GEM 通信エンジン・DSHGemClass (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月

株式会社データマップ

文書番号　DSHGEM-LIB-07-30501-02

［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】

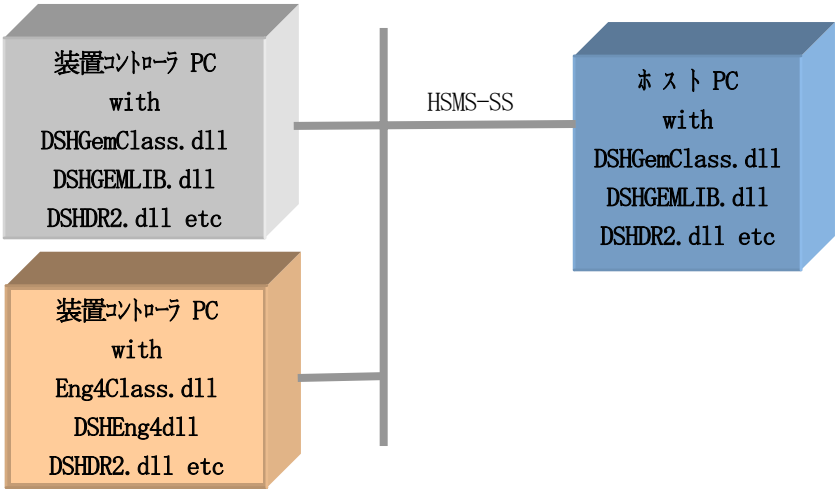
番号	改訂日付	項 目	概 略
1.	2011.4	初版	DSHGEM-07-384-00 の改定したもの DshGemClass ｸﾗｽﾗｲﾌﾞﾗﾘ使用のﾃﾞﾓﾌﾟﾛｸﾞﾗﾑ版。
2.	2011.6	追加されたﾌﾞﾛｯｸﾓｰﾄﾞによる 1 次ﾒｯｾｰｼﾞ送信関連 WP ｼﾐｭﾚｰｼｮﾝをｽﾚｯﾄﾞで実行するようにした。	S6f11,S5F1 送信をそれぞれ、収集ｲﾍﾞﾝﾄ, ｱﾗｰﾑ画面に移した。（元はﾒｲﾝ画面にあった。） 3.3、3.4 参照 load, processing, unload３つのｸﾗｽを設け、その中のｽﾚｯﾄﾞで行うようにした。 1 次ﾒｯｾｰｼﾞの送信はﾌﾞﾛｯｸﾓｰﾄﾞで行う。 また、ﾌﾟﾛｸﾞ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

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 DSHGEM-LIB 通信エンジンならびに DSHGemClass 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、ホスト、装置双方の機能を有し、切り替えでどちらかを選択できます。

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。



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。

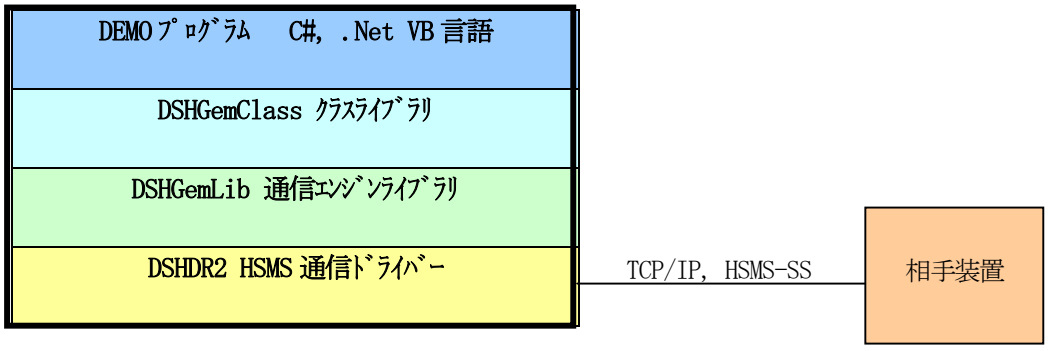
GEM レベルでの通信制御ならびに装置情報の管理には、DSHGemClass クラス・ライブラリに提供されるクラス群を利用するプログラミングになります。

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、簡単なウエハー処理 (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 DSHGemClass 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。

DSHGemClass 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。
DSHGEM-CLASS クラス・ライブラリ　**文書番号　DSHGEM-LIB-07-30305-02　「 プログラミングの手引き-V2 」**

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　DSHGEM-LIB 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）DSHGEMLIB ユーザーズ・ガイド、一般関連ドキュメント - ¥dshgemlib¥doc へ保存

#	文書番号	文書名	注釈
1	DSHGEM-LIB-07-30300-00	DSHGEM-LIB 通信制御ｴﾝｼﾞﾝﾗｲﾌﾞﾗﾘ(SECS/HSMS) ﾕｰｻﾞｰｽﾞ・ｶﾞｲﾄﾞ	DSHGEM-LIB の全般的な機能の説明書です。
2	DSHGEM-LIB-07-30301-00	DSHGEM-LIB 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	DSHGEM-LIB-07-30302-00	DSHGEM-LIB 装置管理情報定義仕様書 （変数、収集ｲﾍﾞﾝﾄ、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	DSHGEM-LIB-07-30303-00	装置管理情報定義ﾌｧｲﾙｺﾝﾊﾟｲﾗ説明書	DSHENG3 と共通です。
5	DSHGEM-LIB-07-30304-00	DSHGEM-LIB への手引き	DSHGEM-LIB 導入時に参考にする作業手順書です。
6	DSHGEM-LIB-07-30305-00	ｲﾝｽﾄｰﾙと保存ﾌｧｲﾙ	（この説明書です。）
7	DSHGEM-LIB-07-30308-00	DSHGEM-LIB, DSHENG3 起動ﾌｧｲﾙ、装置管理情報ﾌｧｲﾙ設定・編集ﾌﾟﾛｸﾞﾗﾑ説明書	DSHENG3,DSHEng4 共通
8	DSHGEM-LIB-07-30310-00	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	DSHGEM-LIB-07-30340-00	ﾕｰｻﾞ作成ﾗｲﾌﾞﾗﾘ関数 ２次ﾒｯｾｰｼﾞ応答関数一覧表	C, C++言語によるﾌﾟﾛｸﾞﾗﾐﾝｸﾞ .Net 用ｸﾗｽﾗｲﾌﾞﾗﾘを使用しない
10	DSHGEM-LIB-07-30351-00	ﾊﾞｯｸｱｯﾌﾟﾌｧｲﾙ参照ﾌﾟﾛｸﾞﾗﾑ説明書	DOS ｺﾏﾝﾄﾞで List 構造で 表示します・
11	DSHGEM-LIB-07-30340-00	ﾕｰｻﾞ作成ﾗｲﾌﾞﾗﾘ関数 ２次ﾒｯｾｰｼﾞ応答関数一覧表	C, C++言語によるﾌﾟﾛｸﾞﾗﾐﾝｸﾞ .Net 用ｸﾗｽﾗｲﾌﾞﾗﾘを使用しない
12	DSHGEM-LIB-07-30351-00	ﾊﾞｯｸｱｯﾌﾟﾌｧｲﾙ参照ﾌﾟﾛｸﾞﾗﾑ説明書	DOS ｺﾏﾝﾄﾞで List 構造で 表示します・

（2）DSHGemClass クラス・ライブラリ関連ドキュメント　¥dshgemlib¥doc-class

#	文書番号	文書名	注釈
1	DSHGEM-07-30361-00	ClassLib-Info-1 Vol-１　エンジン起動と管理情報クラス 編　Part-1	ｴﾝｼﾞﾝ、装置起動 管理情報のｱｸｾｽ
2	DSHGEM-07-30362-00	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	DSHGEM-07-30363-00	ClassLib-Comm Vol-２　メッセージ通信クラス 編	GEM ﾒｯｾｰｼﾞ送信
4	DSHGEM-07-30305-00	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ﾌﾟ 手順も含む
5	DSHGEM-07-30306-00	クラス生成・消滅トレースと表示機能について	ｸﾗｽ・ﾃﾞﾊﾞｯｸﾞ用

（3）DSHEngLib 通信エンジン ライブラリ関連ドキュメント ¥dshgemlib¥doc-lib

#	文書番号	タイトル名と内容
1	DSHGEM-LIB -09-30321-00	１．概要 ２．DSHGEM-LIB 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 DSHGEM-LIB 初期設定関連関数 ３．２ 通信制御関連関数
2	DSHGEM-LIB -09-30322-00	３．３ 変数（EC, SV, DVVAL）情報アクセスと通信サービス
3	DSHGEM-LIB -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	DSHGEM-LIB -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	DSHGEM-LIB -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	DSHGEM-LIB -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	DSHGEM-LIB -09-30327-00	３．13 RCP レシピ情報アクセスサービス関数
8	DSHGEM-LIB -09-30328-00	３．14 CAR キャリア情報アクセスサービス関数
9	DSHGEM-LIB -09-30329-00	３．15 SUBST 基板情報アクセスサービス関数
10	DSHGEM-LIB -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	DSHGEM-LIB -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	DSHGEM-LIB -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	DSHGEM-LIB -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	DSHGEM-LIB -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	DSHGEM-LIB -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 ¥dshgemlib¥doc

#	文書番号	文書名	注釈
1	DSHDR2-06-20000-02	DSHDR2 SECS/HSMS ﾚﾍﾞﾙ-2 通信制御ﾄﾞﾗｲﾊﾞｰ ﾕｰｻﾞｰｽﾞﾏﾆｭｱﾙ	SECS/HSMS 通信制御ﾄﾞﾗｲﾊﾞｰの説明書です。
2	DSHDR2-06-20040-0	DSHDR2 ﾚﾍﾞﾙ 2 通信ﾄﾞﾗｲﾊﾞｰ通信ﾛｸﾞﾓﾆﾀｰ説明書	ﾘｱﾙﾀｲﾑで通信ﾄﾗﾝｻﾞ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 ¥dshgemlib¥doc-demo

#	文書番号	文書名	注釈
1	DSHGEM-LIB-07-30501-02	ｸﾗｽ・ﾗｲﾌﾞﾗﾘ・ﾃﾞﾓﾌﾟﾛｸﾞﾗﾑ-V2 説明書	(本ﾄﾞｷｭﾒﾝﾄです。)
2	DSHGEM-LIB-07-30502-02	DSHGemClass ｸﾗｽ･ﾗｲﾌﾞﾗﾘ版 ﾃﾞﾓﾌﾟﾛｸﾞﾗﾑ インストールと保存ファイル	C#, .Net VB ﾃﾞﾓﾌﾟﾛｸﾞﾗﾑです。

1.2 デモ関連ファイルの保存場所

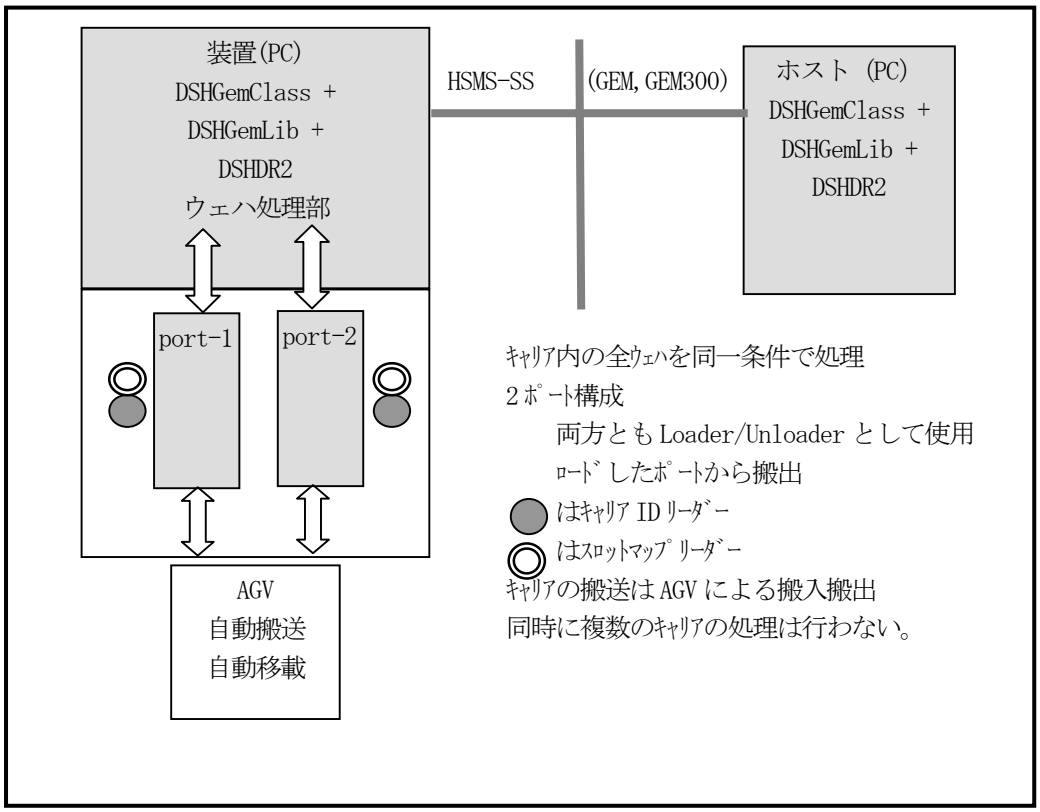
デモプログラムとして提供される setup.exe を実行すると、以下のディレクトリに必要がファイルが保存されます。
c:ドライブの ¥dshgemlib の下に保存されます。

home dir	path		ﾌｧｲﾙ名	種類、機能など
¥dshgemlib¥	bin		GemCsDemoV2.exe GemVbDemoV2.exe dshgemlib.dll, dshgemclass.dll など	実行ﾌﾟﾛｸﾞﾗﾑ・ﾌｧｲﾙ C#の実行ﾌﾟﾛｸﾞﾗﾑ VB の実行ﾌﾟﾛｸﾞﾗﾑ ｴﾝｼﾞﾝ・ﾌﾟﾛｸﾞﾗﾑ ｸﾗｽﾗｲﾌﾞﾗﾘ・ﾌﾟﾛｸﾞﾗﾑ
	cnf		equip00.cnf comm_eq.def host00.cnf comm_host.def eqinfo.fil	環境定義ﾌｧｲﾙ EQ 側 - 装置起動ﾌｧｲﾙ DSHDR2 HSMS 通信定義ﾌｧｲﾙ HOST 側 装置起動ﾌｧｲﾙ(host) DSHDR2 HSMS 通信定義ﾌｧｲﾙ 共通 装置管理情報定義ﾌｧｲﾙ
	tool		dshgemset.exe dshcompile.exe seeback.exe	ﾂｰﾙ・ﾌﾟﾛｸﾞﾗﾑ 装置管理情報編集ﾌﾟﾛｸﾞﾗﾑ 同ｺﾝﾊﾟｲﾙ・ﾌﾟﾛｸﾞﾗﾑ 装置情報ﾊﾞｯｸｱｯﾌﾟ参照ﾌﾟﾛｸﾞﾗﾑ
	logmon		logmon.exe	HSMS 通信ﾛｸﾞ・ﾓﾆﾀｰ(ﾘﾓｰﾄ) ﾘｱﾙﾀｲﾑ・HSMS 通信ﾛｸﾞﾓﾆﾀｰ・ﾌﾟﾛｸﾞﾗﾑ
	GemCsDemoV2¥		gemcsdemo.sln	C#ﾃﾞﾓ・ﾌﾟﾛｸﾞﾗﾑ ｿﾘｭｰｼｮﾝ・ﾌｧｲﾙ
		GemCsDemoV2	*.cs	C#ﾃﾞﾓ・ﾌﾟﾛｸﾞﾗﾑ ｿｰｽﾌｧｲﾙ
	GemCsDemoV2¥		GemCsDemoV2.sln	VB ﾃﾞﾓ・ﾌﾟﾛｸﾞﾗﾑ ｿﾘｭｰｼｮﾝ・ﾌｧｲﾙ
		GemVbDemoV2	*.vb	VB ﾃﾞﾓ・ﾌﾟﾛｸﾞﾗﾑ ｿｰｽﾌｧｲﾙ
	doc			ﾕｰｻﾞｰｽﾞ・ｶﾞｲﾄﾞ 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			dshGemClass ｸﾗｽ・ﾗｲﾌﾞﾗﾘ関連説明書
	doc-lib			dshgemlib 通信ｴﾝｼﾞﾝ・ﾗｲﾌﾞﾗﾘ関連説明書
	doc-demo			ﾃﾞﾓﾌﾟﾛｸﾞﾗﾑ関連説明書

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、DSHGemClass, DSHGemLib 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。

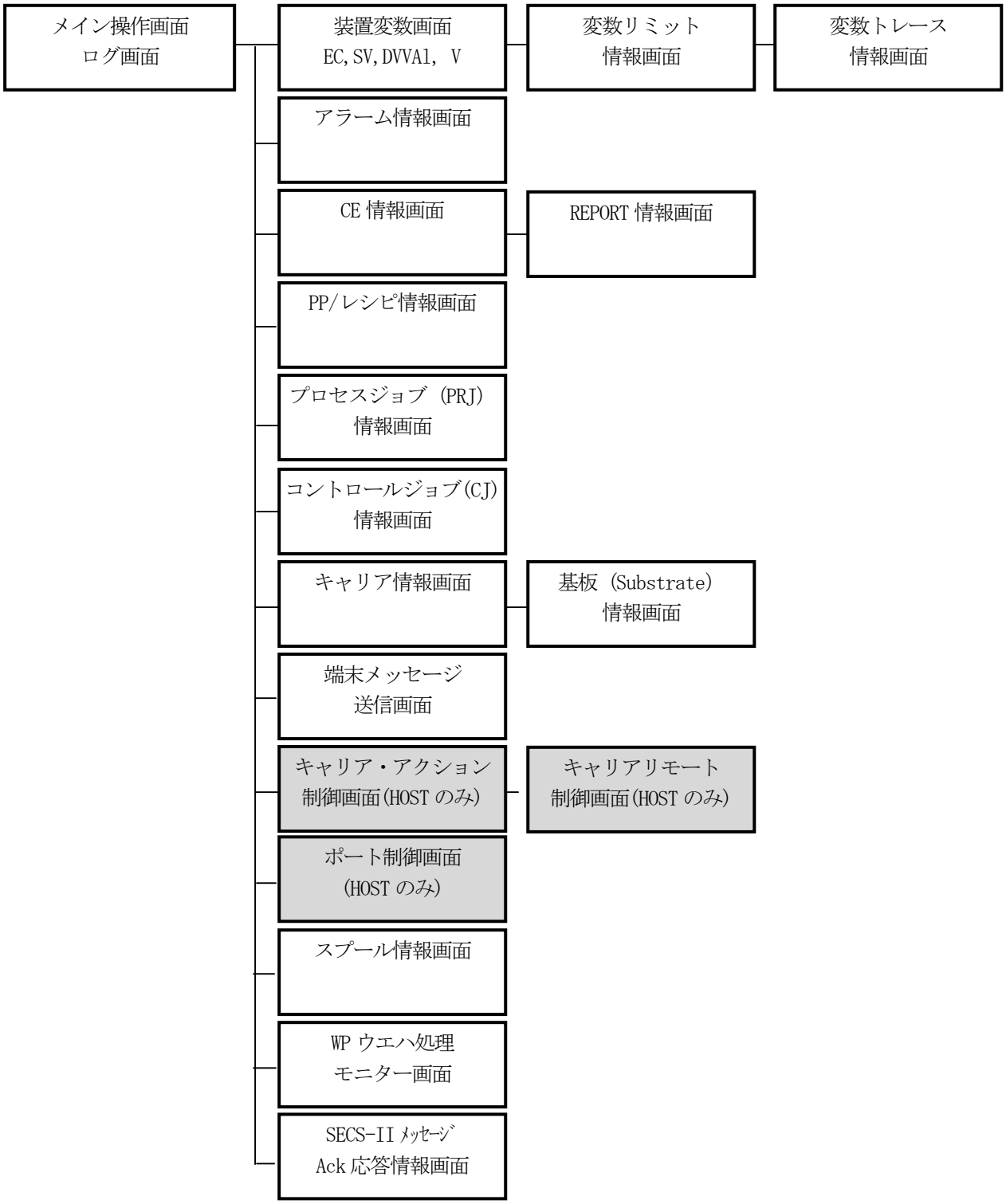
デモプログラムでは、簡単なシナリオモデルにより、WP (Wafer Processing) の連続処理を行う機能も含んでいます。これを WP シミュレーションと呼ぶことにします。

