DSHGEM-LIB　通信エンジンライブラリ(GEM+GEM300)

ソフトウェア・パッケージ

DSHGEM-LIB への手引き

２００８年８月（改-1）

株式会社データマップ

文書番号　DSHGEM-LIB-07-30304-00

［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.	2007.8	初版	
2.	2008.8	付録を追加	関連文書一覧、ｼｽﾃﾑ構成などを概要から付録に移した。 7.1.3 ﾘﾐｯﾄ監視機能用の CE, SV の予約ｲﾝﾃﾞｸｽを追加した。 7.2.8 ﾘﾐｯﾄ監視とﾎｽﾄ通知処理について　を追加した。 その他説明文の訂正を行なった。
3.			
4.			

目　次

１．概要

本説明書は、ユーザが GEM、GEM300 サポート、DSHGEM-LIB 装置通信エンジンライブラリを半導体製造工場のシステムに導入する際、DSHGEM-LIB を理解し、さらに、システムの設計、ソフトウェア開発を進めるための手助けとなるべき基本的な事項について具体的な手順と作業内容を説明するものです。

なお、DSHGEM-LIB 関連資料として **付録－Ａ** の資料を参照することができます。

次章、「２．作業の流れ」のフローチャートに従って作業を進めることになります。

［注釈］

以下、具体的なプログラミングの説明においては、DSHGEM-LIB エンジンが提供する API 関数呼び出しを使っての記述になっています。
その後、.Net 言語(C#, VB)のクラス機能を使用できる DSHGEMClass クラス・ライブラリがサポートが提供されました。アプリケーション・プログラムの開発にはクラス・ライブラリの使用を推奨します。

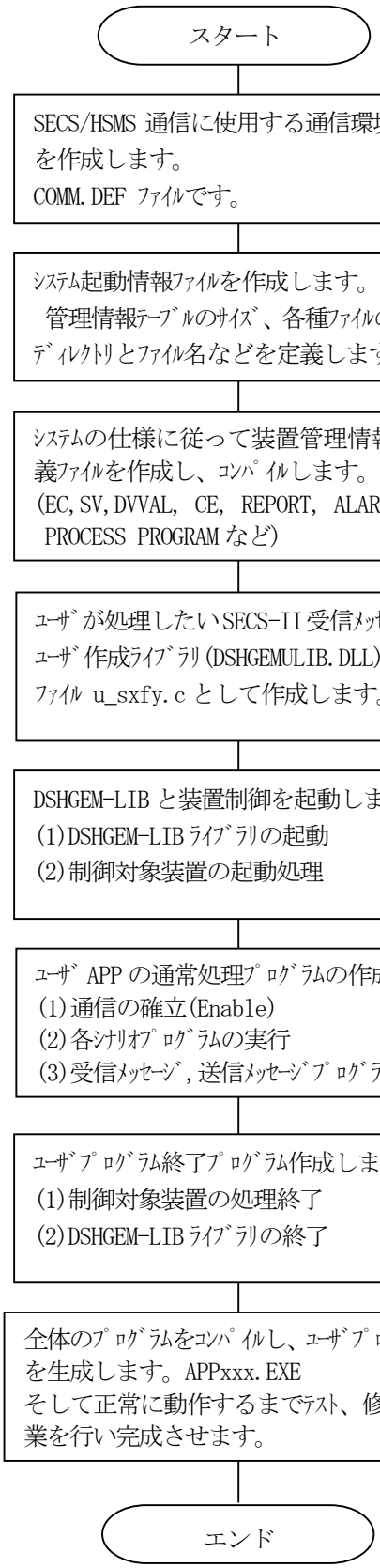
DSHGEMClass クラス・ライブラリについては、下記資料を参照ください。

DSHGEMCLASSクラス・ライブラリ関連文書一覧表

#	文書番号	文書名	注釈
1	DSHGEM-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	ｴﾝｼﾞﾝ、装置起動 管理情報のｱｸｾｽ
2	DSHGEM-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	DSHGEM-09-30363-00	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-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ﾌﾟ 手順も含む
5	DSHGEM-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・ﾃﾞﾊﾞｯｸﾞ用

２．作業の流れ

システムの外部仕様決定後、ソフトウェア設計そしてプログラミングを含めユーザが行うための作業の流れを表します。



「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」参照
装置が通信で使用する PORT,DEVICE の通信ﾊﾟﾗﾒｰﾀを COMM.DEF ﾌｧｲﾙ上に定義します。

DSHGEM-LIB の起動時に必要な起動情報定義ﾌｧｲﾙを作成します。装置単位で準備します。

「DSHGEM-LIB 装置管理情報定義仕様書」参照
編集ﾌﾟﾛｸﾞﾗﾑ、DSHGEMSET.EXE で作成編集可能
「装置管理情報定義ﾌｧｲﾙｺﾝﾊﾟｲﾗ説明書」参照
CEID, REORT ID, VID とのﾘﾝｸ定義が容易にできます。
定義ﾌｧｲﾙをｺﾝﾊﾟｲﾙしｵﾌﾞｼﾞｪｸﾄﾌｧｲﾙを生成します。

「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」6.ﾕｰｻﾞ作成 DSHGEMULIB.DLL ﾌﾟﾛｸﾞﾗﾑ参照してください。
u_sxfy.c ｿｰｽﾌｧｲﾙと各受信ﾒｯｾｰｼﾞ u_s??f??.c ﾌｧｲﾙを準備します。、
そして **DSHGEMULIB**.DLL を生成します。

以下、ﾕｰｻﾞ APP ﾌﾟﾛｸﾞﾗﾑを作成します。
DSHGEM-LIB 内部情報と装置処理起動のための初期化処理です。
GemSetupLibrary(), GemSetupEquipment()関数を使用します。

DSHGEM-LIB API ならびにﾗｲﾌﾞﾗﾘ関数を使ってﾌﾟﾛｸﾞﾗﾐﾝｸﾞします。
通信の確立を行い、ｼﾅﾘｵの実行処理を行うためのﾌﾟﾛｸﾞﾗﾑです。
ﾃﾞﾓﾌﾟﾛｸﾞﾗﾑ(Visual C++)を参考にしてください。

GemTerminateEquipment()と GemTerminateLibrary()関数を使って行います。

ﾃｽﾄは DSHSIM ｼﾐｭﾚｰﾀなどを使って行います。

３．通信環境定義ファイル(COMM.DEF)の作成

通信環境定義ファイルは、DSHDR2.DLL 通信ドライバーに対し、通信ポート、デバイス ID (=セッション ID)プロトコルタイマー値などの通信環境情報を設定するためのテキストファイルです。本ファイル名は GemSetupLibrary()関数の引数になります。

「DSHDR2 SECS/HSMS レベル-2 通信制御ドライバーユーザーズマニュアル」の３．環境ファイル仕様に従って作成します。

ファイルには、基本的にコマンド形式で設定項目とその値を表現します。３種類のコマンドがあります。
START メインコマンド行から始まり、END コマンドで終る行の間に指定するコマンドです。

（1）一般的なコマンド

```
START DSH
    MAX_MSG_SIZE = x100040              # 最大ﾒｯｾｰｼﾞｻｲｽﾞ (ﾊﾞｲﾄ数)          1MB+40
    MAX_TRANSACTION = 512               # 管理する最大ﾄﾗﾝｻﾞｸｼｮﾝ数 2のn乗の値
    LOG_LINE = 100000                   # １個のﾛｸﾞﾌｧｲﾙに保存する最大行数(
    LOG_FILE = ¥dshgemlib¥log¥DSHDR2.LOG  # ﾛｸﾞﾌｧｲﾙを保存するﾌｧｲﾙ名 (ﾌﾙﾊﾟｽ名)
    LOG_MODE = 0                        # save old log file  4世代分のﾛｸﾞﾌｧｲﾙを残す
    MON_PORT = 9999                     # 通信ﾛｸﾞﾓﾆﾀｰﾌﾟﾛｸﾞﾗﾑとのTCP/IP接続ﾎﾟｰﾄを指定
END
```

MAX_MSG_SIZE の値は、仕様上の SECS-II メッセージのバイトサイズとログファイル関連のコマンドについて必要に応じて値を変えてください。

（2）ポート定義コマンド

DSHDR2 通信ドライバー上における通信ポートの番号とそのパラメータを定義します。
相手装置との通信は SECS-I または SECS-SS プロトコルで行いますが、装置との通信のためのポート番号です。(TCP/IP のポートではありません。)この通信ポートに１個以上のデバイスを定義することができます。
ここでは、例として HSMS-SS プロトコルについて説明します。

```
START PORT
    PORT = 1                            # HSMS-SS のﾎﾟｰﾄ番号
    PROTOCOL = HSMS                     # HSMS-SS を指定
    PORT_MODE = ACTIVE                  # Activeﾎﾟｰﾄ (Passiveﾎﾟｰﾄでは Passive)
    TCP_PORT = 5001                     # このﾎﾟｰﾄが使用する TCP/IP のﾎﾟｰﾄ
    IP = 192.168.1.2                    # 接続相手の Pssiveﾎﾟｰﾄの IP ｱﾄﾞﾚｽ
    T3 = 45                             # T3ﾀｲﾑｱｳﾄ時間 (秒単位,小数点１位まで有効)
    T5 = 10                             # T5    “
    T6 = 5                              # T6    “
    T7 = 10                             # T7    “
    T8 = 5                              # T8    “
    LINKTEST = 60                       # Linktest の周期(秒単位)
END
```

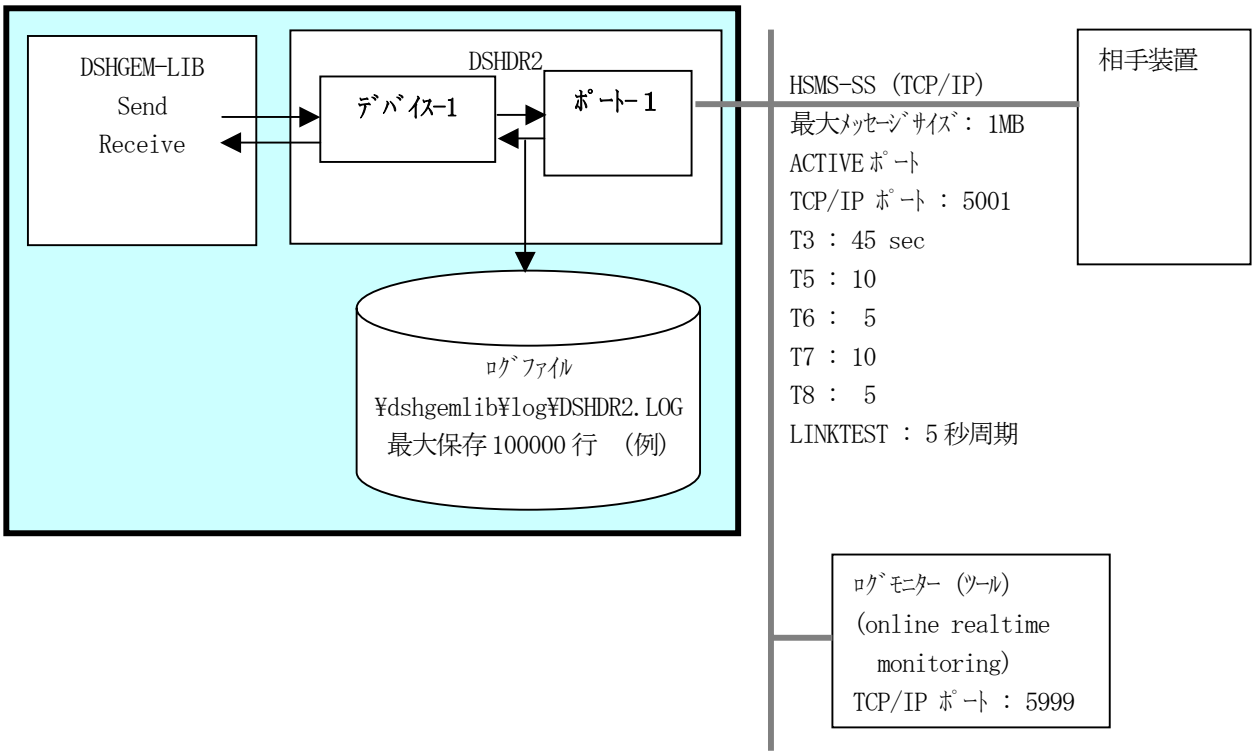
（3）デバイス定義コマンド

DSHDR2 通信ドライバー上における通信デバイスの番号とそのパラメータを定義します。
相手装置との通信は SECS-I または SECS-SS プロトコルで行いますが、（2）で説明したポート番号に含まれ、固有のデバイス ID（セッション ID）を有するデバイス番号です。

```
START DEVICE
    DEVICE = 1                          # ﾃﾞﾊﾞｲｽ番号
    DVID = x1234                        # ﾃﾞﾊﾞｲｽ ID (= ｾｯｼｮﾝ ID)  頭の x は 16 進表現の意
    PORT = 1                            # using port-1
END
```

DSHGEM-LIB プログラムは相手装置との送受信をデバイス単位で行います。

以上述べてきたポート、デバイスなどのコマンドの関連を下図に示します。



４．装置起動情報定義ファイルの作成

装置起動情報定義ファイルは、DSHGEM-LIB の構成情報などの起動情報を設定するファイルで、ファイル拡張子".CNF"を持つテキストファイルです。

本定義ファイルに使用するコマンドと定義の仕方については、「DSHGEM-LIB 起動ﾌｧｲﾙ定義仕様書」を参照ください。

本定義ファイルの編集は、使用する DSHGEMSET プログラムを使って行うことがでします。DSHGEMSET の機能、操作方法については、**「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」**を参照ください。

