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-１５／１５

３．24　オブジェクト関連メッセージ応答情報とエラー情報設定ライブラリ関数

３．25　その他のライブラリ関数

２００９年７月

株式会社データマップ

文書番号　DSHEng4-09-3032F-00

［取り扱い注意］

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

【改訂履歴】

番号	改訂日付	項 目	概 略
1.	2009.7	初版	

目　次

（VOL End）

３．24　オブジェクト関連メッセージの応答情報とエラー情報関連設定ライブラリ関数

オブジェクトを対象とするメッセージは共通の応答情報とそれに含まれるエラー情報を持っています。
DSHEng4 ではこの応答情報とエラー情報を TOBJ_ERR_INFO と TERR_INFO 構造体を使って情報を格納し、応答メッセージの処理を行います。

応答情報 TOBJ_ERR_INFO は objack と エラー情報の数と、その数分の TERR_INFO のリストから構成されます。

```
TOBJ_ERR_INFO   -  int objack
                   int err_count
                   TERR_INFO **err_list     -    error list
```

本 TOBJ_ERR_INFO と TERR_INFO を使用するメッセージには次のものあります。ただし、S3F17, S15F3 などの構造体には別の名前が付いています。

S3F17,
S14F9, S14F11
S15F3, S15F5, S15F13, S15F17

ストリーム 16 (S16Fxx)の応答メッセージについては、応答情報は TOBJ_ERR_INFO ではなく、それぞれ独自の構造体を使用します。ただしその応答情報の中に、TERR_INFO の構造体が使用されます。

S16F5
S16F15, S16F17, S16F27

（1）ライブラリ関数一覧

	ﾗｲﾌﾞﾗﾘ関数名	機能
1	DshInitTOBJ_ERR_INFO()	ｵﾌﾞｼﾞｪｸﾄ関連２次ﾒｯｾｰｼﾞ応答情報を初期設定します。
2	DshInitTERR_INFO_LIST()	ｴﾗｰ情報 LIST を初期設定します。
3	DshPutTERR_INFO_LIST()	ｴﾗｰ情報 LIST にｴﾗｰ情報を１個追加します。
4	DshPutTERR_INFO()	ｴﾗｰ情報を１個 TERR_INFO 構造体に設定します。
5	DshFreeTERR_INFO_LIST()	ｴﾗｰ情報ﾘｽﾄで使用されたﾒﾓﾘを開放します。ﾘｽﾄに使用されているﾒﾓﾘも開放します。

３．24．１　使用する情報格納構造体

次の TERR_INFO がエラー情報構造体です。

（1）エラー情報格納構造体 TERR_INFO

```
typedef struct{
        int     errcode;                    // エラーコード (U1)
        char    *errtext;                   // エラーテキスト
        } TERR_INFO;
```

（2）応答情報格納構造体

```
typedef struct{
        int         objack;
        int         err_count;
        TERR_INFO   **err_list;
        } TOBJ_ERR_INFO;
```

メッセージによって構造体の名前と objack 名が次のようになります。

2 次メッセージ	構造体名	ack 名
S3F18	TCACT_ERR_INFO	caack
S14F10, S14F12	TOBJ_ERR_INFO	objack
S15F4, S15F6, S15F14, S15F18	TRCP_ERR_INFO	rmack

（3）S16F12, S16F16　プロセスジョブ生成要求応答情報

```
typedef struct{
        int         prj_count;          // S16F12 の場合は、prj_count = 1
        char        **prj_list;
        int         acka;               // Boolean
        int         err_count;
        TERR_INFO   **err_list;
} TPRJ_ERR_INFO;
```

（4）S16F6　プロセスジョブコマンド応答情報

```
typedef struct{
        char        *prjobid;
        int         acka;               // Boolean
        int         err_count;
        TERR_INFO   **err_list;
} TPRJ_CMD_ERR_INFO;
```

（5）プロセスジョブ　デキューコマンド応答情報

```
typedef struct{
        int         prj_count;          // # of job dequed
        char        **prj_list;
        int         acka;               // Boolean
        int         err_count;
        TERR_INFO   **err_list;
} TPRJ_DEQ_ERR_INFO;
```

３．24．２　オブジェクトエラー情報設定関連関数

３．24．２．１　DshInitTOBJ_ERR_INFO () - オブジェクト応答情報の初期化

（1）呼出書式

[c,C++]

```
API void APIX   DshInitTOBJ_ERR_INFO(
        TOBJ_ERR_INFO *info,                // ｴﾗｰ情報格納構造体ﾘｽﾄのﾎﾟｲﾝﾀ
        int         objack,                 // ackﾃﾞｰﾀ
        int         err_count               // ｴﾗｰ情報のﾘｽﾄｻｲｽﾞ (個数  0,1,2...)
);
```

[.NET VB]

```
Sub DshInitTOBJ_ERR_INFO (
        ByRef obj_errinfo As dsh_info.TOBJ_ERR_INFO,
        ByVal objack As Int32,
        ByVal errcount As Int32)
```

