CASIO

CASIO DT-930 C Language SDK
GUI Version

拡張機能ライブラリ
リファレンスマニュアル

DT-930の機能を拡張するAPIを、リファレンス形式で説明します。

Ver. 1.00

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。

変更履歴

バージョン	変更日付	ページ	内容
0.90	2007/01/22		新規作成
1.00	2007/02/05	46/53/54 62/64	本機では特に処理を行いません→パラメータチェックをします　に修正
		32	IrDA 4Mbps 通信の使用条件追加
		73	XYMODEM 機能の通信速度の注意追加
		各章先頭	ヘッダ／オブジェクト／ライブラリファイルの記載統一
		84	関数機能の説明内容修正
		88-91	Bluetooth プリンタ詳細機能説明追加
		98	関数名誤記修正(HashRead/HashAssign)
		118	「以下の機能を付加します」誤記修正

目次

1. アプリケーション支援ライブラリ

1.1. 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。

本ライブラリはこの煩雑さを解消するために、1 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提供します。

提供ファイル

DPLIB.H	アプリケーション支援ライブラリのヘッダファイル
DPLIB.LIB	アプリケーション支援ライブラリのライブラリファイル

本ライブラリは、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

本ライブラリは、下記の機能を提供しています。

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	チェックサム計算	○	
	CRC 計算	○	
	LRC 計算	○	
	チェックデジット計算	○	
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	○	

1.2. 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。

機能名	説明
ht_CheckDate	日付妥当性チェック
ht_CheckYear	閏年チェック
ht_ConvertYear	西暦／和暦変換

1.2.1. 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

E_OK	正常(妥当な日付)
E_NG	不正な日付

補足

範囲外(1868 年～2088 年以外)か、ある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);
 }
   :
   :
```

1.2.2. 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

E_OK	閏年
E_NG	通常年
E_PRM	対象範囲外

補足

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);
 }
   :
```

1.2.3. 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

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);
 }
  :
  :
```

1.3. 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。

機能名	説明
ht_CheckSum	チェックサム計算
ht_CalcCRC_ANSI	CRC 計算
ht_CalcCRC_CCITT	CRC 計算
ht_CalcCRC_X	CRC 計算
ht_CalcLRC	LRC 計算
ht_CheckCD	チェックデジット計算

1.3.1. 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 */
   :
   :
```

1.3.2. 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 */
   :
   :
```

1.3.3. 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 */
   :
   :
```

1.3.4. 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 */
   :
   :
```

1.3.5. 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 */
   :
   :
```

1.3.6. 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

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 */
   :
   :
```

1.4. 入力関数群

データの入力は、バーコードおよ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。

機能名	説明
ht_FCWait	制御キー入力
ht_StrInp	脱出条件設定付入力（文字）
ht_NumInp	脱出条件設定付入力（数値）
ht_DateInp	日付入力
ht_TimeInp	時刻入力
ht_ShiftMode	シフトキー制御

日付およ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。

1.4.1. 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		

1.4.2. 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

1.4.3. 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

1.4.4. 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

```
/*****************************************************************************/
/*    〈プログラム名〉                                                       */
/*    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)                                           */
/*    〈パラメータ〉                                                         */
/*    B   mode    : 入力モード                                               */
/*                    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    : バーコード読込み                           */
/*    H   fsize   : フォントサイズ                                           */
/*                    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ポインタ                             */
/*                    格納エリアは最大バイト数+1 の容量が必要)               */
/*    〈リターンコード〉                                                     */
/*    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 : バーコード読込み                           */
/*                  E_NG   : 異常終了                                   */
/*                  E_PRM  : パラメータエラー                           */
/*********************************************************************/
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);      /* バーコード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);
}
```

1.4.5. 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

```
/*****************************************************************************/
/*    〈プログラム名〉                                                       */
/*    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)                                           */
/*    〈パラメータ〉                                                         */
/*    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 : バーコード読込み                              */
/*    H   fsize   : フォントサイズ                                           */
/*                    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ポインタ                             */
/*                    (格納エリアは最大バイト数+1 の容量が必要)              */
/*    〈リターンコード〉                                                     */
/*    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 : バーコード読込み                          */
/*                    E_NG   : 異常終了                                  */
/*                    E_PRM  : パラメータエラー                          */
/************************************************************************/
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);      /* バーコード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_I(string  , 2);
              min =(H)memto_I(string+2, 2);
              sec =(H)memto_I(string+4, 2);
              if(((hour>=0)&&(hour<=23))&&
                 ((min>=0)&&(min<=59))  &&
                 ((sec>=0)&&(sec<=59))) break;
          }
      }
      else break;
   }
   return(ret);
}

/************************************************************************/
/*    <プログラム名>                                                    */
/*    数値変換                                                          */
/*    <機能概要>                                                        */
/*    指定されたバッファ内のデータに NULL を付加して atol する。        */
/*    <文法>                                                            */
/*    W       memto_I( void *buf, size_t len )                          */
/*    <パラメータ>                                                      */
/*    void    *buf : 文字列バッファ                                     */
/*    size_t  len  : 文字列長                                           */
/*    <リターンコード>                                                  */
/*    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_I( void *buf , size_t len )
{
 UB  wk[16];

 memcpy( wk , buf , len );
```

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

1.4.6. 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

1.5. 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提供します。

機能名	説明
ht_MLTsend	マルチドロップ送信
ht_MLTrecv	マルチドロップ受信
ht_FLINKsend	FLINK 送信
ht_FLINKrecv	FLINK 受信
Ir_c_xxx	Ir 簡易制御関数群

※IrDA を使用した、4Mbps の高速通信を行う場合は、ht_FLINKsend または ht_FLINKrecv 関数を使い、USB IOBOX を使用してください。

1.5.1. 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。
詳細は「デバイス制御ライブラリ リファレンスマニュアル 通信ユーティリティ制御」を参照してください。

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

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

1.5.2. 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。
詳細は「デバイス制御ライブラリ リファレンスマニュアル 通信ユーティリティ制御」を参照してください。

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

```
/*********************************************************************/
/*    〈プログラム名〉                                               */
/*    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);                           */
/*    〈パラメータ〉                                                 */
/*    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        */
/*    〈リターンコード〉                                             */
/*    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);
}
```