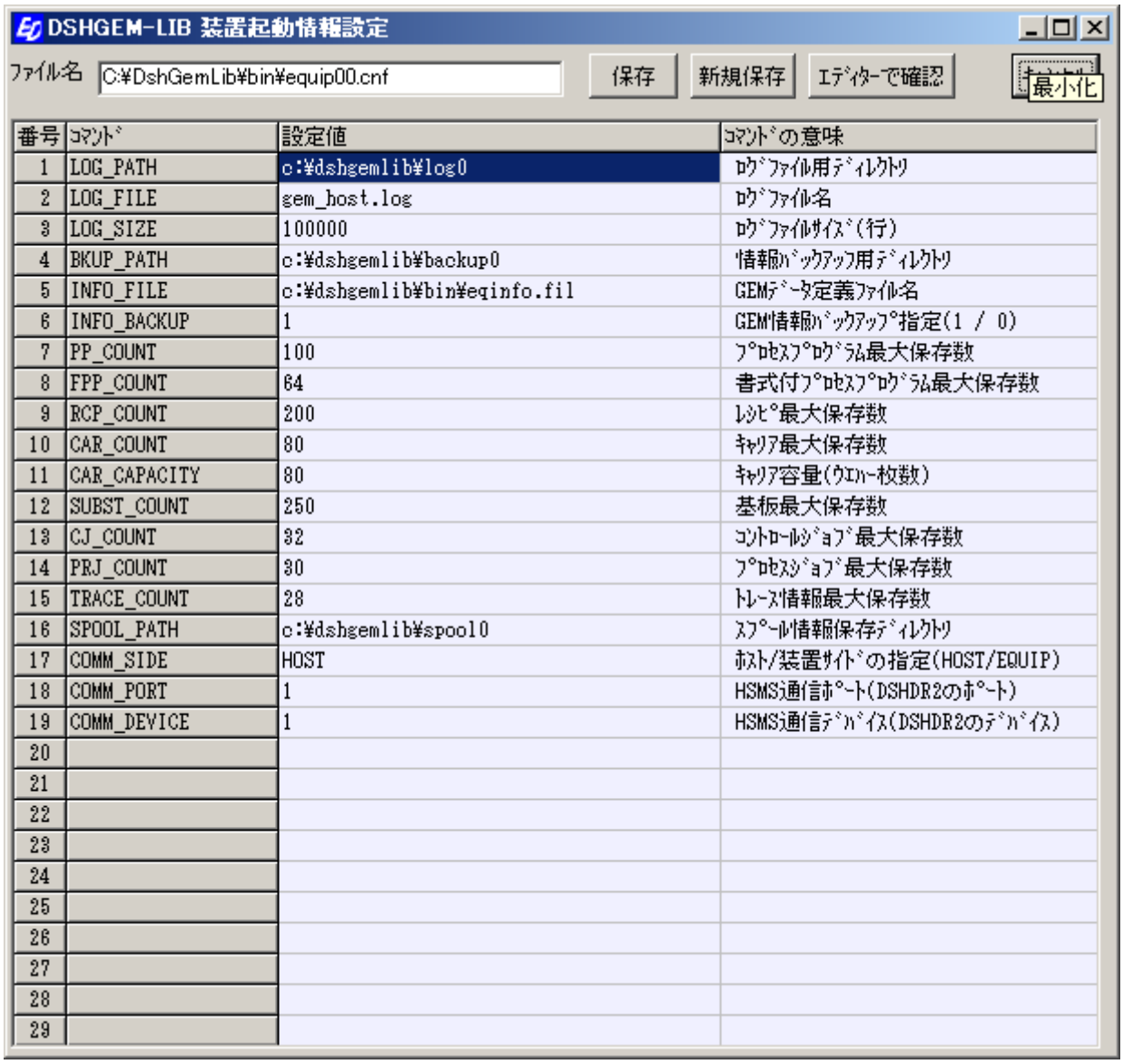
コマンドは、次表の通りです。

#	コマンド名とフォーマット	機　能	コマンド例
1	LOG_PATH = <ﾃﾞｨﾚｸﾄﾘ名>	個別装置のﾛｸﾞﾌｧｲﾙの保存ﾃﾞｨﾚｸﾄﾘ名を指定します。	LOG_PATH = c:¥dshgemlib¥log0
2	LOG_FILE = <ﾌｧｲﾙ名>	個別装置のﾛｸﾞﾌｧｲﾙ名を指定します。	LOG_FILE = “gem_host.log”
3	LOG_SIZE = <行数>	ﾛｸﾞﾌｧｲﾙに保存する最大行数を指定します。	LOG_SIZE = 100000
4	BKUP_PATH = <ﾃﾞｨﾚｸﾄﾘ名>	装置管理情報のﾊﾞｯｸｱｯﾌﾟﾌｧｲﾙを保存するﾃﾞｨﾚｸﾄﾘを指定します。	BKUP_PATH = “C:¥DSHGEM-LIB¥backup0”
5	INFO_FILE = <ﾌｧｲﾙ名>	装置管理情報定義ﾌｧｲﾙ名をﾌﾙﾊﾟｽで指定します。。	INFO_FILE = “c:¥dshgemfil¥eqinfo0.fil”
6	INFO_BACKUP = <1/0>	装置管理情報のﾊﾞｯｸｱｯﾌﾟを行うかどうかを 1,0 で指定します。	BACKUP_FLAG = 1
7	SPOOL_PATH = <ﾃﾞｨﾚｸﾄﾘ名>	SPOOL ﾌｧｲﾙを保存するﾃﾞｨﾚｸﾄﾘを指定します。	SPOOL_PATH = “C:¥DSHGEM-LIB¥spool0”
8	PP_COUNT = <n>	PP(ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ)最大管理数を n 個にします。(S7F3)	PP_COUNT = 64
9	FPP_COUNT = <n>	FPP(書式付ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ)最大管理数を n 個にします。(S7F23)	PP_COUNT = 80
10	RCP_COUNT = <n>	RECIPE 最大管理数を n 個にします。(S15F13)	RCP_COUNT = 80
11	CAR_COUNT = <n>	CARRIER 最大管理数を n 個にします。	CAR_COUNT = 16
12	SUBST_COUNT = <n>	SUBSTRATE 最大管理数を n 個にします。	SUBST_COUNT = 250
13	PRJ_COUNT = <n>	PRJ(ﾌﾟﾛｾｽｼﾞｮﾌﾞ)最大管理数を n 個にします。	PRJ_COUNT = 16
14	CJ_COUNT = <n>	CJ(ｺﾝﾄﾛｰﾙｼﾞｮﾌﾞ)最大管理数を n 個にします。	CJ_COUNT = 16
15	TRACE_COUNT = <n>	TRACE 最大管理数を n 個にする。	TRACE_COUNT = 15
16	CAR_CAPACITY = <n>	１個のｷｬﾘｱの最大収納ｳｴﾊｰ枚数を n 個にします。	CAR_CAPACITY = 25
17	COMM_SIDE = <ｻｲﾄﾞ名>	通信ｻｲﾄﾞを HOST/EQUIPMENT で指定します。	COMM_SIDE = HOST
18	COMM__PORT = <ﾎﾟｰﾄ番号>	DSHDR2 通信ﾄﾞﾗｲﾊﾞｰで使用するﾎﾟｰﾄ番号を指定します。	PORT = 1
19	COMM_DEVICE = <ﾃﾞﾊﾞｲｽ番号>	DSHDR2 通信ﾄﾞﾗｲﾊﾞｰで使用するﾃﾞﾊﾞｲｽ番号を指定します。	DEVICE = 1

(注) 装置変数、収集イベント、レポート、アラーム情報に関する登録個数は、
装置管理情報ファイル内に定義される数が登録数になります。

次の画面は、DSHGEMSET.EXE 編集プログラムでの起動ファイルの編集画面です。

[DSHGEM-LIB 用設定画面]



番号	ｺﾏﾝﾄﾞ	設定値	ｺﾏﾝﾄﾞの意味
1	LOG_PATH	c:¥dshgemlib¥log0	ﾛｸﾞﾌｧｲﾙ用ﾃﾞｨﾚｸﾄﾘ
2	LOG_FILE	gem_host.log	ﾛｸﾞﾌｧｲﾙ名
3	LOG_SIZE	100000	ﾛｸﾞﾌｧｲﾙｻｲｽﾞ(行)
4	BKUP_PATH	c:¥dshgemlib¥backup0	情報ﾊﾞｯｸｱｯﾌﾟ用ﾃﾞｨﾚｸﾄﾘ
5	INFO_FILE	c:¥dshgemlib¥bin¥eqinfo.fil	GEMﾃﾞｰﾀ定義ﾌｧｲﾙ名
6	INFO_BACKUP	1	GEM情報ﾊﾞｯｸｱｯﾌﾟ指定(1 / 0)
7	PP_COUNT	100	ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ最大保存数
8	FPP_COUNT	64	書式付ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ最大保存数
9	RCP_COUNT	200	ﾚｼﾋﾟ最大保存数
10	CAR_COUNT	80	ｷｬﾘｱ最大保存数
11	CAR_CAPACITY	80	ｷｬﾘｱ容量(ｳｴﾊｰ枚数)
12	SUBST_COUNT	250	基板最大保存数
13	CJ_COUNT	32	ｺﾝﾄﾛｰﾙｼﾞｮﾌﾞ最大保存数
14	PRJ_COUNT	30	ﾌﾟﾛｾｽｼﾞｮﾌﾞ最大保存数
15	TRACE_COUNT	28	ﾄﾚｰｽ情報最大保存数
16	SPOOL_PATH	c:¥dshgemlib¥spool0	ｽﾌﾟｰﾙ情報保存ﾃﾞｨﾚｸﾄﾘ
17	COMM_SIDE	HOST	ﾎｽﾄ/装置ｻｲﾄﾞの指定(HOST/EQUIP)
18	COMM_PORT	1	HSMS通信ﾎﾟｰﾄ(DSHDR2のﾎﾟｰﾄ)
19	COMM_DEVICE	1	HSMS通信ﾃﾞﾊﾞｲｽ(DSHDR2のﾃﾞﾊﾞｲｽ)
20			
21			
22			
23			
24			
25			
26			
27			
28			
29			

５．装置管理情報定義ファイルの作成とコンパイル

次に、GEM スタンダードに含まれ管理対象となり、初期設定が必要な情報の定義ファイルを編集作成します。そして、それをコンパイルし、DSHGEM-LIB で使用できるようにします。

本情報ファイルの定義は装置別に作成する必要があります。ただし、複数の装置の制御で全装置の装置管理情報の仕様が全て同じであれば、１つのファイルを全装置に対して使用することができます。

装置管理情報の定義についての詳しい内容は、

「DSHGEM-LIB 装置管理情報定義仕様書」を、

また、編集、コンパイルのために使用する DSHGEMSET プログラムについての機能、操作方法については、

「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」を参照ください。

定義対象になる装置管理情報は以下の情報です。

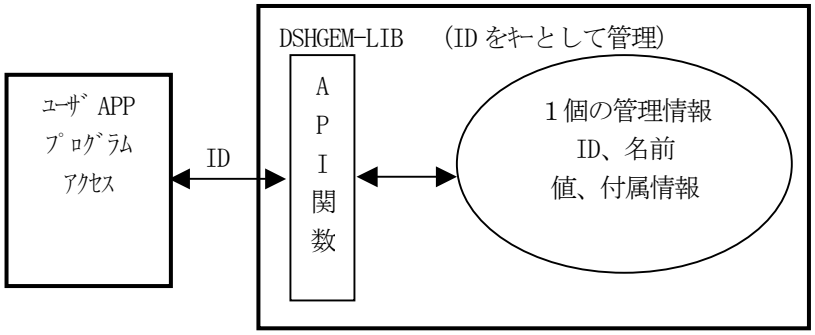
番号	情報区分	情報名		必須は○印	
1	変数	EC	装置定数	○	
		SV	装置状態変数	○	
		DVVAL	装置ﾃﾞｰﾀ変数	○	
2	収集イベント	CE	収集ｲﾍﾞﾝﾄ	○	
		REPORT	レポート	○	
3	アラーム	ALARM	アラーム	○	
4	プロセス	PP	ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ		PP,FPP,RECIPE の何れかを選択
5	プログラム	FPP	書式付ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ		
6	（レシピ）	RECIPE	ﾚｼﾋﾟ		
7	トレース	TRACE	状態変数ﾄﾚｰｽ		
8	スプール	SPOOL	ﾒｯｾｰｼﾞｽﾌﾟｰﾙ		

上の表で、番号１，２，３の情報の設定は必須です。他の情報については、APP プログラムが API 関数を使って必要に応じて DSHGEM-LIB を通して定義し登録することができます。

ここでは、必須定義情報について簡単に例を挙げて説明します。（詳しくは「装置管理情報定義仕様書」を参照）

装置管理情報の中の個々の情報には、基本的にシステム内部でユニークな ID と名前が与えられます。
（装置側の場合、ホストから与えられる管理情報がありますが、これには ID があるけれど名前は与えられません。）

装置管理情報のアクセスあるいは、直接関連するメッセージの送信においては、ID がキーとして使用されます。



５．１　装置管理情報の定義と定義ファイルの作成

システム通信仕様書の中には、通常は、先に述べた装置管理情報の定義仕様書が提供されます。仕様書のフォーマットは様々ですが、大体次に示すようなものです。SV、CE,REPORT について簡単な例を使って説明します。

（1）装置状態変数(SV)

