

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-3 / 1 5

3 . 4 . Limit 変数リミット情報関連関数

3 . 5 . TR 状態変数トレース情報アクセスサービス関数

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

3.4.	Limit 変数リミット情報関連関数.....	1
3.4.1	使用する情報格納構造体.....	4
3.4.2	Limit 変数リミット情報アクセスとリミット監視関連関数.....	6
3.4.2.1	EngSetMultiVLimit() - 複数の変数データリミット値設定関数.....	6
3.4.2.2	EngSetVLimit() - 変数データリミット値設定関数.....	8
3.4.2.3	EngGetVLimit() - 変数データリミット値取得関数.....	9
3.4.2.4	EngDelVLimit() - 変数データリミット値削除関数.....	10
3.4.2.5	EngCheckVLimit() - 変数データ値のリミット値チェック関数.....	11
3.4.2.6	EngSetupVLimitEventList() - 変数リミット監視結果リスト生成関数.....	12
3.4.2.7	EngTerminateVLimitEventList() - 変数リミット監視結果リスト終了関数.....	14
3.4.2.8	EngPutVLimitEventList() - 変数リミット監視結果設定関数.....	15
3.4.2.9	EngGetVLimitEventList() - 変数リミット監視結果取得関数.....	16
3.4.3	Limit 変数リミット関連ライブラリ関数.....	17
3.4.3.1	DshEncodeS2F45() - 変数リミット属性定義情報リストを S2F45 ヘンコード.....	17
3.4.3.2	DshDecodeS2F45() - S2F45 を変数リミット情報リストにデコード.....	19
3.4.3.3	DshDecodeS2F46() - S2F46 を変数リミット応答情報リストにデコード.....	21
3.4.3.4	DshFreeTLIMIT_LIST() - 変数リミット情報リスト構造体メモリの開放.....	22
3.4.3.5	DshFreeTLIMIT_INFO() - 変数リミット情報構造体メモリの開放.....	23
3.4.3.6	LIMIT_LIST() - 変数リミット情報リスト構造体メモリのコピー.....	24
3.4.3.7	DshCopyTLIMIT_INFO() - 変数リミット情報構造体メモリのコピー.....	25
3.4.3.8	DshInitTLIMIT_LIST - 変数リミット属性定義情報 TLIMIT_LIST の初期設定.....	26
3.4.3.9	DshAddTLIMIT_LIST() - 変数リミット属性定義情報の追加.....	27
3.4.3.10	DshInitTLIMIT_INFO - 変数リミット属性定義情報 TLIMIT_INFO の初期設定.....	28
3.4.3.11	DshPutTLIMIT_INFO() - 変数リミット属性定義データの追加.....	30
3.4.3.12	DshInitTLIMIT_ERR_LIST() - 変数リミット応答情報リストの初期化.....	32
3.4.3.13	DshPutTLIMIT_ERR_LIST() - 変数リミット応答情報の設定.....	33
3.4.3.14	DshFreeTLIMIT_ERR_LIST() - S2F46 応答情報リスト構造体メモリの開放.....	34
3.4.3.15	DshInitTLIMIT_ERR_INFO() - 変数リミット応答情報の初期化.....	35
3.4.3.16	DshPutTLIMIT_ERR_ID() - 変数リミット応答情報の設定.....	36
3.4.3.17	DshFreeTLIMIT_ERR_INFO() - S2F46 応答情報構造体メモリの開放.....	37
3.4.3.18	DshMakeS2F45Response() - S2F45 の応答メッセージの生成.....	38
3.4.3.19	DshDecodeS2F48() - S2F48 を変数リミット属性情報リストにデコード.....	40
3.4.3.20	DshFreeTLIMIT_RSP_LIST() - 変数リミット応答情報構造体メモリの開放.....	42
3.4.3.21	DshFreeTVLIMIT_EVENT_INFO() - リミット監視結果情報構造体メモリ開放.....	43
3.5.	TR 状態変数トレース情報アクセスサービス関数.....	44
3.5.1	使用する情報格納構造体.....	46
3.5.2	TRACE トレース情報アクセス関数.....	47
3.5.2.1	EngAllocTrInfo() - トレース情報の登録.....	47
3.5.2.2	EngSetTrInfo() - トレース情報の設定.....	48
	EngSetTrInfoX() - インデクス指定によるトレース情報の設定.....	48
3.5.2.3	EngGetTrInfo() - トレース情報の取得.....	50
	EngGetTrInfoX() - インデクス指定でのトレース情報の取得.....	50
3.5.2.4	EngDelTrInfo() - トレースの削除.....	52
	EngDelTrInfoX() - インデクスでのトレースの削除.....	52
3.5.2.5	EngEnableTrace() - トレース有効状態の設定.....	53
	EngEnableTraceX() - インデクスでのトレース有効状態の設定.....	53

3.5.2.6	EngGetTrList() - 全登録トレース ID 取得関数.....	54
3.5.2.7	EngGetTrId() - インデクスから TRACEID (トレース ID) の取得.....	55
3.5.2.8	EngGetTrIdIndex() -TRACEID (トレース ID) からインデクスの取得	56
3.5.3	TR トレース関連ライブラリ関数.....	57
3.5.3.1	DshDecodeS2F23() - S2F23 をトレース情報にデコード.....	57
3.5.3.2	DshFreeTTRACE_INFO() - トレース情報構造体メモリの開放.....	58
3.5.3.3	DshCopyTTRACE_INFO() - トレース情報構造体メモリのコピー	59
3.5.3.4	DshInitTTRACE_INFO() - トレース S2F43 情報の初期設定	60
3.5.3.5	DshPutTTRACE_INFO() - トレース S2F23 情報の SVID 設定.....	62
3.5.3.6	DshMakeS2F23Response() - S2F23 の応答メッセージの生成	63
3.5.4	ユーザ作成ライブラリ関数	64
3.5.4.1	DshResponseS2F24() - S2F24 トレース設定応答メッセージ	64

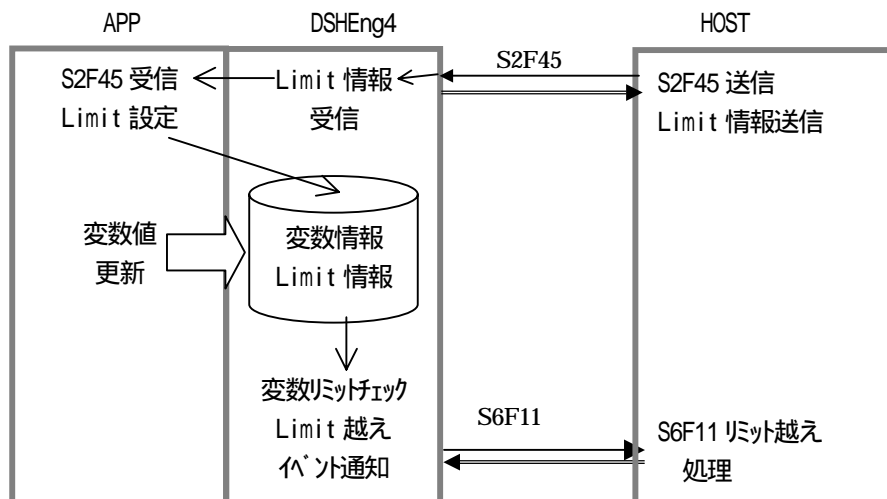
(VOL - 4 に続く)

3.4. Limit 変数リミット情報関連関数

変数については、3.3で述べましたが、ここでは変数リミットに関連する関数について説明します。

変数リミット情報は、DSHEng4 が管理します。従って、APP はこれらの情報をアクセスするために以下の DSHEng4 ライブラリ関数を使用します。また、DSHEng4 は変数値のリミット監視機能をおこなうための関数についても説明します。

変数リミット情報は、DSHEng4 が管理しています。



(1) 情報アクセスとリミット監視情報関連 API 関数

変数リミット情報アクセス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	EngSetMultiVLimit()	S2F45 メッセージの内容で複数の変数リミット値を設定します。
2	EngSetVLimit()	1 個の変数リミット値を設定します。
3	EngGetVLimit()	変数リミット値を取得します。
4	EngDelVLimit()	変数リミット値を削除します。
5	EngCheckVLimit()	変数値のリミットチェックをします。
6	EngSetupVLimitEventList()	装置側 APP での変数リミットイベント受信用リストとイベントの準備をします。
7	EngTerminateVLimitEventList()	装置側 APP での変数リミットイベント受信用リストとイベントを終了します。
8	EngPutVLimitEventList()	DSHEng4 が APP での変数リミット監視イベントを通知します。
9	EngGetVLimitEventList()	装置側 APP が変数リミット監視イベントを受信用リストから取得します。

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

(2) ライブラリ関数

他に APP が使用できる変数リミット情報処理用 API 関数として、以下の関数があります。

	API 関数名	機能
1	DshEncodeS2F45()	TLIMIT_LIST 構造体の変数リミット属性情報を S2F45 メッセージにエンコードするための関数です。
2	DshDecodeS2F45()	S2F45 のメッセージ内変数リミット情報を TLIMIT_LIST 構造体にデコード収納するための関数です。
3	DshDecodeS2F46()	S2F46 のメッセージ内の変数リミット属性応答情報を TLIMIT_ERR_LIST 構造体にデコード収納するための関数です。
4	DshFreeTLIMIT_LIST()	変数リミット情報が格納されている TLIMIT_LIST 構造体と内部で使用されている全メモリを開放するための関数です。
5	DshFreeTLIMIT_INFO()	1 個の変数リミット情報が格納されている TLIMIT_INFO 構造体の内部で使用されている全メモリを開放するための関数です。
6	DshCopyTLIMIT_LIST()	TLIMIT_LIST の変数リミット情報を別の構造体にコピーします。
7	DshCopyTLIMIT_INFO()	TLIMIT_INFO、変数リミット情報構造体メモリの内容を別の構造体にコピーします。
8	DshInitTLIMIT_LIST()	TLIMIT_LIST 変数リミット属性定義リスト格納用構造体の初期設定を行います。(TLIMIT_LIST は複数の TLIMIT_INFO を格納するための構造体)
9	DshAddTLIMIT_LIST()	TLIMIT_LIST 変数リミット属性定義リスト格納用構造体内に TLIMIT_INFO 構造体に格納されている変数リミット情報を加えます。
10	DshInitTLIMIT_INFO()	1 個の変数のための TLIMIT_INFO 変数リミット属性定義リスト格納用構造体内の初期設定を行います。
11	DshPutTLIMIT_INFO()	TLIMIT_INFO 構造体に 1 個のリミット ID の上下限データを TLIMIT_INFO 構造体内に加えます。
12	DshInitTLIMIT_ERR_LIST()	S2F45 の応答メッセージ生成に使用する TLIMIT_ERR_LIST 応答情報構造体の初期設定を行います。このリストは TLIMIT_ERR_INFO 用格納リストです。
13	DshPutTLIMIT_ERR_LIST(S2F45 の応答メッセージ生成に使用する TLIMIT_ERR_LIST 応答情報構造体に 1 個の TLIMIT_ERR_INFO を設定をします。
14	DshFreeTLIMIT_ERR_LIST()	変数リミット情報が格納されている TLIMIT_LIST 構造体内部で使用されている全メモリを開放するための関数です。
15	DshInitTLIMIT_ERR_INFO()	S2F45 の応答メッセージ生成に使用する TLIMIT_ERR_INFO 応答情報構造体の初期設定を行います。
16	DshPutTLIMIT_ERR_ID()	S2F45 の応答メッセージ生成に使用する TLIMIT_ERR_INFO 応答情報構造体にリミット ID を設定します。
17	DshFreeTLIMIT_ERR_INFO()	1 個の変数リミット情報が格納されている TLIMIT_INFO 構造体の内部で使用されている全メモリを開放するための関数です。
18	DshMakeS2F45Response()	TLIMIT_LIST, TLIMIT_ERR_LIST から S2F46 メッセージを生成します。
19	DshEncodeVidList()	変数 ID リストに含まれている VID を DSHMSG 構造体の buffer にエンコードします。3.3.6.6 を参照ください。
20	DshDecodeS2F48()	S2F48 メッセージに含まれる変数リミット情報を TLIMIT_RSP_LIST 構造体にデコード収納するための関数です。
21	DshFreeTLIMIT_RSP_INFO()	S2F47 の応答メッセージ S2F48 のデコード、格納した情報構造体内の内部で使用されている全メモリを開放するための関数です。

