bop ActiveX コントロール
ユーザーズマニュアル

Jazz Soft, Inc.

1 改訂履歴

版	改訂日	改訂者	説明
1.00	Sep,5th,2005	Hikaru Okada	新規作成。
1.00b	Jan,29th,2006	Hikaru Okada	.bop ファイルの構造変更。 BopStudio, BopCompiler, BopRetriever, BopOldRetriever の新規追加。
1.00c	Sep,14th,2009	Hikaru Okada	BopCompiler の CEIDModel クラスの記述で引数が足りなかったのを加筆。

2 ソフトウェア修正履歴

版	改訂日	改訂者	説明
1.00	Sep,5th,2005	Hikaru Okada	新規作成。
1.00a	Nov,21st,2005	Hikaru Okada	HASP キー用ライブラリの変更に伴う修正。
1.00b	Jan,30th,2006	Hikaru Okada	.bop ファイルの構造変更。従来の.bop ファイルでは一部の内部クラスのバージョン番号を正しく知ることができないため。現段階では特に問題はないが、将来の機能追加時に問題となる可能性があるため。 BopStudio, BopCompiler, BopRetriever, BopOldRetriever の新規追加。
1.00c	Sep,14th,2009	Hikaru Okada	LogicalConnectionFileName プロパティで正しくファイル名がセットされない場合があった不具合を修正。

3 はじめに

GEM 対応の装置を開発するには従来は非常に長い開発期間と費用がかかった。開発ライセンスと称して初期投資に数百万円、年間保守契約費用に数百万円といった、一見法外とも思える費用を請求されることも、この業界では当たり前となっている。さらには仕様の難解さや開発の難しさのために、人月単価で百数十万円の開発要員を数人派遣するのがセットになった「抱き合わせ商法」もまかり通っているのが現状である。しかし本当に GEM はそんなに複雑で金のかかる開発なのだろうか？

Jazz Soft では、この素朴な疑問から Swing シリーズの開発・販売を行ってきた。Swing よりもさらに簡単に GEM 対応にするため、このほど bop ActiveX コントロールの販売を開始した。Swing も bop も開発には相当の年数と費用を投じており、まだ採算ラインには達していないが、企業努力によって改善できると確信して、低価格の販売に踏み切った次第である。

bop は GEM 開発ソフトであるが、GEM300 にも対応可能である。今回のバージョンでは GEM300 を直接はサポートしていないが、ユーザ側で機能追加することにより拡張できるよう配慮してある。もちろん Swing だけを使って GEM や GEM300 のシステムを構築することも可能ではあるが、それにはユーザ側でのかなりの機能追加が必要となる。bop はその手間を大きく省いてくれるソフトである。

Jazz Soft の基本姿勢として、

① 永久無償バージョンアップ
② 開発ライセンス不要
③ 保守契約無料

の原則を今後も貫きたい構えである。もちろん、より高機能で低価格な製品というのが目標である。今後の製品にも是非、ご期待願いたい。

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

Hikaru Okada

4 目次

5 使用環境

5.1 開発・動作環境

bop は以下の環境で使用することができる。

O/S	Windows 98(USB 対応版), Windows Me, Windows 2000, Windows XP Professional, Windows XP Home Edition および Windows Server 2003。Windows NT 4.0 と Windows 95 に関しては、O/S が USB をサポートしていないため、USB 版は使用不可能だが、プリンタポート版は使用可能。
開発言語	Microsoft Visual Studio 2005 (C++/CLI, C#, Visual Basic), Visual Studio .NET 2002/2003 (VC++, C#, Visual Basic), Visual Basic 6.0, Visual C++ 6.0, Borland Delphi, C++ Builder などの Active X 対応の 32 ビット開発言語。

5.2 swing の併用

bop 用の HASP キーで swing や SexyM を製品版として動作させることができる。

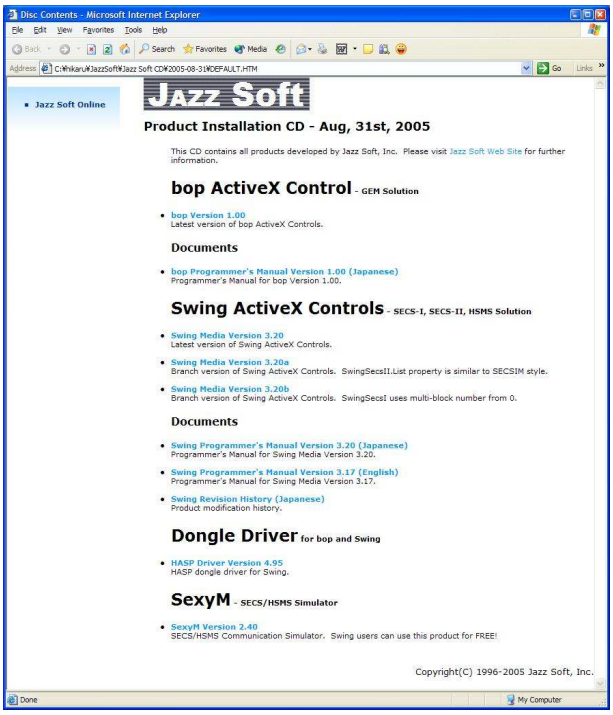
機能	bop	swing
BopStudio	○	○
BopCompiler	○	○
BopRetriever	○	○
BopOldRetriever	○	○
SwingSecsI	○	○
SwingSecsII	○	○
SwingHsms	○	○
SwingComm	○	○
SexyM	○	○
bop	○	×

BopStudio, BopCompiler, BopRetriever, BopOldRetriever は HASP キーなしで利用可能。

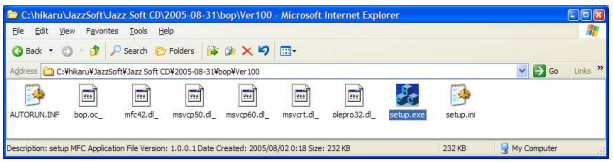
6 インストール

6.1 インストールCDの準備

Jazz Soft のインストールCDをパソコンに入れると以下のような画面が表示される。もし表示されない場合は CD のルートフォルダにある default.htm というファイルをダブルクリックする。



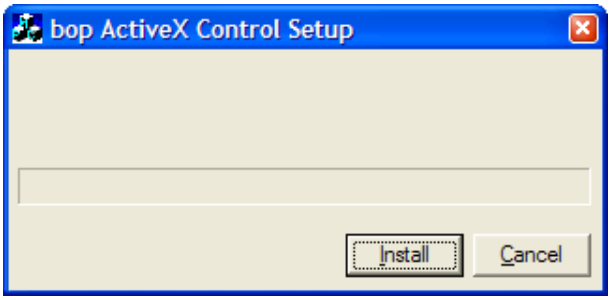
bop や swing はこのCDからインストールすることができる。bop をインストールするには「bop Version 1.00b」をクリックする。



上のように bop のセットアップフォルダの内容が表示される。

6.2 インストーラの実行

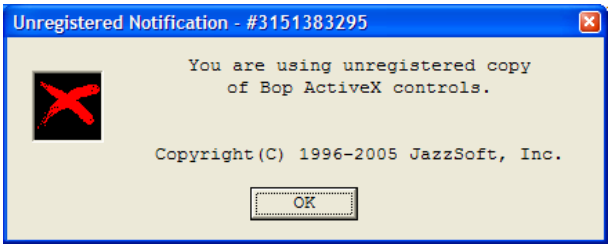
bop のセットアップフォルダから setup.exe をダブルクリックし実行する。



Install ボタンをクリックするとインストールが始まる。完了すると以下のようなメッセージボックスが表示され、インストールが完了する。

6.3 試用版と製品版の違い

bop をインストールしただけでは試用版での動作しかしない。試用版と製品版の違いは特にないが、試用版では頻繁に以下のようなダイアログボックスが表示される。

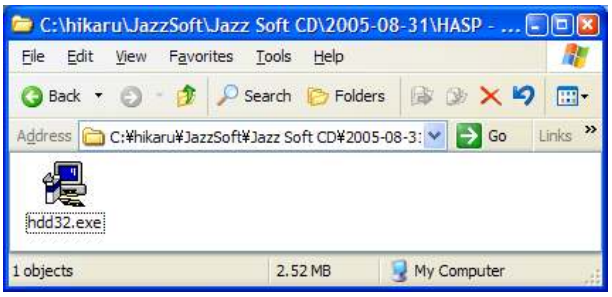


試用版での制限がないため、製品版と同じ動作確認をすることができる。

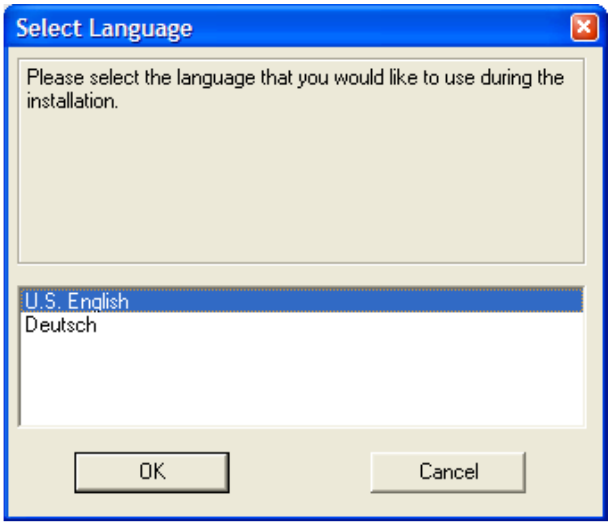
6.4 HASPドライバのインストール

製品版を購入すると HASP キーが送付されてくる。これをパソコンに取り付ければ製品版として動作する。しかしキーをいきなり挿入する前にドライバをインストールする必要がある。

先ほどの Jazz Soft のインストールCDから HASP Driver 4.95[1]をクリックする。



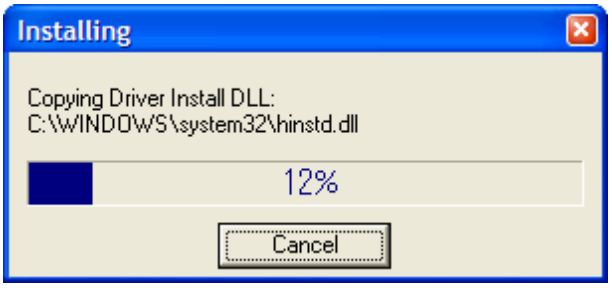
hdd32.exe をダブルクリックする。以下の画面が表示されたら「U.S. English」を選択して OK ボタンを押す。



Next ボタンを押す。

[1] バージョンアップされることがあります。

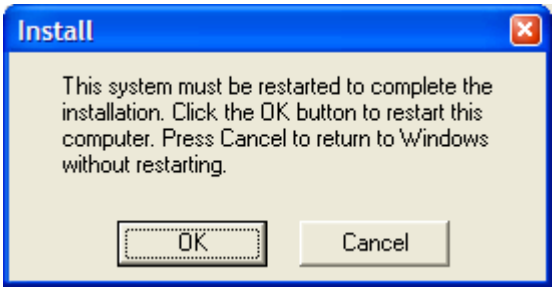
ドライバのコピーが始まる。この時点ではインストールに必要なファイルがコピーされただけである。



インストールの準備ができると、以下のような画面が表示される。Next ボタンを押す。インストールには数分かかる場合もある。

インストールが完了すると以下のような画面が表示される。Finsih ボタンを押す。

一部のバージョンの Windows では以下のように、再起動を促すダイアログボックスが表示されることもある。その場合は OK を押して再起動させる。



7 チュートリアル

何はともあれbopを使うといかに簡単にGEM対応装置を作成できるかを知ってもらうために、シンプルな装置を作ってみよう。このチュートリアルで作成する装置の仕様は以下のようなものである。

■GEMイベント

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

■変数

VID	変数タイプ	型	説明	最小値	最大値	デフォルト値	単位
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				

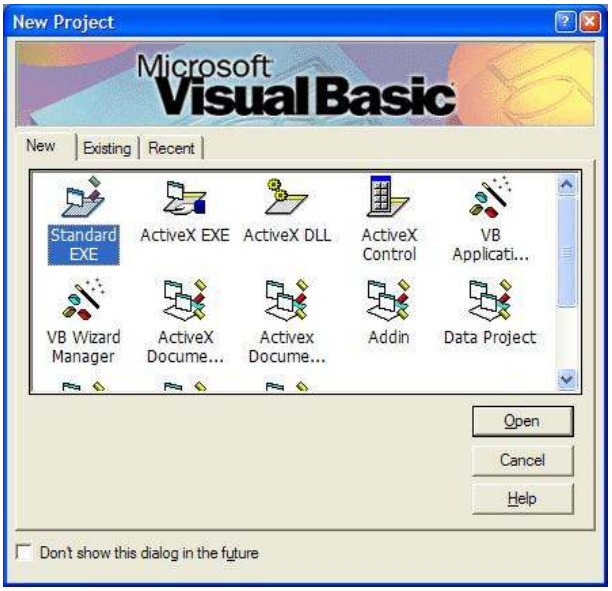
■アラーム

ALID	説明
30000	Alignment Failure

7.1 Visual Basic 6.0編

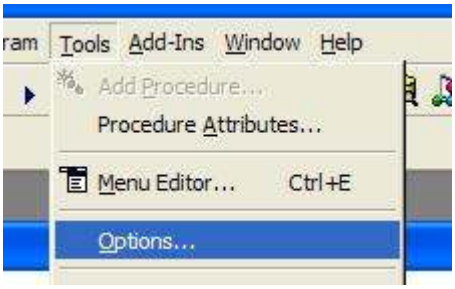
7.1.1 新規プロジェクトの作成

Visual Basic 6を起動すると以下のようなダイアログボックスが表示される。「Standard EXE」が選択されているのを確認してOpenボタンを押す。

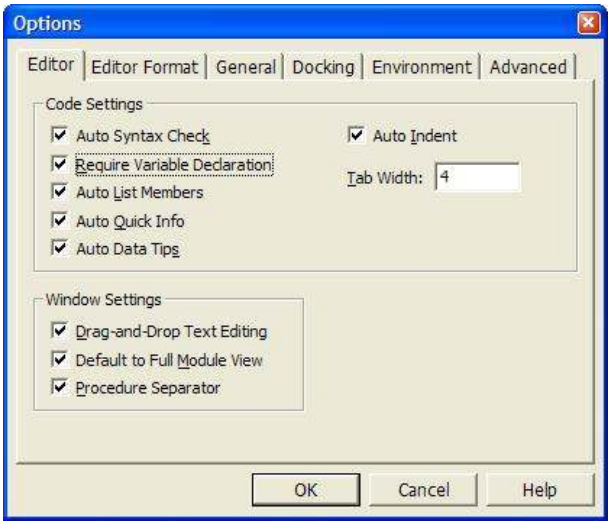


7.1.2 変数の宣言を強要する

デフォルト状態のVisual Basicでは、変数を宣言しなくても使えてしまう。これは昔のBASIC言語の仕様がそうだったからである。このまま開発を行うとスパゲッティのように、ぐちゃぐちゃにからまったプログラムに陥りやすいので、変数の宣言を強要する。メニューから「Tools」-「Options...」を選択する。



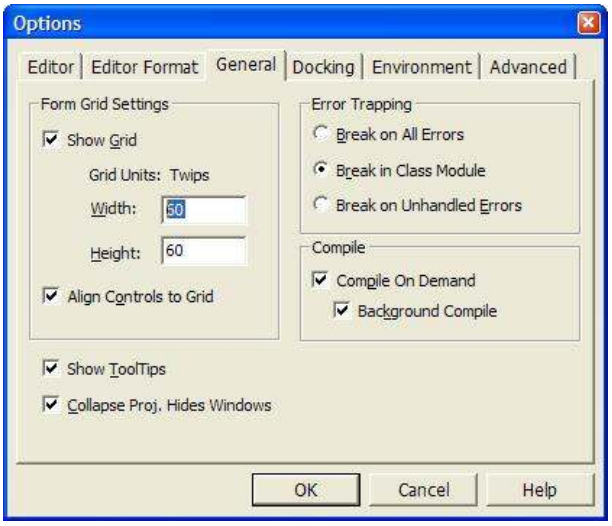
「Editor」タブの「Code Settings」にある「Require Variable Declaration」にチェックマークを入れる。これにより変数の宣言を明示的に行わないで使おうとするとエラーとなる。



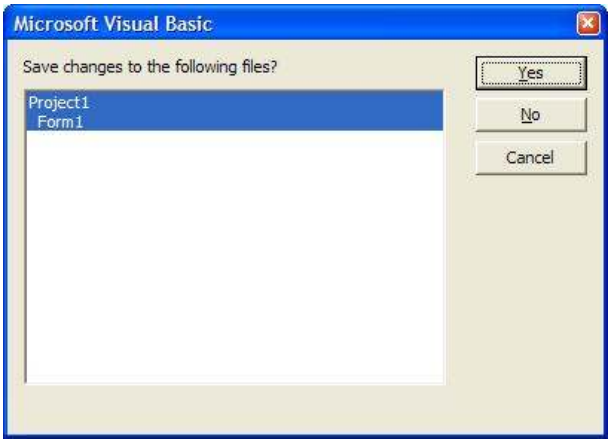
このオプションが有効だと下記のようにソースコードの先頭に「Option Explicit」の宣言が入る。

```
Option Explicit
```

ついでに「General」タブの「Form Grid Settings」にある「Width」と「Height」を60に設定する。デフォルトでは120だが、これではグリッドの目が粗すぎる。



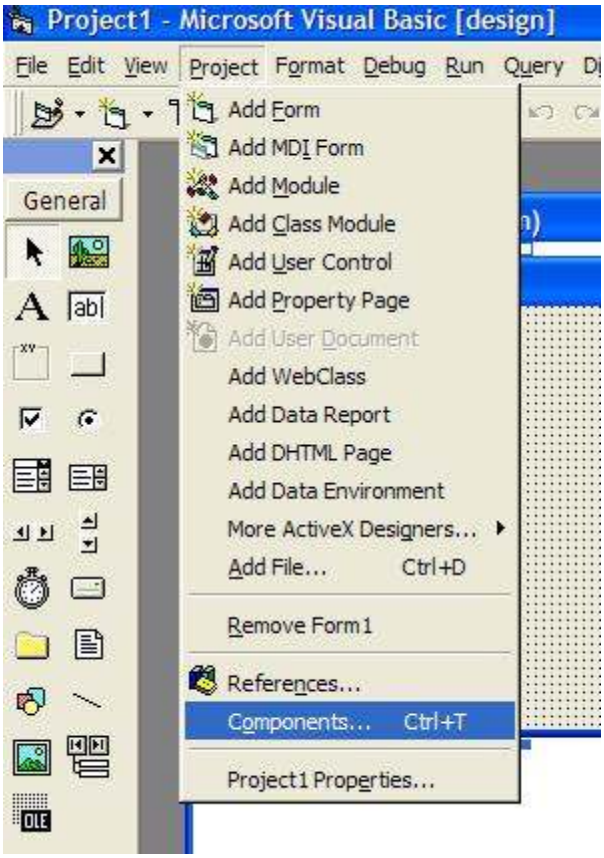
OKボタンを押してOptionダイアログボックスを閉じたら、設定を有効にするために、ここでいったんVisual Basicを終了させて再起動する。終了しようとすると以下のようにプロジェクトを保存するかどうかを聞いてくるが、今回はOptionの設定を行っただけなので、Noボタンを押して保存はしない。



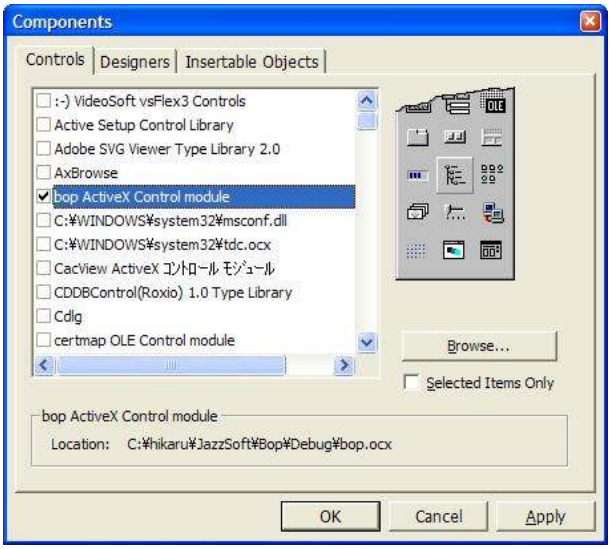
再起動したら再び「Standard EXE」を選択して空のプロジェクトを作成する。

7.1.3 bop をプロジェクトへ追加

まず最初に bop ActiveX コントロールをプロジェクトに追加する。メニューの「Project」-「Components...」を選択する。



インストールされているコンポーネントの一覧から「bop ActiveX Control module」にチェックマークを入れて OK ボタンを押す。



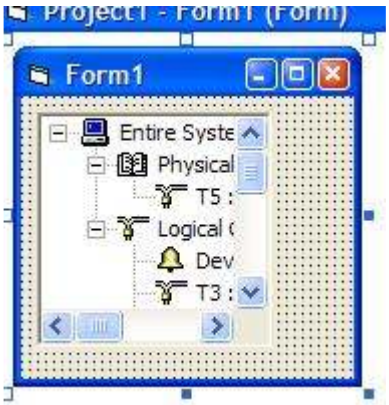
ツールボックスに bop が表示され、いつでも貼り付け可能になった。bop は音符のアイコンで表示される。

7.1.4 画面に貼り付ける

ツールボックスの bop を画面に貼り付けると、以下のようなダイアログボックスが表示される。

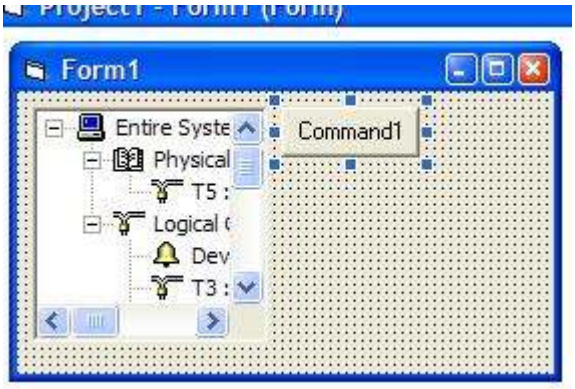
このダイアログボックスは製品版を購入してパソコンに解除キーを取り付けると表示されなくなる。試用版として使用する場合にはたびたび表示されるが、動作には影響しない。邪魔なので OK ボタンを押して閉じる。

bop を画面に貼り付けると以下のようになる。

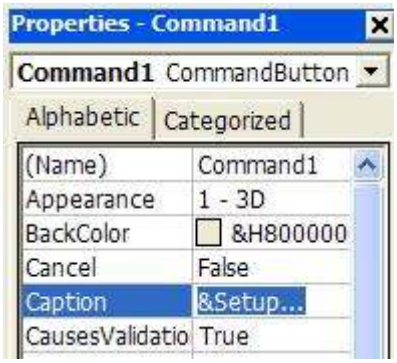


7.1.5 GEM 設定画面を作る

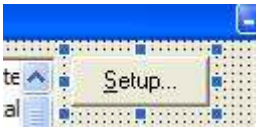
GEM の設定画面を作るのは簡単で、既に bop に用意されているメソッドを呼び出すだけである。まずは画面にボタンを貼り付ける。



ボタンの Caption プロパティを「&Setup...」に変更して設定ボタンらしくする。「&」をつけると下線が表示されショートカットとなる。またボタンを押した結果、ダイアログボックスが表示される場合には、最後に「...」を付加するのが Microsoft 流なので、覚えておくと便利だろう。



Caption プロパティを変更するとボタンの外観は以下のようになる。

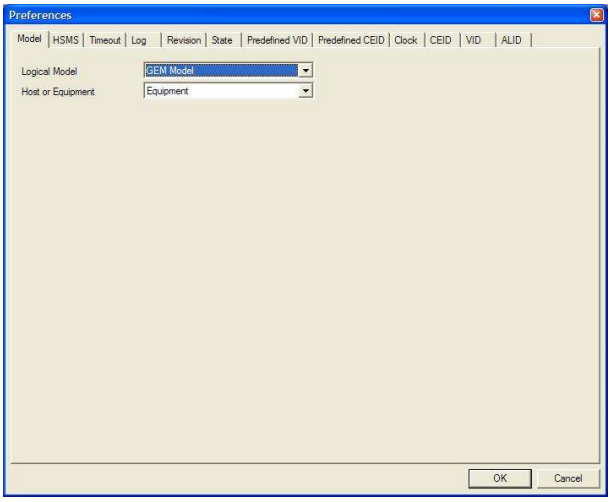


さて外観はともかく中身の実装を行おう。ボタンをダブルクリックするとイベントハンドラ関数が表示される。

Click イベント処理になっているので、このままここに記述する。設定画面を表示させるには bop の Configure メソッドを呼び出す。ここに、

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

とたった一行書くだけである。実に簡単であっけないが、実行してみると設定画面が表示される。



ここまでの動きが確認できたところで、アプリケーションを終了させ、いったんデバッグを終える。

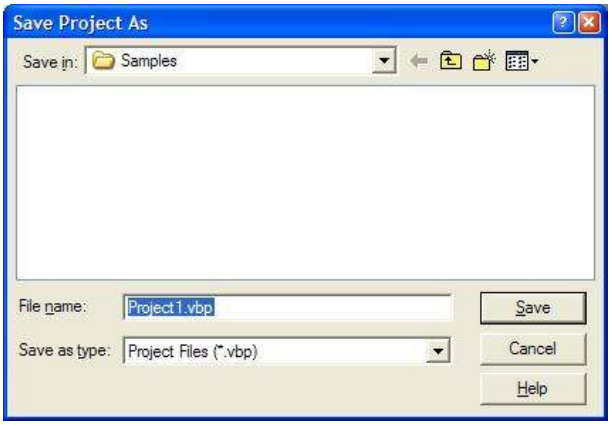
7.1.6 プロジェクトを保存する

Visual Basic 6 で開発をしたことがあれば分かるが、アプリケーションエラーで Visual Basic が死んでしまうことがある。このためせっかくのソースコードが消えてしまわないよう、頻繁に保存するのが安全である。いったんプロジェクトを保存しておこう。

プロジェクトを保存するには、フロッピーディスクのアイコンをクリックする。

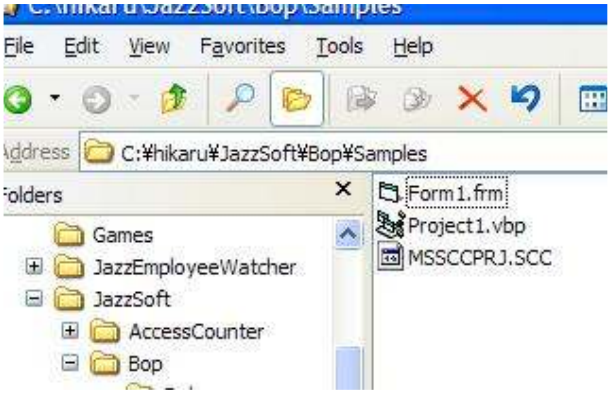
まずフォームを保存するフォルダとファイル名を聞いてくるので、適当なフォルダを作成し、そこに保存する。ファイル名はデフォルトの「Form1.frm」のままで構わない。Save ボタンで保存する。

次にプロジェクトの保存先を聞いてくる。これも同じフォルダに保存する。ファイル名はやはりデフォルトの「Project1.vbp」で構わない。Save ボタンで保存する。



Visual Studio Enterprise Edition をインストールしている場合には、Visual Source Safe (VSS) に登録するかどうかを聞いてくる。今回はチュートリアルなので VSS には登録しないため、No ボタンを押す。

エクスプローラで見ると、Form1.frm と Project1.vbp が同じフォルダに保存されているのが確認できる。



ここで Visual Basic を閉じ、エクスプローラから Project1.vbp をダブルクリックしてプロジェクトが開くことを確認しておこう。新たに Project1.vbw というファイルが追加されていることに気づくだろう。

7.1.7 設定を復元する

先ほどの一行で書けてしまった設定画面だが、設定内容は確かに保存されるものの、次にアプリケーションを起動すると消えてしまう。これはアプリケーションが設定内容をロードしていないからである。フォームをダブルクリックし、Form1 の Load イベントに以下のように記述する。

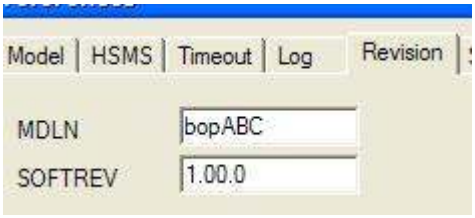
```
Private Sub Form_Load()
   Bop1.LoadIniFile
   Bop1.Load
End Sub
```

7.1.8 Revision の設定

再びアプリケーションを実行し、先ほどの設定画面を設定していこう。まず Revision タブは以下のように設定する。

項目	値
MDLN	bopABC
SOFTREV	1.00.0

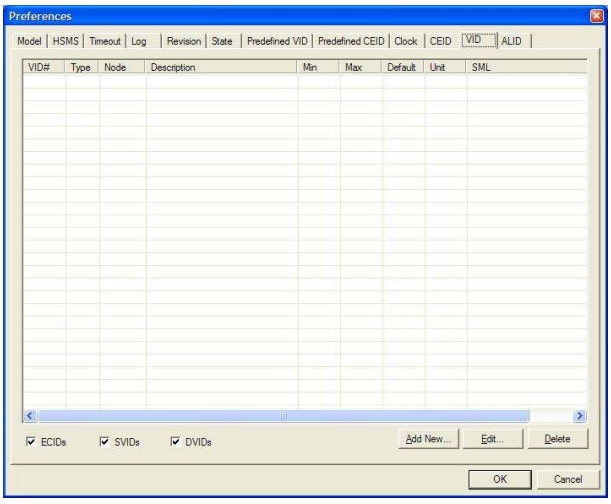
この値は何でも構わないのだが、最大でも 6 文字以下でなければならない。また全角文字や半角カタカナは入れられない。



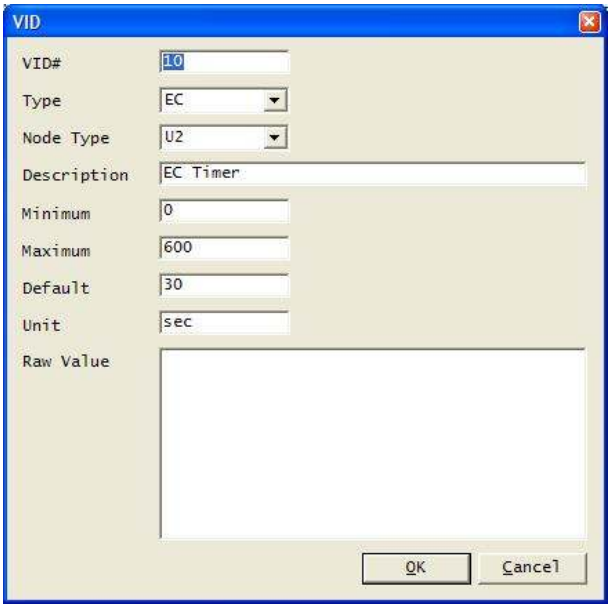
7.1.9 VID の設定

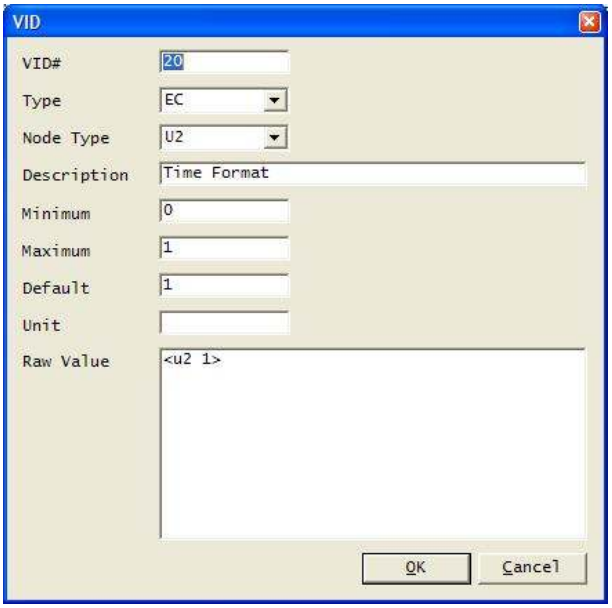
VID タブには今回のチュートリアルの仕様にしたがって、以下の変数を登録する。

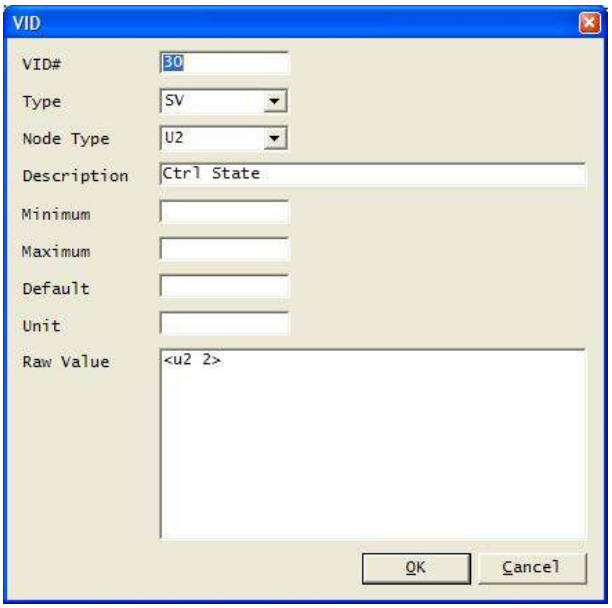
VID	変数タイプ	型	説明	最小値	最大値	デフォルト値	単位
10	EC	U2	EC Timer	0	600	30	sec
20	EC	U2	Time Format	0	1	1	
30	SV	U2	Ctrl State				
40	DV	Ascii	Carrier ID				

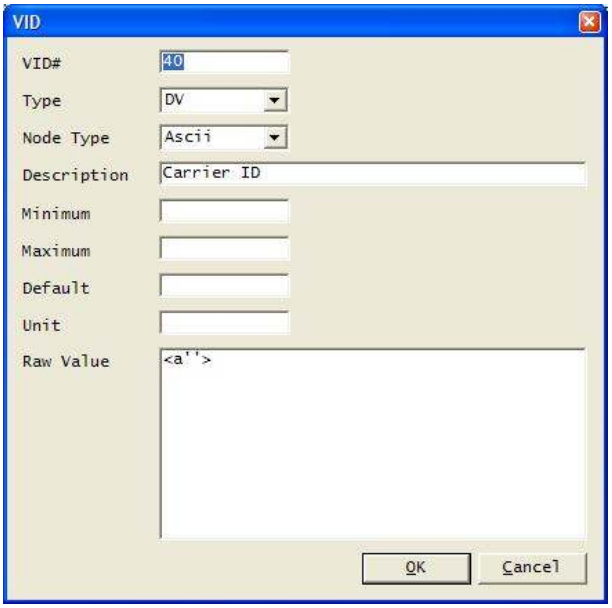


Add New...ボタンを押して一つずつ登録していく。

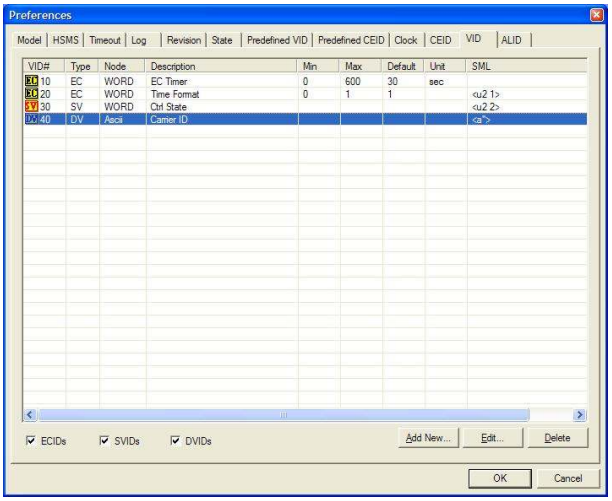








登録が完了すると以下のようになる。



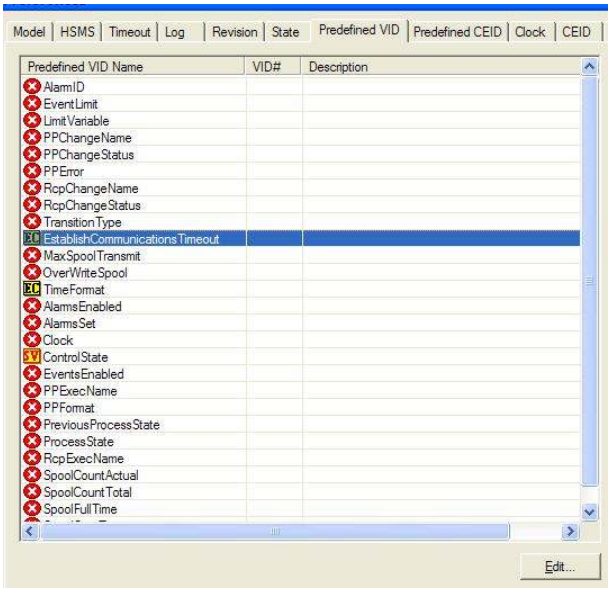
ここで OK ボタンを押して、いったん設定を保存する。

7.1.10 Predefined VID の設定

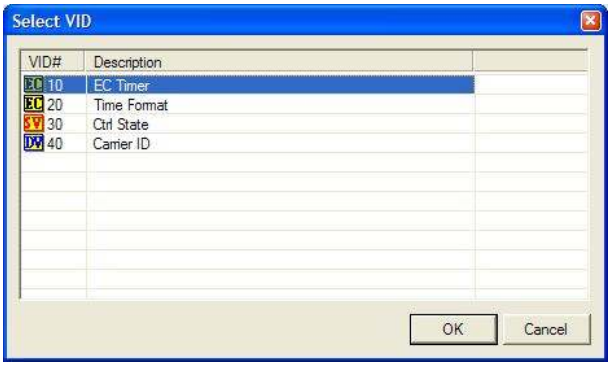
Predefined VID タブは「定義済み変数」の設定を行う。以下のように設定する。

定義済み変数名	VID#
Establish Communications Timeout	10
Time Format	20
Control State	30

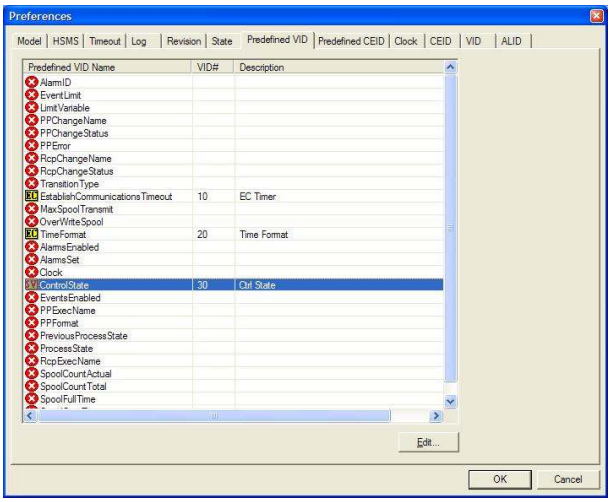
まず「Establish Communications Timeout」を選択し、Edit... ボタンを押す。



先ほど登録した VID が一覧に表示される。いったん設定を保存したのはこのためである。VID#1 を選択して OK ボタンを押す。



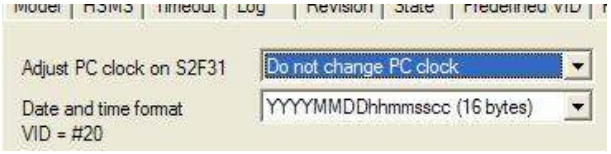
同様に「Time Format」と「Control State」も設定する。



ここでまた OK ボタンを押して、いったん設定を保存しておく。

7.1.11 Clock の設定

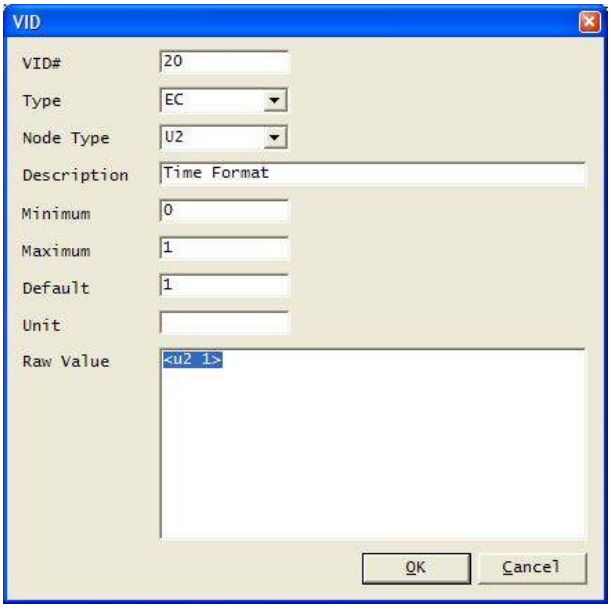
Clock タブは日付・時刻についての設定を行う。この画面の「Date and time format」に、先ほど登録した「VID = #20」が表示されているのが確認できる。



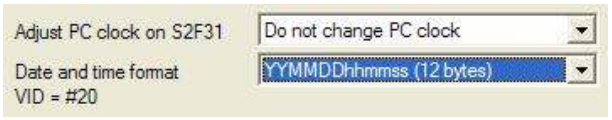
VID タブで確認すると、VID#20 の Raw Value（表では SML と表示されている）が、

```
<u2 1>
```

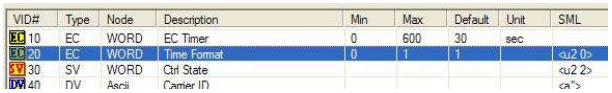
と書き換えられているのが確認できる。



試しに Clock タブの「Date and time format」を「12 bytes」に変更してみよう。



OK ボタンを押して設定を保存し、再び設定画面を開くと VID#20 が書き換えられているのがわかる。

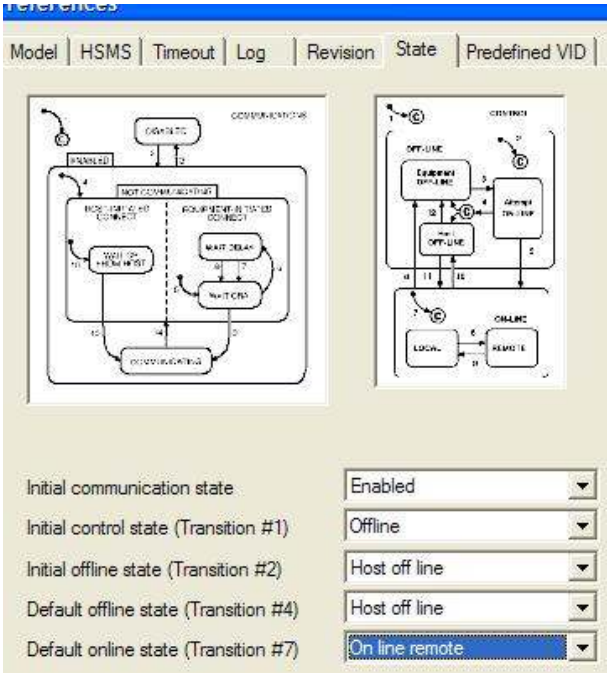


今回のチュートリアルではデフォルトのままで何も変更しない。このため「Date and time format」は「16 bytes」に戻しておく。

7.1.12 State の設定

State タブは以下のように設定する。

項目	値
Initial communication state	Enabled
Initial control state	Offline
Initial offline state	Host offline
Default offline state	Host offline
Default online state	Online remote

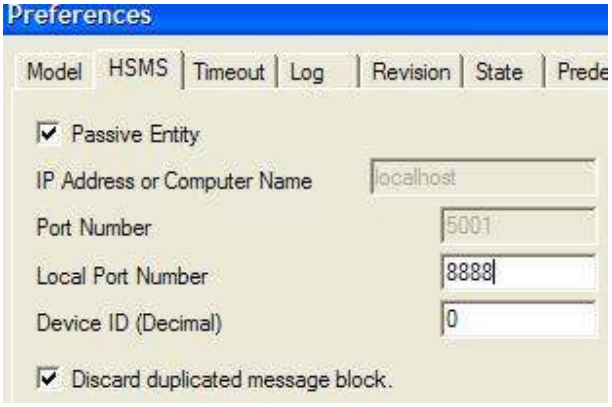


7.1.13 HSMS の設定

HSMS タブはこのチュートリアルでは以下のように設定する。

項目	値
Passive Entity	Yes（チェックマークをつける）
Local Port Number	8888

ほかのパラメータはデフォルトのままでよい。

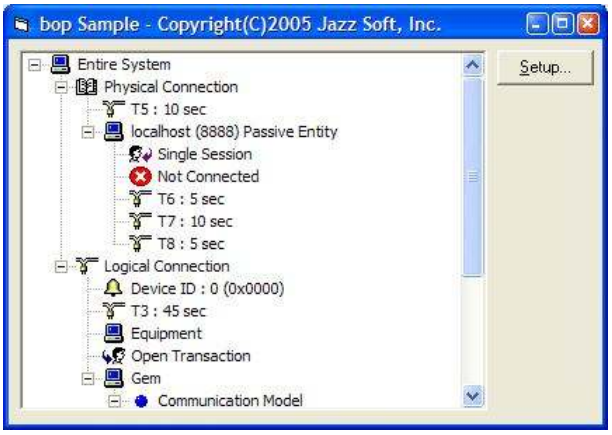


7.1.14 通信を有効化する

HSMS の通信を開始するには、物理接続を有効にすればよい。具体的には、PhysicalConnection プロパティに true をセットする。

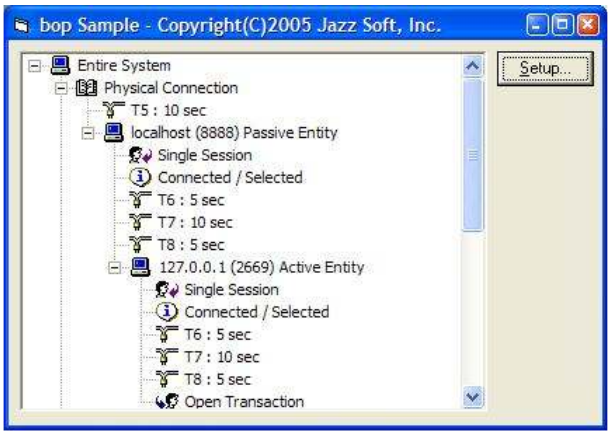
```
Bop1.LoadIniFile
Bop1.Load
Bop1.PhysicalConnection = True
```

