

DSHEng4 装置通信エンジン (GEM+GEM300)

ソフトウェア・パッケージ

# APP インタフェース ライブラリ関数説明書

( C, C++, .Net-Vb,C# )

VOL- 1 / 1 5

- 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
  - 2 . 1 1次メッセージ送信と1次メッセージ受信処理サービス機能について
- 3 . DSHEng4 API 関数
  - 3 . 1 DSHEng4 初期設定関連関数
  - 3 . 2 通信制御関連関数

2010年5月

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.	2010.5	3.1 関数追加	DshCkBkupRpInfo(), DshCkBkupRpInfo()関数の説明を追加した。

## 目 次

1 . 概要.....	1
1 . 1 DSHEngエンジン関数説明書一覧表.....	2
1 . 2 DSHEng4 について.....	3
1 . 3 DSHEngエンジン使用のためのヘダーファイルなど.....	4
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.....	5
2 . 1 1次メッセージ送信と1次メッセージ受信処理サービス機能について.....	6
2 . 1 . 1 1次メッセージ送信の一般的な処理方法.....	6
2 . 1 . 2 受信1次メッセージの一般的な処理方法.....	8
3 . DSHEng4 API関数.....	10
3 . 1 DSHEng4 初期設定関連関数.....	10
3 . 1 . 1 EngStart() - DSHEng4 の初期化処理と起動.....	12
3 . 1 . 2 EngStop() - DSHEng4 の終了.....	14
3 . 1 . 3 EngSetReservedCeid() - 予約CEID (収集イベントID) の登録.....	15
3 . 1 . 4 EngSetReservedEcid() - 予約ECID (装置定数ID) の登録.....	17
3 . 1 . 5 EngSetReservedSvid() - 予約SVID (装置状態変数ID) の登録.....	19
3 . 1 . 6 EngRegisterDefaultSxFy() - ユーザ処理SxFyの登録.....	21
3 . 1 . 7 EngInitSecsMsgReq() - APPが処理する1次メッセージ登録キューの初期化.....	23
3 . 1 . 8 EngRegSecsMsg() - APPが処理する1次メッセージの登録.....	24
3 . 1 . 9 DshCkBkupEcInfo() - バックアップEC情報状況確認.....	26
. DshCkBkupSvInfo() - SV.....	26
. DshCkBkupDvInfo() - DVVAL.....	26
. DshCkBkupPpInfo() - PP(プ 吨ｽﾌﾟ ﾀｸﾞﾗﾑ).....	26
. DshCkBkupFppInfo() - FPP(書式付ﾌﾟ 吨ｽﾌﾟ ﾀｸﾞﾗﾑ).....	26
. DshCkBkupRcpInfo() - RCP(ﾚｼﾌﾟ).....	26
. DshCkBkupCarInfo() - CAR(ｷﾞｱ).....	26
. DshCkBkupSubstInfo() - SUBST(基板).....	26
. DshCkBkupCjInfo() - CJ(ｺﾝﾄﾛｰﾙｼﾞ ﾕﾌﾞ).....	26
. DshCkBkupPrjInfo() - PRJ(ﾌﾟ 吨ｼﾞ ﾕﾌﾞ).....	26
3 . 1 . 13 EngSetDataFormat() - メッセージデータアイテムのフォーマットの設定.....	28
3 . 1 . 14 EngGetDataFormat() - メッセージデータアイテムのフォーマットの取得.....	30
3 . 2 通信制御関連関数.....	31
3 . 2 . 1 EngEnable() - 通信可能状態の確立要求.....	32
3 . 2 . 2 EngCancelEnable() - 通信可能状態確立要求の取消し.....	35
3 . 2 . 3 EngDisable() - 通信中断要求.....	36
3 . 2 . 4 EngSendPrimary() - 1次メッセージの送信要求.....	38
3 . 2 . 5 DshResponseSxFy() - 2次メッセージの送信.....	41
3 . 2 . 6 EngGetSecsMsgReq() - APPが処理する1次メッセージを取出す.....	43
3 . 2 . 7 EngGetEqCommState() - 装置通信状態の取得.....	45
3 . 2 . 8 EngGetCommPortStatus() - 通信ポートプロトコル状態の取得.....	46

(VOL - 2 に続く)



## 1. 概要

本説明書は、DSHEng4 ユーザーズガイドの内容が理解されていることを前提にして説明します。

.NET VB2008, C#2008 言語のサポートを行うための説明が追加され、改訂された版です。各関数について VB, C# の関数呼出書式が記述されています。

各関数について呼出書式が書かれていますが、関数に含まれる引数のデータタイプの指定として、以下のように記述されていますので注意してください。

### (1) .NET VB2008 の関数の記述について

関数 ( eng\_api.vb, eng\_lib.vb ファイル上にでてくる関数) の override についてですが、関数が格納先用の引数を持つものについて、その引数を取り得るデータタイプが、整数(Int32)、文字列(string)その他の型である可能性のあるものがありますが、説明の中では、IntPtr のタイプのデータを引数の変数として呼び出すように説明されています。

しかし、実際には、eng\_api.vb, eng\_lib.vb の関数インタフェース定義ファイル上に、Int32, String 型の引数のための関数も override されて定義されているものもあります。

Int32, String あるいは、それ以外の型のものについては、ユーザによって override の関数を追加する必要がある場合、eng\_api.vb または eng\_lib.vb に override 関数を含めることができます。

### (2) .NET C#2007 の関数

引数について、以下のように記述されています。

#### [ 読出し元引数 ]

関数が、整数(Int32)以外の、string 型などの格納先用の引数を持つものについては、一部を除いて byte[] の変数を指定して関数を呼び出すようにしています。

#### [ 書込み先引数 ]

関数が、整数(Int32)以外の、string 型などの格納先用の引数をもつものについては、byte[] の変数を指定して関数を呼び出すようにしています。

参考用に、DSHEng4 のデモプログラムのソースファイルを参照ください。

#### [ 注釈 ]

本説明書で取り上げるライブラリ関数は、DDSHEng4 通信エンジンライブラリに対し、直接 API 関数を使用してプログラミングする際に必要になるものです。

その後、Eng4Class クラス・ライブラリが開発され、ユーザに提供されるようになりましたので、ユーザは C#, .Net VB 言語を使用される場合には、クラス・ライブラリに登録されているクラスに対するプログラミングによってユーザプログラムを開発されることを推奨します。

なお、VC6, VB6 での開発の場合は、本ライブラリ関数を使用してのプログラミングになります。

## 1.1 DSHEng エンジン関数説明書一覧表

ライブラリ関数説明書は VOL-1 から VOL-15 に分かれており、それぞれの VOL の内容は次表のとおりです。

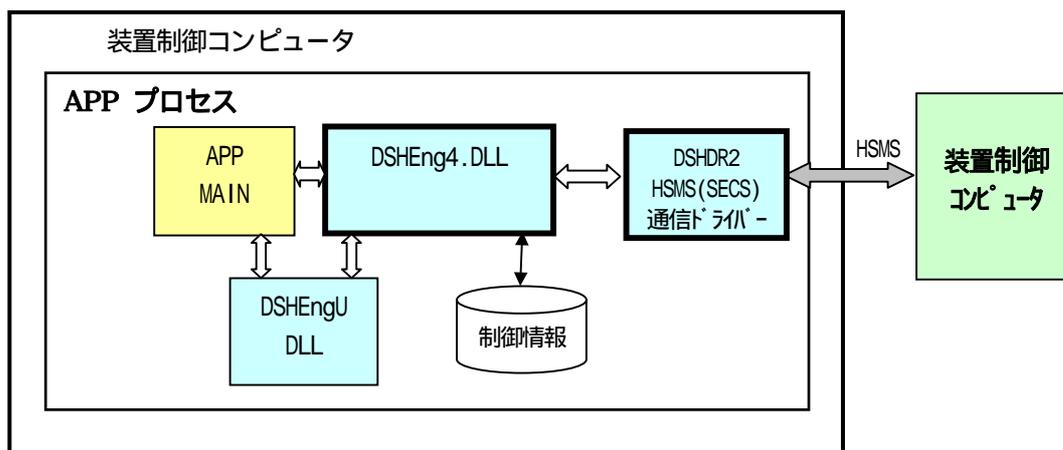
VOL 番号	文書番号	タイトル名と内容
1	DSHEng4-09-30321-00	1 . 概要 2 . DSHEng4 が提供するサービスと1次メッセージの送受信処理 3 . 1 DSHEng4 初期設定関連関数 3 . 2 通信制御関連関数
2	DSHEng4-09-30322-00	3 . 3 変数 ( EC, SV, DVVAL ) 情報アクセスと通信サービス
3	DSHEng4-09-30323-00	3 . 4 Limit 変数リミット情報関連関数 3 . 5 TR トレース情報アクセスサービス関数
4	DSHEng4-09-30324-00	3 . 6 CE 収集イベント情報アクセスと通知関数 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 . 21 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

## 1.2 DSEng4 について

DSSEng4 通信エンジンライブラリは半導体工場に置ける装置側の SEMI-GEM + GEM300 仕様の通信制御サービスをサポートするパッケージプログラムです。

装置を制御するコンピュータにおける APP (装置管理アプリケーションプログラム) は、装置管理情報のアクセスおよび SECS/HSMS 通信メッセージの送受信制御などの機能実行を要求しますが、その際、本説明書で述べる DSEng エンジンが提供する API 関数群を使用できます。

DSSEng4 を使用したソフトウェア構成の中の簡単な制御の関連を下図に示します。



APP プロセスの枠内のプログラム、APP MAIN(.EXE)、DSSEngU.DLL そして DSSEng4.DLL 自身がライブラリ関数を使用します。

各々が有する機能、そのために必要なライブラリプログラムファイルは下表のとおりです。

表 1 システム構成ソフト一覧表

プロセス構成要素	役割	備考
EngApp.EXE(仮称)	対装置通信関連処理プログラム	ユーザが作成します。
DSSEng4.dll	GEM 通信制御エンジンライブラリ DLL プログラム	パッケージに含まれます。
DSSEng4U.dll	ユーザ作成 DLL プログラム	ユーザが作成します。 *1
dshdr2.dll	装置との SECS/HSMS プロトコル通信制御ドライバー機能	パッケージに含まれます。

