CASIO

DT-970SDK

デバイス制御ライブラリ
リファレンスマニュアル

DT-970 に搭載のハードウェアを制御する API を、
リファレンス形式で説明します。

Ver. 1.05

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。

© 2013-2016 カシオ計算機株式会社

Microsoft, MS, ActiveSync, Active Desktop, Outlook, Windows, Windows NT, および Windows ロゴは、米国 Microsoft Corporation の米国およびその他の国における登録商標または商標です。Microsoft 社の製品は、OEM 各社に、Microsoft Corporation の 100%出資子会社である Microsoft Licensing, Inc.によりライセンス供与されています。

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	19,77,226	カシオ IR インタフェースの記述を削除
		28,30,31	FIL_FSTAT、FIND_T、DIR_TBL 構造体の説明を追加
		67,68	ソケット構造体の説明を追加
		87	s_beep 関数の説明を修正
		126	key_fnc_mode 関数に対する通知待ち関数を記載
		145	表示コードの表を修正
		157	文字コードの表示に関する表を修正
		159,160	スクロール制御の説明を修正
		162	改行に関する補足説明を追加
		171	lcd_string2 関数の説明を修正
		11,180,181	ビットマップ表示関数のヘッダについて説明を追加
		182	文字入力モードの図と説明を訂正
		197	OBR 読み取り可能コードの名称を修正
		199,202	OBR 制御の読み取り桁数の値を訂正
		205	OBR 読み取り異常時の LED 制御仕様を訂正
		40-47	OBR_modewt 関数のパラメータ説明を M_TBL 構造体の節へ移動
		40-47 208-210	IATA, RSS-14 Stacked, RSS Expanded Stacked の設定に関する仕様を訂正
		54,278,282	iOS に HID 接続する場合の補足説明を追加
		299-320	ソケット関数に対し、構造体へのハイパーリンクを設定
		70,71,326, 327,404	ソケット関数が使用する構造体の説明場所を移動し、構造体一覧に追加
		328	IrDA-USB クレードルによる通信構成図を修正
		375	ppp_init 関数のパラメータを削除
		394,395	通信部とバーコード部の関数一覧に、補足を記載
1.02	2014.07	89	pwr_vibrator 関数の注意事項を追加
		140,143	イベント通知機能のサンプルコードの誤りを訂正
		405	裏表紙を変更
1.03	2014.09	76,80,395	dat_get_device_id 関数（シリアル番号読み出し）の追加
1.04	2016.06	218	OBR_setmargincheck 関数の注意事項を追加
1.05	2016.08	286	BT_Write 関数の戻り値を追加

目次

1. はじめに

このマニュアルは DT-970 に搭載した各種デバイスを制御するためのライブラリについて記載します。

1.1 構成ファイルについて

DT-970 デバイス制御ライブラリを構成するさまざまなファイルについて説明します。

ヘッダファイル

ファイル名	内容
ITRON.H	システム用データと関数の定義
CMNDEF.H	BIOS 用データと構造体の定義
BIOS1DEF.H	BIOS ファンクションコールジャンプテーブルの型定義
BIOS1MAC.H	BIOS ファンクションコールマクロの定義
BIOS5DEF.H	Bluetooth ファンクションコールジャンプテーブルの型定義
BIOS5MAC.H	Bluetooth 通信用マクロの定義
DTBMP.H	ビットマップ表示関数用データの定義

オブジェクトファイル

AP_START.OBJ	アプリケーション初期化モジュールオブジェクト
AP_START1.OBJ	アプリケーション初期化モジュールオブジェクト(DT-930 互換表示モード 1)
AP_START2.OBJ	アプリケーション初期化モジュールオブジェクト(DT-930 互換表示モード 2)

ライブラリ

RXCLIB.LIB	C 言語標準ライブラリ

1.1.1 基本的な使い方

使用する関数に応じ、下記のヘッダファイルとライブラリファイルを使用してください。

使用する関数	ヘッダファイル	ライブラリファイル
C 言語の標準ライブラリ関数	各関数の定義ヘッダ ※	RXCLIB.LIB
μITRON 関数(ref_flg, clr_flg, wai_flg)	ITRON.H	―
デバイス制御ライブラリ関数	BIOS1MAC.H BIOS1DEF.H BIOS5MAC.H BIOS5DEF.H CMNDEF.H	―

(9) 詳細は“RENESAS ユーザーズマニュアル RX コーディング編”を参照してください。

アプリケーションは、本ライブラリで提供の関数(外部シンボル)定義ファイルを用いて、単独でコンパイル／リンクします。(OS／システムの実体とはリンクしません)

2. データ型

DT-970 C ライブラリで使用するデータ型について説明します。

2.1 基本データ型

C 言語標準データ型（※開発環境のデフォルト設定時）

型	サイズ[*1]	境界[*1]	符号	最小値	最大値
char (signed char) [*3]	1	1	有	-2^7(-128)	2^7-1(127)
unsigned char	1	1	無	0	2^8-1(255)
short	2	2	有	-2^{15}(-32768)	2^{15}-1(32767)
unsigned short	2	2	無	0	2^{16}-1（65535）
int	4	4	有	-2^{31}(-2147483648)	2^{31}-1（2147483647）
unsigned int	4	4	無	0	2^{32}-1(4294967295)
long	4	4	有	-2^{31}(-2147483648)	2^{31}-1（2147483647）
unsigned long	4	4	無	0	2^{32}-1(4294967295)
enum	4	4	有	-2^{31}(-2147483648)	2^{31}-1（2147483647）
float	4	4	有	-∞	∞
double long double	8[*2]	4	有	-∞	∞
ポインタ	4	4	無	0	2^{32}-1(4294967295)

※1 サイズ・境界の単位はバイトです。
※2 double=8 オプションの指定を解除した場合、サイズは 4 バイトになります。
※3 signed_char オプションの指定を解除した場合は、unsigned char 型として扱います。

DT-970 データ型

型	内容	対応する C 言語標準データ型
B	符号付き 8 ビット整数	signed char
H	符号付き 16 ビット整数	short
W	符号付き 32 ビット整数	long
UB	符号なし 8 ビット整数	unsigned char
UH	符号なし 16 ビット整数	unsigned short
UW	符号なし 32 ビット整数	unsigned long
VB	データタイプが一定しない(8 ビット)	signed char
VH	データタイプが一定しない(16 ビット)	short
VW	データタイプが一定しない(32 ビット)	long
VP	データタイプが一定しないものへのポインタ	void *
FP	プログラム先頭アドレス	void (*)()
ID	オブジェクト ID	short
ER	エラーコード	long
FLGPTN	イベントフラグのビット・パターン	unsigned long
MODE	サービス・コールの動作モード	unsigned short
FN	機能コード	W
TMO	タイムアウト（tick 単位）	signed long
in_addr_t	IPv4 アドレス	unsigned long

2.2 構造体

DT-970 C ライブラリで使用する構造体を説明します。

2.2.1 DAT_PWR_STR 構造体

電源関連のシステムデータを格納します。

```
typedef struct sys_pwr{
   W      apo;
   W      abo;
   W      res_md;
} DAT_PWR_STR;
```

メンバ

apo
APO 時間設定を 0～59 分の範囲で格納します。

abo
ABO 時間設定を 10～59 秒の範囲で格納します。

res_md
レジュームの ON/OFF を次の値で格納します。
RESUME_ON　　:レジューム ON
RESUME_OFF　　:レジューム OFF

参照

dat_system 関数

2.2.2 DAT_KEY_STR 構造体

キー関連のシステムデータを格納します。

```
typedef struct sys_key{
   W      clk_md;
} DAT_KEY_STR;
```

メンバ

clk_md
クリック音の ON/OFF を次の値で格納します。
CLICK_ON　　:クリック音 ON
CLICK_OFF　　:クリック音 OFF

参照

dat_system 関数

2.2.3 DAT_OBR_STR 構造体

OBR 関連のシステムデータを格納します。

```
typedef struct sys_obr{
   W      rd_ct;
   W      cmp_ct;
   W      scn_tm;
} DAT_OBR_STR;
```

メンバ

rd_ct
読み取り回数を 1～9 回の範囲で格納します。

cmp_ct
照合回数を 1～9 回の範囲で格納します。

scn_tm
スキャンタイムアウト時間を 1～9 秒の範囲で格納します。

参照

dat_system 関数

2.2.4 DAT_DSP_STR 構造体

表示関連のシステムデータを格納します。

```
typedef struct sys_disp{
   W      font_md;
   W      lang_md;
} DAT_DSP_STR;
```

メンバ

font_md

フォントモードを次の値で格納します。

FONT6_SET	:6 ドットモード
FONT8_SET	:8 ドットモード
FONT10_SET	:10 ドットモード

lang_md

日本語／英語モードを次の値で格納します。

JPN_SET	:日本語モード
ENG_SET	:英語モード

参照

dat_system 関数

2.2.5 DAT_DSP_STR2 構造体

表示関連のシステムデータを格納します。

```
typedef struct sys_disp2{
   W      font_kd;
   W      cont_md;
   W      cont_df;
} DAT_DSP_STR2;
```

メンバ

font_kd
フォント修飾を次の値で格納します。
FONT_NORMAL　:NORMAL
FONT_BOLD　　:BOLD

cont_md
コントラスト設定値を 0～15 の範囲で格納します。

cont_df
コントラスト差分を-7～7 の範囲で格納します。

参照

dat_system 関数

2.2.6 DAT_DSP_STR3 構造体

画面表示に関するシステムデータを格納します。

```
typedef struct sys_disp3 {
  W   line_gap;
  W   task_bar;
} DAT_DSP_STR3;
```

メンバ

line_gap

行間挿入の種類を格納します。

DSP3_LGAP_00	:行間挿入しない
DSP3_LGAP_01	:行間挿入する（縮小 ANK:1dot、標準 ANK／漢字:2dot）
DSP3_LGAP_02	:行間挿入する（縮小 ANK:2dot、標準 ANK／漢字:4dot）
DSP3_LGAP_03	:行間挿入する（縮小 ANK:3dot、標準 ANK／漢字:6dot）
DSP3_LGAP_04	:行間挿入する（縮小 ANK:4dot、標準 ANK／漢字:8dot）
DSP3_LGAP_05	:行間挿入する（縮小 ANK:5dot、標準 ANK／漢字:10dot）
DSP3_LGAP_06	:行間挿入する（縮小 ANK:6dot、標準 ANK／漢字:12dot）
DSP3_LGAP_07	:行間挿入する（縮小 ANK:7dot、標準 ANK／漢字:14dot）
DSP3_LGAP_08	:行間挿入する（縮小 ANK:8dot、標準 ANK／漢字:16dot）
DSP3_LGAP_09	:行間挿入する（縮小 ANK:9dot、標準 ANK／漢字:18dot）

task_bar

タスクバー(シンボル表示領域)を表示するか否かを格納します。

DSP3_TSKBAR_OFF	:タスクバーを表示しない
DSP3_TSKBAR_ON	:タスクバーを表示する

参照

dat_system 関数

2.2.7 DAT_COMINF_STR 構造体

通信関連のシステムデータを格納します。

```
typedef struct sys_tty0{
   W      com_proto;
   W      com_port;
} DAT_COMINF_STR;
```

メンバ

com_proto
プロトコル種別を次の値で格納します。
PRT_FLINK　　:FLINK

com_port
通信ポートを次の値で格納します。
IR_PORT　　:IR ポート

参照

dat_system 関数

2.2.8 DAT_COM_STR 構造体

通信(IrDA)の詳細設定に関するシステムデータを格納します。

```
typedef struct sys_tty{
   W      speed;
   W      length;
   W      parity;
   W      stop_bit;
} DAT_COM_STR;
```

メンバ

speed

転送速度を次の値で格納します。

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

length

データ長を次の値で格納します。

CHAR_8	:8 bit
CHAR_7	:7 bit

parity

パリティビットを次の値で格納します。

PARI_NON	:なし
PARI_ODD	:奇数
PARI_EVN	:偶数

stop_bit

ストップビットを次の値で格納します。

STOP_1	:1 bit
STOP_2	:2 bit

参照

dat_system 関数

2.2.9 DAT_TIM_STR 構造体

ブザー音量に関するシステムデータを格納します。

```
typedef struct sys_time{
   W      buzzer;
} DAT_TIM_STR;
```

メンバ

buzzer

ブザー音量を次の値で格納します。

BUZZ_OFF	:音量 OFF
BUZZ_LOW	:音量小
BUZZ_MID	:音量中
BUZZ_LOUD	:音量大

参照

dat_system 関数

2.2.10 DAT_PRO_STR 構造体

プロトコル関連のシステムデータを格納します。
マルチドロップ、DT500 に関するパラメータは使用できません。

```
typedef struct sys_pro{
   /* マルチドロップ  */ /* 使用しません */
   W      non_rec_tmout;
   W      non_retry_ct;
   W      mal_rec_tmout;
   W      ptp_snd_tmout;
   W      ptp_rec_tmout;
   W      ptp_rec_retry_ct;

   /* FLINK           */
   W      irda_tmout;
   W      irda_rec_tmout;
   W      dr_tmout;
   W      cs_tmout;
   W      cd_tmout;

   /* DT500           */ /* 使用しません */
   W      sirial_no;
   W      level_parity;
   W      bht_tmout;
} DAT_PRO_STR;
```

メンバ

irda_tmout
IrDA セッション確立タイムアウトを 0～3600 秒の範囲で格納します。

irda_rec_tmout
IrDA 受信タイムアウトを 0～600 秒の範囲で格納します。

dr_tmout
IrDA セッション終了タイムアウトを 0～600 秒の範囲で格納します。

cs_tmout, cd_tmout
予約領域。使用しません。

参照

dat_system 関数

2.2.11 DAT_SYS_STR 構造体

端末に関するシステムデータを格納します。

```
typedef struct sys_dat{
   UB     sys_id[7];
   UB     bios_ver[7];
   UW     mac_type;
} DAT_SYS_STR;
```

メンバ

sys_id
機器 ID を格納します。

bios_ver
BIOS バージョンを格納します。読み取り専用です。

mac_type
機器種別を格納します。読み取り専用です。

参照

dat_system 関数

2.2.12 DAT_SYS_STR2 構造体

端末に関するシステムデータを格納します。

```
typedef struct sys_dat2{
   UB     dlr_id[7];
   UB     patch_ver[7];
} DAT_SYS_STR2;
```

メンバ

dlr_id
代理店 ID を格納します。書き込み専用です。

patch_ver
パッチバージョンを格納します。読み取り専用です。

参照

dat_system 関数

2.2.13 DAT_BAT_STR 構造体

電池に関するシステムデータを格納します。

```
typedef struct sys_bat {
  W   auto_disp;
  W   type;
} DAT_BAT_STR;
```

メンバ

auto_disp
電池を交換する度に、電池の種類(乾電池／充電池)の設定画面を表示するか否かを格納します。
BAT_AUTODISP_ON　　:電池交換する度に、設定画面を表示する
BAT_AUTODISP_OFF　　:電池交換しても、設定画面を表示しない

type
電池の種類を格納します。
BAT_TYPE_DRY　　:乾電池
BAT_TYPE_RECHARGE　:充電池

参照

dat_system 関数

2.2.14 DAT_BTH_STR 構造体

Bluetooth に関するシステムデータを格納します。

```
typedef  struct sys_bth {
  UB  bt_dev_name[BT_DEV_NAME_BUF_LEN];
} DAT_BTH_STR;
```

メンバ

bt_dev_name
BT デバイス名称を格納します。
指定可能な文字数は最大 14 バイトです。

参照

dat_system 関数

2.2.15 DAT_LAN_STR 構造体

LAN 接続する際に、クレードルに設定されている IP を優先するか、端末に設定されている IP を優先するかを格納します。

```
typedef struct sys_lan {
  W   ip_pri;
} DAT_LAN_STR;
```

メンバ

ip_pri
IP の優先デバイスを格納します。
LAN_IPADDR_CR　　:クレードル IP を優先する。
LAN_IPADDR_HT　　:端末 IP を優先する。

参照

dat_system 関数

2.2.16 DAT_HIP_STR 構造体

端末 IP に関するシステムデータを格納します。

```
typedef struct sys_hip {
  W     dhcp;
  UB    ipaddr[IPADDR_BUF_LEN];
  UB    smask[IPADDR_BUF_LEN];
  UB    gateway[IPADDR_BUF_LEN];
  UB    dns1[IPADDR_BUF_LEN];
  UB    dns2[IPADDR_BUF_LEN];
} DAT_HIP_STR;
```

メンバ

dhcp
DHCP を使用するか否かを格納します。
LAN_DHCP_ON　　:IP を DHCP から取得する
LAN_DHCP_OFF　　:固定 IP

ipaddr
DT-970 の IP アドレスをドットノーテーション形式(xxx.xxx.xxx.xxx)で格納します。

smask
サブネットマスクをドットノーテーション形式(xxx.xxx.xxx.xxx)で格納します。

gateway
ゲートウェイアドレスをドットノーテーション形式(xxx.xxx.xxx.xxx)で格納します。

dns1, dns2
DNS1 または DNS2 アドレスをドットノーテーション形式(xxx.xxx.xxx.xxx)で格納します。

参照

dat_system 関数

2.2.17 DAT_CIP_STR 構造体

LAN クレードルの IP に関する情報を格納します。

```
typedef struct sys_cip {
  UB   ipaddr[IPADDR_BUF_LEN];
  UB   smask[IPADDR_BUF_LEN];
  UB   gateway[IPADDR_BUF_LEN];
} DAT_CIP_STR;
```

メンバ

ipaddr
LAN クレードルの IP アドレスをドットノーテーション形式(xxx.xxx.xxx.xxx)で格納します。

smask
サブネットマスクをドットノーテーション形式(xxx.xxx.xxx.xxx)で格納します。

gateway
ゲートウェイアドレスをドットノーテーション形式(xxx.xxx.xxx.xxx)で格納します。

参照

dat_system 関数

2.2.18 DAT_USB_STR 構造体

PC から端末のドライブを USB 経由で参照する際の、参照先のドライブおよび接続方式（クレードル／ケーブル）を格納します

```
typedef struct sys_usb {
  W    cnct_mode;
} DAT_USB_STR;
```

メンバ

cnct_mode
参照先のドライブおよび接続方式（クレードル／ケーブル）を格納します。

USB_CNCTMODE_DISABLE	:接続しない
USB_CNCTMODE_A_CABLE	:A ドライブにケーブルで接続
USB_CNCTMODE_B_CABLE	:B ドライブにケーブルで接続
USB_CNCTMODE_D_CABLE	:D ドライブにケーブルで接続
USB_CNCTMODE_A_CRADLE	:A ドライブにクレードルで接続
USB_CNCTMODE_B_CRADLE	:B ドライブにクレードルで接続
USB_CNCTMODE_D_CRADLE	:D ドライブにクレードルで接続

参照

dat_system 関数

2.2.19 DAT_HST_STR 構造体

LAN 接続する際の接続先端末に関する情報を格納します。

```
typedef struct sys_hst {
  UB   ipaddr[IPADDR_BUF_LEN];
  W    port;
} DAT_HST_STR;
```

メンバ

ipaddr
IP アドレスをドットノーテーション形式(xxx.xxx.xxx.xxx)で格納します。

port
ポート番号を格納します。

参照

dat_system 関数

2.2.20 FIL_FSTAT 構造体

ファイル属性情報を格納します。

```
typedef struct stat{
   UW     filesize;
   UH     date;
   UH     time;
   B      attr;
} FIL_FSTAT;
```

メンバ

filesize
ファイルサイズをバイト単位で格納します。

date
ファイルの日付を次のフォーマットで格納します。
15～9 bit　:年(0～99)　0 は 1980 年、99 は 2079 年を示します。
8～5 bit　:月(1～12)
4～0 bit　:日(1～31)

time
ファイルの時刻を次のフォーマットで格納します。
15～11 bit　:時(0～23)
10～5 bit　:分(0～59)
4～0 bit　:秒(0～29)　2 秒単位の値です。1 は 2 秒、29 は 58 秒を示します。

attr
ファイルの属性を次の値の論理和で格納します。
_A_NORMAL　:読み書き可
_A_RDONLY　:読み取り専用
_A_HIDDEN　:隠しファイル
_A_SYSTEM　:システム
_A_VOLID　:ボリューム ID
_A_SUBDIR　:サブディレクトリ
_A_ARCH　:アーカイブ

参照

fil_fstat 関数

2.2.21 FIL_SIZE 構造体

ファイルの個数と総サイズ(4GB 以内)を格納します。

```
typedef struct cnt_and_size{
   UW     cnt;
   UW     size;
} FIL_SIZE;
```

メンバ

cnt
ファイルの個数を格納します。

size
ファイルの総サイズをバイト単位で格納します。

参照

fil_filesize 関数

2.2.22 FIL_SIZE2 構造体

ファイルの個数と総サイズを格納します。

```
typedef struct cnt_and_size2{
   UW     cnt;
   UW     hsize;
   UW     lsize;
} FIL_SIZE2;
```

メンバ

cnt
ファイルの個数を格納します。

hsize
ファイルの総サイズ(64bit の上位 32bit)をバイト単位で格納します。

lsize
ファイルの総サイズ(64bit の下位 32bit)をバイト単位で格納します。

参照

fil_filesize2 関数

2.2.23 FIND_T 構造体

ファイル検索の結果を格納します。

```
typedef struct find_t{
   B      reserved[21];
   B      attrib;
   UH     wr_time;
   UH     wr_date;
   W      size;
   B      name[13];
} FIND_T;
```

メンバ

reserved
予約領域。使用しません。

attrib
ファイルの属性を次の値の論理和で格納します。

_A_NORMAL	:読み書き可
_A_RDONLY	:読み取り専用
_A_HIDDEN	:隠しファイル
_A_SYSTEM	:システム
_A_VOLID	:ボリューム ID
_A_SUBDIR	:サブディレクトリ
_A_ARCH	:アーカイブ

wr_time
ファイルの最終更新時刻を次のフォーマットで格納します。

15～11 bit	:時(0～23)	
10～5 bit	:分(0～59)	
4～0 bit	:秒(0～29)	2 秒単位の値です。1 は 2 秒、29 は 58 秒を示します。

wr_date
ファイルの更新日付を次のフォーマットで格納します。

15～9 bit	:年(0～99)	0 は 1980 年、99 は 2079 年を示します。
8～5 bit	:月(1～12)	
4～0 bit	:日(1～31)	

size
ファイルサイズをバイト単位で格納します。

name
ファイル、またはディレクトリの名前を格納します。

参照

fil_findfirst 関数

2.2.24 DIR_TBL 構造体

DT-700 互換ファイルシステムのファイル情報を格納します。

```
typedef struct fcb {
   B      filename[8];
   B      extension[3];
   W      top_adr;
   W      size;
   UW     date_tm;
   W      attribute;
} DIR_TBL;
```

メンバ

filename
最大 8 バイトのファイル名を格納します。英文字はすべて大文字です。
ファイル名が 8 バイト未満の場合は、NULL をパディングします。

extension
最大 3 バイトの拡張子を格納します。英文字はすべて大文字です。
拡張子が 3 バイト未満の場合は、NULL をパディングします。

top_adr
ファイルデータ領域に格納されている、該当ファイルの先頭アドレスを格納します。

size
ファイルデータ領域に格納されている、該当ファイルのファイルサイズをバイト単位で格納します。

date_tm
ファイル最終更新日時を次のフォーマットで格納します。

31～25 bit	:年(0～99)	0 は 1980 年、99 は 2079 年を示します。
24～21 bit	:月(1～12)	
20～16 bit	:日(1～31)	
15～11 bit	:時(0～23)	
10～5 bit	:分(0～59)	
4～0 bit	:秒(0～59)	2 秒単位の値です。1 は 2 秒、29 は 58 秒を示します。

attribute
ファイルの属性を格納します。(常に 0 を格納します)

参照

dat_fdir 関数

2.2.25 KEY_INP 構造体

1 文字入力情報を格納します。

```
typedef  struct  st_key_inp  {
   UB     ext;
   UB     echo;
   H      font_size;
   H      type;
   UH     column_pos;
   UH     line_pos;
} KEY_INP ;
```

メンバ

ext
入力の終了条件を次の値の組合せで格納します。
KEY_INT_EXT　:イベント通知キー押下
KEY_LB_EXT　:LB 発生
KEY_OBR_EXT　:バーコード読み込み完了
KEY_IO_EXT　:クレードル検出
KEY_NON_EXT　:なし

echo
エコーバックの ON/OFF を次の値で格納します。
ECHO_ON　:エコーバック ON
ECHO_OFF　:エコーバック OFF

font_size
エコーバックのフォントサイズを次の値で格納します。
LCD_ANK_LIGHT　:縮小 ANK
LCD_ANK_STANDARD　:標準 ANK

type
エコーバックの表示方法を次の値の組合せで格納します。
LCD_ATTR_NORMAL　:通常
LCD_ATTR_REVERS　:反転
LCD_ATTR_WIDTH　:強調

column_pos
入力桁座標を格納します。

line_pos
入力行座標を格納します。

解説

エコーバックの強調反転表示を行なう場合は、*type* メンバに LCD_ATTR_REVERS と LCD_ATTR_WIDTH の OR を指定します。
echo メンバに ECHO_OFF を指定した場合、*font_size*、*type*、*column_pos*、*line_pos* それぞれのチェックは行ないません。

参照

key_read 関数

2.2.26 KEY_INPS 構造体

文字列入力／数値入力情報を格納します。

```
typedef struct st_key_inps {
   UB     ext;
   UB     echo;
   H      font_size;
   H      type;
   UH     len;
   UH     column_pos;
   UH     line_pos;
   UH     column_len;
   UH     clr_type ;
} KEY_INPS ;
```

メンバ

ext

入力の終了条件を次の値の組み合わせで格納します。

KEY_INT_EXT	:イベント通知キー押下
KEY_LB_EXT	:LB 発生
KEY_OBR_EXT	:バーコード読み込み完了
KEY_CLR_EXT	:CLR キー押下
KEY_IO_EXT	:クレードル検出
KEY_FULL_BEEP	:入力領域フルで BEEP 音※
KEY_FULL_CHR	:入力領域フルで処理終了
KEY_NON_EXT	:なし

(9) BEEP 音は鳴りますが終了はしません

echo

エコーバックの ON/OFF を次の値で格納します。

ECHO_ON	:エコーバック ON
ECHO_OFF	:エコーバック OFF

font_size

エコーバックのフォントサイズを次の値で格納します。

LCD_ANK_LIGHT	:縮小 ANK
LCD_ANK_STANDARD	:標準 ANK

type

エコーバックの表示方法を次の値の組み合わせで格納します。

LCD_ATTR_NORMAL	:通常
LCD_ATTR_REVERS	:反転
LCD_ATTR_WIDTH	:強調

len

入力文字数をバイト単位で格納します。

column_pos
入力桁座標を格納します。

line_pos
入力行座標を格納します。

column_len
入力文字の位置を半角で格納します。
key_num 関数では、使用しません。

clr_type
初期データ表示後のクリアをする/しないを格納します。
KEY_NUM_CLR_ON　　:クリアします
KEY_NUM_CLR_OFF　　:クリアしません
key_string 関数では、使用しません。

解説

エコーバックの強調反転表示を行なう場合は、*type* パラメータに LCD_ATTR_REVERS と LCD_ATTR_WIDTH の OR を指定します。
echo パラメータに ECHO_OFF を指定した場合、*font_size*、*type*、*column_pos*、*line_pos* それぞれのパラメータのチェックは行ないません。

参照

key_string 関数、key_num 関数

2.2.27 KEYFORM 構造体

キーコードデータを格納します

```
typedef struct stKeyCode{
   UB     attr;
   UB     code;
} KEYFORM ;
```

メンバ

attr

code に指定するキーコードの属性を格納します。

00h　　:*code* メンバがアスキーコード

FFh　　:*code* メンバが内部処理コード

アプリケーションプログラムがキー入力関数でキーコードを取得できるのは、*attr* に 00h を指定した *code* のみです。

code

キーコードのコードを格納します。

attr が 00h の場合は、「11.1.1 1バイトコード」に示すコードを指定します。

attr が FFh の場合は、下記の値を指定します。

00h　　:コントラストのアップ

01h　　:コントラストのダウン

02h　　:バックライトのオン・オフ

03h　　:バーコードの読み取り

参照

key_fnc 関数

2.2.28 KEYSEL 構造体

有効無効キーテーブルを格納します。

```
typedef struct stKeySel {
   UB     s;
   UB     bs;
   UB     clr;
   UB     ten1;
   UB     ten2;
   UB     ten3;
   UB     ten4;
   UB     ten5;
   UB     ten6;
   UB     ten7;
   UB     ten8;
   UB     ten9:
   UB     ten0;
   UB     ten;
   UB     ent;
   UB     func1;
   UB     func2;
   UB     func3;
   UB     func4;
   UB     func5;
   UB     func6;
   UB     func7;
   UB     func8;
   UB     mltr;
   UB     mltl;
   UB     up;
   UB     down;
   UB     right;
   UB     left;
} KEYSEL;
```

メンバ

s	:入力モード切替(S)	*bs*	:後退(BS)	*clr*	:クリア(CLR)
ten1	:テンキー1	*ten2*	:テンキー2	*ten3*	:テンキー3
ten4	:テンキー4	*ten5*	:テンキー5	*ten6*	:テンキー6
ten7	:テンキー7	*ten8*	:テンキー8	*ten9*	:テンキー9
ten0	:テンキー0	*ten*	:小数点	*ent*	:リターン
func1	:F1(-)	*func2*	:F2(←)	*func3*	:F3(→)
func4	:F4(DEL)	*func5*	:F5(SP)	*func6*	:F6(▲)
func7	:F7(▼)	*func8*	:F8(BL)		
mltr	:マルチファンクションキーR	*mltl*	:マルチファンクションキーL		
up	:↑	*down*	:↓	*right*	:→
left	:←				

各メンバに対し、キー入力有効／無効を格納します。

KEY_MODE_ENA　　:キー入力有効
KEY_MODE_DIS　　:キー入力無効

参照

key_select 関数

2.2.29 DAY_DAT 構造体

日付データを格納します。

```
typedef struct day_tabl {
   UH     year;
   UB     month;
   UB     day;
} DAY_DAT;
```

メンバ

year
西暦年を 2000～2100 の範囲で格納します。

month
月を 1～12 の範囲で格納します。

day
日を 1～31 の範囲で格納します。

参照

s_dateset 関数、s_dateget 関数

2.2.30 TIM_DAT 構造体

時刻データを格納します。

```
typedef struct tim_tabl {
   UB     hour;
   UB     mint;
   UB     sec;
} TIM_DAT;
```

メンバ

hour
時を 0～23 の範囲で格納します。

mint
分を 0～59 の範囲で格納します。

sec
秒を 0～59 の範囲で格納します。

参照

s_timeset 関数、s_timeget 関数

2.2.31 M_TBL 構造体

OBR の動作モードを格納します。

```
Typedef struct m_tbl {
  UW      Code;
  UB      Cd39[6];
  UB      Nw7[6];
  UB      Wpcea[6];
  UB      Wpce[6];
  UB      Upcea[6];
  UB      Upce[6];
  UB      Idsf[6];
  UB      Itrf[6];
  UB      Cd93[6];
  UB      Cd128[6];
  UB      Msi[6];
  UB      Iata[6];
  UB      Rss14[6];
  UB      RssLtd[6];
  UB      RssExp[6];
  UB      Rss14S[6];
  UB      RssExpS[6];
  UB      Resv[15][6];
  UB      Type;
  UB      Gain;
  UB      Buzc;
  UB      Ledc;
  UB      Bufc;
  UB      Endc;
  UB      Mode;
  UB      Dumy;
} M_TBL;
```

メンバ

Code

読み取り可能コードを格納します。
詳細は、解説を参照してください。

Cd39[6]

Code39 の以下の情報を格納します。

Cd39[0]	:リザーブ
Cd39[1]	:Code39 最小桁数
Cd39[2]	:Code39 最大桁数
Cd39[3]	:Code39 出力フォーマット
Cd39[4]	:Code39 チェックデジット計算設定
Cd39[5]	:Code39 チェックキャラクタ出力設定

Nw7[6]

NW-7 の以下の情報を格納します。

Nw7[0] :リザーブ
Nw7[1] :NW-7 最小桁数
Nw7[2] :NW-7 最大桁数
Nw7[3] :NW-7 出力フォーマット
Nw7[4] :NW-7 チェックデジット計算設定
Nw7[5] :NW-7 チェックキャラクタ出力設定

Wpcea[6]

WPC Addon の以下の情報を格納します。

Wpcea[0] :リザーブ
Wpcea[1] :WPC Addon 最小桁数
Wpcea[2] :WPC Addon 最大桁数
Wpcea[3] :WPC Addon 出力フォーマット
Wpcea[4] :WPC Addon チェックデジット計算設定
Wpcea[5] :WPC Addon チェックキャラクタ出力設定

※ WPC Addon のチェックデジット計算を無効にする場合は、WPC Addon、WPC 双方のチェックデジット計算を無効に設定する必要があります。

Wpce[6]

WPC の以下の情報を格納します。

Wpce[0] :リザーブ
Wpce[1] :WPC 最小桁数
Wpce[2] :WPC 最大桁数
Wpce[3] :WPC 出力フォーマット
Wpce[4] :WPC チェックデジット計算設定
Wpce[5] :WPC チェックキャラクタ出力設定

※ WPC のチェックデジット計算を無効にする場合は、WPC Addon、WPC 双方のチェックデジット計算を無効に設定する必要があります。

Upcea[6]

UPC-E Addon の以下の情報を格納します。

Upcea[0] :リザーブ
Upcea[1] :UPC-E Addon 最小桁数
Upcea[2] :UPC-E Addon 最大桁数
Upcea[3] :UPC-E Addon 出力フォーマット
Upcea[4] :UPC-E Addon チェックデジット計算設定
Upcea[5] :UPC-E Addon チェックキャラクタ出力設定

※ UPC-E Addon のチェックデジット計算を無効にする場合は、UPC-E Addon、UPC-E 双方のチェックデジット計算を無効に設定する必要があります。

Upce[6]

UPC-E の以下の情報を格納します。

Upce[0] :リザーブ
Upce[1] :UPC-E 最小桁数
Upce[2] :UPC-E 最大桁数
Upce[3] :UPC-E 出力フォーマット
Upce[4] :UPC-E チェックデジット計算設定
Upce[5] :UPC-E チェックキャラクタ出力設定

※ UPC-E のチェックデジット計算を無効にする場合は、UPC-E Addon、UPC-E 双方のチェックデジット計算を無効に設定する必要があります。

Idsf[6]

Industrial 2of5 の以下の情報を格納します。

Idsf[0] :リザーブ
Idsf[1] :Industrial 2of5 最小桁数
Idsf[2] :Industrial 2of5 最大桁数
Idsf[3] :Industrial 2of5 出力フォーマット
Idsf[4] :Industrial 2of5 チェックデジット計算設定
Idsf[5] :Industrial 2of5 チェックキャラクタ出力設定

⑼ Industrial 2of5 と IATA を別々に設定することはできません。
Code に両方のコードを指定した場合、本パラメータ（Idsf[1]-Idsf[4]）に設定した値は IATA に対しても適用されます。

Itrf[6]