[.NET C#]

```
void DshInitTOBJ_ERR_INFO(
        ref TOBJ_ERR_INFO obj_errinfo,
        int objack,
        int errcount );
```

（2）引数

info

TOBJ_ERR_INFO 応答情報構造体のポインタです。

objack

objack - ACK の値です。

err_count

エラー情報構造体の数です。 = 0 の場合はエラー情報がないことになります。

（3）戻り値

なし。

（4）説明

本関数は、オブジェクト関連応答メッセージ TOBJ_ERR_INFO 構造体に初期設定を行います。
info で指定された構造体の objack メンバーに引数 objack の値を設定し、err_count メンバーにも引数 err_count の値を設定します。
もし、err_count > 0 の場合は、err_list に err_count だけの TERR_INFO エラー情報構造体のポインタリストを設けます。

３．24．２．２　DshPutTOBJ_ERR_INFO () - オブジェクトエラー情報の設定

（1）呼出書式

[c,C++]

```
API void APIX   DshPutTOBJ_ERR_INFO (
        TOBJ_ERR_INFO  *errinfo,            // ｴﾗｰ情報格納構造体ﾘｽﾄのﾎﾟｲﾝﾀ
        int         errcode,                // error code
        char        *errtext                // error text
);
```

[.NET VB]

```
Function DshPutTOBJ_ERR_INFO (
         ByRef errinfo As dsh_info.TOBJ_ERR_INFO,
         ByVal errcode As Int32,
         ByVal errtext As String) As Int32
```

[.NET C#]

```
int DshPutTOBJ_ERR_INFO(
         ref TOBJ_ERR_INFO errinfo,
         int errcode,
         byte[] errtext );
```

（2）引数

errinfo

　　オブジェクトエラー情報構造体のポインタです。

errcode

　　設定するエラーコードです。（メッセージ内のアイテムは U1(51)です。）

errtext

　　設定するエラーテキスト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	正常に設定できた。
(-1)	ﾘｽﾄが満杯で設定できなかった。

（4）説明

本関数は、errinfo 内の errlist リストの先頭から空きリストを探します。
もし、空きリストがなければ、(-1)を返却します。
もし、空きリストがあれば、その空きリストに１個 TERR_INFO 構造体領域を設け、その構造体に errcode と errtext を格納し、0 を返却します。TERR_INFO と内部メンバーのメモリは本関数が取得します。

本関数の実行前に DshInitTOBJ_ERR_INFO()関数を使って errinfo を初期化しておく必要があります。

３．24．２．３　DshPutTERR_INFO_LIST() - オブジェクトエラー情報のリストへの設定

（1）呼出書式

[c,C++]

```
API int APIX   DshPutTERR_INFO_LIST(
        TERR_INFO  **errlist,          // ｴﾗｰ情報格納構造体ﾘｽﾄのﾎﾟｲﾝﾀ
        int        list_size,          // ﾘｽﾄのｻｲｽﾞ
        int        errcode,            // 設定するｴﾗｰｺｰﾄﾞ
        char       *errcode            // 設定するｴﾗｰﾃｷｽﾄのﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Function DshPutTERR_INFO_LIST (
        ByRef erlist As IntPtr,
        ByVal list_size As Int32,
        ByVal errcode As Int32,
        ByVal errtext As String) As Int32
```

[.NET C#]

```
int DshPutTERR_INFO_LIST(
        ref TERR_INFO erlist,
        int list_size,
        int errcode,
        byte[] errtext );
```

（2）引数

errlist

エラー情報構造体リストのポインタです。

list_size

errlist のリストのサイズです。(設定できるエラー情報の最大個数です)

errcode

設定するエラーコードです。(メッセージ内のアイテムは U1(51)です。)

errtext

設定するエラーテキスト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	正常に設定できた。
(-1)	ﾘｽﾄが満杯で設定できなかった。

（4）説明

本関数は、errlist の先頭から空きリストを探します。

もし、空きリストがなければ、(-1)を返却します。

もし、空きリストがあれば、その空きリストに１個 TERR_INFO 構造体領域を設け、その構造体に ercode と errtext を格納し、０を返却します。

３．24．２．４　DshPutTERR_INFO () - オブジェクトエラー情報の設定

（1）呼出書式

[c,C++]

```
API void APIX   DshPutTERR_INFO(
        TERR_INFO  *erinfo,                 // ｴﾗｰ情報格納構造体のﾎﾟｲﾝﾀ
        int        errcode,                 // 設定するｴﾗｰｺｰﾄﾞ
        char       *errtext                 // 設定するｴﾗｰﾃｷｽﾄのﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Sub DshPutTERR_INFO (
        ByRef erinfo As dsh_info.TERR_INFO,
        ByVal errcode As Int32,
        ByVal errtext As String)
```