SVID	名前	範囲	規定値	ﾌｫｰﾏｯﾄ	ﾕﾆｯﾄ	内容
1000	SV_Clock			A[20]		装置現時刻 (YYYYMMDDhhmmsscc)
1001	SV_CommunicationState	0～5	0	U1		通信状態
1002	SV_ControlState	1～2	1	U1	state	制御状態
1003	SV_ReadyToLoad	1～2	1	U1		準備できたﾛｰﾄﾞﾎﾟｰﾄ
1004	SV_AccessModeStatus	0～3	0	U1		ﾎﾟｰﾄｱｸｾｽﾓｰﾄﾞ状態
・						
・						

（2）レポート ID

CEID	名前	ｲﾍﾞﾝﾄ発生ﾀｲﾐﾝｸﾞ	ﾘﾝｸ VID
102	RP_Communicating	通信が確立した	SV_Clock
			SV_CommunicationState
103	RP_ControlState	ｺﾝﾄﾛｰﾙ状態が変化した	SV_Clock
			SV_ControlState
104	RP_ReadyToLoad	ﾛｰﾀﾞｰが搬入可能状態になった	SV_Clock
			SV_LoadPortId
105	RP_PortAccessMode	ﾎﾟｰﾄのｱｸｾｽﾓｰﾄﾞが変更になった	SV_Clock
			SV_AccessModeStatus
・			
・			
・			

（3）収集イベント(CE)

CEID	名前	ｲﾍﾞﾝﾄ発生ﾀｲﾐﾝｸﾞ	ﾘﾝｸ ﾚﾎﾟｰﾄ ID
2	CE_Communicating	通信が確立した	RP_Communicating
3	CE_ControlState	ｺﾝﾄﾛｰﾙ状態が変化した	RP_COntrolState
4	CE_ReadyToLoad	ﾛｰﾀﾞｰが搬入可能状態になった	RP_ReadyToLoad
5	CE_Port1AccessMode	ﾎﾟｰﾄ-1 のｱｸｾｽﾓｰﾄﾞが変更になった	RP_Port1AccessMode
・			
・			
・			

装置管理情報定義ファイルはまず、ソースファイルで定義し、その後、コンパイルしてDSHGEM-LIBが解釈できるオブジェクトファイルにコンパイルすることになります。
定義情報ファイルの作成、編集そしてコンパイルは **DSHGEMSET.EXE** プログラムツールを使って行います。

例に出てきた３つの情報例対する定義は、DSHGEM-LIBの装置管理情報定義仕様に従ってそれぞれ以下のようにファイル内に記述します。

① SV情報の定義

```
def_sv  SV_Clock{
        svid:     1000
        format:   A[20]
        nominal:  ""
        }
def_sv  SV_CommunicationState{
        svid:     1001
        format:   U1[1]
        min:      0
        max:      5
        nominal:  0
        }
def_sv  SV_ControlState{
        svid:     1002
        format:   U2[1]
        units:    "state"
        min:      "0"
        max:      "2"
        nominal:  "0"
        event:    "CE_ControlState"  // 値変化時に通知するためのCEID
        }
def_sv  SV_LoadPortId{
        svid:     1003
        format:   U1[1]
        units:    ""
        min:      "1"               // 数値の場合二重引用符で囲んでもOK
        max:      "2"
        nominal:  "1"
        }
def_sv  SV_AccessModeStatus{
        svid:     1004
        format:   U1[1]
        units:    ""
        min:       0
        max:       3
        nominal:  "0"
        }
```

② レポート情報の定義

```
def_report RP_Communicating{
        rptid:    102
        vname:    SV_Clock
        vname:    SV_CommunicationStae
        }
def_report RP_ControlState{
        rptid:    103
        vname:    SV_Clock
        vname:    SV_ControlState
        }
def_report RP_ReadyToLoad{
        rptid:    104
        vname:    SV_Clock
        vname:    SV_LoadPortId
        }
def_report RP_PortAccessMode{
        rptid:     105
        vname:    SV_Clock
        vname:    SV_PortAccessMode
        }
```

③ CE 情報の定義

```
def_ce  CE_Communicating{
        ceid:     2
        enabled:  1
        rptname:  RP_Communicating
        }
def_ce  CE_ControlState{
        ceid:     3
        enabled:  1
        rptname:  RP_ControlState
        }
def_ce  CE_ReadyToLoad{
        ceid:     4
        enabled:  1
        rptname:  RP_ReadyToLoad
        }
def_ce  CE_PortAccessMode{
        ceid:     5
        enabled:  1
        rptname:  RP_PortAccessMode
        }
```

５．２　装置管理情報のコンパイル

５．１で定義した装置管理情報定義ファイルからDSHGEM-LIBが解釈できるオブジェクトファイルにコンパイルしなければなりません。コンパイルの方法としては、次の２つの方法があります。どちらもコンパイラーとしてDshCompile.EXE プログラムを使用します。

DSHGEMSET.EXE 、装置管理情報編集プログラムからは、コンパイルボタンのクリックで簡単にコンパイルできます。

ENG3CONF.EXE の起動書式は次のようになります。

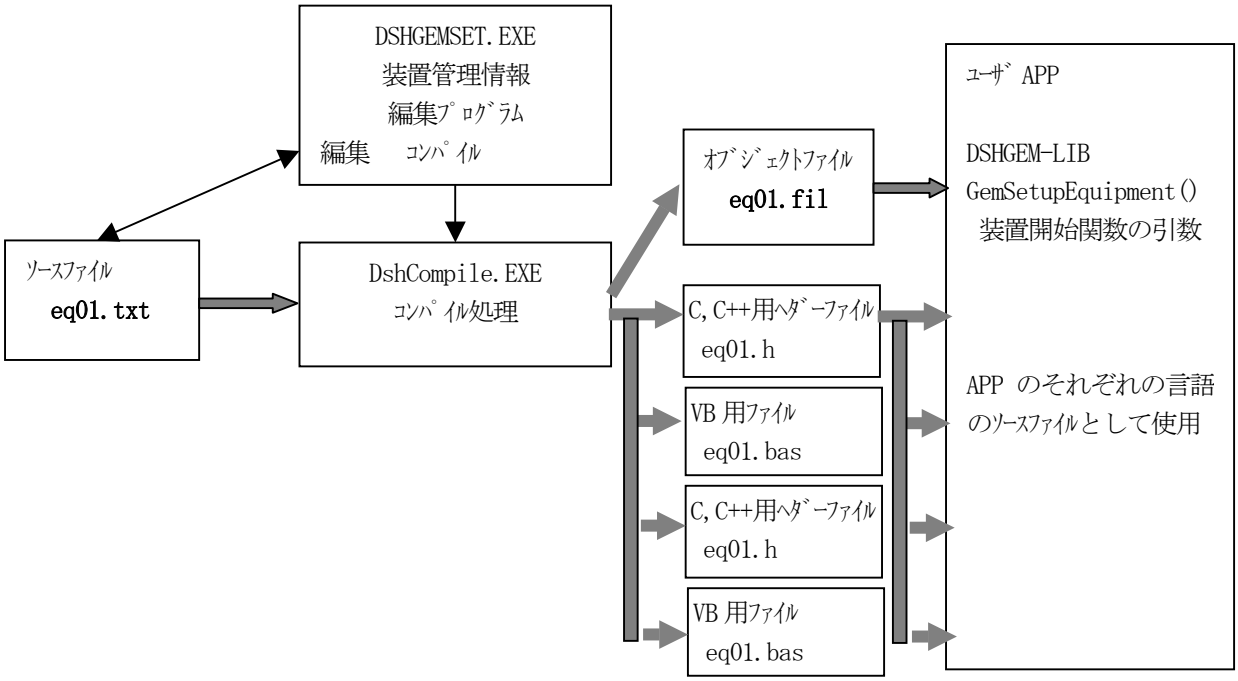
> DshCompile <ｿｰｽﾌｧｲﾙ名> <ﾛｸﾞﾌｧｲﾙ名> <ｴﾝﾄﾞ通知ﾌｧｲﾙ名>
>
> ｿｰｽﾌｧｲﾙ名　　：　装置管理情報定義ﾌｧｲﾙ名(ｿｰｽﾌｧｲﾙ)
> ﾛｸﾞﾌｧｲﾙ名　　：　ｺﾝﾊﾟｲﾙｴﾗｰﾒｯｾｰｼﾞ、終了時のﾒｯｾｰｼﾞなどのﾛｸﾞ用ﾌｧｲﾙ名です。
> 省略可能です。省略時はdsh_conf.logに記録されます。
> ｴﾝﾄﾞ通知ﾌｧｲﾙ名　：　DSHGEMSET.EXEに終了通知するためのﾌｧｲﾙ名です。
> 省略してください。

コンパイルの結果、正常に終了した場合は、ソースファイル名の拡張子が .FIL のオブジェクトファイルが生成されます。DSHGEM-LIBの装置起動時に、このオブジェクトファイルを装置のセットアップ関数の引数として与えます。

例えば、 eq01.txt ファイルのコンパイルによって得られるオブジェクトファイルは eq01.fil になります。

コンパイルの結果、副次的な生成物として、ユーザプログラムのヘダーファイルとして使用することができるファイルが生成されます。C, C++そしてVB6言語用のファイルです。ファイルには、各情報の名前とその値がマクロで定義されています。(次ページ参照)

作業の関連は下図の通りです。



DSHGEMSET.EXEによる編集でeq01.txtファイルを作成し、その後、DshCompile.exeでコンパイルし、eq01.filを生成します。そして、それをGemSetupEquipment()関数で使用します。

５．３　コンパイルによって生成されたヘダーファイル例

生成されるソースファイルの例を示します。

.H ファイル(c, C++用)

```
#define  SV_Clock                65536          // 0x00010000
#define  SV_ReadyToLoad          65600          // 0x00010040
#define  CE_ReadyToLoad          65600          // 0x00010040
#define  RP_LoadPort             65792          // 0x00010100
```

.BAS ファイル(VB6.０用)

```
Public Const SV_Clock =              65536          ’ 0x00010000
Public Const SV_ReadyToLoad =        65600          ’ 0x00010040
Public Const CE_ReadyToLoad =        65600          ’ 0x00010040
Public Const RP_ReadyToLoad =        65600          ’ 0x00010040
```

.vb ファイル(VB2008 用)

```
Public Const SV_Clock As Integer        65536       ‘ 0x00010000
Public Const SV_ReadyToLoad As Integer  65600       ‘ 0x00010040
Public Const CE_ReadyToLoad As Integer  65600       ‘ 0x00010040
Public Const RP_LoadPort As Integer     65792       ‘ 0x00010100
```

.cs ファイル(C#2008 用)

```
public const int  SV_Clock =            65536       // 0x00010000
public const int  SV_ReadyToLoad =      65600       // 0x00010040
public const int  CE_ReadyToLoad =      65600       // 0x00010040
public const int  RP_ReadyToLoad =      65792       // 0x00010100
```

６．ユーザ作成 DLL（DSHGEMULIB）プログラムの準備

DSHGEM-LIB 通信ライブラリを利用する際、ユーザプログラムは通信関連処理の中で、次のような処理が必要になります。

（1）ユーザが処理したい受信メッセージの DSHGEM-LIB への登録　－　装置別になります。
（2）受信１次メッセージの処理を行ったあと、応答メッセージを送信するための処理　－　定型的な処理です。

これらの処理は、もちろん、ユーザプログラムの内部に直接組み込むことができます。しかし、これら２つの処理は定型的な処理であるため、便宜的な意味で DLL プログラム化し、ユーザプログラムの外の処理としてプログラミングする方法で行います。この DLL プログラムの名前は、**DSHGEMULIB.DLL**（以下、**GEMULIB** と呼びます）とします。

DSHGEM-LIB は、DSHULIB の標準的なサンプルプログラムをソースファイルの形で提供されます。
ユーザは、これを雛形にして、ユーザが必要な処理を追加し不要な部分を削除することで利用することができます。

６．１　ユーザ処理対象１次メッセージの登録

DSHGEM-LIB が受信した１次メッセージの中、ユーザが自身で処理したいメッセージを DSHGEM-LIB に対して登録する必要があります。これは、個別の装置制御開始のための起動（初期化）処理の中で行います。

（1）ユーザプログラムが登録するための API 関数は次の通りです。

```
API int  APIX GemRegisterDefaultSxFy( int eqid, int no_of_msg, int ev );
         eqid      : 装置 ID です。
         no_of_msg : 登録ﾒｯｾｰｼﾞの最大数
         ev        : 登録ﾒｯｾｰｼﾞを受信したときに DSHGEM-LIB で通知してもらうｲﾍﾞﾝﾄ ID
                     ev=NULL(=0)の場合は、ｲﾍﾞﾝﾄ通知しません。
```

具体的な受信処理するメッセージの定義は、（2）の u_sxfy.c にコーディングします。

（2）u_sxfy.c ソースファイルへの受信メッセージの登録
サンプルの u_sxfy.c の中にはユーザが処理する受信メッセージとして、ホスト、装置双方の受信メッセージ群が登録されています。(次ページを参照)
また、この受信メッセージの登録は、装置 ID 毎に最大８個分について行うことができます。
u_sxfy.c 上では、pri_pro_info_ptr_tab[] に次ページで示す登録メッセージ定義テーブルのポインタを設定します。

```
static  TPRI_PRO_INFO   *pri_pro_info_ptr_tab[MAX_EQID] ={
        pri_pro_info_tab_1,       // EQID-0
        pri_pro_info_tab_2,       //      -1   (本例では全装置が同じﾒｯｾｰｼﾞの受信処理)
        pri_pro_info_tab_3,       //      -2
        pri_pro_info_tab_4,       //      -3
        pri_pro_info_tab_5,       //      -4
        pri_pro_info_tab_6,       //      -5
        pri_pro_info_tab_7,       //      -6
        pri_pro_info_tab_8        //      -7
        };
```

