

GEM 通信エンジン・DSHEng4Class (GEM+GEM300)

ソフトウェア・パッケージ

## クラス・ライブラリ・デモプログラム V2

### 説 明 書

装置変数、通信メッセージ、WP 通信シナリオ・シミュレーション

Block モードによる 1 次メッセージの送信機能追加版

2011 年 6

株式会社データマップ

**[取り扱い注意]**

- ・ この資料ならびにソフトウェアの一部または全部を無断で使用、複製することはできません。
- ・ 本説明書に記述されている内容は予告なしで変更される可能性があります。
- ・ Windows は米国 Microsoft Corporation の登録商標です。
- ・ ユーザーが本ソフトウェアの使用によって生じた遺失履歴、(株) データマップの予見の有無を問わず発生した特別損害、付随的損害、間接損害およびその他の拡大損害に対して責任を負いません。

**【改訂履歴】**

番号	改訂日付	項 目	概 略
1.	2011. 4	初版	DSHENG4-384-00 の改定したもの DSHEng4Class クラスライブラリ使用のデモプログラム版。
2.	2011. 6	追加されたブロックモードによる 1 次メッセージ送信関連  WP シミュレーションをスロットで実行するようにした。	S6f11, S5F1 送信をそれぞれ、収集イベント、アラーム画面に移した。(元はメイン画面にあった。) 3. 3、3. 4 参照  load, processing, unload 3 つのクラスを設け、その中のスロットで行うようにした。 1 次メッセージの送信はブロックモードで行う。 また、プログラムを簡素化するため PLC シミュレータの使用は止めた。3. 14, 6. 参照
3.			
4.			

## 目 次

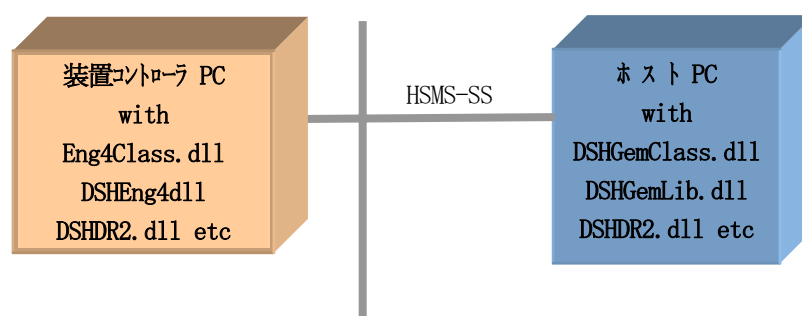
1. 概要 .....	1
1. 1 DSHENG4 通信制御エンジン関連ドキュメント一覧表 .....	2
(1) DSHENG4 ユーザーズ・ガイド、一般関連ドキュメント - ¥DSHENG4¥doc に保存 .....	2
(2) DSHENG4Classクラス・ライブラリ関連ドキュメント ¥DSHENG4¥doc-classに保存 .....	2
(3) DSHENG4 通信エンジン ライブラリ関連ドキュメント ¥DSHENG4¥doc-libに保存 .....	3
(4) HSMS通信ドライバー関連ドキュメント ¥DSHENG4¥docに保存 .....	3
(5) デモプログラム関連ドキュメント ¥DSHENG4¥doc-demoに保存 .....	3
1. 2 デモ関連ファイルの保存場所 .....	4
2. 構成 .....	5
2. 1 装置モデルと構成 .....	5
2. 2 デモプログラムの機能構成 .....	6
2. 2. 1 操作画面構成 .....	6
2. 2. 2 通信関連機能 .....	7
2. 2. 2. 1 1次メッセージの送信 .....	7
2. 2. 2. 2 1次メッセージの受信 .....	7
3. 画面と操作 .....	8
3. 1 メイン画面 .....	8
3. 1. 1 通信制御の開始と停止操作 .....	9
3. 1. 1. 1 開始操作 .....	9
3. 1. 1. 2 情報操作、制御操作画面 .....	11
3. 1. 1. 3 停止操作 .....	12
3. 1. 2 画面内の各種実行ボタンの操作 .....	13
3. 1. 2. 1 バックアップ情報の確認 .....	13
3. 1. 2. 2 GC (Gabage Collection) ボタン操作 .....	13
3. 2 変数関連情報操作画面 .....	14
3. 2. 1 EC 装置定数操作画面 .....	14
3. 2. 2 SV 装置状態変数操作画面 .....	15
3. 2. 3 DVVAL 装置データ変数操作画面 .....	16
3. 2. 4 変数 装置変数 EC, SV, DVVAL操作画面 .....	17
3. 2. 5 SVトレース操作画面 .....	18
3. 2. 6 変数リミット操作画面 .....	19
3. 3 CE収集イベントとRPレポート操作画面 .....	20
3. 3. 1 CE収集イベント情報操作画面 .....	20
3. 3. 2 RPレポート情報操作画面 .....	21
3. 4 アラーム操作画面 .....	22
3. 5 PP プロセス・プログラムとRCP レシピ情報操作画面 .....	23
3. 5. 1 PP プロセス・プログラム操作画面 .....	23
3. 5. 2 RCP レシピ操作画面 .....	24
3. 6 PRJ プロセス・ジョブ操作画面 .....	25
3. 7 CJ コントロール・ジョブ操作画面 .....	26
3. 8 Carrier キャリア操作画面 .....	27
3. 9 Substrate 基板操作画面 .....	28
3. 10 端末メッセージ送信画面 .....	29
3. 11 WPウェハー・プロセスオペレーション操作画面 .....	31
3. 12 スプール設定操作画面 .....	32
3. 13 1次メッセージの送信操作画面 .....	33

3. 14	1 次メッセージに対する応答ACK設定画面	34
3. 15	クラスのトレースとデバッグ機能設定画面	35
3. 15. 1	トレース有効/無効の設定と実績表示機能	36
3. 15. 2	トレースログのファイル保存と表示	36
3. 15. 3	DSHENG4 エンジン、DSHDR2 通信ドライバーのメモリ・リークチェック操作	37
3. 15. 4	その他の機能	37
4.	装置情報と通信メッセージ	38
4. 1	メッセージ	38
4. 2	変数一覧	39
4. 2. 1	EC - 装置定数	39
4. 2. 2	SV - 装置状態変数	40
4. 2. 3	DVVAL- 装置データ変数	42
4. 3	CEイベントID	43
4. 4	レポートID	44
4. 5	アラームID	46
4. 6	予約CEIDと変数	47
5.	ジョブ定義ファイル	48
6.	実現するシナリオ	50
6. 1	通信確立シナリオ	50
6. 2	ウェハ処理シナリオ (正常シナリオ)	51
6. 2. 1	ジョブ定義ファイル	51
6. 2. 2	シナリオ詳細	54
6. 2. 2. 1	Load処理 - WpLoadクラス	55
6. 2. 2. 2	ウェハ処理 - WpProcessクラス	56
6. 2. 2. 3	Unload処理 - WpUnloadクラス	58
7.	各種定義ファイルについて	59
7. 1	DSHDR2 通信環境定義ファイル	59
7. 2	装置起動ファイル	60
7. 3	装置管理情報定義ファイル	61
7. 4	ジョブファイルサンプル (JobSche. txt)	62
8.	バックアップファイルとログファイル	63

## 1. 概要

本説明書は株式会社データマップが開発・販売する半導体製造工場向けシステムのための GEM300 仕様をサポートする DSHENG4 通信エンジンならびに DSHEng4Class ライブラリ・ソフトウェア・パッケージ評価のために作成されたデモ・プログラムの基本的な仕様と操作画面などについて説明します。本デモプログラムは装置の機能を実行します。

基本仕様は、次のシステム構成を想定し、行うことを前提にしています。



ホスト側 PC には、DSHGemLib, DSHGemClass 通信エンジンのデモプログラムを使用します。

OS は、Windows、プログラム言語は、.Net テクノロジによる C#2008、.Net Vb2008 言語を使用します。  
デモプログラムに対する操作は、.Net 言語を使って、GUI インタフェース画面を使って行います。

GEM レベルでの通信制御ならびに装置情報の管理には、DSHEng4Class クラス・ライブラリに提供されるクラス群を利用するプログラミングになります。

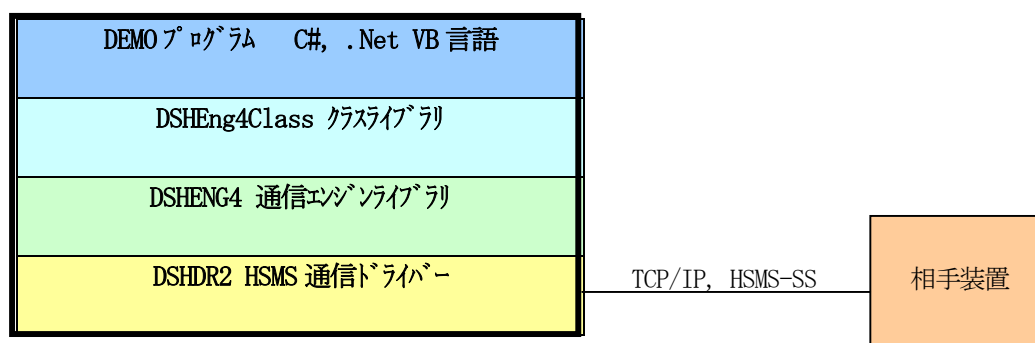
これらによって、装置情報の設定、参照操作、具体的な通信制御機能の実行操作、さらには、装置、ホスト間の連続メッセージ通信機能を確認できます、簡単なウェハー処理 (WP) モデルによるシナリオ処理のデモも含んでいます。

本デモプログラムは、ユーザが DSHEng4Class クラス・ライブラリをアプリケーションに組み込む上でのプログラミングの具体的な 1 つの方法を提供します。ソースファイルがオープンされていますので参照ください。

DSHEng4Class ライブラリによるアプリケーションの開発手順については、下記資料を参照ください。

**DSHENG4CLASS クラス・ライブラリ 文書番号 DSHENG4-09-30305-00 「プログラミングの手引き」**

デモプログラムを構成する要素は、以下のような階層構造になります。



## 1.1 DSHENG4 通信制御エンジン関連ドキュメント一覧表

関連ドキュメントとして以下のものがあります。

### (1) DSHENG4 ユーザーズ・ガイド、一般関連ドキュメント - ¥DSHENG4¥doc に保存

#	文書番号	文書名	注釈
1	DSHENG4-09-30300-00	DSHENG4 通信制御エンジンライブラリ (SECS/HSMS) ユーザーズ・ガイド	DSHENG4 の全般的な機能の説明書です。
2	DSHENG4-09-30301-00	DSHENG4 起動ファイル定義仕様書	装置別の起動情報の定義方法の説明書です。
3	DSHENG4-09-30302-00	DSHENG4 装置管理情報定義仕様書 (変数、収集イベント、アラームその他)	DSHENG3 と同じ内容です。定義ファイルはテキストファイルです。
4	DSHENG4-09-30303-00	装置管理情報定義ファイルコンパイル説明書	DSHENG3 と共通です。
5	DSHENG4-09-30304-00	DSHENG4 への手引き	DSHENG4 導入時に参考にする作業手順書です。
6	DSHENG4-09-30305-00	インストールと保存ファイル	(この説明書です。)
7	DSHENG4-09-30308-00	DSHENG4, DSHENG3 起動ファイル、装置管理情報ファイル設定・編集プログラム説明書	DSHENG3, DSHeng4 共通
8	DSHENG4-09-30310-00	変数リミット監視機能 説明書	リミット監視の考え方、処理方法の説明書です。
9	DSHENG4-09-30340-00	ユーザー作成ライブラリ関数 2 次メッセージ応答関数一覧表	C, C++言語によるプログラミング .Net 用クラスライブラリを使用しない
10	DSHENG4-09-30351-00	バックアップファイル参照プログラム説明書	DOS コマンドで List 構造で表示します・
11	DSHENG4-09-30340-00	ユーザー作成ライブラリ関数 2 次メッセージ応答関数一覧表	C, C++言語によるプログラミング .Net 用クラスライブラリを使用しない
12	DSHENG4-09-30351-00	バックアップファイル参照プログラム説明書	DOS コマンドで List 構造で表示します・

### (2) DSHeng4Class クラス・ライブラリ関連ドキュメント ¥DSHENG4¥doc-class に保存

#	文書番号	文書名	注釈
1	DSHENG4-09-30361-00	ClassLib-Info-1 Vol-1 エンジン起動と管理情報クラス 編 Part-1	エンジン、装置起動管理情報のアクセス
2	DSHENG4-09-30362-00	ClassLib-Info-2 Vol-1 エンジン起動と管理情報クラス 編 Part-2	管理情報のアクセス
3	DSHENG4-09-30363-00	ClassLib-Comm Vol-2 メッセージ通信クラス 編	GEM メッセージ送信
4	DSHENG4-09-30305-00	クラスライブラリ プログラミングの手引き	準備するファイルと開発ステップ 手順も含む
5	DSHENG4-09-30306-00	クラス生成・消滅トレースと表示機能について	クラス・デバッグ 用

### ( 3 ) DSHeng4 通信エンジン ライブラリ関連ドキュメント ¥DSHENG4¥doc-lib に保存

#	文書番号	タイトル名と内容
1	DSHENG4-09-30321-00	1. 概要 2. DSHENG4 が提供するサービスと 1 次メッセージの送受信処理 3. 1 DSHENG4 初期設定関連関数 3. 2 通信制御関連関数
2	DSHENG4-09-30322-00	3. 3 変数 (EC, SV, DVVAL) 情報アクセスと通信サービス
3	DSHENG4-09-30323-00	3. 4 Limit 変数リミット情報関連関数 3. 5 TR トレース情報アクセスサービス関数
4	DSHENG4-09-30324-00	3. 6 CE 収集イベント情報アクセスと通知関数 3. 7 Report レポート情報アクセス関数 3. 8 Alarm アラーム情報アクセスと通知関数
5	DSHENG4-09-30325-00	3. 9 Spool スプール関連関数 3. 10 端末サービス情報関連関数
6	DSHENG4-09-30326-00	3. 11 PP プロセスプログラム情報アクセスサービス関数 3. 12 FPP 書式付プロセスプログラム情報アクセスサービス関数
7	DSHENG4-09-30327-00	3. 13 RCP レシビ情報アクセスサービス関数
8	DSHENG4-09-30328-00	3. 14 CAR キャリア情報アクセスサービス関数
9	DSHENG4-09-30329-00	3. 15 SUBST 基板情報アクセスサービス関数
10	DSHENG4-09-3032A-00	3. 16 キャリアアクションメッセージ(S3F17) 関連関数 3. 17 ポートアクション、アクセスモード(S3F23, S3F25, S3F27) 関連関数
11	DSHENG4-09-3032B-00	3. 18 ホストリモートコマンド(S2F41) 関連関数 3. 19 拡張リモートコマンド(S2F49) 関連関数
12	DSHENG4-09-3032C-00	3. 20 PRJ プロセスジョブ情報アクセス、送信サービス関数
13	DSHENG4-09-3032D-00	3. 21 CJ コントロールジョブ情報アクセスサービス関数
14	DSHENG4-09-3032E-00	3. 22 レチクル制御( S14F19, S14F21) サービス関数 3. 23 レチクル搬送ジョブ要求( S3F35) サービス関数
15	DSHENG4-09-3032F-00	3. 24 オブジェクト関連メッセージの応答情報とエラー情報関連設定 ライブラリ関数 3. 25 その他のライブラリ関数

### ( 4 ) HSMS 通信ドライバー関連ドキュメント ¥DSHENG4¥doc に保存

#	文書番号	文書名	注釈
1	DSHDR2-06-20000-02	DSHDR2 SECS/HSMS レベル2 通信制御ドライバー ユーザズマニュアル	SECS/HSMS 通信制御ドライバーの 説明書です。
2	DSHDR2-06-20040-0	DSHDR2 レベル2 通信ドライバー通信ログモニター説明書	リアルタイムで通信トランザクションをモニター 画面で見ることができます。

### ( 5 ) デモプログラム関連ドキュメント ¥DSHENG4¥doc-demo に保存

#	文書番号	文書名	注釈
1	DSHENG4-09-30501-00	クラス・ライブラリ・デモプログラム説明書	(本ドキュメントです。)
2	DSHENG4-09-30502-00	DSHeng4Class クラス・ライブラリ版 デモプログラム インストールと保存ファイル	C#, .Net VB デモプログラムです。

