SECSeIIme Operation Guide

Release 3.7.E Released Edition
2017/08/24

Contents

CONTENTS 2
SECSEIIME OPERATION GUIDE 5
Introduction *5*
About SECSeIIme 5
SECSeIImeの目的 5
免責 5
本書について 5
Program *6*
Copyright 7
Program Environment 7
確認されている環境 7
問い合わせ 7
SECSeIImeの構造 *8*
SECS-I 8
SECS-Iでは１対１の通信をRS-232Cで行います。 8
SECSeIImeはMaster/SlaveどちらのSimulationも可能です。 8
HSMS 8
HSMS通信はWinSockを用いています。 8
自分のIP Address 8
Task 8
補足 8
Start Up / Program End *9*
起動 9
終了 9
補足 9
Begging *10*
起動の前に 10
まず始めにProjectを作る必要が有ります。 10
ProjectのFile 10
補足 10
SECSEIIME HOW TO OPERATION 11
Setting Section *11*
Directory 11
Operation 11
Protocol 12
SECS-I (RS-232C) 12
HSMS-SS (TCP/IP) 12
Master/(slave) 12
補足 12
Communication Parameter 13
Setting Parameter 13
補足 13
Sender Control Option 13
補足 13
Viewer Option 14
Display Service Option 14
補足 14
Select of Editor 14
補足 14
Header Option 15
Header Check Option 15
補足 15
Header Option 15

補足 15
Protocol Tester *16*
Protocol Simulator / Header Testerの起動 16
SECS-I Level Protocol Simulator 16
Function 16
補足 16
HSMS Header Tester 17
Function 17
補足 17
Stream Function Editor Section *18*
Stream Function Editor起動 18
Editor Operation 18
Function 18
補足 18
HEADER 19
Function 19
補足 19
Data (LIST) 20
Function 20
補足 20
Data (ASCII) 21
Function 21
補足 21
Data (Bin) 22
Function 22
補足 22
Data (Int) 23
Function 23
補足 23
Script *24*
Scriptの基本仕様 24
メインループの記述 24
SUB Rutine 25
Script Editor 26
Short Cut 26
入力支援部 Functions 26
Scriptの実行 *27*
Script Engine 27
Short Cut Icon 27
補足 27
Real Time Tree Viewer *28*
Tree Viewer 28
Short Cut Icon 28
Memo Editor 28
補足 28
Tree Log Viewer *29*
Log Viewerの起動 29
Operation 29
補足 29
Service Functions *30*
Queue Buffer Monitor 30
Send Queue Reset 30
Sequence Reset 30
Send Data Viewer 30
Message Creator 31
Functions 31

補足 31
HSMS Monitor / SECS-I Level Monitor 32
SECS-I Level Monitor 32
HSMS Monitor 32
Event Viewer 33
Functions 33
Script Functions *34*
Function Spec. 34
オフセット概念 34
構文禁則 34
スクリプトの構文解析手順 35
ファイル異常 35
Script Control Function 36
INCOM_F指令 36
SAVE指令 36
READ指令 36
SEND指令 36
MEMO指令 36
DIM指令 36
Calc Function 37
INC ：変数に+1する 37
DEC ：変数に-1する 37
+ ：変数と定数を足し算する 37
- ：変数と定数を引き算する 37
* ：変数と定数を掛け算する 37
/ ：変数と定数を掛け算する 37
LET ：変数に数値（文字）を入れる 37
Timer Function 38
CREATE ：タイマーを生成する 38
KILL ：タイマーを消去する 38
START ：タイマーを起動する 38
CLOSE ：タイマーを停止する 38
INTERVAL ：タイマーのインターバル時間を設定する 38
FLAG_OFF ：タイマーのON Flagをリセットする 38
STRING Function 39
Re_Write Function 39
OFF_SET ：指定箇所の直接変更 39
R_DATA ：構造体のデータを受信データで置き換える 39
DIM ：構造体のデータを変数で置き換える 39
IF Function 40
Function_Name ：最近に受信したSFを比較する 40
Function ：最近に受信したStreamを比較する 40
Function ：最近に受信したFunctionを比較する 40
OFF_Set ：最近に受信したDataのOffSet値を比較する 40
OFF_SET_DIM ：最近に受信したDataのOffSet値と変数の内容を比較する 40
Dim ：指定した変数の値を比較する 40
Timer ：任意のタイマーがONしているか調べる 40
Show Function 41
Print ：MessageBoxを使った表示 41
Printf ：MessageBoxを使った変数の表示 41
Message ：Event Displayへのメッセージの書き出し 41
Value ：EventDisplayに変数名と変数値を表示 41
R_Data ：文字列とオフセットの内容表示 41
その他 *42*
特殊な命令 42

Introduction

About SECSeIIme

このProgramは開発当初Delphi 2.0jで作り、SECSeIIuからVar. 4.0jで作りました。
DelphiのVar. Upに伴い変数の仕様が変わったのでCastしてあります。
SECS規格は”Book Of SEMI STANDARDS 1993/1999 製造装置”を参考にしています。
Program評価にはAlpha1のSECSシミュレータVar. 3.30を使用しました。
ターゲットとしてElmicSystemのI/F Card V50 SECS手順を使用しました。
HSMSのターゲットとしてGW社のSECSIM-Proを使おうとしましたが. . . 頓挫しました。
貸して貰おうとしましたがLicense Keyが消えそうで怖かったのでやめました。
そんな訳でHSMSに関しては実機でDebugしています。
ターゲットはYOKOGAWA EQBrain Var. 4.20です。
その後暫く触る機会が有りませんでしたがDelphi2009をVistaにインストールしたのを機会に再度アップデートしWin7で動作確認しました。 幾つか不具合が有ったのを修正しています。

SECSeIImeの目的

市販されているSECS Simulatorは基本的にホストシステムの代行が主体です。
従って、プログラム制作の初期段階ではSimulatorの機能が痣となってDebugが捗らない等の弊害が有ります。
（たとえば、Time Out監視は初期段階では不要ですし、送信したデータはたとえ構造が出来ていなくても全て見える必要が有ります。）
プログラム開発する側にとって、開発段階に応じてSimulatorに要求する機能は異なって来ます
SECSeIImeはその様に、Programを開発する状況に応じて適切な環境を提供出来る構造に主眼をおいてデザインしました、環境はSECSeIImeの設定内容で大きく変化します。

SECSeIImeはSECS/HSMSのHost Simulator機能の内、
１：装置(Target machine)が必要とする上位通信においての期待する回答をSimulateする。
２：SECS/HSMSの通信機能をDebugする際のTargetとなる。
３：SECS/HSMS Protocol においてイレギュラーな通信をSimulateする。
上記を実現する為の手段として、
１：Script機能によりtargetの通常の通信をSimulateする。
２：ProtocolでSupportされるTimer機能をEnable/Disableに出来る。
３：Protocol Header Testを備え任意のHeaderを生成出来る。
４：SECS-I levelのHandling Testerを備え任意のCharacterを送信出来る。
を備えた装置組込型のSECS Protocolや装置Ladder programを開発するためのToolです。

免責

作者はいかなる場合においても、本プログラムを使用したことによって生じた
いかなる損害に対しても保証する義務を負わないものとさせていただきます。

本書について

SECS/HSMSについての基本的な内容を理解されている事を前提としています。
SECSに関する細かい制約は記述していません。
当方も勉強不足から間違った解釈を行っていることもあり得ます。
その様な内容が有りましたらご連絡下さい。

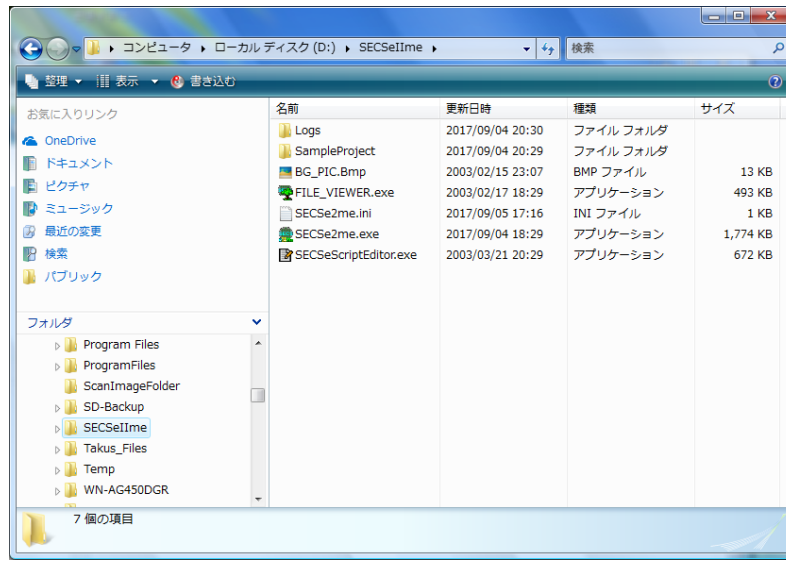
使用している語句について不明な点は別途SECSの仕様書を参照して下さい。

Install

Directly

SECSeIImeは単体で動作します。
基本的には何処に有っても起動することが可能ですが、任意にフォルダを作り使用するファイルを配下に置きます。
SECSeIImeを起動するとSECSeIIme.iniファイルを読み込みますが無い場合は自動生成します。
最初に起動した時に “Propertys”を使ってSECSeIImeの関係するフォルダやLOGファイル用のフォルダを設定してSECSeIImeを終了させるとSECSeIIme.iniが書き換わり再度起動する時に読み込みます。
　圧縮ファイルに有るSECSeIIme.iniには”C:¥SECSeIIme”以下にSECSeIIme.exe以下必要なファイルとディレクトリが存在することを前提としています。
削除して新たにSECSeIIme.iniを作るかPropertys→Directoryで設定を変更して下さい。
SECSeIIme.ini ファイルはテキストエディターでも修正変更できます。

