네트워크가 서로 연결되어 수천수만 대의 컴퓨터가 연결되어 있는 것

### ■ 프로토콜

- 어떻게 통신할지 정의한 규칙

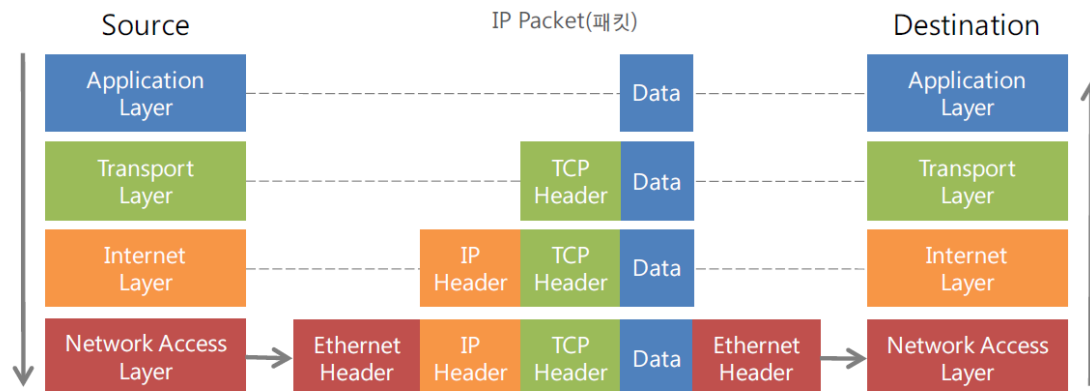
### ■ 인코딩

- 문자나 기호를 통신에 사용할 목적으로 부호화하는 것

### ■ 디코딩

- 인코딩된 데이터를 원래의 문자나 기호로 변환하는 것

그림 13-1 컴퓨터 간 통신 방법



# 1. 네트워크의 개요

## ■ 네트워크 모델

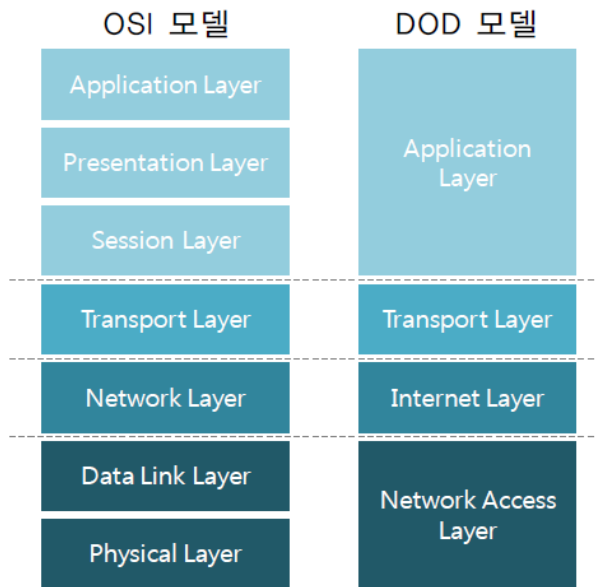
- 네트워크를 구성하는 방식은 데이터 관리 주체에 두 가지로 나뉜다.
- 서버/클라이언트 방식
  - 서버(서비스 제공)라는 컴퓨터에서 모든 데이터를 관리하고 클라이언트(서비스 요청)라는 다른 컴퓨터들은 서버가 제공하는 서비스를 통해 데이터를 사용한다.
- 피어투피어 방식
  - 전용 서버 없이 모든 컴퓨터가 각자 데이터를 관리하고 필요할 때 네트워크를 통해 공유한다. 따라서 각자가 특정한 시점에 서버일 수도 있고 클라이언트일 수도 있다.

# 1. 네트워크의 개요

## ■ 네트워크 계층 모델

- 네트워크를 연결하는 데 필요한 기능을 영역별, 단계별로 규정해 놓은 것
- OSI 7계층 모델
  - 국제표준화기구에서 제정한 개방형 시스템 상호 연결모델
- DOD 4계층 모델
  - 미국 국방성에서 제정
  - OSI 모델을 압축하여 4계층으로 만든 것
  - 인터넷을 연결하는 데 사용되며, 데이터 전송과 주소 지정을 위해 TCP/IP 프로토콜을 사용한다.

그림 13-2 OSI 7 모델과 DOD 모델의 비교



# 1. 네트워크의 개요

## ■ 프로토콜

- 네트워크상에서 서로 데이터를 주고받으려는 컴퓨터들 사이에 언제, 무엇을, 어떻게 통신할 것인지를 정의한 통신 규약
- 데이터를 주는 쪽과 받는 쪽이 반드시 같은 프로토콜을 사용해야 한다.
- 인터넷에서는 TCP와 UDP라는 프로토콜을 사용할 수 있다.
- **TCP 프로토콜**
  - 데이터를 송수신하기 위해 미리 연결을 설정하고 데이터에 순차적인 번호를 부여하여 목적지로 전송하는 연결 지향형 프로토콜
  - 동기 방식
- **UDP 프로토콜**
  - 미리 연결을 설정하지 않고 필요시 데이터를 송신하며, 수신지에서 데이터를 잘 받았는지 확인하지 않는 비연결 지향형 프로토콜
  - 비동기 방식

# 1. 네트워크의 개요

## ■ IP 주소, 포트 번호

### ■ IP 주소

- 인터넷에 연결된 모든 컴퓨터에 지정된 고유한 숫자
- 주소 체계는 계층적 구조로 이뤄져 있으며 앞부분은 네트워크 주소를, 뒷부분은 노드(컴퓨터) 주소를 나타낸다.

### ■ 포트 번호

- IP 주소를 통해 수신된 데이터가 어떤 서비스의 데이터인지 구분하기 위한 번호

표 13-1 포트 번호별 분류

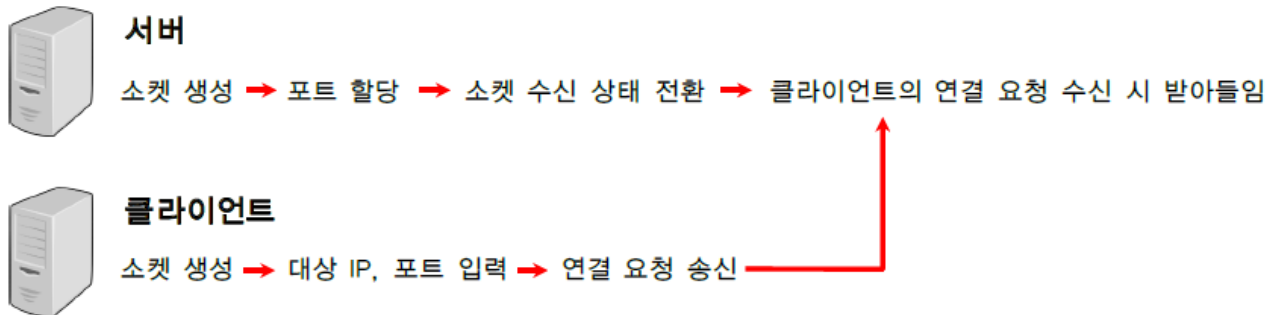
포트 번호	분류
HTTP	80
FTP	20, 21
Telnet	23
POP3	110
IMAP	143
SNMP	162

# 1. 네트워크의 개요

## ■ 윈도 소켓

- 컴퓨터 간 네트워크 통신을 할 때 사용되는 접점
- TCP/IP 계층의 응용 계층과 전송 계층 사이에서 네트워크 프로그램을 간단하게 만들 수 있도록 도와주는 역할
- 닷넷(.Net) 환경에서는 System.Net.Sockets 클래스를 프로그램에 삽입하면 표준 API를 사용할 수 있다.

