

SECS/HSMS communication simulator
(tdISSim)

(Trust Design Simple SECS/HSMS Simulator)

Instruction Manual

Version 12.121 : 2012.12.12
Version 14.040 : 2014.04.25
Version 15.041 : 2015.04.25
Version 15.050 : 2015.05.08
Version 15.080 : 2015.08.01
Version 15.111 : 2015.11.27
Version 16.012 : 2016.02.25
Version 16.040 : 2016.04.05
Version 16.060 : 2016.06.10
Version 17.030 : 2017.03.01
Version 18.011 : 2018.01.13
Version 18.020 : 2018.02.10
Version 18.032 : 2018.03.28
Version 18.041 : 2018.04.23
Version 18.070 : 2018.09.25
Version 19.020 : 2019.02.01
Version 19.070 : 2019.07.05
Version 19.100 : 2019.10.25
Version 20.040 : 2020.04.17
Version 20.090 : 2020.09.01
Version 21.011 : 2021.02.25
Version 21.061 : 2021.07.23
Version 23.041 : 2023.06.15
Version 23.061 : 2023.06.15
Version 24.042 : 2024.05.23
Version 24.060 : 2024.06.20
Version 24.070 : 2024.07.24

Trust Design Limited Liability Company

Chino City Nagano Prefecture Japan

E-mail: info@trust-design.co.jp
URL: <http://www.trust-design.co.jp>

T a b l e o f C o n t e n t s

1. Introduction	1
2. Operation explanation	3
3. Script language specification	24

1. Introduction

This program simulates communication conforming to SEMI standard 1 message transfer (SEMI E4/SECS-1), high-speed SECS message service single session mode (SEMI E37.1/HSMS-SS) and high-speed SECS message service general session (SEMI E37.2/HSMS-GS).

This program has the following features.

- + Supports host side, device side, master side, slave side, passive side and active side.
- + It has message sending function, receiving function, and reply function to received message.
- + The reply can be made by automatically selecting appropriate reply message that matches the received message and replying automatically, or user can select and reply from multiple messages to be replied to.
- + It is possible to automatically operate continuous SECS message communication using script files written in own simple language.
- + SECS Communication trace can be saved in specified number and capacity of specified file.

- + Send and receive specified messages using message definition files in SML format.
- + It is possible to change and set data value of send message item according to send situation. In addition, it is possible to obtain a data value from file contents by preparing a file storing a large amount of data values and specifying file name as the data value of send message item.
- + You can use variable-length fields and items.

- + It is possible to define multi-level, multi-number indefinite number list. It is possible to fix number of lists at runtime and set each item value in the list.

- + The program can automatically operate continuous SECS message communication using its own simple language. This simple language has following mechanism.
 - Multiple scenario (execution sequence) definition
 - Variable (string, integer, real number)
 - Variable operation (four arithmetic operations etc.)
 - Condition judgment by IF statement
 - Processing branch by block IF statement, GOTO statement
 - Iteration by WHILE statement
 - Function call by CALL statement
 - Invoke external program by EXEC statement
 - SECS message sending and receiving
 - Change data item values that make up send message when sending
 - Extract data item value from received message and set as variable
 - Other ...
- + This simple language can be used as a simulation operation for normal SECS communication, but it is unfortunately not good enough in terms of scripting method, operation speed, etc.

- + In this program, there are parts that have been removed regarding error processing, processing speed, help function, simple language specifications, operation instructions, etc., but ... please forgive me.
- + Also, for same reason, it is assumed that well-meaning users who use them with correct settings and correct usage are intended for use. There is a part that does not correspond to some useless ways, etc., but please forgive.

- + For development of communication system by SECS/HSMS, our SECS/HSMS communication package (Trust Design Simple SECS Communication Library) is available.
For more information, please visit our home page.
- + You can use SECS/HSMS protocol converter program (Trust Design Simple SECS/HSMS Protocol converter) is released.
For more information, please visit our home page.
- + For monitoring communications by SECS (HSMS), our network communications monitor (Trust Design Simple) Network Communication Monitor) is available.
For more information, please visit our home page.
- + For monitoring communications by SECS-1 protocol using RS232C Serial Port, our serial communication monitor (Trust Design Simple Serial Port Communication Monitor) is available.
For more information, please visit our home page.

(Note) -----+
| This package uses following ports of UDP/IP for license management. |
| Also use following class D address as UDP/Multicast address. Please set not to block these |
| by firewall etc. of your computer. |
| - 36275/udp |
| - 239.254.200.75 |
| However, even if you can not connect to Internet connection environment, you can use it, and |
| there are no functional restrictions on usage as compared with same environment. |
+-----+

2. Operation explanation

(0) Preparation

Before starting this program, be sure to set up and prepare following two files correctly.

- + SECS/HSMS Communication parameter setting file (.ini file)
- + SECS/HSMS Communication message structure definition file (.sml file)

In addition, when performing automatic operation of communication, it is necessary to create in advance the following files that describe communication sequence with correct settings.

- + Scenario execution sequence definition script file (.ssl file)

The setting method of each file of .ini and .sml is in "Programmer's Manual (TDSE. pdf)" attached to our "SECS/HSMS communication package (Trust Design Simple SECS Communication Library) (TDS)". Please refer to the applicable part (.ini: 2.1(1), .sml: A(1)). "SECS/HSMS communication package" can be downloaded from our web site (<http://www.trust-design.co.jp/>).

Refer to Chapter 3 for the description of .ssl file.

In addition, the sample file of .ini, .sml and .ssl is attached to this package. First of all, it is recommended to use this sample file and revise it as needed.

(Note 1) When setting the SECS/HSMS communication parameter setting file (.ini), pay particular attention to following items. (For details, please refer to the above TDSE.pdf 2.1(1).)

- + SECSMODE : SECS communication parameter
 - bit#0,1 Communication type (SECS-1 or HSMS-SS)
 - 4 Equipment or Host
 - 5 SECS-1 : Master or Slave
 - 6 HSMS : Active or Passive
- + DEVMODE : Device control mode
 - bit#0 Device ID check
 - 1 Processing when receiving secondary message that is not in receiving wait state
 - 8-12 ... S9Fx, Reject Automatic transmission
- + XDEV : Maximum number of connected devices
- + DEVID : Connected device ID
- + XMSGSIZE : Maximum SECS message byte length
 - Specify a numerical value with some margin.
- + SDEVICE : COM port name used when connecting to SECS-1 ("COM1" etc.)
- + HOST : Connection destination host name or IP address for Active connection at HSMS-SS connection.
- + PORT : TCP/IP port number used for HSMS-SS connection
- + LINKINT : Link test execution interval when connecting HSMS-SS
 - When performing link test, specify number of seconds for the execution interval.
- + TRCDIR : Communication trace file storage folder
 - When specifying relative path, folder where .ini file exists is base point.
 - Refer to TDSE.pdf 2.1(3) for file name of communication trace file.
- + TRCTTYPE : Format of communication message output to communication trace
 - Refer to TDSE.pdf 2.1(1) (c)
- + TRCTOUT : Communication trace output mode
- + TRCTLEVEL : Communication trace output level
 - When outputting communication control code in SECS-1, specify a value of 6 or more. When outputting trace related to link test in HSMS-SS, specify a value of 9 or 10 or more.

<< Continue to next page >>

<< Continue from previous page >>

- + MDMSSG : Specify a message definition file (.sml) to be used for communication trace output.
- + MDMXITEM : Maximum number of total data items
- + MDMXMSSG : Maximum number of messages to define
- + MDMXITEM : Maximum number of total data items + maximum number of items when expanding messages
- + MDMXPPOOL : Message definition Setting data storage area size

For these items, specify numerical values with some margin.

(Note 2) If opposite side SECS message transmission frequency is high (message transmission at high speed continues, etc.), please note the setting of following items in order to improve response speed of tdISSimE.

- + INTERO Communication control unit processing interval
(For details, please refer to [Information] of TDSE.pdf 2.1(2) (c))
- + TRCTTYPE Communication message output format to communication trace
(For details, please refer to (Note) of TDS.pdf 2.1(2) (c) TRCTTYPE)
When communication trace output is performed in List format, if it is set to analyze SECS-2 message using message definition file and display message name, item name, etc., many resources are required for the process. Even if list format output is necessary, display output of message name and item name is not necessarily required, set so that it is not displayed.
- + TRCPOUT, TRCUOUT ... tdISSimE outputs processing trace and user I/F function trace in addition to communication trace. Since these trace outputs will also be a load, please do not set these outputs (TRCPOUT=0, TRCUOUT=0).
- + TRCTHOST, TRCTPORT . If external monitoring of communication trace is not required, the processing load on external communication trace output requires a corresponding processing load, so set this function not to be used (TRCTHOST="", TRCTPORT=0).

(Note 3) In the cases shown in (Note 2) (when the frequency of sending SECS message of communication partner is high), the response speed of tdISSimE can be improved by following measures.

- + By unchecking [Display]-[List format display] in screen settings, you can stop LIST format display of SECS message on screen. If you need to display the details of SECS message, please select display in [Hexa decimal format display].
(Of course [Hexadecimal format display] is also expensive, so if you don't do [Hexadecimal format display], the load on tdISSimE will be reduced.)
- + When executing scenario script and performing response processing with scenario script, if it is possible to substitute the automatic reply function that tdISSimE has for reply processing, using automatic reply function instead of scenario script has less load and response speed is improved. If the use of a scenario script is indispensable, if it is acceptable to use an automatic response instead of processing in scenario script for received message, the response can be made by using an automatic response without specifying that message in the RESERVE statement. It can improve the speed.
- + **You can improve the response speed by specifying following items ([Trace file out], [Script real time display]) in [Display]-[Display function temporary stop] in the screen settings.**
In particular, when [SScript real time display] is specified during scenario script execution, a high effect can be obtained.

