

DSHEng4 装置通信エンジン (GEM+GEM300)
ソフトウェア・パッケージ

DSHEng4 への手引き

2011年4月 (改-1)

株式会社データマップ

[取り扱い注意]

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

【改訂履歴】

番号	改訂日付	項目	概略
1.	2009. 7	初版	
2.	2011. 4	DSHEng4Class 関連のコメント	本説明書は、DSHEng4 エンジンの API 関数をベースにした記述です。その後、DSHEng4Class .Net 言語を使用したクラス・ライブラリが使用することができます。
3.			
4.			

目 次

1. 概要	1
2. 作業の流れ	2
3. 通信環境定義ファイル(COMM. DEF)の作成	3
4. 装置起動情報定義ファイルの作成	5
5. 装置管理情報定義ファイルの作成とコンパイル	7
5. 1 装置管理情報の定義と定義ファイルの作成	8
5. 2 装置管理情報のコンパイル	11
5. 3 コンパイルによって生成されたヘッダーファイル例	12
6. ユーザ作成 DLL (DSHEng4U) プログラムの準備	13
6. 1 ユーザ処理対象1次メッセージの登録	13
6. 2 受信1次メッセージ処理後の応答メッセージ送信	15
7. アプリケーションプログラムの作成	17
7. 1 初期化处理	18
7. 1. 1 EngStart()によるDSHEng4の起動(初期化)処理	18
7. 1. 2 予約変数の登録	19
7. 1. 3 ユーザプログラム処理対象受信1次メッセージの登録	20
7. 1. 4 通信確立の処理	20
7. 2 通常処理	21
7. 2. 1 変数のアクセス	21
7. 2. 2 受信1次メッセージの処理	22
7. 2. 3 1次メッセージの送信	24
7. 2. 4 収集イベント送信処理について	25
7. 2. 5 アラーム送信処理について	26
7. 2. 6 スプール機能処理について	27
7. 2. 7 トレース機能処理について	28
7. 2. 8 リミット監視とホスト通知処理について	29
7. 3 終了処理	31
<付録>	32
付録-A DSHEng4 通信制御エンジンライブラリ関連文書一覧表	32
付録-A 1 DSHEng4 通信制御エンジンライブラリ関連文書一覧表	32
付録-A 2 DSHEng4-CLASS クラス・ライブラリ関連文書一覧表	32
付録-A 3 DSHEng4 エンジン・API 関数ライブラリ関連文書一覧表	33
付録-A 4 DSHEng4 エンジン・HSMS 通信ドライバー関連文書一覧表	33
付録-A 5 DSHENG4 エンジン・デモ・プログラム関連文書一覧表	33
付録-B SEMI スタンダードへの対応	34
付録-C 装置管理情報とバックアップファイル	35
付録-D SECS-II の通信制御処理	36

1. 概要

本説明書は、ユーザが GEM、GEM300 サポート、DSHEng4 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、DSHEng4 を理解し、さらに、システム的设计、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、DSHEng4 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、DSHEng4 エンジンが提供する API 関数呼び出しを使つ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 DSHEng4Class 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。

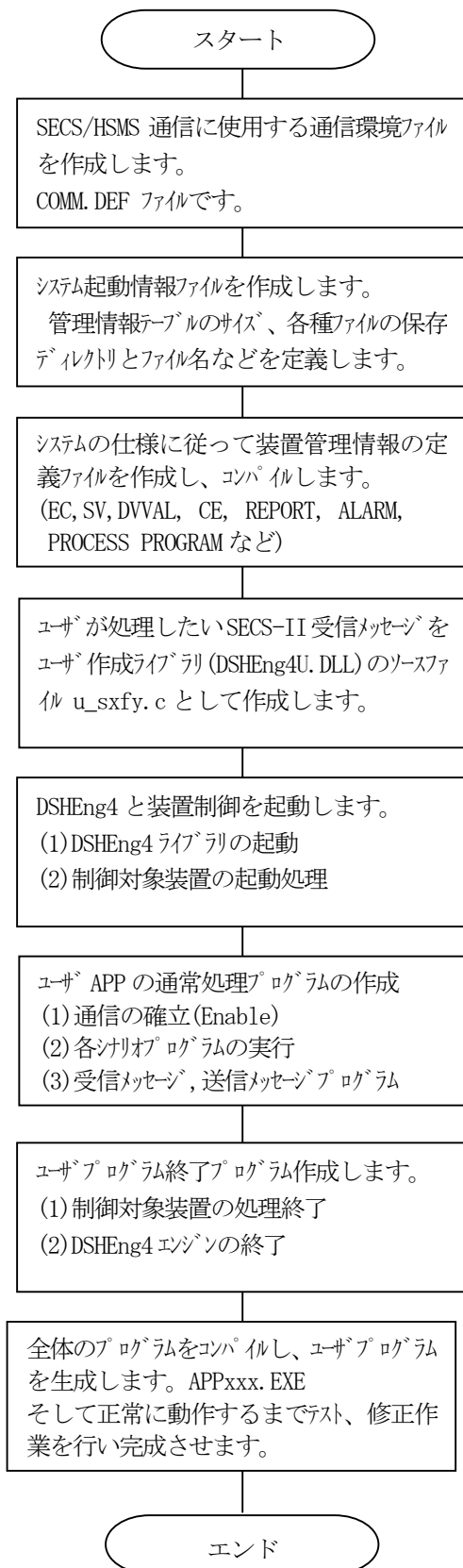
DSHEng4Class クラス・ライブラリについては、下記資料を参照ください。

DSHEng4CLASS クラス・ライブラリ関連文書一覧表

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

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ーが行うための作業の流れを表します。



「DSHDR2 SECS/HSMS レベル-2 通信制御ドライバ-ユーザーズマニュアル」参照
装置が通信で使用する PORT, DEVICE の通信パラメータを COMM. DEF ファイル上に定義します。

DSHEng4 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。

「DSHEng4 装置管理情報定義仕様書」参照
編集プログラム、DSHGEMSET. EXE で作成編集可能
「装置管理情報定義ファイルコンパイル説明書」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パイルしオブジェクトファイルを生成します。

「DSHEng4 通信制御エンジンライブラリ (SECS/HSMS) ユーザーズガイド」 6. ユーザー作成 DSHEng4U. DLL プログ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ジ u_s???f???c ファイルを準備します。
そして **DSHEng4U. DLL** を生成します。

以下、ユーザー APP プログラムを作成します。
DSHEng4 内部情報と装置処理起動のための初期化処理です。
EngStart() 関数を使用します。

DSHEng4 API ならびにライブラリ関数を使ってプログラミングします。
通信の確立を行い、シリオの実行処理を行うためのプログラムです。
デモプログラム (Visual C++) を参考にしてください。