```
/*****************************************************************************/
/*    〈プログラム名〉                                                       */
/*    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);                                   */
/*    〈パラメータ〉                                                         */
/*    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                  */
/*    〈リターンコード〉                                                     */
/*    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);
}
```

1.5.3. 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～*param のパラメータは cu_open と全くおなじです。
（fileSend の“mode”は“sendmode”としています。）
詳細は「デバイス制御ライブラリ リファレンスマニュアル 通信ユーティリティ制御」を参照してください。

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ください。

1.5.4. 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 および cu_fileRecv と全くおなじです。
（fileSend の“mode”は“recvmode”としています。）
詳細は「デバイス制御ライブラリ リファレンスマニュアル 通信ユーティリティ制御」を参照してください。

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。

1.5.5. 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ください。
I／O ボックスをご使用の場合は、通信速度を合わせてください。
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ください。

1.5.6. 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

1.5.7. 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

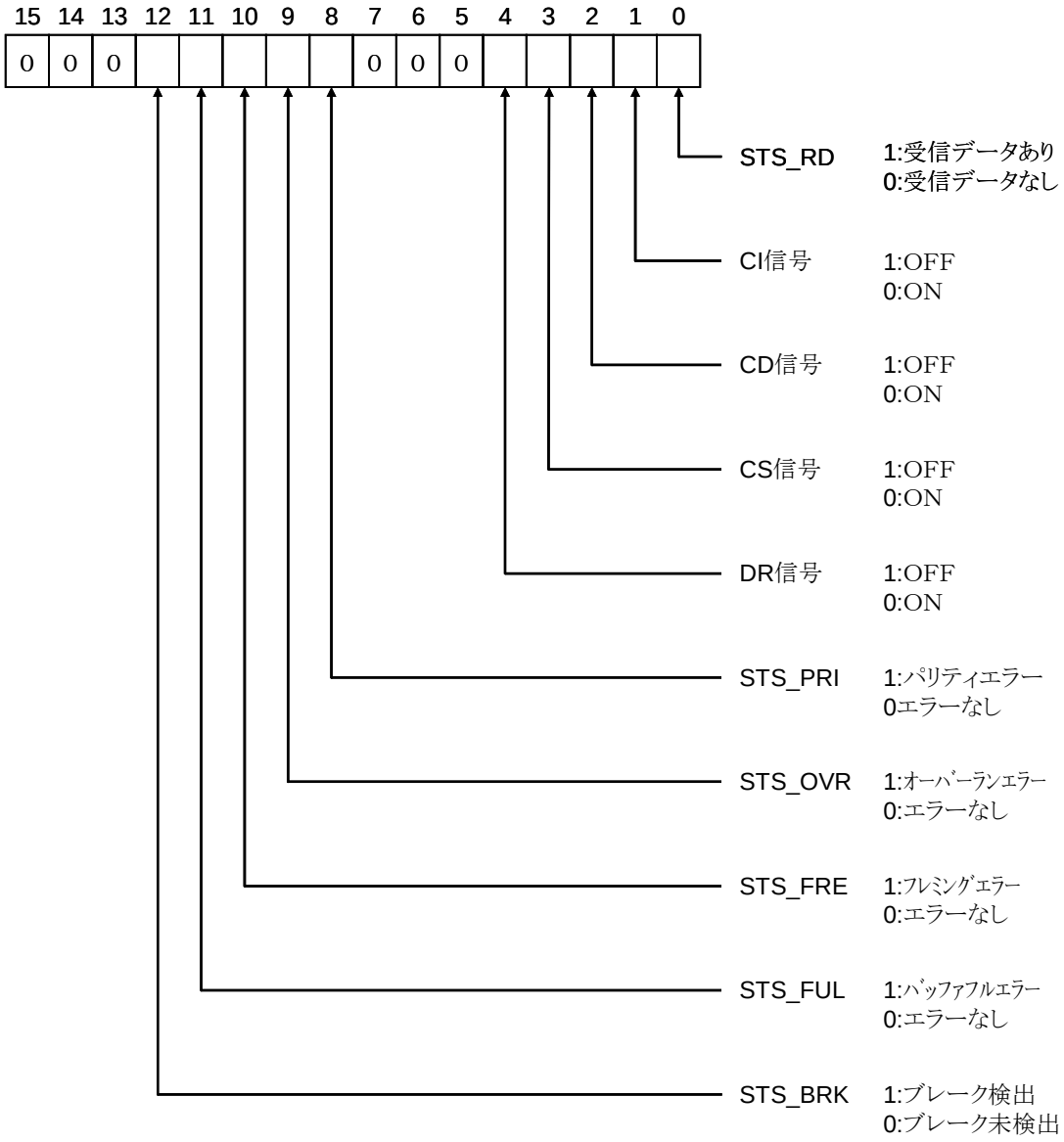
COM0	IR インタフェース

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 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

1.5.8. Ir_c_hold

本関数では、パラメータチェックを行います。(指定された通信ポートに対して、占有または解除は行いません)

```
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

1.5.9. 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();
```

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

1.5.10. 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

1.5.11. 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

1.5.12. Ir_c_tmdin

受信バッファに格納したデータを 1 文字読出します。
受信データが存在しない場合は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。

1.5.13. 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

1.5.14. 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

1.5.15. Ir_c_txrx

本関数では、パラメータチェックを行います。(指定された通信ポートに対して、送受信の有効または無効設定は行いません)

```
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

1.5.16. Ir_c_iobox

本関数では、パラメータチェックを行います。(指定された通信ポートに対して、送信の設定または解除は行いません)

```
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
送信の設定／解除

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

1.5.17. 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

1.5.18. 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

1.5.19. 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

1.5.20. 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

1.5.21. Ir_c_errs

ER および 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

1.5.22. 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

1.5.23. 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

1.5.24. Ir_c_errbfring

本関数では、パラメータチェックを行います。(指定された通信ポートに対して、エラーコードバッファリングの設定は行いません)

```
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

1.5.25. Ir_c_rderrsts

Ir ポートのエラーステータスの読出しおよ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、「デバイス制御ライブラリ リファレンスマニュアル Ir_Err_Get 関数」を参照してください。

1.5.26. Ir_c_chghdr

本関数では、パラメータチェックを行います。(指定された通信ポートに対して、受信ハンドラの切替は行いません)

```
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

1.6. 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。

機能名	説明
ht_fileopen	ファイルオープン
ht_fileclose	ファイルクローズ
ht_fileread	ファイルリード
ht_filewrite	ファイルライト
ht_filesize	ファイルサイズ取得
ht_filelof	レコード数取得

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「DT-700 モード」にしてください。

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 }
};
```

※ 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）
※ 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。

1.6.1. 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「DT-700 モード」にしてください。

1.6.2. 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「DT-700 モード」にしてください。

1.6.3. 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「DT-700 モード」にしてください。

1.6.4. 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「DT-700 モード」にしてください。

1.6.5. 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「DT-700 モード」にしてください。

1.6.6. 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「DT-700 モード」にしてください。

1.7. サービス関数群

プログラム開発に有用な関数群を提供します。

機能名	説明
ht_waitmsec	ウェイト処理
ht_dbgsendmsg	デバッグ補助関数
ht_beep	音制御
xy_modem	XYMODEM
Ir_xy_modem	IrDA 用 XYMODEM

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。

※ 通信速度は最大で、「XYMODEM」では、19200bps、「IrDA 用 XYMODEM」では、115200bps まで設定可能です。

1.7.1. 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

1.7.2. 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 2 重

1.7.3. 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

1.7.4. xy_modem

XMODEM プロトコルおよび 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、モデム操作、プロトコル、エラー検出、パケット長、ポート、'K'送信、ボーレート、メッセージ、ポート操作

