eRun3.5 스크립트 설명서

Reference Manual of Script Functions on eRun Solution

```
void eGEMCommStateChanged(int nState)
{
    // Argument : nState value
    //  Comm_None = -1,
    //  Comm_Disabled = 1,
    //  Comm_WaitCRFromHost = 2
    //  Comm_WaitDelay = 3,
    //  Comm_WaitCRA = 4,
    //  Comm_Communicating = 5,
    //  Comm_Enabled = 6

    @CIM.nCommState = nState;

    if (nState == 1)        { }     // @CIM.sCommState = "Comm Disabled"; }
    else if(nState == 2)    { InitCIM(); }          // @CIM.sCommState = "WaitCRFromHost";
    else if(nState == 3)    { ChangeControlState(1); }          // @CIM.sCommState = "WaitD
    else if(nState == 4)    { }         // @CIM.sCommState = "WaitCRA"; }
    else if(nState == 5)    {
        @DispTag.TopMenuNo = 9;
        _OpenPage("CIM_01", 0);
        ChangeControlState(4);
        //if ( 1 == @D15000.D15911 ) {  ChangeControlState(1);      }           //D15911 :
        //if ( 2 == @D15000.D15911 ) {  ChangeControlState(4);      }           //OffLine :
        //if ( 3 == @D15000.D15911 ) {  ChangeControlState(5);      }
    }           // @CIM.sCommState = "Communicating"; }
    else if(nState == 6)    { }         // @CIM.sCommState = "Comm Enable"; }
    else { }            // @CIM.sCommState = "Unknown"; }
```

JPEOPLE

개 정 이 력

버 전	날 짜	내 용
Rev1	2021년 11월	개정작성
Rev2	2022년 3월	내부함수 추가, _FileWriteString()
Rev3	2022년 8월	COMIZOA cmmSDK 라이브러리 인터페이스 내부함수 추가 _COMI_Method()
Rev4	2022년 9월	COMIZOA 모션리스트 함수 추가 (cmmLmxxxxx)
	2022년 10월	_IsOpenPage() 리턴값 개선. _VisionMethod() 함수 개선. "InvLoad2" 추가됨. _CopyTagValue() 함수 추가.
Rev5	2022/12/20	_CopyTagGroup(), _DeviceWriteWords(), _PrintScreenEx() 함수추가
Rev7	2023/6/21	_SFListUpdateColor(), _SFListUpdateBackColor(), _SFListUpdateTextColor() _ShowProgressBar() 함수추가.
Rev8	2023/10/20	_GetProjectInfo() 누락으로 설명서 추가 _SaveTagValues(), _LoadTagValues() 추가
Rev9	2024/5/10	_UserGetInfo() 함수추가
Rev10	2024/12/20	_SetTagValue() 함수복귀형 변경 _StrFindRev() 함수추가
Rev11	2024/04/21	스크립트 매뉴얼 Rev11 수정
Rev12	2025/6/17	FTP 전송함수 추가 _FTPLogOn(), _FTPLogOff(), _FTPCmd(), _FTPMsg(), _FTPUpload(), _FTPDownload()

- 목 차 -

제 1 장 소개

1.1 eRun Script 개요

eRun으로 제작되는 프로젝트는 다양한 센서, 계측기, 기계 등의 원격감시 및 원격제어를 하기위해 크게 사용자 화면(UI), 데이터베이스, I/O 통신으로 구성됩니다.

프로젝트를 기능별로 나누어서 보면 단순히 데이터 수집용 모니터링 기능을 구현하는 경우와 분석 및 제어기능을 구현하는 다소 복잡한 프로젝트를 구현해야 하는 경우가 있습니다. 이렇게 간단하거나 복잡한 사용자 요구사항에 유연하고 신속하게 대응하기 위해서 eRun 스크립트 언어를 내장하고 있습니다.

eRun 스크립트는 가장 강력한 컴퓨터 프로그래밍 언어 중 하나인 C언어 문법을 따르고 있습니다.
복잡하다고 두려워할 필요가 전혀 없습니다. eRun 스크립트는 프로젝트 구현에 필요한 가장 기본적인 내용만으로 구성되기 때문에 조금의 노력만으로도 충분히 스크립트 프로그래밍을 할 수 있습니다.

eRun 스크립트는 자체적으로 개발된 엔진이며 기능별 필요한 내장함수를 포함하고 있고 무한한 확장성을 가지고 있습니다.
Dd

1.2 특징

- 대소문자 구별.
- 기능별 독립적인 함수의 형태.
- 정수형, 실수형, 문자열 데이터타입 지원.
- 전역변수 지원
- 1차원 배열형 데이터타입 지원
- 260여개의 시스템 내장함수 지원(삼각함수, 파일제어, I/O디바이스제어, SQL, MS-EXCEL ...)
- 논리연산(AND, OR, XOR, NOT) 가능.
- 사칙연산(+, -, *, /,%,^) 가능.
- 제어연산(+=, -=, ++, --, <<, >>) 가능
- 태그연산(@태그그룹.태그명3 = @태그그룹.태그명1 + @태그그룹.태그명2)
- 반복 제어문(if, if else, while).
- 사용자함수에서 사용자함수 호출가능.
- 사용자 UI 오브젝트 제어.
- 상용 SQL(MS-SQL, MySQL, ORACLE, OLEDB, MS-ACCESS) 데이터베이스.
- MS-EXCEL 보고서처리
- I/O 통신 디바이스에 대한 직접 처리
- 경보사운드(WAV) 파일 재생.
- 외부 실행파일 호출 가능.
- 레지스트리 데이터 제어 가능.
- 그래프, 그리드, 경보내용 제어 가능.
- 기타

1.3 스크립트 구성

스크립트 문법구성은 아래와 같은 항목들로 간단하지만, 길어지면 복잡하게 보이는것 뿐입니다.
아래의 구성들로 변수이름, 함수이름, 태그이름들의 다소 긴 문자열이 반복됩니다.

- 함수 (내장함수, 사용자 정의함수)
- 외부 데이터변수 (함수외부)
- 내부 데이터 변수 (함수내부)
- 데이터형 (char, short, int, float, double, string, text)
- 조건 제어문 (if, else, &&, ||)
- 반복문 (while)
- 대입연산(=, -=, +=), 사칙연산(+/*-), 논리연산 (<<, >>, &, |) 비교연산 (==, !=, >=, <=)
- 모듈러 연산(%), 배타적 연산(^)

1.4 함수구성

사용자가 기능별로 분류해서 함수이름을 정의하고 내용을 문번적으로 정의합니다.
함수는 동일한 기능을 반복적으로 사용하는 경우 정의해두면 편리합니다.
함수의 구성형태는 아래와 같습니다.

```
함수형 함수이름(파라메터)
{
   int a, b, c;

   a=100;
   b=200;

   c=a+b;
}
```

함수형은 함수가 실행되고 결과를 돌려주어야 할 경우 데이터의 형태를 지정합니다. 지정 가능한 함수형은 void, char, short, int, float, double, string, text 입니다.

함수이름은 기능을 의미하는 이름으로 정의 하는게 좋고 알파벳으로 시작하며 이름의 길이제한은 32자 이내로 정의합니다.

파라메터는 함수를 호출하는 쪽에서 넘겨주는 데이터 변수이며 0개 이상의 데이터를 지정할 수 있습니다.
함수의 정의 형태는 아래와 같이 여러 형태로 사용 가능합니다.

```
// 돌려주는 데이터는 없음, 받는 데이터 없음.
void main1()
{
}

// 돌려주는 데이터 정수형, 받는 데이터 정수형 데이터변수.
int main2(int num)
{
    return 100;
}

// 돌려주는 데이터 실수형, 받는 데이터 실수형, 문자열형, 정수형 데이터변수.
double main3(double a, string name, int b)
{
    return (a * b);
}

// 돌려주는 데이터 문자열형, 받는 데이터 정수형 데이터변수.
string main(int tel)
{
    return "SEFA Technology";
}
```

호출하는 부분에서 사용하는 함수사용은 아래와 같습니다.
함수명(매개변수);

```
// main1() 함수를 넘겨주는 데이터 없이 호출합니다.
main1();

// main2() 함수를 호출하며 정수형 데이터상수 100을 넘겨줍니다.
main2(100);

// main3()함수를 호출하며 실수형, 문자열형, 정수형 데이터상수를 각각 넘겨줍니다.
main3(-99.5, "대한민국", 20000);
```

1.5 변수의 정의

스크립트에서 데이터 연산을 하기위해서 여러가지 형태의 변수를 선언하고 사용하며 내부와 외부에서 할 수 있습니다.
내부변수는 함수의 정의 부분에서 제일 상단에 선언하여야 하고, 그 함수 내부에서만 사용가능 합니다.
외부변수는 함수 정의부분 밖에서 정의가 되며, 그 스크립트 파일내부에 정의 되어있는 함수내부 어디서든 사용가능 합니다. 외부변수는 여러 함수에서 데이터 공유를 하고 계산을 하는데 사용할 목적으로 정의하고 사용하시면 편리합니다. 단, 스크립트 단위파일의 내부에서만 그 외부변수가 유효합니다.

변수형은 크게 정수형, 실수형, 문자열, 배열형 4종류의 형태가 있습니다.

정수형에는 char(1바이트), short(2바이트), int(4바이트) 3종류가 지원됩니다.
실수형에는 float(4바이트), double(8바이트) 2종류가 지원됩니다.
문자열형에는 string(4096바이트), text(256바이트)를 사용합니다.
배열형은 하나의 데이터 변수에 다수개의 방을 만들어서 사용하도록 합니다.

변수정의는 사용 형식은 아래와 같습니다.

```
// 변수 a, b, c 가 모두 int형 정수형 변수로 선언합니다.
int a, b, c, d[10];
// 변수 d를 실수형 데이터 변수로 선언합니다.
double d;
// 변수 name을 문자열형 데이터 변수로 선언합니다.
string name;
text txtName;
```

1.6 변수의 크기

eRun Script에서 데이터의 계산 처리를 위하여 여러가지 데이터형의 변수를 선언하고 사용합니다. 변수선언은 함수의 정의 부분에서 제일 상단에 선언하여야 합니다.
변수의 종류별 표현가능한 범위는 아래와 같습니다.

- char : -128 ~ +127 1 바이트
- short : -32768 ~ +32767 2 바이트
- int : -2147483648 ~ +2147483647 4 바이트
- float : 3.4e-38 ~ 3.4e+38 4 바이트
- double : 1.7e-308 ~ 1.7e+308 8 바이트
- text : 모든 유니코드 64문자(128바이트)
- string : 모든 유니코드 4096 문자(8192바이트).

- Text : 모든 유니코드 256 문자(512바이트).
- void : 변수크기 없음.

1.7 예약어 및 기본문법

■ 흐름제어문(if문, else문)

어떠한 조건에 따라서 스크립트 실행의 방향을 바꾸기 위한 것으로 여러 관계연산자와 논리연산자 그리고 산술연산자의 조합으로 조건절을 이끌어 내며 그에 따른 수행상태가 변하게 됩니다.
여기에 사용되는 예약어로는 if문과 else if문과 else문으로 세가지를 조합해서 사용합니다.

if문의 구조.

```
if( 조건수식 )
{
   문장
}
```

일반적으로 if의 조건수식에는 비교수식이 들어갑니다. (예를 들면, x>y 또는 c==6 등).
만약 수식의 값이 참(TRUE)이면 (즉, x가 y보다 크다, c의 값이 6이다) 수식의 값이 1이 되고 그 밑의 문장이 수행됩니다. 그렇지않고 거짓(FALSE)이면, 그 밑의 문장은 무시되고 그 다음으로 수행이 넘어갑니다.
앞에서 모든 수식은 값을 가진다고 하였습니다.
이 때에도 if에 연결된 수식이 참이면 1의 값을 갖고, 거짓이면 0을 갖습니다.
조건수식이 참을 수행하는 문장은 하나 이상일 수도 있습니다. 그 때는 물론, 다음과 같이 괄호{}로 묶어주고 복합명령문이라고 합니다.

```
if(score > big)                    // 단순 명령문
   result =1;

 if(value > 100) {                  // 복합명령문
   num++;
   result =100;
}
```

if와 else의 구조문.

```
if( 조건수식 )
   수행할 문장.
else
   수행할 문장.
```

if 만 쓰일 때는 수식이 거짓일 경우 아무것도 하지 않았습니다. 하지만, if-else문의 경우는 수식이 거짓이면 else

밑의 문장을
수행합니다. 다시 말해서 수식이 참이냐, 거짓이냐에 따라서 두가지 문장중의 하나를 선택할 수 있다는 것입니다.

다중선택의 if와 else문의 구조문.

```
if( 조건수식 )
   수행할 문장 A.
else if( 조건수식 )
   수행할 문장 B.
else
   수행할 문장 C.
```

첫줄에서 if문의 수식이 참일 경우 수행할 문장 A를 수행하고, 그렇지 않고 else if문의 수식이 참이면 수행할 문장 B를 수행하며,
첫번째 두번째 수식 모두 거짓일 경우 마지막 문장 C가 수행됩니다.

■ 관계연산자와 수식.

관계연산자는 주로 두개의 수식값을 비교하기 위해서 사용합니다. 이 비교수식에서 사용되는 변수의 데이터형은 정수형, 실수형, 문자열 비교등이 사용되며, 가능하면 두개의 데이터형을 동일하게 사용하는 것이 안정성있게 사용할 수 있습니다.
예를 들어서 정수형의 데이터와 실수형의 데이터를 비교하게 되면 계산시의 오차로 인하여 ==, !=, >=, <=등의 연산을 알아보기 어렵게 되는 경우가 발생 할 수 있습니다. 특히 주의하여 주시기 바랍니다.

관계연산자(비교연산자)로 사용되는 기호로는 아래와 같습니다.

```
A < B          : A가 B보다 작으면
A <= B         : A가 B보다 작거나 같으면
A == B         : A와 B가 같으면
A >= B         : A가 B보다 크거나 같으면
A > B          : A가 B보다 크면
A != B         : A가 B와 같지 않다면.
```

관계수식에서 비교연산의 결과가 참인 경우만 해당 if 또는 else if의 수행문장을 수행하게 되는데, 참이라는 의미는 어떤 연산결과값이 0이면 거짓이고 그렇지 않고 0보다 크다면 참이 되는 것입니다.

위의 연산자에 A와 B에는 정수값, 실수값, 문자열값 등이 올 수 있습니다. 예들 들어서,

```
if( a == 100 )              // a값이 100이면
   // 수행...
else if( name == "SBP" )   // name의 값이 "SBP" 이면
```

```
    // 수행...
  else if(val >= 123.5)        // val의 값이 123.5보다 크거나 같으면
    // 수행...
```

그러나 문자열을 비교 할 경우에는 ==(같음) 관계연산자만 사용 가능합니다.

■ 논리연산자와 관계연산자

논리연산자는 위에서 설명하였듯이 &&(AND), ||(OR), !(NOT) 연산자를 말하며, 관계연산자와 혼용해서 사용할 수 있습니다. 복합적인 조건에 대하여 제어를 하고자 할 경우 사용합니다.

■ 반복문(while)

반복문 while의 사용문법은 아래와 같습니다.

```
while( 조건수식 )
  수행문장.
```

위와 같이 반복 예약어인 while문은 조건수식이 참인 경우 계속적으로 루프(LOOP)를 돌면서 수행 문장을 실행합니다. 조건수식이
거짓일 경우에만 루프에서 빠져나오게 되므로 주의하지 않으면 무한루프를 돌게 되면서 실행이 중단될 수도 있습니다. 사용에 있어서 각별히 주의를 하시기 바랍니다.

while문도 여러 개의 반복 문으로 중첩이 될 수 있으며 수행해야 할 부분이 많은 경우 괄호{}를 사용하여 수행 블록을 코딩할 수 있습니다.
while문은 조건수식이 참인 경우는 정상적으로 괄호 안에 있는 문장이 수행되지만, 반대로 거짓일 경우는 한번도 수행을 하지 않을 수 있습니다.

조건수식에는 if-else구문에서와 마찬가지로 관계연산자를 사용할 수 있으며 또한 논리연산자도 사용가능 합니다.

```
int a;
string str;
a =0;
while( a < 100) {
  str =_NumToStr(a);
  _MsgBox(str, "결과", 0, 0);
  a++;
}
```

위의 구문은 정수형 변수 a를 선언하고, 문자열변수 str을 선언합니다. 그리고 변수 a에 초기값을 0으로 대입하고 루프로 들어가서 a가 100보다 작으면 괄호{}안의 문장을 수행합니다.

수행하는 첫번째 문장은 정수값 a의 현재값을 문자열로 변환해서 변수 str에 대입합니다.
그리고 결과를 화면에 메시지박스를 띄워서 보여집니다. 그리고 변수 a의 값을 1만큼 증가시킵니다.
다시 while문으로 들어가서 조건수식을 검사하는데, 다시 a의 값이 100보다 작은지를 판단하면서 이 조건이 거짓(a가 100보다 큰 경우)이 되면 루프를 빠져 나오게 됩니다.

```
 int a;
string str;
a =0;
while( 1 ) {
   if( a >= 100 ) break;
   str =_NumToStr(a);
   _MsgBox(str, "결과", 0, 0);
   a++;
}
```

위의 문장은 문장에 if 조건문과 루프를 빠져 나오게 하는 break문을 추가해서 수정하였습니다. 결과는 이전 코드와 동일한 수행결과가 나옵니다.

■ 루프의 탈출

무한루프(while)을 돌다가 적당한 조건이 맞았을 경우 루프를 빠져 나오기 위해서 break를 사용합니다. 이 break 예약어는 반드시 괄호{} 블록이 묶여있고 루프구조에서만 사용할 수 있습니다.

■ 누적치환 연산자 (+=, -=, *=, /=)

기본적으로 치환 연산자는 = 으로서 이것은 오른쪽의 값을 왼쪽의 변수에 대입하기 위해서 사용하는 연산자입니다. 이와 유사하게 변수의 값이 일정한 차이 또는 일정한 비율로 변화할 때 사용할 수 있는 간단한 연산자를 제공합니다.

아래에 여러가지 사용 예가 있습니다.

value +=20 은 value = value + 20 과 같은 의미입니다.
value -=20 은 value = value - 20 과 같은 의미입니다.
value *=20 은 value = value * 20 과 같은 의미입니다.
value /=3.5 는 value = value / 3.5 와 같은 의미입니다.
value %=2 는 value = value % 2와 같은 의미이며, 이 연산은 value를 2로 나눈 나머지값을 구합니다.
Value ^=1 은 value = value ^ 1과 같은 의미이며, value 값을 반전시킨 값을 구합니다.

이상의 누적치환 연산자는 증가, 감소연산자와 유사합니다만, 증가, 감소 연산자는 변수의 값을 1만큼씩만 변화시킨다는 점이 다릅니다.

제 2 장 내장함수

2.1 일반 함수

2.1.1 _AsciiCode()

함수원형

int _AsciiCode(strin strText, int nPos)

파라메터	strText : 아스키문자열 nPos : 읽어오려는 위치값 (0부터 시작)
리턴형	정수
설명	아스키문자열에서 nPos번째 문자의 코드값을 읽어온다.
함수활용	Check Sum 계산을 할 경우 유용하게 사용할 수 있다

```
/*********************************************************
예제 : 문자열 중에서 특정 위치의 코드값 얻어오기
*********************************************************/
void GetAsciiCode()
{
	string str;
	int nCode;
	int nStartIdx;

	str = "ABCDE";
	nStartIdx = 0;

	nCode = _AsciiCode(str, nStartIdx); // 'A' 의 코드값 = 65 (0x41)
	_TraceEx("str[%s], StartIndex[%d], nCode[%d][0x%X]", str, nStartIdx, nCode, nCode);

	nStartIdx = 1;
	nCode = _AsciiCode(str, nStartIdx); // 'B' 의 코드값 = 66 (0x42)
	_TraceEx("str[%s], StartIndex[%d], nCode[%d][0x%X]", str, nStartIdx, nCode, nCode);

	/*************************************************************************************************
	_TraceEx를 통해 결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. eRun이 실행되었으면 [F6] 트레이스창 을 활성화 시킨다.
	3. [F7] 함수수동호출 을 활성화 시킨후 본 사용자 정의 함수를 실행시킨다.
	4. 트레이스창의 출력문을 확인한다.
```

```
        ********************************************************************************************
*/
}
```

2.1.2 _AsciiToHx()

string _AsciiToHx(string strText, int nOrder)

파라메터	strText : 아스키문자열 (max. 2048 bytes) nOrder : 상,하 바이트 순서를 지정한다. 1=변경없이, 2=상,하 순서변경
리턴형	16진수 형태의 문자열
설명	아스키문자열을 Hex Code로 변환해서 문자열 형태로 돌려줍니다.
함수활용	"1" -> "31" ; 아스키코드 '1' 은 16진 코드로 0x31. 0x31 을 "31" 2바이트의 문자열로 돌려줍니다. PLC 디바이스 통신을 할 경우 직접 데이터를 보내야 할 경우 사용하면 유용합니다.

```
/*******************************************************
예제 : 문자열의 전체 코드값 얻어오기
*******************************************************/
void AsciiToHex()
{
	string str;
	string sCode;

	str = "ABCDE";
	sCode = _AsciiToHx(str, 1);
	_TraceEx("str[%s], sCode[%s]", str, sCode);

	sCode = _AsciiToHx(str, 2);
	_TraceEx("str[%s], sCode[%s]", str, sCode);

	/*************************************************************************************************
	_TraceEx를 통해 결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. eRun이 실행되었으면 [F6] 트레이스창 을 활성화 시킨다.
	3. [F7] 함수수동호출 을 활성화 시킨후 본 사용자 정의 함수를 실행시킨다.
	4. 트레이스창의 출력문을 확인한다.
	*************************************************************************************************/
}
```

2.1.3 _ClosePage()

함수원형

void _ClosePage(string strPage)

파라메터	strPage : 닫을 페이지이름. 페이지명 : 페이지 이름을 지정하는 경우는 부모 페이지에서 뷰페이지를 닫을 때 사용합니다 (NULL) : 특정페이지가 아닌 전체 페이지를 닫습니다.(프로그램이 종료됩니다)
리턴형	없음
설명	뷰 페이지 화면을 닫습니다.
함수활용	

```
/*********************************************************
예제 : 뷰페이지 닫기
*********************************************************/
void ClosePage()
{

    // 뷰페이지명이 "서브페이지"를 닫습니다.
    _ClosePage("서브페이지");

    // 현재 뷰페이지를 닫습니다.
    _ClosePage("");
}
```

2.1.4 _CreateFolder()

함수원형

int _CreateFolder(string strFolder)

파라메터	strFolder : 생성 폴더명을 문자열 상수 또는 변수를 지정합니다
리턴형	0 : 성공 >0 : 실패 * 실패시 _GetSystemErrorMsg() 함수를 사용해서 시스템오류 메시지 확인을 할 수 있습니다.
설명	서브폴더 포함해서 폴더를 생성합니다.
함수활용	

```
/*********************************************************
예제 : 폴더생성
*********************************************************/
void CreateFolder()
{
    int nRet;
    string strCreateFolderPath, strProjectPath;
    strProjectPath = "c:/test";

     // 문자열 변수와 문자열을 조합하여 문자열 변수 strCreateFolderPath에 저장합니다.
    strCreateFolderPath = strProjectPath + "/Log";

     // strCreateFolderPath에 저장된 폴더 경로명에 폴더를 생성합니다.
    nRet = _CreateFolder(strCreateFolderPath);

     // 폴더가 정상적으로 생성되었으면 성공 메시지 박스를 표시합니다.
    if(nRet == 0)
        _MsgBox("정상적으로 폴더가 생성되었습니다.", "성공", 0, 2);

     // 폴더 생성이 실패했으면 실패 메시지 박스를 표시합니다.
    else {
        strError = _GetSystemErrorMsg();
        _MsgBox(strError, "fail", 0, 4);
    }
}
```

```
/*******************************************************
예제2 : 폴더생성
*******************************************************/
void CreateFolder2()
{
	int ret;
	string str;

	str ="e:/temp4/temp5" + "/temp6";
	ret = _CreateFolder(str); //"e:₩temp3₩temp3₩temp4");

	_TraceEx("CreateFolder() ret %d, msg=%s", ret, _GetSystemErrorMsg());
}
```

2.1.5 _DeleteFolder()

함수원형

int _DeleteFolder(string strFolder, int nMode)

파라메터	strFolder : 삭제 폴더명을 문자열 상수 또는 변수를 지정합니다. nMode : 0 (비어있는 폴더만 삭제), 1 (서브폴더 및 파일제거 후 폴더삭제)
리턴형	0 : 실패 >0 : 성공 * 실패시 _GetSystemErrorMsg() 함수를 사용해서 시스템오류 메시지 확인을 할 수 있습니다.
설명	폴더 삭제합니다.
함수활용	

```
/*********************************************************
예제 : 폴더삭제
*********************************************************/
void DeleteFolder()
{
    int nRet;
    string strFolderPath, strProjectPath, strError;
    strProjectPath = "c:₩test";

     // 문자열 변수와 문자열을 조합하여 문자열 변수 strFolderPath에 저장합니다.
    strFolderPath = strProjectPath + "₩Log";

     // 서브폴더 포함해서 모두 삭제합니다.
    nRet = _DeleteFolder(strFolderPath, 1);

     // 폴더가 정상적으로 삭제되었으면 성공 메시지 박스를 표시합니다.
    if(nRet != 0) {
        _MsgBox("정상적으로 폴더가 삭제되었습니다.", "성공", 0, 2);

     // 폴더삭제 실패했으면 오류 메시지 박스를 표시합니다.
    } else {
        strError = _GetSystemErrorMsg();
        _MsgBox(strError, "에러", 0, 4);
    }
```

```
        /*****************************************************************************************************
        _TraceEx를 통해 결과를 확인하기 위한 방법은 아래와 같습니다.
        1. c:₩test₩Log 폴더를 미리 생성한다.
        2. eRun을 실행한다 [F5]
        3. eRun이 실행되었으면 [F6] 트레이스창 을 활성화 시킨다.
        4. [F7] 함수수동호출 을 활성화 시킨후 본 사용자 정의 함수를 실행시킨다.
        5. 트레이스창의 출력문을 확인한다.
        6. 윈도우 탐색기를 열어 해당 폴더가 삭제되었는지 확인한다.
        *****************************************************************************************************/
}
```

2.1.6 _EnableControl()

함수원형

void _EnableControl(string strObject, int nFlag)

파라메터	strObject : 뷰페이지에 있는 윈도우 오브젝트 이름입니다. nFlag : 지정된 오브젝트를 활성화 또는 비활성 시킵니다. 0 : 비활성모드로 전환되며 입력, 선택 등의 사용자 설정이 불가능하게 됩니다. 1 : 활성모드로 전환되며, 사용자 설정이 가능하게 됩니다.
리턴형	없음
설명	윈도우 오브젝트의 활성화 상태를 설정합니다.
함수활용	시스템 구동 시 일시적으로(자동운전 등)사용자에 의한 입력을 제한하여 운영상의 문제가 발생되지 않도록 보호하는 기능이 필요한 경우 사용합니다.

※ 윈도우오브젝트 : 그리드, 라인그래프, 현재경보, 입력 창, 콤보박스, 리스트박스, 날짜컨트롤, 버튼, 체크박스, 라디오버튼

```
/*********************************************************
예제 : 윈도우 오브젝트 활성상태 제어하기
*********************************************************/
void EnableControl()
{
// 오브젝트 이름이 "그리드"인 윈도우 오브젝트를 비활성화 시킵니다.
_EnableControl("그리드@뷰페이지명", 0);
}
```

2.1.7 _Execute()

함수원형

void _Execute(string strFile)

파라메터	strFile : 경로명 포함된 실행파일이름
리턴형	없음
설명	외부 프로그램을 실행합니다.
함수활용	계산기 프로그램, 통신모듈 프로그램등을 실행 및 특정 응용 프로그램을 실행하는 경우 사용합니다.

```
/*********************************************************
예제 : 외부 실행파일 실행하기
*********************************************************/
void Execute()
{
	// 계산기를 실행합니다.
	_Execute("C:₩₩Windows₩₩System32₩₩calc.exe");

	// C 드라이브 Temp 폴더 탐색기 열기.
	_Execute("C:₩₩Temp");

	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. eRun이 실행되었으면 [F6] 트레이스창 을 활성화 시킨다.
	3. [F7] 함수수동호출 을 활성화 시킨후 본 사용자 정의 함수를 실행시킨다.
	*************************************************************************************************/
}
```

2.1.8 _ExecuteEx()

함수원형

int _ExecuteEx(string strFile, string strParam, string strDirectory, int nShow)

파라메터	strFile : 경로명 포함된 실행파일이름 strParam : 호출되는 파일에 넘겨줄 파라메터 문자열 형태로 지정. strDirectory : 호출되는 파일의 실행환경 폴더지정. nShow : 표시방법에 해당하는 상수값 지정.
리턴형	정수 (0=정상실행)
설명	외부파일을 실행시키기 위해 사용하는데, 전달 파라메터와 실행 폴더 그리고 표시방법을 같이 설정한다.
함수활용	계산기 프로그램, 통신모듈 프로그램등을 실행 및 특정 응용 프로그램을 실행하는 경우 사용합니다.

■ 외부파일 실행형태 상수값

이름	상수값	설명
HIDE	0	Hides the window and activates another window.
MAXIMIZE	3	Maximizes the specified window.
MINIMIZE	6	Minimizes the specified window and activates the next top - level window in the z - order.
RESTORE	9	Activates and displays the window.If the window is minimized or maximized, Windows restores it to its original size and position.An application should specify this flag when restoring a minimized window.
SHOW	5	Activates the window and displays it in its current size and position.
SHOWDEFAULT	10	Sets the show state based on the SW_ flag specified in the STARTUPINFO structure passed to the CreateProcess function by the program that started the application.An application should call ShowWindow with this flag to set the

		initial show state of its main window.
SHOWMAXIMIZED	3	Activates the window and displays it as a maximized window.
SHOWMINIMIZED	2	Activates the window and displays it as a minimized window.
SHOWMINNOACTIVE	7	Displays the window as a minimized window.The active window remains active.
SHOWNA	8	Displays the window in its current state.The active window remains active.
SHOWNOACTIVATE	4	Displays a window in its most recent size and position.The active window remains active.
SHOWNORMAL	1	Activates and displays a window. If the window is minimized or maximized, Windows restores it to its original size and position. An application should specify this flag when displaying the window for the first time.

```
/*********************************************************
예제 : 외부 실행파일 실행하기
*********************************************************/
void ExecuteEx()
{
	int nRet;
	// c:₩Temp 폴더 탐색기를 OPEN합니다.
	// 파일이름 대신 폴더명이 지정되면 해당폴더 탐색기를 OPEN합니다.
	nRet = _ExecuteEx("C:₩temp", "", "", 1);

	// 메모장으로 ExecuteEx.txt 파일을 Open합니다.
	nRet = _ExecuteEx("C:₩Windows₩notepad.exe", "ExecuteEx.txt", 0, 1);

	// 크롬(chrome)부라우저 프로그램을 실행하고, jpeople.co.kr 페이지로 점프합니다.//
```

```
    _ExecuteEx("C:₩₩Program Files (x86)₩₩Google₩₩Chrome₩₩Application₩₩chrome.exe",
    "http://www.jpeople.co.kr", 0, 1);

    /*********************************************************************************************
    결과를 확인하기 위한 방법은 아래와 같습니다.
    1. eRun을 실행한다 [F5]
    2. eRun이 실행되었으면 [F6] 트레이스창 을 활성화 시킨다.
    3. [F7] 함수수동호출 을 활성화 시킨후 본 사용자 정의 함수를 실행시킨다.
    *********************************************************************************************/
}
```

2.1.9 _FocusControl()

함수원형

void _FocusControl(string strObject)

파라메터	strObject : 뷰페이지에 있는 윈도우 오브젝트 이름입니다.
리턴형	없음
설명	콤포넌트 오브젝트 FOCUS 변경을 합니다.
함수활용	

※ 그리드, 현재경보, 라인그래프, 리스트박스, 리스트컨트롤, 콤보박스, 버튼, 체크박스, 라디오버튼, 입력창, 날짜 컨트롤에 대해서만 사용할 수 있습니다.

```
/*********************************************************
예제 : 윈도우 오브젝트 FOCUS 제어하기
*********************************************************/
void FocusControl()
{
	// 오브젝트 명이 "그리드"인 윈도우 오브젝트를 화면에서 숨깁니다.
	_FocusControl("그리드@뷰페이지명", 0);
}
```

2.1.10 _GetDiskFreeSpace()

함수원형

double _GetDiskFreeSpace(string strDrive)

파라메터	strDrive : 디스크 드라이브 문자열 "C:₩", "D:₩"
리턴형	실수값
설명	지정된 하드 드라이브 디스크의 사용가능한 용량을 계산할 때 사용합니다. strDrive의 전체용량을 Kilo bytes로 돌려줍니다.
함수활용	저장장치의 용량을 확인해서 불필요한 파일을 삭제하도록 할 수 있습니다.

※ 리턴값이 Kiro bytes 단위의 값이기 때문에, Mega bytes로 변환하려면 1024로 나누면 되고, Giga bytes로 변환하려면 1024로 한번 더 나누어 주면 된다.

```
fGiga = fTotal / 1024 / 1024;

/*********************************************************
예제 : 디스크 용량 가져오기
*********************************************************/
void GetFreeSpace()
{
double fTotal;
        double fMegaSpace, fGigaSpace;

        fTotal = _GetDiskFreeSpace("C:₩₩");

        fMegaSpace =  fTotal / 1024.0;
        fGigaSpace =  fTotal / 1024 / 1024;

        _TraceEx("C Drive - free disk space %.2f kbytes, %.2f MB, %.2f GB",
                fTotal, fMegaSpace, fGigaSpace);
}
```

2.1.11 _GetDiskTotalSpace()

함수원형

double _GetDiskTotalSpace(string strDrive)

파라메터	strDrive : 디스크 드라이브 문자열 "C:₩", "D:₩" * 네트워크 드라이브, 가상드라이브는 지원하지 않습니다.
리턴형	실수값
설명	지정된 하드드라이브 디스크의 전체용량이 필요할 때 사용합니다. strDrive의 전체용량을 Kilo bytes로 돌려줍니다.
함수활용	저장장치의 용량을 확인하여 불필요한 파일을 삭제 할 수 있도록 할 수 있습니다.

※ 리턴값이 Kiro bytes 단위의 값이기 때문에, Mega bytes로 변환하려면 1024로 나누면 되고, Giga bytes로 변환하려면 1024로 한번 더 나누어 주면 된다.

```
fGiga = fTotal / 1024 / 1024;

/*********************************************************
예제 : 디스크 총용량 가져오기
*********************************************************/
void GetSpace()
{
double fTotal;
        double fMegaSpace, fGigaSpace;

        fTotal = _GetDiskTotalSpace("C:₩₩");

        fMegaSpace =  fTotal / 1024.0;
        fGigaSpace =  fTotal / 1024 / 1024;

        _TraceEx("C Drive - disk space %.2f kbytes, %.2f MB, %.2f GB",
                fTotal, fMegaSpace, fGigaSpace);
}
```

2.1.12 _GetSystemErrorMsg()

함수원형

string _GetSystemErrorMsg()

파라메터	없음
리턴형	문자열
설명	스크립트 함수호출 이후에 결과값이 오류인 경우 시스템 오류메시지를 읽어온다.
함수활용	본 내부함수는 시스템 함수실행 오류발생시 사용합니다.

```
/*********************************************************
예제 : 시스템 오류코드 얻기
*********************************************************/

// NAS 드라이브에 기록.
void NASWrite()
{
	string strFile, strData, strMsg;
	int ret;

/*	if(@COMMON.NAS_CONN == 0) {
		//_Trace("NAS connection fail...");
		ShowLog("NAS connection fail...");
		return;
	}
*/	strFile = _FormatString("Z:₩sf%s.csv", _GetDateFormat("%Y%m%d", 0));
	//
	strData =
_FormatString("%s, %s, %s, %s, %s, %s, %.2f, %.2f, %.2f, %.2f, %.2f, %.2f, %.2f, %.2f, %.2f, %.4f, %.2f, %.2f, %s,
%s, %c%c",
_TimeToString("%Y-%m-%d %H:%M:%S"),
@PLC.JUDGEMENT,
	@PLC.CAR_NAME,
	@PLC.CAR_NO,
	@PLC.BARCODE_F,
	@PLC.BARCODE_R,
	(double)@PLC.ZR2538, (double)@PLC.ZR2532, (double)@PLC.ZR2580,
	(double)@PLC.ZR2582, (double)@PLC.ZR2584, (double)@PLC.ZR2586,
	(double)@PLC.ZR2588, (double)@PLC.ZR2590, (double)@PLC.ZR2592,
(double)@PLC.ZR2594,
```

```
		(double)@PLC.ZR2534,(double)@PLC.ZR2536, @PLC.MARKING, @PLC.MARKING2, 0x0d, 0x0a);

		_Trace(strData);

		ret = _LogToFileEx(strFile, "%s", strData);
		if(ret != 1) {
			//
			// 정상이면 1, 아니면 -1이 돌아온다.
			@COMMON.NAS_WRITE = 0;
			strMsg = _FormatString("_LogToFileEx()..ret=%d, error_code = %d, err_stat=%s", ret,
_GetSystemErrorNo(), _GetSystemErrorMsg());
			ShowLog(strMsg);
		}
		else {
			// 정상이면 1, 아니면 -1이 돌아온다.
			@COMMON.NAS_WRITE = 1;

			strMsg = _FormatString("%s, %s", strFile, strData);
			ShowLog(strMsg);
		}

}
```

2.1.13 _GetSystemErrorNo()

함수원형

int _GetSystemErrorNo()

파라메터	없음
리턴형	없음
설명	스크립트 함수호출 이후에 결과값이 오류인 경우 시스템 오류번호를 읽어온다.
함수활용	본 내부함수는 시스템 함수실행 오류발생시 사용합니다.

```
/*********************************************************
예제 : 시스템 오류코드 얻기
*********************************************************/

// NAS 드라이브에 기록.
void NASWrite()
{
	string strFile, strData, strMsg;
	int ret;

/*	if(@COMMON.NAS_CONN == 0) {
		//_Trace("NAS connection fail...");
		ShowLog("NAS connection fail...");
		return;
	}
*/	strFile = _FormatString("Z:₩sf%s.csv", _GetDateFormat("%Y%m%d", 0));
	//
	strData =
_FormatString("%s, %s, %s, %s, %s, %s, %.2f, %.2f, %.2f, %.2f, %.2f, %.2f, %.2f, %.2f, %.2f, %.4f, %.2f, %.2f, %s,
%s, %c%c",
_TimeToString("%Y-%m-%d %H:%M:%S"),
@PLC.JUDGEMENT,
	@PLC.CAR_NAME,
	@PLC.CAR_NO,
	@PLC.BARCODE_F,
	@PLC.BARCODE_R,
	(double)@PLC.ZR2538, (double)@PLC.ZR2532, (double)@PLC.ZR2580,
	(double)@PLC.ZR2582, (double)@PLC.ZR2584, (double)@PLC.ZR2586,
	(double)@PLC.ZR2588, (double)@PLC.ZR2590, (double)@PLC.ZR2592,
(double)@PLC.ZR2594,
```

```
		(double)@PLC.ZR2534,(double)@PLC.ZR2536, @PLC.MARKING, @PLC.MARKING2, 0x0d, 0x0a);

		_Trace(strData);

		ret = _LogToFileEx(strFile, "%s", strData);
		if(ret != 1) {
			//
			// 정상이면 1, 아니면 -1이 돌아온다.
			@COMMON.NAS_WRITE = 0;
			strMsg = _FormatString("_LogToFileEx()..ret=%d, error_code = %d, err_stat=%s", ret,
_GetSystemErrorNo(), _GetSystemErrorMsg());
			ShowLog(strMsg);
		}
		else {
			// 정상이면 1, 아니면 -1이 돌아온다.
			@COMMON.NAS_WRITE = 1;

			strMsg = _FormatString("%s, %s", strFile, strData);
			ShowLog(strMsg);
		}

}
```

2.1.14 _HexToDec()

함수원형

int _HexToDec(string strHex, int nRetType)

float _HexToDec(string strHex, int nRetType)

파라메터	strHex : 16진수 형태의 문자열 nRetType : 정수 또는 부동소수형의 되돌려받는 형식 0 : 정수로 돌려받음 1 : 부동소수형으로 변환해서 돌려받음
리턴형	32bit 부호없는 정수 또는 부동소수(FLOAT)
설명	16진수(Hex) 형태의 문자열을 10진수(Decimal) 값으로 돌려준다. 정수형, 부동소수형으로 돌려준다.
함수활용	

```
/*********************************************************
예제 : Hex 값을 Decimal값으로 변환
*********************************************************/
void HexToDec()
{
	int nValue;
	double fValue;

	string strHex;
	string strAscii;

	strHex = "3fa5d9c3";
	nValue = _HexToDec(strHex, 0);
	fValue = _HexToDec(strHex, 1);

	//strAscii = _HxToAscii(strHex, 1);

	_TraceEx("_HexToDec(%s, 0) : %d ", strHex, nValue);
	_TraceEx("_HexToDec(%s, 1) : %f", strHex, fValue);

	_TraceEx("_NumToStr(%d, 16) : %s", nValue, _NumToStr(nValue, 16));
	_TraceEx("_NumToStr(float-%f, 16) : %s", fValue, _NumToStr((float)fValue, 16));
	_TraceEx("_NumToStr(double-%f, 16) : %s", fValue, _NumToStr(fValue, 16));
```

```
    /*************************************************************************************************
    결과를 확인하기 위한 방법은 아래와 같습니다.
    1. eRun을 실행한다 [F5]
    2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
    3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
    *************************************************************************************************/
}
```

[실행결과] F6을 눌러서 스크립트 Trace View창을 열어서 확인한다.

```
[2022-12-06 16:43:19.523]  _HexToDec(3fa5d9c3, 0) : 1067833795
[2022-12-06 16:43:19.523]  _HexToDec(3fa5d9c3, 1) : 1.295708
[2022-12-06 16:43:19.524]  _NumToStr(1067833795, 16) : 3fa5d9c3
[2022-12-06 16:43:19.525]  _NumToStr(float-1.295708, 16) : 3FA5D9C3
[2022-12-06 16:43:19.525]  _NumToStr(double-1.295708, 16) : n/a
```

2.1.15 _HiByte()

함수원형

int _HiByte(int nValue)

파라메터	nValue : 16비트의 정수
리턴형	정수형
설명	16비트 값에서 상위 바이트(8bit)의 값을 돌려줍니다.
함수활용	통신에서 송신 횟수를 줄이기 위해서 8비트 데이터를 각각 상,하위에(16bit) 실어서 보내는 경우 상위 바이트의 데이터를 추출할 때 사용합니다.

```
/*********************************************************
예제 : 상,하위 바이트 추출
*********************************************************/
void GetLoHiByte()
{
	int nValue, nLo, nHi;

	nValue = 0x1234;

	// 하위바이트 추출.
	nLo = _LoByte(nValue);

	// 상위바이트 추출.
	nHi = _HiByte(nValue);

	// -> "1234, Lobyte : 34, HiByte : 12"
	_TraceEx("%x, Lobyte : %x, HiByte : %x", nValue, nLo, nHi);
	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
	*************************************************************************************************/
}
```

2.1.16 _HiWord()

함수원형

int _HiWord(int nValue)

파라메터	nValue : 32비트의 정수
리턴형	정수형
설명	32비트 값에서 상위 워드(16bit)의 값을 돌려줍니다.
함수활용	통신에서 송신 횟수를 줄이기 위해서 16비트 데이터를 각각 상,하위에(32bit) 실어서 보내는 경우 상위 바이트의 데이터를 추출할 때 사용합니다.

```
/*********************************************************
예제 : 상, 하위 워드(16bit) 추출
*********************************************************/
void GetLoHiWord()
{
	int nValue, nLo, nHi;

	nValue = 0x12345678;

	// 하위워드 추출.
	nLo = _LoWord(nValue);

	// 상위워드 추출.
	nHi = _HiWord(nValue);

	// -> "1234, Lobyte : 5678, HiByte : 1234"
	_TraceEx("%x, LoWord : %x, HiWord : %x", nValue, nLo, nHi);
	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
	*************************************************************************************************/
}
```

2.1.17 _HxToAscii()

함수원형

string _HxToAscii(string strHex, int nOrder)

파라메터	strHexValue : 16진수 코드값 문자열 nOrder : 상,하바이트 순서를 지정 1 : 코드값 순서로 2 : 코드값 상,하 순서변경
리턴형	아스키문자열
설명	16진수(Hex) 코드값으로 이루어진 문자열을 아스키 문자열로 변환한다.
함수활용	PLC 디바이스 통신을 할 경우 직접 데이터를 보내야 할 경우 사용하면 유용합니다. 16진 코드값에 해당하는 아스키문자를 가져올 때 유용합니다.

```
/*********************************************************
예제 : 16진코드 -> 아스키문자
*********************************************************/
void HxToAscii()
{
	string strHexValue;
	string strAscii1, strAscii2;

	strHexValue = "31323334";
	strAscii1 = _HxToAscii(strHexValue, 1);		//  => "1234"
	_TraceEx("1234 => %s", strAscii1);

	// 1번째 바이트와 2번째 바이트순서 변경
	strAscii2 = _HxToAscii(strHexValue, 2);		//  => "2143"
	_TraceEx("1234 => %s", strAscii2);
	/*********************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
	*********************************************************************************************/
}
```

2.1.18 _IsAscii()

함수원형

int _IsAscii(string strText)

파라메터	strText : 문자열 상수 또는 변수
리턴형	정수형 0 : 주어진 데이터에 Ascii(0x20 ~ 0x7E)문자열 이외의 다른 데이터가 있는 경우 1 : 주어진 데이터에 Ascii(0x20 ~ 0x7E)문자열만 있는 경우
설명	주어진 문자열 파라메터에 Ascii(0x20 ~ 0x7E) 문자열이 있는지 검사합니다.
함수활용	I/O 통신으로 수신된 문자열 데이터에 제어코드(리턴값 0에 해당하는 값)가 있는지 확인하는 경우 사용합니다.

```
/*******************************************************
예제 : 아스키문자 확인
*******************************************************/
void IsAscii()
{
	string str[2];
	int i;
    int nRet;

    // 문자열 "abcde"에 문자이외의 다른 데이터(제어코드)가 있는지 확인합니다.
	str[0] = _FormatString("abcde%c", 0x0d);
	str[1] = "abcde";

	i = 0;
	while (i < 2)
	{
		nRet = _IsAscii(str[i]);
		// 문자 이외의 다른 데이터가 있으면 경고 메시지 표시
		if(nRet == 0)
			_MsgBox("데이터에 문자 이외의 다른 데이터가 포함되어있습니다.", "결과", 0, 2);

		 // 문자만 있으면 확인 메시지 표시
		else
			_MsgBox("데이터에 문자만 있습니다.", "결과", 0, 2);

		i++;
	}
```

```
        /*****************************************************************************************
        결과를 확인하기 위한 방법은 아래와 같습니다.
        1. eRun을 실행한다 [F5]
        2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
        *****************************************************************************************/
}
```

2.1.19 _IsNumeric()

함수원형

int _IsNumeric(string strText)

파라메터	strText : 문자열 상수 또는 변수
리턴형	정수형 0 : 주어진 데이터에 -,+,소수점, 숫자 이외의 다른 데이터가 있는 경우 1 : 주어진 데이터에 -,+,소수점, 숫자만 있는 경우
설명	주어진 문자열 파라메터에 숫자만 존재하는지 검사합니다.
함수활용	통신으로 수신된 데이터 또는 데이터베이스에서 쿼리해온 데이터에 숫자이외 다른 문자가 있는지 체크하는 경우에 사용합니다

```
/*********************************************************
예제 : 숫자확인
*********************************************************/
void IsNumeric()
{
	string str[2];
	int i;
    int nRet;

    // 문자열 "abcde"에 문자이외의 다른 데이터(제어코드)가 있는지 확인합니다.
	str[0] = "123.456";
	str[1] = "ABCD1234";

	i = 0;
	while (i < 2)
	{
		nRet = _IsNumeric(str[i]);
		// 문자 이외의 다른 데이터가 있으면 경고 메시지 표시
		if(nRet == 0)
			_MsgBox("데이터에 숫자 이외의 다른 데이터가 포함되어있습니다.", "결과", 0, 2);

		 // 문자만 있으면 확인 메시지 표시
		else
			_MsgBox("데이터에 숫자만 있습니다.", "결과", 0, 2);

		i++;
	}
```

```
	/*********************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	*********************************************************************************************/
}
```

2.1.20 _IsOpenPage()

함수원형

int _IsOpenPage(string strPage)

파라메터	strPage : 뷰페이지명을 상수 또는 변수를 지정합니다.
리턴형	정수값 -1 : 페이지명이 잘못 입력되었음 0 : 페이지가 열려있지 않음. 1 : 페이지가 열려있음(화면에 보여짐). 2 : 페이지가 열렸지만 숨어있음(화면에 안보임).
설명	경보 등 특정 이벤트 발생에 따라 뷰페이지가 팝업되어 오픈시켜야 하는 경우 페이지가 열려있지 않은 경우 오픈시키는 경우에 사용합니다.
함수활용	특정 뷰페이지가 열렸는지 확인하기

```
/*********************************************************
예제 : 뷰페이지 열린상태 시험
*********************************************************/
void IsOpenPage()
{
	string strPage;
	int nRet;
	strPage = "테스트페이지";

	// strPage 뷰페이지 열린상태.
	nRet = _IsOpenPage(strPage);

	// 페이지가 이미 열려있으면 확인 메시지 박스 표시
	if(nRet == 1)
	_MsgBox("페이지가 열려있습니다.", "확인", 0, 2);

	// 페이지가 열려있지 않으면 확인 메시지 박스 표시
	else if(nRet == 0)
	_MsgBox("페이지가 열려있지 않습니다.", "확인", 0, 2);

	// 페이지가 없으면 경고 메시지 박스 표시
	else if(nRet == -1)
	_MsgBox("페이지 명이 잘못되었습니다.", "확인", 0, 2);
}
```

2.1.21 _LoByte()

함수원형

int _LoByte(int nValue)

파라메터	nValue : 16비트의 정수
리턴형	정수형
설명	16비트 값에서 하위 바이트(8bit)의 값을 돌려줍니다.
함수활용	통신에서 송신 횟수를 줄이기 위해서 8비트 데이터를 각각 상,하위에(16bit) 실어서 보내는 경우 하위 바이트의 데이터를 추출할 때 사용합니다.

```
/*********************************************************
예제 : 하위 바이트 추출
*********************************************************/
void GetLoHiByte()
{
	int nValue, nLo, nHi;

	nValue = 0x1234;

	// 하위바이트 추출.
	nLo = _LoByte(nValue);

	// 상위바이트 추출.
	nHi = _HiByte(nValue);

	// -> "1234, Lobyte : 34, HiByte : 12"
	_TraceEx("%x, Lobyte : %x, HiByte : %x", nValue, nLo, nHi);

	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
	*************************************************************************************************/
}
```

2.1.22 _Lock()

함수원형

void _Lock(int nCode)

파라메터	nCode : 잠금 임의코드값 (1~79)
리턴형	없음
설명	하나의 자원(파일읽기/쓰기 등)에 동시접근을 막기 위해 자원접근 시 하나의 프로세스만 접근 할 수 있게 설정합니다.
함수활용	한개이상의 실행함수 및 프로세스에서 동일한 파일에 접근하여 파일쓰기를 하는 경우 서로 다른 실행함수 및 프로세스에서 동시에 접근하게 되면 파일 오류 및 정상적인 처리가 안 되는 것을 막기 위해 사용합니다.

※ 주의 사항

Lock 내부함수는 항상 UnLock과 사용해야 합니다.

Lock이 되면 UnLock이 되기 전까지는 다른 프로세스에서 접근을 할 수 없습니다.

또한 Lock함수의 매개변수에 쓰이는 고유번호가 UnLock에서 동일하게 사용되어야 합니다.

동시 접근에 위험성이 있는 자원들에 대해서만 사용하시기 바랍니다.

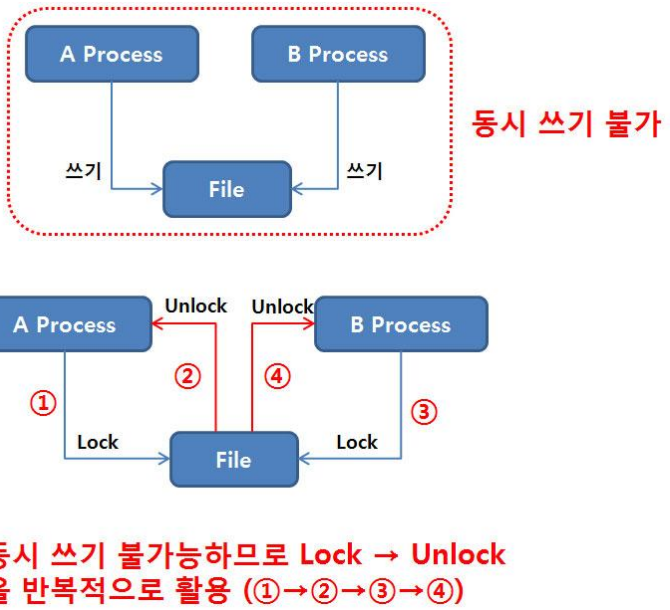
위의 예제에서 하나의 파일에 데이터를 쓰는 경우 또는 데이터베이스에 동시접근성이 있는 경우, 윈도우 오브젝트에 접근하는 경우 등 사용하시기 바랍니다.

동일한 고유번호로 서로 다른 프로세스에 Lock을 설정하게 되면 하나의 프로세스가 UnLock이 되기 전까지는 다른 프로세스에서 접근을 하지 않고 대기를 하고 있습니다.

```
/*********************************************************
예제 : 두 개의 함수에서 하나의 파일에 데이터 쓰기
    두 개의 함수는 Function Script 오브젝트에서 각각 호출
    파일은 최초에 열렸고 FD 번호는 태그에 있다고 가정.
*********************************************************/
void FuncA()
{
        int nFD;
        string strValue;
        strValue = "SEFA";
        // 파일 오픈시 생성된 파일번호로 태그명 @DEMO.FileFD에서 갖고 와서 정수형 변수 nFD에 저장합니다.
        nFD = @DEMO.FileFD;
        // 10번이란 키로 Lock을 설정하여 다른 프로세스에서 파일을 제어할 수 없게 합니다..
        _Lock(10);
        // 정수형 변수 nFD에 저장된 파일번호의 파일에 문자열 변수 strValue에 저장된 값(SEFA)를 파일에 씁니다.
```

```
        _FileWrite(nFD, strValue, 4);
        // 10번이란 키의 Lock을 해제하여 다른 프로세스에서 파일을 제어할 수 있게 합니다..
        _Unlock(10);
}

void FuncB()
{
        int nFD;
        string strValue;
        strValue = "Technology";
        // 파일 오픈시 생성된 파일번호로 태그명 @DEMO.FileFD에서 갖고 와서 정수형 변수 nFD에 저장합니다.
        nFD = @DEMO.FileFD;
        // 10번이란 키로 Lock을 설정하여 다른 프로세스에서 파일을 제어할 수 없게 합니다..
        _Lock(10);
        // 정수형 변수 nFD에 저장된 파일번호의 파일에 문자열 변수 strValue에 저장된 값(Technology)를 파일에 씁니다.
        _FileWrite(nFD, strValue, 10);
        // 10번이란 키의 Lock을 해제하여 다른 프로세스에서 파일을 제어할 수 있게 합니다..
        _Unlock(10);
}
```

■ Lock & Unlock 동작에 대한 이해를 돕기위한 그림입니다.



2.1.23 _Loginout()

함수원형

int _Loginout(int nFlag)

파라메터	nFlag : 로그인 / 로그아웃 / 암호확인 0 : 로그인 1 : 로그아웃 2 : 암호확인
리턴형	정수형 -1 : 사용자 정보 불일치. 0 : 입력취소 1 : 사용자 정보 일치
설명	사용자 접근에 제한이 필요한 경우 정보에서 찾습니다.
함수활용	보안을 요구하는 상황에 있을 경우 이 스크립트 함수를 통해서 접근제한을 할 수 있습니다. 그리고 매개변수에는 3가지의 유형이 있습니다. 페이지에 들어갈때 _Loginout(0)을 사용하고 반대로 나올때는 _Loginout(1)을 사용하면 됩니다. 접근암호만을 요구할 경우는 _Loginout(3)을 사용하면 됩니다. 이때 사용자명과 암호는 프로젝트 사용자정보에 등록되어 있어야 합니다. 로그인을 할 경우는 등록되어있는 사용자정보 어떤것이든 사용해도 상관없지만 로그아웃, 암호요구를 사용할 경우는 로그인된 사용자정보를 가지고 체크합니다.

```
/*********************************************************
예제 : 로그인, 로그아웃, 암호확인
*********************************************************/
void Login()
{
	int nRet;
	nRet = _Loginout(0);

	if(nRet == -1) {
		_Trace("사용자 정보가 틀렸습니다.");
	}
	else if(nRet == 0) {
		_Trace("입력을 취소하였습니다.");
	}
	else if(nRet == 1) {
		_Trace("사용자 정보가 일치합니다.");
	}
	/*****************************************************************************************
```

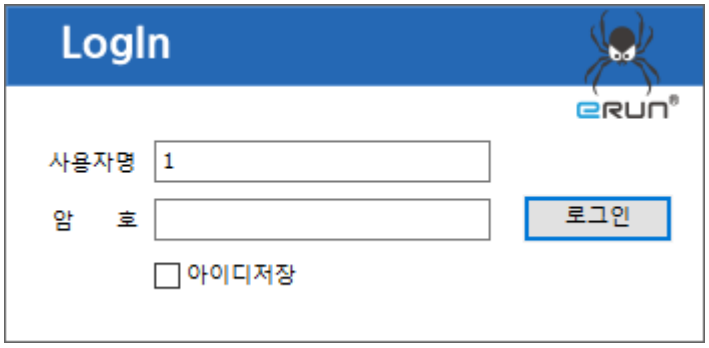
결과를 확인하기 위한 방법은 아래와 같습니다.

1. eRun을 실행한다 [F5]
2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.

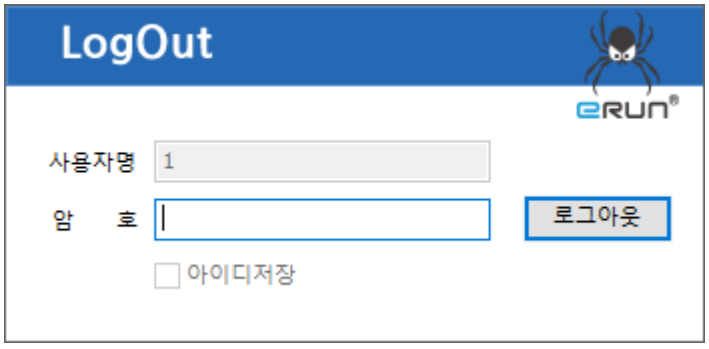
**********/

}

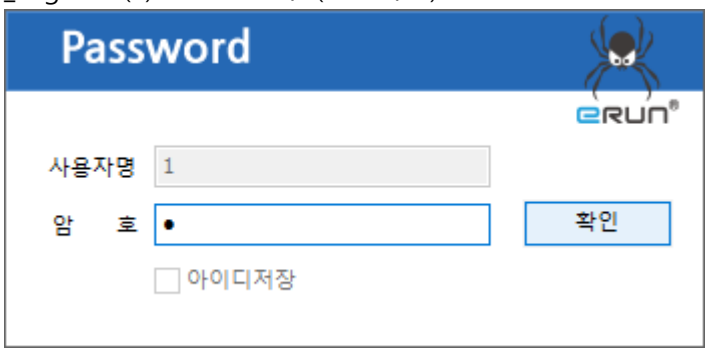
_Loginout(0) 호출된 경우 (로그인)



_Loginout(1) 호출된 경우 (로그아웃)



_Loginout(2) 호출된 경우 (암호확인)



2.1.24 _LoWord()

함수원형

int _LoWord(int nValue)

파라메터	nValue : 32비트의 정수
리턴형	정수형
설명	32비트 값에서 하위 워드(16bit)의 값을 돌려줍니다.
함수활용	통신에서 송신 횟수를 줄이기 위해서 16비트 데이터를 각각 상,하위에(32bit) 실어서 보내는 경우 하위 바이트의 데이터를 추출할 때 사용합니다.

```
/*********************************************************
예제 : 상, 하위 워드(16bit) 추출
*********************************************************/
void GetLoHiWord()
{
	int nValue, nLo, nHi;

	nValue = 0x12345678;

	// 하위워드 추출.
	nLo = _LoWord(nValue);

	// 상위워드 추출.
	nHi = _HiWord(nValue);

	// -> "1234, Lobyte : 5678, HiByte : 1234"
	_TraceEx("%x, LoWord : %x, HiWord : %x", nValue, nLo, nHi);
	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
	*************************************************************************************************/
}
```

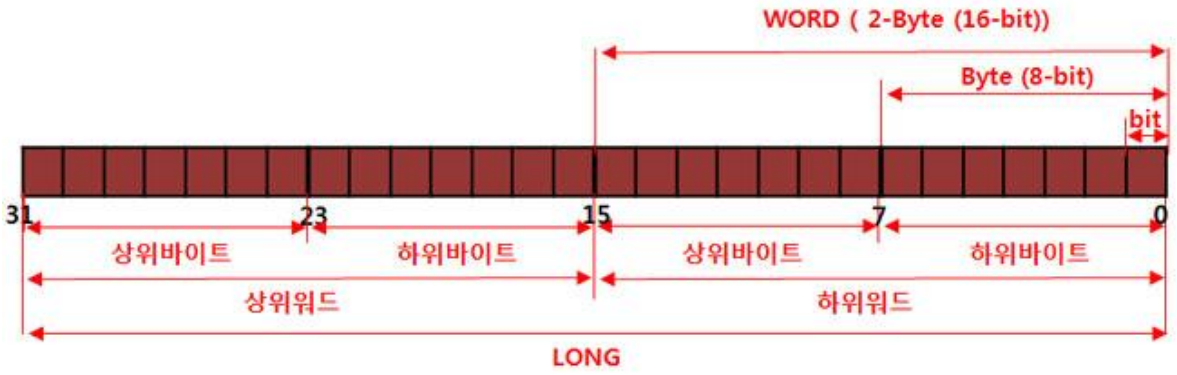
2.1.25 _MakeLong()

함수원형

int _MakeLong(int nLo, int nHi)

파라메터	nLo : 16비트의 정수 nHi : 16비트의 정수
리턴형	정수형
설명	16비트값 2개를 조합하여 Long(32bit)형 데이터로 변환합니다.
함수활용	통신 프로토콜에 따라 2개의 워드데이터를 조합하여 하나의 Long(32bit)데이터를 디바이스 측으로 송신해야 하는 경우에 사용합니다.

※ Long형이란?




```
/*********************************************************
예제 : 2개의 16비트값을 32bit 값으로 조합
*********************************************************/
void MakeLong()
{
int nLo, nHi, nLong;

        nLo = 0x1234;
        nHi = 0x5678;

        //  nLo 16bit 정수값, nHi 16bit 정수값을 조합하여 Long(32bit)형으로.
        nLong = _MakeLong(nLo, nHi);

        _TraceEx("_MakeLong(0x1234, 0x5678) : %08X, LoWord : %04X, HiWord : %04X", nLong, nLo, nHi);
        // ->_MakeLong(0x1234, 0x5678) : 56781234, LoWord : 1234, HiWord : 5678
        /*********************************************************************************************
        결과를 확인하기 위한 방법은 아래와 같습니다.
```

```
        1. eRun을 실행한다 [F5]
        2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
        3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
        ************************************************************************************************/
}
```

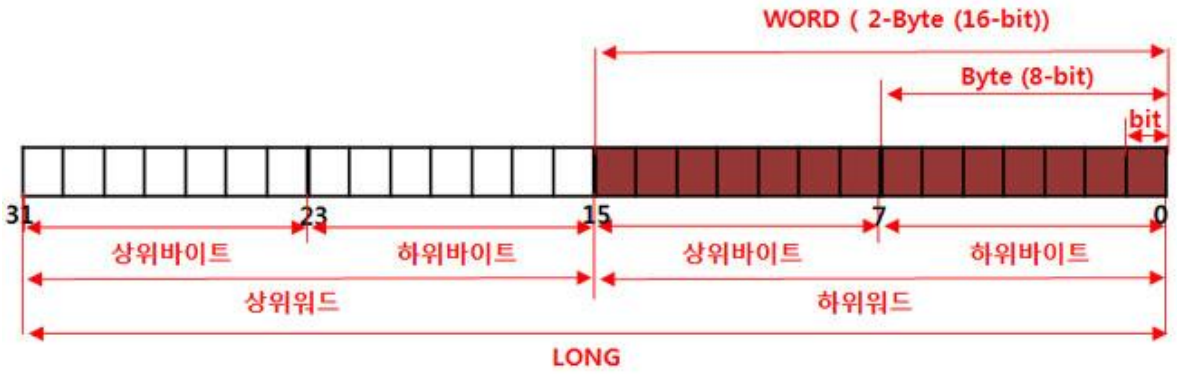
2.1.26 _MakeWord()

함수원형

int _MakeWord(int nLo, int nHi)

파라메터	nLo : 8비트의 정수 nHi : 8비트의 정수
리턴형	정수형
설명	8비트값 2개를 조합하여 Word(16bit)형 데이터로 변환합니다.
함수활용	통신 프로토콜에 따라 2개의 바이트 데이터를 조합하여 하나의 워드형 데이터를 디바이스 측으로 송신해야 하는 경우에 사용합니다.

※ Word형이란?




```
/*********************************************************
예제 : 2개의 8비트값을 16bit 값으로 조합
*********************************************************/
void MakeWord()
{
	int nLo, nHi, nWord;
	nLo = 0xAB;
	nHi = 0x45;

	// nLo 8bit 정수값, nHi 8bit 정수값을 조합하여 Word(16bit)형으로.
	nWord = _MakeWord(nLo, nHi);

	_TraceEx("_MakeWord(0xAB, 0x45) : %04X, LoByte : %02X, HiByte : %02X", nWord, nLo, nHi);
	// -> _MakeWord(0xAB, 0x45) : 45AB, LoByte : AB, HiByte : 45
	/*****************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
```

```
        2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
        3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
        ************************************************************************************************/
}
```

2.1.27 _MsgBox()

함수원형

int _MsgBox(int strText, string strCaption, int nType, int nIcon)

int _MsgBox(double strText, string strCaption, int nType, int nIcon)

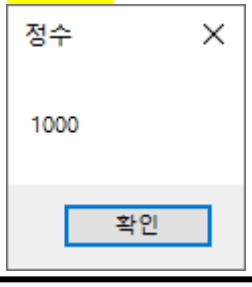
int _MsgBox(string strText, string strCaption, int nType, int nIcon)

int _MsgBox(@TAG.NAME, string strCaption, int nType, int nIcon)

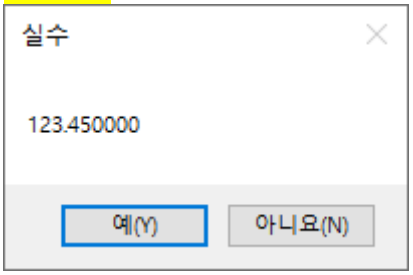
파라메터	strText : 표시내용을 문자열,정수,실수 상수 또는 변수를 지정합니다. strCaption : 제목표시 내용을 문자열 상수 또는 변수를 지정합니다. nTytpe : 사용자 입력 버튼에 대한 번호를 지정합니다. (0, 1, 2) nIcon : 아이콘 번호를 지정합니다. (0,1,2,3,4) ※
리턴형	정수형
설명	메시지 박스를 화면에 표시 후 응답을 기다립니다.
함수활용	사용자 선택을 기다리고 확인후에 다음단계를 처리해야 하는 경우 사용합니다.

주의) _MsgBox 호출 후 사용자가 응답을 할 때까지 다른 작업은 대기합니다.

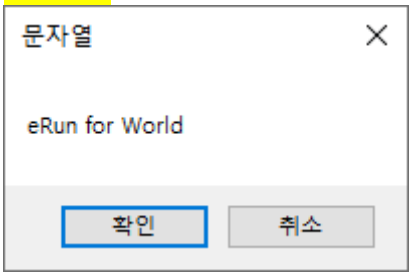
```
/*********************************************************
예제 : 메시지 박스 사용
*********************************************************/
//
void MsgBox()
{
        int nValue;
        string strText;
        double fValue;

        nValue = 1000;
        fValue = 123.45;
        strText = "eRun for World";

        _MsgBox(nValue, "정수", 0, 0);
```



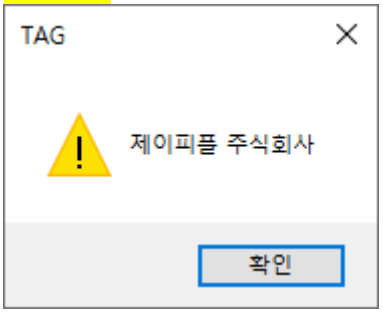
_MsgBox(fValue, "실수", 1, 0);



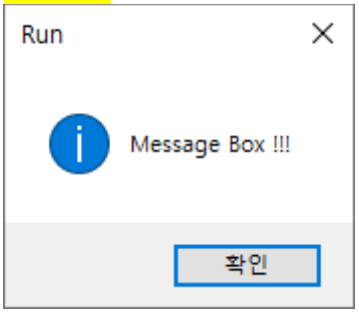
_MsgBox(strText, "문자열", 2, 0);



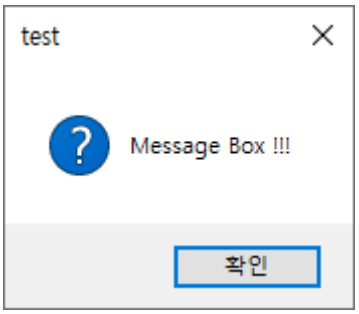
_MsgBox("Message Box !!!", "", 0, 1);



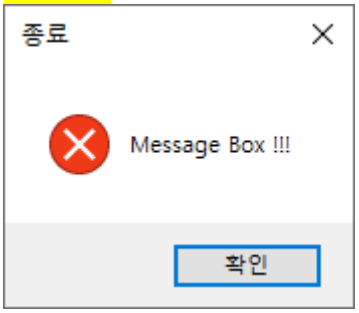
_MsgBox("Message Box !!!", "Run", 0, 2);



_MsgBox("Message Box !!!", "test", 0, 3);




```
_MsgBox("Message Box !!!", "종료", 0, 4);
```




```
/*********************************************************************************************
결과를 확인하기 위한 방법은 아래와 같습니다.
1. eRun을 실행한다 [F5]
2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
*********************************************************************************************/
}
```

2.1.28 _NASConnect()

함수원형

void _NASConnect(string strShareFolder, string strUser, string strPass, string strDrive)

파라메터	strShareFolder : 공유된 네트워크 주소 strUser : 사용자 이름 strPass : 접근 암호 strDrive : 가상 드라이브 문자열 ("Z:") ※ nRet = _NASConnect("₩₩₩₩192.168.0.84₩₩test", "admin", "000000", "K:");
리턴형	정수값
설명	NAS(Network Attached Storage) 저장장치를 가상드라이브로 연결하는 내부함수 입니다.
함수활용	네트워크 드라이브를 통해 파일공유 등을 필요로 할 때 사용합니다.

