송신부

1. 통신으로 주고 받을 자료 형 선언

mJigNum : 지그 번호 Index
mData : 예비
sData : 전달 해줄 String 데이터

```
typedef struct DataInfo{
  int mJigNum;
  int mData;
  char sData[100];
}DINFO, *PINFO;
```

2. 송신 할 부분에서 하기와 같이 처리한다.

```
DINFO myInfo;
//------------------
// 송신 데이터 입력
//------------------
myInfo.mJigNum =1;
myInfo.mData =0;
strcpy(myInfo.sData,"TEST PROGRAM");

// WM_COPYDATA 메시지를 이용하기 위해서는
// COPYDATASTRUCT에 값을 실어서 보내고 받는다.
COPYDATASTRUCT copyData;
copyData.dwData = 333;    // 임의의 Index임
copyData.cbData = sizeof(myInfo);
copyData.lpData = &myInfo;

// 수신 윈도우 핸들 찾기
HWND  hTarWnd = FindWindow(NULL, TEXT("Server") );

// hTarWnd : 보낼 윈도우 핸들
// Handle  : 내 프로그램 핸들
SendMessage( hTarWnd, WM_COPYDATA, (WPARAM)Handle, (LPARAM) &copyData);
```

3. Target의 윈도우 핸들을 얻기 위해 해당 프로그램의 caption 정보를 알아야 한다.

4. SendMessage후 Target에서 메시지를 받았는지 확인 하는 방법 ?

dwData는 임의의 값으로 사용하고, lpData형은 전송될 데이터를 사용자 자료형에 넣고, 이를 lpData에 어드레스를 실어서 SendMessage로 보낸다. cbData는 사용자 자료형의 크기를 넣어 준다.

```
typedef struct DataInfo{
  int mJigNum;
  int mData;
  char sData[100];
}DINFO, *PINFO;
```

```
myInfo.mJigNum =1;
  myInfo.mData =0;
  strcpy(myInfo.sData,"TEST PROGRAM");
```

```
COPYDATASTRUCT copyData;
  copyData.dwData = 333;
  copyData.cbData = sizeof(myInfo);
  copyData.lpData = &myInfo;
```

```
  // 수신 윈도우 핸들 찾기
  HWND  hTarWnd = FindWindow(NULL, TEXT("Server") )
```

```
//메시지 전송
SendMessage( hTarWnd, WM_COPYDATA, (WPARAM)Handle, (LPARAM) &copyData);
```

수신부

1. 통신으로 주고 받을 자료 형 선언

```
typedef struct DataInfo{
  int mJigNum;
  int mData;
  char sData[100];
}DINFO, *PINFO;
```

2. 클래스 선언부에 윈도우 메시지 맵 선언
   비주얼 C++의 경우 Class Wizard를 이용하고, BCB의 경우 코드레벨에서 입력해준다.

```
BEGIN_MESSAGE_MAP
    // WM_COPYDATA에 대한 메시지 맵 작성
    // TMessage를 사용한다.
    VCL_MESSAGE_HANDLER(WM_COPYDATA, TMessage, GetCopyData );
END_MESSAGE_MAP(TForm);
```

3. 해당 처리 함수 선언

```
  void __fastcall GetCopyData(TMessage &Msg);
```

4. 메시지 처리 함수에서 처리

```
void __fastcall TForm1::GetCopyData(TMessage &Msg)
{
  Print("Message Received");

  // copy할 데이터 구조체
  DINFO myInfo;

  // 메시지 송수신 구조체
  COPYDATASTRUCT *pCopyData;

  // Msg의 인자중 LParam이 수신 정보이므로, 이것을 가르킨다.
  // WM_COPYDATA를 사용하므로, COPYDATASTRUCTURE가 이것을 가르키게 한다.

  pCopyData = (COPYDATASTRUCT*) Msg.LParam;    // LParam(copydata)를 가르킨다.

  //  COPYDATASTRUCT의 3개 멤버 변수로 access

  int dwdata = pCopyData->dwData;
  Print( "dwdata DATA = ", dwdata );

  int cbdata = pCopyData->cbData;
  Print( "CB DATA = ", cbdata );

  //  COPYDATASTRUCT의 3개 멤버 변수 중
  // lpdata가 string값을 저장했으므로, 이것을 가르킨다.
  // 구조체로 만들었기 때문에 (순서대로 꺼내오기 위해서)

  PINFO cpInfo = (PINFO)pCopyData->lpData;
  int JigNum = cpInfo->mJigNum;
  Print( "JIG NUM= ", JigNum );
  int mData = cpInfo->mData;
  Print( "mData NUM= ", mData );
  AnsiString  sData = cpInfo->sData;
  Print( "sData = " + sData );
}
```

