HANDY TERMINAL

DT-93D

AP 開発支援ライブラリ解説書

このマニュアルは、AP 開発支援ライブラリの仕様について記載します。

ご注意

- このソフトウェアおよびマニュアルの、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。
- このソフトウェアおよびマニュアルは、本製品の使用許諾契約書のもとでのみ使用することができます。
- このソフトウェアおよびマニュアルを運用した結果の影響については、一切の責任を負いかねますのでご了承ください。
- このソフトウェアの仕様、およびマニュアルに記載されている事柄は、将来予告なしに変更することがあります。
- このマニュアルの著作権はカシオ計算機株式会社に帰属します。
- 本書中に含まれている画面表示は、実際の画面とは若干異なる場合があります。予めご了承ください。

はじめに

バーコードリーダ一体型ハンディターミナルDT－930は、DT－900の後継機として、非常に注目されています。

当然の事ながら、ハードの普及に比例してアプリケーション開発の需要も増えるわけですが、
・開発言語がCという自由度の高い言語
・マシン特有の操作を行うためのオリジナル関数
というポイントで問い合わせが多く発生しています。

こうした不明点やわかりづらさを解消し、より効率の良いプログラム開発が行えるよう作成されたのが、この「DT－930　アプリケーション開発支援ライブラリ」です。

このユーティリティは、
・DT－930専用C関数の上位に位置する関数
・開発の参考となるCプログラムのソースリスト
・当ユーティリティの関数を用いたサンプルプログラム
から構成されています。

目次

1. 概要

DT－930のアプリケーションプログラム開発は、DT－930専用関数とSHC標準関数（一部使用不可）を使って行います。
しかしながら、これらの各関数はデバイスの基本的な制御を司るもので、それをいろいろなパターンで組み合わせることで開発における自由度が膨らみますが、逆に設計やプログラミングに費やす時間が多くなってしまいます。

この煩雑さを解消するために、一つの関数で従来の数ステップ分の処理を賄える関数群をライブラリとして提供するのが「DT－930　アプリケーション支援ライブラリ」です。

このライブラリは、処理内容により、下記の様に分類されます。

（1）日付チェック
（2）ブロックチェック
（3）入力
（4）通信
（5）ファイル
（6）サービス

それぞれの機能は、下記の内容で提供しています。

機能名	処理内容	ライブラリファイルによる提供	リストによる提供
日付チェック	妥当性チェック	○	
	閏年	○	
	西暦／和暦返還	○	
ブロック	チェックサム計算	○	
チェック	CRC計算	○	
	LRC計算	○	
	チェックディジット計算	○	
入力	制御キー入力	○	
	脱出条件（文字）	○	
	脱出条件（数値）	○	
	日付入力	○	○
	時刻入力	○	○
	シフトキー制御	○	
通信	マルチドロップ送信	○	○
	マルチドロップ受信	○	○
	FLINK（LMWIN）送信	○	○
	FLINK（LMWIN）受信	○	○
	Ir 簡易制御関数群	○	
ファイル制御	ファイルオープン	○	
	ファイルクローズ	○	
	ファイルリード	○	
	ファイルライト	○	
	ファイルサイズ取得	○	
	レコード数取得	○	
サービス	ウェイト	○	○
	デバッグ補助	○	
	音制御	○	○
	XYMODEM 通信	○	
	Ir 制御用 XYMODEM 通信	○	

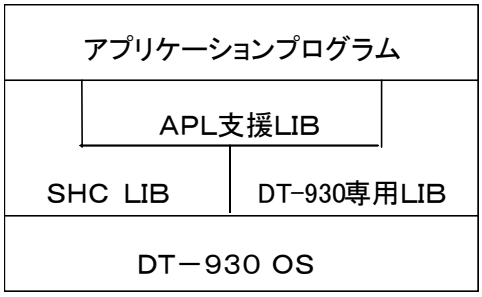
このライブラリは
・ライブラリファイル（ファイル名：DPLIB.LIB）
・インクルードファイル（ファイル名：DPLIB.H）
・関数ソースリスト
・サンプルソースリスト
で構成されています。

2. 当ライブラリの位置づけ

当ライブラリは、DT－930専用関数とSHC標準関数を用いて作成します。

しかしながら、当ライブラリがDT－930のすべての制御を網羅しているわけではないので、必要に応じて専用関数やSHC関数を使って下さい。（当然の事ながら、リンクする場合も当ライブラリとSHCのライブラリ、DT－930専用ライブラリ及びオブジェクトファイルをリンクして下さい）

ソフトウェアの構造は、下記のようになっています。



3. 日付チェック関数群

棚卸や発注業務などで、日付を入力するケースはよくありますが、その際行わなければならないのが入力された日付の妥当性をチェックする処理です。
また、西暦を和暦に変換したり、その逆の処理を行うケースも多々あります。
このような、日付に関する処理を行うものが「日付チェック関数群」であり、これには下記の関数を用意します。

日付妥当性チェック……引数で与えられた日付が妥当かどうかチェックする。
　　閏年もチェックする。
閏年チェック　　……引数で与えられた年が、閏年かどうかを返す。
西暦／和暦変換　……引数の指示に従い、西暦と和暦を変換して返す。

次ページ以降に関数仕様を示します。

3.1.1. ht_CheckDate

日付(西暦4桁月2桁日2桁)の妥当性をチェックします.
日付のチェック範囲は1866年1月1日から2088年12月31日の範囲とします。

```
ER ht_CheckDate(
    H year,
    H month,
    H day
);
```

パラメータ

year
年(西暦1868～2088)

month
月(1～12)

day
日(1～31)

戻り値

チェック結果
E_OK　　正常(妥当な日付)
E_NG　　不正な日付

補足

注意)範囲外(1868年～2088年以外)か、或いは存在しない日付を指定すると、不正日付になります。

【SAMPLE】

```
#include "dplib.h"
          :
          :
      H   year, month, day;
      ER  ercd;
      year=2006;month=2;day=28;
      ercd=ht_CheckDate( year, month, day); /* ２００６年２月２８日をチェック */
      lcd_csr_put( 1, 0);  /* 表示開始位置セット */
      if(ercd == E_OK){
         lcd_string( LCD_ANK_STANDARD, LCD_ATTR_NORMAL, (UB*)"日付ＯＫ！", LCD_LF_OFF);
      }else{
         lcd_string( LCD_ANK_STANDARD, LCD_ATTR_REVERS, (UB*)"日付ＮＧ！", LCD_LF_OFF);
      }
      :
      :
```

3.1.2. ht_CheckYear

入力年が閏年か否かを判定します。
年のチェック範囲は1868年から2088年の範囲とします。

```
ER ht_CheckYear(
    H year
);
```

パラメータ

year
年(西暦1868～2088)

戻り値

チェック結果

E_OK	閏年
E_NG	通常年
E_PRM	対象範囲外

補足

注意)範囲外(1868年～2088年以外)を指定すると、対象範囲外になります。

【SAMPLE】

```
#include "dplib.h"
      :
      :
    H   year;
    ER  ercd;
    year=2006;
    ercd=ht_CheckYear( year); /* ２００６年の閏年チェック */
    lcd_csr_put( 1, 0);  /* 表示開始位置セット */
    if(ercd == E_OK){
        lcd_string( LCD_ANK_STANDARD, LCD_ATTR_NORMAL, (UB*)"閏年です", LCD_LF_OFF);
    }else if(ercd == E_NG){
        lcd_string( LCD_ANK_STANDARD, LCD_ATTR_NORMAL, (UB*)"通常年です！", LCD_LF_OFF);
    }else{
        lcd_string( LCD_ANK_STANDARD, LCD_ATTR_REVERS, (UB*)"範囲外です！", LCD_LF_OFF);
    }
      :
```

3.1.3. ht_ConvertYear

西暦から和暦または、和暦から西暦への変換を行います。
gengou で指定する元号が、西暦(0)の場合は、西暦から和暦に変換し、
和暦(1～4)の場合は和暦から西暦に変換します。

```
ER ht_ConvertYear(
    H in_year,
    H gengou,
    H *out_year,
    H *out_gengou
);
```

パラメータ

in_year
変換対象年

gengou
元号(0:西暦, 1:明治, 2:大正, 3:昭和, 4:平成)

out_year
変換結果年

out_gengou
変換結果元号(0:西暦, 1:明治, 2:大正, 3:昭和, 4:平成)

戻り値

変換結果

E_OK	正常終了
E_NG	変換エラー
E_PRM	西暦から和暦に変換した時、2つの元号にまたがる時。

補足

注意)範囲外(1868年～2088年以外)を指定すると、変換エラーになります。

【SAMPLE】

```
#include "dplib.h"
      :
    H   year, gengou, out_year, out_gengou;
    ER  ercd;
    char msg[33];
      :
      :
    year=2006;gengou = 0;
    ercd=ht_CheckYear( year, gengou, &out_year, &out_gengou); /* 西暦 2006 年を変換 */
    if(ercd == E_OK){
        switch( out_gengou){
           case 1:  sprintf( msg, "西暦%04d 年は明治%02d 年です", year, out_year); break;
           case 2:  sprintf( msg, "西暦%04d 年は大正%02d 年です", year, out_year); break;
           case 3:  sprintf( msg, "西暦%04d 年は昭和%02d 年です", year, out_year); break;
           case 4:  sprintf( msg, "西暦%04d 年は平成%02d 年です", year, out_year); break;
           default: strcpy( msg, "西暦%04d 年は和暦がまたがります", year); break;
        }
        lcd_csr_put( 1, 0);  /* 表示開始位置セット */
        lcd_string( LCD_ANK_STANDARD, LCD_ATTR_NORMAL, (UB*)msg, LCD_LF_ON);
    }

    year=18;gengou = 4;
    ercd=ht_CheckYear( year, gengou, &out_year, &out_gengou); /* 平成 18 年を変換 */
    if(ercd == E_OK){
        sprintf( msg, "平成%02d 年は西暦%04d 年です", year, out_year);
        lcd_csr_put( 1, 0);  /* 表示開始位置セット */
        lcd_string( LCD_ANK_STANDARD, LCD_ATTR_NORMAL, (UB*)msg, LCD_LF_ON);
    }
      :
      :
```

