bop ActiveX Control
Users Manual

Jazz Soft, Inc.

1 Revision History

Rev.	Rev. Date	Rev. By	Description
1.00	Sep,5th,2005	Hikaru Okada	Newly Created
1.00a	Nov,18th,2005	Maryanna Malelu	Translated to English
1.00b	Nov,20th,2005	Hikaru Okada	Modified multi-bytes characters

2 Introduction

Traditionally it has taken extremely long periods of time and high costs to develop GEM-compliant equipment. Called a “Development License”, it has become a standard industry practice to charge prices that at first glace seem almost unlawful, reaching into the tens of thousands of dollars for the initial investment and tens of thousands more every year as a maintenance contract fee. In addition, the diffiulty of comprehending the requirements, and the difficulty of performing the development work itself, has led to the current situation, in which a “package deal” terms of sale commonly require the assignment of several development personnel, at man/month costs in the teens of thousands of dollars per worker. However, let us consider whether GEM compliance is truly as complicated as to require such expenditures.

Jazz Soft has developed and offered its Swing Series products with a new approach to these questions. In order to achieve GEM compliance even more readily than with Swing, we offer our newest product, bop ActiveX Control. A significant number of years and resources have gone into developing both Swing and bop, and although they are not yet self-sustaining products, it is our firm belief that this can be improved with further effort in business enterprising, depending on the point at which we will cross the line into low-priced sales.

Bop is GEM development software, but can also accommodate GEM300. Although this version does not directly support GEM300, full attention has been given to provide user-added functionality to expand its uses. Of course, GEM and GEM300 systems can be structured using only Swing, but such applications require significant functionality to be added by the user. Bop is a software version featuring a very reduced necessity for such time expenditures.

The fundamental principles of Jazz Soft are:

① Never a fee for version upgrades
② No need for development licenses
③ No charge for maintenance contract

These are our policies, to which we will adhere both now and in the future. Our focus will continue to be providing higher functionality products at low cost, so be sure to keep watching for exciting future products!

Jul,12th,2005
Jazz Soft, Inc.
Chief Operating Officer (COO)

Hikaru Okada

3 Table of Contents

4 Usage Environment

4.1 Development/Operating Environment

Bop can be used in the following environments.

O/S	Windows 98(USB version), Windows Me, Windows 2000, Windows XP Professional, Windows XP Home Edition and Windows Server 2003. Regarding Windows NT 4.0 and Windows 95, bop cannot be used with the USB version but can be used with the printer port version.
Development Language	Active X 32-bit development languages such as Microsoft Visual Basic 6.0, Visual C++ 6.0, Visual Basic .NET, Visual C++ .NET, C# .NET, Borland Delphi, C++ Builder, etc.

4.2 Combined Use of swing

With a bop HASP key, it is possible to operate swing and Sexy as the product versions.

Function	bop	swing
SwingSecsI	✓	✓
SwingSecsII	✓	✓
SwingHsms	✓	✓
SwingComm	✓	✓
SexyM	✓	✓
bop	✓	-

5 Installation

5.1 Preparing Installation CD

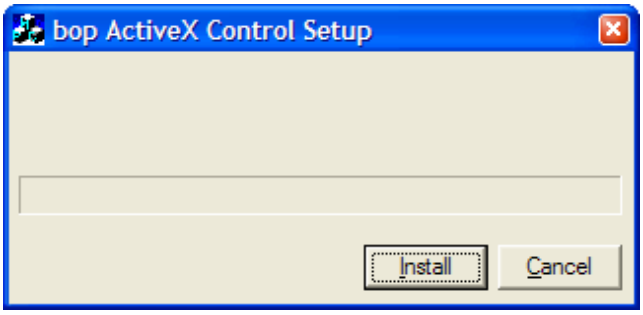
When the Jazz Soft installation CD is placed in a PC, the following screen will be displayed. If it is not displayed, double-click the file named default.htm in the CD's root folder.

Bop and swing can be installed from this CD. To install bop, click "bop Version 1.00".

The contents of the bop setup folder will be displayed as shown above.

5.2 Executing the Installer

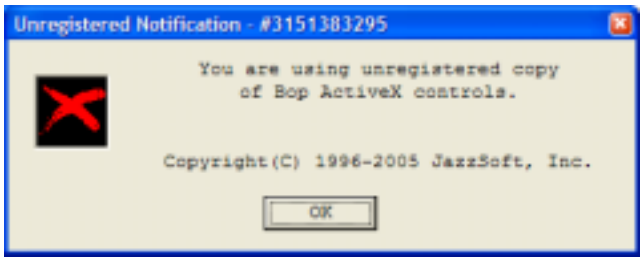
Double-click setup.exe from the bop setup folder to execute the installer.



When the Install button is clicked, installation will begin. When complete, the following message box will be displayed, indicating that installation is complete.

5.3 Difference Between Trial Version and Product Version

Merely installing bop will allow only the trial version to be operated. There are no particular differences between the trial version and product version, but the trial version will show the following dialogue box on a frequent basis.

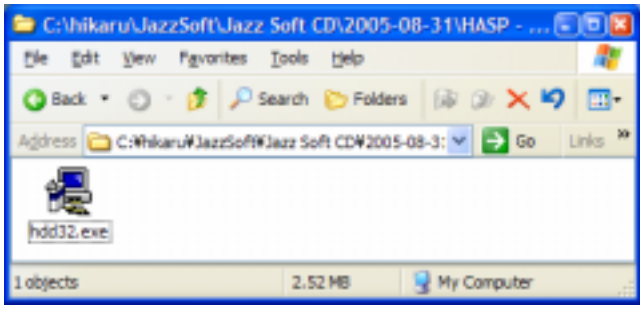


Since there are no restrictions in use of the trial version, it is possible to do the same operation checks as with the product version.

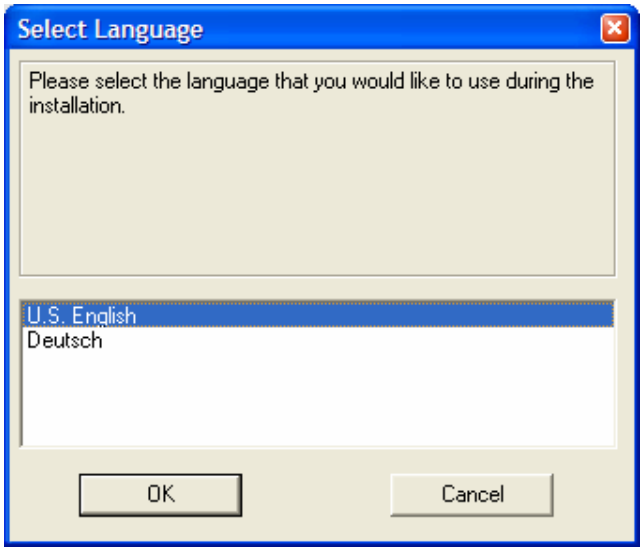
5.4 Installing the HASP Driver

When a product version of this product is purchased, it is accompanied by a HASP key. Installing this in your PC will operate the software as the product version. Please note that it is necessary to install the driver first, before inserting the HASP key.

Click HASP Driver 4.95 in the beforementioned Jazz Soft installation CD.

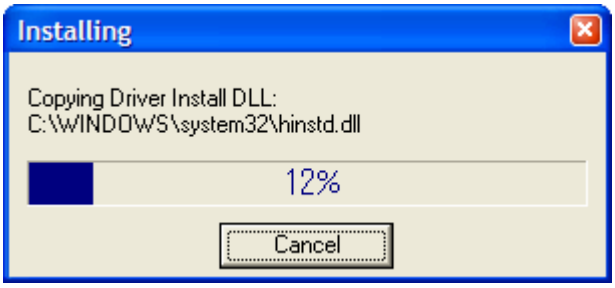


Double-click hdd32.exe. When the following screen is displayed, select "U.S. English" and press the OK button.



Press the Next button.

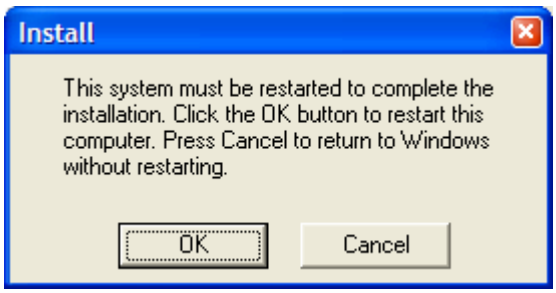
Copying of the driver will start. At this time, the files necessary for installation have only been copied.



When preparations for installation have been completed, the following screen will be displayed. Press the Next button. It may take several minutes for installation to occur.

When installation is completed, the following screen will be displayed. Press the Finish button.

In some versions of Windows, a dialogue box such as that shown below, recommending rebooting, may be displayed. In such cases press OK and reboot.



6 Tutorial

In order to demonstrate the ease with which bop can be usedto create GEM-compliant equipment, let's try creating a simple equipment. The following will be the specifications for the equipment to be created in our tutorial.

■GEM Event

CEID	Description
100	Online to Offline
200	Carrier Loaded
201	Carrier Unloaded

■Variables

VID	Variable Type	Model	Description	Min.Value	Max. Value	Default	Unit
10	EC	U2	EC Timer	0	600	30	sec
20	EC	U2	Time Format	0	1	1	
30	SV	U2	Ctrl State				
40	DV	Ascii	Carrier ID				

■Alarms

ALID	Description
30000	Alignment Failure

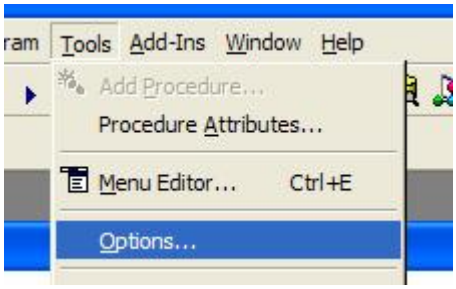
6.1 Visual Basic Version 6.0

6.1.1 Creating a new Project

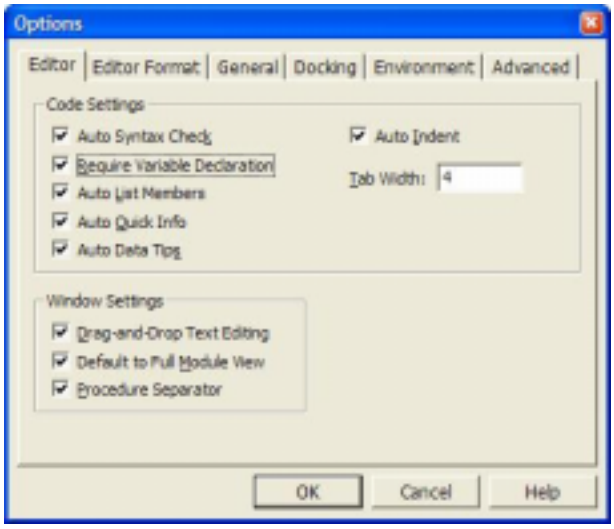
When Visual Basic 6 is launched, the following dialogue box will be displayed. Check to make sure that "Standard EXE" is selected and then press the Open button.

6.1.2 Requiring Variable Declarations

In default status, Visual Basic can be used without declaring variables. This is so because it was that way in the old BASIC language specifications. If development is performed in this status, it is easy to fall into problems with tangled program strands, so here we force the declaration of variables. Select "Tools" – "Options" from the menu.



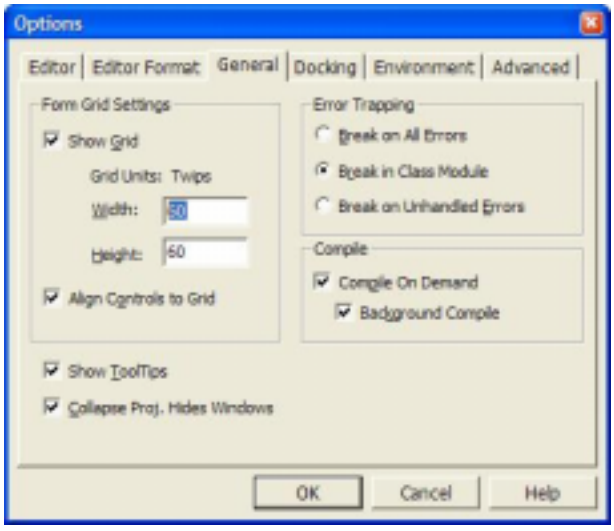
Check the checkbox for "Require Variable Declaration"in "Code Settings" from the "Editor" tab. This will make it so that an error will occur if the program is used without explicitly declaring variables.



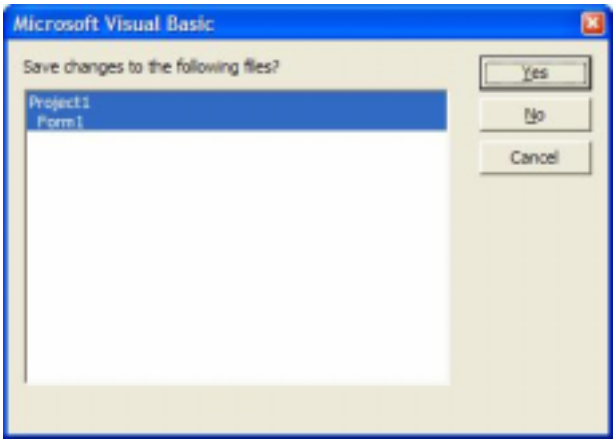
When this option is enabled, an "Option Explicit" declaration will precede the source code as shown below.

```
Option Explicit
```

While we are here, please set the "Width" and "Height" in the "Form Grid Settings" from the "General" tab to 60. Their default setting is 120, but the grid is not fine enough at this setting.



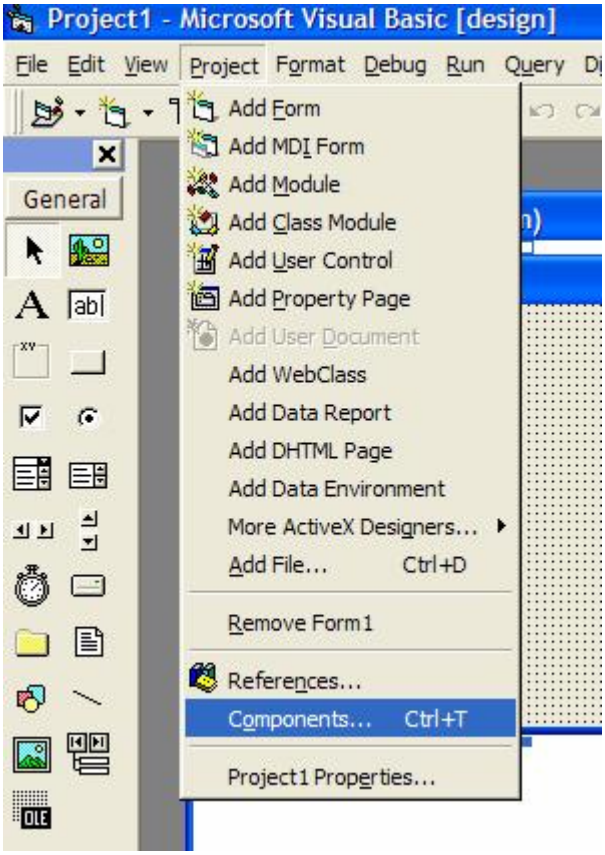
Press the OK button and close the Option dialogue box.In order to enable the settings, close Visual Basic here and then relaunch. When closing, it will ask whether or not to save to project as shown below; in this case only an Option setting was made, so press the No button to avoid saving.



Once the program is rebooted, once again select "Standard EXE" to create a blank project.

6.1.3 Adding bop to a Project

First of all, add bop ActiveX control to your project. Select "Project" – "Components…" from the menu.

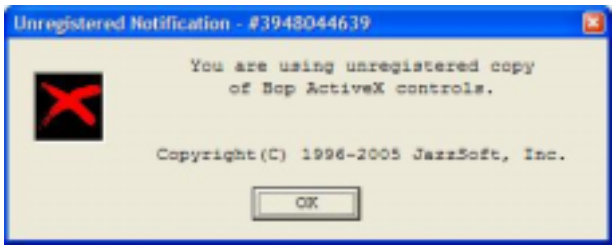


From the list of installed components, place a check mark in the check box for "bop ActiveX Control module" and press the OK button.

Bop will appear in the toolbox, and can then be pasted atanytime. Bop will be indicated by a musical note icon.

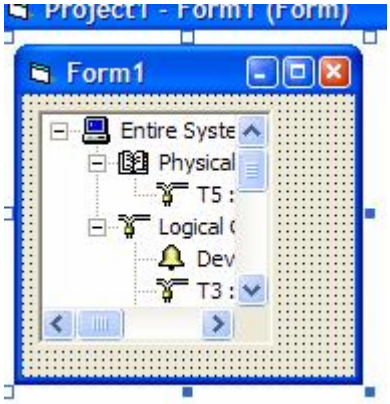
6.1.4 Pasting to a Screen

When bop is pasted to a screen from the toolbox, the following dialogue box will be displayed.



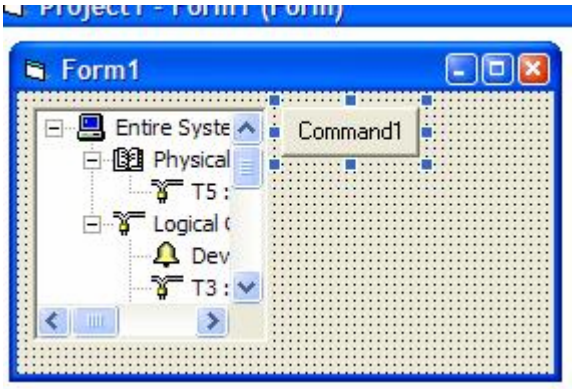
If the product version is purchased and the deletion key is attached, this dialogue box will cease to be displayed. When using the trial version of this product, this dialogue box will be displayed frequently, but will not affect operations. Press the OK button to close this nuisance display.

When bop is pasted to a screen, it will appear as follows.

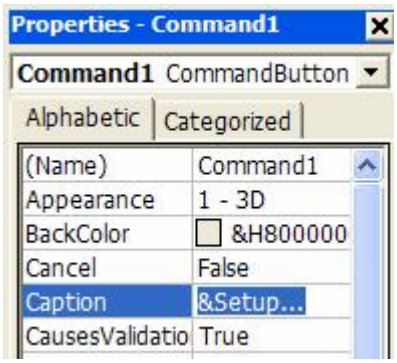


6.1.5 Creating a GEM Setting Screen

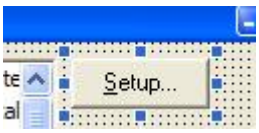
It is easy to create a GEM setting screen, which consists merely of calling up a method already prepared in bop. Start by attaching buttons to the screen.



Change the button's Caption property to "&Setup..." to make it appear appropriate as a setting button. When "&" is appended, underlining appears and it becomes a shortcut. It may also be useful to remember that the Microsoft style of notation is to suffix with "…" when the result of the button being pressed is the display of a dialogue box.



When the Caption properties are changed, the appearance of the button will be as shown below.

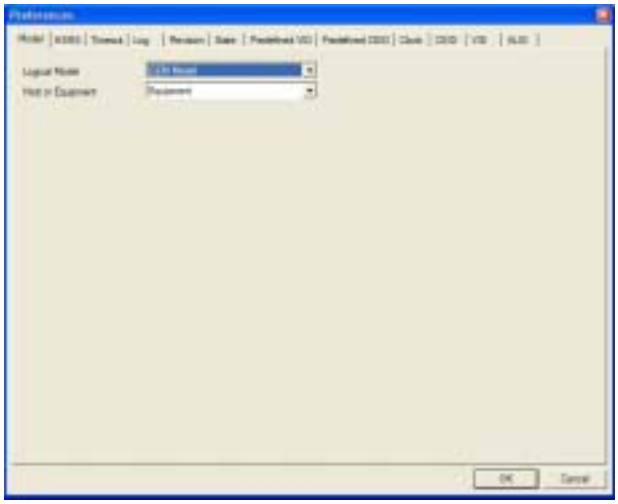


Enough for the external appearance, let's equip the internal workings. When the button is double-clicked, event handler functions will be displayed.

As the setup is for Click event processing, fill this in as-is. In order to display the setting screen, call up the bop Configure method. Here, you only need to

```
Private Sub Command1_Click()
   Bop1.Configure "", -1
End Sub
```

write one line. It may seem to simple to be true, but give it a try and a setting screen will be shown.



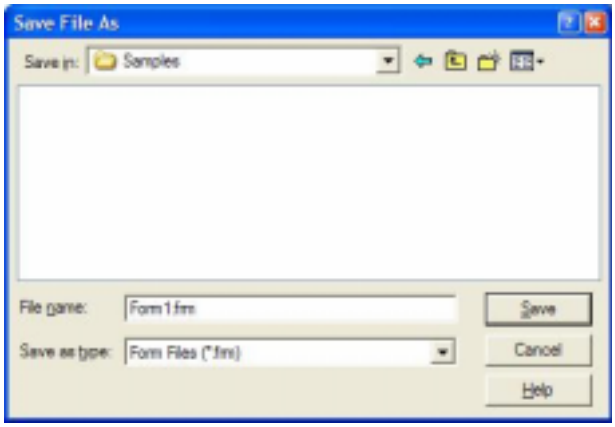
Once the functioning up to this point has been verified to work, close the application and complete this round of debugging.

6.1.6 Saving a Project

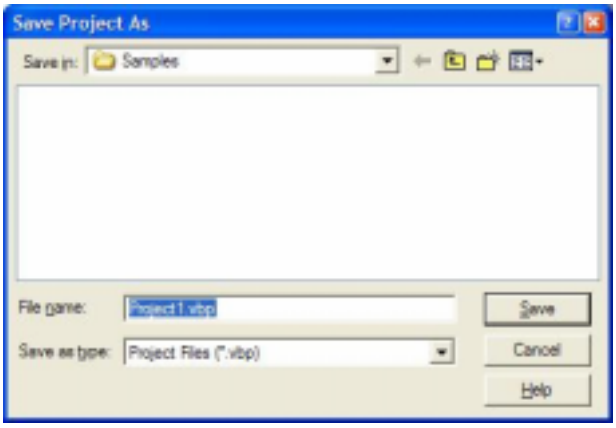
If you have ever developed using Visual Basic 6 you will already know this, but it is possible for Visual Basic to die due to an application error. So that such a problem does not delete the source code you have gone to the trouble to create, it is safer to perform frequent saves. Let us try to save our project at this time.

To save your project, click the floppy disk icon.

It will first ask the folder in which to save the form and the file name, so create an appropriate folder and save it to this folder. For the file name, it can remain at the default name of "Form1.frm". Press the Save button to save.

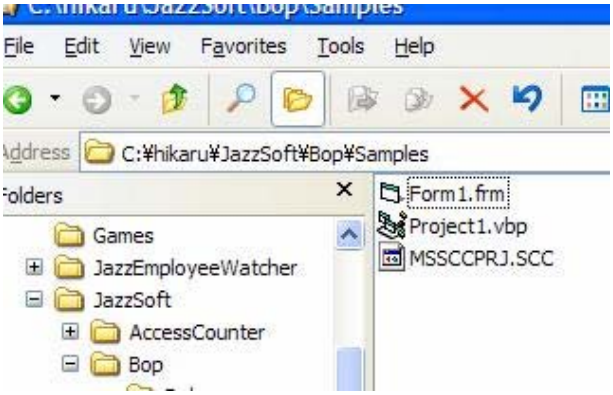


Next it will ask for the project's saving destination. Save this to the same folder also. The file name in this case may also be left at the default name of "Project1.vbp". Press the Save button to save.



If Visual Studio Enterprise Edition has been installed, it will ask whether to register in Visual Source Safe (VSS). In this case, as this is a tutorial, press No to cancel saving in VSS.

When viewed using the Explorer, you can confirm that Form1.frm and Project1.vbp are saved in the same folder.



Close Visual Basic at this point and make sure that your project opens when you double-click Project1.vbp from the Explorer. You will probably notice that a new file named Project1.vbw will be added.

6.1.7 Restoring Settings

Earlier we were able to write a setting screen in just one line, and we confirmed that setting contents were saved, but when the application is next launched they will disappear. This is because the application did not load the setting contents. Double-click the form and write the Form1 Load Event as shown below.

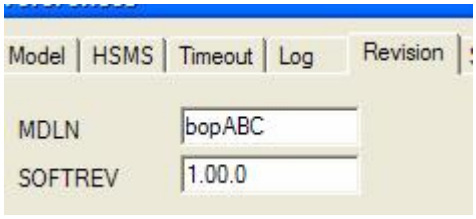
```
Private Sub Form_Load()
   Bop1.LoadIniFile
   Bop1.Load
End Sub
```

6.1.8 Revision Setting

Execute the application again, and set the previous setting screen. Set the Revision tab as shown below.

Item	Value
MDLN	bopABC
SOFTREV	1.00.0

The value may be any desired value, must may only be a maximum of 6 characters long. It is not possible to enter full-size characters or half-size katakana characters.

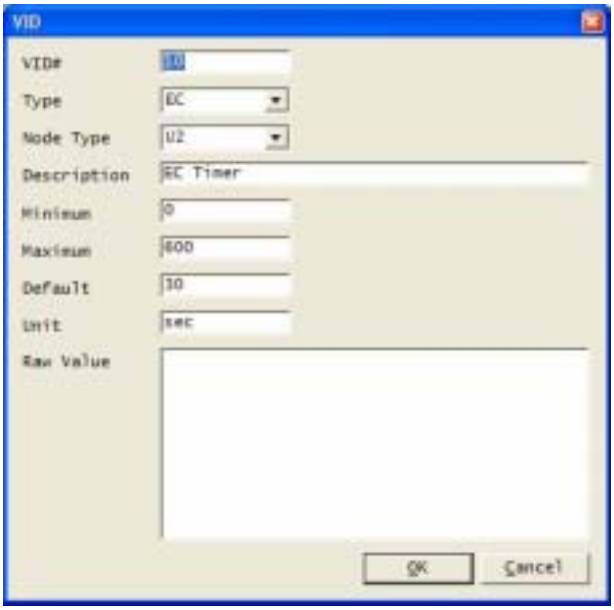


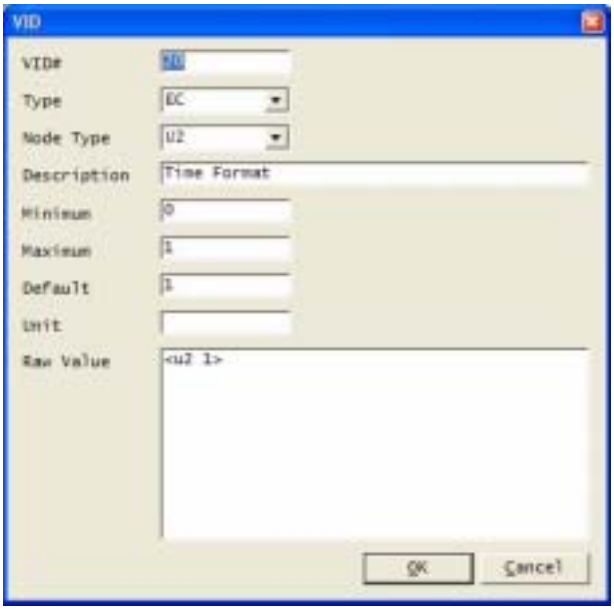
6.1.9 VI Setting

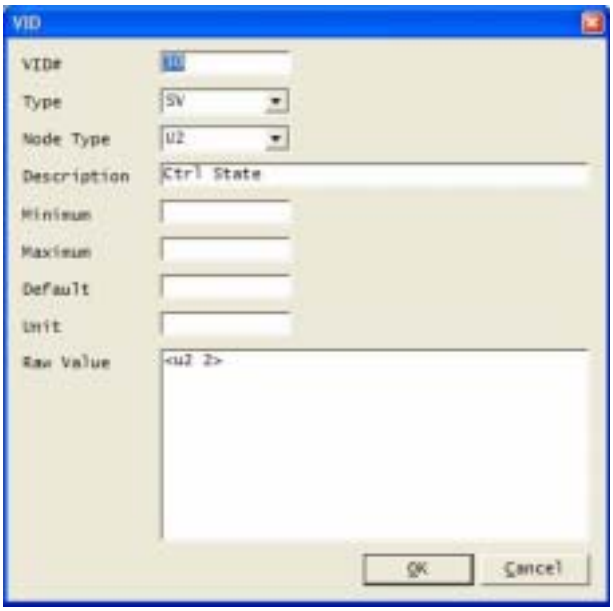
In accordance with this ?????????, register the following variables in the VID tab.

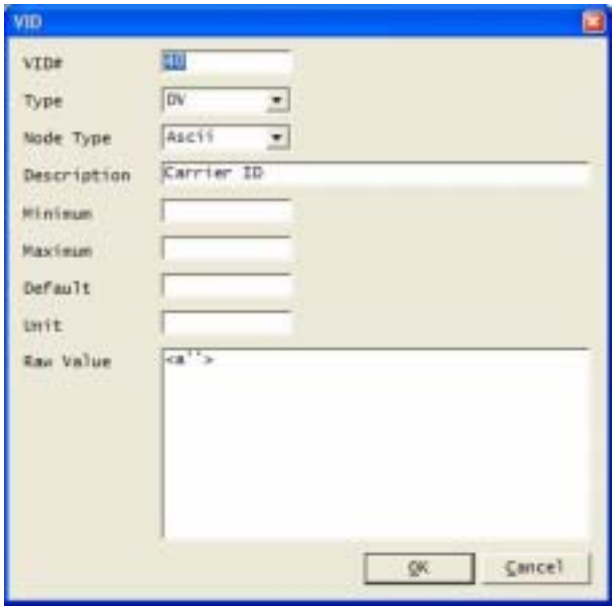
VID	Variable Type	Model	Description	Min. Value	Max. Value	Default Value	Unit
10	EC	U2	EC Timer	0	600	30	sec
20	EC	U2	Time Format	0	1	1	
30	SV	U2	Ctrl State				
40	DV	Ascii	Carrier ID				

Press the Add New... button to register one at a time.









When registration is complete the following will be shown.

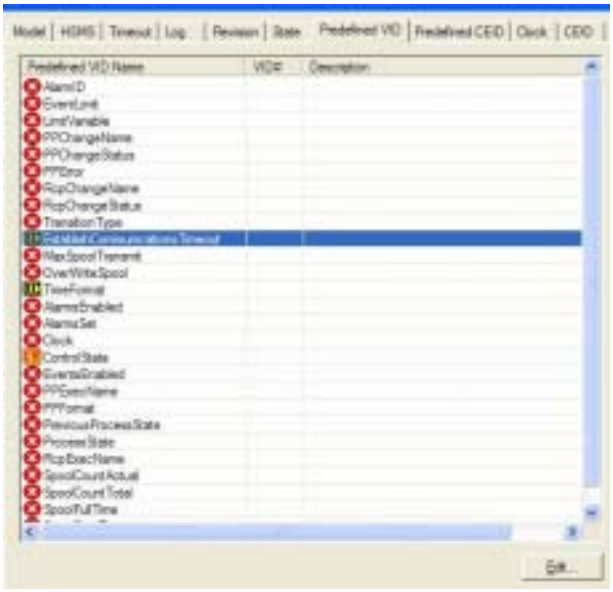
Press the OK button and save the settings.

6.1.10 Predefined VID Setting

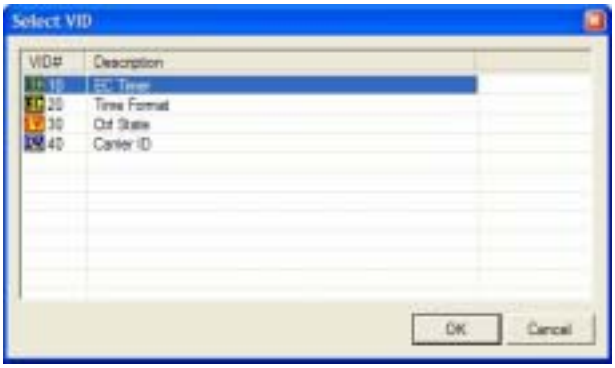
The predefined VID tab performs setting of “predefined variables”. Settings are as shown below.

Name of Predetermined Variable	VID#
Establish Communications Timeout	10
Time Format	20
Control State	30

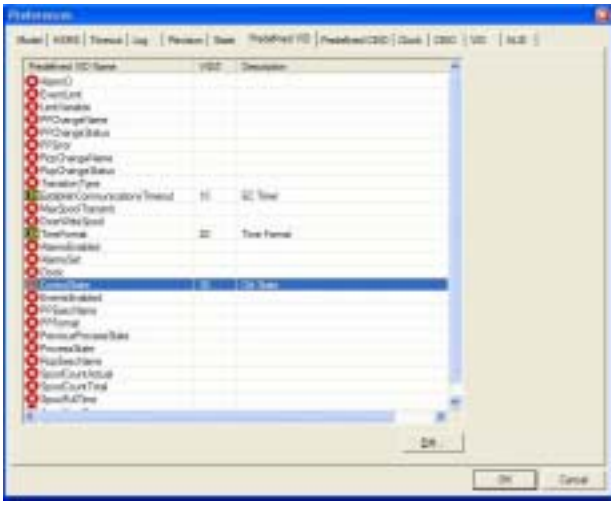
First select “Establish Communications Timeout” and press the Edit... button.



The VID registered earlier will be displayed in the list. This is why we saved the settings. Select VID#1 and press the OK button.



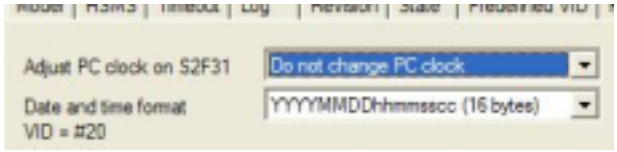
Set “Time Format” and “Control State” also, in the same manner.



Here press the OK button again to save the settings.

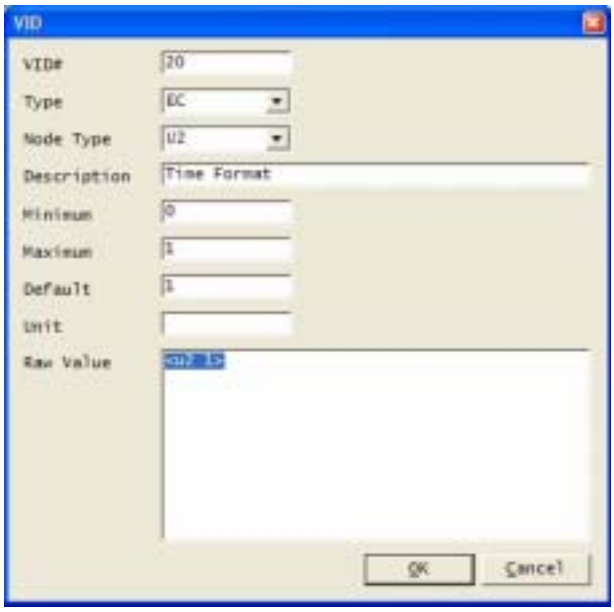
6.1.11 Setting the Clock

The Clock tab sets the date and time. It can be confirmed that”VID = #20” which was registered earlier is displayed in “Date and Time Format” in this screen.

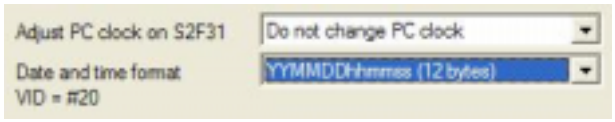


Once confirmed in the VID tab, it is possible to confirm that VID#20’s Raw Value (shown as SML in the table) has been rewritten to

```
<u2 1>
```



As a trial please change the Clock tab’s “Date and time format” to “12 bytes”.



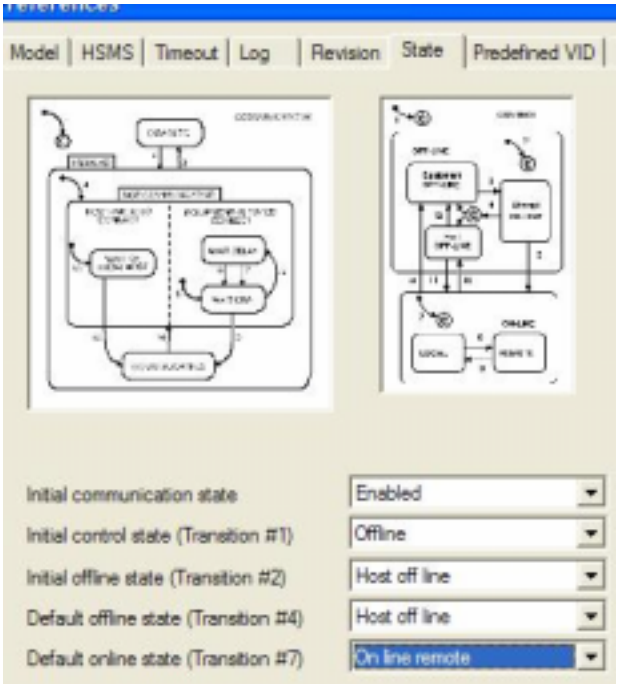
Press the OK button and save the settings. Open the setting screen again and you will see that VID#20 has been rewritten.

In this tutorial, we are leaving default values in place and no changes will occur. Therefore, here we must return the “Date and time format” to “16 bytes”.

6.1.12 State Setting

The State tab is set as follows.

Item	Value
Initial communication state	Enabled
Initial control state	Offline
Initial offline state	Host offline
Default offline state	Host offline
Default online state	Online remote

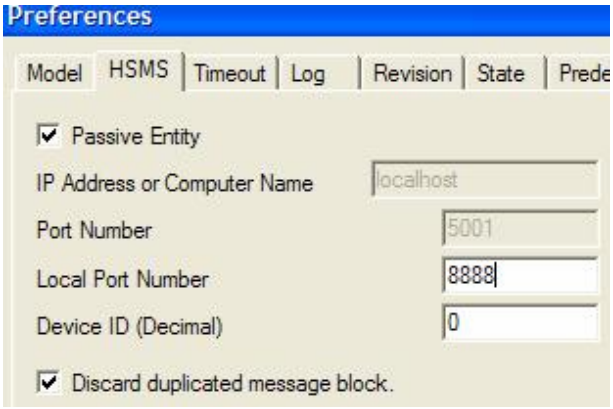


6.1.13 HSMS Setting

The HSMS tabs are set as follows in this tutorial.

Item	Value
Passive Entity	Yes (Checkmark)
Local Port Number	8888

Other parameters can stay at their default values.

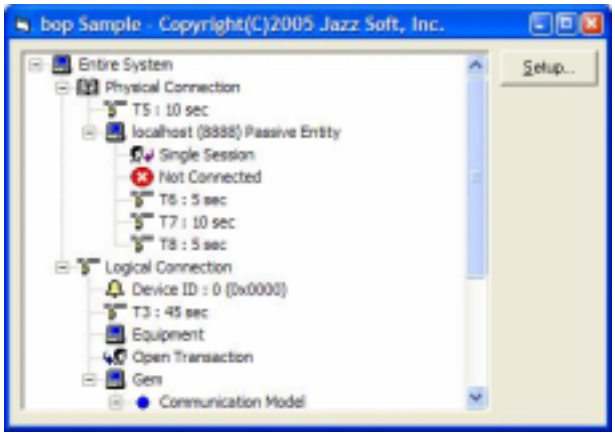


6.1.14 Enabling Communication

To begin HSMS communication, we must enable the physical connection. Specifically, we set the PhysicalConnection property to “true”.

