ＳＥＣＳ／ＨＳＭＳ通信制御ドライバープログラム

Windows 版

（ Level-１ ）

ユーザー・マニュアル

Version-3

[Version-3 の新機能]

1. ﾒｯｾｰｼﾞ組立て作成関連関数追加
   SECS-II ﾒｯｾｰｼﾞを自由に組立てることができます。
   ﾍﾀﾞｰ部、ﾘｽﾄ構造部

2. ﾒｯｾｰｼﾞ情報ﾃﾞｺｰﾄﾞ関数
   受信ﾒｯｾｰｼﾞの解読処理が簡単にできます。
   ﾒｯｾｰｼﾞ ID(SxFy), W, ｼｽﾃﾑﾊﾞｲﾄ 取得
   ﾘｽﾄを構造するデータアイテム情報取得

3. データアイテム編集用関数の追加
   8,16,32,64 ﾋﾞｯﾄﾃﾞｰﾀの 10,16 進文字列変換関数
   現日付時刻ﾃﾞｰﾀ編集関数

４．通信ログメッセージのモニタリング機能を追加しました。
   2006.7.7

２００６年７月

株式会社 データマップ

［改訂履歴］

No.	日　付	内　容	備　考
1.	2002.02.04	バージョン-２の初版	
2.	2003.02.03	製品シリアル番号取得関数 SECS_GetSerialNo()を追加した。	
3.	2004.09.24	2.1.2（2）　通信チャンネルポート定義コマンド表 の no.9　T1,T2,...T8 の説明訂正	
4.	2005.1.6	Version-3 の改訂による API 関数の追加その他 **1. SECS-II 通信ﾒｯｾｰｼﾞ生成関連関数** **2. SECS-II 通信ﾒｯｾｰｼﾞﾃﾞｰﾀｱｲﾃﾑ取得関連関数** **3. 各種編集関数** **4. 環境定義ｺﾏﾝﾄﾞの追加** **(1)日本語/英語指定ｺﾏﾝﾄﾞ LANGUAGE**	
5.	2005.3.31	E_printf, E_hex_dump()の記述を追加	
6.	2006.7.7	ﾘﾓｰﾄでの通信ﾛｸﾞﾓﾆﾀﾘﾝｸﾞできるようにした。 環境定義ｺﾏﾝﾄﾞに MON_PORT ｺﾏﾝﾄﾞを追加した。 2.1.2（1）ｺﾏﾝﾄﾞ表の７番目	

［取り扱い注意］

- 本ドライバーを使用するためには本ドライバー使用ライセンスの取得が必要になります。
  ライセンスは本ソフトウェアを組込むコンピュータに対して１個のライセンス購入が必要になります。
- 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。

- 目　次 -

－以　上－

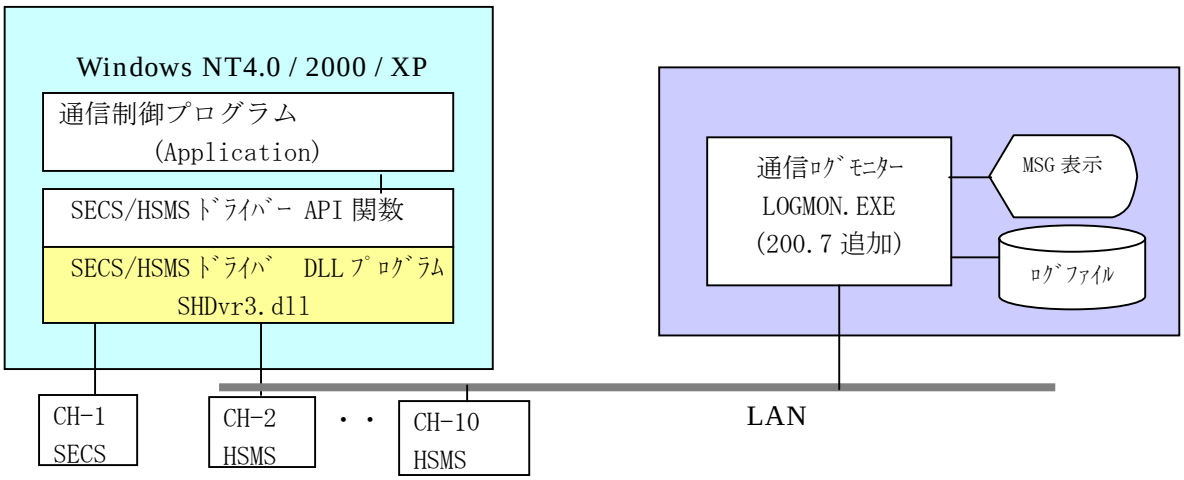
１．　概要

本仕様書は、Windows NT, 2000 の下で動作するＳＥＣＳ／ＨＳＭＳ通信制御ドライバー外部仕様について記述します。

本ドライバーはＳＥＭＩスタンダードＳＥＣＳならびにＨＳＭＳ－ＳＳプロトコルに準拠する仕様をサポートする通信制御ドライバーであり、DLL(Dynamic Link Library)で提供されます。

本通信制御ドライバーの特長は以下の通りです。

- SEMI スタンダード SECS-1、HSMS-SS 通信プロトコルに準拠。
- SECS/HSMS 最大１０チャンネル(=ﾎﾟｰﾄ) の同時通信をサポート。
- 通信チャンネルのﾎﾟｰﾄ割当、通信条件等は、環境定義ファイル上にコマンドで定義。
- 応答期待メッセージの通信に対する送信／受信／T3 タイムアウト監視専用ＡＰＩ関数。
- HSMS における Linktest 通信の実施。
- 送受信 API 関数については制御モードとしてブロック／非ブロックモードを選択可能。
- １次メッセージ送信と受信が１個の API 関数で可能。
- **SECS-II 通信メッセージの生成機能（新規機能）**
- **SECS-II 通信メッセージ内データアイテムの取得機能（新規機能）**



[動作環境]

本ドライバーが動作するＯＳの環境は、Windows NT-4.0 (SP4 以上)、2000 プロフェッショナル、XP プロフェッショナルです。

[提供されるﾌﾟﾛｸﾞﾗﾑﾌｧｲﾙ名]

（１）　SHDvr3.dll
（２）　SHDvr3.lib
（３）　D_API.h
　　(Microsoft Visual C++ Version-6 環境で作成)

[ｱﾌﾟﾘｹｰｼｮﾝ開発言語]

C, C++, VB-6, DELPHI7　(VB-6.0 についての情報は、**付録-C** に記述されています。)

２．　使用にあたっての予備情報

２．１　通信環境定義ファイル

本ドライバーは SECS_Start( )ＡＰＩ関数によって、通信開始のための環境諸条件を設定します。

SECS_Start( )関数には、引数として通信環境定義ファイルのファイル名を指定します。

この通信環境ファイルは、テキスト形式のファイルであり、２．１．１　で述べる環境定義コマンドによって、通信を行うチャンネル（＝ポートと同じ意味）についての通信プロトコル条件などのパラメータが書き連ねられることになります。

また、チャンネルに依存しないログ記録ファイルなどのコマンドも準備され設定することができます。

環境定義コマンドの一覧表を２．２．２に示す。　また、定義例を付録-Ａに示します。

２．１．１　コマンドの書式

コマンドは１行で表し、コマンド文字列に使用する文字コードは全て半角でなければなりません。

（１）　通信チャンネル定義コマンド

次のように START コマンドと END コマンドで囲んで１つのチャンネルの定義を行います。

```
START = 〈ﾁｬﾝﾈﾙ番号〉                    ;  〈ﾁｬﾝﾈﾙ番号 〉: 1～10
      〈ｺﾏﾝﾄﾞ-1〉
            ・
      〈ｺﾏﾝﾄﾞ－ｎ〉

 END
```

（２）　チャンネル定義コマンド以外のコマンド

```
〈ｺﾏﾝﾄﾞ〉 = 〈ﾊﾟﾗﾒｰﾀ〉
```

チャンネル定義コマンド群の中（START～END）以外の行に記述します。
通常は、ファイルの先頭行から記述します。

なお、コマンド行のなかに　‘;’（半角のセミコロン文字）を含めると、それ以降はコメント扱いとなります。

空の行は無視されます。(ブランク＋改行または改行だけの行)

２．１．２　通信環境定義コマンド

（１）　一般コマンド（ = チャンネル定義コマンド以外のコマンド）

以下のコマンドは必須ではありません。

No.	ｺﾏﾝﾄﾞ名	ﾊﾟﾗﾒｰﾀ	機　能
1.	MAX_LENGTH	<ﾒｯｾｰｼﾞ最大長>	送受信できるSECSﾒｯｾｰｼﾞの最大ﾊﾞｲﾄ長を指定する。 ﾃﾞﾌｫﾙﾄ値は 65,536ﾊﾞｲﾄ(= 0x10000ﾊﾞｲﾄ) 例　MAX_LENGTH = 8192
2.	LOGFILE	<ﾛｸﾞﾌｧｲﾙ名>	ﾛｸﾞ情報を記録するﾌｧｲﾙ名を指定する。 本ｺﾏﾝﾄﾞがなければﾛｸﾞ情報の記録はされない。 例　　LOGFILE = c:¥secs¥log¥EQ0100.log
3.	LOGSIZE	<最大行数>	2. の LOGFILE で指定されたﾛｸﾞﾌｧｲﾙに記録される最大行数を設定する。 LOGSIZE で指定された値を n とすると、ﾄﾞﾗｲﾊﾞｰは 記録行数が（ n + n/10 ）に達したら、記録されたﾛｸﾞ情報の古い情報を　n/10 行数分だけ削除し、n 行分のﾌｧｲﾙｻｲｽﾞに調整する。 ﾃﾞﾌｫﾙﾄ値は 100,000 行。 例　LOGSIZE = 100000
4.	LOGFMT	<ITEM/LIST/HEX/CODE>	通信ﾒｯｾｰｼﾞのﾛｸﾞ編集書式を指定する。 ITEM ： ｱｲﾃﾑ形式 LIST ： ﾘｽﾄ構造形式 HEX　： ﾊﾞｲﾄ単位の 16 進形式 CODE：SECS(ｼﾘｱﾙ通信)の場合制御ｺｰﾄﾞ, ENQ, EOT, ACK, NAK もﾛｸﾞに含める。 なお、ITEM と LIST とは排他的であるが、HEX,CODE は排他的ではない。本ｺﾏﾝﾄﾞは複数の設定が許される。 例　LOGFMT = LIST LOGFMT = CODE この例では、LIST 構造＋制御符号のﾛｸﾞの設定になる。

5.	LINKLOG	<ON/OFF>	HSMS 通信での Linktest.req/rsp のﾛｸﾞを記録するかどうかを “ON”, “OFF”で指定する。“ON”ならば、ﾛｸﾞﾌｧｲﾙに記録される。”OFF”ならば記録されない。ﾃﾞﾌｫﾙﾄ値は OFF。 例　LINKLOG = OFF
6.	LANGUAGE	<JAPANESE/ENGLISH>	ﾛｸﾞ情報表示言語を指定します。 ﾃﾞﾌｫﾙﾄは日本語です。 例 LANGUAGE = ENGLISH
7.	MON_PORT	<TCP ﾎﾟｰﾄ番号>	LOG MONITOR で使用する TCP/IP ポートを指定します。 ポートの値が 1024 以上の場合にのみ LOG MONITOR を有効にします。 例 MON_PORT = 9999 できるだけ、HSMS 通信あるいはｱﾌﾟﾘｹｰｼｮﾝで使用されないポートを指定してください。 本コマンドで有効なポートを設定すると他のコンピュータから本ドライバーの通信ログをモニタリングすることができます。 他のコンピュータ側では次プログラムを使用します。 LOGMON.EXE 詳しくは、DSHﾄﾞﾗｲﾊﾞｰ ログモニタ説明書を参照してください。

