

미쯔비시PLC/PC통신 예제 개발

▼ 1. GX Work2 설치

GX Work2는 미쯔비시 PLC 개발 툴이다.

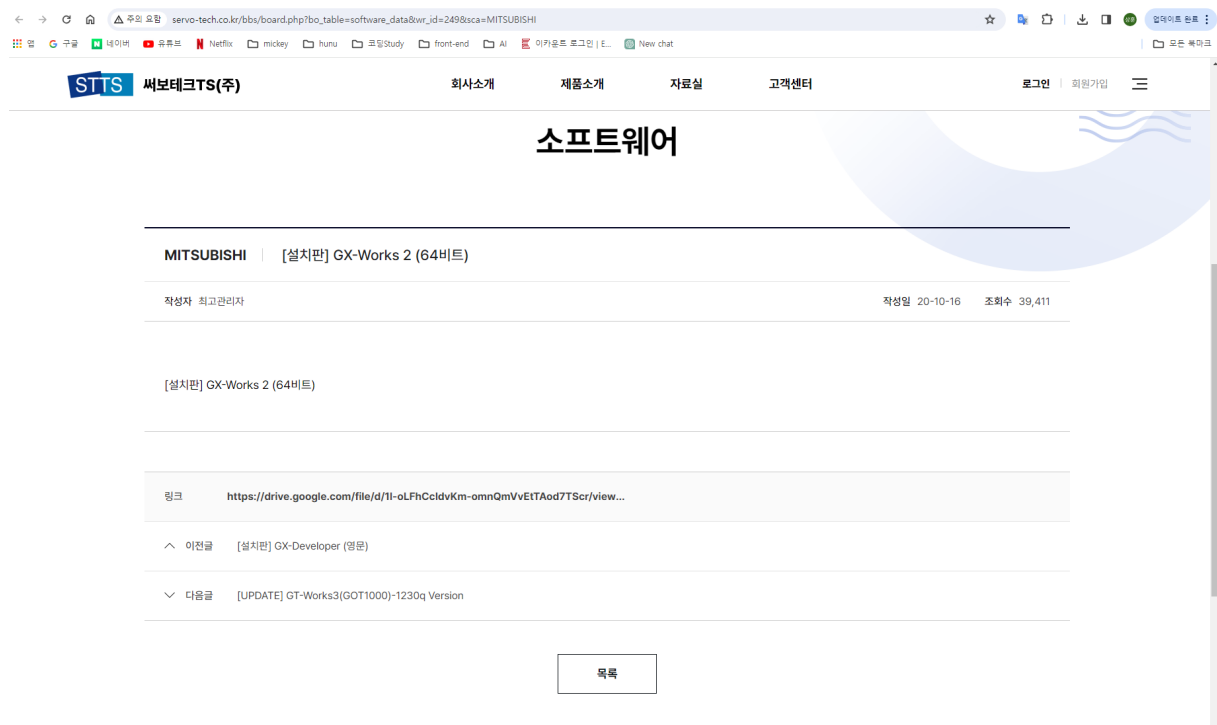
GX Work2 설치는 아래 블로그를 참고하였다.

<https://ihluck77.tistory.com/1>

1)

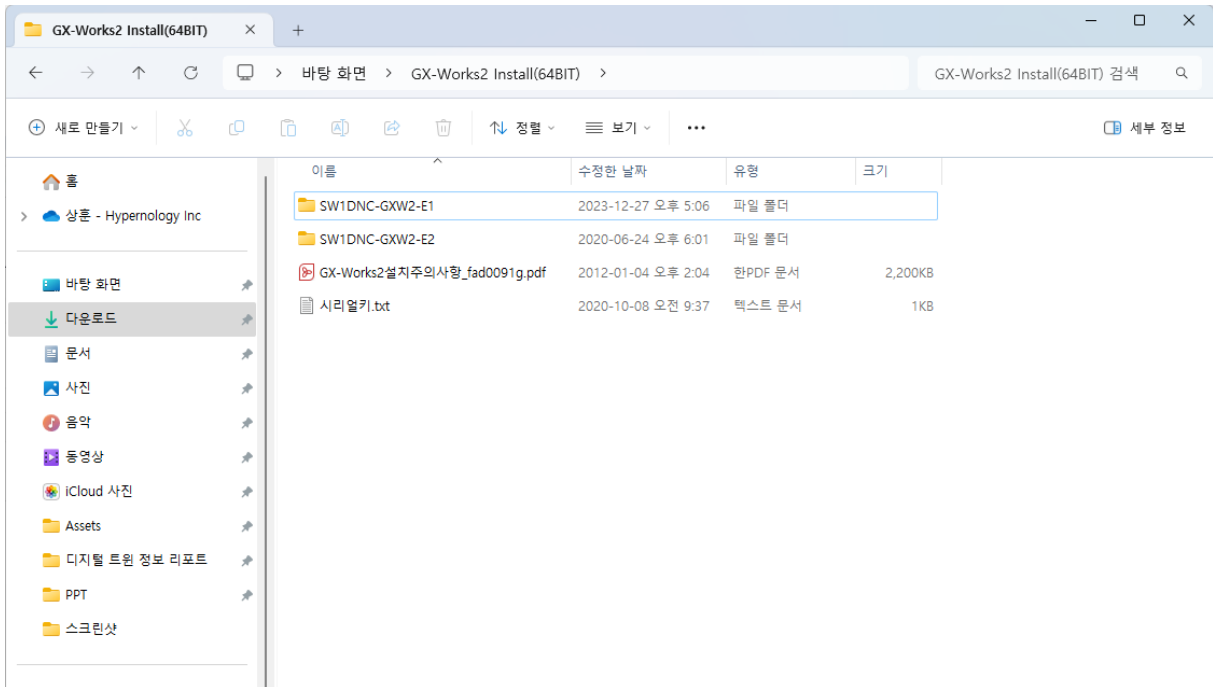
http://www.servo-tech.co.kr/bbs/board.php?bo_table=software_data&wr_id=249&sca=MITSUBISHI

위 링크로 접속하여 GX Work2 setup파일을 다운로드 한다.

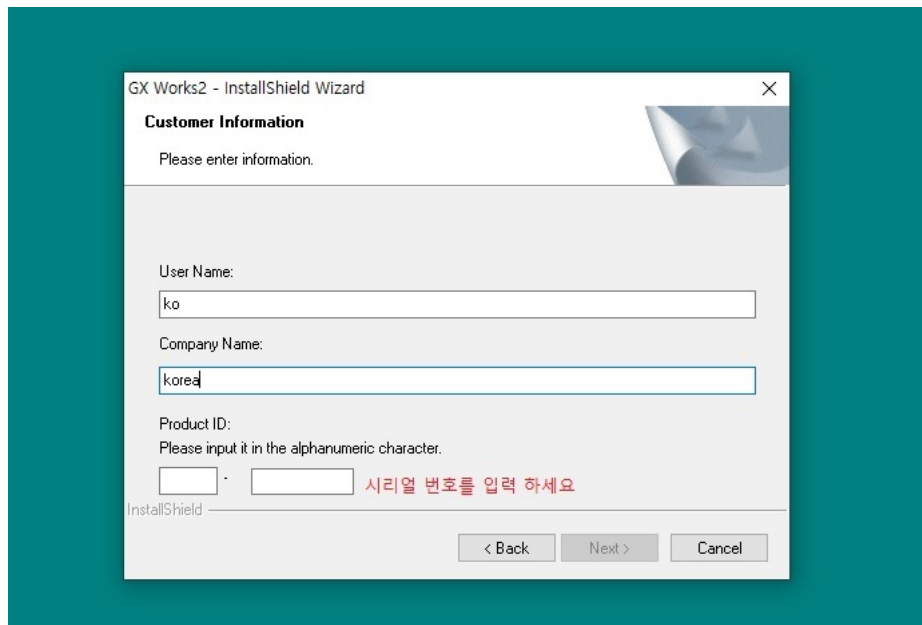


2)