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'
ダイヤル信号モード	'T'（トーンダイヤル）／'P'（パルスダイヤル）
記号	'*','#' （トーンダイヤルのみ有効）
区切り	'-','(',')' （ダイヤル時は無視）
ポーズ	',' （ウェイト時間はモデムの設定による）

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およびディレクトリ名を指定してください。

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
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 を禁止してください。

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ください。

※ ファイル名リスト

ファイル名は必ずフルパスで格納してください。

例）

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ください。

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'）をセットしてください。

A:¥¥TEST.C
B:¥¥TEST.DAT
¥0

1.7.5. Ir_xy_modem

IrDA ポートを介して XMODEM プロトコルおよび 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 を指定してください
受信の際は、ファイルを格納するドライブおよびディレクトリ名を指定してください。

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
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
xy_modem()では、通信速度は 19200bps までででしたが、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

2. ビットマップ表示ライブラリ

2.1. 概要

本ライブラリは、BMP ファイルを指定した座標に表示する機能を提供します。

提供ファイル

DTBMP.H	ビットマップ表示関数のヘッダファイル
DTBMP.OBJ	ビットマップ表示関数のオブジェクトファイル

画像が画面をおさまらない場合はエラーを返し、画像を表示しません。
本ライブラリでは、モノクロの画像しか表示できません。

2.2. 関数一覧

本ライブラリは下記の機能を提供します。

機能名	説明
bmp_iDisplayBmpImage	BMP 形式ファイル指定による表示
bmp_iDisplayBmpData	BMP 形式データ指定による表示

提供ファイル

DPLIB.H	インクルードファイル
DPLIB.LIB	ライブラリファイル

2.2.1. bmp_iDisplayBmpImage

指定した BMP 形式のファイルの画像を指定した座標に表示します。(表示座標は、左上が始点になっています)

```
int bmp_iDisplayBmpImage (
    char            *pszBMPFile,
    unsigned char   ucRow,
    unsigned char   ucClm
)
```

パラメータ

pszBMPFile
ファイル名を指定します。

ucRow
表示する y 座標(0～63)を指定します。

ucClm
表示する x 座標(0～127)を指定します。

戻り値

下記のフラグを返します。

BMP_OPERATONOK	正常終了
BMP_NOMEMORY	メモリが足りない
BMP_NOFILEOPEN	ファイルが開けない
BMP_NOFILEREAD	ファイルが読めない
BMP_NOFILESEEK	ファイルのシークができない
BMP_INVALIDTYPE	BMP ファイル内のイメージヘッダ情報、bfType の値が不正である
BMP_INVALIDWIDTH	画像が画面の右にはみ出している
BMP_INVALIDHEIGHT	画像が画面の下にはみ出している
BMP_NEGATIVEHEIGHT	BMP ファイル内のイメージヘッダ情報、bfHeight の値が不正である
BMP_INVALIDPLANES	BMP ファイル内のイメージヘッダ情報、bfPlanes の値が不正である
BMP_INVALIDBITCNT	BMP ファイル内のイメージヘッダ情報、bfBitcnt の値が不正である
BMP_COMPRESSED	BMP ファイル内のイメージヘッダ情報、bfCompressed の値が不正である
BMP_NOPIXELDATA	BMP ファイル内のイメージデータ開始位置が不正である
BMP_INVALIDROW	ucRow の値が不正である
BMP_INVALIDCLM	ucClm の値が不正である

説明

本関数は指定した BMP 形式のファイルの画像を指定した座標に表示します。(表示座標は、左上が始点になっています)

備考

表示できる画像は白黒のみ(白黒以外は BMP_INVALIDBITCNT)です。
画像データの圧縮形式は扱えません(圧縮形式のときは BMP_COMPRESSED)。

2.2.2. bmp_iDisplayBmpData

指定した BMP 形式のデータの画像を指定した座標に表示します。

```
int bmp_iDisplayBmpData (
    char            *pszBMPData,
    unsigned char   ucRow,
    unsigned char   ucClm
)
```

パラメータ

pszBMPFile
BMP 形式のデータの先頭アドレスを指定します。

ucRow
表示する y 座標（0～63）を指定します。

ucClm
表示する x 座標（0～127）を指定します。

戻り値

下記のフラグを返します。

BMP_OPERATONOK	正常終了
BMP_NOMEMORY	メモリが足りない
BMP_NOFILEOPEN	ファイルが開けない
BMP_NOFILEREAD	ファイルが読めない
BMP_NOFILESEEK	ファイルのシークができない
BMP_INVALIDTYPE	BMP ファイル内のイメージヘッダ情報、bfType の値が不正である
BMP_INVALIDWIDTH	画像が画面の右にはみ出している
BMP_INVALIDHEIGHT	画像が画面の下にはみ出している
BMP_NEGATIVEHEIGHT	BMP ファイル内のイメージヘッダ情報、bfHeight の値が不正である
BMP_INVALIDPLANES	BMP ファイル内のイメージヘッダ情報、bfPlanes の値が不正である
BMP_INVALIDBITCNT	BMP ファイル内のイメージヘッダ情報、bfBitcnt の値が不正である
BMP_COMPRESSED	BMP ファイル内のイメージヘッダ情報、bfCompressed の値が不正である
BMP_NOPIXELDATA	BMP ファイル内のイメージデータ開始位置が不正である
BMP_INVALIDROW	ucRow の値が不正である
BMP_INVALIDCLM	ucClm の値が不正である

説明

本関数は指定した BMP 形式のデータの画像を指定した座標に表示します。（表示座標は、左上が始点になっています）
本関数は、bmp_iDisplayBmpImage 関数とほぼ同等です。
bmp_iDisplayBmpImage 関数では、ファイルのデータを直接表示に反映するのに対し、本関数はファイルの内容を RAM に展開し、そのアドレスのデータを表示に反映するものです。

備考

表示できる画像は白黒のみ（白黒以外は BMP_INVALIDBITCNT を返します）です。
画像データの圧縮形式は扱えません(圧縮形式のときは BMP_COMPRESSED を返します)。

3. Bluetooth プリンタライブラリ

3.1. 概要

本ライブラリは、Bluetooth プリンタを操作するためのアプリケーションインターフェースに関するものです。
通信インターフェースは Bluetooth です。
※ 通信制御に関しては、Bluetooth ライブラリを使用します。
※ 使用するプリンタに対するコマンドパケット／ステータスパケット等の詳細仕様に関しては、各々のプリンタの解説書をご覧ください。

提供ファイル

PRN_BT.H	Bluetooth プリンタ関数のヘッダファイル
PRN_BT.OBJ	Bluetooth プリンタ関数のオブジェクトファイル

接続対象のプリンタは、下記の 2 機種です。

- プチラパン（SATO）
- B-SP2D（東芝テック）

3.2. 機能

3.2.1. 通信の接続準備

プリンタと Bluetooth で接続するためには、プリンタの MAC アドレス設定を予め行なう必要があります。本設定を行なうメニューを用意し、運用開始前に必ず設定する必要があります。