## 1.2 デモ関連ファイルの保存場所

デモプログラムとして提供される setup.exe を実行すると、以下のディレクトリに必要なファイルが保存されます。  
c : ドライブの ¥dsheng4 の下に保存されます。

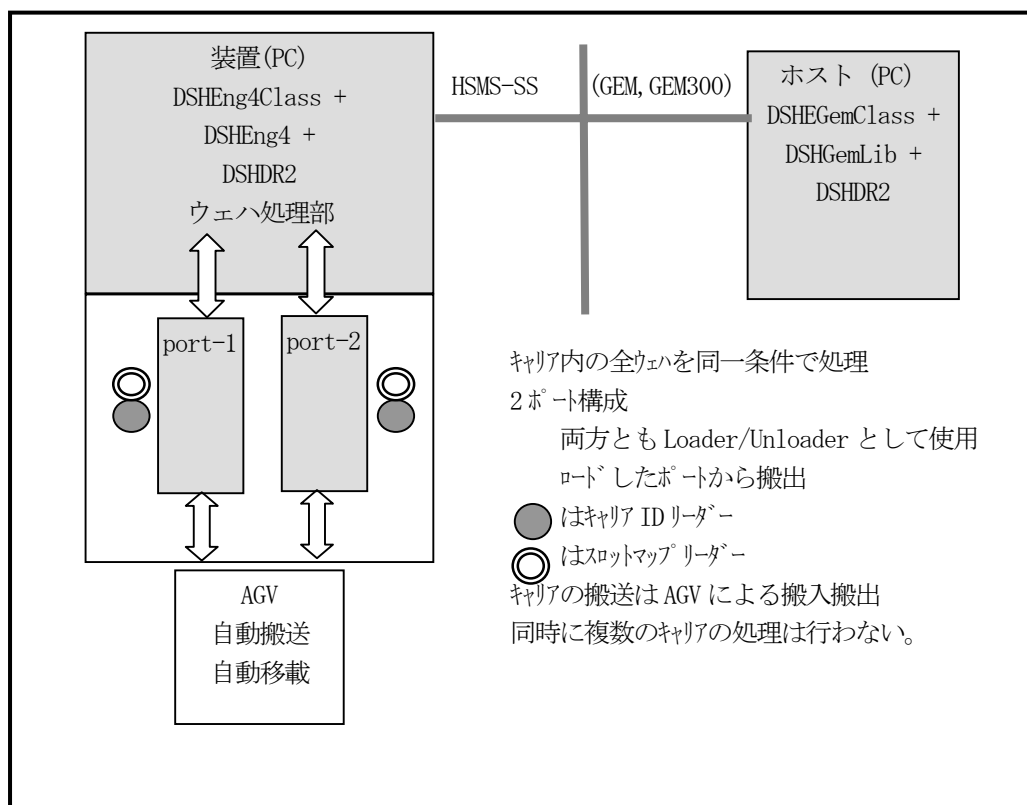
home dir	path	ファイル名	種類、機能など
¥dsheng4¥	bin	eng4csdemoV2.exe eng4vbdemoV2.exe dsheng4.dll, eng4class.dll など	実行プログラム・ファイル C#の実行プログラム VBの実行プログラム エンジン・プログラム クラスライブラリ・プログラム
	cnf	equip.cnf comm.def eqinfo.fil	環境定義ファイル 装置起動ファイル DSHDR2 HSMS 通信定義ファイル 装置管理情報定義ファイル
	tool	dshgemset.exe dshcompile.exe seedback.exe	ツール・プログラム 装置管理情報編集プログラム 同コンパイル・プログラム 装置情報バックアップ参照プログラム
	logmon	logmon.exe	HSMS 通信ログ・モニター(リモート) リアルタイム・HSMS 通信ログモニター・プログラム
	eng4csdemoV2¥		
		engcsdemo.sln	C#デモ・プログラム ソリューション・ファイル
	eng4csdemoV2	*.cs	C#デモ・プログラム ソースファイル
	eng4vbdemoV2¥		
		engvbdemo.sln	VBデモ・プログラム ソリューション・ファイル
	eng4vbdemoV2	*.vb	VBデモ・プログラム ソースファイル
	doc		ユーザーズ・ガイド 装置起動ファイル、装置管理情報定義ファイルの説明書など
	doc-class		Eng4Class クラス・ライブラリ関連説明書
	doc-lib		DSHEng4 通信エンジン・ライブラリ関連説明書
	doc-demo		デモプログラム関連説明書



## 2. 構成

### 2.1 装置モデルと構成

本デモプログラムの制御対象装置として次のようなモデルを想定します。



本デモ・プログラムは、DSEng4Class ならびに DSEng4 通信エンジンを使って、GEM に基づく SECS-II メッセージの通信制御と装置情報管理機能を行うことを目的にしています。

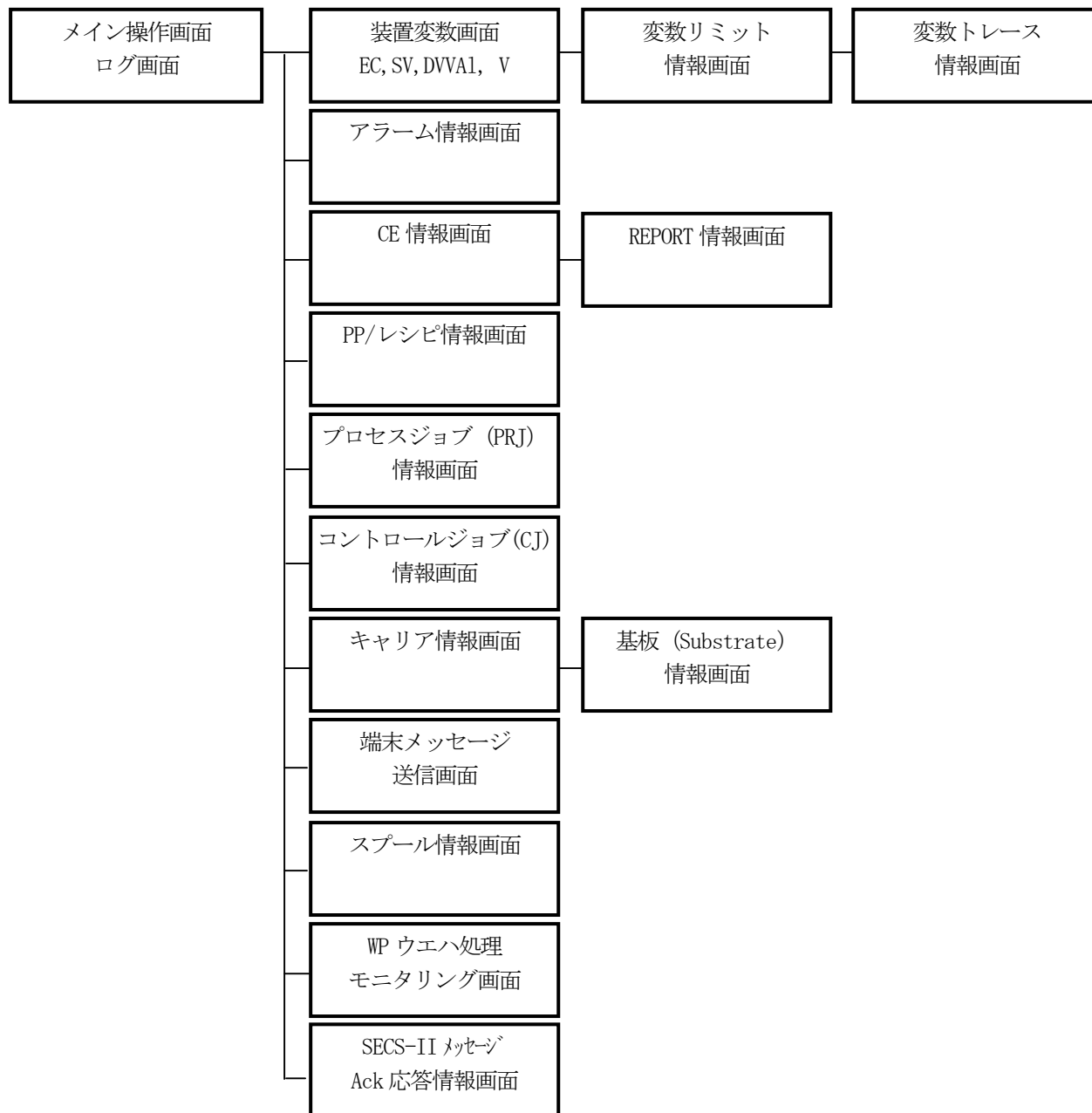
デモプログラムでは、簡単なシナリオモデルにより、WP (Wafer Processing) の連続処理を行う機能も含んでいます。これを WP シミュレーションと呼ぶことにします。

シナリオの中では、装置のポートでのキャリアの Load/Unload は、特にオペレータの操作をしないで自動的に進めることにします。

## 2.2 デモプログラムの機能構成

### 2.2.1 操作画面構成

DSHENG4 エンジンならびに DSHEng4Class クラス・ライブラリのほとんどの機能の確認画面が準備されています。



## 2.2.2 通信関連機能

1 次メッセージの送信処理と 1 次メッセージの受信処理を行います。

### 2.2.2.1 1 次メッセージの送信

1 次メッセージの送信は、DSHEng4Class の各メッセージ送信のために設けられたクラスを使って送信します。

例えば、S6F11 メッセージの送信クラスは、DshS6F11Send になります。

詳しい、説明は以下を参照ください。

文書番号 DSHENG4-09-30363-00-ClassLib-Comm 「クラスライブラリ Vol-2 通信編」  
文書番号 DSHENG4-09-30305-00 「クラスライブラリ プログラミングの手引き」  
5. 1 次メッセージの送信

送信は、ブロックモードか非ブロックモードでおこないます。

ブロックモードでは送信クラスの `send_wait()` メソッドを使用します。

非ブロックモードでは `send()` メソッドに callback 関数のポインタと結果情報を受け取る情報格納領域を引数として渡し、応答メッセージの受信によって、callback 関数を呼び出して貰うことによって送信完了を確認することになります。

### 2.2.2.2 1 次メッセージの受信

相手からの 1 次メッセージの受信処理は、ユーザが受信処理をしたいとエンジンに登録した 1 次メッセージの受信処理になります。

受信したい 1 次メッセージの登録は、DshEngine クラスを使って行います。登録方法については、以下を参照してください。

文書番号 DSHENG4-09-30305-00 「クラスライブラリ プログラミングの手引き」  
3. 4 受信 1 次メッセージの登録

1 次メッセージの受信は、ユーザが DshEngine クラスに受信 Polling を依頼し、依頼時にメッセージ受信イベント・ハンドラーを渡しておきます。DshEngine クラスが 1 次メッセージを受信すると、受信情報をパラメータにして、非同期にイベント・ハンドラーが呼び出してくれます。

また、処理した後、受信したメッセージに対する 2 次メッセージの応答送信を行います。

これらについては、以下の説明書を参照ください。

文書番号 DSHENG4-09-30305-00 「クラスライブラリ プログラミングの手引き V2」  
3. 5 1 次メッセージ受信ポーリングの開始  
4. 受信メッセージの処理

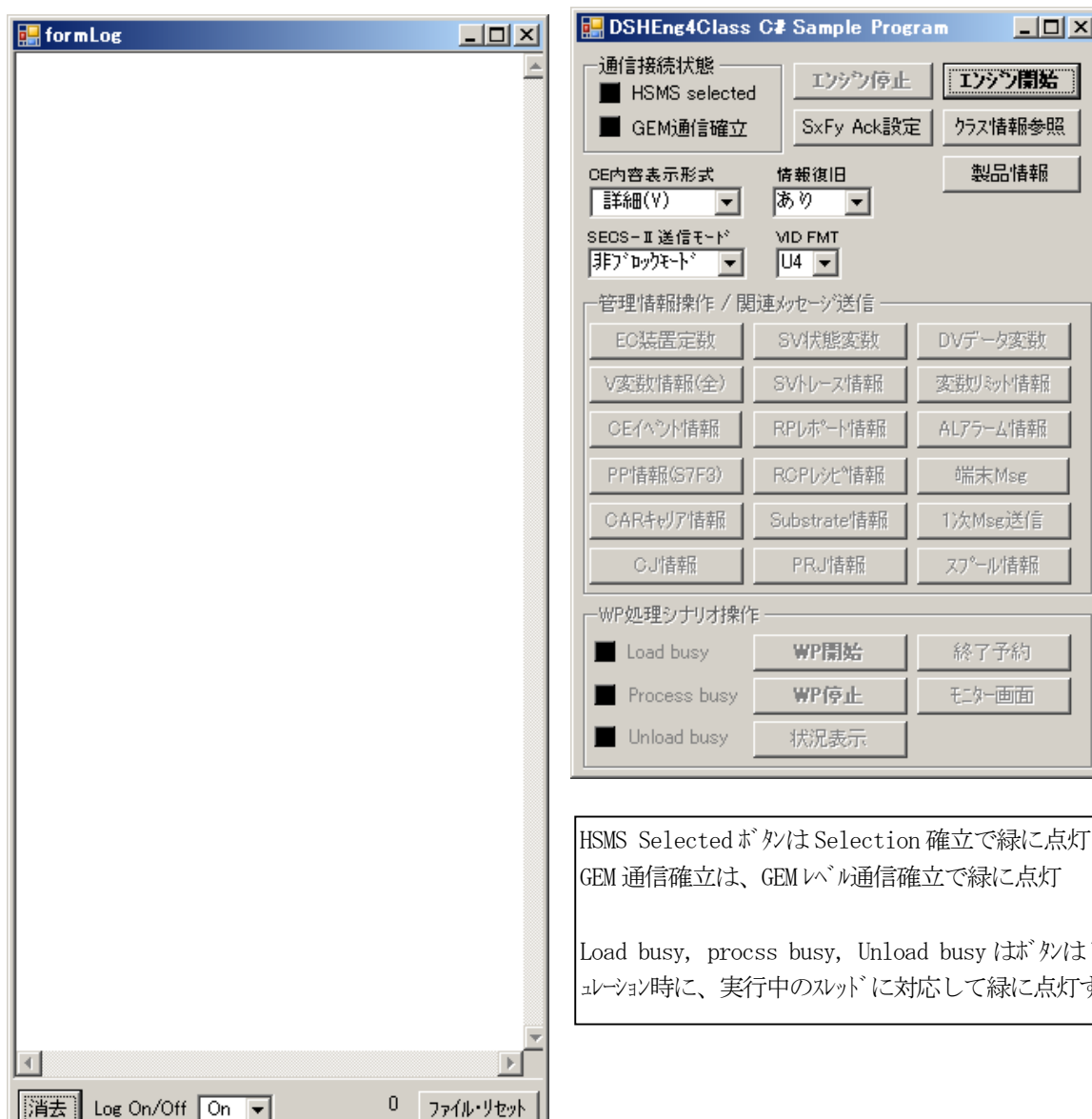
### 3. 画面と操作

#### 3.1 メイン画面

GemCsDemo.exe または GemVbDemo.exe デモプログラムの起動によって、まず、デモプログラムの紹介画面が表示され、そのあと、進むボタンのクリックでメイン画面が表示されます。

2. 2. 1の画面構成図のトップレベルの画面です。下に示します。

左側には、処理のログ記録を表示する画面が位置します。(画面下側の Log On/Off の選択で表示出力しないようにできます。)



formLog

DSHeng4Class C# Sample Program

通信接続状態

☒ HSMS selected

☒ GEM通信確立

エンジン停止

エンジン開始

SxFy Ack設定

クラス情報参照

製品情報

情報復旧

MD FMT

SECS-II 送信モード

管理情報操作 / 関連メッセージ送信

EO装置定数

SV状態変数

DVデータ変数

V変数情報(全)

SVトレース情報

変数リミット情報

CEイベント情報

RPLレポート情報

ALアラーム情報

PP情報(S7F0)