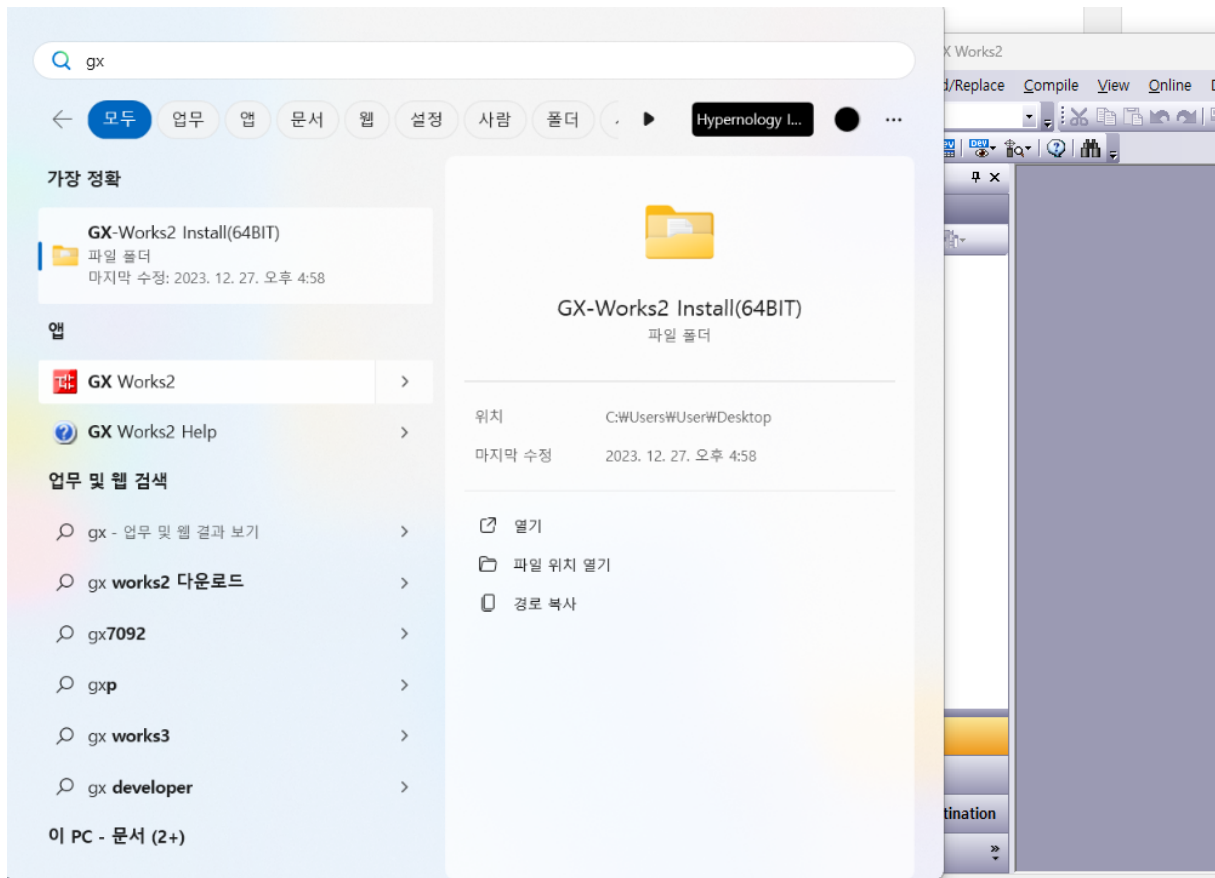
SW1DNC-GXW2-E1 폴더안에 setup.exe을 클릭하여 설치한다.



- 3)
- setup파일 실행시 아래 화면에서
- 이름
- 회사명 (필자는 none으로 기재했다.)
- 시리얼번호 (폴더안에 시리얼키.txt파일을 참고)
- 를 입력하고, 다음 화면부터는 next버튼을 클릭해주면 된다.



- 4)
- GX Works2 설치 완료

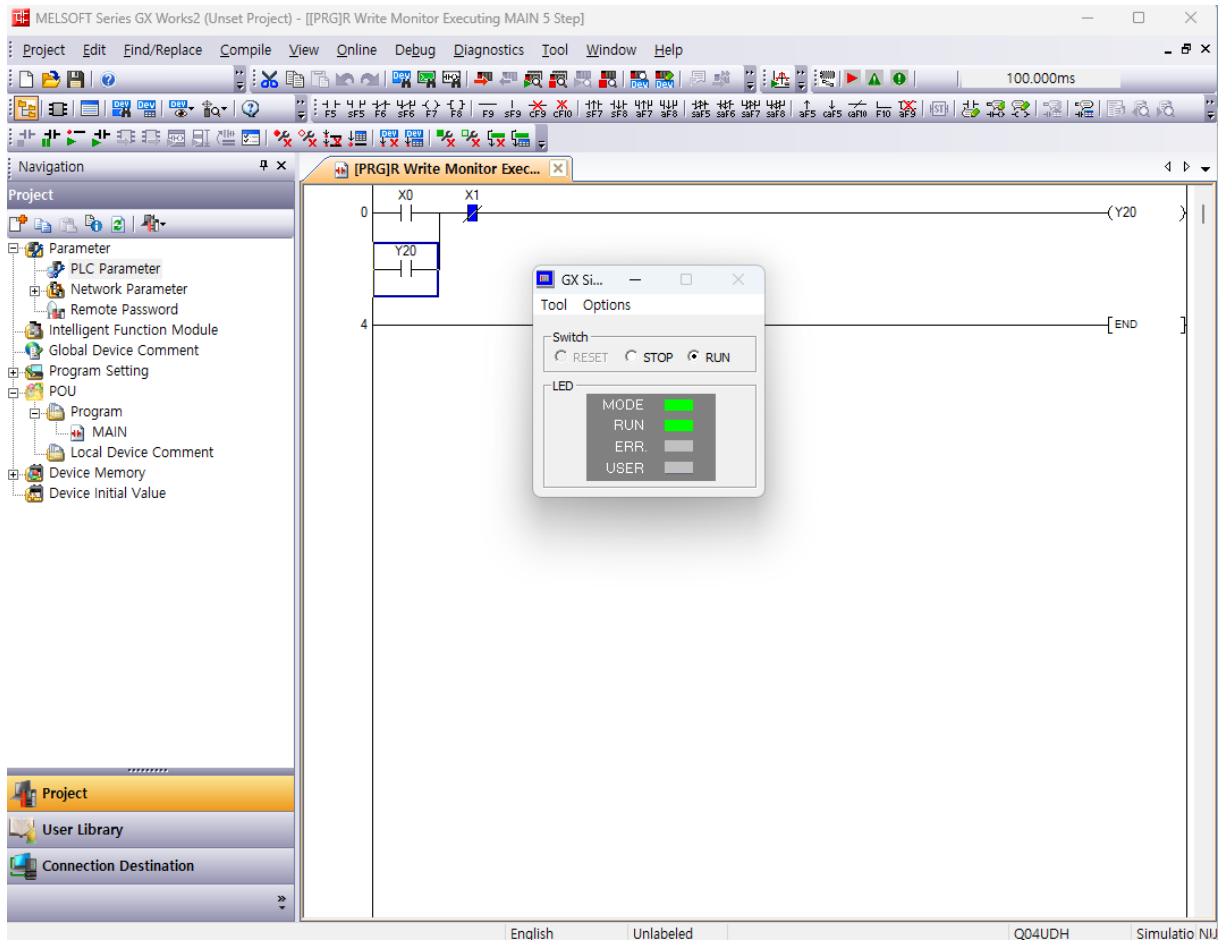


▼ 2. PLC프로젝트 코드 작성

프로그램을 실행시키고 예제 PLC코드를 작성해보았다.

아래 블로그를 참고하였으며, 매우 상세하게 설명하고 있어 추가적인 설명은 생략하도록 하겠다.

<https://abc0123.tistory.com/151>



▼ 3. MX Component 설치 및 설정

필자는 다양한 블로그로 MX Component 설치를 시도했지만, 다양한 이유로 설치가 어려웠

그냥 미쯔비시 **MX Component Ver.4 체험판/시험판 URL**을 접속하여 설치하였다.

[https://kr.mitsubishielectric.com/fa/ko/board.do?](https://kr.mitsubishielectric.com/fa/ko/board.do?act=modifyDefaultBoard&depth=1&board_id=6&pageIndex=1&category_gubun=25&ut=w&num=20381&rurl=&log_id=&category_)

[act=modifyDefaultBoard&depth=1&board_id=6&pageIndex=1&category_gubun=25&ut=w&num=20381&rurl=&log_id=&category_](https://kr.mitsubishielectric.com/fa/ko/board.do?act=modifyDefaultBoard&depth=1&board_id=6&pageIndex=1&category_gubun=25&ut=w&num=20381&rurl=&log_id=&category_)


미쯔비시 홈페이지에서 설치하려면 가입을 해야하는 귀찮음이 있지만,,, 많은 실패로 그냥 정석으로 설치를 진행하였다.

