

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- 6 / 1 5

- 3 . 11 PP プロセスプログラム情報アクセス関数
- 3 . 12 書式付プロセスプログラム情報アクセス関数

2 0 0 9年7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	

目 次

3.11	PP プロセスプログラム情報アクセスサービス関数	1
3.11.1	使用する情報格納構造体	3
3.11.2	PP プロセスプログラム情報アクセス関数	4
3.11.2.1	EngAllocPpInfo0 - プロセスプログラムの登録	4
3.11.2.2	EngSetPpInfo0 - プロセスプログラム情報の設定	5
	EngSetPpInfoX0 - インデクス指定によるプロセスプログラム情報の設定	5
3.11.2.3	EngGetPpInfo0 - プロセスプログラムの取得	7
	EngGetPpInfoX0 - インデクス指定でのプロセスプログラムの取得	7
3.11.2.4	EngDelPpInfo0 - プロセスプログラムの削除	9
	EngDelPpInfoX0 - インデクスでのプロセスプログラムの削除	9
3.11.2.5	EngSetPpState0 - プロセスプログラム状態の設定	10
	EngSetPpStateX0 - インデクスでのプロセスプログラム状態の設定	10
3.11.2.6	EngGetPpState0 - プロセスプログラム状態の取得	12
	EngGetPpStateX0 - インデクスでのプロセスプログラム状態の取得	12
3.11.2.7	EngGetPpList0 - 全登録PPID取得関数	14
3.11.2.8	EngGetPpId0 - インデクスからPPID (プロセスプログラムID) の取得	15
3.11.2.9	EngGetPpIdIndex0 - PPID (プロセスプログラムID) からインデクスの取得	16
3.11.3	PP プロセスプログラム関連通信メッセージ送信要求関数	17
3.11.3.1	EngSendS7F10 - プロセスプログラムロード問合せメッセージ送信関数	17
3.11.3.2	EngSendS7F30 - プロセスプログラム情報メッセージ送信関数	19
3.11.3.3	EngSendS7F50 - プロセスプログラム情報問合せメッセージ送信関数	21
3.11.4	PP プロセスプログラム関連ライブラリ関数	23
3.11.4.1	DshDecodeS7F30 - S7F3 をプロセスプログラム情報にデコード	23
3.11.4.2	DshEncodeS7F30 - プロセスプログラム情報をS7F3へエンコード	24
3.11.4.3	DshFreeTPP_INFO0 - プロセスプログラム情報構造体メモリの開放	25
3.11.4.4	DshCopyTPP_INFO0 - プロセスプログラム情報構造体メモリのコピー	26
3.11.4.5	DshInitPpInfo - プロセスプログラムTPP_INFOの初期設定	27
3.11.4.6	DshMakeS7F3Response0 - S7F3の応答メッセージの生成	29
3.11.4.7	DshDecodeS7F170 - S7F17デコード - 削除プロセスプログラム一覧	30
3.11.4.8	DshFreeTPPID_LIST0 - プロセスプログラムID一覧情報構造体メモリの開放	31
3.11.4.9	DshDecodeS7F270 - S7F27をプロセスプログラム妥当性情報にデコード	32
3.10.4.11	DshFreeTPP_PVS_LIST0 - プロセスプログラム妥当性情報構造体メモリの開放	33
3.11.4	ユーザ作成ライブラリ関数	34
3.11.4.1	DshResponseS7F20 - S7F2 PPロード問合せ応答メッセージ	34
3.11.4.2	DshResponseS7F40 - S7F4 PP送信応答メッセージ	35
3.11.4.3	DshResponseS7F60 - S7F6 PP要求応答メッセージ	36
3.11.4.4	DshResponseS7F180 - S7F18 PP削除指示応答メッセージ	37
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	38
	書式付プロセスプログラムのパラメータを配列位置指定で追加します。	39
3.12.1	使用する情報格納構造体	40
3.12.2	FPP 書式付プロセスプログラム情報アクセス関数	41
3.12.2.1	EngAllocFppInfo0 - 書式付プロセスプログラムの登録	41
3.12.2.2	EngSetFppInfo0 - 書式付プロセスプログラム情報の設定	42
	EngSetFppInfoX0 - インデクス指定による書式付プロセスプログラム情報の設定	42
3.12.2.3	EngGetFppInfo0 - 書式付プロセスプログラムの取得	44
	EngGetFppInfoX0 - インデクス指定での書式付プロセスプログラムの取得	44

3 . 12 . 2 . 4	EngDelFppInfo()	- 書式付プロセスプログラムの削除	46
.	EngDelFppInfoX()	- インデクスでの書式付プロセスプログラムの削除	46
3 . 12 . 2 . 5	EngSetFppState()	- 書式付プロセスプログラム状態の設定	47
.	EngSetFppStateX()	- インデクスでの書式付プロセスプログラム状態の設定	47
3 . 12 . 2 . 6	EngGetFppState()	- 書式付プロセスプログラム状態の取得	49
.	EngGetFppStateX()	- インデクスでの書式付プロセスプログラム状態の取得	49
3 . 12 . 2 . 7	EngGetFppList()	- 全登録FPPID取得関数	51
3 . 12 . 2 . 8	EngGetFppId()	- インデクスからFPPID (書式付プロセスプログラムID) の取得	52
3 . 12 . 2 . 9	EngGetFppIdIndex()	-FPPID (書式付プロセスプログラムID) からインデクスの取得	53
.....			
3 . 12 . 3	FPP	書式付プロセスプログラム関連通信メッセージ送信要求関数	54
3 . 12 . 3 . 1	EngSendS7F23()	- 書式付プロセスプログラム情報送信関数	54
3 . 12 . 3 . 2	EngSendS7F25()	- 書式付プロセスプログラム情報問合せ関数	56
3 . 12 . 4	FPP	書式付プロセスプログラム関連ライブラリ関数	58
3 . 12 . 4 . 1	DshDecodeS7F23()	- S7F23 を書式付プロセスプログラム情報にデコード	58
3 . 12 . 4 . 2	DshEncodeS7F23()	- 書式付プロセスプログラム情報をS7F23へエンコード	59
3 . 12 . 4 . 3	DshFreeTFPP_INFO()	- 書式付プロセスプログラム情報構造体メモリの開放	60
3 . 12 . 4 . 4	DshCopyTFPP_INFO()	- 書式付プロセスプログラム情報構造体メモリのコピー	61
3 . 12 . 4 . 5	DshInitFppInfo	- 書式付プロセスプログラムTFPP_INFOの初期設定	62
3 . 12 . 4 . 6	DshPutFppCCode()	-書式付プロセスプログラムコマンドコード情報の追加	64
3 . 12 . 4 . 7	DshPutFppPPara()	-書式付プロセスプログラムパラメータ情報の追加	66
3 . 12 . 4 . 8	DshPutFppPParaInfo()	-書式付プロセスプログラムパラメータの配列指定追加	68
3 . 12 . 4 . 9	DshMakeS7F23Response()	- S7F23 の応答メッセージの生成	70
3 . 12 . 4	ユーザ作成ライブラリ関数		71
3 . 12 . 4 . 1	DshResponseS7F24()	- S7F24 書式付きPP送信応答メッセージ	71
3 . 12 . 4 . 2	DshResponseS726()	- S7F26 PP要求応答メッセージ	72

(VOL - 7に続く)

(2) ライブラリ関数

他に APP が使用できるプロセスプログラム情報処理用 API 関数として、以下の関数があります。

	API 関数名	機能
1	DshDecodeS7F3()	S7F3 のメッセージ内 PP 情報を TPP_INFO 構造体にデコードするための関数です。
2	DshEncodeS7F3()	TPP_INFO 構造体のプロセスプログラム情報を S7F3 メッセージにエンコードするための関数です。
3	DshFreeTPP_INFO()	PP 情報が格納されている TPP_INFO 構造体とその内部で使用されているメモリを開放するための関数です。
4	DshCopyTPP_INFO()	TPP_INFO のプロセスプログラム情報をコピーします。
5	DshInitPpInfo()	PP 情報構造体 TPP_INFO を初期設定します。
6	DshMakeS7F3Response()	S7F4 応答メッセージを生成し、送信するための関数。u_s7f3.c で使用します。
7	DshDecodeS7F17()	S7F17 を削除プロセスプログラム一覧情報構造体 TPPID_LIST 内にデコードします。
8	DshFreeTPPID_LIST()	TPPID_LIST 構造体内で使用されているメモリを開放します。
9	DshDecodeS7F27()	S7F27 メッセージ内のプロセスプログラム妥当性情報を TPP_PVS_LIST 構造体内にデコードします。
10	DshFreeTPP_PVS_LIST()	TPP_PVS_LIST 構造体内で使用されているメモリを開放します。

(3) ユーザ作成ライブラリ関数

	ライブラリ関数名	機能
1	DshResponseS7F2()	S7F2 プロセスプログラムロード問合せ応答
2	DshResponseS7F4()	S7F4 プロセスプログラム送信応答メッセージ
3	DshResponseS7F6()	S7F6 プロセスプログラム要求応答メッセージ
4	DshResponseS7F18()	S7F18 プロセスプログラム削除指示応答メッセージ

3.11.1 使用する情報格納構造体

(1) プロセスプログラム情報を操作する関数は、共通の情報格納のための TPP_INFO 構造体を使用します。

```
typedef struct{
    int    index;                // 情報格納場所インデックス
    int    state;                // PP の状態(APP が任意に決める)
    char   *name;                // pp 名(装置管理情報定義ファイルで与えられた名前)
    int    ppid_fmt;             // ppid のアイテムフォーマット( Format 10 ASCII が設定されます。)
    int    ppid_size;           // 同配列サイズ(実際の ppid のバイト長です。)
    int    max_ppid_size;       // 配列サイズの最大値
    char   *ppid;                // ppid 値格納ポインタ
    int    ppbody_fmt;          // ppbody のアイテムフォーマット( Format 10 ASCII が設定されます。)
    int    ppbody_size;         // 同配列サイズ(実際の ppbody のバイト長です。)
}TPP_INFO;                      // formatted process program
```

(2) PPID(プロセスプログラム ID)リスト格納用構造体 - S7F19 メッセージの応答メッセージデコード情報用

```
typedef struct{
    int    count;                // ppid_list に含まれる PPID の数
    char   **ppid_list;         // ppid が格納されているリスト
} TPPID_LIST;
```

(3) PP (プロセスプログラム)妥当性情報 - 装置からの S7F27 メッセージデコード情報用

```
typedef struct{
    char   *ppid;                // ppid
    int    err_count;           // 妥当性で検出されたエラー情報の数
    TPP_PVS_INFO **err_list;    // エラー情報のリスト
} TPP_PVS_LIST;

typedef struct{
    int    ackc7a;              // ack
    int    seqnum;              // 番号
    char   *errw7;              // エラー情報(文字列)
} TPP_PVS_INFO;
```

3.11.2 PP プロセスプログラム情報アクセス関数

3.11.2.1 EngAllocPpInfo() - プロセスプログラムの登録

(1) 呼出書式

[C, C++]

```
API int APIX EngAllocPpInfo(
    char *ppid,          // PPID 格納領域のポインタ
    int *index          // 得られた情報領域のインデックス格納用ポインタ
);
```

[.NET VB]

```
Function EngAllocPpInfo (
    ByVal ppid As String,
    ByRef index As Int32) As Int32
```

[.NET C#]

```
int EngAllocPpInfo(
    byte[] ppid,
    ref int index );
```

(2) 引数

ppid

登録したいプロセスプログラムの ID が格納されているポインタです。

index

登録された情報領域のインデックス値が格納される領域のポインタです。

(3) 戻り値

戻り値	意味
0	正常に登録できた。
1	指定された PPID は既に登録されていた。
(-1)	登録できなかった。

(4) 説明

プロセスプログラムを新規にシステムに登録するための関数です。

引数 ppid で与えられるプロセスプログラム ID をシステムに登録します。

正常に登録できた場合は、index で指定される領域に登録された情報領域のインデックスが設定返却されます。

もし、ppid に指定されたプロセスプログラムが既に登録済みであった場合には関数の戻り値 1 を返却します。index には既に登録されている情報領域のインデックスが設定されます。

得られたインデックスを使って、情報の設定、取得、削除などのアクセスを行うことができます。

3.11.2.2 EngSetPpInfo() - プロセスプログラム情報の設定 EngSetPpInfoX() - インデクス指定によるプロセスプログラム情報の設定

(1) 呼出書式

[C, C++]

```
API int APIX EngSetPpInfo(
    TPP_INFO *pinfo           // PP 情報格納構造体のポインタ
);

API int APIX EngSetPpInfoX(
    int index,               // EngAllocPpInfo() で得られたインデクス値
    TPP_INFO *pinfo         // PP 情報格納構造体のポインタ
);
```