※ NAS 드라이브는 제공업체 매뉴얼을 참고해서 설정해 주어야 합니다.
※ NAS 드라이브에 연결된 이후에 파일처리 함수 이용방법으로 파일처리 하면 됩니다.

```
/*********************************************************
예제 : NAS 드라이브 사용
*********************************************************/
void NASConnect()
{
	int ret, nFile;
	string str;

	// ret = _NASConnect("₩₩192.168.0.84₩test", "admin", "000000", "K:");
	ret = _NASConnect(@COMMON.NAS_FOLDER, @COMMON.NAS_USER, @COMMON.NAS_PASS,
@COMMON.NAS_DRIVE);
	_TraceEx("_NASConnect=%d", ret);

	if(ret ==0) {
		str = _FormatString("NAS OK. (%s%s)", @COMMON.NAS_DRIVE, @COMMON.NAS_FOLDER);
		ShowLog(str);
		// 정상이면 1, 아니면 0.
		@COMMON.NAS_CONN = 1;
	}
	else {
		// 정상이면 1, 아니면 0.
```

```
                @COMMON.NAS_CONN = 0;
                str = _GetSystemErrorMsg();
                ShowLog(str);

                //_TraceEx("_NASConnect=%d", ret);

                //ret =_FileAccess("Z:₩data2", 0);
                //_TraceEx("File exist =%d", ret);

                //ret = _CreateFolder("Z:₩data2");
                //_TraceEx("CreateFolder..%d", ret);
        }
}

//
void NASDisconnect()
{
        // 1: 무조건 연결끊는다.
        _NASDisconnect("Z:", 1);
}

//
void NASReconnect()
{
        NASDisconnect();
        NASConnect();
}


// NAS 드라이브에 기록.
void NASWrite()
{
        string strFile, strData, strMsg;
        int ret;

/*      if(@COMMON.NAS_CONN == 0) {
                //_Trace("NAS connection fail...");
                ShowLog("NAS connection fail...");
                return;
        }
```

```
*/		strFile = _FormatString("Z:₩sf%s.csv", _GetDateFormat("%Y%m%d", 0));
		//
		strData =
_FormatString("%s, %s, %s, %s, %s, %s, %.2f, %.2f, %.2f, %.2f, %.2f, %.2f, %.2f, %.2f, %.4f, %.2f, %.2f, %s,
%s, %c%c",
_TimeToString("%Y-%m-%d %H:%M:%S"),
@PLC.JUDGEMENT,
		@PLC.CAR_NAME,
		@PLC.CAR_NO,
		@PLC.BARCODE_F,
		@PLC.BARCODE_R,
		(double)@PLC.ZR2538, (double)@PLC.ZR2532, (double)@PLC.ZR2580,
		(double)@PLC.ZR2582, (double)@PLC.ZR2584, (double)@PLC.ZR2586,
		(double)@PLC.ZR2588, (double)@PLC.ZR2590, (double)@PLC.ZR2592,
(double)@PLC.ZR2594,
		(double)@PLC.ZR2534,(double)@PLC.ZR2536, @PLC.MARKING, @PLC.MARKING2, 0x0d, 0x0a);

		_Trace(strData);

		ret = _LogToFileEx(strFile, "%s", strData);
		if(ret != 1) {
			//
			// 정상이면 1, 아니면 -1이 돌아온다.
			@COMMON.NAS_WRITE = 0;
			strMsg = _FormatString("_LogToFileEx()..ret=%d, err_stat=%s", ret, _GetSystemErrorMsg());
			ShowLog(strMsg);
		}
		else {
			// 정상이면 1, 아니면 -1이 돌아온다.
			@COMMON.NAS_WRITE = 1;

			strMsg = _FormatString("%s, %s", strFile, strData);
			ShowLog(strMsg);
		}
}

void ShowLog(string msg)
{
		int nIndex;
		string strObject, str;
```

```
	strObject = "LOGLIST@메인페이지";

	// 리스트박스의 현재 리스트 개수가 50개가 넘으면 제일 위에 있는 리스트 하나 삭제.
	nIndex = _ListGetCount(strObject);
	if(nIndex > 50)
		_ListDelValue(strObject, 0);

	str = _FormatString("%s > %s", _TimeToString("%Y-%m-%d %H:%M:%S"), msg);
	// 로그 텍스트 리스트박스에 추가 표시.
	_ListAddValue(strObject, str);

	nIndex = _ListGetCount(strObject);
	_ListSetSel(strObject, nIndex-1);
}
```

2.1.29 _NASDisconnect()

함수원형

int _NASDisconnect(string strDrive, int nFlag)

파라메터	strDrive : 가상 드라이브 문자열 ("Z:") nFlag : 연결 끊기 방법 0 : 드라이브 사용 중이면 유지. 1 : 연결에 상관없이 연결종료 ※ nRet = _NASDisonnect("K:", 1);
리턴형	정수값 0 : 정상 0보다 큰수 : 오류 ※ _GetSystemErrorMsg() 함수로 오류메시지 확인 가능합니다.
설명	NAS(Network Attached Storage) 저장장치 연결을 끊습니다.
함수활용	네트워크 드라이브를 통해 파일공유 등을 필요로 할 때 사용합니다.

※ NAS 드라이브는 제공업체 매뉴얼을 참고해서 설정해 주어야 합니다.

```
/*********************************************************
예제 : NAS 드라이브 사용
*********************************************************/
void NASDisconnect()
{
	// 1: 무조건 연결끊는다.
	_NASDisconnect("Z:", 1);
}

void NASConnect()
{
	int ret, nFile;
	string str;

	// ret = _NASConnect("₩₩192.168.0.84₩test", "admin", "000000", "K:");
	ret = _NASConnect(@COMMON.NAS_FOLDER, @COMMON.NAS_USER, @COMMON.NAS_PASS,
@COMMON.NAS_DRIVE);
	_TraceEx("_NASConnect=%d", ret);

	if(ret ==0) {
```

```
                str = _FormatString("NAS OK. (%s%s)", @COMMON.NAS_DRIVE, @COMMON.NAS_FOLDER);
                ShowLog(str);
                // 정상이면 1, 아니면 0.
                @COMMON.NAS_CONN = 1;
        }
        else {
                // 정상이면 1, 아니면 0.
                @COMMON.NAS_CONN = 0;
                str = _GetSystemErrorMsg();
                ShowLog(str);

                //_TraceEx("_NASConnect=%d", ret);

                //ret =_FileAccess("Z:₩data2", 0);
                //_TraceEx("File exist =%d", ret);

                //ret = _CreateFolder("Z:₩data2");
                //_TraceEx("CreateFolder..%d", ret);
        }
}

//

//
void NASReconnect()
{
        NASDisconnect();
        NASConnect();
}


// NAS 드라이브에 기록.
void NASWrite()
{
        string strFile, strData, strMsg;
        int ret;

/*      if(@COMMON.NAS_CONN == 0) {
                //_Trace("NAS connection fail...");
                ShowLog("NAS connection fail...");
```

```
                return;
        }
*/      strFile = _FormatString("Z:₩sf%s.csv", _GetDateFormat("%Y%m%d", 0));
        //
        strData                                                                                                 =
_FormatString("%s, %s, %s, %s, %s, %s, %.2f, %.2f, %.2f, %.2f, %.2f, %.2f, %.2f, %.2f, %.4f, %.2f, %.2f, %s,
%s, %c%c",
_TimeToString("%Y-%m-%d %H:%M:%S"),
@PLC.JUDGEMENT,
        @PLC.CAR_NAME,
        @PLC.CAR_NO,
        @PLC.BARCODE_F,
        @PLC.BARCODE_R,
        (double)@PLC.ZR2538, (double)@PLC.ZR2532, (double)@PLC.ZR2580,
        (double)@PLC.ZR2582, (double)@PLC.ZR2584, (double)@PLC.ZR2586,
        (double)@PLC.ZR2588, (double)@PLC.ZR2590, (double)@PLC.ZR2592,
(double)@PLC.ZR2594,
        (double)@PLC.ZR2534,(double)@PLC.ZR2536, @PLC.MARKING, @PLC.MARKING2, 0x0d, 0x0a);

        _Trace(strData);

        ret = _LogToFileEx(strFile, "%s", strData);
        if(ret != 1) {
                //
                // 정상이면 1, 아니면 -1이 돌아온다.
                @COMMON.NAS_WRITE = 0;
                strMsg = _FormatString("_LogToFileEx()..ret=%d, err_stat=%s", ret, _GetSystemErrorMsg());
                ShowLog(strMsg);
        }
        else {
                // 정상이면 1, 아니면 -1이 돌아온다.
                @COMMON.NAS_WRITE = 1;

                strMsg = _FormatString("%s, %s", strFile, strData);
                ShowLog(strMsg);
        }

}

void ShowLog(string msg)
```

```
{
	int nIndex;
	string strObject, str;

	strObject = "LOGLIST@메인페이지";

	// 리스트박스의 현재 리스트 개수가 50개가 넘으면 제일 위에 있는 리스트 하나 삭제.
	nIndex = _ListGetCount(strObject);
	if(nIndex > 50)
		_ListDelValue(strObject, 0);

	str = _FormatString("%s > %s", _TimeToString("%Y-%m-%d %H:%M:%S"), msg);
// 로그 텍스트 리스트박스에 추가 표시.
	_ListAddValue(strObject, str);

	nIndex = _ListGetCount(strObject);
	_ListSetSel(strObject, nIndex-1);
}
```

2.1.30 _OpenPage()

함수원형

void _OpenPage(string strPage, int nFlag)

파라메터	strPage : OPEN하고자 하는 뷰 페이지 이름을 문자열 상수 또는 변수를 지정합니다. nFlag : 열기모드 0 : 현재 페이지를 닫고 새로운 페이지로 전환합니다. 1 : 현재 페이지의 하부 페이지로 뷰 페이지를 OPEN 합니다
리턴형	없음
설명	지정한 뷰 페이지 파일을 OPEN합니다.
함수활용	오브젝트의 입력속성에서 시스템 처리의 페이지 열기와 동일한 기능이지만 내부함수를 사용해서 페이지를 열어야 하는 경우는 페이지가 열릴 때 다른 처리(데이터베이스 처리, 디바이스 처리 등)가 있는 경우 또는 특정 이벤트 발생시(경보 등)에 사용합니다

```
/*********************************************************
예제 : 뷰페이지 열기
*********************************************************/
void OpenPage()
{
	string strPage;

	// 페이지 전환.
	_OpenPage("ViewPage2", 0);

	// "데이터저장" 뷰페이지를 현재 오브젝트가 있는 뷰페이지의 서브페이지로 오픈합니다.
	_OpenPage("ViewPage3", 1);
	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun Studio에서 테스트를 위한 뷰페이지 "ViewPage2", "ViewPage3" 을 생성합니다.
	2. eRun을 실행한다 [F5]
	3. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	*************************************************************************************************/
}
```

2.1.31 _OpenPageOnPos()

함수원형

void _OpenPageOnPos(string strPage, int X, int Y)

파라메터	strPage : OPEN하고자 하는 뷰 페이지의 문자열 상수 또는 변수를 지정합니다. X : 화면의 픽셀단위 X좌표. Y : 화면의 픽셀단위 X좌표.
리턴형	없음
설명	지정한 뷰 페이지 파일을 X, Y좌표에 하부 윈도우로 OPEN 합니다.
함수활용	화면의 특정위치에 서브페이지로 뷰페이지를 열 때 사용합니다.

```
/*********************************************************
예제 : 뷰 페이지 특정 위치에 열기
*********************************************************/
void OpenPage()
{
	string strPage;

	// "데이터저장" 뷰페이지를 모니터 시작좌표 (100, 100)에 띄웁니다.
	_OpenPageOnPos("데이터저장", 100, 100);
}
```

2.1.32 _Playback()

함수원형

void _Playback(double fDuration)

파라메터	fDuration : 현재시간부터 몇분전 의미의 값
리턴형	없음
설명	eRun UI 리플레이 합니다.
함수활용	eRun 운용상태를 되돌려서 보기로 활용합니다.

```
/*********************************************************
예제 : 리플레이
*********************************************************/
void Playback()
{
	// 1분전 운용상태부터 리플레이 합니다.
	_Playback(1.0);
}
```

2.1.33 _PlayWaveFile()

함수원형

void _PlayWaveFile(string strFile, int nRepeat)

파라메터	strFile : 경로명을 포함한 WAV 파일이름 nRepeat : 반복횟수(정수값) 0 : play 중단 1 : 1회만 play 2 : 무한 반복 play
리턴형	없음
설명	윈도우 기본 음원파일인 확장자가 WAV 파일을 반복횟수 만큼 실행합니다.
함수활용	경보발생 및 특정 이벤트 발생시 사운드 발생시키고 알림이 필요할 때 사용합니다.

```
/*********************************************************
예제 : Wav 파일 실행하기
*********************************************************/
void PlayWaveFile()
{
	// sound.wav 파일을 1회만 play 합니다.
	_PlayWaveFile("C:₩₩sound.wav", 1);

	// sound.wav 파일을 무한정 반복해서 play 합니다.
	_PlayWaveFile("C:₩₩sound.wav", 2);

	// sound.wav 파일의 play를 중단 합니다.
	_PlayWaveFile("C:₩₩sound.wav", 0);
}
```

2.1.34 _PrintScreen()

함수원형

void _PrintScreen()

파라메터	없음
리턴형	없음
설명	현재 모니터 화면을 프린터로 출력합니다.
함수활용	경보발생 및 특정 이벤트 발생시 모니터화면을 출력합니다.

```
/*********************************************************
예제 : 화면 프린터 출력
*********************************************************/
void PrintScreen()
{
	_PrintScreen();
	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	*************************************************************************************************/
}
```

2.1.35 _PrintScreenEx()

함수원형

void _PrintScreenEx(int nMonitor, string strFileName, int nFormat)

파라메터	nMonitor : 모니터번호 0 : 기본화면 10 : 전체화면(듀얼모니터) 1~2 : 모니터번호 strFileName : 저장경로 포함한 파일이름, 확장자는 제외 nFormat : 0 : BMP 포맷 형식으로 저장 1 : PNG 포맷 형식으로 저장 2 : JPEG 포맷
리턴형	정수값 0 : OK -1 : FAIL
설명	현재 모니터 화면을 이미지파일로 지정파일에 저장합니다. 저장은 쓰레드 프로세스로 백그라운드 처리됩니다.
함수활용	경보발생 및 특정 이벤트 발생시 모니터화면을 이미지 파일형식으로 저장합니다.

※ 멀티스크린 환경인경우 스크린번호를 지정합니다.
※ 파일이름은 확장자를 빼고 경로명 포함해서 지정합니다.

```
/*********************************************************
예제 : 화면저장
*********************************************************/
void PrintScreenEx()
{
	string strFile;
	double dTime;

	// 메인페이지를 BMP형식으로 저장
	_PrintScreenEx(0, "c:₩TempData₩test", 0);

	// 메인페이지를 PNG형식으로 자장
	_PrintScreenEx(0, "c:₩TempData₩test", 1);

	// 다중모니터 전체페이지를 JPG형식으로 자장
```

```
	_PrintScreenEx(10, "c:₩TempData₩test", 2);

	dTime = _GetTick(); // 현재 시스템 시간을 double형으로 돌려준다.
	// 파일이름을 시스템시간을 붙여서 만든다. 확장자는 붙이지 않는다.
	strFile = _FormatString("c:₩temp₩sample%.3f", dTime);

	// 전체모니터화면을 BMP파일 형식으로 저장.
	_TraceEx("strFile =%s, return =%d", strFile, _PrintScreenEx(10, strFile, 0));

	/*****************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	*****************************************************************************************/
}
```

2.1.36 _RefreshControl()

함수원형

void _RefreshControl(string strObject)

파라메터	strObject : 뷰페이지에 있는 윈도우 오브젝트 이름입니다.
리턴형	없음
설명	윈도우 오브젝트를 내용을 갱신합니다.
함수활용	스크립트에 의해서 윈도우 오브젝트의 설정 값을 변경하고 변경된 내용을 적용할 때 사용하는 함수입니다. 예를 들어서, 스크립트에서 리스트컨트롤의 값을 변경한 후에 변경된 값을 적용할 때 사용합니다.

※ 윈도우 오브젝트 중에 그리드, 리스트컨트롤만 해당합니다.

```
/*********************************************************
예제 : 리스트 컨트롤 갱신하기
*********************************************************/
void Refreshcontrol()
{
	// 리스트컨트롤 0행 0열에 "SEFA Technology"로 설정하기
	_SFListSetValue("리스트컨트롤@뷰페이지명", 0, 0, "SEFA Technology");
	_RefreshControl("리스트컨트롤@뷰페이지명");
}
```

2.1.37 _RGB()

함수원형

int _RGB(int R, int G, int B)

파라메터	R : 적색의 색상값을 설정합니다.(0~255) G : 녹색의 색상값을 설정합니다.(0~255) B : 청색의 색상값을 설정합니다.(0~255)
리턴형	RGB조합된 4바이트 정수형
설명	RED, GREEN, BLUE의 값을 넘겨주면 색상값을 계산해서 돌려줍니다.
함수활용	그리드의 배경 및 글씨색, 라인그래프 오브젝트의 펜색상, 리스트컨트롤의 배경 및 글씨색 조합할 때 사용합니다.

```
/*********************************************************
예제 : 색상값 확인
*********************************************************/
void RGB()
{
    int nRed, nGreen, nBlue, nResult;

    nRed = 100;
    nGreen = 0;
    nBlue = 255;

    // 적색값:100, 녹색값:0, 파란색값:255을 조합한 색상값을 정수형 변수 nResult에 저장합니다.
    nResult = _RGB(nRed, nGreen, nBlue);

    // 정수형 변수 nResult의 값을 메시지박스에 표시합니다.
    _MsgBox(nResult, "결과", 0, 2);

        /*************************************************************************************************
        결과를 확인하기 위한 방법은 아래와 같습니다.
        1. eRun을 실행한다 [F5]
        2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
        *************************************************************************************************/
}
```

2.1.38 _SelectFolder()

함수원형

string _SelectFolder(string strFolder)

파라메터	strFolder : 폴더선택창에 초기값으로 기본 폴더 명을 상수 또는 변수를 지정합니다
리턴형	선택한 폴더 문자열
설명	폴더 선택창에서 선택한 폴더명을 문자열로 돌려줍니다.
함수활용	데이터 파일 및 로그파일 등을 저장시 저장위치를 설정하는 경우 사용합니다.

```
/*********************************************************
예제 : 폴더선택
*********************************************************/
void SelectFolder()
{
	string strFolder;

	// C 드라이브의 Program Files을 기본 폴더로 해서 폴더 다이얼로그를 오픈합니다.
	// 폴더 다이얼로그에서 선택한 폴더명을 문자열 변수 strFolder에 저장합니다.
	strFolder = _SelectFolder("C:₩Program Files");

	// 문자열 변수 strFolder의 값을 메시지박스에 표시합니다.
	_MsgBox(strFolder, "", 0, 1);
	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	*************************************************************************************************/
}
```

2.1.39 _ShowControl()

함수원형

void _ShowControl(string strObject, int nFlag)

파라메터	strObject : 뷰페이지에 있는 윈도우 오브젝트 이름입니다. nFlag : 지정한 오브젝트를 숨기거나 표시합니다. 0 : 오브젝트를 화면에 숨깁니다.→ 숨겨진 상태에서 계속 동작은 합니다. 1 : 오브젝트를 화면에 표시합니다.
리턴형	없음
설명	윈도우 오브젝트를 설정된 플래그에 따라 화면에 표시 또는 숨기기를 합니다.
함수활용	특정 조건에 따라 입력 제한 및 표시 제한을 해야 하는 경우 사용합니다.

※ 그리드, 현재경보, 라인그래프, 리스트박스, 리스트컨트롤, 콤보박스, 버튼, 체크박스, 라디오버튼, 입력창, 날짜 컨트롤에 대해서만 사용할 수 있습니다.

```
/*********************************************************
예제 : 윈도우 오브젝트 표시 제어하기
*********************************************************/
void ShowControl()
{
	// 오브젝트 명이 "그리드"인 윈도우 오브젝트를 화면에서 숨깁니다.
	_ShowControl("그리드@뷰페이지명", 0);
}
```

2.1.40 _Shutdown()

함수원형

void _Shutdown()

파라메터	없음
리턴형	없음
설명	프로그램을 종료합니다.
함수활용	

```
/*********************************************************
예제 : 프로그램 종료하기
*********************************************************/
void Shutdown()
{
// 프로그램 실행을 중단하고 종료합니다. _ClosePage("") 와 동일한 동작을 합니다.
_Shutdown();

/*************************************************************************************************
        결과를 확인하기 위한 방법은 아래와 같습니다.
        1. eRun을 실행한다 [F5]
        2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
        *************************************************************************************************/
}
```

2.1.41 _Sleep()

함수원형

void _Sleep(int nMillisecond)

파라메터	nMillisecond : 대기시간을 지정합니다. 단위는 ms(1/1000초)입니다. 1초 = 999
리턴형	없음
설명	대기시간만큼 프로그램을 대기시킵니다.
함수활용	설정된 시간만큼 대기 및 유지를 해야 하는 경우 사용합니다. 태그에 값을 쓰고 또는 PLC에 값을 쓰고 잠시 기다려야 하는 경우 사용합니다.

```
/*********************************************************
예제 : 잠시대기
*********************************************************/
void ShowElapseTime()
{
	int nSleepSecond;
	string strTime;

	nSleepSecond = (_Rand() % 10) * 1000;

	_MsgBox(nSleepSecond, "지연동작을 수행합니다.", 0, 0);

	_Sleep(nSleepSecond);	// 0~10초 사이의 난수를 이용해 지연한다.

	_MsgBox(nSleepSecond, "지연동작이 완료되었습니다.", 0, 0);

	// 지정된 sec값을 원하는 string형태로 돌려준다.
	strTime = _ShowElapseTime(nSleepSecond, 0);
	_Trace(strTime);	// "00:01:00"		, H:M:S

	strTime = _ShowElapseTime(nSleepSecond, 1);
	_Trace(strTime);	// "00:01"		, H:M

	strTime = _ShowElapseTime(nSleepSecond, 2);
	_Trace(strTime);	// "00:00:01:00"	, D:H:M:S

	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
```

```
        1. eRun을 실행한다 [F5]
        2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
        3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
        ********************************************************************************************/
}
```

2.1.42 _Trace()

함수원형

void _Trace(int nValue)

void _Trace(string strValue)

파라메터	nValue : 정수형 변수 strValue : 문자열 *정수, 실수, 문자열 모두 사용이 가능합니다
리턴형	없음
설명	파라메터로 입력된 변수의 값을 TraceView 창에 표시합니다. 문자열, 정수, 실수값을 사용합니다.
함수활용	스크립트 실행중 변수, 태그 값등의 변화상태를 디버깅하는데 사용합니다.

```
/*********************************************************
예제 : 트레이스
*********************************************************/
void Trace()
{
	int n;

	// 0 ~ 19999 사이의 임의 수를 정수형 변수 n에 저장합니다.
	n = _Rand() % 20000;

	// 정수형 변수 n의 값을 트레이스 창에 표시합니다.
	_Trace(n);

	// 문자열 "Test Trace"를 트레이스 창에 표시합니다.
	_Trace("Test Trace");

	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
	*************************************************************************************************/
}
```

2.1.43 _TraceEx()

함수원형

void _TraceEx(string strFormat, var1, var2, ...)

파라메터	strFormat : 표현 형식을 지정 (%d, %s, %f, ...) -> _FormatString() 함수와 동일하니 참조하세요. var1 : 표현형식 첫번째 표현식에 해당하는 값 var2 : 2번째 표현식에 해당하는 값. ... : n개의 표현식을 의미하는데, 이것은 표현하고자 하는 값이 가변적이라는 의미.
리턴형	없음
설명	스크립트 실행시 특정값의 변화나 실행결과, 데이터베이스 결과값을 표시함으로써 진행상태 값을 확인할 수 있도록 하는 함수입니다. Runtime 실행시에 Trace View 창을 통해서 항시 확인이 가능하다.
함수활용	스크립트 실행중 변수, 태그 값등의 변화상태를 디버깅하는데 사용합니다.

```
/*********************************************************
예제 : 트레이스
*********************************************************/
void TraceEx()
{
	int n;

	// 0 ~ 19999 사이의 임의 수를 정수형 변수 n에 저장합니다.
	n = _Rand() % 20000;

	// 정수형 변수 n의 값을 트레이스 창에 표시합니다.
	_TraceEx("n = %d", n);

	// 문자열 "Test Trace"를 트레이스 창에 표시합니다.
	_TraceEx("string =%s", "Test Trace");

	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
	*************************************************************************************************/
}
```

```
/*********************************************************
예제2 : 끝문자 CR 떼어내기 시험.
*********************************************************/
void Delete_CR()
{
	int len;
	string sPressure_IG, str;

	sPressure_IG = _FormatString("%s%c", "4.56E-6", 0x0d);

	len = _StrLen(sPressure_IG);			// 0x0d(CR)을 포함하는 문자열의 개수를 얻어온다.
	str = _StrRight(sPressure_IG, 1);	//오른쪽 1개의 문자를 얻어온다.

	if( _AsciiCode(str, 0) == 0x0d)			//마지막이 CR로 되어 있는지 확인한다.
		sPressure_IG = _StrLeft(sPressure_IG, len-1); //마지막 CR을 제외한 문자열을 sPressure_IG로 복사한다.

	// 위의 스크립트는 _StrTrim()을 이용해도 동일한 결과를 얻을 수  있다.

	_TraceEx("sPressure_IG = %s, len=%d, str=%s, sLen=%d", sPressure_IG, _StrLen(sPressure_IG), str, _StrLen(str));
}
```

2.1.44 _Unlock()

함수원형

void _Unlock(int nCode)

파라메터	nCode : 잠금 임의코드값 (1~79)
리턴형	없음
설명	하나의 자원(파일읽기/쓰기 등)에 동시접근을 막기 위해 자원접근 시 하나의 프로세스만 접근 할 수 있게 설정한 Lock을 해제합니다.
함수활용	한개 이상의 실행함수 및 프로세스에서 동일한 파일에 접근하여 파일쓰기를 하는 경우 서로 다른 실행함수 및 프로세스에서 동시에 접근하는 것을 막고 사용한 후 다른 실행함수 및 프로세스에서 접근 가능하게 하기 위해 사용합니다.

※ 주의 사항

Lock 내부함수는 항상 UnLock과 사용해야 합니다.

Lock이 되면 UnLock이 되기 전까지는 다른 프로세스에서 접근을 할 수 없습니다.

또한 Lock함수의 매개변수에 쓰이는 고유번호가 UnLock에서 동일하게 사용되어야 합니다.

동시 접근에 위험성이 있는 자원들에 대해서만 사용하시기 바랍니다.

위의 예제에서 하나의 파일에 데이터를 쓰는 경우 또는 데이터베이스에 동시접근성이 있는 경우, 윈도우 오브젝트에 접근하는 경우 등 사용하시기 바랍니다.

동일한 고유번호로 서로 다른 프로세스에 Lock을 설정하게 되면 하나의 프로세스가 UnLock이 되기 전까지는 다른 프로세스에서 접근을 하지 않고 대기를 하고 있습니다.

```
/*********************************************************
예제 : 두 개의 함수에서 하나의 파일에 데이터 쓰기
    두 개의 함수는 Function Script 오브젝트에서 각각 호출
    파일은 최초에 열렸고 FD 번호는 태그에 있다고 가정.
*********************************************************/
void FuncA()
{
        int nFD;
        string strValue;
        strValue = "SEFA";
        // 파일 오픈시 생성된 파일번호로 태그명 @DEMO.FileFD에서 갖고 와서 정수형 변수 nFD에 저장합니다.
        nFD = @DEMO.FileFD;
        // 10번이란 키로 Lock을 설정하여 다른 프로세스에서 파일을 제어할 수 없게 합니다..
        _Lock(10);
        // 정수형 변수 nFD에 저장된 파일번호의 파일에 문자열 변수 strValue에 저장된 값(SEFA)를 파일에 씁
```

```
니다.
        _FileWrite(nFD, strValue, 4);
        // 10번이란 키의 Lock을 해제하여 다른 프로세스에서 파일을 제어할 수 있게 합니다..
        _Unlock(10);
}

void FuncB()
{
        int nFD;
        string strValue;
        strValue = "Technology";
        // 파일 오픈시 생성된 파일번호로 태그명 @DEMO.FileFD에서 갖고 와서 정수형 변수 nFD에 저장합니
다.
        nFD = @DEMO.FileFD;
        // 10번이란 키로 Lock을 설정하여 다른 프로세스에서 파일을 제어할 수 없게 합니다..
        _Lock(10);
        // 정수형 변수 nFD에 저장된 파일번호의 파일에 문자열 변수 strValue에 저장된 값(Technology)를 파
일에 씁니다.
        _FileWrite(nFD, strValue, 10);
// 10번이란 키의 Lock을 해제하여 다른 프로세스에서 파일을 제어할 수 있게 합니다..
        _Unlock(10);
}
```

2.1.45 _KillProcess()

함수원형

void _KillProcess(string strProcessName)

파라메터	strProcessName : 강제종료 하려고 하는 실행파일이름
리턴형	정수 0 : 성공 -1 : 실패
설명	실행중인 프로세스를 모두 강제종료 합니다.
함수활용	외부 프로그램을 스크립트 통해서 실행시켰을 경우에 강제로 종료하려 할 때 사용하면 편리합니다.

```
/*********************************************************
예제 스크립트
*********************************************************/
void ProcessKill()
{
	int nRet;

	// LogQuery 프로세스를 강제 종료합니다.
	nRet = _KillProcess("LogQuery.exe");
	_TraceEx("_KillProcess('LogQuery') return : %d", nRet);

	// eRunWatch 프로세스를 강제 종료합니다.
	nRet = _KillProcess("eRunWatch.exe");
	_TraceEx("_KillProcess('eRunWatch') return : %d", nRet);

	/*****************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. eRun이 실행되었으면 [F6] 트레이스창 을 활성화 시킨다.
	3. [F7] 함수 수동호출 을 활성화 시킨후 본 사용자 정의 함수를 실행시킨다.
	*****************************************************************************************/
}
실행결과
[2021-12-16 12:07:31.707]  _KillProcess('LogQuery') return : -1
[2021-12-16 12:07:31.720]  _KillProcess('eRunWatch') return : 0
```

2.1.46 _WindowMinimize()

함수원형

void _WindowMinimize()

파라메터	없음
리턴형	없음
설명	윈도우를 최소화 합니다.
함수활용	현재 뷰페이지를 크기를 최소화 시킵니다.

```
/*********************************************************
예제 스크립트
*********************************************************/
void Minimize()
{
	// 현재 뷰페이지를 최소화 모드로 전환합니다.
	_WindowMinimize();

	/*****************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. eRun이 실행되었으면 [F6] 트레이스창 을 활성화 시킨다.
	3. [F7] 함수 수동호출 을 활성화 시킨후 본 사용자 정의 함수를 실행시킨다.
	*****************************************************************************************************/
}
```

2.1.47 _ShowProgressBar()

함수원형

void _ShowProgressBar(string strTitle, int nMin, int nMax, string strTagname)

파라메터	strTitle : 진행바 제목 문자열 nMin : 최소값 (음수 또는 양수) nMax : 최대값 (nMin보다 커야함) strTagName : 위치값 태그이름 문자열
리턴형	없음
설명	진행바 윈도우를 화면에 표시합니다.
함수활용	시간이 소요되는 DB검색 또는 진행상태 확인이 필요할 경우 사용합니다.

다음과 같이 사용합니다.

```
_ShowProgressBar("진행상태", 0, 100, "RESULT.PROG_VALUE");
_ShowProgressBar("DB 검색중", -100, 100, "RESULT.PROG_VALUE");
```

지정된 태그의 값이 최대값과 같으면 진행창은 종료됩니다.

[주의] 태그이름은 문자열로 입력해주어야 합니다. 지정된 태그값으로 진행바의 위치를 결정합니다.

```
/*********************************************************
예제 스크립트
*********************************************************/
void ShowProgBar()
{
	// 타이틀을 "DB검색중..."으로 표시.
	// 진행창을 화면 중앙에 표시하고, 진행위치를 태그 "PROGRESS.PROG1" 값으로 표시한다.
	_ShowProgressBar("Progress testing...", 0, 10, "PROGRESS.PROG1");

	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. eRun이 실행되었으면 [F6] 트레이스창 을 활성화 시킨다.
	3. [F7] 함수 수동호출 을 활성화 시킨후 본 사용자 정의 함수를 실행시킨다.
	*************************************************************************************************/
}
```

2.1.48 _GetProjectInfo()

함수원형

string _GetProjectInfo(int nIndex)

파라메터	nIndex : 프로젝트 정보 항목번호 (0~7) 0 : 프로젝트 엔진버전 1 : 프로젝트 경로 2 : 프로젝트 이름 3 : 고객이름 4 : 프로젝트 설명 5 : 생성일자 6 : 최종 수정일자 7 : 퍼블리시 경로
리턴형	문자열
설명	프로젝트 정보를 문자열로 돌려줍니다.
함수활용	프로젝트 이름, 폴더경로, 생성시간등의 정보가 필요할 경우 사용합니다. 데이터를 파일을 프로젝트 경로에 생성 또는 저장할 경우 사용합니다.

- nIndex(0~7) 번호가 잘못 지정된 경우 "N/A" 문자열을 돌려줍니다.

```
/*********************************************************
예제 스크립트
*********************************************************/
void ShowProjectInfo()
{
	string strInfo;

	// 프로젝트 번호
	strInfo =_GetProjectInfo(0);
	_TraceEx("Project No = %s", strInfo);
	// 프로젝트 경로
	strInfo =_GetProjectInfo(1);
	_TraceEx("Project Path = %s", strInfo);
	// 프로젝트 이름
	strInfo =_GetProjectInfo(2);
	_TraceEx("Project Name = %s", strInfo);
	// 프로젝트 고객이름
	strInfo =_GetProjectInfo(3);
	_TraceEx("Project Customer = %s", strInfo);
```

```
	// 프로젝트 설명
	strInfo =_GetProjectInfo(4);
	_TraceEx("Project Comment = %s", strInfo);
	// 프로젝트 생성일자
	strInfo =_GetProjectInfo(5);
	_TraceEx("Project Created Date = %s", strInfo);
	// 프로젝트 최종수정 일자
	strInfo =_GetProjectInfo(6);
	_TraceEx("Project Last Update Data = %s", strInfo);
	// 프로젝트 퍼블리시 폴더
	strInfo =_GetProjectInfo(7);
	_TraceEx("Project Publish Folder = %s", strInfo);

	/*********************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. eRun이 실행되었으면 [F6] 트레이스창 을 활성화 시킨다.
	3. [F7] 함수 수동호출 을 활성화 시킨후 본 사용자 정의 함수를 실행시킨다.
	*********************************************************************************************/
}
```

[실행결과]

```
[2023-10-20 13:07:26.624]  Project No = 20230116
[2023-10-20 13:07:26.624]  Project Path = C:₩SEFA₩eRun(x64)₩Project₩YST
[2023-10-20 13:07:26.624]  Project Name = YST
[2023-10-20 13:07:26.624]  Project Customer = 와이에스테크놀로지
[2023-10-20 13:07:26.625]  Project Comment = S** Industry Corp.
[2023-10-20 13:07:26.625]  Project Created Date = 2022-12-06 16:49:40
[2023-10-20 13:07:26.625]  Project Last Update Data = 2023-10-20 23:00:33
[2023-10-20 13:07:26.625]  Project Publish Folder = C:₩SEFA₩eRun(x64)₩Publish₩YST₩
```

2.1.49 _UserGetInfo()

함수원형

string _UserGetInfo(int nIndex)

파라메터	nIndex : 사용자 정보 항목번호 (0~7) 0 : 사용자명 1 : 사용자 암호 2 : 사용자 유형("0" :관리자, "1" :일반 사용자 3 : 접근영역 4 : 사용자 설명
리턴형	문자열
설명	현재 로그인 된 사용자 정보를 문자열로 돌려줍니다.
함수활용	다중 사용자가 프로젝트를 공유하고 사용할 때 로그인 사용자에 대한 정보를 활용해서 운영 로그 기록저장이 필요할 경우 또는 지정된 사용자에 따라 기능제한용으로 사용합니다.

- nIndex(0~4) 번호가 잘못 지정된 경우 "N/A" 문자열을 돌려줍니다.

```
/*********************************************************
예제 스크립트
*********************************************************/
void ShowUserInfo()
{
	string strInfo;

	// 사용자 이름
	strInfo =_UserGetInfo(0);
	_TraceEx("User Name = %s", strInfo);

	// 사용자 암호
	strInfo =_UserGetInfo(1);
	_TraceEx("User Pass = %s", strInfo);

	// 사용자 유형("0" :관리자, "1" :일반 사용자
	strInfo =_UserGetInfo(2);
	_TraceEx("User type = %s", strInfo);
```

```
        // 접근영역 ("no restriction", "A", ... "Z")
        strInfo =_UserGetInfo(3);
        _TraceEx("User restriction = %s", strInfo);

        // 사용자 설명
        strInfo =_UserGetInfo(4);
        _TraceEx("User comment = %s", strInfo);

        /*************************************************************************************************
        결과를 확인하기 위한 방법은 아래와 같습니다.
        1. eRun을 실행한다 [F5]
        2. eRun이 실행되었으면 [F6] 트레이스창 을 활성화 시킨다.
        3. [F7] 함수 수동호출 을 활성화 시킨후 본 사용자 정의 함수를 실행시킨다.
        *************************************************************************************************/
}
```

[실행결과]

```
[2024-05-11 11:12:02.605]  User Name = James_Dean
[2024-05-11 11:12:02.605]  User Pass = 1324
[2024-05-11 11:12:02.606]  User type = 0
[2024-05-11 11:12:02.606]  User restriction = No restriction
[2024-05-11 11:12:02.606]  User comment = Equipment service assistant
```

2.2 산술 함수

2.2.1 _Abs()

함수원형

int _Abs(int nValue)

double _Abs(double fValue)

파라메터	nValue : 부호정수. fValue : 부호실수
리턴형	정수절대값 / 실수절대값
설명	실수/정수 값에 대한 절대값을 돌려줍니다.
함수활용	

```
/*********************************************************
예제 : 절대값 구하기
*********************************************************/
void Abs()
{
	_TraceEx("Abs(123.456)=%f, Abs(-123.456)=%f" , _Abs(123.456), _Abs(-123.456) );
	_TraceEx("Abs(123)=%d, Abs(-123)=%d" , _Abs(123), _Abs(-123) );
	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
	*************************************************************************************************/
}
```

2.2.2 _Asin()

함수원형

double _Asin(double fRadian)

파라메터	fRadian : 라디안 단위의 각도입니다.
리턴형	fRadian의 asin값
설명	삼각함수 ASin 함수의 값을 돌려줍니다.
함수활용	

```
/*********************************************************
예제 : asin값 구하기
*********************************************************/
void Trigonometric()
{
	_TraceEx("_Tan(0.5) = %f, _Cos(0.5) = %f, _Sin(0.5) = %f",
					_Tan(0.5), _Cos(0.5), _Sin(0.5) );
	_TraceEx("_Atan(0.5) = %f, _Atan2(0.5, 0.5) = %f, _Acos(0.5) = %f",
					_Atan(0.5), _Atan2(0.5, 0.5), _Acos(0.5) );
	_TraceEx("_Radian(0.5) = %f, _Degree(0.5) = %f",
					_Radian(0.5), _Degree(0.5) );
	/*****************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
	*****************************************************************************************/
}
```

2.2.3 _Acos()

함수원형

double _Acos(double fRadian)

파라메터	fRadian : 라디안 단위의 각도입니다.
리턴형	fRadian의 acos값
설명	삼각함수 acos값을 반환합니다.
함수활용	

```
/*********************************************************
예제 : acos값 구하기
*********************************************************/
void Trigonometric()
{
	_TraceEx("_Tan(0.5) = %f, _Cos(0.5) = %f, _Sin(0.5) = %f",
					_Tan(0.5), _Cos(0.5), _Sin(0.5) );
	_TraceEx("_Atan(0.5) = %f, _Atan2(0.5, 0.5) = %f, _Acos(0.5) = %f",
					_Atan(0.5), _Atan2(0.5, 0.5), _Acos(0.5) );
	_TraceEx("_Radian(0.5) = %f, _Degree(0.5) = %f",
					_Radian(0.5), _Degree(0.5) );
	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
	*************************************************************************************************/
}
```

2.2.4 _Atan()

함수원형

double _Atan(double fRadian)

파라메터	fRadian : 라디안 단위의 각도입니다.
리턴형	fRadian의 atan값
설명	삼각함수 atan값을 반환합니다.
함수활용	

※ 결과값은 라디안 단위로 나오기 때문에 각도로 변환하려면 PI(3.141592)를 곱해주면 됩니다.

```
/*********************************************************
예제 : atan값 구하기
*********************************************************/
void Trigonometric()
{
	_TraceEx("_Tan(0.5) = %f, _Cos(0.5) = %f, _Sin(0.5) = %f",
					_Tan(0.5), _Cos(0.5), _Sin(0.5) );
	_TraceEx("_Atan(0.5) = %f, _Atan2(0.5, 0.5) = %f, _Acos(0.5) = %f",
					_Atan(0.5), _Atan2(0.5, 0.5), _Acos(0.5) );
	_TraceEx("_Radian(0.5) = %f, _Degree(0.5) = %f",
					_Radian(0.5), _Degree(0.5) );
	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
	*************************************************************************************************/
}
```

2.2.5 _Atan2()

함수원형

double _Atan2(double fy, double fx)

파라메터	fy : 라디안 단위의 각도입니다. fx : 라디안 단위의 각도입니다.
리턴형	fRadian의 atan2값
설명	삼각함수 atan2의 값을 반환합니다.
함수활용	두점 사이의 절대각도값을 계산할 때 사용합니다.

※ 결과값은 라디안 단위로 나오기 때문에 각도로 변환하려면 PI(3.141592)를 곱해주면 됩니다.
※ atan2는 두 점 사이의 상대좌표(x, y)를 받아 절대각을 -π ~ π의 라디안 값으로 반환한다.

```
/*********************************************************
예제 : atan2값 구하기
*********************************************************/
void Trigonometric()
{
	_TraceEx("_Tan(0.5) = %f, _Cos(0.5) = %f, _Sin(0.5) = %f",
					_Tan(0.5), _Cos(0.5), _Sin(0.5) );
	_TraceEx("_Atan(0.5) = %f, _Atan2(0.5, 0.5) = %f, _Acos(0.5) = %f",
					_Atan(0.5), _Atan2(0.5, 0.5), _Acos(0.5) );
	_TraceEx("_Radian(0.5) = %f, _Degree(0.5) = %f",
					_Radian(0.5), _Degree(0.5) );
	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
	*************************************************************************************************/
}
```

2.2.6 _Cos()

함수원형

double _Cos(double fRadian)

파라메터	fRadian: 라디안 단위의 각도입니다.
리턴형	fRadian의 cos값
설명	삼각함수 cos값을 반환합니다.
함수활용	

```
/*********************************************************
예제 : cos값 구하기
*********************************************************/
void Trigonometric()
{
	_TraceEx("_Tan(0.5) = %f, _Cos(0.5) = %f, _Sin(0.5) = %f",
					_Tan(0.5), _Cos(0.5), _Sin(0.5) );
	_TraceEx("_Atan(0.5) = %f, _Atan2(0.5, 0.5) = %f, _Acos(0.5) = %f",
					_Atan(0.5), _Atan2(0.5, 0.5), _Acos(0.5) );
	_TraceEx("_Radian(0.5) = %f, _Degree(0.5) = %f",
					_Radian(0.5), _Degree(0.5) );
	/*****************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
	*****************************************************************************************/
}
```

2.2.7 _Degree()

함수원형

double _Degree(double fRadian)

파라메터	fRadian : 라디안 단위의 각도입니다.
리턴형	Radian을 Degree로 변환한 실수값
설명	호도법의 Radian값을 각도법의 Degree 값으로 환산할 때 사용합니다. 각도법은 원주를 360등분한 것의 하나를 1도로 정한 것
함수활용	

■ 파라메터

> Radian : 호도법

π radian = 180 degree

1 radian = 180 / π degree

x radian = x * 180 / π degree

> fDegree : 각도값 (90...)

180 degree = π radian

1 degree = π / 180 radian

x degree = x * π / 180 radian

```
/*********************************************************
예제 : _Degree
*********************************************************/
void Trigonometric()
{
	_TraceEx("_Tan(0.5) = %f, _Cos(0.5) = %f, _Sin(0.5) = %f",
					_Tan(0.5), _Cos(0.5), _Sin(0.5) );
	_TraceEx("_Atan(0.5) = %f, _Atan2(0.5, 0.5) = %f, _Acos(0.5) = %f",
					_Atan(0.5), _Atan2(0.5, 0.5), _Acos(0.5) );
	_TraceEx("_Radian(0.5) = %f, _Degree(0.5) = %f",
					_Radian(0.5), _Degree(0.5) );
	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
	*************************************************************************************************/
}
```

2.2.8 _Exp()

함수원형

double _Exp(double fValue)

파라메터	fValue : 지수값을 구하기 위한 실수값.
리턴형	e의 fValue 곱을 반환합니다.
설명	실수/정수 값에 대한 지수 값을 돌려줍니다.
함수활용	Log의 역함수로 사용합니다.

```
/*******************************************************
예제 : Exp값 구하기
*******************************************************/
void Exp()
{
    double value;
    // 실수 10.0의 지수승의 값을 구해 실수형 변수 value에 저장합니다.
    value =_Exp(10.0);

    _TraceEx("e의 %0.0f의 제곱 결과값 =%f", 10.0, value);

        /*************************************************************************************************
        결과를 확인하기 위한 방법은 아래와 같습니다.
        1. eRun을 실행한다 [F5]
        2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
        3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
        *************************************************************************************************/
}
```

2.2.9 _Log()

함수원형

double _Log(double fValue1, int nLog)

파라메터	fValue1 : 구하고자 하는 지수 값을 실수형태로 상수 또는 변수를 입력합니다. nLog : 구하고자 하는 로그의 지수 값을 정수형태로 입력합니다.
리턴형	실수값
설명	실수 값에 대한 로그 값을 돌려줍니다
함수활용	

```
/*******************************************************
예제 : Log값 구하기
*******************************************************/
void Log()
{
    string str;
    double value;

     // 실수 109.0의 로그10 값을 구해 실수형 변수 value에 저장합니다.
    value = _Log(109.0, 10);

     // 실수형 변수 value 값과 문자를 조합한 문자열을 문자열 변수 str에 저장합니다.
    str = _FormatString("결과값 =%f", value);

     // 문자열 변수 str의 값을 메시지박스에 표시합니다.
    _MsgBox(str, "Log(109) 계산", 0, 0);

        /*************************************************************************************************
        결과를 확인하기 위한 방법은 아래와 같습니다.
        1. eRun을 실행한다 [F5]
        2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
        *************************************************************************************************/
}
```

2.2.10 _Max()

함수원형

double _Max(double fValue1, double fValue2)

파라메터	fValue1 : 비교하고자 하는 데이터 값을 실수형태로 상수 또는 변수를 입력합니다 fValue2 : 비교하고자 하는 데이터 값을 실수형태로 상수 또는 변수를 입력합니다.
리턴형	실수값
설명	두 실수 값을 비교하여 큰 수를 돌려줍니다.
함수활용	

```
/*********************************************************
예제 :  Max값 구하기
*********************************************************/
void Max()
{
	double result;
	double a;
	double b;

	a =100.5;
	b =55.3;

	// 실수 a와 b의 값을 비교해서 큰값을 실수형 변수 result에 저장합니다.
	result = _Max(a, b);		// 결과 result = 100.5;

	// 실수형 변수 result의 값을 메시지박스에 표시합니다.
	_MsgBox(result, "결과", 0, 0);
	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	*************************************************************************************************/
}
```

2.2.11 _Min()

함수원형

double _Min(double fValue1, double fValue2)

파라메터	fValue1 : 비교하고자 하는 데이터 값을 실수형태로 상수 또는 변수를 입력합니다 fValue2 : 비교하고자 하는 데이터 값을 실수형태로 상수 또는 변수를 입력합니다.
리턴형	실수값
설명	두 실수 값을 비교하여 작은 수를 돌려줍니다.
함수활용	

```
/*********************************************************
예제 :  Min값 구하기
*********************************************************/
void Min()
{
	double result;
	double a;
	double b;

	a =100.5;
	b =55.3;

	// 실수 a와 b의 값을 비교해서 작은값을 실수형 변수 result에 저장합니다.
	result = _Min(a, b);		// 결과 result = 55.3;

	// 실수형 변수 result의 값을 메시지박스에 표시합니다.
	_MsgBox(result, "결과", 0, 0);
	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	*************************************************************************************************/
}
```

2.2.12 _Pow()

함수원형

double _Pow(double fValue1, double fValue2)

파라메터	fValue1 : 실수형의 데이터. fValue2 : 실수형의 지수값.
리턴형	실수값
설명	실수값A에 대한 실수값B의 지수승을 연산한 값을 돌려줍니다.
함수활용	

```
/*********************************************************
예제 :  pow값 구하기
*********************************************************/
void Pow()
{
    double result;
    double a;
    double b;

    a =10.0;
    b =5.0;

     // 실수 10의 실수 b승의 지수승 값을 실수형 변수 result에 저장합니다.
    result = _Pow(a, b); // 결과 result = 1000000.0;

     // 실수형 변수 result의 값을 메시지박스에 표시합니다.
    _MsgBox(result, "결과", 0, 0);
        /*************************************************************************************************
        결과를 확인하기 위한 방법은 아래와 같습니다.
        1. eRun을 실행한다 [F5]
        2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
        *************************************************************************************************/
}
```

2.2.13 _Radian()

함수원형

double _Radian(double fDegree)

파라메터	fDegree : 라디안 단위의 각도입니다.
리턴형	실수값
설명	각도법의 Degree를 호도법의 Radian으로 값을 환산할 때 사용합니다. 호도법은 호의 길이를 이용하여 각도를 재는 방법
함수활용	

■ 파라메터

> Radian : 호도법
  π radian = 180 degree
  1 radian = 180 / π degree
  x radian = x * 180 / π degree

> fDegree : 각도값 (90...)

  180 degree = π radian
  1 degree = π / 180 radian
  x degree = x * π / 180 radian

```
/*********************************************************
예제 : _Radian
*********************************************************/
void Trigonometric()
{
        _TraceEx("_Tan(0.5) = %f, _Cos(0.5) = %f, _Sin(0.5) = %f",
                                        _Tan(0.5), _Cos(0.5), _Sin(0.5) );
        _TraceEx("_Atan(0.5) = %f, _Atan2(0.5, 0.5) = %f, _Acos(0.5) = %f",
                                        _Atan(0.5), _Atan2(0.5, 0.5), _Acos(0.5) );
        _TraceEx("_Radian(0.5) = %f, _Degree(0.5) = %f",
                                        _Radian(0.5), _Degree(0.5) );
        /*************************************************************************************************
        결과를 확인하기 위한 방법은 아래와 같습니다.
        1. eRun을 실행한다 [F5]
```

```
        2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
        3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
        ********************************************************************************************/
}
```

2.2.14 _Rand()

함수원형

int _Rand()

파라메터	없음
리턴형	정수형
설명	난수 값을 돌려줍니다.
함수활용	

```
/*********************************************************
예제 : 난수값 구하기
*********************************************************/
void Rand()
{
	int n;
	n = _Rand();

	// 정수형 변수 n의 값을 메시지박스에 표시합니다.
	_MsgBox(n, "결과", 0, 0);

	// 0 ~ 999의 임의의 값을 정수형 변수 n 에 저장합니다.
	n = _Rand() % 1000;

	// 정수형 변수 n의 값을 메시지박스에 표시합니다.
	_MsgBox(n, "결과", 0, 0);

	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	*************************************************************************************************/
}
```

2.2.15 _Sin()

함수원형

double _Sin(double fRadian)

파라메터	fRadian : 라디안 단위의 각도입니다.
리턴형	fRadian의 sin값
설명	삼각함수 sin값을 반환합니다.
함수활용	

```
/*********************************************************
예제 : sin값 구하기
*********************************************************/
void Trigonometric()
{
	_TraceEx("_Tan(0.5) = %f, _Cos(0.5) = %f, _Sin(0.5) = %f",
					_Tan(0.5), _Cos(0.5), _Sin(0.5) );
	_TraceEx("_Atan(0.5) = %f, _Atan2(0.5, 0.5) = %f, _Acos(0.5) = %f",
					_Atan(0.5), _Atan2(0.5, 0.5), _Acos(0.5) );
	_TraceEx("_Radian(0.5) = %f, _Degree(0.5) = %f",
					_Radian(0.5), _Degree(0.5) );
	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
	*************************************************************************************************/
}
```

2.2.16 _Sqrt()

함수원형

double _Sqrt(double fValue)

파라메터	fValue : 실수값
리턴형	제곱근 실수값
설명	파라메터 실수의 제곱근 값을 계산합니다.
함수활용	

```
/*********************************************************
예제 : sqrt값 구하기
*********************************************************/
void Squrt()
{
	_TraceEx("4의 제곱근 = %f", _Sqrt(4.0) );
	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
	*************************************************************************************************/
}
```

2.2.17 _Tan()

함수원형

double _Tan(double dRadian)

파라메터	fRadian : 라디안 단위 입니다. ※ 360 Degree = 2*Pi
리턴형	dRadian의 tan값
설명	삼각함수 tan값을 반환합니다.
함수활용	

※ 결과값은 라디안 단위로 나오기 때문에 각도로 변환하려면 PI(3.141592)를 곱해주면 됩니다.

```
/*********************************************************
예제 : 삼각함수
*********************************************************/
void Trigonometric()
{
	_TraceEx("_Tan(0.5) = %f, _Cos(0.5) = %f, _Sin(0.5) = %f",
					_Tan(0.5), _Cos(0.5), _Sin(0.5) );
	_TraceEx("_Atan(0.5) = %f, _Atan2(0.5, 0.5) = %f, _Acos(0.5) = %f",
					_Atan(0.5), _Atan2(0.5, 0.5), _Acos(0.5) );
	_TraceEx("_Radian(0.5) = %f, _Degree(0.5) = %f",
					_Radian(0.5), _Degree(0.5) );
	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
	*************************************************************************************************/
}
```

2.2.18 _UVWGet()

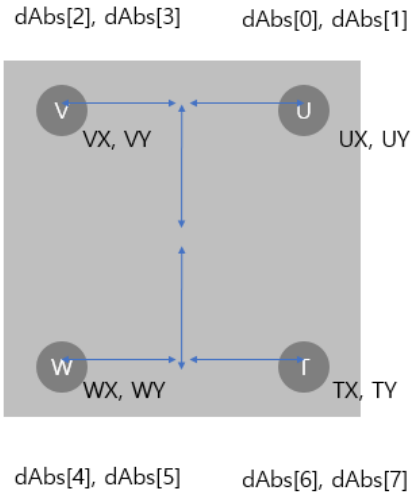
함수원형

void _UVWGet(double fCx, doube fCy, double fDegree, double [IN]fAbs[8], double [OUT]fDelta[8])

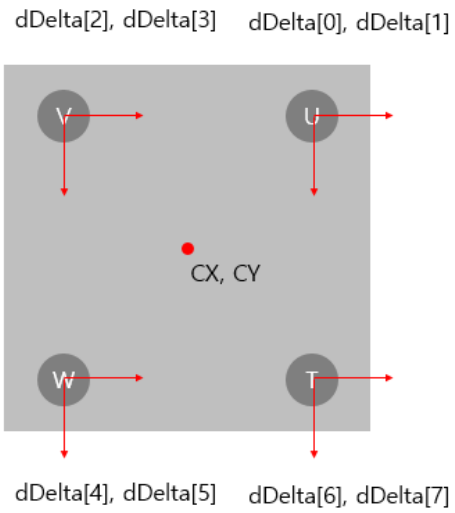
파라메터	double dCx, double dCy : 회전이동할 스테이지의 중심좌표 double dDegree : 회전이동할 스테이지의 각도 double fAbs[0]...[7] : 스테이지의 중심을 0,0 으로 지정하였을때 각 축의 X, Y좌표 double fDelta[0]...[7] : 스테이지가 dCx, dCy를 중심으로 dDegree만큼 회전하였을 경우 각 U, V, W의 이동량
리턴형	없음
설명	반도체 및 FPD 공정에서 사용하는 UVW스테이지의 이동량을 계산한다.
함수활용	아래 참고

사용 방법은 아래와 같다.

1. 각 UVW축과 스테이지의 기구 중심간의 거리를 입력한다.



2. 회전 이동 하기 위한 중심좌표 (CX, CY)/ 회전량(Degree) 입력하여 함수를 호출하면 각 UVW축의 이동량이 계산된다. dDelta[0], dDelta[1]...등으로 전달받은 값을 이동하여 모터를 이동하면 된다.




```
/*********************************************************
예제 : 스테이지 중심에서 회전이동하기
*********************************************************/
300x300 크기의 스테이지를 중심(0,0)을 기준으로 3도 회전이동했을때 각 UVW값을 계산한다.
void UVWGet()
{
	double dCx, dCy;
	double dDegree;
	double dAbs[8], dDelta[8];

	dCx = 0; dCy = 0;
	dDegree = 3;
	dAbs[0] = 150; dAbs[1] = -150;
	dAbs[2] = -150; dAbs[3] = -150;
	dAbs[4] = -150; dAbs[5] = 150;
	dAbs[6] = 150; dAbs[7] = 150;

	_UVWGet(dCx, dCy, dDegree, dAbs, dDelta);

	_TraceEx("스테이지중심 (%0.1f, %0.1f)을 (%0.1f)만큼회전하면 각 UVW는 U(%0.3f, %0.3f), V(%0.3f, %0.3f), W(%0.3f, %0.3f) 만큼 이동합니다.",
				dCx, dCy, dDegree,
				dDelta[0], dDelta[1], dDelta[2], dDelta[3], dDelta[4], dDelta[5], dDelta[6], dDelta[7]);

	/*********************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
```

```
        2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
        3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
        ********************************************************************************************/
}
```

2.3 문자열 함수

2.3.1 _ExtractSubString()

함수원형

int _ExtractSubString(string strData, string strChar, string sResult[n])

파라메터	strData : 구분자를 가지고 있는 소스문자열 strChar : 구분자 문자열 sResult[n] : 구분자로 분리된 문자열을 저장할 문자열 배열버퍼
리턴형	구분자로 분리된 문자열의 갯수
설명	소스문자열을 구분문자로 분리해서 결과문자배열에 넣어줍니다.
함수활용	

```
/*********************************************************
예제1 : 숫자를 문자열로 변환하기
*********************************************************/
void ExtractSubString()
{
	int nCount;
	string strData;		// 반드시 문자열로 지정.
	string sResult[10];		// 반드시 문자열 배열로 지정

	strData = "123*korea*King";
	nCount = _ExtractSubString(strData, "*", sResult);

	_TraceEx("count=%d, sResult[0]=%s, sResult[1]=%s, sResult[2]=%s", nCount,
	sResult[0], sResult[1], sResult[2]);

	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. eRun이 실행되었으면 [F6] 트레이스창 을 활성화 시킨다.
	3. [F7] 함수수동호출 을 활성화 시킨후 본 사용자 정의 함수를 실행시킨다.
	*************************************************************************************************/
}

/*********************************************************************************
```

예제2 : 지수형태로 입력된 문자열을 숫자형 Double로 변환하는 방법
**********/

```
double ExpToNum(string strExpNum)
{
	double dRet;
	double dRetTemp;
	int nFind_e;
	int nFind_E;

	double dNum;	// 숫자부	1.234e-8
	int nExp;		// 지수부
	string sTemp[2];

	double decimals;
	int integer;

	string sExpNumPrint[5];// 소수점 20자리까지 표시하기위한 변수

	int i;

	dRet = 0;
	nFind_e = _StrFind(strExpNum, "e", 0); //0번인덱스 (처음부터) 검색
	nFind_E = _StrFind(strExpNum, "E", 0);

	if ( nFind_e < 0 && nFind_E < 0 )
	{
		_TraceEx("ExpToNum Error Can not find e or E [%s]", strExpNum);
		return dRet;
	}

	if (nFind_e > -1)
	{
		_ExtractSubString(strExpNum, "e", sTemp);
	}

	if (nFind_E > -1)
	{
		_ExtractSubString(strExpNum, "E", sTemp);
	}
```

```
if ( nFinde_e < 0 && nFine_E < 0 )
{
	_Trace("지수형태의 문자열이 아닙니다.");
	return dRet;
}

dNum = _StrToNum(sTemp[0]);
nExp = _StrToNum(sTemp[1]);

if ( nExp < 0 ) // e-8 의 형태일때
{
	dRet = dNum / ( _Pow(10, _Abs(nExp)));
}else{ 			// e+8 의 형태일때
	dRet = dNum * ( _Pow(10, _Abs(nExp)));
}

_TraceEx("ExpToNum (%s) = %f (소수점6째자리까지만 출력됨)", strExpNum, dRet);

//지수 출력 방법
//일반적으로 스크립트에서 실제 데이터가 소수점 6자리 이하까지 있더라도
//화면에 출력하는 범위는 소수점 6째자리까지 만 가능합니다.
//그래서 아래의 방법으로 실제값을 확인하는 것 입니다.
//또한 실제 변환된값 외에 마지막 부분에 부동 소수점 오류로 인해 오차가 발생합니다.

if (dRet < 1)
{
	sExpNumPrint[0] = _FormatString("%.5f", dRet);

	decimals = dRet - _StrToNum(sExpNumPrint[0]);
	decimals *= 100000;
	i = 1;
	while ( i<5 )
	{
		decimals *= 100000;
		integer = decimals;
		sExpNumPrint[i] = _FormatString("%05d", integer);
		decimals -= integer;
		i++;
	}
		_TraceEx("ExpNumPrint [%s][%s][%s][%s][%s]",
```

```
                                sExpNumPrint[0], sExpNumPrint[1], sExpNumPrint[2],
                                sExpNumPrint[3],sExpNumPrint[4]);
        }
        return dRet;


        /*************************************************************************************************
        결과를 확인하기 위한 방법은 아래와 같습니다.
        1. eRun을 실행한다 [F5]
        2. eRun이 실행되었으면 [F6] 트레이스창 을 활성화 시킨다.
        3. [F7] 함수수동호출 을 활성화 시킨후  ExpToNum("1.234e-8")을 입력후 호출 한다.
        ** 결과값 ExpNumPrint [0.00000][00123][40000][00000][00005] 으로 출력되며 마지막 5는 부동소수점
오류임.
        *************************************************************************************************/
}
```

2.3.2 _FormatString()

함수원형

string _FormatString(string strFormat, data, ...)

파라메터	strFormat : 서식지정 형식의 문자열 data : 서식지정자에 해당하는 값 ... : n개의 서식지정자에 해당하는 값
리턴형	문자열
설명	인수 값을 변환형식에 맞추어서 변환된 문자열을 돌려줍니다.
함수활용	문자와 숫자 등을 조합하여 하나의 문자열로 만드는 경우 사용합니다.

여러 종류의 데이터(정수, 실수, 문자열)를 다양한 서식지정자에 맞춰 변환된 값을 문자열로 변환해 줍니다.
사용자 로그 문자열 생성, 태그 값, SQL 명령문을 표현하기 위해서 다양한 서식지정자를 통해서 문자열을 생성하고 활용합니다.
파라메터 개수는 서식지정자, 데이터, 이렇게 최소한 2개이상이어야 하며, 서식지정자(strFormat), data. 서식지정자에 %로 시작하는 형식이 있으면 두번째 파라메터부터 지정된 Data값이 이에 대응된다.
%로 시작하는 서식지정자에는 다음과 같이 다양한 형식이 있습니다.

