SECS Driver (“EZNET”)

ActiveX Control

USER’S MANUAL

(Ver 4.0)

Last Update : 2017/07/19

---

www.nviasoft.co.kr
대표전화) 031-776-2250
팩스(Fax) 031-776-2252
Email) all@nviasoft.co.kr

작성일 : 2017. 10. 12
주소 : 경기 성남시 중원구 사기막골로 124 SKn테크노파크 메

내용

1. 소개 (Introduction)

ezNet SECS Driver는 사용자(User)가 개발자라는 점에 더욱더 주안을 두어 고객의 눈높이와 요구사항을 충분히 수렴, 보다 사용하기 쉽고, 재사용이 가능하며, 간단한 설정만으로 최대의 효과를 발휘할 수 있게끔 개발되었으며 오랜기간 축적된 SECS 지식과 풍부한 현장경험이 철저히 반영된 ActiveX Control입니다.

Edit History

3.3.1	2017. 07 . 19	※skiplog, separatelog 추가	skpark
4.0	2017. 10 . 12	※SetLogFile ※SaveMsgToSpool ※SendSpooledMsg	skpark

2. 특징 (Features)

. OCX형태로 되어 손쉬운 Control 추가

초기화과정이 쉬워 별도의 어려운 설정없이 기존 UI에 추가만 하면 됩니다.

. SECS-I (RS232C) & HSMS 동시 지원

간단한 설정만으로 SECS-I과 HSMS를 전환할 수 있습니다.
Run Time에서는 설정파일(Configuration File)에서 해당 Property만 변경하여 재구동 하면 바로 전환되며, 또한 Property Method를 통해 전환할 수도 있습니다.
(참고) EZNET Light 버전은 HSMS전용 드라이버입니다.

. S9Fn의 자동전송

각종 오류사항 (Undefined Device ID, Undefined Message 등)을 전송하는 S9 Series(S9F1, S9F3, S9F5, S9F7, S9F9)를 자동으로 전송해 줍니다.

. 편리한 Binary파일 처리 기능

PP(Process Program, Recipe) 관련 메세지를 처리하는 방법이 탁월하게 용이하게 되어 있습니다.
해당 파일을 지정해주면 자동으로 처리해주며, 또한 PP수신시 자동으로 파일로 저장을 해주어 개발자가 별도로 처리하지 않아도 되게끔 되어 있습니다.

. 설정파일(Configuration File)의 간단한 편집으로 사용 설정 전환

구동환경의 변화는 주어진 Method를 이용하여 필요한 Property를 바꿀 수도 있지만 설정파일을 간단하게 편집하여 사용 환경을 재설정 할 수 있습니다.

. 풍부한 Event & Log지원

Run Time에서 발생하는 각종 Event와 Log를 다양한 형태로 제공합니다.
특히 로그파일은 매일 자동으로 날짜별로 저장이 되며,
한 프로그램에서 여러개의 인스턴스(Instance)를 사용하더라도 개개의 Instance별로 별도의 Log파일을 지정하게 되어 있어 오류추적이 아주 용이하게 되어 있습니다. 또한 Log파일 내부에서도 간단한 로그만 혹은 아주 세부적인 로그까지 기록으로 남기게끔 할 수 있습니다.

. 빠른 Speed와 최소의 Resource사용

풍부한 경험과 Network구현지식으로 병목현상(Bottle Neck)을 없애 만족할 만한 Speed를 보여주고 있으며 최소의 Resource를 사용하여 Control이 아주 Compact합니다.

. 무제한의 ActiveX Control Instance사용

Instance의 제한이 없어 고객이 원하는 대로 Control을 사용할 수 있고, 각 Instance 별로 환경 설정 파일 및 로그파일을 별도로 둘 수 있습니다.
(참고) EZNET Light 버전은 Single Instance용으로 제작되었음.

. 다양한 Method지원

개발자의 편의를 제공하기 위해 상당히 많은 Method를 지원합니다.
특히 SECS MSG를 구성하거나 접근이 용이하도록 구성하였습니다.

. 풍부한 확장성

다양한 Method를 제공함에 따라 개발자는 다양한 확장성을 기반으로 여러가지 작업을 진행할 수 있습니다.

. 강력한 Format Validation기능

수신할 SECS Message를 미리 정의하여 형식에 맞지 않을 경우 자동으로 에러처리를 하며, Message를 중첩적으로 정의할 수 있어 다양한 형식을 선언할 수 있습니다.

3. Control Properties List

DeviceID

Data Type	Short
Protocol	HSMS
Default Value	0
Note	***short*** **GetDeviceID( );** ***void*** **SetDeviceID(*short*);**

SecsI

*Light Version 해당사항 없음

Data Type	Boolean
Protocol	SECS-I & HSMS
Default Value	false (=HSMS)
Note	SECS-I (RS232 Serial) Mode인지 HSMS Mode인지를 결정 TRUE : SECS-I(Serial) Mode FALSE : HSMS Mode ***BOOL*** **GetSecsI( );** ***void*** **SetSecsI(*BOOL*);**

PassiveMode

Data Type	boolean
Protocol	SECS-I & HSMS
Default Value	false (=Active Mode)
Note	for HSMS Mode ● TRUE ➔ Passive Mode ● FALSE ➔ Active Mode for SECS-I Mode ● TRUE ➔ Secondary Side ● FALSE ➔ Primary Side ***BOOL*** **GetPassiveMode( );** ***void*** **SetPassiveMode(*BOOL*);**

IP

Data Type	BSTR
Protocol	HSMS
Default Value	"127.0.0.1"
Note	HSMS Mode에서 접속 혹은 대기 할 IP주소 별도로 지정하지 않으면 "127.0.0.1"로 접속하게 됩니다. (IP 127.0.0.1은 localhost로 자신의 PC를 의미합니다) ***BSTR* GetIp( );** ***void* SetIp(*BSTR*);**

Port

Data Type	short
Protocol	HSMS
Default Value	5000
Note	HSMS Mode에서 접속 혹은 대기 할 TCP Port번호 ***short* GetPort( );** ***void* SetPort(*short*);**

CommPort

*Light Version 해당사항 없음

Data Type	short
Protocol	SECS-I
Default Value	1 (=COM1)
Note	SECS-I (Serial) Mode에서의 Serial Port 번호 1 : COM1 2 : COM2 3 : COM3 4 : COM4 ***short* GetCommPort( );** ***void* SetCommPort(*short*);**

BaudRate

*Light Version 해당사항 없음

Data Type	long
Protocol	SECS-I
Default Value	9600
Note	***long* GetBaudRate( );** ***void* SetBaudRate(*long*);**

Parity

*Light Version 해당사항 없음

Data Type	short
Protocol	SECS-I
Default Value	0
Note	패러티 검사 (Parity Check) 0 : None (Parity Check 없음) 1 : Odd (홀수비트) 2 : Even(짝수비트) ***short* GetParity( );** ***void* SetParity(*short*);**

DataBits

*Light Version 해당사항 없음

Data Type	short
Protocol	SECS-I
Default Value	8
Note	***short* GetDataBits();** ***void* SetDataBits(*short*);**

StopBits

*Light Version 해당사항 없음

Data Type	short
Protocol	SECS-I
Default Value	1
Note	***short* GetStopBits();** ***void* SetStopBits(*short*);**

T1

*Light Version 해당사항 없음

Data Type	short
Protocol	SECS-I
Default Value	1 (sec)
Note	SECS-I(Serial) mode에서 Data전송시 Char와 Char간의 Time Out을 의미합니다. (“Character TimeOut”) T1이 발생하면 ezNet Driver는 자동으로 NAK을 전송하여 수신중인 Data를 무효화 시킵니다. ***short* GetT1();** ***void* SetT1(*short*);**

T2

*Light Version 해당사항 없음

Data Type	short
Protocol	SECS-I
Default Value	10 (sec)
Note	SECS-I(Serial) mode에서 Data 전송 시 상대측에 전송요청을 하게 됩니다. 상대측은 전송수락(ACK) 또는 거부 (NAK)로 응답을 하게 되는데, ENQ에 대한 응답시간을 의미합니다. (“Block Protocol TimeOut”) ***short* GetT2();** ***void* SetT2(*short*);**

T3

Data Type	short
Protocol	SECS-I & HSMS
Default Value	30 (sec)
Note	SECS-II Message중에 W(Wait) Bit가 1 이면 그 Message를 수신하는 측은 반드시 응답메시지(Reply Message)로 답해야 하는데, 그 응답을 하기 까지 허용된 시간입니다. (“Message Reply TimeOut”) ***short* GetT3();** ***void* SetT3(*short*);**

T4

*Light Version 해당사항 없음

Data Type	short
Protocol	SECS-I
Default Value	10 (sec)
Note	Message가 Multi-Block인 경우 Block과 다음 Block간의 허용 시간 입니다. T4 Timeout이 발생하면 ezNet Driver는 자동으로 NAK을 전송하며 수신된 Message는 무효화 시킵니다. ***short* GetT4();** ***void* SetT4(*short*);**

T5

Data Type	short
Protocol	HSMS
Default Value	5 (sec)
Note	연결실패(Connection Failure)나 끊김(Disconnection) Event가 발생한 경우 재접속(Reconnection)을 시도하기에 앞서 대기할 시간입니다. ("Separate TimeOut") ***short* GetT5();** ***void* SetT5(*short*);**

T6

Data Type	short
Protocol	HSMS
Default Value	20 (sec)
Note	HSMS의 Control 에 대한 허용된 응답시간입니다.("Control TimeOut") HSMS Control은 SELECT.REQ 혹은 LINKTEST.REQ등이 있습니다. 즉, Active측에서 SELECT.REQ를 송신했다면, Passive측은 SELECT.RES를 보내어 State가 SELECTED로 되었음을 응답해야 합니다. 또한 LinkTest.Req를 받았다면 응답인 LinkTest.Res를 보내야 합니다. 설정된 T6를 초과하면 시간초과(TimeOut) Event가 발생하게 됩니다. T6 Event가 발생하면 ezNet Driver는 Connection을 끊고 대기시간(T5)후에 재접속 과정에 들어갑니다. ***short* GetT6();** ***void* SetT6(*short*);**

T7

Data Type	short
Protocol	HSMS
Default Value	5 (sec)
Note	HSMS mode는 TCP Socket Connection되었다고 완전히 연결된 상태는 아닙니다. 소켓연결 (Socket Connection)후에 State 변화가 있어야 비로소 연결이 되는 것입니다. T7는 TCP Socket Connection후에 SELECTED상태로 되기까지의 허용된 시간입니다. (“Not Selected TimeOut”) T7 Event가 발생하면 ezNet Driver는 Socket Connection을 끊고 재접속을 시도하게 됩니다. ***short* GetT7();** ***void* SetT7(*short*);**

T8

Data Type	short
Protocol	HSMS
Default Value	6 (sec)
Note	SECS-I의 T1과 대응되는 것으로 Char간의 전송허용시간입니다. ***short* GetT8();** ***void* SetT8(*short*);**

Retry

Data Type	short
Protocol	SECS-I & HSMS
Default Value	3 (times)
Note	송신된 Message에 대한 응답 Message가 없는 경우 자동으로 재 전송하게 되는데, 재 전송을 몇번 까지 시도할 지 지정하는 값입니다. Retry Count가 Limit에 도달하면 ezNet Driver는 S9F9를 자동으로 전송하여 그 응답 Message를 무효화 시킵니다. ***short* GetRetry();** ***void* SetRetry(*short*);**

LinkTestInterval

Data Type	short
Protocol	HSMS
Default Value	30 (sec)
Note	연결된 상태에서 송수신이 없을 때 주기적으로 연결확인을 위한 LinkTest.Req와 LinkTest.Res가 오고 가는데, 그 주기를 말합니다. 이 값을 제로(0)로 설정하면 LinkTest를 수행하지 않습니다. ***short* GetLinkTestInterval();** ***void* SetLinkTestInterval(*short*);**

HostMode

Data Type	Boolean
Protocol	SECS-I & HSMS
Default Value	TRUE
Note	Host인지 Equipment를 결정하는 Value TRUE : Host FALSE : Equipment ***BOOL* GetHostMode();** ***void* SetHostMode(*BOOL*);**