（D:直下に”SECSeIIme”フォルダを追加してファイルを置いた例）



SECSeIImeにはInstallerは有りません。
任意のフォルダに展開して貰えれば問題なく動作可能です。

補足

１：任意のフォルダを作って展開する場合はSECSeIIme.iniを削除して起動してください。
　この場合最初にフォルダ設定を行って一旦終了し、再び起動したのち使用可能となります。
　→　P11参照

Program

Copyright

Program Environment

初期の開発はWindows95 OSR1で作っています。
SECSeIIuでWindows98/NTを使用しました。
現在のVerはWindows7で動作確認しました。

確認されている環境

Environment	Status	NOTE
Windows XP Home E.	OK	
Windows XP Pro.	OK	
Windows Vista	OK	32bit Application
Windows 7	OK	32bit Application

問い合わせ

問い合わせは電子メイルでお願いします。
enquiry@polaris-wks.mail-box.ne.jp
スパムメイルやウイルス予防のため、Keywordが無い場合読む前に削除しています。

SECSeIImeの構造

SECS-I

SECS-Iでは１対１の通信をRS-232Cで行います。

COM-Portが２つ有るPCを用いればSECSeIImeを２つ起動し、Port間をSerial Cableで繋いで通信する事が出来ます。
但し、同じExeを２重に起動すると、Fileの保護異常が発生するので、それぞれ異なるDirectoryから起動する必要が有ります。

SECSeIImeはMaster/SlaveどちらのSimulationも可能です。

同時にENQが発生した場合、Master側に設定された方に優先権が有ります。
SECSeIImeでは送信するMessageをQueueに蓄えてからInterval Clockを用いて送信しますので、SECS-Iでも連続送信による問題が発生しにくい構造となっています。
このInterval Clock Timingは“Setting”の項で設定します。

HSMS

HSMS通信はWinSockを用いています。

WinSockを用いてTCP/IP Protocolを行っています。
SECSeIImeをHSMS Modeで立ち上げると、自動的にWinsockをSaver Modeで起動し、Connectされるのを待っています。
同時期に複数のConnectが起こっても、その都度異なるWinSockを起動しますので、受信に関しては複数のIP Addressに対応します。
この時の返信は送り元に返されますが、Script等で意識的に送るMessageは最初に指定したIP Addressにのみ送ることが出来ます。

自分のIP Address

自分のIP AddressはWindowsの“マイコンピュータ”－“コントロールパネル”からTCP/IP Protocolの設定に設定されたIP Addressとなります。
もし、DCHPを使用している場合はNetworkにLoginする度に変わってしまう事になります。
また、Port番号は単一で使用していますので、相手方として設定したPort番号になります。
設定方法は“Setting”の項を参照して下さい。→ P12参照

Task

SECSeIImeはWindows95で開発された関係でMulti taskを用いていません。
COMの監視やWinsockにThreadを用いていますがそれ以外の部分では単にWin Messageを透過してそれらしく見せています。
従って、高負荷に成ったときにはTree-Viewer等の多くのリソースを必要とする処理は使用しないよう配慮して下さい。

補足

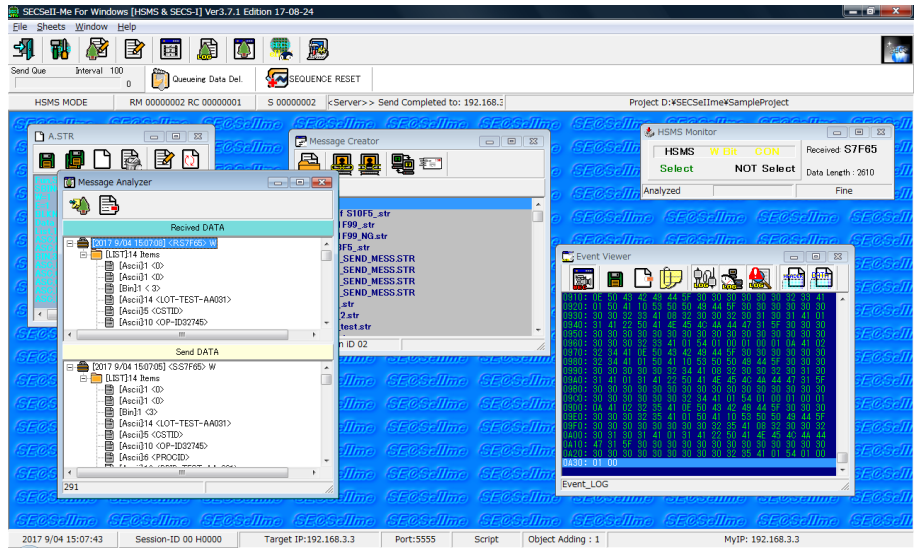
全ての送信MessageはLogicが生成後例外なく、一旦Send Queueに蓄えて送信する構造のため、場合によって会話が入れ子になる事が有ります。
つまり、Slave側が一次Messageを送る為の手続きを開始した時に、Master側が同時期にENQを送ると、Slave側は送信手続きを中止します、Logicでは１次Messageを受けたので２次Messageを作成しますが、Queueの中では先に送ろうとした１次Messageが残って居るのでそれを先に送信し、更に続けて２次Messageを送ります。
SECSeIImeではどちらが先に送られても問題有りませんが、受ける側で同様に成っているとは限らないので注意が必要です。

Start Up / Program End

起動

SECSeIImeの起動はSECSeIImeのIconをダブルクリックして下さい。

SECSeIIme.exeのみで起動します。



起動した状態では各Windowが重なっていますので、適当に配置し直して下さい。

終了

画面の”X”では終了しません、このIconをクリックする事で終了します。

補足

SECSeIImeは必要に応じてObjectを生成し、いくつかのObjectは更にObjectを生成します。
子供(Object)より先に親(MDI)が終了すると子供が迷子になります。
本来は親の終了時に子供も終了するはずですが、なぜか拒む事があり、終了出来ない事が有りました。一応、親が終了する際に子供の状況を確認するように工夫したつもりですが、
時々見失う事が有ることと、孫が生成されたときなどは綺麗に消えないことが有ります。
そのため、いくつかの制約が有ります。
１：HSMSはDisconnectしてください。➔ [SEQUENCE RESET]をクリックして初期化して下さい。
２：Tree Viewerを先に終了してください。
３：Scriptは停止してください。

Begging

起動の前に

まず始めにProjectを作る必要が有ります。

SECSeIImeではProjectは一つのDirectoryを指します。
つまりUniqueなDirectoryを作りその中に構造を宣言したStructure Fileや応答用の
Script Fileを作り管理します。
必要に応じてその下にLog Fileを作ります。（必ずしも下とは限りませんが）
　既存のProjectを流用するような場合はDirectory単位でCopyして後から内部のFileを修正するやり方を意図しています。
SECSeIImeにDirectoryをCopyする機能は有りませんのでExplorerなどを利用して下さい。
Project はどこに作ってもかまいませんが一応SECSeIImeのDirectoryの下に作って下さい。
　参考：
SECSeIImeに付属している”Sample” Directoryを適当なDirectoryにCopyしてDirectoryの名称をTarget Project名へ変更して下さい。
そのあと、Project内部にLogを保存する為のDirectoryを作ります。
StructureのFileを調べて必要な者を残し不要なFileを消去します。
➔ よく使われるS1F1/2とかS2F25/26とか
SECSeIImeのStructure EditorでStructure Fileを修正し、Project内のStructure Fileを作って下さい。

ProjectのFile

拡張子	用途	備考
.str/STR	構造体ファイル SECSの構造体を記述	Text File ➔ Structure Editor
.ssc/SSC	Script File 自動実行ファイル	Text File ➔ Script Editor

補足

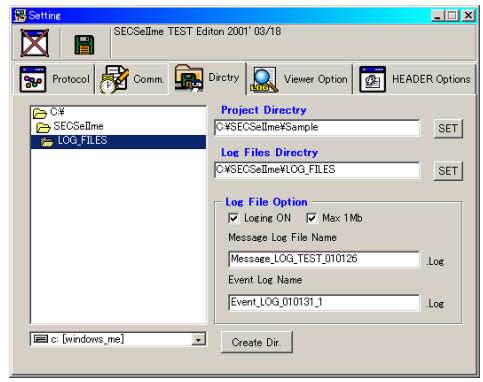
Directoryを正しく設定しないまま何らかの通信を行おうとすると、書き込み先が見つからない等の異常が発生します。
古い環境ではSECSeIImeが落ちることも有ります。
必ず、先にFileの設定を行って下さい。P11参照 ➔　重要！

Setting Section

Directory

SettingのTab[DIRECTORY]を選択してProjectを選んで[SET]でPathを登録します。
ここでの手続きを行わないまま通信等の処理を行うと、Log Fileを生成する段階で異常終了する事が有ります。（どんな内容のTESTでもこの手続きを行ってから開始して下さい。）

➔ 起動はこのIconをクリックするか、FILE – Setting で起動します。



Function	意味	NOTE
Project Directory	Project DirectoryのPathを設定します。	内部にはStructure FileとScript Fileを作って置きます。 後のScriptやSend dataのSectionで参照されます。
Log File Directory	通信LogとEvent logの保存先です。	リムーバルドライブに設定する事も可能ですが、速度が著しく低下しますのでHD上に設定して下さい
Logging ON	このSelectがONの時Loggingします。	
Max 1Mb	Log Fileの最大サイズを1Mbで固定します。	Log Fileが1Mbを越えるとそのFileは拡張子を.OLDに換えて新たにLog Fileを生成します。
Message Log File Name	通信Log Fileの名称を設定します。	拡張子は不要です。
Event Log Name	Event Log Fileの名称を設定します。	拡張子は不要です。
Create Dir	新しくDirectoryを作ります。	選択されたDirの下に生成します。