■ 서식지정자

%c ASCII 문자로 표시
%s 문자열
%d 부호 있는 10진 정수
%i 부호 있는 10진 정수 (%d와 동일)
%f 고정 소수점으로 표현한 실수 (소수점 이하 6자리까지 표현)
%o 부호 없는 8진 정수
%u 부호 없는 10진 정수
%x 부호 없는 16진 정수 (소문자 사용)
%X 부호 없는 16진 정수 (대문자 사용)
%e 부동 소수점으로 표현한 실수 (지수형태)
%E 부동 소수점으로 표현한 실수 (지수형태)
%g 값에 따라 %f나 %e를 사용함.
%G 값에 따라 %f나 %E를 사용함.
%% 퍼센트(%) 기호 출력

■ 파라메터

> strFormat : 서식지정자("%d,%s,%f,...), %지정자에 맞게 데이터 개수가 반드시 일치해야 합니다.
"%[플래그(flag)][폭(width)][.정밀도][크기(length)]서식 문자(specifier)"

> data : 서식지정자에 해당하는 값
> ... : n개의 서식지정자에 해당하는 값

■ 유사함수
다음의 내부함수들이 _FormatString과 같이 동일한 형식으로 가변 파라메터 개수를 사용합니다.
_TraceEx(), _LogToFileEx(), _ComMethod(), _HSMSMethod(), _VisionMethod()

※ 정수 0을 포함해 특정 길이로 표현하는 경우 변환형태 %와 알파벳 사이에 0과 길이수를 씁니다.
예) 숫자 123을 문자열로 00123으로 표현하는 경우
_FormatString("%05d", 123);

※ 실수 0을 포함해 특정 길이로 표현하는 경우 변환형태 %와 알파벳 사이에 0과 길이수(소수점 포함 총 길이) 그리고 소수점 길이수로 씁니다.
예) 숫자 1.23을 문자열로 01.230으로 표현하는 경우
_FormatString("%07.3f", 1.23);

```
/*********************************************************
예제1 : 숫자를 문자열로 변환하기
*********************************************************/
void TestFormatString()
{
string strText;

// 서식지정자가 없는 경우, 두번째 문자열 텍스트를 줄바꿈 개행문자(₩n) 포함해서 strText에 저장한다.
strText = _FormatString("함수는 서식 지정자를 통해 출력할 데이터의 서식을 지정할 수 있어요!₩n");

// int형 데이터를 나타내기 위해서 '%d'라는 서식 지정자 사용하였고, 변환된 문자열을 TraceView 창에 표시한다.
_Trace(_FormatString("변수에 저장된 숫자는 %d입니다.", 10));

// value=12345.678을 TraceView 창에 표시한다.
strText = _FormatString("%s=%f₩n", "value", 12345.678);

/*************************************************************************************************
        결과를 확인하기 위한 방법은 아래와 같습니다.
        1. eRun을 실행한다 [F5]
        2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
        3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
```

```
	*****************************************************************************************************/
}

/*******************************************************
예제3 : 다양한 출력 방법
*******************************************************/
void TestFormatString2()
{
	string str;

	str = _FormatString("%%d를 사용한 결과 : %d", 123);			// 정수출력
	_Trace(str);

	str = _FormatString("%%05d를 사용한 결과 : %05d", 123);		// 정수 5자리로 표현 00123
	_Trace(str);

	str = _FormatString("%%0.3f를 사용한 결과 : %0.3f", 0.123456);	// 소수점 3자리까지 출력
	_Trace(str);

	str = _FormatString("%%f를 사용한 결과 : %f", 0.123456789);	// 소수점 최대 6자리까지 출력
	_Trace(str);

	str = _FormatString("%%o를 사용한 결과 : %o", 123);		// 8진수
	_Trace(str);

	str = _FormatString("%%x를 사용한 결과 : %x", 123);		// 16진수
	_Trace(str);

	str = _FormatString("%%g를 사용한 결과 : %g", 0.123456);	// 소수점 길이에 따라 지수형태로 변경하여 출력
	_Trace(str);

	str = _FormatString("%%g를 사용한 결과 : %g", 0.123456789);
	_Trace(str);

	str = _FormatString("%%G를 사용한 결과 : %G", 0.123456789);
	_Trace(str);

	str = _FormatString("%%c를 사용한 결과 : %c", 0x41);			// 'A' 문자출력
	_Trace(str);
```

```
        str = _FormatString("%%s를 사용한 결과 : %s", "JPeople.co.kr");          // 문자열출력
        _Trace(str);

        /*****************************************************************************************
        결과를 확인하기 위한 방법은 아래와 같습니다.
        1. eRun을 실행한다 [F5]
        2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
        3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
        *****************************************************************************************/
}
/*********************************************************
예제3 : 참고하세요
*********************************************************/
// 선택된 차종에 대해서 SQL DB에 2000개의 recipe data를 기록한다.
void SaveRecipe2DB()
{
        int ret;
        int nIndex, nGroup, cnt;
        string strSQL;
        string strTag;
        string strCarNo;

        int nValue;

        ret =_SQLOpen("RECIPE", "CARINFO");

        cnt =0;
        nIndex =0;
        nGroup =0;

        // 우선 이전 데이터를 DB 테이블에서 삭제한다.
        strSQL = _FormatString("delete * from recipe where car_no = '%s'", strCarNo);
        ret =_SQLRun("RECIPE", strSQL);

        //
        while(nGroup < 4) {
                nGroup ++;
                nIndex = 0;
                while(nIndex < 500) {
```

```
                    strTag = _FormatString("CARINFO%d.%d", nGroup, nIndex);
                    nValue = _GetTagValue(strTag);

                    strSQL = _FormatString("Insert into recipe(car_name,car_no,value1,rec_time)
Values('%s', '%s', %d, '%s')", @PLC.CAR_NAME, @PLC.CAR_NO, nValue,
_TimeToString("%Y-%m-%d %H:%M:%S"));
                    ret =_SQLRun("RECIPE", strSQL);

                    //_TraceEx("n=%d, %s, %d", cnt, strTag, ret);

                    nIndex++;
                    cnt++;
                }
            }
            _TraceEx("Complete insert into db...count=%d", cnt);
            _SQLClose("RECIPE");
}
```

2.3.3 _NumToStr()

함수원형

string _NumToStr(int nValue, int nDecimal)

string _NumToStr(double fValue, int nHex)

파라메터	nValue : 정수 또는 실수 nDecimal : 변환 진수를 입력합니다. (10, 16, 8)
리턴형	문자열
설명	숫자데이터를 문자열 데이터로 변환합니다.
함수활용	내부함수 _FormatString()함수를 사용해서 문자형을 얻을수도 있지만, 수치만 문자열로 변환하는 경우 사용합니다.

```
/*********************************************************
예제 : 숫자를 문자열로 변환하기
*********************************************************/
void NumToStr()
{
	int nValue;
	string strValue;
	nValue = 120;
	fValue = 36.5;

	// 숫자 10진수 120을 문자열로 변환하여 문자열 변수 strValue에 저장합니다.
	strValue = _NumToStr(nValue, 10);
	// 문자열 변수 strValue의 값을 메시지박스에 표시합니다.
	_MsgBox(strValue, "10진수 120을 문자열로 출력", 0, 0); // 메시지박스의 형식 : _MsgBox(메시지표시할 문자열, 메시지창의이름 문자열, 버튼번호, 아이콘번호)

	// 숫자 10진수 120을 8진수 문자열로 변환하여 문자열 변수 strValue에 저장합니다.
	strValue = _NumToStr(nValue, 8);
	// 문자열 변수 strValue의 값을 메시지박스에 표시합니다.
	_MsgBox(strValue, "10진수 120을 8진수 문자열로 출력", 0, 0);

	// 숫자 10진수 120을 16진수 문자열로 변환하여 문자열 변수 strValue에 저장합니다.
	strValue = _NumToStr(nValue, 16);
	// 문자열 변수 strValue의 값을 메시지박스에 표시합니다.
	_MsgBox(strValue, "10진수 120을 16진수 문자열로 출력", 0, 0);
```

```
	// FLOAT형 값 36.5를 16진수 형태변환하고 문자열변수 strValue에 저장합니다.
	strValue = _NumToStr(fValue, 16);
	_MsgBox(strValue, "10진수 36.5을 16진수 문자열로 출력", 0, 0);


	/*********************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	3. 메시지창의 내용을 확인합니다.
	*********************************************************************************************/
}

/*********************************************************
예제2 : 참고하세요
*********************************************************/
void Click_RF_POW_Set()
{
	if ( 1 == @M4000.M4031 )
return;	//Processing 상태면 입력 못함.

	@Util.SaveDataLabel = "RF Power Set";
	@Util.SaveDataValue = _NumToStr(@Main.nRF_POW_Set, 10) ; //포멧스트링으로 처리해야 할 수도 있음. 자리수 확인필요.
	@Util.SaveDataUnit = "Watt";
	@Util.SaveDataSource = "Main.nRF_POW_Set";

	_OpenPage("SaveData", 1);
}
```

2.3.4 _StrFind()

함수원형

int _StrFind(string strText, string strFind, int nPos)

파라메터	strText : 문자열 상수 또는 문자열변수 strFind : 찾고자 하는 문자열 nPos : 문자열의 검색 시작위치 (0 부터 검색시작)
리턴형	검색 문자가 나타난 위치값 (찾을수 없으면 -1)
설명	문자열에서 검색문자열을 시작위치에서부터 일치하는 문자열의 위치 값을 돌려줍니다. 일치하는 문자열이 없는 경우에는 -1을 돌려줍니다.
함수활용	숫자와 기호, 문자 등으로 구성된 문자열에서 특정 기호의 위치를 찾는 경우 사용합니다.

```
/*****************************************************************************
예제 : 문자열 찾기
*****************************************************************************/

void StringFind()
{
	string strString, strFind, strMsg;
	int nRet;

	strString = "abcdenfghijnklmnopqnrstuvwxyz";
	strFind = "n";
	// 지정된 문자열에서 처음부터 검색하여 일치 되는 위치의 값을 정수형 변수 nRet에 저장합니다.
	nRet = _StrFind(strString, strFind, 0);

	// 찾는 문자가 있으면
	if(nRet > -1)
	{
		// 찾는 문자열의 위치와 문자를 조합한 문자열을 strMsg에 표시합니다.
		 strMsg = _FormatString("찾는 문자열이 %d번째 위치에 있습니다.", nRet);
	}
	// 찾는 문자가 없으면
	else
		strMsg = "일치하는 문자열이 없습니다.";

	// 문자열 변수 strMsg의 값을 메시지박스에 표시합니다.
	_MsgBox(strMsg, "_StrFind 결과", 0, 4);
```

```
    /*********************************************************************************************
    결과를 확인하기 위한 방법은 아래와 같습니다.
    1. eRun을 실행한다 [F5]
    2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
    3. 메시지창의 내용을 확인합니다.
    *********************************************************************************************/
}
```

2.3.5 _StrFindRev()

함수원형

int _StrFindRev(string strText, int ch)

파라메터	strText : 일반문자열 ch : 찾고자 하는 문자 (아스키코드)
리턴형	검색 문자가 나타난 위치값 (찾을수 없으면 -1)
설명	문자열에서 검색문자를 문자열의 끝위치에서부터 지정 문자의 위치 값을 돌려줍니다. 일치하는 문자가 없는 경우에는 -1을 돌려줍니다.
함수활용	파일이름 또는 경로명등에서 파일 확장자 또는 폴더명을 찾을 경우 활용합니다.

```
/*********************************************************************************
예제 : 문자열 찾기
*********************************************************************************/

void String_Find_Reverse()
{
	string strString, strExt, strMsg;
	int nRet, nLen;

	strString = "Datafile.txt";

	// 지정된 파일명의 확장자명을 가져오기 분리문자 '.'의 위치를 뒤에서부터 찾아서 위치의 값을 정수형 변수 nRet에 저장합니다.
	// 0x2E는 아스키문자 '.' 에 대한 코드값.
	nRet = _StrFindRev(strString, 0x2E); // 결과는 8

	// 찾는 문자가 있으면
	if(nRet > -1)
	{
		// 찾는 문자열의 위치와 문자를 조합한 문자열을 strMsg에 표시합니다.
		strMsg = _FormatString("찾는 문자열이 %d번째 위치에 있습니다.", nRet);
		// 주어진 문자열의 길이
		nLen = _StrLen(strString);
		// 주어진 문자열의 길이에서 문자를 찾은 위치를 빼고 1을 더 빼준다.
		// 주어진 문자열의 오른쪽에서 확장자만 빼온다.
		strExt = _StrRight(strString, nLen - nRet -1);  // 12 - 8 = 4 - 1 = 3;
```

```
            _TraceEx("확장자는 %s 입니다. ", strExt);
        }
        // 찾는 문자가 없으면
        else
            strMsg = "일치하는 문자열이 없습니다.";

        // 문자열 변수 strMsg의 값을 메시지박스에 표시합니다.
        _MsgBox(strMsg, "_StrFindRev 결과", 0, 4);



        /*********************************************************************************************
        결과를 확인하기 위한 방법은 아래와 같습니다.
        1. eRun을 실행한다 [F5]
        2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
        3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
        *********************************************************************************************/
}
```

2.3.6 _StrLeft()

함수원형

string _StrLeft(string strText, int nLen)

파라메터	strText : 문자열 상수 또는 문자열변수 nLen : 읽어오려는 문자길이를 설정합니다.
리턴형	왼쪽 기준 길이만큼 읽은 문자열
설명	문자열의 제일 왼쪽에서 오른쪽방향으로 지정문자길이 수만큼 돌려줍니다.
함수활용	숫자와 기호, 문자 등으로 구성된 문자열에서 특정 문자열을 취합하는 경우 사용합니다.

```
/*********************************************************
예제 : 문자열에서 지정된 개수만큼 왼쪽부터 찾기
*********************************************************/
void StringLeft()
{
	string str;
	// 문자열 "ABCDEF"의 왼쪽 시작에서부터 3개의 문자열을 문자열 변수 str에 저장합니다.

	str = _StrLeft("ABCDEF", 3);

	// 문자열 변수 str의 값을 메시지박스에 표시합니다.
	_MsgBox(str, "_StrLeft 결과", 0, 0);
	//결과 str = ABC

	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	3. 메시지창의 내용을 확인합니다.
	*************************************************************************************************/
}
```

2.3.7 _StrLen()

함수원형

int _StrLen(string strText)

파라메터	strText : 문자열 상수 또는 문자열변수, 문자열태그
리턴형	정수
설명	파라메터 문자열의 길이(문자수)를 정수 단위로 돌려줍니다.
함수활용	문자열 길이를 이용해서 해당 문자열이 NULL("")인지 아닌지 체크할 수 있습니다.

```
/*********************************************************
예제1 : 문자열 길이 표시하기
*********************************************************/
void StrLen()
{
int nLen;
// 문자열 "Sample"의 길이를 정수형 변수 nLen에 저장합니다.
nLen = _StrLen("Sample");
// 정수형 변수 nLen의 값을 메시지박스에 표시합니다.
_MsgBox(nLen, "문자열 길이", 0, 0);
/*************************************************************************************************
        결과를 확인하기 위한 방법은 아래와 같습니다.
        1. eRun을 실행한다 [F5]
        2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
        3. 메시지창의 내용을 확인합니다.
        *************************************************************************************************/
}

/*********************************************************
예제2 : CRYO PUMP에서 통신으로 받은 상태정보를 분석하여
PLC의 D6000번지대에 데이터를 쓰는 예제입니다.
*********************************************************/
void SetALARM()
{
        string strTag, strLeft;

        // Regen Cycle
        if ( @PTCL130.REGEN_CYCLE_OLD != @PTCL130.REGEN_CYCLE )
        {
                strTag = @PTCL130.REGEN_CYCLE;
```

```
                strLeft  = _StrLeft( strTag , _StrLen(strTag)-1);
                @PTCL130.strLeft = strLeft;

                if ( "RND" == strLeft ) {    @D6000.D6066 = 1; }
                if ( "INI" == strLeft ) {    @D6000.D6066 = 2; }
                if ( "DLY" == strLeft ) {    @D6000.D6066 = 3; }
                if ( "WRM" == strLeft ) {    @D6000.D6066 = 4; }
                if ( "PRG" == strLeft ) {    @D6000.D6066 = 5; }
                if ( "EXP" == strLeft ) {    @D6000.D6066 = 6; }
                if ( "RPR" == strLeft ) {    @D6000.D6066 = 7; }
                if ( "RSD" == strLeft ) {    @D6000.D6066 = 8; }
                if ( "RGH" == strLeft ) {    @D6000.D6066 = 9; }
                if ( "ROR" == strLeft ) {    @D6000.D6066 = 10; }
                if ( "CSD" == strLeft ) {    @D6000.D6066 = 11; }

                if ( "CHL" == strLeft ) {    @D6000.D6066 = 12; }
                if ( "CMP" == strLeft ) {    @D6000.D6066   =   13;   @M6000_W.M6066   =   1;   }   else
{ @M6000_W.M6066 = 0; }
                if ( "ABT" == strLeft ) {    @D6000.D6066 = 14; }
        }
}
```

2.3.8 _StrMid()

함수원형

string _StrMid(string strText, int nPos, int nLen)

파라메터	strText : 문자열 상수 또는 문자열변수 nPos : 시작위치를 설정합니다.(0부터 시작합니다.) nLen : 읽어오려는 문자길이를 설정합니다.
리턴형	길이만큼 읽은 문자열
설명	문자열의 설정한 위치에서 오른쪽 방향으로 지정문자길이 수만큼 문자열을 돌려줍니다.
함수활용	숫자와 기호, 문자 등으로 구성된 문자열에서 특정 문자열을 취합하는 경우 사용합니다.

```
/*********************************************************
예제1 : 문자열에서 지정된 개수만큼 지정된 위치부터 찾기
*********************************************************/
void StringMid()
{
string str;
// 문자열 "ABCDEF"의 2번째 위치부터 3개의 문자열을 문자열 변수 str에 저장합니다.
str = _StrMid("ABCDEF", 2, 3);
// 문자열 변수 str의 값을 메시지박스에 표시합니다.
_MsgBox(str, "", 0, 0);
//결과 str = CDE
/*************************************************************************************************
        결과를 확인하기 위한 방법은 아래와 같습니다.
        1. eRun을 실행한다 [F5]
        2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
        3. 메시지창의 내용을 확인합니다.
        *************************************************************************************************/
}

/*********************************************************
예제2 : VAT 밸브의 상태를 읽어와서 PLC에 해당 상태를 쓰는 스크립트 입니다.
*********************************************************/
void GetAlarm_GV100()
{
        if ( 1 == @GV100.ERROR_STAT_COM )
        {
                if (  @GV100.ERROR_STAT_OLD != @GV100.ERROR_STAT )
                {
```

```
                        if ( "020" == @GV100.ERROR_STAT ) {     @M6000.M6930    =   1;   }   else
{ @M6000.M6930 = 0; }   //limit stop of valve unit not detected
                        if ( "021" == @GV100.ERROR_STAT ) {     @M6000.M6931    =   1;   }   else
{ @M6000.M6931 = 0; }   //rotation angle of valve plate limited during power up
                        if ( "022" == @GV100.ERROR_STAT ) {     @M6000.M6932    =   1;   }   else
{ @M6000.M6932 = 0; }   //rotation angle of valve plate limited during operation
                        if ( "040" == @GV100.ERROR_STAT ) {     @M6000.M6933    =   1;   }   else
{ @M6000.M6933 = 0; }   //motor drive faulure deteced
                        if ( "D 0" == @GV100.ERROR_STAT ) {     @M6000.M6934    =   1;   }   else
{ @M6000.M6934 = 0; }   //motor interlock is open
                        if ( "S R" == @GV100.ERROR_STAT ) {     @M6000.M6935    =   1;   }   else
{ @M6000.M6935 = 0; }   //service reqiest
                        if ( "M C" == @GV100.ERROR_STAT ) {     @M6000.M6936    =   1;   }   else
{ @M6000.M6936 = 0; }   //maintenace mode active
                }
                @GV100.ERROR_STAT_OLD = @GV100.ERROR_STAT;
        }

        if ( 1 == @GV100.WARNING_STAT_COM )
        {
                if ( @GV100.WARNING_STAT_OLD != @GV100.WARNING_STAT )
                {
                        if ( "1" == _StrMid(@GV100.WARNING_STAT,0, 1) ) { @M6000.M6920 = 1; } else
{ @M6000.M6920 = 0; }   //VAT 밸브 실 확인
                        if ( "1" == _StrMid(@GV100.WARNING_STAT,1, 1) ) { @M6000.M6921 = 1; } else
{ @M6000.M6921 = 0; }   //VAT 밸브 REVENE 데이터 세트 없음
                        if ( "1" == _StrMid(@GV100.WARNING_STAT,2, 1) ) { @M6000.M6922 = 1; } else
{ @M6000.M6922 = 0; }   //전원고장 배터리가 준비되지 않음
                        if ( "1" == _StrMid(@GV100.WARNING_STAT,3, 1) ) { @M6000.M6923 = 1; } else
{ @M6000.M6923 = 0; }   //압축 공기 공급상태가 양호하지 않음
                }
                @GV100.WARNING_STAT_OLD = @GV100.WARNING_STAT;
        }
}
```

2.3.9 _StrRight()

함수원형

string _StrRight(string strText, int nLen)

파라메터	strText : 문자열 상수 또는 문자열변수 nLen : 읽어오려는 문자길이를 설정합니다.
리턴형	문자열
설명	문자열의 가장 오른쪽에서 왼쪽방향으로 지정 문자길이 수만큼 문자열을 돌려줍니다.
함수활용	숫자와 기호, 문자 등으로 구성된 문자열에서 특정 문자열을 취합하는 경우 사용합니다.

```
/*********************************************************
예제1 : 문자열에서 지정된 개수만큼 오른쪽부터 찾기
*********************************************************/
void StringRight()
{
string str;
// 문자열 "ABCDEF"의 오른쪽 끝에서 부터 3개의 문자열을 문자열 변수 str에 저장합니다.
str = _StrRight("ABCDEF", 3);
// 문자열 변수 str의 값을 메시지박스에 표시합니다.
_MsgBox(str, "", 0, 0);
//결과 str = "DEF"
/*************************************************************************************************
        결과를 확인하기 위한 방법은 아래와 같습니다.
        1. eRun을 실행한다 [F5]
        2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
        3. 메시지창의 내용을 확인합니다.
        *************************************************************************************************/
}

/*********************************************************
예제2 : 참고하세요
*********************************************************/
void SET_START_ADDRESS()
{
        string strAddress, strRegister;
        string strHeader, strHeaderOffset;
        string str;
        int nStartAdd, nHeaderAddress, nHeaderOffset;
        int nAddress, n;
```

```
	int nCnt, nRowCnt, nSum;
	int nMax;

	@COMMON.CHECKINIT = 0;
	//_TraceEx("nHeaderAddress:%d, nHeaderOffset:%d, nAddress:%d", nHeaderAddress, nHeaderOffset,
nAddress);

	while(nRowCnt < 7)
	{
		nCnt = 0;
		strAddress = GET_START_ADDRESS(nRowCnt);

		strRegister = _StrLeft(strAddress, 1);
		nHeaderAddress = _StrToNum(_StrMid(strAddress, 1, 1));
		strHeaderOffset = _StrMid(strAddress, 2, 1);

		if(nRowCnt == 0 || nRowCnt == 6)
			nStartAdd = 240;
		else if(nRowCnt == 3)
			nStartAdd = 176;
		else if(nRowCnt == 5)
			nStartAdd = 128;
		else
			nStartAdd = _StrToNum(_StrRight(strAddress, 2));		//F0

		if(nRowCnt == 0)
			nMax = 1;
		else if(nRowCnt == 1)
			nMax = 16;
		else if(nRowCnt == 2)
			nMax = 16;
		else if(nRowCnt == 3)
			nMax = 32;
		else if(nRowCnt == 4)
			nMax = 6;
		else if(nRowCnt == 5)
			nMax = 32;
		else if(nRowCnt == 6)
			nMax = 5;
```

```
                while(nCnt < nMax)
                {
                        nAddress = nStartAdd+nSum;
                        str = _FormatString("%s%d%s%02X", strRegister, nHeaderAddress, strHeaderOffset,
nAddress);
                        //_TraceEx("nAddress:%d, str:%s", nAddress, str);
                        _GridSetValue("어드레스@뷰페이지명", nCnt, nRowCnt, str);

                        nCnt++;
                        nSum++;
                }
                nSum = 0;
                nRowCnt++;
        }
        _RefreshControl("어드레스@뷰페이지명");
}
```

2.3.10 _StrToNum()

함수원형

int _StrToNum(string strText)

double _StrToNum(string strText)

파라메터	strText : 정수 또는 실수형 문자열
리턴형	정수 또는 실수
설명	숫자 형 문자열을 정수 값 또는 실수 값으로 변환합니다.
함수활용	그리드, 리스트컨트롤, 입력창, 문자열태그 등에서 설정된 수치 문자열 데이터를 숫자형으로 변환할 때 사용합니다.

```
/*********************************************************
예제 : 숫자 형 문자열 숫자로 변환하기
*********************************************************/
void StrtoNum()
{
	int nNum;
	double fReal;

	// 문자열 "12345"를 정수형 변수 nNum에 저장합니다.
	nNum = _StrToNum("12345");
	// 정수형 변수 nNum의 값을 메시지박스에 표시합니다.
	_TraceEx("12345문자를 숫자로 출력 : %d", nNum);

	fReal = _StrToNum("345.67");
	_TraceEx("345.67문자를 숫자로 출력 : %f", fReal);
	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
	*************************************************************************************************/
}
```

2.3.11 _StrTrim()

함수원형

string _StrTrim(string strText)

파라메터	strText : 스페이스(' ') 문자가 포함된 문자열
리턴형	앞, 뒤부분 스페이스가 삭제된 문자열
설명	문자열의 왼쪽, 오른쪽의 Space를 모두 제거합니다. 단 문자열 중간에 포함된 스페이스 문자는 제거되지 않습니다.
함수활용	디바이스로부터 들어온 문자열 데이터나, 스크립트에서 문자열 처리를 할 경우 눈에 보이지 않는 스페이스 문자가 있을 경우 길이 연산등에 오류가 발생할 수 있습니다. 이러한 오류를 없애기 위해서 사용합니다.

```
/*****************************************************************************
기본예제
*****************************************************************************/

void Trim()
{
	string strText1, strText2;
	int len1, len2;

	strText1 = " ABCD1234 ";
	strText2 = _StrTrim(strText1);
	len1 = _StrLen(strText1);
	len2 = _StrLen(strText2);

	_TraceEx("#%s# (len1=%d), Trim:%s (len2=%d)",  strText1, len1, strText2, len2);

	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
	*************************************************************************************************/
}
```

2.3.12 _WithComma()

함수원형

string _WithComma(string strNumeric)

파라메터	strNumeric : 정수형 또는 실수형 문자열
리턴형	구분자 삽입된 문자열
설명	파라메터로 입력된 수치형 문자열을 천단위 구분자를 삽입해서 돌려줍니다.
함수활용	

```
/*****************************************************************************
기본예제 : 정수형 또는 실수형 수치데이터를 1000단위 문자(,)로 구분해서 돌려받는다.
*****************************************************************************/
void Comma()
{
	_TraceEx("result =%s", _WithComma("12345.67")); // 12,345.67,
	_TraceEx("result = %s", _WithComma(12345678)); // 12,345,678

	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
	*************************************************************************************************/
}
// 참고하세요
void FlowSumYearly()
{
	int nRet, nCount, cnt, nMonth, nYear;
	string strYear, strMonth, timeStart, timeEnd;
	string strObject, strResult, strKey, str;
	int nDayLong[12];
	double dValue;

	strObject = "YEARLY_LIST@뷰페이지명";
	strYear =  @SUMMARY.YEAR3;

	strKey = "FlowSumYearly_DB";
	nRet = _SFDBOpen(strKey, "RAWDATA");
	if(nRet < 1) {
```

```
		str = _FormatString("_SFDBOpen(FlowSumYearly_DB, RAWDATA) fail, nRet=%d", nRet);
		WriteLog(str);
		return;
}

str = _FormatString("FlowSumYealy() called, year=%s", strYear);
WriteLog(str);

nDayLong[0]=31;   //1월
nDayLong[1]=28;   //2월
nDayLong[2]=31;   //3월
nDayLong[3]=30;   //4
nDayLong[4]=31;   //5
nDayLong[5]=30;   //6
nDayLong[6]=31;   //7
nDayLong[7]=31;   //8
nDayLong[8]=30;   //9
nDayLong[9]=31;   //10
nDayLong[10]=30; //11
nDayLong[11]=31; //12

nYear = _StrToNum(strYear);

// 2월이면 윤달계산.
if((nYear%4)==0 || (nYear%100)==0 || (nYear%400)==0)
		nDayLong[1] =29;

cnt =0;
while(cnt < 12) {
		nMonth = cnt+1;

		strMonth = _FormatString("%s-%02d", strYear, nMonth);
		timeStart = _FormatString("%s-01 00:00:00", strMonth);
		timeEnd = _FormatString("%s-%d 23:59:59", strMonth, nDayLong[cnt]);

		_TraceEx("(%s - %s)", timeStart, timeEnd);
		_SFListSetValue(strObject, 0, cnt, strMonth);

		// 월시작 TOTAL FLOW.
		nCount = _SFDBGetValue(strKey, "SUMMARY.TOTAL_금일시작값",  timeStart, timeEnd,  16,
```

```
"SUMMARY.RESULT", "");
                    _SFListSetValue(strObject, 1, cnt, _WithComma(_FormatString("%.2f", @SUMMARY.RESULT)));
                    // 월 공급량 (월끝값 - 월시작값)
                    strResult = _FormatString("SUMMARY.MONTH_BAR%d", cnt);
                    nCount = _SFDBGetValue(strKey, "SUMMARY.TOTAL_금일시작값", timeStart, timeEnd, 12,
strResult, "");
                    dValue = _Max(0.0, _GetTagValue(strResult));

                    _SetTagValue(strResult, dValue);
                    _SFListSetValue(strObject, 2, cnt, _WithComma(_FormatString("%.2f", dValue)));

                    cnt++;
                    @REPORT.STEP = ((double)cnt / (double)12) * 100.0;
          }
          _RefreshControl(strObject@뷰페이지명);
          _SFDBClose(strKey);
}
```

2.4 파일 함수

2.4.1 _DeleteElapsedFile()

함수원형

int _DeleteElapsedFile(string strFolder, int nElapsedDay)

파라메터	strFolder : 폴더명을 문자열 상수 또는 변수를 지정합니다. nElapsedDay : 경과일수
리턴형	삭제된 파일의 수
설명	지정된 폴더에서 날자가 경과된 파일, 폴더를 삭제한다.
함수활용	

```
/*********************************************************
예제 : 90일 경과된 파일들 삭제하기
*********************************************************/
void CleanFolder()
{
	int ret, nElapsed;
	string strFolder;

	strFolder = "D:₩temp";
	nElapsed = 90;

	// D:₩temp 폴더에 있는 파일들 중에서 90일이 경과된 파일을 모두 삭제한다.
	ret = _DeleteElapsedFile(strFolder, nElapsed);
	_TraceEx("폴더 %s에서 %d개의 파일이 삭제되었습니다.", strFolder, ret);

	/*************************************************************************************************
	_TraceEx를 통해 결과를 확인하기 위한 방법은 아래와 같습니다.
	1. D:₩₩temp 폴더에 90일 이전에 생성한 임의파일과 이후의 임의의 파일을 준비한다.
	2. eRun을 실행한다 [F5]
	3. eRun이 실행되었으면 [F6] 트레이스창 을 활성화 시킨다.
	4. [F7] 함수수동호출 을 활성화 시킨후 본 사용자 정의 함수를 실행시킨다.
	5. 트레이스창의 출력문을 확인한다.
	6. 윈도우 탐색기를 열어 해당 파일이 삭제되었는지 확인한다.
	*************************************************************************************************/
}
```

2.4.2 _FileAccess()

함수원형

int _FileAccess(string strFile, int nFlag)

파라메터	strFile : 경로명이 포함된 파일명을 문자열 상수 또는 변수를 지정합니다. nFlag : 액세스 플래그 0 : 파일 존재여부. 1 : 쓰기가능 여부. 2 : 읽기가능 여부. 3 : 읽기쓰기 가능 여부.
리턴형	0 :성공 -1 : 실패
설명	지정된 파일명이 있는지 검사합니다.
함수활용	특정 파일을 오픈하기 전 파일의 상태를 확인하여 파일접근이 가능한지 확인하기 위해 사용합니다.

```
/*******************************************************
예제 : 파일체크
*******************************************************/
void FileAccess()
{
    int nRet;
    string str;

     // notepad.exe 파일이 상태를 정수형 변수 nRet에 저장합니다.
    nRet = _FileAccess("C:₩Windows₩notepad.exe", 0);

     // 정수형 변수 nRet의 값을 문자와 조합한 문자열 변수 str에 저장합니다.
    str = _FormatString("FileAccess : %d", nRet);

     // 문자열 변수 str의 값을 메시지박스에 표시합니다.
    _MsgBox(str, "", 0, 2);

        /*************************************************************************************************
        결과를 확인하기 위한 방법은 아래와 같습니다.
        1. eRun을 실행한다 [F5]
        2. [F7] 함수수동호출 을 활성화 시킨후 본 사용자 정의 함수를 실행시킨다.
        *************************************************************************************************/
```

2.4.3 _FileClose()

함수원형

void _FileClose(int nFile)

파라메터	nFile : 파일 핸들번호.
리턴형	없음
설명	열려있는 파일을 닫습니다.
함수활용	파일을 읽거나 쓰기위해서 파일을 오픈 후 더이상 파일에 작업을 하지 않는 경우 사용합니다.

```
/*********************************************************
예제 : 파일닫기
*********************************************************/
void FileCloseTest()
{
	int nFile;
	string strFileName;

	// 초기경로 C 드라이브의 루트로 파일다이얼로그를 띄우고 선택된 파일명을 문자열 변수 filename에 저장합니다.
	strFileName = _FileDialog("C:₩₩", "*.txt", 0);

	// 문자열 변수 filename에 저장된 파일을 오픈하고 상태를 정수형 변수 file에 씁니다.
	nFile =_FileOpen(strFileName, 0);

	// 파일이 정상적으로 열렸으면
	if(nFile>0)
	{
		// 파일에 문자열 상수값 "KOREA"를 씁니다.
		_FileWrite(nFile, "KOREA", 5);

		// 파일쓰기를 종료합니다.
		_FileClose(nFile);
	}
	else {
		_TraceEx("Fail : %s", _GetSystemErrorMsg());
	}

	/*********************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
```

1. 윈도우에 기본으로 설치되어 있는 메모장 등을 이용하여 임의의 txt파일을 생성 및 저장을 한다.
2. eRun을 실행한다 [F5]
3. [F7] 함수수동호출 을 활성화 시킨후 본 사용자 정의 함수를 실행시킨다.
4. 미리 만들어 놓은 임의의 txt파일을 선택한다.
5. 탐색기를 이용해 임의의 txt파일의 내용을 확인한다.

**********/

}

2.4.4 _FileCopy()

함수원형

int _FileCopy(string strDestFile, string strSourceFile, int nFlag)

파라메터	strDestFile : 복사 대상파일 (경로포함) strSourceFile : 복사하려는 원본파일 (경로포함) nFlag : 파일복사 플래그 1 : 파일이 있어도 복사한다. (Overwirte) 0 : 파일이 이미 있으면 복사하지 않는다.
리턴형	0 : 실패 0이 아닌 정수 : 성공
설명	strSourceFile에 해당하는 파일을 strDestFile로 복사한다
함수활용	

```
/*******************************************************
예제 : 파일복사
*******************************************************/
void FileCopy()
{
    int nRet;
    string str;

     // test2.txt 파일이 이미 있으면 복사하지 않는다.
    nRet = _FileCopy("d:₩₩test2.txt", "d:₩₩test.txt", 0);
      str = _FormatString("FileCopy : %d", nRet);

     // 문자열 변수 str의 값을 메시지박스에 표시합니다.
    _MsgBox(str, "", 0, 2);

        /*************************************************************************************************
        결과를 확인하기 위한 방법은 아래와 같습니다.
        1. 윈도우에 기본으로 설치되어 있는 메모장 등을 이용하여 d드라이브에 test.txt 파일을 생성 및 저장을 한다.
        2. eRun을 실행한다 [F5]
        3. [F7] 함수수동호출 을 활성화 시킨후 본 사용자 정의 함수를 실행시킨다.
        4. 탐색기를 이용해 새로생성된 txt파일의 확인한다.
        *************************************************************************************************/
}
```

2.4.5 _FileDelete()

함수원형

void _FileDelete(string strFile)

파라메터	strFile : 경로명이 포함된 삭제할 파일명의 문자열 상수 또는 변수를 지정합니다.
리턴형	없음
설명	특정파일을 삭제합니다.
함수활용	

```
/*******************************************************
예제 : 파일삭제
*******************************************************/
void FileDelete()
{
    int nRet;
    string str;

     // test2.txt 파일이 상태를 정수형 변수 nRet에 저장합니다.
    nRet = _FileAccess("d:₩₩test2.txt", 0);

     // test2.txt 파일이 존재하면 파일을 삭제합니다.
    if(nRet == 0)
        _FileDelete("d:₩₩test2.txt");

        /*************************************************************************************************
        결과를 확인하기 위한 방법은 아래와 같습니다.
        1. 윈도우에 기본으로 설치되어 있는 메모장 등을 이용하여 d드라이브에 test2.txt 파일을 생성 및 저장
을 한다.
        2. eRun을 실행한다 [F5]
        3. [F7] 함수수동호출 을 활성화 시킨후 본 사용자 정의 함수를 실행시킨다.
        4. 탐색기를 이용해 삭제된 txt파일의 확인한다.
        *************************************************************************************************/
}
```

2.4.6 _FileDialog()

함수원형

string _FileDialog(string strInitPath, string strExt, int nFlag)

파라메터	strInitPath : 초기경로(문자열), 파일선택창이 OPEN될 때 나타나는 초기 폴더 경로를 문자열 상수 또는 변수를 지정합니다. strExt : 필터(문자열), 파일명에 대한 확장자를 문자열변수를 지정합니다. (ex) "*.csv" nFlag : 오픈 플래그(정수형) 저장 또는 읽기에 대한 플래그를 지정. 0 : 다른 이름으로 저장 1 : 파일열기
리턴형	경로가 포함된 문자열
설명	파일을 저장 또는 읽기 위한 파일선택 대화창을 OPEN하고 선택된 파일명을 문자열로 돌려준다.
함수활용	

```
/*********************************************************
예제 : 파일 선택창 열기
*********************************************************/
void FileDialog()
{
	int nFile;
	string strFileName;

	// 초기경로 C 드라이브의 루트로 파일다이얼로그를 띄우고 선택된 파일명을 문자열 변수 filename에 저장합니다.
	strFileName = _FileDialog("C:₩₩", "*.txt", 0);

	// 문자열 변수 filename에 저장된 파일을 오픈하고 상태를 정수형 변수 file에 씁니다.
	nFile =_FileOpen(strFileName, 0);

	// 파일이 정상적으로 열렸으면
	if(nFile>0)
	{
		// 파일에 문자열 상수값 "KOREA"를 씁니다.
		_FileWrite(nFile, "KOREA", 5);

		// 파일쓰기를 종료합니다.
		_FileClose(nFile);
	}
```

```
	else {
		_TraceEx("Fail : %s", _GetSystemErrorMsg());
	}

	/*********************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. 윈도우에 기본으로 설치되어 있는 메모장 등을 이용하여 임의의 txt파일을 생성 및 저장을 한다.
	2. eRun을 실행한다 [F5]
	3. [F7] 함수수동호출 을 활성화 시킨후 본 사용자 정의 함수를 실행시킨다.
	4. 미리 만들어 놓은 임의의 txt파일을 선택한다.
	5. 탐색기를 이용해 임의의 txt파일의 내용을 확인한다.
	*********************************************************************************************/
}
```

2.4.7 _FileOpen()

함수원형

int _FileOpen(string strFile, int nFlag)

파라메터	strFile : 경로명이 포함된 OPEN할 파일명의 문자열 상수 또는 변수를 지정합니다. nFlag : 파일 저장 또는 읽기에 대한 플래그를 지정합니다. 0 : 저장모드 1 : 읽기모드
리턴형	> 0 : 파일 핸들번호 -1 : 실패 * 실패시 시스템오류 메시지확인을 위해서 _GetSystemErrorMsg() 함수를 사용해서 확인하면 된다
설명	파일을 쓰기 또는 읽는 경우 사용합니다.
함수활용	

```
/*********************************************************
예제 : 파일처리
*********************************************************/
void FileOpen()
{
	int nFile;
	string strFileName;

	// 초기경로 C 드라이브의 루트로 파일다이얼로그를 띄우고 선택된 파일명을 문자열 변수 filename에 저장합니다.
	strFileName = _FileDialog("C:₩₩", "*.txt", 0);

	// 문자열 변수 filename에 저장된 파일을 오픈하고 상태를 정수형 변수 file에 씁니다.
	nFile = _FileOpen(strFileName, 0);

	// 파일이 정상적으로 열렸으면
	if(nFile>0)
	{
		// 파일에 문자열 상수값 "KOREA"를 씁니다.
		_FileWrite(nFile, "KOREA", 5);

		// 파일쓰기를 종료합니다.
		_FileClose(nFile);
	}
```

```
	else {
		_TraceEx("Fail : %s", _GetSystemErrorMsg());
	}

	/*********************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. 윈도우에 기본으로 설치되어 있는 메모장 등을 이용하여 임의의 txt파일을 생성 및 저장을 한다.
	2. eRun을 실행한다 [F5]
	3. [F7] 함수수동호출 을 활성화 시킨후 본 사용자 정의 함수를 실행시킨다.
	4. 미리 만들어 놓은 임의의 txt파일을 선택한다.
	5. 탐색기를 이용해 임의의 txt파일의 내용을 확인한다.
	*********************************************************************************************/
}
```

2.4.8 _FileRead()

함수원형

int _FileRead(int nFile, int nSize)

double _FileRead(int nFile, int nSize)

string _FileRead(int nFile, int nSize)

파라메터	nFile : 현재 OPEN되어 있는 파일의 번호를 정수형 상수 또는 변수를 지정합니다 nSize : 읽으려고 하는 바이트 수(길이)를 정수형 상수 또는 변수를 지정합니다.
리턴형	정수형, 실수형, 문자열형
설명	파일에서 읽을 바이트 수만큼 읽어서, 정수, 실수, 문자열 값으로 돌려줍니다.
함수활용	

```
/*********************************************************
예제 : 파일데이터 읽어오기
*********************************************************/
void FileRead()
{
    int nFile;
    double nTemp;
    string strFileName, str;

     // 파일 읽기를 위한 모드로 파일선택창을 열어서 파일을 선택합니다.
    strFileName = _FileDialog("C:₩₩", "*.txt", 1);

     // 파일을 읽기모드로 오픈합니다.
    nFile = _FileOpen(strFileName, 1);

     // 파일이 정상적으로 오픈되었으면
    if(nFile > 0)
    {
         // 8바이트를 읽어서 문자열 변수 str에 저장합니다.
        str = _FileRead(nFile, 8);

                    _Trace(str);

         // 파일읽기를 종료합니다.
        _FileClose(nFile);
    }
```

```
	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. 윈도우에 기본으로 설치되어 있는 메모장 등을 이용하여 임의의 txt파일을 생성 합니다.
	2. 위에서 생성한 파일에 "ABCDEFGHIJKLMNOPQR" 을 입력후 저장합니다.
	3. eRun을 실행한다 [F5]
	4. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	5. [F6] 트레이스창을 활성화 시켜 결과를 확인합니다.
	*************************************************************************************************/
}
```

2.4.9 _FileSeek()

함수원형

int _FileSeek(int nFile, int nSize, int nFlag)

파라메터	nFile : 현재 OPEN되어 있는 파일의 번호를 정수형 상수 또는 변수를 지정합니다 nSize : 파일에서 파일 포인터를 이동시킬 바이트 수(길이)를 정수형 상수 또는 변수를 지정합니다. nFlag : 파일포인트 이동 방향 플래그 0 : 파일의 시작위치에서 정해진 바이트 수만큼 이동. 1 : 파일의 현재위치에서 정해진 바이트 수만큼 이동 2 : 파일의 끝에서 정해진 바이트 수만큼 앞으로 이동
리턴형	새롭게 이동한 위치값을 돌려준다.
설명	현재 OPEN된 파일에서 특정 바이트 수 만큼 파일포인터를 이동시킵니다.
함수활용	파일의 구조를 알고있고, 파일내용을 변경하거나 할 때 사용합니다.

```
/*********************************************************
예제 : 파일 포인트 위치이동
*********************************************************/
void FileSeek()
{
    int nFile;
    string strFileName, str;

     // 파일 읽기를 위한 모드로 파일 다이얼로그를 열어서 파일을 선택한다.
    strFileName = _FileDialog("C:₩₩", "*.txt", 1);

     // 파일을 읽기모드로 오픈합니다.
    nFile = _FileOpen(strFileName, 1);

     // 파일이 정상적으로 오픈되었으면
    if(nFile > 0)
    {
        // 파일의 위치를 현재위치에서 8바이트 뒤로 이동시킵니다.
        _FileSeek(nFile, 8, 1);
        // 8바이트를 읽어서 문자열 변수 str에 저장합니다.
        str = _FileRead(nFile, 8);

                    _Trace(str);
```

```
        // 파일읽기를 종료합니다.
        _FileClose(nFile);
    }
        /*************************************************************************************************
        결과를 확인하기 위한 방법은 아래와 같습니다.
        1. 윈도우에 기본으로 설치되어 있는 메모장 등을 이용하여 임의의 txt파일을 생성 합니다.
        2. 위에서 생성한 파일에 "ABCDEFGHIJKLMNOPQR" 을 입력후 저장합니다.
        3. eRun을 실행한다 [F5]
        4. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
        5. [F6] 트레이스창을 활성화 시켜 결과를 확인합니다.
        *************************************************************************************************/
}
```

2.4.10 _FileSize()

함수원형

double _FileSize(string strFile)

파라메터	strFile : 대상파일 (경로포함)
리턴형	파일크기(바이트단위) 실수값.
설명	지정된 파일의 바이트 단위의 크기값을 읽어옵니다.
함수활용	

```
/*********************************************************
예제 : 파일크기 구함
*********************************************************/
void FileSize()
{
    double fFileSize;
    string str;

    // notepad.exe 파일의 크기를 돌려줍니다.
    fFileSize = _FileSize("C:₩windows₩notepad.exe");

    // newfile.dat 파일의 바이트단위 크기와, 1024로 나눈값이 KB 단위 크기가 됩니다.
    str = _FormatString("File size = %d (%.2f Kbytes)", (int)fFileSize, fFileSize / 1024.0);
    _Trace(str);

        /*************************************************************************************************
        결과를 확인하기 위한 방법은 아래와 같습니다.
        1. eRun을 실행한다 [F5]
        2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
        3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
        *************************************************************************************************/
}
```

2.4.11 _FileWrite()

함수원형

int _FileWrite(int nFile, int nData, int nSize)

int _FileWrite(int nFile, double fData, int nSize)

int _FileWrite(int nFile, string strData, int nSize)

파라메터	nFile : 현재 OPEN되어 있는 파일의 번호를 정수형 상수 또는 변수를 지정합니다 nData : 4바이트 크기의 정수형 데이터 fData : 8바이트 크기의 실수형 데이터 strData : nSize 크기만큼의 문자열 데이터 nSize : 쓰려고 하는 바이트 수.
리턴형	0 : 실패
설명	파일에 정해진 바이트 수만큼 정수, 실수, 문자열데이터를 기록합니다.
함수활용	데이터 형에 따 바이트수 char 1바이트 short 2바이트 int 4바이트 float 4바이트 double 8바이트 string 최대 4096바이트

데이터 형에 따라 바이트 길이는 다음과 같다
char 1바이트
short 2바이트
int 4바이트
float 4바이트
double 8바이트
string 최대 4096바이트

```
/*********************************************************
예제 : 파일에 데이터 쓰기
*********************************************************/
void FileWirte()
{
 int nFile;
        string strFileName;

        // 초기경로 C 드라이브의 루트로 파일다이얼로그를 띄우고 선택된 파일명을 문자열 변수 filename에
```

저장합니다.

```
strFileName = _FileDialog("C:₩₩", "*.txt", 0);

// 문자열 변수 filename에 저장된 파일을 오픈하고 상태를 정수형 변수 file에 씁니다.
nFile =_FileOpen(strFileName, 0);

// 파일이 정상적으로 열렸으면
if(nFile>0)
{
	// 파일에 문자열 상수값 "KOREA"를 씁니다.
	_FileWrite(nFile, "KOREA", 5);

	// 파일쓰기를 종료합니다.
	_FileClose(nFile);
}
else {
	_TraceEx("Fail : %s", _GetSystemErrorMsg());
}

/*********************************************************************************************
결과를 확인하기 위한 방법은 아래와 같습니다.
1. 윈도우에 기본으로 설치되어 있는 메모장 등을 이용하여 임의의 txt파일을 생성 및 저장을 한다.
2. eRun을 실행한다 [F5]
3. [F7] 함수수동호출 을 활성화 시킨후 본 사용자 정의 함수를 실행시킨다.
4. 미리 만들어 놓은 임의의 txt파일을 선택한다.
5. 탐색기를 이용해 임의의 txt파일의 내용을 확인한다.
*********************************************************************************************/ }
```

2.4.12 _FileWriteString()

함수원형

int _FileWriteString(int nFile, string strData)

파라메터	nFile : 현재 OPEN되어 있는 파일의 번호를 정수형 상수 또는 변수를 지정합니다 strData : 파일에 쓸 문자열 데이터 (최대 4096개 바이트)
리턴형	0 : 실패
설명	파일에 문자열데이터를 쓰는데 기록합니다. 문자열 뒤에 줄바꿈 문자(₩n)가 자동추가 됩니다.
함수활용	주기적으로 태그데이터를 CSV파일에 기록할 때 사용할 수 있습니다. 프로젝트 실행로그를 임의의 형식으로 기록할 때 사용할 수 있습니다.

```
/*********************************************************
예제 : 파일에 데이터 쓰기
*********************************************************/
void FileWirteString()
{
	int nFile, ret;
	string strData;
	string strFileName;

	// 초기경로 C 드라이브의 루트로 파일다이얼로그를 띄우고 선택된 파일명을 문자열 변수 filename에 저장합니다.
	strFileName = _FileDialog("C:₩₩", "*.txt", 0);

	// 문자열 변수 filename에 저장된 파일을 오픈하고 상태를 정수형 변수 file에 씁니다.
	nFile =_FileOpen(strFileName, 0);

	// 파일이 정상적으로 열렸으면
	if(nFile>0)
	{
		// 파일에 문자열 상수값 "KOREA"를 씁니다.
		ret = _FileWriteString(nFile, "KOREA");

		strData = _FormatString("%d,%f,%s₩r₩a", 100, 123.5, "this is sample");
		ret = _FileWriteString(nFile, strData);
		if(ret <0) {
			_TraceEx("Fail : %s", _GetSystemErrorMsg());
		}
```

```
		// 파일쓰기를 종료합니다.
		_FileClose(nFile);
	}
	else {
		_TraceEx("Fail : %s", _GetSystemErrorMsg());
	}
}
	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. 윈도우에 기본으로 설치되어 있는 메모장 등을 이용하여 임의의 txt파일을 생성 및 저장을 한다.
	2. eRun을 실행한다 [F5]
	3. [F7] 함수수동호출 을 활성화 시킨후 본 사용자 정의 함수를 실행시킨다.
	4. 미리 만들어 놓은 임의의 txt파일을 선택한다.
	5. 탐색기를 이용해 임의의 txt파일의 내용을 확인한다.
	*************************************************************************************************/ }
```

2.4.13 _LogToFile()

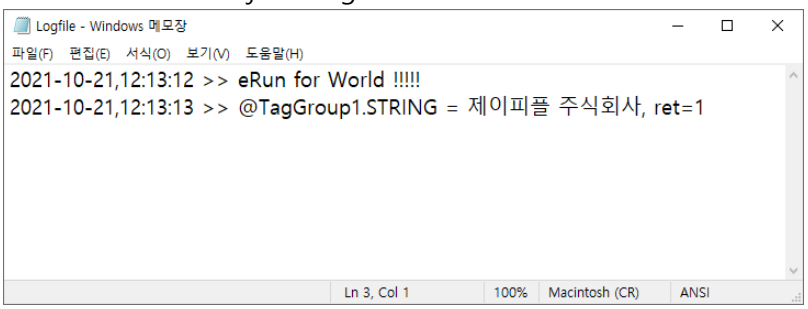
함수원형

void _LogToFile(string strText)

파라메터	strText : 저장할 메시지를 문자열 상수 또는 변수를 지정합니다.
리턴형	없음
설명	Publish 폴더에 Logfile.txt 파일에 메시지를 기록합니다.
함수활용	입력 이벤트 또는 사용자 입력사항 등을 텍스트로그로 저장할 때 사용합니다

```
/*************************************************************
예제 : 로그기록
*************************************************************/
void LogToFile()
{
	_LogToFile("eRun for World !!!!!");
	_LogToFile("제이피플 주식회사");


	/*****************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	3. c:₩₩SEFA₩eRun₩₩Publish₩₩프로젝트이름₩₩LogFile.txt 를 열어 결과를 확인합니다.
	*****************************************************************************************************/
}
```

※ Publish₩TestProject₩Logfile.txt



2.4.14 _LogToFileEx()

함수원형

void _LogToFileEx(string strFile, string strFormat, var1, var2, ...)

파라메터	strFile : 경로가 포함한 로그파일명 strFormat : 서식지정자("%d,%s,%f,...) var1 : 서식지정자에 해당하는 값1 var2 : 서식지정자에 해당하는 값2 ... : n개의 서식지정자에 해당하는 값
리턴형	없음
설명	프로젝트 실행상태, 운용내용 등을 지정파일에 기록합니다.
함수활용	입력 이벤트 또는 사용자 입력사항 등을 텍스트로그로 저장할 때 사용합니다

여러 종류의 데이터(data)를 다양한 서식에 맞춰 파일에 출력할 수 있게 해줍니다. 스크립트 실행 로그를 기록하거나, 태그값 등을 시간별 기록합니다.
파라메터 개수가 최소 3개이상 이어야 하며, 파일명(strFile), 서식지정자(strFormat), var.
서식지정자에 %로 시작하는 형식이 있으면 셋째 파라메터에 지정된 var값이 이에 대응됩니다.

■ 서식지정자

%c ASCII 문자로 표시
%s 문자열
%d 부호 있는 10진 정수
%i 부호 있는 10진 정수 (%d와 동일)
%f 고정 소수점으로 표현한 실수 (소수점 이하 6자리까지 표현)
%o 부호 없는 8진 정수
%u 부호 없는 10진 정수
%x 부호 없는 16진 정수 (소문자 사용)
%X 부호 없는 16진 정수 (대문자 사용)
%e 부동 소수점으로 표현한 실수 (지수형태)
%E 부동 소수점으로 표현한 실수 (지수형태)
%g 값에 따라 %f나 %e를 사용함.
%G 값에 따라 %f나 %E를 사용함.
%% 퍼센트(%) 기호 출력

■ 파라메터

> strFormat : 서식지정자("%d,%s,%f,...), %지정자에 맞게 데이터 개수가 반드시 일치해야 합니다.
"**%[플래그(flag)][폭(width)][.정밀도][크기(length)]서식 문자(specifier)** "

> data : 서식지정자에 해당하는 값
> ... : n개의 서식지정자에 해당하는 값

■ 유사함수
다음의 내부함수들이 동일한 형식으로 가변 파라메터 개수를 사용합니다.
_TraceEx(), _FormatString(), _ComMethod(), _HSMSMethod(), _VisionMethod()

■ 예제

```
/*********************************************************
예제 : 로그기록
*********************************************************/
void LogToFile()
{
	_LogToFileEx("C:₩₩temp₩₩logtest.txt", "함수는 서식 지정자를 통해 출력할 데이터의 서식을 지정할 수 있어요!₩n");
	// -> 서식지정자가 없는 경우, 두번째 문자열 텍스트를 logtest.txt파일에 줄바꿈 개행문자(₩n) 포함해서 저장한다.
	_LogToFileEx("C:₩₩temp₩₩logtest.txt", "변수에 저장된 숫자는 %d입니다.", 10);
	// -> int형 데이터를 나타내기 위해서 '%d'라는 서식 지정자 사용하였다.

	_LogToFileEx("C:₩₩temp₩₩logtest.txt", "%s=%f₩n", "value", 12345.678);
	// -> value=12345.678을 파일에 기록한다.

	_LogToFileEx("C:₩₩temp₩₩logtest.txt", "tag_name = %s₩n", @MAIN.TAGNAME);
	// -> tag_name = '홍길동'을 파일에 기록한다. (@MAIN.TAGNAME의 태그값이 홍길동 이라면)

	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. "C:₩₩temp" 폴더를 생성합니다.
	2. eRun을 실행한다 [F5]
	3. F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	4 "C:₩₩temp" 폴더에 생성된 파일을 확인합니다.
	*************************************************************************************************/
}
```

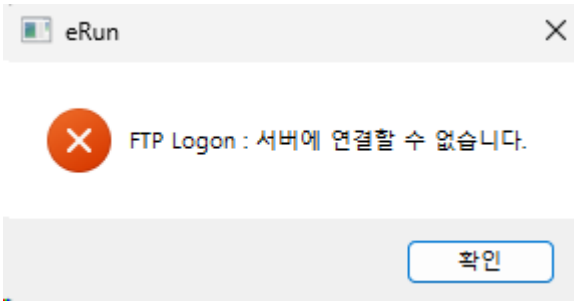
2.4.15 _FTPLogOn()

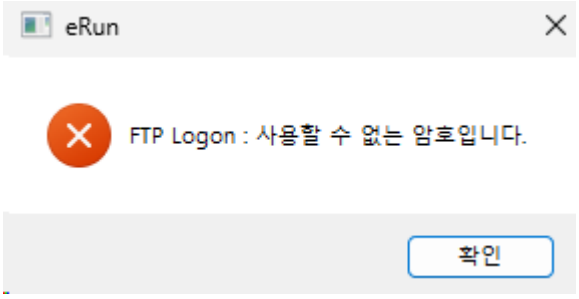
함수원형

int _FTPLogOn(string strKey, string strHost, int nPort, string strUserId, string strPass)

파라메터	strKey : FTP 서버에 연결하기 위한 KEY strHost : FTP 서버 호스트이름 또는 아이피 nPort : 포트번호 (기본 21) strUserId : 사용자 아이디 strPass : 사용자 암호
리턴형	정수 0 : 정상 <0 : 오류 -1 : 이미 사용중인 키값으로 접속을 할 경우. -2 : FTP 서버 접속정보 확인을 하시기 바랍니다.
설명	FTP 서버연결을 시도합니다.
함수활용	리턴값이 0보다 작을 경우 _FTPMsg()함수로 오류메시지 확인이 가능합니다.

[주의] FTP (File Transfer Protocol) 연결을 하기위한 함수입니다. 제공되는 함수는 SSL/TLS 인증서를 사용하지 않기 때문에 FTP 서버측에서 인증서 확인 절차를 OFF 해주어야 정상적인 접속이 가능합니다.
FTP 서버 주소가 정상인 경우 아이디, 암호 확인후 즉시 리턴코드를 확인할 수 있으며, 틀린 주소인 경우 약 20초(time out)정도 기다리면 접속오류 메시지가 나타나니 회신코드가 올 때까지 기다리시기 바랍니다.





■ 파라메터

> strKey : FTP 서버연결이 되면 돌려주는 핸들값입니다.

> strHost : 서버주소를 문자열로 입력합니다.

> nPort : 서버에 설정된 포트번호를 입력합니다. 기본포트번호는 21입니다.

> strUserId : FTP 아이디를 문자열로 입력합니다.

> strPass : FTP 암호를 문자열로 입력합니다.

■ 예제

```
/*********************************************************
예제 : FTP 서버접속
*********************************************************/
void FTPOpen()
{
	int ret;
	text sKey;

	// FTP연결의 핸들로 사용할 KEY 문자열을 임의로 지정합니다.
	sKey = "key10";
	ret = _FTPLogOn(sKey, "192.168.0.65", 21, "ftpuser", "ftppw1234!!@”);
	// 연결이 성공하면 0을 돌려준다.
	_TraceEx("ftp_logon, ret=%d, msg=%s", ret, _FTPMsg(sKey));
	//
}
```

2.4.16 _FTPLogOff()

함수원형

int _FTPLogOff(string strKey)

파라메터	strKey : FTP 서버 연결 Key 문자열
리턴형	정수 성공 : 0 실패 : -1
설명	로그온 된 FTP서버 연결을 종료합니다.
함수활용	리턴값이 0보다 작을 경우 _FTPMsg()함수로 오류메시지 확인이 가능합니다.

■ 파라메터

> strKey : FTP 서버연결이 되면 돌려주는 핸들값입니다.

2.4.17 _FTPUpload()

함수원형

int _FTPUpload(string strKey, string strLocalFile, string strRemoteFile)

파라메터	strKey : FTP 서버 연결 Key 문자열 strLocalFile : 로컬 PC의 경로포함된 파일이름 strRemoteFile : FTP서버의 현재경로에 저장할 파일이름
리턴형	정수 성공 : 0 실패 : -1
설명	로컬 PC에서 FTP서버로 파일을 저장합니다.
함수활용	리턴값이 0보다 작을 경우 _FTPMsg()함수로 오류메시지 확인이 가능합니다.

■ 예제

```
/*********************************************************
예제 : 파일 업로드
*********************************************************/
void FTPUpload()
{
	int ret;
	text sKey;

	sKey = "key10";

	// 로컬PC의 D: 드라이브에 있는 파일을 FTP서버에 _2025_01_24.log  파일로 저장합니다.
	ret =_FTPUpload(sKey, "d:/_25.log", "/_2025_01_24.log");

	// 0보다 작은 값이 들어오면 실패.
	if(ret <0) {
		_TraceEx("ftp_upload, ret=%d, msg=%s", ret, _FTPMsg(sKey));
	}
}
```

2.4.18 _FTPDownload()

함수원형

int _FTPDownload(string strKey, string strRemoteFile, string strLocalFile)

파라메터	strKey : FTP 서버 연결 Key 문자열 strRemoteFile : 로컬 PC의 경로포함된 파일이름 strLocalFile : FTP서버의 현재경로에 저장할 파일이름
리턴형	정수 성공 : 0 실패 : -1
설명	FTP서버의 현재 경로에서 파일을 다운로드해서 로컬 PC 지정폴더에 저장합니다.
함수활용	리턴값이 0보다 작을 경우 _FTPMsg()함수로 오류메시지 확인이 가능합니다.

[주의] FTP 서버에서 특정파일을 다운로드하는데 파라메터의 순서에 주의하시기 바랍니다. 로컬 PC에 동일한 이름의 파일이 있어도 덮어쓰기 됩니다.

■ 예제

```
/*********************************************************
예제 : FTP서버에서 파일 다운로드
*********************************************************/
void FTPDownload()
{
	int ret;
	text sKey;

	sKey = "key10";

	ret =_FTPDownload(sKey, "ReadMe.txt", "d://ReadMe_loc.txt");
	// 0보다 작은 값이 들어오면 실패.
	if(ret <0) {
		_TraceEx("ftp_download, ret=%d, msg=%s", ret, _FTPMsg(sKey));
	}
}
```

2.4.19 _FTPCmd()

함수원형

int _FTPCmd(string strKey, string strCommand)

파라메터	strKey : FTP 서버 연결 Key 문자열 strCommand : FTP서버에 FTP명령을 직접 보냅니다.
리턴형	정수 성공 : 0 실패 : -1 (_FTPMsg 함수
설명	표준화된 FTP 커맨드를 직접 FTP서버로 전송해서 명령을 처리하도록 합니다.
함수활용	리턴값이 0보다 작을 경우 _FTPMsg()함수로 오류메시지 확인이 가능합니다.

[주의] FTP명령을 서버로 직접 보내는 것이기 때문에 사용하실 때 주의를 하셔야 합니다. 파일삭제, 경로변경등 파일의 전송과 관련된 명령을 직접 처리가 필요한 경우에만 사용을 권합니다.

■ FTP 명령어 리스트

NOOP 더미패킷 전송, Keep Alive 처리용
MKD 현재 디렉토리에 폴더를 생성 (MKD dir)
RMD 디렉토리 삭제 (RMD dir)
CDUP 상위 디렉토리로 이동
CWD 디렉토리 이동
DELE 파일삭제

■ 예제

```
/*********************************************************
예제 : FTP 명령어 처리
*********************************************************/
void FTPCmd()
{
	int ret;
	text sKey;

	sKey = "key10";
	ret =_FTPCmd(sKey, “NOOP”);
	_TraceEx("ftp_command(%s), ret=%d, msg=%s", @FTP.Command , ret, _FTPMsg(sKey));
}
```

2.4.20 _FTPMsg()

함수원형

string _FTPMsg(string strKey)

파라메터	strKey : FTP 서버 연결 Key 문자열
리턴형	문자열
설명	FTP 명령을 처리한 후 결과메시지를 읽어옵니다.
함수활용	

■ 예제

```
/*********************************************************
FTP 열기, 닫기, 업로드, 다운로드, 메시지, 커멘드
*********************************************************/
void FTPOpen()
{
	int ret;
	text sKey;

	sKey = "key10";
	ret =  FTPLogOn(sKey, "192.168.0.65", 21, "root", "roowpw1234");
	_TraceEx("ftp_logon, ret=%d, msg=%s", ret, _FTPMsg(sKey));

	if(ret == 0) {
		@FTP.Open_status = 1;
	}
}

void FTPClose()
{
	int ret;
	text sKey;

	sKey = "key10";
	ret =_FTPLogOff(sKey);
	_TraceEx("ftp_logoff, ret=%d, msg=%s", ret, _FTPMsg(sKey));

	if(ret == 0) {
		@FTP.Open_status = 0;
```

```
		}
}

void FTPCmd(int nCategory)
{
		int ret;
		text sKey;

		sKey = "key10";
		if(nCategory==1) {
			ret =_FTPCmd(sKey, "NOOP");
			_TraceEx("ftp_command(NOOP), ret=%d, msg=%s", ret, _FTPMsg(sKey));
		}
		else {
			ret =_FTPCmd(sKey, @FTP.Command);
			_TraceEx("ftp_command(%s), ret=%d, msg=%s", @FTP.Command , ret, _FTPMsg(sKey));
		}
}

void FTPUpload()
{
		int ret;
		text sKey;

		sKey = "key10";
		ret =_FTPUpload(sKey, "d:/_25.log", "/_2025_01_24.log");
		_TraceEx("ftp_upload, ret=%d, msg=%s", ret, _FTPMsg(sKey));
}

void FTPDownload()
{
		int ret;
		text sKey;

		sKey = "key10";
		ret =_FTPDownload(sKey, "ReadMe.txt", "d://ReadMe_loc.txt");
		_TraceEx("ftp_download, ret=%d, msg=%s", ret, _FTPMsg(sKey));
}
```

2.5 태그 함수

2.5.1 _GetTagInfo()

함수원형

int _GetTagInfo(string strTagname, int nIndex)

double _GetTagInfo(string strTagname, int nIndex)

string _GetTagInfo(string strTagname, int nIndex)

파라메터	strTagname : 태그이름 nIndex : 태그속성 항목 인덱스번호
리턴형	정수형 0 : 성공 -1 : 실패
설명	런타임(run-time)중에 태그속성 항목에 설정된 값을 읽어옵니다.
함수활용	런타임(run-time)중에 경보태그의 설정 값을 변경하거나 경보사용 여부 등을 설정할 때 사용합니다.

※ 인덱스번호는 아래와 같이 정의되어 있습니다.

인덱스	매크로	설명
1	TAG_SCAN	태그 값 스캔
2	TAG_COMMENT	태그설명
3	TAG_SAVE	종료시 태그값 저장
4	TAG_DEFAULT_CHECK	초기값 설정
5	TAG_DEFAULT_VALUE	초기값
101	TAG_ANALOG_OFFSET	아날로그 태그 보정 값
102	TAG_ANALOG_DEADBAND	아날로그 태그 데드밴드 값
103	TAG_ANALOG_SCALE_CHECK	아날로그 태그 스케일 체크
104	TAG_ANALOG_SCALE	아날로그 태그 스케일
105	TAG_ANALOG_RAW_MIN	아날로그 태그 최소 입력값
106	TAG_ANALOG_RAW_MAX	아날로그 태그 최대 입력값
107	TAG_ANALOG_ENG_MIN	아날로그 태그 최소 변환값
108	TAG_ANALOG_ENG_MAX	아날로그 태그 최대 변환값
201	TAG_ALARM_DEADBAND	경보태그 데드밴드 값
202	TAG_ALARM_EQ_CHECK	Digital Alarm 인 경우 경보처리 할 것인가?
203	TAG_ALARM_EQ_VALUE	Digital Alarm인 경우 경보 기준 값
204	TAG_ALARM_HH_CHECK	HiHi Alarm인 경우 경보처리 할 것인가?
205	TAG_ALARM_HH_VALUE	HiHi Alarm인 경우 경보 기준 값

206	TAG_ALARM_H_CHECK	Hi Alarm인 경우 경보처리 할 것인가?
207	TAG_ALARM_H_VALUE	Hi Alarm인 경우 경보 기준 값
208	TAG_ALARM_L_CHECK	Lo Alarm인 경우 경보처리 할 것인가?
209	TAG_ALARM_L_VALUE	Lo Alarm인 경우 경보 기준 값
210	TAG_ALARM_LL_CHECK	LoLo Alarm인 경우 경보처리 할 것인가?
211	TAG_ALARM_LL_VALUE	LoLo Alarm인 경우 경보 기준 값
212	TAG_ALARM_MESSAGE	

```
/*********************************************************
예제 : 특정 태그그룹의 태그정보 갖고 오기
*********************************************************/
void GetTagInfo()
{
	int nRet, nCount;
	double fValue;
	string strText;

	// 태그 "DEMO.출력1" 의 아날로그 보정값 정보를 읽어와서 메시지 박스에 표시합니다.
	fValue = _GetTagInfo("DEMO.출력값1", 101)
	_MsgBox(fValue, "TAG_SCAN", 0, 2);

	// 태그 "DEMO.출력1" 의 태그설명 정보를 읽어와서 메시지 박스에 표시합니다.
	strText = _GetTagInfo("DEMO.출력값1", 2);
	_MsgBox(strText, "태그설명", 0, 2);
}
```

2.5.2 _GetTagValue()

함수원형

int _GetTagValue(string strTagname)

double _GetTagValue(string strTagname)

string _GetTagValue(string strTagname)

파라메터	strTagname : 태그이름
리턴형	정수, 실수, 문자열값
설명	태그에 해당하는 태그값을 메모리에서 읽어옵니다
함수활용	태그이름을 변수로 사용해서 태그값을 읽을 때 사용합니다.

※ 태그값을 가져오는 방법은 두가지가 있습니다.

(1) 직접 태그로부터 가져오는 방법은 아래와 같은 형식입니다.
변수 = @TAG_GROUP.VALUE;

(2) 함수로부터 가져오는 방법은 아래와 같은 형식입니다.
변수 = _GetTagValue("TAG_GROUP.VALUE");

```
/*********************************************************
예제 : 태그 값 읽기
*********************************************************/
void GetTagValue()
{
string name;
double nTemp;
	int nValue;

	// 태그 "sample.name"의 값을 _GetTagValue() 함수를 통해서 읽어옵니다.
	name = _GetTagValue("sample.name");
	// 문자열 변수 name의 값을 메시지박스에 표시합니다.
	_MsgBox (name, "확인", 0, 0);

	// 태그 "sample.name"의 값을 @접두어를 사용하여 문자열 변수 name에 저장합니다.
	name = @sample.name;
	// 태그 "메인태그.현재값"의 값을 _GetTagValue() 함수를 통해서 정수형 값을 읽어옵니다.
	nValue =_GetTagValue("메인태그.현재값");
	_MsgBox (nValue, "확인", 0, 0);
```

```
    // 태그명이 "메인태그.현재값"의 값을 @접두어를 사용하여 문자열 변수 name에 저장합니다.
    nValue = @메인태그.현재값;
    // 정수형 변수 nValue의 값을 메시지박스에 표시합니다.
    _MsgBox (nValue, "확인", 0, 0);

    // 태그 "sample.온도"의 값을 _GetTagValue() 함수를 통해서 실수형 값을 읽어옵니다.
    nTemp =_GetTagValue("sample.온도");
    _MsgBox (nTemp, "확인", 0, 0);

    // 태그 "sample.온도"의 값을 @접두어를 사용해서 읽어옵니다
    nTemp = @sample.온도;
    _MsgBox (nTemp, "확인", 0, 0);
}
```

2.5.3 _InitTagValue()

함수원형

void _InitTagValue(string strTagGroup, int nInitMode)

파라메터	strTagGroup : 태그그룹 이름 nInitMode : 초기값 처리모드 0 : 무조건 초기값으로 변경 1 : 변경없음
리턴형	없음
설명	특정 태그그룹에 있는 태그들의 현재 값을 초기값으로 변경합니다.
함수활용	

```
/*******************************************************
예제 : 태그속성에서 경보설정치 변경하기
*******************************************************/
void InitTagValue()
{
	// 태그그룹명이 "DEMO"인 태그의 현재값을 초기값으로 변경합니다.
	_InitTagValue("DEMO", 0);
}
```

2.5.4 _SetTagInfo()

함수원형

int _SetTagInfo(string strTagname, int nIndex, int nValue)

int _SetTagInfo(string strTagname, int nIndex, double fValue)

int _SetTagInfo(string strTagname, int nIndex, string strValue)

파라메터	strTagname : 태그이름 nIndex : 설정 값을 변경하고자 하는 태그속성 인덱스번호 nValue : 정수형 변경 값 fValue : 실수형 변경 값 strValue : 문자열형 변경 값
리턴형	정수형 0 : 성공 -1 : 실패
설명	런타임(run-time)중에 태그속성을 변경해야 할 경우 사용합니다.
함수활용	런타임(run-time)중에 경보태그의 설정 값을 변경하거나 경보사용 여부 등을 설정할 때 사용합니다.

※ 인덱스번호에 따라서 FALSE(0), TRUE(1) 또는 정수, 실수, 문자열 값을 입력합니다

※ 인덱스번호는 아래와 같이 정의되어 있습니다.

인덱스	매크로	설명
1	TAG_SCAN	태그 값 스캔
2	TAG_COMMENT	태그설명
3	TAG_SAVE	종료시 태그값 저장
4	TAG_DEFAULT_CHECK	초기값 설정
5	TAG_DEFAULT_VALUE	초기값
101	TAG_ANALOG_OFFSET	아날로그 태그 보정 값
102	TAG_ANALOG_DEADBAND	아날로그 태그 데드밴드 값
103	TAG_ANALOG_SCALE_CHECK	아날로그 태그 스케일 체크
104	TAG_ANALOG_SCALE	아날로그 태그 스케일
105	TAG_ANALOG_RAW_MIN	아날로그 태그 최소 입력값
106	TAG_ANALOG_RAW_MAX	아날로그 태그 최대 입력값
107	TAG_ANALOG_ENG_MIN	아날로그 태그 최소 변환값
108	TAG_ANALOG_ENG_MAX	아날로그 태그 최대 변환값
201	TAG_ALARM_DEADBAND	경보태그 데드밴드 값
202	TAG_ALARM_EQ_CHECK	Digital Alarm 인 경우 경보처리 할 것인가?

203	TAG_ALARM_EQ_VALUE	Digital Alarm인 경우 경보 기준 값
204	TAG_ALARM_HH_CHECK	HiHi Alarm인 경우 경보처리 할 것인가?
205	TAG_ALARM_HH_VALUE	HiHi Alarm인 경우 경보 기준 값
206	TAG_ALARM_H_CHECK	Hi Alarm인 경우 경보처리 할 것인가?
207	TAG_ALARM_H_VALUE	Hi Alarm인 경우 경보 기준 값
208	TAG_ALARM_L_CHECK	Lo Alarm인 경우 경보처리 할 것인가?
209	TAG_ALARM_L_VALUE	Lo Alarm인 경우 경보 기준 값
210	TAG_ALARM_LL_CHECK	LoLo Alarm인 경우 경보처리 할 것인가?
211	TAG_ALARM_LL_VALUE	LoLo Alarm인 경우 경보 기준 값
212	TAG_ALARM_MESSAGE	

```
/*********************************************************

예제 : 태그속성에서 경보설정치 변경하기
*********************************************************/

void SetTagInfo(int mode)
{
        // 매크로 사용시.
        if(mode==1) {
    _SetTagInfo("MBSVR.VALUE", TAG_COMMENT, "측정값 설명");
    _SetTagInfo("MBSVR.VALUE", TAG_ANALOG_RAW_MIN , 0);
    _SetTagInfo("MBSVR.VALUE", TAG_ANALOG_RAW_MAX , 250);
    _SetTagInfo("MBSVR.VALUE", TAG_ANALOG_ENG_MIN , 0);
    _SetTagInfo("MBSVR.VALUE", TAG_ANALOG_ENG_MAX , 250);
  }
  else if(mode==2) { // 인덱스 사용시.
    _SetTagInfo("MBSVR.VALUE", 2  , "Tag Comment");
    _SetTagInfo("MBSVR.VALUE", 105 , 0);
    _SetTagInfo("MBSVR.VALUE", 106 , 100);
    _SetTagInfo("MBSVR.VALUE", 107 , 0);
    _SetTagInfo("MBSVR.VALUE", 108 , 100);
  }
}
```

2.5.5 _SetTagValue()

함수원형

int _SetTagValue(string strTagname, int nValue)

int _SetTagValue(string strTagname, double fValue)

int _SetTagValue(string strTagname, string strValue)

파라메터	strTagname : 태그이름 nValue : 정수값 fValue : 실수값 strValue : 문자열 값
리턴형	정수값 0이면 정상 0보다 작으면 FAIL -1 : 일반오류 -2 : 정의 안된 태그사용 -3 : 경보태그는 쓰기 못함 -4 : 이전값과 동일한 값을 쓸경우(무조건 내보내기 속성이 없는 태그의 경우)
설명	태그에 값을 저장할 때 사용합니다. 정의안된 태그가 사용되었을 경우 0보다 작은수를 돌려줍니다.
함수활용	태그이름을 변수로 사용해서 반복적인 수행을 해서 태그값을 저장할 때 사용합니다.

※ 태그값을 저장하는 방법은 두가지가 있습니다.

(1) 직접 태그명에 대입해서 저장하는 방법은 아래와 같은 형식입니다.
@TAG_GROUP.VALUE = 값;

(2) 함수이용해서 태그값 저장 방법은 아래와 같은 형식입니다.
_SetTagValue("TAG_GROUP.VALUE", 1000);
_SetTagValue("TAG_GROUP.VALUE", 10.456);
_SetTagValue("TAG_GROUP.VALUE", "Run");

```
/*********************************************************
예제 : 태그 값 쓰기
*********************************************************/
void SetTagValue()
{
        // 태그명 "sample.name"에 _SetTagValue()를 이용하여 문자열 "SEFATECH"를 저장합니다.
        _SetTagValue("sample.name", "SEFATECH");
```

```
    // 태그명 "sample.name"에 @접두어를 이용하여 문자열 "SEFATECH"를 저장합니다.
    @sample.name = "SEFATECH";

    // 태그명 "메인태그.현재값"에 _SetTagValue()를 이용하여 정수 30을 저장합니다.
    _SetTagValue("메인태그.현재값", 30);
    // 태그명 "메인태그.현재값"에 @접두어를 이용하여 정수 30을 저장합니다.
    @메인 태그.현재값 = 30;

    // 태그명 " sample.온도"에 _SetTagValue()를 이용하여 실수 23.5을 저장합니다.
    _SetTagValue("sample.온도", 23.5);
    // 태그명 " sample. 온도 "에 @접두어를 이용하여 정수 23.5을 저장합니다.
    @sample.온도 = 23.5;
}
```

2.5.6 _ShowTagView()

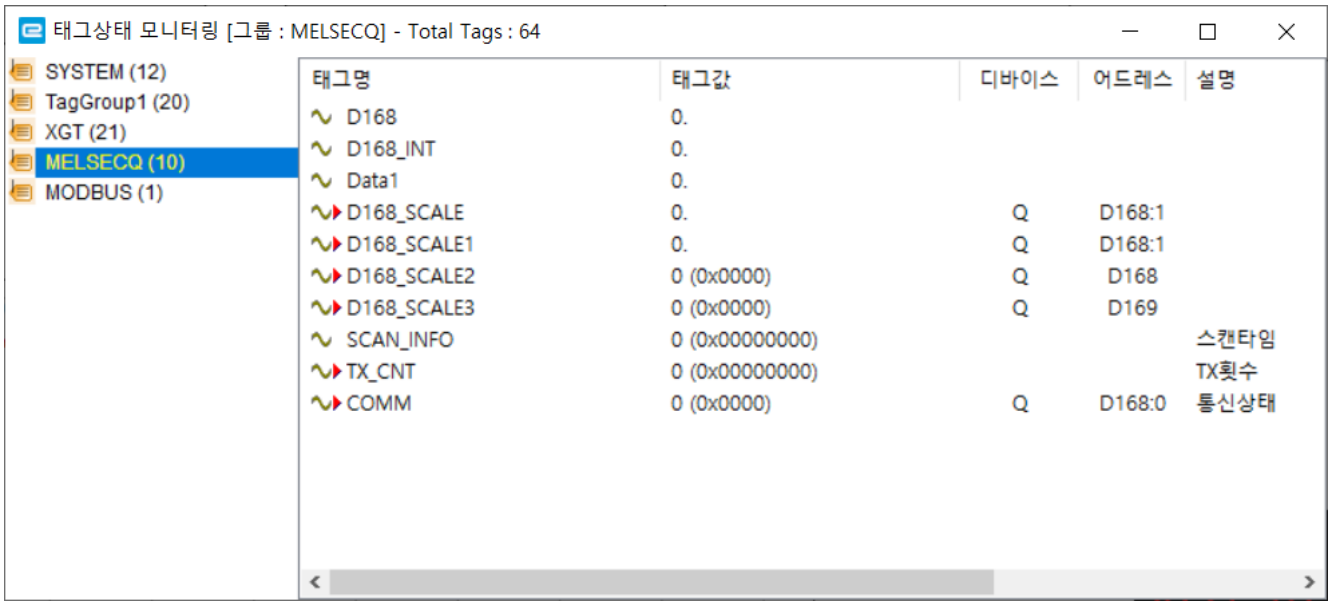
함수원형

void _ShowTagView()

파라메터	없음
리턴형	없음
설명	태그값 모니터링 창을 띄웁니다.
함수활용	팝업메뉴 창 비활성화 되었을 경우에 함수로 창 띄울 때 사용합니다.

```
/*********************************************************
예제 : 태그 모니터링 창 띄우기
*********************************************************/
void ShowTagView()
{
        _ShowTagView();
}
```

태그창



2.5.7 _CopyTagValue()

함수원형

int _CopyTagValue(string strTagnameTo, string strTagNameFr)

파라메터	strTagnameTo : 대상 태그명 strTagnameFr : 원본 태그명
리턴형	정수값 0이면 정상 0보다 작으면 FAIL -1 : 일반오류 -2 : 정의 안된 태그사용 -3 : 경보태그는 쓰기 못함 -4 : 이전값과 동일한 값을 쓸경우(무조건 내보내기 속성이 없는 태그의 경우)
설명	태그값을 지정태그에 복사합니다.
함수활용	다수의 태그값을 복사할 경우 사용합니다.

