CASIO

DT-970SDK

拡張機能ライブラリ
リファレンスマニュアル

DT-970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. 1.03

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
1.00	2013.10		新規作成
1.01	2013.12	6,19,29, 111	ht_ShiftMode 関数を追加
		36	Ir_c_open 関数の説明を修正
		74	Bluetooth ライブラリのリンクに関する補足説明を追加
		80-83	戻り値（E_NG）の説明を修正
		80-83	戻り値のデータ型を H から ER に訂正
		84,113	TEC IrDA プリンタライブラリの削除
1.02	2014.07	115	裏表紙を変更
1.03	2019.03	106	iHashRead 関数のパラメータ説明を修正

目次

1. アプリケーション支援ライブラリ

1.1. 概要

DT-970 のアプリケーションプログラム開発は、DT-970 専用関数と RX ファミリ C 言語標準関数（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。

1.1.1. 提供ファイル

下記のファイルを提供します。

ファイル名	内容
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

1.1.2. 機能一覧

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
日付チェック	日付妥当性チェック 閏年チェック 西暦／和暦変換
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 文字列入力 数値入力 日付入力 時刻入力 シフトキー制御
通信	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 Ir 制御用 XYMODEM 通信

1.2. 日付チェック関数

1.2.1. 関数一覧

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。

関数名	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

1.2.2. ht_CheckDate

日付の妥当性をチェックします。

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

パラメータ

year

チェックする年を西暦(1868～2088)で指定します。

month

チェックする月(1～12)を指定します。

day

チェックする日付(1～31)を指定します。

戻り値

下記の値を返します。

E_OK	:正常(妥当な日付)
E_NG	:不正な日付

説明

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。
範囲外(1868 年～2088 年以外)、または存在しない日付を指定すると、E_NG(不正な日付)を返します。

使用例

```
#include "dplib.h"
   :
   :
 H    year, month, day;
 ER   ercd;
 year=2013;month=2;day=28;
 ercd=ht_CheckDate( year, month, day);      /* 2013 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);
 }
   :
```

1.2.3. ht_CheckYear

入力年が閏年か否かを判定します。

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

パラメータ

year

チェックする年を西暦（1868～2088）で指定します。

戻り値

下記の値を返します。

E_OK	:閏年
E_NG	:通常年
E_PRM	:対象範囲外

説明

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。
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。

使用例

```
#include "dplib.h"
   :
   :
 H    year;
 ER   ercd;
 year=2013;
 ercd=ht_CheckYear( year);     /* 2013 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);
 }
   :
```

1.2.4. 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。

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

パラメータ

in_year

変換する年を指定します。

gengou

元号(0:西暦、1:明治、2:大正、3:昭和、4:平成)を指定します。

out_year

変換された年を格納するバッファのポインタを指定します。

out_gengou

変換された元号(0:西暦、1:明治、2:大正、3:昭和、4:平成)を格納するバッファのポインタを指定します。

戻り値

下記の値を返します。

E_OK	:正常終了
E_NG	: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

説明

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。

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。

使用例

```
#include "dplib.h"
  :
H       year, gengou, out_year, out_gengou;
ER      ercd;
char    msg[33];
  :
  :
year=2013;
gengou = 0;
ercd=ht_CheckYear( year, gengou, &out_year, &out_gengou);     /* 西暦 2013 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=25;
gengou = 4;
ercd=ht_CheckYear( year, gengou, &out_year, &out_gengou);     /* 平成 25 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

1.3.1. 関数一覧

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。

関数名	説明
ht_Checksumm	チェックサム計算
ht_CalcCRC_ANSI	CRC 計算（ANSI 規格）
ht_CalcCRC_CCITT	CRC 計算（CCITT 規格）
ht_CalcCRC_X	CRC 計算（XMODEM 用）
ht_CalcLRC	LRC 計算
ht_CheckCD	チェックデジット計算

1.3.2. ht_Checksumm

指定したデータのチェックサム値を求めます。

```
UH  ht_Checksumm (
  unsigned char  *check_data,
  H              size
)
```

パラメータ

check_data

対象のデータを指定します。

size

データ長を指定します。

戻り値

チェックサム値を返します。

使用例

```
#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 */
   :
```

1.3.3. ht_CalcCRC_ANSI

指定したデータの ANSI 規格の CRC 値を求めます。

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

パラメータ

check_data

対象のデータを指定します。

size

データ長を指定します。

戻り値

ANSI 規格の CRC 値を返します。

使用例

```
#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  */
   :
```

1.3.4. ht_CalcCRC_CCITT

指定したデータの CCITT 規格の CRC 値を求めます。

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

パラメータ

check_data

対象のデータを指定します。

size

データ長を指定します。

戻り値

CCITT 規格の CRC 値を返します。

使用例

```
#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  */
   :
```

1.3.5. ht_CalcCRC_X

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

対象のデータを指定します。

size

データ長を指定します。

戻り値

XMODEM 用の CRC 値を返します。

使用例

```
#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 */
   :
```

1.3.6. ht_CalcLRC

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

対象のデータを指定します。

size

データ長を指定します。

戻り値

LRC 値を返します。

使用例

```
#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 */
   :
```

1.3.7. ht_CheckCD

指定したバーコードデータのチェックデジット計算を行います。

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

パラメータ

check_data

対象のデータを指定します。

mode

下記のバーコード種別を指定します。

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

下記の値を返します。

E_OK	:CD 正常
E_NG	:CD 異常

使用例

```
#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

1.4.1. 関数一覧

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。

関数名	説明
ht_FCWait	制御キー入力
ht_StrInp	文字列入力
ht_NumInp	数値入力
ht_DateInp	日付入力
ht_TimeInp	時刻入力
ht_ShiftMode	シフトキー制御

1.4.2. ht_FCWait

指定した脱出条件になるまで、キー入力を待ちます。

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

パラメータ

escape

下記の脱出条件を指定します。

OR 論理で複数の指定が可能です。

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		
KY_IO	:クレードル検出／USB 接続検出		

戻り値

発生した脱出条件、もしくは下記の値を返します。

E_NG	:異常終了
E_PRM	:パラメータエラー

1.4.3. ht_StrInp

文字列の入力を行います。

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

パラメータ

escape

下記の脱出条件を指定します。
OR 論理で複数の指定が可能です。

KY_F01	:F1 キー押下	KY_ENT	:実行キー押下
KY_F02	:F2 キー押下	KY_LB	:LB 発生
KY_F03	:F3 キー押下	KY_IO	:クレードル検出／USB 接続検出
KY_F04	:F4 キー押下	KY_OBR	:バーコード読み込み検出
KY_F05	:F5 キー押下	KY_FULL	:入力領域フル
KY_F06	:F6 キー押下		
KY_F07	:F7 キー押下		
KY_F08	:F8 キー押下		

fsize

表示する文字列のフォントサイズを指定します。

LCD_ANK_LIGHT	:縮小 ANK
LCD_ANK_STANDARD	:標準 ANK

type

表示属性を指定します。

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を指定します。

clm

入力文字列の表示桁位置をキャラクタ座標で指定します。
キャラクタ座標については、「デバイス制御ライブラリ リファレンスマニュアル」を参照してください。

line

入力文字列の表示行位置をキャラクタ座標で指定します。
キャラクタ座標については、「デバイス制御ライブラリ リファレンスマニュアル」を参照してください。

string

入力文字列を格納するバッファのポインタを指定します。

最大バイト数(*len*)+1 のサイズが必要です。

戻り値

発生した脱出条件、もしくは下記の値を返します。

E_NG	:異常終了
E_PRM	:パラメータエラー

説明

本関数は、指定文字数の文字列入力関数です。

本関数を実行するとキーの入力画面に切り替わり、文字列の入力を受け付けます。入力された文字は画面に表示され、脱出条件になるまで繰り返されます。入力された文字列は ***string*** に格納されます。

1.4.4. ht_NumInp

数字の入力を行います。

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

パラメータ

escape

下記の脱出条件を指定します。
OR 論理で複数の指定が可能です。

KY_F01	:F1 キー押下	KY_ENT	:実行キー押下
KY_F02	:F2 キー押下	KY_LB	:LB 発生
KY_F03	:F3 キー押下	KY_IO	:クレードル検出／USB 接続検出
KY_F04	:F4 キー押下	KY_OBR	:バーコード読み込み検出
KY_F05	:F5 キー押下	KY_FULL	:入力領域フル
KY_F06	:F6 キー押下		
KY_F07	:F7 キー押下		
KY_F08	:F8 キー押下		

fsize

表示する数字のフォントサイズを指定します。

LCD_ANK_LIGHT	:縮小 ANK
LCD_ANK_STANDARD	:標準 ANK

type

表示属性を指定します。

LCD_ATTR_NORMAL	:通常
LCD_ATTR_REVERS	:反転
LCD_ATTR_WIDTH	:強調

len

入力数字の最大バイト数を指定します。

clm

入力文字列の表示桁位置をキャラクタ座標で指定します。
キャラクタ座標については、「デバイス制御ライブラリ リファレンスマニュアル」を参照してください。

line

入力文字列の表示行位置をキャラクタ座標で指定します。
キャラクタ座標については、「デバイス制御ライブラリ リファレンスマニュアル」を参照してください。

string

入力数字を格納するバッファのポインタを指定します。

最大バイト数(*len*)+1 のサイズが必要です。

戻り値

発生した脱出条件、もしくは下記の値を返します。

E_NG	:異常終了
E_PRM	:パラメータエラー

説明

本関数は、指定文字数の電卓型数値入力関数です。

本関数を実行するとキーの入力画面に切り替わり、数字の入力を受け付けます。入力された数字は画面に表示され、脱出条件になるまで繰り返されます。入力された数字は ***string*** に格納されます。