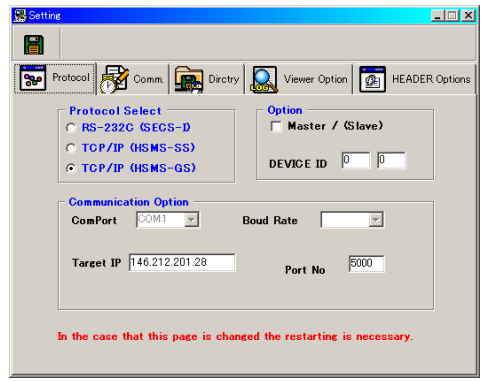
Operation

Projectの設定とLog File Directoryの設定

１：Directory TreeからProjectのPathを選択する。
２：Project Directoryの[SET] Buttonを押す。➔ DirectoryのPathが表示される。
３：Directory TreeからLog Fileを保存するDirectoryを選択する。
４：Log Files Directoryの[SET]を押す。 ➔ Log Files DirectoryのPathが表示される。
５：FDのIconをクリックしてSettingを終了する。
　➔ このときこの操作が再起動後有効になる旨メッセージが出ます。
６：SECSeIImeを再起動して下さい。
もし他の設定が必要なら項５の前に他の設定を行って下さい。

Protocol

SECSeIImeはSECS-I(RS-232C)とHSMS-SS(TCP/IP)の２種のProtocolをサポートします。
この変更は再起動後有効になります。



SECS-I (RS-232C)

Protocol SelectでRS-232Cを選択し、
Comport（通信ポート番号）とBound Rate（通信速度）を設定します。
その他のパラメータはSECS-Iで規定されたパラメータに準拠しています。

項目	内容	備考
通信方式	半２重	
同期方式	スタートビット　1Bit	
データレングス	8Bits	
ストップビット	1Bit	
パリティ	無し	

HSMS-SS (TCP/IP)

通信する相手のIP AddressとPort番号を設定します。入力に関して特に範囲を設けていませんので、あらゆる入力が有っても入力異常には成りませんが再起動後に通信を開始する際に異常と成ります。Target IPはコンピュータ名による参照も可能です。
1.HSMS-SS シングルセッション（相手の一対の通信）：相手はActive Mode
2.HSMS-GS ゼネラルセッション（サーバーの様に相手を特定しない）：相手はPassive Mode
SECSeIImeではGSは待ち受け（Active Mode)の使用で、相手がPassive Modeの時この設定にして相手のConnect信号を待ちます。

Master/(slave)

SECS-Iの時MasterかSlaveのどちらかの挙動をとります。
これは通信の衝突時にどちらを優先するか決定します。

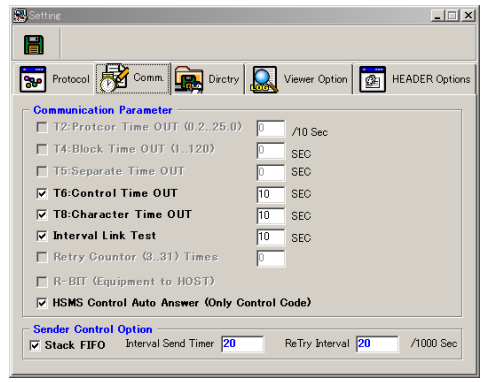
設定	同時に通信が発生した際	NOTE
Master	強行的に送信します。	Multi Functionの時は送信しません。
Slave	受信の為に送信を中断します。	

補足

マルチファンクションを受信中はその通信が全項目受信終了するまで送信を行いません。
但しマルチファンクションはHSMSではサポートしていません。
長い受信が有った場合で受信中に何らかの送信イベントが発生した場合、
本来二次メッセージを送るべき所をイベントによる一次メッセージが先に送信され、
続けて先に受信したメッセージの二次メッセージを送信します。
マルチファンクションの受信中に異常により中断した場合受信したメッセージは破棄され
受信履歴にも残りません。

Communication Parameter

タイムアウトやR-Bit等の設定を行います。
各設定項目の前にあるCheckをON/OFFする事で監視を有効／無効に切り替えます。



Setting Parameter

Parameter	NOTE	USED
T2: Protocol Time Out		SECS-I
T4: Block Time Out		SECS-I
T5: Separate Time	NO Supports	現在サポートされていません。
T6: Control Time OUT	コントロールコード応答時間規制	
T8: Character Time Out	補足参照	
Interval Link Test	一定時間で自動的にLink Testを行う。	HSMS
HSMS Control Auto Answer	コントロールコードの自動応答	HSMS
R-Bit	Message方向、ON: to HOST	SECS-I
Retry Counter	Retry回数設定	SECS-I

補足

１：HSMSにおいてSECSeIImeはタイムアウトが発生しても切り離しは行いません。
２：T8 Time Outは本来キャラクタ間のタイムアウトを監視しますが、
　SECSeIImeでは構造上の理由によりメッセージ単位の受信時間を監視します。
３：T8はPassiveの為必要有りませんが将来Activeが追加される可能性が有ります。

Sender Control Option

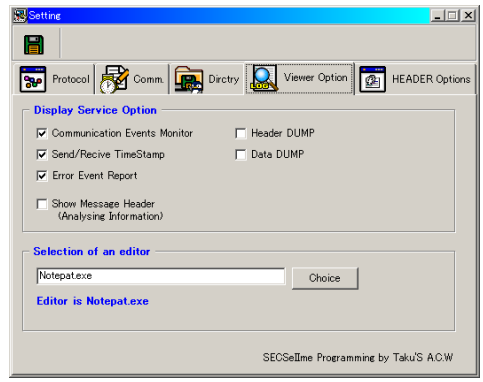
Parameter	Significance	NOTE
Interval Send Timer	SECSeIImeは内部でInterval Timerを用いて送信タイミングを計っています。ここでは送信Queueから送信に送られるタイミングを設定します。	同時に作成された２つ以上のメッセージを送信する場合でも受信側に負担を掛けないようにDelayする様に成っています。
Retry Interval	失敗したMessageを再送する場合のIntervalを設定します。	Interval Send Timerの値をここの値で上書きします。
Stack FIFO	送信Queueの設定です。 ON: FIFO / OFF: FILO	入れ子を作って正常に応答できるか実験する場合に使用します

補足

１：Messageの生成から送信するTimingはScriptでも設定する事が出来ます。

Viewer Option

Event LoggingのLog Levelの設定、Sender(Send Queue)の設定をします。



Display Service Option

Parameter	Significance	NOTE
Communication Event Monitor	SECS-Iではシェイクハンドの様子を時間で表示。 HSMSではControl Codeを時間で表示します。	Protocol LevelでのDebug用途
Send/Receive Time Stamp	Stream Functionの送受信を時間で表示します。	補足参照
Error Event Report	通信異常の発生を報告します。 リトライの様子を表示します。	
Header Dump	送受信HeaderをDumpします。	Header 10 Bytes
Data Dump	送受信DataをDumpします。	

補足

１：有効でない項目はloggingされません。（表示のみ無効に成るわけではない）
２：Event Logの生成先をFD等の遅いデバイスに設定されていると全体的なレスポンスが著しく遅くなります。（基本的にはHDに生成して後にFD等に書き出すことを前提としている）
３：Targetの完成Levelに応じてLoggingされる内容が変わることが前提となっている。
初期はProtocol LevelのLogが必要で後に通信Logが重要になると考えている。
４：Display Service Option はEvent ViewerのIconでも変更出来ます。

Select of Editor

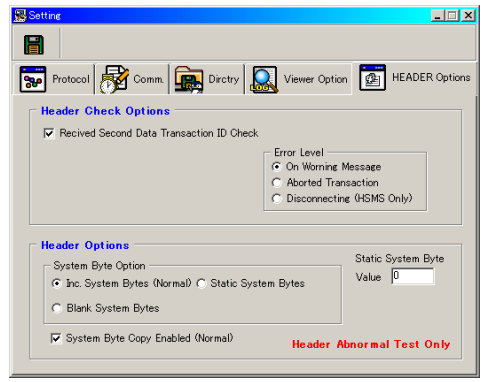
任意のEditorのPathを設定します。
基本的にはSECSeIIme付属のSECSeScriptEditorを使用する事が可能です。
設定後”Sending Data”Windowと”Script”Windowの[Editor]Buttonを押すと起動します。

補足

１：Windowから起動するEditorへはファイル名を自動的にわたします。

Header Option

特にHSMSのControl Headerに関する設定を行います。
TargetのHeader Systemに関するDebugの為にSECSeIImeをカスタマイズします。



Header Check Option

Function	Significance	NOTE
Show Message Header	受信したHeaderのID情報をDumpします。	将来はこの機能を拡張します。
Received Second Data Transaction ID Check	受信した２次メッセージのID情報をチェックして次のError Levelに基づいた動作を行います。	
Error Level	１：警告のみ ２：Transactionの無効 ３：Disconnecting	SEMIでは状態に相当する状態移行が定義されていますが､DEBUG環境では必ずしも必要では無いためこの様なOptionが有ります。

補足

１：現在のFunctionは次のVar.で改正されることが予想されます。（実験的に行っている状況）

Header Option

Function	Significance	NOTE
System Byte Option	１：一次を送信する毎にInc(+1)される ２：Blankで送る ３：固定値を与える	通信仕様に合わせる。
System Byte Copy Enabled	受信したHeaderのSystem ByteをCopyする。	異常なDeaderを生成して応答に関するDebugに使用します。

補足

１：TargetのHeader異常監視機能を評価するための特殊なOptionです。

Protocol Tester

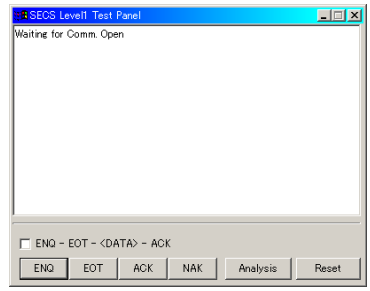
Protocol Simulator / Header Testerの起動

SECSeIImeがSECS-I Modeで起動しているときは、SECS-I Level Protocol Simulator、
SHSMS Modeで起動しているときはHSMS Header Testerが起動します。