[.NET C#]

```
void DshPutTERR_INFO(
        ref TERR_INFO erinfo,
        int errcode,
        byte[] errtext );
```

（2）引数

erinfo

エラー情報構造体のポインタです。

errcode

設定するエラーコードです。（メッセージ内のアイテムは U1(51)です。）

errtext

設定するエラーテキストが格納されている領域のポインタです。

（3）戻り値

なし。

（4）説明

erinfo で指定された TERR_INFO 構造体に ercode と errtext を格納します。
構造体内の errtext メンバーに必要なメモリは本関数が取得した上でエラーテキストを格納します。

本関数を最初に実行する前に、DshInitTERR_INFO_LIST()関数を使ってリストの初期設定を行ってください。

３．24．２．５　DshFreeTERR_INFO_LIST() - エラー情報構造体リストメモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTERR_INFO_LIST(
     TERR_INFO  **errlist,                  // ﾒﾓﾘを開放したいｴﾗｰ情報構造体ﾘｽﾄのﾎﾟｲﾝﾀ
     int        list_size                   // ﾘｽﾄのｻｲｽﾞ
);
```

[.NET VB]

```
Sub DshFreeTERR_INFO_LIST (
        ByRef erlist As IntPtr,
        ByVal list_size As Int32)
```

[.NET C#]

```
void DshFreeTERR_INFO_LIST(
        ref TERR_INFO erlist,
        int list_size );
```

（2）引数

errlist

　　メモリを解放したいエラー情報構造体リストのポインタです。

list_size

　　リストのサイズです。

（3）戻り値

なし。

（4）説明

errlist のリストに含まれる list_size 分のエラー情報構造体で使用されているメモリを開放します。
また、errlist に使用されているメモリそのものも開放します。

３．24．２．６　DshFreeTERR_INFO () - エラー情報構造体内メモリの開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeTERR_INFO(
     TERR_INFO  *erinfo                    // ﾒﾓﾘを開放したいｴﾗｰ情報構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Sub DshFreeTERR_INFO (
        ByRef erinfo As dsh_info.TERR_INFO)
```

[.NET C#]

```
void DshFreeTERR_INFO(
        ref TERR_INFO erinfo );
```

（2）引数

erinfo

メモリを解放したいエラー情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

erinfo 内の errtext メンバーに使用されているメモリを開放します。

erinfo 自体のメモリの開放は行いません。

３．24．３　オブジェクトタイプ、属性名取得関連ライブラリ関数

本節では、S3F17、S14F9、S14F11、S14F19、S14F21、S16F5、S16F27 メッセージで使用されるコマンド名、属性名など文字列で表現される情報について以下の機能を提供する関数です。

（1）コマンド名などの情報をシステムに合わせて設定変更するための関数

（2）コマンド名などの情報をインデクス値に変換するため、またインデクス値からコマンド名に変換する関数

関数の種類と設定変更関数ばほぼ同じように準備されています。

３．24．３．１　S3F17 キャリアアクション関連

（1）キャリアアクション名

① 関数一覧表

	関数名とプロトタイプ	機能
1	API int APIX DshSetCaActionName(int index, // ｱｸｼｮﾝ index char *name // ｱｸｼｮﾝ名文字列);	index 値に対応した Action 名を設定します。 index 値とﾃﾞﾌｫﾙﾄのｱｸｼｮﾝ名については、下の表を参照ください。
2	API int APIX DshGetCaActionIndex(char *ca_action // ｱｸｼｮﾝ名);	ca_action で指定された名前のｱｸｼｮﾝに対するｲﾝﾃﾞｸｽ値を取得します。
3	API char *APIX DshGetCaActionName(int index // ｱｸｼｮﾝｲﾝﾃﾞｸｽ);	index で指定されたｱｸｼｮﾝ名を取得します。 ｱｸｼｮﾝ名のﾎﾟｲﾝﾀで与えられます。

② アクションインデクス値とデフォルトアクション名一覧表

action ｲﾝﾃﾞｸｽ値	ｲﾝﾃﾞｸｽ用ﾏｸﾛ名	ﾃﾞﾌｫﾙﾄの文字列
0	CA_Bind	"Bind"
1	CA_CancelBind	"CancelBind"
2	CA_CancelAllCarrierOut	"CancelAllCarrierOut"
3	CA_CancelCarrier	"CancelCarrier"
4	CA_CancelCarrierAtPort	"CancelCarrierAtPort"
5	CA_CancelCarrierNotification	"CancelCarrierNotification"
6	CA_CancelCarrierOut	"CancelCarrierOut"
7	CA_CarrierIn	"CarrierIn"
8	CA_CarrierNotification	"CarrierNotification"
9	CA_CarrierOut	"CarrierOut"
10	CA_CarrierReCreate	"CarrierReCreate"
11	CA_CarrierRelease	"CarrierRelease"
12	CA_ProceedWithCarrier	"ProceedWithCarrier"