Interleaved 2of5 の以下の情報を格納します。

Itrf[0] :リザーブ
Itrf[1] :ITF 最小桁数
Itrf[2] :ITF 最大桁数
Itrf[3] :ITF 出力フォーマット
Itrf[4] :ITF チェックデジット計算設定
Itrf[5] :ITF チェックキャラクタ出力設定

Cd93[6]

Code93 の以下の情報を格納します。

Cd93[0] :リザーブ
Cd93[1] :Code93 最小桁数
Cd93[2] :Code93 最大桁数
Cd93[3] :Code93 出力フォーマット
Cd93[4] :Code93 チェックデジット計算設定
Cd93[5] :Code93 チェックキャラクタ出力設定

Cd128[6]

Code128 の以下の情報を格納します。

Cd128[0] :リザーブ
Cd128[1] :Code128 最小桁数
Cd128[2] :Code128 最大桁数
Cd128[3] :Code128 出力フォーマット
Cd128[4] :Code128 チェックデジット計算設定
Cd128[5] :Code128 チェックキャラクタ出力設定

Msi[6]

MSI の以下の情報を格納します。

Msi[0] :リザーブ
Msi[1] :MSI 最小桁数
Msi[2] :MSI 最大桁数
Msi[3] :MSI 出力フォーマット
Msi[4] :MSI チェックデジット計算設定
Msi[5] :MSI チェックキャラクタ出力設定

Iata[6]

IATA の以下の情報を格納します。

Iata[0] :リザーブ
Iata[1] :IATA 最小桁数
Iata[2] :IATA 最大桁数
Iata[3] :IATA 出力フォーマット
Iata[4] :IATA チェックデジット計算設定
Iata[5] :IATA チェックキャラクタ出力設定

⑼ Industrial 2of5 と IATA を別々に設定することはできません。
Code に両方のコードを指定した場合、*Idsf* に設定した値が IATA に対して適用され、本パラメータの値は使用されません。

Rss14[6]

RSS-14 の以下の情報を格納します。

Rss14[0] :リザーブ
Rss14[1] :RSS-14 最小桁数
Rss14[2] :RSS-14 最大桁数
Rss14[3] :RSS-14 出力フォーマット
Rss14[4] :RSS-14 チェックデジット計算設定
Rss14[5] :RSS-14 チェックキャラクタ出力設定

⑼ RSS-14 と RSS-14 Stacked を別々に設定することはできません。
Code に両方のコードを指定した場合、本パラメータに設定した値は RSS-14 Stacked に対しても適用されます。

RssLtd[6]

RSS Limited の以下の情報を格納します。

RssLtd[0] :リザーブ
RssLtd[1] :RSS Limited 最小桁数
RssLtd[2] :RSS Limited 最大桁数
RssLtd[3] :RSS Limited 出力フォーマット
RssLtd[4] :RSS Limited チェックデジット計算設定
RssLtd[5] :RSS Limited チェックキャラクタ出力設定

RssExp[6]

RSS Expanded の以下の情報を格納します。

RssExp[0]　:リザーブ
RssExp[1]　:RSS Expanded 最小桁数
RssExp[2]　:RSS Expanded 最大桁数
RssExp[3]　:RSS Expanded 出力フォーマット
RssExp[4]　:RSS Expanded チェックデジット計算設定
RssExp[5]　:RSS Expanded チェックキャラクタ出力設定

⑼ RSS Expanded と RSS Expanded Stacked を別々に設定することはできません。
Code に両方のコードを指定した場合、本パラメータに設定した値は RSS Expanded Stacked に対しても適用されます。

Rss14S[6]

RSS-14 Stacked の以下の情報を格納します。

Rss14S[0]　:リザーブ
Rss14S[1]　:RSS-14 Stacked 最小桁数
Rss14S[2]　:RSS-14 Stacked 最大桁数
Rss14S[3]　:RSS-14 Stacked 出力フォーマット
Rss14S[4]　:RSS-14 Stacked チェックデジット計算設定
Rss14S[5]　:RSS-14 Stacked チェックキャラクタ出力設定

⑼ RSS-14 と RSS-14 Stacked を別々に設定することはできません。
Code に両方のコードを指定した場合、*Rss14* に設定した値が RSS-14 Stacked に対して適用され、本パラメータの値は使用されません。

RssExpS[6]

RSS Expanded Stacked の以下の情報を格納します。

RssExpS[0]　:リザーブ
RssExpS[1]　:RSS Expanded Stacked 最小桁数
RssExpS[2]　:RSS Expanded Stacked 最大桁数
RssExpS[3]　:RSS Expanded Stacked 出力フォーマット
RssExpS[4]　:RSS Expanded Stacked チェックデジット計算設定
RssExpS[5]　:RSS Expanded Stacked チェックキャラクタ出力設定

⑼ RSS Expanded と RSS Expanded Stacked を別々に設定することはできません。
Code に両方のコードを指定した場合、*RssExp* に設定した値が RSS Expanded Stacked に対して適用され、本パラメータの値は使用されません。

Resv[15][6]

リザーブ領域です。

Type

読み取り方式の設定情報を格納します。

Gain

リザーブ領域です。

Buzc

ブザーの設定情報を格納します。

Ledc
LED の設定情報を格納します。

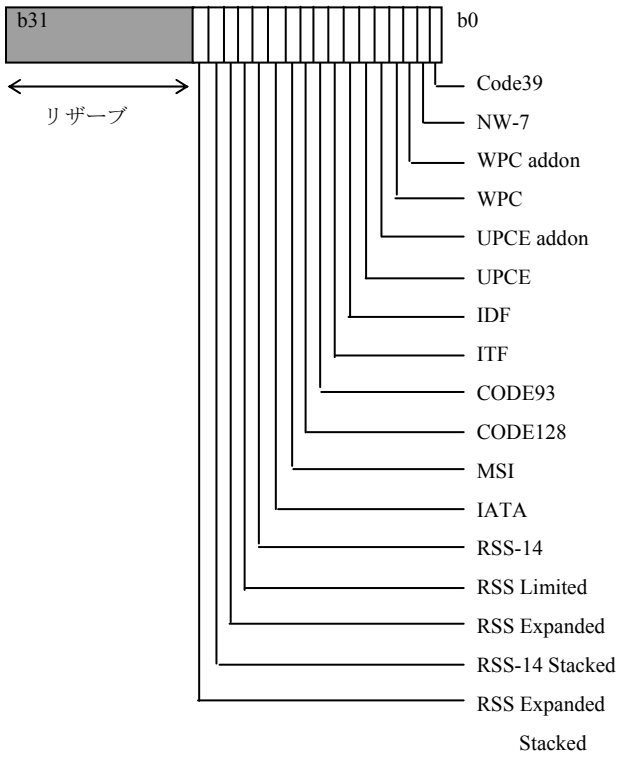
Bufc
出力方式の設定情報を格納します。

Endc
終了コードの設定情報を格納します。

Mode
段数読みの設定情報を格納します。

Dumy
リザーブ領域です。

解説

項目	内容	初期値	設定	参照
読み取り可能コード (4 バイト)	 *1 読み取るコードに該当するビットを ON (1)にします。 *2 読み取り性能を向上させるため、必要な読み取りコードのみ設定することを推奨します。	RSS 以外すべて (FFFh)	○	○

項目	内容	初期値	参照	設定
読み取り桁数の設定 出力フォーマットの設定 チェックデジットの実行指定 チェックキャラクタの出力指定 (192 バイト)		左の表参照	○	○

ﾘｻﾞｰﾌﾞ	Min	Max	出力 ﾌｫｰﾏｯﾄ	ﾁｪｯｸ ﾃﾞｼﾞｯﾄ	ﾁｪｯｸ ｷｬﾗｸﾀ	
FFh	2	38	0 *1	0	1	CODE39
FFh	2	38	0 *2	—	—	NW-7
FFh	10	18	0 *6	1	—	WPC(UPCE以外)addon
FFh	8	13	0 *7	1	—	WPC(UPCE以外)
FFh	9	12	0 *8	1	1	UPCE addon
FFh	7	7	1 *3	1	0	UPCE
FFh	2	40	—	0	1	Industrial 2of5 (*12)
FFh	4	40	—	1	1	ITF
FFh	3	40	—	1	—	CODE93
FFh	2	64	0 *4	1	—	CODE128
FFh	2	40	—	1 *5	1	MSI
FFh	2	40	—	1	—	IATA (*12)
FFh	14	14	0 *9	1	1	RSS-14 (*13)
FFh	14	14	0 *10	1	1	RSS Limited
FFh	1	74	—	1	1	RSS Expanded (*14)
FFh	14	14	0 *11	1	1	RSS-14 Stacked (*13)
FFh	1	74	—	1	1	RSS Expanded Stacked (*14)
FFh	—	—	—	—	—	(ﾘｻﾞｰﾌﾞ)

— 指定無し(各種コードの固定値)
変更不可(値のチェックなし)

チェックデジット　00h :チェック無し
01h :チェック有り

チェックキャラクタ　00h :出力無し
01h :出力有り

*1 CODE39出力フォーマット　00h :Start/Stopコード有り
01h :Start/Stopコード無し
02h: Full ASCII Start/Stopコード有り
03h: Full ASCII Start/Stopコード無し

*2 NW-7出力フォーマット　00h :Start/Stopコード有り
01h :Start/Stopコード無し

*3 UPCE出力フォーマット　00h :UPCAへの復元コード出力有り
01h :UPCAへの復元コード出力無し
02h :GTINフォーマットで出力
04h :先頭の0を削除して出力

*4 CODE128出力フォーマット　00h :変換後(ASCII)のデータを出力
02h :CODE128(×) EAN128(変換後)出力
08h :CODE128(変換後) EAN128(GS変換)出力
0Ah :CODE128(×) EAN128(GS変換)出力

*5 MSIチェックデジット　01h :1桁, mod10
03h :2桁, 1st:mod10 2nd:mod10

*6 WPCaddon出力フォーマット　00h :UPCAの先頭に0を付加して出力
01h :UPCAの先頭に0を削除して出力

*7 WPC出力フォーマット　bit0(0) :UPCAの先頭に0を付加して出力
bit0(1) :UPCAの先頭に0を削除して出力
bit1(0) :UPCAを通常フォーマットで出力
bit1(1) :UPCAをGTINフォーマットで出力
bit2(0) :JAN8/EAN8を通常フォーマットで出力
bit2(1) :JAN8/EAN8をGTINフォーマットで出力
bit3(0) :JAN13/EAN13を通常フォーマットで出力
bit3(1) :JAN13/EAN13をGTINフォーマットで出力
設定の優先順位は、bit1>bit0

*8 UPCEaddon出力フォーマット　00h :先頭の0を付加して出力
04h :先頭の0を削除して出力

*9 RSS-14出力フォーマット　bit0(0) :標準出力
bit0(1) :先頭の"01"を出力しない

*10 RSS Limited出力フォーマット　bit0(0) :標準出力
bit0(1) :先頭の"01"を出力しない

*11 RSS-14 Stacked 出力フォーマット　bit0(0) :標準出力
bit0(1) :先頭の"01"を出力しない

*12 Industrial 2of5とIATAを別々に設定することはできません。
両方を指定した場合は、Industrial 2of5の設定が反映されます。

*13 RSS-14とRSS-14 Stackedを別々に設定することはできません。
両方を指定した場合は、RSS-14の設定が反映されます。

*14 RSS ExpandedとRSS Expanded Stackedを別々に設定することはできません。
両方を指定した場合は、RSS Expandedの設定が反映されます。

項目	内容	初期値	参照	設定
読み取り方式の設定 (1 バイト)	b7 b0 00h:単発読み 01h:連続読み(ﾄﾘｶﾞｷｰ有り)	連続読み (01h)	○	○
ブザー制御の設定 (1 バイト)	b7 b0 00h:ブザー制御なし 01h:ブザー制御あり	ブザーあり (01h)	○	○
LED/Vibrator 制御の設定 (1 バイト)	b7 b0 00h:LED制御無し・Vibrator制御無し 01h:LED制御有り・Vibrator制御無し 02h:LED制御有り(ｴﾗｰ除く)・Vibrator制御無し 10h:LED制御無し・Vibrator制御有り 11h:LED制御有り・Vibrator制御有り 12h:LED制御有り(ｴﾗｰ除く)・Vibrator制御有り	LED あり Vib あり (11h)	○	○
出力バッファの参照 (1 バイト)	b7 b0 00h:OBRバッファに出力 01h:(予約) 02h:KEYバッファに出力(※1)	OBR バッファ (00h)	○	×
終了コードの設定 (バーコードの最後尾に付加するコード) (1 バイト)	b7 b0 00h:CR 01h:LF 02h:CR+LF	CR のみ (00h)	○	○
読み取り動作の設定 (1 バイト)	b7 b0 00h:通常読み 01h:段数読み	通常読み (00h)	○	○

参照

OBR_moderd 関数、OBR_modewt 関数

2.2.32 TIM_TBL 構造体

シリアル通信制御の監視タイムアウト値を格納します。

```
typedef struct {
   H      cs;
   H      dr;
   H      cd;
} TIM_TBL;
```

メンバ

cs
CS タイムアウト監視値を 0～32767 の範囲で格納します。単位は 7.8ms です。

dr
DR タイムアウト監視値を 0～32767 の範囲で格納します。単位は 7.8ms です。

cd
CD タイムアウト監視値を 0～32767 の範囲で格納します。単位は 7.8ms です。

参照

c_open 関数

2.2.33 DEL_TBL 構造体

シリアル通信制御のデリートコード設定を格納します。

```
typedef struct {
   B      del_n;
   UB     del_c[4];
} DEL_TBL;
```

メンバ

del_n
デリートコード数を 0～4 の範囲で格納します。

del_c
デリートコードを 0x00～0xff の範囲で格納します。

参照

c_open 関数

2.2.34 COM_STS 構造体

シリアル通信制御における受信バッファのステータスを格納します。

```
typedef struct{
   H      char_no;
   H      rest_no;
   UB     char_cod;
} COM_STS;
```

メンバ

char_no
受信文字数を格納します。

rest_no
受信可能残り文字数を格納します。

char_cod
先頭文字コードを格納します。

2.2.35 State_DCB 構造体

IrDA 制御の通信パラメータを格納します。

```
struct State_DCB {
   H      station;
   H      Wire;
   H      DataWaitTime;
   H      LineWaitTime;
   H      baudRate;
   H      DataLen;
   H      StopBit;
   H      ParityBit;
};
```

メンバ

station

局に次の値を格納します。

PRIMARY	:自局を 1 次局に設定
SECONDARY	:自局を 2 次局に設定

Wire

次の値を格納します。

WIRE3RAW	:3-wire raw に設定
WIRE3	:3-wire に設定
WIRE9	:9-wire に設定
WIRELPT	:LPT(3-wire raw)に設定

DataWaitTime

データ待ち時間に次の値を格納します。

1-600	:秒単位にデータ読み込み／書き込み待ち時間を設定
THROUGH	:データ読み込み／書き込み待ちを行なわない
FOREVER	:タイマ指定なしでデータ読み込み／書き込み待ちを行なう

LineWaitTime

DR/CS/CD 信号待ち時間に次の値を格納します。

1-600	:秒単位に DR/CS/CD 信号待ち時間を設定
THROUGH	:DR/CS/CD 信号のチェックを行なわない
FOREVER	:タイマ指定なしで DR/CS/CD 信号待ちを行なう

baudRate

RS232C の通信速度に次の値を格納します。

BPS_12	:RS232C の通信速度を 1200bps に設定
BPS_24	:RS232C の通信速度を 2400bps に設定
BPS_48	:RS232C の通信速度を 4800bps に設定
BPS_96	:RS232C の通信速度を 9600bps に設定
BPS_192	:RS232C の通信速度を 19200bps に設定
BPS_384	:RS232C の通信速度を 38400bps に設定
BPS_576	:RS232C の通信速度を 57600bps に設定
BPS_1152	:RS232C の通信速度を 115200bps に設定

DataLen

RS232C のデータ長に次の値を格納します。

LEN_7B	:RS232C のデータ長を 7bit に設定
LEN_8B	:RS232C のデータ長を 8bit に設定

StopBit

RS232C のストップビットに次の値を格納します。

STOP_1B	:RS232C のｽﾄｯﾌﾟﾋﾞｯﾄを 1bit に設定
STOP_2B	:RS232C のｽﾄｯﾌﾟﾋﾞｯﾄを 2bit に設定

ParityBit

RS232C のパリティビットに次の値を格納します。

PRI_ODD	:RS232C のパリティﾋﾞｯﾄを奇数パリティに設定
PRI_EVN	:RS232C のパリティﾋﾞｯﾄを偶数パリティに設定
PRI_NON	:RS232C のパリティﾋﾞｯﾄをパリティなしに設定

参照

Ir_State_Set 関数

2.2.36 SetPortConfig_DCB 構造体

IrDA 制御における自局能力の設定を格納します。

```
struct SetPortConfig_DCB {
   UB     irBaud;
   UB     MaxTurnTime;
   UB     FrameSize;
   UB     WindowSize;
   UB     BofCount;
   UB     MinTurnTime;
   UB     DiscTime;
};
```

メンバ

irBaud

ボーレイトに次の値を OR して格納します。

IRBPS_24	:IR 接続速度を 2400bps に設定可能
IRBPS_96	:IR 接続速度を 9600bps に設定可能
IRBPS_192	:IR 接続速度を 19200bps に設定可能
IRBPS_384	:IR 接続速度を 38400bps に設定可能
IRBPS_576	:IR 接続速度を 57600bps に設定可能
IRBPS_1152	:IR 接続速度を 115200bps に設定可能

MaxTurnTime

最大ターンアラウンドタイムを格納します。

TURN_500MS	:最大ターンアラウンドタイムを 500ms に設定

FrameSize

フレームサイズを格納します。

FRAME_1024B	:フレームサイズを 1024 バイトに設定

WindowSize

ウインドウサイズを格納します。

WINDOW_4	:ウィンドウサイズを 4 フレームウィンドウに設定

BofCount

BOF 数を格納します。IR ボーレートによって比例して増減します。BOF 数は 115.2Kbps の場合です。

BOF_48	:BOF を 48 個追加
BOF_24	:BOF を 24 個追加
BOF_12	:BOF を 12 個追加
BOF_5	:BOF を 5 個追加
BOF_3	:BOF を 3 個追加
BOF_2	:BOF を 2 個追加
BOF_1	:BOF を 1 個追加
BOF_0	:BOF を 0 個追加

MinTurnTime

最小ターンアラウンドタイムを格納します。

TURN_5MS	:最小ターンアラウンドタイムを 5ms に設定
TURN_1MS	:最小ターンアラウンドタイムを 1ms に設定

DiscTime

リンク開放時間に次の値を OR して格納します。

RELEASE_3S	:リンクを開放する時間を 3s に設定可能
RELEASE_8S	:リンクを開放する時間を 8s に設定可能
RELEASE_12S	:リンクを開放する時間を 12s に設定可能
RELEASE_16S	:リンクを開放する時間を 16s に設定可能
RELEASE_20S	:リンクを開放する時間を 20s に設定可能
RELEASE_25S	:リンクを開放する時間を 25s に設定可能
RELEASE_30S	:リンクを開放する時間を 30s に設定可能
RELEASE_40S	:リンクを開放する時間を 40s に設定可能

参照

Ir_SetPortConfig 関数

2.2.37 BT_LOCALINFO 構造体

Bluetooth 制御において本体のデバイス情報を格納します。

```
typedef struct {
   B      LocalAddr[18];
   B      LocalName[82];
   H      LocalClass;
} BT_LOCALINFO;
```

メンバ

LocalAddr
本体の Bluetooth アドレスを ASCII の 16 進数で格納します。アドレスの書式は"XX:XX:XX:XX:XX:XX"です。

LocalName
本体の Bluetooth デバイス名を最長 81 文字で格納します。

LocalClass
本体の Bluetooth デバイスクラスを格納します。
iOS に HID で接続する場合には、LocalClass に 0x0540 を設定します。それ以外の場合は、0 を設定してください。

参照

BT_GetLocalInfo 関数、BT_SetLocalInfo 関数

2.2.38 BT_DEVINFO 構造体

Bluetooth 制御において他の Bluetooth 機器のデバイス情報を格納します。

```
typedef struct {
   UW     ErrFlag;
   B      DevAddr[18];
   B      DevName[82];
   H      DevClass;
} BT_DEVINFO;
```

メンバ

ErrFlag
Bluetooth デバイス名が取得できなかった場合にエラーコードを格納します。

DevAddr
本体の Bluetooth アドレスを ASCII の 16 進数で格納します。アドレスの書式は"XX:XX:XX:XX:XX:XX"です。

DevName
本体の Bluetooth デバイス名を最長 81 文字で格納します。

DevClass
本体の Bluetooth デバイスクラスを格納します。

参照

BT_GetDevInfo 関数、BT_GetDevName 関数、BT_SaveDevInfo 関数、BT_LoadDevInfo 関数

2.2.39 CU_RSPRM 構造体

FLINK プロトコルにおける通信パラメータを格納します。

```
typedef struct    {
   H   speed;
   H   length;
   H   parity;
   H   stop_bit;
} CU_RSPRM;
```

メンバ

speed

転送速度を格納します。

CU_B1200	:ボーレート 1200bps の指定
CU_B2400	:ボーレート 2400bps の指定
CU_B4800	:ボーレート 4800bps の指定
CU_B9600	:ボーレート 9600bps の指定
CU_B19K	:ボーレート 19.2Kbps の指定
CU_B38K	:ボーレート 38.4Kbps の指定
CU_B57K	:ボーレート 57.6Kbps の指定
CU_B115K	:ボーレート 115Kbps の指定

length

データ長を格納します。

CU_CHAR7	:データ長 7 ビットの指定
CU_CHAR8	:データ長 8 ビットの指定

parity

パリティに次の値を格納します。

CU_PARI_NON	:パリティなしの設定
CU_PARI_ODD	:奇数パリティの指定
CU_PARI_EVN	:偶数パリティの指定

stop_bit

ストップビットに次の値を格納します。

CU_STOP1	:1 ビットパリティ指定
CU_STOP2	:2 ビットパリティ指定

参照

cu_open 関数

2.2.40 CU_LANPRM 構造体

LAN 通信におけるパラメータを格納します。

```
typedef struct    {
   B    ipaddr[4];
   H    port;
} CU_LANPRM;
```

メンバ

ipaddr
IP アドレスを格納します。

port
ポート番号を格納します。

参照

cu_open 関数

2.2.41 CU_GRAPHSET 構造体

FLINK プロトコルにおける進捗グラフ表示情報を格納します。

```
typedef struct {
   H  graphMode;
   H  graphPos;
   H  graphCol;
   H  graphName;
   H  graphLine;
} CU_GRAPHSET;
```

メンバ

graphMode
グラフ表示モードに次の値を格納します。
CU_GRAPH_ON_1　　:転送全体を 100%として表示
CU_GRAPH_ON_2　　:1 ファイルを 100%として表示
CU_GRAPH_OFF　　:表示しない

graphPos
ファイル名を表示する行番号を 0～11 の範囲で格納します。CU_GRAPH_OFF 時は参照しません。

graphCol
ファイル名を表示するカラム番号を 0～25 の範囲で格納します。CU_GRAPH_OFF 時は参照しません。

graphName
ファイル名の表示方法に次の値を格納します。CU_GRAPH_OFF 時は参照しません。
CU_GRAPH_NM_PATH　:全パス表示
CU_GRAPH_NM_FILE　:ファイル名のみ

graphLine
ファイル名を表示する領域の行数を 1～12 の範囲で格納します。CU_GRAPH_OFF 時は参照しません。

参照

cu_fileSend 関数、cu_fileAdd 関数、cu_fileRecv 関数、cu_idle 関数、
cu_cmdRecv 関数、cu_fchklog_Create 関数、cu_fchklog_Check 関数

2.2.42 CU_ERRINFO 構造体

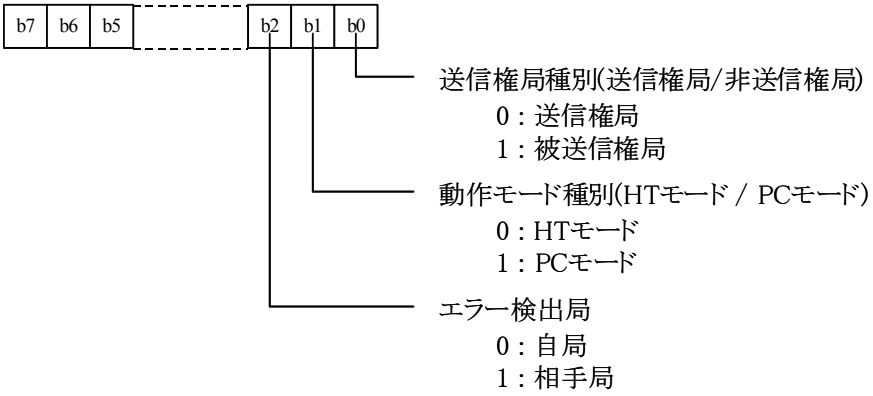
FLINK プロトコルにおけるエラー情報を格納します。

```
typedef struct {
   UB  kind;
   UB  command;
   UB  category;
   UB  detail;
   UW  biosStat;
} CU_ERRINFO;
```

メンバ

kind

エラー種別を次の値から格納します。



command

ファイル名を表示する行番号を 0～11 の範囲で格納します。CU_GRAPH_OFF 時は参照しません。

CU_CMD_NON	:該当コマンドなし
CU_CMD_FSEND_TINFO	:ファイル転送情報コマンド
CU_CMD_FSEND_FINFO	:ファイル情報コマンド
CU_CMD_FRECV_TREQ	:ファイル受信要求コマンド
CU_CMD_FADD	:ファイル追加コマンド
CU_CMD_FDATA	:ファイルデータコマンド
CU_CMD_FDEL	:ファイル削除コマンド
CU_CMD_FMOV	:ファイル移動コマンド
CU_CMD_MAKEDIR	:ディレクトリ作成コマンド
CU_CMD_TIME_SET	:日付時刻設定コマンド
CU_CMD_TIME_GET	:日付時刻取得コマンド
CU_CMD_DISP	:メッセージ表示コマンド
CU_CMD_BEEP	:ブザー鳴動コマンド
CU_CMD_FINFO_GET	:ファイル情報取得コマンド
CU_CMD_FINFO_SET	:ファイル情報設定コマンド
CU_CMD_DINFO_GET	:ディスク情報取得コマンド
CU_CMD_SYS_GET	:システム情報取得コマンド
CU_CMD_IDLE	:IDLE 通知コマンド
CU_CMD_END	:終了指示コマンド

category, detail

カテゴリと詳細エラーコードでエラー状態を判断します。

値		意味	
カテゴリ	詳細		
正常終了状態			
00	00	正常終了	
DC～F5	00	フォーマット指示コマンド(A～Z)	
F6	00	電源 OFF 終了通知	
F7	00	リセット指定終了通知	
F8	00	中断キーによる終了通知	
F9～FF	－	予約領域	
プロトコルエラー			
01	00	受信フレームファンクションコード未定義エラー	
	01	受信フレームサブファンクションコード未定義エラー	
	03	受信フレームチェックサムエラー	
	04	シーケンスエラー	
	05	シーケンス番号エラー	
	07	受信フレーム内情報パラメータエラー	
	08	受信タイムアウト	
	10	コマンドレングスエラー	
ファイルエラー[プロトコル論理]			
04	00	リードオンリィファイルアクセスエラー	
ユーティリティエラー			
10	00	回線オープンエラー	(9) 回線がオープンされていない ・オープン時にエラーが発生していないか確認
	01	使用関数フェーズエラー	(9) 関数の使い方に誤りがある ・動作モード/信権局モードを確認
	02	使用関数パラメータエラー	(9) 関数パラメータに誤りがある ・指定パラメータを確認
	03	指定ファイル未検出エラー	(9) 指定されたファイルが存在しない ・指定ファイルを確認
	04	相手局未検出	(9) セッション確立待ちタイムアウト ・通信設定、回線経路を確認
	05	システム日付設定エラー	・指定日付を確認
	06	システム時刻設定エラー	・指定時刻を確認
	07	タイマー使用エラー	(9) タイマーが登録できなかった ・アプリケーションで使用しているタイマ数を確認
	08	CPU クロック切り替えエラー	・CPU 切り替え禁止状態でないか確認
	09	致命的エラー	(9) IrDA、通信関数からのエラー ・ローバッテリの発生等が考えられる
	0A	通信中回線断エラー	(9) 通信中に回線が切断された ・回線経路を確認
	0B	ドライブ容量不足	・指定ドライブの容量が足りない

ファイルエラー[ファイル関数]			
11	00	クリエートエラー	
	01	オープンエラー	
	02	リードエラー	
	03	ライトエラー	
	04	シークエラー	
	05	ファイル削除エラー	
	06	ディレクトリ削除エラー	
	07	ファイル名変更移動エラー	
	08	タイムスタンプ設定エラー	
	09	タイムスタンプ取得エラー	
	0A	ファイル属性設定エラー	
	0B	ファイル属性取得エラー	
	0C	ディレクトリ作成エラー	
	0D	ファイルサイズ変更エラー	
システムメニュー通信エラー			
20	00	フォーマット実行エラー	(9) フォーマット中にエラー発生 ・再フォーマットする
	01	環境設定ファイル未存在エラー	・CONFIG.HTS ファイルが存在しない
	02	環境設定ファイル更新エラー	(9) CONFIG.HTS 異常 ・ファイルレイアウトを確認
	03	相手局不正	(9) 想定している相手局ではない ・相手局を確認
	04	指定ドライブなし	・子機作成時、送信側指定ドライバが受信側に存在しない
システム異常エラー			
0F	0x	FTP 部内部エラー	
	1x	通信ユーティリティ内部エラー	

biosStat

システム領域エラーエリア（comNo が COM0 時は IrDA 部関数、COM1 時は通信関数のエラーを格納します）

参照

cu_readErrStat 関数

2.2.43 CU_DATETIME 構造体

FLINK プロトコルにおいて日付時刻情報を格納します。

```
typedef struct {
   UB  day;
   UB  month;
   UH  year;
   UB  sec;
   UB  min;
   UB  hour;
} CU_DATETIME;
```

メンバ

day
日を 1～31 の範囲で格納します。

month
月を 1～12 の範囲で格納します。

year
年を 1980～2079 の範囲で格納します。

sec
秒を 0～59 の範囲で格納します。

min
分を 0～59 の範囲で格納します。

hour
時を 0～23 の範囲で格納します。

参照

cu_makeDir 関数

2.2.44 CU_FINFO 構造体

FLINK プロトコルにおいてファイル情報を格納します。

```
typedef struct {
   B               name[256];
   CU_DATETIME     datetime;
   W               size;
   B               atr;
} CU_FINFO;
```

メンバ

name
検索したファイル名を格納します。

datetime
ファイルの日付時刻情報を格納します。

size
ファイルのサイズを格納します。

atr
ファイルの属性を次の値で格納します。

_A_NORMAL	:通常ファイル(読み書き可能)
_A_HIDDEN	:不可視ファイル
_A_RDONLY	:読み出し専用ファイル
_A_SYSTEM	:システムファイル
_A_SUBDIR	:ディレクトリ
_A_ARCH	:アーカイブ

参照

cu_getFileInfo 関数、cu_setFileInfo 関数

2.2.45 CU_DINFO 構造体

FLINK プロトコルにおいてディスク情報を格納します。

```
typedef struct {
   UW     size;
   UW     freex;
   UB     status;
} CU_DINFO;
```

メンバ

size
ディスク容量を格納します。

freex
ディスク空き容量を格納します。

status
ディスク状態を格納します。

CU_DINFO_NORMAL	:ディスクあり（フォーマット済み）
CU_DINFO_NOFMT	:ディスクあり（未フォーマット）
CU_DINFO_NODISK	:ディスクなし

参照

cu_getDiskInfo 関数

2.2.46 CU_SYSINFO 構造体

FLINK プロトコルにおいて相手局のシステム情報を格納します。

```
typedef struct {
   UH  id;
   UB  ftpver;
   UB  code[3];
   UB  model;
} CU_SYSINFO;
```

メンバ

id
セッション ID を格納します。PC との接続以外、この値は 0 固定です。

ftpver
FTP プロトコルのバージョンを格納します。

code
機種コードを格納します。

"710"	:ハンディーターミナル
その他	:PC

model
モデル情報を格納します。この値は 0 固定です。

参照

cu_getSysInfo 関数

2.2.47 T_RFLG 構造体

イベントフラグの情報を格納します。

```
typedef struct t_rflg  {
   ID         wtskid;
   FLGPTN     flgptn;
} T_RFLG;
```

メンバ

wtskid

イベントフラグの待ちキューにタスクが設定されているか否かを格納します。

TSK_NONE	:待ちキューにタスクは設定されていない
その他	:待ちキューの先頭に設定されているタスクの ID

flgptn

イベントフラグの現在のビット・パターンを格納します。

参照

ref_flg 関数

2.2.48 sockaddr_in 構造体

ソケット関数で使用する IPv4 アドレス情報を格納します。

```
struct sockaddr_in  {
  short               sin_family;
  unsigned short      sin_port;
  struct in_addr      sin_addr;
  char                sin_zero[8];
};
```

メンバ

sin_family
アドレスファミリーを格納します。通常は AF_INET を格納します。

sin_port
ネットワークバイトオーダでのポート番号を格納します。

sin_addr
IPv4 アドレス構造体(in_addr)を格納します。

sin_zero
使用しません。

参照

net_bind 関数、net_connect 関数、net_sendto 関数、net_recvfrom 関数

2.2.49 in_addr 構造体

ソケット関数で使用する IPv4 アドレスを格納します。

```
typedef sunsigned long in_addr_t;

struct in_addr {
  in_addr_t       s_addr;
};
```

メンバ

s_addr
ネットワークバイトオーダでの IPv4 アドレスを格納します。

参照

net_getsockopt 関数、net_inet_aton 関数、net_inet_ntoa 関数、sockaddr_in 構造体

2.2.50 sockaddr 構造体

ソケット関数で使用するアドレス情報を格納します。

```
struct sockaddr {
  unsigned short  sa_family;
  char            sa_data[14];
};
```

メンバ

sa_family
アドレスファミリーを格納します。通常は AF_INET を格納します。

sa_data
ソケットデータを格納します。

参照

net_bind 関数、net_connect 関数、net_accept 関数、net_sendto 関数、net_recvfrom 関数

2.2.51 timeval 構造体

ソケット関数で使用する時間情報を格納します。

```
struct timeval {
  long    tv_sec;
  long    tv_usec;
};
```

メンバ

tv_sec
秒単位の値を格納します。

tv_usec
マイクロ秒単位の値を格納します。

参照

net_select 関数、net_getsockopt 関数、net_setsockopt 関数

2.2.52 fd_set 構造体

ソケット関数で使用するソケット情報を格納するために使用します。

```
typedef struct {
  unsigned long   fds_bits[1];
};
```

メンバ

fds_bits[1]
ソケット情報

参照

net_select 関数

2.2.53 T_MYIP_PARAM 構造体

IP アドレスに関する情報を格納するために使用します。

```
typedef struct T_MYIP_PARAM{
  unsigned char   state;
  unsigned long   ipaddr;
  unsigned long   subnet;
  unsigned long   geteway;
  unsigned long   dns1;
  unsigned long   dns2;
};
```