4. ブロックチェック関数群

データ通信時に大概必要になるのが水平パリティのコードであり、一部のバーコードにおいては、バーコードデータの最後にチェックディジットがつくケースがあります。
これらの値を算出するには、そのロジックをプログラムに反映すればいいのですが、まずはロジックを調べることから始まるため、新規に作る場合は少々面倒です。
そこでこの計算ロジックを関数にして提供するのが「ブロックチェック関数群」です。

チェックサム計算	指定データのチェックサムを計算し、値を返す。
CRC計算	指定データのCRC値を計算し、返す（ANSI、CCITT及びXMODEM）
LRC計算	指定データのLRC値を計算し、返す。
ﾁｪｯｸﾃﾞｨｼﾞｯﾄ計算	指定データに対し、指示された計算方式でﾁｪｯｸﾃﾞｨｼﾞｯﾄ を算出し、値を返す。

次ページ以降に関数仕様を示します。

4.1.1. ht_Checksum

check_data で指定したデータのチェックサム値を求めます。

```
UH ht_Checksum(
    unsigned char *check_data,
    H size
);
```

パラメータ

check_data
対象データ

size
データ長

戻り値

チェックサム値

補足

【SAMPLE】

```
#include "dplib.h"
      :
      :
    unsigned char in_data[128];
    UH            summ;

    strcpy( in_data, "0123456789");
    summ = ht_Checksumm( in_data, strlen(in_data));  /* チェックサム値計算 */
      :
      :
```

4.1.2. ht_CalcCRC_ANSI

check_data で指定したデータのANSI規格CRC値を求めます。

```
UH ht_CalcCRC_ANSI(
    unsigned char *check_data,
    H size
);
```

パラメータ

check_data
対象データ

size
データ長

戻り値

ANSI規格CRC値

補足

【SAMPLE】

```
#include "dplib.h"
      :
      :
    unsigned char in_data[128];
    UH            crc;

    strcpy( in_data, "0123456789");
    crc = ht_CalcCRC_ANSI( in_data, strlen(in_data));  /* ＡＮＳＩ（ＣＲＣ）値計算 */
      :
      :
```

4.1.3. ht_CalcCRC_CCITT

check_data で指定したデータのCCITT規格CRC値を求めます。

```
UH ht_CalcCRC_CCITT(
    unsigned char *check_data,
    H size
);
```

【パラメータ】

check_data
対象データ

size
データ長

戻り値

CCITT規格CRC値

補足

【SAMPLE】

```
#include "dplib.h"
      :
      :
    unsigned char in_data[128];
    UH            crc;

    strcpy( in_data, "0123456789");
    crc = ht_CalcCRC_CCITT( in_data, strlen(in_data));  /* ＣＣＩＴＴ（ＣＲＣ）値計算 */
      :
      :
```

4.1.4. ht_CalcCRC_X

check_data で指定したデータのXMODEM用CRC値を求めます。

```
UH ht_CalcCRC_X(
    unsigned char *check_data,
    H size
);
```

パラメータ

check_data
対象データ

size
データ長

戻り値

XMODEM用CRC値

補足

【SAMPLE】

```
#include "dplib.h"
      :
      :
    unsigned char in_data[128];
    UH            crc;

    strcpy( in_data, "0123456789");
    crc = ht_CalcCRC_X( in_data, strlen(in_data));  /* ＸＭＯＤＥＭ（ＣＲＣ）値計算 */
      :
      :
```

4.1.5. ht_CalcLRC

check_data で指定したデータに含まれるデータのLRC値を求めます。

```
UH ht_CalcLRC(
    unsigned char *check_data,
    H size
);
```

パラメータ

check_data
対象データ

size
データ長

戻り値

LRC値

補足

【SAMPLE】

```
#include "dplib.h"
      :
      :
    unsigned char in_data[128];
    UH             lrc;

    strcpy( in_data, "0123456789");
    lrc = ht_CalcLRC( in_data, strlen(in_data));  /* ＬＲＣ値計算 */
      :
      :
```

4.1.6. ht_CheckCD

check_data で指定したバーコードデータのチェックデジットチェック行います。
CDチェックの対象となるバーコード種別は、以下の4種類です。
・WPC(JAN)
・CODE39
・MSI(3種)
・ITF
・IDF
・CODE93
・CODE128

```
ER ht_CheckCD(
    unsigned char *check_data,
    H mode
);
```

パラメータ

check_data
対象データ

mode
バーコード種別

BAR_WPC	WPC, JAN, UPC(A)
BAR_CODE39	CODE39
BAR_MSI1	MSI(1桁 mod10)
BAR_MSI2	MSI(2桁 mod11,mod10)
BAR_MSI3	MSI(2桁 mod10,mod10)
BAR_ITF	Interleave 2 of 5
BAR_IDF	Industrial 2 of 5
BAR_CODE93	CODE93
BAR_CODE128	CODE128

戻り値

チェックデジット判定結果

E_OK:	CD正常
E_NG:	CD異常

補足

【SAMPLE】

```
#include "dplib.h"
      :
      :
    unsigned char in_data[50];
    ER            key, ercd;
      :
    key = key_string( &pkey_inps, in_data);  /* 文字列入力 */
    if( key == E_OK){
        ercd = ht_CheckCD( in_data, OBR_WPC);/* ＷＰＣＣＤチェック */
      :
      :
```

5. 入力関数群

データの入力は、バーコード及びキー入力により行いますが、入力を抜ける条件として、通常の入力以外の条件（ファンクションキー押下、後退キー押下等）を設定するのは若干面倒な処理が入ります。
それらの設定を引数指定により実行できる関数が下記の入力関数です。

制御キー入力	テンキー、電源キー及びシフトキーを除いた制御キーのうち、脱出するキーを引数により指定し、そのキーが押されたら終了する。
脱出条件設定付入力(文字)	引数として脱出条件を渡し、それに合致する条件が発生したら入力処理を抜け、バーコード入力があったら、バーコードバッファから読み出し、文字列に代入する。 脱出した情報は戻り値及びグローバル変数にて知らせる。 ＊脱出条件：ファンクションキー入力、LB発生、及び key_string で設定できる脱出条件
脱出条件設定付入力(数値)	引数として脱出条件を渡し、それに合致する条件が発生したら入力処理を抜け、バーコード入力があったら、バーコードバッファから読み出し、文字列に代入する。 脱出した情報は戻り値及びグローバル変数にて知らせる。 ＊脱出条件：ファンクションキー入力、LB発生、及び key_string で設定できる脱出条件
日付入力	年月日の入力を行い、システム日付を書き換える
時刻入力	時分秒の入力を行い、システム時間を書き換える
シフトキー制御	シフトキーの状態設定・取得を行う

次ページ以降に関数仕様を示します。
日付及び時刻入力については、カスタマイズに用いていただける様に、その関数のソースリストを添付します。

5.1.1. ht_FCWait

入力指定した脱出条件キーが押されるまで、キー入力待ちをします。
脱出条件となるキーはF1～F8, クリア, 後退, 0～9, 実行 およびLB発生です。

```
ER ht_FCWait(
    ER escape
);
```

パラメータ

escape

脱出条件(以下の条件をOR論理で複数指定可)

KY_F01	F1キー押下	KY_1	1キー押下
KY_F02	F2キー押下	KY_2	2キー押下
KY_F03	F3キー押下	KY_3	3キー押下
KY_F04	F4キー押下	KY_4	4キー押下
KY_F05	F5キー押下	KY_5	5キー押下
KY_F06	F6キー押下	KY_6	6キー押下
KY_F07	F7キー押下	KY_7	7キー押下
KY_F08	F8キー押下	KY_8	8キー押下
KY_CLR	クリアキー押下	KY_9	9キー押下
KY_ENT	実行キー押下	KY_0	0キー押下
KY_BS	後退キー押下		
KY_LB	LB発生		

戻り値

押されたキー情報 (上記入力条件と同じ)

KY_F01	F1キー押下	KY_1	1キー押下
KY_F02	F2キー押下	KY_2	2キー押下
KY_F03	F3キー押下	KY_3	3キー押下
KY_F04	F4キー押下	KY_4	4キー押下
KY_F05	F5キー押下	KY_5	5キー押下
KY_F06	F6キー押下	KY_6	6キー押下
KY_F07	F7キー押下	KY_7	7キー押下
KY_F08	F8キー押下	KY_8	8キー押下
KY_CLR	クリアキー押下	KY_9	9キー押下
KY_ENT	実行キー押下	KY_0	0キー押下
KY_BS	後退キー押下		
KY_LB	LB発生		
E_NG	異常終了		

補足

5.1.2. ht_StrInp

指定文字数の文字列入力関数です。
脱出条件となるキーはF1～F8，LB発生，OBR読み込みです。

```
ER ht_StrInp(
    ER escape,
    H fsize,
    H type,
    UH len,
    UH clm,
    UH line,
    UB *string
);
```

パラメータ

escape
脱出条件(以下の条件をOR論理で複数指定可)
※キーコードについては、ht__FCWaitを参照してください。
KY_OBR　　バーコード読み込み時

fsize
フォントサイズ
LCD_ANK_LIGHT　　縮小ANK
LCD_ANK_STANDARD　　標準ANK

type
表示属性
LCD_ATTR_NORMAL　　通常
LCD_ATTR_REVERS　　反転
LCD_ATTR_WIDTH　　強調

len
入力文字列の最大バイト数

clm
文字列桁位置(0～15)

line
文字列行位置(0～7)

string
入力文字列格納エリアポインタ
(格納エリアは最大バイト数＋1の容量が必要)

戻り値

押されたキー情報（発生イベント）

KY_F01	F1キー
KY_F02	F2キー
KY_F03	F3キー
KY_F04	F4キー
KY_F05	F5キー
KY_F06	F6キー
KY_F07	F7キー
KY_F08	F8キー
KY_ENT	実行キー
KY_LB	LB発生
KY_OBR	バーコード読み込み
E_NG	異常終了