(1) 周囲の Bluetooth 機器の検索および設定

次の手順で処理を行ってください。
①Bluetooth の使用を開始(BT_Start)
②Bluetooth 機器の探索(BT_Inquiry)
③Bluetooth 機器のデバイス情報取得(BT_GetDevInfo)
④取得した情報の中から対象となるプリンタを選択
⑤通信する Bluetooth 機器の情報をファイルに保存(BT_SaveDevInfo)
⑥Bluetooth の使用を終了(BT_Stop)

※ 但し、B-SP2D は電源 ON 直後から 1 分間の間しか Inquiry を受け付けません。
※ Inquiry で探索できるのは最大 9 件なので、周りにたくさん Bluetooth 機器が有る場合は対象の機器が出てこない可能性があります。

(2) 固定の MAC アドレスで設定

次の手順で処理を行ってください。
①テスト印字を行なう。(方法は、各プリンタの説明書参照)
②Bluetooth の使用を開始(BT_Start)
③テスト印字内のアドレスを入力
④③で入力した値で Bluetooth 機器のデバイス名取得(BT_GetDevName)
⑤通信する Bluetooth 機器の情報をファイルに保存(BT_SaveDevInfo)
⑥Bluetooth の使用を終了(BT_Stop)

※ B-SP2D では、MAC アドレスを CODE128 のバーコードとして印字できるので、③の部分をバーコード読込に変えることも可能です。

3.2.2. 通信のオープン・クローズ

オープン

Bluetooth 接続処理の中で、BT_Start()と BT_Open()に時間がかかるため、関数の構成としてこれら 2 つの関数は、分けて実行するようにしています。オープン関数の中には、BT_Open()が含まれています。従って、オープン関数を呼ぶ前には、必ず BT_Start()(最短 1.5 秒)を呼ぶ必要があります。
オープン関数の中では、次の処理を行なっています。
①通信する Bluetooth 機器の情報をファイルから取得(BT_LoadDevInfo)
②通信する Bluetooth 機器を選択(BT_SelectDev)
③Bluetooth 機器との接続(BT_Open　最短 3 秒)

クローズ

オープン処理で BT_Start()と BT_Open()を分離しているので、クローズ処理もそれに対応して、BT_Close()と BT_Stop()を分離しています。完全に終了させるためには、クローズ後に BT_Stop()を呼んでください。
クローズ関数の中では、次の処理を行なっています。
①Bluetooth 機器との切断(BT_Close)

3.2.3. 通信の送信・受信

送受信処理では、プリンタにデータ送信後はステータスの要求と受信を行いプリンタの状態監視を行ないます。

データ送信

指定された送信データをプリンタに送ります。
送信関数の中では、次の処理を行なっています。
①指定文字数分のデータを送信(BT_Write)

ステータス受信

STX で始まる文字列を指定文字数分(STX を含む)受信します。
受信関数の中では、次の処理を行なっています。
①指定文字数分のデータを受信(BT_Read)
②受信タイムアウトをチェック

3.2.4. エラー処理

Bluetooth ライブラリから戻るエラーがありますが、そのエラーが発生した場合の対処方法を以下に示します。

No	エラー値	対処方法	備考
1	BTERR_LOCK	IrDA を終了させる	
2	BTERR_DISCONNECT	切断処理後、接続し、リトライする	
3	BTERR_PARAMETER	コーディングを見直す(発生した関数のパラメータを確認)	
4	BTERR_BREAK_EVNT	切断処理をする	
5	BTERR_LB0	次回電源 ON で切断処理後、再度接続し、リトライする	
6	BTERR_LB1	切断処理後本体電源を OFF、電池交換して電源 ON 後、接続し、リトライする	
7	BTERR_LB2		
8	BTERR_LB4	操作中には発生しません	
9	BTERR_LB5	次回電源 ON で切断処理後、再度接続し、リトライする	
10	BTERR_NOTSTART	コーディングを見直す(prn_BTinf_open を呼んでいるか確認)	
11	BTERR_TIMEOUT	プリンタの状態(電源など)を確認してからリトライする	
12	BTERR_PARITY	ほとんど発生しないが、発生した場合は切断処理後、再度接続し、リトライする	
13	BTERR_OVERRUN		
14	BTERR_FRAMING		
15	BTERR_TRANSFER		
16	BTERR_FILEOPEN	コーディングを見直す	
17	BTERR_FILEACCESS		
18	BTERR_NAK00	ほとんど発生しないが、発生した場合はリトライする	
19	BTERR_NAK01		
20	BTERR_NAK02	コーディングを見直す(BT_SetPassKey を呼んでいるか確認)	
21	BTERR_NAK03	プリンタの状態(電源など)を確認してからリトライする	
22	BTERR_NAK04		
23	BTERR_NAK05	PIN コードの設定を確認する	
24	BTERR_NAK06	Bluetooth 故障の可能性あり	※
25	BTERR_NAK08	ほとんど発生しないが、発生した場合は BT_Stop を呼ぶ	
26	BTERR_NAK09		
27	BTERR_NAK10	Bluetooth 故障の可能性あり	※
28	BTERR_NAK11	コーディングを見直す(デバイス名の長さを確認)	
29	BTERR_NAK12	ほとんど発生しないが、発生した場合は BT_Stop を呼ぶ	
30	BTERR_NAK13	Bluetooth 故障の可能性あり	※
31	BTERR_NAK14	ほとんど発生しないが、発生した場合はリトライする	
32	BTERR_NAKMINUS1	Bluetooth 故障の可能性あり	※

※ リトライしてもつながらない場合は、保守に連絡してください。

3.3. 関数一覧

本ライブラリは下記の機能を提供します。

機能名	説明
prn_BTinf_open	Bluetooth 通信のオープン
prn_BTinf_close	Bluetooth 通信のクローズ
prn_BTinf_send	Bluetooth でのコマンド送信
prn_BTinf_recvSts	Bluetooth でのステータス受信

3.3.1. prn_BTinf_open

Bluetooth を占有します。

```
H prn_BTinf_open (
    B    *buff,
    H    tout,
    UH   len,
    B    *fname
)
```

パラメータ

buff
受信バッファアドレスを指定します。
最低でも 52byte の領域を確保してください。

tout
装着待ちタイムアウト時間（1～3600 秒）を指定します。

ucClm
受信バッファレングスを指定します。

fname
デバイス情報格納ファイル名を指定します。

戻り値

下記の値を返します。

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー
E_PRN_PON	オープン済み

説明

本関数は Bluetooth を占有します。
本関数を呼ぶ前に必ず BT_Start を呼んでください。
本関数が E_NG を返す場合は BT_Err_Get で詳細なエラーステータスを確認してください。