メンバ

state
I/F の状態
0　:クローズ
1　:オープン

ipaddr
自 IP アドレス

subnet
サブネットマスク

geteway
ゲートウェイアドレス

dns1
DNS1 サーバアドレス

dns2
DNS2 サーバアドレス

参照

net_get_myip_info 関数

2.2.54 T_IPV4EP 構造体

アドレス解決した IP アドレスの情報を格納するために使用します。

```
typedef struct {
  UW      ipaddr;
  UH      portno;
} T_IPV4EP;
```

メンバ

ipaddr
IPv4 アドレス

portno
ポート番号

参照

net_get_ipaddr 関数

3. C 言語標準関数

下表は、"RXファミリ用C言語標準関数"のうち、DT-970で使用可能な関数の一覧です。それぞれの関数仕様は、" RENESAS ユーザーズマニュアル RX コーディング編"を参照してください。

※ 低水準関数(open、close、read、write、lseek、sbrk)はハードウェアに依存するため、デバイス制御ライブラリとして実装しています。本章に記載の C 言語標準関数のうち、ファイルやメモリに依存するものは、内部で低水準関数を呼び出しています。

判定／変換

関数	機能概要
isalnum	英字・10 進数字の判定
isalpha	英字の判定
iscntrl	制御文字の判定
isdigit	10 進数字の判定
isgraph	空白を除く印字文字の判定
islower	英小文字の判定
isprint	空白を含む印字文字の判定
ispunct	特殊文字の判定
isspace	空白類文字の判定
isupper	英大文字の判定
isxdigit	16 進数字の判定
tolower	英大文字を英小文字に変換
toupper	英小文字を英大文字に変換

数値計算

関数	機能概要
acos	浮動小数点数の逆余弦
asin	浮動小数点数の逆正弦
atan	浮動小数点数の逆正接
atan2	浮動小数点どうしを除算した結果の逆正接
cos	浮動小数点数のラディアン値の余弦
sin	浮動小数点数のラディアン値の正弦
tan	浮動小数点数のラディアン値の正接
cosh	浮動小数点数の双曲線余弦
sinh	浮動小数点数の双曲線正弦
tanh	浮動小数点数の双曲線正接
exp	浮動小数点数の指数
frexp	浮動小数点数を(0.5,1.0)の値として 2 のべき乗の積に分解
ldexp	浮動小数点数と 2 のべき乗の乗算
log	浮動小数点数との自然対数
log10	浮動小数点数の 10 を底とする対数
modf	浮動小数点数を整数部分と小数部分に分解
pow	浮動小数点数のべき乗
sqrt	浮動小数点数の正の平方根
ceil	浮動小数点数の小数点以下を切り上げた整数値
fabs	浮動小数点数の絶対値
floor	浮動小数点数の小数点以下を切り捨てた整数値
fmod	浮動小数点数どうしを除算した結果の余り

関数間の制御移動

関数	機能概要
setjmp	現在実行中の関数の実行環境を、指定した記憶域に待避
longjmp	setjmp で退避した関数の実行環境を回復し、setjmp 関数を呼び出した位置に制御を移動

可変個引数の参照

マクロ	機能概要
va_start	可変個の引数を参照するための初期値を設定
va_arg	可変個の引数を持つ関数に対し、現在参照中引数の次の引数を参照
va_end	可変個の引数を持つ関数の引数への参照を終了

ストリーム入出力

関数	機能概要
fclose	ファイルのクローズ
fopen	ファイルのオープン
freopen	現在オープンしているファイルをクローズし、新たに指定ファイル名のファイルをオープン
sprintf	データを書式に従って変換し、指定領域に出力
sscanf	指定領域からデータを入力し、書式に従って変換
fread	ファイルから指定領域にデータを入力
fwrite	指定領域からファイルにデータを出力
fseek	ファイルの現在の読み書き位置を移動
ftell	ファイルの現在の読み書き位置を取得
rewind	ファイルの現在の読み書き位置をファイル先頭に移動
ferror	ファイルがエラー状態であるかを判定
clearerr	ファイルのエラー状態をクリア

※ データファイルへの入出力のみサポートします。標準入出力ファイル（コンソール、プリンタ、デイスクファイル等）についての入出力はサポートしません。

C プログラム標準処理

関数	機能概要
atof	数を表現する文字列を double 型の浮動小数点数に変換
atoi	10 進数を表現する文字列を int 型の整数値に変換
atoll	10 進数を表現する文字列を long 型の整数値に変換
strtod	数を表現する文字列を double 型の浮動小数点数に変換
strtol	数を表現する文字列を long 型の整数値に変換
srand	rand 関数で生成する疑似乱数列の初期値を設定
calloc	記憶域を確保し、確保したすべての領域を 0 クリア
free	指定した記憶域を解放
malloc	記憶域を確保
realloc	記憶域の大きさを指定した大きさに変更
abort	プログラムの異常終了
exit	プログラムの正常終了
bsearch	二分割検索
qsort	ソートの実行
abs	int 型整数の絶対値
div	int 型整数の除算の商と余り
labs	long 型整数の絶対値
ldiv	long 型整数の除算の商と余り

文字配列操作

関数	機能概要
memcpy	複写元の記憶域の内容を、指定サイズ分複写先の記憶域に複写
strcpy	複写元の文字列を複写先の記憶域に NULL も含めて複写
strncpy	複写元の文字列を指定文字数分、複写先の記憶域に複写
strcat	文字列の後に文字列を連結
memcmp	指定した 2 つの記憶域の比較
strcmp	指定した 2 つの文字列の比較
strncmp	指定した 2 つの文字列を、指定文字数分まで比較
memchr	指定記憶域で、指定文字が最初に現れる位置を検索
strchr	指定文字列で、指定文字が最初に現れる位置を検索
strcspn	指定文字列を先頭から調べ、別の指定文字列以外の文字が先頭から何文字続くかを取得
strpbrk	指定文字列で、別の指定文字列が最初に現れる位置を検索
strrchr	指定文字列で、指定文字が最後に現れる位置を検索
strspn	指定文字列を先頭から調べ、別の指定文字列が先頭から何文字続くかを取得
strstr	指定文字列で、別の指定文字列が最初に現れる位置を検索
memset	指定記憶域の先頭から指定文字を指定した文字数分、設定
strerror	エラーメッセージを設定
strlen	文字列の長さを取得

4. システムデータ管理

ここでは、DT-970 のシステムデータ管理関数について説明します。

4.1 関数リファレンス

DT-970 では、次の関数を使用することで、アプリケーションからシステムデータの取得と設定が可能です。

関数	機能概要
dat_system	システムデータの設定/取得
dat_OSVer_Read	OS バージョンの取得
dat_dealer_chk	代理店 ID のチェック
dat_mem_size	メモリ領域の空きサイズの取得
dat_get_device_id	シリアル番号の取得

4.1.1 dat_system

電源、KEY、OBR、表示、通信、タイマ等に関するシステムデータの取得と設定を行ないます。

```
ER dat_system(
   FN      fnc,
   ID      sys_id,
   VP      *sys_dt
);
```

パラメータ

fnc

システムデータを取得するか、設定するかを指定します。

SYSD_FNC_READ	:システムデータを取得します。
SYSD_FNC_WRITE	:システムデータを設定します。

sys_id

取得／設定対象のシステムデータを識別する ID を指定します。

SYSD_PWR	:電源	
SYSD_KEY	:KEY	
SYSD_OBR	:OBR	
SYSD_DSP	:表示	フォントモード、言語
SYSD_DSP2	:〃	フォント修飾、コントラスト
SYSD_DSP3	:〃	行間挿入、タスクバー表示
SYSD_COM	:通信	（共通）
SYSD_COM0	:IrDA インタフェース	
SYSD_TIM	:タイマ	
SYSD_SYS	:システム	機器情報
SYSD_SYS2	:〃	代理店 ID、パッチバージョン
SYSD_PRO	:プロトコル	
SYSD_BAT	:電池	
SYSD_BTH	:Bluetooth	
SYSD_LAN	:LAN	
SYSD_HIP	:IP	端末 IP
SYSD_CIP	:〃	クレードル IP
SYSD_USB	:USB	
SYSD_HST	:接続先	

sys_dt

取得／設定対象のシステムデータ構造体アドレスを指定します。

sys_id の指定によって、*sys_dt* パラメータに指定する構造体は次のようになります。

SYSD_PWR	:DAT_PWR_STR 構造体
SYSD_KEY	:DAT_KEY_STR 構造体
SYSD_OBR	:DAT_OBR_STR 構造体
SYSD_DSP	:DAT_DSP_STR 構造体
SYSD_DSP2	:DAT_DSP_STR2 構造体
SYSD_DSP3	:DAT_DSP_STR3 構造体
SYSD_COM	:DAT_COMINF_STR 構造体
SYSD_COM0	:DAT_COM_STR 構造体
SYSD_TIM	:DAT_TIM_STR 構造体
SYSD_SYS	:DAT_SYS_STR 構造体
SYSD_SYS2	:DAT_SYS_STR2 構造体
SYSD_PRO	:DAT_PRO_STR 構造体
SYSD_BAT	:DAT_BAT_STR 構造体
SYSD_BTH	:DAT_BTH_STR 構造体
SYSD_LAN	:DAT_LAN_STR 構造体
SYSD_HIP	:DAT_HIP_STR 構造体
SYSD_CIP	:DAT_CIP_STR 構造体
SYSD_USB	:DAT_USB_STR 構造体
SYSD_HST	:DAT_HST_STR 構造体

戻り値

関数が成功すると E_OK が返ります。失敗すると次のエラーが返ります。

E_PRM	:パラメータエラー
E_NG	:登録データエラー

解説

システム関連データの BIOS バージョン・パッチバージョンと機器種別を設定することはできません。

fnc パラメータに上記以外の値を指定した場合、dat_system 関数はエラーを返します。

sys_id パラメータに上記以外の値を指定した場合、dat_system 関数はエラーを返します。

fnc パラメータに SYSD_FNC_WRITE を指定して関数が失敗した場合は、設定しようとしたすべてのシステムデータの登録を行ないません。

4.1.2 dat_OSVer_Read

OS バージョンを取得します。

```
void dat_OSVer_Read(
  B     *rd_buf
);
```

パラメータ

rd_buf
OS バージョンを取得するバッファのアドレスを指定します。
16 バイト以上の領域を指定してください。

戻り値

ありません。

解説

取得する OS バージョンのフォーマットを以下に示します。

*.***(SP)**.**.**(SP)(SP)

1～6 バイト	バージョン NO.
7 バイト	スペース
8～9 バイト	年
10 バイト	.
11～12 バイト	月
13 バイト	.
14～15 バイト	日
16 バイト	スペース

4.1.3 dat_dealer_chk

代理店 ID のチェックを行ないます。アプリケーションの不正コピー防止などに使用します。

```
ER dat_dealer_chk(
   UB     *dealer_no
);
```

パラメータ

dealer_no
チェック対象代理店 ID を格納した領域のアドレスを指定します。

戻り値

代理店 ID が一致すると E_OK が返ります。
一致しないと E_NG が返ります。

4.1.4 dat_mem_size

メモリ領域の未使用領域サイズを取得します。

```
UW dat_mem_size();
```

パラメータ

ありません。

戻り値

メモリ領域の未使用領域サイズが、バイト単位で返ります。

4.1.5 dat_get_device_id

本体に登録されているシリアル番号(16 バイト)を取得します。

```
ER dat_get_device_id(
   UB     *pSerial
);
```

パラメータ

pSerial
シリアル番号を取得するバッファのアドレスを指定します。

戻り値

関数が成功すると E_OK が返ります。失敗すると次のエラーが返ります。
E_PRM　:パラメータエラー
E_NG　:異常終了

5. 電源管理

5.1 概要

ここでは、DT-970 の電源管理について説明します。

自動電源 OFF (APO:Auto Power OFF)
APO とは一定時間操作をしていない状態が続いた場合に、自動的に電源を OFF する機能です。
APO 実行までの時間は、1～59 分の範囲(1 分単位)で設定することができます。
APO で自動的に電源を OFF した場合、次回の電源 ON は、システム設定のレジューム ON/OFF の設定にかかわらずレジューム ON 起動となります。

APO イベントの通知を有効に設定している場合は、イベントフラグに FL_LB_INT_LB4 を立て、電源を OFF しません。

pwr_hold_apo 関数を使用して APO を禁止することができます。

通信ユーティリティは自分で APO を禁止しますので、アプリケーションプログラムから禁止にする必要はありません。

自動バックライト OFF (ABO:Auto Backlight OFF)
ABO とは一定時間操作をしていない状態が続いた場合に、自動的にバックライトを OFF する機能です。
ABO 実行までの時間は、10～59 秒の範囲(1 秒単位)で設定できます。
ABO で自動的に OFF したバックライトは、キー入力で再び ON します。

低消費電力制御
key_read、key_string、key_num 関数を実行してシステムがキー待ち状態になった場合、CPU を SLEEP 状態にして消費電力を抑えます。

電源 OFF
pwr_off 関数を使用して、アプリケーションから DT-970 の電源を OFF することができます。

クレードル起動
pwr_IoboxBootMode 関数を使用して、クレードルの接続を検出したときに DT-970 の電源を自動的に ON するように設定できます。

5.2 関数リファレンス

ファンクション詳細を次ページより示します。

関数	機能概要
pwr_hold_apo	APO 禁止の設定
pwr_off	電源の OFF
pwr_IoboxBootMode	クレードルの起動設定

5.2.1 pwr_hold_apo

APO 禁止の設定と解除を行ないます。

```
ER pwr_hold_apo(
   UH     OnOff,
   UW     BitPtrn
);
```

パラメータ

OnOff

APO 禁止の設定、解除を次の値で指定します。

PWR_ON　　:APO 禁止を設定します
PWR_OFF　　:APO 禁止を解除します

BitPtrn

ビットパターンを指定します。FL_INV_APO_USR を指定してください。

戻り値

関数が成功すると E_OK が返ります。失敗すると次のエラーが返ります。

E_PRM　　:パラメータエラー

5.2.2 pwr_off

電源を OFF にします。

```
ER pwr_off(
   UH     OnOff
);
```

パラメータ

OnOff

次回起動時の設定を次の値で指定します。

PWR_ON　　:次回電源 ON 時、レジューム ON モードで起動します。
PWR_OFF　　:次回電源 ON 時、レジューム OFF モードで起動します。

戻り値

関数が成功すると E_OK が返ります。失敗すると次のエラーが返ります。

E_PRM　　:パラメータエラー

5.2.3 pwr_IoboxBootMode

DT-970 をクレードルに載せたときに、自動的に電源を ON するかどうかを設定します。

```
ER pwr_IoboxBootMode(
   UH      OnOff
);
```

パラメータ

OnOff

電源 ON 設定を次の値で指定します。

PWR_ON_CRDL	:クレードルに載せたときに、自動的に電源を ON します。
PWR_OFF_CRDL	:クレードルに載せたときに、自動的に電源を ON しません。
PWR_ON_USBB	:USB-B コネクタに接続したときに、自動的に電源を ON します。
PWR_OFF_USBB	:USB-B コネクタに接続したときに、自動的に電源を ON しません。
PWR_ON	:クレードルに載せたときに、自動的に電源を ON します。 (DT-930 互換用。PWR_ON_CRDL と同じです。)
PWR_OFF	:クレードルに載せたときに、自動的に電源を ON しません。 (DT-930 互換用。PWR_OFF_CRDL と同じです。)

戻り値

関数が成功すると E_OK が返ります。失敗すると次のエラーが返ります。

E_PRM	:パラメータエラー

6. ブザー鳴動

6.1 概要

DT-970 は、次の 3 つの音を鳴動させることができます。

- キークリック音
- エラービープ音
- サウンド音

	キークリック音	エラービープ音	サウンド音
内容	キー押下時に使用します。	入力禁止中のキー押下／エラー発生時等に使用します。	周波数／長さを指定してサウンド音を鳴動します。 サウンド音鳴動前に、鳴動中のブザーを停止します。
周波数	2048Hz	4096Hz	0 128～4096Hz
長さ	50msec	100msec	1～160（×25msec） 0（停止）
その他	システム専用	アプリケーション使用可能	アプリケーション使用可能

エラービープ音は s_beep 関数を、サウンド音は s_sound 関数を使用して、アプリケーションからブザーを鳴動することができます。

6.2 ブザー鳴動の優先順位

ブザー鳴動の要求が同時に発生した場合は、キークリック音 < サウンド音 = エラービープ音の優先順位にしたがいます。

		状態				
		キークリック鳴動中	サウンドウェイト中	サウンドバッファリング中	サウンド鳴動中	エラービープ鳴動中
鳴動要求	キークリック	要求無効	要求無効	要求無効	要求無効	要求無効
	サウンド	鳴動中止要求受入れ	ー	要求サウンドウェイト	要求サウンドバッファリング	要求無効
	エラービープ	鳴動中止要求受入れ	ウェイト中止要求受入れ	バッファリング中止要求受入れ	鳴動中止要求受入れ	要求無効

6.3 関数リファレンス

ファンクション詳細を次ページより示します。

関数	機能概要
s_beep	エラービープ音の発生
s_sound	サウンド音の発生

6.3.1 s_beep

エラービープ音を鳴らします。

```
void s_beep();
```

パラメータ

ありません

戻り値

ありません

解説

ビープ音の音量は、dat_system 関数 DAT_TIM_STR 構造体で設定した値にしたがいます。

6.3.2 s_sound

任意の周波数と音長のサウンド音を鳴らします。

```
ER s_sound(
   UW      freq,
   UW      leng
);
```

パラメータ

freq
サウンド周波数を 0、または 128～4096 Hz の範囲で指定します。

leng
サウンド音長を 0、または 1 ～ 160×25msec の範囲で指定します。
leng に 0 を設定した場合、鳴動中のサウンド音 1 またはキークリック音は停止します。

戻り値

関数が成功すると E_OK が返ります。失敗すると次のエラーが返ります。
E_PRM : パラメータエラー

解説

ビープ音の音量は、dat_system 関数 DAT_TIM_STR 構造体で設定した値に従います。

7. バイブレータ

本体内蔵のバイブレータを振動させることができます。

7.1 関数リファレンス

ファンクション詳細を次ページより示します。

関数	機能概要
pwr_vibrator	バイブレータの動作開始

7.1.1 pwr_vibrator

バイブレータの振動の開始と終了を行ないます。

```
ER pwr_vibrator(
  UH    OnOff
);
```

パラメータ

OnOff

バイブレータ振動の設定を次の値で指定します。

PWR_ON　　：バイブレータ振動開始。

PWR_OFF　　：バイブレータ振動停止。

戻り値

関数が成功すると E_OK が返ります。失敗すると次のエラーが返ります。

E_PRM　　：パラメータエラー

注意

pwr_vibrator 関数でバイブレータ振動開始後、約 4 秒で自動的に停止します。

8. 日時設定

日付の設定と取得

s_dateset 関数と s_dateget 関数を使用して、現在日付の設定と取得を行なうことができます。
日付の設定範囲は 2000 年 1 月 1 日～2100 年 12 月 31 日です。
閏年にのみ 2 月 29 日の設定が可能です。
s_dateset 関数で設定範囲外の日付を設定しようとした場合、関数が失敗し日付の設定を行ないません。

時刻の設定と取得

s_timeset 関数と s_timeget 関数を使用して、現在時刻の設定と取得を行なうことができます。
時刻の設定範囲は 00:00:00～23:59:59 です。
s_timeset 関数で設定範囲外の時刻を設定しようとした場合、関数が失敗し時刻の設定を行ないません。

8.1 関数リファレンス

ファンクション詳細を次ページより示します。

関数	機能概要
s_dateset	日付の設定
s_dateget	日付の取得
s_timeset	時刻の設定
s_timeget	時刻の取得

8.1.1 s_dateset

日付を設定します。

```
ER s_dateset(
   DAY_DAT    *day_dat
);
```

パラメータ

day_dat
日付を格納する DAY_DAT 構造体のアドレスを指定します。

戻り値

関数が成功すると E_OK が返ります。失敗すると、次のエラーが返ります。
E_PRM　　:パラメータエラー

解説

日付の設定範囲は 2000 年 1 月 1 日～2100 年 12 月 31 日です。
閏年にのみ 2 月 29 日の設定が可能です。
設定範囲外の日付を指定した場合、関数が失敗し日付の設定を行ないません。

参照

DAY_DAT 構造体、s_dateget 関数

8.1.2 s_dateget

現在の日付を取得します。

```
ER s_dateget(
   DAY_DAT     *day_dat
);
```

パラメータ

day_dat
日付を格納する DAY_DAT 構造体のアドレスを指定します。

戻り値

関数が成功すると E_OK が返ります。失敗すると、次のエラーが返ります。
E_PRM　　:パラメータエラー

解説

メモリ内の日付データをそのまま取得します。
日付データ内容のチェックは行ないません。

参照

DAY_DAT 構造体、s_dateset 関数

8.1.3 s_timeset

時刻を設定します。

```
ER s_timeset(
   TIM_DAT     *tim_dat
);
```

パラメータ

tim_dat
時刻を格納する TIM_DAT 構造体のアドレスを指定します。

戻り値

関数が成功すると E_OK が返ります。失敗すると、次のエラーが返ります。
E_PRM　　:パラメータエラー

解説

時刻の設定範囲は 00:00:00～23:59:59 です。
設定範囲外の時刻を指定した場合、関数が失敗し時刻の設定を行ないません。

参照

TIM_DAT 構造体、s_timeget 関数

8.1.4 s_timeget

現在の時刻を取得します。

```
ER s_timeget(
   TIM_DAT     *tim_dat
);
```

パラメータ

tim_dat
時刻を格納する TIM_DAT 構造体のアドレスを指定します。

戻り値

関数が成功すると E_OK が返ります。失敗すると、次のエラーが返ります。
E_PRM　　:パラメータエラー

解説

メモリ内の時刻データをそのまま取得します。
時刻データ内容のチェックは行ないません。

参照

TIM_DAT 構造体、s_timeset 関数

9. ファイル管理

9.1 関数リファレンス

DT-970 では、次の関数を使用することで、アプリケーションからディレクトリやファイルに対する処理を行なうことができます。

高水準関数

関数	機能概要	ファイルシステム	
		FAT	DT-700
fil_mkdir	ディレクトリの作成	○	×
fil_rmdir	ディレクトリの削除	○	×
fil_remove	ファイルの削除	○	○
fil_rename	ファイル名の変更/移動	○	○
fil_fstat	ファイルの日時・サイズ・属性の取得	○	○
fil_chsize	ファイルサイズの変更	○	○
fil_getsize	ファイル領域空きサイズの取得	○	○
fil_getsize2	ファイル領域空きサイズの取得（4GB 以上対応版）	○	○
fil_findfirst	ファイルの検索	○	○
fil_findnext	検索ファイル次候補の取得	○	○
fil_filesize	ファイルの個数と総サイズの取得	○	○
fil_filesize2	ファイルの個数と総サイズの取得（4GB 以上対応版）	○	○
fil_filefind	ファイル全パス名の取得	○	○
dat_fdir	ファイル格納情報の取得	×	○
dat_fsize	ファイル空き領域サイズの取得	×	○
dat_fdel	ファイルの削除	×	○
dat_frname	ファイル名の変更	×	○
dat_F_Search	ファイルデータの検索	×	○

※ DT-700 互換ファイルシステムの場合、ドライブおよびディレクトリの概念がありません。ルートディレクトリのみ有効です。

※ DT-700 互換ファイルシステムの場合、ファイル属性がありません。ファイルの検索(fil_findfirst)では、ファイル属性は検索条件対象外です。

低水準インタフェース

ファイルについての処理（状態管理、書込／読込等）、およびメモリ管理を行ないます。

関数	機能概要
open	ファイルのオープン
close	ファイルのクローズ
read	ファイルのリード
write	ファイルのライト
lseek	ファイルリード／ライト位置の設定
sbrk	メモリ領域の割り当て

※ 低水準インタフェース関数と高水準関数を併用した場合、誤動作の原因となります。高水準関数との併用は避けてください。

9.1.1 fil_mkdir

ディレクトリを作成します。

```
ER fil_mkdir(
   const char   *path
);
```

パラメータ

path
作成するディレクトリのフルパスを指定します。

戻り値

関数が成功すると E_OK が返ります。失敗すると、次のエラーが返ります。
E_NG　　:異常終了

解説

本関数は、DT-700 互換ファイルシステムでは使用できません。

MS-DOS の予約デバイス(con、prt 等)に相当するファイル名の制限はありません。
path パラメータに、9 文字以上 11 文字のディレクトリ名を指定した場合は、8.3 形式に変換してディレクトリを作成します。
（例）A:¥12345678901　　→　　A:¥12345678.901

path パラメータに、¥を連続して指定した場合は、次のよう¥と¥の間にスペースを指定したものとしてディレクトリを作成します。
（例）A:¥A¥¥ABC　　→　　A:¥A¥(スペース)¥ABC

先頭コードが 80h 以上のディレクトリ・ファイル名は、作成しないでください。検索系関数で検索ができなくなります。
すでに最大同時オープン数をオープンしている場合、本関数は失敗します。

9.1.2 fil_rmdir

ディレクトリを削除します。

```
ER fil_rmdir(
  const char    *path
 );
```

パラメータ

path
削除するディレクトリのフルパスを指定します。

戻り値

関数が成功すると E_OK が返ります。失敗すると、次のエラーが返ります。
E_NG　　:異常終了

解説

本関数は、DT-700 互換ファイルシステムでは使用できません。
すでに最大同時オープン数をオープンしている場合、本関数は失敗します。

9.1.3 fil_remove

ファイルを削除します。

```
ER fil_remove(
   const char    *pathname
);
```

パラメータ

pathname
削除するファイルのパスを指定します。

戻り値

関数が成功すると E_OK が返ります。失敗すると、次のエラーが返ります。
E_NG　　:異常終了
E_PRM　:パラメータエラー

解説

Pathname パラメータに指定するパスは、ファイルシステムにより指定可能な形式が異なります。

指定形式		ファイルシステム	
		FAT	DT-700 互換
形式 1	ファイル名.拡張子 nnnnnnnn.mmm	指定可能	指定可能
形式 2	ドライブ:パス¥ファイル名.拡張子 d:¥pppppppp¥nnnnnnnn.mmm	指定可能	ルートディレクトリのみ指定可能 （パス名は指定不可）

すでに最大同時オープン数をオープンしている場合、本関数は失敗します。

9.1.4 fil_rename

ファイル名の変更、またはファイルの移動を行ないます。

```
ER fil_rename(
   const char  *oldname,
   const char  *newname
);
```

パラメータ

oldname
既存ファイルのパスを指定します。

newname
新しいファイルのパスを指定します。

戻り値

関数が成功すると E_OK が返ります。失敗すると、次のエラーが返ります。

E_NG	:異常終了
E_PRM	:パラメータエラー

解説

すでに最大同時オープン数をオープンしている場合、本関数は失敗します。
すでにファイル数が上限に達している場合、本関数は失敗します。
異なるドライブ間のファイル移動はできません。本関数は失敗します。

oldname、*newname* パラメータに指定するパスは、ファイルシステムにより指定可能な形式が異なります。詳細は fil_remove 関数を参照してください。

9.1.5 fil_fstat

ファイル番号を指定して、ファイルの属性情報を取得します。

```
ER fil_fstat(
   int         handle,
   FIL_FSTAT  *buffer
);
```

パラメータ

handle
属性を取得するファイル番号を指定します。

buffer
取得する属性を格納する、FIL_FSTAT 構造体のアドレスを指定します。

戻り値

関数が成功すると E_OK が返ります。失敗すると、次のエラーが返ります。

E_NG	:異常終了
E_PRM	:パラメータエラー

解説

handle パラメータに指定するファイル番号は、open 関数を使用して取得します。
DT-700 互換モードの場合、取得するファイル属性は常に 0 です。

参照

FIL_FSTAT 構造体

9.1.6 fil_chsize

ファイルのサイズを変更します。

```
ER fil_chsize(
  B     *path,
  UW    *fsize
);
```

パラメータ

path
変更対象のファイル名を、フルパスで指定します。

fsize
変更するファイルのサイズを、バイト単位で指定します。

戻り値

関数が成功すると E_OK が返ります。失敗すると、次のエラーが返ります。
E_NG　　:異常終了

解説

ファイルサイズを大きくした場合、既存のサイズをこえるファイル領域は NULL で初期化します。

ファイルサイズを小さくした場合、先頭から指定サイズまでをファイルサイズとします。元のサイズまでの領域の内容を参照してはいけません。参照した場合の内容は保証できません。
オープン中のファイルに対して本関数を実行してはいけません。実行した場合、その内容は保証できません。

すでに最大同時オープン数をオープンしている場合、本関数は失敗します。

9.1.7 fil_getsize

ファイル領域の空きサイズ(4GB 以内)を取得します。

```
UW fil_getsize(
  B     *path
);
```

パラメータ

path
空きサイズを取得するドライブ名を指定します。

戻り値

関数が成功するとファイル領域の空き領域のサイズが返ります。
失敗すると、次のエラーが返ります。
E_NG :異常終了
E_PRM :パラメータエラー

解説

ファイル領域の空きサイズが 4GB 以上の場合は、戻り値は 4,294,967,295 となります。

9.1.8 fil_getsize2

ファイル領域の空きサイズを取得します。

```
ER fil_getsize2(
  B     *path,
  UW    *h_size,
  UW    *l_size
);
```

パラメータ

path
空きサイズを取得するドライブ名を指定します。

h_size
空きサイズの上位桁(64bit の上位 32bit)を格納する変数のポインタを指定します。

l_size
空きサイズの下位桁(64bit の下位 32bit)を格納する変数のポインタを指定します。

戻り値

以下の値が返ります。
E_OK :正常終了
E_NG :異常終了
E_PRM :パラメータエラー

9.1.9 fil_findfirst

ファイルの検索条件を指定して、条件に一致するファイル名を取得します。

```
ER fil_findfirst(
   B          *path,
   UH         attr,
   FIND_T     *buffer
);
```

パラメータ

path
検索対象ファイルのパスを、形式 2 で指定します。

attr
検索対象ファイルの属性を、次の値の組み合わせで指定します。

_A_NORMAL	:読み書き可能
_A_VOLID	:ボリューム ID
_A_RDONLY	:読み取り専用※
_A_SUBDIR	:サブディレクトリ
_A_HIDDEN	:隠しファイル
_A_ARCH	:アーカイブ※
_A_SYSTEM	:システムファイル

(9) アーカイブ／読み取り専用属性は、指定の有無にかかわらず常に検索対象となります

buffer
検索結果を格納する、FIND_T 構造体のアドレスを指定します。

戻り値

関数が成功すると E_OK が返ります。失敗すると、次のエラーが返ります。

E_NG	： 異常終了
E_PRM	： パラメータエラー

解説

path パラメータの指定形式については、fil_remove 関数を参照してください。
path パラメータには、ワイルドカードを使用することができます。
次候補を読み出すときは、fil_findnext 関数を使用してください。
ファイルシステムが DT-700 互換モードの場合、*attr* パラメータの指定は無視します。

参照

FIND_T 構造体、fil_findnext 関数

9.1.10 fil_findnext

fil_findfirst 関数で取得したファイルの、次候補を取得します。

```
ER fil_findnext(
   FIND_T     *buffer
);
```

パラメータ

buffer
検索結果を格納する、FIND_T 構造体のアドレスを指定します。

戻り値

関数が成功すると E_OK が返ります。失敗すると、次のエラーが返ります。
E_NG　　:異常終了
E_PRM　　:パラメータエラー

解説

次候補がない場合は、本関数は失敗します。

参照

FIND_T 構造体、fil_findfirst 関数

9.1.11 fil_filesize

ファイルの個数と総サイズ(4GB 以内)を取得します。

```
ER fil_filesize(
   B          *path,
   FIL_SIZE   *buffer,
   UB         find_sw
);
```

パラメータ

path
検索対象ファイルのパスを、形式 2 で指定します。

buffer
取得結果を格納する、FIL_SIZE 構造体のアドレスを指定します。
ファイルの総サイズが 4GB 以上の場合は、*buffer->size* には 4,294,967,295 が返ります。

find_sw
サブディレクトリの検索をする／しないを次の値で指定します。
FIL_SUBDIR_ON　　:サブディレクトリ下を検索します
FIL_SUBDIR_OFF　　:サブディレクトリ下は検索しません

戻り値

関数が成功すると E_OK が返ります。失敗すると、次のエラーが返ります。
E_NG　　:異常終了
E_PRM　　:パラメータエラー

解説

path パラメータの指定形式については、fil_remove 関数を参照してください。
path パラメータには、ワイルドカードを使用することができます。
該当するファイルが存在しない場合、個数 0 個／ファイルサイズ 0 バイトで本関数は正常終了します。

本関数が失敗する要因には次のものがあります。
- ファイル／パス名異常(使用不可コード混在)
- パス長異常(128 文字以上の指定)
- 指定ドライブ未フォーマット
- 指定ドライブ未搭載
- ドライブ指定外(C または F 以降)

参照

FIL_SIZE 構造体

9.1.12 fil_filesize2

ファイルの個数と総サイズを取得します。

```
ER fil_filesize2(
   B          *path,
   FIL_SIZE2  *buffer,
   UB         find_sw
);
```

パラメータ

path
検索対象ファイルのパスを、形式 2 で指定します。

buffer
取得結果を格納する、FIL_SIZE2 構造体のアドレスを指定します。

find_sw
サブディレクトリの検索をする／しないを次の値で指定します。
FIL_SUBDIR_ON　　:サブディレクトリ下を検索します
FIL_SUBDIR_OFF　　:サブディレクトリ下は検索しません

戻り値

関数が成功すると E_OK が返ります。失敗すると、次のエラーが返ります。
E_NG　　:異常終了
E_PRM　　:パラメータエラー

解説

path パラメータの指定形式については、fil_remove 関数を参照してください。
path パラメータには、ワイルドカードを使用することができます。
該当するファイルが存在しない場合、個数 0 個／ファイルサイズ 0 バイトで本関数は正常終了します。

本関数が失敗する要因には次のものがあります。
- ファイル／パス名異常（使用不可コード混在）
- パス長異常（128 文字以上の指定）
- 指定ドライブ未フォーマット
- 指定ドライブ未搭載
- ドライブ指定外（C または F 以降）

参照

FIL_SIZE2 構造体

9.1.13 fil_filefind

ファイルの検索を行ないます。
異なるパスに同じ名前のファイルが複数存在する場合、合致する順番のファイルを取得します。

```
ER fil_filefind(
   B      *path,
   UB     *buffer,
   UB     find_sw,
   UH     seq_no
);
```

パラメータ

path
検索対象ファイルのパスを、形式 2 で指定します。

buffer
検索結果ファイルのパスを、形式 2 で格納する領域のアドレスを指定します。

find_sw
サブディレクトリの検索をする／しないを次の値で指定します。
FIL_SUBDIR_ON　　:サブディレクトリ下を検索します
FIL_SUBDIR_OFF　　:サブディレクトリ下は検索しません

seq_no
同じ名前のファイルが複数存在する場合に、何番目のファイルを取得するかを指定します。

戻り値

関数が成功すると E_OK が返ります。失敗すると、次のエラーが返ります。
E_NG　　:異常終了
E_PRM　　:パラメータエラー

解説

path パラメータの指定形式については、fil_remove 関数を参照してください。
path パラメータには、ワイルドカードを使用することができます。

本関数が失敗する要因には次のものがあります。
- ファイル／パス名異常（使用不可コード混在）
- パス長異常（128 文字以上の指定）
- 指定ドライブ未フォーマット
- 指定ドライブ未搭載
- ドライブ指定外（C または F 以降）
- ファイル未検出
- 指定ファイルが存在しない