22	DshFreeTVLIMIT_EVENT_INFO	TVLIMIT_EVENT_INFO 構造体内に使用されているメモリを開放します。
----	---------------------------	---

3.4.1 使用する情報格納構造体

変数リミット属性定義情報を操作する関数は、情報格納のための TLIMIT_LIST などの構造体を使用します。TLIMIT_LIST とのその内部で使用する他の構造体は下記のとおりです。

(1) TLIMIT_LIST - Limit Information List

```
typedef struct{
    int      vid_count;
    TLIMIT_INFO **limit_list;
} TLIMIT_LIST;
```

vid_count = 0 の場合、変数リミット情報が1つも無いことを意味します。

vid_count > 0 の場合、limit_list[] 内に変数リミット情報が格納されている TLIMIT_INFO のポインタが格納されています。

(2) TLIMIT_INFO - Limit Information

```
typedef struct{
    TDVID      vid;
    int      limit_count;
    TLIMIT_ID_INFO **limitid_list;
    int      format;           // for upperdb & lowerdb
    int      asize;           // " " "
} TLIMIT_INFO;
```

vid は変数リミットを設定する変数 ID です。

limit_count は、vid に対し何個のリミット値があるかを示す個数です。

limitid_list[] には limit_count の数だけのリミット ID と上下限値が格納されている TLIMIT_ID_INFO へのポインタが格納されます

format は上下限値のデータタイプです。

asize は上下限データの配列サイズです。

(3) TLIMIT_ID_INFO Limit ID Information

```
typedef struct{
    TLIMITID      limit_id;
    void          *upperdb;           // 上限値
    void          *lowerdb;          // 下限値
} TLIMIT_ID_INFO;
```

limit_id は vid に対するリミット ID です。

upperdb は上限値が格納されている領域のポインタです。

lowerdb は下限値が格納されている領域のポインタです。

(4) TLIMIT_ERR_LIST - S2F46 応答情報格納用構造体

```
typedef struct{
    int      vlaack;           // S2F46 の VLAACK
    int      err_count;       // 含まれていたエラー情報の数
    TLIMIT_ERR_INFO **limit_list; // エラー情報のリスト
} TLIMIT_ERR_LIST;
```


(5) TLIMIT_ERR_INFO – S2F46 に含まれる 1 個のエラー情報格納構造体

```
typedef struct{
    TVID      vid;           // 変数 ID
    int       lvack;        // 変数 ID のリミット情報に対する LVACK
    int       lmt_count;    // リミット ID の数(limit_idリスト内の)
    TLIMITID  *limit_id;    // リミット ID 格納リスト
    int       *limitack;    // limit_id 格納リスト内のリミット ID に対する ack
} TLIMIT_ERR_INFO;
```

(6) TLIMIT_RSP_LIST – S2F48 に含まれるリミット情報リスト格納構造体

```
typedef struct{
    int       vid_count;    //
    TLIMIT_RSP_INFO **limit_list;
} TLIMIT_RSP_LIST;
```

(7) TLIMIT_RSP_INFO – S2F48 応答情報内のリミット情報格納構造体

```
typedef struct{
    TVID      vid;           // 変数 ID
    char      *units;       // 単位名
    int       format;       // for upperdb & lowerdb
    int       asize;        // " " "
    void      *limit_min;   // 最小リミット値
    void      *limit_max;   // 最大リミット値
    int       limit_count;  // リミット ID の数
    TLIMIT_ID_INFO **limitid_list; // リミット ID 情報リストのポインタ
} TLIMIT_RSP_INFO;
```

(8) TVLIMIT_EVENT_INFO – リミット監視結果情報格納構造体

```
typedef struct{
    TVID      vid;           // vid
    char      *value;       // value in ascii
    int       limitid;      // limit id
    int       dir;          // transient direction (0=up, 1=down)
}TVLIMIT_EVENT_INFO;
```

3.4.2 Limit 変数リミット情報アクセスとリミット監視関連関数

3.4.2.1 EngSetMultiVLimit() - 複数の変数データリミット値設定関数

(1) 呼出書式

[c,C++]

```
API int APIX EngSetMultiVLimit(
    TLIMIT_LIST *Imlist          // LIMIT 情報リスト構造体ポインタ
);
```

[.NET VB]

```
Function EngSetMultiVLimit (
    ByRef Imlist As dsh_info.TLIMIT_LIST) As Int32
```

[.NET C#]

```
int EngSetMultiVLimit(
    ref TLIMIT_LIST Imlist );
```

(2) 引数

Imlist

例えば S2F45 メッセージをエンコードするとき使用する LIMIT 情報リスト格納構造体のポインタです。

エンコードは、DshEncodeS2F45()関数を使用して行います。

APP が、自身で Limit 情報を Imlist に設定することもできます。

(3) 戻り値

戻り値	意味
0	正常に設定できた。
(-1)	Imlist の内容が正しくなかったので設定できなかった。

(4) 説明

Imlist のリミット情報を DSHEng4 が管理する装置変数情報内に登録設定します。

もし、Imlist 内のメンバー vid_count が=0 の場合は、DSHEng4 内に登録されている全 VID のリミット値を無効にします。

正常に設定できた場合は、関数の戻り値は 0 になります。

Imlist に含まれるリミット情報の内容が 1 つでも正しくなかった場合、Limit 値の設定は行われません。

APP 側で TLIMIT_LIST 構造体領域を設け、設定したいリミット情報をその中にセットした上で、本関数を使って DSHEng4 に設定登録することができます。

(5) 補足事項

Imlist が指定する TLIMIT_LIST 構造体内には 設定したい vid 数と各 vid に対する以下のリミット情報が格納されていなければなりません。

[TLIMIT_LIST 内の情報]

```
vid_counVID -1 --- limit_count1
                                limitid11, upperdb11, lowerdb11 (0個以上のリスト)
                                limitid12, upperdb12, lowerdb12
```

```

vid-2 --- limit_count2
          limitid21, upperdb21, lowerdb21

```

vid_count =0 の場合は、全VID に対するリミット値を無効にします。
 各 vid について、limit_count の数が=0 の場合は、その vid についてはリミット値が無効となります。

DSHEng4 との関連と全体の処理順序は次のようになります。矢印の方向に処理が行われます。

APP プログラム	DSHEng4.DLL		ホスト
<p>APP は EngGetSecsMsgReq()関数 で S2F45 を受け取る。</p> <p>以下取得した S2F45 を処理 関数による処理を行う。 DshDecodeS2F45() で Limit 情報をデコードする。(lmtlist に) APP は lmtlist 内の Limit 情報を評価する。 その結果を lmtlist 内の Ack 情報として設定する。 評価結果が正常の場合、DshRegisterLimit()で DSHGEM-LIB に登録する。</p> <p>戻り値が結果 DshResponseS2F46()で S2F46 メッセージを応答する。</p>	<p>← 受信MSGキューに登録 関数を呼出す、</p> <p>Limit 情報を保存</p>		<p>S2F45 送信 リミット情報</p> <p>S2F46 を受信する。</p>

3.4.2.2 EngSetVLimit() - 変数データリミット値設定関数

(1) 呼出書式

[C, C++]

```
API int APIX EngSetVLimit(
    TVID vid,                // 変数 ID
    TLIMIT_INFO *lminfo     // リミット情報格納構造体ポインタ
);
```

[.NET VB]

```
Function EngSetVLimit (
    ByVal vid As Int32,
    ByRef lminfo As dsh_info.TLIMIT_INFO) As Int32
```

[.NET C#]

```
int EngSetVLimit(
    uint vid,
    ref TLIMIT_INFO lminfo );
```

(2) 引数

vid

リミット値を設定する装置変数 ID です。
装置管理情報定義ファイルに登録されている装置変数の ID でなければなりません。

lminfo

設定したいリミット値が格納されている構造体の格納ポインタです。

(3) 戻り値

戻り値	意味
0	正常に設定された。
(-1)	vid の値が正しくなかった。(登録されていなかった)

(4) 説明

vid で指定された装置変数に対し新しいリミット情報を設定します。

戻り値が設定結果になります。

正常に設定できた場合は、関数の戻り値は 0 になります。

指定された vid が登録されていない場合は(-1)が戻ります。

[TLIMIT_INFO 内の情報]

```
vid-1 --- limit_count
        limitid1, upperdb1, lowerdb1 (0 個以上のリスト)
        limitid2, upperdb2, lowerdb2
```

limit_count = 0 の場合は、指定 vid に対し、リミット情報を無効にします。

3.4.2.3 EngGetVLimit() - 変数データリミット値取得関数

(1) 呼出書式

[C, C++]

```
API int APIX EngGetVLimit(
    TVID vid,                // 変数 ID
    TLIMIT_INFO *lminfo     // リミット情報格納構造体ポインタ
);
```