EngStop() 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 EngStart() 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          # 最大メッセージサイズ (バイト数)          1MB+40
  MAX_TRANSACTION = 512           # 管理する最大トランザクション数 2 の n 乗の値
  LOG_LINE = 100000              # 1 個のログファイルに保存する最大行数(
  LOG_FILE = ¥dsheng4¥log¥DSHDR2.LOG # ログファイルを保存するファイル名 (フルパス名)
  LOG_MODE = 0                   # save old log file 4 世代分のログファイルを残す
  MON_PORT = 9999                # 通信ログモニタープログラムとの TCP/IP 接続ポートを指定
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 のポート番号
  PROTOCOL = HSMS                 # HSMS-SS を指定
  PORT_MODE = ACTIVE              # Active ポート (Passive ポートでは Passive)
  TCP_PORT = 5001                 # このポートが使用する TCP/IP のポート
  IP = 192.168.1.2                # 接続相手の Passive ポートの IP アドレス
  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

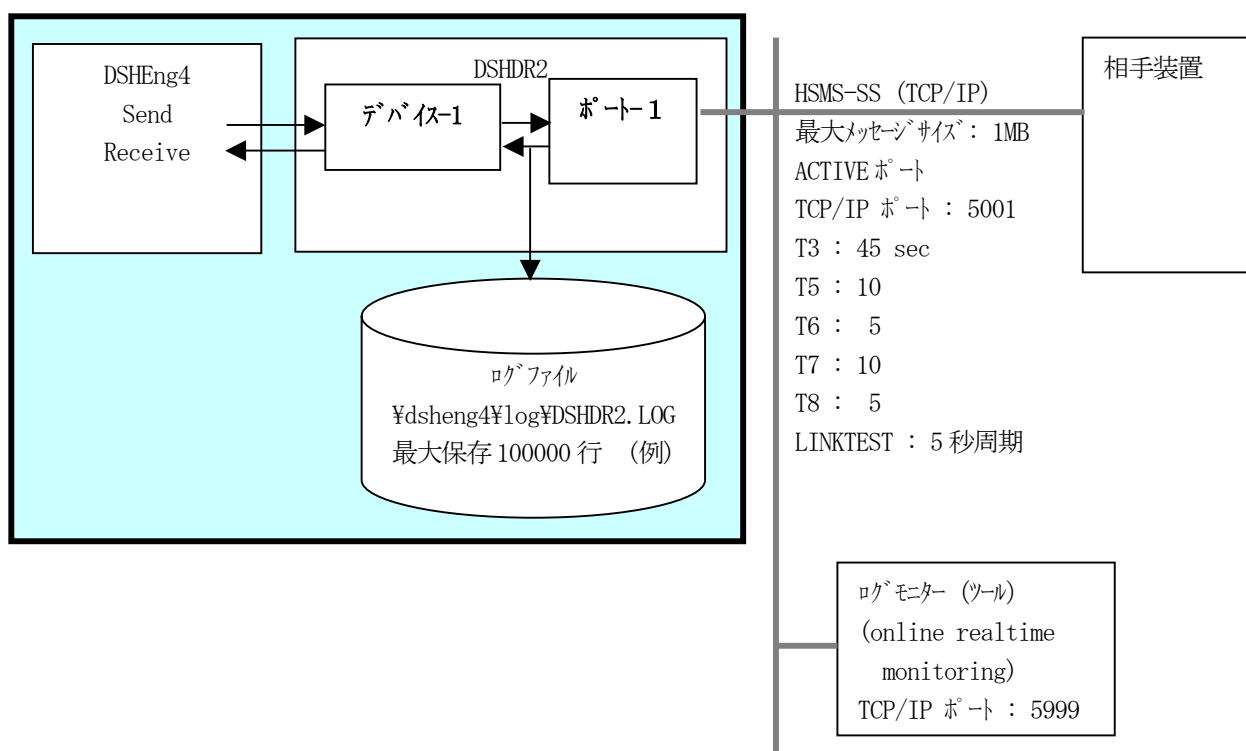
DSHDR2 通信ドライバー上における通信デバイスの番号とそのパラメータを定義します。