ではここで HSMS で本当につながるか確認してみよう。アプリケーションを実行すると、以下のように物理接続が有効になったのがわかる。



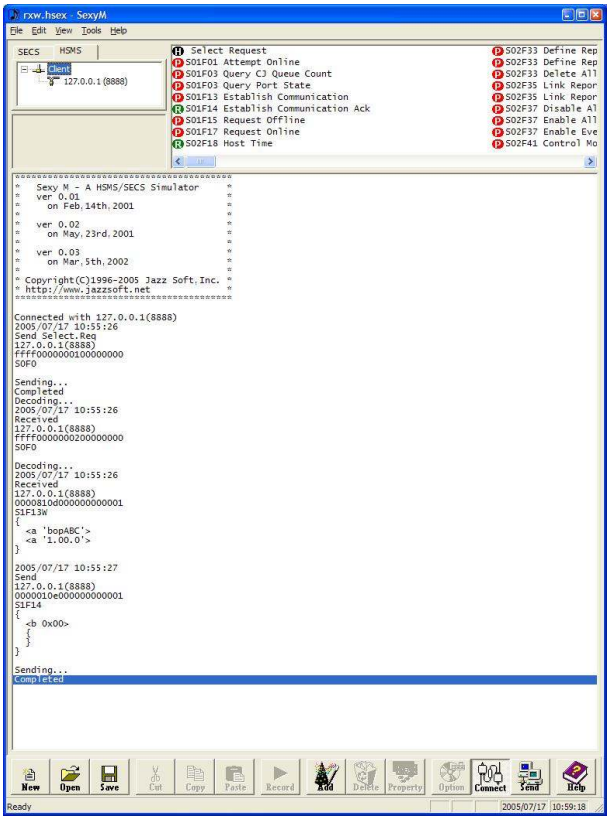
しかしこの時点ではサーバ（パッシブエンティティ）が起動しただけであり、クライアント（アクティブエンティティ）からの接続がなければ、コネクションは成立していない。

そこで通信シミュレータを使い、実際にアプリケーションに接続してみよう。通信シミュレータソフトは何でも構わないが、ここでは無償[2]でも利用できる Jazz Soft の SexyM を使ってみることにする。



接続すると上のようにアクティブエンティティが表示される。シミュレータ側でも以下のように接続したことがわかる。

[2] 機能的な制限は一切ありませんが、Swing もしくは bop のキーがないとダイアログボックスが表示されます。



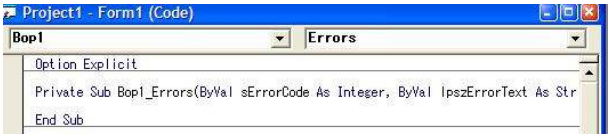
さきほど Revision タブで設定した MDLN と SOFTREV が S1F13 通信確立要求(CR) で送られてくるのが確認できる。MDLN と SOFTREV は S1F2 オンラインデータ(D) でも使われるので、オンライン移行するときにも確認してみてほしい。

しかし S1F13/14 のトランザクションが完結したはずなのに、しばらくすると T3 タイムアウトが発生してしまう。これはどうしてなのか？

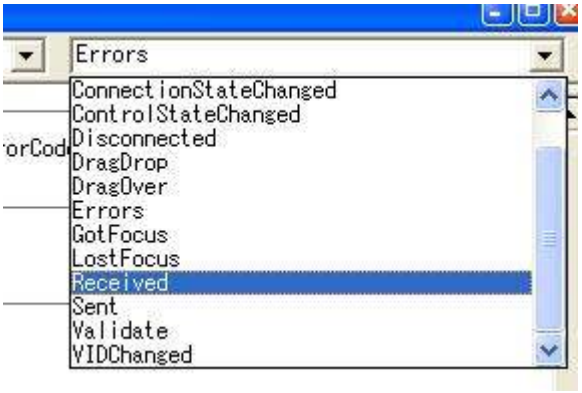
7.1.15 メッセージ処理

bop ではメッセージを受信すると、まずアプリケーションに通知し、自動的に処理することはない。アプリケーション側ではこのメッセージを画面に表示したり、履歴を残したり、何かの処理を実行することもできる。しかし大半のメッセージは bop に任せてしまえばよい。今回は全てのメッセージを bop に委ねることにする。

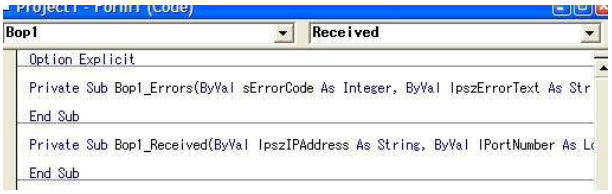
デザイン画面で bop をダブルクリックし、イベントハンドラを作成する。最初はコードウィンドゥには Errors イベントのハンドラが作成されている。



コンボボックスで Received イベントに変更する。



下のように Received イベントのハンドラが作成される。Errors イベントのハンドラは、今回は使わないので削除する。



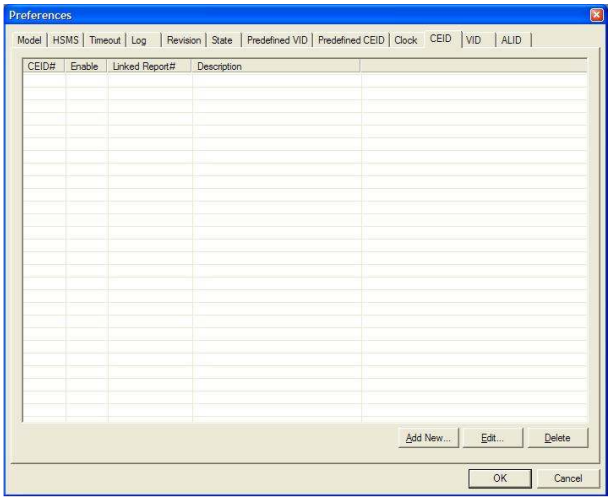
Received イベントハンドラ中に以下のように記述する。またしても一行だけのシンプルさである。

```
Bop1.DefProc
```

これにより受け取ったメッセージは bop が自動的に処理するようになった。

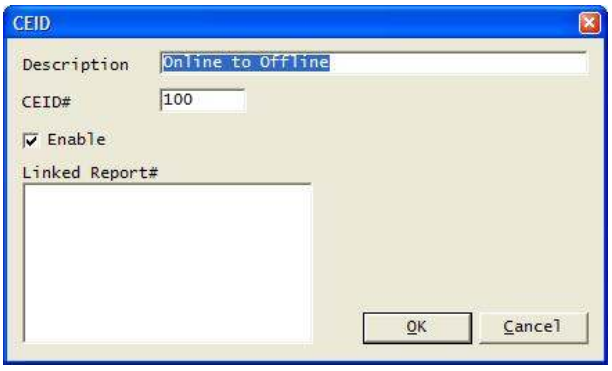
7.1.16 CEID の設定

GEM イベントを発生させるには、CEID を登録する必要がある。アプリケーションを起動し CEID タブを編集する。



Add New...ボタンを押して今回のチュートリアルの仕様にしたがって、以下の CEID を登録する。

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



「Enable」にチェックマークを入れないと、GEM イベントは無効になる。GEM イベントの有効・無効の設定は S2F37 有効、無効イベントレポート(EDER) でも設定可能である。

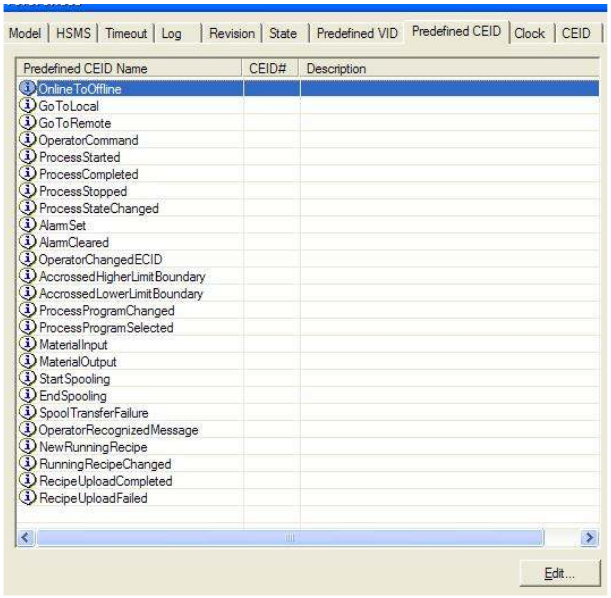
ここで OK ボタンを押して、いったん設定を保存する。

7.1.17 Predefined CEID の設定

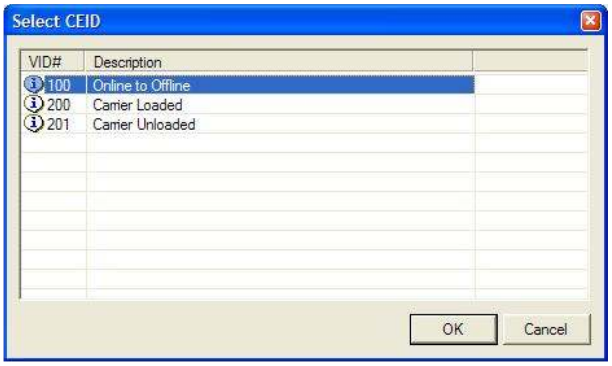
Predefined CEID タブは「定義済み GEM イベント」の設定を行う。以下のように設定する。

定義済み変数名	VID#
Online To Offline	100

まず「Online To Offline」を選択し、Edit...ボタンを押す。



先ほど登録した CEID が一覧に表示される。いったん設定を保存したのはこのためである。CEID#100 を選択して OK ボタンを押す。



ここで OK ボタンを押して設定を保存し、本当に GEM イベントが出るか確認してみよう。S1F13/14 のトランザクションが完結したのを確認して、シミュレータから S1F17 オンライン要求(RONL) を送信してみよう。

```
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>
```

SexyM は詳細なデータを表示するので、それらを削ってメッセージだけを表示することにする。

```
Send
S1F17W
```

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

正しくオンライン移行できたようできる。念のため S1F1 オンライン確認要求(R) を送信して確認してみよう。

```
Send
S1F1W
```

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

正しくオンライン移行したのが確認できた。次に S1F15 オフライン要求(ROFL) を送信してオフラインに移行してみよう。

```
Send
S1F15W
```

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

オフラインに正しく移行できた。これによって Online To Offline イベントが発生する。

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

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

7.1.18 GEM イベントの有効･無効

GEM イベントの有効・無効は通信でも可能だということはすでに述べた。では実際に設定してみよう。S2F37 有効､無効イベントレポート(EDER) で CEED に false を指定し、続くリストが 0 長だと全ての GEM イベントが無効になる。

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

Received
S2F38
```

```
<b 0x00>
```

オフライン移行してみる。

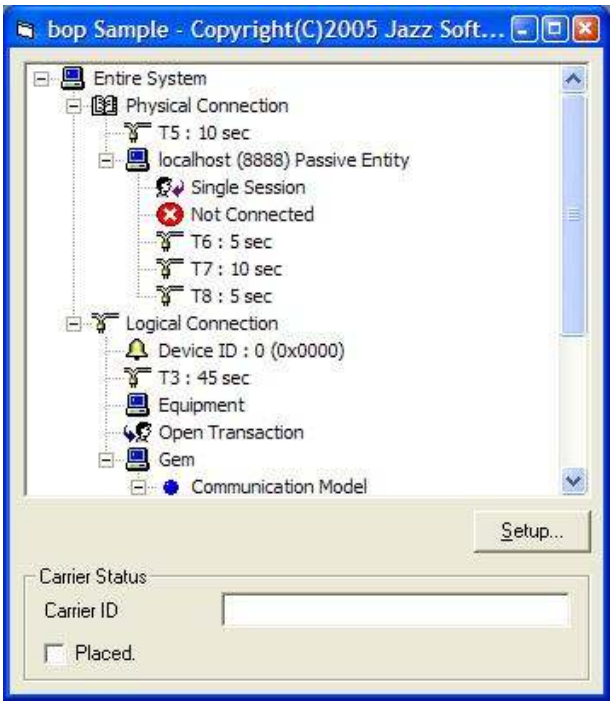
```
Send
S1F15W

Received
S1F16
<b 0x00>
```

GEM イベントが無効になったので、S6F11 は発生しないのか確認できた。

上の例では全ての GEM イベントを一括で無効にしたが、GEM イベントの有効・無効は、CEID 単位で個別に設定することもできる

7.1.19 GEM イベントの送信

今度は、ロードポートにキャリアが置かれたり、外されたりした場合に GEM イベントを発生させることにしよう。まず下のように画面にテキストボックスとチェックボックスを貼り付ける。



チェックボックスの Click イベントに以下のように記述する。

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

全ての GEM イベントを有効にしてみる。

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

Received
S2F38
<b 0x00>
```

アプリケーションを実行し、チェックボックスをクリックすると CEID #200 Carrier Loaded イベントが送信される。

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

Send
S6F12
<b 0x00>
```

もう一回チェックボックスをクリックすると CEID #201 Carrier Unloaded イベントが送信される。

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

Send
S6F12
<b 0x00>
```

7.1.20 動的レポート定義

先ほどの GEM イベントにはレポートが付いていなかったので、通信シミュレータ側からレポートを定義してみよう。まず全ての GEM イベントを無効にする。

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

Received
S2F38
<b 0x00>
```

この状態でチェックボックスをクリックしても、GEM イベントは送信されなくなる。

次に全てのレポートを破棄する。

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

Received
S2F34
<b 0x00>
```

新たなレポートを定義する。ここではレポート#1000 に VID #40 が貼られている。

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

Received
S2F34
<b 0x00>
```

レポートを GEM イベントにリンクする。ここでは CEID #200 にレポート#1000 を、CEID #201 にレポート#1000 をリンクしている。

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

Received
S2F36
<b 0x00>
```

最後に全ての GEM イベントを有効にする。

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

Received
S2F38
<b 0x00>
```

アプリケーションのチェックボックスをクリックすると、今度は GEM イベントにレポートが付いて送信される。

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

Send
S6F12
<b 0x00>
```

7.1.21 変数の更新

VID #40 が更新されていないので、先ほどの GEM イベントでは空の文字列が送信されてしまった。GEM イベントを送信する直前に VID を更新してみよう。VID の更新はまたしても簡単で、一行追加するだけである。

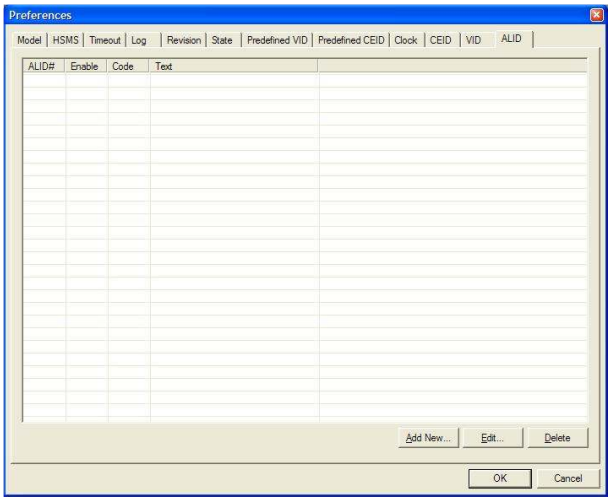
```
Private Sub Placed_Click()
   Bop1.VIDValue(40) = CarrierID.Text
   If Placed.Value = 1 Then
      Bop1.InvokeEvent 200
   Else
      Bop1.InvokeEvent 201
   End If
End Sub
```

アプリケーションのテキストボックスに文字列を入れると、下のようにその文字列が送信されるようになる。

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

Send
S6F12
<b 0x00>
```

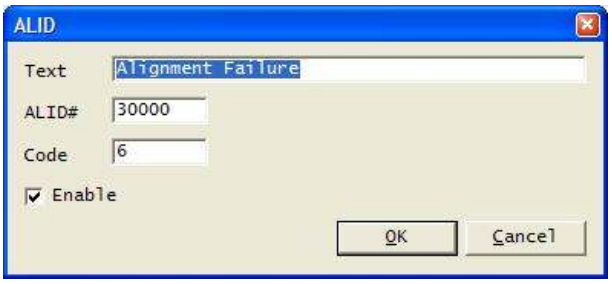
7.1.22 ALID の設定

アラームを発生させるには、ALID を登録する必要がある。アプリケーションの設定画面で ALID タブを編集する。



Add New...ボタンを押して今回のチュートリアルの仕様にしたがって、以下の ALID を登録する。

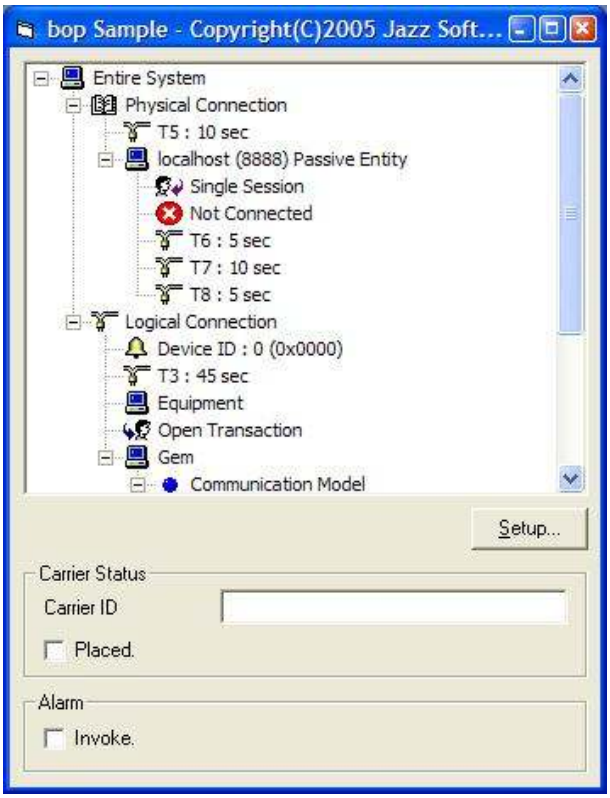
ALID	説明
30000	Alignment Failure



「Enable」にチェックマークを入れないと、アラームは無効になる。アラームの有効・無効も通信経由で設定可能である。ここで OK ボタンを押して、いったん設定を保存する。

7.1.23 アラームの送信

アプリケーションにアラーム発生機能を追加しよう。画面にチェックボックスを貼り付ける。



チェックボックスの Click イベントに以下のように記述する。

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

このコードについて説明しよう。チェックマークが付いた場合には InvokeAlarm メソッドの引数に「-1」を指定している。これはアラームの発生を意味する。同様にチェックマークが外れた場合には「0」を指定しているが、これはアラームの解除を意味する。アラームの発生がなければ、解除は送信されないので注意が必要である。

7.1.24 全ソースコード

さてここまでで、ざっとではあるが bop の機能を紹介してきた。まだまだたくさんの機能があるので全部は解説できなかったが、bop の強力な機能の一端を知ることができたと思う。

このサンプルプログラムは紛れもなく GEM 準拠である。しかしソースコードは信じられないほど短く、全部で 33 行しかない。これは驚嘆に値するだろう。ソフトウェアは行数に比例してバグも増大するので、短いソースコードによってバグは確実に減少する。当然ながら開発期間が短縮され、費用が浮くという訳である。

```
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
```

一点付け加えておく。Visual Basic では With を使ってオブジェクトを省略することができる。

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

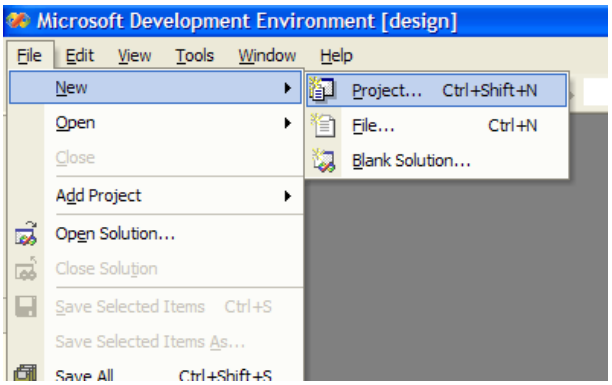
以降の章での説明にはこの省略形を用いる。

7.2 Visual Basic.NET 2003編

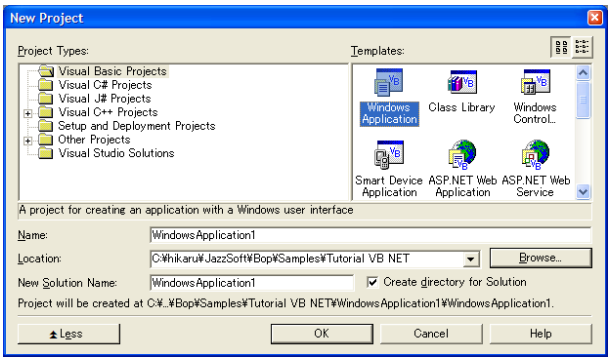
Visual Basic.NET の場合も Visual Basic 6.0 とあまり違いはない。

7.2.1 新規プロジェクトの作成

Visual Studio .NET 2003 を起動し、メニューから「File」-「New」-「Project...」をクリックする。

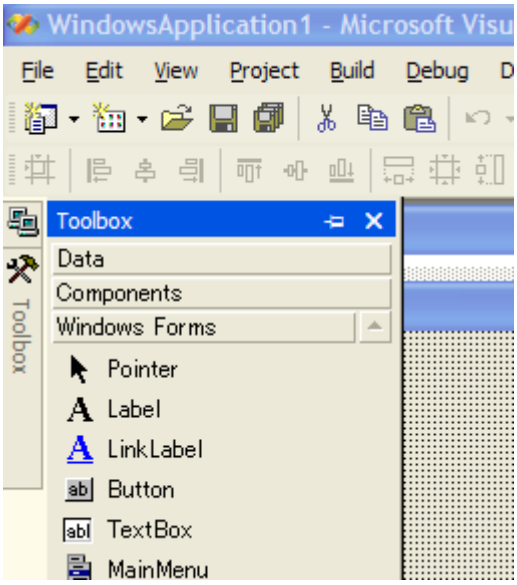


プロジェクトのタイプ一覧から Visual Basic Project を選択し、テンプレートとして Windows Application を選択する。保存するフォルダを指定し、OK ボタンを押す。

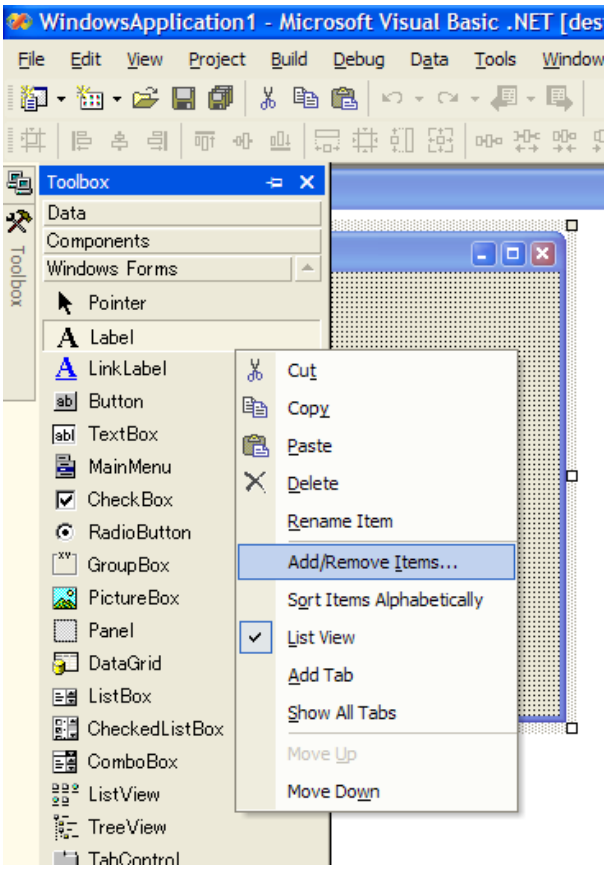


7.2.2 bop をツールボックスに追加

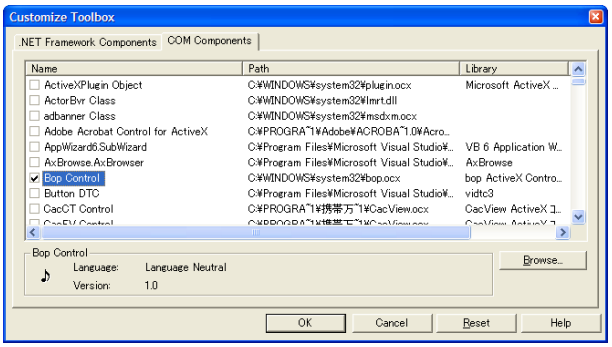
ツールボックスの上にマウスをかざすと、以下のようにツールボックスが開く。



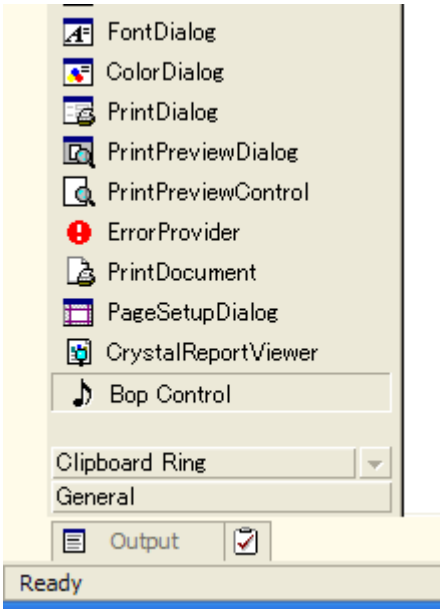
この開いたツールボックス上で右クリックし、Add/Remove Items... を選択する。



COM Components タブを選択し、一覧から Bop Control にチェックマークをつけて OK ボタンを押す。

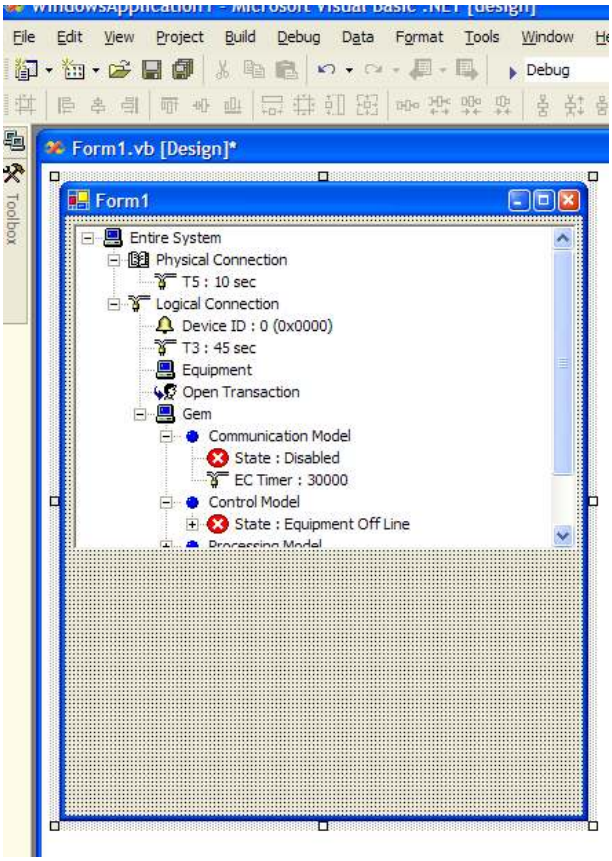


ツールボックスに Bop Control が登録されたのが確認できる。

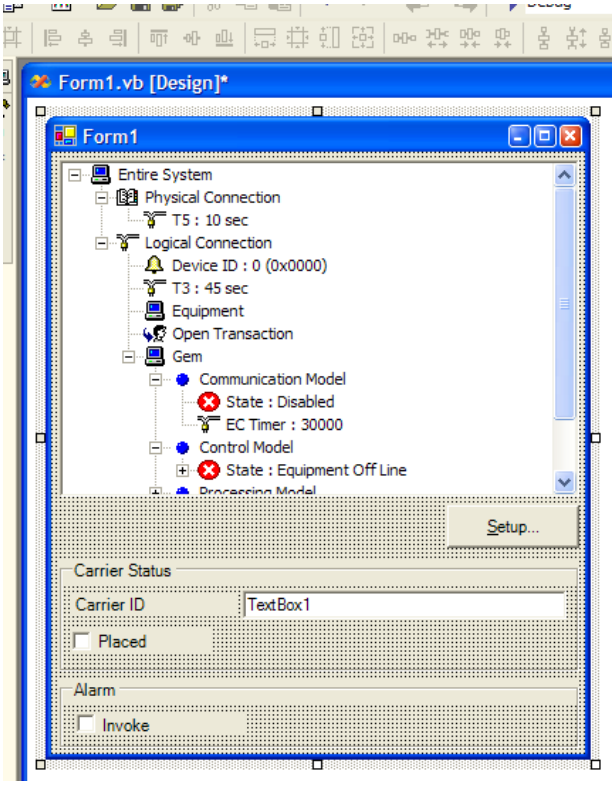


7.2.3 画面に貼り付ける

bop を画面に貼り付けると以下のようになる。

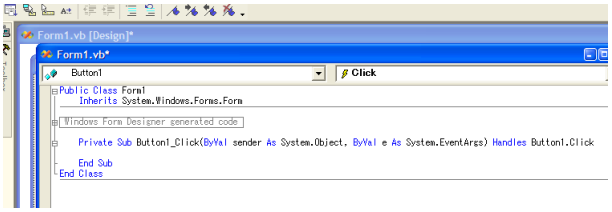


Visual Basic 6.0 編と同じように他のコントロールも貼り付ける。



7.2.4 GEM 設定画面を作る

「Setup...」と書いたボタンをダブルクリックすると、以下のような画面が現れる。



ここに以下のように記述する。

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

Visual Basic 6.0 と同様、一行で書ける。

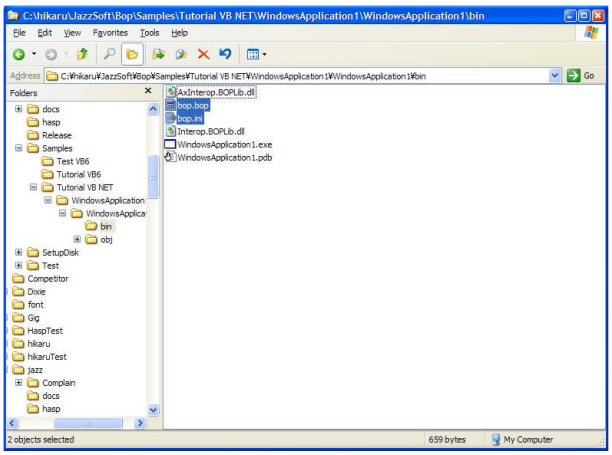
7.2.5 設定を復元する

フォームをダブルクリックし、Form1 の Load イベントに以下のように記述する。

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

7.2.6 設定ファイルをコピーする

GEM の設定方法は Visual Basic 6.0 編と同じである。ここでは Visual Basic 6.0 編で作成した bop.bop と bop.ini をコピーして流用することにする。この二つのファイルを実行フォルダ（bin フォルダ）にコピーする。



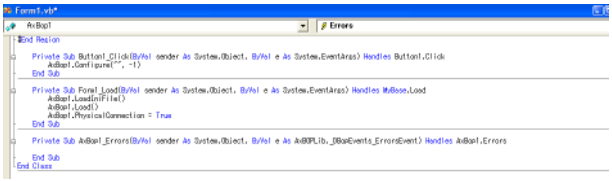
7.2.7 通信を有効化する

HSMS の通信を開始するには、PhysicalConnection プロパティに true をセットする。

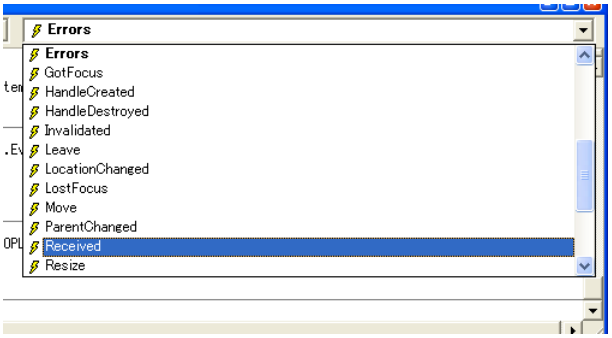
```
AxBop1.LoadIniFile()
AxBop1.Load()
AxBop1.PhysicalConnection = True
```

7.2.8 メッセージ処理

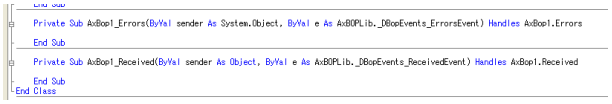
メッセージ受信処理を記述するには、Visual Basic 6.0 編と同様、bop をダブルクリックする。以下のようにまず Errors イベントのハンドラ関数が作成される。



Received イベントに選びなおす。



以下のように Received イベントのハンドラ関数が作成される。Errors イベントのハンドラ関数は消しておく。



ここに以下のように記述する。

```
   AxBop1.DefProc()
```

7.2.9 GEM イベントの送信

GEM イベントを送信する処理も Visual Basic 6.0 編とほとんど同じであるが、Visual Studio .NET では配列型のプロパティへアクセスするための文法が若干変更になっている。

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

7.2.10 アラームの送信

アラームを送信する処理はほとんど変わりがない。

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

7.2.11 全ソースコード

Windows Form Designer が生成したコードを除くと、Visual Basic 6.0 と同様に、非常に短い行数で記述できることが分かる。

```
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
```

7.3 Visual C++ 6.0 編

Visual C++の場合は Visual Basic と言語が異なるため、若干の違いがある。

日本では Basic 言語に対する根強い蔑視の風潮がある。アマチュアプログラマが愛用した N88-BASIC の印象が強いようである。だが Visual Basic は非常に洗練された言語となり、もはや簡易言語の枠を超えたといえるのだが、依然として偏見が残っている。

この Basic 蔑視の風潮から、日本では C++の方が一流で、Basci でプログラムを作成することを嫌う傾向がある。しかし C++は非常に難しい言語であり、たとえ C 言語を完璧に使いこなしているプロフェッショナルでも、容易にはマスターできない。事実、自称 C++プログラマーと称している者の 90%以上が、「コンパイラ環境だけ C++で、中身はほとんど C で記述」というようなお粗末な状況である。これは C++より機能が少ない JAVA のプログラマですら非常に少ないという事実からも明らかだろう。

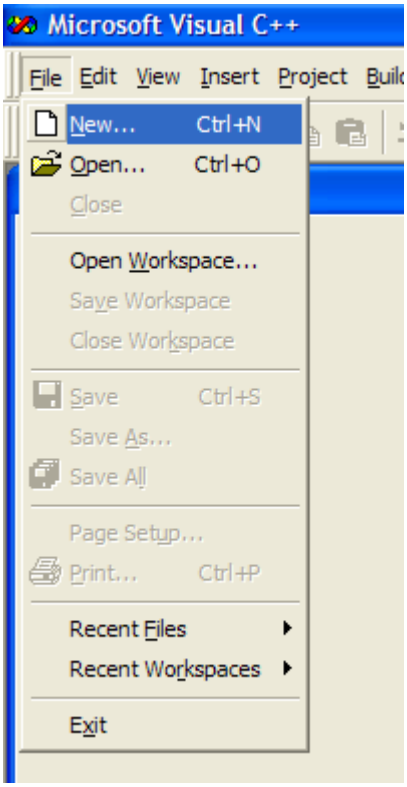
わたしの周りの自称 C++プログラマを見てみると、C++歴 10 年にもなるのに、「デストラクタは virtual にしなければならない」というような基本中の基本すら知らない者も多い。このように一般のプログラマの知識レベルは低いため、C++のエキスパートだという自信がないのなら、開発はあきらめた方がよい。生産性が落ちるだけである。

Visual Studio .NET になってからは、Basic と C++の境界がほとんどなくなったと言っても過言ではない。このため C++脱落組らが考えを改め、「Basic ならマスターできるのではないか」と、少しずつではあるが Basic を使い始めるようになってきている。COBOL や FORTLAN からの転向組には土台、C++をマスターするのは絶望的であり、Basic にすぐ飛びつきたくても、世間の Basic に対する偏見の目が気になっていた。Basic しか分からないようでは恥だと。この Basic 復権の流れは、いいチャンスだといえるだろう。

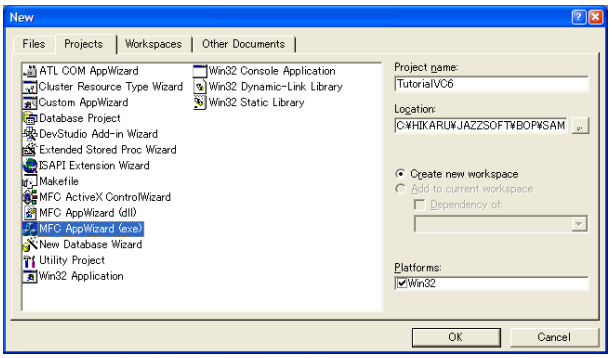
さて話は脱線したが、VC++で作成したチュートリアルのソースコードも Visual Basic と大した違いはない。同じ仕様のソフトを作っているのだから当たり前の話である。ただ、難解な箇所が Basic より多いだけである。このチュートリアルでは VC++や MFC（Microsoft Foundation Class Library）の詳細には踏み込まないので、疑問点は MSDN（Microsoft Developer Network）などを使って、自分で学習していただきたい。[3]

7.3.1 App Wizard で新規プロジェクトの作成

Visual C++ 6.0 を起動し、メニューから「File」-「New...」をクリックする。



MFC AppWizard (.exe)を選択し、プロジェクト名、フォルダを指定したら OK ボタンを押す。

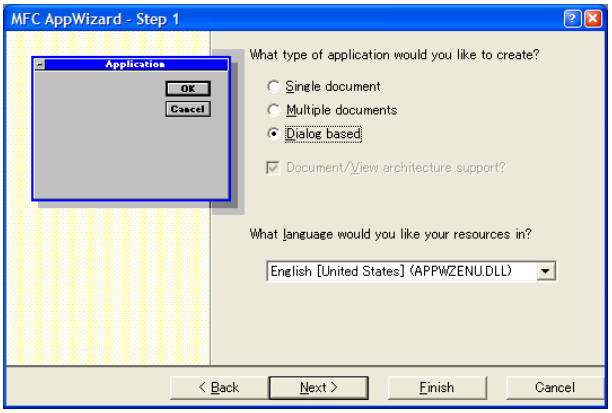


Visual C++ 6.0 で作成できるプロジェクトの種類は 3 種類ある。

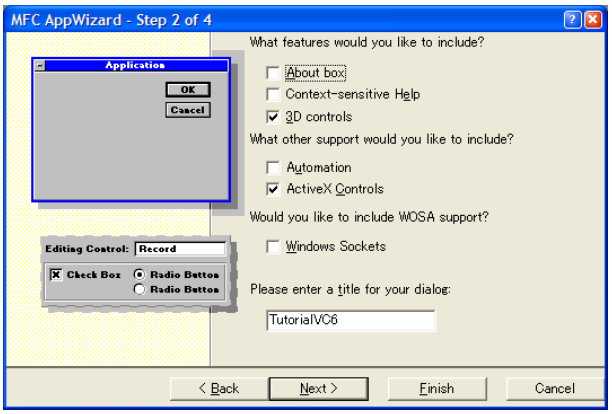
プロジェクト	説明
Single document	「メモ帳」のように一度に一つのドキュメントを扱うアプリケーション。SDI（Single Document Interface）と呼ばれる。
Multiple documents	「Visual C++ 6.0」のように一度に複数のドキュメントを扱うことができるアプリケーション。MDI（Multiple Document Interface）と呼ばれる。
Dialog based	Visual Basic や C#で作成されたソフトのように、ダイアログボックスで始まる簡易アプリケーション。Document/View 構造は使うことができない。

今回は Dialog based アプリケーションを作成することにする。Dialog based のラジオボタンを選択し、Next ボタンを押す。

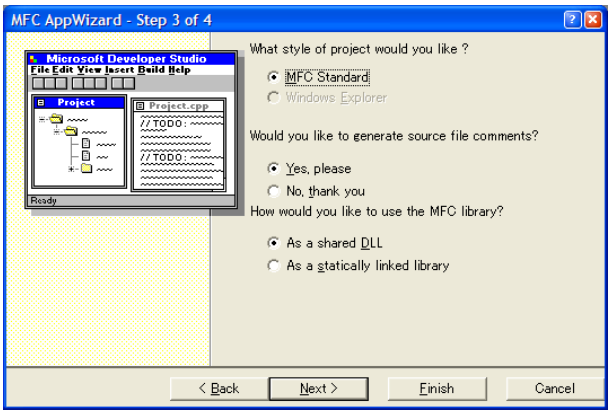
[3] Jazz Soft では bop、swing、VC++のトレーニングも行っています（別途有償）。



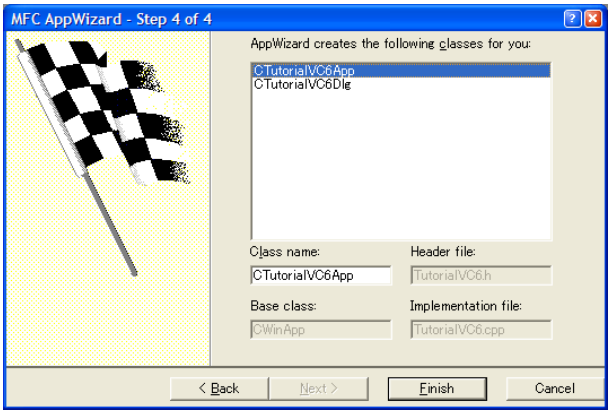
About box（バージョン情報ダイアログボックス）は特に必要ないので、チェックマークを外し、Next ボタンを押す。



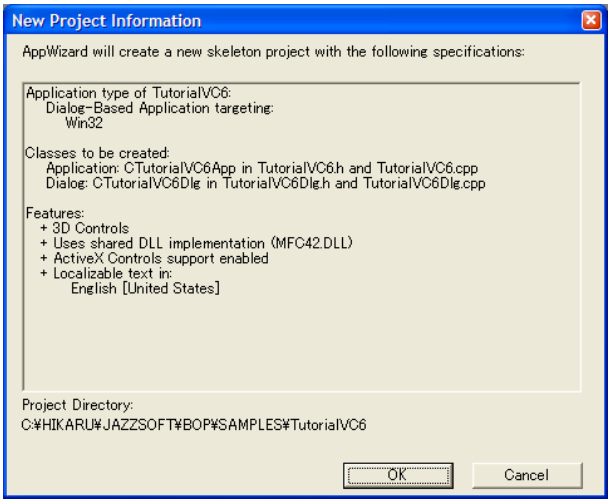
この画面はデフォルトのままで構わないので、そのまま Next ボタンを押す。



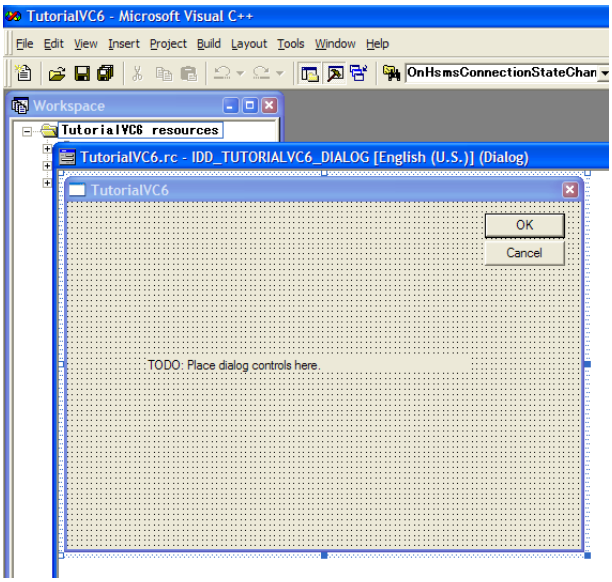
この画面もデフォルトのままで OK なので、Finish ボタンを押す。



最終確認画面が表示される。OK ボタンを押す。



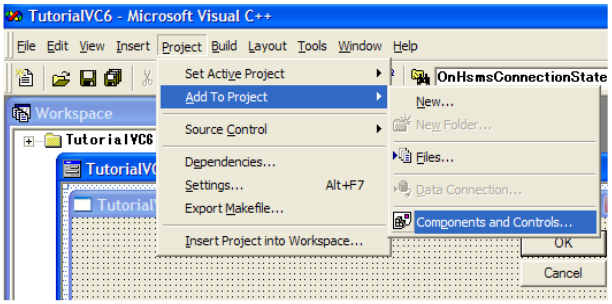
プロジェクトが作成される。



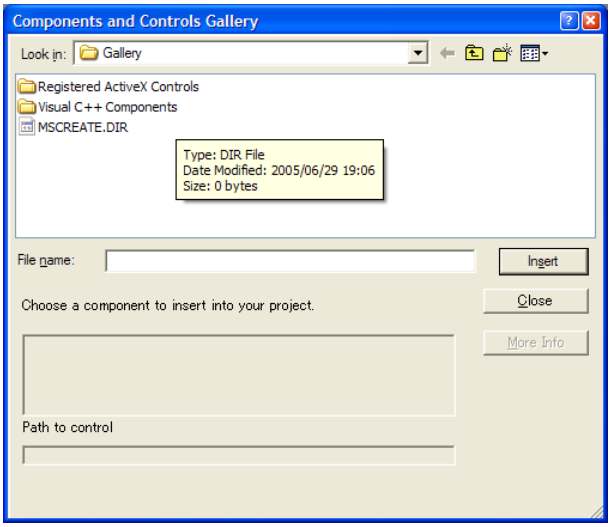
7.3.2 bop を挿入

Visual C++ 6.0 では、bop を使う前に「挿入」（Insert）という前処理が必要となる。これは ActiveX コントロールのタイプライブラリからラッパークラスを作成する。

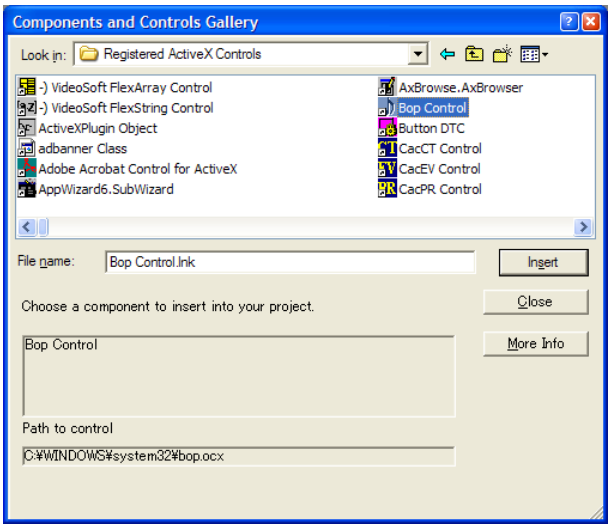
メニューから「Project」-「Add To Project」-「Components and Controls...」を選択する。



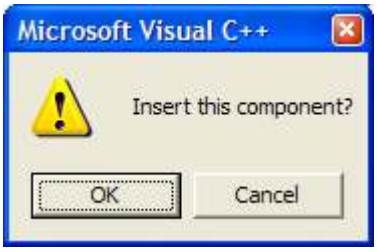
Registered ActiveX Controls ダブルクリックし、そのフォルダに移動する。



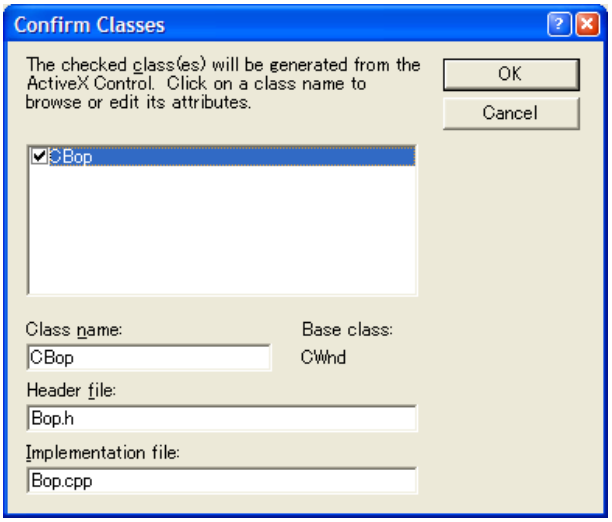
COM の一覧から bop Control を選択し、Insert ボタンを押す。



挿入するかどうかを確認してくるので、OK ボタンを押す。



確認画面が表示されるが、ラッパークラス名、ファイル名ともにデフォルトのままで構わないので、そのまま OK ボタンを押す。

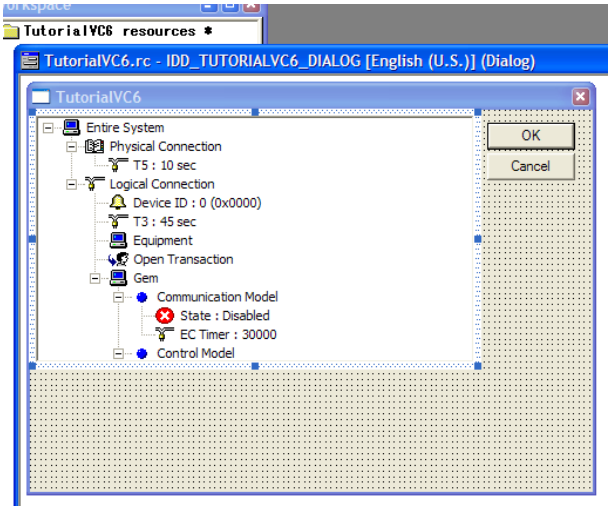


COM の一覧画面に戻ったら、Close ボタンを押して閉じる。コントロールの一覧に bop が追加されているのが確認できる。

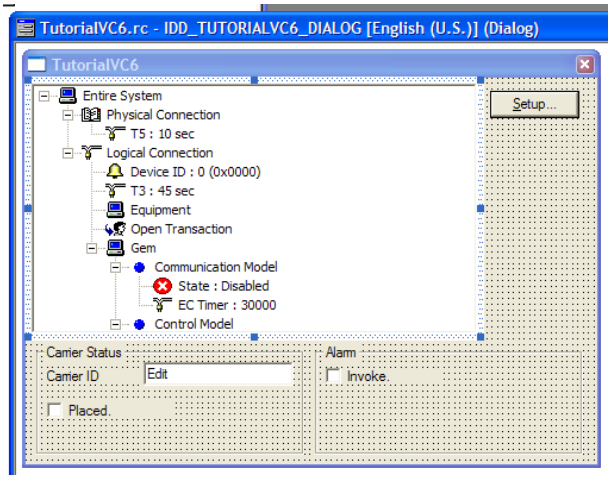
7.3.3 画面に貼り付ける

ダイアログボックス画面には「TODO: Place dialog controls here.」と書かれたテキストボックスが既に貼り付けられているが、これは必要ないので削除する。また Cancel ボタンも、ダイアログボックス右上隅にある×マークを押せばアプリケーションが終了するため不要である。これも削除する。

bop を画面に貼り付けると以下のようになる。



Visual Basic 6.0 編と同じように他のコントロールも貼り付ける。OK ボタンは Setup ボタンとして利用することにする。

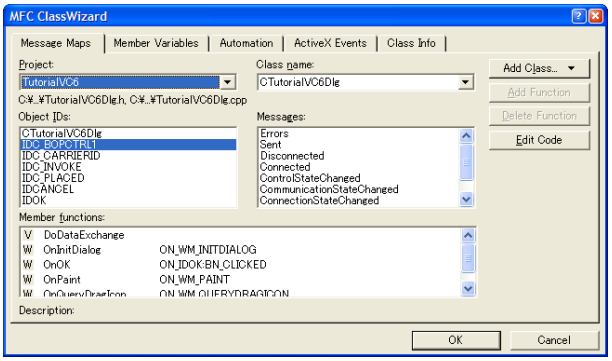


7.3.4 メンバ変数にマップする

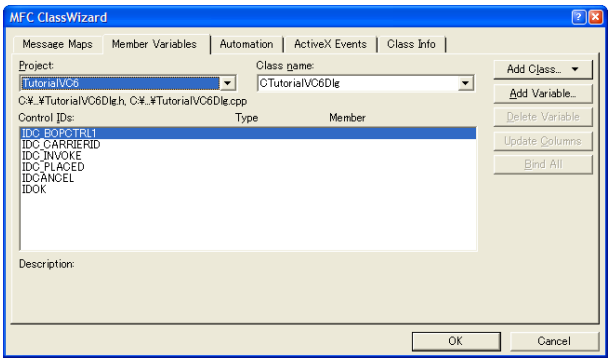
Visual C++ 6.0 ではコントロールを貼り付けただけでは名前がない。GetDlgItem() を使ってコントロールのポインタを取得することもできるが、もっと簡単な方法がある。メンバ変数としてしまうのである。

Visual C++ 6.0 ではメンバ変数を作成したり、イベントハンドラ関数を作成する場合など、ほとんど全ての作業に Class Wizard を使う。メニューから「View」-「Class Wizard...」を選択する。

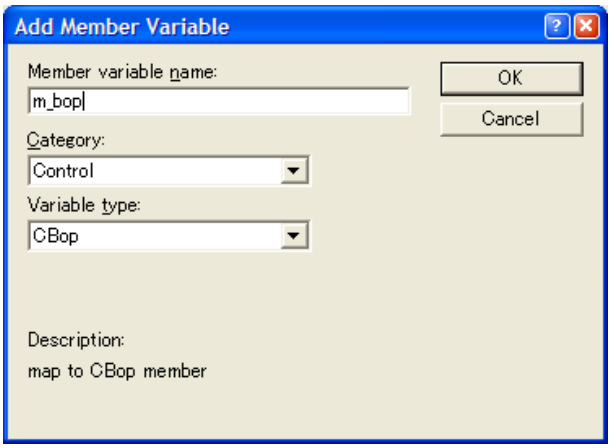
Class Wizard が開いたら、まずクラス名として CtutorialVC6Dlg が選択されているのを確認する。今回のチュートリアルではダイアログボックスクラスとアプリケーションクラスの二つしかないため非常にシンプルだが、MDI や SDI などでたくさんの画面を生成していくと、対象とするクラスを間違えてしまうことがある。



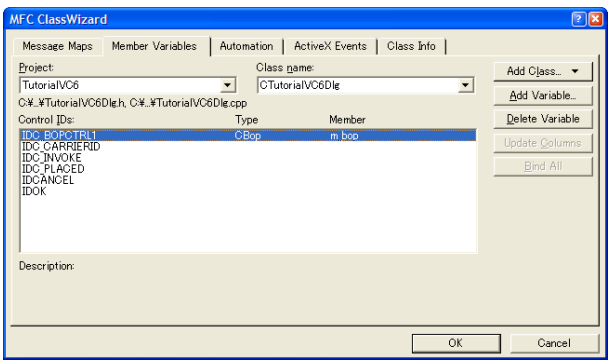
Member Variables タブを選択し、Add Variable ボタンを押す。



以下のようなダイアログボックスが開いたら、変数名を入力する。Microsoft の流儀としてメンバ変数は「m_」で始めることになっているので、それに従うことにしよう。ここでは「m_bop」という名前を入力した。入力し終わったら OK ボタンを押す。

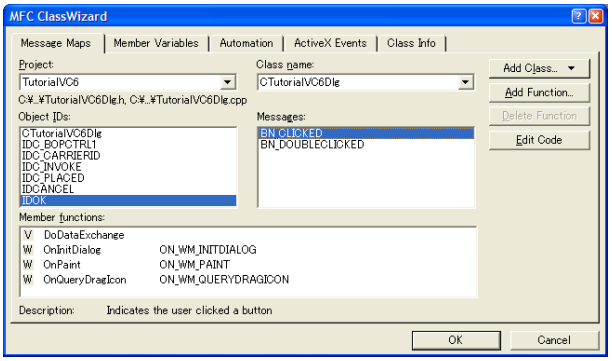


一覧にメンバ変数が表示されているのが確認できる。

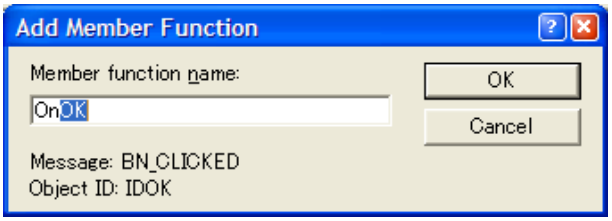


7.3.5 GEM 設定画面を作る

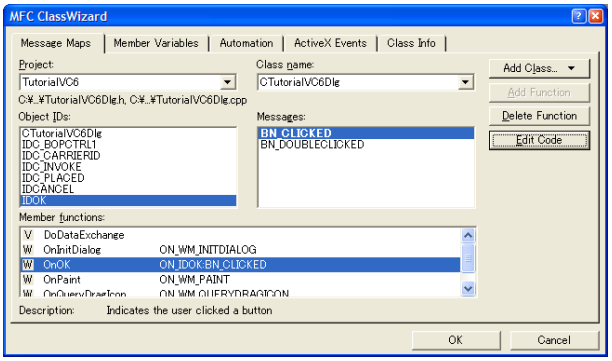
GEM の設定画面を追加するのにも Class Wizard を使う。Setup ボタンは IDOK という ID なので、これを選択し、BN_CLICKED（クリックされた）イベントのハンドラ関数を作成する。Add Function ボタンを押す。



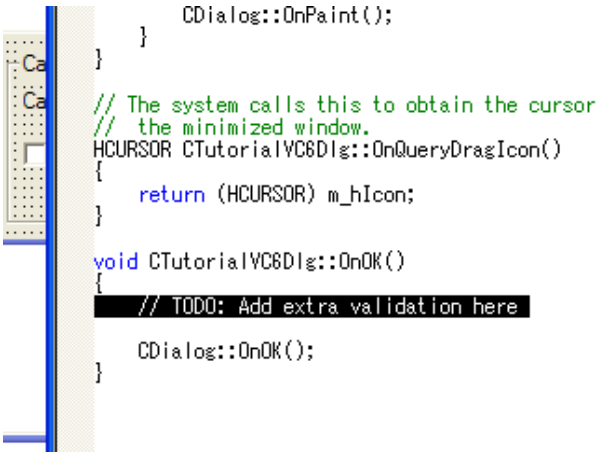
関数名を確認するダイアログボックスが表示されるが、デフォルトのままで構わないので OK ボタンを押す。



イベントハンドラ関数が作成される。Edit Code ボタンを押す。



イベントハンドラ関数にジャンプする。