受信メッセージ定義登録テーブルのサンプルでは、次のように設定されています。ここには、ホスト、装置、両サイドの受信メッセージが登録されています。
装置側の通信を行う場合は、ホスト側のメッセージをリストから削除してください。ホスト側通信の場合は装置側のメッセージを削除してください。

```
static  TPRI_PRO_INFO   pri_pro_info_tab_1[] ={
//        S   F x1 x2

//----- from HOST ------------------------------------------
        { 1, 15, 1, 0 },        // OFFLINE Request
        { 1, 17, 1, 0 },        // ONLINE  Request

        { 2, 23, 1, 0 },        // Trace Command
        { 2, 41, 1, 0 },        // Remote Command
        { 2, 43, 1, 0 },        // Spool
        { 2, 45, 1, 0 },        // Limit
        { 2, 49, 1, 0 },        // Enhanced Remote Command

        { 3, 17, 1, 0 },        // Carrier Action
        { 3, 23, 1, 0 },        // Port Group Action

//----- from EQ --------------------------------------------
        { 5,  1, 1, 0 },        // Alarm Report Send

        { 6,  1, 1, 0 },        // Trace Data Send
        { 6, 11, 1, 0 },        // Event Report Send
        { 6, 13, 1, 0 },        // Annotated Event Report Send
        {10, 1,  1, 0 },        // Terminal Text

        { 0, 0, 0, 0 }          // Terminator(終わり)
```

ここで、TPRI_PRO_INFO 構造体は、次のようになります。

```
typedef struct{
        int     s;                      // stream   Sx
        int     f;                      // function Fy
        int     queue_flag;
        int     (WINAPI *func)();
        }TPRI_PRO_INFO;
```

(注)　int　　(WINAPI *func)();　は現在使用していません。

６．２　受信１次メッセージ処理後の応答メッセージ送信

ユーザは、受信した１次メッセージの処理を終えた後、それに対する２次メッセージを応答送信する必要があります。この２次メッセージの応答送信処理は、メッセージによっても違いますが、大体、定型的な処理になります。

DSHGEM-LIB では、基本的に、応答２次メッセージの送信処理は、GEMULIB（ユーザ作成 DLL プログラム）内に準備して行うようにします。すなわち、GEMULIB に、受信する全１次メッセージに対する応答送信関数を準備することになります。

以下、例として、S1F17 メッセージ処理に使用する関数について説明します。

（1）S1F17 オフライン通知の応答処理例 – u_s1f17.c ソースファイルを参照してください。

最初に、GemGetSecsMsgReq()関数を使って受信メッセージ S1F17 を取得します。TMSG_QUEUE_INFO 構造体内に受信メッセージ情報が格納されます。TMSG_QUEUE_INFO 構造体は次のメンバーを含んでいます。

```
typedef struct{
        ID_TR   trid;                   // DSHDR2ﾄﾞﾗｲﾊﾞｰが与えるﾄﾗﾝｻﾞｸｼｮﾝ ID
        DSHMSG  *smsg;                  // 受信ﾒｯｾｰｼﾞが格納されている DSHDR2 ﾒｯｾｰｼﾞ構造体ﾎﾟｲﾝﾀ
        DSHMSG  **prmsg;                // 使用しない。(応答 msg のﾎﾟｲﾝﾀ)
        int     (WINAPI **callback)();  // 使用しない。終了時に呼出す関数(=0 ならば関数が無い)
        } TMSG_QUEUE_INFO;
```

ユーザプログラムは、TMSG_QUEUE_INFO 構造体の smsg に格納されているメッセージを取り出し、そのの処理を行い、終ったら、応答メッセージ S1F18 を送信するための関数、DshResponseS1F18()を呼出します。DshResponseS1F18()関数は、次の通りです。引数として 装置 ID eqid, トランザクション ID trid, ならびにＳ１F18 メッセージ内に設定 ACK の値、onlack です。

```
API int  APIX DshResponseS1F18( int eqid, ID_TR trid, int onlack )
{
        DSHMSG  *prmsg;
        BYTE    *text_ptr;
        int     ei;
    prmsg    = calloc( sizeof(DSHMSG), 1 );
    text_ptr = malloc( S1F18_SIZE);                    // onlack 用
    DshSetupMsgForSend( prmsg, 1, 17+1, 0, text_ptr, S1F18_SIZE );
    D_PutItem( prmsg, ICODE_B, &onlack, 1 );
    ei = DshResponseSxFy( eqid, trid, prmsg, DshAcceptS1F17Callback );
    U_DshFreeMsg( prmsg );
    return 0;
}

API int  APIX  DshAcceptS1F17Callback( int eqid, DSHMSG *rmsg )
{
    DshFreeMsg( rmsg );                 // free rmsg->buffer and rmsg
    return 0;
}
```

ここで、DshSetupMsgForSend()は、prmsg 内に text_ptr, と Stream(=1), Function(=18)を設定する関数です。

D_PutItem()で onlack(1 バイト)をテキストに設定した上で、DshResponseSxFy()関数を使って S1F18 メッセージを送信します。
DshResponseSxFy()関数は、次の通りです。

```
API int  APIX DshResponseSxFy( int eqid, ID_TR trid, DSHMSG *prmsg,
                                                int (WINAPI *callback)() );

            eqid     : 装置 ID (0,1,2.. )
            trid     : DSHDR2 から受信した１次ﾒｯｾｰｼﾞに与えられたﾄﾗﾝｻﾞｸｼｮﾝ ID
            prmsg    : 受信した１次ﾒｯｾｰｼﾞが格納されている DSHMSG 構造体のﾎﾟｲﾝﾀ
            callback: 応答ﾒｯｾｰｼﾞ送信終了後に呼び出されるｺｰﾙﾊﾞｯｸ関数を指定します。
```

（注）callback 関数の引数には、装置 ID と DSHMSG 構造体のﾎﾟｲﾝﾀがありますが、
DSHMSG 構造体とその内容は、DSHGEM-LIB で新しいﾒﾓﾘを確保し、ｺﾋﾟｰされていますので、ここで callback 関数に与えられた rmsg のﾒﾓﾘを開放してください。

もし、ユーザにおいて、さらに追加したい処理があれば追加してもかまいません。

７．アプリケーションプログラムの作成

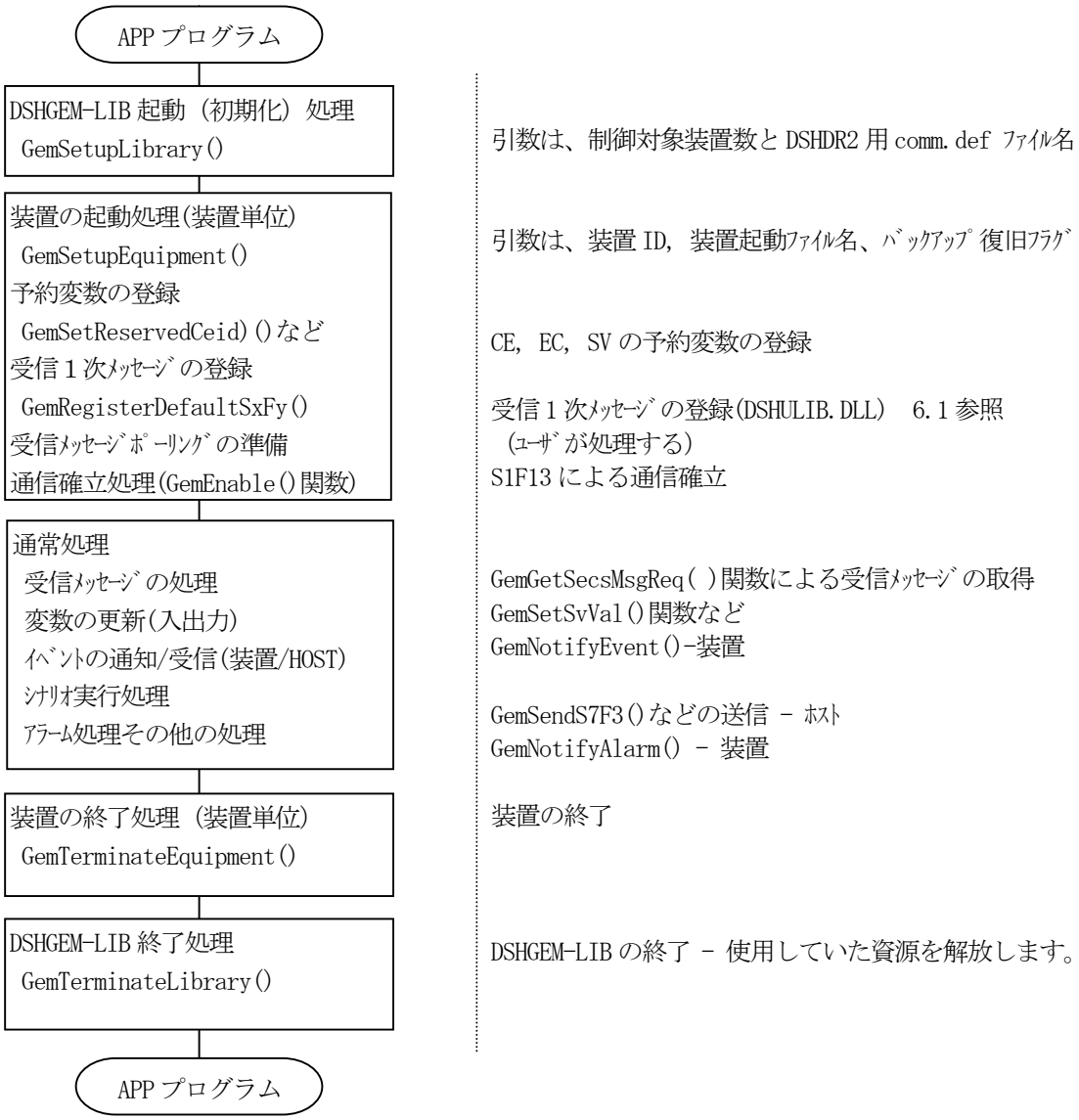
ユーザが作成するアプリケーションプログラムが、DSHGEM-LIB 関連で行うべき、あるいは行うことができる処理について説明します。(デモプログラムの参照によって、より詳細について理解できます)

大きく３つに分けて説明します。

（１）DSHGEM-LIB の初期化処理
- DSHGEM-LIB の初期起動処理
- 装置の初期起動と初期設定処理

（２）DSHGEM-LIB を使用しての通常処理
- イベント通知、アラーム通知
- １次メッセージの受信処理
- 装置管理情報の値の更新など

（３）DSHGEM-LIB の終了処理
- 装置の終了処理
- DSHGEM-LIB の終了処理



７．１　初期化処理

最初に、DSHMGEM-LIB を起動し、その後、制御する装置を個別に起動します。

７．１．１　GemSetupLibrary()による DSHGEM-LIB の起動（初期化）処理

DSHGEM-LIB の初期設定と起動を GemSetupLibrary()関数を使って行います。

```
API int APIX GemSetupLibrary (
    int  max_eq,                  // 最大制御装置数
    char *comm_def                // DSHDR2ﾄﾞﾗｲﾊﾞｰ用通信環境定義ファイル名
);
```

本関数で、通信制御したい装置数（装置、ホスト含めた合計）と DSHDR2 HSMS 通信ドライバーで使用する通信環境定義ファイル名を指定します。

GemSetupLibrary()関数は、以下の処理を行います。

（1）DSHGEM-LIB 内に max_eq 台の装置を制御するためのメモリなどの資源を確保します。
①装置管理情報用メモリの確保（実体は GemSetupEquipment()で設定されます。）
②DSHDR2 通信ドライバーの起動を行います。

（2）DSHDR2 通信ドライバーの起動は、comm_def で指定された通信環境情報に基づいて行います。
ここでは、ドライバーを起動するだけで、ポート、デバイスでの通信に対する制御は行われません。ポート、デバイスの通信開始は、7.1.2 の装置の起動時に行われます。

７．１．２　GemSetupEquipment()による装置制御の起動（初期化）処理

装置 ID 別に装置制御のための起動処理を行い、GemSetupEquipment()関数を使用します。
（装置側もホスト側の制御も同じです。）

```
API int  APIX GemSetupEquipment (
    int  eqid,                      // 通信対象装置 ID(0,1,2,...)
    char *conf_file,                // 装置起動ファイル名
    int  restore_bkup,              // ﾊﾞｯｸｱｯﾌﾟ情報復元指定ﾌﾗｸﾞ
    char *err_str                   // ｴﾗｰ発生時のｴﾗｰ情報領域ﾎﾟｲﾝﾀ
);
```

eqid は、起動したい装置の ID です。装置 ID は０から始まります。

conf_file は、３．で述べた装置の起動のためのコマンドが書き記されているファイルの名前を指定します。

restore_bkup は、前にバックアップされた装置管理情報を復元させるかどうかを指定します。=1 の場合、復元し、=0 の場合は復元しない指定になります。

err_str は、起動処理中にエラーを検出した際、その理由などのメッセージをセットするためのバッファです。

GemSetupEquipment()関数によって次のような処理が行われます。

（１）装置制御に必要な制御ブロックのためのメモリを準備します。
conf_file ファイル内に情報の最大管理数が指定されているものについては、指定された管理数分の領域を準備します。（ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ、ｷｬﾘｱ、基板、CJ、PJ 情報など）

（２）conf_file 内に指定されている装置管理情報定義ファイル名から装置管理のための定義情報を読出し、（１）で準備した情報領域内に登録します。

（３）restore_bkup フラグがセットされている場合には、バックアップファイルの有無と内容が正しいかどうかのチェックを行い、正しければ、バックアップファイルの情報を装置管理情報領域に復元します。
チェックで１個でも内容が正しくなかった場合、エラーを検出したとし、= 5 を返します。この場合、全てのバックアップ情報の復元を行いません。（この場合、restore_bkup フラグ=0 にして起動してください。）