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                # デバイス番号
  DVID = x1234              # デバイス ID (= セッション ID)  頭の x は 16 進表現の意
  PORT = 1                  # using port-1
END
```

DSHeng4 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、DSHEng4 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、「DSHEng4 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きます。DSHGEMSET の機能、操作方法については、「DSHEng4, DSHENG3 起動ファイル、装置管理情報ファイル設定・編集プログラム説明書」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 = <ディレクトリ名>	個別装置のログファイルの保存ディレクトリ名を指定します。	LOG_PATH = c:\dsheng4\log0
2	LOG_FILE = <ファイル名>	個別装置のログファイル名を指定します。	LOG_FILE = “gem_host.log”
3	LOG_SIZE = <行数>	ログ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 = <ディレクトリ名>	装置管理情報のバックアップファイルを保存するディレクトリを指定します。	BKUP_PATH = “C:\DSHEng4\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パスで指定します。	INFO_FILE = “c:\dshgemfil\eqinfo0.fil”
6	INFO_BACKUP = <1/0>	装置管理情報のバックアップを行うかどうかを 1, 0 で指定します。	BACKUP_FLAG = 1
7	SPOOL_PATH = <ディレクトリ名>	SPOOL ファイルを保存するディレクトリを指定します。	SPOOL_PATH = “C:\DSHEng4\spool0”
8	PP_COUNT = <n>	PP (プロセスプログラム) 最大管理数を n 個にします。(S7F3)	PP_COUNT = 64
9	FPP_COUNT = <n>	FPP (書式付プロセスプログラム) 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 (プロジェクト) 最大管理数を n 個にします。	PRJ_COUNT = 16
14	CJ_COUNT = <n>	CJ (コントロールジョブ) 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エハー枚数を n 個にします。	CAR_CAPACITY = 25
17	COMM_SIDE = <サイト名>	通信サイトを HOST/EQUIPMENT で指定します。	COMM_SIDE = EQUIPMENT
18	COMM_PORT = <ポート番号>	DSHDR2 通信ドライバ-で使用するポート番号を指定します。	PORT = 1
19	COMM_DEVICE = <デバイス番号>	DSHDR2 通信ドライバ-で使用するデバ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。

[DSHEng4 用設定画面]

DSHGEM-LIB 装置起動情報設定

ファイル名

番号	コメント	設定値	コメントの意味
1	LOG_PATH	c:\dshgemlib\log0	ログファイル用ディレクトリ
2	LOG_FILE	gem_host.log	ログファイル名
3	LOG_SIZE	100000	ログファイルサイズ(行)
4	BKUP_PATH	c:\dshgemlib\backup0	情報バックアップ用ディレクトリ
5	INFO_FILE	c:\dshgemlib\bin\eqinfo.fil	GEMデータ定義ファイル名
6	INFO_BACKUP	1	GEM情報バックアップ指定(1 / 0)
7	PP_COUNT	100	フームのフーム最大保存数
8	FPP_COUNT	64	書式付フームのフーム最大保存数
9	RCP_COUNT	200	ロット最大保存数
10	CAR_COUNT	80	キャリア最大保存数
11	CAR_CAPACITY	80	キャリア容量(ウェル枚数)
12	SUBST_COUNT	250	基板最大保存数
13	CJ_COUNT	32	コントロールジョブ最大保存数
14	PRJ_COUNT	30	フームジョブ最大保存数
15	TRACE_COUNT	28	トレース情報最大保存数
16	SPOOL_PATH	c:\dshgemlib\spool0	スプール情報保存ディレクトリ
17	COMM_SIDE	HOST	ホスト/装置サイトの指定(HOST/EQUIP)
18	COMM_PORT	1	HSMS通信ポート(DSHDR2のポート)
19	COMM_DEVICE	1	HSMS通信デバイス(DSHDR2のデバイス)
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、DSHEng4 で使用できるようにします。

装置管理情報の定義についての詳しい内容は、
「DSHEng4 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、
「DSHEng4, DSHENG3 起動ファイル、装置管理情報ファイル設定・編集プログラム説明書」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。

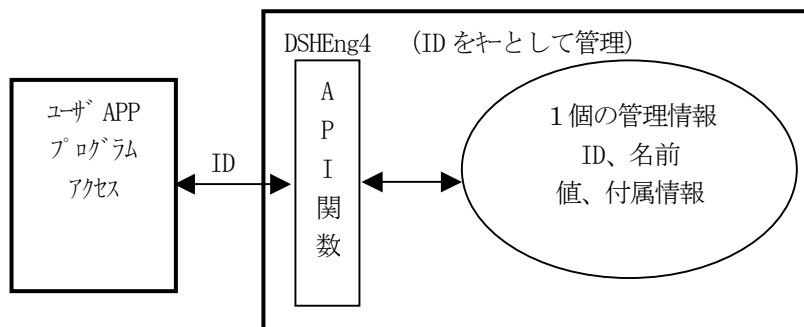
番号	情報区分	情報名	必須は○印	
1	変数	EC 装置定数	○	
		SV 装置状態変数	○	
		DVVAL 装置ゲータ変数	○	
2	収集イベント	CE 収集イベント	○	
		REPORT レポート	○	
3	アラーム	ALARM アラーム	○	
4	プロセス プログラム (レシピ)	PP プロセスプログラム		PP, FPP, RECIPE の 何れかを選択
5		FPP 書式付プロセスプログラム		
6		RECIPE レシピ		
7	トレース	TRACE 状態変数トレース		
8	スプール	SPOOL メッセージスプール		

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DSHEng4 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		準備できたポートポート
1004	SV_AccessModeStatus	0~3	0	U1		ポートアクセスモード状態
.						
.						

(2) レポート ID

CEID	名前	イベント発生タイミング	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	ローダーが搬入可能状態になった	SV_Clock
			SV_LoadPortId
105	RP_PortAccessMode	ポートのアクセスモードが変更になった	SV_Clock
			SV_AccessModeStatus
.			
.			
.			

(3) 収集イベント(CE)

CEID	名前	イベント発生タイミング	リンクレポート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	ローダーが搬入可能状態になった	RP_ReadyToLoad
5	CE_Port1AccessMode	ポート1のアクセスモードが変更になった	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 DSHEng4 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 つの情報例に対する定義は、DSHEng4 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DSHEng4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。

DshCompile.exe の起動書式は次のようになります。

DshCompile.exe <ソースファイル名> <ログファイル名> <エンド通知ファイル名>

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)

ログファイル名 : コンパイルエラーメッセージ、終了時のメッセージなどのログ用ファイル名です。
省略可能です。省略時は dsh_conf.log に記録されます。

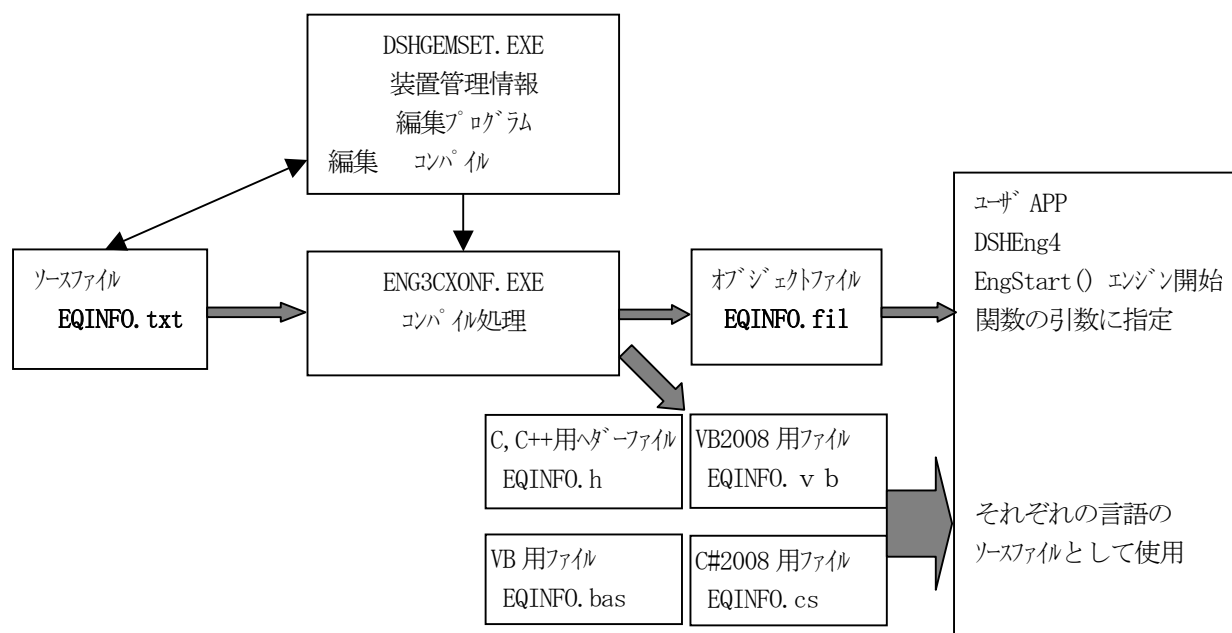
エンド通知ファイル名 : DSHGEMSET.EXE に終了通知するためのファイル名です。
省略してください。

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。DSHEng4 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、EQINFO.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EQINFO.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++、VB6、VB2008、C#2008 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EQINFO.txt ファイルを作成し、その後、DshCompile.exe でコンパイルし、EQINFO.fil を生成します。そして、それをEngStart()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ッダファイル例

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 (DSHEng4U) プログラムの準備

DSHEng4 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 DSHEng4 への登録
- (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、**DSHEng4U.DLL** (以下、**ULIB** と呼びます) とします。

DSHEng4 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

DSHEng4 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 DSHEng4 に対して登録する必要があります。これは、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 EngRegisterDefaultSxFy( int no_of_msg, int ev );
no_of_msg : 登録するメッセージの最大数
ev       : 登録メッセージを受信したときに DSHEng4 で通知してもらいたいイベント ID
           ev=NULL(=0)の場合は、イベント通知しません。
```

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 [1]={
    pri_pro_info_tab_1,    // EQ
};
```

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。ここには装置の受信メッセージが登録されています。