그림 13-3 소켓을 사용한 네트워크 프로그래밍 과정





# 1. 네트워크의 개요

---

## ■ 멀티태스킹

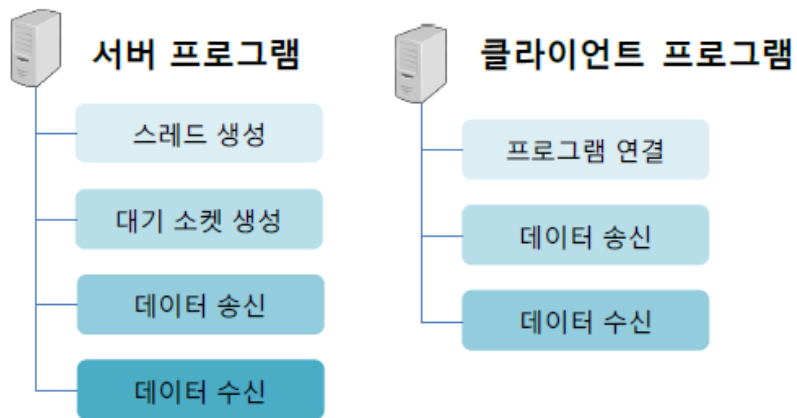
- 여러 개의 작업을 동시에 처리하는 것
- 멀티프로세싱
  - 하나의 응용 프로그램을 여러 개의 프로세스로 구성(여러 개 실행)하여 각 프로세스(실행된 한 프로그램)가 하나의 작업을 처리하도록 하는 방법
- 멀티스레드
  - 응용 프로그램을 여러 개의 스레드로 구성하고 각 스레드가 하나의 태스크를 처리하도록 하는 방법

## 2. 소켓 통신 프로그램 설계하기

### ■ 소켓 통신 프로그램의 기능

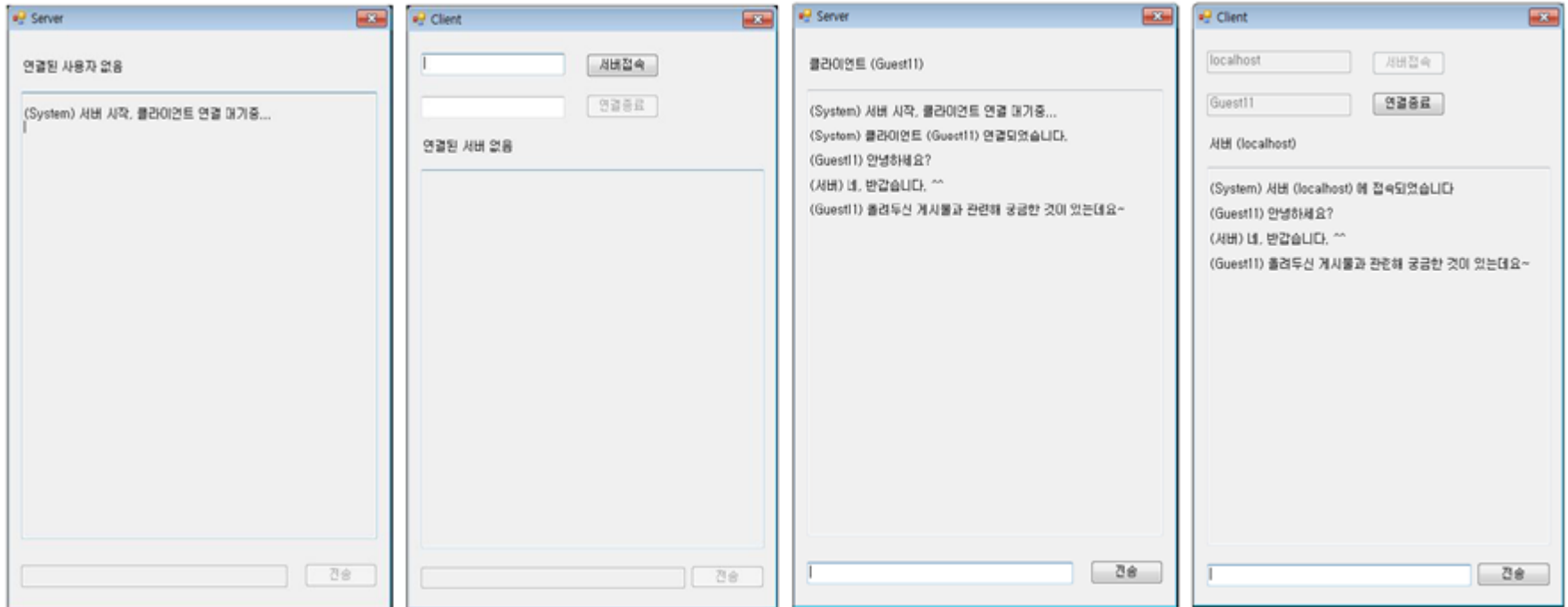
- 서버에서 클라이언트의 접속을 대기하고 있다가 클라이언트가 접속 요청을 하면 허가하고 통신을 시작하여 메시지를 주고받는다.
- 클라이언트가 접속을 해제하면 서버는 자신에게 저장된 클라이언트의 정보를 지운다.

그림 13-4 프로그램별 구현 기능



## 2. 소켓 통신 프로그램 설계하기

### ■ 소켓 통신 프로그램 최종 화면

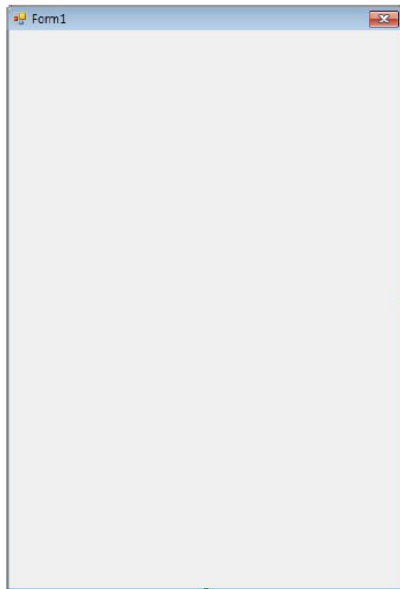


### 3. 서버 프로그램 화면 디자인 및 소켓 클래스 추가하기

#### ■ 서버 프로그램

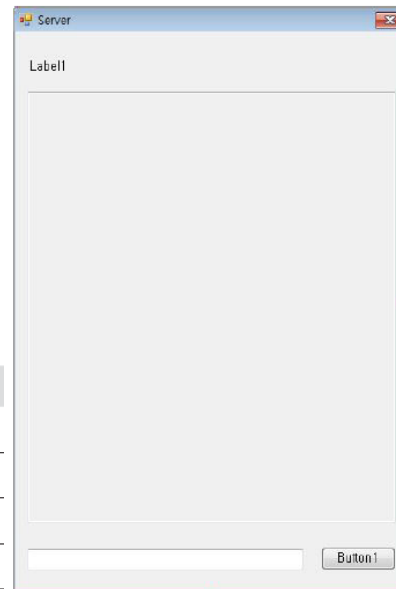
##### ■ 속성 설정

그림 13-6 서버 폼 속성 설정



컨트롤	속성	속성값
Form1	Text	Server
	FormBorderStyle	FixedSingle
	MaximizeBox	False
	MinimizeBox	False