（２）　通信チャンネルポート定義コマンド

通信チャンネルについて個別に規定するためのコマンドであり、下表のコマンドが準備されています。

No.	ｺﾏﾝﾄﾞ名	ﾊﾟﾗﾒｰﾀ	機 能
1.	START	<ﾁｬﾝﾈﾙ番号>	<ﾁｬﾝﾈﾙ番号>のﾁｬﾝﾈﾙについてﾌﾟﾛﾄｺﾙの種類、通信条件について定義する。 例　START = ２
2.	END	（なし）	START ｺﾏﾝﾄﾞで始まる１つのﾁｬﾝﾈﾙの定義の終わりを示す。 例　END
２.	SECS	<MASTER/SLAVE>	指定ﾁｬﾝﾈﾙが SECS(ｼﾘｱﾙ)通信であり、ﾊﾟﾗﾒｰﾀで MASTER か SLAVE 側かを指定する。 例　SECS = MASTER
4.	HSMS	<ACTIVE/PASSIVE>	指定ﾁｬﾝﾈﾙが HSMS 通信であり、ﾊﾟﾗﾒｰﾀで ACTIVE か PASSIVE のどちらなのかを指定する。 例　HSMS = ACTIVE
5.	PORT	<COMi> <TCP ﾎﾟｰﾄ>	(1) SECS の場合、NT ｼｽﾃﾑ内のｼﾘｱﾙﾎﾟｰﾄの番号を指定する。 例　PORT = COM1 (2) HSMS の場合、指定ﾁｬﾝﾈﾙが使用する TCP ﾎﾟｰﾄ番号を指定 する。表現は１０進表現である。 例　PORT = 5000
6.	DVID	<ﾃﾞﾊﾞｲｽ ID>	SECS-I ﾚﾍﾞﾙの Device ID を１６進数４桁で指定する。 HSMS では Session ID となる。 例　DVID = 0012
7.	BAUD	<ﾎﾞｰﾚｰﾄ>	SECS 通信時のｼﾘｱﾙﾎﾟｰﾄの通信速度 BAUDRATE を数値で指定する。 数値は、600, 1200, 2400, 4800, 9600 のいずれかの値でなければならない。 例　BAUD = 9600

8.	IP	<IP ｱﾄﾞﾚｽ>	HSMS の ACTIVE ﾎﾟｰﾄ時,接続対象となる PASSIVE 側の IP ｱﾄﾞﾚｽ 32 ﾋﾞｯﾄを 8bit を単位としてﾄﾞｯﾄで区切って表記する。 例　192.168.1.1
9.	T1 T2 T3 T4 T5 T6 T7 T8	<時間値>	SECS/HSMS ﾌﾟﾛﾄｺﾙ上のﾀｲﾏｰ値を、100ms 単位の数値で指定する。 SECS 通信では, T1,T2,T3,T4 の設定が必要である。 HSMS 通信では, T3,T5,T6,T7,T8 の設定が必要である。 例 T3 = 450　　(45.0 秒)
10.	RETRY	<回数>	SECS 通信時、ﾌﾟﾛﾄｺﾙ上のｴﾗｰを検出したときに、連続して何回の再試行をするかを回数で指定する。 (HSMS ﾌﾟﾛﾄｺﾙでは必要ありません。) 例　RETRY = 3
11.	LINKTIME	<時間値>	HSMS 通信時のﾘﾝｸﾃｽﾄを自ﾎﾟｰﾄから行うための Linktest.req 送信を行う時間間隔を秒単位で指定する。 値が = 0 の場合は、Linktest.req を送信しない。 例　LINKTIME = 10　　(10 秒間隔)
12.	DVID_CK	<ON/OFF>	接続された相手からの受信ﾒｯｾｰｼﾞを受け入れるかどうかの 判断を DeviceID の照合で行うかどうかを指定する。 = ON の場合、Device ID (Session ID)の照合で、一致した場合だけ受け入れることになる。 =OFF の場合は Device ID の照合は行わないで受け入れる。 例　DVID_CK = ON
13.	S9F1_RSP	<ON/OFF>	受信したﾒｯｾｰｼﾞのﾃﾞﾊﾞｲｽ ID が期待したものでなかった時、自動的に S9F1 ﾒｯｾｰｼﾞを相手に送信するかどうかを指定する。 本ｺﾏﾝﾄﾞは、DVID_CK ｺﾏﾝﾄﾞとも連動する。 S9F1 が送信される条件は、①DVID が不一致 ②DVID_CK=ON ③S9F1_RSP=ON の３つが満たされることである。 例　S9F1_RSP = ON
14.	S9F9_RSP	<ON/OFF>	応答ﾒｯｾｰｼﾞを期待する１次ﾒｯｾｰｼﾞ送信後、T3 時間経過しても応答ﾒｯｾｰｼﾞを受信できなかった場合、S9F9

			ﾒｯｾｰｼﾞを自動的に送信するかどうかを指定する。 =ON のとき送信する。 例　S9F9_RSP = ON 本ｺﾏﾝﾄﾞは SECS_SendRecv()API 関数が実行されたときにだけ有効である。

３．　通信制御ＡＰＩ関数

本章では、ＳＥＣＳ,ＨＳＭＳ通信制御のためのＡＰＩ関数について説明します。

説明は、c 言語用の API 関数について説明します。なお、VB-6.0 用の関数宣言などについては、付録-C を参照願います。

本ドライバーでは、次の９つの通信制御関連ＡＰＩ関数が提供されます。

番号	関数名	機能概略
1.	SECS_Start( )	SECS/HSMSﾄﾞﾗｲﾊﾞｰの環境準備と開始
2.	SECS_Terminate( )	SECS/HSMSﾄﾞﾗｲﾊﾞｰの終了
3.	SECS_Open( )	ﾁｬﾝﾈﾙのｵｰﾌﾟﾝ
4.	SECS_Close( )	ﾁｬﾝﾈﾙのｸﾛｰｽﾞ
5.	SECS_GetStatus( )	通信ﾁｬﾝﾈﾙの送受信準備確認
6.	SECS_SendRecv( )	１次ﾒｯｾｰｼﾞ送信後応答２次ﾒｯｾｰｼﾞ受信
7.	SECS_Send( )	ﾒｯｾｰｼﾞの送信
8.	SECS_RecvCk( )	受信ﾒｯｾｰｼﾞの有無の確認
9.	SECS_Recv( )	ﾒｯｾｰｼﾞ受信

送信、受信については、次の２つの制御モードのどちらかを指定することができます。

（１）　**ブロックモード**

ブロックモードを指定すると、プログラムのコントロールは、その関数が指定した送受信が終了するまで呼出元に戻ってきません。

（２）　**非ブロックモード**

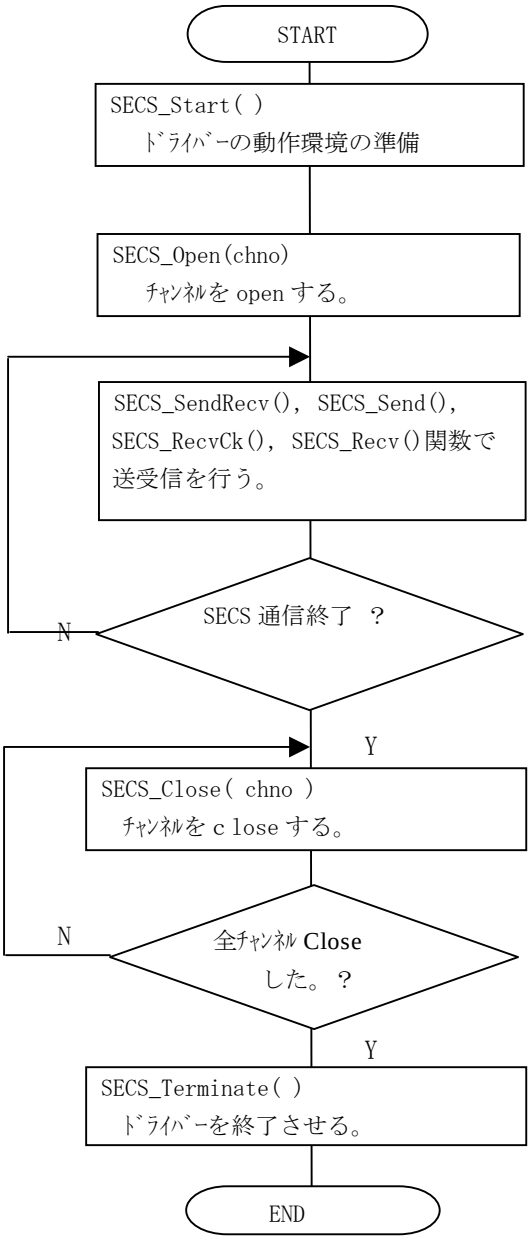
非ブロックモードを指定すると、API 関数が指定する制御をドライバーに委託し、ただちに呼出元に戻ってきます。指定制御が終了したら、API 関数が与えた終了結果情報変数内に終了結果を格納します。呼出元は、終了したかどうかの確認をこの結果情報のポーリング（値が(-1)から別の値への変化するのを確認する）によって知ることができます。