\*1 提供されるデフォルトプログラムを基にユーザが完成させます。

本パッケージには、ホスト側、装置側双方のデモプログラムが提供されます。デモプログラムには、本説明書に含まれる沢山の典型的な API 関数の使用例が含まれています。是非参考にされることをお勧めします。

### 1.3 DSEng エンジン使用のためのヘッダファイルなど

DSEng4 がユーザに提供するライブラリ関数を使用するためには、言語別に以下のファイルをヘッダファイルとして、あるいはプロジェクトの中を含める必要があります。

c, C++, VB2008, C#2008 別に基本的なファイルを次表に示します。  
(これらファイルはデモプログラムの中に含まれています。)

	c, C++	VB2008	C#2008	注釈
1	dshdr2.h	dshdr2.vb	dshdr2.cs	DSHDR2 HSMS 通信ドライバの定数、構造体、API 関数が定義されています。
2	dsh_info.h	dsh_info.vb	dsh_info.cs	DSEng4 が使用する定数、情報格納用構造体が定義されています。
3	eng_api.h	eng_api.vb	eng_api.cs	DSEng4 の開始、終了、装置開始、終了から装置(HOST)に対する GEM 関連情報アクセスと通信関数が定義されています。
4	eng_lib.h	eng_lib.vb	eng_lib.cs	装置(HOST)を問わない共通のライブラリ関数が定義されています。 SECS-II メッセージのデコード、エンコード、構造体の定義、開放の関数の定義がされています。
5	eng_ulib.h	eng_ulib.h	eng_ulibcs	ユーザ作成 DLL 関数が定義されています。 APP が受信処理する 1 次メッセージの登録、2 次メッセージの応答関数の定義です。
6	user_def.h	eng_user.vb	eng_user.cs	ユーザが定義が再定義できる定数が定義されています。 SEMI で規定される SECS-II メッセージのデータ型の定義などです。 予約 CEID, ECID, SVID の定数

なお、本説明書のなかでは、情報を格納する構造体について使用する構造体の項で説明していますが、構造体の表現は C 言語によるものだけです。したがって VB, C# 言語については、特に説明していません。

VB, C# 言語表現によるものは、それぞれ dsh\_info.vb、dsh\_info.cs ファイルを参照ください。

## 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

サービスの種類は以下のように分類できます。これらのサービスはユーザ APP によって DSHeng4.DLL が提供する API 関数を呼び出すことによって行うことができます。

### ( 1 ) SECS-II 通信関連サービス

通信確立、イベント、アラームなどの装置主導 SECS-II メッセージの送信サービス  
(送信するための SECS-II 通信メッセージの作成は DSHeng4 が行います。)  
装置からの通信メッセージ配信サービス (ユーザ APP が処理する SECS-II メッセージ)

### ( 2 ) 情報アクセスと装置通信関連サービス

以下の管理情報について情報アクセスと装置への各情報の通知、指令、問合せなどを行います。

変数、収集イベント、アラーム関連情報の管理とアクセスサービス

EC 装置定数

SV 装置状態変数

DWAL 装置データ変数

変数リミットチェック

CE 収集イベント

RP(レポート情報

AL(アラーム情報)

PP プロセスプログラム情報

FPP 書式付プロセスプログラム情報

スプール情報

トレース情報

PORT ポート情報 (変数)

Carrier キャリア情報

Substrate 基板情報

Recipe レシピ情報

CJ コントロールジョブ情報

PRJ プロセスジョブ情報

ホスト指令、キャリアアクション指令情報の送信

レチクル搬送ジョブとサービス制御

## 2.1 1次メッセージ送信と1次メッセージ受信処理サービス機能について

ユーザプログラムが行う SECS-II メッセージの通信処理が最も重要な処理の1つです。DSHEng4 はメッセージの送受信のための標準的な処理のための様々な構造体、API 関数のサービスを提供します。

### 2.1.1 1次メッセージ送信の一般的な処理方法

1次メッセージの送信処理を、DSHEng4 が提供する手段を使って行うことを前提にしています。そしてそのために準備されている情報格納構造体と関数を使って一般的にどのように処理するかについて説明します。

送信1次メッセージ SxFy に含まれる情報をエンコード結果格納するための構造体を使用します。  
説明のため、これを仮に TXXYY\_INFO とします。

SxFy に含める情報を TXXYY\_INFO 構造体に次の関数を使って設定します。エンコードするための構造体です。

DshInitTXXYY\_INFO() - 構造体の初期設定関数

DshPutTXXYY\_INFO() - 構造体内への情報(パラメータなど)の設定関数

TXXYY\_INFO 構造体内に格納された情報から SxFy メッセージを生成するために次の関数を使用します。  
DshEncodeSxFy() 関数が用意されています。

SxFy メッセージを次の関数を使って送信します。

EngSendSxFy()

送信関数の戻り方の指定には、2つのモードがあります。

- a. 送信し、送信終了(応答メッセージを受信)するまで待機するブロックモード
- b. 送信する前に送信依頼だけを行ってすぐ制御を戻し、送信終了は、EngSendSxFy()関数で指定したコールバック関数で通知してもらうようにするアンブロックモード  
(処理しているプログラムがブロックすると他の処理ができなくなり、問題が生じる場合は、コールバック関数を指定するモードをお勧めします。)

また、応答2次メッセージを受信する方法として、生の応答メッセージではなく、その中に含まれる応答情報を格納する領域のポインタを与え、そこに情報を返してもらうこととなります。

応答情報格納用領域としては、単純に ACK だけを格納する場合、あるいは、リスト構造を持つような情報を所定の構造体内に格納する場合の大体二通りの方法がありますが、どちらが採用されるかはメッセージによって決まります。

受信した2次メッセージから得られた応答情報を処理します。

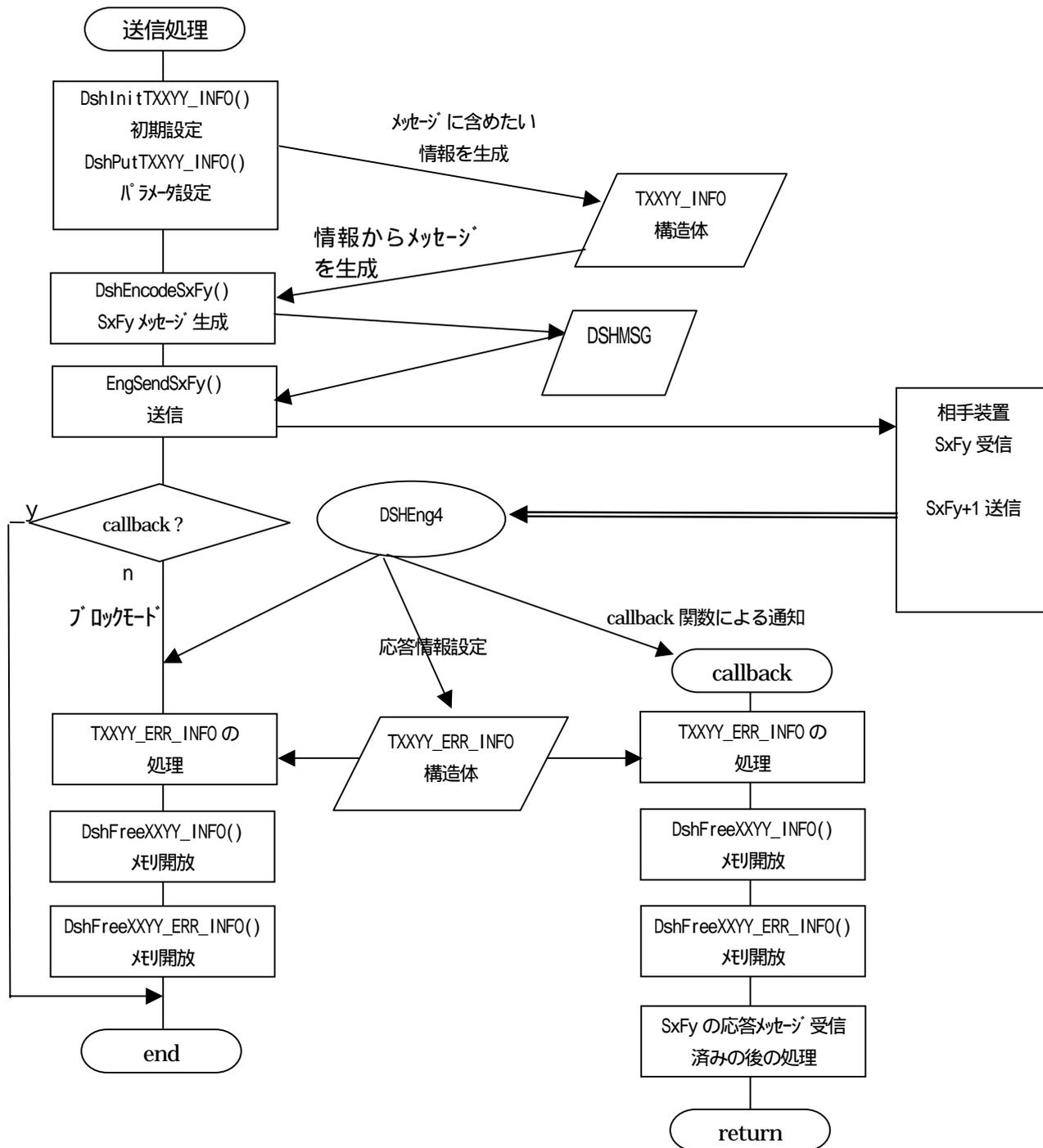
SxFy の送信が完了した後、 の SxFy の1次メッセージを生成した時、DshInitTXXYY(), DshPutTXXYY() 関数で構造体内に割付使用したメモリを開放します。