[.NET VB]

```
Function EngSetPpInfo (
    ByRef pinfo As dsh_info.TPP_INFO) As Int32

Function EngSetPpInfoX (
    ByVal index As Int32,
    ByRef pinfo As dsh_info.TPP_INFO) As Int32
```

[.NET C#]

```
int EngSetPpInfo(
    ref TPP_INFO pinfo );

int EngSetPpInfoX(
    int index,
    ref TPP_INFO pinfo );
```

(2) 引数

pinfo

設定したいプロセスプログラム情報が格納されている格納構造体領域のポインタです。

index

PPID 情報のインデクスです。登録時に EngAllocPpInfo() 関数によって与えられます。

インデクスは、PPID から EngGetPpIdIndex() 関数で取得することができます。

(3) 戻り値

戻り値	意味
0	正常に設定できた。
(-1)	設定できなかった。

(4) 説明

本関数は、pinfo に格納されているプロセスプログラム情報の設定・変更に使います。index 値の指定によっても設定できます。

引数 pinfo 内の ppid メンバーに指定されるプロセスプログラム ID の情報として設定されます。

pinfo 内には PPID の他、 PPBODY 情報が含まれます。
指定した PPID が既に登録済みである場合、 pinfo 内の情報にすべて書き換えられます。

pinfo 内の PPID が未登録であった場合は、登録手続きをしてから PP 情報を設定します。
(EngAllocPpInfo()関数で行われる登録と同じ登録が行われます。)

3.11.2.3 EngGetPpInfo() - プロセスプログラムの取得 EngGetPpInfoX() - インデクス指定でのプロセスプログラムの取得

(1) 呼出書式

[C, C++]

```
API int APIX EngGetPpInfo(
    char *ppid, // PPID が格納されている領域のポインタ
    TPP_INFO *pinfo // PP 情報を格納する構造体ポインタ
);
```

```
API int APIX EngGetPpInfoX(
    int index, // PPID 情報のインデクス
    TPP_INFO *pinfo // PP 情報を格納する構造体ポインタ
);
```

[.NET VB]

```
Function EngGetPpInfo (
    ByVal ppid As String,
    ByRef pinfo As dsh_info.TPP_INFO) As Int32
```

```
Function EngGetPpInfoX (
    ByVal index As Int32,
    ByRef pinfo As dsh_info.TPP_INFO) As Int32
```

[.NET C#]

```
int EngGetPpInfo(
    byte []ppid,
    ref TPP_INFO pinfo );
```

```
int EngGetPpInfoX(
    int index,
    ref TPP_INFO pinfo );
```

(2) 引数

ppid

PPID が格納されている領域のポインタです。

pinfo

取得した PP 情報を格納する構造体領域のポインタです。

index

PPID 情報のインデクスです。登録時に EngAllocPpInfo()関数によって与えられます。
インデクスは、PPID から EngGetPpIdIndex()関数で取得することができます。

(3) 戻り値

戻り値	意味
0	正常に取得できた。
(-1)	取得できなかった。(未登録であった)

(4) 説明

ppid または index に指定されている PPID のプロセスプログラム情報を pinfo で指定されたメモリ領域に取得格納します。

TPP_INFO 構造体の中に情報を格納するために必要なメモリは、DSHEng4 が準備確保します。即ち、構造体のメンバーの中でポインタになっている情報の実体即ち、ppid, ppbody のためのメモリは DSHEng4 が準備します。

これらのメモリは、使用后、ユーザが DSHEng4 の API 関数を使って次のように開放してください。

```
TPP_INFO  *pinfo;

if ( EngGetPpInfo( ppidinfo ) == 0 ){
    pinfo の処理
    処理終了後
    DshFreeTPP_INFO( pinfo );          // pinfo 内に使用されているメモリの開放
}
```

3.11.2.4 EngDelPpInfo() - プロセスプログラムの削除 EngDelPpInfoX() - インデクスでのプロセスプログラムの削除

(1) 呼出書式

[C, C++]

```
API int APIX EngDelPpInfo(
    char *ppid; // PPID が格納されている領域のポインタ
);

API int APIX EngDelPpInfoX(
    int index // インデクス(0,1,2,...)
);
```

[.NET VB]

```
Function EngDelPpInfo (
    ByVal ppid As String) As Int32

Function EngDelPpInfoX (
    ByVal index As Int32) As Int32
```

[.NET C#]

```
int EngDelPpInfo(
    byte[] ppid );

int EngDelPpInfoX(
    int index );
```

(2) 引数

ppid

PPID が格納されている領域のポインタです。

index

PPID 情報のインデクスです。登録時に EngAllocPpInfo() 関数によって与えられます。
インデクスは、PPID から EngGetPpIdIndex() 関数で取得することができます。

(3) 戻り値

戻り値	意味
0	正常に削除できた。
(-1)	未登録であった。

(4) 説明

ppid または index に指定されている PPID または、index 指定されたプロセスプログラム情報をシステムの登録から削除します。

3.11.2.5 EngSetPpState() - プロセスプログラム状態の設定 EngSetPpStateX() - インデクスでのプロセスプログラム状態の設定

(1) 呼出書式

[C, C++]

```
API int APIX EngSetPpState(
    char    *ppid,           // PPID が格納されている領域のポインタ
    int     state           // 設定したい状態値
);
```

```
API int APIX EngSetPpStateX(
    int     index,          // PPID 情報のインデクス
    int     state           // 設定したい状態値
);
```

[.NET VB]

```
Function EngSetPpState (
    ByVal ppid As String,
    ByVal state As Int32) As Int32
```

```
Function EngSetPpStateX (
    ByVal index As Int32,
    ByVal state As Int32) As Int32
```

[.NET C#]

```
int EngSetPpState(
    byte[] ppid,
    int state);
```

```
int EngSetPpStateX(
    int index,
    int state );
```

(2) 引数

ppid

PPID が格納されている領域のポインタです。

state

設定したい PPID の状態値です。

値=(-1)は使用不可を意味する値になります。

index

PPID 情報のインデクスです。登録時に EngAllLocPpInfo()関数によって与えられます。

インデクスは、PPID から EngGetPpIdIndex()関数で取得することができます。

(3) 戻り値

戻り値	意味
0	正常に設定できた。
(-1)	PPID が未登録であった。

(4) 説明

プロセスプログラム情報の状態値を設定します。

状態値の意味合いは値=(-1)以外についてはユーザが定義して使用します。

3.11.2.6 EngGetPpState() プロセスプログラム状態の取得 EngGetPpStateX() インデクスでのプロセスプログラム状態の取得

(1) 呼出書式

[C, C++]

```
API int WINAPI EngGetPpState(
    char    *ppid,           // PPID が格納されている領域のポインタ
    int     *state          // 取得した状態値格納用領域ポインタ
);
```

```
API int WINAPI EngGetPpStateX(
    int     index,          // PPID 情報のインデクス
    int     *state          // 取得した状態値格納用領域ポインタ
);
```

[.NET VB]

```
Function EngGetPpState (
    ByVal ppid As String,
    ByRef state As Int32) As Int32
```

```
Function EngGetPpStateX (
    ByVal index As Int32,
    ByRef state As Int32) As Int32
```

[.NET C#]

```
int EngGetPpState(
    byte[] ppid,
    ref int state);
```

```
int EngGetPpStateX(
    int index,
    ref int state );
```

(2) 引数

ppid

PPID が格納されている領域のポインタです。

state

取得した PPID の状態値を格納する領域のポインタです。

index

PPID 情報のインデクスです。登録時に EngAllocPpInfo()関数によって与えられます。
インデクスは、PPID から EngGetPpIdIndex()関数で取得することができます。

(3) 戻り値

戻り値	意味
0	正常に取得できた。
(-1)	PPID が未登録であった。

(4) 説明

プロセスプログラム情報の状態値を取得します。
状態値の意味合いは値=(-1)以外についてはユーザが定義して使用します。

3.11.2.7 EngGetPpList() 全登録 PPID 取得関数

(1) 呼出書式

[C, C++]

```
API int APIX EngGetPpList(
    TTEXT_DLIST **list           // 取得リスト格納ポインタの格納ポインタ
);
```

[.NET VB]

```
Function EngGetPpList (
    ByRef list As IntPtr) As Int32
```

[.NET C#]

```
int EngGetPpList(
    IntPtr list );
```

(2) 引数

list

取得できた PPID が格納されている TTEXT_DLIST 構造体のポインタを格納する領域のポインタです。

(3) 戻り値

戻り値	意味
0	正常に取得できた。
(-1)	取得できなかった。

(4) 説明

システムに登録されている PPID とその名前を TTEXT_DLIST 構造体に出すための関数です。

出す名前は、装置管理情報定義ファイルで PPID 定義時に与えられた名前です。

取得した情報の処理が終了した後、DshFreeTTEXT_DLIST()関数で list 内部の情報格納用に使用されているメモリを開放してください。

TTEXT_DLIST 構造体は次のとおりです。

```
typedef struct{
    int         count;           // 取得できた ID 数
    char        **id_list;       // 取得できた ID 格納用配列
    char        **name_list;     // 取得できた名前格納ポインタ配列
}TTEXT_DLIST;
```

3.11.2.8 EngGetPpid() インデクスから PPID (プロセスプログラム ID) の取得

(1) 呼出書式

[C, C++]

```
API int APIX EngGetPpid(
    int index,           // PP 情報のインデクス
    char *ppid          // 取得した PPID を格納する領域のポインタ
);
```

[.NET VB]

```
Function EngGetPpid (
    ByVal index As Int32,
    ByVal ppid As String) As Int32
```

[.NET C#]

```
int EngGetPpid(
    int index,
    byte[] ppid );
```

(2) 引数

index

PP 情報のインデクスです。登録時に EngAllPpidInfo() 関数によって与えられます。インデクスは、PPID から EngGetPpidIndex() 関数で取得することができます。

ppid

PPID を格納するための領域のポインタです。

(3) 戻り値

戻り値	意味
0	正常に取得できた。
(-1)	取得できなかった。(未登録)

(4) 説明

PP 情報のインデクスから PPID (プロセスプログラム ID) を取得し、ppid に格納します。正常に取得できた場合は関数戻り値として 0 が返却されます。

3.11.2.9 EngGetPpidIndex() PPID (プロセスプログラム ID) からインデクスの取得

(1) 呼出書式

[C, C++]

```
API int APIX EngGetPpidIndex(
    char *ppid,           // PPID が格納されている領域のポインタ
    int *index           // 取得したインデクスを格納するための領域のポインタ
);
```

[.NET VB]

```
Function EngGetPpidIndex (
    ByVal ppid As String,
    ByRef index As Int32) As Int32
```

[.NET C#]

```
int EngGetPpidIndex(
    byte[] ppid,
    ref int index);
```

(2) 引数

ppid

インデクスを取得したい対象の PPID が格納されている領域のポインタです。

index

取得したインデクスの値を格納するための領域のポインタです。

(3) 戻り値

戻り値	意味
0	正常に取得できた。
(-1)	取得できなかった。(未登録)

(4) 説明

ppid に指定される PPID (プロセスプログラム ID) から PP 情報インデクスを取得するための関数です。取得されたインデクスは index で指定された領域に格納されます。正常に取得できた場合は関数戻り値として 0 が返却されます。

3.11.3 PP プロセスプログラム関連通信メッセージ送信要求関数

3.11.3.1 EngSendS7F1() プロセスプログラムロード問合せメッセージ送信関数

(1) 呼出書式

[C, C++]

```
API int APIX EngSendS7F1(
    char *ppid, // PPID 格納領域のポインタ
    int length, // ロード問合せするプロセスプログラムのバイト長
    int *ppgnt, // S7F2 メッセージの PPGNT 格納領域ポインタ
    int (WINAPI *callback)(), // 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 EngSendS7F1 (
    ByVal ppid As String,
    ByVal length As Int32,
    ByRef ackc7 As Int32,
    ByVal callback As vcallback.callback_S7F1,
    ByVal upara As Int32) As Int32
```

[.NET C#]

```
int EngSendS7F1(
    byte[] ppid,
    uint length,
    ref int ackc7,
    CallbackS7F1 callback,
    uint upara );
```

(2) 引数

ppid

ロードしたいプロセスプログラム ID が格納されているポインタです。

length

ロードしたいプロセスプログラムのバイト長です。

ppgnt

問合せ結果 PPGNT を格納する領域のポインタです。

callback

DSHEng4 によるメッセージ送信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指定できます。

コールバックの指定が=0 の場合はブロックモードになります。

upara

コールバック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。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します。