9.1.14 dat_fdir

ファイル格納情報を取得します。(DT-700 互換モード専用)

```
ER dat_fdir(
   B          id,
   DIR_TBL    *buf
);
```

パラメータ

id

ファイル管理テーブルの読み込み位置を、次の値で指定します。

DAT_FILE_TOP　　:ファイル管理テーブルの先頭から読み込みます
DAT_FILE_NEXT　　:次のファイル管理テーブルを読み込みます

buf

ファイル格納情報を取得する、DIR_TBL 構造体のアドレスを指定します。

戻り値

関数が成功すると E_OK が返ります。失敗すると、次のエラーが返ります。

E_PRM　　:ファイル管理テーブルの読み込み位置指定エラー
E_NG　　:ファイル格納情報が存在しない

参照

DIR_TBL 構造体

9.1.15 dat_fsize

ファイル格納領域の未使用領域サイズを取得します。(DT-700 互換モード専用)

```
UW dat_fsize();
```

パラメータ

ありません。

戻り値

関数が成功するとファイル格納領域の未使用領域サイズが返ります。

9.1.16 dat_fdel

ファイルを削除します。(DT-700 互換モード専用)

```
ER dat_fdel(
  B     *name
);
```

パラメータ

name
削除対象ファイルのパスを、形式 1 で指定します。

戻り値

関数が成功すると E_OK が返ります。失敗すると、次のエラーが返ります。
E_PRM　　:不当ファイル名
E_NG　　:指定ファイルなし

解説

name パラメータの指定形式については、fil_remove 関数を参照してください。
name パラメータには、ANK コードのみ指定することできます。シフト JIS を指定すると、本関数は失敗します。
name パラメータに空白を指定した場合、本関数は失敗します。

9.1.17 dat_frname

ファイル名の変更を行ないます。(DT-700 互換モード専用)

```
ERdat_frname(
   UB     *old,
   UB     *new
);
```

パラメータ

old
既存ファイルのパスを、形式 1 で指定します。

new
新しいファイルのパスを、形式 1 で指定します。

戻り値

関数が成功すると E_OK が返ります。失敗すると、次のエラーが返ります。
E_NG　　:異常終了

解説

old、*new* パラメータの指定形式については、fil_remove 関数を参照してください。
old、*new* パラメータには、ANK コードのみ指定することできます。シフト JIS を指定すると、本関数は失敗します。

9.1.18 dat_F_Search

検索条件を指定して、条件に一致するファイルデータを取得します。(DT-700 互換モード専用)

```
ER dat_F_Search(
   B      *filename,
   W      start_adr,
   H      fieldsize,
   H      keypos,
   H      keylen,
   UB     *code,
   UB     *sdata,
   W      *fpos
);
```

パラメータ

filename
検索対象のファイル名を指定します。

start_adr
検索を開始する相対アドレスを指定します。

fieldsize
1 検索データのサイズを指定します。

keypos
検索コードの格納先相対アドレスを指定します。

keylen
検索コードのデータサイズを指定します。

code
比較検索コードの格納先アドレスを指定します。

sdata
検索結果のデータを格納する領域のアドレスを指定します。

fpos
検索結果のデータアドレスを格納するアドレスを指定します。

戻り値

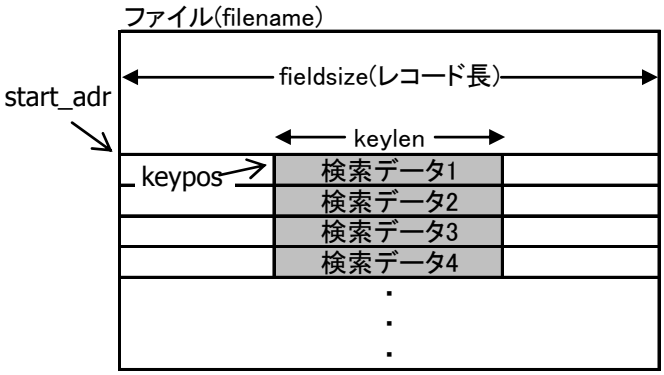
関数が成功すると E_OK が返ります。失敗すると、次のエラーが返ります。

E_NG	:異常終了
E_PRM	:パラメータエラー
E_NON	:検索データなし

解説

オープンしていないファイルに対して本関数を実行した場合、E_NG を返します。

それぞれのパラメータの関係は次のとおりです。



検索データ(code)

<例: 検索データ3で検索できた場合>
*srata = 検索データ3
*fpos = 検索データ3のアドレス

9.1.19 open

指定ファイルをオープンして、ファイル操作を可能にします。

```
int open(
   char   *name,
   int    mode
);
```

パラメータ

name
オープンするファイルのパスを指定します。

mode
ファイルのオープンモードを、次の値の組み合わせで指定します。

O_RDONLY	:読み取り専用でオープンします
O_WRONLY	:書き込みみ専用でオープンします
O_RDWR	:読み取り／書き込みみ両用でオープンします
O_CREAT	:ファイルを新規作成します
O_TRUNC	:ファイルの内容を破棄して、サイズを 0 にします
O_APPEND	:読み取り／書き込みを行なう位置を、ファイルの最後設定します O_APPEND を指定しない場合は、読み取り／書き込みを行なう位置をファイルの先頭に設定します。

設定可能な値の組み合わせは、解説を参照してください。

戻り値

関数が成功すると、0～15 のファイル番号が返ります。失敗すると、次のエラーが返ります。

E_LOWERR	:異常終了

解説

name パラメータの形式については、fil_remove 関数を参照してください。
mode パラメータに設定可能ファイルモードは次のとおりです。

■ ファイルシステムが FAT モードの場合

O_RDONLY	O_WRONLY	O_RDWR	O_CREAT	O_TRUNC	O_APPEND
○					
	○		○	○	
	○		○		○
		○			
		○	○	○	
		○	○		○

上記以外の組み合わせを指定した場合、本関数は失敗します。

■ ファイルシステムが DT-700 互換モードの場合
O_CREAT/O_TRUNC/O_APPEND を組み合わせることはできません。本関数は失敗します。

9.1.20 close

ファイルをクローズします。

```
int close(
   int    fileno
);
```

パラメータ

fileno
クローズするファイル番号を指定します。

戻り値

関数が成功すると E_OK が返ります。失敗すると、次のエラーが返ります。
E_LOWERR　　:異常終了

解説

fileno パラメータには、open 関数で取得したファイル番号を指定します。
本関数が正常終了すると、ファイルの最終更新日付を更新します。

9.1.21 read

指定ファイル番号のファイルに対し、現在の読み出し位置から指定バイト数分のデータを、指定の領域へ読み込みます。

```
int read(
   int            fileno,
   char           *buf,
   unsigned int   count
);
```

パラメータ

fileno
読み込み対象のファイル番号を指定します。

buf
読み込みデータを格納する領域のアドレスを指定します。

count
読み込データの要求バイト数を指定します。

戻り値

関数が成功すると実際に読み込んだデータのバイト数が返ります。
失敗すると、次のエラーが返ります。
E_LOWERR　　:異常終了

解説

fileno パラメータには、open 関数で取得したファイル番号を指定します。
指定バイト数以下でファイルが終了した場合は、そこで読み込みを終了します。

読出し位置は、読み込んだバイト数だけ先に進みます。正常終了した場合は、実際に読み込んだバイト数を返します。
データを読み込む前に、該当データブロックのチェックサムの確認を行ないます。チェックサムが正しくない場合は、本関数は失敗します。
書き込み専用モードファイルに対し本関数を実行した場合、リターンパラメータ値は E_LOWERR を返します。

9.1.22 write

指定ファイルに対し、現在の書き込み位置からデータを書き込みます。書き込み位置は、書き込めたデータ数だけ先に進みます。

```
int write(
   int            fileno,
   char           *buf,
   unsigned int   count
);
```

パラメータ

fileno
書き込み対象のファイル番号を指定します。

buf
書き込みデータを格納する領域のアドレスを指定します。

count
書き込みデータの要求バイト数を指定します。

戻り値

関数が成功すると実際に書き込んだデータのバイト数が返ります。
失敗すると、次のエラーが返ります。
E_LOWERR　　:異常終了

解説

fileno パラメータには、open 関数で取得したファイル番号を指定します。
読み込専用モードファイルに対し本関数を実行した場合、リターンパラメータ値は E_LOWERR を返します。

正常終了した場合は、実際に書き込めたデータバイト数を返します。
書き込み途中でファイルデータ領域が満杯になった場合も正常終了します。
連続して戻り値が 0 となるような場合、満杯状態と判断して異常終了します。

9.1.23 lseek

指定ファイルの読み込み／書き込み位置をバイト単位で設定します。

```
long lseek(
   int        fileno,
   long       offset,
   int        base
);
```

パラメータ

fileno
対象のファイル番号を指定します。

offset
読み込み／書き込み位置を指定します。
base パラメータからのオフセット値を指定します。

base
オフセットの基準を次の値で指定します。
0　:ファイルの先頭を基準とします
1　:現在の読み込み／書み込み位置を基準とします
2　:ファイルの終端を基準とします

戻り値

関数が成功するとファイルの先頭からのオフセット値が返ります。
失敗すると、次のエラーが返ります。
E_LOWERR　:異常終了

解説

オフセット値が負になる場合は、現在位置を更新しません。
ファイル終端を超える場合、DT-700 互換ファイルシステムでは、現在位置を更新しません。
一方、FAT システムでは、ファイル終端を越える場合、ファイルサイズを拡張します。

9.1.24 sbrk

要求されたデータサイズ分の領域をメモリ領域の下位アドレスから割り付けます。

```
char* sbrk(
   unsigned long  size
);
```

パラメータ

size
要求データのサイズを 1～16KB バイトの範囲で指定します。

戻り値

関数が成功すると割り付けた領域の先頭アドレスが返ります。
失敗すると、次のエラーが返ります。
E_LOWERR　　:異常終了

10.イベント通知機能

10.1 概要

デフォルトでは、電源キーを押下するとシステムが自動的に電源を OFF します。また、一定時間端末を無操作で放置した場合も自動的に電源を OFF します。しかし、時にはデフォルト動作を変更して、アプリケーションプログラム独自の処理を行いたいこともあります。このような目的のために用意されたのがイベント通知機能であり、電源の他に、ファンクションキーやタイマーもイベントとして取得することができます。

本機能で取得できるイベントは次の 3 種類に分類されます。

- 電源イベント
  電源 OFF キー、主電池電圧低下、APO（オートパワーOFF）、クレードル接続、その他

- キーイベント
  ファンクションキー押下、マルチファンクションキー押下

これらのイベントはデフォルトでシステムが処理しますが、アプリケーションプログラムは後述の関数を使って、イベントの発生を自分に通知するように設定することができます。

10.2 通知イベントの種類

10.2.1 電源イベント

電源 OFF キーイベント
このイベントの通知を有効に設定すると、電源キーが押されても本体の電源を OFF せず、電源 OFF キーのイベント(LB5)を通知します。

APO（オートパワーOFF）イベント
マシンを一定時間以上操作しないで放置すると、APO によって本体の電源を自動的に OFF します。アプリケーションプログラムがイベント通知を有効にすると、システムは電源を OFF する代わりに APO イベント(LB4)を通知します。

主電池電圧低下警告イベント
（主電池の電圧が一定以下になると画面下のステータス行に主電池電圧低下(LB1)シンボル()を表示します。）アプリケーションプログラムはイベント通知を有効にすることにより、このタイミングを主電池電圧低下(LB1)イベントによって知ることができます。

主電池なし、または電池蓋外しによる緊急 OFF イベント
主電池電圧が警告レベルを過ぎて、動作できないレベルに達した場合、または電源が ON の状態で電池蓋が開けられた場合、システムは安全のため、本体の電源を自動的に OFF します。アプリケーションプログラムはイベント通知を有効に設定することにより、緊急 OFF (LB0)の発生を次回の電源 ON 時に知ることができます。

副電池電圧低下警告イベント
副電池の電圧が一定以下になると画面下のステータス行に主電池電圧低下(LB2)シンボル()を表示します。アプリケーションプログラムはイベント通知を有効にすることにより、このタイミングを副電池電圧低下(LB2)イベントによって知ることができます。

クレードル接続検出イベント
アプリケーションプログラムはイベント通知を有効にすることにより、マシンがクレードルに接続されたイベントを取得することができます。

電源イベント通知設定／解除

NO	通知項目	デフォルト処理	通知有効時処理	通知タイミング	備考
1	電源 OFF キー (LB5)	電源 OFF	電源 OFF しない イベント通知	発生時	
2	APO (LB4)	電源 OFF	電源 OFF しない イベント通知	発生時	
3	主電池電圧低下警告 (LB1)	LB1 インジケート	シンボル表示 イベント通知	発生時	※
4	主電池なし、電池蓋開け (LB0)	電源 OFF	電源 OFF 処理 イベント通知	次回立上げ時	
5	副電池電圧低下警告 (LB2)	シンボル表示	シンボル表示 イベント通知	発生時	※
6	クレードル接続検出	何もしない	イベント通知	発生時	

※ 警告状態から復帰した場合、通知済み（未取得）のイベントをクリアします。

電源イベントのクリア

電源イベントを受け取ったアプリケーションプログラムは、pwr_inhabit_clr 関数を使ってそのイベントをクリアしなければなりません。これを行わないと、システムは何度も同じイベントを通知します。

10.2.2 キーイベント

キーイベントの通知を有効にすることにより、アプリケーションプログラムはキーコードを受け取らずにキー待ちから抜けることができます。これは、キー入力関数で入力待ち中にファンクションキーを処理する場合などに有効です。

下表のとおり、キーイベントの通知を有効にできるのは、ファンクションキー、マルチファンクションキーのみです。

	キー		設定可能
制御キー	入力モード切替(S)		不可
	後退(BS)		
	クリア(CLR)		
テンキー	テンキー 1	テンキー 2	不可
	テンキー 3	テンキー 4	
	テンキー 5	テンキー 6	
	テンキー 7	テンキー 8	
	テンキー 9	テンキー 0	
	テンキー .	テンキー ENT	
ファンクションキー	F1(-)	F2(←)	可能
	F3(→)	F4(DEL)	
	F5(SP)	F6(▲)	
	F7(▼)	F8(BL)	
マルチファンクションキー	マルチファンクションキー R		可能
	マルチファンクションキー L		
トリガーキー	ライトトリガーキー		不可
	レフトトリガーキー		
	センタートリガーキー		
カーソルキー	アップキー		不可
	ダウンキー		
	レフトキー		
	ライトキー		

※ マルチファンクションキーに OBR 読み込み機能を登録した状態で、さらに通知を有効に設定した場合、通知の設定を優先するため OBR の読み込みは開始されません。

10.2.3 タイマイベント

アプリケーションプログラムは2種類のインターバル・タイマを使うことができます。アプリケーションプログラムが通知を有効に設定すると、指定時間を経過するたびに、繰り返しイベントを通知します。

タイマ 1:1 秒単位のインターバルタイマです。

項目	仕様
最小単位	1sec
設定時間	1(1sec)～3600(1Hour)
誤差	要求時間+(最大)1sec
最大登録数	10
タイムアウト時の処理	指定時間経過後、指定のイベントフラグによって通知します。

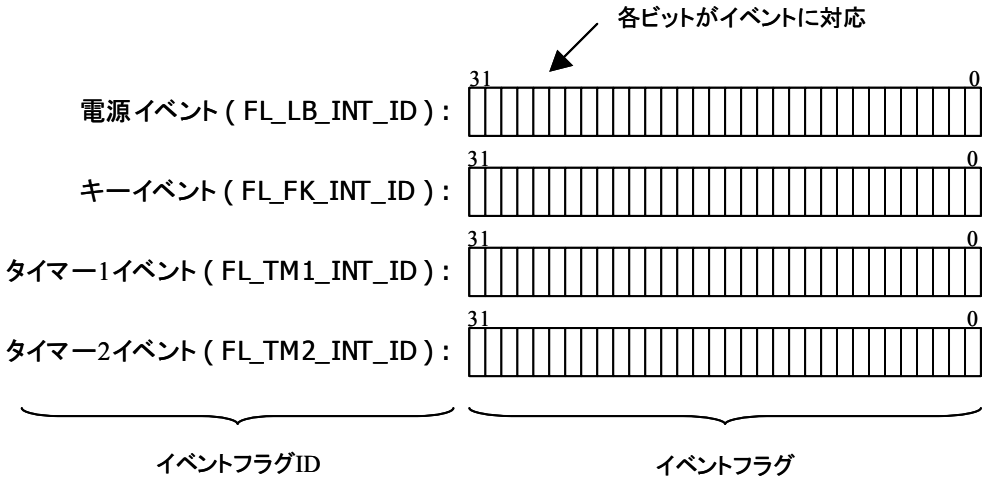
タイマ 2:31.25msec 単位のインターバルタイマです。

項目	仕様
最小単位	31.25msec
設定時間	1(31.25msec)～115200(1Hour)
誤差	要求時間+(最大)31.25msec
最大登録数	10
タイムアウト時の処理	指定時間経過後、指定のイベントフラグによって通知します。

10.3 イベント通知のメカニズム

一般的なアプリケーションプログラムは key_read、key_string、key_num 関数によってユーザーの入力を待ちます。イベント駆動型プログラミング環境であれば、この状態からイベントに対応する個別の処理を呼び出すことができるのですが、本環境にはこのような仕組みがありません。そのため、イベントの通知を有効に設定すると、当該イベントの発生によって key_read、key_string、key_num 関数は入力待ちを抜け、呼び元であるアプリケーションプログラムに戻り値をもってイベントの発生を伝えます。他にも ref_flg 関数でイベントの発生状況を調べることや、wai_flg 関数によってイベントが発生するまで待つこともできます。

発生したイベントはアプリケーションプログラムから参照するために、システム内に確保されたイベントフラグと呼ばれる領域（下図参照）に保持されます。イベントフラグは下図のように、イベント種別毎に用意された 32 ビットのフラグであり、それぞれがイベントフラグ ID によって管理されます。



※ 上記の 4 フラグはシステム内に用意された多数のイベントフラグのうち、アプリケーションプログラムに解放されたものです。

イベントフラグの各ビットが詳細イベントに対応しており、下表に示す名前で定義されています。ビットを数値の代わりに名前で定義するのは、機種間での互換性を確保するためです。

イベントが発生するとシステムは該当するイベントフラグの対応ビットを ON します。そして、このビットはアプリケーションプログラムが後述の関数によって OFF するまで保持されます。

電源イベントの通知（FL_LB_INT_ID 用）

ビットパターン	内容
FL_LB_INT_LB0	主電池なし、または電池蓋開け(LB0)
FL_LB_INT_LB1	主電池電圧低下警告(LB1)
FL_LB_INT_LB2	副電池電圧低下警告(LB2)
FL_LB_INT_LB4	APO(LB4)
FL_LB_INT_LB5	電源 OFF キー(LB5)
FL_SET_INT_IO	クレードル接続検出／USB 接続検出

キーイベントの通知（FL_FK_INT_ID 用）

ビットパターン	内容
FL_FK_INT_FNC1	F1 キー押下イベント用
FL_FK_INT_FNC2	F2 キー押下イベント用
FL_FK_INT_FNC3	F3 キー押下イベント用
FL_FK_INT_FNC4	F4 キー押下イベント用
FL_FK_INT_FNC5	F5 キー押下イベント用
FL_FK_INT_FNC6	F6 キー押下イベント用
FL_FK_INT_FNC7	F7 キー押下イベント用
FL_FK_INT_FNC8	F8 キー押下イベント用
FL_FK_INT_MLTL	マルチファンクションキーL 用
FL_FK_INT_MLTR	マルチファンクションキーR 用

タイマイベントの通知（FL_TM1_INT_ID、FL_TM2_INT_ID 用）

ビットパターン TM1	ビットパターン TM2	内容
FL_TM1_INT_RTC1	FL_TM2_INT_ITU1	タイムアウト設定用
FL_TM1_INT_RTC2	FL_TM2_INT_ITU2	
FL_TM1_INT_RTC3	FL_TM2_INT_ITU3	
FL_TM1_INT_RTC4	FL_TM2_INT_ITU4	
FL_TM1_INT_RTC5	FL_TM2_INT_ITU5	
FL_TM1_INT_RTC6	FL_TM2_INT_ITU6	
FL_TM1_INT_RTC7	FL_TM2_INT_ITU7	
FL_TM1_INT_RTC8	FL_TM2_INT_ITU8	
FL_TM1_INT_RTC9	FL_TM2_INT_ITU9	
FL_TM1_INT_RTC10	FL_TM2_INT_ITU10	
FL_TM1_INT_RTC11	FL_TM2_INT_ITU11	
FL_TM1_INT_RTC12	FL_TM2_INT_ITU12	
FL_TM1_INT_RTC13	FL_TM2_INT_ITU13	
FL_TM1_INT_RTC14	FL_TM2_INT_ITU14	
FL_TM1_INT_RTC15	FL_TM2_INT_ITU15	
FL_TM1_INT_RTC16	FL_TM2_INT_ITU16	
FL_TM1_INT_RTC17	FL_TM2_INT_ITU17	
FL_TM1_INT_RTC18	FL_TM2_INT_ITU18	
FL_TM1_INT_RTC19	FL_TM2_INT_ITU19	
FL_TM1_INT_RTC20	FL_TM2_INT_ITU20	
FL_TM1_INT_RTC21	FL_TM2_INT_ITU21	
FL_TM1_INT_RTC22	FL_TM2_INT_ITU22	
FL_TM1_INT_RTC23	FL_TM2_INT_ITU23	
FL_TM1_INT_RTC24	FL_TM2_INT_ITU24	
FL_TM1_INT_RTC25	FL_TM2_INT_ITU25	
FL_TM1_INT_RTC26	FL_TM2_INT_ITU26	
FL_TM1_INT_RTC27	FL_TM2_INT_ITU27	
FL_TM1_INT_RTC28	FL_TM2_INT_ITU28	
FL_TM1_INT_RTC29	FL_TM2_INT_ITU29	
FL_TM1_INT_RTC30	FL_TM2_INT_ITU30	
FL_TM1_INT_RTC31	FL_TM2_INT_ITU31	

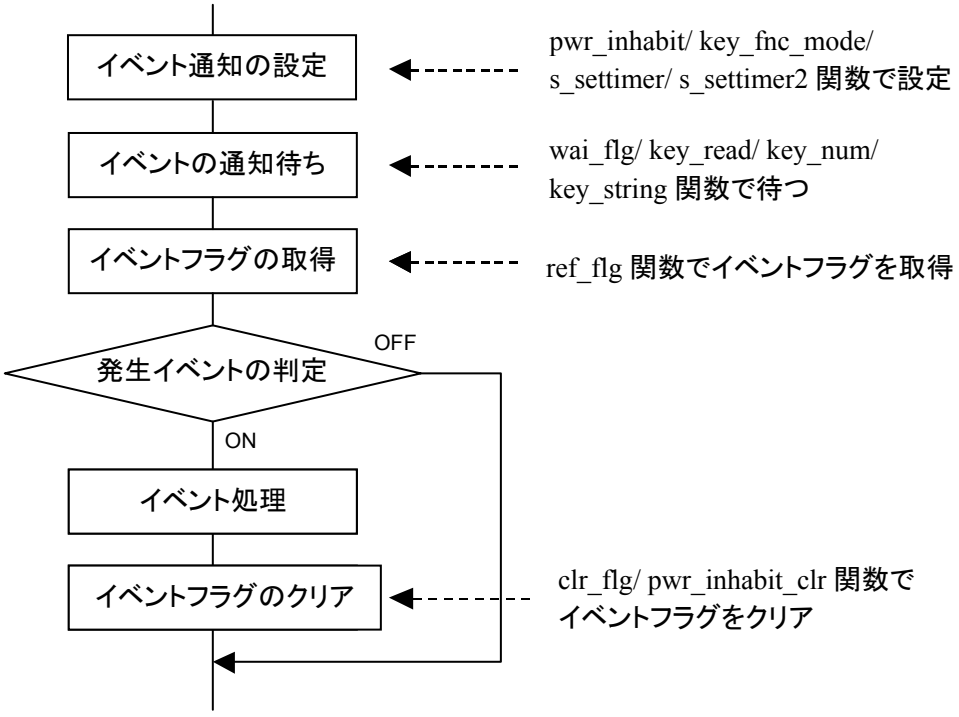
10.4 イベント通知の設定、取得とクリア

イベント通知の設定には、イベントフラグ ID 毎に用意された関数を使用します。イベントの通知状態（イベントフラグ）は、イベントの種類によらず ref_flg 関数で取得することができます。また、通知されたイベントをクリアする場合、電源イベントには pwr_inhabit_clr を使い、それ以外には clr_flg を使います。

イベントフラグ ID	通知を有効	通知を無効	イベントフラグを取得	イベントフラグをクリア	通知待ち
FL_LB_INT_ID	pwr_inhabit	←	ref_flg	pwr_inhabit_clr	wai_flg、key_read、key_string、key_num
FL_FK_INT_ID	key_fnc_mode	←		clr_flg	
FL_TM1_INT_ID	s_settimer	s_timerend			wai_flg
FL_TM2_INT_ID	s_settimer2	s_timerend2			

10.5 通知待ちによる処理の待機

イベント通知の設定を行なった後、wai_flg、key_read、key_string、key_num 関数を呼ぶと通知待ちの状態になります。



10.6 関数リファレンス

アプリケーションプログラムからイベントの通知に関する処理を行うには、以下の関数を使用します。

電源イベント

関数	機能概要
pwr_inhabit	電源イベントの通知を有効、または無効に設定
pwr_inhabit_clr	通知された電源イベントのクリア

キーイベント

関数	機能概要
key_fnc_mode	ファンクションキーイベントの通知を有効、または無効に設定

タイマイベント

関数	機能概要
s_settimer	タイマ 1 の登録
s_timerend	タイマ 1 の削除
s_settimer2	タイマ 2 の登録
s_timerend2	タイマ 2 の削除

イベントフラグ操作

関数	機能概要
ref_flg	指定のイベントフラグ ID に対応するイベントフラグを取得
clr_flg	指定のイベントフラグ ID に対して通知されたイベントをクリア

イベント待機

関数	機能概要
wai_flg	イベント発生待ち

10.6.1 pwr_inhabit

電源イベントの通知を有効、または無効に設定します。

```
ER pwr_inhabit(
   UH     onoff,
   ID     flgid,
   UW     bitptrn
);
```

パラメータ

onoff

イベント通知の有効・無効を次の値で指定します。

PWR_ON　　:*bitptrn* で指定したビットに対応するイベント通知を有効に設定します。
PWR_OFF　　:*bitptrn* で指定したビットに対応するイベント通知を無効に設定します。

flgid

設定対象のイベントフラグ ID を指定します。アプリケーションプログラムから呼び出す場合、この値は FL_LB_INT_ID 固定です。

bitptrn

通知を有効、または無効に設定する対象イベントを指定します。イベントが発生すると、イベントフラグの該当ビットが ON されます。複数のビットを OR で並べることで、複数のイベント通知を同時に設定することもできます。

FL_LB_INT_LB0　　:主電池なし、または電池蓋開け (LB0)
FL_LB_INT_LB1　　:主電池電圧低下警告 (LB1)
FL_LB_INT_LB2　　:副電池電圧低下警告 (LB2)
FL_LB_INT_LB4　　:APO (LB4)
FL_LB_INT_LB5　　:電源 OFF キー (LB5)
FL_SET_INT_IO　　:クレードル接続検出／USB 接続検出

戻り値

関数が成功すると E_OK が返ります。失敗すると次のエラーが返ります。

E_PRM　　:パラメータエラー

注意

電源イベントを受け取ったアプリケーションプログラムは、pwr_inhabit_clr 関数を使ってそのイベントをクリアしなければなりません。これを行わないと、システムは何度も同じイベントを通知します。

10.6.2 pwr_inhabit_clr

通知した電源イベント、およびクレードル検出イベントをクリアします。

```
ER pwr_inhabit_clr(
   ID     flgid,
   UW     bitptrn
);
```

パラメータ

flgid

設定対象のイベントフラグ ID を指定します。アプリケーションプログラムから呼び出す場合、この値は FL_LB_INT_ID 固定です。

bitptrn

クリア対象のイベントを指定します。複数を OR で並べることで、複数を同時にクリアすることもできます。

FL_LB_INT_LB0	:主電池なし、または電池蓋開け (LB0)
FL_LB_INT_LB1	:主電池電圧低下警告 (LB1)
FL_LB_INT_LB2	:副電池電圧低下警告 (LB2)
FL_LB_INT_LB4	:APO (LB4)
FL_LB_INT_LB5	:電源 OFF キー (LB5)
FL_SET_INT_IO	:クレードル接続検出／USB 接続検出

戻り値

関数が成功すると E_OK が返ります。失敗すると次のエラーが返ります。

E_PRM	:パラメータエラー

注意

電源イベントを受け取ったアプリケーションプログラムは、本関数でそのイベントをクリアしなければなりません。これを行わないと、システムは何度も同じイベントを通知します。

LB1、LB2 イベントは、本関数でクリアしても繰り返し通知されます。これは操作者が警告を見落としたことによる緊急 OFF の発生を避けるためです。

10.6.3 key_fnc_mode

ファンクションキーおよびマルチファンクションキーのイベント通知を有効・無効に設定、および設定値の取得を行ないます。

```
ER key_fnc_mode(
   UB      mode,
   UB      func_num,
   ID      *flgid,
   UW      *setptn
);
```

パラメータ

mode

イベント通知の有効・無効、および設定値の取得を次の値で指定します。

FNC_MODE_SET	:*func_num* で指定したキーのイベント通知を有効に設定します
FNC_MODE_CLR	:*func_num* で指定したキーのイベント通知を無効に設定します
FNC_MODE_RED	:*func_num* で指定したキーに設定された *flgid* と *setptn* を取得します

func_num

設定、および設定取得の対象とするファンクションキーを次の値で指定します。複数のキーを OR で指定することはできません。

FNC_1	:ファンクションキー1
FNC_2	:ファンクションキー2
FNC_3	:ファンクションキー3
FNC_4	:ファンクションキー4
FNC_5	:ファンクションキー5
FNC_6	:ファンクションキー6
FNC_7	:ファンクションキー7
FNC_8	:ファンクションキー8
MLT_R	:マルチファンクションキーR
MLT_L	:マルチファンクションキーL

flgid

FNC_MODE_SET 時は、対象のイベントフラグ ID（アプリケーションからは FL_FK_INT_ID 固定）を格納した領域のアドレスを指定します。

FNC_MODE_CLR 時は、システムはこの引数を参照しません。

FNC_MODE_RED 時は、*func_num* で指定したファンクションキーに設定された *flgid* を取得し、未設定の場合は 0 を返します。

setptn

FNC_MODE_SET 時は、*func_num* で指定するファンクションキーのイベント発生時にイベントフラグにセットするビットを指定します。

FNC_MODE_CLR 時は、システムはこの引数を参照しません。

FNC_MODE_RED 時は、*func_num* で指定したファンクションキーに設定された *setptn* を取得します。

戻り値

関数が成功すると E_OK が返ります。失敗すると次のエラーが返ります。

E_PRM　　:パラメータエラー

解説

本関数は *mode* に指定した値によって他の引数の扱いが変わります。
この関係を表したのが下表です。

mode	*func_num*	*flgid*	*setptn*
FNC_MODE_SET	設定対象のキー	FL_FK_INT_ID 固定	イベント発生時にセットするビットパターン
FNC_MODE_CLR	クリア対象のキー		
FNC_MODE_RED	取得対象のキー	取得した *flgid* 値を返します。未設定時はゼロを返します。(※)	取得した *flgid* 値を返します。未設定時はゼロを返します。(※)

(9) key_fnc_mode 関数呼出し前に値をセットする必要はありません。

10.6.4 s_settimer

タイマ 1 にイベントの発生間隔を登録し、通知を有効にします。イベントの発生間隔は 1 秒単位（最大 60 分）で指定でき、最大で 10 種類の値を登録することができます。登録に成功すると 0～9 のタイマ登録 ID を返します。このタイマ登録 ID は、s_timerend 関数による登録の削除に使います。

```
ER s_settimer(
   ID      flgid,
   UW      setptn,
   UW      tmcnt
);
```

パラメータ

flgid
対象のイベントフラグ ID を指定します。アプリケーションプログラムから呼び出す場合、この値は FL_TM1_INT_ID 固定です。

setptn
タイムアップ時にイベントフラグにセットするビットを FL_TM1_INT_RTC0 ～ FL_TM1_INT_RTC31 の範囲で指定します。複数のビットを OR で指定することもできます。

tmcnt
タイマカウントを 1～3600（1 カウント = 1 秒）の範囲で指定します。

戻り値

登録に成功すると 0～9 のタイマ登録 ID が返ります。失敗すると次のエラーが返ります。

E_PRM	:パラメータエラー
E_TID_OVER	:登録数オーバー

注意

本関数で登録したタイマーは s_timerend 関数で削除するまで、指定された発生間隔でイベントの通知を繰り返します。

イベントを受け取ったアプリケーションプログラムは適当な処理を行い、clr_flg 関数でイベントをクリアしてください。
イベントの通知が不要になったら s_timerend 関数でタイマの登録を削除してください。
タイマは最大+1 秒の誤差が生じます。

参照

s_timerend 関数

10.6.5 s_timerend

s_settimer 関数で登録したタイマを削除します。

```
ER s_timerend(
   ER     del_id
);
```

パラメータ

del_id
s_settimer 関数から返されたタイマ登録 ID を指定します。

戻り値

関数が成功すると E_OK が返ります。失敗すると次のエラーが返ります。

E_PRM	:パラメータエラー
E_TID_NON	:未登録タイマの削除

解説

不要になったタイマは、本関数を使って必ず削除して下さい。

参照

s_settimer 関数

10.6.6 s_settimer2

タイマ 2 にイベントの発生間隔を登録し、通知を有効にします。イベントの発生間隔は 31.25 ミリ秒単位（最大 60 分）で指定でき、最大で 10 種類の値を登録することができます。登録に成功すると 0～9 のタイマ登録 ID を返します。このタイマ登録 ID は、s_timerend2 関数による登録の削除に使います。

```
ER s_settimer2(
   ID      flgid,
   UW      setptn,
   UW      tmcnt
);
```

パラメータ

flgid
対象のイベントフラグ ID を指定します。アプリケーションプログラムから呼び出す場合、この値は FL_TM2_INT_ID 固定です。

setptn
タイムアップ時にイベントフラグにセットするビットを FL_TM2_INT_ITU0 ～ FL_TM2_INT_ITU31 の範囲で指定します。複数のビットを OR で指定することもできます。

tmcnt
タイマカウントを 1～115200（1 カウント = 31.25 ミリ秒）の範囲で指定します。

戻り値

関数が成功すると 00h～09h のタイマ登録 ID が返ります。失敗すると次のエラーが返ります。

E_PRM	:パラメータエラー
E_TID_OVER	:登録数オーバー

解説

本関数で登録したタイマーは s_timerend2 関数で削除するまで、指定された発生間隔でイベントの通知を繰り返します。

イベントを受け取ったアプリケーションプログラムは適当な処理を行い、clr_flg 関数でイベントをクリアしてください。
イベントの通知が不要になったら s_timerend2 関数でタイマの登録を削除してください。
タイマは最大+31.25 ミリ秒の誤差が生じます。