3.3.2. prn_BTinf_close

Bluetooth の占有を解除します。

```
H prn_BTinf_open ()
```

パラメータ

なし

戻り値

下記の値を返します。

E_OK	正常終了
E_NG	異常終了
E_PRN_POFF	未オープン

説明

本関数は Bluetooth の占有を解除します。
完全に終了するためには、本関数呼出し後、BT_Stop を呼んでください。
本関数が E_NG を返す場合は、BT_Err_Get で詳細なエラーステータスを確認してください。

3.3.3. prn_BTinf_send

プリンタに対して指定レングス分のコマンドを送信します。

```
H prn_BTinf_send (
    B       *command_ptr,
    UH      len
)
```

パラメータ

command_ptr
送信するコマンドのアドレスを指定します。

len
コマンドレングスを指定します。

戻り値

下記の値を返します。

E_OK	正常終了
E_NG	異常終了
E_PRN_POFF	未オープン

説明

本関数はプリンタに対して指定レングス分の各種コマンドを送信します。
本関数が E_NG を返す場合は BT_Err_Get で詳細なエラーステータスを確認してください。

3.3.4. prn_BTinf_recvSts

プリンタから送られてくる、プリンタステータスを受信するために使用します。

```
H prn_BTinf_recvSts (
    UH      len,
    H       tout
)
```

パラメータ

len
受信予定データ数（STX を含むデータ数）を指定します。

tout
装着待ちタイムアウト時間（1～3600 秒）を指定します。

戻り値

下記の値を返します。

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー
E_PRN_POFF	未オープン
E_PRN_TIMEOUT	受信タイムアウト

説明

本関数はプリンタから送られてくるプリンタステータスを受信するために使用します。
prn_BTinf_open()で指定したバッファにデータを受信します。
指定するレングスは、STX からの受信予定データ数を指定します。
本関数が E_NG を返す場合は BT_Err_Get で詳細なエラーステータスを確認してください。

4. 高速ファイルサーチライブラリ

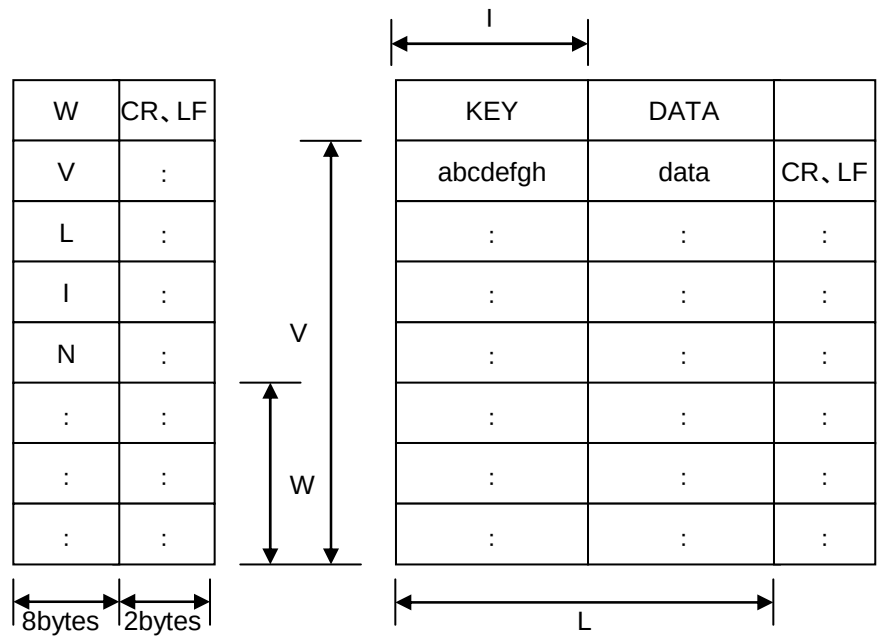
4.1. 概要

本高速ファイル検索ライブラリは、HASH 法を使用した高速ファイル検索機能セットです。
HASH 法とは、キーデータを数値に変換した特別なインデックスファイルを用い、データレコード検索時間を最小限にする技法です。
本ライブラリを用いることにより、特に大容量ファイルを扱う場合は、従来の逐次検索に比べ、高速にファイルを検索することができます。

提供ファイル

HASHLIB.H	高速ファイルサーチライブラリのヘッダファイル
HASHLIB.LIB	高速ファイルサーチライブラリのライブラリファイル

4.1.1. ファイル構造



INDEX FILE(*.idx)　　DATA FILE

I　:キーフィールドの長さ (Bytes).
L　:データレコード長 (KEY+DATA)
V　:総データレコード数
W　:総インデックス数
　　インデックスの衝突を避けるために、最低でも W≧1.2×V が必要です。
　　W を V の 2～3 倍程度確保すると、より効率の良いファイルが構築できます。
N　:使用レコード数

※ データファイルは、上記フォーマットに従い、ユーザ側で作成しなければなりません。
※ データレコードは、データファイルの先頭から順に格納されていなければなりません。
※ キーは、ユニークでなければなりません。
※ インデックスファイルおよびデータファイルの OPEN／CLOSE は、ユーザ側で行ってください。
※ DT-900 のドライブ（特に B ドライブ）は、デバイスの特性上、書き込み速度が遅いため HashRead 関数以外（特に HashAssign 関数）は、PC 上で実行することを強く推奨します。

4.2. 関数一覧

本ライブラリは下記の機能を提供します。

機能名	説明
HashAssign	データファイルからインデックスファイルを生成します。
HashRead	入力されたキーに該当するデータをデータファイルから読み出します。
HashWrite	入力されたキーに該当するデータの内容を書き換えます。
HashAdd	データファイルにデータを追加します。 （同時にインデックスファイルも更新します）

4.2.1. iHashAssign

データファイルからインデックスファイルを作成します。

```
int iHashAssign (
    FILE    *DataFilePointer,
    FILE    *IndexFilePointer,
    long    lI,
    long    lV,
    long    lL,
    long    lW1
)
```

パラメータ

DataFilePointer

fopen 関数によりオープンしたデータファイルポインタを指定します。
本関数を使用する前に、データファイルは下記のようにオープンしていなければなりません。

```
#include <stdio.h>
FILE *DataFilePointer;
DataFilePointer = fopen (char* file,char* mode);
file     :  データファイル名
mode     :  ファイルアクセスモード (“rb”)
```

IndexFilePointer

fopen 関数によりオープンしたインデックスファイルポインタ。
本関数を使用する前に、データファイルは下記のようにオープンしていなければなりません。

```
#include <stdio.h>
FILE *IndexFilePointer;
IndexFilePointer = fopen (char* file,char* mode);
file     :  インデックスファイル名
mode     :  ファイルアクセスモード (“wb+”)
```

lI, lV, lL, lW1

これらの値は、インデックスファイルのヘッダ部に格納され、下記のような意味を持ちます。

lI　:データファイルのキーフィールド長
lV　:データファイルの総レコード数
　　ただし、将来データ数が増える見込みがある場合は、現在のデータファイルの実際の総レコード数よりも大きい値を指定します。
lL　:データファイルのレコード長(キー＋データ)
lW1　:インデックスファイルの総インデックス数(ヘッダ情報除く)
　　(通常 、 lW1 >= lV * 1.2　上記変数は、long 型です。)