(1) Start-up

Start `tdlSSimE.exe` (or a shortcut to `tdlSSimE.exe`) in the installed folder by double-clicking etc.

This program uses "MS Gothic" as the font used.

Please execute in the environment where the font can be used.

(Reference) The following can be specified as startup options.

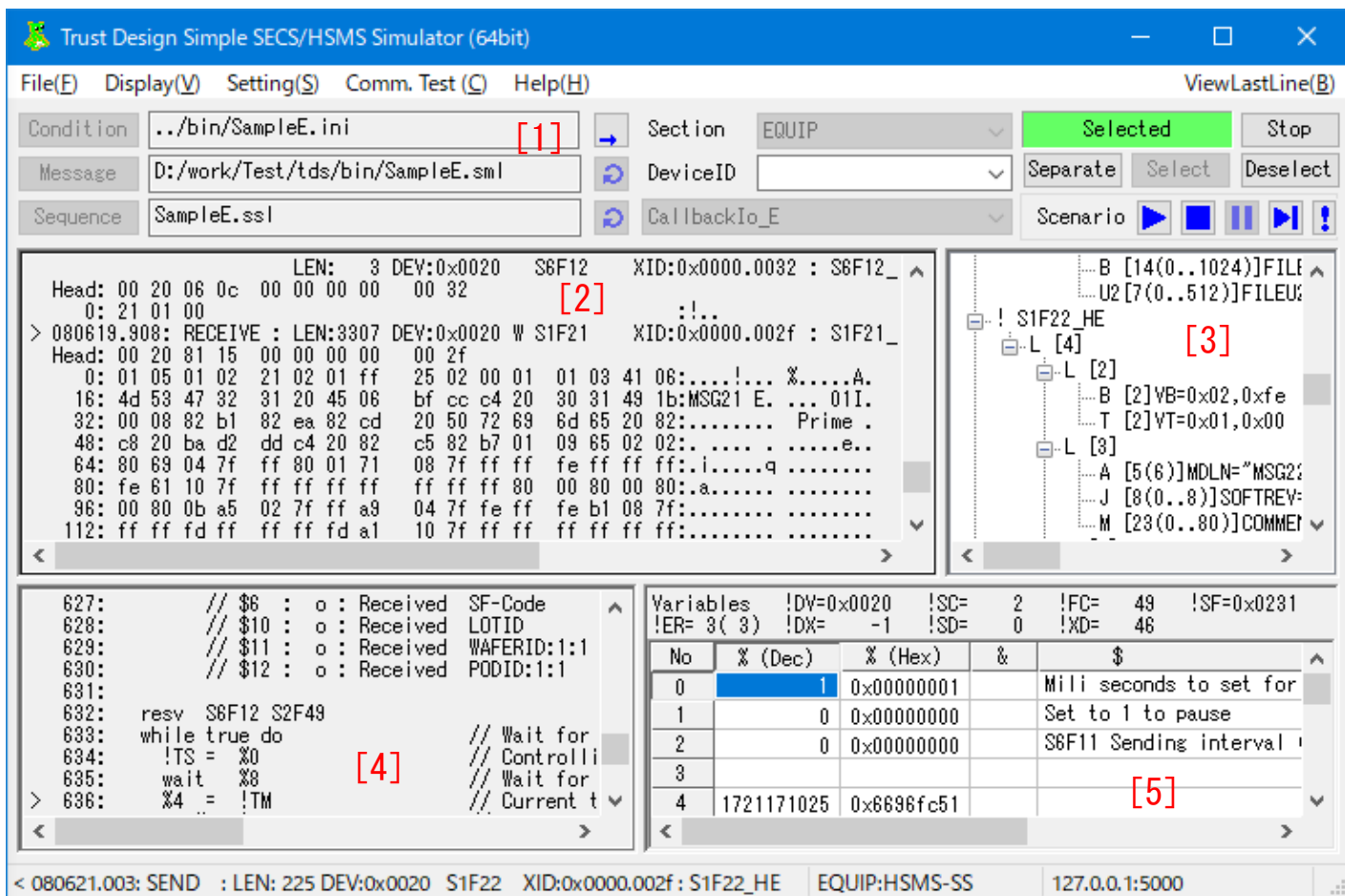
<code>-h</code>	: Displays a list of startup options.
<code>+A</code>	: [Start] automatically after startup.
<code>+A ProgName</code>	: If you specify a script program name, the specified script program will be automatically executed after [Start], and will be automatically [Stop] after the specified script program ends.
<code>+AC ProgName</code>	: Similar to "+A ProgName", but continues processing even after the specified script program ends.

(Note) The [Setting] status of `tdlSSimE` when operating with this function depends on the status at the time of last normal startup, except for the script program name, automatic connection specification, and automatic response specification.

(Example)

<u><code>tdlSSimE +A</code></u>	: After launching <code>tdlSSim</code> , start the process automatically, and in the case of HSMS, CONNECT and SELECT processing are automatically performed.
<u><code>tdlSSimE +A BOOTSEQ</code></u>	: After launching <code>tdlSSim</code> , start the process automatically, and in the case of HSMS, CONNECT and SELECT processing. After confirming the connection, automatically execute the sequence program "BOOTSEQ". When the sequence program is completed, the operation of <code>tdlSSim</code> is terminated (after SEPARATE processing in the case of HSMS).
<u><code>tdlSSimE +AC BOOTSEQ</code></u>	: Similar to the above "+A", but the simulator operation will continue after the sequence program completion.

(2) Screen operation explanation



- [1] : Overall control panel
Setting of operating conditions.
Start and stop simulation. Perform operations such as communication scenario execution.
- [2] : Communication trace display
Displays trace of SECS communication messages sent and received.
(Note) Displayed transaction ID (XID) of sent message assigned may differ from the value when actually sent. (Especially when "SRCID0 value == SRCID1 value" in .ini file.)
For the correct value, check 'Extended communication trace display' or 'Communication trace file'.
- [3] : Communication message edit
Edit SECS communication message to be send/receive, which is determined by specified "Message definition file", and edit message items that make up send message.
- [4] : Auto run script
The contents of specified "Sequence definition file" are displayed, the script corresponding to selected scenario is confirmed, and when the scenario is automatically executed, execution statement is shown.
- [5] : Script variable
Displays contents (values) of "Script variable" used by automatic execution script. Also, specify (change) the value.

[Note] The display language used in dialogs such as "File selection dialog" depends on your Windows environment. For example, when `tdlSSimE.exe` is started in Japanese environment Windows, display language of "File selection dialog" will be Japanese.

< Reference > General operation procedure

1. Select .ini file that describes SECS/HSMS communication parameters with [Condition] button.
2. When automatically executing a scenario, use [Sequence] button to select .ssl file containing scenario execution sequence definition script.
3. In the [Display] menu, specify the display format for communication trace window [2].
4. In the [Setting] menu, select "Equipment" or "Host".
5. Start the simulation process with [Start] button.
6. Processing when "Auto Connect" is not specified in [Settings] menu of HSMS active connection.
For HSMS-SS, press [Connect] and [Select] buttons in order to establish communication with the passive side.
For HSMS-GS, after connecting to passive side with [Connect], select the DeviceID to be selected, and with [Select] button, establish communication regarding target DeviceID.
7. After that, please enjoy your favorite operation.

(Note 1) If the following error occurs when pressing [Start] button, change the specified value of the corresponding part of specified .ini.

```
-941 : Insufficient space in data item definition table ..... MDMXITEM
-942 : Insufficient space in message definition table ..... MDMXMSSG
-943 : Insufficient space in table for storing items for each message ..... MDMXMITEM
-944 : Insufficient data storage area to set/check items per message ..... MDMXPOOL
```

Other error numbers are same as "TDS". Please refer to corresponding section in "Programmer's Manual (TDSE.pdf)" attached to above-mentioned "SECS/HSMS Communication Package (TDS)".

(Note 2) When this program operates in automatic response mode, basically all messages that may be received should be defined in message definition file (.sml). The following secondary messages can be sent back regarding receiving of specified SF-Code, regardless of message structure.

- + Define default secondary messages in message definition file. For details, please refer to A. (1) (b) of TDSE.pdf described in (Note 1).
- + Use scenario execution script. Refer to the notes in Chapter 3. (0) for details.

(a) Menu

(a-1) [File]

- + End of application .. Exit tdISimE.