※ 태그값을 복사하는 방법은 두가지가 있습니다.

(1) 직접 태그명에 대입해서 저장하는 방법은 아래와 같은 형식입니다.

@TAG_GROUP.VALUE = @TAG_SUB.VALUE

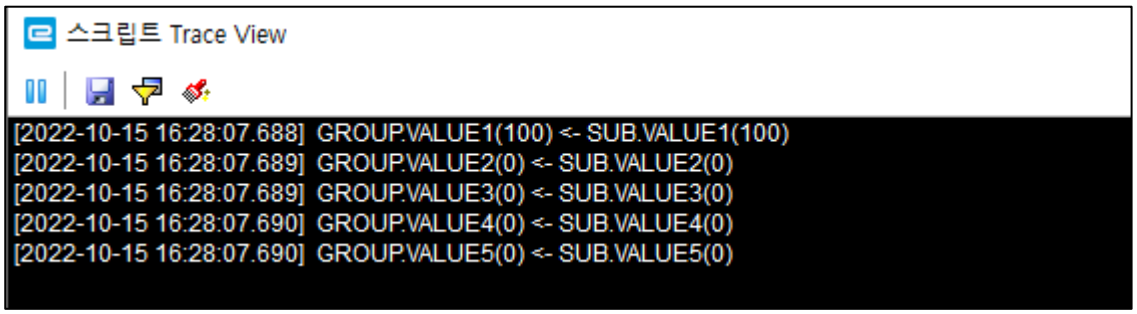
(2) 함수이용해서 태그값을 복사하는 경우 아래와 같은 형식입니다.

_CopyTagValue("TAG_GROUP.VALUE", "TAG_SUB.VALUE");

```
/*********************************************************
예제 : 태그 값 복사
*********************************************************/
void CopyTagValue()
{
	int cnt;
	string strTo, strFr;

	// 정수값 복사
	@SUB.VALUE1 = 100;
	_CopyTagValue("GROUP.VALUE1", "SUB.VALUE1");

	// 문자열 복사
	@SUB.VAL_STRING = "eRun SCADA";
	_CopyTagValue("GROUP.VAL_STRING", "SUB.VAL_STRING");
```

```
        cnt =0;
        while(cnt < 5) {
                strTo = _FormatString("GROUP.VALUE%d", cnt+1);
                strFr = _FormatString("SUB.VALUE%d", cnt+1);

                _CopyTagValue(strTo, strFr);
                // 복사된 태그값을 TraceEx() 함수로 확인해 본다.
                _TraceEx("%s(%d) <- %s(%d)", strTo, _GetTagValue(strTo), strFr, _GetTagValue(strFr));
                cnt++;
        }
}
```

[함수 실행결과]



2.5.8 _CopyTagGroup()

함수원형

int _CopyTagGroup(string strTagGroupTo, string strTagGroupFr, int nOption)

파라메터	strTagGroupTo : 대상 태그그룹명 strTagGroupFr : 원본 태그그룹명 nOption : 예약.
리턴형	정수형 0 : 성공 >0 : 원본 태그그룹에서 복사실패한 개수. -1 : 대상 태그그룹명을 찾을 수 없슴. -2 : 원본 태그그룹명을 찾을 수 없슴. -3 : 대상, 원본 태그그룹의 태그개수가 다름. -4 : 대상그룹이 "SYSTEM" 은 사용불가능.
설명	태그그룹내에 있는 모든 태그값을 다른 태그그룹내의 태그값으로 복사한다. 조건은 태그그룹의 태그명, 태그타입, 태그개수 모두 동일해야 한다.
함수활용	그룹내의 모든 태그값을 일괄적으로 복사할 경우 사용한다.

※ 태그값을 복사하는 방법은 두가지가 있습니다.

_CopyTagGroup("GROUP1", "GROUP2");
"GROUP2"에 있는 모든 태그값을 "GROUP1"에 있는 태그에 복사합니다.
태그리스트의 개수와, 태그이름, 태그속성이 동일한 조건에서 사용가능합니다.

```
/*********************************************************
예제 : 태그그룹 복사
*********************************************************/
void CopyTagGroup()
{
	int ret;

	// "LAYER2" 그룹의 모든 태그값을 "LAYER1" 태그그룹에 복사한다.
	// 세번째 파라메터는 옵션값(예약)
	ret = _CopyTagGroup("LAYER2", "LAYER1", 0);
	_TraceEx("_CopyTagGroup() return %d", ret);
}
```

2.5.9 _SaveTagValues()

함수원형

int _SaveTagValues(string strTagList[], int nCount, string strFilename)

파라메터	strTagList : 태그리스트 문자열 배열변수 nCount : 태그리스트 개수 (최대 1024) strFileName : 경로포함 저장 파일이름.
리턴형	정수형 -1 : FAIL 0 : 저장된 태그가 없음 >0 : 파일에 기록된 태그의 개수.
설명	태그값을 파일에 저장할 경우 사용합니다. 태그리스트는 배열의 형태로 여러 개의 태그그룹명을 포함한 태그명을 첫번째 파라메터로 지정해주고 두번째 파라메터는 태그이름이 들어가있는 배열의 개수를 지정합니다. 세번째 파라메터에 지정된 파일에 "태그그룹.태그이름=태그데이터형,태그값" 형태로 저장합니다.
함수활용	특정한 태그의 값들을 파일에 저장해 두었다가 필요할 때 사용합니다.

※ 시스템 태그, 경보태그는 기록되지 않습니다.

예시 : "TagValues.ini" 파일에 저장된 내용
만약 파일의 내용을 직접 편집할 경우 정상적으로 태그값을 읽어오지 않을 수 있으니 주의해야 합니다.

```
[tag value list]
TEST.ANA0=0,100
TEST.ANA1=6,128.500000
TEST.DIGI0=8,1
TEST.STR_NAME=9,jpeople.co.kr
```

```
/*********************************************************
예제 : 태그데이터 파일에 저장하기
*********************************************************/
void SaveTagValuesToFile()
{
	int ret;
	string sTagList[5];

	sTagList[0] = "TEST.ANA0";
	sTagList[1] = "TEST.ANA1";
```

```
        sTagList[2] = "TEST.DIGI0";
        sTagList[3] = "TEST.STR_NAME";
        sTagList[4] = "TEST.ALARM1";
        //
        ret = _SaveTagValues(sTagList, 5, "d:₩temp₩TagValues.ini");
        _TraceEx("_SaveTagValue()..Save count= %d", ret);
}
```

2.5.10 _LoadTagValues()

함수원형

int _LoadTagValues(string strTagList[], int nCount, string strFilename)

파라메터	strTagList : 태그리스트 문자열 배열변수 nCount : 태그리스트 개수 strFileName : 경로포함 읽어올 파일이름.
리턴형	정수형 -1 : FAIL 0 : 태그개수가 없음 >0 : 파일에서 읽어온 태그의 개수.
설명	파일에 저장된 태그값을 불러올 경우 사용합니다. 함수의 파라메터는 _SaveTagValues() 함수와 동일합니다. 태그리스트는 배열의 형태로 여러 개의 태그그룹명을 포함한 태그명을 첫번째 파라메터로 지정해주고 두번째 파라메터는 태그이름이 들어가있는 배열의 개수를 지정합니다. 세번째 파라메터에 경로가 포함된 파일이름을 지정합니다.
함수활용	그룹내의 모든 태그값을 일괄적으로 복사할 경우 사용한다.

※ 시스템 태그, 경보태그는 읽어오지 않습니다.

```
/*******************************************************
예제 : 태그데이터를 파일에서 읽어오기
*******************************************************/
void LoadTagValuesFromFile()
{
	int ret;
	string sTagList[5];

	sTagList[0] = "TEST.ANA0";
	sTagList[1] = "TEST.ANA1";
	sTagList[2] = "TEST.DIGI0";
	sTagList[3] = "TEST.STR_NAME";
	sTagList[4] = "TEST.ALARM1";		// 경보태그는 읽지 않음.

	// 읽을 태그개수를 돌려준다.
	ret = _LoadTagValues(sTagList, 5, "d:₩temp₩TagValues.ini");
	_TraceEx("_LoadTagValue()..Loadcount = %d", ret);
}
```

2.6 SQL 함수

2.6.1 _SQLClose()

함수원형

int _SQLClose(string strKey)

파라메터	strKey : 연결해제 하려는 KEY
리턴형	정수형 0 : 실패 1 : 성공
설명	SQL DB연결을 종료합니다.
함수활용	

※ _SQLOpen함수와 _SQLClose 함수는 항상 같이 사용하시기 바랍니다.

```
/*********************************************************
예제 : SQL 데이터베이스 연결
*********************************************************/
void SQLOpen()
{
	int nRet;
	// 데이터베이스 SAMPLE_DB에 "K1"키로 연결을 하고 상태를 정수형 변수 nRet에 저장합니다.
	nRet = _SQLOpen("K1", "SAMPLE_DB");
	// 연결에 실패하면 0
	if(nRet < 1)
	{
		// 메시지박스에 SFDB 상태 메시지를 메시지 박스에 표시합니다.
		_MsgBox(_SQLMsg ("K1"), "DB오픈에러", 0, 2);
		return;
	}
	// "K1"키로 데이터베이스에 연결을 종료합니다.
	_SQLClose("K1");
}
```

2.6.2 _SQLGet()

함수원형

int _SQLGet(string strKey, string strSQL, string strField, int nRecord)

double _SQLGet(string strKey, string strSQL, string strField, int nRecord)

string _SQLGet(string strKey, string strSQL, string strField, int nRecord)

파라메터	strKey : SQL DB연결하기 위한 KEY strSQL : 검색 조건에 대한 SQL 문장 strField : 필드이름 nRecord : 필드의 레코드 번호 (0부터 시작)
리턴형	정수형 0 : 테이블 미존재 1 : 테이블 존재
설명	직접 접속한 데이터베이스에 있는 테이블에서 특정 필드의 특정레코드에 해당하는 값을 읽어서 문자열 형태로 돌려줍니다.
함수활용	상용 데이터베이스에서 한개의 데이터(최대값/최소값/평균값/적산차 등)를 검색해야 하는 경우 사용합니다.

※ _SQLGet()함수는 결과값에 따라 Data형으로 자동 변환되므로 변화내부함수 _StrToNum()을 사용할 필요가 없습니다.

```
/*********************************************************
예제 : SQL 특정 필드의 레코드값
*********************************************************/
void SQLGet()
{
	int nRet, nValue;
	string strData, strSQL;

	// 데이터베이스 SAMPLE_DB에 "mdb"키로 연결을 하고 상태를 정수형 변수 nRet에 저장합니다.
	nRet = _SQLOpen("mdb", "SAMPLE_DB");
	strSQL = "Select Top 1, Tag_Value as expr from RAWDATA order by rec_time desc";
	// 쿼리문 strSQL을 실행하고 설정된 레코드번호의 데이터를 정수형 변수 nValue에 저장합니다.
	nValue = _SQLGet("mdb", strSQL, "expr", 0);

	// 정수형 변수 nValue의 값을 메시지박스에 표시합니다.
	_MsgBox(nValue, "결과값", 0, 2);
	// "mdb"키로 데이터베이스에 연결을 종료합니다.
```

```
	_SQLClose("mdb");
}

/*******************************************************
참고하세요
*******************************************************/
void ChangeModel()
{
	string strObject, strSQL;

	_SQLOpen("K10", "SEM");
	strSQL = _FormatString("select Range as expr from [Model] where ModelName = '%s'",
@MODEL.SELECT_MODEL);
	@MODEL.RANGE = _SQLGet("K10", strSQL, "expr", 0);

	//strSQL = _FormatString("select MODE as expr from [Model] where ModelName = '%s'",
@MODEL.SELECT_MODEL);
	//@MODEL.MODE = _SQLGet("K10", strSQL, "expr", 0);
	//----절연
	strSQL = _FormatString("select VRange as expr from [Model] where ModelName = '%s'",
@MODEL.SELECT_MODEL);
	@MODEL.VRange = _SQLGet("K10", strSQL, "expr", 0);

	strSQL = _FormatString("select MARange as expr from [Model] where ModelName = '%s'",
@MODEL.SELECT_MODEL);
	@MODEL.MARange = _SQLGet("K10", strSQL, "expr", 0);
	//----내전압
	strSQL = _FormatString("select IRVRange as expr from [Model] where ModelName = '%s'",
@MODEL.SELECT_MODEL);
	@MODEL.IRVRange = _SQLGet("K10", strSQL, "expr", 0);

	strSQL = _FormatString("select OHMRange as expr from [Model] where ModelName = '%s'",
@MODEL.SELECT_MODEL);
	@MODEL.IR옴 = _SQLGet("K10", strSQL, "expr", 0);
	//-----
	strSQL = _FormatString("select DMM  as expr from [Model] where ModelName = '%s'",
@MODEL.SELECT_MODEL);
	@MODEL.DMM = _SQLGet("K10", strSQL, "expr", 0);

	strSQL = _FormatString("select Temp  as expr from [Model] where ModelName = '%s'",
```

```
@MODEL.SELECT_MODEL);
        @MODEL.Temp = _SQLGet("K10", strSQL, "expr", 0);

        strSQL = _FormatString("select Master  as expr from [Model] where ModelName = '%s'",
@MODEL.SELECT_MODEL);
        @MODEL.Master = _SQLGet("K10", strSQL, "expr", 0);

        strSQL = _FormatString("select L  as expr from [Model] where ModelName = '%s'",
@MODEL.SELECT_MODEL);
        @MODEL.L = _SQLGet("K10", strSQL, "expr", 0);

        strSQL = _FormatString("select H  as expr from [Model] where ModelName = '%s'",
@MODEL.SELECT_MODEL);
        @MODEL.H = _SQLGet("K10", strSQL, "expr", 0);

        strSQL = _FormatString("select OFFSET  as expr from [Model] where ModelName = '%s'",
@MODEL.SELECT_MODEL);
        @MODEL.OFFSET = _SQLGet("K10", strSQL, "expr", 0);

        strSQL = _FormatString("select TimeACW  as expr from [Model] where ModelName = '%s'",
@MODEL.SELECT_MODEL);
        @MODEL.Time_ACW = _SQLGet("K10", strSQL, "expr", 0);

        strSQL = _FormatString("select TimeIR  as expr from [Model] where ModelName = '%s'",
@MODEL.SELECT_MODEL);
        @MODEL.Time_IR = _SQLGet("K10", strSQL, "expr", 0);




        _SQLClose("K10");
}
```

2.6.3 _SQLMsg()

함수원형

string _SQLMsg(string strKey)

파라메터	strKey : SQL DB연결하기 위한 KEY
리턴형	문자열
설명	직접 접속한 데이터베이스의 SQL 명령을 실행한 후 데이터베이스로 부터 받은 결과메시지를 돌려줍니다.
함수활용	상용 데이터베이스에 접속 또는 쿼리문 실행 결과에 따른 상태메시지를 메시지박스 또는 트레이스로 확인하여 올바르게 수행이 되는지 확인할 수 있습니다.

```
/*********************************************************
예제 : SQL 실행 결과(확장)
*********************************************************/
void SQLMsg()
{
	int nRet;
	string strResult;
	// 데이터베이스 SAMPLE_DB에 "mdb"키로 연결을 하고 상태를 정수형 변수 nRet에 저장합니다.
	nRet = _SQLOpen("mdb","SAMPLE_DB");
	// 연결에 실패하면
	if(nRet < 1)
	{
		// SQL 상태 메시지를 문자열 변수 strResult에 저장합니다.
		strResult = _SQLMsg("mdb");
		 // 문자열 변수 strResult의 값을 메시지박스에 표시합니다.
		_MsgBox(strResult, "SQL결과", 0, 0);
		return;
	}
	// "mdb"키로 데이터베이스에 연결을 종료합니다.
	_SQLClose("mdb");
}
```

2.6.4 _SQLOpen()

함수원형

int _SQLOpen(string strKey, string strDatabase)

파라메터	strKey : SQL DB연결하기 위한 KEY strDatabase : 데이터베이스 리스트에 등록된 DB이름.
리턴형	정수형 0 : 실패 1 : 성공
설명	SQL 데이터베이스 모델에 접속합니다.
함수활용	상용 데이터베이스에 저장된 데이터를 조회 및 삭제, 삽입, 갱신 작업을 하기 위해 특정 데이터베이스에 연결을 합니다.

※ _SQLOpen함수와 _SQLClose 함수는 항상 같이 사용하시기 바랍니다.

```
/*********************************************************
예제 : SQL 데이터베이스 연결
*********************************************************/
void SQLOpen()
{
	int nRet;
	// 데이터베이스 SAMPLE_DB에 "mdb"키로 연결을 하고 상태를 정수형 변수 nRet에 저장합니다.
	nRet = _SQLOpen("mdb", "SAMPLE_DB");
	// 연결에 실패하면 0
	if(nRet < 1)
	{
		// 메시지박스에 SQL 상태 메시지를 메시지 박스에 표시합니다.
		_MsgBox(_SQLMsg("mdb"), "DB에러", 0, 2);
		return;
	}
	// "mdb"키로 데이터베이스에 연결을 종료합니다.
	_SQLClose("mdb");
}
```

2.6.5 _SQLOpenCount()

함수원형

int _SQLOpenCount(string strDatabase)

파라메터	strDatabase : 데이터베이스 이름
리턴형	정수형
설명	지정된 데이터베이스에 접속 되어있는 모든 키를 조사해서 연결개수를 돌려줍니다.
함수활용	

```
/*********************************************************
예제 : 접속수
*********************************************************/
void SQLOpenCount()
{
	int nRet;
	string strResult;

	// 데이터베이스 SAMPLE_DB에 "mdb"키로 연결을 하고 상태를 정수형 변수 nRet에 저장합니다.
	nRet = _SQLOpen("mdb","SAMPLE_DB");
	// 연결에 실패하면
	if(nRet < 1)
	{
		// SQL 상태 메시지를 문자열 변수 strResult에 저장합니다.
		strResult = _SQLMsg("mdb");
		 // 문자열 변수 strResult의 값을 메시지박스에 표시합니다.
		_MsgBox(strResult, "SQL결과", 0, 0);
		return;
	}
	nCount = _SQLOpenCount("SAMPLE_DB");
	_TraceEx("연결된 카운트 = %d", nCount);

	// "mdb"키로 데이터베이스에 연결을 종료합니다.
	_SQLClose("mdb");
}
```

2.6.6 _SQLOpenStatus()

함수원형

int _SQLOpenStatus(string strKey)

파라메터	strKey : SQL DB연결하기 위한 KEY
리턴형	정수형 0 : 실패 1 : 열림
설명	DB연결키로 OPEN된 데이터베이스가 있는지 확인합니다.
함수활용	DB연결을 하기전에 이미 열려 있는지 확인할 경우 사용합니다.

```
void SQLRun()
{
	int nRet, nOpen;
	string strSQL, str;

	// "mdb" 키로 DB접속이 되어있지 않으면 OPEN합니다.
	nOpen = _SQLOpenStatus("mdb"), "DB에러", 0, 0);
	if(nOpen == 0) {
		// 데이터베이스 SAMPLE_DB에 "mdb"키로 연결합니다.
		nRet = _SQLOpen("mdb","SAMPLE_DB");

		// 연결에 실패하면
		if(nRet < 1)
		{
		// 메시지박스에 SQL 상태 메시지를 메시지 박스에 표시합니다.
		_MsgBox(_SQLMsg("mdb"), "DB에러", 0, 0);
			return;
		}
	}

	// INSERT 쿼리문을 문자열 변수 strSQL에 저장합니다.
	strSQL = "Insert into Sample_Table(name) values('홍길동')";
	// 쿼리문을 수행하고 결과를 정수형 변수 nRet에 저장합니다.
	nRet = _SQLRun("mdb", strSQL);

	// 쿼리문 실패
	if(nRet < 1)
	{
```

```
            // SQL 상태 메시지를 문자열 변수 str에 저장합니다.
            str = _SQLMsg("mdb");

            // 문자열 변수 str의 값을 메시지박스에 표시합니다.
            _MsgBox(str, "SQL오류", 0, 0);
        }
        // "mdb"키로 데이터베이스에 연결을 종료합니다.
        _SQLClose("mdb");
}
```

2.6.7 _SQLQuery()

함수원형

int _SQLQuery(string strKey, string strObject, string strSQL)

파라메터	strKey : SQL DB연결하기 위한 KEY strObject : 결과 출력 오브젝트 strSQL : SQL 쿼리문
리턴형	검색결과 개수
설명	Open된 SQL 서버에 sql 검색명령을 수행하고 그 결과를 지정된 object에 표시합니다.
함수활용	데이터 삽입, 삭제, 업데이트 등을 수행할 때 사용합니다.

※ SQL 실행문장 문법은 표준 SQL 문법을 따릅니다.

```
/*********************************************************
예제 : SQL 쿼리
*********************************************************/
void SQLQuery()
{
	int nRet;
	string strQuery;

	// 데이터베이스 SAMPLE_DB에 "mdb"키로 연결을 하고 상태를 정수형 변수 nRet에 저장합니다.
	nRet = _SQLOpen("mdb", "SAMPLE_DB");

	// 연결에 실패하면
	if(nRet <= 0)
	{
		// 메시지박스에 SQL 상태 메시지를 메시지 박스에 표시합니다.
		_MsgBox(_SQLMsg("mdb"), "DB에러", 0, 2);
		return;
	}

	// Sample_Table에서 name 필드의 데이터를 요청하는 쿼리문을 문자열 변수 strSQL에 저장합니다.
	//strQuery = "Select name from Sample_Table";
	//strQuery = "Select * from summary where car_no = '1234' order by idx";
	strQuery = "Select car_no from summary order by idx";

	// strQuery 명령을 수행하고 결과를 오브젝트명이 "grid"인 윈도우 오브젝트에 표시합니다.
	nRet = _SQLQuery("mdb", "grid@뷰페이지명", strQuery);
```

```
        // "MAIN_VIEW" 뷰페이지에서 오브젝트 이름이 "grid" 오브젝트에 결과를 표시한다.
        nRet = _SQLQuery("mdb", "grid@MAIN_VIEW", strQuery);

        // 결과 오브젝트 표시는 GRID,

        // "MAIN_VIEW" 뷰페이지에서 오브젝트명이 "콤보"인 오브젝트에 결과를 표시합니다.
        nRet = _SQLQuery("mdb", "콤보@MAIN_VIEW", strQuery);

        // "MAIN_VIEW" 뷰페이지에서 오브젝트명이 "입력창"인 오브젝트에 결과를 표시합니다.
        nRet = _SQLQuery("mdb", "입력창", strQuery);

        // "mdb"키로 데이터베이스에 연결을 종료합니다.
        _SQLClose("mdb");
}
```

2.6.8 _SQLQueryEx()

함수원형

int _SQLQueryEx(string strKey, string strObject, string strSQL, string strTag)

파라메터	strKey : SQL DB연결하기 위한 KEY strObject : 결과 출력 오브젝트 strSQL : SQL 쿼리문 strTag : 정수형 태그를 지정해주고, 수행이 끝나면 1이 들어온다.
리턴형	검색결과 개수
설명	Open된 SQL 서버에 sql 검색명령을 수행하고 그 결과를 지정된 object에 표시하도록 한다.
함수활용	_SQLQuery함수와 다른점은 쿼리를 실행결과를 기다리지 않고 QUERY가 끝나면 지정된 태그에 1을 넣어준다.

※ SQL 실행문장 문법은 표준 SQL 문법을 따릅니다.

```
/*********************************************************
예제 : SQL 쿼리
*********************************************************/
void SQLQuery()
{
	int nRet;
	string strQuery;

	// 데이터베이스 SAMPLE_DB에 "mdb"키로 연결을 하고 상태를 정수형 변수 nRet에 저장합니다.
	nRet = _SQLOpen("mdb", "SAMPLE_DB");

	// 연결에 실패하면
	if(nRet <= 0)
	{
		// 메시지박스에 SQL 상태 메시지를 메시지 박스에 표시합니다.
		_MsgBox(_SQLMsg("mdb"), "DB에러", 0, 2);
		return;
	}

	// Sample_Table에서 name 필드의 데이터를 요청하는 쿼리문을 문자열 변수 strSQL에 저장합니다.
	//strQuery = "Select name from Sample_Table";
	//strQuery = "Select * from summary where car_no = '1234' order by idx";
	strQuery = "Select car_no from summary order by idx";
```

```
	// "MAIN_VIEW" 뷰페이지에서 오브젝트 이름이 "grid"인 오브젝트에 결과를 표시합니다.
	nRet = _SQLQueryEx("mdb", "grid@MAIN_VIEW", strQuery, "TAG.RESULT");

	// "MAIN_VIEW" 뷰페이지에서 오브젝트명이 "콤보"인 오브젝트에 결과를 표시합니다.
	nRet = _SQLQueryEx("mdb", "콤보@MAIN_VIEW", strQuery, "TAG.RESULT");

	// "MAIN_VIEW" 뷰페이지에서 오브젝트명이 "입력창"인 오브젝트에 결과를 표시합니다.
	nRet = _SQLQueryEx("mdb", "입력창@MAIN_VIEW", strQuery, "TAG.RESULT");

	// "mdb"키로 데이터베이스에 연결을 종료합니다.
	_SQLClose("mdb");
}
```

2.6.9 _SQLQueryToFile()

함수원형

int _SQLQueryToFile(string strKey, string strCSVFile, string strSQL, int nCreation)

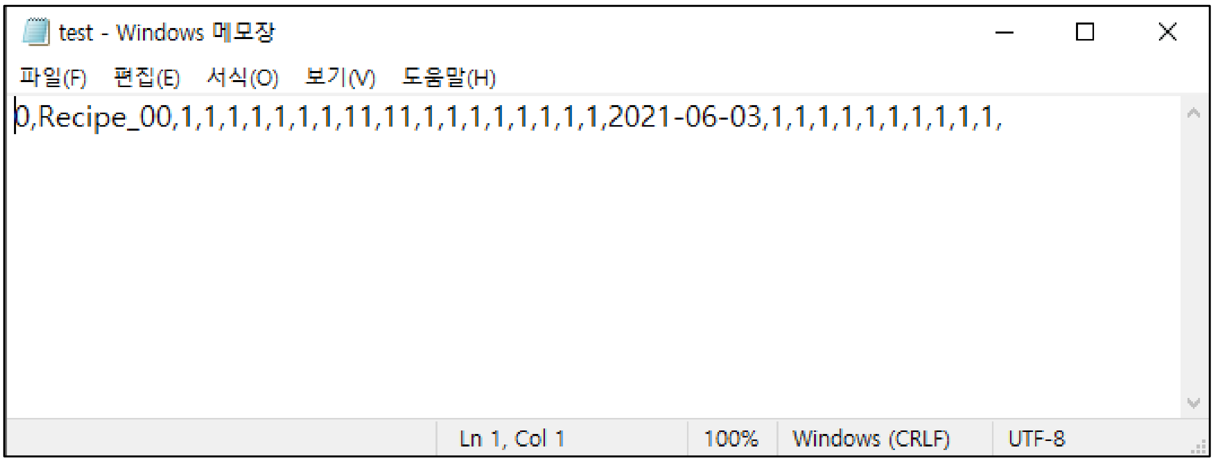
파라메터	strKey : SQL DB연결하기 위한 KEY strCSVFile : 저장파일 문자열 strSQL : SQL 쿼리문 nCreation : 파일생성 옵션, 0=무조건 생성, 1=파일이 없는 경우만 생성.
리턴형	정수형 < 0 : 실패 >=0 : 저장된 개수
설명	Open된 SQL서버에 쿼리명령을 실행하고 그 결과를 CSV 형식의 텍스트파일에 저장합니다.
함수활용	

※ SQL 실행문장 문법은 표준 SQL 문법을 따릅니다.
※ 필드명은 파일에 저장하지 않습니다.

```
/*********************************************************
예제1 : SQL쿼리결과를 파일을 새로 생성하고 CSV 포맷으로 저장.
-> 필드명은 표시되지 않습니다.
*********************************************************/
void QueryToFile1()
{
	int nRet;
	string strQuery;

	// 데이터베이스 SAMPLE_DB에 "mdb"키로 연결을 하고 상태를 정수형 변수 nRet에 저장합니다.
	nRet = _SQLOpen("mdb", "SAMPLE_DB");

	// 연결에 실패하면
	if(nRet <= 0)
	{
		// 메시지박스에 SQL 상태 메시지를 메시지 박스에 표시합니다.
		_MsgBox(_SQLMsg("mdb"), "DB에러", 0, 2);
		return;
	}
	// Sample_Table에서 name 필드의 데이터를 요청하는 쿼리문을 문자열 변수 strSQL에 저장합니다.
	//strQuery = "Select name from Sample_Table";
	//strQuery = "Select * from summary where car_no = '1234' order by idx";
```

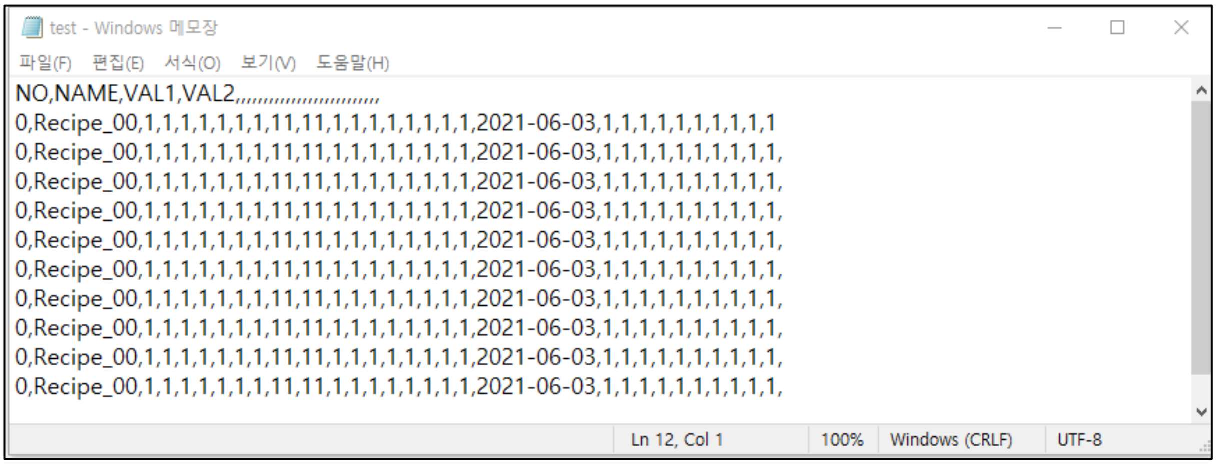
```
	strQuery = "Select car_no from summary order by idx";

	// strQuery 명령을 수행하고 결과를 result.csv 파일에 CSV포맷으로 저장한다.
	// "c:₩₩result.csv" 파일이 이미 있으면 삭제하고 만듭니다.
	nRet = _SQLQueryToFile("mdb", "c:₩₩result.csv", strQuery, 0);

	// "mdb"키로 데이터베이스에 연결을 종료합니다.
	_SQLClose("mdb");
}

/*********************************************************
예제2 : 필드이름을 CSV에 표시하고, SQL쿼리결과를 추가
*********************************************************/
void QueryToFile2()
{
	int ret, row1, row2;
	string strSQL;

	strSQL = "select * from recipeTable";
	ret = _SQLOpen("K1", "RecipeDB");
	if(ret <=0) {
		_MsgBox(_SQLMsg("K1"), "SQL오류", 0, 0);
		return;
	}
	row1 =_SQLQuery("K1", "SFLIST@뷰페이지명", strSQL);

	// CSV파일에 헤더 텍스트를 우선 저장합니다.
	_LogToFileEx("e:₩₩temp2₩₩test.csv", "NO,NAME,%s,%s₩n", "VAL1", "VAL2");

	// 텍스트 헤더가 추가된 파일에 검색결과를 추가 저장합니다.
	row2 =_SQLQueryToFile("K1", "e:₩₩temp2₩₩test.csv", strSQL, 1);
	if(row2<0) {
		_MsgBox(_SQLMsg("K1"), "SQL오류", 0, 0);
	}
	_TraceEx("_SQLQueryToFile..row=%d", row2);
	_SQLClose("K1");
}
```

■ 저장결과 (creation 옵션이 0인 경우)



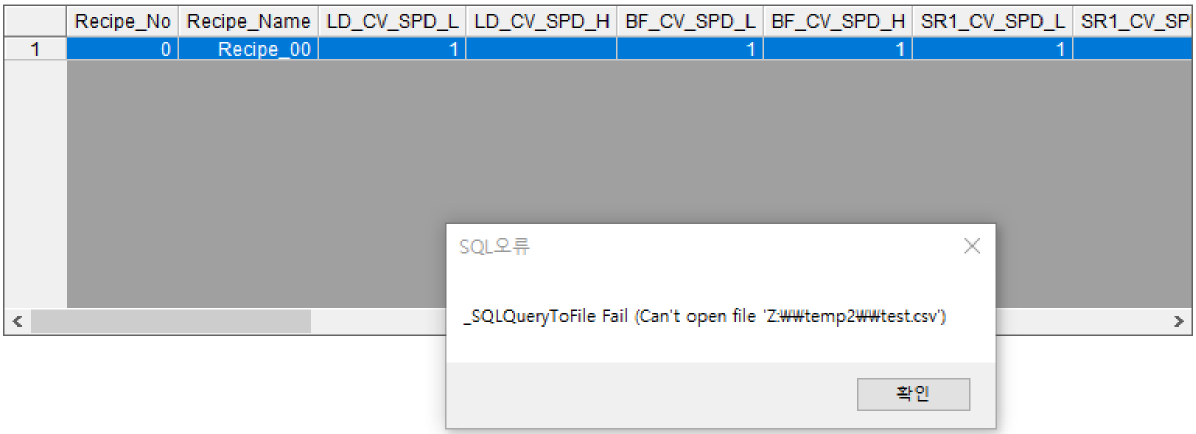
동일한 파일이 있으면 지우고 생성한다.

■ 저장결과 (creation 옵션이 1인 경우) - 헤더표시



파일 생성하지 않고 이미 있는 파일(헤더 표시된)의 마지막줄부터 추가해서 저장한다.

■ 파일명에 오류가 있을경우



2.6.10 _SQLRun()

함수원형

int _SQLRun(string strKey, string strSQL)

파라메터	strKey : SQL DB연결하기 위한 KEY strSQL : SQL 실행문
리턴형	정수형 0 : 실패 1 : 성공
설명	특정 OPEN되어 있는 데이터베이스에서 SQL 문장을 실행합니다.
함수활용	데이터 삽입, 삭제, 업데이트 등을 수행할 때 사용합니다.

※ SQL 실행문장 문법은 표준 SQL 문법을 따릅니다.

```
/*********************************************************
예제 : SQL 명령실행
*********************************************************/
void SQLRun()
{
	int nRet;
	string strSQL, str;

	// 데이터베이스 SAMPLE_DB에 "mdb"키로 연결을 하고 상태를 정수형 변수 nRet에 저장합니다.
	nRet = _SQLOpen("mdb","SAMPLE_DB");

	// 연결에 실패하면
	if(nRet < 0)
	{
	// 메시지박스에 SQL 상태 메시지를 메시지 박스에 표시합니다.
	_MsgBox(_SQLMsg("mdb"), "DB에러", 0, 2);
		return;
	}

	// 테이블 "Sample_Table"에 name 필드에 "Kevin"을 넣어두고 한 개의 레코드를 추가합니다.
	strSQL = "Insert into Sample_Table(name) values('Kevin')";

	// 쿼리문을 수행하고 결과를 정수형 변수 nRet에 저장합니다.
	nRet = _SQLRun("mdb", strSQL);
```

```
	// 쿼리문 실패
	if(nRet < 1)
	{
		// SQL 상태 메시지를 문자열 변수 str에 저장합니다.
		str = _SQLMsg("mdb");

		// 문자열 변수 str의 값을 메시지박스에 표시합니다.
		_MsgBox(str, "SQL오류", 0, 0);
	}
	// "mdb"키로 데이터베이스에 연결을 종료합니다.
	_SQLClose("mdb");
}

/*********************************************************

예제2 : SQL 명령실행

*********************************************************/

void InputTargetDC1()
{
	//타겟이름으로 dB검색후 있으면 usage 업데이트, 없으면 usage 0으로 리셋
	int nRet;
	string sSQL;
	int nCount;
	float fPow;

	if ( "" == _StrTrim(@Main.sTarget_1_Input) )
	{	return;		}

	if (  _SQLOpen("KT1In", "MariaDB") < 0 )	{
		_MsgBox(_SQLMsg("KT1In"), "DB에러", 0, 2);
	}

	sSQL = "SELECT COUNT(*) FROM target WHERE `TargetID`='" + _StrTrim(@Main.sTarget_1_Input) +
"'";
	nRet = _SQLQuery("KT1In", "GRID337@00", sSQL); // _SQLRun("K_Target", sSQL); //GRID337@00
nRet = _SQLQuery("mdb","그리드@뷰페이지명", strSQL);
	nCount = _StrToNum(_GridGetValue("GRID337@00", 0, 0)); //개수 리턴.

	_TraceEx("InputTargetDC1() Select Count(*) nRet[%d], nCount[%d]", nRet, nCount);
```

```
	if ( 1 > nCount)//타겟이 존재하지 않음
	{
		sSQL = "INSERT INTO wafer.target (`TargetID`) VALUES ('" +
_StrTrim(@Main.sTarget_1_Input) + "')";
		nRet = _SQLRun("KT1In", sSQL);

		@Main.sTarget_1 = @Main.sTarget_1_Input;
		_TraceEx("InputTargetDC1() [INSERT INTO wafer.target (`TargetID`)...] nRet[%d]", nRet);
	}
	_SQLClose("KT1In");
}

void UpdateTargetDC1Usage()
{
	//DC 실제 출력(@D6000.D6122)이 있는지 확인해서 출력이 되고 있으면 1초에 한번씩 db update
	int nRet;
	string sSQL;
	int nCount;
	float fPow;
//	float fRate;

	fPow = @D6000.D6122 / 3600.0;
	//fPow = 1.1 / 3600.0;//임시 테스트

	if ( "" == _StrTrim(@Main.sTarget_1) )	{	return;	}

//	if (  _SQLOpen("KT1Us", "MariaDB") < 0 )  {
//		_MsgBox(_SQLMsg("KT1Us"), "DB에러", 0, 2);
//	}

	//현재값 읽어오기
	sSQL = _FormatString("SELECT `Usage` FROM target WHERE `TargetID`='%s'",
_StrTrim(@Main.sTarget_1));
	@Main.nTarget1_Usage = _SQLGet("KT1Us", sSQL, "Usage", 0);


	nCount = _SQLGet("KT1Us", "SELECT COUNT(*) FROM target WHERE `TargetID`='TEST_DC1'",
"COUNT", 0);
//	_TraceEx("InputTargetDC2() _SQLGet nRet = %d", nCount);
```

```
	//Pow 출력되면 업데이트
	if ( 0 < fPow )
	{
		sSQL = _FormatString("UPDATE target SET `Usage`= `Usage` + %f  WHERE
`TargetID`='%s'", fPow, _StrTrim(@Main.sTarget_1) );
		nRet = _SQLRun("KT1Us", sSQL);
//		_TraceEx("UpdateTargetDC1Usage() [%s] nRet = %d", sSQL, nRet);
	}
//	_SQLClose("KT1Us");

	//사용량 확인해서 Life의 몇퍼센트 이상되면 워닝/ 100%이상 알람처리

	if ( 0 >= @Main.nTarget1_Life ) { return;	}

	@Main.nTarget1_Rate = ( @Main.nTarget1_Usage + fPow ) / @Main.nTarget1_Life * 100;

	if ( @Main.nTarget1_Rate >= @Main.nTarget1_Waring){
		//워닝처리
		if ( 1 != @M6000.M6840 ) { @M6000_W.M6840 = 1; }

	}else{
		if ( 0 != @M6000.M6840 ) { @M6000_W.M6840 = 0; }
	}

	if (100 <= @Main.nTarget1_Rate){
		//알람처리
		if ( 1 != @M6000.M6841 ) { @M6000_W.M6841 = 1; }

		if ( 0 == _IsOpenPage("Tar1Alarm") ) { _OpenPage("Tar1Alarm", 1); }

	}else{
		if ( 1 == _IsOpenPage("Tar1Alarm") ) {  _ClosePage("Tar1Alarm"); }
		if ( 0 != @M6000.M6841 ) { @M6000_W.M6841 = 0; }
	}

}

void InputTargetDC2()
```

```
{
	//타겟이름으로 dB검색후 있으면 usage 업데이트, 없으면 usage 0으로 리셋
	int nRet;
	string sSQL;
	int nCount;
	float fPow;

	if ( "" == _StrTrim(@Main.sTarget_2_Input) )		{		return;		}

	if (  _SQLOpen("KT2In", "MariaDB") < 0 )	{
		_MsgBox(_SQLMsg("KT2In"), "DB에러", 0, 2);
	}

	sSQL = "SELECT COUNT(*) FROM target WHERE `TargetID`='" + _StrTrim(@Main.sTarget_2_Input) +
"'";
	nRet = _SQLQuery("KT2In", "GRID338@00", sSQL); // _SQLRun("K_Target", sSQL); //GRID337@00
nRet = _SQLQuery("mdb","그리드", strSQL);
	nCount = _StrToNum(_GridGetValue("GRID338@00", 0, 0)); //개수 리턴.
	_TraceEx("InputTargetDC2() Select Count(*) nRet = %d", nRet);

	if ( 1 > nCount)//타겟이 존재하지 않음
	{
		sSQL = "INSERT INTO wafer.target (`TargetID`) VALUES ('" +
_StrTrim(@Main.sTarget_2_Input) + "')";
		nRet = _SQLRun("KT2In", sSQL);

		@Main.sTarget_2 = @Main.sTarget_2_Input;
//		_TraceEx("InputTargetDC2() [INSERT INTO wafer.target (`TargetID`)...] nRet = %f", nRet);
	}
	_SQLClose("KT2In");
}
```

2.6.11 _SQLTableExist()

함수원형

int _SQLTableExist(string strKey, string strTable)

파라메터	strKey : SQL DB연결하기 위한 KEY strTable : 테이블 이름
리턴형	정수형 0 : 테이블 미존재 1 : 테이블 존재
설명	지정된 데이터베이스 내에 지정된 테이블이 있는지 확인합니다.
함수활용	정시 및 시스템 시작 시 테이블이 없으면 테이블을 생성하여 데이터를 삽입 후 매시간단위 저장된 데이터를 업데이트 해야 하는 경우 사용합니다.

```
/*********************************************************
예제 : 테이블 존재 확인
*********************************************************/
void SQLTableExist()
{
	int nRet;
	string strSQL, str;

	// 데이터베이스 SAMPLE_DB에 "mdb"키로 연결을 하고 상태를 정수형 변수 nRet에 저장합니다.
	nRet = _SQLOpen("mdb", "SAMPLE_DB");

	// 연결에 실패하면
	if(nRet < 0)
	{
		// 메시지박스에 SQL 상태 메시지를 메시지 박스에 표시합니다.
	_MsgBox(_SQLMsg("mdb"), "DB에러", 0, 2);
		return;
	}
	// "mdb" 키로 연결되어있는 데이터베이스에서 "BadProduct" 테이블이 있는지 체크합니다.
	nRet = _SQLTableExist("mdb", "BadProduct");

	// 테이블이 존재하지 않는 경우 메시지 박스 표시
	if(nRet < 1)
	_MsgBox("테이블이 존재하지 않습니다.", "확인", 0, 2);
	}
  // "mdb"키로 데이터베이스에 연결을 종료합니다.
```

```
	_SQLClose("mdb");
}
```

2.6.12 _SQLFormatQuery()

함수원형

int _SQLFormatQuery(string strKey, string strObject, string strSQL, string strFormat)

파라메터	strKey : SQL DB연결하기 위한 KEY strObject : 결과 출력 오브젝트 strSQL : SQL 쿼리문 strFormat : 쿼리 필드에 표시형식 문자열
리턴형	검색결과 개수
설명	Open된 SQL 서버에 sql 검색명령을 수행하고 그 결과를 지정된 필드 표시형식으로 object에 표시하도록 한다.
함수활용	

※ strFormat에 대한 표시형식 문자열
쿼리 필드수 만큼 매치되는 서식지정자 포맷형식을 아래와 같이 사용합니다.

"**%f/%s**" : 필드 2개
"%f**//%d**" : 필드 3개
"**/%f/%.3f/** : 필드 개수 4개

■ 서식지정자
/ 필드구분문자
%s 문자열
%d 부호 있는 10진 정수
%f 고정 소수점으로 표현한 실수 (소수점 이하 6자리까지 표현)
%u 부호 없는 10진 정수
%x 부호 없는 16진 정수 (소문자 사용)
%e 부동 소수점으로 표현한 실수 (지수형태)
%g 값에 따라 %f나 %e를 사용함.

※ SQL 실행문장 문법은 표준 SQL 문법을 따릅니다.

```
/*********************************************************
예제 : SQL 쿼리
*********************************************************/
void SQLQuery()
{
        int nRet;
```

```
	string strQuery;

	// 데이터베이스 SAMPLE_DB에 "mdb"키로 연결을 하고 상태를 정수형 변수 nRet에 저장합니다.
	nRet = _SQLOpen("mdb", "SAMPLE_DB");

	// 연결에 실패하면
	if(nRet <= 0)
	{
		// 메시지박스에 SQL 상태 메시지를 메시지 박스에 표시합니다.
		_MsgBox(_SQLMsg("mdb"), "DB에러", 0, 2);
		return;
	}

	// Sample_Table에서 name 필드의 데이터를 요청하는 쿼리문을 문자열 변수 strSQL에 저장합니다.
	//strQuery = "Select name from Sample_Table";
	//strQuery = "Select * from summary where car_no = '1234' order by idx";
	strQuery = "Select car_no from summary order by idx";

	// strQuery 명령을 수행하고 결과를 오브젝트명이 "LIST"인 윈도우 오브젝트에 표시합니다.
	nRet = _SQLFormatQuery("mdb", "LIST@뷰페이지명", strQuery, "%f/%s/%.2f");

	// "mdb"키로 데이터베이스에 연결을 종료합니다.
	_SQLClose("mdb");
}

/*********************************************************
예제2 : SQL 쿼리
*********************************************************/
void Scan()
{
	string strSQL;
	int isOpen;

	isOpen = _SQLOpenStatus("K10");
	if(isOpen != 1) {
		_Trace("데이터베이스가 Open되지 않아서 기록할 수 없습니다.");
		return;
	}

	strSQL = _FormatString("select judgement,rec_time,car_name,car_no,barcode_f,barcode_r,shell_tq,₩
```

```
                shell_size,loadcell,servo,mark_info,shell_diameter₩
                from product where rec_time >= #%s# and rec_time <= #%s#",
                        @COMMON.SCAN_START, @COMMON.SCAN_END, 3);

        //_SQLQuery("K10", "생산이력리스트@뷰페이지명", strSQL);
        _SQLFormatQuery("K10", "생산이력리스트@뷰페이지명", strSQL,
"%s//%s/%s/%s/%s/%.2f/%.3f/%.1f/%.2f");

        _Trace(strSQL);
        _Trace(_SQLMsg("K10"));
}
```

판정	생산시간	차종이름	차종번호	측매바코드F	측매바코드R	쉘서포트토크	쉘길이측정값	로드셀	래핑스터핑토크
	2021-11-22 오후 1:18:38					1.50	12.560	0.0	0.00
	2021-11-22 오후 1:19:08					1.50	12.560	0.1	0.00
	2021-11-22 오후 1:19:58					1.50	12.560	0.1	0.20
	2021-11-22 오후 1:20:27					1.50	12.560	0.1	0.02

2.7 eRunDB 함수

2.7.1 _SFDBClose()

함수원형

int _SFDBClose(string strKey)

파라메터	strKey : 연결해제 하려는 KEY
리턴형	정수형 0 : 실패 1 : 성공
설명	SFDB의 연결을 종료합니다.
함수활용	

※ _SFDBOpen함수와 _SFDBClose 함수는 항상 같이 사용하시기 바랍니다.

```
/*********************************************************
예제 : SFDB 데이터베이스 연결
*********************************************************/
void SFDBOpen ()
{
	int nRet;
	// 데이터베이스 SAMPLE_DB에 "K1"키로 연결을 하고 상태를 정수형 변수 nRet에 저장합니다.
	nRet = _SFDBOpen("K1", "SAMPLE_DB");
	// 연결에 실패하면
	if(nRet < 1)
	{
		// 메시지박스에 SFDB 상태 메시지를 메시지 박스에 표시합니다.
		_MsgBox(_SFDBMsg ("K1"), "DB오픈에러", 0, 2);
		// 아래 함수구문은 실행되지 않고 종료합니다.
		return;
	}
	// "K1"키로 데이터베이스에 연결을 종료합니다.
	_SFDBClose("K1");
}
```

2.7.2 _SFDBDelete()

함수원형

int _SFDBDelete(string strKey, string strTag, string strStart, string strEnd, string strFilter)

파라메터	strKey : 연결 KEY strTag : 삭제태그 strStart : 시작일자 strEnd : 종료일자 strFilter : 검색 조건
리턴형	정수형 0 : 실패 1 : 성공
설명	DB의 저장된 데이터를 조회하여 데이터를 삭제합니다.
함수활용	저장된 데이터에서 오래된 데이터 또는 특정 조건을 벗어난 데이터에 대해서 삭제를 할 수 있습니다.

※ 시작 일자, 종료일자 문자열은 타임포맷 "%Y-%m-%d %H:%M:%S"으로 설정합니다.
※ 삭제조건: 검색에 특별한 조건을 설정할 수 있습니다.
조건이 필요 없는 경우 ""로 설정합니다. 조건 설정시 해당 태그명은 $로 표기합니다.
태그의 값이 100보다 큰 경우 "$ > 100" 같이 설정합니다.

```
void SFDBDelete()
{
	int nRet, nCount;

	// 데이터베이스 SAMPLE_DB에 "K1"키로 연결을 하고 상태를 정수형 변수 nRet에 저장합니다.
	nRet = _SFDBOpen("K1", "SAMPLE_DB");

	// 연결에 실패하면
	if(nRet < 1)
	{
	// 메시지박스에 SFDB 에러 메시지를 메시지 박스에 표시합니다.
	_MsgBox(_SFDBMsg("K1"), "DB오픈에러", 0, 2);
		return;
	}
	// 태그명 DEMO.1의 2013-08-05 00:00:00 ~ 2013-08-05 01:00:00 사이의 데이터 중에서 100보다
	// 큰 값을 삭제합니다.
	// 쿼리결과 상태를 정수형 변수 nCount에 저장합니다.
```

```
	nCount = _SFDBDelete("K1", "DEMO.1",  "2013-08-05 00:00:00", "2013-08-05 01:00:00", "$ > 100");

	// 쿼리실패하면
	if(nCount < 1)
	{
	// 메시지박스에 SFDB 에러 메시지를 메시지 박스에 표시합니다.
	_MsgBox(_SFDBMsg("K1"), "삭제에러", 0, 2);
	}

	// "K1"키로 데이터베이스에 연결을 종료합니다.
	_SFDBClose("K1");
}
```

2.7.3 _SFDBGetValue()

함수원형

int _SFDBGetValue(string strKey, string strTag, string strStart, string strEnd, int nIndex, string strRetTag, string strFilter)

파라메터	strKey : 연결 KEY strTag : 검색태그 strStart : 시작일자 strEnd : 종료일자 nIndex : 데이터종류 인덱스번호 strRetTag : 검색결과 저장태그 strFilter : 검색 조건
리턴형	정수형 0 : 실패 1 : 성공
설명	DB에서 특정한 데이터 결과값을 태그에 저장합니다. 함수가 실행되고 결과를 얻을 때까지 기다립니다.
함수활용	DB에서 한개의 데이터(최대값/최소값/평균값/적산차 등)를 검색해야 하는 경우 사용합니다.

※ 검색 시작 일자, 검색 종료일자 문자열은 타임포맷 "%Y-%m-%d %H:%M:%S"으로 설정합니다.

※ 데이터종류 인덱스번호

0 기본 저장값
1 통계데이터 모두 . 시 간격
2 통계데이터 모두 . 일 간격
3 통계데이터 모두 . 월 간격
10 순간값 (정시값)
11 평균값
12 적산차
13 최소값
14 최대값
15 합계값
16 시작값 (해당시간 첫번째 값)
17 마지막 값 (해당시간 마지막 값)
18 레코드 개수 (주어진 시간내에 기록된 데이터 개수)
30 운전시간 1
31 운전시간 2
32 운전횟수
33 정지시간 1

34 정지시간 2
35 정지횟수
101 통계요청이 시간별 데이터 요청
102 통계요청이 일별 데이터 요청
103 통계요청이 우러별 데이터 요청
201 경보이력

※ 검색조건: 검색에 특별한 조건을 설정할 수 있습니다.
검색 조건이 필요 없는 경우 ""로 설정합니다. 검색조건 설정시 검색 태그명은 $로 표기합니다.
검색태그의 값이 100보다 큰 경우 "$ > 100" 같이 설정합니다.

※ _SFDBGetValue()는 동기방식으로 검색합니다. 동일한 함수 내에서 _SFDBGetValue() 호출 후
_SFDBClose()가 있는경우 검색이 끝날때까지 _SFDBClose를 호출되지 않습니다.
_SFDBGetValueEx()는 비동기방식으로 검색합니다. 동일한 함수 내에서 _SFDBGetValue() 호출 후
_SFDBClose()가 있는경우 검색이 끝나기전에 _SFDBClose를 호출할 수 있습니다.
동기방식은 쿼리 결과에 대한 응답이 올때까지 시스템이 대기를 하게 되고
비동기방식은 쿼리 결과에 대한 응답을 시스템이 기다리지 않고 다음처리를 수행합니다.
따라서 비동기방식인 경우 SFDBClose는 같은 함수에 있으면 쿼리결과가 정상적으로 처리되지 않을 수 있습니다. 비동기방식으로 처리할 경우에는 SFDBOpen 함수는 시스템 시작시 SFDBClose 함수는 시스템 종료시에 한번만 처리하여 사용해야 합니다.

```
/*********************************************************

예제 : DB에서 쿼리하여 리스트 컨트롤에 표시하기
*********************************************************/

void DBGetValue()
{
	int nRet, nCount;
	string strStart, strEnd;

	// 데이터베이스 SAMPLE_DB에 "K1"키로 연결을 하고 상태를 정수형 변수 nRet에 저장합니다.
	nRet = _SFDBOpen("K1", "SAMPLE_DB");

	// 연결에 실패하면
	if(nRet < 1)
	{
		// 메시지박스에 SFDB 에러 메시지를 메시지 박스에 표시합니다.
	_MsgBox(_SFDBMsg("K1"), "DB오픈에러", 0, 2);
		// 아래 함수구문은 실행되지 않고 종료합니다.
		return;
```

```
    }
// 태그명 DEMO.1의 2013-08-05 00:00:00 ~ 2013-08-05 01:00:00 사이의 데이터 중에서 100보다
    // 큰 값중에서 최소값(13)을 "DEMO.2" 태그에 저장합니다.
    // 실행결과 성공이면 1, 실패면 0을 nCount에 저장합니다.
    // 1일전
    strStart = _GetDateFormat("%Y-%m-%d %H:%M:%S", -1);
    // 현재시간
    strEnd = _GetDateFormat("%Y-%m-%d %H:%M:%S", 0);

    //
    // 최소값을 요청하고 결과를 기다립니다. (WAIT)
    // 검색이 끝나면 결과값을 태그에 DEMO.2에 저장합니다.
    nCount = _SFDBGetValue("K1", "DEMO.1",  strStart, strEnd, 13, "DEMO.2", "$ > 100");

    // 쿼리실패하면
    if(nCount < 1)
    {
        // 메시지박스에 SFDB 에러 메시지를 메시지 박스에 표시합니다.
    MsgBox(_SFDBMsg("K1"), "쿼리에러", 0, 2)
    }
    // "K1" 키연결을 종료합니다.
    _SFDBClose("K1");
}
```

2.7.4 _SFDBGetValueEx()

함수원형

int _SFDBGetValueEx(string strKey, string strTag, string strStart, string strEnd, int nIndex, string strRetTag, string strFilter)

파라메터	strKey : 연결 KEY strTag : 검색태그 strStart : 시작일자 strEnd : 종료일자 nIndex : 데이터종류 인덱스번호 strRetTag : 검색결과 저장태그 strFilter : 검색 조건
리턴형	정수형 0 : 실패 1 : 성공
설명	DB에서 특정한 데이터 결과값을 태그에 저장합니다. 함수가 실행되고 결과를 얻을 때까지 기다리지 않습니다.
함수활용	DB에서 한개의 데이터(최대값/최소값/평균값/적산차 등)를 검색해야 하는 경우 사용합니다.

※ 검색 시작 일자, 검색 종료일자 문자열은 타임포맷 "%Y-%m-%d %H:%M:%S"으로 설정합니다.
※ 데이터종류 인덱스번호
0 기본 저장값
1 통계데이터 모두 . 시 간격
2 통계데이터 모두 . 일 간격
3 통계데이터 모두 . 월 간격
10 순간값 (정시값)
11 평균값
12 적산차
13 최소값
14 최대값
15 합계값
16 시작값 (해당시간 첫번째 값)
17 마지막 값 (해당시간 마지막 값)
18 레코드 개수 (주어진 시간내에 기록된 데이터 개수)
30 운전시간 1
31 운전시간 2
32 운전횟수
33 정지시간 1

34 정지시간 2
35 정지횟수
101 통계요청이 시간별 데이터 요청
102 통계요청이 일별 데이터 요청
103 통계요청이 우러별 데이터 요청
201 경보이력

※ 검색조건: 검색에 특별한 조건을 설정할 수 있습니다.
검색 조건이 필요 없는 경우 ""로 설정합니다. 검색조건 설정시 검색 태그명은 $로 표기합니다.
검색태그의 값이 100보다 큰 경우 "$ > 100" 같이 설정합니다.

※ _SFDBGetValue()는 동기방식으로 검색합니다. 동일한 함수 내에서 _SFDBGetValue() 호출 후
_SFDBClose()가 있는경우 검색이 끝날때까지 _SFDBClose를 호출되지 않습니다.
_SFDBGetValueEx()는 비동기방식으로 검색합니다. 동일한 함수 내에서 _SFDBGetValue() 호출 후
_SFDBClose()가 있는경우 검색이 끝나기전에 _SFDBClose를 호출할 수 있습니다.
동기방식은 쿼리 결과에 대한 응답이 올때까지 시스템이 대기를 하게 되고
비동기방식은 쿼리 결과에 대한 응답을 시스템이 기다리지 않고 다음처리를 수행합니다.
따라서 비동기방식인 경우 SFDBClose는 같은 함수에 있으면 쿼리결과가 정상적으로 처리되지 않을 수 있습니다. 비동기방식으로 처리할 경우에는 SFDBOpen 함수는 시스템 시작시 SFDBClose 함수는 시스템 종료시에 한번만 처리하여 사용해야 합니다.

```
/*********************************************************

예제 : DB에서 쿼리하여 리스트 컨트롤에 표시하기
*********************************************************/

void DBGetValue()
{
	int nRet, nCount;
	string strStart, strEnd;

	// 데이터베이스 SAMPLE_DB에 "K1"키로 연결을 하고 상태를 정수형 변수 nRet에 저장합니다.
	nRet = _SFDBOpen("K1", "SAMPLE_DB");

	// 연결에 실패하면
	if(nRet < 1)
	{
		// 메시지박스에 SFDB 에러 메시지를 메시지 박스에 표시합니다.
	_MsgBox(_SFDBMsg("K1"), "DB오픈에러", 0, 2);
		// 아래 함수구문은 실행되지 않고 종료합니다.
		return;
```

```
    }
// 태그명 DEMO.1의 2013-08-05 00:00:00 ~ 2013-08-05 01:00:00 사이의 데이터 중에서 100보다
    // 큰 값중에서 최소값(13)을 "DEMO.2" 태그에 저장합니다.
    // 실행결과 성공이면 1, 실패면 0을 nCount에 저장합니다.
    // 1일전
    strStart = _GetDateFormat("%Y-%m-%d %H:%M:%S", -1);
    // 현재시간
    strEnd = _GetDateFormat("%Y-%m-%d %H:%M:%S", 0);

    // 최소값을 비동기적으로 요청하고 빠져나옵니다. (NO WAIT)
    // 검색이 끝나면 결과값을 태그에 DEMO.2에 저장합니다.
    nCount = _SFDBGetValueEx("K1", "DEMO.1", strStart, strEnd, 13, "DEMO.2", "$ > 100");

    // 쿼리실패하면
    if(nCount < 1)
    {
        // 메시지박스에 SFDB 에러 메시지를 메시지 박스에 표시합니다.
    MsgBox(_SFDBMsg("K1"), "쿼리에러", 0, 2)
    }
    // "K1" 키연결을 종료합니다.
    _SFDBClose("K1");
}
```

2.7.5 _SFDBInsert()

함수원형

int _SFDBInsert(string strKey, string strTag, double fValue)

파라메터	strKey : 연결 KEY strTag : 저장 태그 fValue : 저장값
리턴형	정수형 0 : 실패 1 : 성공
설명	DB에서 수동으로 특정값을 저장합니다.
함수활용	수동으로 태그값을 DB에 저장합니다.

```
/*********************************************************
예제 : DB에 수동으로 데이터 저장하기
*********************************************************/
void DBInsert()
{
	int nRet;

	// 데이터베이스 SAMPLE_DB에 "K1"키로 연결을 하고 상태를 정수형 변수 nRet에 저장합니다.
	nRet = _SFDBOpen("K1", "SAMPLE_DB");

	// 연결에 실패하면
	if(nRet < 1)
	{
		// 메시지박스에 SFDB 에러 메시지를 메시지 박스에 표시합니다.
	_MsgBox(_SFDBMsg("K1"), "DB오픈에러", 0, 2);
		return;
	}
	// "DEMO.1" 태그키에 100을 DB에 저장합니다.
	nRet = _SFDBInsert("K1", "DEMO.1", 100);

	// 실패하면
	if(nRet < 1)
	{
	    // 메시지박스에 SFDB 에러 메시지를 메시지 박스에 표시합니다.
	MsgBox(_SFDBMsg("K1"), "에러", 0, 2)
```

```
        }
        // "K1" 키연결을 종료합니다.
        _SFDBClose("K1");
}
```

2.7.6 _SFDBMsg()

함수원형

string _SFDBMsg(string strKey)

파라메터	strKey : DB연결 키
리턴형	문자열
설명	SFDB의 데이터베이스의 명령을 실행한 후 데이터베이스로 부터 받은 결과메시지를 돌려줍니다.
함수활용	SFDB에 접속 또는 쿼리문 실행 결과에 따른 상태메시지를 메시지박스 또는 트레이스로 확인하여 올바르게 수행이 되는지 확인할 수 있습니다.

```
/*********************************************************
예제 : SFDB의 실행결과
*********************************************************/
void SFDBMsg()
{
	int nRet;
	// 데이터베이스 SAMPLE_DB에 "K1"키로 연결을 하고 상태를 정수형 변수 nRet에 저장합니다.
	nRet = _SFDBOpen("K1", "SAMPLE_DB");
	// 연결에 실패하면
	if(nRet < 1)
	{
		// 메시지박스에 SFDB 에러 메시지를 메시지 박스에 표시합니다.
		_MsgBox(_SFDBMsg("K1"), "에러", 0, 2);
		return;
	 }
	// "K1"키로 데이터베이스에 연결을 종료합니다.
	_SFDBClose("K1");
}
```

2.7.7 _SFDBOpen()

함수원형

int _SFDBOpen(string strKey, string strDatabase)

파라메터	strKey : SFDB를 직접 연결하기 위한 KEY strDatabase : 데이터베이스 리스트에 등록된 DB이름.
리턴형	정수형 0 : 실패 1 : 성공 2 : 이미 연결되어 있는 상태
설명	SFDB의 데이터베이스에 연결합니다.
함수활용	SFDB에 저장된 데이터를 조회 및 삭제 작업시 특정 데이터베이스에 연결을 합니다.

※ _SFDBOpen함수와 _SFDBClose 함수는 항상 같이 사용하시기 바랍니다.

```
/*********************************************************
예제 : SFDB 데이터베이스 연결
*********************************************************/
void SFDBOpen ()
{
	int nRet;
	// 데이터베이스 SAMPLE_DB에 "K1"키로 연결을 하고 상태를 정수형 변수 nRet에 저장합니다.
	nRet = _SFDBOpen("K1", "SAMPLE_DB");
	// 연결에 실패하면
	if(nRet < 1)
	{
		// 메시지박스에 SFDB 상태 메시지를 메시지 박스에 표시합니다.
		_MsgBox(_SFDBMsg ("K1"), "DB오픈에러", 0, 2);
		// 아래 함수구문은 실행되지 않고 종료합니다.
		return;
	}
	// "K1"키로 데이터베이스에 연결을 종료합니다.
	_SFDBClose("K1");
}
```

2.7.8 _SFDBQuery()

함수원형

int _SFDBQuery(string strKey, string strObject, string strTag, int nInterval, string strStart, string strEnd, int nIndex, string strFilter)

파라메터	strKey : 연결 KEY strObject : 쿼리결과를 담을 오브젝트 이름 strTag : 검색태그 nInterval : 검색간격 (초단위) strStart : 시작일자 strEnd : 종료일자 nIndex : 데이터종류 인덱스번호 strFilter : 검색 조건
리턴형	정수형 0 : 실패 1 : 성공
설명	DB에서 특정한 데이터 결과값을 태그에 저장합니다. 함수가 실행되고 결과를 얻을 때까지 기다립니다.
함수활용	DB에서 한개의 데이터(최대값/최소값/평균값/적산차 등)를 검색해야 하는 경우 사용합니다.

※ 검색 시작 일자, 검색 종료일자 문자열은 타임포맷 "%Y-%m-%d %H:%M:%S"으로 설정합니다.

※ 데이터종류 인덱스번호

0 기본 저장값
1 통계데이터 모두 . 시 간격
2 통계데이터 모두 . 일 간격
3 통계데이터 모두 . 월 간격
10 순간값 (정시값)
11 평균값
12 적산차
13 최소값
14 최대값
15 합계값
16 시작값 (해당시간 첫번째 값)
17 마지막 값 (해당시간 마지막 값)
18 레코드 개수 (주어진 시간내에 기록된 데이터 개수)
30 운전시간 1
31 운전시간 2
32 운전횟수

33 정지시간 1
34 정지시간 2
35 정지횟수
101 통계요청이 시간별 데이터 요청
102 통계요청이 일별 데이터 요청
103 통계요청이 우러별 데이터 요청
201 경보이력

※ 검색조건: 검색에 특별한 조건을 설정할 수 있습니다.
검색 조건이 필요 없는 경우 ""로 설정합니다. 검색조건 설정시 검색 태그명은 $로 표기합니다.
검색태그의 값이 100보다 큰 경우 "$ > 100" 같이 설정합니다.

※ _SFDBQuery()는 동기방식으로 검색합니다. 동일한 함수 내에서 _SFDBQuery() 호출 후
_SFDBClose()가 있는경우 검색이 끝날때까지 _SFDBClose를 호출되지 않습니다.
_SFDBQueryEx()는 비동기방식으로 검색합니다. 동일한 함수 내에서 _SFDBQueryEx() 호출 후
_SFDBClose()가 있는경우 검색이 끝나기전에 _SFDBClose를 호출할 수 있습니다.
동기방식은 쿼리 결과에 대한 응답이 올때까지 시스템이 대기를 하게 되고
비동기방식은 쿼리 결과에 대한 응답을 시스템이 기다리지 않고 다음처리를 수행합니다.
따라서 비동기방식인 경우 SFDBClose는 같은 함수에 있으면 쿼리결과가 정상적으로 처리되지 않을 수 있습니다. 비동기방식으로 처리할 경우에는 SFDBOpen 함수는 시스템 시작시 SFDBClose 함수는 시스템 종료시에 한번만 처리하여 사용해야 합니다.

```
/*************************************************************
예제 : SFDB에서 쿼리하여 리스트 컨트롤에 표시하기
*************************************************************/
void DBQuery()
{
	int nRet, nCount;
	string strStart, strEnd;

	// 데이터베이스 SAMPLE_DB에 "K1"키로 연결을 하고 상태를 정수형 변수 nRet에 저장합니다.
	nRet = _SFDBOpen("K1", "SAMPLE_DB");
	// 연결에 실패하면
	if(nRet < 1)
	{
		// 메시지박스에 SFDB 에러 메시지를 메시지 박스에 표시합니다.
		_MsgBox(_SFDBMsg("K1"), "DB오픈에러", 0, 2);
		// 아래 함수구문은 실행되지 않고 종료합니다.
		return;
	}
```

```
	// 태그명 DEMO.1을 2013-08-05 00:00:00 ~ 2013-08-05 01:00:00 사이의 데이터 중에서 100보다
	// 큰 값을 조회하여 오브젝트명이 "리스트인" 윈도우 오브젝트에 결과를 표시합니다.
	// 쿼리결과 상태를 정수형 변수 nCount에 저장합니다.
	// 1일전
	strStart = _GetDateFormat("%Y-%m-%d %H:%M:%S", -1);
	// 현재시간
	strEnd = _GetDateFormat("%Y-%m-%d %H:%M:%S", 0);

	nCount = _SFDBQuery("K1", "리스트@뷰페이지명", "DEMO.1", 0, strStart, strEnd, 0, "$ > 100");
	// 쿼리실패하면
	if(nCount < 1)
	{
		// 메시지박스에 SFDB 에러 메시지를 메시지 박스에 표시합니다.
		_MsgBox(_SFDBMsg("K1"), "쿼리에러", 0, 2);
	 }
	// "K1"키로 데이터베이스에 연결을 종료합니다.
	_SFDBClose("K1");
}
```

2.7.9 _SFDBQueryEx()

함수원형

int _SFDBQueryEx(string strKey, string strObject, string strTag, int nInterval, string strStart, string strEnd, int nIndex, string strRetTag, string strFilter)

파라메터	strKey : 연결 KEY strObject : 쿼리결과를 담을 오브젝트 이름 strTag : 검색태그 nInterval : 검색간격 (초단위) strStart : 시작일자 strEnd : 종료일자 nIndex : 데이터종류 인덱스번호 strRetTag : 검색결과 저장태그 strFilter : 검색 조건
리턴형	정수형 0 : 실패 1 : 성공
설명	DB에서 데이터를 검색해서 오브젝트에 표시합니다. 함수가 실행되고 결과를 얻을 때까지 기다립니다.
함수활용	SFDB에 저장된 RAWDATA를 특정 윈도우 오브젝트(그리드, 리스트컨트롤, 콤보박스, 리스트박스, 입력창)를 통해서 데이터를 확인할 수 있습니다.