[.NET VB]

```
Function EngGetVLimit (
    ByVal vid As Int32,
    ByRef lminfo As dsh_info.TLIMIT_INFO) As Int32
```

[.NET C#]

```
int EngGetVLimit(
    uint vid,
    ref TLIMIT_INFO lminfo );
```

(2) 引数

vid

リミット値を取得する装置変数 ID です。
装置管理情報定義ファイルに登録されている装置変数の ID でなければなりません。

lminfo

取得したいリミット値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	正常に取得できた。
(-1)	vid の値が正しくなかった。(登録されていなかった)

(4) 説明

vid で指定された装置変数のリミット情報を DSHEng4 から取得します。

戻り値が取得結果になります。

正常に取得できた場合は、関数の戻り値は 0 になります。

指定された vid が登録されていない場合は(-1)が戻ります。

[TLIMIT_INFO 内の情報]

```
vid-1 --- limit_count
        limitid1, upperdb1, lowerdb1 (0 個以上のリスト)
        limitid2, upperdb2, lowerdb2
        .
        .
```

limit_count = 0 の場合は、指定 vid に対し、リミット情報がないことを意味します。

3.4.2.4 EngDelVLimit() - 変数データリミット値削除関数

(1) 呼出書式

[C, C++]

```
API int APIX EngDelVLimit(
    TVID vid // 変数 ID
);
```

[.NET VB]

```
Function EngDelVLimit (
    ByVal vid As Int32) As Int32
```

[.NET C#]

```
int EngDelVLimit(
    uint vid );
```

(2) 引数

vid

削除したいリミット値の装置変数 ID です。
装置管理情報定義ファイルに登録されている装置変数の ID でなければなりません。

(3) 戻り値

戻り値	意味
0	正常に削除できた。
(-1)	vid の値が正しくなかった。(登録されていなかった)

(4) 説明

vid で指定された装置変数のリミット情報を削除します。
戻り値が削除結果になります。
正常に削除できた場合は、関数の戻り値は 0 になります。
指定された vid が登録されていない場合は(-1)が戻ります。

3.4.2.5 EngCheckVLimit() - 変数データ値のリミット値チェック関数

(1) 呼出書式

[C, C++]

```
API int APIX EngCheckVLimit(
    TVID vid,                // 変数 ID
    TLIMITID limitid,       // リミット ID
    void *value              // チェック変数データ値格納ポインタ
);
```

[.NET VB]

```
Function EngCheckVLimit (
    ByVal vid As Int32,
    ByVal limitid As Int32,
    ByVal value As Int32) As Int32
```

[.NET C#]

```
int EngCheckVLimit(
    uint vid,
    uint limitid,
    byte[] value );
```

(2) 引数

vid

チェックしたいリミット値の装置変数 ID です。
装置管理情報定義ファイルに登録されている装置変数の ID でなければなりません。

limitid

指定変数データ情報内のチェックするリミットの limitid を指定する。

value

チェックしたい変数データ値が格納されている領域のポインタです。
データタイプは、vid によって決まりますので value の値もそれに合わせておかなければなりません。

(3) 戻り値

戻り値	意味
0	value がリミット値内の範囲内であった。
(-1)	vid の値が正しくなかった。(登録されていなかった)
(-2)	lowerdb 値よりも小さかった。
(-3)	upperdb 値よりも大きかった。

(4) 説明

装置変数の値 value を、vid の装置変数の limitid で指定されたリミット値(upperdb, lowerdb)の範囲内の値であるかどうかをチェックします。戻り値がチェック結果になります。
指定された vid が登録されていない場合は(-1)が戻ります。

3.4.2.6 EngSetupVLimitEventList() - 変数リミット監視結果リスト生成関数

(1) 呼出書式

[C, C++]

```
API int APIX EngSetupVLimitEventList(
    HANDLE event // 通知イベント
);
```

[.NET VB]

```
Function EngSetupVLimitEventList (
    ByVal evt As Int32) As Int32
```

[.NET C#]

```
int EngSetupVLimitEventList(
    uint evt );
```

(2) 引数

event

リミット監視において領域の遷移が起こった場合に DSHeng4 から APP に通知するイベントのハンドルです。

CreateHandle()Windows 関数で生成したものを指定します。ただし、イベント通知を望まない場合は、NULL を指定してください。

(3) 戻り値

戻り値	意味
0	正常に設定された。
(-1)	装置 ID が正しくなかった。

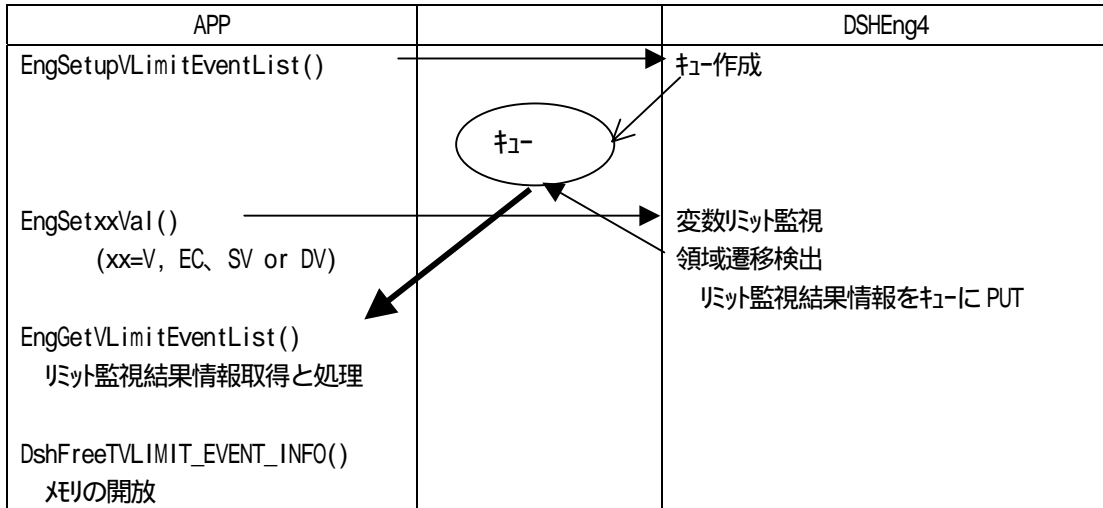
(4) 説明

DSHeng4 に対し、変数リミット監視によってデッドバンドを抜け、別の領域に遷移した場合、検出した対象の変数 ID、変数値、リミット ID ならびに遷移方向の情報 (リミット監視結果情報) をキューと通して取得できるように専用キューを作成することを要求します。

event の値が有効 (NULL でない) である場合は、領域遷移を検出した際、APP に対し event 通知するためイベントハンドルを指定します。

DSHeng4 は最大 32 個の情報を溜めることができるキューを作成します。

作成したキューからの監視結果情報の取得は、後述する EngGetVLimitEventList() 関数を使って行います。監視結果情報が格納されている TVLIMIT_EVENT_INFO 構造体領域のポインタを得ることができます。



3.4.2.7 EngTerminateVLimitEventList () -変数リミット監視結果リスト終了関数

(1) 呼出書式

[C, C++]

```
API void APIX EngTerminateVLimitEventList();
```

[.NET VB]

```
sub EngTerminateVLimitEventList( )
```

[.NET C#]

```
void EngTerminateVLimitEventList();
```

(2) 引数

なし。

(3) 戻り値

なし。

(4) 説明

3.4.2.6の EngSetupVLimitEventList()関数で生成したキューの削除と指定されたイベントの開放を行うための関数です。

DSHEng4 は、EngTerminateEquipment()関数が実行されたときに、本関数と同じ処理を行います。従って、APP が明示的に行う必要がある場合にのみ使用してください。

3.4.2.8 EngPutVLimitEventList () - 変数リミット監視結果設定関数

(1) 呼出書式

[C, C++]

```
EngPutVLimitEventList(
    TVLIMIT_EVENT_INFO *info          // 監視結果格納構造体のポインタ
);
```

[.NET VB]

```
Function EngPutVLimitEventList (
    ByRef info As dsh_info.TVLIMIT_EVENT_INFO) As Int32
```

[.NET C#]

```
int EngPutVLimitEventList(
    ref TVLIMIT_EVENT_INFO info );
```

(2) 引数

info

DSHEng4 がリミット監視結果情報をリストに加えるリミット結果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	正常に設定できた。
(-1)	装置 ID が正しくない。
(-2)	リストが満杯であったので設定できなかった。

(4) 説明

info で指定された構造体内のリミット監視結果情報をリストに加えます。

戻り値=0 の場合、情報を設定できたことを意味します。

通常、本関数は DSHEng4 が内部で使用します。

3.4.2.9 EngGetVLimitEventList () - 変数リミット監視結果取得関数

(1) 呼出書式

[C, C++]

```
EngGetVLimitEventList(
    TVLIMIT_EVENT_INFO *info // 監視結果格納構造体のポインタ
);
```

[.NET VB]

```
Function EngGetVLimitEventList (
    ByRef info As dsh_info.TVLIMIT_EVENT_INFO) As Int32
```

[.NET C#]

```
int EngGetVLimitEventList(
    ref TVLIMIT_EVENT_INFO info );
```

(2) 引数

info

リミット監視結果情報をリストから取得し、その内容を格納するための構造体のポインタです。

(3) 戻り値

戻り値	意味
0	リストに情報が無かった。
>0	取得できた。
(-1)	装置 ID が正しくない。

(4) 説明

APP が変数リミット監視結果情報がリストに残っているかどうかを調べ、残っていれば、1個分の情報を info で指定した構造体内に取得します。

戻り値が正の場合、情報を取得できたことを意味します。

取得した info 情報の処理が終了した後は、info 内に割当てられたメモリを開放するために次の関数を実行してください。

```
DshFreeTVLIMIT_EVENT_INFO( info );
```

TVLIMIT_EVENT_INFO 内には、変数値が value メンバーで与えられますが、この値は、数値を文字列表現したのになります。

3.4.3 Limit 変数リミット関連ライブラリ関数

3.4.3.1 DshEncodeS2F45() - 変数リミット属性定義情報リストを S2F45 へエンコード

(1) 呼出書式

[C, C++]

```
API int APIX DshEncodeS2F45(
    DSHMSG *smsg,           // SECS メッセージ 情報構造体のポインタ
    BYTE *buffer,          // S2F45 を格納するバッファポインタ
    int buflen,            // buffer のバイトサイズ
    TLIMIT_LIST *pinfo     // エンコードしたい変数リミット情報リスト構造体のポインタ
);
```

[.NET VB]

```
Function DshEncodeS2F45 (
    ByRef smsg As dshdr2.DSHMSG,
    ByRef buff As Byte,
    ByVal buflen As Int32,
    ByRef info As dsh_info.TLIMIT_LIST) As Int32
```