補足

5.1.3. ht_NumInp

指定文字数の電卓型数値入力関数です。
脱出条件となるキーはF1～F8，LB発生，OBR読み込みです。

```
ER ht_StrInp(
    ER escape,
    H fsize,
    H type,
    UH len,
    UH clm,
    UH line,
    UB *string
);
```

パラメータ

escape
脱出条件（以下の条件をOR論理で複数指定可）
※キーコードについては、ht__FCWaitを参照してください。

KY_OBR	バーコード読み込み時

fsize
フォントサイズ

LCD_ANK_LIGHT	縮小ANK
LCD_ANK_STANDARD	標準ANK

type
表示属性

LCD_ATTR_NORMAL	通常
LCD_ATTR_REVERS	反転
LCD_ATTR_WIDTH	強調

len
入力文字列の最大バイト数

clm
文字列桁位置（0～15）

line
文字列行位置（0～7）

string
入力文字列格納エリアポインタ
（格納エリアは最大バイト数＋1の容量が必要）

戻り値

押されたキー情報（発生イベント）

KY_F01	F1キー
KY_F02	F2キー
KY_F03	F3キー
KY_F04	F4キー
KY_F05	F5キー
KY_F06	F6キー
KY_F07	F7キー
KY_F08	F8キー
KY_ENT	実行キー
KY_LB	LB発生
KY_OBR	バーコード読み込み
E_NG	異常終了

補足

5.1.4. ht_DateInp

日付の入力を行います。
脱出条件となるキーはF1～F8, クリア, 後退, 0～9, LB発生です。

```
ER ht_DateInp(
    B md,
    ER escape,
    H fsize,
    H type,
    UH len,
    UH clm,
    UH line,
    UB *string
);
```

パラメータ

md
入力モード

SEIREKI_8	西暦年4桁＋月2桁＋日2桁
SEIREKI_6	西暦年2桁＋月2桁＋日2桁
WAREKI_6	和暦(平成)年2桁＋月2桁＋日2桁

escape
脱出条件(以下の条件をOR論理で複数指定可)
※キーコードについては、ht__FCWaitを参照してください。

fsize
フォントサイズ

LCD_ANK_LIGHT	縮小ANK
LCD_ANK_STANDARD	標準ANK

type
表示属性

LCD_ATTR_NORMAL	通常
LCD_ATTR_REVERS	反転
LCD_ATTR_WIDTH	強調

len
入力文字列の最大バイト数

clm
文字列桁位置(0～15)

line
文字列行位置(0～7)

string

入力文字列格納エリアポインタ
（格納エリアは最大バイト数＋1の容量が必要）

戻り値

押されたキー情報（発生イベント）
※キーコードについては、ht__FCWaitを参照してください。

補足

```
/*****************************************************************************/
/*  < ﾌﾟﾛｸﾞﾗﾑ名 >                                                             */
/*    日付入力                                                               */
/*  < 機能概要 >                                                            */
/*    日付の入力を行います。                                                 */
/*    ･SEIREKI_8 が指定された場合、範囲は 1868 年 01 月 01 から              */
/*     2088 年 12 月 31 日です。                                             */
/*    ･SEIREKI_6 が指定された場合で、年が 80 年以上のときは、1900 年代      */
/*     としてﾁｪｯｸします。80 年未満の場合は、2000 年代としてﾁｪｯｸします。*/
/*    ･入力した日付が正しくない場合、当関数から Exit しません。             */
/*  < 文法 >                                                                */
/*    ER   ht_DateInp(  B mode  ,                                            */
/*                      ER escape,                                           */
/*                       H fsize ,                                           */
/*                       H type  ,                                           */
/*                      UH clm   ,                                           */
/*                      UH line  ,                                           */
/*                      UB *string  )                                        */
/*  < ﾊﾟﾗﾒｰﾀ >                                                               */
/*     B mode   : 入力ﾓｰﾄﾞ                                                   */
/*                SEIREKI_8 : 西暦 4 桁+月 2 桁+日 2 桁                       */
/*                SEIREKI_6 : 西暦 2 桁+月 2 桁+日 2 桁                       */
/*                WAREKI_6  : 和暦(平成)2 桁+月 2 桁+日 2 桁                  */
/*    UH escape : 脱出条件(以下の条件を OR 論理で複数指定)                    */
/*                KY_F01 : F1 ｷｰ                                             */
/*                KY_F02 : F2 ｷｰ                                             */
/*                KY_F03 : F3 ｷｰ                                             */
/*                KY_F04 : F4 ｷｰ                                             */
/*                KY_F05 : F5 ｷｰ                                             */
/*                KY_F06 : F6 ｷｰ                                             */
/*                KY_F07 : F7 ｷｰ                                             */
/*                KY_F08 : F8 ｷｰ                                             */
/*                KY_ENT : 実行ｷｰ                                            */
/*                KY_LB  : LB 発生                                            */
/*                KY_OBR : ﾊﾞｰｺｰﾄﾞ読込み                                      */
/*     H fsize  : ﾌｫﾝﾄｻｲｽﾞ                                                  */
/*                LCD_ANK_LIGHT    : 縮小 ANK                                 */
/*                LCD_ANK_STANDARD: 標準 ANK                                  */
/*     H type   : 表示属性                                                    */
/*                LCD_ATTR_NORMAL : 通常                                      */
/*                LCD_ATTR_REVERS : 反転                                      */
/*                LCD_ATTR_WIDTH  : 強調                                      */
/*    UH clm    : 文字列桁位置(0～15)                                        */
/*    UH line   : 文字列行位置(0～ 7)                                        */
/*    UB *string: 入力文字列格納ｴﾘｱﾎﾟｲﾝﾀ                                     */
/*                格納ｴﾘｱは最大ﾊﾞｲﾄ数+1 の容量が必要)                         */
/*  < ﾘﾀｰﾝｺｰﾄﾞ >                                                             */
/*    押下されたｷｰ情報(以下のｷｰ情報を返します)                              */
/*                KY_F01 : F1 ｷｰ                                             */
```

```
/*                KY_F02 : F2ｷｰ                                          */
/*                KY_F03 : F3ｷｰ                                          */
/*                KY_F04 : F4ｷｰ                                          */
/*                KY_F05 : F5ｷｰ                                          */
/*                KY_F06 : F6ｷｰ                                          */
/*                KY_F07 : F7ｷｰ                                          */
/*                KY_F08 : F8ｷｰ                                          */
/*                KY_ENT : 実行ｷｰ                                        */
/*                KY_LB  : LB発生                                         */
/*                KY_OBR : ﾊﾞｰｺｰﾄﾞ読込み                                  */
/*                E_NG   : 異常終了                                       */
/*                E_PRM  : ﾊﾟﾗﾒｰﾀｴﾗｰ                                      */
/*************************************************************************/
ER  ht_DateInp(B mode, ER escape, H fsize, H type, UH clm, UH line, UB *string)
{
ER ret;
UH len;
H  year,wyear,gengo;
H  month;
H  day;
ER wescape;

    ret=0L;
    wescape=(escape & KY_OBR_CANCEL);         /* ﾊﾞｰｺｰﾄﾞを使用させないため     */
    for(;;){
        if     (mode==SEIREKI_8) len=8;       /* 西暦4桁+月2桁+日2桁         */
        else if(mode==SEIREKI_6) len=6;       /* 西暦2桁+月2桁+日2桁         */
        else if(mode==WAREKI_6 ) len=6;       /* 和暦(平成)2桁+月2桁+日2桁   */
        else{ ret=E_PRM; break; }

        ret=ht_NumInp(wescape, fsize, type, len, clm, line, string);
        if(ret==KY_ENT){
            if (mode==SEIREKI_8){
                if(strlen((const char *)string)==8){
                    year =(H)memto_I(string  , 4);
                    month=(H)memto_I(string+4, 2);
                    day  =(H)memto_I(string+6, 2);
                    if(ht_CheckDate(year, month, day)==E_OK) break;
                }
            }
            else if (mode==SEIREKI_6){
                if(strlen((const char *)string)==6){
                    year =(H)memto_I(string  , 2);
                    month=(H)memto_I(string+2, 2);
                    day  =(H)memto_I(string+4, 2);
                    if(year<80) year += 2000;
                    else        year += 1900;
                    if(ht_CheckDate(year, month, day)==E_OK) break;
                }
            }
```

```
            else{
                if(strlen((const char *)string)==6){
                    year =(H)memto_l(string  , 2);
                    month=(H)memto_l(string+2, 2);
                    day  =(H)memto_l(string+4, 2);
                    if(ht_ConvertYear(year, 4, &wyear, &gengo)==E_OK){
                        if(ht_CheckDate(wyear, month, day)==E_OK) break;
                    }
                }
            }
        }
        else break;
    }
    return(ret);
}
```

5.1.5. ht_TimeInp

時刻の入力を行います。
脱出条件となるキーはF1～F8，クリア，後退，0～9，LB発生です。

```
ER ht_TimeInp(
    ER escape,
    H fsize,
    H type,
    UH len,
    UH clm,
    UH line,
    UB *string
);
```