4. Method (Function)

1)초기 설정관련 **Methods**

void SetConfigFile(LPCTSTR strConfigFile)	
Function	환경파일(Configuration File)을 등록하고 읽어들이는 기능입니다. (Note) ➔ 여러 개의 인스턴스(Instance) 사용하는 경우가 많기 때문에 환경파일을 등록하는 것을 권장합니다. ➔ 환경파일을 등록하지 않으면, 초기값이 그대로 사용되며, Properties List에서 언급된 Method(Setter-Getter)를 통해 개별적으로 변경할 수 있습니다. ➔ 환경파일 편집은 사용자가 직접하거나, 환경설정도우미 즉 Configurator를 이용 하셔도 됩니다. ➔ 환경파일을 등록했는데, 그 파일이 존재하지 않거나 경로(Path)가 틀린경우 RunTime에서 곧바로 MessageBox로 Error를 알려줍니다.
Protocol	SECS-I & HSMS
Parameters	LPCTSTR strConfigFile (각종 설정값이 담긴 환경 파일경로 및 파일이름. 절대경로/상대경로 무관)
Return Value	NONE
Example	m_secs.**SetConfigFile**("C:\\temp\\Test1.cfg"); short nResult = m_secs.Start();

void SetComm(short nComPort, long lBaudRate, short nParityBits, short nDataBits, short nStopBits)	
Function	환경파일(Configuration File)을 사용하지 않고 직접적으로 SECS-I 설정을 할 때 사용합니다. 아래의 개별 설정 Method를 이용하는 것과 동일한 기능이지만, 한꺼번에 처리를 할 수 있습니다. void SetComPort(short) void SetBaudRate(long) void SetParityBits(short) void SetDataBits(short); void SetStopBits(short) *Light Version 해당사항 없음
Protocol	SECS-I
Parameters	short nComPort : COM Port번호 (1,2,3,4) COM1➔1 , COM2➔2 , COM3➔3 , COM4➔4 초기값 : 1 long lBaudRate : 전송속도 (2400,4800, 9600, 19600, …) 초기값 : 9600 short nParityBits : 패리티체크 (0, 1, 2) NONE➔0 , ODD➔1 , EVEN➔2 초기값 : 0 short nDataBits : 데이터비트 초기값 : 8 short nStopBits : 스톱비트 초기값 : 1
Return Value	= 0 ➔ 성공 < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.
Example	m_secs.SetSecsI(TRUE); // 이것을 지정하지 않으면 HSMS로 작동됨 m_secs.**SetComm**(1,9600,0,8,1); short nResult = m_secs.Start();

void EnableLog()　/　void DisableLog()	
Function	로그를 적용하거나 생략합니다. 환경파일(Configuration File)의 LOG항목과 동일하기 때문에, 환경파일을 적용하면 이 Method는 불필요합니다. 초기값은 LOG=TRUE이기 때문에 로그를 적용하게끔 되어 있습니다.
Protocol	SECS-I & HSMS
Parameters	NONE
Return Value	NONE
Example	m_secs.**DisableLog**(); m_secs.**EnableLog**();

void SetLogFile(LPCTSTR strFileName, short bLogSecsII)	
Function	환경파일을 사용하지 않고 직접적으로 로그파일을 설정할 때 사용합니다. 로그파일을 설정하지 않으면 default log파일(ezNet.log)에 기록됩니다. 만약 여러 개의 인스턴스(Instance)를 사용하는 경우 로그파일을 지정하지 않으면 한 개의 로그파일에 모든 인스턴스 로그를 남기게 됩니다. 그러므로 가능한한 로그파일을 지정하는 것을 권장합니다. ~~참고로, 로그는 날짜별로 Backup됩니다.~~
Protocol	SECS-I & HSMS
Parameters	LPCTSTR strFileName : ➔ 로그파일 경로 및 파일명 (파일이 존재:이어서 작성/없으면 생성합니다) ※ 파일이름에 "_(언더 바)"를 사용하지 않아야 합니다. short bLogSecsII ➔ SECS-II 메시지 내용도 로그로 남길것인지 결정합니다. TRUE➔SECS-II Message도 Log에 남김 FALSE➔Event, Alarm, Socket State만 남기고 SECS-II는 남기지 않습니다.
Return Value	NONE
Example	m_secs.**SetLogFile**("LOG\\GEMLOG.log", TRUE); 일 경우 LOG\\20171012\\ GEMLOG_20171012_[0001].LOG로 생성 20171012는 로그 생성일(시스템의 today)

void SetLogRetention(short nDays)	
Function	날짜별로 백업된 로그파일을 몇일간 보유할 것인지를 설정합니다.
Protocol	SECS-I & HSMS
Parameters	Short nDays : ➔ 로그파일 보유일

void SkipLog(short nStream , short nFunction)	
Function	설정된 Sx,Fy에 대하여 로그에 남기지 않음.
Protocol	SECS-I & HSMS
Parameters	Short nStream , Short nFunction ➔ 로그에 남기지 않을 메시지의 Stream과 Function번호

void SeparateLog(LPCTSTR strFileName,short nStream,short nFunction)	
Function	설정된 Sx,Fy에 대하여 해당 파일에 로그를 남김.
Protocol	SECS-I & HSMS
Parameters	Short nStream , Short nFunction , LPCTSTR strFileName ➔ strFileName 파일에 특정 메시지(SxFy)를 남김.

short GetLogRetention()	
Function	로그파일을 보유기한 값을 읽어옵니다.
Protocol	SECS-I & HSMS
Return Value	로그파일 보유일

void SetFormatCheck(short bEnabled)	
Function	미리정의된 포맷파일 (SML) 상의 Stream/Function 구조(Structure) 확인 기능을 사용할지 사용하지 않을지를 선택합니다. (Format Validation)
Protocol	SECS-I & HSMS
Parameters	short bEnabled : 0 or FALSE (사용하지 않음) 1 or TRUE (사용함) Default 값으로 OFF되어 있습니다.

short GetFormatCheck()	
Function	Format 검증(Validation)기능이 활성화되었는지 확인합니다.
Protocol	SECS-I & HSMS
Return Value	0 : Format Validation OFF, 1 : Format Validation ON

void SetFormatFile(BSTR strSMLFile)	
Function	SML Format File을 지정합니다.
Protocol	SECS-I & HSMS
Parameters	BSTR strSMLFile Format (Structure) 이 정의된 파일입니다. 제공되는 편집 프로그램을 이용하시거나, 직접 작성이 가능합니다.

2) ezNet SECS driver의 구동 및 정지

short Start()	
Function	ezNet SECS driver를 구동합니다. ➔ HSMS Protocol 인 경우 - Active mode인 경우 TCP Socket(소켓) 연결 및 SELECT.REQ를 보내어 연결을 요청합니다. - Passive mode인 경우 연결을 기다립니다. ➔ SECS-I Protocol 인 경우 - Serial COM port를 열어 연결합니다.
Parameters	NONE
Return Value	= 0 ➔ 성공 < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.
Example	m_secs.SetConfigFile("C:\\temp\\TEST1.CFG"); short nResult = m_secs.**Start**();

short Stop()	
Function	ezNet SECS driver 작동을 중단합니다.
Parameters	NONE
Return Value	= 0 ➔ 성공 < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.
Example	short nResult = m_secs.**Stop**();

3) SECS Message의 작성 및 전송 관련 Method

일반적인 SECS Message 전송 절차

➔ Message Structure 생성 및 Message ID를 부여 받습니다.

Stream/Function/WBit 는 User가 지정합니다.
그외 나머지는 자동으로 설정하고, 시스템바이트(System Byte)는 자동으로 부여됩니다.

➔ 부여받은 MessageID를 이용하여 Message를 구성합니다. (각종 Item지정)

➔ Message를 전송합니다.

long CreateMsg(short nStream, short nFunction, short bWaitReply)	
Function	SECS Message Object를 생성하고 해당 MessageID를 반환합니다.
Parameters	.nStream ➔ 스트림(Stream)번호 .nFunction ➔ 펑션(Function)번호 .bWaitReply ➔ 응답요구여부
Return Value	> 0 ➔ 성공 (MessageID를 반환) <= 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.
Example	long lMsgId = m_secs.**CreateMsg**(1,1,TRUE); // S1F1W

short SendMsg(long lMsgId)	
Function	Message를 전송합니다.
Parameters	long lMsgId ➔ 전송하고자 하는 Message ID
Return Value	= 0 ➔ 성공 < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.
Example	long lMsgId = m_secs.CreateMsg(1,1,TRUE); // S1F1W m_secs.**SendMsg**(lMsgId);

short OpenListItem(long lMsgId) / CloseListItem(long lMsgId) short AddListItem(long lMsgId, long lSize)	
Function	리스트아이템(List Item)을 열고 닫습니다. = 리스트아이템(List Item)을 추가합니다 (동일한 개념) Open/Close 두개의 Method가 반드시 한 쌍이 되어야 합니다. 즉, 열어놓은 후 다른 아이템(또 다른 리스트포함) 추가 후 반드시 닫아주어 해당 리스트아이템의 마지막임을 알려주어야 합니다. ● 통신사양서를 보면서 프로그램을 작성할 때 사양서 표현방법에 따라 선택적으로 사용할 수 있습니다. (List Item내의 Subitem개수 = Open-Close함수 사이의 개수) **<L** → **OpenListItem**(msgid) <A MLDN> <A SOFTREV> **>** → **CloseListItem**(msgid) 만약 사양서 표현이 아래와 같다면 L, 2 <A MDLN> <A SOFTREV> 이때는 **AddListItem**(msgid, **2**) 함수가 유용합니다. 이 경우 드라이버에게 SubItem이 두개가 추가될 것을 미리 알려주는 역할을 합니다.
Parameters	작성중인 Message ID
Return Value	= 0 ➔ 성공 < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.

Example	➔ 리스트아이템이 한 개이면서 다른 sub item이 없는 경우 // S1F5 // <L // > long lMsgId = m_secs.CreateMsg(1,5,FALSE); m_secs.**OpenListItem**(lMsgId); // List Item을 추가(Open)합니다. m_secs.**CloseListItem**(lMsgId); // List Item을 닫습니다. m_secs.SendMsg(lMsgId); 이경우 아래와 같이 표현할 수도 있습니다. long lMsgId = m_secs.CreateMsg(1,5,FALSE); m_secs.**AddListItem**(lMsgId, **0**); m_secs.SendMsg(lMsgId); ➔ 리스트아이템 내에 다른 리스트아이템이 있는 경우(중첩된경우) long lMsgId = m_secs.CreateMsg(64,4,FALSE); // S64F4 m_secs.**OpenListItem**(lMsgId); // <L m_secs.**OpenListItem**(lMsgId); // <L m_secs.**OpenListItem**(lMsgId); // <L m_secs.**CloseListItem**(lMsgId); // > m_secs.**CloseListItem**(lMsgId); // > m_secs.**CloseListItem**(lMsgId); // > m_secs.SendMsg(lMsgId); 이경우 아래와 같이 표현할 수도 있습니다. long lMsgId = m_secs.CreateMsg(1,5,FALSE); m_secs.**AddListItem**(lMsgId, **1**); m_secs.**AddListItem**(lMsgId, **1**); m_secs.**AddListItem**(lMsgId, **0**); m_secs.SendMsg(lMsgId);

short AddBinaryItem(long lMsgId, short *pnValue, long lCount)	
Function	바이너리 아이템 (Binary Item)을 추가합니다.
Parameters	. long lMsgId ➔ 작성중인 MessageID . short *pnValue ➔ 추가할 Item의 포인터변수 . long lCount ➔ Item의 크기 또는 개수
Return Value	= 0 ➔ 성공 < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.
Example	// 아이템이 한 개로 구성된 경우 (Non Array Item) // S1F9 // <L // <B 0x01> // <B 0xFF> // > short binValueA = 0x01; short binValueB = 0xFF; long lMsgId = m_secs.CreateMsg(1,9,TRUE); m_secs.OpenListItem(lMsgId); m_secs.**AddBinaryItem**(lMsgId, &binValueA, 1); m_secs.**AddBinaryItem**(lMsgId, &binValueB, 1); m_secs.CloseListItem(lMsgId); m_secs.SendMsg(lMsgId); // 아이템이 배열로 구성된 경우 (Array Item) // S2F3W // <B 0x01 0x02 0x03 0x04 0xFF> short binValue[] = { 0x01, 0x02, 0x03, 0x04, 0xFF }; long lMsgId = m_secs.CreateMsg(2,3,TRUE); m_secs.**AddBinaryItem**(lMsgId, binValue, 5); m_secs.SendMsg(lMsgId);

short AddFileBinaryItem(long lMsgId, LPCTSTR pszFile)	
Function	파일을 읽어 그것을 바이너리 아이템 (Binary Item)으로 추가합니다. 주로, S7Fn (Process Program 관련) 사용시 편리합니다.
Parameters	. long lMsgId ➔ 작성중인 MessageID . LPCTSTR pszFile ➔ Binary Item으로 변환할 파일경로 및 이름
Return Value	= 0 ➔ 성공 < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.
Example	long lMsgId = m_secs.CreateMsg(7, 3, TRUE); m_secs.OpenListItem(lMsgId); m_secs.**AddFileBinaryItem**(lMsgId, "C:\\temp\\PPSample"); m_secs.CloseListItem(lMsgId); m_secs.SendMsg(lMsgId);

short AddAsciiItem(long lMsgId, LPCTSTR pszValue, long lCount)	
Function	문자열(Ascii String) 형태의 Item을 추가합니다.
Parameters	. long lMsgId ➔ 작성중인 MessageID . LPCTSTR pszValue ➔ 추가할 Item . long lCount ➔ Item의 개수
Return Value	= 0 ➔ 성공 < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.
Example	// S7F6 // <L // <A "PPSample"> // <B 01 00 FF AA A0 …> // > CString strPPName = "PPSample"; long lMsgId = m_secs.CreateMsg(7, 6, FALSE); m_secs.OpenListItem(lMsgId); m_secs.**AddAsciiItem**(lMsgId, strPPName, strPPName.GetLength()); m_secs.AddFileBinaryItem(lMsgId, "C:\\temp\\PPSample"); m_secs.CloseListItem(lMsgId); m_secs.SendMsg(lMsgId);

short AddBoolItem(long lMsgId, short *pnValue, long lCount)	
Function	BOOLEAN형의 Item을 추가합니다.
Parameters	. long lMsgId ➔ 작성중인 MessageID . short *pnValue ➔ 추가할 Item의 포인터변수 . long lCount ➔ Item의 개수
Return Value	= 0 ➔ 성공 < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.
Example	// Item이 한 개인 경우 (Non Array Item) // S15F19W // <L // <BOOLEAN 0x01> // > short boolValue = 0x01; long lMsgId = m_secs.CreateMsg(15, 19, TRUE); m_secs.OpenListItem(lMsgId); m_secs.**AddBoolItem**(lMsgId, &boolValue, 1); m_secs.CloseListItem(lMsgId); m_secs.SendMsg(lMsgId); // Item이 배열로 구성된 경우 (Array Item) // S15F19W // <L // <BOOLEAN 0x01 0x00 0x00 0x01 0x00> // > short boolValue[5] = { 0x01, 0x00, 0x00, 0x01, 0x00 }; long lMsgId = m_secs.CreateMsg(15, 19, TRUE); m_secs.OpenListItem(lMsgId); m_secs.**AddBoolItem**(lMsgId, boolValue, 5); m_secs.CloseListItem(lMsgId); m_secs.SendMsg(lMsgId);

short AddI1Item(long lMsgId, short *pnValue, long lCount)	
Function	I1(1 Byte Signed Integer) Item을 추가합니다.
Parameters	. long lMsgId ➔ 작성중인 MessageID . short *pnValue ➔ 추가할 Item의 포인터변수 . long lCount ➔ Item의 개수
Return Value	= 0 ➔ 성공 < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.
Example	// Item이 한 개인 경우 (Non Array Item) // S1F3W // <L // <I1 10> // > short nValue=10; long lMsgId = m_secs.CreateMsg(1,3,TRUE); m_secs.OpenListItem(lMsgId); m_secs.**AddI1Item**(lMsgId, &nValue, 1); m_secs.SendMsg(lMsgId); // Item이 배열로 된 경우 (Array Item) // S1F3W // <L // <I1 2 4 6 8 10> // > short nValue[5] = { 2, 4, 6, 8, 10 }; long lMsgId = m_secs.CreateMsg(1,3,TRUE); m_secs.OpenListItem(lMsgId); m_secs.**AddI1Item**(lMsgId, nValue, 5); m_secs.SendMsg(lMsgId);

short AddI2Item(long lMsgId, short *pnValue, long lCount)	
Function	I2(2 Byte Signed Integer) Item을 추가합니다.
Parameters	. long lMsgId ➔ 작성중인 MessageID . short *pnValue ➔ 추가할 Item의 포인터변수 . long lCount ➔ Item의 개수
Return Value	= 0 ➔ 성공 < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.