(a-2) [Display]

- + Clear communication trace
Clear Communication trace display window [2].
- + View last line of communication trace
Set the scroll bar of Communication trace display Window[2] to the state where the last line is displayed. (Used when final line does not easily appear in normal scroll bar operation, etc., with auto scrolling at high speed, etc.)
- + List format display
Displays send/receive SECS message in specified List format in communication trace window.
- + Hexa format display
Displays send/receive SECS message in Hexadecimal format in communication trace window.
- + List Hexa Mix format display
When displaying List, Hexa display is added to the back of each line.
- + Script trace display
When executing a scenario automatically, displays the script statement currently executed.
- + Extended communication trace display
A more detailed communication trace is displayed in another window. The extended communication trace display window is closed in that window.
(Note 1) The extended communication trace display has the same contents as communication trace recorded in communication trace file.
Therefore, not only the contents displayed in communication trace window [2] but also SECS-1 control code, HSMS LinkTest request and response, etc. can be displayed.
However, presence or absence of those outputs depends on TRCTTYPE, TRCTLEVEL value in specified operating condition file (.ini).
For details, refer to description of related tokens and parameters in TDSE.pdf 2.1 (1)(c).
- + Display function temporary stop
 - Trace file out If specified, output to all trace files will be suspended.
Extended communication trace display is also paused.
Please note that the events that occur during the pause will only be displayed in trace output area on screen.
 - Script real time display .. Pauses the real-time display of the current execution line in autorun script area if script is running.
(Note 2) This function is specified to speed up execution of this AP as much as possible. In particular, the specification of [Script real time display] greatly contributes to improvement of operation speed when executing script. In order to increase execution speed as much as possible, specify this. And To suppress display in the communication trace area, stop above [List display], [Hexa display], and [Script trace display].
- (Note 3) The [Trace file out] setting is cleared when simulation starts.
- + Save windows position and size
When the program ends, information such as window position at end time, size, and specified conditions are saved. The next time you start up, the screen state will be restored based on that information. (The status is saved in tdISSimEWin.ini in same folder as tdISSimE.exe.)
- + Status bar display
Display status bar.

(a-3) [Setting]

+ Equipment

The simulation is performed on the "Equipment" side.

If [Section] is not specified in [Condition], [EQUIP] section of .ini file containing specified SECS/HSMS communication parameters is used.

+ Host

The simulation is performed on the "Host" side.

If [Section] is not specified in [Condition], [HOST] section of .ini file containing specified SECS/HSMS communication parameters is used.

+ Automatic connection

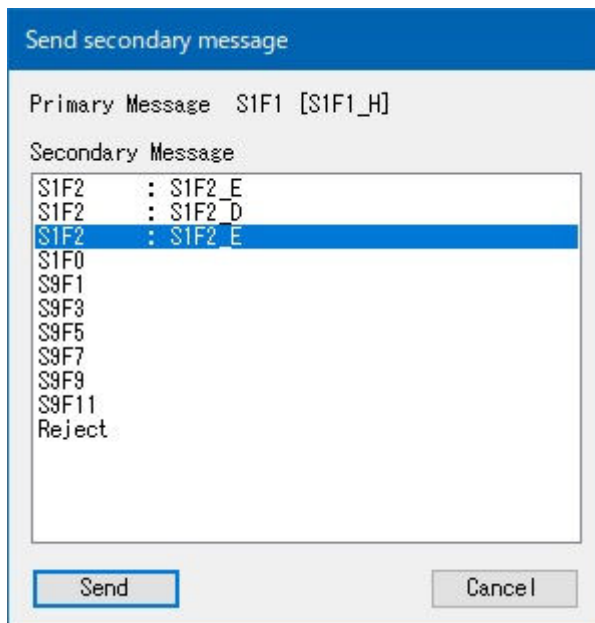
If the setting in .ini file that describes specified SECS/HSMS communication parameters is HSMS active connection, Connection to passive side and Select processing are automatically performed after the simulation is started.

(Note 4) Automatic connection is made in case of "Active connection" setting.

+ Automatic response

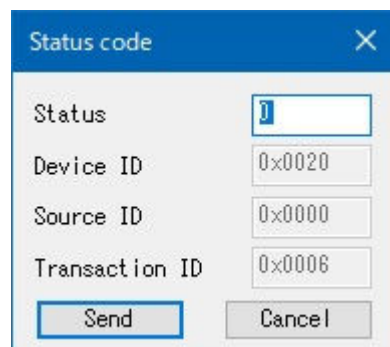
If checked, when primary message that requires reply is received from the other side, the optimal secondary message is automatically selected and sent according to settings in specified message definition file (.sml).

(Note 5) When multiple secondary messages with same SF-Code to be replied are set in .sml file, select the first secondary message after the primary message definition determined by parsing the received primary message. If there is no definition of target secondary message behind received primary message definition, search will return to beginning of definition file and search. If still can not find it, reply S9F3, S9F5 etc.



When unchecked, when receiving primary message that requires reply from the other side, dialog box for selecting reply message is displayed, and message to be sent is selected.

If you specify Reject and send, specify Status (F-Code) in the following dialog.



+ Message direction evaluation

The validity of the message for each "Equipment" and "Host" specified in message definition file (.sml) is judged, and only the valid transmission target messages can be selected for transmission. When unchecked, it is possible to select all messages as transmission targets regardless of the current operating status, "Equipment" or "Host".

- + Statistic Information Instructs processing related to statistical information such as the number of transmissions and receptions for each SF-Code that the simulator internally collects.
 - Display Displays the collected statistical information by SF-Code in the communication trace display area. Please refer to the explanation about each member of statistical information table structure of TDS.pdf 2,2 (5) for the display contents.
 - Output List to Comm. Trace File .. Outputs collected statistical information by SF-Code to the communication trace file in List format.
 - Output CSV to Comm. Trace File ... Outputs collected statistical information by SF-Code to the communication trace file in CSV format.
 - Initialize Clears and initializes internally held statistical information for each SF-Code.

(Note 6) To enable this feature, set STATISINFO to 1 in your configuration file (.ini).

+ Length Byte Size

Specify to fix the number of bytes of "Length Byte" used by each data item that composes SECS message to be sent.

- 0 Auto Determined by the number of data items Automatically determined by the number of data item bytes.
- 2 Byte Length Byte Fixed to 2 bytes.
- 3 Byte Length Byte Fixed to 3 bytes.

(Note 7) Normally, set it to [0 Auto]. Specify this when the communication partner can interpret only a fixed number of Length Bytes.

If the number of data item bytes does not fit in 2 bytes even if [2 Byte] is specified, the Length Byte will be 3.

(Note 8) This specification does not apply to messages such as S9Fx and SxF0 that simulator itself responds to. Only messages that can be modified by user are applicable.

(a-4) [Comm. Test]

+ Communication test mode

If checked, it will be in the mode to do a simple communication test.

In this mode, you can specify the following.

- Specification of each item value of SECS header part of transmission data
- Byte length of SECS message part of transmission data
- In the case of HSMS connection, specify the byte length of sending TCP/IP packet and divide sending
- In the case of SECS-1 connection, simulate various abnormal conditions at sending and receiving transfer blocks.

In this mode, when send message is selected, following dialog will be displayed, and the above parameters can be specified.

The header value of SECS message to be sent next will be displayed by default, so correct the values as necessary. When entering hexadecimal number, add "0x" as a prefix.

+ P-Type Code : See below (Note 10)

+ S-Type Code : See below (Note 10)

+ Message length

Specifies byte length of SECS message part of send message.

+ Sending size

When connecting HSMS, specify 1 or more numeric value when transmitting divided transmission packets. If it is 0, SECS message part will be sent at once.

When connecting SECS-1, specify transmission byte length of data part (0 to 242) at the time of multi block transmission. If it is 0, it will be 242 bytes.

(Note 9) This mode can be specified only when script execution is not performed and automatic response is disabled.

(Note 10) When specifying P-Type and S-Type Code, if the communication format is HSMS, you can specify P-Type and S-Type Code themselves. When sending a normal SECS-2 message, P=0 and S=0, so be careful when specifying anything else.

When SECS-1 is connected, specify the error occurrence state when sending BLOCK.

P-Type : Specify the initial value of BLOCK number

S-Type : Specify the inspection content

- 0- 9 : Specify increment of BLOCK number and simulate BLOCK abnormality.
Check the receiving process on the other side (receiving operation of primary message sent from here)
- 0 : Duplicate block transmission
Simulate duplicate BLOCK (no ACK received) only on first transmission
- 1 : Normal condition
- 2 : Specify illegal BLOCK number transmission (This process may be meaningless)
Correct BLOCK number is sent at first retry due to ACK not received or ACK received from the receiving side.
Return to normal value at first Retry.
- 3 : Behaves same as 2., and regular BLOCK number is sent at final retry.
- 4 : Behaves same as 2., and regular BLOCK number is not sent.
- 5-8 : <reserved> Current status is same as set 1.
- 9 : Behaves same as 0., and simulate duplicate BLOCK repeat until final retry transmission

< Continue to next page >

< Continue from previous page >

- 10-14 : CheckSum Specify abnormality
 Check the receiving process on the other side (receiving operation of primary message sent from here)
- 10 : Return to normal value at first Retry
- 11 : Return to normal value at final Retry
- 12 : Do not return to normal value
- 13 : CheckSum abnormality only on first transmission of BLOCK#2
- 14 : CheckSum abnormality only on all transmission of BLOCK#2

- 20-21 : [ENQ] Mimicking conflict in collisions
- 20 : This performs processing on the primary message sent by this side.
 This side is the sender who sends the primary message.
 The other side is the receiver who receives the primary message.
 Processing will be performed assuming that [ENQ] has been returned in response to the sender's [ENQ] (the [EOT] that was originally returned will be treated as [ENQ]).
- 21 : This performs processing on the secondary message sent back by the other side.
 The other side is the sender who sends the secondary message.
 This side is the receiver who receives the secondary message.
 The process is performed by returning [ENQ] instead of [EOT] to the sender's [ENQ]. If this side is the 'Master', continue to wait for [EOT] and then send a 'Dummy Message' (S127F255, DataLength=0, W-Bit=0, SystemByte=(SRCID0|XIDMAX)).
 (Note) Dummy messages are recorded in communication trace file. On the screen, it is displayed only in extended communication trace display window.