1.4.5. ht_DateInp

日付の入力を行います。

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

パラメータ

md

下記の入力モードを指定します。

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

下記の脱出条件を指定します。

OR 論理で複数の指定が可能です。

KY_F01	:F1 キー押下	KY_ENT	:実行キー押下
KY_F02	:F2 キー押下	KY_LB	:LB 発生
KY_F03	:F3 キー押下	KY_IO	:クレードル検出／USB 接続検出
KY_F04	:F4 キー押下	KY_FULL	:入力領域フル
KY_F05	:F5 キー押下		
KY_F06	:F6 キー押下		
KY_F07	:F7 キー押下		
KY_F08	:F8 キー押下		

fsize

表示する日付のフォントサイズを指定します。

LCD_ANK_LIGHT	:縮小 ANK
LCD_ANK_STANDARD	:標準 ANK

type

表示属性を指定します。

LCD_ATTR_NORMAL	:通常
LCD_ATTR_REVERS	:反転
LCD_ATTR_WIDTH	:強調

clm

入力文字列の表示桁位置をキャラクタ座標で指定します。
キャラクタ座標については、「デバイス制御ライブラリ リファレンスマニュアル」を参照してください。

line

入力文字列の表示行位置をキャラクタ座標で指定します。
キャラクタ座標については、「デバイス制御ライブラリ リファレンスマニュアル」を参照してください。

string

入力日付を格納するバッファのポインタを指定します。
入力モード(*md*)の桁数+1 のサイズが必要です。

戻り値

発生した脱出条件、もしくは下記の値を返します。

E_NG	:異常終了
E_PRM	:パラメータエラー

説明

本関数は、日付の入力関数です。
本関数を実行するとキーの入力画面に切り替わり、日付の入力を受け付けます。入力の形式は、入力モード(*md*)に従います。入力された数字は画面に表示され、脱出条件になるまで繰り返されます。入力された日付は *string* に格納されます。

1.4.6. ht_TimeInp

時刻の入力を行います。

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

パラメータ

escape

下記の脱出条件を指定します。
OR 論理で複数の指定が可能です。

KY_F01	:F1 キー押下	KY_ENT	:実行キー押下
KY_F02	:F2 キー押下	KY_LB	:LB 発生
KY_F03	:F3 キー押下	KY_IO	:クレードル検出／USB 接続検出
KY_F04	:F4 キー押下	KY_FULL	:入力領域フル
KY_F05	:F5 キー押下		
KY_F06	:F6 キー押下		
KY_F07	:F7 キー押下		
KY_F08	:F8 キー押下		

fsize

表示する時刻のフォントサイズを指定します。

LCD_ANK_LIGHT	:縮小 ANK
LCD_ANK_STANDARD	:標準 ANK

type

表示属性を指定します。

LCD_ATTR_NORMAL	:通常
LCD_ATTR_REVERS	:反転
LCD_ATTR_WIDTH	:強調

clm

入力文字列の表示桁位置をキャラクタ座標で指定します。
キャラクタ座標については、「デバイス制御ライブラリ リファレンスマニュアル」を参照してください。

line

入力文字列の表示行位置をキャラクタ座標で指定します。
キャラクタ座標については、「デバイス制御ライブラリ リファレンスマニュアル」を参照してください。

string

入力時刻を格納するバッファのポインタを指定します。
7 バイト(時刻 6 桁+1)のサイズが必要です。

戻り値

発生した脱出条件、もしくは下記の値を返します。

E_NG	:異常終了
E_PRM	:パラメータエラー

説明

本関数は、時刻の入力関数です。

本関数を実行するとキーの入力画面に切り替わり、時刻の入力を受け付けます。入力は”HHMMSS”の形式（24 時間制）で指定してください。入力された数字は画面に表示され、脱出条件になるまで繰り返されます。入力された時刻は *string* に格納されます。

1.4.7. 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

下記のモードを指定します。

SFT_READ	:シフト状態読み出し
SFT_SET_OFF	:シフト OFF の状態に設定
SFT_SET_ON	:シフト ON の状態に設定

戻り値

読み出しもしくは設定を行った結果を返します。

E_SFT_ON	:シフト ON 状態
E_SFT_OFF	:シフト OFF 状態
E_PRM	:パラメータエラー

1.5. 通信関数群

1.5.1. 関数一覧

FLINK プロトコルを使用したファイル転送、および Ir 簡易制御に関する機能を提供します。

関数名	説明
ht_FLNKsend	FLINK 送信
ht_FLNKrecv	FLINK 受信
Ir_c_xxx	Ir 簡易制御関数群 デバイス制御ライブラリのシリアル通信関数(c_xxx)と同等のインターフェースで、Ir の制御を行うことが可能です。

1.5.2. 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             *s_file,
  H             *phase
)
```

パラメータ

com_no

下記の通信ポートを指定してください。

COM0　　:IR インターフェース
COM9　　:USB インターフェース

irSpeed

com_no に COM0 を指定した場合は、赤外通信最高速度を指定します。
COM0 以外を指定した場合は、0 を指定してください。

CU_B9600　　:　9600 bps
CU_B19K　　: 19200 bps
CU_B38K　　: 38400 bps
CU_B57K　　: 57600 bps
CU_B115K　　:115200 bps

rsPrm

使用しません。

mode

局モードを指定します。

CU_MODE_HT　　:HT モード

sendmode

転送モードを指定します。cu_fileSend 関数の *mode* パラメータに該当します。

CU_TRANS_NORMAL　　:通常転送
　　指定されたファイルのみを転送します。
CU_TRANS_RECURSIVE　:再帰呼び出し転送
　　送信ファイル名(*fnam_area*)で指定されたディレクトリ傘下の指定ファイルすべてが転送対象となります。

fnam_area

送信ファイル名の配列を指定します。配列の最後は NULL をセットしてください。

例）“A:¥SEND¥AAA.DAT”

ファイル名にはワイルドカード(*,?)を指定することが可能です。

dir

送信先ディレクトリ名を指定します。

送信先に指定のディレクトリが存在しない場合は、自動的に作成します。

例）“B:¥RECV¥”

protect

強制上書きを行うかどうかを指定します。

強制上書き(CU_PROTECT_INVALID)を指定すると、受信側に同一ファイルが書込み禁止モードで存在した場合、属性変更して上書きを行います。強制上書きしない(CU_PROTECT_VALID)を指定した場合はエラーとなります。

CU_PROTECT_VALID	:強制上書きしない
CU_PROTECT_INVALID	:強制上書きする

s_file

送信ファイル名の配列(*fnam_area*)に対し、送信の完了した配列の数を格納するポインタを指定します。

phase

異常終了による途中中断時のフェーズを格納するバッファへのポインタを指定します。

0	:送信前(オープン処理)
1	:送信中
2	:送信後(クローズ処理)

戻り値

下記の値を返します。

E_OK	:正常終了
E_NG	:通信エラー 詳細はデバイス制御ライブラリの cu_readErrStat 関数を確認してください。
E_PRM	:パラメータエラー

説明

FLINK プロトコルを使用し、従局(DT-970)から主局(ホスト PC)へのファイル転送を行います。

本関数は、デバイス制御ライブラリの cu_open、cu_fileSend、cu_close までを一括して行う関数です。

fnam_area で指定した配列の数だけ cu_fileSend を実行します。

各パラメータの詳細は、「デバイス制御ライブラリ リファレンスマニュアル」を参照してください。

本関数を実行する場合、LMWIN はサーバモードの状態にしてください。

1.5.3. ht_FLNKrecv

FLINK プロトコルによる受信を行います。