※ 검색 시작 일자, 검색 종료일자 문자열은 타임포맷 "%Y-%m-%d %H:%M:%S"으로 설정합니다.

※ 데이터종류 인덱스번호

0 기본 저장값

1 통계데이터 모두 . 시 간격

2 통계데이터 모두 . 일 간격

3 통계데이터 모두 . 월 간격

10 순간값 (정시값)

11 평균값

12 적산차

13 최소값

14 최대값

15 합계값

16 시작값 (해당시간 첫번째 값)

17 마지막 값 (해당시간 마지막 값)

18 레코드 개수 (주어진 시간내에 기록된 데이터 개수)

30 운전시간 1

31 운전시간 2
32 운전횟수
33 정지시간 1
34 정지시간 2
35 정지횟수
101 통계요청이 시간별 데이터 요청
102 통계요청이 일별 데이터 요청
103 통계요청이 우러별 데이터 요청
201 경보이력

※ 검색조건: 검색에 특별한 조건을 설정할 수 있습니다.
검색 조건이 필요 없는 경우 ""로 설정합니다. 검색조건 설정시 검색 태그명은 $로 표기합니다.
검색태그의 값이 100보다 큰 경우 "$ > 100" 같이 설정합니다.

※ _SFDBQuery()는 동기방식으로 검색합니다. 동일한 함수 내에서 _SFDBQuery() 호출 후
_SFDBClose()가 있는경우 검색이 끝날때까지 _SFDBClose를 호출되지 않습니다.
_SFDBQueryEx()는 비동기방식으로 검색합니다. 동일한 함수 내에서 _SFDBQueryEx() 호출 후
_SFDBClose()가 있는경우 검색이 끝나기전에 _SFDBClose를 호출할 수 있습니다.
동기방식은 쿼리 결과에 대한 응답이 올때까지 시스템이 대기를 하게 되고
비동기방식은 쿼리 결과에 대한 응답을 시스템이 기다리지 않고 다음처리를 수행합니다.
따라서 비동기방식인 경우 SFDBClose는 같은 함수에 있으면 쿼리결과가 정상적으로 처리되지 않을 수 있습니다. 비동기방식으로 처리할 경우에는 SFDBOpen 함수는 시스템 시작시 SFDBClose 함수는 시스템 종료시에 한번만 처리하여 사용해야 합니다.

```
/*********************************************************
예제 : SFDB에서 쿼리하여 리스트 컨트롤에 표시하기
*********************************************************/
void DBQuery()
{
	int nRet, nCount;
	string strStart, strEnd;

	// 데이터베이스 SAMPLE_DB에 "K1"키로 연결을 하고 상태를 정수형 변수 nRet에 저장합니다.
	nRet = _SFDBOpen("K1", "SAMPLE_DB");
	// 연결에 실패하면
	if(nRet < 1)
	{
		// 메시지박스에 SFDB 에러 메시지를 메시지 박스에 표시합니다.
		_MsgBox(_SFDBMsg("K1"), "DB오픈에러", 0, 2);
		// 아래 함수구문은 실행되지 않고 종료합니다.
```

```
            return;
        }
        // 태그명 DEMO.1을 2013-08-05 00:00:00 ~ 2013-08-05 01:00:00 사이의 데이터 중에서 100보다
        // 큰 값을 조회하여 오브젝트명이 "리스트인" 윈도우 오브젝트에 결과를 표시합니다.
        // 쿼리결과 상태를 정수형 변수 nCount에 저장합니다.
        // 1일전
        strStart = _GetDateFormat("%Y-%m-%d %H:%M:%S", -1);
        // 현재시간
        strEnd = _GetDateFormat("%Y-%m-%d %H:%M:%S", 0);

        nCount = _SFDBQuery("K1", "리스트@뷰페이지명", "DEMO.1", 0, strStart, strEnd, 0, "$ > 100");
        // 쿼리실패하면
        if(nCount < 1)
        {
            // 메시지박스에 SFDB 에러 메시지를 메시지 박스에 표시합니다.
            _MsgBox(_SFDBMsg("K1"), "쿼리에러", 0, 2);
        }
        // "K1"키로 데이터베이스에 연결을 종료합니다.
        _SFDBClose("K1");
}
```

2.7.10 _SFDBRecovery()

함수원형

int _SFDBRecovery(string strKey, strTag, string strStart, strEnd, strFilter)

파라메터	strKey : DB연결 키 strTag : 복구태그 strStart : 복구 시작일자 strEnd : 복구 종료일자 strFilter : 복구조건
리턴형	문자열
설명	_SFDBDelete 내부함수에 의해 삭제된 데이터를 복구합니다.
함수활용	사용자에 의해 실수로 삭제된 데이터를 복구할 수 있습니다.

※ 복구조건: 복구조건을 설정할 수 있습니다.
복구조건이 필요 없는 경우 ""로 설정합니다. 복구 태그명은 $로 표기합니다.
예를 들어서, 복구하려고 하는 태그의 값이 100보다 큰 경우 "$ > 100" 같이 설정합니다.

```
/*********************************************************
예제 : SFDB에서 삭제된 데이터 복구하기
*********************************************************/
void SFDBRecovery()
{
	int nRet, nCount;
	string strStart, strEnd;

	nRet = _SFDBOpen("K1", "SAMPLE_DB");
	if(nRet < 1)
	{
	_MsgBox(_SFDBMsg("K1"), "DB오픈에러", 0, 2);
	return;
	}
	// 2013-08-05 00:00:00 ~ 2013-08-05 01:00:00 사이의 삭제된 데이터 중에서 100보다 큰값 복구.
	strStart = "2013-08-05 00:00:00";
	strEnd = "2013-08-05 01:00:00";
	nCount = _SFDBRecovery("K1", "DEMO.1", strStart, strEnd, "$ > 100");

	if(nCount < 1)
	{
```

```
        _MsgBox(_SFDBMsg("K1"), "복구에러", 0, 2);
        }
        _SFDBClose("K1");
}
```

2.7.11 _SFDBUpdate()

함수원형

int _SFDBUpdate(string strKey, strTag, string strStart, strEnd, double fRate, double fValue, string strFilter)

파라메터	strKey : DB연결 키 strTag : 업데이트 태그 strStart : 시작일자 strEnd : 종료일자 fRate : 업데이트 형태 (0(절대값), 1(백분율값)->검색값 * (value / 100.0) 으로 지정) fValue : 변경값 또는 변경비율값. strFilter : 업데이트 조건
리턴형	정수값 0 : 실패 1 : 성공
설명	DB에 저장된 태그값을 변경합니다.
함수활용	특정 태그에 대하여 데이터를 변경하는데, 절대값(사용자 지정값)으로 변경할 수 있고, 기록된 값의 비례값으로 변경할 수 있습니다.

```
void SFDBUpdate()
{
	int nRet, nCount;

	nRet = _SFDBOpen("K1", "SAMPLE_DB");
	if(nRet < 1)
	{
	_MsgBox(_SFDBMsg("K1"), "DB오픈에러", 0, 2);
	return;
	}
	// 2013-08-05 00:00:00 ~ 2013-08-05 01:00:00 사이의 데이터 중에서 100보다
	// 큰 값을 90으로 변경합니다.
	nCount = _SFDBUpdate("K1", "DEMO.1", 13, "2013-08-05 00:00:00", "2013-08-05 01:00:00",  90, "$ > 100");
	if(nCount < 1)
	{
	_MsgBox(_SFDBMsg("K1"), "갱신에러", 0, 2);
	}
	_SFDBClose("K1");
}
```

2.8 레지스트리 함수

2.8.1 _RegGetValue()

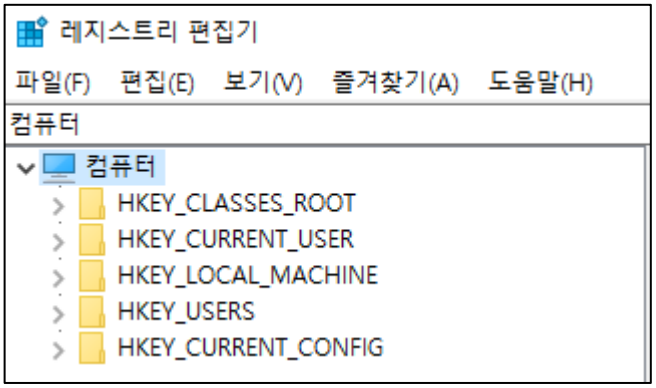
함수원형

int _RegGetValue(string strKey, string strSubKey, string strEntry)

string _RegGetValue(string strKey, string strSubKey, string strEntry)

파라메터	strKey : 윈도우 레지스트리에 등록 되어있는 레지스트리 키를 지정합니다. strSubKey : 레지스트리 키 내에 있는 서브 키를 지정합니다. strEntry : 서브 키에 등록되어 있는 항목에 대한 이름을 지정합니다. ※
리턴형	정수값, 문자열
설명	레지스트리에 등록되어 있는 값을 읽어옵니다.
함수활용	설정값 등을 레지스트리를 통해서 공유 할 필요가 있을 경우 사용합니다.

※ 레지시트리 키 종류는 아래와 같습니다.




```
/*********************************************************
예제 : 레지스트리 값 읽기
*********************************************************/
void Registry()
{
	int nAge;
	string strName;

	_RegSetValue("HKEY_CURRENT_USER", "Software₩₩SEFA Technology₩₩eRun₩₩Path", "age", 100);
	nAge = _RegGetValue("HKEY_CURRENT_USER", "Software₩₩SEFA Technology₩₩eRun₩₩Path",
"age");

	// 정수형 변수 nAge값을 메시지박스에 표시합니다.
```

```
    _MsgBox(nAge, "레지스트리값 읽기", 0, 0);

    /*********************************************************************************************
    결과를 확인하기 위한 방법은 아래와 같습니다.
    1. Regedit(레지스트리 편집기) 프로그램을 실행시킵니다.
    2. Software₩₩SEFA Technology₩₩eRun₩₩Path 키를 새로 만듭니다.
    3. 새로만들어진 키에 DWORD 값을 새로 만듭니다.
    4. eRun을 실행한다 [F5]
    5. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
    6. 변경된 레지시트리를 확인합니다.
    *********************************************************************************************/
}
```

2.8.2 _RegSetValue()

함수원형

void _RegSetValue(string strKey, string strSubKey, string strEntry, int nValue)

void _RegSetValue(string strKey, string strSubKey, string strEntry, double fValue)

void _RegSetValue(string strKey, string strSubKey, string strEntry, string strText)

파라메터	strKey : 윈도우 레지스트리에 등록 되어있는 레지스트리 키를 지정합니다. strSubKey : 레지스트리 키 내에 있는 서브 키를 지정합니다. strEntry : 서브 키에 등록되어 있는 항목에 대한 이름을 지정합니다. nValue : 등록하려는 정수형 데이터, 실수형, 문자열 데이터 ※
리턴형	없음
설명	레지스트리에 정수갑, 실수값, 문자열 값을 등록합니다. 레지스트리에 미리생성된 키와, 값을 설정해야 합니다.
함수활용	초기값 및 설정값, 외부 프로그램과의 데이터 공유시 레지스트리를 활용할 수 있습니다.

* 레지스트리 키는 아래와 같이 5종류가 있습니다.
"HKEY_CLASSES_ROOT", "HKEY_CURRENT_USER", "HKEY_LOCAL_MACHINE","HKEY_USERS",
"HKEY_CURRENT_CONFIG"

```
/*******************************************************
예제 : 레지스트리 값 쓰기
*******************************************************/
void Registry()
{
	int nAge;
	string strName;

	_RegSetValue("HKEY_CURRENT_USER", "Software₩₩SEFA Technology₩₩eRun₩₩Path", "age", 100);
	nAge = _RegGetValue("HKEY_CURRENT_USER", "Software₩₩SEFA Technology₩₩eRun₩₩Path",
"age");

	// 정수형 변수 nAge값을 메시지박스에 표시합니다.
	_MsgBox(nAge, "레지스트리값 읽기", 0, 0);

	/*************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. Regedit(레지스트리 편집기) 프로그램을 실행시킵니다.
```

2. Software₩₩SEFA Technology₩₩eRun₩₩Path 키를 새로 만듭니다.
3. 새로만들어진 키에 DWORD 값을 새로 만듭니다.
4. eRun을 실행한다 [F5]
5. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
6. 변경된 레지시트리를 확인합니다.

**********/

}

2.9 보고서 함수

2.9.1 _RepGetValue()

함수원형

void _RepGetValue(string strReport, int nSheet, string strCell, string strTag, string strTime, int nIndex, string strFilter)

파라메터	strReport : 보고서 이름 nSheet : 엑셀의 시트번호 strCell : 셀명 strTag : 검색태그 strTime : 지정시간. nIndex : 검색종류. strFilter : 검색필터 표현식.
리턴형	없음
설명	지정된 태그를 데이터베이스에서 지정된 시간과 값 종류에 따라 값을 읽어와서 지정한 보고서 시트의 셀에 태그 값을 출력합니다.
함수활용	

※ 이 함수는 엑셀 보고서설정에서만 사용할 수 있는 내부함수 입니다.

- ❖ 보고서모델명 (문자열) : 프로젝트에 등록되어 있는 보고서 모델이름을 문자열 형태로 지정합니다.
- ❖ 시트번호 (정수) : 엑셀시트번호를 0번부터 지정합니다. 보고서 모델에 시트가 등록되어 있어야 합니다.
- ❖ 셀명 (문자열) : 해당 시트의 특정 셀 번호를 입력합니다. 예를 들어서 "C10" 같은 형식으로 셀의 번호를 입력합니다. 동일한 시트에 동일한 셀을 두번 사용하면 두 번째 셀은 무시됩니다.
- ❖ 태그명 (문자열) : 검색하고자 하는 태그 이름을 입력합니다. (예 - "TagGroup1.POWER")
- ❖ 지정시간 (문자열) : 필요한 순간의 데이터를 검색하기 위해서 입력합니다.

1) 일간보고서 출력에 필요한 형태

형식은 : DAY + 'D' + ',' + TIME + 'H'

기준시간일자를 정해서 특정시간 데이터를 가져옵니다. 몇일, 몇시 데이터를 가져오는지에 대한 의미입니다.

기준시간이 설정되어있지 않는 경우는 오늘 날짜를 기준시간으로 사용합니다.

DAY : 기준시간보다 며칠 전(-) 또는 며칠 후(+)의 정수를 입력합니다. 0을 입력하면 기준시간을 사용합니다.

TIME : 몇 시 데이터를 의미하며 1시(00:00:00~00:59:59) 데이터부터 24시(23:00:00~23:59:59) 데이터를 사용합니다.

(사용 예)

기준시간일자 : 2009-1-9 09:00:00 설정되어있을 경우.

0D,1H : 2009-1-9 00:00:00 ~ 2009-1-8 00:59:59
0D,24H : 2009-1-9 23:00:00 ~ 2009-1-8 23:59:59
-1D,1H : 2009-1-8 00:00:00 ~ 2009-1-8 00:59:59
-1D,24H : 2009-1-8 23:00:00 ~ 2009-1-8 23:59:59
1D,1H : 2009-1-10 00:00:00 ~ 2009-1-10 00:59:59
1D,24H : 2009-1-10 23:00:00 ~ 2009-1-10 23:59:59

2) 주간보고서 출력에 필요한 형태.

형식은 : WEEK + 'W' + ',' + DATE
기준시간일자를 정해서 특정요일의 종일 데이터를 가져옵니다. 몇째 주 몇 요일 데이터를 가져오는지에 대한 의미입니다.
기준시간이 설정되어있지 않는 경우는 오늘 날짜를 기준시간으로 사용합니다.
WEEK : 기준일자시간보다 몇 주 전(-) 또는 몇 주 후(+)의 정수를 입력한다. 0을 입력하면 기준시간을 사용합니다.
DATE : 주간 요일을 정해진 문자열로 입력합니다. (SUN, MON, TUE, WED, THU, FRI, SAT)

(사용 예)
기준시간일자 : 2013-4-29 09:00:00 설정되어있을 경우.
0W,SUN : 기준시간일자가 속해있는 주의 앞선 일요일의 종일 데이터
(2013-4-29 00:00:00 ~ 2013-4-29 23:59:59)
-1W,SUN : 일주일 전 일요일 종일 데이터
(2013-4-21 00:00:00 ~ 2013-4-21 23:59:59)
1W,SUN : 일주일 후 일요일 종일 데이터
(2013-5-5 00:00:00 ~ 2013-5-5 23:59:59)

3) 월간보고서 출력에 필요한 형태.

형식은 : MONTH + 'M' + ',' + DAY + 'D'
기준시간일자를 정해서 특정 월의 특정일의 종일 데이터를 가져옵니다. 몇 월 며칠 데이터를 가져오는지에 대한 의미입니다.
기준시간이 설정되어있지 않는 경우는 오늘 날짜를 기준시간으로 사용합니다.
MONTH : 기준일자시간보다 몇 개월 전(-) 또는 몇 개월 후(+)의 정수를 입력합니다. 0을 입력하면 기준월을 사용합니다.
DAY : 지정일자를 정수로 입력합니다. (1 ~31)

(사용 예)
기준시간일자 : 2009-1-9 09:00:00 설정되어있을 경우.
0M,1D : 2009-1-1 00:00:00 ~ 2009-1-1 23:59:59 (당월 1일 데이터)
0M,31D : 2009-1-31 00:00:00 ~ 2009-1-31 23:59:59 (당월 31일 데이터)
-1M,1D : 2008-12-1 00:00:00 ~ 2008-12-1 23:59:59 (전월 1일 데이터)

-1M,31D : 2008-12-31 00:00:00 ~ 2008-12-31 23:59:59 (전월 31일 데이터)
1M,1D : 2009-2-1 00:00:00 ~ 2009-2-1 23:59:59 (익월 1일 데이터)
1M,31H : 2009-2-28 0:00:00 ~ 2009-2-28 23:59:59 (익월 31일 데이터)

4) 연간보고서 출력에 필요한 형태.

형식은 : YEAR + 'Y' + ',' + MONTH + 'M'
기준시간일자를 정해서 특정 년도의 특정월의 1개월 데이터를 가져옵니다. 몇년 몇월 데이터를 가져오는 것을 의미합니다.
기준시간이 설정되어있지 않는 경우는 오늘 날짜를 기준시간으로 사용합니다.
YEAR : 기준일자시간보다 몇 년 전(-) 또는 몇 년 후(+)의 정수를 입력합니다. 0을 입력하면 기준년도를 사용합니다.
MONTH : 지정 월을 정수로 입력합니다. (1 ~12)

(사용 예)
기준시간일자 : 2009-1-9 09:00:00 설정되어있을 경우.
0Y,1M : 2009-1-1 00:00:00 ~ 2009-1-31 23:59:59 (당해 1월 데이터)
0Y,12M : 2009-12-1 00:00:00 ~ 2009-12-31 23:59:59 (당해 12월 데이터)
-1Y,1M : 2008-1-1 00:00:00 ~ 2008-1-31 23:59:59 (전년 1월 데이터)
-1Y,12M : 2008-12-1 00:00:00 ~ 2008-12-31 23:59:59 (전년 12월 데이터)
1Y,1M : 2010-1-1 00:00:00 ~ 2010-1-31 23:59:59 (익년 1월 데이터)
1Y,12M : 2010-12-1 0:00:00 ~ 2010-12-31 23:59:59 (익년 12월 데이터)

❖ 값 종류(정수 값) : 검색 종류를 (정시 값, 평균값, 최소값, 최대값 등) 입력합니다. 값 종류에 대한 인덱스 번호와 매크로는 아래와 같습니다.

일반적으로 내부스크립트 함수에서는 인덱스 번호를 사용합니다.

매크로	번호	설명
raw_value	0	기본 저장 값
stat_tvalue	1	통계데이터 모두 . 시 간격
stat_dvalue	2	통계데이터 모두 . 일 간격
stat_mvalue	3	통계데이터 모두 . 월 간격
flow_value	10	순간 값 (정시 값)
avg_value	11	평균값
differ_value	12	적산차
min_value	13	최소값
max_value	14	최대값
sum_value	15	합계 값
start_value	16	시작 값 (해당시간 첫 번째 값)

last_value	17	마지막 값 (해당시간 마지막 값)
record_cnt	18	레코드 개수 (주어진 시간 내에 기록된 데이터 개수)
run_time1	30	운전시간 1
run_time2	31	운전시간 2
run_count	32	운전횟수
stop_time1	33	정지시간 1
stop_time2	34	정지시간 2
stop_count	35	정지횟수
cat_daily	101	통계요청이 시간 별 데이터 요청
cat_monthly	102	통계요청이 일별 데이터 요청
cat_yearly	103	통계요청이 월별 데이터 요청
alarm_data	201	경보이력

❖ 검색필터 표현식 : 검색값에 대한 조건식을 넣어줍니다. 예를 들어서 검색값이 특정값이거나 범위를 벗어나는 값 검사.
"$ > 100" -> 검색값이 100보다 큰 경우만 결과값으로 돌려줍니다. ($는 검색된 값)

```
/*********************************************************
예제 : 하루 전 정시 평균값 구하기
*********************************************************/
void                                                    RepGetValue
{
        // "Demo.Test1" 태그의 하루 전 0시00분 ~ 0시59분까지의 데이터 중에서 100보다 큰 값의 평균값을
        //                 일일보고서                 시트의                 A2셀에                 저장합니다.
        _RepGetValue("보고서", "일일보고서", "A2", "Demo.Test1", "-1D0H", 11, "$>100");
}
```

2.9.2 _RepSetValue()

함수원형

void _RepSetValue(string strReport, int nSheet, string strCell, int nValue)

void _RepSetValue(string strReport, int nSheet, string strCell, double fValue)

void _RepSetValue(string strReport, int nSheet, string strCell, string strValue)

파라메터	strReport : 보고서 이름 nSheet : 엑셀의 시트번호 strCell : 셀명 nValue : 정수값 fValue : 실수값 strValue : 문자열
리턴형	없음
설명	엑셀의 지정된 셀에 정수값, 실수값, 문자열등의 데이터를 출력합니다.
함수활용	

※ 이 함수는 엑셀 보고서설정에서만 사용할 수 있는 내부함수 입니다.

보고서모델명 (문자열) : 프로젝트에 등록되어 있는 보고서 모델이름을 문자열 형태로 지정합니다.
시트번호 (정수) : 엑셀시트번호를 0번부터 지정합니다. 보고서 모델에 시트가 등록되어 있어야 합니다.
셀명 (문자열) : 해당 시트의 특정 셀 번호를 입력합니다. 예를 들어서 "C10" 같은 형식으로 셀의 번호를 입력합니다.
동일한 시트에 동일한 셀을 두번 사용하면 두 번째 셀은 무시됩니다. 주의하시기 바랍니다.
출력 값(정수, 실수, 문자열) : 출력하고자 하는 값을 입력합니다. 입력 값은 일반숫자, 문자열, 변수 등을 지정할 수 있습니다.

```
/*********************************************************
예제 : 엑셀의 특정 셀에 데이터 쓰기
*********************************************************/
void RepSetValue
{
	// 일일보고서 첫 번째 시트의 셀 번호 C10에 숫자 123을 저장합니다.
	_RepSetValue("일일보고서", 0, "C10", 123);
	// 일일보고서 첫 번째 시트의 셀 번호  C11에 문자열 "SEFA Builder"를 저장합니다.
	_RepSetValue("일일보고서", 0, "C11", "SEFA Builder");
	// 일일보고서 첫 번째 시트의 셀 번호  C122에 @TagGroup1.POWER의 값을 저장합니다.
	_RepSetValue("일일보고서", 0, "C12", @TagGroup1.POWER);
}
```

2.10 날짜/시간 함수

2.10.1 _DateToTick()

함수원형

int _DateToTick(string strFormat, int nFlag)

파라메터	strFormat : 변환하고자 하는 날짜의 형식을 문자열 형태로 지정합니다 nFlag : 변환플래그
리턴형	변환된 정수형
설명	시간포맷을 정수 형으로 변환해서 돌려준다.
함수활용	시간차 연산 시 사용합니다. 특정일자 간의 시간을 연산할 때 사용합니다. 단위는 초(sec)단위입니다.

■ 시간형식

※ 대문자는 연도 모두 표시 [ex) 2013]하며 , 소문자는 연도 뒤에 두자리 표시[ex)13]합니다.

※ 시간 포맷 설정시 대소문자 주의하시기 바랍니다.

%y-%m-%d %H:%M:%S 02-06-05 09:00:00
%Y/%m/%d %H:%M:%S 2002/06/05 09:00:00
%y-%m-%d %I:%M:%S %p 2002/06/05 09:00:00
%I:%M:%S %p %Y/%m/%d 09:00:00 오전 2002/06/05
%y-%m-%d %H:%M [%a] 2002-06-05, 09:00 [수]
%I:%M:%S %p %d/%m/%y 05:01:58 오전 03/12/08
%Y-%m-%d %H:%M:%S 2008-12-03 17:01:58
%y/%m/%d %H:%M:%S 08/12/03 17:01:58

■ 사용예시

```
// 데이트형 문자열을 time_t값으로 변환해서 돌려준다.
// _DateToTick("1969-12-26 00:00:00", 0) -> default
// _DateToTick("1969-12-26 09:00:00", 1) -> Time value only (일자는 무시)
// _DateToTick("1969-12-26 10:00:00", 2) -> Date only (시간은 무시)

/*******************************************************
예제 : 지정된 문자열 날짜시간을 정수형 시간값으로 변환하기
*******************************************************/
void DateToTick()
{
        int nTime;
```

```
    string strTime;

    // 첫번째 파라메터 일자시간에 대한 정수값으로 변환
    nTime = _DateToTick("1969-12-26 00:00:00", 0);
    _TraceEx("time value : %d", nTime);

    // 현재 시간을 정수형 변수 nTime에 저장합니다.
    strTime = _TimeToString("%Y/%m/%d %H:%M:%S");
    nTime = _DateToTick(strTime, 0);

    // 시간 정수값을 메시지박스에 표시합니다.
    _MsgBox(nTime, "현재시간", 0, 1);
}
```

2.10.2 _DTGetValue()

함수원형

string _DTGetValue(string strObject, string strFormat)

파라메터	strObject : 날짜 오브젝트 이름 strFormat : 변환하고자 하는 날짜형식의 문자열
리턴형	문자열
설명	날짜 컨트롤에 설정 되어있는 날짜를 strFormat 형식으로 변환해서 문자열로 돌려줍니다.
함수활용	데이터베이스 검색에서 특정 날짜 조건이 들어가야 하는 경우 날짜컨트롤을 이용해서 날짜를 설정하여 사용합니다.

■ 날짜형식

※ 대문자는 연도 모두 표시 [ex) 2013]하며 , 소문자는 연도 뒤에 두자리 표시[ex)13]합니다.

※ 시간 포맷 설정시 대소문자 주의하시기 바랍니다.

%y-%m-%d %H:%M:%S 02-06-05 09:00:00

%Y/%m/%d %H:%M:%S 2002/06/05 09:00:00

%y-%m-%d %I:%M:%S %p 2002/06/05 09:00:00

%I:%M:%S %p %Y/%m/%d 09:00:00 오전 2002/06/05

%y-%m-%d %H:%M [%a] 2002-06-05, 09:00 [수]

%I:%M:%S %p %d/%m/%y 05:01:58 오전 03/12/08

%Y-%m-%d %H:%M:%S 2008-12-03 17:01:58

%y/%m/%d %H:%M:%S 08/12/03 17:01:58

```
/*********************************************************
예제 : 날짜오브젝트에서 날짜 갖고 오기
*********************************************************/
void GetDTGetValue()
{
	string strTime;
	// "DT1" 날짜 오브젝트에 설정된 날짜시간정보를 문자열 변수 strTime에 저장합니다.
	strTime = _DTGetValue("DT1@뷰페이지명", "%y-%m-%d %H:%M:%S");

	// 문자열 변수 strTime값을 메시지박스에 표시합니다.
	_MsgBox(strTime, "현재", 0, 2);
}
```

2.10.3 _DTSetValue()

함수원형

void _DTSetValue(string strObject, string strFormat)

파라메터	strObject : 날짜 오브젝트 이름 strFormat : 변환하고자 하는 날짜형식의 문자열
리턴형	없음
설명	시간포맷을 날짜 컨트롤에 설정합니다.
함수활용	날짜컨트롤을 초기값은 현재 시간이 표시됩니다. 현재 시간이 아닌 특정 시간(한시간 후, 익일 등)으로 설정하는 경우 사용합니다.

■ 날짜형식

※ 대문자는 연도 모두 표시 [ex) 2013]하며 , 소문자는 연도 뒤에 두자리 표시[ex)13]합니다.

※ 시간 포맷 설정시 대소문자 주의하시기 바랍니다.

%y-%m-%d %H:%M:%S 02-06-05 09:00:00

%Y/%m/%d %H:%M:%S 2002/06/05 09:00:00

%y-%m-%d %I:%M:%S %p 2002/06/05 09:00:00

%I:%M:%S %p %Y/%m/%d 09:00:00 오전 2002/06/05

%y-%m-%d %H:%M [%a] 2002-06-05, 09:00 [수]

%I:%M:%S %p %d/%m/%y 05:01:58 오전 03/12/08

%Y-%m-%d %H:%M:%S 2008-12-03 17:01:58

%y/%m/%d %H:%M:%S 08/12/03 17:01:58

```
/*********************************************************
예제 : 날짜오브젝트에서 날짜 설정하기
*********************************************************/
void DTSetValue()
{
	string strTime;
	// 현재시간을 갖고와서 문자열 변수 strTime에 저장합니다.
	strTime = _GetDateFormat("%y-%m-%d %H:%M:%S", 0);

	// 날짜컨트롤 오브젝트 명이 "시작일자"에 문자열 변수 strTime에 설정된 값을 적용합니다.
	_DTSetValue("시작일자@뷰페이지명", strTime);
	// 문자열 변수 strTime값을 메시지박스에 표시합니다.
	_MsgBox(strTime, "현재", 0, 2);
}
```

2.10.4 _GetLocalTime()

함수원형

string _GetLocalTime()

파라메터	없음
리턴값	문자열
설명	현재 시스템시간을 문자열형태로 읽어온다
함수활용	HSMS등에서 HOST PC와 LOCAL PC간 시간을 동기화 할 때 사용 할 수 있습니다.

■ 사용예

```
/*********************************************************
예제 : 현재 시간을 문자열로 돌려받기
*********************************************************/
void GetLocalTime()
{
	string strTime;

	// "년월일시분초밀리초" 문자열 형식으로 돌려준다

// 2021년 7월 26일 09시 00분 00초 999ms
	strTime = _GetLocalTime();

// "20210726090000999"
	_Trace(strTime);
}
```

2.10.5 _GetTick()

함수원형

double _GetTick()

파라메터	없음
리턴형	실수형 경과시간
설명	경과시간을 밀리초(1/1000) 단위의 실수값으로 돌려줍니다.
함수활용	특정 기능에 대해 소요시간을 체크하고자 하는 경우 사용합니다.

※ 리턴값은 소수점 이하값은 밀리초(1/1000)값이고, 소수점 앞의 값은 초단위 값입니다.

```
/*********************************************************
예제 : 10초 동안만 반복문 실행하기
*********************************************************/
void GetTick()
{
	double nTime1, nTime2;

	// 현재 시스템 시간을 실수형 변수 nTime1에 저장합니다.
	nTime1 = _GetTick();

	// 무한반복합니다.
	while(1)
	{
		// 현재 시스템 시간을 실수형 변수 nTime2에 저장합니다.
	nTime2 = _GetTick();

	// nTime2와 nTime1의 시간차를 구해서 10(초)보다 큰지 체크합니다.
	if((nTime2-nTime1) > 10)
	{
			// 10보다 크면 메시지 박스를 표시하고 아래 구문은 실행하지 않고 종료합니다.
			_MsgBox("10초경과","확인",0,2);
			return;
	}
	else
	{
		// 10보다 작으면 현재 소요시간을 트레이스 창에 표시합니다.
			_TraceEx("소요시간:%f", nTime2 - nTime1);
	}
```

```
            // 1초간 대기합니다.
        _Sleep(1000);
        }
}
```

2.10.6 _GetTime()

함수원형

int _GetTime()

파라메터	없음
리턴형	정수값
설명	시스템 현재시간을 정수값으로 돌려줍니다.
함수활용	시간연산을 하는 경우 사용합니다. 1분후 또는 1시간 전 등을 계산해야 하는 경우 현재시간의 정수값에 초단위로 환산하여 계산을 하여 사용합니다

```
/*********************************************************
예제 : 현재 시간을 정수값으로 가져오기
*********************************************************/
void TickToDate()
{
	_TraceEx("%s", _TickToDate( _GetTime() , "%I:%M:%S %p %y/%m/%d") );
	/*****************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
	*****************************************************************************************************/
}
```

2.10.7 _SetLocalTime()

함수원형

void _SetLocalTime(string strTime)

파라메터	strTime : 년월일시분초밀리초 형식의 문자열
리턴값	없음
설명	현재 시스템시간(LOCAL PC)을 변경합니다.
함수활용	HSMS에서 LOCAL PC의 시간을 설정합니다. 보통 HOST로부터 현재시간을 전달받거나 얻어와서 시간을 변경합니다.

■ 사용예

```
/*********************************************************
예제 : PC 시간 변경하기
*********************************************************/
void SetLocalTime()
{
	string strTime;

	// "년월일시분초밀리초" 문자열 형식으로 변경시간 설정
strTime = "20210726090000999";

// 2021년 7월 26일 09시 00분 00초 999ms
	_SetLocalTime(strTime);
}
```

2.10.8 _ShowElapseTime()

함수원형

string _ShowElapseTime(int nSecond, int nFlag)

파라메터	nSecond : 경과시간 (초) nFlag : 경과시간 표시형태 (돌려주는 문자열 형태) 0 : "시:분:초" 1 : "시:분" 2 : "일:시:분:초"
리턴값	문자열
설명	Stop Watch기능을 구현할 때 경과시간을 시:분:초 형태의 문자열로 변환합니다.
함수활용	레시피 자동공정 프로세스 구현할 때 경과시간 표시할 경우 사용하면 편리합니다.

```
/*********************************************************
예제 : 현재 시간을 문자열로 돌려받기
*********************************************************/
void ShowElapseTime()
{
	int nSleepSecond;
	string strTime;

	nSleepSecond = (_Rand() % 10) * 1000;

	_MsgBox(nSleepSecond, "지연동작을 수행합니다.", 0, 0);

	_Sleep(nSleepSecond);	// 0~10초 사이의 난수를 이용해 지연한다.

	_MsgBox(nSleepSecond, "지연동작이 완료되었습니다.", 0, 0);

	// 지정된 sec값을 원하는 string형태로 돌려준다.
	strTime = _ShowElapseTime(nSleepSecond, 0);
	_Trace(strTime);	// "00:01:00"		, H:M:S

	strTime = _ShowElapseTime(nSleepSecond, 1);
	_Trace(strTime);	// "00:01"		, H:M
	strTime = _ShowElapseTime(nSleepSecond, 2);
	_Trace(strTime);	// "00:00:01:00"	, D:H:M:S

	/*****************************************************************************************
```

```
        결과를 확인하기 위한 방법은 아래와 같습니다.
        1. eRun을 실행한다 [F5]
        2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
        3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
        ********************************************************************************************/
}
```

2.10.9 _TickToDate()

함수원형

string _TickToDate(int nTime, string strFormat)

파라메터	nTime : 년월일시분초의 정수형 데이터. strFormat : 변환하고자 하는 날짜의 형식을 문자열 형태로 지정합니다
리턴형	문자열
설명	시간연산을 하는 경우 사용합니다. 시간값을 지정형식의 문자열로 돌려줍니다.
함수활용	1분후 또는 1시간 전 등을 계산해야 하는 경우 현재시간의 정수값에 초단위로 환산하여 계산을 하여 나온 정수값을 시간포맷에 맞게 표현할 때 사용합니다.

■ 시간형식

※ 대문자는 연도 모두 표시 [ex) 2013]하며 , 소문자는 연도 뒤에 두자리 표시[ex)13]합니다.

※ 시간 포맷 설정시 대소문자 주의하시기 바랍니다.

%y-%m-%d %H:%M:%S 02-06-05 09:00:00

%Y/%m/%d %H:%M:%S 2002/06/05 09:00:00

%y-%m-%d %I:%M:%S %p 2002/06/05 09:00:00

%I:%M:%S %p %Y/%m/%d 09:00:00 오전 2002/06/05

%y-%m-%d %H:%M [%a] 2002-06-05, 09:00 [수]

%I:%M:%S %p %d/%m/%y 05:01:58 오전 03/12/08

%Y-%m-%d %H:%M:%S 2008-12-03 17:01:58

%y/%m/%d %H:%M:%S 08/12/03 17:01:58

```
/*********************************************************
예제 : 현재 시간을 포맷에 맞게 출력하기
*********************************************************/
void TickToDate()
{
	_TraceEx("%s", _TickToDate(_GetTime(), "%I:%M:%S %p %y/%m/%d") );
	/*****************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
	*****************************************************************************************************/
}
```

2.10.10 _TimeToString()

함수원형

string _TimeToString(string strFormat)

파라메터	strFormat : 변환하고자 하는 날짜의 형식을 문자열 형태로 지정합니다
리턴형	문자열
설명	현재 PC시간을 원하는 형식의 문자열로 돌려줍니다.
함수활용	날짜 컨트롤 오브젝트 없이 현재 시간을 문자열로 표현해야 하는 경우(데이터베이스 검색 또는 트렌드 검색시 현재시간이 필요한경우) 사용합니다.

■ 시간형식

※ 대문자는 연도 모두 표시 [ex) 2013]하며 , 소문자는 연도 뒤에 두자리 표시[ex)13]합니다.

※ 시간 포맷 설정시 대소문자 주의하시기 바랍니다.

%y-%m-%d %H:%M:%S 02-06-05 09:00:00

%Y/%m/%d %H:%M:%S 2002/06/05 09:00:00

%y-%m-%d %I:%M:%S %p 2002/06/05 09:00:00

%I:%M:%S %p %Y/%m/%d 09:00:00 오전 2002/06/05

%y-%m-%d %H:%M [%a] 2002-06-05, 09:00 [수]

%I:%M:%S %p %d/%m/%y 05:01:58 오전 03/12/08

%Y-%m-%d %H:%M:%S 2008-12-03 17:01:58

%y/%m/%d %H:%M:%S 08/12/03 17:01:58

```
/*********************************************************
예제 : 현재 시간을 포맷에 맞게 메시지박스에 출력하기
*********************************************************/
void TimeToString()
{
	_TraceEx("표현1: %s", _TimeToString("%d/%m/%y %I:%M:%S %p"));
	_TraceEx("표현2: %s", _TimeToString("%Y/%m/%d %H:%M:%S"));
	_TraceEx("표현3: %s", _TimeToString("%y/%m/%d %H:%M:%S"));

	/*************************************************************************************************
	결과를 확인하기 위한 방법은 아래와 같습니다.
	1. eRun을 실행한다 [F5]
	2. [F7] 함수수동호출을 활성화 시킨후 본 사용자 정의 함수를 실행시킵니다.
	3. [F6] 트레이스창을 활성화 하여 결과를 확인합니다.
	*************************************************************************************************/
}
```

2.11 그래프 함수

2.11.1 _GraphExportCSV()

함수원형

int _GraphExportCSV(string strObject, string strFile)

파라메터	strObject : 그래프 오브젝트 이름. strFile : CSV파일 이름
리턴형	정수값 0 : 성공 -1 : 실패
설명	라인그래프의 데이터를 CSV 파일 형식의 텍스트 파일로 저장합니다.
함수활용	

; CSV(Comma Separated Value) 형식의 파일은 엑셀 등에서 읽어서 활용합니다.

```
/*****************************************************************************
예제 : CSV저장.
*****************************************************************************/
void ExportCSV()
{
	// "LG1" 라인그래프의 그래프 데이터를 CSV파일에 저장.
	_GraphExportCSV("LG1@뷰페이지명", "C:₩₩export.csv");
}

/*****************************************************************************
예제2 : 파일이름 선택하고 CSV저장.
*****************************************************************************/
void SaveLotReport()
{
	string strFileName;
	int nFindCSV;

	strFileName = _FileDialog(@Base.sLotReportFolder, ".csv", 0);
	nFindCSV = _StrFind(strFileName, ".csv", 0);
	strFileName = _StrLeft(strFileName, nFindCSV);

	_GraphExportCSV("LINEGRAPH28@80", strFileName + "_POW.csv");
```

```
        _GraphExportCSV("LINEGRAPH26@81", strFileName + "_MFC.csv");
        _GraphExportCSV("LINEGRAPH28@81", strFileName + "_VOLT.csv");
        _GraphExportCSV("LINEGRAPH26@82", strFileName + "_CRYO.csv");
        _GraphExportCSV("LINEGRAPH26@83", strFileName + "_PRESSURE.csv");
}
```

2.11.2 _GraphGetMax()

함수원형

double _GraphGetMax(string strObject, int nAxis)

파라메터	strObject : 그래프 오브젝트 이름. nAxis : 축 번호 0 : X축 1 : Y축
리턴형	실수형
설명	라인그래프 오브젝트의 X축 또는 Y축 범위의 최대값을 돌려줍니다.
함수활용	

```
/*****************************************************************************
예제 : 그래프 오브젝트의 X, Y축 범위 최소, 최대값.
*****************************************************************************/
void DisplayMinMax()
{
	double X1, X2;
	double Y1, Y2;

	// "변화곡선" 그래프 오브젝트의 X축 최소, 최대값
	X1 = _GraphGetMin("변화곡선@뷰페이지명", 0);
	X2 = _GraphGetMax("변화곡선@뷰페이지명", 0);

	// "변화곡선" 그래프 오브젝트의 Y축 최소, 최대값
	Y1 = _GraphGetMin("변화곡선@뷰페이지명", 1);
	Y2 = _GraphGetMax("변화곡선@뷰페이지명", 1);

	_TraceEx("변화곡선 오브젝트 X1=%f, X2=%f, Y1=%f, Y2=%f", X1, X2, Y1, Y2);
}
```

2.11.3 _GraphGetMin()

함수원형

double _GraphGetMin(string strObject, int nAxis)

파라메터	strObject : 그래프 오브젝트 이름. nAxis : 축 번호 0 : X축 1 : Y축
리턴형	실수형
설명	라인그래프 오브젝트의 X축 또는 Y축 범위의 최소값을 돌려줍니다.
함수활용	

```
/*****************************************************************************
예제 : 그래프 오브젝트의 X, Y축 범위 최소, 최대값.
*****************************************************************************/
void DisplayMinMax()
{
	double X1, X2;
	double Y1, Y2;

	// "변화곡선" 그래프 오브젝트의 X축 최소, 최대값
	X1 = _GraphGetMin("변화곡선@뷰페이지명", 0);
	X2 = _GraphGetMax("변화곡선@뷰페이지명", 0);

	// "변화곡선" 그래프 오브젝트의 Y축 최소, 최대값
	Y1 = _GraphGetMin("변화곡선@뷰페이지명", 1);
	Y2 = _GraphGetMax("변화곡선@뷰페이지명", 1);

	_TraceEx("변화곡선 오브젝트 X1=%f, X2=%f, Y1=%f, Y2=%f", X1, X2, Y1, Y2);
}
```

2.11.4 _GraphGetPenColor()

함수원형

int _GraphGetPenColor(string strObject, int nPenNo)

파라메터	strObject : 그래프 오브젝트 이름. nPenNo : 펜 번호 (0~15)
리턴형	정수값
설명	라인그래프 오브젝트의 인덱스 번호에 해당하는 펜의 색상 값을 돌려줍니다.
함수활용	라인그래프 오브젝트에서는 eRun 실행시 범례를 표시할 수 있습니다. 범례가 표시하지 않은 경우 해당 내부함수를 이용하여 특정 펜(태그)에 대해서 색상을 확인할 수 있습니다. 라인그래프가 표시될 때 특정 펜(태그)이 어떤 색상인지 확인하고자 할 때 사용합니다.

```
/*****************************************************************************
예제 : 라인 (펜) 색상 표시.
*****************************************************************************/
void GraphPenColor()
{
	int nPen;
	int nColor;

	nPen =0;
	while(nPen < 16) {
		nColor = _GraphGetPenColor("LG14@뷰페이지명", nPen);
		_TraceEx("LineGraph Pen Color (%d) : %06X", nPen, nColor);
		nPen ++;
	}
}
```

함수 실행 결과

```
[2021-10-24 07:50:28.886]  LineGraph Pen Color (0) : FF0000   ; R=255(FF), G=00, B=00
[2021-10-24 07:50:28.886]  LineGraph Pen Color (1) : FFC000
[2021-10-24 07:50:28.887]  LineGraph Pen Color (2) : 150088
[2021-10-24 07:50:28.887]  LineGraph Pen Color (3) : 241CED
[2021-10-24 07:50:28.888]  LineGraph Pen Color (4) : 277FFF
[2021-10-24 07:50:28.888]  LineGraph Pen Color (5) : 00F2FF
[2021-10-24 07:50:28.888]  LineGraph Pen Color (6) : 4CB122
[2021-10-24 07:50:28.889]  LineGraph Pen Color (7) : E8A200
```

[2021-10-24 07:50:28.889] LineGraph Pen Color (8) : CC483F
[2021-10-24 07:50:28.889] LineGraph Pen Color (9) : A449A3
[2021-10-24 07:50:28.890] LineGraph Pen Color (10) : 577AB9
[2021-10-24 07:50:28.890] LineGraph Pen Color (11) : C9AEFF
[2021-10-24 07:50:28.890] LineGraph Pen Color (12) : 0EC9FF
[2021-10-24 07:50:28.890] LineGraph Pen Color (13) : 1DE6B5
[2021-10-24 07:50:28.891] LineGraph Pen Color (14) : BE9270
[2021-10-24 07:50:28.891] LineGraph Pen Color (15) : E7BFC8

2.11.5 _GraphGetPenName()

함수원형

string _GraphGetPenName(string strObject, int nPenNo)

파라메터	strObject : 그래프 오브젝트 이름. nPenNo : 펜 번호 (0~15)
리턴형	문자열
설명	라인그래프 오브젝트의 펜에 연결된 태그이름을 읽어옵니다.
함수활용	라인그래프 오브젝트에서는 eRun 실행시 범례를 표시할 수 있습니다. 범례가 표시하지 않은 경우 해당 내부함수를 이용하여 특정 펜(태그)에 대해서 태그명을 확인할 수 있습니다. 라인그래프가 표시될 때 특정 펜(태그)이 어떤 태그인지 확인하고자 할 때 사용합니다.

```
/*****************************************************************************
예제 : 라인 (펜) 태그명 표시.
*****************************************************************************/
void DisplayPenName()
{
	string strPenName;

	// "LG1" 라인그래프 오브젝트의 첫번째 펜 펜이름(태그명).
	strPenName = _GraphGetPenName("LG1@뷰페이지명", 0);

	// 펜이름 메시지박스에 표시하기
	_MsgBox(strPenName, "1번 펜 태그 명", 0, 2);

}
```

2.11.6 _GraphGetZoom()

함수원형

int _GraphGetZoom(string strObject)

파라메터	strObject : 그래프 오브젝트 이름.
리턴형	정수값 0 : no Zoom 1 : 좌우 2 : 상하 3 : 상하좌우 -1 : 실패
설명	라인그래프 오브젝트의 영역확대 기능 선택 설정값을 읽어옵니다.
함수활용	라인그래프 오브젝트의 도구바에 영역확대 버튼이 있지만, 별도의 버튼을 구성할 경우 이 함수를 이용해서 영역확대 선택상태를 표시합니다.

```
/*****************************************************************************
예제 : 라인그래프 ZOOM 선택.
*****************************************************************************/
void GraphZoom()
{
	int nZoom;

	// "LG1" 라인그래프 오브젝트의 ZOOM 선택상태.
	nZoom = _GraphGetZoom("LG1@뷰페이지명");
	@LOT.GRAPH1 = nZoom;

	_TraceEx("LG1 ZOOM =%d", nZoom);
}
```

2.11.7 _GraphPlay()

함수원형

void _GraphPlay(string strObject, int nFlag)

파라메터	strObject : 그래프 오브젝트 이름. nFlag : 실시간 갱신 또는 정지 0 : 멈춤 1 : 갱신
리턴형	없음
설명	실시간 그래프의 화면갱신을 하거나 멈춥니다.
함수활용	라인그래프 오브젝트에서 실시간으로 표시될때 사용자가 일시적으로 라인그래프의 표시를 정지하여 표시된 데이터를 확인을 할 수 있고 특정 조건에 따라 라인그래프에 표시되지 않게 할 수 있습니다

※ 그래프 형태가 실시간이어야만 합니다.

```
/*****************************************************************************
예제 : 실시간 그래프 갱신.
*****************************************************************************/
void GraphPlay(int nPlay)
{
	// "LG1" 라인그래프 실시간 갱신(1), 멈춤(0).
	_GraphPlay("LG1@뷰페이지명", nPlay);
}
```

2.11.8 _GraphReset()

함수원형

void _GraphReset(string strObject)

파라메터	strObject : 그래프 오브젝트 이름.
리턴형	없음
설명	실시간 그래프 오브젝트를 초기화 시킵니다.
함수활용	현재 라인 그래프 오브젝트를 초기화하고 라인그래프를 처음부터 표시합니다. 프로그램 테스트 및 필요없는 데이터가 수신되어 트렌드로 표시되는 경우 초기화를 라인 그래프 오브젝트를 초기화하여 표시된 부분을 삭제합니다.

※ 그래프 형태가 실시간이어야만 합니다.

```
/*****************************************************************************
예제 : 실시간 그래프 화면 초기화.
*****************************************************************************/
void GraphReset()
{
	// "LG1" 라인그래프 RESET.
	_GraphReset("LG1@뷰페이지명");
}

void ResetGraph()
{
	_GraphReset("SERVO_GRAPH@뷰페이지명");
}

void PlayGraph()
{
	_GraphPlay("SERVO_GRAPH@뷰페이지명", 1);
}

void PauseGraph()
{
	_GraphPlay("SERVO_GRAPH@뷰페이지명", 0);
}
```

2.11.9 _GraphScan()

함수원형

void _GraphScan(string strObject)

파라메터	strObject : 그래프 오브젝트 이름.
리턴형	없음
설명	특정기간을 데이터베이스에서 검색해서 검색결과를 그래프로 표시합니다.
함수활용	툴바가 없는 경우 및 자동으로 검색해야하는 경우 사용합니다.

※ 그래프 형태가 히스토리로 설정되어 있어야 합니다.

```
/*****************************************************************************
예제 : 특정일 검색 후 그래프에 표시하기
*****************************************************************************/
void ScanTrend()
{
    string strStart, strEnd;

    // 날짜컨트롤의 검색시간 일자와 검색 종료일자를 가지고 옵니다.
    strStart =_DTGetValue("시작일자@뷰페이지명", "%Y-%m-%d %H:%M:%S");
    strEnd =_DTGetValue("종료일자@뷰페이지명", "%Y-%m-%d %H:%M:%S");

    // 트렌드 X축의 최소, 최대값을 설정합니다.
    _GraphSetMinMax ("변화곡선", 0, 0, strStart, strEnd);

    // 그래프 오브젝트에 설정된 태그를 데이터베이스에서 검색하고 결과를 그래프로 출력합니다.
    _GraphScan("변화곡선@뷰페이지명");

}
```

2.11.10 _GraphSetBaseLine()

함수원형

void _GraphSetBaseLine(string strObject, double fMin, double fMax, int nRGB)

파라메터	strObject : 그래프 오브젝트 이름. fMin : 최소값 fMax : 최대값 nRGB : 색상값
리턴형	없음
설명	그래프 오브젝트의 설정된 최저값, 최고값, 베이스 라인의 색을 설정합니다.
함수활용	라인그래프에 표시되는 데이터에 대해서 특정영역의 범위를 설정하여 범위를 벗어나는지에 대하여 라인그래프에 베이스라인을 설정하여 확인할 수 있습니다. 라인그래프에 오류데이터가 표시되는지 확인 시 사용할 수 있습니다.

※ 라인그래프 오브젝트 속성에 기준선 표시 설정이 되어 있어야 합니다..

```
/*****************************************************************************
예제 : 베이스라인 설정
*****************************************************************************/
void SetBaseLine()
{
	// 라인그래프의 최저값:0, 최고값100에 파란색으로 베이스라인을 표시합니다.
	_GraphSetBaseLine("변화곡선@뷰페이지명", 0.0, 100.0, 0xff0000);

}
```

2.11.11 _GraphSetMinMax()

함수원형

void _GraphSetMinMax(string strObject, int nAxis, int nPenNo, double fMin, double fMax)

void _GraphSetMinMax(string strObject, int nAxis, int nPenNo, string strMin, string strMax)

파라메터	strObject : 그래프 오브젝트 이름. nAxis : 축 번호 0 : X축 1 : Y축 nPenNo : 펜번호 (0~15) fMin : 최소값 strMin : 날짜형식의 시작일자 문자열 ("YYYY-mm-dd HH:MM:SS") fMax : 최대값 strMax : 날짜형식의 종료일자 문자열 ("YYYY-mm-dd HH:MM:SS")
리턴형	실수형
설명	라인그래프 오브젝트의 X축 또는 Y축의 최소, 최대값을 변경합니다.
함수활용	입력창 오브젝트 등 입력처리가 가능한 오브젝트를 사용하여 라인 그래프의 최대/최소값을 변경할 수 있습니다. 서로 다른 다양한 범위로 라인그래프를 표시할 경우 각 펜의 최소, 최대값을 함수를 통해서 변경합니다.

※ 그래프 속성에서 "Y축 그리드 간격값으로 사용", "X축 그리드 시간 간격값으로 사용"이 설정되어야 내부함수가 처리됩니다.

```
void GraphSetMinMax()
{
	int nPenNo;
	int X, Y;

	X = 0;	// X축
	Y = 1;	// Y축
	nPenNo = 0;	// 첫번째 펜.

	// X축 최소값:0, X축 최대값: 1000으로 설정합니다.
	_GraphSetMinMax("변화곡선@뷰페이지명", Y, nPenNo, 0.0, 1000.0);

	// X축의 최소/최대값을 특정일자로 설정합니다.
	_GraphSetMinMax("변화곡선@뷰페이지명", X, nPenNo, "2002-6-5 00:00:00", "2007-10-22 23:59:59");
```

```
        // 히스토리 그래프인 경우 갱신함수를 호출합니다.
        _RefreshControl ("변화곡선@뷰페이지명");

        // 라인 그래프 오브젝트 속성이 실시간인 경우에는 X축의 최대최소값 변경은 적용되지 않습니다.
        // 실시간 그래프의 경우 _RefreshControl()는 사용하지 않아도 됩니다. (자동갱신)
}
```

2.11.12 _GraphSetPenColor()

함수원형

void _GraphSetPenColor(string strObject, int nPenNo, int nRGB)

파라메터	strObject : 그래프 오브젝트 이름. nPenNo : 펜 번호 (0~15) nRGB : 색상
리턴형	없음
설명	라인그래프 오브젝트의 펜 색상을 변경합니다.
함수활용	

```
/*****************************************************************************
예제 : 라인 (펜) 색상 표시.
*****************************************************************************/
void GraphPenColor()
{
	int nPen;
	int nColor;

	nPen =0;
	while(nPen < 16) {
		nColor = _GraphGetPenColor("LG14@뷰페이지명", nPen);
		_TraceEx("LineGraph GetPen Color (%d) : %06X", nPen, nColor);

		_GraphSetPenColor("LG14@뷰페이지명", nPen, 0x00FF00);
		nColor = _GraphGetPenColor("LG14@뷰페이지명", nPen);
		_TraceEx("--> Pen Color (%d) : %06X", nPen, nColor);
		nPen ++;
	}
}
```

함수 실행 결과

```
[2021-10-24 08:00:49.394]  LineGraph GetPen Color (0) : FF0000
[2021-10-24 08:00:49.395]  --> Pen Color (0) : 00FF00
[2021-10-24 08:00:49.395]  LineGraph GetPen Color (1) : FFC000
[2021-10-24 08:00:49.396]  --> Pen Color (1) : 00FF00
[2021-10-24 08:00:49.396]  LineGraph GetPen Color (2) : 150088
[2021-10-24 08:00:49.396]  --> Pen Color (2) : 00FF00
[2021-10-24 08:00:49.397]  LineGraph GetPen Color (3) : 241CED
```

[2021-10-24 08:00:49.397] --> Pen Color (3) : 00FF00
[2021-10-24 08:00:49.397] LineGraph GetPen Color (4) : 277FFF
[2021-10-24 08:00:49.398] --> Pen Color (4) : 00FF00
[2021-10-24 08:00:49.398] LineGraph GetPen Color (5) : 00F2FF
[2021-10-24 08:00:49.398] --> Pen Color (5) : 00FF00
[2021-10-24 08:00:49.399] LineGraph GetPen Color (6) : 4CB122
[2021-10-24 08:00:49.399] --> Pen Color (6) : 00FF00
[2021-10-24 08:00:49.400] LineGraph GetPen Color (7) : E8A200
[2021-10-24 08:00:49.400] --> Pen Color (7) : 00FF00
[2021-10-24 08:00:49.400] LineGraph GetPen Color (8) : CC483F
[2021-10-24 08:00:49.401] --> Pen Color (8) : 00FF00
[2021-10-24 08:00:49.401] LineGraph GetPen Color (9) : A449A3
[2021-10-24 08:00:49.401] --> Pen Color (9) : 00FF00
[2021-10-24 08:00:49.402] LineGraph GetPen Color (10) : 577AB9
[2021-10-24 08:00:49.402] --> Pen Color (10) : 00FF00
[2021-10-24 08:00:49.402] LineGraph GetPen Color (11) : C9AEFF
[2021-10-24 08:00:49.403] --> Pen Color (11) : 00FF00
[2021-10-24 08:00:49.403] LineGraph GetPen Color (12) : 0EC9FF
[2021-10-24 08:00:49.403] --> Pen Color (12) : 00FF00
[2021-10-24 08:00:49.403] LineGraph GetPen Color (13) : 1DE6B5
[2021-10-24 08:00:49.404] --> Pen Color (13) : 00FF00
[2021-10-24 08:00:49.404] LineGraph GetPen Color (14) : BE9270
[2021-10-24 08:00:49.404] --> Pen Color (14) : 00FF00
[2021-10-24 08:00:49.405] LineGraph GetPen Color (15) : E7BFC8
[2021-10-24 08:00:49.405] --> Pen Color (15) : 00FF00

2.11.13 _GraphSetPenName()

함수원형

void _GraphSetPenName(string strObject, int nPenNo, string strTagname)

파라메터	strObject : 그래프 오브젝트 이름. nPenNo : 펜 번호 (0~15) strTagname : 태그명 ("태그그룹.이름")
리턴형	없음
설명	라인그래프 오브젝트의 펜에 연결된 태그이름을 변경합니다.
함수활용	하나의 라인 그래프 오브젝트에서 사용자가 특정 조건 및 선택에 따라 라인 그래프의 펜을 변경하여 표시해야 하는 경우 사용합니다. 예를 들어서 서로 상관없는 두개의 장비에 대해 장비선택에 따라 라인 그래프 표시하는 경우 해당됩니다.

표시하지 말아야하는 펜의 경우에 대해서는 태그명에 ""을 설정하면 됩니다.

```
/*****************************************************************************
예제 : 라인 (펜) 색상 표시.
*****************************************************************************/
void SetPenName()
{
	// 태그명에 "@"문자는 포함 시키지 않습니다.
	// "LG1" 라인그래프의 첫번째 펜 이름(태그명)을 "태그그룹.태그명17"로 변경합니다.
	_GraphSetPenName("LG1@뷰페이지명", 0, "태그그룹.태그명17");

	// 히스토리 그래프인 경우
	_RefreshControl("LG1@뷰페이지명");

}
```

2.11.14 _GraphSetZoom()

함수원형

void _GraphSetZoom(string strObject, int nZoom)

파라메터	strObject : 그래프 오브젝트 이름. nZoom : 확대방법 0 : no Zoom 1 : 좌우 2 : 상하 3 : 상하좌우
리턴형	없음
설명	라인그래프 오브젝트의 영역확대 방법을 선택합니다.
함수활용	라인그래프 오브젝트의 도구바 사용하지 않고, 별도의 버튼으로 확대방법을 구현할 경우 사용합니다.

```
/*****************************************************************************
예제 : 라인그래프 ZOOM 선택.
*****************************************************************************/
void GraphZoom()
{
	// "LG1" 라인그래프 오브젝트의 ZOOM 선택.
	_GraphSetZoom("LG1@뷰페이지명", @LOT.GRAPH1);
}
```

2.11.15 _GraphShowPen()

함수원형

void _GraphShowPen(string strObject, int nPenNo, int nFlag)

파라메터	strObject : 그래프 오브젝트 이름. nPenNo : 펜번호 (0~15) nFlag : 보이기 또는 숨기기 0 : 숨기기 1 : 보이기
리턴형	없음
설명	라인그래프 오브젝트에 펜을 보이거나 숨깁니다.
함수활용	

```
/*****************************************************************************
예제 : 펜 보이기 숨기기
*****************************************************************************/
void ShowPen(int nPen, int nShow)
{
	// 첫번째 PEN 보이기.
	_GraphShowPen("변화곡선@뷰페이지명", nPen, nShow);

}
```

2.12 그리드 함수

2.12.1 _GridAppend()

함수원형

void _GridAppend(string strObject, int nAxis)

파라메터	strObject : 오브젝트이름. nAxis : 축번호 0 : 행방향(가로)으로 1개의 행 추가. 1 : 열방향(세로)으로 1개의 열 추가.
리턴형	없음
설명	지정된 그리드 오브젝트의 행 또는 열을 마지막에 추가합니다.
함수활용	데이터베이스 쿼리 결과가 표시되어 자동으로 행열수가 조정된 경우 또는 초기 행열개수 이상으로 셀이 필요한 경우 사용합니다.

```
/*********************************************************
예제 : 행/열 추가하기
*********************************************************/
void GridAppend()
{
	// "GD20" 그리드 오브젝트에 1개 열을 추가합니다.
	_GridAppend("GD20@뷰페이지명", 1);

	// "GD20" 그리드 오브젝트에 1개 행을 추가합니다.
	_GridAppend("GD20@뷰페이지명", 0);
}
```

2.12.2 _GridClearRange()

함수원형

void _GridClearRange(string strObject, int nCol1, int nRow1, int nCol2, int nRow2)

파라메터	strObject : 오브젝트이름. nCol1 : 시작 행번호.(0부터 시작) nRow1 : 시작 열번호. nCol2 : 마지막 행번호. nRow2 : 마지막 열번호.
리턴형	없음
설명	지정된 그리드 오브젝트의 시트에서 시작 행/열부터 마지막 행/열까지 공란으로 만듭니다.
함수활용	그리드에 데이터를 반복해서 표시되는 경우 이전에 표시된 데이터의 행/열 개수가 다른 경우 설정된 영역을 공란을 만들고 다른 데이터를 표시할 때 사용합니다.

```
/*********************************************************
예제 : 그리드 내용 전체 삭제하기
*********************************************************/
void GridClearRange()
{
	int nCols, nRows;

	// "GD20" 그리드 오브젝트의 행 개수
	nCols = _GridColCount("GD20@뷰페이지명")-1;
	// "GD20" 그리드 오브젝트 열 개수.
	nRows = _GridRowCount("GD20@뷰페이지명")-1;

	// 그리드 오브젝트 명이 "GD20" 인 그리드의 전체영역의 내용을 삭제합니다.
	_GridClearRange("GD20@뷰페이지명", 0, 0, nCols, nRows);

	// 그리드 오브젝트 명이 "GD20" 를 갱신합니다.
	_RefreshControl("GD20@뷰페이지명");
}
```

2.12.3 _GridColCount()

함수원형

int _GridColCount(string strObject)

파라메터	strObject : 오브젝트이름.
리턴형	행개수 정수값
설명	지정된 그리드 오브젝트의 행의 개수를 돌려줍니다.
함수활용	그리드 오브젝트의 셀을 반복하여 처리할 때마다 행의 개수가 조건에 따라 바뀌는 경우 (데이터베이스 쿼리 결과에 따라 행수가 조정되는 경우, 열이 추가된 경우) 사용합니다.

```
/*********************************************************
예제 : 그리드 내용 전체 삭제하기
*********************************************************/
void GridClearRange()
{
	int nCols, nRows;

	// "GD20" 그리드 오브젝트의 행 개수
	nCols = _GridColCount("GD20@뷰페이지명")-1;
	// "GD20" 그리드 오브젝트 열 개수.
	nRows = _GridRowCount("GD20@뷰페이지명")-1;

	// 그리드 오브젝트 명이 "GD20" 인 그리드의 전체영역의 내용을 삭제합니다.
	_GridClearRange("GD20@뷰페이지명", 0, 0, nCols, nRows);

	// 그리드 오브젝트 명이 "GD20" 를 갱신합니다.
	_RefreshControl("GD20@뷰페이지명");
}
```

2.12.4 _GridDelete()

함수원형

void _GridDelete(string strObject, int nAxis, int nCount)

파라메터	strObject : 오브젝트이름. nAxis : 축번호 0 : 행(가로)축 1 : 열(세로)축 nCount : 삭제하고자 하는 행 번호 또는 열의 개수
리턴형	없음
설명	지정된 그리드 오브젝트의 행 또는 열을 삭제합니다.
함수활용	총 행/열 개수 고정된 그리드에서 추가 또는 삽입으로 행/열의 개수를 맞춰야 하는 경우 사용합니다.

```
/*********************************************************
예제 : 그리드 삭제하기
*********************************************************/
void GridDelete()
{
	// 그리드 오브젝트 명이 "그리드"인 그리드의 10번 행을 삭제합니다.
	// 10번 행을 삭제 후에는 10번행 이후의 행들은 왼쪽으로 이동합니다.
	_GridDelete("그리드@뷰페이지명", 0, 10);

	// 그리드 오브젝트 명이 "그리드" 인 그리드의 10번 열을 삭제합니다.
	// 10번 열을 삭제 후에는 10번 열 이후의 열들은 위쪽으로 이동합니다.
	_GridDelete("그리드@뷰페이지명", 1, 10);
}
```

2.12.5 _GridDeleteAll()

함수원형

void _GridDeleteAll(string strObject)

파라메터	strObject : 오브젝트이름.
리턴형	없음
설명	그리드의 모든 행 / 열을 삭제합니다.
함수활용	데이터베이스의 쿼리를 다시하는 경우 또는 그리드를 초기화하는 경우 사용합니다.

```
/*********************************************************
예제 : 그리드 행, 열 모두 삭제하기
*********************************************************/
void GridDeleteAll()
{
	// "GD20" 그리드 오브젝트의 모든 셀을 삭제합니다.
	_GridDeleteAll("GD20@뷰페이지명");
}
```

2.12.6 _GridEnable()

함수원형

void _GridEnable(string strObject, int nCol, int nRow, int nFlag)

파라메터	strObject : 오브젝트이름. nCol : 행번호 nRow : 열번호 nFlag : 입력 가능 또는 불가능하게. 0 : 키보드 입력 불가 1 : 키보드 입력 가능
리턴형	없음
설명	그리드오브젝트의 특정 셀에 대하여 입력 허용할 것인지 설정합니다.
함수활용	데이터베이스에서 데이터를 쿼리 한 결과가 표시된 경우 데이터 수정을 못하게 하는 경우 사용합니다.

```
/*********************************************************
예제 : 그리드 입력 제어하기
*********************************************************/
void GridEnable ()
{
	// 그리드 오브젝트 명이 "그리드"인 셀(5, 10)의 입력이 불가능하게 설정합니다.
	_GridEnable("그리드@뷰페이지명", 5, 10, 0);

	// 그리드 오브젝트 명이 "그리드"인 셀(10, 20)의 입력이 가능하게 설정합니다.
	_GridEnable("그리드@뷰페이지명", 10, 20, 1);
}
```

2.12.7 _GridEnableEx()

함수원형

void _GridEnableEx(string strObject, int nCol1, int nRow1, int nCol2, int nRow2, int nFlag)

파라메터	strObject : 오브젝트이름. nCol1 : 시작 행번호 nRow1 : 시작 열번호 nCol2 : 마지막 행번호 nRow2 : 마지막 열번호 nFlag : 입력 가능 또는 불가능하게. 0 : 키보드 입력 불가 1 : 키보드 입력 가능
리턴형	없음
설명	그리드오브젝트의 특정 셀에 대하여 입력 허용할 것인지 설정합니다.
함수활용	데이터베이스에서 데이터를 쿼리 한 결과가 표시된 경우 데이터 수정을 못하게 하는 경우 사용합니다.

```
/*********************************************************
예제 : 그리드 입력 제어하기
*********************************************************/
void GridEnable ()
{
	// 그리드 오브젝트 명이 "그리드"인 셀(5, 10)의 입력이 불가능하게 설정합니다.
	_GridEnable("그리드@뷰페이지명", 5, 10, 0);

	// 그리드 오브젝트 명이 "그리드"인 셀(10, 20)의 입력이 가능하게 설정합니다.
	_GridEnable("그리드@뷰페이지명", 10, 20, 1);

	// "그리드" 오브젝트 CELL(0, 0) 부터 CELL(5, 10)까지 입력이 불가능하게 설정합니다.
	_GridEnableEx("그리드@뷰페이지명", 0, 0, 5, 10, 0);
}
```

2.12.8 _GridExportCSV()

함수원형

int _GridExportCSV(string strObject, string strFile)

파라메터	strObject : 오브젝트이름. strFile : 경로포함 파일이름
리턴형	정수값 0 : 성공 -1 : 실패
설명	지정된 그리드 오브젝트의 내용을 지정된 CSV 파일로 저장합니다.
함수활용	레시피 및 특정 설정값을 입력 후 파일로 저장하여 관리하는 경우 사용합니다. 데이터베이스에서 검색결과를 파일로 저장하는 경우 사용합니다.