- 30-49 : Simulates the state when no control code is received.
 Check the receiving process on the other side (receiving operation of primary message sent from here.)
- 31 : The sending side will ignore first [EOT] it receives, and will receive [EOT] normally next time it retries.
- 32 : Do the same process of '31.' for [LEN]
- 33 : Make the receiving side will be in state where the received length is insufficient. When retrying, the state will be normal.
- 34 : Do the same process of '31.' for [ACK]
- 41 : Do the same process of '31.'. But not return to normal state.
- 42 : Do the same process of '32.'. But not return to normal state.
- 43 : Do the same process of '33.'. But not return to normal state.
- 44 : Do the same process of '34.'. But not return to normal state.

- 50-69 : Simulates the state when no control code is sent.
 Check the other side sending process (the sending of secondary message in reply to primary message sent by this side.)
- 51 : This side will ignore first [EOT] it from other side, and will receive [EOT] normally next time it retries.
- 52 : Do the same process of '51.' for [LEN]
- 53 : Make this side will be in state where received length is insufficient. When retrying, the state will be normal.
- 54 : Do the same process of '51.' for [ACK]
- 61 : Do the same process of '51.'. But not return to normal state.
- 62 : Do the same process of '52.'. But not return to normal state.
- 63 : Do the same process of '53.'. But not return to normal state.
- 64 : Do the same process of '54.'. But not return to normal state.

The behavior of tdIISim when receiving multiblocks depends on the (SECSMODE&0x60000000) value and (DEVMODE&0x0800) value in .ini. Please refer to relevant section of TDS.pdf.

(Note 11) If you change the value of each item in SECS header and send it, the other side's operation can not be predicted.

In tdIISimE, it processes "it is appropriate." (It does not necessarily mean normal operation.)

Also, if tdIISim is waiting for timeout while sending or receiving data with the other side, tdIISim may suspend processing and prevent you from operating the display screen. Screen operations will return when a timeout occurs, so please wait.

+ Stop control response message test

In the case of HSMS connection, Suppresses transmission of responses to received control messages.

- Select response Do not send "Select response"
- Deselect response Do not send "Deselect response"
- LinkTest response Do not send "LinkTest response"

+ Asynchronous control message sending test

In the case of HSMS connection, send a control message regardless of the current connection state in order to test the communication.

Sending this control message does not change the connection state of this simulator.

(Depending on the response from the other party receiving this transmission, connection state of this simulator may change.)

For "SessionID (DeviceID)" of send request message, set 0xffff for HSMS-SS, and set the value specified by [DeviceID] on screen for HSMS-GS.

- Select request Send a "Select request"
- Select response Send a "Select response"
- Deselect request Send a "Deselect request"
- Deselect response Send a "Deselect response"
- LinkTest request Send a "LinkTest request"
- LinkTest response Send a "LinkTest response"
- Reject message Send a "Reject message"

For Response and Reject, specify the parameters of outgoing message using following dialog.

Status : Specify status code

Device ID : Specify SessionID

Source ID : SourceID (SystemByte Upper 16Bit)

Transaction ID : TransactionID (SystemByte Lower 16Bit)

Each item value defaults to the value of previous related Request Message or Data Message as initial value.

(Note 12) The communication trace display window does not show the exchange of control messages by sending asynchronous control messages.

Please follow the steps below to confirm.

- Select [Display] -- [Extended Communication Trace Display] from the menu, to see the contents displayed in [Extended Communication Trace Display Window].
- Browse the communications trace file.

(Note 13) The sending and receiving status of LinkTest will not be recorded in communication trace file if TRCTLEVEL value in the .ini file is the initial value of SampleE.ini, =5. To record the status, specify a value of =6 or higher (=19 is recommended).

For details, please refer to TDS.pdf 2.1 (1)(c) TRCTLEVEL.

(a-5) [Help]

- + Display error code list
Displays error code list of error numbers displayed by this program.
- + Version information
Displays version information dialog of this program.

(a-6) [ViewLastLine]

- + Positioned on the last line of communication trace display.
(Same operation as (a-2) [View last line of communication trace])

(b) Overall control panel

+ [Condition]

Select .ini file that describes SECS/HSMS communication parameters. Direct input is also possible.



+ Positioned on the last line of communication trace display.

+ [Message]

Select SECS message definition file (.sml) that this program uses for display operation on the window. Direct input is also possible. **When specifying relative path, it will be based on the folder where .ini file exists.**

If this item is not specified, the file specified in MDMSSG of SECS/HSMS communication parameter setting file (.ini) is used as message definition file.

(Note 1) **When displaying message name and item name of analyzed message structure in communication trace file which is output by specification (TRCTOUT, TRCTTYPE etc. of .ini), it is the file specified in MDMSSG of .ini.**

Therefore, it is recommended to specify the SECS message definition file (.sml) that is normally used in the communication parameter setting file (.ini) and operate this item as unspecified.



+ Reloads the specified SECS message definition file (.sml) again.

Use if you change the .sml file after simulation process started.

(Note 2) **In this process, reloading is performed on the window display and message definition related to operations. The message structure, message names, item names, etc. output to communication trace file are message definition files specified in MDMSSG of .ini. If this file is changed during simulation, it will be reflected in communication trace output after a while.**

+ [Sequence]

Select the file (.ssl) containing auto-execution script to be used to execute the SECS communication scenario automatically. Direct input is also possible. **When specifying relative path, it will be based on the folder where .ini file exists.** If you do not want to automatically execute communication scenario, you do not have to specify it.



+ Reloads the specified auto-execution script file (.ssl) again.

Use if you changed the .ssl file after simulation process started.

+ Section

Select the section name to be used in specified .ini file.

If this item is not specified, [EQUIP] is used when operating as "Equipment", and [HOST] is used as section name when operating as "Host".

+ DeviceID

Select or specify DeviceID (SessionID) to be assigned to primary message to be sent.

(If you specify in hexadecimal, add "0x" as a prefix. The drop-down list shows a list of up to 64 DeviceIDs defined in DEVID of the specified .ini file.

DeviceID (SessionID) already in Select state at the time of display is marked with "*".

If this item is blank, the first value defined in DEVID of .ini file is used.

When using HSMS-GS, this field value is also used when sending Select, Deselect, and Separate requests.

+ Scenario selection

Select a scenario to execute automatically.

Select from the list of program names specified in the program statement of specified automatic execution script file (.ssl).

+ [Start] / [Stop] Start : Start simulation processing.
Stop : Stop simulation processing

+ [Connect] / [Separate]

Connect : When not connected automatically when connecting HSMS, connect to Passive side.

Separate : Send "Separate request" to opposite side.

When running HSMS-SS, it will disconnect as it is.

When running HSMS-GS, the operation differs depending on setting of .ini file.

For details, refer to description of SECSMODE in TDSE.pdf 2.1 (1)(c) described in 2. (0).

+ [Select]

Select : If HSMS connection is not automatic connection, send "Select request" to passive side.






When running HSMS-GS, send "Select request" by specified DeviceID (SessionID) to passive side.

+ [Deselect]

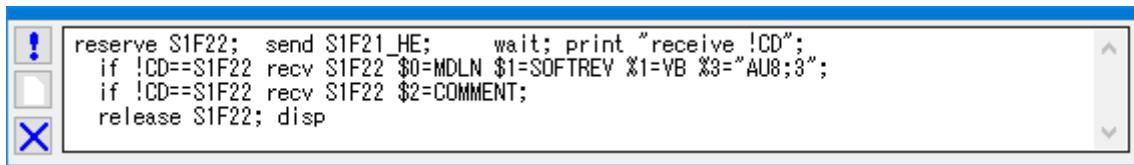
Deselect : Send "Deselect request" to opposite side.




When running HSMS-GS, send "Deselect request" by specified DeviceID (SessionID) to passive side.

(Note 3) In HSMS-SS, Deselect is not usually used.

- +  Start execution of auto-execution script of selected scenario.
- +  Stop execution of currently running auto-execution script.
- +  Pauses execution of currently running auto-execution script.
- +  Executes one line of currently running auto-execution script and pauses.
- +  Execute "OneLine statement".
Details of "OneLine statement" will be given on the next page.

Execute "OneLine statement" window



- + This function is enabled when script execution is paused.
- + Executes "OneLine statement" entered.
"OneLine statement" can contain multiple statements, separated by ';'. You can also describe it as multiple lines. When specifying the second or subsequent item in 'receive' statement, enclose the item name in ". (For example, "AU8;3"). (Refer to 3. (18) scv=item.)
- + The following statements cannot be executed as "OneLine statement".
 - Declaration
(program, function, end, LABEL:)
 - Statement that changes stopping position so that next line to be executed is no longer next line.
(exit, if then BLOCK, while BLOCK, break, continue, call, return, puase)
(Note 4) Although 'goto' can be executed, statements following the 'goto' statement will not be executed. Also, the 'goto' statement will be executed when the paused script execution is released.
- + To move window, mouse right button down and drag it in input area.
- + Pressing [CTRL] and [ENTER] in input area will execute specified "OneLine statement".
- + Pressing [CTRL] and [UpArrow][DownArrow] in input area, "OneLine statement" that was executed and stacked will be displayed in input area, allowing you to reuse it.
(Note 5) The [CTRL] pressed state will continue until you press any key other than [ENTER], [UpArrow], [DownArrow], or operate mouse.
- +  Execute "OneLine statement".
- +  Initialize saved "OneLine statement" stack.
- +  Ends "OneLine statement" processing.

(Note 6) "OneLine statement" does not have its own 'wait' state, shares the state with normal script execution. For example, in the above diagram, "reserve S1F22" means that if S1F1 is reserved in the original script, 'wait' in this "OneLine statement" will complete even if S1F1 is received.