戻り値

下記の値を返します。

0 (HASH_OPERATIONOK)	正常終了
102 (DATA_NOFILEREAD)	データファイルリードエラー
105 (DATA_NOFILEOPEN)	データファイル未オープン
111 (DATA_NOFILESEEK)	データファイルシークエラー
202 (IDX_NOFILEREAD)	インデックスファイルリードエラー
203 (IDX_NOFILEWRITE)	インデックスファイルライトエラー
205 (IDX_NOFILEOPEN)	インデックスファイル未オープン
211 (IDX_NOFILESEEK)	インデックスファイルシークエラー
320 (HASH_NOMEMORY)	実行メモリ不足
400 (PARAM_INVALID)	パラメータエラー

説明

本関数は、データファイルからインデックスファイルを作成します。

4.2.2. iHashRead

指定した入力キーデータに対応するデータを検索します。

```
int iHashRead (
    char    *pszKey,
    FILE    *DataFilePointer,
    FILE    *IndexFilePointer,
    char    *pszBuff
)
```

パラメータ

pszKey

検索するキーデータのポインタを指定します。
キーデータの末尾には、NULL 文字を入れてください。
キーデータ長が、データファイル内の実際のキーデータ長と一致しない場合は、エラーとなります。

DataFilePointer

fopen 関数によりオープンしたデータファイルポインタを指定します。
本関数を使用する前に、データファイルは下記のようにオープンしていなければなりません。

```
#include <stdio.h>
FILE *DataFilePointer;
DataFilePointer = fopen (char* file,char* mode);
file     :  データファイル名
mode     :  ファイルアクセスモード (“rb”)
```

IndexFilePointer

fopen 関数によりオープンしたインデックスファイルポインタ。
本関数を使用する前に、データファイルは下記のようにオープンしていなければなりません。

```
#include <stdio.h>
FILE *IndexFilePointer;
IndexFilePointer = fopen (char* file,char* mode);
file     :  インデックスファイル名
mode     :  ファイルアクセスモード (“wb+”)
```

pszBuff

入力されたキーと一致するデータレコードが格納されるバッファポインタを指定します。
(ただし、データレコードには、キーは含まれません)
データレコードの最終には、NULL 文字が付加されます。
もし、入力されたキーと一致するデータレコードが見つからない場合は NULL ポインタを返します。

戻り値

下記の値を返します。

0 (HASH_OPERATIONOK)	正常終了
102 (DATA_NOFILEREAD)	データファイルリードエラー
105 (DATA_NOFILEOPEN)	データファイル未オープン
111 (DATA_NOFILESEEK)	データファイルシークエラー
202 (IDX_NOFILEREAD)	インデックスファイルリードエラー
205 (IDX_NOFILEOPEN)	インデックスファイル未オープン
211 (IDX_NOFILESEEK)	インデックスファイルシークエラー
320 (HASH_NOMEMORY)	実行メモリ不足
330 (HASH_KEYNOTFOUND)	該当インデックスなし
331 (HASH_INVALIDKEY)	不正キー入力（キー長不一致）
332 (HASH_KEYNULL)	キーポインタ不正 (NULL)

説明

本関数は、指定した入力キーデータに対応するデータを検索します。

4.2.3. iHashWrite

指定キーのデータレコードを書き換えます。

```
int iHashWrite (
    char    *pszKey,
    FILE    *DataFilePointer,
    FILE    *IndexFilePointer,
    char    *pszBuff
)
```

パラメータ

pszKey

変更するデータのキーデータポインタを指定します。
キーデータの末尾には、NULL 文字を入れてください。
キーデータ長が、データファイル内の実際のキーデータ長と一致しない場合は、エラーとなります。

DataFilePointer

fopen 関数によりオープンしたデータファイルポインタを指定します。
本関数を使用する前に、データファイルは下記のようにオープンしていなければなりません。

```
#include <stdio.h>
FILE *DataFilePointer;
DataFilePointer = fopen ( char* file, char* mode);
file     :  データファイル名
mode     :  ファイルアクセスモード ( “rb”)
```

IndexFilePointer

fopen 関数によりオープンしたインデックスファイルポインタ。
本関数を使用する前に、データファイルは下記のようにオープンしていなければなりません。

```
#include <stdio.h>
FILE *IndexFilePointer;
IndexFilePointer = fopen ( char* file, char* mode);
file     :  インデックスファイル名
mode     :  ファイルアクセスモード ( “wb+”)
```

pszBuff

書き換えるデータのポインタ(キーは、含みません)を指定します。
先頭から、NULL 文字までをデータとみなします。
データ長は、インデックスファイル作成時に指定した値(IL-II)と一致していなければなりません。
一致しない場合は、エラーとなります。

戻り値

下記の値を返します。

0 (HASH_OPERATIONOK)	正常終了
102 (DATA_NOFILEREAD)	データファイルリードエラー
103 (DATA_NOFILEWRITE)	データファイルライトエラー
105 (DATA_NOFILEOPEN)	データファイル未オープン
111 (DATA_NOFILESEEK)	データファイルシークエラー
202 (IDX_NOFILEREAD)	インデックスファイルリードエラー
205 (IDX_NOFILEOPEN)	インデックスファイル未オープン
211 (IDX_NOFILESEEK)	インデックスファイルシークエラー
320 (HASH_NOMEMORY)	実行メモリ不足
330 (HASH_KEYNOTFOUND)	該当インデックスなし
331 (HASH_INVALIDKEY)	不正キー入力（キー長不一致）
332 (HASH_KEYNULL)	キーポインタ不正（NULL）
333 (HASH_INVALIDDATA)	不正データ入力（データ長不一致）
334 (HASH_DATANULL)	データポインタ不正（NULL）

説明

本関数は、指定した入力キーデータに対応するデータを検索します。

4.2.4. iHashAdd

データファイルに新しいレコード(キー＋データ)を追加します。

```
int iHashAdd (
    char    *pszKey,
    FILE    *DataFilePointer,
    FILE    *IndexFilePointer,
    char    *pszAppendData
)
```

パラメータ

pszKey

追加するキーデータのポインタを指定します。
キーデータの末尾には、NULL 文字を入れてください。
キーデータ長が、データファイル内の実際のキーデータ長と一致しない場合は、エラーとなります。

DataFilePointer

fopen 関数によりオープンしたデータファイルポインタを指定します。
本関数を使用する前に、データファイルは下記のようにオープンしていなければなりません。

```
#include <stdio.h>
FILE *DataFilePointer;
DataFilePointer = fopen ( char* file, char* mode);
file     :  データファイル名
mode     :  ファイルアクセスモード ( “rb”)
```