```
        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();
}
```

ここを以下のように変更する。

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

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

CDialog::OnOK() は親クラスの仮想メンバ関数を呼び出しているのだが、この中でダイアログボックスを閉じる処理が書かれている。このためこの部分をコメントアウトしている。

Configure() メソッドの第一引数は以下のように書くこともできる。

```
   m_bop.Configure("",-1);
```

Basic ではポインタが使えないので「""」を使用したが、C++では NULL が使える。ただしどちらでも同じである。

7.3.6 設定を復元する

Class Wizard で、OnInitDialog() に以下のように記述する。OnInitDialog() は WM_INITDIALOG イベントのハンドラ関数である。

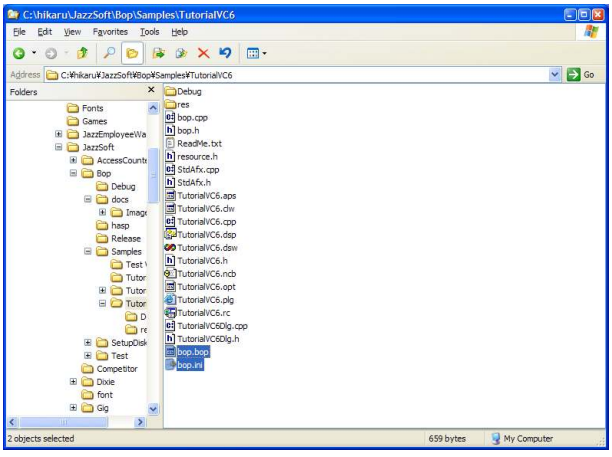
```
   m_bop.LoadIniFile();
   m_bop.Load();
```

追加後の OnInitDialog() は以下のようになる。

```
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
}
```

7.3.7 設定ファイルをコピーする

GEM の設定方法は Visual Basic 6.0 編と同じである。ここでは Visual Basic 6.0 編で作成した bop.bop と bop.ini をコピーして流用することにする。この二つのファイルをソースのフォルダ（.dsw のあるフォルダ）にコピーする。



7.3.8 設定ファイルを逆コンパイルする

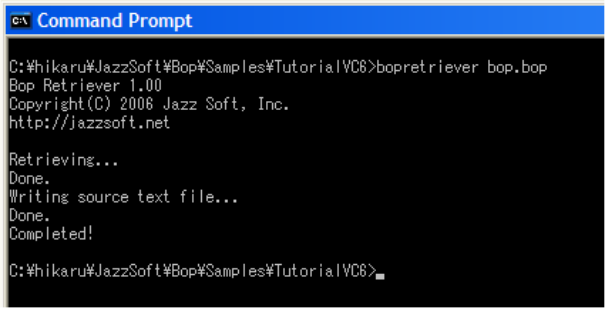
設定ファイル（.bop ファイル）はバイナリ形式であり、Configure() メソッドで編集も可能ではあるが、プログラマとしては ASCII 形式のテキストファイルの方がメンテナンスがしやすいと言える。このため.bop

ファイルを逆コンパイルし、.bopsource ファイルを生成してみよう。

まずスタートメニューのアクセサリから「コマンドプロンプト」を起動する。TutorialVC6 のフォルダに移動したら、下記のコマンドを実行する。

```
BopRetriever bop.bop
```

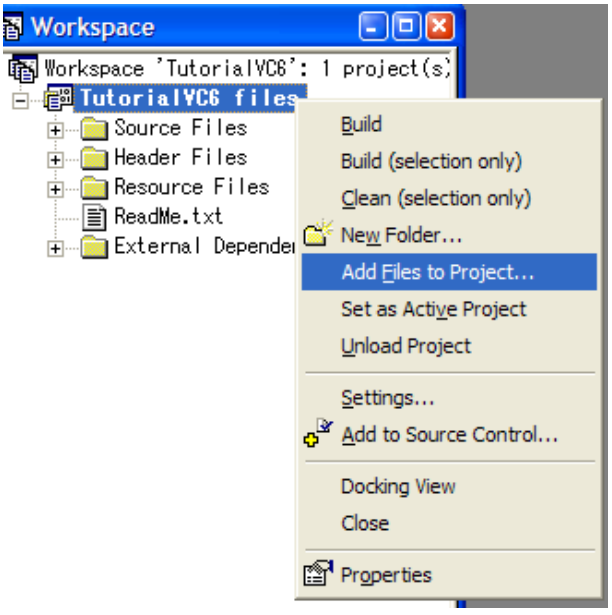
逆コンパイルされて bop.bopsource というファイルが生成される。



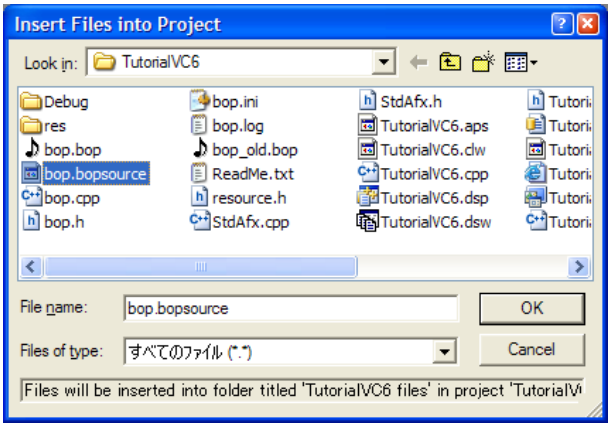
7.3.9 プロジェクトに追加する

.bopsource ファイルは Visual C++のプロジェクトに追加できる。これにより Visual Studio の統合開発環境で編集したり、.bop ファイルへのコンパイルが可能となる。

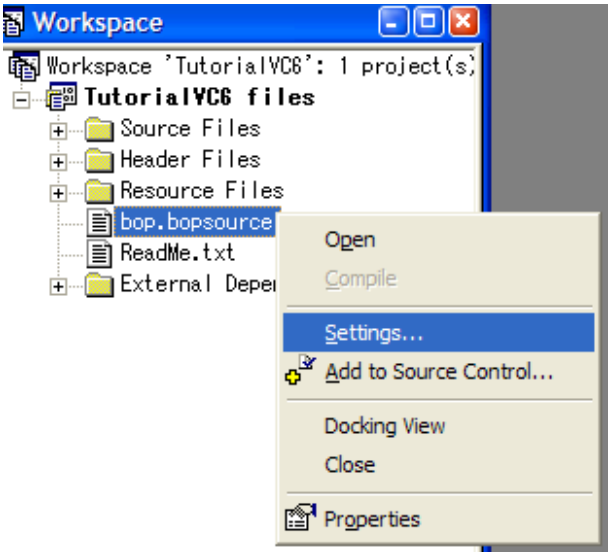
エクスプローラから TutorialVC6.dsw をダブルクリックしてプロジェクトを開く。Workspace の File View タブで TutorialVC6 のプロジェクトを右クリックし、「Add Files to Project...」を選択する。



「Files of type」コンボボックスを「すべてのファイル(*.*)」に変更し、bop.bopsource を選択して OK ボタンを押す。

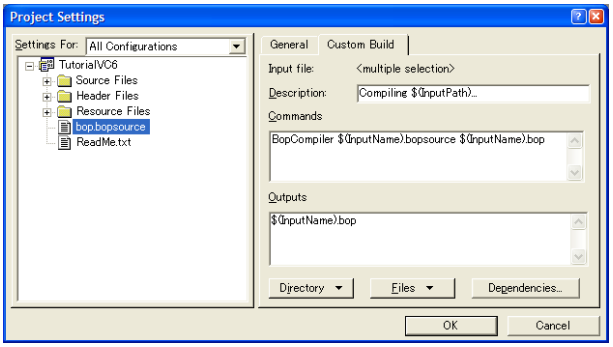


プロジェクトに bop.bopsource が追加されたので、右クリックして「Settings...」を選択する。

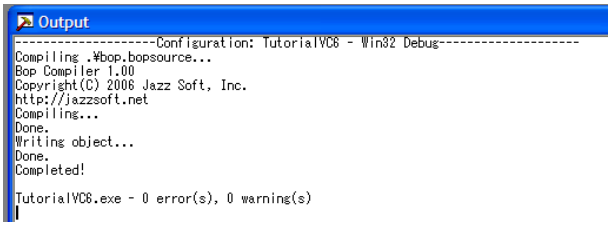


設定画面が開いたら、「Settings For」を「All Configurations」に変更した上で、下記のように入力する。

項目	値
Description	Compiling $(InputPath)...
Commands	BopCompiler $(InputName).bopsource $(InputName).bop
Outputs	$(InputName).bop

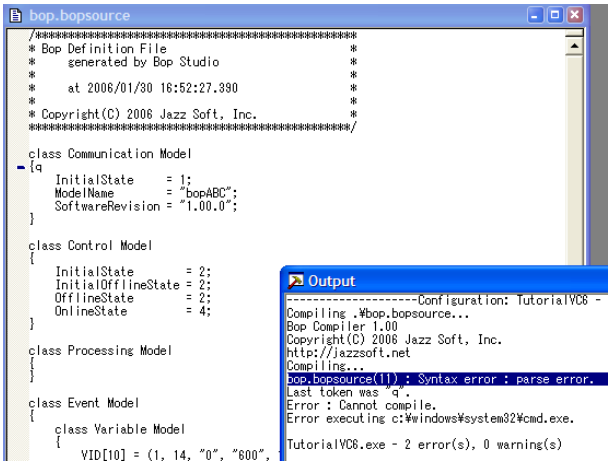


コンパイルを実行すると以下のように Output に出力される。



コンパイルエラーが発生すると、以下のように Output にエラーメッセ

ージが表示される。ここにはファイル名と行番号も表示されるので、ダブルクリックするとソースコードの問題箇所に自動的にジャンプする。



因みに上記の例では「{」の後に「q」という間違った文字が書かれているのでコンパイルエラーとなった。

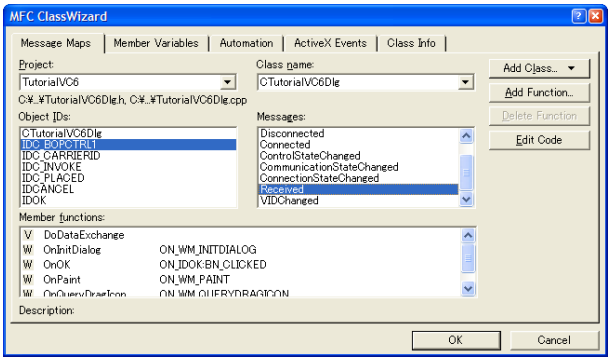
7.3.10 通信を有効化する

HSMS の通信を開始するには、PhysicalConnection プロパティに true をセットする。

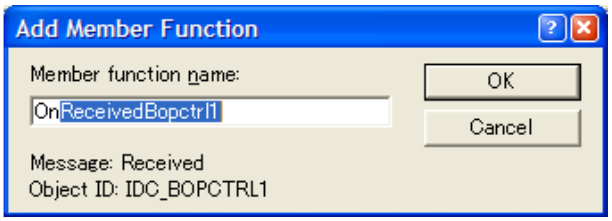
```
m_bop.LoadIniFile();
m_bop.Load();
m_bop.SetPhysicalConnection(true);
```

7.3.11 メッセージ処理

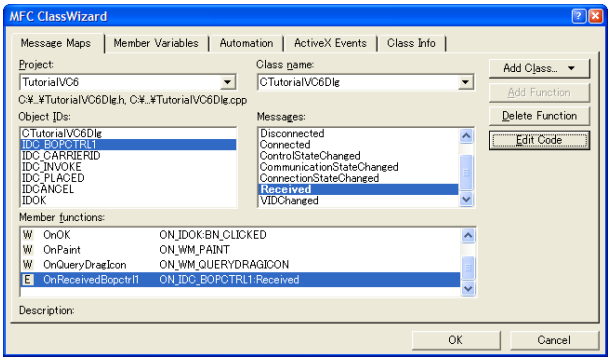
メッセージ受信処理を記述するのにも Class Wizard を使う。IDC_BOPCTRL1（bop のリソース ID 名）を選択し、Received イベントを選択して Add Function ボタンを押す。



ハンドラ関数名を確認するダイアログボックスが表示されるが、デフォルトの OnReceivedBopctrl1 で構わないので、そのまま OK ボタンを押す。



ハンドラ関数が作成されたので、Edit Code ボタンを押す。



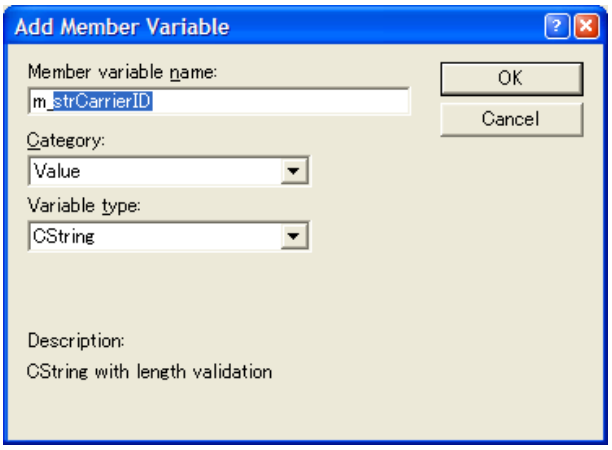
OnReceivedBopctrl1() にジャンプするので、以下のように記述する。

```
m bop.DefProc()
```

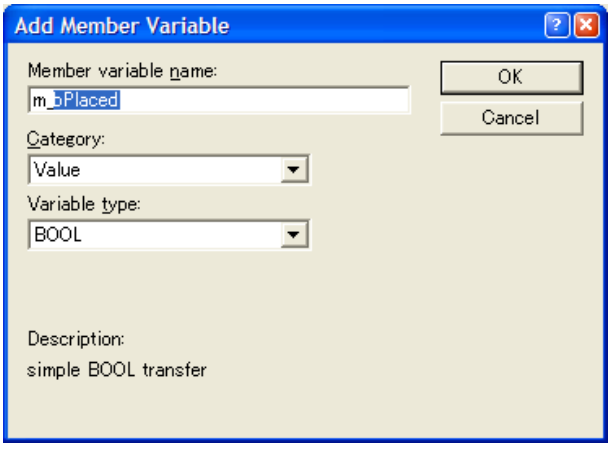
7.3.12 GEM イベントの送信

イベントを送信する処理は Visual Basic 6.0 とも Visual Basic .NET とも異なる。全てのプロパティへのアクセスをメソッドのように関数呼び出しとして記述する必要がある。

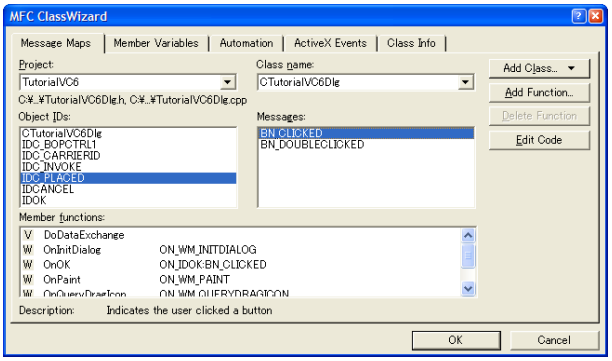
まずキャリア ID をメンバ変数にマップする。Class Wizard の Member Varialbes タブで IDC_CARRIERID を選択し、Add Variable ボタンを押す。表示されたダイアログボックスで変数名を入力し OK ボタンを押す。ここでは m_strCarrierID という名前にしている。



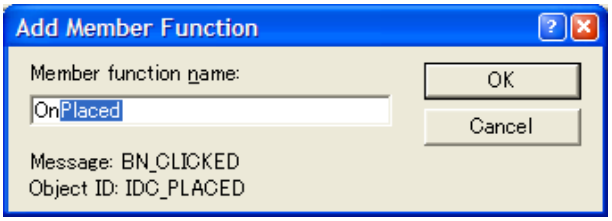
IDC_PLACED は以下のように命名する。



Message Maps タブで IDC_PLACED の BN_CLICKED を選択し、Add Function ボタンを押す。



関数名はデフォルトのままで構わないので、そのまま OK ボタンを押す。

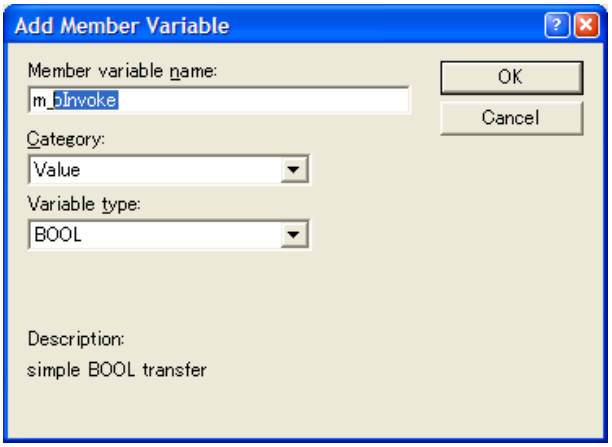


Class Wizard から OnPlaced()にジャンプし、以下のように記述する。

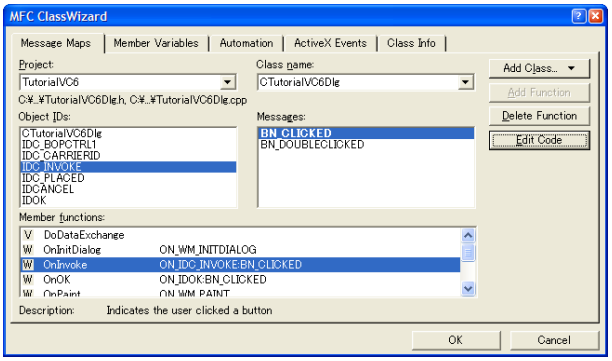
```
UpdateData();

m_bop.SetVIDValue(40,m_strCarrierID);

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

7.3.13 アラームの送信

アラームを送信する処理も若干異なるが、GEM イベントの時と似ている。まず Class Wizard で IDC_INVOKE に変数名をつける。



Message Maps タブで IDC_INVOKE のクリックイベントのハンドラ関数を作成しジャンプする。



以下のように記述する。

```
   UpdateData();

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

7.3.14 全ソースコード

App Wizard が自動生成したコードを除くと、Visual Basic 6.0 と同様に、非常に短い行数で記述できることが分かる。

```
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);
}
```

しかし C++をマスターするのは容易ではないので、自信がなければ無理

に格好つけずに Visual Basic でプログラミングするのを推奨する。

8 ActiveX コントロール・インターフェース

8.1 プロパティ

8.1.1 ALIDCode

■説明

ALID の分類コード。ALCD として扱われる。ALIDCode に値をセットしても、下位 7 ビットしか記録されない。ALCD の最上位ビットは、アラームの発生・解除の意味で使われるが、このビットは InvokeAlarm の引数で指定する。

ALID が登録されていない場合は値をセットすることはできない。登録されている ALID の一覧を得るには以下の方法を使う。

まず ALIDCount がいくつかを調べる。

```
lCount = .ALIDCount
```

ALIDCount は登録されている ALID の数を返すので、インデックスとして使用可能な値は 0 から (ALIDCount - 1) までとなる。これを IndexToALID を使って ALID に変換する。

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

ALID が得られたので ALIDCode と ALIDDescription にアクセスすることができる。

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

これを For 文で繰り返して列挙すればよい。全ソースコードを表示する。

```
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
```

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

型	説明
lALID	ALID。

■関連事項

ALIDCount, ALIDDescription, IndexToALID, InvokeAlarm

8.1.2 ALIDCount

■説明

登録されている ALID の総数。0 の場合は一つも登録されていないことを意味する。

ALIDCount は登録されている ALID の数を返すので、インデックスとして使用可能な値は 0 から (ALIDCount - 1) までとなる。これを IndexToALID を使って ALID に変換する。

■宣言

■Visual C++ 6

```
long GetALIDCount();
```

■Visual Basic 6

```
ALIDCount As Long
```

■特記事項

読み出し専用プロパティ。

■関連事項

ALIDCode, ALIDDescription, IndexToALID, InvokeAlarm, CEIDCount, VIDCount

8.1.3 ALIDDescription

■説明

ALID の説明。S5F1 アラーム報告送信 (ARS) では ALTX として送信される。

ALTX は SECS-II では最大 40 文字に制限されているが、bop ではこの制限はない。

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

型	説明
lALID	ALID。

8.1.4 CEIDCount

■説明

登録されている CEID の総数。

CEIDCount は登録されている CEID の数を返すので、インデックスとして使用可能な値は 0 から (CEIDCount - 1) までとなる。これを IndexToCEID を使って CEID に変換する。

■宣言

■Visual C++ 6

```
long GetCEIDCount();
```

■Visual Basic 6

```
CEIDCount As Long
```

■特記事項

読み出し専用プロパティ。

■関連事項

ALIDCount, VIDCount

8.1.5 CEIDDescription

■説明

CEID の説明。

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

型	説明
lCEID	CEID。

8.1.6 Communication

■説明

通信状態モデル。通信状態は以下のいずれかである。

名称	値	説明
Disabled	0	通信無効。
NotCommunicating	1	通信中断。
Communicating	2	通信実行中。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
Communication As Integer
```

8.1.7 ControlState

■説明

コントロール状態モデル。コントロール状態は以下のいずれかである。

名称	値	説明
EquipmentOffLine	0	装置オフライン。
AttemptOnLine	1	オンライン確立試行。
HostOffLine	2	ホストオフライン。
OnLineLocal	3	オンラインローカル。
OnLineRemote	4	オンラインリモート。

コントロール状態の状態遷移には許可されないものがある。このためオンラインとオフラインの切り替えは ControlStateSwitch を使う。許可される状態遷移については、コントロール状態モデルを参照のこと。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
ControlState As Integer
```

■関連事項

ControlStateSwitch, ControlStateChanged

8.1.8 ControlStateSwitch

■説明

オンライン／オフライン切り替えスイッチ。コントロール状態の状態遷移には許可されないものがある。このためオンラインとオフラインの切り替えは ControlStateSwitch を使う。許可される状態遷移については、コントロール状態モデルを参照のこと。

値	説明
true	オフラインからオンラインに移行する。
false	オンラインからオフラインに移行する。

実際に状態遷移した場合は ControlStateChanged イベントが発生する。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
ControlStateSwitch As Boolean
```

■特記事項

書き込み専用プロパティ。

■関連事項

ControlState, ControlStateChanged

8.1.9 DeviceID

■説明

デバイス ID。SessionID はメッセージヘッダの先頭 16 ビットのことである。DeviceID は SessionID の最上位ビットを除いた 15 ビットのことである。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
DeviceID As Long
```

■関連事項

SessionID

8.1.10 DiscardDuplicatedBlock

■説明

二重ブロックを除外するかどうかを指定する。true の場合は全く同一のメッセージを続けて受信したら、後から来た方のメッセージは無視される。false の場合は無視されず、Received イベントでアプリケーションに通知される。通常は true に設定しておくのがよい。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
DiscardDuplicatedBlock As Boolean
```

■関連事項

Received

8.1.11 Function

■説明

ファンクション番号。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
Function As Integer
```

■関連事項

Stream

8.1.12 HexDump

■説明

メッセージを 16 進数文字列で取得する。

SML に文字列をセットすると bop は文字列をコンパイルし、内部にデータ構造を構築する。このとき HexDump を読み出すとバイナリ構造に変換し、それを 16 進数文字列にしたものを返す。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
HexDump As String
```

■関連事項

SML

8.1.13 Host

■説明

ホスト側かクライアント側かを設定する。bop は GEM（装置）を実装するための製品なので、Host プロパティは常に false に設定する。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
Host As Boolean
```

8.1.14 IniFile

■説明

設定を保存する ini ファイル名。フルパス名か相対パス名で指定すると、そのフォルダに ini ファイルが作成されるが、ファイル名だけを指定すると Windows のフォルダに作成されてしまうので注意が必要である。カレントフォルダに作成したい場合は、

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

のように指定するとよい。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
IniFile As String
```

8.1.15 IPAddress

■説明

接続する IP アドレス。パッシブエンティティの場合は自分の IP アドレスなので指定する必要はない。アクティブエンティティの場合は接続先の IP アドレスを指定する。

プロパティ	パッシブ	アクティブ
IPAddress	不要	必要
PortNumber	不要	必要
LocalPortNumber	必要	必要（通常は 0）

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
IPAddress As String
```

■関連事項

PortNumber, LocalPortNumber

8.1.16 LocalPortNumber

■説明

ローカルポート番号。パッシブエンティティの場合はクライアントに対して公開するポート番号となる。

アクティブエンティティの場合は通常 0 を指定する。0 を指定すると空いているポートが自動的に選択される。0 以外を指定すると、そのポート番号が空いていなければ接続することができない。Windows では接続を切断しても、しばらく（数分間）はそのポート番号を占有する。このため 0 以外を指定した場合は数分間、接続ができなくなるので注意が必要である。

プロパティ	パッシブ	アクティブ
IPAddress	不要	必要
PortNumber	不要	必要
LocalPortNumber	必要	必要（通常は 0）

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
LocalPortNumber As Long
```

■関連事項

IPAddress, PortNumber

8.1.17 LogFileBakCount

■説明

ログファイルのバックアップファイル数。ログファイル名は以下のようになる。

ファイル名	説明
XXXXX.log	時間的に一番新しいログファイル。
XXXXX001.log	時間的に一つ前のバックアップファイル。
XXXXX002.log	時間的に二つ前のバックアップファイル。
...	...

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
LogFileBakCount As Integer
```

■関連事項

LogFileName, LogFileSize, LogFileEnableCommunication, LogFileEnable

8.1.18 LogFileEnable

■説明

ログファイルに記録するかどうかを指定する。このプロパティが true のときはファイルに記録し、false のときは記録しない。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
LogFileEnable As Boolean
```

■関連事項

LogFileName, LogFileSize, LogFileEnableCommunication, LogFileBakCount

8.1.19 LogFileEnableCommunication

■説明

通信部分のログをファイルに記録するかどうかを指定する。このプロパティが true のときはファイルに記録し、false のときは記録しない。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
LogFileEnableCommunication As Boolean
```

■関連事項

LogFileName, LogFileSize, LogFileEnable, LogFileBakCount

8.1.20 LogFileName

■説明

ログファイル名。ログファイル名を指定するときは、拡張子を付加しない。自動的に拡張子「.log」が付加される。

ファイル名	説明
XXXXX.log	時間的に一番新しいログファイル。
XXXXX001.log	時間的に一つ前のバックアップファイル。
XXXXX002.log	時間的に二つ前のバックアップファイル。
...	...

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
LogFileName As String
```

■関連事項

LogFileName, LogFileSize, LogFileEnableCommunication, LogFileBakCount

8.1.21 LogFileSize

■説明

ログファイルのサイズ。このファイルサイズを超えるとバックアップファイルが作成される。

単位は KB（キロバイト）となる。このため例えば 1024 と指定すると、1MB（メガバイト）の意味である。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
LogFileSize As Long
```

■関連事項

LogFileName, LogFileBakCount, LogFileBakCount, LogFileEnableCommunication

8.1.22 LogicalConnection

■説明

論理接続のタイプ。以下のものが定義されている。

値	説明
0	基本モデル。特に何か処理を行わない。
1	GEM モデル。

GEM を実装するために bop を使っているので、LogicalConnection は常に 1 となる。[4]

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
LogicalConnection As Integer
```

8.1.23 LogicalConnectionFileName

■説明

論理接続の設定内容を保存するファイル名。拡張子は指定しないこと（自動的に「.bop」が付加される）。デフォルトでは bop.bop（プロパティの値としては「./bop」）となる。

GEM ではレポート定義などを不揮発性の媒体に記憶するよう求めている。このため LogicalConnectionFileName を使用する必要がある。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
LogicalConnectionFileName As String
```

8.1.24 Node

■説明

ノード指定子。ノードとはメッセージ構造中の操作対象である。

パソコンでハードディスク内のフォルダは「ツリー構造」にすることができる。フォルダを作ってその中にファイルを入れるのである。ファイルだけではなく、フォルダの中にフォルダを作ることもできる。ツリー構造では、フォルダは「枝」、ファイルは「葉」に相当する。

これと同様に、SML のデータ構造もツリー構造にすることが可能である。リストが「枝」、その他のアイテムが「葉」である。ノードは「枝」「葉」の位置を特定するための識別子である。

ノードは“/”（スラッシュ）とノード番号で構成される。ノードが“”（空）の場合はルートが指定されたとみなされる。一般にルートはリスト型であることが多いが、他の型も指定できる。子ノードがある場合は必ずルートがリスト型となる。

例えば以下のような SML があるとする。

```
{
  <a 'Kelly'>
  {
    <a 'Brenda'>
    {
      <a 'Donna'>
    }
  }
  <a 'Valerie'>
  {
    {
      {
        <a 'Andrea'>
      }
    }
  }
}
```

ノードを特定しやすいよう、番号を付けてみよう。

```
1 {
  1 <a 'Kelly'>
  2 {
    1 <a 'Brenda'>
    2 {
      1 <a 'Donna'>
      }
    }
  3 <a 'Valerie'>
  4 {
    1 {
```

[4] bop の現在のバージョンでは LogicalConnection プロパティは常に 1 を指定すること。

```
      1 {
        1 <a'Andrea'>
        }
      }
    }
  }
```

Kelly, Brenda, Donna, Valerie, Andrea をノード位置の記述であらわすと以下のようになる。ノードのネスティングレベルに制限はない。

値	ノード
Kelly	1
Brenda	2/1
Donna	2/2/1
Valerie	3
Andrea	4/1/1/1

ノードが配列になっている場合に、特定の要素を取り出すには[]を使ってインデックスを指定する。例えば次のような SML があるとする。

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

この 2 番目の要素「3.14」を取り出したい場合はインデックスを指定する。インデックスは 0 から始まるので 2 番目の要素なら 1 となる。

```
.Node = "1[1]"
```

これを読み出すと以下のように特定の要素だけが返る。

```
"3.14"
```

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
Node As String
```

■関連事項

NodeCount, NodeType, NodeValue, NodeValueHex

8.1.25 NodeCount

■説明

ノードのアイテム数。ノードは配列になっているものがある。リストはその最たる例である。

下の例ではリストは 3 つの子ノードを持っているため、NodeCount は 3 である。リストの要素数があらかじめ分からない場合などは、まず NodeCount を読み出して回数分ループさせるとよい。

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

次の例では u2 アイテムは 1 個、u4 アイテムは 2 個、f8 アイテムは 3 個の NodeCount である。

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

次のリストの NodeCount は 0 である。

```
{
}
```

■宣言

■Visual C++ 6

```
long GetNodeCount();
```

■Visual Basic 6

```
NodeCount As Long
```

■特記事項

読み出し専用プロパティ。

8.1.26 NodeType

■説明

ノードの種類。以下のいずれかである。

ノードの型	値	説明
NodeTypeList	1	リスト
NodeTypeBinary	2	バイナリ
NodeTypeBoolean	3	ブーリアン
NodeTypeAscii	4	アスキー文字列
NodeTypeJis	5	JIS-8 コード
NodeTypeLong8	6	8 バイト符号付き整数
NodeTypeChar	7	1 バイト符号付き整数
NodeTypeShort	8	2 バイト符号付き整数
NodeTypeLong	9	4 バイト符号付き整数
NodeTypeDouble	10	8 バイト浮動小数点数
NodeTypeFloat	11	4 バイト浮動小数点数
NodeTypeDWord8	12	8 バイト符号なし整数
NodeTypeByte	13	1 バイト符号なし整数
NodeTypeWord	14	2 バイト符号なし整数
NodeTypeDWord	15	4 バイト符号なし整数
NodeTypeAscii2	16	2 バイトアスキー文字列

NodeType を読み出したときに、値が 0 の場合は「無効な型」であることを意味する。

■宣言

■Visual C++ 6

```
short GetNodeType();
```

■Visual Basic 6

```
NodeType As Integer
```

■特記事項

読み出し専用プロパティ。

8.1.27 NodeValue

■説明

ノードの値。バイナリ型、数値型（符号付き整数、符号なし整数、浮動小数点数）は 10 進数表現の文字列に変換される。ブーリアン型の true は"1"、false は"0"に変換される。

アイテムが配列（NodeCount が 1 より大きい）の場合、各要素はスペースコード 1 文字で区切られる。例えば次のような SML のメッセージがあるとする。

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

この u2 アイテムを読み出すには Node に"1"を指定する。

```
.Node = "1"
```

読み出すと以下のような文字列が返る。

```
"333 444 555"
```

配列の特定の要素だけ読み出したい場合は、[]でインデックスを指定する。

```
.Node = "1[1]"
```

これを読み出すと以下のように特定の要素だけが返る。

```
"444"
```

■宣言

■Visual C++ 6

```
BSTR GetNodeValue();
```

■Visual Basic 6

```
NodeValue As String
```

■特記事項

読み出し専用プロパティ。

■関連事項

NodeValueHex

8.1.28 NodeValueHex

■説明

ノードの値の 16 進数表現。例えば次のような SML のメッセージがあるとする。

```
<u2 254>
```

これを読み出すと以下のような文字列が返る。

```
"fe"
```

■宣言

■Visual C++ 6

```
BSTR GetNodeValueHex();
```

■Visual Basic 6

```
NodeValueHex As String
```

■特記事項

読み出し専用プロパティ。

■関連事項

NodeValue

8.1.29 OfflineRequest

■説明

S1F15 オフライン要求(ROFL)を受け入れるかどうかを指定する。このプロパティが true の時はオフライン要求を受け入れ、S1F16 オフライン要求確認(OFLA)で OFLACK=0 を返す。false の時は要求を拒否し、OFLACK=1 を返す。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
OfflineRequest As Boolean
```

■関連事項

OnlineRequest

8.1.30 OnlineRequest

■説明

S1F17 オンライン要求(RONL)を受け入れるかどうかれるかどうかを指定する。このプロパティが true の時はオンライン要求を受け入れ、S1F18 オンライン要求確認(ONLA)で ONLACK=0 を返す。false の時は要求を拒否し、ONLACK=1 を返す。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
OnlineRequest As Boolean
```

■関連事項

OfflineRequest

8.1.31 PassiveEntity

■説明

パッシブエンティティ（サーバ）か、アクティブエンティティ（クライアント）かを指定する。このプロパティが true の時はパッシブエンティティ、false の場合はアクティブエンティティとなる。

値	説明
true	パッシブエンティティ
false	アクティブエンティティ

装置がパッシブエンティティとなるか、アクティブエンティティとなるかについての明確な規定はない。しかし装置側がパッシブエンティティとなる方が理にかなっている場合が多いので、昨今の解釈では装置をパッシブエンティティとすることが多い。bop はどちらにも設定することができる。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
PassiveEntity As Boolean
```

8.1.32 PhysicalConnection

■説明

物理接続モデルが有効かどうかを指定する。このプロパティが true の時は物理接続モデルが有効となり、false の時は無効となる。

このプロパティを true にすることで通信ポートがオープンされ、通信が可能になる。ただし実際に通信ポートが開かれるのは、プロパティを true にセットした瞬間ではなく、直後である。このため以下のようなプログラムを書いても、ポートがオープンできたかどうかを知ることはできない。

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

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
PhysicalConnection As Boolean
```

8.1.33 PortNumber

■説明

接続するポート番号。パッシブエンティティの場合は相手に接続しにいくわけではないので、指定する必要はない（無視される）。アクティブエンティティの場合は接続先のポート番号を指定する。

プロパティ	パッシブ	アクティブ
IPAddress	不要	必要
PortNumber	不要	必要
LocalPortNumber	必要	必要（通常は 0）

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
PortNumber As Long
```

■関連事項

IPAddress, LocalPortNumber

8.1.34 PType

■説明

HSMS の P タイプ。メッセージの内容が SECS-II の場合は 0 となる。ただし現時点で 0 以外の P タイプは規定されていないので、0 以外はエラーである。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
PType As Integer
```

8.1.35 Reply

■説明

操作対象となっているメッセージの返信部分かを選択する。Received イベントでメッセージを受信した場合、WorkSpace0 にセットされる。受信メッセージを解析したい場合は、そのまま処理を行うことができる。受信メッセージに対して返信したい場合には、Reply を true にセットし、返信部分を編集する。この後にイベントハンドラ関数から DefProc を呼び出すと、返信が必要なメッセージの場合は返信部分に書かれた返信メッセージを送信する。