```
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
```

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。

DSHEng4では、基本的に、応答2次メッセージの送信処理は、ULIB（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、ULIB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、EngGetSecsMsgReq()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ドライバ-が与えるトランザクションID
    DSHMSG   *smsg;              // 受信メッセージが格納されているDSHDR2メッセージ構造体ポインタ
    DSHMSG   **prmsg;           // 使用しない。(応答msgのポインタ)
    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、終わったら、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( 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( trid, prmsg, DshAcceptS1F17Callback );
    U_DshFreeMsg( prmsg );
    return 0;
}

API int APIX DshAcceptS1F17Callback( 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(ID_TR trid, DSHMSG *prmsg,
                             int (WINAPI *callback)() );
```

trid : DSHDR2 から受信した 1 次メッセージに与えられたトランザクション ID
prmsg : 受信した 1 次メッセージが格納されている DSHMSG 構造体のポインタ
callback: 応答メッセージ送信終了後に呼び出されるコールバック関数を指定します。

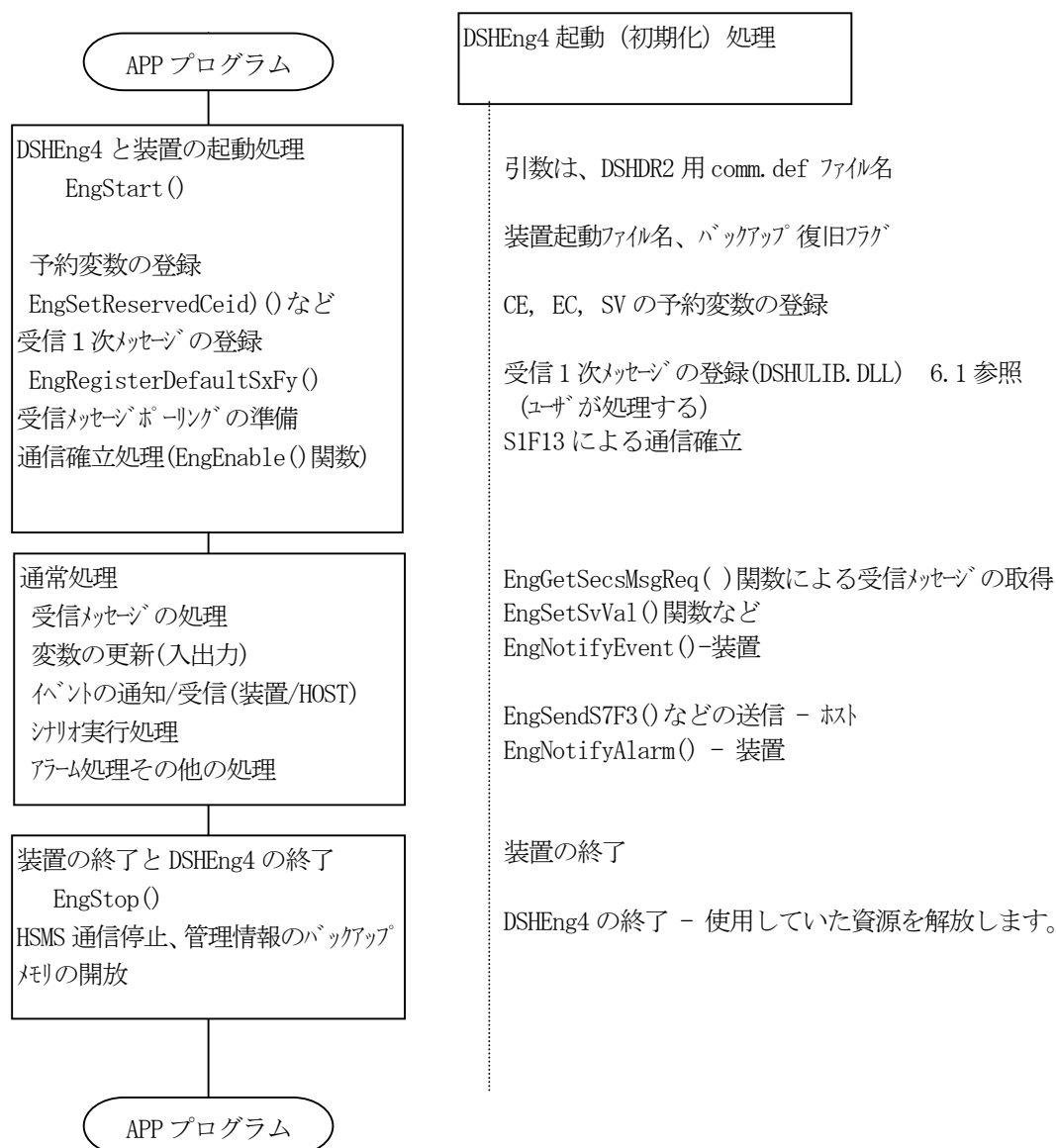
(注) callback 関数の引数には、DSHMSG 構造体のポインタがありますが、DSHMSG 構造体とその内容は、DSHEng4 で新しいメモリを確保し、コピーされていますので、ここで callback 関数に与えられた rmsg のメモリを開放してください。

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、DSHEng4 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) DSHEng4 の初期化処理
 - ・ DSHEng4 と装置制御の初期起動処理
- (2) DSHEng4 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) DSHEng4 の終了処理
 - ・ 装置の終了処理
 - ・ DSHEng4 の終了処理



7.1 初期化処理

最初に、DSHEng4 を起動し、その後、制御する装置を起動します。

7.1.1 EngStart()による DSHEng4 の起動 (初期化) 処理

DSHEng4 の初期設定と起動を EngStart() 関数を使って行います。

```

API int APIX EngStart (
    char *comm_def,           // DSHDR2 ドライバー用通信環境定義ファイル名
    char *conf_file,         // 装置起動ファイル名
    int  restore_bkup,       // バックアップ 情報復元指定フラグ
    char *err_str            // エラー発生時のエラー情報領域ポインタ
);

```

本関数で、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、装置起動ファイル名、バックアップ情報を復元するかどうかを指定します。

EngStart() 関数は、以下の処理を行います。

- (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。(プロセッサ、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み出し、(1) で準備した情報領域内に登録します。
- (3) restore_bkup フラグが =0 以外の値の場合、バックアップされた装置管理情報を復元します。
先に DSHEng4 が停止した際にバックアップ保存された管理情報を DSHEng4 の管理情報内に読み込みます。
- (3) 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、ドライバーを起動し、ポート、デバイスの通信開始も行われます。

7.1.2 予約変数の登録

EngStart() 関数による装置の起動処理が正常に終わった後、ユーザは、DSHEng4 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。

予約する必要がある情報として、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

EngSetReservedCeid() 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ベント ID
1	CEX_RSV_SPOOL_END	スプールデータ送信終了時に通知するイベント ID
2	CEX_RSV_LIMIT	変数リミット監視におけるホスト通知用イベント ID

(2) 装置定数 (EC)

EngSetReservedEcid() 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	SIF14 で使用する装置モデル名
1	ECX_RSV_SOFTREV	SIF14 で使用するソフトウェアバージョンコード
2	ECX_RSV_SPOOL_MAX	最大スプール数
3	ECX_RSV_SPOOL_OVERWRITE	スプールのプール数
4	ECX_RSV_INIT_COMMSTATE_	エンジン起動時の自動通信 Enable の指定用変数