（４）次に、当該装置に与えられた DSHDR2 のポートとデバイスを開きます。
conf_file ファイル内に指定されたポートとでデバイスです。

７．１．３　予約変数の登録

GemSetupEquipment()関数による装置の起動処理が正常に終った後、ユーザは、DSHGEM-LIB で予約されている次の装置管理情報の登録を行う必要があります。
予約する必要のある情報として、次の３種類のものがあります。

・収集イベント (CE)
・装置定数 (EC)
・装置状態変数 (SV)

（１）予約収集イベント ID の登録
GemSetReservedCeid()関数を使って、次の予約 CEID を登録します。

index 値	マクロ名	収集イベント
0	CEX_RSV_COMMUNICATING	ﾎｽﾄと通信確立時に通知するｲﾍﾞﾝﾄ ID
1	CEX_RSV_SPOOL_END	ｽﾌﾟｰﾙﾃﾞｰﾀ送信終了時に通知するｲﾍﾞﾝﾄ ID
2	CEX_RSV_LIMIT	変数ﾘﾐｯﾄ監視におけるﾎｽﾄ通知用ｲﾍﾞﾝﾄ ID

（２）装置定数(EC)
GemSetReservedEcid()関数を使って、次の予約 EC を登録します。

index 値	マクロ名	装置定数
0	**ECX_RSV_MDLN**	S1F14 で使用する装置ﾓﾃﾞﾙ名
1	**ECX_RSV_SOFTREV**	S1F14 で使用するｿﾌﾄｳｴｱﾚﾋﾞｼﾞｮﾝｺｰﾄﾞ
2	**ECX_RSV_SPOOL_MAX**	最大ｽﾌﾟｰﾙ数
3	**ECX_RSV_SPOOL_OVERWRITE**	ｽﾌﾟｰﾙのﾌﾟｰﾙ数
4	ECX_RSV_INIT_COMMSTATE_	ｴﾝｼﾞﾝ起動時の自動通信 Enable の指定用変数

（３）装置状態変数(SV)
GemSetReservedSvid()関数を使って、次の予約 SV を登録します。

index 値	マクロ名	装置状態変数
0	**SVX_RSV_CLOCK**	ｼｽﾃﾑの日付時刻変数(DSHGEM-LIB が値を更新する)
1	**SVX_RSV_COMMUNICATING**	通信状態
2	**SVX_RSV_SPOOL_STATE**	ｽﾌﾟｰﾙ状態
3	**SVX_RSV_SPOOL_TOTAL**	ｽﾌﾟｰﾙ合計
4	**SVX_RSV_SPOOL_ACTUAL**	実ｽﾌﾟｰﾙ数(貯えられた)
5	**SVX_RSV_SPOOL_STIME**	ｽﾌﾟｰﾙ開始時刻
6	**SVX_RSV_SPOOL_FTIME**	ｽﾌﾟｰﾙ満杯時刻
7	SVX_RSV_LIMIT_V	リミット監視対象変数 ID
8	SVX_RSV_LIMIT_DVVAL	同変数値(文字列)
9	SVX_RSV_LIMIT_ID	同リミット ID
10	SVX_RSV_LIMIT_DIR	同遷移方向

７．１．４　ユーザプログラム処理対象受信１次メッセージの登録

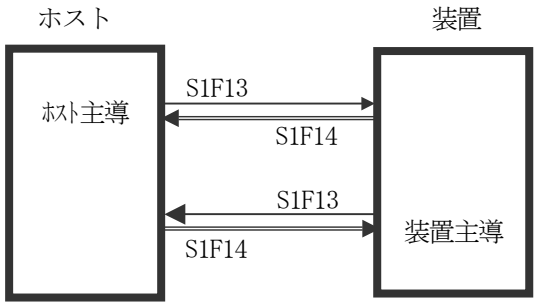
６．で説明したユーザ処理対象１次メッセージの登録を GemRegisterDefaultSxFy () 関数を使って行います。

７．１．５　通信確立の処理

ホスト、装置ともに相手装置との通信を行うためには、S1F13,14 のやり取りによる通信確立のための通信を行う必要があります。

相手装置との通信通信**開始**は、**GemEnable()**関数を呼出して行います

本関数によって通信状態を ENABLED にし､その後の S1F13,14 メッセージのやり取りの成功で COMMUNICATING 状態にします。



どちらの主導でも先に成立したら通信確立です。(S1F14 の Ack=0 の場合)

通信**停止**は、**GemDisable()**関数を使って、装置との通信状態を DISABLED 状態にします。

相手装置との通信確立処理は、DSHGEM-LIB が全て行います。APP は、通信状態変数の参照によって通信が確立しているかどうかを確認することになります。

GemSetReservedSvid()関数を使って SVX_RSV_COMMUNICATING のマクロに対し、状態変数として指定した変数名の値を取得し、判断します。User_def.h ファイルに通信状態変数の値がマクロ定義されています。

```
#define ST_COMM_DISABLED        0
#define ST_COMM_ENABLED         1
#define ST_NOT_COMMUNICATING    2
#define ST_WAIT_CRA             3
#define ST_WAIT_DELAY           4
#define ST_COMMUNICATING        5
```

通信状態が COMMUNICATING 状態でない状態で装置から S1F13 以外の１次メッセージを受信した場合、DSHGEM-LIB は無条件に Function=0 のメッセージを返します。

７．２　通常処理

通常処理については、ユーザがシステムの仕様に基づいてプログラムを設計し、作成しますが、ここでは、シナリオなどを除く一般的な処理について説明します。

７．２．１　変数のアクセス

変数にはEC(装置定数)、SV(装置状態変数)そしてDVVAL(装置ﾃﾞｰﾀ変数)があります。これら変数は装置の起動処理時にDSHGEM-LIB内に登録されます。

ユーザプログラムは、これら変数値の取得、設定操作を次のAPI関数で行います。

	変数	取得関数	設定関数
1	EC	GemSetEcVal()	GemGetEcVal()
2	SV	GemSetSvVal()	GemGetSvcVal()
3	DVVAL	GemSetDvVal()	GemGetDvVal()
4	V (EC,SV,V)	GemSetVVal()	GemGetVVal()

上の表で、4のVの関数は、EC,SV,DVVALのIDがそれぞれユニークである条件で、EC,SV,DVVALのどのIDに対してもアクセスできる関数です。

これらの関数は以下の処理も行います。

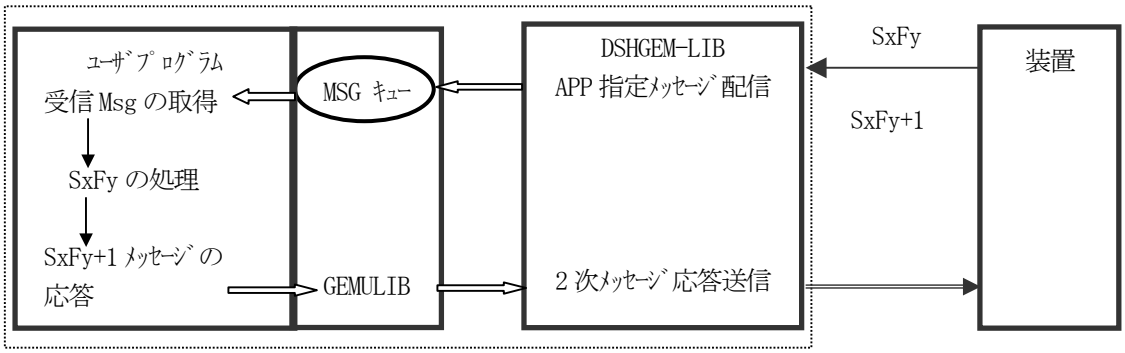
（1）設定値のチェック（最小、最大値による）
変数の設定に関して、もし、その変数に最小値、最大値の指定がある場合、DSHGEM-LIBは設定する際、設定しようとしている値が、この最大値、最小値の範囲内にあるかどうかを調べます。もし、範囲から外れていれば、値を設定しないで、エラーを返却します。
（　最小、最大値の設定は、装置管理情報定義ファイルでの指定、または、GemSet??Min(),
GemSet??Max()関数による指定による。 ??は Ec, Sv, Dv またはV　）

（2）収集イベント(S6F11)の通知（装置側）
設定しようとした変数値が元の値と違っていて、しかも、収集イベント ID(CEID)が指定されていた場合、DSHGEM-LIBは自動的にその指定収集イベントをホストに通知します。

（3）変数値のバックアップ
DSHGEM-LIBは、変数の値が変更になった場合、情報のバックアップ処理を自動的に行います。
バックアップの処理は、装置起動情報ファイルのINFO_BACKUPコマンドが =1 に指定されていた場合に適用されます。DSHGEM-LIBによるバックアップの対象は、変数だけでなくプロセスプログラム、レシピ、コントロールジョブ、プロセスジョブ、キャリア、基板管理情報も含まれます。

７．２．２　受信１次メッセージの処理

７．１．４で登録されたユーザ処理対象１次メッセージの処理の流れは次のようになります。



MSG キューは、受信メッセージが一時的に保存される待ち行列で、装置 ID 単位で個別に設けられます。

受信メッセージの取り出しと応答２次メッセージの送信については、既に述べた６．２を参照ください。

受信メッセージの処理の中では、例えば、ホストが装置からの収集イベント通知、S6F11 を受信した場合を例に説明します。

ユーザプログラムは、受信した S6F11 メッセージが有するレポート ID と変数値を取り出し、必要があれば変数の値を更新する必要があります。

DSHGEM-LIB は、S6F11 のデコード関数 DshDecodeS6F11()を提供していますので、ユーザはこの関数を使って、ユーザが処理しやすい構造体の中に S6F11 に含まれる情報を展開することができます。

DshDecodeS6F11 関数のプロトタイプと使用する構造体は次の通りです。

```
API int APIX DshDecodeS6F11(
    DSHMSG *smsg,                   // SECS ﾒｯｾｰｼﾞ情報構造体のﾎﾟｲﾝﾀ
    TCE_CONTENT *pinfo              // ﾃﾞｺｰﾄﾞした情報を格納する構造体のﾎﾟｲﾝﾀ
);

typedef struct{
        TCEID       ceid;           // 収集ｲﾍﾞﾝﾄ ID
        int         rp_count;       // ﾘﾝｸされているﾚﾎﾟｰﾄ数
        TRP_CONTENT **rp_list;      // ﾘﾝｸﾚﾎﾟｰﾄ情報のﾘｽﾄ
        int         next_rp;        // (関数内部使用)
        int         next_v;         // (関数内部使用)
        } TCE_CONTENT;

typedef struct{
        TRPID       rpid;           // ﾘﾝｸされているﾚﾎﾟｰﾄ ID
        int         v_count;        // ﾚﾎﾟｰﾄにﾘﾝｸされている変数の数
        TV_CONTENT  **v_list;       // ﾘﾝｸ変数情報のﾘｽﾄ
        } TRP_CONTENT;

typedef struct{
```

```
    TVID        vid;            // ﾘﾝｸされている変数 ID
    int         format;         // 変数のﾌｫｰﾏｯﾄ
    int         asize;          // 変数ﾃﾞｰﾀの配列ｻｲｽﾞ
    void        *dptr;          // 変数ﾃﾞｰﾀのﾎﾟｲﾝﾀ   (fmt L ならばnesting ->TV_CONTENT)
    } TV_CONTENT;
```

TCE_CONTENT 内にデコードした情報を、rp_list に載っている rp_count 分のレポートにリンクされている変数の値について処理することになります。

例えば、含まれる変数値を全て装置管理情報に反映させるためには次のようにプログラミングします

```
int s6f11_proc( int eqid, DSHMSG *smsg, TMSG_QUEUE_INFO *qmsg )
{
    TCE_CONTENT info;
    TRP_CONTENT rpinfo;
    TV_CONTENT  vinfo;
    int         ackc6 = 0;
    int ei = DshDecodeS6F11( smsg, &info );                    // S6F11 を info 内にﾃﾞｺｰﾄﾞ
    if ( ei >=0 ){
        for ( int i=0; i < info.rp_count && ackc6 == 0; i++ ){
            rpinfo = *info.rp_list[i];
            ei = GemGetRpName( eqid, rpinfo.rpid, name );      // rpid 有効 ? (名前取得で確認)
            if ( ei < 0 ){ ackc6 = 2; break; }
            for ( int j=0; j < rpinfo.v_count && ackc6 == 0; j++ ){
                vinfo = *rpinfo.v_list[j];
                ei = GemGetVName( eqid, vinfo.vid, name, &n ); // VID 有効 ?
                if ( ei < 0 ){ ackc6 = 2; break; }
                if ( vinfo.format == ICODE_END ) continue      // ICODE_END ならばﾃﾞｰﾀなし
                ei = GemSetVVal( eqid, vinfo.vid, vinfo.dptr, vinfo.asize );  // V 値を設定
                if ( ei < 0 ){ ackc6 = 2; break; }
            }
        }
    }else{
         ackc6 = 2;
    }
    ei = DshResponseS6F12( eqid, qmsg->trid, NULL, ackc6 );    // S6F12 を応答
    DshFreeTCE_CONTENT( &info );                               // info 内に使用されたﾒﾓﾘの開放
    return ackc6;
}
```

DSHGEM-LIB ではこのようなメッセージの内容をユーザプログラム処理を容易にするためのメッセージデコード関数が準備されています。詳しくは、「APP インタフェースライブラリ関数説明書」を参照ください。