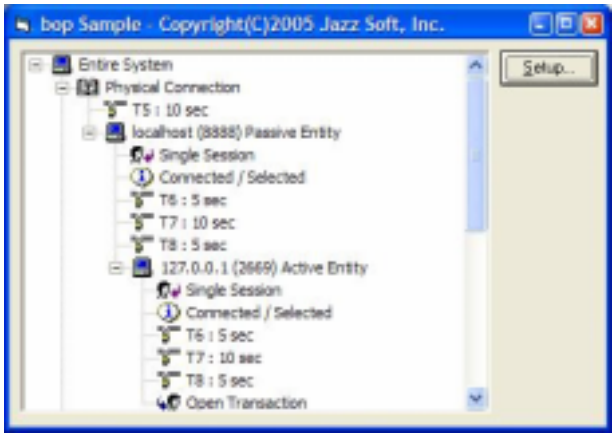
```
Bop1.LoadIniFile
Bop1.Load
Bop1.PhysicalConnection = True
```

Let us check to see whether HSMS really makes the connection. When the application is executed, the following physical connection can be seen to have been enabled.

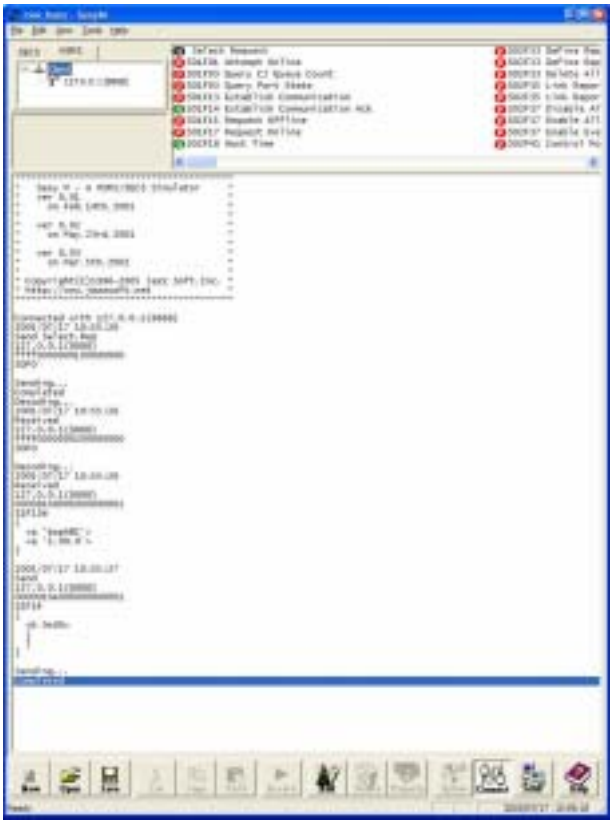


However, at this point the server (passive entity) has just launched, and if there is not connection from the client (active entity), no connection will be established.

Therefore here we can use a communication simulator and try to make the actual application connect. Any communication simulator software may be used; here we will try using Jazz Soft’s Sexy, which can be used free of charge.



When connected the above active entity will be displayed.On the simulator side as well we can see that the connection was made, as shown below.



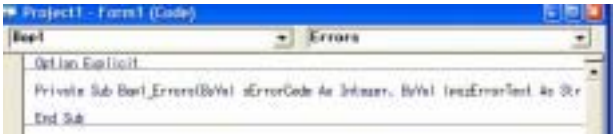
We can see that MDLN and SOFTREV, which were set in the Revision tab earlier, are coming via S1F13 Establish Communication Request(CR). MDLN and SOFTREV are also used by SlF2 Online Data(D), so please check when transitioning to online, as well.

Even though the S1F13/14 transaction should have been concluded, in a short while a T3 time-out will occur. Why is this?

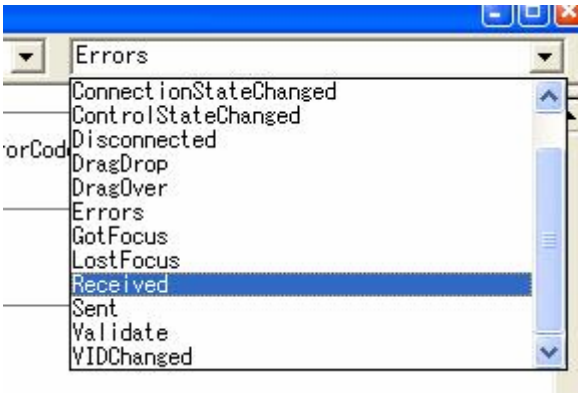
6.1.15 Message Processing

In bop, when a message is received, it is first communicated to the application, and it is not processed automatically. On the application side, the message may be displayed on the screen, history may be recorded, or other types of processing may be executed. However, most messages only need to be turned over to bop. In this case, we will choose refer all messages to bop.

Double-click bop in the design screen, to generate the event handler. First, an error events handler will be created in the code window.



Change to a Received event in the combo box.



A Received event handler will be created as shown below. The error event handler will not be used this time so we will delete it.

Write as shown below in the Received event handler. Note the simplicity of only having to write this one line.

```
Bop1.DefProc
```

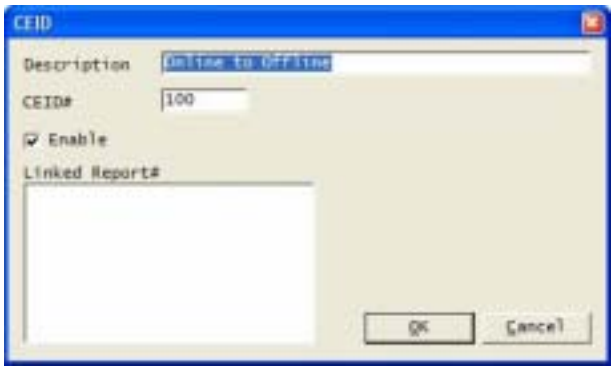
Messages received here will now be automatically be processed by bop.

6.1.16 CEID Setting

To generate a GEM event, EID must be registered. Launch the application and edit the CEID tab.

Press the Add New... button and register CEID as follows,in accordance with the specifications for this tutorial.

CEID	Description
100	Online to Offline
200	Carrier Loaded
201	Carrier Unloaded



If no checkmark is placed in “Enable”, the GEM event will be disabled. Settings to enable/disable GEM events may also be made in S2F37 Enable/Disable Event Report (EDER).

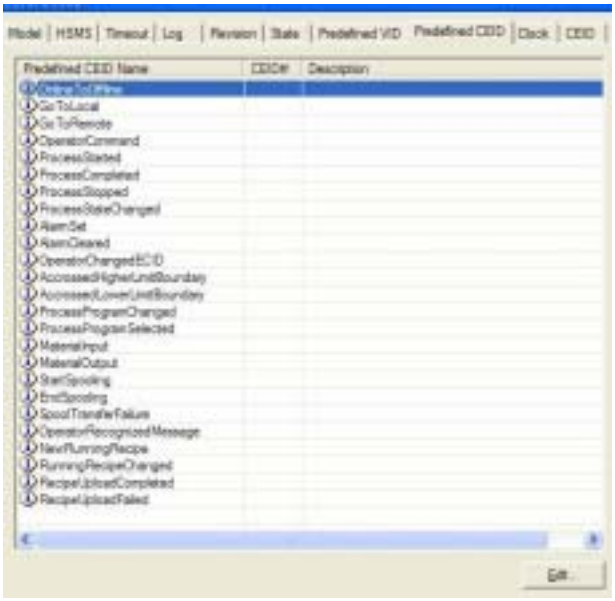
Press the OK button and save the settings at this time.

6.1.17 Predefined CEID Setting

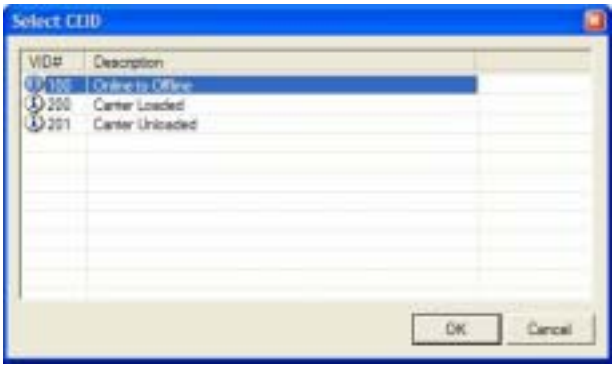
The Predefined CEID tab sets “Predefined GEM Events”. Setting is as follows.

Predefined Variable Name	VID#
Online To Offline	100

First select “Online To Offline” and press the Edit...button.



The CEID list registered earlier will be displayed. This is the reason we saved the settings. Select CEID#100 and press the OK button.



Press the OK button to save the settings, and let us check to make sure that the GEM event is really sent. Check to verify that the S1F13/14 transaction has been concluded, and send S1F17 Online Request(RONL)from the simulator.

```
Sending...
Completed
2005/07/17 11:39:09
Send
127.0.0.1(8888)
00008111000000000031
S1F17W

Sending...
Completed
Decoding...
2005/07/17 11:39:09
Received
127.0.0.1(8888)
00000112000000000031
S1F18
<b 0x00>
```

Sexy will display detailed data, so we will make it so that these items are deleted and only the message is shown.

```
Send
S1F17W
```

```
Received
S1F18
<b 0x00>
```

It appears that online transition was performed correctly. Just to be sure, send S1F1 Online Check Request(R)to confirm it.

```
Send
S1F1W
```

```
Received
S1F2
{
   <a 'bopABC'>
   <a '1.00.0'>
}
```

We have confirmed that online transition occurred correctly. Next, let’s send S1F15 Request Offline (ROFL) and try to transition to offline operation.

```
Send
S1F15W
```

```
Received
S1F16
<b 0x00>
```

We transitioned to offline correctly. This generates an Online To Offline event.

```
Received
S6F11W
{
   <u4 1>
   <u4 100>
   {
   }
}
```

```
Send
S6F12
<b 0x00>
```

6.1.18 Enabling/Disabling a GEM Event

We have already learned that GEM events can be enabled/disabled through communication as well. Let us now try to actually perform this setting. When “false” is specified for CEED in S2F37 Enable/Disable Event Report(EDER), and when the continuing list length is zero, all GEM events are disabled.

```
Send
S2F37W
{
   <bool false>
   {
   }
```

```
}

Received
S2F38
<b 0x00>
```

Let’s try to transition offline.

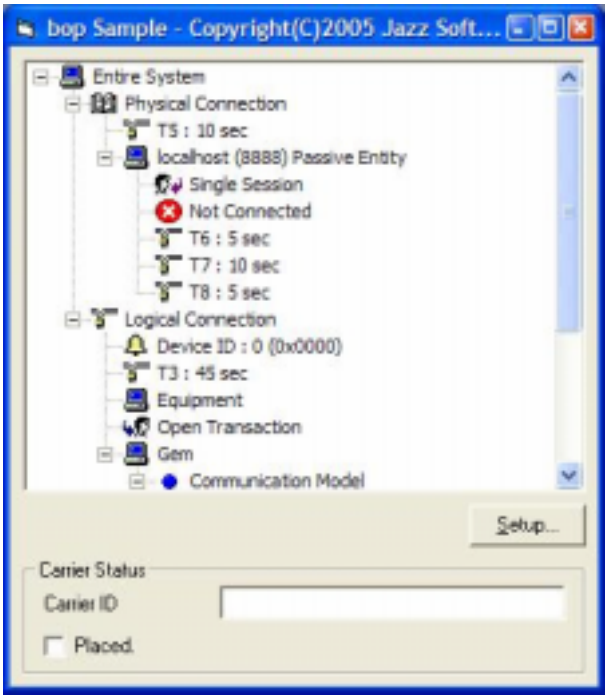
```
Send
S1F15W

Received
S1F16
<b 0x00>
```

We were able to verify that since the GEM event was disabled, S6F11 was not generated.

In the example above, we disabled all GEM events at once,but enabling/disabling GEM events can be set for each individual CEID unit as well.

6.1.19 Sending a GEM Event

This time, let us try generating a GEM event when a carrier is placed on or removed from a load port. First, paste a text box and check box to a screen as follows.



The check box Click event is written as follows.

```
Private Sub Placed_Click()
   If Placed.Value = 1 Then
      Bop1.InvokeEvent 200
   Else
      Bop1.InvokeEvent 201
   End If
End Sub
```

Try enabling all GEM events.

```
Send
S2F37W
{
   <bool true>
   {
   }
}

Received
S2F38
<b 0x00>
```

Execute the application and click the check box. The CEID #200 Carrier Loaded event will be sent.

```
Received
S6F11W
{
   <u4 1>
   <u4 200>
   {
   }
}

Send
S6F12
<b 0x00>
```

When the check box is clicked once again, the CEID #201 Carrier Unloaded event is sent.

```
Received
S6F11W
{
   <u4 2>
   <u4 201>
   {
   }
}

Send
S6F12
<b 0x00>
```

6.1.20 Defining a Dynamic Report

In the previous GEM events, no reports were attached. Now let’s define a report from the communication simulator side. First, disable all GEM events.

```
Send
S2F37W
{
   <bool false>
   {
   }
}

Received
S2F38
<b 0x00>
```

In this status, even if the check box is clicked, GEM events will stop being sent.

Next, discard all reports.

```
Send
S2F33W
{
   <u4 0>
   {
   }
}

Received
S2F34
<b 0x00>
```

Define a new report. Here, VID #40 is pasted to report #1000

```
Send
S2F33W
{
   <u4 0>
   {
      {
         <u4 1000>
         {
            <u4 40>
         }
      }
   }
}

Received
S2F34
<b 0x00>
```

We will link the report to the GEM event. Here we have linked report #1000 to CEID #200 and Report #1000 to CEID #201.

```
Send
S2F35W
{
   <u4 0>
   {
      {
         <u4 200>
         {
            <u4 1000>
         }
      }
      {
         <u4 201>
         {
            <u4 1000>
         }
      }
   }
}

Received
S2F36
<b 0x00>
```

Finally, we will enable all GEM events.

```
Send
S2F37W
{
   <bool true>
   {
   }
}

Received
S2F38
<b 0x00>
```

When the application's check box is checked, reports are now attached when GEM events are sent.

```
Received
S6F11W
{
   <u4 3>
   <u4 200>
   {
      {
         <u4 1000>
         {
            <a>
         }
      }
   }
}

Send
S6F12
<b 0x00>
```

6.1.21 Updating Variables

As VID #40 has not been updated, a blank character string has been sent with the previous GEM event. Let's try updating VID immediately before sending a GEM event. Updating VID is simple and only requires the addition of one line.

```
Private Sub Placed_Click()
   Bop1.VIDValue(40) = CarrierID.Text
   If Placed.Value = 1 Then
      Bop1.InvokeEvent 200
   Else
      Bop1.InvokeEvent 201
   End If
End Sub
```

When a character string is put into a text box in the application, the character string will be sent as shown below.

```
Received
S6F11W
{
   <u4 6>
   <u4 201>
   {
      {
         <u4 1000>
         {
            <a 'Be Bop'>
         }
      }
   }
}

Send
S6F12
<b 0x00>
```

6.1.22 Setting an ALID

It is necessary to register an ALID to generate an alarm. We will edit the ALID tab in the application's setting screen.

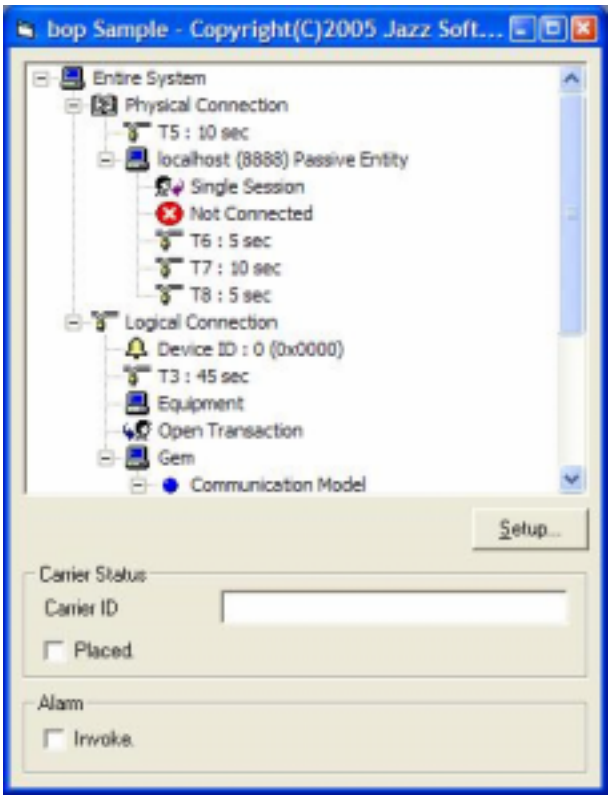
Press the Add New... button, and register the following ALID in accordance with the specifications for this tutorial.

ALID	Description
30000	Alignment Failure

If "Enable" is not checkmarked, the alarm will be disabled. It is also possible to set whether an alarm is enabled or disabled via communication. Press the OK button at this time and save the settings.

6.1.23 Sending an Alarm

Let's add an alarm generating function to our application. Paste a check box to the screen.



The check box's Click event is written as follows.

```
Private Sub Invoke_Click()
   If Invoke.Value = 1 Then
      Bop1.InvokeAlarm 30000, -1
   Else
      Bop1.InvokeAlarm 30000, 0
   End If
End Sub
```

To explain this code, when a checkmark is present it specifies "-1" for the InvokeAlarm method argument. This means that an alarm will be generated. In the same way, if the check box is not checked, it specifies "0", and this means clearing of the alarm. Please note that if there is no alarm occurrence, an alarm clear will not be sent.

6.1.24 Full Source Code

The above has been an overview introducing bop's functions. It has many more functions, and thus not all of them could be covered here, but the above should have provided a look at some of bop's powerful functions.

This sample program is GEM compliant in every respect. However, the source code is unbelievably short, totaling only 33 lines in all. This may be surprising! The more lines a piece of software has, the more chance there is for bugs to be present, so a shorter source code will always lead to fewer bugs. Of course, this also shortens development time and reduces development costs as well.

```
Option Explicit

Private Sub Bop1_Received(ByVal lpszIPAddress As String,
ByVal lPortNumber As Long)
   Bop1.DefProc
End Sub

Private Sub Command1_Click()
   Bop1.Configure "", -1
End Sub

Private Sub Form_Load()
   Bop1.LoadIniFile
   Bop1.Load
   Bop1.PhysicalConnection = True
End Sub

Private Sub Invoke_Click()
   If Invoke.Value = 1 Then
      Bop1.InvokeAlarm 30000, -1
   Else
      Bop1.InvokeAlarm 30000, 0
   End If
End Sub

Private Sub Placed_Click()
   Bop1.VIDValue(40) = CarrierID.Text
   If Placed.Value = 1 Then
      Bop1.InvokeEvent 200
   Else
      Bop1.InvokeEvent 201
   End If
End Sub
```

One more thing to add: In Visual Basic it is possible to use With to abbreviate objects.

```
   With Bop1
      .LoadIniFile
      .Load
      .PhysicalConnection = True
   End With
```

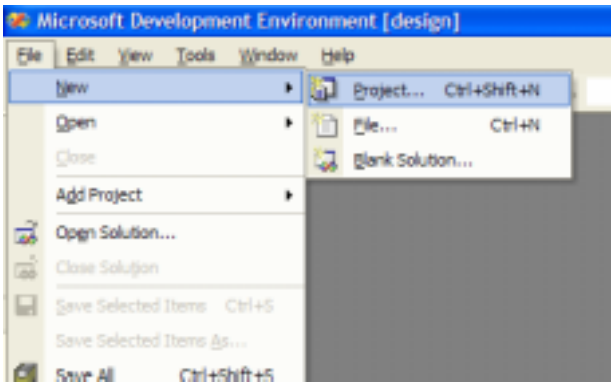
The Descriptions which follow use this abbreviation format.

6.2 Visual Basic.NET Version 2003 Version

In the case of Visual Basic.NET as well, there is not much difference versus Visual Basic 6.0.

6.2.1 Creating a New Project

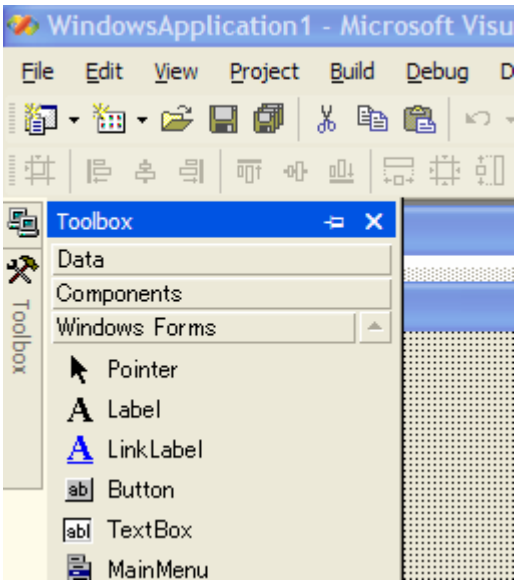
Launch Visual Studio .NET 2003 and click "File" – "New"-"Project" from the menu.



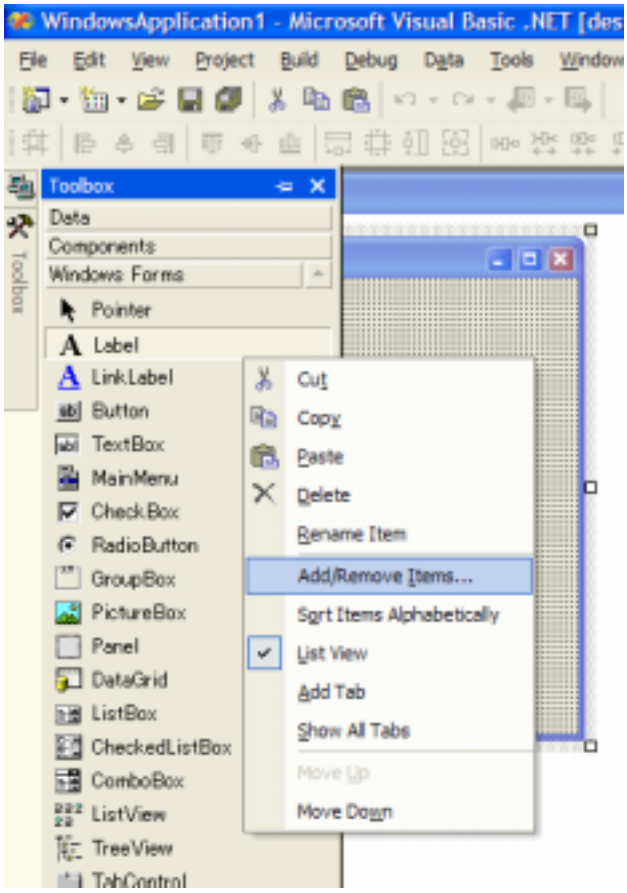
Select Visual Basic Project from the project type list, and select Windows Application as the template. Select a folder in which to save your project and press the OK button.

6.2.2 Adding bop to a Toolbox

When you hold the mouse above the toolbox, the following toolbox will open.

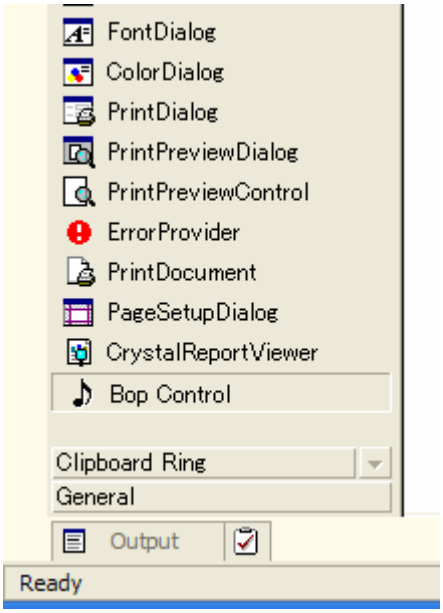


Right-click on top of this open toolbox and select Add/Remove Items....



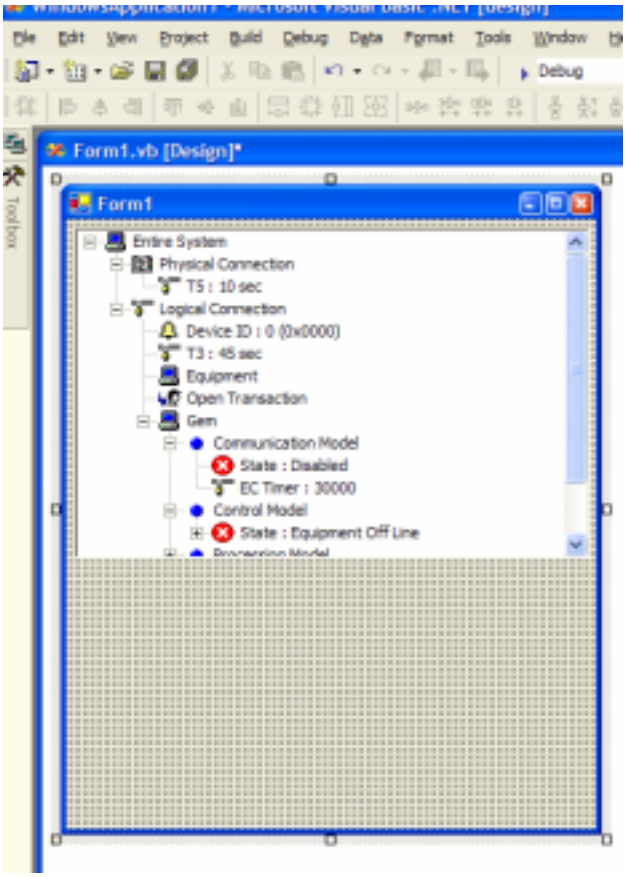
Select the COM Components tab, check Bop Control from the list and press the OK button.

Check to make sure that Bop Control was registered in the toolbox.

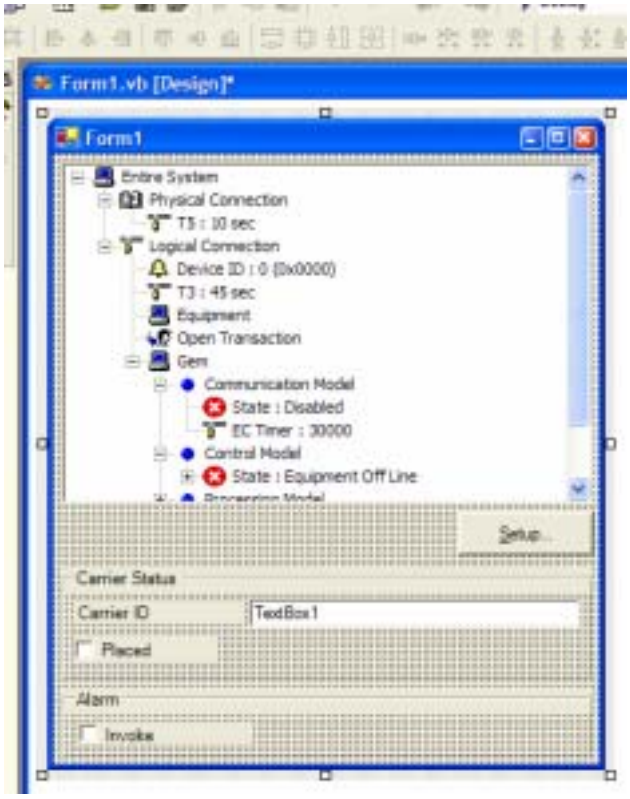


6.2.3 Pasting to a Screen

When bop is pasted to a screen it will appear as follows.



Paste the other controls in the same manner as for Visual Basic Version 6.0.



6.2.4 Creating a GEM Setting Screen

When the button marked “Setup...” is double-clicked, the following screen will appear.

Write as follows at this time.

```
AxBop1.Configure("", -1)
```

This can be written with a single line, just like in Visual Basic 6.0.

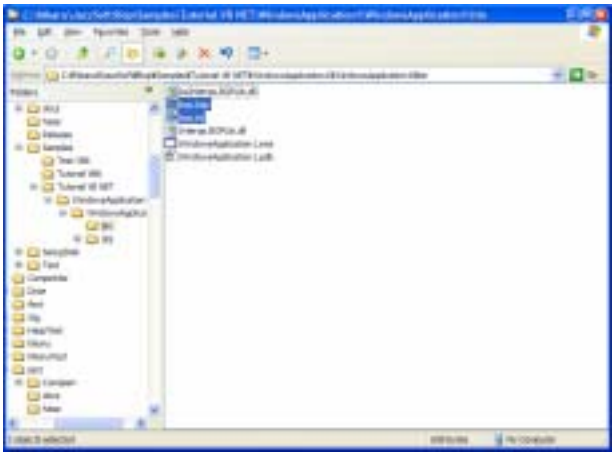
6.2.5 Restoring Settings

Double-click the form and write as follows in the Form 1 Load event.

```
AxBop1.LoadIniFile()
AxBop1.Load()
```

6.2.6 Copying a Setting File

The GEM setting method is the same as for Visual Basic Version 6.0. Here, let’s copy and reuse the bop.bop and bop.ini files created in Visual Basic Version 6.0. Copy these two files into the execute folder (bin folder).



6.2.7 Enabling Communication

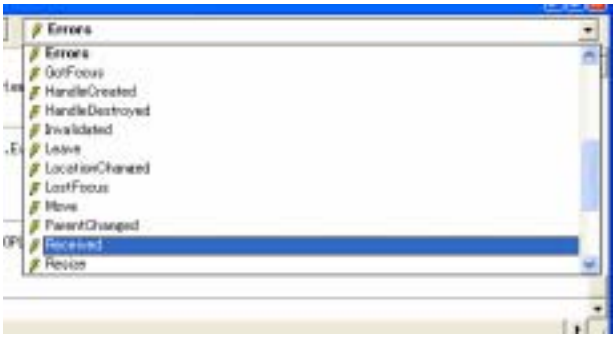
To initiate HSMS communication, set “true” for the PhysicalConnection property.

```
AxBop1.LoadIniFile()
AxBop1.Load()
AxBop1.PhysicalConnection = True
```

6.2.8 Message Processing

To write a message receiving process, double-click bop in the same fashion as in Visual Basic Version 6.0. Errors event handler functions will be created first, as shown below.

Re-select to a Received event.



Received event handler functions will be created as shown below. Go ahead and erase the Errors event handler functions.

Here, write as follows.

```
AxBop1.DefProc()
```

6.2.9 Sending a GEM Event

The process for sending a GEM event is also virtually the same as with Visual Basic Version 6.0. However, with Visual Studio .NET, there is a slight change in the grammar for accessing the properties of the array type.

```
AxBop1.set_VIDValue(40, CarrierID.Text)
If Placed.Checked Then
  AxBop1.InvokeEvent(200)
Else
  AxBop1.InvokeEvent(201)
End If
```

6.2.10 Sending an Alarm

The process of sending an alarm is almost the same.

```
If Invoke.Checked Then
  AxBop1.InvokeAlarm(30000, -1)
Else
  AxBop1.InvokeAlarm(30000, 0)
End If
```

6.2.11 Full Source Code

If we leave out the code created by Windows Form Designer, we can see that very few lines of coding is required, similar to with Visual Basic 6.0.

```
Public Class Form1
   Inherits System.Windows.Forms.Form

Windows Form Designer generated code

   Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
     AxBop1.Configure("", -1)
   End Sub

   Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
     AxBop1.LoadIniFile()
     AxBop1.Load()
     AxBop1.PhysicalConnection = True
   End Sub

   Private Sub AxBop1_Received(ByVal sender As Object,
ByVal e As AxBOPLib._DBopEvents_ReceivedEvent) Handles
AxBop1.Received
     AxBop1.DefProc()
   End Sub

   Private Sub Placed_CheckedChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Placed.CheckedChanged
     AxBop1.set_VIDValue(40, CarrierID.Text)
     If Placed.Checked Then
       AxBop1.InvokeEvent(200)
     Else
       AxBop1.InvokeEvent(201)
     End If
   End Sub

   Private Sub Invoke_CheckedChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
Invoke.CheckedChanged
     If Invoke.Checked Then
       AxBop1.InvokeAlarm(30000, -1)
     Else
       AxBop1.InvokeAlarm(30000, 0)
     End If
   End Sub
End Class
```

6.3 Visual C++ Version 6.0

In the case of Visual C++, since the language is different from Visual Basic, there are a few differences.

In Japan there is a strong climate of disdain for the Basic language. It appears to have inherited the image of N88-BASIC,the language of choice for amateur programmers. However, Visual Basic is an extremely refined language which can be said to have exceeded the limits of a simple language, but the old prejudices still linger.

In this climate of disdain for Basic, C++ has become the main language of interest in Japan, and the tendency is to dislike programs created using Basic. Nonetheless, C++ is an extremely difficult language, to the extent that even professionals with a complete grasp of C cannot master is easily. In reality, 90% or more of those who call themselves C++ programmers are in a lamentable situation of writing in C , with only the compiler environment using C++. This situation is probably also clear when looking at the very few numbers of JAVA programmers, which has fewer functions than C++.

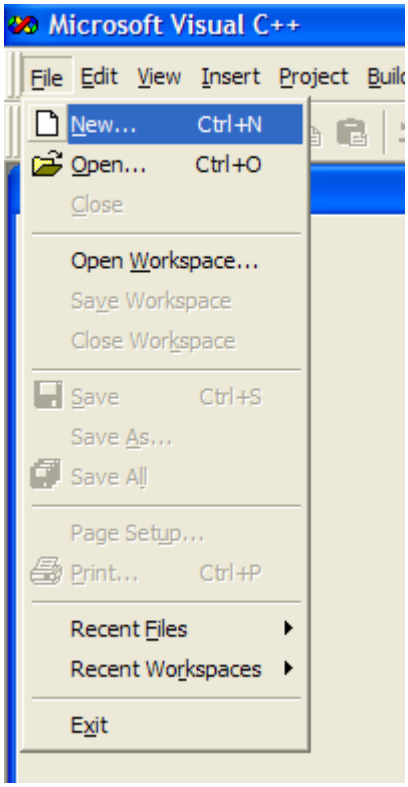
Observing the self-styled C++ programmers around me, there are many whom even at 10 years of C++ experience, do not know the very basics, such as that “destructors must be virtual”. Since the knowledge level of programmers in general is this low, anyone without confidence in their C++ expertise had better not even try development using it. It will only decrease productivity.

As we moved forward to Visual Studio .NET, it would not be an exaggeration to say that the boundary between Basic and C++ has almost disappeared. This is why these days, C++ dropouts have revised their thinking and although only gradually, have started using Basic, thinking that “with Basic there is at least a chance of mastery”. Converts from COBOL and FORTRAN despair of basic mastering of C++, and even when they hop right over to Basic, they worry about the world’s prejudicial attitudes toward Basic. They say they would be embarrassed to know only Basic. Perhaps the tide of restoration in Basic’s reputation will be a good opportunity.

The above has diverged from our topic, but the source code we have created in our tutorial with VC++ is not significantly different from Visual Basic. Since we are creating the same specification of software, this is obviously the case. It is just that there are more obscure areas than in Basic. In this tutorial, we will not be touching on VC++ or MFC (Microsoft Foundation Class Library) details, so it would be desirable for those with points of inclarity to study on their own, using MSDN (Microsoft Developer Network) , etc.[1]

6.3.1 Creating a New Project with the App Wizard

Launch Visual C++ 6.0 and click “File” – “New” from the menu.



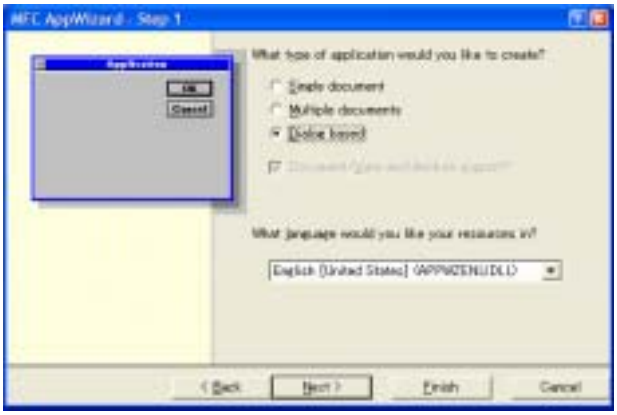
Select the MFC AppWizard (.exe), specify the project name and folder, and press the OK button.

There are three types of projects that can be created with Visual C++ 6.0.

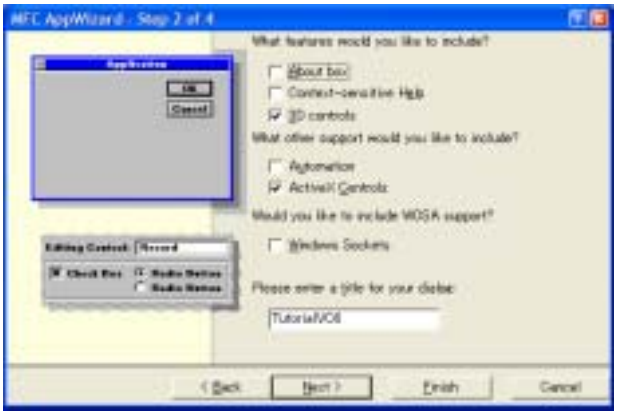
Project	Description
Single document	Applicating handling one document at a time, such as “Memo Pad”. This is called an SDI (Single Document Interface) .
Multiple documents	Applications which can handle multipledocuments at a time, such as “Visual C++ 6.0” This is called an MDI (Multiple Document Interface) .
Dialog based	Simple application starting from a dialogue box, such as software created using Visual Basic or C#. Document/View structure cannot be used.

Here we will create a Dialog based application. Select a dialog-based radio button and press the Next button.

[1] Jazz Soft also provides bop、swing、and VC++ training (for a separate fee)

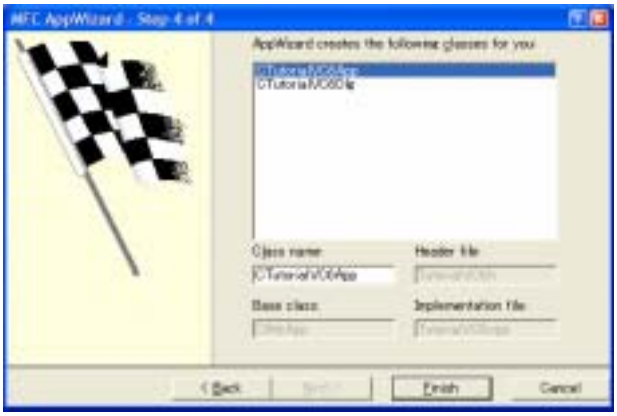


As no About box (version information dialogue box) is needed, uncheck it and press the Next button.