DshFreeTXXYY\_INFO() 関数が用意されています。

また、受信した2次メッセージの応答情報を構造体に返却してもらった場合には、応答情報の処理後、同様に構造体内部に使用されたメモリを開放する必要があります。

DshFreeTXXYY\_ERR\_INFO() 関数が用意されています。

次ページに処理の流れ図を示します。



## 2.1.2 受信1次メッセージの一般的処理方法

相手装置から送信されてくる1次メッセージで、アプリケーションプログラムが処理すべきものを、EngGetSecsMsgReq()関数で取得します。(DSHEng4 が受信メッセージキューに入れてくれたもの) 取得したメッセージの処理は一般的に次のように行うことができます。

説明する上で、受信したSxFyを処理するケースの場合、受信1次メッセージをデコードした情報をTXXYY\_INFO構造体に格納し、応答する2次メッセージ用の情報をTXXYY\_ERR\_INFO構造体に格納して処理することを前提にします。

まず、EngGetSecsMsgReq()関数を使って、受信した1次メッセージがあるかどうかチェックします。もし、受信したメッセージがあれば、それをTMSG\_QUEUE\_INFO構造体に取り込みます。TMSG\_QUEUE\_INFO構造体には、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msgがあります。(この部分はサンプルプログラムのpoll.cppのソースファイルを参照してください。)

msg内にあるメッセージをdshDecodeSxFy()関数を使ってTXXYY\_INFO構造体に情報をデコードし取り込みます。

TXXYY\_INFO構造体内の情報について必要な処理をユーザプログラムが行います。

の処理の結果を、応答メッセージを生成するときに使用する応答情報構造体TXXYY\_ERR\_INFO内に設定します。使用する関数は、DshInitTXXYY\_ERR\_INFO()とDshPutTXXYY\_ERR\_INFO()関数などです。

この後、応答メッセージを生成し、送信する処理になりますが、この部分はユーザ作成DLLプログラムDSHEng4U.dll内で処理します。DSHEng4Uの参考プログラムのソースプログラムは製品に同梱されます。SxFyの応答メッセージの送信処理は、u\_sxfy.cファイルに記述されています。関数名はDshResponseSxFz()です。SxFzのzは(y+1)の値です。

- a. ユーザプログラムは、DshResponseSxFz()関数を呼出します。  
このとき、 で得られたTMSG\_QUEUE\_INFO構造体に格納されているtrid(トランザクションID)を  
そして、 で生成されたTXXYY\*\_INFO、TXXYY\_ERR\_INFOのポインタを引数で与えます。
- b. 応答メッセージを生成するためにDSHDR2通信ドライバーに応答送信要求するためのDSHMSG構造体の領域とメッセージテキスト格納に必要なメモリを確保します。そのあと、stream、function、テキストバッファの情報をDshSetupMsgForSend()関数を使ってDSHMSG内に設定します。
- c. 次にDSHMSG内のテキストバッファ内にメッセージを組み立てますが、このとき、 で生成したTXXYY\_ERR\_INFO構造体の情報を引数にして、DshMakeSxFyResponse()関数を呼出して組み立てます。
- d. そして、DshResponseSxFy()関数を使ってSxFyの応答メッセージを送信します。  
このとき、送信が終了したタイミングで呼び出してもらうためのコールバック関数を指定します。コールバック関数DSHEng4が準備し、使用したDSHMSG構造体のメモリをDshFreeMsg()関数を使って開放し、0を返却するようにします。ユーザはこのコールバック関数で実際に応答メッセージの送信が終了したことを確認できます。
- e. 最後にa,で確保したDSHMSGとテキストバッファメモリをU\_FreeMsg()関数を使って開放します。また、DshResponseSxFz()関数の中で、それぞれ、 で生成したTXX\_Y\_INFO、TXXYY\_ERR\_INFOに使用したメモリをDshFreeTXXYY\_INFO(), DshFreeTXXYY\_ERR\_INO()関数を使って開放します。もちろん、TXX\_Y\_INFO、TXXYY\_ERR\_INFOのメモリを開放したくない場合は、u\_sxfy.cのソースの中から、TXX\_Y\_INFO、TXXYY\_ERR\_INFOのメモリ開放部分をコメントアウトできます。この場合、ユーザAPPプログラムの中でメモリの開放を行ってください。

次ページに処理の流れ図を示します。



### 3 . DSEng4 API 関数

以下、API 関数について説明します。

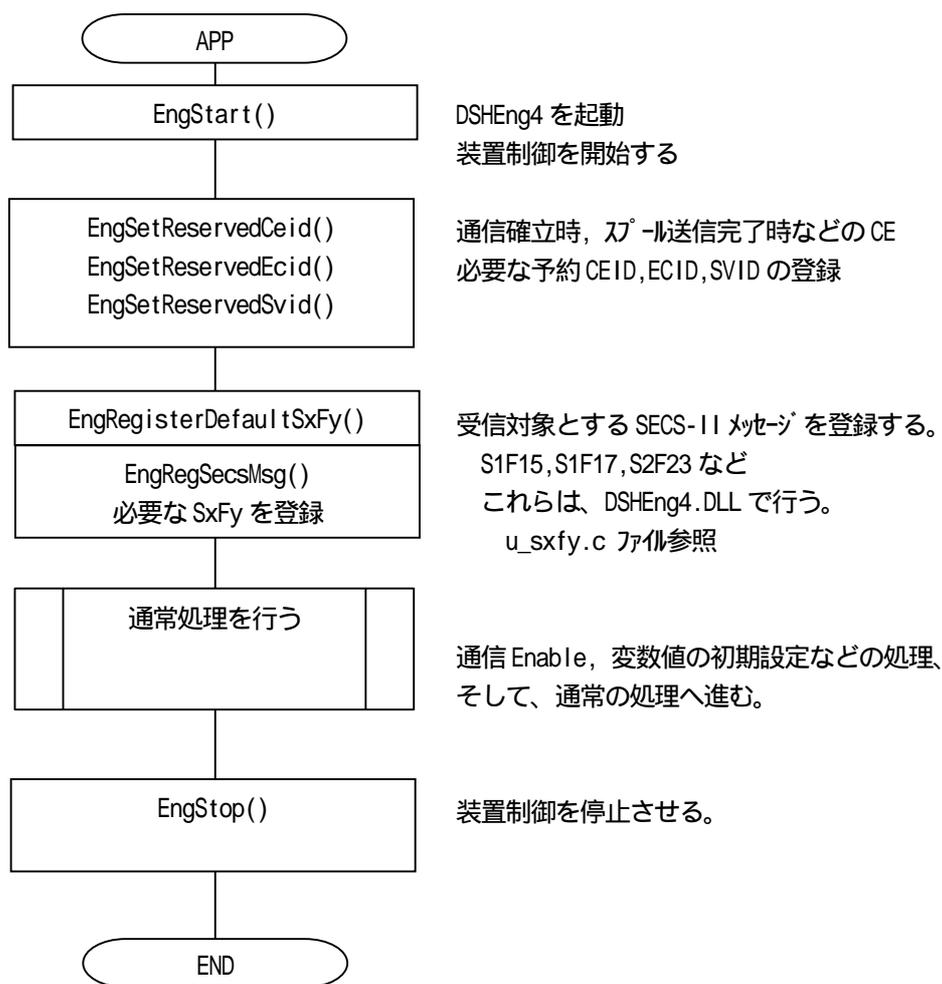
#### 3 . 1 DSEng4 初期設定関連関数

DSEng3 初期化時に使用するための API 関数は下の一覧表のとおりです。

表 3.1 DSHENG4 初期設定関連関数一覧

	関数名	機 能	備 考
1	EngStart()	DSEng4 起動と初期設定を行います。 HSMS 通信の準備もします。	引数は装置起動ファイル、装置管理情報定義 ファイル名とバックアップ 復元フラグ
2	EngStop()	装置制御を停止し、DSEng4 エンジン を停止します。	
3	EngSetReservedCeid()	予約 CEID を設定登録します。	通信確立時の収集イベントなど。 イベント関連 API 関数の節で説明します。
4	EngSetReservedEcid()	予約 ECID を設定登録します。	Model, softrev などの定数など
5	EngSetReservedSvid()	予約 SVID を設定登録します。	日付時刻, 通信状態などの状態変数
6	EngRegisterDefaultSxFy()	APP が処理します。受信 1 次メッセージの 登録を行います。	u_sxfy.c プログラム内の関数です。
7	EngInitSecsReq()	受信メッセージのキーの作成などを行いま す	EngRegisterDefaultSxFy() によって 使用される。
8	EngRegSecsMsg()	受信 1 次メッセージを処理します。処理関 数を登録します。	u_sxfy.c プログラム内で呼び出す関数*1 ユーザ 作成ライブラリ関数の節で説明します。
9	DshCkBkupEcInfo() DshCkBkupSvInfo() DshCkBkupDvInfo() DshCkBkupPpInfo() DshCkBkupFppInfo() DshCkBkupRcpInfo() DshCkBkupCarInfo() DshCkBkupSubstInfo() DshCkBkupCjInfo() DshCkBkupPrjInfo()	管理情報履歴ファイルに情報が正常に保 存します。かどうか確認します。	装置制御起動前にバックアップ 情報が正常 に残っているかどうかを確認します。 バックアップ ファイルには最大 4 世代分が保存 されています。 新しいものから順にファイルを確認します。

APP の DSHeng4 の初期設定処理は以下のように行います。



DSHeng4 を起動  
装置制御を開始する

通信確立時, スプール送信完了時などの CE  
必要な予約 CEID, ECID, SVID の登録

受信対象とする SECS-11 メッセージを登録する。  
S1F15, S1F17, S2F23 など  
これらは、DSHeng4.DLL で行う。  
u\_sxfy.c ファイル参照