← 起動はこのIconをクリックするかFile Menuから起動します。

SECS-I Level Protocol Simulator

Settingで指定のPortから任意のControl Characterを送信する事が出来ます。
各ボタンのクリックでENQ,EOT,等のControl Characterを手動で送信します。
TargetのProtocol Level Testが目的です。
画面には送受信した時のControl Characterとタイムスタンプが表示されます。



Function

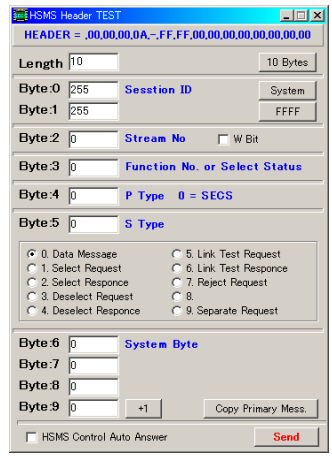
Function	Significance	NOTE
ENQ	ENQ Character 送信 Button	Buttonを押す毎にENQを送信します。 ENQ = 05h（送信要求）
EOT	EOT Character 送信 Button	Buttonを押す毎にEOTを送信します。 EOT = 04h（受信可能応答）
ACK	ACK Character 送信 Button	Buttonを押す毎にACKを送信します。 ACK = 06h（了解応答）
NAK	NAK Character 送信 Button	Buttonを押す毎にNAKを送信します。 NAK = 15h（不解応答）
ENQ - EOT - <DATA> - ACK	受信Sequenceの自動	ENQから始まる一連のハンドリングを自動で行います。
Analysis	受信データの解析	受診したデータを解析します。
Reset	受信Sequenceのハンドリングリセット	ENQから始まる一連のSequenceが途中で終了できなかった場合にResetします。 ENQがくるのを待機します。

補足

１：このFunctionを使用する前にComm Portを開ける必要が有ります。
　メインの上部にある[Comm.Port Open]をclickして下さい。
２：HSMS Modeの時はこのFunctionは起動しません。
３：SECS-I LevelでのProtocol Testを目的として作られていますので、連続するメッセージに対応できない事があります。

HSMS Header Tester

Headerに関するTEST環境の為のFunctionで、任意のHSMS Headerを作り、送ることができます。



Function

Headerの構造を縦に展開した形にデザインしてあります。
各項目は手入力と定数の直接選択入力が出来ます。

Function	Significance	NOTE
Length	Headerの長さ	4ByteのBig Indian型になっています。
Byte:0 **Byte:1**	Session ID	Control Codeの時は"FFFF"を設定します。
Byte:2	Stream Number	W BitのCheckが有ります。 ONで80Hが加算されます。
Byte:3	Function No or Select Status	Dataの場合はFunction番号を、Control Codeの場合はSelect Statusを入力します。
Byte:4	P Type	SECSの場合は"0"の固定です。 このコードはSEMIの規格でも将来の拡張用です
Byte:5	S Type	表中のコード番号を参照してください。
Byte:6..9	System Bytes	[+1]Keyで加算します。 [Same 1St]で一次メッセージのSystem Byte をCopyします。
HSMS Control Auto Answer	Auto Answer	受信したControl Codeに自動で応答します。 このFunctionを起動したときはOFFに成っています。
Send	送信ボタン	作成したHeaderを送信します。

補足

1：任意のHeaderを作って送信できるようにデザインしてあります。
　内容に制限はありませんので全部Nullでも送信する事が出来ます。
2：Setting にあるHSMS Control Auto Answerの内容はこのFunction起動時に記憶して、
　終了時に元に戻しています。
3：[+1]はByte9のみ加算します。

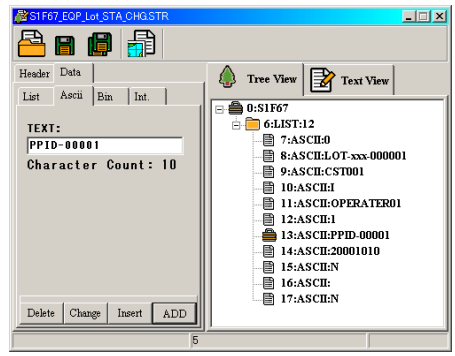
Stream Function Editor Section

Stream Function Editor起動

← このButtonのクリックか、File MenuよりStream Function Editorを選択します。

File選択のDialogがOpenします。Dialogで任意のFileを選択します。
（Stream FunctionはSECSeIImeでStructure Fileと呼ばれ、このFileの拡張子はSTRです。）

Editor Operation



Function

Editorは右半分が入力支援部、左半分が構造表示になっています。
(1)　(2)　(3)　(4)

Speed ButtonとFunction

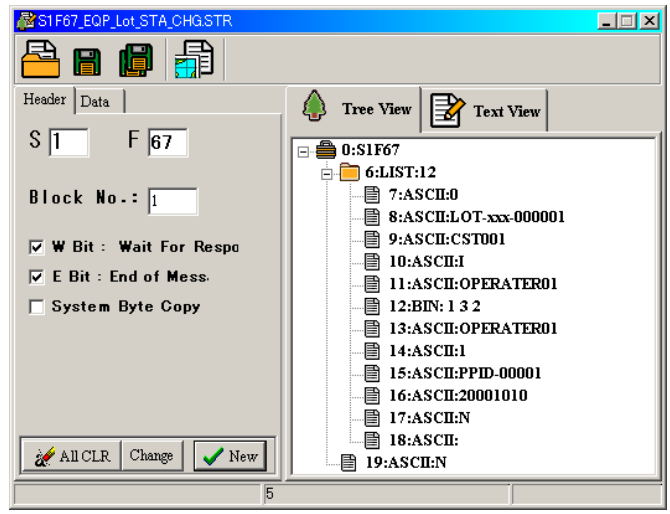
Function	意味	NOTE
1.新規FileのOpen	修正中のFileを放棄して新しくFileをOpenします。	保存されない場合既存のデータは破棄されます。
2.上書き保存	編集中のFileをOver Writeします。	編集中に保存することが出来ます。
3.名前を変えて保存	別の名前で保存します。	File名が変更されます。
4.Remappimg	編集中のFileをTree構造に再編成します	通常必要有りません
ADD	最後に追加します。	
Insert	前に挿入します。	
Change	選択された物と交換します。	
Delete	選択された物を削除します。	

１：編集中のFileはTree構造とText Modeで見ることが出来ます。
２：Tree構造の表示状態で任意のItemをクリックすると内容が支援部に送られ、表示します。
３：上の例では11番目のItemを選択し、支援部のAsciiに”OPERATOR01”が表示されています。

補足

１：作成されるDataはSECS-IとHSMSの区別が有りません。
SECS-Iは一回で送れる電文の長さが128文字までですので設計時に適当な長さで
分割して下さい。（自動的には分割されません。）

HEADER



Function

Tree ViewのHeader部をクリックするとその内容が支援部のHEADRに反映され表示されます。

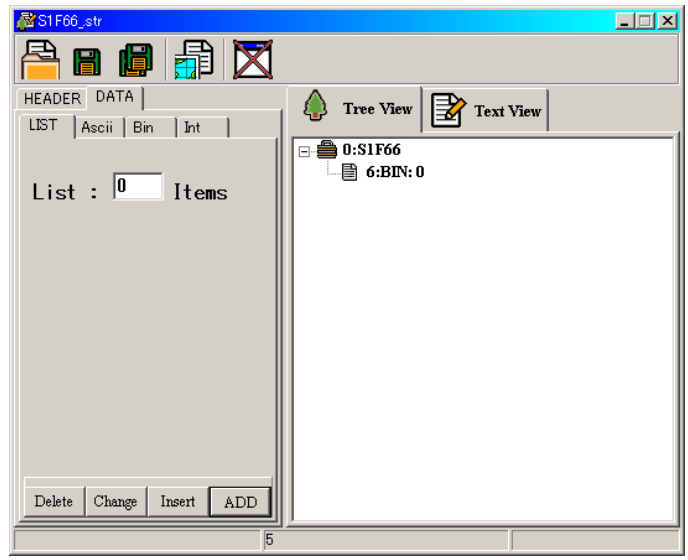
Function	意味	NOTE
S	Stream番号	
F	Function番号	
Block No.	ブロック番号	マルチファンクションの時の番号（HSMSの時は無効です）
W Bit	W-Bitの有無	
E Bit	E-Bitの有無	
System Byte Copy	２次Message場合一次MessageのSystem ByteをCopyする	
All Clear	全てのDataをClearする。	
Change	支援部のDataと入れ替える。	
New	Headerを残しData部をClearする。	

補足

System Byte Copyは明示する必要が有ります。

➔ 一次Messageを受けていない状況でこのCheckはNullを当てはめます。

Data (LIST)



Function

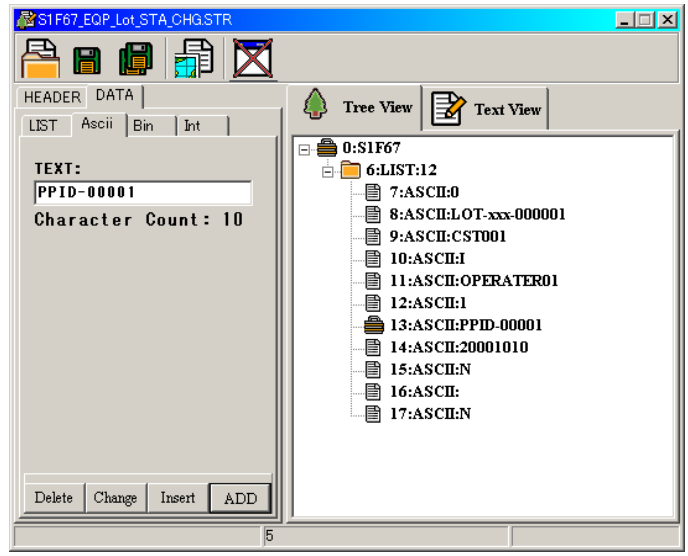
List文を生成します。

Function	意味	NOTE
Items	Item数設定	数値
Delete	List文を消去します。	
Change	List文に入れ替えます。	
Insert	List文を挿入します。	
ADD	List文を追加します。	