```
ER  ht_FLNKrecv (
  H             com_no,
  H             irSpeed,
  CU_RSPRM      *rsPrm,
  H             mode,
  H             recvmode,
  B             *fnam_area[],
  B             *dir,
  H             protect,
  H             *r_file,
  H             *phase
)
```

パラメータ

com_no

下記の通信ポートを指定してください。

COM0　　:IR インターフェース
COM9　　:USB インターフェース

irSpeed

com_no に COM0 を指定した場合は、赤外通信最高速度を指定します。
COM0 以外を指定した場合は、0 を指定してください。

CU_B9600　　:　9600 bps
CU_B19K　　: 19200 bps
CU_B38K　　: 38400 bps
CU_B57K　　: 57600 bps
CU_B115K　　:115200 bps

rsPrm

使用しません。

mode

局モードを指定します。

CU_MODE_HT　　:HT モード

recvmode

下記の転送モードのいずれかを指定します。cu_fileRecv 関数の *mode* パラメータに該当します。

CU_TRANS_NORMAL　　:通常転送
　　指定されたファイルのみを転送します。
CU_TRANS_RECURSIVE　:再帰呼び出し転送
　　受信ファイル名(*fnam_area*)で指定されたディレクトリ傘下の指定ファイルすべてが転送対象となります。

fnam_area

受信ファイル名の配列を指定します。配列の最後は NULL を設定してください。

例）“A:¥SEND¥AAA.DAT”

ファイル名にはワイルドカード(*,?)を指定することが可能です。

dir

受信先ディレクトリ名を指定します。

受信先に指定のディレクトリが存在しない場合は、自動的に作成します。

例）“B:¥RECV¥”

protect

強制書き込みを行うかどうかを指定します。

強制上書き(CU_PROTECT_INVALID)を指定すると、受信先に同一ファイルが書込み禁止モードで存在した場合、属性変更して上書きを行います。強制上書きしない(CU_PROTECT_VALID)を指定した場合はエラーとなります。

CU_PROTECT_VALID	:強制書き込みしない
CU_PROTECT_INVALID	:強制書き込みする

r_file

受信ファイル名の配列(*fnam_area*)に対し、受信の完了した配列の数を格納するポインタを指定します。

phase

異常終了による途中中断時のフェーズを格納するバッファへのポインタを指定します。

0	:受信前(オープン処理)
1	:受信中
2	:受信後(クローズ処理)

戻り値

下記の値を返します。

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

説明

FLINK プロトコルを使用し、主局(ホスト PC)から従局(DT-970)へのファイル転送を行います。

本関数は、デバイス制御ライブラリの cu_open、cu_fileRecv、cu_close までを一括して行う関数です。

fnam_area で指定した配列の数だけ cu_fileRecv を実行します。

各パラメータの詳細は、「デバイス制御ライブラリ リファレンスマニュアル」を参照してください。

本関数を実行する場合、LMWIN はサーバモードの状態にしてください。

1.5.4. 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

下記の通信ポートを指定してください。

COM0　　:IR インターフェース

param

下記の通信形式パラメータを指定します。

各パラメータの論理和で指定してください。

ボーレート	B_1200	: 1200 bps
	B_2400	: 2400 bps
	B_4800	: 4800 bps
	B_9600	: 9600 bps
	B_19200	: 19200 bps
	B_38400	: 38400 bps
	B_57600	: 57600 bps
	B_115200	:115200 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
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

buff, buf_l, tim_out, del_cod, busy_ch, nonbusy_ch

使用しません。

戻り値

下記の値を返します。

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

説明

Ir 回線をオープンします。

デフォルトの Ir の設定は

局	: 2 次局
Wire	: 3WIRE-RAW
DR/CS/CD 信号待ち時間	: タイマ指定なし(信号を待ち続ける)
データ待ち時間	: タイマ指定なし(データ読み込み／書き込みを待ち続ける)

としています。

必要に応じて変更を行ってください。

1.5.5. 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

下記の通信ポートを指定してください。

COM0　　:IR インターフェース

戻り値

下記の値を返します。

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

1.5.6. Ir_c_status

Ir 回線のステータスを取得します。

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

パラメータ

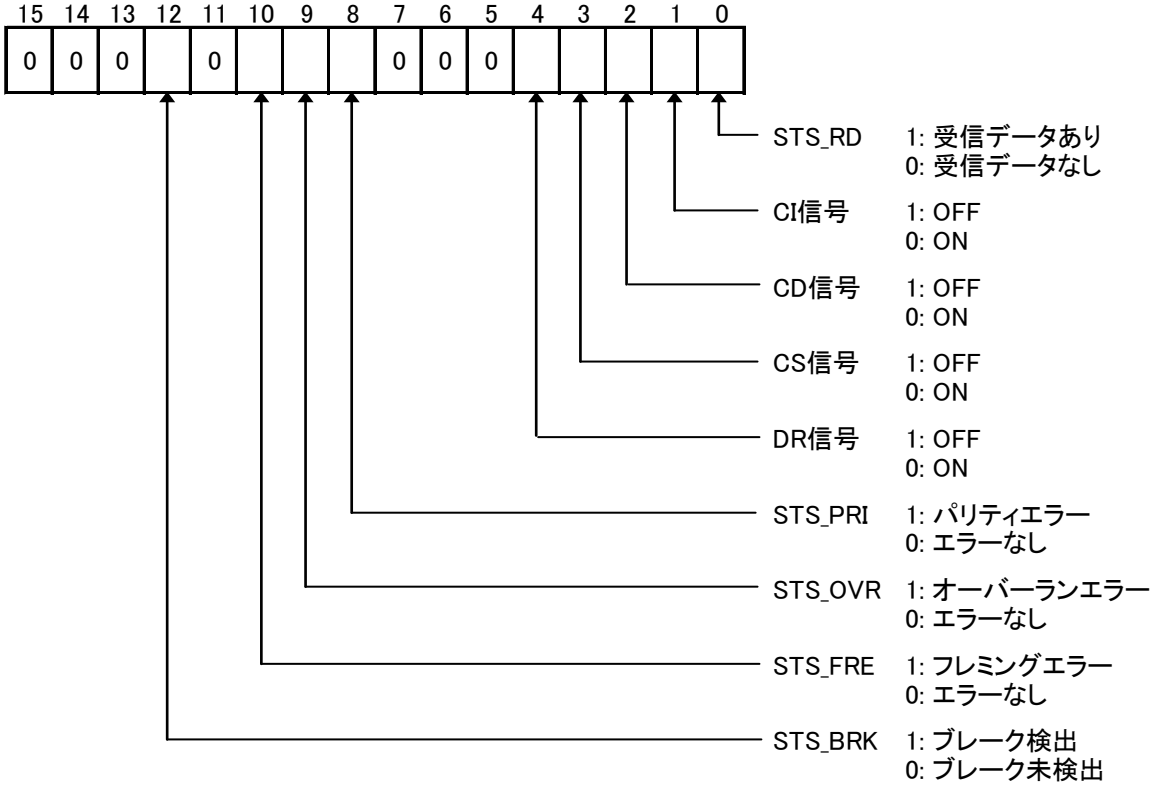
com_no

下記の通信ポートを指定してください。

COM0　　:IR インターフェース

戻り値

正常終了した場合は、下図に示すステータス値を返します。



エラーの場合は、下記の値を返します。

E_NG　　:回線エラー

E_PRM　　:パラメータエラー

1.5.7. Ir_c_hold

本関数は、シリアル通信関数(c_xxx)と同等のインターフェースを提供するためのダミー関数です。
通信ポートの占有または解除は行いません。パラメータのチェックのみを行います。

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

パラメータ

com_no

下記の通信ポートを指定してください。

COM0	:IR インターフェース

mode

下記の占有設定を指定してください。

HOLD_ON	:占有する
HOLD_OFF	:占有を解除する

戻り値

下記の値を返します。

E_OK	:正常終了
E_PRM	:パラメータエラー

1.5.8. 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

ありません。

戻り値

下記の値を返します。

1	:回線オープン中
0	:回線未オープン

1.5.9. 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分送信します。

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

パラメータ

com_no

下記の通信ポートを指定してください。

COM0　　:IR インターフェース

buffer

送信バッファのアドレスを指定します。

length

送信文字数(バイト数)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

説明

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。

1.5.10. Ir_c_din

受信バッファに格納されたデータを 1 文字読み出します。

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

パラメータ

com_no

下記の通信ポートを指定してください。

COM0　　:IR インターフェース

buffer

受信バッファのアドレスを指定します。

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_PRM　　:パラメータエラー

説明

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

1.5.11. Ir_c_tmdin

タイムアウトを設定し、受信バッファに格納したデータを 1 文字読み出します。

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

パラメータ

com_no

下記の通信ポートを指定してください。

COM0　　:IR インターフェース

buffer

受信バッファのアドレスを指定します。

rcv_time

受信タイムアウト監視値を指定します。

0～32767(×7.8ms)の範囲で指定してください。

戻り値

下記の値を返します。

E_OK　　:正常終了
E_NG　　:異常終了
E_PRM　　:パラメータエラー

説明

受信バッファに格納したデータを 1 文字(1 バイト)読み出します。
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。
「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。

1.5.12. Ir_c_out

Ir ポートから 1 文字(1 バイト)分送信します。

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

パラメータ

com_no

下記の通信ポートを指定してください。

COM0　　:IR インターフェース

snddata

送信データを指定します。

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

1.5.13. 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,
  B      mode
)
```

パラメータ

com_no

下記の通信ポートを指定してください。

COM0　　:IR インターフェース

mode

ブレーク信号制御方法を指定します。

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

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

1.5.14. Ir_c_txrx

本関数は、シリアル通信関数(c_xxx)と同等のインターフェースを提供するためのダミー関数です。
送受信の有効または無効設定は行いません。パラメータのチェックのみを行います。

```
ER  Ir_c_txrx (
  H      com_no,
  B      mode
)
```

パラメータ

com_no

下記の通信ポートを指定してください。

COM0　　:IR インターフェース

mode

送受信の有効、無効を指定します。

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

下記の値を返します。

E_OK　　:正常終了
E_PRM　　:パラメータエラー

1.5.15. Ir_c_iobox

本関数は、シリアル通信関数(c_xxx)と同等のインターフェースを提供するためのダミー関数です。
送信の設定または解除は行いません。パラメータのチェックのみを行います。

```
ER  Ir_c_iobox (
  H      com_no,
  B      mode
)
```