（注）インデクス値のマクロは dsh_lib.h ファイルにあります。

（2）属性 ID(AttributeID)

① 関数一覧表

	関数名とプロトタイプ	機能
1	API int APIX DshSetCaAttrName(int index, // 属性 ID index char *name // 属性 ID 文字列);	index 値に対応した Attr 名を設定します。 index 値とﾃﾞﾌｫﾙﾄの属性 ID については、下の表を参照ください。
2	API int APIX DshGetCaAttrIndex(char *ca_attr // ｱｸｼｮﾝ名);	ca_attr で指定された名前の属性 ID に対するｲﾝﾃﾞｸｽ値を取得します。
3	API char *APIX DshGetCaAttrName(int index // 属性 ID ｲﾝﾃﾞｸｽ);	index で指定された属性 ID を取得します。 属性 ID のﾎﾟｲﾝﾀで与えられます。

② 属性 ID インデクス値とデフォルト属性 ID 一覧表

属性 ID ｲﾝﾃﾞｸｽ値	ｲﾝﾃﾞｸｽ用ﾏｸﾛ名	ﾃﾞﾌｫﾙﾄの文字列
0	CA_ObjId	"ObjId"
1	CA_Capacity	"Capacity"
2	CA_CarrierAccessingStatus	"CarrierAccessingStatus"
3	CA_CarrierIDStatus	"CarrierIDStatus"
4	CA_ContentMap	"ContentMap"
5	CA_LocationID	"LocationID"
6	CA_SlotMap	"SlotMap"
7	CA_SlotMapStatus	"SlotMapStatus"
8	CA_SubStrateCoun	"SubStrateCount"
9	CA_Usage	"Usage"

３．24．３．２　S14F9 オブジェクト生成関連

（1）オブジェクトタイプ (ControlJob / Substrate)

① 関数一覧表

	関数名とプロトタイプ	機能
1	API int APIX DshSetObjTypeName(int index, char *name);	index 値に対応したﾀｲﾌﾟ名を設定します。 index 値とﾃﾞﾌｫﾙﾄのﾀｲﾌﾟ名については、下の表を参照ください。
2	API int APIX DshGetObjTypeIndex(char *objtype);	objtype で指定された名前のｵﾌﾞｼﾞｪｸﾄﾀｲﾌﾟに対するｲﾝﾃﾞｸｽ値を取得します。
3	API char* APIX DshGetObjTypeByIndex(int index);	index で指定されたｵﾌﾞｼﾞｪｸﾄﾀｲﾌﾟ名を取得します。 ｵﾌﾞｼﾞｪｸﾄﾀｲﾌﾟ名のﾎﾟｲﾝﾀで与えられます。

② オブジェクトタイプインデクス値とデフォルトオブジェクトタイプ名一覧表

ｵﾌﾞｼﾞｪｸﾄﾀｲﾌﾟ ｲﾝﾃﾞｸｽ値	ｲﾝﾃﾞｸｽ用ﾏｸﾛ名	ﾃﾞﾌｫﾙﾄの文字列
0	EN_ControlJob	"ControlJob "
1	EN_Substrate	"Substrate "

（注）インデクス値のマクロは dsh_lib.h ファイルにあります。

（2）Control Job **オブジェクト属性** ID(AttributeID)

① 関数一覧表

	関数名とプロトタイプ	機能
1	API int APIX DshSetCjObjAttrIdName (int index, // 属性 ID index char *name // 属性 ID 文字列);	index 値に対応した Attr 名を設定します。 index 値とﾃﾞﾌｫﾙﾄの属性 ID については、下の表を参照ください。
2	API int DshGetObjAttrIdIndex (char *cj_attr // ｱｸｼｮﾝ名);	cj_attr で指定された名前の属性 ID に対するｲﾝﾃﾞｸｽ値を取得します。
3	API char * DshGetCjObjAttrIdByIndex (int index // 属性 ID ｲﾝﾃﾞｸｽ);	index で指定された属性 ID を取得します。 属性 ID のﾎﾟｲﾝﾀで与えられます。

② 属性 ID インデクス値とデフォルト属性 ID 一覧表

属性 ID ｲﾝﾃﾞｸｽ値	ｲﾝﾃﾞｸｽ用ﾏｸﾛ名	ﾃﾞﾌｫﾙﾄの文字列
0	EN_ObjID	"ObjID"
1	EN_CarrierInputSpec	"CarrierInputSpec"
2	EN_CurrentPRJob	"CurrentPRJob"
3	EN_DataCollectionPlan	"DataCollectionPlan"
4	EN_MtrlOutByStatus	"MtrlOutByStatus"
5	EN_MtrlOutSpec	"MtrlOutSpec"
6	EN_PauseEvent	"PauseEvent"
7	EN_ProcessingCtrlSpec	"ProcessingCtrlSpec"
8	EN_ProcessingOrderMgmt	"ProcessOrderMgmt"
9	EN_PRJobStatusList	"PRJobStatusList"
10	EN_StartMethod	"StartMethod"
11	EN_State	"State"