詳しくは、次節に示すフローチャートを参照してください。

３．１　ＡＰＩ関数の使用のためのフローチャート

次節で述べるＡＰＩ関数を使ってどのように通信制御を行うかをフローチャートで表現します。

３．１．１　開始／終了

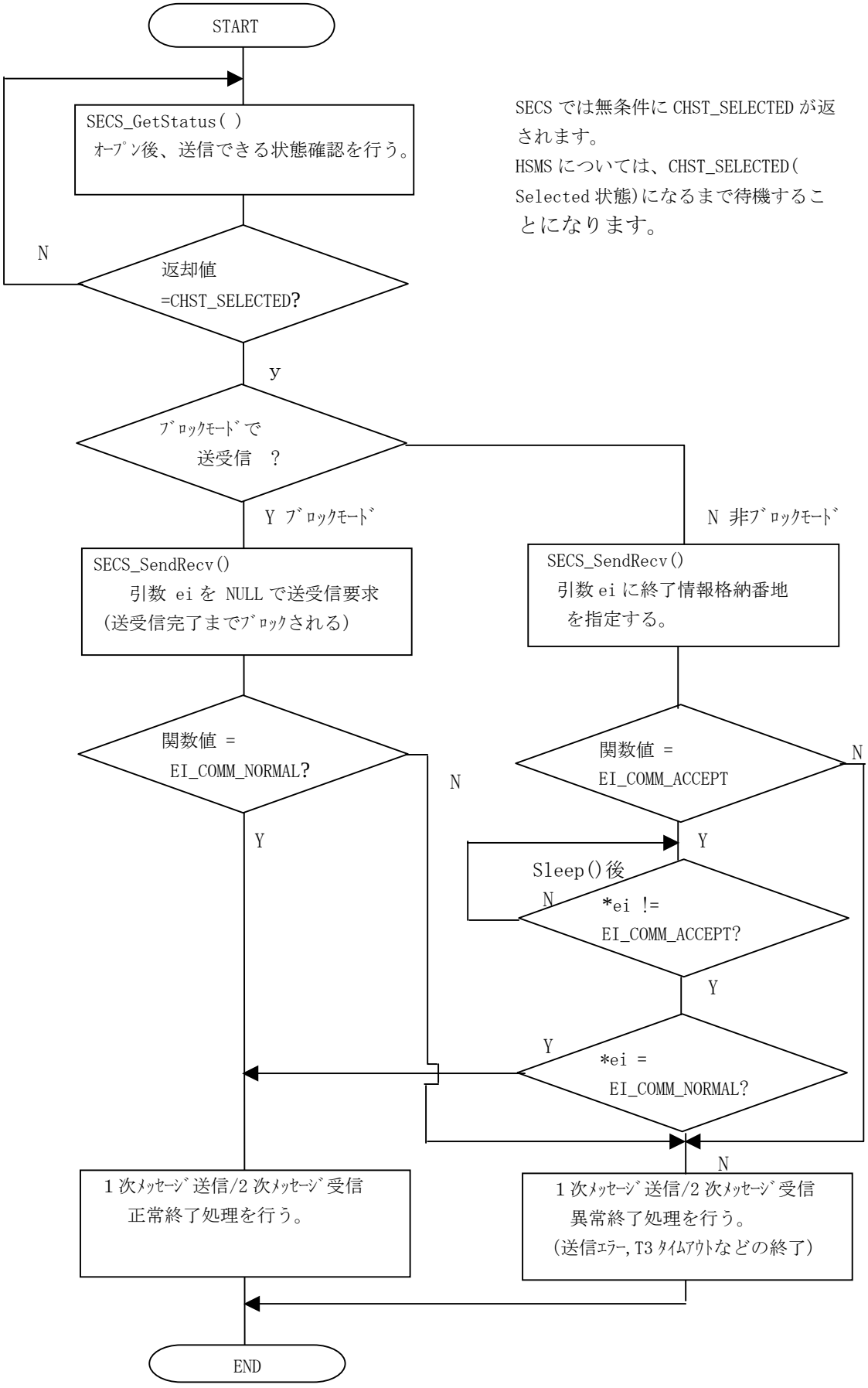


環境定義ﾌｧｲﾙ名を指定します。

ﾁｬﾝﾈﾙ毎にｵｰﾌﾟﾝします。

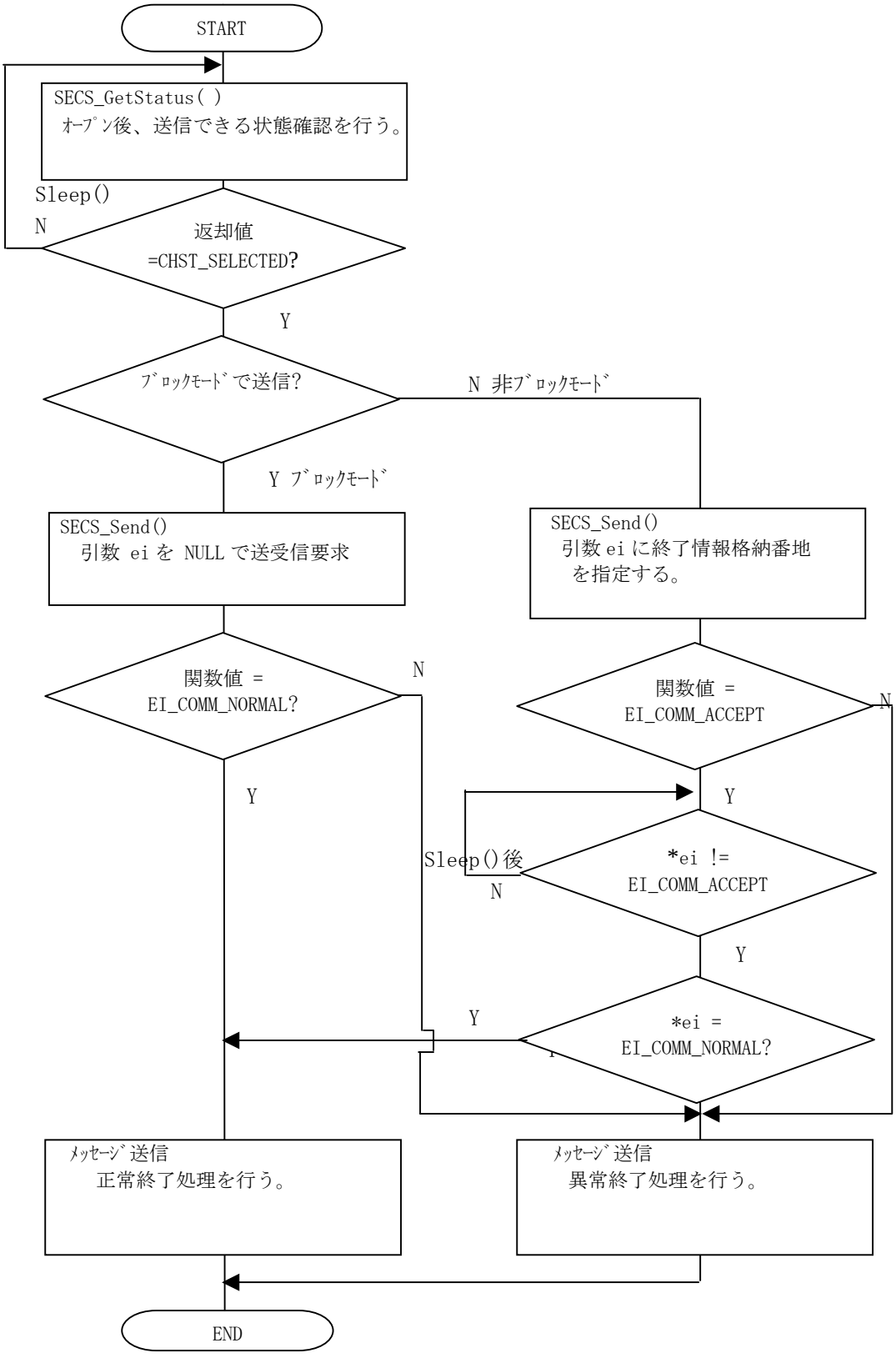
通信接続を完了するまで、必要に応じてこれらを繰り返します。

３．１．２　SECS_SendRecv() 1 次メッセージ(W=1)送信と応答 2 次メッセージ受信

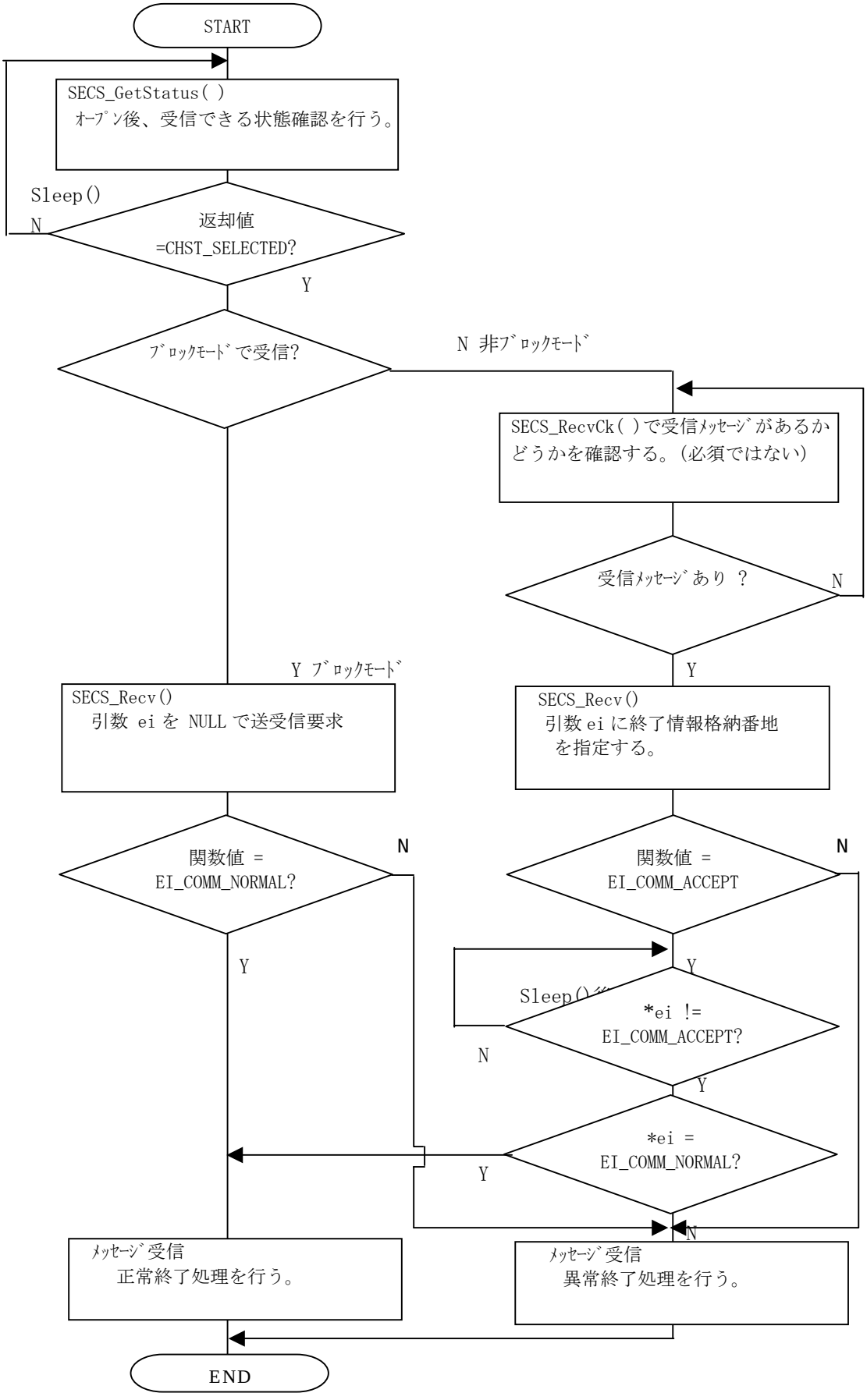


SECS では無条件に CHST_SELECTED が返されます。
HSMS については、CHST_SELECTED(Selected 状態)になるまで待機することになります。

３．１．３　SECS_Send()による１次または２次メッセージの送信



３．１．４　１次または２次メッセージの受信



３．２　ＳＥＣＳ通信制御関数

ＳＥＣＳ－I,ＨＳＭＳ-ＳＳプロトコル仕様に基づいて制御するためのドライバーＡＰＩ関数について説明します。

次の 10 個の通信制御関連 API 関数が提供されます。

SECS_Start( )	:	SECSﾄﾞﾗｲﾊﾞｰの環境準備と開始
SECS_Terminate( )	:	SECSﾄﾞﾗｲﾊﾞｰの終了
SECS_Open( )	:	ﾁｬﾝﾈﾙのｵｰﾌﾟﾝ
SECS_Close( )	:	ﾁｬﾝﾈﾙのｸﾛｰｽﾞ
SECS_GetStatus( )	:	通信ﾁｬﾝﾈﾙの送受信準備確認
SECS_SendRecv( )	:	１次ﾒｯｾｰｼﾞ送信後応答２次ﾒｯｾｰｼﾞの受信
SECS_Send( )	:	ﾒｯｾｰｼﾞの送信
SECS_RecvCk( )	:	受信ﾒｯｾｰｼﾞの有無の確認
SECS_Recv( )	:	ﾒｯｾｰｼﾞ受信
SECS_GetSerialNo( )	:	製品のシリアル番号の取得

３．２．１　SECS_Start( ) －　ＳＥＣＳドライバー開始関数

（１）　書式

```
#include "D_API.h"

int     SECS_Start( char *DefFile )

        DefFile ： 通信環境定義ﾌｧｲﾙ名へのﾎﾟｲﾝﾀ
```

（２）　返却情報

関数値	記　号	意　味
0	EI_COMM_NORMAL	正常に開始できた。
6	EI_COMM_BUSY	既に開始されている。
20	EI_COMM_OTHER_ERROR	その他のｴﾗｰ検出により開始できなかった。