シナリオの中では、装置側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

DSHGEMLIB エンジンならびに DSHGemClass 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、DSHGemClass 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、S3F17 メッセージの送信クラスは、DshS3F17Send になります。

詳しい、説明は以下を参照ください。

文書番号 DSHGEM-07-30363-00-ClassLib-Comm　「クラスライブラリ Vol-2　通信編」
文書番号 DSHGEM-LIB-07-30305-00　「クラスライブラリ プログラミングの手引き V2」
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、DshEquipment 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 DSHGEM-LIB-07-30305-00　「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 DshEquipment 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。DshEquipment 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。

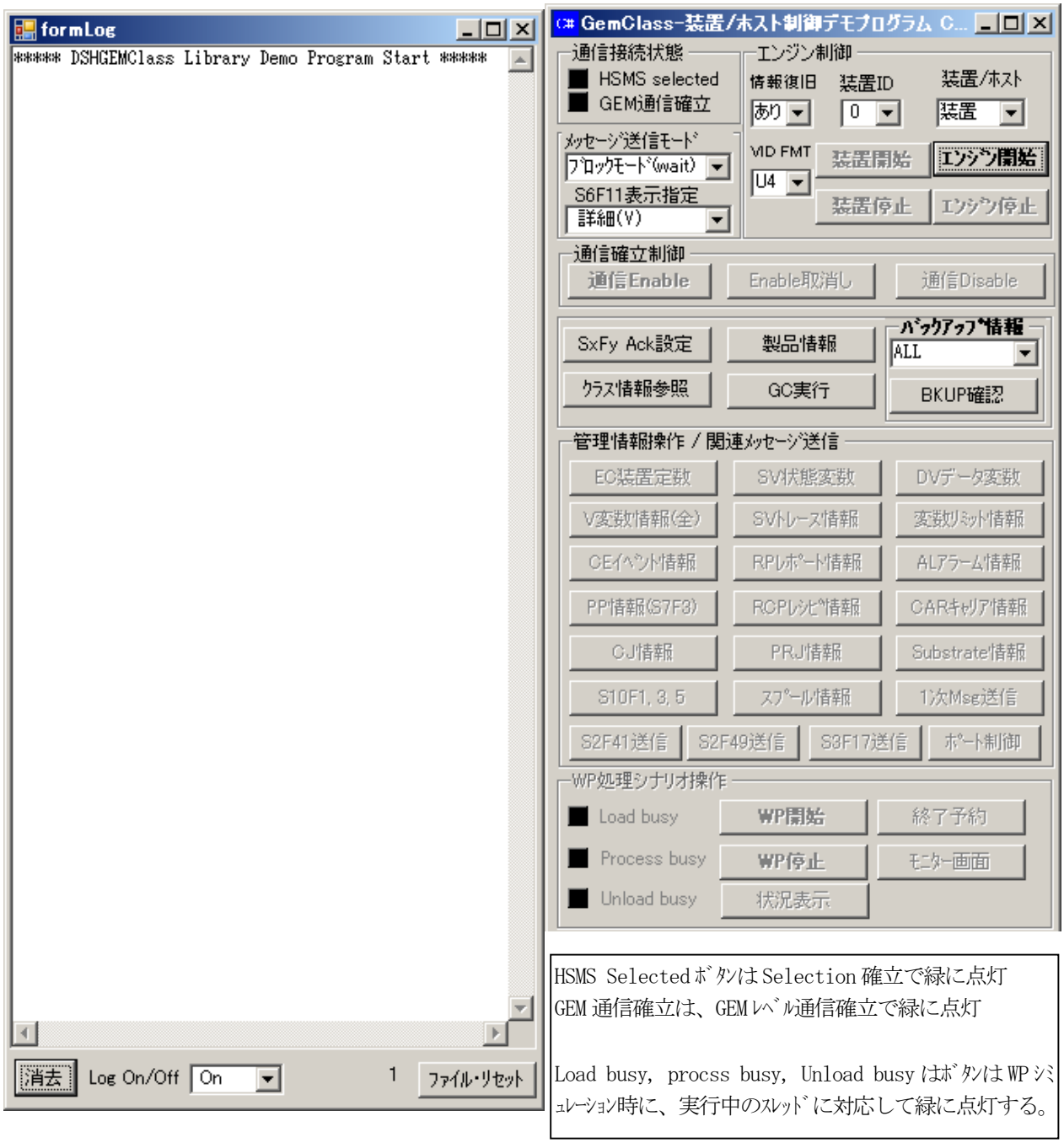
文書番号 DSHGEM-LIB-07-30305-00　「クラスライブラリ プログラミングの手引き」
３．５　１次メッセージ受信ポーリングの開始
４．受信メッセージの処理

３．画面と操作

３．１　メイン画面

GemCsDeme.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。)



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, procss 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

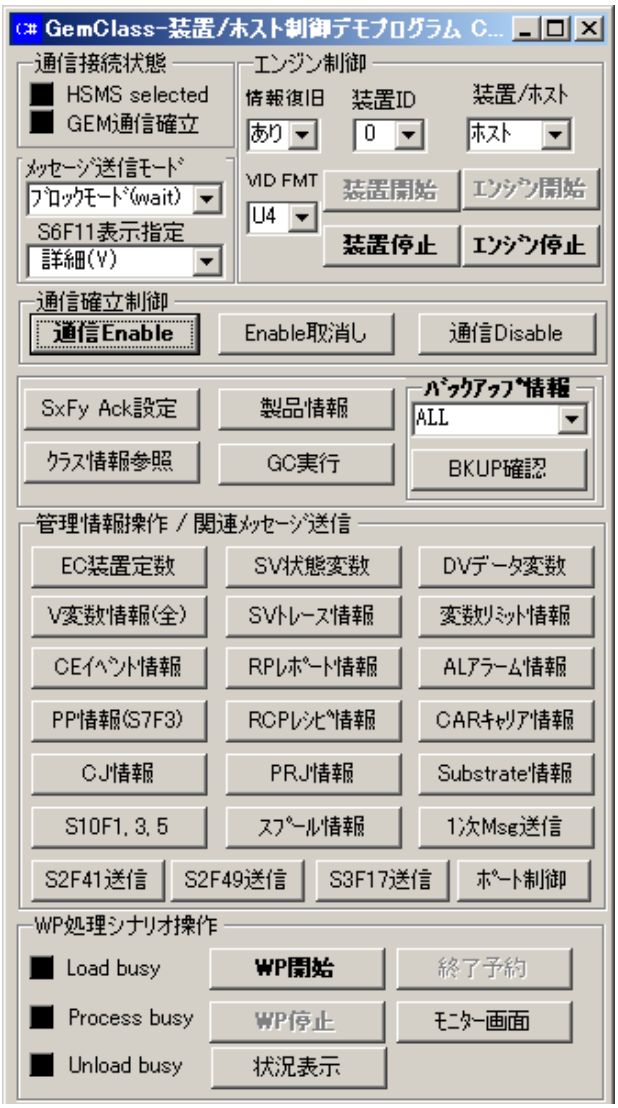
デモ・プログラムが起動から、ｴﾝｼﾞﾝ、装置開始まで流れは次のようになります。

操作(ボタン)	ﾌﾟﾛｸﾞﾗﾑの処理	画面の状態と操作
ﾃﾞﾓﾌﾟﾛｸﾞﾗﾑ起動	side.txt (装置/ﾎｽﾄのｻｲﾄﾞの保存ﾌｧｲﾙ) があれば、その内容を装置/ﾎｽﾄ選択ｺﾝﾎﾞﾎﾞｯｸｽに反映する。	**ｴﾝｼﾞﾝ開始**ﾎﾞﾀﾝが有効になり、他のﾎﾞﾀﾝは 1 部を除き、無効にする。 ｻｲﾄﾞを変えたい場合はｺﾝﾎﾞﾎﾞｯｸｽの選択を変える。 装置変数 ID などのﾌｫｰﾏｯﾄを U2 にしたい場合は、ｺﾝﾎﾞﾎﾞｯｸｽで選択しておく。
ｴﾝｼﾞﾝ開始 ｸﾘｯｸ	DSHGemClass の DshEngine ｸﾗｽによって開始する。 起動ﾊﾟﾗﾒｰﾀをｾｯﾄし、DshGemLib ｴﾝｼﾞﾝを開始する。(start ﾒｿｯﾄﾞ) 装置変数 ID(数値 ID)が U2 に選択されていれば、ｴﾝｼﾞﾝに設定する。	開始が正常終了ならば **ｴﾝｼﾞﾝ開始**ﾎﾞﾀﾝ　= 無効 **ｴﾝｼﾞﾝ停止**ﾎﾞﾀﾝ　= 有効 **装置開始**ﾎﾞﾀﾝ　= 有効にする。
装置開始 ｸﾘｯｸ	装置/ﾎｽﾄの選択されているｻｲﾄﾞの操作に必要な操作ﾎﾞﾀﾝを有効にする。 DshEquipment 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 ﾒｿｯﾄﾞを実行する。 装置開始が正常にできたら、 ・予約変数、ｲﾍﾞﾝﾄ ID の設定をする。 ・ﾕｰｻﾞが処理する受信 1 次ﾒｯｾｰｼﾞを登録する。 ・装置側の場合、S6F11, S5F1 の ID list をｺﾝﾎﾞﾎﾞｯｸｽに設定する。 ・DshEquipment ｸﾗｽに SECS-II 受信ﾒｯｾｰｼﾞのﾎﾟｰﾘﾝｸﾞを開始させる。 ・timer-1 を開始し、HSMS-SS 通信確立と GEM ﾚﾍﾞﾙでの通信確立を監視する。 ・GC を周期的に実行するための timer-4 を開始する。	装置またはﾎｽﾄの選択されているｻｲﾄﾞの画面になる。 **装置開始**ﾎﾞﾀﾝ　= 無効 **装置停止**ﾎﾞﾀﾝ　= 有効 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 ﾌﾟﾛﾄｺﾙ確立のための制御も開始されます。 HSMS-SS のｾﾚｸｼｮﾝ確立したら、HSMS-SS 確立を示す画面がﾎﾟｯﾌﾟｱｯﾌﾟされます。

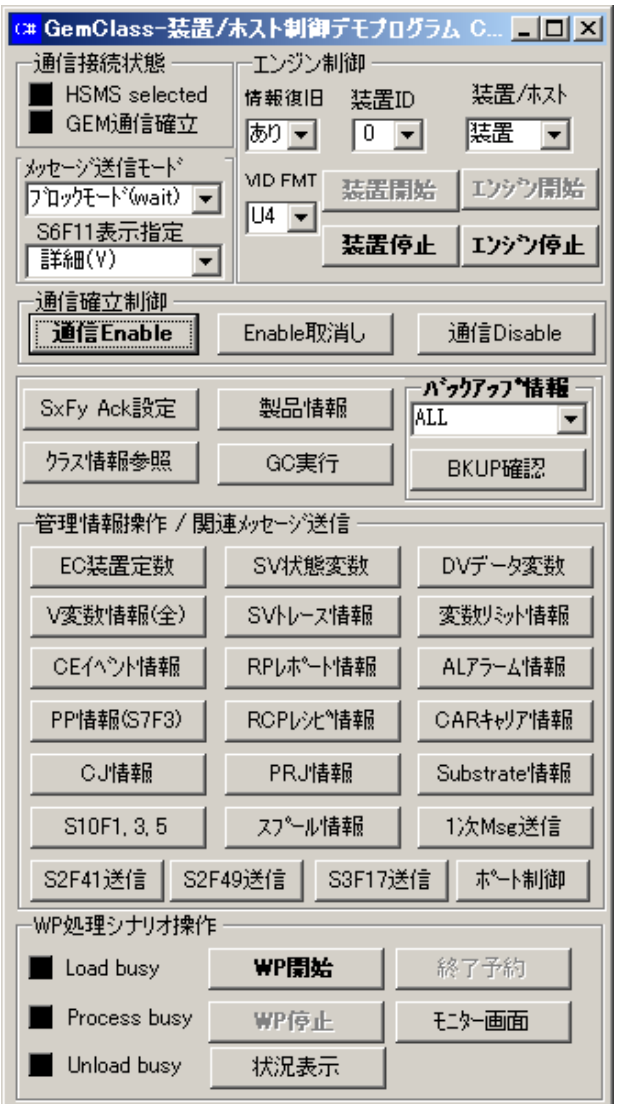
次ページに、エンジン、装置起動後のホスト、装置それぞれのサイドのメイン操作画面表示します。

使用できないボタンは無効になっています。

ホスト側


GemClass-装置/ホスト制御デモプログラム C...
通信接続状態
HSMS selected
GEM通信確立
エンジン制御
情報復旧
装置ID
装置/ホスト
あり
0
ホスト
メッセージ送信モード
ブロックモード(wait)
VID FMT
U4
装置開始
エンジン開始
S6F11表示指定
詳細(V)
装置停止
エンジン停止
通信確立制御
通信Enable
Enable取消し
通信Disable
SxFy Ack設定
製品情報
バックアップ情報
ALL
クラス情報参照
GC実行
BKUP確認
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
RPレポート情報
ALアラーム情報
PP情報(S7F3)
RCPレシピ情報
CARキャリア情報
CJ情報
PRJ情報
Substrate情報
S10F1, 3, 5
スプール情報
1次Msg送信
S2F41送信
S2F49送信
S3F17送信
ポート制御
WP処理シナリオ操作
Load busy
WP開始
終了予約
Process busy
WP停止
モニター画面
Unload busy
状況表示

装置側


GemClass-装置/ホスト制御デモプログラム C...
通信接続状態
HSMS selected
GEM通信確立
エンジン制御
情報復旧
装置ID
装置/ホスト
あり
0
装置
メッセージ送信モード
ブロックモード(wait)
VID FMT
U4
装置開始
エンジン開始
S6F11表示指定
詳細(V)
装置停止
エンジン停止
通信確立制御
通信Enable
Enable取消し
通信Disable
SxFy Ack設定
製品情報
バックアップ情報
ALL
クラス情報参照
GC実行
BKUP確認
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
RPレポート情報
ALアラーム情報
PP情報(S7F3)
RCPレシピ情報
CARキャリア情報
CJ情報
PRJ情報
Substrate情報
S10F1, 3, 5
スプール情報
1次Msg送信
S2F41送信
S2F49送信
S3F17送信
ポート制御
WP処理シナリオ操作
Load busy
WP開始
終了予約
Process busy
WP停止
モニター画面
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。

３．１．１．３　停止操作

制御の停止は、**装置停止**ボタン、**ｴﾝｼﾞﾝ停止**ボタンを順にクリックします。

全て停止したら、**ｴﾝｼﾞﾝ開始**ボタンが有効になり、**装置開始**、**装置停止**、**ｴﾝｼﾞﾝ停止**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、装置開始した後に実行できる操作です。

３．１．２．１　通信 Enable / Enable 取消し / 通信 Disable の操作

GEM レベルでの通信確立のための操作です。
DshEquipment クラスを使ってこれらの機能を実行します。

enable メソッドによって、S1F13 の実際のやり取りはエンジンが処理します。

ボタン	ﾒｿｯﾄﾞ	callback のｸﾗｽ	動作
通信 Enable	enable()	DshCallback.callback_enable	S1F13 の送信で通信接続手続きを行う。 終了したら callback で通知される。
Enable 取消し	cancel()	（なし）	enable()を取りやめる。
通信 Disable	disable()	DshCallback.callback_disable	通信確立状態を未確立状態にする。 終了したら callback で通知される。

enable()結果のコールバック関数が呼び出され、結果が正常であれば通信確立したことになります。

通信確立後、相手装置との間で、アプリケーションレベルのメッセージの送受信が可能になります。

３．１．２．２　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。

DshEquipment 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) error backup file = xxxxx
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)