パラメータ

com_no

下記の通信ポートを指定してください。

COM0　　:IR インターフェース

mode

送信の設定、解除を指定します。

C_IOBOXENB　　:送信に設定
C_IOBOXDSB　　:送信を解除

戻り値

下記の値を返します。

E_OK　　:正常終了
E_PRM　　:パラメータエラー

1.5.16. Ir_c_irout

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

下記の通信ポートを指定してください。

COM0　　:IR インターフェース

buffer

送信バッファのアドレスを指定します。

length

送信文字数(バイト数)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

1.5.17. Ir_c_timer

本関数は、シリアル通信関数(c_xxx)と同等のインターフェースを提供するためのダミー関数です。
CS、DR、CD 信号のタイムアウト値の設定は行いません。パラメータのチェックのみを行います。

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

パラメータ

com_no

下記の通信ポートを指定してください。

COM0　　:IR インターフェース

cs_time

CS 信号のタイムアウト値を指定します。
0～32767(×7.8ms)の範囲で指定してください。

dr_time

DR 信号のタイムアウト値を指定します。
0～32767(×7.8ms)の範囲で指定してください。

cd_time

CD 信号のタイムアウト値を指定します。
0～32767(×7.8ms)の範囲で指定してください。

戻り値

下記の値を返します。

E_OK　　:正常終了
E_NG　　:異常終了
E_PRM　　:パラメータエラー

1.5.18. 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

下記の通信ポートを指定してください。

COM0　　:IR インターフェース

mode

RS 信号の ON／OFF を指定します。

RS_ON　　:RS 信号 ON

RS_OFF　　:RS 信号 OFF

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

1.5.19. 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

下記の通信ポートを指定してください。

COM0　　:IR インターフェース

mode

ER 信号の ON／OFF を指定します。

ER_ON　　:ER 信号 ON

ER_OFF　　:ER 信号 OFF

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

1.5.20. 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

下記の通信ポートを指定してください。

COM0　　:IR インターフェース

mode

ER 信号と RS 信号の ON／OFF を指定します。

ERRS_ON　　:ER／RS 信号 ON

ERRS_OFF　　:ER／RS 信号 OFF

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

1.5.21. 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

下記の通信ポートを指定してください。

COM0　　:IR インターフェース

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

1.5.22. Ir_c_bfsts

Ir ポートのバッファの文字数をチェックします。

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

パラメータ

com_no

下記の通信ポートを指定してください。

COM0　　:IR インターフェース

bfsts

下記構造体のバッファを指定してください。

```
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

下記の値を返します。

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

説明

Ir ポートのバッファの文字数をチェックします。
構造体のメンバー*char_no* に受信文字数が格納されます。それ以外のメンバーは 0 になります。

1.5.23. Ir_c_errbfring

本関数は、シリアル通信関数(c_xxx)と同等のインターフェースを提供するためのダミー関数です。
エラーコードバッファリングの設定は行いません。パラメータのチェックのみを行います。

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

パラメータ

com_no

下記の通信ポートを指定してください。

COM0	:IR インターフェース

mode

エラーコードバッファリングの設定を指定してください。

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

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

1.5.24. Ir_c_rderrsts

エラーステータスを取得します。

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

パラメータ

com_no

下記の通信ポートを指定してください。

COM0　　:IR インターフェース

com_status

エラーステータスを受け取る変数を指定します。

戻り値

下記の値を返します。

E_OK　　:正常終了

E_PRM　　:パラメータエラー

説明

エラーステータスを取得します。取得後はエラーステータスをクリアします。

各関数の戻り値が異常終了(E_NG)の場合、本関数で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。

エラーステータスの詳細は、「デバイス制御ライブラリ リファレンスマニュアル」の Ir_Err_Get 関数を参照してください。

1.5.25. Ir_c_chghdr

本関数は、シリアル通信関数(c_xxx)と同等のインターフェースを提供するためのダミー関数です。
受信ハンドラの切り替えは行いません。パラメータのチェックのみを行います。

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

パラメータ

com_no

下記の通信ポートを指定してください。

COM0　　:IR インターフェース

mode

受信ハンドラの切り替え設定を指定します。

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

下記の値を返します。

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

1.6.1. 関数一覧

DT-970 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、読み込みおよび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。

関数名	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-970 のファイルモードを必ず「DT-700 モード」にしてください。

1.6.2. ファイル管理テーブル

ファイル制御関数で使用するファイルは、ファイル管理テーブルに事前に定義しておく必要があります。
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、flietbl という名称で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ようにしてください。

```
struct   ht_filetbl {
    char    fname[14];     /* ファイル名称                                    */
    H       rsize;         /* 1 レコードのバイト数                            */
    FILE    *fp;           /* ストリーム保存（ファイル制御関数内で使用）      */
};
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（ファイルクローズ状態）で初期設定しておいてください。
※ 最終行は、必ず上記のようにファイル名なし、レコードサイズ 0 を付加してください。

1.6.3. ht_fileopen

ファイルをオープンします。

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

パラメータ

fno

ファイル管理テーブルに定義されたファイルの番号(1～)を指定します。

戻り値

現在のレコード数を返します。

エラーの場合は、下記の値が返ります。

-1	:C 標準入出力関数でエラーが発生しました。
-2	:ファイル番号の指定が不適切です。
-3	:ファイル管理テーブル(1 レコードのバイト数)の値が異常です。
	:既にファイルがオープンされています。

説明

ファイル管理テーブルで定義されたファイルをオープンします。

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_fileclose

ファイルをクローズします。

```
void  ht_fileclose (
  B      fno
)
```

パラメータ

fno

クローズするファイル番号を指定します。

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

ありません。

説明

ht_fileopen 関数でオープンされた、指定番号のファイルをクローズします。

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_fileread

ファイルから 1 レコードを読み込みます。

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

パラメータ

fno

対象のファイル番号を指定します。

rno

読み込むレコードの序数(1～)を指定します。

buffer

読み込んだレコードデータを格納するバッファのポインタを指定します。

戻り値

読み込んだレコードの序数を返します。
エラーの場合は、下記の値が返ります。

0	:指定のファイル番号、およびレコード序数の値が不適切です。
	:ファイル管理テーブル(1 レコードのバイト数)の値が異常です。
-1	:C 標準入出力関数でエラーが発生しました。

説明

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。
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_filewrite

ファイルに 1 レコードを書き込みます。

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

パラメータ

fno

対象のファイル番号を指定します。

rno

書き込むレコードの序数(1～現在のレコード数+1)を指定します。

buffer

書き込むレコードデータのポインタを指定します。

戻り値

書き込んだレコードの序数を返します。
エラーの場合は、下記の値が返ります。

0	:指定のファイル番号、およびレコード序数の値が不適切です。
	:ファイル管理テーブル(1 レコードのバイト数)の値が異常です。
-1	:C 標準入出力関数でエラーが発生しました。
-2	:レコード序数の指定が、既存レコード数+1 を越えています。
-3	:ファイル格納領域の空き容量が不足しています。

説明

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。
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。

1.6.7. ht_filesize

ファイルのサイズを取得します。

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

パラメータ

fno

対象のファイル番号(1～)を指定します。

戻り値

ファイルのサイズを返します。

エラーの場合は、下記の値が返ります。

0　　:指定番号のファイルは存在していません。

-1　　:ファイル番号の指定が不適切です。

説明

ファイル管理テーブルで定義されたファイルのサイズを取得します。

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。

1.6.8. ht_filelof

ファイルの登録レコード数を取得します。

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

パラメータ

fno

対象のファイル番号(1～)を指定します。

戻り値

ファイルのレコード数を返します。

エラーの場合は、下記の値が返ります。

0	:指定番号のファイルは存在していません。
-1	:ファイル番号の指定が不適切です。
	:ファイル管理テーブル(1 レコードのバイト数)の値が異常です。

説明

ファイル管理テーブルで定義されたファイルの登録レコード数を取得します。

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

1.7.1. 関数一覧

プログラム開発に有用な関数群を提供します。

関数名	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
Ir_xy_modem	IrDA 用 XYMODEM

1.7.2. ht_waitmsec

指定した時間を待ちます。

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

パラメータ

count

待ち時間を、250 ミリ秒単位で指定します。
250 ミリ秒～1 時間までの範囲で指定してください。
例）count = 40;　　/* 待ち時間 = 40 x 0.25 秒 = 10 秒 */

戻り値

下記の値を返します。

E_OK	:正常終了
E_PRM	:パラメータエラー

1.7.3. ht_dbgsendmsg

指定の文字列を赤外線ポートより送信します。

```
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を指定します。

len

送信データのバイト数を指定します。

戻り値

ありません。

説明

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。
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

1.7.4. ht_beep

エラー音を鳴らします。

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

パラメータ

type

エラー音のタイプを指定します。

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

戻り値

ありません。

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

通信モードを指定します。

下記の設定内容を論理和演算子で指定します。
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
詳細は「補足」を参照してください。

tel

電話番号文字列を指定します。省略時は NULL 文字列を指定してください。
使用可能文字は下記の通りです。

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 コマンド文字列を指定します。省略時は NULL 文字列を指定してください。

file_list[]

ファイル名のリストを指定します。
詳細は「補足」を参照してください。

path

受信の場合に、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ます。
送信の場合は、必ず NULL を指定してください。

戻り値

下記の値を返します。

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)
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、本関数内で通知フラグを一度クリアします。

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
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、本関数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。

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
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。

50 :内部エラー

51 :タイムアウト

52 :外部機器エラー

補足

＜通信モード＞

	パラメータ	説明
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)
'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	メッセージを表示しない
	XY_MON	メッセージを表示する
ポート操作	XY_232ON	RS-232C を操作する
	XY_232OFF	RS-232C を操作しない

※ はデフォルト値を示しています。

通信モードを指定する時は、下記の点に注意してください。

・ 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（機能、モデム操作、ポート操作）の指定における処理内容を下表に示します。

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

※ *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＞という形式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
```