그림 13-7 서버 폼 컨트롤 추가 및 속성 설정

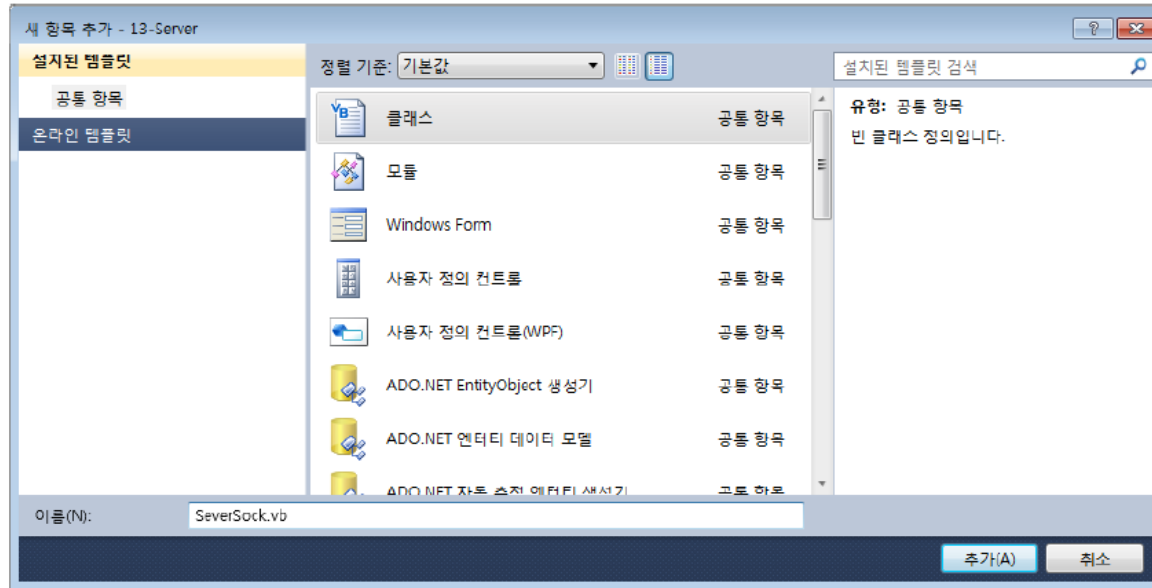


컨트롤	속성	속성값
Label1	(Name)	Label1
TextBox1	Multiline	True
	ReadOnly	True
TextBox2	Enabled	False
Button1	Enabled	False
	Text	전송

### 3. 서버 프로그램 화면 디자인 및 소켓 클래스 추가하기

- [프로젝트] - [클래스 추가]를 클릭하고 [클래스]를 선택한다. 클래스 이름으로 ServerSock.vb를 입력한다.

그림 13-8 서버 소켓 클래스 추가



## 4. 서버 프로그램 코드 작성하기

### ■ 서버 소켓 클래스 코드 작성하기

- 소켓 통신을 하기 위한 라이브러리를 추가한다.

```
Imports System.Net.Sockets '소켓 라이브러리
Imports System.IO '동기 및 비동기 읽기/쓰기를 가능하게 하는 형식들을 포함하는 라이브러리
Imports System.Text '수신된 데이터를 인코딩하기 위한 텍스트 라이브러리
```

## 4. 서버 프로그램 코드 작성하기

### ■ 서버 소켓 클래스 코드 작성하기

- 소켓 통신을 하기 위해 추가한 서버 소켓 클래스에 다음과 같이 변수와 함수를 선언한다.

```
Public Class ServerSock
    Private clientSock As TcpClient
    Private Buffer(b_size) As Byte
    Const b_size As Integer = 1024
    Private connName As String '연결된 사용자 이름
    Private sendWriter As StreamWriter '데이터 송신 스트림
    Private encoderKor As Encoding '한글 인코더
```

## 4. 서버 프로그램 코드 작성하기

### ■ 서버 소켓 클래스 코드 작성하기

- 서버 소켓 클래스에 다음과 같이 소켓을 생성하기 위한 함수를 추가한다.

```
Public Sub New(ByVal client As TcpClient) 'ServerSock을 생성하면 이루어지는 작업
    Me.clientSock = client '접속한 클라이언트 소켓을 현재 소켓에 저장한다
    '데이터 전송시 한글을 인코딩하여 전송하기 위한 설정
    Me.encoderKor = Encoding.GetEncoding(949) '한글로 인코딩한다
    Me.sendWriter = New StreamWriter(clientSock.GetStream, Me.encoderKor)
    '비동기 읽기를 준비한다. 버퍼만큼의 데이터를 받아내고 읽기가 끝나면
    'ReceiveData라는 프로시저에 따라 처리한다.
    Me.clientSock.GetStream.BeginRead(Buffer, 0, b_size, AddressOf ReceiveData, Nothing)
End Sub
```



## 4. 서버 프로그램 코드 작성하기

### ■ 서버 소켓 클래스 코드 작성하기

- 서버 소켓 클래스에 다음과 같이 수신데이터를 처리하기 위한 함수를 추가한다.

```
Public Sub ReceiveData(ByVal iar As IAsyncResult) '수신 데이터 처리
    Dim data_size As Integer ' 수신된 데이터의 크기
    Dim data As String ' 수신된 데이터
    Try
        SyncLock clientSock.GetStream
            ' 읽기의 끝부분을 데이터의 크기로 산정한다.
            data_size = clientSock.GetStream.EndRead(iar) End SyncLock

            '한국어로 변환 후 맨끝의 변환문자를 삭제
            data = encoderKor.GetString(Buffer, 0, data_size - 1)
            '이벤트를 발생시켜 메인폼에서 데이터가 처리되게 한다.
            RaiseEvent ReceiveProc(Me, data)

            '여러개의 실행 스레드가 동시에 같은 문장을 실행하는 것을 방지한다.
            SyncLock clientSock.GetStream
                clientSock.GetStream.BeginRead(Buffer, 0, b_size, AddressOf ReceiveData, Nothing)
            End SyncLock

        Catch ex As Exception
        End Try
    End Sub
```

## 4. 서버 프로그램 코드 작성하기

### ■ 서버 소켓 클래스 코드 작성하기

- 서버 소켓 클래스에 다음과 같이 데이터를 송신 처리하기 위한 함수를 추가한다.

```
Public Sub SendData(ByVal data As String) '데이터 송신 처리
    SyncLock clientSock.GetStream
        sendWriter.Write(data & vbNewLine) '송신 스트림에 데이터를 기록한다.
        sendWriter.Flush() '버퍼의 내용을 지우면서 데이터를 내부 스트림에 기록한다.
    End SyncLock
End Sub
```

#### ■ SyncLock

- SyncLock은 여러 스레드가 같은 코드를 동시에 실행하지 못하게 하는 명령문이다.
- 스레드가 SyncLock에 도착하면 다른 스레드에서 잠금을 풀기 전까지 기다린다.
- 같은 코드를 동시에 실행하면 결과가 섞여서 의도치 않은 결과가 나올 수 있기 때문이다.

## 4. 서버 프로그램 코드 작성하기

### ■ 서버 소켓 클래스 코드 작성하기

- 서버 소켓 클래스에 다음과 같이 접속한 클라이언트의 이름을 설정하거나 반환하는 함수를 추가한다.