short AddI4Item(long lMsgId, long *plValue, long lCount)	
Function	I4(4 Byte Signed Integer) Item을 추가합니다.
Parameters	. long lMsgId ➔ 작성중인 MessageID . long *plValue ➔ 추가할 Item의 포인터변수 . long lCount ➔ Item의 개수
Return Value	= 0 ➔ 성공 < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.

short AddI8Item(long lMsgId, long *plValue, long lCount)	
Function	I8(8 Byte Signed Integer) Item을 추가합니다.
Parameters	. long lMsgId ➔ 작성중인 MessageID . long *plValue ➔ 추가할 Item의 포인터변수 . long lCount ➔ Item의 개수
Return Value	= 0 ➔ 성공 < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.

short AddU1Item(long lMsgId, short *pnValue, long lCount)	
Function	U1 (1 Byte Unsigned Integer) Item을 추가합니다.
Parameters	. lMsgId ➔ 작성중인 MessageID . *pnValue ➔ 추가할 Item의 포인터변수 . lCount ➔ Item의 개수
Return Value	= 0 ➔ 성공 < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.

short AddU2Item(long lMsgId, long *plValue, long lCount)	
Function	U2 (2 Byte Unsigned Integer) Item을 추가합니다.
Parameters	. long lMsgId ➔ 작성중인 MessageID . long *plValue ➔ 추가할 Item의 포인터변수 . long lCount ➔ Item의 개수
Return Value	= 0 ➔ 성공 < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.

short AddU4Item(long lMsgId, double *prValue, long lCount)	
Function	U4 (4 Byte Unsigned Integer) Item을 추가합니다.
Parameters	. long lMsgId ➔ 작성중인 MessageID . double *prValue ➔ 추가할 Item의 포인터변수 . long lCount ➔ Item의 개수
Return Value	= 0 ➔ 성공 < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.

short AddU8Item(long lMsgId, double *prValue, long lCount)	
Function	U8 (8 Byte Unsigned Integer) Item을 추가합니다.
Parameters	. long lMsgId ➔ 작성중인 MessageID . double *prValue ➔ 추가할 Item의 포인터변수 . long lCount ➔ Item의 개수
Return Value	= 0 ➔ 성공 < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.

short AddF4Item(long lMsgId, float *pfValue, long lCount)	
Function	F4 (4 Byte Floating Pointer) Item을 추가합니다.
Parameters	. long lMsgId ➔ 작성중인 MessageID . float *pfValue ➔ 추가할 Item의 포인터변수 . long lCount ➔ Item의 개수

Return Value	= 0 ➔ 성공 < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.

short AddF8Item(long lMsgId, double *prValue, long lCount)	
Function	F8 (8 Byte Floating Pointer) Item을 추가합니다.
Parameters	. long lMsgId ➔ 작성중인 MessageID . double *prValue ➔ 추가할 Item의 포인터변수 . long lCount ➔ Item의 개수
Return Value	= 0 ➔ 성공 < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.

short AddJ8Item(long lMsgId, LPCTSTR pszValue, long lCount)	
Function	JIS-8(Japanese Industrial Standard-8 Byte) Item을 추가합니다.
Parameters	. long lMsgId ➔ 작성중인 MessageID . LPCTSTR pszValue ➔ 추가할 Item . long lCount ➔ Item의 크기
Return Value	= 0 ➔ 성공 < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.
Example	// <S65F1W> // <J8 "012ABC"> CString strJIS = "012ABC"; long lMsgId = m_secs.CreateMsg(65,1,1); m_secs.AddJ8Item(lMsgId, strJIS, strJIS.GetLength()); m_secs.SendMsg(lMsgId);

void SetSystemByteBase(double nSystemByteBase)	
Function	송신 Message의 System Byte 시작 값을 강제로 설정
Parameters	. double nSytemByteBase : SystemByte 시작값 (본 드라이버는 홀수값만 적용함, 짝수지정시 자동으로 +1 처리함)

long AddArrayItem(long lMsgId, LPCTSTR pszType, long lCount, LPCTSTR pszData)	
Function	포인터를 사용할수 없는 개발언어(ex C#)에서 배열아이템을 추가하기위한 별도의 특별한 함수입니다. ● Item값을 읽어오는 함수는 GetArrayItem입니다.
Parameters	. long lMsgId ➔ 작성중인 MessageID . LPCTSTR pszType ➔ 추가할 Item의 포맷코드(타입) ex) "U1", "U2", "U4", "U8", "I1", "I2", "I4", "I8", "B", "F4","F8" . long lCount ➔ Item 배열의 개수 . LPCTSTR pszData ➔ 추가할 ITEM값 (구분자-delimeter) Ex) "1,2,3,4,5" (구분자는 콤마를 이용합니다)
Return Value	> 0 ➔ 성공 (추가된 배열 item count) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.
Example	// <S2F71> // <U2 1 2 3 4 5> long lMsgId = m_secs.CreateMsg(2, 71, FALSE); m_secs.AddArrayItem(lMsgId, "U2", "1,2,3,4,5"); m_secs.SendMsg(lMsgId);

long SaveMsgToSpool(long lMsgId, LPCTSTR szFileName)	
Function	. 생성한 Messasge를 szFileName의 텍스트 파일로 저장
Parameters	. long lMsgId ➔ 작성중인 MessageID . LPCTSTR szFileName ➔ 생성한 lMsgID를 szFileName의 바이너리 파일로 생성하는 파일명
Return Value	> 0 ➔ 성공 (파일로 생성) < 0 ➔ 실패 (파일 생성 실패 or lMsgId가 존재하지 않음)
Example	long lMsgId = m_secs.CreateMsg(2, 71, FALSE); m_secs.AddArrayItem(lMsgId, "U2", "1,2,3,4,5"); if(nControlState = CONTROL_OFFLINE) m_secs.SendMsg(lMsgId); else m_secs.SaveMsgToSpool(lMsgId, "20171012_110601_LOT.spl");

long SendSpooledMsg(LPCTSTR szFileName)	
Function	. SaveMsgToSpool로 생성한 파일을 Message로 HOST에 전송
Parameters	. LPCTSTR szFileName ➔ SaveMsgToSpool생성한 파일명
Return Value	> 0 ➔ 성공 (생성한 파일을 메시지로 전송) < 0 ➔ 실패 (저장된 파일이 없거나 현재 전송할 수 없는 경우)
Example	long lMsgId = m_secs.CreateMsg(2, 71, FALSE); m_secs.AddArrayItem(lMsgId, "U2", "1,2,3,4,5"); if(nControlState = CONTROL_OFFLINE) m_secs.SendMsg(lMsgId); else m_secs.SaveMsgToSpool(lMsgId, "20171012_110601_LOT.spl"); m_secs.SendSpooledMsg("20171012_110601_LOT.spl");

4) 사용자가 직접 처리하는 SECS Message 전송 절차

➔ Empty Message Structure 생성 및 Message ID를 부여 받습니다.

CreateEmptyMsg() 함수를 이용하여 Message ID를 부여받습니다.

이때 모든 Header Byte는 비어 있습니다.

➔ 부여받은 MessageID를 이용하여 Message를 구성합니다. (각종 Item지정)

SetHeaderByte(lMsgId, short *pnBuff); // 헤더부분을 지정합니다.

SetMsgByte(lMsgId, short *pnBuff, long lSize); // Message부분을 지정합니다.

➔ Message를 전송합니다.

SendMsg(lMsgId);

Example)

S1F3W

<L

<U2 10>

>

```
UCHAR szHeader[10], szMsg[6];
//// 메시지 직접 구성
memset(szHeader, '\0', 10);
memset(szMsg,'\0',6);
szHeader[2] = (UCHAR)0x81;  // Stream
szHeader[3] = (UCHAR)0x03;  // Function
szHeader[9] = (UCHAR)0x12;  // System Byte
szMsg[0] = (UCHAR) 0x01;
szMsg[1] = (UCHAR) 0x01;
szMsg[2] = (UCHAR) 0xA9;
szMsg[3] = (UCHAR) 0x02;
szMsg[4] = (UCHAR) 0x00;
szMsg[5] = (UCHAR) 0x0A;

/// Message ID를 할당 받습니다.
long lMsgId = m_secs.CreateEmptyMsg();
```

```
m_secs.SetHeaderByte(lMsgId, (short *) szHeader);  // 헤더 지정
m_secs.SetMsgByte(lMsgId, (short *) szMsg, 6); // SECS message 지정
m_secs.SendMsg(lMsgId);
```

* 주의사항

헤더와 메시지 구성을 직접할 경우 시스템바이트(System Byte)중복이 생길수 있습니다. 현재 송수신과정(Open Transaction)에 있거나 Message Queue에서 제거되지 않은 메시지와 동일한 시스템바이트값을 지정하는 경우 에러가 발생할 수 있습니다.

시스템바이트 중복을 피하기위해서는 수신한 메시지를 사용후 **CloseMsg(lMsgId)**를 통해 Message Queue에서 곧바로 제거하시면 혹시나 생길수 있는 시스템바이트의 중복을 피할 수 있습니다.

long CreateEmptyMsg()	*Light Version 해당사항 없음
Function	. 사용자가 직접 구성할 메시지 ID를 생성, 할당 받습니다. 일반적인 SECS Message를 생성할 때 사용하는 함수인 CreateMsg(Stream,Function,Reply)와 달리 CreateEmptyMsg()함수는 단지 Message ID만 생성하고 Stream, Function, Sytem byte등 모든값이 0으로 비어있습니다.
Return Value	> 0 ➔ 성공 (생성된 Message ID) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.

short SetHeaderByte(long lMsgId, short *pnBuff)	
Function	. 직접 구성한 헤더바이트를 지정합니다. * Light Version 해당사항 없음

Parameters	. long lMsgId ➔ 작성중인 MessageID . short *pnBuff ➔ 지정할 Message Header header byte구성시 unsigned char의 배열(혹은 포인터)로 구성하여 함수 호출시 (short *) 형으로 캐스팅을 해야합니다.
Return Value	>= 0 ➔ 성공 (지정된 Header Byte의 길이 = 10 bytes) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.

short GetHeaderByte(long lMsgId, short *pnBuff)	
Function	. (수신 혹은 작성중인 메시지의) 헤더바이트를 읽어옵니다. *Light Version 해당사항 없음
Parameters	. long lMsgId ➔ MessageID . short *pnBuff ➔ 읽어올 메모리 unsigned char의 배열(혹은 포인터)를 할당하여 함수 호출시 (short *) 형으로 캐스팅을 해야합니다.
Return Value	>= 0 ➔ 성공 (읽어온 Header Byte의 길이 = 10 bytes) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.

long SetMsgByte(long lMsgId, short *pnBuff, long lSize)	
Function	. 직접 구성한 SECS Message Byte를 지정합니다. *Light Version 해당사항 없음
Parameters	. long lMsgId ➔ 작성중인 MessageID . short *pnBuff ➔ 지정할 Message Buffer Message byte구성시 unsigned char의 배열(혹은 포인터)로 구성하여 함수 호출시 (short *) 형으로 캐스팅을 해야합니다. . long lSize ➔ 지정할 Message Buffer의 길이
Return Value	>= 0 ➔ 성공 (지정된 Message의 길이) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.

long GetMsgByte(long lMsgId, short *pnBuff, long lSize)	
Function	. SECS Message Byte를 읽어옵니다. *Light Version 해당사항 없음

Parameters	. long lMsgId ➔ 읽어올 Message ID . short *pnBuff ➔ 읽어올 Message를 저장할 Message Buffer unsigned char의 배열(혹은 포인터)을 할당하여 함수 호출시 (short *) 형으로 캐스팅을 해야합니다. . long lSize ➔ 읽어올 Message Buffer의 길이
Return Value	>= 0 ➔ 성공 (읽은 Message 의 길이) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.

5) 주요 **Event**함수

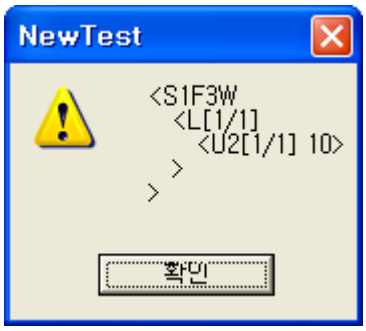
void OnConnected(LPCTSTR strHost, short nPort)
Equipment측(Server) 에 연결이 되는 순간 Event가 발생합니다.

void OnDisconnected(LPCTSTR strHost, short nPort)
연결이 끊어진 경우 Event가 발생합니다. Disgraceful Disconnection인 경우 연결을 재시도하게 됩니다.

void OnMsgReceived(long lMsgId)
SECS Message 수신시 발생하는 Event함수 입니다. Parameter인 lMsgId를 이용하여 해당 Message에 접근하게 됩니다.

void OnOtherEvent(short nEventId, long lParam)
위의 3가지 Major Event를 제외한 나머지 이벤트를 알려주는 Event함수입니다. 해당 이벤트함수 및 파라미터는 마지막부분의 Appendix II를 참고 바랍니다.