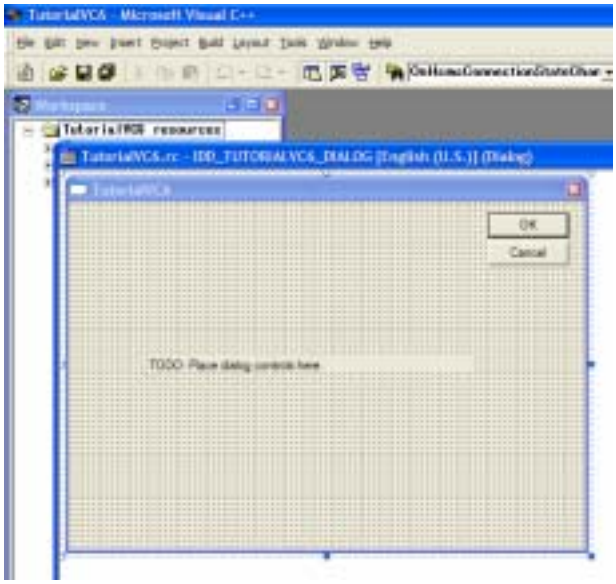
This screen can remain in its default state, so simply press the Next button.

This screen can also remain in its default state, so press the Finish button.



The final check screen will be displayed. Press the OK button.

The project will be created.

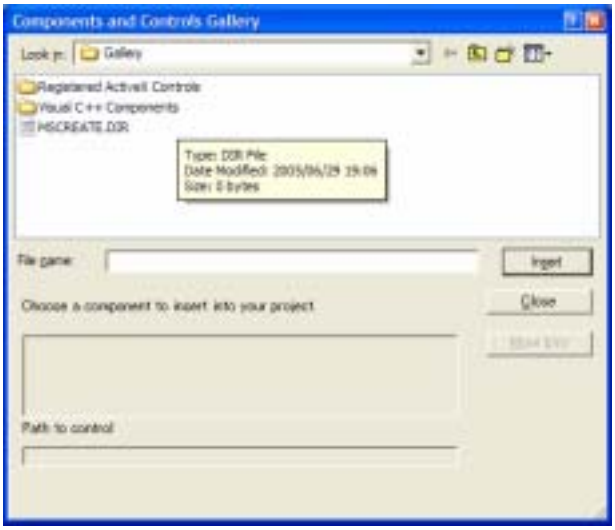


6.3.2 Inserting bop

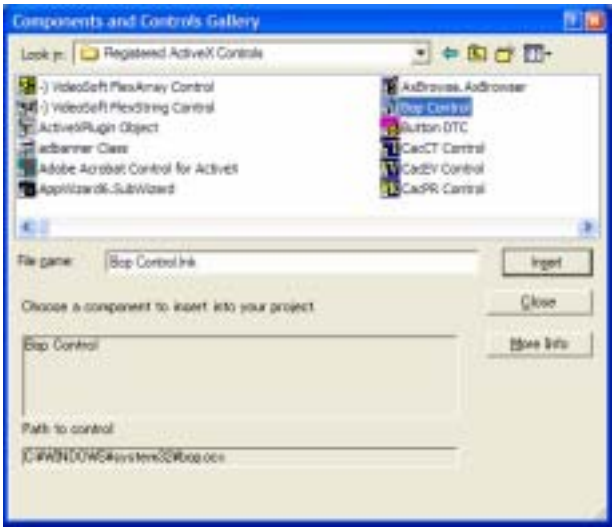
In Visual C++ 6.0, it is necessary to perform the Insert process prior to using bop. This creates a wrapper class from the ActiveX control type library.

Select “Project”-“Add To Project”-“Components and Controls...” from the menu.

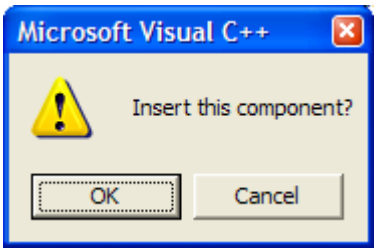
Double-clidk Registered ActiveX Controls and move to that folder.



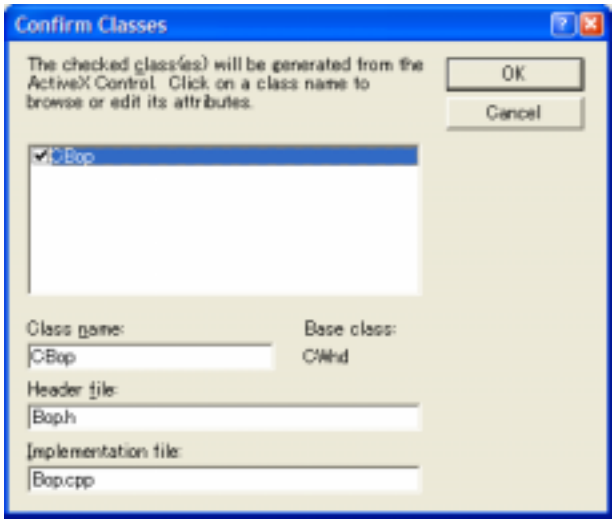
Select bop Control from the COM list and press the Insert button.



It will confirm whether or not to insert; press the OK button.



The check screen will be displayed, but the wrapper class name and file name can both remain in their default state, so simply press the OK button.

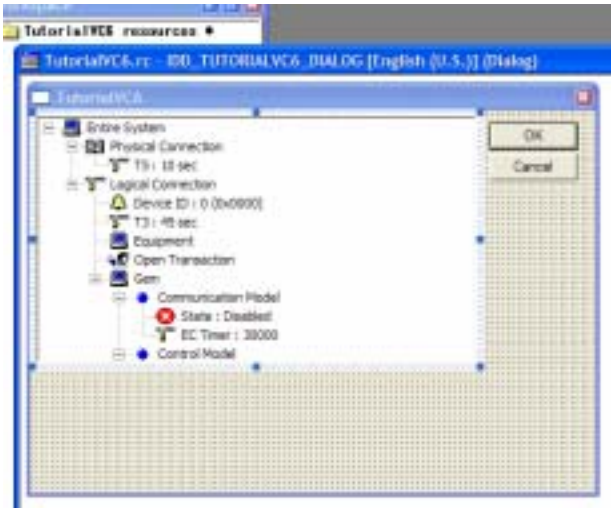


When we return to the COM list screen, press the Close button to close the screen. Check to confirm that bop has been added to the control list.

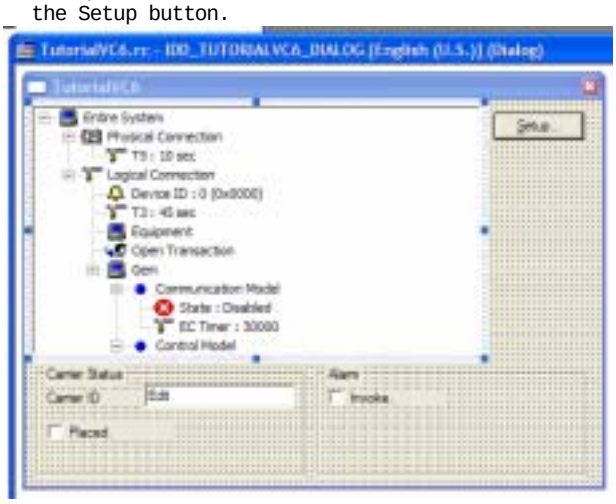
6.3.3 Pasting to a Screen

There is alrealy a text box saying "TODO: Place dialog controls here." pasted in the dialogue box screen; erase it as it is not needed. Also, we do not need a cancel button because pressing the × mark at the top right of the dialogue box will close the application, so delete this as well.

When bop is pasted to the screen it will be as shown below.



In the same manner as with Visual Basic Version 6.0, we will paste other controls. Let's use the OK button as the Setup button.



6.3.4 Mapping to a Member Variable

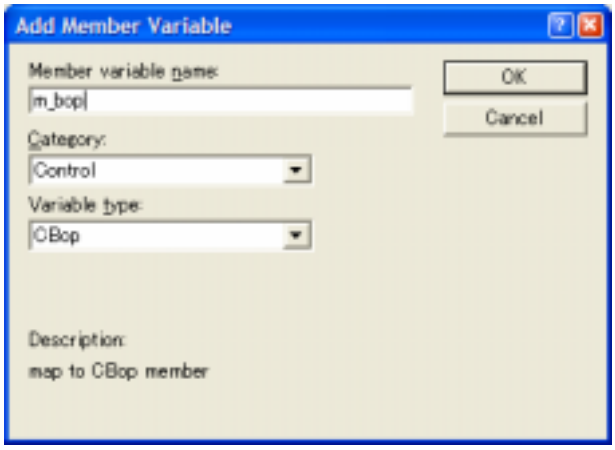
With only pasting a control, there is no name in Visual C++ 6.0. It is possible to acquire a pointer for the control using GetDlgItem(), but there is an easier way. This is by making it a member variable.

In Visual C++ 6.0, Class Wizard is used for almost everything, such as creating member variables, creating event handler functions, etc. Select "View"-"Class Wizard..." from the menu.

When Class Wizard opens, first select the Member Variables tab as the class name, and verify that CtutorialVC6Dlg has been selected. In this tutorial, since we only have 2 classes, a dialogue box class and an application class, it is very simple. However, when many screens are created such as MDI and SDI, etc., there are some cases in which errors can be made in selecting the target class.

Select the Member Variables tab, and press the Add Variable button.

When a dialogue box such as that below opens, enter the variable name. In Microsoft style, member variables start with "m_" so let us follow that convention. Here we have input the name "m_bop". When entry is finished press the OK button.

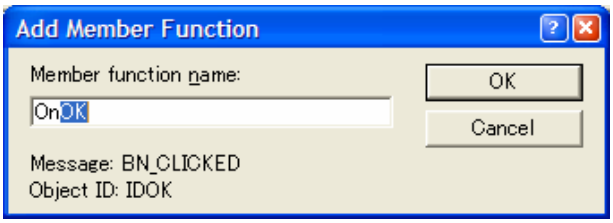


Here we can confirm that the member variable is displayed in the list.

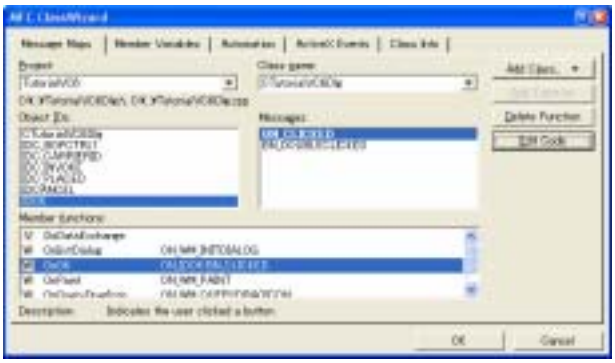
6.3.5 Creating a GEM Setting Screen

Class Wizard is used also for adding GEM setting screens. The Setup button has an ID of IDOK, so select this, and create a BN_CLICKED even handler function. Press the Add Function button.

A dialogue box confirming the function name will be displayed, but this can remain at the default status so press the OK button.



The event handler function is created. Press the Edit Code button.



Jump to the event handler function.

```
        CDialog::OnPaint();
    }
}

// The system calls this to obtain the cursor
//  the minimized window.
HCURSOR CTutorialVC6Dlg::OnQueryDragIcon()
{
    return (HCURSOR) m_hIcon;
}

void CTutorialVC6Dlg::OnOK()
{
    // TODO: Add extra validation here

    CDialog::OnOK();
}
```

Change as follows.

```
void CTutorialVC6Dlg::OnOK()
{
   m_bop.Configure(NULL,-1);

// CDialog::OnOK();
}
```

CDialog::OnOK() calls up a hypothetical member function in the parent class, and in this is written the process for closing the dialogue box. That is why this section is commented out.

The first argument in the Configure()method can be written as follows.

```
   m_bop.Configure(“”,-1);
```

In Basic, pointers cannot be used, so we have used “ ”” ”, but in C++ we can use NULL. However, they are both the same.

6.3.6 Restoring Settings

In Class Wizard, write as follows in OnInitDialog(). OnInitDialog() is the WM_INITDIALOG event handler function.

```
   m_bop.LoadIniFile();
   m_bop.Load();
```

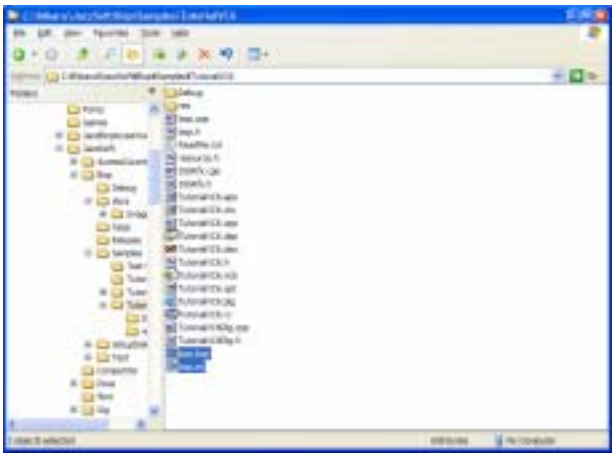
Following addition, OnInitDialog() will be as follows.

```
BOOL CTutorialVC6Dlg::OnInitDialog()
{
   CDialog::OnInitDialog();

   // Set the icon for this dialog.  The framework does
this automatically
   //  when the application's main window is not a dialog
   SetIcon(m_hIcon, TRUE);      // Set big icon
   SetIcon(m_hIcon, FALSE);     // Set small icon

   m_bop.LoadIniFile();
   m_bop.Load();

   return TRUE;  // return TRUE  unless you set the focus
to a control
}
```

6.3.7 Copying a Setting File

The method of setting in GEM is the same as in Visual Basic Version 6.0. Here we will copy and re-use the bop.bop and bop.ini files we created in Visual Basic

Version 6.0. Copy these 2 files to the source folder (folder with .dsw) .



6.3.8 Enable Communication

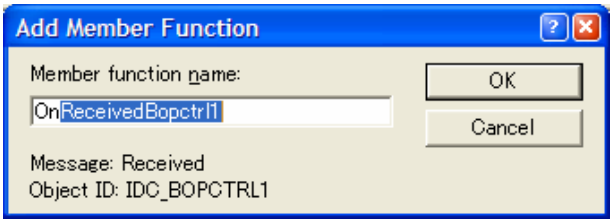
To start HSMS communication, set "true" for the PhysicalConnection property.

```
m_bop.LoadIniFile();
m_bop.Load();
m_bop.SetPhysicalConnection(true);
```

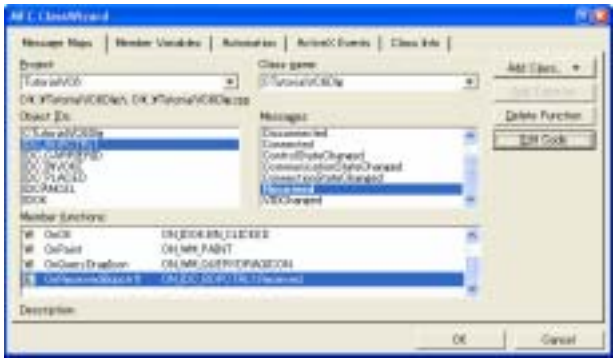
6.3.9 Processing Messages

Class Wizard is also used to write the message receiving process. Select IDC_BOPCTRL1 (bop resource ID name) , select Received event and press the Add Function button.

A dialogue box confirming the handler function name will be displayed. This can remain in its default seting of OnReceivedBopctrl1, so simply press the OK button.



The handler function was created, so press the Edit Code button.



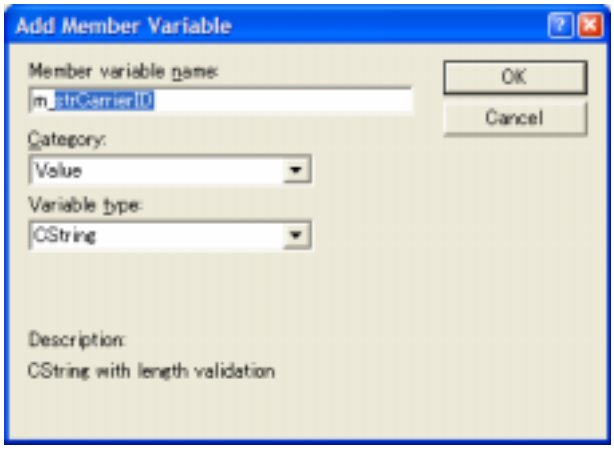
It will jump to OnReceivedBopctrl1(), so write as follows.

```
m_bop.DefProc()
```

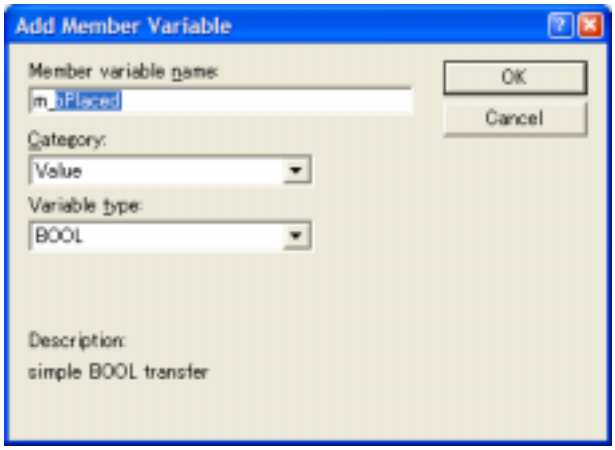
6.3.10 Sending a GEM Event

The event sending process is different from Visual Basic 6.0 and Visual Basic .NET. It is necessary to write accesses to all properties as a function call, as with the method.

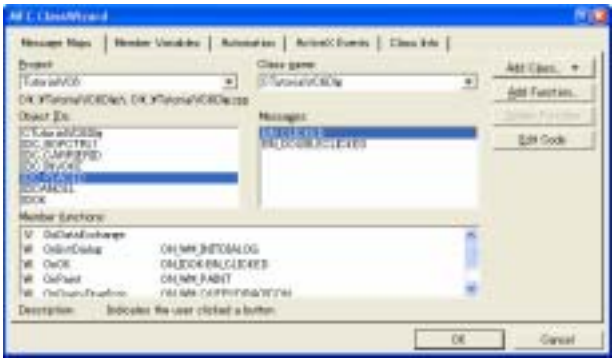
First map the carrier ID to a member variable. Select IDC_CARRIERID from the Class Wizard Member Variables tab, and press the Add Variable button. Enter the variable name in the dialogue box that is displayed and press the OK button. Here we have named it m_strCarrierID.



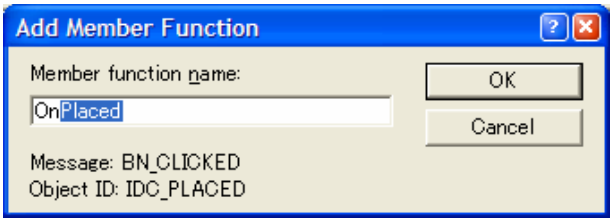
IDC_PLACED is designated as follows.



In the Message Maps tab, select IDC_PLACED の BN_CLICKED and press the Add Function button.



The function name may remain at the default setting, so simply press the OK button.

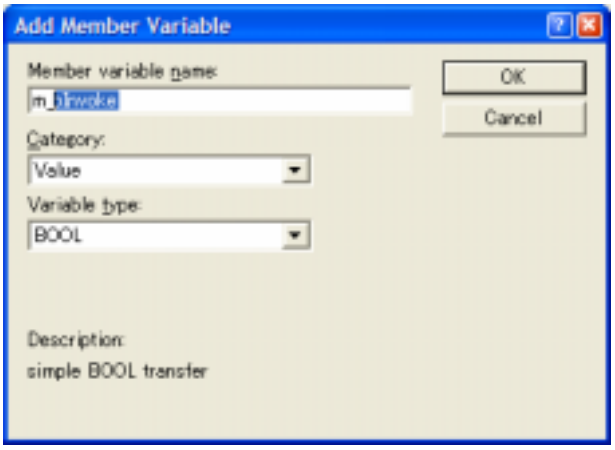


Jump from Class Wizard to OnPlaced(), and write as follows.

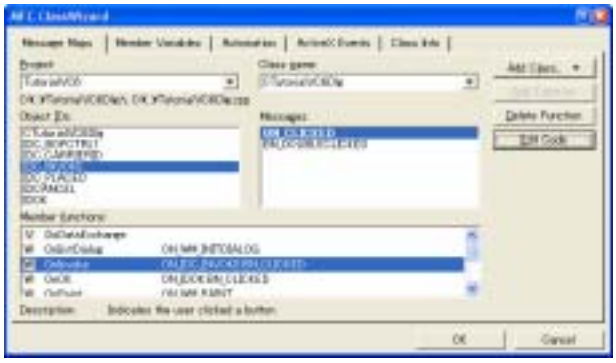
```
UpdateData();

m_bop.SetVIDValue(40,m_strCarrierID);

if(m_bPlaced)
   m_bop.InvokeEvent(200);
else
   m_bop.InvokeEvent(201);
```

6.3.11 Sending an Alarm

The process of sending alarms is also somewhat different, but it is similar to using a GEM event. First, assign a variable name to IDC_INVOKE in Class Wizard.



Create the Click event handler function for IDC_INVOKE in the Message Maps tab and jump.



Write as shown below.

```
UpdateData();

if(m_bInvoke)
   m_bop.InvokeAlarm(30000, -1);
else
   m_bop.InvokeAlarm(30000, 0);
```

6.3.12 Full Source Code

Except for code automatically created by App Wizard, very few lines are needed,similar to Visual Basic 6.0.

```
BOOL CTutorialVC6Dlg::OnInitDialog()
{
   CDialog::OnInitDialog();

   // Set the icon for this dialog.  The framework does
this automatically
   //  when the application's main window is not a dialog
   SetIcon(m_hIcon, TRUE);      // Set big icon
   SetIcon(m_hIcon, FALSE);     // Set small icon

   m_bop.LoadIniFile();
   m_bop.Load();
   m_bop.SetPhysicalConnection(true);

   return TRUE;  // return TRUE  unless you set the focus
to a control
}

void      CTutorialVC6Dlg::OnReceivedBopctrl1(LPCTSTR
lpszIPAddress, long lPortNumber)
{
   m_bop.DefProc();
}

void CTutorialVC6Dlg::OnPlaced()
{
   UpdateData();

   m_bop.SetVIDValue(40,m_strCarrierID);

   if(m_bPlaced)
      m_bop.InvokeEvent(200);
   else
      m_bop.InvokeEvent(201);
}

void CTutorialVC6Dlg::OnInvoke()
{
   UpdateData();

   if(m_bInvoke)
      m_bop.InvokeAlarm(30000, -1);
   else
      m_bop.InvokeAlarm(30000, 0);
}
```

However, since mastering C++ is not easy, if one is not fully confident in that ability we recommend programming in Visual Basic.

7 ActiveX Control Interface

7.1 Properties

7.1.1 ALIDCode

■Description

ALID classification code. Handled as ALCD. Even when a value is set in ALIDCode, only the last 7 bits are recorded. The top bit of ALCD is used to mean generation/clearing of alarms, and this bit is specified by the InvokeAlarm argument.

If ALID is not registered, it is not possible to set a value. The following method is used to obtain a list of registered ALID.

First find how many ALIDCount there are.

```
lCount = .ALIDCount
```

ALIDCount will return the number of registered ALID, so values that can be used as the index are those from 0 to (ALIDCount - 1). Using IndexToALID, this is converted to ALID.

```
lALID = .IndexToALID(lCnt)
```

The ALID has been obtained, so it is possible to access ALIDCode and ALIDDescription.

```
nALCD = .ALIDCode(lALID)
strALTX = .ALIDDescription(lALID)
```

This should be arrayed by repeating the For statement. The full source code will be displayed.

```
Dim lCount As Long
lCount = .ALIDCount

Dim lCnt As Long
For lCnt = 0 to lCount – 1
  Dim lALID As Long
  lALID = .IndexToALID(lCnt)

  Dim nALCD As Integer
  nALCD = .ALIDCode(lALID)

  Dim strALTX As String
  strALTX = .ALIDDescription(lALID)
Next lCnt
```

■Declaration

■Visual C++ 6

```
short GetALIDCode(long lALID);
void SetALIDCode(long lALID, short nNewValue);
```

■Visual Basic 6

```
Property ALIDCode(lALID As Long) As Integer
```

Format	Description
lALID	ALID

■Related Items

ALIDCount, ALIDDescription, IndexToALID, InvokeAlarm

7.1.2 ALIDCount

■Description

Total number of registered ALID. If 0, none are registered.

ALIDCount will return the number of registered ALID, so values that can be used as the index are those from 0 to (ALIDCount - 1). Using IndexToALID, this is converted to ALID.

■Declaration

■Visual C++ 6

```
long GetALIDCount();
```

■Visual Basic 6

```
ALIDCount As Long
```

■Special Notes

Read-only property.

■Related Items

ALIDCode, ALIDDescription, IndexToALID, InvokeAlarm, CEIDCount, VIDCount

7.1.3 ALIDDescription

■Description

ALID Description. With S5F1 Send Alarm Report(ARS), this is sent as ALTX.

In SECS-II, ALTX has a limitation on the maximum number of characters, at 40 characters. However, bop does not have this limitation.

■Declaration

■Visual C++ 6

```
BSTR GetALIDDescription(long lALID);
void SetALIDDescription(long lALID, LPCTSTR lpszNewValue);
```

■Visual Basic 6

```
Property ALIDDescription(lALID As Long) As String
```

Format	Description
lALID	ALID

7.1.4 CEIDCount

■Description

Total number of registered CEID.

CEIDCount will return the number of registered CEID, so the values that can be used as the index are those from 0 to (CEIDCount - 1). Using IndexToCEID, this is converted to CEID.

■Declaration

■Visual C++ 6

```
long GetCEIDCount();
```

■Visual Basic 6

```
CEIDCount As Long
```

■Special Notes

Read-only property.

■Related Items

ALIDCount, VIDCount

7.1.5 CEIDDescription

■Description
CEID Description.

■Declaration
■Visual C++ 6

```
BSTR GetCEIDDescription(long lCEID);
void SetCEIDDescription(long lCEID, LPCTSTR lpszNewValue);
```

■Visual Basic 6

```
Property CEIDDescription(lCEID As Long) As String
```

Format	Description
lCEID	CEID。

7.1.6 Communication

■Description
Communication status model. Communication status is one of the following.

Name	Val.	Description
Disabled	0	Comm disabled
NotCommunicating	1	Comm aborted
Communicating	2	Comm active

■Declaration
■Visual C++ 6

```
short GetCommunication();
void SetCommunication(short nNewValue);
```

■Visual Basic 6

```
Communication As Integer
```

7.1.7 ControlState

■Description
Control status model. Control status is one of the following.

Name	Val.	Description
EquipmentOffLine	0	Tool offline
AttemptOnLine	1	Attempting online
HostOffLine	2	Host offline
OnLineLocal	3	Online local
OnLineRemote	4	Online remote

Some control status transitions are not allowed. For this reason, ControlStateSwitch is used of online/offline switching. Refer to Control Status Model regarding allowed status transitions.

■Declaration
■Visual C++ 6

```
short GetControlState();
void SetControlState(short nNewValue);
```

■Visual Basic 6

```
ControlState As Integer
```

■Related Items
ControlStateSwitch, ControlStateChanged

7.1.8 ControlStateSwitch

■Description
Online/offline switching switch. Some control status transitions are not allowed. For this reason, ControlStateSwitch is used of online/offline switching. Refer to Control Status Model regarding allowed status transitions.

Value	Description
true	Transition from offline to online
false	Transition from online to offline

When a status is actually transitioned a ControlStateChanged event is generated.

■Declaration
■Visual C++ 6

```
void SetControlStateSwitch(BOOL bNewValue);
```

■Visual Basic 6

```
ControlStateSwitch As Boolean
```

■Special Notes
Write-only property

■Related Items
ControlState, ControlStateChanged

7.1.9 DeviceID

■Description
Device ID. SessionID is the first 16 bits of the message header. DeviceID are the 15 bits excluding the top bit of the SessionID.

■Declaration
■Visual C++ 6

```
long GetDeviceID();
void SetDeviceID(long nNewValue);
```

■Visual Basic 6

```
DeviceID As Long
```

■Related Items
SessionID

7.1.10 Discard Duplicated Block

■Description
Specifies whether or not to discard duplicated blocks. If true, if the exact same message is continuously received, messages arriving later will be disregarded. If false, they will not be disregarded, and will be communicated to the application via Received event. It is best to set this to true under normal circumstances.

■Declaration
■Visual C++ 6

```
BOOL GetDiscardDuplicatedBlock();
void SetDiscardDuplicatedBlock(BOOL bNewValue);
```

■Visual Basic 6

```
DiscardDuplicatedBlock As Boolean
```

■Related Items
Received

7.1.11 Function

■Description
Function Number

■Declaration

■Visual C++ 6

```
short GetFunction();
void SetFunction(short nNewValue);
```

■Visual Basic 6

```
Function As Integer
```

■Related Items

Stream

7.1.12 HexDump

■Description

Obtains message as hexadecimal character string.

When a character string is set in SML, bop compiles the character string and builds the data structure inside. At this time, when HexDump is read out, it converts to binary structure and returns it converted to a hexadecimal character string.

■Declaration

■Visual C++ 6

```
BSTR GetHexDump();
void SetHexDump(LPCTSTR lpszNewValue);
```

■Visual Basic 6

```
HexDump As String
```

■Related Items

SML

7.1.13 Host

■Description

Set from the host side to the client side. Since the purpose of the bop product is to mount GEM (tools) , set Host properties as always false.

■Declaration

■Visual C++ 6

```
BOOL GetHost();
void SetHost(BOOL bNewValue);
```

■Visual Basic 6

```
Host As Boolean
```

7.1.14 IniFile

■Description

Ini file name for saving settings. Please note that when full path name is specified as relative path name, the ini file fill be created in that folder, but if only a file name is specified, it will be created in the Windows folder. If we wish to created it in the current folder, please specify as follows.

```
.IniFile = "./bop.ini"
```

■Declaration

■Visual C++ 6

```
BSTR GetIniFile();
void SetIniFile(LPCTSTR lpszNewValue);
```

■Visual Basic 6

```
IniFile As String
```

7.1.15 IPAddress

■Description

Continuing IP address. In the case of a passive entity, it is its own IP address, so it does not need to be specified. In the case of an active entity, specify the connection destination's IP address.

Property	Passive	Active
IPAddress	Not required	Required
PortNumber	Not required	Required
LocalPortNumber	Required	Required (Normally 0)

■Declaration

■Visual C++ 6

```
BSTR GetIPAddress();
void SetIPAddress(LPCTSTR lpszNewValue);
```

■Visual Basic 6

```
IPAddress As String
```

■Related Items

PortNumber, LocalPortNumber

7.1.16 LocalPortNumber

■Description

Local port number. In the case of a passive entity, this is a public port number with respect to the client.

In the case of an active entity, this is normally specified as 0. When 0 is specified, an open port will automatically be selected. When something other than 0 is specified, it cannot connect unless that particular port number is open. In Windows, even if connection is broken, for a short while (several minutes) that port number is still appropriated. Please note that for this reason, if something other than 0 is specified, connection will not be possible for several minutes.

Property	Passive	Active
IPAddress	Not required	Required
PortNumber	Not required	Required
LocalPortNumber	Required	Required (Normally 0)

■Declaration

■Visual C++ 6

```
long GetLocalPortNumber();
void SetLocalPortNumber(long nNewValue);
```

■Visual Basic 6

```
LocalPortNumber As Long
```

■Related Items

IPAddress, PortNumber

7.1.17 LogFileBakCount

■Description

Number of log file back-up files. The log file name is as shown below.

File Name	Description
XXXXX.log	Newest log file in terms of time
XXXXX001.log	Back-up file that is one prior to the above, in terms of time
XXXXX002.log	Back-up file that is two prior to the above, in terms of time.
...	...

■Declaration

■Visual C++ 6

```
short GetLogFileBakCount();
void SetLogFileBakCount(short nNewValue);
```

■Visual Basic 6

```
LogFileBakCount As Integer
```

■Related Items

LogFileName, LogFileSize, LogFileEnableCommunication, LogFileEnable

7.1.18 LogFileEnable

■Description

Specifies whether or not to record log files. When this property is true files are recorded, and when false they are not recorded.

■Declaration

■Visual C++ 6

```
BOOL GetLogFileEnable();
void SetLogFileEnable(BOOL bNewValue);
```

■Visual Basic 6

```
LogFileEnable As Boolean
```

■Related Items

LogFileName, LogFileSize, LogFileEnableCommunication, LogFileBakCount

7.1.19 LogFileEnableCommunication

■Description

Specifies whether or not to record communication module logs in files. When this property is true they are recorded in files, and when false they are not recorded.

■Declaration

■Visual C++ 6

```
BOOL GetLogFileEnableCommunication();
void SetLogFileEnableCommunication(BOOL bNewValue);
```

■Visual Basic 6

```
LogFileEnableCommunication As Boolean
```

■Related Items

LogFileName, LogFileSize, LogFileEnable, LogFileBakCount

7.1.20 LogFileName

■Description

Log file name. When specifying a log file name, a suffix is not added. The suffix ".log" will be added automatically.

File Name	Description
XXXXX.log	Newest log file in terms of time
XXXXX001.log	Back-up file that is one prior to the above, in terms of time
XXXXX002.log	Back-up file that is 2 prior to the above, in terms of time
...	...

■Declaration

■Visual C++ 6

```
BSTR GetLogFileName();
void SetLogFileName(LPCTSTR lpszNewValue);
```

■Visual Basic 6

```
LogFileName As String
```

■Related Items

LogFileName, LogFileSize, LogFileEnableCommunication, LogFileBakCount

7.1.21 LogFileSize

■Description

Log file size. When this file size is exceeded, a back-up file will be created.

Units are KB (kilobytes). This means that for example, if 1024 is specified, the size will be 1MB (megabyte).

■Declaration

■Visual C++ 6

```
long GetLogFileSize();
void SetLogFileSize(long nNewValue);
```

■Visual Basic 6

```
LogFileSize As Long
```

■Related Items

LogFileName, LogFileBakCount, LogFileBakCount, LogFileEnableCommunication

7.1.22 LogicalConnection

■Description

Type of logical connection. The following items are defined.

Value	Description
0	Basic model. No particular processing occurs.
1	GEM model.

We are using bop to mount GEM, so the LogicalConnection is always 1.[2]

■Declaration

■Visual C++ 6

```
short GetLogicalConnection();
void SetLogicalConnection(short nNewValue);
```

■Visual Basic 6

```
LogicalConnection As Integer
```

7.1.23 LogicalConnectionFileName

■Description

File name in which to save setting contents for logical connection. No suffix is to be specified (".bop" will be automatically added). Default is bop.bop (property

[2] In the current bop version, always set the LogicalConnection property to 1.。

value is “./bop”).

In GEM, we are requesting that report definitions, etc. are recorded in non-volatile memory media. For this reason, LogicalConnectionFileName must be used.

■Declaration

■Visual C++ 6

```
BSTR GetLogicalConnectionFileName();
void SetLogicalConnectionFileName(LPCTSTR lpszNewValue);
```

■Visual Basic 6

```
LogicalConnectionFileName As String
```

7.1.24 Node

■Description

Node specifier. A node is the operand in the message structure.

Folders in a hard disk can be given a “tree structure” on a PC. A folder is created and files are placed inside of it. Not only files but also folders can be made inside of a folder. In a tree format, the folder is the “branch” and the files are the “leaves”.

In the same way, an SML data structure can also be given a tree structure. The list will be the “branch”, and the other items will be the “leaves”. The node is a classifier which identifies the “branch” and “leaf” positions.

Nodes are made up of slashes (“ / ”) and the node number. If a node is empty (“ ”), the root will be considered what was specified. In general roots are in list format, but other formats may also be specified. The root must be in list format when there are sub-nodes.

For example, let us assume an SML as follows.

```
{
  <a‘Kelly’>
  {
    <a‘Brenda’>
    {
      <a‘Donna’>
    }
  }
  <a‘Valerie’>
  {
    {
      {
        <a‘Andrea’>
      }
    }
  }
}
```

Let us assign numbers so it is easy to identify nodes.

```
1 {
  1 <a‘Kelly’>
  2 {
    1 <a‘Brenda’>
    2 {
      1 <a‘Donna’>
      }
    }
  3 <a‘Valerie’>
  4 {
    1 {
      1 {
        1 <a‘Andrea’>
        }
      }
    }
}
```

If Kelly, Brenda, Donna, Valerie, and Andrea are shown denoted as node positions, they would be as follows. There are no restrictions on the node nesting level.

Value	Node
Kelly	1
Brenda	2/1
Donna	2/2/1
Valerie	3
Andrea	4/1/1/1

If the node is in array format, use [] to extract identifiers and specify and index. For example, let us assume an SML such as that below.

```
{
  <f8 9.11 3.14>
}
```

If we wished to extract the second element, “3.14”, we would specify the index. Since the index starts from 0, if it is the 2nd element, it would be “1”.

```
.Node = “1[1]”
```

When this is read out, only the identifier returns, as shown below.

```
“3.14”
```

■Declaration

■Visual C++ 6

```
BSTR GetNode();
void SetNode(LPCTSTR lpszNewValue);
```

■Visual Basic 6

```
Node As String
```

■Related Items

NodeCount, NodeType, NodeValue, NodeValueHex

7.1.25 NodeCount

■Description

Number of node items. Some nodes are in array format. Lists are the most common example.

In the example below, since the list has 3 sub-nodes, the NodeCount is 3. If you do not know the number of list elements beforehand, etc. it is good to first read out NodeCount and loop it for the number of repetitions.

```
{
  <a‘Yoda’>
  <a‘R2-D2’>
  <a‘C-3PO’>
}
```

In the example below, there is one u2 item, 2 u4 items, and 3 f8 items in the NodeCount.

```
{
  <u2 99>
  <u4 1024 4096>
  <f8 1.05 2.26 3.14>
```

```
}
```

The NodeCount of the following list is 0.

```
{
}
```

■Declaration

■Visual C++ 6

```
long GetNodeCount();
```

■Visual Basic 6

```
NodeCount As Long
```

■Special Items

Read-only property

7.1.26 NodeType

■Description

Nodes are one of the following types.

Node Format	Val.	Description
NodeTypeList	1	List
NodeTypeBinary	2	Binary
NodeTypeBoolean	3	Boolean
NodeTypeAscii	4	ASCII character string
NodeTypeJis	5	JIS-8 code
NodeTypeLong8	6	8-byte signed integer
NodeTypeChar	7	1-byte signed integer
NodeTypeShort	8	2-byte signed integer
NodeTypeLong	9	4-byte signed integer
NodeTypeDouble	10	8-byte floating pt no.
NodeTypeFloat	11	4 byte floating pt no.
NodeTypeDWord8	12	8 byte floating pt no.
NodeTypeByte	13	1 byte floating pt no.
NodeTypeWord	14	2 byte floating pt no.
NodeTypeDWord	15	4 byte floating pt no.
NodeTypeAscii2	16	2 byte ASCII char. string

When NodeType is read out, if the value is 0, it means it is an “invalid format”.