'접속한 클라이언트의 이름을 설정하거나 반환해주는 Getter 겸 Setter

```
Public Property Name() As String
```

```
    Get
```

```
        Return connName
```

```
    End Get
```

```
    Set(ByVal value As String)
```

```
        connName = value
```

```
    End Set
```

```
End Property
```

- 서버 소켓 클래스에 다음과 같이 메인 폼에서 데이터를 처리하는데 필요한 이벤트를 할당하는 함수를 추가한다.

```
Public Event ReceiveProc(ByVal sender As ServerSock, ByVal data As String)
```

```
End Class
```

## 4. 서버 프로그램 코드 작성하기

### ■ 서버 폼 코드 작성하기

- 서버 프로그램의 메인 폼을 더블클릭하여 코드 편집 창을 열고 다음과 같이 코드를 작성한다.
- 메인 폼 최상단에 다음과 같이 소켓 라이브러리와 스레드 라이브러리를 추가한다.

```
Imports System.Net.Sockets '소켓 라이브러리  
Imports System.Threading '스레드 라이브러리
```

## 4. 서버 프로그램 코드 작성하기

### ■ 서버 폼 코드 작성하기

- 메인 폼 최상위 클래스에 다음과 같이 변수와 함수를 선언한다.

```
Public Class Form1
    Private tcplisten As TcpListener '연결을 확인하는 클라이언트
    Private listenthread As Thread '연결을 대기하는 스레드
    Const port_n As Integer = 8080 '서버가 연결을 받을 포트
    Dim clientSock As ServerSock '대기에 사용할 소켓
    Dim isConn As Boolean = False
```

## 4. 서버 프로그램 코드 작성하기

### ■ 서버 폼 코드 작성하기

- 메인 폼 클래스에 다음과 같이 연결 대기를 시작하는 함수를 정의한다.

```
Private Sub initListen() '연결대기 시작
    '스레드를 처리할 프로시저를 지정하면서 스레드 생성
    listenthread = New Threading.Thread(AddressOf actListen)
    listenthread.Start() ' 스레드를 시작

    Output("(System) 서버 시작, 클라이언트 연결 대기중...")
    Label1.Text = "연결된 사용자 없음"
End Sub
```

## 4. 서버 프로그램 코드 작성하기

### ■ 서버 폼 코드 작성하기

- 메인 폼 클래스에 다음과 같이 연결 신청이 들어왔을 때 처리하는 함수를 정의한다.

```
Private Sub actListen() '스레드에 연결작업이 들어왔을시 수행하는 프로시저
    Try
        tcplisten = New TcpListener(port_n) '대기할 소켓을 만든다
        tcplisten.Start() '대기할 소켓을 대기상태로 시작한다
        Do
            '대기할 소켓에 클라이언트의 소켓이 도착하면 서버소켓을 생성해 도착한 소켓을 저장한다.
            clientSock = New ServerSock(tcplisten.AcceptTcpClient)
            '저장된 소켓에 있는 데이터를 수신하기 위한 프로시저를 지정한다.
            AddHandler clientSock.ReceiveProc, AddressOf Receiving
        Loop Until False
    Catch ex As Exception

    End Try
End Sub
```

## 4. 서버 프로그램 코드 작성하기

### ■ 서버 폼 코드 작성하기

- 메인 폼 클래스에 다음과 같이 헤더 설정하는 코드를 정의한다.

'헤더 설정 각번호에 따라 패킷의 종류를 구분한다. 0=연결 1=연결종료 2=데이터  
Public Enum Header  
    Connect = 0  
    Disconn = 1  
    Data = 2  
End Enum

- Enum문

- 서로 연관관계가 있는 정수형 상수를 묶어서 리스트를 정의하고자 할 때 사용한다.



## 4. 서버 프로그램 코드 작성하기

### ■ 서버 폼 코드 작성하기

- 수신된 패킷을 분해하여 처리하는 함수를 정의한다.

'수신된 데이터를 분해하여 데이터의 유형을 파악해 각각 처리 프로시저로 보낸다.

```
Private Sub Receiving(ByVal sender As ServerSock, ByVal data As String)
```

```
    Dim h_type As Integer
```

```
    h_type = CInt(data.Substring(0, 1)) '헤더를 잘라내어 변수에 저장한다.
```

```
    data = data.Substring(1) '헤더를 제외한 나머지 데이터를 변수에 재저장한다.
```

```
    ' 헤더 값에 따라 데이터를 처리한다.
```

```
    Select Case h_type
```

```
        Case 0      ' 헤더가 0이면 클라이언트에게 연결 패킷을 보낸다.
```

```
            Connect(data, sender)
```

```
        Case 1      ' 헤더가 1이면 클라이언트에게 연결 종료 패킷을 보낸다.
```

```
            Disconnect(sender)
```

```
        Case 2      ' 헤더가 2이면 수신한 데이터를 TextBox에 출력한다.
```

```
            Output("(" & sender.Name & ") " & data)
```

```
    End Select
```

```
End Sub
```

## 4. 서버 프로그램 코드 작성하기

### ■ 서버 폼 코드 작성하기

- 클라이언트에게 연결 데이터를 전송하는 함수를 정의한다.

```
Private Sub Connect(ByVal name As String, ByVal sender As ServerSock)
    ' 데이터를 보낸 클라이언트의 이름을 폼의 사용자명 변수에 저장한다.
    sender.Name = name
    Output( " (System) 클라이언트 ( " & name & " ) 연결되었습니다. " )
    Label1.Text = " 클라이언트 ( " & name & " ) "
    Dim h_type As Integer
    h_type = Header.Connect
    ' 연결 헤더를 포함한 데이터를 보내라고 회신을 처리하는 프로시저에 전달한다.
    SendtoSender(h_type.ToString, sender)
    isConn = True
    TextBox2.Enabled = True
    Button1.Enabled = True
End Sub
```

## 4. 서버 프로그램 코드 작성하기

### ■ 서버 폼 코드 작성하기

- 서버에서 전달받은 데이터를 클라이언트에게 회신하는 함수를 정의한다.

```
Private Sub SendtoSender(ByVal data As String, ByVal sender As ServerSock)
    sender.SendData(data)
End Sub
```

## 4. 서버 프로그램 코드 작성하기

### ■ 서버 폼 코드 작성하기

- 연결 종료 패킷을 전송하는 함수를 정의한다.

```
Private Sub Disconnect(ByVal sender As ServerSock)
    Dim h_type As Integer
    ' 연결 종료 헤더를 포함한 데이터를 클라이언트에게 전송한다.
    h_type = Header.Disconn
    Try
        SendtoSender(h_type.ToString, sender)
    Catch ex As Exception

    End Try

    Output("(System) 클라이언트의 연결이 종료되었습니다")
    Label1.Text = "연결된 사용자 없음"
    isConn = False
    TextBox2.Enabled = False
    Button1.Enabled = False
End Sub
```

## 4. 서버 프로그램 코드 작성하기

### ■ 서버 폼 코드 작성하기

- 수신데이터를 TextBox1에 출력하는 함수를 정의한다.

```
Private Sub Output(ByVal msg As String)
    TextBox1.Text = TextBox1.Text & vbNewLine
    TextBox1.AppendText(msg & vbNewLine)
End Sub
```

## 4. 서버 프로그램 코드 작성하기

### ■ 서버 폼 코드 작성하기

- 메인폼을 불러와서 서버를 시작하는 함수를 정의한다.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)  
    Handles MyBase.Load  
        initListen()  
  
End Sub
```

## 4. 서버 프로그램 코드 작성하기

### ■ 서버 폼 코드 작성하기

- 메인 폼을 종료하는 함수를 정의한다.