5.SECS Message의 수신 및 응답

BSTR GetMsgText(long lMsgId)	*Light Version 해당사항 없음
Function	. 수신 혹은 작성중인 Message를 SML format으로 반환합니다. 별도의 로그를 구성하거나 GUI에서 표현하고자 할 때 유용합니다. 상황에 따라서 적절하게 조치가 필요할 수도 있습니다. CString strText = m_secs.GetMsgText(lMsgId); AfxMessageBox(strText); 읽어들인 Message Text는 아래와 같은 형식으로 표현됩니다.  **<S1F3W** **<L[1/1]** **<U2[1/1] 10>** **>** **>** 한 ITEM의 길이(개수)가 255를 초과할 경우 그 이후 부분은 "…"으로 표현됩니다.
Parameters	. long lMsgId ➔ 읽고자하는 Message ID
Return Value	. Message Text

short GetMsgInfo(long lMsgId, short *pnStream, short *pnFunction, short *pnWbit, long *plLength)	
Function	MessageID를 이용하여 해당 Message정보를 가져옵니다. OnMsgReceived()가 보내주는 Message ID를 이용하여 수신한 Message처리할 때 (일반적으로) 가장 먼저 수행하여야 할 함수 입니다. 해당 Message의 Stream 및 Function 번호를 알고 그 값에 따라 각각의 Action을 정립합니다.
Parameters	. long lMsgId➔읽어올 Message의 Message ID . short *pnStream➔해당 Message의 Stream . short *pnFunction➔해당 Message의 Function . short *pnWbit➔해당 Message의 Wbit여부 (응답을 해야하는지여부) . long *plLength➔해당 Message의 크기 (Message Length)
Return Value	= 0 ➔ 성공 < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.
Example	short nStream, nFunction, nWait; long lLen; m_secs.**GetMsgInfo**(lMsgId, &nStream, &nFunction, &nWait, &lLen); if (nStream == 1 && nFunction == 1 && nWait==TRUE) { …. } else if (….

long CreateReplyMsg(long lMsgId)	
Function	수신된 Message ID를 이용하여 응답Message를 생성합니다. 이것을 수행하면 Stream 번호는 그대로, Function 번호는 1이 증가되어 짝수로 반들며, 수신한 메시지와 동일한 시스템바이트를 구성, 기타 각종 Header Byte를 자동으로 맞추어주는 편리한 함수입니다.
Parameters	. long lMsgId➔수신된 Message의 ID
Return Value	> 0 ➔ 성공 (응답 Message의 MessageID) <= 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.
Example	void CTestDlg::OnMsgReceived(LPCTSTR strHost, short nPort, long lMsgId) { short nStream, nFunction, nWait; long lLen; m_secs.GetMsgInfo(lMsgId, &nStream, &nFunction, &nWait, &lLen); if (nStream==1 && nFunction == 1 && nWait) { long lReplyMsgId = m_secs.**CreateReplyMsg**(lMsgId); //S1F2 m_secs.OpenListItem(lReplyMsgId); m_secs.CloseListItem(lReplyMsgId); m_secs.SendMsg(lReplyMsgId); } }

short CloseMsg(long lMsgId)	
Function	수신된 Message를 메시지큐(Message Que)에서 제거합니다. 모든 송/수신 메시지는 Message Queue에 등록이 됩니다. 사용자가 제거하지 않아도 SECS driver가 주기적(버전에 따라 10-20분)으로 제거를 해주는 작업을 수행하기는 하지만, 수신한후 응답하지 않는 메시지나 응답메세지등은 곧바로 삭제해 주는 것이 드라이버를 안정적으로 구동되게 합니다. 특히 Header Byte및 Message Byte를 직접 구성하는 경우 CloseMsg()를 적절히 하지 않으면, 시스템 바이트가 중복이 되어 원하는 않는 결과를 초래할 수도 있습니다. 수신된 Message에서 필요한 항목을 가져온후 더 이상 필요치 않거나 응답하지 않는 경우에는 곧바로 이 함수를 호출하여 제거해주시기 바랍니다.
Parameters	. long lMsgId➔수신된 Message의 ID
Return Value	= 0 ➔ 성공 < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.
Example	void CTestDlg::OnMsgReceived(LPCTSTR strHost, short nPort, long lMsgId) { short nStream, nFunction, nWait; long lLen; m_secs.GetMsgInfo(lMsgId, &nStream, &nFunction, &nWait, &lLen); if (nStream==1 && nFunction==1 && nWait) { long lReplyMsgId = m_secs.CreateReplyMsg(lMsgId); //S1F2 m_secs.OpenListItem(lReplyMsgId); m_secs.CloseListItem(lReplyMsgId); m_secs.SendMsg(lReplyMsgId); } else m_secs.**CloseMsg**(lMsgId); }

short GetMsgQueSize ()	*Light Version 해당사항 없음
Function	메시지큐 (Message Que)에서 처리되지 않은(Open) Message 개수를 구합니다.
Parameters	NONE
Return Value	>= 0 ➔ 성공 (Message Que의 크기) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.
Example	short nMsgQueSize = m_secs.**GetMsgQueSize**();

CString GetHeaderMsg (long lMsgId)	*Light Version 해당사항 없음
Function	수신된 Message나 작성중인 Message의 Header부분을 읽어옵니다.
Parameters	Message Id
Return Value	해당 Message의 Header
Example	CString strMsgHeader = m_secs.**GetHeaderMsg**(lMsgId); AfxMessageBox(strMsgHeader);

CString GetSystemByte(long lMsgId)	*Light Version 해당사항 없음
Function	시스템바이트(System Byte)를 읽어옵니다.
Parameters	Message Id
Return Value	해당 Message의 시스템바이트(System Byte)
Example	CString strSystemByte = m_secs.**GetSystemByte**(lMsgId); AfxMessageBox(strSystemByte);

6.수신된 SECS Message의 Item Access

short GetListItemOpen(long lMsgId**)** , **GetListItemClose**(**long** lMsgId) **long GetListItem(long** lMsgId**)**	
Function	Sub Item을 읽기 위해 List Item을 열고 닫습니다. SECS Message작성 및 송신부분에 나온것과 마찬가지로 GetListItemOpen – GetListItemClose 함수가 하나의 쌍으로 작용합니다. <L[2] → **GetListItemOepn**(..) <A MDLN> → GetAsciiItem(..) <A SOFTREV> → GetAsciiItem(..) > → **GetListItemClose**(..) // 더 이상 access하지 않는 경우 // 생략가능 L, 2 <A MDLN> <A SOFTREV> ➔ GetListItem(lMsgId); // 함수의 리턴값은 2 (subitem count) GetAsciiItem(lMsgId); GetAsciiItem(lMsgId);
Parameters	Message Id
Return Value	>= 0 ➔ 성공 (List Item속의 SubItem 개수) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.

Example	void CTestDlg::OnMsgReceived(LPCTSTR strHost, short nPort, long lMsgId) { short nStream, nFunction, nWait; long lLen, lValue; m_secs.GetMsgInfo(lMsgId, &nStream, &nFunction, &nWait, &lLen); // S1F3W // <L // <U2 1> // > if (nStream==1 && nFunction==3 && nWait) { m_secs.**GetListItem**(lMsgId); // m_secs.**GetListItemOpen**(..) m_secs.GetU2Item(lMsgId, &lValue); // m_secs.**GetListItemClose**(lMsgId); long lReplyMsgId = m_secs.CreateReplyMsg(lMsgId); //S1F4 m_secs.AddListItem(lReplyMsg, 0); m_secs.SendMsg(lReplyMsgId); } else m_secs.CloseMsg(lMsgId); }

long GetAsciiItem(long lMsgId, BSTR* pszValue)	
Function	해당 Message에서 Ascii Item을 읽습니다.
Parameters	long lMsgId ➔ Message Id BSTR *pszValue ➔ Ascii Item을 읽어들일 포인터변수
Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.

Example	void CTestDlg::OnMsgReceived(LPCTSTR strHost, short nPort, long lMsgId) { short nStream, nFunction, nWait; long lLen; m_secs.GetMsgInfo(lMsgId, &nStream, &nFunction, &nWait, &lLen); // S7F5W // <A "PPSAMPLE"> if (nStream==7 && nFunction==5 && nWait) { char buff[0xff]; BSTR bstrBuff; long lSize = m_secs.**GetAsciiItem**(lMsgId, &bstrBuff); WideCharToMultiByte(CP_ACP,0,bstrBuff,lSize, buff, lSize,NULL,NULL); long lReplyMsgId = m_secs.CreateReplyMsg(lMsgId); //S7F6 m_secs.OpenListItem(lReplyMsgId); m_secs.AddAsciiItem(lReplyMsgId, buff, lSize); m_secs.AddFileBinaryItem(lReplyMsgId,"c:\\temp\\sample"); m_secs.CloseListItem(lReplyMsgId); m_secs.SendMsg(lReplyMsgId); } else m_secs.CloseMsg(lMsgId); }

CString GetAsciiItemString (long lMsgId)	*Light Version 해당사항 없음
Function	해당 Message에서 Ascii Item을 읽습니다. GetAsciiItem(…)과 같이 Ascii Item을 읽어옵니다. 그러나 CString type의 값으로 가져오기 때문에, 변환이 필요없습니다.
Parameters	long lMsgId ➔ Message Id
Return Value	CString형의 Ascii Item Value

Example	void CTestDlg::OnMsgReceived(LPCTSTR strHost, short nPort, long lMsgId) { short nStream, nFunction, nWait; long lLen; m_secs.GetMsgInfo(lMsgId, &nStream, &nFunction, &nWait, &lLen); // S7F5W // <L // <A "PPSAMPLE"> // > if (nStream==7 && nFunction==5 && nWait) { m_secs.GetListItemOpen(lMsgId); CString strPPName = m_secs.**GetAsciiItemString**(lMsgId); m_secs.GetListItemClose(lMsgId); long lReplyMsgId = m_secs.CreateReplyMsg(lMsgId); //S7F6 m_secs.OpenListItem(lReplyMsgid); m_secs.AddAsciiItem(lReplyMsgId, strPPName, strPPName.GetLength()); m_secs.CloseListItem(lReplyMsgId); m_secs.SendMsg(lReplyMsgId); } else { m_secs.CloseMsg(lMsgId); } }

long GetJ8Item(long lMsgId, BSTR* pszValue)	
Function	해당 Message에서 J-8 Item을 읽습니다.
Parameters	long lMsgId ➔ Message Id BSTR *pszValue ➔ Item을 읽어들일 포인터변수

Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.
Example	void CTestDlg::OnMsgReceived(LPCTSTR strHost, short nPort, long lMsgId) { short nStream, nFunction, nWait; long lLen; m_secs.GetMsgInfo(lMsgId, &nStream, &nFunction, &nWait, &lLen); // S65F1 // <J "012ABC"> if (nStream==65 && nFunction==1 && nWait) { long lSize = m_secs.**GetJ8Item**(lMsgId, &bstrBuff); long lReplyMsgId = m_secs.CreateReplyMsg(lMsgId); //S65F2 m_secs.SendMsg(lReplyMsgId); } else { m_secs.CloseMsg(lMsgId); } }

CString GetJ8ItemString (long lMsgId)	*Light Version 해당사항 없음
Function	해당 Message에서 J-8 Item을 읽습니다. GetJ8Item(…)과 같이 J-8 Item을 읽어옵니다. 그러나 CString type의 값으로 가져오기 때문에, 변환이 필요없습니다.
Parameters	long lMsgId ➔ Message Id

Return Value	CString형의 J8 Item Value
Example	void CTestDlg::OnMsgReceived(LPCTSTR strHost, short nPort, long lMsgId) { short nStream, nFunction, nWait; long lLen; m_secs.GetMsgInfo(lMsgId, &nStream, &nFunction, &nWait, &lLen); // S65F1W // <L // <J "012ABC"> // > if (nStream==65 && nFunction==1 && nWait) { m_secs.GetListItemOpen(lMsgId); CString strJIS = m_secs.**GetJ8ItemString**(lMsgId); m_secs.GetListItemClose(lMsgId); long lReplyMsgId = m_secs.CreateReplyMsg(lMsgId); //S65F2 m_secs.OpenListItem(lReplyMsgid); m_secs.CloseListItem(lReplyMsgId); m_secs.SendMsg(lReplyMsgId); } else { m_secs.CloseMsg(lMsgId); } }
long GetFileBinaryItem (long lMsgId, LPCTSTR pszFile)	
Function	해당 Message에서 Binary Item을 읽고 파일로 저장합니다. *Light Version 해당사항 없음
Parameters	long lMsgId ➔ Message Id LPCTSTR pszFile ➔ 저장할 파일 경로 및 파일명

Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.
Example	void CTestDlg::OnMsgReceived(LPCTSTR strHost, short nPort, long lMsgId) { short nStream, nFunction, nWait, nACK=0x00; long lLen; m_secs.GetMsgInfo(lMsgId, &nStream, &nFunction, &nWait, &lLen); // S7F3W // <L // <A "PPSAMPLE"> // <B 0x01 0x02 0xFF …> // > if (nStream==7 && nFunction==3 && nWait) { m_secs.GetListItemOpen(lMsgId); CString strPPName = m_secs.GetAsciiItemString(lMsgId); m_secs.**GetFileBinaryItem**(lMsgId, strPPName); m_secs.GetListItemClose(lMsgId); long lReplyMsgId = m_secs.CreateReplyMsg(lMsgId); //S7F4 m_secs.AddBinaryItem(lReplyMsgId, &nACK, 1); m_secs.SendMsg(lReplyMsgId); } else { m_secs.CloseMsg(lMsgId); } }

long GetBinaryItem (long lMsgId, short *pnValue)	
Function	해당 Message에서 Binary Item을 가져옵니다.
Parameters	long lMsgId ➔ Message Id short *pnValue ➔ Binary Item Value를 저장할 포인터변수

Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.
Example	void CTestDlg::OnMsgReceived(LPCTSTR strHost, short nPort, long lMsgId) { short nStream, nFunction, nWait, nACK=0x00, nValue; UCHAR nArray[0xFF]; long lLen; m_secs.GetMsgInfo(lMsgId, &nStream, &nFunction, &nWait, &lLen); // S12F13W // <L // <A "WAFERMAP01"> // <B 0xFF> // <B 0x01 0x00 0x01 0x00> // > if (nStream==12 && nFunction==13 && nWait) { m_secs.GetListItemOpen(lMsgId); CString strWaferID = m_secs.GetAsciiItemString(lMsgId); m_secs.**GetBinaryItem**(lMsgId, &nValue); m_secs.**GetBinaryItem**(lMsgId, (short *)nArray); m_secs.GetListItemClose(lMsgId); long lReplyMsgId = m_secs.CreateReplyMsg(lMsgId); //S12F14 m_secs.AddBinaryItem(lReplyMsgId, &nACK, 1); m_secs.SendMsg(lReplyMsgId); } else m_secs.CloseMsg(lMsgId); }

long GetBoolItem (long lMsgId, short *pnValue)	
Function	해당 Message에서 Boolean Item을 가져옵니다.
Parameters	long lMsgId ➔ Message Id short *pnValue ➔ Boolean Item Value를 저장할 포인터변수
Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.
Example	void CTestDlg::OnMsgReceived(LPCTSTR strHost, short nPort, long lMsgId) { short nStream, nFunction, nWait, nACK=0x00, nValue, nArray[0xFF]; long lLen; m_secs.GetMsgInfo(lMsgId, &nStream, &nFunction, &nWait, &lLen); // S1F11W // <L // <BOOLEAN 0x01> // <BOOLEAN 0x00 0x01 0x01 0x00 0X01> // > if (nStream==1 && nFunction==11 && nWait) { m_secs.GetListItemOpen(lMsgId); m_secs.**GetBoolItem**(lMsgId, &nValue); m_secs.**GetBoolItem**(lMsgId, nArray); m_secs.GetListItemClose(lMsgId); long lReplyMsgId = m_secs.CreateReplyMsg(lMsgId); //S1F12 m_secs.AddBinaryItem(lReplyMsgId, &nACK, 1); m_secs.SendMsg(lReplyMsgId); } else { m_secs.CloseMsg(lMsgId); } }

long GetI1Item (long lMsgId, short *pnValue)	
Function	해당 Message에서 I1 (Signed Integer 1 Byte) Item을 가져옵니다.
Parameters	long lMsgId ➔ Message Id short *pnValue ➔ I1 Item Value를 저장할 포인터변수
Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.

long GetI2Item (long lMsgId, short *pnValue)	
Function	해당 Message에서 I2 (Signed Integer 2 Byte) Item을 가져옵니다.
Parameters	long lMsgId ➔ Message Id short *pnValue ➔ I2 Item Value를 저장할 포인터변수
Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.

long GetI4Item (long lMsgId, long *plValue)	
Function	해당 Message에서 I4 (Signed Integer 4 Byte) Item을 가져옵니다.
Parameters	long lMsgId ➔ Message Id long *plValue ➔ I4 Item Value를 저장할 포인터변수
Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.

long GetI8Item (long lMsgId, long *plValue)	
Function	해당 Message에서 I8 (Signed Integer 8 Byte) Item을 가져옵니다.
Parameters	long lMsgId ➔ Message Id long *plValue ➔ I8 Item Value를 저장할 포인터변수
Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.

long GetU1Item (long lMsgId, short *pnValue)	
Function	해당 Message에서 U1 (Unsigned Integer 1 Byte) Item을 가져옵니다.
Parameters	long lMsgId ➔ Message Id short *pnValue ➔ U1 Item Value를 저장할 포인터변수
Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.

long GetU2Item (long lMsgId, long *plValue)	
Function	해당 Message에서 U2 (Unsigned Integer 2 Byte) Item을 가져옵니다.
Parameters	long lMsgId ➔ Message Id long *plValue ➔ U2 Item Value를 저장할 포인터변수
Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.

long GetU4Item (long lMsgId, double *prValue)	
Function	해당 Message에서 U4 (Unsigned Integer 4 Byte) Item을 가져옵니다.
Parameters	long lMsgId ➔ Message Id double *prValue ➔ U4 Item Value를 저장할 포인터변수
Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.

long GetU8Item (long lMsgId, double *prValue)	
Function	해당 Message에서 U8 (Unsigned Integer 8 Byte) Item을 가져옵니다.
Parameters	long lMsgId ➔ Message Id double *prValue ➔ U8 Item Value를 저장할 포인터변수
Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.

long GetF4Item (long lMsgId, float *pfValue)	
Function	해당 Message에서 F4 (Floating Point 4 Byte) Item을 가져옵니다.
Parameters	long lMsgId ➔ Message Id float *pfValue ➔ F4 Item Value를 저장할 포인터변수
Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.

long GetF8Item (long lMsgId, double *prValue)	
Function	해당 Message에서 F8 (Floating Point 8 Byte) Item을 가져옵니다.
Parameters	long lMsgId ➔ Message Id double *prValue ➔ F8 Item Value를 저장할 포인터변수
Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.

short GetSubItemFormat(long lMsgId)	
Function	List Item을 읽은후 그 List Item의 첫번째 SubItem Format을 가져옵니다.
Parameters	long lMsgId ➔ Message Id
Return Value	0 = 현재 읽어들인 List Item의 첫번째 Sub Item도 List임 -1 = 더 이상 Sub Item이 존재하지 않음 10 = BINARY, 11=BOOLEAN, 20=ASCII, 21=JIS8 30 = INT8B, 31=INT1B, 32=INT2B, 34=INT4B 40 = FLOAT8B, 44=FLOAT4B, 50=UINT8B, 51=UINT1B 52 = UINT2B, 54= UINT4B

short GetNextItemFormat(long lMsgId)	
Function	다음 Item의 Format을 가져옵니다.
Parameters	long lMsgId ➔ Message Id
Return Value	Format Code는 위와 동일하며, 매뉴얼의 부록편에도 있습니다.

long GetArrayItem(long lMsgId, BSTR * pszValue)	
Function	포인터를 사용할수 없는 개발언어(ex C#)에서 배열아이템을 읽어오기위한 별도의 특별한 함수입니다. ● Item값을 추가하는 함수는 AddArrayItem입니다.
Parameters	. long lMsgId ➔ 작성중인 MessageID . BSTR *pszValue ➔ 값을 읽어들일 문자열변수 값은 구분자로 연결되어 반환됩니다. 예) "1,2,3,4,5"
Return Value	> 0 ➔ 성공 (읽어들인 배열 item count) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.
Example	

7.Direct Item Access

아주 복잡한 SECS Message 구조에서 특정 Item을 빨리 Access하여
해당 Item Value를 얻고자 할 때, Direct Access Method를 이용합니다.

*Light Version 해당사항 없음

일반적인 SML 구조적 방법으로의 Item접근	
S2F33W <L <U1 0> <L <L <I2 100> <L **<U2 10>** > > >	If (nStream==2 && nFunction==33) { m_secs.GetListItemOpen(lMsgId); m_secs.GetU1Item(lMsgId, &nU1); m_secs.GetListItemOpen(lMsgId); m_secs.GetListItemOpen(lMsgId); m_secs.GetI2Item(lMsgId,&nI2); m_secs.GetListItemOpen(lMsgId); m_secs.GetU2Item(lMsgId, &nU2); m_secs.GetListItemClose(lMsgId); m_secs.GetListItemClose(lMsgId); m_secs.GetListItemClose(lMsgId); m_secs.GetListItemClose(lMsgId); }

보다 개선된 Direct Access Method를 통한 Item접근	
S2F33W <L / <U1 0> /1 <L /2 <L /2/1/ <I2 100> /2/1/1 <L /2/1/2 **<U2 10>** **/2/1/2/1** > > > >	➔ Direct Access Method를 이용하면 If (nStream==2 && nFunction==33) { m_secs.**GetU2ItemD**(lMsgId, "/2/1/2/1" , &val); }

■ Direct Access Method는 앞에서 언급된 일반 Access method 마지막에 대문자 "D"가 붙어 있고, 위치를 가르키는 지시자(Location Pointer) 파라미터가 있습니다. 사용법은 일반 Access method와 동일합니다.

long GetAsciiItemD (long lMsgId, LPCTSTR strPointer, BSTR *pszValue)	
Function	해당 Message에서 위치지시자가 가르키는 Ascii Item을 가져옵니다. *Light Version 해당사항 없음
Parameters	long lMsgId ➔ Message Id LPCTSTR strPointer ➔ 위치를 가르키는 위치지시자(Location Pointer) BSTR *pszValue ➔ Ascii Item Value를 저장할 포인터변수
Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조.

CString GetAsciiItemStringD (long lMsgId, LPCTSTR strPointer)	
Function	해당 Message에서 위치지시자가 가르키는 Ascii Item (String Type)을 가져옵니다. *Light Version 해당사항 없음
Parameters	long lMsgId ➔ Message Id LPCTSTR strPointer ➔ 위치를 가르키는 위치지시자(Location Pointer)
Return Value	String Type의 Ascii Item

long GetJ8ItemD (long lMsgId, LPCTSTR strPointer, BSTR *pszValue) *Light Version 해당사항 없음	
Function	해당 Message에서 위치지시자가 가르키는 JIS-8 Item을 가져옵니다.
Parameters	long lMsgId ➔ Message Id LPCTSTR strPointer ➔ 위치를 가르키는 위치지시자(Location Pointer) BSTR *pszValue ➔ 읽어올 Item을 저장할 포인터변수
Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조

CString GetJ8temStringD (long lMsgId, LPCTSTR strPointer)	
Function	해당 Message에서 위치지시자가 가르키는 J8 Item (String Type)을 가져옵니다. *Light Version 해당사항 없음
Parameters	long lMsgId ➔ Message Id LPCTSTR strPointer ➔ 위치를 가르키는 위치지시자(Location Pointer)
Return Value	String Type의 JIS-8 Item

long GetFileBinaryItemD (long lMsgId, LPCTSTR strPointer, LPCTSTR pszFile)	
Function	해당 Message에서 위치지시자가 가르키는 Binary Item을 읽고 파일로 저장합니다. *Light Version 해당사항 없음
Parameters	long lMsgId ➔ Message Id LPCTSTR strPointer ➔ 위치를 가르키는 위치지시자(Location Pointer) LPCTSTR pszFile ➔ 읽어올 Binary Item을 저장할 파일 경로 및 파일명
Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조

long GetBinaryItemD (long lMsgId, LPCTSTR strPointer, short *pnValue) *Light Version 해당사항 없음	
Function	해당 Message에서 위치지시자가 가르키는 Binary Item을 가져옵니다.
Parameters	long lMsgId ➔ Message Id LPCTSTR strPointer ➔ 위치를 가르키는 위치지시자(Location Pointer) short *pnValue ➔ 읽어올 Item을 저장할 포인터변수
Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조

long GetBoolItemD (long lMsgId, LPCTSTR strPointer, short *pnValue) *Light Version 해당사항 없음	
Function	해당 Message에서 위치지시자가 가르키는 Boolean Item을 가져 옵니다.
Parameters	long lMsgId ➔ Message Id LPCTSTR strPointer ➔ 위치를 가르키는 위치지시자(Location Pointer) short *pnValue ➔ 읽어올 Item을 저장할 포인터변수
Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조

long GetI1ItemD (long lMsgId, LPCTSTR strPointer, short *pnValue) *Light Version 해당사항 없음	
Function	해당 Message에서 위치지시자가 가르키는 I1 Item을 가져 옵니다.
Parameters	long lMsgId ➔ Message Id LPCTSTR strPointer ➔ 위치를 가르키는 위치지시자(Location Pointer) short *pnValue ➔ 읽어올 Item을 저장할 포인터변수
Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조

long GetI2ItemD (long lMsgId, LPCTSTR strPointer, short *pnValue) *Light Version 해당사항 없음	
Function	해당 Message에서 위치지시자가 가르키는 I2 Item을 가져 옵니다.
Parameters	long lMsgId ➔ Message Id LPCTSTR strPointer ➔ 위치를 가르키는 위치지시자(Location Pointer) short *pnValue ➔ 읽어올 Item을 저장할 포인터변수
Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조

long GetI4ItemD (long lMsgId, LPCTSTR strPointer, long *plValue) *Light Version 해당사항 없음

Function	해당 Message에서 위치지시자가 가르키는 I4 Item을 가져 옵니다.
Parameters	long lMsgId ➔ Message Id LPCTSTR strPointer ➔ 위치를 가르키는 위치지시자(Location Pointer) long *plValue ➔ 읽어올 Item을 저장할 포인터변수
Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조

long GetI8ItemD (long lMsgId, LPCTSTR strPointer, long *plValue) *Light Version 해당사항 없음	
Function	해당 Message에서 위치지시자가 가르키는 I8 Item을 가져 옵니다.
Parameters	long lMsgId ➔ Message Id LPCTSTR strPointer ➔ 위치를 가르키는 위치지시자(Location Pointer) long *plValue ➔ 읽어올 Item을 저장할 포인터변수
Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조

long GetU1ItemD (long lMsgId, LPCTSTR strPointer, short *pnValue) *Light Version 해당사항 없음	
Function	해당 Message에서 위치지시자가 가르키는 U1 Item을 가져 옵니다.
Parameters	long lMsgId ➔ Message Id LPCTSTR strPointer ➔ 위치를 가르키는 위치지시자(Location Pointer) short *pnValue ➔ 읽어올 Item을 저장할 포인터변수
Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조

long GetU2ItemD (long lMsgId, LPCTSTR strPointer, long *plValue) *Light Version 해당사항 없음	
Function	해당 Message에서 위치지시자가 가르키는 U2 Item을 가져 옵니다.