[.NET C#]

```
int DshEncodeS2F45(
    ref DSHMSG smsg,
    byte[] buff,
    int buflen,
    ref TLIMIT_LIST info );
```

(2) 引数

msg

エンコードした S2F45 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 S2F45 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

エンコードしたい変数リミット属性定義情報リストが格納されている構造体のポインタです。

(3) 戻り値

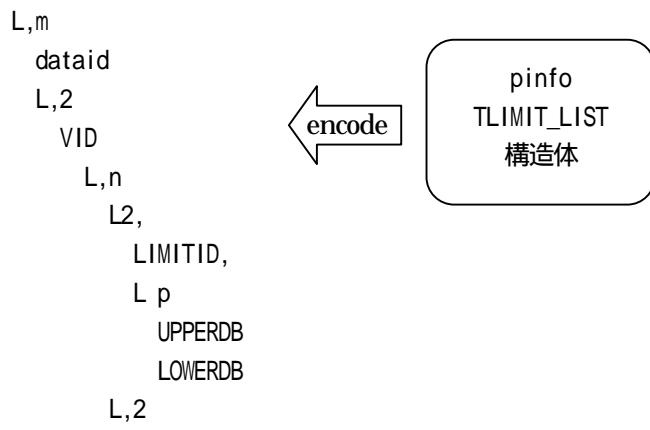
戻り値	意味
0	正常にエンコードできた。
(-1)	msg を正しくエンコードできなかった。

(4) 説明

TLIMIT_LIST 情報リスト構造体に格納されている変数リミット情報を、S2F45 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 S2F45



3.4.3.2 DshDecodeS2F45() - S2F45 を変数リミット情報リストにデコード

(1) 呼出書式

[C, C++]

```
API int APIX DshDecodeS2F45(
    DSHMSG *msg, // SECS メッセージ 情報構造体のポインタ
    TLIMIT_LIST *pinfo // デコードした情報リストを格納する構造体のポインタ
);
```

[.NET VB]

```
Function DshDecodeS2F45 (
    ByRef msg As dshdr2.DSHMSG,
    ByRef info As dsh_info.TLIMIT_LIST) As Int32
```

[.NET C#]

```
int DshDecodeS2F45(
    ref DSHMSG msg,
    ref TLIMIT_LIST info );
```

(2) 引数

msg

S2F45 の SECS メッセージ 情報構造体のポインタです。

pinfo

デコードした結果の変数リミット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

S2F45 メッセージに含まれる変数リミット情報リストを、ユーザプログラムが処理しやすい TLIMIT_LIST 構造体の中にデコードします。

変数リミット構造体メンバーのために使用するメモリは全て DSHEngLIB 側が準備します。

従って、**情報処理終了後は、pinfo のメモリを DshFreeTLIMIT_LIST() 関数を使って解放してください。**

msg S2F45

L,m

dataid

L,2

VID

L,n

L,2,

LIMITID,

L p

UPPERDB

LOWERDB

L,2



(5) 関連関数

- DshEncodeLimitMSG() : TLIMIT_LIST の内容を S2F45 メッセージにエンコードします。
- DshFreeTLIMIT_LIST() : TLIMIT_LIST のメモリを解放します。
- DshCopyTLIMIT_LIST() : TLIMIT_LIST の内容を別の構造体にコピーします。
- DshMakeS2F45Response() : TLIMIT_LIST の ACK 情報から S2F46 応答メッセージを作成します。

3.4.3.3 DshDecodeS2F46() - S2F46 を変数リミット応答情報リストにデコード

(1) 呼出書式

[C, C++]

```
API int APIX DshDecodeS2F46(
    DSHMSG *msg, // SECS メッセージ 情報構造体のポインタ
    TLIMIT_ERR_LIST *pinfo // デコードした応答情報リストを格納する構造体のポインタ
);
```

[.NET VB]

```
Function DshDecodeS2F48 (
    ByRef msg As dshdr2.DSHMSG,
    ByRef list As dsh_info.TLIMIT_RSP_LIST) As Int32
```

[.NET C#]

```
int DshDecodeS2F46(
    ref DSHMSG msg,
    ref TLIMIT_ERR_LIST erlist);
```

(2) 引数

msg

S2F46 の SECS メッセージ 情報構造体のポインタです。

pinfo

デコードした結果の変数リミット応答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

S2F46 メッセージに含まれる変数リミット応答情報リストを、ユーザプログラムが処理しやすい TLIMIT_ERR_LIST 構造体の中にデコードします。TLIMIT_ERR_LIST 内に複数の TLIMIT_ERR_INFO 構造体の情報が含まれます。

変数リミット構造体メンバーのために使用するメモリは全て DSHEng4 側が準備します。

従って、情報処理終了後は、pinfo のメモリを DshFreeTLIMIT_ERR_LIST()関数を使って解放してください。

msg S2F46

L,2

vlaack

L,m

L,3

VID

LVACK

L,n

LIMITID,

limitack

decode



処理



3.4.3.4 DshFreeTLIMIT_LIST() - 変数リミット情報リスト構造体メモリの開放

(1) 呼出書式

[C, C++]

```
API void APIX DshFreeTLIMIT_LIST(  
    TLIMIT_LIST *pinfo // メモリを開放したい変数リミット情報リスト構造体の格納ポインタ  
);
```

[.NET VB]

```
Sub DshFreeTLIMIT_LIST (  
    ByRef info As dsh_info.TLIMIT_LIST)
```

[.NET C#]

```
void DshFreeTLIMIT_LIST(  
    ref TLIMIT_LIST info );
```

(2) 引数

pinfo

メモリを解放したい変数リミット情報リスト構造体のポインタです。

(3) 戻り値

なし。

(4) 説明

TLIMIT_LIST 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。

3.4.3.5 DshFreeTLIMIT_INFO() - 変数リミット情報構造体メモリの開放

(1) 呼出書式

[C, C++]

```
API void APIX DshFreeTLIMIT_INFO(  
    TLIMIT_INFO *pinfo // メモリを開放したい変数リミット情報リスト構造体の格納ポインタ  
);
```

[.NET VB]

```
Sub DshFreeTLIMIT_INFO (  
    ByRef info As dsh_info.TLIMIT_INFO)
```

[.NET C#]

```
void DshFreeTLIMIT_INFO(  
    ref TLIMIT_INFO info );
```

(2) 引数

pinfo

メモリを解放したい変数リミット情報構造体のポインタです。

(3) 戻り値

なし。

(4) 説明

TLIMIT_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。

3.4.3.6 LIMIT_LIST() - 変数リミット情報リスト構造体メモリのコピー

(1) 呼出書式

[C, C++]

```
int DshCopyTLIMIT_LIST(
    TLIMIT_LIST *dinfo,           // 北°-先のポインタ
    TLIMIT_LIST *sinfo           // 北°-元のポインタ
);
```

[.NET VB]

```
Function DshCopyTLIMIT_LIST (
    ByRef pinfo As dsh_info.TLIMIT_LIST,
    ByRef sinfo As dsh_info.TLIMIT_LIST) As Int32
```

[.NET C#]

```
int DshCopyTLIMIT_LIST(
    ref TLIMIT_LIST pinfo,
    ref TLIMIT_LIST sinfo );
```

(2) 引数

dinfo

コピー先の変数リミット情報リスト構造体のポインタです。

sinfo

コピー元の変数リミット情報リスト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

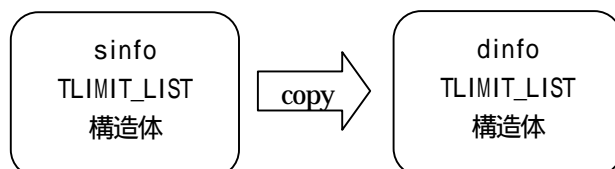
戻り値	意味
0	正常に北°-できた。
(-1)	sinfo または dinfo の値が NULL だったので北°-できなかった。

(4) 説明

sinfo が指す TLIMIT_LIST 構造体内に格納されている変数リミット情報リストの内容を dinfo で指定されたメモリの TLIMIT_LIST 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、DshFreeTLIMIT_LIST()関数を使って開放してください。



3.4.3.7 DshCopyTLIMIT_INFO() - 変数リミット情報構造体メモリのコピー

(1) 呼出書式

[C, C++]

```
int DshCopyTLIMIT_INFO(
    TLIMIT_INFO *dinfo,           // 北°-先のポインタ
    TLIMIT_INFO *sinfo           // 北°-元のポインタ
);
```

[.NET VB]

```
Function DshCopyTLIMIT_INFO (
    ByRef pinfo As dsh_info.TLIMIT_INFO,
    ByRef sinfo As dsh_info.TLIMIT_INFO) As Int32
```

[.NET C#]

```
int DshCopyTLIMIT_INFO(
    ref TLIMIT_INFO pinfo,
    ref TLIMIT_INFO sinfo );
```

(2) 引数

dinfo

変数リミット情報構造体のコピー先構造体メモリのポインタです。

sinfo

コピー元の変数リミット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

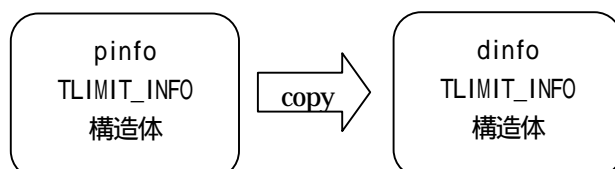
戻り値	意味
0	正常に北°-できた。
(-1)	sinfo または dinfo の値が NULL だったので北°-できなかった。

(4) 説明

sinfo が指す TLIMIT_INFO 構造体内に格納されている変数リミット情報の内容を dinfo で指定された構造体メモリ内にコピーします。

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、DshFreeTLIMIT_INFO()関数を使って開放してください。



3.4.3.8 DshInitTLIMIT_LIST 変数リミット属性定義情報 TLIMIT_LIST の初期設定

(1) 呼出書式

[C, C++]

```
API void APIX DshInitTLIMIT_LIST(
    TLIMIT_LIST *list,           // TLIMIT_LIST 構造体のポインタ
    int          count          // リミット属性情報を設定したい変数の数
);
```

[.NET VB]

```
Sub DshInitTLIMIT_LIST (
    ByRef list As dsh_info.TLIMIT_LIST,
    ByVal count As Int32)
```

[.NET C#]

```
void DshInitTLIMIT_LIST(
    ref TLIMIT_LIST list,
    int count );
```