パラメータ

escape
脱出条件（以下の条件をOR論理で複数指定可）
※キーコードについては、ht__FCWaitを参照してください。

fsize
フォントサイズ

LCD_ANK_LIGHT	縮小ANK
LCD_ANK_STANDARD	標準ANK

type
表示属性

LCD_ATTR_NORMAL	通常
LCD_ATTR_REVERS	反転
LCD_ATTR_WIDTH	強調

len
入力文字列の最大バイト数

clm
文字列桁位置（0～15）

line
文字列行位置（0～7）

string
入力文字列格納エリアポインタ
（格納エリアは最大バイト数＋1の容量が必要）

戻り値

押されたキー情報（発生イベント）
※キーコードについては、ht__FCWaitを参照してください。

補足

```
/*************************************************************************/
/*  < ﾌﾟﾛｸﾞﾗﾑ名 >                                                         */
/*    時刻入力                                                           */
/*  < 機能概要 >                                                        */
/*    時刻の入力を行います。                                             */
/*    ･時間は 24 時間制でﾁｪｯｸします。                                   */
/*    ･入力した時刻が正しくない場合、当関数から Exit しません。          */
/*  < 文法 >                                                            */
/*    ER   ht_TimeInp( ER escape,                                        */
/*                     H fsize ,                                         */
/*                     H type  ,                                         */
/*                    UH clm   ,                                         */
/*                    UH line  ,                                         */
/*                    UB *string  )                                      */
/*  < ﾊﾟﾗﾒｰﾀ >                                                          */
/*    UH escape : 脱出条件(以下の条件を OR 論理で複数指定)               */
/*                KY_F01 : F1ｷｰ                                         */
/*                KY_F02 : F2ｷｰ                                         */
/*                KY_F03 : F3ｷｰ                                         */
/*                KY_F04 : F4ｷｰ                                         */
/*                KY_F05 : F5ｷｰ                                         */
/*                KY_F06 : F6ｷｰ                                         */
/*                KY_F07 : F7ｷｰ                                         */
/*                KY_F08 : F8ｷｰ                                         */
/*                KY_ENT : 実行ｷｰ                                       */
/*                KY_LB  : LB 発生                                       */
/*                KY_OBR : ﾊﾞｰｺｰﾄﾞ読込み                                 */
/*     H fsize  : ﾌｫﾝﾄｻｲｽﾞ                                             */
/*                LCD_ANK_LIGHT    : 縮小 ANK                            */
/*                LCD_ANK_STANDARD: 標準 ANK                             */
/*     H type   : 表示属性                                               */
/*                LCD_ATTR_NORMAL : 通常                                 */
/*                LCD_ATTR_REVERS : 反転                                 */
/*                LCD_ATTR_WIDTH  : 強調                                 */
/*    UH clm    : 文字列桁位置(0～15)                                   */
/*    UH line   : 文字列行位置(0～ 7)                                   */
/*    UB *string: 入力文字列格納ｴﾘｱﾎﾟｲﾝﾀ                                */
/*                格納ｴﾘｱは最大ﾊﾞｲﾄ数+1 の容量が必要)                     */
/*  < ﾘﾀｰﾝｺｰﾄﾞ >                                                        */
/*    押下されたｷｰ情報(以下のｷｰ情報を返します)                          */
/*                KY_F01 : F1ｷｰ                                         */
/*                KY_F02 : F2ｷｰ                                         */
/*                KY_F03 : F3ｷｰ                                         */
/*                KY_F04 : F4ｷｰ                                         */
/*                KY_F05 : F5ｷｰ                                         */
/*                KY_F06 : F6ｷｰ                                         */
/*                KY_F07 : F7ｷｰ                                         */
/*                KY_F08 : F8ｷｰ                                         */
```

```
/*                KY_ENT : 実行ｷｰ                                          */
/*                KY_LB  : LB 発生                                          */
/*                KY_OBR : ﾊﾞｰｺｰﾄﾞ読込み                                    */
/*                E_NG   : 異常終了                                         */
/*                E_PRM  : ﾊﾟﾗﾒｰﾀｴﾗｰ                                        */
/*--------------------------------------------------------------------------*/
ER  ht_TimeInp( ER escape, H fsize, H type, UH clm, UH line, UB *string )
{
ER ret;
UH len;
H  hour;
H  min;
H  sec;
ER wescape;

    ret=0L;
    len=6;
    wescape=(escape & KY_OBR_CANCEL);          /* ﾊﾞｰｺｰﾄﾞを使用させないため     */
    for(;;){
        ret=ht_NumInp(wescape, fsize, type, len, clm, line, string);
        if(ret==KY_ENT){
            if(strlen((const char *)string)==6){
                hour=(H)memto_l(string  , 2);
                min =(H)memto_l(string+2, 2);
                sec =(H)memto_l(string+4, 2);
                if(((hour>=0)&&(hour<=23))&&
                   ((min>=0)&&(min<=59))  &&
                   ((sec>=0)&&(sec<=59))) break;
            }
        }
        else break;
    }
    return(ret);
}


/*--------------------------------------------------------------------------*/
/*  < ﾌﾟﾛｸﾞﾗﾑ名 >                                                           */
/*    数値変換                                                              */
/*  < 機能概要 >                                                           */
/*    指定されたﾊﾞｯﾌｧ内のﾃﾞｰﾀに NULL を付加して atol する。                  */
/*  < 文法 >                                                               */
/*    W memto_l( void *buf, size_t len )                                    */
/*  < ﾊﾟﾗﾒｰﾀ >                                                             */
/*    void   *buf : 文字列ﾊﾞｯﾌｧ                                           */
/*    size_t len  : 文字列長                                              */
/*  < ﾘﾀｰﾝｺｰﾄﾞ >                                                           */
/*    変換後の値。                                                        */
/*  ☆日付、時刻入力で使っている内部関数です。                              */
/*--------------------------------------------------------------------------*/
```

```
W    memto_l( void *buf , size_t len )
{
UB   wk[16];

     memcpy( wk , buf , len );
     wk[len] = 0x00;
     return( atol((char *) wk ) );
}
```

5.1.6. ht_ShiftMode

シフトキーの状態の読み出しあるいは設定を行います。

```
ER ht_ShiftMode(
    H mode
);
```

パラメータ

mode
読出／設定モード

SFT_READ	シフト状態読み出し
SFT_SET_OFF	シフトオフの状態に設定
SFT_SET_ON	シフトオンの状態に設定

戻り値

読出／設定結果

E_SFT_ON	シフトオン状態
E_SFT_OFF	シフトオフ状態
E_PRM	パラメータエラー

補足

6. 通信関数群

マルチドロップ、FLINKプロトコルに対応したファイル転送を一括で行えるような関数を用意します。

ﾏﾙﾁﾄﾞﾛｯﾌﾟ送信	通信条件及び送信ファイルテーブルをセットすることで、マルチドロッププロトコルの通信を行う。
ﾏﾙﾁﾄﾞﾛｯﾌﾟ受信	通信条件をセットすることで、マルチドロッププロトコルによる受信処理を行う。
FLINK送信	通信条件、送信ファイル、送信先ディレクトリをセットすることで、FILNKプロトコルの通信を行う。
FLINK受信	通信条件、受信ファイル、受信先ディレクトリをセットすることで、FILNKプロトコルによる受信処理を行う。
Ir簡易制御関数群	IrDA制御関数を、カシオIr関数（c__XX）と同等に扱えるようにした関数。

次ページ以降に関数仕様を示します。
また、カスタマイズに用いていただける様に各関数のソースリストを添付します。

☆上記の関数群のうち
マルチドロップ送信／受信は、ベーシックI／Oボックス
FLINK送信／受信、Ir簡易制御関数群は、マスタ及びサテライトI／Oボックスに機能します。

6.1.1. ht_MLTsend

マルチドロッププロトコルによる送信を行います。

```
ER ht_MLTsend(
    H com_no,
    UB connectMode,
    DAT_COM_STR *param
    B fnam_tbl[12][],
    H *s_file,
    H *phase
);
```

パラメータ

※com__no～＊paramのパラメータはcu__openと全くおなじです。
Cライブラリ解説書の「8. 通信ユーティリティ部関数」をご覧ください。

fnum_tbl[12][]

送信ファイル名テーブル(★最終は先頭が￥0)

```
例)
  fnum_tbl[12][]={
        "SHOHIN□□MST",     /* SHOHIN.MST  */
        "KOKYAKU□MST",     /* KOKYAKU.MST  */
        "TORIHIKITRN",     /* TORIHIKI.TRN */
        ""};
                          □はスペースコード
```

s_file

送信済みファイル数の格納ポインタ(正常時は全件が格納される)

phase

異常終了時のフェーズの格納ポインタ

0	オープン
2	ファイル送信中
3	終了時(クローズ)

戻り値

関数結果

E_OK	正常終了
E_NG	通信エラー(詳細は cu_readErrStat で確認してください)
E_PRM	パラメータエラー

6.1.2. ht_MLTrecv

マルチドロッププロトコルによる受信を行います。

```
ER ht_MLTrecv(
    H com_no,
    UB connectMode,
    DAT_COM_STR *param,
    H *r_file,
    H *phase
);
```

パラメータ

※com__no～＊paramのパラメータはcu__openと全くおなじです。
Cライブラリ解説書の「8. 通信ユーティリティ部関数」をご覧ください。

r_file
受信済みファイル数の格納ポインタ

phase
異常終了時のフェーズの格納ポインタ

0	オープン
2	ファイル受信中
3	終了時(クローズ)