■Declaration

■Visual C++ 6

```
short GetNodeType();
```

■Visual Basic 6

```
NodeType As Integer
```

■Special Notes

Read-only property

7.1.27 NodeValue

■Description

Node value. Binary format, numerical format (signed integers, unsigned integers, floating point numbers) will be converted to decimal expression character strings. Boolean format “true” will be converted to “1” and “false” will be converted to “0”.

In the case of an item array, (NodeCount greater than 1), each element is separated by one character of space code. For example, let’s assume the SML message below.

```
{
    <u2 333 444 555>
}
```

To read out this u2 item, specicy “1” in Node.

```
.Node = “1”
```

When read out, a character string as shown below will return.

```
“333 444 555”
```

If we only wish to read out specific elements of the array, specify the index using [].

```
.Node = “1[1]”
```

When this is read, the specific elements only will return, as shown below.

```
“444”
```

■Declaration

■Visual C++ 6

```
BSTR GetNodeValue();
```

■Visual Basic 6

```
NodeValue As String
```

■Special Items

Read-only property

■Related Items

NodeValueHex

7.1.28 NodeValueHex

■Description

Node value in hexadecimal expression. For example, let’s assume the following SML message.

```
<u2 254>
```

When read out, the following character string will return.

```
“fe”
```

■Declaration

■Visual C++ 6

```
BSTR GetNodeValueHex();
```

■Visual Basic 6

```
NodeValueHex As String
```

■Special Notes

Read-only property

■Related Items

NodeValue

7.1.29 Request Offline

■Description

Specifies whether or not to accept S1F15 Request Offline(ROFL). When this property is true, offline requests are accepted, and OFLACK=0 is returned by S1F16 Offline Request Acknowledge acknowledgment(OFLA). When false, requests are denied and OFLACK=1 is returned.

■Declaration

■Visual C++ 6

```
BOOL GetOfflineRequest();
void SetOfflineRequest(BOOL bNewValue);
```

■Visual Basic 6

```
OfflineRequest As Boolean
```

■Related Items

OnlineRequest

7.1.30 OnlineRequest

■Description

Specifies whether or not to accept S1F17 online request(RONL). When this property is true, online requests are accepted and ONLACK=0 is returned by S1F18 Online Request Acknowledge(ONLA). When false, requests are denied and ONLACK=1 is returned.

■Declaration

■Visual C++ 6

```
BOOL GetOnlineRequest();
void SetOnlineRequest(BOOL bNewValue);
```

■Visual Basic 6

```
OnlineRequest As Boolean
```

■Related Items

OfflineRequest

7.1.31 PassiveEntity

■Description

Specifies whether passive entity (server9 or active entity (client). When this property is true the setting is to passive entity, and when false to active entity.

Value	Description
true	Passive entity
false	Active entity

There is not clear standard as to whether a tool is A passive or an active entity. However, since there are many cases in which setting the tool side as a passive entity is logical, in modern-day interpretations it is more common to set tools as passive entities. Bop can accommodate both settings.

■Declaration

■Visual C++ 6

```
BOOL GetPassiveEntity();
void SetPassiveEntity(BOOL bNewValue);
```

■Visual Basic 6

```
PassiveEntity As Boolean
```

7.1.32 PhysicalConnection

■Description

Specifies whether or not the physical connection Model is enabled or not. If this property is true, the physical connection model is enabled, and if false, it is disabled.

By setting this property as true, communication ports are opened, and communication becomes possible. However, the communication ports are actually opened not at the moment the property is set to true, but rather immediately afterward. For this reason, even if the following type of program is written, it is not possible to know whether or not the port was opened.

```
.PhysicalConnection = True
If .PhysicalConnection Then
  ...
```

■Declaration

■Visual C++ 6

```
BOOL GetPhysicalConnection();
void SetPhysicalConnection(BOOL bNewValue);
```

■Visual Basic 6

```
PhysicalConnection As Boolean
```

7.1.33 PortNumber

■Description

Connecting port number. In the case of a passive entity, since it is not the case that it will go to connect with its partner, there is no need to specify (it will be disregarded). In the case of an active entity, specify the connection's destination port number.

Property	Passive	Active
IPAddress	Not required	Required
PortNumber	Not required	Required
LocalPortNumber	Required	Required (Normally 0)

■Declaration

■Visual C++ 6

```
long GetPortNumber();
void SetPortNumber(long nNewValue);
```

■Visual Basic 6

```
PortNumber As Long
```

■Related Items

IPAddress, LocalPortNumber

7.1.34 PType

■Description

HSMS P type. If the message content is SECS-II, thisis 0. However, since at the present time no other P type than 0 has been provided for, anything but 0 will cause an error.

■Declaration

■Visual C++ 6

```
short GetPType();
void SetPType(short nNewValue);
```

■Visual Basic 6

PType As Integer

7.1.35 Reply

■Description

Selects the reply portion of operand messages.When a message is received via Received Event, it will be set in WorkSpace0. If we wish to analyze a received message, we can process it as-is. If we wish to respond to a received message, set Reply to true, and edit the reply portion. After this, when DefProc is called by the event handler function, messages requiring replies will send the reply message written in the reply section.

If we wish to send a message not from the Received Event handler but from the normal process, we create a message in Reply=false of WorkSpace0.

■Declaration

■Visual C++ 6

```
BOOL GetReply();
void SetReply(BOOL bNewValue);
```

■Visual Basic 6

```
Reply As Boolean
```

■Related Items

WorkSpace, Send, Received

7.1.36 SessionID

■Description

Section ID. The SessionID is the first 16 bits of the message header. DeviceID is the 15 bits, excluding the topmost bit, of SessionID.

■Declaration

■Visual C++ 6

```
long GetSessionID();
void SetSessionID(long nNewValue);
```

■Visual Basic 6

```
SessionID As Long
```

■Related Items

DeviceID

7.1.37 SML

■Description

Message character string expression. SECS-II messages are in binary structure, and cannot be comprehended by a human as-is. In order to make it more understandable, we express it by an ASCII character string.

SML Expression	Description
l	List
b	Binary
bool	Boolean
a	ASCII char. string
j	JIS-8
a2	2-byte ASCII char. string
i8	8-byte signed integer
i1	1-byte signed integer
i2	2-byte signed integer
i4	4-byte signed integer
f8	8-byte floating pt. number
f4	4-byte floating pt. number
u8	8-byte unsigned integer
u1	1-byte unsigned integer
u2	2-byte unsigned integer
u4	4-byte unsigned integer

When a character string is set in SML, bop compiles the character string, and builds a data structure inside it. At this time, if HexDump is read out, it converts to binary structure, and returns that converted into a hexadecimal character string. When SML is read out, a character string is generated from the bop's internal data structure.

When creating a message, first create the message's SMLcharacter string, and set that in SML. Refer to SML Reference for details on SML syntax.

■Declaration

■Visual C++ 6

```
BSTR GetSML();
void SetSML(LPCTSTR lpszNewValue);
```

■Visual Basic 6

```
SML As String
```

■Related Items

HexDump, SML Reference

7.1.38 Stream

■Description

Stream number.

■Declaration

■Visual C++ 6

```
short GetStream();
void SetStream(short nNewValue);
```

■Visual Basic 6

```
Stream As Integer
```

■Related Items

Function

7.1.39 SType

■Description

HSMS S type. S types provided are the following.

S Type	Meaning	Description
0	Data Message	Normal SECS-II message having streams/functions
1	Select.req	Select request
2	Select.rsp	Select request response
3	Deselect.req	Deselect request
4	Deselect.rsp	Deselect request response
5	Linktest.req	Link test request
6	Linktest.rsp	Link test request response
7	Reject.req	Reject request
9	Separate.req	Separate request

In HSMS-SS, Deselect.req and Deselect.rsp are not used.

■Declaration

■Visual C++ 6

```
short GetSType();
void SetSType(short nNewValue);
```

■Visual Basic 6

```
SType As Integer
```

7.1.40 SystemBytes

■Description

System bytes. System bytes are the last 4 bytes Among the 10 bytes in the SECS header. As long as this value is unique to the transaction waiting for a reply (open transaction) it can be anything, but normally it would be good to have it increase by +1 for each primary message sent.

The secondary message's SystemBytes must be the same as that of the primary message.

■Declaration

■Visual C++ 6

```
long GetSystemBytes();
void SetSystemBytes(long nNewValue);
```

■Visual Basic 6

```
SystemBytes As Long
```

7.1.41 T1

■Description

SECS-I T1 timeout. The unit is a millisecond.

■Declaration

■Visual C++ 6

```
short GetT1();
void SetT1(short nNewValue);
```

■Visual Basic 6

```
T1 As Integer
```

■Special Notes

This property is an item for use by SECS-I, so it is not used.

7.1.42 T2

■Description

SECS-I T2 timeout. The unit is a millisecond.

■Declaration

■Visual C++ 6

```
short GetT2();
void SetT2(short nNewValue);
```

■Visual Basic 6

```
T2 As Integer
```

■Special Items

This property is an item for use by SECS-I, so it is not used.

7.1.43 T3

■Description

T3 timeout. Also called transaction timeout. The unit is a second.

Time from sending of the primary message to receipt of the secondary message. If more than this amount of time elapses, a T3 timeout occurs, and the tool will send S9F9 Transaction Timer Timeout(TIN).

■Declaration

■Visual C++ 6

```
short GetT3();
void SetT3(short nNewValue);
```

■Visual Basic 6

```
T3 As Integer
```

7.1.44 T4

■Description

SECS-I T4 timeout. The unit is a second.

■Declaration

■Visual C++ 6

```
short GetT4();
void SetT4(short nNewValue);
```

■Visual Basic 6

```
T4 As Integer
```

■Special Notes

This property is an item for use by SECS-I, so it is not used.

7.1.45 T5

■Description

HSMS T5 timeout. Also called a connection separation timeout. The unit is a second.

This property only relates in the case of an active entity.This amount of time or more must be waited if a connection has failed to be made, or was disconnected.

■Declaration

■Visual C++ 6

```
short GetT5();
void SetT5(short nNewValue);
```

■Visual Basic 6

```
T5 As Integer
```

7.1.46 T6

■Description

HSMS T6 timeout. Also called control transaction timeout.The uit is a second.

In control messages (message with an S type other than 0), this is the time between sending of a request and receipt of a response. T3 timeout monitors the transaction versus the data message, and this could be called the control message version of that.

Timing occurs when the following times time out, specifically.

S Type	Description
1	Time from sending of Select.req to receipt of Select.rsp.
3	Time from sending of Deselect.req to receipt of Deselect.rsp.
5	Time from sending of Linktest.req to receipt of Linktest.rsp.

■Declaration

■Visual C++ 6

```
short GetT6();
void SetT6(short nNewValue);
```

■Visual Basic 6

```
T6 As Integer
```

7.1.47 T7

■Description

HSMS T7 timeout. Also called NOT SELECTED timeout. The unit is a second.

If the TCP/IP level is connected via HSMS but it is left without transitioning to the Selected status, it will disconnect after a preset time has elapsed. Also, even if it transitions from Selected status to Deselected status via Deselect.req, if it is left without restoring to Selected status, it will also be disconnected.

■Declaration

■Visual C++ 6

```
short GetT7();
void SetT7(short nNewValue);
```

■Visual Basic 6

```
T7 As Integer
```

7.1.48 T8

■Description

HSMS T8 timeout. Also called network character timeout. The unit is a second.

Even if the HSMS connection is not cut off, if data is broken off for a short while during receipt of one message, it will be unable to determine whether or not what comes next is a continuation of the message. If the time allotted by the T8 timer elapses, it will be deemed a “communication failure”, and the result will be disconnection.

It is similar to a SECS-I T1 timeout.

■Declaration

■Visual C++ 6

```
short GetT8();
void SetT8(short nNewValue);
```

■Visual Basic 6

```
T8 As Integer
```

7.1.49 Verification

■Description

Result of verification of a received message. This is one of the following.

Result	Val.	Description
Correct	0	In compliance with SEMI E.5 (SECS-II) .
UserDefined	1	User-defined message.
Incorrect	2	Not in compliance.
IncorrectAndReply	3	Not in compliance, but reply is via secondary message rather than S9F7
NOWBit	4	Required W bit absent
WBit	5	Unneeded W bit present
WrongDirection	6	Message direction is backwards
UnrecognizedStream	7	Undefined stream
UnrecognizedFunction	8	Undefined function

When a message is received, the message structure is verified within bop. The result of this check is set in Verification, and a Received Event is generated.

■Declaration

■Visual C++ 6

```
short GetVerification();
void SetVerification(short nNewValue);
```

■Visual Basic 6

```
Verification As Integer
```

■Related Items

Received Event

7.1.50 VIDCount

■Description

Total number of registered VID.

VIDCount returns the number of registered VID, so the values which can be used for the index are those from 0 to (VIDCount - 1). Using IndexToVID, this is converted to VID.

■Declarations

■Visual C++ 6

```
long GetVIDCount();
```

■Visual Basic 6

```
VIDCount As Long
```

■Special Notes

Read-only property

■Related Items

ALIDCount, CEIDCount

7.1.51 VIDDefault

■Description

VID default value. ECDEF of S2F30 Equipment Constant Namelist(ECN).

■Declaration

■Visual C++ 6

```
BSTR GetVIDDefault(long lVID);
void SetVIDDefault(long lVID, LPCTSTR lpszNewValue);
```

■Visual Basic 6

```
Property VIDDefault(lVID As Long) As String
```

Format	Description
lVID	VID。

■Related Items

S2F30 Equipment Constant Namelist(ECN)

7.1.52 VIDDescription

■Description

VID Description. Depending on the VIDType,the way in which this property is treated varies slightly.

In the case of SVID, it is treated as an SVNAME of S1F12 Status Variable Namelist Reply (SVNRR). In the case of ECID, it is treated as an ECNAME of S2F30 Equipment Constant Namelist(ECN).

In the case of DVID, we would want to treat it as a DVNAME, but unfortunately in GEM, since neither S6F4 or S6F8 are defined, this is not automatically processed in bop either. However, it is possible for users to use VIDDescription to make it able to handle S6F4 and S6F8.

■Declaration

■Visual C++ 6

```
BSTR GetVIDDescription(long lVID);
void SetVIDDescription(long lVID, LPCTSTR lpszNewValue);
```

■Visual Basic 6

```
Property VIDDescription(lVID As Long) As String
```

Format	Description
lVID	VID。

■Related Items

S1F12 Status Variable Namelist Reply(SVNRR), S2F30 Equipment Constant Namelist(ECN)

7.1.53 VIDMax

■Description

Maximum value of VID. ECMAX of S2F30 Equipment Constant Namelist(ECN).

■Declaration

■Visual C++ 6

```
BSTR GetVIDMax(long lVID);
void SetVIDMax(long lVID, LPCTSTR lpszNewValue);
```

■Visual Basic 6

```
Property VIDMax(lVID As Long) As String
```

Format	Description
lVID	VID。

■Related Items

S2F30 Equipment Constant Namelist(ECN)

7.1.54 VIDMin

■Description

Minimum value of VID. ECMIN of S2F30 Equipment Constant Namelist(ECN).

■Declaration

■Visual C++ 6

```
BSTR GetVIDMin(long lVID);
void SetVIDMin(long lVID, LPCTSTR lpszNewValue);
```

■Visual Basic 6

```
Property VIDMin(lVID As Long) As String
```

Format	Description
lVID	VID。

■Related Items

S2F30 Equipment Constant Namelist(ECN)

7.1.55 VIDNodeType

■Description

VID SECS-II node format. It is one of the following.

Node Format	Val.	Description
NodeTypeList	1	List
NodeTypeBinary	2	Binary
NodeTypeBoolean	3	Boolean
NodeTypeAscii	4	ASCII char. string
NodeTypeJis	5	JIS-8 code
NodeTypeLong8	6	8-byte signed integer
NodeTypeChar	7	1-byte signed integer
NodeTypeShort	8	2-byte signed integer
NodeTypeLong	9	4-byte signed integer
NodeTypeDouble	10	8-byte floating pt. no.
NodeTypeFloat	11	4-byte floating pt. no.
NodeTypeDWord8	12	8-byte unsigned integer
NodeTypeByte	13	1-byte unsigned integer
NodeTypeWord	14	2-byte unsigned integer
NodeTypeDWord	15	4-byte unsigned integer
NodeTypeAscii2	16	2-byte ASCII char. string

When VIDNodeType is read, if the value is 0 it means"disabled format".

■Declaration

■Visual C++ 6

```
short GetVIDNodeType(long lVID);
void SetVIDNodeType(long lVID, short nNewValue);
```

■Visual Basic 6

```
Property VIDNodeType(lVID As Long) As Integer
```

Format	Description
lVID	VID。

■Related Items

NodeType

7.1.56 VIDRawValue

■Description

VID raw SML character string. If a n SML character string is set here, when an event is sent by S6F11 Send Event Report(ERS), the message will be assembled in accordance with the report. At this time, the individual VID values are used by this VIDRawValue. Stream and function cannot be written in SML.

In bop, since it does not check for grammatical errors and VIDNodeType are not checked, so it is necessary to set the correct SML. To set values in accordance with the VIDNodeType format, use VIDValue.

■Declaration

■Visual C++ 6

```
BSTR GetVIDRawValue(long lVID);
void SetVIDRawValue(long lVID, LPCTSTR lpszNewValue);
```

■Visual Basic 6

```
Property VIDRawValue(lVID As Long) As String
```

Format	Description
lVID	VID。

■Related Items

SML, VIDValue, VIDNodeType

7.1.57 VIDType

■Description

VID type. This is one of the following.

Type	Description
1	ECID。
2	SVID。
4	DVID。

It is not possible to specify a type other than the above.

■Declaration

■Visual C++ 6

```
short GetVIDType(long lVID);
void SetVIDType(long lVID, short nNewValue);
```

■Visual Basic 6

```
Property VIDType(lVID As Long) As Integer
```

Format	Description
lVID	VID。

7.1.58 VIDUnit

■Description

VID unit. UNITS in S2F30 Equipment Constant Namelist(ECN).

■Declaration

■Visual C++ 6

```
BSTR GetVIDUnit(long lVID);
void SetVIDUnit(long lVID, LPCTSTR lpszNewValue);
```

■Visual Basic 6

```
Property VIDUnit(lVID As Long) As String
```

Format	Description
lVID	VID。

■Related Items

S2F30 Equipment Constant Namelist(ECN)

7.1.59 VIDValue

■Description

VID value. SML is created in VIDRawValue in accordance with the format specified in VIDNodeType.

■Declaration

■Visual C++ 6

```
BSTR GetVIDValue(long lVID);
void SetVIDValue(long lVID, LPCTSTR lpszNewValue);
```

■Visual Basic 6

```
Property VIDValue(lVID As Long) As String
```

Format	Description
lVID	VID。

7.1.60 ViewStyle

■Description

Screen display style. This is one of the following.

Style	Val.	Description
RedrawNone	0	Nothing displayed.
RedrawHsms	1	Only HSMS physical connection status displayed.
RedrawGem	2	Only GEM logical connection status displayed.
RedrawNormal	3	Everything displayed.

■Declaration

■Visual C++ 6

```
short GetViewStyle();
void SetViewStyle(short nNewValue);
```

■Visual Basic 6

```
ViewStyle As Integer
```

7.1.61 WaitBit

■Description

W (Wait) bit.

■Declaration

■Visual C++ 6

```
BOOL GetWaitBit();
void SetWaitBit(BOOL bNewValue);
```

■Visual Basic 6

```
WaitBit As Boolean
```

7.1.62 WorkSpace

■Description

Working area of operand message. In bop, 3 WorkSpace areas have been provided. Each one is set to “primary message” Reply=true and “secondary message” Reply=false, so it is possible to handle a total of 6 messages. However, it is basically set up to be able to process just using WorkSpace0.

WorkSpace	Reply	Purpose
0	false	Send/receive message
0	true	Reply to received message
1	false	Sent messages
1	true	Can be set as desired
2	false	Can be set as desired
2	true	Can be set as desired

When a message is received and communicated about via Received Event , the received message is stored in 、WorkSpace0 Reply=false. Please note that even if the received message is a secondary message, it will be stored in Reply=false. The moment there is notification of an event, WorkSpace0 Reply=false is selected, but when processing returns from the event handler to bop, it will return to the previously selected WorkSpace and Reply.

For primary messages, the “recommended reply” is alsostored in Reply=true. If replying in this status, it is OK just to switch Reply to true and call Send, but it is also possible to edit the contents.

When sending a primary message, use WorkSpace0 Reply=false. When sending is complete, this will be communicated by a Sent Event, and at that time, WorkSpace1 Reply=false will be selected.

■Declaration

■Visual C++ 6

```
short GetWorkSpace();
void SetWorkSpace(short nNewValue);
```

■Visual Basic 6

```
WorkSpace As Integer
```

■Related Items

Reply, Send, Received

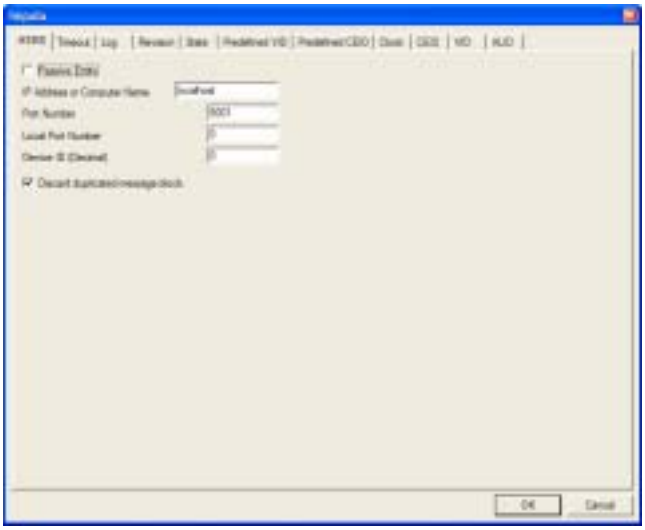
7.2 Method

7.2.1 Configure

■Description

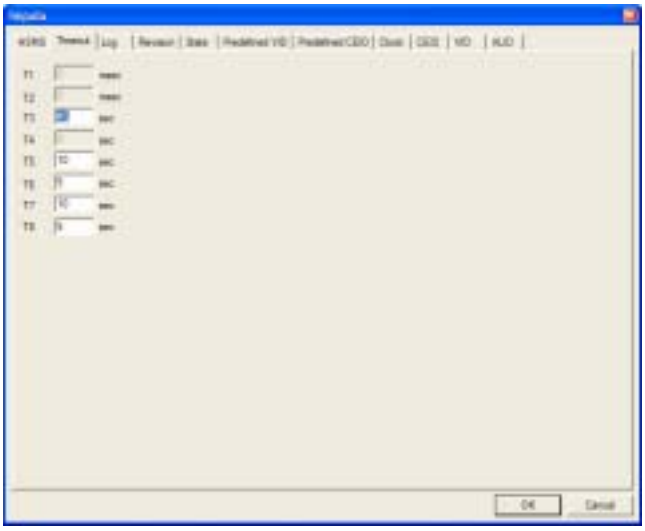
Displays setting screen

■HSMS Tab



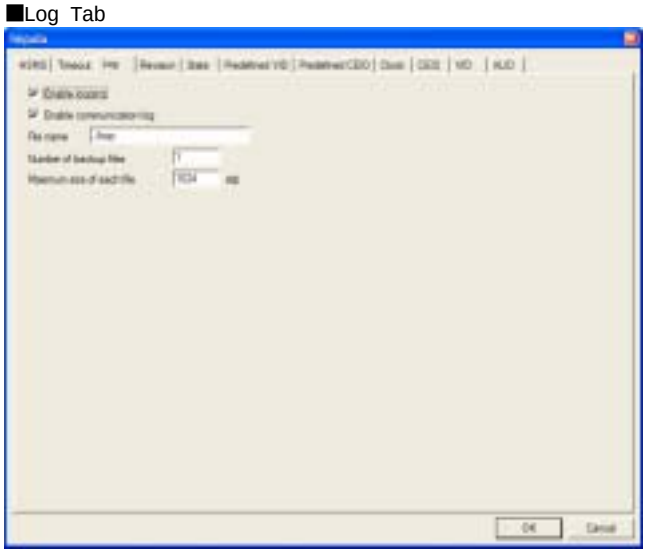
Item	Description
Passive Entity	Checkmark indicates a passive entity (server). No checkmark indicates an active entity (client)
IP Address or Computer Name	Selects whether IP address (xxx.xxx.xxx.xxx) or computer name. This setting is unnecessary for a passive entity, so entry of this item will not be possible.
Port Number	Port number of other side. Since this is not known until a connection is made from the other side in the case of a passive entity, entry of this item will not be possible.
Local Port Number	Local (self) port number. If 0 is set in the case of an active entity, an open port number will be assigned automatically. If other than 0 is specified, it will not be possible to re-connect for several minutes in some cases.
Device ID (Decimal)	Device ID. Settable within the range of 0～32767.
Discard duplicated message block.	When checked, if the same message is continuously received, messages arriving later will be disregarded.

■Timeout Tab



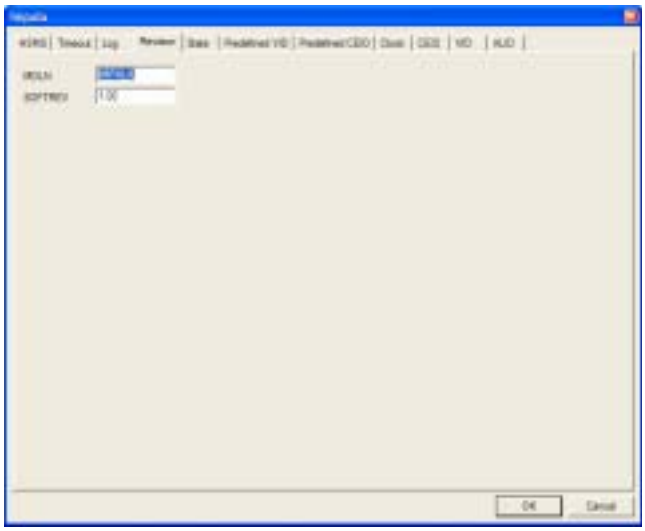
Item	Description
T1	(Not used)
T2	(Not used)
T3	T3 timeout in 1-second increments
T4	(Not used)
T5	T5 timeout in 1-second increments
T6	T6 timeout in 1-second increments
T7	T7 timeout in 1-second increments
T8	T8 timeout in 1-second increments

■Log Tab



Item	Description
Enable logging	When checked, will write to log file. When unchecked, the following items cannot be entered.
Enable communication log	When checked, will record to communication log.
File name	Log file name. No suffix may be added. The suffix .log will be added automatically.
Number of backup files	Number of back-up files. Back-up filenames are as follows. Filename001.log Filename002.log
Maximum size of each file	Maximum size of log files. If this size is exceeded, a back-up file will be created. In 1-kilobyte units.

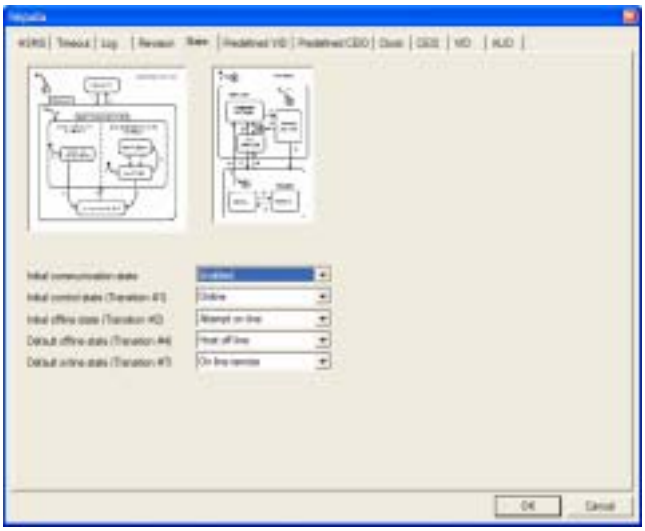
■Revision Tab



Item	Description
MDLN	MDLN character string in messages such as S1F13 Establish Communication Request(CR). Equipment model name. max 6 bytes.

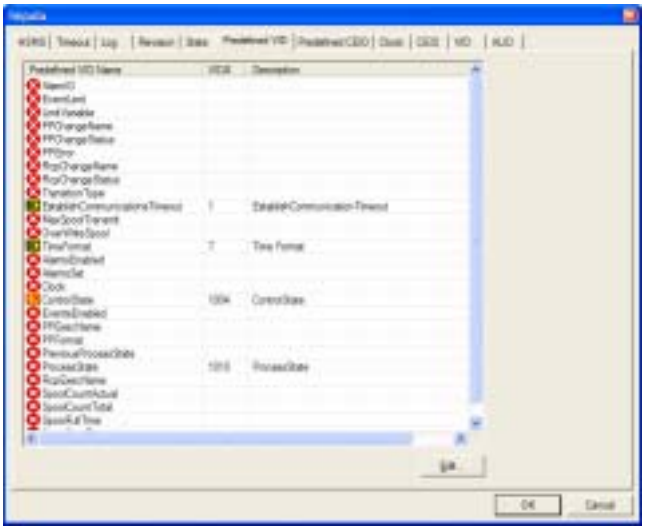
SOFTREV	SOFTREV character string. Revision (version) number. Max 6 bytes.

■State Tab

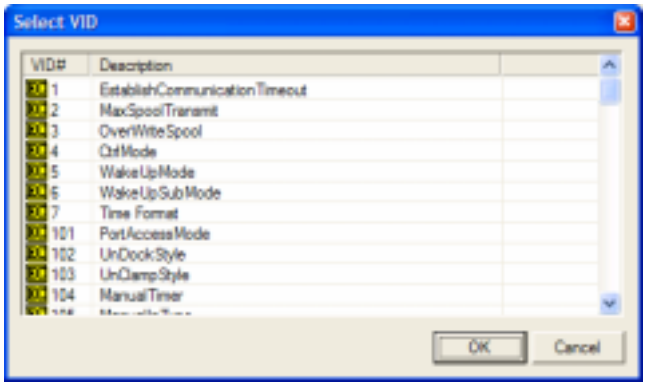


Item	Description
Initial communication state	Comm status at time of application launch. Select one of the following. Disabled Enabled
Initial control state (Transition #1)	Control status at time of launch. Status of transition at Status Transition #1. Select one of the following. Offline Online
Initial offline state (Transition #2)	Offline status at time of launch. Status of transition at Status Transition #2. Select one of the following. Equipment offline Attempt online Host offline
Default offline state (Transition #4)	Default offline status. Status of transition at Status Transition #4. Select one of the following. Equipment offline Host offline
Default online state (Transition #7)	Default online status. Status of transition at Status Transition #7. Select one of the following. Online local Online remote

■Predefined VID Tab



Item	Description
Predefined VID Name	Defined VID name
VID#	VID number
Description	Description
Edit...	"Edit" button. When one defined VID is selected and this button is pressed, the following dialogue box will be displayed.

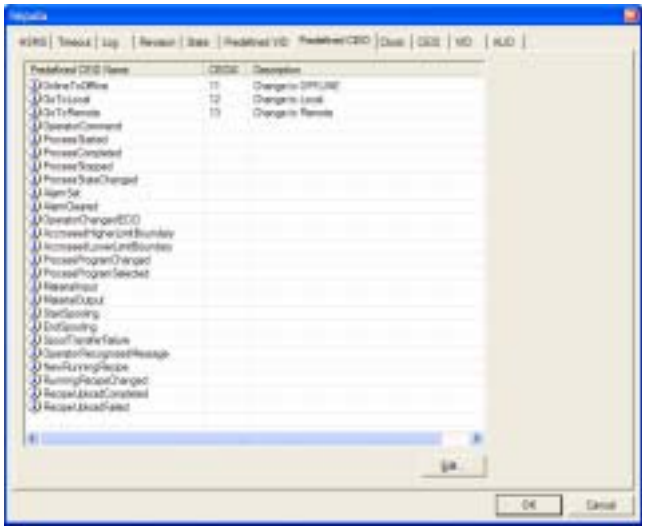


Here, a list of defined VID is shown in the VID tab. Select a VID with the same meaning as a defined VID and press the OK button.

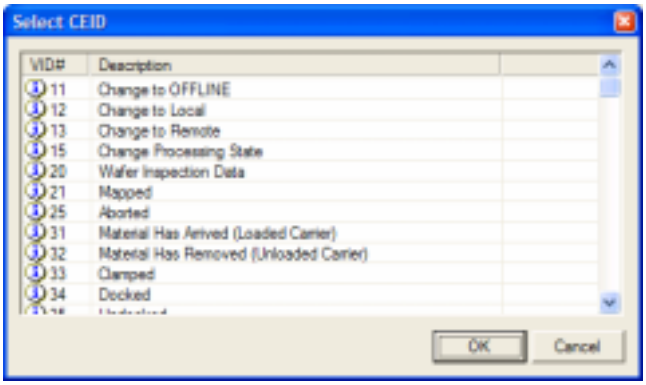
Many defined VID are displayed in the list, but only the following are actually used in the current version.

Item	Description
Establish Communications Timeout	EC timeout. This is the spacing for re-sending S1F13 Establish Communication Request(CR) when communication is not established.
Time Format	Time format. Either 12 or 16 bytes.
Control State	Control status
Process State	Processing status

■Predefined CEID Tab

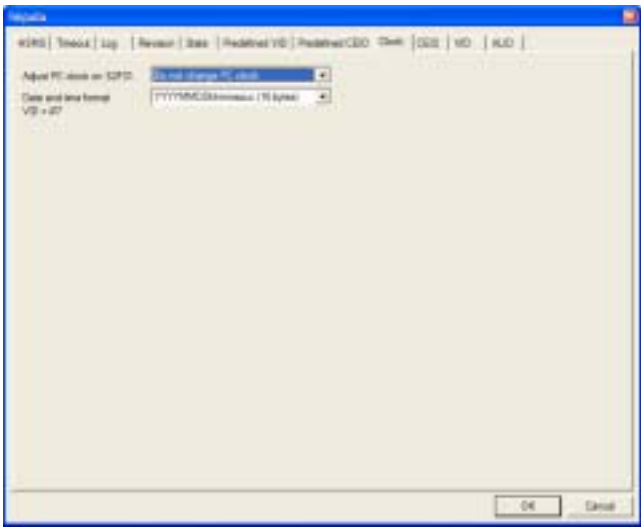


Item	Description
Predefined CEID Name	Defined CEID name
CEID#	CEID number
Description	Description
Edit...	"Edit" button. When a defined CEID is selected and this button is pressed, the following dialogue box will be displayed.



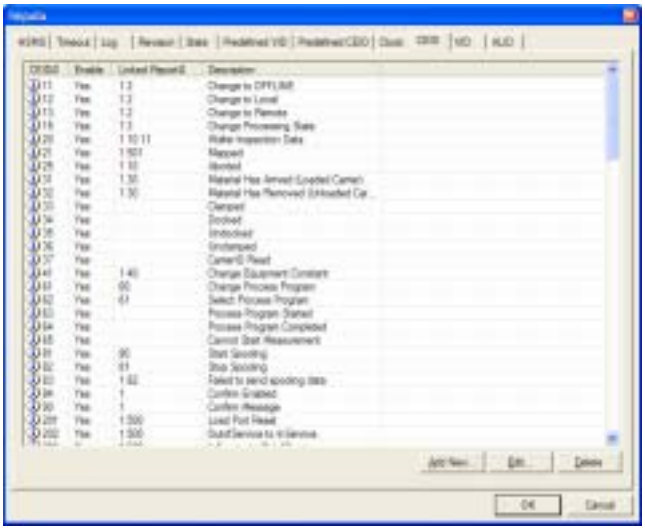
Here, defined CEID are displayed in a list in the CEID tab. Select a CEID with the same meaning as the defined CEID and press the OK button.

■Clock Tab



Item	Description
Adjust PC clock on S2F31	Selects whether or not to change the PC clock when S2F31 is received.
Date and time format	Date/time format. Prior to setting this item, it must be registered as a Predefined VID. If it is registered, a VID number will be displayed.

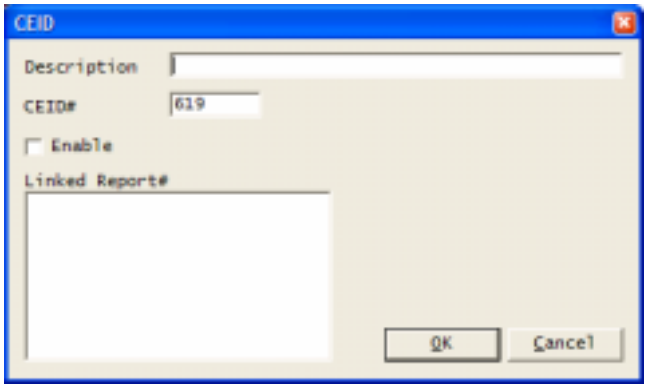
■CEID Tab



Item	Description
CEID#	CEID number
Enable	Yes=Enabled; No=Disabled
Linked Report#	Linked report number
Description	Description

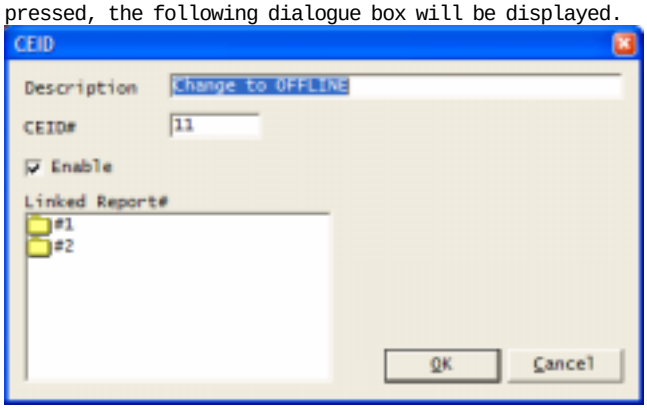
Add New...	Add new
Edit...	Edit selected CEID
Delete	Delete selected CEID

When the “Add New...” button is pressed, the following dialogue box will be displayed.



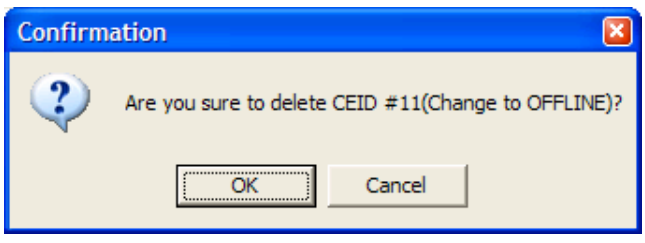
Item	Description
Description	CEID description
CEID#	CEID number. A number one greater than the largest registered CEID goes in as the default.
Enable	Checked = Enabled; Unchecked= Disabled.
Lined Report#	Linked report number. Report definition is set by communication.