(2) 引数

list

変数リミット属性定義情報格納用 TLIMIT_LIST 構造体のポインタです。このメンバーを初期設定します。

count

list に設定したい変数の数です。

count の値が =0 の場合は、全変数のリミット属性定義が未定義にすることになります。

(3) 戻り値

なし。

(4) 説明

本関数は S2F45 変数リミット属性定義メッセージを送信する際に使用することができます。

最初に list 内をクリアします。そして、引数で指定された count 分の変数のリミット属性定義に必要な TLIMIT_INFO 構造体のポインタリストを準備します。count は vid_count メンバーに設定されます。

1個の変数のリミット属性定義情報を list 内に加えるために DshAddTLIMIT_INFO()関数が準備されています。

TLIMIT_LIST 構造体の使用後は DshFreeTLIMIT_LIST()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。

3.4.3.9 DshAddTLIMIT_LIST() 変数リミット属性定義情報の追加

(1) 呼出書式

[C, C++]

```
API int APIX DshAddTLIMIT_LIST(
    TLIMIT_LIST *list,           // TLIMIT_LIST 構造体のポインタ
    TLIMIT_INFO *info          // 追加するリミット属性定義情報構造体のポインタ
);
```

[.NET VB]

```
Function DshAddTLIMIT_LIST (
    ByRef list As dsh_info.TLIMIT_LIST,
    ByRef info As dsh_info.TLIMIT_INFO) As Int32
```

[.NET C#]

```
int DshAddTLIMIT_LIST(
    ref TLIMIT_LIST list,
    ref TLIMIT_INFO info );
```

(2) 引数

list

変数属性定義データを加えるの変数リミットリスト情報構造体のポインタです。

info

1個の変数に対するリミット属性定義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	正常に追加できた。
(-1)	list が既に満杯になっていた。

(4) 説明

先にDshInitTLIMIT_LIST()で初期設定された変数リミット属性定義リストに info で指定された属性定義を1個加えます。

info 内にはリミット ID, 上下限データ値が格納されていなければなりません。

正常に加えることができたなら 0 を返却します。

もし、list 内の vid_count メンバーで指定された分の変数リミット属性定義が既に設定されていた場合は、(-1)を返却することによって既に満杯になっていたことを知らせます。

3.4.3.10 DshInitTLIMIT_INFO 変数リミット属性定義情報 TLIMIT_INFO の初期設定

(1) 呼出書式

[C, C++]

```
API void APIX DshInitTLIMIT_INFO(
    TLIMIT_INFO *info,           // TLIMIT_INFO 構造体のポインタ
    int vid,                     // 変数 ID
    int fmt,                     // 変数のフォーマット
    int asize,                   // 変数データの配列サイズ
    int count                    // 設定したいリミット ID(データ)数
);
```

[.NET VB]

```
Sub DshInitTLIMIT_INFO (
    ByRef info As dsh_info.TLIMIT_INFO,
    ByVal vid As Int32,
    ByVal fmt As Int32,
    ByVal asize As Int32,
    ByVal count As Int32)
```

[.NET C#]

```
void DshInitTLIMIT_INFO(
    ref TLIMIT_INFO info,
    uint vid,
    int fmt,
    int asize,
    int count );
```

(2) 引数

info

変数リミット属性定義情報格納用 TLIMIT_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。

vid

リミット情報を設定する対象の変数 ID です。

fmt

変数リミットデータのフォーマットです。(ICODE_A, ICODE_U1 など指定します。)

asize

変数リミットデータの配列サイズです。

count

info に設定するリミット ID の数です。1 個のリミット ID に対し UPPERDB, LOWERDB のリミット値が定義されます。

(3) 戻り値

なし。

(4) 説明

本関数は S2F45 変数リミット属性定義メッセージを送信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。

info 内に count で指定された分の変数リミットデータ格納用 TLIMIT_ID_INFO 構造体ポインタリストを設けます。TLIMIT_INFO 内には 1 個の変数のためのリミット属性定義情報を格納することができます。

リミット ID と上下限リミットデータ行を info 内に順番に設定する関数として DshPutTLIMIT_INFO() が準備されています。

S2F45 送信については複数の変数のリミット属性定義を行うことができるため TLIMIT_INFO 構造体の集合を格納する TLIMIT_LIST 構造体の情報を使用します。

TLIMIT_INFO 構造体の使用後は DshFreeTLIMIT_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。

3.4.3.11 DshPutTLIMIT_INFO() 変数リミット属性定義データの追加

(1) 呼出書式

[C, C++]

```
API int APIX DshPutTLIMIT_INFO(
    TLIMIT_INFO *info,           // TLIMIT_INFO 構造体のポインタ
    int limit_id,               // 変数リミット ID
    void *upperdb,              // 上限リミットデータの格納ポインタ
    void *lowerdb               // 下限リミットデータの格納ポインタ
);
```

[.NET VB]

```
Function DshPutTLIMIT_INFO (
    ByRef info As dsh_info.TLIMIT_INFO,
    ByVal limit_id As Int32,
    ByVal upperdb As IntPtr,
    ByVal lowerdb As IntPtr) As Int32
```

[.NET C#]

```
int DshPutTLIMIT_INFO(
    ref TLIMIT_INFO info,
    int limit_id,
    byte[] upperdb,
    byte[] lowerdb );
```

(2) 引数

info

変数属性定義データを加えるの変数リミット情報構造体のポインタです。

limit_id

リミット ID を指定します。1 個の limit_id に対し上下限データが定義されます。

upperdb

リミットの上限值が格納されているメモリのポインタです。

lowerdb

リミットの下限值が格納されているメモリのポインタです。

(3) 戻り値

戻り値	意味
0	正常に追加できた。
(-1)	info 内が既に満杯になっていた。

(4) 説明

先に DshInitTLIMIT_INFO() で初期設定された変数リミット情報構造体内の属性定義データリストに引数 limit_id, upperdb, lowerdb で指定された属性定義データを 1 個加えます。

upperdb, lowerdb には DshInitTLIMIT_INFO() 関数で指定したフォーマットと配列サイズのデータを設定しなければなりません。

info 内の属性定義データリスト上に、本関数が実行される順に引数で与えられた属性定義データを追加していきます。

設定後 0 を返却します。

もし、info 内の limit_count メンバーで指定された分の属性定義データが既に設定済みであった場合は、(-1)を返却することによって既に満杯になっていたことを知らせます。

3.4.3.12 DshInitTLIMIT_ERR_LIST () 変数リミット応答情報リストの初期化

(1) 呼出書式

[C,C++]

```
API void APIX DshInitTLIMIT_ERR_LIST(
    TLIMIT_ERR_LIST *list,           // 応答情報格納構造体リストのポインタ
    int vlaack,                       // ackデータ
    int err_count                     // 応答情報のリストサイズ (個数 0,1,2...)
);
```

[.NET VB]

```
Function DshPutTLIMIT_ERR_LIST (
    ByRef list As dsh_info.TLIMIT_ERR_LIST,
    ByRef errinfo As dsh_info.TLIMIT_ERR_INFO) As Int32
```

[.NET C#]

```
void DshInitTLIMIT_ERR_LIST(
    ref TLIMIT_ERR_LIST list,
    int vlaack,
    int err_count );
```

(2) 引数

list

TLIMIT_ERR_LIST 応答情報構造体のポインタです。

vlaack

vlaack - ACK の値です。

err_count

変数リミット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、S2F45 に対する応答メッセージのための応答情報を TLIMIT_ERR_LIST 構造体に初期設定します。list で指定された構造体の vlaack メンバーに引数 vlaack の値を設定し、err_count メンバーに引数 err_count の値を設定します。そして、もし、err_count > 0 の場合は、err_list に err_count だけの TERR_LIST エラー情報構造体のポインタリストを設けます。

err_info へのリミットエラー情報の設定には DshPutTLIMIT_ERR_LIST() 関数を使用します。

3.4.3.13 DshPutTLIMIT_ERR_LIST () 変数リミット応答情報の設定

(1) 呼出書式

[C, C++]

```
API void APIX DshPutTLIMIT_ERR_LIST (
    TLIMIT_ERR_LIST *list,          // エラ-情報格納構造体リストのポインタ
    TLIMIT_ERR_INFO *info          // 1個のエラ-情報格納構造体のポインタ
);
```

[.NET VB]

```
Function DshPutTLIMIT_ERR_LIST (
    ByRef list As dsh_info.TLIMIT_ERR_LIST,
    ByRef errinfo As dsh_info.TLIMIT_ERR_INFO) As Int32
```

[.NET C#]

```
int DshPutTLIMIT_ERR_LIST(
    ref TLIMIT_ERR_LIST list,
    ref TLIMIT_ERR_INFO errinfo );
```

(2) 引数

list

プロセスジョブエラー情報構造体のポインタです。

info

1個のエラ-情報 TLIMIT_ERR_INFO 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、list 内の limit_list 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 info (ポインタ) を格納し、0 を返却します。本関数の実行前に DshInitTLIMIT_ERR_LIST()関数を使って list を初期化しておく必要があります。

info 内のメモリはそのまま list 内に使用されますので、info 内のメモリは、DshFreeTLIMIT_ERR_LIST()関数を使って行ってください。

3.4.3.14 DshFreeTLIMIT_ERR_LIST() S2F46 応答情報リスト構造体メモリの開放

(1) 呼出書式

[C, C++]

```
API void APIX DshFreeTLIMIT_ERR_LIST(
    TLIMIT_ERR_LIST *pinfo // メモリを開放したい変数リスト情報リスト構造体の格納ポインタ
);
```

[.NET VB]

```
Sub DshFreeTLIMIT_ERR_LIST (
    ByRef pinfo As dsh_info.TLIMIT_ERR_LIST)
```

[.NET C#]

```
void DshFreeTLIMIT_ERR_LIST(
    ref TLIMIT_ERR_LIST pinfo );
```

(2) 引数

pinfo

メモリを解放したいS2F46 応答情報リスト構造体のポインタです。

(3) 戻り値

なし。

(4) 説明

TLIMIT_ERR_LIST 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。

3.4.3.15 DshInitTLIMIT_ERR_INFO () 変数リミット応答情報の初期化

(1) 呼出書式

[C,C++]

```
API void APIX DshInitTLIMIT_ERR_INFO(
    TLIMIT_ERR_INFO *info,           // 応答情報格納構造体のポインタ
    TVID vid,                         // 対象変数 ID
    int lvack,                         // ackデータ
    int lmt_count                     // 応答情報の数(個数 0,1,2...)
);
```

[.NET VB]

```
Sub DshInitTLIMIT_ERR_INFO (
    ByRef pinfo As dsh_info.TLIMIT_ERR_INFO,
    ByVal vid As Int32,
    ByVal lvack As Int32,
    ByVal lmt_count As Int32)
```