Received イベントハンドラから返信するのではなく、通常の処理からメッセージを送信したい場合は、WorkSpace0 の Reply=false にメッセージを作成する。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
Reply As Boolean
```

■関連事項

WorkSpace, Send, Received

8.1.36 SessionID

■説明

セッション ID。SessionID はメッセージヘッダの先頭 16 ビットのことである。DeviceID は SessionID の最上位ビットを除いた 15 ビットのことである。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
SessionID As Long
```

■関連事項

DeviceID

8.1.37 SML

■説明

メッセージの文字列表現。SECS-II メッセージはバイナリ構造となっており、このままでは人間が理解できない。このため読解しやすいよう、アスキー文字列で表現する。

SML 表現	説明
l	リスト
b	バイナリ
bool	ブーリアン
a	アスキー文字列
j	JIS-8
a2	2 バイトアスキー文字列
i8	8 バイト符号付き整数
i1	1 バイト符号付き整数
i2	2 バイト符号付き整数
i4	4 バイト符号付き整数
f8	8 バイト浮動小数点数
f4	4 バイト浮動小数点数
u8	8 バイト符号なし整数
u1	1 バイト符号なし整数
u2	2 バイト符号なし整数
u4	4 バイト符号なし整数

SML に文字列をセットすると bop は文字列をコンパイルし、内部にデータ構造を構築する。このとき HexDump を読み出すとバイナリ構造に変換し、それを 16 進数文字列にしたものを返す。SML を読み出すと bop 内部のデータ構造から文字列を生成する。

メッセージを作成する場合は、まずメッセージの SML 文字列を作成し、それを SML にセットする。SML の構文の詳細については SML リファレンスを参照のこと。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
SML As String
```

■関連事項

HexDump, SML リファレンス

8.1.38 Stream

■説明

ストリーム番号。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
Stream As Integer
```

■関連事項

Function

8.1.39 SType

■説明

HSMS の S タイプ。S タイプは以下のものが規定されている。

S タイプ	意味	説明
0	データメッセージ	ストリーム･ファンクションを持つ通常の SECS-II メッセージ
1	Select.req	セレクト要求。
2	Select.rsp	セレクト要求応答。
3	Deselect.req	ディセレクト要求。
4	Deselect.rsp	ディセレクト要求応答。
5	Linktest.req	リンクテスト要求。
6	Linktest.rsp	リンクテスト要求応答。
7	Reject.req	リジェクト要求。
9	Separate.req	セパレート要求。

HSMS-SS では Deselect.req、Deselect.rsp を使用しない。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
SType As Integer
```

8.1.40 SystemBytes

■説明

システムバイト。システムバイトは SECS ヘッダ 10 バイトのうち、最後の 4 バイトである。この値は返信を待っている（オープンな）トランザクションにおいてユニーク（一意）であれば何でも構わないが、通常は一次メッセージを送信するごとに+1 ずつ加算していくようにするとよい。

二次メッセージの SystemBytes は一次メッセージと同一でなければならない。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
SystemBytes As Long
```

8.1.41 T1

■説明

SECS-I の T1 タイムアウト。単位はミリ秒。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
T1 As Integer
```

■特記事項

このプロパティは SECS-I に付随する項目なので未使用となっている。

8.1.42 T2

■説明

SECS-I の T2 タイムアウト。単位はミリ秒。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
T2 As Integer
```

■特記事項

このプロパティは SECS-I に付随する項目なので未使用となっている。

8.1.43 T3

■説明

T3 タイムアウト。別名、トランザクションタイムアウト。単位は秒。

一次メッセージを送信してから二次メッセージが返信されるまでの時間。この時間以上経過すると T3 タイムアウトとなり、装置は S9F9 トランザクションタイマタイムアウト(TIN)を送信する。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
T3 As Integer
```

8.1.44 T4

■説明

SECS-I の T4 タイムアウト。単位は秒。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
T4 As Integer
```

■特記事項

このプロパティは SECS-I に付随する項目なので未使用となっている。

8.1.45 T5

■説明

HSMS の T5 タイムアウト。別名、接続セパレーションタイムアウト。単位は秒。

このプロパティはアクティブエンティティの場合のみ関係する。接続に失敗した場合や、切断した場合に、この時間以上は待たなければならない。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
T5 As Integer
```

8.1.46 T6

■説明

HSMS の T6 タイムアウト。別名、コントロールトランザクションタイムアウト。単位は秒。

コントロールメッセージ（S タイプが 0 以外のメッセージ）で、リクエストを送信してから、レスポンスを受信するまでの時間。T3 タイムアウトはデータメッセージに対してのトランザクションを監視するが、その

コントロールメッセージ版だといえる。

具体的には、以下の時間がタイムアウトになったタイミングで発生する。

S タイプ	説明
1	Select.req を送信してから Select.rsp を受信するまでの時間。
3	Deselect.req を送信してから Deselect.rsp を受信するまでの時間。
5	Linktest.req を送信してから Linktest.rsp を受信するまでの時間。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
T6 As Integer
```

8.1.47 T7

■説明

HSMS の T7 タイムアウト。別名、NOT SELECTED タイムアウト。単位は秒。

HSMS で TCP/IP レベルでは接続はしたものの、Selected 状態に移行しないで放置しておくと、設定した時間が経過した後に切断される。また Deselect.req で Selected 状態から Not Selected 状態に移行しても、再び Selected 状態に移行しないで放置しておくと、やはり切断される。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
T7 As Integer
```

8.1.48 T8

■説明

HSMS の T8 タイムアウト。別名、ネットワークキャラクタ間タイムアウト。単位は秒。

HSMS 接続が切れていなくても、１つのメッセージ受信の途中でしばらくの間データが途切れると、メッセージの続きなのかどうかの判断ができない。T8 タイマー以上経過すると、「通信の失敗」とみなされ、結果的に切断されることになる。

SECS-I の T1 タイムアウトに似ている。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
T8 As Integer
```

8.1.49 Verification

■説明

受信したメッセージを検証した結果。以下のいずれかとなる。

結果	値	説明
Correct	0	SEMI E.5 (SECS-II) に準拠している。
UserDefined	1	ユーザ定義メッセージ。
Incorrect	2	準拠していない。
IncorrectAndReply	3	準拠してはいないが S9F7 ではなく 2 次メッセージで応答する。
NoWBit	4	W ビットが必要なのに無い。
WBit	5	W ビットが不要なのにある。
WrongDirection	6	メッセージ方向が逆。
UnrecognizedStream	7	ストリームが定義されていない。
UnrecognizedFunction	8	ファンクションが定義されていない。

メッセージを受信すると bop 内部でメッセージ構造が検証される。その結果が Verification にセットされ、Received イベントが発生する。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
Verification As Integer
```

■関連事項

Received イベント

8.1.50 VIDCount

■説明

登録されている VID の総数。

VIDCount は登録されている VID の数を返すので、インデックスとして使用可能な値は 0 から (VIDCount - 1) までとなる。これを IndexToVID を使って VID に変換する。

■宣言

■Visual C++ 6

```
long GetVIDCount();
```

■Visual Basic 6

```
VIDCount As Long
```

■特記事項

読み出し専用プロパティ。

■関連事項

ALIDCount, CEIDCount

8.1.51 VIDDefault

■説明

VID のデフォルト値。S2F30 装置定数名リスト(ECN) の ECDEF。

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

型	説明
lVID	VID。

■関連事項

S2F30 装置定数名リスト(ECN)

8.1.52 VIDDescription

■説明

VID の説明。このプロパティは VIDType によって扱いが若干異なり、SVID の場合は S1F12 状態変数名リスト応答(SVNRR) の SVNAME として扱われる。ECID の場合は S2F30 装置定数名リスト(ECN) の ECNAME として扱われる。

DVID の場合には DVNAME として扱いたいところだが、あいにく GEM では S6F4 も S6F8 も定義されていないので、bop でも自動処理されることはない。しかしユーザが VIDDescription を使用して S6F4 や S6F8 に対応させることは可能である。

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

型	説明
lVID	VID。

■関連事項

S1F12 状態変数名リスト応答(SVNRR), S2F30 装置定数名リスト(ECN)

8.1.53 VIDMax

■説明

VID の最大値。S2F30 装置定数名リスト(ECN) の ECMAX。

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

型	説明
lVID	VID。

■関連事項

S2F30 装置定数名リスト(ECN)

8.1.54 VIDMin

■説明

VID の最小値。S2F30 装置定数名リスト(ECN) の ECMIN。

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

型	説明
lVID	VID。

■関連事項

S2F30 装置定数名リスト(ECN)

8.1.55 VIDNodeType

■説明

VID の SECS-II のノード型。以下のいずれかとなる。

ノードの型	値	説明
NodeTypeList	1	リスト
NodeTypeBinary	2	バイナリ
NodeTypeBoolean	3	ブーリアン
NodeTypeAscii	4	アスキー文字列
NodeTypeJis	5	JIS-8 コード
NodeTypeLong8	6	8 バイト符号付き整数
NodeTypeChar	7	1 バイト符号付き整数
NodeTypeShort	8	2 バイト符号付き整数
NodeTypeLong	9	4 バイト符号付き整数
NodeTypeDouble	10	8 バイト浮動小数点数
NodeTypeFloat	11	4 バイト浮動小数点数
NodeTypeDWord8	12	8 バイト符号なし整数
NodeTypeByte	13	1 バイト符号なし整数
NodeTypeWord	14	2 バイト符号なし整数
NodeTypeDWord	15	4 バイト符号なし整数
NodeTypeAscii2	16	2 バイトアスキー文字列

VIDNodeType を読み出したときに、値が 0 の場合は「無効な型」であることを意味する。

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

型	説明
lVID	VID。

■関連事項

NodeType

8.1.56 VIDRawValue

■説明

VID の生 SML 文字列。ここに SML 文字列をセットすると、S6F11 イベントレポート送信 (ERS) でイベントを送信する際に、レポートに従ってメッセージが組み立てられる。このとき個々の VID の値はこの VIDRawValue が使われる。SML にはストリーム、ファンクションは記述できない。

bop では文法エラーや VIDNodeType をチェックしないので、正しい SML をセットする必要がある。もし VIDNodeType の型に従って値をセットしたい場合は、VIDValue を使用する。

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

型	説明
lVID	VID。

■関連事項

SML, VIDValue, VIDNodeType

8.1.57 VIDType

■説明

VID のタイプ。以下のうちいずれか１つとなる。

タイプ	説明
1	ECID。
2	SVID。
4	DVID。

上記以外のタイプを指定することはできない。

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

型	説明
lVID	VID。

8.1.58 VIDUnit

■説明

VID の単位。S2F30 装置定数名リスト (ECN) の UNITS。

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

型	説明
lVID	VID。

■関連事項

S2F30 装置定数名リスト (ECN)

8.1.59 VIDValue

■説明

VID の値。VIDNodeType で指定された型に従って、VIDRawValue に SML が生成される。

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

型	説明
lVID	VID。

8.1.60 ViewStyle

■説明

画面の表示スタイル。以下のうちいずれかを指定する。

スタイル	値	説明
RedrawNone	0	何も表示しない。
RedrawHsms	1	HSMS の物理接続状態だけを表示。
RedrawGem	2	GEM の論理接続状態だけを表示。
RedrawNormal	3	全て表示。

■宣言

■Visual C++ 6

```
short GetViewStyle();
```

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

■Visual Basic 6

```
ViewStyle As Integer
```

8.1.61 WaitBit

■説明

W（ウェイト）ビット。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
WaitBit As Boolean
```

8.1.62 WorkSpace

■説明

操作対象となっているメッセージの作業領域。bop には 3 つの WorkSpace が用意されている。それぞれに「一次メッセージ用」の Reply=true と「二次メッセージ用」の Reply=false があるので、全部で 6 つのメッセージを扱うことができる。しかし基本的には WorkSpace0 だけで処理できるようになっている。

WorkSpace	Reply	用途
0	false	送受信メッセージ。
0	true	受信メッセージの返信。
1	false	送信済みメッセージ。
1	true	自由に使用可能。
2	false	自由に使用可能。
2	true	自由に使用可能。

メッセージを受信して Received イベントで通知される際には、WorkSpace0 の Reply=false に受信したメッセージが格納される。受信したメッセージが二次メッセージだとしても、Reply=false に格納されているので注意が必要である。イベントが通知された瞬間は WorkSpace0 の Reply=false が選択されているが、イベントハンドラから bop に処理が戻ると、以前に選択されていた WorkSpace と Reply に戻る。

一次メッセージの場合は、この時点で「推奨される返信メッセージ」も Reply=true に格納されている。このまま返信する場合は、Reply を true に切り替えて Send を呼ぶだけでよいが、内容を編集することもできる。

一次メッセージを送信する場合は WorkSpace0 の Reply=false を使用する。送信が完了すると Sent イベントで通知されるが、その際には自動的に WorkSpace1 の Reply=false が選択されている。

■宣言

■Visual C++ 6

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

■Visual Basic 6

```
WorkSpace As Integer
```

■関連事項

Reply, Send, Received

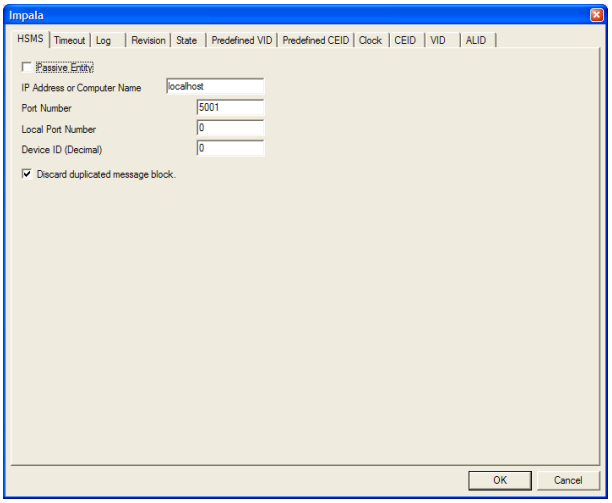
8.2 メソッド

8.2.1 Configure

■説明

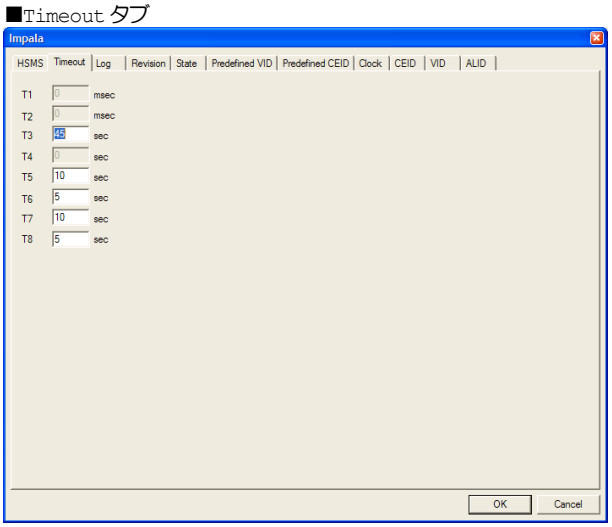
設定画面を表示する。

■HSMS タブ



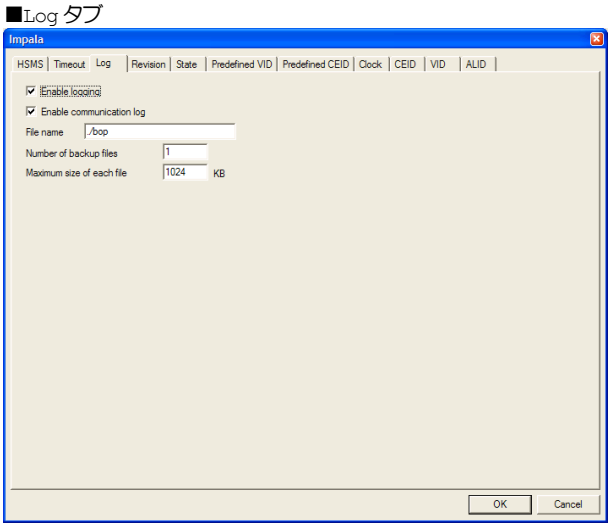
項目	説明
Passive Entity	チェックが付いているときはパッシブエンティティ（サーバ）を表す。付いていないときはアクティブエンティティ（クライアント）となる。
IP Address or Computer Name	IP アドレス（xxx.xxx.xxx.xxx）か、コンピュータ名を指定する。パッシブエンティティの場合は指定する必要がないので、この項目は入力できなくなる。
Port Number	相手のポート番号。パッシブエンティティの場合は接続してくるまで分からないので、この項目は入力できなくなる。
Local Port Number	ローカル（自分の）ポート番号。アクティブエンティティの場合に 0 を指定すると、自動的に空いているポート番号が割り当てられる。0 以外を指定すると、数分間は再接続できなくなることがある。
Device ID (Decimal)	デバイス ID。0～32767 の範囲で指定する。
Discard duplicated message block.	チェックが付いているときは同一メッセージが連続して来た場合に（二重ブロック）、後から来た方のメッセージを無視するようになる。

■Timeout タブ



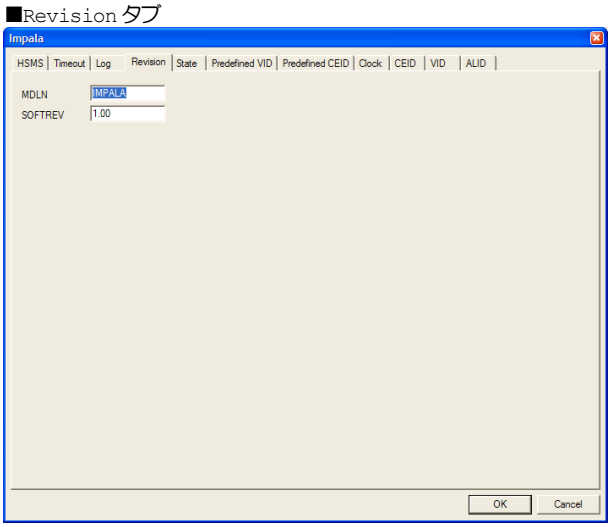
項目	説明
T1	（未使用）
T2	（未使用）
T3	T3 タイムアウト。単位は秒。
T4	（未使用）
T5	T5 タイムアウト。単位は秒。
T6	T6 タイムアウト。単位は秒。
T7	T7 タイムアウト。単位は秒。
T8	T8 タイムアウト。単位は秒。

■Log タブ



項目	説明
Enable logging	チェックが付いているときはログファイルに書き出す。チェックが付いていないときは以下の項目が入力できなくなる。
Enable communication log	チェックが付いているときは通信ログを記録する。
File name	ログファイル名。拡張子はつけないこと。自動的に拡張子.log が付加される。
Number of backup files	バックアップファイルの数。バックアップファイル名は、 ファイル名 001.log ファイル名 002.log のようになる。
Maximum size of each file	ログファイルの最大サイズ。このサイズを超えるとバックアップファイルが作られる。単位はキロバイト。

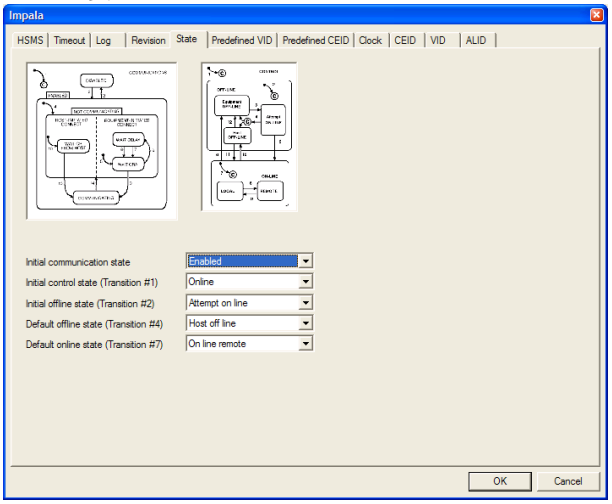
■Revision タブ



項目	説明
MDLN	S1F13 通信確立要求 (CR) などのメッセージに含まれる MDLN の文字列。装置のモデル名。最大 6 バイトまで。

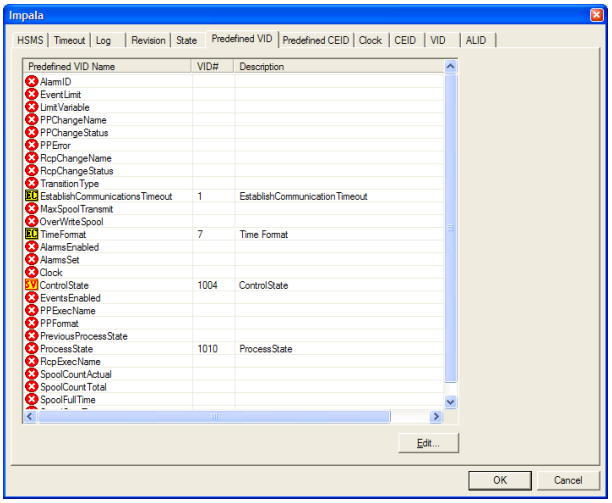
SOFTREV	SOFTREVの文字列。リビジョン（バージョン）番号。最大6バイトまで。

■State タブ

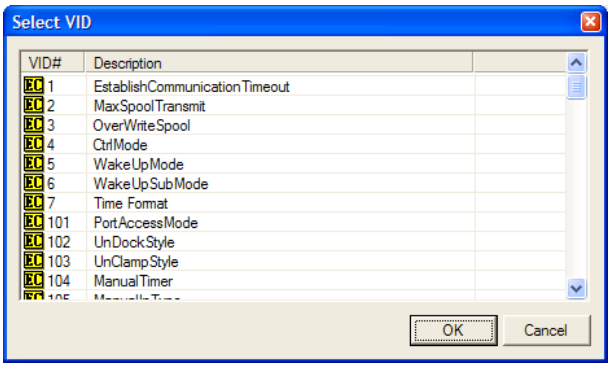


項目	説明
Initial communication state	アプリケーション起動時の通信状態。以下の状態のうちいずれかを選択する。 Disabled　通信無効 Enabled　通信有効
Initial control state (Transition #1)	起動時のコントロール状態。状態遷移#1で移行する状態。以下の状態のうちいずれかを選択する。 Offline　オフライン Online　オンライン
Initial offline state (Transition #2)	起動時のオフライン状態。状態遷移#2で移行する状態。以下の状態のうちいずれかを選択する。 Equipment offline　装置オフライン Attempt online　オンライン移行 Host offline　ホストオフライン
Default offline state (Transition #4)	デフォルトのオフライン状態。状態遷移#4で移行する状態。以下の状態のうちいずれかを選択する。 Equipment offline　装置オフライン Host offline　ホストオフライン
Default online state (Transition #7)	デフォルトのオンライン状態。状態遷移#7で移行する状態。以下の状態のうちいずれかを選択する。 Online local　オンラインローカル Online remote　オンラインリモート

■Predefined VID タブ



項目	説明
Predefined VID Name	定義済みVID名。
VID#	VID番号。
Description	説明。
Edit...	「編集」ボタン。定義済みVIDを1つ選んでこのボタンを押すと、以下のダイアログボックスが表示される。

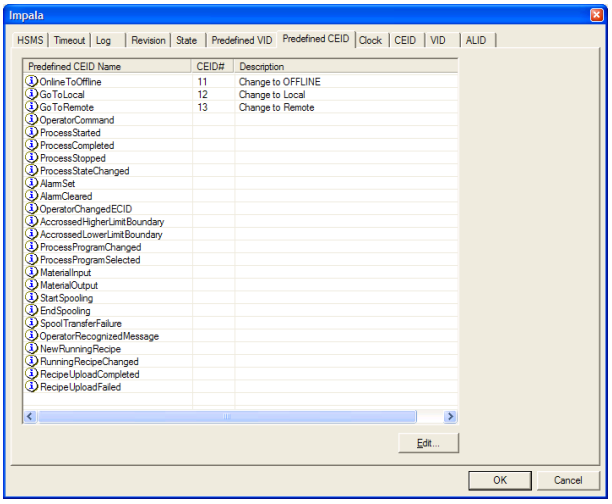


ここにはVIDタブで登録済みのVID一覧が表示される。定義済みVIDと同一の意味のVIDを選択してOKボタンを押す。

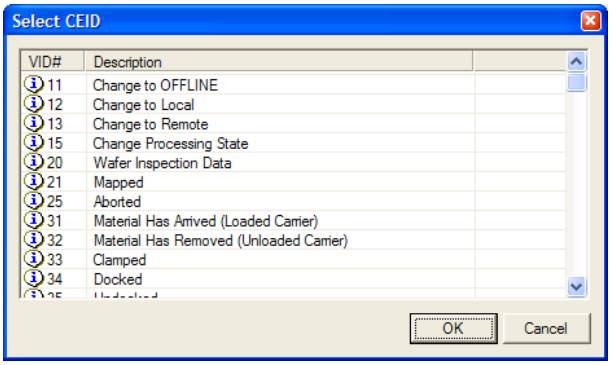
一覧にはたくさんの定義済みVIDが表示されているが、現バージョンで実際に使用しているのは以下のものだけである。

項目	説明
Establish Communications Timeout	ECタイムアウト。通信が確立していないときに、再びS1F13 通信確立要求(CR)を投げる間隔である。
Time Format	時間のフォーマット。12バイトか16バイトかである。
Control State	コントロール状態。
Process State	プロセッシング状態。

■Predefined CEID タブ

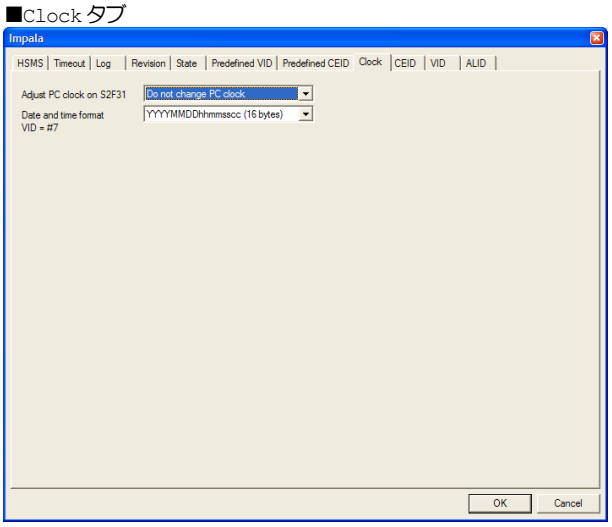


項目	説明
Predefined CEID Name	定義済みCEID名。
CEID#	CEID番号。
Description	説明。
Edit...	「編集」ボタン。定義済みCEIDを1つ選んでこのボタンを押すと、以下のダイアログボックスが表示される。



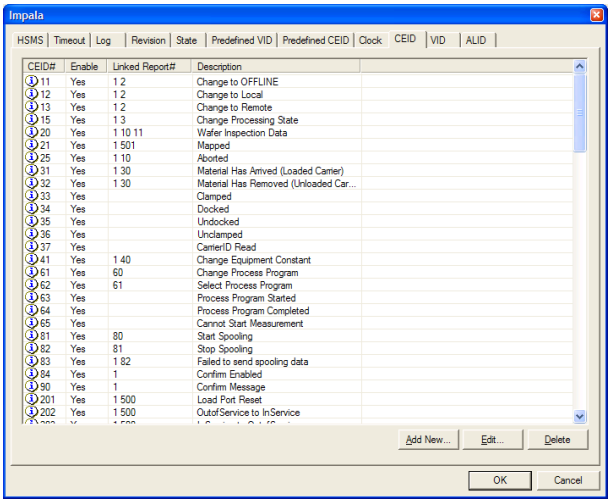
ここには CEID タブで登録済みの CEID 一覧が表示される。定義済み CEID と同一の意味の CEID を選択して OK ボタンを押す。

■Clock タブ



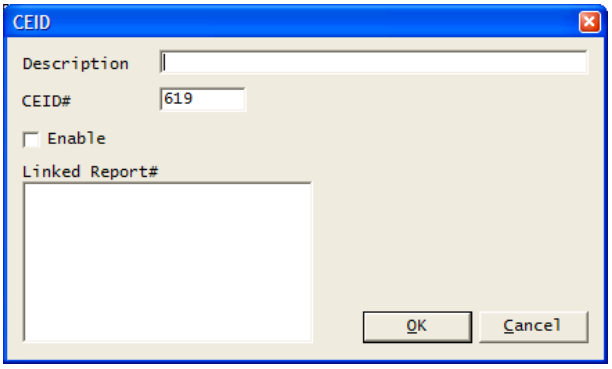
項目	説明
Adjust PC clock on S2F31	S2F31 を受け取ったときにパソコンの時計を変更するかどうかを選択する。
Date and time format	日付・時刻のフォーマット。この項目を設定する前に、Predefined VID で登録しておく必要がある。登録されていれば VID 番号が表示される。

■CEID タブ



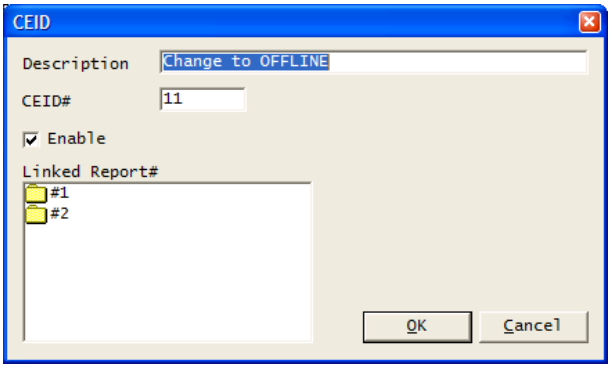
項目	説明
CEID#	CEID 番号。
Enable	Yes なら有効、No なら無効であることを表す。
Linked Report#	リンクされているレポート番号。
Description	説明。
Add New...	新規に追加する。
Edit...	選択された CEID を編集する。
Delete	選択された CEID を削除する。

「Add New...」ボタンを押すと以下のダイアログボックスが表示される。



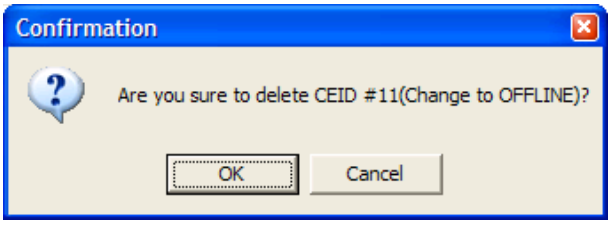
項目	説明
Description	CEID の説明。
CEID#	CEID 番号。登録されている一番大きい数字の CEID より 1 つ大きい数字がデフォルトで入る。
Enable	チェックが付いていると有効、付いていなければ無効。
Lined Report#	リンクされているレポート番号。レポート定義は通信で設定する。

CEID を 1 つ選択し、「Edit...」ボタンを押すと以下のダイアログボックスが表示される。



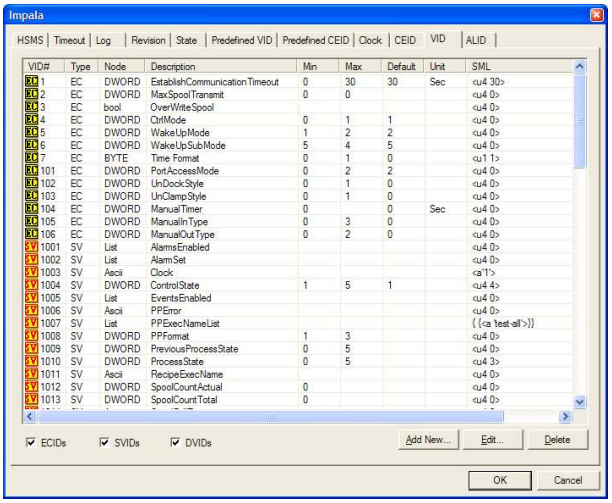
このCEIDはレポート「#1」と「#2」が定義されていることが分かる。

CEID を 1 つ選択し、「Delete」ボタンを押すと以下のメッセージボックスが表示される。



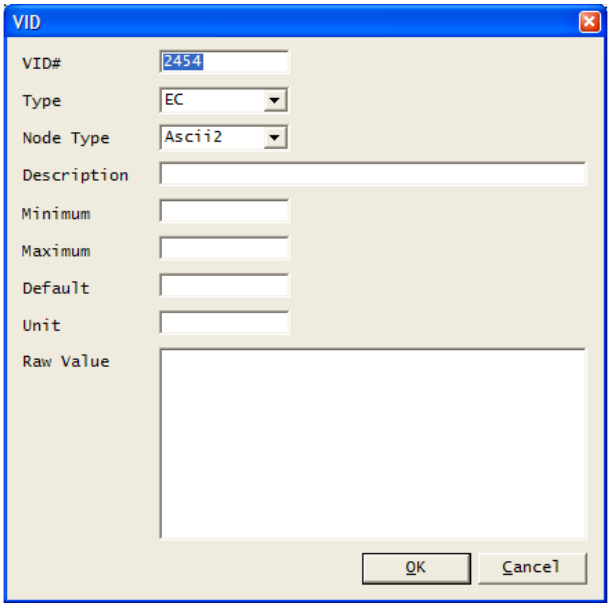
OK ボタンを押すと、選択された CEID が消去される。

■VID タブ



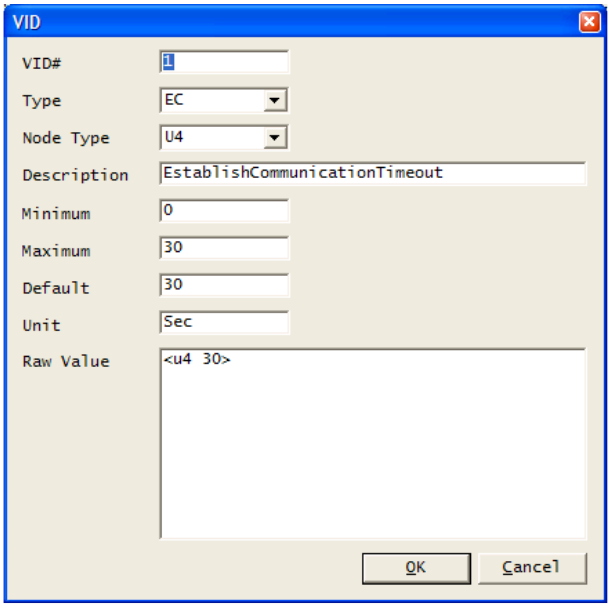
項目	説明
VID#	VID 番号。
Type	変数タイプ。以下の 3 つがある。 EC　装置定数 SV　システム変数 DV　データ変数
Node	ノードの型。C++風に表現される。 List　リスト構造 Binary　バイナリ bool　ブーリアン代数 Ascii　アスキー文字列 JIS 8　JIS8 文字列（半角カタカナ） int64　符号なし 8 バイト整数 char　符号なし 1 バイト整数 short　符号なし 2 バイト整数 long　符号なし 4 バイト整数 double　8 バイト浮動小数点数 float　4 バイト浮動小数点数 uint64　符号つき 8 バイト整数 BYTE　符号つき 1 バイト整数 WORD　符号つき 2 バイト整数 DWORD　符号つき 4 バイト整数 MBCS　2 バイト文字列
Description	説明。
Min	最小値。
Max	最大値。
Default	デフォルト値。
Unit	単位。
SML	実際の変数の値の SML 表現。
ECID	チェックが付いていると ECID が一覧に表示される。付いていないと一覧には表示されない。
SVID	チェックが付いていると SVID が一覧に表示される。付いていないと一覧には表示されない。
DVID	チェックが付いていると DVID が一覧に表示される。付いていないと一覧には表示されない。
Add New...	新規に追加する。
Edit...	選択された VID を編集する。
Delete	選択された VID を削除する。

「Add New...」ボタンを押すと以下のダイアログボックスが表示される。

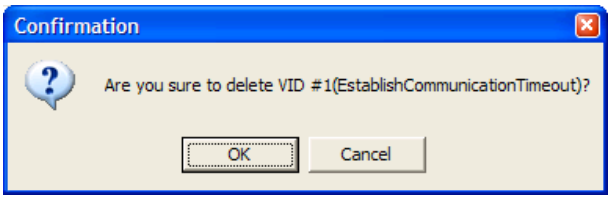


項目	説明
VID#	VID 番号。登録されている一番大きい数字の VIDより 1 つ大きい数字がデフォルトで入る。
Type	変数タイプ。
Node Type	ノードの型。以下のいずれかから選択する。 List　リスト構造 Binary　バイナリ Boolean　ブーリアン代数 Ascii　アスキー文字列 JIS8　JIS8 文字列（半角カタカナ） I8　符号なし 8 バイト整数 I1　符号なし 1 バイト整数 I2　符号なし 2 バイト整数 I4　符号なし 4 バイト整数 F8　8 バイト浮動小数点数 F4　4 バイト浮動小数点数 U8　符号つき 8 バイト整数 U1　符号つき 1 バイト整数 U2　符号つき 2 バイト整数 U4　符号つき 4 バイト整数 Ascii2　2 バイト文字列
Description	説明。
Minimum	最小値。
Maximum	最大値。
Default	デフォルト値。
Unit	単位。
Raw Value	実際の変数の値の SML 表現。

VID を 1 つ選択し、「Edit...」ボタンを押すと以下のダイアログボックスが表示される。



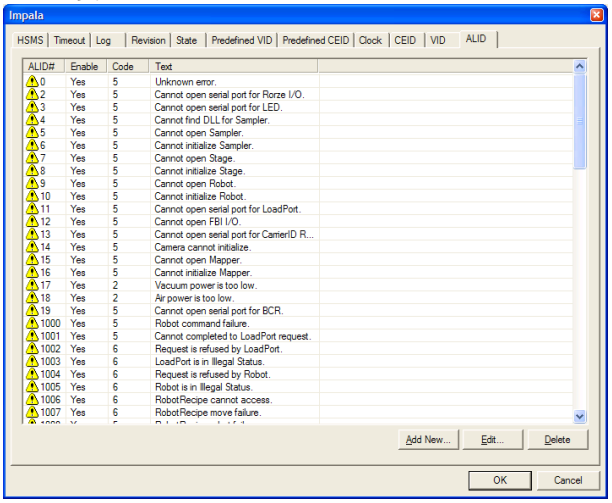
VID を 1 つ選択し、「Delete」ボタンを押すと以下のメッセージボックスが表示される。



OK ボタンを押すと、選択された VID が消去される。

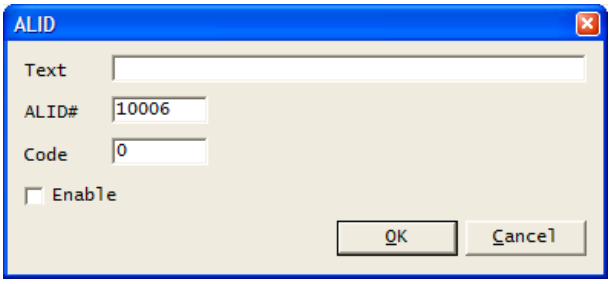
データが実際に登録されるのは設定ダイアログボックスの OK ボタンが押されたときである。このため VID タブで登録した内容を Predefined VID タブで使用する場合は、いったん OK ボタンを押して登録し、その後もう一度 Configure メソッドを呼び出す必要がある。

■ALID タブ

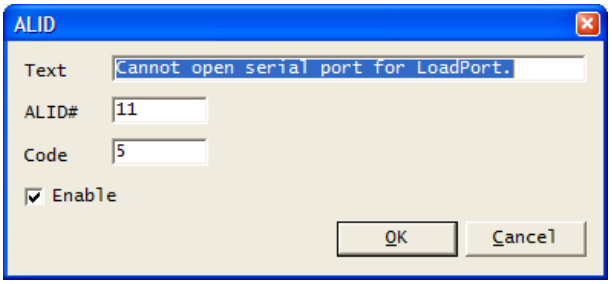


項目	説明
ALID#	ALID 番号。
Enable	Yes なら有効、No なら無効であることを表す。
Code	アラームコード（ALCD）。
Text	アラームテキスト（ALTX）。
Add New...	新規に追加する。
Edit...	選択された ALID を編集する。
Delete	選択された ALID を削除する。

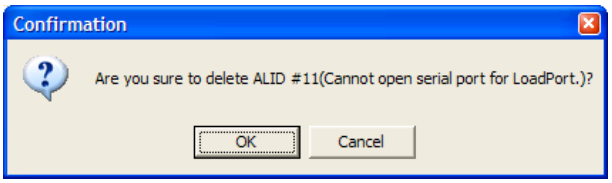
「Add New...」ボタンを押すと以下のダイアログボックスが表示される。



ALID を 1 つ選択し、「Edit...」ボタンを押すと以下のダイアログボックスが表示される。



ALID を 1 つ選択し、「Delete」ボタンを押すと以下のメッセージボックスが表示される。



OK ボタンを押すと、選択された ALID が消去される。

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

引数	説明
lpszCaption	ダイアログボックスのキャプションタイトル。この値が NULL かレングスが 0 の文字列の場合には「Preferences」と表示される。
lOptionFlag	オプションフラグ。

オプションフラグは下記の値から少なくとも 1 つ以上指定する。指定されたタブが表示される。下記の表の値は 16 進数で表記されている。

値	表示されるタブ
0x0001	モデル
0x0002	HSMS
0x0004	タイムアウト
0x0008	リビジョン
0x0010	ステートモデル
0x0020	クロック
0x0040	CEID
0x0080	VID
0x0100	ALID
0x0200	ログファイル
0x0400	定義済み VID
0x0800	定義済み CEID

将来の機能追加によってタブが増えることが予想される。このため全てのタブを表示したい場合は-1 を指定する。

■戻り値

型	説明
BOOL	設定が変更された場合は true が、変更されなかった場合は false が返る。

8.2.2 DefProc

■説明

メッセージを受信したときのデフォルトの処理を行わせる。

■宣言

■Visual C++ 6

```
BOOL DefProc();
```

■Visual Basic 6

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

■戻り値

型	説明
BOOL	処理を行った場合は true が、行わなかった場合は false が返る。

8.2.3 IndexToALID

■説明

インデックスを ALID に変換する。

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

引数	説明
lIndex	0 から始まるインデックス。

■戻り値

型	説明
long	変換された ALID。インデックスが範囲外の場合はマイナス値が返る。

8.2.4 IndexToCEID

■説明

インデックスを CEID に変換する。

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

引数	説明
lIndex	0 から始まるインデックス。

■戻り値

型	説明
long	変換された CEID。インデックスが範囲外の場合はマイナス値が返る。

8.2.5 IndexToVID

■説明

インデックスを VID に変換する。

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

引数	説明
lIndex	0 から始まるインデックス。

■戻り値

型	説明
long	変換された VID。インデックスが範囲外の場合はマイナス値が返る。

8.2.6 InvokeAlarm

■説明

アラームを発生させる。具体的にはホストに S5F1 アラーム報告送信 (ARS) を送信する。ALID が登録されていないか無効になっている場合には送信されない。

sALCD にどのような値を指定しても下位 7 ビットは無視される。ALCD はバイナリ型なので 8 ビットしかない。このため sALCD で実際に使われるのは 8 ビット目だけということになる。このビットが 1 なら「アラームの発生」を、0 なら「アラームの解除」となる。

「アラームの解除」を送信するには、事前に「アラームの発生」が行われている必要がある。「アラームの発生」を送信すると bop 内部でその ALID に対して「未解除フラグ」がセットされる。この「未解除フラグ」がセットされていない場合は、「アラームの解除」を送信することはできない。「アラームの解除」を送信すると「未解除フラグ」はリセットされる。

「未解除フラグ」はオンとオフという情報だけが記録されるので、「アラームの発生」を連続して 2 回送信しても「アラームの解除」を 1 回送信しただけで「未解除フラグ」はリセットされる。

「未解除フラグ」の設定情報はファイルに記録されるので、アプリケーションを終了させても復元させることができる。

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

引数	説明
lALID	ALID。ALID は事前に登録されている必要がある。
sALCD	ALCD。実際に使用されるのは 8 ビット目だけである。

■戻り値

型	説明
BOOL	アラームが送信された場合には true が、送信されなかった場合には false が返る。

8.2.7 InvokeEvent

■説明

イベントを発生させる。具体的にはホストに S6F11 イベントレポート送信 (ERS) を送信する。

イベントにレポートがリンクされている場合は、レポートも自動的に生成される。

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

型	説明
lCEID	CEID。

■戻り値

型	説明
BOOL	イベントが送信された場合は true が、送信されなかった場合は false が返る。

8.2.8 IsValidVID

■説明

VID が正しいか検証する。

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

型	説明
lVID	VID。

■戻り値

型	説明
BOOL	VID が登録されている場合は true が、登録されていない場合は false が返る。

8.2.9 Load

■説明

保存しておいた.bop ファイルをロードする。

■宣言

■Visual C++ 6

```
BOOL Load();
```

■Visual Basic 6

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

型	説明
BOOL	ロードできた場合は true が、できなかった場合は false が返る。

8.2.10 LoadIniFile

■説明

保存しておいた.ini ファイルをロードする。

■宣言

■Visual C++ 6

```
BOOL LoadIniFile();
```

■Visual Basic 6

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

型	説明
BOOL	ロードできた場合は true が、できなかった場合は false が返る。

8.2.11 RegisterALID

■説明

ALID を新規に登録する。S5F3 アラーム報告有効/無効送信 (EAS) での設定に影響するので、RegisterALID の使用は推奨できない場合もある。基本的には事前に Configure で追加しておき、Load で読み出すようにする。

■宣言

■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
```

型	説明
lALID	ALID。
sALCD	ALCD。

lpszALTX	ALTX。

■戻り値

型	説明
BOOL	登録に成功した場合は true が、失敗した場合は false が返る。

8.2.12 RegisterVID

■説明

VID を新規に登録する。

■宣言

■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
```

型	説明
lVID	VID。
sType	タイプ。
sNodeType	SECS-II のノード型。
lpszMin	ECMIN。
lpszMax	ECMAX。
lpszDefault	ECDEF。
lpszUnit	UNITS。
lpszDescription	ECNAME。

sType は以下のうちいずれか１つとなる。

タイプ	説明
1	ECID。
2	SVID。
4	DVID。

sNodeType は以下のいずれかとなる。

ノードの型	値	説明
NodeTypeList	1	リスト
NodeTypeBinary	2	バイナリ
NodeTypeBoolean	3	ブーリアン
NodeTypeAscii	4	アスキー文字列
NodeTypeJis	5	JIS-8 コード
NodeTypeLong8	6	8 バイト符号付き整数
NodeTypeChar	7	1 バイト符号付き整数
NodeTypeShort	8	2 バイト符号付き整数
NodeTypeLong	9	4 バイト符号付き整数
NodeTypeDouble	10	8 バイト浮動小数点数
NodeTypeFloat	11	4 バイト浮動小数点数
NodeTypeDWord8	12	8 バイト符号なし整数
NodeTypeByte	13	1 バイト符号なし整数
NodeTypeWord	14	2 バイト符号なし整数
NodeTypeDWord	15	4 バイト符号なし整数
NodeTypeAscii2	16	2 バイトアスキー文字列