ROPLシフト情報

端末Msg

CARキャリア情報

Substrate情報

1次Msg送信

CJ情報

PRJ情報

スプール情報

WP処理シナリオ操作

☒ Load busy

☒ Process busy

☒ Unload busy

WP開始

WP停止

終了予約

モニター画面

状況表示

消去

Log On/Off

On

0

ファイルリセット

HSMS Selected ボタンは Selection 確立で緑に点灯  
GEM 通信確立は、GEM レベル通信確立で緑に点灯

Load busy, process busy, Unload busy はボタンは WP シミュレーション時に、実行中のスレッドに対応して緑に点灯する。

### 3.1.1 通信制御の開始と停止操作

#### 3.1.1.1 開始操作

デモ・プログラムが起動から、エンジン、装置開始まで流れは次のようになります。

操作(ボタン)	プログラムの処理	画面の状態と操作
デモプログラム起動	イントロ画面が表示される	<b>エンジン開始</b> ボタンが有効になり、装置変数 ID などのフォーマットを U2 にしたい場合は、コンボボックスで選択しておく。
<b>エンジン開始</b> クリック	<p>装置変数 ID(数値 ID) が U2 に選択されていれば、エンジンに設定する。</p> <p>DshEngine クラスを使って装置制御を開始します。装置 config ファイル名をセットし、start メソッドを実行する。</p> <p>装置開始が正常にできたら、</p> <ul style="list-style-type: none"> <li>・予約変数、イベント ID の設定をする。</li> <li>・ユーザが処理する受信 1 次メッセージを登録する。</li> <li>・装置側の場合、S6F11, S5F1 の ID list をコンボボックスに設定する。</li> <li>・DshEngine クラスに SECS-II 受信メッセージのポーリングを開始させる。</li> <li>・timer-1 を開始し、HSMS-SS 通信確立と GEM レベルでの通信確立を監視する。</li> </ul> <p>その後、自動的に Enable(通信確立)の通信(S1F13 のやり取り)を開始します。</p>	<p>開始が正常終了ならば</p> <p><b>エンジン開始</b> ボタン = 無効 <b>エンジン停止</b> ボタン = 有効</p> <p>正常に開始されると、comm. def に指定された PORT を通して相手装置と HSMS-SS プロトコル確立のための制御も開始されます。</p> <p>HSMS-SS のセクション確立したら、HSMS-SS 確立を示す画面がポップアップされます。</p>

次ページに、エンジン、装置起動後のメイン操作画面表示します。

使用しないボタンは無効になっています。

なお、コンボボックスは以下の意味をもっています。

<b>CE 内容表示形式</b> の選択	: 送信した S6F11 のログ表示の選択です。
<b>SECS-II 送信モード</b> の選択	: ブロックモードか非ブロックモードかの選択です。
<b>情報復旧の選択</b>	: 前回終了時にバックアップした管理情報を復旧するかどうかの選択です。
<b>VID FMT</b>	: VID, CEID, RPID のフォーマットを U2 にしたいときに切り替えます。

DSHEng4Class C# Sample Program

通信接続状態  
☒ HSMS selected  
☒ GEM通信確立

エンジン停止    エンジン開始

SxFy Ack設定    クラス情報参照

製品情報

GE内容表示形式    情報復旧  
 詳細(V)    あり

SECS-II 送信モード    MD FMT  
 オフ・バックモード    U4

管理情報操作 / 関連メッセージ送信

EC装置定数	SV状態変数	DVデータ変数
V変数情報(全)	SVトレース情報	変数リミット情報
CEイベント情報	RPLレポート情報	ALアラーム情報
PP情報(S7F3)	RCレポート情報	端末Msg
CARキャリア情報	Substrate情報	1次Msg送信
CJ情報	PRJ情報	スプール情報

WP処理シナリオ操作

<input checked="" type="checkbox"/> Load busy	WP開始	終了予約
<input checked="" type="checkbox"/> Process busy	WP停止	モニター画面
<input checked="" type="checkbox"/> Unload busy	状況表示	

### 3. 1. 1. 2 情報操作、制御操作画面

操作ボタンと各情報操作画面は次のとおりです。

ボタ	画面	ボタ	画面
変数情報(全)	装置変数画面 V (EC, SV, DVVAL)	S10F1	端末メッセージ 送信画面
EC 装置定数	装置定数画面 EC	WP モニター	WP ウエハ処理 モニター画面
SV 状態変数	装置状態変数画面 SV	スプール	スプール情報画面
DVVAL データ変数	データ変数画面 DVVAL	SxFx Ack 設定	SECS-II メッセージ Ack 応答情報画面
SV トレース情報	変数トレース 情報画面		
変数リミット情報	変数リミット 情報画面		
CE イベント情報	CE 情報画面		
レポート情報	REPORT 情報画面		
アラーム情報	アラーム情報画面		
PP 情報	PP (Process Program) 情報画面		
レシビ 情報	RCP (レシビ) 情報画面		
PRJ 情報	プロセスジョブ (PRJ) 情報画面		
CJ 情報	コントロールジョブ (CJ) 情報画面		
キャリア情報	キャリア情報画面		
基板情報	基板 (Substrate) 情報画面		

### 3. 1. 1. 3 停止操作

制御の停止は、**エンジン停止** ボタンをクリックします。

全て停止したら、**エンジン開始** ボタンが有効になり、**エンジン停止** ボタンは、無効になります。



### 3.1.2 画面内の各種実行ボタンの操作

前述、3.1.1.2に挙げたボタン以外のもので、クリックで直接、機能実行するためのボタン操作について記述します。基本的には、エンジン開始した後に実行できる操作です。

#### 3.1.2.1 バックアップ情報の確認

以前に保存した装置管理情報のバックアップファイルの内容が正しい形式で保存されているかどうかの確認を行うための操作です。

エンジン開始前に、この確認をすることができます。

情報全体を一度に確認する方法と、情報を個別に確認する方法があります。この選択はコンボボックスによって指定します。



情報を選択し、**BKUP 確認** ボタンをクリックすることで、その結果がログ表示されます。

DshEngine クラスのメソッドを使用します。

確認結果は次のように表示されます。

	選択	結果	表示
1	全情報	OK	Backup File Check Result = 0
		NG	Backup File Check Result = (-1) <b>error backup file = xxxxx</b>
2	個別	ファイルなし	check_backup_xxxx : result = 0 No backup file
		OK	check_backup_XX : result = 0 Valid FileName : xxxxx.bkp Time Stamp : 2011040119274117 # of Records : 81 Generation : 0
		NG	check_backup_XX : result = (-1)

#### 3.1.2.2 GC (Gabage Collection) ボタン操作

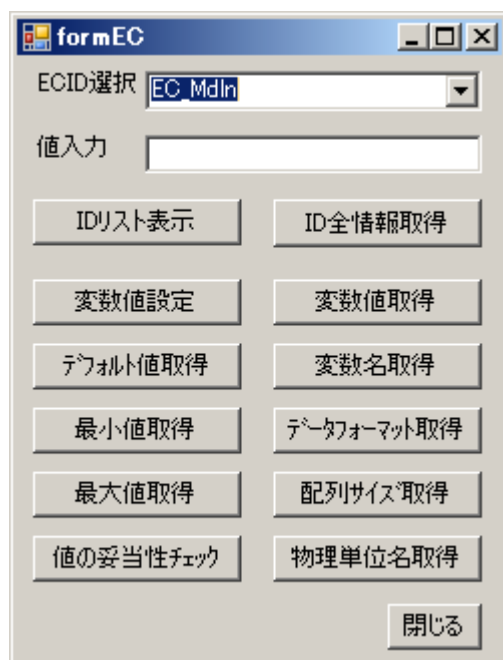
GC ボタンのクリックで、.Net 内のGCを実行します。

## 3.2 変数関連情報操作画面

装置変数には、3種類の変数があります。装置定数(EC)、装置状態変数(SV)、データ変数(DVVAL)です。これら3つの操作画面が独立してあります。また、3種類の変数を一括りにした変数画面があります。

### 3.2.1 EC 装置定数操作画面

装置定数 EC 操作画面は次のようになります。  
それぞれのボタンの機能は、ボタンの表示が示すとおりです。  
操作結果は、LOG 画面に表示されます。

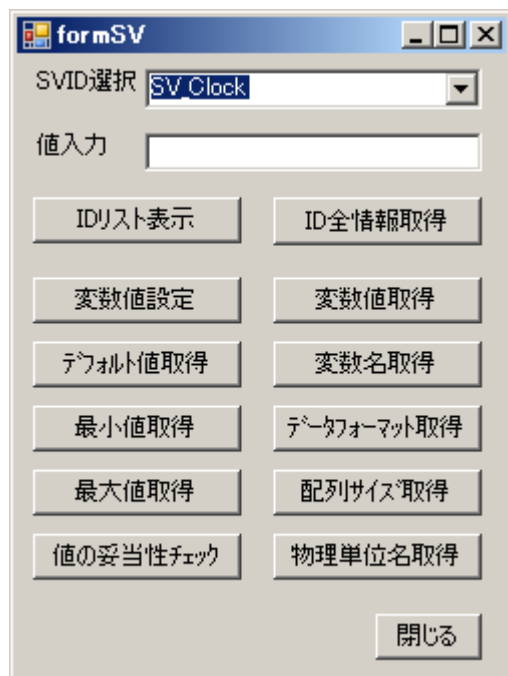


以下のクラスを使用します。

ボタン	クラス	メソッド
情報アクセス関連	DshEC	get_id_list(), get(), set() 他

### 3.2.2 SV 装置状態変数操作画面

装置状態変数 SV 操作画面は次のようになります。  
それぞれのボタンの機能は、ボタンの表示名のとおりです。  
操作結果は、LOG 画面に表示されます。

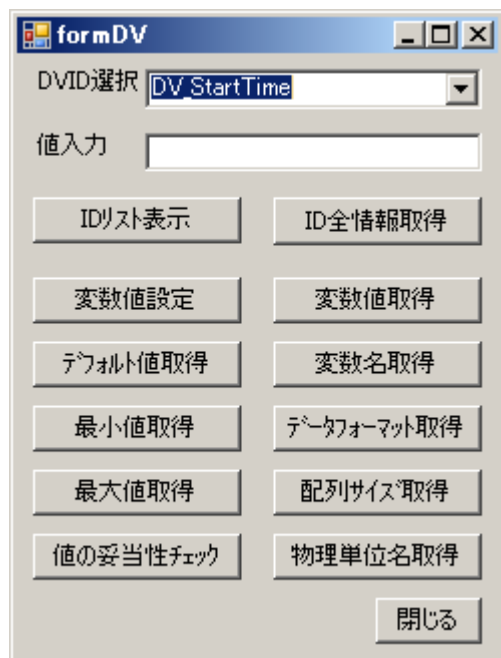


以下のクラスを使用します。

ボタン	クラス	メソッド
情報アクセス関連	DshSV	get_id_list(), get(), set()他

### 3.2.3 DVVAL 装置データ変数操作画面

装置データ変数 DVVAL 操作画面は次のようになります。  
それぞれのボタンの機能は、ボタンの表示名のとおりです。  
操作結果は、LOG 画面に表示されます。

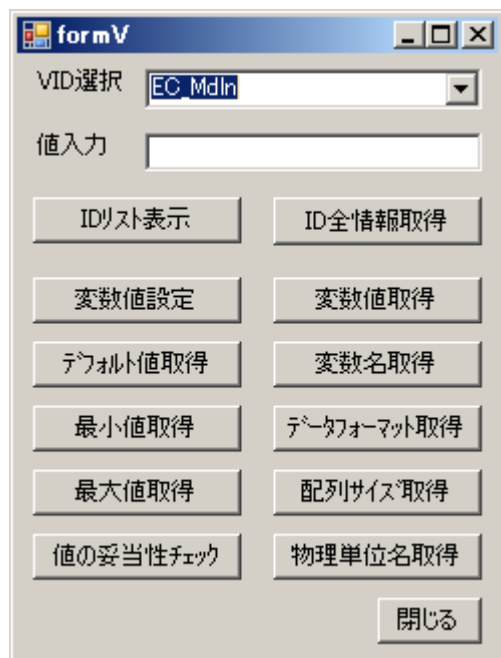


以下のクラスを使用します。

ボタン	クラス	メソッド
情報アクセス関連	DshDV	get_id_list(), get(), set() 他

### 3.2.4 変数 装置変数 EC, SV, DVVAL 操作画面

変数( EC, SV, DVVAL )全体の操作画面は次のようになります。  
それぞれのボタンの機能は、ボタンの表示名のとおりです。  
操作結果は、LOG 画面に表示されます。



以下のクラスを使用します。

ボタン	クラス	メソッド
情報アクセス関連	DshV	get_id_list(), get(), set()他

### 3.2.5 SVトレース操作画面

ホストなしでローカルにSVトレース機能をテストできます。

装置状態変数SV操作画面は次のようになります。

それぞれのボタンの機能は、ボタンの表示名のとおりです。

操作結果は、LOG画面に表示されます。

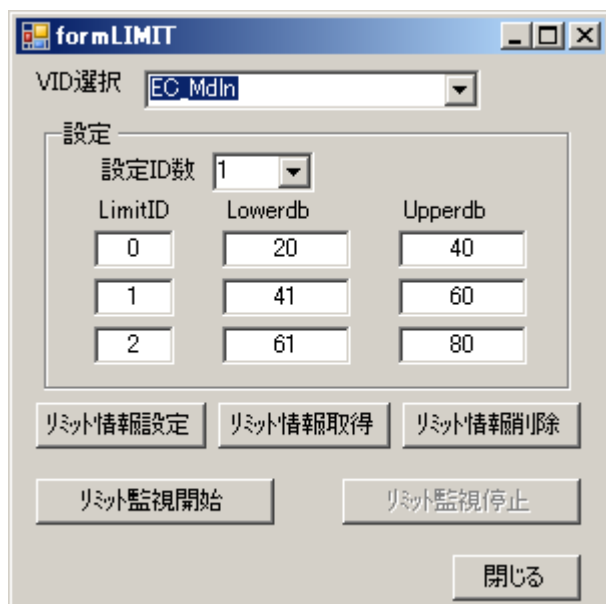
以下のクラスを使用します。

ボタン	クラス	メソッド
情報アクセス関連	DshSV	get_id_list(), get(), set() 他
	DshTrace	get(), set()

### 3.2.6 変数リミット操作画面

ホストなしでローカルに変数リミット監視機能をテストできます。  
変数リミット操作画面は次のようになります。

それぞれのボタンの機能は、ボタンの表示名のとおりです。  
操作結果は、LOG 画面に表示されます。



以下のクラスを使用します。

ボタン	クラス	メソッド
情報アクセス関連	DshV	get_id_list()
	DshLimit	set_id(), clear(), set(), get(), add_limitid, delete()
	DshLimitList	add_limit(),

### 3.3 CE 収集イベントと RP レポート操作画面

CE 収集イベント情報と RP レポート情報の操作のための画面です。

#### 3.3.1 CE 収集イベント情報操作画面

それぞれのボタンの機能は、ボタンの表示名のとおりです。  
操作結果は、LOG 画面に表示されます。



以下のクラスを使用します。

ボタン	クラス	メソッド
情報アクセス関連	DshCE	get_id_list(), set_ceid(), set_enabled(), get_content()他
S6F11 イベント通知	DshS6F11Send	send() - 非ブロックモード send_wait() - ブロックモード



### 3.3.2 RP レポート情報操作画面

それぞれのボタンの機能は、ボタンの表示名のとおりです。装置側では、S2F33, S6F19 の送信は行いません。操作結果は、LOG 画面に表示されます。

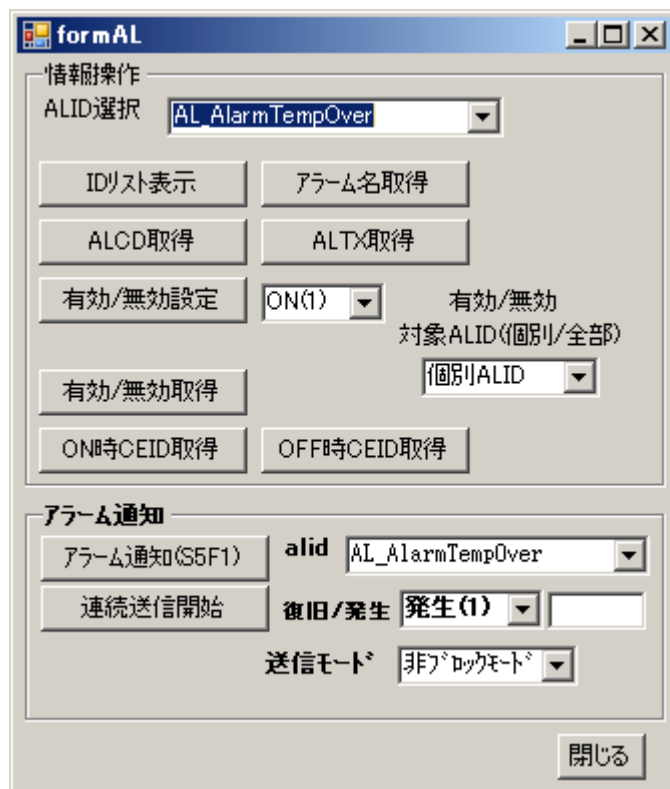


以下のクラスを使用します。

ボタン	クラス	メソッド
情報アクセス関連	DshReport	get_id_list(), set_rpid(), get_content() 他

### 3.4 アラーム操作画面

AL アラーム通知情報の操作画面です。  
それぞれのボタンの機能は、ボタンの表示名のとおりです。  
操作結果は、LOG 画面に表示されます。



The screenshot shows a window titled 'formAL' with two main sections: '情報操作' (Information Operation) and 'アラーム通知' (Alarm Notification).  
In the '情報操作' section, there is a dropdown menu for 'ALID選択' (ALID Selection) currently showing 'AL\_AlarmTempOver'. Below it are buttons for 'IDリスト表示' (Display ID List), 'アラーム名取得' (Get Alarm Name), 'ALCD取得' (Get ALCD), and 'ALTx取得' (Get ALTx). Further down are buttons for '有効/無効設定' (Set Valid/Invalid) and '有効/無効取得' (Get Valid/Invalid). The '有効/無効設定' button is accompanied by a dropdown set to 'ON(1)' and a label '有効/無効 対象ALID(個別/全部)' (Valid/Invalid Target ALID (Individual/All)). The '有効/無効取得' button is accompanied by a dropdown set to '個別ALID' (Individual ALID). At the bottom of this section are buttons for 'ON時CEID取得' (Get CEID at ON) and 'OFF時CEID取得' (Get CEID at OFF).  
The 'アラーム通知' section contains a button 'アラーム通知(S5F1)' (Alarm Notification (S5F1)), a dropdown for 'alid' (currently 'AL\_AlarmTempOver'), a button '連続送信開始' (Start Continuous Transmission), a dropdown for '復旧/発生' (Recovery/Occurrence) set to '発生(1)' (Occurrence (1)), and a dropdown for '送信モード' (Transmission Mode) set to '非ブロックモード' (Non-block mode). A '閉じる' (Close) button is located at the bottom right of the window.