(3) 装置状態変数 (SV)

EngSetReservedSvid() 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 (DSHEng4 が値を更新する)
1	SVX_RSV_COMMUNICATING	通信状態
2	SVX_RSV_SPOOL_STATE	スプール状態
3	SVX_RSV_SPOOL_TOTAL	スプール合計
4	SVX_RSV_SPOOL_ACTUAL	実スプール数 (貯えられた)
5	SVX_RSV_SPOOL_STIME	スプール開始時刻
6	SVX_RSV_SPOOL_FTME	スプール満杯時刻
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

7.1.3 ユーザプログラム処理対象受信1次メッセージの登録

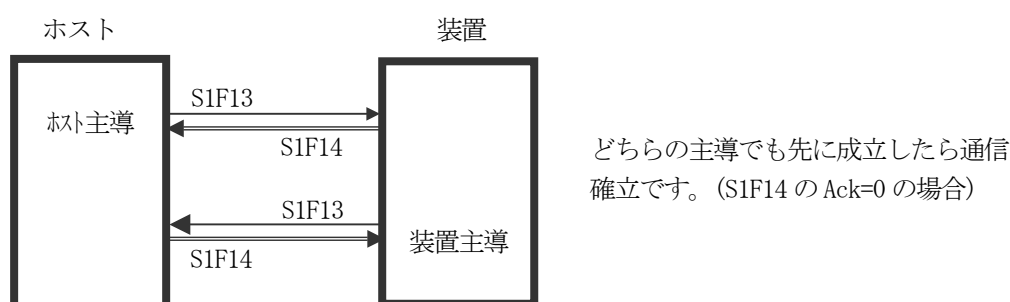
6. で説明したユーザ処理対象1次メッセージの登録を EngRegisterDefaultSxFy () 関数を使って行います。

7.1.4 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、**EngEnable()** 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。



通信停止は、**EngDisable()** 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、DSHEng4 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。

EngSetReservedSvid() 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、DSHEng4 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(装置データ変数)があります。これら変数は装置の起動処理時にDSHEng4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	EngSetEcVal ()	EngGetEcVal ()
2	SV	EngSetSvVal ()	EngGetSvVal ()
3	DVVAL	EngSetDvVal ()	EngGetDvVal ()
4	V (EC, SV, V)	EngSetVVal ()	EngGetVVal ()

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という条件で、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、DSHEng4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、EngSet??Min(), EngSet??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、しかも、収集イベント ID(CEID)が指定されていた場合、DSHEng4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。

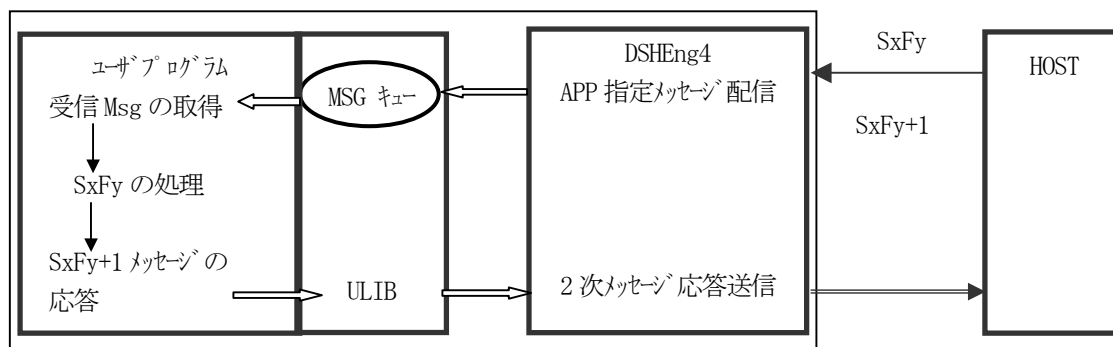
(3) 変数値のバックアップ

DSHEng4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。DSHEng4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、ホストからのレシピ生成、S15F13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 S15F13 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。

DSHEng4 は、S15F13 のデコード関数 `DshDecodeS15F13()` 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 S15F13 に含まれる情報を展開することができます。

`DshDecodeS15F13` 関数のプロトタイプと使用する構造体は次の通りです。

(1) S15F13 のデコード関数

```
API int APIX DshDecodeS15F13(
    DSHMSG *msg,           // SECSメッセージ情報構造体のポインタ
    TRCP_INFO *pinfo      // デコードした情報を格納する構造体のポインタ
);
```

(2) レシピ情報格納構造体

```
typedef struct {           // TRCP_INFO - Recipe Information
    int    index;
    int    state;         // レシピ状態
    char   *name;        // name
    char   *rcpspec;     // rcpid
    int    para_count;   // # of pparameter
    TRCP_PARA **para_list; // parameter list
    char   *rcpbody;
} TRCP_INFO;             // Recipe Information
```

(注) `ppara_count = 0` の場合、パラメータがないことを意味します。

```

typedef struct{                                     // TRCP_PARA - Recipe Parameter Information
    char      *rcpparm;                             // para name
    int       par_fmt;                               // format
    int       par_size;                             // array size
    void      *rcpparval;                           // para value;
}TRCP_PARA;                                       // Recipe Parameter

```

(3) レシピ情報の DSHEng4 への登録と設定

装置は、受信したレシピ情報を DSHEng4 内に登録、設定します。

具体的には、EngAllocRcpInfo() 関数を使って、まず、レシピ ID を登録します。

その後、EngSetRcpInf() 関数を使って、TRCP_INFO の内容を DSHEng4 のレシピ情報として設定します。

そして、必要な処理を行った後、DshFreeTRCP_INFO() 関数を使って、TRCP_INFO 内に使用したメモリを開放します。

(4) S15F13 に対する S15F14 の応答

応答情報は、次の構造体内に応答情報を生成し、それを引数として、ULIB 内の DshResponseS15F14() 関数を使用して S15F14 メッセージを送信します。

```

typedef struct{                                     // TRCP_ERR_INFO - Response Information - S15F14
    int       rmack;                                 // U1
    int       err_count;
    TERR_INFO **err_list;
} TRCP_ERR_INFO;

typedef struct{                                     // TERR_INFO - Response Error Information
    int       errcode;
    char      *errtext;
} TERR_INFO;

```

この構造体に情報を設定するための次の関数が準備されています。

DshInitTRCP_ERR_INFO()、DshPutTRCP_ERR_INFO()

応答が終了した後、DshFreeTRCP_ERR_INFO() 関数で、TRCP_ERR_INFO 内で使用したメモリを開放します。

DSHEng4 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や応答メッセージに設定する情報のための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. callback - コールバック関数 (=0 ではコールバックなし)
2. 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)
3. 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、DSHEng4 にそのコールバック関数を呼び出してもらおう方法 (プログラムはブロックされません)