1)

파일에서 zip파일 다운로드 후 압축 해

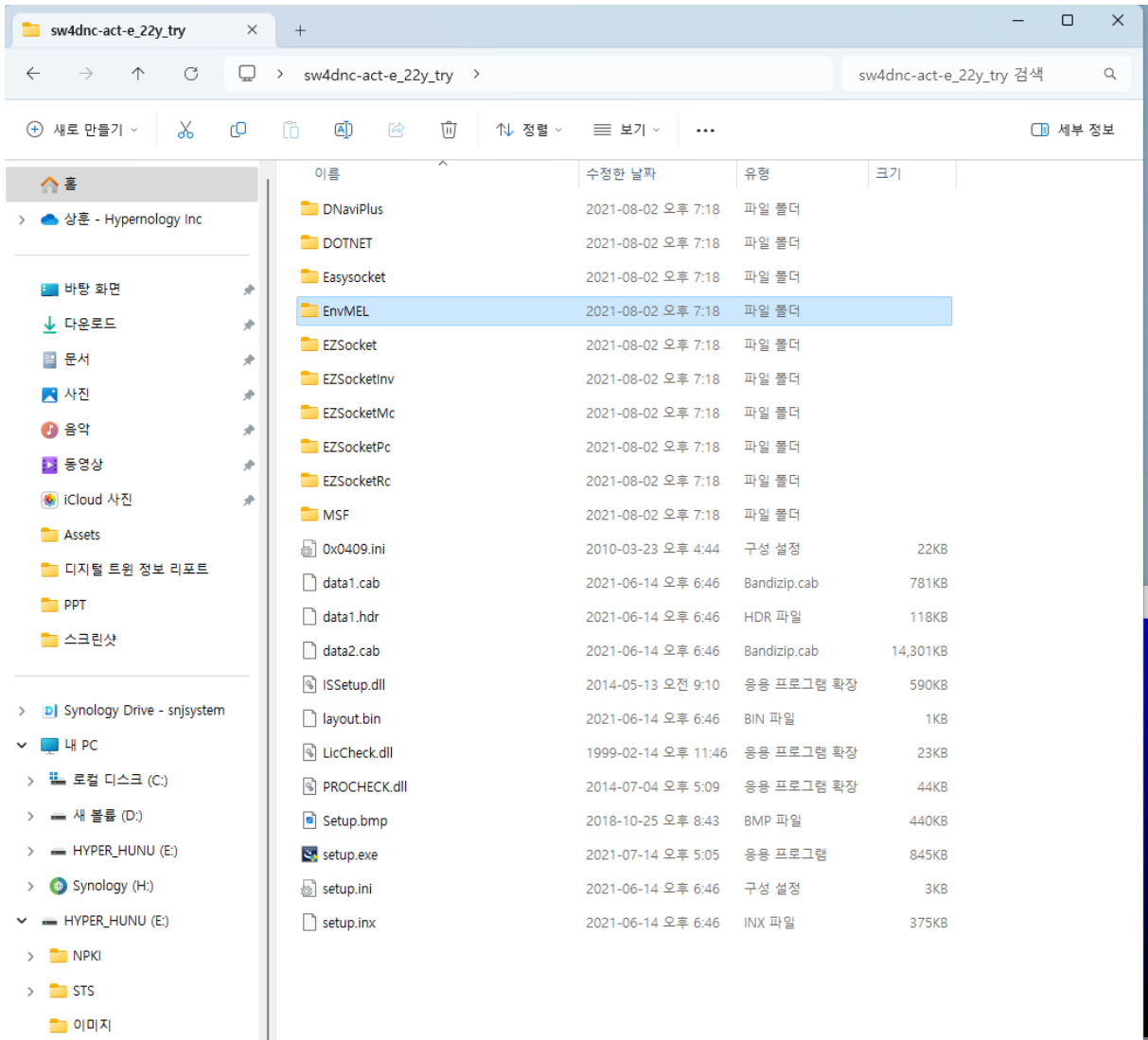
Download**MX Component Ver.4 체험판/시험판**

- 내용** 통신용 라이브러리 MX Component Ver.4의 체험판입니다.
- 소개**
- 본소프트웨어는 MX Component Ver.4(영어판)의 체험판입니다.
 - 제품판과 같은 기능을 인스톨 후 30일간 체험할 수 있습니다.
- 주의사항**
- 인스톨 후 30일이 지나면 실행 할 수 없습니다.
 - 체험판에서 제품판으로 버전 업그레이드 등은 지원하지 않습니다.
 - 제품판(이전의 버전을 포함)이 설치 되어 있는 경우는 반드시 제품판을 삭제바랍니다.
 - 체험판 설치 후, 제품판을 설치하는 경우에는 반드시 체험판을 삭제바랍니다.
 - 체험판 설치 후에 버전 업그레이드판 또는 업데이트판을 덮어쓰기 할 수 없습니다.

파일  sw4dnc-act-e_22y_try.zip

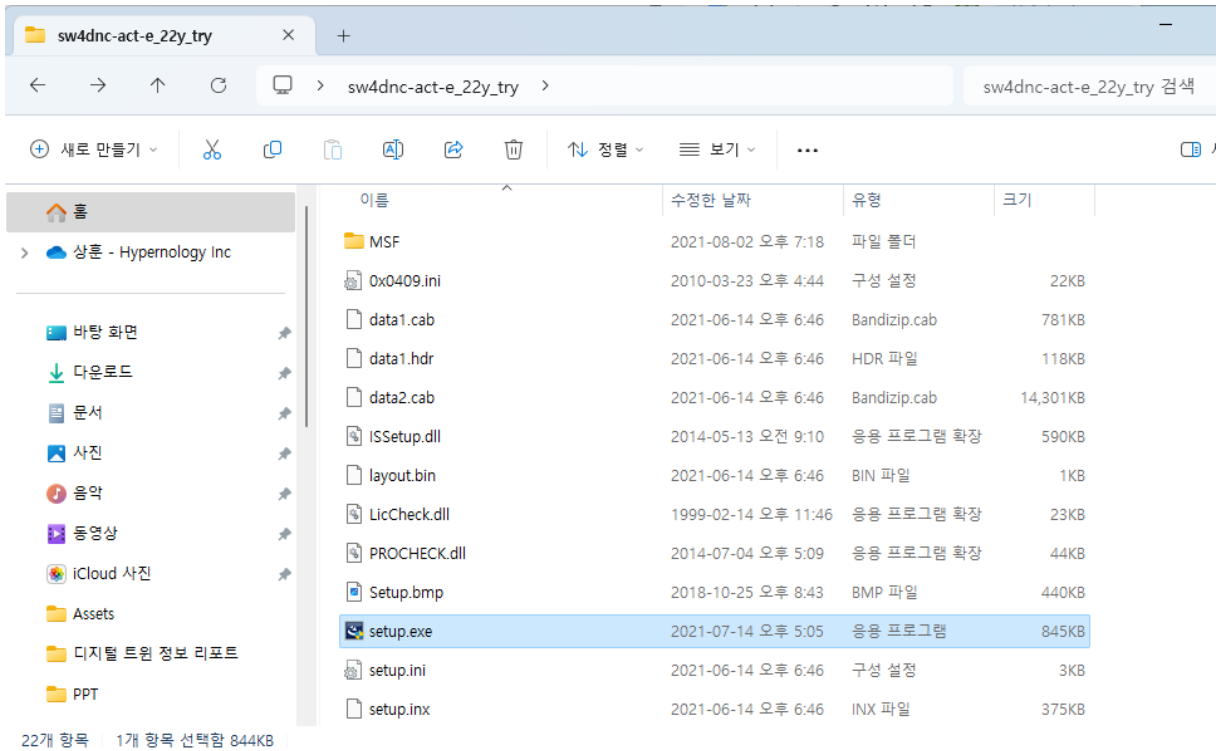
2)

먼저 EnvMEL 폴더안에 setup.exe 파일을 "관리자권한으로 실행"하여 설치한다.