以下のクラスを使用します。

ボタン	クラス	メソッド*
情報アクセス関連	DshAlarm	get_id_list(), set_alid(), set_enabled()他
アラーム通知 S5F1	DshS5F1Send	send() - 非ブロックモード send_wait() - ブロックモード

### 3.5 PP プロセス・プログラムとRCP レシビ情報操作画面

DSHENG4 エンジンでは、プロセス・プログラムとレシビの双方の機能サポートを行います。  
それぞれの操作画面について記述します。

#### 3.5.1 PP プロセス・プログラム操作画面

PP プロセス・プログラム情報の操作画面です。新規 ID 登録、指定された ID に対する情報設定/取得アクセスができます。また、関連するメッセージの送信もあります。

それぞれのボタンの機能は、ボタンの表示名のとおりです。操作結果は、LOG 画面に表示されます。

以下のクラスを使用します。

#### (1) 情報アクセス関連

ボタン	クラス	メソッド
情報アクセス関連	DshPP	get_id_list(), alloc_id(), set_ppbody(), set(), get(), delete()他

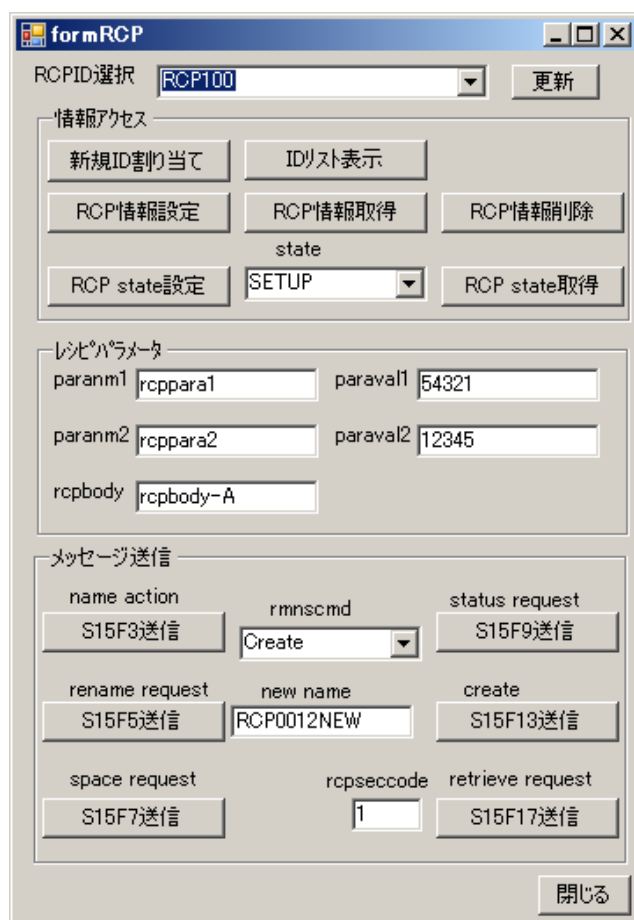
#### (2) メッセージ送信関連

ボタン	情報クラス	MSG 送信クラス	Msg 受信クラス	送信完了 callback クラス
S7F1 送信	DshS7F1Send	send(), send_wait()	-	DshCallback.callback_s7f1
S7F3 送信	DshS7F3Send	send(), send_wait()	-	DshCallback.callback_s7f3
S7F5 送信	DshS7F5Send	send(), send_wait()	DshPP	DshCallback.callback_s7f5
S7F19 送信	DshS7F19Send	send(), send_wait()	-	DshCallback.callback_s7f19
S7F27 送信	DshS7F27Send	send(), send_wait()	-	DshCallback.callback_s7f27
S7F29 送信	DshS7F29Send	send(), send_wait()	-	DshCallback.callback_s7f29

### 3.5.2 RCP レシピ操作画面

レシピ情報の操作画面です。新規 ID 登録、指定された ID に対する情報設定/取得アクセスができます。また、関連するメッセージの送信もあります。

それぞれのボタンの機能は、ボタンの表示名のとおりです。操作結果は、LOG 画面に表示されます。



以下のクラスを使用します。

#### (1) 情報アクセス関連

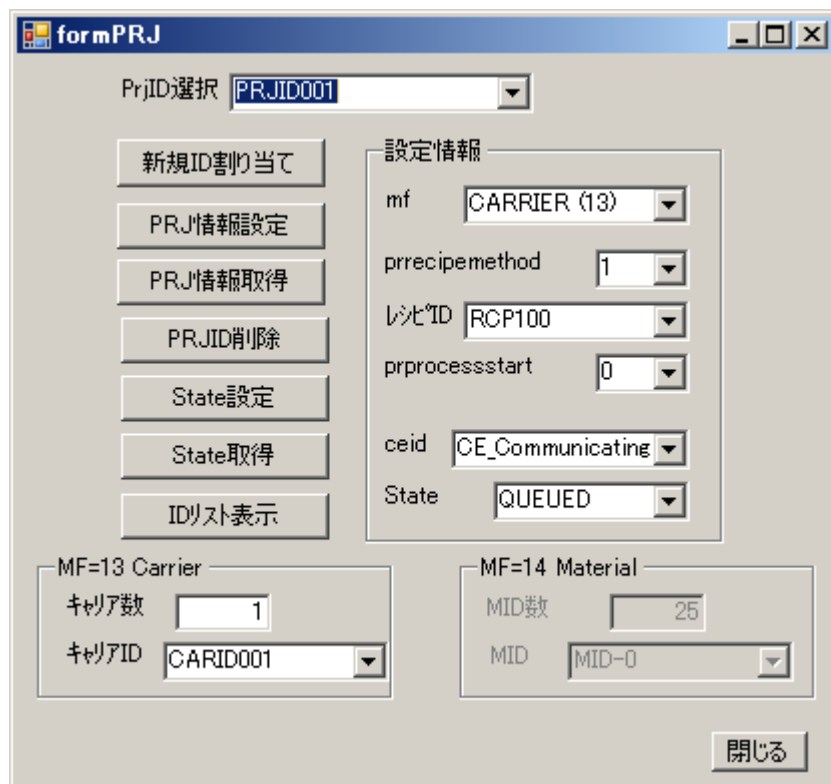
ボタン	クラス	メソッド
情報アクセス関連	DshRecipe	get_id_list(), alloc_id(), set_id(), set(), get(), delete()他

#### (2) メッセージ送信関連

ボタン	情報クラス	MSG 送信クラス	Msg 受信クラス	送信完了 callback クラス
S15F3 送信	DshS15F3Send	send(), send_wait()	DshS15Rsp	DshCallback.callback_s15f3
S15F5 送信	DshS15F5Send	send(), send_wait()	DshS15Rsp	DshCallback.callback_s15f5
S15F7 送信	DshS15F7Send	send(), send_wait()	DshS15Rsp	DshCallback.callback_s15f7
S15F9 送信	DshS15F9Send	send(), send_wait()	DshS15Rsp	DshCallback.callback_s15f9
S15F13 送信	DshS15F13Send	send(), send_wait()	DshS15Rsp	DshCallback.callback_s15f13
S15F17 送信	DshS15F17Send	send(), send_wait()	DshRcpS15F18Rsp	DshCallback.callback_s15f17

### 3.6 PRJ プロセス・ジョブ操作画面

PRJ プロセス・ジョブ情報操作画面です。新規 ID 登録、指定された ID に対する情報設定/取得アクセスができます。それぞれのボタンの機能は、ボタンの表示名のとおりです。操作結果は、LOG 画面に表示されます。



The screenshot shows a window titled "formPRJ" with a "PrjID選択" dropdown set to "PRJID001". On the left is a vertical stack of buttons: "新規ID割り当て", "PRJ情報設定", "PRJ情報取得", "PRJID削除", "State設定", "State取得", and "IDリスト表示". The main area is divided into "設定情報" (Settings) and two material sections. "設定情報" includes fields for "mf" (CARRIER (13)), "prrecipemethod" (1), "ジョブID" (RCP100), "prprocessstart" (0), "ceid" (CE\_Communicating), and "State" (QUEUED). The "MF=13 Carrier" section has "キャリア数" (1) and "キャリアID" (CARID001). The "MF=14 Material" section has "MID数" (25) and "MID" (MID-0). A "閉じる" button is at the bottom right.

以下のクラスを使用します。

ボタン	クラス	メソッド
情報アクセス関連	DshPrj	get_id_list(), alloc_id(), set_id(), set(), get(), delete()他

### 3.7 CJ コントロール・ジョブ操作画面

CJ コントロール・ジョブ情報操作画面です。新規 ID 登録、指定された ID に対する情報設定/取得アクセスができます。

それぞれのボタンの機能は、ボタンの表示名のとおりです。操作結果は、LOG 画面に表示されます。

以下のクラスを使用します。

ボタン	クラス	メソッド
情報アクセス関連	DshCj	get_id_list(), alloc_id(), set_id(), set(), get(), delete()他

### 3.8 Carrier キャリア操作画面

Carrier キャリアの操作画面です。新規 ID 登録、指定された ID に対する情報設定/取得アクセスができます。それぞれのボタンの機能は、ボタンの表示名のとおりです。操作結果は、LOG 画面に表示されます。



CarID選択: CARID 03

設定情報

IdStatus: 0 MapStatus: 0

AccStatus: 0

Usage: PRODUCT Location: SOURCE

Mid: K1234 SubstId: K1234

SlotCount: 0

Buttons:

- 新規ID登録
- IDリスト表示
- キャリア情報設定
- キャリア情報取得
- キャリアID削除
- IdStatus設定
- IdStatus取得
- MapStatus設定
- MapStatus取得
- Location設定
- Location取得
- Usage設定
- Usage取得
- AccStatus設定
- AccStatus取得
- SlotMap設定
- SlotMap取得

slot id

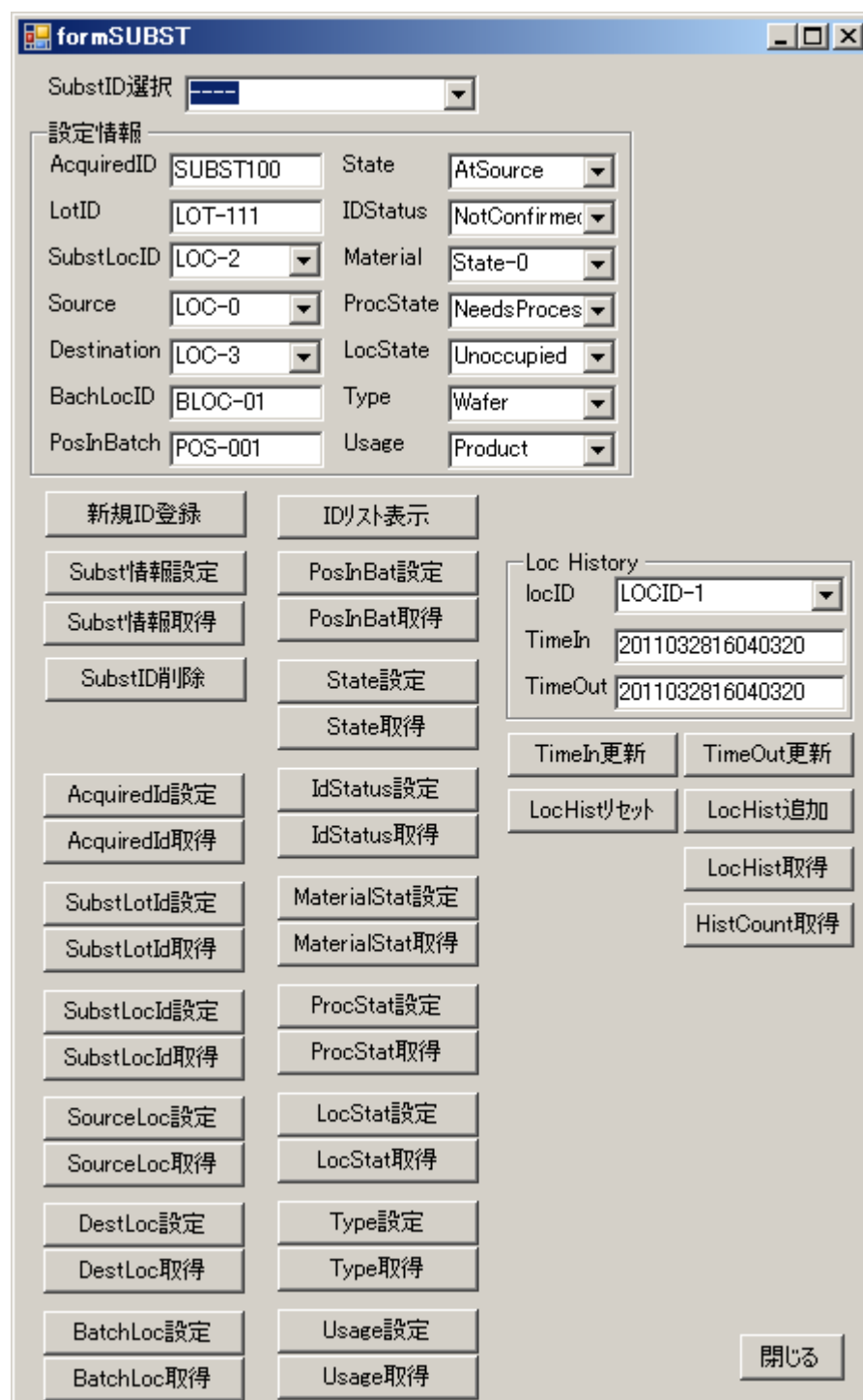
1	11	21
2	12	22
3	13	23
4	14	24
5	15	25
6	16	
7	17	
8	18	
9	19	
10	20	

slot pos(1,2,...,25): 0

閉じる

### 3.9 Substrate 基板操作画面

Substrate 基板の操作画面です。新規 ID 登録、指定された ID に対する情報設定/取得アクセスができます。それぞれのボタンの機能は、ボタンの表示名のとおりです。操作結果は、LOG 画面に表示されます。

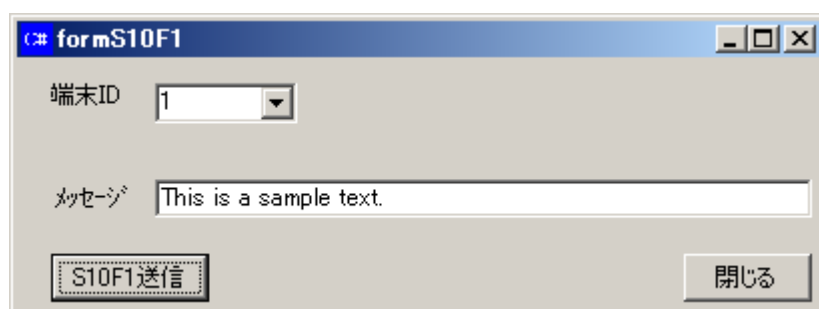


The screenshot shows the 'formSUBST' application window. At the top, there is a 'SubstID選択' dropdown menu. Below it is a '設定情報' (Setting Information) section with two columns of fields: 'AcquiredID' (SUBST100), 'LotID' (LOT-111), 'SubstLocID' (LOC-2), 'Source' (LOC-0), 'Destination' (LOC-3), 'BatchLocID' (BLOC-01), 'PosInBatch' (POS-001), 'State' (AtSource), 'IDStatus' (NotConfirmed), 'Material' (State-0), 'ProcState' (NeedsProces), 'LocState' (Unoccupied), 'Type' (Wafer), and 'Usage' (Product). Each field has a corresponding '設定' (Set) and '取得' (Get) button. To the right of these buttons is a 'Loc History' section with a 'locID' dropdown (LOCID-1), 'TimeIn' and 'TimeOut' text boxes (both containing 2011032816040320), and buttons for 'TimeIn更新', 'TimeOut更新', 'LocHistリセット', 'LocHist追加', 'LocHist取得', and 'HistCount取得'. At the bottom right is a '閉じる' (Close) button.