2. Bluetooth プリンタライブラリ

2.1. 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、デバイス制御ライブラリの Bluetooth 関数を使用しています。
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確認ください。

2.1.1. 接続対象プリンタ

接続可能なプリンタを下表に示します。

プリンタ		DT-970	プリンタの特性	
			Inquiry 受付時間	その他
SATO プチラパン	PT200e	○		
	PT208e	◎		
東芝テック	B-SP2D	○	電源 ON 直後から 1 分間	CODE128 のバーコードとして BD アドレスの印字が可能
	B-EP2DL	◎		

◎:認定デバイス ／ ○:動作確認済みデバイス

2.1.2. 提供ファイル

下記のファイルを提供します。

ファイル名	内容
PRN_BT.H	Bluetooth プリンタライブラリのヘッダファイル
PRN_BT.OBJ	Bluetooth プリンタライブラリのオブジェクトファイル
PRN_BT.LIB	Bluetooth プリンタライブラリのライブラリファイル

※ オブジェクトファイルまたはライブラリファイルのどちらか一方をリンクさせてください。

2.2. 機能

2.2.1. 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 BD 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。

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。

① Bluetooth の使用を開始 (BT_Start)
② Bluetooth 機器の探索 (BT_Inquiry)
③ Bluetooth 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)
⑥ Bluetooth の使用を終了 (BT_Stop)

※ 但し、B-SP2D、B-EP2DL 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。

固定の BD アドレスで設定

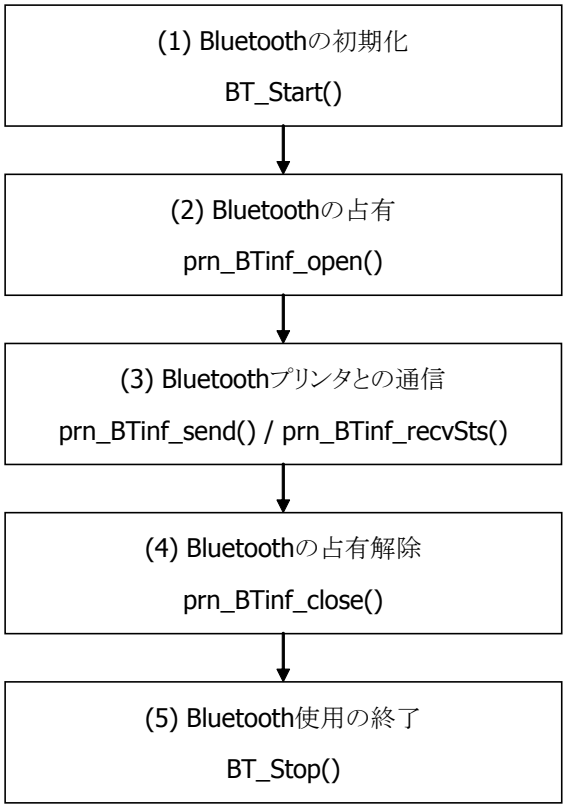
次の手順で処理を行ってください。

① テスト印字を行なう。(方法は各プリンタの説明書を参照してください)
② Bluetooth の使用を開始 (BT_Start)
③ テスト印字内のアドレスを入力
④ ③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)
⑥ Bluetooth の使用を終了 (BT_Stop)

※ B-SP2D、B-EP2DL は、BD 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み込みに変えることも可能です。

2.2.2. 関数の実行手順

通信の接続準備ができましたら、下記の手順で Bluetooth プリンタとの通信を実行することができます。



2.2.3. 通信のオープン・クローズ

オープン（prn_BTinf_open）

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、分けて実行でき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)
③ Bluetooth 機器との接続 (BT_Open 最短 3 秒)

クローズ（prn_BTinf_close）

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。

① Bluetooth 機器との切断 (BT_Close)

2.2.4. 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（prn_BTinf_send）

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（prn_BTinf_recvSts）

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

2.2.5. エラーステータス

デバイス制御ライブラリの BT_Err_Get 関数を使用すると、エラーステータスを取得することができます。
以下にエラーステータスと対処方法の一覧を示します。

No	エラーステータス	対処方法
1	BTERR_DISCONNECT	切断処理後、接続し、リトライしてください。
2	BTERR_PARAMETER	関数のパラメータが適切かどうか、プログラムを確認してください。
3	BTERR_BREAK_EVNT	切断処理をしてください。
4	BTERR_LB0	次回電源 ON で切断処理後、再度接続し、リトライしてください。
5	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、リトライしてください。
6	BTERR_LB2	
7	BTERR_LB4	操作中には発生しません。
8	BTERR_LB5	次回電源 ON で切断処理後、再度接続し、リトライしてください。
9	BTERR_NOTSTART	prn_BTinf_open を呼んでいるかどうか、プログラムを確認してください。
10	BTERR_TIMEOUT	電源など、プリンタの状態を確認してからリトライしてください。
11	BTERR_PARITY	切断処理後、再度接続し、リトライしてください。 本来、本エラーが発生することはほとんどありません。
12	BTERR_OVERRUN	
13	BTERR_FRAMING	
14	BTERR_TRANSFER	
15	BTERR_FILEOPEN	プログラムの内容を見直してください。
16	BTERR_FILEACCESS	
17	BTERR_NAK00	リトライしてください。 本来、本エラーが発生することはほとんどありません。
18	BTERR_NAK01	
19	BTERR_NAK02	BT_SetPassKey を呼んでいるかどうか、プログラムを確認してください。
20	BTERR_NAK03	電源など、プリンタの状態を確認してからリトライしてください。
21	BTERR_NAK04	
22	BTERR_NAK05	PIN コードの設定を確認してください。
23	BTERR_NAK06	Bluetooth の機能が故障した可能性があります。
24	BTERR_NAK08	BT_Stop を呼んでください。 本来、本エラーが発生することはほとんどありません。
25	BTERR_NAK09	
26	BTERR_NAK10	Bluetooth の機能が故障した可能性があります。
27	BTERR_NAK11	デバイス名の長さが適切かどうか、プログラムを確認してください。
28	BTERR_NAK12	BT_Stop を呼んでください。 本来、本エラーが発生することはほとんどありません。
29	BTERR_NAK13	Bluetooth の機能が故障した可能性があります。
30	BTERR_NAK14	リトライしてください。 本来、本エラーが発生することはほとんどありません。
31	BTERR_NAKMINUS1	Bluetooth の機能が故障した可能性があります。

2.3. 関数仕様

2.3.1. 関数一覧

本ライブラリは、下記の関数を提供します。

関数名	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

2.3.2. prn_BTinf_open

Bluetooth を占有します。

```
ER  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。

len

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_PRM	:パラメータエラー
E_PRN_PON	:オープン済み
E_NG	:その他のエラー

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。

2.3.3. prn_BTinf_close

Bluetooth の占有を解除します。

```
ER  prn_BTinf_close ()
```

パラメータ

ありません。

戻り値

下記の値を返します。

E_OK	:正常終了
E_PRN_POFF	:未オープン
E_NG	:その他のエラー

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。

2.3.4. prn_BTinf_send

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。

```
ER  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_PRN_POFF	:未オープン
E_NG	:その他のエラー