３．１．２．３　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。

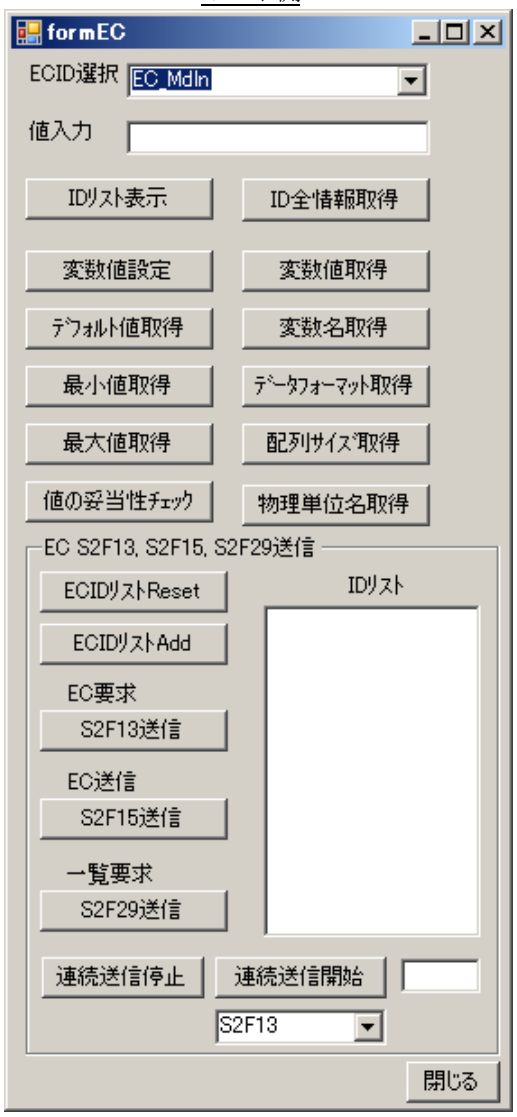
３．２　変数関連情報操作画面

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。

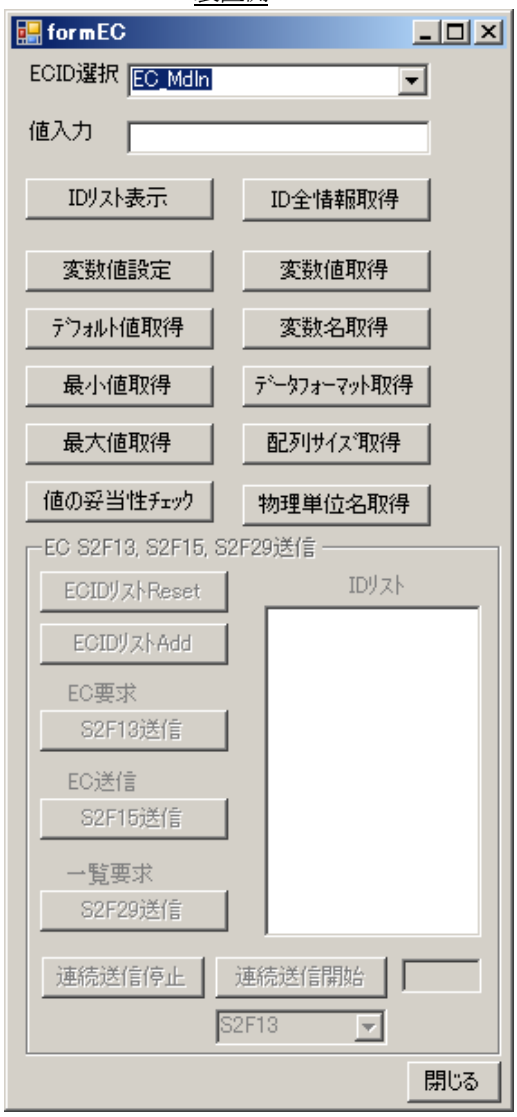
３．２．１ EC 装置定数操作画面

装置定数 EC 操作画面は次のようになります。装置の画面では、S2F13, S2F15, S2F29 送信部を使用しません。
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。

ホスト側



装置側



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	ﾒｿｯﾄﾞ
情報ｱｸｾｽ関連	DshEC	get_id_list(), get(), set()他

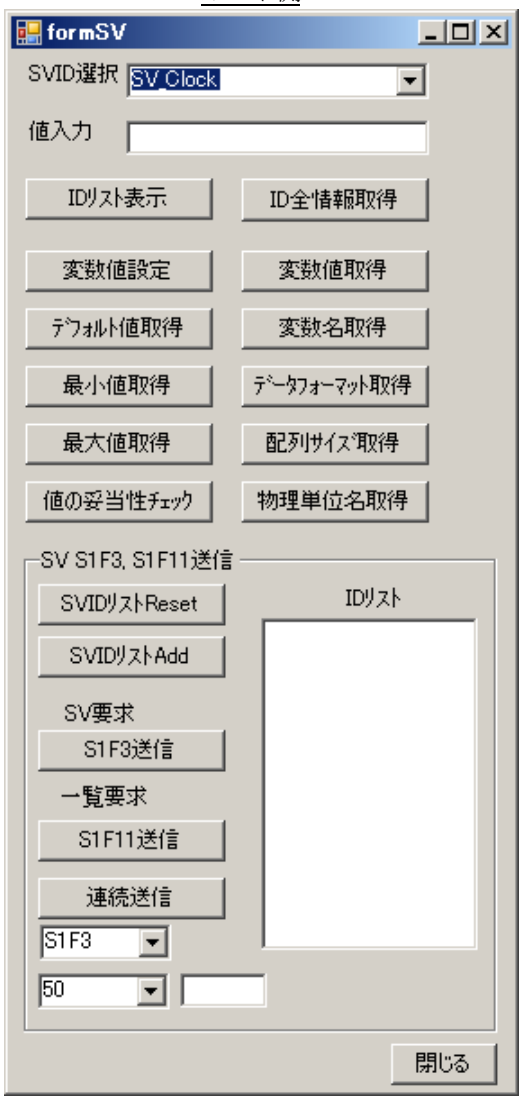
（2）メッセージ送信関連

ﾎﾞﾀﾝ	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
S2F13 送信	DshS2F13Send	send(), send_wait()	DshV_ValueList	DshCallback.callback_s2f13
S2F15 送信	DshS2F15Send	send(), send_wait()	-	DshCallback.callback_s2f15
S2F29 送信	DshS2F29Send	send(), send_wait()	DshEC_NameList	DshCallback.callback_s2f29

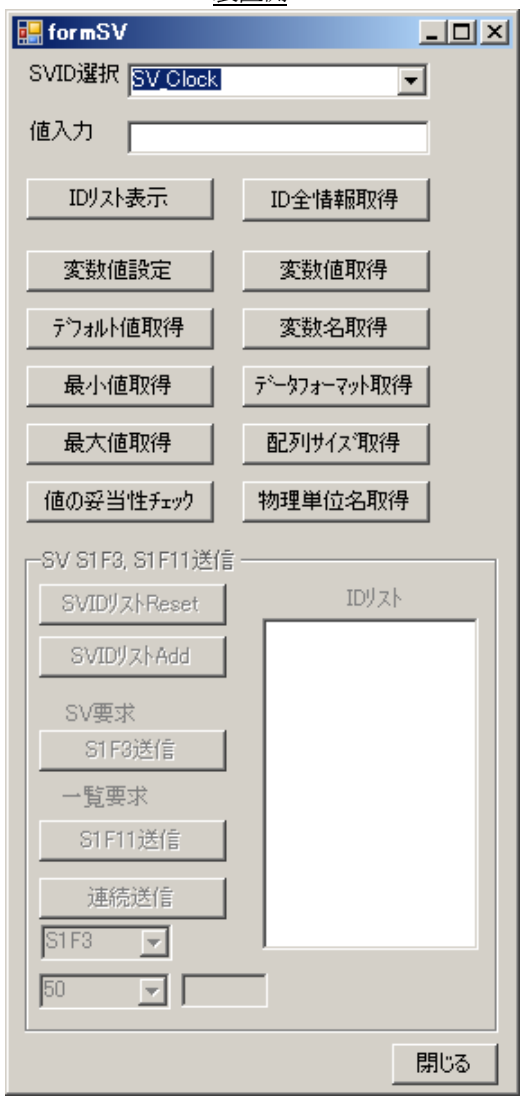
３．２．２ SV 装置状態変数操作画面

装置状態変数 SV 操作画面は次のようになります。装置の画面では、S1F3, S1F11 送信部を使用しません。
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。

ホスト側



装置側



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	ﾒｿｯﾄﾞ
情報ｱｸｾｽ関連	DshSV	get_id_list(), get(), set()他

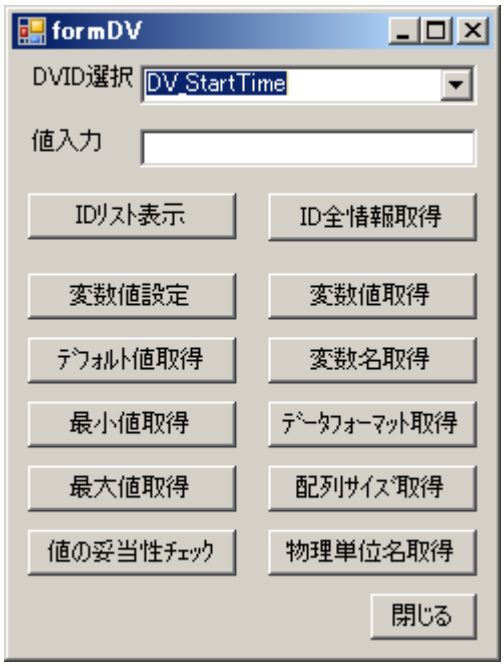
（2）メッセージ送信関連

ﾎﾞﾀﾝ	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
S1F3 送信	DshS1F3Send	send(), send_wait()	DshV_ValueList	DshCallback.callback_s1f3
S1F11 送信	DshS1F11Send	send(), send_wait()	DshSV_NameList	DshCallback.callback_s1f11

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。ホスト、装置とも同じ画面です。
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。

HOST 側 & 装置側



以下のクラスを使用します。

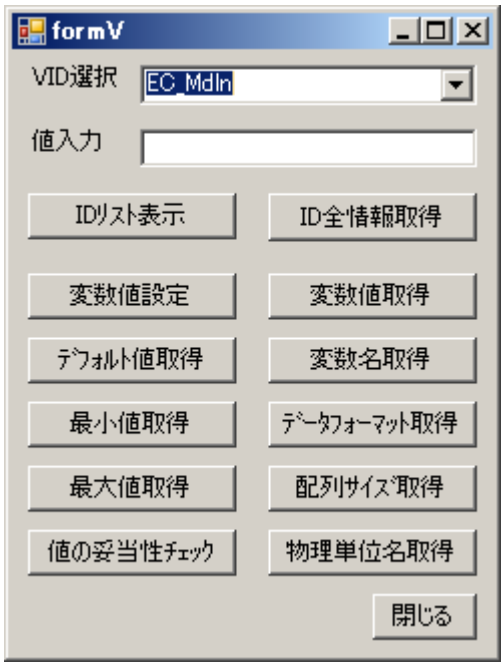
（1）情報アクセス関連

ボタン	ｸﾗｽ	ﾒｿｯﾄﾞ
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。ホスト、装置とも同じ画面です。
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。

HOST 側 & 装置側



以下のクラスを使用します。

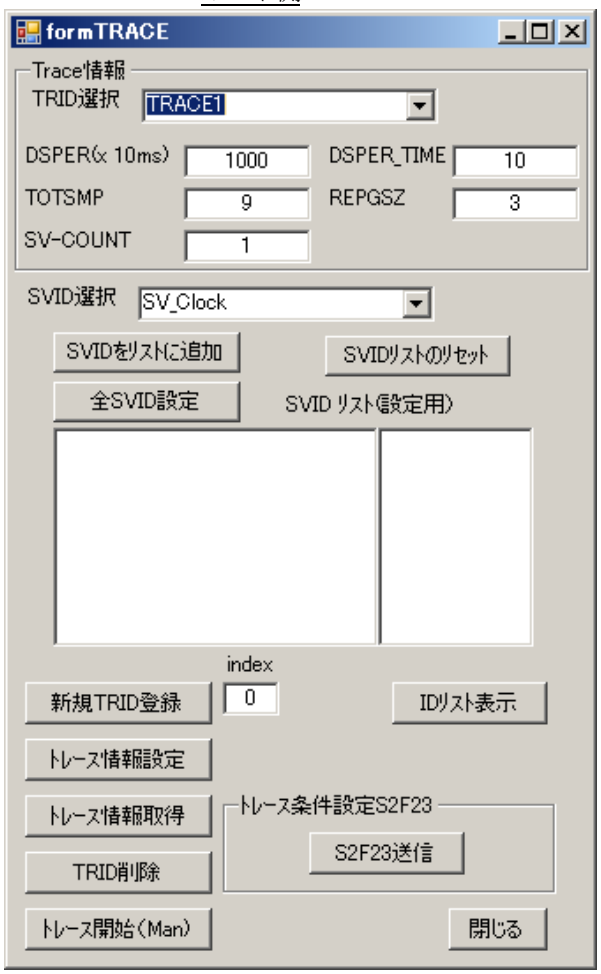
（1）情報アクセス関連

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

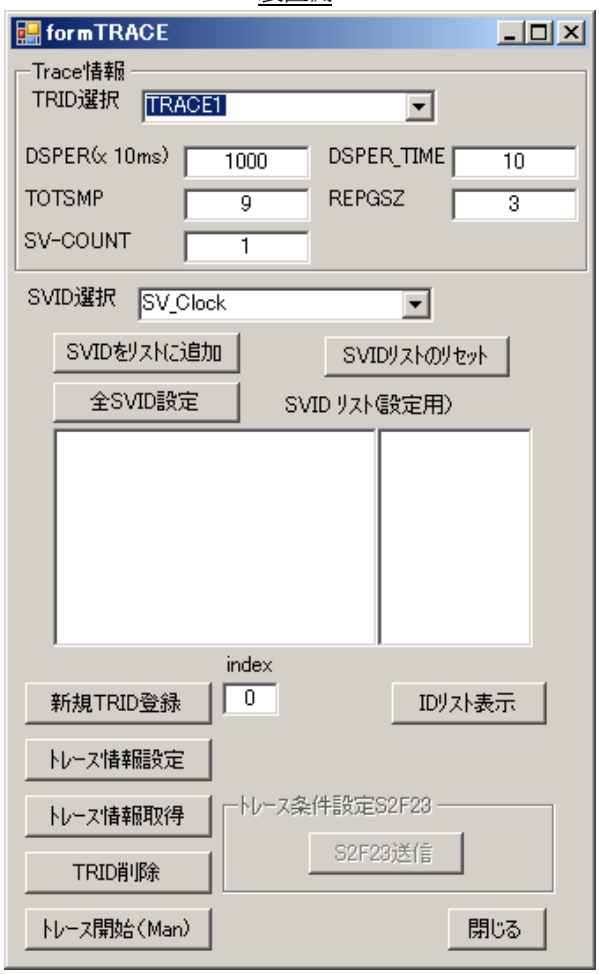
３．２．５　SV トレース操作画面

装置状態変数 SV 操作画面は次のようになります。装置の画面では、S2F23 送信部を使用しません。
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。

ホスト側



装置側



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	ﾒｿｯﾄﾞ
情報ｱｸｾｽ関連	DshSV	get_id_list(), get(), set()他
	DshTrace	get(), set()

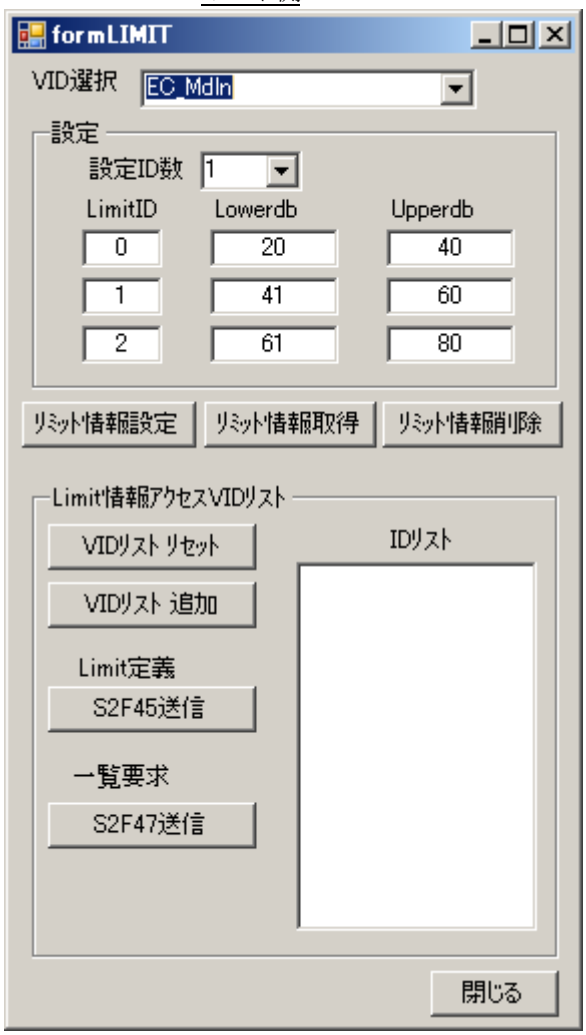
（2）メッセージ送信関連

ﾎﾞﾀﾝ	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
S2F23 送信	DshS2F23Send	send(), send_wait()	-	DshCallback.callback_s2f23

３．２．６　変数リミット操作画面

変数リミット操作画面は次のようになります。装置の画面では、S2F45, S2F47 送信部を使用しません。
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。

ホスト側



装置側

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	ﾒｿｯﾄﾞ
情報ｱｸｾｽ関連	DshV	get_id_list()
	DshLimit	set_id(), clear(), set(), get(), **add_limitid, delete()**
	DshLimitList	**add_limit(),**

（2）メッセージ送信関連

ﾎﾞﾀﾝ	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
S2F45 送信	DshS2F45Send	send(), send_wait()	-	DshCallback.callback_s2f45
S2F47 送信	DshS2F47Send	send(), send_wait()	DshS2F48LimitRspList	DshCallback.callback_s2f47