(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。ppgnt に PPGNT 結果が返却される。

	(2) 非ブロック : 要求が受け付けられた。
(-1)	送信できなかった。
(-14)	T3タイムアウトを検出した。

(4) 説明

装置にプロセスプログラム情報のロードを問い合わせます。

ppid に指定された PPID と length で与えられるプロセスプログラム長を S7F1 メッセージに設定し、装置に送信します。

通信が正常に行われた場合、受信した S7F2 応答メッセージ内の PPGNT を ppngt に格納し、返却します。送信要求から S7F2 応答メッセージ受信までの制御は引数の S7F1 コールバック関数指定の有無によって次のようになります。

callback の指定	制御の流れ
なし (=0)	S7F1 送信後、応答メッセージ S7F2 が受信されるか、またはエラーを検出するまで制御が要求元に戻ってきません。 正常終了であれば、ppngt に S7F2 メッセージ内の PPGNT 値が渡されます。
あり	送信要求後、S7F1 の送信前に制御が戻されます。実行結果はコールバック関数で与えられます。 コールバック関数の end_status が=0 ならば正常であり、引数 ppngt に S7F2 メッセージ内の PPGNT 値が渡されます。 エラーが検出された場合、(-1) が end_status にセットされます。

(5) コールバック関数

[C, C++]

```
API int APIX callback(
    int end_status,           // 実行結果
    int *ppngt,              // PPGNT が格納されているメモリのポインタ
    ULONG upara              // 呼出時に指定したパラメータ
);
```

[.NET VB]

```
Function callback_S7F1(ByVal end_status As Integer, ByRef ackc7 As Integer, ByVal upara As Integer)
As Integer
```

[.NET C#]

```
int CallbackS7F1(int status, int* ack, uint upara);
```

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。ppngt に応答メッセージの PPGNT 値が返却される。
(-1)	送信エラーを検出した。
(-14)	T3タイムアウトを検出した。

3.11.3.2 EngSendS7F3() プロセスプログラム情報メッセージ送信関数

(1) 呼出書式

[C, C++]

```
API int WINAPI EngSendS7F3(
    char *ppid,                // PPID 格納領域のポインタ
    void *ppbody,              // ppbody データ格納ポインタ
    int format,                 // ppbody データのフォーマット( ICODE_??)
    int asize,                  // ppbody データの配列サイズ
    int ackc7,                  // S7F4 応答 ACKC7
    int (WINAPI *s7f3Callback)(), // 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 EngSendS7F3 (
    ByVal ppid As String,
    ByRef ackc7 As Int32,
    ByVal callback As vcallback.callback_S7F3,
    ByVal upara As Int32) As Int32
```

[.NET C#]

```
int EngSendS7F3(
    byte[] ppid,
    ref int ackc7,
    CallbackS7F3 callback,
    uint upara );
```

(2) 引数

ppid

送信したいプロセスプログラム ID が格納されているポインタです。

ppbody

送信したいプロセスプログラム PPBODY データが格納されているポインタです。

format

ppbody データのフォーマットです。DSHDR2 ドライバーで使用するデータアイテムのフォーマットです。

asize

ppbody データの配列サイズです。(ICODE_A)の場合は文字列バイト長になります。)

ackc7

受信した応答メッセージ S7F4 の ACKC7 返却用メモリのポインタです。

s7f3Callback

DSHeng4 によるメッセージ送信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指定できます。

コールバックの指定が=0 の場合はブロックモードになります。

upara

コールバック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。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します。

(3) 戻り値

戻り値	意味
0	(1) プロセスト : 正常に送信できた。 ackc7 に ACKC7 が返却されます。 (2) 非プロセスト : 要求が受け付けられた。
(-1)	送信できなかった。
(-14)	T3タイムアウトを検出した。

(4) 説明

プロセスプログラム情報の装置への送信要求を行います。S7F3 メッセージで送信します。
送信対象プロセスプログラム情報の指定はプロセスプログラム ID ならびに PPBODY データで行います。
PPBODY についてはフォーマットと配列サイズを指定する必要があります。

送信要求から S7F4 応答メッセージ受信までの制御は引数の S7F3 コールバック関数指定の有無によって次のようになります。

s7f3Callback の指定	制御の流れ
なし (=0)	S7F3 送信後、応答メッセージ S7F4 が受信されるか、またはエラーを検出するまで制御が要求元に戻ってきません。 正常終了であれば、ackc7 に ACKC7 が渡されます。
あり	送信要求後、S7F3 の送信前に制御が戻されます。実行結果はコールバック関数で与えられます。 コールバック関数の end_status が=0 ならば正常で引数 ackc7 に ACKC7 が渡されます。 エラーが検出された場合、(-1)が end_status にセットされます。

(5) コールバック関数

[C, C++]

```
API int APLX s7f3Callback(
    int end_status,           // 実行結果
    int *ackc7,              // 応答 ACKC7 が格納されているポインタ
    ULONG upara              // 呼出時に指定したパラメータ
);
```

[.NET VB]

```
Function callback_S7F3(ByVal end_status As Integer, ByRef ackc7 As Integer, ByVal upara As Integer)
As Integer
```

[.NET C#]

```
int CallbackS7F3(int end_status, int* ack, uint upara);
```

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。ackc7 に S7F4 の ACKC7 が設定されます。
(-1)	送信エラーを検出した。
(-14)	T3タイムアウトを検出した。

3.11.3.3 EngSendS7F5() - プロセスプログラム情報問合せメッセージ送信関数

(1) 呼出書式

[C, C++]

```
API int WINAPI EngSendS7F5(
    char *ppid,                // PPID 格納領域のポインタ
    TPP_INFO *ppinfo,         // S7F6 応答メッセージのデコード結果格納ポインタ
    int (WINAPI *s7f5Callback)(), // 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 EngSendS7F5 (
    ByVal ppid As String,
    ByRef ppinfo As dsh_info.TPP_INFO,
    ByVal callback As vcallback.callback_S7F5,
    ByVal upara As Int32) As Int32
```

[.NET C#]

```
int EngSendS7F5(
    byte[] ppid,
    ref TPP_INFO ppinfo,
    CallbackS7F5 callback,
    uint upara );
```

(2) 引数

ppid

装置に問合せたいプロセスプログラム ID が格納されているポインタです。

ppinfo

受信した応答メッセージ S7F6 をデコードした PP 情報を格納する TPP_INFO 構造体の指定します。

s7f5Callback

DSHEng4 によるメッセージ送信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指定できます。

コールバックの指定が=0 の場合はブロックモードになります。

upara

コールバック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。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します。

(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。 ppinfo に S7F6 で得られた PP 情報が格納され、返却されます。 (2) 非ブロックモード : 要求が受け付けられた。
(-1)	送信できなかった。
(-14)	T3 タイムアウトを検出した。

(4) 説明

S7F5 メッセージを使って装置にプロセスプログラム情報の問合せ要求を行います。問合せは ppid で指定さ

れるプロセスプログラム ID で行います。要求を受けた DSHeng4 は、S7F5 メッセージにエンコードし、装置に送信します。

送信要求から S7F6 応答メッセージ受信までの制御は引数の S7F5 コールバック関数指定の有無によって次のようになります。

s7f5Callback の指定	制御の流れ
なし (=0)	S7F5 送信後、応答メッセージ S7F6 が受信されるか、またはエラーを検出するまで制御が要求元に戻ってきません。 正常終了であれば、rmsg に S7F6 メッセージ 情報格納領域のポインタが渡されます。
あり	送信要求後、S7F5 の送信前に制御が戻されます。実行結果はコールバック関数で与えられます。 コールバック関数の end_status が=0 ならば正常で引数 rmsg に S7F6 メッセージ 情報構造体のポインタが与えられます。 エラーが検出された場合、(-1)が end_status にセットされます。

DSHeng4 は S7F6 で得られた情報を装置管理情報としての登録は行いません。必要に応じてユーザ側プログラムで登録処理をしてください。

正常に応答メッセージを受信した場合、PP 情報構造体のポインタが ppinfo に渡されますが、ユーザ側での処理を終えた後、その構造体で使用されているメモリを解放してください。

解放は次のように DshFreeTPP_INFO()関数を使って行ってください。

```
DshFreeTPP_INFO(ppinfo);
```

(5) コールバック関数

[C, C++]

```
API int APIX s7f5Callback(
    int end_status,           // 実行結果
    TPP_INFO *ppinfo,       // 問合せ結果の PP 情報が格納されている構造体のポインタ
    ULONG upara             // 呼出時に指定したパラメータ
);
```

[.NET VB]

```
Function callback_S7F5(ByVal end_status As Integer, ByRef info As dsh_info.TPP_INFO, ByVal upara As Integer) As Integer
```

[.NET C#]

```
int CallbackS7F5(int end_status, ref TPP_INFO info, uint upara);
```

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。ppinfo の構造体に PP 情報が格納されます。
(-1)	送信エラーを検出した。
(-14)	T3 タイムアウトを検出した。

3.11.4 PP プロセスプログラム関連ライブラリ関数

3.11.4.1 DshDecodeS7F3() - S7F3 をプロセスプログラム情報にデコード

(1) 呼出書式

[C, C++]

```
API int APIX DshDecodeS7F3(
    DSHMSG *smsg,           // SECS メッセージ 情報構造体のポインタ
    TPP_INFO *pinfo        // デコードした情報を格納する構造体のポインタ
);
```

[.NET VB]

```
Function DshDecodeS7F3 (
    ByRef smsg As dshdr2.DSHMSG,
    ByRef info As dsh_info.TPP_INFO) As Int32
```

[.NET C#]

```
int DshDecodeS7F3(
    ref DSHMSG smsg,
    ref TPP_INFO info );
```

(2) 引数

smsg

S7F3 の SECS メッセージ 情報が格納されている構造体のポインタです。

pinfo

デコードしたプロセスプログラム情報を格納する構造体のポインタです。

(3) 戻り値

戻り値	意味
0	正常にデコードできた。
(-1)	smsg を正しくデコードできなかった。

(4) 説明

S7F3 メッセージに含まれるプロセスプログラム情報を、ユーザプログラムが処理しやすい TPP_INFO 構造体の中にデコードします。

なお、構造体使用後は、構造体内部で使用されたメモリを DshFreeTPP_INFO() 関数を使って開放してください。(DshResponseS7F4() を使う場合、DshResonseS7F4() が開放してくれます。)

smsg S7F3

L,2
ppid
ppbody



3.11.4.2 DshEncodeS7F3() - プロセスプログラム情報を S7F3 へエンコード

(1) 呼出書式

[C, C++]

```
API int APIX DshEncodeS7F3(
    DSHMSG *smsg,           // SECS メッセージ 情報構造体のポインタ
    BYTE *buffer,          // S7F3 を格納するバッファポインタ
    int buflen,            // buffer のバイトサイズ
    TPP_INFO *pinfo        // エンコードしたいPP情報が格納されている構造体のポインタ
);
```

[.NET VB]

```
Function DshEncodeS7F3 (
    ByRef smsg As dshdr2.DSHMSG,
    ByRef buff As Byte,
    ByVal buflen As Int32,
    ByRef info As dsh_info.TPP_INFO) As Int32
```

[.NET C#]

```
int DshEncodeS7F3(
    ref DSHMSG smsg,
    byte[] buff,
    int buflen,
    ref TPP_INFO info );
```

(2) 引数

smsg

エンコードした S7F3 メッセージを格納するメッセージ情報構造体のポインタです。

buffer

エンコードした S7F3 のテキストを格納するためのバッファポインタです。

buflen

buffer のバイトサイズです。

pinfo

エンコードしたいプロセスプログラム情報が格納されている構造体のポインタです。

(3) 戻り値

戻り値	意味
0	正常にエンコードできた。
(-1)	smsg を正しくエンコードできなかった。

(4) 説明

TPP_INFO 構造体に格納されているプロセスプログラム情報を S7F3 の SECS メッセージにエンコードします。
buffer にはエンコードしたテキストを格納するために必要なメモリを準備しておく必要があります。

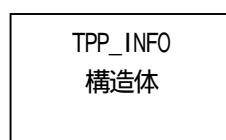
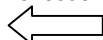
smsg S7F3

L,2

ppid

ppbody

encode



3.11.4.3 DshFreeTPP_INFO() - プロセスプログラム情報構造体メモリの開放

(1) 呼出書式

[C, C++]

```
API void APIX DshFreeTPP_INFO(  
    TPP_INFO *pinfo // メモリを開放したいPP情報が格納されている構造体のポインタ  
);
```

[.NET VB]

```
Sub DshFreeTPP_INFO (  
    ByRef info As dsh_info.TPP_INFO)
```

[.NET C#]

```
void DshFreeTPP_INFO_ALL(  
    ref TPP_INFO info );
```

(2) 引数

pinfo

メモリを解放したいプロセスプログラム情報構造体のポインタです。

(3) 戻り値

なし。

(4) 説明

TPP_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。
開放した後、TPP_INFO の内容を全て 0 で初期設定します。
pinfo が NULL ならば、何も処理しません。

3.11.4.4 DshCopyTPP_INFO() - プロセスプログラム情報構造体メモリのコピー

(1) 呼出書式

[C, C++]

```
API int APIX DshCopyTPP_INFO(
    TPP_INFO *dinfo,           // 北°-先のポインタ
    TPP_INFO *sinfo           // 北°-元のポインタ
);
```