参照

s_timerend2 関数

10.6.7 s_timerend2

s_settimer2 関数で登録したタイマを削除します。

```
ER s_timerend2(
   ER     del_id
);
```

パラメータ

del_id
s_settimer2 関数で取得したタイマ登録 ID を指定します。

戻り値

関数が成功すると E_OK が返ります。失敗すると次のエラーが返ります。

E_PRM	:パラメータエラー
E_TID_NON	:未登録タイマの削除

解説

不要になったタイマは、本関数を使って必ず削除して下さい。

参照

s_settimer2 関数

10.6.8 ref_flg

指定のイベントフラグ ID に対するイベントの詳細情報を取得します。

```
ER ref_flg(
   ID          flgid,
   T_RFLG      *pk_rflg
);
```

パラメータ

flgid
対象のイベントフラグ ID を指定します。

pk_rflg
イベントフラグ情報を格納する下記構造体を指定します。

```
typedef struct   t_rflg {
   ID            wtskid;          /*  待ちタスクの有無                */
   FLGPTN        flgptn;          /*  現在のビット・パターン          */
} T_RFLG;
```

戻り値

関数が成功すると E_OK が返ります。失敗すると次のエラーが返ります。

E_ID	:ID 番号不正
E_CTX	:コンテキストエラー(タスクの実行環境エラー)

参照

T_RFLG 構造体

10.6.9 clr_flg

指定のイベントフラグ ID に対し、イベントをクリアします。

```
ER clr_flg(
   ID          flgid,
   FLGPTN      clrptn
);
```

パラメータ

flgid
クリア対象のイベントフラグ ID を指定します。

clrptn
クリアするイベントをビットで指定します。

戻り値

関数が成功すると E_OK が返ります。失敗すると次のエラーが返ります。

E_ID	:ID 番号不正
E_CTX	:コンテキストエラー（タスクの実行環境エラー）

説明

flgid で指定されたイベントフラグのビット・パターンと *clrptn* で指定されたビット・パターンの論理積 AND をとり、その結果を対象イベントフラグに設定します。

イベント通知を受け取ったアプリケーションプログラムは適当な処理を行い、本関数でイベントをクリアしてください。

10.6.10 wai_flg

指定のイベントフラグ ID に対し、待機条件を満たすまで待機します。

```
ER wai_flg(
   ID          flgid,
   FLGPTN      waiptn,
   MODE        wfmode,
   FLGPTN      *p_flgptn
);
```

パラメータ

flgid
待機対象のイベントフラグ ID を指定します。

waiptn
待機対象のイベントをビットで指定します。

wfmode
待機条件を指定します。
TWF_ANDW　: *waiptn* に指定したすべてのビットが ON になるまで待機します
TWF_ORW　: *waiptn* に指定したビットがひとつでも ON になるまで待機します

p_flgptn
条件成立時のビット・パターンを格納する領域へのポインタを指定します。

戻り値

関数が成功すると E_OK が返ります。失敗すると次のエラーが返ります。
E_PAR　:パラメータエラー
E_ID　:ID 番号不正
E_CTX　:コンテキストエラー（タスクの実行環境エラー）
E_ILUSE　:サービスコール不正使用
E_RLWAI　:待ち状態の強制解除

説明

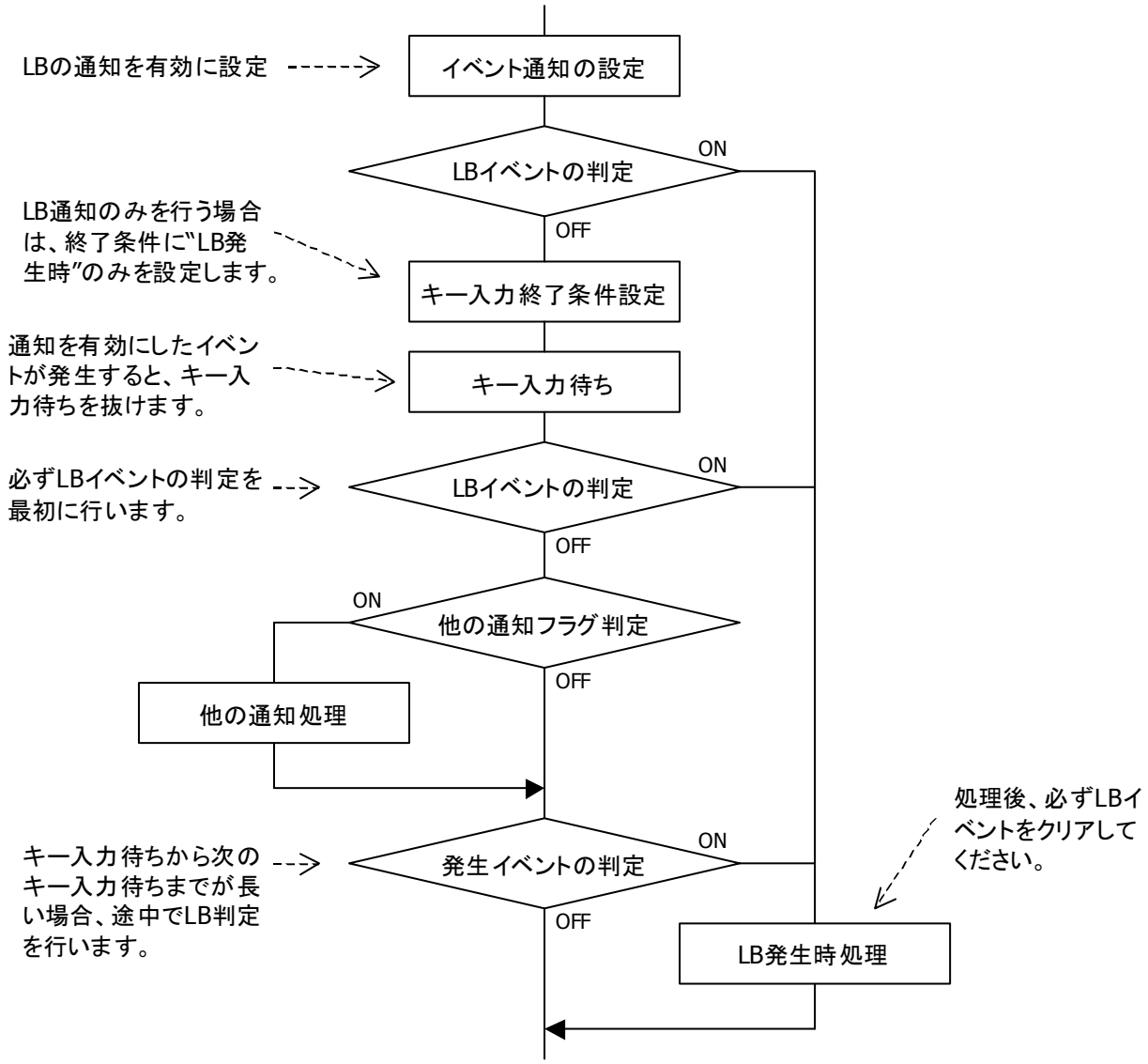
waiptn で指定された要求ビット・パターンと *wfmode* で指定された条件を満足するビット・パターンが *flgid* で指定されたイベントフラグに設定されているか否かをチェックします。
既に待機条件が満たされている場合は、対象イベントフラグのビット・パターンを *p_flgptn* で指定された領域に格納します。
待機条件を満足していなかった場合は，自タスクをイベントフラグ待ち状態へと遷移させます。

10.7 サンプルコード

10.7.1 電源イベント通知

LB0、1、2 に対するイベント通知のサンプルを示します。

処理の流れ



ソースコード

```
ER          err, retcd;
UW          i;
T_RFLG      rflg;
KEY_INP     keyinf;

/* イベント通知を設定 LB0,LB1,LB2 */
pwr_inhabit( PWR_ON, FL_LB_INT_ID, FL_LB_INT_LB0|FL_LB_INT_LB1|FL_LB_INT_LB2 );
    ・
    ・
    ・
for( i = 0, retcd = E_KEY_LB; i < 2 && retcd == E_KEY_LB; i++ )
{
   /* イベントフラグ状態の取得 */
   rflg.wtskid = 0;
   rflg.flgptn = 0;
   err =  ref_flg( FL_LB_INT_ID, &rflg );

   if( rflg.flgptn & FL_LB_INT_LB0 )   /* LB0 通知あり */
   {
       /* LB0 イベントのクリア */
       pwr_inhabit_clr( FL_LB_INT_ID, FL_LB_INT_LB0 );
       /* LB0 イベント処理を実行 */
       sub_lb0();
   }
   else if( rflg.flgptn & FL_LB_INT_LB1 )  /* LB1 通知あり */
   {
       /* LB1 フラグ状態のクリア */
       pwr_inhabit_clr( FL_LB_INT_ID, FL_LB_INT_LB1 );
       /* LB1 イベント処理を実行 */
       sub_lb1();
   }
   else if( rflg.flgptn & FL_LB_INT_LB2 )  /* LB2 通知あり */
   {
       /* LB2 イベントのクリア */
       pwr_inhabit_clr( FL_LB_INT_ID, FL_LB_INT_LB2 );
       /* LB2 イベント処理を実行 */
       sub_lb2();
   }

   /* 1 文字入力 リターン条件の設定 */
   keyinf.ext = KEY_LB_EXT;         /* LB 通知による脱出を指定 */
   keyinf.echo = ECHO_OFF;
   keyinf.font_size = LCD_ANK_STANDARD;
   keyinf.type = LCD_ATTR_NORMAL;
   keyinf.column_pos = 0;
   keyinf.line_pos = 0;
```

```
      /* 1 文字入力 */
      retcd = key_read( &keyinf );
   }

   /* イベント通知の解除 */
   pwr_inhabit( PWR_OFF, FL_LB_INT_ID, FL_LB_INT_LB0|FL_LB_INT_LB1|FL_LB_INT_LB2);
      ・
      ・
      ・
/* LB0 イベント処理*/
void sub_lb0( void )
{
      ・
      ・
   return;
}

/* LB1 イベント処理*/
void sub_lb1( void )
{
      ・
      ・
   return;
}

/* LB2 イベント処理*/
void sub_lb2( void )
{
      ・
      ・
   return;
}
```

注意

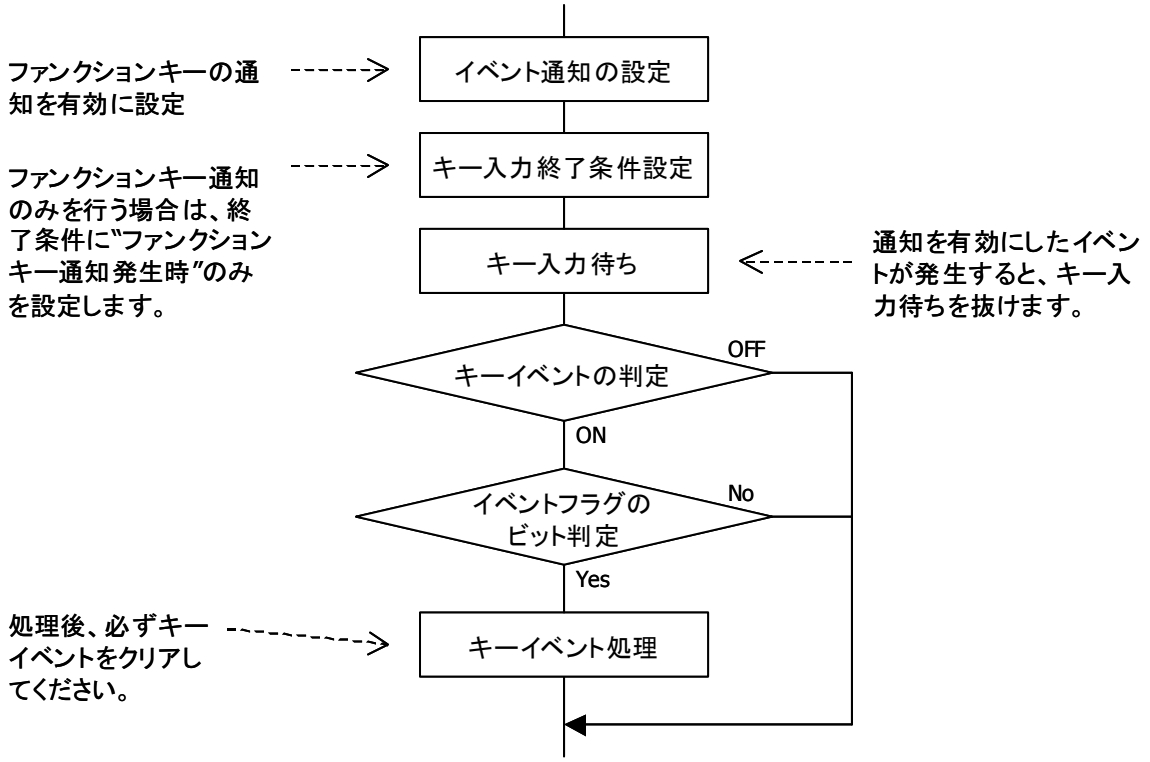
電源イベントに対するイベント通知を使用する場合、下記の点を注意してください。

- 複数のイベントに対して通知を有効に設定する場合、必ずLBイベントを最優先して処理してください。
- イベントの発生を確認する処理は、キー入力待ちの前後で行なってください。
- イベントの発生を確認する処理の間隔が長いと、イベントに対応した処理が遅れることになります。この間隔が長くなり過ぎないように注意してください。
- キー入力待ちの終了条件にLB 発生時を加えてください。
- 処理済みのイベントフラグは必ずクリアしてください。

10.7.2 キーイベント通知

ファンクションキー1、2 に対するイベント通知のサンプルを示します。

処理の流れ



ソースコード

```
ER          err, retcd;
UW          ptn;
T_RFLG      rflg;
KEY_INP     keyinf;
ID          fid;
        .
        .
/* イベント通知の設定 ファンクションキー1 */
fid = FL_FK_INT_ID;
ptn = FL_FK_INT_FNC1;
err = key_fnc_mode( FNC_MODE_SET, FNC_1, &fid, &ptn );

/* イベント通知の設定 ファンクションキー2 */
fid = FL_FK_INT_ID;
ptn = FL_FK_INT_FNC2;
err = key_fnc_mode( FNC_MODE_SET, FNC_2, &fid, &ptn );
        .
        .
        .
/* 1文字入力 リターン条件の設定 */
keyinf.ext = KEY_INT_EXT;  /* KEY通知による脱出を指定 */
keyinf.echo = ECHO_OFF;
keyinf.font_size = LCD_ANK_STANDARD;
keyinf.type = LCD_ATTR_NORMAL;
keyinf.column_pos = 0;
keyinf.line_pos = 0;
retcd = key_read( &keyinf );
if( retcd == E_KEY_INT)    /* KEYによる脱出 */
{
   /* フラグ状態の取得 */
   rflg.wtskid = 0;
   rflg.flgptn = 0;
   err =  ref_flg( FL_FK_INT_ID, &rflg );
   if( rflg.flgptn & FL_FK_INT_FNC1 )  /* ファンクションキー1通知 */
   {
       /* ファンクションキー1フラグ状態のクリア */
       clr_flg( FL_FK_INT_ID, FL_FK_INT_FNC1 );
       /* ファンクションキー1イベント処理を実行 */
       sub_fnc1();
   }
   else if( rflg.flgptn & FL_FK_INT_FNC2 ) /* ファンクションキー2通知 */
   {
       /* ファンクションキー2フラグ状態のクリア */
       clr_flg( FL_FK_INT_ID, FL_FK_INT_FNC2 );
       /* ファンクションキー2イベント処理を実行 */
       sub_fnc2();
   }
}
```

```
   /* イベント通知の解除 ファンクションキー1 */
   fid = FL_FK_INT_ID;
   ptn = FL_FK_INT_FNC1;
   err = key_fnc_mode( FNC_MODE_CLR, FNC_1, &fid, &ptn );

   /* イベント通知の解除 ファンクションキー2 */
   fid = FL_FK_INT_ID;
   ptn = FL_FK_INT_FNC2;
   err = key_fnc_mode( FNC_MODE_CLR, FNC_2, &fid, &ptn );
          .
          .
          .
/* ファンクションキー1 イベント処理*/
void sub_fnc1( void )
{
      .
      .
   return;
}

/* ファンクションキー2 イベント処理*/
void sub_fnc2( void )
{
      .
      .
   return;
}
```

注意

キーに対するイベントの通知を有効にする場合、下記の点に注意してください。

- イベントに対応する処理は、キー入力待ちの後で行なってください。
- 押下されたキーの識別は、イベントフラグのビットで判別してください。
- キー入力待ちの終了条件に"イベント通知キー押下(E_KEY_INT)"を加えてください。

11.画面表示

DT-970 の表示について説明します。
画面サイズは、シンボル表示領域を含めて 128×128 ドットです。

11.1 表示コード

表示コードは、シフト JIS コードを使用します。DT-970 のコード体系を以下に示します。

11.1.1 1バイトコード

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL		SP	0	@	P	`	p				-	タ	ミ		
1			!	1	A	Q	a	q			。	ア	チ	ム		
2			”	2	B	R	b	r			「	イ	ツ	メ		
3			#	3	C	S	c	s			」	ウ	テ	モ		
4			$	4	D	T	d	t			、	エ	ト	ヤ		
5			%	5	E	U	e	u			・	オ	ナ	ユ		
6			&	6	F	V	f	v			ヲ	カ	ニ	ヨ		
7			`	7	G	W	g	w			ァ	キ	ヌ	ラ		
8			(	8	H	X	h	x			ィ	ク	ネ	リ		
9			)	9	I	Y	i	y			ゥ	ケ	ノ	ル		
A	LF		*	:	J	Z	j	z			ェ	コ	ハ	レ		
B		ESC	+	;	K	[	k	{			ォ	サ	ヒ	ロ		
C			,	<	L	¥	l	|			ャ	シ	フ	ワ		
D	CR		-	=	M	]	m	}			ュ	ス	ヘ	ン		
E			.	>	N	^	n	~			ョ	セ	ホ	゛		
F			/	?	O	_	o				ッ	ソ	マ	゜		

※ ‘￥’コード
‘￥’(5Ch)を表示する場合、日本語モード時は‘￥’を、英語モード時は‘＼’を表示します。
日本語/英語モードは、動作環境メニューまたは、dat_system 関数で変更します。
‘＼’フォントデータは ROM に搭載していません。表示関数独自のデータを表示します。

制御コード	CR(0Ah)、LF(0Dh)、ESC(1Bh)
ANK	01h～80h、A0h～DFh、FDh～FFh

11.1.2 2バイトコード

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00																
10																
20																
30																
40																
50																
60																
70																
80					8140h											84FCh
										889Fh						
90									989Eh	989Fh						9FFCh
A0																
B0																
C0																
D0																
E0					E040h											EAFCh
					EB40h								EBC0h			
F0																

※ 2 バイト目が、7F のコード（例: EB7Fh）は存在しません。

第 1 水準	8140h～84FCh、889Fh～989Eh
第 2 水準	989Fh～9FFCh、E040h～EAFCh
外字	EB40h～EBCOh

11.2 フォントモード

DT-970 には、次の 3 つのフォントモードがあります。

- 6 ドットフォントモード
- 8 ドットフォントモード
- 10 ドットフォントモード

フォントモードは、dat_system 関数を使用して設定します。

複数のフォントモードのフォントを混在して表示することはできません。

11.3 フォントデータ

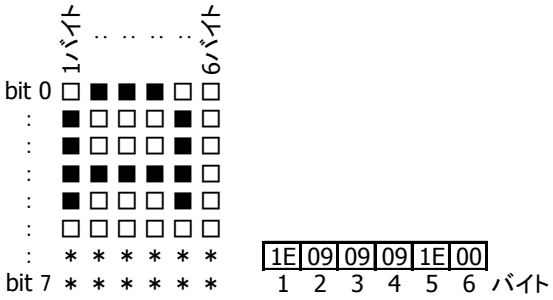
DT-970 では、ROM 搭載の縮小 ANK、標準 ANK、漢字の 3 種類のフォントを表示することができます。
6/8/10 ドットフォントモード用に、縮小 ANK、標準 ANK、漢字フォントを、それぞれ ROM に搭載しています。
フォントモードとフォント種類の関係とそれぞれのフォントサイズを以下に示します。

フォント	フォントサイズ		
	6 ドット	8 ドット	10 ドット
縮小 ANK	6×6	8×8	10×10
標準 ANK	6×12	8×16	10×20
（標準 ANK 縦強調）			
漢字	12×12	16×16	20×20

縦強調フォントを表示する場合は、dat_system 関数を使用して設定します。
ユーザーフォント表示の場合は、縦強調指定は無効になります。

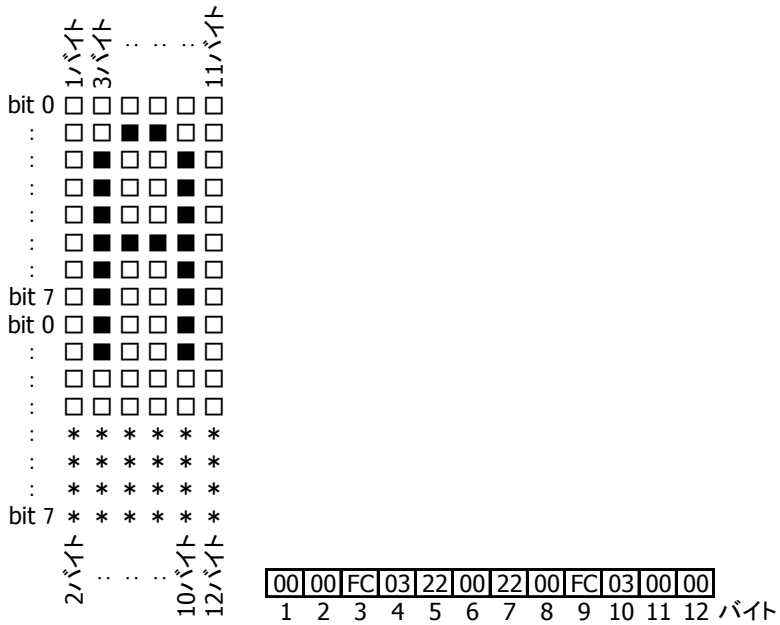
11.3.1 6 ドットフォント

縮小 ANK（6×6 ドット）
1 フォント 6 バイト構造



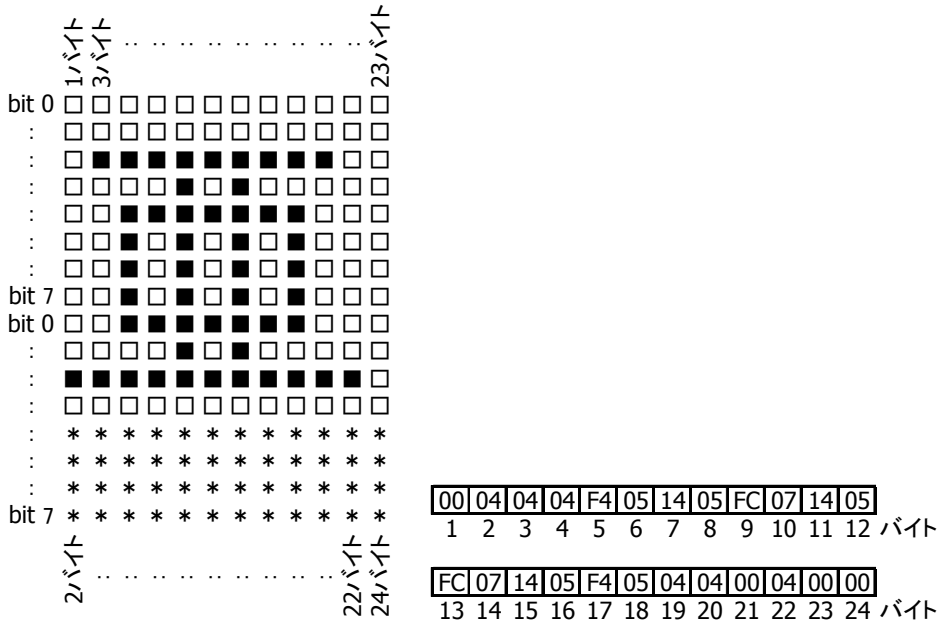
標準 ANK (6×12 ドット)

1 フォント 12 バイト構造



漢字 (12×12 ドット)

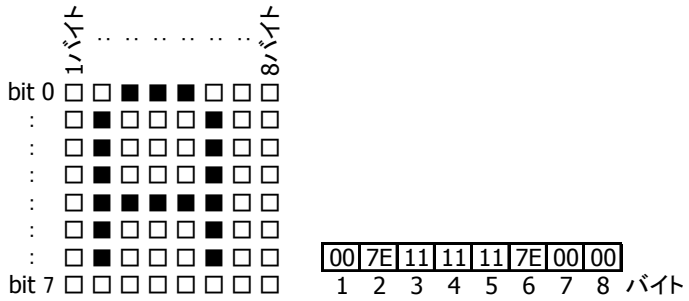
1 フォント 24 バイト構造



11.3.2 8ドットフォント

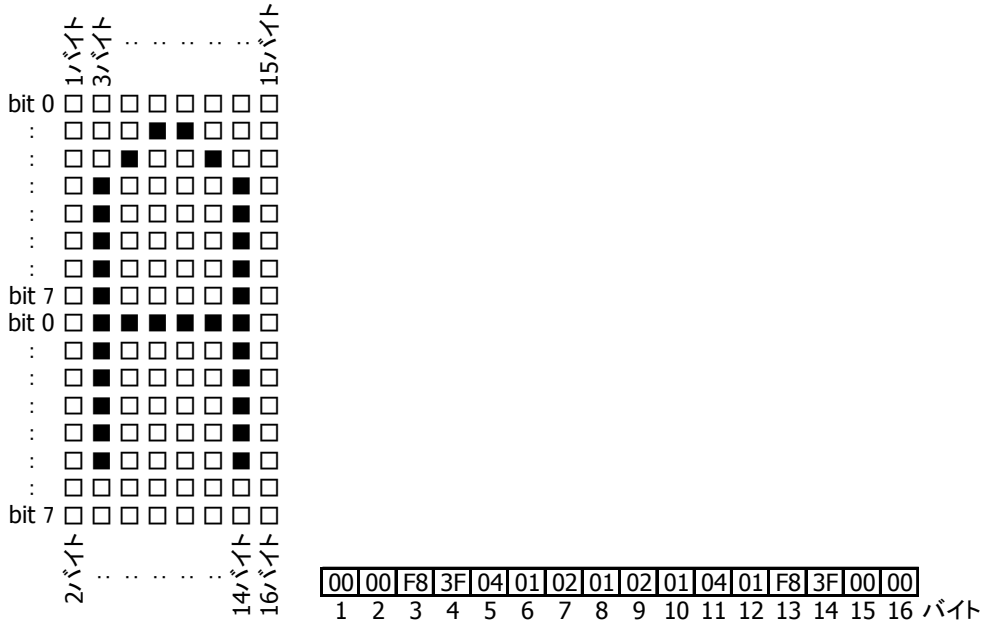
縮小 ANK (8×8ドット)

1フォント8バイト構造



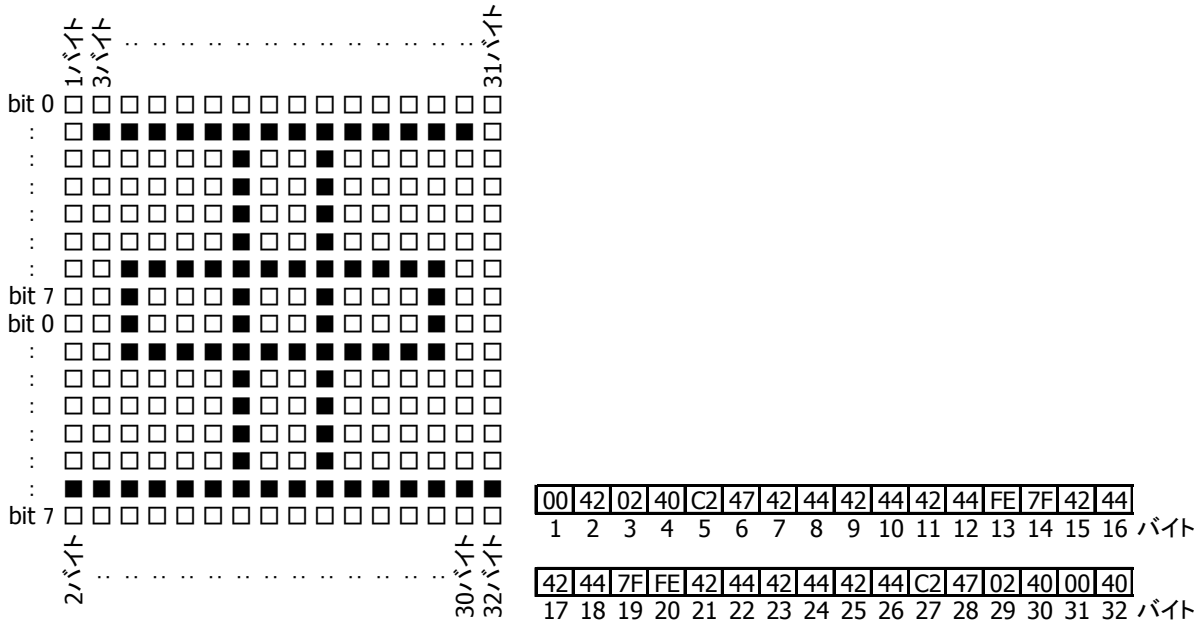
標準 ANK (8×16ドット)

1フォント12バイト構造



漢字（16×16 ドット）

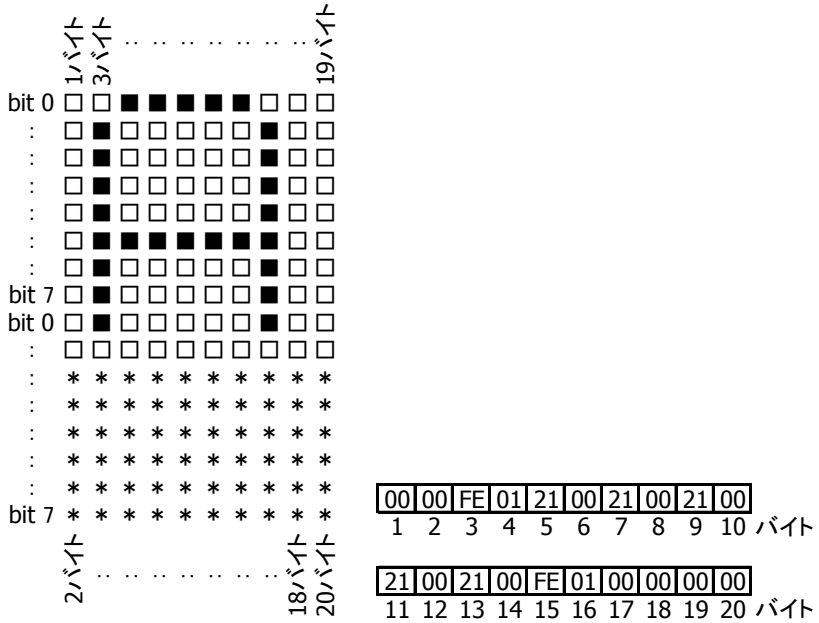
1 フォント 32 バイト構造



11.3.3 10 ドットフォント

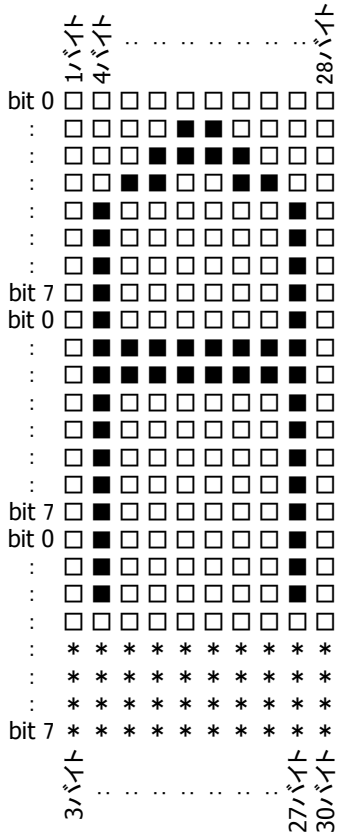
縮小 ANK（10×10 ドット）

1 フォント 20 バイト構造



標準 ANK（10×20 ドット）

1 フォント 30 バイト構造



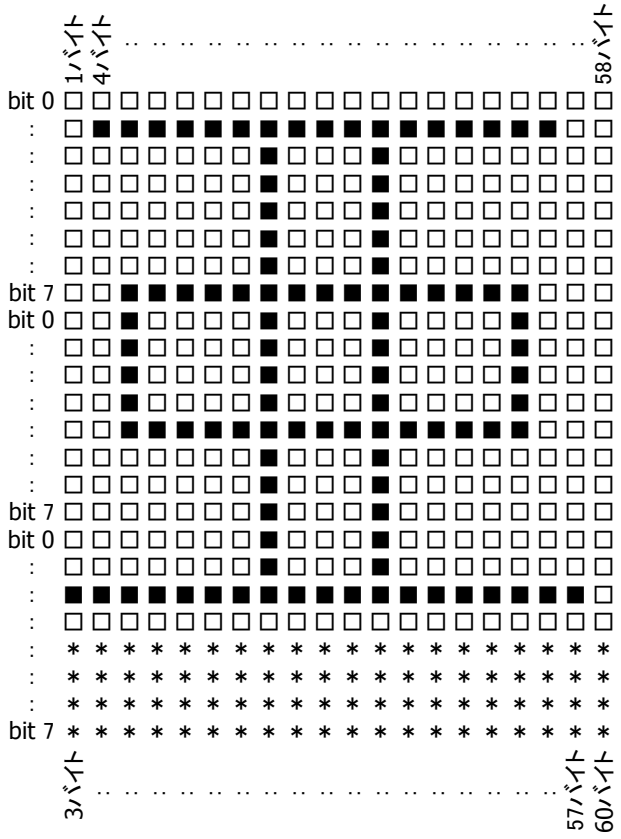
00	00	00	F0	FF	07	18	0C	00	0C
1	2	3	4	5	6	7	8	9	10 バイト