３．３　CE 収集イベントと RP レポート操作、S6F11 送信画面

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。

ホストでは、CE 関連で、S2F35. S2F37, S6F15 と RP 関連で, S2F33, S6F19 の送信があります。

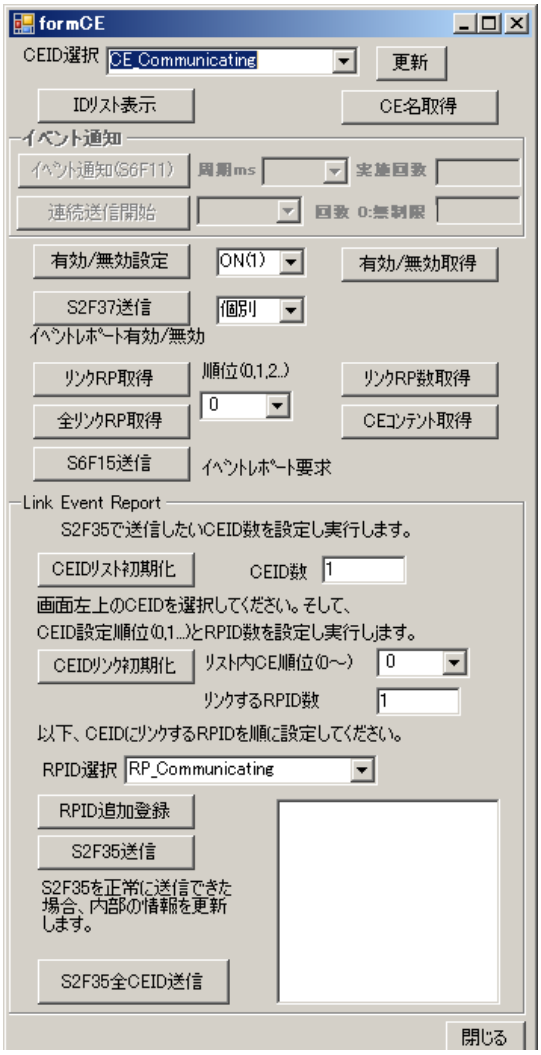
装置では、CE 関連だけで、S6F11 送信があります。

３．３．１　CE 収集イベント情報操作画面

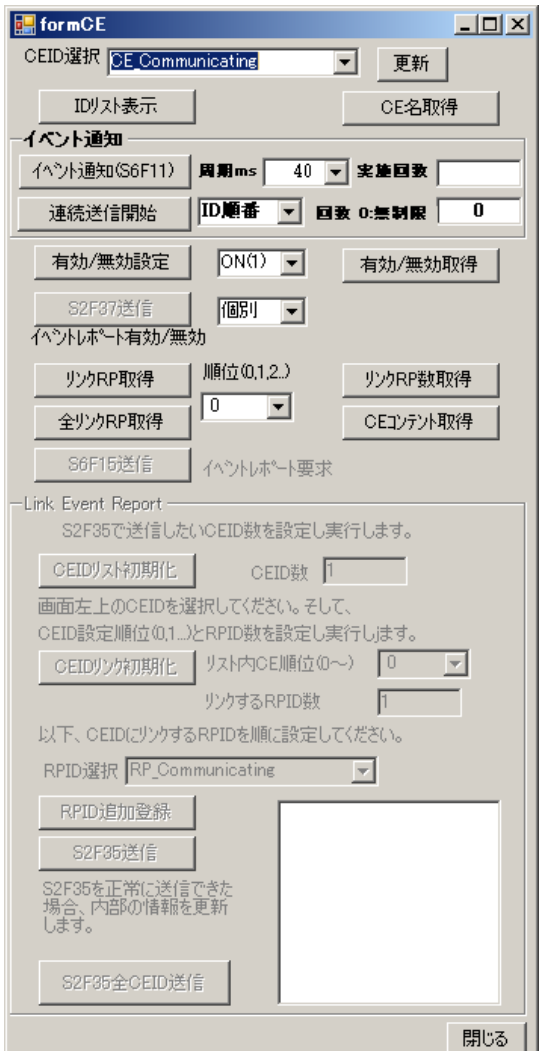
それぞれのボタンの機能は、ボタンの表示名のとおりです。ホスト側から S2F35．S2F37，S6F15 の送信を行い、装置側からは S6F11 の送信を行います。

操作結果は、LOG 画面に表示されます。

ホスト側



装置側



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	ﾒｿｯﾄﾞ
情報ｱｸｾｽ関連	DshCE	get_id_list(), set_ceid(), set_enabled(), get_content()他

（2）メッセージ送信関連

ﾎﾞﾀﾝ	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
S2F35 送信	DshS2F35Send	send(), send_wait()	-	DshCallback.callback_s2f35
S2F37 送信	DshS2F37Send	send(), send_wait()	-	DshCallback.callback_s2f37
S6F11 送信	DshS6F11Send	send(), send_wait()		DshCallback.callback_s6f11
S6F15 送信	DshS5F15Send	send(), send_wait()	DshCeContent	DshCallback.callback_s6f15

３．３．１．１　S6F11 収集イベントの送信操作

装置側の選択のときに操作できます。

装置開始時に、システムに登録されている収集イベント名は、ceid のコンボボックスのリストに登録され、表示されます。

送信は、次のボタン操作で行うことができます。

ボタン	ｸﾗｽ	ﾒｿｯﾄﾞ	callback のｸﾗｽ	動作
ｲﾍﾞﾝﾄ通知(S6F11)	DshS6F11Send	send() send_wait()	DshCallback.callback_s6f11	選択された CEID の S6F11 を送信する。
連続送信開始 連続送信停止	同上	同上	同上	連続的に S6F11 を送信します。送信する CEID は、ｺﾝﾎﾞﾎﾞｯｸｽの選択によります。固定の場合は、そのとき選択されている CEID のみ、順番の場合は、そのとき選択されている CEID から CEID ﾘｽﾄの登録にしたがって順次送信します。 連続送信停止で停止させます。 回数の設定もでき、送信された回数ｶｳﾝﾄも表示される。

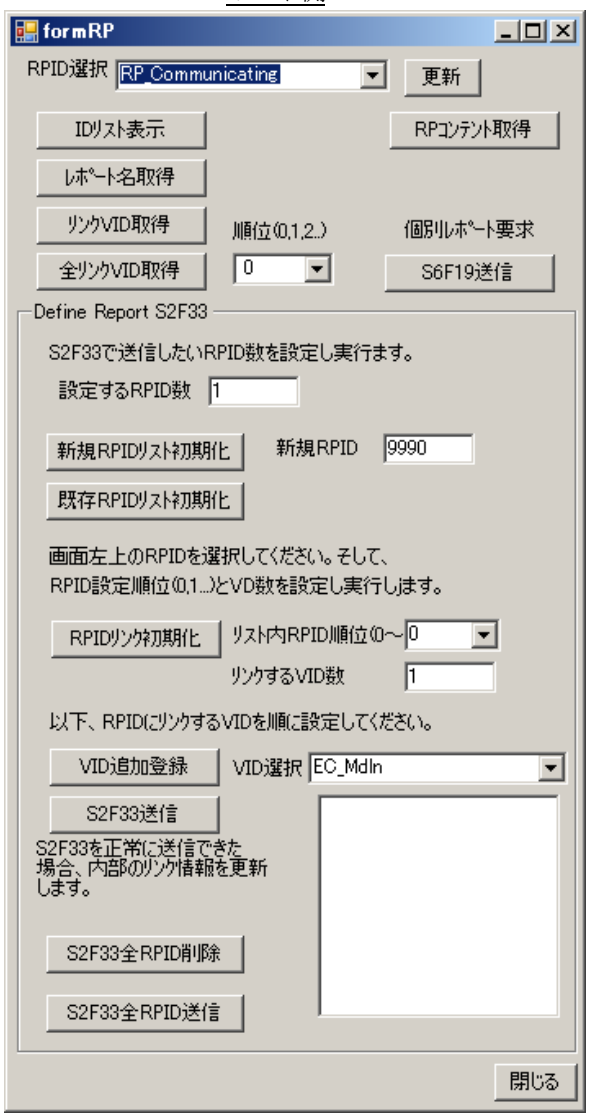
ブロックモード、非ブロックモードのどちらを使用するかは、**メインフォームのメッセージ送信モード選択用のコンボボックスの指定**に従って行います。

ブロックモードでは、send_wait()メソッド、非ブロックモードでは、send()または send_S6F11()メソッドを使用します。

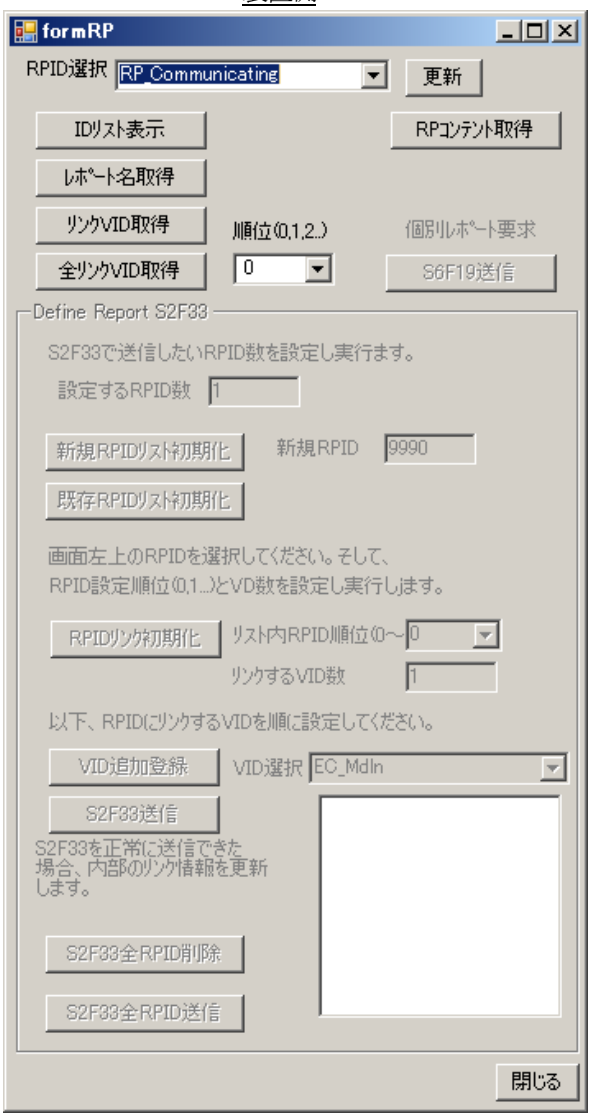
３．３．２　RP レポート情報操作画面

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。

ホスト側



装置側



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	ﾒｿｯﾄﾞ
情報ｱｸｾｽ関連	DshReport	get_id_list(), set_rpid(), get_content()他

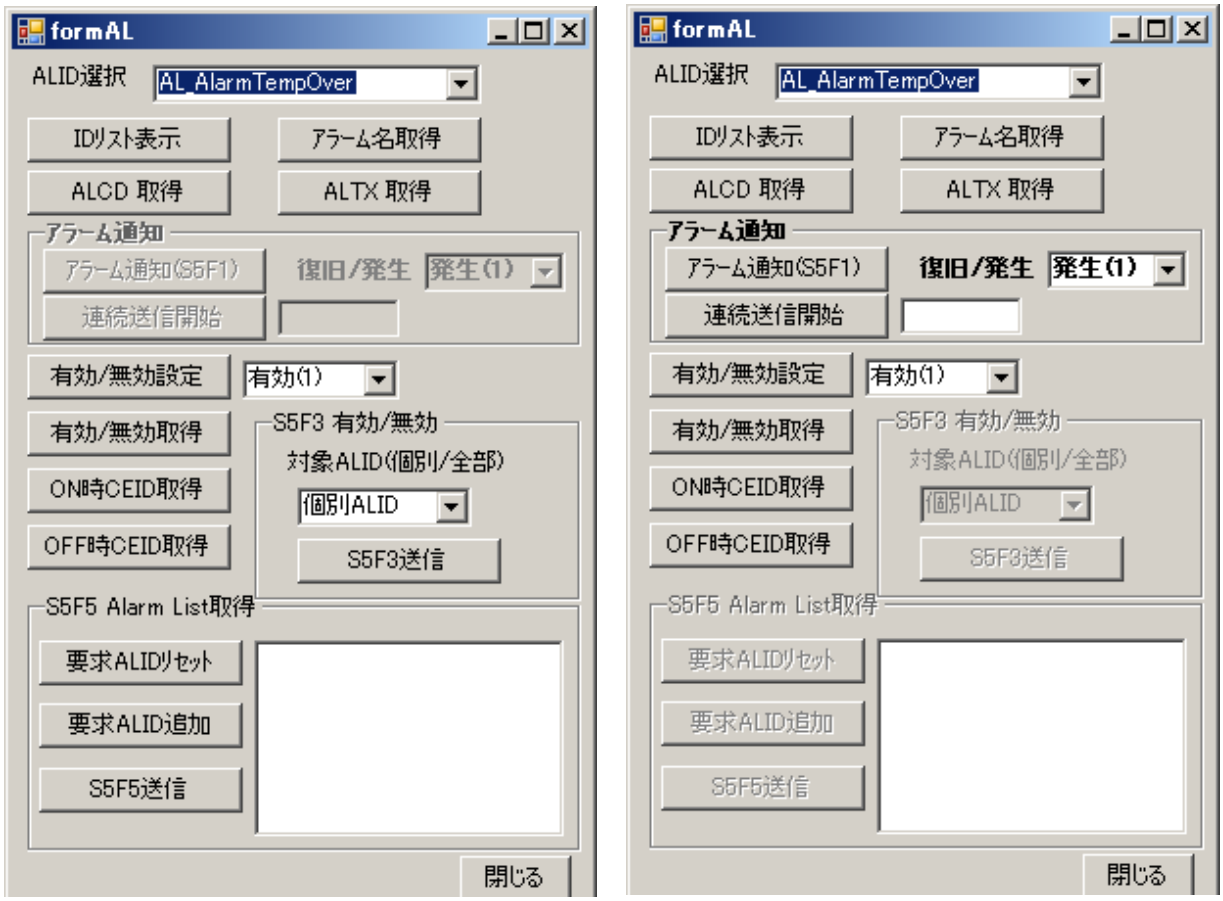
（2）メッセージ送信関連

ﾎﾞﾀﾝ	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
S2F33 送信	DshS2F33Send	send(), send_wait()	-	DshCallback.callback_s2f33
S6F19 送信	DshS6F19Send	send(), send_wait()	DshRpContent	DshCallback.callback_s6f19

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。
ホストでは、S5F3. S5F5 の送信があります。
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。

ホスト側　　装置側



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	ﾒｿｯﾄﾞ
情報ｱｸｾｽ関連	DshAlarm	get_id_list(), set_alid(), set_enabled()他

（2）メッセージ送信関連

ﾎﾞﾀﾝ	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
S5F1 送信	DshS5F1Send	send(), send_wait()	-	DshCallback.callback_s5f1
S5F3 送信	DshS5F3Send	send(), send_wait()	-	DshCallback.callback_s5f3
S5F5 送信	DshS5F5Send	send(), send_wait()	DshAlarmList	DshCallback.callback_s5f5

３．４．１ S5F1 アラームの送信操作

装置側の選択のときに操作できます。

装置開始時に、システムに登録されているアラーム名は、alid のコンボボックスのリストに登録され、表示されます。

送信は、次のボタン操作で行うことができます。

ボタン	ｸﾗｽ	ﾒｿｯﾄﾞ	callback のｸﾗｽ	動作
ｱﾗｰﾑ通知(S5F1)	DshS5F1Send	send() send_wait()	DshCallback.callback_s5f1	選択された ALID の S5F1 を送信する。
連続送信開始 連続送信停止	同上	同上	同上	連続的に S5F1 を送信します。送信する ALID はそのとき選択されている ALID から ALID ﾘｽﾄの登録にしたがって順次送信します。 連続送信停止で停止させます。 送信された回数ｶｳﾝﾄも表示される。

ブロックモード、非ブロックモードのどちらを使用するかは、メインフォームのメッセージ送信モード選択用のコンボボックスの指定に従って行います。

ブロックモードでは、send_wait()メソッド、非ブロックモードでは、send()または send_S5F1 ()メソッドを使用します。

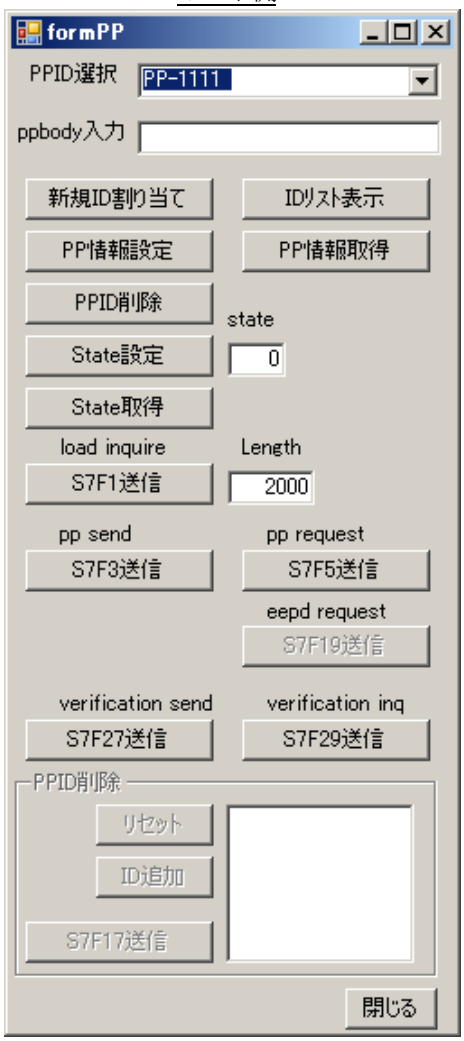
３．５　PP プロセス・プログラムと RCP レシピ情報操作画面

DSHGemLib エンジンでは、プロセス・プログラムとレシピ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。

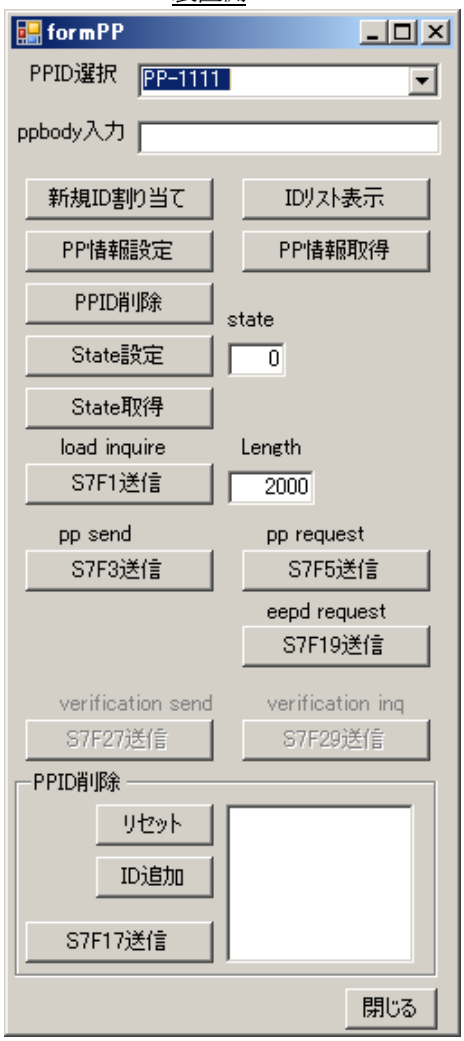
３．５．１　PP プロセス・プログラム操作画面

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。

ホスト側



装置側



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	ﾒｿｯﾄﾞ
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