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、DSHEng4 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
S5F1	H←E	アラーム報告	EngNotifyAlarm()
S6F11	H←E	イベントレポート送信	EngNotifyEvent()
S6F13	H←E	注釈付きイベントレポート送信	EngAnNotifyEvent()
S7F1	H→E	プロセスプログラムポート問合せ	EngSendS7F1()
S7F3	H→E	プロセスプログラム送信	EngSendS7F3()
S7F5	H→E	プロセスプログラム要求	EngSendS7F5()
S7F23	H→E	フォーマット付プロセスプログラム送信	EngSendS7F23()
S7F25	H→E	フォーマット付プロセスプログラム要求	EngSendS7F25()
S7F27	H←E	プロセスプログラム妥当性送信	EngSendS7F27()
S7F29	H←E	プロセスプログラム妥当性問合せ	EngSendS7F29()
S10F1	H←E	端末要求	EngSendS10F1()
S14F21	H←E	Generic Service End	EngSendS14F21()
S15F1	H→E	Recipe management m-blk inq	EngSendS15F1()
S15F3	H→E	Recipe Namespace action Req	EngSendS15F3()
S15F5	H→E	Recipe Namespace rename Req	EngSendS15F5()
S15F7	H→E	Recipe Space Request	EngSendS15F7()
S15F9	H→E	Recipe Status Request	EngSendS15F9()
S15F13	H→E	Recipe Create Request	EngSendS15F13()
S15F17	H→E	Recipe Retrieve Req	EngSendS15F17()

7.2.4 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。

- DSHEng4 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 EngNotifyEvent () 関数を使ってイベントを通知する。

(1) DSHEng4 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、DSHEng4 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。

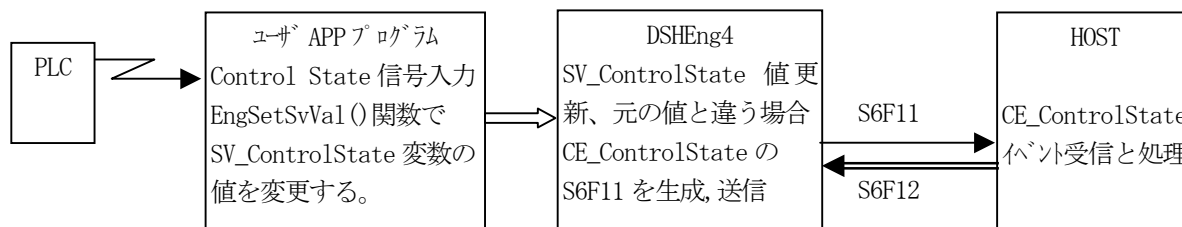
① 装置管理情報定義ファイル上に予め、状態変数の変化に対する収集イベント ID を指定しておきます。

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。

```
EngSetSvVal (SV_ControlState, &new_control_state); // new_control_state= 新変数値
```

この SV 値設定関数によって DSHEng4 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、DSHEng4 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 EngNotifyEvent () 関数によるイベント通知