[.NET C#]

```
void DshInitTLIMIT_ERR_INFO(
    ref TLIMIT_ERR_INFO pinfo,
    uint vid,
    int lvack,
    int lmt_count );
```

(2) 引数

info

TLIMIT_ERR_INFO 応答情報構造体のポインタです。

vid

対象となった変数 ID です。

lvack

lvack - ACK の値です。

lmt_count

変数リミットエラー情報の数です。 = 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、S2F45 に対する応答メッセージのための応答情報を TLIMIT_ERR_INFO 構造体に初期設定します。info で指定された構造体の lvack メンバーに引数 lvack の値を設定し、lmt_count メンバーに引数 lmt_count の値を設定します。

そして、もし、lmt_count > 0 の場合は、limit_id、limitack 格納のために lmt_count 分の領域を設けます。

err_info へのリミットエラー情報 (limit_id, limitack) の設定には DshPutTLIMIT_ERR_ID() 関数を使用します。

3.4.3.16 DshPutTLIMIT_ERR_ID () 変数リミット応答情報の設定

(1) 呼出書式

[C, C++]

```
API void APIX DshPutTLIMIT_ERR_ID (
    TLIMIT_ERR_INFO *info,           // 応答情報格納構造体のポインタ
    int limit_id,                    // リミット結果設定対象リミット ID です。
    int limitack                      // 同 ack
);
```

[.NET VB]

```
Function DshPutTLIMIT_ERR_ID (
    ByRef pinfo As dsh_info.TLIMIT_ERR_INFO,
    ByVal limit_id As Int32,
    ByVal limitack As Int32) As Int32
```

[.NET C#]

```
int DshPutTLIMIT_ERR_ID(
    ref TLIMIT_ERR_INFO pinfo,
    int limit_id,
    int limitack );
```

(2) 引数

info

変数リミット応答情報構造体のポインタです。

limit_id

リミットデータに与えられたリミット ID です。

limitack

リミットデータによるチェック結果によるリミット ACK 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、info 内の limit_info の先頭から limit_id の空きを探します。

もし、空きがなければ、(-1)を返却します。

もし、空きがあれば、その空きのリスト位置に limit_id と limitack を格納し、0 を返却します。本関数の実行前に DshInitTLIMIT_ERR_INFO()関数を使って info を初期化しておく必要があります。

3.4.3.17 DshFreeTLIMIT_ERR_INFO() - S2F46 応答情報構造体メモリの開放

(1) 呼出書式

[C, C++]

```
API void APIX DshFreeTLIMIT_ERR_INFO(  
    TLIMIT_ERR_INFO *pinfo // メモリを開放したい S2F46 応答情報構造体の格納ポインタ  
);
```

[.NET VB]

```
Sub DshFreeTLIMIT_ERR_INFO (  
    ByRef info As dsh_info.TLIMIT_ERR_INFO)
```

[.NET C#]

```
void DshFreeTLIMIT_ERR_INFO(  
    ref TLIMIT_ERR_INFO info);
```

(2) 引数

pinfo

メモリを解放したい S2F46 応答情報構造体のポインタです。

(3) 戻り値

なし。

(4) 説明

TLIMIT_ERR_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。

3.4.3.18 DshMakeS2F45Response() - S2F45 の応答メッセージの生成

(1) 呼出書式

[C, C++]

```
API int APIX DshMakeS2F45Response(
    TLIMIT_INFO *info,           // 変数リミット情報が格納されている構造体のポインタ
    TLIMIT_ERR_LIST *erinfo,    // S2F46 応答情報が格納されている構造体ポインタ
    DSHMSG *msg,                // S2F46 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,                 // S2F46 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 DshMakeS2F45Response (
    ByRef info As dsh_info.TLIMIT_LIST,
    ByRef erlist As dsh_info.TLIMIT_ERR_LIST,
    ByRef msg As dshdr2.DSHMSG,
    ByRef buff As Byte,
    ByVal buff_size As Int32) As Int32
```

[.NET C#]

```
int DshMakeS2F45Response(
    ref TLIMIT_LIST info,
    ref TLIMIT_ERR_LIST erlist,
    ref DSHMSG msg,
    byte[] buff,
    int buff_size );
```

(2) 引数

info

変数リミット情報が格納されている構造体のポインタです。

erinfo

S2F46 応答情報 (VID 毎のチェック結果) が格納されているリストの構造体のポインタです。

msg

S2F46 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

S2F46 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

S2F45 に対する S2F46 応答メッセージを erinfo に含まれる変数リミット応答情報に従って作成します。応答情報は、変数 ID 別にチェック結果が erinfo のリスト内に格納されていますので、それに従って S2F46、

メッセージを作成します。

3.4.3.19 DshDecodeS2F48() - S2F48 を変数リミット属性情報リストにデコード

(1) 呼出書式

[C, C++]

```
API int APIX DshDecodeS2F48(
    DSHMSG *msg, // SECS メッセージ 情報構造体のポインタ
    TLIMIT_RSP_LIST *list // デコードした情報リストを格納する構造体のポインタ
);
```

[.NET VB]

```
Function DshDecodeS2F48 (
    ByRef msg As dshdr2.DSHMSG,
    ByRef list As dsh_info.TLIMIT_RSP_LIST) As Int32
```

[.NET C#]

```
int DshDecodeS2F48(
    ref DSHMSG msg,
    ref TLIMIT_RSP_LIST list );
```

(2) 引数

msg

S2F48 の SECS メッセージ 情報構造体のポインタです。

list

デコードした結果の変数リミット属性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

S2F48 メッセージに含まれる変数リミット属性情報リストを、ユーザプログラムが処理しやすい TLIMIT_RSP_LIST 構造体の中にデコードします。TLIMIT_RSP_LIST 内に複数の TLIMIT_RSP_INFO 構造体の情報が含まれます。

変数リミット構造体メンバーのために使用するメモリは全て DSHEng4 側が準備します。

従って、情報処理終了後は、list のメモリを DshFreeTLIMIT_RSP_LIST()関数を使って解放してください。

msg S2F48

L,m

L,2

VID

L,p

UNITS,

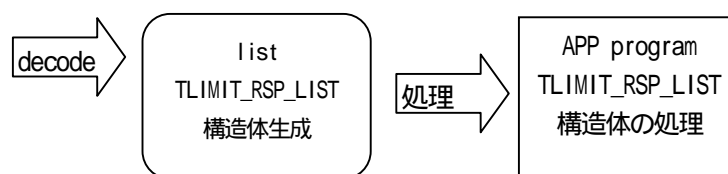
LIMITMIN

LIMITMAX

L,n

L,3

LIMITID1





UPPERDB1

LOWERDB1

.

.

3.4.3.20 DshFreeTLIMIT_RSP_LIST() - 変数リミット応答情報構造体メモリの開放

(1) 呼出書式

[C, C++]

```
API void APIX DshFreeTLIMIT_RSP_LIST(
    TLIMIT_RSP_LIST *pinfo // メモリを開放したい変数リミット情報リスト構造体の格納ポインタ
);
```

[.NET VB]

```
Sub DshFreeTLIMIT_RSP_LIST (
    ByRef list As dsh_info.TLIMIT_RSP_LIST)
```

[.NET C#]

```
void DshFreeTLIMIT_RSP_LIST(
    ref TLIMIT_RSP_LIST list );
```

(2) 引数

pinfo

メモリを解放したい変数リミット応答情報構造体のポインタです。

(3) 戻り値

なし。

(4) 説明

TLIMIT_RSP_LIST 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。

3.4.3.21 DshFreeTVLIMIT_EVENT_INFO() - リミット監視結果情報構造体メモリ開放

(1) 呼出書式

[C, C++]

```
API void APIX DshFreeTVLIMIT_EVENT_INFO(  
    TVLIMIT_EVENT_INFO *pinfo // メモリを開放したいリミット結果情報構造体の格納ポインタ  
);
```

[.NET VB]

```
Sub DshFreeTVLIMIT_EVENT_INFO (  
    ByRef info As dsh_info.TVLIMIT_EVENT_INFO)
```

[.NET C#]

```
void DshFreeTVLIMIT_EVENT_INFO(  
    ref TVLIMIT_EVENT_INFO info );
```