3)

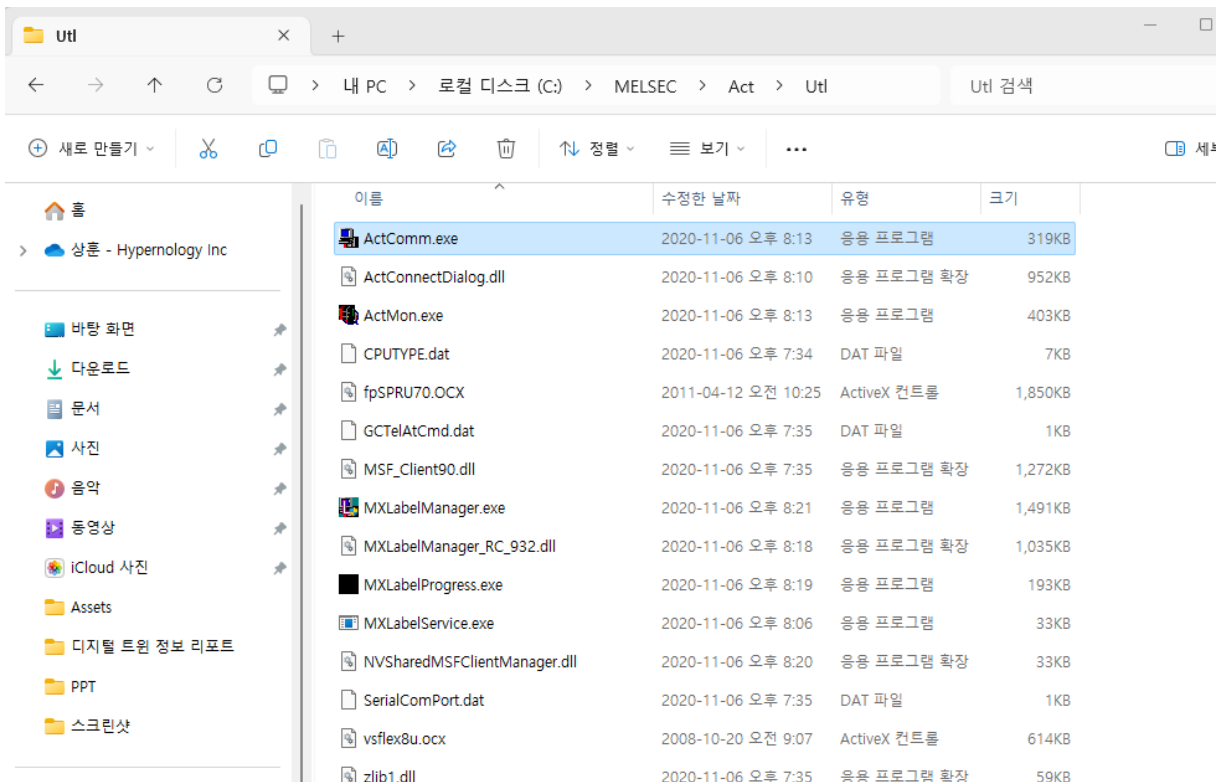
다음으로 최상위 폴더의 setup.exe파일을 같은 방식으로 설치한다.



4)

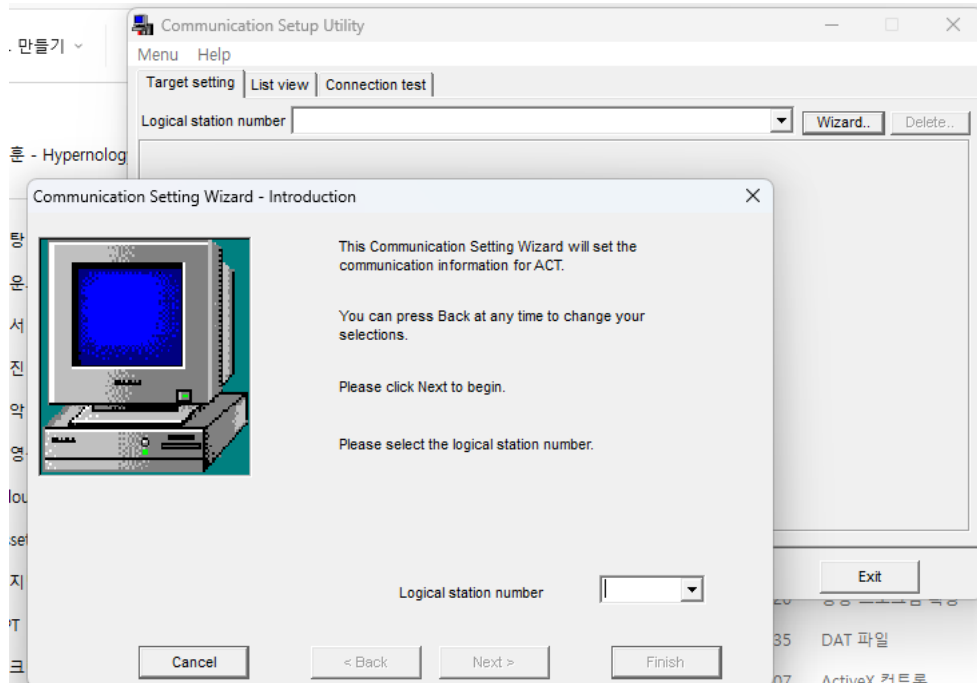
설치 후

C:\MELSEC\Act\Util ← 이 경로를 들어가서 ActComm.exe파일을 실행한다

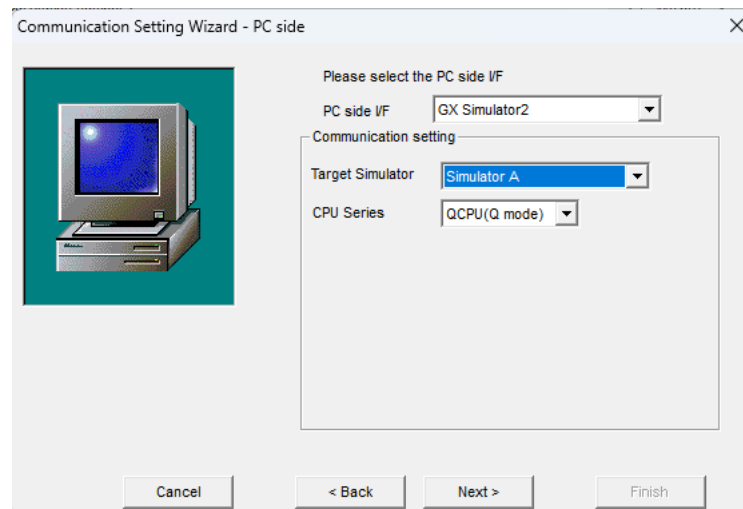


5)

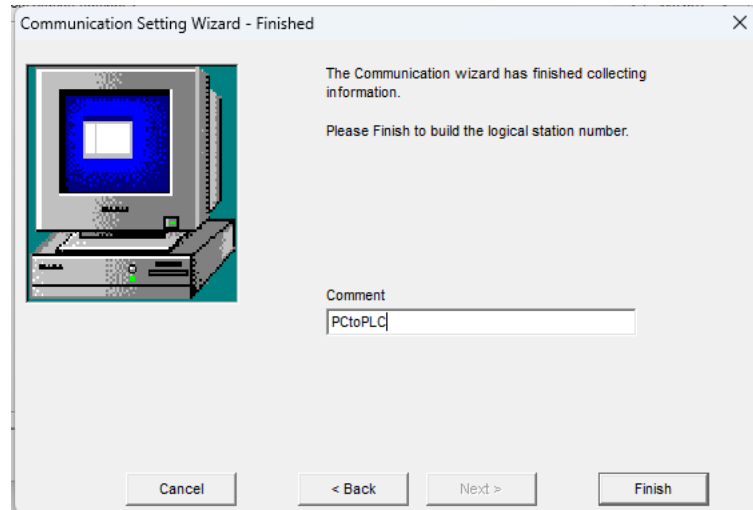
상단 우측 쪽 'Wizard' 버튼을 클릭하면 아래와 같은 화면이 뜬다.