Parameters	long lMsgId ➔ Message Id LPCTSTR strPointer ➔ 위치를 가르키는 위치지시자(Location Pointer) long *plValue ➔ 읽어올 Item을 저장할 포인터변수
Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조

long GetU4ItemD (long lMsgId, LPCTSTR strPointer, double *prValue) *Light Version 해당사항 없음	
Function	해당 Message에서 위치지시자가 가르키는 U4 Item을 가져 옵니다.
Parameters	long lMsgId ➔ Message Id LPCTSTR strPointer ➔ 위치를 가르키는 위치지시자(Location Pointer) double *prValue ➔ 읽어올 Item을 저장할 포인터변수
Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조

long GetU8ItemD (long lMsgId, LPCTSTR strPointer, double *prValue) *Light Version 해당사항 없음	
Function	해당 Message에서 위치지시자가 가르키는 U8 Item을 가져 옵니다.
Parameters	long lMsgId ➔ Message Id LPCTSTR strPointer ➔ 위치를 가르키는 위치지시자(Location Pointer) double *prValue ➔ 읽어올 Item을 저장할 포인터변수
Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조

long GetF4ItemD (long lMsgId, LPCTSTR strPointer, float *pfValue) *Light Version 해당사항 없음	
Function	해당 Message에서 위치지시자가 가르키는 F4 Item을 가져 옵니다.
Parameters	long lMsgId ➔ Message Id LPCTSTR strPointer ➔ 위치를 가르키는 위치지시자(Location Pointer) float *pfValue ➔ 읽어올 Item을 저장할 포인터변수

Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조

long GetF8ItemD (long lMsgId, LPCTSTR strPointer, double *prValue) *Light Version 해당사항 없음	
Function	해당 Message에서 위치지시자가 가르키는 F8 Item을 가져 옵니다.
Parameters	long lMsgId ➔ Message Id LPCTSTR strPointer ➔ 위치를 가르키는 위치지시자(Location Pointer) double *prValue ➔ 읽어올 Item을 저장할 포인터변수
Return Value	>= 0 ➔ 성공 (해당 Item의 Size) < 0 ➔ 실패 (ErrorCode반환) ErrorCode는 본 매뉴얼의 Error Code List를 참조

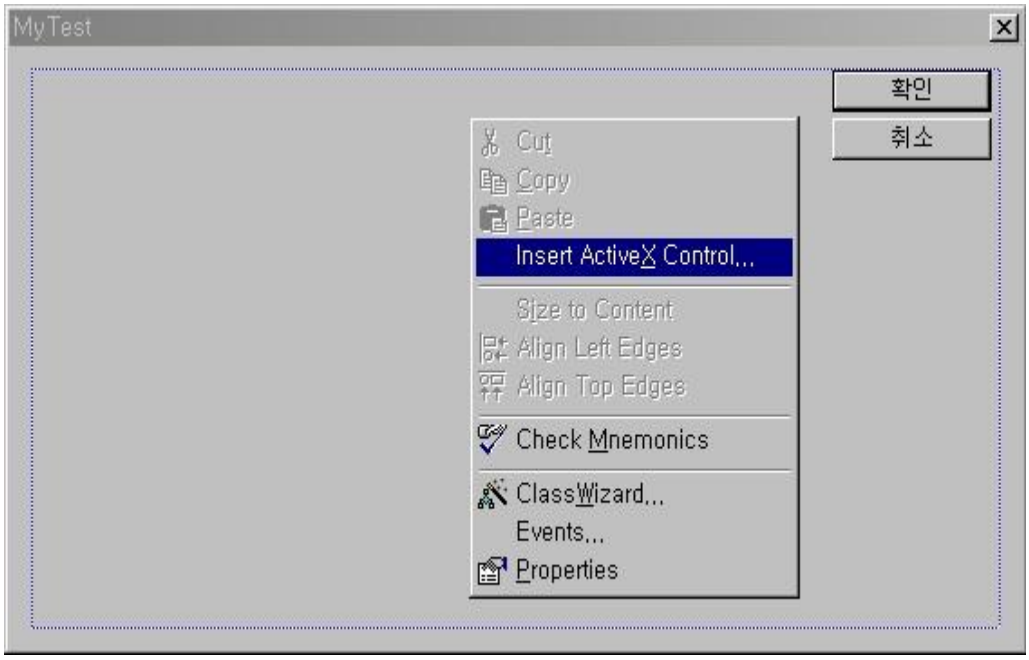
void EnableErrorMessage() void DisableErrorMessage()	
Function	S9계열의 Message (S9F1, S9F3, S9F5, S9F7, S9F9)를 자동으로 전송할지 여부를 설정합니다. (EQ mode인 경우)
Parameters	NONE
Return Value	NONE

short GetRuntimeState()	
Function	현재 구동되고 있는 제품의 정품여부를 확인하는 함수
Parameters	NONE
Return Value	0 ➔ Evaluation Mode 1 ➔ Runtime Mode

8. MFC를 이용한 Sample Program 작성 Guide

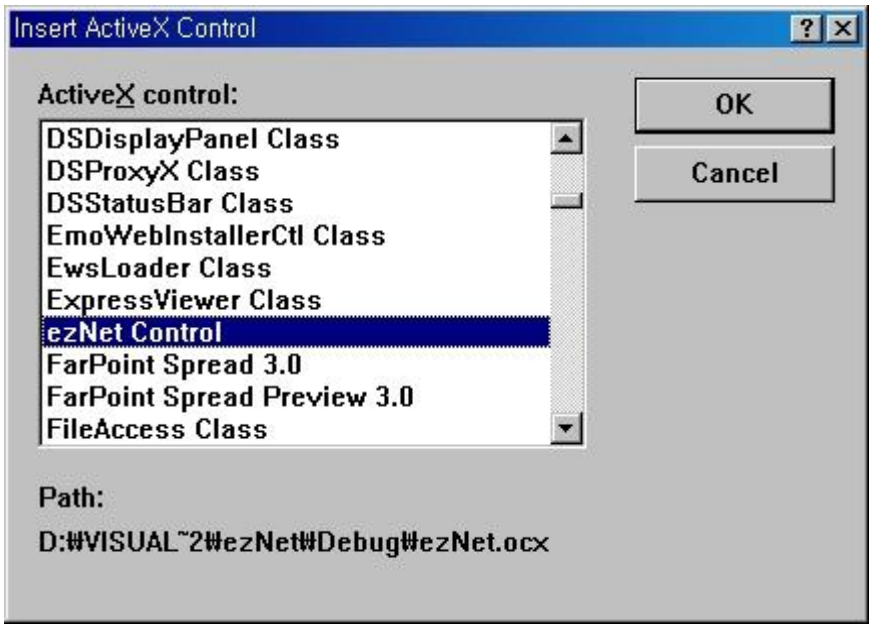
1) ezNet ActiveX Control 등록

다이얼로그에서 오른쪽 마우스버튼를 클릭하여 아래 그림과
같이 Context Menu에서 "**Insert ActiveX Control**"을 선택합니다.



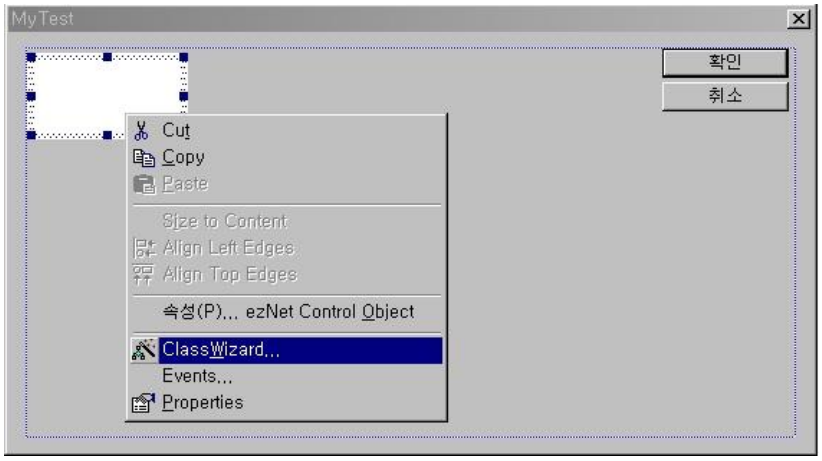
2) ezNet ActiveX Control 추가

ActiveX Control List에서 **ezNet Control**을 선택합니다.

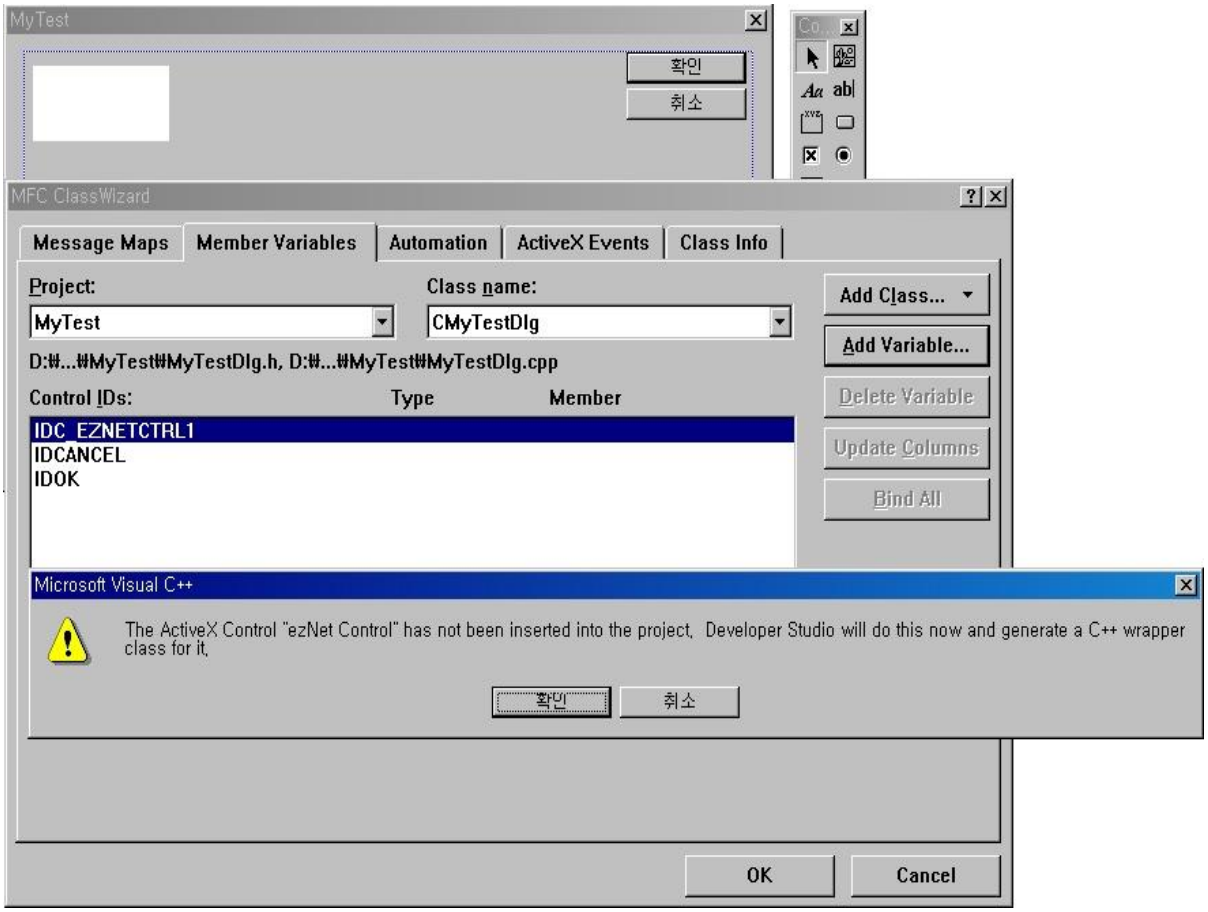


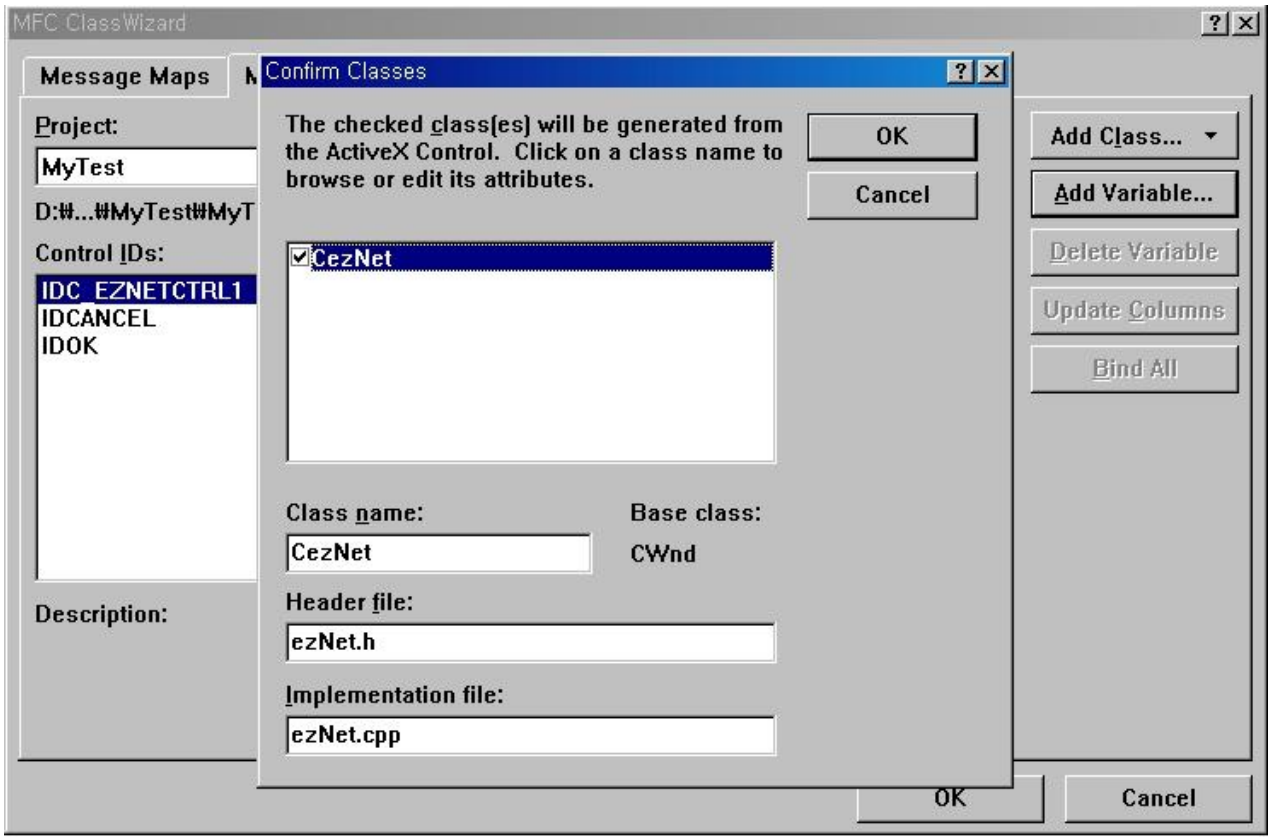
3) 클래스위저드(Class Wizard) 호출

마우스포인터를 추가된 ActiveX Control 에 놓고 오른쪽 마우스버튼를 클릭하여 아래 그림과 같이 Context Menu에서 "**Class Wizard**" 를 선택합니다.