ControlState 信号を入力した後、EngSetSvVal () 関数で値を更新し、ユーザプログラムの判断で、EngNotifyEvent () 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 EngNotifyEvent (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 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、EngNotifyAlarm()です。

```
API int APIX EngNotifyAlarm(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、DSHEng4 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。

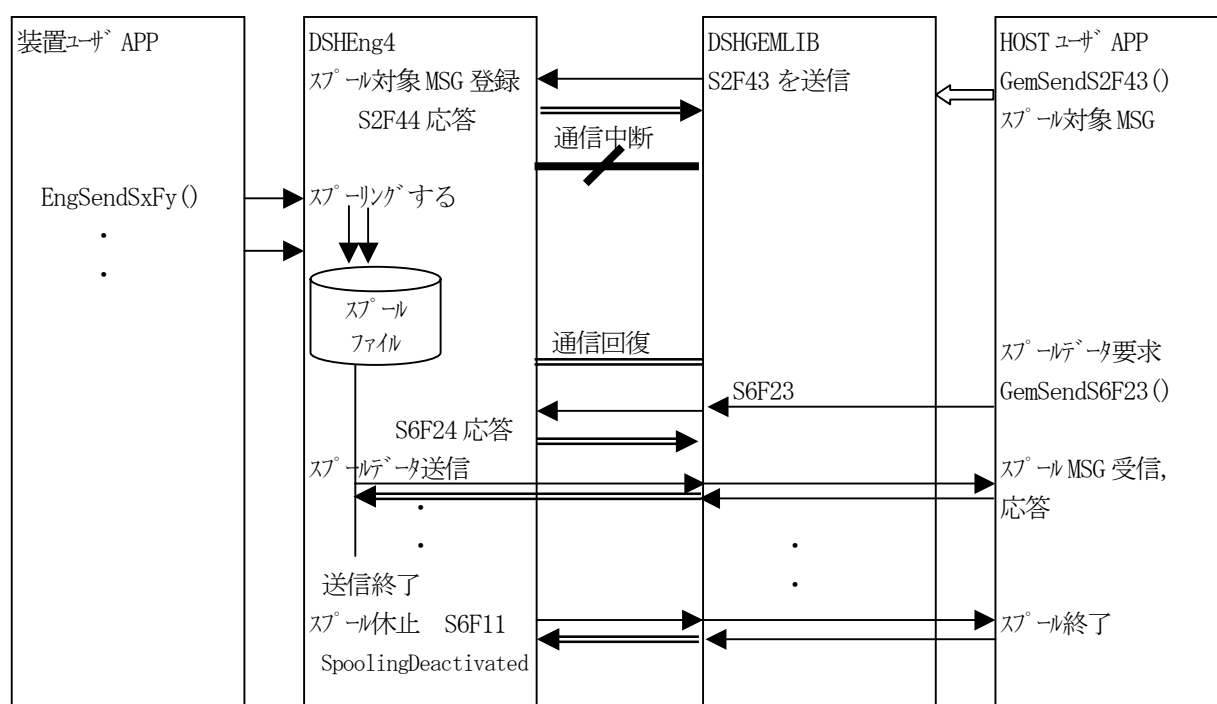
7.2.6 スプール機能処理について

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よ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。DSHEng4 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 DSHEng4 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、DSHEng4 が自動的に行ってくれます。

7.2.7 トレース機能処理について

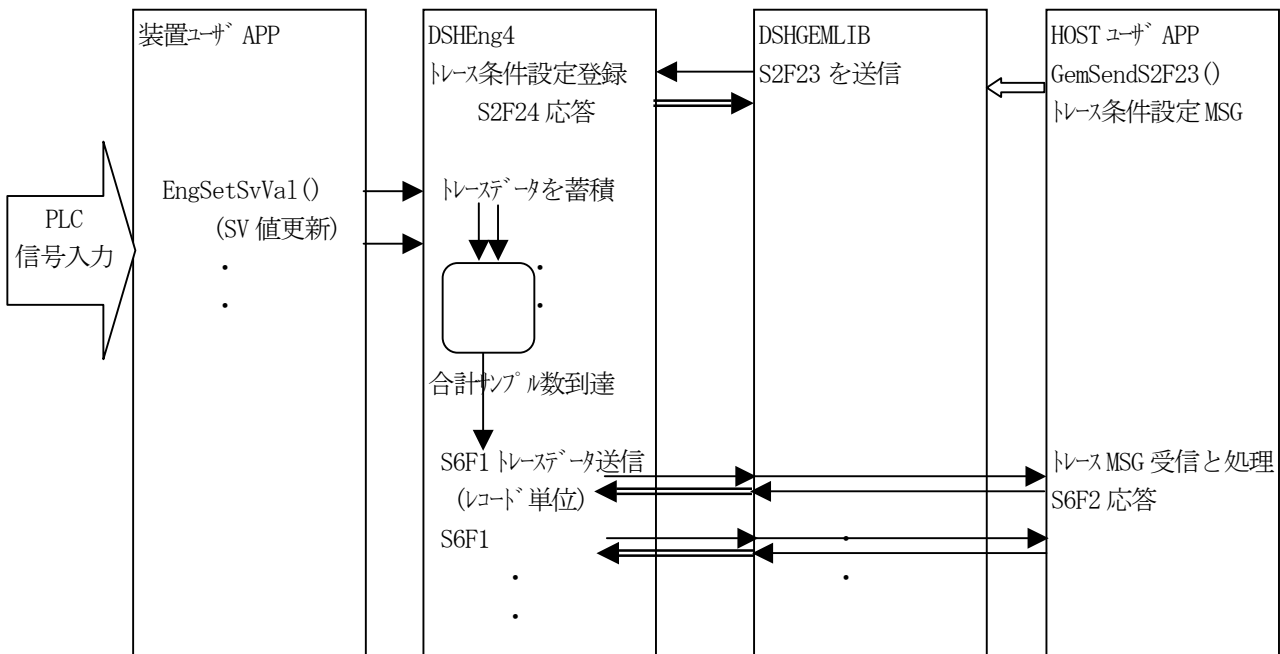
ホストは S2F23 メッセージを使って装置に対し指定した装置状態変数を周期的にトレース監視し、その結果を S6F1 メッセージで報告することができます。これは状態変数の値をサンプリングするための機能です。

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 : 合計サンプル数
 repgsz : グループレコードサイズ svid : 装置変数(1 個以上)
 dsper 時間間隔でサンプルリングし、repgsz 回毎に S6F1 で結果をホストに報告する。
 サンプルング回数が 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 EngSetSvVal() で設定します。

DSHEng4 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 DSHEng4 が自動的に行います。

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 DSHEng4 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。

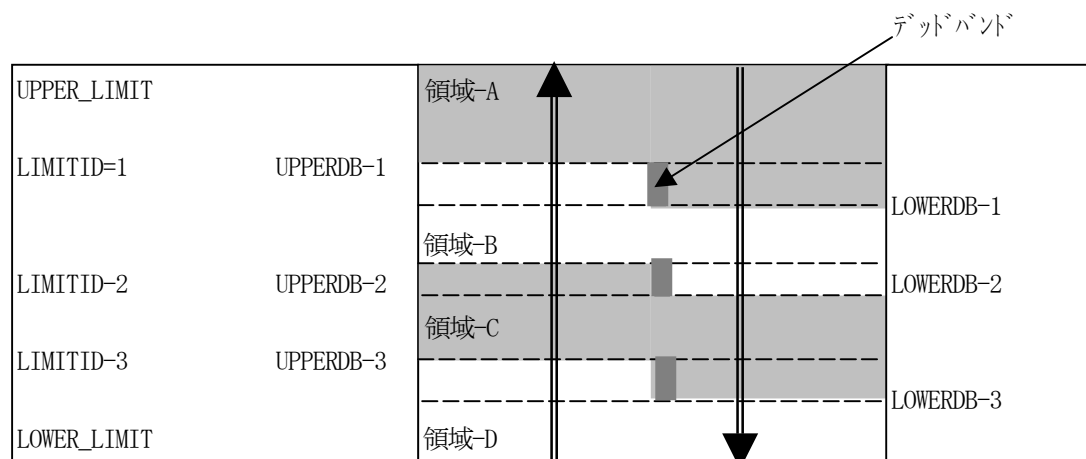
この場合、S2F23 のトレース条件を DSHEng4 に登録した後、EngEnableTrace() 関数の実行でトレース処理の開始を DSHEng4 に指示します。そして、DSHEng4 がトレースデータの収集と S6F1 の送信を行うこととなります。

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。

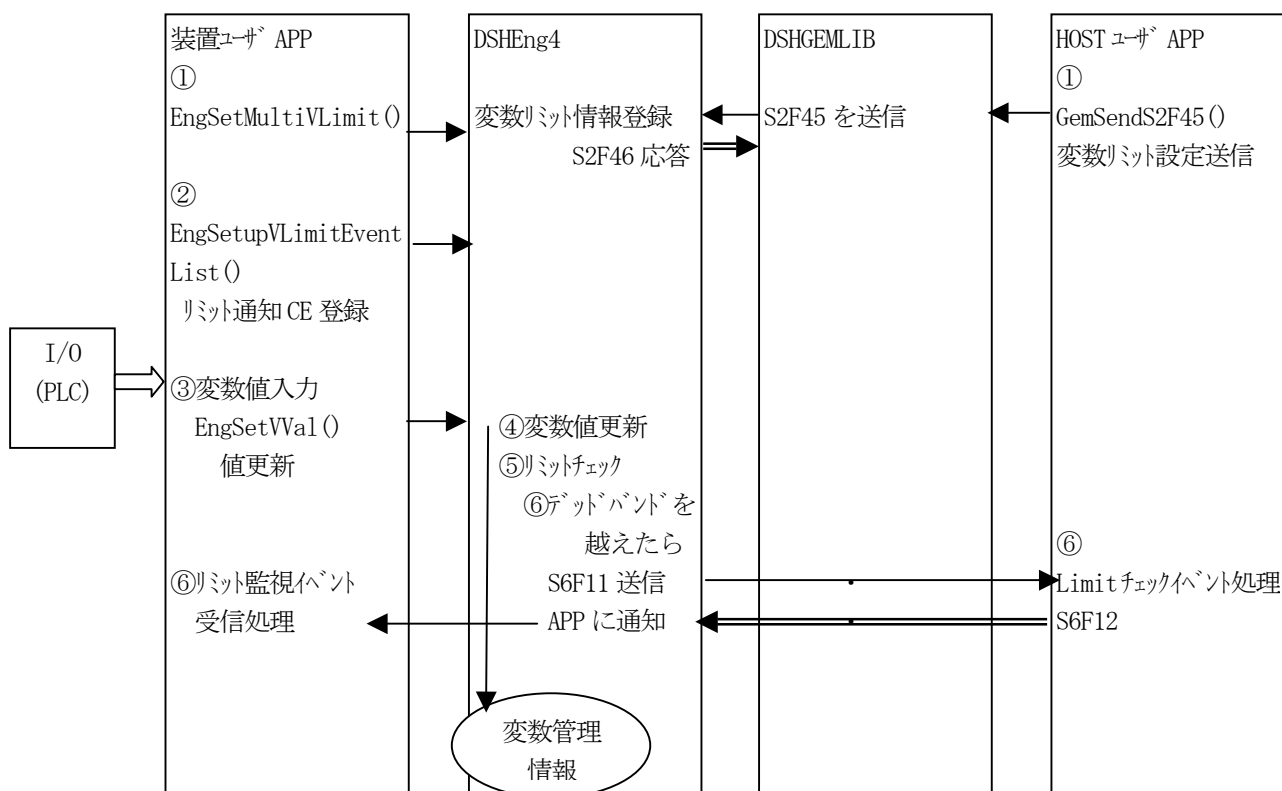
- ①変数値が 領域-A 以外の領域から UPPERDB-1 の値以上になり、領域-A に遷移したときにイベントを発生させます。
- ②変数値が 領域-A から LOWERDB-1 の値以下になり、他の領域に遷移したときにイベントを発生させます。

UPPER_LIMIT、LOWER_LIMIT 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、DSHEng4 では、リミット監視イベントをホストに通知するだけでなく、DSHEng4 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、DSHEng4 の管理情報を更新します。
 EngSetVVal() 関数を使用して行ないます。
- ④ DSHEng4 は EngSetVVal() 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 DSHEng4 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 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

7.1.1においてEngStart()関数で起動した装置の処理とDSHEng4とを終了させるための処理です。

次のEngStop()関数を使って行います。