通信 Enable, 変数値の初期設定などの処理、  
そして、通常の処理へ進む。

装置制御を停止させる。

### 3.1.1 EngStart() – DSHeng4 の初期化処理と起動

#### (1) 呼出書式

##### [C, C++]

```
API int APIX EngStart (
    char *comm_def,           // DSHDR2 ドライバ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            // エラ発生時のエラー情報領域ポインタ
);
```

##### [.NET VB]

```
Function EngStart (
    ByVal comm_file As String,
    ByVal sysconf As String,
    ByVal restore_bkup As Int32,
    ByVal err_str As String) As Int32
```

##### [.NET C#]

```
int EngStart(
    string comm_file,
    string sysconf,
    int restore_bkup,
    string err_str );
```

#### (2) 引数

##### comm\_def

DSHDR2 HSMS 通信ドライバーに指定する通信環境定義ファイル名を指定します。

##### conf\_file

ファイルはテキストタイプファイルであり、中に、バックアップ、スプール、ログファイルの保存ディレクトリ、ファイル名、管理情報の管理数、HSMS 通信で使用するポート、デバイス情報など装置固有の情報が定義されています。

起動ファイルの詳細については、「**起動ファイルコマンド定義仕様書**」を参照ください。

##### restore\_bkup

前回エンジンが終了した時点でバックアップされた装置管理情報の内容を復元させるかどうかを指定します。

値 = 0 は復元させない指定に、= 1 は復元させる指定になります。

##### err\_str

本関数実行中に先に進めないエラーを検出したときに、そのエラーの種類を示すメッセージを格納する char の領域を指定します。err\_str のバッファは、ユーザが準備してください。領域は 128 バイト以上のものを準備してください。

#### (3) 戻り値

戻り値	意味
0	正常に初期化とエンジン起動処理が終了した。
(-1)	エラーを検出した。

#### (4) 説明

DSHeng4 を起動し、装置との HSMS 通信の準備ならびに装置通信制御のための初期化処理も行います。

HSMS 通信の準備は、comm\_file で指定されたファイルに定義されている通信定義情報に従って dshdr2 HSMS 通信ドライバーを起動します。

conf\_file で指定された装置情報定義ファイルに従って DSHEng4 内に、変数-EC,SV,DVVAL、CE、RP、ALARM などの情報を管理するための処理も行います。(CE,RP,変数によるリンク関連情報も構築します。)

本関数が正常に終了した後、APP は指定された装置 ID の装置に対し、管理情報のアクセス、装置との通信サービスを利用することができます。

装置との通信確立は EngEnable()関数を使って行ってください。

異常終了した場合は、DSHEng4 を利用することができません。異常原因が引数 err\_str にメッセージで返されます。

restore\_bkup=1 が指定されたケースで正常だった場合、前回 DSHEng4 終了時にバックアップされた装置管理情報がエンジン内部に取り込まれます。restore\_bkup=0 の場合、管理情報は初期化された状態のままになります。

(注) エンジン起動前にバックアップファイルが正常に保存されているかどうかの判断は 3.1.12 で述べる関数を使って行うことができます。

一旦、装置が正常に開始されたら、DSHEng4 が提供する GEM 関連情報のアクセスを行うことができます。(情報取得、情報設定など)

また、HSMS 通信で実際に GEM 関連メッセージの通信を行うためには、以下の前処理を行う必要があります。

- ・ DSHEng4 によって予約されている、収集イベント(CE)、定数(EC)、状態変数(SV)の登録 EngSetReservedCeid()、EngSetReservedEcid()、EngSetReservedSvid()関数を使います。
- ・ 相手装置から受信する 1 次メッセージの DSHEng4 への登録 EngRegisterDefaultSxFy()関数を使います。ユーザ作成 DLL の中にある関数です。

その後、EngEnable()関数を使って相手装置との間で通信確立を行うことになります。( S1F13,S1F14 による通信確立です。)

EngEnable()が正常に完了した後、相手装置との GEM の SECS-II メッセージの送信と、相手装置から 1 次メッセージの受信処理を行うことができます。

送信は EngSendSxFy()、NotifyXXXX()を使って、また、受信は EngGetSecsMsgReq()関数を使って受信チェック(ポーリング)と受信を行うことができます。

### 3.1.2 EngStop() – DSHEng4 の終了

#### (1) 呼出書式

##### [C, C++]

```
API int APIX EngStop ( );
```

##### [.NET VB]

```
Function EngStop ( ) As Int32
```

##### [.NET C#]

```
int EngStop();
```

#### (2) 引数

なし

#### (3) 戻り値

戻り値	意味
0	常時0 が返却されます。

#### (4) 説明

**EngStart()**関数で起動した DSHEng4 を終了させます。

本関数は、以下の終了処理を行います。

- ・装置の処理を全て停止、終了させます。  
装置変数情報のバックアップをバックアップファイルに取ります。そして、装置関連情報に使用されていたメモリを開放します。
- ・HSMS レベルの通信ドライバーを停止させます。
- ・ログ収集を停止させ、DSHEng4 が内部で使用していたメモリを開放します。

### 3.1.3 EngSetReservedCeid() – 予約CEID (収集イベント ID) の登録

#### (1) 呼出書式

##### [C, C++]

```
API int APIX EngSetReservedCeid(
    int index,           // 登録したいCEIDのインデクス値を指定
    TCEID ceid          // インデクスに割当ててるCEID
);
```

##### [.NET VB]

```
Function EngGetReservedCeid (
    ByVal index As Int32,
    ByRef ceid As Int32) As Int32
```

##### [.NET C#]

```
int EngSetReservedCeid(
    int index,
    uint ceid );
```

#### (2) 引数

index

CEID登録のためのインデクス値です。ここで登録するCEIDはシステムによって予約されており、DSHEng4内部で使用されます。

インデクス値、そしてそのC言語のマクロとCEIDの対応表を下に示します。C言語のマクロは、user\_def.hに定義されています。

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	スプール送信の終了時に通知するイベント ID
2	CEX_RSV_LIMIT	変数リミット監視結果通知イベント ID
3		
4		

ceid

index 値で指定された収集イベントとして登録するイベント ID です。

#### (3) 戻り値

戻り値	意味
0	正常に登録できた。
(-1)	インデクス値が間違っている。

#### (4) 説明

DSHEng4 が内部で必要に応じて自動的に装置に通知するための予約収集イベントを、登録インデクス値に対応させて登録します。

例えば、装置が DSHEng4 との通信が確立したときに DSHEng4 に対し通信確立イベントを通知しますが、そのときのイベントのインデクス値は =0 に予約されています。

APP はシステム起動時に、システムで定義された通信確立イベント ID の値をインデクス-0 に登録します。

DSHEng4 は通信確立した時に、インデクス-0 に登録されているイベント ID のイベントを装置に通知することになります。もし、登録されていないければこのためのイベントは通知されないこととなります。

予約CEIDは EngGetReservedCeid()で取得できます。(3.4.2.9 参照)

### 3.1.4 EngSetReservedEcid() – 予約 ECID (装置定数 ID) の登録

#### (1) 呼出書式

##### [C, C++]

```
API int APIX EngSetReservedEcid(
    int index,           // 登録したい ECID のインデックス値を指定
    TECID ecid          // インデックスに割り当てる ECID
);
```

##### [.NET VB]

```
Function EngSetReservedEcid (
    ByVal index As Int32,
    ByVal ecid As Int32) As Int32
```

##### [.NET C#]

```
int EngSetReservedEcid(
    int index,
    uint ecid );
```

#### (2) 引数

index

ECID 登録のためのインデックス値です。ここで登録する ECID は DSHeng4 システムによって予約されており、DSHeng4 内部で使用されます。

インデックス値、そしてその C 言語のマクロと ECID の対応表を下に示します。C 言語のマクロは、user\_def.h に定義されています。

index 値	マクロ名	装置定数
0	ECX_RSV_MDLN	S1F14 で使用する装置モデル名
1	ECX_RSV_SOFTREV	S1F14 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

ecid

index 値で指定された予約装置定数として登録するイベント ID です。

#### (3) 戻り値

戻り値	意味
0	正常に登録できた。
(-1)	インデックス値が間違っている。

#### (4) 説明

装置側が必要に応じて自動的に DSHeng4 に通知される予約装置定数を登録インデックス値に対応させて登録します。

以下、ECX\_RSV\_INIT\_COMMSTATE の予約変数の設定目的、方法、作用について説明します。

ECX\_RSV\_INIT\_COMMSTATE は通信エンジン起動後、エンジン自身で自動的に通信 ENABLE にするかどうかを判断するための装置定数 ID を指定するために使います。

例えば、InitCommState マクロで定義された ECID の設定は次のように行います。

```
EngSetReservedEcid(ECX_RSV_INIT_COMMSTATE, EC_InitCommState );
```

この関数が実行された後エンジンは指定された EC\_InitCommState 変数の値を調べ、 = 1 の場合、自動的に通信 ENABLE の処理を行います。

( EngEnable ( ) API 関数に対する処理と同様の処理を自動的に行います。 )

### 3.1.5 EngSetReservedSvid() – 予約 SVID (装置状態変数 ID) の登録

#### (1) 呼出書式

##### [C, C++]

```
API int APIX EngSetReservedSvid(
    int index,           // 登録したい SVID のインデックス値を指定
    TSVID svid          // インデックスに割当ててる SVID
);
```