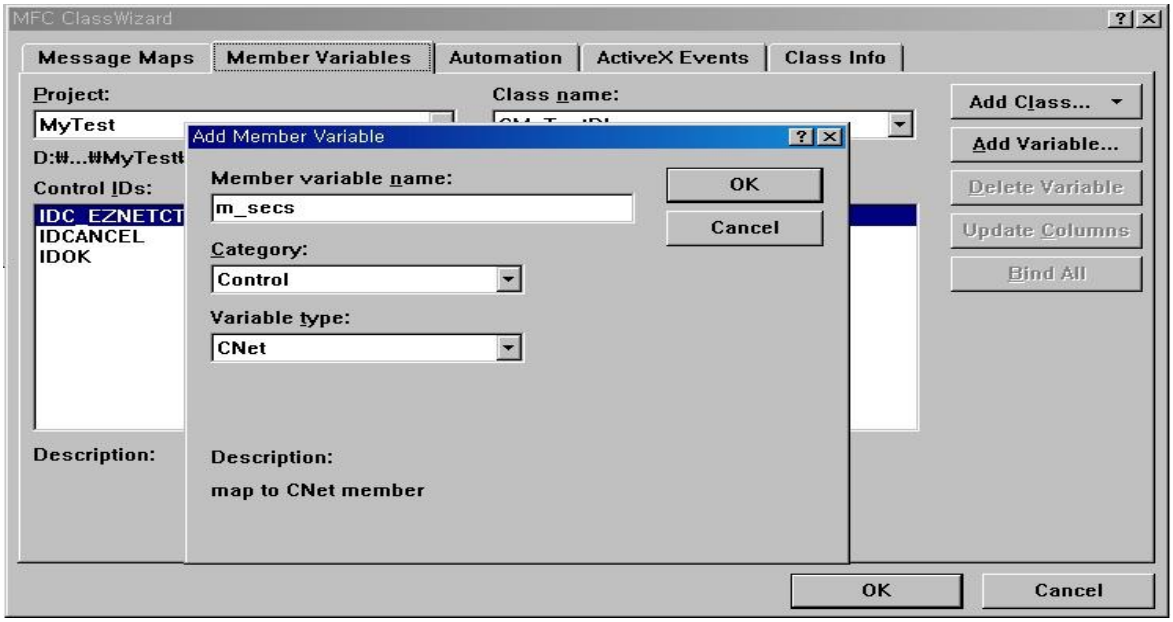


4) 클래스등록 (Wrapper Class추가)

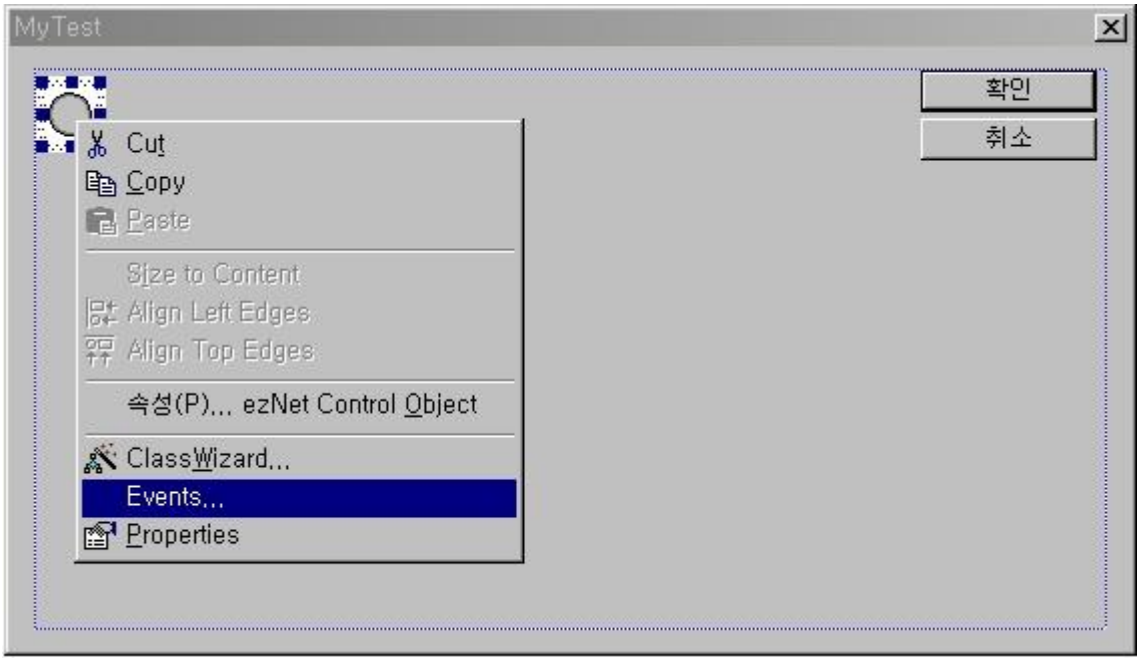


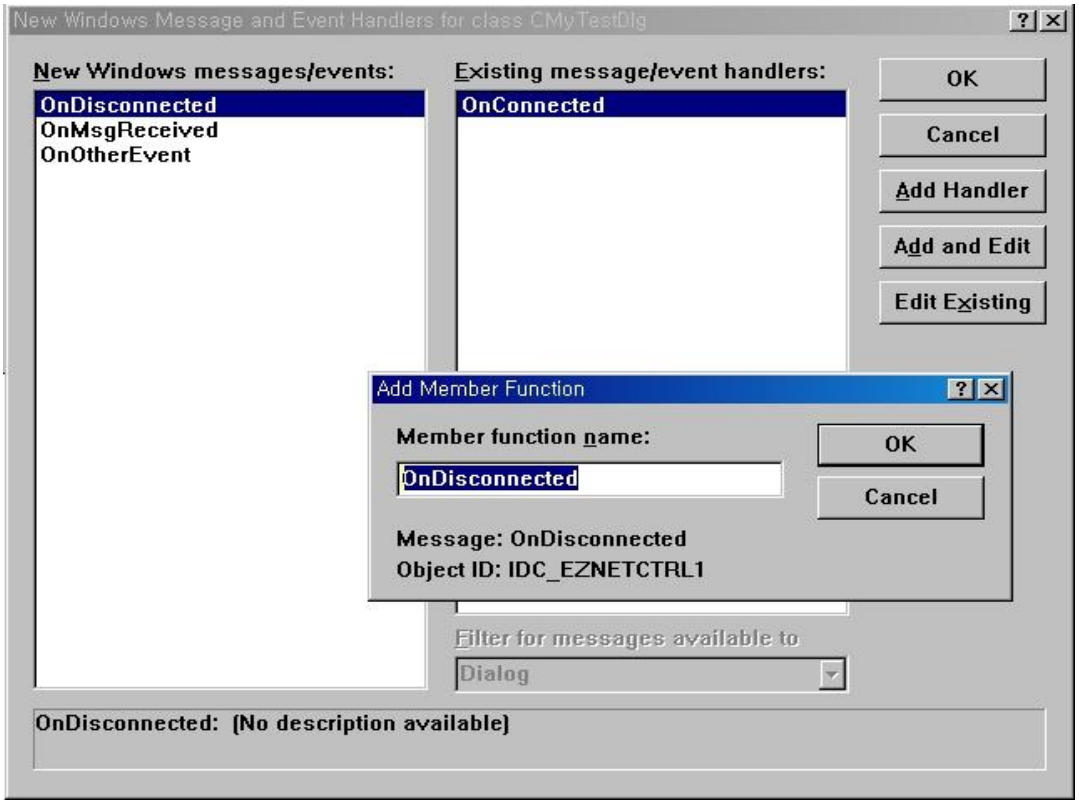


5) 클래스멤버변수(Class Member Variable) 지정



6) 이벤트핸들러 추가


MyTest
확인
취소
Cut
Copy
Paste
Size to Content
Align Left Edges
Align Top Edges
속성(P)... ezNet Control Object
ClassWizard...
Events...
Properties


New Windows Message and Event Handlers for class CMyTestDlg
New Windows messages/events:
OnDisconnected
OnMsgReceived
OnOtherEvent
Existing message/event handlers:
OnConnected
OK
Cancel
Add Handler
Add and Edit
Edit Existing
Add Member Function
Member function name:
OnDisconnected
OK
Cancel
Message: OnDisconnected
Object ID: IDC_EZNETCTRL1
Filter for messages available to
Dialog
OnDisconnected: (No description available)

7) 프로그램 코딩(Coding)

```
BEGIN_EVENTSINK_MAP(CMyTestDlg, CDialog)
    //{{AFX_EVENTSINK_MAP(CMyTestDlg)
    ON_EVENT(CMyTestDlg, IDC_EZNETCTRL1, 1   /* OnConnected */, OnConnected, VTS_BSTR VTS_I2)
    ON_EVENT(CMyTestDlg, IDC_EZNETCTRL1, 2   /* OnDisconnected */, OnDisconnected, VTS_BSTR VTS_I2)
    ON_EVENT(CMyTestDlg, IDC_EZNETCTRL1, 3   /* OnMsgReceived */, OnMsgReceived, VTS_I4)
    //{{AFX_EVENTSINK_MAP
END_EVENTSINK_MAP()

void CMyTestDlg::OnConnected(LPCTSTR strHost, short nPort)
{
    // TODO:   Add you control notification handler code here
    AfxMessageBox("연결되었습니다");
}

void CMyTestDlg::OnDisconnected(LPCTSTR strHost, short nPort)
{
    // TODO:   Add you control notification handler code here
    AfxMessageBox("연결이 끊어졌습니다");
}

void CMyTestDlg::OnMsgReceived(long lMsgId)
{
    // TODO:   Add you control notification handler code here
    short nStream,nFunction,nWbit;, nAck=0, nResult
    long lMsgLength;
    CString strTemp;

    m_secs.GetMsgInfo(lmsgId, &nStream, &nFunction, &nWbit, &lMsglength);

    strTemp.Format("메시지가 들어왔습니다.   S%dF%d, Length=%ld", nStream,nFunction,lMsgLength);
    AfxMessageBox(strTemp);

    if   (nStream == 1 && nFunction == 1 && nWbit)    // "Are you there ?"
    {
        long lMsgIdReply = m_secs.CreateReplyMsg(lMsgId);         // S1F2

        m_secs.OpenListItem(lMsgId_Reply);
        m_secs.CloseListItem(lMsgId_Reply);

        nResult = m_secs.SendMsg(llMsgId_Reply)
        if (nResult != 0)
          AfxMessageBox("S1F2전송 실패했습니다.");
    }
    else if (nStream == 1 && nFunction == 13 && nWbit)
    {
        long lMsgIdReply = m_secs.CreateReplyMsg(lMsgId);         // S1F14

        m_secs.OpenListItem(lMsgId_Reply);
            m_secs.AddBinaryItem(lMsgId_Reply, nAck, 1);
```

```
            m_secs.OpenListItem(lMsgId_Reply);
            m_secs.CloseListItem(lMsgId_Reply);
        m_secs.CloseListItem(lMsgId_Reply);

        nResult = m_secs.SendMsg(lIMsgId_Reply)
    }
    else
        m_secs.CloseMsg(lMsgId);
}

void CMyTestDlg::OnButtonS1F1()
{
    long lMsgId = m_secs.CreateMsg(1,1,TRUE);
    m_secs.SendMsg(lMsgId);
}
```

9. SML Format Editor 사용법 및 구성

1. 소개 (Introduction)

ezNet SML Format Editor는 수신할 SECS Message Format(Structure)을 미리 정의하여 편집하는 보조 프로그램입니다.

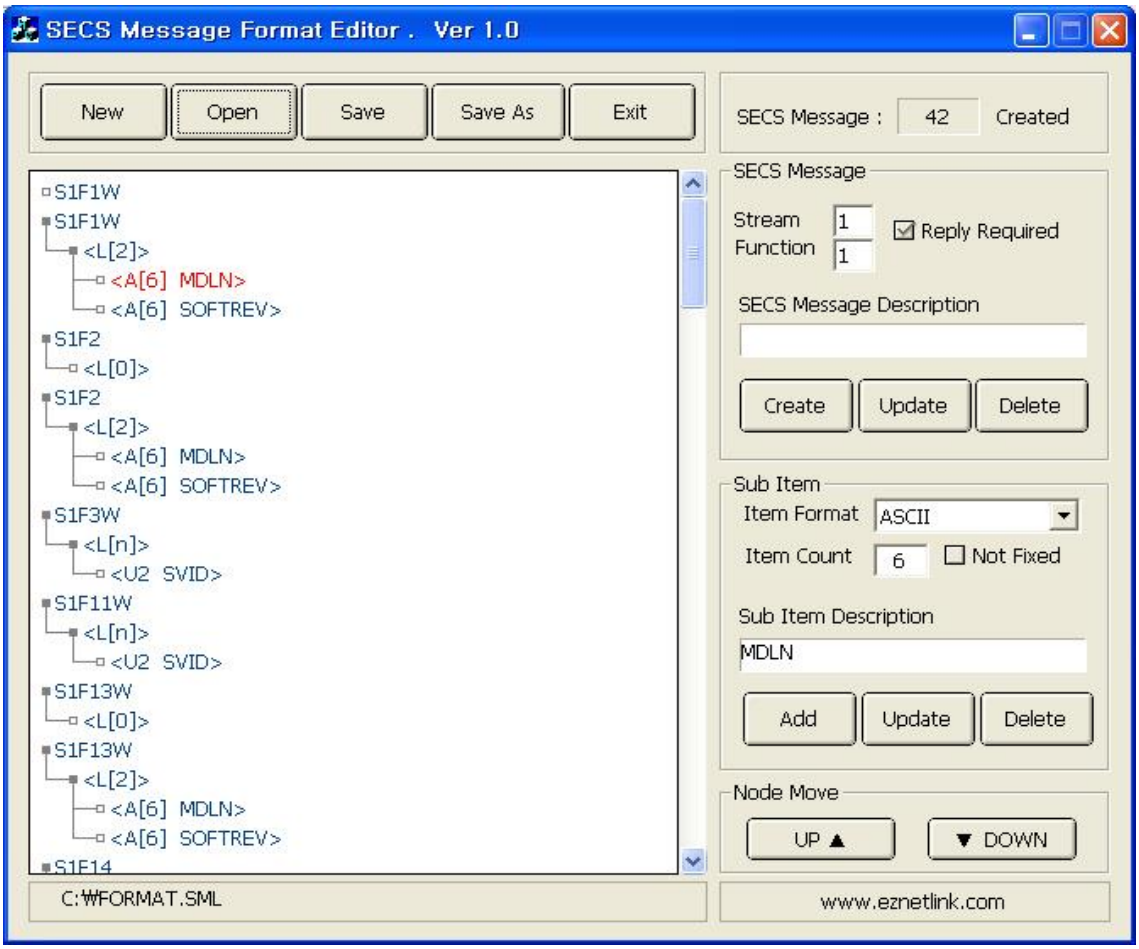
2. 특징 (Features)

- 트리형식의 노드로 구성되어 있어 아주 편리하게 편집할 수 있습니다.