（3）Substrate Object 属性 ID (AttributeID)

① 関数一覧表

	関数名とプロトタイプ	機能
1	API int APIX DshSetSubstObjAttrIdName (int index, // 属性 ID index char *name // 属性 ID 文字列);	index 値に対応した Attr 名を設定します。 index 値とﾃﾞﾌｫﾙﾄの属性 ID については、下の表を参照ください。
2	API int DshGetSubstObjAttrIdIndex (char *subst_attr // 属性名);	subst_attr で指定された名前の属性 ID に対するｲﾝﾃﾞｸｽ値を取得します。
3	API char * DshGetSubstObjAttrIdByIndex (int index // 属性 ID ｲﾝﾃﾞｸｽ);	index で指定された属性 ID を取得します。 属性 ID のﾎﾟｲﾝﾀで与えられます。

② 属性 ID インデクス値とデフォルト属性 ID 一覧表

属性 ID ｲﾝﾃﾞｸｽ値	ｲﾝﾃﾞｸｽ用ﾏｸﾛ名	ﾃﾞﾌｫﾙﾄの文字列
0x10	SO_ObjID	"ObjID"
0x11	SO_AcquiredID	"AcquiredID"
0x12	SO_BatchLocID	"BatchLocID"
0x13	SO_LotID	"LocID"
0x14	SO_MaterialStatus	"MaterialStatus"
0x15	SO_SubstDestination	"SubstDestination"
0x16	SO_SubstHistory	"SubstHistry"
0x17	SO_SubstIDStatus	"SubstIDStatus"
0x18	SO_SubstLocID	"SubstLocID"
0x19	SO_SubstProcState	"SubstProcState"
0x1a	SO_SubstSource	"SubstSource"
0x1b	SO_SubstState	"SubstState"
0x1c	SO_SubstType	"SubstType"
0x1d	SO_SubstUsage	"SubstUsage"

３．24．３．３　S16F27 オブジェクトコマンド関連

（1）Control Job コマンド

① 関数一覧表

	関数名とプロトタイプ	機能
1	API int APIX DshSetCjCmdName (int index, // cj_cmd ｲﾝﾃﾞｸｽ char *name // cj_cmd 文字列);	index 値に対応したｺﾏﾝﾄﾞ名を設定します。 index 値とﾃﾞﾌｫﾙﾄのｺﾏﾝﾄﾞ名については、下の表を参照ください。
2	API int DshGetCjCmdIndex (char *cj_cmd // CJ ｺﾏﾝﾄﾞ名);	cj_cmd で指定された名前のｺﾏﾝﾄﾞに対するｲﾝﾃﾞｸｽ値を取得します。
3	API char * DshGetCjCmdByIndex (int index //cj_cmd ｲﾝﾃﾞｸｽ);	index で指定されたｺﾏﾝﾄﾞ名を取得します。 ｺﾏﾝﾄﾞ名のﾎﾟｲﾝﾀで与えられます。

② ｺﾏﾝﾄﾞ ID インデクス値とデフォルトｺﾏﾝﾄﾞ ID 一覧表

ｺﾏﾝﾄﾞ ID ｲﾝﾃﾞｸｽ値	ｲﾝﾃﾞｸｽ用ﾏｸﾛ名	ﾃﾞﾌｫﾙﾄの文字列
0	CJ_Start	"CjStart"
1	CJ_Pause	"CjPause"
2	CJ_Resume	"CjResume"
3	CJ_Cancel	"CjCancel"
4	CJ_Deselect	"CjDeselect"
5	CJ_Stop	"CjStop"
6	CJ_Abort	"CjAbort"
7	CJ_CJHOQ	"CJHOQ"

（2）Substrate コマンド

① 関数一覧表

	関数名とプロトタイプ	機能
1	API int APIX DshSetSubstCmddName (int index, // ｺﾏﾝﾄﾞ ID index char *name // ｺﾏﾝﾄﾞ ID 文字列);	index 値に対応したｺﾏﾝﾄﾞ名を設定します。 index 値とﾃﾞﾌｫﾙﾄのｺﾏﾝﾄﾞについては、下の表を参照ください。
2	API int DshGetSubstCmdIndex (char *subst_cmd // ｺﾏﾝﾄﾞ名);	subst_cmd で指定された名前のｺﾏﾝﾄﾞに対するｲﾝﾃﾞｸｽ値を取得します。
3	API char * DshGetSubstCmdByIndex (int index // ｺﾏﾝﾄﾞｲﾝﾃﾞｸｽ);	index で指定されたｺﾏﾝﾄﾞを取得します。 ｺﾏﾝﾄﾞ名のﾎﾟｲﾝﾀで与えられます。