0C	06	00	0C	06	00	0C	00	0C	0C
11	12	13	14	15	16	17	18	19	20 バイト

00	18	0C	00	F0	FF	07	00	00	00
21	22	23	24	25	26	27	28	29	30 バイト

漢字（20×20 ドット）

1 フォント 60 バイト構造



00	00	04	02	00	04	82	1F	04	82
1	2	3	4	5	6	7	8	9	10 バイト

10	04	82	10	04	82	10	04	82	10
11	12	13	14	15	16	17	18	19	20 バイト

04	FE	FF	07	82	10	04	82	10	04
21	22	23	24	25	26	27	28	29	30 バイト

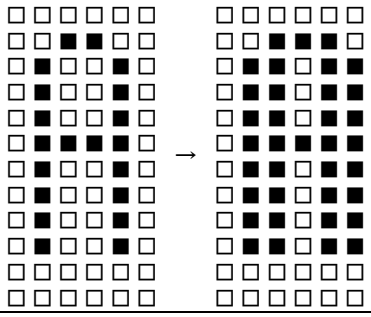
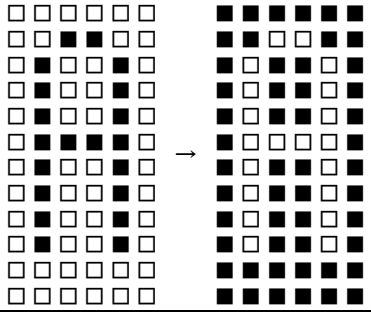
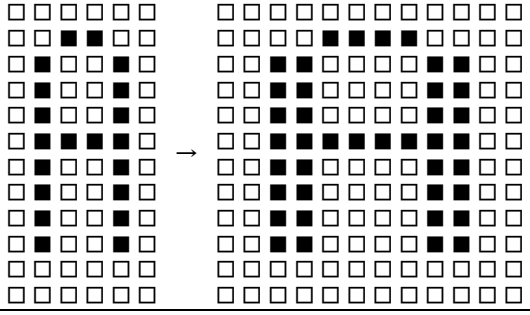
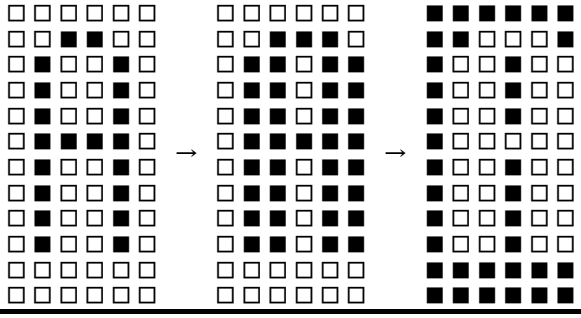
82	10	04	FE	FF	07	82	10	04	82
31	32	33	34	35	36	37	38	39	40 バイト

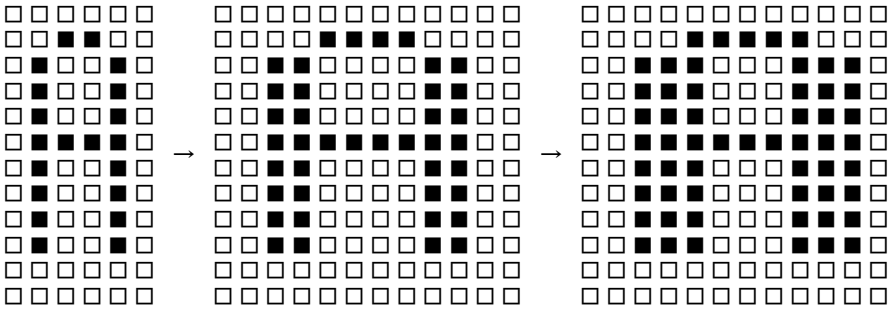
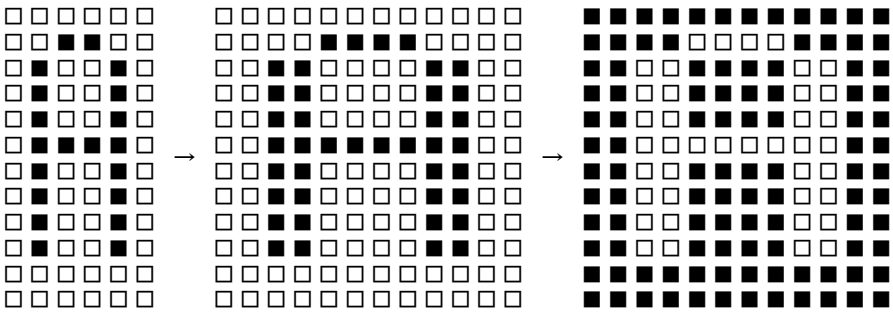
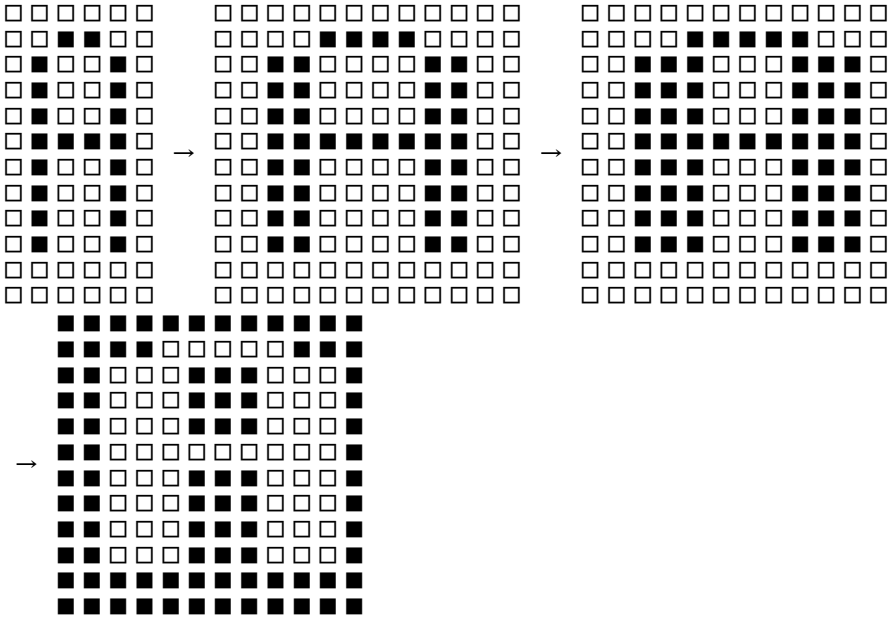
10	04	82	10	04	82	10	04	82	1F
41	42	43	44	45	46	47	48	49	50 バイト

04	02	00	04	00	00	04	00	00	00
51	52	53	54	55	56	57	58	59	60 バイト

11.4 フォント修飾

フォントに対する強調／反転／横倍角修飾は、次の処理を適用します。

修飾	処理
強調	標準フォントと標準フォントを 1 ドット右へずらしたビットパターンの論理和（OR）を表示します。 
反転	標準フォントのビットパターンを反転します。 
横倍角	標準フォントのビットパターンを横方向へ 2 倍にします。 
強調＋反転	強調修飾を行なったビットパターンに、反転修飾を行ないます。 

修飾	処理
強調＋横倍角	横倍角修飾を行なったビットパターンに、強調修飾を行ないます。 
反転+横倍角	横倍角修飾を行なったビットパターンに、反転修飾を行ないます。 
強調+反転+横倍角	横倍角修飾を行なったビットパターンに、強調修飾を行ない、その結果のビットパターンを反転します。 

11.5 フォントファイル

ROM 搭載以外のフォントを表示することができます。
フォント全部を交換するユーザーフォントとフォントを一部追加する外字フォントがあります。

11.5.1 ユーザーフォント

ユーザーが独自に作成したフォントを"ユーザーフォント"としてシステムに登録することで、ROM 搭載フォントを代替することができます。
ユーザーフォントの表示を行なう場合は、lcd_usrfont 関数を使用してシステムにユーザーフォントを登録します。ROM フォントに戻す場合は、lcd_romfont 関数を使用します。これによりユーザーフォントと ROM フォントを同一画面に混在して表示することができます。

※ DT-970 は、従来機種(DT-900)とフォントデータ構造が異なります。
※ DT-900 用のユーザーフォントファイルを、DT-970 で使用するためには、フォントコンバートツールで変換する必要があります。

ファイルフォーマット

フォント	ファイルフォーマット	備考
縮小 ANK 標準 ANK	File top 00h フォントデータ 01h フォントデータ : FFh フォントデータ File end	ファイルヘッダはありません、 00h～FFh までのフォントデータを連続して格納します。 フォントデータを途中までしか格納していない場合は、それ以後のコードはスペースを表示します。
漢字	File top 8140h フォントデータ : 84FFh フォントデータ 889Fh フォントデータ : 9FFFh フォントデータ E040h フォントデータ : EAFFh フォントデータ File end	ファイルヘッダはありません、 XX00h～XX3Fh、および 8840h～889Eh のフォントデータを格納する必要はありません。詰めて格納してください。 XX7Fh、XXFDh、XXFEh、XXFFh のフォントデータは、表示対象外ですが、ダミーデータを格納してください。 フォントデータを途中までしか格納していない場合は、それ以後のコードはスペースを表示します。

フォントファイルの最大容量

フォントモード	フォント	容量
6 ドット	縮小 ANK	1,536 バイト（6 バイト × 256 文字）
	標準 ANK	3,072 バイト（12 バイト× 256 文字）
	漢字	177,432 バイト（24 バイト×7,393 文字）
8 ドット	縮小 ANK	2,048 バイト（8 バイト× 256 文字）
	標準 ANK	4,096 バイト（16 バイト× 256 文字）
	漢字	236,576 バイト（32 バイト×7,393 文字）
10 ドット	縮小 ANK	5,120 バイト（20 バイト× 256 文字）
	標準 ANK	10,240 バイト（40 バイト× 256 文字）
	漢字	443,580 バイト（60 バイト×7,393 文字）

11.5.2 外字フォント

ユーザーが独自に作成したフォントを"外字フォント"としてシステムに登録することで、ROM 搭載フォントとともに表示することができます。
外字フォントのコード範囲は 0xEB40～0xEBC0 です。（ただし 0xEB7F は除きます）

外字フォントの表示を行なう場合は、lcd_gaiji 関数を使用してシステムに外字フォントを登録します。

※ DT-970 は、従来機種(DT-900)とはフォントデータ構造が異なります。
※ DT-900 用の外字フォントファイルを、DT-970 で使用するためには、フォントコンバートツールで変換する必要があります。

ファイルフォーマット

ファイルフォーマット	備考
File top EB40h フォントデータ EB41h フォントデータ : : EB7Eh フォントデータ EB80h フォントデータ : : EBC0h フォントデータ File end	ファイルヘッダはありません、 EB7Fh のフォントデータを格納する必要はありません。詰めて格納してください フォントデータを途中までしか格納していない場合は、それ以後のコードはスペースを表示します。

フォントファイルの最大容量

フォントモード	容量
6 ドット	3,072 バイト（24 バイト×128 文字）
8 ドット	4,096 バイト（32 バイト×128 文字）
10 ドット	7,680 バイト（60 バイト×128 文字）

11.6 表示

ここでは、画面出力仕様を説明します。

- 表示座標系
- 文字表示
- 制御コード表示
- ESC シーケンス表示
- スクロール制御
- 例外処理
- DT-930 互換表示
- 行挿入
- 拡大表示

11.6.1 表示座標系

DT-970 には、キャラクタ座標とグラフィック座標の 2 種類の表示座標系があります。

キャラクタ座標

キャラクタ座標とは、フォントモードの縮小 ANK を基準に、左上を 0 行 0 桁とする、行列座標系です。
フォントの表示は、キャラクタ座標で行ないます。
表示可能な桁数×行数は、フォントモードとフォント種類に依存して、次のとおり変動します。

フォントモード	フォントサイズ	タスクバー表示時		タスクバー非表示時	
		桁数×行数	最大表示文字数	桁数×行数	最大表示文字数
6 ドット	6× 6	21×18	378	21×21	441
	6×12	21× 9	189	21×10	210
	12×12	10× 9	90	10×10	100
8 ドット	8× 8	16×14	224	16×16	256
	8×16	16× 7	112	16× 8	128
	16×16	8× 7	56	8× 8	64
10 ドット	10×10	12×10	120	12×12	144
	10×20	12× 5	60	12× 6	72
	20×20	6× 5	30	6× 6	36

グラフィック座標

グラフィック座標とは、左上を(0,0)、右下を(127,111)（タスクバー表示時）、または右下を(127,127)（タスクバー非表示時）とする、ドット座標系です。
グラフィック座標は、フォントモードおよびフォント種類には依存しません。

11.6.2 文字表示

キャラクタ座標系に表示する関数には、1 文字表示(lcd_char)、文字列表示(lcd_string、lcd_string2)があります。関数に指定するコードと実際に表示される文字の関係を以下に示します。

文字表示(lcd_char)

1 バイト目	2 バイト目	ROM フォント時	ユーザーフォント指定時
00	00	何も表示しません	
	0A,0D	コントロールコード （「11.6.3 制御コード表示」参照）	
	01～09,0B,0C 0E～1F,81～9F E0～FC	ANK スペース	ユーザー登録の文字
	20～7F,A0～DF FD～FF	ANK コード表の文字	ユーザー登録の文字
81～84 89～9F,E0～EA	40～7E,80～FC	漢字コード表の文字	ユーザー登録の文字
	00～3F,7F,FD～FF	漢字スペース	8140h のフォント
88	9F～FC	漢字コード表の文字	ユーザー登録の文字
	00～9E,FD～FF	漢字スペース	8140h のフォント
EB	40～7E 80～C0	外字フォント文字 （外字ファイルあり時） 漢字スペース （外字ファイルなし時）	
	00～3F,7F,C1～FF	漢字スペース	8140h のフォント
上記以外	00～FF	何も表示しません	

文字列表示(lcd_string, lcd_string2)

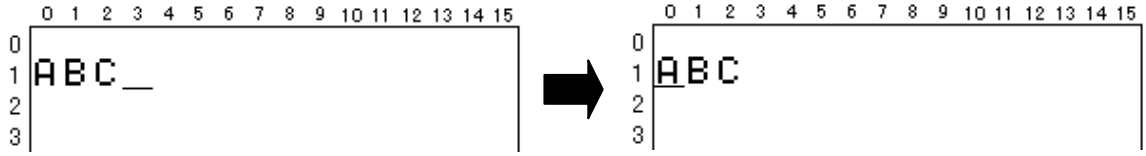
1 バイト目	2 バイト目	ROM フォント時	ユーザーフォント指定時
00		文字列表示の終了	
0A,0D		コントロールコード （「11.6.3 制御コード表示」参照）	
1B		ESC 制御 （「11.6.4 ESC シーケンス」参照）	
01～09,0B,0C 0E～1A,1C～1F 81～9F,E0～FC		ANK スペース	ユーザー登録の文字
20～80,A0～DF FD～FF		ANK コード表の文字	ユーザー登録の文字
81～84	00	文字列表示の終了 （画面表示しません）	
89～9F	40～7E,80～FC	漢字コード表の文字	ユーザー登録の文字
E0～EA	01～3F,7F,FD～FF	漢字スペース	8140h のフォント
88	00	文字列表示の終了 （画面表示しません）	
	9F～FC	漢字コード表の文字	ユーザー登録の文字
	01～9E,FD～FF	漢字スペース	8140h のフォント
EB	00	文字列表示の終了 （画面表示しません）	
	40～7E,80～C0	外字フォント文字 （外字ファイルあり時） 漢字スペース （外字ファイルなし時）	
	01～3F,7F,C1～FF	漢字スペース	8140h のフォント

※ ユーザー文字列表示(lcd_userstr)の場合は、漢字の表示はありません。すべて 1 バイト目を ANK コードとしてみます。81-84、88-9F、E0-EB のコードは ROM フォント時に ANK スペースを、ユーザーフォント時にはユーザー登録文字をそれぞれ表示します。

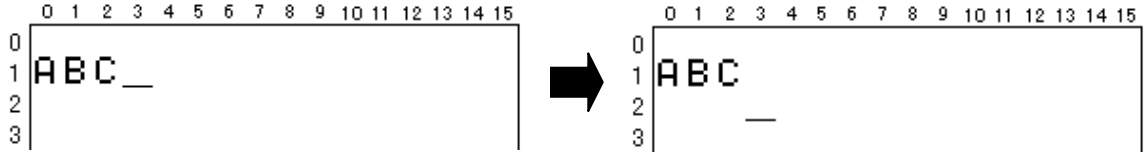
11.6.3 制御コード表示

1 文字表示／文字列表示では、制御コードを以下ように扱います。

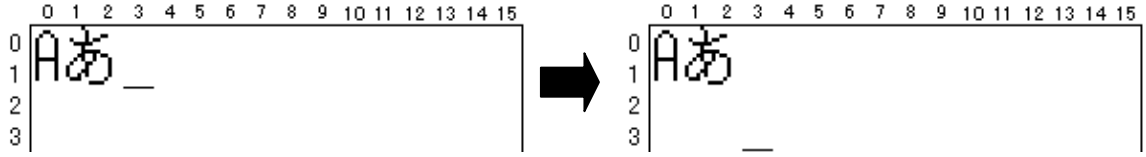
CR (0Dh)
キャリッジリターン動作を行ないます。



LF (0Ah)
縮小 ANK 文字列中の LF コードは、1 行改行します。

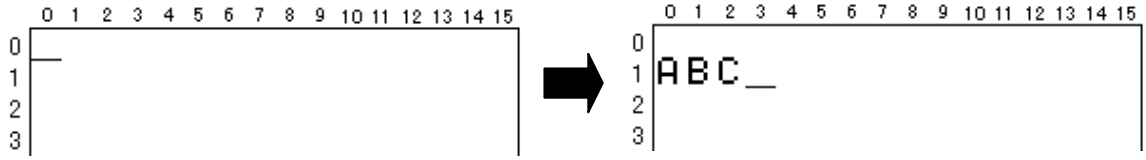


標準 ANK および漢字の文字列中に LF コードが含まれている場合は、2 行改行します。

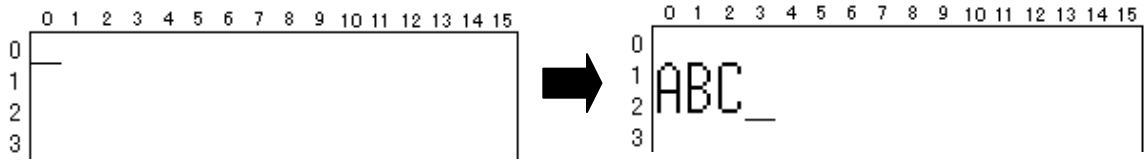


文字列表示の先頭が LF コードまたは、1 文字表示で LF コードの場合、入力パラメータの ANK モードが縮小の場合は 1 行、標準の場合は 2 行改行します。

例）座標(0,0)に縮小 ANK で"LFABC"を表示する場合



例）座標(0,0)に標準 ANK で"LFABC"を表示する場合



11.6.4 ESC シーケンス

画面クリア（ ESC[2J]
表示データをすべてスペースクリアします。カレントカーソル位置は 0 桁、0 行に移動します。

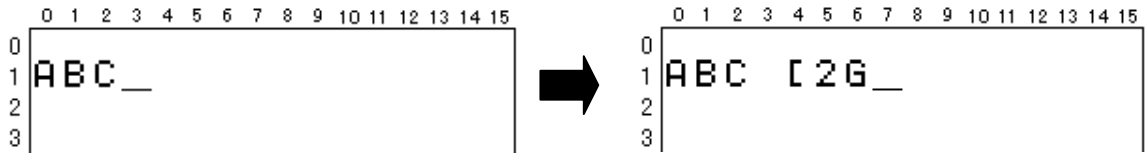
カーソル位置設定（ ESC[Pn;PmH　ESC[Pn;Pmf]
カレントカーソル位置を移動します。
n：行を指定します。
m：桁を指定します。

行列座標の始点は左上を 1 行、1 桁とします。座標の範囲は現在のフォントモードの縮小 ANK を基準とします。

Pm、Pn は省略することができます。省略した場合は1が指定されたものとします。

フォーマットエラー時の処理
ESC シーケンスのフォーマットに誤りがある場合は、文字列として表示します。

例）＂ESC [2J＂を＂ESC [2G＂と誤って指定した場合

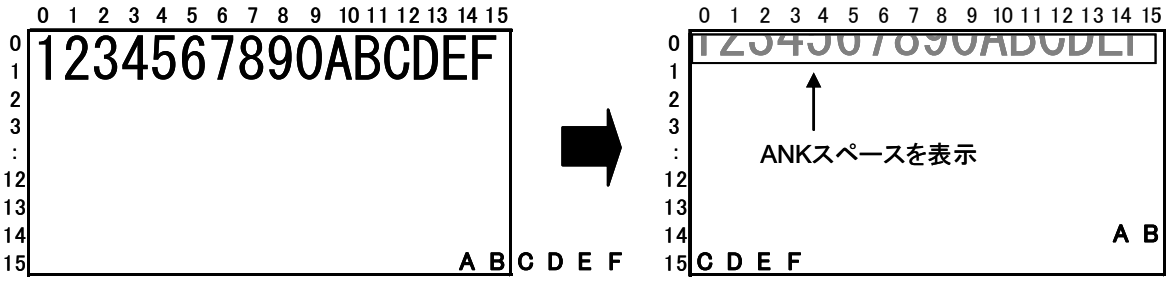


11.6.5 スクロール制御

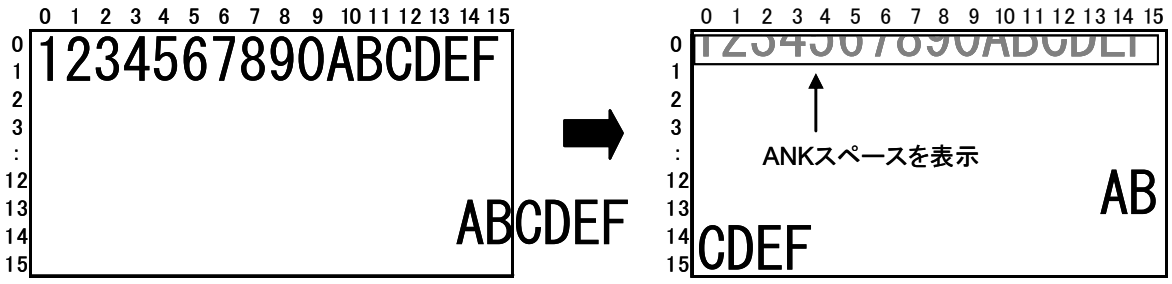
カレントカーソル位置が最下行で、表示する文字列が画面上から溢れる場合、スクロール制御を行ないます。スクロールする高さは、標準フォントと縮小フォントで異なります。
標準フォントの表示でスクロールが発生した場合は縮小フォントで 2 行分、縮小フォントの表示でスクロールが発生した場合は縮小フォントで 1 行分スクロールします。

縮小フォントの表示でスクロールが発生した時、最上行が標準フォントなら、最上行の標準フォントの下 1 行分は縮小フォントの ANK スペースになります。

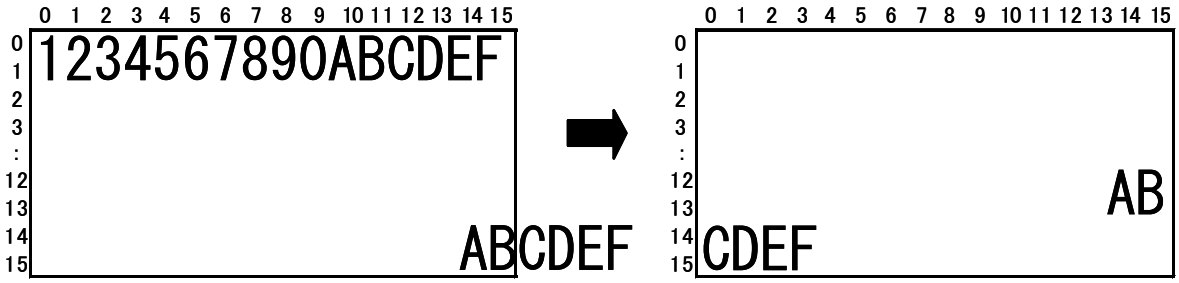
(9)　　最上行に標準 ANK が表示されているときに、最下行に縮小 ANK を表示する場合
座標(14,15)に縮小 ANK"ABCDEF"を表示



（例 2） 最上行に標準 ANK が表示されているときに、最下行-1 に標準 ANK を表示する場合
座標(14,14)に標準 ANK"ABCDEF"を表示



（例 3） 最上行に標準 ANK が表示されているときに、最下行に標準 ANK を表示する場合
座標(14,15)に標準 ANK"ABCDEF"を表示



※ スクロール抑制の抑止について
文字列表示2(lcd_string2)でスクロールを抑制できます。改行モードあり指定の時、最下行でスクロールが発生する条件になった場合でもスクロールを行ないません。表示しきれなかった文字については無視します。

11.6.6 例外処理

1 文字／文字列表示を行なう際、表示位置によって、すでに表示されている文字を ANK スペースコードでブロッククリアし、その上に新たな文字を表示する場合と、表示しない場合があります。

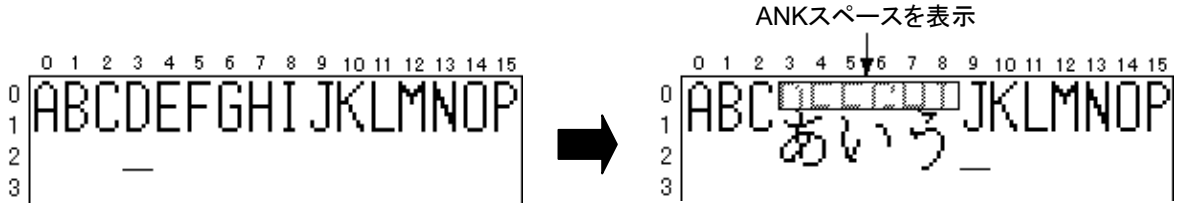
例外動作

動作	内容
表示データの重複	表示データと重なる部分の文字は、重なった文字全体をスペースクリアし、その上に新たな文字を表示します
最上位行での標準 ANK／漢字表示	画面からはみ出すのでその位置にスペースを表示します
行端での自動改行制御	文字列表示を行なうとき、行端で表示仕切れない場合かつ改行ありモードの時は、自動改行します
行端での漢字表示	行端で切れ端になる場合は、改行モードありの時、自動改行します

表示データの重複

表示データと重なる部分の文字はスペースクリアします。

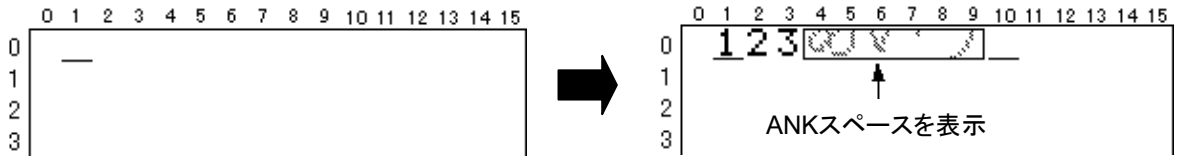
（例） 座標(3,2)に"あいう"を表示



最上位行での標準 ANK/漢字表示

カレントカーソル位置が最上位行で標準 ANK／漢字を表示する場合、
文字数分（漢字の場合は文字数×2）ANK スペースを表示します。

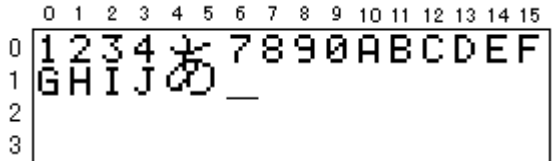
（例） 座標(1,0)に"123 あいう"を表示



行端での自動改行制御

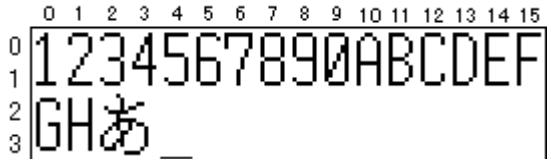
文字列表示を行なうとき、行端で表示仕切れない場合には先頭文字により、1または2行の改行を自動で行ないます。(ただし、改行モードあり指定時)

⑼　　先頭文字が縮小 ANK の場合、座標(0,0)に“1234567890ABCDEFGHIJ あ”を表示



※ 56 は上書きされます。

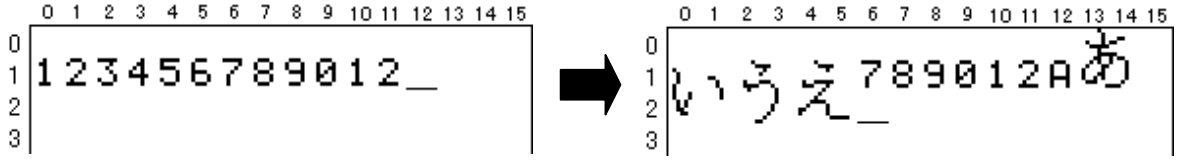
(例 2) 先頭文字が標準 ANK/漢字の場合、座標(0,0)に”1234567890ABCDEFGH あ”表示



行端での漢字表示制御

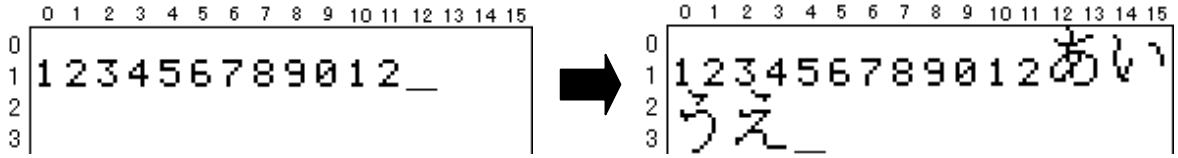
行端で切れ端になる場合には、1 または 2 行の改行を自動で行ないます。(ただし、改行モードありのとき)

⑼　　先頭文字が縮小 ANK の場合、座標(12,1)に“A あいうえ”を表示



※ い‘が入りきらないので改行します。
※ 先頭文字が縮小 ANK なので、1 行分改行します。
※ 123456 は上書きされます。

(例 2) 先頭文字が標準 ANK/漢字の場合、座標(12,1)に”あいうえ”表示



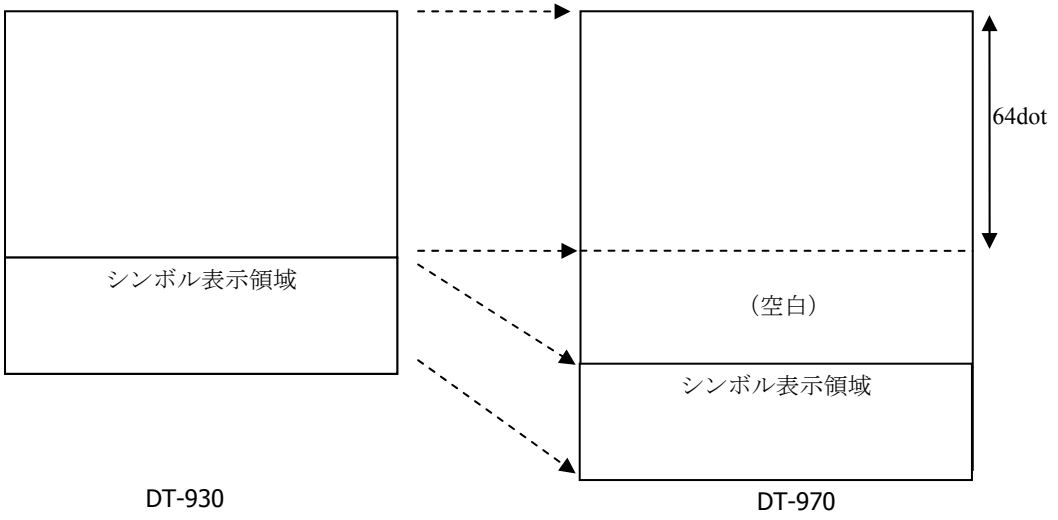
※ ‘う’が入りきらないので改行します。
※ 先頭文字が標準 ANK なので、2 行分改行します。

11.6.7 DT-930 互換表示

本表示関数では、DT-930 と互換を取るため通常表示モードの他に、2 つの互換モードを提供します。
アプリケーション作成時に互換モード用のオブジェクト（AP_START1.OBJ、AP_START2.OBJ）をリンクしてください。

DT-930 互換モード1

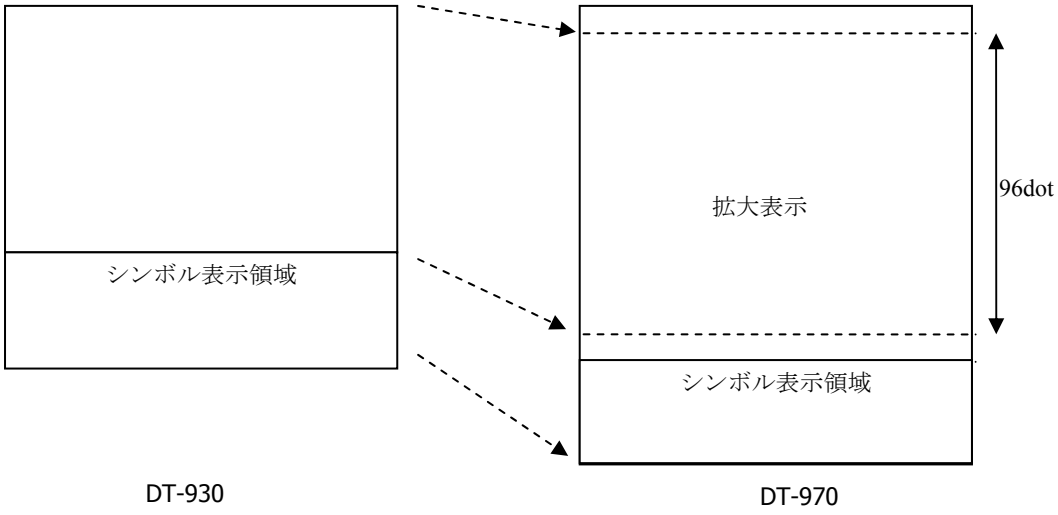
- 画面の上部 64dot に、DT-930 のアプリケーションを表示します。



（2）DT-930 互換モード2

- 各フォントモードのフォントを縦方向に 1.5 倍に拡大し、表示します。

※ グラフィック描画は互換性なし。（アプリケーションプログラムの修正が必要）



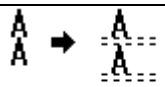
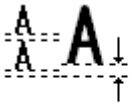
11.6.8 行挿入（行間ピッチ設定）

行間ピッチを設定することにより、行間を広げて表示します。
行間ピッチを設定する場合は、dat_system 関数を使用して設定します。

行間ピッチの設定範囲は、下記の通りです。

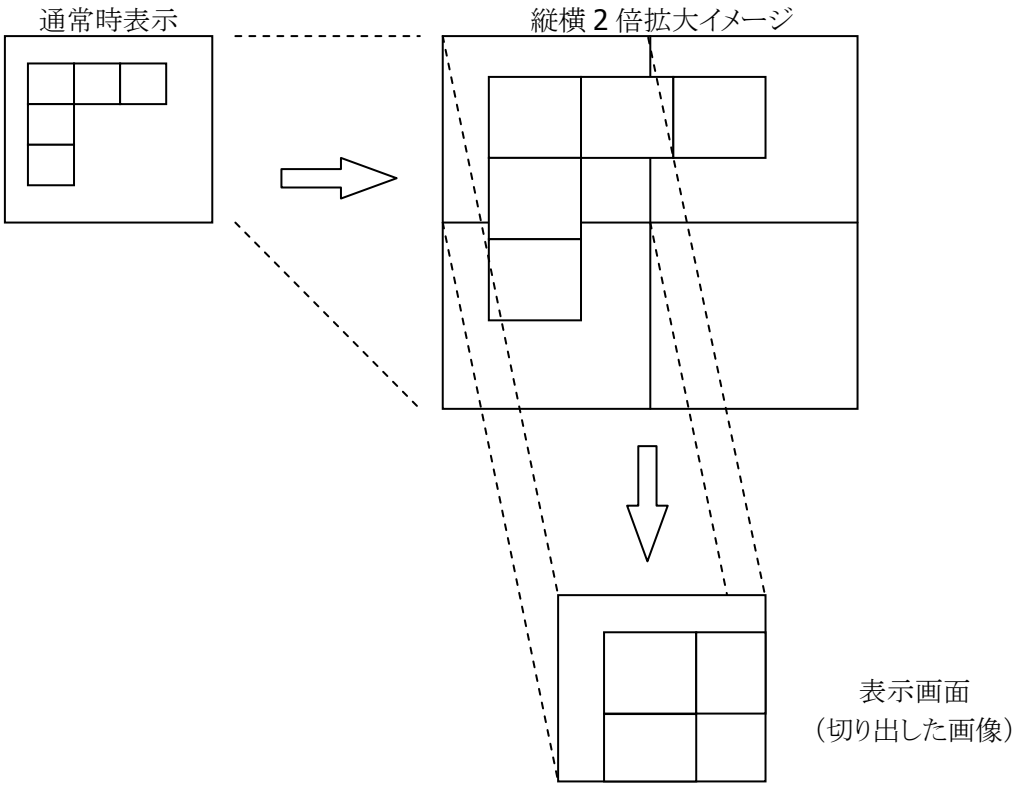
フォントサイズ	設定範囲
縮小 ANK	0～9
標準 ANK	縮小 ANK の 2 倍
漢字	縮小 ANK の 2 倍