補足

List数を変更後、Remappingを行うことでTree構造の変更が行われます。

Data (ASCII)



Function

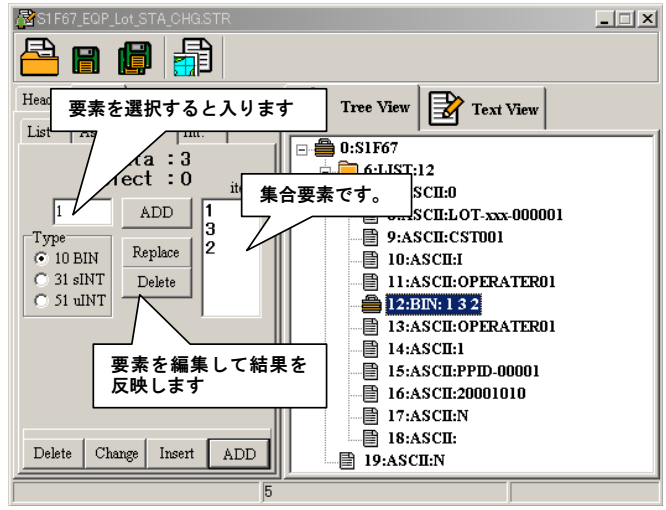
任意の長さのText文を生成します。

Function	意味	NOTE
Items	Text設定	文字列（漢字可能）
Character Count	Textの文字列の現在数を表示します。	
Delete	ASCII文を消去します。	
Change	ASCII文に入れ替えます。	
Insert	ASCII文を挿入します。	
ADD	ASCII文を追加します。	

補足

１：文字列の長さは２５６文字まで可能です。
但し、SECS-Iでは一回の送信電文の長さが１２８文字までです。（自主規制して下さい）
２：Textには漢字文字列が可能ですが、SECSではASCII文字列ということに成っています。
３：空白文字列も入力可能です、Character Countで文字列の長さに注意して下さい。

Data (Bin)



Function

任意の長さのBin(Binary)文を生成します。

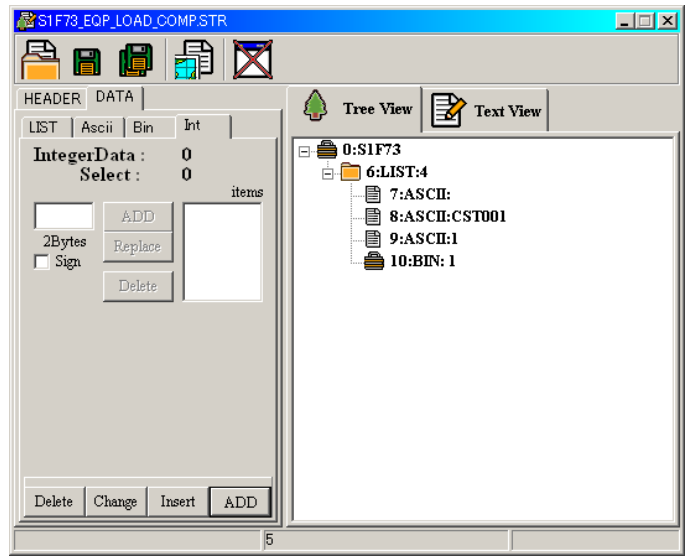
Function	意味	NOTE
Data	入力値（入力Box） Selectされた数値	数値(8Bits)
Select	複数の数値が有る場合に、 変更したい数値の位置を指定します。	
Type	8Bits数値の形式を指定します。 10:Bin 31:sINT 51:uINT	 Bin ➔ Binary(8Bits) sINT ➔ Signed Integer uINT ➔ Unsigned Integer
ADD Button	入力値を追加します。	
Replace Button	入力値と交換します。	
Delete Button	Selectされた値を消去します。	
Delete	BIN文を消去します。	
Change	BIN文に入れ替えます。	
Insert	BIN文を挿入します。	
ADD	BIN文を追加します。	

上記の例では、１２：BIN:１　２　３
が選択されています。
この意味は、バイナリー配列で”1”,”3”,”2”がその要素です。
この様に集合を扱う事が出来ます。
要素の編集は個々の要素を選択し、変更後コマンドボタン[ADD]/[Replace]/[Delete]で編集を反映します。
最後に画面下の[Delete]/[Change]/[Insert]/[ADD] ButtonのItem単位で変更を反映します。

補足

１：複数のItemを入力する事が出来ます。
➔ Item数に制限は有りません（自主規制して下さい）
２：8Bitsの数値ですが実は制限されていません（変換時に強制的に上位を削除しています）
３：入れ替え等のFunctionを行う前にSelectして対象を明示して下さい。

Data (Int)



Function

任意の長さのInt(Integer)文を生成します。

Function	意味	NOTE
Data	入力値（入力Box） Selectされた数値	数値(16Bits)
Select	複数の数値が有る場合に、 変更したい数値の位置を指定します。	
Sign	符号の有無を指定します。	符号付き ➔ Signed Integer
ADD Button	入力値を追加します。	
Replace Button	入力値と交換します。	
Delete Button	Selectされた値を消去します。	
Delete	INT文を消去します。	
Change	INT文に入れ替えます。	
Insert	INT文を挿入します。	
ADD	INT文を追加します。	

補足

１：複数のItemを入力する事が出来ます。
➔ Item数に制限は有りません（自主規制して下さい）
２：16Bitsの数値ですが実は制限されていません（変換時に強制的に上位を削除しています）
３：入れ替え等のFunctionを行う前にSelectして対象を明示して下さい。

Script

Scriptの基本仕様

メインループの記述

SECSeIImeではイベント駆動型のScriptをイメージしてデザインしています。
メインではイベント発生後関連する処理を行う分岐点としてメインを位置づけています。
先頭はProcedureです、Procedureはサブルーチンからの帰り命令Retの戻り位置です。
イベント待機文を記述して発生を待ち、イベントが発生すると次の行を実行します。
普通If文を記述しイベントを解析し、条件で流れを変えます（サブルーチンへ）
終端にENDを記述します、ENDはRetと同様にProcedureへ流れを返します。

記述例：

```
 1 // Program init.
 2 Show,Message,Start Try Normal Case,1
 3 Dim,%PassCount,0
 4 Dim,%CST1,1
 5 Dim,%CST2,1
 6 Dim,%P1_LOTID,Nothing
 7 Dim,%P1_CSTID,Nothing
 8 Dim,%P2_LOTID,Nothing
 9 Dim,%P2_CSTID,Nothing
10 Dim,%Temp,0
11 Dim,%Temp1,0
12 Dim,%Temp11,0
13 Dim,%Temp2,0
14 Dim,%Temp21,0
15 Dim,%ACT_Port,0
16 Dim,%MODE,0
17 // MODE=0 HOST Mode
18 // MODE=1 OPE. Mode
19 PROCEDURE
20 // Loop Sequence ///
21 // Event Waiting
22 WAIT_Event
23 // Case of
24 if,FUNCTION_NAME,S1F1,OnlineSequence
25 if,FUNCTION_NAME,S2F17,TIME_SET
26 if,FUNCTION_NAME,S1F97,MC_Stat
27 if,FUNCTION_NAME,S6F85,Load_Request
28 if,FUNCTION_NAME,S6F95,Change_EQP_Mode
29 if,FUNCTION_NAME,S7F87,CST_ID_Report
30 if,FUNCTION_NAME,S6F81,LOT_STATUS_Report
31 if,FUNCTION_NAME,S6F91,EQP_STATUS_Report
32 if,FUNCTION_NAME,S6F83,PROCESS_DATA
33 if,FUNCTION_NAME,S7F89,LOT_CANCEL
34 if,FUNCTION_NAME,S7F84,LOT_INFO_ACK
35 if,FUNCTION_NAME,S2F25,S2F26
36 if,FUNCTION_NAME,S5F1,Alarm
37 END
```

この行以降にサブルーチンが記述される。

上記の例では１行目から使用する変数の宣言と初期化を行い、
１９行目にPROCEDUREの記述が有りますが、ここからメインのループになります。
２２行目のWAIT_Eventは受信やタイマータイムアップのイベントが発生するまで待機します。
２４行目から発生したイベントが何かによって処理先を切り替えています。

SUB Routine

サブルーチンの例

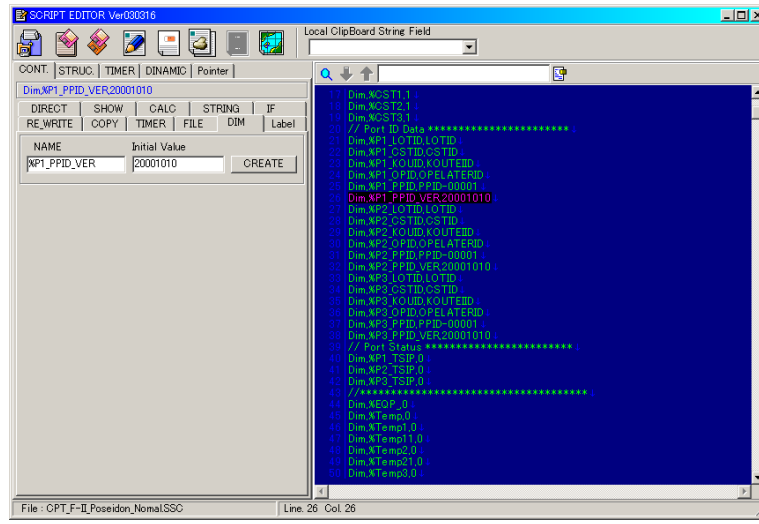
```
51 // Sub RTN
52 // ONline Sequence Procedure
53
54 @OnlineSequence
55 Send,S1F2_,10
56 Ret
57
58 @TIME_SET
59 Send,S2F18_,10
60 Ret
61
62 @MC_Stat
63 Send,S1F98_,10
64 Ret
65
66 @Load_Request
67 If,OFF_SET,4,1,PSD_1
68 If,OFF_SET,4,2,PSD_2
69 If,OFF_SET,4,3,PSD_3
70 Show,Message,<<ERROR>> S6F85 BAD Port No,2
71 Send,S6F86_,10
72 Ret
```