```
' 폼을 종료하는데 클라이언트가 연결된 상태라면  
' 연결 종료 패킷을 클라이언트에게 보낸 후 프로그램을 종료한다.  
Private Sub Form1_FormClosing(ByVal sender As Object, ByVal e As  
System.Windows.Forms.FormClosingEventArgs) Handles Me.FormClosing  
    If isConn = True Then  
        Disconnect(clientSock)  
    End If  
    tcplisten.Stop()  
End Sub
```

## 4. 서버 프로그램 코드 작성하기

### ■ 서버 폼 코드 작성하기

- 데이터 전송 버튼이 클릭될 때 동작하는 함수를 정의한다.

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
    If TextBox2.Text = "" Then
    Else
        Dim data As String
        Dim h_type As Integer
        ' 데이터를 출력하는 헤더를 포함한 패킷을 클라이언트에게 전송한다.
        h_type = Header.Data
        data = h_type & " (서버) " & TextBox2.Text
        clientSock.SendData(data)
        ' 수신한 데이터의 헤더를 제외하고 TextBox에 출력한다.
        Output(data.Substring(1))
        TextBox2.Text = ""
        TextBox2.Focus()
    End If
End Sub
```



## 4. 서버 프로그램 코드 작성하기

### ■ 서버 폼 코드 작성하기

- TextBox2에서 Enter키를 누르면 입력된 메시지를 전송하는 함수를 정의한다.

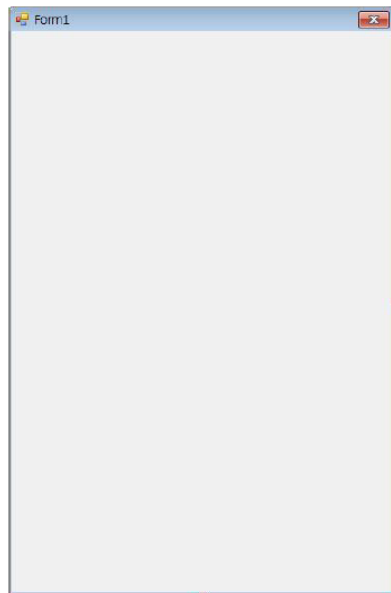
```
Private Sub TextBox2_KeyDown(ByVal sender As Object, ByVal e As  
System.Windows.Forms.KeyEventArgs) Handles TextBox2.KeyDown  
    If e.KeyCode = Keys.Enter Then  
        Button1.PerformClick()  
    End If  
End Sub  
End Class
```

# 5. 클라이언트 프로그램 화면 디자인 및 소켓 클래스 추가하기

## ■ 클라이언트 프로그램

### ■ 속성 설정

그림 13-9 클라이언트 폼 속성 설정



컨트롤	속성	속성값
Form1	Text	Client
	FormBorderStyle	FixedSingle
	MaximizeBox	False
	MinimizeBox	False

그림 13-10 클라이언트 폼 컨트롤 추가 및 속성 설정

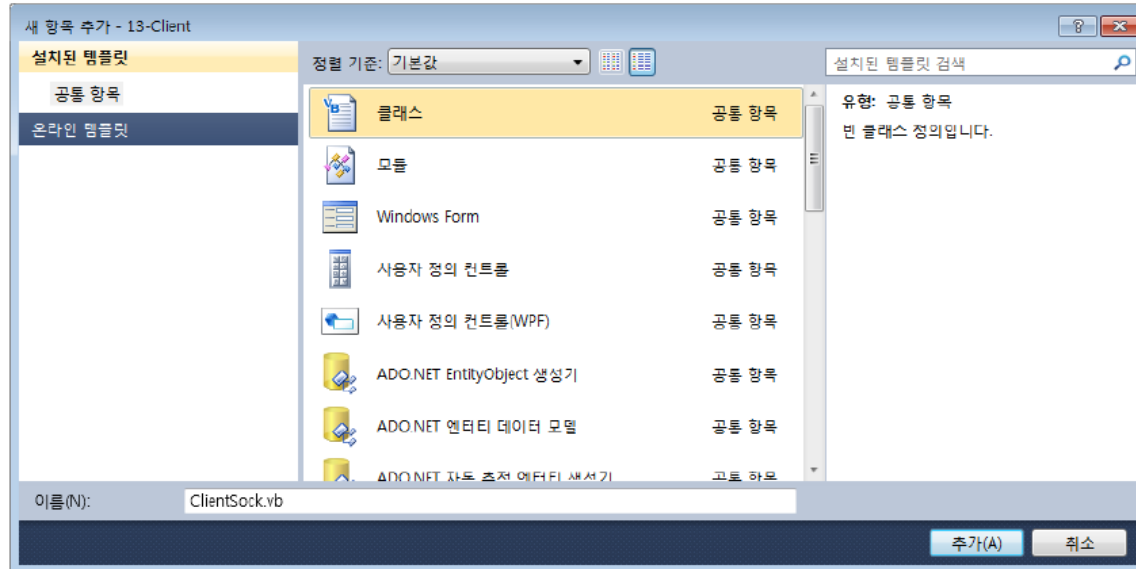


컨트롤	속성	속성값
Label1	(Name)	Label1
TextBox1	(Name)	TextBox1
TextBox2	(Name)	TextBox2
TextBox3	Multiline	True
	ReadOnly	True
TextBox4	Enabled	False
Button1	Text	서버 접속
Button2	Enabled	False
	Text	연결 종료
Button3	Enabled	False
	Text	전송

## 5. 클라이언트 프로그램 화면 디자인 및 소켓 클래스 추가하기

- [프로젝트] - [클래스 추가]를 클릭하고 [클래스]를 선택한다. 클래스 이름으로 ClientSock.vb를 입력한다.

그림 13-11 클라이언트 소켓 클래스 추가



## 6. 클라이언트 프로그램 코드 작성하기

### ■ 클라이언트 소켓 클래스 코드 작성하기

- 기능 구현에 필요한 클래스를 추가한다.

```
Imports System.Net.Sockets  
Imports System.IO  
Imports System.Text
```

## 6. 클라이언트 프로그램 코드 작성하기

### ■ 클라이언트 소켓 클래스 코드 작성하기

- TCP 네트워크 서비스를 제공하는 클래스를 상속받고, 기능 구현에 필요한 변수를 선언한다.

```
Public Class ClientSock  
    Inherits TcpClient  
  
    Const b_size As Integer = 1024  
    Private buffer(b_size) As Byte  
    Private userName As String  
    Private sendWriter As StreamWriter  
    Private encoderKor As Encoding
```

- 상속받은 클래스이므로 생성 시 기능을 오버라이딩한다.

```
Public Sub New()  
  
    End Sub
```

## 6. 클라이언트 프로그램 코드 작성하기

### ■ 클라이언트 소켓 클래스 코드 작성하기

- 클라이언트 소켓을 소멸시키는 함수를 정의한다.

```
Public Sub Del()  
    If Me.Active = True Then '연결되었을시  
        Me.Dispose(True) '소켓을 삭제한다. 자기자신  
    End If  
End Sub
```

- 이벤트를 선언하여 데이터를 메인 폼에서 처리하게 하는 함수를 정의한다.

```
Public Event ReceiveProc(ByVal sender As ClientSock, ByVal data As String)
```