[.NET VB]

```
Function DshCopyTPP_INFO (
    ByRef pinfo As dsh_info.TPP_INFO,
    ByRef sinfo As dsh_info.TPP_INFO) As Int32
```

[.NET C#]

```
int DshCopyTPP_INFO(
    ref TPP_INFO pinfo,
    ref TPP_INFO sinfo );
```

(2) 引数

pinfo

コピー先構造体メモリのポインタです。

sinfo

コピー元のプロセスプログラム情報が格納されている構造体メモリのポインタです。

(3) 戻り値

戻り値	意味
0	正常に北°-できた。
(-1)	sinfo または dinfo の値が NULL だったので北°-できなかった。

(4) 説明

sinfo が指す TPP_INFO 構造体内に格納されているプロセスプログラム情報を dinfo で指定された TPP_INFO 構造体メモリにコピーします。

dinfo 内のメンバーで新しいメモリが必要なものは本関数が取得します。

dinfo 内メンバーで確保されたメモリは、使用後、DshFreeTPP_INFO()関数を使って開放してください。

3.11.4.5 DshInitPpInfo プロセスプログラム TPP_INFO の初期設定

(1) 呼出書式

[C, C++]

```
API int APIX DshInitPpInfo(
    TPP_INFO    *info,           // TPP_INFO 構造体のポインタ
    int         ppid_fmt,       // ppid のフォーマット
    int         ppid_size,     // ppid の配列サイズ
    char        *ppid,         // プロセスプログラム ID
    int         ppbody_fmt,    // ppbody のフォーマット
    int         ppbody_size,   // ppbody の配列サイズ
    char        *ppbody       // ppbody 値
);
```

[.NET VB]

```
Sub DshInitPpInfo (
    ByRef info As dsh_info.TPP_INFO,
    ByVal ppid_fmt As Int32,
    ByVal ppid_size As Int32,
    ByVal ppid As String,
    ByVal ppbody_fmt As Int32,
    ByVal ppbody_size As Int32,
    ByVal ppbody As String)
```

[.NET C#]

```
void DshInitPpInfo(
    ref TPP_INFO info,
    int ppid_fmt,
    int ppid_size,
    byte[] ppid,
    int ppbody_fmt,
    int ppbody_size,
    byte[] ppbody );
```

(2) 引数

info

プロセスプログラム TPP_INFO 構造体のポインタです。このメンバーを初期設定します。

ppid_fmt

設定するプロセスプログラム ID のフォーマットです。(ICODE_A を指定してください。)

ppid_size

設定するプロセスプログラム ID の配列サイズです。

ppid

設定するプロセスプログラム ID です。

ppbody_fmt

設定するプロセスプログラムの PPBODY フォーマットです。(ICODE_A を指定してください。)

ppbody_size

設定するプロセスプログラムの PPBODY 配列サイズです。

ppbody_fmt

設定するプロセスプログラム PPBODY 値です。

(3) 戻り値
なし。

(4) 説明

本関数は APP が OFFLINE でプロセッサプログラム情報を生成する際に使用することができます。
最初に info 内をクリアします。そして、引数で指定された情報を info 内に設定します。
メモリが必要なメンバーについてはメモリを確保し情報をコピーします。

TPP_INFO 構造体の使用後は DshFreeTPP_INFO() 関数を使って構造体内部で使用したメモリを開放してください。

3.11.4.6 DshMakeS7F3Response() - S7F3 の応答メッセージの生成

(1) 呼出書式

[C, C++]

```
API int APIX DshMakeS7F3Response(
    BYTE ackc7,           // S7F4 に設定する ACKC7 です。
    DSHMSG *msg,         // S7F4 メッセージを格納するメッセージ構造体のポインタ
    BYTE *buff,          // S7F4 のテキスト格納バッファポインタ
    int buff_size        // buff のバイトサイズ
);
```

[.NET VB]

```
Function DshMakeS7F3Response (
    ByVal ackc7 As Byte,
    ByRef msg As dshdr2.DSHMSG,
    ByRef buff As Byte,
    ByVal buff_size As Int32) As Int32
```

[.NET C#]

```
int DshMakeS7F3Response(
    byte ackc7,
    ref DSHMSG msg,
    byte[] buff,
    int buff_size );
```

(2) 引数

ackc7

S7F4 メッセージの ACKC7 です。

msg

S7F4 応答メッセージ情報を格納するためのメッセージ構造体のポインタです。

buff

S7F4 応答メッセージのテキストを格納するためのバッファポインタです。

buff_size

buff のバイトサイズです。

(3) 戻り値

戻り値	意味
0	正常に生成できた。
(-1)	生成できなかった。(buff 領域不足)

(4) 説明

S7F4 応答メッセージを msg 内に生成し、ackc7 を S7F4 の ACKC7 として設定します。
ackc7 の設定はユーザが PP 情報を評価した結果です。

3.11.4.7 DshDecodeS7F17() - S7F17 デコード - 削除プロセスプログラム一覧

(1) 呼出書式

[C, C++]

```
API int APIX DshDecodeS7F17(
    DSHMSG *msg, // SECS メッセージ 情報構造体のポインタ
    TPPID_LIST *list // デコードした PPID 一覧情報を格納する構造体のポインタ
);
```

[.NET VB]

```
Function DshDecodeS7F17 ( _
    ByRef msg As dshdr2.DSHMSG, _
    ByRef list As dsh_info.TPPID_LIST) As Int32
```

[.NET C#]

```
int DshDecodeS7F17(
    ref DSHMSG msg,
    ref TPPID_LIST list );
```

(2) 引数

msg

S7F17 の SECS メッセージ 情報が格納されている構造体のポインタです。

list

デコードした PPID 一覧情報を格納する構造体のポインタです。

(3) 戻り値

戻り値	意味
0	正常にデコードできた。
(-1)	msg を正しくデコードできなかった。

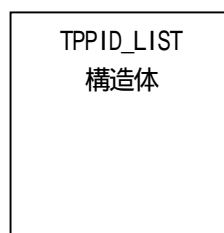
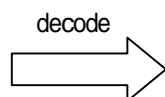
(4) 説明

S7F17 メッセージに含まれるプロセスプログラム ID 情報を、ユーザプログラムが処理しやすい TPPID_LIST 構造体の中にデコードします。

なお、構造体使用後は、構造体内部で使用されたメモリを DshFreeTPPID_LIST() 関数を使って開放してください。

msg S7F17

```
L,n
ppid1
ppid2
.
.
```



3.11.4.8 DshFreeTPPID_LIST() - プロセスプログラム ID 一覧情報構造体メモリの開放

(1) 呼出書式

[C, C++]

```
API void APIX DshFreeTPPID_LIST(  
    TPPID_LIST *list           // メモリを開放したいPPID 情報が格納されている構造体のポインタ  
);
```

[.NET VB]

```
Sub DshFreeTPPID_LIST (  
    ByRef list As dsh_info.TPPID_LIST)
```

[.NET C#]

```
void DshFreeTPPID_LIST(  
    ref TPPID_LIST list );
```

(2) 引数

list

メモリを解放したいプロセスプログラム ID 一覧情報構造体のポインタです。

(3) 戻り値

なし。

(4) 説明

TPPID_LIST 構造体内で PPID 情報格納用に使われているメモリを全て解放します。

開放した後、TPPID_LIST の内容を全て 0 で初期設定します。

list が NULL ならば、何も処理しません。

3.11.4.9 DshDecodeS7F27() - S7F27 をプロセスプログラム妥当性情報にデコード

(1) 呼出書式

[C, C++]

```
API int APIX DshDecodeS7F27(
    DSHMSG *msg, // SECS メッセージ 情報構造体のポインタ
    TPP_PVS_LIST *list // デコードした PP 妥当性チェック情報を格納する構造体のポインタ
);
```

[.NET VB]

```
Function DshDecodeS7F27 (
    ByRef msg As dshdr2.DSHMSG,
    ByRef list As dsh_info.TPP_PVS_LIST) As Int32
```

[.NET C#]

```
int DshDecodeS7F27(
    ref DSHMSG msg,
    ref TPP_PVS_LIST list );
```

(2) 引数

msg

S7F27 の SECS メッセージ 情報が格納されている構造体のポインタです。

list

デコードした PP 妥当性チェック情報を格納する構造体のポインタです。

(3) 戻り値

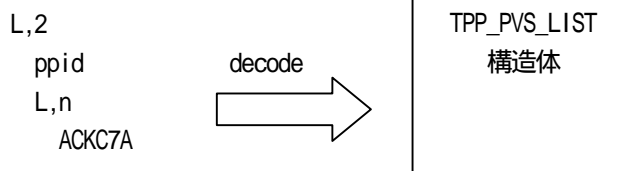
戻り値	意味
0	正常にデコードできた。
(-1)	msg を正しくデコードできなかった。

(4) 説明

S7F27 メッセージに含まれるプロセスプログラム ID 情報を、ユーザプログラムが処理しやすい TPP_PVS_LIST 構造体の中にデコードします。

なお、構造体使用後は、構造体内部で使用されたメモリを DshFreeTPP_PVS_LIST()関数を使って開放してください。

msg S7F27



3.10.4.11 DshFreeTPP_PVS_LIST() - プロセスプログラム妥当性情報構造体メモリの開放

(1) 呼出書式

[C, C++]

```
API void APIX DshFreeTPP_PVS_LIST(  
    TPP_PVS_LIST *list           // メモリを開放したい妥当性情報が格納されている構造体のポインタ  
);
```

[.NET VB]

```
Sub DshFreeTPP_PVS_LIST (  
    ByRef list As dsh_info.TPP_PVS_LIST)
```

[.NET C#]

```
void DshFreeTPP_PVS_LIST(  
    ref TPP_PVS_LIST list );
```

(2) 引数

list

メモリを解放したいプロセスプログラム妥当性情報構造体のポインタです。

(3) 戻り値

なし。

(4) 説明

TPP_PVS_LIST 構造体内で妥当性情報格納用に使われているメモリを全て解放します。

開放した後、TPP_PVS_LIST の内容を全て 0 で初期設定します。

list が NULL ならば、何も処理しません。

3.11.4 ユーザ作成ライブラリ関数

3.11.4.1 DshResponseS7F2() S7F2 PP ロード問合せ応答メッセージ

(1) 呼出書式

[C, C++]

```
API int APIX DshResponseS7F2(
    ID_TR trid, // DSHDR2 のトランザクション ID
    TPPINQ_INFO *info, // PP ロード問合せメッセージ 情報格納領域のポインタ
    int ppgnt // S7F2 応答 ACK
);
```

[.NET VB]

```
Function DshResponseS7F2 (
    ByVal trid As Int32,
    ByRef info As dsh_info.TPPINQ_INFO,
    ByVal ppgnt As Int32) As Int32
```

[.NET C#]

```
int DshResponseS7F2(
    uint trid,
    ref TPPINQ_INFO info,
    int ppgnt );
```

(2) 引数

trid

S7F1 信時に DSHEng4 から渡される DSHDR2 通信ドライバーのトランザクション管理のための ID です。

info

PP ロード問合せ情報が格納されている構造体のポインタです。

ppgnt

送信する応答メッセージ S7F2 の許可情報

(3) 戻り値

戻り値	意味
0	正常に送信できた。
(-1)	送信できなかった。

(4) 説明

PP ロード問合せメッセージ S7F1 に対する応答メッセージを送信します。

本関数はユーザ作成ライブラリ DLL (DSHEngU.dll) に含まれる関数ですが、ここでは DSHEng4 パッケージに標準的な関数として付属されているものです。(ユーザ独自による作成も可能です)

引数に指定されている ppgnt 情報から S7F2 メッセージを組み立て、その後、S7F2 メッセージを送信します。

なお、S7F2 メッセージの組み立てに、DshMakeS7F2Response() 関数を使用できます。

3.11.4.2 DshResponseS7F4() S7F4 PP 送信応答メッセージ

(1) 呼出書式

[C, C++]

```
API int APIX DshResponseS7F4(
    ID_TR trid,                // DSHDR2 のトランザクション ID
    TPP_INFO *info,           // PP 送信メッセージ 情報格納領域のポインタ
    int ackc7                 // S7F4 応答 ACK
);
```

[.NET VB]

```
Function DshResponseS7F4 (
    ByVal trid As Int32,
    ByRef info As dsh_info.TPP_INFO,
    ByVal ackc7 As Int32) As Int32
```

[.NET C#]

```
int DshResponseS7F4(
    uint trid,
    ref TPP_INFO info,
    int ackc7 );
```

(2) 引数

trid

S7F3 信時に DSHEng4 から渡される DSHDR2 通信ドライバーのトランザクション管理のための ID です。

info

PP 送信情報が格納されている構造体のポインタです。

ackc7

送信する応答メッセージ S7F4 の許可情報

(3) 戻り値

戻り値	意味
0	正常に送信できた。
(-1)	送信できなかった。

(4) 説明

PP 送信メッセージ S7F3 に対する応答メッセージを送信します。

本関数はユーザ作成ライブラリ DLL (DSHEngU.dll) に含まれる関数ですが、ここでは DSHEng4 パッケージに標準的な関数として付属されているものです。(ユーザ独自による作成も可能です)