##### [.NET VB]

```
Function EngSetReservedSvid (
    ByVal index As Int32,
    ByVal svid As Int32) As Int32
```

##### [.NET C#]

```
int EngSetReservedSvid(
    int index,
    uint svid );
```

#### (2) 引数

index

SVID 登録のためのインデックス値です。ここで登録する SVID はシステムによって予約されており、DSHEng4 内部で使用されます。

インデックス値、そしてその C 言語のマクロと SVID の対応表を下に示します。C 言語のマクロは、user\_def.h に定義されています。

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_VID	リミット領域遷移した変数 ID
8	SVX_RSV_LIMIT_DVVAL	リミット領域遷移したときの変数値
9	SVX_RSV_LIMIT_ID	リミット領域遷移したときのリミット ID
10	SVX_RSV_LIMIT_DIR	リミット領域遷移した方向 (up, down)

svid

index 値で指定された予約装置状態として登録するイベント ID です。

#### (3) 戻り値

戻り値	意味
0	正常に登録できた。
(-1)	インデックス値が間違っている。

#### (4) 説明

装置側が必要に応じて自動的に DSHEng4 に通知される予約装置状態変数を登録インデックス値に対応させて

登録します。

### 3.1.6 EngRegisterDefaultSxFy() – ユーザ処理 SxFy の登録

#### (1) 呼出書式

##### [C, C++]

```
API int APIX EngRegisterDefaultSxFy(
    int no_of_msg,           // 登録できるメッセージ数
    HANDLE event            // メッセージ受信時に APP にポストするイベント
);
```

##### [.NET VB]

```
Function EngSetReservedSvid (
    ByVal index As Int32,
    ByVal svid As Int32) As Int32
```

##### [.NET C#]

```
int EngRegisterDefaultSxFy(
    int no_of_msg,
    uint ev );
```

#### (2) 引数

no\_of\_msg

登録する最大メッセージ数を指定します。

event

登録されたメッセージが受信された際、APP(アプリケーションプログラム)に通知するイベントです。本イベントは Win32API の CreateEvent() を使ってシステムから取得してください。値が=0 の場合は、通知イベントが無いことを意味します。

#### (3) 戻り値

戻り値	意味
0	正常に登録できた。
(-1)	DSHEng4 が受け入れてくれなかった。

#### (4) 説明

DSHEng4 が装置から受信する 1 次メッセージの中から APP 側で直接処理したいメッセージを DSHEng4 に登録します。

登録する 1 次メッセージ群は DSHEng4U.dll の u\_sxfy.c ソースファイル上に定義します。

(static TPRI\_PRO\_INFO pri\_pro\_info\_tab[] 参照)

no\_of\_msg は DSHEng4 内で登録する最大メッセージ数を指定します。

event は、登録されたメッセージを受信した際、受信したことを APP 側に知らせるためのイベントです。

event の値が=0 の場合は、イベントを通知しません。

なお、通信エンジン終了時、event が有効であれば、DSHEng4 がこの event をシステムに返却します。

APP は、受信された 1 次メッセージを受け取るために使用する関数は EngGetSecsMsgReq() です。

APP における処理方法は次のどちらかになります。

イベント通知を利用しない方法

一定時間周期で EngGetSecsMsgReq() を使ってポーリングし、戻り値が >0 の場合、受け取ったメッセージ情報に従って処理します。

## イベント通知を利用する方法

Windows API、WaitForSingleObject()関数を使ってイベントを待機し、イベント通知を受けたらEngGetSecsMsgReq()を使ってメッセージ情報を取得し、処理します。

### 3.1.7 EngInitSecsMsgReq() – APP が処理する 1 次メッセージ登録キューの初期化

#### (1) 呼出書式

##### [C, C++]

```
API void APIX EngInitSecsMsgReq(
    int    no_of_msg,           // 登録できるメッセージ数
    HANDLE event               // メッセージ受信時に APP にポストするイベント
);
```

##### [.NET VB]

```
Sub EngInitSecsMsgReq (
    ByVal count As Int32,
    ByVal evt As Int32)
```

##### [.NET C#]

```
void EngInitSecsMsgReq(
    int count,
    uint evt );
```

#### (2) 引数

no\_of\_msg

登録する最大メッセージ数を指定します。

event

登録されたメッセージが受信された際、APP(アプリケーションプログラム)に通知するイベントです。本イベントは Win32API の CreateEvent() を使ってシステムから取得してください。値が=0 の場合は、通知イベントが無いことを意味します。

#### (3) 戻り値

なし

#### (4) 説明

EngRegisterDefaultSxFy() から呼び出され、APP が処理する 1 次メッセージの登録と受信したメッセージを貯えるためのキューの作成を行います。

本関数は、先に説明しました EngRegisterDefaultSxFy() 関数によって使用されます。APP で直接使用することはありません。

### 3.1.8 EngRegSecsMsg() – APP が処理する 1 次メッセージの登録

#### (1) 呼出書式

```
API int APIX EngRegSecsMsg(
    int      stream,          // Sx メッセージストリーム
    int      function,       // Fy メッセージファンクション
    int      q_flag,         // キューフラグ
    int (WINAPI *msgproc)()  // メッセージ処理関数名
);
```

#### [.NET VB]

```
Function EngRegSecsMsg (
    ByVal s As Int32,
    ByVal f As Int32,
    ByVal queue_flag As Int32,
    ByVal callback As vcallback.callback_RegSecsMsg,
    ByVal upara As Int32) As Int32
```

#### [.NET C#]

```
void EngInitSecsMsgReq(
    int count,
    uint evt );
```

#### (2) 引数

stream

登録したい SECS メッセージのストリームです。

function

登録したい SECS メッセージのファンクションです。

q\_flag

受信時にキューに入れるかどうかを指定します。

=0 がキューに入れない、=1 がキューに入れる指定になります。

func

受信メッセージを処理する関数名です。値=0 の場合は、処理する関数がないことを意味します。

#### (3) 戻り値

戻り値	意味
0	登録が正常にできた。
-1	登録できなかった。

#### (4) 説明

APP が DSHeng4 に対し装置から受信した 1 次メッセージの中で渡して欲しい受信 1 次メッセージのメッセージ ID を DSHeng4 に登録するための関数です。

q\_flag は受信時に受信キューに入れるかどうかを指定します。EngGetSecsMsgReq()関数を使ってメッセージをキューから取り出し処理する場合は、=1 を指定します。

func は通常、受信キューに入れないで、即時処理をしたい場合、その処理関数名を指定します。EngGetSecsMsgReq()関数を使ってキューから取り出して処理する場合は、=0 を指定します。通常は =1 を指定します。

本関数は、先に説明しました EngRegisterDefaultSxFy()関数によって使用されます。  
APP で直接使用することはありません。

3 . 1 . 9	DshCkBkupEcInfo()	-	バックアップ EC 情報状況確認
.	DshCkBkupSvInfo()	-	SV
.	DshCkBkupDvInfo()	-	DVVAL
.	DshCkBkupRpInfo()	-	REPORT
.	DshCkBkupCeInfo()	-	CE(収集イベント)
.	DshCkBkupPpInfo()	-	PP(プロセッサのログラム)
.	DshCkBkupFppInfo()	-	FPP(書式付プロセッサのログラム)
.	DshCkBkupRcpInfo()	-	RCP(レジスタ)
.	DshCkBkupCarInfo()	-	CAR(キャリア)
.	DshCkBkupSubstInfo()	-	SUBST(基板)
.	DshCkBkupCjInfo()	-	CJ(コントロールシヨブ)
.	DshCkBkupPrjInfo()	-	PRJ(プロセッサシヨブ)

( 1 ) 呼出書式

[C, C++]

```
API int APIX DshCkBkupEcInfo(
    char *bkup_dir, // バックアップ ファイル保存ディレクトリ
    int up_flag, // 更新フラグ
    TBACKUP_INFO *info // バックアップ 状況情報格納構造体のポインタ
);
```

[.NET VB]

```
Function DshCkBkupEcInfo (
    ByVal dir As String,
    ByVal up_flag As Int32,
    ByRef info As dsh_info.TBACKUP_INFO) As Int32
```

[.NET C#]

```
int DshCkBkupEcInfo(
    byte[] dir,
    int up_flag,
    ref TBACKUP_INFO info );
```

( 2 ) 引数

bkup\_dir

バックアップファイルが保存されているディレクトリ名です。(ドライブ名も含む)

up\_flag

正規(最新)のファイルに情報が正しく保存されていなく、前の世代のバックアップ情報が正常である場合、前の世代のバックアップファイルを正規のファイルに戻すかどうかを指定します。

up\_flag が 0 以外の場合、前の世代のファイルを正規ファイルに戻します。up\_flag = 0 の場合は戻しません。

正規のファイルが正常の場合、up\_flag は使用されません。

info

確認した状況を info で指定された TBACKUP\_INFO 構造体に設定して返却します。これは、関数の戻り値が = 0 の場合だけ有効です。

TBACKUP\_INFO 構造体の内容は以下のとおりです。

```
typedef struct{
```

```

char fname[256];           // 正規のファイル名
char fname_old[256];      // 戻した古いファイル名
char time_stamp[32];      // 正規のファイル名のタイムスタンプ (YYYYMMDDHHNNSSCC)
int  rec_count;           // 含まれている情報の合計数
int  file_index;          // 戻したファイルの世代番号 (0,1,2 or 3 )
} TBACKUP_INFO;

```

正規のファイルが正常であった場合、fname と fname\_old は同じになり、file\_index=0 となります。世代番号は0が正規ファイル、0以外はfile\_indexの値が示す前の世代を意味します。

### (3) 戻り値

戻り値	意味
0	確認結果が正常であった。
1,2 or 3	正規ファイルが正常でなく、戻り値が示す前の世代のものが正常であった。
100	EI_NO_FILE、バックアップファイルが無かった。

### (4) 説明

これらの関数は、前回エンジンが起動され動作し、終了した時に、先に延べた管理情報が正常にバックアップファイルに保存されたかどうかを確認するための関数です。

ユーザは、次にエンジン起動する前に本関数を使うことによってバックアップ状況を知ることができます。もし、正しく保存されていないケースではバックアップ情報を復元するかどうかの判断をこれら関数の結果を使って行うことができます。

バックアップ情報ファイルは各情報について最新のものを含め最大4世代分が保存されます。たとえば、装置状態変数SVの場合、バックアップファイル名は世代別につぎのようになります。