When one CEID is selected and the “Edit...” button is pressed, the following dialogue box will be displayed.



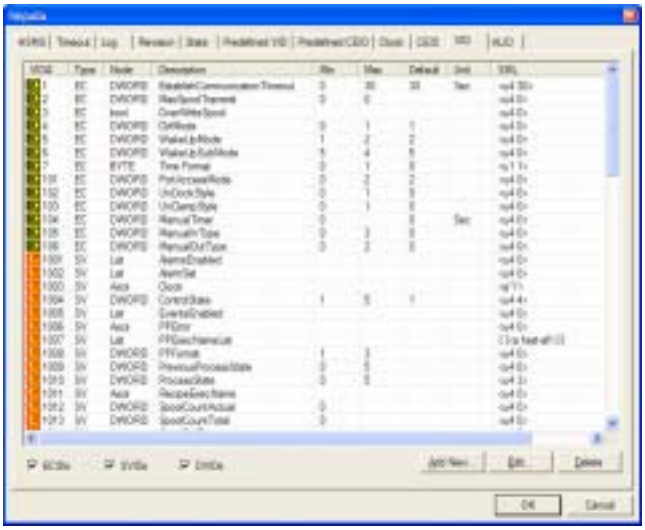
We can see that in this CEID, Report “#1” and “#2” are defined.

When one CEID is selected and the “Delete” button is pressed, the following message box will be displayed.



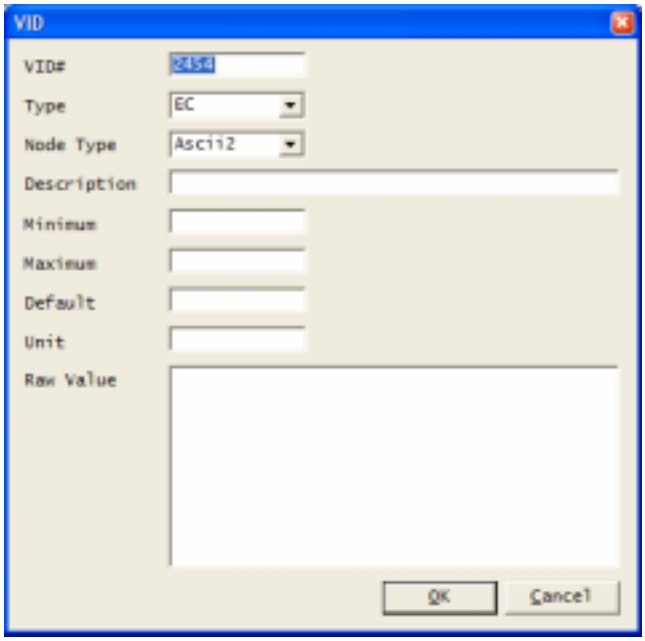
When the OK button is pressed, the selected CEID is deleted.

■VID Tab



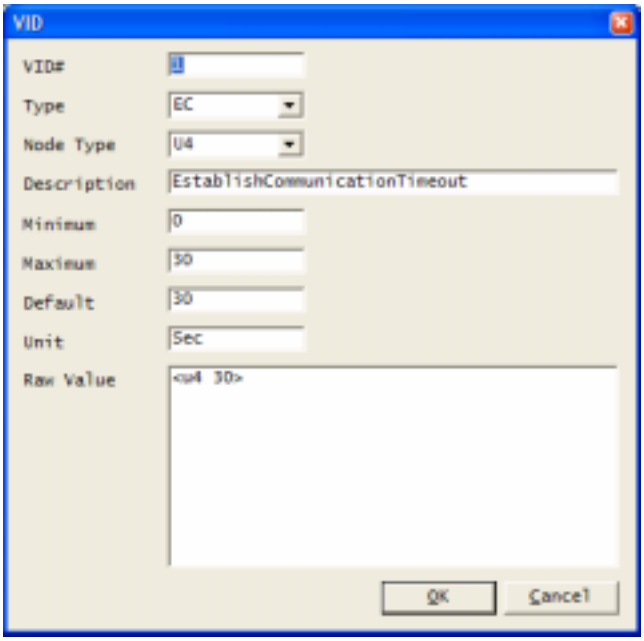
Item	Description
VID#	VID number
Type	Variable type. There are the following 3 types EC Equipment constant SV System variable DV Data variable
Node	Node form. Expressed in C++ style List List structure Binary Binary bool Bookean algebra Ascii ASCII char. string JIS 8 JIS8 char. string (half-size katakana) int64 Unsigned 8-byte integer char Unsigned 1-byte integer short Unsigned 2-byte integer long Unsigned 4-byte integer double 8-byte floating pt. no. float 4-byte floating pt. no. uint64 Signed 8-byte integer BYTE Signed 1-byte integer WORD Signed 2-byte integer DWORD Signed 4-byte integer MBCS 2-byte char. string
Description	Description
Min	Minimum value
Max	Maximum value
Default	Default value
Unit	Units
SML	Actual variable value SML expression
ECID	Checkmark indicates ECID shown in list. Unchecked indicates not shown.
SVID	Checkmark indicates SVID shown in list. Unchecked indicates not shown.
DVID	Checkmark indicates DVID shown in list. Unchecked indicates not shown.
Add New...	Add new
Edit...	Edit selected VID
Delete	Delete selected VID

When the “Add New...” button is pressed the following dialogue box will be displayed.

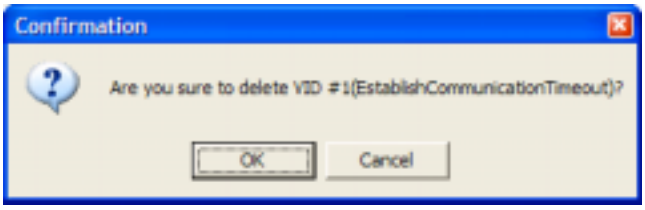


Item	Description
VID#	VID number. A number one greater than the largest registered VID number goes in as the default.
Type	Variable type
Node Type	Node format. Select from the following. List List structure Binary Binary Boolean Boolean algebra Ascii ASCII char. string JIS8 JIS8 char. string (half-size katakana) I8 Unsigned 8-byte integer I1 Unsigned 1-byte integer I2 Unsigned 2-byte integer I4 Unsigned 4-byte integer F8 8-byte floating pt. no. F4 4-byte floating pt. no. U8 Signed 8-byte integer U1 Signed 1-byte integer U2 Signed 2-byte integer U4 Signed 4-byte integer Ascii2 2-byte char. string
Description	Description
Minimum	Minimum value
Maximum	Maximum value
Default	Default value
Unit	Units
Raw Value	Actual variable number SML expression

When one VID is selected and the “Edit...” button is pressed, the following dialogue box will be displayed.



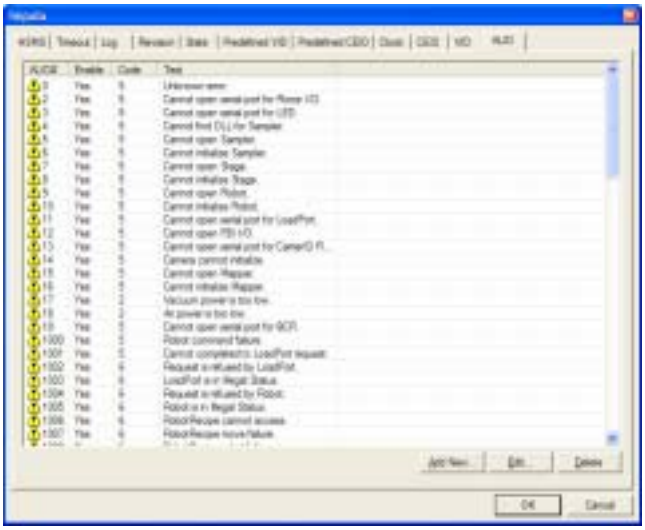
When one VID is selected and the “Delete” button is pressed, the following message box will be displayed.



When the OK button is pressed, the selected VID will be deleted.

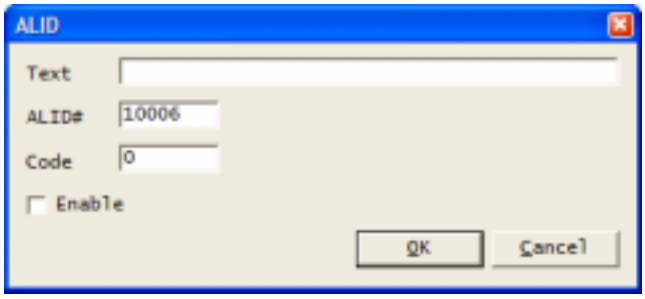
The data is actually registered only when the OK button in the setting dialogue box is pressed. For this reason, when using contents registered in the VID tab in a Predefined VID tab, you must press the OK button once to register, and then call the Configure method again.

■ALID Tab

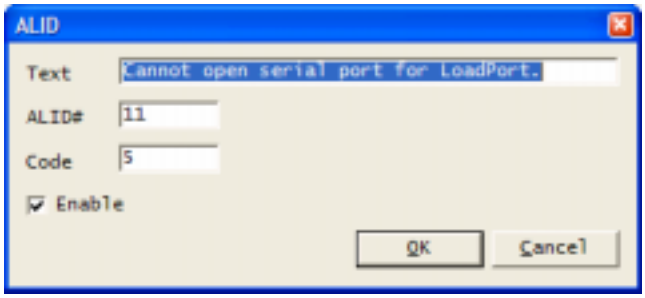


Item	Description
ALID#	ALID number
Enable	Yes= Enabled; No= Disabled
Code	Alarm code (ALCD)。
Text	Alarm text (ALTX)。
Add New...	Add new
Edit...	Edit selected ALID
Delete	Delete selected ALID

When the “Add New...” button is pressed, the following dialogue box will be displayed.



When one ALID is selected and the “Edit...” button is pressed, the following dialogue box will be displayed.



When one ALID is selected and the “Delete” button is pressed, the following message box will be displayed.

When the OK button is pressed, the selected ALID will be deleted.

■Declaration

■Visual C++ 6

```
BOOL Configure(LPCTSTR lpszCaption, long lOptionFlag);
```

■Visual Basic 6

```
Function Configure(lpszCaption As String, lOptionFlag As Long) As Boolean
```

Argument	Description
lpszCaption	Dialogue box caption title. If thisvalue is NULL or a character string with a length of 0, it will display as “Preferences”.
lOptionFlag	Option flag

At least one or more of the values below are specified as an option flag. The specified tab will be displayed. The numbers in the list below are expressed in hexadecimal format.

Value	Tab Displayed
0x0001	Model
0x0002	HSMS
0x0004	Timeout
0x0008	Revision
0x0010	State Model
0x0020	Clock
0x0040	CEID
0x0080	VID
0x0100	ALID
0x0200	Log File
0x0400	Predefined VID
0x0800	Predefined EID

Tabs are expected to increase in number in the future, as new functions are added. For this reason, set -1 to display all tabs.

■Return Value

Format	Description
BOOL	Returns True if a setting was changed and False if a setting was not changed.

7.2.2 DefProc

■Description

Causes processing of default when message received.

■Declaration

■Visual C++ 6

```
BOOL DefProc();
```

■Visual Basic 6

```
Function DefProc() As Boolean
```

■Return Value

Format	Description
BOOL	Returns true if processing was performed and false if not performed.

7.2.3 IndexToALID

■Description

Converts index to ALID

■Declaration

■Visual C++ 6

```
long IndexToALID(long lIndex);
```

■Visual Basic 6

```
Function IndexToALID(lIndex As Long) As Long
```

Argument	Description
lIndex	Index beginning from 0

■Return Value

Format	Description
long	Converted ALID. If index is out of range, a negative value will be returned.

7.2.4 IndexToCEID

■Description

Converts index to CEID

■Declaration

■Visual C++ 6

```
long IndexToCEID(long lIndex);
```

■Visual Basic 6

```
Function IndexToCEID(lIndex As Long) As Long
```

Argument	Description
lIndex	Index beginning from 0

■Return Value

Format	Description
long	Converted CEID. If index is out of range, a negative value will be returned.

7.2.5 IndexToVID

■Description

Converts index to VID

■Declaration

■Visual C++ 6

```
long IndexToVID(long lIndex);
```

■Visual Basic 6

```
Function IndexToVID(lIndex As Long) As Long
```

Argument	Description
lIndex	Index beginning from 0

■Return Value

Format	Description
long	Converted VID. If index is out of range, a negative value will be returned.

7.2.6 InvokeAlarm

■Description

An alarm is generated. Specifically, S5F1 Alarm Report Send(ARS) is sent to the Host. This is not sent if ALID is not registered or is disabled.

No matter what number is specified in sALCD, the first 7 bits will be disregarded. As ALCD is in binary format, it only has 8 bits. Therefore, only the 8th bit is actually used in ALCD. If this bit is 1, it means an alarm was generated; if 0, it means an alarm was cleared.

To send “alarm clear”, there must previously have been an “alarm generation”. When “alarm generation” is sent, within bop an “uncleared flag” will be set versus that ALID. If this “uncleared flag” has not been set, it is not possible to send “alarm clear”. When “alarm clear” is send, the “uncleared flag” is reset.

As the “uncleared flag” only records on/off as data, even if an “alarm generation” is sent twice in succession, one “alarm clear” sent will reset the “uncleared flag”.

The setting information for “uncleared flags” is recorded in a file so that it can be recovered even if an application has been closed.

■Declaration

■Visual C++ 6

```
BOOL InvokeAlarm(long lALID, short sALCD);
```

■Visual Basic 6

```
Function InvokeAlarm(lALID As Long, sALCD As Integer) As Boolean
```

Argument	Description
lALID	ALID. ALID must be registered beforehand.
sALCD	ALCD. Only the 8th bit is actually used.

■Return Value

Format	Description
BOOL	Returns true if an alarm was sent and false if not sent.

7.2.7 InvokeEvent

■Description

Generates an event. Specifically, this sends S6F11 Event Report Send(ERS) to the Host.。

If a report is linked to an event, the report is also automatically generated.

■Declaration

■Visual C++ 6

```
BOOL InvokeEvent(long lCEID);
```

■Visual Basic 6

```
Function InvokeEvent(lCEID As Long) As Boolean
```

Format	Description
lCEID	CEID。

■Return Value

Format	Description
BOOL	Returns true if an event was sent and false if not sent.

7.2.8 IsValidVID

■Description

Verifies whether VID is correct.

■Declaration

■Visual C++ 6

```
BOOL IsValidVID(long lVID);
```

■Visual Basic 6

```
Function IsValidVID(lVID As Long) As Boolean
```

Format	Description
lVID	VID。

■Return Value

Format	Description
BOOL	Returns true if a VID was registered and false if not registered.

7.2.9 Load

■Description

Loads saved .bop files.

■Declaration

■Visual C++ 6

```
BOOL Load();
```

■Visual Basic 6

```
Function Load() As Boolean
```

Format	Description
BOOL	Returns true if loading was successful and false if unsuccessful.

7.2.10 LoadIniFile

■Description

Loads saved .ini files.

■Declaration

■Visual C++ 6

```
BOOL LoadIniFile();
```

■Visual Basic 6

```
Function LoadIniFile() As Boolean
```

Format	Description
BOOL	Returns true if loading was successful and false if not successful.

7.2.11 RegisterALID

■Description

Newly registers ALID. As this impacts settings in S5F3 Enable/Disable Alarm Send(EAS), in some cases the use of RegisterALID is not recommended. The basic sequence is to add Configure beforehand, and read using Load.

■Declaration

■Visual C++ 6

```
BOOL RegisterALID(long lALID, short sALCD, LPCTSTR lpszALTX);
```

■Visual Basic 6

```
Function RegisterALID(lALID As Long, sALCD As Integer, lpszALTX As String) As Boolean
```

Format	Description
lALID	ALID
sALCD	ALCD
lpszALTX	ALTX

■Return Value

Format	Description
BOOL	Returns true if registration was successful and false if unsuccessful.

7.2.12 RegisterVID

■Description

Newly registers VID.

■Declaration

■Visual C++ 6

```
BOOL RegisterVID(long lVID, short sType, short sNodeType, LPCTSTR lpszMin, LPCTSTR lpszMax, LPCTSTR lpszDefault, LPCTSTR lpszUnit, LPCTSTR lpszDescription);
```

■Visual Basic 6

```
Function RegisterVID(lVID As Long, sType As Integer, sNodeType As Integer, lpszMin As String, lpszMax As String, lpszDefault As String, lpszUnit As String, lpszDescription As String) As Boolean
```

Format	Description
lVID	VID。
sType	Type
sNodeType	SECS-II node format
lpszMin	ECMIN
lpszMax	ECMAX
lpszDefault	ECDEF
lpszUnit	UNITS
lpszDescription	ECNAME

sType is one of the following.

Type	Description

1	ECID。
2	SVID。
4	DVID。

sNodeType is one of the following.

Node Format	Value	Description
NodeTypeList	1	List
NodeTypeBinary	2	Binary
NodeTypeBoolean	3	Boolean
NodeTypeAscii	4	ASCII char. string
NodeTypeJis	5	JIS-8 code
NodeTypeLong8	6	8-byte signed integer
NodeTypeChar	7	1-byte signed integer
NodeTypeShort	8	2-byte signed integer
NodeTypeLong	9	4-byte signed integer
NodeTypeDouble	10	8-byte floating pt. no.
NodeTypeFloat	11	4-byte floating pt. no.
NodeTypeDWord8	12	8-byte unsigned integer
NodeTypeByte	13	1-byte unsigned integer
NodeTypeWord	14	2-byte unsigned integer
NodeTypeDWord	15	4-byte unsigned integer
NodeTypeAscii2	16	2-byte ASCII char. string

■Return Value

Format	Description
BOOL	Returns true if registration was successful and false if unsuccessful.

■Related Items
VIDType, VIDNodeType, VIDMin, VIDMax, VIDDefault, VIDUnit, VIDDescription

7.2.13 Save

■Description
Saves setting to .bop files.

■Declaration
■Visual C++ 6

```
BOOL Save();
```

■Visual Basic 6

```
Function Save() As Boolean
```

Format	Description
BOOL	Returns true if saving was successful and false if saving was unsuccessful.

7.2.14 Send

■Description
Sends messages selected in WorkSpace and Reply.

■Declaration
■Visual C++ 6

```
BOOL Send();
```

■Visual Basic 6

```
Function Send() As Boolean
```

Format	Description
BOOL	Returns true if sent successfully and false if sending was unsuccessful.

■Related Items
WorkSpace, Reply

7.2.15 UnregisterALID

■Description
Deletes ALID. Deleting will fail if ALID is not registered.

■Declaration
■Visual C++ 6

```
BOOL UnregisterALID(long lALID);
```

■Visual Basic 6

```
Function UnregisterALID(lALID As Long) As Boolean
```

Format	Description
lALID	ALID

■Return Value

Format	Description
BOOL	Returns true if deleted successfully and falseif not deleted successfully.

7.2.16 UnregisterVID

■Description
Deletes VID. Deleting will fail if VID is not registered.

■Declaration
■Visual C++ 6

```
BOOL UnregisterVID(long lVID);
```

■Visual Basic 6

```
Function UnregisterVID(lVID As Long) As Boolean
```

Format	Description
lVID	VID

■Return Value

Format	Description
BOOL	Returns true if deleted successfully and false if not deleted successfully.

7.2.17 WriteToLogFile

■Description
Writes to log files.

■Declaration
■Visual C++ 6

```
void WriteToLogFile(LPCTSTR lpszText);
```

■Visual Basic 6

```
Sub WriteToLogFile(lpszText As String)
```

Format	Description
lpszText	Writing character string

7.3 Events

7.3.1 CommunicationStateChanged

■Description

Communication status changed. Communication status is one of the following.

Name	Value	Description
Disabled	0	Communication disabled
NotCommunicating	1	Communication aborted
Communicating	2	Communication in progress

■Declaration

■Visual C++ 6

```
void FireCommunicationStateChanged(short sNewState, short sPrevState);
```

■Visual Basic 6

```
Event CommunicationStateChanged(sNewState As Integer, sPrevState As Integer)
```

Format	Description
sNewState	New status
sPrevState	Prior status

7.3.2 Connected

■Description

Connected with partner via communication.

■Declaration

■Visual C++ 6

```
void FireConnected(LPCTSTR lpszIPAddress, long lPortNumber);
```

■Visual Basic 6

```
Event Connected(lpszIPAddress As String, lPortNumber As Long)
```

Format	Description
lpszIPAddress	IP address of communication partner
lPortNumber	TCP port no. of communication partner

7.3.3 ConnectionStateChanged

■Description

Communication status changed.

Name	Value	Explanation
NotConnected	0	Not connected
NotSelected	1	Connected (unselected)
Selected	2	Connected (selected)

■Declaration

■Visual C++ 6

```
void FireConnectionStateChanged(short sNewState, short sPrevState);
```

■Visual Basic 6

```
Event ConnectionStateChanged(sNewState As Integer, sPrevState As Integer)
```

Format	Description
sNewState	New status
sPrevState	Prior status

7.3.4 ControlStateChanged

■Description

Control status changed. Control status is one of the following.

Name	Value	Description
EquipmentOffLine	0	Equip. offline
AttemptOnLine	1	Attempt online
HostOffLine	2	Host offline
OnLineLocal	3	Online local
OnLineRemote	4	Online remote

■Declaration

■Visual C++ 6

```
void FireControlStateChanged(short sNewState, short sPrevState);
```

■Visual Basic 6

```
Event ControlStateChanged(sNewState As Integer, sPrevState As Integer)
```

Format	Description
sNewState	New status
sPrevState	Prior status

7.3.5 Disconnected

■Description

Disconnected from communication with partner

■Declaration

■Visual C++ 6

```
void FireDisconnected(LPCTSTR lpszIPAddress, long lPortNumber);
```

■Visual Basic 6

```
Event Disconnected(lpszIPAddress As String, lPortNumber As Long)
```

Format	Description
lpszIPAddress	IP address of communication partner
lPortNumber	TCP port no. of communication partner

7.3.6 Errors

■Description

Error occurred.

■Declaration

■Visual C++ 6

```
void FireErrors(short sErrorCode, LPCTSTR lpszErrorText);
```

■Visual Basic 6

```
Event Errors(sErrorCode As Integer, lpszErrorText As String)
```

Format	Description
sErrorCode	Error code
lpszErrorText	Error character string

7.3.7 Received

■Description

A message was received. Normally, DefProc is called and message processing is caused in bop. To process a message without bop, it is also possible to trigger unique processing.

The received message is set in WorkSpace0 Reply=false. To analyze a received message, they may be processed as-is. To reply to a received message, set Reply to true and edit the reply portion. Following this, when DefProc is called from the event handler function, if the message requires a reply a reply message written in the reply section will be sent.

■Declaration

■Visual C++ 6

```
void FireReceived(LPCTSTR lpszIPAddress, long lPortNumber);
```

■Visual Basic 6

```
Event Received(lpszIPAddress As String, lPortNumber As Long)
```

Format	Description
lpszIPAddress	IP address of communication partner
lPortNumber	TCP port no. of communication partner

■Related Items

WorkSpace, Reply, Send, DefProc

7.3.8 Sent

■Description

A message was sent

■Declaration

■Visual C++ 6

```
void FireSent();
```

■Visual Basic 6

```
Event Sent()
```

■Related Items

WorkSpace, Reply, Send

7.3.9 VIDChanged

■Description

A message was sent.

■Declaration

■Visual C++ 6

```
void FireVIDChanged(long lVID);
```

■Visual Basic 6

```
Event VIDChanged(lVID As Long)
```

Format	Description
lVID	VID

8 SML Reference

8.1 General Points of Note

8.1.1 White Space

White spaces (spaces, tabs, line breaks, restore codes) only serve the purpose of spacing characters. For this reason, it is possible to insert appropriate tabs and line breaks in order to enhance legibility. Please note that they will be treated as characters when located within comments or character strings.

8.1.2 Comments

Comments consist of everything following an asterisk (*) until the end of that line. However, this does not apply to asterisks forming part of a character string.

8.1.3 Numbers

Numbers consist of the characters from 0~9, and the minus sign (-). To express in hexadecimal format, add '0x' to the beginning. In this case, it is also possible to use characters from a~f and A~F. Decimals may also be written in Western style, skipping the "0" at the beginning of the number ("0.9" -> ".9"). Index numbers expressions may also be used. It is also possible to use true (=1) and false (=0) as reserved words.

8.1.4 Character String Expressions

Character strings are located in ranges bounded by single quotation marks (‘). It is not possible to include line break codes and single quotation marks themselves inside a character string. For this reason, if these characters must be included for some reason, use a hexadecimal expression such as "0x0a".

8.2 SML Grammar

Bolded sections in descriptive passages indicate that those characters are to be written. Basically, these characters may be either upper-case or lower-case characters. Italic characters reference individual descriptions. Sections enclosed in brackets ([]) can be omitted.

8.2.1 Syntax

```
[sxxfyy[w]] Body
```

Element	Description
xx	Stream number. Do not add a space between the characters "s" and "f".
yy	Function number. Do not add a space betweenthecharacters "f" and "w".
w	Wait bit. Written as "w" when specifying. May be omitted.
Body	Message body.

Since the stream, function and wait bit are recognized as a single combined unit, do not add spaces or line break code between these. It is also possible to omit stream and function and write only the message body.

8.3 Message Body

The message body has a hierarchical structure.

SML Expression	Description
l	List
b	Binary
bool	Boolean
a	ASCII char. string
j	JIS-8
a2	2-byte ASCII char. string
i8	8-byte signed integer
i1	1-byte signed integer
i2	2-byte signed integer
i4	4-byte signed integer
f8	8-byte floating pt. no.
f4	4-byte floating pt. no.
u8	8-byte unsigned integer
u1	1-byte unsigned integer
u2	2-byte unsigned integer
u4	4-byte unsigned integer

8.3.1 List

```
{[l [Number]]Body}
<[l [Number]]Body>
```

Element	Description
Number	List number. Provided only for compatibility with SECSIM. This number is disregarded.
Body	Message body. Other items can be lined up.

8.3.2 Binary

```
<b [Numbers]>
```

Element	Description
Numbers	Number. For example, written as follows: `<b 0xff 0x3e 255 0>`

8.3.3 Boolean

```
<bool [Numbers]>
<boolean [Numbers]>
```

Element	Description
Numbers	Number. For example, written as follows: `<bool true false 1 0>`

8.3.4 ASCII Character Strings

```
<a [Strings]>
```

Element	Description

Strings	Character strings. It is possible to break apart long character strings when writing. It is also possible to write in direct character code. For example, an SML such as that below:
	`<a ‘ABC’ ‘DEF’ ‘012’ 0x33 ‘4’ 53 54 ‘789’>`
	would be the same as the following:
	`<a ‘ABCDEF0123456789’>`

8.3.5 2-byte Character Strings

```
<a2 [Strings]>
```

Element	Description
Strings	2-byte character string. The current version only accommodates MBCS.

8.3.6 JIS-8 Character Strings

```
<j [Strings]>
```

Handled in the same fashion as ASCII format.[3]

Element	Description

Strings	Character string. It is possible to break up long character strings when writing. It is also possible to write direct character code. For example, an SML such as the one below:
	
	`<a 'ABC' 'DEF' '012' 0x33 '4' 53 54 '789'>`
	
	Would be the same as the following:
	
	`<a 'ABCDEF0123456789'>`

8.3.7 Integers

```
<i1[Numbers]>
<i2[Numbers]>
<i4[Numbers]>
<i8[Numbers]>
<u1[Numbers]>
<u2[Numbers]>
<u4[Numbers]>
<u8[Numbers]>
```

Elements	Description

Numbers	Numbers have the following meanings.
	
	`i1 Signed 8-bit integer`
	`i2 Signed 16-bit integer`
	`i4 Signed 32-bit integer`
	`i8 Signed 64-bit integer`
	`u1 Unsigned 8-bit integer`
	`u2 Unsigned 16-bit integer`
	`u4 Unsigned 32-bit integer`
	`u8 Unsigned 64-bit integer`
	
	It is possible to line up several numbers together when writing. In this case, they will become an array. For example, they could be written as follows:
	
	`<i1 1 0x02 3>`
	
	In the current version it is not possible to enter very large values such as i8 or u8.

8.3.8 Floating Point Numbers

```
<f4[FNumbers]>
<f8[FNumbers]>
```

Element	Description

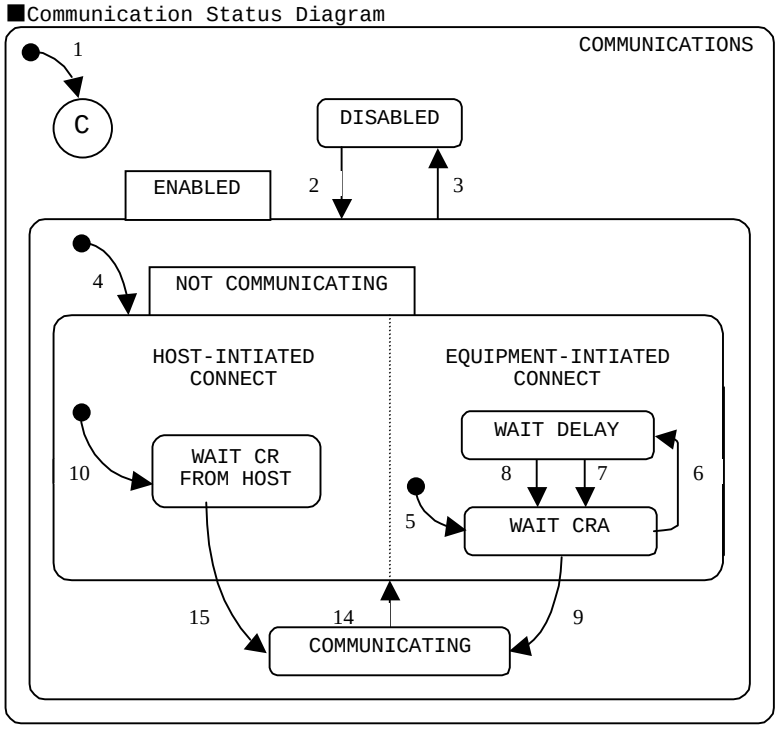
Fnumbers	Floating point number. Meanings are as follows:
	
	`f4 32-bit floating point number`
	`f8 64-bit floating point number`
	
	For example, these can be written as follows:
	
	`<f4 0 1.0 3.14>`

[3] I have never personally seen a message using JIS-8.

9 GEM

9.1 Communication Status Model

■Communication Status Diagram

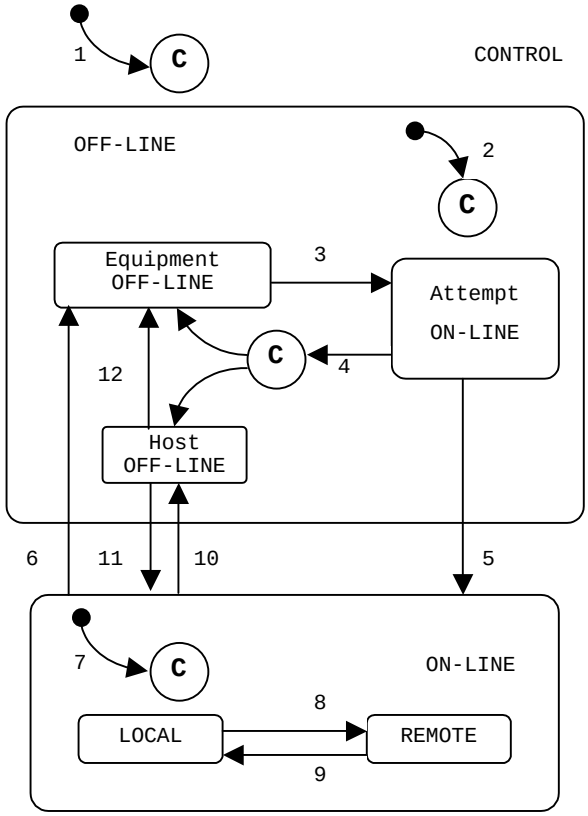


■Communication Status Transitions

#	Current Status	Trigger	New Status	Operation	Comments
1	(Comm. Status)	System intialization	System Default	None	System default is setto either comm. disabled or comm. enabled.
2	DISABLED (Comm. Disabled)	Operator switches from comm. disabled to comm. enabled.	ENABLED (Communication enabled)	None	SECS-Ⅱ communication is enabled.
3	ENABLED (Comm. Enabled)	Operator switches from comm. enabled to comm. disabled.	DISABLED (Communication disabled)	None	SECS-Ⅱ communication is is prohibited
4	(Enable Comm.)	Enter communication enabled status	NOT COMMUNICATING (Communication aborted)	None	Can go into comm. enabled from system initialization, or operator can switch over to comm. enabled.
5	(Enter tool-initiated connection)	(Enter communication aborted status)	WAIT CRA (Awaiting communication request acknowledgment)	Com. initialization Set CommDelay timer to timeout. Send S1F13.	Start communication establishment.
6	WAIT CRA (Awaiting communication request Acknowledge)	Communication transaction failure	WAIT DELAY (Awaiting delay timer timeout)	Initialize CommDelay timer. Take all messages queued for sending out of queue.	When appropriate, putmessages taken from queue into spool buffer in orderor creation. Await timer timing out.
7	WAIT DELAY (Awaiting delay timer timeout)	Communication delay timer timed out	WAIT CRA (Awaiting comm. request Acknowledge)	Send S1F13.	Await S1F14. May also receive S1F13 from host.
8	WAIT DELAY (Awaiting delay timer timeout)	Receive message other than S1F13	WAIT CRA (Awaiting communication request Acknowledge)	Discard message. No response. Set comm. delay timer to “timeout”. Send S1F13.	This means there is a chance of establishing communication.
9	WAIT CRA (Awaiting comm. req. acknowledgment)	Receive COMMACK=0 S1F14 which was awaited	COMMUNICATING (Executing communication)	None	Communication is established.
10	(Enter host-initiated connection)	(Enter communication aborted status)	WAIT CR FROM HOST (Awaiting comm. from from Host)	None	Awaiting S1F13 from host.
11	COMMUNICATING (Execute communication)	Loss of communication(Refer to SEMI E4(SECS-Ⅰ)or SEMI E37 (HSMS) regarding definition of comm. Failure protocol)	NOT COMMUNICATING (Communication aborted)	Take out all messages queued for sending from queue.	As needed, messages taken out of queue are put into spool buffer.
12	WAIT CR FROM HOST (Awaiting comm. establishment from host)	Receive S1F13	COMMUNICATING (Executing communication)	Send S1F14 by COMMACK=0.	Communication is established.

9.2 Control Status Model

■Control Status Diagram



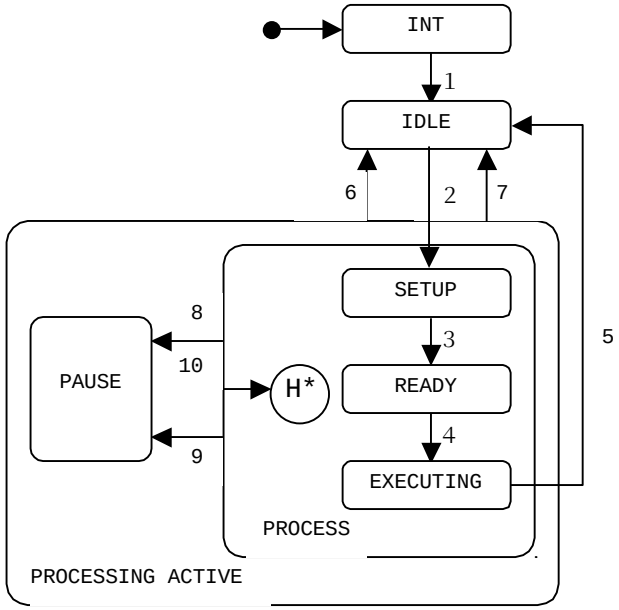
■Control Status Transitions

#	Current Status	Trigger	New Status	Action	Comments
1	(Undefined)	Enter control status (system start-up)	CONTROL Control status (subordinate status depends on settings)	None	Depending on default setting, equipment goes online or offline.[4]
2	(Undefined)	Enter offline status	OFF-LINE Offline status (subordinate status depends on settings)	None	Depending on default setting, equipment may go into any subordinate status of offline.
3	EQUIPMENT OFF-LINE	Operator switches to online	ATTEMPT ON-LINE	None	When in online established status, note that S1F1 can be sent at any time.
4	ATEMP ON-LINE	S1F0	New status differs depending on set conditions	None	Due to loss of communication, reply timeout, or receipt of SIF0.[5] Depending on settings, transitions to equipment offline or host offline.
5	ATTEMPT ON-LINE	Equip. receives awaited S1F2 from host.	ON-LINE	None	Notifies that host will transition to online in Transition 7.
6	ON-LINE	Operator switches to offline	EQUIPMENT OFF-LINE	None	“Equipment offline” event generated.[6] If offline, event reply msg.discarded.
7	(Undefined)	Enter online status	ON-LINE Online status (subordinate status depends on setting of remote/local switch)	None	“Control status local” or”control status remote” event generated. Event report shown in subordinate status of online actual transition.
8	LOCAL	Operator sets front-panel switch to remote	REMOTE	None	“Control status remote” event generated.
9	REMOTE	Operator sets front-panelswitch to local.	LOCAL	None	“Control status local” event generated.
10	ON-LINE	Equip. receives “offline switch” (S1F15) msg.from host (S1F15)	HOST OFF-LINE	None	“Equipment offline” event generated.
11	HOST OFF-LINE	Equipment acknowledges “online transition request” (S1F17)	ON-LINE	None	Notifies that host will transition to online in Transition 7.
12	HOST OFF-LINE	Operator switches to offline	EQUIPMENT OFF-LINE	None	“Equipment offline” event generated.

[4]Condition settings mentioned in Transition 1 and 2 must be single settings. User can thereby select whether to go into equipment offline, attempt online, host offline, or online.
[5]Loss of communication is specified in the protocol. Please refer to the corresponding protocol standard for the definition of loss of communication protocol specifications (Example:SEMI E4 or E37)
[6]All host-initiated transactions versus the equipment must be completed. To accomplish this, either send the appropriate reply message to the host prior to the sending of the event message, or send SxF0 after the event message (in other words, after the transaction is over).