引数に指定されている ackc7 情報から S7F4 メッセージを組み立て、その後、S7F4 メッセージを送信します。

なお、S7F4 メッセージの組み立てに、DshMakeS7F4Response() 関数を使用できます。

3.11.4.3 DshResponseS7F6() S7F6 PP 要求応答メッセージ

(1) 呼出書式

[C, C++]

```
API int APIX DshResponseS7F6(
    ID_TR trid,                // DSHDR2 のトランザクション ID
    TPP_INFO *info,           // PP 要求メッセージ 情報格納領域のポインタ
    int ackc7                 // S7F6 応答 ACK
);
```

[.NET VB]

```
Function DshResponseS7F6 (
    ByVal trid As Int32,
    ByRef info As dsh_info.TPP_INFO,
    ByVal ackc7 As Int32) As Int32
```

[.NET C#]

```
int DshResponseS7F6(
    uint trid,
    ref TPP_INFO info,
    int ackc7 );
```

(2) 引数

trid

S7F5 信時に DSHEng4 から渡される DSHDR2 通信ドライバーのトランザクション管理のための ID です。

info

PP 要求情報が格納されている構造体のポインタです。この内容が応答されます。

ackc7

送信する応答メッセージ S7F6 の許可情報

(3) 戻り値

戻り値	意味
0	正常に送信できた。
(-1)	送信できなかった。

(4) 説明

PP 要求メッセージ S7F5 に対する応答メッセージを送信します。

本関数はユーザ作成ライブラリ DLL (DSHEngU.dll) に含まれる関数ですが、ここでは DSHEng4 パッケージに標準的な関数として付属されているものです。(ユーザ独自による作成も可能です)

引数に指定されている info と ackc7 情報から S7F6 メッセージを組み立て、その後、S7F6 メッセージを送信します。

ackc7 が 0 以外の値 (エラー) であれば、List=0 のメッセージを応答します。

なお、S7F6 メッセージの組み立てに、DshEncodePp () 関数を使用できます。

送信後は、info の内部で使用されているメモリを DshFreeTPP_INFO() で開放します。

3.11.4.4 DshResponseS7F18() S7F18 PP 削除指示応答メッセージ

(1) 呼出書式

[C, C++]

```
API int APIX DshResponseS7F18(
    ID_TR trid,           // DSHDR2 のトランザクション ID
    int ackc7            // S7F18 応答 ACK
);
```

[.NET VB]

```
Function DshResponseS7F18 (
    ByVal trid As Int32,
    ByVal ackc7 As Long) As Int32
```

[.NET C#]

```
int DshResponseS7F18(
    ID_TR trid,
    int ackc7);
```

(2) 引数

trid

S7F17 信時に DSHEng4 から渡される DSHDR2 通信ドライバーのトランザクション管理のための ID です。

ackc7

送信する応答メッセージ S7F18 の許可情報

(3) 戻り値

戻り値	意味
0	正常に送信できた。
(-1)	送信できなかった。

(4) 説明

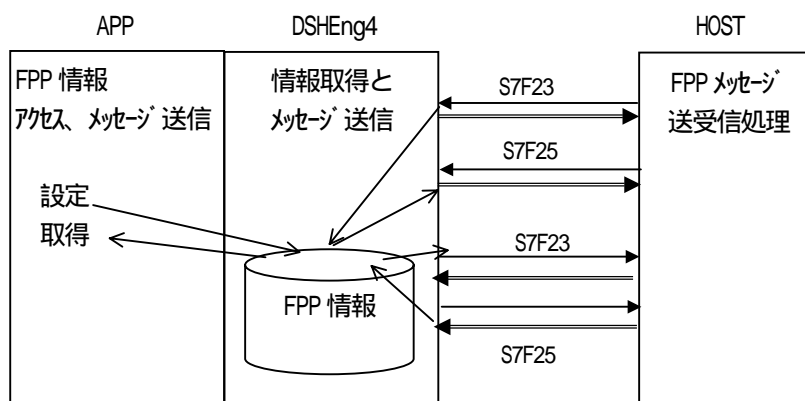
PP 削除指示メッセージ S7F17 に対する応答メッセージを送信します。

本関数はユーザ作成ライブラリ DLL (DSHEngU.dll) に含まれる関数ですが、ここでは DSHEng4 パッケージに標準的な関数として付属されているものです。(ユーザ独自による作成も可能です)

引数に指定されている ackc7 情報から S7F18 メッセージを組み立て、その後、S7F18 メッセージを送信します。

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

ここで述べる書式付プロセスプログラム情報は、DSHEng4 が管理します。従って、APP はこれらの情報をアクセスと関連メッセージを送信するために以下の DSHEng4 API 関数を使用します。



(1) 情報アクセスと送信 API 関数

プロセスプログラム情報のアクセスと装置へのメッセージ送信に関連するサービスのための API 関数名は一覧表のとおりです。

	API 関数名	機能
1	EngAllLocFppInfo()	PPID 情報領域を割当て登録します。
2	EngSetFppInfo()	書式付プロセスプログラム情報を設定・変更します。
3	EngGetFppInfo()	PPID 指定で書式付プロセスプログラム情報を取得します。
4	EngGetFppInfoX()	PPID 情報インデックス指定で書式付プロセスプログラム情報を取得します。
5	EngDelFppInfo()	PPID 指定で書式付プロセスプログラム情報を削除します。
6	EngDelFppInfoX()	PPID 情報インデックス指定で書式付プロセスプログラム情報を削除します。
7	EngSetFppState()	PPID 指定で PPID の状態を設定します。
8	EngSetFppStateX()	PPID 情報インデックス指定で PPID の状態を設定します。
9	EngGetFppState()	PPID 指定で PPID の状態を取得します。
10	EngGetFppStateX()	PPID 情報インデックス指定で PPID の状態を取得します。
11	EngGetFppId()	指定した PPID 情報インデックスの PPID を取得します。
12	EngGetFppIdIndex()	指定した PPID の情報インデックスを取得します。
13	EngGetFppList()	PPID の一覧リストを取得します。
14	EngSendS7F23()	S7F23 メッセージで PPID 情報を送信します。
15	EngSendS7F25()	S7F25 メッセージを使って PPID 情報を取得します。

PPID インデックスは、DSHEng4 が管理する各 PPID 領域の番号です。このインデックスの値は、EngAllLocFppInfo() 関数実行時に DSHEng4 によって割当てられ、APP に渡されます。また、PPID 情報の取得時に、情報格納構造体メンバーの index に設定されます。

(2) ライブラリ関数

他に APP が使用できる書式付プロセスプログラム情報処理用 API 関数として、以下の関数があります。

	API 関数名	機能
1	DshDecodeS7F23()	S7F23 のメッセージ内の書式付 PP 情報を TFPP_INFO 構造体にデコードするための関数です。
2	DshEncodeS7F23()	TFPP_INFO 構造体のプロセッサプログラム情報を S7F23 メッセージにエンコードするための関数です。
3	DshFreeTFPP_INFO()	書式付 PP 情報が格納されている TFPP_INFO 構造体とその内部で使用されているメモリを開放するための関数です。
4	DshCopyTFPP_INFO()	TFPP_INFO の書式付 PP 情報をコピーします。
5	DshInitFppInfo()	書式付 PP 情報、TFPP_INFO 構造体の初期設定を行います。
6	DshPutFppCCode()	書式付 PP 情報にコマンドコード情報を設定します。
7	DshPutFppPPara()	書式付 PP 情報のコマンドコード情報にパラメータ情報を設定します。
8	DshPutFppPParaInfo()	書式付プロセッサプログラムパラメータを配列位置指定で追加します。
9	DshMakeS7F23Response()	S7F24 応答メッセージを生成し、送信するための関数。u_s7f23.c で使用します。

(3) ユーザ作成ライブラリ関数

	ライブラリ関数名	機能
1	DshResponseS7F24()	S7F24 書式付きプロセスプログラム送信応答メッセージ
2	DshResponseS7F26()	S7F26 書式付きプロセスプログラム要求応答メッセージ

3.12.1 使用する情報格納構造体

書式付プロセスプログラム情報を操作する関数は、共通の情報格納のための TFPP_INFO 構造体を使用します。

TFPP_INFO と、その内部で使用する他の構造体は下記のとおりです。

(1) TFPP_INFO - Formatted Process Program Information

```
typedef struct{
    int    index;           // 情報格納場所インデックス
    int    state;          // PP の状態
    char   *name;          // pp 名(装置管理情報定義ファイルで与えられた名前)
    int    ppid_fmt;       // ppid のアイテムフォーマット( Format 10 ASCII が設定されます。)
    int    ppid_size;      // 同配列サイズ(実際の ppid のバイト長です。)
    int    max_ppid_size;  // 配列サイズの最大値
    char   *ppid;          // ppid 値格納ポインタ
    char   *mdl;           // mdl 格納ポインタ
    char   *softrev;       // softrev 格納ポインタ
    int    ccode_count;    // # of process commands
    TFPP_CCODE **ccode_list; // コマンドコード情報がインテリス
}TFPP_INFO;              // formatted process program
(注) ccode_count = 0 の場合、コマンドコードがないことを意味します。
```

(2) TFPP_CCODE - Process Command Information

```
typedef struct{
    int    ccode_fmt;      // ccode のアイテムフォーマット
    int    ccode_size;    // 同配列サイズ
    void   *ccode;        // command code 値格納ポインタ
    int    ppara_count;   // # of para count
    TFPP_PARA **ppara_list; // パラメータ情報がインテリス
} TFPP_CCODE;
(注) ppara_count = 0 の場合、パラメータがないことを意味します。
```

(3) Formatted Program Parameter

```
typedef struct{
    int    ppara_fmt;     // ppara のアイテムフォーマット
    int    ppara_size;   // 同配列サイズ
    void   *ppara;       // パラメータ値格納ポインタ
} TFPP_PARA;
```

3.12.2 FPP 書式付プロセスプログラム情報アクセス関数

3.12.2.1 EngAllocFppInfo() - 書式付プロセスプログラムの登録

(1) 呼出書式

[C, C++]

```
API int APIX EngAllocFppInfo(
    char* ppid,           // 書式付 PPID 格納領域のポインタ
    int *index           // 得られた情報領域のインデックス格納用ポインタ
);
```

[.NET VB]

```
Function EngAllocFppInfo (
    ByVal ppid As String,
    ByRef index As Int32) As Int32
```

[.NET C#]

```
int EngAllocFppInfo(
    byte *ppid,
    ref int index );
```

(2) 引数

ppid

登録したい書式付プロセスプログラム ID が格納されているポインタです。

index

登録された情報領域のインデックス値を格納する領域のポインタです。

(3) 戻り値

戻り値	意味
0	正常に登録できた。
1	指定された PPID は既に登録されていた。
(-1)	登録できなかった。

(4) 説明

書式付プロセスプログラムを新規にシステムに登録するための関数です。

引数 ppid で与えられる書式付プロセスプログラム ID をシステムに登録します。

正常に登録できた場合は、index で指定された領域に登録された情報領域のインデックスが設定返却されます。もし、ppid に指定された書式付プロセスプログラムが既に登録済みであった場合には関数の戻り値 = 1 を返却します。index には既に登録されている情報領域のインデックスが設定されます。

得られたインデックスを使って、情報の設定、取得、削除などのアクセスを行うことができます。

3.12.2.2 EngSetFppInfo() - 書式付プロセスプログラム情報の設定 EngSetFppInfoX() インデクス指定による書式付プロセスプログラム情報の設定

(1) 呼出書式

[C, C++]

```
API int APIX EngSetFppInfo(
    TFPP_INFO *pinfo           // 書式付 PP 情報格納構造体のポインタ
);

API int APIX EngSetFppInfoX(
    int index,                 // EngAllocFppInfo() で得られたインデクス値
    TFPP_INFO *pinfo           // 書式付 PP 情報格納構造体のポインタ
);
```

[.NET VB]

```
Function EngSetFppInfo (
    ByRef pinfo As dsh_info.TFPP_INFO) As Int32

Function EngSetFppInfoX (
    ByVal index As Int32,
    ByRef pinfo As dsh_info.TFPP_INFO) As Int32
```

[.NET C#]

```
int EngSetFppInfo(
    ref TFPP_INFO pinfo );

int EngSetFppInfoX(
    int index,
    ref TFPP_INFO pinfo );
```

(2) 引数

pinfo

設定したい書式付プロセスプログラム情報が格納されている格納構造体領域のポインタです。

index

書式付 PPID 情報のインデクスです。登録時に EngAllocFppInfo() 関数によって与えられます。インデクスは、PPID から EngGetFppIdIndex() 関数で取得することができます。