Logical station number는 0 next하고



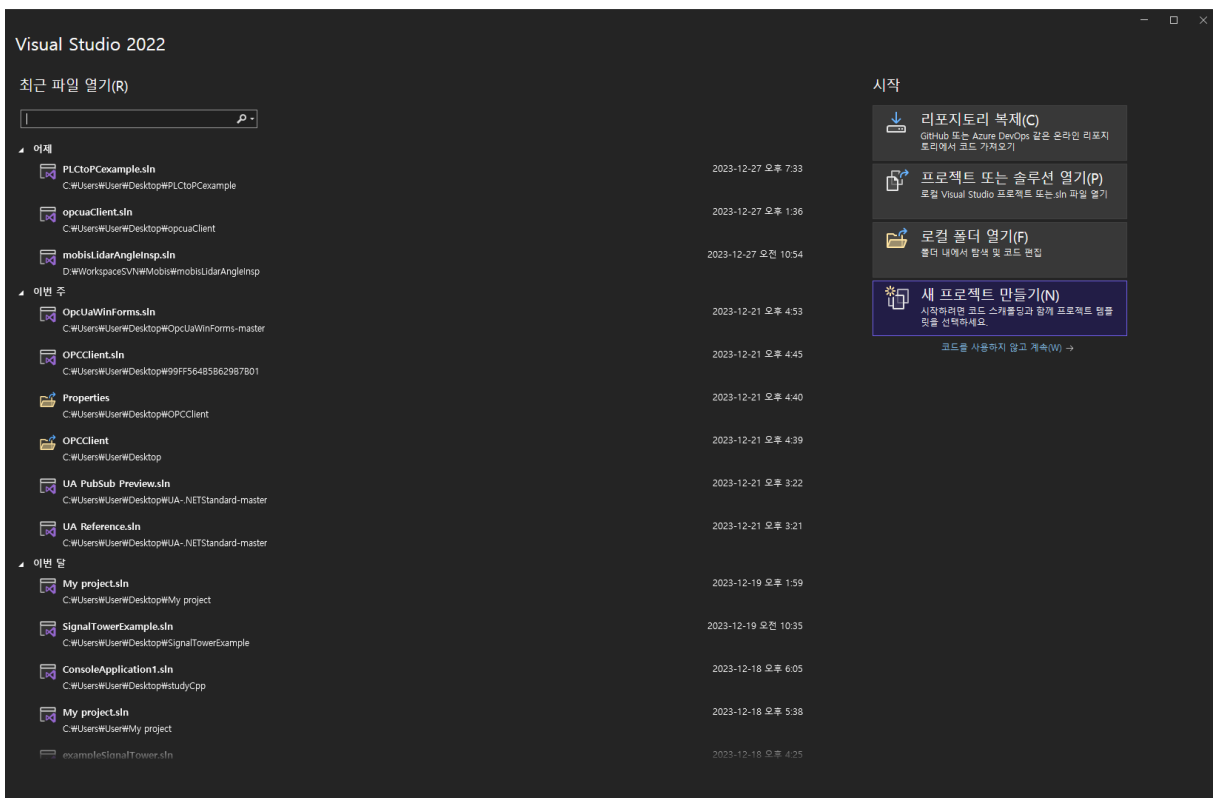
위와 같이 설정을 맞춰준 다음 next



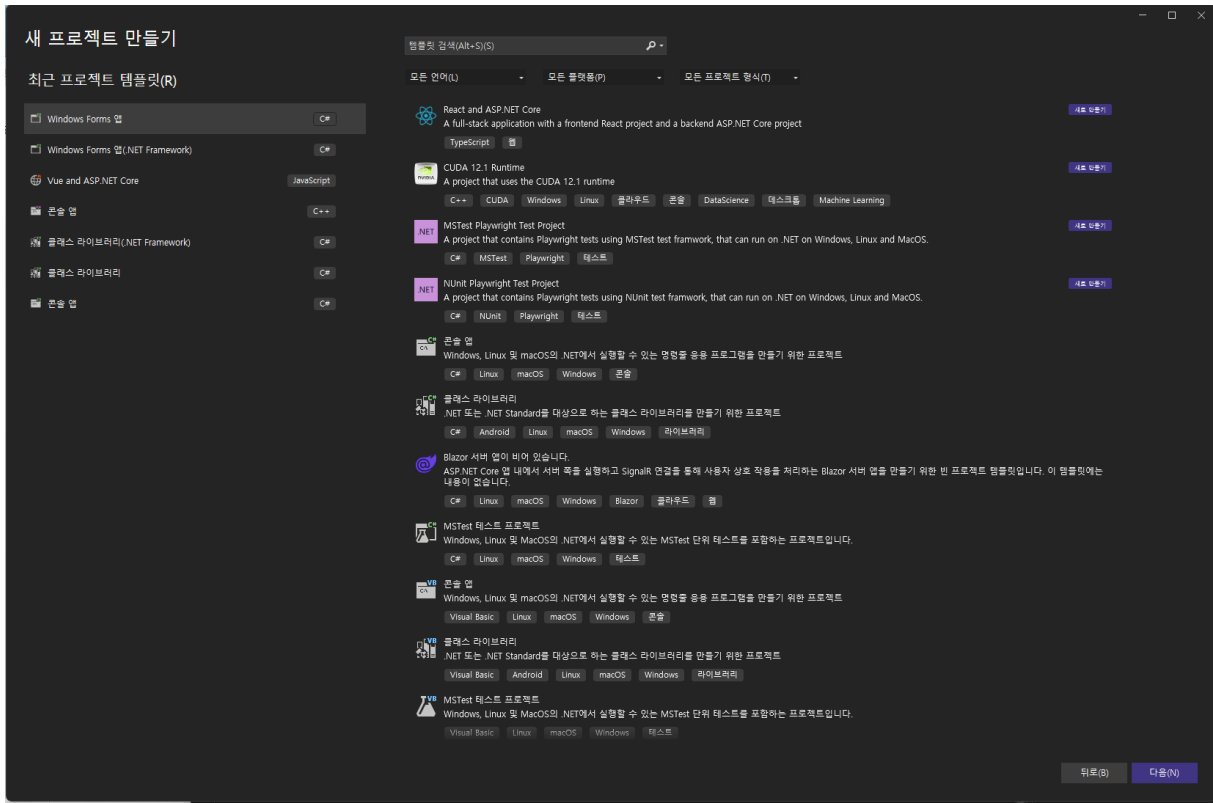
Comment는 알아보기 쉬운걸로 하고 Finish

▼ 4. PLC데이터 리딩 프로그램 생성

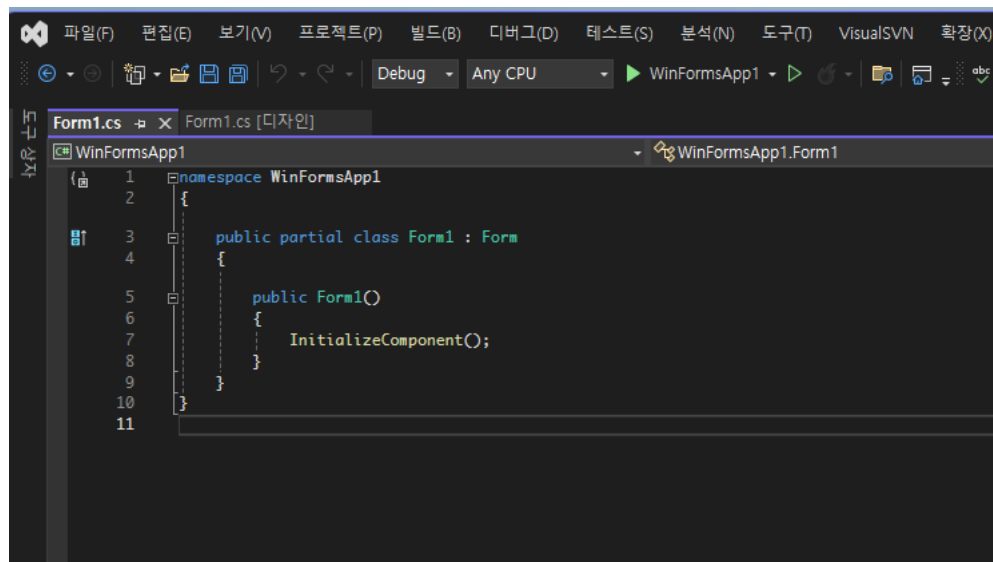
1) 새프로젝트 만들기



2) Windos Form



개발 프레임워크는 다양하게 사용 가능하겠지만, 필자에게 익숙한 winform으로 개발하였고,
다음 순서로는 프로젝트 명, .net버전 등 설정하여 프로젝트를 만들면 다음과 같은 초기 화면이 생성된다.