### 3.10 端末メッセージ送信画面

装置側から S10F1 メッセージを送信するための画面です。



以下のクラスを使用します。

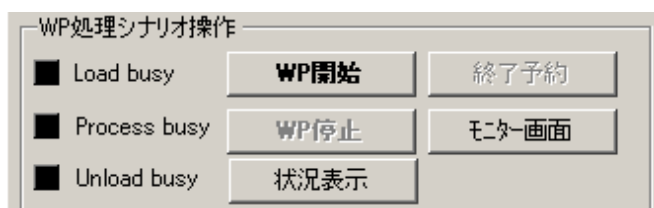
以下のクラスを使用します。

ボタン	クラス	メソッド
情報アクセス関連	DshTermMsg	get_id_list(), set_alid(), set_enabled()他
S10F1 送信	DshS10F1Send	send() - 非ブロックモード send_wait() - ブロックモード



### 3.11 WP ウェハー・プロセスオペレーション操作画面

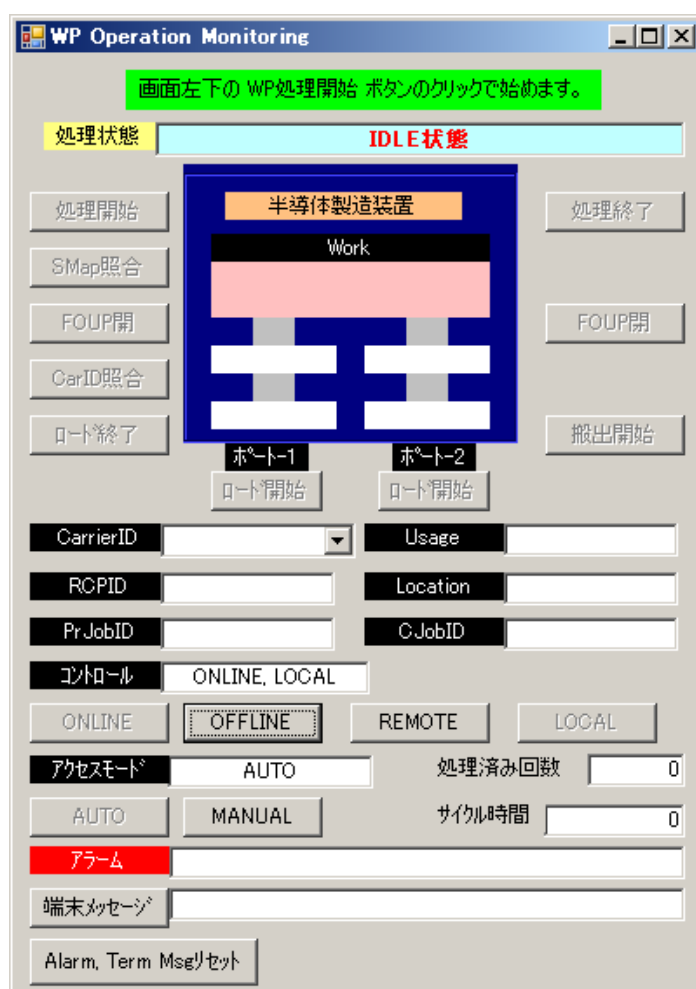
簡単な WP ウェハー・プロセス・シナリオのモデルを作り、それを実現するための操作です。  
詳しいシナリオについては 6. 2 ウェハー処理シナリオを参照ください。



メインフォームの **WP 開始** ボタンクリックによって開始します。開始によって、以下の処理を行います。

- (1) FormMonitor の表示 - オペレーションの進行状況を表示する画面です。  
下の3つのスレッドから与えられる情報によって表示を更新します。
- (2) シナリオを処理する3つのクラスを生成し、そのスレッドを開始します。  
 WpLoad クラス : キャリアの搬入制御を行います。  
 WpProc クラス : ウェハー処理制御をします。  
 WpUnload クラス : キャリアの搬出制御を行います。

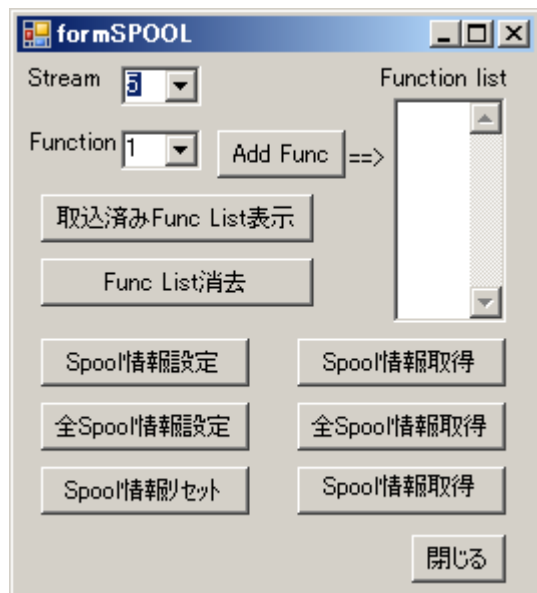
モニター画面



### 3.12 スプール設定操作画面

メッセージのスプールの設定操作画面です。

stream を選び、同じストリームの function を順次リストに追加し、それを設定します。



以下のクラスを使用します。

ボタン	クラス	メソッド
情報アクセス関連	DshSpool	get(), set(), add_func(), reset(), clear_stream()他

### 3.13 1次メッセージの送信操作画面

DSHENG4 エンジンでサポートしていないユーザ固有の1次メッセージを、ユーザが DSHDR2 通信ドライバーのメッセージ組み立て機能を使って準備し、DshEngine クラスの `send_request()` メソッドを使って、その1次メッセージを送信するサンプルを含む画面です。

例として、とりあえず、S5F1, S7F5, S1F1 の送信例が示されています。

デモプログラム formSendReq フォームには、メッセージ組み立てと送信、そして2次メッセージの受信と処理のコーディングが含まれています。

使用するクラスは次のとおりです。・

クラス	メソッド	構造体	備考
HSMS	D_InitItemPut(), D_PutItem()	DSHMSG	1 次 MSG 組立
	D_InitItemGet(), D_GetItem()	DSHMSG	2 次 MSG 解読
DshEngine	send_request() (非ブロックモード) send_request_wait() (ブロックモード)		1 次 MSG 送信
DshCallback	callback_send_request		

画面

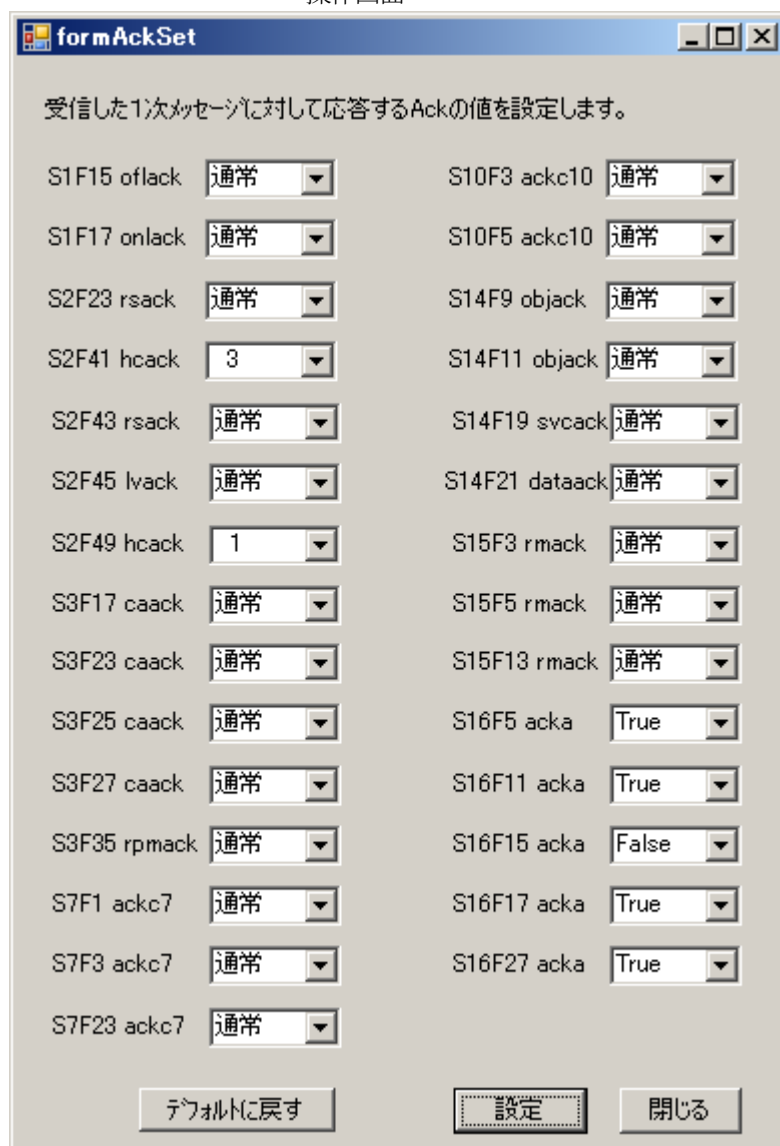


### 3.14 1次メッセージに対する応答ACK設定画面

ホストから受信する1次メッセージに返す応答メッセージにAck（1バイト）情報を故意にエラーの値に設定しするための画面です。本画面に表示されているメッセージを受信した時に、ここで設定したACKが応答メッセージにセットし、送信されます。

装置側が応答するメッセージに対するものだけです。

操作画面



受信した1次メッセージに対して応答するAckの値を設定します。

S1F15 oflack	通常	S10F3 ackc10	通常
S1F17 onlack	通常	S10F5 ackc10	通常
S2F23 rsack	通常	S14F9 objack	通常
S2F41 hcack	3	S14F11 objack	通常
S2F43 rsack	通常	S14F19 svcack	通常
S2F45 lvack	通常	S14F21 dataack	通常
S2F49 hcack	1	S15F3 rmack	通常
S3F17 caack	通常	S15F5 rmack	通常
S3F23 caack	通常	S15F13 rmack	通常
S3F25 caack	通常	S16F5 acka	True
S3F27 caack	通常	S16F11 acka	True
S3F35 rpmack	通常	S16F15 acka	False
S7F1 ackc7	通常	S16F17 acka	True
S7F3 ackc7	通常	S16F27 acka	True
S7F23 ackc7	通常		

デフォルトに戻す      設定      閉じる

### 3.15 クラスのトレースとデバッグ機能設定画面

DSHEng4Class のクラスの生成と消滅のトレースとその累積カウント監視に関する操作と、DSHENG4 通信エンジンならびに DSHDR2 HSMS 通信ドライバーのメモリ・リークチェックの有効、無効の操作を行います。

これらは、どちらもユーザ・プログラムの開発段階に使用できるデバッグ機能です。

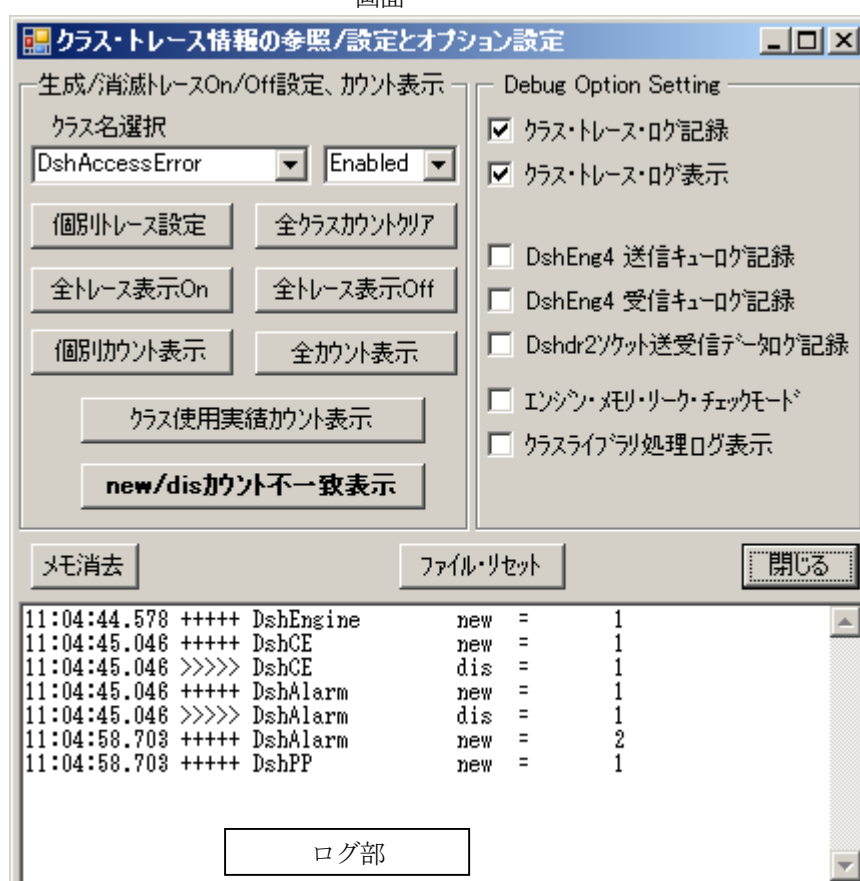
ここで設定されたチェックボックスの設定情報は、ファイルに保存され、次に起動された際に、前回の情報が復元されます。

クラスのトレースのプログラミングについては、下記説明書を参照してください。

文書番号 DSHENG4-09-30305-00 「DSHENG4CLASS クラス・ライブラリ プログラミングの手引き」

#### 6. 4 クラスのトレース機能

画面



Timestamp	Event	Class	Type	Count
11:04:44.578	+++++	DshEngine	new	1
11:04:45.046	+++++	DshCE	new	1
11:04:45.046	>>>>>	DshCE	dis	1
11:04:45.046	+++++	DshAlarm	new	1
11:04:45.046	>>>>>	DshAlarm	dis	1
11:04:58.703	+++++	DshAlarm	new	2
11:04:58.703	+++++	DshPP	new	1

### 3.15.1 トレース有効/無効の設定と実績表示機能

以下の操作ができます。表示結果は

#### (1) トレースの有効/無効の設定

トレースをするかしないかを画面上で操作します。

- ・ **個別トレース設定** : クラス名を選択し、クラス単位でトレースの有効/無効の設定操作します。
- ・ **全クラス有効設定** : 全クラスを1度にトレース有効/無効の設定します。
- ・ **全クラス無効設定**