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。

2.3.5. 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します。

```
ER  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_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
E_NG	:その他のエラー

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。

3. TEC IrDA プリンタライブラリ

本ライブラリは、東芝テック製のポータブルプリンタ（B-SP2D）を赤外線通信（TEC プロトコル使用）で操作するために、DT-930 で提供していたライブラリです。

DT-970 では、本ライブラリは未サポートとなります。
赤外線通信によるプリンタ操作を行う場合は、次章に記載のモバイルプリンタ制御ライブラリをご使用ください。

4. モバイルプリンタ制御ライブラリ

4.1. 概要

IrDA／Bluetooth 経由で携帯型プリンタ（以下、プリンタと記述します）を制御するためのライブラリです。
IrDA と Bluetooth の通信仕様の違いに対し、統一されたインターフェースを提供します。

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確認ください。

4.1.1. 接続対象プリンタ

接続可能なプリンタを下表に示します。

プリンタ		IrDA	Bluetooth
SATO プチラパン	PT200e	○	○
	PT208e		◎
東芝テック	B-SP2D	○	○
	B-EP2DL		◎

◎:認定デバイス ／ ○:動作確認済みデバイス

4.1.2. 提供ファイル

下記のファイルを提供します。

ファイル名	内容
MPRLIB.H	モバイルプリンタ制御ライブラリのヘッダファイル
MPRLIB.LIB	モバイルプリンタ制御ライブラリのライブラリファイル

4.2. 通信機能

4.2.1. IrDA

本ライブラリにおける IrDA 通信仕様を下記に示します。

種別	内容	備考
通信プロトコル	IrCOMM	通信前にプリンタの IrDA の設定を IrCOMM に設定してください。
通信速度	ネゴシエーションにて自動決定	
フロー制御	なし	
デバイス名	無効	デバイス名は無視して通信します。
受信タイムアウト	設定関数にて設定可能	
IrDA オープンタイムアウト	設定関数にて設定可能	

4.2.2. Bluetooth

本ライブラリにおける Bluetooth 通信仕様を下記に示します。

種別		内容	備考
通信プロファイル		シリアルポートプロファイル	
BD アドレス		設定関数にて設定可能	プリンタの BD アドレス
受信タイムアウト		設定関数にて設定可能	
問い合わせ時間		設定関数にて設定可能	
セキュリティ (※)			PT200e、PT208e のみ設定可
	パスキー (PIN コード)	設定関数にて設定可能	
	認証設定		
	暗号化設定		

※ 本体の BT のセキュリティを設定する場合、通信対象のプリンタで BT のセキュリティ設定が可能かどうかを事前に確認して下さい。

※ 本体の BT のセキュリティを設定にした場合、Bluetooth でプリンタと通信する前にプリンタの BT のセキュリティ設定を本体の設定に合わせて下さい。

4.2.3. プリンタとの通信制御

通信処理

本ライブラリは、アプリケーションから指定されたデータの送信、または受信を行いますので、アプリケーションは下記のように制御して下さい。

■ 送信処理
アプリケーションで送信パケット（送信データ）作成後、本ライブラリ関数を使用してデータを送信して下さい。

※ 本ライブラリは、送信データを各 I/F のプロトコルスタックに対して正常に渡した段階で終了します。したがって、プリンタのステータス受信等を行って、正しく送信できたか否かを確認する必要があります。

■ 受信処理
本ライブラリ関数に対して受信データ数を指定して、データを受信して下さい。

※ 送受信時のデータパケットは、プリンタ種／インターフェース／コマンドに依存しますので、各プリンタのパケット仕様を確認して下さい。

通信シーケンス

通信シーケンスは、各プリンタの通信シーケンスの仕様を確認して下さい。

Suspend/Resume への対応

通信中（オープン中）に本体の電源 OFF/ON が発生した場合は、復帰処理（クローズとオープン）が必要です。復帰処理が行われない間は、送信／受信共にエラーが返ります。

4.3. 関数仕様

4.3.1. 関数一覧

本ライブラリは、下記の関数を提供します。

関数名	説明
MPRInit	リソースの確保、及び通信回線を初期化します。
MPRDeinit	リソースの解放、及び通信回線の終了処理を行います。
MPROpen	通信回線をオープンします。
MPRClose	通信回線をクローズします。
MPRSend	指定されたデータを送信します。
MPRReceive	指定された長さのデータを受信します。
MPRCalcCRC16	指定されたデータの CRC(CRC-16)を算出します。
MPRSetIrDA	IrDA の通信パラメータを設定します。
MPRGetIrDA	IrDA の通信パラメータを取得します。
MPRSetBluetooth	Bluetooth の通信パラメータを設定します。
MPRGetBluetooth	Bluetooth の通信パラメータを取得します。

4.3.2. MPRInit

リソースの確保、及び指定された通信ポートを電源投入／初期化します。

```
int  MPRInit (
  UW     Port,
  UW     Printer
)
```

パラメータ

Port

通信ポートを指定します。

MPR_PORT_IR	:IrDA (IrCOMM)
MPR_PORT_BT	:Bluetooth（シリアルプロファイル）

Printer

プリンタの種別を指定します。

MPR_PRN_BSP2D	:TEC 製 B-SP2D シリーズ
MPR_PRN_PT200E	:SATO 製 PT200e シリーズ
MPR_PRN_BEP2DL	:TEC 製 B-EP2DL シリーズ
MPR_PRN_PT208E	:SATO 製 PT208e シリーズ

戻り値

下記の値を返します。

MPR_SUCCESS	:正常終了
MPR_INVALID_PARAM	:パラメータエラー
MPR_ERR_FAILED	:初期化済み、または初期化失敗

説明

指定した通信ポートに対し、サポート外のプリンタを指定した場合は、パラメータエラー(MPR_INVALID_PARAM)となります。

通信ポートとプリンタの組み合わせについては、「4.1.1 接続対象プリンタ」を参照してください。

4.3.3. MPRDeinit

リソースの解放、及び引数に応じて使用中の通信ポートの電源断／終了処理を行います。

```
int  MPRDeinit (
  int    Off
)
```

パラメータ

Off

使用中の通信ポートの電源断の可否を指定します。

0 以外　　:通信ポートの電源を切ります。

0　　:通信ポートの電源を切りません。

戻り値

下記の値を返します。

MPR_SUCCESS　　:正常終了

4.3.4. MPROpen

初期化された通信ポートをオープンし、プリンタとのセッションを確立します。

```
int  MPROpen ()
```

パラメータ

ありません。

戻り値

下記の値を返します。

MPR_SUCCESS	:正常終了
MPR_ERR_NOTINITIALIZED	:通信ポートの未初期化
MPR_ERR_FAILED	:二重オープン、または通信ポートのオープン／セッション確立失敗

4.3.5. MPRClose

現在使用中の通信ポートをクローズします。

```
int  MPRClose ()
```

パラメータ

ありません。

戻り値

下記の値を返します。

MPR_SUCCESS	:正常終了
MPR_ERR_NOTOPENED	:通信ポート未オープン

4.3.6. MPRSend

指定されたデータを送信します。

```
int  MPRSend (
  UB     *pBuffer,
  UW     Length,
  UW     *pResult
);
```

パラメータ

pBuffer

送信するデータが入ったバッファへのポインタを指定します。

Length

送信するデータサイズをバイト単位で指定します。

pResult

本関数で送信したデータサイズを格納する領域へのポインタを指定します。

戻り値

下記の値を返します。

MPR_SUCCESS	:正常終了
MPR_ERR_NOTOPENED	:通信ポート未オープン
MPR_ERR_SUSPENDED	:サスペンド発生
MPR_ERR_FAILED	:送信異常終了

4.3.7. MPRReceive

指定されたデータ数のデータを受信します。

```
int  MPRReceive (
  UB     *pBuffer,
  UW     Length,
  UW     *pResult
)
```

パラメータ

pBuffer

バッファへのポインタを指定します。このバッファに、受信したデータが格納されます。

Length

受信するデータサイズをバイト単位で指定します。

pResult

本関数で受信したデータサイズを格納する領域へのポインタを指定します。

戻り値

下記の値を返します。

MPR_SUCCESS	:正常終了
MPR_ERR_NOTOPENED	:通信ポート未オープン
MPR_ERR_SUSPENDED	:サスペンド発生
MPR_ERR_TIMEOUT	:受信タイムアウト
MPR_ERR_FAILED	:送信異常終了

4.3.8. MPRCalcCRC16

指定されたデータの CRC(CRC-16)を算出します。

```
UH  MPRCalcCRC16 (
  UH     crc,
  UB     *pBuffer,
  UW     Length
)
```

パラメータ

crc

CRC の初期値を指定します。

pBuffer

CRC 算出対象のデータバッファへのポインタを指定します。

Length

CRC 算出対照のデータサイズをバイト単位で指定します。

戻り値

算出した CRC 値が返ります。

説明

本関数は CRC 算出対象のデータが分割していても、前回までの CRC を初期値として指定することで、全てのデータを対象とした CRC を算出することが可能です。

例）buff1、buff2、buff3 の三つのデータを対象とした CRC を算出します。

```
crc = MPRCalcCRC16(0, buff1, 2);      // buff1 を対象とした CRC を算出（CRC の初期値は 0）
crc = MPRCalcCRC16(crc, buff2, 3);    // buff1,buff2 を対象とした CRC を算出
crc = MPRCalcCRC16(crc, buff3, 3);    // buff1,buff2,buff3 を対象とした CRC を算出
```

4.3.9. MPRSetIrDA

IrDA の通信パラメータを設定します。

```
int  MPRSetIrDA (
  ST_IRPARAMS    *pstIrParams
)
```

パラメータ

pstIrParams

ST_IRPARAMS 構造体へのポインタを指定します。

```
typedef struct _ST_IRPARAMS {
    UW      OpenTimeout;
    UW      Timeout;
    UW      Reserved1;
    UW      Reserved2;
} ST_IRPARAMS;
```

OpenTimeout	:IrDA オープンタイムアウト時間を秒単位で指定します。 最小値 : 0 秒 最大値 :60 秒 デフォルト :10 秒
Timeout	:IrDA 受信タイムアウト時間を秒単位で指定します。 最小値 : 0 秒 最大値 :30 秒 デフォルト :10 秒
Reserved1	:予約。
Reserved2	:予約。

戻り値

下記の値を返します。

MPR_SUCCESS	:正常終了
MPR_INVALID_PARAM	:パラメータエラー
MPR_ERR_FAILED	:送信異常終了

説明

本関数は、通信ポート初期化前（MPRInit 実行前）に使用して下さい。

4.3.10. MPRGetIrDA

IrDA の通信パラメータを取得します。

```
int  MPRGetIrDA (
  ST_IRPARAMS    *pstIrParams
)
```

パラメータ

pstIrParams

ST_IRPARAMS 構造体へのポインタを指定します。

```
typedef struct _ST_IRPARAMS {
    UW      OpenTimeout;
    UW      Timeout;
    UW      Reserved1;
    UW      Reserved2;
} ST_IRPARAMS;
```

OpenTimeout　　:IrDA オープンタイムアウト時間（秒単位）を取得します。
Timeout　　:IrDA 受信タイムアウト時間（秒単位）を取得します。
Reserved1　　:予約。
Reserved2　　:予約。

戻り値

下記の値を返します。

MPR_SUCCESS　　:正常終了

4.3.11. MPRSetBluetooth

Bluetooth の通信パラメータを設定します。