戻り値

関数結果

E_OK	正常終了
E_NG	通信エラー(詳細は cu_readErrStat で確認してください)
E_PRM	パラメータエラー

補足

```
/*********************************************************************/
/*  < ﾌﾟﾛｸﾞﾗﾑ名 >                                                     */
/*    ホスト送信                                                     */
/*  < 機能概要 >                                                     */
/*    パソコンに対してＦＬＩＮＫプロトコル                           */
/*    にてファイルを送信する。                                       */
/*  < 文法 >                                                         */
/*    ER ht_MLTsend(H com_no, H irSpeed, CU_RSPRM *param,            */
/*                        B fnam_tbl[12][], B *dir,                  */
/*                        H *s_file, H *phase);                      */
/*  < ﾊﾟﾗﾒｰﾀ >                                                       */
/*    H       com_no          :  ＣＯＭ番号                        */
/*                                COM0,COM1                          */
/*    H       irSpeed             :    赤外通信最大速度             */
/*     CU_RSPRM          *param :  通信パラメータデータのポインタ    */
/*                        typedef struct {                           */
/*                            W         speed;      転送速度 UB      */
/*                            W         length;   データ長           */
/*                            W         parity;   パリティビット     */
/*                            W         stop_bit; ストップビット     */
/*                        } CU_RSPRM;                                */
/*     B      fnam_tbl[][]    :  送信するファイル名称テーブル       */
/*     B       *dir           :    送信ディレクトリ                 */
/*     H      *s_file             :  送信完了ファイル数             */
/*     H      *phase              :  途中中断時の送信フェーズ       */
/*  < ﾘﾀｰﾝｺｰﾄﾞ >                                                    */
/*    ER       ercd       :  関数処理結果                         */
/*                               E_OK     : 正常終了               */
/*                               E_NG  : 異常終了                  */
/*                               E_PRM    : パラメータエラー       */
/*********************************************************************/
ER       ht_MLTsend( H com_no, H irSpeed, CU_RSPRM *param,
                        B *fnam_tbl[], B *dir, H *s_file, H *phase)
{
    UB                      fileCnt, i;
    ER                      errStat;
    ER                      retCode;
    W                       totalsize;
    W                       fsize;
    CU_GRAPHSET             graphSet;

    /* 送信ファイル数をチェックする */
    totalsize=0;
    for(fileCnt = 0;;fileCnt++){
        if( fnam_tbl[fileCnt]==0x00){
            break;
```

```
        }
        if( fnam_tbl[fileCnt][0]==0x00){
            break;
        }
    }
    if( fileCnt == 0){
        return(E_PRM);
    }

    graphSet.graphMode = CU_GRAPH_ON_2;
    graphSet.graphPos  = 0;
    graphSet.graphCol  = 0;
    graphSet.graphName = CU_GRAPH_NM_FILE;
    graphSet.graphLine = 1;

    *phase = 0;   /* オープンフェーズ */
    *s_file = 0;/* 送信完了ファイル数初期化 */
    /* マルチドロップ通信オープン */
    if ((retCode = cu_open(com_no, irSpeed, param, CU_MODE_HT)) == E_OK) {
        *phase = 1;  /* 送信フェーズ */
        /* １ファイル送信のループ */
        for(i=0;i<fileCnt;i++){
            /* 送信を起動 */
            retCode=cu_fileSend(com_no, CU_TRANS_NORMAL, fnam_tbl[i], dir, CU_PROTECT_VALID,
                    &graphSet);
            if (retCode != E_OK) {
                break;
            }
            (*s_file)++;
        }
        if( retCode == 0){
            *phase = 2;
        }
        errStat = cu_close(com_no, CU_CLOSE_NORMAL);
        if( retCode == 0){
            retCode  = errStat;
        }
    }
    return(retCode);
}
```

```
/*************************************************************************/
/*  < ﾌﾟﾛｸﾞﾗﾑ名 >                                                         */
/*    ホスト受信                                                         */
/*  < 機能概要 >                                                         */
/*    パソコンよりＦＬＩＮＫプロトコルにてファイルを受信する。           */
/*  < 文法 >                                                             */
/*    ER ht_MLTrecv(H com_no, H irSpeed, CU_RSPRM *param, B *dir,        */
/*                          H *r_file, H *phase);                        */
/*  < ﾊﾟﾗﾒｰﾀ >                                                           */
/*    H       com_no          :  ＣＯＭ番号                            */
/*                                    COM0,COM1                          */
/*    H       irSpeed           :    赤外通信最大速度                   */
/*     CU_RSPRM          *param :  通信パラメータデータのポインタ       */
/*                       typedef struct {                               */
/*                            W         speed;      転送速度 UB         */
/*                            W         length;    データ長             */
/*                            W         parity;    パリティビット       */
/*                            W         stop_bit; ストップビット        */
/*                       } CU_RSPRM;                                    */
/*     B      fnam_tbl[][]    :  送信するファイル名称テーブル           */
/*     B       *dir           :    受信ディレクトリ                     */
/*     H      *r_file             :  受信完了ファイル数                 */
/*     H      *phase              :  途中中断時の送信フェーズ           */
/*  < ﾘﾀｰﾝｺｰﾄﾞ >                                                         */
/*    ER       ercd       :  関数処理結果                               */
/*                               E_OK     : 正常終了                   */
/*                               E_NG  : 異常終了                      */
/*                               E_PRM    : パラメータエラー           */
/*************************************************************************/
ER       ht_MLTrecv(H com_no, H irSpeed, CU_RSPRM *param,
                       B *fnam_tbl[], B *dir, H *s_file, H *phase)
{
    UB                      fileCnt, i;
    ER                      errStat;
    ER                      retCode;
    W                       totalsize;
    W                       fsize;
    CU_GRAPHSET             graphSet;

    /* 受信ファイル数をチェックする */
    totalsize=0;
    for(fileCnt = 0;;fileCnt++){
        if( fnam_tbl[fileCnt]==0x00){
            break;
        }
        if( fnam_tbl[fileCnt][0]==0x00){
            break;
        }
    }
```

```
    if( fileCnt == 0){
        return(E_PRM);
    }

    graphSet.graphMode = CU_GRAPH_ON_2;
    graphSet.graphPos  = 0;
    graphSet.graphCol  = 0;
    graphSet.graphName = CU_GRAPH_NM_FILE;
    graphSet.graphLine = 1;

    *phase = 0;   /* オープンフェーズ */
    *s_file = 0;/* 受信完了ファイル数初期化 */
    /* マルチドロップ通信オープン */
    if ((retCode = cu_open(com_no, irSpeed, param, CU_MODE_HT)) == E_OK) {
        *phase = 1;  /* 受信フェーズ */
        /* １ファイル受信のループ */
        for(i=0;i<fileCnt;i++){
            /* 受信を起動 */
            retCode=cu_fileRecv(com_no, CU_TRANS_NORMAL, fnam_tbl[i], dir, CU_PROTECT_VALID,
                        &graphSet);
            if (retCode != E_OK) {
                break;
            }
            (*s_file)++;
        }
        if( retCode == 0){
            *phase = 2;
        }
        errStat = cu_close(com_no, CU_CLOSE_NORMAL);
        if( retCode == 0){
            retCode  = errStat;
        }
    }
    return(retCode);
}
```

6.1.3. ht_FLNKsend

FLINKプロトコルによる送信を行います。

```
ER ht_FLNKsend(
    H com_no,
    H irSpeed,
    CU_RSPRM *rsPrm,
    H mode,
    H sendmode,
    B *fnam_area,
    B *dir,
    H protect,
    H *sdir,
    H *phase
);
```

パラメータ

※com__no～protectのパラメータはcu__open及びcu__fileSendと全くおなじです。
（fileSendの“mode”は“sendmode”としています。）
Cライブラリ解説書の「8. 通信ユーティリティ部関数」をご覧ください。

s_file
送信済みファイル数の格納ポインタ（正常時は全件が格納される）

phase
異常終了時のフェーズの格納ポインタ

0	オープン
2	ファイル送信中
3	終了時(クローズ)

戻り値

関数結果

E_OK	正常終了
E_NG	通信エラー（詳細は cu_readErrStat で確認してください）
E_PRM	パラメータエラー

補足

LMWINは、サーバモードの状態にして下さい。

6.1.4. ht_FLNKrecv

FLINKプロトコルによる受信を行います。

```
ER ht_MLTrecv(
    H com_no,
    H irSpeed,
    CU_RSPRM *rsPrm,
    H mode,
    H recvmode,
    B *fnam_area,
    B *dir,
    H protect,
    H *sdir,
    H *phase
);
```

パラメータ

※com__no～protectのパラメータはcu__open及びcu__fileRecvと全くおなじです。
（fileSendの“mode”は“recvmode”としています。）
Cライブラリ解説書の「8. 通信ユーティリティ部関数」をご覧ください。

r_file
受信済みファイル数の格納ポインタ

phase
異常終了時のフェーズの格納ポインタ

0	オープン
2	ファイル受信中
3	終了時(クローズ)

戻り値

関数結果

E_OK	正常終了
E_NG	通信エラー(詳細は cu_readErrStat で確認してください)
E_PRM	パラメータエラー

補足

LMWINは、サーバモードの状態にして下さい。

6.1.5. Ir_c_open

Ir回線をオープンします。

```
ER Ir_c_open(
    H com_no,
    UW param,
    B *buff,
    H buf_l,
    TIM_TBL *tim_out,
    DEL_TBL *del_cod,
    B busy_ch,
    B nonbusy_ch
);
```

パラメータ

com_no

通信ポート

COM0	IRインタフェース

param

通信形式パラメータ(各パラメータの論理和で指定)

ボーレート	B_115200	115200 bps
	B_57600	57600 bps
	B_38400	38400 bps
	B_19200	19200 bps
	B_9600	9600 bps
	B_4800	4800 bps
	B_2400	2400 bps
	B_1200	1200 bps
パリティビット	PARI_NON	なし
	PARI_ODD	奇数
	PARI_EVN	偶数
キャラクタレングス	CHAR_8	8ビット
	CHAR_7	7ビット
ストップビット	STOP_1	1ビット
	STOP_2	2ビット
SI／SO制御	SI_ON	制御する
	SI_OFF	制御しない
フロー制御	BUSY_OFF	制御しない
	XON_XOFF	DC1, DC3による XON/XOFF 制御
	BUSY_CHAR	指定コードによる XON/XOFF 制御
	RS_CS	RS／CSによるRS／CSﾌﾛｰ制御
RS信号制御	RTS_ON	RS信号ON
	RTS_OFF	RS信号OFF
ER信号制御	ER_ON	ER信号ON
	ER_OFF	ER信号OFF

busy_ch
XOFFコード(受信不可時のコード)

nonbusy_ch
XONコード(受信可能時のコード)

buff
受信バッファアドレス

buf_l
受信バッファレングス
(0の時BIOS内部の16バイトエリアを受信バッファとして使用します)

tim_out

```
typedef  struct  {
H    cs;            : ＣＳタイムアウト監視値(0～32767(×7.8ms))
H    dr;            : ＤＲタイムアウト監視値(0～32767(×7.8ms))
H    cd;            : ＣＤタイムアウト監視値(0～32767(×7.8ms))
}  TIM_TBL;
```

del_cod

```
typedef  struct  {
B    del_n;         : デリートコード数 (0～4)
UB   del_c[4];      : デリートコード (0x00～0xff)
}  DEL_TBL;
```

戻り値

関数結果

E_OK	正常終了
E_NG	オープンエラー
E_PRM	パラメータエラー

補足

☆ビジー制御は、本関数での設定は無視されます。I／Oボックスをご使用の場合は、ディップスイッチで設定して下さい。

☆I／Oボックスをご使用の場合は、通信速度を合わせて下さい。
☆デフォルトのIrの設定は
　　2次局
　　3WIRE－RAW
としています。
必要に応じて変更を行って下さい。