■戻り値

型	説明
BOOL	登録に成功した場合は true が、失敗した場合は false が返る。

■関連事項

VIDType, VIDNodeType, VIDMin, VIDMax, VIDDefault, VIDUnit, VIDDescription

8.2.13 Save

■説明

設定を.bop ファイルに保存する。

■宣言

■Visual C++ 6

```
BOOL Save();
```

■Visual Basic 6

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

型	説明
BOOL	正常に保存された場合は true が、失敗した場合は false が返る。

8.2.14 Send

■説明

WorkSpace と Reply で選択されているメッセージを送信する。

■宣言

■Visual C++ 6

```
BOOL Send();
```

■Visual Basic 6

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

型	説明
BOOL	正常に送信された場合は true が、送信できなかった場合は false が返る。

■関連事項

WorkSpace, Reply

8.2.15 UnregisterALID

■説明

ALID を削除する。ALID が登録されていない場合は削除に失敗する。

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

型	説明
lALID	ALID。

■戻り値

型	説明
BOOL	削除に成功した場合は true が、失敗した場合は false が返る。

8.2.16 UnregisterVID

■説明

VID を削除する。VID が登録されていない場合は削除に失敗する。

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

型	説明
lVID	VID。

■戻り値

型	説明
BOOL	削除に成功した場合は true が、失敗した場合は false が返る。

8.2.17 WriteToLogFile

■説明

ログファイルに書き出す。

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

型	説明
lpszText	書き出す文字列。

8.3 イベント

8.3.1 CommunicationStateChanged

■説明

通信状態が変化した。通信状態は以下のいずれかである。

名称	値	説明
Disabled	0	通信無効。
NotCommunicating	1	通信中断。
Communicating	2	通信実行中。

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

型	説明
sNewState	新しい状態。
sPrevState	以前の状態。

8.3.2 Connected

■説明

通信で相手と接続した。

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

型	説明
lpszIPAddress	通信相手の IP アドレス。
lPortNumber	通信相手の TCP ポート番号。

8.3.3 ConnectionStateChanged

■説明

通信の接続状態が変化した。

名称	値	説明
NotConnected	0	未接続。
NotSelected	1	接続（未選択）。
Selected	2	接続（選択）。

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

型	説明
sNewState	新しい状態。
sPrevState	以前の状態。

8.3.4 ControlStateChanged

■説明

コントロール状態が変化した。コントロール状態は以下のいずれかである。

名称	値	説明
EquipmentOffLine	0	装置オフライン。
AttemptOnLine	1	オンライン確立試行。
HostOffLine	2	ホストオフライン。
OnLineLocal	3	オンラインローカル。
OnLineRemote	4	オンラインリモート。

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

型	説明
sNewState	新しい状態。
sPrevState	以前の状態。

8.3.5 Disconnected

■説明

通信で相手と切断した。

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

型	説明
lpszIPAddress	通信相手の IP アドレス。
lPortNumber	通信相手の TCP ポート番号。

8.3.6 Errors

■説明

エラーが発生した。

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

型	説明
sErrorCode	エラーコード。
lpszErrorText	エラー文字列。

8.3.7 Received

■説明

メッセージを受信した。通常は DefProc を呼び出して bop にメッセージ処理を行わせる。メッセージを bop に処理させたくない場合には、独自の処理を行わせることもできる。

受信したメッセージは WorkSpace0 の Reply=false にセットされる。受信メッセージを解析したい場合は、そのまま処理を行うことができる。受信メッセージに対して返信したい場合には、Reply を true にセットし、返信部分を編集する。この後にイベントハンドラ関数から DefProc を呼び出すと、返信が必要なメッセージの場合は返信部分に書かれた返信メッセージを送信する。

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

型	説明
lpszIPAddress	通信相手の IP アドレス。
lPortNumber	通信相手の TCP ポート番号。

■関連事項

WorkSpace, Reply, Send, DefProc

8.3.8 Sent

■説明

メッセージを送信した。

■宣言

■Visual C++ 6

```
void FireSent();
```

■Visual Basic 6

```
Event Sent()
```

■関連事項

WorkSpace, Reply, Send

8.3.9 VIDChanged

■説明

メッセージを送信した。

■宣言

■Visual C++ 6

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

■Visual Basic 6

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

型	説明
lVID	VID。

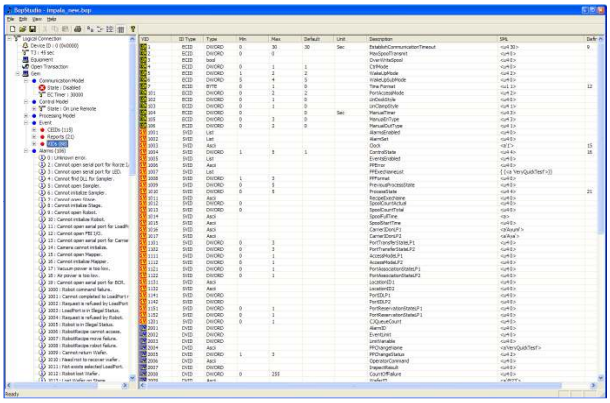
9 BopStudio

Jazz Soft では bop 用開発支援ツールも提供している。これらは無償で提供されており、HASP キーは必要ない。

BopStudio とは.bop の編集を行うソフトである。bop ActiveX コントロールの Configure メソッドを用いれば編集は可能なのだが、使い勝手を向上させている。現バージョンではまだ編集機能は使えないが、アスキー形式の.bopsource ファイルのコンパイル機能、.bop ファイルから.bopsource ファイルへの逆コンパイル機能などがある。これらの機能は Win32 コンソールアプリケーション版でも、BopCompiler、BopRetriever として提供している。

9.1 画面説明

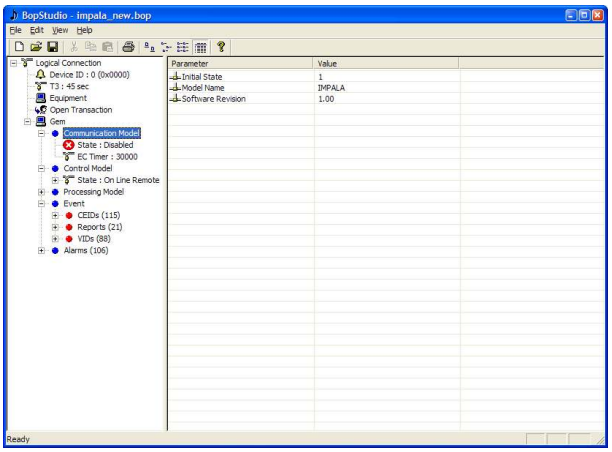
BopStudio は以下のような画面である。



画面はエクスプローラと同じように左右２つのペインに別れており、左ペインの選択を変えることで右ペインの情報も変わる。

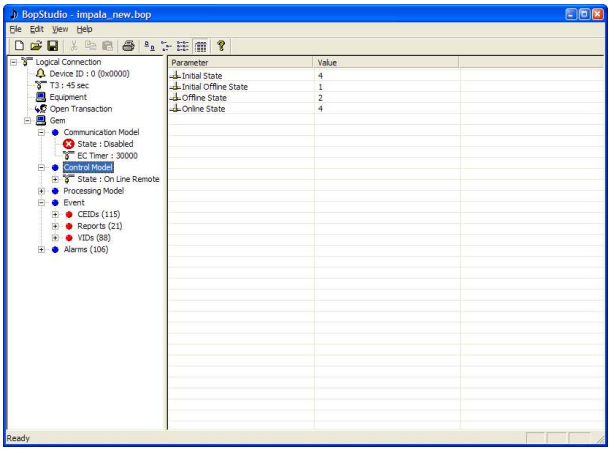
9.1.1 Communication Model

通信の接続モデルである。



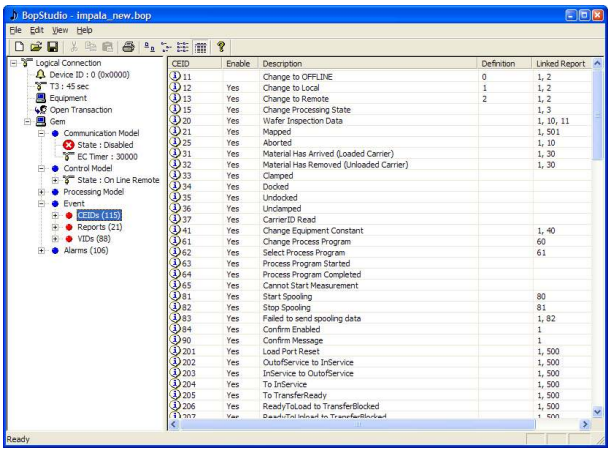
9.1.2 Control Model

コントロール状態モデルである。



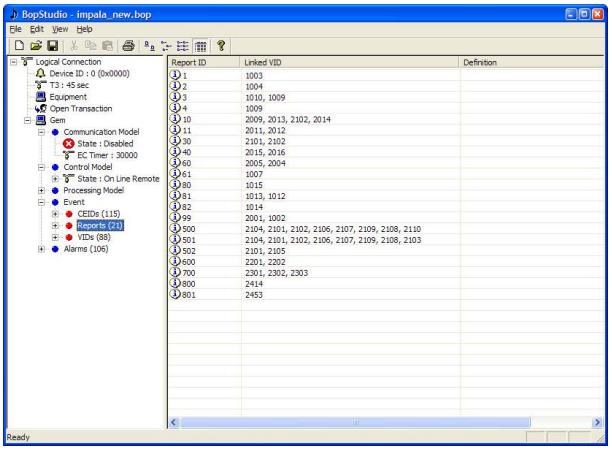
9.1.3 CEIDs

イベント定義である。



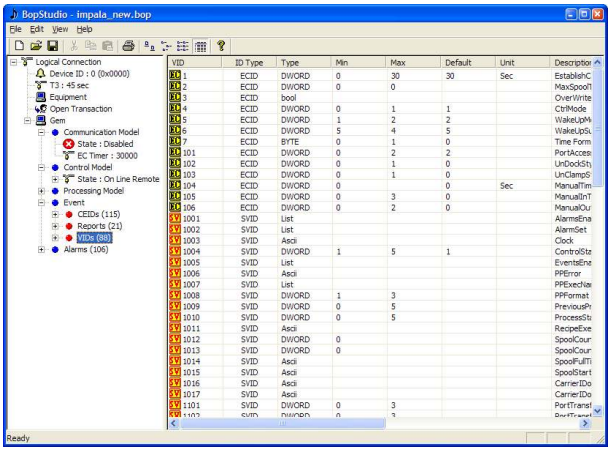
9.1.4 Reports

レポート定義である。



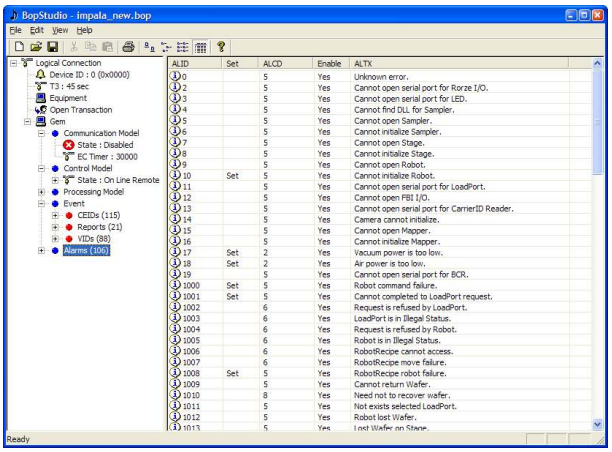
9.1.5 VIDs

変数定義である。



9.1.6 Alarms

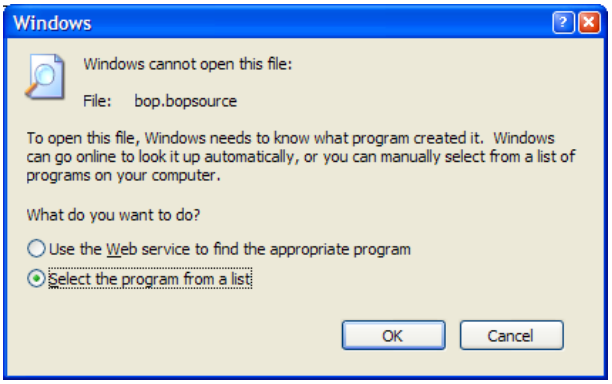
アラーム定義である。



9.2 拡張子の関連付け

一度 BopStudio を起動すると、拡張子.bop は BopStudio に自動的に関連付けられる。これにより以後はエクスプローラなどで.bop ファイルをダブルクリックするだけで BopStudio が起動し、自動的に.bop ファイルの内容が読み込まれる。

拡張子.bopsource は直接的には BopStudio に関連付けはしないが、これも関連付けすることは可能である。エクスプローラで.bopsource ファイルをダブルクリックすると、以下のようなダイアログボックスが表示される。「Select the program from a list」を選択して OK ボタンを押す。



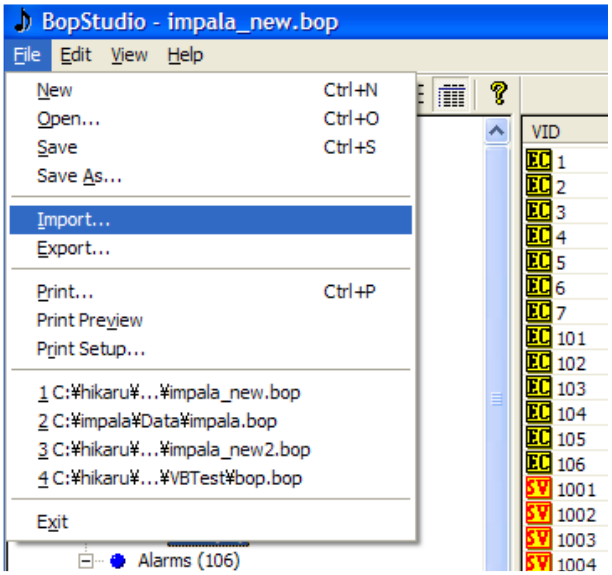
「Open With」ダイアログボックスが表示されたら、Other Programs の中から BopStudio を選択し、「Always use the selected program to open this kind of file」にチェックマークが入っていることを確認した上で OK ボタンを押す。

これで今後はエクスプローラなどで.bopsource ファイルをダブルクリックすると自動的に BopStudio が起動する。.bopsource は ASCII 形式のテキストファイルだが、BopStudio は読み込み時に自動的にコンパイルされる。

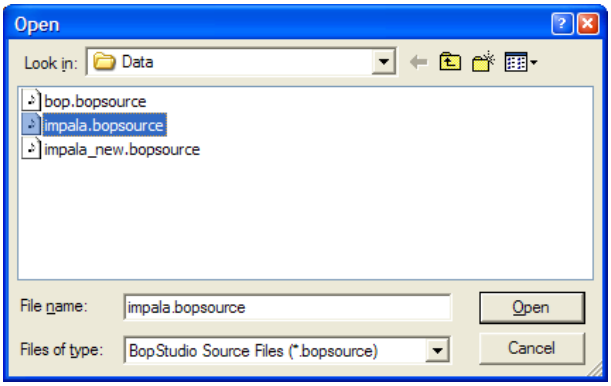
9.3 インポート

インポートとは.bopsource ファイルをコンパイルして読み込むことである。BopCompiler では必要最小限の情報しか表示していないが、インポート画面ではもう少し詳細な情報が表示される。

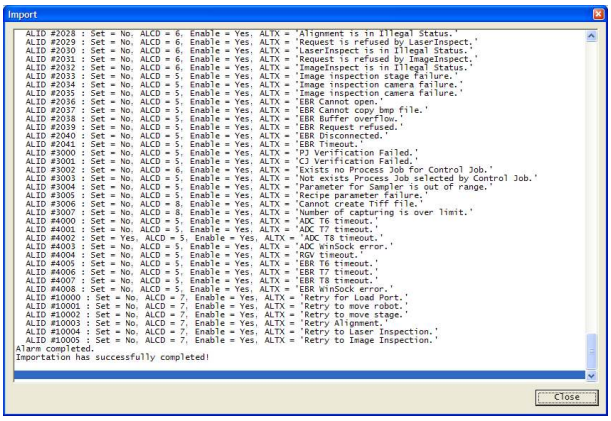
インポートするには File メニューから「Import...」を選択する。



ファイル名を指定するダイアログボックスが表示されるので、.bopsource ファイルを選んで Open ボタンを押す。



コンパイルが実行され、情報がリストボックスに表示される。一番最後に「Importation has successfully completed!」と表示されればコンパイルが正常に終了したことになる。

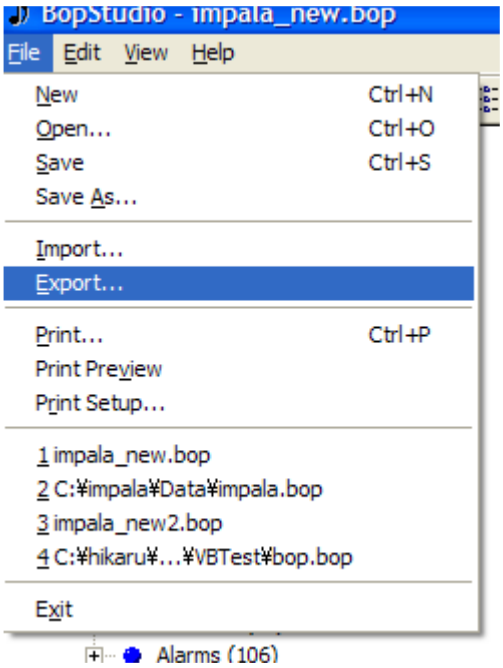


インポートはASCII形式のテキストファイルを読み込みながら、字句解析、構文解析というコンパイル作業を行うため、数秒～数十秒の時間を要する（コンピュータの処理能力による）。

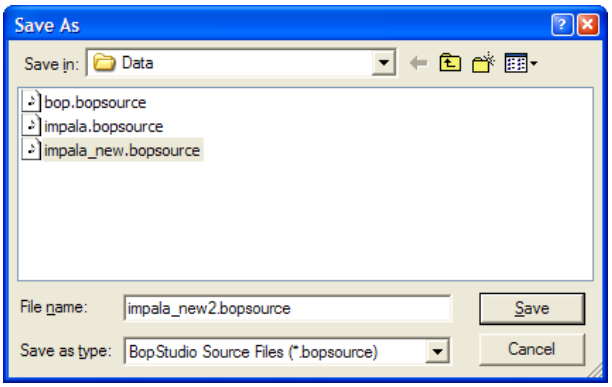
9.4 エクスポート

エクスポートとは.bopsource ファイルに書き出すことである。.bop ファイルを読み込んだ内容をエクスポートすることもできるし、.bopsource ファイルを読み込んでも構わない（あまり意味はないが）。

エクスポートするにはFileメニューから「Export...」を選択する。



ファイル名を指定するダイアログボックスが表示されるので、適当なファイル名を指定してSaveボタンを押す。



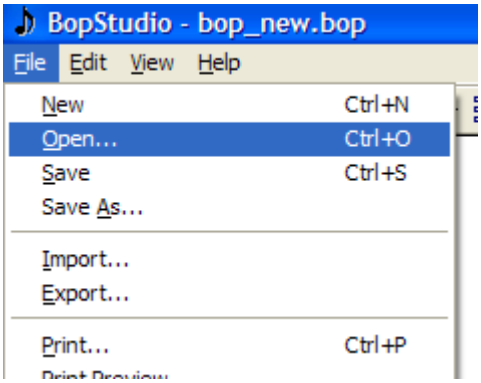
エクスポートが完了すると以下のように「Completed!」というメッセージボックスが表示される。

エクスポートはバイナリ化された内部構造をASCII形式のテキストファイルに書き出すだけなので、一瞬にして完了する。

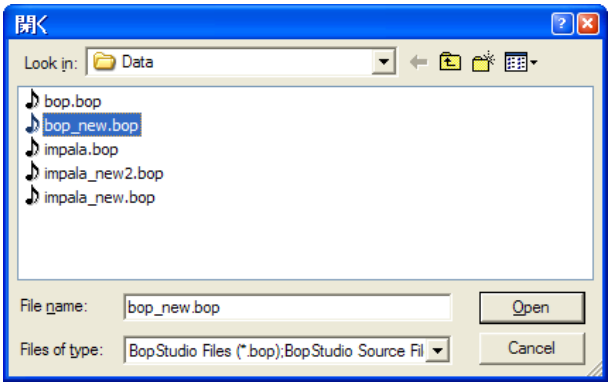
9.5 オープン

オープンとは.bop ファイル、もしくは.bopsource ファイルを読み込むことである。.bop ファイルはバイナリ形式なので「デシリアライズ（Deserialize）」、.bopsource ファイルはASCII形式なので「コンパイル（Compile）」となり、処理は異なる。

オープンするにはFileメニューから「Open...」を選択する。



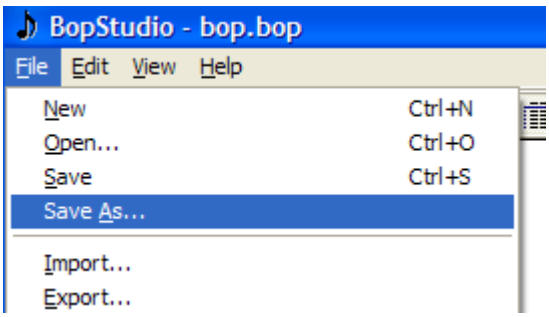
ファイル名を指定するダイアログボックスが表示されるので、選択してOpen ボタンを押す。このとき.bop ファイルのみが表示されるが、「Files of type」で「すべてのファイル(*.*)」を選択すると.bopsource ファイルも表示させることができる。



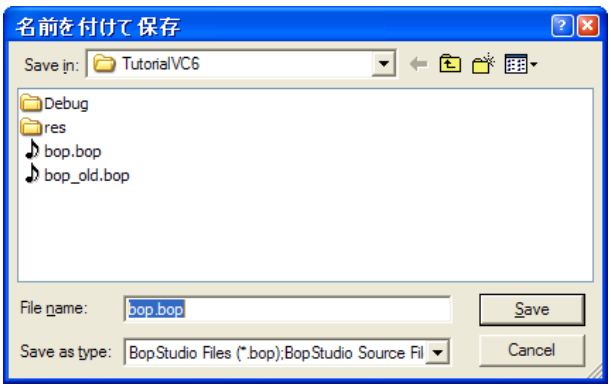
9.6 セーブ

セーブとは.bop ファイル、もしくは.bopsource ファイルを保存することである。.bop ファイルはバイナリ形式なので「シリアライズ（Serialize)」、.bopsource ファイルは ASCII 形式なので「ASCII 出力」となり、処理は異なる。

セーブするには File メニューから「Save As...」を選択する。



ファイル名を指定するダイアログボックスが表示されるので、適当なファイル名を指定して Save ボタンを押す。



10 BopCompiler

Jazz Soft では bop 用開発支援ツールも提供している。これらは無償で提供されており、HASP キーは必要ない。

BopStutdio にも搭載されている.bopsource ファイルのコンパイラである。出力ファイルとして、バイナリ形式の.bop ファイルが得られる。Win32 コンソールアプリケーションとして実装されているので、コマンドプロンプトで使用できる。また Visual Studio を始めとする各種統合開発環境への組み込みも可能である。

10.1 使用方法

コマンドプロンプトで以下のように指定する。

```
BopCompiler  bopsource ファイル名  bop ファイル名
```

引数	説明
bopsource ファイル名	コンパイルする bopsource ファイルを指定する。ASCII 形式のテキストファイルである。
bop ファイル名	コンパイルして生成される bop ファイルを指定する。

10.2 bopsource ファイル文法規則

.bopsource ファイルは ASCII 形式のテキストファイルである。C, C++, Java, C#言語と少し似ているところがある。

10.2.1 2バイト文字

2バイト文字はたとえコメント中であっても使用することができない。また半角カタカナも使用できない。これらの文字が含まれているとコンパイルエラーとなる。

10.2.2 ホワイトスペース

C, C++, Java, C#言語と同様、ホワイトスペース（スペース、タブ、改行、復改コード）は基本的に区切り文字としての意味しかない。このため適度にタブや改行コードを挿入することで見易くすることができる。ただしコメント中および文字列中は文字として扱われる。

10.2.3 コメント

C++, Java, C#言語と同様、「//」から行末まではコメントとなる。また C, C++, Java, C#言語と同様、「/*」から「*/」まではコメントとなる。文字列中にこれらのコードが含まれていても無視される。

「/*」で始まったコメント中で、「//」の後ろに「*/」が書かれている場合、C++言語ではコメントの末尾としているようである。しかし bopsource ファイルではこれをコメントアウトすることができ、引き続きコメントが継続される。

10.2.4 クラス

データは全て何らかのクラスに属している。クラスには以下のものがある。

クラス名	説明
CommunicationModel	通信状態モデル。
ControlModel	コントロール状態モデル。
EventModel	イベントモデル。
AlarmModel	アラームモデル。

イベントモデルは以下のサブクラスを内包している。

クラス名	説明
VariableModel	変数定義。
ReportModel	レポート定義。
CEIDModel	イベント定義。

クラスの構文は以下のようになっている。

```
class  クラス名
{
    文 もしくは クラス
}
```

クラスを分割して記述することもできる。クラスに含まれている文はコンパイル時にまとめられる。

```
class  CommunicationModel
{
    InitialState  =  1;
}

他のクラス定義

class  CommunicationModel
{
    ModelName  =  "IMPALA";
    SoftwareRevision  =  "1.00";
}
```

これは以下のように書いたのと同じ意味である。

```
class  CommunicationModel
{
    InitialState  =  1;
    ModelName  =  "IMPALA";
    SoftwareRevision  =  "1.00";
}
```

10.2.5 文

クラスの中には文が含まれる。含めないこともできるが、それではクラスの意味がない。「;」（セミコロン）までが1つの文となる。文の構文は以下のようになっている。

```
プロパティ  =  値  ;
```

ここで言うプロパティは bop ActiveX コントロールのプロパティとは異なるので注意が必要である。それぞれのクラスごとにプロパティは違うので、詳細についてはクラス説明を読んで欲しい。

同じプロパティに対する文を複数記述すると、後で記述されている内容で上書きされる。

```
    SoftwareRevision  =  "1.00";
    SoftwareRevision  =  "1.10";
```

上記のように記述すると、最初に記述した SoftwareRevision プロパティは上書きされてしまうので、"1.10"がセットされる。

10.2.6 プロパティ

プロパティは C++/Java/C#言語で言うところのメンバ変数のことである。Visual Basic ではプロパティと呼ばれている。ただし bop ActiveX コントロールのプロパティとは異なるので注意が必要である。下記の場合、「ModelName」がプロパティ名である。

```
    ModelName  =  "IMPALA";
```

一部のプロパティは配列になっているものもある。

```
VID[1]   =   (1, 15, "0", "30", "30", "Sec", "ECTimeout", "<u4 30>", 9) ;
```

配列は「[」と「]」で囲まれた範囲で、中に数値が入る。

```
プロパティ[数値]
```

C/C++/Java/C#言語では配列は 0 から始まる連続した数字だが、bopsource ファイルでは連続していないので、注意が必要である。

10.2.7 数値

数値は 0～9 までの文字の連続である。16 進数で記述したい場合は「0x」または「&H」を先頭に付加する。この場合は a～f と A～F までの文字も使用できる。bopsource で使われる数値は全て正の整数なので、マイナス符号や小数点を含むことはできない。

10.2.8 数値リスト

数値を並べたものである。数値と数値の間には「&」を入れる。

例えば、レポート#100 に VID#1 と VID#5 を定義するには、数値リストを使って並べればよい。

```
ReportID[100] = ( "", 1 & 5 ) ;
```

10.2.9 組み込み定数

組み込み定数として以下の表現を使用することもできる。

■ブーリアン系

定数	数値	定数	数値
Yes	1	No	0
Enable	1	Disable	0
Enabled	1	Disabled	0
true	1	false	0
Set	1	Clear	0
		Cleared	0

■SECS-II ノード型

定数	数値	定数	数値
List	1	F8	10
Binary	2	double	10
bool	3	F4	11
Boolean	3	float	11
Ascii	4	U8	12
JIS	5	DWORD8	12
JIS8	5	unsigned __int64	12
I8	6	U1	13
long8	6	BYTE	13
__int64	6	U2	14
I1	7	WORD	14
char	7	U4	15
I2	8	DWORD	15
short	8	MBCS	16
I4	9	DBCS	16
long	9	UNICODE	16

■VID 型

定数	数値
SCID	1
SVID	2
DVID	4

10.2.10 算術演算子

数値は演算子と組み合わせて使用することができる。

■「+」演算子
加算を行う。

例	値の意味
1 + 1	2
3 + 5	8
49 + 1 + 50	100

■「-」演算子
減算を行う。

例	値の意味
2 - 1	1
100 - 98	2
1 - 50 + 49	0

■「*」演算子
乗算を行う。

例	値の意味
1 * 1	1
33333 * 0	0
1 * 2 * 3 * 4	24

■「/」演算子
除算を行い商を求める。0 で割ることはできない。

例	値の意味
1 / 1	1
8 / 2	4
7 / 3	2

■「%」演算子
除算の余りを求める。0 で割ることはできない。

例	値の意味
4 % 2	0
314 % 100	14
999 % 10	9

10.2.11 演算子の優先順位

次のような計算式では2通りの構文木が考えられる。

```
1+2*3
```

■解釈 1

```
(1+2)*3 ……答え :9
```

■解釈 2

```
1+(2*3) ……答え :7
```

数学では乗除演算子「×」「÷」は、加減演算子「+」「-」よりも優先順位が高い。このため上記の解釈 1 は間違いである。bopsource ファイルもこの数学的解釈を踏襲し、乗除演算子は加減演算子より優先順位が高い。

```
1+2*3+4*5 ……答え :27
```

「(」「)」を用いて、より明示的に記述することも可能である。

```
1+(2*3)+(4*5)
```

単項演算子の「-」符号はサポートしていない。bopsource で表現され

る数値は全て正の整数なので、負の数は使用しないため問題はない。しかしどうしても表現したい場合は、上記の「(」「)」を用いればマイナス値を表現することもできる。

```
(0-4) ……マイナス4の意味
```

10.2.12 文字列

文字列はダブルクォーテーション「"」で囲まれた範囲となる。ただし文字列中には改行コードとダブルクォーテーション自身を含めることはできない。

```
ModelName = "IMPALA";
```

10.2.13 連結文字列

SML に関しては改行して可読性を高めたいところである。下記のように記述すると読みにくい。

```
"{<a'ABC'><a'1.00'>}"
```

SML に限ってはC/C++のように文字列を並べて表現できる。並べられた文字列は連結される。文字列と文字列の間には改行コードを入れることもできる。

```
"{"
"       <a'ABC'>"
"       <a'1.00'>"
"}"
```

SML 以外の文字列では連結文字列表現を使うことはできない。

10.2.14 ブロックデータ

数値や文字列がひとかたまりになったデータである。「(」で始まり、「)」で終わる。

```
VID[1] = (1, 15, "0", "30", "30", "Sec", "ECTimeout", "<u4 30>", 9);
```

10.3 CommunicationModel クラス

CommunicationModel クラスは通信状態モデルの設定である。以下のプロパティがある。

プロパティ名	説明
InitialState	アプリケーション起動時の通信状態。
ModelName	S1F13 通信確立要求(CR)などのメッセージに含まれる *MDLN* の文字列。装置のモデル名。最大6バイトまで。
SoftwareRevision	*SOFTREV* の文字列。リビジョン(バージョン)番号。最大6バイトまで。

10.3.1 InitialState

■説明

アプリケーション起動時の通信状態を指定する。

■宣言

```
InitialState = 数値;
```

通信状態は以下のいずれかとなる。

値	説明
0	通信無効。
1	通信有効。

10.3.2 ModelName

■説明

モデル名である。最大6バイト。

■宣言

```
ModelName = 文字列;
```

10.3.3 SoftwareRevision

■説明

ソフトウェアのリビジョン(バージョン)番号。最大6バイト。

■宣言

```
SoftwareRevision = 文字列;
```

10.4 ControlModel クラス

ControlModel クラスはコントロール状態モデルの設定である。以下のプロパティがある。

プロパティ名	説明
InitialState	起動時のコントロール状態。状態遷移#1で移行する状態。
InitialOfflineState	起動時のオフライン状態。状態遷移#2で移行する状態。
OfflineState	デフォルトのオフライン状態。状態遷移#4で移行する状態。
OnlineState	デフォルトのオンライン状態。状態遷移#7で移行する状態。

10.4.1 InitialState

■説明

アプリケーション起動時のコントロール状態を指定する。これは状態遷移#1で移行する状態である。

■宣言

```
InitialState = 数値;
```

コントロール状態は以下のいずれかとなる。

値	説明
0	オフライン。
4	オンライン。

10.4.2 InitialOfflineState

■説明

アプリケーション起動時のオフライン状態を指定する。これは状態遷移#2で移行する状態である。

■宣言

```
InitialOfflineState = 数値;
```

オフライン状態は以下のいずれかとなる。

値	説明
0	装置オフライン。
1	オンライン移行。
2	ホストオフライン。

10.4.3 OfflineState

■説明

デフォルトのオフライン状態を指定する。これは状態遷移#4 で移行する状態である。

■宣言

```
OfflineState = 数値;
```

オフライン状態は以下のいずれかとなる。

値	説明
0	装置オフライン。
2	ホストオフライン。

10.4.4 OnlineState

■説明

デフォルトのオンライン状態を指定する。これは状態遷移#7 で移行する状態である。

■宣言

```
OnlineState = 数値;
```

オンライン状態は以下のいずれかとなる。

値	説明
3	オンラインローカル。
4	オンラインリモート。

10.5 EventModel クラス

EventModel クラスはイベントモデルの設定である。イベントモデルは以下のサブクラスから構成される。

クラス名	説明
VariableModel	変数定義。
ReportModel	レポート定義。
CEIDModel	イベント定義。

10.6 VariableModel クラス

VariableModel クラスは変数定義の設定である。

10.6.1 VID

■説明

配列型プロパティである。配列のインデックスは VID を示す。

```
VID[数値] = ( ブロックデータ );
```

代入されるのはブロックデータである。ブロックデータの構造は以下のようになっている。

#	型	説明
1	数値	変数の種別。下記のうちいずれかである。 1 = ECID 2 = SVID 4 = DVID
2	数値	SECS-II ノード型。
3	文字列	最小値。
4	文字列	最大値。
5	文字列	デフォルト値。
6	文字列	単位。
7	文字列	名称。
8	連結文字列	SML。
9	数値	定義済み VID とのリンク。

10.7 ReportModel クラス

10.7.1 ReportID

配列型プロパティである。配列のインデックスはレポート ID を示す。

```
ReportID[数値] = ( ブロックデータ );
```

代入されるのはブロックデータである。ブロックデータの構造は以下のようになっている。

#	型	説明
1	文字列	レポートの説明。
2	数値リスト	レポートに含まれている変数のリスト。

10.8 CEIDModel クラス

10.8.1 CEID

配列型プロパティである。配列のインデックスは CEID を示す。

```
CEID[数値] = ( ブロックデータ );
```

代入されるのはブロックデータである。ブロックデータの構造は以下のようになっている。

#	型	説明
1	数値	イベントの有効・無効。 1 = 有効 0 = 無効
2	文字列	イベントの説明。
3	数値	定義済み CEID とのリンク。255 を指定すると、リンクはないとみなされる。
4	数値リスト	イベントにリンクされているレポートのリスト。

10.9 AlarmModel クラス

10.9.1 ALID

配列型プロパティである。配列のインデックスはアラーム ID を示す。

```
ALID[数値] = ( ブロックデータ );
```

代入されるのはブロックデータである。ブロックデータの構造は以下の

ようになっている。

#	型	説明
1	数値	アラームの発生・解除。 1 = 発生 0 = 解除
2	数値	アラームのレベル。下表を参照のこと。
3	数値	アラームの有効・無効。 1 = 有効 0 = 無効
4	文字列	アラームの説明。

アラームのレベルか以下のうちいずれかである。

アラームレベル	説明
0	未使用。
1	人間の安全に関わるもの。
2	装置の安全に関わるもの。
3	パラメータコントロールアラーム。
4	パラメータコントロールエラー。
5	回復不能エラー。
6	装置状態の警告。
7	注意フラグ。
8	データの保証不可。
>8	その他のカテゴリ。
9~63	保留。

10.10BNF 表記

参考までに BNF（Backus Naur Form）表記法を用いて bopsource ファイルの文法を定義しておく。

```
Grammar             →   ε | Grammar Communication | Grammar Control | Grammar Processing | Grammar Event | Grammar Alarm
Communication       →   CLASSCOMM '{' CommunicationParam '}'
CommunicationParam  →   ε | CommunicationParam INITIALSTATE '=' Number ';' | CommunicationParam MDLN '=' STR ';'
                        | CommunicationParam SOFTREV '=' STR ';'
Control             →   CLASSCTRL '{' ControlParam '}'
ControlParam        →   ε | ControlParam INITIALSTATE '=' Number ';' | ControlParam INITIALOFFLINESTATE '=' Number ';'
                        | ControlParam OFFLINESTATE '=' Number ';' | ControlParam ONLINESTATE '=' Number ';'
Processing          →   CLASSPROC '{' ProcessingParam '}'
ProcessingParam     →   ε
Event               →   CLASSEVENT '{' EventDefinition '}'
EventDefinition     →   ε | EventDefinition Ceid | EventDefinition Report | EventDefinition Variable
Ceid                →   CLASSCEID '{' CeidList '}'
CeidList            →   ε | CeidList CEID '[' Number ']' '=' '(' Number ',' STR ',' Number ',' LinkedReport ')' ';'
LinkedReport        →   ε | LinkedReport Number | LinkedReport '&' Number CLASSREPORT '{' ReportList '}'
ReportList          →   ε | ReportList RPTID '[' Number ']' '=' '(' STR ',' LinkedVid ')' ';'
LinkedVid           →   ε | LinkedVid Number | LinkedVid '&' Number
Variable            →   CLASSVARIABLE '{' VariableList '}'
VariableList        →   ε | VariableList VID '[' Number ']' '=' '(' Number ',' Number ',' STR ',' STR ',' STR ',' STR ',' STR ',' Strings ',' Number ')' ';'
Alarm               →   CLASSALARM '{' AlarmList '}'
AlarmList           →   ε | AlarmList ALID '[' Number ']' '=' '(' Number ',' Number ',' Number ',' STR ')' ';'
Strings             →   ε | Strings STR
Number              →   Number '+' Number | Number '-' Number | Number '*' Number | Number '/' Number | Number '%' Number | '(' Number ')' | NUM
```

11 BopRetriever

Jazz Soft では bop 用開発支援ツールも提供している。これらは無償で提供されており、HASP キーは必要ない。

BopStudio にも搭載されている.bop ファイルの逆コンパイラである。出力ファイルとして、アスキー形式の.bopsource ファイルが得られる。これも Win32 コンソールアプリケーションとして実装されているので、コマンドプロンプトで使用できる。

11.1 使用方法

コマンドプロンプトで以下のように指定する。

```
BopCompiler  bop ファイル名  [bopsource ファイル名]
```

引数	説明
bop ファイル名	逆コンパイルする bop ファイルを指定する。
bopsource ファイル名	逆コンパイルで生成される bopsource ファイルを指定する。省略すると bop ファイル名に"source"を足したファイル名となる。

【例 1】
Impala.bop を逆コンパイルして Impala_new.bopsource を生成する。

```
BopCompiler  Impala.bop  Impala_new.bopsource
```

【例 2】
Impala.bop を逆コンパイルして Impala.bopsource を生成する。

```
BopCompiler  Impala.bop
```

明示的に指定しても構わない。

```
BopCompiler  Impala.bop  Impala.bopsource
```

12 BopOldRetriever

Jazz Soft では bop 用開発支援ツールも提供している。これらは無償で提供されており、HASP キーは必要ない。

bop ActiveX コントロール 1.00, 1.00a で既に.bop ファイルを作成してしまったユーザ用の.bop ファイルの逆コンパイラである。.bop ファイルの形式は 1.00b より若干異なっており、それ以前の.bop ファイルを直接読み込むことができない。

出力ファイルとして、アスキー形式の.bopsource ファイルが得られる。これも Win32 コンソールアプリケーションとして実装されているので、コマンドプロンプトで使用できる。

BopOldRetriever は BopRetriever と使い方は全く同じである。使用方法については BopRetriever を参照頂きたい。

新しい形式の.bop に変換するには以下の手順で行う。

(1) 旧.bop ファイルの逆コンパイル。

```
BopOldRetriever  旧 bop ファイル名
```

(2) .bopsource ファイルのコンパイル。

```
BopCompiler  bopsource ファイル名  新 bop ファイル名
```

【例】bop.bop を新しい形式に変換する

```
ren  bop.bop  bop_old.bop
BopOldRetriever  bop_old.bop
BopCompiler  bop_old.bopsource  bop.bop
```

13 SML リファレンス

13.1 一般的な注意

13.1.1 ホワイトスペース

ホワイトスペース（スペース、タブ、改行、復改コード）は区切り文字としての意味しかない。このため適度にタブや改行コードを挿入することで見易くすることができる。ただしコメント中および文字列中は文字として扱われる。

13.1.2 コメント

アスタリスク「*」から行末まではコメントとなる。ただし文字列中のアスタリスクは除く。

13.1.3 数値

数値は 0～9 までの文字とマイナス「-」から構成される。16 進数で記述したい場合は’0x’を先頭に付加する。この場合は a～f と A～F までの文字も使用できる。小数は欧米式に「0.9」を「.9」というように、先頭の「0」を省略して記述することもできる。指数表現も可能。また予約語として true（=1）と false（=0）を使うこともできる。

13.1.4 文字列表現

文字列はシングルクォーテーション「’」で囲まれた範囲となる。文字列中には改行コードとシングルクォーテーション自身を含めることはできない。このためどうしてもこれらの文字を入れたい場合は、「0x0a」などのように 16 進数表現を併用する。

13.2 SML 文法

説明文中の太字部分はその文字を記述することを表す。基本的にこれらの文字は大文字でも小文字でも構わない。斜体字はそれぞれの説明を参照すること。また[]で囲まれた部分は省略することができる。

13.2.1 構文

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

要素	説明
xx	ストリーム番号。文字「s」と「f」の間にはスペースを入れないこと。
yy	ファンクション番号。文字「f」と「w」の間にはスペースを入れないこと。
w	ウェイトビット。指定する場合は「w」と記述する。省略可能。
Body	メッセージのボディ。

ストリーム、ファンクション、ウェイトビットはひとかたまりで認識するため、これらの間にスペースや改行コードを入れないようにする。またストリーム、ファンクションを全て省略してメッセージボディのみを記述することもできる。

13.3 メッセージボディ

メッセージのボディは階層構造になっている。

SML 表現	説明
l	リスト
b	バイナリ
bool	ブーリアン
a	アスキー文字列
j	JIS-8
a2	2 バイトアスキー文字列
i8	8 バイト符号付き整数
i1	1 バイト符号付き整数
i2	2 バイト符号付き整数
i4	4 バイト符号付き整数
f8	8 バイト浮動小数点数
f4	4 バイト浮動小数点数
u8	8 バイト符号なし整数
u1	1 バイト符号なし整数
u2	2 バイト符号なし整数
u4	4 バイト符号なし整数

13.3.1 リスト

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

要素	説明
Number	リストの数。SECSIM との互換性のためだけに用意されている。この数字は無視される。
Body	メッセージのボディ。他のアイテムを並べることができる。

13.3.2 バイナリ

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

要素	説明
Numbers	数値。例えば、 `<b 0xff 0x3e 255 0>` のように記述する。

13.3.3 ブーリアン

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

要素	説明
Numbers	数値。例えば、 `<bool true false 1 0>` のように記述する。

13.3.4 アスキー文字列

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

要素	説明
Strings	文字列。長い文字列は分割して記述することもできる。また直接文字コードを記述することもできる。例えば、 `<a ‘ABC’ ‘DEF’ ‘012’ 0x33 ‘4’ 53 54 ‘789’>` のような SML は、

	<a 'ABCDEF0123456789'> と同じである。

13.3.5 2バイト文字列

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

要素	説明
Strings	2バイト文字列。現在のバージョンではMBCSにのみ対応。

13.3.6 JIS-8文字列

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

アスキー型と同じように扱われる。[5]

要素	説明
Strings	文字列。長い文字列は分割して記述することもできる。また直接文字コードを記述することもできる。例えば、 <a 'ABC' 'DEF' '012' 0x33 '4' 53 54 '789'> のようなSMLは、 <a 'ABCDEF0123456789'> と同じである。

13.3.7 整数

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

要素	説明
Numbers	数値。それぞれ以下の意味となる。 i1 符号付き8ビット整数 i2 符号付き16ビット整数 i4 符号付き32ビット整数 i8 符号付き64ビット整数 u1 符号なし8ビット整数 u2 符号なし16ビット整数 u4 符号なし32ビット整数 u8 符号なし64ビット整数 いくつかの数字を並べて記述することもできる。この場合は配列となる。例えば、 <i1 1 0x02 3> のように記述することができる。 現在のバージョンではi8とu8に巨大な値を入れることはできない。

13.3.8 浮動小数点数

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

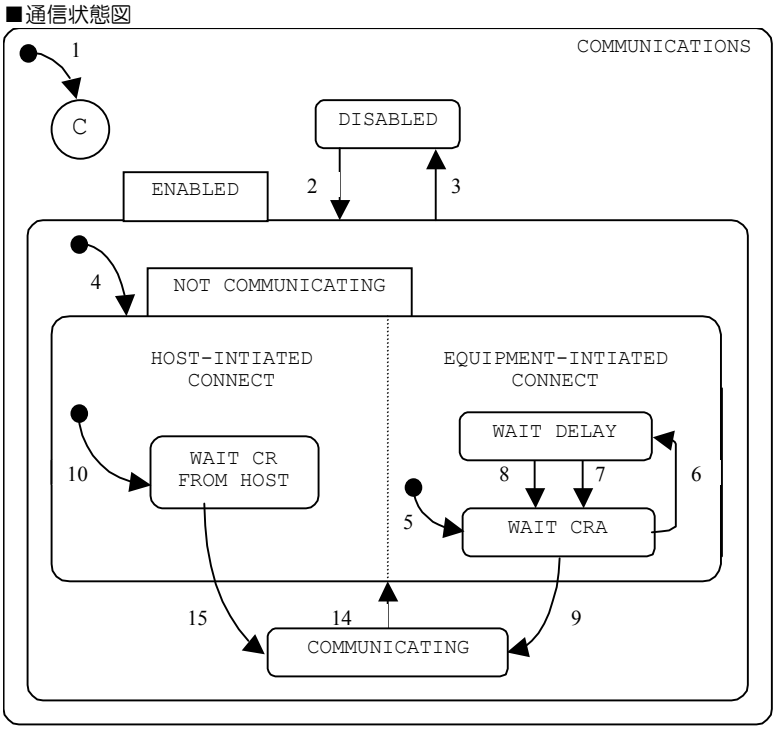
要素	説明
Fnumbers	浮動小数点数。それぞれ以下の意味となる。 f4 32ビット浮動小数点数 f8 64ビット浮動小数点数 例えば、 <f4 0 1.0 3.14> のように記述する。