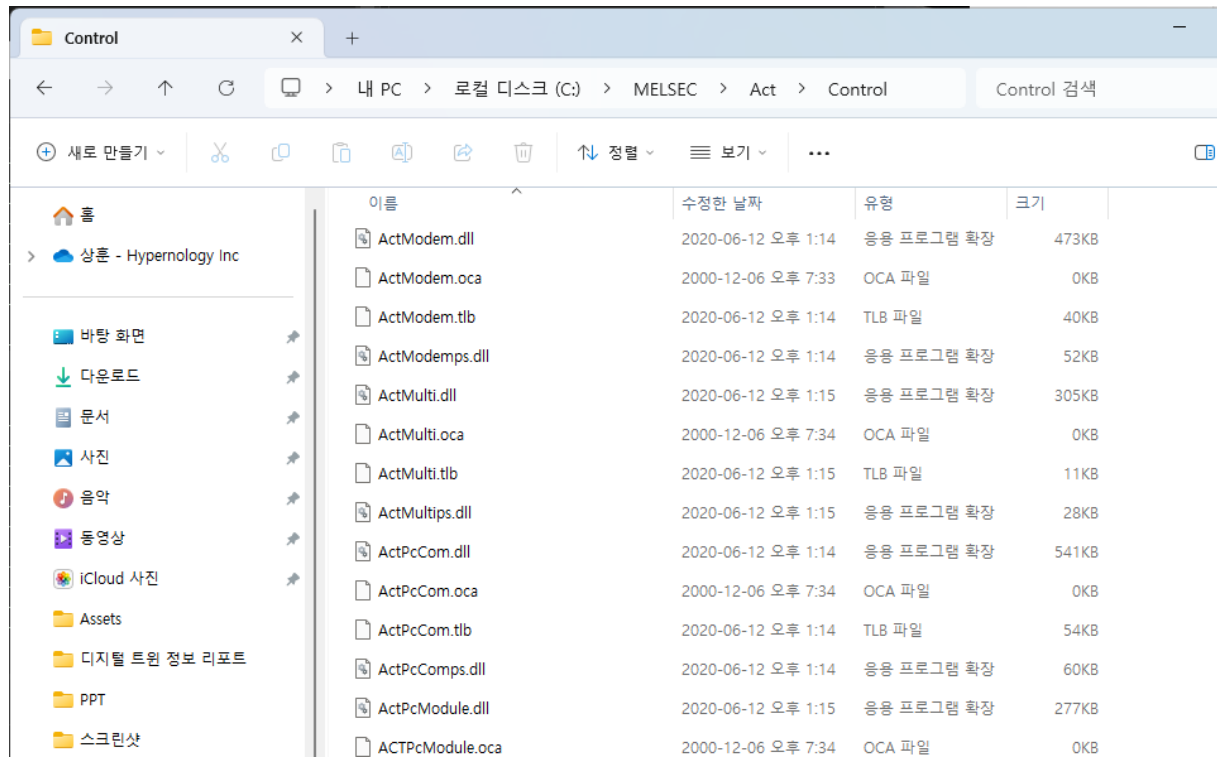


3) 참조추가

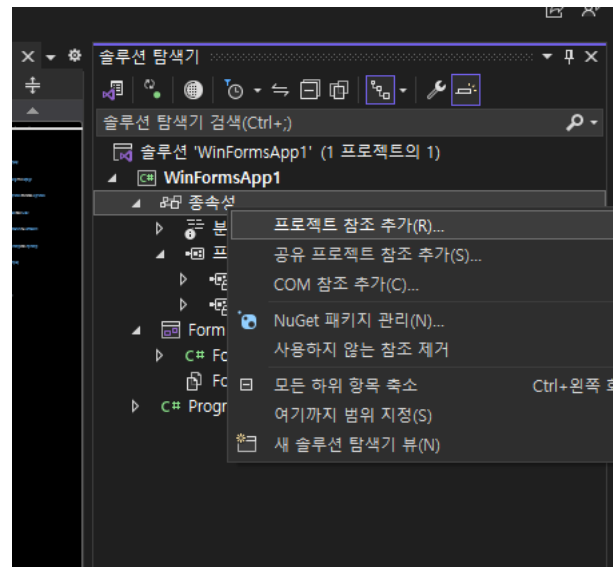
미쯔비시에서 제공하는 PLC 연결, 사용 등에 관련된 클래스를 사용하기 위해 .dll파일을 추가하여야 한다.

MX Component를 설치하면 생기는 경로 안에서 dll 파일을 확인 하여야한다.

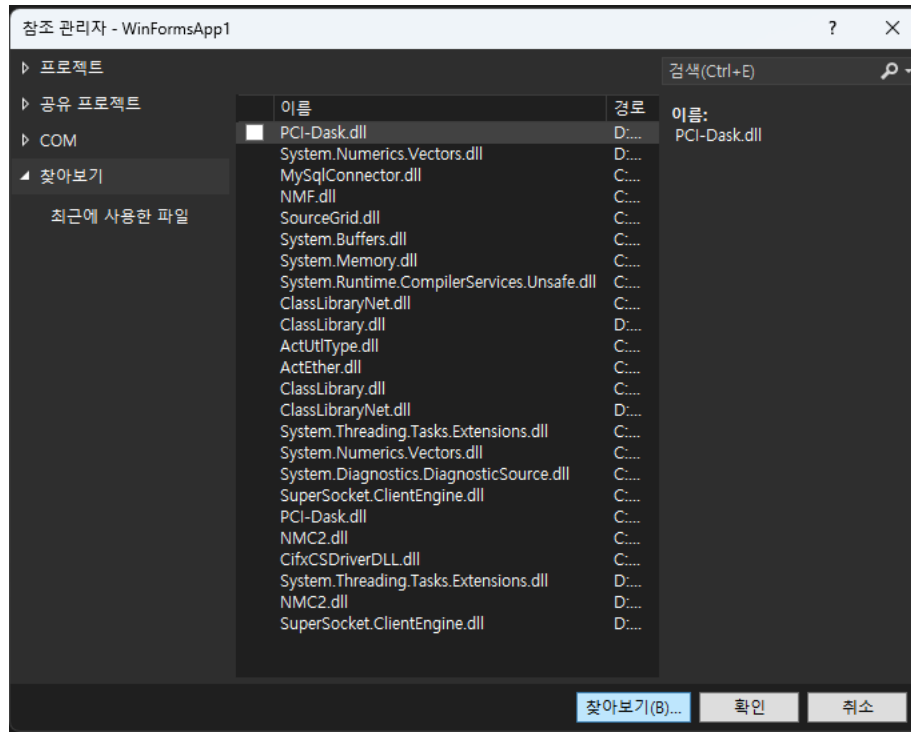
경로 : C:\MELSEC\Act\Control



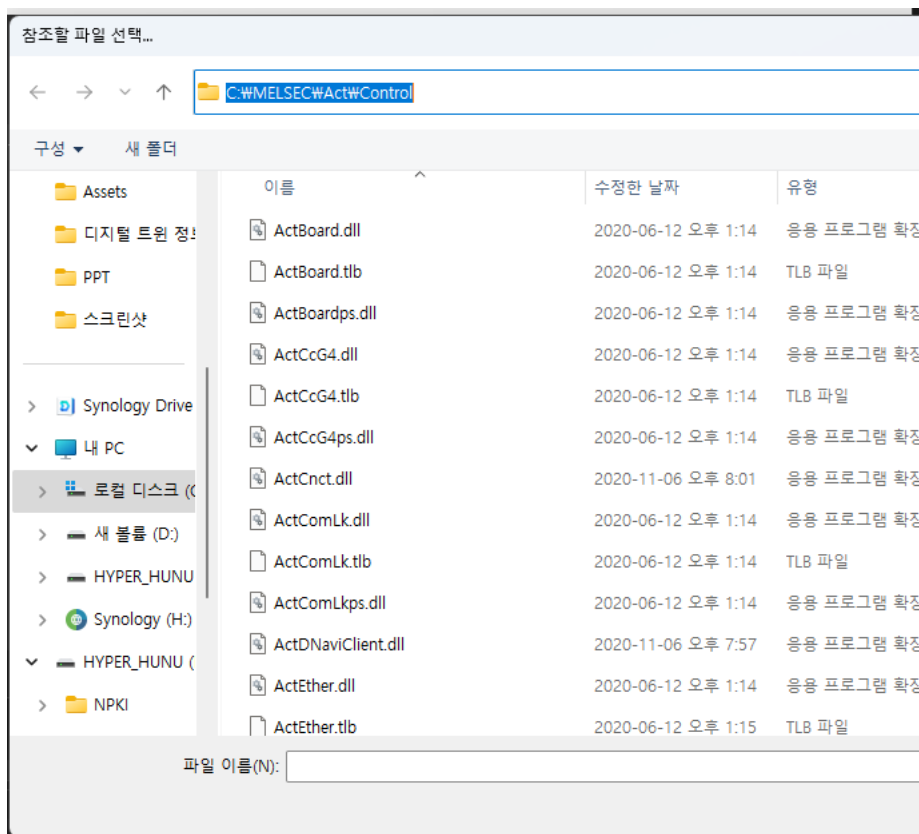
확인되면 다시 vs프로젝트 솔루션 탐색기에서 프로젝트명 아래 종속성 마우스 우클릭, 프로젝트 참조 추가를 한다



다음 찾아보기 탭에서 찾아보기 클릭



아래 경로대로 찾고 사용할 dll 파일을 추가해준다



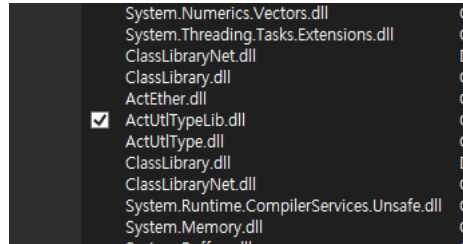
일단 아래 3가지 파일을 참조하였는데.