-

- **VARIANT** Item Type과 **Unfixed** Item Size의 도입으로
  모든 Dynamic Message Structure도 표현할 수 있습니다.

-깔끔한 GUI를 이용하여 사용자에게 친숙함을 제공합니다.



3. 포맷파일 적용 설정

포맷파일에 등록된 Structure를 적용하고자 하는 경우, 환경설정파일이나
소스코드에서 그 기능을 Activate시킨후 포맷파일을 지정합니다.
아래의 두가지 방법중 한가지를 선택하시면 됩니다.

(1) 환경설정파일에서 지정할 경우

```
FORMATCHECK=TRUE
FORMATFILE = "C:\\MYFORMAT.SML"
```

(2) 프로그램 소스코드에서 지정할 경우. (Visual C++기준)

```
m_secs.SetFormatCheck(TRUE);
m_secs.SetFormatFile("C:\\MYFORMAT.SML");
```

4. 포맷구성

4.1 참고 및 주의사항

4.1.1 하나의 SECS Message의 마지막에는 마침표로서 Message의 끝을 알립니다.
4.1.2 동일한 Stream/Function을 다중으로 정의할 수 있습니다.
4.1.3 수신 Message만 정의하고 송신 Message는 정의하지 않습니다.
4.1.4 Item의 크기(SIZE)가 1인 경우 생략해도 됩니다.
4.1.5 List Item의 크기는 생략이 가능합니다.
4.1.6 한줄 주석문은 Double Slash입니다. //주석문
4.1.7 여러줄의 주석문은 /*주석문 …. */ 으로 표현할 수 있습니다.

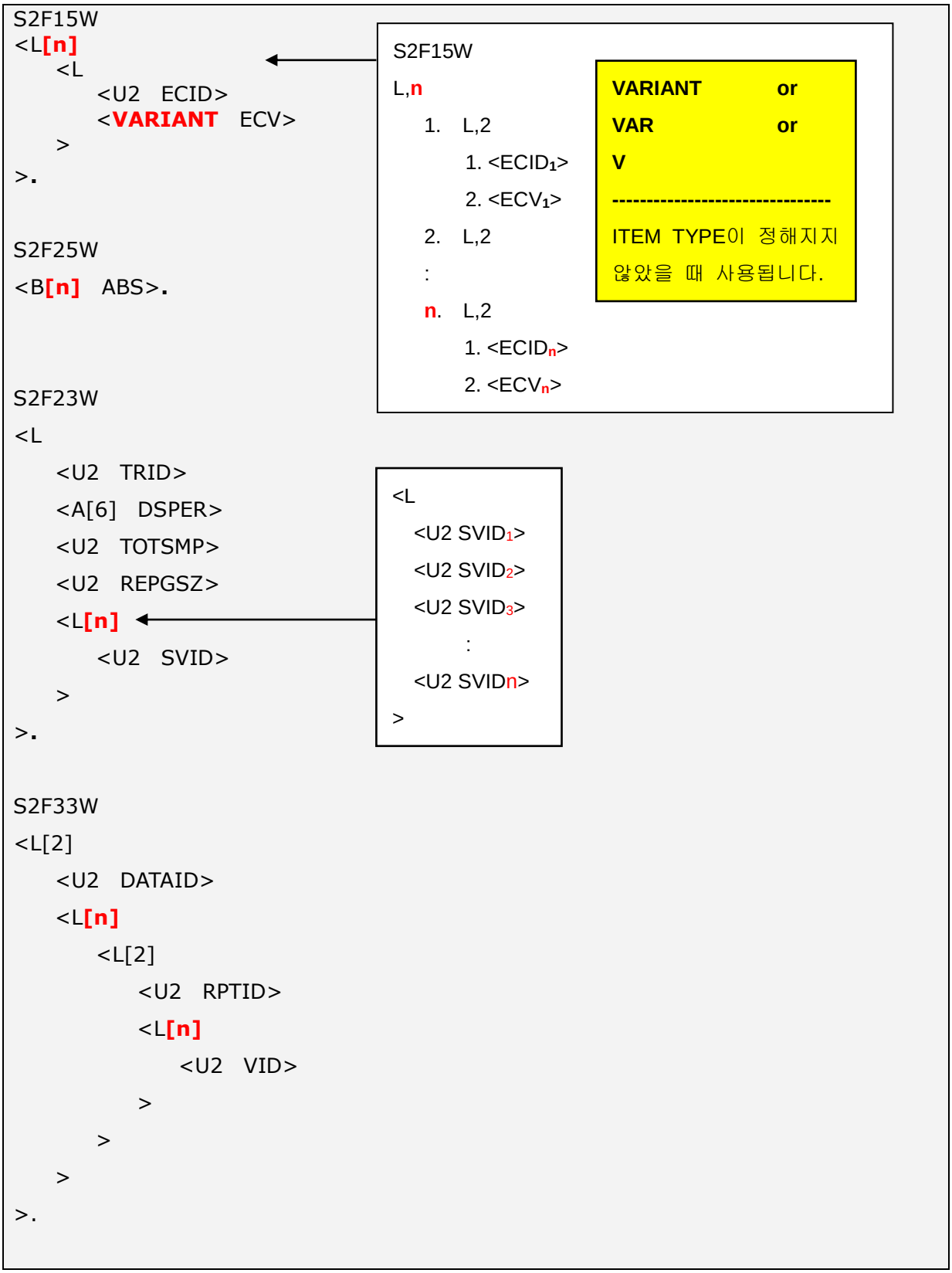
4.2 사용 예

```
S1F1W.

S1F1W
<L>.

S1F3W
<L[n]
  <U2  SVID>
>.

S1F13W
<L>.

S1F13W
<L
<A[6]  MDLN>
<A[6]  SOFTREV>
>.

S1F14
<L
<B  COMMACK>
<L[0]>
>.

S1F14
<L
<B  COMMACK>
<L
<A[6]  MDLN>
<A[6]  SOFTREV>
>
>.
```

S1F1W 이 다중으로 정의되었습니다.

[n]은 Not Fixed Size입니다.
0~n 개 까지 수신할 수 있다는 뜻입니다.

Ascii Item의 경우 문자열의 최대크기 입니다.
또한 반드시 크기를 표시해야 합니다.


S2F15W
<L[n]
<L
<U2 ECID>
<VARIANT ECV>
>
>.
S2F15W
L,n
1. L,2
1. <ECID1>
2. <ECV1>
2. L,2
:
n. L,2
1. <ECIDn>
2. <ECVn>
VARIANT or
VAR or
V
ITEM TYPE이 정해지지 않았을 때 사용됩니다.
S2F25W
<B[n] ABS>.
S2F23W
<L
<U2 TRID>
<A[6] DSPER>
<U2 TOTSMP>
<U2 REPGSZ>
<L[n]
<U2 SVID>
>
>.
<L
<U2 SVID1>
<U2 SVID2>
<U2 SVID3>
:
<U2 SVIDn>
>
S2F33W
<L[2]
<U2 DATAID>
<L[n]
<L[2]
<U2 RPTID>
<L[n]
<U2 VID>
>
>
>
>.

```
S2F35W
<L[2]
    <U2  DATAID>
    <L[n]
        <L[2]
            <U2  ECID>
            <L[n]
                <U2  RPTID>
            >
        >
    >
>.

S2F41W
<L[2]
    <A[40]  RCMD>
    <L[n]
        <L[2]
            <A[40]  CPNAME>
            <VARIANT  CPVAL>
        >
    >
>.
```

10 .(Appendix I) 에러코드 (Error Codes)

Name	Value	Description
LICENSE_PROBLEM	-1	라이센스(License) 문제 발생
CONFIG_FILE_NOTFOUND	-101	환경설정파일(Configuration File)이 존재하지 않음
FILE_NOT_FOUND	-102	파일이 존재하지 않음
FILE_IO_FAILURE	-103	파일 입출력 (I/O) 실패
FILE_CREATION_ERROR	-104	파일 생성 실패
NO_ITEM	-201	아이템(Item)이 존재하지 않음
TYPE_MISMATCH	-202	ITEM의 타입이 맞지 않음
SIZE_MISMATCH	-203	ITEM의 크기/개수가 맞지 않음
ITEM_I1_OVERFLOW	-204	I1 (1Byte Integer) Overflow발생
ITEM_I2_OVERFLOW	-205	I2 (2Byte Integer) Overflow발생
ITEM_I4_OVERFLOW	-206	I4 (4Byte Integer) Overflow발생
ITEM_U1_OVERFLOW	-207	U1(1Byte Unsigned Integer) Overflow발생
ITEM_U2_OVERFLOW	-208	U2(2Byte Unsigned Integer) Overflow발생
ITEM_U4_OVERFLOW	-209	U4(4Byte Unsigned Integer) Overflow발생
INVALID_MSGID	-210	유효하지않은 Message ID
INVALID_STREAM	-211	유효하지않은 Stream번호
INVALID_FUNCTION	-212	유효하지않은 Function번호
DUPLICATE_SYSTEMBYTE	-213	시스템바이트의 중복
UNDEFINED_STRUCTURE	-214	정의되지 않은 Message Structure를 수신
NOT_CONNECTED	-301	아직 연결이 안된 상태임
LOW_LEVEL_ERROR	-302	Operating System상의 Error
NOT_SUPPORTED	-303	지원되지 않는 기능
ALREADY_CONNECTED	-304	이미 연결되어 있음
ALREADY_STATRED	-305	이미 구동된 상태임
THREAD_NULL	-306	각종 쓰레드(THREAD) 에러
CREATE_EVENT_FAIL	-307	이벤트(EVENT) 발생 에러
SERIAL_OPEN_FAIL	-308	시리얼포트 구동 에러
SERIAL_SETUP_FAIL	-309	시리얼포트 초기화 에러
TIMER_CREATE_FAIL	-310	타이머 생성 실패
NOT_STARTED_YET	-311	프로그램이 구동되지 않았음
SOCKET_INUSE	-501	소켓(Socket)이 이미 사용중임

SOCKET_STARTERROR	-502	소켓 구동 에러
SOCKET_INVALID	-503	소켓이 유효하지 않음
SOCKET_WINDOWERROR	-504	메시지통신을 위한 Worker Window에러
SOCKET_LOCALNAME	-505	Host Name을 알아오는 과정에서 에러발생
SOCKET_CONNECTERROR	-506	소켓접속에러
SOCKET_SETEVENT	-507	소켓이벤트설정에러
SOCKET_RESOLVE	-508	소켓함수 중 Resolve() 수행 중 에러

11. (Appendix II) 이벤트코드 (Event Codes)

ezNet SECS driver가 자체제공하는 이벤트함수중에서

void OnOtherEvent(LPCTSTR lpszHost, short nPort, short nEventCode,long lParm)

에서 처리되는 Event Code입니다.

Name	Value	Description	Parameter
EVALUATION_MODE_START	10	평가판 (Evaluation Mode) 제품 모드 시작	
RUNTIME_MODE_START	11	정식제품 모드 시작	
LINKTEST_REQUEST_IN	101	LinkTest.Req (요청) 수신	0
LINKTEST_REQUEST_OUT	102	LinkTest.Req (요청) 발신	0
LINKTEST_RESPONSE_IN	103	LinkTest.Res (응답) 수신	0
LINKTEST_RESPONSE_OUT	104	LinkTest.Res (응답) 발신	0
TRANSACTION_TIMEOUT	202	S9F9 (Transaction Timeout) 발생	Message ID
UNRECOGNIZED_DEVICEID	203	S9F1(정의되지 않은 디바이스ID) 수신	Device ID
UNRECOGNIZED_STREAM	204	S9F3 (정의되지 않은 스트림) 수신	Stream#
UNRECOGNIZED_FUNCTION	205	S9F5 (정의되지 않은 펑션) 수신	Function #
INVALID_DATA	206	S9F7 (유효하지 않은 데이터) 수신	Message ID
DISCARD_MSG	207	처리되지 않은 메시지가 폐기 처리됨	Message ID
TIMEOUT_T1	301	T1 발생	
TIMEOUT_T2	302	T2 발생	
TIMEOUT_T3	303	T3 발생	
TIMEOUT_T4	304	T4 발생	
TIMEOUT_T5	305	T5 발생	
TIMEOUT_T6	306	T6 발생	
TIMEOUT_T7	307	T7 발생	
TIMEOUT_T8	308	T8 발생	

12. (Appendix III) Item Format Code

수신한 **SECS Message**를 읽고 각 **Item**에 접근시
Sub Item이나 다음 **Item**의 **Format**을 알고자 할 때 아래 두개의 메소드를 이용합니다**.**

short GetSubItemFormat(long lMsgId);
short GetNextItemFormat(long lMsgId);

이 두 메소드가 반환하는 **Item Format Code**는 아래와 같습니다**.**

Item Format	Value	Description
L	0	LIST ITEM
B	10	BINARY ITEM
BOOL	11	BOOLEAN ITEM
A	20	ASCII ITEM
J8	21	JIS-8 Byte ITEM
I1	31	1 Byte Signed Integer
I2	32	2Byte Signed Integer
I4	34	4 Byte Signed Integer
I8	30	8 Byte Signed Integer
U1	51	1 Byte Unsigned Integer
U2	52	2 Byte Unsigned Integer
U4	54	4 Byte Unsigned Integer
U8	50	8 Byte Unsigned Integer
F4	44	4 Byte Floating Pointer
F8	40	8 Byte Floating Pointer
NO ITEM	-1	더 이상 Item이 존재하지 않음