上記の例ではメインループ内の24 if,FUNCTION_NAME,S1F1,OnlineSequenceで
”受信したストリームファンクションがS1F1なら"OnlineSequence"に処理を移す”命令で、
54 @OnlineSequence に制御が移ります。
55 Send,S1F2_,10 でS1F2を送信し、(S1F2_.STRを読み込んで送信Queueに入れる)
56 Ret 命令でメインの19 PROCEDUREに制御が戻ります。
サブルーチン内に記述する内容は実際の処理に関する記述です。
もちろんIf文による制御も出来ます。
このほかに、GOTO指令による制御移動が有ります。
スタックポインタ方式では有りません、Ret文でProcedureへ戻ります。

Script Editor

SECSeIImeのScript文は高速化の為、構造チェックや中間コード化等の処理を行っていません。
従って、単純ですが読みやすさを犠牲にしています。
そこで、全てのFunctionをクリックと簡単な入力で生成出来るような支援機能を設けました。


← このIconをクリックすると（ScriptEditorが予め登録されていれば）起動します。
単体でも起動することが出来ます。



左側は入力支援部で右側がText Editorに成っています。

Short Cut

#	Short Cut Name	意味
1	Open	Script Fileを開きます。
2	Save	編集中のFileをSaveします（上書き保存）
3	Save as ..	編集中のFileに名前を付けて保存します。
4	TOOL Hide / Show	入力支援の表示／非表示を切り替えます。
5	ALL Clear	編集画面のデータを消去します。
6	Print	編集中のデータをPrintします。
7	ANDO	編集前の状態に戻します。（編集後有効に成ります）
8	Look UP Remapping	入力支援のデータを更新します。
9	Local Clipboard String Field	生成したInstructionがここに書き込まれ同時にClipboardにも書き込まれます。(EditorへはPastで書き込めます)

入力支援部 Functions

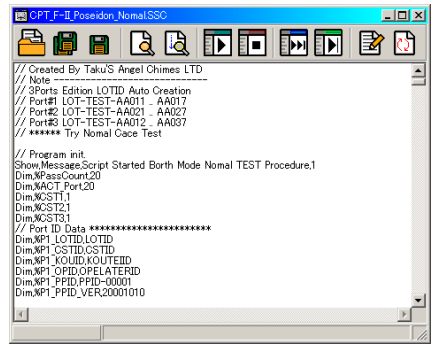
Function	意味
CONT.	Instruction Codeを生成します。
STRUC.	Projectに登録されたStructureの一覧表を表示します。
TIMER	Script内で宣言されたTimerの一覧表を表示します。
DYNAMIC	Script内で宣言された変数の一覧表を表示します。
Pointer	Script内で宣言されたPointerの一覧表を表示します。

Scriptの実行

Script Engine

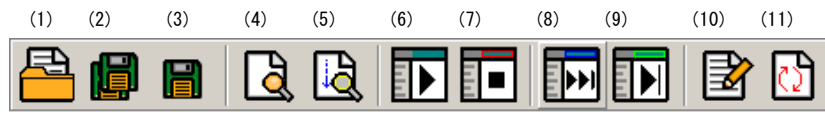
Scriptを使って自動返信等の処理を行う事が出来ます。

← このIconをクリックするとScript Consoleが起動します。



起動直後はScript Fileが選択されていない状態ですので何も表示しません。
（上はScript Fileを指定した後の状態です。）
OpenからFile Dialogを開き、Scriptを読み込みます。
読み込み後の状態及び実行中も内容を編集する事が出来ます。
Scriptは単純に順次行単位で命令を読み込んで解釈し実行する構造で
もし、理解できない指令が有っても単に無視するだけです。
従って実行中に何らかの変更を加えても即有効と成りますが
変数の名称変更など実行直後に変更すると既に解釈実行後で有れば効果が有りません。
実行中でも変更する内容に規制は有りませんので構造を理解の上行って下さい。

Short Cut Icon



#	Function Name	意味
1	Open	Script Fileを開きます。 この操作を行った後、他のボタンが有効になります。
2	Save as ...	Consoleに呼び出されたDataに名前を付けて保存します。
3	Save	Dataを上書き保存します。
4	String Find	文字列の検索
5	Find Next	文字列の検索次の行以降
6	RUN	実行
7	STOP	停止
8	DEBUG MODE Flag	処理ステップをEvent Viewerへ書き出します。
9	STEP RUN	１行実行毎に確認Dialogが出ます。
10	Editor Stat	ScriptをEditorで起動します。（予めEditorを登録）
11	Script Reload	Scriptを読み直します。

補足

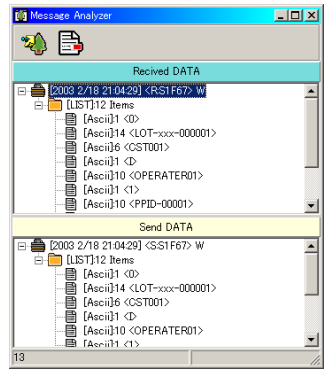
SECSeIImeではScriptを使ってT3 Time Outを実現しようとしています。
ですから、ParameterにはT3に関する設定が有りません。
また、Script以外の方法で２次Messageを自動的に返信する機能は有りません。

Real Time Tree Viewer

Tree Viewer

Real Timeに受信Messageを解析してTree構造のViewerに表示します。
Messageは受信、送信のそれぞれ別のViewerに表示します。

← このIconをクリックする事で起動します。
２重起動を防止しています。

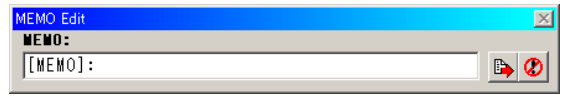


Short Cut Icon

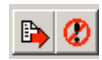
(1)　(2)

#	Function Name	意味
1	MEMO	MEMO Editorを開きます。 Tree構造体にMemoを挿入する事が出来ます。
2	Clear All	画面を消去します。

Memo Editor



Tree構造にMemoを挿入し、通信logにも残します。
通信中のLogにMemoをいれることが出来ます。



挿入とキャンセルボタン

補足

このViewerはCPU負荷が多きいので、Powerの少ないPCでScriptを実行中はEventを取り損なう事が有ります。

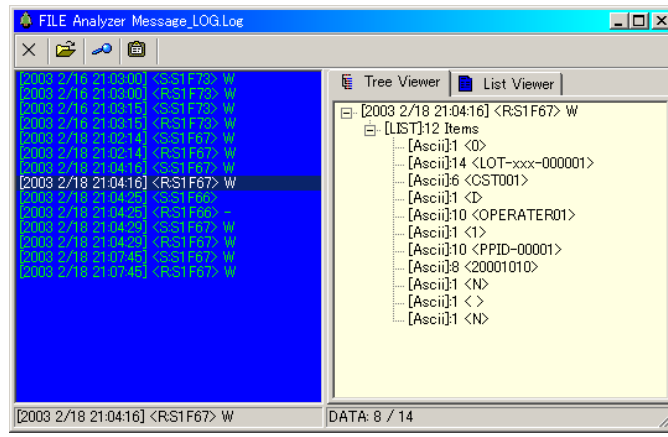
Tree Log Viewer

Log Viewerの起動

“Setting”でLogging OptionをONしてあれば送受信Dataは指定されたFileにSaveされます。
このDataはTree Log Viewerで見ることが出来ます。
このUtilityは独立したProgramですので他のPCにInstallしてLogを見ることが出来ます。

← SECSeIImeから起動する場合はこのIconをクリックします。

File_Viewer.exe を直接起動することも出来ます。
SECSeIImeから起動する場合でも終了は監視しません。



Operation

起動後File Open で解析したいLog Fileを選択します。
読み込まれたFileは時間毎にStream/Functionが表示されます。
MouseかKey BoardのCursor Keyで解析したいMessageを選択すると、Tree表示します。
List Viewerに切り替えるとText表示します。
(1) (2) (3) (4)

#	Function Name	意味
1	終了	終了します。
2	File Open	Log Fileを開くDialogが表示します。
3	Find	Find Dialogが開きます。
4	Copy to Clip Board	MessageをClipboardにCopyします。

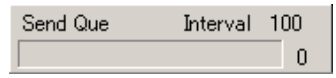
補足

File Viewerはいくつでも起動する事が出来ますが、リソースを大量に使用しますので
Powerの少ないPCでは同時に起動するProgramの数に注意して下さい。

Service Functions

SECSeIImeのService Functionの機能を説明します。

Queue Buffer Monitor

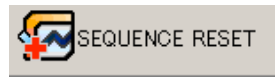


常時画面の左角に位置し、Queue Bufferの状態を常時Monitorしています、Queueには最大50 Messagesの蓄積が出来ます。（５０を越えると新しいMessageは消去されます。）
Intervalの項はQueueの先頭にあるMessageに設定されたInterval時間が表示されます。