７．２．３　１次メッセージの送信

１次メッセージの送信は、各１次メッセージに必要な引数を付けてメッセージ送信用 API 関数を呼出して送信することになります。

どのメッセージ送信関数についても共通に次の引数を有します。

1. eqid　　－ 装置 ID
2. callback　－ コールバック関数（=0 ではコールバックなし）
3. upara　　－ ユーザパラメータ（コールバック関数での呼出関数を同定するための情報）
4. メッセージエンコードに必要な情報（0 個以上）

また、メッセージの送信処理は、相手からの応答メッセージの受信とその処理によって完結しますが、ユーザプログラムにとって送信完了を待つ方法として次の２つの方法があります。

1. 引数 callback = 0 にし、そのままブロックモードで送信完了を待つ方法
2. 引数 callback にコールバック関数を指定して、応答メッセージ受信後、DSHGEM-LIB にそのコールバック関数を呼び出してもらう方法（ﾌﾟﾛｸﾞﾗﾑはﾌﾞﾛｯｸされません）

ユーザプログラムから送信要求される GEM その他の標準的なメッセージについては、DSHGEM-LIB の API 関数として下表の通りの関数が準備されています。関数についての詳細は「APP インタフェースライブラリ関数説明書」を参照ください。

1 次メッセージと API 送信関数一覧表

1 次 MSG	方向	役　割	API 関数名
S1F3	H→E	装置状態要求	GemSendS1F3()
S1F11	H→E	状態変数一覧要求	GemSendS1F11()
S1F15	H→E	ｵﾌﾗｲﾝ要求	GemSendS1F15()
S1F17	H→E	ｵﾝﾗｲﾝ要求	GemSendS1F17()
S2F13	H→E	装置定数要求	GemSendS2F13()
S2F15	H→E	装置定数変更	GemSendS2F15()
S2F23	H→E	ﾄﾚｰｽ条件設定	GemSendS2F23()
S2F29	H→E	装置定数名一覧要求	GemSendS2F29()
S2F31	H→E	時刻設定要求	GemSendS2F31()
S2F33	H→E	ﾚﾎﾟｰﾄ設定	GemSendS2F33()
S2F35	H→E	ｲﾍﾞﾝﾄﾚﾎﾟｰﾄ設定	GemSendS2F35()
S2F37	H→E	ｲﾍﾞﾝﾄﾚﾎﾟｰﾄ有効/無効設定	GemSendS2F37()
S2F41	H→E	ﾎｽﾄｺﾏﾝﾄﾞ送信	GemSendS2F41()
S2F43	H→E	ｽﾌﾟｰﾙの設定	GemSendS2F43()
S2F45	H→E	変数ﾘﾐｯﾄ属性定義	GemSendS2F45()
S2F47	H→E	変数ﾘﾐｯﾄ属性一覧要求	GemSendS2F47()
S2F49	H→E	Enhanced Remote Command	GemSendS2F49()
S3F17	H→E	ｷｬﾘｱｱｸｼｮﾝ要求	GemSendS3F17()
S3F23	H→E	ﾎﾟｰﾄｸﾞﾙｰﾌﾟｱｸｼｮﾝ要求	GemSendS3F23()
S3F25	H→E	ﾎﾟｰﾄｱｸｼｮﾝ要求	GemSendS3F25()
S3F27	H→E	Change Access	GemSendS3F27()

S5F1	H←E	ｱﾗｰﾑ報告	GemNotifyAlarm()
S5F3	H→E	ｱﾗｰﾑ有効/無効設定	GemSendS5F3()
S5F5	H→E	ｱﾗｰﾑﾘｽﾄ要求	GemSendS5F5()
S6F11	H←E	ｲﾍﾞﾝﾄﾚﾎﾟｰﾄ送信	GemNotifyEvent()
S6F13	H←E	注釈付きｲﾍﾞﾝﾄﾚﾎﾟｰﾄ送信	GemAnNotifyEvent()
S6F15	H→E	ｲﾍﾞﾝﾄﾚﾎﾟｰﾄ要求	GemSendS6F15()
S6F19	H→E	個別ﾚﾎﾟｰﾄ要求	GemSendS6F19()
S6F23	H→E	ｽﾌﾟｰﾙﾃﾞｰﾀ要求	GemSendS6F23()
S7F1	H←→E	ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑﾛｰﾄﾞ問合せ	GemSendS7F1()
S7F3	H←→E	ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ送信	GemSendS7F3()
S7F5	H←→E	ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ要求	GemSendS7F5()
S7F17	H→E	ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ削除指示	GemSendS7F15()
S7F19	H→E	ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ一覧要求	GemSendS7F13()
S7F23	H←→E	ﾌｫｰﾏｯﾄ付ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ送信	GemSendS7F23()
S7F25	H←→E	ﾌｫｰﾏｯﾄ付ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ要求	GemSendS7F25()
S7F27	H←E	ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ妥当性送信	GemSendS7F27()
S7F29	H←E	ﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ妥当性問合せ	GemSendS7F29()
S10F1	H←E	端末要求	GemSendS10F1()
S10F3	H→E	端末表示、ｼﾝｸﾞﾙﾌﾞﾛｯｸ	GemSendS10F3()
S10F5	H→E	端末表示、ﾏﾙﾁﾌﾞﾛｯｸ	GemSendS10F5()
S14F9	H→E	Create Object Request(Cj)	GemSendS14F9()
S14F11	H→E	Delete Object Request(Cj)	GemSendS14F11()
S15F1	H←→E	Recipe management m-blk inq	GemSendS15F1()
S15F3	H←→E	Recipe Namespace action Req	GemSendS15F3()
S15F5	H←→E	Recipe Namespace rename Req	GemSendS15F5()
S15F7	H←→E	Recipe Space Request	GemSendS15F7()
S15F9	H←→E	Recipe Status Request	GemSendS15F9()
S15F13	H←→E	Recipe Create Request	GemSendS15F13()
S15F17	H←→E	Recipe Retrieve Req	GemSendS15F17()
S16F5	H→E	Process Job Cmd Request	GemSendS16F5()
S16F11	H→E	PrJob Create Enh	GemSend16F11()
S16F15	H→E	PrJob Multi Create	GemSendS16F15()
S16F17	H→E	PrJob Deque	GemSendS16F17()
S16F19	H→E	Pr Get All Job	GemSendS16F19()
S16F21	H→E	Pr Get Space	GemSendS16F21()
S16F27	H→E	Control Job Command Request	GemSendS16F27()

７．２．４　装置側の収集イベント送信処理について

装置側では、装置状態変数値の変化の要因によってホストに対し収集イベント通知メッセージ、 S6F11 を送信する必要があります。

装置側から収集イベントを通知する手段として２つの方法があります。

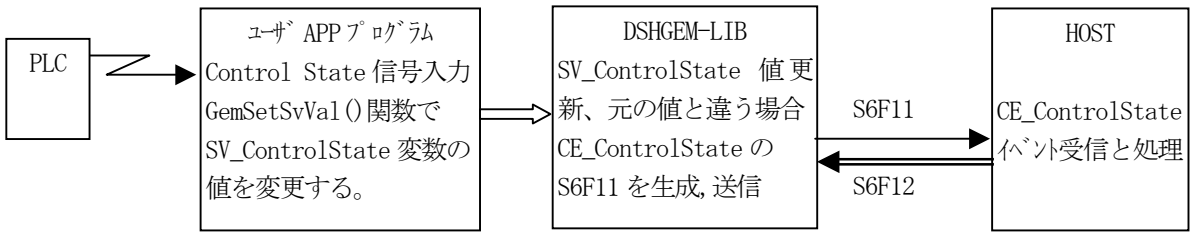
・DSHGEM-LIB が装置状態変数の変化に伴って自動的に収集イベントを通知する。
・ユーザプログラムが GemNotifyEvent()関数を使ってイベントを通知する。

（1）DSHGEM-LIB による自動イベント通知

ユーザプログラムが装置状態変数の値を変更することによって、予めその変数値の変化によって通知する CEID(収集イベント ID)が指定されていた場合、DSHGEM-LIB は、指定されている CEID の定義内容に従って S6F11 メッセージを組み立てホストに対し自動的に送信します。
自動通知の仕組みは次の通りです。
ここでは、仮に状態変数を **SV_ControlState**、通知イベント ID を CE_ControlState として説明します。

①装置管理情報定義ファイル上に予め、状態変数の変化に対する収集イベント ID を指定しておきます。
SV_ControlState の値が変化した際に自動的に CE_ControlState の収集イベントを通知するように指定。しておきます。（定義については、５．１　（３）－①の定義例を参照）

②ユーザが SV_ControlState の値を次の関数で変更します。
GemSetSvVal(eqid, SV_ControlState, &new_control_state);　// new_control_state= 新変数値
この SV 値設定関数によって DSHGEM-LIB は、SV_ControlState の SVID を持つ装置状態変数値の更新を行います。そして、新しい設定値が元の値と違っていた場合、DSHGEM-LIB は、自動的に CE_ControlState のイベント通知を行います。



（2）ユーザプログラムの GemNotifyEvent()関数によるイベント通知

ControlState 信号を入力した後、GemSetSvVal()関数で値を更新し、ユーザプログラムの判断で、GemNotifyEvent()関数を使って収集イベントをホストに通知します。収集イベントに関係する引数としては CEID（収集イベント ID）だけ指定すればよく、メッセージの組み立ては、DSHGEM-LIb が全て行ってくれます。

```
API int APIX GemNotifyEvent( int eqid, TCEID ceid, int (WINAPI *callback)(), ULONG upara);
```

通知が完了する待ち方としては、コールバック指定（する／しない）によってブロックモード、ノンブロックモードを選択することができます。

７．２．５　装置側のアラーム送信処理について

装置側では、人間あるいは装置の安全に関わる危険な状態が発生し、それを検出した際、また、危険な状態が解除された際、ホストに対しアラーム通知メッセージ、 S5F1 を送信する必要があります。

使用する関数は、GemNotifyAlarm()です。

```
API int APIX GemNotifyAlarm( int eqid, TALID alid, int on_off,
                                                    int(WINAPI *callback)(), ULONG upara);
```

引数の中に、on_off がありますが、発生／復旧を１／０で指定します。

装置管理情報の中のアラーム情報の定義において、ce_on, ce_off コマンドで、そのアラームの発生／復旧時に通知する収集イベント ID が指定されていれば、DSHGEM-LIB は、アラームと同時に収集イベントの通知(S6F11 の送信)も自動的に行います。

通知が完了する待ち方としては、コールバック指定（する／しない）によってブロックモード、ノンブロックモードを選択することができます。

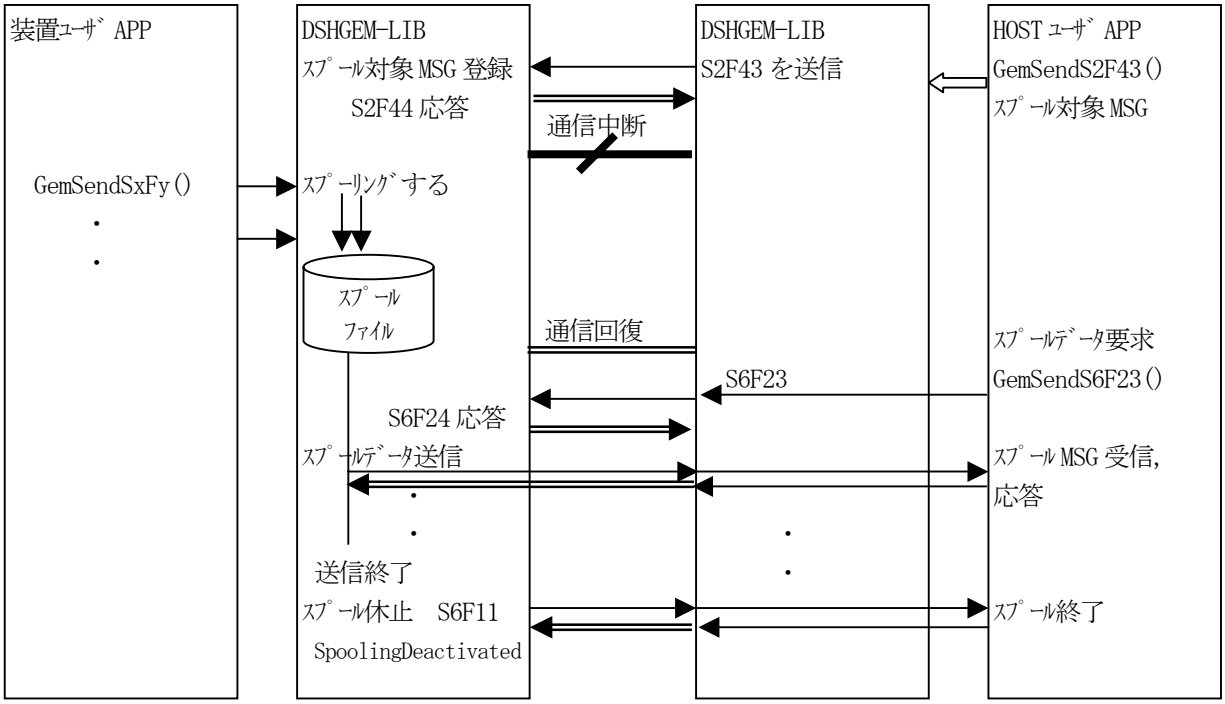
７．２．６　装置側のスプール機能処理について

スプール処理は、装置側での処理であり、通信が中断した後、装置側からホストへ送信しょうとしたメッセージの中、スプール対象に指定されたメッセージをディスクメモリに一時的に保存し、通信が復旧した際に、ホストの指示に従ってスプールしたメッセージをホストに送信するものです。