```
int  MPRSetBluetooth (
  ST_BTPARAMS    *pstBTParams
)
```

パラメータ

pstBTParams

ST_BTPARAMS 構造体へのポインタを指定します。

```
typedef struct _ST_BTPARAMS {
    UW      InqTimeout;
    UW      Timeout;
    B       Address[18];
    int     EnablePassKey;
    B       PassKey[17];
    int     Authentication;
    int     Encryption;
    UW      Reserved1;
    UW      Reserved2;
} ST_BTPARAMS;
```

InqTimeout　:Bluetooth 機器問い合わせタイムアウト時間を秒単位で指定します。
　最小値　: 0 秒
　最大値　:60 秒
　デフォルト　:10 秒

Timeout　:Bluetooth 受信タイムアウト時間を秒単位で指定します。
　最小値　: 0 秒
　最大値　:30 秒
　デフォルト　:10 秒

Address[18]　:通信対象プリンタの BD アドレスを指定します。
（コロン付き 16 進フォーマットの文字列）
　例　12:34:56:78:AB:CD
　初期値　00:00:00:00:00:00

EnablePassKey　:パスキーを設定するか、否かを指定します。
　TRUE　:パスキーを設定します。
　FALSE　:パスキーを設定しません。
　初期値　:FALSE

PassKey[17]　:設定するパスキーを ASCII 文字列で指定します。
空文字(\"")を指定すると、他の Bluetooth 機器からのパスキー要求を拒否します。
　最小値　:0 桁
　最大値　:16 桁
　初期値　:0 桁

Authentication　:Bluetooth 認証設定を指定します。
　　TRUE　　:有効
　　FALSE　　:無効
　　初期値　　:無効
Encryption　　:暗号設定を指定します。
　　TRUE　　:有効
　　FALSE　　:無効
　　初期値　　:無効
Reserved1　　:予約。
Reserved2　　:予約。

戻り値

下記の値を返します。

MPR_SUCCESS　　:正常終了
MPR_INVALID_PARAM　　:パラメータエラー
MPR_ERR_FAILED　　:初期化済み

説明

本関数は、通信ポート初期化前（MPRInit 実行前）に使用して下さい。
暗号設定は、Bluetooth 認証が有効な場合のみ有効にすることができます。Bluetooth 認証設定が無効な場合は、暗号設定を無効にして下さい。
Bluetooth 機器問い合わせタイムアウトは 0～60 秒の値で指定可能ですが、オープン時のタイムアウトは約 10 秒間隔で監視されます。

4.3.12. MPRGetBluetooth

Bluetooth の通信パラメータを取得します。

```
int  MPRGetBluetooth (
  ST_BTPARAMS    *pstBTParams
)
```

パラメータ

pstBTParams

ST_BTPARAMS 構造体へのポインタを指定します。

```
typedef struct _ST_BTPARAMS {
    UW      InqTimeout;
    UW      Timeout;
    B       Address[18];
    int     EnablePassKey;
    B       PassKey[17];
    int     Authentication;
    int     Encryption;
    UW      Reserved1;
    UW      Reserved2;
} ST_BTPARAMS;
```

InqTimeout	:Bluetooth 機器問い合わせタイムアウト時間(秒単位)を取得します。
Timeout	:Bluetooth 受信タイムアウト時間(秒単位)を取得します。
Address[18]	:通信対象プリンタの BD アドレスを取得します。 (コロン付き 16 進フォーマットの文字列)
EnablePassKey	:パスキーを設定するか、否かを取得します。
PassKey[17]	:MPRSetBluetooth で設定したパスキーを取得します。
Authentication	:Bluetooth 認証設定を取得します。
Encryption	:暗号設定を取得します。
Reserved1	:予約。
Reserved2	:予約。

戻り値

下記の値を返します。

MPR_SUCCESS	:正常終了

5. 高速ファイルサーチライブラリ

5.1. 概要

高速ファイルサーチライブラリは、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、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。

5.1.1. 提供ファイル

下記のファイルを提供します。

ファイル名	内容
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

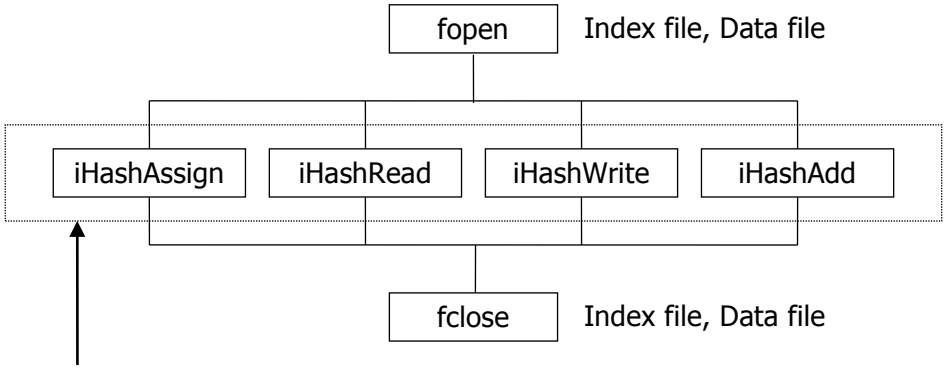
5.2. 機能

5.2.1. 概要

本ライブラリは、ユーザが作成したデータファイルに対してインデックスファイルを作成することで、高速にデータ検索を行うことができます。
本ライブラリは、高速検索を実現するために次の機能を提供します。

1) インデックスの生成 (iHashAssign)
2) データ検索 (iHashRead)
3) データ更新 (iHashWrite)
4) データ追加 (iHashAdd)

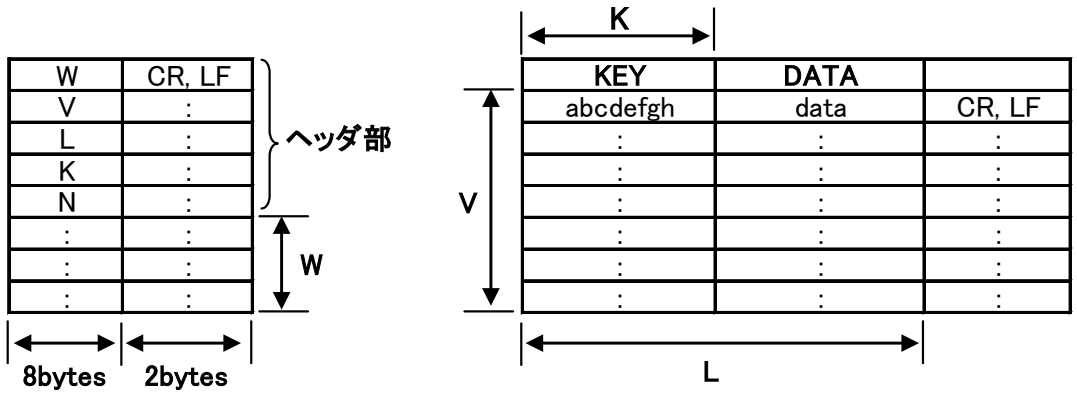
5.2.2. モジュール構成



高速ファイルサーチライブラリ

5.2.3. ファイル構造

データファイルとインデックスファイルの構造について記載します。



インデックスファイル（*.idx）　　**データファイル**

記号	説明
K	キーフィールドの長さ（Byte）
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

注意事項

1. 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。
2. 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。
3. キーは、ユニークでなければなりません。
4. 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。
5. DT-970 の 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、iHashRead 関数以外（特に iHasshAssign 関数）は、他のドライブで使用してください。

5.3. 関数仕様

5.3.1. 関数一覧

本ライブラリは、下記の関数を提供します。

関数名	説明
iHashAssign	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。
iHashRead	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。
iHashWrite	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。
iHashAdd	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。）

5.3.2. 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     K,
  long     V,
  long     L,
  long     W
)
```

パラメータ

DataFilePointer

データファイルのファイルポインタを指定します。
本関数を使用する前に、fopen 関数を使用してバイナリ読み込みモード(“rb“)でデータファイルをオープンしてください。

IndexFilePointer

インデックスファイルのファイルポインタを指定します。
本関数を使用する前に、fopen 関数を使用してバイナリ新規作成モード(“wb+“)でインデックスファイルをオープンしてください。

K, V, L, W

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。

K	:データファイルのキーフィールド長
V	: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。
L	:データファイルのレコード長(キー＋データ)
W	: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、W >= V * 1.2 上記変数は、long 型です。)

戻り値

下記の値を返します。

HASH_OPERATIONOK	:正常終了
DATA_NOFILEREAD	:データファイル リードエラー
DATA_NOFILEOPEN	:データファイル 未オープン
DATA_NOFILESEEK	:データファイル シークエラー
IDX_NOFILEREAD	:インデックスファイル リードエラー
IDX_NOFILEWRITE	:インデックスファイル ライトエラー
IDX_NOFILEOPEN	:インデックスファイル 未オープン
IDX_NOFILESEEK	:インデックスファイル シークエラー
HASH_NOMEMORY	:実行メモリ不足
PARAM_INVALID	:パラメータエラー

5.3.3. iHashRead

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。
キーデータの末尾は NULL 文字をセットしてください。
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