Send Queue Reset

Queue Bufferに蓄えられているMessage Dataを消去します。
通信異常等で不本意に蓄えられたDataを消去したい場合に使用します。

Sequence Reset

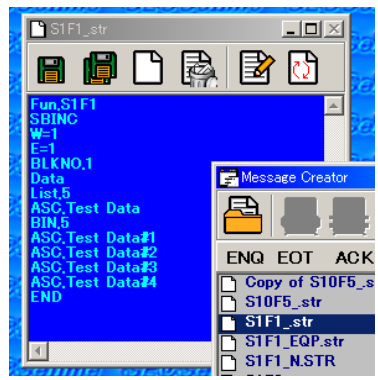


通信異常等で正常なHandlingが続行出来ない場合に初期化します。
SECS-Iの時とHSMSの時ではその効果が異なります。

MODE	効果	NOTE
SECS-I (RS-232C)	１：ENQ待ちにHandshakeを戻す。 ２：Retry Flagを初期化する。	
HSMS (TCP/IP)	２：Disconnectする。 ３：Winsockを初期化する。	TCP/IPがConnect状態では SECSeIImeを終了できません。

＊：SECSeIImeを終了する場合このFunctionを実行するように心がけて下さい。

Send Data Viewer

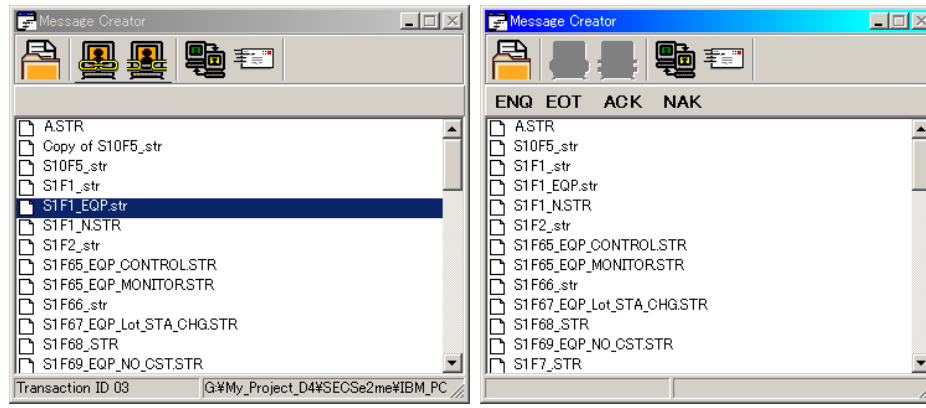


Message Creatorの任意のMessageをクリックするとそのDataを表示します。
ダブルクリックすると表示して送信します。
Sending Dataの内容はその場で変更することが出来ます。
簡単な返信Messageのパラメータを変更して送信する際に便利です。
但し、構造を理解しないで変更すると思いがけないトラブルが発生する事が予想できます。

Message Creator

手動でMessageを送信する場合に使用します、このFormは常時表示しています。

HSMSの時　　SECS-Iの時



Projectとして設定されたDirectoryの構造体宣言DataのFile Nameを表示します。

File Nameをダブルクリックするとそのメッセージを読み込んで送信します。

この時Send Message Viewerにも選択されたDataが表示しますので、必要に応じて修正することが出来ます。（Send Data Viewerの項参照）

Functions

(1)　(2)　(3)　(4)　(5)

#	Function Name	意味
1	File Open	Projectを開きます。（Directoryを選択します）
2	Connect	HSMSの時Connect要求を送信します。
3	Disconnect	HSMSの時Disconnect信号を送信します。
4	Loop Back Test / Link Test	SECS-I：S2F25を送信します。 HSMS：Link Test信号を送信します。
5	Resend Message	ダブルクリック／クリックにより選択された Messageを再送します。 Send Data Viewerで変更する場合このButtonで送信して下さい。

補足

１：Settingに正しいProject Directoryを設定していないとFormにFile Nameが現れません。

２：再送前にSend Data Viewerの内容に変更を加えると再送時に反映されます。
（再びクリックするとSend ViewerのDataは上書きされます。）

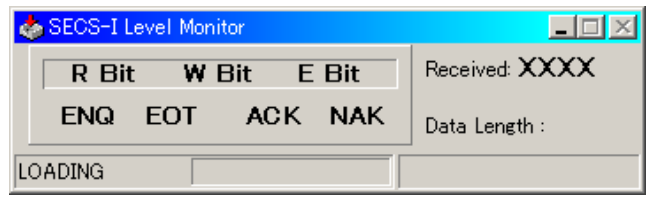
３：クリックから送信までの流れ
クリック ➔ Fileの読み込み ➔ Send Viewerへ書き込み ➔
Send Data Viewerの文字列から送信Dataの生成 ➔ QueueへDataが送られる ➔ 送信

４：再送の時の流れ
再送 ➔ Send Data Viewerの文字列から送信Dataの生成 ➔
QueueへDataが送られる ➔ 送信

HSMS Monitor / SECS-I Level Monitor

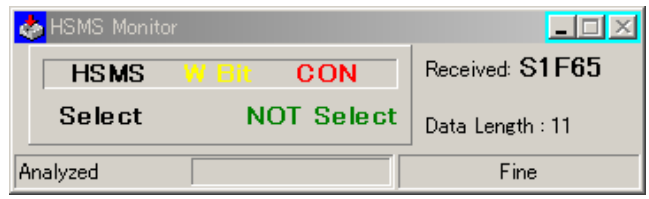
HSMS / SECS-IそれぞれのModeによって表示が異なります。
このFormは閉じることが出来ません。

SECS-I Level Monitor



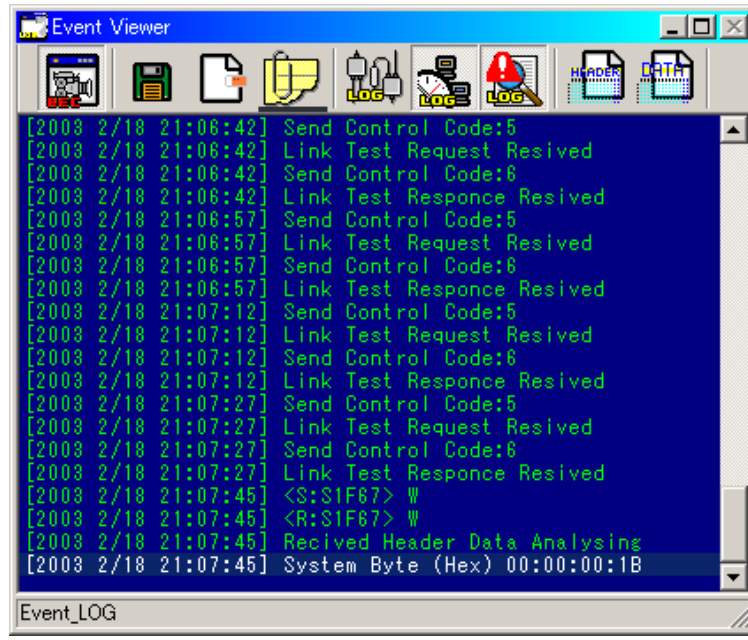
受信したDataのBit情報と現在のhandshake情報、最後に受信したMessageと受信Dataの長さを表示します。
“LOADING”は受信Dataの状況をHeaderのLengthを元にBar表示します。

HSMS Monitor



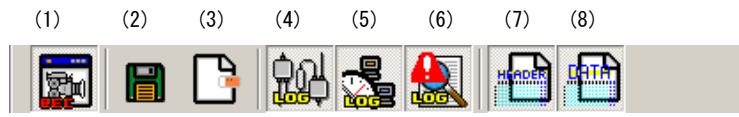
受信したMessageのHeader情報からW-Bitの状態と現在Connect中であれば、
CONの文字列が赤（こちらからConnectすると赤で他方からConnectされると緑）で表示されます
AnalyzedはMessageの解析状況をBar表示します。

Event Viewer



SECSeIImeで発生したEventをReal Timeに表示します。

Functions



#	Function Name	意味
1	Event Viewer Active	Eventの表示 ON/OFF Event Viewerが負荷に成っている場合OFFする事が出来ます。
2	Save	Save Function（最新の300個のData）➔ 表示分のみ 現在表示中のEventを任意のFileにSaveします。
3	Clear All	Event Viewerの表示を消去
4	Common Event	SECS-I / HSMSのFlagやControl CodeのEvent表示 ON/OFF 低レベルのEvent表示を規制する事が出来ます。
5	Send / Receive Time Stamp	実際に送信された時間と実際に受信した時間の表示 *1
6	Error Event Report	通信異常の表示 ON/OFF Retry等の状況を観測する事が出来ます。
7	Header Dump	HeaderをDump表示します。
8	Data Dump	DataをDump表示します。

＊１：送信Data Logは送信Messageを生成した時にLoggingされます、この為実際の送信時間と異なっています（Intervalの設定とそのときQueueにDataが貯まって居る場合）
このOptionは実際に送信したときに表示されますので正確な送信時刻となります。

Script Functions

Function Spec.