9.3 Processing Status Model

■Processing Status Diagram



■Processng Status Transitions

#	Current Status	Trigger	New Status	Action	Comment
1	INT	Equipment initialization complete	IDLE	None	None
2	IDLE	Set-up command given	SETUP	None	None
3	SETUP	Set-up actions all complete, equipment ready to receive start command	READY	Action depends on equipment	None
4	READY	Equipment received start command (START) from host or operator console	EXECUTING	Action depends on equipment	None
5	EXECUTING	Processing work done	IDLE	None	None
6	PROCESSING ACTIVE	Equipment received stop command (STOP) from host or operator console	IDLE	None	None
7	PROCESSING ACTIVE	Equipment received abort command (ABORT) from host or operator console	IDLE	Action depends on equipment	None
8	PROCESS	Decided to pause equipment due to alarm conditions, etc.	PAUSE	Action depends on equipment	This type of abnormality typically requires an operator assist.
9	PROCESS	Equipment received pause command (PAUSE) from host or operator console	PAUSE	Action depends on equipment	None
10	PAUSE	Equipment received resume command (RESUME) from host or operator console	Previous PROCESS Sub-state	Action depends on equipment	None

9.4 Establishing Communication

9.4.1 Establishing Communication from Host

Comment	Host	Equipment	Comment
	Comm. status is comm. enabled (ENABLED) (subordinate status may be anything)		
Establish comm. request	S1F13→		
		←S1F14	Responds with COMMACK=Accepted Comm. status=Executing communication (COMMUNICATING)

9.4.2 Establishing Communication from Equipment, Host Acknowledge Reply

Comment	Host	Equipment	Comment
	Comm. status =Comm. aborted (NOT COMMUNICATING)		

			[LOOP]
			[LOOP]……SEND
		←S1F13	Establish communication request
Estalish communication request Acknowledge	S1F14→		
			[IF]S1F14 is received without time-out
			[THEN]Out of loop……SEND
			[ELSE]Delay by time set in Establish Communications-Timeout
			[ENDIF]
			[END_LOOP]……SEND
			[IF]COMMACK=Accept
			[THEN]Comm. status=Executing communication (COMMUNICATING) Out of loop……
			[ELSE]Reset timer for delay, delay by time set in Establish Communications-Timeout
			[ENDIF]
			[END-LOOP]

9.5 GEM Compliance

It is not the case that bop provides all GEM functions. For example, for “Material Transfer” it is necessary to issue S6F11 from the application side, and there is nothing corresponding to this in bop. Although other manufacturers may put on airs stating in their GEM development support environments that they are “compliant”, this can only be called overstated advertisement, based on a distortion of the truth.

This product has been designed so that even those items marked as not providing the stated performance may be added by the user as required. This is why we can mark “GEM compliant” in every item. Please note that the items “variable data collection”, “trace data collection”, “limit monitoring” and “spooling” are in the majority of cases probably not requirements.

GEM Compliance				
GEM Basic Conditions	Provided?		GEM Compliance	
Status Model	☑Yes	☐No	☑Yes[7]	☐No
Equipment Process Status	☑Yes	☐No		
Host-Initiated S1F13/F14 Scenario	☑Yes	☐No		
Event Notification	☑Yes	☐No		
Online Checking	☑Yes	☐No		
Error Messaging	☑Yes	☐No		
Controls (Operator-Activated)	☑Yes	☐No		
Documentation	☑Yes	☐No		
Additional Performance Items	Provided?		GEM Compliance[8]	
Establishing Communication	☑Yes	☐No	☑Yes	☐No
Dynamic Event Report Setting Changes	☑Yes	☐No	☑Yes	☐No
Variable Data Collection	☐Yes	☑No	☐Yes	☐No
Trace Data Collection	☐Yes	☑No	☐Yes	☐No
Status Data Collection	☑Yes	☐No	☑Yes	☐No
Alarm Management	☑Yes	☐No	☑Yes	☐No
Remote Control	☑Yes	☐No	☑Yes	☐No
Equipment Constant	☑Yes	☐No	☑Yes	☐No
Process Program Management	☐Yes	☑No	☐Yes	☐No
Material Transfer	☐Yes	☑No	☐Yes	☐No
Equipment Terminal Service	☐Yes	☑No	☐Yes	☐No
Clock	☑Yes	☐No	☑Yes	☐No
Limit Monitoring	☐Yes	☑No	☐Yes	☐No
Spooling	☐Yes	☑No	☐Yes	☐No
Controls (Host-Activated)	☑Yes	☐No	☑Yes	☐No

“Process Program Management” has not been provided because the process program (recipe) structure varies largely from equipment type to equipment type, but it is not very difficult to add. Only events need to be added for “Material Transfer”. “Equipment Terminal Service” only involves displaying somewhere in the application, so this could be added very easily. If these functionalities are added, their corresponding items above could be marked “Yes” for Provided and “Yes” for GEM Compliance.

The materials necessary for the “Documentation” item have already been stated in this Users Manual, so with appropriate touching-up and corrections, it could be used to satisfy this item. As the SEMI-issued E.5 (SECS-II) Manual is extremely difficult to read, we have added descriptions of each message to the item dictionary in this Users Manual.

[7] It is only possible to mark “Yes” here when all basic GEM conditions have been achieved in accordance with GEM.
[8] If basic GEM conditions are not GEM-compliant, added performances cannot belisted as GEM-compliant, or in other words, “YES” cannot be marked for some items.

10 SECS-II Messages

In the following we have extracted those messages used by bop from the SEMI E.5 (SECS-II) Manual, and rewritten them for ease of readability.

In bop, there are some items with restrictions on item format. For example, for CEID, in SEMI E.5, although it states that either "binary", "signed integers" and "unsigned integers" are acceptable, in actuality the use of u4 format is common. In binary, only numbers from 0~255 can be accommodated, and with signed integers there may also be negative values, so let us say that the SEMI standards are slightly odd. For this reason, in bop we use a fixed u4 format.

10.1 Item Dictionary

Item descriptions are given where each message is discussed, so in the following we may only need to make notes of a few aspects.

10.1.1 ACKC5

■Description

Acknowledgement code. 1 byte.

Value	Description
0	Acknowledged
>0	Error, cannot acknowledge
1~63	Reserved

■Format

b[1]

■Related Messages

S5F2 Alarm Report Acknowledge(ARA)
S5F4 Enable/Disable Alarm Acknowledge (EAA)

10.1.2 ACKC6

■Description

Acknowledgement code. 1 byte.

Value	Description
0	Acknowledgement
>0	Error, cannot acknowledge
1~63	Reserved

■Format

b[1]

■Related Messages

S6F12 Event Report Acknowledge(ERA)

10.1.3 ACKC7

■Description

Verification code. 1 byte.

Value	Description
0	Authorized
1	Not Authorized
2	Length Error
3	Array Overflow
4	*PPID* Undefined
5	Mode Error
>5	Other Error
6~63	Reserved

■Format

b[1]

■Related Messages

S7F4 Process Program Acknowledge(PPA)
S7F18 Delete Process Program Acknowledge(DPA)
S7F24 Formatted Process Prog. Acknowledge(FPA)

10.1.4 ACKC7A

■Description

Confirmation code. 1 byte.

Value	Description
0	Acknowledged
1	MDLN does not match
2	SOFTREV does not match
3	Invalid CCODE。
4	Invalid *PPARM* value
5	Other error (displayed via *ERRW7*)
6~63	Reserved

■Format

i1[1], u1[1]

■Related Messages

S7F27 Process Program Verification Send(PVS)

10.1.5 ACKC10

■Description

Confirmation code. I byte.

Value	Description
0	Display acknowledged
1	Message not displayed
2	Cannot use terminal
3~63	Reserved

■Format

b[1]

■Related Message

S10F2 Terminal Request Acknowledge(TRA)
S10F4 Terminal Display, Single Block Acknowledge (VTA)
S10F6 Terminal Request, Multi Block Acknowledge (VMA)

10.1.6 AGENT

■Description

■Format

a

■Related Messages

S15F21 Recipe Action Request
S15F22 Recipe Action Acknowledge

10.1.7 ALCD

■Description

Alarm Code

Bit	Value	Explanation
bit8	1	Alarm status generated
	0	Alarm status cleared
bit7~1	Alarm Classification Code	
	0	Not yet used
	1	Human safety-related
	2	Tool safety-related
	3	Parameter control alarm
	4	Parameter control error
	5	Unrecoverable error
	6	Equipment status report
	7	Warning flag
	8	Cannot verify data
	>8	Other category
	9~63	Reserved

■Format

b

■Related Messages
S5F1 Alarm Report Send(ARS)
S5F6 List Alarm Data (LAD)

10.1.8 ALED

■Description
Alarm Enable/Disable Code. 1 byte.

Bit	Value	Description
bit8	1	Alarm enable
	0	Alarm disable

■Format
b[1]

■Related Messages
S5F3 Enable/Disable Alarm Send(EAS)

10.1.9 ALID

■Description
Alarm ID

■Format
u4

■Related Messages
S5F1 Alarm Report Send(ARS)
S5F3 Enable/Disable Alarm Send(EAS)
S5F5 List Alarm Request(LAR)
S5F6 List Alarm Data(LAD)

10.1.10 ALTX

■Description
Alarm text. Max 40 characters.

■Format
a

■Related Messages
S5F1 Alarm Report Send(ARS)
S5F6 List Alarm Data(LAD)

10.1.11 ATTRDATA

■Description
Holds special attribute values of particular objects

■Format
l, b, bool, a, i*, f*, u*

■Related Messages
S14F1 Get Attribute Request(GAR)
S14F2 Get Attribute Data(GAD)

10.1.12 ATTRID

■Description
Attribute identifier for objects of particular types

■Format
a, u*

■Related Messages
S14F1 Get Attribute Request(GAR)
S14F2 Get Attribute Data(GAD)

10.1.13 ATTRRELN

■Description
Regulation of relationships between specific limit values and object instance attribute values (interest numbers).

Value	Description
0	Limit value is equivalent to interest value
1	Limit value not equivalent to interest value
2	Limit value less than interest value
3	Limit value less than interest value, but is equivalent
4	Limit value is more than interest value
5	Limit value is more than interest value, but is equivalent
6	Interest value is included in limit value (part of the set)
7	Interest value is not included in limit value(not part of the set)
>7	Reserved

■Format
u1

■Related Messages
S14F1 Get Attribute Request(GAR)

10.1.14 CCODE

■Description
Command code. Each command code corresponds to an individual process operation that can be performed by the device.

■Format
i2, u2

■Related Messages
S7F23 Formatted Process Program Send(EPS)
S7F26 Formatted Program Data(FPD)
S7F27 Process Program Verification Send(PVS)

10.1.15 CEED

■Description
Collected event or trace enable/disable code. 1 byte.

Value	Description
False	Disable
True	Enable

■Format
bool[1]

■Related Messages
S2F37 Enable/Disable Event Report(EDER)

10.1.16 CEID

■Description
Collected event ID

■Format
u4

■Related Messages
S2F33 Defined Report(DR)
S2F35 Link Event Report(LER)
S2F36 Link Event Report Acknowledge(LERA)
S2F37 Enable/Disable Event Report(EDER)
S2F38 Enable/Disable Event Report Acknowledge (EERA)
S6F11 Event Report Send(ERS)
S6F15 Event Report Request(ERR)
S6F16 Event Report Data(ERD)

10.1.17 CEPACK

■Description
Command extended parameter grant. If a particular value of *CPNAME* is defined as having a list *CEPVAL*, *CEPACK* will have the same structure as the corresponding *CEPVAL* list format so that it is used by S2F49[9] (Enhanced Remote command). Other than this case, *CEPACK* will be a 1-byte integer.

Value	Description
0	No error
1	Parameter name (*CPNAME*) does not exist
2	Incorrect value is specified in *CEPVAL*
3	Incorrect format is specified in *CEPVAL*
4	Parameter name (*CPNAME*) invalid use
5~63	Reserved

■Format
l, u1[1]

■Related Messages
S2F50 Expansion Remote Command Acknowledge

10.1.18 CEPVAL

■Description
Command extended parameter value.*CPNAME* value is used, so when there is a certain way in which *CEPVAL* is used, it will always be known. *CEPVAL* has the following formats:Single (not list) value (Ex:*CPVAL*)、Same format & type single item list, or format item list.

```
{
   <CPNAME>
   <CEPVAL>
}
```

■Format
l, b, bool, a, j, i*, f*, u*

■Related Messages
S2F49 Enhanced Remote Command
S2F50 Enhanced Remote Command Acknowledge

10.1.19 COMMACK

■Description
Communication establishment confirmation code. 1 byte.

Value	Description
0	Acknowledged
1	Denied. Retry
2~63	Reserved

■Format
b[1]

■Related Messages
S1F14 Establish Communication Request Acknowledge (CRA)

10.1.20 CPACK

■Description
Command parameter confirmation code. 1 byte.

Value	Description
1	Parameter name (*CPNAME*) does not exist
2	Illegal value specified for *CPVAL* use
3	Illegal format specified for *CPVAL* use
>3	Other equipment-specific error
4~63	Reserved

■Format
b[1]

■Related Messages
S2F42 Host Command Acknowledge(HCA)

10.1.21 CPNAME

■Description
Command parameter name

■Format
a

■Related Messages
S2F41 Host Command Send(HCS)
S2F42 Host Command Acknowledge(HCA)
S2F49 Enhanced Remote Command
S2F50 Enhanced Remote Command Acknowledge

10.1.22 CPVAL

■Description
Command parameter value

■Format
b, bool, a, j, i*, u*

■Related Messages
S2F41 Host Command Send(HCS)
S2F49 Enhanced Remote Command
S2F50 Enhanced Remote Command Acknowledge

10.1.23 DATAID

■Description
Data ID

■Format
u4

■Related Messages
S2F33 Defined Report(DR)
S2F35 Link Event Report(LER)
S2F39 Multi Block Inquire(DMBI)
S2F40 Multi Block Grant(MBG)
S2F49 Enhanced Remote Command
S6F5 Multi Block Data Send Inquire(MBI)
S6F11 Event Report Send(ERS)
S6F16 Event Report Data(ERD)
S15F1 Recipe Management Multi Block Inquire
S15F21 Recipe Action Request
S15F27 Recipe Download Request
S15F29 Recipe Verify Request
S15F35 Recipe Delete Request

10.1.24 DATALENGTH

■Description
Total number of bytes of sent data

■Format
u4

■Related Messages
S2F39 Multi Block Inquire(DMBI)
S6F5 Multi Block Data Send Inquire(MBI)

10.1.25 DRACK

■Description
Defined report acknowledgement code. 1 byte.

Value	Description
0	Acknowledged
1	Denied. Insufficient space

[9] ?

2	Denied. Invalid format
3	Denied. 1 or more *RPTID already defined*
4	Denied. 1 or more *VID does not exist*
>4	Other error
5~63	Reserved

■Format
b[1]

■Related Messages
S2F34 Defined Report Grant(DRA)

10.1.26 EAC

■Description
Equipment verification code. 1 byte.

Value	Description
0	Acknowledged
1	Denied. 1 or more constants do not exist
2	Denied. Busy
3	Denied. 1 or more constants out of range
>3	Other equipment-specific error
4~63	Reserved

■Format
b[1]

■Related Messages
S2F16 New Equipment Constant Acknowledge (ECA)

10.1.27 ECDEF

■Description
Equipment constant default value

■Format
b, bool, a, j, i*, f*, u*

■Related Messages
S2F30 Equipment Contant Namelist(ECN)

10.1.28 ECID

■Description
Equipment constant ID

■Format
u4

■Related Messages
S2F13 Equipment Contant Request(ECR)
S2F15 New Equipment Constant Send(ECS)
S2F29 Equipment Constant Namelist Request(ECNR)
S2F30 Equipment Contant Namelist(ECN)

10.1.29 ECMAX

■Description
Equipment constant maximum value

■Format
b, bool, a, j, i*, f*, u*

■Related Messages
S2F30 Equipment Constant Namelist(ECN)

10.1.30 ECMIN

■Description
Equipment constant minimum value

■Format
b, bool, a, j, i*, f*, u*

■Related Messages
S2F30 Equipment Constant Namelist(ECN)

10.1.31 ECNAME

■Description
Equipment constant name

■Format
a

■Related Messages
S2F30 Equipment Constant Namelist(ECN)

10.1.32 ECV

■Description
Equipment constant

■Format
b, bool, a, j, i*, f*, u*

■Related Messages
S2F14 Equipment Constant Data(ECD)
S2F15 New Equipment Constant Send(ECS)

10.1.33 EDID

■Description
Data ID expected to be received. The following are the 3 possibilities.

MEXP	EDID	EDID
S2F3	<SPID>	A[6]
S3F13	<PTN>	B[1]
S7F3	<PPID>	A[16],B[16]

■Format
b, a, i*, u*

■Related Messages
S9F13 Conversation Timeout(CTN)

10.1.34 ERACK

■Description
Enable/Disable event report. Check code. 1 byte.

Value	Description
0	Acknowledged
1	Denied. 1 or more *CEID* does not exist
>1	Other error
2~63	Reserved

■Format
b[1]

■Related Messages
S2F38 Enable/Disable Event Report Acknowledge (EERA)

10.1.35 ERRCODE

■Description
Error classification code

Value	Description
0	No error
1	Unknown object in object specifier
2	Unknown target object type
3	Unknown object instance
4	Unknown attribute name
5	Read-only attribute. Access denied.
6	Unknown object type
7	Invalid attribute value

8	Syntax error
9	Validation error
10	Verification error
11	Object specifier in use
12	Incorrect parameter specified
13	Not all parameters needing specification are specified
14	Requested option not supported
15	In use
16	Processing preparations incomplete
17	Command invalid in current status
18	No changed materials
19	Materials partially processed
20	Materials all processed
21	Recipe setting-related error
22	Failed, during processing
23	Failed, not during processing
24	Failed due to insufficient materials
25	Job abort
26	Job stop
27	Job cancel
28	Cannot change selected recipe
29	Undefined event
30	Duplicate report ID
31	Undefined data report
32	Data report not linked
33	Undefined trace report
34	Duplicate trace ID
35	Too many data reports
36	Sample period out of range
37	Group size too large
38	Recovery action currently invalid
39	Other recovery currently underway which prevents requested recovery
40	No active recovery action
41	Exceptional recovery failure
42	Exceptional recovery abort
43	Invalid table element
44	Undefined table element
45	Previously set item cannot be deleted
46	Invalid token
47	Invalid parameter
48~63	Reserved

■Format

u*

■Related Messages

S14F2 Get Attribute Data(GAD)
S15F22 Recipe Action Acknowledge
S15F28 Recipe Download Acknowledge
S15F30 Recipe Verify Acknowledge
S15F32 Recipe Unload Data
S15F36 Recipe Delete Acknowledge

10.1.36 ERRTEXT

■Description

Character string showing error displayed in *ERRCODE*.
Max *80* characters

■Format

a

■Related Messages

S14F2 Get Attribute Data(GAD)
S15F22 Recipe Action Acknowledge
S15F28 Recipe Download Acknowledge
S15F30 Recipe Verify Acknowledge
S15F32 Recipe Unload Data
S15F36 Recipe Deletion Acknowledge

10.1.37 ERRW7

■Description

Character string showin error found in process program

■Format

a

■Related Messages

S7F27 Process Program Verification Send(PVS)

10.1.38 GRANT

■Description

Grant code. 1 byte.

Value	Description
0	Authorized
1	Busy. Retry
2	Receiving space insufficient
3	Duplicate *DATAID*
>3	Equipment-specific error code
4~63	Reserved

■Format

b[1]

■Related Messages

S2F40 Multi Block Grant(MBG)

10.1.39 GRANT6

■Description

Send Grant. 1 byte.

Value	Description
0	Send authorized
1	Busy. Retry request
2	Not required
>2	Other error
3~63	Reserved

■Format

b[1]

■Related Messages

S6F6 Multi Block Grant(MBG)

10.1.40 HCACK

■Description

Host command parameter confirmation code. 1 byte.

Value	Description
0	Confirmed. Command was executed
1	Command does not exist
2	Currently cannot execute
3	1 or more parameters invalid
4	Confirmed. Command was executed and completion announced via event.
5	Denied. Already in requested status
6	Object does not exist
7~63	Reserved

■Format

b[1]

■Related Messages

S2F42 Host Command Acknowledge(HCA)
S2F50 Enhanced Remote Command Acknowledge

10.1.41 LENGTH

■Description

Byte length of service program or process program.

■Format

i*, u*

■Related Messages

S7F1 Process Program Load Inquire(PPI)
S7F29 Process Program Verification Inquire(PVI)

10.1.42 LINKID

■Description
Used to link operation execution request and completion message.*LINKID*はis set in *RMOPID* value included in first request. In exceptional cases, it is the completion message that will be sent last, and in those cases it is set to 0.

■Format
u4

■Related Messages
S15F22 Recipe Action Acknowledge
S15F30 Recipe Verify Acknowledge

10.1.43 LRACK

■Description
Link report confirmation code. 1 byte.

Value	Description
0	Acknowledged
1	Denied. Insufficient space
2	Denied. Invalid format
3	Denied. 1 or more *CEID* link already defined
4	Denied. 1 or more *CEID* does not exist
5	Denied. 1 or more *RPTID* does not exist
>5	Other error
6~63	Reserved

■Format
b[1]

■Related Messages
S2F36 Link Event Report Acknowledge(LERA)

10.1.44 MDLN

■Description
Equipment model. Max 6 bytes.

■Format
a

■Related Messages
SlF2 Online Data(D)
S1F13 Establish Communication Request(CR)
S1F14 Establish Communication Request Acknowledge (CRA)
S7F23 Formatted Process Program Send(EPS)
S7F26 Formatted Program Data (FPD)
S7F27 Process Program Verification Send(PVS)

10.1.45 MEXP

■Description
Stream/function which should be received

■Format
a

■Related Messages
S9F13 Conversation Timeout(CTN)

10.1.46 MHEAD

■Description
Message header indicating error

■Format
b

■Related Messages
S9F1 Unrecognized Device ID(UDN)
S9F3 Unrecognized Stream Type (USN)
S9F5 Unrecognized Function Type (UFN)
S9F7 Illegal Data(IDN)
S9F11 Data Too Long(DLN)

10.1.47 OBJACK

■Description
Confirmation code

Value	Description
0	Requested data command executed
1	Error
>1	Reserved

■Format
u1

■Related Messages
S14F2 Get Attribute Data(GAD)

10.1.48 OBJID

■Description
Classifier for objects

■Format
a, u*

■Related Messages
S14F1 Get Attribute Request(GAR)
S14F2 Get Attribute Data(GAD)

10.1.49 OBJSPEC

■Description
Text string having an internal format and used to indicate a particular object instance. This string is made up of a chain of formatted sub-strings, each specifying the object type and classifier. Sub-string formats are made up of the following 4 fields.

- Object type
- Colon “:”
- Object classifier
- Inequality sign “>”

The colon “:” is used at the end of the object type. The inequality sign “>” is used at the end of the classifier field. Object types are also determined through other methods, so this may be omitted. The final “>” sign is optional.

■Format
a

■Related Messages
S2F49 Enhanced Remote Command
S14F1 Get Attribute Request(GAR)
S14F2 Get Attribute Data(GAD)
S15F21 Recipe Action Request
S15F28 Recipe Download Acknowledge
S15F29 Recipe Validation Request
S15F30 Recipe Verify Acknowledge
S15F35 Recipe Deletion Request

10.1.50 OBJTYPE

■Description
Classifier for object groups or classes. It must be possible for all of the same type of object to use the same attribute set.

■Format
a, u*

■Related Messages
S14F1 Get Attribute Request(GAR)

10.1.51 OFLACK

■Description
Confirmation code for offline requests

Value	Description
0	Offline acknowledgement
1~63	Reserved

■Format
b

■Related Messages
S1F16 Offline Request Acknowledge (OFLA)

10.1.52 ONLACK

■Description
Confirmation code for online requests

Value	Description
0	Online acknowledgement
1	Online not authorized
2	Equipment already online
3~63	Reserved

■Format
b

■Related Messages
S1F18 Online Request Acknowledge(ONLA)

10.1.53 OPID

■Description
Operation ID. Unique integer greated by operation requestor, used when multiple completion confirmations occur.

■Format
u1

■Related Messages
S15F21 Recipe Action Request
S15F29 Recipe Validation Request
S15F30 Recipe Verify Acknowledge

10.1.54 PPARM

■Description
Process parameter. Parameter giving information necessary to complete processing command. Number or true/false SECS data item. One or multiple values or character string.

■Format
bool, a, i*, f*, u*

■Related Messages
S7F23 Formatted Process Program Send(EPS)
S7F26 Formatted Program Data(FPD)
S7F27 Process Program Verification Send(PVS)

10.1.55 PPBODY

■Description
Process program main body. This consists of the operations to perform to process the materials received by the equipment, in the equipment's own language.

■Format
b, a, i*, u*

■Related Messages
S7F3 Process Program Send(PPS)
S7F6 Process Program Data(PPD)

10.1.56 PPGNT

■Description
Process program Grant status. 1 byte.

Value	Description
0	OK。
1	Already has
2	No space
3	Invalid *PPID*
4	Busy. Retry request
5	Unauthorized
>5	Other error
6~63	Reserved

■Format
b[1]

■Related Messages
S7F2 Process Program Load Grant(PPG)
S7F30 Process Program Verification Grant(PVG)

10.1.57 PPID

■Description
Process Program ID. Max 80 bytes. *PPID* format is host-dependent. When used inside the equipment, *PPID* is handled as a binary pattern. If there is no device to display the code that is sent, it will be in hexadecimal form.

■Format
b, a

■Related Messages
S7F1 Process Program Load Inquire(PPI)
S7F2 Process Program Load Grant (PPG)
S7F3 Process Program Send(PPS)
S7F4 Process Program Acknowledge(PPA)
S7F5 Process Program Request(PPR)
S7F6 Process Program Data(PPD)
S7F17 Delete Process Program Command(DPS)
S7F18 Delete Process Program Acknowledge(DPA)
S7F19 Current EPPD Request(RER)
S7F20 Current EPPD Data(RED)
S7F23 Formatted Process Program Send(EPS)
S7F24 Formatted Process Program Acknowledge(FPA)
S7F25 Formatted Process Program Request(FPR)
S7F26 Formatted Program Data(FPD)
S7F27 Process Program Verification Send(PVS)
S7F30 Process Program Verification Grant(PVG)

10.1.58 RCMD

■Description
Remote control command code or command string

■Format
a

■Related Messages
S2F41 Host Command Send(HCS)
S2F49 Enhanced Remote Command

10.1.59 RCPATTRDATA

■Description
Recipe attribute contents (values)

■Format
l, b, bool, a, i*, f*, u*

■Related Messages
S15F27 Recipe Download Request
S15F28 Recipe Download Acknowledge
S15F30 Recipe Verify Acknowledge
S15F32 Recipe Unload Data

10.1.60 RCPATTRID

■Description
Non-classifier attribute name (classifier)

■Format
a

■Related Messages
S15F27 Recipe Download Request
S15F28 Recipe Download Acknowledge
S15F30 Recipe Verify Acknowledge
S15F32 Recipe Unload Data

10.1.61 RCPBODY

■Description
Recipe main body

■Format
b, a, i*, u*

■Related Messages
S15F27 Recipe Download Request
S15F32 Recipe Unload Data

10.1.62 RCPCMD

■Description
Indicates actions executed by recipe

Value	Description
0~4	Reserved
5	Delete
6~7	Reserved
8	No save
9	Save
10	Validate
11	Link
12	Clear link
13	Authenticate
14	Clear verification
15	Download
16	Upload
17~63	Reserved

■Format
u1

■Related Messages
S15F21 Recipe Action Request
S15F22 Recipe Action Acknowledge

10.1.63 RCPDEL

■Description

Value	Description
0	Delete
1	Clear selection
>1	Reserved

■Format
u1

■Related Messages
S15F35 Recipe Delete Request

10.1.64 RCPID

■Description
Recipe classifier. Formatted text complies with *OBJSPEC* requirements.

■Format
a

■Related Messages
S15F21 Recipe Action Request
S15F28 Recipe Download Acknowledge
S15F29 Recipe Validation Request
S15F30 Recipe Verify Acknowledge
S15F35 Recipe Delete Request

10.1.65 RCPOWCODE

■Description
Indicates whether or not previously existing recipe will be overwritten at time of download.

Value	Description
TRUE	Overwrite
FALSE	No overwrite

■Format
bool

■Related Messages
S15F27 Recipe Download Request

10.1.66 RCPSPEC

■Description
Recipe specifier. Recipe object specifier.

■Format
a

■Related Messages
S15F1 Recipe Management Multi Block Inquire
S15F27 Recipe Download Request
S15F31 Recipe Unload Request
S15F32 Recipe Unload Data

10.1.67 RESPEC

■Description
Recipe executor object specifier

■Format
a

■Related Messages
S15F29 Recipe Validation Request
S15F35 Recipe Delete Request

10.1.68 RMACK

■Description
Communicates whether the requested action was completed successfully, was denied, was completed due to an error, or was completed with notification to requestor.

Value	Description
0	Successfully completed
1	Cannot execute target action
2	Completed due to error
3	Target action completed and notification

	sent.
4	Target action existence not required

■Format
u1

■Related Messages
S15F22 Recipe Action Acknowledge
S15F28 Recipe Download Acknowledge
S15F30 Recipe Verify Acknowledge
S15F32 Recipe Unload Data
S15F36 Recipe Delete Acknowledge

10.1.69 RMDATASIZE

■Description
Maximum length of multi block message expressed in bytes. Used to make receiver determine whether or not expected message exceeds receiver capacity.

■Format
u*

■Related Messages
S15F1 Recipe Management Multi Block Inquire

10.1.70 RMGRANT

■Description
Grant code. Used to grant or deny a request. 1 byte.

Value	Description
0	Granted
1	Cannot currently grant. Retry
2	No space
3	Request on standby
4~64	Reserved

■Format
b[1]

■Related Messages
S15F2 Recipe Management Multi Block Grant

10.1.71 RMNSSPEC

■Description
Recipe name space object specifier

■Format
a

■Related Messages
S15F21 Recipe Action Request

10.1.72 RPTID

■Description
Report ID

■Model
u4

■Related Messages
S2F33 Defined Report(DR)
S2F34 Defined Report Acknowledge(DRA)
S2F35 Link Event Report(LER)
S2F36 Link Event Report Acknowledge(LERA)
S6F11 Event Report Send(ERS)
S6F16 Event Report Data(ERD)
S6F19 Individual Report Request(IRR)
S6F20 Individual Report Data(IRD)

10.1.73 SEQNUM

■Description
Command number. Commands specified by numbers indicating position in process command list. In process program first command.*SEQNUM* is 1.

■Format
i*, u*

■Related Messages
S7F27 Process Program Verification Send(PVS)

10.1.74 SHEAD

■Description
Message header related to transaction timer.

■Format
b

■Related Messages
S9F9 Transaction Timer Timeout(TIN)

10.1.75 SOFTREV

■Description
Software revision code. Max 6 bytes.

■Format
a

■Related Messages
SlF2 Online Data(D)
S1F13 Establish Communication Request(CR)
S1F14 Establish Communication Request Acknowledge (CRA)
S7F23 Formatted Process Program Send(EPS)
S7F26 Formatted Program Data(FPD)
S7F27 Process Program Verification Send(PVS)

10.1.76 SV

■Description
Status variable data

■Format
l, b, bool, a, j, i*, f*, u*

■Related Messages
S1F4 Specify Equipment Status Data(SSD)

10.1.77 SVID

■Description
Status variable ID. Status variables include all parameters that are sampled over time increments, such as temperature, amount of consumables, etc.

■Format
u4

■Related Messages
SlF3 Specify Equipment Status Request(SSR)
S1F4 Specify Equipment Status Data(SSD)
S1F11 Status Variable Namelist Request(SVNR)
S1F12 Status Variable Namelist Reply(SVNRR)

10.1.78 SVNAME

■Description
Status variable name

■Format
a

■Related Messages

S1F12 Status Variable Namelist Reply(SVNRR)

10.1.79 TEXT

■Description
One-line character

■Format
b, a, a2, i*, u*

■Related Messages
S10F1 Terminal Request(TRN)
S10F3 Terminal Display, Single Block (VTN)
S10F5 Terminal Display, Multi Block (VTN)

10.1.80 TIACK

■Description
Time confirmation code. 1 byte.

Value	Description
0	OK.
1	Error. Not acknowledged
2~63	Reserved

■Format
b[1]

■Related Messages
S2F32 Date and Time Set Acknowledge(DTA)

10.1.81 TID

■Description
Terminal number. 1 byte.

Value	Description
0	Single or primary terminal
>0	Additional terminal at same equipment

■Format
b[1]

■Related Messages
S10F1 Terminal Request(TRN)
S10F3 Terminal Display, Single Block(VTN)
S10F5 Terminal Display, Multi Block(VTN)
S10F7 Multi Block Not Allowed(MNN)

10.1.82 TIME

■Description
Time. 12 or 16 bytes.

12 Bytes		
YYMMDDhhmmss		
YY	Yr	00~99
MM	Mo	01~12
DD	Day	01~31
hh	Hr	00~23
mm	Min	00~59
ss	Sec	00~59

16 Bytes		
YYYYMMDDhhmmsscc		
YYYY	Yr	0000~9999
MM	Mo	01~12
DD	Day	01~31
hh	Hr	00~23
mm	Min	00~59
ss	Sec	00~59
cc	1/100 Sec	00~99

■Notes
The 16-byte format is currently an option. The 16-byte format is a requirement for new or updated implementations on and after January 1, 1998. The 12-byte format will continue to be supported as a configurable option using the equipment constant TimeFormat. This item only refers to the required conditions for format and is unrelated to precision and accuracy.

■Format
a

■Related Messages
S2F18 Date and Time Data(DTD)
S2F31 Date and Time Set Request(DTS)

10.1.83 UNITS

■Description
Recognises units. Units are those allowed by Items E5 and 9 (Measurement Units)

■Format
a

■Related Messages
S1F12 Status Variable Namelist Reply(SVNRR)
S2F30 Equipment Constant Namelist(ECN)

10.1.84 V

■Description
Variable data

■Format
l, b, bool, a, j, i*, f*, u*

■Related Messages
S6F11 Event Report Send(ERS)
S6F16 Event Report Data(ERD)
S6F20 Individual Report Data(IRD)

10.1.85 VID

■Description
Variable ID

■Format
u4

■Related Messages
S2F33 Defined Report(DR)
S2F34 Defined Report Acknowledge(DRA)

10.2 Messages

Primary Messages	Secondary Messages	Description
S1F1	S1F2	Online Acknowledge Request
S1F3	S1F4	Specify Equipment Status Request
S1F11	S1F12	Status Variable Namelist Request
S1F13	S1F14	Establish Communication Request
S1F15	S1F16	Request Offline
S1F17	S1F18	Online Request
S2F13	S2F14	Equipment Constant Request
S2F15	S2F16	New Equipment Constant Send
S2F17	S2F18	Date and Time Request
S2F29	S2F30	Equipment Constant Namelist Request
S2F31	S2F32	Date and Time Set Request
S2F33	S2F34	Defined Report
S2F35	S2F36	Link Event Report
S2F37	S2F38	Enable/Disable Event Report
S2F39	S2F40	Multi Block Inquire
S2F41	S2F42	Host Command Send
S2F49	S2F50	Enhanced Remote Command
S5F1	S5F2	Alarm Report Send
S5F3	S5F4	Enable/Disable Alarm Send
S5F5	S5F6	List Alarm Request
S6F5	S6F6	Multi Block Data Send Inquire
S6F11	S6F12	Event Report Send
S6F15	S6F16	Event Report Request
S6F19	S6F20	Invididual Report Request
S7F1	S7F2	Process Program Inquire
S7F3	S7F4	Process Program Send
S7F5	S7F6	Process Program Request
S7F17	S7F18	Delete Process Program Command
S7F19	S7F20	Current EPPD Request
S7F23	S7F24	Formatted Process Program Send
S7F25	S7F26	Formatted Process Program Request
S7F27	S7F28	Process Program Verification Send
S7F29	S7F30	Process Program Verification Inquire
S9F1		Unrecognized Device ID
S9F3		Unrecognized Stream Type
S9F5		Unrecognized Function Type
S9F7		Illegal Data
S9F9		Transaction Timer Timeout
S9F11		Data Too Long
S9F13		Conversation Timeout
S10F1	S10F2	Terminal Request
S10F3	S10F4	Terminal Display, Single Block
S10F5	S10F6	Terminal Display, Multi Block
S10F7		Multi Block Not Allowed
S14F1	S14F2	Get Attribute Request
S15F1	S15F2	Recipe Management Multi Block Inquire
S15F21	S15F22	Recipe Action Request
S15F27	S15F28	Recipe Download Request
S15F29	S15F30	Recipe Validation Request
S15F31	S15F32	Recipe Unload Request
S15F35	S15F36	Recipe Delete Request

10.2.1 SlF1 Online Acknowledge Request(R)

Are You There Request
S,H←→E,Reply

■Description
Confirms whether equipment is online or not. If Function 0 response occurs, communication cannot be initiated. After the equipment has sent the host SlF1 Online Acknowledge Request(R), if Function 0 is received, it has the same meaning as a message receipt timeout.

■Structure
Header only

```
s1f1w
```

10.2.2 SlF2 Online Data (D)

0n Line Data
S,H←→E

■Description
Declaration that communication is online

■Structure

```
s1f2
{
   <a MDLN>
   <a SOFTREV>
}
```

Name	Format	Description
MDLN	a	Equipment model. Max 6 bytes.
SOFTREV	a	Software revision code. Max 6 bytes.

■Exceptions
In the case of the host, a list with length 0 is sent to the equipment.

```
s1f2
{
}
```

10.2.3 SlF3 Selected Equipment Status Request (SSR)

Selected Equipment Status Request
S,H→E,Reply

■Description
Host requests selected status variable from equipment.

■Structure
The following structure is in accordance with all item formats. All new implementations should use this structure.

```
s1f3w
{
   <u4 SVID1>
   .
   .
   <u4 SVIDn>
}
```

The structure shown below is for the purpose of ensuring compatibility with previous implementation.

```
s1f3w
<u4 SVID1 ... SVIDn>
```

Name	Format	Description
SVID	u4	Status variable ID. Status variables include all parameters that are sampled over time increments, such as temperature, amount of consumables, etc.

■Exceptions
Items with length 0 are requests for all SVID.

```
s1f3w
{
}
```

```
s1f3w
<u4>
```

10.2.4 S1F4 Selected Equipment Status Data (SSD)

Se1ected Equipment Status Data
M,H←E

■Description
Reports each SV value in the order they were requested by the equipment. The host must remember which SVID were requested.

■Structure

```
s1f4
{
   <u4 SV1>
   .
   .
   <u4 SVn>
}
```

Name	Format	Description
SV	l, b, bool, a, j, i*, f*, u*	Status variable data
SVID	u4	Status variable ID. Status variables include all parameters that are sampled over time increments, such as temperature, amount of consumables, etc.

■Exceptions
Lists with length 0 indicate that there is no response data.

```
s1f4
{
}
```

If SVi is an item with length 0, it indicates that SVIDi does not exist.

```
s1f4
{
   <u4>
}
```

10.2.5 S1F11 Status Variable Namelist Request (SVNR)

```
Status Variable Namelist Request
S,H→E, Reply
```

■ Description

Request for status variable confirmation from host to equipment.

■Structure

```
s1f11w
{
   <u4 SVID1>
   .
   .
   <u4 SVIDn>
}
```

Name	Format	Description
SVID	u4	Status variable ID. Status variables include all parameters that are sampled over time increments, such as temperature, amount of consumables, etc.