(3) 戻り値

戻り値	意味
0	正常に設定できた。
(-1)	設定できなかった。

(4) 説明

本関数は、pinfo に格納されている書式付プロセスプログラム情報の設定・変更に使います。

引数 pinfo 内の ppid メンバーに指定される書式付プロセスプログラム ID の情報として情報が設定されません。

pinfo 内には PPID の他、 コマンドコード、 パラメータなどの情報が含まれます。
指定した PPID が既に登録済みである場合、 pinfo 内の内容にすべて書き換えられます。

pinfo 内の PPID が未登録であった場合は、 登録手続きをしてから PP 情報を設定します。
(EngAllocFppInfo()関数で行われる登録と同じ登録が行われます。)

3.12.2.3 EngGetFppInfo() - 書式付プロセスプログラムの取得 EngGetFppInfoX() - インデクス指定での書式付プロセスプログラムの取得

(1) 呼出書式

[C, C++]

```
API int WINAPI EngGetFppInfo(
    char *ppid; // PPID が格納されている領域のポインタ
    TFPP_INFO *pinfo // 書式付 PP 情報を格納する構造体ポインタ
);
```

```
API int WINAPI EngGetFppInfoX(
    int index, // PPID 情報のインデクス
    TFPP_INFO *pinfo // 書式付 PP 情報を格納する構造体ポインタ
);
```

[.NET VB]

```
Function EngGetFppInfo (
    ByVal ppid As String,
    ByRef pinfo As dsh_info.TFPP_INFO) As Int32
```

```
Function EngGetFppInfoX (
    ByVal index As Int32,
    ByRef pinfo As dsh_info.TFPP_INFO) As Int32
```

[.NET C#]

```
int EngGetFppInfo(
    byte *ppid,
    ref TFPP_INFO pinfo );
```

```
int EngGetFppInfoX(
    int index,
    ref TFPP_INFO pinfo );
```

(2) 引数

ppid

書式付 PPID が格納されている領域のポインタです。

pinfo

取得した書式付 PP 情報を格納するための構造体領域のポインタです。

index

書式付 PPID 情報のインデクスです。登録時に EngAllocFppInfo()関数によって与えられます。インデクスは、PPID から EngGetFppIdIndex()関数で取得することができます。

(3) 戻り値

戻り値	意味
0	正常に取得できた。
(-1)	取得できなかった。(PPID が未登録であった。)

(4) 説明

ppid または index に指定されている PPID のプロセスプログラム情報を pinfo が指す構造体領域に取得します。

TFPP_INFO 構造体の中に情報を格納するために必要なメモリは、DSHEng4 が準備確保します。即ち、構造体のメンバーの中でポインタになっている情報の実体即ち、ppid, mdIn などのためのメモリは DSHEng4 が準備します。

これらのメモリは、使用后、ユーザが DSHEng4 の API 関数を使って次のように開放してください。

```
TFPP_INFO *pinfo;

if ( EngGetFppInfo( ppidinfo ) == 0 ){
    pinfo の処理
    処理終了後
    DshFreeTFPP_INFO( pinfo );          // pinfo 内に使用されているメモリの開放
}
```

3.12.2.4 EngDelFppInfo() - 書式付プロセスプログラムの削除 EngDelFppInfoX() - インデクスでの書式付プロセスプログラムの削除

(1) 呼出書式

[C, C++]

```
API int WINAPI EngDelFppInfo(
    char *ppid; // PPID が格納されている領域のポインタ
);

API int WINAPI EngDelFppInfoX(
    int index; // インデクス(0,1,2,...)
);
```

[.NET VB]

```
Function EngDelFppInfo (
    ByVal ppid As String) As Int32

Function EngDelFppInfoX (
    ByVal index As Int32) As Int32
```

[.NET C#]

```
int EngDelFppInfo(
    byte *ppid );

int EngDelFppInfoX(
    int index );
```

(2) 引数

ppid

書式付 PPID が格納されている領域のポインタです。

index

書式付 PPID 情報のインデクスです。登録時に EngAllocFppInfo()関数によって与えられます。インデクスは、PPID から EngGetFppIdIndex()関数で取得することができます。

(3) 戻り値

戻り値	意味
0	正常に削除できた。
(-1)	PPID が未登録であった。

(4) 説明

ppid または index に指定されている書式付 PPID のプロセスプログラム情報をシステムの登録から削除します。

3.12.2.5 EngSetFppState() 書式付プロセスプログラム状態の設定 EngSetFppStateX() インデクスでの書式付プロセスプログラム状態の設定

(1) 呼出書式

[C, C++]

```
API int WINAPI EngSetFppState(
    char *ppid, // PPID が格納されている領域のポインタ
    int state // 設定したい状態値
);
```

```
API int WINAPI EngSetFppStateX(
    int index, // PPID 情報のインデクス
    int state // 設定したい状態値
);
```

[.NET VB]

```
Function EngSetFppState (
    ByVal ppid As String,
    ByVal state As Int32) As Int32
```

```
Function EngSetFppStateX (
    ByVal index As Int32,
    ByVal state As Int32) As Int32
```

[.NET C#]

```
int EngSetFppState(
    byte *ppid,
    int state );
```

```
int EngSetFppStateX(
    int index,
    int state );
```

(2) 引数

ppid

書式付 PPID が格納されている領域のポインタです。

state

設定したい PPID の状態値です。

値=(-1)は使用不可を意味する値になります。

index

書式付 PPID 情報のインデクスです。登録時に EngAllocFppInfo()関数によって与えられます。インデクスは、PPID から EngGetFppIdIndex()関数で取得することができます。

(3) 戻り値

戻り値	意味
0	正常に設定できた。
(-1)	PPID が未登録であった。

(4) 説明

書式付プロセスプログラム情報の状態値を設定します。

状態値の意味合いは値=(-1)以外についてはユーザが定義して使用します。

3.12.2.6 EngGetFppState() 書式付プロセスプログラム状態の取得 EngGetFppStateX() インデクスでの書式付プロセスプログラム状態の取得

(1) 呼出書式

[C, C++]

```
API int WINAPI EngGetFppState(
    char    *ppid,           // PPID が格納されている領域のポインタ
    int     *state          // 取得した状態値格納用領域ポインタ
);
```

```
API int WINAPI EngGetFppStateX(
    int     index,          // PPID 情報のインデクス
    int     *state          // 取得した状態値格納用領域ポインタ
);
```

[.NET VB]

```
Function EngGetFppState (
    ByVal ppid As String,
    ByRef state As Int32) As Int32
```

```
Function EngGetFppStateX (
    ByVal index As Int32,
    ByRef state As Int32) As Int32
```

[.NET C#]

```
int EngGetFppState(
    byte *ppid,
    ref int state );
```

```
int EngGetFppStateX(
    int index,
    ref int state );
```

(2) 引数

ppid

書式付 PPID が格納されている領域のポインタです。

state

取得した PPID の状態値を格納する領域のポインタです。

index

書式付 PPID 情報のインデクスです。登録時に EngAllLocFppInfo()関数によって与えられます。インデクスは、PPID から EngGetFppIdIndex()関数で取得することができます。

(3) 戻り値

戻り値	意味
0	正常に取得できた。
(-1)	PPID が未登録であった。

(4) 説明

書式付プロセスプログラム情報の状態値を取得します。
状態値の意味合いは値=(-1)以外についてはユーザが定義して使用します。

3.12.2.7 EngGetFppList() 全登録 FPPID 取得関数

(1) 呼出書式

[C, C++]

```
API int APIX EngGetFppList(
    TTEXT_DLIST **list           // 取得リスト格納ポインタの格納ポインタ
);
```

[.NET VB]

```
Function EngGetFppList (
    ByRef list As IntPtr) As Int32
```

[.NET C#]

```
int EngGetFppList(
    IntPtr list );
```

(2) 引数

list

取得できた FPPID が格納されている TTEXT_DLIST 構造体のポインタを格納する領域のポインタです。

(3) 戻り値

戻り値	意味
0	正常に取得できた。
(-1)	取得できなかった。

(4) 説明

システムに登録されている FPPID(書式付 PPID)とその名前を TTEXT_DLIST 構造体に出すための関数です。出す名前は、装置管理情報定義ファイルで FPPID 定義時に与えられた名前です。

取得した情報の処理が終了した後、DshFreeTText_DLIST()関数で list 内部の情報格納用に使用されているメモリを開放してください。

TTEXT_DLIST 構造体は次のとおりです。

```
typedef struct{
    int         count;           // 取得できた ID 数
    char        **id_list;      // 取得できた ID 格納用配列
    char        **name_list;    // 取得できた名前格納ポインタ配列
}TTEXT_DLIST;
```

3.12.2.8 EngGetFppId() インデクスから FPPID (書式付プロセスプログラム ID) の取得

(1) 呼出書式

[C, C++]

```
API int APIX EngGetFppId(
    int index,           // FPP 情報のインデクス
    char *ppid          // 取得した PPID を格納する領域のポインタ
);
```

[.NET VB]

```
Function EngGetFppId (
    ByVal index As Int32,
    ByVal ppid As String) As Int32
```

[.NET C#]

```
int EngGetFppId(
    int index,
    byte *ppid );
```

(2) 引数

index

FPP 情報のインデクスです。登録時に EngAllocFppInfo()関数によって与えられます。インデクスは、PPID から EngGetFppIdIndex()関数で取得することができます。

ppid

FPPID を格納するための領域のポインタです。

(3) 戻り値

戻り値	意味
0	正常に取得できた。
(-1)	取得できなかった。(未登録)

(4) 説明

FPP 情報のインデクスから PPID (書式付プロセスプログラム ID) を取得し、ppid に格納します。正常に取得できた場合は関数戻り値として 0 が返却されます。

3.12.2.9 EngGetFppIdIndex() FPPID (書式付プロセスプログラム ID) からインデクスの取得

(1) 呼出書式

[C, C++]

```
API int APIX EgnGetFppIdIndex(
    char *ppid,           // PPID が格納されている領域のポインタ
    int *index           // 取得したインデクスを格納するための領域のポインタ
);
```

[.NET VB]

```
Function EngGetFppIdIndex (
    ByVal ppid As String,
    ByRef index As Int32) As Int32
```

[.NET C#]

```
int EngGetFppIdIndex(
    byte *ppid,
    ref int index );
```

(2) 引数

ppid

インデクスを取得したい対象の PPID が格納されている領域のポインタです。

index

取得したインデクスの値を格納するための領域のポインタです。

(3) 戻り値

戻り値	意味
0	正常に取得できた。
(-1)	取得できなかった。(未登録)

(4) 説明

ppid に指定される PPID (書式付プロセスプログラム ID) からインデクスを取得するための関数です。
 取得されたインデクスは index で指定された領域に格納されます。
 正常に取得できた場合は関数戻り値として 0 が返却されます。

3.12.3 FPP 書式付プロセスプログラム関連通信メッセージ送信要求関数

3.12.3.1 EngSendS7F23() 書式付プロセスプログラム情報送信関数

(1) 呼出書式

[C, C++]

```
API int APIX EngSendS7F23(
    TFPP_INFO *info,           // FPP 情報が格納されている構造体のポインタ
    int *ackc7,               // S7F24 応答 ACKC7 格納ポインタ
    int (WINAPI *s7f23Callback)(), // 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 EngSendS7F23 (
    ByVal ppid As String,
    ByRef ackc7 As Int32,
    ByVal callback As vcallback.callback_S7F23,
    ByVal upara As Int32) As Int32
```

[.NET C#]

```
int EngSendS7F23(
    byte[] ppid,
    ref int ackc7,
    CallbackS7F23 callback,
    uint upara );
```

(2) 引数

info

送信したい書式付プロセスプログラム情報が格納されている構造体のポインタです。

ackc7

受信した応答メッセージの ACKC7 を格納する領域のポインタです。

s7f23Callback

DSEng4 によるメッセージ送信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指定できます。

コールバックの指定が=0 の場合はブロックモードになります。

upara

コールバック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。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します。

(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。 ackc7 に S7F24 の ACKC7 が格納されます。 (2) 非ブロックモード : 要求が受け付けられた。
(-1)	送信できなかった。
(-14)	T3タイムアウトを検出した。

(4) 説明

装置に info 内に格納されている書式付プロセスプログラム情報の送信を行います。S7F23 メッセージで送信します。

本関数は info の内容を S7F23 メッセージにエンコードし、装置に送信します。

送信要求から S7F24 応答メッセージ受信までの制御は引数の S7F23 コールバック関数指定の有無によって次のようになります。

s7f23Callback の指定	制御の流れ
なし (=0)	S7F23 送信後、応答メッセージ S7F24 が受信されるか、またはエラーを検出するまで制御が要求元に戻ってきません。 正常終了であれば、ackc7 に S7F24 の ACKC7 が格納されます。
あり	送信要求後、S7F23 の送信前に制御が戻されます。実行結果はコールバック関数で与えられます。 コールバック関数の end_status が=0 ならば正常で、ackc7 に S7F24 の ACKC7 が格納されます。エラーが検出された場合、(-1)が end_status にセットされます。