（３）　機能

ＳＥＣＳドライバーを使用開始するための準備をします。
指定された通信環境定義ファイルには、「２．１　通信環境定義ファイル」で説明される定義コマンドを使用してアプリケーションが使用する通信チャンネルに対して、ポートならびに通信条件が定義されていなければなりません。

本関数の結果が正常終了（EI_COMM_NORMAL） でなかった場合には、その後に他のＡＰＩ関数が実行されても全てエラー終了になります。

アプリケーションは、通信チャンネルをオープンして使用する前に、必ず本関数を実行する必要があります。

アプリケーションが本ドライバーの使用中止または終了させる場合は、SECS_Terminate() 関数を実行する必要があります。

３．２．２　SECS_Terminate（ ）　－　ＳＥＣＳﾄﾞﾗｲﾊﾞｰの終了

（１）　書式

```
＃include　　“D_API.h”

void　　SECS_Terminate( void )
```

（２）　返却情報

なし。

（３）　機能

ドライバーの使用を終了させます。

ドライバー内の全チャンネルが Close されていれば、ドライバーが使用している資源をシステムに返却します。

もし、Open 中のチャンネルが残っている場合は、 それらチャンネルを強制的に Close させた上で使用終了させ、システムの資源を返却します。

３．２．３ SECS_Open( ) － ＳＥＣＳ/ＨＳＭＳのチャンネルのオープン

（１） 書式

＃include “D_API.h”

int SECS_Open(int chno)

chno ： ｵｰﾌﾟﾝ対象ﾁｬﾝﾈﾙ番号 (1～10)

（２） 返却情報

関数値	記 号	意 味
0	EI_COMM_NORMAL	正常にオープンできた。
1	EI_COMM_OPEN_ERROR	オープンできなかった。
2	EI_COMM_CH_ERROR	CHNO の値が正しくない。
6	EI_COMM_OPENED_ALREADY	既にオープンされている。

（３） 機能

chno に指定された,チャンネルをオープン（送受信可能）状態にします。

チャンネルの通信条件は SECS_Start() の通信環境定義ファイルに設定された内容に従ってオープンされます。

オープンが正常に実行された後、 SECS についてはただちに送受信可能状態になり、HSMS については、TCP/IP 接続、そして、Select 確立がなされた後にメッセージの送受信が可能になります。

なお、メッセージの送受信が可能かどうかの確認は後述の SECS_GetStatus() 関数で確認し、メッセージの送受信を行います。
（SECS については、SECS_GetStatus()の応答は常時、送受信可能である応答が返されます。）

（４） 特記事項

HSMS プロトコルの場合、本 API で一旦オープンされますと、ドライバー内部で自動的に相手装置とのＴＣＰ／ＩＰの接続そして、Select 確立のための処理を行います。また、一旦、Select が確立したあと、何らかの通信の事情で HSMS 接続が切れた場合も、ドライバーは内部で自動的に接続と Select 確立の復帰処理を行います。
HSMS のテキスト送受信は Select 確立が行われないと実行できませんので、アプリケーション側で Select 確立されているかどうかを、先に述べた SECS_GetStatus()関数で確認した上で送受信を行うことをお勧めします。

３．２．４ SECS_Close（） － ＳＥＣＳのクローズ

（１） 書式

＃include “D_API.h”

int SECS_Close(int chno)

chno ： ｸﾛｰｽﾞ対象ﾁｬﾝﾈﾙ番号 （ 1～10 ）

（２） 返却情報

関数値	記 号	意 味
0	EI_COMM_NORMAL	クローズできた。
2	EI_COMM_CH_ERROR	chno の値が正しくない。

（３） 機能

２．３でオープンされたチャンネル chno をクローズします。

オープンされていなかった場合も、正常終了結果が返却されます。

対象が HSMS プロトコルチャンネルの場合、もし、通信状態が SELECTED 状態であれば、Seaparate.Req 制御メッセージを送信した上で、TCP/IP の接続を切ります。

３．２．５　SECS_GetStatus（）　－　通信ﾁｬﾝﾈﾙの送受信準備確認

（１）　書式

＃include　“D_API.h”

int　　SECS_GetStatus(int chno)

chno　　：受信対象ﾁｬﾝﾈﾙ番号（ 1～10 ）

（２）　返却情報

関数値	記　号	意　味
0	CHST_NOT_OPEN	ﾁｬﾝﾈﾙが定義されていないか、ｵｰﾌﾟﾝされていない。
1	CHST_CONNECT_WAIT	接続待ち中である。
2	CHST_SELECT_WAIT	Selection 確立待ち中である。
3	CHST_SELECTED	Selection 確立済みで送受信可能状態である。

（３）　機能

chno で指定されたチャンネルが、送受信できる状態にあるかどうかを確認するため、本関数を使用して状態情報を取得します。

メッセージを送受信できる状態は、CHST_SELECTED 状態時だけです。

ＳＥＣＳ(シリアル通信）の場合には、オープンが正常に行われた後、即時、送受信可能であるため、正常にオープンできたあとは、常時、CHST_SELECTED が返却されます。

３．２．６　SECS_SendRecv( )　－　応答待ち１次ﾒｯｾｰｼﾞ送信と２次ﾒｯｾｰｼﾞ受信

（１）　書式

```
#include  “D_API.h”

int       SECS_SendRecv( int chno, UCHAR *smsg, int slen, UCHAR *rmsg, int rsize,
                                                          int *rlen,  int *ei )
```

chno　：送信対象 channel no.（ 1～10 ）
smsg　：送信１次ﾒｯｾｰｼﾞﾃﾞｰﾀのﾎﾟｲﾝﾀ
slen　：送信１次ﾒｯｾｰｼﾞ smsg のﾃﾞｰﾀﾊﾞｲﾄ長（ﾍﾀﾞｰ，ﾃﾞｰﾀを含む）
rmsg　：受信２次ﾒｯｾｰｼﾞ格納ﾊﾞｯﾌｧﾎﾟｲﾝﾀ
rsize　：受信２次ﾒｯｾｰｼﾞ格納ﾊﾞｯﾌｧのﾊﾞｲﾄｻｲｽﾞ
rlen　：実際に受信した２次ﾒｯｾｰｼﾞのﾊﾞｲﾄ長（ﾍﾀﾞｰ，ﾃﾞｰﾀを含む）
ei　：非ﾌﾞﾛｯｸﾓｰﾄﾞ時の終了情報の格納番地(=NULL ならばﾌﾞﾛｯｸﾓｰﾄﾞ)

（２）　結果情報

関数値/終了情報	記　号	意　味	
(-1)	EI_COMM_BUSY	要求が受け入れられた	非ﾌﾞﾛｯｸﾓｰﾄﾞ時のみ
0	EI_COMM_NORMAL	正常に送受信が終了した。	
1	EI_COMM_OPEN_ERROR	ｵｰﾌﾟﾝされていない。	
2	EI_COMM_CH_ERROR	CHNO の値が正しくない。	
3	EI_COMM_SEND_ERROR	送信ｴﾗｰを検出した。	
4	EI_COMM_RECV_ERROR	受信ｴﾗｰを検出した。	
13	EI_COMM_T3_TIMEOUT	T3 ﾀｲﾑｱｳﾄを検出した。	

（３）　機能

指定した chno のチャンネルへ smsg に格納されている W-bit = 1, 長さ=slen の SECS メッセージを送信し、相手装置からの２次メッセージを受信し，rmsg で指定されるバッファに格納します。受信したメッセージのバイト長は rlen が指す領域に格納されます。

ポインタ ei の値は、ブロック/非ブロックモードの指定を兼ねており、本関数の結果が次のように返却されます。

a．　ﾌﾞﾛｯｸﾓｰﾄﾞ(ei = NULL)の場合、関数の返却値に結果が返却される。
b．　非ﾌﾞﾛｯｸﾓｰﾄﾞ(ei != NULL)の場合、ei が指す領域に結果が格納される。
　　（但し、(-1)，1 または２の場合は、関数値として返却されます。）

応答２次メッセージを T3 時間以内に受信できなかった場合、エラー情報として EI_COMM_T3_TIMEOUT が返却されます。

（４）　特記事項

本関数はドライバーに T3 タイムアウトの監視を任せる場合に使用できる関数です。
T3 タイムアウトの監視を上位側（ドライバーに対して）で行う場合は、SECS_SendRecv() 関数の代わりに SECS_Send()関数を使用することになります。

送信メッセージのシステムバイトの中のトランザクション ID(ｼｽﾃﾑﾊﾞｲﾄの 3,4 番目のﾊﾞｲﾄ) は、ドライバー側で設定した上で送信します。(ソースＩＤは呼び出し元がセットしなければなりません。)

また、デバイス ID は、通信環境定義ファイルに指定されているものをセットした上でメッセージを送信します。

ＳＥＣＳ通信の場合、デバイス ID の R ビットのセットは通信環境定義ファイルの指定に従ってドライバーが行います。

ＳＥＣＳ通信の場合、マルチブロックメッセージの場合における送信時のブロック単位の送信制御は本ドライバーが行います。したがって、アプリケーション側は、ＳＥＣＳメッセージをマルチブロックに分割する必要はありません。

３．２．７　SECS_Send( ) －　メッセージ送信

（１）　書式

```
＃include　“D_API.h”

int　　SECS_Send ( int chno, UCHAR *smsg, int slen, int *ei )
```

chno　　：送信対象 channel no. (1～10)
smsg　　：送信１次または２次ﾒｯｾｰｼﾞﾃﾞｰﾀのﾎﾟｲﾝﾀ
slen　　：送信ﾒｯｾｰｼﾞ smsg のﾃﾞｰﾀﾊﾞｲﾄ長 (ﾍﾀﾞｰ, ﾃﾞｰﾀを含む)
ei　　：非ﾌﾞﾛｯｸﾓｰﾄﾞ時の終了情報の格納番地(=NULL ならばﾌﾞﾛｯｸﾓｰﾄﾞ)

（２）　結果情報

関数値/終了情報	記　号	意　味	
(-1)	EI_COMM_BUSY	要求が受け入れられた	非ﾌﾞﾛｯｸﾓｰﾄﾞ時のみ
0	EI_COMM_NORMAL	正常に送受信が終了した。	
1	EI_COMM_OPEN_ERROR	ｵｰﾌﾟﾝされていない。	
2	EI_COMM_CH_ERROR	CHNO の値が正しくない。	
4	EI_COMM_SEND_ERROR	送信ｴﾗｰを検出した。	

（３）　機能

指定した chno のチャンネルへ smsg に格納されている slen の長さの SECS メッセージを送信します。
本関数は１次、２次メッセージの両方のメッセージの送信に使用することができます。

SECS_Send 関数の結果は次のように返却されます。
ａ．　ﾌﾞﾛｯｸﾓｰﾄﾞ(ei = NULL)の場合、関数の返却値に結果が返却されます。
ｂ．　非ﾌﾞﾛｯｸﾓｰﾄﾞ(ei != NULL)の場合、ei が指す領域に結果コードが格納されます。
（但し、(-1), 1 または２の場合は、関数値として返却されます。）

（４）　特記事項

１次メッセージのシステムバイトならびに、１，２次メッセージのデバイス ID については、SECS_SendRecv() と同様にドライバーが自動的に設定した上で送信します。

応答２次メッセージを期待する１次メッセージに対する応答メッセージ受信するまでの T3 タイムアウト監視をドライバーに任せたい場合には、本関数ではなく前述の SECS_SendRecv() 関数を使用することになります。

本関数を使用して応答を期待するメッセージを送信したあと、２次メッセージの受信は SECS_Recv()で受信することになります。受信するまでの T3 タイムアウト監視は上位（呼び出し側）で行わなければなりません。