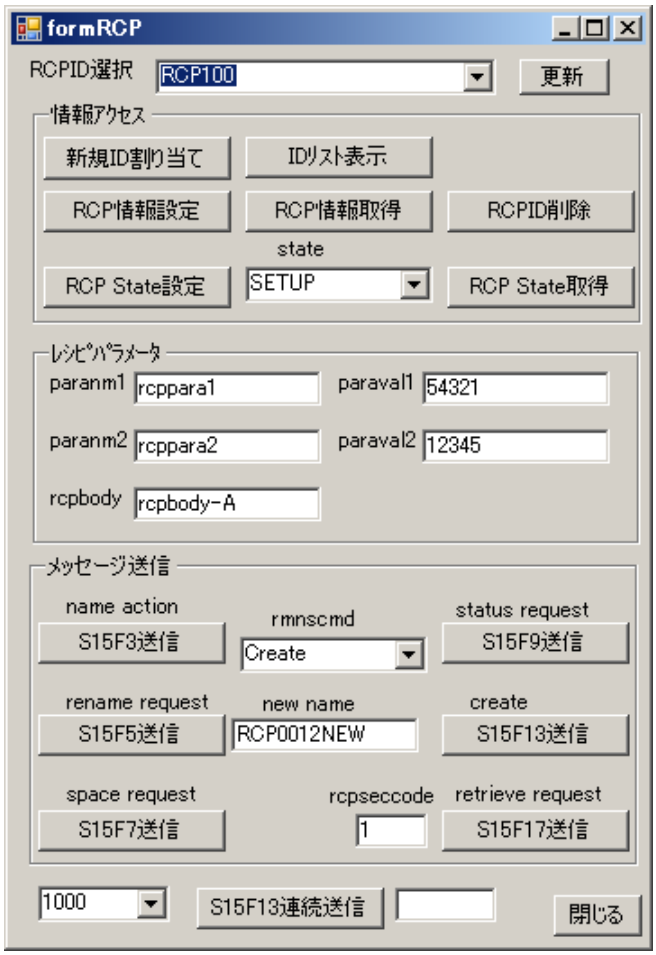
ボタン	情報クラス	MSG 送信クラス	Msg 受信クラス	送信完了 callback クラス
S7F1 送信	DshS7F1Send	send(), send_wait()	-	DshCallback.callback_s7f1
S7F3 送信	DshS7F3Send	send(), send_wait()	-	DshCallback.callback_s7f3
S7F5 送信	DshS7F5Send	send(), send_wait()	DshPP	DshCallback.callback_s7f5
S7F17 送信	DshS7F17Send	send(), send_wait()	DshStrList	DshCallback.callback_s7f17
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。ホストと装置は同じ画面です。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。

HOST 側 & 装置側



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	ﾒｿｯﾄﾞ
情報ｱｸｾｽ関連	DshRecipe	get_id_list(), alloc_id(), set_id(), set(), get(), delete()他

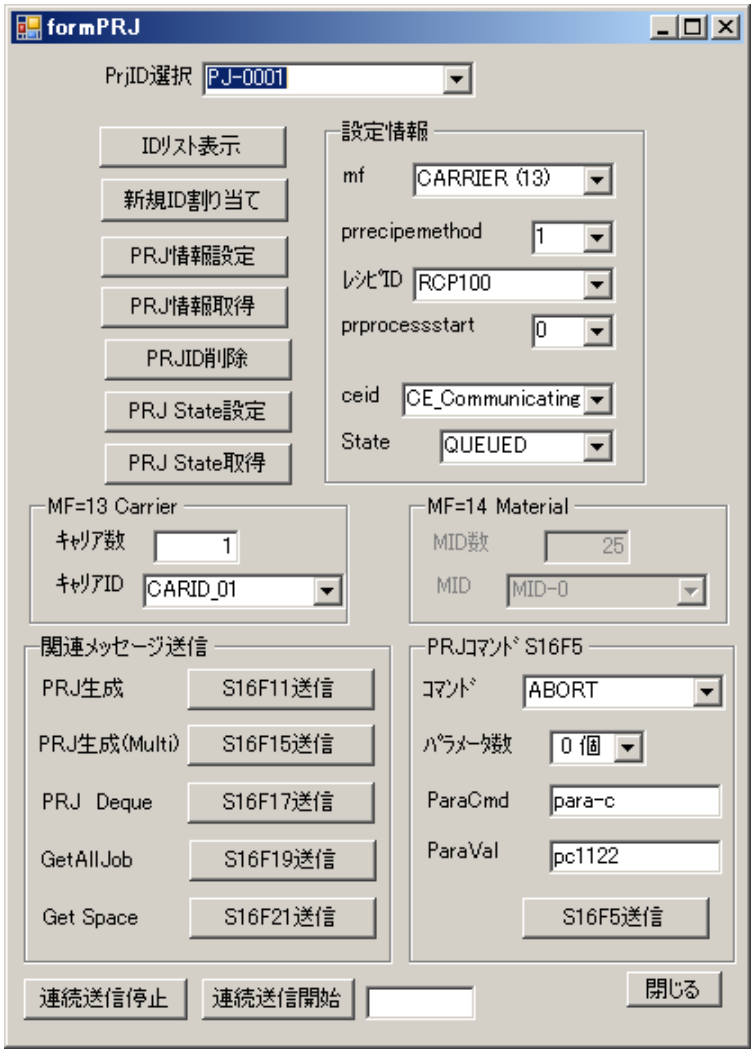
（2）メッセージ送信関連

ﾎﾞﾀﾝ	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

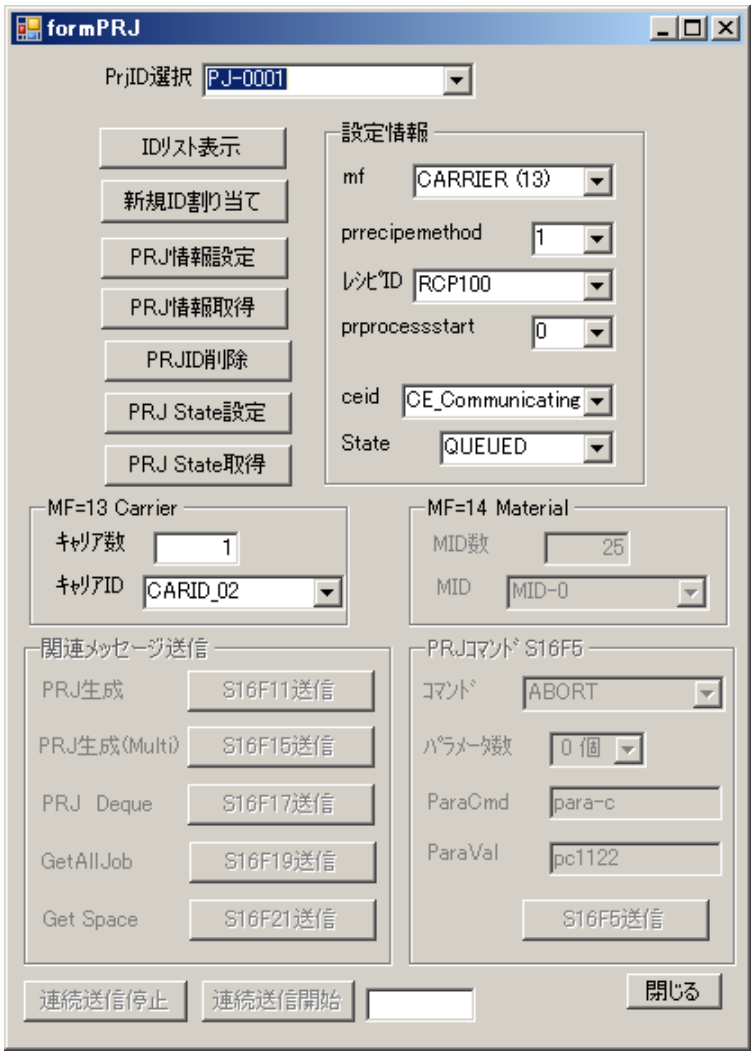
３．６　PRJ プロセス・ジョブ操作画面

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。
また、関連するメッセージの送信もあります。
但し、装置側には、通信メッセージの送信がありません。
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。　装置側画面は次ページにあります。

HOST 側



装置側



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	DshPrj	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
S16F5 送信	DshS16F5Send	send(), send_wait()	DshS16Rsp	DshCallback.callback_s16f5
S16F11 送信	DshS16F11Send	send(), send_wait()	DshS16Rsp	DshCallback.callback_s16f11
S16F15 送信	DshS16F15Send	send(), send_wait()	DshS16MultiPrjRsp	DshCallback.callback_s16f15
S16F17 送信	DshS16F17Send	send(), send_wait()	DshS16MultiPrjRsp	DshCallback.callback_s16f17
S16F19 送信	DshS16F19Send	send(), send_wait()	DshPrjStateList	DshCallback.callback_s16f19
S16F21 送信	DshS16F21Send	send(), send_wait()	-	DshCallback.callback_s16f21

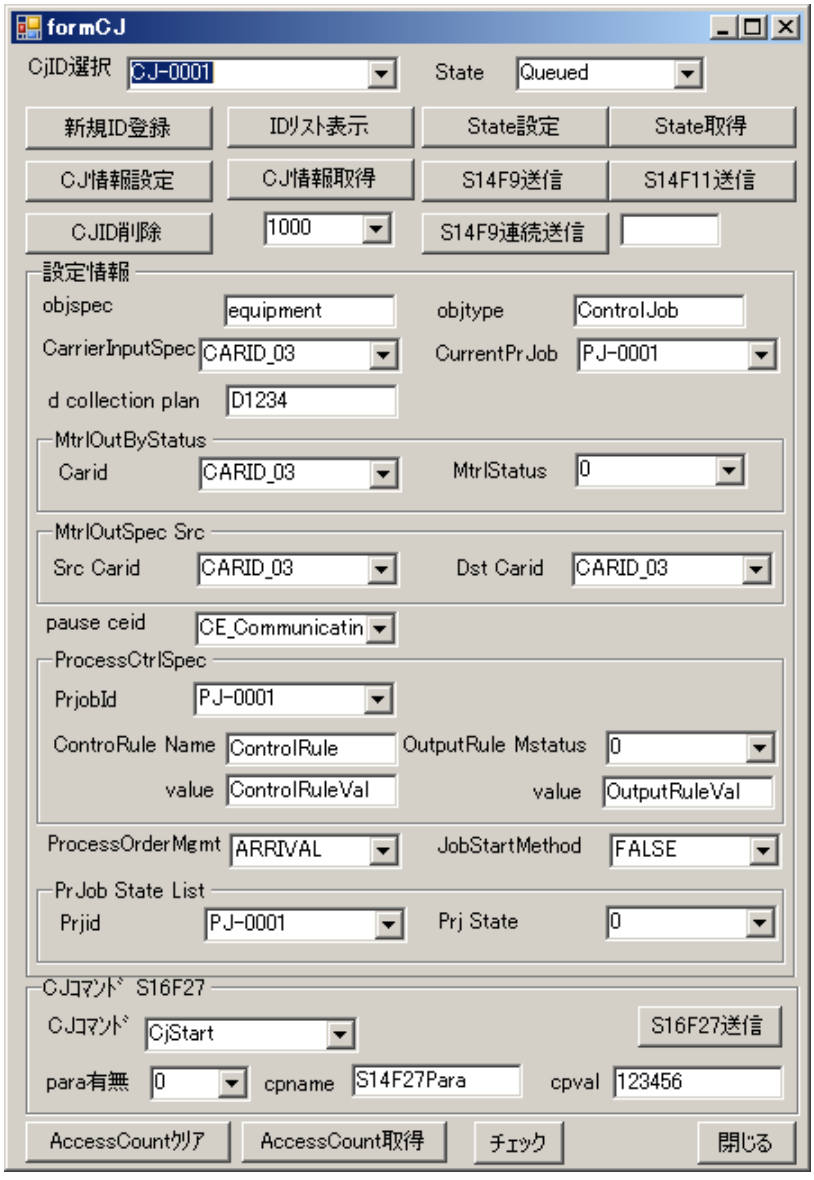
３．７　CJ コントロール・ジョブ操作画面

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。

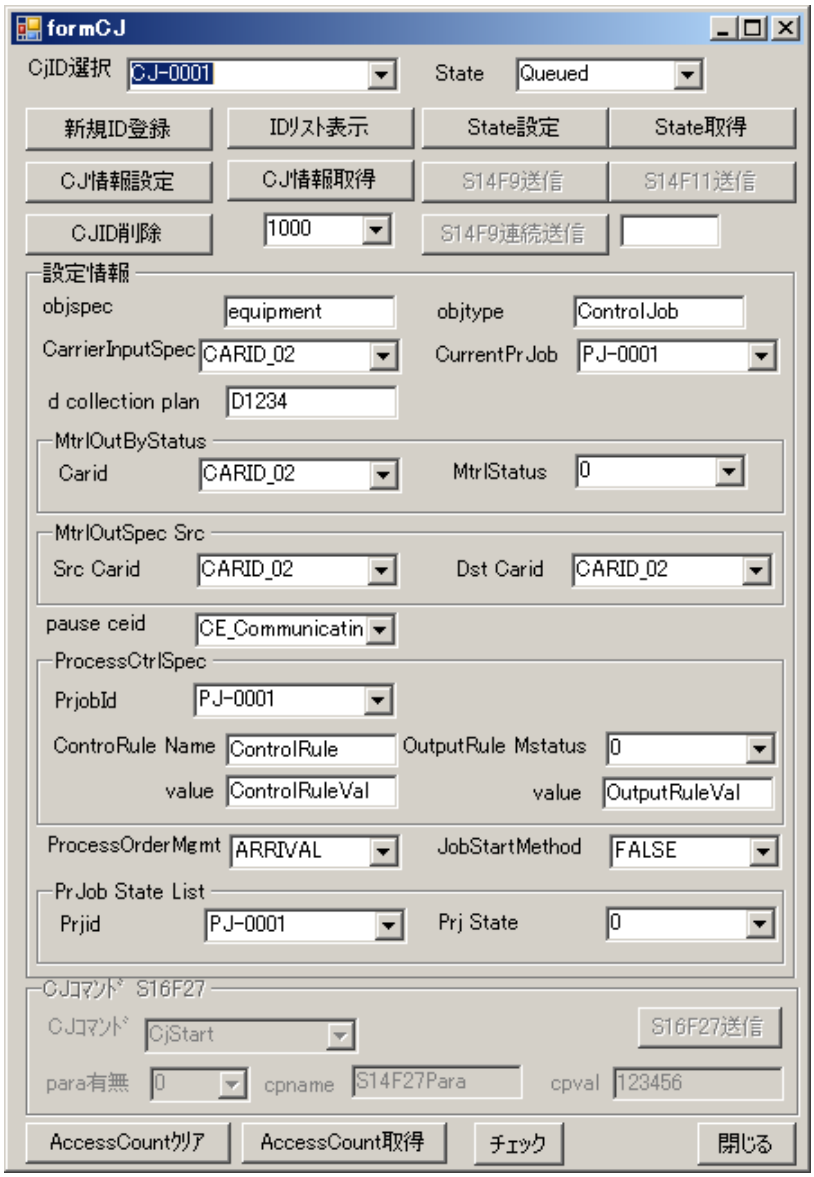
また、関連するメッセージの送信もあります。　但し、装置側には、通信メッセージの送信がありません。

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。　装置側画面は次ページにあります。

HOST 側



装置側



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	ﾒｿｯﾄﾞ
情報ｱｸｾｽ関連	DshCj	get_id_list(), alloc_id(), set_id(), set(), get(), delete()他

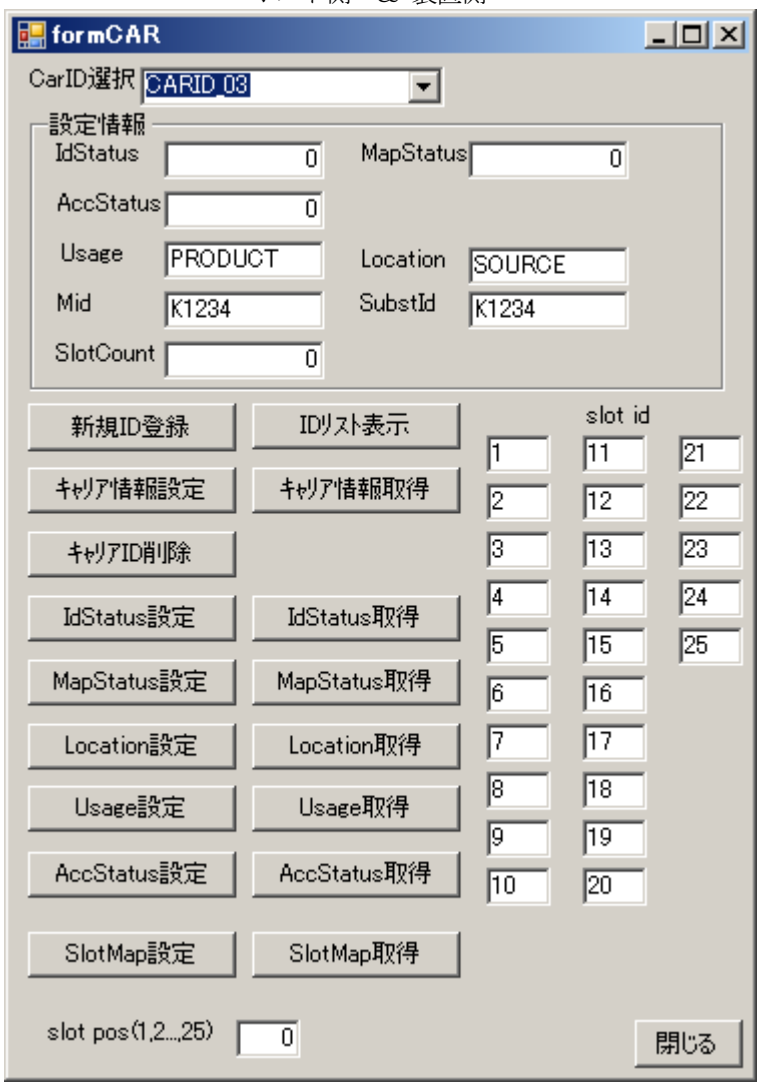
（2）メッセージ送信関連

ﾎﾞﾀﾝ	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
S14F9 送信	DshS14F9Send	send(), send_wait()	DshS14Rsp	DshCallback.callback_s14f9
S14F11 送信	DshS14F11Send	send(), send_wait()	DshS14Rsp	DshCallback.callback_s14f11