[5] わたしはJIS-8を使ったメッセージを見たことがない。

14 GEM

14.1 通信状態モデル

■通信状態図



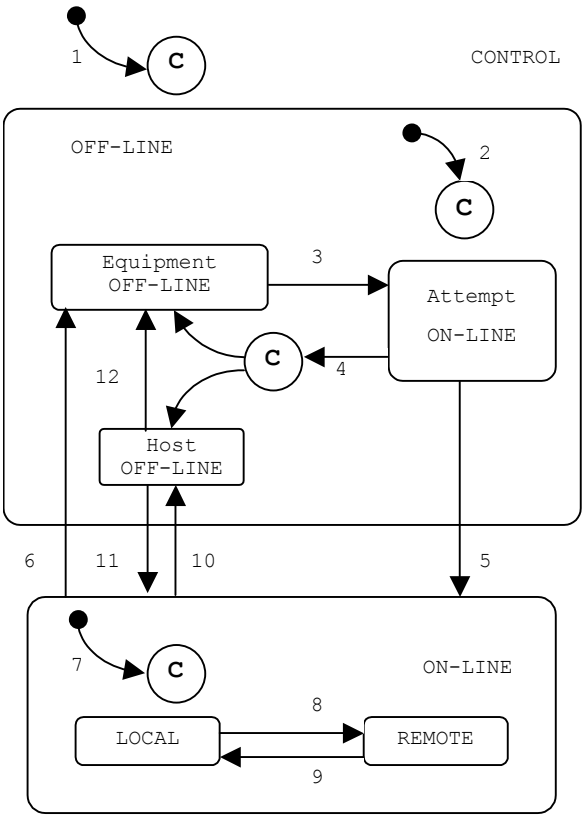
■通信状態遷移

#	現在の状態	トリガー	新しい状態	動作	コメント
1	（通信状態）	システムの初期化。	システムデフォルト	なし。	システムのデフォルトは通信無効または有効に設定される。
2	DISABLED（通信無効）	オペレータが通信無効から通信有効に切り換える。	ENABLED（通信有効）	なし。	SECS-Ⅱ通信が有効になる。
3	ENABLED（通信有効）	オペレータが通信有効から通信無効に切り換える。	DISABLED（通信無効）	なし。	SECS-Ⅱ通信が禁止になる。
4	（通信有効に入る）	通信有効状態に入る。	NOT COMMUNICATING（通信中断）	なし。	システムの初期化から通信有効に入ってもよいし、オペレータが通信有効に切り換えてもよい。
5	（装置開始接続に入る）	（通信中断状態に入る）	WAIT CRA（通信確立要求確認待ち）	通信初期化。CommDelay タイマーを時間切れにセット。S1F13 を送信。	通信確立開始。
6	WAIT CRA（通信確立要求確認待ち）	通信トランザクションの失敗	WAIT DELAY（遅延タイマータイムアウト待ち）	CommDelay タイマーを初期化する。送信するためにキューに入れておいたメッセージを全てキューから出す。	適切な場合には、キューから出されたメッセージは生成順にスプールバッファに入れる。タイマーが時間切れになるのを待つ。
7	WAIT DELAY（遅延タイマータイムアウト待ち）	通信遅延タイマーが時間切れになる。	WAIT CRA（通信確立要求確認待ち）	S1F13 を送信。	S1F14 を待つ。ホストからの S1F13 を受信することもある。
8	WAIT DELAY（遅延タイマータイムアウト待ち）	S1F13 以外のメッセージを受信する。	WAIT CRA（通信確立要求確認待ち）	メッセージを捨てる。応答なし。通信遅延タイマーを“時間切れ”にセットする。S1F13 を送信。	通信を確立するチャンスがあることを意味する。
9	WAIT CRA（通信確立要求確認待ち）	待っていた、COMMACK=0 の S1F14 を受信する。	COMMUNICATING（通信実行）	なし。	通信が確立される。
10	（ホスト開始接続に入る。）	（通信中断状態に入る）	WAIT CR FROM HOST（ホストからの通信確立待ち）	なし。	ホストからの S1F13 を待つ。
11	COMMUNICATING（通信実行）	通信の喪失。（通信の喪失のプロトコル独自の定義については SEMI E4(SECS-Ⅰ) あるいは SEMI E37 (HSMS) を参照すること。）	NOT COMMUNICATING（通信中断）	送信のためにキューに入っていたメッセージを全てキューから取り除く。	キューから取り除かれたメッセージは必要に応じてスプールバッファに入れる。

12	WAIT CR FROM HOST（ホストからの通信確立待ち）	S1F13 を受信する。	COMMUNICATING（通信実行）	COMMACK=0 で S1F14 を受信する。	通信が確立されている。

14.2 コントロール状態モデル

■コントロール状態モデル



■コントロール状態遷移

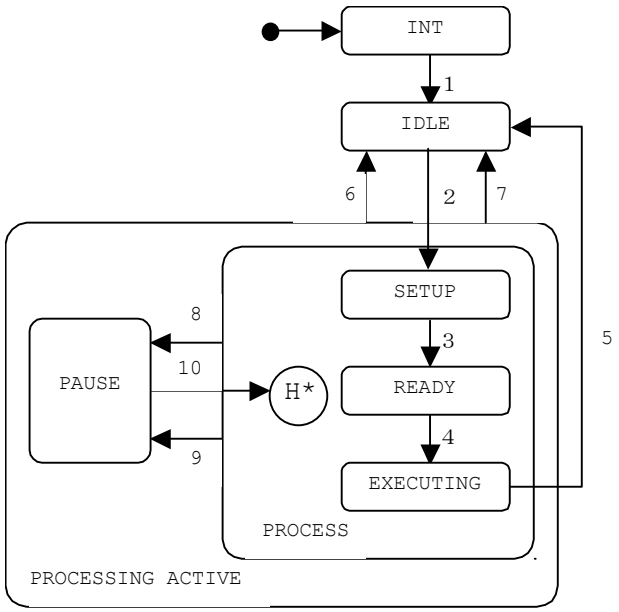
#	現在の状態	トリガー	新しい状態	動作	コメント
1	（未定義）	コントロール状態に入る。（システム立ち上げ）	CONTOROL コントロール状態(下位状態は設定により異なる)	なし。	装置はデフォルト設定でオンラインもしくはオフラインにオフラインになる[6]。
2	（未定義）	オフライン状態に入る。	OFF-LINE オフライン状態(下位状態は設定により異なる)	なし。	装置はデフォルト設定でオフラインのどんな下位状態にでもなる。
3	EQUIPMENT OFF-LINE（装置オフライン）	オペレータがスイッチをオンラインに切り替える。	ATEMP ON-LINE オンライン試行。	なし。	オンライン確立状態にある時はいつでも S1F1 が送信される事に注意。
4	ATEMP ON-LINE（オンライン確立試行）	S1F0	設定条件により異なる新しい状態。	なし。	通信の喪失、返信タイムアウト、もしくは S1F0 の受信による[7]。設定条件により装置オフライン、もしくはホストオフラインに移行する。
5	ATEMP ON-LINE（オンライン確立試行）	装置はホストから期待した S1F2 を受信する。	ON-LINE オンライン。	なし。	ホストは遷移 7 でオンラインに移行する事を通知される。
6	ON-LINE（オンライン）	オペレータがスイッチをオフラインに切り替える。	EQUIPMENT OFF-LINE 装置オフライン。	なし。	“装置オフライン”イベント発生[8]。オフラインの時、イベント返信メッセージは捨てられる。
7	（未定義）	オンライン状態に入る。	ON-LINE オンライン状態(下位状態はリモート/ローカルのスイッチの設定によって決まる)。	なし。	“コントロール状態ローカル”または“コントロール状態リモート”イベント発生。イベントレポートは実際に移行したオンラインの下位状態を示す。
8	LOCAL（ローカル）	オペレータがフロントパネルのスイッチをリモートにセットす	REMOTE リモート。	なし。	“コントロール状態リモート”イベント発生。

[6] 遷移 1 および 2 で述べている条件設定は単一設定でなければならない、これによってユーザーは、装置オフライン、オンライン確立試行、ホストオフライン、オンラインのどれに入るか選択する事ができる。
[7] 通信の喪失は、プロトコール仕様である。通信喪失のプロトコール仕様の定義として、該当するプロトコール規格を参照ください。(例 SEMI E4 または E37)
[8] 装置に対してホストが開始したトランザクションは全て完了しなければならない。そうするには、イベントメッセージの送信に先立ってホストに適切な返信メッセージを送信するか、イベントメッセージの後で（つまりトランザクション終了後）SxF0 を送信する。

		る。			
9	REMOTE （リモート）	オペレータがフロントパネルのスイッチをローカルにセットする。	LOCAL ローカル。	なし。	“コントロール状態ローカル”イベント発生。
10	ON-LINE （オンライン）	装置は“オフライン切り替え”メッセージ（S1F15）をホストから受け取る。	HOST OFF-LINE ホストオフライン。	なし。	“装置オフライン”イベント発生。
11	HOST OFF-LINE （ホストオフライン）	装置は“オンライン移行要求（S1F17）を了解する。	ON-LINE オンライン。	なし。	ホストは遷移 7 でオンラインに移行する事を通知される。
12	HOST OFF-LINE （ホストオフライン）	オペレータがスイッチをオフラインに切り替える。	EQUIPMENT OFF-LINE 装置オフライン。	なし。	“装置オフライン”イベント発生。

14.3 プロセッシング状態モデル

■プロセシング状態図



■プロセシング状態遷移

#	現在の状態	トリガー	新しい状態	動作	コメント
1	INT	装置の初期化完了	IDLE	なし。	なし。
2	IDLE	セットアップ命令がだされた。	SETUP	なし。	なし。
3	SETUP	セットアップ活動は全て完了し、装置は開始支持を受信する用意が整っている。	READY	この活動は装置によって異なる。	なし。
4	READY	装置はホストもしくはオペレータコンソールから開始指示（START）を受信した。	EXECUTING	この活動は装置によって異なる。	なし。
5	EXECUTING	プロセシング作業が完了した。	IDLE	なし。	なし。
6	PROCESSING ACTIVE	装置はホストもしくはオペレータコンソールから停止指示（STOP）を受信した。	IDLE	なし。	なし。
7	PROCESSING ACTIVE	装置はホストもしくはオペレータコンソールから中断指示（ABORT）を受信した。	IDLE	この活動は装置によって異なる。	なし。
8	PROCESS	装置はたとえばアラームのような条件により、一時停止（PAUSE）する事に決めた。	PAUSE	この活動は装置によって異なる。	このタイプの異常については普通オペレータの補助が必要である。
9	PROCESS	装置はホストもしくはオペレータコンソールから一時停止指示（PAUSE）を受信した。	PAUSE	この活動は装置によって異なる。	なし。
10	PAUSE	装置はホストもしくはオペレータコンソールから一時停止解除指示（RESUME）を受信した。	前 PROCESS サブステート	この活動は装置によって異なる。	なし。

14.4 通信確立

14.4.1 ホストからの通信確立

コメント	ホスト	装置	コメント
	通信状態は通信有効（ENABLED）である。（下位状態は何でもよい）		
通信確立要求	S1F13→		
		←S1F14	COMMACK=受諾で応答 通信状態=通信実行（COMMUNICATING）

14.4.2 装置からの通信確立、ホストの確認応答

コメント	ホスト	装置	コメント
	通信状態=通信中断（NOT COMMUNICATING）		
			[LOOP]
			[LOOP]……SEND
		←S1F13	通信確立要求
通信確立要求確認	S1F14→		
			[IF]S1F14 がタイムアウトなしに受信されたら [THEN]ループを出る……SEND [ELSE]Establish Communications-Timeout の間隔分遅延する。 [ENDIF] [END_LOOP]……SEND [IF]COMMACK=受諾 [THEN]通信状態=通信実行（COMMUNICATING）ループを出る…… [ELSE]遅延するためにタイマーをリセットし、Establish Communications-Timeout で指定された間隔分遅延する。 [ENDIF] [END-LOOP]

14.5 GEM 準拠

bop は GEM の全ての機能を実装している訳ではない。例えば「材料移送」などは、アプリケーション側から S6F11 を発行する必要があり、bop 内部で対応できるというものではない。これについては他社の GEM 開発支援環境では堂々と「準拠している」と記述をしているところもあるが、事実を歪曲した誇大広告だと言わざるを得ないだろう。

「性能の有無」で「無し」となっている項目も、必要であればユーザ側で実装できるように配慮してある。このため全ての項目で「GEM 準拠」が可能ではある。ただし「変数データ収集」「トレースデータ収集」「リミット監視」「スプーリング」については、ほとんどの場合で必要とはされないであろう。

GEM 準拠		
GEM の基本条件	性能の有無	GEM への準拠
状態モデル	☑有り ☐無し	☑有り[9] ☐無し
装置プロセス状態	☑有り ☐無し	
ホストが開始する S1F13/F14 シナリオ	☑有り ☐無し	
イベント通知	☑有り ☐無し	
オンライン確認	☑有り ☐無し	
エラーメッセージ	☑有り ☐無し	
コントロール（オペレータ起動）	☑有り ☐無し	
文書化	☑有り ☐無し	
追加性能	性能の有無	GEM への準拠[10]
通信確立	☑有り ☐無し	☑有り ☐無し
動的イベントレポート設定変更	☑有り ☐無し	☑有り ☐無し
変数データ収集	☐有り ☑無し	☐有り ☐無し
トレースデータ収集	☐有り ☑無し	☐有り ☐無し
状態データ収集	☑有り ☐無し	☑有り ☐無し
アラーム管理	☑有り ☐無し	☑有り ☐無し
リモートコントロール	☑有り ☐無し	☑有り ☐無し
装置定数	☑有り ☐無し	☑有り ☐無し
プロセスプログラム管理	☐有り ☑無し	☐有り ☐無し
材料移送	☐有り ☑無し	☐有り ☐無し
装置端末サービス	☐有り ☑無し	☐有り ☐無し
クロック	☑有り ☐無し	☑有り ☐無し
リミット監視	☐有り ☑無し	☐有り ☐無し
スプーリング	☐有り ☑無し	☐有り ☐無し
コントロール（ホスト起動）	☑有り ☐無し	☑有り ☐無し

「プロセスプログラム管理」はプロセスプログラム（レシピ）の構造が、装置によって千差万別なので実装していないが、実装するのもそれほど困難ではない。「材料移送」はイベントを追加するだけである。「装置端末サービス」はアプリケーションのどこかに表示するだけなので、非常に簡単に実装可能である。これらの性能を追加した場合は、「性能の有無」を「有り」、「GEM への準拠」を「有り」にすることができる。

「文書化」に必要な資料は、すでに本ユーザーズマニュアルに記述してあるので、それを適宜加筆修正して使用して構わない。また SEMI 発行の E.5(SECS-II)のマニュアルは非常に読みにくいので、本ユーザーズマニュアルではアイテム辞書を各メッセージの説明に加えてある。

[9] GEM の基本条件全てが GEM に従って実現されている場合に限り、YES に印を付ける事ができる。
[10] GEM の基本条件が GEM に準拠していないと、追加性能の中には GEM に準拠しているとはいえない。つまり、YES に印を付けられないものもある。

15 SECS-II メッセージ

SEMI の E.5（SECS-II）のマニュアルから、bop で使用しているメッセージを抜粋し、読みやすいように書き直した。

bop ではアイテムの型に制限のあるものもある。例えば CEID は SEMI の E.5 では「バイナリ」「符号つき整数」「符号なし整数」のいずれでも構わないと書いてあるが、実際には u4 型を用いることが多い。バイナリでは 0～255 までしか扱えないし、符号つき整数だとマイナス値もありうるということになってしまうので、SEMI の規格が少々おかしいと言えよう。このため bop では u4 固定になっている。

15.1 アイテム辞書

アイテムの説明は各メッセージのところにも書いてあるので、ここを見なければならないような局面は少ないだろう。

15.1.1 ACKC5

■説明

確認コード。1 バイト。

値	説明
0	了解。
>0	エラー。了解できず。
1~63	保留。

■型

```
b[1]
```

■関連メッセージ

S5F2 アラーム報告確認(ARA)
S5F4 アラーム報告有効/無効確認(EAA)

15.1.2 ACKC6

■説明

確認コード。1 バイト。

値	説明
0	了解。
>0	エラー。了解できず。
1~63	保留。

■型

```
b[1]
```

■関連メッセージ

S6F12 イベントレポート確認(ERA)

15.1.3 ACKC7

■説明

確認コード。1 バイト。

値	説明
0	許可された。
1	不許可。
2	レングスエラー。
3	配列オーバーフロー。
4	*PPID* 未定義。
5	モードエラー。
>5	他のエラー。
6~63	保留。

■型

```
b[1]
```

■関連メッセージ

S7F4 プロセスプログラム確認(PPA)
S7F18 プロセスプログラム削除確認(DPA)
S7F24 フォーマット付きプロセスプログラム確認(FPA)

15.1.4 ACKC7A

■説明

確認コード。1 バイト。

値	説明
0	了解された。
1	MDLN が一致しない。
2	SOFTREV が一致しない。
3	無効な CCODE。
4	無効な *PPARM* の値。
5	他のエラー。（*ERRW7* によって示される）。
6~63	保留。

■型

```
i1[1], u1[1]
```

■関連メッセージ

S7F27 プロセスプログラム妥当性送信(PVS)

15.1.5 ACKC10

■説明

確認コード。1 バイト。

値	説明
0	表示了解。
1	メッセージは表示されない。
2	端末使用できない。
3~63	保留。

■型

```
b[1]
```

■関連メッセージ

S10F2 端末要求確認(TRA)
S10F4 端末表示、シングルブロック確認(VTA)
S10F6 端末要求、マルチブロック確認(VMA)

15.1.6 AGENT

■説明

■型

```
a
```

■関連メッセージ

S15F21 レシピアクション要求
S15F22 レシピアクション確認

15.1.7 ALCD

■説明

アラームコード。

ビット	値	説明
bit8	1	アラーム状態発生。
	0	アラーム状態解除。
bit7~1	アラーム区分コード	
	0	未使用。
	1	人間の安全に関わるもの。
	2	装置の安全に関わるもの。
	3	パラメータコントロールアラーム。
	4	パラメータコントロールエラー。
	5	回復不能エラー。
	6	装置状態の警告。
	7	注意フラグ。
	8	データの保証不可。
	>8	その他のカテゴリ。
	9~63	保留。

■型

```
b
```

■関連メッセージ

S5F1 アラーム報告送信(ARS)
S5F6 アラームリストデータ(LAD)

15.1.8 ALED

■説明

アラーム有効/無効コード、1 バイト。

ビット	値	説明
bit8	1	アラーム有効。
	0	アラーム無効。

■型

```
b[1]
```

■関連メッセージ

S5F3 アラーム報告有効/無効送信(EAS)

15.1.9 ALID

■説明

アラーム ID。

■型

```
u4
```

■関連メッセージ

S5F1 アラーム報告送信(ARS)
S5F3 アラーム報告有効/無効送信(EAS)
S5F5 アラームリスト要求(LAR)
S5F6 アラームリストデータ(LAD)

15.1.10 ALTX

■説明

アラームテキスト。最大 40 文字。

■型

```
a
```

■関連メッセージ

S5F1 アラーム報告送信(ARS)
S5F6 アラームリストデータ(LAD)

15.1.11 ATTRDATA

■説明

特定のオブジェクトの特定属性値を持つ。

■型

```
l, b, bool, a, i*, f*, u*
```

■関連メッセージ

S14F1 属性要求(GAR)
S14F2 属性データ要求(GAD)

15.1.12 ATTRID

■説明

特定タイプのオブジェクトのため属性識別子。

■型

```
a, u*
```

■関連メッセージ

S14F1 属性要求(GAR)
S14F2 属性データ要求(GAD)

15.1.13 ATTRRELN

■説明

特定の限定値とオブジェクトインスタンスの属性数値（インタレスト数値）との関係を規定する。

値	説明
0	限定値がインタレストの値と等しい。
1	限定値がインタレストの値と等しくない。
2	限定値がインタレストの値より少ない。
3	限定値がインタレストの値より少ないか、等しい。
4	限定値がインタレストの値より大きい。
5	限定値がインタレストの値より大きいか、等しい。
6	限定値にはインタレストの値（セットに含まれる）がある。

7	限定値にはインタレストの値（セットに含まれない）がない。
>7	保留。

■型

```
u1
```

■関連メッセージ

S14F1 属性要求(GAR)

15.1.14 CCODE

■説明

コマンドコード。各コマンドコードは、機械が行なうことができる個別のプロセス操作に対応する。

■型

```
i2, u2
```

■関連メッセージ

S7F23 フォーマット付きプロセスプログラム送信(EPS)
S7F26 フォーマット付きプログラムデータ(FPD)
S7F27 プロセスプログラム妥当性送信(PVS)

15.1.15 CEED

■説明

収集イベントあるいはトレース有効/無効コード。1バイト。

値	説明
偽	無効。
真	有効。

■型

```
bool[1]
```

■関連メッセージ

S2F37 有効、無効イベントレポート(EDER)

15.1.16 CEID

■説明

収集イベントID。

■型

```
u4
```

■関連メッセージ

S2F33 規定レポート(DR)
S2F35 リンクイベントレポート(LER)
S2F36 リンクイベントレポート確認(LERA)
S2F37 有効、無効イベントレポート(EDER)
S2F38 有効/無効イベントレポート確認(EERA)
S6F11 イベントレポート送信(ERS)
S6F15 イベントレポート要求(ERR)
S6F16 イベントレポートデータ(ERD)

15.1.17 CEPACK

■説明

コマンド拡張パラメータ承認。*CPNAME* の特定の値がリストの *CEPVAL* を持つと定義されている場合には、*CEPACK* は、S2F49[11]（拡張リモートコマンド）で使用されているように、対応する *CEPVAL* のリストフォーマットと同じ構造になる。それ以外、*CEPACK* は1バイトの整数になる。

値	説明
0	エラーなし。
1	パラメータ名（*CPNAME*）が存在しない。
2	不正な値が *CEPVAL* に指定されている。
3	不正なフォーマットが CEPVAL に指定されている。
4	パラメータ名（*CPNAME*）の使用法が有効でない。
5~63	保留。

■型

```
l, u1[1]
```

■関連メッセージ

S2F50 拡張リモートコマンド確認

15.1.18 CEPVAL

■説明

コマンド拡張パラメータ値。*CPNAME* 値が使われているので、*CEPVAL* が特定の使われ方をしている場合は常に分かる。*CEPVAL* には下記の形式がある：単一の（リストでない）値（例：*CPVAL*）、同一フォーマット及びタイプの単一のアイテムのリスト、あるいは形式のアイテムのリスト。

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

■型

```
l, b, bool, a, j, i*, f*, u*
```

■関連メッセージ

S2F49 拡張リモートコマンド
S2F50 拡張リモートコマンド確認

15.1.19 COMMACK

■説明

通信確立確認コード。1バイト。

値	説明
0	了解。
1	否定。再試行。
2~63	保留。

■型

```
b[1]
```

■関連メッセージ

S1F14 通信確立要求確認(CRA)

[11] ?

15.1.20 CPACK

■説明

コマンドパラメータ確認コード。1 バイト。

値	説明
1	パラメータ名（*CPNAME*）は存在しない。
2	*CPVAL* 用として指定された違法な値。
3	*CPVAL* 用として指定された違法なフォーマット。
>3	他の装置固有のエラー。
4~63	保留。

■型

```
b[1]
```

■関連メッセージ

S2F42 ホストコマンド確認(HCA)

15.1.21 CPNAME

■説明

コマンドパラメータ名。

■型

```
a
```

■関連メッセージ

S2F41 ホストコマンド送信(HCS)
S2F42 ホストコマンド確認(HCA)
S2F49 拡張リモートコマンド
S2F50 拡張リモートコマンド確認

15.1.22 CPVAL

■説明

コマンドパラメータ値。

■型

```
b, bool, a, j, i*, u*
```

■関連メッセージ

S2F41 ホストコマンド送信(HCS)
S2F49 拡張リモートコマンド
S2F50 拡張リモートコマンド確認

15.1.23 DATAID

■説明

データ ID。

■型

```
u4
```

■関連メッセージ

S2F33 規定レポート(DR)
S2F35 リンクイベントレポート(LER)
S2F39 マルチブロック問い合わせ(DMBI)
S2F40 マルチブロック許可(MBG)
S2F49 拡張リモートコマンド
S6F5 マルチブロックデータ送信問い合わせ(MBI)
S6F11 イベントレポート送信(ERS)
S6F16 イベントレポートデータ(ERD)
S15F1 レシピ管理マルチブロック問い合わせ
S15F21 レシピアクション要求
S15F27 レシピダウンロード要求
S15F29 レシピ検証要求
S15F35 レシピ削除要求

15.1.24 DATALENGTH

■説明

送信データの総バイト数。

■型

```
u4
```

■関連メッセージ

S2F39 マルチブロック問い合わせ(DMBI)
S6F5 マルチブロックデータ送信問い合わせ(MBI)

15.1.25 DRACK

■説明

定義報告了解コード。1 バイト。

値	説明
0	了解。
1	否定。スペース不十分。
2	否定。無効フォーマット。
3	否定。少なくとも 1 つの *RPTID* は既に定義されている。
4	否定。少なくとも 1 つの *VID* は存在しない。
>4	その他のエラー。
5~63	保留。

■型

```
b[1]
```

■関連メッセージ

S2F34 規定レポート確認(DRA)

15.1.26 EAC

■説明

装置確認コード。1 バイト。

値	説明
0	了解。
1	否定。少なくとも 1 つの定数が存在しない。
2	否定。ビジー。
3	否定。少なくとも 1 つの定数が範囲外にある。
>3	他の装置固有のエラー。
4~63	保留。

■型

```
b[1]
```

■関連メッセージ

S2F16 新装置定数変更確認(ECA)

15.1.27 ECDEF

■説明

装置定数デフォルト値。

■型

```
b, bool, a, j, i*, f*, u*
```

■関連メッセージ

S2F30 装置定数名リスト(ECN)

15.1.28 ECID

■説明

装置定数 ID。

■型

```
u4
```

■関連メッセージ

S2F13 装置定数要求(ECR)
S2F15 新装置定数変更(ECS)
S2F29 装置定数名リスト要求(ECNR)
S2F30 装置定数名リスト(ECN)

15.1.29 ECMAX

■説明

装置定数最大値。

■型

```
b, bool, a, j, i*, f*, u*
```

■関連メッセージ

S2F30 装置定数名リスト(ECN)

15.1.30 ECMIN

■説明

装置定数最小値。

■型

```
b, bool, a, j, i*, f*, u*
```

■関連メッセージ

S2F30 装置定数名リスト(ECN)

15.1.31 ECNAME

■説明

装置定数名。

■型

```
a
```

■関連メッセージ

S2F30 装置定数名リスト(ECN)

15.1.32 ECV

■説明

装置定数。

■型

```
b, bool, a, j, i*, f*, u*
```

■関連メッセージ

S2F14 装置定数データ(ECD)
S2F15 新装置定数変更(ECS)

15.1.33 EDID

■説明

受信すべきデータ ID。次の 3 つが考えられる。

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

■型

```
b, a, i*, u*
```

■関連メッセージ

S9F13 会話タイムアウト(CTN)

15.1.34 ERACK

■説明

有効/無効イベント報告。確認コード。1 バイト。

値	説明
0	了解。
1	否定。少なくとも 1 つの *CEID* が存在しない。
>1	その他のエラー。
2~63	保留。

■型

```
b[1]
```

■関連メッセージ

S2F38 有効/無効イベントレポート確認(EERA)

15.1.35 ERRCODE

■説明

エラー識別コード。

値	説明
0	エラーなし。
1	オブジェクト指定子中のオブジェクト不明。
2	ターゲットオブジェクトタイプ不明。
3	オブジェクトインスタンス不明。
4	属性名称不明。
5	リードオンリー属性。アクセス拒否。
6	オブジェクトタイプ不明。
7	無効属性地。
8	シンタックスエラー。
9	検証エラー。
10	妥当性エラー。
11	オブジェクト指定子が使用中。
12	パラメータが正しく指定されていない。
13	指定すべきパラメータがすべて指定されていない。
14	要求されたオプションはサポートされていない。
15	使用中。
16	処理の準備ができていない。
17	現行の状態に無効なコマンド。
18	変更された材料なし。
19	材料は部分的に処理された。
20	材料は全て処理された。
21	レシピの指定に関連するエラー。
22	処理中に失敗。
23	処理中でない時に失敗。
24	材料不足による失敗。
25	ジョブの中断。
26	ジョブの停止。
27	ジョブの取り消し。
28	選択されたレシピは変更できない。
29	未定義イベント。
30	レポート ID の重複。
31	未定義データレポート。
32	データレポートがリンクされていない。
33	未定義トレースレポート
34	トレース ID の重複。
35	データレポートが多すぎる。
36	サンプル期間が範囲外。
37	グループサイズが大きすぎる。
38	回復アクションは現在無効。
39	要求した回復の実行を妨げる別の回復が現在実行中。
40	アクティブな回復アクションなし。
41	例外回復の失敗。
42	例外回復の中断。
43	無効表要素。
44	未定義表要素。
45	前もって設定済みのものは削除できない。
46	無効トークン。
47	無効パラメータ。
48~63	保留。

■型

```
u*
```

■関連メッセージ

S14F2 属性データ要求(GAD)
S15F22 レシピアクション確認
S15F28 レシピダウンロード確認
S15F30 レシピ検証確認
S15F32 レシピアンロードデータ
S15F36 レシピ削除確認

15.1.36 ERRTEXT

■説明

ERRCODE で示されるエラーを記述する文字列。最大 80 文字。

■型

```
a
```

■関連メッセージ

S14F2 属性データ要求(GAD)
S15F22 レシピアクション確認
S15F28 レシピダウンロード確認
S15F30 レシピ検証確認
S15F32 レシピアンロードデータ
S15F36 レシピ削除確認

15.1.37 ERRW7

■説明

プロセスプログラム内に見つかったエラーを示す文字列。

■型

```
a
```

■関連メッセージ

S7F27 プロセスプログラム妥当性送信(PVS)

15.1.38 GRANT

■説明

許可コード。1 バイト。

値	説明
0	許可。
1	ビジー。再試行。
2	受信スペースが不足。
3	*DATAID* の重複。
>3	装置固有のエラーコード。
4~63	保留。

■型

```
b[1]
```

■関連メッセージ

S2F40 マルチブロック許可(MBG)

15.1.39 GRANT6

■説明

送信許可。1 バイト。

値	説明
0	送信許可。

1	ビジー。リトライ要求。
2	不要。
>2	他のエラー。
3~63	保留。

■型

```
b[1]
```

■関連メッセージ

S6F6 マルチブロック許可(MBG)

15.1.40 HCACK

■説明

ホストコマンドパラメータ確認コード。1バイト。

値	説明
0	確認。コマンドは実行された。
1	コマンドは存在しない。
2	現在実行できない。
3	少なくとも1つのパラメータは無効。
4	確認。コマンドは実行され、イベントにより完了が知らされる。
5	拒否。既に要求された状態にある。
6	そのオブジェクトは存在しない。
7~63	保留。

■型

```
b[1]
```

■関連メッセージ

S2F42 ホストコマンド確認(HCA)
S2F50 拡張リモートコマンド確認

15.1.41 LENGTH

■説明

サービスプログラムまたはプロセスプログラムのバイト長。

■型

```
i*, u*
```

■関連メッセージ

S7F1 プロセスプログラムロード問い合わせ(PPI)
S7F29 プロセスプログラム妥当性問い合わせ(PVI)

15.1.42 LINKID

■説明

オペレーション実行の要求と完了メッセージをリンクするのに使用される。*LINKID* は最初の要求に含まれている *RMOPID* の値のにセットされる。例外は最後に送信されることになる完了メッセージで、その場合には0にセットされる。

■型

```
u4
```

■関連メッセージ

S15F22 レシピアクション確認
S15F30 レシピ検証確認

15.1.43 LRACK

■説明

リンク報告確認コード。1バイト。

値	説明
0	了解。
1	否定。スペースが不十分。
2	否定。無効フォーマット。
3	否定。少なくとも1つの *CEID* リンクが既に定義されている。
4	否定。少なくとも1つの *CEID* が存在しない。
5	否定。少なくとも1つの *RPTID* が存在しない。
>5	その他のエラー。
6~63	保留。

■型

```
b[1]
```

■関連メッセージ

S2F36 リンクイベントレポート確認(LERA)

15.1.44 MDLN

■説明

装置の形式。最大6バイト。

■型

```
a
```

■関連メッセージ

S1F2 オンラインデータ(D)
S1F13 通信確立要求(CR)
S1F14 通信確立要求確認(CRA)
S7F23 フォーマット付きプロセスプログラム送信(EPS)
S7F26 フォーマット付きプログラムデータ(FPD)
S7F27 プロセスプログラム妥当性送信(PVS)

15.1.45 MEXP

■説明

受信すべきストリーム／ファンクション。

■型

```
a
```

■関連メッセージ

S9F13 会話タイムアウト(CTN)

15.1.46 MHEAD

■説明

エラーになったメッセージヘッダ。

■型

```
b
```

■関連メッセージ

S9F1 未定義デバイス ID(UDN)
S9F3 未定義ストリームタイプ(USN)
S9F5 未定義ファンクションタイプ(UFN)
S9F7 不正データ(IDN)
S9F11 データが長すぎる(DLN)

15.1.47 OBJACK

■説明

確認コード。

値	説明
0	要求データのコマンド実行終了。
1	エラー。
>1	保留。

■型

```
u1
```

■関連メッセージ

S14F2 属性データ要求(GAD)

15.1.48 OBJID

■説明

オブジェクトのための識別子。

■型

```
a, u*
```

■関連メッセージ

S14F1 属性要求(GAR)
S14F2 属性データ要求(GAD)

15.1.49 OBJSPEC

■説明

内部フォーマットを持ち、特定のオブジェクトインスタンスを示すのに用いられるテキストストリング。このストリングは、フォーマットされたサブストリングの連続からなり、それぞれがオブジェクトタイプと識別子を特定している。サブストリングのフォーマットは以下の 4 つのフィールドからなる。

- オブジェクトタイプ
- コロン“:”
- オブジェクト識別子
- 不等号“>”

コロン“:”は、オブジェクトタイプの最後に用いられる。不等号“>”は、識別フィールドの最後に用いられる。オブジェクトタイプは他の方法でも決められるので省略しても良い。最後の“>”は任意である。

■型

```
a
```

■関連メッセージ

S2F49 拡張リモートコマンド
S14F1 属性要求(GAR)
S14F2 属性データ要求(GAD)
S15F21 レシピアクション要求
S15F28 レシピダウンロード確認
S15F29 レシピ検証要求
S15F30 レシピ検証確認
S15F35 レシピ削除要求

15.1.50 OBJTYPE

■説明

オブジェクトのグループあるいはクラスの識別子。同一タイプの全てのオブジェクトは同一の属性のセットを利用できるようにしなければならない。

■型

```
a, u*
```

■関連メッセージ

S14F1 属性要求(GAR)

15.1.51 OFLACK

■説明

オフライン要求に対する確認コード。

値	説明
0	オフライン了解。
1~63	保留。

■型

```
b
```

■関連メッセージ

S1F16 オフライン要求確認(OFLA)

15.1.52 ONLACK

■説明

オンライン要求に対する確認コード。

値	説明
0	オンライン了解。
1	オンラインを許可しない。
2	装置は既にオンラインである。
3~63	保留。

■型

```
b
```

■関連メッセージ

S1F18 オンライン要求確認(ONLA)

15.1.53 OPID

■説明

オペレーション ID。オペレーションの要求者が作成するユニークな整数で、複数の完了確認が発生する場合に使用される。

■型

```
u1
```

■関連メッセージ

S15F21 レシピアクション要求
S15F29 レシピ検証要求
S15F30 レシピ検証確認

15.1.54 PPARM

■説明

プロセスパラメータ。
処理コマンドを完了させるために必要な情報を与えるパラメータ。数値または真/偽値の SECS のデータアイテム。1 個または複数個の値または文字列。

■型

```
bool, a, i*, f*, u*
```

■関連メッセージ

S7F23 フォーマット付きプロセスプログラム送信(EPS)
S7F26 フォーマット付きプログラムデータ(FPD)
S7F27 プロセスプログラム妥当性送信(PVS)

15.1.55 PPBODY

■説明

プロセスプログラム本体。
装置が受け取る材料を処理するための動作を装置自身の言語で記述したものである。

■型

```
b, a, i*, u*
```

■関連メッセージ

S7F3 プロセスプログラム送信(PPS)
S7F6 プロセスプログラムデータ(PPD)

15.1.56 PPGNT

■説明

プロセスプログラムの許可状態。1 バイト。

値	説明
0	OK。
1	すでに持っている。
2	スペースなし。
3	無効 *PPID*
4	ビジー。リトライ要求。
5	不許可。
>5	他のエラー。
6~63	保留。

■型

```
b[1]
```

■関連メッセージ

S7F2 プロセスプログラムロード許可(PPG)
S7F30 プロセスプログラム妥当性許可(PVG)

15.1.57 PPID

■説明

プロセスプログラム ID。最大 80 バイト。*PPID* フォーマットは、ホストに依存する。装置内で使用する時は、*PPID* は 2 進のパターンとして扱われる。送られたコードをディスプレイするローカル機器がない場合は、16 進形になる。

■型

```
b, a
```

■関連メッセージ

S7F1 プロセスプログラムロード問い合わせ(PPI)
S7F2 プロセスプログラムロード許可(PPG)
S7F3 プロセスプログラム送信(PPS)
S7F4 プロセスプログラム確認(PPA)
S7F5 プロセスプログラム要求(PPR)
S7F6 プロセスプログラムデータ(PPD)
S7F17 プロセスプログラム削除指示(DPS)
S7F18 プロセスプログラム削除確認(DPA)
S7F19 現在の EPPD 要求(RER)
S7F20 現在の EPPD データ(RED)
S7F23 フォーマット付きプロセスプログラム送信(EPS)
S7F24 フォーマット付きプロセスプログラム確認(FPA)
S7F25 フォーマット付きプロセスプログラム要求(FPR)
S7F26 フォーマット付きプログラムデータ(FPD)
S7F27 プロセスプログラム妥当性送信(PVS)
S7F30 プロセスプログラム妥当性許可(PVG)

15.1.58 RCMD

■説明

リモコンコマンドコードまたはコマンド列。

■型

```
a
```

■関連メッセージ

S2F41 ホストコマンド送信(HCS)
S2F49 拡張リモートコマンド

15.1.59 RCPATTRDATA

■説明

レシピ属性の内容（値）。

■型

```
l, b, bool, a, i*, f*, u*
```

■関連メッセージ

S15F27 レシピダウンロード要求
S15F28 レシピダウンロード確認
S15F30 レシピ検証確認
S15F32 レシピアンロードデータ

15.1.60 RCPATTRID

■説明

非識別子属性の名称（識別子）。

■型

```
a
```

■関連メッセージ

S15F27 レシピダウンロード要求
S15F28 レシピダウンロード確認
S15F30 レシピ検証確認
S15F32 レシピアンロードデータ

15.1.61 RCPBODY

■説明

レシピ本体。

■型

```
b, a, i*, u*
```

■関連メッセージ

S15F27 レシピダウンロード要求
S15F32 レシピアンロードデータ

15.1.62 RCPCMD

■説明

レシピで実行されるアクションを表す。

値	説明
0~4	保留。
5	削除する。
6~7	保留。
8	保護しない。
9	保護する。
10	検証する。
11	リンクする。
12	リンクをはずす。
13	証明する。
14	証明を取り消す。
15	ダウンロードする。
16	アップロードする。
17~63	保留。

■型

```
u1
```

■関連メッセージ

S15F21 レシピアクション要求
S15F22 レシピアクション確認

15.1.63 RCPDEL

■説明

値	説明
0	削除。
1	選択解除。
>1	保留。

■型

```
u1
```

■関連メッセージ

S15F35 レシピ削除要求

15.1.64 RCPID

■説明

レシピ識別子。フォーマットされたテキストは *OBJSPEC* の要求事項に準拠している。

■型

```
a
```

■関連メッセージ

S15F21 レシピアクション要求
S15F28 レシピダウンロード確認
S15F29 レシピ検証要求
S15F30 レシピ検証確認
S15F35 レシピ削除要求

15.1.65 RCPOWCODE

■説明

ダウンロード時に、以前に存在したレシピが上書きされるかどうかを示す。

値	説明
TRUE	上書きされる。
FALSE	上書きされない。

■型

```
bool
```

■関連メッセージ

S15F27 レシピダウンロード要求

15.1.66 RCPSPEC

■説明

レシピ指定子。レシピのオブジェクト指定子。

■型

```
a
```

■関連メッセージ

S15F1 レシピ管理マルチブロック問い合わせ
S15F27 レシピダウンロード要求
S15F31 レシピアンロード要求
S15F32 レシピアンロードデータ

15.1.67 RESPEC

■説明

レシピエグゼキュータのオブジェクト指定子。

■型

```
a
```

■関連メッセージ

S15F29 レシピ検証要求
S15F35 レシピ削除要求

15.1.68 RMACK

■説明

要求されたアクションが成功して完了したか、拒否されたか、エラーで完了したか、あるいは要求者へ通知して完了するか、ということを伝達。

値	説明
0	成功して完了。
1	該当アクションを実行できない。
2	エラーで完了。
3	該当アクションは完了し通知は送信される。
4	当該アクションが存在することを要求しない。

■型

```
u1
```

■関連メッセージ

S15F22 レシピアクション確認
S15F28 レシピダウンロード確認
S15F30 レシピ検証確認
S15F32 レシピアンロードデータ
S15F36 レシピ削除確認

15.1.69 RMDATASIZE

■説明

マルチブロックメッセージの最大長をバイト数で表したもので、期待しているメッセージがレシーバの容量を越えているかどうかをレシーバが決めるのに使用される。

■型

```
u*
```

■関連メッセージ

S15F1 レシピ管理マルチブロック問い合わせ

15.1.70 RMGRANT

■説明

許可コード。要求を許可あるいは拒否するのに使われる。1 バイト。

値	説明
0	許可。
1	現時点では許可できない。再試行。
2	スペースなし。
3	要求待機。
4~64	保留。

■型

```
b[1]
```

■関連メッセージ

S15F2 レシピ管理マルチブロック許可

15.1.71 RMNSSPEC

■説明

レシピネームスペースのオブジェクト指定子。

■型

```
a
```

■関連メッセージ

S15F21 レシピアクション要求

15.1.72 RPTID

■説明

レポート ID。

■型

```
u4
```

■関連メッセージ

S2F33 規定レポート(DR)
S2F34 規定レポート確認(DRA)
S2F35 リンクイベントレポート(LER)
S2F36 リンクイベントレポート確認(LERA)
S6F11 イベントレポート送信(ERS)
S6F16 イベントレポートデータ(ERD)
S6F19 個別レポート要求(IRR)
S6F20 個別レポートデータ(IRD)

15.1.73 SEQNUM

■説明

コマンド番号。処理コマンドのリスト内での位置を示す番号でコマンドを指定する。プロセスプログラムの最初のコマンドは *SEQNUM* が 1 である。

■型

```
i*, u*
```

■関連メッセージ

S7F27 プロセスプログラム妥当性送信(PVS)

15.1.74 SHEAD

■説明

トランザクションタイマに関連したメッセージのヘッダ。

■型

```
b
```

■関連メッセージ

S9F9 トランザクションタイマタイムアウト(TIN)

15.1.75 SOFTREV

■説明

ソフトウェアのリビジョンコード。最大 6 バイト。

■型

```
a
```

■関連メッセージ

S1F2 オンラインデータ(D)
S1F13 通信確立要求(CR)
S1F14 通信確立要求確認(CRA)
S7F23 フォーマット付きプロセスプログラム送信(EPS)
S7F26 フォーマット付きプログラムデータ(FPD)
S7F27 プロセスプログラム妥当性送信(PVS)

15.1.76 SV

■説明

状態変数データ。

■型

```
l, b, bool, a, j, i*, f*, u*
```

■関連メッセージ

S1F4 指定装置状態データ(SSD)

15.1.77 SVID

■説明

状態変数 ID。状態変数には温度や消耗品の量等、時間単位でサンプリングされるあらゆるパラメータが含まれる。

■型

```
u4
```

■関連メッセージ

S1F3 指定装置状態要求(SSR)
S1F4 指定装置状態データ(SSD)
S1F11 状態変数名リスト要求(SVNR)
S1F12 状態変数名リスト応答(SVNRR)

15.1.78 SVNAME

■説明

状態変数名。

■型

```
a
```

■関連メッセージ

S1F12 状態変数名リスト応答(SVNRR)

15.1.79 TEXT

■説明

一行の文字。

■型

```
b, a, a2, i*, u*
```

■関連メッセージ

S10F1 端末要求(TRN)
S10F3 端末表示、シングルブロック(VTN)
S10F5 端末表示、マルチブロック(VTN)

15.1.80 TIACK

■説明

時間確認コード。1 バイト。

値	説明
0	OK。
1	エラー。了解されない。
2~63	保留。

■型

```
b[1]
```

■関連メッセージ

S2F32 日付及び時刻セット確認(DTA)

15.1.81 TID

■説明

端末番号。1 バイト。

値	説明
0	単一または主端末。
>0	同じ装置の付加端末。

■型

```
b[1]
```

■関連メッセージ

S10F1 端末要求(TRN)
S10F3 端末表示、シングルブロック(VTN)
S10F5 端末表示、マルチブロック(VTN)
S10F7 マルチブロック不許可(MNN)

15.1.82 TIME

■説明

日時。12 あるいは 16 バイト。

12 バイト		
YYMMDDhhmmss		
YY	年	00~99
MM	月	01~12
DD	日	01~31
hh	時	00~23
mm	分	00~59
ss	秒	00~59

16 バイト		
YYYYMMDDhhmmsscc		
YYYY	年	0000~9999
MM	月	01~12
DD	日	01~31
hh	時	00~23
mm	分	00~59
ss	秒	00~59
cc	1/100 秒	00~99

■注意

16 バイトフォーマットは現在のところオプションである。1998 年 1 月 1 日以降、新規あるいはアップデートされたインプリメンテーションには 16 バイトフォーマットが要求される。12 バイトフォーマットのサポートは装置定数 TimeFormat を使ってコンフィグレーション可能なオプションとして持続される。これはフォーマットの要求条件のみで、精度も正確さも意味していない。

■型

```
a
```

■関連メッセージ

S2F18 日付及び時刻データ(DTD)
S2F31 日付及び時刻セット要求(DTS)

15.1.83 UNITS

■説明

単位を認識するもの。E5、9 項（測定の単位）で許されたもの。

■型

```
a
```

■関連メッセージ

S1F12 状態変数名リスト応答(SVNRR)
S2F30 装置定数名リスト(ECN)

15.1.84 V

■説明

変数データ。

■型

```
l, b, bool, a, j, i*, f*, u*
```

■関連メッセージ

S6F11 イベントレポート送信(ERS)
S6F16 イベントレポートデータ(ERD)
S6F20 個別レポートデータ(IRD)