(note 7) Even if you enter a 'new line' in input area, the 'new line' does not become a 'separator'. Even if you enter a 'new line', the 'statement' continues on the next line. To enter the next statement, ';' is required even at the end of line. If you want to continue a statement on next line, '¥' is not required.

(c) Communication trace display

It displays the trace of sent and received SECS communication messages, and also displays the execution statement of automatic execution script as specified ([Script trace display] in the [Display] menu).

(Note 1) The form of list format display of SECS communication messages is determined by following parameters of the .ini file.

- + TRCTTYPE ... Communication message output format to communication trace
 - bit#2 Item data display format
 - =0: Display each item on one line only and omit the back if it does not fit on one line.
 - 1: Each item is displayed on multiple lines, 20 items for numerical items and 100 bytes for character string items.
 - bit#4,5,6 .. List output format (Normally set to 2)
 - =0: TDS Format
 - 2: SML Format
 - bit#7 Data item name display (Normally set to 1)
 - bit#8,9 Message definition file format
 - (Normally set to 0 and prepare the message definition file in SML format)
 - =0: SML Format
 - bit#10 Number of leading spaces in list view
 - =0: 2 1: 0
 - bit#14 Multi-byte character item code representation format
 - =0: K 1: M99 (Add 'Local String Header' value)
 - bit#15 Display non-printable bytes in ¥377 format for ASCII items and JIS-8 items
 - (Note: M(K) items do not do ¥377 escape notation)
 - =0: No 1: Yes

(Note 2) The form of hexadecimal display of SECS communication messages is determined by following parameters of .ini file.

- + TRCTTYPE ... Communication message output format to communication trace
 - bit#3 Hexadecimal display format
 - =0: Display 16 Bytes on one line.
 - 1: Display 20 Bytes on one line.

(Note 3) The communication driver (TDS.dll) automatically issues S9F9 according to the setting ((DEVMODE&0x0100)!=0) of .ini when there is no receiving of secondary message for sending primary message and T3 timeout occurs. The SECS messages automatically issued by the communication driver are also displayed in communication trace display window.

(Note 4) For example, when a corresponding secondary message is received from opposite side after a T3 timeout occurs. If the setting of .ini does not receive an invalid secondary message ((DEVMODE&0x0002)==0), an invalid secondary message is not displayed in the communication trace display window at the time of received.

To also display these messages in communication trace display window, set the DEVMODE setting in .ini to ((DEVMODE&0x0002)!=0).

However, regardless of DEVMODE setting, all SECS communication messages are output to communication trace file.

(Note 5) The LinkTest message at the time of HSMS connection is not displayed on communication trace display window. Output to communication trace file in case of TRCTLEVEL>=9. If the extended communication trace display window is running, it is displayed when TRCTLEVEL>=9 in that window.

(Note 6) Communication control code when sending/receiving by SECS-1 connection is not displayed in communication trace display window. Output to communication trace file in case of TRCTLEVEL>=6. If the extended communication trace display window is running, it is displayed when TRCTLEVEL>=6 in that window.

(Note 7) The output to communication trace file depends on .ini TRCDIR, TRCTTYPE, TRCTOUT, TRCTLEVEL, and TRCTSIZE. Please refer to TDSE.pdf 2.1 (1) for details.

(d) Communication message editing

Display the contents of the specified message definition file.

The message to be sent on your side is displayed with the '!' Character at the beginning of message name.

Therefore, when [Message direction evaluation] is checked in the [Settings] menu, only messages with '!' Can be sent to opposite side. If not checked, all messages can be sent.

- + When you double-click on a message name, it will be sent to oppsite side with the contents of the items that make up currently set message.
- + The following menu can be executed by right-clicking on the message name.
 - [Send] : Same as double-clicking on a message name.
 - [Initialize] : Returns each item that composes a message to definition in message definition file.
- + You can change the data item values by opening message name and viewing the data items that make up the message. If you execute [Edit] of the menu displayed by double-clicking item name or right-clicking, "Edit SECS Message" dialog will appear, and edit the data item values.
 - For the variable quantity list item, enter [No. of Item].
 - Enter multiple numeric items separated by ','. When specifying a hexadecimal number for a numeric item, add 0x as a prefix. (Example: 12,0x3a, 4321,0xff)
 - For items configured with multiple items, both item value input and setting "No. of Item" input can be performed, and the number specified later will be valid.
 - String fields (A, J, M item) allow you to enter escape codes of the ¥377 format to enter non-printable byte values. However, ¥000 and ¥377 cannot be specified. The ¥377 notation indicates 1-byte value with 3 octal digits. ¥001 - ¥376 can be specified. (Example) "01¥00345" means 0x30, 0x31, 0x03, 0x34, 0x35 in hexadecimal notation. "

(Note 1) You cannot change SECS message structure itself (addition, deletion, etc. of data items) here. If it is necessary, change the message definition file itself and apply file change to screen by clicking reload button of message definition file.

In this case, please note that data item value set from screen is initialized to the value of file contents.

(Note 2) If the message has a variable count list, you must first determine the number of variable count lists. When a plurality of variable number lists are provided, the number of top level list is determined, and if the level is the same, the number of upper lists is determined, and the number of all variable number lists is determined. After that, set up other data items.

(Note 3) The message edited here is used to send message by double-clicking (or [Send]) message name, and to send message within an auto-execution sequence.

(Note 4) The send message constructed by send statement of automatic execution script is not reflected in this window.

(Note 5) [Start] initializes the message definition when simulation process is started. Even if you use the same message definition file as previous one, I will initialize with no regrets.

(Note 6) The current number of items in the character string items (A, J, M item) displayed on [Communication message edit screen] is the number of bytes when the setting character string is converted to the format to be sent to the other party. (It is not the number of bytes in the display string. Also, the message item count range value is not evaluated at this time.)

(Please note that the number of bytes displayed may differ from the number of bytes sent when including half-width katakana (JIS-8), Multi byte code, or ¥377 Escape characters.)

(Note 7) On the message item edit screen for text items, you cannot change the number of items, even for items with variable item lengths. The number of items for variable-length items is the transmission byte length calculated from the entered item string.

Regarding the character string item, [No. of Item] on this screen does not have much meaning. The number of bytes after conversion or the number of bytes of the input string is displayed as appropriate.

(Note 8) In "Edit SECS Message" dialog, **you can enter up to 4000 bytes as an input string.**

If you want to specify an item value that exceeds this size, set the item value in "Message definition file (.sml)" so that it is specified in "Item value data file (.dat)".

Please change it at any time by editing contents of "Item value data file (.dat)".

For details, please refer to TDS.pdf A. (1) (d) (Note 6) @FILEXXXX description.

(e) Auto run script

After selecting an auto-execution scenario and starting execution of auto-execution sequence, the current execution status is displayed.

It is shown by adding '>' at the beginning of the line to be executed next.

(f) Script variable

Displays the values of script variables and special variables used in automatic execution sequence. Script variables can change their values. For details on script variables and special variables, refer to Chapter 3 (5).

% (Dec) and % (Hex) is same integer script variable.

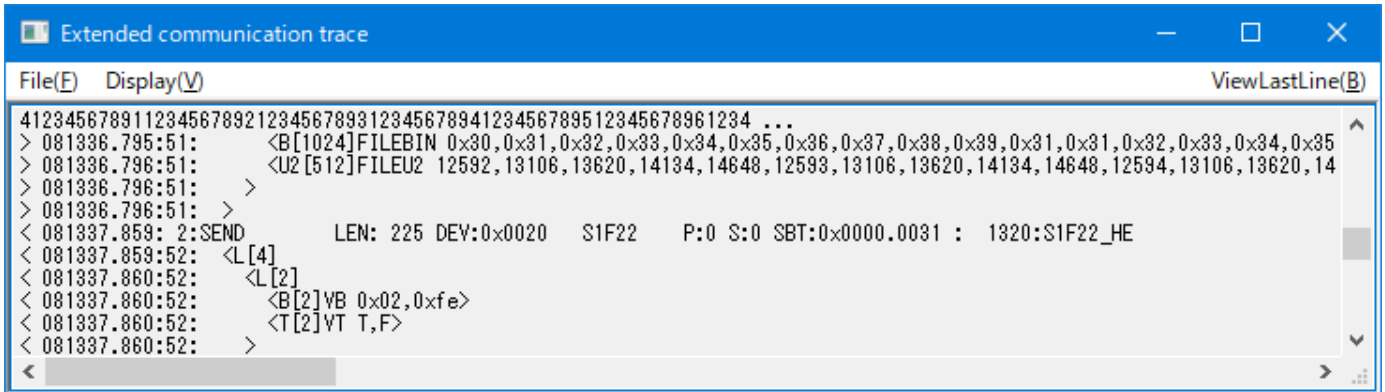
& is a real script variable.

\$ is a string script variable.

You can change the value of corresponding script variable by clicking "cell" of each script variable and directly keying in the value.

(Note 1) Special variables and script variables are initialized at start of execution of automatic execution scenario.

(g) Extended communication trace window



In the window displayed when [Extended communication trace display] is specified in (a-2), the contents output to the communication trace file set in the specified .ini are displayed in real time.

Note the parameter values for .ini file token below.

+ TRCTTYPE : Communication message output format to communication trace

+ TRCTLEVEL : Communication trace output level

The parameter values for these tokens can be changed while simulation is running.

For more information on parameter values, see the description of related tokens in 2.1 (1)(c) in TDSE.pdf. It takes a certain amount of time (INTER3) for the changed parameter value to be reflected in the operation of tdlSSim.