② ｺﾏﾝﾄﾞ ID インデクス値とデフォルトｺﾏﾝﾄﾞ ID 一覧表

ｺﾏﾝﾄﾞ ID ｲﾝﾃﾞｸｽ値	ｲﾝﾃﾞｸｽ用ﾏｸﾛ名	ﾃﾞﾌｫﾙﾄの文字列
0	SC_CancelSubstrate	"CancelSubstrate"
1	SC_ChangeSubstrateState	"ChangeSubstrateState"
2	SC_ProceedWithSubstrate	"ProceedWithSubstrate"
3	SC_UpdateSubstrateObject	"UpdateSubstrateObject"

３．24．３．４　S16F5 プロセスジョブ名関連

（1）プロセスジョブコマンド

① 関数一覧表

	関数名とプロトタイプ	機能
1	API int APIX DshSetPjCmdName (int index, // prj_cmd ｲﾝﾃﾞｸｽ char *name // prj_cmd 文字列);	index 値に対応したｺﾏﾝﾄﾞ名を設定します。 index 値とﾃﾞﾌｫﾙﾄのｺﾏﾝﾄﾞ名については、下の表を参照ください。
2	API int DshGetPrjCmdIndex (char *prj_cmd // Prjｺﾏﾝﾄﾞ名);	prj_cmd で指定された名前のｺﾏﾝﾄﾞに対するｲﾝﾃﾞｸｽ値を取得します。
3	API char * DshGetPrjCmdByIndex (int index //prj_cmd ｲﾝﾃﾞｸｽ);	index で指定されたｺﾏﾝﾄﾞ名を取得します。 ｺﾏﾝﾄﾞ名のﾎﾟｲﾝﾀで与えられます。

② ｺﾏﾝﾄﾞ ID インデクス値とデフォルトｺﾏﾝﾄﾞ ID 一覧表

ｺﾏﾝﾄﾞ ID ｲﾝﾃﾞｸｽ値	ｲﾝﾃﾞｸｽ用ﾏｸﾛ名	ﾃﾞﾌｫﾙﾄの文字列
0	PRC_ABORT	"ABORT"
1	PRC_STOP	"STOP"
2	PRC_CANCEL	"CANCEL"
3	PRC_PAUSE	"PAUSE"
4	PRC_RESUME	"RESUME"

３．25　その他のライブラリ関数

DSHEng4 が提供する各種ライブラリ関数です。DSHDR2 で使用する(SECS-II)メッセージ構造体内のメモリを開放する関数などが含まれます。

（1）関数一覧表

	ﾗｲﾌﾞﾗﾘ関数名	機能
1	DshGetSerialNo(	DSHEng4 の製品情報を取得します。
2	DshFreeMsg()	DSHDR2 通信ﾄﾞﾗｲﾊﾞが使用する DSHMSG ﾒｯｾｰｼﾞ情報構造体に使用されているﾒﾓﾘを DSHMSG 構造体を含めて開放します。
3	DshSetupMsgForSend()	送信用 DSHMSG 構造体に情報を設定します。
4	DshResponseSxFy()	２次応答ﾒｯｾｰｼﾞを送信します。
5	DshCreateThread()	ｽﾚｯﾄﾞを生成実行する。
6	DshCreateProcess()	ｺﾝｿｰﾙｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｸﾞﾗﾑを生成し実行する。
7	DshFormatDateTime()	現在日付時刻を書式付で編集し、文字列変数に格納する。

３．25．１　DSHEng4 製品情報取得関数

３．25．１．１　DshGetSerial_No () - 製品情報を取得する

（1）呼出書式

[c,C++]

```
API void APIX   DshGetSerialNo(
      char *buff                    // 製品情報 (ｼﾘｱﾙ番号) 格納ﾊﾞｯﾌｧ
);
```

[.NET VB]

```
Sub DshGetSerialNo (
       ByVal bb As String)
```

[.NET C#]

```
void DshGetSerialNo(
       byte[] bb );
```

（2）引数

buff

製品情報文字列を格納するためのバッファーポインタです。

（3）戻り値

なし。

（4）説明

現在使用している DSHEng4 の製品コードとシリアル番号を取得します。

例えば次のような製品情報を得ることができます。

"DshEngLib-2007-1 Product S/N : 1196753751-3"

３．25．２　SECS-II メッセージ送受信、データアイテム情報取得関連関数

３．25．２．１　DshSetupMsgForSend() - DSHMSG 構造体内に情報を設定する。

（1）呼出書式

[c,C++]

```
API void APIX DshSetupMsgForSend(
      DSHMSG *smsg,                  // 設定したい DSHMSG 構造体のﾎﾟｲﾝﾀ
      int    stream,                 // stream     Sx の x
      int    function,               // function   Fy の y
      BYTE   *buff,                  // text 用ﾊﾞｯﾌｧﾎﾟｲﾝﾀ
      int    buff_size               // text 用ﾊﾞｯﾌｧのﾊﾞｲﾄｻｲｽﾞ
);
```