３．２．８　SECS_RecvCk ()　－　受信済みメッセージ有無の確認

（１）　書式

```
＃include　“D_API.h”

int　　SECS_RecvCk( int chno, int　*rsize )

　　chno　　：受信対象ﾁｬﾝﾈﾙ番号（ 1～10 ）
　　rsize　　：受信メッセージの長さ格納ポインタ
　　　　（ =0 は、受信メッセージがなかったことを意味する。）
```

（２）　返却情報

関数値	意　　味
0	受信済みﾒｯｾｰｼﾞがない。
1	受信済みﾒｯｾｰｼﾞがある。

（３）　機能

chno で指定されたポートに受信された SECS メッセージがあるかどうかの確認を行います。

受信済みメッセージがない場合は、関数値として　= 0 が返却されます。

受信済みメッセージがある場合には、関数値として = 1 が返却され、同時に受信メッセージのバイト長が rsize が指す領域に設定されます。

受信メッセージの取得については、rsize に設定された以上の長さのバッファを準備し、SECS_Recv()を使用することになります。

３．２．９　SECS_Recv ()　－　受信メッセージの取得

（１）　書式

＃include　“D_API.h”

int　　SECS_Recv(int chno, void *rbuff,　int　rsize, int *rlen, int *ei)

chno　　：受信対象ﾁｬﾝﾈﾙ番号（ 1～10 ）
rbuff　　：受信した SECS メッセージを格納するバッファポインタ
rsize　　：準備された受信バッファのバイトサイズ
rlen　　：受信メッセージの長さ格納ポインタ
（ =0 は、受信メッセージがなかったことを意味する。）
ei　　：受信結果情報を格納するポインタ(=NULL ならばﾌﾞﾛｯｸﾓｰﾄﾞ)

（２）　結果情報

関数値/終了情報	記　号	意　味	
(-1)	EI_COMM_BUSY	要求が受け入れられた	非ﾌﾞﾛｯｸﾓｰﾄﾞ時のみ
0	EI_COMM_NORMAL	正常に送受信が終了した。	
2	EI_COMM_OPEN_ERROR	ｵｰﾌﾟﾝされていない。	
3	EI_COMM_CH_ERROR	CHNO の値が正しくない。	
5	EI_COMM_RECV_ERROR	受信ｴﾗｰを検出した。	
21	EI_COMM_RCV_BUFF_OVERFLOW	受信ﾒｯｾｰｼﾞ長が受信ﾊﾞｯﾌｧより大きい。	

（３）　機能

chno で指定されたポートで受信した SECS メッセージを rbuff 領域に取得する。受信されたメッセージのバイト長は　rlen が指定する領域に設定返却されます。

なお、rsize で指定されたバイト長を超えるメッセージが受信されていた場合は、終了情報として EI_COMM_RCV_BUFF_OVERFLOW が返却されます。この場合、rlen に取得されるべきメッセージのバイト長が返却されますので、その長さに充分な受信バッファを準備し直した上で、SECS_Recv()関数を再実行することによって受信メッセージを取得することができます。

ei の値は、ブロック/非ブロックモードの指定を兼ねており、本関数の結果が次のように返却されます。

ａ．　ﾌﾞﾛｯｸﾓｰﾄﾞ(ei = NULL)の場合、関数の返却値に結果が返却されます。
ｂ．　非ﾌﾞﾛｯｸﾓｰﾄﾞ(ei != NULL)の場合、ei が指す領域に結果が格納されます。

（４）　特記事項

ドライバーが 2 次メッセージを受信していて、本関数が実行された場合、そのメッセージに対する１次メッセージが SECS_SendRecv()関数で送信されたものに対する応答メッセージでない場合ものについてのみ本 SECS_Recv()関数がその２次メッセージを取得することになります。

３．２．10　SECS_GetSerialNo（ ）　－　ＳＥＣＳﾄﾞﾗｲﾊﾞｰの製品情報の取得

（１）　書式

＃include　　“D_API.h”

void　　SECS_GetSerialNo(char *snbuff)

snbuff ： 製品名とシリアル番号の製品情報を格納するバッファアドレス
(160 バイト分のバッファが必要です。)

（２）　返却情報

なし。

（３）　機能

ドライバーの製品名、シリアル番号の製品情報を取得します。

例えば、次のように情報が取得されます。

“DSHDVR-2002-1 SECS/HSMS Level-1 Driver S/N : 1043824564”
“Copyright (C) 2001-2003 Data Map Corporation. All rights reserved.”

４．　通信メッセージ生成、データアイテム取得関連 API 関数

レビジョン-３では、SECS-II 通信メッセージをプログラミングによって作成するための関数、ならびに受信したメッセージからデータアイテムを順次取り出すための関数を新たに追加しました。

（１）　関連する関数は以下のとおりです。

番号	関数名	機能概略
1.	SECS_set_msg_head()	作成するﾒｯｾｰｼﾞのﾍﾀﾞｰ情報を設定する。
2.	SECS_add_msg_item()	ﾒｯｾｰｼﾞにデータアイテム情報を加える。
3.	SECS_get_item_init()	ﾒｯｾｰｼﾞ取得情報の初期ｾｯﾄｱｯﾌﾟ
4.	SECS_get_msg_head()	ﾒｯｾｰｼﾞﾍﾀﾞｰ情報を取得する。
5.	SECS_get_item_code_size()	ﾒｯｾｰｼﾞｱｲﾃﾑｺｰﾄﾞとｻｲｽﾞを取得する。
6.	SECS_get_data_item()	指定ｱｲﾃﾑｺｰﾄﾞのﾃﾞｰﾀを取得する。
7.	SECS_get_item_any_data()	ﾃﾞｰﾀｱｲﾃﾑを取得する。(ｱｲﾃﾑｺｰﾄﾞ指定なし)
8.	SECS_get_item_unit_size()	指定ｱｲﾃﾑｺｰﾄﾞの単位ﾊﾞｲﾄ長を取得する。

（２）　メッセージ操作構造体

SECS_get_item_unit_size()以外の関数には、ドライバー内部のメッセージ操作処理用として以下の TMSG_DEF 領域ををユーザ側で準備し、関数の引数として与える必要があります。TMSG_DEF 構造体内部情報はドライバーが操作用に使用しますので、ユーザ側では値を変更しないでください。

```
typedef struct{
        UCHAR   s,f,w;
        UCHAR   sybt[4];
        UCHAR   *text;
        int     max_len;
        int     len;
        int     next;
        int     item_qty;
        UCHAR    temp[8];
        } TMSG_DEF;
```

（３）　アイテムの記号

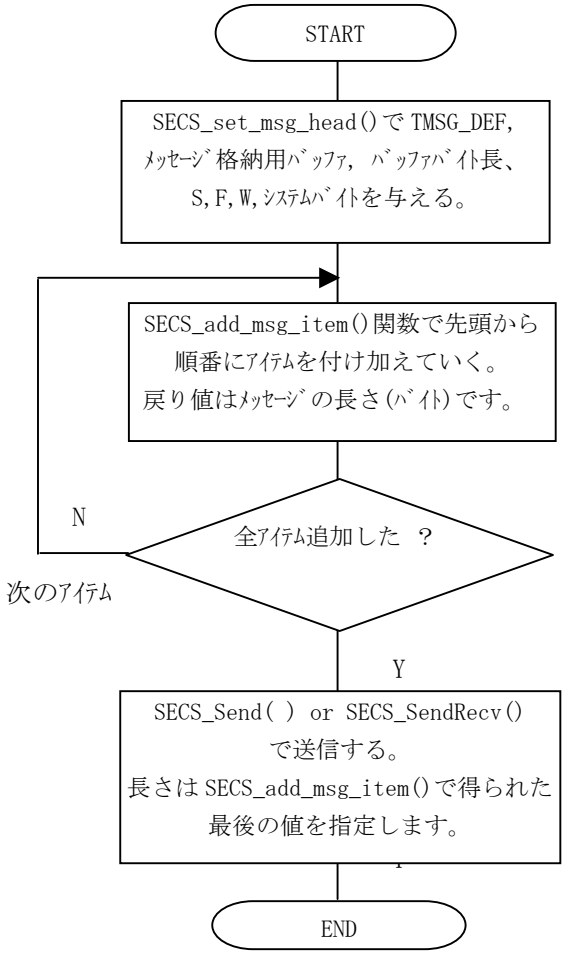
アイテムコードに関連する関数が使用するアイテム記号は下表の記号または値を使用します。

番号	記　号	意　味	16 進表現(*1)
1.	I_L	ﾘｽﾄ（要素の長さ）	00
2.	I_B	2 進	20
3.	I_BOOL または I_T	真理値	24
4.	I_A	アスキー	40
5.	I_J	JIS-8	44
6.	I_S8	8ﾊﾞｲﾄ整数(符号付き)	60
7.	I_S1	1ﾊﾞｲﾄ整数(符号付き)	64
8.	I_S2	2ﾊﾞｲﾄ整数(符号付き)	68
9.	I_S4	4ﾊﾞｲﾄ整数(符号付き)	70
10.	I_D	8ﾊﾞｲﾄ浮動小数点	80
11.	I_F	4ﾊﾞｲﾄ浮動小数点	90
12.	I_U8	8ﾊﾞｲﾄ整数(符号無し)	A0
13.	I_U1	1ﾊﾞｲﾄ整数(符号無し)	A4
14.	I_U2	2ﾊﾞｲﾄ整数(符号無し)	A8
15.	I_U4	4ﾊﾞｲﾄ整数(符号無し)	B0