6.1.6. Ir_c_close

Ir回線をクローズします。

```
ER Ir_c_close(
    H com_no
);
```

パラメータ

com_no
通信ポート
COM0　　IRインタフェース

戻り値

関数結果
E_OK　　正常終了
E_NG　　クローズエラー
E_PRM　　パラメータエラー

6.1.7. Ir_c_status

Ir回線のステータスをチェックします。

```
ER Ir_c_status(
    H com_no
);
```

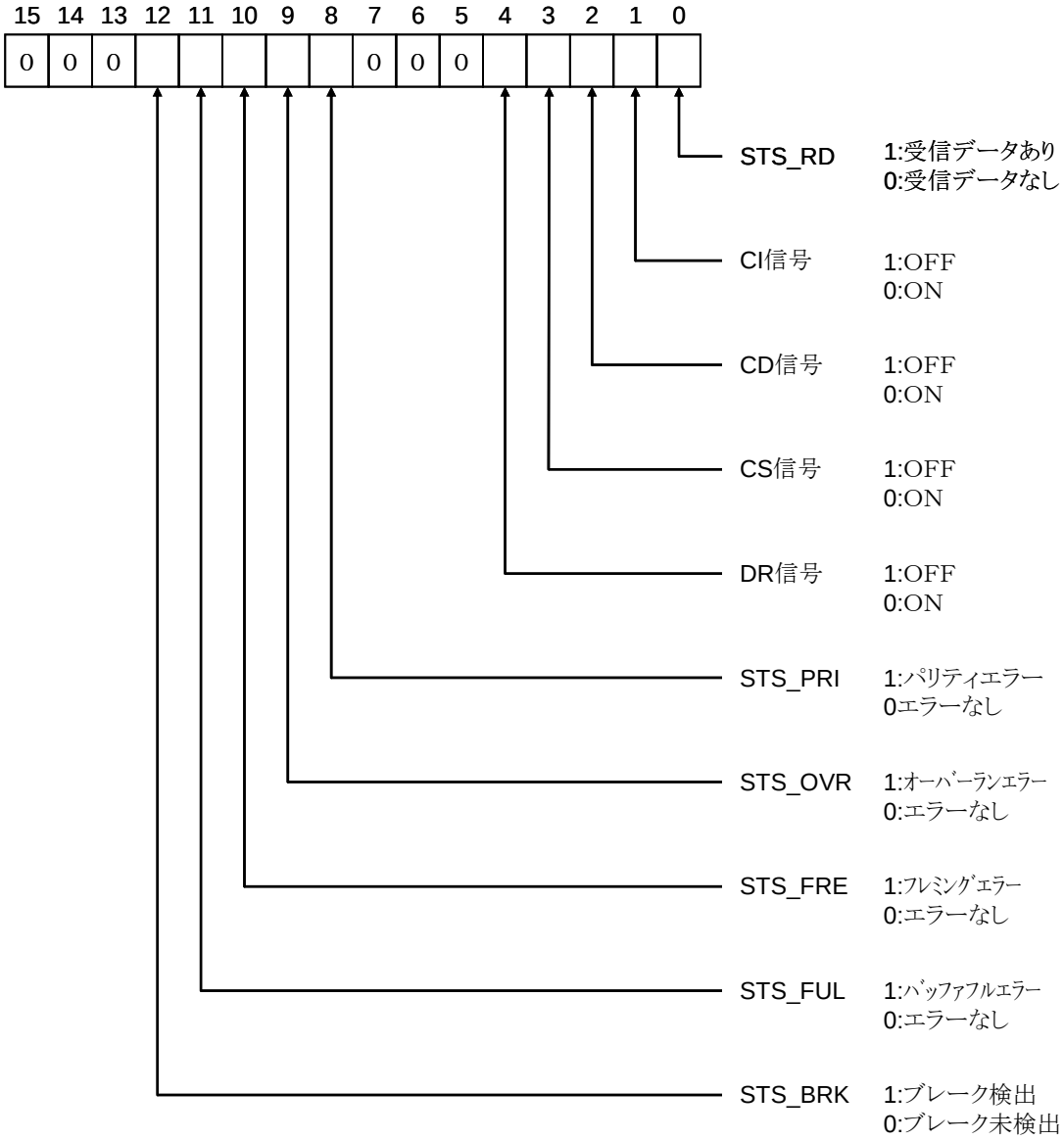
パラメータ

com_no
通信ポート
COM0　　IRインタフェース

戻り値

関数結果
ステータス　　正常終了(次ページ参照)
E_NG　　回線エラー
E_PRM　　パラメータエラー

COMステータス


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0 0 0 0 0
STS_RD
1:受信データあり
0:受信データなし
CI信号
1:OFF
0:ON
CD信号
1:OFF
0:ON
CS信号
1:OFF
0:ON
DR信号
1:OFF
0:ON
STS_PRI
1:パリティエラー
0エラーなし
STS_OVR
1:オーバーランエラー
0:エラーなし
STS_FRE
1:フレミングエラー
0:エラーなし
STS_FUL
1:バッファフルエラー
0:エラーなし
STS_BRK
1:ブレーク検出
0:ブレーク未検出

6.1.8. Ir_c_hold

本関数では、特に処理を行いません。(パラメータチェックのみ)

```
ER Ir_c_hold(
    H com_no,
    B mode
);
```

パラメータ

com_no

通信ポート

COM0	IRインタフェース

mode

占有設定

HOLD_ON	占有する
HOLD_OFF	占有解除

戻り値

関数結果

E_OK	正常終了
E_PRM	パラメータエラー

6.1.9. Ir_c_chkopen

Irポートがオープンされているかどうかをチェックします。

```
ER Ir_c_chkopen(void);
```

パラメータ

なし

戻り値

関数結果

1	回線オープン中
0	回線未オープン

6.1.10. Ir_c_dout

Irポートから送信バッファに格納されたデータを指定された文字数(バイト数)分送信します。
指定の送信文字数が0である場合は、送信バッファ内の NULL 文字の手前までの文字を送信します。

```
ER Ir_c_dout(
    H com_no,
    B *buffer,
    H length
);
```

パラメータ

com_no
通信ポート
COM0　　カシオ IR インタフェース

buffer
送信バッファアドレス

length
送信文字数(バイト数)

戻り値

関数結果
E_OK　　正常終了
E_NG　　異常終了
E_PRM　　パラメータエラー

6.1.11. Ir_c_din

受信バッファに格納されたデータを1文字(byte)読出します。
受信データが存在しない場合、受信データを待ちとなります。

```
ER Ir_c_din(
    H com_no,
    B *buffer
);
```

パラメータ

com_no
通信ポート
COM0　　カシオ IR インタフェース

buffer
送信バッファアドレス

戻り値

関数結果
E_OK　　正常終了
E_NG　　異常終了
E_PRM　　パラメータエラー

6.1.12. Ir_c_tmdin

受信バッファに格納さてたデータを1文字読出します。
受信データが存在しない場合、受信タイムアウト監視値の間受信データ待ちとなります。
タイムアウト監視値が0の場合は、タイムアウト監視を行いません。

```
ER Ir_c_tmdin(
    H com_no,
    B *buffer,
    H rcv_time
);
```

パラメータ

com_no
通信ポート
COM0　　カシオ IR インタフェース

buffer
送信バッファアドレス

rcv_time
受信タイムアウト監視値 0～32767 （×7.8ms）

戻り値

関数結果
E_OK　　正常終了
E_NG　　異常終了
E_PRM　　パラメータエラー

補足

「DR／CS／CD タイムアウト監視値の設定」ファンクションにより信号線の監視を行います。

6.1.13. Ir_c_out

Irポートから1文字数(バイト数)分送信します。

```
ER Ir_c_out(
    H com_no,
    UB snddata
);
```

パラメータ

com_no
通信ポート
COM0　　カシオ IR インタフェース

snddata
送信データ

戻り値

関数結果
E_OK　　正常終了
E_NG　　異常終了
E_PRM　　パラメータエラー

6.1.14. Ir_c_break

ブレーク信号の送出または、送出停止を行います。

```
ER Ir_c_break(
    H com_no,
    UB mode
);
```

パラメータ

com_no
通信ポート
COM0　　カシオ IR インタフェース

mode
ブレーク信号制御
BRK_ON　　ブレーク信号を送出する
BRK_OFF　　ブレーク信号を停止する

戻り値

関数結果
E_OK　　正常終了
E_NG　　異常終了
E_PRM　　パラメータエラー

6.1.15. Ir_c_txrx

本関数では、特に処理を行いません。(パラメータチェックのみ)

```
ER Ir_c_txrx(
    H com_no,
    UB mode
);
```

パラメータ

com_no

通信ポート

COM0	カシオ IR インタフェース

mode

送受信の有効／無効

C_RXENB	受信を有効に設定
C_TXENB	送信を有効に設定
C_RXDSB	受信を無効に設定
C_TXDSB	送信を無効に設定
C_RXTXENB	送受信を有効に設定
C_RXTXDSB	送受信を無効に設定

戻り値

関数結果

E_OK	正常終了
E_PRM	パラメータエラー

6.1.16. Ir_c_iobox

本関数では、特に処理を行いません。(パラメータチェックのみ)

```
ER Ir_c_iobox(
    H com_no,
    UB mode
);
```

パラメータ

com_no

通信ポート

COM0	カシオ IR インタフェース

mode

送受信の有効／無効

C_IOBOXENB	送信に設定
C_IOBOXDSB	送信を解除

戻り値

関数結果

E_OK	正常終了
E_PRM	パラメータエラー

6.1.17. Ir_c_irout

本関数は、Ir_c_dout と同等の処理を行います。

```
ER Ir_c_irout(
    H com_no,
    B *buffer,
    H length
);
```

パラメータ

com_no
通信ポート
COM0　　カシオ IR インタフェース

buffer
送信バッファアドレス

length
送信文字数(バイト数)

戻り値

関数結果
E_OK　　正常終了
E_NG　　異常終了
E_PRM　　パラメータエラー

6.1.18. Ir_c_timer

CS、DR、CD信号のタイムアウト値を設定します。

```
ER Ir_c_timer(
    H com_no,
    H cs_time,
    H dr_time,
    H cd_time
);
```