## 6. 클라이언트 프로그램 코드 작성하기

### ■ 클라이언트 소켓 클래스 코드 작성하기

- 서버와 연결하고 그 결과를 Boolean 값으로 반환하는 함수를 정의한다.

```
Public Function ConnectServer(ByVal ip_n As String, ByVal port_n As Integer) As Boolean
    Try
        Me.Connect(ip_n, port_n) '서버에 접속한다
    Catch ex As Exception '예외 발생시
        Return False '거짓을 반환
    End Try
    Return True '연결 완료시 참을 반환
End Function
```

- 한글로 변환해주는 인코더를 생성하고 데이터를 읽을 준비를 하는 함수를 정의한다.

```
Public Sub Initiate()
    Me.encoderKor = Encoding.GetEncoding(949) '한글로 인코딩하게 준비시킨다.
    Me.sendWriter = New StreamWriter(Me.GetStream, Me.encoderKor)
    Me.GetStream.BeginRead(buffer, 0, b_size, AddressOf ReceiveData, Nothing)
End Sub
```

## 6. 클라이언트 프로그램 코드 작성하기

### ■ 클라이언트 소켓 클래스 코드 작성하기

- 데이터를 전송하는 함수를 정의한다.

```
Public Sub SendData(ByVal data As String)
    Try
        sendWriter.Write(data & vbCr)
        sendWriter.Flush()
    Catch ex As Exception

    End Try
End Sub
```



## 6. 클라이언트 프로그램 코드 작성하기

### ■ 클라이언트 소켓 클래스 코드 작성하기

- 데이터를 수신하는 함수를 정의한다.

```
Public Sub ReceiveData(ByVal iar As IAsyncResult)
    Dim data_size As Integer
    Dim data As String

    Try
        SyncLock Me.GetStream
            data_size = Me.GetStream.EndRead(iar)
        End SyncLock

        data = encoderKor.GetString(buffer, 0, data_size - 1)
        ' 메인 폼에서 데이터를 처리하기 위한 이벤트를 발생시킨다.
        RaiseEvent ReceiveProc(Me, data)

        SyncLock Me.GetStream
            Me.GetStream.BeginRead(buffer, 0, b_size, AddressOf ReceiveData, Nothing)
        End SyncLock
    Catch ex As Exception

    End Try
End Sub
End Class
```

## 6. 클라이언트 프로그램 코드 작성하기

### ■ 클라이언트 폼 코드 작성하기

- 클라이언트 프로그램의 메인 폼을 더블클릭하여 코드 편집창을 열고 다음과 같이 코드를 작성한다.
- 우선 기능 구현에 필요한 클래스를 추가한다.

```
Imports System.Net.Sockets  
Imports System.IO  
Imports System.Text
```

- 메인 폼 클래스에서 다음과 같이 기능 구현에 필요한 변수를 선언한다.

```
Public Class Form1  
    Private client As ClientSock  
    Private userName As String  
    Private servAddr As String  
    Const port_n As Integer = 8080  
    Public IsConn As Boolean = False
```

## 6. 클라이언트 프로그램 코드 작성하기

### ■ 클라이언트 폼 코드 작성하기

- 헤더 유형을 정의하는 코드를 추가한다.

```
Public Enum Header  
    Connect = 0  
    Disconn = 1  
    Data = 2  
End Enum
```

- 수신된 메시지를 TextBox3에 출력하는 함수를 정의한다.

```
Private Sub Output(ByVal msg As String)  
    TextBox3.Text = TextBox3.Text & vbNewLine  
    TextBox3.AppendText(msg & vbNewLine)  
End Sub
```

## 6. 클라이언트 프로그램 코드 작성하기

### ■ 클라이언트 폼 코드 작성하기

- 서버에 접속하기 위한 함수를 추가한다.

```
Private Function ServerConnect() As Boolean
    Try
        client = New ClientSock()          ' 클라이언트 측 소켓 객체를 생성한다.
        ' 연결이 성공하면
        If client.ConnectServer(servAddr, port_n) Then
            client.Initiate()
            ' ReceiveProc 이벤트가 발생하여 Receiving 프로시저가 실행된다.
            AddHandler client.ReceiveProc, AddressOf Receiving

            Dim data As String
            Dim h_type As Integer = Header.Connect
            ' 보낼 메시지(헤더 + 이름)를 만든다.
            data = h_type.ToString & userName
            ' 클라이언트가 연결되었음을 알리는 메시지를 서버에 전송한다.
            client.SendData(data)
```

## 6. 클라이언트 프로그램 코드 작성하기

### ■ 클라이언트 폼 코드 작성하기

- 서버에 접속하기 위한 함수를 추가한다.

```
' 연결에 실패하면
Else
    MsgBox("서버와의 연결에 실패하였습니다", vbOKOnly, "에러")
    ' 클라이언트 소켓을 소멸시킨다.
    client.Del()
    ' Try 문에 False 값을 반환하여 Catch 문으로 흐름을 이동시킨다.
    Return False
End If
Catch

End Try
' 서버와 연결되면 연결 여부를 표시하는 변수 IsConn에 True를 할당한다.
IsConn = True
Return True
End Function
```

## 6. 클라이언트 프로그램 코드 작성하기

### ■ 클라이언트 폼 코드 작성하기

- 서버에 접속을 해제하기 위한 함수를 추가한다.

```
Private Sub Disconnect()  
    If IsConn = True Then      ' 서버와 연결된 상태라면  
        Dim h_type As Integer = Header.Disconn  
        client.SendData(h_type.ToString) '접속해제 메시지를 서버에게 전송하고  
        IsConn = False        ' 서버 연결 상태를 표시하는 변수를 False로 설정한다.  
  
        client.Close()         ' 클라이언트 소켓을 닫는다.  
        client.Del()           ' 클라이언트 소켓을 소멸시킨다.  
    End If  
End Sub
```

## 6. 클라이언트 프로그램 코드 작성하기

### ■ 클라이언트 폼 코드 작성하기

- 수신 메시지에서 헤더를 분리하는 함수를 추가한다.

```
Private Sub Receiving(ByVal sender As ClientSock, ByVal data As String)
    Dim h_type As Integer
    h_type = CInt(data.Substring(0, 1)) ' 메시지에서 헤더를 분리하여 저장한다.
    data = data.Substring(1)           ' 헤더가 분리된 데이터를 저장한다.
    ' 분리된 헤더 값에 따라 서로 다른 메시지를 표시한다.
    Select Case h_type
        Case 0
            Output("(System) 서버 (" & servAddr & ") 에 접속되었습니다.")
        Case 1
            Output("(System) 서버와의 연결이 종료되었습니다.")
            ControlSelector("disconn")
        Case 2
            Output(data)
    End Select
End Sub
```

## 6. 클라이언트 프로그램 코드 작성하기

### ■ 클라이언트 폼 코드 작성하기

- 연결상태에 따라 컨트롤을 활성화여부를 설정하는 함수를 추가한다.

```
Private Sub ControlSelector(ByVal mode As String)

    Select Case mode
        Case "conn"
            Button1.Enabled = False
            Button2.Enabled = True
            Button3.Enabled = True
            TextBox1.Enabled = False
            TextBox2.Enabled = False
            TextBox4.Enabled = True
            Label1.Text = "서버 (" & servAddr & ")"
            TextBox4.Focus()
```



## 6. 클라이언트 프로그램 코드 작성하기

### ■ 클라이언트 폼 코드 작성하기

- 서버에 접속을 해제하기 위한 함수를 추가한다.

```
Case "disconn"  
    Button1.Enabled = True  
    Button2.Enabled = False  
    Button3.Enabled = False  
    TextBox1.Enabled = True  
    TextBox2.Enabled = True  
    TextBox4.Enabled = False  
    Label1.Text = "연결된 서버 없음"  
    TextBox1.Focus()  
End Select  
End Sub
```

## 6. 클라이언트 프로그램 코드 작성하기

### ■ 클라이언트 폼 코드 작성하기

- “서버 접속” 버튼을 구현하기 위한 함수를 추가한다.