(g-a) Menu

(g-a-1) [File]

+ End of display Closes extended communication trace display window.

(g-a-2) [Display]

+ Display trace With the window open, specify whether to display the trace.

+ Clear display Clear the window contents.

+ View last line Set the scroll bar to the state where the last line is displayed. (Used when final line does not easily appear in normal scroll bar operation, etc., with auto scrolling at high speed, etc.)

+ Display target In the Cascade Menu, select the type of trace message you want to display. Shows checked items.

(Note) **Items for which communication trace output is not set for TRCTOUT and TRCTLEVEL in .ini file will not be output as traces even if they are displayed here.**

(g-a-3) [ViewLastLine] Set the scroll bar to the state where the last line is displayed. (Same as (g-a-2) [View last line])

(Note 1) If transmission/reception processing is fast, trace screen display processing load is high and display cannot keep up. In such a case, please reduce the load by following means.

+ Reduce the number of display items by selecting [Display target].

+ Set the .ini TRCTTYPE setting value to (TRCTTYPE&x0080)==0, and do not display message name and item name using the message definition in communication trace output.

3. Script language specification

Describes how to write scenario execution script file for automatic execution of communication processing in this program.

(0) Overall structure

Multiple execution units can be described in one scenario execution script file (.ssl file). During automatic execution, only one of the execution units described in selected script file is selected to operate. The operation can operate the written statement continuously or one step at a time.

- + One execution unit starts with "program" statement and ends with "end" statement.
- + If there is "/" notation in the line, it will be treated as a comment after that.
- + If the end of the line is '¥', it is treated as continuing to the next line. However, the length of the entire line including continuation line is up to 4000 bytes.

(Note 1) If "/" exists before the continuation symbol, the comment is ignored and continued on the next line.

For example ... the next 2 lines

```
%1=123    // This is a comment ¥
```

```
%2=456
```

will be "%1=123 %2=456".

(Note 2) This script assumes that "good-meaning users" use them "right". Even if there is an incorrect description of the grammar etc., it will process as it is, without making an error report etc. in particular.

(Note 3) By setting a value to the special variable !TS, it is possible to delay the execution speed of each line of script, and it is possible to make the process contents confirmation of the script (slightly) easier. Please refer to (5) "set" for details.

(Note 4) When this script operates in automatic response mode, basically all messages that may be received should be defined in the message definition file (.sml). However, for the receiving of specified SF-Code, the following secondary message can be sent regardless of message structure by following method.

- + Define default secondary messages in message definition file.

For details, please refer to A. (1)(b) of TDSE.pdf described in Chapter 2 (0).

- + Use scenario execution script.

The scenario execution script can reply to the specified secondary message regarding receiving of the specified SF-Code, regardless of the message structure. The receiving process of scenario execution script is not performed for defined message structure, but is performed for the specified SF-Code. Please refer to AnyS5F1_S6F11 of attached sample script (SampleE.ssl) for actual script writing method etc.

(Note 5) This script may not be able to fully handle "Multi Open Transactions" (although it is not that it does not support them). If you have problems with this, you should wait for receiving of secondary message corresponding to previously sent primary message before sending primary message used in the script.

This script consists of the following statements.

- (1) program (prog) Declare the start of the run unit
- (2) function (func) Declare the start of a function (subroutine)
- (3) end Declare end of execution unit and function
- (4) exit Execution end
- (5) set {Optional} Assignment to script variable, operation
- (6) if Conditional branching
- (7) while Repeat control
- (8) break Exit from the current while block
- (9) continue Move to the end endwhile of the current while block
- (10) goto Move processing to specified label line
- (11) call Execute specified function
- (12) return End the execution of the function and return to the calling position
- (13) exec Execution of external program
- (14) reserve (resv) Waiting for receiving SF-Code specified for processing by script
- (15) cancel Cancellation of waiting for receiving SF-Code
- (16) wait Waiting to receive the SF-Code message specified in "reserve"
- (17) release (rels) Release received SECS message area
- (18) receive (recv) Parsing received SECS messages
- (19) send Send specified SECS Message
- (20) sleep Sleep for specified time
- (21) display (disp) Output script variable to communication trace
- (22) print Output specified string to communication trace
- (23) pause Move to STEP mode (execute one line at a time)
- (24) trace ON/OFF switching of script execution trace display

(1) program (prog) Declare the start of the run unit

```
program XXXXX    or    prog  XXXXXX
~~~~~
```

XXXXXX : Program name

From here, the scenario execution sequence of this name begins.

When a program name of sequence to be executed is specified at the time of script execution, a sequence of program names matching the name is executed.

The end of a series of program execution units must end with an END statement.

(Note 1) The program name must consist of uppercase letters, lowercase letters, numbers, and '_' (underscore), with no numbers as the first character, and a maximum of 30 characters.

(2) function (func) Declare the start of a function (subroutine)

```
function XXXXX    or    func  XXXXXX
~~~~~
```

XXXXXX : Function name

From here, the sequence of this function name begins.

A function is transferred from another by CALL statement, and the process moves to the next of the CALL statement by return statement or the last END statement in function.

The end of a series of function execution units must end with an end statement.

(Note 1) The function name must consist of uppercase letters, lowercase letters, numbers, and '_' (underscore), with no numbers as the first character, and a maximum of 30 characters.

(3) end Declare end of execution unit and functio

```
end
~~~
```

(4) exit Execution end

```
exit
~~~~
```

(5) set {Optional} Assignment to script variable, operation

[set] par=[=]exp [par=[=]exp [...]] (Note 1) The keyword "set" is optional

~~~~~

par= exp : Assign constants, script variables and their calculated values to specified

par==exp : script variables and some script constants.

Assignment operator == is meaningful only when both left and right sides are string type. If the right-hand side expression is only string script variable that does not have an operator, or only string constant, the script variable value or constant value is assigned as it is.

In other cases (when right-hand side performs an operation and assignment operator is =), script variable value in right-hand side character string is converted to that value and assigned.

Script variables are as follows, and any variable can be specified in any item.

\$00 .. \$99 : String variable

%00 .. %99 : Integer variable

&00 .. &99 : Real variable

(Note 2) The prefix unary operator "-" can be used for integer variables and real variables.

The notation such as %[0] or \$[%2] is also possible. %[0] is the same as %00, and \$[%2] is the index value of the \$variable as the %2 value at the time of evaluation.

(Note 3) Only %variables can be specified in []. Also, expressions such as [%2+1] cannot be specified.

The following can be specified as script constants. For script constants, only simple assignment is possible (operation assignment is not possible).

!TW : Timeout period from receiving SECS message to script performs receiving process completes reception-related processing. (Specified in milliseconds, but resolution is in seconds).

The assignment to !TW takes effect from the next time he receives a SECS message.

!TS : Execution delay time (mili seconds) of each script execution line.

Specifying a value >0 during debugging etc. makes script execution speed slower and makes it easier to follow the operation.

(Reference) If you write, for example, !TS=%0 in the script loop, you can adjust script execution speed by setting %0 value from the screen, which makes it easier to test the script. (Refer to CallbackIo\_H and \_E of SampleE.ssl)

The following special variables can be specified for 'par'. Special variables can only be assigned by simple assignment (operation assignment is not possible).

!DX : Device ID to be assigned to next transmission primary message

The following can be specified as 'exp'.

Constant : "String", Number

You can embed script variables in "strings" that you assign to string variables (\$xx). When describing script variables and special variables themselves

(eg \$09, !ER), put the same character before script variables etc. and escape.

(For example, describe as \$\$09, !!ER.) Also, if you include in "string", escape it with ¥.

(For example, if you write \$1="¥"ABC¥" - ¥"XYZ¥"", \$1 becomes "ABC" - "XYZ")

Hexadecimal numbers are specified in 0xff format.

Variable : \$9、%9、&9

<< Continue to next page >>

<< Continue from previous page >>

Special variable :

|                                                                         |                                   |      |
|-------------------------------------------------------------------------|-----------------------------------|------|
| !TM : Current time                                                      | (Total seconds)                   |      |
| !YY : Current years lower two digits                                    | (YY : 0 - 99)                     |      |
| !MD : Current month                                                     | (MM : 1 - 12)                     |      |
| !DD : Current date                                                      | (DD : 1 - 31)                     |      |
| !HH : Current hour                                                      | (HH : 0 - 23)                     |      |
| !MM : Current minute                                                    | (MM : 0 - 59)                     |      |
| !SS : Current second                                                    | (SS : 0 - 59)                     |      |
| !MS : Current mili second                                               | (MS : 0 - 999)                    |      |
| !T3 : T3 Timeout value                                                  | (mili second)                     |      |
| !TS : Script execution delay time                                       | (mili second)                     |      |
| !TW : Script receiving processing Timeout value                         | (mili second)                     |      |
| !ER : SECS send/receive execution error code                            |                                   | (*1) |
| =0:OK 1:Receive error 2:Send error 3:Wait Timeout                       |                                   |      |
| !DX : Device ID to be assigned to the next transmission primary message |                                   |      |
| !DV : Final received Device ID                                          |                                   |      |
| !SC : Final received S-Code                                             | ( 8bit)                           |      |
| !FC : Final received F-Code                                             | ( 8bit)                           |      |
| !SF : Final received SF-Code                                            | (16bit S-Code*0x0100 + F-Code)    |      |
| !CD : Final received SF-Code                                            | (S99F99 Format string)            |      |
| !MN : Final received Message name                                       | (Name in message definition file) |      |
| !SD : Final received Source-ID                                          | (16bit)                           |      |
| !XD : Final received Transaction-ID                                     | (16bit)                           |      |
| !SX : Final received System-byte                                        | (32bit) (!SD*0x00010000 + !XD)    |      |

(\*1) !ER : Value displayed on script variable screen is in form of 99(99) and indicates current value (last error value).

Formula : Variable1 Operator Variable2

Variable1 and variable2 may be constant

Variable1 may be special (integer type) variable

Variable1 must be of the same type as 'par'

Operator is any of the following for each type of 'par'

String variable : +

Integer variable : +, -, \*, /, ^, %, &, |, <<, >>

Real variable : +, -, \*, /, ^

(Note 4) Only one operation can be used in an expression. Multiple operations cannot be used as one expression.

|           |                                  |                             |                      |  |
|-----------|----------------------------------|-----------------------------|----------------------|--|
| (Valid)   | %12=%11*2                        | %12=%12+1                   | &16=&[%12]*&10       |  |
|           | %16=16                           | \$16="&&16 = &[%16] on %16" | \$17="&&16 = " + &16 |  |
| (Invalid) | %12=%11*2+1                      | &16=&11*&10*2.0             |                      |  |
|           | \$16="&&16 = " + &16 + " on %16" |                             |                      |  |

( 6) if ..... Conditional branching