ホストからのスプール対象メッセージの指定は、S2F43 メッセージで与えられます。DSHGEM-LIB は受信した S2F43 メッセージの内容に従ってスプールの準備をします。（ユーザプログラムは関知する必要がありません。）

本 DSHGEM-LIB では、スプール対象メッセージを装置管理情報定義ファイルの中で指定することができます。そして、ユーザからの API 関数によってスプール対象メッセージの登録あるいは取り消しを行うこともできます。

スプール処理の流れは次のようになります。



このチャートで判りますように、装置側でのほとんどのスプール処理は、DSHGEM-LIB が自動的に行ってくれます。

７．２．７　装置側のトレース機能処理について

ホストはS2F23メッセージを使って装置に対し指定した装置状態変数を周期的にトレース監視し、その結果をS6F1メッセージで報告さことができます。これは状態変数の値をサンプリングするための機能です。

ホストは1個以上の状態変数を、指定したレコードサイズ単位で指定合計数だけ指定周期でサンプリングするように指示できます。

S2F23に含まれるトレース条件は次のとおりです。

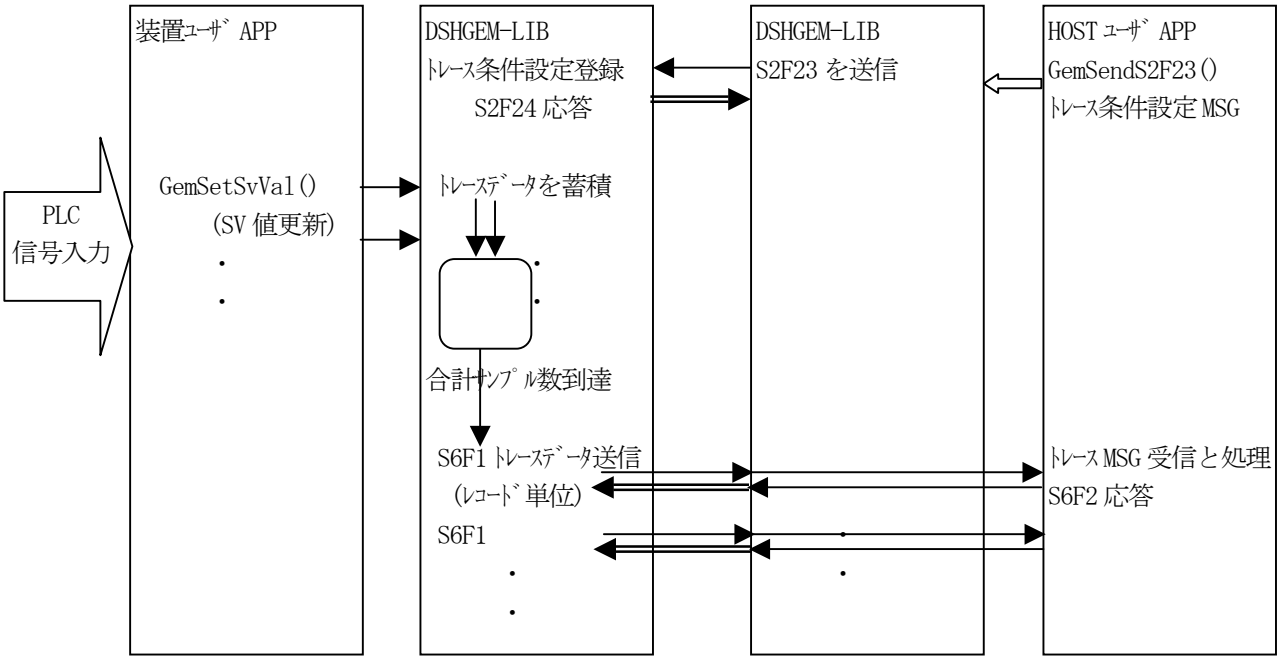
dsper　：周期(sec)　　totsmp：合計ｻﾝﾌﾟﾙ数

repgsz：ｸﾞﾙｰﾌﾟﾚｺｰﾄﾞｻｲｽﾞ　　svid　：装置変数(1個以上)

dsper時間間隔でｻﾝﾌﾟﾘﾝｸﾞし、repgsz回毎にS6F1で結果をﾎｽﾄに報告する。

ｻﾝﾌﾟﾘﾝｸﾞ回数がtotsmpに達したら、最後の情報を送信し、ﾄﾚｰｽを終了します。

処理の流れは概略下図のようになります。



ユーザプログラムはPLCからの入力信号の入力などを行い、装置状態変数の値をGemSetSvVal()で設定します。

DSHGEM-LIBは、ホストからのトレース指示S2F23に従ってトレースデータを収集しトレースデータ送信の条件を満たしたらホストにトレースデータをS6F1メッセージにエンコードし送信します。この場合、S2F23に対するトレース条件の処理、トレースデータの収集ならびにトレースデータの送信をDSHGEM-LIBが行います。

ユーザプログラムがS2F23、S6F1メッセージの処理を直接行う場合は、S2F23メッセージをユーザが処理するメッセージとしてDSHGEM-LIBに登録し、S2F23受信に基づき処理することになります。トレース条件の設定のためのAPI関数も準備されています。

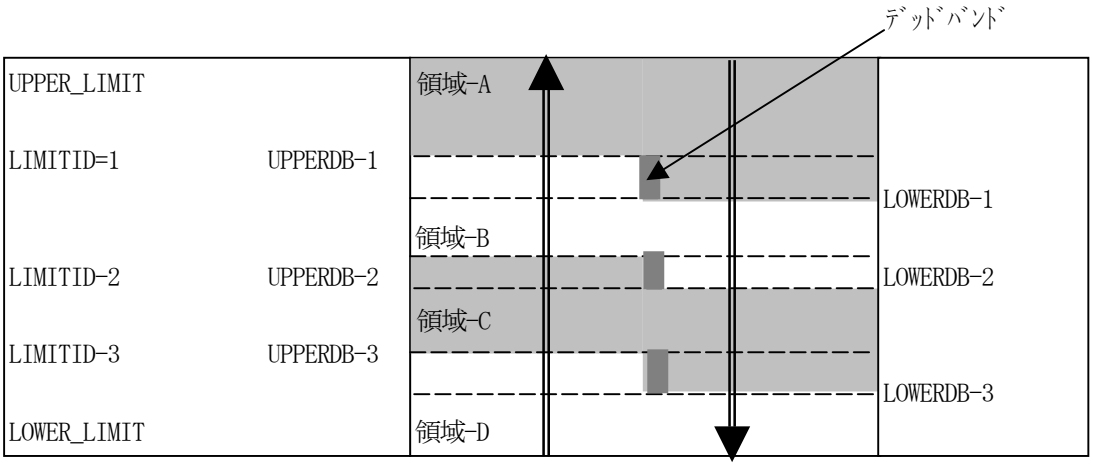
この場合、S2F23のトレース条件をDSHGEM-LIBに登録した後、GemEnableTrace()関数の実行でトレース処理の開始をDSHGEM-LIBに指示します。そして、DSHGEM-LIBがトレースデータの収集とS6F1の送信を行うことになります。

７．２．８　リミット監視とホスト通知処理について

監視対象の変数の値が、ある領域から別の領域へ遷移する際に、領域の境界にUPPERDB と LOWERDB で与えられたデッドバンドを抜け出したときにイベントを通知し、ホストに伝える機能が変数リミット監視機能です。

変数値の領域は、最大８つに分割できます。これにより設けることができるデッドバンドは最大７つになります。

下の図は、４つの領域に分割されたケースです。（３つの LIMITID が存在します。）



例えば、領域-A について説明すると、UPPERDB-1 と LOWERDB-1 で挟まれた部分が領域-A と他の領域間を遷移する際のデッドバンドになります。そして、次のような変数値の変化による遷移でイベントを発生させます。

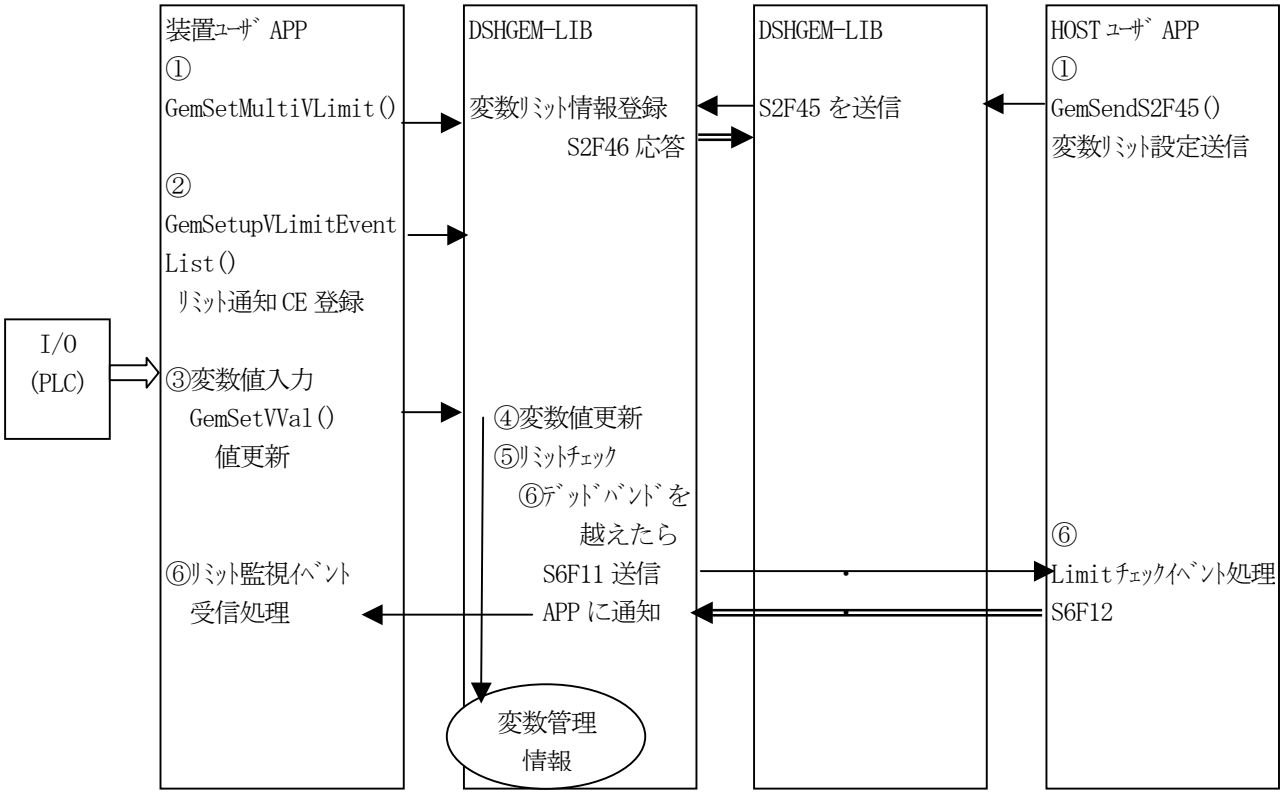
①変数値が 領域-A以外の領域からUPPERDB-1の値以上になり、領域-Aに遷移したときにイベントを発生させます。
②変数値が領域-A から LOWERDB-1 の値以下になり、他の領域に遷移したときにイベントを発生させます。

UPPER_LIMIT、LOWER_LIMT は、それぞれ、その変数に与えられた MAX、MIN の値と同じです。
領域-B,C,D についても領域-A と同様になります。

また、DSHGEM-LIB では、**リミット監視イベント**をホストに通知するだけではなく、**DSHGEM-LIB のアプリケーションプログラムに対しても通知する機能**を提供します。

次ページに処理の流れを示します。

リミット監視機能については、**「変数リミット監視機能　説明書」**を参照してください。



① リミット監視を行なう変数に対する設定には２つの手段があります。
ホストからの設定　：　S2F45 メッセージを装置に送る。
装置自身の設定　　：　GsmSetMultiVLimit()関数を実行する。

② 装置側 APP がリミット監視処理によってデッドバンドを越えたときにイベントを期待するための設定です。
装置側 APP のオプションです。
変数値の遷移がデッドバンドを越えたときに、Windows OS のイベントを APP 側に通知してもらうための設定です。

③ 装置側 APP が入力信号の値を入力し、DSHGEMLIB の管理情報を更新します。
GemSetVVal()関数を使用して行ないます。

④ DSHGEM-LIB は GemSetVVal()関数を実行します。
指定された変数の値を更新するとともに、リミット監視処理の指定があった場合、今回の値と前回の値を比較して値が相違し、かつ、デッドバンドを超える遷移があった場合に、⑤の処理を行います。

⑤ DSHGEM-LIB は、以下の２つの処理を行います。
ホストに対し、リミット監視の結果デッドバンド遷移のイベントを S6F11 を使って送信します。
イベント ID はについては、予め、CE 予約インデクス =CEX_RSV_LIMIT を使って登録しておく必要があります。（７．１．３参照）

装置側 APP に対しては、②の設定が行なわれていれば、Windows API SetEvent()関数を使って、イベントを通知します。

⑥ 装置側 APP, ホスト APP 側では、⑤に対する処理を行います。

７．３　終了処理

終了処理には、装置の終了と DSHGEM-LIB ライブラリの終了処理があります。

装置の終了、DSHGEM-LIB の終了の順に行います。

７．３．１　装置の終了処理

７．１．２において装置 ID 指定して GemSetupEquipment()関数で起動した装置の処理を終了させるための処理です。

装置終了は、次の GemTerminateEquipment()関数を使って行います。

```
API void APIX GemTerminateEquipment( int eqid );
```