行間ピッチ設定時は、下記のように表示されます。

 行間設定無　　行間設定有	
縮小 ANK 時 (点線部が行間ピッチ)	縮小 ANK と標準 ANK 混在時 (点線部が行間ピッチ。標準 ANK 文字の行間ピッチは 2 倍となる)

11.6.9 拡大表示

縦 2 倍、横 2 倍に拡大して表示します。拡大表示の有効無効は表示関数を使用して設定することが可能です。
縦横 2 倍拡大イメージに対して、表示したいイメージ位置の左上座標を指定することにより、その座標から拡大イメージを切り出して表示します。



11.7 関数リファレンス

ファンクション詳細を次ページより示します。

関数	機能概要
lcd_cls	画面クリア
lcd_csr_set	カーソル ON/OFF 設定
lcd_csr_put	カーソル位置設定
lcd_csr_get	カーソル位置読み出し
lcd_char	1文字表示
lcd_string	文字列表示
lcd_string2	文字列表示 2(スクロール抑制)
lcd_userstr	ユーザー文字列表示
lcd_line	直線描画／削除
lcd_gaiji	外字切り替え
lcd_usrfont	ユーザーフォントファイル登録
lcd_romfont	ROM フォント設定
lcd_led	LED ON/OFF 設定
lcd_el	EL バックライト ON/OFF 設定
lcd_expand_set	拡大表示を設定します。
lcd_expand_get	拡大表示設定状態を取得します
lcd_expand_pos_set	拡大表示の表示開始位置を設定します。
lcd_expand_pos_get	拡大表示の表示開始位置を取得します。
bmp_iDisplayBmpImage	BMP 画像ファイルを表示します。
bmp_iDisplayBmpData	BMP 画像データを表示します。

11.7.1 lcd_cls

表示データをすべてスペースクリアします。

```
ER lcd_cls();
```

パラメータ

ありません。

戻り値

関数が成功すると E_OK が返ります

解説

カレントカーソル位置をホームポジション(0,0)へ移動します。

11.7.2 lcd_csr_set

カーソル表示タイプ(カーソル非表示、アンダーラインカーソル、ブロックカーソル)を設定します。

```
ER lcd_csr_set(
   H  csr_type
);
```

パラメータ

csr_type

カーソルの表示タイプを、次の値を組み合わせて指定します。

LCD_CSR_OFF	:カーソル非表示
LCD_CSR_UNDER	:アンダーラインカーソル
LCD_CSR_BLOCK	:ブロックカーソル

戻り値

関数が成功すると E_OK が返ります。失敗すると次のエラーが返ります。

E_PRM	:パラメータエラー

解説

カーソルの形状は、カレントカーソル位置の表示コード種別(ANK／漢字)に関係なく横のドット数は ANK サイズとなります。

11.7.3 lcd_csr_put

カーソル位置を設定します。

```
ER lcd_csr_put(
   H  csr_line,
   H  csr_colum
)
```

パラメータ

csr_line

カーソルの行位置を、次の範囲で指定します。

6ドットモード　　:0～20 行
8ドットモード　　:0～15 行
10ドットモード　　:0～11 行

csr_colum

カーソルの桁位置を、次の範囲で指定します。

6ドットモード　　:0～20 桁
8ドットモード　　:0～15 桁
10ドットモード　　:0～11 桁

戻り値

関数が成功すると E_OK が返ります

解説

指定範囲の最大行、最大桁は各フォントモードの縮小 ANK を基準とします。
また、行／桁が最大値をこえる場合は、一番近い行／桁にカーソル位置を設定します。
行／桁は左上端を(0,0)とします。

11.7.4 lcd_csr_get

カレントカーソル位置とカーソル表示タイプを取得します。

```
ER lcd_csr_get(
  H  *csr_line,
  H  *csr_colum,
  H  *csr_type
);
```

パラメータ

csr_line
カーソルの行位置を格納する領域のアドレスを指定します。

csr_colum
カーソルの桁位置を格納する領域のアドレスを指定します。

csr_type
カーソルの表示タイプを格納する領域のアドレスを指定します。
下記の値が格納されます。

LCD_CSR_OFF	:カーソル非表示
LCD_CSR_UNDER	:アンダーラインカーソル
LCD_CSR_BLOCK	:ブロックカーソル

戻り値

関数が成功すると E_OK が返ります

11.7.5 lcd_char

現在のカーソル位置に 1 文字表示します。
ANK／漢字コードの表示ができます。
（標準／縮小 ANK のフォントデータ区別には引数の ANK モードを参照します。）
引数の表示属性で文字修飾表示が可能です。

```
ER lcd_char(
   H      ank_mode,
   H      disp_attr,
   UH     disp_data,
   H      lf_mode
);
```

パラメータ

ank_mode
ANK モードを次の値で指定します。
LCD_ANK_LIGHT　　:縮小 ANK モード
LCD_ANK_STANDARD　:標準 ANK モード

disp_attr
表示属性を次の値の組み合わせで指定します。
LCD_ATTR_NORMAL　:通常表示
LCD_ATTR_REVERS　:反転表示
LCD_ATTR_WIDTH　:強調表示
LCD_ATTR_DOUBLE　:横倍角表示

disp_data
表示対象のコードを指定します。

lf_mode
改行モードを次の値で指定します。
LCD_LF_OFF　:改行なし
LCD_LF_ON　:改行あり

戻り値

関数が成功すると E_OK が返ります。失敗すると次のエラーが返ります。
E_PRM　:パラメータエラー

解説

disp_data パラメータに 00h,0Dh,0Ah を指定した場合の動作は次のとおりです。
■ 00h コード
NULL(0x00)コードは、終了コードとして扱います。
■ 0Dh,0Ah コード
CR(0x0D),LF(0x0A)の表示動作が可能です。

文字の行端表示が改行ありモードの場合は自動改行します。改行なしモードの場合は改行しません。

11.7.6 lcd_string

現在のカーソル位置から文字列を表示します。

```
ER lcd_string(
  H     ank_mode,
  H     disp_attr,
  UB    *disp_data,
  H     lf_mode
);
```

パラメータ

ank_mode

ANK モードを次の値で指定します。

LCD_ANK_LIGHT	:縮小 ANK モード
LCD_ANK_STANDARD	:標準 ANK モード

disp_attr

表示属性を次の値の組み合わせで指定します。

LCD_ATTR_NORMAL	:通常表示
LCD_ATTR_REVERS	:反転表示
LCD_ATTR_WIDTH	:強調表示
LCD_ATTR_DOUBLE	:横倍表示

disp_data

表示データを格納する領域のアドレスを指定します。

lf_mode

改行モードを次の値で指定します。

LCD_LF_OFF	:改行なし
LCD_LF_ON	:改行あり

戻り値

関数が成功すると E_OK が返ります。失敗すると次のエラーが返ります。

E_PRM	:パラメータエラー

解説

disp_data パラメータに指定する文字列の有効バイト数は 1024 バイトです。ANK は 1024 文字、漢字は 512 文字までを表示することが可能です。1024 バイト以降は無視します。

disp_data パラメータに 00h,0Dh,0Ah を指定した場合の動作は次のとおりです。

- 00h コード
  NULL(0x00)コードは、終了コードとして扱います。
- 0Dh,0Ah コード
  CR(0x0D),LF(0x0A)の表示動作が可能です。

文字の行端表示が改行ありモードの場合は自動改行します。改行なしモードの場合は改行しません。（次の文字以降は無視します。）

11.7.7 lcd_string2

現在のカーソル位置から文字列を表示します。lcd_string 関数と同等の処理をしますが、改行ありモード時に、最下行でのスクロールを抑制します。

```
ER lcd_string2(
   H      ank_mode,
   H      disp_attr,
   UB     *disp_data,
   H      lf_mode
);
```

パラメータ

ank_mode
ANK モードを指定します。
LCD_ANK_LIGHT　　:縮小 ANK モード
LCD_ANK_STANDARD　　:標準 ANK モード

disp_attr
表示属性を指定します。
LCD_ATTR_NORMAL　:通常表示
LCD_ATTR_REVERS　:反転表示
LCD_ATTR_WIDTH　:強調表示
LCD_ATTR_DOUBLE　:横倍表示

disp_data
表示データを格納する領域のアドレスを指定します。

lf_mode
改行モードを指定します。
LCD_LF_OFF　　:改行なし
LCD_LF_ON　　:改行あり

戻り値

関数が成功すると E_OK が返ります。失敗すると次のエラーが返ります。
E_PRM　　:パラメータエラー

解説

disp_data パラメータに指定する文字列の有効バイト数は 1024 バイトです。ANK は 1024 文字、漢字は 512 文字までを表示することが可能です。1024 バイト以降は無視します。
改行ありモードで最下行右端になった場合はスクロールをしないで改行なしモードに切り替ります。
改行コード(CR･LF)は通常の改行処理を行ない、最下行にある場合はスクロールを行ないます。
各パラメータの詳細は、lcd_string 関数を参照してください。

参照

lcd_string 関数

11.7.8 lcd_userstr

現在のカーソル位置から ANK 文字列を表示します。

```
ER lcd_userstr(
   H       ank_mode,
   H       disp_attr,
   UB      *disp_data,
   H       lf_mode
);
```

パラメータ

ank_mode
ANK モードを指定します。
LCD_ANK_LIGHT　　:縮小 ANK モード
LCD_ANK_STANDARD　　:標準 ANK モード

disp_attr
表示属性を指定します。
LCD_ATTR_NORMAL　:通常表示
LCD_ATTR_REVERS　:反転表示
LCD_ATTR_WIDTH　:強調表示
LCD_ATTR_DOUBLE　:横倍表示

disp_data
表示データを格納する領域のアドレスを指定します。

lf_mode
改行モードを指定します。
LCD_LF_OFF　　:改行なし
LCD_LF_ON　　:改行あり

戻り値

関数が成功すると E_OK が返ります。失敗すると次のエラーが返ります。
E_PRM　　:パラメータエラー

解説

漢字の表示はできません。すべてのコードを ANK として扱います。
disp_data パラメータに指定する文字列の有効バイト数は 1024 バイトです。1024 バイト以降は無視します。
各パラメータの詳細は、lcd_string 関数を参照してください。

参照

lcd_string 関数

11.7.9 lcd_line

直線の描画と削除を行ないます。

```
ER lcd_line(
   H   dot_mode,
   H   strart_x,
   H   start_y,
   H   end_x,
   H   end_y
);
```

パラメータ

dot_mode
直線表示モードを次の値で指定します。
LCD_LINE_OFF　　:直線を削除します
LCD_LINE_ON　　:直線を描画します

start_x
開始 X 座標を 0～127dot の範囲で指定します。

start_y
開始 Y 座標を 0～127dot の範囲で指定します。

end_x
終了 X 座標を 0～127dot の範囲で指定します。

end_y
終了 Y 座標を 0～127dot の範囲で指定します。

戻り値

関数が成功すると E_OK が返ります。

解説

開始、終了座標が画面をはみ出す場合でもエラーにはなりません。
片方の座標がはみ出す場合は、描画できるところまで線を引きます。

11.7.10 lcd_gaiji

外字フォントデータファイルの登録(切り替え)を行ないます。

```
ER lcd_gaiji(
  H   file_mode,
  B   *filename
);
```

パラメータ

file_mode

ファイルモードを次の値で指定します。

LCD_6DOT_FILE	:6ドット外字登録ファイル
LCD_8DOT_FILE	:8ドット外字登録ファイル
LCD_10DOT_FILE	:10ドット外字登録ファイル

filename

登録対象の外字ファイル名を指定します。

戻り値

関数が成功するとE_OKが返ります。失敗すると次のエラーが返ります。

E_PRM	:パラメータエラー

解説

本関数を呼び出した時点で、外字ファイルからメモリへ外字フォントデータを展開します。
外字ファイルを更新した場合は、再登録が必要です。
ファイルオープンまたは、ファイルリードでエラーが発生した場合は、パラメータエラーを返します。

11.7.11 lcd_usrfont

ユーザーフォントをシステムに登録します。

```
ER lcd_usrfont(
  H   file_kind,
  B   *filename
);
```

パラメータ

file_kind

ファイル種別を次の値で指定します。

LCD_AL6_FILE	:縮小 ANK 6 ドットフォント
LCD_AS6_FILE	:標準 ANK 6 ドットフォント
LCD_K6_FILE	:漢字 6 ドットフォント
LCD_AL8_FILE	:縮小 ANK 8 ドットフォント
LCD_AS8_FILE	:標準 ANK 8 ドットフォント
LCD_K8_FILE	:漢字 8 ドットフォント
LCD_AL10_FILE	:縮小 ANK 10 ドットフォント
LCD_AS10_FILE	:標準 ANK 10 ドットフォント
LCD_K10_FILE	:漢字 10 ドットフォント

filename

登録対象のユーザーフォントファイル名を指定します。

戻り値

関数が成功すると E_OK が返ります。失敗すると次のエラーが返ります。

E_PRM	:パラメータエラー

解説

ファイル種別とフォントモードが妥当でない場合、ファイル名の登録のみを行ないます。ユーザーフォントデータの登録は行ないません。(ROM フォントデータを表示します。)

ユーザーフォントが ANK フォントの場合は、メモリにデータを読み込みます。
現在のフォントモードと異なるドットサイズのフォントファイルを登録した場合は。フォントモードの設定を行なうことで、登録ファイルのフォントが表示可能となります。

登録ファイルを無効にする場合は、lcd_romfont 関数を実行してください。
オープンエラーが発生した場合は、パラメータエラーを返します。

11.7.12 lcd_romfont

ユーザーフォントデータ表示から ROM フォントデータ表示へ切り替えます。

```
ER lcd_romfont();
```

パラメータ

ありません。

戻り値

関数が成功すると E_OK が返ります。

解説

ユーザーフォントと ROM フォントの混在表示が可能です。

11.7.13 lcd_led

LED の点灯と消灯を行ないます。

```
ER lcd_led(
  H   led_mode,
  H   led_kind
);
```

パラメータ

led_mode

LED モードを次の値で指定します。

LCD_LED_OFF	:LED 消灯
LCD_LED_ON	:LED 点灯

led_kind

LED 点灯種別を次の値で指定します。

LCD_LED_GREEN	:緑点灯
LCD_LED_RED	:赤点灯
LCD_LED_BLUE	:青点灯

戻り値

関数が成功すると E_OK が返ります。失敗すると次のエラーが返ります。

E_PRM	:パラメータエラー

解説

LED モードが LED 点灯の場合、すでに点灯している LED を消灯して点灯します。
LED モードが LED 消灯の場合には、点灯種別は無視します。(点灯中の LED を消灯します)

11.7.14 lcd_el

EL バックライトの点灯／消灯を行ないます。

```
ER lcd_el(
  H   el_mode
);
```

パラメータ

el_mode

EL モードを次の値で指定します。

LCD_EL_OFF	:EL 消灯
LCD_EL_ON	:EL 点灯（白色バックライト）
LCD_EL_HIGH	:EL 点灯（白色バックライト）
LCD_EL_ON_R	:EL 点灯（赤白色バックライト）

戻り値

関数が成功すると E_OK が返ります。失敗すると次のエラーが返ります。

E_PRM	:パラメータエラー

11.7.15 lcd_expand_set

画面の拡大表示を設定します。
拡大表示有効設定時は、本関数実行時の画面を拡大して表示します。

```
ER lcd_expand_set(
   H   expand_mode
);
```

パラメータ

expand_mode
拡大表示の有効／無効を次の値で指定します。
LCD_EXPAND_OFF　:拡大表示　無効
LCD_EXPAND_ON　:拡大表示　有効

戻り値

関数が成功すると E_OK が返ります。失敗すると次のエラーが返ります。
E_PRM　:パラメータエラー

11.7.16 lcd_expand_get

画面の拡大表示の設定状態を取得します。

```
ER lcd_expand_get(
   H   *pexpand_mode
);
```

パラメータ

pexpand_mode
拡大表示の設定（有効／無効）を格納する領域のアドレスを指定します。
下記の値が格納されます。
LCD_EXPAND_OFF　:拡大表示　無効
LCD_EXPAND_ON　:拡大表示　有効

戻り値

関数が成功すると E_OK が返ります。失敗すると次のエラーが返ります。
E_PRM　:パラメータエラー

11.7.17 lcd_expand_pos_set

拡大した画面イメージの表示開始位置(左上座標)を設定します。
開始位置は、8ドット単位で指定します。

```
ER lcd_expand_pos_set(
  H   xpos,
  H   ypos
);
```

パラメータ

xpos
拡大した画面イメージの表示開始位置のX座標(8ドット単位)を0～16の範囲で指定します。

ypos
拡大した画面イメージの表示開始位置のY座標(8ドット単位)を0～16の範囲で指定します。

戻り値

関数が成功するとE_OKが返ります。失敗すると次のエラーが返ります。
E_PRM　　:パラメータエラー

11.7.18 lcd_expand_pos_get

拡大した画面イメージの表示開始位置(左上座標)を取得します。
取得する開始位置は、8ドット単位の座標となります.

```
ER lcd_expand_pos_get(
  H   *pxpos,
  H   *pypos
);
```

パラメータ

pxpos
拡大した画面イメージの表示開始位置のX座標(8ドット単位)を格納する領域のアドレスを指定します。

pypos
拡大した画面イメージの表示開始位置のY座標(8ドット単位)を格納する領域のアドレスを指定します。

戻り値

関数が成功するとE_OKが返ります。失敗すると次のエラーが返ります。
E_PRM　　:パラメータエラー

11.7.19 bmp_iDisplayBmpImage

指定した BMP 画像のファイルを指定した座標に表示します。

```
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 座標を 0～127 の範囲で指定します。

ucClm
表示する x 座標を 0～127 の範囲で指定します。

戻り値

下記の値を返します。

BMP_OPERATONOK	:正常終了
BMP_NOMEMORY	:メモリが足りません。
BMP_NOFILEOPEN	:ファイルが開けません。
BMP_NOFILEREAD	:ファイルが読めません。
BMP_NOFILESEEK	:ファイルのシークができません。
BMP_INVALIDTYPE	:BMP 形式画像ではありません。
BMP_INVALIDWIDTH	:画像が画面の右にはみ出しています。
BMP_INVALIDHEIGHT	:画像が画面の下にはみ出しています。
BMP_NEGATIVEHEIGHT	:BMP ヘッダ情報の画像の高さの値が不正です。
BMP_INVALIDPLANES	:BMP ヘッダ情報のプレーン数の値が不正です。
BMP_INVALIDBITCNT	:白黒画像ではありません。
BMP_COMPRESSED	:圧縮形式の BMP 画像です。
BMP_NOPIXELDATA	:イメージデータ開始位置が不正です。
BMP_INVALIDROW	:y 座標の値が不正です。
BMP_INVALIDCLM	:x 座標の値が不正です。

解説

本関数は指定した BMP 形式画像のファイルを指定した座標に表示します。表示座標は、左上が始点(0,0)となります。
表示できる画像は白黒のみです。カラー画像の場合は BMP_INVALIDBITCNT エラーとなります。
圧縮形式の BMP 画像は扱えません。圧縮形式の場合は BMP_COMPRESSED エラーとなります。

本関数を使用する場合は、DTBMP.H をインクルードしてください。

11.7.20 bmp_iDisplayBmpData

指定した BMP 画像のデータ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

pszBMPData
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 座標を 0～127 の範囲で指定します。

ucClm
表示する x 座標を 0～127 の範囲で指定します。

戻り値

下記の値を返します。

BMP_OPERATONOK	:正常終了
BMP_NOMEMORY	:メモリが足りません。
BMP_NOFILEOPEN	:ファイルが開けません。
BMP_NOFILEREAD	:ファイルが読めません。
BMP_NOFILESEEK	:ファイルのシークができません。
BMP_INVALIDTYPE	:BMP 形式画像ではありません。
BMP_INVALIDWIDTH	:画像が画面の右にはみ出しています。
BMP_INVALIDHEIGHT	:画像が画面の下にはみ出しています。
BMP_NEGATIVEHEIGHT	:BMP ヘッダ情報の画像の高さの値が不正です。
BMP_INVALIDPLANES	:BMP ヘッダ情報のプレーン数の値が不正です。
BMP_INVALIDBITCNT	:白黒画像ではありません。
BMP_COMPRESSED	:圧縮形式の BMP 画像です。
BMP_NOPIXELDATA	:BMP データ開始位置が不正です。
BMP_INVALIDROW	:y 座標の値が不正です。
BMP_INVALIDCLM	:x 座標の値が不正です。

解説

本関数は指定した BMP 形式画像のデータを指定した座標に表示します。表示座標は、左上が始点(0,0)となります。
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 RAM に展開されたデータを表示に反映します。
表示できる画像は白黒のみです。カラー画像の場合は BMP_INVALIDBITCNT エラーとなります。
圧縮形式の BMP 画像は扱えません。圧縮形式の場合は BMP_COMPRESSED エラーとなります。

本関数を使用する場合は、DTBMP.H をインクルードしてください。

12.キー制御

DT-970 では、キーに対して次の処理を行なうことができます

- キーモードの切替え
- 1 文字の入力
- 文字列の入力
- 数値の入力
- キーコードの設定
- キーイベント通知の設定
- キー入力無効の設定
- 多点押しの制御
- キーロールオーバー

12.1 キーモード

DT-970 のキーモードには、数値入力モードと文字入力モードの 2 種類があります。

[S]キーを押下することで、数値入力モードと文字入力モードが順次切り替わります。

キーモードが文字入力モードの場合、ハードアイコンの S シンボルを表示します。

数値入力モード

0～9 の数値、小数点入力、+、-、入力の確定キーの入力が可能です。
ただし、+キーは、ファンクションキー(F1～F8)等にキーコード登録をした場合に入力できます。

文字入力モード

英字(A～Z、SP)、記号(-、$、/、+、%、:、＊)、数値(0～9、.)の入力が可能です。
英記号は、めくり入力です。

例）

ABC 2		キーを押すたびに“A”→”B”→”C”→”2”の順に候補を表示します。 入力は、ENT キーまたは他のキーの入力で確定します。 ただし内部処理コードの場合は除きます。

key_select 関数を使用して、アプリケーションからキーモードの取得と設定が可能です。

12.2 文字入力

12.2.1 1 文字入力

任意の位置で 1 文字の入力を行ないます。
アスキーコードが入力されるか、終了条件を検出するまで待機します。
アスキーコードが確定すると、アプリケーションにそのアスキーコードを返します。
エコーバックの指定がある場合は、指定の位置にエコーバックを行ないます。

12.2.2 文字列入力

任意の位置から右に指定文字数分を入力領域とし、文字列の入力を行ないます。
指定の領域にアスキーコードを格納し、確定キーまたは、終了条件を検出するまで待機します。
制御コードを入力した場合は、そのコードにしたがった処理を行ないます。

入力文字列の編集操作
文字列入力中は、次のキーにより入力文字列の編集操作を行なうことができます。
編集操作に使用するキーは入力文字としてあつかいません。(格納エリアには格納しません)
入力文字列の編集操作は入力領域中でのみ有効です。

名前	デフォルトキー	キーコード		機能	動作例	
		属性	コード		入力前	入力後
←	←(F2) or カーソルレフトキー	00h	1Dh	カーソルを 1 文字左へ移動します。	123 1234567890 1234 1	123 1234567890 1234 1
→	→(F3) or カーソルライトキー	00h	1Ch	カーソルを 1 文字右へ移動します。	123 1234567890 1234 1234	123 1234567890 1234 1234
クリア	クリア (CLR)	00h	0Ch	入力文字をすべて削除します。	123 1234567890 1234 1234	
後退	後退(BS)	00h	08h	カーソル前の 1 文字を削除します。	123 1234567890 abcd 12345	12 123456789a bcd 2345
削除	DEL(F4)	00h	10h	カーソル上の 1 文字を削除します。	123 1234567890 1234567890 abcd	13 123456789 1234567890 bcd

12.2.3 数値入力

任意の位置から右に指定文字数分を入力領域とし、入力領域の最右端から数値の入力を行ないます。
指定の領域に数値を格納し、確定キーまたは、終了条件を検出するまで待機します。
制御コードを入力した場合は、そのコードにしたがった処理を行ないます。
数値の入力は数値データのみ有効とし、めくり文字が入力された場合、数値データに変換して処理を行ないます。

入力文字列の編集操作

数値入力中は、次のキーにより入力文字列の編集操作を行なうことができます。
編集操作に使用するキーは入力文字としてあつかいません。(格納エリアには格納しません)
入力文字列の編集操作は入力領域中でのみ有効です。

名前	デフォルトキー	キーコード		機能	動作例	
		属性	コード		入力前	入力後
+ (プラス)	なし	00h	2Bh	"-"(マイナス記号)表示中の場合、 "-"を削除します。	-123 1 	123 1
ー (マイナス)	-(F3)	00h	2Dh	"-"(マイナス記号)を数値の最上位位置に付加します。 (入力領域がフルの場合、無効になります)	-123 1 1234567890	-123 -1 - 1234567890
. (小数点)	.(F4)	00h	2Eh	カーソル位置に"."(ピリオド)を付加します。 (入力領域がフルの場合と、すでに"."が付加されている場合、無効になります)	123 -123 123.4 1234567890	123. -123. . 123.4 1234567890
クリア	クリア(CLR)	00h	0Ch	入力文字をすべて削除します。	123 1 12345 1234567890	
後退	後退(BS)	00h	08h	カーソル前の1文字を削除します。	123 1 12345 1234567890	12 12345 6123456789
削除	DEL(F4)	00h	10h	カーソル上の1文字を削除します。	123 1 12345 1234567890	12 12345 6123456789

12.3 ファンクションキー制御

12.3.1 キーコードの設定

ストロークキー、マルチファンクションキー、およびトリガーキーに、任意のキーコードを設定することができます。キーコードの詳細は、KEYFORM 構造体を参照してください。

設定可能なキーコードの組み合わせを以下に示します。

	コード値		機能	1 文字入力	文字列入力	数値入力
	属性	コード				
機能コード	FFh	00h	コントラストを 1 段濃くします	×	×	×
		01h	コントラスト 1 段を淡くします			
		02h	バックライト ON/OFF 切替え			
		03h	バーコード読み込み開始※1			
制御コード	00h	08h	1 文字後退	○	×	×
		0Ah	改行			
		0Ch	入力領域のクリア			
		0Dh	復帰			
		10h	1 文字削除			
		1Ch	カーソル右移動			
		1Dh	カーソル左移動			
		1Eh	カーソル上移動			
		1Fh	カーソル下移動			
その他(ANK)	00h	XXh	文字※2	○	○	数字(0～9)+,-,.のみ

※1　バーコード読み込み開始機能はトリガーキー／マルチファンクションキーに対してのみ設定することが可能です。

※2　ANK コードおよび制御コードの設定が可能です。
設定可能なコード範囲は(01h～80h、A0h～DFh、FDh～FFh)です。
ただし文字列入力では制御コードは返りません。

キーコードを設定可能なキーを以下に示します。

	キー		設定可能	設定可能キーコード
ストロークキー	入力モード切替(S)		不可	ー
	後退(BS)			
	クリア(CLR)			
	テンキー 1	テンキー 2	不可	ー
	テンキー 3	テンキー 4		
	テンキー 5	テンキー 6		
	テンキー 7	テンキー 8		
	テンキー 9	テンキー 0		
	テンキー .	テンキー ENT		
	F1(-)	F2(←)	可能	バーコード読み込みを除くすべて
	F3(→)	F4(DEL)		
	F5(SP)	F6(▲)		
	F7(▼)	F8(BL)		
マルチファンクションキー	マルチファンクションキー R		可能	すべて
	マルチファンクションキー L			
トリガーキー	ライトトリガーキー		不可	ー
	レフトトリガーキー			
	センタートリガーキー			
カーソルキー	アップキー		不可	
	ダウンキー			
	ライトキー			
	レフトキー			

12.3.2 キー入力無効の設定

ストロークキー、マルチファンクションキー、およびトリガーキーに、キー入力を無効にすることができます。

キー入力有効/無効を設定可能なキーを以下に示します。

	キー		設定可能
ストロークキー	入力モード切替(S)		可能
	後退(BS)		
	クリア(CLR)		
	テンキー 1	テンキー 2	可能
	テンキー 3	テンキー 4	
	テンキー 5	テンキー 6	
	テンキー 7	テンキー 8	
	テンキー 9	テンキー 0	
	テンキー .	テンキー ENT	
	F1(-)	F2(←)	可能
	F3(→)	F4(DEL)	
	F5(SP)	F6(▲)	
	F7(▼)	F8(BL)	
マルチファンクションキー	マルチファンクションキー R		可能
	マルチファンクションキー L		
トリガーキー	ライトトリガーキー		不可
	レフトトリガーキー		
	センタートリガーキー		
カーソルキー	アップキー		不可
	ダウンキー		
	ライトキー		
	レフトキー		

※ キー入力を無効に設定した場合、キーを押してもキークリック音はなりません。

※ マルチファンクションキーに OBR 読込機能が登録時にキー入力を無効にした場合は、通常キーとして動作します。(OBR 読込は開始しません)

12.4 キーバッファ

キーバッファのサイズはバッファ内に格納できるキーコードの個数で、DT-970 では 128 キー固定としています。

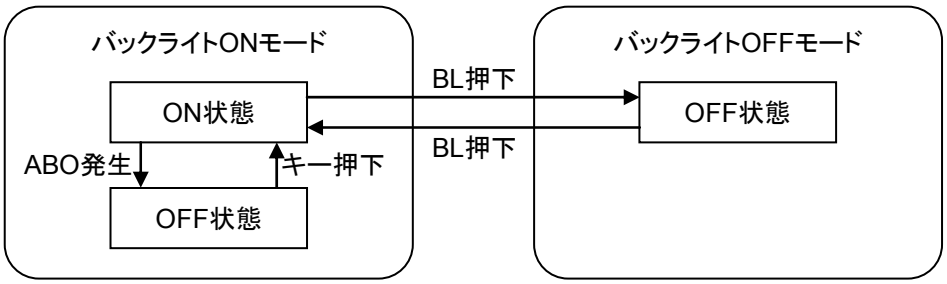
電源 OFF→ON(レジューム ON 立上げ)時、キーバッファはクリアされます。

ただし、めくり文字入力中の場合は、エコーバック表示とのずれを防ぐためクリアされません。

12.5 バックライト制御

バックライトは F8(BL)キーで ON/OFF できます。バックライト ON 後、一定時間キー操作が行なわれない場合は ABO(Auto Backlight Off)します。ABO でバックライトが消えた場合、次のキータッチでバックライトは ON します。
再度、F8(BL)キーの押下でバックライト OFF モードに移行します。

バックライト ON/OFF の状態遷移を以下に示します。



12.6 多点押し処理

DT-970 ではキーを OBR キーと通常キーとに分けています。
OBR キーとは、バーコード読み込みを開始するキーコードが設定されているキーのことです。
(トリガーキーR・L・C、マルチファンクションキーR・L に設定可能です。)
通常キーとはアスキーコードおよびコントラスト調整等が設定されているキーで OBR キー以外のことを指します。
OBR 登録キーが押された場合は、優先して実行されます。(OBR がオープン中の場合)

同時押し処理

- OBR キー同士の同時押し
  バーコード読み込みが開始されます。ただし、OBR オープン中のみで、OBR が未オープン時は無視されます。
- 通常キー同士の同時押し
  どちらか一方が離された時点で確定します。(ロールオーバー機能)
- OBR キーと通常キーの同時押し
  OBR キーが優先されバーコード読み込みが開始されます。ただし、OBR が未オープンの場合は、OBR キーは無視され通常キーが入力されます。

同時押し時の入力キー

押下キー	動作	備考
OBR キーと OBR キー	バーコード読み込み開始	
OBR キーと通常キー	バーコード読み込み開始	
通常キーと通常キー	未確定	どちらか一方を離した時点で確定
通常キーと無効キー	通常キー確定	
無効キーと無効キー	無視	

※ 無効キー:OBR キーであっても OBR が未オープン状態の場合

多点押し(順次押下)

- OBR キー押下後の OBR キー押下
  変化はありません。継続してバーコードを読み込みます。すべての OBR キーが離された時点でバーコード読み込みを中止します。
- OBR キー押下後の通常キー
  通常キーは無視されます。OBR キーが離された時点で通常キーは入力されます。
- 通常キー押下後の OBR キー押下
  バーコード読み込みを開始します。

多点押し時の入力キー

1 キー目	2 キー目	2 キー目の動作	備考
OBR キー	OBR キー	無視	バーコード読み込み継続
OBR キー	通常キー	無視(OBR キーリリース後確定)	バーコード読み込み継続
通常キー	OBR キー	バーコード読み込み開始	
通常キー	通常キー	未確定	ロールオーバー機能
通常キー	無効キー	無視	
無効キー	通常キー	キー確定	
無効キー	無効キー	無視	

※ 無効キー:OBR キーであっても OBR が未オープン状態の場合

キーロールオーバー

本キー関数は、通常キーに対してのみ 2 キーロールオーバー機能を有します。

(例1)
[1] キー押下(押したまま) → 1入力
↓
[2] キー押下(押したまま) → そのまま(2の入力は行われない)
↓
[1] キー解放 → 2入力

(例2)
[1] キー押下(押したまま) → 1入力
↓
[2] キー押下(押したまま) → そのまま(2の入力は行われない)
↓
[2] キー解放 → そのまま(1の入力は行われない)
↓
[2] キー押下(押したまま) → そのまま(2の入力は行われない)
↓
[2] キー解放 → そのまま(1の入力は行われない)

12.7 関数リファレンス

キーモード設定

関数	機能概要
key_select	キー入力モードの設定

キー入力

関数	機能概要
key_read	1 文字の入力
key_string	文字列の入力
key_num	数値の入力
key_check	キーバッファのステータスチェック
key_clear	キーバッファのクリア

ファンクションキー制御

関数	機能概要
key_fnc	ファンクションキーコードの設定

12.7.1 key_select

キー入力モードの取得、設定、および解除を行ないます。