#### (2) トレースの実績カウント情報表示

エンジン起動からカウントを行います。たとえば、実績クラス表示は次のようになります。

<new/dis/fin 実績クラス表示>					
DshAlarm	new=	1	dis=	1	(fin= 1)
DshCE	new=	1	dis=	1	(fin= 1)
DshEngine	new=	2	dis=	1	(fin= 0)
DshEngine	new=	1	dis=	0	(fin= 0)
DshSV	new=	1	dis=	1	(fin= 0)
total		5			

new : 生成、dis : Dispose fin : ファイナライズ

- ・ **個別カウント表示** : クラス名を選択し、個別に表示します。
- ・ **全カウント表示** : 全クラス分を表示します。
- ・ **クラス使用実績カウント表示** : 使用されたクラスについてだけ、表示する。
- ・ **new/dis カウント不一致表示** : 発生、消滅カウンタの値が一致しないクラスの情報だけを表示する。
- ・ **全クラスカウントクリア** : トレースを一旦、全てクリアすることもできます。

### 3.15.2 トレースログのファイル保存と表示

トレースの情報は、トレース画面のログ部に表示し、同時にディスクファイルに記録保存することができます。ファイルへの保存 On/Off、ログ表示の On/Off、 は、画面右のチェックボックスの操作で行います。

- ☐ クラス・トレース・ログ記録 - ログを保存するかどうかを選択します。
- ☐ クラス・トレース・ログ表示 - トレース画面のログ表示するかどうかを選択します。

ログ表示は、ログ記録が有効状態になっているとき有効になります。

ログファイル名は、debug.log です。DSHENG4.exe が起動されたディレクトリに保存されます。

保存、表示は、次のように行われます。

16:24:09.436	+++++	DshS6F11Send	new	=	2
16:24:09.468	>>>>>	DshS6F11Send	dis	=	2
16:24:10.385	+++++	DshCE	new	=	3
16:24:10.416	>>>>>	DshCE	dis	=	3

new, は生成、dis はDispose を意味します。後ろの数値は、累積回数です。



### 3.15.3 DSHENG4 エンジン、DSHDR2 通信ドライバーのメモリ・リークチェック操作

アプリケーション・ソフトウェアのプログラミングについては、メモリ・リークが発生していないかどうか気になる場所です。

メモリ・リークが発生しているかどうかは、通信エンジンの核である、DSHENG4.dll と dshdr2.dll の2つを対象に確認することができます。

メモリ・リークチェックの機能は、アプリケーションプログラムの開発段階で使用することを前提に提供されています。

#### (1) メモリ・リークチェックの設定

設定は、画面右側のチェックボックスの操作で行います。

☐ エンジン・メモリ・リーク・チェックモード

このチェックの変更は、DSHENG4 エンジンが開始される前に行う必要があります。

DSHENG4 エンジンが開始された後に変更するとプログラム異常が発生する可能性があります。

チェック(有効)の設定で、DSHDR2 のメモリ・リークチェックも連動します。

#### (2) メモリ・リークチェック結果は、エンジン停止後にデモプログラムの下の以下のファイルに保存されます。

DSHENG4.dll エンジン : malloc.log

DSHDR2.dll ドライバー : dr2\_malloc.log

### 3.15.4 その他の機能

その他、以下の機能をチェックボックスの有効設定で可能です。

チェックボックス	機能(有効の場合)	備考
<input type="checkbox"/> クラス処理ログ表示	クラスの内部デバッグ用	(DSHEng4Class 開発のため)
<input type="checkbox"/> DSHENG4 送信キューログ記録	APP からの 1 次メッセージ送信要求が受け付けられたかどうかの確認ログが DSHENG4 の共通ログファイルに追加されます。	
<input type="checkbox"/> DSHENG4 受信キューログ記録	上と同様に、相手から受信したメッセージが受け付けられたかどうかの確認ログを共通ログファイルに追加されます。	
<input type="checkbox"/> Dshdr2 ソケット送受信データログ記録	DSHDR2 HSMS 通信ドライバーが TCP/IP ソケットで送受信した生データを 16 進表記で DSHDR2 のログファイルに含めます。	

## 4．装置情報と通信メッセージ

ここでは、本デモプログラムが、装置－ホスト間の通信確立シナリオならびに、ウェハー処理シナリオ実行時に使用するメッセージならびに変数のデータフォーマットなどについて記述します。

(注) DSHENG4 は下表以外の GEM で使用するメッセージの送受信をサポートしています。

### 4．1 メッセージ

通信確立、ウェハー処理シナリオなどで使用するメッセージは次の通りです。

メッセージ一覧表

1 次 MSG	2 次 MSG	方向	役 割
S1F13	S1F14	H←→E	通信確立(エンジンが実施)
S1F15	S1F16	H→E	オフライン要求
S1F17	S1F18	H→E	オンライン要求
S3F17	S3F18	H→E	キャリアアクション要求
S3F27	S3F28	H→E	Change Access
S5F1	S5F2	H←E	アラーム報告
S6F11	S6F12	H←E	イベントレポート送信
S10F1	S10F2	H←E	端末要求
S10F3	S10F4	H→E	端末表示、シングルブロック
S10F5	S10F6	H→E	端末表示、マルチブロック
S14F9	S14F10	H→E	Create Object Request (Cj)
S14F11	S14F12	H→E	Delete Object Request (Cj)
S15F13	S15F14	H→E	Recipe Create Request
S16F11	S16F12	H→E	PrJob Create Enh

なお、メッセージに含まれる CEID, RPID, VID のデータフォーマットが符号無し 4 バイトか、符号無し 2 バイトのどちらかを選択できます。

## 4.2 変数一覧

変数（EC，SV，DVVAL）について説明します。

（この定義情報は、EQINFO.TXT ファイルに保存されています。）

デモ用の参考例として定義されているため、目的、変数値についてあまり意味がないものも含まれます。

### 4.2.1 EC - 装置定数

#	NAME	ID	Format	値	UNIT	備考
1	EC_Mdln	1	A[6]			S1F13 など用
2	EC_SoftRev	2	A[6]			S1F13 など用
3	EC_InitCommState	4	U2[1]	0～5		
4	EC_InitControlState	5	U2[1]	0～1		0=offline, 1=online
5	EC_InitOfflineSubState	6	U2[1]	1～3		
6	EC_ControlMode	80	U2[1]	0～2		
7	EC_PortAccessMode	256	U2[1]	0～2		
8	EC_Port1AccessMode	257	U2[1]	0～2		
9	EC_Port2AccessMode	258	U2[1]	0～2		
10	EC_ManualTimer	275	U1[1]	0～255		
11	EC_ManualInType	276	U1[1]			
12	EC_ManualOutType	278	U1[1]			
13	EC_MaxSpoolTransmit	512	U4[1]			
14	EC_OverWriteSpool	513	Boolean[1]			
15	EC_BinTest	1024	B[1]			フォーマットテスト用
16	EC_Int1Test	1025	I1[1]			フォーマットテスト用
17	EC_Int2Test	1026	I2[1]			フォーマットテスト用
18	EC_Int4Test	1027	I4[1]			フォーマットテスト用
19	EC_SingleTest	1028	F4[1]			フォーマットテスト用
20	EC_DoubleTest	1029	F8[1]			フォーマットテスト用

## 4.2.2 SV - 装置状態変数

#	NAME	ID	Format	値	UNIT	備考
1	SV_Clock	8192	A[16]			YYYYMMDDHHMMSSCC 年 月日時分秒 100ms
2	SV_CommunicationState	8193	U1[1]	0～5		0=disabled 1=enabled 2=not_communicating 3=wait CRA 4=wait delay 5=Communicating
3	SV_ControlState	8194	U2[1]	0～2		0=offline, 1=online local 2=online remote
4	SV_CJExecName1	8195	A[0..80]		CJOB	CJ 名 port-1
5	SV_CJExecName2	8196	A[0..80]		CJOB	CJ 名 port-2
6	SV_ReadyToLoad	8198	A[0..80]			ポート可能状態文字列
7	SV_LoadPortTransferStatus	8199	U1[1]	0～3		0=out of service 1=blocked 2=ready to load 3=ready to unload
8	SV_LoadPortId	8200	U1[1]	1～2		現ポート番号
9	SV_LoadCarId	8201	A[0..32]			ポートキャリア ID
10	SV_LoadCarIdStatus	8202	U1[1]	0～4		キャリア ID 読取り状態 0=not read 1=wait for host 2=Id verification OK 3=Id verification faild 4=carid not exists
11	SV_LoadSlotMapStatus	8203	U1[1]	0～3		スロットマップ 読取り状態
12	SV_LoadPortAssociationStatus	8204	U1[1]	0～1		0=not associated 1=associated
13	SV_LoadCarAccessStatus	8205	U1[1]	0～4		キャリア処理アクセス状態 0=procss not accessed 1=process in acess 2=process complete 3=process stopped 4=proc not exists
14	SV_LotId	8207	A[0..32]			ロット番号
15	SV_SubstID	8208	A[0..32]			基板 ID (未使用)
16	SV_SubstProcState	8209	U1[1]	0～1		基板処理アクセス状態 (未使用)
17	SV_SubstState	8210	U1[1]			基板状態(未使用)
18	SV_CarUsage	8211	A[0..32]			キャリア用途
19	SV_CarLocation	8212	A[0..32]			キャリア所在位置
20	SV_SubstList	82131	L	1		RPTID link nest 用 list

21	SV_UnloadPortId	8214	U1[1]	1～2		アンロードポート
22	SV_UnloadCarId	8215	A[0..32]			アンロードキャリア ID
23	SV_UnloadCarIdStatus	8216	U1[1]	0～5		アンロードキャリア状態(未使用)
24	SV_UnloadSlotMapStatus	8217	U1[1]	0～5		アンロードキャリアマップ 状態(未使用)
25	SV_UnloadPortAssociationStatus	8218	U1[1]	0～5		アンロードキャリアアソシエーション状態 (未使用)
26	SV_UnloadCarAccessStatus	8219	U1[1]	0～5		アンロードキャリアアクセス状態(未使用)
27	SV_AccessModeStatus	8220	U1[1]			(未使用)
28	SV_Access1ModeStatus	8221	U1[1]			(未使用)
29	SV_Access2ModeStatus	8222	U1[1]			(未使用)
30	SV_Port1ReservedStatus	8224	U1[1]			
31	SV_Port2ReservedStatus	8225	U1[1]			
32	SV_PPID	8226	A[1..80]			Process Program ID (=recipe id) S7F3/S7F23/S15F13 から得る
33	SV_PrjID	8227	A[1..80]			Process Job ID S16F11/S16F15 より得る
34	SV_PrjState	8228	A[1..80]			Process Job の状態 (文字列)
35	SV_CJID	8229	A[1..80]			Control Job ID S14F9 より得る
36	SV_CjState	8230	A[1..80]			Control Job の状態 (文字列)
37	SV_SpoolState	8231	U1[1]			スプールの状態 DSHENG4 が内部で使用する
38	SV_SpoolCountActual	8232	U4[1]			スプール数 DSHENG4 が内部で使用する
39	SV_SpoolCountTotal	8233	U4[1]			スプールカウント合計 DSHENG4 が内部で使用する
40	SV_SpoolFullTime	8234	A[16]			スプール満杯時刻記憶 DSHENG4 が内部で使用する
41	SV_SpoolStartTime	8235	A[16]			スプール開始時刻 DSHENG4 が内部で使用する
42	SV_WaferTransportState	8237	U1[1]			(未使用)
43	SV_WaferProcesStatus	8238	U1[]1			(未使用)
44	SV_WaferLocationStatus	8239	U1[1]			(未使用)
45	SV_List1	254	L[1]	5		CE イベント内、レポート ID にリンクされる SV リスト数
46	SV_List1_End	255	LEND[1]	0		同 SV リストの終わり
47	SV_List2	252	L[1]	3		CE イベント内、レポート ID にリンクされる SV リスト数
48	SV_List2_End	253	LEND[1]	0		同 SV リストの終わり
49	SV_LIST	251	LEND[1]	0		SV リストの終わり

#### 4.2.3 DVVAL- 装置データ変数

#	NAME	ID	Format	値	UNIT	備考
1	DV_StartTime	8300	A[19]			処理開始時刻 YYYY-MM-DD-HH:MM:SS
2	DV_EndTime	8301	A[19]			処理終了時刻 YYYY-MM-DD-HH:MM:SS
3	DV_Temp1	8302	A[8]		Degree	温度 1 (ASCII)
4	DV_Temp1_bin	8303	U2[1]		Degree	温度 1 (Binary)
5	DV_Temp1	8304	A[8]		Degree	温度 2 (ASCII)
6	DV_Temp1_bin	8305	U2[1]		Degree	温度 2 (Binary)

### 4.3 CEイベントID

#	NAME	ID	リンクポート ID	備考
1	CE_Communicating	2	RP_Communicating	通信確立通知
2	CE_SpoolDeactivated	999	(なし)	スプール終了通知
3	CE_ControlState	100	RP_ControlState	
4	CE_AlarmOn	200	(なし)	
5	CE_AlarmOff	201	(なし)	
6	CE_PortAccessMode	2200	RP_PortAccessMode	
7	CE_Port1AccessMode	2201	RP_Port1AccessMode	
8	CE_Port2AccessMode	2202	RP_Port2AccessMode	
9	CE_ReadyToLoad	15600	RP_ReadyToLoad	
10	CE_LoadTransferBlocked	15792	RP_LoadPort	
11	CE_LoadMaterialArrived	15793	RP_LoadPort	
12	CE_LoadWaitingForHost	15794	RP_LoadPort	
13	CE_LoadAssociated	15795	RP_LoadPort	
14	CE_IdVerifyOK	15796	RP_LoadPort	
15	CE_OpenFoup	15797	RP_LoadPort	
16	CE_SlotMapVerifyOK	15798	RP_LoadPort	
17	CE_SubstAtSource	15799	RP_SubstAtSource	
18	CE_SubstOccupied	15800	RP_SubstReport	
19	CE_PrJobPooled	15801	RP_PrJobReport	
20	CE_CJobQueued	15808	RP_CJobReport	
21	CE_CJobSelected	15809	RP_CJobReport	
22	CE_ExecBegan	15810	RP_CJobReport	
23	CE_PrJobSetup	15811	RP_PrJobReport	
24	CE_CarInAccess	15812	RP_LoadPort	
25	CE_SubstAtWork	15813	RP_SubstReport	
26	CE_SubstInProcess	15814	RP_SubstReport	
27	CE_PrJobProcessing	15815	RP_PrJobReport	
28	CE_SubstProcessingComplete	15816	RP_SubstReport	
29	CE_ProcessingComplete	15817	RP_PrJobReport	
30	CE_SubstAtDestination	15824	RP_SubstReport	
31	CE_PrJobComplete	15825	RP_PrJobReport	
32	CE_CJobComplete	15826	RP_CJobReport	
33	CE_CJobDeleted	15827	RP_CJobReport	
34	CE_CloseFoup	15828	RP_LoadPort	
35	CE_CarComplete	15829	RP_LoadPort	
36	CE_ReadyToUnload	15856	RP_LoadPort	
37	CE_UnLoadTransferBlocked	15857	RP_LoadPort	
38	CE_MaterialRemoved	15858	RP_LoadPort	
39	CE_CarDeleted	15859	RP_LoadPort	
40	CE_LoadNotAssociated	15861	RP_LoadPort	
41	CE_TestList	254	RP_LoadPort RP_TestList	複数 RPID のケースのテスト用

## 4.4 レポートID

#	NAME	ID	リンク VID	備考
1	RP_Communicating	10	SV_Clock	
2	RP_ControlState	100	SV_Clock SV_ControlState	
3	RP_PortAccessMode	1200	EC_PortAccessMode	
4	RP_Port1AccessMode	1201	EC_Port1AccessMode	
5	RP_Port2AccessMode	1202	EC_Port2AccessMode	
6	RP_ReadyToLoad	15600	SV_Clock SV_ReadyToLoad	
7	RP_LoadPort	15792	SV_Clock SV_LoadPortTransferStatus SV_LoadPortId SV_LoadCarId SV_LoadCarIdStatus SV_LoadSlotMapStatus SV_LoadPortAssociationStatus SV_LoadCarAccessStatus	
8	RP_SubstAtSource	15856	SV_Clock SV_LotId SV_SubstList SV_SubstID SV_ListEnd SV_CarLocation SV_SubstProcState SV_SubstState SV_CarUsage	
9	RP_SubstReport	15857	SV_Clock SV_LotId SV_SubstList SV_SubstID SV_ListEnd	
10	RP_PrjJobReport	15870	SV_Clock SV_PrjID SV_PrjState SV_PPID	
11	RP_CJobReport	15871	SV_Clock SV_CJID SV_CjState	
12	RP_UnloadPort	15808	SV_Clock SV_LoadPortTransferStatus SV_UnloadPortId SV_UnloadCarId SV_UnloadCarIdStatus SV_UnloadSlotMapStatus SV_UnloadPortAssociationStatus SV_UnloadCarAccessStatus	