```
API void APIX EngStop();
```

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。

DSHEng4 エンジン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。

<付録>

付録 - A DSHEng4 通信制御エンジンライブラリ関連文書一覧表

付録 - A 1 DSHEng4 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	DSHENG4-09-30300-00	DSHENG4 通信制御エンジンライブラリ (SECS/HSMS) ユーザーズ・ガイド	DSHENG4 の全般的な機能の説明書です。
2	DSHENG4-09-30301-00	DSHENG4 起動ファイル定義仕様書	装置別の起動情報の定義方法の説明書です。
3	DSHENG4-09-30302-00	DSHENG4 装置管理情報定義仕様書 (変数、収集イベント、アラームその他)	DSHENG3 と同じ内容です。定義ファイルはテキストファイルです。
4	DSHENG4-09-30303-00	装置管理情報定義ファイルコンパイル説明書	DSHENG3 と共通です。
5	DSHENG4-09-30304-00	DSHENG4 への手引き	DSHENG4 導入時に参考にする作業手順書です。
6	DSHENG4-09-30305-00	インストールと保存ファイル	製品インストール手順です。
7	DSHENG4-09-30308-00	DSHENG4, 起動ファイル、装置管理情報ファイル設定・編集プログラム説明書	DSHGEM-LIB, DSHEng4 共通
8	DSHENG4-09-30310-00	変数リミット監視機能 説明書	リミット監視の考え方、処理方法の説明書です。
9	DSHENG4-09-30340-00	ユーザ作成ライブラリ関数 2次メッセージ応答関数一覧表	C, C++言語によるプログラミング .Net 用クラスライブラリを使用しない
10	DSHENG4-09-30351-00	バックアップ ファイル参照プログラム説明書	DOS コマンドで List 構造表示します・
11	DSHENG4-09-30340-00	ユーザ作成ライブラリ関数 2次メッセージ応答関数一覧表	C, C++言語によるプログラミング .Net 用クラスライブラリを使用しない
12	DSHENG4-09-30351-00	バックアップ ファイル参照プログラム説明書	DOS コマンドで List 構造で表示します・

付録 - A 2 DSHEng4-CLASS 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	エンジン、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 メッセージ送信
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プ 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・デバッグ用

付録 - A 3 DSHeng4 エンジン・API 関数ライブラリ関連文書一覧表

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

付録 - A 4 DSHeng4 エンジン・HSMS 通信ドライバー関連文書一覧表

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

付録 - A 5 DSHENG4 エンジン・デモ・プログラム関連文書一覧表

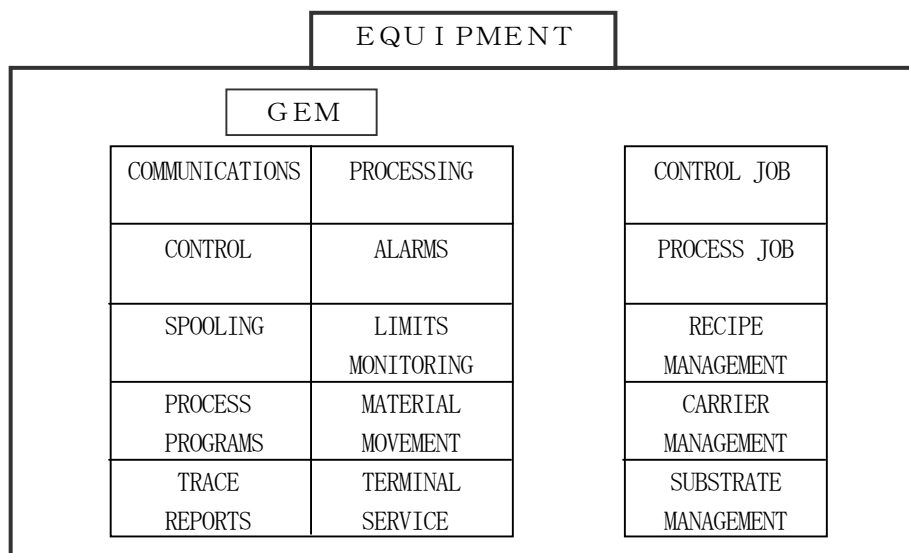
#	文書番号	文書名	注釈
1	DSHENG4-09-30501-00	クラス・ライブラリ・デモプログラム説明書	(本ドキュメントです。)
2	DSHENG4-09-30502-00	DSHGemClass クラス・ライブラリ版 デモプログラム インストールと保存ファイル	C#, .Net VB デモプログラムです。

付録 - B SEMI スタンドへの対応

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



GEM スタン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 DSHEng4 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。

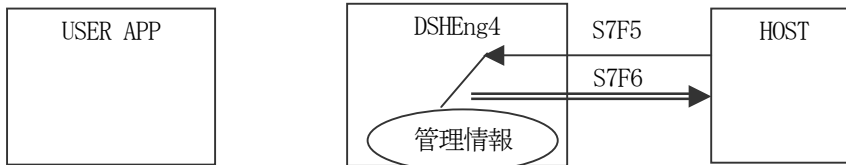
付録 - C 装置管理情報とバックアップファイル

- (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 DSHEng4 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、DSHEng4 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、DSHEng4 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 - D SECS-II の通信制御処理

DSHEng4 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に仕組みと手段を提供します。

- ① DSHEng4 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。

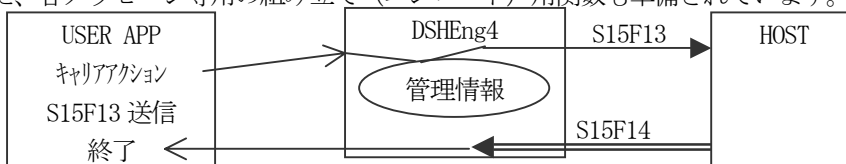


- ② APP が処理したい受信メッセージの場合、DSHEng4 が受信キューを通して APP に渡します。

DSHEng4 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。