(2) 引数

pinfo

メモリを解放したい変数リミット監視結果情報構造体のポインタです。

(3) 戻り値

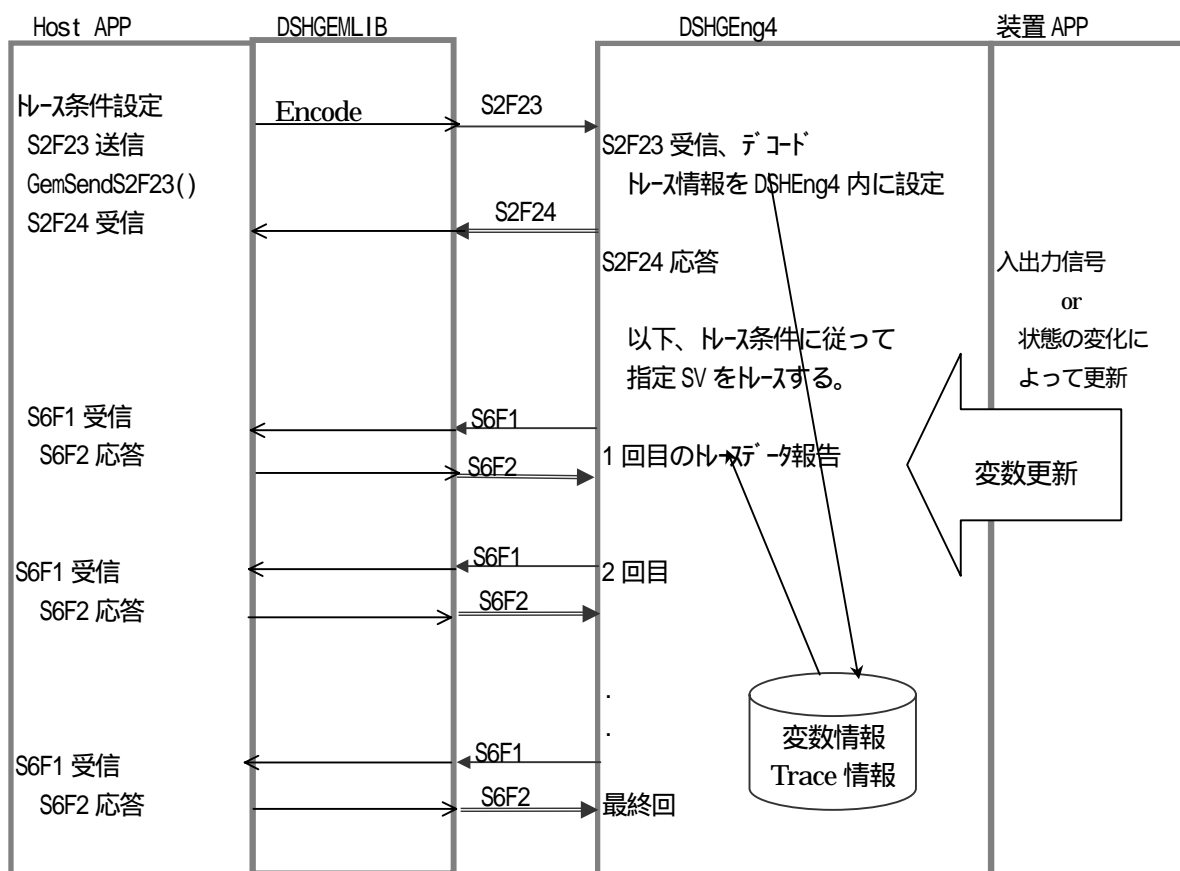
なし。

(4) 説明

TVLIMIT_EVENT_INFO 構造体内で情報格納用に使われているメモリを解放します。

3.5. TR 状態変数トレース情報アクセスサービス関数

トレース情報は、DSHEng4 が管理します。これらの情報をアクセスするために以下の DSHEng4 API 関数を使用します。



(1) 情報アクセスと送信 API 関数

トレース情報のアクセス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	EngAllocTrInfo()	指定された TRID(トレース ID)のトレース情報領域を割当て登録します。
2	EngSetTrInfo()	トレース情報を設定・変更します。
3	EngSetTrInfoX()	トレース情報をインデックス指定で設定・変更します。
4	EngGetTrInfo()	TRID 指定でトレース情報を取得します。
5	EngGetTrInfoX()	TRID 情報インデックス指定でトレース情報を取得します。
6	EngDelTrInfo()	TRID 指定でトレース情報を削除します。
7	EngDelTrInfoX()	TRID 情報インデックス指定でトレース情報を削除します。
8	EngGetTrId()	指定した TRID 情報インデックスの TRID を取得します。
9	EngGetTrIndex()	指定した TRID の情報インデックスを取得します。
10	EngEnableTrace()	TRID 指定で TRID の有効状態を指定します。
11	EngEnableTraceX()	TRID 情報インデックス指定で TRID の有効状態を指定します。
12	EngGetTrList()	TRID の一覧を取得します。

TRID インデクスは、DSHEng4 が管理する各 TRID 領域の番号です。このインデクスの値は、EngAllocTrInfo()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、TRID 情報の取得時に、情報格納構造体のメンバー、index に設定されます。

(2) ライブラリ関数

他に APP が使用できるトレース情報処理用関数として、以下の関数があります。

	API 関数名	機能
1	DshDecodeS2F23()	S2F23 メッセージ を TTRACE_INFO 構造体内にデコード します。
2	DshFreeTTRACE_INFO()	トレース情報が格納されている TTRACE_INFO 構造体の内部で使用されているメモリを開放するための関数。
3	DshCopyTTRACE_INFO()	TTRACE_INFO のトレース情報を別の構造体メモリにコピー します。
4	DshInitTTRACE_INFO()	TTRACE_INFO 構造体内にトレース情報を設定するための初期化関数です。トレース ID, 間隔時間などの設定を行います。
5	DshPutTTRACE_INFO()	TTRACE_INFO 構造体内の SVID リストの指定位置に 1 個の SVID を設定 します。
6	DshMakeS2F23Response()	S2F24 の応答メッセージ を DSHMSG 内に生成する。

(3) ユーザ作成ライブラリ関数

	ライブラリ関数名	機能
1	DshResponseS2F24()	S2F23 トレース設定応答 (DSHEng4 が内部で使用)

3.5.1 使用する情報格納構造体

(1) トレース設定情報用

トレース情報を操作する関数は、共通の情報格納のための TTRACE_INFO 構造体を使用します。TTRACE_INFO とのその内部で使用する他の構造体は下記のとおりです。

```
typedef struct{
    int    index;
    int    state;
    char   *name;           // trace name
    char   *trid;          // trace id
    int    format;         // trace id format
    int    asize;          // trace id array size
    int    max_asize;
    char   *dsper;         // trace 時間周期
    int    dsper_time;     // trace 時間周期-数値
    int    totsamp;       // total sample 数
    int    tot_fmt;
    int    tot_asize;
    int    repgsz;        // report group size
    int    gsz_fmt;
    int    gsz_asize;
    int    svid_count;     // svid list の size
    TSVID  *svid_list;    // svid list
} TTRACE_INFO;
```

(2) トレースデータ情報

装置からの S6F1 メッセージをデコードしたトレースデータを格納するための構造体です。

```
typedef struct{
    void    *trid;         // trace id
    int     format;       // trace id format
    int     asize;        // trace id array size
    int     smpln;        // sampling no.
    char    *stime;       // start time
    int     count;        // no. of data
    TTRACE_SV **sv_list; // sv ptr list
} TTRACE_DATA;
```

```
typedef struct{
    int     format;
    int     asize;
    void    *sv;          // Status Variable data
} TTRACE_SV;
```

3.5.2 TRACE トレース情報アクセス関数

3.5.2.1 EngAllocTrInfo() - トレース情報の登録

(1) 呼出書式

[C, C++]

```
API int APIX EngAllocTrInfo(
    char *trid,          // トレース ID 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 EngAllocTrInfo (
    ByVal trid As String,
    ByRef index As Int32) As Int32
```

[.NET C#]

```
int EngAllocTrInfo(
    byte[] trid,
    ref int index );
```

(2) 引数

trid

登録したいトレース 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 TRID は既に登録されていた。
(-1)	登録できなかった。

(4) 説明

トレース情報を新規にシステムに登録するための関数です。

登録は、引数 trid で与えられるトレース 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、trid に指定されたトレース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。

3.5.2.2 EngSetTrInfo() - トレース情報の設定

EngSetTrInfoX() インデクス指定によるトレース情報の設定

(1) 呼出書式

[C, C++]

```
API int APIX EngSetTrInfo(
    TTRACE_INFO *pinfo // トレース情報格納構造体のポインタ
);

API int APIX EngSetTrInfoX(
    int index, // EngAllocTrInfo() で得られたインデクス値
    TTRACE_INFO *pinfo // トレース情報格納構造体のポインタ
);
```

[.NET VB]

```
Function EngSetTrInfo (
    ByRef trinfo As dsh_info.TTRACE_INFO) As Int32

Function EngSetTrInfoX (
    ByVal index As Int32,
    ByRef trinfo As dsh_info.TTRACE_INFO) As Int32
```

[.NET C#]

```
int EngSetTrInfo(
    ref TTRACE_INFO trinfo );

int EngSetTrInfoX(
    int index,
    ref TTRACE_INFO trinfo );
```

(2) 引数

pinfo

設定したいトレース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

トレース情報のインデクスです。登録時に EngAllocTrInfo() 関数によって与えられます。インデクスは、トレース ID から EngGetTrdIndex() 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 に格納されているトレース情報の設定・変更に使います。

引数 pinfo 内の trid メンバーに指定されるトレース ID の情報として設定されます。

pinfo 内には TRID の他、トレース対象となる変数 ID、トレース周期、合計サンプル数などの情報が含ま

れます。

指定した TRID が既に登録済みである場合には、pinfo 内の情報に書き換えられます。

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

3.5.2.3 EngGetTrInfo() - トレース情報の取得 EngGetTrInfoX() - インデクス指定でのトレース情報の取得

(1) 呼出書式

[C, C++]

```
API int APIX EngGetTrInfo(
    char    *trid,           // TRID が格納されている領域のポインタ
    TTRACE_INFO *pinfo      // トレース情報を格納する構造体のポインタ
);
```

```
API int APIX EngGetTrInfoX(
    int    index,          // TRID 情報のインデクス
    TTRACE_INFO *pinfo    // トレース情報を格納する構造体のポインタ
);
```

[.NET VB]

```
Function EngGetTrInfo (
    ByVal trid As String,
    ByRef trinfo As dsh_info.TTRACE_INFO) As Int32
```

```
Function EngGetTrInfoX (
    ByVal index As Int32,
    ByRef trinfo As dsh_info.TTRACE_INFO) As Int32
```

[.NET C#]

```
int EngGetTrInfo(
    byte[] trid,
    ref TTRACE_INFO trinfo );
```

```
int EngGetTrInfoX(
    int index,
    ref TTRACE_INFO trinfo );
```

(2) 引数

trid

TRID が格納されている領域のポインタです。

pinfo

取得したトレース情報を格納する構造体領域のポインタです。

index

TRID 情報のインデクスです。登録時に EngAllocTrInfo() 関数によって与えられます。
インデクスは、TRID から EngGetTrIdIndex() 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。(TRID が未登録であった。)

(4) 説明

trid に指定されている TRID のトレース情報を取得し、pinfo 構造体に格納します。

TTRACE_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、trid, vid 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。

```
TTRACE_INFO      *pinfo;

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

3.5.2.4 EngDelTrInfo() - トレースの削除

EngDelTrInfoX() - インデクスでのトレースの削除

(1) 呼出書式

[C, C++]

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

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

[.NET VB]

```
Function EngDelTrInfo (
    ByVal trid As String) As Int32

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

[.NET C#]

```
int EngDelTrInfo(
    byte[] trid );

int EngDelTrInfoX(
    int index );
```

(2) 引数

trid
TRID が格納されている領域のポインタです。

index
削除したいトレース情報のインデクスです。

(3) 戻り値

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

(4) 説明

trid または index に指定されている TRID のトレース情報を登録から削除します。

3.5.2.5 EngEnableTrace() トレース有効状態の設定 EngEnableTraceX() インデクスでのトレース有効状態の設定

(1) 呼出書式

[C, C++]

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

```
API int APIX EngSEnableTraceX(
    int index // TRID 情報のインデクス
);
```

[.NET VB]

```
Function EngEnableTrace (
    ByVal trid As String) As Int32
```

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

[.NET C#]

```
int EngEnableTrace(
    byte[] trid );
```

```
int EngEnableTraceX(
    int index );
```

(2) 引数

trid

TRID (トレース ID) が格納されている領域のポインタです。

index

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

(3) 戻り値

戻り値	意味
0	正常に有効状態に設定できた。
1	既に有効状態になっていた。
(-1)	TRID が未登録であった。

(4) 説明

トレース情報の有効状態を設定します。

トレース状態が有効に設定されると、DSHEng4 は、trid のために登録されているトレース情報に従って、登録されている VID 変数のトレースを開始します。トレースされた情報は、SS6F1 メッセージを使って装置に送信されます。