IndexFilePointer

fopen 関数によりオープンしたインデックスファイルポインタ。
本関数を使用する前に、データファイルは下記のようにオープンしていなければなりません。

```
#include <stdio.h>
FILE *IndexFilePointer;
IndexFilePointer = fopen ( char* file, char* mode);
file     :  インデックスファイル名
mode     :  ファイルアクセスモード (“wb+”)
```

pszAppendData

追加するレコードのポインタを指定します。
レコードの末尾には、NULL を入れてください。
レコード長がデータファイルの実際のレコード長と一致しない場合は、エラーとなります。

戻り値

下記の値を返します。

0 (HASH_OPERATIONOK)	正常終了
103 (DATA_NOFILEWRITE)	データファイルライトエラー
104 (DATA_FILEOVERFLOW)	データファイルフル
105 (DATA_NOFILEOPEN)	データファイル未オープン
111 (DATA_NOFILESEEK)	データファイルシークエラー
202 (IDX_NOFILEREAD)	インデックスファイルリードエラー
203 (IDX_NOFILEWRITE)	インデックスファイルライトエラー
205 (IDX_NOFILEOPEN)	インデックスファイル未オープン
211 (IDX_NOFILESEEK)	インデックスファイルシークエラー
320 (HASH_NOMEMORY)	実行メモリ不足
330 (HASH_KEYNOTFOUND)	該当インデックスなし
331 (HASH_INVALIDKEY)	不正キー入力（キー長不一致）
332 (HASH_KEYNULL)	キーポインタ不正（NULL）
333 (HASH_INVALIDDATA)	不正データ入力（データ長不一致）
334 (HASH_DATANULL)	データポインタ不正（NULL）

説明

本関数は、データファイルに新しいレコード（キー＋データ）を追加します。
1 度に追加できるレコードは 1 レコードのみで、既存レコードの末尾に追加されます。
（既存レコードの間に追加することはできません）
データファイルがオーバーフローしていないこと、データレコードの数がインデックスヘッダーに指定している数より少ないことが条件です。

5. TEC IrDA プリンタライブラリ

5.1. 概要

本ライブラリは、ポータブルプリンタを操作するための機能を提供します。
本ライブラリは、赤外線通信を対象としています。
赤外線インタフェースでは、プリンタのすべてのコマンドの使用が可能になります。
インタフェースの使用手順は、C 言語の関数群としてアプリケーションに提供します。
通信部は、通信 BIOS を使用します。

提供ファイル

BZIIDEF.H	TEC IrDA プリンタ関数のヘッダファイル
BZIX9.OBJ	TEC IrDA プリンタ関数のオブジェクトファイル

5.2. 関数一覧

本ライブラリは下記の機能を提供します。

機能名	説明
prn_tecinf_open	赤外線通信のオープン
prn_tecinf_close	赤外線通信のクローズ
prn_tecinf_send	赤外線でのコマンド送信
prn_tecinf_status	赤外線でのステータスリード
prn_tecinf_send2	赤外線でのコマンド送信(タイムアウト付き)

5.2.1. prn_tecinf_open

赤外線通信のオープンを行います。

```
ER prn_tecinf_open (
    B    *buff
)
```

パラメータ

buff
受信バッファアドレスを指定します。
最低でも 24byte の領域を確保してください。

戻り値

下記の値を返します。

E_OK	正常終了
E_PRN_PON	オープン済み
E_MBU_EXC	排他エラー
E_PRN_LB	ローバッテリー

説明

本関数は COM0 をオープンし、占有します。
COM0 の初期化および COM0 を IrDA に切り替えます。

- IrDA インタフェース電源を使用可
- 通信速度を 19200bps に設定
- 割り込み禁止

5.2.2. prn_tecinf_close

赤外線通信をクローズします。

```
ER prn_tecinf_close ()
```

パラメータ

なし

戻り値

下記の値を返します。

E_OK	正常終了
E_PRN_POF	クローズ済み
E_PRN_LB	ローバッテリー （ただし、ローバッテリーの場合でも赤外線をクローズします）

説明

本関数は COM0 をクローズし、占有を解除します。

5.2.3. prn_tecinf_send

プリンタに対して指定レングス分のコマンドを送信します。

```
ER prn_tecinf_send (
    B       *command_prt,
    UB      status
)
```

パラメータ

command_prt
送信するコマンドのアドレスを指定します。

status
通信の継続／終了を指定します。

P_CONTINUE	継続
P_END	終了

戻り値

下記の値を返します。

E_OK	正常終了
E_TIMEOUT	受信タイムアウト
E_PRN_ERR	エラーステータス受信 （STX と CRC を除いたもの）
E_NG	異常終了
E_PRN_POF	未オープン
E_PRN_LB	ローバッテリー
E_PRN	パラメータエラー

説明

本関数はプリンタに対して各種コマンドの送信を行います。
本関数では、プリンタが受け付けたコマンドの実行を開始し、処理終了時にハンディターミナルに返す ACK の待ち時間は 12 秒固定となっています。
印字コマンドなどで、印字開始から、印字終了までの時間が 12 秒以上になる場合は、タイムアウト指定付きの prn_tecinf_send2 を使用してください。

5.2.4. prn_tecinf_send2

プリンタに対して指定レングス分のコマンドをタイムアウト付きで送信します。

```
ER prn_tecinf_send2(
    B       *command_prt,
    UB      status,
    H       end_wait
)
```

パラメータ

command_prt
送信するコマンドのアドレスを指定します。

status
通信の継続／終了を指定します。

P_CONTINUE	継続
P_END	終了

end_wait
送信コマンドの処理終了待ち時間を 1～255（秒単位）で指定します。

戻り値

下記の値を返します。

E_OK	正常終了
E_TIMEOUT	受信タイムアウト
E_PRN_ERR	エラーステータス受信 （STX と CRC を除いたもの）
E_NG	異常終了
E_PRN_POF	未オープン
E_PRN_LB	ローバッテリー
E_PRN	パラメータエラー

説明

本関数はプリンタに対して各種コマンドの送信を行います。
プリンタが受け付けたコマンドの実行を開始し、処理終了時にハンディターミナルに返す ACK の待ち時間を指定できます。
指定した時間内にプリンタがコマンド処理終了の ACK を返せない場合は、受信タイムアウトを返します。
（印字コマンドでは、印字開始から、印字終了までの時間は、周囲温度、バッテリ電圧、印字長、印字デューティなどによって変わりますので、出力状態に合った適切な時間を設定する必要があります。周囲温度による影響は下記備考を参考にして計算してください。）

備考

温度による印字速度の変化は下記の規定です（電池電圧=8.4V、ヘッド抵抗値=170Ωの場合）。

印字ヘッド温度	印字速度
-5～0℃	0.6～0.8 インチ／秒
0～20℃	0.8～2.0 インチ／秒
20℃～	2.0 インチ／秒

5.2.5. prn_tecinf_status

ステータスコマンドを送信し、プリンタの状態を受信します。

```
ER prn_tecinf_status (
    B       *recv_status,
    UB      status
)
```