■Exceptions

Lists with length 0 are a request for all reports regarding SVID.

```
s1f11w
{
}
```

10.2.6 S1F12 Status Variable Namelist Reply (SVNRR)

```
Status Variable Namelist Reply
M,H←E
```

■Description

Reports name and units of status variable requested by equipment.

■Structure

```
s1f12
{
   {
     <u4 SVID1>
     <a SVNAME1>
     <a UNITS1>
   }
   .
   .
   {
     <u4 SVIDn>
     <a SVNAMEn>
     <a UNITSn>
   }
}
```

Name	Format	Description
SVID	u4	Status variable ID. Status variables include all parameters that are sampled over time increments, such as temperature, amount of consumables, etc.
SVNAME	a	Status variable name
UNITSn	a	Recognizes units. Units are those permitted by Items E5 and 9 (Measurement Units)

■Exceptions

SVNAMEi UNITSi both having character string item length 0 indicate that the SVID does not exist.

```
s1f12
{
   {
     <u4 SVID1>
     <a>
     <a>
   }
}
```

10.2.7 S1F13 Establish Communication Request(CR)

```
Establish Communication Request
S,H←→E, Reply
```

■Description

The objective of this message is to give the formal meaning of "communication start" in logical application levels when powering on following termination of communication or aborting of communication. This must be the first message sent following aborting of communication.

Using the confirmation reply code that acknowledges establishment, attempts to send S1F13 Establish Communication Request(CR) by the time that S1F14 Establish Communication Request Reply(CRA) is received during the transaction timeout period should be repeated over the programmed interval.

■Structure

```
s1f13w
{
   <a MDLN>
   <a SOFTREV>
}
```

Name	Format	Description
MDLN	a	Equipment model. Max 6 bytes.
SOFTREV	a	Software revision code. Max 6 bytes.

■Exceptions

Host sends list of length 0 to equipment.

```
s1f13w
{
}
```

10.2.8 S1F14 Establish Communication Request Acknowledge (CRA)

```
Establish Communication Request Acknowledge
S,H←→E
```

■Description

S1F13 Establish Communication (CR) Acknowledge or denial. *MDLN and SOFTREV* are online data andonly

valid when *COMMACK*=0.

■Structure

```
s1f14
{
   <b COMMACK>
   {
     <a MDLN>
     <a SOFTREV>
   }
}
```

Name	Format	Description
COMMACK	b	Establish communication acknowledge code. 1 byte. 0 Acknowleged 1 Denied. Retry 2~63 Reserved
MDLN	a	Equipment model. Max 6 bytes.
SOFTREV	a	Software revision code. Max 6 bytes.

■Exceptions

Host sends list of length 0 to equipment.

```
s1f14
{
   <b COMMACK>
   {
   }
}
```

10.2.9 S1F15 Request Offline (ROFL)

```
Request OFF-LINE
S,H→E, Reply
```

■Description

Host requests transition to offline status to equipment.

■Structure

Header only

```
s1f15w
```

10.2.10 S1F16 Offline Request Acknowledge(OFLA)

```
OFF-LINE Acknowledge
S,H←E
```

■Description

OK or NG reply to S1F15 Request Offline (ROFL)

■Structure

```
s1f16
<b OFLACK>
```

Name	Format	Description
OFLACK	b	Acknowledgement code versus offline request 0 Offine acknowledged 1~63 Reserved

10.2.11 S1F17 Request Online (RONL)

```
Request ON-LINE
S,H→E, 返信
```

■Description

Host requests transition to online status to equipment.

■Structure

Header only

```
s1f17w
```

10.2.12 S1F18 Online Request Acknowledge(ONLA)

```
ON-LINE Acknowledge
S,H←E
```

■Description

OK or NG reply to S1F17 Request Online(RONL)

■Structure

```
s1f18
<b ONLACK>
```

Name	Format	Description
ONLACK	b	Acknowledgement code versus online request 0 Online acknowledged 1 Online not granted 2 Equipment already online 3~63 Reserved

10.2.13 S2F13 Equipment Constant Request(ECR)

```
Equipment Constant Request
S,H→E, Reply
```

■Description

Query constants that almost never change with this message, such as offsets, servo gain, alarm limit values, data collection modes, etc.

■Structure

The structure below is in accordance with all item formats. Any new implementations should use this structure.

```
s2f13w
{
   <u4 ECID1>
   .
   .
   <u4 ECIDn>
}
```

The structure shown below is for the purpose of ensuring compatibility with previous implementations.

```
s2f13w
<u4 ECID1 ... ECIDn>
```

Name	Format	Description
ECID	u4	Equipment

		Constant ID.

■Exceptions

Lists (Structure 1) or Items with length 0 (Structure 2) request all constants in the order they were predefined.

```
s2f13w
{
}
```

```
s2f13w
<u4>
```

10.2.14 S2F14 Equipment Constant Data (ECD)

```
Equipment Constant Data
M,H←E
```

■Description

Reply constants S2F13 Equpment Constant Request(ECR) come in the order they were requested.

■Structure

```
s2f14
{
   <ECV1>
   .
   .
   <ECVn>
}
```

Name	Format	Description
ECV	b, bool, a, j, i*, f*, u*	Equipment constant

■Exceptions

If ECVi is a list with length 0, it indicates that the target ECIDi is not present. The list format of this data item is prohibited except in this case.

```
s2f14
{
   {
   }
}
```

10.2.15 S2F15 New Equipment Constant Send 定数変更 (ECS)

```
New Equipment Constant Send
S,H→E,Reply
```

■Description

Changes one or multiple equipment constants

■Structure

```
s2f15w
{
   {
     <u4 ECID1>
     <ECV1>
   }
   .
   .
   {
     <u4 ECIDn>
     <ECVn>
   }
}
```

Name	Format	Description
ECID	u4	Equipment constant ID
ECV	b, bool, a, j, i*, f*, u*	Equipment constant

10.2.16 S2F16 New Equipment Constant Acknowledge (ECA)

```
New Equipment Constant Acknowledge
S,H←E
```

■Description

OK or NG reply to S2F15 New Equipment Constant Send(ECS). If EAC is an error code other than 0, no value in the ECID selected by the equipment in S2F15 should be changed.

■Structure

```
s2f16
<b EAC>
```

Name	Format	Description
EAC	b	Equipment acknowledgement code. 1 byte. 0 Acknowledged 1 Denied. 1 or more constants do not exist 2 Denied. Busy 3 Denied. 1 or more constants are out of range. >3 Other equipment-specific error 4~63 Reserved

10.2.17 S2F17 Date and Time Request(DTR)

```
Date and Time Request
S,H←→E, Reply
```

■Description

Used to synchronize equipment time base checks and host time base.

■Structure

Header only

```
s2f17w
```

10.2.18 S2F18 Date and Time Data (DTD)

```
Date and Time Data
S,H←→E
```

■Description

Current time

■Structure

```
s2f18
<a TIME>
```

Name	Format	Description
TIME	a	Time. 12 or 16 bytes. If 12 bytes, format is: YYMMDDhhmmss If 16 bytes, format is: YYYYMMDDhhmmsscc

■Exceptions

Items with length 0 indicate that it does not have a clock.

```
s2f18
<a>
```

10.2.19 S2F29 Equipment Constant Namelist Request (ECNR)

```
Equipment Constant Namelist Request
S,H→E, Reply
```

■Description

Host collects basic information regarding valid equipment constants inside the equipment.

■Structure

```
s2f29w
{
  <u4 ECID1>
  .
  .
  <u4 ECIDn>
}
```

Name	Format	Description
ECID	u4	Equipment constant ID

■Exceptions

If length is 0, sending of all *ECID* information is indicated.

10.2.20 S2F30 Equipment Constant Namelist(ECN)

```
Equipment Constant Namelist
M,H←E
```

■Description

Reply to S2F29 Equipment Constant Namelist Request (ECNR)

■Structure

```
s2f30
{
  {
    <u4 ECID1>
    <a ECNAME1>
    <ECMIN1>
    <ECMAX1>
    <ECDEF1>
    <a UNITS1>
  }
  .
  .
  {
    <u4 ECIDn>
    <a ECNAMEn>
    <ECMIINn>
    <ECMAXn>
    <ECDEFn>
    <a UNITSn>
  }
}
```

Name	Format	Description
ECID	u4	Equipment constant ID
ECNAME	a	Equip. constant name
ECMIN	b, bool, a, j, i*, f*, u*	Minimum equipment constant value
ECMAX	b, bool, a, j, i*, f*, u*	Maximum equipment constant value
ECDEF	b, bool, a, j, i*, f*, u*	Equipment constant default value
UNITS	a	Recognizes units. Units are those permitted in Items E5 and 9 (Meas. Units)

■Exceptions

Character string items with *ECNAMEi*、*ECMINi*、*ECMAXi*、*ECDEFi*、and *UNITSi* length of 0 indicate that the ECID does not exist.

```
s2f30
{
  {
    <u4 ECID1>
    <a>
    <a>
    <a>
    <a>
    <a>
  }
}
```

10.2.21 S2F31 Date and Time Set Request(DTS)

```
Date and Time Set Request
S,H→E, Reply
```

■Description

Used to synchronize equipment time to host computer time base.

■Structure

```
s2f31w
<a TIME>
```

Name	Format	Description
TIME	a	Time. 12 or 16 bytes. If 12-byte, the format is: YYMMDDhhmmss If 16-byte, the format is: YYYYMMDDhhmmsscc

10.2.22 S2F32 Date and Time Set Acknowledge (DTA)

```
Date and Time Set Acknowledge
S,H←E
```

■Description

Acknowledges date and time setting

■Structure

```
s2f32
<b TIACK>
```

Name	Format	Description
TIACK	b	Time acknowledgement code. 1

		byte. 0 OK。 1 Error. Not acknowledged 2~63 Reserved

10.2.23 S2F33 Define Report (DR)

Define Report
M,H→E, Reply

■Description

The objective of this message is to specify the series of reports to the equipment from the host computer. The type of report sent is specified to in Boolean format, in accordance with equipment constants. False equipment constants indicate that S6F11 Event Report Send(ERS) will be sent, and True equipment constants indicate that S6F13 Annotated Event Report Send(AERS) will be sent.[10]. If S2F33 is multi block, S2F39 Multi Block Inquire (DMBI) and S2F40 Multi Block Grant(MBG) transactions must precede.

■Structure

```
s2f33w
{
   <u4 DATAID>
   {
      {
         <u4 RPTID1>
         {
            <u4 VID1>
            .
            .
            <u4 VIDb>
         }
      }
      {
         <u4 RPTIDa>
         {
            <u4 VID1>
            .
            .
            <u4 VIDc>
         }
      }
   }
}
```

Name	Format	Description
DATAID	u4	Data ID
RPTID	u4	Report ID
VID	u4	Variable ID

■Exceptions

Lists of length 0 which follow *DATAID* clear all report regulations and related links. Please refer to S2F35 Link Event Report(LER).

```
s2f33w
{
   <u4 DATAID>
   {
   }
}
```

Lists of length 0 which follow *RPTID* clear report type *RPTID*. All *CEID* links to this *RPTID* will also be deleted.

```
s2f33w
{
   <u4 DATAID>
   {
      {
         <u4 RPTID>
         {
         }
      }
   }
}
```

Name	Format	Description
CEID	u4	Collection event ID

10.2.24 S2F34 Define Report Acknowledge (DRA)

Define Report Acknowledge
S,H←E

■Description

Acknowledgement or error. When an error status is detected, all messages are denied. In other words, partial changes are not allowed.

■Structure

```
s2f34
<b DRACK>
```

Name	Format	Description
DRACK	b	Define report acknowledgement code. 1 byte. 0 Acknowledged 1 Denied. Insufficient space 2 Denied. Invalid format 3 Denied. 1 or more *RPTID* is already defined. 4 Denied. 1 or more *VID* does not exist. >4 Other error 5~63 Reserved
RPTID	u4	Report ID
VID	u4	Variable ID

10.2.25 S2F35 Link Event Report(LER)

Link Event Report
M,H→E, Reply

■Description

The objective of this message is to link the host computer to event ID (*CEID*) which communicate reports . These linked event reports are disabled and not executed even when linked. In other words, even if an event occurs, reports will not be sent unless they are set to enabled status.

If S2F35 is a multi block, S2F39 Multi Block Inquire(DMBI) and S2F40 Multi Block Grant (MBG) transactions must precede.

■Structure

```
s2f35w
{
   <u4 DATAID>
   {
      {
         <u4 CEID1>
         {
            <u4 RPTID1>
            .
            .
            <u4 RPTIDb>
         }
      }
      .
      .
      {
```

[10]In GEM, S6F13 is not used.

```
      <u4 CEIDa>
      {
        <u4 RPTID1>
        .
        .
        <u4 RPTIDc>
      }
    }
  }
}
```

Name	Format	Description
DATAID	u4	Data ID
CEID	u4	Collection event ID
RPTID	u4	Report ID

■Exceptions

Lists of length 0 which follow *CEID* clear all reports linked to that event.

```
s2f35w
{
  <u4 DATAID>
  {
    {
      <u4 CEID>
      {
      }
    }
  }
}
```

10.2.26 S2F36 Link Event Report Acknowledge (LERA)

```
Link Event Report Acknowledge
S,H←E
```

■Description

Acknowledgement or error. If an error status is detected, all messages will be denied. In other words, partial changes are not allowed.

■Structure

```
s2f36
<b LRACK>
```

Name	Format	Description
LRACK	b	Link report acknowledgement code. 1 byte. 0 Acknowledged 1 Denied. Insufficient space 2 Denied. Invalid format 3 Denied. 1 or more *CEID* links are already defined. 4 Denied. 1 or more *CEID* does not exist. 5 Denied. 1 or more *RPTID* does not exist. >5 Other error 6~63 Reserved
CEID	u4	Collection event ID.
RPTID	u4	Report ID.

10.2.27 S2F37 Enable/Disable Event Report(EDER)

```
Enable/Disable Event Report
S,H→E, Reply
```

■Description

The objective of this message is for the host computer to enable or diable the series of reports with respect to communication event IDs (*CEIDs*) .

■Structure

```
s2f37w
{
  <bool CEED>
  {
    <u4 CEID1>
    .
    .
    <u4 CEIDn>
  }
}
```

Name	Format	Description
CEED	bool	Collection event or trace enable/disable code. 1 byte. False Disable True Enable
CEID	u4	Collection event ID

■Exceptions

Lists of length 0 following *CEED* indicate all *CEIDs*.

```
s2f37w
{
  <bool CEED>
  {
  }
}
```

10.2.28 S2F38 Enable/Disable Event Rport Acknowledge (EERA)

```
Enable/Disable Event Report Acknowledge
S,H←E
```

■Description

Acknowledgement or error. If an error status is detected, all messages will be denied. In other words, partial changes are not allowed.

■Structure

```
s2f38
<b ERACK>
```

Name	Format	Description
ERACK	b	Enable/Disable event report acknowledgement code. 1 byte. 0 Acknowledged 1 Denied. 1 or more *CEID* does not exist. >1 Other error 2~63 Reserved
CEID	u4	Collection event ID

10.2.29 S2F39 Multi Block Inquire (DMBI)

```
Multi-block Inquire
S,H→E, Reply
```

■Description

If S2F23 Trace Condition Set(TIS),S2F33 Define Report(DR), S2F35 Link Event Report(LER), S2F45 Define Variable Limit Attribute (DVLA) or S2F49 Enhanced Remote Command messages were more than 1 block, this transaction must always come before the message.

■Structure

```
s2f39w
{
   <u4 DATAID>
   <u4 DATALENGTH>
}
```

Name	Format	Description
DATAID	u4	Data ID
DATALENGTH	u4	Total no. of bytes of send data

10.2.30 S2F40 Multi Block Grant (MBG)

Multi-block Grant
S,H←E

■Description

Grants sending of multi-block messages

■Structure

```
s2f40
<b GRANT>
```

Name	Format	Description
GRANT	b	Grant code. 1 byte. 0 Granted 1 Busy. Retry 2 Insufficient receive space 3 Duplicate *DATAID* >3 Equip-specific error code 4~63 Reserved
DATAID	u4	Data ID

10.2.31 S2F41 Host Command Send (HCS)

Host Command Send
S,H→E, Reply

■Description

Host requests execution of a specific remote command having parameters related to the equipment.

■Structure

```
s2f41w
{
   <a RCMD>
   {
      {
         <a CPNAME1>
         <CPVAL1>
      }
      .
      .
      {
         <a CPNAMEn>
         <CPVALn>
      }
   }
}
```

Name	Format	Description
RCMD	a	Remote control command code or command string
CPNAME	a	Command parameter name
CPVAL	b, bool, a, j, i*, u*	Command parameter value

10.2.32 S2F42 Host Command Acknowledge (HCA)

Host Command Acknowledge
S,H←E

■Description

Host command or error acknowledgement. If a command is not accepted due to 1 or more invalid parameters, an invalid parameter list including the parameter names and reason for invalidity will be returned.

■Structure

```
s2f42
{
   <b HCACK>
   {
      {
         <a CPNAME1>
         <b CPACK1>
      }
      .
      .
      {
         <a CPNAMEn>
         <b CPNAMEn>
      }
   }
}
```

Name	Format	Description
HCACK	b	Host command parameter acknowledgement code. 1 byte. 0 Acknowledged. Command executed. 1 Command does not exist 2 Cannot currently execute 3 1 or more parameters are invalid 4 Acknowledged. Command will be executed and completion announced by event. 5 Denied. Already in requested status. 6 Object does not exist. 7~63 Reserved
CPNAME	a	Command parameter name
CPACK	b	Command parameter acknowledgement code. 1 byte. 1 Parameter name (*CPNAME*) does not exist. 2 Illegal value specified for *CPVAL* use. 3 Illegal format specified for *CPVAL* use. >3 Other equipment-specific error 4~63 Reserved

■Exceptions

If there is no invalid parameter, a list of length 0 will be sent.

```
s2f42
{
   <b HCACK>
   {
   }
}
```

10.2.33 S2F49 Enhanced Remote Command

Enhanced Remote Command
M,H→E

■Description

Host makes request to object which attaches parameters related to the specified remote command. If multi block, S2F39 Multi Block Inquire (DMBI) and S2F40 Multi Block Grant (MBG) transactions must precede.

■Structure

```
s2f49w
{
   <u4 DATAID>
   <a OBJSPEC>
   <a RCMD>
   {
      {
         <a CPNAME1>
         <CEPVAL1>
      }
      {
         <a CPNAME2>
         <CEPVAL2>
      }
      .
      .
      {
         <a CPNAMEm>
         <CEPVALm>
      }
   }
}
```

If a particular value of *CPNAME* has a list and defined *CEPVAL*, it will always be list. If the *CEPVAL* related to the *CPNAME* of that specific value is defined as other than list, the result will be a format error.

Name	Format	Description
DATAID	u4	Data ID
OBJSPEC	a	Text string used to indicate a specific object instance having an internal format. This string is composed of a series of formatted sub-strings, each of which identifies object type and classifier. Sub-string format is composed of the following 4 fields. Object type Colon “:” Object classifier Inequality sign “>” The colon “:” is used at the end of the object type. The inequality sign “>” is used at the end of the classifier. Object type is also determined by other methods so it may be omitted. The final “>” is optional.
RCMD	a	Remote control command or command string.
CPNAME	a	Command parameter name
CEPVAL	l, b, bool, a, j, i*, f*, u*	Command extended parameter value. Since *CPNAME* value is used, if *CEPVAL* has a specific method of use, it will always be known. *CEPVAL* has the following format: Single (not list) value (Ex: *CPVAL*), same format and type single item list, or list of format items.
		<L <CPNAME> <CEPVAL> >
CPVAL	b,bool,a,j,i*,u*	Command parameter value

■Exceptions

Lists of length 0 indicate that parameters were not sent together with commands.

```
s2f49w
{
   <u4 DATAID>
   <a OBJSPEC>
   <a RCMD>
   {
   }
}
```

OBJSPEC may also be items with length 0.

```
s2f49w
{
   <u4 DATAID>
   <a>
   <a RCMD>
   {
   }
}
```

■Notes

If *CEPVAL* is a list, the items in that list will be in one of the following formats.

- List of items with identical formats
- List of combinations of *CPNAME* and *CEPVAL* as shown below.

```
s2f49w
{
   <u4 DATAID>
   <a OBJSPEC>
   <a RCMD>
   {
      {
         <a CEPNAMEa>
         {
            <CEPVALa1>
            <CEPVALa2>
            .
            .
            <CEPVALam>
         }
      }
   }
}
```

```
s2f49w
{
   <u4 DATAID>
   <a OBJSPEC>
   <a RCMD>
   {
      {
         <a CPNAMEb>
         {
            {
               <a CPNAMEb1>
               <CEPVALb1>
            }
            .
            .
            {
               <a CPNAMEbn>
```

```
            <CEPVALbn>
          }
        }
      }
    }
}
```

10.2.34 S2F50 Enhanced Remote Command Acknowledge

Enhanced Remote Command Acknowledge
M,H←E

■Description

This is used for the equipment to acknowledge an enhanced remote command, or report an error. If the command is not accepted due to 1 or more invalid parameters, a list of the invalid parameters showing parameter names and reason for invalidity is returned to (*HCACK*=3) .

■Structure

```
s2f50
{
   <b HCACK>
   {
      {
        <a CPNAME1>
        <u1 CEPACK1>
      }
      .
      .
      {
        <a CPNAMEn>
        <u1 CEPACKn>
      }
   }
}
```

Name	Format	Description
HCACK	b	Host command parameter acknowledgement code. 1 byte 0 Acknowledged. Command executed. 1 Command does not exist 2 Cannot execute now 3 1 or more parameters invalid 4 Acknowledged. Command will be executed and completion announced by event. 5 Denied. Already in requested status 6 Object does not exist 7~63 Reserved
CPNAME	a	Command parameter name。
CEPACK	l,u1	Command enhanced parameter acknowledgement.If a particular value in *CPNAME* is defined as having a list *CEPVAL*, *CEPACK* will be in the same format as the corresponding *CEPVAL* list format so as to be used by S2F49 [11] (enhanced remote command). Except for this case, *CEPACK* will be a 1-byte integer. 0 No error 1 Parameter name (*CPNAME*) does not exist 2 Illegal value specified in *CEPVAL* 3 Illegal format specified in CEPVAL. 4 Parameter name (*CPNAME*) usage method is invalid 5~63 Reserved
CEPVAL	l, b, bool, a, j, i*, f*, u*	Command enhanced parameter value. Since the *CPNAME* value is used, if *CEPVAL* has a specific method of use, it will always be known. *CEPVAL* has the following format： Single (not list) value (Ex：*CPVAL*) , same format and type single item list, or list of format items. <L <CPNAME> <CEPVAL> >
CPVAL	b, bool, a, j, i*, u*	Command parameter value

10.2.35 S5F1 Alarm Report Send (ARS)

Alarm Report Send
S,H←E, Reply

■Description

This message is used to announce alarm status occurrence or clearing. When an alarm is set or cleared, this message is sent. It is OK if there are no corresponding clear messages for unrecoverable errors or waring flags.

■Structure

```
s5f1w
{
   <b ALCD>
   <u4 ALID>
   <a ALTX>
}
```

Name	Format	Description
ALCD	b	Alarm code bit8 1=Alarm status occurred 0=Alarm status cleared bit7~1 Alarm classify code 0 Not used 1 Human safety-related 2 Tool safety-related 3 Parameter control alarm 4 Parameter control error 5 Unrecoverable error 6 Equipment status warning 7 Warning flag 8 Data cannot be guaranteed >8 Other category 9~63 Reserved
ALID	u4	Alarm ID
ALTX	a	Alarm text. Max 40 characters.

10.2.36 S5F2 Alarm Report Acknowledge (ARA)

Alarm Report Acknowledge
S,H→E

■Description

OK or NG reply to S5F1 Alarm Report Send (ARS)

■Structure

```
s5f2
```

[11] ?

```
<b ACKC5>
```

Name	Format	Description
ACKC5	b	Acknowledge code. 1 byte. 0 Acknowledged >0 Error. Can't acknowledge 1~63 Reserved

10.2.37 S5F3 Enable/Disable Alarm Send(EAS)

```
Enable/Disable Alarm Send
S,H→E, Reply
```

■Description

Perfoms setting/resetting of enable bit of equipment alarm announcement. In the equipment, whether or not to send an alarm notification to the host is determined by this bit. Some alarms cannot be controlled by this method.

■Structure

```
s5f3w
{
   <b ALED>
   <u4 ALID>
}
```

Name	Format	Description
ALED	b	Alarm enable/disable code. 1 byte. bit8 1=Alarm enable bit8 0=Alarm disable
ALID	u4	Alarm ID

■Exceptions

Items with *ALID* length of 0 indicate setting/resetting of all alarms.

```
s5f3w
{
   <b ALED>
   <u4>
}
```

10.2.38 S5F4 Enable/Disable Alarm Acknowledge (EAA)

```
Enable/Disable Alarm Acknowledge
S,H←E
```

■Description

OK or NG reply to S5F3 Enable/Disable Alarm Send(EAS)

■Structure

```
s5f4
<b ACKC5>
```

Name	Format	Description
ACKC5	b	Acknowledge code. 2 byte. 0 Acknowledged >0 Error. Can't acknowledge 1~63 Reserved

10.2.39 S5F5 List Alarm Request (LAR)

```
List Alarm Request
S,H→E, Reply
```

■Description

Host makes a request to the equipment to send an alarm information list.

■Structure

```
s5f5w
<u4 ALID1 ... ALIDn>
```

Name	Format	Description
ALID	u4	Alarm ID

■Exceptions

Items with length 0 request all alarms.

10.2.40 S5F6 List Alarm Data (LAD)

```
List Alarm Data
M,H←E
```

■Description

Equipment current alarm status (alarm occurrence/clearing); can be multiple alarm data.

■Structure

```
s5f6
{
   {
      <b ALCD1>
      <u4 ALID1>
      <a ALTX1>
   }
   .
   .
   {
      <b ALCDm>
      <u4 ALIDm>
      <a ALTXm>
   }
}
```

Name	Format	Description
ALCD	b	Alarm code bit8 1=Alarm status occurred bit8 0=Alarm status cleared bit7~1 Alarm classification code 0 Not used 1 Human safety-related 2 Tool safety-related 3 Parameter control alarm 4 Parameter control error 5 Unrecoverable error 6 Equipment status warning 7 Warning flag 8 Data cannot be guaranteed >8 Other category 9~63 Warning
ALID	u4	Alarm ID
ALTX	a	Alarm text. Max 40 characters

■Exceptions

Lists with length m of 0 have no alarm data. Items with *ALCDi* or *ALTXi* length of 0 indicate that the corresponding alarm has no data.

10.2.41 S6F5 Multi Block Data Send Inquire (MBI)

```
Multi-block Data Send Inquire
S,H←E, Reply
```

■Description

If Discrete Data Report, S6F3 Discrete Variable Send (DVS), S6F9 Formatted Variable Send(FVS)、S6F11 Event Report Send(ERS), or S6F13 Annotated Event Report Send (AERS) require multiple blocks, this process must be performed prior to sending.

■Structure

```
s6f5W
{
   <? DATAID>
   <? DATALENGTH>
}
```

Name	Format	Description
DATAID	u4	Data ID
DATALENGTH	u4	Total No. of Bytes of send data

10.2.42 S6F6 Multi Block Grant (MBG)

Multi-block Grant
S,H→E

■Description

OK or NG reply to S6F5 Multi Block Data Send Inquire (MBI).

■Structure

```
s6f6
<b GRANT6>
```

Name	Format	Description
GRANT6	b	Send granting. 1 byte. 0 Send granted 1 Busy. Retry request 2 Unnecessary >2 Other error 3~63 Reserved

10.2.43 S6F11 Event Report Send (ERS)

Event Report Send
M,H←E, Reply

■Description

The objective of this message is for the equipment to send defined valid report groups to the host computer when an event occurs. (*CEID*)

If S6F11 is multi block, S6F5 Multi Block Data Send Inquire (MBI) and S6F6 Multi Block Grant (MBG)transactions must precede this.[12]

■Structure

```
s6f11w
{
   <u4 DATAID>
   <u4 CEID>
   {
      {
         <u4 RPTID1>
         {
            <V1>
            .
            .
            <Vp>
         }
      }
      .
      .
      {
         <u4 RPTIDa>
         {
            <V1>
            .
            .
            <Vc>
         }
      }
   }
}
```

Name	Format	Description
DATAID	u4	Data ID
CEID	u4	Collection event ID
RPTID	u4	Report ID
V	l, b, bool, a, j, i*, f*, u*	Variable data

■Exceptions

If there are no reports linked to an event, it will be a list of length 0, a=0. Lists of length 0 for report numbers indicate that there is no report linked to the given *CEID*.

10.2.44 S6F12 Event Report Acknowledge(ERA)

Event Report Acknowledge
S,H→E

■Description

Acknowledgement or error.

■Structure

```
s6f12
<b ACKC6>
```

Name	Format	Description
ACKC6	b	Acknowledge code. 1 byte. 0 Acknowledged >0 Error. Cannot acknowledge 1~63 Reserved

10.2.45 S6F15 Event Report Request (ERR)

Event Report Request
S,H←→E, Reply

■Description

This message is for the host computer to request a given series of reports from the equipment.

■Structure

```
s6f15w
<u4 CEID>
```

Name	Format	Description
CEID	u4	Collection event ID.

10.2.46 S6F16 Event Report Data (ERD)

Event Report Data
M,H←E, Reply

[12] However, since in HSMS there is no multi block message, s6f5 will not occur with this product.

■Description
The equipment sends reports linked to the given *CEID*.

■Structure
Same as the structure of S6F11 Event Report Send(ERS)

```
s6f16w
{
   <u4 DATAID>
   <u4 CEID>
   {
      {
         <u4 RPTID1>
         {
            <V1>
            .
            .
            <Vp>
         }
      }
      .
      .
      {
         <u4 RPTIDa>
         {
            <V1>
            .
            .
            <Vc>
         }
      }
   }
}
```

Name	Format	Description
DATAID	u4	Data ID
CEID	u4	Collection event ID
RPTID	u4	Report ID
V	l, b, bool, a, j, i*, f*, u*	Variable data

■Exceptions
Items of length 0 indicate that there are no reports linked to the given *CEID*.

10.2.47 S6F19 Individual Report Request (IRR)

Individual Report Request
S,H→E, Reply

■Description
The objective of this message is for the host to request defined reports from the equipment.

■Structure

```
s5f19w
<u4 RPTID>
```

Name	Format	Description
RPTID	u4	Report ID

10.2.48 S6F20 Individual Report Data (IRD)

Individual Report Data
M,H←E

■Description
The equipment sends defined variable data for a given *RPTID* to the host.

■Structure

```
s5f20
{
   <V1>
   .
   .
   <Vn>
}
```

Name	Format	Description
V	l, b, bool, a, j, i*, f*, u*	Variable data

■Exceptions
Lists of length 0 indicate that *RPTID* is undefined.

Name	Format	Description
RPTID	u4	Report ID

10.2.49 S7F1 Process Program Load Inquire (PPI)

Process Program Load Inquire
S,H←→E, Reply

■Description
This message is used to start loading or unloading of process programs. It is used preceding S7F3 Process Program Send(PPS), S7F4 Process Program Acknowledge(PPA) or S7F23 Formatted Process Program Send(EPS),S7F24 Formatted Process Program Acknowledge(FPA) 、S7F31 Verification Request Send(VRS) and S7F32 Verification Request Acknowledge(VRA).

■Structure

```
s7f1w
{
   <a PPID>
   <LENGTH>
}
```

Name	Format	Description
PPID	a	Process program ID. Max 80 bytes. *PPID* format depends on the host. When used by the equip., *PPID* is handled as a binary pattern. If there is no device to display sent code, it will be in hexadecimal format.
LENGTH	i*, u*	Service program or process program byte length.

10.2.50 S7F2 Process Program Load Grant (PPG)

Process Program Load Grant
S,H←→E

■Description
This message gives load grants for process programs.

■Structure

```
s7f2
<b PPGNT>
```

Name	Format	Description
PPGNT	b	Process program grant status. 1

		byte. 0 OK。 1 Already have 2 No space 3 Invalid *PPID* 4 Busy. Retry request 5 Not granted >5 Other error 6~63 Reserved

10.2.51 S7F3 Process Program Send (PPS)

Process Program Send
M,H←→E, Reply

■Description
Process Program sending. If S7F3 is multi block, S7F1 Process Program Load Inquire (PPI) and S7F2 Process Program Load Grant(PPG) transactions must precede.

■Structure

```
s7f3w
{
   <a PPID>
   <PPBODY>
}
```

Name	Format	Description
PPID	a	Process program ID. Max 80 bytes. *PPID* format depends on the host. When used by the equip., *PPID* is handled as a binary pattern. If there is no local device to display the send code, it will be in hexadecimal format.
PPBODY	b, a, i*, u*	Process program main body. Statement of operations for equipment to perform to process received materials, in equipment's own language.

10.2.52 S7F4 Process Program Acknowledge (PPA)

Process Program Acknowledge
S,H←→E

■Description
OK or NG reply to S7F3 Process Program Send (PPS).

■Structure

```
s7f4
<b ACKC7>
```

Name	Format	Description
ACKC7	b	Acknowledgement code. 1 byte. 0 Granted 1 Not granted 2 Length error 3 Array overflow 4 Undefined *PPID* 5 Mode error >5 Other error 6~63 Reserved

10.2.53 S7F5 Process Program Request (PPR)

Process Program Request
S,H←→E, Reply

■Description
Requests sending of process programs.

■Structure

```
s7f5w
<a PPID>
```

Name	Format	Description
PPID	a	Process program ID. Max 80 bytes . *PPID* format depends on host. When used by the equip., *PPID* is handled as a binary pattern. If there is no local device to display the send code, it will be in hexadecimal format.

10.2.54 S7F6 Process Program Data (PPD)

Process Program Data
M,H←→E

■Description
Sending of process program

■Structure

```
s7f6
{
   <a PPID>
   <PPBODY>
}
```

Name	Format	Description
PPID	a	Process program ID. Max 80 bytes. *PPID* format depends on the host. When used by the equip., *PPID* is handled as a binary pattern. If there is no local device to display the send code, it will be in hexadecimal format.
PPBODY	b, a, i*, u*	Process program main body. Statement of operations for equipment to perform to process received materials, in equipment's own language.

■Exceptions
Lists of length 0 indicate that the request was denied.

■Note
By setting the header R bit to 1, it is possible to send process programs made on the equipment side to the host. By doing so, even if equipment programs for the equipment cannot be created on the host side, the equipment can be used.

10.2.55 S7F17 Delete Process Program Send (DPS)

Delete Process Program Send
S,H→E, Reply

■Description
Host requests equipment process program deletion.

■Structure

```
s7f17w
{
```

```
   <a PPID1>
   .
   .
   <a PPIDn>
}
```

Name	Format	Description
PPID	a	Process program ID. Max 80 bytes. *PPID* format depends on the host. When used by the equip.., *PPID* is handled as a binary pattern. If there is no local device to display the send code, it will be in hexadecimal format.

■Exceptions
Lists of length 0 with n=o will delete all process programs.

10.2.56 S7F18 Delete Process Program Acknowledge (DPA)

Delete Process Program Acknowledge
S,H←E

■Description
OK or NG reply to S7F17 Delete Process Program Send(DPS)

■Structure

```
s7f18
<b ACKC7>
```

Name	Format	Description
ACKC7	b	Acknowledgement code. 1 byte. 0 Granted 1 Not granted 2 Length error 3 Array overflow 4 *PPID* Undefined 5 Mode error >5 Other error 6~63 Reserved
PPID	b, a	Process program ID. Max 80 bytes. *PPID* format depends on the host. When used by the equip.., *PPID* is handled as a binary pattern. If there is no local device to display the send code, it will be in hexadecimal format.

10.2.57 S7F19 Current EPPD Request (RER)

Current EPPD Request
S,H→E, Reply

■Description
This message is used to request the current equipment process program directory (EPPD). This is a list of all *PPID* of process programs in the equipment's memory.

■Structure
Header only

```
s7f19w
```

Name	Format	Description
PPID	a	Process program ID. Max 80 bytes. *PPID* format depends on the host. When used by the equip.., *PPID* is handled as a binary pattern. If there is no local device to display the send code, it will be in hexadecimal format.

10.2.58 S7F20 Current EPPD Data (RED)

Current EPPD Data
M,H←E

■Description
This message is used to communicate the current EPPD.

■Structure

```
s7f20
{
   <a PPID1>
   .
   .
   <a PPIDn>
}
```

Name	Format	Description
PPID	a	Process program ID. Max 80 bytes. *PPID* format depends on the host. When used by the equip.., *PPID* is handled as a binary pattern. If there is no local device to display the send code, it will be in hexadecimal format.

10.2.59 S7F23 Formatted Process Program Send (EPS)

Formatted Process Program Send
M,H←→E, Reply

■Description
This message is used to communicate formatted process programs between the equipment and host.*MDLN and SOFTREV* values are obtained from S7F22 Equipment Process Capacity Data (PCD) used in creating the process program. If S7F23 is multi block, S7F1 Process Program Load Inquire(PPI) and S7F2 Process Program Load Grant (PPG) transactions must precede this.

■Structure

```
s7f23w
{
   <a PPID>
   <a MDLN>
   <a SOFTREV>
   {
      {
         <CCODE>
         {
            <PPARM1>
            .
            .
            <PPARMp>
         }
      }
         <CCODEc>
         {
            <PPARM1>
            .
            .
            <PPARMq>
         }
      }
   }
}
```

Name	Format	Description
PPID	a	Process program ID. Max 80 bytes. *PPID* format depends on the host. When used by the equip.., *PPID* is handled as a binary pattern. If there is no local device to display the send code, it will be in hexadecimal format.
MDLN	a	Equipment model. Max 6 bytes
SOFTREV	a	Software revision code. Max6 bytes.
CCODE	i2, u2	Command code. Each command code corresponds to an individual process operation which the device can perform.
PPARM	bool, a, i*, f*, u*	Process parameter. Parameter which gives information needed to complete process command.May be numerical or True/False SECS data item. May be single or multiple value or character string.

10.2.60 S7F24 Formatted Process Program Acknowledge (FPA)

Formatted Process Program Acknowledge
S,H←→E

■Description

Reply for receipt of formatted process program and receipt of process program via inter-printer. “Acknowledged” reply from inter-printer only means that message was understood. Validity of process program contents is communicated via different transactions: S7F27 Process Program Verification Send (PVS) and S7F28 Process Program Verification Acknowledge(PVA).

■Structure

s7f24
<b ACKC7>

Name	Format	Description
ACKC7	b	Acknowledgement code. 1 byte 0 Granted 1 Not granted 2 Length error 3 Array overflow 4 *PPID* undefined 5 Mode error >5 Other error 6~63 Reserved
PPID	a	Process program ID. Max 80 bytes. *PPID* format depends on the host. When used by the equip.., *PPID* is handled as a binary pattern. If there is no local device to display the send code, it will be in hexadecimal format.

10.2.61 S7F25 Formattted Process Program Request (FPR)

Formatted Process Program Request
S,H←→E, Reply

■Description

This message is used for the host or equipment to request sending of process programs.