トレース開始条件としては、トレース情報が登録されており、トレースが休止状態であることとなります。既に有効になっていた場合には、戻り値 = 1 を返却します。

3.5.2.6 EngGetTrList() 全登録トレース ID 取得関数

(1) 呼出書式

[C, C++]

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

[.NET VB]

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

[.NET C#]

```
int EngGetTrList(
    IntPtr list );
```

(2) 引数

list

取得できた TRID 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 TRID 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 TRID 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;
```

3.5.2.7 EngGetTrId() インデクスから TRACEID (トレース ID) の取得

(1) 呼出書式

[C, C++]

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

[.NET VB]

```
Function EngGetTrId (
    ByVal index As Int32,
    ByVal trid As String) As Int32
```

[.NET C#]

```
int EngGetTrId(
    int index,
    byte[] trid );
```

(2) 引数

index

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

trid

TRACEID 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

TRACE 情報のインデクスから TRACEID (トレース ID) を取得し、trid 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。

3.5.2.8 EngGetTrIdIndex() TRACEID (トレース ID) からインデクスの取得

(1) 呼出書式

[C, C++]

```
API int APIX EngGetTrIdIndex(
    char *trid,           // TRACEID 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 EngGetTrIdIndex (
    ByVal trid As String,
    ByRef index As Int32) As Int32
```

[.NET C#]

```
int EngGetTrIdIndex(
    byte[] trid,
    ref int index);
```

(2) 引数

trid

インデクスを取得したい対象の TRACEID 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

trid に指定される TRACEID (トレース ID) から TRACE 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。

3.5.3 TR トレース関連ライブラリ関数

3.5.3.1 DshDecodeS2F23() - S2F23 をトレース情報にデコード

(1) 呼出書式

[C, C++]

```
API int APIX DshDecodeS2F23(
    DSHMSG *msg, // SECS メッセージ 情報構造体のポインタ
    TTRACE_INFO *pinfo // デコードした情報を格納する構造体のポインタ格納用
);
```

[.NET VB]

```
Function DshDecodeS2F23 (
    ByRef msg As dshdr2.DSHMSG,
    ByRef info As dsh_info.TTRACE_INFO) As Int32
```

[.NET C#]

```
int DshDecodeS2F23(
    ref DSHMSG msg,
    ref TTRACE_INFO info );
```

(2) 引数

msg

S2F23 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

デコードしたトレース情報を格納した構造体ポインタを格納するポインタです。

(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

S2F23 メッセージに含まれるトレース情報を、ユーザプログラムが処理しやすい TTRACE_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 DshFreeTTRACE_INFO() 関数を使って開放してください。

msg S2F23

```
L,5
trid
dsper
totsmp
.
.
```



3.5.3.2 DshFreeTTRACE_INFO() - トレース情報構造体メモリの開放

(1) 呼出書式

[C, C++]

```
API void APIX DshFreeTTRACE_INFO(  
    TTRACE_INFO *pinfo // メモリを開放したいトレース情報が格納されている構造体のポインタ  
);
```

[.NET VB]

```
Sub DshFreeTTRACE_INFO (  
    ByRef info As dsh_info.TTRACE_INFO)
```

[.NET C#]

```
void DshFreeTTRACE_INFO(  
    ref TTRACE_INFO info );
```

(2) 引数

pinfo

メモリを解放したいトレース情報構造体のポインタです。

(3) 戻り値

なし。

(4) 説明

TTRACE_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、TTRACE_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。

3.5.3.3 DshCopyTTRACE_INFO() - トレース情報構造体メモリのコピー

(1) 呼出書式

[C, C++]

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

[.NET VB]

```
Function DshCopyTTRACE_INFO (
    ByRef pinfo As dsh_info.TTRACE_INFO,
    ByRef sinfo As dsh_info.TTRACE_INFO) As Int32
```

[.NET C#]

```
int DshCopyTTRACE_INFO(
    ref TTRACE_INFO pinfo,
    ref TTRACE_INFO sinfo );
```

(2) 引数

pinfo

トレース情報のコピー先構造体メモリのポインタです。

sinfo

コピー元のトレース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 TTRACE_INFO 構造体内に格納されているトレース情報を dinfo で指定された TTRACE_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、DshFreeTTRACE_INFO()関数を使って開放してください。

3.5.3.4 DshInitTTRACE_INFO() トレース S2F43 情報の初期設定

(1) 呼出書式

[C, C++]

```
API int APIX DshInitTTRACE_INFO(
    TTRACE_INFO *info,           // トレース情報構造体の格納ポインタ
    char *trid,                 // トレース ID へのポインタ
    char *dsper,                // データ収集時間データ格納ポインタ(文字列)
    int totsmp,                 // 総サンプル数
    int repgsz,                 // レポートグループのサイズ
    int svid_count              // トレース対象とする SV の数
);
```

[.NET VB]

```
Function DshInitTTRACE_INFO (
    ByRef info As dsh_info.TTRACE_INFO,
    ByVal trid As String,
    ByVal dsper As String,
    ByVal totsmp As Int32,
    ByVal repgsz As Int32,
    ByVal svid_count As Int32) As Int32
```

[.NET C#]

```
int DshInitTTRACE_INFO(
    ref TTRACE_INFO info,
    byte[] trid,
    byte[] dsper,
    int totsmp,
    int repgsz,
    int svid_count );
```

(2) 引数

info

初期設定する TTRACE_INFO トレース情報構造体のポインタです。

trid

トレース ID (文字列) が格納されている領域のポインタです。

dsper

トレースデータ収集時間を文字列です。“hhmmsscc”(8文字固定)の文字列で表現します。
hh:時間, mm:分, ss:秒, cc:1/100秒

totsmp

合計サンプル数を指定します。

repgsz

レポートグループのサイズです。(装置が1個のS6F1で送信するサンプル数)

svid_count

トレース対象のSVの数です。(TTRACE_INFO内のリストに設定するSV数)

(3) 戻り値

戻り値	意味
-----	----

0	正常に設定できた。
(-1)	引数に指定された値が正しくなかった。

(4) 説明

本関数は、info で指定された TTRACE_INFO トレース情報構造体の初期設定を行います。

最初に info 内をゼロクリアします。

info のメンバー内にそれぞれの引数を設定します。

TTRACE_INFO メンバー内の以下のメンバーについては固定データを設定します。

format = ICODE_A (ICODE_A は DSHDR2 ドライバが定義するデータ型の FORMAT の ASCII です)

tot_fmt = ICODE_A, tot_asize=0

gsz_fmt = ICODE_A, gsz_asize=0

引数の値が以下のケースの場合はエラー(-1)を返却します。

trid 文字列長が=0、dsper 文字列長が8でない。

totsmp, repgsz または svid_count の値が=0である。

トレースする SVID の TTRACE_INFO 内への設定は DshPutTTRACE_INFO()関数を使用します。

TPOOL_INFO info 使用が済んだら、情報内に確保して使用したメモリは DshFreeTTRACE_INFO ()関数を使って開放することができます。

3.5.3.5 DshPutTTRACE_INFO() トレース S2F23 情報の SVID 設定

(1) 呼出書式

[C, C++]

```
API int APIX DshPutTTRACE_INFO(
    TTRACE_INFO *info,           // トレース情報構造体の格納ポインタ
    int order,                   // SVID リストの設定位置(0,1,...)
    TSVID svid                   // 設定する SVID(状態変数 ID)
);
```

[.NET VB]

```
Function DshPutTTRACE_INFO (
    ByRef info As dsh_info.TTRACE_INFO,
    ByVal order As Int32,
    ByVal svid As Int32) As Int32
```

[.NET C#]

```
int DshPutTTRACE_INFO(
    ref TTRACE_INFO info,
    int order,
    uint svid );
```

(2) 引数

info

初期設定する TTRACE_INFO トレース情報構造体のポインタです。

order

info 内の svid_list リスト内の設定位置を指定します。位置は先頭が 0 から始まります。

svid

設定したい装置状態変数 ID です。

(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(order の値が間違っていた)

(4) 説明

本関数は、info で指定された TTRACE_INFO トレース情報構造体内のメンバー svid_list のリストの order 番目の位置に svid で指定された装置状態変数 ID を設定します。

order の値が、info 内の svid_count の値以上であった場合は、エラー(-1)を返却します。

3.5.3.6 DshMakeS2F23Response() - S2F23 の応答メッセージの生成

(1) 呼出書式

[C, C++]

```
API int APIX DshMakeS2F23Response(
    BYTE tiaack,           // S2F24 に設定する TIAACK です。
    DSHMSG *smsg,         // S2F24 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,           // S2F24 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 DshMakeS2F23Response (
    ByVal tiaack As Int32,
    ByRef smsg As dshdr2.DSHMSG,
    ByRef buff As Byte,
    ByVal buff_size As Int32) As Int32
```

[.NET C#]

```
int DshMakeS2F23Response(
    int tiaack,
    ref DSHMSG smsg,
    byte[] buff,
    int buff_size );
```

(2) 引数

tiaack
S2F24 メッセージの TIAACK です。

msg
S2F24 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
S2F24 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

S2F23 に対する S2F24 応答メッセージを sinfo に含まれるトレース情報と応答情報に従って作成します。応答情報は、tiaack を S2F24 の TIAACK として設定します。tiaack はユーザがトレース情報を評価した結果です。

3.5.4 ユーザ作成ライブラリ関数

3.5.4.1 DshResponseS2F24() S2F24 トレース設定応答メッセージ

(1) 呼出書式

[C, C++]

```
API int APIX DshResponseS2F24(
    ID_TR trid, // DSHDR2 のトランザクション ID
    TTRACE_INFO *info, // トレース設定メッセージ 情報格納領域のポインタ
    int tiaack // S2F24 応答情報格納用構造体のポインタ
);
```

[.NET VB]

```
Function DshResponseS2F44 (
    ByVal trid As Int32,
    ByRef info As dsh_info.TSPOOL_INFO,
    ByRef erinfo As dsh_info.TSPOOL_ERR_INFO) As Int32
```

[.NET C#]

```
int DshResponseS2F24(
    uint trid,
    ref TTRACE_INFO info,
    int tiaack );
```

(2) 引数

trid

S2F23 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

トレース設定モード情報が格納されている構造体のポインタです。

tiaack

送信する応答メッセージ ack の値です。

(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

トレース設定メッセージ S2F23 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(dsh_ulib.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 tiaack 情報から S2F24 メッセージを組み立て、その後、S2F24 メッセージを送信します。

なお、S2F24 メッセージの組み立てに、DshMakeS2F24Response()関数を使用できます。