(前の世代のファイル名には世代番号が付きます。)

```

世代-0 : 最新のバックアップファイル   ec_bkup.bkp
  1    : 1つ前の世代                   ec_bkup1.bkp
  2    : 2つ前の世代                   ec_bkup2.bkp
  3    : 3つ前の世代                   ec_bkup3.bkp

```

本関数は、世代-0から順に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をチェックします。もし、世代-0のものが正しかった場合、info内にそのファイルに関する情報を設定し、戻り値=0を返します。(注 - 変数などの値の正当性のチェックは含まれません。)

もし、世代-0が正しく保存されていない場合、1つ前の世代-1のファイルをチェックします。このファイルが正しかった場合、このファイル名がfname\_oldに設定され、世代番号=1が設定されます。そして、up\_flagの値が=0以外の場合、世代-1のファイルを正規のファイル(=世代-0)にします。すなわち、ec\_bkup1.bkpをec\_bkup.bkpファイル名に変えます。

そしてもし、世代-1も正常でない場合、次の世代-2のチェックを行います。

正しいファイルが見つかるまで順にチェックを行います。そして存在している世代の全ファイルが正しくない場合は、関数戻り値=(-1)を返却します。

それから、もし、全世代のバックアップファイルが存在しなかった場合、戻り値としてEI\_NO\_FILE(=100)を返却します。

### 3.1.13 EngSetDataFormat() – メッセージデータアイテムのフォーマットの設定

#### (1) 呼出書式

##### [C, C++]

```
API int APIX EngSetDataFormat(
    int index,           // 設定したいデータアイテムのインデックス値を指定
    int format          // フォーマット(ICODE_??)
);
```

##### [.NET VB]

```
Function EngSetDataFormat (
    ByVal x As Int32,
    ByVal format As Int32) As Int32
```

##### [.NET C#]

```
int EngSetDataFormat(
    int index,
    int format
);
```

#### (2) 引数

index

データアイテム設定のためのインデックス値です。ここで設定するデータアイテムはシステムによって予約されており、DSHEng4 内部で使用されます。

インデックス値、そしてそのC言語のマクロとデータアイテムの対応表を下に示します。C言語のマクロは、user\_def.h に定義されています。

index 値	マクロ名	データアイテム	デフォルトフォーマット
0	X_FMT_ALID	ALID - A5F1	ICODE_U4
1	X_FMT_VID	VID	ICODE_U4
2	X_FMT_ECID	ECID	ICODE_U4
3	X_FMT_DVID	DVID	ICODE_U4
4	X_FMT_SVID	SVID	ICODE_U4
5	X_FMT_LIMITID	LIMITID - S2F45	ICODE_B
6	X_FMT_TRID	TRID - S2F23, S6F1	ICODE_A
7	X_FMT_SMP LN_	SMP LN - S6F1	ICODE_U2
8	X_FMT_RPID	RPTID - S2F33, S6F11 etc	ICODE_U4
9	X_FMT_CEID	CEID - S6F11	ICODE_U4
10	X_FMT_DATAID	DATAID - S6F11 etc	ICODE_U4
11	X_FMT_PPID	PPID - S7F3, S7F23 etc	ICODE_A
12	X_FMT_PPBODY	PPBODY - S7F3 etc	ICODE_A
13	X_FMT_LENGTH	プロダクト問合せ S7F1	ICODE_U4
14	X_FMT_PTN	PTN - S3F17, F25	ICODE_U1
15	X_FMT_DATALENGTH	DATALENGTH - S6F5	ICODE_U4
16	X_FMT_OPID	OPID - S14F19, F21	ICODE_U4
17	X_FMT_ERRCODE	ERRCODE - S14, S15, S16	ICODE_U1

format

DSHDR2 通信ドライバーで定義するフォーマットの表記を使ってフォーマットを指定します。

(3) 戻り値

戻り値	意味
0	正常に設定できた。
(-1)	インデクス値が間違っている。

(4) 説明

DSHEng4 が内部でメッセージのエンコード / デコードのために使用するデータアイテムを設定します。

(2) の表のデフォルトフォーマットの値がシステム初期化時に設定されます。変更の必要がないものについては何もする必要がありません。

本関数は、RegSetupLibrary()関数の直後で、EngSetupEquipment()関数の前に実行してください。

(別途設定変更したいものがあれば追加可能です。)

### 3.1.14 EngGetDataFormat() – メッセージデータアイテムのフォーマットの取得

#### (1) 呼出書式

##### [C, C++]

```
API int APIX EngSetDataFormat(
    int index // 取得したいデータアイテムのインデックス値を指定
);
```

##### [.NET VB]

```
Function EngGetDataFormat (
    ByVal x As Int32) As Int32
```

##### [.NET C#]

```
int EngSetDataFormat(
    int index
);
```

#### (2) 引数

index

取得したいデータアイテムのためのインデックス値です。

値とデータアイテムについては、3.1.10のEngSetDataFormat()関数の説明を参照ください。

#### (3) 戻り値

戻り値	意味
>=0	正常に取得できた。戻り値がフォーマットの値です。
(-1)	インデックス値が間違っている。

#### (4) 説明

index で指定したデータアイテムのフォーマットを取得します。

値は、DSHDR2 通信ドライバーが使用するフォーマット値を返却します。

### 3.2 通信制御関連関数

SECS 通信メッセージに関連するサービス用 API 関数名は一覧表のとおりです。

	関数名	機能	備考
1	EngEnable()	通信 Enable(S1F13 通信確立手続きの要求) にします。	S1F13,14
2	EngCancelEnable()	先に行った EngEnable() をキャンセルします。	
3	EngDisable()	通信 Disable 状態にします。	
4	EngSendPrimary()	1 次メッセージを送信します。(応答受信も含めて)	
5	EngGetSecsMsgReq()	APP 側で処理する受信 1 次メッセージを取得します。	Polling 用関数です。

### 3.2.1 EngEnable() – 通信可能状態の確立要求

#### (1) 呼出書式

##### [C, C++]

```
API int APIX EngEnable(
    int (WINAPI *EnableCallback)(), // 実行終了時の callback 関数
    ULONG upara // callback 時のパラメータ
);
```

##### [.NET VB]

```
Function EngEnable (
    ByVal callback As vcallback.callback_Enable,
    ByVal upara As Int32) As Int32
```

##### [.NET C#]

```
int EngEnable(
    CallbackDefault callback,
    uint upara );
```

#### (2) 引数

##### EnableCallback

DSHEng4 による Enable 処理が終了したときに呼出されるコールバック関数を指定します。  
ユーザは任意の関数名を使用できます。  
0 の指定の場合コールバック関数はなく、ブロックモードになります。  
コールバック関数の書式は (5) を参照してください。

##### upara

callback されたときに引数で渡してもらうためのパラメータです。  
callback 時に参照したい情報 (値、構造体などのポインタなど) を指定できます。

#### (3) 戻り値

戻り値	意味
0	(1) ブロックモード : 通信が COMMICATING 状態になった。 (S1F13, 14 の通信が正常に終了した。) (2) 非ブロックモード : 要求が受け付けられた。
1	現在 Enable 処理中であった。
2	EngCancelEnable() 関数によって Enable キャンセルされた。
(-1)	要求が受け付けられなかった。

#### (4) 説明

(6) の状態遷移図に示す装置との通信確立のための処理を要求します。  
結果はブロックモードの場合は戻り値で、そして非ブロックモードでは callback 関数の end\_status で与えられます。  
通信確立のための処理を行っている間 (未確立状態である) に EngCancelEnable() で通信確立要求の取消しを行うことができます。

#### (5) コールバック関数

##### [C, C++]

```
API int APIX EnableCallback(
```

```

int end_status,           // 実行結果
ULONG upara              // 呼出時に指定したパラメータです
);

```

**[VB]**

Function callback\_Enable(ByVal end\_status As Integer, ByVal upara As Integer) As Integer

**[C#]**

```

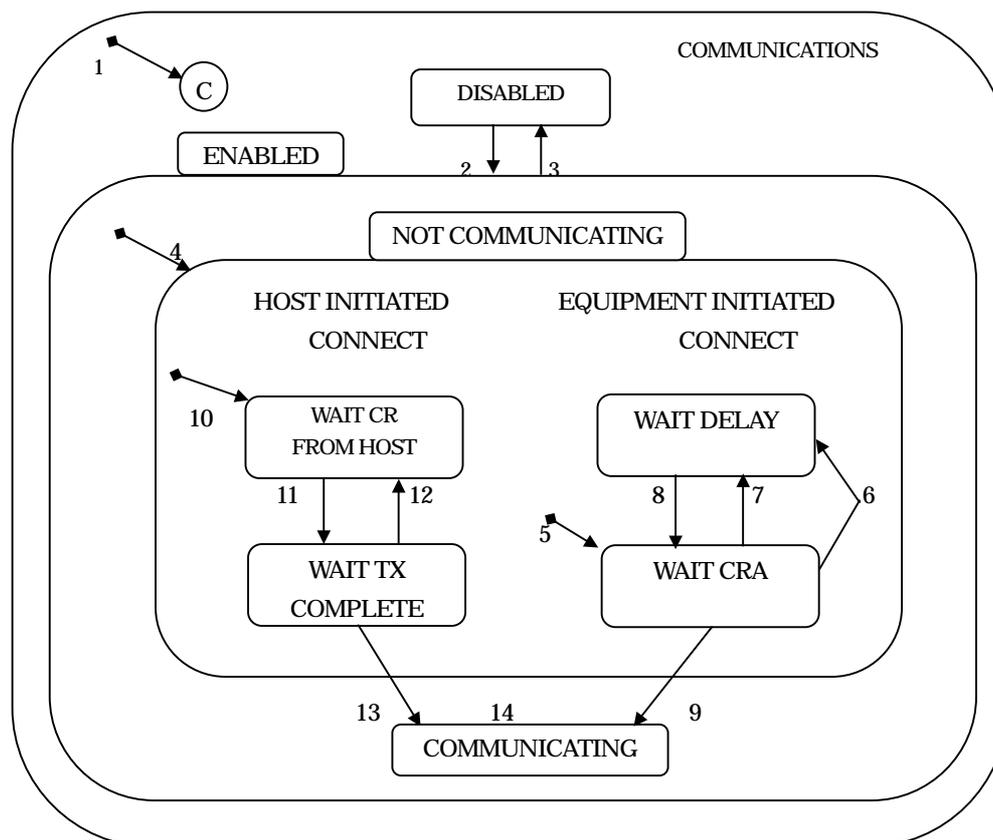
int CallbackDefault(int end_status, uint upara);

```

end\_status には以下の値が設定されます。

結果	意味
0	通信が COMMUNICATING 状態になった。(S1F13,14 の通信が正常に終了した。)
1	現在 Enable 処理中であった。
2	EngCancelEnable()関数によって Enable がキャンセルされた。
(-1)	要求が受け付けられなかった。