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles Button1.Click

    servAddr = TextBox1.Text           ' TextBox1에 입력된 서버 주소로 접속하고,
    username = TextBox2.Text           ' TextBox2에 입력된 값을 사용자 이름으로 설정한다.
    ' 서버에 접속하면 데이터 전송/입력 관련 컨트롤을 활성 상태로 설정한다.
    If ServerConnect() = True Then
        ControlSelector("conn")
    End If
End Sub
```

- “연결 종료 ” 버튼을 구현하기 위한 함수를 정의한다.

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles Button2.Click
    Disconnect()                       ' 서버와 연결을 해제한다.
    ControlSelector("disconn")         ' 데이터 전송/입력 관련 컨트롤을 비활성화한다.
End Sub
```

## 6. 클라이언트 프로그램 코드 작성하기

### ■ 클라이언트 폼 코드 작성하기

- “전송” 버튼을 구현하기 위한 함수를 추가한다.

```
Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles Button3.Click
        If TextBox4.Text = "" Then                ' 입력 메시지가 없으면 아무 동작도 하지 않는다.

            Else                                  ' TextBox4에 입력된 메시지에 헤더를 추가하여 전송한다.
                Dim h_type As Integer = Header.Data
                Dim data As String
                Output("(" & userName & ") " & TextBox4.Text)
                data = h_type.ToString & TextBox4.Text
                client.SendData(data)
                TextBox4.Text = ""
                TextBox4.Focus()
            End If
        End Sub
```

## 6. 클라이언트 프로그램 코드 작성하기

## ■ 클라이언트 폼 코드 작성하기

- 메인 폼을 실행하면 서버 정보를 나타내는 레이블에 연결 서버가 없음을 표시하는 함수를 추가한다.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles MyBase.Load
        Label1.Text = "연결된 서버 없음"
    End Sub
```

- **메인 품을 종료하기 위한 함수를 정의한다.**

```
Private Sub Form1_FormClosing(ByVal sender As Object, ByVal e As  
System.Windows.Forms.FormClosingEventArgs) Handles Me.FormClosing  
    If IsConn = True Then                                ' 클라이언트가 연결상태라면  
        Dim h_type As Integer = Header.Disconn  
        client.SendData(h_type.ToString)                ' 연결종료 메시지를 전송한다.  
  
        client.Close()                                  ' 클라이언트 소켓을 닫는다.  
        client.Del()                                    ' 클라이언트 소켓을 소멸시킨다.  
    End If  
End Sub
```

## 6. 클라이언트 프로그램 코드 작성하기

### ■ 클라이언트 폼 코드 작성하기

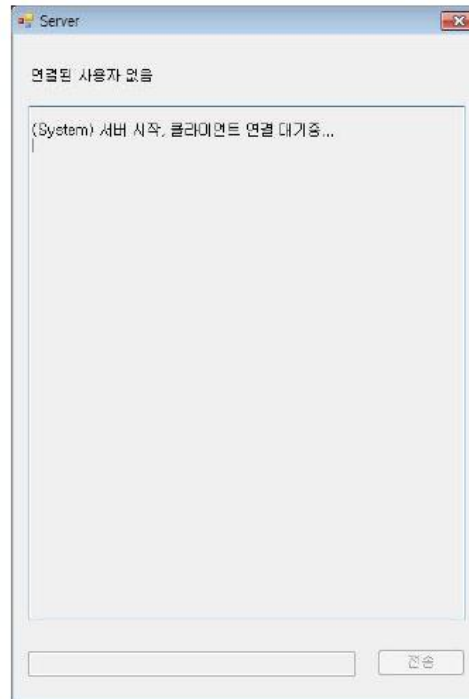
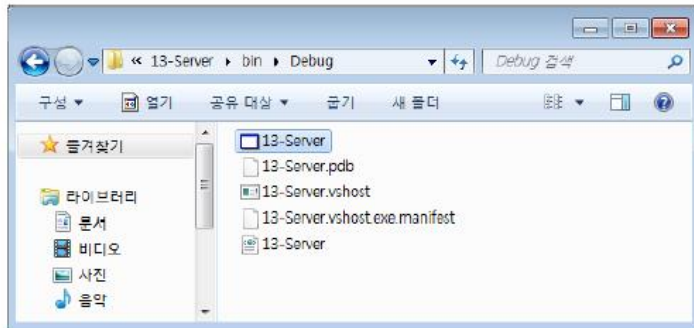
- TextBox2에서 Enter키를 누르면 데이터를 전송하는 함수를 추가한다.