正常に応答メッセージを受信した場合、ackc7 に S7F24 の ACKC7 の値が格納されます。

(5) コールバック関数

[C, C++]

```
API int APIX s7f23Callback(
    int end_status,           // 実行結果
    int *ackc7,              // S7F24 の ACKC7 の値が格納されているメモリポインタ
    ULONG upara              // 呼出時に指定したパラメータ
);
```

[.NET VB]

```
Function callback_S7F23(ByVal end_status As Integer, ByRef ackc7 As Integer, ByVal upara As Integer)
As Integer
```

[.NET C#]

```
int CallbackS7F23(int end_status, int *ackc7, uint upara);
```

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。ackc7 に S7F24 の ACKC7 が格納されます。
9	ストールされた。
(-1)	送信エラーを検出した。
(-14)	T3タイムアウトを検出した。

3.12.3.2 EngSendS7F25() - 書式付プロセスプログラム情報問合せ関数

(1) 呼出書式

[C, C++]

```
API int APIX EngSendS7F25(
    char *ppid, // 書式付 PPID 格納領域のポインタ
    TFPP_INFO *info, // S7F26 で取得した FPP 情報格納構造体のポインタ
    int (WINAPI *s7f25Callback)(), // 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 EngSendS7F25 (
    ByVal ppid As String,
    ByRef ppinfo As dsh_info.TFPP_INFO,
    ByVal callback As vcallback.callback_S7F25,
    ByVal upara As Int32) As Int32
```

[.NET C#]

```
int EngSendS7F25(
    byte[] ppid,
    ref TFPP_INFO ppinfo,
    CallbackS7F25 callback,
    uint upara );
```

(2) 引数

ppid

装置に問合せたい書式付プロセスプログラム ID が格納されているポインタです。

info

受信した応答メッセージ S7F26 の FPP 情報を格納する構造体のポインタを指定します。

s7f25Callback

DSHEng4 によるメッセージ送信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指定できます。

コールバックの指定が=0 の場合はブロックモードになります。

upara

コールバック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。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します。

(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。 info 内に FPP 情報が格納されます。 (2) 非ブロックモード : 要求が受け付けられた。
(-1)	送信できなかった。
(-14)	T3タイムアウトを検出した。

(4) 説明

装置に書式付プロセスプログラム情報の問合せ要求を行います。S7F25 メッセージを使います。

問合せ指定は書式付プロセスプログラム ID で行います。

要求を受けた DSHeng4 は、S7F25 メッセージにエンコードし、装置に送信します。

送信要求から S7F26 応答メッセージ受信までの制御は引数の S7F25 コールバック関数指定の有無によって次のようになります。

s7f25Callback の指定	制御の流れ
なし (=0)	S7F25 送信後、応答メッセージ S7F26 が受信されるか、またはエラーを検出するまで制御が要求元に戻ってきません。 正常終了であれば、info 内に FPP 情報が格納されます。
あり	送信要求後、S7F25 の送信前に制御が戻されます。実行結果はコールバック関数で与えられます。 コールバック関数の end_status が=0 ならば正常で引数 info 内に FPP 情報が格納されます。 エラーが検出された場合、(-1)が end_status にセットされます。

DSHeng4 は S7F26 で得られた情報の装置管理情報としての登録は行いません。必要に応じてユーザ側プログラムで登録のための処理をしてください。

正常に応答メッセージを受信した場合、info 内の情報の処理を終えた後、その構造体で使用されているメモリを解放してください。

解放は次のように DshFreeTFPP_INFO()関数を使って行ってください。

```
DshFreeTFPP_INFO(info);
```

(5) コールバック関数

```
API int APIX s7f25Callback(
    int end_status,           // 実行結果
    TFPP_INFO *info,         // 応答メッセージ S7F26 の FPP 情報が格納されている構造体のポインタ
    ULONG upara              // 呼出時に指定したパラメータ
);
```

[.NET VB]

```
Function callback_S7F25(ByVal end_status As Integer, ByRef info As dsh_info.TFPP_INFO, ByVal upara As Integer) As Integer
```

[.NET C#]

```
int CallbackS7F25(int status, ref TFPP_INFO *info, uint upara);
```

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。info 内に FPP 情報が格納されます。
(-1)	送信エラーを検出した。
(-14)	T3タイムアウトを検出した。

3.12.4 FPP 書式付プロセスプログラム関連ライブラリ関数

3.12.4.1 DshDecodeS7F23() - S7F23 を書式付プロセスプログラム情報にデコード

(1) 呼出書式

[C, C++]

```
API int APIX DshDecodeS7F23(
    DSHMSG *msg,           // SECS メッセージ 情報構造体のポインタ
    TFPP_INFO *pinfo      // デコードした情報を格納する構造体のポインタ
);
```

[.NET VB]

```
Function DshDecodeS7F23 (
    ByRef msg As dshdr2.DSHMSG,
    ByRef info As dsh_info.TFPP_INFO) As Int32
```

[.NET C#]

```
int DshDecodeS7F23(
    ref DSHMSG msg,
    ref TFPP_INFO info );
```

(2) 引数

msg

S7F23 の SECS メッセージ情報が格納されている構造体のポインタです。

pinfo

デコードした書式付プロセスプログラム情報を格納する構造体のポインタです。

(3) 戻り値

戻り値	意味
0	正常にデコードできた。
(-1)	msg を正しくデコードできなかった。

(4) 説明

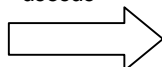
S7F23 メッセージに含まれる書式付プロセスプログラム情報を、ユーザプログラムが処理しやすい TFPP_INFO 構造体の中にデコードします。

なお、構造体使用後は、構造体内部で使用されたメモリを DshFreeTFPP_INFO()関数を使って開放してください。(DshResponseS7F24()を使う場合、DshResonseS7F24()が開放してくれます。)

msg S7F23

L,4
ppid
mdlIn
.

decode



TFPP_INFO
構造体

3.12.4.2 DshEncodeS7F23() - 書式付プロセスプログラム情報を S7F23 へエンコード

(1) 呼出書式

[C, C++]

```
API int APIX DshEncodeS7F23(
    DSHMSG *msg,           // SECS メッセージ 情報構造体のポインタ
    BYTE *buffer,         // S7F23 を格納するバッファポインタ
    int buflen,           // buffer のバイトサイズ
    TFPP_INFO *pinfo       // エンコードしたい PP 情報が格納されている構造体のポインタ
);
```

[.NET VB]

```
Function DshEncodeS7F23 (
    ByRef msg As dshdr2.DSHMSG,
    ByRef buff As Byte,
    ByVal buflen As Int32,
    ByRef info As dsh_info.TFPP_INFO) As Int32
```

[.NET C#]

```
int DshEncodeS7F23(
    ref DSHMSG msg,
    byte[] buff,
    int buflen,
    ref TFPP_INFO info );
```

(2) 引数

msg

エンコードした S7F23 メッセージを格納するメッセージ情報構造体のポインタです。

buffer

エンコードした S7F23 のテキストを格納するためのバッファポインタです。

buflen

buffer のバイトサイズです。

pinfo

エンコードしたい書式付プロセスプログラム情報が格納されている構造体のポインタです。

(3) 戻り値

戻り値	意味
0	正常にエンコードできた。
(-1)	msg を正しくエンコードできなかった。

(4) 説明

TFPP_INFO 構造体に格納されている書式付プロセスプログラム情報を、S7F23 の SECS メッセージにエンコードします。

buffer にはエンコードしたテキストを格納するために必要なメモリを準備しておく必要があります。

msg S7F23

L,4

ppid

mdlIn

← encode



3.12.4.3 DshFreeTFPP_INFO() - 書式付プロセスプログラム情報構造体メモリの開放

(1) 呼出書式

[C, C++]

```
API void APIX DshFreeTFPP_INFO(  
    TFPP_INFO *pinfo // メモリを開放したいPP情報が格納されている構造体のポインタ  
);
```

[.NET VB]

```
Sub DshFreeTFPP_INFO (  
    ByRef info As dsh_info.TFPP_INFO)
```

[.NET C#]

```
void DshFreeTFPP_INFO(  
    ref TFPP_INFO info );
```

(2) 引数

pinfo

メモリを解放したい書式付プロセスプログラム情報構造体のポインタです。

(3) 戻り値

なし。

(4) 説明

TFPP_INFO 構造体内で情報格納用に使用されているメモリを全て解放します。

開放した後、TFPP_INFO の内容を全て0で初期設定します。

pinfo が NULL ならば、何も処理しません。

3.12.4.4 DshCopyTFPP_INFO() - 書式付プロセスプログラム情報構造体メモリのコピー

(1) 呼出書式

[C, C++]

```
API int APIX DshCopyTFPP_INFO(
    TFPP_INFO *dinfo,           // 北°-先のポインタ
    TFPP_INFO *sinfo           // 北°-元のポインタ
);
```

[.NET VB]

```
Function DshCopyTFPP_INFO (
    ByRef info As dsh_info.TFPP_INFO,
    ByRef sinfo As dsh_info.TFPP_INFO) As Int32
```

[.NET C#]

```
int DshCopyTFPP_INFO(
    ref TFPP_INFO info,
    ref TFPP_INFO sinfo );
```

(2) 引数

pinfo

コピー先の書式付プロセスプログラム情報を格納する構造体メモリのポインタです。

sinfo

コピー元の書式付プロセスプログラム情報が格納されている構造体メモリのポインタです。

(3) 戻り値

戻り値	意味
0	正常に北°-できた。
(-1)	sinfo または dinfo の値が NULL だったので北°-できなかった。

(4) 説明

sinfo が指す TFPP_INFO 構造体内に格納されている書式付プロセスプログラム情報を dinfo で指定された TFPP_INFO 構造体にコピーします。

dinfo 内のメンバーで新しいメモリが必要なものは本関数が取得します。

dinfo 内メンバーで確保されたメモリは、使用后、DshFreeTFPP_INFO()関数を使って開放してください。

3.12.4.5 DshInitFppInfo 書式付プロセスプログラム TFPP_INFO の初期設定

(1) 呼出書式

[C, C++]

```
API int APIX DshInitFppInfo(
    TFPP_INFO *info,           // TFPP_INFO 構造体のポインタ
    char *ppid,               // 書式付きプロセスプログラム ID
    char *mdlIn,              // model 名
    char *softrev,           // soft revision
    int ccode_count           // コマンドコード情報リストのサイズ
);
```

[.NET VB]

```
Sub DshInitFppInfo (
    ByRef info As dsh_info.TFPP_INFO,
    ByVal ppid As String,
    ByVal mdlIn As String,
    ByVal softrev As String,
    ByVal ccode_count As Int32)
End Sub
```

[.NET C#]

```
void DshInitFppInfo(
    ref TFPP_INFO info,
    byte[] ppid,
    byte[] mdlIn,
    byte[] softrev,
    int ccode_count );
```

(2) 引数

info

書式付プロセスプログラム TFPP_INFO 構造体のポインタです。このメンバーを初期設定します。

ppid

設定する書式付きプロセスプログラム ID (文字列) です。

mdlIn

書式付きプロセスプログラムのモデル名です。

softrev

書式付きプロセスプログラムの Software Revision です。

ccode_count

TFPP_INFO に含むコマンドコードの最大数です。

(3) 戻り値

なし。

(4) 説明

本関数は APP が OFFLINE で書式付きプロセスプログラム情報を生成する際に使用することができます。最初に info 内をクリアします。そして、引数で指定された情報を info 内に設定します。メモリが必要なメンバーについてはメモリを確保し情報をコピーします。



また、書式付きプロセスプログラムのコマンドコード情報の設定については次の `DshPutFppCCode()` 関数を使ってください。

`TFPP_INFO` 構造体の使用後は `DshFreeTFPP_INFO()` 関数を使って構造体内部で使用したメモリを開放してください。

3.12.4.6 DshPutFppCCode() 書式付プロセスプログラムコマンドコード情報の追加

(1) 呼出書式

[C, C++]

```
API int APIX DshPutFppCCode(
    TFPP_INFO *info,           // 書式付きプロセスプログラム情報構造体のポインタ
    int ccode_fmt,           // ccode 値の format
    int ccode_size,         // ccode 値の配列サイズ
    int max_ccode_size,     // ccode 値の最大配列サイズ
    void *ccode,            // ccode 値が格納されている領域のポインタ
    int ppara_count        // 付属パラメータの数
);
```

[.NET VB]

```
Function DshPutFppCCode (
    ByRef info As dsh_info.TFPP_INFO,
    ByVal ccode_fmt As Int32,
    ByVal ccode_size As Int32,
    ByVal max_ccode_size As Int32,
    ByVal ccode As String,
    ByVal ppara_count As Int32) As Int32
```