パラメータ

com_no
通信ポート
COM0　　カシオ IR インタフェース

cs_time
CSタイムアウト値(0～32767)×7.8ms

dr_time
CSタイムアウト値(0～32767)×7.8ms

cs_time
CSタイムアウト値(0～32767)×7.8ms

戻り値

関数結果
E_OK　　正常終了
E_NG　　異常終了
E_PRM　　パラメータエラー

6.1.19. Ir_c_rs

RS信号のON／OFFを設定します。

```
ER Ir_c_rs(
    H com_no,
    B mode
);
```

パラメータ

com_no

通信ポート

COM0	カシオ IR インタフェース

mode

RS信号設定

RS_ON	RS信号ON
RS_OFF	RS信号OFF

戻り値

関数結果

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

6.1.20. Ir_c_er

ER信号のON／OFFを設定します。

```
ER Ir_c_er(
    H com_no,
    B mode
);
```

パラメータ

com_no

通信ポート

COM0	カシオ IR インタフェース

mode

ER信号設定

ER_ON	ER信号ON
ER_OFF	ER信号OFF

戻り値

関数結果

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

6.1.21. Ir_c_errs

ER及びRS信号のON／OFFを設定します。

```
ER Ir_c_errs(
    H com_no,
    B mode
);
```

パラメータ

com_no

通信ポート

COM0	カシオ IR インタフェース

mode

ER信号設定

ERRS_ON	ER／RS信号ON
ERRS_OFF	ER／RS信号OFF

戻り値

関数結果

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

6.1.22. Ir_c_flush

Irポートのバッファをクリアします。

```
ER Ir_c_flush(
    H com_no
);
```

パラメータ

com_no
通信ポート
COM0　　カシオ IR インタフェース

戻り値

関数結果
E_OK　　正常終了
E_NG　　異常終了
E_PRM　　パラメータエラー

6.1.23. Ir_c_bfsts

Irポートのバッファの文字数をチェックします。
受信文字数を構造体のメンバーchar_no に格納し、それ以外のメンバーは0になります。

```
ER Ir_c_bfsts(
    H com_no,
    COM_STS *bfsts
);
```

パラメータ

com_no
通信ポート
COM0　　カシオ IR インタフェース

bfsts
受信バッファステータス

```
typedef struct {
    H  char_no      : 受信文字数
    H  rest_no      : 受信可能残り文字数
    UB char_cod     : 先頭文字コード
} COM_STS
```

戻り値

関数結果
E_OK　　正常終了
E_NG　　異常終了
E_PRM　　パラメータエラー

6.1.24. Ir_c_errbfring

本関数では、特に処理を行いません(パラメータチェックのみです)。

```
ER Ir_c_errbfring(
    H com_no,
    B mode,
    UB c_errcd
);
```

パラメータ

com_no
通信ポート
COM0　　カシオ IR インタフェース

mode
設定／解除
ERRCD_ON　　エラーコードバッファリング制御する
ERRCD_OFF　　エラーコードバッファリング制御しない

c_errcd
エラーコード(任意)

戻り値

関数結果
E_OK　　正常終了
E_PRM　　パラメータエラー

6.1.25. Ir_c_rderrsts

Irポートのエラーステータスの読出し及びクリアを行います。
各ファンクションのリターンコードが異常終了であるとき本ファンクションでエラーステータスを読出し詳細を調べることができます。
エラーステータスは複数の場合があります。

```
ER Ir_c_rderrsts(
    H com_no,
    UW *com_status
);
```

パラメータ

com_no
通信ポート
COM0　　カシオ IR インタフェース

com_status
エラーステータス

戻り値

関数結果
E_OK　　正常終了
E_PRM　　パラメータエラー

補足

エラーステータス詳細は、Cライブラリ解説書「6．4．2 エラー詳細」を参照して下さい。

6.1.26. Ir_c_chghdr

本関数では、特に処理を行いません(パラメータチェックのみです)。

```
ER Ir_c_chghdr(
    H com_no,
    B mode
);
```

パラメータ

com_no

通信ポート

COM0	カシオ IR インタフェース

mode

受信ハンドラ切替え

STAND_HDR	標準受信ハンドラ設定
HIGH_HDR	簡易受信ハンドラ設定

戻り値

関数結果

E_OK	正常終了
E_PRM	パラメータエラー

7. ファイル制御関数群

DT－930では、ファイル制御はCの標準関数（fopenやopen等）にて行います。
このため、読み込み・書き込み時に特定のレコードにアクセスする場合は、自分でファイルポインタを移動する必要があります。
こういった手間を省き、BASICライクなファイル制御を実現するのが「ファイル制御関数群」です。
この関数群を使用する場合は、ファイル名とレコード長のテーブルを用意し、アクセスはユニークなファイル番号で行います。

ファイルオープン	ファイル（引数にてテーブルを渡し、指定）をオープンする。
ファイルクローズ	ファイル（ファイル番号にて指定）をクローズする。
ファイルリード	ファイル（ファイル番号にて指定）の指定レコード番号のデータを読み込む。
ファイルライト	ァイル（ファイル番号にて指定）の指定レコード番号にデータを書き込む。
ファイルサイズ取得	定ファイル（ファイル番号にて指定）のファイルサイズを返す。
レコード数取得	ファイル（ファイル番号にて指定）のレコード数を返す。

☆この関数群を使用する場合は、DT－930のファイルモードを必ず「DT700モード」にして下さい。

次ページ以降に関数仕様を示します。

ファイル構造体配列テーブル

次ページ以降のファイル制御関数で使用するファイル管理テーブルの構造は以下の通りです。
ファイル管理テーブルの名称は、固定になっていますので、ファイル制御関数を使用する際は、
必ずこの名前にてテーブルをグローバル定義する様にしてください。

```
struct  ht_filetbl {
         char       fname[14];        /* ファイル名称 */
         H          rsize;            /* １レコードのバイト値 */
         FILE       *fp;              /* ストリーム保存（ファイル制御関数で使用） */
};
typedef struct ht_filetbl  FILE_TBL;
```

```
例）
FILE_TBL   filetbl[]={
        { "shohin.mst"   , 120, (FILE*)0 },
        { "kokyaku.mst"  ,  56, (FILE*)0 },
        { "tana.trn"     ,  32, (FILE*)0 },
        { ""             ,   0, (FILE*)0 }
};
```

注意1）ストリームポインタはNULL値で初期設定しておいてください。（ファイルクローズ状態）
注意2）最終行は、必ず上記の様にファイル名なし、レコードサイズ0を付加してください。

7.1.1. ht_fileopen

ファイル構造体配列テーブルで定義されたファイルをオープンします。

```
int ht_fileopen(
    B fno
);
```

パラメータ

fno
オープンするファイル番号(1～)

戻り値

現在のレコード数あるいはエラー情報

正の数	現在のレコード数
-1	fopen( )関数エラー
-2	テーブル有効数を超えたファイル番号
-3	パラメータ異常

補足

当関数を使用する場合は、必ずファイルモードを「DT700モード」にして下さい。

7.1.2. ht_fileclose

ht_fileopen 関数でオープンされたファイルをクローズします。

```
ht_fileclose(
    B fno
);
```

パラメータ

fno

クローズするファイル番号(0あるいは1～)

0	オープンしているファイルすべてをクローズ
1～	指定のファイルをクローズ

戻り値

なし

補足

当関数を使用する場合は、必ずファイルモードを「DT700モード」にして下さい。

7.1.3. ht_fileread

ht_fileopen 関数でオープンされたファイルから1レコード読み込みます。

```
int ht_fileread(
    B fno,
    int rno,
    char *buffer
);
```

パラメータ

fno
処理対象のファイル番号

rno
読み込みするレコード序数(1～)

buffer
レコードデータ読み込みエリアポインタ

戻り値

読み込んだレコード序数
正の数	読み込んだレコード序数(正常であればパラメータと同一)
-1	fread( )関数エラー
0	存在しないレコード序数

補足

当関数を使用する場合は、必ずファイルモードを「DT700モード」にして下さい。

7.1.4. ht_filewrite

ht_fileopen 関数でオープンされたファイルに1レコード書き込みます。

```
int ht_filewrite(
    B fno,
    int rno,
    char *buffer
);
```

パラメータ

fno
処理対象のファイル番号

rno
書き込みするレコード序数(1～)

buffer
レコードデータ書き込みエリアポインタ

戻り値

書き込んだレコード序数

正の数	書き込んだレコード序数(正常であればパラメータと同一)
-1	fwrite( )関数エラー
-2	既存レコード数＋2以降に書き込みしようとした
-3	空き容量不足

補足

当関数を使用する場合は、必ずファイルモードを「DT700モード」にして下さい。

7.1.5. ht_filesize

ファイル構造体配列テーブルで定義されたファイルのサイズを与えます。

```
int ht_filesize(
    B fno
);
```

パラメータ

fno
ファイル番号(1～)

戻り値

レコードサイズ

正の数	ファイルサイズ(バイト数)
0	ファイルが存在しない
－1	パラメータ異常(ファイル番号0以下)

補足

当関数を使用する場合は、必ずファイルモードを「DT700モード」にして下さい。

7.1.6. ht_filelof

ファイル構造体配列テーブルで定義されたファイルの登録レコード数を与えます。

```
int ht_filelof(
    B fno
);
```

パラメータ

fno
ファイル番号(1～)

戻り値

レコード数

正の数	レコード数
0	ファイルが存在しない
－1	パラメータ異常(ファンル番号が0以下)

補足

当関数を使用する場合は、必ずファイルモードを「DT700モード」にして下さい。

8. サービス関数群

プログラム開発に有用な関数群です。

ウェイト処理	渡された時間（単位：秒）だけウェイトします。
デバッグ補助関数	れた変数やメッセージを通信ポートへ出力する。
音制御	のエラー音を引数により分ける
XYMODEM	DEM手順によるファイルの送受信を行う関数です。
IrDA用XYMODEM	DAポートでXYMODEMを行う関数です。