13	RP_TestList	SV_Clock SV_UnloadPortId SV_List1 SV_UnloadCarId SV_List2 SV_UnloadCarId SV_UnloadCarIdStatus SV_List2_End SV_List2 SV_UnloadCarId SV_UnloadCarIdStatus SV_List2_End SV_UnloadCarIdStatus SV_List1_End SV_UnloadSlotMapStatus SV_UnloadPortAssociationStatus SV_UnloadCarAccessStatus	Neting 構造 report テスト用 他に意味はありません
----	-------------	---	-------------------------------------

## 4.5 アラームID

#	NAME	ID	ALCD	ALTX
1	AL_AlarmTempOver	1	2	"Chamber Temperature Over"
2	AL_AlarmPressure_1_Low	101	2	"Chamber Pressure-1 Over"
3	AL_AlarmPressure_2_Low	102	2	"Chamber Pressure-2 Over"
4	AL_AlarmPressure_3_Low	103	3	"Chamber Pressure-3 Over"

## 4.6 予約CEIDと変数

DSHENG4 が内部で自動処理するために必要な予約収集イベント ID と予約変数 ID の登録には、以下のものを使用します。

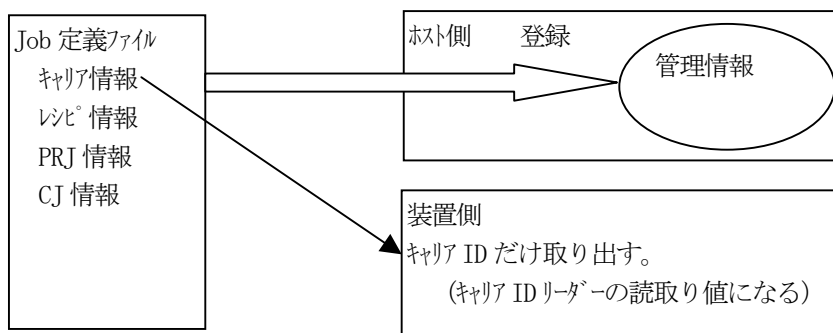
装置開始時に登録します。

CE/変数	予約インデックス	ID 名	備考
CE 収集イベント	CEX_RSV_COMMUNICATING	CE_Communicating	
	CEX_RSV_SPOOL_END	CE_SpoolDeactivated	
EC 装置定数	ECX_RSV_MDLN	EC_Mdln	
	ECX_RSV_SOFTREV	EC_SoftRev	
	ECX_RSV_SPOOL_MAX	EC_MaxSpoolTransmit	
	ECX_RSV_SPOOL_OVERWRITE	EC_OverWriteSpool	
	ECX_RSV_INIT_COMMSTATE	EC_InitCommState	
SV 状態変数	SVX_RSV_CLOCK	SV_Clock	
	SVX_RSV_COMMUNICATING	SV_CommunicationState	
	SVX_RSV_SPOOL_STATE	SV_SpoolState	
	SVX_RSV_SPOOL_TOTAL	SV_SpoolCountTotal	
	SVX_RSV_SPOOL_ACTUAL	SV_SpoolCountActual	
	SVX_RSV_SPOOL_STIME	SV_SpoolStartTime	
	SVX_RSV_SPOOL_FTIME	SV_SpoolFullTime	

## 5 . ジョブ定義ファイル

ウェハ処理を行うキャリアとそれに対する処理情報（レシピ、プロセスジョブ、コントロールジョブ）をテキストファイル内にジョブ単位で定義します。そして、実際のウェハ処理シミュレーション時に、そのファイルを指定し、そこで定義されている順にジョブ単位の処理を行います。このファイルには、1 個以上のジョブを登録することになります。

ジョブ定義ファイルで定義された情報は、全てホスト側のデモプログラムで管理情報として登録されます。また、装置側では、ポートへのロード時にキャリア ID だけを取り出して使用します。



### (1) 定義コマンド

#	メインコマンド	サブコマンド	意味
1	\$START		1 個のジョブの開始を示す。
2	\$END		1 個のジョブの終了を示す。
3	CARRIER		ジョブで処理されるキャリアを定義します。
		PORT	ロードポートを指定します。
		ID	キャリア ID を指定します。
		USAGE	usage を指定します。
		SLOT	スロット情報を指定します。(スロット ID, LOTID など)
4	RECIPE		レシピ (または プログラム) を定義します。
		ID	レシピ ID または PPID を指定します。
		BODY	RCPBODY (S15F13) または PPBODY (S7F3) を指定します。
		PARA	レシピ (S15F13) または FPP (S7F23) のパラメータを定義します。
		F_CCODE	FPP (S7F23) のコマンドコードを指定します。
5	PROCESS_JOB		プロセスジョブを定義します。
		ID	プロセスジョブ ID を指定します。
6	CONTROL_JOB		コントロールジョブを定義します。
		ID	コントロールジョブ ID を指定します。

コマンドの文字は大文字でも小文字でも構いません。

## (2) コマンド一般形式

```

$START
  <メインコマンド>{
    <サブコマンド>: <パラメータ 1> [, <パラメータ 2>[,.],....]
                      (必要なだけ)
  }
  <メインコマンド>{
    <サブコマンド>: <パラメータ 1> [, <パラメータ 2>[,.],....]
                      (必要なだけ)
  }
  .
  .
$END

```

- ①メインコマンドは、コマンド文字列の後、“{” 文字と “}” 文字で囲みます。
- ②サブコマンドは、その後に、“:” 文字を付け、その後、1 個以上のパラメータを続けます。  
パラメータ間は“,” で区切ります。

## (3) サブコマンド個別の形式

メインコマンド	サブコマンド	サブコマンドパラメータ
CARRIER	ID	<id> キャリア ID 文字列
	PORT	<port no.> 1 または 2
	USAGE	<usage> “PRODUCT”、“MONITOR”、“DUMMY”などの文字列
	SLOT	<slot#>, <slotid>, <ロット id>, <基板 id>, <ロケーション> <slot#> はスロット①順位 (0, 1,.. 25) <slotid> は 2 桁の数値 <ロット id> は文字列 <基板 ID> は文字列 <ロケーション> は文字列
RECIPE	ID	<id> レシピ または プロセスプログラム ID で、文字列
	BODY	<body> 文字列
	PARA	<name>, <value> <name>はパラメータ名で文字列 <value>はパラメータ値で、文字列
	f_ccode	<name>, <value1> [, <value2> [, <value3>, ].. ] <name>はコマンドコード名で文字列 <value>はコマンド値で、文字列
PROCESS_JOB	ID	<id> プロセスジョブ ID 文字列
CONTROL_JOB	ID	<id> コントロールジョブ ID 文字列

8. 4に1 個のジョブ情報のサンプルを示します。

## 6．実現するシナリオ

本デモプログラムでは、シナリオ処理例として、次の2つのシナリオを実現します。

- (1) 通信確立シナリオ
- (2) ウェハ処理シナリオ（正常ケース）

なお、レシピ（プロセスプログラム）ならびに、プロセスジョブ送信に使用する SECS-II メッセージを次のメッセージを使用します。

レシピ                   : S15F13  
プロセスジョブ       : S16F11

ウェハ処理シナリオ実行プログラムの中では、制御モードはオンライン・リモート、ポートアクセスモードはオートモードで行います。（デモプログラム内では、特にチェックしていません。）

また、ウェハ処理シナリオについては単なるサンプルですので、流れ、処理ステップ定義についても大雑把に決められています。

エラーに対する細かな処理も行っていないのでご了承ください。

### 6．1 通信確立シナリオ

	装置側デモプログラム	通信 MSG	ホスト側プログラム
1.	装置主導 Establish Communication Request CE_Communicating	S1F13 → ← S1F14 S6F11 → ← S6F12 ← S2F31 S2F32 ==>	Establish Communication Request 通信確立通知イベント Date and Time Set Request
2.	ホスト主導	← S1F13 S1F14 ==> S6F11 → ← S6F12 ← S2F31 S2F32 ==>	Establish Communication Request 通信確立通知イベント Date and Time Set Request

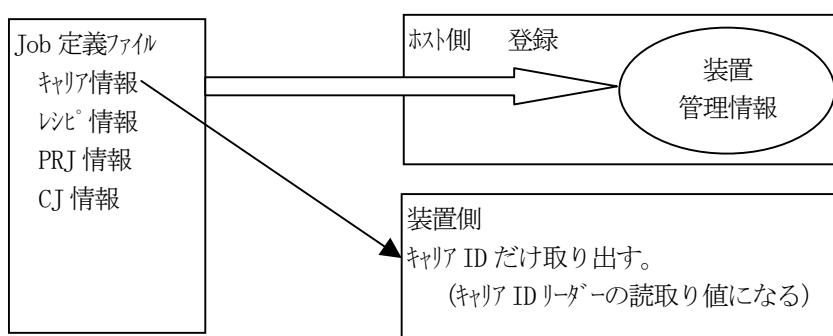
通信確立要求には、S1F13, S6F11 通信のやり取りは、GemEnable() API 関数が全てを処理してくれます。

## 6.2 ウェハ処理シナリオ（正常シナリオ）

### 6.2.1 ジョブ定義ファイル

ウェハ処理を行うキャリアとそれに対する処理情報（レシピ、プロセスジョブ、コントロールジョブ）をテキストファイル内にジョブ単位で定義します。そして、実際のウェハ処理シミュレーション時に、そのファイルを指定し、そこで定義されている順にジョブ単位の処理を行います。このファイルには、1個以上のジョブを登録することになります。

ジョブ定義ファイルで定義された情報は、全てホスト側のデモプログラムで管理情報として登録されます。また、装置側では、ポートへのロード時にキャリア ID だけを取り出して使用します。



#### (1) 定義コマンド

#	メインコマンド	サブコマンド	意味
1	\$START		1 個のジョブの開始を示す。
2	\$END		1 個のジョブの終了を示す。
3	CARRIER		ジョブで処理されるキャリアを定義します。
		PORT	ロードポートを指定します。
		ID	キャリア ID を指定します。
		USAGE	usage を指定します。
		SLOT	スロット情報を指定します。(スロット ID, LOTID など)
4	RECIPE		レシピ (またはプロセスプログラム) を定義します。
		ID	レシピ ID または PPID を指定します。
		BODY	RCPBODY (S15F13) または PPBODY (S7F3) を指定します。
		PARA	レシピ (S15F13) または FPP (S7F23) のパラメータを定義します。
		F_CCODE	FPP (S7F23) のコマンドコードを指定します。
5	PROCESS_JOB		プロセスジョブを定義します。
		ID	プロセスジョブ ID を指定します。
6	CONTROL_JOB		コントロールジョブを定義します。
		ID	コントロールジョブ ID を指定します。

コマンドの文字は大文字でも小文字でも構いません。

## (2) コマンド一般形式

```

$START
  <メインコマンド>{
    <サブコマンド>: <パラメータ 1> [, <パラメータ 2>[,.],....]
                      (必要なだけ)
  }
  <メインコマンド>{
    <サブコマンド>: <パラメータ 1> [, <パラメータ 2>[,.],....]
                      (必要なだけ)
  }
  .
  .
$END

```

- ①メインコマンドは、コマンド文字列の後、“{”文字と “}”文字で囲みます。
- ②サブコマンドは、その後に、“:”文字を付け、その後、1個以上のパラメータを続けます。  
パラメータ間は“,”で区切ります。

## (3) サブコマンド個別の形式

メインコマンド	サブコマンド	サブコマンドパラメータ
CARRIER	ID	<id> キャリア ID 文字列
	PORT	<port no.> 1 または 2
	USAGE	<usage> “PRODUCT”、“MONITOR”、“DUMMY”などの文字列
	SLOT	<slot#>, <slotid>, <ロット id>, <基板 id>, <ロケーション> <slot#> はスロット①順位 (0, 1, .. 25) <slotid> は 2 桁の数値 <ロット id> は文字列 <基板 ID> は文字列 <ロケーション> は文字列
RECIPE	ID	<id> レシピ または プロセスプログラム ID で、文字列
	BODY	<body> 文字列
	PARA	<name>, <value> <name>はパラメータ名で文字列 <value>はパラメータ値で、文字列
	f_ccode	<name>, <value1> [, <value2> [, <value3>, ].. ] <name>はコマンドコード名で文字列 <value>はコマンド値で、文字列
PROCESS_JOB	ID	<id> プロセスジョブ ID 文字列
CONTROL_JOB	ID	<id> コントロールジョブ ID 文字列

次ページに 1 個のジョブ情報のサンプルを示します。



(4) ジョブファイルサンプル (JobSche. txt)

```

$START
//-----
carrier{
    port:    1
    id:      "CARID_01"
    usage:   "PRODUCT"
    slot:    0, 10, "LOT_000", "SUBST_01_00", "LOC_A"    // 0
    slot:    1, 11, "LOT_001", "SUBST_01_01", "LOC_A"    // 1
    slot:    2, 12, "LOT_001", "SUBST_01_02", "LOC_A"    // 2
    slot:    3, 13, "LOT_001", "SUBST_01_03", "LOC_A"    // 3
    slot:    4, 14, "LOT_001", "SUBST_01_04", "LOC_A"    // 4
    slot:    5, 15, "LOT_001", "SUBST_01_05", "LOC_A"    // 5
    slot:    6, 16, "LOT_001", "SUBST_01_06", "LOC_A"    // 6
    slot:    7, 17, "LOT_001", "SUBST_01_07", "LOC_A"    // 7
    slot:    8, 18, "LOT_001", "SUBST_01_08", "LOC_A"    // 8
    slot:    9, 19, "LOT_001", "SUBST_01_09", "LOC_A"    // 9
    slot:    10, 20, "LOT_001", "SUBST_01_10", "LOC_A"    // 10
    slot:    11, 21, "LOT_001", "SUBST_01_11", "LOC_A"    // 11
    slot:    12, 22, "LOT_001", "SUBST_01_12", "LOC_A"    // 12
    slot:    13, 23, "LOT_001", "SUBST_01_13", "LOC_A"    // 13
    slot:    14, 24, "LOT_001", "SUBST_01_14", "LOC_A"    // 14
    slot:    15, 25, "LOT_001", "SUBST_01_15", "LOC_A"    // 15
    slot:    16, 26, "LOT_001", "SUBST_01_16", "LOC_A"    // 16
    slot:    17, 27, "LOT_001", "SUBST_01_17", "LOC_A"    // 17
    slot:    18, 28, "LOT_001", "SUBST_01_18", "LOC_A"    // 18
    slot:    19, 29, "LOT_001", "SUBST_01_19", "LOC_A"    // 19
    slot:    20, 30, "LOT_001", "SUBST_01_20", "LOC_A"    // 20
    slot:    21, 31, "LOT_001", "SUBST_01_21", "LOC_A"    // 21
    slot:    22, 32, "LOT_001", "SUBST_01_22", "LOC_A"    // 22
    slot:    23, 33, "LOT_001", "SUBST_01_23", "LOC_A"    // 23
    slot:    24, 34, "LOT_001", "SUBST_01_24", "LOC_A"    // 24
}
//-----
recipe{
    id:      "RCP-0001"
    body:    "RCPBODY-001-001-001-A"
    para:    "PARA10", "100.00"
    para:    "PARA11", "101.01"
    f_ccode: "fppcmd1", "1.00", "1.01", "1.02"
}
//-----
process_job{
    id:      "PJ-0001"
}
//-----
control_job{
    id:      "CJ-0001"
}
$END