(6) 通信確立のための状態遷移図

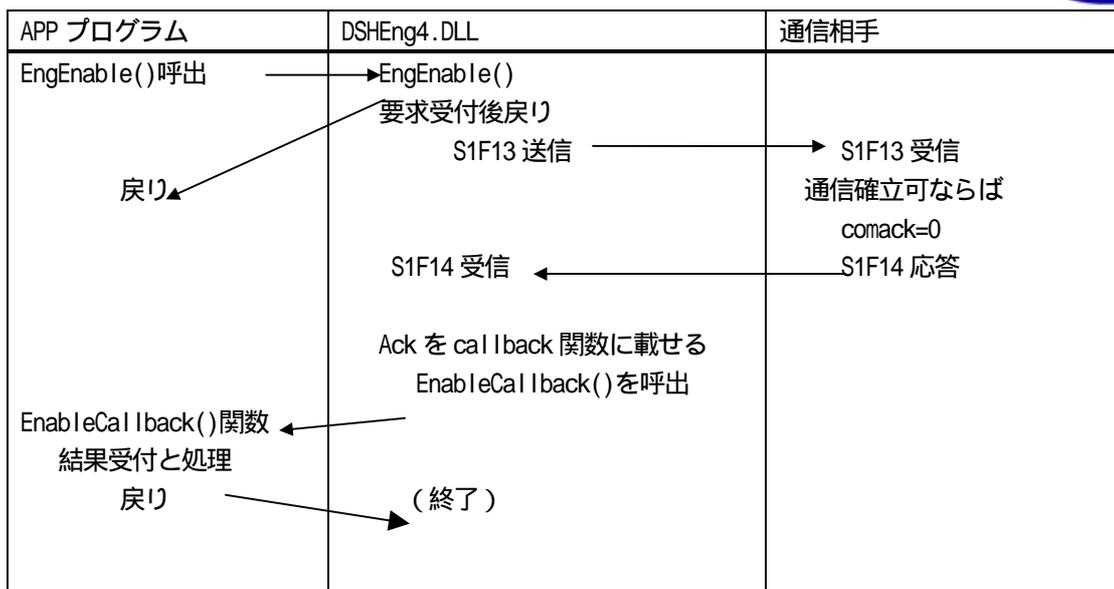


(7) 補足

DSHEng4 に対し、装置との通信を可能にするための処理を要求します。

プロトコル確立が成立していることが条件になります。HSMS の場合は、Selection が確立していることが前提になります。

CENG\_DLL との関連と制御の流れは次のチャートようになります。(上から下に処理が進みます。)



### 3.2.2 EngCancelEnable() – 通信可能状態確立要求の取消し

(1) 呼出書式

[C, C++]

```
API int APIX EngCancelEnable();
```

[.NET VB]

```
Function EngCancelEnable () As Int32
```

[.NET C#]

```
int EngCancelEnable();
```

(2) 引数

なし。

(3) 戻り値

戻り値	意味
0	取消しが正常に終了した。
1	既に確立していた。(COMMUNICATING 状態)

(4) 説明

EngEnable()関数で要求した通信状態確立要求を取り消すための関数です。

通信状態確立処理の状態によって戻り値が変わります。

EngEnable()関数が実行されていなかった場合、またはEnableの処理を取り消すことができた場合には、=0が返却されます。

既に通信状態が確立していた場合は、=1が返却されます。

強制的にEnable状態をやめたい場合はEngDisable()関数を実行してください。

### 3.2.3 EngDisable() – 通信中断要求

#### (1) 呼出書式

##### [C, C++]

```
API int APIX EngDisable(
    TCEID ceid, // 中断通知 ceid( 使用されません)
    int (WINAPI *DisableCallback)(), // 実行終了時の callback 関数
    ULONG upara // callback 時のパラメータ
);
```

##### [.NET VB]

```
Function EngDisable (
    ByVal ceid As Int32,
    ByVal callback As vcallback.callback_Disable,
    ByVal upara As Int32) As Int32
```

##### [.NET C#]

```
int EngDisable(
    uint ceid,
    CallbackDefault DisableCallback,
    uint upara );
```

#### (2) 引数

ceid

本引数は DSHEng4 では使用されません。従ってどんな値でもかまいません。

DisableCallback

DSHEng4 による Disable 処理が終了したときに呼出される callback 関数を指定します。

ユーザは任意の関数名を使用できます。

0 の指定の場合コールバック関数はなく、ブロックモードになります。

コールバック関数の書式は (5) を参照のこと。

upara

callback されたときに引数で渡してもらうためのパラメータです。

callback 時に参照したい情報 ( 値、構造体など ) を指定します。

#### (3) 戻り値

戻り値	意味
0	(1) ブロックモード : 通信中断状態になった。 (2) 非ブロックモード : 要求が受け付けられた。
(-1)	要求が受け付けられなかった。

#### (4) 説明

S1F13, 14 のやりとりで確立された通信状態を、通信中断状態に移行します。

通信を中断した以降に装置から受信した 1 次メッセージに対して、DSHEng4 は function = 0 の 2 次メッセージを応答することになります。

再び、通信可能にするためには、EngEnable() 関数で通信確立を行う必要があります。

#### (5) コールバック関数

##### [C, C++]

```
API int APIX DisableCallback(  
    int end_status,           // 実行結果  
    ULONG upara              // 呼出時に指定したパラメータ  
);
```

**[VB]**

Function callback\_Disable(ByVal end\_status As Integer, ByVal upara As Integer) As Integer

**[C#]**

int CallbackDefault(int end\_status, uint upara);

end_status 値	意味
0	通信中断状態になった。

### 3.2.4 EngSendPrimary() – 1次メッセージの送信要求

#### (1) 呼出書式

##### [C, C++]

```
API int APIX EngSendPrimary(
    DSHMSG *smsg,           // 送信メッセージ情報格納構造体ポインタ
    DSHMSG **rmsg,         // 受信メッセージ情報格納構造体ポインタの格納ポインタ
    int (WINAPI *PrimaryCallback)(), // 実行終了時のコールバック関数
    ULONG upara            // callback時のパラメータ
);
```

##### [.NET VB]

```
Function EngSendPrimary (
    ByRef smsg As dshdr2.DSHMSG,
    ByRef rmsg As IntPtr,
    ByVal callback As vcallback.callback_SendPrimary,
    ByVal upara As Int32) As Int32
```

##### [.NET C#]

```
int EngSendPrimary(
    ref DSHMSG smsg,
    ref DSHMSG As IntPtr,
    CallbackSendPrimary callback,
    uint upara );
```

#### (2) 引数

smsg

送信したい1次メッセージ情報が格納されている DSHMSG 構造体のポインタです。

rmsg

受信した2次メッセージ情報格納構造体のポインタを格納する領域のポインタを指定します。ブロックモード(callback関数=0)指定時に受信した2次メッセージ受信のためのパラメータです。rmsgの処理終了後、rmsgで使用されているメモリをDshFreeMsg()関数を使って解放してください。非ブロックモードまたは受信メッセージを受取る必要がない場合には NULL を指定してください。

PrimaryCallback

DSHEng4によるメッセージのトランザクションが終了したときに呼出されるコールバック関数を指定します。ユーザは任意の関数名を使用できます。

応答メッセージを受信した場合、コールバック関数の引数にメッセージ情報構造体のポインタが渡されます。

本パラメータとして=0が指定された場合は、コールバック関数が無く、ブロックモードとみなされ、関数で指定されたメッセージの通信トランザクションが終了するまで制御は戻ってきません。

upara

callbackされたときに使用するためのパラメータです。

関数実行終了時にコールバックされた際、何かの判別情報として使用したい値、構造体ポインタなどを設定することができます。

#### (3) 戻り値

戻り値	意味
-----	----

0	(1) ブロックモード：正常に送信できた。 rmsg に応答メッセージが格納されているポインタが返却される。 (2) 非ブロックモード：要求が受け付けられた。
9	スプールされた。
(-1)	要求が受け付けられなかったか送信エラーを検出した。
(-14)	T3タイムアウトを検出した。

#### (4) 説明

smsg に準備された SECS1 次メッセージを装置に送信するための関数です。

メッセージの生成は DSHDR2 ドライバーの D\_InitItemPut(), D\_PutItem() 関数を使います。

要求元は、本関数を要求する前に smsg 内に送信メッセージ情報を設定しなければなりません。

詳しくは DSHDR2 通信ドライバーのドキュメントを参照ください。

PrimaryCallback コールバック関数を指定するかどうかによってプログラムの制御モードが次のように変わります。

PrimaryCallback=0 の場合ブロックモードになります。

この場合、通信トランザクションが終了しないと本関数から制御が戻ってきません。終了結果は本関数の戻り値で返却されます。

正常に応答メッセージを受信して終了した場合、rmsg に応答メッセージ格納情報体のポインタが渡されます。rmsg のメッセージの処理が終了した後、APP プログラム側の責任で rmsg と内部で使用されているメモリを DshFreeMsg(rmsg) 関数呼出しによって解放してください。