[.NET C#]

```
int DshPutFppCCode(
    ref TFPP_INFO info,
    int ccode_fmt,
    int ccode_size,
    int max_ccode_size,
    byte[] ccode,
    int ppara_count );
```

(2) 引数

info

書式付きプロセスプログラム情報構造体のポインタです。

ccode_fmt

コマンドコード値のフォーマットです。

ccode_size

コマンドコード値の配列サイズです。

max_ccode_size

コマンドコード値の最大配列サイズです。

ccode

コマンドコード値が格納されている領域のポインタです。

ppara_count

コマンドコード情報に付属するパラメータの数です。

(3) 戻り値

戻り値	意味
0	正常に追加できた。

(-1)	コマドコードの数が指定数を超えている。
------	---------------------

(4) 説明

先にDshInitFppInfo()で初期設定された書式付きプロセスプログラム情報構造体内のccode_listリストにコマンドコード情報TFPP_CC CODEを1個加えます。

ccode_list内のコマンドコード情報リスト上に、本関数が実行される順に情報を追加していきます。

設定後0を返却します。

もし、info内のccode_countで指定された分のコマンドコードが既に設定済みであった場合は、(-1)を返却します。

TFPP_CC CODE内へのパラメータ情報の設定はDshPutFppPPara()関数を使って行ってください。

3.12.4.7 DshPutFppPPara() 書式付プロセスプログラムパラメータ情報の追加

(1) 呼出書式

[C, C++]

```
API int APIX DshPutFppPPara(
    TFPP_CC CODE *info,           // 書式付きプロセスプログラムコマンド情報構造体のポインタ
    int ppara_fmt,               // ppara 値の format
    int ppara_size,             // ppara 値の配列サイズ
    int max_ppara_size,         // ppara 値の最大配列サイズ
    void *ppara,                 // ppara 値が格納されている領域のポインタ
);
```

[.NET VB]

```
Function DshPutFppPPara (
    ByRef info As dsh_info.TFPP_CC CODE,
    ByVal ppara_fmt As Int32,
    ByVal ppara_size As Int32,
    ByVal max_ppara_size As Int32,
    ByVal ppara As IntPtr) As Int32
```

```
Function DshPutFppPParaInfo (
    ByRef info As dsh_info.TFPP_INFO,
    ByVal order As Int32,
    ByVal ppara_fmt As Int32,
    ByVal ppara_size As Int32,
    ByVal max_ppara_size As Int32,
    ByVal ppara As String) As Int32
```

```
Function DshPutFppPParaInfo (
    ByRef info As dsh_info.TFPP_INFO,
    ByVal order As Int32,
    ByVal ppara_fmt As Int32,
    ByVal ppara_size As Int32,
    ByVal max_ppara_size As Int32,
    ByVal ppara As IntPtr) As Int32
```

[.NET C#]

```
int DshPutFppPPara(
    ref TFPP_CC CODE info,
    int ppara_fmt,
    int ppara_size,
    int max_ppara_size,
    byte[] ppara );
```

(2) 引数

info

書式付きプロセスプログラムコマンドコード情報構造体のポインタです。

ppara_fmt

パラメータ値のフォーマットです。

ppara_size

パラメータ値の配列サイズです。

max_ppara_size

パラメータ値の最大配列サイズです。

ppara

パラメータ値が格納されている領域のポインタです。

(3) 戻り値

戻り値	意味
0	正常に追加できた。
(-1)	パラメータの数が指定数を超えている。

(4) 説明

先に DshPutFppCCode() で設定された書式付きプロセスプログラムのコマンドコード情報構造体リスト ppara_list にパラメータ情報 TFPP_PPARG を 1 個加えます。

ppara_list 内のパラメータ情報リスト上に、本関数が実行される順に情報を追加していきます。

設定後 0 を返却します。

もし、info 内の ppara_count で指定された分のパラメータが既に設定済みであった場合は、(-1) を返却します。

3.12.4.8 DshPutFppPParaInfo() 書式付プロセスプログラムパラメータの配列指定追加

(1) 呼出書式

[C, C++]

```
API int APIX DshPutFppPParaInfo(
    TFPP_INFO *info,           // 書式付きプロセスプログラム情報構造体のポインタ
    int index,                 // ccode_list[]配列インデックス
    int ppara_fmt,            // ppara 値の format
    int ppara_size,           // ppara 値の配列サイズ
    int max_ppara_size,       // ppara 値の最大配列サイズ
    void *ppara,              // ppara 値が格納されている領域ポインタ
);
```

[.NET VB]

```
Function DshPutFppPParaInfo (
    ByRef info As dsh_info.TFPP_INFO,
    ByVal order As Int32,
    ByVal ppara_fmt As Int32,
    ByVal ppara_size As Int32,
    ByVal max_ppara_size As Int32,
    ByVal ppara As String) As Int32
```

```
Function DshPutFppPParaInfo (
    ByRef info As dsh_info.TFPP_INFO,
    ByVal order As Int32,
    ByVal ppara_fmt As Int32,
    ByVal ppara_size As Int32,
    ByVal max_ppara_size As Int32,
    ByVal ppara As IntPtr) As Int32
```

[.NET C#]

```
int DshPutFppPParaInfo(
    ref TFPP_INFO pinfo,
    int order,
    int ppara_fmt,
    int ppara_size,
    int max_ppara_size,
    byte[] ppara );
```

(2) 引数

info
書式付きプロセスプログラム情報構造体のポインタです。

index
ccode_list[]配列の位置(0,1,2...)

ppara_fmt
パラメータ値のフォーマットです。

ppara_size
パラメータ値の配列サイズです。

max_ppara_size

パラメータ値の最大配列サイズです。

ppara

パラメータ値が格納されている領域のポインタです。

(3) 戻り値

戻り値	意味
0	正常に追加できた。
(-1)	パラメータの数が指定数を超えている。

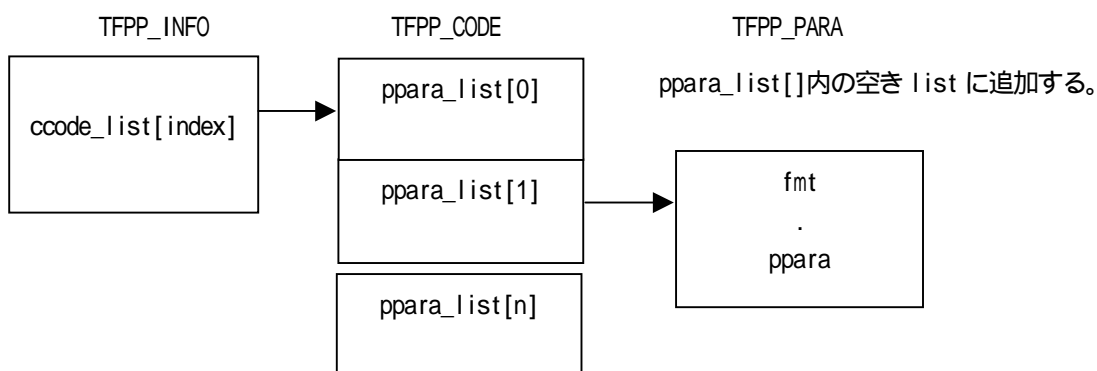
(4) 説明

先に DshPutFppCCode() で設定された書式付きプロセスプログラムのコマンドコード情報構造体リスト ppara_list の index で指定された配列位置にパラメータ情報 TFPP_PPARA を 1 個加えます。

ppara_list 内のパラメータ情報リスト上に、本関数が実行される順に情報を追加していきます。

設定後 0 を返却します。

もし、index の値が ppara_list の配列サイズを超えている場合、または、ppara_list の ppara_count で指定された分のパラメータが既に設定済みであった場合は、(-1)を返却します。



3.12.4.9 DshMakeS7F23Response() - S7F23 の応答メッセージの生成

(1) 呼出書式

[C, C++]

```
API int APIX DshMakeS7F23Response(
    BYTE ackc7,           // S7F24 に設定する ACKC7 です。
    DSHMSG *msg,         // S7F24 メッセージを格納するメッセージ構造体のポインタ
    BYTE *buff,          // S7F24 のテキスト格納バッファポインタ
    int buff_size        // buff のバイトサイズ
);
```

[.NET VB]

```
Function DshMakeS7F23Response (
    ByVal ackc7 As Int32,
    ByRef info As dsh_info.TFPP_INFO,
    ByRef msg As dshdr2.DSHMSG,
    ByRef buff As Byte,
    ByVal buff_size As Int32) As Int32
```

[.NET C#]

```
int DshMakeS7F23Response(
    int ackc7,
    ref TFPP_INFO info,
    ref DSHMSG msg,
    byte[] buff,
    int buff_size );
```

(2) 引数

ackc7

S7F24 メッセージの ACKC7 です。

msg

S7F24 応答メッセージ情報を格納するためのメッセージ構造体のポインタです。

buff

S7F24 応答メッセージのテキストを格納するためのバッファポインタです。

buff_size

buff のバイトサイズです。

(3) 戻り値

戻り値	意味
0	正常に生成できた。
(-1)	生成できなかった。(buff 領域不足)

(4) 説明

S7F23 に対する S7F24 応答メッセージを msg, buff 内に作成します。

応答情報は、ackc7 を S7F24 の ACKC7 として設定します。

ackc7 の設定はユーザが書式付 PP 情報を評価した結果です。

3.12.4 ユーザ作成ライブラリ関数

3.12.4.1 DshResponseS7F24() S7F24 書式付き PP 送信応答メッセージ

(1) 呼出書式

[C, C++]

```
API int APIX DshResponseS7F24(
    ID_TR trid,           // DSHDR2 のトランザクション ID
    TFPP_INFO *info,     // 書式付き PP 送信メッセージ 情報格納領域のポインタ
    int ackc7           // S7F24 応答 ACK
);
```

[.NET VB]

```
Function DshResponseS7F24 (
    ByVal trid As Int32,
    ByRef info As dsh_info.TFPP_INFO,
    ByVal ackc7 As Int32) As Int32
```

[.NET C#]

```
int DshResponseS7F24(
    uint trid,
    ref TFPP_INFO info,
    int ackc7 );
```

(2) 引数

trid

S7F23 信時に DSHEng4 から渡される DSHDR2 通信ドライバーのトランザクション管理のための ID です。

info

書式付き PP 送信情報が格納されている構造体のポインタです。

ackc7

送信する応答メッセージ S7F24 の許可情報

(3) 戻り値

戻り値	意味
0	正常に送信できた。
(-1)	送信できなかった。

(4) 説明

書式付き PP 送信メッセージ S7F23 に対する応答メッセージを送信します。

本関数はユーザ作成ライブラリ DLL (DSHEngU.dll) に含まれる関数ですが、ここでは DSHEng4 パッケージに標準的な関数として付属されているものです。(ユーザ独自による作成も可能です)

引数に指定されている ackc7 情報から S7F24 メッセージを組み立て、その後、S7F24 メッセージを送信します。

なお、S7F24 メッセージの組み立てに、DshMakeS7F24Response() 関数を使用できます。

3.12.4.2 DshResponseS726() S7F26 PP 要求応答メッセージ

(1) 呼出書式

[C, C++]

```
API int APIX DshResponseS726(
    ID_TR trid,                // DSHDR2 のトランザクション ID
    TFPP_INFO *info,          // 書式付き PP 要求メッセージ 情報格納領域のポインタ
    int ackc7                 // S726 応答 ACK
);
```

[.NET VB]

```
Function DshResponseS7F26 (
    ByVal trid As Int32,
    ByRef info As dsh_info.TFPP_INFO,
    ByVal ackc7 As Int32) As Int32
```

[.NET C#]

```
int DshResponseS7F26(
    uint trid,
    ref TFPP_INFO info,
    int ackc7 );
```

(2) 引数

trid

S7F25 信時に DSHEng4 から渡される DSHDR2 通信ドライバーのトランザクション管理のための ID です。

info

書式付き PP 要求情報が格納されている構造体のポインタです。この内容が応答されます。

ackc7

送信する応答メッセージ S726 の許可情報

(3) 戻り値

戻り値	意味
0	正常に送信できた。
(-1)	送信できなかった。

(4) 説明

書式付き PP 要求メッセージ S7F25 に対する応答メッセージを送信します。

本関数はユーザ作成ライブラリ DLL (DSHEngU.dll) に含まれる関数ですが、ここでは DSHEng4 パッケージに標準的な関数として付属されているものです。(ユーザ独自による作成も可能です)

引数に指定されている info と ackc7 情報から S7F26 メッセージを組み立て、その後、S726 メッセージを送信します。

ackc7 が 0 以外の値 (エラー) であれば、List=0 のメッセージを応答します。

なお、S7F26 メッセージの組み立てに、DshEncodeFpp () 関数を使用できます。

送信後は、info の内部で使用されているメモリを DshFreeTFPP_INFO() で開放します