３．８　Carrier キャリア、Substrate 基板操作画面

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。

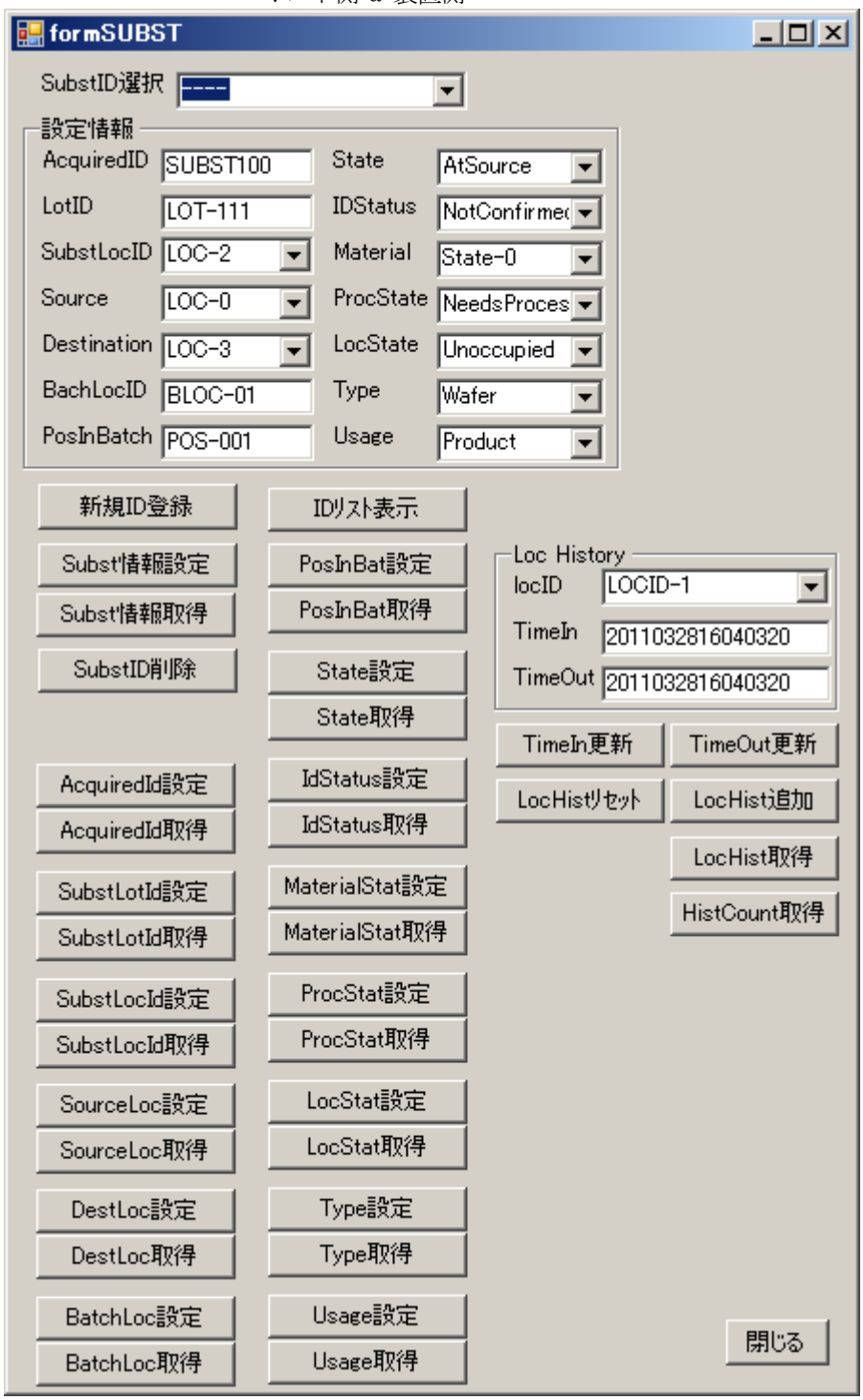
ホスト側　＆　装置側



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。

ホスト側 & 装置側

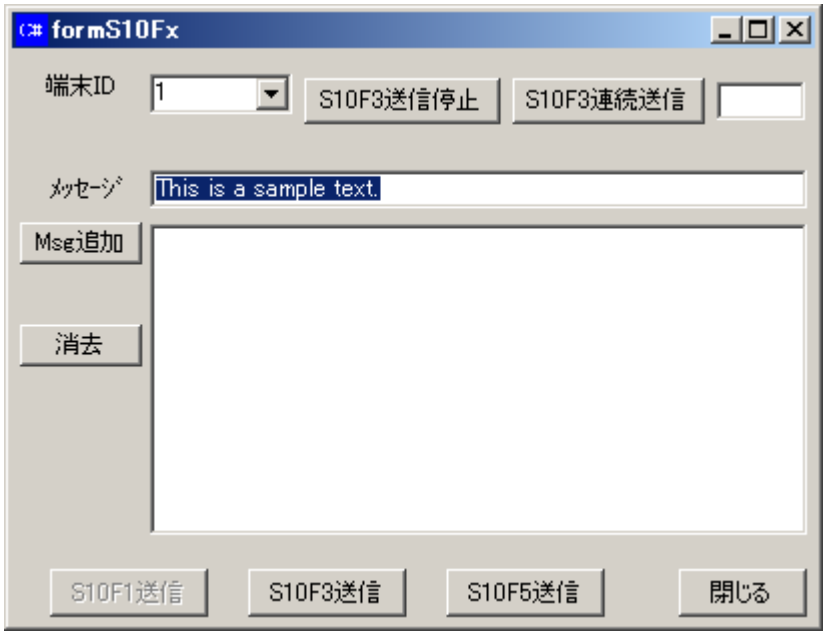


3．10　端末メッセージ送信画面

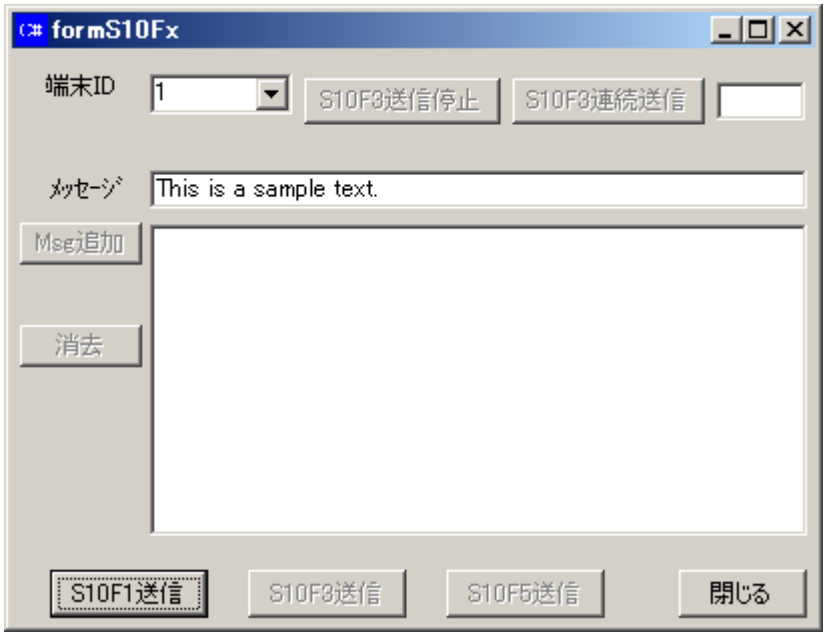
装置側から S10F1、ホスト側から S10F3, S10F5 メッセージを送信するための画面です。
以下のクラスを使用します。

ﾒｯｾｰｼﾞ	情報ｸﾗｽ	Msg 送信ｸﾗｽ	送信完了 callback ｸﾗｽ
S10F1	DshTermMsg	DshS10F1Send	DshCallback.callback_s10f1
S10F3	DshTermMsg	DshS10F3Send	DshCallback.callback_s10f3
S10F5	DshTermMultiMsg	DshS10F5Send	DshCallback.callback_s10f5

ホスト側



装置側



３．11　ホスト・リモート・コマンドメッセージ送信画面

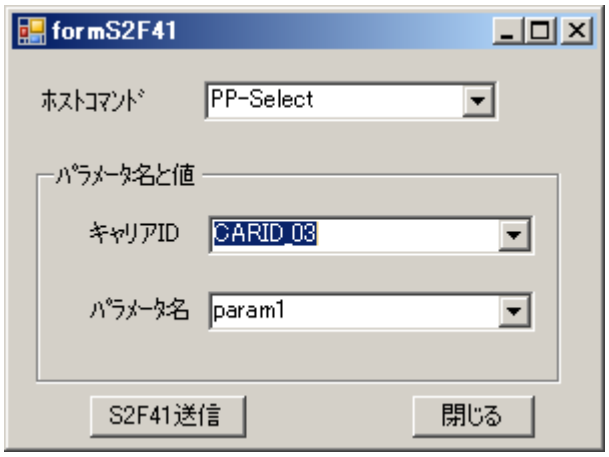
S2F41 ホスト・コマンドと S2F49 拡張リモート・コマンドの２つの画面があります。
ホスト側からだけの送信メッセージであるため、装置側にはありません。

３．11．１　S2F41　ホスト・コマンドの送信操作画面

S2F41 に含めたいコマンド、キャリア ID を選択し、パラメータを設定し、それをホストに送信します。
次のクラスを使用します。

ﾒｯｾｰｼﾞ	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
S2F41	DshRCmd	DshS2F41Send	DshRCmdRsp	DshCallback.callback_s2f41
	DshCar			

操作画面



３．11．２　S2F49　拡張リモート・コマンドの送信操作画面

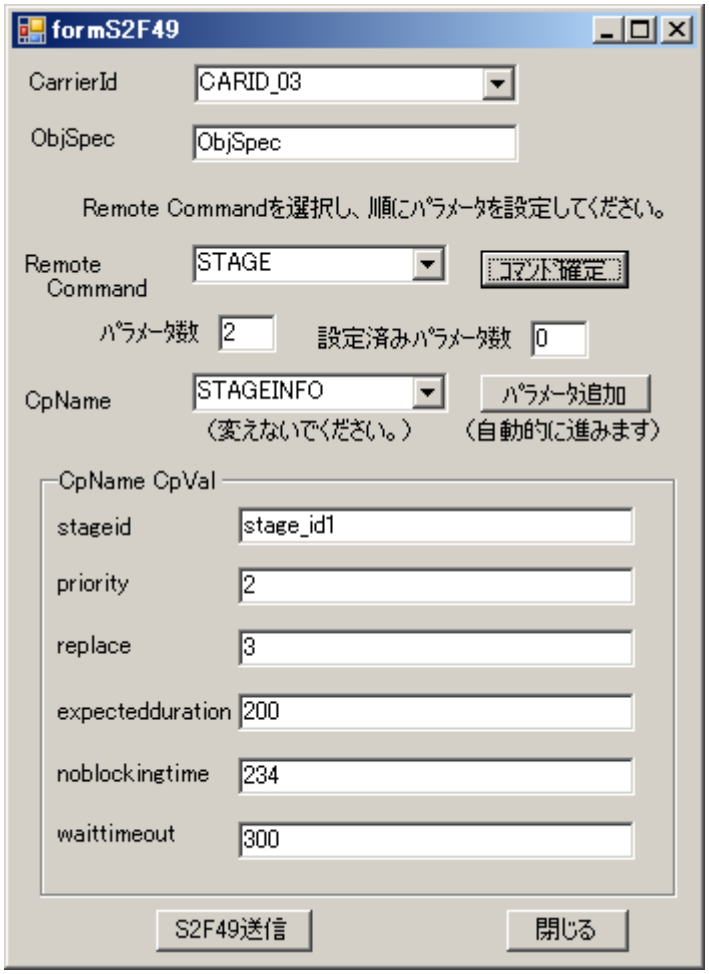
S2F49 に含めたいコマンド、キャリア ID を選び、順次パラメータを情報画面上に設定し、それをホストに送信します。

リモートコマンドが決まると、パラメータ項目入力画面が自動的に変わります。

次のクラスを使用します。

ﾒｯｾｰｼﾞ	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
S2F49	DshERCmd	DshS2F49Send	DshRCmdRsp	DshCallback.callback_s2f49
	DshCar			

操作画面



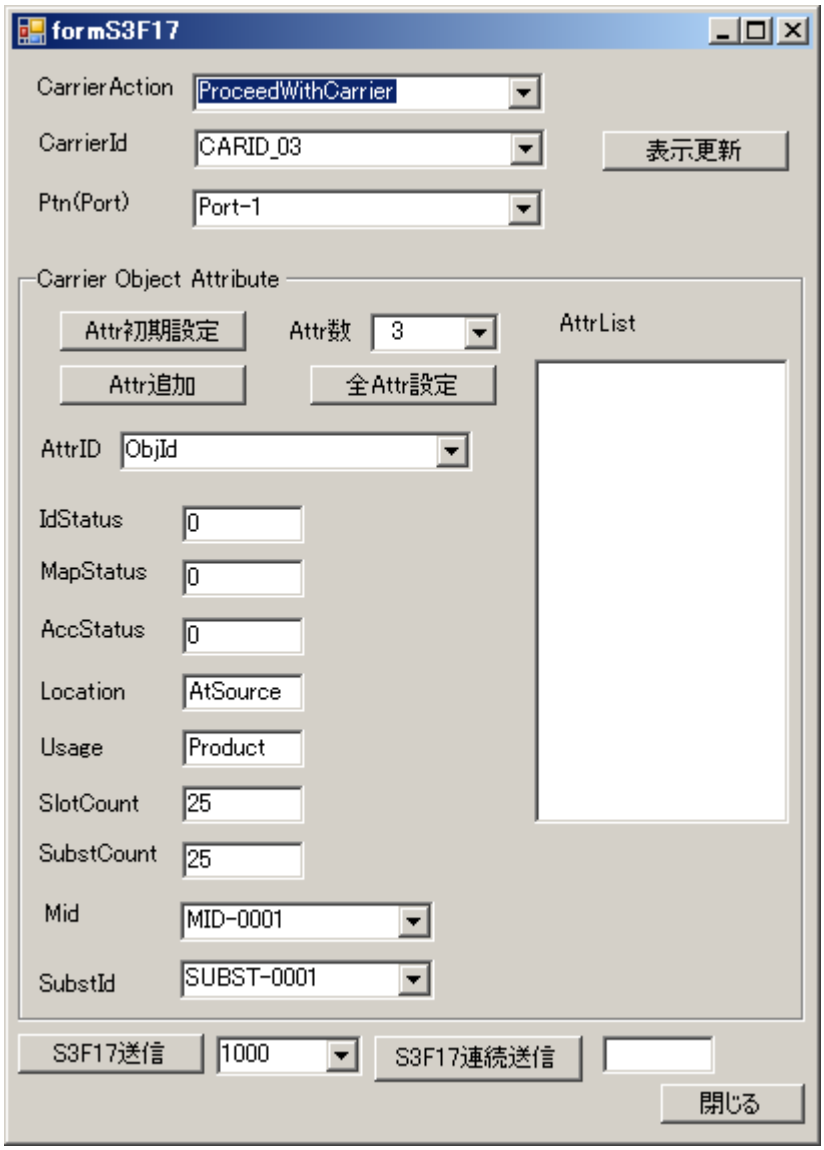
３．12　S3F17 キャリア・アクション・コントロールメッセージ送信画面

S3F17 ホスト・コマンドの操作画面です。
ホスト側からだけの送信メッセージであるため、装置側にはありません。

S3F17 に含めたいアクション、キャリア、ポートを選択し、パラメータを設定し、それをホストに送信します。
次のクラスを使用します。

ﾒｯｾｰｼﾞ	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
S3F17	DshCarAction	DshS3F17Send	DshCarActionRsp	DshCallback.callback_s3f17
	DshCar			

操作画面

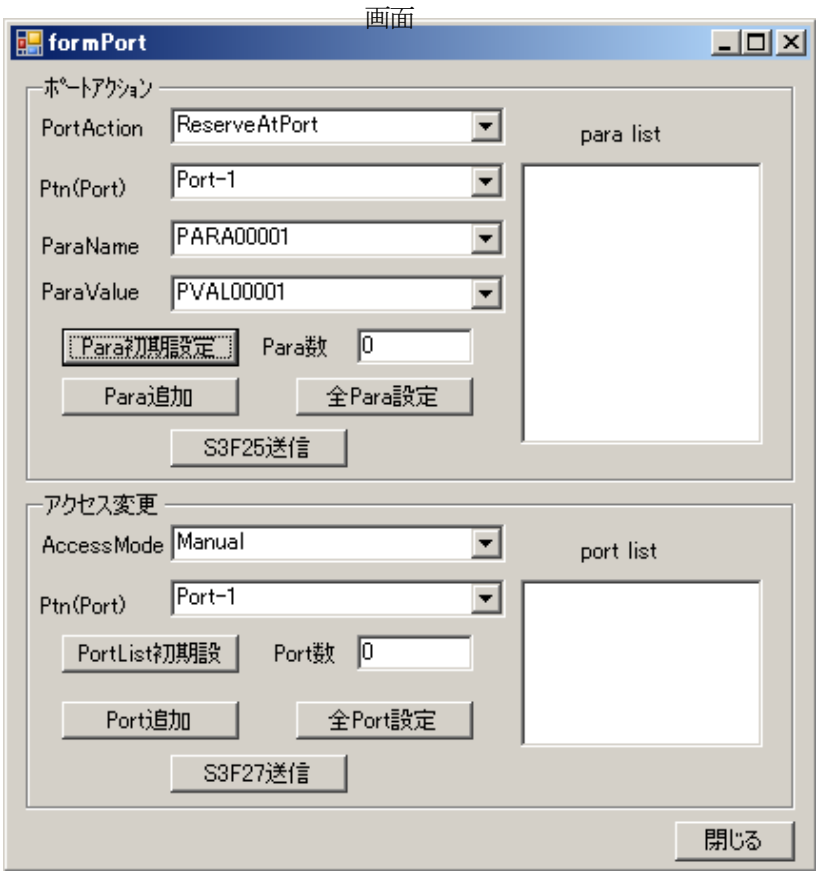


３．13　ポート・コントロールメッセージ送信操作画面

１つの画面に S3F25 ポート・アクションと S3F27 ポート・アクセス変更メッセージ送信の２つの操作を行うことができます。

S3F25，S3F27 は、ホスト側からだけの送信メッセージであるため、装置側にはありません。

画面



３．13．１　S3F25 ポート・アクションメッセージ送信画面

１個のポートと、それに対するポート・アクションを選び、さらにパラメータを設定し、S3F25 メッセージを送信します。　以下のクラスを使用します。

ﾒｯｾｰｼﾞ	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
S3F25	DshPortAction	DshS3F25Send	DshPortActionRsp	DshCallback.callback_S3F25