（注-*1）　16 進表現は、アイテムコード 8 ビットをそのまま 16 進で表現したものです。
（下位 2ﾋﾞｯﾄは 00 です。）

４．１　フローチャート

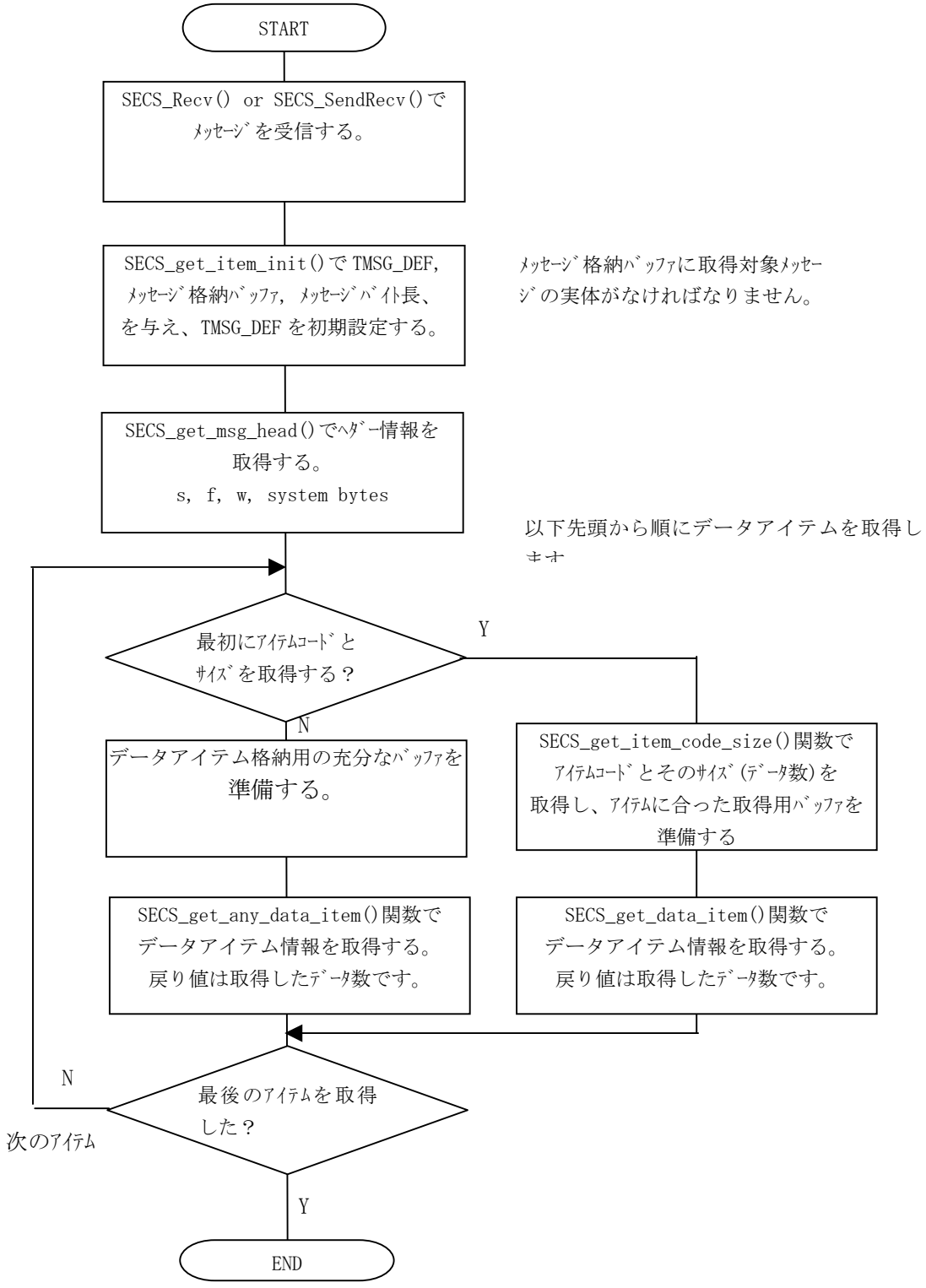
４．１．１　ＳＥＣＳメッセージ作成フロー



ｼｽﾃﾑﾊﾞｲﾄは 2 次応答ﾒｯｾｰｼﾞ作成時だけ必要です。
ﾒｯｾｰｼﾞはﾒｯｾｰｼﾞ格納用ﾊﾞｯﾌｧ内に組み立てられます。

全てのｱｲﾃﾑを追加するまで、繰り返します。
得られるﾒｯｾｰｼﾞ長は、ﾍﾀﾞｰを含めた長さです。

４．１．２　ＳＥＣＳメッセージ情報取得フロー



４．２　メッセージ作成およびデータアイテム取得関数

４．２．１　SECS_set_msg_head ()　－　メッセージヘダーを設定する

（１）　書式

```
#include “D_API.h”

int     SECS_set_msg_head( TMSG_DEF *mdef, UCHAR *msg_ptr, int max_len,
                                    int s, int f, int w, UCHAR *sybt )
```

mdef　　：メッセージ操作構造体のポインタ
msg_ptr ：メッセージを組立て保存するためのバッファのポインタ
max_len ：作成できるメッセージの最大バイト長(ヘダー部も含む)
　　msg_ptr 用バッファのサイズは max_len 以上必要です。
s　　：Stream ｺｰﾄﾞ(1～127)
f　　：Function ｺｰﾄﾞ(0～255)
w　　：Wait bit(0 or 1) 値が 1 の場合 2 次メッセージ期待を意味する。
sybt　　：メッセージに設定するシステムバイト(4 ﾊﾞｲﾄ)
　　2 次メッセージ応答時に（受信した 1 次メッセージのもの）必要。

（２）　結果情報
正常の場合、設定されたメッセージバイト長が返却されます。

関数値/終了情報	記　号	意　味	
(-30) 10	EI_DEF_MSG_ERR (なし)	max_len が 10 未満であった。 現ﾒｯｾｰｼﾞﾊﾞｲﾄ長	

（３）　機能
与えられたメッセージバッファに、指定されたメッセージヘダー情報を設定します。
設定する情報は、Stream, Function, W-bit です。また、Function コードが偶数(2 次メッセージ)の場合には、sybt に与えられたシステムバイトも設定します。
また、その後、アイテム情報を SECS_add_msg_item()関数で付け加えられるようにするために mdef で指定される TMSG_DEF 領域を初期化します。

（４）　特記事項
後述の SECS_add_msg_item()を実行する前に、本関数の実行を行う必要があります。

４．２．２　SECS_add_msg_item ()　－　メッセージアイテムを追加する

（１）　書式

＃include　“D_API.h”

int　　SECS_add_msg_item(TMSG_DEF *mdef, int item_code, int dlen, void *val)

mdef　　：メッセージ操作構造体のポインタ
　　(SECS_set_msg_head()で指定したものを指定する。)
item_code：加えたいアイテムのアイテムコード
dlen　　：データアイテム数（データ単位の配列サイズ）
　　LIST アイテムの場合は、リストサイズを指定する。
val　　：データアイテムが格納されている領域のポインタ
　　LIST アイテムの場合は、NULL(=0) でよい。

（２）　結果情報
正常の場合、アイテムが追加された後のメッセージのバイト長が返却される。

関数値/終了情報	記　号	意　味	
(-31)	EI_DEF_MSG_LEN_ERR	ﾒｯｾｰｼﾞ長が指定最大長を超えた。	
(-32)	EI_ITEM_CODE_ERR	指定ｱｲﾃﾑｺｰﾄﾞが正しくない。	
＞ 10		追加後のﾒｯｾｰｼﾞﾊﾞｲﾄ長	

（３）　機能
与えられたアイテムコードと長さに従って、メッセージ内にアイテム情報を追加します。
メッセージ用バッファとバッファサイズは、SECS_set_msg_head()関数で指定されたものを使用します。もし、バッファサイズを超えるアイテム情報を追加しようとした場合、エラーになります。

データアイテムの単位長が、2 バイト以上のものについて、バイト順位変換処理は本関数が行います。(val(Little Endian) ==> SECS ﾒｯｾｰｼﾞ(Big Endian))

LIST アイテムの場合、val は使用しません。

（４）　特記事項

LIST アイテムの場合、それに続くアイテム数を指定しますが、結果として、指定されたリスト数分のアイテムが設定されたかどうかについてはドライバーは何らチェックを行いません。ユーザの責任でアイテム数を合わせる必要があります。

データアイテム設定位置は、本関数によって自動的に次に進められます。

４．２．３　SECS_get_msg_init（）　－　メッセージ情報取得の初期化

（１）　書式

```
＃include　“D_API.h”

int      SECS_get_msg_init( TMSG_DEF *mdef, UCHAR *msg_ptr, int msg_len )

         mdef     ： メッセージ操作構造体のポインタ
         msg_ptr  ： 取得対象メッセージ格納ポインタ
         msg_len  ： メッセージのバイト長
```

（２）　結果情報

正常の場合、０が返却される。

関数値/終了情報	記　号	意　味	
(-30)	EI_DEF_MSG_ERR	ﾒｯｾｰｼﾞ長が 10 ﾊﾞｲﾄ未満である。	
0		正常に初期化できた。	

（３）　機能

受信（または送信）した SECS-II メッセージの構成情報を取得するための準備（初期設定）を行うための関数です

（４）　特記事項

後述の SECS_get_msg_head(), SECS_get_item_code_size(), SECS_get_data_item() ならびに SECS_get_data_item_any()関数を使用する前に、本関数を実行しなければなりません。

４．２．４　SECS_get_msg_head ()　－　メッセージヘダー情報を取得する

（１）　書式

```
＃include　“D_API.h”

int      SECS_get_msg_head( TMSG_DEF *mdef, int *s, int *f, int *w, UCHAR *sybt )
```

mdef　　：メッセージ操作構造体のポインタ
　　(SECS_get_msg_init()で指定したものを指定する。)
s　　：Stream コード格納ポインタ
f　　：Function コード格納ポインタ
w　　：Wait bit(0 or 1) 格納ポインタ
sybt　　：システムバイト(４バイト)格納ポインタ
　　w=1 の１次メッセージに対する応答メッセージ用にのみ必要。

（２）　結果情報

正常の場合、取得された Stream コードが返却される。

（３）　機能

SECS_get_msg_init()で指定されたメッセージから、ヘダー情報を取得します。
取得する情報は、Stream, Function, W-bit です。また、sybt にはメッセージが有するシステムバイトを取得します。

４．２．５　SECS_get_item_code_size（）　－　アイテムコードとサイズ情報を取得する

（１）　書式

```
＃include　“D_API.h”

int      SECS_get_item_code_size( TMSG_DEF *mdef, int *item_code, int *size )

         mdef     ： メッセージ操作構造体のポインタ
                    (SECS_get_msg_init()で指定したものを指定する。)
         item_code： アイテムコード格納ポインタ
         size     ： データアイテム数格納ポインタ
                     LIST アイテムの場合は、リストサイズ
```

（２）　結果情報

正常の場合、アイテムコードが返却される。

関数値/終了情報	記　号	意　味	
(-31)	EI_DEF_MSG_LEN_ERR	ｱｲﾃﾑが存在しない。	
(-32)	EI_ITEM_CODE_ERR	ｱｲﾃﾑｺｰﾄﾞが正しくなかった。	
>=0		ｱｲﾃﾑｺｰﾄﾞ(I_L,I_B,...)	

（３）　機能

次に取得できるアイテムのアイテムコードとそのサイズを取得します。
本関数は、メッセージのリスト構造が不明のときに、データアイテムを取り出す前にバッファの準備などのためにアイテムコードとその大きさを知るために使用します。

４．２．６　SECS_get_data_item ()　－　データアイテムを取得する

（１）　書式

```
#include “D_API.h”

int     SECS_get_data_item( TMSG_DEF *mdef, int item_code, int size, void *val )
```

mdef　　：メッセージ操作構造体のポインタ
　　(SECS_set_msg_head()で指定したものを指定する。)
item_code：取得したいアイテムコード
size　　：データアイテム数（データ単位配列サイズ）
　　LIST アイテムの場合は、リストサイズ
val　　：取得データアイテムを格納する領域のポインタ
　　LIST アイテムの場合は、NULL(=0) でよい。

（２）　結果情報
正常の場合、取得したデータ数が返却される。

関数値/終了情報	記　号	意　味	
(-1)	EI_NO_MORE_ITEM	ﾃﾞｰﾀｱｲﾃﾑが無い	
(-32)	EI_ITEM_CODE_ERR	ｱｲﾃﾑｺｰﾄﾞが一致しないか、正しくない。	
(-33)	EI_ITEM_SIZE_ERR	ｱｲﾃﾑｻｲｽﾞが指定ｻｲｽﾞより大きい。	
>= 0	ﾃﾞｰﾀ数	取得できたﾃﾞｰﾀ数	

（３）　機能
item_code と size で指定されたデータアイテムを val で指定した領域に取得します。
正しく取得できる条件は、アイテムコードが item_code と一致し、また、データアイテムサイズが size と等しいか、または少ない場合です。

データアイテムの単位長が、2 バイト以上のものについて、バイト順位変換処理は本関数が行います。(val(Little Endian) <== SECS ﾒｯｾｰｼﾞ(Big Endian))

LIST アイテムの場合、val は使用しません。

（４）　特記事項

データアイテム取得位置は、本関数によって自動的に次に進められます。

４．２．７　SECS_get_any_data_item（）　－　アイテムコード指定なしでデータを取得する

（１）　書式

＃include　“D_API.h”

int　　SECS_get_any_data_item(TMSG_DEF *mdef, int *item_code, int *size, void *val)

mdef　　：メッセージ操作構造体のポインタ
　　(SECS_set_msg_head()で指定したものを指定する。)
item_code：取得したアイテムコード格納ポインタ
size　　：データアイテム数（データ単位配列サイズ）格納ポインタ
　　LIST アイテムの場合は、リストサイズ
val　　：取得データアイテムを格納する領域のポインタ
　　(LIST ｱｲﾃﾑの場合は使用しません。)