正常に終了しなかった場合 (3) の戻り値の 1 つが返却されます。

PrimaryCallback が=0 でない場合は非ブロックモードになります。

この場合、DSHEng4 は送信要求を受け付けた段階でただちに制御を戻します。従って戻ってきた時点ではまだ動作が終了していません。

終了結果は、PrimaryCallback 関数の引数 end\_status として渡されます。

正常に終了した場合、受信メッセージ情報は PrimaryCallback 関数の rmsg に渡されます。

コールバック関数で渡される rmsg は APP がメッセージを処理したあと DshFreeMsg() 関数で解放してください。

本関数の終了情報はブロックモードの場合は、関数の戻り値で与えられ、非ブロックモードの場合は、この後で説明される コールバック関数の引数 end\_status に終了情報が渡されます。

もし通信相手との通信接続が確立していない状態で本関数が実行された場合は終了情報として (-1) が返されます。

#### (5) コールバック関数の呼出書式

##### [C, C++]

```
API int APIX PrimaryCallback(
    int end_status,           // 終了結果
    DSHMSG *rmsg,            // 応答メッセージ用の rmsg です。
    ULONG upara              // 呼出時に指定したパラメータ
);
```

##### [.NET VB]

```
Function callback_SendPrimary(ByVal end_status As Integer, ByRef rmsg As dshdr2.DSHMSG, ByVal upara As Integer) As Integer
```

##### [.NET C#]

```
int CallbackSendPrimary(int end_status, ref DSHMSG rmsg, uint upara );
```

( 6 ) コールバック関数の引数

end\_status

EngSendPrimary 関数の終了結果が渡されます。

rmsg

2 次メッセージ情報が格納されている構造体のポインタが渡されます。

終了結果 end\_status が 0 の時にだけ本 rmsg が有効です。

APP プログラムはメッセージの処理後、DshFreeMsg()関数を使って rmsg に使用されているメモリを開放してください。

upara

EngSendPrimary()関数で指定したものがそのまま渡されます。

end\_status には以下の値が設定されます。

結果	意味
0	正常に送信できた。rmsg に応答メッセージが格納されているポインタが返却される。
9	スプールされた。
(-1)	送信エラーを検出した。
(-14)	T3 タイムアウトを検出した。

### 3.2.5 DshResponseSxFy() - 2次メッセージの送信

#### (1) 呼出書式

##### [C, C++]

```
API int APIX DshResponseSxFy(
    ID_TR      trid,           // EngGetSecsMsgReq で得られたトランザクション ID
    DSHMSG     *pmsg,         // 応答メッセージ情報格納構造体のポインタ
    int (WINAPI *Callback)() // 実行終了時のコールバック関数
);
```

##### [.NET VB]

```
Function DshResponseSxFy (
    ByVal trid As Int32,
    ByRef prmsg As dshdr2.DSHMSG,
    ByRef callback As Int32) As Int32
```

##### [.NET C#]

```
int DshResponseSxFy(
    uint trid,
    ref DSHMSG prmsg,
    eng_api.CallbackSendResponse callback );
```

#### (2) 引数

trid

EngGetSecsMsgReq()関数で受信キューから得られたメッセージ構造体 TMSG\_QUEUE\_INFO の trid メンバーに与えられた通信トランザクション ID です。

rmsg

応答したい2次メッセージが格納されている構造体のポインタを指定します。

callback

DSHEng4 によるメッセージのトランザクションが終了したときに呼出されるコールバック関数を指定します。ユーザは任意の関数名を使用できます。

#### (3) 戻り値

戻り値	意味
0	正常

#### (4) 説明

装置から受信した1次メッセージに対する2次メッセージの送信を行うための関数です。

ユーザは rmsg メッセージ情報構造体の中に DSHDR2 ドライバの D\_InitItemPut(), D\_PutItem()関数を使って2次メッセージを作成します。

その後、EngGetSecsMsgReq()関数で受信キューから得られたメッセージ構造体 TMSG\_QUEUE\_INFO の trid メンバーに与えられた通信トランザクション ID, rmsg, callback 引数を伴って本関数を実行します。callback で与えられる引数は、DSHEng4 デーモンに要求が渡された時点でコールバックされる関数のエントリです。

本関数のサンプルは Eng\_User.DLL のソースプログラム(u\_s2f41.c など)を参照してください。

#### (5) コールバック関数の呼出書式

##### [C, C++]

```
API int APIX callback(DSHMSG *rmsg );
```

#### [.NET VB]

```
Function callback_SendResponse(ByRef rmsg As dshdr2.DSHMSG) As Integer
```

#### [.NET C#]

```
int CallbackSendResponse(ref DSHMSG rmsg);
```

ここで、rmsg は、DshResponseSxFy()関数で与えたメッセージ情報構造体のポインタです。  
次のように、rmsg を開放して戻ります。

```
API int APIX callback(DSHMSG *rmsg )  
{  
    DshFreeMsg( rmsg );  
}
```

(注) この場合、rmsg のメモリはプロセスのヒープから確保されたことが前提です。

VB では、eng\_lib.DshFreeMsg( rmsg )

C#では、eng\_lib.DshFreeMsg( ref rmsg );

### 3.2.6 EngGetSecsMsgReq() – APP が処理する 1 次メッセージを取出す

#### (1) 呼出書式

##### [C, C++]

```
API int APIX EngGetSecsMsgReq(
    TMSG_QUEUE_INFO *msg           // 取得した情報を格納する領域のポインタ
);
```

##### [.NET VB]

```
Function EngGetSecsMsgReq (
    ByRef msg As dsh_info.TMSG_QUEUE_INFO) As Int32
```

##### [.NET C#]

```
int EngGetSecsMsgReq(
    ref TMSG_QUEUE_INFO msg );
```

#### (2) 引数

msg

キューに受信したメッセージがあった場合に、そのメッセージ情報を格納するための領域です。

#### (3) 戻り値

戻り値	意味
0	キューに受信したメッセージが無かった。
1	受信したメッセージを取得できた。

#### (4) 説明

EngRegisterDefaultSxFy()関数で登録した 1 次メッセージを受信したかどうかを調べ、もしあれば、キューからその情報を取得するための関数です。

msg は以下の構造体のポインタです。

```
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;       // 応答 msg のポインタ
    int      (WINAPI *callback)(); // 終了時に呼び出す関数(=0 ならば関数が無い) - 未使用です。
} TMSG_QUEUE_INFO;
```

APP は、smsg に格納されている構造体のメッセージを処理することになります。  
標準的な処理の方法は、デモプログラムのソースファイルを参考にしてください。

それから、受信した 1 次メッセージを処理した後、2 次メッセージを応答することになります。  
2 次メッセージには、ヘダーのみのも、Ack(1 バイト)だけ持つもの、LIST 構造を持つ応答情報を持つものがあります。

リスト構造を持つ複雑な応答情報を含むメッセージの生成のために、DSHEng4 は、応答情報を構造体に詰めるための関数、ならびにそれから、実際に応答するメッセージを生成する関数も準備しています。

ともかく、APP は eng\_lib(.h, .vb, .cs)に含まれるライブラリ関数を使って応答情報を作成し、メッセ

ー지를組み立てて応答メッセージを送信する処理を行います。

デモプログラムでは、APP 側で応答情報を準備し、それを DSHEng4U.dll ユーザ作成 DLL に準備されている応答関数に渡して送信してもらうようにしています。

それから、受信したメッセージの処理が終わった後、本関数で得られた TMSG\_QUEUE\_INFO 構造体の中の smsg メンバー (DSHMSG 構造体) のメモリを DshFreeMsg() 関数を使って開放してください。

(例) `DshFreeMsg( msg->smsg );`

### 3.2.7 EngGetEqCommState() – 装置通信状態の取得

(1) 呼出書式

**[C, C++]**

```
API int APIX EngGetEqCommState();
```

**[.NET VB]**

```
Function EngGetEqCommState () As Int32
```

**[.NET C#]**

```
int EngGetEqCommState ();
```

(2) 引数

なし。

(3) 戻り値

戻り値	意味
5	ST_COMMUNICATING, 通信確立状態です。
その他	未確立状態

(4) 説明

本 API 関数はあるいはホストが相手装置との間で通信確立されているかどうかを示す状態値を取得します。通信の確立は、EngEnable() API 関数で相手に S1F13 メッセージを使って行われます。S1F13 に対する S1F14 で commack = 0 の応答を受信したとき通信確立されます。

戻り値 = 5、(ST\_COMMUNICATING) であれば、通信確立がなされていることを意味します。  
GEM レベルでメッセージの通信が可能ということです。

戻り値 = その他(5 以外)の場合は、通信が未確立であることを意味します。

### 3.2.8 EngGetCommPortStatus() – 通信ポートプロトコル状態の取得

(1) 呼出書式

**[C, C++]**

```
API int APIX EngGetCommPortStatus ();
```

**[.NET VB]**

```
Function EngGetEqCommPort () As Int32
```

**[.NET C#]**

```
int EngGetEqCommPort();
```

(2) 引数

なし。

(3) 戻り値

戻り値	意味
0	Selected 状態
1	Not Selected 状態
(-1)	エンジンが起動されていない。

(4) 説明

本 API 関数は通信エンジンが使用する通信ポートの HSMS-SS プロトコル通信状態を取得するために使用します。

戻り値の値によって Selection が確立しているかどうかを判断することができます。

戻り値 = 0 は、Selection が確立されている状態を表し、HSMS-SS プロトコル上で、相手装置とのメッセージの送受信ができることを意味します。

戻り値 = 1 は、Selection が未確立であり、HSMS-SS プロトコルの接続待ちを意味しています。

戻り値 = (-1) は、DSHEng4 がまだ起動されていないことを意味します。

通信プロトコルとして SECS-I が採用されて場合、DSHEng4 が起動されていれば無条件に =0 が返却されます。