```
if exp [then] statement          // 'then' can be omitted
~~~~~                          // (Note 1) In this case, you can not put elseif and else
 // next to construct a block if.
```

or

```
if exp0 then
 statement block if 'exp0' is true
elseif exp1 then // elseif block is optional
 statement block if 'exp1' is true
else // else block is optional
 Statement block when all if and elseif
 conditional expressions are false
endif
~~~~~
```

exp : Comparison operator or constant (true or false)

Expression1 Comparison-operator Expression2

The following can be specified as 'Expression1'

Variable : \$99, %99, &99

Special variable :

|                                                                         |                                   |
|-------------------------------------------------------------------------|-----------------------------------|
| !TM : Current time                                                      | (Total seconds)                   |
| !YY : Current years lower two digits                                    | (YY : 0 - 99)                     |
| !MD : Current month                                                     | (MM : 1 - 12)                     |
| !DD : Current date                                                      | (DD : 1 - 31)                     |
| !HH : Current hour                                                      | (HH : 0 - 23)                     |
| !MM : Current minute                                                    | (MM : 0 - 59)                     |
| !SS : Current second                                                    | (SS : 0 - 59)                     |
| !MS : Current mili second                                               | (MS : 0 - 999)                    |
| !T3 : T3 Timeout value                                                  | (mili second)                     |
| !TS : Script execution delay time                                       | (mili second)                     |
| !TW : Script receiving processing Timeout value                         | (mili second)                     |
| !ER : SECS send/receive execution error code                            |                                   |
| =0:OK 1:Receive error 2:Send error 3:Wait Timeout                       |                                   |
| !DX : Device ID to be assigned to the next transmission primary message |                                   |
| !DV : Final received Device ID                                          |                                   |
| !SC : Final received S-Code                                             | ( 8bit)                           |
| !FC : Final received F-Code                                             | ( 8bit)                           |
| !SF : Final received SF-Code                                            | (16bit S-Code + F-Code)           |
| !CD : Final received SF-Code                                            | (S99F99 Format string)            |
| !MN : Final received Message name                                       | (Name in message definition file) |
| !SD : Final received Source-ID                                          | (16bit)                           |
| !XD : Final received Transaction-ID                                     | (16bit)                           |
| !SX : Final received System-byte                                        | (32bit) (!SD*0x00010000+!XD)      |

<< Continue to next page >>

<< Continue from previous page >>

The following can be specified as 'Expretion2'

Constant : "String", Number (Hexadecimal is 0xff format)

Variable : \$99, %99, &99

The following can be specified for each form of 'Expression1' as a comparison operator

String : ==, !=, <, <=, >=, >

Integer : ==, !=, <, <=, >=, >

Real : ==, !=, <, <=, >=, >

(Note 1) This "Note 1" is invalid.

(Note 2) The prefix unary operator "--" can be used for integer variables and real variables in both 'Expretion1' and 'Expretion2'.

label : Label for jump to (Refer to 'goto')

statement : Executable statement

(Note 3) If you write a block 'if' statement, you can nest 'if' within a statement block.

(Note 4) There is no space between the 'else' and 'if' in the 'elseif' statement and between the 'end' and the 'if' in the 'endif' statement.

(Note 5) Upward 'goto' cannot cross the hierarchy of if block and while block.



( 7) while ..... Repeat control

```
while exp do
  "A statement block that executes repeatedly while 'exp' is true"
endwhile
~~~~~
```

exp : Comparison formula (Refer to (6) 'if' statement)  
Specifying "true" as 'exp' results in an infinite loop.

(Note 1) Nesting of while blocks can be configured up to 30 levels in the entire execution program (including in functions by CALL statements).

(Note 2) We do not assume that if -- endif blocks and while--endwhile blocks get worn. Even if the block configuration is wrong, no error indication occurs.

(Example) The following is an incorrect configuration. But do it as it is.

```
if true then
 %2=0
 while %2==0 do
 "Zzzzzz...."
 endif
endwhile
```

( 8) break ..... Exit from the current while block

```
break
~~~~~
```

( 9) continue ..... Move to the end endwhile of the current while block

```
continue
~~~~~
```

(10) goto ..... Move processing to specified label line

```
goto label
~~~~~
```

label : Jump destination label  
The label is described as "XXXXX:".  
**Do not put an executable statement on the label line.**  
Label lines must be within the currently executing processing unit (program -- end or function -- end).  
Also, **upward "goto" cannot cross the hierarchy of if block and while block.**

- (11) call ..... Execute specified function

```
call func_name
~~~~~
```

func\_name : Name of called function

- (12) return ..... End the execution of the function and return to the calling position

```
return
~~~~~
```

- (13) exec ..... Execution of external program

```
exec [+w] "program [parameters...]"
~~~~~
```

+w : When this option is specified, the specified program is started with window displayed. If not specified, the specified program has no window.  
(Note 1) This option is valid only for AP that have access to standard input/output without windows.  
In case of AP with a window, window is displayed regardless of designation.  
If AP does not have a window and does not have access to standard input/output, window will not be displayed.

program : External program to launch (Path name of .exe)

parameters : Arguments to pass to 'program'. Like print statements, it can contain script variables and special variables.  
Also, if you add '&' at the end (before closing), processing will be continued without waiting for the specified program to finish.

(Reference) The program started by 'exec' statement creates the contents of SECS message item according to run time argument and records the result in file such as "secs\_item.dat". In the subsequent 'send' statement, you can create SECS message to be sent as "send SxFx\_xx @file=secs\_item.dat,section" for specifying SECS message item. Refer to 'send' statement.

(14) reserve (resv) ..... Waiting for receiving SF-Code specified for processing by script

```
reserve SxFx [SxFx] ... or resv SxFx [SxFx] ...
~~~~~
```

SxFx : SF code for receiving reservation ((Note) Not a message name)  
Write S0F0 to wait for all messages.

(Description) Reserve receiving messages for script processing. The specified SF-Code is reserved in addition to other already reserved SF-Codes. If the specified SF-Code is already reserved, nothing will be done.

If receive an SF-Code that does not 'reserve' for receiving, in the case of a primary message, return a secondary message by automatic response, and in the case of a secondary message, receive it as it is.

When receiving the SF-Code that was reserved, wait for message using 'wait' statement. Then, if necessary, received message is analyzed using the 'receive' statement, or if received message is a primary message, it is performed using 'send' statement of corresponding secondary message.

After that, if you do not want to perform the same SF receiving next time, use 'release' statement to cancel the reservation.

If you want to wait for receiving again with SF-Code that has been released, reserve it again using the 'reserve' statement.

(Note 1) The 'wait' statement detects that process related to the message receiving (message analysis, saving message item values, sending secondary messages, etc.) is completed, and start next message receiving process. Therefore, if a script executes many statements or do 'sleep' statement after processing for a received message is completed before executing the next 'wait' statement, the receiving processing may be delayed.

In that case, **by specifying received SF-Code as a processing target again using 'reserve' statement, the next receiving process will start in background, and messages with SF-Code that have not been reserved for receiving can be automatically received and processed.**

**When receiving an SF-Code message that has been reserved for receiving, the receiving process will stop until the script starts receiving process by executing 'wait' statement.** However, if the next message is received in background, the values of special variables such as !SF may change, so processing such as secondary message reply due to receiving of primary message must be completed before issuing 'reserve' statement.

(Note 2) See also explanation of the 'wait' and 'release' statement.

(Note 3) For example, in the case of sequence of receive S1F13, send S1F14, receive S1F1, send S1F2, it shows the case where S1F1 sending on the opposite side is quick after sending of S1F14. If you are not ready to receive this S1F2, you may not be able to receive S1F2 in the script. Therefore, if the opposite side sending/receiving for my side sending/receiving is quick, it is necessary to perform 'reserve' processing of the next received message prior to sending processing.

That is, in the case of the above sequence, it is as follows.

```
reserve S1F13      // Reservation to receive S1F13
reserve S1F1      // Before S1F14 send, perform the next S1F1 reserve

wait              // Waiting for receiving S1F13, S1F1
receive S1F13     // Not necessarily necessary
send S1F14_XX     // Reply S1F14 (However, S1F1 receiving is not considered)
release S1F13     // Release received information of S1F13
                  // (Release after sending the secondary message)

wait              // Waiting for receiving S1F1
receive S1F1      // Not necessarily necessary
send S1F2_XX     // Reply S1F12
release S1F1      // Release received information of S1F1
```

(15) cancel ..... Cancellation of waiting for receiving SF-Code

```
cancel SxFx [SxFx] ...    or    cancel all
~~~~~
```

SxFx : Awaiting receiving SF-code to cancel ((Note) Not a message name)  
all : Cancel all pending SF-code.