15.1.85 VID

■説明

変数 ID。

■型

```
u4
```

■関連メッセージ

S2F33 規定レポート(DR)
S2F34 規定レポート確認(DRA)

15.2 メッセージ

一次メッセージ	二次メッセージ	説明
S1F1	S1F2	オンライン確認要求
S1F3	S1F4	指定装置状態要求
S1F11	S1F12	状態変数名リスト要求
S1F13	S1F14	通信確立要求
S1F15	S1F16	オフライン要求
S1F17	S1F18	オンライン要求
S2F13	S2F14	装置定数要求
S2F15	S2F16	新装置定数変更
S2F17	S2F18	日付および時刻要求
S2F29	S2F30	装置定数名リスト要求
S2F31	S2F32	日付及び時刻セット要求
S2F33	S2F34	規定レポート
S2F35	S2F36	リンクイベントレポート
S2F37	S2F38	有効、無効イベントレポート
S2F39	S2F40	マルチブロック問い合わせ
S2F41	S2F42	ホストコマンド送信
S2F49	S2F50	拡張リモートコマンド
S5F1	S5F2	アラーム報告送信
S5F3	S5F4	アラーム報告有効/無効送信
S5F5	S5F6	アラームリスト要求
S6F5	S6F6	マルチブロックデータ送信問い合わせ
S6F11	S6F12	イベントレポート送信
S6F15	S6F16	イベントレポート要求
S6F19	S6F20	個別レポート要求
S7F1	S7F2	プロセスプログラムロード問い合わせ
S7F3	S7F4	プロセスプログラム送信
S7F5	S7F6	プロセスプログラム要求
S7F17	S7F18	プロセスプログラム削除指示
S7F19	S7F20	現在の EPPD 要求
S7F23	S7F24	フォーマット付きプロセスプログラム送信
S7F25	S7F26	フォーマット付きプロセスプログラム要求
S7F27	S7F28	プロセスプログラム妥当性送信
S7F29	S7F30	プロセスプログラム妥当性問い合わせ
S9F1		未定義デバイス ID
S9F3		未定義ストリームタイプ
S9F5		未定義ファンクションタイプ
S9F7		不正データ
S9F9		トランザクションタイマタイムアウト
S9F11		データが長すぎる
S9F13		会話タイムアウト
S10F1	S10F2	端末要求
S10F3	S10F4	端末表示、シングルブロック
S10F5	S10F6	端末表示、マルチブロック
S10F7		マルチブロック不許可
S14F1	S14F2	属性要求
S15F1	S15F2	レシピ管理マルチブロック問い合わせ
S15F21	S15F22	レシピアクション要求
S15F27	S15F28	レシピダウンロード要求
S15F29	S15F30	レシピ検証要求
S15F31	S15F32	レシピアンロード要求
S15F35	S15F36	レシピ削除要求

15.2.1 S1F1 オンライン確認要求(R)

```
Are You There Request
S,H←→E,返信
```

■説明

装置がオンラインかどうかを確認する。ファンクション 0 の応答があった時は通信が開始できない。装置がホストに S1F1 オンライン確認要求(R) を発信した後、ファンクション 0 の受信をしたなら受信タイムアウトの発生と同じ意味をもつ。

■構造

ヘッダのみ。

```
s1f1w
```

15.2.2 S1F2 オンラインデータ(D)

```
On Line Data
S,H←→E
```

■説明

オンラインであるという宣言。

■構造

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

名称	型	説明
MDLN	a	装置の形式。最大 6 バイト。
SOFTREV	a	ソフトウェアのリビジョンコード。最大 6 バイト。

■例外

ホストの場合には、レングスが 0 のリストを装置に送る。

```
s1f2
{
}
```

15.2.3 S1F3 指定装置状態要求(SSR)

```
Selected Equipment Status Request
S,H→E,返信
```

■説明

ホストが装置に対して指定した状態変数の値を要求する。

■構造

以下の構造は全てアイテムフォーマットに従っている。新しくインプリメントされるものについてはこの構造を使用すべきである。

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

次に示す構造は既にインプリメントされているものとの互換性を保つためのものである。

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

名称	型	説明
SVID	u4	状態変数 ID。状態変数には温度や消耗品の量等、時間単位でサンプリングされるあらゆるパラメータが含まれる。

■例外

レングスが 0 のアイテムの場合、すべての SVID の要求である。

```
s1f3w
{
}
```

```
s1f3w
<u4>
```

15.2.4 S1F4 指定装置状態データ(SSD)

```
Selected Equipment Status Data
M,H←E
```

■説明

装置は要求された順に各 SV の値を報告する。ホストは、どの SVID を要求したかを記憶しておくこと。

■構造

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

名称	型	説明
SV	l, b, bool, a, j, i*, f*, u*	状態変数データ。
SVID	u4	状態変数 ID。状態変数には温度や消耗品の量等、時間単位でサンプリングされるあらゆるパラメータが含まれる。

■例外

もしレングスが 0 のリストなら応答データが何もないことを意味する。

```
s1f4
{
}
```

SVi がレングスが 0 のアイテムの場合、SVIDi が存在しないことを意味する。

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

15.2.5 S1F11 状態変数名リスト要求 (SVNR)

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

■説明

ホストから装置に対する状態変数確認のための要求。

■構造

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

名称	型	説明
SVID	u4	状態変数 ID。状態変数には温度や消耗品の量等、時間単位でサンプリングされるあらゆるパラメータが含まれる。

■例外

レングスが 0 のリストの場合は、全ての SVID についての報告を要求する。

```
s1f11w
{
}
```

15.2.6 S1F12 状態変数名リスト応答 (SVNRR)

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

■説明

装置が要求された状態変数の名称と単位を報告する。

■構造

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

名称	型	説明
SVID	u4	状態変数 ID。状態変数には温度や消耗品の量等、時間単位でサンプリングされるあらゆるパラメータが含まれる。
SVNAME	a	状態変数名。
UNITSn	a	単位を識別するもの。E5、9 項（測定の単位）で許されたもの。

■例外

SVNAMEi と UNITSi の両方のレングスが 0 の文字列アイテムの場合はその SVID が存在しないことを表す。

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

15.2.7 S1F13 通信確立要求 (CR)

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

■説明

このメッセージの目的は、通信終了及び通信中断後のパワーオンという論理適用レベルにおいて、通信開始の正式な意味を与えることである。それは通信中断後に送信される最初のメッセージであるべきである。

確立を了解した確認応答コードを使用して S1F14 通信確立要求応答 (CRA) がトランザクションタイムアウト期間中に受信されるまでに、S1F13 通信確立要求 (CR) を送信しようとする試みは、プログラムされた間隔で繰り返されるべきである。

■構造

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

名称	型	説明
MDLN	a	装置の形式。最大 6 バイト。
SOFTREV	a	ソフトウェアのリビジョンコード。最大 6 バイト。

■例外

ホストはレングスが 0 のリストを装置に送る。

```
s1f13w
{
}
```

15.2.8 S1F14 通信確立要求確認(CRA)

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

■説明

S1F13 通信確立要求(CR)の了解または否定。*MDLN* 及び *SOFTREV* はオンラインデータであり、*COMMACK*=0 の場合のみ有効である。

■構造

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

名称	型	説明
COMMACK	b	通信確立確認コード。1 バイト。 0 了解。 1 否定。再試行。 2~63 保留。
MDLN	a	装置の形式。最大 6 バイト。
SOFTREV	a	ソフトウェアのリビジョンコード。最大 6 バイト。

■例外

ホストはレングスが 0 のリストを装置に送る。

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

15.2.9 S1F15 オフライン要求(ROFL)

```
Request OFF-LINE
S,H→E, 返信
```

■説明

ホストが装置にオフライン状態への移行を要求する。

■構造

ヘッダのみ。

```
s1f15w
```

15.2.10 S1F16 オフライン要求確認(OFLA)

```
OFF-LINE Acknowledge
S,H←E
```

■説明

S1F15 オフライン要求(ROFL)の OK または NG 応答。

■構造

```
s1f16
<b OFLACK>
```

名称	型	説明
OFLACK	b	オフライン要求に対する確認コード。 0 オフライン了解。 1~63 保留。

15.2.11 S1F17 オンライン要求(RONL)

```
Request ON-LINE
S,H→E, 返信
```

■説明

ホストが装置にオンライン状態への移行を要求する。

■構造

ヘッダのみ。

```
s1f17w
```

15.2.12 S1F18 オンライン要求確認(ONLA)

```
ON-LINE Acknowledge
S,H←E
```

■説明

S1F17 オンライン要求(RONL)の OK または NG 応答。

■構造

```
s1f18
<b ONLACK>
```

名称	型	説明
ONLACK	b	オンライン要求に対する確認コード。 0 オンライン了解。 1 オンラインを許可しない。 2 装置は既にオンラインである。

		3~63　保留。

15.2.13　S2F13 装置定数要求(ECR)

Equipment Constant Request
S,H→E, 返信

■説明

補正値、サーボゲイン、アラームの限界値、データ収集モードなど、ほとんど変化しない定数をこのメッセージで問い合わせる。

■構造

以下の構造は全てアイテムフォーマットに従っている。新しくインプリメントされるものについてはこの構造を使用すべきである。

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

次に示す構造は既にインプリメントされているものとの互換性を保つためのものである。

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

名称	型	説明
ECID	u4	装置定数 ID。

■例外

レングスが 0 のリスト（構造 1）またはアイテム（構造 2）の場合は、予め定義されている順番で全ての定数を要求する。

```
s2f13w
{
}
```

```
s2f13w
<u4>
```

15.2.14　S2F14 装置定数データ(ECD)

Equipment Constant Data
M,H←E

■説明

要求された順番で S2F13 装置定数要求(ECR) に対する応答定数が返る。

■構造

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

名称	型	説明
ECV	b, bool, a, j, i*, f*, u*	装置定数。

■例外

ECVi がレングスが 0 のリストの場合は当該 ECIDi がないことを意味する。このデータアイテムのリストフォーマットはこの場合を除き、禁止されている。

```
s2f14
{
   {
   }
}
```

15.2.15　S2F15 新装置定数変更(ECS)

New Equipment Constant Send
S,H→E,返信

■説明

1 つまたは複数の装置定数の変更。

■構造

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

名称	型	説明
ECID	u4	装置定数 ID。
ECV	b, bool, a, j, i*, f*, u*	装置定数。

15.2.16　S2F16 新装置定数変更確認(ECA)

New Equipment Constant Acknowledge
S,H←E

■説明

S2F15 新装置定数変更(ECS) の OK または NG 応答。EAC が 0 以外のエラーコードの場合、装置は S2F15 で指定された ECID のどの値も変更すべきではない。

■構造

```
s2f16
<b EAC>
```

名称	型	説明
EAC	b	装置確認コード。1 バイト。 0 了解。 1 否定。少なくとも 1 つの定数が存在しない。 2 否定。ビジー。 3 否定。少なくとも 1 つの定数が範囲外にある。 >3 他の装置固有のエラー。 4~63 保留。

15.2.17 S2F17 日付および時刻要求 (DTR)

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

■説明

装置のタイムベースチェックやホストタイムベースとの同期に利用する。

■構造

ヘッダのみ。

```
s2f17w
```

15.2.18 S2F18 日付及び時刻データ (DTD)

```
Date and Time Data
S,H←→E
```

■説明

現在の時刻。

■構造

```
s2f18
<a TIME>
```

名称	型	説明
TIME	a	日時。12 あるいは 16 バイト。 12 バイトならフォーマットは YYMMDDhhmmss 16 バイトならフォーマットは YYYYMMDDhhmmsscc

■例外

レングスが 0 のアイテムの場合は、時計をもっていないことを意味する。

```
s2f18
<a>
```

15.2.19 S2F29 装置定数名リスト要求 (ECNR)

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

■説明

ホストが、装置内で有効な装置定数に関する基本的な情報を収集する。

■構造

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

名称	型	説明
ECID	u4	装置定数 ID。

■例外

レングスが 0 の場合は、全ての ECID の情報を送ることを意味する。

15.2.20 S2F30 装置定数名リスト (ECN)

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

■説明

S2F29 装置定数名リスト要求 (ECNR) の応答。

■構造

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

名称	型	説明
ECID	u4	装置定数 ID。
ECNAME	a	装置定数名。
ECMIN	b, bool, a, j, i*, f*, u*	装置定数最小値。
ECMAX	b, bool, a, j, i*, f*, u*	装置定数最大値。
ECDEF	b, bool, a, j, i*, f*, u*	装置定数デフォルト値。

UNITS	a	単位を認識するもの。E5、9項（測定の単位）で許されたもの。

■例外

ECNAMEi、*ECMINi*、*ECMAXi*、*ECDEFi*、および *UNITSi* のレングスが 0 の文字列アイテムの場合はその *ECID* が存在しないことを表す。

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

15.2.21 S2F31 日付及び時刻セット要求(DTS)

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

■説明

装置タイムをホストコンピュータタイムベースに同期させるのに有用。

■構造

```
s2f31w
<a TIME>
```

名称	型	説明
TIME	a	日時。12 あるいは 16 バイト。 12 バイトならフォーマットは YYMMDDhhmmss 16 バイトならフォーマットは YYYYMMDDhhmmsscc

15.2.22 S2F32 日付及び時刻セット確認(DTA)

```
Date and Time Set Acknowledge
S,H←E
```

■説明

日付及び時刻セットの了解。

■構造

```
s2f32
<b TIACK>
```

名称	型	説明
TIACK	b	時間確認コード。1 バイト。 0 OK。 1 エラー。了解されない。 2~63 保留。

15.2.23 S2F33 規定レポート(DR)

```
Define Report
M,H→E, 返信
```

■説明

このメッセージの目的は、ホストコンピュータが、装置に対する一連のレポートを規定するためである。発信されるレポートのタイプは装置定数によってブーリアンで指定される。False の装置定数は S6F11 イベントレポート送信(ERS)が送信されることを意味し、True の装置定数は、S6F13 注釈付きイベントレポート送信(AERS)が送信されることを意味する[12]。もし、S2F33 がマルチブロックなら、S2F39 マルチブロック問い合わせ(DMBI)、S2F40 マルチブロック許可(MBG)トランザクションを先行しなければならない。

■構造

```
s2f33w
{
   <u4 DATAID>
   {
      {
         <u4 RPTID1>
         {
            <u4 VID1>
            .
            .
            <u4 VIDb>
         }
      }
      {
         <u4 RPTIDa>
         {
            <u4 VID1>
            .
            .
            <u4 VIDc>
         }
      }
   }
}
```

名称	型	説明
DATAID	u4	データ ID。
RPTID	u4	レポート ID。
VID	u4	変数 ID。

■例外

DATAID に続くレングスが 0 のリストは、全てのレポート規定及び関連リンクを消去する。S2F35 リンクイベントレポート(LER)参照。

```
s2f33w
{
   <u4 DATAID>
   {
   }
}
```

RPTID に続くレングスが 0 のリストは、レポートタイプ *RPTID* を消去する。この *RPTID* への全ての *CEID* リンクもまた消去される。

```
s2f33w
{
   <u4 DATAID>
   {
      {
         <u4 RPTID>
```

[12] GEM では S6F13 は使用しない。

```
        {
        }
      }
   }
}
```

名称	型	説明
CEID	u4	収集イベント ID。

15.2.24 S2F34 規定レポート確認 (DRA)

```
Define Report Acknowledge
S,H←E
```

■説明

了解またはエラー。エラー状態が検出されると、全てのメッセージは拒否される。すなわち、部分的な変更は許されない。

■構造

```
s2f34
<b DRACK>
```

名称	型	説明
DRACK	b	定義報告了解コード。1 バイト。 0 了解。 1 否定。スペース不十分。 2 否定。無効フォーマット。 3 否定。少なくとも 1 つの *RPTID* は既に定義されている。 4 否定。少なくとも 1 つの *VID* は存在しない。 >4 その他のエラー。 5~63 保留。
RPTID	u4	レポート ID。
VID	u4	変数 ID。

15.2.25 S2F35 リンクイベントレポート (LER)

```
Link Event Report
M,H→E, 返信
```

■説明

このメッセージの目的は、ホストコンピュータが、レポートを通知イベント ID (*CEID*) にリンクするためのものである。これらのリンクされたイベントレポートは、リンクされても履行せず無効となる。すなわち、イベントが発生しても有効な状態になるまでレポートは発信されない。

もし、S2F35 がマルチブロックなら、S2F39 マルチブロック問い合わせ (DMBI)、S2F40 マルチブロック許可 (MBG) トランザクションが先行しなければならない。

■構造

```
s2f35w
{
   <u4 DATAID>
   {
      {
         <u4 CEID1>
         {
            <u4 RPTID1>
            .
            .
            <u4 RPTIDb>
         }
      }
      .
      .
      {
         <u4 CEIDa>
         {
            <u4 RPTID1>
            .
            .
            <u4 RPTIDc>
         }
      }
   }
}
```

名称	型	説明
DATAID	u4	データ ID。
CEID	u4	収集イベント ID。
RPTID	u4	レポート ID。

■例外

CEID に続くレングスが 0 のリストは、そのイベントにリンクした全ての報告を消去する。

```
s2f35w
{
   <u4 DATAID>
   {
      {
         <u4 CEID>
         {
         }
      }
   }
}
```

15.2.26 S2F36 リンクイベントレポート確認 (LERA)

```
Link Event Report Acknowledge
S,H←E
```

■説明

了解またはエラー。エラー状態が検出されると、全てのメッセージは拒否される。すなわち、部分的な変更は許されない。

■構造

```
s2f36
<b LRACK>
```

名称	型	説明
LRACK	b	リンク報告確認コード。1 バイト。 0 了解。 1 否定。スペースが不十分。 2 否定。無効フォーマット。 3 否定。少なくとも 1 つの *CEID* リンクが既に定義されている。 4 否定。少なくとも 1 つの *CEID* が存在しない。 5 否定。少なくとも 1 つの *RPTID* が存在しない。 >5 その他のエラー。

		6~63 保留。
CEID	u4	収集イベント ID。
RPTID	u4	レポート ID。

15.2.27 S2F37 有効、無効イベントレポート (EDER)

Enable/Disable Event Report
S,H→E, 返信

■説明

このメッセージの目的は、ホストコンピュータが、通知イベント ID（*CEIDs*）に対する一連の報告を有効または無効にするためのものである。

■構造

```
s2f37w
{
   <bool CEED>
   {
      <u4 CEID1>
      .
      .
      <u4 CEIDn>
   }
}
```

名称	型	説明
CEED	bool	収集イベントあるいはトレース有効/無効コード。1 バイト。 偽　無効。 真　有効。
CEID	u4	収集イベント ID

■例外

CEED に続くレングスが 0 のリストは、全て *CEIDs* を意味する。

```
s2f37w
{
   <bool CEED>
   {
   }
}
```

15.2.28 S2F38 有効/無効イベントレポート確認 (EERA)

Enable/Disable Event Report Acknowledge
S,H←E

■説明

了解またはエラー。エラー状態が検出されると、全てのメッセージは拒否される。すなわち、部分的な変更は許されない。

■構造

```
s2f38
<b ERACK>
```

名称	型	説明
ERACK	b	有効/無効イベント報告。 確認コード。1 バイト。 0　了解。 1　否定。少なくとも 1 つの *CEID* が存在しない。 >1　その他のエラー。 2~63　保留。
CEID	u4	収集イベント ID

15.2.29 S2F39 マルチブロック問い合わせ (DMBI)

Multi-block Inquire
S,H→E, 返信

■説明

S2F23 トレース条件設定 (TIS)、S2F33 規定レポート (DR)、S2F35 リンクイベントレポート (LER)、S2F45 変数リミット属性定義 (DVLA)、あるいは S2F49 拡張リモートコマンドメッセージが 1 つ以上のブロックだった場合、このトランザクションが必ずメッセージの前にこなければならない。

■構造

```
s2f39w
{
   <u4 DATAID>
   <u4 DATALENGTH>
}
```

名称	型	説明
DATAID	u4	データ ID。
DATALENGTH	u4	送信データの総バイト数。

15.2.30 S2F40 マルチブロック許可 (MBG)

Multi-block Grant
S,H←E

■説明

マルチブロックメッセージの送信を許可こする。

■構造

```
s2f40
<b GRANT>
```

名称	型	説明
GRANT	b	許可コード。1 バイト。 0　許可。 1　ビジー。再試行。 2　受信スペースが不足。 3　*DATAID* の重複。 >3　装置固有のエラーコード。 4~63　保留。
DATAID	u4	データ ID。

15.2.31 S2F41 ホストコマンド送信 (HCS)

Host Command Send

S,H→E, 返信

■説明

装置が関連パラメータを持った特定のリモートコマンドを実行することをホストが要求する。

■構造

```
s2f41w
{
   <a RCMD>
   {
      {
         <a CPNAME1>
         <CPVAL1>
      }
      .
      .
      {
         <a CPNAMEn>
         <CPVALn>
      }
   }
}
```

名称	型	説明
RCMD	a	リモコンコマンドコードまたはコマンド列。
CPNAME	a	コマンドパラメータ名。
CPVAL	b, bool, a, j, i*, u*	コマンドパラメータ値。

15.2.32 S2F42 ホストコマンド確認(HCA)

```
Host Command Acknowledge
S,H←E
```

■説明

ホストコマンドまたはエラー確認。もし、1 つまたはそれ以上の無効パラメータのために、コマンドが受領されない場合には、パラメータ名及び無効である理由も含めて、無効パラメータのリストが返送される。

■構造

```
s2f42
{
   <b HCACK>
   {
      {
         <a CPNAME1>
         <b CPACK1>
      }
      .
      .
      {
         <a CPNAMEn>
         <b CPNAMEn>
      }
   }
}
```

名称	型	説明
HCACK	b	ホストコマンドパラメータ確認コード。1バイト。 0 確認。コマンドは実行された。 1 コマンドは存在しない。 2 現在実行できない。 3 少なくとも 1 つのパラメータは無効。 4 確認。コマンドは実行され、イベントにより完了が知らされる。 5 拒否。既に要求された状態にある。 6 そのオブジェクトは存在しない。 7~63 保留。
CPNAME	a	コマンドパラメータ名
CPACK	b	コマンドパラメータ確認コード。1 バイト。 1 パラメータ名（*CPNAME*）は存在しない。 2 *CPVAL* 用として指定された違法な値。 3 *CPVAL* 用として指定された違法なフォーマット。 >3 他の装置固有のエラー。 4~63 保留。

■例外

もし無効パラメータがない場合には、レングスが 0 のリストが送られる。

```
s2f42
{
   <b HCACK>
   {
   }
}
```

15.2.33 S2F49 拡張リモートコマンド

```
Enhanced Remote Command
M,H→E
```

■説明

ホストは指定したリモートコマンドに関連パラメータを付けて実行するようなオブジェクトに要求する。マルチブロックなら、S2F39 マルチブロック問い合わせ(DMBI)、S2F40 マルチブロック許可(MBG)トランザクションが先に行われる。

■構造

```
s2f49w
{
   <u4 DATAID>
   <a OBJSPEC>
   <a RCMD>
   {
      {
         <a CPNAME1>
         <CEPVAL1>
      }
      {
         <a CPNAME2>
         <CEPVAL2>
      }
      .
      .
      {
         <a CPNAMEm>
         <CEPVALm>
      }
   }
}
```

CPNAME の特定の値がリストと定義されている *CEPVAL* を持つ場合には、常にリストとなる。その特定の値の *CPNAME* に関連した *CEPVAL* がリス

ト以外と定義されている場合には、フォーマットエラーという結果になる。

名称	型	説明
DATAID	u4	データ ID。
OBJSPEC	a	内部フォーマットを持ち、特定のオブジェクトインスタンスを示すのに用いられるテキストストリング。このストリングは、フォーマットされたサブストリングの連続からなり、それぞれがオブジェクトタイプと識別子を特定している。サブストリングのフォーマットは以下の 4 つのフィールドからなる。 オブジェクトタイプ コロン ":" オブジェクト識別子 不等号 ">" コロン ":" は、オブジェクトタイプの最後に用いられる。不等号 ">" は、識別フィールドの最後に用いられる。オブジェクトタイプは他の方法でも決められるので省略しても良い。最後の ">" は任意である。
RCMD	a	リモコンコマンドコードまたはコマンド列。
CPNAME	a	コマンドパラメータ名。
CEPVAL	l, b, bool, a, j, i*, f*, u*	コマンド拡張パラメータ値。*CPNAME* 値が使われているので、*CEPVAL* が特定の使われ方をしている場合は常に分かる。*CEPVAL* には下記の形式がある： 単一（リストでない）の値（例：*CPVAL*）、同一フォーマット及びタイプの単一のアイテムリスト。あるいは形式のアイテムのリスト。 <L <CPNAME> <CEPVAL> >
CPVAL	b,bool,a,j,i*,u*	コマンドパラメータ値。

■例外

レングスが 0 のリストはコマンドと一緒にパラメータが送信されていないことを示す。

```
s2f49w
{
   <u4 DATAID>
   <a OBJSPEC>
   <a RCMD>
   {
   }
}
```

OBJSPEC はレングスが 0 のアイテムでもよい。

```
s2f49w
{
   <u4 DATAID>
   <a>
   <a RCMD>
   {
   }
}
```

■注意

CEPVAL がリストなら、そのリストのアイテムは下記の形式のどれか 1 つとなる。

- 同一のフォーマットを持つアイテムのリスト。
- 下記のような *CPNAME*、*CEPVAL* の組み合わせのリスト。

```
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>
            }
         }
      }
   }
}
```

15.2.34 S2F50 拡張リモートコマンド確認

```
Enhanced Remote Command Acknowledge
M,H←E
```

■説明

装置は拡張リモートコマンドを確認、あるいはエラーを報告する。1 つ以上の無効パラメータのためにコマンドが受諾されない場合（*HCACK*=3）には、無効パラメータのリストにパラメータ名及び無効の理由を付けて返す。

■構造

```
s2f50
{
   <b HCACK>
   {
      {
         <a CPNAME1>
```

```
        <u1 CEPACK1>
      }
      .
      .
      {
        <a CPNAMEn>
        <u1 CEPACKn>
      }
    }
}
```

名称	型	説明
HCACK	b	ホストコマンドパラメータ確認コード。1バイト。 0 確認。コマンドは実行された。 1 コマンドは存在しない。 2 現在実行できない。 3 少なくとも 1 つのパラメータは無効。 4 確認。コマンドは実行され、イベントにより完了が知らされる。 5 拒否。既に要求された状態にある。 6 そのオブジェクトは存在しない。 7~63 保留。
CPNAME	a	コマンドパラメータ名。
CEPACK	1,u1	コマンド拡張パラメータ承認。*CPNAME* の特定の値がリストの *CEPVAL* を持つと定義されている場合には、*CEPACK* は、S2F49[13]（拡張リモートコマンド）で使用されているように、対応する *CEPVAL* のリストフォーマットと同じ構造になる。それ以外、*CEPACK* は 1 バイトの整数になる。 0 エラーなし。 1 パラメータ名（*CPNAME*）が存在しない。 2 不正な値が *CEPVAL* に指定されている。 3 不正なフォーマットがCEPVALに指定されている。 4 パラメータ名（*CPNAME*）の使用法が有効でない。 5~63 保留。
CEPVAL	l, b, bool, a, j, i*, f*, u*	コマンド拡張パラメータ値。*CPNAME* 値が使われているので、*CEPVAL* が特定の使われ方をしている場合は常に分かる。*CEPVAL* には下記の形式がある：単一の（リストでない）値（例：*CPVAL*）、同一フォーマット及びタイプの単一のアイテムのリスト、あるいは形式のアイテムのリスト。 <L <CPNAME> <CEPVAL> >
CPVAL	b, bool, a, j, i*, u*	コマンドパラメータ値。

15.2.35 S5F1 アラーム報告送信(ARS)

```
Alarm Report Send
S,H←E, 返信
```

■説明

このメッセージでアラーム状態の発生または解除を通知する。このメッセージはアラームがセットされた時、またはアラームが解除された時、送信される。回復不能エラーと注意フラグには、対応する解除メッセージがなくてもよい。

■構造

```
s5f1w
{
   <b ALCD>
   <u4 ALID>
   <a ALTX>
}
```

名称	型	説明
ALCD	b	アラームコード。 bit8 1=アラーム状態発生。 0=アラーム状態解除。 bit7~1 アラーム区分コード。 0 未使用。 1 人間の安全に関わるもの。 2 装置の安全に関わるもの。 3 パラメータコントロールアラーム。 4 パラメータコントロールエラー。 5 回復不能エラー。 6 装置状態の警告。 7 注意フラグ。 8 データの保証不可。 >8 その他のカテゴリ。 9~63 保留。
ALID	u4	アラーム ID。
ALTX	a	アラームテキスト。最大 40 文字。

15.2.36 S5F2 アラーム報告確認(ARA)

```
Alarm Report Acknowledge
S,H→E
```

■説明

S5F1 アラーム報告送信(ARS)の OK または NG 応答。

■構造

```
s5f2
<b ACKC5>
```

名称	型	説明
ACKC5	b	確認コード。1 バイト。 0 了解。 >0 エラー。了解できず。 1~63 保留。

15.2.37 S5F3 アラーム報告有効/無効送信(EAS)

```
Enable/Disable Alarm Send
S,H→E, 返信
```

■説明

装置のアラーム通知の有効ビットのセット/リセットを行なう。装置では、このビットによりホストへアラーム通知を送信するかどうかを。判定す

[13] ?

る。この方法で制御できないアラームもある。

■構造

```
s5f3w
{
   <b ALED>
   <u4 ALID>
}
```

名称	型	説明
ALED	b	アラーム有効/無効コード、1 バイト。 bit8　1=アラーム有効。 bit8　0=アラーム無効。
ALID	u4	アラーム ID。

■例外

ALID のレングスが 0 のアイテムの場合は、全てのアラームのセット/リセットを意味する。

```
s5f3w
{
   <b ALED>
   <u4>
}
```

15.2.38　S5F4 アラーム報告有効/無効確認(EAA)

```
Enable/Disable Alarm Acknowledge
S,H←E
```

■説明

S5F3 アラーム報告有効/無効送信(EAS)の OK または NG 応答。

■構造

```
s5f4
<b ACKC5>
```

名称	型	説明
ACKC5	b	確認コード。1 バイト。 0　了解。 >0　エラー。了解できず。 1~63　保留。

15.2.39　S5F5 アラームリスト要求(LAR)

```
List Alarm Request
S,H→E, 返信
```

■説明

装置に対して、アラーム情報リストを送信するようにホストが要求する。

■構造

```
s5f5w
<u4 ALID1 ... ALIDn>
```

名称	型	説明
ALID	u4	アラーム ID。

■例外

レングスが 0 のアイテムの場合は、全てのアラームを要求している。

15.2.40　S5F6 アラームリストデータ(LAD)

```
List Alarm Data
M,H←E
```

■説明

装置の現在のアラーム状態（発生中/解除中）であり、アラームデータとしては複数ありえる。

■構造

```
s5f6
{
   {
      <b ALCD1>
      <u4 ALID1>
      <a ALTX1>
   }
   .
   .
   {
      <b ALCDm>
      <u4 ALIDm>
      <a ALTXm>
   }
}
```

名称	型	説明
ALCD	b	アラームコード。 bit8　1=アラーム状態発生。 bit8　0=アラーム状態解除。 bit7~1　アラーム区分コード。 0　未使用。 1　人間の安全に関わるもの。 2　装置の安全に関わるもの。 3　パラメータコントロールアラーム。 4　パラメータコントロールエラー。 5　回復不能エラー。 6　装置状態の警告。 7　注意フラグ。 8　データの保障不可。 >8　その他のカテゴリ。 9~63　保留。
ALID	u4	アラーム ID。
ALTX	a	アラームテキスト。最大 40 文字。

■例外

レングス m が 0 のリストの場合は、アラームデータなし。*ALCDi* または *ALTXi* のレングスが 0 のアイテムの場合は、当該アラームにデータがないことを意味する。

15.2.41　S6F5 マルチブロックデータ送信問い合わせ(MBI)

```
Multi-block Data Send Inquire
```

S,H←E, 返信

■説明

離散型データ報告 S6F3 離散型変数データ送信(DVS)、S6F9 フォーマット付き変数データ送信(FVS)、S6F11 イベントレポート送信(ERS)、S6F13 注釈付きイベントレポート送信(AERS)が複数ブロックを必要とする場合には、この処理は送信に先立って行なわなければならない。

■構造

```
s6f5W
{
   <? DATAID>
   <? DATALENGTH>
}
```

名称	型	説明
DATAID	u4	データ ID。
DATALENGTH	u4	送信データの総バイト数。

15.2.42 S6F6 マルチブロック許可(MBG)

Multi-block Grant
S,H→E

■説明

S6F5 マルチブロックデータ送信問い合わせ(MBI)の OK または NG 応答。

■構造

```
s6f6
<b GRANT6>
```

名称	型	説明
GRANT6	b	送信許可。1 バイト。 0 送信許可。 1 ビジー。リトライ要求。 2 不要。 >2 他のエラー。 3~63 保留。

15.2.43 S6F11 イベントレポート送信(ERS)

Event Report Send
M,H←E, 返信

■説明

このメッセージの目的は、装置がイベント発生と同時に、規定されている有効な一群のレポートをホストコンピュータに送ることである。(*CEID*)

もし、S6F11 がマルチブロックなら、S6F5 マルチブロックデータ送信問い合わせ(MBI)、S6F6 マルチブロック許可(MBG)トランザクションが先行しなければならない。[14]

■構造

```
s6f11w
{
   <u4 DATAID>
   <u4 CEID>
   {
      {
         <u4 RPTID1>
         {
            <V1>
            .
            .
            <Vp>
         }
      }
      .
      .
      {
         <u4 RPTIDa>
         {
            <V1>
            .
            .
            <Vc>
         }
      }
   }
}
```

名称	型	説明
DATAID	u4	データ ID。
CEID	u4	収集イベント ID
RPTID	u4	レポート ID
V	l, b, bool, a, j, i*, f*, u*	変数データ。

■例外

もし、イベントにリンクしたレポートがない場合には、レングスが 0 のリスト、a=0 となる。。レポートの番号に対するレングスが 0 のリストは与えられた *CEID* にリンクするレポートがないことを意味する。

15.2.44 S6F12 イベントレポート確認(ERA)

Event Report Acknowledge
S,H→E

■説明

了解またはエラー。

■構造

```
s6f12
<b ACKC6>
```

名称	型	説明
ACKC6	b	確認コード。1 バイト。 0 了解。 >0 エラー。了解できず。 1~63 保留。

15.2.45 S6F15 イベントレポート要求(ERR)

Event Report Request
S,H←→E, 返信

[14] ただし HSMS ではマルチブロックメッセージはないので、本製品では s6f5 は発生しない。

■説明

このメッセージは、ホストコンピュータが、与えられた一連のレポートを装置に要求するためのものである。

■構造

```
s6f15w
<u4 CEID>
```

名称	型	説明
CEID	u4	収集イベント ID。

15.2.46 S6F16 イベントレポートデータ(ERD)

```
Event Report Data
M,H←E, 返信
```

■説明

装置は与えられた *CEID* にリンクするレポートを送信する。

■構造

S6F11 イベントレポート送信(ERS)の構造と同じ。

```
s6f16w
{
   <u4 DATAID>
   <u4 CEID>
   {
      {
         <u4 RPTID1>
         {
            <V1>
            .
            .
            <Vp>
         }
      }
      .
      .
      {
         <u4 RPTIDa>
         {
            <V1>
            .
            .
            <Vc>
         }
      }
   }
}
```

名称	型	説明
DATAID	u4	データ ID。
CEID	u4	収集イベント ID。
RPTID	u4	レポート ID。
V	l, b, bool, a, j, i*, f*, u*	変数データ。

■例外

レングスが 0 のアイテムは与えられた *CEID* にリンクするレポートがないことを意味する。

15.2.47 S6F19 個別レポート要求(IRR)

```
Individual Report Request
S,H→E, 返信
```

■説明

このメッセージの目的は、ホストが装置からの定義されたレポートを要求するためのものである。

■構造

```
s5f19w
<u4 RPTID>
```

名称	型	説明
RPTID	u4	レポート ID。

15.2.48 S6F20 個別レポートデータ(IRD)

```
Individual Report Data
M,H←E
```

■説明

装置は、与えられた *RPTID* に対し定義された変数データをホストに送信する。

■構造

```
s5f20
{
   <V1>
   .
   .
   <Vn>
}
```

名称	型	説明
V	l, b, bool, a, j, i*, f*, u*	変数データ。

■例外

レングスが 0 のリストは、*RPTID* が定義されていないことを意味する。

名称	型	説明
RPTID	u4	レポート ID。

15.2.49 S7F1 プロセスプログラムロード問い合わせ(PPI)

```
Process Program Load Inquire
S,H←→E, 返信
```

■説明

本メッセージは、プロセスプログラムのロードまたはアンロード開始に使用される。S7F3 プロセスプログラム送信(PPS)、S7F4 プロセスプログラム確認(PPA)または S7F23 フォーマット付きプロセスプログラム送信(EPS)、S7F24 フォーマット付きプロセスプログラム確認(FPA)、S7F31 妥当性要求送信(VRS)、S7F32 妥当性要求確認(VRA)に先立っ

て使用される。

■構造

```
s7f1w
{
   <a PPID>
   <LENGTH>
}
```

名称	型	説明
PPID	a	プロセスプログラム ID。最大 80 バイト。*PPID* フォーマットは、ホストに依存する。装置内で使用する時は、*PPID* は 2 進のパターンとして扱われる。送られたコードをディスプレイするローカル機器がない場合は、16 進形になる。
LENGTH	i*, u*	サービスプログラムまたはプロセスプログラムのバイト長。

15.2.50 S7F2 プロセスプログラムロード許可(PPG)

```
Process Program Load Grant
S,H←→E
```

■説明

このメッセージは、プロセスプログラムのロード許可を与える。

■構造

```
s7f2
<b PPGNT>
```

名称	型	説明
PPGNT	b	プロセスプログラムの許可状態。1 バイト。 0 OK。 1 すでに持っている。 2 スペースなし。 3 無効 *PPID* 4 ビジー。リトライ要求。 5 不許可。 >5 他のエラー。 6~63 保留。

15.2.51 S7F3 プロセスプログラム送信(PPS)

```
Process Program Send
M,H←→E, 返信
```

■説明

プロセスプログラムの送信。もし、S7F3 がマルチブロックなら S7F1 プロセスプログラムロード問い合わせ(PPI)、S7F2 プロセスプログラムロード許可(PPG)トランザクションが先行しなければならない。

■構造

```
s7f3w
{
   <a PPID>
   <PPBODY>
}
```

名称	型	説明
PPID	a	プロセスプログラム ID。最大 80 バイト。*PPID* フォーマットは、ホストに依存する。装置内で使用する時は、*PPID* は 2 進のパターンとして扱われる。送られたコードをディスプレイするローカル機器がない場合は、16 進形になる。
PPBODY	b, a, i*, u*	プロセスプログラム本体。 装置が受け取る材料を処理するための動作を装置自身の言語で記述したものである。

15.2.52 S7F4 プロセスプログラム確認(PPA)

```
Process Program Acknowledge
S,H←→E
```

■説明

S7F3 プロセスプログラム送信(PPS)の OK または NG 応答。

■構造

```
s7f4
<b ACKC7>
```

名称	型	説明
ACKC7	b	確認コード。1 バイト。 0 許可された。 1 不許可。 2 レングスエラー。 3 配列オーバーフロー。 4 *PPID* 未定義。 5 モードエラー。 >5 他のエラー。 6~63 保留。

15.2.53 S7F5 プロセスプログラム要求(PPR)

```
Process Program Request
S,H←→E, 返信
```

■説明

プロセスプログラムの転送を要求する。

■構造

```
s7f5w
<a PPID>
```

名称	型	説明
PPID	a	プロセスプログラム ID。最大 80 バイト。*PPID* フォーマットは、ホストに依存する。装置内で使用する時は、*PPID* は 2 進のパターンとして扱われる。送られたコードをディスプレイするローカル機器がない場合は、16 進形になる。

15.2.54 S7F6 プロセスプログラムデータ(PPD)

```
Process Program Data
M,H←→E
```

■説明

プロセスプログラムの転送。

■構造

```
s7f6
{
   <a PPID>
   <PPBODY>
}
```

名称	型	説明
PPID	a	プロセスプログラム ID。最大 80 バイト。*PPID* フォーマットは、ホストに依存する。装置内で使用する時は、*PPID* は 2 進のパターンとして扱われる。送られたコードをディスプレイするローカル機器がない場合は、16 進形になる。
PPBODY	b, a, i*, u*	プロセスプログラム本体。装置が受け取る材料を処理するための動作を装置自身の言語で記述したものである。

■例外

レングスが 0 のリストは要求が拒否されたことを意味する。

■注意

ヘッダの R ビットを 1 とすることにより、装置側で作られたプロセスプログラムをホストへ転送できる。このことにより、ホスト側でその装置のプログラムが作成できなくても、その装置は使用可能となる。

15.2.55 S7F17 プロセスプログラム削除指示(DPS)

```
Delete Process Program Send
S,H→E, 返信
```

■説明

ホストより装置のプロセスプログラムの削除を要求する。

■構造

```
s7f17w
{
   <a PPID1>
   .
   .
   <a PPIDn>
}
```

名称	型	説明
PPID	a	プロセスプログラム ID。最大 80 バイト。*PPID* フォーマットは、ホストに依存する。装置内で使用する時は、*PPID* は 2 進のパターンとして扱われる。送られたコードをディスプレイするローカル機器がない場合は、16 進形になる。

■例外

レングスが 0 のリスト、n=0 の場合は、全てのプロセスプログラムを削除する。

15.2.56 S7F18 プロセスプログラム削除確認(DPA)

```
Delete Process Program Acknowledge
S,H←E
```

■説明

S7F17 プロセスプログラム削除指示(DPS)の OK または NG 応答。

■構造

```
s7f18
<b ACKC7>
```

名称	型	説明
ACKC7	b	確認コード。1 バイト。 0 許可された。 1 不許可。 2 レングスエラー。 3 配列オーバーフロー。 4 *PPID* 未定義。 5 モードエラー。 >5 他のエラー。 6~63 保留。
PPID	b, a	プロセスプログラム ID。最大 80 バイト。*PPID* フォーマットは、ホストに依存する。装置内で使用する時は、*PPID* は 2 進のパターンとして扱われる。送られたコードをディスプレイするローカル機器がない場合は、16 進形になる。

15.2.57 S7F19 現在の EPPD 要求(RER)

```
Current EPPD Request
S,H→E, 返信
```

■説明

本メッセージは、現在の装置プロセスプログラムのディレクトリ（EPPD）を要求するのに使用される。これは装置が記憶しているプロセスプログラムの全 *PPID* のリストである。

■構造

ヘッダのみ。

```
s7f19w
```

名称	型	説明
PPID	a	プロセスプログラム ID。最大 80 バイト。*PPID* フォーマットは、ホストに依存する。装置内で使用する時は、*PPID* は 2 進のパターンとして扱われる。送られたコードをディスプレイするローカル機器がない場合は、16 進形となる。

15.2.58 S7F20 現在のEPPD データ(RED)

```
Current EPPD Data
M,H←E
```

■説明

本メッセージは、現在のEPPD の伝送に使われる。

■構造

```
s7f20
{
   <a PPID1>
   .
   .
   <a PPIDn>
}
```

名称	型	説明
PPID	a	プロセスプログラム ID。最大 80 バイト。*PPID* フォーマットは、ホストに依存する。装置内で使用する時は、*PPID* は 2 進のパターンとして扱われる。送られたコードをディスプレイするローカル機器がない場合は。16 進形となる。

15.2.59 S7F23 フォーマット付きプロセスプログラム送信(EPS)

```
Formatted Process Program Send
M,H←→E, 返信
```

■説明

本メッセージでフォーマット付きのプロセスプログラムを装置とホスト間で伝送する。*MDLN* と *SOFTREV* の値は、プロセスプログラムの作成に使用した S7F22 装置処理能力データ(PCD)から得られる。もし、S7F23 がマルチブロックなら、S7F1 プロセスプログラムロード問い合わせ(PPI)、S7F2 プロセスプログラムロード許可(PPG)トランザクションが先行しなければならない。

■構造

```
s7f23w
{
   <a PPID>
   <a MDLN>
   <a SOFTREV>
   {
      {
         <CCODE>
         {
            <PPARM1>
            .
            .
            <PPARMp>
         }
      }
         <CCODEc>
         {
            <PPARM1>
            .
            .
            <PPARMq>
         }
      }
   }
}
```

名称	型	説明
PPID	a	プロセスプログラム ID。最大 80 バイト。*PPID* フォーマットは、ホストに依存する。装置内で使用する時は、*PPID* は 2 進のパターンとして扱われる。送られたコードをディスプレイするローカル機器がない場合は。16 進形となる。
MDLN	a	装置の形式。最大 6 バイト。
SOFTREV	a	ソフトウェアのリビジョンコード。最大 6 バイト。
CCODE	i2, u2	コマンドコード。 各コマンドコードは、機械が行なうことができる個別のプロセス操作に対応する。
PPARM	bool, a, i*, f*, u*	プロセスパラメータ。処理コマンドを完了させるために必要な情報を与えるパラメータ。数値または真/偽値の SECS のデータアイテム。1 個または複数個の値または文字列。

15.2.60 S7F24 フォーマット付きプロセスプログラム確認(FPA)

```
Formatted Process Program Acknowledge
S,H←→E
```

■説明

フォーマット付きのプロセスプログラムの受信とインタプリンタによるプロセスプログラムの受け入れ応答。インタプリンタによる“了解”の返信は、メッセージが理解されたことだけを意味する。プロセスプログラムの内容の妥当性については別のトランザクション S7F27 プロセスプログラム妥当性送信(PVS)、S7F28 プロセスプログラム妥当性確認(PVA)トランザクションによって通知される。

■構造

```
s7f24
<b ACKC7>
```

名称	型	説明
ACKC7	b	確認コード。1 バイト。 0 許可された。 1 不許可。 2 レングスエラー。 3 配列オーバーフロー。 4 *PPID* 未定義。 5 モードエラー。 >5 他のエラー。 6~63 保留。
PPID	a	プロセスプログラム ID。最大 80 バイト。*PPID* フォーマットは、ホストに依存する。装置内で使用する時は、*PPID* は 2 進のパターンとして扱われる。送られたコードをディスプレイするローカル機器がない場合は。16 進形となる。

15.2.61 S7F25 フォーマット付きプロセスプログラム要求(FPR)

```
Formatted Process Program Request
S,H←→E, 返信
```

■説明

本メッセージはホストまたは装置によって、プロセスプログラムの送信を要求するために使われる。

■構造

```
s7f25w
<a PPID>
```

名称	型	説明
PPID	a	プロセスプログラム ID。最大 80 バイト。*PPID* フォーマットは、ホストに依存する。装置内で使用する時は、*PPID* は 2 進のパターンとして扱われる。送られたコードをディスプレイするローカル機器がない場合は。16 進形となる。