（２）　結果情報
正常の場合、取得したアイテムコードが返却される。

関数値/終了情報	記　号	意　味	
(-1)	EI_NO_MORE_ITEM	ﾃﾞｰﾀｱｲﾃﾑが無い	
(-32)	EI_ITEM_CODE_ERR	ｱｲﾃﾑｺｰﾄﾞが正しくない。	
>= 0	(取得ｱｲﾃﾑｺｰﾄﾞ)	取得ｱｲﾃﾑｺｰﾄﾞ	

（３）　機能
カレント取得位置からデータアイテムを取得します。
先の SECS_get_data_item()関数と違うところは、アイテムコードとサイズ予め知らない状態でデータアイテム情報を取得するための関数です。取得したデータアイテムのコードとサイズは、指定された item_code, size 領域に格納されます。

データアイテムの単位長が、2 バイト以上のものについて、バイト順位変換処理は本関数が行います。(val(Little Endian) <== SECS ﾒｯｾｰｼﾞ(Big Endian))

LIST アイテムの場合、val は使用しません。

（４）　特記事項

データアイテム取得位置は、本関数によって自動的に次に進められます。

４．２．８　SECS_get_item_unit_size（）　－　アイテムコードの単位バイト長を取得する

（１）　書式

＃include　“D_API.h”

int　　SECS_get_item_unit_size(int item_code)

item_code：取得したいアイテムコード（4.　（３）アイテムコード表参照）

（２）　結果情報

正常の場合、取得したアイテムコードの単位バイトサイズが返却される。

関数値/終了情報	記　号	意　味	
(-31)	EI_ITEM_CODE_ERR	ｱｲﾃﾑｺｰﾄﾞが正しくない。	
>= 0	単位ﾊﾞｲﾄｻｲｽﾞ	ｱｲﾃﾑｺｰﾄﾞの単位ﾊﾞｲﾄｻｲｽﾞ	

（３）　機能

指定されたアイテムコードのデータ１個分のバイトサイズを取得します。

I_LIST（リストアイテム）の場合、１が返却されます。

５．　その他の関数

本ドライバーが提供するその他の関数について説明します。これらの関数は、SECS,HSMS の通信には、直接関係はありません。データアイテム取得した後、値を文字列表現するための変換関数が主です。

他に、ログファイルにアプリケーションプログラムからログメッセージを書込むための関数も準備されています。

５．１　整数データ符号付き１０進表現への変換

８、１６、３２または６４ビット符号付整数データを１０進文字列に変換します。

以下の関数を使用できます。

（1）　void SECS_int8_to_dec(char *d, char *buff, int n)
（2）　void SECS_int16_to_dec(short *d, char *buff, int n)
（3）　void SECS_int32_to_dec(int *d, char *buff, int n)
（4）　void SECS_int64_to_dec(__int64 *d, char *buff, int n)

関数の引数は次のとおりです。

d　　：符号付きﾃﾞｰﾀが格納されている領域のﾎﾟｲﾝﾀ
buff ：10 表現結果文字列を格納するﾊﾞｯﾌｧﾎﾟｲﾝﾀ
n　　：変換するﾃﾞｰﾀ数

データが複数ある場合(n >= 2)、データとデータの間をカンマ (,) で区切ります。
符号は、負の場合だけ数値文字列の頭に付加します。

５．２　整数データ符号なし１０進表現への変換

８、１６、３２または６４ビット符号なし整数データを１０進文字列に変換します。

以下の関数を使用できます。

（1）　void SECS_uint8_to_dec(UCHAR *d, char *buff, int n)
（2）　void SECS_uint16_to_dec(USHORT *d, char *buff, int n)
（3）　void SECS_uint32_to_dec(UINT *d, char *buff, int n)
（4）　void SECS_uint64_to_dec(__int64 *d, char *buff, int n)

(注) UCHAR は unsigned char
USHORT は unsigned short
UINT は、 unsigned int を意味します。

関数の引数は次のとおりです。

d　　：符号なしﾃﾞｰﾀが格納されている領域のﾎﾟｲﾝﾀ
buff ：10 表現結果文字列を格納するﾊﾞｯﾌｧﾎﾟｲﾝﾀ
n　　：変換するﾃﾞｰﾀ数

データが複数ある場合(n >= 2)、データとデータの間をカンマ (,) で区切ります。

５．３　数値データ１６進表現への変換

８、１６、３２または６４ビット整数データを１６進文字列に変換します。
以下の関数を使用できます。

（1）　void SECS_int8_to_hex(char *d, char *buff, int n)
（2）　void SECS_int16_to_hex(short *d, char *buff, int n)
（3）　void SECS_int32_to_hex(int *d, char *buff, int n)
（4）　void SECS_int64_to_hex(__int64 *d, char *buff, int n)

関数の引数は次のとおりです。
d　　：整数ﾃﾞｰﾀが格納されている領域のﾎﾟｲﾝﾀ
buff ：16 表現結果文字列を格納するﾊﾞｯﾌｧﾎﾟｲﾝﾀ
n　　：変換するﾃﾞｰﾀ数

数値文字列の前に 16 進を表すために、"0x" を付加します。
データが複数ある場合(n >= 2)、データとデータの間をカンマ（,）で区切ります。

５．４　浮動小数点データ１０進表現への変換

３２または６４ビット浮動小数点データを１０進文字列に変換します。
以下の関数を使用できます。

（1）　void SECS_f32_to_dec(float *d, char *buff, int n)
（2）　void SECS_f64_to_dec(double *d, char *buff, int n)

関数の引数は次のとおりです。
d　　：浮動小数点ﾃﾞｰﾀが格納されている領域のﾎﾟｲﾝﾀ
buff ：10 表現結果文字列を格納するﾊﾞｯﾌｧﾎﾟｲﾝﾀ
n　　：変換するﾃﾞｰﾀ数

データが複数ある場合(n >= 2)、データとデータの間をカンマ（,）で区切ります。

５．５　文字列の合成

２つの文字列を合成します。

（1）　void SECS_str_merge(char *s1, char *s2)

関数の引数は次のとおりです。
s1　：第 1 文字列のﾎﾟｲﾝﾀ
s2　：第 2 文字列のﾎﾟｲﾝﾀ

s2 文字列が、s1 文字列に合成されます。

５．６　日付・時刻取得関数

現在の日付・時刻を文字列で取得します。

（１）　void SECS_FormatDateTimeExt(char *buff, char *fmt)

関数の引数は次のとおりです。
buff ： 結果文字列格納領域ﾎﾟｲﾝﾀ
fmt　： 取得する日付・時刻書式指定文字列

書式 fmt は下表の文字を使用します。

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。 （最大 2 桁）
D	日	1 文字が日 1 桁分を指定します。 （最大 2 桁）
H	時	1 文字が時 1 桁分を指定します。 （最大 2 桁）
N	分	1 文字が分 1 桁分を指定します。 （最大 2 桁）
S	秒	1 文字が秒 1 桁分を指定します。 （最大 2 桁）
C	1/100 秒	1 文字が 1/100 秒の 1 桁を指定します。 （最大 2 桁）

例　fmt = “現在日付時刻 = YYYY-MM-DD HH:NN:SS.CC” と指定します。
現在日付時刻が　20004 年 12 月 24 日 13:15:24.67 の場合、buff には、次のような文字列が表示されます。

“現在日付時刻 = 2004-12-24 13:15:24.67”

５．７　E_printf( )　ー　情報の文字列への編集とログファイルへの書込み関数

（１）　書式

＃include　“D_API.h”

void　　E_printf(char *format, ...)

（２）　返却情報

なし。

（３）　機能

本関数は、ログファイルにアプリケーションプログラムから文字列メッセージを書込むための関数です。

引数 format（書式指定）は printf()関数と同じものが使用できます。
format の仕様については、Microsoft-C の関連文書を参照ください。

本関数は、format の指定に従って結果を文字列に編集した後、ログファイルへログ情報として書き込みます。

５．８　E_hex_dump()　－　メモリ領域１６進ダンプ情報のログファイルへの書込み関数

（１）　書式

＃include　“D_API.h”

void　　E_hex_dump(char *title, UCHAR *mem, int len)

（２）　返却情報

なし。

（３）　機能

本関数は、ログファイルへアプリケーションメモリの内容を１６進形式でログファイルに書込むための関数でしす。

本関数は、title に指定された文字列をまず、書込み、さらに mem で指定されたメモリから、len バイト分のデータをバイト単位で１６進表現の文字列に変換し、その結果をログファイルに順次書き込みます。

付録－Ａ　通信環境情報定義ファイル例

```
;/------------------------------------------------------------------------//
;/  通信環境情報定義ファイル                                               //
;/------------------------------------------------------------------------//
MAX_LENGTH = 8192
LOGFILE = SHTEST.LOG
LINKLOG = ON
LOG_SIZE  = 100000
MON_PORT = 9999
START = 1                                          ; ch-1 の定義開始
  SECS  = MASTER
  PORT  = COM1
  DVID  = 1111
  BAUD  = 9600
  T1    = 50
  T2    = 50
  T3    = 300
  T4    = 500
  RETRY = 3
  DVID_CK  = ON
  S9F1_RSP = ON
  S9F9_RSP = ON
  LOGSVRCH = SHTEST1.DEF
END
START = 2
  HSMS  = PASSIVE
  PORT  = 5001
  DVID  = 2222
  T3    = 100
  T5    = 100
  T6    = 50
  T7    = 100
  T8    = 50
  DVID_CK  = ON
  S9F1_RSP = ON
  S9F9_RSP = ON
  S9F11_RSP= ON
  LINKTIME = 10
  LOGSVRCH = SHTEST3.DEF
END

START = 3
  HSMS  = ACTIVE
  PORT  = 5001
  IP     = 192.168.1.4
  DVID  = 5556
  T3    = 450
  T5    = 100
  T6    = 50
```

付録－B　通信ログサンプル

```
12/28 09:17:15 ---- CH-1 config info -------------
12/28 09:17:15     HSMS - ACTIVE
12/28 09:17:15     dvid  = 1234 (session id)
12/28 09:17:15     port  = 5001
12/28 09:17:15     svrip = 192.168.1.4
12/28 09:17:15     T3    =  450 x 100 ms
12/28 09:17:15     T5    =  100 x 100 ms
12/28 09:17:15     T6    =   50 x 100 ms
12/28 09:17:15     T7    =  100 x 100 ms
12/28 09:17:15     T8    =   50 x 100 ms
12/28 09:17:15     LINK  = 15 sec
12/28 09:17:15     dvidck= OFF
12/28 09:17:15     S9F1  = OFF
12/28 09:17:15     S9F9  = OFF
12/28 09:17:15     S9F11 = OFF
12/28 09:17:15 HCBLK.sybt=0x60b
12/28 09:17:25 CH-1 Snd P=0 S=1 (Select.Req)   len=0010 sybt=0000060c
12/28 09:17:25 CH-1 Rcv P=0 S=2 (Select.Rsp)   len=0010 sybt=0000060c
12/28 09:17:28 CH-1 Rcv P=0 S=5 (Linktest.Req) len=0010 sybt=00000057
12/28 09:17:28 CH-1 Snd P=0 S=6 (Linktest.Rsp) len=0010 sybt=00000057
12/28 09:17:30 CH-1 Snd P=0 S=5 (Linktest.Req) len=0010 sybt=0000060d
12/28 09:17:30 CH-1 Rcv P=0 S=6 (Linktest.Rsp) len=0010 sybt=0000060d
12/28 09:17:40 CH-1 Snd S1F1     len=0010 dvid=1234  W  blk=0000 sybt=0000060e
12/28 09:17:40 CH-1 Rcv S1F2     len=0028 dvid=1234     blk=0000 sybt=0000060e
               <L 2
                 <A[6]="EQ1000">
                 <A[6]="REV-10">
               >
12/28 09:17:44 CH-1 Rcv P=0 S=5 (Linktest.Req) len=0010 sybt=00000058
12/28 09:17:44 CH-1 Snd P=0 S=6 (Linktest.Rsp) len=0010 sybt=00000058
12/28 09:17:45 CH-1 Snd P=0 S=5 (Linktest.Req) len=0010 sybt=0000060f
12/28 09:17:46 CH-1 Rcv P=0 S=6 (Linktest.Rsp) len=0010 sybt=0000060f
12/28 09:17:48 CH-1 Rcv S1F1     len=0010 dvid=1234  W  blk=0000 sybt=00000059
12/28 09:17:48 CH-1 Snd S1F2     len=8285 dvid=1234     blk=0000 sybt=00000059
               <L 1
                 <L 9
                   <2A[8192]="REVX.X
">
                   <U1[2]=18,255>
                   <S1[2]=18,-1>
                   <U2[4]=4660,65535>
                   <S2[4]=4660,-1>
                   <U4[8]=305419896,4294967295>
                   <S4[8]=305419896,-1>
                   <U8[16]=x0123456789abcdef,xffffffffffffffff>
                   <S8[16]=x0123456789abcdef,xffffffffffffffff>
                 >
```