메시지 처리함수는 인자로 TMessage형을 받으며, 정의는 하기와 같다. 데이터를 담아서 보내는 파라미터는 LParam이며, WM_COPYDATA 메시지를 사용할 경우에는 COPYDATASTRUCT 구조체를

사용해야 한다.

```
struct TMessage
{
        Cardinal Msg;
        union
        {
                struct
                {
                        Word WParamLo;
                        Word WParamHi;
                        Word LParamLo;
                        Word LParamHi;
                        Word ResultLo;
                        Word ResultHi;

                };
                struct
                {
                        int WParam;
                        int LParam;
                        int Result;
                };

        };
} ;
```

```
typedef struct tagCOPYDATASTRUCT {  // cds
    DWORD dwData;
    DWORD cbData;
    PVOID lpData;
} COPYDATASTRUCT;

dwData
Specifies up to 32 bits of data to be passed to the receiving application.
cbData
Specifies the size, in bytes, of the data pointed to by the lpData member.
lpData
Points to data to be passed to the receiving application. This member can be NULL.
```

메시지 처리 함수의 인자인 Msg의 Msg.LParam을 COPYDATASTRUCTURE가 가르키도록 하고

```
pCopyData = (COPYDATASTRUCT*) Msg.LParam;    // LParam(copydata)를 가르킨다.
```

COPYDATASTRUCTURE 구조체의 멤버 변수는 하기의 방법으로 Access 한다.

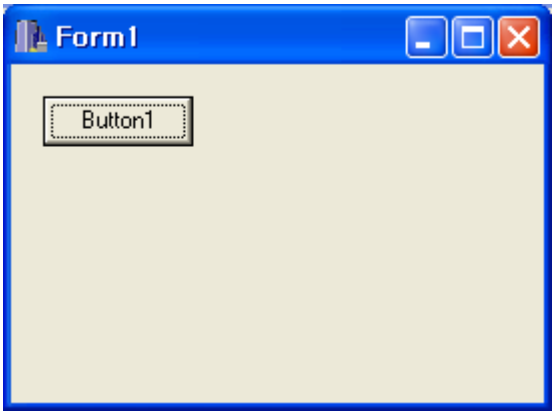
```
  int dwdata = pCopyData->dwData;
  Print( "dwdata DATA = ", dwdata );

  int cbdata = pCopyData->cbData;
  Print( "CB DATA = ", cbdata );

  //  COPYDATASTRUCT의 3개 멤버 변수 중
  // lpdata가 string값을 저장했으므로, 이것을 가르킨다.
  // 구조체로 만들었기 때문에 (순서대로 꺼내오기 위해서)
```

사용자 정의 메시지를 얻기 위해서 하기의 방법을 이용한다.
COPYDATASTRUCT 포인터 변수를 이용해 사용자 정의 메시지 구조체로 접근한다.

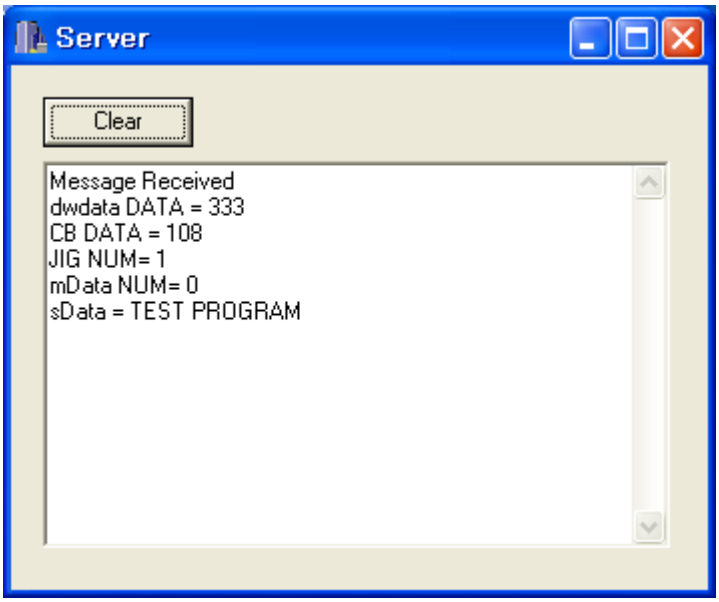
```
PINFO cpInfo = (PINFO)pCopyData->lpData;
```

```
        PINFO cpInfo = (PINFO)pCopyData->lpData;
        int JigNum = cpInfo->mJigNum;
        Print( "JIG NUM= ", JigNum );
        int mData = cpInfo->mData;
        Print( "mData NUM= ", mData );
        AnsiString  sData = cpInfo->sData;
        Print( "sData = " + sData );
```

<<응용 프로그램 예>>



송신부



수신부