ActUtilTypeLib.dll

ActPcUsb.dll

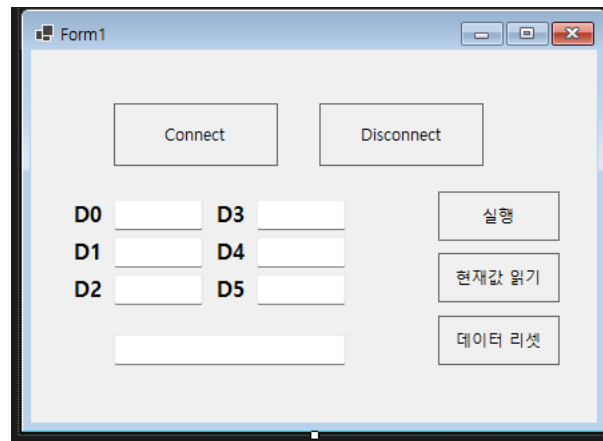
ActEther.dll

이번 예제에서는 ActUtilTypeLib.dll의 클래스만 사용하고 나머지는 어떤 dll인지 모르겠지만, 파일명을 보았을 때 통신 방식별로 지정되어 있는 dll인 것 같다.



참조가 잘 적용되었다면 끝

4) 화면 구성



라벨들은 버튼으로 만들어도 무관하다. 라벨 이름은 아래 대로 설정

Connect

Disconnect

Exe

ReadData

DataReset

TextBox이름은

D0~D5, EventMemo 순서이다.

다른 속성들은 보류하고

설명하자면 PLC연결, PLC연결종료, PLC 코드실행, 현재값 읽기, 데이터 리셋

D0~D5는 PLC코드 메모리 주소이고 EventMemo는 현재 상태표시줄 정도로 사용할 것이다.

5) 코드 작성

코드는 토크 안에 작성하고, 각 라벨별로 클릭 이벤트 및 PLC데이터 저장 변수, 초기화 등 간단한 코드로 이루어져 있다.

▼ C# PLC 접속 코드

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using ActUtilTypeLib;

namespace PLCToPCexample
{
    public partial class Form1 : Form
    {
        int DataRead1 = 0;
        int DataRead2 = 0;
        int DataRead3 = 0;
        int DataRead4 = 0;
        int DataRead5 = 0;
        int DataRead6 = 0;

        ActUtilType ActUtilType;

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            Connect.Enabled = true;
            Disconnect.Enabled = false;
            Exe.Enabled = false;
            ReadData.Enabled = false;
            DataReset.Enabled = false;
        }

        private void Connect_Click(object sender, EventArgs e)
        {
            ActUtilType = new ActUtilType();
            // ActUtilType.ActLogicalStationNumber값은 MX Component에서 설정한 Station 번호입니
            ActUtilType.ActLogicalStationNumber = 0;
            // 리턴값이 "0"이면 PLC 연결 성공
            int nRtn = ActUtilType.Open();

            if (nRtn == 0)
            {
                EventMemo.Text = "PLC 접속 성공";
                Timer.Enabled = true;
                Timer.Interval = 100;

                Connect.Enabled = false;
            }
        }
    }
}

```

```

        Disconnect.Enabled = true;
        Exe.Enabled = true;
        ReadData.Enabled = true;
        DataReset.Enabled = true;
    }
    else
    {
        EventMemo.Text = "PLC 접속 실패";

        Connect.Enabled = true;
        Disconnect.Enabled = false;
        Exe.Enabled = false;
        ReadData.Enabled = false;
        DataReset.Enabled = false;
    }
}

private void Disconnect_Click(object sender, EventArgs e)
{
    Timer.Enabled = false;
    int nRtn = ActUtilType.Close();

    if (nRtn == 0)
        EventMemo.Text = "PLC 접속 해제";

    Connect.Enabled = true;
    Disconnect.Enabled = false;
    Exe.Enabled = false;
    ReadData.Enabled = false;
    DataReset.Enabled = false;
}

private void Exe_Click(object sender, EventArgs e)
{
    int plcData = 0;
    // 스위치 "M1"의 값 Read

    ActUtilType.GetDevice("M1", out plcData);

    if (plcData == 0)
    {
        ActUtilType.SetDevice("M1", 1);
        EventMemo.Text = "스위치 켜기";
    }
    else
    {
        ActUtilType.SetDevice("M1", 0);
        EventMemo.Text = "스위치 끄기";
    }
}

private void ReadData_Click(object sender, EventArgs e)
{
    ActUtilType.GetDevice("Y20", out DataRead1);
}

```

```

        ActUtilType.GetDevice("D1", out DataRead2);
        ActUtilType.GetDevice("D2", out DataRead3);
        ActUtilType.GetDevice("D3", out DataRead4);
        ActUtilType.GetDevice("D4", out DataRead5);
        ActUtilType.GetDevice("D5", out DataRead6);

        D0.Text = DataRead1.ToString();
        D1.Text = DataRead2.ToString();
        D2.Text = DataRead3.ToString();
        D3.Text = DataRead4.ToString();
        D4.Text = DataRead5.ToString();
        D5.Text = DataRead6.ToString();
    }

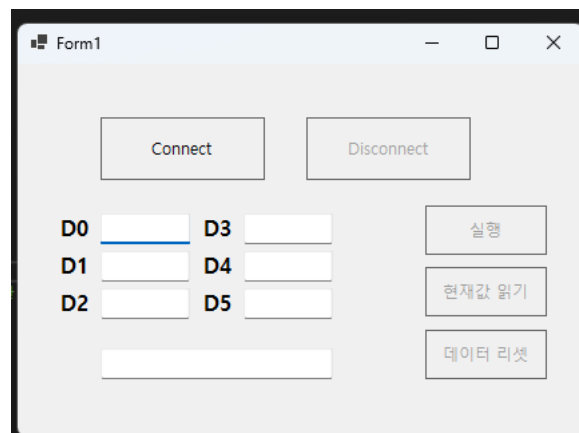
    private void DataReset_Click(object sender, EventArgs e)
    {
        ActUtilType.SetDevice("D0", DataRead1 = 0);
        ActUtilType.SetDevice("D1", DataRead2 = 0);
        ActUtilType.SetDevice("D2", DataRead3 = 0);
        ActUtilType.SetDevice("D3", DataRead4 = 0);
        ActUtilType.SetDevice("D4", DataRead5 = 0);
        ActUtilType.SetDevice("D5", DataRead6 = 0);

        D0.Text = DataRead1.ToString();
        D1.Text = DataRead2.ToString();
        D2.Text = DataRead3.ToString();
        D3.Text = DataRead4.ToString();
        D4.Text = DataRead5.ToString();
        D5.Text = DataRead6.ToString();
    }
}
}