※ _GridLoad() 함수는 _GridSave() 함수를 이용해서 저장된 파일만 정상적으로 Load할 수 있습니다.
또한 저장시 그리드의 CELL이 Join된 상태에 대해서는 Load시 자동으로 CELL을 Join 하지 않습니다.

```
/*********************************************************
예제 : 그리드 내용 CSV 파일에 저장하기
*********************************************************/
void GridExportCSV()
{
	int nRet;
	// "GD20" 그리드 오브젝트의 내용을
	// C드라이브의 루트에 "TEST.CSV" 파일명으로 저장합니다.
	nRet = _GridExportCSV("GD20@뷰페이지명", "C:₩TEST.CSV");

	if(nRet < 0) {
		_MsgBox(GetSystemErrorMsg(), "error", 0, 0);
	}
}
```

2.12.9 _GridGetCol()

함수원형

int _GridGetCol(string strObject)

파라메터	strObject : 오브젝트이름.
리턴형	행번호 정수값
설명	현재 선택한 셀의 행번호를 돌려줍니다.
함수활용	그리드의 현재 포커스가 되어있는 셀에 대해서 처리(색상처리, 정렬, 셀값 읽기 등)를 하기 위해 사용합니다.

```
/*********************************************************
예제 : 그리드 선택
*********************************************************/
void GridCell()
{
	int nCol, nRow;

	// "GD20" 오브젝트의 현재 선택 되어있는 행의 번호를 가져온다.
	nCol = _GridGetCol("GD20@뷰페이지명");

	// "GD20" 오브젝트의 현재 선택 되어있는 열의 번호를 가져온다.
	nRow = _GridGetRow("GD20@뷰페이지명");

	_TraceEx("Selected cell -- col=%d, row=%d", nCol, nRow);
}
```

2.12.10 _GridGetOldCol()

함수원형

int _GridGetOldCol(string strObject)

파라메터	strObject : 오브젝트이름
리턴형	행번호 정수값
설명	지정된 오브젝트의 셀 포커스가 변경되었을 경우 변경전의 행 번호 값을 돌려줍니다.
함수활용	이전 셀에 대해 셀에 입력된 값이 유효한지 확인하는 경우 이전 행번호를 참고하여 사용 합니다.

```
/*********************************************************
예제 : 변경 전 셀번호값 표시하기
*********************************************************/
void GridOldCell()
{
	int nOldCol;
	int nOldRow;
	string strObject, strText;

	strObject = "GD20@뷰페이지명";
	// "GD20" 오브젝트의 셀 선택바가 이동되기전의 행번호를 가져온다.
	nOldCol = _GridGetOldCol(strObject);
	// "GD20" 오브젝트의 셀 선택바가 이동되기전의 열번호를 가져온다.
	nOldRow = _GridGetOldRow(strObject);

	strText = _GridGetValue("strObject", nOldCol, nOldRow);
	_TraceEx("Old col=%d, Old row=%d, text =%s", nOldCol, nOldRow, strText);
}
```

2.12.11 _GridGetOldRow()

함수원형

int _GridGetOldRow(string strObject)

파라메터	strObject : 오브젝트이름
리턴형	열번호 정수값
설명	지정된 오브젝트의 셀 포커스가 변경되었을 경우 변경전의 열 번호 값을 돌려줍니다.
함수활용	이전 셀에 대해 셀에 입력된 값이 유효한지 확인하는 경우 이전 열번호를 참고하여 사용 합니다.

```
/*********************************************************
예제 : 변경 전 셀번호값 표시하기
*********************************************************/
void GridOldCell()
{
	int nOldCol;
	int nOldRow;
	string strObject, strText;

	strObject = "GD20@뷰페이지명";
	// "GD20" 오브젝트의 셀 선택바가 이동되기전의 행번호를 가져온다.
	nOldCol = _GridGetOldCol(strObject);
	// "GD20" 오브젝트의 셀 선택바가 이동되기전의 열번호를 가져온다.
	nOldRow = _GridGetOldRow(strObject);

	strText = _GridGetValue("strObject", nOldCol, nOldRow);
	_TraceEx("Old col=%d, Old row=%d, text =%s", nOldCol, nOldRow, strText);
}
```

2.12.12 _GridGetRow()

함수원형

int _GridGetRow(string strObject)

파라메터	strObject : 오브젝트이름.
리턴형	열번호 정수값
설명	현재 선택한 셀의 열번호를 돌려줍니다.
함수활용	그리드의 현재 포커스가 되어있는 셀에 대해서 처리(색상처리, 정렬, 셀값 읽기 등)를 하기 위해 사용합니다.

```
/*********************************************************
예제 : 그리드 선택열
*********************************************************/
void GridCell()
{
	int nCol, nRow;

	// "GD20" 오브젝트의 현재 선택 되어있는 행의 번호를 가져온다.
	nCol = _GridGetCol("GD20@뷰페이지명");

	// "GD20" 오브젝트의 현재 선택 되어있는 열의 번호를 가져온다.
	nRow = _GridGetRow("GD20@뷰페이지명");

	_TraceEx("Selected cell -- col=%d, row=%d", nCol, nRow);
}
```

2.12.13 _GridGetValue()

함수원형

string _GridGetValue(string strObject, int nCol, int nRow)

파라메터	strObject : 오브젝트이름. nCol : 행번호 nRow : 열번호
리턴형	문자열
설명	그리드 오브젝트의 지정한 셀의 값을 읽어 옵니다.
함수활용	

```
/*********************************************************
예제 : 그리드 오브젝트 셀의 텍스트 가져오기
*********************************************************/
void Grid()
{
	// 열 1개 추가
	_GridAppend("GD20@뷰페이지명", 1);
	//
	// col0, row0에 문자열 표시
	_GridSetValue("GD20@뷰페이지명", 0, 0, "KOREA");
	// col1, row0에 문자열 표시
	_GridSetValue("GD20@뷰페이지명", 1, 0, "World networking");
	// col2, row0에 실수값 표시
	_GridSetValue("GD20@뷰페이지명", 2, 0, 100.5);

	// 0번째 ROW에 대해서 높이를 50pixel로 변경합니다.
	_GridSetHeight("GD20@뷰페이지명", 0, 50);
	_RefreshControl("GD20@뷰페이지명");

	strText = _GridGetValue("GD20@뷰페이지명", 0, 0);
	_TraceEx("text = %s", strText);
}
```

2.12.14 _GridGoto()

함수원형

void _GridGoto(string strObject, int nCol, int nRow)

파라메터	strObject : 오브젝트이름. nCol : 행번호 nRow : 열번호
리턴형	없음
설명	그리드 오브젝트의 지정한 위치로 선택바를 이동시킵니다.
함수활용	상황 및 조건에 따라 그리드의 셀의 특정위치로 선택바를 옮겨야 하는 경우 사용합니다.

```
/*********************************************************
예제 : 특정 셀로 이동하기
*********************************************************/
void GridGoto()
{
	// "GD20" 오브젝트의 셀(0, 10)으로 선택바를 이동시킵니다.
	_GridGoto("GD20@뷰페이지명", 0, 10);
}
```

2.12.15 _GridImportCSV()

함수원형

int _GridImportCSV(string strObject, string strFile)

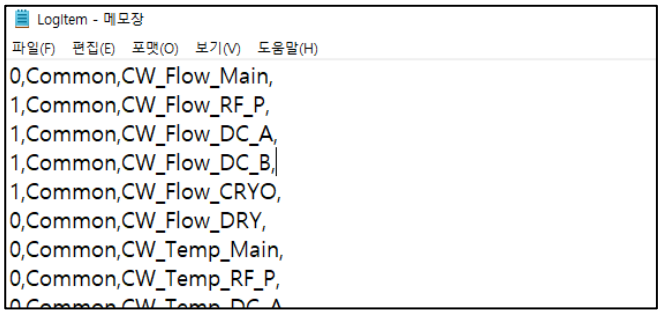
파라메터	strObject : 오브젝트이름. strFile : 경로포함 파일이름
리턴형	정수값 0 : 성공 -1 : 실패
설명	CSV (Comma Separated Values) 형식의 텍스트 파일을 그리드 오브젝트에 읽어서 표시합니다.
함수활용	레시피 및 특정 설정 값을 CSV파일로 저장하고 읽어서 관리하는 경우 사용합니다.

```
/*********************************************************
예제 : CSV 포맷 읽어오기
*********************************************************/
void GridImport()
{
	int nRet;

	// 첫 COL의 타입을 CheckBox 형태로 변경합니다.
	_GridSetTypeEx("GD31@뷰페이지명", 0, 0, 0, 48, "CHECKBOX", "","");    //체크박스 삽입

	// CSV파일을 읽어서 그리드 오브젝트 "GD31"에 표시합니다.
	nRet = _GridImportCSV("GD31@뷰페이지명", "e:₩₩Temp2₩₩LogItem2.csv");
	//
	_TraceEx("_GridImportCSV(GD31@뷰페이지명) ret =%d", nRet);
	// 그리드 갱신
	_RefreshControl("GD31@뷰페이지명");
}
```

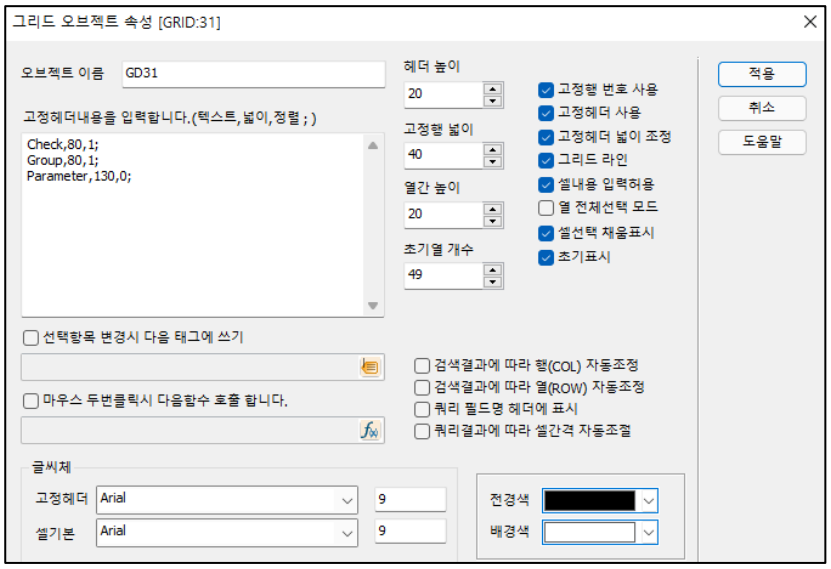
트레이스 창(F6)을 열어서 오류메시지 확인가능합니다.

```
[2021-12-02 15:17:39.265] e:\\Temp2\\LogItem2.csv을(를) 찾을 수 없습니다.
[2021-12-02 15:17:39.274] _GridImportCSV(GD31) ret =-1
```

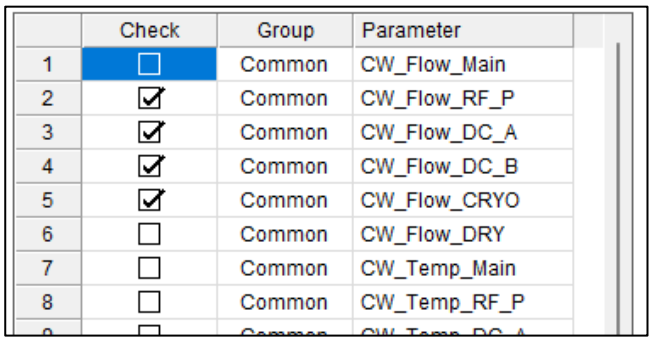
LogItem.csv 파일입니다.



그리드 속성창



실행결과 화면



	Check	Group	Parameter
1	☐	Common	CW_Flow_Main
2	☑	Common	CW_Flow_RF_P
3	☑	Common	CW_Flow_DC_A
4	☑	Common	CW_Flow_DC_B
5	☑	Common	CW_Flow_CRYO
6	☐	Common	CW_Flow_DRY
7	☐	Common	CW_Temp_Main
8	☐	Common	CW_Temp_RF_P

2.12.16 _GridInsert()

함수원형

void _GridInsert(string strObject, int nAxis, int nNum)

파라메터	strObject : 오브젝트이름. nAxis : 축번호 0 : 행방향(가로)으로 삽입 1 : 열방향(세로)으로 삽입 nNum : 삽입하고자 하는 행번호 또는 열번호
리턴형	없음
설명	그리드 오브젝트의 지정한 행 또는 열에 셀을 삽입합니다.
함수활용	특정 위치에 셀을 추가해야 하는 경우 사용합니다.

```
/*********************************************************
예제 : 특정 위치에 셀 삽입하기
*********************************************************/
void GridInsert()
{
	// x축으로 첫번째 삽입을 합니다. 현재 있던 셀은 우측으로 1칸씩 밀려납니다.
	_GridInsert("GD20@뷰페이지명", 0, 0);
	// 갱신해주어야 합니다.
	_RefreshControl("GD20@뷰페이지명");
}

void Grid()
{
	// 행1개 추가
	_GridAppend("GD20@뷰페이지명", 0);
	// 열1개 추가
	_GridAppend("GD20@뷰페이지명", 1);
	//
	// col0, row0에 문자열 표시
	_GridSetValue("GD20@뷰페이지명", 0, 0, "KOREA");
	// col1, row0에 문자열 표시
	_GridSetValue("GD20@뷰페이지명", 1, 0, "World networking");
	// col2, row0에 실수값 표시
	_GridSetValue("GD20@뷰페이지명", 2, 0, 100.5);

	// 배경색을 녹색으로 변경한다.
```

```
	_GridSetBackColor("GD20@뷰페이지명", 0, 0, _RGB(0x00, 0xFF, 0x00));
	//
	_RefreshControl("GD20@뷰페이지명");
}
```

2.12.17 _GridJoinCells()

함수원형

void _GridJoinCells(string strObject, int nCol1, int nRow1, int nCol2, int nRow2)

파라메터	strObject : 오브젝트이름. nCol1 : 시작 행번호.(0부터 시작) nRow1 : 시작 열번호. nCol2 : 마지막 행번호. nRow2 : 마지막 열번호.
리턴형	없음
설명	지정한 시작행/시작열부터 마지막행/마지막 열까지 셀을 병합합니다.
함수활용	셀의 내용이 제목 및 중요항목으로 여러개의 셀에 위치해야 하는 경우 사용합니다.

※ 그리드의 셀을 병합 후 병합된 셀의 데이터를 갖고 오기 위해 _GridGetValue()를 이용 시 행, 열의 값은 병합 시 설정한 시작 행, 시작열 번호로 설정하면 됩니다.

```
/*********************************************************
예제 : 그리드 특정셀 병합
*********************************************************/
void GridJoinCell()
{
	// "GD20" 오브젝트의 셀(1, 1)부터 셀(2,2)까지 하나의 셀로 병합합니다.
	_GridJoinCells("GD20@뷰페이지명", 1, 1, 2, 2);

	_GridSetValue("GD20@뷰페이지명", 1, 1, "Join test");
	_RefreshControl("GD20@뷰페이지명");
}
```

2.12.18 _GridLoad()

함수원형

int _GridLoad(string strObject, string strFile)

파라메터	strObject : 오브젝트이름. strFile : 경로포함 파일이름
리턴형	정수형 0 : 성공 -1 : 실패
설명	파일의 내용을 읽어서 지정된 그리드에 표시합니다.
함수활용	

※ _GridLoad() 함수는 _GridSave() 함수를 이용해서 저장된 파일만 정상적으로 Load할 수 있습니다.
또한 저장시 그리드의 CELL이 Join된 상태에 대해서는 Load시 자동으로 CELL을 Join 하지 않습니다.

```
/*********************************************************
예제 : 파일을 읽어서 그리드에 로드하기
*********************************************************/
void GridLoad()
{
	int nRet;

	// C 드라이브의 main.grid 파일을 오브젝트명이 "GD20"인 오브젝트에 불러들입니다.
	nRet = _GridLoad("GD20@뷰페이지명", "C:₩₩main.grid");

	// 파일 읽기를 실패했으면
	if(nRet != 0)
	{
		// 파일 읽기 실패 경고 메시지 박스 표시
		_MsgBox("읽기 실패", "읽기", 0, 4);

		// 아래구문은 더이상 실행하지 않고 함수를 종료합니다.
		return;
	}
	// 그리드 오브젝트명이 "그리드"를 갱신합니다.
	_RefreshControl("GD20@뷰페이지명");
}
```

2.12.19 _GridLock()

함수원형

void _GridLock(string strObject, int nCol, int nRow)

파라메터	strObject : 오브젝트이름. nCol : 행번호 (0 : 행 잠금해제) nRow : 열번호 (0 : 열 잠금해제)
리턴형	없음
설명	그리드의 지정 행, 열을 고정하고 지정된 행, 열을 기준으로 상,하,좌,우 스크롤이 되지 않도록 한다.
함수활용	특정 셀의 내용이 제목이나 타이틀인 경우에 스크롤시 제목이나 타이틀에 대해서는 스크롤이 되지 않게 하는 경우 사용한다.

```
/*********************************************************
예제 : 특정 영역 그리드 잠그기
*********************************************************/
void GridLock()
{
	// 그리드 오브젝트명이 " GD20"인 세로방향 0번열부터 5번열까지 잠금을 설정합니다.
	// 그리드 오브젝트명이 " GD20"인 가로방향은 잠금이 해제됩니다.
	_GridLock("GD20@뷰페이지명", 0, 5);

	// 그리드 오브젝트명이 " GD20"인 가로방향 0번행부터 3번행까지 잠금을 설정합니다.
	// 그리드 오브젝트명이 " GD20"인 세로방향은 잠금이 해제됩니다.
	_GridLock("GD20@뷰페이지명", 3, 0);

	_RefreshControl("GD20@뷰페이지명");
}
```

2.12.20 _GridRowCount()

함수원형

int _GridRowCount(string strObject)

파라메터	strObject : 오브젝트이름.
리턴형	열개수 정수값
설명	지정된 그리드 오브젝트의 열의 개수를 돌려줍니다.
함수활용	그리드 오브젝트의 셀을 반복하여 처리할 때마다 열의 개수가 조건에 따라 바뀌는 경우(데이터베이스 쿼리 결과에 따라 열수가 조정되는 경우, 열이 추가된 경우) 사용합니다.

```
/*********************************************************
예제 : 그리드 내용 전체 삭제하기
*********************************************************/
void GridClearRange()
{
	int nCols, nRows;

	// "GD20" 그리드 오브젝트의 행 개수
	nCols = _GridColCount("GD20@뷰페이지명")-1;
	// "GD20" 그리드 오브젝트 열 개수.
	nRows = _GridRowCount("GD20@뷰페이지명")-1;

	// 그리드 오브젝트 명이 "GD20" 인 그리드의 전체영역의 내용을 삭제합니다.
	_GridClearRange("GD20@뷰페이지명", 0, 0, nCols, nRows);

	// 그리드 오브젝트 명이 "GD20" 를 갱신합니다.
	_RefreshControl("GD20@뷰페이지명");
}
```

2.12.21 _GridSave()

함수원형

int _GridSave(string strObject, string strFile)

파라메터	strObject : 오브젝트이름. strFile : 경로포함 파일이름
리턴형	정수형 0 : 성공 -1 : 실패
설명	그리드 오브젝트의 셀 내용을 파일에 저장합니다.
함수활용	현재 그리드 오브젝트의 셀 내용을 지정한 파일로 저장할때 사용합니다.

※ _GridLoad() 함수는 _GridSave() 함수를 이용해서 저장된 파일만 정상적으로 Load할 수 있습니다.
또한 저장시 그리드의 CELL이 Join된 상태에 대해서는 Load시 자동으로 CELL을 Join 하지 않습니다.

```
/*********************************************************
예제 : 그리드 내용 파일로 저장하기
*********************************************************/
void GridSave()
{
    int nRet;

    // "GD20" 오브젝트의 내용을 C 드라이브의 main.grid 파일에 저장합니다.
    nRet = _GridSave("GD20@뷰페이지명", "C:₩₩main.grid");

    // 파일 저장이 실패했으면
    if(nRet != 0) {

        // 파일 저장 실패 경고 메시지 박스 표시
        _MsgBox("저장실패", "저장", 0, 4);

        // 아래구문은 더이상 실행하지 않고 함수를 종료합니다.
        return;
    }
}
```

2.12.22 _GridSetAlignment()

함수원형

void _GridSetAlignment(string strObject, int nCol, int nRow, int nFlag)

파라메터	strObject : 오브젝트이름. nCol : 행번호 nRow : 열번호 nFlag : 정렬방법 0 : 왼쪽 맞춤 1 : 가운데 맞춤 2 : 오른쪽 맞춤
리턴형	없음
설명	지정된 행/열의 셀에 표시되는 문자를 설정된 정렬방법으로 정렬합니다.
함수활용	데이터베이스에서 쿼리를 하거나 파일을 읽어서 표시가 된 경우 문자나 숫자 등의 정렬을 다르게 표시 하는 경우 사용합니다.

```
/*********************************************************
예제 : 그리드 특정 셀의 오른쪽 정렬하기
*********************************************************/
void GridSetAlignment()
{
	// "GD20" 그리드 오브젝트의 셀(5, 3) 텍스트를 왼쪽 맞춤으로 정렬합니다.
	_GridSetAlignment("GD20@뷰페이지명", 5, 3, 0);
	// "GD20" 그리드 오브젝트의 셀(6, 3) 텍스트를 가운데 맞춤으로 정렬합니다.
	_GridSetAlignment("GD20@뷰페이지명", 6, 3, 1);
	// "GD20" 그리드 오브젝트의 셀(7, 3) 텍스트를 오른쪽 맞춤으로 정렬합니다.
	_GridSetAlignment("GD20@뷰페이지명", 7, 3, 2);

	// "GD20"를 갱신합니다.
	_RefreshControl ("GD20@뷰페이지명");

}
```

2.12.23 _GridSetAlignmentEx()

함수원형

void _GridSetAlignmentEx(string strObject, int nCol1, int nRow1, int nCol2, int nRow2, int nFlag)

파라메터	strObject : 오브젝트이름. nCol1 : 행 시작번호 nRow1 : 열 시작번호 nCol2 : 행 마지막번호 nRow2 : 열 마지막번호 nFlag : 정렬방법 0 : 왼쪽 맞춤 1 : 가운데 맞춤 2 : 오른쪽 맞춤
리턴형	없음
설명	특정 셀의 범위를 설정하여 한번에 문자를 설정된 정렬방법으로 정렬합니다.
함수활용	데이터베이스에서 쿼리를 하거나 파일을 읽어서 표시가 된 경우 문자나 숫자 등의 정렬을 다르게 표시 하는 경우 사용합니다.

```
/*********************************************************
예제 : 그리드 특정 셀의 오른쪽 정렬하기
*********************************************************/
void GridSetAlignment()
{
	// "GD20" 그리드 오브젝트의 COL5, COL6, COL7, ROW3의 텍스트를 왼쪽 맞춤으로 정렬합니다.
	_GridSetAlignmentEx("GD20@뷰페이지명", 5, 3, 7, 3, 0);

	// "GD20"를 갱신합니다.
	_RefreshControl ("GD20@뷰페이지명");

}
```

2.12.24 _GridSetBackColor()

함수원형

void _GridSetBackColor(string strObject, int nCol, int nRow, int rgbBack)

파라메터	strObject : 오브젝트이름. nCol : 행번호 nRow : 열번호 rgbBack : 배경색
리턴형	없음
설명	그리드 오브젝트의 행(col), 열(row)의 배경색을 변경합니다.
함수활용	그리드의 특정 셀에 대해 색상을 설정하여 주기적으로 깜빡이는 효과를 보이게 하거나 현재 색상의 반전 처리를 하는 경우 사용합니다.

```
/*********************************************************
예제 : 그리드 시험
*********************************************************/
void Grid()
{
	// 행1개 추가
	_GridAppend("GD20@뷰페이지명", 0);
	// 열1개 추가
	_GridAppend("GD20@뷰페이지명", 1);
	//
	// col0, row0에 문자열 표시
	_GridSetValue("GD20@뷰페이지명", 0, 0, "KOREA");
	// col1, row0에 문자열 표시
	_GridSetValue("GD20@뷰페이지명", 1, 0, "World networking");
	// col2, row0에 실수값 표시
	_GridSetValue("GD20@뷰페이지명", 2, 0, 100.5);

	// 배경색을 녹색으로 변경한다.
	_GridSetBackColor("GD20@뷰페이지명", 0, 0, _RGB(0x00, 0xFF, 0x00));

	//  그리드 갱신
	_RefreshControl("GD20@뷰페이지명");
}
```

2.12.25 _GridSetBackColorEx()

함수원형

void _GridSetBackColorEx(string strObject, int nCol1, int nRow1, int nCol2, int nRow2, int rgbBack)

파라메터	strObject : 오브젝트이름. nCol1 : 행 시작번호 nRow1 : 열 시작번호 nCol2 : 행 마지막번호 nRow2 : 열 마지막번호 rgbBack : 배경색
리턴형	없음
설명	그리드 오브젝트의 지정된 범위의 셀 배경색을 변경합니다.
함수활용	그리드의 특정 셀 영역에 대해 색상을 설정하여 주기적으로 깜빡이는 효과를 보이게 하거나 현재 색상의 반전 처리를 하는 경우 사용합니다.

```
/*********************************************************
예제 : 그리드 시험
*********************************************************/
void Grid()
{
	int nRed, nGreen, nBlue, nColor;

	// 행1개 추가
	_GridAppend("GD20@뷰페이지명", 0);
	// 열1개 추가
	_GridAppend("GD20@뷰페이지명", 1);
	//
	// col0, row0에 문자열 표시
	_GridSetValue("GD20@뷰페이지명", 0, 0, "KOREA");
	// col1, row0에 문자열 표시
	_GridSetValue("GD20@뷰페이지명", 1, 0, "World networking");
	// col2, row0에 실수값 표시
	_GridSetValue("GD20@뷰페이지명", 2, 0, 100.5);

	nRed = 100;
	nGreen = 0;
	nBlue = 255;
	// 적색 값:100, 녹색 값:0, 파란색 값:255을 조합한 색상 값을 정수형 변수 nColor에 저장합니다.
```

```
        nColor = _RGB(nRed, nGreen, nBlue);

        // 배경색을 녹색으로 변경한다.
        _GridSetBackColorEx("GD20@뷰페이지명", 0, 0, 1, 9, nColor);

        //  그리드 갱신
        _RefreshControl("GD20@뷰페이지명");
}
```

2.12.26 _GridSetHeight()

함수원형

void _GridSetHeight(string strObject, int nRow, int nHeight)

파라메터	strObject : 오브젝트이름. nRow : 열번호 nHeight : 정수형 높이값 (픽셀)
리턴형	없음
설명	그리드 오브젝트의 특정 열(row)의 높이를 변경합니다.
함수활용	

```
/*********************************************************
예제 : 그리드 오브젝트 열높이 변경
*********************************************************/
void Grid()
{
	// 열 1개 추가
	_GridAppend("GD20@뷰페이지명", 1);
	//
	// col0, row0에 문자열 표시
	_GridSetValue("GD20@뷰페이지명", 0, 0, "KOREA");
	// col1, row0에 문자열 표시
	_GridSetValue("GD20@뷰페이지명", 1, 0, "World networking");
	// col2, row0에 실수값 표시
	_GridSetValue("GD20@뷰페이지명", 2, 0, 100.5);

	// 0번째 ROW에 대해서 높이를 50pixel로 변경합니다.
	_GridSetHeight("GD20@뷰페이지명", 0, 50);

	_RefreshControl("GD20@뷰페이지명");
}
```

2.12.27 _GridSetMetrics()

함수원형

void _GridSetMetrics(string strObject, int nCols, int nRows)

파라메터	strObject : 오브젝트이름. nCols : 조종할 행의 개수 nRows : 조종할 열의 개수
리턴형	없음
설명	그리드 오브젝트의 지정된 그리드 오브젝트의 행 또는 열의 개수를 조종합니다.
함수활용	초기 생성된 그리드의 행/열 개수를 추가/삽입 방식이 아닌 한번에 개수를 조정해야 하는 경우 사용합니다.

※ 행 열의 개수가 정해졌을 경우 개별적으로 반복해서 행,열을 추가하는 것보다 효율적입니다.

```
/*********************************************************
예제 : 그리드 행, 열 개수 조종하기
*********************************************************/
void GridSetMetrics()
{
	// "GD20" 그리드의 행개수를 10개, 열의 개수를 10개로 변경합니다.
	_GridSetMetrics("GD20@뷰페이지명", 10, 10);
	_RefreshControl("GD20@뷰페이지명");
}
```

2.12.28 _GridSetTextColor()

함수원형

void _GridSetTextColor(string strObject, int nCol, int nRow, int rgbText)

파라메터	strObject : 오브젝트이름. nCol : 행번호 nRow : 열번호 rgbText : 글씨색
리턴형	없음
설명	그리드 오브젝트의 행(col), 열(row)의 글씨색을 변경합니다.
함수활용	그리드의 특정 셀에 대해 글씨 색상을 설정하여 주기적으로 깜빡이는 효과를 보이게 하거나 현재 색상의 반전 처리를 하는 경우 사용합니다.

```
/*********************************************************
예제 : 그리드 시험
*********************************************************/
void Grid()
{
	// 행1개 추가
	_GridAppend("GD20@뷰페이지명", 0);
	// 열1개 추가
	_GridAppend("GD20@뷰페이지명", 1);
	//
	// col0, row0에 문자열 표시
	_GridSetValue("GD20@뷰페이지명", 0, 0, "KOREA");
	// col1, row0에 문자열 표시
	_GridSetValue("GD20@뷰페이지명", 1, 0, "World networking");
	// col2, row0에 실수값 표시
	_GridSetValue("GD20@뷰페이지명", 2, 0, 100.5);

	// 글씨색을 흰색으로 변경한다.
	_GridSetTextkColor("GD20@뷰페이지명", 0, 0, _RGB(0xFF, 0xFF, 0xFF));
	// 배경색을 녹색으로 변경한다.
	_GridSetBackColor("GD20@뷰페이지명", 0, 0, _RGB(0x00, 0xFF, 0x00));

	//  그리드 갱신
	_RefreshControl("GD20@뷰페이지명");
}
```

2.12.29 _GridSetTextColorEx()

함수원형

void _GridSetTextColorEx(string strObject, int nCol1, int nRow1, int nCol2, int nRow2, int rgbText)

파라메터	strObject : 오브젝트이름. nCol1 : 행 시작번호 nRow1 : 열 시작번호 nCol2 : 행 마지막번호 nRow2 : 열 마지막번호 rgbText : 글씨색
리턴형	없음
설명	그리드 오브젝트의 지정된 범위의 셀 글씨색을 일괄적으로 변경합니다.
함수활용	그리드의 특정 셀 영역에 대해 글씨색을 설정하여 주기적으로 깜빡이는 효과를 보이게 하거나 현재 색상의 반전 처리를 하는 경우 사용합니다.

```
/*********************************************************
예제 : 그리드 시험
*********************************************************/
void Grid()
{
	int nRed, nGreen, nBlue, nColor;

	// 행1개 추가
	_GridAppend("GD20@뷰페이지명", 0);
	// 열1개 추가
	_GridAppend("GD20@뷰페이지명", 1);
	//
	// col0, row0에 문자열 표시
	_GridSetValue("GD20@뷰페이지명", 0, 0, "KOREA");
	// col1, row0에 문자열 표시
	_GridSetValue("GD20@뷰페이지명", 1, 0, "World networking");
	// col2, row0에 실수값 표시
	_GridSetValue("GD20@뷰페이지명", 2, 0, 100.5);

	nRed = 100;
	nGreen = 0;
	nBlue = 255;
	// 적색 값:100, 녹색 값:0, 파란색 값:255을 조합한 색상 값을 정수형 변수 nColor에 저장합니다.
```

```
        nColor = _RGB(nRed, nGreen, nBlue);

        // 글씨색을 녹색으로 변경한다.
        _GridSetTextColorEx("GD20@뷰페이지명", 0, 0, 1, 9, nColor);

        //  그리드 갱신
        _RefreshControl("GD20@뷰페이지명");
}
```

2.12.30 _GridSetType()

함수원형

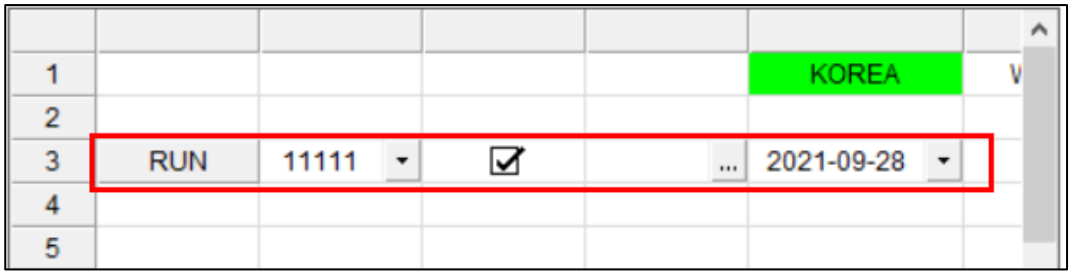
void _GridSetType(string strObject, int nCol, int nRow, string sType, string sLabel, string sFunc)

파라메터	strObject : 오브젝트이름. nCol : 행번호 nRow : 열번호 sType : 셀 종류. 5가지의 선택형태를 지원합니다. "BUTTON" : 버튼형태 "DROPLIST" : 콤보박스 "CHECKBOX" : 체크박스 "ELLIPSIS" : "..." 버튼 "DATETIME" : 날짜선택 버튼 sLsbel : 셀 내용으로 표시될 텍스트 문자열 sFunc : 사용자 호출함수 이름
리턴형	없음
설명	eRun에서 지원하는 그리드의 각셀의 형태를 설정합니다.
함수활용	특정셀에 대해서 선택박스, 콤보, 버튼, 체크박스등을 설정하여 선택사항이나 설정항목을 좀 더 자세히 설정 할 경우 사용합니다.

```
void GridSetType()
{
	//*********************************************************
	// PUSH BUTTON 셀 타입 설정 하기
	//*********************************************************
	// 셀(1,2)의 셀 타입을 BUTTON으로 설정하고 LABEL을 "RUN"으로 설정을 합니다.
	// 버튼이 CLICK되면 사용자가 정의한 스크립트 함수 "PushFunc()"를 호출
	_GridSetType("GD20@뷰페이지명", 0, 2, "BUTTON", "RUN", "PushFunc()");


	//*********************************************************
	// DROP LIST 셀 타입 설정 하기
	//*********************************************************
	// DROPLIST로 설정하고 LABEL을 "1111₩n22222"로 설정.
	// 항목이 CLICK되면 사용자가 정의한 스크립트 함수 "DropFunc()"를 호출합니다.
	// 셀 TYPE이 DROP LIST인 경우에는 LABEL에 들어가는 것이 항목리스트 이기 때문에
	// 항목간 구분자로 '₩n'를 항목과 항목 사이에 넣어주면 됩니다.
	_GridSetType("GD20@뷰페이지명", 1, 2, "DROPLIST", "11111₩n22222₩n", "DropFunc()");
```

```
	//*********************************************************
	// CHECK BOX 셀 타입 설정 하기
	//*********************************************************
	// 셀(1,2)의 셀 타입을 CHECK BOX로 설정합니다.
	// CHECK BOX는 셀의 중앙부에 오며 LABEL은 없습니다.
	// CHECK BOX를 CLICK하면 사용자가 정의한 스크립트 함수 "CheckFunc()"를 호출합니다.
	_GridSetType("GD20@뷰페이지명", 2, 2, "CHECKBOX", "", "CheckFunc()");


	//*********************************************************
	// ELLIPSIS BUTTON 셀 타입 설정 하기
	//*********************************************************

	// 셀(1,2)의 셀 타입을 ELLIPSIS BUTTON으로 설정합니다.
	// 사용자가 ELLIPSIS BUTTON을 CLICK하게 되면 사용자가
	// 정의한 스크립트 함수 "EllipsisFunc()"를 호출합니다.
	_GridSetType("GD20@뷰페이지명", 3, 2, "ELLIPSIS", "FileName", "EllipsisFunc()");


	//*********************************************************
	// DATETIME 셀 타입 설정 하기
	//*********************************************************/

	// 셀(1,2)의 셀 타입을 DATETIME으로 설정합니다.
	// 항목이 CLICK되면 사용자가 정의한 스크립트 함수 "Datetime()"를 호출합니다.
	_GridSetType("GD20@뷰페이지명", 4, 2, "DATETIME", "", "Datetime()");

	// GD20를 갱신합니다.
	_RefreshControl("GD20@뷰페이지명");
}
```

실행결과



2.12.31 _GridSetTypeEx()

함수원형

void _GridSetTypeEx(string strObject, int nCol1, int nRow1, int nCol2, int nRow2, string sType, string sLabel, string sFunc)

파라메터	strObject : 오브젝트이름. nCol1 : 시작 행번호 nRow1 : 시작 열번호 nCol2 : 마지막 행번호 nRow2 : 마지막 열번호 sType : 셀 종류. 5가지의 선택형태를 지원합니다. "BUTTON" : 버튼형태 "DROPLIST" : 콤보박스 "CHECKBOX" : 체크박스 "ELLIPSIS" : "..." 버튼 "DATETIME" : 날짜선택 버튼 sLsbel : 셀 내용으로 표시될 텍스트 문자열 sFunc : 사용자 호출함수 이름
리턴형	없음
설명	특정 셀의 범위를 설정하여 한번에 셀의 타입을 설정합니다.
함수활용	특정 셀 영역에 대해서 선택박스, 콤보, 버튼, 체크박스 등을 설정하여 선택사항이나 설정항목을 좀 더 자세히 설정하고자하는 경우 사용합니다.

※ _GridSetType()과 GridSetTypEx()의 차이점.

_GridSetType()는 보통 한 개의 셀에 대해서 셀 타입을 적용할 때 사용을 합니다.
위의 예제에서처럼 한 행에 대한 모든 열을 셀 타입을 적용할 때 _GridSetType() 함수를 사용하게 되면 반복문을 사용하여 열의 개수만큼 타입을 설정해야 합니다.
이런 경우 시스템에 부하를 주기 때문에 _GridSetType()는 한 개의 셀에 대해서만 사용하시기 바랍니다.

```
/*********************************************************
예제 : Grid 특정 셀에 타입 설정하기(확장)
*********************************************************/
void GridSetTypeEx()
{
	string strRange;
	int nMaxRow;
```

```
        strRange = "Ω₩nkΩ₩nMΩ₩n";
        // 그리드 오브젝트 명이 "그리드"인 열의 전체 개수를 정수형 변수 nMaxRow에 저장합니다.
        nMaxRow = _GridRowCount("그리드@뷰페이지명");

        // 1행의 모든 열을 DropList로 설정합니다.
        // 0부터 시작하기 때문에 nMaxRow에서 하나를 뺍니다..
        _GridSetTypeEx("그리드@뷰페이지명", 1, 0, 1, nMaxRow-1, "DROPLIST", strRange, "");
        //10행의 모든 열을 Button으로 설정합니다.
        _GridSetTypeEx("그리드@뷰페이지명", 10, 0, 10, nMaxRow-1, "BUTTON", "저장", "SpecSave()");
        //11행의 모든 열을 Button으로 설정합니다.
        _GridSetTypeEx("그리드@뷰페이지명", 11, 0, 11, nMaxRow-1, "BUTTON", "삭제", "SpecDelete()");

        // 그리드 오브젝트 명이 "그리드"를 갱신합니다.
        _RefreshControl("그리드@뷰페이지명");
}
```

2.12.32 _GridSetValue()

함수원형

void _GridSetValue(string strObject, int nCol, int nRow, int nValue)

void _GridSetValue(string strObject, int nCol, int nRow, double fValue)

void _GridSetValue(string strObject, int nCol, int nRow, string strValue)

파라메터	strObject : 오브젝트이름. nCol : 행번호 nRow : 열번호 nValue : 정수값 fValue : 실수값 strValue : 문자열
리턴형	없음
설명	지정된 그리드 오브젝트의 행, 열에 값을 설정합니다.
함수활용	

```
/*********************************************************
예제 : 특정 셀에 설정 값 표시하기
*********************************************************/
void GridSetValue()
{
	// 그리드 오브젝트 명이 "그리드"의 셀(10, 10)에 문자열 "TEST"를 설정합니다.
	_GridSetValue("그리드@뷰페이지명", 10, 10, "Test");
	// 그리드 오브젝트 명이 "그리드"의 셀(3, 5)에 실수 값 123.45를 설정합니다.
	_GridSetValue("그리드@뷰페이지명", 3, 5, 123.45);
	// 그리드 오브젝트 명이 "그리드"를 갱신합니다.
	_RefreshControl("그리드@뷰페이지명");
}
```

2.12.33 _GridSetWidth()

함수설명

void _GridSetWidth(string strObject, int nCol, int nWidth)

파라메터	strObject : 오브젝트이름. nCol : 행번호 nWidth : 정수형 넓이값 (픽셀)
리턴형	없음
설명	지정된 오브젝트의 해당 행(COL)의 넓이를 변경합니다.
함수활용	

```
/*********************************************************
예제 : 그리드 오브젝트 행넓이 변경
*********************************************************/
void Grid()
{
	// 행 1개 추가
	_GridAppend("GD20@뷰페이지명", 0);
	//
	// col0, row0에 문자열 표시
	_GridSetValue("GD20@뷰페이지명", 0, 0, "KOREA");

	// 0번째 COL에 대해서 넓이를 300pixel로 변경합니다.
	_GridSetHeight("GD20@뷰페이지명", 0, 300);

	_RefreshControl("GD20@뷰페이지명");
}
```

2.12.34 _GridUnJoinCells()

함수원형

void _GridUnJoinCells(string strObject, int nCol, int nRow)

파라메터	strObject : 오브젝트이름 nCol : 해제 시작 행번호(0부터 시작) nRow : 해제 시작 열번호
리턴형	없음
설명	지정된 오브젝트의 병합되어있는 셀의 행과 열을 병합에서 해제합니다.
함수활용	

※ 그리드의 셀을 병합 후 병합된 셀의 데이터를 갖고 오기 위해 _GridGetValue()를 이용 시 행, 열의 값은 병합 시 설정한 시작 행, 시작열 번호로 설정하면 됩니다.

```
/*********************************************************
예제 : 그리드 특정셀 병합 / 해제
*********************************************************/
void GridUnJoinCell()
{
	// "GD20" 오브젝트의 셀(1, 1)부터 셀(2,2)까지 하나의 셀로 병합합니다.
	_GridJoinCells("GD20@뷰페이지명", 1, 1, 2, 2);
	_RefreshControl("GD20@뷰페이지명");

	_MsgBox("NEXT", "", 0, 0);

	// 셀(1, 1)부터 셀(2,2)까지 병합되어 있던 셀을 분리합니다.
	_GridUnJoinCells("GD20@뷰페이지명", 1, 1);

	_GridSetValue("GD20@뷰페이지명", 1, 1, "Un join 1");
	_GridSetValue("GD20@뷰페이지명", 2, 2, "Un join 2");

	_RefreshControl("GD20@뷰페이지명");
}
```

2.13 경보 함수

2.13.1 _AlarmAck()

함수원형

int _AlarmAck(string strObject, int nRow)

파라메터	strObject : 현재경보창 이름 nRow : 열번호
리턴형	정수형 0 : 성공 -1 : 실패
설명	현재경보 발생순서에 따라 지정된 열 번호 항목을 ACK 처리합니다.
함수활용	스크립트 함수를 이용해서 현재 발생 되어있는 경보항목을 확인(ACK)처리 합니다.

```
/*********************************************************
예제 : 현재 경보 확인처리(ACK)
*********************************************************/
void AlarmAck()
{
	// 현재경보 창의 0번째 경보내용을 ACK처리합니다.
	_AlarmAck("현재경보@뷰페이지명", 0);
}
```

2.13.2 _AlarmAckAll()

함수원형

int _AlarmAckAll()

파라메터	없음
리턴형	정수형 (0)
설명	현재 발생된 모든경보를 확인(ACK) 처리합니다.
함수활용	

```
/*********************************************************
예제 : 현재 경보 확인처리(ACK)
*********************************************************/
void AlarmAckAll()
{
	// 현재경보 모두 확인.
	_AlarmAckAll();
}
```

2.13.3 _AlarmAckUser()

함수원형

void _AlarmAckUser(string strUser)

파라메터	strUser : 확인자 이름
리턴형	없음
설명	Alarm Ack시 누가 확인했는지 기록하기 위해서 사용자명을 설정할 때 사용합니다.
함수활용	경보확인자 이름을 남길 때 사용합니다.

```
/*********************************************************
예제 : 경보 확인자 변경
*********************************************************/
void AlarmAckUser()
{
	_AlarmAckUser(@ALARM.USER_NAME);
}
```

2.13.4 _AlarmCount()

함수원형

int _AlarmCount()

파라메터	없음
리턴형	정수형
설명	현재경보 발생건수를 돌려줍니다.
함수활용	

```
/*********************************************************
예제 : 경보 확인자 변경
*********************************************************/
void Alarm()
{
	@ALARM.COUNT = _AlarmCount();
	_TraceEx("Current alarm count = %d", @ALARM.COUNT);
}
```

2.13.5 _AlarmExportCSV()

함수원형

int _AlarmExportCSV(string strObject, string strFile)

파라메터	strObject : 경보 오브젝트 이름 strFile : 저장파일 이름(경보포함)
리턴형	정수형 0 : 성공 -1 : 경보창 찾을수 없음
설명	경보창 오브젝트 내용을 CSV 파일 포맷으로 저장합니다.
함수활용	TOUCH 모니터 사용할 경우 활용합니다.

※ 현재경보, 경보이력 속성 모두 사용 가능합니다.

```
/*********************************************************
예제 : 경보내용 CSV 저장
*********************************************************/
void AlarmCSV()
{
	int ret;
	string strPath, strCSV;

	// PUBLISH 폴더
	strPath = _GetProjectInfo(1);
	// CSV 파일명
	strCSV = strPath + "₩Alarm.CSV";
	_TraceEx("Project Path = %s, CSV file=%s", strPath, strCSV);

	// CSV 저장
	ret = _AlarmExportCSV("ALM30@뷰페이지명", strCSV);

	// ALM30 경보창 없으면 돌아간다.
	if(ret < 0) {
		_MsgBox("ALM30 object fail", "error", 0, 0);
		return;
	}
	// 저장폴더 탐색기 오픈.
	_Execute(strPath);
}
```

2.13.6 _AlarmPageDown()

함수원형

int _AlarmPageDown(string strObject)

파라메터	strObject : 경보 오브젝트 이름
리턴형	정수형 0 : 성공 -1 : 경보창 찾을수 없음
설명	경보창 오브젝트에서 경보수가 많을 경우 커서를 위로 스크롤해서 내용확입합니다.
함수활용	TOUCH 모니터 사용할 경우 활용합니다.

※ 현재경보, 경보이력 속성 모두 사용 가능합니다.
※ 페이지 스크롤 수는 경보창 표시개수 단위로 이동합니다.

```
/*********************************************************
예제 : 경보내용 스크롤
*********************************************************/
void AlarmUp()
{
	int ret;
	ret = _AlarmPageUp("ALARM20@뷰페이지명");
	if(ret < 0) {
		_MsgBox("ALARM20 경보창이 없습니다.", "error", 0, 0);
	}
}

void AlarmDown()
{
	int ret;
	ret = _AlarmPageDown("ALARM20@뷰페이지명");
	if(ret < 0) {
		_MsgBox("ALARM20 경보창이 없습니다.", "error", 0, 0);
	}
}
```

2.13.7 _AlarmPageUp()

함수원형

int _AlarmPageUp(string strObject)

파라메터	strObject : 경보 오브젝트 이름
리턴형	정수형 0 : 성공 -1 : 경보창 찾을수 없음
설명	경보창 오브젝트에서 경보수가 많을 경우 커서를 위로 스크롤해서 내용확입합니다.
함수활용	TOUCH 모니터 사용할 경우 활용합니다.

※ 현재경보, 경보이력 속성 모두 사용 가능합니다.
※ 페이지 스크롤 수는 경보창 표시개수 단위로 이동합니다.

```
/*********************************************************
예제 : 경보내용 스크롤
*********************************************************/
void AlarmUp()
{
	int ret;
	ret = _AlarmPageUp("ALARM20@뷰페이지명");
	if(ret < 0) {
		_MsgBox("ALARM20 경보창이 없습니다.", "error", 0, 0);
	}
}

void AlarmDown()
{
	int ret;
	ret = _AlarmPageDown("ALARM20@뷰페이지명");
	if(ret < 0) {
		_MsgBox("ALARM20 경보창이 없습니다.", "error", 0, 0);
	}
}
```

2.13.8 _AlarmScan()

함수원형

void _AlarmScan(string strObject, string strStart, string strEnd, string strFilter)

파라메터	strObject : 경보 오브젝트 이름 strStart : 검색시작 날짜 strEnd : 검색종료 날짜 strFilter : 검색조건 필터번호 문자열
리턴형	없음
설명	경보이력 검색을 하기위해서 사용합니다.
함수활용	

※ 경보이력 검색을 위해서는 경보창 속성에서 실시간이 아닌 경보이력속성으로 설정 되어있어야 합니다.
※ 마지막 파라메터의 검색필터는 문자열로 지정하고 다음과 같습니다.
"1" : 필터링 없이 모든 경보검색
"2" : 발생한 경보
"4" : 확인된 경보
"8" : 해제된 경보

```
/*********************************************************
예제 : 경보검색
*********************************************************/
void Alarm()
{
	string strStart, strEnd;

	strStart = _GetDateFormat("%Y-%m-%d %H:%M:%S", -1);
	strEnd = _GetDateFormat("%Y-%m-%d %H:%M:%S", 0);

	// 해당기간내에 발생한 모든 경보를 검색해서 "경보이력" 오브젝트에 표시합니다.
	nCount = _AlarmScan("경보이력@뷰페이지명", strStart, strEnd, "1");

	// 해당기간내에 발생한 경보중에서 확인된(ACK) 것만 "경보이력" 오브젝트에 표시합니다.
	nCount = _AlarmScan("경보이력4@뷰페이지명", strStart, strEnd, "4");
}
```

2.13.9 _AlarmScanEx()

함수원형

void _AlarmScanEx(string strObject, int nDays, string strFilter)

파라메터	strObject : 경보 오브젝트 이름 nDays : 몇일동안 발생한 경보 strFilter : 검색조건 필터번호 문자열
리턴형	없음
설명	경보이력 검색을 간편하게 하기위해서 사용합니다.
함수활용	현재날짜 시간기준으로 몇 일전부터 발생한 경보를 검색할 경우 사용합니다.

※ 경보이력 검색을 위해서는 경보창 속성에서 실시간이 아닌 경보이력속성으로 설정 되어있어야 합니다.
※ nDays는 음수를 주어야 앞의 날짜로 계산합니다.
※ 마지막 파라메터의 검색필터는 문자열로 지정하고 다음과 같습니다.
"1" : 필터링 없이 모든 경보검색
"2" : 발생한 경보
"4" : 확인된 경보
"8" : 해제된 경보

```
/*********************************************************
예제 : 경보이력 검색
*********************************************************/
void Alarm()
{
	// 10일전부터 현재시간까지 발생한 모든 경보를 검색해서 "경보이력" 오브젝트에 표시합니다.
	nCount = _AlarmScanEx("경보이력@뷰페이지명", -10, "1");

	// 30일전부터 발생한 경보중에서 확인된(ACK) 것만 "경보이력" 오브젝트에 표시합니다.
	nCount = _AlarmScanEx("경보이력@뷰페이지명", -30, "4");
```

2.15 콤보박스 함수

2.15.1 _ComboAddValue()

함수원형

void _ComboAddValue(string strObject, string strText)

void _ComboAddValue(string strObject@Viewpage, string strText)

파라메터	strObject : 뷰 페이지에 있는 콤보박스 오브젝트 이름. strObject@Viewpage : Viewpage에 있는 strObject 오브젝트 이름. strText : 콤보박스 오브젝트에 추가할 문자열을 문자열 상수 또는 변수를 지정합니다.
리턴형	없음
설명	콤보박스에 문자열을 추가합니다.
함수활용	데이터 검색조건을 콤보박스에서 선택해서 검색하는 경우 새로운 검색조건이 추가되는 경우 사용합니다.

※ 오브젝트 이름에 '@' 문자를 사용하면 @문자뒤에 있는 뷰페이지에서 오브젝트를 찾아서 지정합니다.
오브젝트 이름만 사용하는 경우는 해당 뷰페이지에서 오브젝트를 찾아서 지정합니다.

```
/*********************************************************
예제 : 콤보박스 오브젝트에 문자열 추가하기
*********************************************************/
void ComboAddValue()
{
	string strName;
	strName = "홍길동";

	// "COMBO" 오브젝트에 "Test Value" 문자열을 추가합니다.
	_ComboAddValue("COMBO@뷰페이지명", "Test Value");

	// "COMBO" 오브젝트 마지막에 "홍길동"이라고 저장된 문자열 변수 strName의 값을 추가합니다.
	_ComboAddValue("COMBO@뷰페이지명", strName);

	// 뷰페이지 "MAIN_PAGE"에서 "COMBO" 오브젝트를 찾아서 strName 문자열 값을 추가합니다.
	_ComboAddValue("COMBO@MAIN_PAGE", strName);
}
```

2.15.2 _ComboDelValue()

함수원형

void _ComboDelValue(string strObject, int nIndex)

void _ComboDelValue(string strObject@Viewpage, int nIndex)

파라메터	strObject : 뷰 페이지에 있는 콤보박스 오브젝트 이름. strObject@Viewpage : Viewpage에 있는 strObject 오브젝트 이름. nIndex : 삭제하려는 인덱스 번호 (0부터 시작)
리턴형	없음
설명	콤보박스의 지정된 항목번호의 내용을 삭제합니다.
함수활용	nIndex=0 을 설정하고 항목 개수만큼 반복호출해서 삭제한다.

※ 오브젝트 이름에 '@' 문자를 사용하면 @문자뒤에 있는 뷰페이지에서 오브젝트를 찾아서 지정합니다.
오브젝트 이름만 사용하는 경우는 해당 뷰페이지에서 오브젝트를 찾아서 지정합니다.

```
/*********************************************************
예제 : 콤보박스 오브젝트에 특정 항목 삭제하기
*********************************************************/
void ComboDelValue()
{
	// "COMBO" 오브젝트에서 첫번째 항목을 삭제합니다.
	_ComboDelValue("COMBO@뷰페이지명", 0);

	// 뷰페이지 "MAIN_PAGE"에서 "COMBO" 오브젝트를 찾아서 첫번째 항목을 삭제합니다.
	_ComboAddValue("COMBO@MAIN_PAGE", 0);
}
```

2.15.3 _ComboGetCount()

함수원형

int _ComboGetCount(string strObject)

int _ComboGetCount(string strObject@Viewpage)

파라메터	strObject : 뷰 페이지에 있는 콤보박스 오브젝트 이름. strObject@Viewpage : Viewpage에 있는 strObject 오브젝트 이름.
리턴형	정수값
설명	콤보박스의 항목개수를 돌려줍니다.
함수활용	

※ 오브젝트 이름에 '@' 문자를 사용하면 @문자뒤에 있는 뷰페이지에서 오브젝트를 찾아서 지정합니다.
오브젝트 이름만 사용하는 경우는 해당 뷰페이지에서 오브젝트를 찾아서 지정합니다.

```
/*********************************************************
예제 : 콤보박스 오브젝트에 항목 개수 출력하기
*********************************************************/
void ComboGetCount()
{
	string strMsg;
	int nCount;

	// 오브젝트 이름이 "콤보"의 항목갯수를 정수형변수 nCount에 저장합니다.
	nCount = _ComboGetCount("콤보@뷰페이지명");

	// 정수형 변수 nCount의 값과 문자를 조합하여 문자열 변수 strMsg에 저장합니다.
	strMsg = _FormatString("콤보박스 항목개수는 %d개 입니다.", nCount);

	// 문자열 변수 strMsg의 값을 메시지박스에 표시합니다.
	_MsgBox(strMsg, "콤보항목개수", 0, 2);
}
```

2.15.4 _ComboGetSel()

함수원형

int _ComboGetSel(string strObject)

int _ComboGetSel(string strObject@Viewpage)

파라메터	strObject : 뷰 페이지에 있는 콤보박스 오브젝트 이름. strObject@Viewpage : Viewpage에 있는 strObject 오브젝트 이름.
리턴형	정수값
설명	콤보박스의 선택된 항목번호를 돌려줍니다. (0부터 시작)
함수활용	콤보박스에서 항목명을 갖고 오는 경우 사용합니다.

※ 오브젝트 이름에 '@' 문자를 사용하면 @문자뒤에 있는 뷰페이지에서 오브젝트를 찾아서 지정합니다.
오브젝트 이름만 사용하는 경우는 해당 뷰페이지에서 오브젝트를 찾아서 지정합니다.

```
/*********************************************************
예제 : 콤보박스 오브젝트에 선택한 항목 번호 출력하기
*********************************************************/
void ComboGetSel()
{
    int nIndex;

     // 오브젝트 이름이 "콤보"의 현재 선택되어있는 항목의 번호를 정수형변수 nIndex에 저장합니다.
    nIndex = _ComboGetSel("콤보@뷰페이지명");

     // 정수형 변수 nIndex의 값을 메시지박스에 표시합니다.
    _MsgBox(nIndex, "선택항목", 0, 2);
}
```

2.15.5 _ComboGetValue()

함수원형

string _ComboGetValue(string strObject)

string _ComboGetValue(string strObject@Viewpage)

파라메터	strObject : 뷰 페이지에 있는 콤보박스 오브젝트 이름. strObject@Viewpage : Viewpage에 있는 strObject 오브젝트 이름.
리턴형	문자열
설명	콤보박스에서 현재 선택되어 항목을 문자열로 돌려줍니다.
함수활용	콤보박스에서 현재 선택된 문자열을 이용해서 다른 처리(데이터베이스 처리, 태그처리 등)를 해야 하는 경우 사용합니다.

※ 오브젝트 이름에 '@' 문자를 사용하면 @문자뒤에 있는 뷰페이지에서 오브젝트를 찾아서 지정합니다.
오브젝트 이름만 사용하는 경우는 해당 뷰페이지에서 오브젝트를 찾아서 지정합니다.

```
/*********************************************************
예제 : 콤보박스 오브젝트에 선택한 항목문자열 가져오기
*********************************************************/
void ComboGetValue()
{
    string strItem;

     // 오브젝트 이름이 "콤보"의 현재 선택되어있는 항목을 문자열 변수 strItem에 저장합니다.
    strItem = _ComboGetValue("콤보@뷰페이지명");

    // 문자열 변수 strMsg의 값을 메시지박스에 표시합니다.
    _MsgBox(strItem, "선택", 0, 2);
}
```

2.15.6 _ComboReset()

함수원형

void _ComboReset(string strObject)

void _ComboReset(string strObject@Viewpage)

파라메터	strObject : 뷰 페이지에 있는 콤보박스 오브젝트 이름. strObject@Viewpage : Viewpage에 있는 strObject 오브젝트 이름.
리턴형	정수값
설명	지정된 콤보박스의 내용을 모두 삭제합니다.
함수활용	현재 표시되어있는 항목들을 갱신해야 하는 경우 초기화를 할때 사용합니다.

※ 오브젝트 이름에 '@' 문자를 사용하면 @문자뒤에 있는 뷰페이지에서 오브젝트를 찾아서 지정합니다.
오브젝트 이름만 사용하는 경우는 해당 뷰페이지에서 오브젝트를 찾아서 지정합니다.

```
/*********************************************************
예제 : 콤보박스 오브젝트에 내용 모두 삭제하기
*********************************************************/
void ComboReset()
{

    // 오브젝트 이름이 "COMBOBOX14"의 모든 내용을 삭제하고 초기화 합니다.
    _ComboReset("COMBOBOX14@뷰페이지명");
}
```

2.15.7 _ComboSetSel()

함수원형

void _ComboSetSel(string strObject, int nIndex)

void _ComboSetSel(string strObject@Viewpage, int nIndex)

파라메터	strObject : 뷰 페이지에 있는 콤보박스 오브젝트 이름. strObject@Viewpage : Viewpage에 있는 strObject 오브젝트 이름. nIndex : 선택할 항목번호
리턴형	없음
설명	콤보박스에 있는 여러 항목에서 특정항목을 선택하게 합니다.
함수활용	콤보박스의 특정 항목을 미리 설정하여 콤보박스가 오브젝트가 있는 뷰페이지가 오픈될 때 자동으로 선택되게 설정해야 하는 경우 사용합니다

※ 오브젝트 이름에 '@' 문자를 사용하면 @문자뒤에 있는 뷰페이지에서 오브젝트를 찾아서 지정합니다.
오브젝트 이름만 사용하는 경우는 해당 뷰페이지에서 오브젝트를 찾아서 지정합니다.

```
/*********************************************************
예제 : 콤보박스 오브젝트에 특정 항목 선택하기
*********************************************************/
void ComboSetSel()
{
	int nSelectItem;

	nSelectItem = 3;

	// "COMBOBOX14"의 nSelectItem에 저장된 값에 해당되는 항목번호를 선택하게 합니다.
	_ComboSetSel("COMBOBOX14@뷰페이지명", nSelectItem);
}
```

2.15.8 _ComboSetValue()

함수원형

void _ComboSetValue(string strObject, string strText)

void _ComboSetValue(string strObject@Viewpage, string strText)

파라메터	strObject : 뷰 페이지에 있는 콤보박스 오브젝트 이름. strObject@Viewpage : Viewpage에 있는 strObject 오브젝트 이름. strText : 입력창에 표시할 문자열
리턴형	없음
설명	콤보박스 속성이 입력가능하도록 설정된 경우 입력부분에 문자열을 출력합니다.
함수활용	콤보박스에서 현재 선택된 문자열을 이용해서 다른 처리(데이터베이스 처리, 태그처리 등)를 해야 하는 경우 사용합니다.

※ 오브젝트 이름에 '@' 문자를 사용하면 @문자뒤에 있는 뷰페이지에서 오브젝트를 찾아서 지정합니다.
오브젝트 이름만 사용하는 경우는 해당 뷰페이지에서 오브젝트를 찾아서 지정합니다.

※ 콤보박스의 속성에서 "입력가능 콤보"부분이 설정되어 있어야만 사용할 수 있습니다.

```
/*********************************************************
예제 : 콤보박스 입력창에 문자열 표시하기
*********************************************************/
void ComboSetValue()
{
	string strMsg;
	int nSelectItem;

	strMsg = "입력창의 문자열";

	// 콤보오브젝트 strMsg 값을 표시합니다.
	_ComboSetValue("COMBOBOX14@뷰페이지명", strMsg);

	// "COMBOBOX14@Viewpage3"에 "Sample" 문자열을 표시합니다.
	_ComboSetValue("COMBOBOX14@Viewpage3", "Sample");
}
```

2.16 리스트박스 함수

2.16.1 _ListAddValue()

함수원형

void _ListAddValue(string strObject, string strText)

파라메터	strObject : 뷰 페이지에 있는 리스트박스 오브젝트 이름. strText : 추가 문자열.
리턴형	없음
설명	리스트박스에 문자열을 추가합니다.
함수활용	사용자 처리 메시지 표시가 필요한 경우 사용합니다.

```
/*********************************************************
예제 : 리스트박스에 문자열 쓰기
*********************************************************/
void ListAddValue()
{
	// 리스트박스 오브젝트 명이 "리스트박스"의 마지막에 문자열 "New"를 추가합니다.
	_ListAddValue("리스트박스@뷰페이지명", "New");
}

// 로그표시 함수
void ShowLog(string msg)
{
	int nIndex;
	string strObject, str;
	strObject = "LOGLIST@메인페이지";
	// 리스트박스의 현재 리스트 개수가 50개가 넘으면 제일 위에 있는 리스트 하나 삭제.
	nIndex = _ListGetCount(strObject);
	if(nIndex > 50)
		_ListDelValue(strObject, 0);
	str = _FormatString("%s > %s", _TimeToString("%Y-%m-%d %H:%M:%S"), msg);
// 로그 텍스트 리스트박스에 추가 표시.
	_ListAddValue(strObject, str);
	nIndex = _ListGetCount(strObject);
	_ListSetSel(strObject, nIndex-1);
}
```

2.16.2 _ListDelValue()

함수원형

void _ListDelValue(string strObject, int nIndex)

파라메터	strObject : 뷰 페이지에 있는 리스트박스 오브젝트 이름. nIndex : 삭제할 항목번호 (0부터 시작)
리턴형	없음
설명	리스트박스 오브젝트 목록에서 지정된 인덱스 번호에 해당하는 항목을 삭제합니다.
함수활용	리스트 항목에서 필요하지 않은 항목을 삭제할 때 사용합니다.

```
/*********************************************************
예제 : 리스트박스에 첫 번째 항목 삭제하기
*********************************************************/
void ListDelValue()
{
	// 리스트박스 오브젝트 명이 "리스트박스"의 첫 번째 항목을 삭제합니다.
	_ListDelValue("리스트박스@뷰페이지명", 0);
}

// 로그표시 함수
void ShowLog(string msg)
{
	int nIndex;
	string strObject, str;

	strObject = "LOGLIST@메인페이지";

	// 리스트박스의 현재 리스트 개수가 50개가 넘으면 제일 위에 있는 리스트 하나 삭제.
	nIndex = _ListGetCount(strObject);
	if(nIndex > 50)
		_ListDelValue(strObject, 0);

	str = _FormatString("%s > %s", _TimeToString("%Y-%m-%d %H:%M:%S"), msg);
// 로그 텍스트 리스트박스에 추가 표시.
	_ListAddValue(strObject, str);

	nIndex = _ListGetCount(strObject);
	_ListSetSel(strObject, nIndex-1);
}
```

2.16.3 _ListFindFile()

함수원형

void _ListFindFile(string strObject, string strFolder, string strFileExt, int nFlag)

파라메터	strObject : 리스트콤포넌트 오브젝트 이름 strFolder : 표시하고자 하는 폴더이름 strFileExt : 확장자 필터. "*.xls" : 엑셀파일만 표시 "*.*" : 모든파일 표시 nFlag : 경로명 표시 할 것인지 설정 0 : 경로포함 1 : 파일명만
리턴형	없음
설명	지정 폴더에 있는 파일 및 폴더명을 모두 표시합니다
함수활용	

```
/*********************************************************
예제 : 폴더 파일리스트
*********************************************************/
void FindFile()
{
	// C드라이브의 모든파일을 경로포함해서 표시
	_ListFindFile("LISTBOX22@뷰페이지명", "c:₩₩", "*.*", 0);
	_MsgBox("next", "", 0, 0);

	// 리스트 모두 삭제.
	_ListReset("LISTBOX22@뷰페이지명");

	// C드라이브의 모든파일을 파일만 표시
	_ListFindFile("LISTBOX22@뷰페이지명", "c:₩₩", "*.txt", 1);

	// MAIN_PAGE 뷰페이지에 있는 "LB10" 오브젝트에 e:₩₩temp 폴더에 있는 모든 폴더, 파일명 파일을 경로명 포함해서 모두 표시한다.
	_ListFindFile("LB10@MAIN_PAGE", "e:₩₩temp", "*.*", 0);
}
```

2.16.4 _ListGetCount()

함수원형

int _ListGetCount(string strObject)

파라메터	strObject : 뷰 페이지에 있는 리스트박스 오브젝트 이름.
리턴형	정수값
설명	리스트박스 오브젝트의 전체 항목 개수를 돌려줍니다.
함수활용	리스트박스의 항목 개수만큼 반복을 하면서 항목명을 갖고 와야 하는 경우 사용합니다.

```
/*********************************************************
예제 : 리스트박스의 전체 항목개수 표시하기
*********************************************************/
void ListGetCount()
{
int nCount;
	// 리스트박스 오브젝트 명이 "리스트박스"의 항목개수를 정수형변수 nCount에 저장합니다.
	nCount = _ListGetCount("리스트박스@뷰페이지명");

	// 정수형 변수 nCount의 값을 메시지박스에 표시합니다.
	_MsgBox(nCount, "전체개수", 0, 0);
}

// 로그표시 함수
void ShowLog(string msg)
{
	int nIndex;
	string strObject, str;

	strObject = "LOGLIST@메인페이지";

	// 리스트박스의 현재 리스트 개수가 50개가 넘으면 제일 위에 있는 리스트 하나 삭제.
	nIndex = _ListGetCount(strObject);
	if(nIndex > 50)
		_ListDelValue(strObject, 0);

	str = _FormatString("%s > %s", _TimeToString("%Y-%m-%d %H:%M:%S"), msg);
// 로그 텍스트 리스트박스에 추가 표시.
	_ListAddValue(strObject, str);
```

```
        nIndex = _ListGetCount(strObject);
        _ListSetSel(strObject, nIndex-1);
}
```

2.16.5 _ListGetSel()

함수원형

int _ListGetSel(string strObject)

파라메터	strObject : 뷰 페이지에 있는 리스트박스 오브젝트 이름.
리턴형	정수값
설명	리스트박스 오브젝트에서 현재 선택된 항목의 순번을 돌려줍니다. (0부터 시작합니다.)
함수활용	리스트박스에서 항목명을 갖고 오는 경우 사용합니다.

```
/*********************************************************
예제 : 리스트박스에 선택한 항목 순번 표시하기
*********************************************************/
void ListGetSel()
{
	int nCount;

	// "리스트박스" 오브젝트의 현재 선택한 항목의 번호를 nCount에 돌려줍니다.
	nCount = _ListGetSel("리스트박스@뷰페이지명");
	// 정수형 변수 nCount의 값을 메시지박스에 표시합니다.
	_MsgBox(nCount, "순번", 0, 0);
}

/*********************************************************
예제2 : 로그표시 함수
*********************************************************/
void ShowLog(string msg)
{
	int nIndex;
	string strObject, str;

	strObject = "LOGLIST@메인페이지";

	// 리스트박스의 현재 리스트 개수가 50개가 넘으면 제일 위에 있는 리스트 하나 삭제.
	nIndex = _ListGetCount(strObject);
	if(nIndex > 50)
		_ListDelValue(strObject, 0);

	str = _FormatString("%s > %s", _TimeToString("%Y-%m-%d %H:%M:%S"), msg);
// 로그 텍스트 리스트박스에 추가 표시.
```

```
        _ListAddValue(strObject, str);

        nIndex = _ListGetCount(strObject);
        _ListSetSel(strObject, nIndex-1);
}
```

2.16.6 _ListGetValue()

함수원형

string _ListGetValue(string strObject, int nIndex)

파라메터	strObject : 뷰 페이지에 있는 리스트박스 오브젝트 이름. nIndex : 항목번호 (0부터 시작).
리턴형	문자열
설명	리스트박스 오브젝트에서 인덱스 번호에 해당하는 항목을 문자열 형태로 돌려줍니다.
함수활용	리스트박스에서 현재 선택한 인덱스 번호의 문자열을 이용해서 다른 처리(데이터베이스 처리, 태그처리 등)를 해야 하는 경우 사용합니다.

```
/*********************************************************
예제 : 리스트박스에 두 번째 항목 내용 표시하기
*********************************************************/
void ListGetValue()
{
	string strValue;
	// "리스트박스" 오브젝트의 두 번째 항목의 값을 문자열 변수 strValue에 저장합니다.
	strValue = _ListGetValue("리스트박스@뷰페이지명", 1);

	// 문자열 변수 strValue의 값을 메시지박스에 표시합니다.
	_MsgBox(strValue, "내용", 0, 0);
}

// 로그표시 함수
void ShowLog(string msg)
{
	int nIndex;
	string strObject, str;

	strObject = "LOGLIST@메인페이지";

	// 리스트박스의 현재 리스트 개수가 50개가 넘으면 제일 위에 있는 리스트 하나 삭제.
	nIndex = _ListGetCount(strObject);
	if(nIndex > 50)
		_ListDelValue(strObject, 0);

	str = _FormatString("%s > %s", _TimeToString("%Y-%m-%d %H:%M:%S"), msg);
// 로그 텍스트 리스트박스에 추가 표시.
```

```
        _ListAddValue(strObject, str);

        nIndex = _ListGetCount(strObject);
        _ListSetSel(strObject, nIndex-1);
}
```

2.16.7 _ListReset()

함수원형

void _ListReset(string strObject)

파라메터	strObject : 뷰 페이지에 있는 리스트박스 오브젝트 이름.
리턴형	없음
설명	해당 리스트박스 오브젝트의 내용을 모두 삭제합니다.
함수활용	현재 표시되어있는 항목들을 갱신해야 하는 경우 초기화를 시킬 때 사용합니다.

```
/*********************************************************
예제 : 리스트박스의 내용 모두 지우기
*********************************************************/
void ListReset()
{
	// "리스트박스" 오브젝트의 모든 내용을 삭제하고 초기화 합니다.
_ListReset("리스트박스@뷰페이지명");
}

/*********************************************************
예제2 : 로그표시 함수
사용자 정의 메시지를 리스트박스에 추가해서 표시 합니다.
항목의 개수가 50개를 넘어가면 처음 항목을 삭제 합니다.
마지막 항목을 선택표시 합니다.
*********************************************************/
void ShowLog(string msg)
{
	int nIndex;
	string strObject, str;

	strObject = "LOGLIST@메인페이지";

	// 현재 리스트 개수가 50개가 넘으면 리스트박스 내용을 모두 삭제합니다.
	nIndex = _ListGetCount(strObject);
	if(nIndex > 50) {
		_ListReset(strObject);
}

	str = _FormatString("%s > %s", _TimeToString("%Y-%m-%d %H:%M:%S"), msg);
// 로그 텍스트 리스트박스에 추가 표시.
```

```
        _ListAddValue(strObject, str);

        nIndex = _ListGetCount(strObject);
        _ListSetSel(strObject, nIndex-1);
}
```

2.16.8 _ListSetSel()

함수원형

void _ListSetSel(string strObject, int nIndex)

파라메터	strObject : 뷰 페이지에 있는 리스트박스 오브젝트 이름. nIndex : 항목번호 (0부터 시작)
리턴형	없음
설명	리스트박스 오브젝트에서 설정한 인덱스 번호에 해당하는 항목을 선택합니다.
함수활용	특정 항목을 선택하거나 뷰페이지가 오픈시 특정 항목을 자동으로 선택하게 하는 경우 사용합니다.

```
/*********************************************************
예제1 : 리스트박스의 세 번째 항목 선택하기
*********************************************************/
void ListSetSel()
{
	// "리스트박스" 오브젝트의 세번째 항목을 선택합니다.
_ListSetSel("리스트박스@뷰페이지명", 2);
}

/*********************************************************
예제2 : 로그표시 함수
사용자 정의 메시지를 리스트박스에 추가해서 표시 합니다.
항목의 개수가 50개를 넘어가면 처음 항목을 삭제 합니다.
마지막 항목을 선택표시 합니다.
*********************************************************/
void ShowLog(string msg)
{
	int nIndex;
	string strObject, str;

	strObject = "LOGLIST@메인페이지";

	// 리스트박스의 현재 리스트 개수가 50개가 넘으면 제일 위에 있는 리스트 하나 삭제.
	nIndex = _ListGetCount(strObject);
	if(nIndex > 50)
		_ListDelValue(strObject, 0);

	str = _FormatString("%s > %s", _TimeToString("%Y-%m-%d %H:%M:%S"), msg);
```

```
// 로그 텍스트 리스트박스에 추가 표시.
        _ListAddValue(strObject, str);

        nIndex = _ListGetCount(strObject);
        _ListSetSel(strObject, nIndex-1);
}
```

2.17 리스트콘트롤 함수

2.17.1 _SFListAlign()

함수원형

void _SFListAlign(string strObject, int nCol, int nFlag)

파라메터	strObject : 오브젝트이름. nCol : 행번호 (가로방향의 셀 위치를 나타냅니다.) nFlag : 정렬방법 0 : 왼쪽 정렬 1 : 중앙 정렬(기본) 2 : 오른쪽 정렬
리턴형	없음
설명	라인그래프 오브젝트에 펜을 보이거나 숨깁니다.
함수활용	

```
/*********************************************************
예제 : 리스트컨트롤의 특정 셀 오른쪽 정렬하기
*********************************************************/
void SFListAlign()
{
	//  "SFLIST1" 오브젝트의 5행의 내용을 우측 정렬을 합니다.
	_SFListAlign("SFLIST1@뷰페이지명", 5, 2);

	// "SFLIST1" 오브젝트를 갱신합니다.
	_RefreshControl("SFLIST1@뷰페이지명");
}
```

2.17.2 _SFListAppend()

함수원형

void _SFListAppend(string strObject, int nCount)

파라메터	strObject : 오브젝트이름. nCount : 추가하고자 열의 개수
리턴형	없음
설명	리스트콘트롤 오브젝트에 열을 nCount만큼 추가합니다.
함수활용	

```
/*********************************************************
예제 : 리스트컨트롤 열추가
*********************************************************/
void SFListAppend()
{
	// " LC20" 오브젝트에 10개의 열을 추가합니다.
	_SFListAppend("LC20@뷰페이지명", 10);
}
```

2.17.3 _SFListClearRange()

함수원형

void _SFListClearRange(string strObject, int nCol1, int nRow1, int nCol2, int nRow2)

파라메터	strObject : 오브젝트이름. nCol1 : 시작 행번호 nRow1 : 시작 열번호 nCol2 : 마지막 행번호 nRow2 : 마지막 열번호
리턴형	없음
설명	리스트콘트롤 시작행, 열부터 마지막 행,열의 내용을 모두 지웁니다. 그러나 배경색은 그대로 남아있습니다.
함수활용	내용을 반복해서 변경하는 경우 이전에 표시된 데이터의 행/열 갯수가 다른 경우 설정된 영역을 공란을 만들고 다른 데이터를 표시할 때 사용합니다.

```
/*********************************************************
예제 : 리스트컨트롤 셀내용 지우기
*********************************************************/
void SFList()
{
	// 열 10개 추가
	_SFListAppend("LC20@뷰페이지명", 10);
	//
	// col0, row0에 문자열 표시
	_SFListSetValue("LC20@뷰페이지명", 0, 0, "KOREA");

	// col1, row0에 문자열 표시, 배경색(빨강), 글씨색(흰색)
	_SFListSetValueEx("LC20@뷰페이지명", 1, 0, "World networking", _RGB(255,0,0), 0xFFFFFF);

	_MsgBox("NEXT", "", 0, 0);

	// col1, row0에 문자열 표시, 배경색(파랑), 글씨색(흰색)
	_SFListSetValueEx("LC20@뷰페이지명", 1, 0, "World networking", 0xFF0000, 0xFFFFFF);

	// col2, row0에 실수값 표시
	_SFListSetValue("LC20@뷰페이지명", 2, 0, 100.5);

	_MsgBox("NEXT", "", 0, 0);
```

```
        // 내용은 모두 지우지만, 배경색등은 지워지지 않고 남는다.
        _SFListClearRange("LC20@뷰페이지명", 0, 0, 2, 0);

        // 리스트콘트롤 함수는 갱신함수 호출할 필요 없습니다.
        //_RefreshControl("LC20@뷰페이지명");
}
```

2.17.4 _SFListColCount()

함수원형

int _SFListColCount(string strObject)

파라메터	strObject : 오브젝트이름.
리턴형	정수값
설명	리스트콘트롤 행개수를 읽어옵니다.
함수활용	

```
/*********************************************************
예제 : 리스트컨트롤
*********************************************************/
void SFList()
{
	string str1, str2, str3;
	int nCols, nRows;

	_SFListAppend("LC20@뷰페이지명", 10);
	// col0, row0에 문자열 표시
	_SFListSetValue("LC20@뷰페이지명", 0, 0, "KOREA");
	// col1, row0에 문자열 표시
	_SFListSetValueEx("LC20@뷰페이지명", 1, 0, "World networking", _RGB(255,0,0), 0xFFFFFF);
	// col2, row0에 실수값 표시
	_SFListSetValue("LC20@뷰페이지명", 2, 0, 100.5);

	// 설정된 값 읽어서 TRACE창에 표시
	str1 = _SFListGetValue("LC20@뷰페이지명", 0, 0);
	str2 = _SFListGetValue("LC20@뷰페이지명", 1, 0);
	str3 = _SFListGetValue("LC20@뷰페이지명", 2, 0);
	_TraceEx("(0,0):%s, (1,0):%s, (2,0):%s", str1, str2, str3);

	nCols = _SFListColCount("LC20@뷰페이지명");
	nRows = _SFListRowCount("LC20@뷰페이지명");
	_TraceEx("cols = %d, rows = %d", nCols, nRows);
}
```

2.17.5 _SFListCSVIn()

함수원형

void _SFListCSVIn(string strObject, string strFilename)

파라메터	strObject : 오브젝트이름. strFilename : CSV 형식의 텍스트파일
리턴형	정수값 0 : 성공 -1 : 실패
설명	CSV (Comma Separated Values) 형식의 텍스트 파일을 리스트컨트롤 오브젝트에 읽어서 표시합니다.
함수활용	오브젝트에 헤더가 있으면 CSV파일의 첫번째 데이터는 헤더처리로 넘어가고, 헤더가 없을 경우는 CSV파일의 첫번째 데이터를 내용으로 표시합니다.