15.2.62 S7F26 フォーマット付きプログラムデータ (FPD)

```
Formatted Process Program Data
M,H←→E
```

■説明

本メッセージで *PPID* の要求の返信としてプロセスプログラムを転送する。*MDLN* と *SOFTREV* の値は、プロセスプログラムの作成に使用した S7F22 装置処理能力データ (PCD) から得られる。

■構造

```
s7f26
{
   <a PPID>
   <a MDLN>
   <a SOFTREV>
   {
      {
         <CCODE>
         {
            <PPARM1>
            .
            .
            <PPARMp>
         }
      }
         <CCODEc>
         {
            <PPARM1>
            .
            .
            <PPARMq>
         }
      }
   }
}
```

名称	型	説明
PPID	a	プロセスプログラム ID。最大 80 バイト。*PPID* フォーマットは、ホストに依存する。装置内で使用する時は、*PPID* は 2 進のパターンとして扱われる。送られたコードをディスプレイするローカル機器がない場合は。16 進形となる。
MDLN	a	装置の形式。最大 6 バイト。
SOFTREV	a	ソフトウェアのリビジョンコード。最大 6 バイト。
CCODE	i2, u2	コマンドコード。各コマンドコードは、機械が行なうことができる個別のプロセス操作に対応する。
PPARM	bool, a, i*, f*, u*	プロセスパラメータ。処理コマンドを完了させるために必要な情報を与えるパラメータ。数値または真/偽値の SECS のデータアイテム。1 個または複数個の値または文字列。

■例外

レングスが 0 のリスト、c=0 の場合は、要求が拒否されたことを示す。

15.2.63 S7F27 プロセスプログラム妥当性送信 (PVS)

```
Process Program Verification Send
M,H←E, 返信
```

■説明

このメッセージは、プロセスプログラムが受信され装置によってチェックされたことを、ホストに対し示す。チェックの結果は、エラーのリストにより指定される。空エラーリスト（レングスが 0 のリスト、n=0）または 0 値を有する。*ACKC7A* を持つ単一要素リストは、プロセス中にエラーが見つからなかったことを示す。装置は、適切であると考えただけのエラーを報告することができる。装置はいかなるフォーマットされたプロセスプログラム S7F23 フォーマット付きプロセスプログラム送信 (EPS) または S7F26 フォーマット付きプログラムデータ (FPD) または S7F31 妥当性要求送信 (VRS)）を受け取った場合でも、ホストに対し、このメッセージのコピーを送る責任がある。もし、S7F27 がマルチブロックなら、S7F29 プロセスプログラム妥当性問い合わせ (PVI)）、S7F30 プロセスプログラム妥当性許可 (PVG) トランザクションがそれに先行しなければならない。

■構造

```
s7f27w
{
   <a PPID>
   {
      {
         <ACKC7A>
         <SEQNUM>
         <a ERRW7>
      }
      .
      .
      {
         <ACKC7An>
         <SEQNUMn>
         <a ERRW7n>
      }
   }
}
```

名称	型	説明
PPID	b, a	プロセスプログラム ID。最大 80 バイト。*PPID* フォーマットは、ホストに依存する。装置内で使用する時は、*PPID* は 2 進のパターンとして扱われる。送られたコードをディスプレイするローカル機器がない場合は。16 進形となる。
ACKC7A	i1, u1	確認コード。1 バイト。 0　了解された。 1　*MDLN* が一致しない。 2　*SOFTREV* が一致しない。

		3 無効な *CCODE*。 4 無効な *PPARM* の値。 5 他のエラー。(*ERRW7* によって示される) 6~63 保留。
SEQNUM	i*, u*	コマンド番号。処理コマンドのリスト内での位置を示す番号でコマンドを指定する。プロセスプログラムの最初のコマンドは *SEQNUM* が 1 である。
ERRW7	a	プロセスプログラム内に見つかったエラーを示す文字列。
MDLN	a	装置の形式。最大 6 バイト。
SOFTREV	a	ソフトウェアのリビジョンコード。最大 6 バイト。
CCODE	i2, u2	コマンドコード。各コマンドコードは、機械が行なうことができる個別のプロセス操作に対応する。
PPARM	bool, a, i*, f*, u*	プロセスパラメータ。処理コマンドを完了させるために必要な情報を与えるパラメータ。数値または真/偽値の SECS のデータアイテム。1 個または複数個の値または文字列。

15.2.64 S7F28 プロセスプログラム妥当性確認(PVA)

```
Process Program Verification Acknowledge
S,H→E
```

■説明

ホストによる装置への S7F27 プロセスプログラム妥当性送信(PVS)の受信を了解する応答。

■構造

ヘッダのみ。

```
s7f28
```

15.2.65 S7F29 プロセスプログラム妥当性問い合わせ(PVI)

```
Process Program Verification Inquire
S,H←E, 返信
```

■説明

本メッセージで、装置はホストにマルチブロックの S7F27 プロセスプログラム妥当性送信(PVS)を送る許可を問い合わせる。

■構造

```
s7f29w
<LENGTH>
```

名称	型	説明
LENGTH	i*, u*	サービスプログラムまたはプロセスプログラムのバイト長。

15.2.66 S7F30 プロセスプログラム妥当性許可(PVG)

```
Process Program Verification Grant
S,H→E
```

■説明

S7F29 プロセスプログラム妥当性問い合わせ(PVI)に対するホストから装置への返信。

■構造

```
s7f30
<b PPGNT>
```

名称	型	説明
PPGNT	b	プロセスプログラムの許可状態。1 バイト。 0 OK。 1 既に持っている。 2 スペースなし。 3 無効 *PPID*。 4 ビジー。リトライ要求。 5 不許可。 >5 他のエラー。 6~63 保留。
PPID	b, a	プロセスプログラム ID。最大 80 バイト。*PPID* フォーマットは、ホストに依存する。装置内で使用する時は、*PPID* は 2 進のパターンとして扱われる。送られたコードをディスプレイするローカル機器がない場合は、16 進形となる。

15.2.67 S9F1 未定義デバイス ID(UDN)

```
Unrecognized Device ID
S,H←E
```

■説明

メッセージブロックヘッダ内のデバイス ID がノード内で未定義である。

■構造

```
s9f1
<b MHEAD>
```

名称	型	説明
MHEAD	b	エラーになったメッセージヘッダ。

15.2.68 S9F3 未定義ストリームタイプ(USN)

```
Unrecognized Stream Type
S,H←E
```

■説明

メッセージブロックヘッダ内のストリームタイプが装置内で未定義である。

■構造

```
s9f3
<b MHEAD>
```

名称	型	説明
MHEAD	b	エラーになったメッセージヘッダ。

15.2.69 S9F5 未定義ファンクションタイプ(UFN)

```
Unrecognized Function Type
S,H←E
```

■説明

メッセージ ID 内のファンクションタイプが装置内で未定義である。

■構造

```
s9f5
<b MHEAD>
```

名称	型	説明
MHEAD	b	エラーになったメッセージヘッダ。

15.2.70 S9F7 不正データ(IDN)

```
Illegal Data
S,H←E
```

■説明

ストリームとファンクションは理解できたが、データフォーマットが解釈できない。

■構造

```
s9f7
<b MHEAD>
```

名称	型	説明
MHEAD	b	エラーになったメッセージヘッダ。

15.2.71 S9F9 トランザクションタイマタイムアウト(TIN)

```
Transaction Timer Timeout
S,H←E
```

■説明

トランザクション（T3）タイマがタイムアウトになり、処理中のトランザクションを強制終了したことを示す。システムを適切な運転状態に保つために、このエラーにどういう対応をするのかはホストが決定する。

■構造

```
s9f9
<b SHEAD>
```

名称	型	説明
SHEAD	b	トランザクションタイマに関連したメッセージのヘッダ。

15.2.72 S9F11 データが長すぎる(DLN)

```
Data Too Long
S,H←E
```

■説明

処理可能以上の長さのデータが装置に送られたことを示す。

■構造

```
s9f11
<b MHEAD>
```

名称	型	説明
MHEAD	b	エラーになったメッセージヘッダ。

15.2.73 S9F13 会話タイムアウト(CTN)

```
Conversation Timeout
S,H←E
```

■説明

データが受信されるはずなのに、適切な時間内に受信できなかった。リソースはクリアされる。

■構造

```
s9f13
{
   <a MEXP>
   <EDID>
}
```

名称	型	説明
MEXP	a	受信すべき SxxFyy。
EDID	b, a, i*, u*	受信すべきデータ ID。次の 3 つが考えられる。 MEXP EDID EDID S2F3 <SPID> a[6] S3F13 <PTN> b[1] S7F3 <PPID> a[16], b[16]

15.2.74 S10F1 端末要求(TRN)

```
Terminal Request
S,H←E, 返信
```

■説明

端末からホストへのテキストメッセージ。

■構造

```
s10f1w
{
   <b TID>
   <a TEXT>
}
```

名称	型	説明
TID	b	端末番号。1バイト。 0 単一または主端末。 >0 同じ装置の付加端末。
TEXT	a, a2	一行の文字。

15.2.75 S10F2 端末要求確認(TRA)

```
Terminal Request Acknowledge
S,H→E
```

■説明

S10F1 端末要求(TRN)のOKまたはNG応答。

■構造

```
s10f2
<b ACKC10>
```

名称	型	説明
ACKC10	b	確認コード。1バイト。 0 表示了解。 1 メッセージは表示されない。 2 端末使用できない。 3~63 保留。

15.2.76 S10F3 端末表示、シングルブロック(VTN)

```
Terminal Display,Single
S,H→E, 返信
```

■説明

表示されるデータ。

■構造

```
s10f3w
{
   <b TID>
   <a TEXT>
}
```

名称	型	説明
TID	b	端末番号。1バイト。 0 単一または主端末。 >0 同じ装置の付加端末。
TEXT	a, a2*	一行の文字。

15.2.77 S10F4 端末表示、シングルブロック確認(VTA)

```
Terminal Display,Single Acknowledge
S,H←E
```

■説明

S10F3 端末表示、シングルブロック(VTN)のOKまたはNG応答。

■構造

```
s10f4
<b ACKC10>
```

名称	型	説明
ACKC10	b	確認コード。1バイト。 0 表示了解。 1 メッセージは表示されない。 2 端末使用できない。 3~63 保留。

15.2.78 S10F5 端末表示、マルチブロック(VTN)

```
Terminal Display,Multi-block
M,H→E, 返信
```

■説明

表示されるデータ。

■構造

```
s10f5w
{
   <b TID>
   {
      <a TEXT1>
      .
      .
      <a TEXTn>
   }
}
```

名称	型	説明
TID	b	端末番号。1バイト。 0 単一または主端末。 >0 同じ装置の付加端末。
TEXT	a, a2	一行の文字。

15.2.79 S10F6 端末要求、マルチブロック確認(VMA)

```
Terminal Display,Multi-block Acknowledge
S,H←E
```

■説明

S10F5 端末表示、マルチブロック(VTN)のOKまたはNG応答。

■構造

```
s10f6
<b ACKC10>
```

名称	型	説明

ACKC10	b	確認コード。1バイト。 0　　表示了解。 1　　メッセージは表示されない。 2　　端末使用できない。 3~63　保留。

15.2.80　S10F7 マルチブロック不許可(MNN)

Multi-block Not Allowed
S,H←E

■説明

S10F5 端末表示、マルチブロック(VTN)のマルチブロックメッセージが処理できない端末からのエラーメッセージ。

■構造

```
s10f7
<b TID>
```

名称	型	説明
TID	b	端末番号。1バイト。 0　　単一または主端末。 >0　　同じ装置の付加端末。

15.2.81　S14F1 属性要求(GAR)

GetAttr Request
S,H←→E, 返信

■説明

このメッセージは、１つあるいはそれ以上のオブジェクトの特定の属性のセットを要求するのに使われる。これは以下のもので構成されている。ターゲットオブジェクト（インタレストなオブジェクト）の所有者のための指定子、ターゲットオブジェクトのタイプ、ターゲットオブジェクトの識別子のリスト、フィルタ（制約関係のリスト、つまり、これによってインタレストのターゲットオブジェクトをフィルタ内に含まれる全ての制約を満たすオブジェクトだけに限定する。）及び値を入れる必要のある特定の属性。

オブジェクト指定子は、ターゲットオブジェクトの所有者を指定するものである。これには、階層構造のオブジェクト関係のシーケンスも含まれている。オブジェクト指定子の各要素は、シーケンス内の次のオブジェクトインスタンスの高位にあるオブジェクトインスタンスを識別する。シーケンス内の最後のオブジェクトインスタンスは、ターゲットオブジェクトとの階層構造の関係の中にある。ターゲットオブジェクトタイプは、ターゲットオブジェクトのタイプを示している。また、オブジェクト識別子のリストは、インタレストであるそのオブジェクトタイプの特定インスタンスを示す。ターゲットタイプは、オブジェクト識別子がその他の全てのオブジェクトタイプと一致せず、識別子のリストが空いていない場合には、省略しても良い。

オブジェクトフィルタは、制約（インタレストであるオブジェクトインスタンスに適用する条件を与える。）の任意リストである。インタレストである各制約オブジェクトは、指定された制約を全て満たすオブジェクトである。

属性関係の数量子は、望ましいオブジェクトタイプの各インスタンスの対応する属性に対して指定された制約値 *ATTRDATAi* が所有する論理的バイナリ関係 *ATTRRELNi* である。このフィルタによって制限されるオブジェクトは、ステートメント“*ATTRDATAi*、*ATTRRLENi*、*Vi*”が真であるような属性値 *Vi* をもっている。*ATTRRLENi* が省略された場合、同等の関係であるとみなされる。

文字列の属性値 *ATTRDATAi* の場合、疑問符“?”とアスタリスク“*”の記号は、特定のオブジェクトタイプをフィルタにかけるためのワイルド文字として使用される。“?”記号は、任意の属性値あるいはASCII形式では重要な属性値において任意の1文字を表すために使用することができる。しかも繰り返し使用することが可能である。アスタリスク記号“*”もレングスが0の文字列を含む可変長ストリングを表すために疑問符記号“?”と同様に使用される。ストリング“*X”は、“X”で終了する可変長ストリングを表す。ストリング“X*”は、“X”で始まる可変長ストリングを表す。また、ストリング“*”は0以外の長さのストリングを表す。テキスト文字の比較において大文字小文字の区別はない。

特にワイルド文字をサポートするために、あるいは一般的に属性フィルタをサポートするために必要な装置はない。

■構造

```
s14f1w
{
   <a OBJSPEC>
   <OBJTYPE>
   {
      <OBJID1>
      .
      .
      <OBJIDi>
   }
   {
      {
         <ATTRID1>
         <ATTRDATA1>
         <u1 ATTRRELN1>
      }
      .
      .
      {
         <ATTRIDq>
         <ATTRDATAq>
         <u1 ATTRRELNq>
      }
   }
   {
      <ATTRID1>
      .
      .
      <ATTRIDa>
   }
}
```

名称	型	説明
OBJSPEC	a	内部フォーマットを持ち、特定のオブジェクトインスタンスを示すのに用いられるテキストストリング。このストリングは、フォーマットされたサブストリングの連続からなり、それぞれがオブジェクトタイプと識別子を特定している。サブストリングのフォーマットは以下の4つのフィールドからなる。 オブジェクトタイプ コロン“:” オブジェクト識別子 不等号“>” コロン“:”は、オブジェクトタイプの最後に用いられる。不等号“>”は、識別フィールドの最後に用いられる。オブジェクトタイプは他の方法でも決められるので省略しても良い。最後の“>”は任意である。
OBJTYPE	a, u*	オブジェクトのグループあるいはクラスの識別子。同一タイプの全てのオブジェクトは同一の属性のセットを利用できるようにしなければならない。

OBJID	a, u*	オブジェクトのための識別子。
ATTRID	a, u*	特定タイプのオブジェクトのため属性識別子。
ATTRDATA	l, b, bool, a, i*, f*, u*	特定のオブジェクトの特定属性値を持つ。
ATTRRELN	u1	特定の限定値とオブジェクトインスタンスの属性数値（インタレスト数値）との関係を規定する。 0 限定値がインタレストの値と等しい。 1 限定値がインタレストの値と等しくない。 2 限定値がインタレストの値より少ない。 3 限定値がインタレストの値より少ないか、等しい。 4 限定値がインタレストの値より大きい。 5 限定値がインタレストの値より大きいか、等しい。 6 限定値にはインタレストの値（セットに含まれる）がある。 7 限定値にはインタレストの値（セットに含まれない）がない。 >7 保留。
V	l, b, bool, a, j, i*, f*, u*	変数データ。

■例外

*OBJSPEC*が、レングスが0のアイテムの場合、オブジェクト指定子は用意されない。i=0の時は、フィルタだけが付加される。q=0の時は、フィルタは指定されない。iもqも0の場合、オブジェクトの全てのインスタンス情報が要求される。a=0の場合、全ての属性が要求される。

15.2.82 S14F2 属性データ要求(GAD)

GetAttr Data
M,H←→E

■説明

このメッセージは、特定のオブジェクトの要求された属性のセットを転送するのに、用いられる。属性の順番は、一次メッセージから保持される。

■構造

```
s14f2
{
   {
      {
         <OBJID1>
         {
            {
               <ATTRID1>
               <ATTRDATA1>
            }
            .
            .
            {
               <ATTRIDa>
               <ATTRDATAa>
            }
         }
      }
      .
      .
      {
         <OBJIDn>
         {
            {
               <ATTRID1>
               <ATTRDATA1>
            }
            .
            .
            {
               <ATTRIDb>
               <ATTRDATAb>
            }
         }
      }
   }
   {
      <u1 OBJACK>
      {
         {
            <ERRCODE1>
            <a ERRTEXT1>
         }
         .
         .
         {
            <ERRCODEp>
            <a ERRTEXTp>
         }
      }
   }
}
```

名称	型	説明
OBJID	a, u*	オブジェクトのための識別子。
ATTRID	a, u*	特定タイプのオブジェクトのため属性識別子。
ATTRDATA	l, b, bool, a, i*, f*, u*	特定のオブジェクトの特定属性値を持つ。
OBJACK	u1	確認コード。 0 要求データのコマンド実行終了。 1 エラー。 >1 保留。
ERRCODE	u*	エラー識別コード。 0 エラーなし。 1 オブジェクト指定子中のオブジェクト不明。 2 ターゲットオブジェクトタイプ不明。 3 オブジェクトインスタンス不明。 4 属性名称不明。 5 リードオンリー属性。アクセス拒否。 6 オブジェクトタイプ不明。 7 無効属性地。 8 シンタックスエラー。 9 検証エラー。 10 妥当性エラー。 11 オブジェクト指定子が使用中。 12 パラメータが正しく指定されていない。 13 指定すべきパラメータがすべて指定されていない。 14 要求されたオプションはサポートされていない。 15 使用中。 16 処理の準備ができていない。 17 現行の状態に無効なコマンド。

		18 変更された材料なし。 19 材料は部分的に処理された。 20 材料は全て処理された。 21 レシピの指定に関連するエラー。 22 処理中に失敗。 23 処理中でない時に失敗。 24 材料不足による失敗。 25 ジョブの中断。 26 ジョブの停止。 27 ジョブの取り消し。 28 選択されたレシピは変更できない。 29 未定義イベント。 30 レポート ID の重複。 31 未定義データレポート。 32 データレポートがリンクされていない。 33 未定義トレースレポート 34 トレース ID の重複。 35 データレポートが多すぎる。 36 サンプル期間が範囲外。 37 グループサイズが大きすぎる。 38 回復アクションは現在無効。 39 要求した回復の実行を妨げる別の回復が現在実行中。 40 アクティブな回復アクションなし。 41 例外回復の失敗。 42 例外回復の中断。 43 無効表要素。 44 未定義表要素。 45 前もって設定済みのものは削除できない。 46 無効トークン。 47 無効パラメータ。 48~63 保留。
ERRTEXT	a	*ERRCODE* で示されるエラーを記述する文字列。最大 80 文字。

■例外

OBJSPEC が、レングスが 0 のアイテムの場合、オブジェクト指定子は用意されない。n=0 の場合、特定のフィルタに適合するオブジェクトはない。p=0 の場合、エラーは検出されていない。

名称	型	説明
OBJSPEC	a	内部フォーマットを持ち、特定のオブジェクトインスタンスを示すのに用いられるテキストストリング。このストリングは、フォーマットされたサブストリングの連続からなり、それぞれがオブジェクトタイプと識別子を特定している。サブストリングのフォーマットは以下の 4 つのフィールドからなる。 オブジェクトタイプ コロン “:” オブジェクト識別子 不等号 “>” コロン “:” は、オブジェクトタイプの最後に用いられる。不等号 “>” は、識別フィールドの最後に用いられる。オブジェクトタイプは他の方法でも決められるので省略しても良い。最後の “>” は任意である。

15.2.83 S15F1 レシピ管理マルチブロック問い合わせ

```
Recipe Management Multi-block Inquire
S,H←→E, 返信
```

■説明

このメッセージは、マルチブロックメッセージ全体の最大メッセージ長に基づいて、そのマルチブロックメッセージを送信する許可を要求する。

■構造

```
s15f1w
{
   <u4 DATAID>
   <a RCPSPEC>
   <RMDATASIZE>
}
```

名称	型	説明
DATAID	u4	データ ID。
RCPSPEC	a	レシピ指定子。レシピのオブジェクト指定子。
RMDATASIZE	u*	マルチブロックメッセージの最大長をバイト数で表したもので、期待しているメッセージがレシーバの容量を越えているかどうかをレシーバが決めるのに使用される。

■例外

RCPSPEC がレングスが 0 の文字列の場合、送信許可の対象となるマルチブロックメッセージにはレシピは含まれていない。

15.2.84 S15F2 レシピ管理マルチブロック許可

```
Recipe Management Multi-block Grant
S,H←→E
```

■説明

このメッセージはマルチブロックメッセージの送信を許可する。あるいは拒否する。

■構造

```
s15f2
<b RMGRNT>
```

名称	型	説明
RMGRNT	b	許可コード。要求を許可あるいは拒否するのに使われる。1 バイト。 0 許可。 1 現時点では許可できない。再試行。 2 スペースなし。 3 要求待機。 4~64 保留。

15.2.85 S15F21 レシピアクション要求

```
Recipe Action Request
M,H←→E, 返信
```

■説明

このメッセージは、ネームスペース内にある1つ以上のレシピで実行される特定のアクションの要求を確認するのに使用される。

■構造

```
s15f21w
{
   <u4 DATAID>
   <u1 RCPCMD>
   <a RMNSSPEC>
   <u* OPID>
   <a AGENT>
   {
      <a RCPID1>
      .
      .
      <a RCPIDn>
   }
}
```

名称	型	説明
DATAID	u4	データ ID。
RCPCMD	u1	レシピで実行されるアクションを表す。 0~4 保留。 5 削除する。 6~7 保留。 8 保護しない。 9 保護する。 10 検証する。 11 リンクする。 12 リンクをはずす。 13 証明する。 14 証明を取り消す。 15 ダウンロードする。 16 アップロードする。 17~63 保留。
RMNSSPEC	a	レシピネームスペースのオブジェクト指定子。
OPID	u*	オペレーション ID。オペレーションの要求者が作成するユニークな整数で、複数の完了確認が発生する場合に使用される。
AGENT	a	
RCPID	a	レシピ識別子。フォーマットされたテキストは *OBJSPEC* の要求事項に準拠している。
OBJSPEC	a	内部フォーマットを持ち、特定のオブジェクトインスタンスを示すのに用いられるテキストストリング。このストリングは、フォーマットされたサブストリングの連続からなり、それぞれがオブジェクトタイプと識別子を特定している。サブストリングのフォーマットは以下の 4 つのフィールドからなる。 オブジェクトタイプ コロン “:” オブジェクト識別子 不等号 “>” コロン “:” は、オブジェクトタイプの最後に用いられる。不等号 “>” は、識別フィールドの最後に用いられる。オブジェクトタイプは他の方法でも決められるので省略しても良い。最後の “>” は任意である。

■例外

証明、証明取り消し、ダウンロード、アップロードの要求以外は *AGENT* はレングスが 0 の文字列になる。

15.2.86 S15F22 レシピアクション確認

Recipe Action Acknowledge

M,H←→E

■説明

このメッセージは新たなレシピを生成するようにという要求を確認するのに使用される。

■構造

```
s15f22
{
   <a AGENT>
   <u4 LINKID>
   <u1 RCPCMD>
   {
      <u1 RMACK>
      {
         {
            <ERRCODE1>
            <a ERRTEXT1>
         }
         .
         .
         {
            <ERRCODEp>
            <a ERRTEXTp>
         }
      }
   }
}
```

名称	型	説明
AGENT	a	
LINKID	u4	オペレーション実行の要求と完了メッセージをリンクするのに使用される。*LINKID* は最初の要求に含まれている *RMOPID* の値のにセットされる。例外は最後に送信されることになる完了メッセージで、その場合には 0 にセットされる。
RCPCMD	u1	レシピで実行されるアクションを表す。 0~4 保留。 5 削除する。 6~7 保留。 8 保護しない。 9 保護する。 10 検証する。 11 リンクする。 12 リンクをはずす。 13 証明する。 14 証明を取り消す。 15 ダウンロードする。 16 アップロードする。 17~63 保留。
RMACK	u1	要求されたアクションが成功して完了したか、拒否されたか、エラーで完了したか、あるいは要求者へ通知して完了するか、ということを伝達。 0 成功して完了。 1 該当アクションを実行できない。 2 エラーで完了。 3 該当アクションは完了し通知は送信される。 4 当該アクションが存在することを要求しない。
ERRCODE	u*	エラー識別コード。 0 エラーなし。 1 オブジェクト指定子中のオブジェクト不明。 2 ターゲットオブジェクトタイプ不明。 3 オブジェクトインスタンス不明。

		4 属性名称不明。 5 リードオンリー属性。アクセス拒否。 6 オブジェクトタイプ不明。 7 無効属性地。 8 シンタックスエラー。 9 検証エラー。 10 妥当性エラー。 11 オブジェクト指定子が使用中。 12 パラメータが正しく指定されていない。 13 指定すべきパラメータがすべて指定されていない。 14 要求されたオプションはサポートされていない。 15 使用中。 16 処理の準備ができていない。 17 現行の状態に無効なコマンド。 18 変更された材料なし。 19 材料は部分的に処理された。 20 材料は全て処理された。 21 レシピの指定に関連するエラー。 22 処理中に失敗。 23 処理中でない時に失敗。 24 材料不足による失敗。 25 ジョブの中断。 26 ジョブの停止。 27 ジョブの取り消し。 28 選択されたレシピは変更できない。 29 未定義イベント。 30 レポート ID の重複。 31 未定義データレポート。 32 データレポートがリンクされていない。 33 未定義トレースレポート 34 トレース ID の重複。 35 データレポートが多すぎる。 36 サンプル期間が範囲外。 37 グループサイズが大きすぎる。 38 回復アクションは現在無効。 39 要求した回復の実行を妨げる別の回復が現在実行中。 40 アクティブな回復アクションなし。 41 例外回復の失敗。 42 例外回復の中断。 43 無効表要素。 44 未定義表要素。 45 前もって設定済みのものは削除できない。 46 無効トークン。 47 無効パラメータ。 48~63 保留。
ERRTEXT	a	*ERRCODE* で示されるエラーを記述する文字列。最大 80 文字。

■例外

要求されたアクションが全て完了した場合のみ *LINKID* は 0 である。*RMACK* がエラーなしを表している場合のみ、p=0。

15.2.87 S15F27 レシピダウンロード要求

```
Recipe Download Request
M,H→E, 返信
```

■説明

このメッセージはレシピエグゼキュータにレシピを送信するのに使用される。この場合には、前に S15F1 レシピ管理マルチブロック問い合わせ、S15F2 レシピ管理マルチブロック許可トランザクションがある。

■構造

```
s15f27w
{
   <u4 DATAID>
   <bool RCPOWCODE>
   <a RCPSPEC>
   {
      {
         <a RCPATTRID1>
         <RCPATTRDATA1>
      }
      .
      .
      {
         <a RCPATTRIDm>
         <RCPATTRDATAm>
      }
   }
   <RCBODY>
}
```

名称	型	説明
DATAID	u4	データ ID。
RCPOWCODE	bool	ダウンロード時に、以前に存在したレシピが上書きされるか（=TRUE）、上書きされないか（=FALSE）を示す。
RCPSPEC	a	レシピ指定子。レシピのオブジェクト指定子。
RCPATTRID	a	非識別子属性の名称（識別子）。
RCPATTRDATA	l, b, bool, a, i*, f*, u*	レシピ属性の内容（値）。
RCBODY	b, a, i*, u*	レシピ本体。

15.2.88 S15F28 レシピダウンロード確認

```
Recipe Download Acknowledge
M,H←E
```

■説明

このメッセージはレシピエグゼキュータがレシピを受信したということを確認するのに使用される。レシピの検証が成功したら結果がセンターに返される。オブジェクト形式の派生レシピが検証の間に生成された場合、*RCPID* にはその派生レシピの識別子が含まれる。

■構造

```
s15f28
{
   <a RCPID>
   {
      {
         <a RCPATTRID1>
         <RCPATTRDATA1>
      }
      .
      .
      {
         <ERRCODE1>
         <a ERRTEXT1>
      }
      .
      .
```

```
    {
      <ERRCODEp>
      <a ERRTEXTp>
    }
  }
}
```

名称	型	説明
RCPID	a	レシピ識別子。フォーマットされたテキストは *OBJSPEC* の要求事項に準拠している。
OBJSPEC	a	内部フォーマットを持ち、特定のオブジェクトインスタンスを示すのに用いられるテキストストリング。このストリングは、フォーマットされたサブストリングの連続からなり、それぞれがオブジェクトタイプと識別子を特定している。サブストリングのフォーマットは以下の 4 つのフィールドからなる。 オブジェクトタイプ コロン “:” オブジェクト識別子 不等号 “>” コロン “:” は、オブジェクトタイプの最後に用いられる。不等号“>”は、識別フィールドの最後に用いられる。オブジェクトタイプは他の方法でも決められるので省略しても良い。最後の “>” は任意である。
RCPATTRID	a	非識別子属性の名称（識別子）。
RCPATTRDATA	l, b, bool, a, i*, f*, u*	レシピ属性の内容（値）。
RMACK	u1	要求されたアクションが成功して完了したか、拒否されたか、エラーで完了したか、あるいは要求者へ通知して完了するか、ということを伝達。 0 成功して完了。 1 該当アクションを実行できない。 2 エラーで完了。 3 該当アクションは完了し通知は送信される。 4 当該アクションが存在することを要求しない。
ERRCODE	u*	エラー識別コード。 0 エラーなし。 1 オブジェクト指定子中のオブジェクト不明。 2 ターゲットオブジェクトタイプ不明。 3 オブジェクトインスタンス不明。 4 属性名称不明。 5 リードオンリー属性。アクセス拒否。 6 オブジェクトタイプ不明。 7 無効属性地。 8 シンタックスエラー。 9 検証エラー。 10 妥当性エラー。 11 オブジェクト指定子が使用中。 12 パラメータが正しく指定されていない。 13 指定すべきパラメータがすべて指定されていない。 14 要求されたオプションはサポートされていない。 15 使用中。 16 処理の準備ができていない。 17 現行の状態に無効なコマンド。 18 変更された材料なし。 19 材料は部分的に処理された。 20 材料は全て処理された。 21 レシピの指定に関連するエラー。 22 処理中に失敗。 23 処理中でない時に失敗。 24 材料不足による失敗。 25 ジョブの中断。 26 ジョブの停止。 27 ジョブの取り消し。 28 選択されたレシピは変更できない。 29 未定義イベント。 30 レポート ID の重複。 31 未定義データレポート。 32 データレポートがリンクされていない。 33 未定義トレースレポート 34 トレース ID の重複。 35 データレポートが多すぎる。 36 サンプル期間が範囲外。 37 グループサイズが大きすぎる。 38 回復アクションは現在無効。 39 要求した回復の実行を妨げる別の回復が現在実行中。 40 アクティブな回復アクションなし。 41 例外回復の失敗。 42 例外回復の中断。 43 無効表要素。 44 未定義表要素。 45 前もって設定済みのものは削除できない。 46 無効トークン。 47 無効パラメータ。 48~63 保留。
ERRTEXT	a	*ERRCODE* で示されるエラーを記述する文字列。最大 80 文字。

■例外

レングスが 0 のアイテムの場合、オブジェクト形式の派生レシピは作られない。レシピが検証されなかった場合、あるいは検証に失敗した場合のみ、n=0。*RMACK* がエラーなしを表している場合のみ、P=0。

15.2.89 S15F29 レシピ検証要求

```
Recipe Verify Request
M,H→E, 返信
```

■説明

このメッセージは 1 つあるいは複数のレシピを検証するようレシピエグゼキュータが要求するのに使用される。マルチブロックの場合には、S15F1 レシピ管理マルチブロック問い合わせ、S15F2 レシピ管理マルチブロック許可トランザクションがこの前にある。オペレーション識別子 *OPID* は複数の検証要求が未処理の場合に使用され、レシピエグゼキュータが現在来ている検証要求を全部完了するまで、それ以上検証を要求されない場合、0 になる。それ以外の場合には *OPID* はリクエスターごとにユニークなものになる。*RESPEC* はレシピエグゼキューターのオブジェクト指定子である。

■構造

```
s15f29w
{
   <u4 DATAID>
   <OPID>
   <a RESPEC>
   {
      <a RCPID1>
      .
      .
      <a RCPIDm>
   }
}
```

名称	型	説明
DATAID	u4	データ ID。
OPID	u*	オペレーション ID。オペレーションの要求者が作成するユニークな整数で、複数の完了確認が発生する場合に使用される。
RESPEC	a	レシピエグゼキュータのオブジェクト指定子。
RCPID	a	レシピ識別子。フォーマットされたテキストは *OBJSPEC* の要求事項に準拠している。
OBJSPEC	a	内部フォーマットを持ち、特定のオブジェクトインスタンスを示すのに用いられるテキストストリング。このストリングは、フォーマットされたサブストリングの連続からなり、それぞれがオブジェクトタイプと識別子を特定している。サブストリングのフォーマットは以下の 4 つのフィールドからなる。 オブジェクトタイプ コロン “:” オブジェクト識別子 不等号 “>” コロン “:” は、オブジェクトタイプの最後に用いられる。不等号 “>” は、識別フィールドの最後に用いられる。オブジェクトタイプは他の方法でも決められるので省略しても良い。最後の “>” は任意である。

■例外

RESPEC がレングスが 0 のアイテムの場合、ターゲットはメッセージの受取人である。

15.2.90 S15F30 レシピ検証確認

```
Recipe Verify Acknowledge
M,H←E
```

■説明

このメッセージは 1 つあるいは複数のレシピの検証要求を確認するのに使用される。レシピを 1 つ検証するよう要求されてその検証が成功した場合には、このメッセージで結果がセンターに返される。オブジェクト形式の派生レシピが検証中に生成された場合にはその派生レシピの識別子が RCPID に含まれる。複数のレシピの検証が要求された場合には *LINKID* は 0 以外になる。

■構造

```
s15f30
{
   <OPID>
   <u4 LINKID>
   <A RCPID>
   {
      {
         <a RCPATTRID1>
         <RCPATTRDATA1>
      }
      .
      .
      {
         <a RCPATTRIDn>
         <RCPATTRDATAn>
      }
   }
   {
      <u1 RMACK>
      {
         {
            <ERRCODE1>
            <a ERRTEXT1>
         }
         .
         .
         {
            <ERRCODEp>
            <a ERRTEXTp>
         }
      }
   }
}
```

名称	型	説明
OPID	u*	オペレーション ID。オペレーションの要求者が作成するユニークな整数で、複数の完了確認が発生する場合に使用される。
LINKID	u4	オペレーション実行の要求と完了メッセージをリンクするのに使用される。*LINKID* は最初の要求に含まれている *RMOPID* の値にセットされる。例外は最後に送信されることになる完了メッセージで、その場合には 0 にセットされる。
RCPID	a	レシピ識別子。フォーマットされたテキストは *OBJSPEC* の要求事項に準拠している。
RCPATTRID	a	非識別子属性の名称（識別子）。
RCPATTRDATA	l, b, bool, a, i*, f*, u*	レシピ属性の内容（値）。
RMACK	u1	要求されたアクションが成功して完了したか、拒否されたか、エラーで完了したか、あるいは要求者へ通知して完了するか、ということを伝達。 0 成功して完了。 1 該当アクションを実行できない。 2 エラーで完了。 3 該当アクションは完了し通知は送信される。 4 当該アクションが存在することを要求しない。
ERRCODE	u*	エラー識別コード。 0 エラーなし。

		1 オブジェクト指定子中のオブジェクト不明。 2 ターゲットオブジェクトタイプ不明。 3 オブジェクトインスタンス不明。 4 属性名称不明。 5 リードオンリー属性。アクセス拒否。 6 オブジェクトタイプ不明。 7 無効属性地。 8 シンタックスエラー。 9 検証エラー。 10 妥当性エラー。 11 オブジェクト指定子が使用中。 12 パラメータが正しく指定されていない。 13 指定すべきパラメータがすべて指定されていない。 14 要求されたオプションはサポートされていない。 15 使用中。 16 処理の準備ができていない。 17 現行の状態に無効なコマンド。 18 変更された材料なし。 19 材料は部分的に処理された。 20 材料は全て処理された。 21 レシピの指定に関連するエラー。 22 処理中に失敗。 23 処理中でない時に失敗。 24 材料不足による失敗。 25 ジョブの中断。 26 ジョブの停止。 27 ジョブの取り消し。 28 選択されたレシピは変更できない。 29 未定義イベント。 30 レポート ID の重複。 31 未定義データレポート。 32 データレポートがリンクされていない。 33 未定義トレースレポート 34 トレース ID の重複。 35 データレポートが多すぎる。 36 サンプル期間が範囲外。 37 グループサイズが大きすぎる。 38 回復アクションは現在無効。 39 要求した回復の実行を妨げる別の回復が現在実行中。 40 アクティブな回復アクションなし。 41 例外回復の失敗。 42 例外回復の中断。 43 無効表要素。 44 未定義表要素。 45 前もって設定済みのものは削除できない。 46 無効トークン。 47 無効パラメータ。

		48~63 保留。
ERRTEXT	a	*ERRCODE* で示されるエラーを記述する文字列。最大 80 文字。
OBJSPEC	a	内部フォーマットを持ち、特定のオブジェクトインスタンスを示すのに用いられるテキストストリング。このストリングは、フォーマットされたサブストリングの連続からなり、それぞれがオブジェクトタイプと識別子を特定している。サブストリングのフォーマットは以下の 4 つのフィールドからなる。 オブジェクトタイプ コロン “:” オブジェクト識別子 不等号 “>” コロン “:” は、オブジェクトタイプの最後に用いられる。不等号 “>” は、識別フィールドの最後に用いられる。オブジェクトタイプは他の方法でも決められるので省略しても良い。最後の “>” は任意である。

■例外

レシピを 1 つだけ検証するよう要求されて完了した場合のみ、*LINKID* は 0 になる。アイテム 3 がレングスが 0 のアイテムなら、オブジェクト形式の派生レシピは作られていない。レシピが検証されなかった場合あるいはレシピの検証が失敗した場合のみ、n=0。*RMACK* がエラーなしを表している場合のみ、p=0。

15.2.91 S15F31 レシピアンロード要求

```
Recipe Unload Request
S,H→E, 応答
```

■説明

このメッセージはレシピエグゼキュータに実行レシピを要求するのに使用される。

■構造

```
s15f31
<a RCSPEC>
```

名称	型	説明
RCPSPEC	a	レシピ指定子。レシピのオブジェクト指定子。

15.2.92 S15F32 レシピアンロードデータ

```
Recipe Unload Data
M,H←E
```

■説明

このメッセージはレシピエグゼキュータから実行レシピを送信するのに使用される。

■構造

```
s15f32
{
   <a RCPSPEC>
   {
      {
         <a RCPATTRID1>
         <RCPATTRDATA1>
      }
      .
      .
      {
         <a RCPATTRIDm>
         <RCPATTRDATAm>
      }
   }
   <RCPBODY>
   {
      <u1 RMACK>
      {
         {
            <ERRCODE1>
            <a ERRTEXT1>
         }
         .
         .
         {
            <ERRCODEp>
            <a ERRTEXTp>
         }
      }
   }
}
```

名称	型	説明
RCPSPEC	a	レシピ指定子。レシピのオブジェクト指定子。
RCPATTRID	a	非識別子属性の名称（識別子）。
RCPATTRDATA	l, b, bool, a, i*, f*, u*	レシピ属性の内容（値）。
RCPBODY	b, a, i*, u*	レシピ本体。
RMACK	u1	要求されたアクションが成功して完了したか、拒否されたか、エラーで完了したか、あるいは要求者へ通知して完了するか、ということを伝達。 0 成功して完了。 1 該当アクションを実行できない。 2 エラーで完了。 3 該当アクションは完了し通知は送信される。 4 当該アクションが存在することを要求しない。
ERRCODE	u*	エラー識別コード。 0 エラーなし。 1 オブジェクト指定子中のオブジェクト不明。 2 ターゲットオブジェクトタイプ不明。 3 オブジェクトインスタンス不明。 4 属性名称不明。 5 リードオンリー属性。アクセス拒否。 6 オブジェクトタイプ不明。 7 無効属性地。 8 シンタックスエラー。 9 検証エラー。 10 妥当性エラー。 11 オブジェクト指定子が使用中。 12 パラメータが正しく指定されていない。 13 指定すべきパラメータがすべて指定されていない。 14 要求されたオプションはサポートされていない。 15 使用中。 16 処理の準備ができていない。 17 現行の状態に無効なコマンド。 18 変更された材料なし。 19 材料は部分的に処理された。 20 材料は全て処理された。 21 レシピの指定に関連するエラー。 22 処理中に失敗。 23 処理中でない時に失敗。 24 材料不足による失敗。 25 ジョブの中断。 26 ジョブの停止。 27 ジョブの取り消し。 28 選択されたレシピは変更できない。 29 未定義イベント。 30 レポート ID の重複。 31 未定義データレポート。 32 データレポートがリンクされていない。 33 未定義トレースレポート 34 トレース ID の重複。 35 データレポートが多すぎる。 36 サンプル期間が範囲外。 37 グループサイズが大きすぎる。 38 回復アクションは現在無効。 39 要求した回復の実行を妨げる別の回復が現在実行中。 40 アクティブな回復アクションなし。 41 例外回復の失敗。 42 例外回復の中断。 43 無効表要素。 44 未定義表要素。 45 前もって設定済みのものは削除できない。 46 無効トークン。 47 無効パラメータ。 48~63 保留。
ERRTEXT	a	*ERRCODE* で示されるエラーを記述する文字列。最大 80 文字。

■例外

RMACK がエラーなしを表している場合のみ、P=0。

15.2.93 S15F35 レシピ削除要求

```
Recipe Delete Request
M,H→E, 返信
```

■説明

このメッセージは 1 つあるいは複数のレシピを削除するあるいは選択を解除するよう要求するのに使用される。マルチブロックなら、この前に S15F1 レシピ管理マルチブロック問い合わせ、S15F2 レシピ管理マルチブロック許可トランザクションがある。

■構造

```
s15f35w
{
   <u4 DATAID>
   <a RESPEC>
   <u1 RCPDEL>
   {
      <a RCPID1>
      .
      .
      <a RCPIDn>
   }
}
```

名称	型	説明
DATAID	u4	データ ID。
RESPEC	a	レシピエグゼキュータのオブジェクト指定子。
RCPDEL	u1	0 削除。 1 選択解除。 >1 保留。
RCPID	a	レシピ識別子。フォーマットされたテキストは *OBJSPEC* の要求事項に準拠している。
OBJSPEC	a	内部フォーマットを持ち、特定のオブジェクトインスタンスを示すのに用いられるテキストストリング。このストリングは、フォーマットされたサブストリングの連続からなり、それぞれがオブジェクトタイプと識別子を特定している。サブストリングのフォーマットは以下の 4 つのフィールドからなる。 オブジェクトタイプ コロン “:” オブジェクト識別子 不等号 “>” コロン “:” は、オブジェクトタイプの最後に用いられる。不等号 “>” は、識別フィールドの最後に用いられる。オブジェクトタイプは他の方法でも決められるので省略しても良い。最後の “>” は任意である。

■例外

リスト、n=0 でレシピが選択解除される場合（*RCPDEL*=1）なら、現在選択されているレシピ全部が表される。

15.2.94 S15F36 レシピ削除確認

```
Recipe Delete Acknowledge
M,H←E
```

■説明

このメッセージはレシピを削除するあるいは選択解除するようにという要求を確認するのに使用される。

■構造

```
s15f36
{
   <u1 RMACK>
   {
      {
         <ERRCODE1>
         <a ERRTEXT1>
      }
      .
      .
      {
         <ERRCODEp>
         <a ERRTEXTp>
      }
   }
}
```

名称	型	説明
RMACK	u1	要求されたアクションが成功して完了したか、拒否されたか、エラーで完了したか、あるいは要求者へ通知して完了するか、ということを伝達。 0 成功して完了。 1 該当アクションを実行できない。 2 エラーで完了。 3 該当アクションは完了し通知は送信される。 4 当該アクションが存在することを要求しない。
ERRCODE	u*	エラー識別コード。 0 エラーなし。 1 オブジェクト指定子中のオブジェクト不明。 2 ターゲットオブジェクトタイプ不明。 3 オブジェクトインスタンス不明。 4 属性名称不明。 5 リードオンリー属性。アクセス拒否。 6 オブジェクトタイプ不明。 7 無効属性地。 8 シンタックスエラー。 9 検証エラー。 10 妥当性エラー。 11 オブジェクト指定子が使用中。 12 パラメータが正しく指定されていない。 13 指定すべきパラメータがすべて指定されていない。 14 要求されたオプションはサポートされていない。 15 使用中。 16 処理の準備ができていない。 17 現行の状態に無効なコマンド。 18 変更された材料なし。 19 材料は部分的に処理された。 20 材料は全て処理された。 21 レシピの指定に関連するエラー。 22 処理中に失敗。 23 処理中でない時に失敗。 24 材料不足による失敗。 25 ジョブの中断。 26 ジョブの停止。 27 ジョブの取り消し。 28 選択されたレシピは変更できない。 29 未定義イベント。 30 レポート ID の重複。 31 未定義データレポート。 32 データレポートがリンクされていない。 33 未定義トレースレポート 34 トレース ID の重複。 35 データレポートが多すぎる。 36 サンプル期間が範囲外。 37 グループサイズが大きすぎる。

		38 回復アクションは現在無効。 39 要求した回復の実行を妨げる別の回復が現在実行中。 40 アクティブな回復アクションなし。 41 例外回復の失敗。 42 例外回復の中断。 43 無効表要素。 44 未定義表要素。 45 前もって設定済みのものは削除できない。 46 無効トークン。 47 無効パラメータ。 48~63 保留。
ERRTEXT	a	*ERRCODE* で示されるエラーを記述する文字列。最大 80 文字。

■例外

RMACK がエラーなしを表している場合のみ、p=0。