オフセット概念

スクリプト上で扱う構造体（ストラクチャー）はオフセット概念を用いて指定されます。
オフセットはストラクチャーのData記述位置以降の行番号に対するオフセットです。

オフセット指示の例

```
 1 Fun,S2F41
 2 SBINC
 3 W=1
 4 E=1
 5 BLKNO,1
 6 Data
 7 List,2
 8 ASC,START
 9 List,3
10 List,2
11 ASC,PORTID
12 BIN,2
13 List,2
14 ASC,CSTID
15 ASC,TESTCSTID00202
16 List,2
17 ASC,MSG
18 ASC,HOST Message Today is Sunny Day 19 END
```

上記の例では、８行目の"ASC,START"は6 Data から見た時 2番の位置なのでオフセットの２に成ります。
例えば、STARTをCANCELに書き換えたい場合はS2F41のオフセット２に"CANCEL"を書き込みます。
ASCの部分は書き換えられません。
その必要が有る場合はファイル名を変えて他の名前で予め登録しておく必要が有ります。

受信データも同様にオフセットで与えられた数値が指す位置のデータが対象となります。

構文禁則

スクリプトでは解析出来ない指令文字列をスキップします。
将来、高速可の為と拡張性のためにスクリプト全体の変更を行い
不正文字列の表示を行う予定です。

スクリプトの構文解析手順

スクリプトを実行するモジュールをインタプリタと呼ぶ事にします。
インタプリタは構文をカンマ単位で分解し左側より順に解析します。

EX) If,Function_Name,S2F25,Loop_Back の例

インタプリタはIfから続く文字列が”Function_Name”なので続く”S2F25”が受信ファンクション名か確認して結果＝真なら更に次の”Loop_Back”からアドレス@Loop_Backに移行します。
つまり、Function_NameがS2F25なら@Loop_Backに移行する　という意味に成ります。
解析段階でエラーを認識すると処理を中止するか、無視するかの選択が出ます
中止を選択するとスクリプトを強制終了します。

ファイル異常

スクリプトからファイルを操作したときに発生する異常では、読み込み時に存在しないファイル名を与えた場合と、書き込みに失敗した場合に限り異常を発行します。
現在のバージョンではハードウエア異常などに対して応答できるかの検証はされていません。

Script Control Function

INCOM_F指令

使用例

```
INCOM_F
```

受信フラグの初期化を行います。
（旧仕様のため、現在はWit_Incomで初期化されます）

SAVE指令

使用例

```
SAVE,TEST
```

内部に置かれた構造体を"TEST.STR"としてファイルに書き出します。
同じ名前のファイルが存在すると消去後書き出します。
SEND命令で送信することが出来ます。

READ指令

使用例

```
Read,S7F83_RECIPE,TEMP
```

ファイル"S7F83_RECIPE.str"を読み込んで、内部では"TEMP"と言う名称で扱います。
この指令以後Re_Writeメソッドの対象がTEMPに成ります。
送信する場合は一旦 "SAVE"コマンドで書き出してから"SEND"命令で送信します。

SEND指令

使用例

```
SEND,S7F83_RECIPE,20
```

ファイルS7F83_RECIPE.strを送信します。
インターバルタイマーに20mmsecのパラメータを与えます。
(送信可能状態から更に20mmSecの遅延を設けます。)
インターバルタイマーの値が"0"の場合送信Que Stackの先頭に送られます。

MEMO指令

使用例

```
MEMO,Time OUT Error
```

Tree viewにﾒｯｾｰｼﾞ(Time OUT Error)を書き込む
LOGに残ります。

DIM指令

使用例

```
Dim,%P1_LOTID,Nothing
```

P1_LOTIDと言う変数を生成して"Nothing"で初期値する。
変数の名称規則は有りません。（解るように工夫する必要が有ります）
内部では全てACIIで保管されます。
演算指令は一旦Integerに置き換えられて演算します。
この時内容が数値でない場合エラーを報告します。

Calc Function

INC ：変数に+1する

使用例

```
CALC,INC,%Data
```

%Dataに＋１する。
変数の内容が数値でない場合エラーが発生します。

DEC ：変数に-1する

使用例

```
CALC,DEC,%Data
```

%Dataに－１する。
変数の内容が数値でない場合エラーが発生します。

+ ：変数と定数を足し算する

使用例

```
CALC,+,%Data,5
```

%Dataと5を加算し%Dataに渡す。
変数の内容が数値でない場合エラーが発生します。

- ：変数と定数を引き算する

使用例

```
CALC,-,%Data,5
```

%Dataから5を引き算し%Dataに渡す。
変数の内容が数値でない場合エラーが発生します。

* ：変数と定数を掛け算する

使用例

```
CALC,*,%Data,5
```

%Dataと5を掛け算し%Dataに渡す。
変数の内容が数値でない場合エラーが発生します。

/ ：変数と定数を掛け算する

使用例

```
CALC,/,%Data,5
```

%Dataを5で除算し%Dataに渡す。
あまりは消去されます。（戻り値はIntegerです）
変数の内容が数値でない場合エラーが発生します。

LET ：変数に数値（文字）を入れる

使用例

```
CALC,LET,%Data,5
```

%Dataに5を入れる。
変数の内容が数値でない場合でも処理します。

Timer Function

CREATE ：タイマーを生成する

使用例

TIMER,CREATE,T1

T1という名前のタイマーを生成する。

KILL ：タイマーを消去する

使用例

TIMER,KILL,T1

T1という名前のタイマーを消去する。

START ：タイマーを起動する

使用例

TIMER,START,T1

T1という名前のタイマーをスタートする。

CLOSE ：タイマーを停止する

使用例

TIMER,CLOSE,T1

T1という名前のタイマーを停止する。

INTERVAL ：タイマーのインターバル時間を設定する

使用例

TIMER,INTERVAL,T1,1000

T1という名前のタイマーのインターバルを1000mmSecにする。

FLAG_OFF ：タイマーのON Flagをリセットする

使用例

TIMER,FLAG_OFF,T1

T1という名前のタイマーのON Flagをリセットする。
タイマーのON Flagは一旦セットされるとリセットを行わない限り持続します。
ONのまま再びタイムアップしてもFlagに変化有りません。

STRING Function

使用例

STRING,%STAR-1,%STR-2

文字列足し算の処理　：指定された変数の値をに変数の内容を追加する。
STRING,<VALUE1>,<VALUE2>　Value1 = Value1 + Value2

Re_Write Function

このグループは予めREAD命令で構造体を読み込む必要が有ります。

OFF_SET ：指定箇所の直接変更

使用例

RE_WRITE,OFF_SET,5,Cancel

READ命令で読み込まれた構造体のオフセット５のデータを"Cancel"で置き換える

R_DATA ：構造体のデータを受信データで置き換える

使用例

RE_WRITE,OFF_R_DATA,5,3

READ命令で読み込まれた構造体のオフセット5のデータを受信データのオフセット3のデータで置き換える。

DIM ：構造体のデータを変数で置き換える

使用例

RE_WRITE,DIM,5,%DATA

READ命令で読み込まれた構造体のオフセット５のデータを変数%DATAで置き換える。

IF Function

Function_Name ：最近に受信したSFを比較する

使用例

if,FUNCTION_NAME,S2F17,TIME_SET

最近受信したStream Functionが"S2F17"ならば@TIME_SETに移る

Function ：最近に受信したStreamを比較する

使用例

if,STREAM,1,TIME_SET

最近受信したSTREAMが"1"ならば@TIME_SETに移る

Function ：最近に受信したFunctionを比較する

使用例

if,FUNCTION,17,TIME_SET

最近受信したFunctionが"17"ならば@TIME_SETに移る

OFF_Set ：最近に受信したDataのOffSet値を比較する

使用例

If,OFF_SET,4,1,PSD_1

最近受信したDataのオフセット４が"1"であれば@PSD_1に移る
比較対照は全て　ASCIIに置き換えられる。

☆　NOT_OFF_SET：OffSetのNOTケース

OFF_SET_DIM ：最近に受信したDataのOffSet値と変数の内容を比較する

使用例

If,OFF_SET,4,%DATA,PSD_1

最近受信したDataのオフセット４が%DATAの内容と同じであれば@PSD_1に移る
比較対照は全てASCIIに置き換えられる。

Dim ：指定した変数の値を比較する

使用例

If,Dim,%ACT_Port,1,PORT_1_Procedure

変数%ACT_Portが"1"であれば@PORT_1_Procedureに移る
比較対照は全てASCIIに置き換えられる。

☆　NOT_Dim：DimのNOTケース

Timer ：任意のタイマーがONしているか調べる

使用例

If,Timer,2,TIMER_2_ON

タイマー"2"がONであれば@TIMER_2_ONに移る

Show Function

Print ：MessageBoxを使った表示

使用例

```
Show,Print,Start OK? at Ready
```

Start OK? at Readyと書かれた確認BOX(Message Box)が現れる。
OKを押すまでステップは進まない。

Printf ：MessageBoxを使った変数の表示

使用例

```
Show,Printf,Pass Count:,%Pass_C
```

Pass Count:と変数%Pass_cの内容が書かれた確認BOX(Message Box)が現れる。
OKを押すまでステップは進まない。

Message ：Event Displayへのメッセージの書き出し

使用例

```
Show,Message,Hello World,1
```

Event DiplayにHello Worldがタイムスタンプ付きで表示される
１＝タイムスタンプ付き
２＝文字列のみ

Value ：EventDisplayに変数名と変数値を表示

使用例

```
Show,Value,%P1_LOTID,1
```

Event Diplayに変数名"%P1_LOTID"とその内容がタイムスタンプ付きで表示される
１＝タイムスタンプ付き
２＝文字列のみ

R_Data ：文字列とオフセットの内容表示

使用例

```
Show,R_Data,PORT#1=> ,5,1
```

Event Diplayに文字列"PORT#1=> "と受信データのオフセット５がタイムスタンプ付きで
報告される。
１＝タイムスタンプ付き
２＝文字列のみ

その他

特殊な命令

ストラクチャのData記述部に記述する事で送信時点まで解らないデータをストラクチャに与えます。
この記述はストラクチャエディターでは編集する事が出来ないのでテキストエディターを使用してください。

$Date : YYYY/MM/DD/hh/mm/ssの形式で時間データを与えます、戻り値はASCです。
$Date_OLD :YY/MM/DD/hh/mm/ssの形式で時間データを与えます、戻り値はASCです。
$Date_16 :YYYY/MM/DD/hh/mm/ss/ccの形式で時間データを与えます、ccは常にＯＯです。

使用例

```
 1 Fun,S2F18
 2 SBCOPY
 3 W=0
 4 E=1
 5 BLKNO,1
 6 Data
 7 ASC,$DATE
 8 End
```

$Copy :受信したデータ構造でエコーバックします。

使用例

```
 1 // Echo Back Function
 2 // Special Data $COPY
 3 Fun,S2F26
 4 SBCOPY
 5 W=0
 6 E=1
 7 BLKNO,1
 8 DATA
 9 $COPY
10 END
```

ASCII <STRING> :<>内の文字列を引用してAscii入力Dialog Boxを開きます。
入力されたAscii文字列はメッセージに反映します。
この特殊命令のキーワードは"<"です。