```
Private Sub TextBox2_KeyDown(ByVal sender As Object, ByVal e
As System.Windows.Forms.KeyEventArgs) Handles TextBox4.KeyDown
    If e.KeyCode = Keys.Enter Then
        Button3.PerformClick()
    End If
End Sub
End Class
```

## 7. 소켓 통신 프로그램 실행하기

### ■ 서버 프로그램 실행

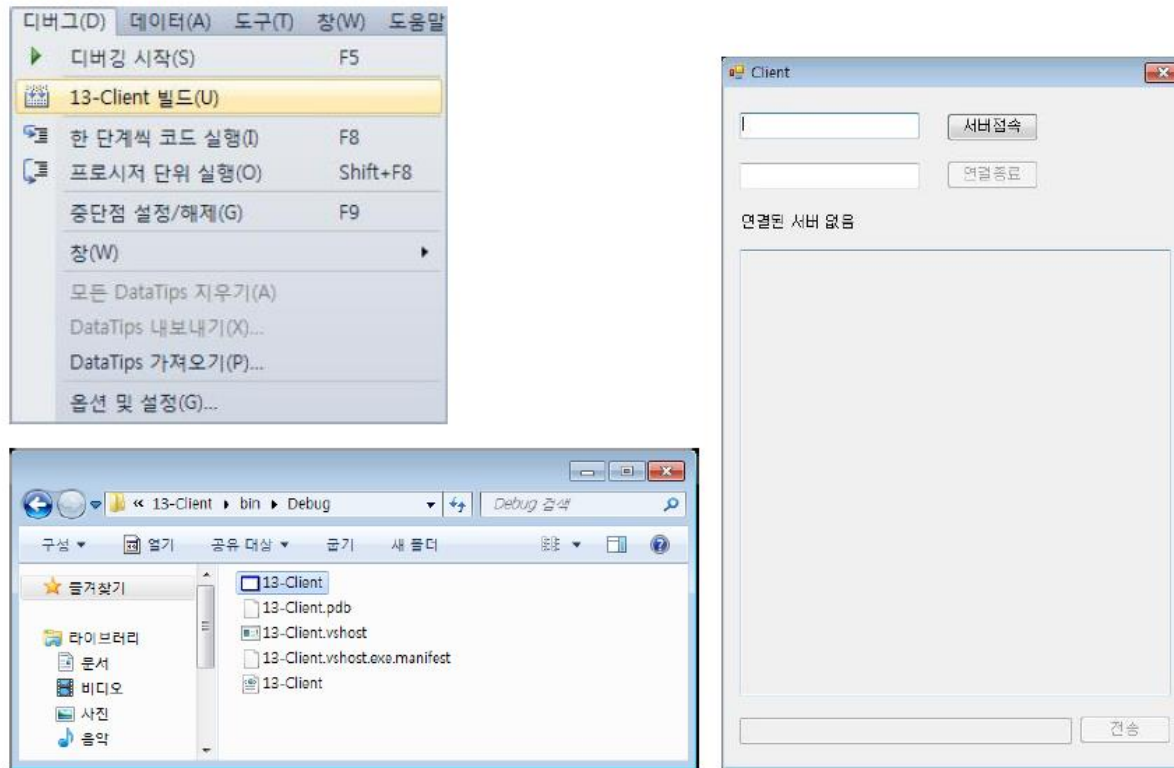
그림 13-12 서버 프로그램 실행



## 7. 소켓 통신 프로그램 실행하기

### 클라이언트 프로그램 실행

그림 13-13 클라이언트 프로그램 실행



## 7. 소켓 통신 프로그램 실행하기

### ■ 서버와 클라이언트 연결

그림 13-14 서버 접속

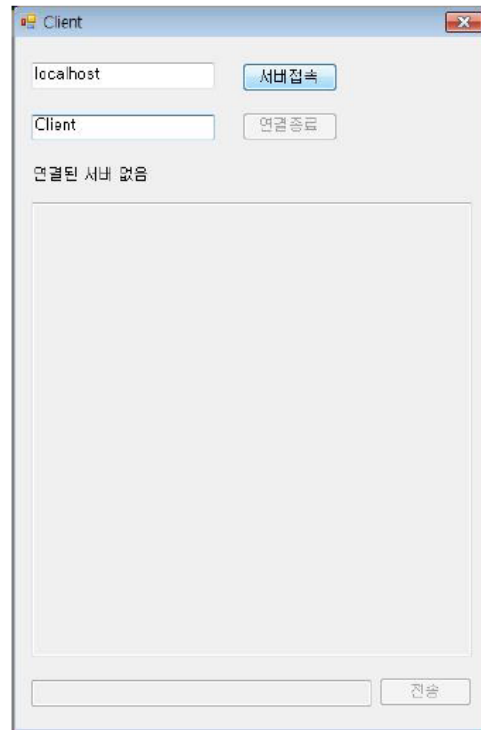
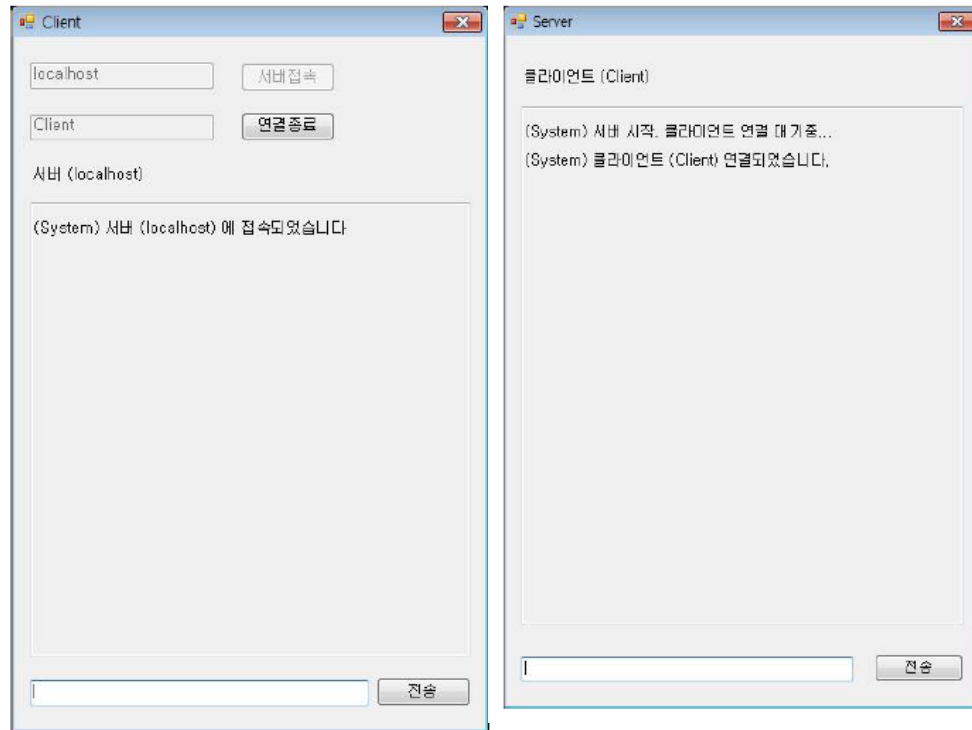


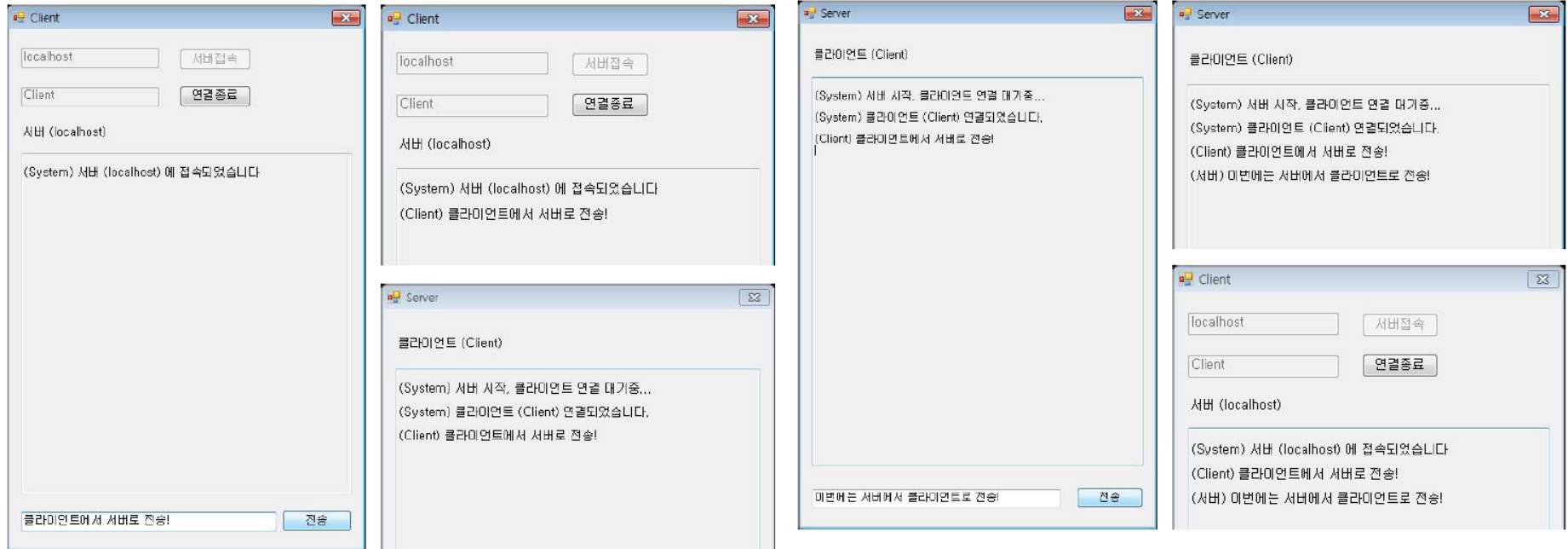
그림 13-15 클라이언트와의 연결 확인





# 7. 소켓 통신 프로그램 실행하기

그림 13-16 서버 프로그램과 클라이언트 프로그램 간의 대화 전송





Thank You

---