データファイルのファイルポインタを指定します。
本関数を使用する前に、fopen 関数を使用してバイナリ読み込みモード(“rb“)でデータファイルをオープンしてください。

IndexFilePointer

インデックスファイルのファイルポインタを指定します。
本関数を使用する前に、fopen 関数を使用してバイナリ読み込みモード(“rb“)でインデックスファイルをオープンしてください。

pszBuff

入力されたキーと一致するデータレコードを格納するバッファポインタを指定します。
データレコードの末尾は、NULL 文字がセットされます。
バッファはキーとデータレコードと NULL のサイズだけ確保してください。
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。

HASH_OPERATIONOK	:正常終了
DATA_NOFILEREAD	:データファイル リードエラー
DATA_NOFILEOPEN	:データファイル 未オープン
DATA_NOFILESEEK	:データファイル シークエラー
IDX_NOFILEREAD	:インデックスファイル リードエラー
IDX_NOFILEOPEN	:インデックスファイル 未オープン
IDX_NOFILESEEK	:インデックスファイル シークエラー
HASH_NOMEMORY	:実行メモリ不足
HASH_KEYNOTFOUND	:該当インデックスなし
HASH_INVALIDKEY	:不正キー入力 (キー長不一致)
HASH_KEYNULL	:キーポインタ不正 (NULL)
HASH_DATANULL	:データポインタ不正 (NULL)

5.3.4. 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。
キーデータの末尾は NULL 文字をセットしてください。
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

データファイルのファイルポインタを指定します。
本関数を使用する前に、fopen 関数を使用してバイナリ書き込みモード(“rb+“)でデータファイルをオープンしてください。

IndexFilePointer

インデックスファイルのファイルポインタを指定します。
本関数を使用する前に、fopen 関数を使用してバイナリ読み込みモード(“rb“)でインデックスファイルをオープンしてください。

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(L-K)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。

HASH_OPERATIONOK	正常終了
DATA_NOFILEREAD	データファイル リードエラー
DATA_NOFILEWRITE	データファイル ライトエラー
DATA_NOFILEOPEN	データファイル 未オープン
DATA_NOFILESEEK	データファイル シークエラー
IDX_NOFILEREAD	インデックスファイル リードエラー
IDX_NOFILEOPEN	インデックスファイル 未オープン
IDX_NOFILESEEK	インデックスファイル シークエラー
HASH_NOMEMORY	実行メモリ不足
HASH_KEYNOTFOUNT	該当インデックスなし
HASH_INVALIDKEY	不正キー入力（キー長不一致）
HASH_KEYNULL	キーポインタ不正（NULL）
HASH_INVALIDDATA	不正データ入力（データ長不一致）
HASH_DATANULL	データポインタ不正（NULL）

5.3.5. 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     *pszBuff
)
```

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。
キーデータの末尾は NULL 文字をセットしてください。
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

データファイルのファイルポインタを指定します。
本関数を使用する前に、fopen 関数を使用してバイナリ書き込みモード(“rb+“)でデータファイルをオープンしてください。

IndexFilePointer

インデックスファイルのファイルポインタを指定します。
本関数を使用する前に、fopen 関数を使用してバイナリ書き込みモード(“rb+“)でインデックスファイルをオープンしてください。

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(L-K)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。

HASH_OPERATIONOK	:正常終了
DATA_NOFILEWRITE	:データファイル ライトエラー
DATA_FILEOVERFLOW	:データファイル フル
DATA_NOFILEOPEN	:データファイル 未オープン
DATA_NOFILESEEK	:データファイル シークエラー
IDX_NOFILEREAD	:インデックスファイル リードエラー
IDX_NOFILEWRITE	:インデックスファイル ライトエラー
IDX_NOFILEOPEN	:インデックスファイル 未オープン
IDX_NOFILESEEK	:インデックスファイル シークエラー
HASH_NOMEMORY	:実行メモリ不足
HASH_KEYNOTFOUND	:該当インデックスなし
HASH_INVALIDKEY	:不正キー入力（キー長不一致）
HASH_KEYNULL	:キーポインタ不正（NULL）
HASH_INVALIDDATA	:不正データ入力（データ長不一致）
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。

一度に追加できるレコードは、1 レコードで、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されている数より少ないことが条件です。

6. 付録

6.1. 機能比較

端末毎のサポート状況を示します。

6.1.1. 関数一覧

アプリケーション支援ライブラリ

(1) 日付チェック関数

関数名	機能	DT-930	DT-970
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	○	○

(2) ブロックチェック関数

関数名	機能	DT-930	DT-970
ht_CheckSumm	チェックサム計算	○	○
ht_CalcCRC_ANSI	CRC 計算（ANSI 規格）	○	○
ht_CalcCRC_CCITT	CRC 計算（CCITT 規格）	○	○
ht_CalcCRC_X	CRC 計算（XMODEM 用）	○	○
ht_CalcLRC	LRC 計算	○	○
ht_CheckCD	チェックデジット計算	○	○

(3) 入力関数

関数名	機能	DT-930	DT-970
ht_FCWait	制御キー入力	○	○
ht_StrInp	文字列入力	○	○
ht_NumInp	数値入力	○	○
ht_DateInp	日付入力	○	○
ht_TimeInp	時刻入力	○	○
ht_ShiftMode	シフトキー制御	○	○

(4) 通信関数

関数名	機能	DT-930	DT-970
ht_MLTsend	マルチドロップ送信	○	
ht_MLTrecv	マルチドロップ受信	○	
ht_FLNKsend	FLINK 送信	○	○
ht_FLNKrecv	FLINK 受信	○	○
Ir_c_open	Ir 回線のオープン	○	○
Ir_c_close	Ir 回線のクローズ	○	○
Ir_c_status	Ir 回線のステータス取得	○	○
Ir_c_hold	通信ポートの占有／解除	※	※
Ir_c_chkopen	Ir ポートのオープンチェック	○	○
Ir_c_dout	指定文字数分のデータ送信	○	○
Ir_c_din	受信バッファの 1 文字読み出し	○	○
Ir_c_tmdin	受信バッファの 1 文字読み出し(タイムアウト付き)	○	○
Ir_c_out	1 文字送信	○	○
Ir_c_break	ブレーク信号の送出／停止	○	○
Ir_c_txrx	送受信の有効／無効	※	※
Ir_c_iobox	送信の設定／解除	※	※
Ir_c_irout	指定文字数分のデータ送信	○	○
Ir_c_timer	CS、DR、CD 信号のタイムアウト設定	※	※
Ir_c_rs	RS 信号の ON／OFF 設定	○	○
Ir_c_er	ER 信号の ON／OFF 設定	○	○
Ir_c_errs	ER および RS 信号の ON／OFF 設定	○	○
Ir_c_flush	Ir ポートのバッファクリア	○	○
Ir_c_bfsts	Ir ポートのバッファ文字数チェック	○	○
Ir_c_errbfring	エラーコードバッファリングの設定	※	※
Ir_c_rderrsts	エラーステータスの取得	○	○
Ir_c_chghdr	受信ハンドらの切り替え	※	※

※ シリアル通信関数(c_xxx)と同等のインターフェースを提供するためのダミー関数です。
パラメータのチェックのみが行われ、本来の機能は働きません。

(5) ファイル制御関数

関数名	機能	DT-930	DT-970
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	○	○

(6) サービス関数

関数名	機能	DT-930	DT-970
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	○	○

Bluetooth プリンタライブラリ

関数名	機能	DT-930	DT-970
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	○	○

TEC IrDA プリンタライブラリ

関数名	機能	DT-930	DT-970
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_send2	赤外線でのタイムアウト付きコマンド送信	○	
prn_tecinf_status	赤外線でのステータスリード	○	

モバイルプリンタ制御ライブラリ

関数名	機能	DT-930	DT-970
MPRInit	リソースの確保、及び通信回線の初期化	○	○
MPRDeinit	リソースの解放、及び通信回線の終了処理	○	○
MPROpen	通信回線のオープン	○	○
MPRClose	通信回線のクローズ	○	○
MPRSend	データの送信	○	○
MPRReceive	データの受信	○	○
MPRCalcCRC16	CRC(CRC-16)の算出	○	○
MPRSetIrDA	IrDA 通信パラメータの設定	○	○
MPRGetIrDA	IrDA 通信パラメータの取得	○	○
MPRSetBluetooth	Bluetooth 通信パラメータの設定	○	○
MPRGetBluetooth	Bluetooth 通信パラメータの取得	○	○

高速ファイルサーチライブラリ

関数名	機能	DT-930	DT-970
iHashAssign	インデックスファイルの生成	○	○
iHashRead	データの読み出し	○	○
iHashWrite	データの書き換え	○	○
iHashAdd	データの追加	○	○

ビットマップ表示ライブラリ

関数名	機能	DT-930	DT-970
bmp_iDisplayBmpImage	BMP 形式ファイル指定による表示	○	※1
bmp_iDisplayBmpData	BMP 形式データ指定による表示	○	※1

※1 デバイス制御ライブラリの画面表示関数として提供

送受信切替ライブラリ

関数名	機能	DT-930	DT-970
c_out	c_out の差し替え関数	○	
c_dout	c_dout の差し替え関数	○	

IO BOX 検出切替ライブラリ

関数名	機能	DT-930	DT-970
io_detect	IO BOX 検出の有効／無効の切り替え	○	

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）

https://casio.jp/support/ht/

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)