[.NET VB]

```
Sub DshSetupMsgForSend (
        ByRef smsg As dshdr2.DSHMSG,
        ByVal s As Int32,
        ByVal f As Int32,
        ByVal wbit As Int32,
        ByRef buff As Byte,
        ByVal len As Int32)
```

[.NET C#]

```
void DshSetupMsgForSend(
        ref DSHMSG smsg,
        int s,
        int f,
        int wbit,
        byte[] buff,
        int len );
```

（2）引数

smsg

　　情報を設定したい DSHMSG 情報構造体のポインタです。

stream

　　smsg の stream メンバーに設定するストリームです。

function

　　smsg の function メンバーに設定するファンクションです。

buff

　　smsg の buffer メンバーに設定する text 用バッファポインタです。

buff_size

　　smsg の length メンバーに設定するバッファのバイトサイズです。

（3）戻り値

なし。

（4）説明

smsg が指すDSHMSG 構造体内に引数に与えられた情報を設定します。

text 用バッファが不要の際は、buff = NULL, size = 0 を設定してください。
なお、本関数は戻る前に、DSHDR2 のD_InitItemPut()関数を実行し、DSHMSG smsg の内容を初期設定します。
従って、text 情報が存在する場合は、本関数の後、ただちに D_PutItem()関数でﾃﾞｰﾀｱｲﾃﾑの設定処理に入ってください。

３．25．２．２　DshFreeMsg() - DSHMSG 構造体内のメモリを開放

（1）呼出書式

[c,C++]

```
API void APIX DshFreeMsg(
      DSHMSG *smsg                    // ﾒﾓﾘを開放したいDSHMSG構造体のﾎﾟｲﾝﾀ
);
```

[.NET VB]

```
Sub DshFreeMsg (
        ByRef smsg As dshdr2.DSHMSG)
```

[.NET C#]

```
void DshFreeMsg(
        ref DSHMSG smsg );
```

（2）引数

smsg

メモリを解放したいDSHMSG情報構造体のポインタです。

（3）戻り値

なし。

（4）説明

smsgが指すDSHMSG構造体内でTEXT格納用に使用されているメモリを解放し、DSHMSG構造体そのものも開放します。

３．25．２．３　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,                    // Poll 時に渡された TransactionID
        DSHMSG *smsg,                   // 送信したい２次 MSG の DSHMSG 構造体ﾎﾟｲﾝﾀ
        int (WINAPI *callback)()        // 送信依頼終了時の callback 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,
        dsh_api.CallbackSendResponse callback );
```

（2）引数

trid

１次メッセージを取得したときに渡される DSHDR2 通信ドライバーのトランザクション ID です。

smsg

情報を設定したい DSHMSG 情報構造体のポインタです。

callback

２次メッセージをデーモン(DSHENG)プロセスに渡した後、コールバックさせたい場合の CALLBACK 関数を指定します。CALLBACK させない場合は 0 を指定してください。

（3）戻り値

戻り値	意味
0	正常に要求できた。
(-1)	trid に対応する１次ﾒｯｾｰｼﾞのﾄﾗﾝｻﾞｸｼｮﾝ ID が見つからなかった。

（4）説明

デーモンから渡された１次メッセージに対する２次メッセージを応答するときに使用します。
応答メッセージは予め smsg メッセージ構造体に設定済みであることが条件です。

trid はデーモンの DSHDR2 通信ドライバーが１次メッセージを受信したときに発行したトランザクション ID です。この trid は EngGetSecsMsgReq() API 関数を使って１次メッセージを取得した際に与えられます。誤った trid を指定すると２次メッセージは送信されないで戻り値 が(-1)になります。

callback は、応答メッセージを IPC 通信でデーモンプロセスに渡した後、制御を戻してもらい,、何を送信したかを確認したい場合に指定します。確認が不要の場合は = 0 を指定してください。
callback 関数を指定した場合、callback 関数の呼出形式は次のようになります。

```
API int WINAPI callback( DSHMSG *prmsg )
```

ここで、prmsg は本関数内で smsg の内容をコピーしたものです。 smsg で与えられたメモリではありません。
callback 関数の処理が終わったら、DshFreeMsg(prmsg)を実行し、prmsg のメモリを開放してください。
callback = 0 の場合は、デーモンに渡すための待ち行列に情報を載せたあと、ただちに戻ってきます。

本関数から戻った後、smsg のメモリを開放する必要がある場合は、DshFreeMsg()関数を使って開放してください。

本関数はユーザ作成 DLL の中で使用されています。

３．25．３　プロセス、スレッド起動関数

３．25．３．１　DshCreateThread() - スレッドの生成と開始

（1）呼出書式

[c,C++]