次ページ以降に関数仕様を示します。
また、カスタマイズのためにウェイト関数、及び音制御についてはその関数のソースリストを添付します。

☆I／Oボックスを用いてXYMODEM通信を行う場合、
「XYMODEM」なら、ベーシックI／Oボックス
「IrDA用XYMODEM」なら、マスタもしくはサテライトI／Oボックス
をご使用下さい。

8.1.1. ht_waitmsec

250ミリ秒の単位で250ミリ秒～最大1時間までの時間を待ちます。

```
ER ht_waitmsec(
    UW count
);
```

パラメータ

count
待ち時間(250ミリ秒単位)
例:count = 40;　/* 待ち時間＝10秒 */

戻り値

関数結果

E_OK	正常終了(count*250msec 後に戻る)
E_PRM	パラメータエラー

補足

8.1.2. ht_dbgsendmsg

指定の文字列を固定の通信設定値で赤外線ポートより送信します。
通信を使用しないアプリケーションのデバッグ時に任意の文字列をI／Oボックス経由で、パソコン等の機器に送信します。

```
void ht_dbgsendmsg(
    UB *buffer,
    H len
);
```

パラメータ

buffer
送信データの先頭ポインタ

len
送信バイト数

戻り値

なし

補足

この関数は、下記の設定でCOM0からデータを送信します。
通信速度：9600BPS／データ長：8ビット／ストップビット：1ビット／全二重

8.1.3. ht_beep

3種類のエラー音を鳴らします。

(1)長音1回
(2)短音3回
(3)短音5回

```
void ht_beep(
    B type
)
```

パラメータ

type

エラー音のタイプ

1	長音1回(750msec)
2	短音3回(250msec鳴動, 125msec休止)×3回
3	短音5回(250msec鳴動, 125msec休止)×5回

補足

8.1.4. xy_modem

XMODEMプロトコル及びYMODEMプロトコルによるデータ通信を行います。

```
ER xy_modem(
    UH mode,
    B *tel,
    B *at,
    B *file_list[],
    B *path
)
```

パラメータ

mode

通信モード(下記の内容を論理和演算子で指定、詳細は後述)

※設定内容

機能、ﾓﾃﾞﾑ操作、ﾌﾟﾛﾄｺﾙ、エラー検出、ﾊﾟｹｯﾄ長、ﾎﾟｰﾄ、'K'送信、ﾎﾞｰﾚｰﾄ、ﾒｯｾｰｼﾞ、ﾎﾟｰﾄ操作

tel

電話番号文字列(省略時はNULL文字列を指定)

※使用可能文字

番号	'0'～'9'
ﾀﾞｲﾔﾙ信号ﾓｰﾄﾞ	'T'(ﾄｰﾝﾀﾞｲﾔﾙ)／'P'(ﾊﾟﾙｽﾀﾞｲﾔﾙ)
記号	'*','#' (ﾄｰﾝﾀﾞｲﾔﾙのみ有効)
区切り	'-','(',')' (ﾀﾞｲﾔﾙ時は無視)
ポーズ	',' (ウェイト時間はﾓﾃﾞﾑの設定による)

at

ATコマンド文字列(省略時はNULL文字列を指定)

file_list[]

ファイル名リスト(詳細は後述)

path

送信の際は、必ずNULLを指定して下さい

受信の際は、ファイルを格納するドライブ及びディレクトリ名を指定して下さい。

戻り値

xy_modem()の戻り値は終了情報を表します。
詳細は下記の通りです。

0	正常終了
1	強制終了(ファンクションキー押下による中止)
2	パラメータエラー
6	入力ファイルなし
7	出力ファイル作成エラー
9	通信エラー
11	電圧低下による中止(LB通知を行った場合のみ)
32	2重オープン
33	応答なし
34	回線接続不可
35	話し中
36	ホストからキャンセルされ
37	回線未オープン
38	モデムエラー
50	内部エラー
51	タイムアウト
52	外部機器エラー

☆上記戻り値の1(強制終了)は、アプリケーションでキー通知モードの設定(key_fnc_mode関数にて設定)を行ったときのみ発生します。
発生時には、グローバル変数 xy_error に通知フラグをセットし、xy_modem() 内で通知フラグを一度クリアします。

☆上記戻り値の11(電圧低下)は、アプリケーションでLB通知モードの設定(pwr_inhabit関数、にて設定)を行ったときのみ発生します。
発生時には、グローバル変数 xy_error に通知フラグをセットし、xy_modem() 内で通知フラグを一度クリアします。

☆APOの発生が懸念される場合、pwr_hold_apo 関数にてAPOを禁止して下さい。

☆上記戻り値が38の時、グローバル変数 xy_error にモデムからのリザルトコードを数字でセットします。

補足

xy_modem()

・xy_modem の通信モードの詳細（　はデフォルト値）

機能	指定なし	:送受信を行わない
	XY_SEND	:送信
	XY_RECEIVE	:受信
モデム操作	指定なし	:モデム操作を行わない
	XY_MODEM	:発(着)信を行う
プロトコル	XY_YMODEM	:YMODEM
	XY_XMODEM	:XMODEM
エラー検出	XY_CRC	:CRC
	XY_CHKSUM	:チェックサム
パケット長	XY_LONG	:ロング(1024バイト)
	XY_SHORT	:ショート(128バイト)
ポート	XY_0	:赤外線インターフェース(COM0)
	XY_1	:シリアルインターフェース(COM1)
'K' 送信	XY_KOFF	:'K' を送信しない
	XY_KON	:'K' を送信する
ボーレート	XY_1200	:1200bps
	XY_2400	:2400bps
	XY_4800	:4800bps
	XY_9600	:9600bps
	XY_19200	:19200bps
	XY_38400	:38400bps
	XY_57600	:57600bps
	XY_115200	:115200bps
メッセージ	XY_MOFF	:メッセージ表示しない
	XY_MON	:メッセージ表示する
ポート操作	XY_232ON	:RS-232Cを操作する
	XY_232OFF	:RS-232Cを操作しない

☆通信モード指定時の注意点

・機能＝XY_SEND の場合　：　通信モードのプロトコル、パケット長を参照します。
YMODEMの場合、file_list には複数の指定が可能です。

・機能＝XY_RECEIVE の場合　：　通信モードのプロトコル、エラー検出を参照します。
XMODEMの場合、file_list の指定が必要です。
YMODEMの場合、'K'送信の指定が必要です。

・モデム操作＝XY_MODEM の場合：電話番号の指定がある場合、発信を行います
(at の指定が可能)。
電話番号の指定がない場合、着信を行います
(at の指定が可能)。

☆通信モードにおける機能、モデム操作、ポート操作の指定における xy_modem()の処理内容

機能	モデム操作	ポート操作	xy_modem() の処理内容
指定なし	指定なし	XY_232ON XY_232OFF	パラメータエラー パラメータエラー
	XY_MODEM	XY_232ON XY_232OFF	パラメータエラー 発(着)信
XY_SEND	指定なし	XY_232ON XY_232OFF	オープン→送信→クローズ 送信
	XY_MODEM	XY_232ON XY_232OFF	オープン→発(着)信→送信→回線断→クローズ 発(着)信→送信→回線断
XY_RECEIVE	指定なし	XY_232ON XY_232OFF	オープン→受信→クローズ 受信
	XY_MODEM	XY_232ON XY_232OFF	オープン→発(着)信→受信→回線断→クローズ 発(着)信→受信→回線断

☆モデム操作にて XY_MODEM を指定した場合、モデムに対しては、以下の設定を行います。

ダイヤルトーン検出・ビジートーン検出	ATX4
*キャラクタエコーなし	ATE0
*リザルトコードを数字で返す	ATV0
自動応答しない	ATS0＝0
*CDを常にON	AT&C0

*の印の付いた項目は変更しないで下さい。

・ファイル名リストについて

ファイル名は必ずフルパスで格納して下さい。

ex. Aドライブのルートディレクトリにある TEST.C というファイルは
＜ドライブ名＋ディレクトリ＋ファイル名＞という形でセットして下さい。

```
memcpy(&(file_list[0]), "A:¥¥TEST.C", 10);
```

また、最終テーブルの先頭にはNULLコード('¥0')をセットして下さい。

```
A:¥¥TEST.C
B:¥¥TEST.DAT
¥0
```

8.1.5. Ir_xy_modem

IrDAポートを介してXMODEMプロトコル及びYMODEMプロトコルによるデータ通信を行います。

```
ER Ir_xy_modem(
    UH mode,
    B *tel,
    B *at,
    B *file_list[],
    B *path
)
```

パラメータ

mode
通信モード

tel
電話番号文字列(省略時はNULL文字列を指定)

at
ATコマンド文字列(省略時はNULL文字列を指定)

file_list[]
ファイル名リスト

path
送信の際は、必ずNULLを指定して下さい
受信の際は、ファイルを格納するドライブ及びディレクトリ名を指定して下さい。

戻り値

終了情報(内容は xy_modem()と同様)

補足

パラメータの設定内容や終了情報は、基本的には、xy_modem()関数と同様ですが、下記の2点が異なります。

☆通信ポートについて
本関数の通信ポート指定は、COM0(IrDAポート)のみです。
引数「通信モード」のポートには、COM0(XY__0)のみ、指定可能です。

☆通信速度について
xy_modem()では、通信速度は19200bpsまででしたが、本関数では、115200bpsまで設定可能です。(速度設定の定義パラメータは下記の通り)

XY_38400	38400bps
XY_115200	115200bps

9. サンプルプログラムについて

本ライブラリの関数を使用したサンプルプログラムを用意しました。
プログラムの内容は、それぞれ下記のようになっています。

・SAMPLE1.SRC・・・入力処理サンプル
・SAMPLE2.SRC・・・日付・時刻入力関数サンプル
・SAMPLE3.SRC・・・ファイル制御関数サンプル
・SAMPLE4.SRC・・・日付チェック関数サンプル
・SAMPLE5.SRC・・・ビープ音、ウェイト関数サンプル
・SAMPXY.SRC ・・・XYMODEM関数サンプル

各関数を使用する際に、参考になさって下さい。

DT－930
AP開発支援ライブラリ解説書
平成１８年２月　Rev1.0発行

カシオ計算機株式会社