付録－Ｃ　ＶＢ（Visual Basic）アプリケーションのために

付録－Ｃ１　ＤＬＬ関数宣言　一覧

```
Private Declare Function SECS_Start Lib "shdvr3" _
                Alias "_SECS_Start@4" (ByVal configfile As String) As Long
Private Declare Sub SECS_Terminate Lib "shdvr3" _
           Alias "_SECS_Terminate@0" ()
Private Declare Function SECS_Open Lib "shdvr3" _
                Alias "_SECS_Open@4" (ByVal chno As Long) As Long
Private Declare Function SECS_Close Lib "shdvr3" _
                Alias "_SECS_Close@4" (ByVal chno As Long) As Long
Private Declare Function SECS_SendRecv Lib "shdvr3" _
                Alias "_SECS_SendRecv@28" (ByVal chno As Long, _
                ByRef smsg As Byte, ByVal slen As Long, ByRef rmsg As Byte, _
                ByVal rsize As Long, ByRef rlen As Long, ByVal eia As Long) As Long
Private Declare Function SECS_Send Lib "shdvr3" _
                Alias "_SECS_Send@16" (ByVal chno As Long, ByRef smsg As Byte, _
                ByVal slen As Long, ByRef eia As Long) As Long
Private Declare Function SECS_Recv Lib "shdvr3" _
                Alias "_SECS_Recv@20" (ByVal chno As Long, ByRef rmsg As Byte, _
                ByVal rsize As Long, ByRef rlen As Long, ByVal eia As Long) As Long
Private Declare Function SECS_RecvCk Lib "shdvr3" _
                Alias "_SECS_RecvCk@8" (ByVal chno As Long, ByRef rlen As Long) As Long
Private Declare Function SECS_GetStatus Lib "shdvr3" _
                Alias "_SECS_GetStatus@4" (ByVal chno As Long) As Long
Private Declare Sub SECS_GetSerialNo Lib "shdvr3" _
           Alias "_SECS_GetSerialNo@4" (ByVal snbuff As String)
Private Declare Function SECS_set_msg_head Lib "shdvr3" _
                Alias "_SECS_set_msg_head@28" (ByRef mdef As TMSG_DEF, ByRef msg_ptr As Byte, _
                ByVal max_len As Long, ByVal s As Long, ByVal f As Long, ByVal w As Long, _
                ByRef sybt As Byte) As Long
Private Declare Function SECS_add_msg_item Lib "shdvr3" _
                Alias "_SECS_add_msg_item@16" (ByRef mdef As TMSG_DEF, ByVal item_code As Long, _
                ByVal dlen As Long, val As Any) As Long
Private Declare Function SECS_get_item_unit_size Lib "shdvr3" _
                Alias "_SECS_get_item_size@4" (ByVal item_code As Long) As Long
Private Declare Function SECS_get_msg_init Lib "shdvr3" _
                Alias "_SECS_get_msg_init@12" (ByRef mdef As TMSG_DEF, ByRef msg_ptr As Byte, _
                ByVal msg_len As Long) As Long
Private Declare Function SECS_get_msg_head Lib "shdvr3" _
                Alias "_SECS_get_msg_head@20" (ByRef mdef As TMSG_DEF, ByRef s As Long, _
                ByRef f As Long, ByRef w As Long, ByRef sybt As Byte) As Long
```

```
Private Declare Function SECS_get_item_code_size Lib "shdvr3" _
                Alias "_SECS_get_item_code_size@12" (ByRef mdef As TMSG_DEF, _
                ByRef item_code As Long, ByRef size As Long) As Long
Private Declare Function SECS_get_item_any_data Lib "shdvr3" _
                Alias "_SECS_get_item_any_data@16" (ByRef mdef As TMSG_DEF, _
                ByRef item_code As Long, ByRef size As Long, ByRef data As Any) As Long
Private Declare Function SECS_get_item_data Lib "shdvr3" _
                Alias "_SECS_get_item_data@16" (ByRef mdef As TMSG_DEF, ByVal item_code As Long, _
                ByVal size As Long, ByRef data As Any) As Long
Private Declare Sub SECS_int8_to_dec Lib "shdvr3" _
            Alias "_SECS_int8_to_dec@12" (ByRef d As Any, ByVal buff As String, ByVal n As Integer)
Private Declare Sub SECS_int16_to_dec Lib "shdvr3" _
            Alias "_SECS_int16_to_dec@12" (ByRef d As Any, ByVal buff As String, ByVal n As Integer)
Private Declare Sub SECS_int32_to_dec Lib "shdvr3" _
            Alias "_SECS_int32_to_dec@12" (ByRef d As Any, ByVal buff As String, ByVal n As Integer)
Private Declare Sub SECS_int64_to_dec Lib "shdvr3" _
            Alias "_SECS_int64_to_dec@12" (ByRef d As Any, ByVal buff As String, ByVal n As Integer)
Private Declare Sub SECS_uint8_to_dec Lib "shdvr3" _
            Alias "_SECS_uint8_to_dec@12" (ByRef d As Any, ByVal buff As String, ByVal n As Integer)
Private Declare Sub SECS_uint16_to_dec Lib "shdvr3" _
            Alias "_SECS_uint16_to_dec@12" (ByRef d As Any, ByVal buff As String, ByVal n As Integer)
Private Declare Sub SECS_uint32_to_dec Lib "shdvr3" _
            Alias "_SECS_uint32_to_dec@12" (ByRef d As Any, ByVal buff As String, ByVal n As Integer)
Private Declare Sub SECS_uint64_to_dec Lib "shdvr3" _
            Alias "_SECS_uint64_to_dec@12" (ByRef d As Any, ByVal buff As String, ByVal n As Integer)
Private Declare Sub SECS_int8_to_hex Lib "shdvr3" _
            Alias "_SECS_int8_to_hex@12" (ByRef d As Any, ByVal buff As String, ByVal n As Integer)
Private Declare Sub SECS_int16_to_hex Lib "shdvr3" _
            Alias "_SECS_int16_to_hex@12" (ByRef d As Any, ByVal buff As String, ByVal n As Integer)
Private Declare Sub SECS_int32_to_hex Lib "shdvr3" _
            Alias "_SECS_int32_to_hex@12" (ByRef d As Any, ByVal buff As String, ByVal n As Integer)
Private Declare Sub SECS_int64_to_hex Lib "shdvr3" _
            Alias "_SECS_int64_to_hex@12" (ByRef d As Any, ByVal buff As String, ByVal n As Integer)
Private Declare Sub SECS_f32_to_dec Lib "shdvr3" _
            Alias "_SECS_f32_to_dec@12" (ByRef d As Any, ByVal buff As String, ByVal n As Integer)
Private Declare Sub SECS_f64_to_dec Lib "shdvr3" _
            Alias "_SECS_f64_to_dec@12" (ByRef d As Any, ByVal buff As String, ByVal n As Integer)
Private Declare Sub SECS_str_merge Lib "shdvr3" _
            Alias "_SECS_str_merge@8" (ByVal s1 As String, ByVal s2 As String)
Private Declare Sub SECS_FormatDateTimeExt Lib "shdvr3" _
            Alias "_SECS_FormatDateTimeExt@8" (ByVal buf As String, ByVal fmt As String)
```

付録－Ｃ２　関数結果シンボルと値　一覧

```
'---- API 終了情報(API の返却値  ------------------------------------------------
Const EI_COMM_NORMAL = 0                        ' 正常終了
Const EI_COMM_OPEN_ERROR = 1                    ' 未ｵｰﾌﾟﾝ
Const EI_COMM_CH_ERROR = 2                      ' ﾁｬﾝﾈﾙ番号不正
Const EI_COMM_SEND_ERROR = 3                    ' 送信ﾘﾄﾗｲｴﾗｰ
Const EI_COMM_RECV_ERROR = 4                    ' 受信ﾘﾄﾗｲｴﾗｰ
Const EI_COMM_CKSM_ERROR = 5                    ' Cksm ﾘﾄﾗｲｴﾗｰ
Const EI_COMM_OPENED_ALREADY = 6                ' 既に Opened
Const EI_COMM_WRITE_LENGTH_ERR = 7              ' 送信ﾒｯｾｰｼﾞ長ｴﾗｰ
Const EI_COMM_BUSY = 8                          ' 通信 Driver Busy
Const EI_COMM_T3_TIMEOUT = 13                   ' T3 ﾀｲﾑｱｳﾄｴﾗｰ
Const EI_COMM_OTHER_ERROR = 20                  ' その他のｴﾗｰ
Const EI_COMM_RCV_BUFF_OVERFLOW = 21            ' 受信ﾊﾞｯﾌｧ overflow
Const EI_COMM_DISCONNECTED = 22                 ' 切断エラー
Const EI_REQ_ACCEPT = (-1)                      ' 要求受付け OK=  End 待ち

'---- 以下は、メッセージ生成、データアイテム取得用 API 関数戻り値 ----------------
Const EI_DEF_MSG_ERR = (-30)                    ' ﾒｯｾｰｼﾞ定義(長さ) ｴﾗｰ
Const EI_DEF_MSG_LEN_ERR = (-31)                ' define msg error len err
Const EI_ITEM_CODE_ERR = (-32)                  ' item code err-get_item_data
Const EI_ITEM_SIZE_ERR = (-33)                  ' item size err-get_item_data
Const EI_NO_MORE_ITEM = (-1)                    ' これ以上のデータアイテムなし
```

付録－Ｃ３　アイテムコードシンボルと値　一覧

```
'---- Item Code Symbol for Message definition API ------------------------------
Const I_L = &H0
Const I_B = &H20
Const I_BOOL = &H24
Const I_T = &H24            '( I_BOOL と同じ )
Const I_A = &H40
Const I_J = &H44
Const I_S8 = &H60
Const I_S1 = &H64
Const I_S2 = &H68
Const I_S4 = &H70
Const I_D = &H80
Const I_F = &H90
Const I_U8 = &HA0
Const I_U1 = &HA4
Const I_U2 = &HA8
Const I_U4 = &HB0
```

付録－Ｃ４　メッセージ操作構造体（作成・データアイテム取得管理用）

```
Private Type TMSG_DEF
    s       As Byte
    f       As Byte
    w       As Byte
    sybt(0 To 3) As Byte
    text    As Long
    max_len As Long
    len     As Integer
    next    As Integer
    item_qty As Integer
    work(8)  As Byte
End Type
```