■Structure

s7f25w
<a *PPID*>

Name	Format	Description
PPID	a	Process program ID. Max 80 bytes. *PPID* format depends on the host. When used by the equip.., *PPID* is handled as a binary pattern. If there is no local device to display the send code, it will be in hexadecimal format.

10.2.62 S7F26 Formatted Process Program Data (FPD)

Formatted Process Program Data
M,H←→E

■Description

This message is used to send process programs as replies to *PPID* requests. *MDLN* and *SOFTREV* values are obtained from S7F22 Equipment Process Capacity Data(PCD) used in creating the process program.

■Structure

```
s7f26
{
   <a PPID>
   <a MDLN>
   <a SOFTREV>
   {
     {
       <CCODE>
       {
         <PPARM1>
         .
         .
         <PPARMp>
       }
     }
       <CCODEc>
       {
         <PPARM1>
         .
         .
         <PPARMq>
       }
     }
   }
}
```

Name	Format	Description
PPID	a	Process program ID. Max 80 bytes. *PPID* format depends on the host. When used by the equip.., *PPID* is handled as a binary pattern. If there is no local device to display the send code, it will be in hexadecimal format.
MDLN	a	Equip. model. Max 6 bytes
SOFTREV	a	Software revision code. Max 6 bytes.
CCODE	i2, u2	Command code. Each command code corresponds to an individual process operation which the device can perform.
PPARM	bool, a, i*, f*, u*	Process parameter. Parameter which gives information needed to

		complete process command.May be numerical or True/False SECS data item. May be single or multiple value or character string.

■Exceptions
Lists of length 0, where c=0, indicate that the request was denied.

10.2.63 S7F27 Process Program Verification Send (PVS)

Process Program Verification Send
M,H←E, Reply

■Description
This message notifies the host that the equipment has received and checked the process program. The check results are specified via the error list. There may be an empty list (list of length 0 where n=0) or a 0 value. Single element lists having *ACKC7A* indicate that no error was found during processing. The equipment can report as many errors as it deems suitable. No matter what the format, when the equipment has received process programs of any format, S7F23 Formatted Process Program Send(EPS) or S7F26 Formatted Program Data (FPD)or S7F31 Verification Request Send(VRS), it has the duty to send a copy of this message to the host. If S7F27 is multi block, the S7F29 Process Program Verification Inquire(PVI) and S7F30 Process Program Verification Grant(PVG) transactions must precede it.

■Structure

```
s7f27w
{
   <a PPID>
   {
      {
         <ACKC7A>
         <SEQNUM>
         <a ERRW7>
      }
      .
      .
      {
         <ACKC7An>
         <SEQNUMn>
         <a ERRW7n>
      }
   }
}
```

Name	Format	Description
PPID	b, a	Process program ID. Max 80 bytes. *PPID* format depends on the host. When used by the equip.., *PPID* is handled as a binary pattern. If there is no local device to display the send code, it will be in hexadecimal format.
ACKC7A	i1, u1	Acknowledge code. 1 byte 0 Acknowledged 1 *MDLN* inconsistent 2 *SOFTREV* inconsistent 3 Invalid *CCODE* 4 Invalid PPARM *val.* 5 Other error (shown via *ERRW7)* 6~63 Reserved
SEQNUM	i*, u*	Command number indicating by number the position of the command in the process command list. In the first command in the process program, *SEQNUM is* 1.
ERRW7	a	Character string showing errors found in the process program.
MDLN	a	Equip. model. Max 6 bytes
SOFTREV	a	Software revision number. Max 6 bytes
CCODE	i2, u2	Command code. Each command code corresponds to an individual process operation which the device can perform.
PPARM	bool, a, i*, f*, u*	Process parameter. Parameter which gives information needed to complete process command.May be numerical or True/False SECS data item. May be single or multiple value or character string.

10.2.64 S7F28 Process Program Verification Acknowledge (PVA)

Process Program Verification Acknowledge
S,H→E

■Description
Reply from host acknowledging receipt of S7F27 Process Program Verification Send(PVS) from equipment.

■Structure
Header only

```
s7f28
```

10.2.65 S7F29 Process Program Verification Inquire (PVI)

Process Program Verification Inquire
S,H←E, Reply

■Description
This message is used by the equipment to ask the host for permission to send multi block S7F27 Process Program Verification Send(PVS).

■Structure

```
s7f29w
<LENGTH>
```

Name	Format	Description
LENGTH	i*, u*	Service program and process program byte length.

10.2.66 S7F30 Process Program Verification Grant (PVG)

Process Program Verification Grant
S,H→E

■Description
Reply from host to equipment regarding S7F29 Program

Process Verification Inquire(PVI).

■Structure

s7f30
<b PPGNT>

Name	Format	Description
PPGNT	b	Process program grant status. 1 byte. 0 OK 1 Already have 2 No space 3 Invalid *PPID*。 4 Busy. Retry request 5 Not granted >5 Other error 6~63 Reserved
PPID	b, a	Process program ID. Max 80 bytes. *PPID* format depends on the host. When used by the equip.., *PPID* is handled as a binary pattern. If there is no local device to display the send code, it will be in hexadecimal format.

10.2.67 S9F1 Unrecognized Device ID (UDN)

Unrecognized Device ID
S,H←E

■Description
The device ID in the message block header is undefined for that node.

■Structure

s9f1
<b MHEAD>

Name	Format	Description
MHEAD	b	Message header that errored

10.2.68 S9F3 Unrecognized Stream Type (USN)

Unrecognized Stream Type
S,H←E

■Description
The stream type in the message block header is undefined for that equipment.

■Structure

s9f3
<b MHEAD>

Name	Format	Description
MHEAD	b	Message header that errored

10.2.69 S9F5 Unrecognized Function Type (UFN)

Unrecognized Function Type
S,H←E

■Description
The function type in the message ID is undefined for that equipment.

■Structure

s9f5
<b MHEAD>

Name	Format	Description
MHEAD	b	Message header that errored

10.2.70 S9F7 Illegal Data (IDN)

Illegal Data
S,H←E

■Description
The stream and function were understood but the data format could not be interpreted.

■Structure

s9f7
<b MHEAD>

Name	Format	Description
MHEAD	b	Message header that errored

10.2.71 S9F9 Transaction Timer Timeout (TIN)

Transaction Timer Timeout
S,H←E

■Description
Indicates that Transaction (T3) timer has timed out and that the transaction in progress was forced to terminate. The host determines what response to this error is necessary in order to maintain proper operational status of the system.

■Structure

s9f9
<b SHEAD>

Name	Format	Description
SHEAD	b	Header of message relating to transaction time.

10.2.72 S9F11 Data Too Long (DLN)

Data Too Long
S,H←E

■Description
Indicates that data of length too long to process was sent to equipment.

■Structure

s9f11
<b MHEAD>

Name	Format	Description
MHEAD	b	Message header that errored.

10.2.73 S9F13 Conversation Timeout (CTN)

Conversation Timeout

S,H←E

■Description
Data was expected to be received but was not within the appropriate timeframe. Resources are cleared.

■Structure

```
s9f13
{
   <a MEXP>
   <EDID>
}
```

Name	Format	Description
MEXP	a	SxxFyy to be received.
EDID	b, a, i*, u*	Data ID to be received. One of the following 3 is possible: MEXP EDID EDID S2F3 <SPID> a[6] S3F13 <PTN> b[1] S7F3 <PPID> a[16], b[16]

10.2.74 S10F1 Terminal Request (TRN)

Terminal Request
S,H←E, Reply

■Description
Text message from terminal to host.

■Structure

```
s10f1w
{
   <b TID>
   <a TEXT>
}
```

Name	Format	Description
TID	b	Terminal no. 1 byte. 0 Single or main terminal >0 Added terminal on same equipment
TEXT	a, a2	One-line character

10.2.75 S10F2 Terminal Request Acknowledge (TRA)

Terminal Request Acknowledge
S,H→E

■Description
OK or NG reply to S10F1 Terminal Request(TRN)

■Structure

```
s10f2
<b ACKC10>
```

Name	Format	Description
ACKC10	b	Acknowledgement code. 1 byte. 0 Display acknowledged 1 Message not displayed 2 Cannot use terminal 3~63 Reserved

10.2.76 S10F3 Terminal Display, Single Block (VTN)

Terminal Display,Single
S,H→E, Reply

■Description
Data to be displayed

■Structure

```
s10f3w
{
   <b TID>
   <a TEXT>
}
```

Name	Format	Description
TID	b	Terminal no. 1 byte. 0 Single or main terminal >0 Added terminal on same equipment
TEXT	a, a2*	One-line character

10.2.77 S10F4 Terminal Display, Single Block Acknowledge (VTA)

Terminal Display,Single Acknowledge
S,H←E

■Description
OK or NG reply to S10F3 Terminal Display, Single Block (VTN)

■Structure

```
s10f4
<b ACKC10>
```

Name	Format	Description
ACKC10	b	Acknowledgement code. 1 byte. 0 Display acknowledged 1 Message not displayed 2 Cannot use terminal 3~63 Reserved

10.2.78 S10F5 Terminal Display, Multi Block (VTN)

Terminal Display,Multi-block
M,H→E, Reply

■Description
Data to be displayed

■Structure

```
s10f5w
{
   <b TID>
   {
      <a TEXT1>
      .
      .
      <a TEXTn>
   }
}
```

Name	Format	Description
TID	b	Terminal no. 1 byte. 0 Single or main terminal >0 Added terminal on same equipment
TEXT	a, a2	One-line character

10.2.79 S10F6 Terminal Display, Multi Block Acknowledge (VMA)

Terminal Display,Multi-block Acknowledge
S,H←E

■Description

OK or NG reply to S10F5 Terminal Display, Multi Block (VTN)

■Structure

s10f6
<b ACKC10>

Name	Format	Description
ACKC10	b	Acknowledgement code. 1 byte. 0 Display acknowledged 1 Message not displayed 2 Terminal cannot be used 3~63 Reserved

10.2.80 S10F7 Multi Block Not Allowed (MNN)

Multi-block Not Allowed
S,H←E

■Description

Error message from terminal that cannot process multi block message in S10F5 Terminal Display, Multi Block(VTN)

■Structure

s10f7
<b *TID*>

Name	Format	Description
TID	b	Terminal Number. 1 byte. 0 Single or main terminal >0 Added terminal at same equip.

10.2.81 S14F1 Get Attribute Request (GAR)

GetAttr Request
S,H←→E, Reply

■Description

This message is used to request a particular attributes set of one or more objects. This is composed of the following items: Specifiers for use by owner of target object (object of interest), target object type, list of target object classifiers, filters (restriction-related list, in other words, limiting to only those objects , among the target objects of interest, which fulfuill all restrictions included in the filter) and particular attributes for which entry of values is necessary.

Object specifiers select the owner of the target object. They also sequences related to hierarchically-structured objects. Each element of the object specifier recognizes the object instance which is in the highest position of the object instance next in the sequence. The final object instance in the sequence is in a hierarchical structure with the target object. The target object type indicates the type of the target object. The object classifier list indicates specific instances of the object type that is the subject of interest. In the target type, if the object classifier does not match that of all the other object types, and if the classifier list is not empty, this may be omitted.

The object filter is a nominal list of restrictions (supplying conditions to apply to object instances of interest). Each restricted object of interest is an object which meets all specified restrictions.

Attribute-related quantifiers are logical, binary-related *ATTRRELNi* owned by the restriction value *ATTRDATAi* specified with respect to the attributes corresponding to each instance of the desired object type. Objects restricted by this filter have attribute values, *Vi,* for which the *"ATTRDATAi*、*ATTRRLENi*、*Vi* " statement is True. If *ATTRRLENi* is omitted, it will be considered to have an equivalent relationship.

In the case of a character string attribute value *ATTRDATAi, the question mark* " ? " and asterisk " * " symbols are used as wild-card characters for applying filtering to specific object types. The " ? " symbol can be used to indicate one nominal character for a nominal attribute value or an important attribute value in ASCII format. It can even be used repeatedly. The asterisk symbol " * " can be used in the same way as the question mark "?", to indicate variable-length strings, including character strings of length 0. String *X " indicates a variable-length string ending in " X " . String " X* " indicates a variable-length string beginning with " X " . String " * " indicates a string with length other than 0. In text character comparison, there is no distinction between upper- and lower-case characters.

No particular extra equipment is necessary in order to support the wild-card characters or the general attribute filters.

■Structure

```
s14f1w
{
  <a OBJSPEC>
  <OBJTYPE>
  {
    <OBJID1>
    .
    .
    <OBJIDi>
  }
  {
    {
      <ATTRID1>
      <ATTRDATA1>
      <u1 ATTRRELN1>
    }
    .
    .
    {
      <ATTRIDq>
      <ATTRDATAq>
      <u1 ATTRRELNq>
    }
  }
  {
    <ATTRID1>
    .
    .
    <ATTRIDa>
  }
}
```

Name	Format	Description
OBJSPEC	a	Text string used to indicate a specific object instance having an

		internal format. This string is composed of a series of formatted sub-strings, each of which identifies object type and classifier. Sub-string format is composed of the following 4 fields. Object type Colon “ : ” Object classifier Inequality sign “ > ” The colon “ : ” is used at the end of the object type. The inequality sign“ > ” is used at the end of the classifier. Object type is also determined by other methods so it may be omitted. The final “ > ” is optional.
OBJTYPE	a, u*	Object group or class classifier. It must be possible to use the same attribute set for all objects of a single type
OBJID	a, u*	Classifier for objects
ATTRID	a, u*	Attribute classifier for specific types of objects
ATTRDATA	l, b, bool, a, i*, f*, u*	Holds specific attribute value of specific objects.
ATTRRELN	u1	Specifies the relationship between specific restriction values and object instance attribute values (values of interest) 0 Restriction value is equivalent to value of interest 1 Restriction value is not equivalent to value of interest 2 Restriction value is less than value of interst 3 Restriction value is less than value of interest but is equivalent 4 Restriction value is more than value of interest 5 Restriction value is more than value of interest but is equivalent 6 Restriction value includes value of interest (part of set) 7 Restriction value does not include value of interst (not part of set). >7 Reserved
V	l, b, bool, a, j, i*, f*, u*	Variable data

■Exceptions

If *OBJSPEC* is an item of length 0, an object specifier is not prepared. When i=0, only a filter is added. When q=0, a filter is not specified. When both i and q are 0, information on all instances of the object is requested. When a=0, all attributes are requested.

10.2.82 S14F2 Get Attribute Data(GAD)

GetAttr Data
M,H←→E

■Description

This message is used to transmit a requested set of attributes of a particular object. The order of attributes is retained from the primary message.

■Structure

```
s14f2
{
  {
    {
      <OBJID1>
      {
        {
          <ATTRID1>
          <ATTRDATA1>
        }
        .
        .
        {
          <ATTRIDa>
          <ATTRDATAa>
        }
      }
    }
    .
    .
    {
      <OBJIDn>
      {
        {
          <ATTRID1>
          <ATTRDATA1>
        }
        .
        .
        {
          <ATTRIDb>
          <ATTRDATAb>
        }
      }
    }
  }
  {
    <u1 OBJACK>
    {
      {
        <ERRCODE1>
        <a ERRTEXT1>
      }
      .
      .
      {
        <ERRCODEp>
        <a ERRTEXTp>
      }
    }
  }
}
```

Name	Format	Description
OBJID	a, u*	Classifier for objects
ATTRID	a, u*	Attribute classifier for specific types of objects
ATTRDATA	l, b, bool, a, i*, f*, u*	Holds specific attribute values for specific objects
OBJACK	u1	Acknowledgement code 0 Requested data command executed 1 Error

		>1 Reserved
ERRCODE	u*	Error classification code 0 No error 1 Unknown object in object specifier 2 Unknown target object type 3 Unknown object instance 4 Unknown attribute name 5 Read-only attribute. Access denied. 6 Unknown object type 7 Disabled object val. 8 Syntax error 9 Validation error 10 Verification error 11 Object specifier in use 12 Parameter specified incorrectly 13 Not all parameters to be specified are specified 14 Requested option not supported 15 In use 16 Processing preparation not ready 17 Invalid command in current status 18 No changed materials 19 Materials partially processed 20 Materials all processed 21 Error relating to recipe specification 22 Failed during process 23 Failed, not during process。 24 Failed due to insufficient material 25 Job abort 26 Job stop 27 Job clear 28 Recipe selected cannot be changed 29 Undefined event 30 Duplicate report ID 31 Undefined data report 32 Data report unlinked 33 Undefined trace report 34 Duplicate trace ID 35 Too many data reports 36 Sample period out of range 37 Group size too large 38 Recovery action currently invalid 39 Other recovery preventing requested recovery currently underway 40 No active recovery action 41 Exceptional recovery failure 42 Exceptional recovery abort 43 Invalid table element 44 Undefined table element 45 Previously set item cannot be deleted 46 Invalid token 47 Invalid parameter 48~63 Reserved
ERRTEXT	a	Character string showing *ERRCODE*. Max 80 characters

■Exceptions

If *OBJSPEC* is an object of length 0, an object specifier is not prepared. When n=o, there is no object compliant with a particular filter. When p=0, no error was detected.

Name	Format	Description
OBJSPEC	a	Text string used to indicate a specific object instance having an internal format. This string is composed of a series of formatted sub-strings, each of which identifies object type and classifier. Sub-string format is composed of the following 4 fields. Object type Colon “ : ” Object classifier Inequality sign “ > ” The colon “ : ” is used at the end of the object type. The inequality sign “ > ” is used at the end of the classifier. Object type is also determined by other methods so it may be omitted. The final “ > ” is optional.

10.2.83 S15F1 Recipe Management Multi Block Inquire

Recipe Management Multi-block Inquire
S,H←→E, Reply

■Description

This message requests authorization to send a multi block message, based on the maximum message length for the full multi block message.

■Structure

```
s15f1w
{
   <u4 DATAID>
   <a RCPSPEC>
   <RMDATASIZE>
}
```

Name	Format	Description
DATAID	u4	Data ID
RCPSPEC	a	Recipe specifier. Recipe object specifier.
RMDATASIZE	u*	Indicates maximum length for multi block message in number of bytes; used to cause the receiver determine whether an expected message exceeds receiver capacity

■Exceptions

If *RCPSPEC* is a character string of length 0, no multi block message subject to send authorization is included in the recipe.

10.2.84 S15F2 Recipe Management Multi Block Grant

Recipe Management Multi-block Grant
S,H←→E

■Description

This message grants or denies multi block message sending.

■Structure

```
s15f2
<b RMGRNT>
```

Name	Format	Description
RMGRNT	b	Grant code. Used to grant or denya request. 1 byte. 0 Granted 1 Cannot currently grant. Retry 2 No space 3 Request on standby 4~64 Reserved

10.2.85 S15F21 Recipe Action Request

```
Recipe Action Request
M,H←→E, Reply
```

■Description

This message is used to verify a specific action request to be executed in one or more recipes in the name space.

■Structure

```
s15f21w
{
  <u4 DATAID>
  <u1 RCPCMD>
  <a RMNSSPEC>
  <u* OPID>
  <a AGENT>
  {
    <a RCPID1>
    .
    .
    <a RCPIDn>
  }
}
```

Name	Format	Description
DATAID	u4	Data ID
RCPCMD	u1	Indicates action to be executed in the recipe 0~4 Reserved 5 Delete 6~7 Reserved 8 No save 9 Save 10 Validate 11 Link 12 Clear link 13 Authenticate 14 Cancel verification 15 Download 16 Upload 17~63 Reserved
RMNSSPEC	a	Recipe name space object specifier
OPID	u*	Operation ID. Unique integer created by operation requestor; used when multiple completion verifications occur.
AGENT	a	
RCPID	a	Recipe classifier. Formatted text is in accordance with requirements of *OBJSPEC*.
OBJSPEC	a	Text string used to indicate a specific object instance having an internal format. This string is composed of a series of formatted sub-strings, each of which identifies object type and classifier. Sub-string format is composed of the following 4 fields. Object type Colon “:” Object classifier Inequality sign “>” The colon “:” is used at the end of the object type. The inequality sign “>” is used at the end of the classifier. Object type is also determined by other methods so it may be omitted. The final “>” is optional.

■Exceptions

Except for verification, cancel verification, download and upload requests, *AGENT* will be a character string of length *0*.

10.2.86 S15F22 Recipe Action Acknowledge

```
Recipe Action Acknowledge
M,H←→E
```

■Description

This message is used to verify a request to create a new recipe.

■Structure

```
s15f22
{
  <a AGENT>
  <u4 LINKID>
  <u1 RCPCMD>
  {
    <u1 RMACK>
    {
      {
        <ERRCODE1>
        <a ERRTEXT1>
      }
      .
      .
      {
        <ERRCODEp>
        <a ERRTEXTp>
      }
    }
  }
}
```

Name	Format	Description
AGENT	a	
LINKID	u4	Used to link operation execution requests and completion messages. *LINKID* is set to the *MOPID* value included in the initial request.In exceptional cases, this is a completion message to be sent at the end, and in such cases it is set to 0.
RCPCMD	u1	Shows action to be executed in recipe. 0~4 Reserved 5 Delete 6~7 Reserved 8 No save 9 Save

		10 Validate 11 Link 12 Clear link 13 Authenticate 14 Cancel verification 15 Download 16 Upload 17~63 Reserved
RMACK	u1	Communicates whether requested action was successfully completed, was denied, terminated due to error, or completed with notification to the requestor. 0 Completed successfully 1 Cannot execute corresponding action 2 Terminated due to error 3 Corresponding action will be completed and notification send 4 Corresponding action does not require existence
ERRCODE	u*	Error classification code 0 No error 1 Unkown object in object specifier 2 Unknown target object type 3 Unknown object instance 4 Unknown attribute name 5 Read-only attribute. Access denied. 6 Unknown object type 7 Invalid attribute val. 8 Syntax error 9 Validation error 10 Verification error 11 Object specifier in use 12 Parameter not correctly specified 13 Not all parameters to be specified are specified. 14 Requested option not supported 15 In use 16 Processing preparation not ready 17 Invalid command in current status 18 No changed material 19 Material partially processed 20 Material all processed 21 Error relating to recipe specification 22 Failed during process 23 Failed, not during process。 24 Failed due to insufficient material 25 Job abort 26 Job stop 27 Job clear 28 Recipe selected cannot be changed 29 Undefined event 30 Duplicate report ID 31 Undefined data report 32 Data report unlinked 33 Undefined trace report 34 Duplicate trace ID 35 Too many data reports 36 Sample period out of range 37 Group size too large 38 Recovery action currently invalid 39 Other recovery preventing requested recovery currently underway 40 No active recovery action 41 Exceptional recovery failure 42 Exceptional recovery abort 43 Invalid table element 44 Undefined table element 45 Previously set item cannot be deleted 46 Invalid token 47 Invalid parameter 48~63 Reserved
ERRTEXT	a	Character string showing error indicated in *ERRCODE*. Max 80 characters

■Exceptions

Only when all requested actions are comleted is *LINKID* 0. Only when *RMACK* indicates no errors is p=0.

10.2.87 S15F27 Recipe Download Request

Recipe Download Request
M,H→E, Reply

■Description

This message is used to send recipes to a recipe executor. In this case, the following transactions precede: S15F1 Recipe Management Multi Block Inquire、S15F2 Recipe Management Multi Block Grant

■Structure

```
s15f27w
{
  <u4 DATAID>
  <bool RCPOWCODE>
  <a RCPSPEC>
  {
    {
      <a RCPATTRID1>
      <RCPATTRDATA1>
    }
    .
    .
    {
      <a RCPATTRIDm>
      <RCPATTRDATAm>
    }
  }
  <RCBODY>
}
```

Name	Format	Description
DATAID	u4	Data ID
RCPOWCODE	bool	Indicates whether previously existing recipes will be over-written (=TRUE) or not (=FALSE) when downloading occurs.
RCPSPEC	a	Recipe specifier. Recipe object specifier
RCPATTRID	a	Non-classifier attribute name (classifier)
RCPATTRDATA	l, b, bool, a, i*, f*, u*	Recipe attribute contents (value)
RCBODY	b, a, i*, u*	Recipe main body

10.2.88 S15F28 Recipe Download Acknowledge

Recipe Download Acknowledge
M,H←E

■Description

This message is used to verify that the recipe executor sent a recipe. If recipe validation is successful, the results are returned to the center. If an object-format derivative recope is created during validation, *RCPID* will include a classifier for this derivative recipe.

■Structure

```
s15f28
{
  <a RCPID>
  {
    {
      <a RCPATTRID1>
      <RCPATTRDATA1>
    }
    .
    .
    {
      <ERRCODE1>
      <a ERRTEXT1>
    }
    .
    .
    {
      <ERRCODEp>
      <a ERRTEXTp>
    }
  }
}
```

Name	Form-at	Description
RCPID	a	Recipe classifier. Formatted text is in accordance with requirements of *OBJSPEC.*
OBJSPEC	a	Text string used to indicate a specific object instance having an internal format. This string is composed of a series of formatted sub-strings, each of which identifies object type and classifier. Sub-string format is composed of the following 4 fields. Object type Colon “ : ” Object classifier Inequality sign “ > ” The colon “ : ” is used at the end of the object type. The inequality sign“ > ” is used at the end of the classifier. Object type is also determined by other methods so it may be omitted. The final “ > ” is optional.
RCPATTRID	a	Non-classifier attribute name (classifier)
RCPATTRDATA	l, b, bool, a, i*, f*, u*	Recipe attribute contents (value)
RMACK	u1	Communicates whether requested action was successfully completed, was denied, terminated due to error, or completed with notification to the requestor. 0 Completed successfully 1 Cannot execute corresponding action 2 Terminated due to error 3 Corresponding action will be completed and notification send 4 Corresponding action does not require existence
ERRCODE	u*	Error classification code 0 No error 1 Unkown object in object specifier 2 Unknown target object type 3 Unknown object instance 4 Unknown attribute name 5 Read-only attribute. Access denied. 6 Unknown object type 7 Invalid attribute val. 8 Syntax error 9 Validation error 10 Verification error 11 Object specifier in use 12 Parameter not correctly specified 13 Not all parameters to be specified are specified. 14 Requested option not supported 15 In use 16 Processing preparation not ready 17 Invalid command in current status 18 No changed material 19 Material partially processed 20 Material all processed 21 Error relating to recipe specification 22 Failed during process 23 Failed, not during process。 24 Failed due to insufficient material 25 Job abort 26 Job stop 27 Job clear 28 Recipe selected cannot be changed 29 Undefined event 30 Duplicate report ID 31 Undefined data report 32 Data report

		unlinked 33 Undefined trace report 34 Duplicate trace ID 35 Too many data reports 36 Sample period out of range 37 Group size too large 38 Recovery action currently invalid 39 Other recovery preventing requested recovery currently underway 40 No active recovery action 41 Exceptional recovery failure 42 Exceptional recovery abort 43 Invalid table element 44 Undefined table element 45 Previously set item cannot be deleted 46 Invalid token 47 Invalid parameter 48~63 Reserved
ERRTEXT	a	Character string stating error shown in *ERRCODE*. Max 80 characters

■Exceptions

In the case of items of length 0, object-format derivative recipes cannot be created. Only if the recipe was not verified or failed verification will n=0. Only when *RMACK* indicates no errors will P=0.

10.2.89 S15F29 Recipe Verify Request

Recipe Verify Request
M,H→E, Reply

■Description

This message is used for the recipe executor to request recipe verification of a single or multiple recipes. In the case of multi block, the S15F1 Recipe Management Multi Block Inquire and S15F2 Recipe Managemenet Multi Block Grant transactions precede this. The operation classifier *OPID* is used when multiple verification requests are not yet processed, and if no further verifications are requested until all verification requests received by that time by the recipe executor are completed, it will be 0. In any other case, *OPID* will be unique for each requestor. *RESPEC* is a recipe executor object specifier.

■Structure

```
s15f29w
{
  <u4 DATAID>
  <OPID>
  <a RESPEC>
  {
    <a RCPID1>
    .
    .
    <a RCPIDm>
  }
}
```

Name	Format	Description
DATAID	u4	Data ID
OPID	u*	Operation ID. Unique integer created by operation requestor; used when multiple completion verifications occur.
RESPEC	a	Recipe executor object specifier
RCPID	a	Recipe classifier. Formatted text is in accordance with requirements of *OBJSPEC*.
OBJSPEC	a	Text string used to indicate a specific object instance having an internal format. This string is composed of a series of formatted sub-strings, each of which identifies object type and classifier. Sub-string format is composed of the following 4 fields. Object type Colon “ : ” Object classifier Inequality sign “ > ” The colon “ : ” is used at the end of the object type. The inequality sign “ > ” is used at the end of the classifier. Object type is also determined by other methods so it may be omitted. The final “ > ” is optional.

■Exceptions

If *RESPEC* is an item of length 0, the target is the receiver of the message.

10.2.90 S15F30 Recipe Verify Acknowledge

Recipe Verify Acknowledge
M,H←E

■Description

This message is used to verify single or multiple recipe verification requests. If one recipe verification was requested and that verification was successful, the result will be sent to the senter using this message. If an object-format derivative recipe was created during verification, the classifier of that derivative recipe will be included in RCPID. If the verification of multiple recipes was requested, *LINKID* will be other than 0.

■Structure

```
s15f30
{
  <OPID>
  <u4 LINKID>
  <A RCPID>
  {
    {
      <a RCPATTRID1>
      <RCPATTRDATA1>
    }
    .
    .
    {
      <a RCPATTRIDn>
      <RCPATTRDATAn>
    }
  }
```

```
  {
    <u1 RMACK>
    {
      {
        <ERRCODE1>
        <a ERRTEXT1>
      }
      .
      .
      {
        <ERRCODEp>
        <a ERRTEXTp>
      }
    }
  }
}
```

Name	Form-at	Description
OPID	u*	Operation ID. Unique integer created by operation requestor; used when multiple completion verifications occur.
LINKID	u4	Used to link operation execution requests and completion messages. *LINKID* is set to the *MOPID* value included in the initial request.In exceptional cases, this is a completion message to be sent at the end, and in such cases it is set to 0.
RCPID	a	Recipe classifier. Formatted text is in accordance with requirements of *OBJSPEC*.
RCPATTRID	a	Non-classifier attribute name (classifier)
RCPATTRDATA	l, b, bool, a, i*, f*, u*	Recipe attribute contents (value)
RMACK	u1	Communicates whether requested action was successfully completed, was denied, terminated due to error, or completed with notification to the requestor. 0 Completed successfully 1 Cannot execute corresponding action 2 Terminated due to error 3 Corresponding action will be completed and notification sent 4 Corresponding action does not require existence
ERRCODE	u*	Error classification code 0 No error 1 Unkown object in object specifier 2 Unknown target object type 3 Unknown object instance 4 Unknown attribute name 5 Read-only attribute. Access denied. 6 Unknown object type 7 Invalid attribute val. 8 Syntax error 9 Validation error 10 Verification error 11 Object specifier in use 12 Parameter not correctly specified 13 Not all parameters to be specified are specified. 14 Requested option not supported 15 In use 16 Processing preparation not ready 17 Invalid command in current status 18 No changed material 19 Material partially processed 20 Material all processed 21 Error relating to recipe specification 22 Failed during process 23 Failed, not during process。 24 Failed due to insufficient material 25 Job abort 26 Job stop 27 Job clear 28 Recipe selected cannot be changed 29 Undefined event 30 Duplicate report ID 31 Undefined data report 32 Data report unlinked 33 Undefined trace report 34 Duplicate trace ID 35 Too many data reports 36 Sample period out of range 37 Group size too large 38 Recovery action currently invalid 39 Other recovery preventing requested recovery currently underway 40 No active recovery action 41 Exceptional recovery failure 42 Exceptional recovery abort 43 Invalid table element 44 Undefined table element 45 Previously set item cannot be deleted 46 Invalid token 47 Invalid parameter

		48~63 Reserved
ERRTEXT	a	Character string stating error shown in *ERRCODE*. Max 80 characters
OBJSPEC	a	Text string used to indicate a specific object instance having an internal format. This string is composed of a series of formatted sub-strings, each of which identifies object type and classifier. Sub-string format is composed of the following 4 fields. Object type Colon " : " Object classifier Inequality sign " > " The colon " : " is used at the end of the object type. The inequality sign" > " is used at the end of the classifier. Object type is also determined by other methods so it may be omitted. The final " > " is optional.

■Exceptions

Only if verification of just one recipe was requested and it was completed, will *LINKID be* 0. if Item 3 is an item of length 0, no object-format derivative recipe has been created. Only if a recipe was not verified, or if verification failed, will n=0. Only if *RMACK* indicates no error will p=0.

10.2.91 S15F31 Recipe Unload Request

Recipe Unload Request
S,H→E, Reply

■Description

This message is used to request recipes to be executed from the recipe executor.

■Structure

```
s15f31
<a RCSPEC>
```

Name	Format	Description
RCPSPEC	a	Recipe specifier. Recipe object specifier.

10.2.92 S15F32 Recipe Unload Data

Recipe Unload Data
M,H←E

■Description

This message is used to send recipes to be executed from the recipe executor.

■Structure

```
s15f32
{
  <a RCPSPEC>
  {
    {
      <a RCPATTRID1>
      <RCPATTRDATA1>
    }
    .
    .
    {
      <a RCPATTRIDm>
      <RCPATTRDATAm>
    }
  }
  <RCPBODY>
  {
    <u1 RMACK>
    {
      {
        <ERRCODE1>
        <a ERRTEXT1>
      }
      .
      .
      {
        <ERRCODEp>
        <a ERRTEXTp>
      }
    }
  }
}
```

Name	Form-at	Description
RCPSPEC	a	Recipe specifier. Recipe object specifier.
RCPATTRID	a	Non-classifier attribute name (classifier)
RCPATTRDATA	l, b, bool, a, i*, f*, u*	Recipe attribute contents (value)
RCPBODY	b, a, i*, u*	Recipe main body
RMACK	u1	Communicates whether requested action was successfully completed, was denied, terminated due to error, or completed with notification to the requestor. 0 Completed successfully 1 Cannot execute corresponding action 2 Terminated due to error 3 Corresponding action will be completed and notification sent 4 Corresponding action does not require existence
ERRCODE	u*	Error classification code 0 No error 1 Unkown object in object specifier 2 Unknown target object type 3 Unknown object instance 4 Unknown attribute name 5 Read-only attribute. Access denied. 6 Unknown object type 7 Invalid attribute val. 8 Syntax error

		9 Validation error 10 Verification error 11 Object specifier in use 12 Parameter not correctly specified 13 Not all parameters to be specified are specified. 14 Requested option not supported 15 In use 16 Processing preparation not ready 17 Invalid command in current status 18 No changed material 19 Material partially processed 20 Material all processed 21 Error relating to recipe specification 22 Failed during process 23 Failed, not during process。 24 Failed due to insufficient material 25 Job abort 26 Job stop 27 Job clear 28 Recipe selected cannot be changed 29 Undefined event 30 Duplicate report ID 31 Undefined data report 32 Data report unlinked 33 Undefined trace report 34 Duplicate trace ID 35 Too many data reports 36 Sample period out of range 37 Group size too large 38 Recovery action currently invalid 39 Other recovery preventing requested recovery currently underway 40 No active recovery action 41 Exceptional recovery failure 42 Exceptional recovery abort 43 Invalid table element 44 Undefined table element 45 Previously set item cannot be deleted 46 Invalid token 47 Invalid parameter 48~63 Reserved
ERRTEXT	a	Character string stating error shown in *ERRCODE*. Max 80 characters

■Exceptions

Only when *RMACK* shows no errors will P=0.

10.2.93 S15F35 Recipe Delete Request

Recipe Delete Request
M,H→E, Reply

■Description

This message is used to request deletion or deselection of a single or multiple recipes. In the case of multi block, the S15F1 Recipe Management Multi Block Inquire、S15F2 Recipe Management Multi Block Grant transactions precede this.

■Structure

```
s15f35w
{
  <u4 DATAID>
  <a RESPEC>
  <u1 RCPDEL>
  {
    <a RCPID1>
    .
    .
    <a RCPIDn>
  }
}
```

Name	Format	Description
DATAID	u4	Data ID
RESPEC	a	Recipe executor object specifier
RCPDEL	u1	0 Delete 1 Deselect >1 Reserved
RCPID	a	Recipe classifier. Formatted text is in accordance with requirements of *OBJSPEC*.
OBJSPEC	a	Text string used to indicate a specific object instance having an internal format. This string is composed of a series of formatted sub-strings, each of which identifies object type and classifier. Sub-string format is composed of the following 4 fields. Object type Colon “ : ” Object classifier Inequality sign “ > ” The colon “ : ” is used at the end of the object type. The inequality sign “ > ” is used at the end of the classifier. Object type is also determined by other methods so it may be omitted. The final “ > ” is optional.

■Exceptions

Lists with n=0 in cases of deselecting a recipe (*RCPDEL*=1) indicate that all currently selected recipes will be displayed.

10.2.94 S15F36 Recipe Delete Acknowledge

Recipe Delete Acknowledge
M,H←E

■Description

This message is used to acknowledge requests to delete or deselect recipes.

■Structure

```
s15f36
{
  <u1 RMACK>
  {
    {
      <ERRCODE1>
      <a ERRTEXT1>
    }
    .
    .
    {
      <ERRCODEp>
      <a ERRTEXTp>
    }
  }
}
```

Name	Format	Description
RMACK	u1	Communicates whether requested action was successfully completed, was denied, terminated due to error, or completed with notification to the requestor. 0 Completed successfully 1 Cannot execute corresponding action 2 Terminated due to error 3 Corresponding action will be completed and notification sent 4 Corresponding action does not require existence
ERRCODE	u*	Error classification code 0 No error 1 Unkown object in object specifier 2 Unknown target object type 3 Unknown object instance 4 Unknown attribute name 5 Read-only attribute. Access denied. 6 Unknown object type 7 Invalid attribute val. 8 Syntax error 9 Validation error 10 Verification error 11 Object specifier in use 12 Parameter not correctly specified 13 Not all parameters to be specified are specified. 14 Requested option not supported 15 In use 16 Processing preparation not ready 17 Invalid command in current status 18 No changed material 19 Material partially processed 20 Material all processed 21 Error relating to recipe specification 22 Failed during process 23 Failed, not during process 24 Failed due to insufficient material 25 Job abort 26 Job stop 27 Job clear 28 Recipe selected cannot be changed 29 Undefined event 30 Duplicate report ID 31 Undefined data report 32 Data report unlinked 33 Undefined trace report 34 Duplicate trace ID 35 Too many data reports 36 Sample period out of range 37 Group size too large 38 Recovery action currently invalid 39 Other recovery preventing requested recovery currently underway 40 No active recovery action 41 Exceptional recovery failure 42 Exceptional recovery abort 43 Invalid table element 44 Undefined table element 45 Previously set item cannot be deleted 46 Invalid token 47 Invalid parameter 48~63 Reserved
ERRTEXT	a	Character string stating error shown in *ERRCODE*. Max 80 characters

■Exceptions

Only if RMACK shows no errors will p=0.