```
API HANDLE  APIX DshCreateThread(
        LPTHREAD_START_ROUTINE th_func,        // 生成するｽﾚｯﾄﾞ関数アドレス
        int para,                      // 渡すﾊﾟﾗﾒｰﾀ
        LPDWORD thid                           // ｽﾚｯﾄﾞ ID 格納ﾎﾟｲﾝﾀ
);
```

[.NET VB]

N/A

[.NET C#]

N/A

（2）引数

th_func

新規スレッドの実行を起動する関数の開始アドレス

para

スレッド関数に引数とそて渡したいパラメータ

thid

生成されたスレッド ID を格納する領域のポインタ

（3）戻り値

戻り値	意味
(-1)以外	正常に生成できた。
(-1)	生成できなかった。

（4）説明

th_func で指定された関数をスレッドとして生成し実行を開始します。実行時に para を引数として th_func に渡します。正常に生成できた場合、生成したスレッドの ID を thid に格納します。
正常に生成実行できた場合は、(-1)以外のハンドルを返します。
失敗した場合は、(-1)を返します。
生成されたスレッドは、Windows によって呼出元プログラムとは独立した制御の下で処理することができます。(並列処理)
スレッドに関する詳しいことについては Windows の関連資料を参照してください。

（5）例

```
void sample()
 {
        HANDLE   h;
        DWORD    thid;
        h = DshCreateThread((LPTHREAD_START_ROUTINE)func_a, 345, &thid);
        if ( h==(-1) ){ printf( “thread creation error¥n” );
.
}
```

```
void     func_a( int para )
{
         printf( "para=%d¥n", para );
         .
         .
}
```

３．25．３．２　DshCreateProcess( ) - プロセスを生成する

（1）呼出書式

[c,C++]

```
API int   APIX DshCreateProcess(
        char *exe_file,              // 生成するﾌﾟﾛｾｽﾌﾟﾛｸﾞﾗﾑ名
        char *argv,                  // ﾌﾟﾛｾｽに渡す引数(文字列)
        char *title,                 // ﾌﾟﾛｾｽﾀｲﾄﾙ名
        char *error                  // 生成結果ｴﾗｰ情報格納用
);
```

[.NET VB]

N/A

[.NET C#]

N/A

（2）引数

exe_file

　　生成したいプロセスの実行形式プログラムファイル名(.EXE ファイル)

arg

　　生成したプロセスのメイン関数に渡す引数文字列

title

　　生成するプロセスのタイトル名

error

　　生成エラーが発生したときのエラー文字列を格納するバッファ

（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。

（4）説明

exe_file で指定されたコンソールアプリケーションプログラムをウインドウズプロセスとして生成し実行を開始します。(GUI プログラムには使用できません。)
生成されたコンソール画面は表示されません。

実行開始時に arv で指定された文字列を引数としてメイン関数に渡します。
title にはウインドウズに登録するプロセスタイトル名です。
正常に生成できた場合には、=0 を返却します。
生成に失敗した場合は、=(-1)が返却され、error にエラー情報が格納されます。

３．25．４　日付時刻編集関数

３．25．４．１　DshFormatDateTime () - 日付時刻の編集

（1）呼出書式

[c,C++]

```
API char* APIX DshFormatDateTime(
        char *buff,                          // 編集結果格納用ﾊﾞｯﾌｧ
        char *fmt                            // 編集ﾌｫｰﾏｯﾄ指定
);
```

[.NET VB]

```
Function DshFormatDateTime (
         ByVal tbuf As String,
         ByVal fmt As String) As Int32
```

[.NET C#]

```
void DshFormatDateTime(
         byte[] buff,
         byte[] fmt);
```

（2）引数

buff

編集結果文字列を格納するためのバッファーポインタ(ヌルで終る文字列)

fmt

編集フォーマット(書式)を文字列で指定します。

（3）戻り値

戻り値	意味
ﾎﾟｲﾝﾀ	引数１に与えられたﾊﾞｯﾌｧﾎﾟｲﾝﾀを返却します。

（4）説明

現在日付時刻を書式付で編集し、文字列変数に格納します。

fmt 内で書式用に使用する文字は Y,M,D,H,N,S,C を次表のように使用します。

書式文字	意味	出力書式
Y	年	1 文字が年１桁分を指定します。 YYYY は たとえば 2004 と編集されます。 4 文字未満の場合は、下位の桁から指定された文字分の出力を行います。
M	月	1 文字が月 1 桁分を指定します。
D	日	1 文字が日 1 桁分を指定します。
H	時	1 文字が時 1 桁分を指定します。
N	分	1 文字が分 1 桁分を指定します。
S	秒	1 文字が秒 1 桁分を指定します。
C	1/100 秒	1 文字が 0.01 秒単位の 1 桁を指定します。

（5）例

DshFormatDateTime(buff, “現在日付時刻は YYYY-MM-DD HH:NN:SS.CC です。”);

によって、buff に指定したバッファに次のような文字列を得ることができます。

“現在日付時刻は 2008-01-23 10:15:11.34 です。”