※ 함수 실행후 오브젝트 갱신하기 위한 함수 _RefreshControl() 호출할 필요 없습니다.
함수 실행후 포커스가 오브젝트 상단 첫번째 셀로 이동하면서 갱신 됩니다.

```
/*********************************************************
예제 : C:₩₩Test.csv 파일 리스트컨트롤에 표시하기
*********************************************************/

"Test.csv" 파일내용
Item,Load 1,Load 2,Load 3,
0Kg,1.00000,100,200,
UP1,2.00000,300,400,
UP2,3.00000,500,600,
UP3,4.00000,700,800,

void SFListCSVIn()
{
    int ret;
    ret = _SFListCSVIn("LC20@뷰페이지명", "C:₩₩Test.csv");

    _TraceEx("_SFListCSVIn().....%d", ret);
}
```

2.17.6 _SFListCSVOut()

함수원형

void _SFListCSVOut(string strObject, int nCol1, int nRow1, int nCol2, int nRow2, string strFile)

파라메터	strObject : 오브젝트이름. nCol1 : 시작 행번호. (0부터 시작) nRow1 : 시작 열번호. nCol2 : 마지막 행번호. nRow2 : 마지막 열번호. strFile : CSV 형식의 텍스트 파일
리턴형	정수값 0 : 성공 -1 : 실패
설명	리스트컨트롤 오브젝트의 지정된 범위의 셀데이터를 지정한 파일에 CSV(Comma Separated Values) 형식의 텍스트파일로 저장합니다.
함수활용	현재 리스트컨트롤의 데이터를 백업 및 참고사항으로 CSV파일로 저장을 해야 하는 경우 사용합니다. 헤더가 정의되어있으면 헤더내용까지 출력합니다.

※ 함수 실행후 오브젝트 갱신하기 위한 함수 _RefreshControl() 호출할 필요 없습니다.
함수 실행후 포커스가 오브젝트 상단 첫번째 셀로 이동하면서 갱신 됩니다.

```
/*********************************************************
예제 : 리스트컨트롤의 부분내용을 CSV 파일로 저장
*********************************************************/
void SFListCSVOut()
{
	int nCols, nRows;
	string strFile;
	strFile = "C:₩₩RawData.csv";
	// "LC20" 오브젝트의 행 개수. 행번호는 0부터 시작하기 때문에 개수에서 하나를 빼 줍니다.
	nCols = _SFListColCount("LC20@뷰페이지명")-1;
	// "LC20" 오브젝트의 열 개수.
	// 행번호는 0부터 시작하기 때문에 갯수에서 하나를 빼줍니다.
	nRows = _SFListRowCount("LC20@뷰페이지명")-1;
	// "LC20" 리스트컨트롤의 셀 전체의 내용을 파일에 저장합니다.
	_SFListCSVOut("LC20@뷰페이지명", 0, 0, nCols, nRows, strFile);
}
```

2.17.7 _SFListDelete()

함수원형

void _SFListDelete(string strObject, int nRow1, int nRow2)

파라메터	strObject : 오브젝트이름. nRow1 : 삭제 시작할 열번호 (0부터 시작) nRow2 : 삭제 마지막 열번호
리턴형	없음
설명	지정된 리스트컨트롤 오브젝트의 시작열부터 마지막열까지 삭제합니다.
함수활용	총 열 갯수 고정된 리스트컨트롤에서 추가 또는 삽입으로 열의 갯수를 맞춰야 하는 경우 사용합니다.

```
/*******************************************************
예제 : 리스트컨트롤 셀변경
*******************************************************/
void SFList()
{
	// 열 10개 추가
	_SFListAppend("LC20@뷰페이지명", 10);
	//
	// col0, row0에 문자열 표시
	_SFListSetValue("LC20@뷰페이지명", 0, 0, "KOREA");
	// col1, row0에 문자열 표시
	_SFListSetValue("LC20@뷰페이지명", 1, 0, "World networking");
	// col2, row0에 실수값 표시
	_SFListSetValue("LC20@뷰페이지명", 2, 0, 100.5);

	_MsgBox("NEXT", "", 0, 0);
	//
	_SFListDelete("LC20@뷰페이지명", 0, 0);
}
```

2.17.8 _SFListFind()

함수원형

void _SFListFind(string strObject, int nCol, string strText)

파라메터	strObject : 오브젝트이름. nCol : 문자열 검색할 행번호 strText : 검색할 문자열
리턴형	정수형 -1 : 실패 0보다 크거나 같으면 찾은 열번호
설명	리스트컨트롤 오브젝트의 해당 행에서 찾고자 하는 문자열을 검색하여 문자열 위치(열번호)를 돌려줍니다.
함수활용	데이터베이스 쿼리 결과를 리스트컨트롤에 표시한 경우 쿼리 결과내에서 특정 문자열이 있는지 검사해야 하는 경우 사용합니다.

```
/*********************************************************
예제 : 문자열 찾기
*********************************************************/
void SFList()
{
	int nFindRow;

	// 열 10개 추가
	_SFListAppend("LC20@뷰페이지명", 10);

	// col0, row0에 문자열 표시
	_SFListSetValue("LC20@뷰페이지명", 0, 0, "KOREA");
	// col1, row0에 문자열 표시
	_SFListSetValue("LC20@뷰페이지명", 1, 0, "World networking");
	// col2, row0에 실수값 표시
	_SFListSetValue("LC20@뷰페이지명", 2, 0, 100.5);

	_MsgBox("NEXT", "", 0, 0);
	// 0번 col에서 "KOREA" 문자열을 찾아서 그 행번호를 돌려줍니다.
	nFindRow = _SFListFind("LC20@뷰페이지명", 0, "KOREA");
	_TraceEx("Find text 'KOREA', row number = %d", nFindRow);
}
```

2.17.9 _SFListGetEndRow()

함수원형

int _SFListGetEndRow(string strObject)

파라메터	strObject : 오브젝트이름.
리턴형	정수값
설명	리스트컨트롤의 선택범위를 지정했을 때 끝나는 열번호를 돌려줍니다.
함수활용	

```
/*********************************************************
예제 : 리스트컨트롤
*********************************************************/
void SFList()
{
	string str1, str2, str3;
	int nCols, nRows, nRow1, nRow2;

	_SFListAppend("LC20@뷰페이지명", 10);

	// col0, row0에 문자열 표시
	_SFListSetValue("LC20@뷰페이지명", 0, 0, "KOREA");
	// col1, row0에 문자열 표시
	_SFListSetValueEx("LC20@뷰페이지명", 1, 0, "World networking", _RGB(255,0,0), 0xFFFFFF);
	// col2, row0에 실수값 표시
	_SFListSetValue("LC20@뷰페이지명", 2, 0, 100.5);

	// 설정된 값 읽어서 TRACE창에 표시
	str1 = _SFListGetValue("LC20@뷰페이지명", 0, 0);
	str2 = _SFListGetValue("LC20@뷰페이지명", 1, 0);
	str3 = _SFListGetValue("LC20@뷰페이지명", 2, 0);
	_TraceEx("(0,0):%s, (1,0):%s, (2,0):%s", str1, str2, str3);

	nRow1 = _SFListGetStartRow("LC20@뷰페이지명");
	nRow2 = _SFListGetEndRow("LC20@뷰페이지명");
	_TraceEx("Range row1 =%d, Range row2 = %d", nRow1, nRow2);

}
```

2.17.10 _SFListGetStartRow()

함수원형

int _SFListGetStartRow(string strObject)

파라메터	strObject : 오브젝트이름.
리턴형	정수값
설명	리스트컨트롤의 선택범위를 지정했을 때 시작지점의 열번호를 돌려줍니다.
함수활용	

```
/*********************************************************
예제 : 리스트컨트롤
*********************************************************/
void SFList()
{
	string str1, str2, str3;
	int nCols, nRows, nRow1, nRow2;

	_SFListAppend("LC20@뷰페이지명", 10);
	// col0, row0에 문자열 표시
	_SFListSetValue("LC20@뷰페이지명", 0, 0, "KOREA");
	// col1, row0에 문자열 표시
	_SFListSetValueEx("LC20@뷰페이지명", 1, 0, "World networking", _RGB(255,0,0), 0xFFFFFF);
	// col2, row0에 실수값 표시
	_SFListSetValue("LC20@뷰페이지명", 2, 0, 100.5);

	// 설정된 값 읽어서 TRACE창에 표시
	str1 = _SFListGetValue("LC20@뷰페이지명", 0, 0);
	str2 = _SFListGetValue("LC20@뷰페이지명", 1, 0);
	str3 = _SFListGetValue("LC20@뷰페이지명", 2, 0);
	_TraceEx("(0,0):%s, (1,0):%s, (2,0):%s", str1, str2, str3);

	nRow1 = _SFListGetStartRow("LC20@뷰페이지명");
	nRow2 = _SFListGetEndRow("LC20@뷰페이지명");
	_TraceEx("Range row1 =%d, Range row2 = %d", nRow1, nRow2);

}
```

2.17.11 _SFListGetValue()

함수원형

string _SFListGetValue(string strObject, int nCol, int nRow)

파라메터	strObject : 오브젝트이름. nCol : 행번호 nRow : 열번호
리턴형	문자열
설명	리스트콘트롤 행(col), 열(row)의 값을 문자열로 읽어옵니다.
함수활용	

```
/*********************************************************
예제 : 리스트컨트롤
*********************************************************/
void SFList()
{
	string str1, str2, str3;
	_SFListAppend("LC20@뷰페이지명", 10);
	// col0, row0에 문자열 표시
	_SFListSetValue("LC20@뷰페이지명", 0, 0, "KOREA");
	// col1, row0에 문자열 표시
	_SFListSetValueEx("LC20@뷰페이지명", 1, 0, "World networking", _RGB(255,0,0), 0xFFFFFF);
	// col2, row0에 실수값 표시
	_SFListSetValue("LC20@뷰페이지명", 2, 0, 100.5);
	// 설정된 값 읽어서 TRACE창에 표시
	str1 = _SFListGetValue("LC20@뷰페이지명", 0, 0);
	str2 = _SFListGetValue("LC20@뷰페이지명", 1, 0);
	str3 = _SFListGetValue("LC20@뷰페이지명", 2, 0);
	_TraceEx("(0,0):%s, (1,0):%s, (2,0):%s", str1, str2, str3);
	_MsgBox("NEXT", "", 0, 0);
	_SFListClearRange("LC20@뷰페이지명", 0, 0, 2, 0);
	// 호출할 필요 없습니다.
	//_RefreshControl("LC20@뷰페이지명");
}
```

실행경과

[2021-10-25 10:09:36.506] (0,0):KOREA, (1,0):World networking, (2,0):100.500000

2.17.12 _SFListGoto()

함수원형

void _SFListGoto(string strObject, int nRow)

파라메터	strObject : 오브젝트이름. nRow : 이동하고자 하는 열번호
리턴형	없음
설명	리스트컨트롤 오브젝트의 지정한 위치로 선택바를 이동시킵니다.
함수활용	상황 및 조건에 따라 리스트컨트롤의 셀의 특정위치로 선택바를 옮겨야 하는 경우 사용 합니다.

```
/*********************************************************
예제 : 특정 셀로 이동하기
*********************************************************/
void SFList()
{
	int cnt;
	string strText;
	string str1, str2, str3;

	_SFListAppend("LC20@뷰페이지명", 10);

	cnt =0;
	while(cnt < 10) {
		strText = _FormatString("no%d", cnt);
		_SFListSetValue("LC20@뷰페이지명", 0, cnt, strText);

		strText = _FormatString("loop %d", cnt);
		_SFListSetValueEx("LC20@뷰페이지명", 1, cnt, strText, _RGB(255,0,0), 0xFFFFFF);

		_SFListSetValue("LC20@뷰페이지명", 2, cnt, _Rand() % 1000);

		cnt++;
	}
	// 설정된 값 읽어서 TRACE창에 표시
	str1 = _SFListGetValue("LC20@뷰페이지명", 0, 0);
	str2 = _SFListGetValue("LC20@뷰페이지명", 1, 0);
	str3 = _SFListGetValue("LC20@뷰페이지명", 2, 0);
	_TraceEx("(0,0):%s, (1,0):%s, (2,0):%s", str1, str2, str3);
```

```
        _MsgBox("NEXT", "", 0, 0);
        //
        _SFListGoto("LC20@뷰페이지명", 5);
}
```

실행결과

	name	comment	Value
1	no0	loop 0	938
2	no1	loop 1	74
3	no2	loop 2	656
4	no3	loop 3	431
5	no4	loop 4	137
6	no5	loop 5	860
7	no6	loop 6	629
8	no7	loop 7	146
9	no8	loop 8	977

2.17.13 _SFListRowCount()

함수원형

int _SFListRowCount(string strObject)

파라메터	strObject : 오브젝트이름.
리턴형	정수값
설명	리스트콘트롤 열개수를 읽어옵니다.
함수활용	

```
/*********************************************************
예제 : 리스트컨트롤
*********************************************************/
void SFList()
{
	string str1, str2, str3;
	int nCols, nRows;

	_SFListAppend("LC20@뷰페이지명", 10);
	// col0, row0에 문자열 표시
	_SFListSetValue("LC20@뷰페이지명", 0, 0, "KOREA");
	// col1, row0에 문자열 표시
	_SFListSetValueEx("LC20@뷰페이지명", 1, 0, "World networking", _RGB(255,0,0), 0xFFFFFF);
	// col2, row0에 실수값 표시
	_SFListSetValue("LC20@뷰페이지명", 2, 0, 100.5);

	// 설정된 값 읽어서 TRACE창에 표시
	str1 = _SFListGetValue("LC20@뷰페이지명", 0, 0);
	str2 = _SFListGetValue("LC20@뷰페이지명", 1, 0);
	str3 = _SFListGetValue("LC20@뷰페이지명", 2, 0);
	_TraceEx("(0,0):%s, (1,0):%s, (2,0):%s", str1, str2, str3);

	nCols = _SFListColCount("LC20@뷰페이지명");
	nRows = _SFListRowCount("LC20@뷰페이지명");
	_TraceEx("cols = %d, rows = %d", nCols, nRows);

}
```

2.17.14 _SFListSetBackColor()

함수원형

void _SFListSetBackColor(string strObject, int nCol, int nRow, int rgbBack)

파라메터	strObject : 오브젝트이름. nCol : 행번호 nRow : 열번호 rgbBack : 배경색
리턴형	없음
설명	리스트콘트롤 행(col), 열(row)의 배경색을 변경합니다.
함수활용	

```
/*********************************************************
예제 : 리스트컨트롤 셀변경
*********************************************************/
void SFList()
{
	// 열 10개 추가
	_SFListAppend("LC20@뷰페이지명", 10);
	//
	// col0, row0에 문자열 표시
	_SFListSetValue("LC20@뷰페이지명", 0, 0, "KOREA");
	// col1, row0에 문자열 표시
	_SFListSetValue("LC20@뷰페이지명", 1, 0, "World networking");
	// col2, row0에 실수값 표시
	_SFListSetValue("LC20@뷰페이지명", 2, 0, 100.5);

	// 배경색을 녹색으로 변경한다.
	_SFListSetBackColor("LC20@뷰페이지명", 0, 0, _RGB(0x00, 0xFF, 0x00));
}
```

2.17.15 _SFListSetBackColorEx()

함수원형

void _SFListSetBackColorEx(string strObject, int nCol1, int nRow1, int nCol2, int nRow2, int rgbBack)

파라메터	strObject : 오브젝트이름. nCol1 : 시작 행번호 nRow1 : 시작 열번호 nCol2 : 마지막 행번호 nRow2 : 마지막 열번호 rgbBack : 배경색
리턴형	없음
설명	리스트콘트롤 지정 범위의 배경색을 모두 변경합니다.
함수활용	

```
/*********************************************************
예제 : 리스트컨트롤 셀변경
*********************************************************/
void SFList()
{
	// 열 10개 추가
	_SFListAppend("LC20@뷰페이지명", 10);
	//
	// col0, row0에 문자열 표시
	_SFListSetValue("LC20@뷰페이지명", 0, 0, "KOREA");
	// col1, row0에 문자열 표시
	_SFListSetValue("LC20@뷰페이지명", 1, 0, "World networking");
	// col2, row0에 실수값 표시
	_SFListSetValue("LC20@뷰페이지명", 2, 0, 100.5);

	// 0,0 부터 2,0 영역의 배경색을 녹색으로 변경한다.
	_SFListSetBackColorEx("LC20@뷰페이지명", 0, 0, 2, 0, _RGB(0x00, 0xFF, 0x00));
}
```

2.17.16 _SFListSetRange()

함수원형

void _SFListSetRange(string strObject, string strStart, string strEnd)

파라메터	strObject : 오브젝트이름. strStart : 검색 시작일자 문자열 strEnd : 검색 종료일자 문자열
리턴형	없음
설명	eRunDB의 데이터를 조회하기 위하여 리스트컨트롤에 표시할 데이터의 검색범위(날짜)를 설정합니다.
함수활용	특정태그에 대해 eRunDB에 저장된 데이터를 조회하여 리스트컨트롤에 표시할 수 있습니다.

```
/*********************************************************
예제 : 리스트컨트롤 검색범위 설정
*********************************************************/
void SFListSetRange()
{
	string strStart;
	string strEnd;

	// 시작시간이 2014-06-05 00:00:00이고 종료시간이 2014-06-05 01:00:00 범위로 설정합니다.

	strStart = "2014-06-05 00:00:00";
	strEnd = "2014-06-05 01:00:00";

	_SFListSetRange("LC20@뷰페이지명", strStart, strEnd);
}
```

2.17.17 _SFListSetScanTag()

함수원형

void _SFListSetScanTag(string strObject, int nIndex, string strTag)

파라메터	strObject : 오브젝트이름. nIndex : 검색태그에 대한 인덱스번호 (0~15) strTag : 검색할 태그명
리턴형	없음
설명	eRunDB의 데이터를 조회하기 위하여 리스트컨트롤에 표시할 태그를 설정합니다.
함수활용	특정태그에 대해 eRunDB에 저장된 데이터를 조회하여 리스트컨트롤에 표시할 수 있습니다.

```
/*********************************************************
예제 : 리스트컨트롤 검색태그 설정
*********************************************************/
void SFListSetScanTag()
{
	// 태그명이 DEMO.PRESS_PV를 리스트컨트롤의 0번째 검색태그로 등록합니다.
	_SFListSetScanTag("LC20@뷰페이지명",  0, "DEMO.PRESS_PV");

	// 태그명이 DEMO.TEMP_PV를 리스트컨트롤의 1번째 검색태그로 등록합니다.
	_SFListSetScanTag("LC20@뷰페이지명",  1, "DEMO.TEMP_PV");
}
```

2.17.18 _SFListSetTextColor()

함수원형

void _SFListSetTextColor(string strObject, int nCol, int nRow, int rgbText)

파라메터	strObject : 오브젝트이름. nCol : 행번호 nRow : 열번호 rgbText : 글씨색
리턴형	없음
설명	리스트콘트롤 행(col), 열(row)의 글씨색을 변경합니다.
함수활용	

```
/*********************************************************
예제 : 리스트컨트롤 셀변경
*********************************************************/
void SFList()
{
	// 열 10개 추가
	_SFListAppend("LC20@뷰페이지명", 10);
	//
	// col0, row0에 문자열 표시
	_SFListSetValue("LC20@뷰페이지명", 0, 0, "KOREA");
	// col1, row0에 문자열 표시
	_SFListSetValue("LC20@뷰페이지명", 1, 0, "World networking");
	// col2, row0에 실수값 표시
	_SFListSetValue("LC20@뷰페이지명", 2, 0, 100.5);

	// 글씨색을 녹색으로 변경한다.
	_SFListSetTextColor("LC20@뷰페이지명", 0, 0, _RGB(0x00, 0xFF, 0x00));
}
```

2.17.19 _SFListSetTextColorEx()

함수원형

void _SFListSetTextColorEx(string strObject, int nCol1, int nRow1, int nCol2, int nRow2, int rgbText)

파라메터	strObject : 오브젝트이름. nCol1 : 시작 행번호 nRow1 : 시작 열번호 nCol2 : 마지막 행번호 nRow2 : 마지막 열번호 rgbText : 글씨색
리턴형	없음
설명	리스트콘트롤 지정 범위의 글씨색을 모두 변경합니다.
함수활용	

```
/*********************************************************
예제 : 리스트컨트롤 셀변경
*********************************************************/
void SFList()
{
	// 열 10개 추가
	_SFListAppend("LC20@뷰페이지명", 10);
	//
	// col0, row0에 문자열 표시
	_SFListSetValue("LC20@뷰페이지명", 0, 0, "KOREA");
	// col1, row0에 문자열 표시
	_SFListSetValue("LC20@뷰페이지명", 1, 0, "World networking");
	// col2, row0에 실수값 표시
	_SFListSetValue("LC20@뷰페이지명", 2, 0, 100.5);

	// 0,0 부터 2,0 영역의 배경색을 청색으로 변경한다.
	_SFListSetTextColorEx("LC20@뷰페이지명", 0, 0, 2, 0, _RGB(0x00, 0x00, 0xFF));
}
```

2.17.20 _SFListSetValue()

함수원형

void _SFListSetValue(string strObject, int nCol, int nRow, int nValue)

void _SFListSetValue(string strObject, int nCol, int nRow, double fValue)

void _SFListSetValue(string strObject, int nCol, int nRow, string strText)

파라메터	strObject : 오브젝트이름. nCol : 행번호 nRow : 열번호 nValue : 정수형 설정값 fValue : 실수형 설정값 strText : 문자열
리턴형	없음
설명	리스트콘트롤 행(col), 열(row)에 값을 설정합니다.
함수활용	데이터 연산 및 변수값 또는 태그값을 특정 셀에 표시하고자 할 때 사용합니다.

```
/*********************************************************
예제 : 리스트컨트롤 셀내용 변경
*********************************************************/
void SFList()
{
	// 열 10개 추가
	_SFListAppend("LC20@뷰페이지명", 10);
	//
	// col0, row0에 문자열 표시
	_SFListSetValue("LC20@뷰페이지명", 0, 0, "KOREA");
	// col1, row0에 문자열 표시
	_SFListSetValue("LC20@뷰페이지명", 1, 0, "World networking");
	// col2, row0에 실수값 표시
	_SFListSetValue("LC20@뷰페이지명", 2, 0, 100.5);
}
```

2.17.21 _SFListSetValue()

함수원형

void _SFListSetValue(string strObject, int nCol, int nRow, int nValue, int rgbBack, int rgbText)

void _SFListSetValue(string strObject, int nCol, int nRow, double fValue, int rgbBack, int rgbText)

void _SFListSetValue(string strObject, int nCol, int nRow, string strText, int rgbBack, int rgbText)

파라메터	strObject : 오브젝트이름. nCol : 행번호 nRow : 열번호 nValue : 정수형 설정값 rgbBack : 배경색 rgbText : 글씨색 fValue : 실수형 설정값 strText : 문자열
리턴형	없음
설명	리스트콘트롤 행(col), 열(row)에 값을 변경하면서, 배경색, 글씨색도 변경합니다.
함수활용	

```
/*********************************************************
예제 : 리스트컨트롤 셀내용 변경
*********************************************************/
void SFList()
{
	// 열 10개 추가
	_SFListAppend("LC20@뷰페이지명", 10);
	// col0, row0에 문자열 표시
	_SFListSetValue("LC20@뷰페이지명", 0, 0, "KOREA");
	// col1, row0에 문자열 표시, 배경색(빨강), 글씨색(흰색)
	_SFListSetValueEx("LC20@뷰페이지명", 1, 0, "World networking", _RGB(255,0,0), 0xFFFFFF);
	_MsgBox("NEXT", "", 0, 0);

	// col1, row0에 문자열 표시, 배경색(파랑), 글씨색(흰색)
	_SFListSetValueEx("LC20@뷰페이지명", 1, 0, "World networking", 0xFF0000, 0xFFFFFF);

	// col2, row0에 실수값 표시
	_SFListSetValue("LC20@뷰페이지명", 2, 0, 100.5);
}
```

2.17.22 _SFListUpdateColor()

함수원형

void _SFListUpdateColor(string strObject, int nCount, double fMin[], double fMax[], int rgbBack, int rgbText)

파라메터	strObject : 오브젝트이름. nCount : 행개수 fMin : 최소값을 가지고 있는 배열 fMax : 최대값을 가지고 있는 배열 rgbBack : 배경색 정수값 rgbText : 글씨색 정수값
리턴형	없음
설명	셀값들에 대해서 최소, 최대값의 범위를 벗어날경우 지정된 배경색, 전경색으로 변경합니다. 리스트 모든 행, 열을 일괄적으로 색상을 변경하며, 행에 대해서는 개수를 제한할 수 있습니다
함수활용	DB검색 이후 결과값의 식별을 용이하게 하기위해서 색상을 일괄적으로 변경합니다.

```
/*********************************************************
예제 : 리스트컨트롤 셀색상을 셀값의 조건에 따라 변경
*********************************************************/
void SFList()
{
	double fMin[1], fMax[1];
	fMin[0] = 8.0;
	fMax[0] = 10.5;

	// 열 10개 추가
	_SFListAppend("LC20@뷰페이지명", 10);
	// col0, row0에 문자열 표시
	_SFListSetValue("LC20@뷰페이지명", 0, 0, "12.5");

	// col 1의 최소,최대값을 벗어나면 배경색(빨강), 글씨색(흰색)으로 모든 열(row)을 변경한다.
	_SFListUpdateColor("LC20@뷰페이지명", 1, fMin, fMax, _RGB(255,0,0), _RGB(255,255,255));
	//
	_SFListUpdateTextColor("LC20@뷰페이지명", 1, fMin, fMax, _RGB(255,0,0));
	_SFListUpdateBackColor("LC20@뷰페이지명", 1, fMin, fMax, _RGB(255,0,0));

}
```

2.17.23 _SFListUpdateBackColor()

함수원형

void _SFListUpdateBackColor(string strObject, int nCount, double fMin[], double fMax[], int rgbBack)

파라메터	strObject : 오브젝트이름. nCount : 행개수 fMin : 최소값을 가지고 있는 배열 fMax : 최대값을 가지고 있는 배열 rgbBack : 배경색 정수값
리턴형	없음
설명	셀값들에 대해서 최소, 최대값의 범위를 벗어날경우 지정된 배경색으로 변경합니다. 리스트 모든 행, 열을 일괄적으로 색상을 변경하며, 행에 대해서는 개수를 제한할 수 있습니다
함수활용	DB검색 이후 결과값의 식별을 용이하게 하기위해서 색상을 일괄적으로 변경합니다.

```
/*********************************************************
예제 : 리스트컨트롤 셀색상을 셀값의 조건에 따라 변경
*********************************************************/
void SFList()
{
	double fMin[1], fMax[1];
	fMin[0] = 8.0;
	fMax[0] = 10.5;

	// 열 10개 추가
	_SFListAppend("LC20@뷰페이지명", 10);
	// col0, row0에 문자열 표시
	_SFListSetValue("LC20@뷰페이지명", 0, 0, "12.5");

	// col 1에 한해서 최소,최대값을 벗어나면 배경색(빨강)으로 모든 열(row)을 변경한다.
	_SFListUpdateBackColor("LC20@뷰페이지명", 1, fMin, fMax, _RGB(255,0,0));

}
```

2.17.24 _SFListUpdateTextColor()

함수원형

void _SFListUpdateTextColor(string strObject, int nCount, double fMin[], double fMax[], int rgbText)

파라메터	strObject : 오브젝트이름. nCount : 행개수 fMin : 최소값을 가지고 있는 배열 fMax : 최대값을 가지고 있는 배열 rgbText : 전경색 정수값
리턴형	없음
설명	셀값들에 대해서 최소, 최대값의 범위를 벗어날경우 지정된 전경색으로 변경합니다. 리스트 모든 행, 열을 일괄적으로 색상을 변경하며, 행에 대해서는 개수를 제한할 수 있습니다
함수활용	DB검색 이후 결과값의 식별을 용이하게 하기위해서 색상을 일괄적으로 변경합니다.

```
/*********************************************************
예제 : 리스트컨트롤 셀색상을 셀값의 조건에 따라 변경
*********************************************************/
void SFList()
{
	double fMin[1], fMax[1];
	fMin[0] = 8.0;
	fMax[0] = 10.5;

	// 열 10개 추가
	_SFListAppend("LC20@뷰페이지명", 10);
	// col0, row0에 문자열 표시
	_SFListSetValue("LC20@뷰페이지명", 0, 0, "12.5");

	// col 1에 한해서 최소,최대값을 벗어나면 전경색(빨강)으로 모든 열(row)을 변경한다.
	_SFListUpdateTextColor("LC20@뷰페이지명", 1, fMin, fMax, _RGB(255,0,0));

}
```

2.18 I/O 디바이스 함수

2.18.1 _DeviceControl()

함수원형

int _DeviceControl(string strDevice, string strCmd)

파라메터	strDevice : I/O디바이스 이름 strCmd : 명령 문자열
리턴형	정수형 0 : 정상. -1 : 실패
설명	Runtime중 통신 디바이스를 재시작과 같이 제어할 경우 사용합니다.
함수활용	특정 디바이스 드라이버와의 연결을 다시 해야 할 필요가 있을경우 사용합니다. -restart

※ I/O 디바이스 드라이버에서 지원하는 경우만 사용합니다.

※ 명령 문자열은 현재 다음과 같습니다.
-restart : 재시작

```
/*********************************************************
예제 : 디바이스 드라이버 제어
*********************************************************/
void DeviceControl()
{
        _DeviceControl("PLC", "-restart")
}
```

2.18.2 _DeviceGetAddress()

함수원형

int _DeviceGetAddress(string strDevice, string strFrame, string strTag)

파라메터	strDevice : I/O디바이스 이름 strFrame : 프레임코드 strTag : 돌려받을 시작어드레스 태그(문자열태그)
리턴형	정수형 0 : 정상. -1 : 실패
설명	Runtime중 통신 디바이스의 특정 프레임의 시작어드레스를 읽어서 돌려받을 태그에 저장합니다.
함수활용	

※ I/O 디바이스 드라이버에서 지원하는 경우만 사용합니다.

※ _DeviceSetAddress() 함수 참조하세요.

```
/*********************************************************
예제 : 디바이스의 시작어드레스 변경
*********************************************************/
// nIndex = 1 to 200th
// 200종류의 자동자 정보가 있다.
//
void DeviceGetAddress()
{
	// 디바이스 "PLC"의 "ZR1프레임"의 어드레스를 읽어서 문자열 태그"PLC.ZR1_ADDR"에 저장합니다.
	_DeviceGetAddress("PLC" ,"ZR1", "PLC.ZR1_ADDR");

	// 디바이스 "PLC"의 "ZR2프레임"의 어드레스를 읽어서 문자열 태그"PLC.ZR2_ADDR"에 저장합니다.
	_DeviceGetAddress("PLC" ,"ZR2", "PLC.ZR2_ADDR");

	// 디바이스 "PLC"의 "ZR3프레임"의 어드레스를 읽어서 문자열 태그"PLC.ZR3_ADDR"에 저장합니다.
	_DeviceGetAddress("PLC" ,"ZR3", "PLC.ZR3_ADDR");

	// 디바이스 "PLC"의 "ZR4프레임"의 어드레스를 읽어서 문자열 태그"PLC.ZR4_ADDR"에 저장합니다.
	_DeviceGetAddress("PLC" ,"ZR4", "PLC.ZR4_ADDR");
}

void DeviceSetAddress(int nIndex)
```

```
{
	int dwAddress;

	// 1st 차종 데이터영역은 20000에서
	dwAddress = 20000;

	// 20000
	dwAddress = dwAddress + (nIndex-1) * 2000;
	_DeviceSetAddress("PLC", "ZR1", _NumToStr(dwAddress, 10));  // 20000 - 20499

	dwAddress = dwAddress + 500;
	_DeviceSetAddress("PLC", "ZR2", _NumToStr(dwAddress, 10));  // 20500 - 20999

	dwAddress = dwAddress + 500;
	_DeviceSetAddress("PLC", "ZR3", _NumToStr(dwAddress, 10));  // 21000 - 21499

	dwAddress = dwAddress + 500;
	_DeviceSetAddress("PLC", "ZR4", _NumToStr(dwAddress, 10));  // 21500 - 21999
}
```

2.18.3 _DeviceGetRxCount()

함수원형

int _DeviceGetRxCount(string strDevice, string strFrame, string strTag)

파라메터	strDevice : I/O디바이스 이름 strFrame : 프레임코드 strTag : 결과값 저장용 태그
리턴형	정수형 0 : 정상. -1 : 실패
설명	Runtime중 통신 디바이스의 프레임에 대한 수신횟수를 태그에 저장한다.
함수활용	본 함수를 스크립트함수로 반복적으로 실행하면 세번째 파라메터에 설정된 태그에 수신회수가 저장된다.

```
/*********************************************************
예제 : MELSEC Q PLC디바이스 수신회수 읽기
*********************************************************/
void GetRxCount()
{
	int ret;

	// 프로젝트에 등록된 I/O 디바이스 "Q" 의 "D168" 프레임의 송출횟수를
	// 태그("PLC.TX_CNT")의 값에 표시합니다.
	ret = _DeviceGetTxCount("Q", "D168", "PLC.TX_CNT");
	// OK : 0, FAIL : -1

	// 프로젝트에 등록된 I/O 디바이스 "Q" 의 "D168" 프레임의 수신횟수를
	// 태그("PLC.RX_CNT")의 값에 표시합니다.
	ret = _DeviceGetRxCount("Q", "D168", "PLC.RX_CNT");
	// OK : 0, FAIL : -1
}
```

2.18.4 _DeviceGetScanInfo()

함수원형

int _DeviceGetScanInfo(string strDevice, string strFrame, string strTag)

파라메터	strDevice : I/O디바이스 이름 strFrame : 프레임코드 strTag : 결과값 저장용 태그
리턴형	정수형 0 : 정상. -1 : 실패
설명	Runtime중 통신 디바이스의 프레임에 대한 스캔정보를 태그에 저장한다.
함수활용	프레임 설정값은 스캔타임, 타임아웃 값을 의미하고, 특정 프레임의 통신을 Enable/Disable 시킨다. 읽어온 값은 세번째 파라메터 태그에 저장된다.

※ 세번째 파라메터에 지정된 태그에 스캔정보가 다음과 같이 저장됩니다.
하위 16bit WORD값 (_LoWord) : 스캔타임.
상위 16bit WORD값 (_HiWord) : 타임아웃.

Scantime = _LoWord(nValue)
Timeout = _HiWord(nValue),

```
/*********************************************************
예제 : Runtime중 "Q" 디바이스의 스캔타임, 타임아웃 읽기
*********************************************************/
void GetScanInfo()
{
	int ret;
	int nScantime;
	int nTimeout;
	int nScanInfo;

	// 디바이스 이름 : "Q"
// 프레임 코드 : "D168"
// 설정값 저장할 태그명 : MELSECQ.SCAN_INFO

	ret = _DeviceGetScanInfo("Q", "D168", "MELSECQ.SCAN_INFO");
	// 약 1초간 기다린다.
_Sleep(1000);
```

```
	nScanInfo = @MELSECQ.SCAN_INFO;
	nScantime = _LoWord(nScanInfo);  // 하위 16비트.
	nTimeout = _HiWord(nScanInfo);   // 상위 16비트

	// OK : 0, FAIL : -1
	_TraceEx("_DeviceGetScanInfo(ret=%d) scantime =%d, timeout=%d", ret, nScantime, nTimeout);
}

/*********************************************************
예제2 : 참고하세요.
*********************************************************/
// nIndex = 1 to 200th
// 차종데이터 영역 읽을 것인지
// 어드레스를 변경하기 전에 스캔을 멈추고 변경후 스캔을 한번 하게 한다.
void DeviceSetScanInfo(int nScanTime)
{
	int dwScanInfo;
	int nTimeOut;

	nTimeOut = 1000;
	// 스캔을 멈추게 한다.
	// 스캔타임을 0으로 하면 스캔하지 않는다.
	dwScanInfo = _MakeLong(nScanTime, nTimeOut);

	_DeviceSetScanInfo("PLC", "ZR1", dwScanInfo);
	_DeviceSetScanInfo("PLC", "ZR2", dwScanInfo);
	_DeviceSetScanInfo("PLC", "ZR3", dwScanInfo);
	_DeviceSetScanInfo("PLC", "ZR4", dwScanInfo);
}

void DeviceGetScanInfo()
{
	int dwScanInfo;
	int nTimeOut;

	_DeviceGetScanInfo("PLC", "ZR1", "PLC.ZR1_SCAN");
	_DeviceGetScanInfo("PLC", "ZR2", "PLC.ZR2_SCAN");
	_DeviceGetScanInfo("PLC", "ZR3", "PLC.ZR3_SCAN");
	_DeviceGetScanInfo("PLC", "ZR4", "PLC.ZR4_SCAN");
}
```

2.18.5 _DeviceGetTxCount()

함수원형

int _DeviceGetTxCount(string strDevice, string strFrame, string strTag)

파라메터	strDevice : I/O디바이스 이름 strFrame : 프레임코드 strTag : 결과값 저장용 태그
리턴형	정수형 0 : 정상. -1 : 실패
설명	Runtime중 통신 디바이스의 프레임에 대한 송출횟수를 태그에 저장한다.
함수활용	프레임 송출제어시 사용하는데, 본 함수를 스크립트함수로 반복적으로 실행하면 세번째 파라메터에 설정된 태그에 송출회수가 저장된다.

```
/*********************************************************
예제 : MELSEC Q PLC디바이스 송출회수 읽기
*********************************************************/
void GetTxCount()
{
	int ret;

	// 프로젝트에 등록된 I/O 디바이스 "Q" 의 "D168" 프레임의 송출횟수를
	// 태그("PLC.TX_CNT")의 값에 표시합니다.
	ret = _DeviceGetTxCount("Q", "D168", "PLC.TX_CNT");
	// OK : 0, FAIL : -1

	// 프로젝트에 등록된 I/O 디바이스 "Q" 의 "D168" 프레임의 수신횟수를
	// 태그("PLC.RX_CNT")의 값에 표시합니다.
	ret = _DeviceGetRxCount("Q", "D168", "PLC.RX_CNT");
	// OK : 0, FAIL : -1
}
```

2.18.6 _DeviceSetAddress()

함수원형

int _DeviceSetAddress(string strDevice, string strFrame, string strAddress)

파라메터	strDevice : I/O디바이스 이름 strFrame : 프레임코드 strAddress : 시작어드레스
리턴형	정수형 0 : 정상. -1 : 실패
설명	Runtime중 통신 디바이스의 특정 프레임 시작어드레스를 변경합니다.
함수활용	대량의 데이터를 읽어야 하는 경우 특수하게 사용합니다.

※ I/O 디바이스 드라이버에서 지원하는 경우만 사용합니다.

```
/*********************************************************
예제 : 디바이스의 시작어드레스 변경
*********************************************************/
// nIndex = 1 to 200th
// 200종류의 자동자 정보가 있다.
void DeviceSetAddress(int nIndex)
{
	int dwAddress;
	// 1st 차종 데이터영역은 20000에서
	dwAddress = 20000;

	// 20000
	dwAddress = dwAddress + (nIndex-1) * 2000;
	_DeviceSetAddress("PLC", "ZR1", _NumToStr(dwAddress, 10)); // 20000 - 20499

	dwAddress = dwAddress + 500;
	_DeviceSetAddress("PLC", "ZR2", _NumToStr(dwAddress, 10)); // 20500 - 20999

	dwAddress = dwAddress + 500;
	_DeviceSetAddress("PLC", "ZR3", _NumToStr(dwAddress, 10)); // 21000 - 21499

	dwAddress = dwAddress + 500;
	_DeviceSetAddress("PLC", "ZR4", _NumToStr(dwAddress, 10)); // 21500 - 21999
}
```

2.18.7 _DeviceSetScanInfo()

함수원형

int _DeviceSetScanInfo(string strDevice, string strFrame, int nInfoValue)

파라메터	strDevice : I/O디바이스 이름 strFrame : 프레임코드 nInfoValue : 스캔타임, 타임아웃의 32bit 조합값이다.
리턴형	정수형 0 : 정상. -1 : 실패
설명	Runtime중 통신 디바이스의 프레임에 대한 스캔정보를 변경합니다.
함수활용	프레임 설정값은 스캔타임, 타임아웃 값을 의미하고, 특정 프레임의 통신을 Enable/Disable 시킨다. 초기 스캔타임이 0으로 설정된 프레임은 이 함수호출이 무의미하므로, Runtime중 변경을 하려면 해당프레임의 스캔타임이 0보다 큰 값으로 설정 되어있어야 한다.

※ 세번째 파라메터에 설정되는 값은 다음과 같이 설정합니다.
하위 16bit WORD값 : 스캔타임.
상위 16bit WORD값 : 타임아웃.

nInfoValue = _MakeLong(scantime, timeout)

```
/*********************************************************
예제 : Runtime중 "Q" 디바이스의 스캔타임, 타임아웃 변경
*********************************************************/
void SetScanInfo()
{
	int ret;
	int nScantime;
	int nTimeout;
	int nScanInfo;

	nScantime = (_Rand() % 1000) + 10;
	nTimeout = 3000;
	nScanInfo = _MakeLong(nScantime, nTimeout);

	//
	ret = _DeviceSetScanInfo("Q", "D168", nScanInfo);

	// OK : 0, FAIL : -1
```

```
	_TraceEx("_DeviceSetScanInfo(scan=%d, timeout=%d,  ret : %d", nScantime, nTimeout, ret);
}

/*******************************************************
예제2 : 참고하세요.
*******************************************************/
// nIndex = 1 to 200th
// 차종데이터 영역 읽을 것인지
// 어드레스를 변경하기 전에 스캔을 멈추고 변경후 스캔을 한번 하게 한다.
void DeviceSetScanInfo(int nScanTime)
{
	int dwScanInfo;
	int nTimeOut;

	nTimeOut = 1000;
	// 스캔을 멈추게 한다.
	// 스캔타임을 0으로 하면 스캔하지 않는다.
	dwScanInfo = _MakeLong(nScanTime, nTimeOut);

	_DeviceSetScanInfo("PLC", "ZR1", dwScanInfo);
	_DeviceSetScanInfo("PLC", "ZR2", dwScanInfo);
	_DeviceSetScanInfo("PLC", "ZR3", dwScanInfo);
	_DeviceSetScanInfo("PLC", "ZR4", dwScanInfo);
}

//
void DeviceGetScanInfo()
{
	int dwScanInfo;
	int nTimeOut;

	_DeviceGetScanInfo("PLC", "ZR1", "PLC.ZR1_SCAN");
	_DeviceGetScanInfo("PLC", "ZR2", "PLC.ZR2_SCAN");
	_DeviceGetScanInfo("PLC", "ZR3", "PLC.ZR3_SCAN");
	_DeviceGetScanInfo("PLC", "ZR4", "PLC.ZR4_SCAN");
}
```

2.18.8 _DeviceWriteBytes()

함수원형

int _DeviceWriteBytes(string strDevice, string strFrame, int nOffset, char data[4000], int nSize)

파라메터	strDevice : I/O디바이스 이름 strFrame : 프레임코드 nOffset : 통신프레임에 설정된 시작주소로부터 변위값 data : 내보낼 배열버퍼 nSize : 내보낼 바이트길이
리턴형	정수형 0 : 정상 -1 : 실패
설명	런타임중에 통신 디바이스에 출력할 바이트(8비트) 또는 워드(16) 데이터를 보낸다. (Max. 4000 Bytes)
함수활용	PLC등의 내부 버퍼를 가지고 있는 디바이스 통신을 할 경우 연속적으로 보내야 하는 자료를 한번에 보낼 때 사용합니다.

※ 세번째 파라메터 오프셋은 PLC 디바이스와 같이 연속버퍼를 가지고 있을경우 몇 번째부터 데이터를 변경할지 정할 수 있습니다. (기본값은 1)
※ 내보내는 데이터는 배열변수로 선언하고 변수이름만 적어줍니다.

```
/*********************************************************
예제 : 바이트 데이터 송출
*********************************************************/
// MelsecQEtn Driver.(MC Protocol)
// 5개의 바이트 데이터를 보낸다. (5 bytes)
void WriteBytes()
{
	int ret;
	char data[5];
	string szDevice, szFrame;
	int offset;

	data[0] = 0x10;
	data[1] = 0x20;
	data[2] = 0x30;
	data[3] = 0x40;
	data[4] = 0x50;
```

```
	szDevice = "MelsecQEtn";
	szFrame = "C1";
	offset  = 1; // 옵셋은 1부터 시작한다.
	ret = _DeviceWriteBytes(szDevice, szFrame, offset, data, 5);
}

//------------------------------------------------------------------------
// MelsecQEtn Driver.(MC Protocol)
// 5개의 워드값을 보낸다 (10 bytes)
void WriteWords()
{
	int ret;
	short nData[5];

	nData[0] = 0x0110;
	nData[1] = 0x0220;
	nData[2] = 0x0330;
	nData[3] = 0x0440;
	nData[4] = 0x0550;

	ret = _DeviceWriteBytes("MelsecQEtn", "C1", 1, nData, 10);
}
```

2.18.9 _DeviceWriteWords()

함수원형

int _DeviceWriteWords(string strDevice, string strFrame, int nOffset, short data[4000], int nSize)

파라메터	strDevice : I/O디바이스 이름 strFrame : 프레임코드 nOffset : 통신프레임에 설정된 시작주소로부터 변위값 data : 내보낼 배열버퍼. Short형(2바이트) nSize : 내보낼 배열버퍼 개수
리턴형	정수형 0 : 정상 -1 : 실패
설명	런타임중에 통신 디바이스에 출력할 워드(16비트) 데이터를 보냅니다. (Max. 4000 words)
함수활용	PLC등의 내부 버퍼를 가지고 있는 디바이스 통신을 할 경우 연속적으로 보내야 하는 자료를 한번에 보낼 때 사용합니다.

※ I/O 디바이스 드라이버에서 지원하는 경우만 사용합니다.

※ 세번째 파라메터 오프셋은 PLC 디바이스와 같이 연속버퍼를 가지고 있을경우 몇 번째부터 데이터를 변경할 지 정할 수 있습니다. (기본값은 1)
※ 내보내는 데이터는 배열변수로 선언하고 변수이름만 적어줍니다.

```
/*********************************************************
예제 : 워드(2바이트) 데이터 송출
*********************************************************/
// MelsecQEtn Driver.(MC Protocol)
// 5개의 바이트 데이터를 보낸다. (5 bytes)
void WriteWords()
{
	int ret;
	short data[5];
	string szDevice, szFrame;
	int offset;

	data[0] = 0x1234;
	data[1] = 0x5678;
	data[2] = 0xabcd;
	data[3] = 0x12ab;
```

```
	data[4] = 0x0050;

	szDevice = "MelsecQEtn";
	szFrame = "C1";
	offset  = 1; // 옵셋은 1부터 시작한다.
	// word size 5는 word의 개수를 의미한다.
	ret = _DeviceWriteWords(szDevice, szFrame, offset, data, 5);
}

// 아래의 함수예제와 같이 사용할 경우 의미는 같습니다.
//---------------------------------------------------------------------
// MelsecQEtn Driver.(MC Protocol)
// 5개의 워드값을 보낸다 (10 bytes)
void WriteWords()
{
	int ret;
	short nData[5];

	nData[0] = 0x0110;
	nData[1] = 0x0220;
	nData[2] = 0x0330;
	nData[3] = 0x0440;
	nData[4] = 0x0550;

	// 마지막 파라메터 10은 바이트 개수를 의미합니다.
	ret = _DeviceWriteBytes("MelsecQEtn", "C1", 1, nData, 10);
}
```

2.18.10 _DeviceWriteString()

함수원형

int _DeviceWriteString(string strDevice, string strFrame, string strData)

파라메터	strDevice : I/O디바이스 이름 strFrame : 프레임코드 strData : 출력 문자열
리턴형	정수형 0 : 정상 -1 : 실패
설명	런타임중에 통신 디바이스에 출력할 문자열을 보낸다. (Max. 4000bytes)
함수활용	PLC등의 내부 버퍼를 가지고 있는 디바이스 통신을 할 경우 연속적으로 보내야 하는 자료를 한번에 보낼 때 사용합니다.

```
/*********************************************************
예제 : 문자열형태 데이터를 PLC로 송출
*********************************************************/
// MelsecQEtn Driver.(MC Protocol)
void WriteString()
{
	int ret;
	int nCnt;
	string strData;
	string sResult[10];
	strData = "123*korea*King";
	// strData 문자열을 * 문자구분으로 분리해서 sResult 문자열배열에 넣어준다.
	nCnt = _ExtractSubString(strData, "*", sResult);
	// MelsecQEtn Driver.(MC Protocol)

	ret = _DeviceWriteString("MelsecQEtn", "C2", "test string");
	_Sleep(1000);
	ret = _DeviceWriteString("MelsecQEtn", "C2", strData);
	_Sleep(1000);
	ret = _DeviceWriteString("MelsecQEtn", "C2", sResult[0]);
	_Sleep(1000);
	ret = _DeviceWriteString("MelsecQEtn", "C2", @TAG.TEXT);
}
```

2.19 ACTIVE-X 함수

2.19.1 _ComGetValue()

함수원형

int _ComGetValue(string strObject, string strVarName)

double _ComGetValue(string strObject, string strVarName)

string _ComGetValue(string strObject, string strVarName)

파라메터	strObject : ACTIVE-X 오브젝트 이름 strVarName : 모듈에서 제공하는 변수이름
리턴형	정수형, 실수형, 문자열
설명	ActiveX 모듈의 변수에 해당되는 값을 읽어옵니다.
함수활용	

※ 내부변수 이름 및 변수형은 모듈 매뉴얼을 참고해야 합니다.

```
/*********************************************************
예제 : ActiveX 모듈의 값을 읽어오기
*********************************************************/
void ComGetValue()
{
	string str;
	// 곰플레이어의 버전 읽어와서 문자열 변수 str에 저장합니다.
	str = _ComGetValue("곰플레이어@뷰페이지명", "Version");

	// 문자열 변수 str의 값을 메시지박스에 표시합니다.
	_MsgBox(str, "", 0, 1);
}
```

2.19.2 _ComMethod()

함수원형

void _ComMethod(string strObject, string strFuncName, data, ...)

파라메터	strObject : ACTIVE-X 오브젝트 이름 strFuncName : 모듈에서 제공하는 함수이름 data : strFuncName 함수의 파라메터 ... : strFuncName 함수의 파라메터 개수만큼 지정.
리턴형	없음
설명	ActiveX 모듈의 함수의 프로토 타입에 맞춰서 함수를 호출합니다.
함수활용	지정 외부 ACTIVE-X 모듈에서 제공하는 함수를 호출하고, 명령을 실행합니다.

※ 파라매터 개수는 호출된 ACTIVE-X 모듈에서 제공하는 함수를 참고하여야 합니다.
※ 모듈 매뉴얼을 참고해야 합니다.

```
/*********************************************************
예제 : ActiveX 모듈 함수 호출하기
*********************************************************/
void _ComMethod()
{
	// TestActiveX 모듈의 Save 함수를 호출하여 C루트에 TestActiveX.txt 파일을 저장합니다.
	_ComMethod("TestActiveX@뷰페이지명", "Save", "C:₩₩", "testActiveX.txt");
}
```

2.19.3 _ComSetValue()

함수원형

void _ComSetValue(string strObject, string strVarName, int nValue)

void _ComSetValue(string strObject, string strVarName, double fValue)

void _ComSetValue(string strObject, string strVarName, string strValue)

파라메터	strObject : ACTIVE-X 오브젝트 이름 strVarName : 모듈에서 제공하는 변수이름 nValue : strVarName 정수형 변수값 fValue : strVarName 실수형 변수값 strValue : strVarName 문자열 변수값
리턴형	없음
설명	ActiveX 모듈에서 사용되는 변수의 값을 변경합니다.
함수활용	

※ 내부변수 이름 및 설정방법은 모듈 매뉴얼을 참고해야 합니다.

```
/*********************************************************
예제 : ActiveX 모듈에 값 쓰기
*********************************************************/
void ComSetValue()
{
	// 곰플레이어의 CapImage파일명을 설정합니다.
	_ComSetValue("곰플레이어@뷰페이지명", "CapImageFile", "C:₩MSX.mpeg");
}
```

2.20 HSMS 함수

2.20.1 _HSMSMethod()

함수원형

void _HSMSMethod(string strAPI, data1, data2, ...)

파라메터	strAPI : xGEM에서 제공하는 인터페이스 함수명 data1, data2 : 인터페이스 함수에 따라서 사용되는 파라메터 ... : strAPI 인터페이스 함수 파라메터 개수만큼 지정.
리턴형	없음
설명	링크제니시스의 xGEM 라이브러리에서 제공하는 인터페이스 함수를 호출합니다.
함수활용	eRun에서 GEM통신 인터페이스 구현할 때 사용합니다.

※ 파라매터 개수는 호출된 xGem300 모듈에서 제공하는 함수를 참고하여야 합니다.
※ 모듈 매뉴얼을 참고해야 합니다.

■ 일반적인 작업 방법

eRun에서는 간편한 SECS통신을 위해 통신드라이버는 링크제니시스사의 xGem300Pro를 기반으로 하고 있습니다.

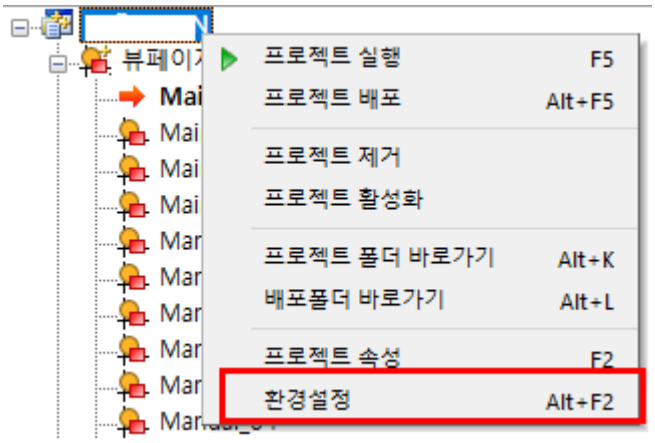
일반적인 CIM 프로그램 개발 방법은 아래와 같은 순서를 따릅니다.

1. EQ의 특성에 따라 HOST와 통신하는 시나리오를 정리합니다.
2. EQ에서 표출할 수 있는 각종 이벤트(CEID)와 데이터(VID)를 정리합니다.
3. HOST가 요구하는 Stream and Function을 정리합니다.
4. 링크제니시스사의 Configuration Tool을 이용하여 앞서 정리한 내용을 작성합니다.
5. eRun에서 위의 결과파일을 로딩합니다.
6. 작화작업 및 이벤트와 메써드들을 작성합니다.
7. 최종 결과파일을 작성합니다.

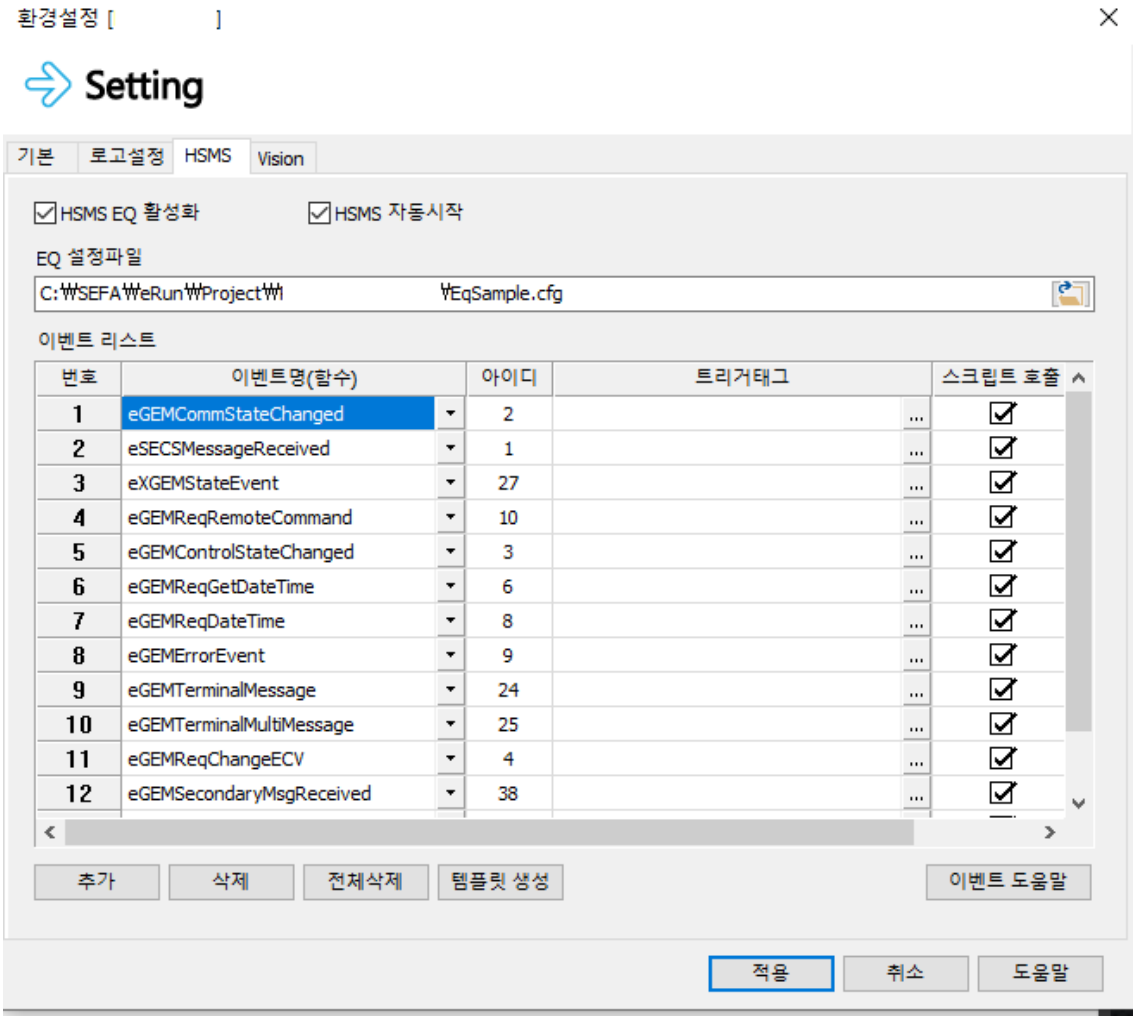
본 문서는 위의 내용중 5, 6, 7에 해당하는 후반부 작업에 대해 기술 합니다. 기타 사항은 드라이버 제조사에 문의하시기 바랍니다.

■ eRun에서 링크제니시스사의 Configuration Tool의 결과파일을 로딩하는 방법

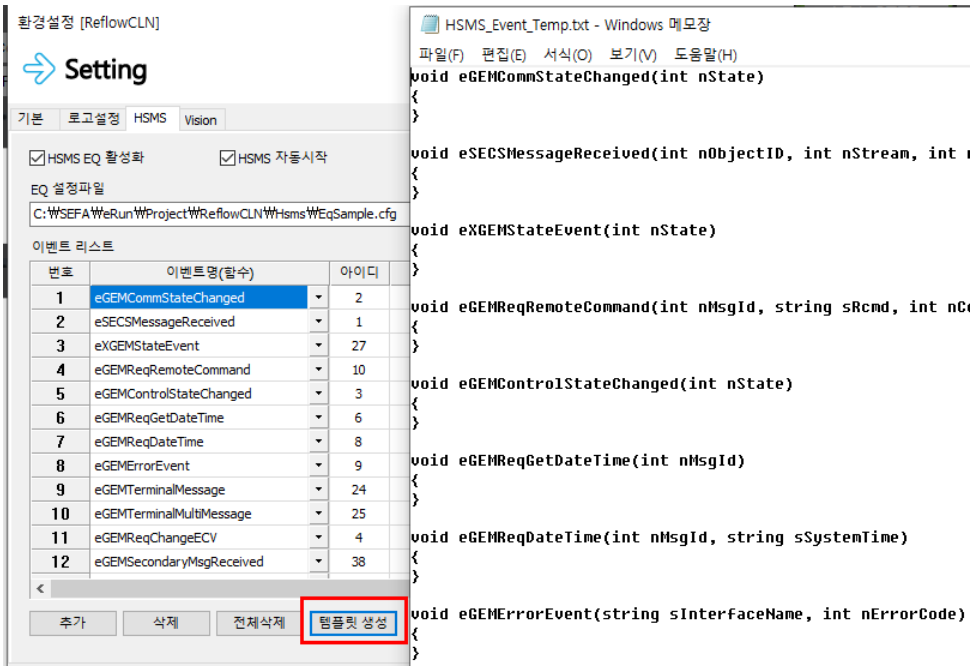
아래의 그림과 같이 프로젝트리스트에서 오른쪽 버튼을 클릭하여 환경설정을 합니다.



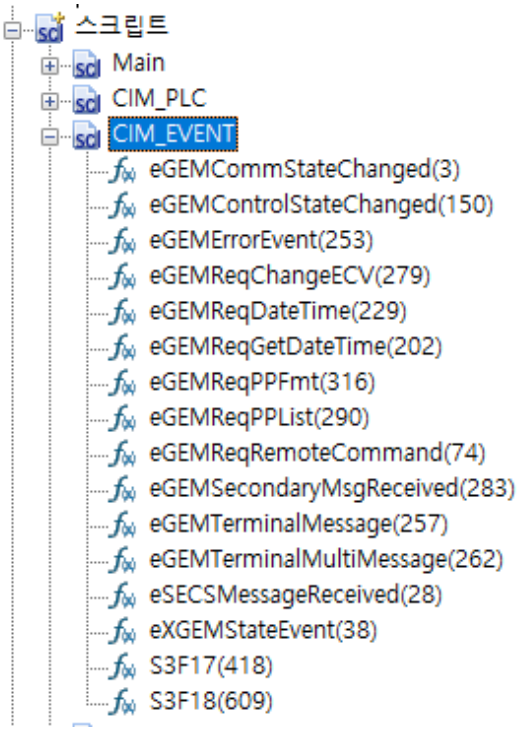
아래의 그림과 같이 설정파일을 로딩하고, 사용할 이벤트들을 추가합니다.



아래의 그림과 같이 탬플릿 생성을 하여 eRun에서 편리하게 해당 이벤트들을 작성할 수 있습니다.



아래의 그림처럼 스크립트파일을 생성한후 탬플릿으로 생성된 코드를 붙여 넣기합니다.



생성된 이벤트의 내용을 작성합니다.

```
201  //S2F17
202  void eGEMReqGetDateTime(int nMsgId)
203  {
204      string strYYYY;
205      string strMM;
206      string strDD;
207      string strHH;
208      string strmm;
209      string strSS;
210      string strDateTime;
211      string strLog;
212
213      strYYYY = _FormatString("%04d",@SYSTEM.N_YEAR) ;
214      strMM = _FormatString("%02d", @SYSTEM.N_MONTH) ;
215      strDD = _FormatString("%02d",@SYSTEM.N_DAY) ;
216      strHH = _FormatString("%02d",@SYSTEM.N_HOUR) ;
217      strmm = _FormatString("%02d",@SYSTEM.N_MIN) ;
218      strSS = _FormatString("%02d",@SYSTEM.N_SEC) ;
219
220      strDateTime = _FormatString("%s%s%s%s%s%s", strYYYY, strMM, strDD, strHH, strmm, strSS);
221
222      WriteLog("[H->E] S2F17");
223
224      _HSMSMethod("GEMRspGetDateTime", nMsgId, strDateTime);
225      WriteLog("[H<-E] S2F18");
226  }
227
228  //S2F31
229  void eGEMReqDateTime(int nMsgId, string sSystemTime)
230  {
231      string strLog;
232      string strYYYY, strMM, strDD, strHH, strmm, strSS, strcc;
233      string strDateTime;
234
235      WriteLog("[H->E] S2F31");
236
237      _HSMSMethod("GEMRspDateTime", nMsgId, 0); //0: ACK, 0<: NAK
238      WriteLog("[H<-E] S2F32");
```

이때 실제 SECS통신의 Stream and Function에 해당하는 이벤트 목록은 링크제시시스(주)의 메뉴얼을 참고하시기 바랍니다.

■ eRun에서 SECS통신 METHOD 사용방법

각 이벤트의 응답에 필요한 Method를 작성합니다.

SECS통신은 홀수 번호와 짝수번호가 하나의 쌍으로 이루고 있으며,

홀수 번호로 요청을 하면, 짝수번호로 규약된 메시지 형태로 응답을 해야 합니다.

이때 메써드를 사용해야 합니다.

자세한 사항은 예제 프로젝트를 참고하시기 바랍니다.

2.21 VISION 함수

2.21.1 _VisionMethod()

함수원형

void _VisionMethod(string strAPI, data1, data2, ...)

파라메터	strAPI : Sherlock™에서 제공하는 인터페이스 함수명 data1, data2 : 인터페이스 함수에 따라서 사용되는 파라메터 ... : strAPI 인터페이스 함수 파라메터 개수만큼 지정.
리턴형	없음
설명	Sherlock™ Vision 라이브러리에서 제공하는 인터페이스 함수를 호출합니다.
함수활용	

※ 파라매터 개수는 호출된 Sherlock™ 모듈에서 제공하는 함수를 참고하여야 합니다.
※ 모듈 매뉴얼을 참고해야 합니다.