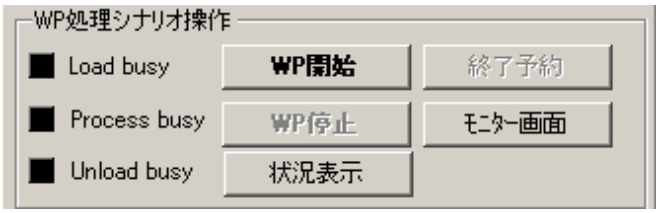
３．13．２　S3F27 ポート・アクセス変更メッセージ送信操作

ポートと、それに対するアクセスモードを選び、S3F27 メッセージを送信します。　以下のクラスを使用します。

ﾒｯｾｰｼﾞ	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
S3F27	DshPortAccess	DshS3F27Send	DshPortAccessRsp	DshCallback.callback_S3F27

３．14　WP(Wafer Processing)オペレーション操作

簡単なWPウエハー・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。

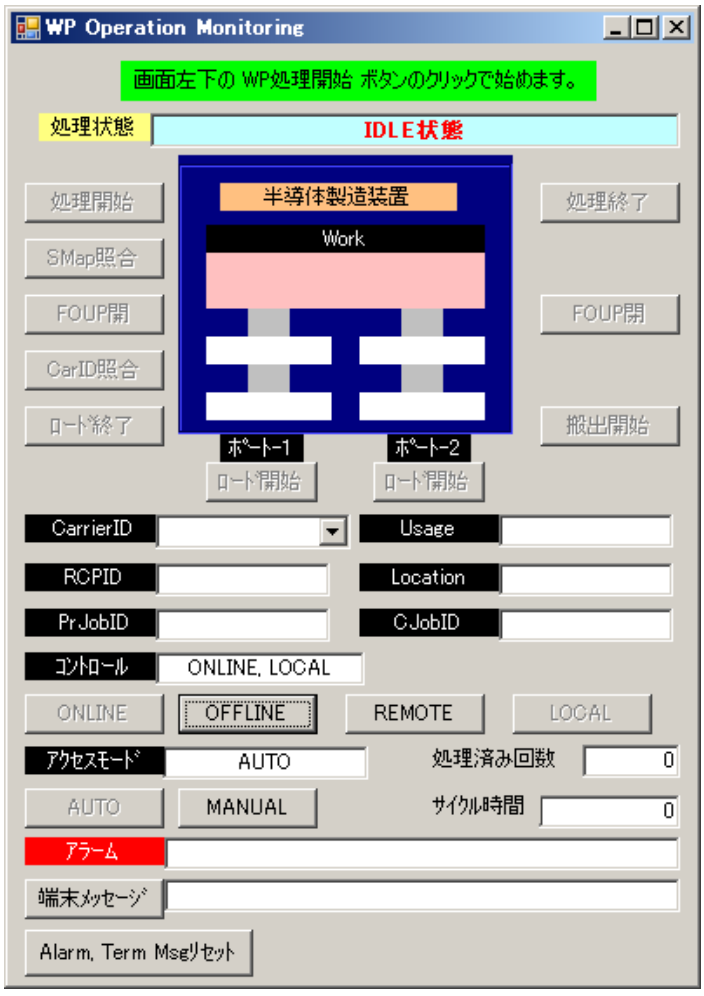


メインフォームのWP 開始ボタンクリックによって開始します。開始によって、以下の処理を行います。

（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　　：　ウエ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

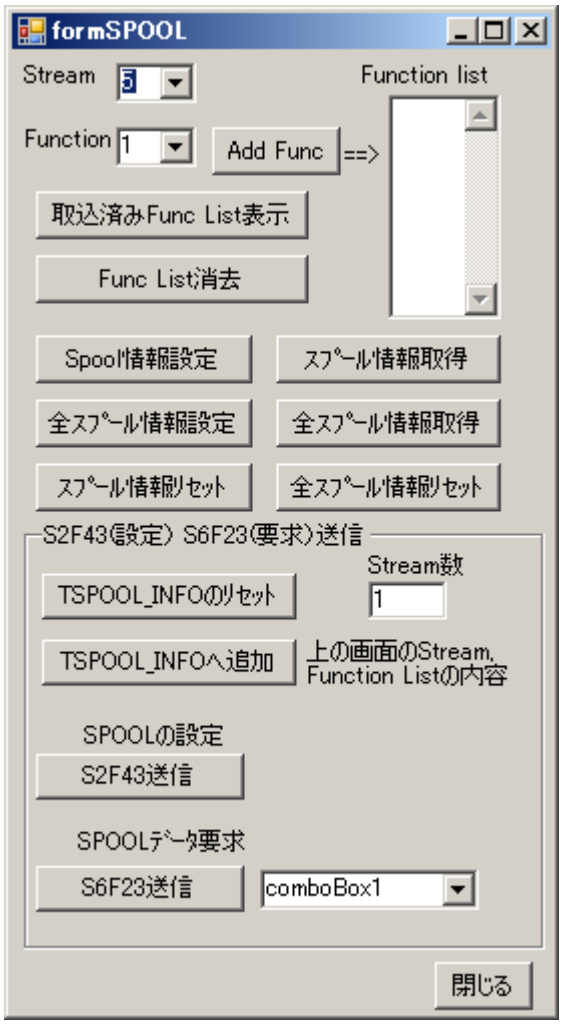


3．15　スプール設定操作画面

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。
ホスト側では、スプール設定 S2F43 の送信、スプールデータ要求の S6F23 のメッセージ送信も行います。
以下のクラスを使用します。

ﾒｯｾｰｼﾞ	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
S2F43	DshSpool	DshS2F43Send	DshSpoolRsp	DshCallback.callback_s2f43
S6F23	－	DshS6F23Send	－	DshCallback.callback_s6f23

ホスト側



装置側

３．16　１次メッセージの送信操作画面

DSHGemLib 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, S6F11 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	ﾒｿｯﾄﾞ	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()		1 次 MSG 送信
DshCallback	callback_send_request		

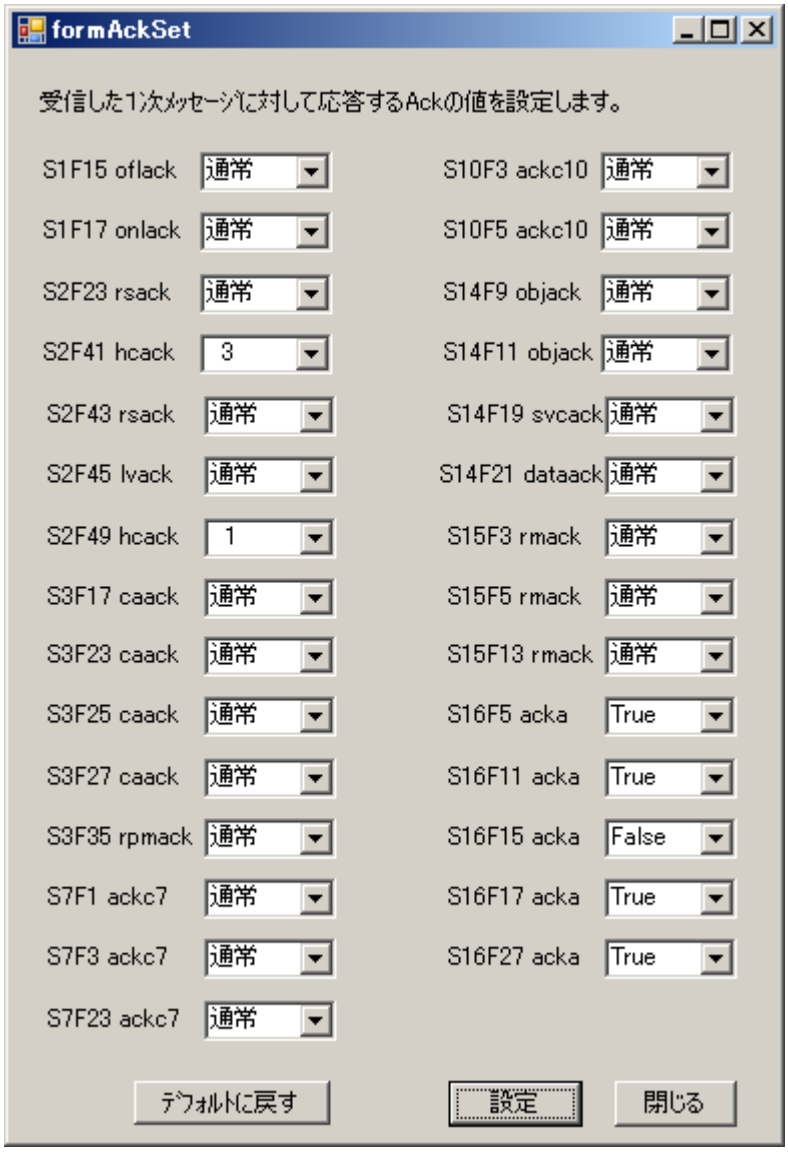
画面

３．17　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



３．18　クラスのトレースとデバッグ機能操作画面

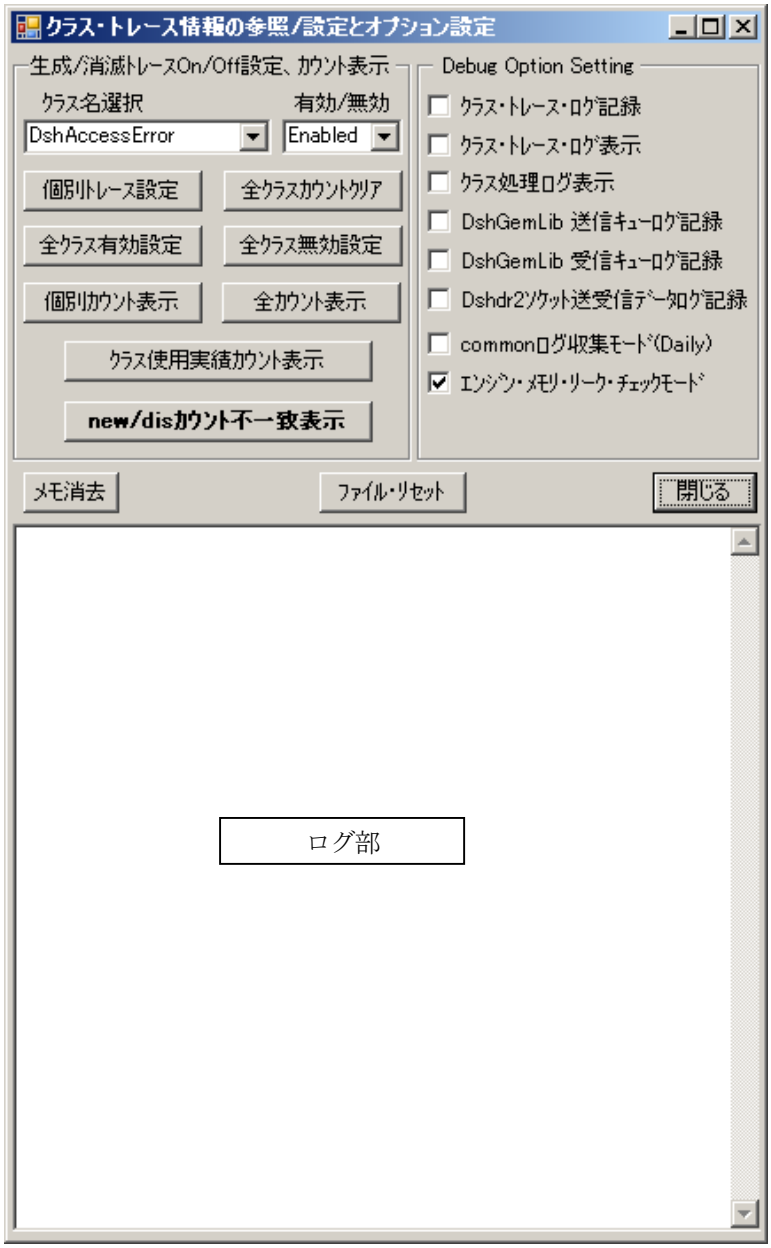
DshGemClass のクラスの生成と消滅のトレースとその累積カウント監視に関する操作と、DSHGemLib 通信エンジンならびにDSHDR2　HSMS 通信ドライバーのメモリ・リークチェックの有効、無効の操作を行います。
これらは、どちらもユーザ・プログラムの開発段階に使用できるデバッグ機能です。

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　DSHGEM-LIB-07-30305-00　　「DSHGEM-CLASS 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