```

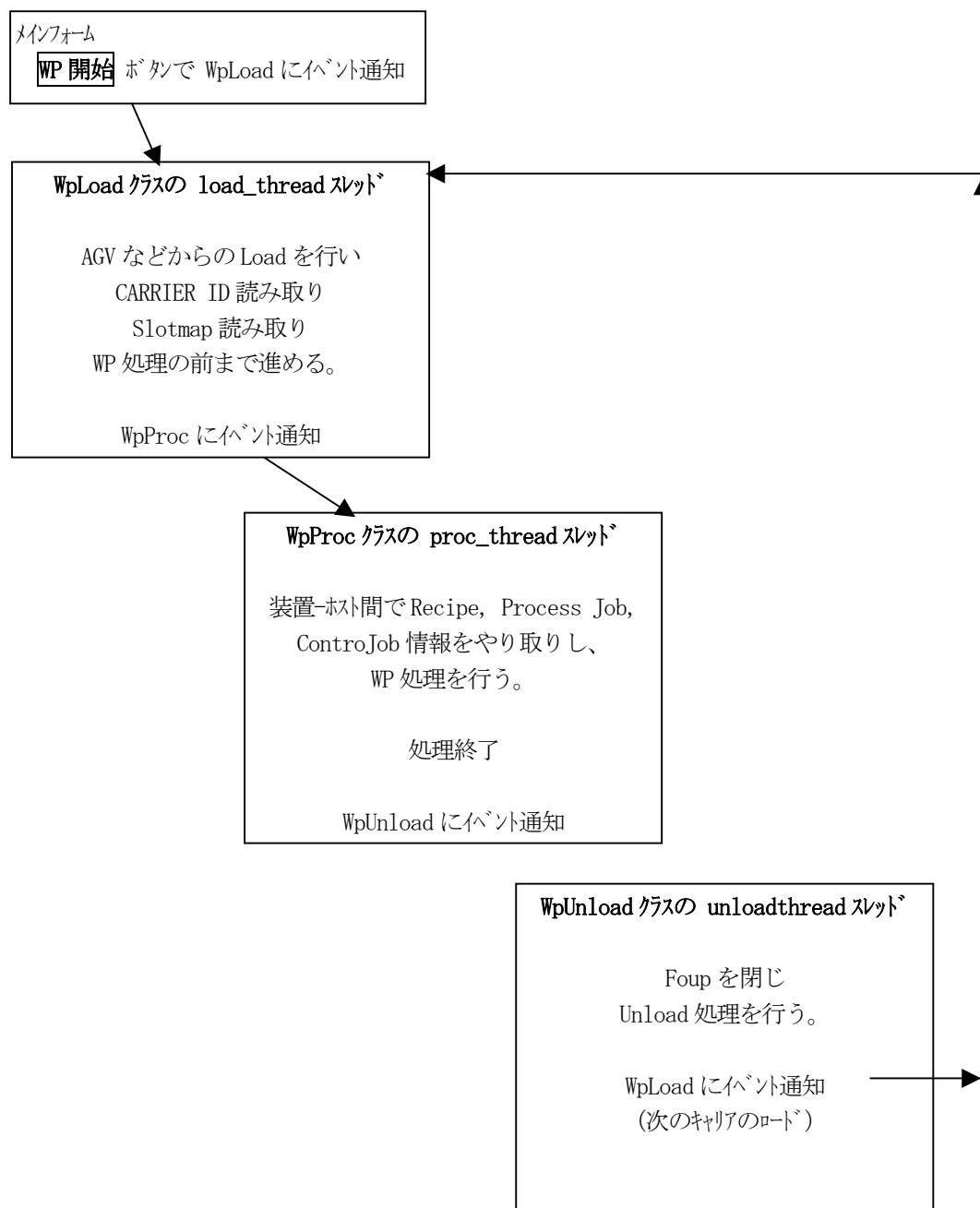
## 6.2.2 シナリオ詳細

以下のシーケンスで行う。本シナリオは、WpLoad, WpProc, WpUnload クラス内のスレッドで実行することになります。  
進行状況は WP オペレーションモニター画面に表示します。

シーケンスは、あくまでデモ用のものです。従って、通信の流れを簡単に実現するために細かいステップでの処理項目を省略しています。

シナリオは、ポートー 1, 2 から交互に搬入し、処理を行い、搬入と同じポートから搬出します。

処理は、Load, Processing, Unload の 3 つのクラスのスレッドによって行われます。



## 6.2.2.1 Load 処理 - WpLoad クラス

	装置側デモプログラム	通信 MSG	ホスト側プログラム	状態
1	(Load イベント受信) 搬送開始 - port-1		(Load イベント受信)	SOPE_LOAD_IDLE
2	CE_RwadyToLoad	S6F11 → ←== S6F12		SOPE_LOAD_START
3	CE_LoadTransferBlocked (イベント通知) (キャリア到着)	S6F11 → ←== S6F12	状態更新	SOPE_LOAD_END
4	CE_LoadMaterialArrived  (キャリア ID 読取り、登録)	S6F11 → ←== S6F12	キャリア ID 取り込み	SOPE_READ_CARID キャリア ID 登録
5	CE_LoadWaitingForHost (キャリア ID 付き)	S6F11 → ←== S6F12		
6	キャリアコンテンツ登録	← S3F17 S3F18 ==>	キャリアアクション CONTENT ProceedWithCarrier	SOPE_PORT_ASSOCIATED
7	CE_LoadAssociated	S6F11 → ←== S6F12	ID 確認 OK	SOPE_ID_VERIFY_END
8	(キャリア情報格納)		移動	
9	CE_IdVerifyOK	S6F11 → ←== S6F12		
10	CE_SubstAtSource	S6F11 → ←== S6F12		
11	CE_SubstOccupied  (Foup Open)	S6F11 → ←== S6F12		SOPE_OPEN_FOUP
12	CE_OpenFoup (スポットマップ 照合)	S6F11 → ←== S6F12		SOPE_READ_SLOTMAP
13	CE_LoadWaitingForHost	S6F11 → ←== S6F12		SOPE_SLOTMAP_VERIFY_END
		← S3F17 S3F18 ==>	キャリアアクション ProceedWithCarrier	
14	CE_SlotMapVerifyOK	S6F11 → ←== S6F12		SOPE_LOAD_IDLE (終了) WpProc スロット イベント通知

## 6.2.2.2 ウエハー処理 - WpProcess クラス

	装置側デモプログラム	通信 MSG	ホスト側プログラム	状態
1	(WpLoad からイベント)		(WpLoad からイベント)	SOPE_PROC_IDLE
2		<== S15F13 S15F14 ==>	ビット送信 3 つのうち 1 個	SOPE_PPID_READY ビット情報登録
3		<== S16F11 S16F12 ==>	プロセスジョブ送信 (または S16F15)	SOPE_PRJ_READY PRJ 情報登録
4	CE_PrJobPooled	S6F11 --> <== S6F12		
5		<== S14F9 S14F10 ==>		SOPE_CJ_READY CJ 情報登録
6	CE_CJobQueued	S6F11 --> <== S6F12		
7	CE_CJobSelected) (処理開始)	S6F11 --> <== S6F12		
8	CE_ExecBegan (処理中)	S6F11 --> <== S6F12		SOPE_PROC_START
9	CE_PrJobSetup	S6F11 --> <== S6F12		
10	CE_CarInAccess	S6F11 --> <== S6F12		
11	CE_SubstAtWork	S6F11 --> <== S6F12		
12	CE_SubstInProcess	S6F11 --> <== S6F12		
13	CE_PrJobProcessing	S6F11 --> <== S6F12		
14	(処理終了) CE_SubstProcessingComplete	S6F11 --> <== S6F12		SOPE_PROC_END

15	CE_ProcessingComplete	S6F11 → ←== S6F12		
16	CE_SubstAtDestination	S6F11 → ←== S6F12		
17	CE_PrJobComplete プロセスジョブ削除	S6F11 → ←== S6F12		SOPE_PRJ_COMPLETE
18	CE_CJobComplete	S6F11 → ←== S6F12		SOPE_CJ_COMPLETE
19	コントロールジョブ削除	← S14F11 S14F12 ==>	コントロールジョブ delete	
20	CE_CJobDeleted	S6F11 → ←== S6F12		SOPE_CJ_DELETE

## 6.2.2.3 Unload 処理 - WpUnload クラス

	装置側デモプログラム	通信 MSG	ホスト側プログラム	状態
	(WpProc からイベント)		(WpProc からイベント)	SOPE_UNLOAD_IDLE
1	(Foup 閉)			SOPE_CLOSE_FOUP
2	CE_CloseFoup	S6F11 → ← S6F12		
3	CE_CarComplete	S6F11 → ← S6F12		
4	CE_ReadyToUnload	S6F11 → ← S6F12		SOPE_UNLOAD_START
5	CE_LoadTransferBlocked	S6F11 → ← S6F12		
6	CE_MaterialRemoved	S6F11 → ← S6F12		
7	CE_LoadNotAssociated	S6F11 → ← S6F12		(SOPE_UNLOAD_END)
8	CE_ReadyToLoad (次のキャリア受入れ可)	S6F11 → ← S6F12		SOPE_UNLOAD_IDLE  WpLoad イベント (次のキャリア Load)

## 7．各種定義ファイルについて

DSHENG4 では、以下の定義ファイルを準備します。

- (1) 通信環境定義ファイル
- (2) 装置起動ファイル
- (3) 装置管理情報定義ファイル
- (4) PLC I/O デバイス定義ファイル

### 7.1 DSHDR2通信環境定義ファイル

DSHENG4 の HSMS 通信制御ドライバーDSHDR2.DLL に必要なファイルです。

装置側のチャンネルー 1 用のファイル例を示します。

```
#-----
# DSHDR2 環境ファイルのサンプル - HSMS-SS & SECS
#-----

START DSH
    MAX_MSG_SIZE = x100040      # max msg size = 1M + x40
    MAX_TRANSACTION = 512
    LOG_LINE = 100000
    LOG_FILE = c:\¥DSHENG4¥log¥DSHDR2.LOG
    LOG_TYPE = LIST              # List structure
    LOG_MODE = 0                 # save old log file
    MON_PORT = 9999              # Log Monitoring TCP Port
END

#-----
START PORT
    PORT = 1                     # HSMS
    PROTOCOL = HSMS              # SS
    PORT_MODE = ACTIVE
    TCP_PORT = 5001
    IP = 192.168.1.21            # remote passive ip addr
    T3 = 45
    T5 = 10
    T6 = 5
    T7 = 10
    T8 = 5
    LINKTEST = 60
END

#-----
START DEVICE
    DEVICE = 1
    DVID = x1234
    PORT = 1                     # using port-1
END
```

## 7.2 装置起動ファイル

装置別に必要な起動条件ファイルです。

例を次に示します。

```
//----- equip.cnf - 装置起動ファイル -----  
  
LOG_PATH      = c:\dsheng4\log  
LOG_MODE      = daily  
LOG_LIFE      = 6  
  
//LOG_MODE    = 0  
//LOG_FILE    = equip.log  
//LOG_SIZE    = 100000  
  
INFO_FILE     = c:\dsheng4\cnf\eqinfo.fil  
BKUP_PATH     = c:\dsheng4\backup  
INFO_BACKUP   = 1  
PP_COUNT      = 100  
FPP_COUNT     = 64  
RCP_COUNT     = 200  
CAR_COUNT     = 80  
CAR_CAPACITY  = 80  
SUBST_COUNT   = 250  
CJ_COUNT      = 32  
PRJ_COUNT     = 30  
TRACE_COUNT   = 28  
SPOOL_PATH    = c:\dsheng4\spool  
COMM_PORT     = 1  
COMM_DEVICE   = 1  
S1F13_SEND    = 2
```



### 7.3 装置管理情報定義ファイル

装置の管理情報（装置変数、収集イベント、レポート、アラームなど）を定義するファイルです。

Eng3Conf.exe プログラムでコンパイルした上で DSHENG4 で使用します。

次に、例を示します。（一部抜粋したものです。）

```
//----- C:\DSHENG4\bin\EQINFO.TXT -----

//----- EC - Equipment Constant -----
def_ec EC_MdlIn{
    ecid:      1                      // =0x00000001
    format:    A[6..6]
    nominal:   "DSH100"
}

def_ec EC_SoftRev{
    ecid:      2                      // =0x00000002
    format:    A[6..6]
    nominal:   "REV001"
}

def_ec EC_InitCommState{
    ecid:      4                      // =0x00000004
    format:    U2[1]
    min:       "0"
    max:       "1"
    nominal:   "0"
}

def_ec EC_InitControlState{
    ecid:      5                      // =0x00000005
    format:    U2[1]
    min:       "0"
    max:       "1"
    nominal:   "0"
}

def_ec EC_InitOfflineSubState{
    ecid:      6                      // =0x00000006
    format:    U2[1]
    min:       "1"
    max:       "3"
    nominal:   "1"
}
```

(以下省略)

## 7.4 ジョブファイルサンプル (JobSche.txt)

```

$START
//-----
carrier{
    port: 1
    id: "CARID_01"
    usage: "PRODUCT"
    slot: 0, 10, "LOT_000", "SUBST_01_00", "LOC_A" // 0
    slot: 1, 11, "LOT_001", "SUBST_01_01", "LOC_A" // 1
    slot: 2, 12, "LOT_001", "SUBST_01_02", "LOC_A" // 2
    slot: 3, 13, "LOT_001", "SUBST_01_03", "LOC_A" // 3
    slot: 4, 14, "LOT_001", "SUBST_01_04", "LOC_A" // 4
    slot: 5, 15, "LOT_001", "SUBST_01_05", "LOC_A" // 5
    slot: 6, 16, "LOT_001", "SUBST_01_06", "LOC_A" // 6
    slot: 7, 17, "LOT_001", "SUBST_01_07", "LOC_A" // 7
    slot: 8, 18, "LOT_001", "SUBST_01_08", "LOC_A" // 8
    slot: 9, 19, "LOT_001", "SUBST_01_09", "LOC_A" // 9
    slot: 10, 20, "LOT_001", "SUBST_01_10", "LOC_A" // 10
    slot: 11, 21, "LOT_001", "SUBST_01_11", "LOC_A" // 11
    slot: 12, 22, "LOT_001", "SUBST_01_12", "LOC_A" // 12
    slot: 13, 23, "LOT_001", "SUBST_01_13", "LOC_A" // 13
    slot: 14, 24, "LOT_001", "SUBST_01_14", "LOC_A" // 14
    slot: 15, 25, "LOT_001", "SUBST_01_15", "LOC_A" // 15
    slot: 16, 26, "LOT_001", "SUBST_01_16", "LOC_A" // 16
    slot: 17, 27, "LOT_001", "SUBST_01_17", "LOC_A" // 17
    slot: 18, 28, "LOT_001", "SUBST_01_18", "LOC_A" // 18
    slot: 19, 29, "LOT_001", "SUBST_01_19", "LOC_A" // 19
    slot: 20, 30, "LOT_001", "SUBST_01_20", "LOC_A" // 20
    slot: 21, 31, "LOT_001", "SUBST_01_21", "LOC_A" // 21
    slot: 22, 32, "LOT_001", "SUBST_01_22", "LOC_A" // 22
    slot: 23, 33, "LOT_001", "SUBST_01_23", "LOC_A" // 23
    slot: 24, 34, "LOT_001", "SUBST_01_24", "LOC_A" // 24
}
//-----
recipe{
    id: "RCP-0001"
    body: "RCPBODY-001-001-001-A"
    para: "PARA10", "100.00"
    para: "PARA11", "101.01"
    f_ccode: "fppcmd1", "1.00", "1.01", "1.02"
}
//-----
process_job{
    id: "PJ-0001"
}
//-----
control_job{
    id: "CJ-0001"
}
$END

```

## 8 . バックアップファイルとログファイル

管理情報バックアップファイル、各種ログファイルが生成されます。



- (1) 管理情報バックアップファイル  
DSHENG4 が管理する情報のバックアップファイルです。  
変数 (EC, SV, DVVAL)、プロセスプログラム (PP)/レシビ、プロセスジョブ (PRJ)、コントロールジョブ (CJ) キャリア、基板情報が対象です。装置起動ファイル内に指定されるディレクトリにバックアップされます。  
バイナリ形式のファイルです。
- (2) 通信ログファイル  
DSHDR2 通信ドライバーが相手装置と行った通信の SECS-II レベルのメッセージがリスト構造で記録されます。  
日付別にファイルを作成することができます。
- (3) DSHENG4 装置共通ログファイル  
装置固有でない、DSHENG4 のログ情報が記録されます。内部プログラムの開始、終了情報も含まれます。  
日付別にファイルを作成することができます。  
ユーザプログラムも Kprintf() 関数を使って文字列メッセージを加えることができます。
- (4) DSHENG4 装置 ID 別ログファイル  
装置 ID 別に作成されるログファイルです。日付別にファイルを作成することができます。  
ユーザプログラムも Kprintf() 関数を使って文字列メッセージを加えることができます。
- (5) デモプログラムログファイル  
デモプログラムがアプリケーションレベルのログを記録します。