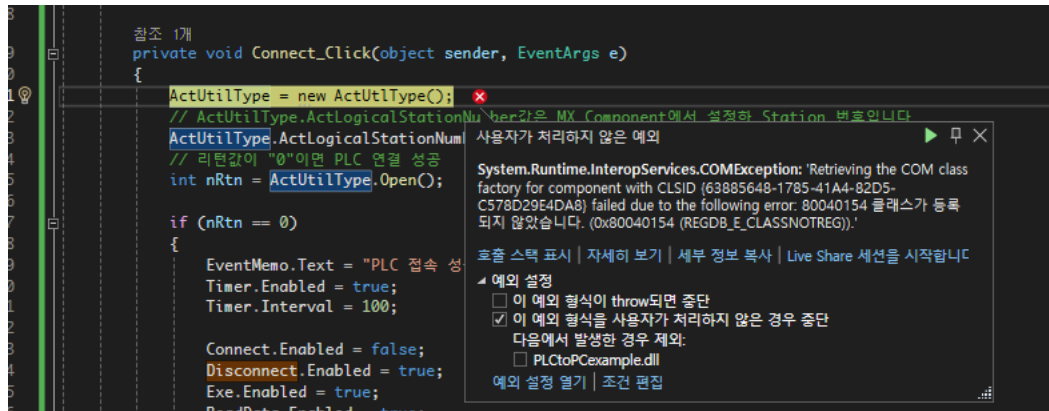
```

▼ 5. 프로그램 실행

프로젝트를 빌드하고 디버그한다.

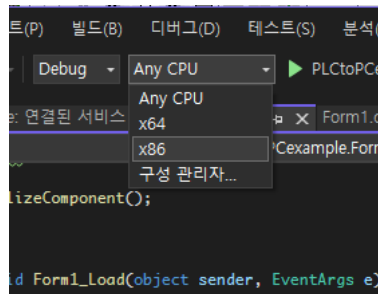


실행 시, 위와 같은 UI를 확인할 수 있고, Connect라벨을 클릭해 PLC연결을 시도하였다.

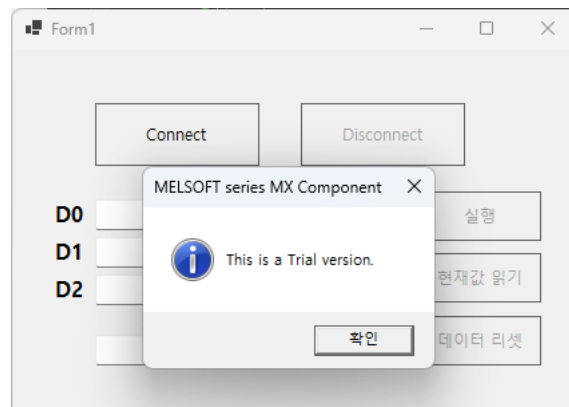


시도하니 위와 같이 dll파일로 참조한 ActUtilType()에서 에러를 발생하였다.

어떤 상황에서 발생한 에러인지는 모르겠지만 해결 방법을 찾았다.

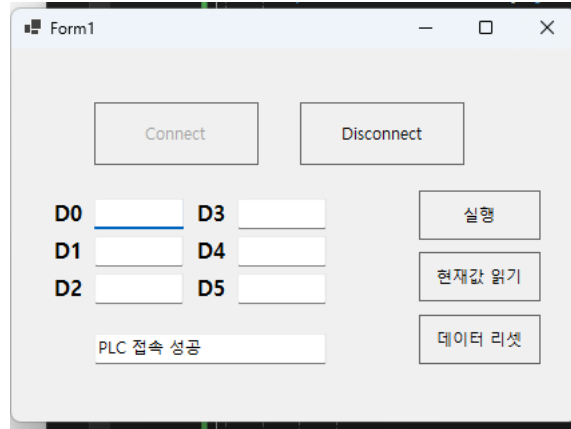


구성 관리자의 플랫폼을 x86으로 설정하고 Connect버튼을 클릭하니 잘 연결되었다.



필자는 MX Component 테스트버전?을 사용하여 팝업이 떴다.

정상 연결시 Enabled처리와 상태표시줄 text를 바꿔주어서



위 화면과 같은 결과를 얻었다.

▼ 6. 프로그램 기능 검토 및 PLC수정

먼저 '2. PLC프로젝트 코드 작성'에서 작성된 PLC코드를 수정해야 한다.

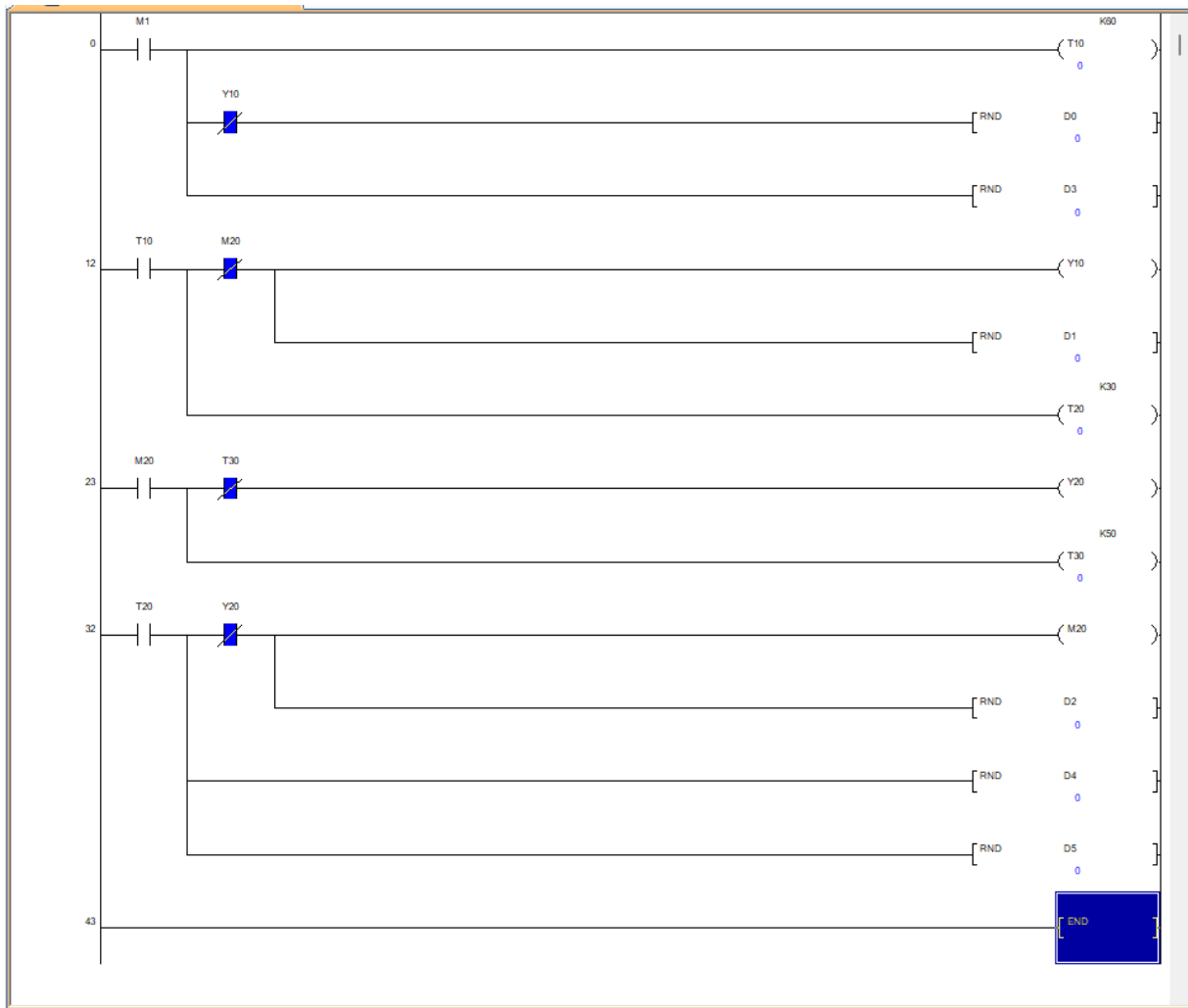
이유는 C#코드에서 GetDevice하여 불러올 PLC의 데이터 메모리 위치를 D0~D5로 설정하였기 때문에 (2번에서 작성한 PLC예제 코드로는 데이터 변화 확인에 한계가 있음) 이에 맞춰서 PLC코드를 수정해 본다.

```

참조 1개
private void ReadData_Click(object sender, EventArgs e)
{
    ActUtilType.GetDevice("D0", out DataRead1);
    ActUtilType.GetDevice("D1", out DataRead2);
    ActUtilType.GetDevice("D2", out DataRead3);
    ActUtilType.GetDevice("D3", out DataRead4);
    ActUtilType.GetDevice("D4", out DataRead5);
    ActUtilType.GetDevice("D5", out DataRead6);
}

```

필자도 PLC코드는 익숙하지 않아서 간단하게 아래 사진처럼 만들어 보았다.



로직을 파악하면서 작성한 래더는 아니라서 로직은 잘 모르겠다.. 시간이 된다면 로직을 맞춰서 PLC데이터를 가져오고 데이터를 활용한 디지털트윈, 또는 데이터값 계산을 해보고싶다.

우선 결과 영상으로

https://prod-files-secure.s3.us-west-2.amazonaws.com/c2b85089-2f86-4fb3-8dfe-bbc5e774c185/9cf64ae9-6e1d-4e39-a8ad-a5ccf6b22b2b/%EB%85%B9%ED%99%94_2023_12_28_15_34_36_906.mp4

PLC연결부터, M1에 값을 넘겨 실행, 종료.

값 연속 읽기, 1회읽기.

데이터 리셋

연결 종료 등의 기능 가진 프로그램을 완성하였다.