３．18．１　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)
 DshEquipment          new=      1  dis=      0  (fin=      0)
 DshSV                 new=      1  dis=      1  (fin=      0)
    total 5
```

new ： 生成、 dis ： Dispose　　fin ： ﾌｧｲﾅﾗｲｻﾞｰ

・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。

３．18．２　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・ﾛｸﾞ記録 - ログを保存するかどうかを選択します。

○ ｸﾗｽ・ﾄﾚｰｽ・ﾛｸﾞ表示 - 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。DshGemLib.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。

３．18．３　DSHGemLib 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

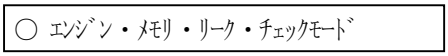
アプリケーション・ソフトウェアのプログラミングについては、メモリ・リークが発生していないかどうかが気になるところです。

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、DshGemLib.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。



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

DSHGemLib.dll エンジン　：　**malloc.log**
DSHDr2.dll ドライバー　：　**dr2_malloc.log**

３．18．４　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。

ﾁｪｯｸﾎﾞｯｸｽ	機能(有効の場合)	備考
○ ｸﾗｽ処理ログ表示	ｸﾗｽの内部ﾃﾞﾊﾞｯｸﾞ用	(DSHGemClass 開発のため)
○ DshGemLib 送信ｷｭｰﾛｸﾞ記録	APP からの 1 次ﾒｯｾｰｼﾞ送信要求が受け付けられたかどうかの確認ﾛｸﾞが DSHGemLib の共通ﾛｸﾞﾌｧｲﾙに追加されます。	
○ DshGemLib 受信ｷｭｰﾛｸﾞ記録	上と同様に、相手から受信したﾒｯｾｰｼﾞが受け付けられたかどうかの確認ﾛｸﾞを共通ﾛｸﾞﾌｧｲﾙに追加されます。	
○ Dshdr2 ｿｹｯﾄ送受信ﾃﾞｰﾀﾛｸﾞ記録	DSHDR2 HSMS 通信ﾄﾞﾗｲﾊﾞｰが TCP/IP ｿｹｯﾄで送受信した生ﾃﾞｰﾀを 16 進表記で DSHDR2 のﾛｸﾞﾌｧｲﾙに含めます。	
○ common ログ収集ﾓｰﾄﾞ(Daily)	DSHGemLib の共通ﾛｸﾞﾌｧｲﾙの記録を１日単位で取るように選択します。	これは、formMain で、選択のためのｺｰﾃﾞｨﾝｸﾞをしない時に、ここで選択できます。ｴﾝｼﾞﾝ開始前に選択が必要です。(開始後は無効)

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、装置ー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） DSHGEM-LIB 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、ウエ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(ｴﾝｼﾞﾝが実施)
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	ｲﾍﾞﾝﾄﾚﾎﾟｰﾄ送信
S10F1	S10F2	H←E	端末要求
S10F3	S10F4	H→E	端末表示、ｼﾝｸﾞﾙﾌﾞﾛｯｸ
S10F5	S10F6	H→E	端末表示、ﾏﾙﾁﾌﾞﾛｯｸ
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]			ﾛｰﾄﾞ可能状態文字列
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		現ﾛｰﾄﾞﾎﾟｰﾄ番号
9	SV_LoadCarId	8201	A[0..32]			ﾛｰﾄﾞｷｬﾘｱ 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		ｽﾛｯﾄﾏｯﾌﾟ 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		ｱﾝﾛｰﾄﾞﾎﾟｰﾄ
22	SV_UnloadCarId	8215	A[0..32]			ｱﾝﾛｰﾄﾞｷｬﾘｱ ID
23	SV_UnloadCarIdStatus	8216	U1[1]	0～5		ｱﾝﾛｰﾄﾞｷｬﾘｱ状態(未使用)
24	SV_UnloadSlotMapStatus	8217	U1[1]	0～5		ｱﾝﾛｰﾄﾞｷｬﾘｱﾏｯﾌﾟ 状態(未使用)
25	SV_UnloadPortAssociationStatus	8218	U1[1]	0～5		ｱﾝﾛｰﾄﾞ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		ｱﾝﾛｰﾄﾞ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]			ｽﾌﾟｰﾙの状態 DSHGEM-LIB が内部で使用する
38	SV_SpoolCountActual	8232	U4[1]			ｽﾌﾟｰﾙ数 DSHGEM-LIB が内部で使用する
39	SV_SpoolCountTotal	8233	U4[1]			ｽﾌﾟｰﾙｶｳﾝﾄ合計 DSHGEM-LIB が内部で使用する
40	SV_SpoolFullTime	8234	A[16]			ｽﾌﾟｰﾙ満杯時刻記憶 DSHGEM-LIB が内部で使用する
41	SV_SpoolStartTime	8235	A[16]			ｽﾌﾟｰﾙ開始時刻 DSHGEM-LIB 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 ｲﾍﾞﾝﾄ内､ﾚﾎﾟｰﾄ 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 ｲﾍﾞﾝﾄ内､ﾚﾎﾟｰﾄ 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	ﾘﾝｸﾚﾎﾟｰﾄ ID	備考
1	CE_Communicating	2	RP_Communicating	通信確立通知
2	CE_SpoolDeactivated	999	(なし)	ｽﾌﾟｰﾙ終了通知
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_PrJobReport	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

DSHGEM-LIB 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。

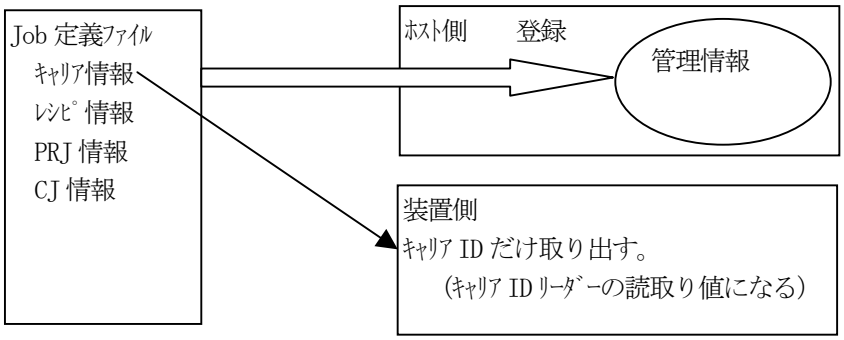
装置開始時に登録します。

CE/変数	予約ｲﾝﾃﾞｸｽ	ID 名	備考
CE 収集ｲﾍﾞﾝﾄ	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

#	ﾒｲﾝｺﾏﾝﾄﾞ	ｻﾌﾞｺﾏﾝﾄﾞ	意味
1	$START		１個のｼﾞｮﾌﾞの開始を示す。
2	$END		１個のｼﾞｮﾌﾞの終了を示す。
3	CARRIER		ｼﾞｮﾌﾞで処理されるｷｬﾘｱを定義します。
		PORT	ﾛｰﾄﾞﾎﾟｰﾄを指定します。
		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		ﾚｼﾋﾟ（またはﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ）を定義します。
		ID	ﾚｼﾋﾟ 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	ﾚｼﾋﾟ（S15F13）または FPP（S7F23）のﾊﾟﾗﾒｰﾀを定義します。
		F_CCODE	FPP（S7F23）のｺﾏﾝﾄﾞｺｰﾄﾞを指定します。
5	PROCESS_JOOB		ﾌﾟﾛｾｽｼﾞｮﾌﾞを定義します。
		ID	ﾌﾟﾛｾｽｼﾞｮﾌﾞ ID を指定します。
6	CONTROL_JOOB		ｺﾝﾄﾛｰﾙｼﾞｮﾌﾞを定義します。
		ID	ｺﾝﾄﾛｰﾙｼﾞｮﾌﾞ 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
  <ﾒｲﾝｺﾏﾝﾄﾞ>{
      <ｻﾌﾞｺﾏﾝﾄﾞ>: <ﾊﾟﾗﾒｰﾀ 1> [,<ﾊﾟﾗﾒｰﾀ 2>[,.],....]
          (必要なだけ)
  }
  <ﾒｲﾝｺﾏﾝﾄﾞ>{
      <ｻﾌﾞｺﾏﾝﾄﾞ>: <ﾊﾟﾗﾒｰﾀ 1> [,<ﾊﾟﾗﾒｰﾀ 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

ﾒｲﾝｺﾏﾝﾄﾞ	ｻﾌﾞｺﾏﾝﾄﾞ	ｻﾌﾞｺﾏﾝﾄﾞﾊﾟﾗﾒｰﾀ
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> ﾚｼﾋﾟまたはﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ ID で、文字列
	BODY	<body> 文字列
	PARA	<name>, <value> <name>はﾊﾟﾗﾒｰﾀ名で文字列 <value>はﾊﾟﾗﾒｰﾀ値で、文字列
	f_ccode	<name>, <value1> [, <value2> [, <value3>,]..] <name>はｺﾏﾝﾄﾞｺｰﾄﾞ名で文字列 <value>はｺﾏﾝﾄﾞ値で、文字列
PROCESS_JOB	ID	<id> ﾌﾟﾛｾｽｼﾞｮﾌﾞ ID 文字列
CONTROL_JOB	ID	<id> ｺﾝﾄﾛｰﾙｼﾞｮﾌﾞ 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、ウェハー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ません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 通信確立通知ｲﾍﾞﾝﾄ Date and Time Set Request
2.	ホスト主導	<--- S1F13 S1F14 ===> S6F11 --->| <=== S6F12 <--- S2F31 S2F32 ===>	Establish Communication Request 通信確立通知ｲﾍﾞﾝﾄ Date and Time Set Request

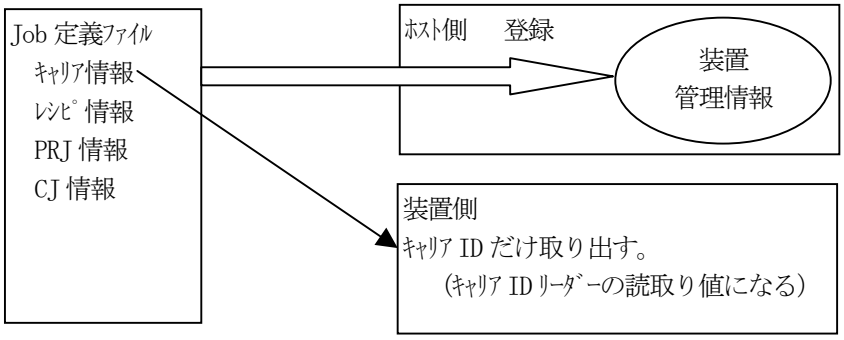
通信確立要求には、S1F13, S6F11 通信のやり取りは、DshEquipment クラスの enable()メソッドが全てを処理してくれます。

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

#	ﾒｲﾝｺﾏﾝﾄﾞ	ｻﾌﾞｺﾏﾝﾄﾞ	意味
1	$START		１個のｼﾞｮﾌﾞの開始を示す。
2	$END		１個のｼﾞｮﾌﾞの終了を示す。
3	CARRIER		ｼﾞｮﾌﾞで処理されるｷｬﾘｱを定義します。
		PORT	ﾛｰﾄﾞﾎﾟｰﾄを指定します。
		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		ﾚｼﾋﾟ（またはﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ）を定義します。
		ID	ﾚｼﾋﾟ 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	ﾚｼﾋﾟ（S15F13）または FPP（S7F23）のﾊﾟﾗﾒｰﾀを定義します。
		F_CCODE	FPP（S7F23）のｺﾏﾝﾄﾞｺｰﾄﾞを指定します。
5	PROCESS_JOB		ﾌﾟﾛｾｽｼﾞｮﾌﾞを定義します。
		ID	ﾌﾟﾛｾｽｼﾞｮﾌﾞ ID を指定します。
6	CONTROL_JOB		ｺﾝﾄﾛｰﾙｼﾞｮﾌﾞを定義します。
		ID	ｺﾝﾄﾛｰﾙｼﾞｮﾌﾞ 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
  〈ﾒｲﾝｺﾏﾝﾄﾞ〉{
      〈ｻﾌﾞｺﾏﾝﾄﾞ〉: 〈ﾊﾟﾗﾒｰﾀ 1〉 [,〈ﾊﾟﾗﾒｰﾀ 2〉[,.],....]
          (必要なだけ)
  }
  〈ﾒｲﾝｺﾏﾝﾄﾞ〉{
      〈ｻﾌﾞｺﾏﾝﾄﾞ〉: 〈ﾊﾟﾗﾒｰﾀ 1〉 [,〈ﾊﾟﾗﾒｰﾀ 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

ﾒｲﾝｺﾏﾝﾄﾞ	ｻﾌﾞｺﾏﾝﾄﾞ	ｻﾌﾞｺﾏﾝﾄﾞﾊﾟﾗﾒｰﾀ
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〉 ﾚｼﾋﾟ またはﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ ID で、文字列
	BODY	〈body〉 文字列
	PARA	〈name〉, 〈value〉 〈name〉はﾊﾟﾗﾒｰﾀ名で文字列 〈value〉はﾊﾟﾗﾒｰﾀ値で、文字列
	f_ccode	〈name〉, 〈value1〉 [, 〈value2〉 [, 〈value3〉,]..] 〈name〉はｺﾏﾝﾄﾞｺｰﾄﾞ名で文字列 〈value〉はｺﾏﾝﾄﾞ値で、文字列
PROCESS_JOB	ID	〈id〉 ﾌﾟﾛｾｽｼﾞｮﾌﾞ ID 文字列
CONTROL_JOB	ID	〈id〉 ｺﾝﾄﾛｰﾙｼﾞｮﾌﾞ 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）

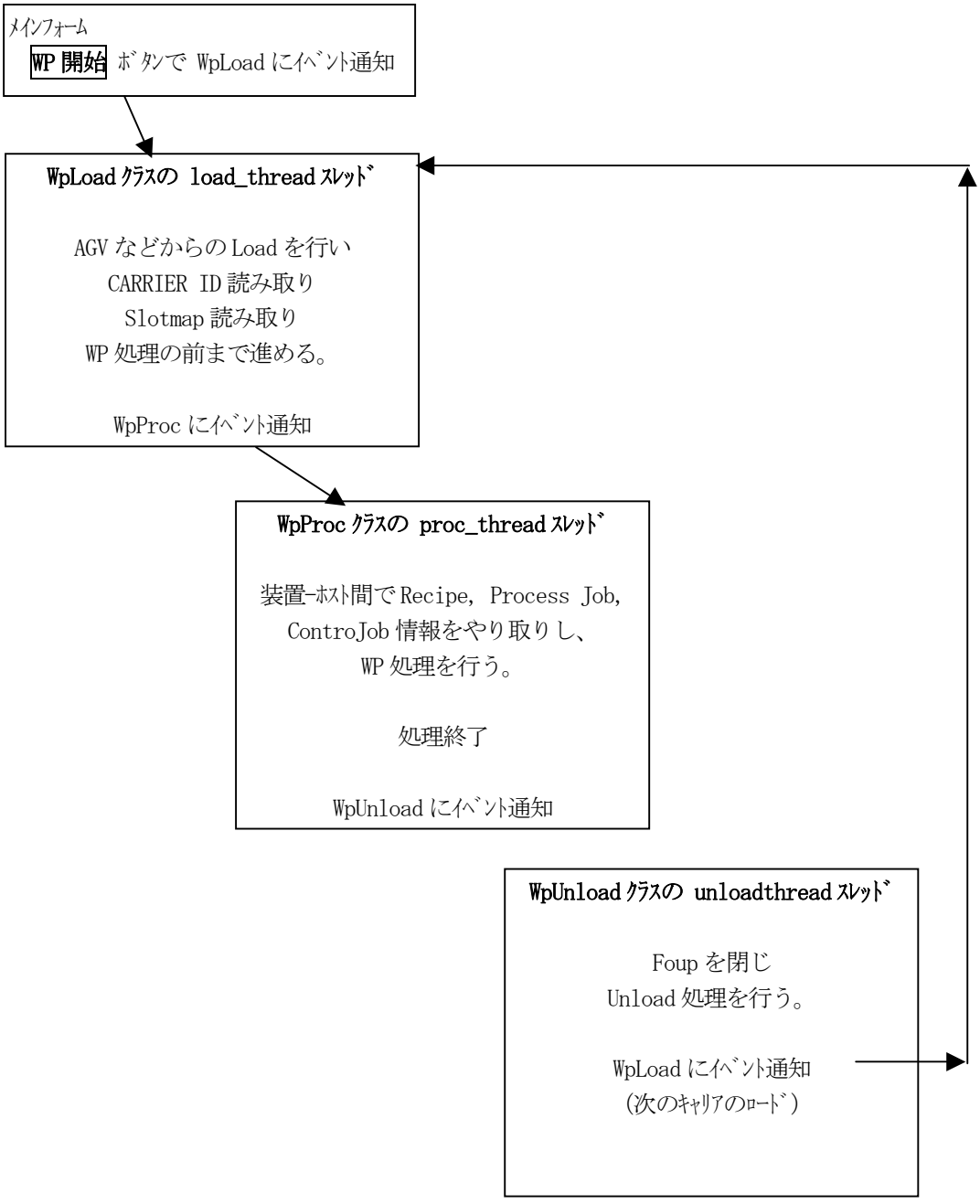
```
$START
//-------------------------------------------------------------
carrier{
        port:   1
        id:     "CARID_01"
        usage:  "PRODUCT"
        slot:   0,10,"LOT_000","SUBST_01_00","LOC_A"     // 0
        slot:   1,11,"LOT_001","SUBST_01_01","LOC_A"     // 1
        slot:   2,12,"LOT_001","SUBST_01_02","LOC_A"     // 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します。
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
	(Load ｲﾍﾞﾝﾄ受信)		(Load ｲﾍﾞﾝﾄ受信)	SOPE_LOAD_IDLE
1	搬送開始 - port-1			
2	CE_RwadyToLoad	S6F11 --->		
		<=== S6F12		SOPE_LOAD_START
3	CE_LoadTransferBlocked	S6F11 --->	状態更新	
	(ｲﾍﾞﾝﾄ通知)	<=== S6F12		SOPE_LOAD_END
	(ｷｬﾘｱ到着)			
4	CE_LoadMaterialArrived	S6F11 --->		
		<=== S6F12		SOPE_READ_CARID
	(ｷｬﾘｱ ID 読取り、登録)		ｷｬﾘｱ ID 取り込み	ｷｬﾘｱ ID 登録
5	CE_LoadWaitingForHost	S6F11 --->		
	(ｷｬﾘｱ ID 付き)	<=== S6F12		
6	ｷｬﾘｱｺﾝﾃﾝﾄ登録	<--- S3F17	ｷｬﾘｱｱｸｼｮﾝ CONTENT	
		S3F18 ===>	ProceedWithCarrier	SOPE_PORT_ASSOCIATED
7	CE_LoadAssociated	S6F11 --->	ID 確認 OK	
		<=== S6F12		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	S6F11 --->		
		<=== S6F12		SOPE_OPEN_FOUP
	(Foup Open)			
12	CE_OpenFoup			SOPE_READ_SLOTMAP
	(ｽﾛｯﾄﾏｯﾌﾟ 照合)	S6F11 --->		
		<=== S6F12		
13	CE_LoadWaitingForHost	S6F11 --->		SOPE_SLOTMAP_VERIFY_END
		<=== S6F12		
		<--- S3F17	ｷｬﾘｱｱｸｼｮﾝ	
		S3F18 ===>	ProceedWithCarrier	
14	CE_SlotMapVerifyOK	S6F11 --->		
		<=== S6F12		SOPE_LOAD_IDLE(終了)
				WpProc ｽﾚｯﾄﾞｲﾍﾞﾝﾄ通知

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 からｲﾍﾞﾝﾄ)		(WpLoad からｲﾍﾞﾝﾄ)	SOPE_PROC_IDLE
2		<--- S15F13 S15F14 ===>	ﾚｼﾋﾟ送信 3 つのうち 1 個	SOPE_PPID_READY ﾚｼﾋﾟ情報登録
3		<--- S16F11 S16F12 ===>	ﾌﾟﾛｾｽｼﾞｮﾌﾞ送信 （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		
	(処理終了)			
14	CE_SubstProcessingComplete	S6F11 --->| <=== S6F12		SOPE_PROC_END

15	CE_ProcessingComplete	S6F11 ---> <=== S6F12		
16	CE_SubstAtDestination	S6F11 ---> <=== S6F12		
17	CE_PrJobComplete ﾌﾟﾛｾｽｼﾞｮﾌﾞ削除	S6F11 ---> <=== S6F12		SOPE_PRJ_COMPLETE
18	CE_CJobComplete	S6F11 ---> <=== S6F12		SOPE_CJ_COMPLETE
19	ｺﾝﾄﾛｰﾙｼﾞｮﾌﾞ削除	<--- S14F11 S14F12 ===>	ｺﾝﾄﾛｰﾙｼﾞｮﾌﾞ 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 からｲﾍﾞﾝﾄ)		(WpProc からｲﾍﾞﾝﾄ)	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 にｲﾍﾞﾝﾄ (次のｷｬﾘｱ Load)

７．各種定義ファイルについて

DSHGEM-LIB 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

DSHGEM-LIB 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－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:¥dshgemlib¥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。

装置 ID-0 用 のファイル例を次に示します。

```
// equip00.cnf - EQID = 0 ------------------------------------

BKUP_PATH = "c:¥DSHGEMLIB¥backup0"
SPOOL_PATH = "c:¥DSHGEMLIB¥spool0"
LOG_PATH  = "c:¥DSHGEMLIB¥log0"
LOG_MODE  = daily
LOG_LIFE  = 6
// LOG_FILE   = "eq00.log"
INFO_FILE = "c:¥dshgemlib¥cnf¥eqinfo.fil"

PP_COUNT = 100
FPP_COUNT = 64
RCP_COUNT = 200
CAR_COUNT = 80
CAR_CAPACITY = 25
SUBST_COUNT = 250
PRJ_COUNT = 30
CJ_COUNT  = 32
TRACE_COUNT = 28
COMM_SIDE = EQUIPMENT
//COMM_SIDE = HOST
COMM_PORT = 1
COMM_DEVICE = 1
INFO_BACKUP = 1
S1F13_SEND  = 2
RP_LISTV = 1
```

ホスト側でも同様の起動ファイルが必要です。装置側との違いは、COMM_SIDE コマンドです。

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 DSHGEM-LIB で使用します。

次に、例を示します。（一部抜粋したものです。）

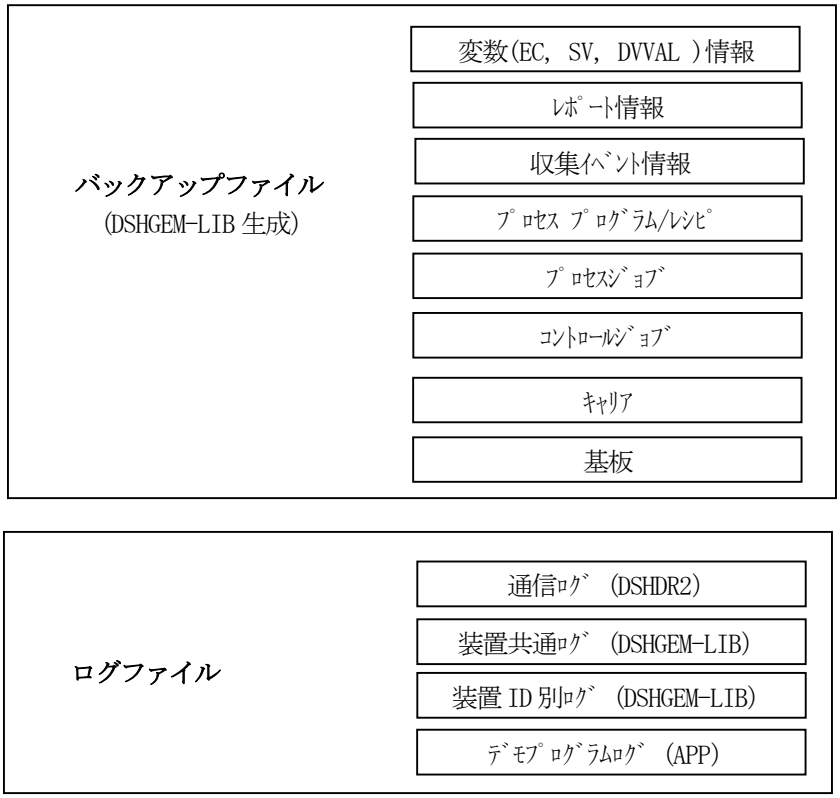
```
//----- C:¥DshGemLib¥bin¥EQINFO.TXT ----------------------------------


//----- EC - Equipment Constant ---------------------------------------------
def_ec  EC_Mdln{
        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
DSHGEM-LIB 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)/レシピ、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）DSHGEM-LIB 装置共通ログファイル
装置固有でない、DSHGEM-LIB 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）DSHGEM-LIB 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。