(Description) Cancel waiting for receivingg messages to be processed by script.

Delete the specified SF-Code from the reserved list. When 'release' statement is executed on received message, the SF-Code of that message is automatically deleted from reservation list. Therefore, there is no need to execute 'cancel' statement for that SF-Code. When waiting for receiving with same SF-Code again, 'reserve' statement must be executed again.

(16) wait ..... Waiting to receive the SF-Code message specified in "reserve"

```
wait [msec]
~~~~~
```

msec : Timeout value for releasing the wait state (mili seconds)  
In case of omission or msec=0, timeout does not occur.  
Script variables and special variables (!T3) can be used.

(Description) Wait until the message of the SF-code reserved by 'reserve' statement is received.

The information of the SECS message received by the wait statement is stored in the special variable at this time.

When a timeout occurs, !ER==3.

The script knows the completion of process related to received message by next 'wait' statement or re-registering received SF-Code using 'reserve' statement, and starts next message receiving process.

It is not always necessary to re-register the received SF-Code using 'reserve' statement.

(Note 1) **It must be within 10 seconds from the time processing returns to script due to SECS message receiving from 'wait' statement until receiving process of SECS message is completed by next 'wait' or 'reserve' statement. (Can be changed using set !TW=20000 etc. See 'set' statement)**  
**Be especially careful when executing STEP.**

(Note 2) See also explanation of the 'reserve' and 'release' statement.

(17) release (rels) ..... Release received SECS message area

```
release SxFx [SxFx] ...      or      rels SxFx [SxFx] ...
~~~~~
```

SxFx : SF code to be processed

((Note) It is not a message name. specified in reserve statement)

(Description) Cancel receiving reservation made by 'reserve' statement of specified SF-Code.  
Execute after completing all processing related to received message, such as parsing message received with 'wait' statement using 'receive' statement, acquiring message item values, saving special variables, and sending secondary messages.

**(Note 1) If you want to continue receiving received SF-Code, do not use 'release' statement. If you do not want to receive the SF-Code, you must use the 'release' statement to release received SF-Code.**

Refer to the following (Example 1), (14) reserve statement, and (16) wait statement.

(Note 2) After 'release' statement is executed, the contents of special variables (such as !ER) are not guaranteed.

That is, after the wait statement, for example, processing such as branching processing by the value of !CD can be performed, but once 'release' statement is executed, processing such as determination can not be performed thereafter.

When using in a script after the 'release' statement, assign the value of the special variable to the script variable before executing the 'release' statement.

(Example 1) For example, in the case of sequence in which S6F11 is continuously received and S6F12 is returned, it is as follows.

```
reserve S6F11 // Reservation to receive S6F11
while true do // Infinite loop
 wait // Waiting for receiving S6F11
 receive S6F11 // Not necessarily necessary
 send S6F12_XX // Reply S6F12
 if "It's over" break
endwhile
```

(18) receive (recv) ..... Parsing received SECS messages

```
receive SxFx [scv=item [scv=item ...]] or recv SxFx [scv=item [scv=item ...]]
~~~~~
```

SxFx : SF code to be processed

(Note) It is not a message name. SF-Code specified in 'reserve' statement.

(Even if a name is specified, only SF-Code part is used.)

Select and analyze message that matches last received communication message structure from among multiple definition messages of specified SF-Code.

You can also specify !CD and !MN

scv=item : Get data value from received message, and instruct assignment to script variable.

scv (script variable) has the following, and any item and any variable can specify message items of all types.

\$00 .. \$99 : String variable

%00 .. %99 : Integer variable

&00 .. &99 : Real variable

'item' is item name set in message definition file, and in the case of an indefinite number of items, specify item number such as "item:9" (9 is 1 or more item number, and ':' is a colon)

Also, in the case of a numeric item, if there are multiple item values, and if it is not the first item, specify in format "item:9" (9 is a 0 or more in-item subscript value, and ';' is a semicolon).

For example, it describes as follows. If the description becomes long, put '¥' at the end of line to continue to next line.

In addition, it is possible to divide and describe in multiple lines.

```
receive S2F49 $1=RCMD $2=LOTID %1=STKD %2=IDATA:1;7 ¥
               &2=RDATA:3
```

-- or --

```
receive S2F49 $1=RCMD $2=LOTID
receive S2F49 %1=STKD %2=IDATA:1;7 &2=RDATA:3
```

(Note 1) If the last received message is the specified SF-Code, perform processing on the message of that SF code.

Nothing is processed for messages other than the specified SF-Code.

(Note 2) You must execute 'wait' statement before 'receive' statement.

'receive' statement does not have to be executed.

Refer to (14) reserve statement, and (16) wait statement.

(Note 3) The 'receive' statement corresponding to one message receiving in 'wait' statement can be executed multiple times.

If there are many message items to be processed, it may be divided into multiple steps.

(19) send ..... Send specified SECS Message

```
send SxFx_xx [item=value [item=value ... [@file="data_file.dat[, sect]] ...]]  
~~~~~
```

SxFx\_xx : Message name set in the message definition file

(Note) It is not SF-Code.

For example, specify a message name such as 'S6F11\_STATUS'.

It is also possible to incorporate a variable as 'S6F11\_STATUS\_%1'. Of course, the message name resulting from evaluating the variable must be defined in message definition file.

item=value : Set values for the items that make up specified message.

The message data to be used is the currently set content with the value specified in this option overwritten. The contents set here are not reflected in the original message data.

Unfixed number list item : item=99

String item : item="xxxxxxxxx"

Numeric item : item=99 or =0xff or =99.999 or =9.999e999

Script variables and special variables can be specified as 'value'.

Refer to 'set' statement for script variables and special variables.

Any type variable can be specified for any item.

When setting a string constant in a string item, script variables and special variables can be embedded in the string constant. For example, you can specify "ABC %1 XYZ !!TM = !TM" for a string item. (Refer to 'set' statement)

For example, it describes as follows. If the description becomes long, put '¥' at the end of line to continue to next line.

```
send s2f49_hload RCMD="LOADDATA" LOTID="This is LOTID" ¥
PORT=2 MTKD=%5 ¥
PDATA="%12,%13,%14,%15" RECIPE=$1 CASSETE=$3
```

(Note 1) If there is an undefined number list in the specified message, specify the number of all undefined number lists in order from the upper list item, and then specify the normal item.

When the number of lists is fixed, the item names in the list change according to the rules.

Please refer to TDSE.pdf described in Chapter 1 for the rule of item name at the time of fixed quantity list fixed.

(See the notes and examples in \_TDSMDMssgBuild().)

(Note 2) When specifying multiple numeric item values, enclose entire value with "" and specify as item="1,2,%12,%13,99".

(Note 3) It is also possible to specify only a part of the item value.

Therefore, it is possible to specify "item:3=12,13,%5,%6" using the format as in 'receive' statement.

When setting non-consecutive data, the same item can be specified multiple times, but it must be specified continuously.

For example, the item ITEM having 10 data values will be described. When you set ITEM[3]=3, [5]=5, and [6]=6, you specify as ITEM:3=3 ITEM:5="5,6".

<< Continue to next page >>

<< Continue from previous page >>

@file="data\_file.dat,section" : Specify the values of items that make up specified message in file (item value setting file).

data\_file.dat : File path name with item value specified  
                    If you specify a relative path name, it will be a relative path from  
                    the directory where the script file (.ssl file) exists.  
section : Section name to be used in the specified file

The item value specification file has the following .ini file format configuration.

```
// Comment

// First, specify the field value to be used when 'section' is not specified.
RCMD = "LOADDATA" // Data type
PORT = 3 // Port number

// Next, specify the item value when 'section' is specified.
[SECT01]
RCMD = "LOAD#1" // First data type
PORT = 2 // First port number

// In the following, write in the same way.

// For example, when using the setting of SECT01, if this file is "sample.dat",
// write as below.
// send s2f49_hload @file="sample.dat,SECT01"
```

(Note 4) The deviceID added to the sending primary message uses the value set in !DX.  
If the !DX value is not specified, the first deviceID specified in DEVID of configuration file (.ini) is used. The deviceID assigned to the sending secondary message uses the deviceID (!DV) assigned to the primary message received immediately before.

(Note 5) !ER=2 is set when a transmission error occurs.



- (20) sleep ..... Sleep for specified time

```
sleep msec
~~~~~
```

msec : Wait time (mili seconds)  
It is possible to use script variables.

- (21) display (disp) ..... Output script variable to communication trace

```
display [scp] or disp [scp]
~~~~~
```

scp : Script variable  
If omitted, all script variables and special variables are displayed.

- (22) print ..... Output specified string to communication trace

```
print string
~~~~~
```

string : Print output string  
You can include \$99,%99,&99,!ER, etc. in the string.

- (23) pause ..... Move to STEP mode (execute one line at a time)

```
pause
~~~~~
```

Switch to Conversation (STEP) mode.

- (24) trace ..... ON/OFF switching of script execution trace display

```
trace {on|off} [{on|off}]
```

on|off (1st) : Update running script display window  
on|off (2nd) : display in trace output window  
If omitted, it is considered to be same as first specification.

(Note 1) When the script starts, it always starts in the 'on' state.