本関数は、装置処理のために使用したメモリ、管理情報などをシステムに返却するとともに装置処理のために生成されていたスレッドの終了も行います。また、装置が使用していた DSHDR2 通信ドライバーのポートとデバイスの停止も行います。

eqid で指定された装置の処理が終了した後、GemTerminateEquipment()関数から戻ってきます。

７．３．２　DSHGEM-LIB の終了処理

７．１．１において GemSetupLibrary()関数で起動したライブラリの処理を全て終了させます。

装置終了は、次の GemTerminateLibrary()関数を使って行います。

```
API void APIX GemTerminateLibrary( void );
```

本関数は、DSHGEM-LIB ライブラリ処理のために使用した管理領域のためのメモリの開放、処理のために生成されていたスレッドの終了などを行います。それから DSHDR2 通信ドライバーの停止も行います。

また、まだ、終了していない装置が残っていた場合には、それら終了していない装置を全て強制的に終了させます。

全終了処理が終了した後、GemTerminateLibrary()関数から戻ってきます。

＜付録＞

付録－Ａ　DSHGEM-LIB 通信制御エンジンライブラリ関連文書一覧表

付録－Ａ１　DSHGEM-LIB 通信制御エンジンライブラリ関連文書一覧表

#	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-07-30308-00	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 でも使用可能です。
7	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。
8	DSHGEM-LIB-07-30320-00 DSHGEM-LIB-07-30321-00 DSHGEM-LIB-07-30322-00 DSHGEM-LIB-07-30323-00 DSHGEM-LIB-07-30323-00	DSHGEM-LIB APIﾗｲﾌﾞﾗﾘ関数説明書 VOL-1 VOL-2 VOL-3 VOL-4 VOL-5	ﾕｰｻﾞが使用できる API 関数とﾗｲﾌﾞﾗﾘ関数の説明書です。
9	DSHGEM-LIB-07-30340-00	ﾕｰｻﾞ作成ﾗｲﾌﾞﾗﾘ関数 次ﾒｯｾｰｼﾞ応答関数一覧表	
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	
11	DSHGEM-LIB-07-60101-01	DSHENG3 から DSHGEM-LIB への互換性のために	
12	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。

付録－Ａ２　デモプログラム関連文書一覧表

#	文書番号	文書名	注釈
1	DSHGEM-LIB-07-30380-00	DSHGEM-LIBﾃﾞﾓﾌﾟﾛｸﾞﾗﾑ 外部仕様書	ﾃﾞﾓﾌﾟﾛｸﾞﾗﾑの機能仕様書です。
2	DSHGEM-LIB-07-30381-00	DSHGEM-LIB ﾎｽﾄﾃﾞﾓﾌﾟﾛｸﾞﾗﾑ説明書 (Visual C++)	ﾕｰｻﾞ APP ﾌﾟﾛｸﾞﾗﾑのﾎｽﾄ側ﾃﾞﾓﾌﾟﾛｸﾞﾗﾑです。
3	DSHGEM-LIB-07-30382-00	DSHGEM-LIB 装置ﾃﾞﾓﾌﾟﾛｸﾞﾗﾑ説明書 (Visual C++)	ﾕｰｻﾞ APP ﾌﾟﾛｸﾞﾗﾑの装置側ﾃﾞﾓﾌﾟﾛｸﾞﾗﾑです。
4	DSHGEM-LIB-07-30383-00	ﾃﾞﾓ・ﾌﾟﾛｸﾞﾗﾑｾｯﾄｱｯﾌﾟｶﾞｲﾄﾞ (Windows 用)	
5	DSHGEM-LIB-07-30384-00	DSHGEM-LIBﾃﾞﾓﾌﾟﾛｸﾞﾗﾑ With DSHPLCSim PLC ｼﾐｭﾚｰﾀ 外部仕様書	PLC ｼﾐｭﾚｰﾀ使用する ﾃﾞﾓﾌﾟﾛｸﾞﾗﾑの機能仕様書です。
6	DSHGEM-LIB-07-30385-00	DSHGEM-LIB ｿﾌﾄｳｪｱ/ﾊﾟｯｹｰｼﾞ ﾎｽﾄ － 装置 － PLC 通信ﾃﾞﾓﾌﾟﾛｸﾞﾗﾑ説明書	PLC ｼﾐｭﾚｰﾀ使用する ﾃﾞﾓﾌﾟﾛｸﾞﾗﾑの説明書です。
7	DSHGEM-LIB-07-30386-00	DSHGEM-LIB ﾃﾞﾓ・ﾌﾟﾛｸﾞﾗﾑ with PLC ｼﾐｭﾚｰﾀ ｾｯﾄｱｯﾌﾟｶﾞｲﾄﾞ (Windows 用)	setup.exe のｲﾝｽﾄｰﾙ操作説明書です。
8	DSHGEM-LIB-07-30387-00	ﾃﾞﾓ・ﾌﾟﾛｸﾞﾗﾑ説明書 C 言語(Windows, Linux)	
9	DSHGEM-LIB-07-30388-00	ﾃﾞﾓ・ﾌﾟﾛｸﾞﾗﾑｾｯﾄｱｯﾌﾟｶﾞｲﾄﾞ (Redhat Linux 用)	

付録－Ａ３　DSPLCSIM プログラム関連文書一覧表

#	文書番号	文書名	注釈
1	DSHPLCSIM-08-40300-00	DSHPLCSim ﾕｰｻﾞｰｽﾞ・ｶﾞｲﾄﾞ	DSHPLCSIM の機能説明書です。
2	DSHPLCSIM-08-40301-00	DSHPLCSim 操作ﾏﾆｭｱﾙ	操作についての具体的な説明書です。
3	DSHPLCSIM-08-40320-00	DSHPLCSim ｲﾝﾀﾌｪｰｽ関数説明書	装置ｺﾝﾄﾛｰﾗ側が使用できるｲﾝﾀﾌｪｰｽ関数の説明書です。
4	DSHPLCSIM-08-40380-00	DSHPLCSim PLC 制御デモ・プログラムについて (VB6.0 言語版)	VB6 のﾃﾞﾓﾌﾟﾛｸﾞﾗﾑの説明書です。
5	DSHPLCSIM-08-40381-00	DSHPLCSim PLC 制御デモ・プログラムについて (VC++6.0 言語版)	Visual C++のﾃﾞﾓﾌﾟﾛｸﾞﾗﾑの説明書です。

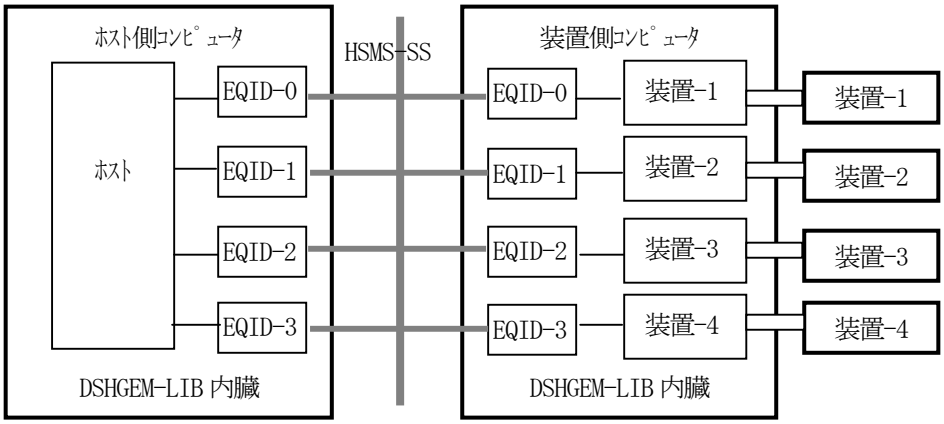
付録－Ｂ　対応できるシステム構成

以下、DSHGEM-LIB が適応できるシステム構成、SEMI スタンダードへの対応、機能等について概要を記述します。

DSHGEM-LIB は、複数の装置、ホストに対応でき、最大８台の装置＋ホストの制御を行うことができます。

装置、ホストの組み合わせは自由です。また、管理情報はそれぞれの装置で同じである必要はありません。

単純な構成例ですが、ホスト１台、装置４台の制御の場合、次のようになります。

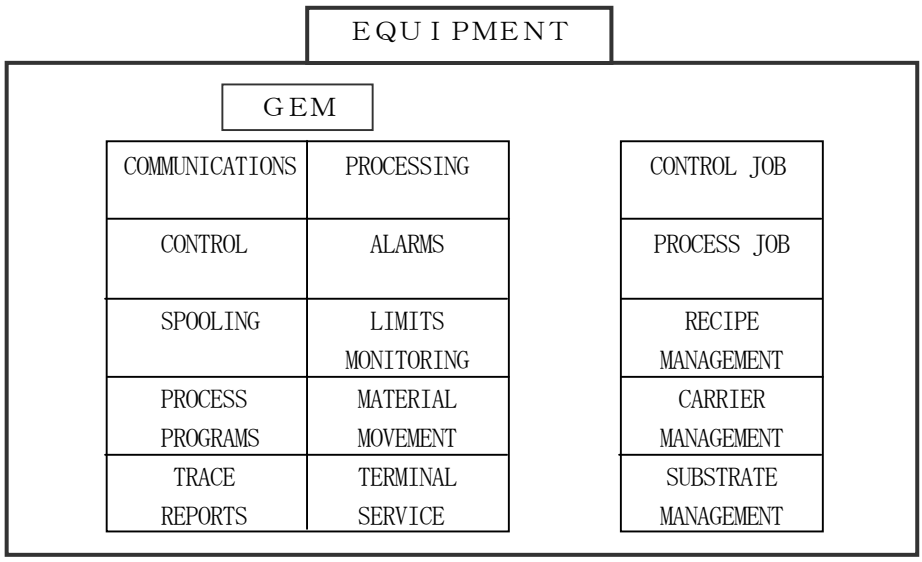


付録－C　SEMI スタンダードへの対応

通信プロトコルは、SECS-I ならびに HSMS-SS をサポートします。

SECS-II メッセージのエンコード、デコードと装置管理情報（変数など）の管理を行います。

SEMI スタンダードが提唱する GEM ならびに CJ, PJ, RMS, CMS、STS に関するスタンダードをサポートします。



GEM スタンダードに含まれない製造工場特有なメッセージの送受信も可能なように DSHGEM-LIB はユーザが SECS-II メッセージの送受信をするための API 関数も準備されています。

付録－Ｄ　装置管理情報とバックアップファイル

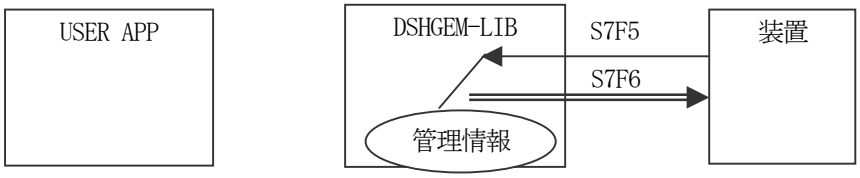
（1）装置固有管理情報（変数、収集イベント、アラームなど）の定義ファイルによる登録
変数、収集イベント情報などについて、名前と ID、それに付随するパラメータ（フォーマット、値、リンク情報など）の情報をテキストファイル上に DSHGEM-LIB が定める書式で定義します。これは装置別に作成したものをシステムに登録します。
専用の管理情報編集ソフトウェアツール、DSHGEMSET.EXE プログラムを使って編集可能です。

（2）装置管理情報のアクセス機能ならびにバックアップ機能
（1）で定義し、DSHGEM-LIB 内に登録された装置管理情報に対するアクセス機能が準備されています。ID をキーにした値の取得、更新設定などのアクセスを SHGEM-LIB が提供する API 関数を使って自由に行うことができます。
また、DSHGEM-LIB は、装置管理情報をファイルに随時バックアップします。このバックアップファイルは終了後、次回の開始時に復元し、前回の終了時の情報を継続して使用することを可能にします。

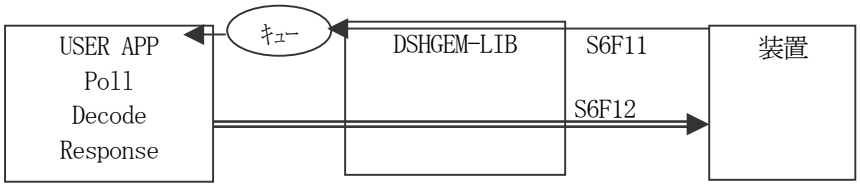
付録－E　SECS-II の通信制御処理

DSHGEM-LIB は、装置間の SECS-II メッセージ通信処理について、ユーザができる限りシンプルに通信プログラミングできるような仕組みと手段を提供します。

① DSHGEM-LIB が自身で対応できる受信１次メッセージに対しユーザの手を煩わすことなく自動的に処理します。



② APP が処理したい受信メッセージの場合、DSHGEM-LIB が受信キューを通して APP に渡します。
DSHGEM-LIB は受信キューをポーリングするための関数を提供します。また、メッセージ内の情報を、プログラムが処理しやすいように、メッセージ専用の構造体にデコードするための関数も提供します。これによってユーザは SECS-II メッセージの構造を意識する必要がありません。また、応答メッセージの送信も応答情報を構造体に詰めるための関数ならびに応答するための関数を提供します。



③ ホストまたは装置が送信する S3F17, S7F3 メッセージなどの１次メッセージの送信には、専用 API 関数を使って簡単に送信できます。勿論、ユーザ自身で任意のメッセージを組み立て送信することも可能です。また、各メッセージ専用の組み立て（エンコード）用関数も準備されています。

メッセージの送信については、通信のトランザクションの完了を待機する方法として、プログラムの制御をそのままブロックして待つ方法と、プログラムの制御を完了を待たずに進め、完了を指定したコールバック関数を呼び出させる方法の２つの方法があります。