■ eRun에서 지원하는 Sherlock™ 함수 대응 표

eRun 함수명	Sherlock 함수명	비고
InvLoad	InvLoad	ivs 파일 로드 함수
SetDisplay	ConnectImgWindow	isv 파일내의 Window와 eRun Object간의 연결 명령
SetMode	InvModeSet	검사 모드 변경 명령
GetMode	InvModeGet	검사 모드 확인 명령
Toolbar	ShowToolbar	eRun Vision Object의 영상 확대/ 축소 등의 툴바 표시 기능 설정 명령
Zoom	SetZoom	영상의 확대/ 축소 명령
SetDouble	VarSetDouble	Ivs파일내의 실수형 변수를 설정하는 명령
GetDouble	VarGetDouble	Ivs파일내의 실수형 변수를 읽어오는 명령
SetBool	VarSetBool	Ivs파일내의 부울형 변수를 설정하는 명령
GetBool	VarGetBool	Ivs파일내의 부울형 변수를 읽어오는 명령
SetString	VarSetString	Ivs파일내의 문자열 변수를 설정하는 명령
GetString	VarGetString	Ivs파일내의 문자열 변수를 읽어오는 명령
ImageLoad	ImageLoad	저장된 이미지를 영상에 표시하기 위해 읽어오는 명령
ImageSave	ImageSave	현재의 이미지를 저장하는 명령
LiveSet	LiveSet	영상을 실시간으로 촬상 하는 명령

■ 함수 사용예시

각 함수의 형태는 _VisionMethod("함수명", "eRunVisionObject명", "인자값") 의 형태를 갖습니다. 각 인자의 자료형과 개수는 Sherlock의 것을 따릅니다.

```
// VISION10 Object에 Sample1.ivs 파일을 로딩하여 검사를 할 수 있도록 준비합니다.
// "Sample1.ivs" 파일명은 경로포함해서 지정합니다.
nRet = _VisionMethod("InvLoad", "VISION10", "Simple1.ivs");

// VISION10 Object에 표시할 ivs내의 윈도우 이름을 지정합니다.
nRet = _VisionMethod("SetDisplay", "VISION10", "imgA");

// VISION10 Object에 표시할 ivs내의 윈도우 이름을 같이 지정합니다.
// 한번의 함수 호출로 로드("InvLoad")하고 표시윈도우를 설정("SetDisplay")을 실행합니다.
nRet = _VisionMethod("InvLoad2", "VISION10", "Simple1.ivs", "imgA");

// 비전동작 방법을 지정합니다.
nRet = _VisionMethod("SetMode", "object", nMode);
//I_EXE_MODE_ONCE = 0,
//I_EXE_MODE_CONT = 1,
//I_EXE_MODE_CALIB = 2,
//I_EXE_MODE_HALT = 3,
//I_EXE_MODE_HALT_AFTER_ITERATION = 4

// 현재 설정된 비전 동작 방법을 얻어옵니다.
nRet = _VisionMethod("GetMode", "object");

// 화면에 툴바를 표시/ 비표시 합니다.
nRet = _VisionMethod("Toolbar", "VISION10", 0 or 1);

// 영상의 확대/ 축소를 합니다.
nRet = _VisionMethod("Zoom", "VISION10", 0 or 1);

// ivs 파일에 설정한 실수형 변수에 값을 저장합니다.
nRet = _VisionMethod("SetDouble", " VISION10", "varA", dValue);

// ivs 파일에 저장된 실수형 변수의 값을 읽어옵니다.
dValue = _VisionMethod("GetDouble", " VISION10", "varA");

// ivs 파일에 설정한 부울형 변수에 값을 저장합니다.
nRet = _VisionMethod("SetBool", " VISION10", "varA", bVal);
```

```
// ivs 파일에 저장된 부울형 변수의 값을 읽어옵니다.
bVal = _VisionMethod("GetBool", " VISION10", "varA");

// ivs 파일에 설정한 문자열 변수에 값을 저장합니다.
nRet = _VisionMethod("SetString",  " VISION10", "varstring", "text....");

// ivs 파일에 저장된 문자열 변수의 값을 읽어옵니다.
strValue = _VisionMethod("GetString",  " VISION10", "varstring");


// 이미지 파일을 읽어 화면에 표시합니다.
nRet = _VisionMethod("ImageLoad", " VISION10", "file path");

// 현재 이미지를 저장합니다.
nRet = _VisionMethod("ImageSave", " VISION10", "file path");

// 영상을 실시간 촬상 모드로 변환 합니다.
nRet = _VisionMethod("LiveSet", " VISION10", 0 or 1);
```

■ 일반적인 개발 절차

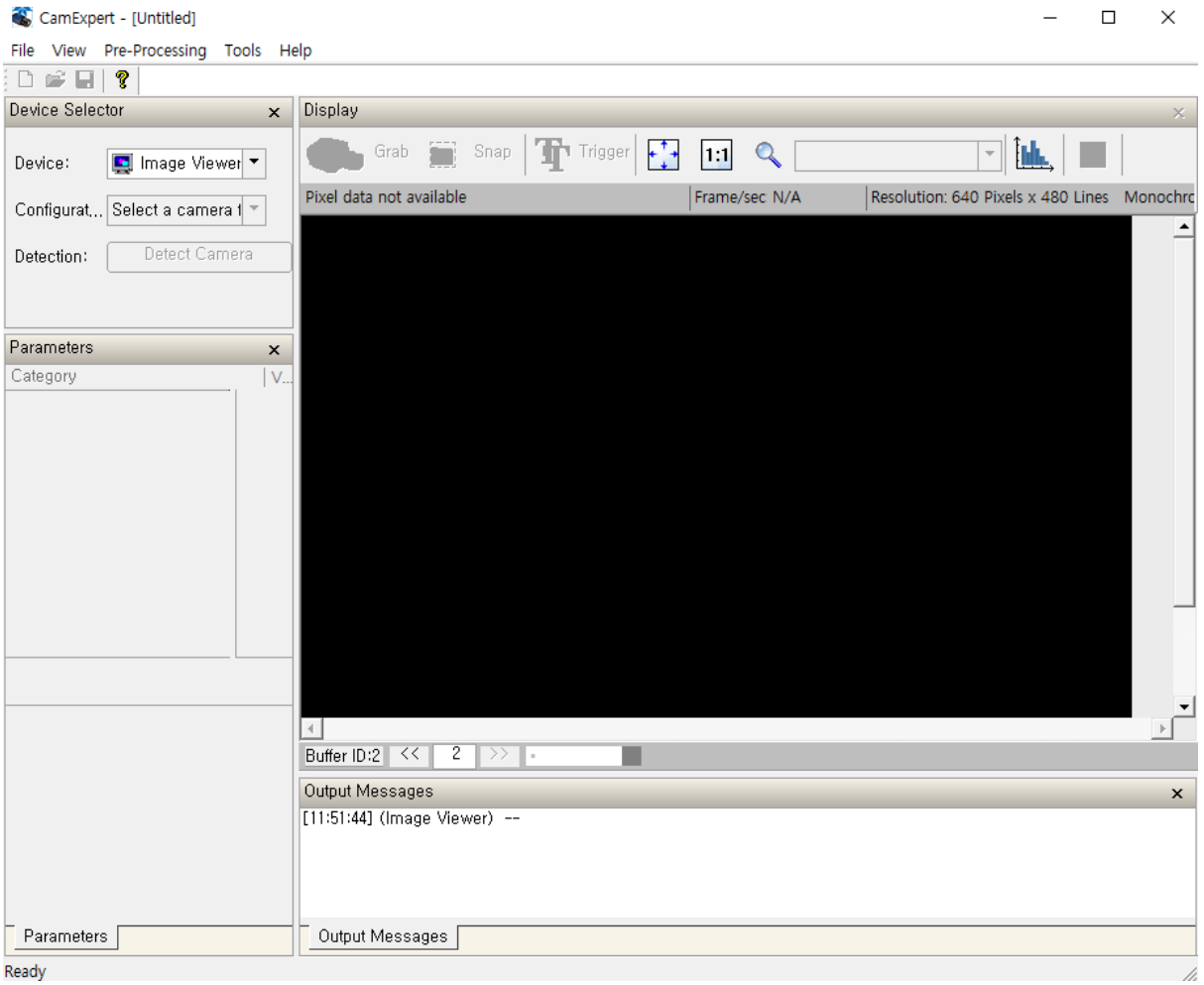
eRun Vision을 이용하여 개발하는 일반적인 절차는 아래와 같습니다.

1. Camera 세팅
2. Sherlock을 이용한 비전 알고리즘 선정
3. eRun과 데이터를 공유하기 위한 변수설정.
4. eRun에서 작화 작업 진행
5. eRun과 Sherlock간의 변수 연결
6. 최종 파일 생성

1. Camera 세팅

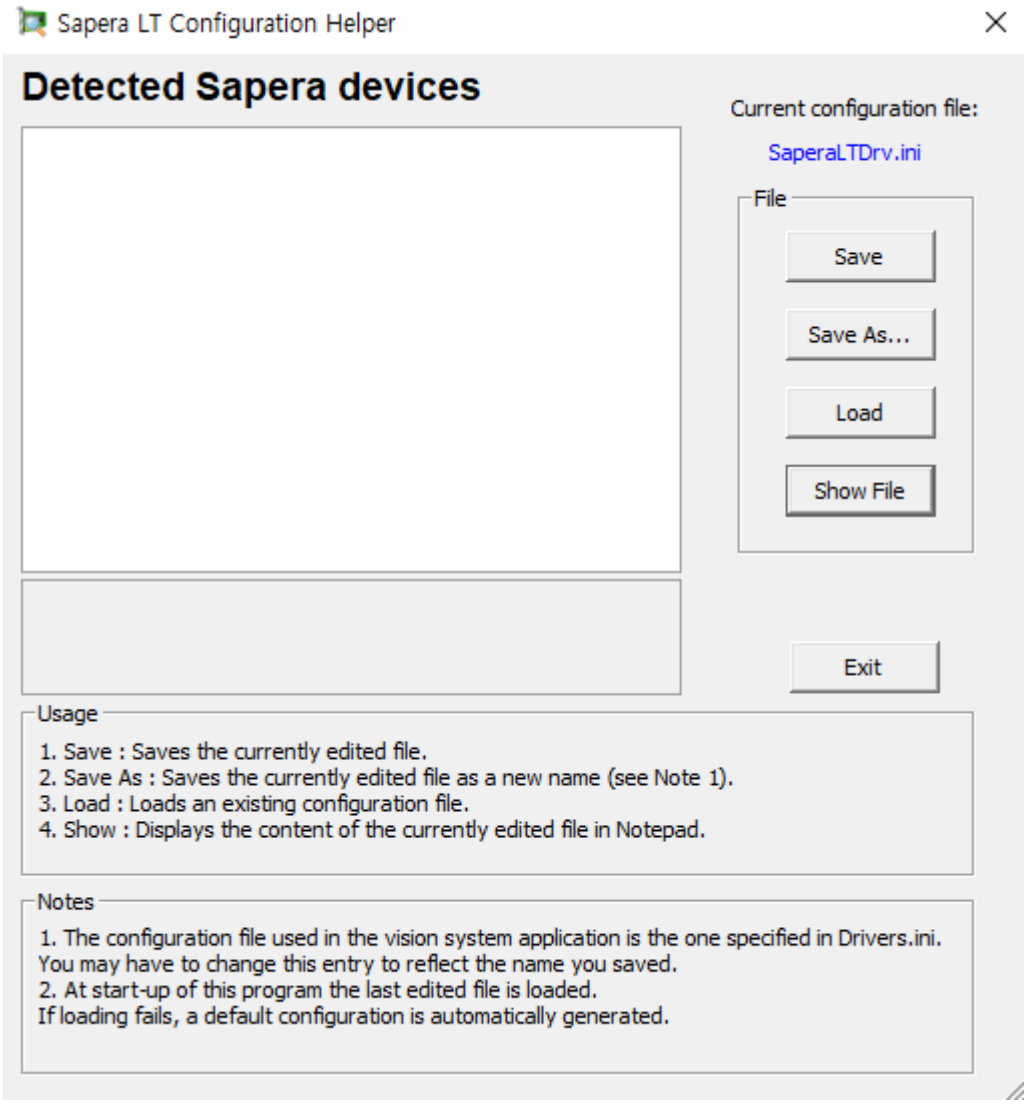
   GigE 카메라의 네트워크 설정을 아래의 그림과 같이 NetworkConfigurator를 이용하여 설정합니다. -자세한 사항은 카메라 제조사에 문의 바랍니다-

아래의 그림과 같이 Sapera CamExpert를 이용하여 카메라의 각종 파라미터들을 설정하여, 영상을 취득 테스트를 합니다. -자세한 사항은 카메라 제조사에 문의 바랍니다-



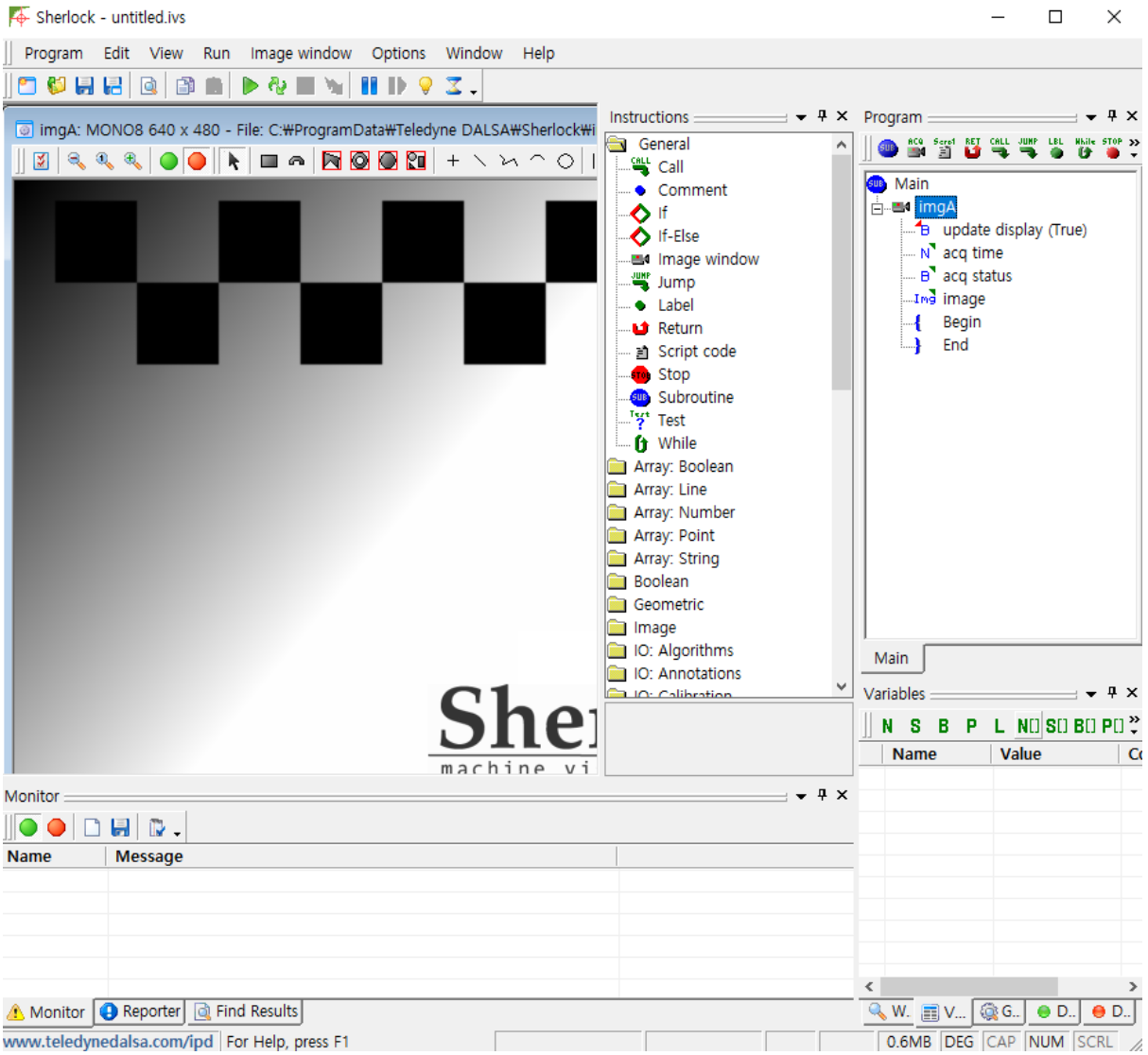
설정된 상태를 파일(.ccf)로 저장을 합니다.

Sapera Acquisition Wizard을 이용하여 Sherlock에서 사용할 카메라를 설정합니다. -자세한 사항은 카메라 제조사에 문의 바랍니다-

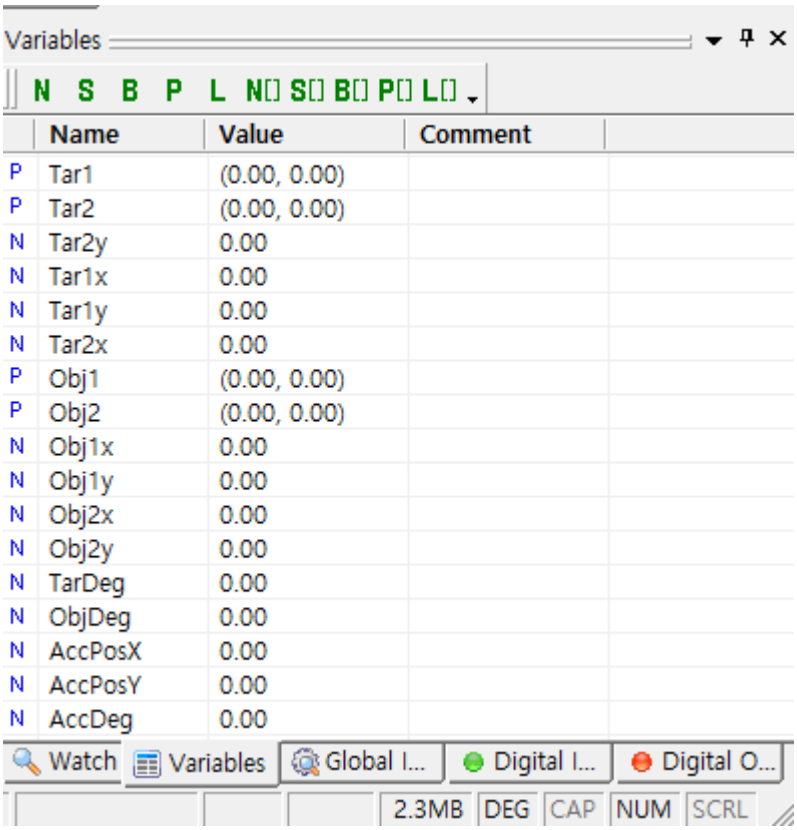


2. Sherlock을 이용한 비전 알고리즘 선정

   아래의 그림과 같이 Sherlock프로그램을 이용하여 비전 알고리즘 작업을 합니다.



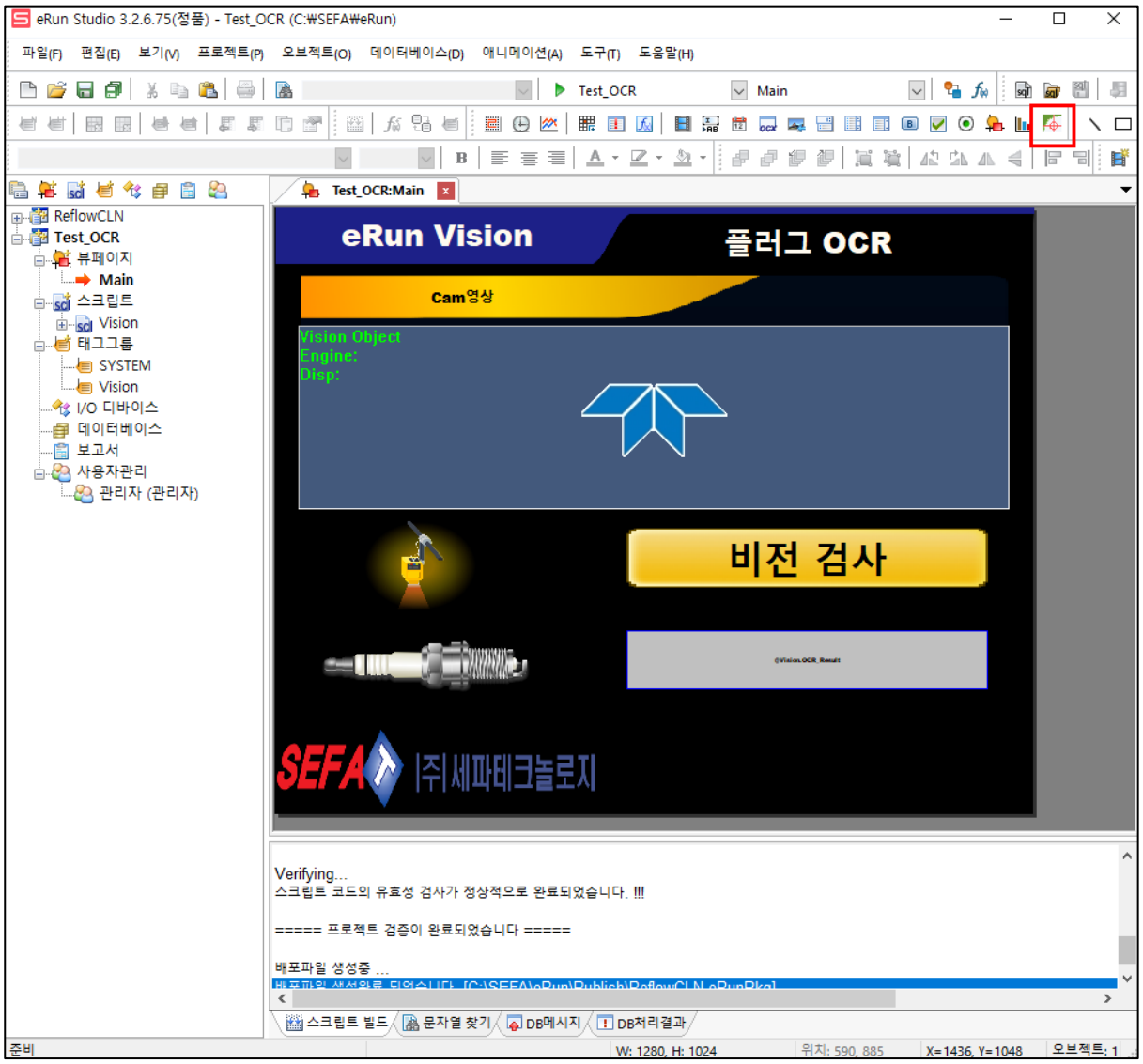
3. eRun과 데이터 공유를 위한 변수 설정
   아래의 그림과 같이 데이터 타입을 고려하여 변수들을 설정합니다.



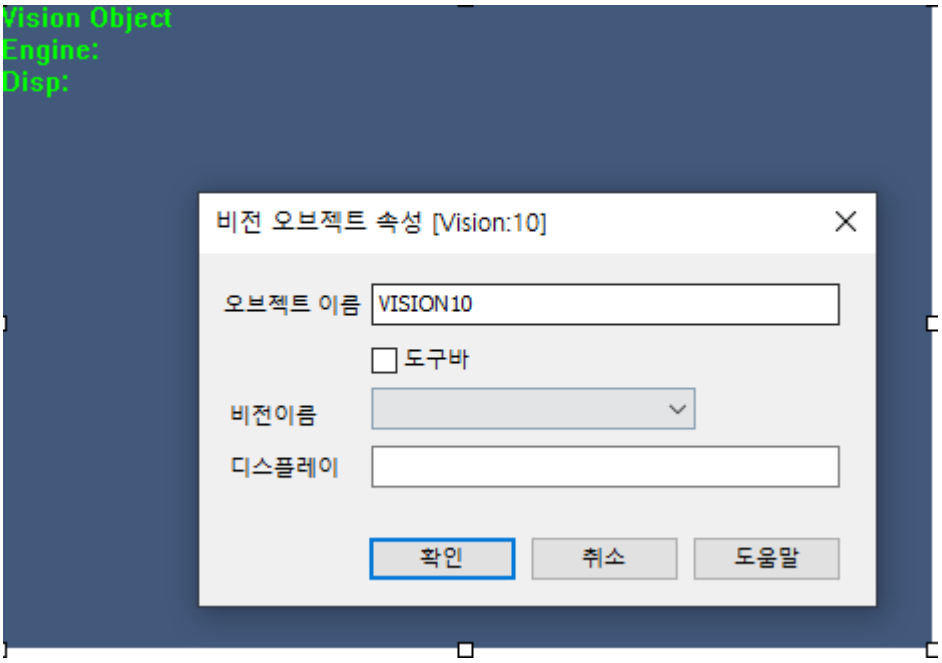
변수타입	비고
N	Number 숫자형 데이터
S	String 문자형 데이터
B	Bool 1또는 0의 값을 갖는 데이터
P	Point X,Y값을 쌍으로 갖는 데이터
L	Line 기울기와 절편을 갖는 데이터
N[]	Number의 배열형 데이터
S[]	String의 배열형 데이터
B[]	Bool의 배열형 데이터
P[]	Point의 배열형 데이터
L[]	Line의 배열형 데이터

4. eRun에서 작화 작업 수행

   eRun Studio를 이용하여 작화 작업을 합니다. 이때 영상 출력 및 비전 알고리즘 수행을 위한 eRunVsion Object를 이용합니다.

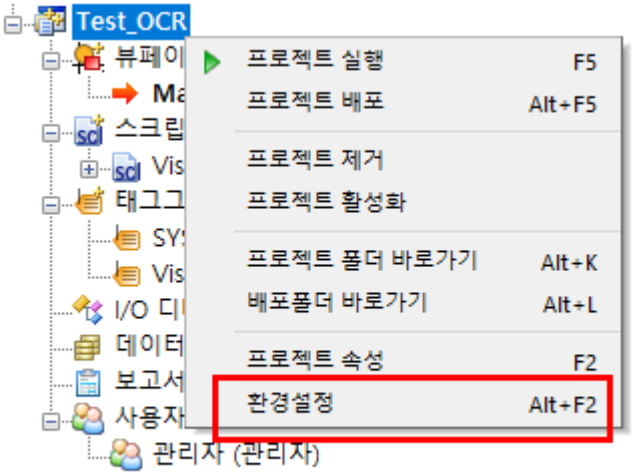


Vision Object를 더블 클릭하여 속성을 설정합니다.

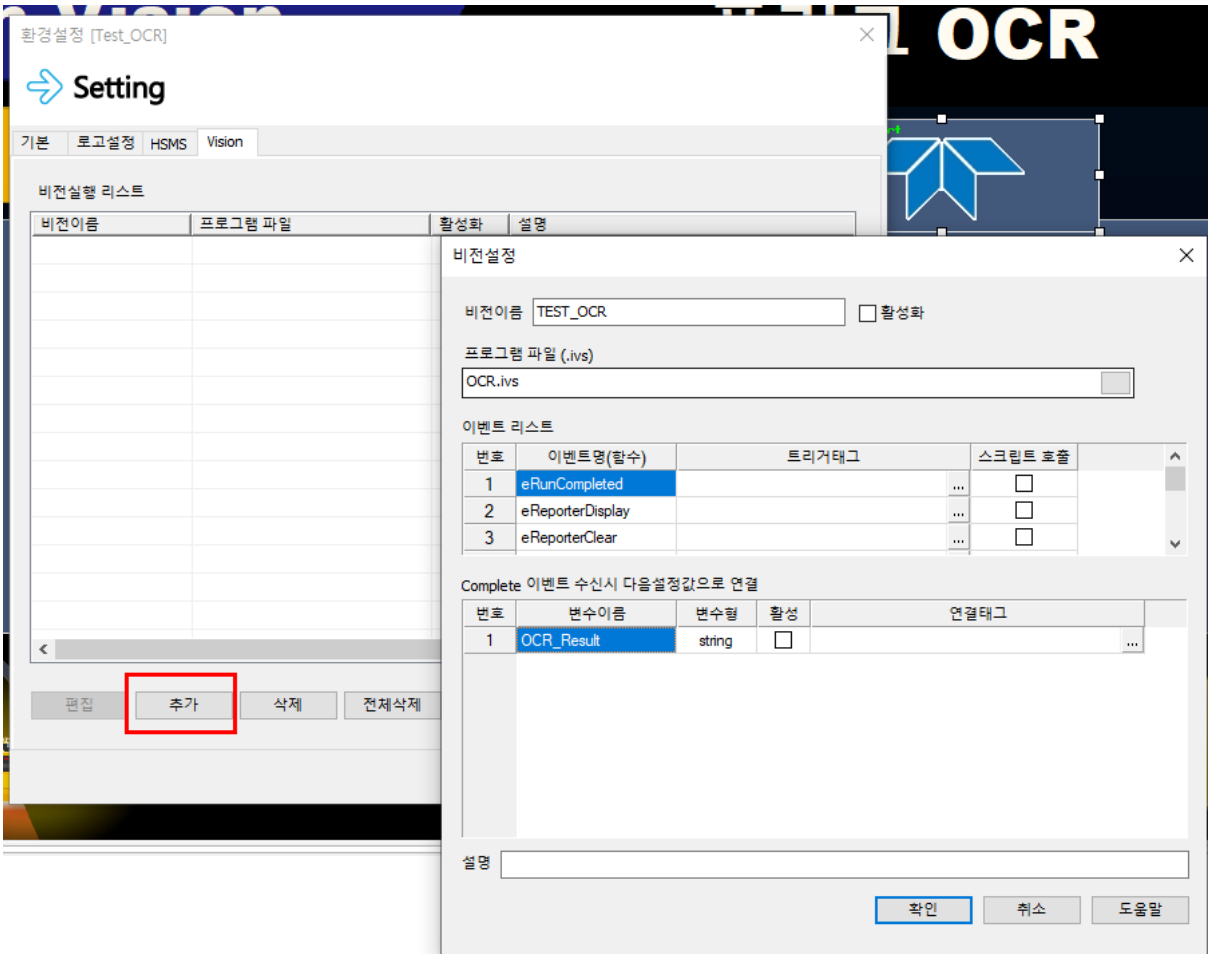


5. eRun과 Sherlock간의 변수 연결

아래의 그림과 같이 프로젝트리스트에서 오른쪽 버튼을 누르고 환경설정을 합니다.



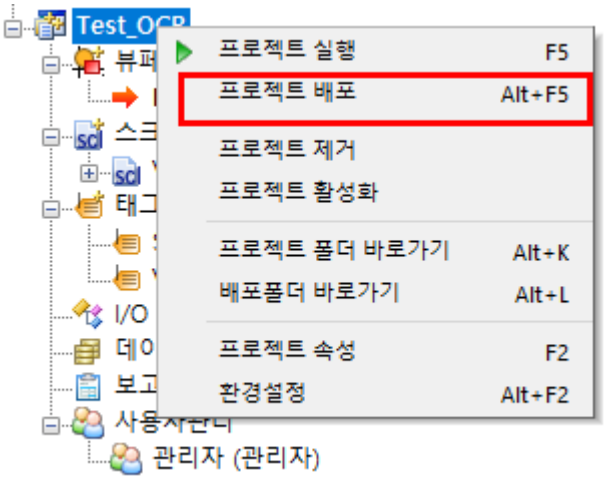
비전 탭을 선택한 후 앞서 작성한 ivs파일을 로딩합니다.



이때 ivs파일의 변수들이 자동으로 등록되며, eRun에서 사용할 태그와 연결합니다.

6. 최종 파일 생성

아래의 그림과 같이 배포파일을 만들어 최종 파일을 생성합니다.



기타 자세한 사항은 예제 프로젝트를 참고하시기 바랍니다.

2.22 COMIZOA 함수

2.22.1 _COMI_Method()

함수원형

int _COMI_Method(string strAPI, data1, data2, ...)

파라메터	strAPI : COMIZOA 라이브러리에서 제공하는 인터페이스 함수명 data1, data2 : 인터페이스 함수에 따라서 사용되는 파라메터 ... : strAPI 인터페이스 함수 파라메터 개수만큼 지정.
리턴형	오류코드
설명	커미조아㈜의 CMMSDK 라이브러리에서 제공하는 인터페이스 함수를 호출합니다.
함수활용	eRun에서 COMIZOA PCI보드 인터페이스 구현할 때 사용합니다.

※ 파라매터 설명부분은 COMIZOA 라이브러리(CMMSDK)에서 제공하는 함수 매뉴얼을 참고하시기 바랍니다.

※ 함수실행 반환값이 -1인 경우 라이브러리가 정상적으로 로드되지 않은경우입니다. 프로젝트 시작부분에서 **_COMI_Method("cmmLoadDll")**; 호출해 주어야 합니다. 그외의 음의수(0보다 작은수)가 나올경우 본 문서 뒤에 나오는 오류코드 리스트를 참고하시기 바랍니다.

※ eRun에서 지원하는 COMIZOA(CMMSDK 함수) 리스트

-. 파라미터중에서 변수앞에 [out]이 붙어있는 파라메터는 cmmSDK 라이브러리로부터 반환되는 변수라는 것을 의미합니다. 스크립트에서 함수실행 후 그변수값이 반환됩니다.

NO	함수리스트
1	**cmmLoadDll()** cmmSDK 라이브러리를 로드합니다. 스크립트에서 사용하려면 최소 1 회 실행합니다.
2	**cmmUnloadDll()** 라이브러리를 메모리에서 해제합니다.
3	**cmmGnDeviceLoad(int IsResetDevice, [out] int NumAxes)** 모션컨트롤러를 로드하고 초기화합니다.
4	**cmmGnDeviceUnload()** 모션컨틀롤러를 언로드하고 장치사용을 종료합니다.
5	**cmmGnDeviceIsLoaded([out] int IsLoaded)** 장치로드 상태확인
6	**cmmGnDeviceReset()** 장치를 리셋한 후에 초기화합니다.
7	**cmmGnInitFromFile(string sFile)** CME 파일을 통한 환경초기화

8	**cmmGnSetServoOn([in] int Axis, [in] int Enable)** 서보동작 Turn On/Off 신호출력 제어
9	**cmmGnGetServoOn([in] int Axis, [out] int Enable)** 현재의 SERVO-ON 신호의 출력상태 읽기
10	**cmmGnPulseAlarmRes(int Axis, int IsOnPulse, int dwDuration, int IsWaitPulseEnd)** 알람리셋 펄스출력 제어
11	**cmmGnSetAlarmRes(int Axis, int IsOn)** 알람리셋 신호출력제어
12	**cmmGnGetAlarmRes(int Axis, [out] int IsOn)** 알람리셋 신호의 출력상태 반환
13	**cmmGnSetEmergency(int IsEnable, int IsDecStop)** 소프트웨어적으로 모션컨트롤러를 Emergency 상태로 설정합니다.
14	**cmmGnGetEmergency([out] int IsEnabled)** 모션컨트롤러의 소프트웨어적인 Emergency 상태를 반환합니다.
15	**cmmCfgSetMioProperty(int Axis, int PropId, int PropVal)** 모션 입출력 신호에 대한 환경을 설정합니다.
16	**cmmCfgGetMioProperty(int Axis, int PropId, [out] int ProVal)** 모션 입출력신호에 대하여 현재 설정된 환경설정값을 반환합니다.
17	**cmmCfgSetFilter(int Axis, int IsEnable)** I/O 신호 노이즈 필터활성화
18	**cmmCfgGetFilter(int Axis, [out] int IsEnable)** I/O 신호 노이즈 필터활성화 상태 반환합니다.
19	**cmmCfgSetInMode(int Axis, int InputMode, int IsInverse)** 인코더 펄스신호 입력모드 설정
20	**cmmCfgGetInMode(int Axis, [out] int InputMode, [out] int IsInverse)** 인코더 펄스신호 입력모드 설정값 반환
21	**cmmCfgSetOutMode(int Axis, int OutputMode)** 펄스신호 출력모드 설정
22	**cmmCfgGetOutMode(int Axis, [out] int OutputMode)** 펄스신호 출력모드 설정값 반환
23	**cmmCfgSetInOutRatio(int Axis, double Ratio)** 입력펄스와 출력펄스의 분해능 설정
24	**cmmCfgGetInOutRatio(int Axis, [out] double Ratio)** 입력펄스와 출력펄스의 분해능 설정값 반환
25	**cmmCfgSetUnitDist(int Axis, double UnitDist)** 논리적 거리단위 설정
26	**cmmCfgGetUnitDist(int Axis, [out] double UnitDist)** 논리적 거리단위 설정값 반환
27	**cmmCfgSetSpeedRange(int Axis, double MaxPPS)**

	모션속도 제한 설정
28	**cmmCfgGetSpeedRange(int Axis, [out] double MinPPS, [out] double MaxPPS)** 모션속도 제한 설정값 반환
29	**cmmCfgSetUnitSpeed(int Axis, double UnitSpeed)** 논리적 단위속도에 대한 실제펄스 출력속도(PPS)를 설정합니다.
30	**cmmCfgGetUnitSpeed(int Axis, [out] double UnitSpeed)** 논리적 단위속도에 대한 실제펄스 출력속도(PPS)값을 반환합니다.
31	**cmmCfgSetSpeedPattern(int Axis, int SpeedMode, double WorkSpeed, double Accel, double Decel)** 지정한 축에 대해 속도 모드, 작업속도 및 가속 및 감속도를 설정합니다.
32	**cmmCfgGetSpeedPattern([in] int Axis, [out] int SpeedMode, [out] double WorkSpeed, [out] double Accel, [out] double Decel)** 지정한 축에 대해 속도 모드, 작업속도 및 가속 및 감속도를 설정값을 반환합니다.
33	**cmmSxSetSpeedRatio(int Axis, int SpeedMode, double VelRatio, double AccRatio, double DecRatio)** 단축이송 속도비율 설정
34	**cmmSxGetSpeedRatio(int Axis, [out] int SpeedMode, [out] double VelRatio, [out] double AccRatio, [out] double DecRatio)** 단축이송 속도비율 설정값 반환
35	**cmmSxMoveStart(int Axis, double Distance)** 해당축에 대하여 지정한 거리만큼 이동수행. 모션을 시작시킨후 바로 반환합니다.
36	**cmmSxMove(int Axis, [in] double Distance, int IsBlocking)** 해당축에 대하여 지정한 거리만큼 이동수행. 파라메터 IsBlocking 이 TRUE 인경우 모션이 완료될때까지 반환하지 않습니다.
37	**cmmSxMoveToStart(int Axis, double Position)** 해당축에 대하여 지정한 절대좌표 이동수행. 모션을 시작시킨후 바로 반환합니다.
38	**cmmSxMoveTo(int Axis, double Position, int IsBlocking)** 해당축에 대하여 지정한 절대좌표 이동수행. 파라메터 IsBlocking 이 TRUE 인경우 모션이 완료될때까지 반환하지 않습니다.
39	**cmmSxVMoveStart(int Axis, int Dir)** 작업속도까지 가속한 후에 작업속도를 유지하며 정지함수가 호출될때까지 지정한 방향으로 모션을 계속수행 합니다. 이 함수는 모션시작후 바로 반환합니다. Jog수행 할 때 사용합니다.
40	**cmmSxStop(int Axis, int IsWaitComplete, int IsBlocking)** 단축 이송정지. 정지시 감속후 정지수행합니다.
41	**cmmSxStopEmg(int Axis)** 단축 이송비상정지. 감속없이 즉시정지 수행합니다.
42	**cmmSxIsDone(int Axis, [out] int IsDone)** 해당 축에 대하여 모션완료 확인
43	**cmmSxWaitDone(int Axis, int IsBlocking)**

<table>
<tr><td></td><td>해당축에 대하여 모션완료될 때까지 기다립니다.</td></tr>
<tr><td>44</td><td>cmmStSetCount(int Axis, int Target, int Count)
사용자정의 하드웨어 카운트값(펄스수) 설정</td></tr>
<tr><td>45</td><td>cmmStGetCount (int Axis, int Source, [out] int Count)
사용자정의 하드웨어 카운트값 반환</td></tr>
<tr><td>46</td><td>cmmStSetPosition(int Axis, int Target, double Position)
사용자정의 논리적 카운트값 설정</td></tr>
<tr><td>47</td><td>cmmStGetPosition(int Axis, int Source, [out] double Position)
사용자정의 논리적 카운트값 반환</td></tr>
<tr><td>48</td><td>cmmStGetSpeed(int Axis, int Source, [out] double Speed)
논리적속도 반환</td></tr>
<tr><td>49</td><td>cmmStReadMotionState(int Axis, [out] int MotStates)
모션 동작상태 반환. 2 번째 변수 MotStates 값의미.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>음수</td><td>모션작업 도중 에러 발생 => 에러코드 참조.</td></tr>
<tr><td>0 또는 cmMST_STOP</td><td>Stop</td></tr>
<tr><td>1 또는 cmMST_WAIT_DR</td><td>Waiting for DR input signal</td></tr>
<tr><td>2 또는 cmMST_WAIT_STA</td><td>Waiting for STA input signal</td></tr>
<tr><td>3 또는 cmMST_WAIT_INSYNC</td><td>Waiting for internal sync. signal</td></tr>
<tr><td>4 또는 cmMST_WAIT_OTHER</td><td>Waiting other axis</td></tr>
<tr><td>5 또는 cmMST_WAIT_ERC</td><td>Waiting for ERC output finished</td></tr>
<tr><td>6 또는 cmMST_WAIT_DIR</td><td>Waiting for DIR change (DIR 은 외부입력 신호가 아닌 내부적인 신호임)</td></tr>
<tr><td>7 또는 cmMST_RESERVED1</td><td>Reserved</td></tr>
<tr><td>8 또는 cmMST_WAIT_PLSR</td><td>Waiting for PA/PB input signal</td></tr>
<tr><td>9 또는 cmMST_IN_RVSSPD</td><td>In home special speed (reverse speed)</td></tr>
<tr><td>10 또는 cmMST_IN_INISPD</td><td>In start velocity motion</td></tr>
<tr><td>11 또는 cmMST_IN_ACC</td><td>In acceleration</td></tr>
<tr><td>12 또는 cmMST_IN_WORKSPD</td><td>In working velocity</td></tr>
<tr><td>13 또는 cmMST_IN_DEC</td><td>In deceleration</td></tr>
<tr><td>14 또는 cmMST_WAIT_INP</td><td>Waiting for INP input signal</td></tr>
<tr><td>15 또는 cmMST_SPARE0</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>50</td><td>cmmStReadMioStatuses(int Axis, [out] int MotStates)
모션 MIO 상태 반환. 2 번째 변수 MotStates 값의미.</td></tr>
</table>

Bit No.	Name	Meaning
0 (cmIOST_RDY)	RDY	Servo ready signal input status(1=ON)
1 (cmIOST_ALM)	ALM	Alarm signal status(1=ON)
2 (cmIOST_ELP)	+EL	Positive limit switch status(1=ON)
3 (cmIOST_ELN)	-EL	Negative limit switch status(1=ON)
4 (cmIOST_ORG)	ORG	Origin switch status(1=ON)
5 (cmIOST_DIR)	DIR	Operating direction status(1=ON)
6 (cmIOST_RSV1)	Reserved	
7 (cmIOST_PCS)	PCS	PCS signal input status(1=ON)
8 (cmIOST_ERC)	ERC	ERC pin output status(1=ON)
9 (cmIOST_EZ)	EZ	Index signal status(1=ON)
10 (cmIOST_CLR)	CLR	Clear input status(1=ON)
11 (cmIOST_LTC)	Latch	Latch signal input status(1=ON)
12 (cmIOST_SD)	SD	Slow Down signal input status(1=ON)
13 (cmIOST_INP)	INP	In-Position signal input status(1=ON)
14 (cmIOST_DRP)	DRP	+DR input signal status(1=ON)
15 (cmIOST_DRN)	DRN	-DR input signal status(1=ON)
16 (cmIOST_STA)	STA	STA input signal status(1=ON)
17 (cmIOST_STP)	STP	STP input signal status(1=ON)
18 ~31	Reserved	

51	**cmmStGetMstString(int MstCode, [out] string Buffer, int BufferLen)** 모션 동작상태와 관련된 문자열 반환합니다.
52	**cmmCmpTrgSetConfig(int Axis, int CmpSrc, int CmpMethod)** 위치비교기 조건을 설정합니다.
53	**cmmCmpTrgGetConfig(int Axis, int CmpSrc, [out] int CmpMethod)** 위치비교기 조건을 설정된 값을 반환합니다.
54	**cmmCmpTrgSetOneData(int Axis, double Data)** 위치비교기에 1 회의 비교데이터 설정합니다.
55	**cmmCmpTrgGetCurData(int Axis, [out] double Data)** 위치비교출력기에 현재 설정된 비교데이터를 반환합니다.
56	**cmmCmpTrgContRegTable(int Axis, [array] double Buffer, int NumData)** 연속적인 비교위치 데이터를 등록합니다. 2 번째 파라메터 [array]는 NumData 개수만큼 배열변수를 의미합니다.
57	**cmmCmpTrgContBuildTable(int Axis, double StartData, double Interval, int NumData)** 일정간격의 비교위치 데이터를 등록합니다. 연속적인 위치 비교 출력 기능을 사용하기 위해서 일정한 위치 간격을 가지는 연속적인 위치 데이터를 자동으로 생성하여 등록 (登錄) 하도록 합니다. 이 함수는 일정한 위치 간격으로 CMP 출력을 내보낼 때 cmmCmpTrgContRegTable() 함수 대신에 사용할 수 있습니다.
58	**cmmCmpTrgContStart(int Axis)** 연속위치비교 출력기능을 시작합니다.
59	**cmmCmpTrgContStop(int Axis)** 연속위치비교 출력기능을 종료합니다.
60	**cmmCmpTrgContIsActive(int Axis, [out] int IsActive)** 지정된 축의 위치 비교 출력 상태를 반환합니다.`

61	**cmmHomeSetConfig(int Axis, int HomeMode, int EzCount, double EscDist, double Offset)** 원점복귀 환경설정
62	**cmmHomeGetConfig(int Axis, [out] int HomeMode, [out] int EzCount, [out] double EscDist, [out] double Offset)** 원점복귀 환경설정값 반환.
63	**cmmHomeSetSpeedPattern(int Axis, int SpeedMode, double Vel, double Accel, double Decel, double RevVel)** 원점복귀 속도설정
64	**cmmHomeGetSpeedPattern(int Axis, [out] int SpeedMode, [out] double Vel, [out] double Accel, [out] double Decel, [out] double RevVel)** 원점복귀 속도설정값 반환
65	**cmmHomeMove(int Axis, int Direction, int IsBlocking)** 원점복귀 작업수행을 합니다. 이함수는 모션이 완료되기 전까지 반환하지 않습니다.
66	**cmmHomeMoveStart(int Axis, int Direction)** 원점복귀 작업수행을 하며, 함수 호출이후 즉시 반환합니다.
67	**cmmHomeMoveAll(int NumAxes, [array] int ChannelList, [array] int DirList, int IsBlocking)** 다축 원점복귀 수행. 함수 호출이후 모션완료 전까지 반환하지 않습니다. 2,3 번 파라메터는 축번호, 축이동방향을 배열변수로 사용합니다.
68	**cmmHomeMoveAllStart(int NumAxes, [array] int ChannelList, [array] int DirList)** 다축 원점복귀 수행을 하며, 함수 호출이후 즉시 반환합니다. 2,3번 파라메터는 축번호, 축이동방향을 배열변수로 사용합니다.
69	**cmmHomeIsBusy(int Axis, [out] int IsBusy)** 원점 복귀 모션진행 상태 반환.
70	**cmmHomeWaitDone(int nAxis, int IsBlocking)** 원점 복귀 완료될때까지 기다립니다.
71	**cmmHomeSetSuccess(int nAxis, int IsSuccess)** 원점 복귀의 성공여부에 대한 플래그 값을 강제로 설정합니다.
72	**cmmHomeGetSuccess(int nAxis, [out] int IsSuccess)** 현재 원접복귀가 성공적으로 완료되었는지 반환합니다.
73	**cmmLmMapAxes (int LmIndex, int MapMask1, int MapMask2)** Listed Motion 에서 사용되는 모든 축들을 지정한 Index 로 등록합니다. 32Bit 형 전달 인자 2 개를 통해 최대 64개의 모션 채널을 대상(對象)으로 리스트 모션에 사용될 수 있습니다.
74	**cmmLmBeginList (int LmIndex)** Listed Motion 에서 사용되는 함수의 등록(登錄)을 시작합니다. 이 함수가 시작 된 이후(以後)로는 리스트모션을 지원하는 함수는 cmmLmEndList 함수를 사용할 때까지 리스트 모션 대상(對象) 함수로서의 자격으로 추가(追加)됩니다.
75	**cmmLmEndList (int LmIndex)** Listed Motion 에서 사용되는 함수의 등록을 종료(終了)합니다. cmmLmBeginList 함수와 cmmLmEndList 함수사이에 있는 존재하는 리스트 모션을 지원하는 함수의 등록을 명시적으로

	완료(完了)하게 됩니다.
76	**cmmLmIsDone (int LmIndex, [out] int IsDone)** Listed Motion 의 동작의 완료(完了) 상태를 확인(確認)합니다.
77	**cmmLmWaitDone (int LmIndex, int IsBlocking)** Listed Motion 의 동작의 완료(完了)를 위해 대기(待機) 합니다.
78	**cmmLmCurSequence (int LmIndex, [out] int SeqIndex)** Listed Motion 에서 등록(登錄)된 작업 중에서 현재 수행되고 있는 작업의 번호를 반환합니다.
79	**cmmLmImmediacySet (int LmIndex)** Listed Motion 의 추후(追後)의 추가(追加) 기능을 위해 예약(豫約)된 함수입니다.
80	**cmmLmStartMotion (int LmIndex)** Listed Motion 에서 등록(登錄)된 작업을 대상으로 실제 Listed Motion 을 수행(遂行)합니다.
81	**cmmLmAbortMotion (int LmIndex)** Listed Motion 을 수행(遂行)하고 있는 상황에서 현재 실행중인 Listed Motion 을 중단(中斷)합니다.
82	**cmmLmDoPutOne (int LmIndex, int hDoDevice, int Channel, int OutState)** Listed Motion 에서 단일(單一) 채널을 대상으로 디지털 출력(出力) 명령을 추가합니다. 실제 Listed Motion 수행(遂行)시에 한 개의 채널에 대해서 지정한 디지털 출력(出力) 명령을 수행(遂行)합니다. 이 함수에서 사용되는 출력(出力) 채널은 모션 보드 상의 지역(地域) 출력(出力) 채널입니다.
83	**cmmLmDoPutMulti (int LmIndex, int hDoDevice, int ChannelGroup, int Mask, int OutStates)** Listed Motion 에서 다중(多重) 채널을 대상으로 디지털 출력 명령을 추가합니다. 실제 Listed Motion 수행(遂行)시에 다수의 채널에 대해서 지정한 디지털 출력 명령을 수행합니다. 이 수에서 사용되는 출력채널은 모션 보드 상의 지역(地域) 출력 채널입니다.
84	**cmmLmDoPulseOne (int LmIndex, int hDoDevice, int Channel, int OutState, int Duration)** Listed Motion 에서 단일(單一) 채널을 대상으로 디지털 펄스 신호 출력(出力) 명령을 추가합니다. 실제 Listed Motion 수행(遂行)시에 단일의 채널에 대해서 지정한 디지털 펄스 신호 출력(出力) 명령을 수행합니다. 이 함수에서 사용되는 출력(出力) 채널은 모션 보드 상의 지역(地域) 출력 채널입니다.
85	**cmmLmDoPulseMulti (int LmIndex, int hDoDevice, int ChannelGroup, int Mask, int OutStates, int Duration)** Listed Motion 에서 다중(多重) 채널을 대상으로 디지털 펄스 신호 출력(出力) 명령을 추가합니다. 실제 Listed Motion 수행시에 다중의 채널에 대해서 지정한 디지털 펄스 신호 출력(出力) 명령을 수행합니다. 이 함수에서 사용되는 출력(出力) 채널은 모션 보드 상의 지역(地域) 출력 채널입니다.
86	**cmmLmxStart (int LmIndex, int AxisMask1, int AxisMask2)** Extend Listed Motion 에서 등록(登錄)된 작업을 대상으로 실제 Listed Motion 을 수행(遂行)합니다.
87	**cmmLmxPause (int LmIndex)** Extend Listed Motion 수행 중 명령 시퀀스를 일시 정지하고 싶을 때 사용되는 함수입니다.
88	**cmmLmxResume (int LmIndex, int IsClearQue)** Extend Listed Motion 수행 중 cmmLmxPause() 명령에 의해 일시 정지된 명령 시퀀스를 재개하고자 할 때 사용합니다.
89	**cmmLmxEnd (int LmIndex)**

	Extend Listed Motion 작업을 종료 합니다.
90	**cmmLmxSetSeqMode (int LmIndex, int SeqMode)** Extend Listed Motion 수행 중에 새로운 이송 명령을 예약하려 하는데 이미 명령 버퍼(Extend Listed Motion Buffer) 가 이미 꽉 차있는 경우에 어떻게 처리할 지에 대한 모드를 설정합니다.
91	**cmmLmxGetSeqMode (int LmIndex, int SeqMode)** Extend Listed Motion 수행 중에 새로운 이송 명령을 예약하려 하는데 이미 명령 버퍼(Extend Listed Motion Buffer) 가 이미 꽉 차있는 경우에 어떻게 처리할 지에 대해 설정된 모드를 반환합니다.
91	**cmmLmxSetNextItemId (int LmIndex, int SeqId)** Extend Listed Motion 에서 수행할 명령(Item)에 대해 Sequence Item Id 를 설정합니다.
92	**cmmLmxGetNextItemId (int LmIndex, int SeqId)** Extend Listed Motion 에서 다음 수행할 명령(Item)에 해당하는 Sequence Item Id 를 반환합니다.
93	**cmmLmxSetNextItemParam (int LmIndex, int ParamIdx, int ParamData)** Extend Listed Motion 에서 다음 수행 예정인 명령에 대한 함수 파라미터 설정 값 설정합니다.
94	**cmmLmxGetNextItemParam (int LmIndex, int ParamIdx, [out] int ParamData)** Extend Listed Motion 에서 다음 수행 예정인 명령에 대한 함수 파라미터 설정 값 반환합니다.
95	**cmmLmxGetRunItemParam (int LmIndex, int ParamIdx, [out] int ParamData)** Extend Listed Motion 에서 현재 수행 중인 명령에 대한 함수 파라미터 설정 값 반환합니다.
96	**cmmLmxGetRunItemStaPos(int LmIndex, int Axis, [out] double Position)** Extend Listed Motion 수행 중에 현재 수행 중인 명령(Current Sequence Item Id) 이 수행되기 직전에 해당 축의 Command Pulse Position 값을 반환 합니다.
97	**cmmLmxGetRunItemTargPos(int LmIndex, int Axis, [out] double Position)** Extend Listed Motion 수행 중에 현재 수행 중인 명령(Current Sequence Item Id) 에 대해 해당 축의 목표 좌표에 해당하는 Command Pulse Position 값을 반환 합니다.
98	**cmmLmxGetSts(int LmIndex, int LmxStsId, [out] int LmxStsVal)** Extend Listed Motion 에서 현재 수행 중인 Sequence 명령에 대한 상태 값을 Status Id 값을 통해 확인이 가능한 함수 입니다.
99	**cmmDiSetInputLogic ([in] VT_I4 Channel, [in] VT_I4 InputLogic)** 대상(對象) 디지털 입력(Digital Input) 채널의 입력 논리(Input Logic)를 설정(設定)합니다
100	**cmmDiGetInputLogic ([in] VT_I4 Channel, [out] VT_PI4 InputLogic)** 대상(對象) 디지털 입력(Digital Input) 채널의 입력 논리(Input Logic)의 설정 상태를 반환(返還)합니다
101	cmmDiGetOne ([in] VT_I4 Channel, [out] VT_PI4 InputState) 대상(對象) 디지털 입력(Digital Input) 채널의 단일(單一) 디지털입력(入力) 채널의 상태를 반환(返還)합니다.
102	**cmmDiGetMulti ([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [out] VT_PI4 InputStates)** 대상(對象) 디지털 입력(Digital Input) 채널 범위의 다중(多重) 디지털입력(入力) 채널의 상태를 반환(返還)합니다.
103	**cmmDoSetOutputLogic ([in] VT_I4 Channel, [in] VT_I4 OutputLogic)** 대상(對象) 디지털 출력(Digital Output) 채널의 출력 논리(Output Logic)를 설정(設定)합니다
104	**cmmDoGetOutputLogic ([in] VT_I4 Channel, [out] VT_PI4 OutputLogic)** 대상(對象) 디지털 출력(Digital Output) 채널의 출력 논리(Output Logic)의 설정(設定) 상태를 반환(返還)합니다.

105	**cmmDoPutOne ([in] VT_I4 Channel, [in] VT_I4 OutState)** 대상(對象) 디지털 출력(Digital Output) 채널의 단일(單一) 디지털 채널을 통해 디지털 출력(Digital Output)을 발생시킵니다.
106	**cmmDoGetOne ([in] VT_I4 Channel, [out] VT_PI4 OutState)** 대상(對象) 디지털 출력(Digital Output) 채널의 단일(單一) 디지털 채널을 통해 디지털 출력(Digital Output) 상태를 반환합니다.
107	**cmmDoPutMulti ([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [in] VT_I4 OutputStates)** 대상(對象) 디지털 출력(Digital Output) 채널 범위의 다중(多重) 디지털 출력 채널을 통해 동시에 디지털 출력(Digital Output)을 발생시킵니다
108	**cmmDoGetMulti ([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [out] VT_PI4 OutputStates)** 대상(對象) 디지털 출력(Digital Output) 채널 범위의 다중(多重) 디지털 채널을 통해 동시에 디지털 출력(Digital Output) 상태를 확인(確認)합니다
109	**cmmDoPulseOne ([in] VT_I4 Channel, [in] VT_I4 IsOnPulse, [in] VT_I4 dwDuration, [in] VT_I4 IsWaitPulseEnd)** 대상(對象) 디지털 출력(Digital Output) 채널의 단일(單一) 디지털 채널을 통해 단일 펄스 출력(出力)을 발생시킵니다
110	**cmmDoPulseMulti ([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [in] VT_I4 OutStates, [in] VT_I4 dwDuration, [in] VT_I4 IsWaitPulseEnd)** 대상(對象) 디지털 출력(Digital Output) 채널 범위의 지정한 다중(多重) 디지털 채널을 통해 단일 펄스 출력(出力)을 발생시킵니다.
111	**cmmDiGetOneF ([in] VT_I4 Channel, [in] VT_I4 CutoffTime_us, [out] VT_PI4 InputState)** 대상(對象) 디지털 입력(Digital Input) 채널의 단일(單一) 채널을 대상으로 노이즈 필터(Noise Filter) 기능을 활성화 하여, 디지털 입력(入力) 상태를 반환(返還)합니다.
112	**cmmDiGetMultiF ([in] VT_I4 IniChannel, [in] VT_I4 NumChannels, [in] VT_I4 CutoffTime_us, [out] VT_PI4 InputStates)** 대상(對象) 디지털 입력(Digital Input) 채널 범위의 다중(多重) 채널을 대상으로 노이즈 필터(Noise Filter) 기능을 활성화 하여, 디지털 입력(入力) 상태를 반환(返還)합니다

다음은 _COMI_Method 호출후 반환되는 함수실행결과 코드설명입니다.

code	macro	description
0	cmERR_NONE	No Error
-290	cmERR_MEM_ALLOC_FAIL	Memory allocation fail
-292	cmERR_GLOBAL_MEM_FAIL	Global memory allocation fail
-310	cmERR_ISR_CONNEC_FAIL	ISR registration fail
-400	cmERR_DIVIDE_BY_ZERO	Cause divide by zero error
-500	cmERR_WORNG_NUM_DATA	Number of data is too small or too big
-600	cmERR_VER_MISMATCH	Version(of file or device) mismatch
-1010	cmERR_INVALID_DEVICE_ID	Invalid device id => Load Device 또는 SetDeviceId()에서...
-1020	cmERR_INVALID_HANDLE	
-1030	cmERR_UNSUPORTED_FUNC	
-1101	cmERR_INVALID_PARAMETER	
-1105	cmERR_INVALID_CHANNEL	
-1111	cmERR_INVALID_INPUT_RANGE	Invalid range value (AI, AO)
-1121	cmERR_INVALID_FREQ_RANGE	Invalid input or output frequency
-1501	cmERR_FILE_CREATE_FAIL	File create fail
-1511	cmERR_FILE_OPEN_FAIL	File open fail
-1522	cmERR_FILE_READ_FAIL	File reading fail
-1550	cmERR_EVENT_CREATE_FAIL	Event handle creation fail
-1560	cmERR_INT_INSTANCE_FAIL	Interrupt event instance creation fail
-1570	cmERR_DITHREAD_CRE	D/I state change monitor thread creation fail
-1580	cmERR_BUFFER_SMALL	Buffer size is too small
-1590	cmERR_HIGH_TIMER_UNSUPP	The installed hardware does not support a high-resolution performance counter (cmmUtlDelayMicroSec() function fails)
-1600	cmERR_OUT_OF_RANGE	The range of some parameter is out of range
-1610	cmERR_INVALID_BUFFER	buffer pointer is NULL
-1620	cmERR_INVALID_BUF_SIZE	buffer size if too big
-1630	cmERR_LTCQ_ERROR_BASE	
-1631	cmERR_LTCQ_NULL_BUFFER	Queue buffer is not allocated (refer to cmmLtcQue_SetSize() function)
-1632	cmERR_LTCQ_BUF_EMPTY	There's no new latch data in the queue berffer
-1633	cmERR_LTCQ_INVALID_INDEX	'Index' argument of cmmLtcQue_PeekAt() function is invalid
-5001	cmERR_ON_MOTION	
-5002	cmERR_STOP_BY_SLP	Abnormally stopped by positive soft limit
-5003	cmERR_STOP_BY_SLN	Abnormally stopped by negative soft limit
-5004	cmERR_STOP_BY_CMP3	Abnormally stopped by comparator3

-5005	cmERR_STOP_BY_CMP4	Abnormally stopped by comparator4
-5006	cmERR_STOP_BY_CMP5	Abnormally stopped by comparator5
-5007	cmERR_STOP_BY_ELP	Abnormally stopped by (+) external limit
-5008	cmERR_STOP_BY_ELN	Abnormally stopped by (-) external limit
-5009	cmERR_STOP_BY_ALM	Abnormally stopped by alarm input signal
-5010	cmERR_STOP_BY_CSTP	Abnormally stopped by CSTP input signal
-5011	cmERR_STOP_BY_CEMG	Abnormally stopped by CEMG input signal
-5012	cmERR_STOP_BY_SD	Abnormally stopped by SD input signal
-5013	cmERR_STOP_BY_DERROR	Abnormally stopped by operation data error
-5014	cmERR_STOP_BY_IP	Abnormally stopped by other axis error during interpolation
-5015	cmERR_STOP_BY_PO	An overflow occurred in the PA/PB input buffer
-5016	cmERR_STOP_BY_AO	Out of range position counter during interpolation
-5017	cmERR_STOP_BY_EE	An EA/EB input error occurred (does not stop)
-5018	cmERR_STOP_BY_PE	An PA/PB input error occurred (does not stop)
-5019	cmERR_STOP_BY_SLVERR	Abnormally stopped because slave axis has been stopped
-5020	cmERR_STOP_BY_SEMG	Abnormally stopped by software emergency setting
-5110	cmERR_MOT_MAOMODE	Master output mode is not CW/CCW mode Master/Slave 동작시에 Master output 모드가 CW/CCW 모드가 아니다.
-5120	cmERR_MOT_SLAVE_SET	Slave start fail (Motion state 가 "Wait for Pulsar Input"으로 변하지 않는다.
-5130	cmERR_SPEED_RANGE_OVER	
-5140	cmERR_INVALID_SPEED_SET	Speed setting value is not valid
-5150	cmERR_INVALID_IXMAP	Invalid interpolation map
-5160	cmERR_INVALID_LMMAP	Invalid List-Motion Map
-5170	cmERR_MOT_SEQ_SKIPPED	Motion command is skipped because the axis is already running.
-5180	cmERR_CMPIX_INVALID_MAP	Interpolated position compare output map is not valid
-5190	cmERR_INVALID_ARC_POS	Position data for circular interpolation is invalid
-5200	cmERR_LMX_ADD_ITEM_FAIL	failed to add an job item to "extend list motion"
-5300	cmERR_LMX_IS_NOT_ACTIVE	'Extended ListMotion' is not active.
-9999	cmERR_UNKNOWN	

※ 스크립트 작성예시

```
// 라이브러리 로드
void COMI_Start()
{
	int ret;
```

```
	int nAxes;

	ret = _COMI_Method("cmmLoadDll");
	_TraceEx("cmmGnLoadDll ret %d", ret);

	ret = _COMI_Method("cmmGnDeviceLoad", 1, nAxes);
	_TraceEx("cmmGnDeviceLoad ret %d, axes=%d", ret, nAxes);
}

// 라이브러리 로드해제
void COMI_End()
{
	int ret;
	int nAxes;

	ret = _COMI_Method("cmmGnDeviceUnload");
	_TraceEx("cmmGnDeviceUnload ret %d", ret);

	ret = _COMI_Method("cmmUnloadDll");
	_TraceEx("cmmGnUnloadDll ret %d", ret);
}
```

```
[2022-08-12 14:55:11.672]  cmmGnLoadDll ret 1
[2022-08-12 14:55:11.686]  cmmGnDeviceLoad ret 0, axes=0
[2022-08-12 14:55:13.701]  cmmGnDeviceUnload ret 0
[2022-08-12 14:55:13.705]  cmmGnUnloadDll ret 0
```

제 3 장 SQL (Structured Query Language)

■ SELECT 명령문
// tableName에 해당하는 테이블의 모든 열을 읽는다.
> select * from tablename;

// 위와 동일 (대소문자 구별 X)
> SELECT * from tableName;

예약어와 DB 객체명은 대소문자를 구별하지 않는다.
검색 조건 지정

열 지정
> select 열1, 열2 from 테이블명 where 조건식
> SELECT address, name FROM sample;

행 지정
// select 구 -> from 구 -> where 구 (순서 중요)
> select 열 from 테이블명 where 조건식

// 수치형
> SELECT * FROM sample WHERE id=2; // id 열 값이 2와 동일한 행만 검색
> SELECT * FROM sample WHERE id<>2; // id 열 값이 2가 아닌 행만 검색

// 문자열형
> SELECT * FROM sample WHERE name='아무개';

// NULL값
> SELECT * FROM sample WHERE birthday IS NULL;
where 구(조건식)는 '열과 연산자, 상수로 구성되는 식'이다.
검색 조건 조합
AND, OR, NOT
> 조건식1 AND 조건식2
> SELECT * FROM sample WHERE a<>0 AND b<>0; // a열, b열이 모두 0이 아닌 행 검색

> 조건식1 OR 조건식2
> SELECT * FROM sample WHERE a<>0 OR b<>0; // a열이 0이 아니거나 b열이 0이 아닌 행 검색

> NOT 조건식
> SELECT * FROM sample WHERE NOT(a<>0 OR b<>0); // a열이 0이 아니거나 b열이 0이 아닌 행을 제외한 나머지 행 검색
AND는 OR에 비해 우선 순위가 높다.
패턴 매칭에 의한 검색
특정 문자나 문자열이 포함되어 있는지를 검색하고 싶은 경우 '패턴 매칭'(부분 검색)을 사용
LIKE
> 열 LIKE '패턴'

// text 열에 'SQL'을 포함하는 행을 검색
> SELECT * FROM sample WHERE text LIKE 'SQL%'; // (전방일치)
> SELECT * FROM sample WHERE text LIKE '%SQL%'; // (중간일치)

// text 열에 '%'(메타문자)을 포함하는 행을 검색
> SELECT * FROM sample WHERE text LIKE '%₩%%'; // 이스케이프 문자(/) 사용

// text 열에 'It's'을 포함하는 행을 검색
> SELECT * FROM sample WHERE text LIKE 'It''s'; // '를 문자열 상수 안에 포함할 경우 2개를 연속해서 기술
사용할 수 있는 메타문자(또는 와일드카드)는 '%'와 '_'가 있다.
%(퍼센트): 임의의 문자열과 매치, 빈 물자열에도 매치한다.
_(언더스코어): 임의의 문자 하나
*는 사용할 수 없다.

■ INSERT 명령문

SELECT 명령의 경우 실행하면 그 결과가 클라이언트에게 반환되지만, INSERT 명령은 데이터가 클라이언트에서 서버로 전송되므로 반환되는 결과가 없다.

기본 사용 예시
> INSERT INTO 테이블명 VALUES(값1, 값2, ...);
> INSERT INTO sample VALUES(1, 'HEEE', '2019-04-27');

값을 저장할 열 지정
> INSERT INTO 테이블명 (열1, 열2, ...) VALUES(값1, 값2, ...);
> INSERT INTO sample (name, no) VALUES('HEEE2', 2);

■ DELETE 명령문

- DB의 테이블에서 행을 삭제
- 삭제는 행 단위로 수행되며, 열을 지정하여 해당 열만 삭제할 수는 없다.
- DELETE 명령을 실행할 때는 재확인을 위한 대화창 같은 것이 표시되지 않기 때문에 주의해야 한다.

기본 사용 예시

> DELETE FROM 테이블명 WHERE 조건식;

// sample 테이블의 모든 데이터 삭제

> DELETE FROM sample;

// sample 테이블의 no 열이 2인 행 삭제

> DELETE FROM sample where no=2;

물리삭제와 논리삭제

DB에서 데이터를 삭제하는 방법은 용도에 따라 크게 '물리삭제'와 '논리삭제'로 나뉜다.

물리삭제

SQL의 DELETE 명령을 사용해 직접 데이터를 삭제하는 것

Ex) SNS 서비스에서의 사용자 탈퇴

논리삭제

테이블에 '삭제 플래그'를 두어 행을 삭제하는 대신, SQL의 UPDATE 명령을 사용해 해당 플래그의 값을 유효하게 갱신해두는 것

장점: 삭제 되기 전의 상태로 간단히 되돌릴 수 있다.

단점: 삭제해도 DB의 저장공간이 늘어나지 않는다. 이로 인해 DB 크기가 증가하여 검색속도가 떨어진다.

Ex) 쇼핑 사이트에서의 주문 취소

■ UPDATE 명령문

- DB의 테이블에서 데이터를 갱신
- 웹 페이지에서 '등록'이나 '갱신'와 같은 버튼을 클릭했을 때 처리되는 데이터 갱신 기능
- DELETE 명령어와 달리 셀 단위로 데이터를 갱신할 수 있다.
- WHERE 구를 생략한 경우에는 테이블의 모든 행이 갱신된다.
- 테이블에 존재하지 않는 열을 지정하면 에러가 발행하여 해당 명령은 실행되지 않는다.

기본 사용 예시

> UPDATE 테이블명 SET 열1=값1, 열2=값2, ... WHERE 조건식;

> UPDATE sample SET name='HEEE' WHERE no=2;

> UPDATE sample SET name='HEEE', date='2019-04-26' WHERE no=2;

> UPDATE sample SET name=NULL; // NULL로 초기화

SET 구의 실행순서
> UPDATE sample SET no=no+1, a=no; // 1번
> UPDATE sample SET a=no, no=no+1; // 2번

MySQL에서는 기술한 식의 순서에 따라 갱신 처리가 일어난다.
1번에서의 no 열의 값은 갱신된 이후의 값을 반환하고 a 열은 갱신된 값을 참조한다.
Ex) 1번의 경우, 초기 no: 2, 3 -> 갱신 후 no: 3, 4 / a: 3, 4
2번에서의 a 열은 갱신되기 전의 값을 참조하고 이후에 no 열이 갱신된다.
Ex) 2번의 경우, 초기 no: 2, 3 -> 갱신 후 no: 3, 4 / a: 2, 3

즉, MySQL 의 경우 갱신식 안에서 열을 참조할 때는 처리 순서를 고려할 필요가 있다.

Oracle에서는 SET 구에 기술한 식의 순서가 처리에 영향을 주지 않는다.
1, 2번 모두 no 열의 값이 항상 갱신 이전의 값을 반환하기 때문에 no 열을 참조하는 a 열의 값은 갱신 전의 값이 된다.
이후에 no 열이 갱신된다.
Ex) 1, 2번 모두, 초기 no: 2, 3 -> 갱신 후 no: 3, 4 / a: 2, 3