パラメータ

recv_status
受信バッファアドレスを指定します。
最低でも 24byte の領域を確保してください。

status
通信の終了を指定します。

P_END	終了

戻り値

下記の値を返します。

E_OK	正常終了
E_TIMEOUT	受信タイムアウト
E_NG	異常終了
E_PRN_POF	未オープン
E_PRN_LB	ローバッテリー
E_PRN	パラメータエラー
E_CRC	CRC エラー

説明

本関数はステータスリードコマンドを送信し、プリンタの状態を受信します。
リターンするステータスは、STX と CRC を除いたものです。

備考

下表に受信したステータスのプリンタ状態を示します。

状態コード	プリンタ状態	対応
00H	通常状態（アイドル中）	印刷できます
01H	カバーオープン状態	プリンタのカバーを閉めてください
02H	コマンドシンタックスエラー	コマンドの設定に誤りがあります
03H	フィードジャムー	コマンドの設定に誤りがあります
04H	ラベルエンド	コマンド送信の順序に矛盾があります
05H	カバーオープンエラー	プリンタカバーを閉めてください
06H	サマールヘッド切断エラー	プリンタの故障で修理が必要です
07H	サマールヘッド異常高温エラー	ヘッド温度が冷めるのを待ってから再印字してください
08H	フラッシュ ROM 書き込みエラー	プリンタの故障で修理が必要です
09H	フラッシュ ROM 消去	プリンタの故障で修理が必要です
0AH	ローバッテリー（印字不可状態）	バッテリーの充電が必要です
0BH	動作中	動作中です

5.3. プリンタコマンド

ラベル発行モード

ラベル発行モードの記述内容は、プリンタ ID、レングス、プリンタコマンドです。
下図に prn_tecinf_send でのコマンド記述形式を示します。

```
static char Lprn_comm[] =
    { 0x00, 0x00, 0x06, 0x1B, 'M', ';', '0', 0x0A, 0x00 };
```

プリンタID　　プリンタコマンド

送信データレングス

レシート発行モード

レシート発行モードの記述内容は、プリンタ ID、レングス、モード、フラグ、プリンタコマンドです。
下図に prn_tecinf_send でのコマンド記述形式を示します。

```
static char Rprn_comm[] =
    { 0x03, 0x51, 0x03, 'Y', 0x81, 0x1B, '3', 0x1E };
```

プリンタID　　プリンタコマンド

フラグ

送信データレングス　　モード

6. 送受信切替ライブラリ

6.1. 概要

本機で提供している通信関数で、カシオ IR ポートを使用して通信する場合は、送信／受信のタイミングで、送受信方向の切替えが必要です。
オープン直後は受信可能状態（送信不可）となっているため、送信時には、送信可能状態に切り替える必要があります。

本オブジェクトは、上記の「送受信切替え」を自動的に行うものであり、これにより、従来機(DT-700)で動作しているアプリケーションの制御をそのまま使用することが可能となります。

機能
c_out/c_dout 関数に以下の機能を付加します

- 送信関数を実行すると、自動的に送信可能状態にします。
- 送信終了時に、自動的に受信状態に戻します。

※ 本機能は「カシオ IR ポート:COM0」に対応しています。COM0 以外のポートは、従来機同等で「全二重（送受信可能状態）」で動作します。

既に提供している関数「c_out/c_dout」をそのまま使用可能であり、インタフェース／リターン情報共に、上記関数と同等のものです。

提供ファイル

COM_APLM.H	送受信自動切替え関数のヘッダファイル
COM_APLM.OBJ	送受信自動切替え関数のオブジェクトファイル

使用方法
c_out/c_dout 関数の仕様は、従来機のものと同等ですので、プログラムの実行部に対する修正は不要です。

下記の修正を行ってください。

1. アプリケーションに以下のファイルをインクルードしてください。

- BIOS1MAC.H
- COM_APLM.H

※ COM_APLM.H は必ず BIOS1MAC.H の前にインクルードしてください。

2. アプリケーションのリンク時に以下のファイルをリンクしてください。

- COM_APLM.OBJ

3. アプリケーションソースに修正を加え、ビルドしてください。

7. IOBOX 検出切替ライブラリ

7.1. 概要

DT500 プロトコルは IO ボックスとの IR 通信を前提としているため、通信中に IO ボックス検出を行っており、通常、検出されない場合は通信できません。
本関数は、DT500 プロトコルによる、本機と MCU60 との IR 通信を可能にするため、IO ボックス検出機能の有効／無効を切り替える機能を提供します。

MCU60

- SS&T 社製 IO ボックス
- プロトコル：DT500 プロトコル（ROM 内容の交換により他のプロトコルバージョンあり）
- プリンタ内蔵
- モデム内蔵
- RS-232C によるホスト接続

機能
IO BOX の検出を有効/無効にします。

有効：IO ボックス検出を行う　　（BASIC IO 使用時）
無効：IO ボックス検出を行わない（MCU60使用時）

※ pwr_inhabit()等の「IO ボックス検出」機能には影響しません。プロトコル上での IO ボックス検出の切替えを行うのみです。
※ 本関数で設定した内容は、リセット／レジューム OFF で初期化(有効状態)となります。

提供ファイル

IO_APLM.H	IO BOX 検出切替え関数のヘッダファイル
IO_APLM.OBJ	IO BOX 検出切替え関数のオブジェクトファイル

使用方法
1. アプリケーションに下記のファイルをインクルードし、「io_detect 関数」で IO BOX 検出の切替えを行ってください。

- IO_APLM.H

2. アプリケーションのリンク時に以下のファイルをリンクしてください。

- IO_APLM.OBJ

3. アプリケーションソースに修正を加え、ビルドしてください。

7.2. 関数一覧

本ライブラリは下記の機能を提供します。

機能名	説明
io_detect	IO BOX 検出の有効／無効を切替え

7.2.1. io_detect

「マルチドロッププロトコル」「DT500 プロトコル」時の IO BOX 検出の有効／無効を切り替えます。

```
ER io_detect(
    UB      mode
)
```

パラメータ

mode

IO BOX 検出の有効／無効を下記の値で指定します。

IO_ENA　IO BOX 検出を有効にします

IO_DIS　IO BOX 検出を無効にします

戻り値

切替えが正常に終了した場合は E_OK を、エラー終了した場合は ER_PRM を返します。

補足

本関数により設定した状態は、リセット／レジューム OFF により、強制的に「検出有効」となります。

カシオ計算機お問い合わせ窓口

製品に関する最新情報

●製品サポートサイト（カシオペア・ハンディターミナル）

http://casio.jp/support/pa/

製品の取扱い方法のお問い合わせ

●情報機器コールセンター

0570-022066
市内通話料金でご利用いただけます。
携帯電話・PHS 等をご利用の場合、03-5294-7251

カシオ計算機株式会社
〒151-8543　東京都渋谷区本町 1-6-2
TEL 03-5334-4638(代)