```
ER key_select(
   UB          mode,
   KEYSEL      *key_sel
);
```

パラメータ

mode

キー入力モードの設定、解除、取得を次の値で指定します。

SEL_SET	:キー入力モードを設定します
SEL_GET	:キー入力モードを取得します
SEL_RES	:キー入力モードを解除します

key_sel

有効無効キーテーブルを格納する、KEYSEL 構造体のアドレスを指定します。

戻り値

関数が成功すると E_OK が返ります。失敗すると次のエラーが返ります。

E_PRM	:パラメータエラー

解説

mode パラメータに SEL_RES を指定した場合は、設定可能なすべてのキー入力が有効になります。

参照

KEYSEL 構造体

12.7.2 key_read

1 文字入力を行ないます。

```
ER key_read(
   KEY_INP    *pkey_inp
);
```

パラメータ

pkey_inp
1 文字入力情報を格納する KEYINP 構造体のアドレスを指定します。

戻り値

関数が成功すると入力した ANK コードが返ります。失敗または文字入力以外のイベントによって終了すると次のコードが返ります。

E_KEY_INT	:イベント通知キー押下検出
E_KEY_LB	:LB 発生検出
E_KEY_OBR	:バーコード読み込み完了検出
E_KEY_IO	:クレードル検出／USB 接続検出
E_PRM	:パラメータエラー

参照

KEY_INP 構造体

12.7.3 key_string

文字列入力を行ないます。

```
ER key_string(
   KEY_INPS   *pkey_inps,
   UB         *string
);
```

パラメータ

pkey_inps
文字列入力情報を格納する KEY_INPS 構造体のアドレスを指定します。

string
入力文字列を格納する領域のアドレスを指定します。

戻り値

関数が成功するとE_OKが返ります。失敗または文字入力以外のイベントによって終了すると次のコードが返ります。

E_KEY_INT	:イベント通知キー押下検出
E_KEY_LB	:LB 発生検出
E_KEY_OBR	:バーコード読み込み完了検出
E_KEY_CLR	:クリアキー押下検出
E_KEY_FUL	:入力領域フル終了
E_KEY_IO	:クレードル検出／USB 接続検出
E_PRM	:パラメータエラー

解説

string パラメータには、入力桁数+1 の領域サイズが必要です。

参照

KEY_INPS 構造体

12.7.4 key_num

数値文字列入力を行ないます。数値(0～9)および記号(+, -, .)以外は無視します。

```
ER key_num(
   KEY_INPS   *pkey_inps,
   UB         *string
);
```

パラメータ

pkey_inps
数値入力情報を格納する KEY_INPS 構造体のアドレスを指定します。

string
入力文字列を格納する領域のアドレスを指定します。

戻り値

関数が成功するとE_OKが返ります。失敗または文字入力以外のイベントによって終了すると次のコードが返ります。

E_KEY_INT	:イベント通知キー押下検出
E_KEY_LB	:LB 発生検出
E_KEY_OBR	:バーコード読み込み完了検出
E_KEY_CLR	:クリアキー押下検出
E_KEY_FUL	:入力領域フル終了
E_KEY_IO	:クレードル検出／USB 接続検出
E_PRM	:パラメータエラー

解説

string パラメータには、入力桁数+1 の領域サイズが必要です。

参照

KEY_INPS 構造体

12.7.5 key_check

キーバッファの先頭に格納されているキーコードを読み出します。キーバッファの読み込み位置は更新しません。

```
ER key_check ();
```

パラメータ

ありません。

戻り値

キーバッファにデータがある場合には、そのデータの ANK コードが返ります。
キーバッファにデータがない、または入力途中の場合は、次のコードが返ります。

E_KEY_MD	:入力途中(アルファベット記号入力中です)
E_NG	:データなし

12.7.6 key_clear

キーバッファをクリアします。

```
ER key_clear ();
```

パラメータ

ありません。

戻り値

関数が成功すると E_OK が返ります。

12.7.7 key_fnc

ファンクションキーおよびマルチファンクションキーの、キーコードの取得と設定を行ないます。

```
ER key_fnc(
   UB          func_mode,
   UB          func_num,
   KEYFORM     *func_data
);
```

パラメータ

func_mode

キーコードを取得するか、設定するかを次の値で指定します。

FNC_GET	:キーコードを取得します。
FNC_SET	:キーコードを設定します。

func_num

取得／設定対象のファンクションキー番号を次の値で指定します。

FNC_1	:ファンクションキー1
FNC_2	:ファンクションキー2
FNC_3	:ファンクションキー3
FNC_4	:ファンクションキー4
FNC_5	:ファンクションキー5
FNC_6	:ファンクションキー6
FNC_7	:ファンクションキー7
FNC_8	:ファンクションキー8
MLT_R	:マルチファンクションキーR
MLT_L	:マルチファンクションキーL

func_data

キーコードデータを格納する KEYFORM 構造体のアドレスを指定します。

戻り値

関数が成功すると E_OK が返ります。失敗すると次のエラーが返ります。

E_PRM	:パラメータエラー

参照

KEYFORM 構造体

13.OBR 制御

DT-970 OBR について説明します。

13.1 OBR 基本仕様

OBR の基本仕様について説明します。

レーザースキャナ性能

項目	仕様
発行素子	赤色半導体レーザー
走査方式	往復振動ミラー
走査回数	100±20scan/sec
レーザー光走査角度	50±5deg
読み取り角度	40deg

読み取り可能コード

WPC	JAN 規格	JIS X0501 JAN-13,JAN-8 JAN-13 addon(+2,+5), JAN-8 addon(+2,+5)
	EAN 規格	General Specification for the Article Symbol Marking EAN-13, EAN-8 EAN-13 addon(+2,+5), EAN-8 addon(+2,+5)
	UPC 規格	UPC Symbol Specification Jan 1986 UPC-A, UPC-E UPC-A addon(+2,+5), UPC-E addon(+2,+5)
CODE-39		
Codabar (NW-7)		
Interleaved 2of5		
Industrial 2of5		
CODE-93		
CODE-128		
MSI		
IATA		
GS1 DataBar Omnidirectional / GS1 DataBar Truncated (RSS-14)		
GS1 DataBar Limited (RSS Limited)		
GS1 DataBar Expanded (RSS Expanded)		
GS1 DataBar Stacked / GS1 DataBar Stacked Omnidirectional (RSS-14 Stacked)		
GS1 DataBar Expanded Stacked (RSS Expanded Stacked)		

読み取り桁数と出力フォーマット

バーコード	規格	桁数	出力フォーマット	備考
WPC	JAN-13	13	FFMMMMMNNNNNC」	F: カントリーフラグ M 生産者コード : N: 商品コード S: システムメンバーキャラクタ A: addon データ 」: 終了コード (CR、LF、または CRLF) C: チェックデジット(mod 10) UPC-B を除きチェクデジット(mod 10)の計算は必ず行います。
	EAN-13	13	FFMMMMMNNNNNC」	
	JAN-8	8	FFMMMNC」	
	EAN-8	8	FFMMMNC」	
	JAN-13 addon+2	15	FFMMMMMNNNNNCAA」	
	EAN-13 addon+2	15	FFMMMMMNNNNNCAA」	
	JAN-13 addon+5	18	FFMMMMMNNNNNCAAAAA」	
	EAN-13 addon+5	18	FFMMMMMNNNNNCAAAAA」	
	JAN-8 addon+2	10	FFMMMMNCAA」	
	EAN-8 addon+2	10	FFMMMMNCAA」	
	JAN-8 addon+5	13	FFMMMMNCAAAAA」	
	EAN-8 addon+5	13	FFMMMMNCAAAAA」	
	UPC-A	12	0SMMMMMNNNNNC」	
	UPC-A addon+2	14	0SMMMMMNNNNNCAA」	
	UPC-A addon+5	17	0SMMMMMNNNNNCAAAAA」	
	UPC-E※	(7),8	0MMNNNMC」	最後の M: 0～2
		(7),8	0MMMNN3C」	
		(7),8	0MMMMN4C」	
		(7),8	0MMMMMNC」	最後の N: 5～9
	UPC-E addon+2 ※	(9),10	0MMNNNMCAA」	最後の M: 0～2
		(9),10	0MMMNN3CAA」	
		(9),10	0MMMMN4CAA」	
		(9),10	0MMMMMNCAA」	最後の N: 5～9
	UPC-E addon+5 ※	(12),13	0MMNNNMCAAAAA」	最後の M: 0～2
		(12),13	0MMMNN3CAAAAA」	
		(12),13	0MMMMN4CAAAAA」	
		(12),13	0MMMMMNCAAAAA」	最後の N: 5～9
	UPC-E(+UPC-A) ※	6+12	MMNNNMC」SMMM0000NNNC」	6 桁目の M: 0～2
		6+12	MMMNN3C」SMMM00000NNC」	
		6+12	MMMMN4C」SMMMM00000NC」	
		6+12	MMMMMNC」SMMMMM0000NC」	6 桁目の: 5～9
	JAN-13	14	0FFMMMMMNNNNNC」	GTIN
	EAN-13	14	0FFMMMMMNNNNNC」	GTIN
	JAN-8	14	0000000FFMMMNC」	GTIN
	EAN-8	14	0000000FFMMMNC」	GTIN
	UPC-A	14	00SMMMMMNNNNNC」	GTIN
	UPC-E	14	0000000MMNNNMC」	GTIN 最後の M: 0～2
		14	0000000MMMNN3C」	GTIN
		14	0000000MMMMN4C」	GTIN
		14	0000000MMMMMNC」	GTIN 最後の N: 5～9
Code39		3～50	SBBB・・・・BBCS」	A: データ B: FULL ASCII 変換後のデータ C: チェックデジット(mod43)
		3～50	SAAA・・・・AACS」	
		1～48	BBB・・・・・BBC」	

		1～48	AAA……AAC」	チェックデジットなしの場合、 データとなります。 S: スタート・ストップキャラクタ
NW-7		3～40	SDDD……DDDE」	S: スタートコード (A,B,C,D のいずれか) E: エンドコード (A,B,C,D のいずれか) D: データ
		1～38	DDD……DDD」	
Interleaved 2 of 5		2～40	DDD……DDDC」	D: データ C: チェックデジット(mod 10) チェックデジットなしの場合、 データとなります。 読み取り桁数は偶数桁のみ
Industrial 2of5		2～40	DDD……DDDC」	D: データ C: チェックデジット(mod 10) チェックデジットなしの場合、 データとなります。
Code93		1～40	DDD……DDD」	D: データ
Code128		1～64	AAA……AAA」	A: ASCII 変換後データ B: ASCII 変換前データ C: チェックデジット(mod47) チェックデジットなしの場合、 データとなります。
		1～64	SBBB……BBCS」	
MSI		2～40	DDD……DDCC」	D: データ C: チェックデジット (mod10,mod10) チェックデジットなしの場合、 データとなります。
IATA		2～40	PADDD……DDDDC」	P: クーポン NO. A: エアライン NO. D: データ C: チェックデジット(IATA 仕様) チェックデジットなしの場合、 データとなります。
RSS-14		16 14	01DDDDDDDDDDDDDC」 DDDDDDDDDDDDDC」	D: 数字データ C: チェックデジット(mod 10)
RSS Limited		16 14	01DDDDDDDDDDDDDC」 DDDDDDDDDDDDDC」	D: 数字データ C: チェックデジット(mod 10)
RSS Expanded		1～74 1～41	DDD……DDD」 AAA……AAA」	D: 数字データ A: アルファベットデータ
RSS-14 Stacked		16 14	01DDDDDDDDDDDDDC」 DDDDDDDDDDDDDC」	D: 数字データ C: チェックデジット(mod 10)
RSS Expanded Stacked		1～74 1～41	DDD……DDD」 AAA……AAA」	D: 数字データ A: アルファベットデータ

※ 読み取り桁数が、カッコの桁の場合は、出力フォーマットに"C"を付加しません

13.2 OBR 制御機能

レーザーを点灯し、バーコードの読み取りができる読み取り可能状態と、レーザーを消灯し、バーコードの読み取りができない読み取り待機状態の切り替えを行ないます。
また、現在の状態を参照することができます。

開始処理は読み取りコードの設定を行なうことも可能です。読み取りコードの設定についての詳細は設定を参照してください。

1 文字の読み込み
OBR_getc 関数を使用して、OBR バッファから 1 文字を読み出します。

文字列の読み込み
OBR_gets 関数を使用して、OBR バッファから 1 ラベル（コード）の文字列を読み出します。

OBR データバッファの状態チェック
OBR_stat 関数を使用して、OBR バッファ内の残りバイト数と残り段数を取得することができます。

OBR データバッファのクリア
OBR_flush 関数を使用して、OBR バッファのクリアを行ないます。

格納先バッファの切り替え
OBR_chgbuf 関数を使用して、バーコードデータの格納先を OBR バッファとキーバッファに切り替えることができます。
格納先をキーバッファに変更することで、読み取りデータをキー入力と同時に扱うことができます。
初期状態は、OBR バッファに設定しています。

※ OBR バッファとは、バーコードの読み取りデータを専用に格納するバッファで、バーコード 9 個分のサイズがあります。

13.2.1 動作モードの設定

OBR_modewt 関数と OBR_moderd 関数を使用して、OBR の種々の動作モード設定と取得を行うことができます。

読み取り可能コード種別

特定のコードしか読み取る必要がない場合は、必要なコードのみを読み取り対象に設定します。
必要なコードのみを読み取り対象とすることで、デコード処理の処理時間を短縮できます。

設定条件	コード	備考
自動区別 （すべてのコードを選択した場合）	NW-7 CODE-39 Industrial 2of5 Interleaved 2of5 CODE-93 CODE-128 MSI WPC（UPC-E 以外） addon +2(5) WPC（UPC-E 以外） UPC-E addon +2(5) UPC-E IATA RSS-14 RSS-14 Stacked RSS Limited RSS Expanded RSS Expanded Stacked	デコード優先順位 高 ↑ ↓ 低
コード限定	CODE-39 NW-7 WPC（UPC-E 以外） addon +2(5) WPC（UPC-E 以外） UPC-E addon +2(5) UPC-E Industrial 2of5 / IATA Interleaved 2of5 CODE-93 CODE-128 MSI RSS-14 / RSS-14 Stacked RSS Limited RSS Expanded / RSS Expanded Stacked	複数設定可能 複数コードを設定した場合の優先順位は、上段のとおり

※ EAN128 を読み取る場合は、読み取り可能コードに CODE128 を指定し、出力フォーマット指定に EAN128 出力を指定してください。

読み取り桁数

読み取り対象のコードごとに、読み取り桁数の有効範囲を設定します。

読み取り桁数の設定範囲は次のとおりです。

コード	設定範囲 （単一コード設定）	設定範囲 （複数コード設定）	備考
WPC	桁数固定	同左	桁数固定のため、設定できません
CODE-39	1～48 桁	2～48 桁	スタート／ストップキャラクタを含みません
NW-7	1～38 桁	2～38 桁	スタート／ストップキャラクタを含みません
Industrial 2of5	2～40 桁	同左	
Interleaved 2of5	2～40 桁	4～40 桁	
CODE-93	1～40 桁	同左	
CODE-128	1～64 桁	同左	
MSI	2～40 桁	同左	
IATA	2～40 桁	同左	
RSS-14	桁数固定	同左	桁数固定のため、設定できません
RSS Limited	桁数固定	同左	桁数固定のため、設定できません
RSS Expanded	1～74 桁	同左	
RSS-14 Stacked	桁数固定	同左	桁数固定のため、設定できません
RSS Expanded Stacked	1～74 桁	同左	

※ 複数のコードを同時に設定した場合、CODE-39、 NW-7、 Interleaved 2of5 は、誤読防止のため桁数の設定可能範囲が変化します。

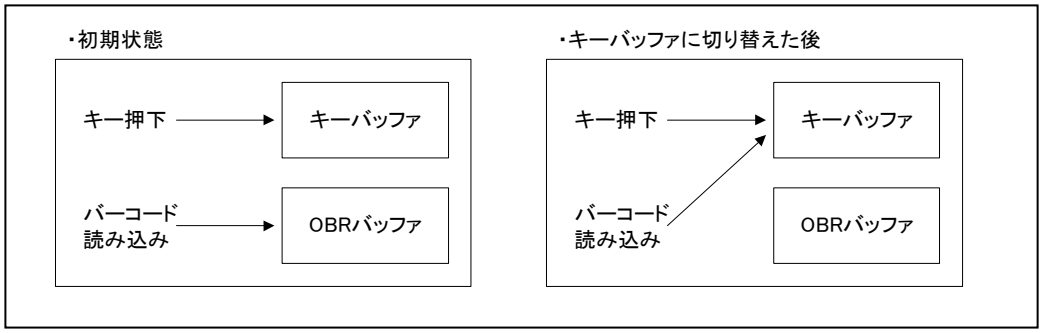
※ Interleaved 2of5 で奇数桁の指定をした場合、最小桁は指定+1 の偶数まで、最大桁は指定-1 の偶数までが読み取り可能となります。

このため、最大最小桁に同一の奇数を指定した場合、何も読み取れなくなります。

※ CODE-39 の 1 桁、NW-7 の 1 桁、Interleaved 2of5 の 2 桁を読み取る場合は、コード限定の読み取りを指定する必要があります。

出力バッファ

バーコードデータの格納先を OBR バッファまたはキーバッファに切り替えることができます。
格納先をキーバッファに変更することで、読み取りデータをキー入力と同様に扱うことができます。
初期状態は、OBR バッファに設定されています。



バッファ内にデータが残っている場合に、出力バッファを切り替えた場合の、データのあつかいは次のとおりです。

OBR バッファ	保存します。 バッファ内のデータを使用しない場合は、切り替え後に OBR バッファをクリアして下さい。
キーバッファ	保存します。

出力フォーマット

次に記載するバーコードの種類は、出力フォーマットの設定が可能です。

バーコードの種類	設定内容	初期状態
CODE-39	スタート／スキップキャラクタの出力の有無を設定	出力有り
	Full ASCII 変換の有無を設定	変換無し
NW-7	スタート／スキップキャラクタの出力の有無を設定	出力有り
UPC-E	UPC-A の復元コードの出力の有無を設定	出力無し
CODE-128	変換前データ／変換後データ(ASCII)のどちらを出力するかを設定	変換後データ

終了コード

バーコードデータの最後につける制御コードを次の 3 種類から選択できます。初期状態は CR です。

- CR
- LF
- CR+LF

チェックキャラクタの出力

次のバーコードは、チェックキャラクタの出力する／しないを変更することできます。

- CODE-39
- UPC-E
- Industrial 2 of 5
- Interleaved 2 of 5

読み取り方法
バーコードの読み取り方法を、“単発読み”／“連続読み”から選択することができます。初期状態は“連続読み(トリガーキーあり)”です。

読み取り方法	内容	読み取り終了条件
単発読み	トリガーキーを押下すると読み取り可能となり、読み取り完了後に待機状態となります。	スキャン時間経過 読み取り完了
連続読み (トリガーキーあり)	トリガーキーを押下している間、常に読み取り可能状態となります。	前コード読み取り完了後にスキャン時間経過 指定読み取り回数分の読み取り完了 トリガーキー離し

スキャン時間
トリガーキーを押下した後の読み取り可能時間を設定できます。
設定した時間を経過すると、自動的に読み取り待機状態となります。
1～9 秒まで設定することが可能です。
スキャン時間は、クローズ中(OBR_open 実行前)に dat_system 関数を使用して設定します。

読み取り回数
連続読みの場合の読み取り可能回数を設定できます。
設定した回数分読み取りを完了すると、自動的に読み取り待機状態となります。
1～9 回まで設定することが可能です。
読み取り回数は、クローズ中(OBR_open 実行前)に dat_system 関数を使用して設定します。

照合回数
読み取ったデータに対する信頼性を向上するための照合回数を設定できます。
照合回数をもとに内部で設定した回数の読み取りを行ない照合します。
1～9 回まで設定することが可能です。
照合回数は、クローズ中(OBR_open 実行前)に dat_system 関数を使用して設定します。

チェックデジットの計算
チェックキャラクタと、コードごとの計算方式の結果を照合します。
コードごとにチェックデジット計算の有効／無効を設定することができます。
初期状態は“有効”です。

同一ラベルの二度読み防止
読み取り回数を 2 回以上に設定し、連続読みにて読み取りを行なっている場合、1 回の読み取り中(1 回のトリガキー押下)に同一ラベルを連続して読むことはできません。

ブザー制御
1 コードごとの読み取り完了を、ブザー音によって通知することができます。
ブザー制御を無効にすることも可能です。
※ システム全体としてのブザーの音量は"動作環境メニュー"または、dat_system 関数によって設定することができます。システム全体の音量が OFF になっている場合、ブザー音による通知を設定してあっても音はなりません。

LED 制御

1 コードごとの読み取り完了を LED の点灯によって通知することができます。

- 読み取りコードが正常な場合、LED を一定秒間緑色に点灯したのち消灯します。
- 読み取りコードが異常な場合は、LED を点灯しません。

LED 制御を無効にすることも可能です。

読み取りコードが異常になる要因には、次のものがあります。

- 指定した桁数の範囲外のバーコードを読み取った
- チェックデジッド指定時のチェックデジッドエラー
- CODE-39、CODE-93 における Full ASCII 変換エラー

バイブレータ制御

1 コードごとの読み取り完了を、バイブレータの振動によって通知することができます。
また、バイブレータ制御を無効にもできます。

- 読み取りコードが正常な場合、バイブレータを振動させます。
- 読み取りコードが異常な場合は、バイブレータを振動させません。

読み取り動作

通常読み： オープン後、クローズするまで連続して読み取りが行なわれます。
段数読み： オープン後、指定された回数分の読み取りが行なわれます。

立ち上げモード

立ち上げモードとは、トリガキー押下により電源を ON するかしないかのことです。
OBR では次に記載する各モードによりこの立上げモード選択できます。

立上げモード一覧

モード / OBR の状態	0	1	2
OPEN状態	立ち上げ不可能	立ち上げ可能	立ち上げ可能
CLOSE状態	立ち上げ不可能	立ち上げ可能	立ち上げ不可能

13.2.2 レーザー発光幅の制御

遠距離のバーコードスキャンを行なうと、レーザーの走査幅が広がり、読み取り対象バーコード以外の複数のバーコードをスキャンしてしまうことがあります。

DT-970 では、読み取り対象バーコードの幅にあわせてレーザーの照射を変更することにより、確実にバーコードを読み取ることができます。

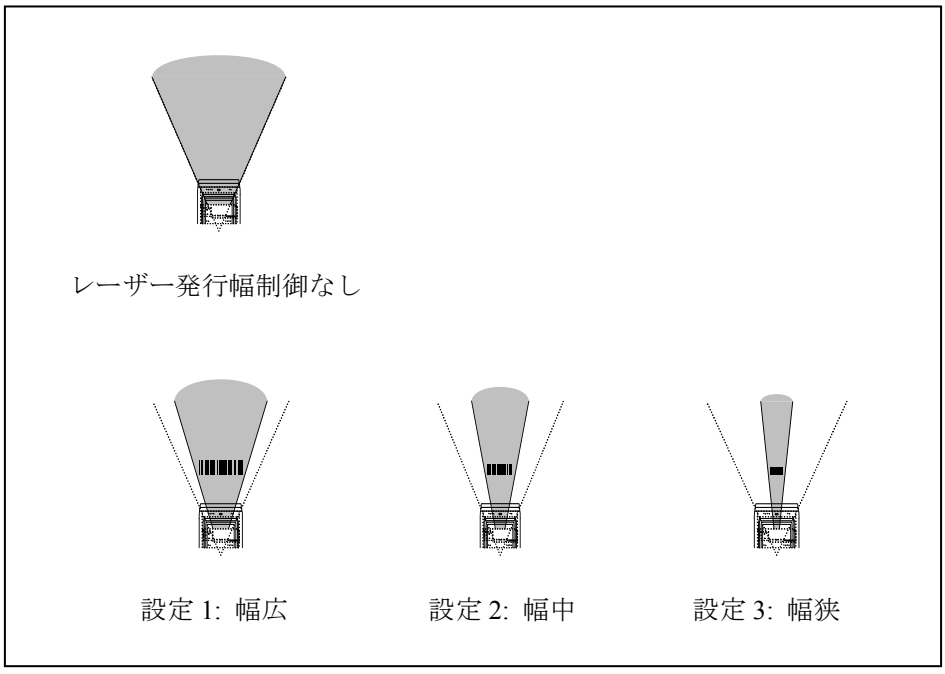
レーザー発光幅制御の機能を以下に示します。

機能	内容
キャリブレーション	DT-970 個体差によるレーザー発光幅位置を補正します。
レーザー発光幅設定	レーザー発光幅を 4 段階で設定します。

キャリブレーション （wkup_calib 関数）
DT-970 の個体差によるレーザー発光幅位置（レーザーの発光幅）を調節します。

キャリブレーション（wkup_calib 関数）は電源イベントを処理しません。そのため、アプリケーションプログラムが電源イベントの通知を有効に設定した状態で呼び出すと、電源キーを押してもオフできないなどの問題が発生します。そのため、呼び出しに際しては電源イベントの通知を解除してください。

レーザー発光幅設定機能 （OBR_swing 関数）
レーザー発光幅を 4 段階で設定します



※ リセットにより、レーザー発光幅制御量は"間口幅制御なし"に設定されます。

レーザー発光幅制御量	レーザー発光幅
発行幅制御なし	40°
幅広	32°
幅中	24°
幅狭	16°

13.3 関数リファレンス

ファンクション詳細を次ページより示します。

関数	機能概要
OBR_open	OBR のオープン
OBR_close	OBR のクローズ
OBR_getc	OBR データ 1 文字のリード
OBR_gets	OBR データ文字列のリード
OBR_flush	OBR バッファのクリア
OBR_stat	OBR バッファステータスのチェック
OBR_moderd	OBR 動作モードの取得
OBR_modewt	OBR 動作モードの設定
OBR_chgbuf	OBR バッファの切り替え
OBR_trigmode	トリガーキーによる電源 ON 設定
OBR_swing	レーザー発光幅の設定／参照
OBR_widenarrow	サポートしません
OBR_getadjust	サポートしません
OBR_setadjust	サポートしません
OBR_getmargincheck	マージンチェック倍率の取得
OBR_setmargincheck	マージンチェック倍率の設定
OBR_getfocus	レーザフォーカスモードの取得
OBR_setfocus	レーザフォーカスモードの設定

13.3.1 OBR_open

OBR のオープンを行ないます。

```
ER OBR_open(
  UW  mode
);
```

パラメータ

mode

オープンモード(次の項目の論理和で指定)

■ 使用コード

OBR_CD39	:CODE39
OBR_NW_7	:NW-7
OBR_WPCA	:WPC(UPC-E 以外)addon
OBR_WPC	:WPC(UPC-E 以外)
OBR_UPEA	:UPC-E addon
OBR_UPE	:UPC-E
OBR_IDF / OBR_IATA	:Industrial 2of5 と IATA
OBR_ITF	:Interleaved 2of5
OBR_CD93	:CODE93
OBR_CD128	:CODE128
OBR_MSI	:MSI
OBR_RSS14 / OBR_RSS14S	:RSS-14 と RSS-14 Stacked
OBR_RSSLTD	:RSS Limited
OBR_RSSEXP / OBR_RSSEXPS	:RSS Expanded と RSS Expanded Stacked

■ チェックデジット実行指示

OBR_CHK_ON

■ チェックキャラクタ出力指示

OBR_OUT_ON

※ 現在設定されている動作モードでオープンする場合は、オープンモードを"0"にします。

※ オープンモードが"0"以外の場合、動作モードは初期状態になります。
(使用コードのチェックデジット／チェックキャラクタ以外の項目は初期状態になります。初期状態については、OBR_modewt 関数を参照してください。)

※ 本関数で設定できない項目は、動作モードの設定関数で設定してください。
読み取り回数、照合回数、読み取り時間は、システムデータ管理関数で設定してください。

(9) Industrial 2of5 と IATA、RSS-14 と RSS-14 Stacked、RSS Expanded と RSS Expanded Stacked の制御は共通です。別々に制御することはできません。

戻り値

E_OK	:正常終了
E_OBR_PON	:オープン済み

13.3.2 OBR_close

OBR の終了処理を行ないます。

```
ER OBR_close();
```

パラメータ

なし

戻り値

E_OK	:正常終了
E_OBR_POF	:クローズ済み

13.3.3 OBR_getc

OBR データを 1 文字読み込ます。

```
UB OBR_getc(
  UW  *rcd
);
```

パラメータ

rcd

読み取りコード格納先へのポインタ

OBR_NONDT	:データなし
OBR_CD39	:CODE39
OBR_NW_7	:NW-7
OBR_WPCA	:WPC(UPC-E 以外)addon
OBR_WPC	:WPC(UPC-E 以外)
OBR_UPEA	:UPC-E addon
OBR_UPE	:UPC-E
OBR_IDF	:Industrial 2of5
OBR_ITF	:Interleaved 2of5
OBR_CD93	:CODE93
OBR_CD128	:CODE128
OBR_MSI	:MSI
OBR_IATA	:IATA
OBR_RSS14	:RSS-14 または RSS-14 Stacked
OBR_RSSLTD	:RSS Limited
OBR_RSSEXP	:RSS Expanded または RSS Expanded Stacked

戻り値

OBR データ(1 文字)

13.3.4 OBR_gets

OBR データを文字列で読み込ます。

```
ER OBR_gets(
   UB     *buff,
   UW     *rcd,
   UB     *lengs
);
```

パラメータ

buff
OBR データ出力先へのポインタ

rcd
読み取りコード格納先へのポインタ

OBR_NONDT	:データなし
OBR_CD39	:CODE39
OBR_NW_7	:NW-7
OBR_WPCA	:WPC(UPC-E 以外)addon
OBR_WPC	:WPC(UPC-E 以外)
OBR_UPEA	:UPC-E addon
OBR_UPE	:UPC-E
OBR_IDF	:Industrial 2of5
OBR_ITF	:Interleaved 2of5
OBR_CD93	:CODE93
OBR_CD128	:CODE128
OBR_MSI	:MSI
OBR_IATA	:IATA
OBR_RSS14	:RSS-14 または RSS-14 Stacked
OBR_RSSLTD	:RSS Limited
OBR_RSSEXP	:RSS Expanded または RSS Expanded Stacked

lengs
データバイト数格納先へのポインタ

戻り値

E_OK	:正常終了

13.3.5 OBR_flush

OBR バッファのクリアを行ないます。
（OBR バッファの管理情報のみをクリアし、バッファ内のデータはクリアしません）

```
ER OBR_flush();
```

パラメータ

なし

戻り値

E_OK　　:正常終了

13.3.6 OBR_stat

OBR バッファの状態を読み込ます。

```
ER OBR_stat(
   W      */eng ,
   UB     */cnt
);
```

パラメータ

leng
バッファ内の残りバイト数格納先へのポインタ

lcnt
バッファ内の残り段数格納先へのポインタ

戻り値

E_OK　　:正常終了

13.3.7 OBR_moderd

OBR の動作モードを読み込ます。

```
ER OBR_moderd(
   M_TBL  *modtbl
);
```

パラメータ

modtbl
動作モードテーブル格納エリアへのポインタ

戻り値

E_OK　　:正常終了

参照

M_TBL 構造体

13.3.8 OBR_modewt

OBR の動作モードを設定します。
読み取り誤動作を防止するため、OBR オープン中の動作モード設定は禁止します。
また、動作モード設定時、OBR バッファ内にデータが残っていてはいけません。
設定パラメータ内にエラーを発見した場合、そのパラメータについては無効となりますが、引き続きパラメータ設定を行ないます。
（パラメータ内に 1 つ以上エラーがあった場合は、E_PRM とします。）

```
ER OBR_modewt(
   M_TBL  *modtbl
);
```

パラメータ

modtbl
動作モードテーブル格納エリアへのポインタ

戻り値

E_OK	:正常終了
E_PRM	:パラメータエラー
E_OBR_PON	:オープン済み

解説

本関数は、OBR_moderd 関数で現在設定を読み込んで、それに対して変更をかけるようにしてください。
本関数で設定した内容を有効にするには、OBR_open 関数のパラメータに"0"を指定してください。
OBR_open(0);
本関数は、OBR_open 関数をコールする前に実行してください。

参照

M_TBL 構造体

13.3.9 OBR_chgbuf

OBR データの出力先を、OBR バッファまたは、KEY バッファのどちらかに切り替えます。

```
ER OBR_chgbuf(
   UB      buftype
);
```

パラメータ

buftype
バッファ種別
OBR_BUFOBR　　:OBR バッファ
OBR_STOFF　　:KEY バッファ

戻り値

E_OK　　:正常終了
E_PRM　　:パラメータエラー

13.3.10 OBR_trigmode

トリガーキーによる電源 ON モードの設定を行ないます。

```
ER OBR_trigmode(
   UH      trigmode
);
```

パラメータ

trigmode
OBR 立上げモード設定
OBR_TRIG0　　:モード 0
OBR_TRIG1　　:モード 1
OBR_TRIG2　　:モード 2

モード / 状態	0	1	2
オープン	×	○	○
クローズ	×	○	×

戻り値

E_OK　　:正常終了
E_PRM　　:パラメータエラー

13.3.11 OBR_swing

3 段階の発光幅を設定します。
発光幅制御なしの設定も可能です。
現在の設定状態を確認できます。

```
ER OBR_swing(
  H     mode,
  H     *pattern
);
```

パラメータ

mode
機能選択
SW_SET :設定
SW_READ :取得

pattern
設定状態を格納するエリアのアドレス
SW_RESET :設定解除
SW_SET1 :発光幅 1(幅広)
SW_SET2 :発光幅 2(幅中)
SW_SET3 :発光幅 3(幅狭)

戻り値

E_OK :正常終了
E_PRM :パラメータエラー

解説

設定したレーザー発光幅の段階は、バーコードの OPEN／CLOSE 時にも保存されます。
リセットを行なうと、設定は"設定解除"になります。

13.3.12 OBR_widenarrow

設定されているレーザー発光幅の微調整を行ないます。

```
ER OBR_widenarrow(
  UH    mode
);
```

パラメータ

mode
機能選択
SW_WIDE　　:広くする
SW_NARROW　:狭める

戻り値

E_OK　　:正常終了

解説

設定した微調整の値は、レーザー発光幅設定を行なうか、リセットを行なうとクリアされます。

注意

DT-970 では本関数はサポートしません。

13.3.13 OBR_getadjust

設定されているバー幅補正値を取得します。

```
ER OBR_getadjust(
 UH *padjust
);
```

パラメータ

padjust
設定状態を格納するエリアのアドレス
OBR_BAR_NORMAL　:補正なし
OBR_BAR_BLACK0　:黒バーを細らせる
OBR_BAR_BLACK1　:黒バーを太らせる
OBR_BAR_WHITE0　:白バーを細らせる
OBR_BAR_WHITE1　:白バーを太らせる

戻り値

E_OK　　:正常終了

注意

DT-970 では本関数はサポートしません。

13.3.14 OBR_setadjust

バー幅補正値を設定します。

```
ER OBR_setadjust(
   UH     adjust
);
```

パラメータ

adjust
設定状態を格納するエリアのアドレス
OBR_BAR_NORMAL　:補正なし
OBR_BAR_BLACK0　:黒バーを細らせる
OBR_BAR_BLACK1　:黒バーを太らせる
OBR_BAR_WHITE0　:白バーを細らせる
OBR_BAR_WHITE1　:白バーを太らせる

戻り値

E_OK　　:正常終了
E_PRM　　:パラメータエラー
E_OBR_PON　　:オープン済

注意

DT-970 では本関数はサポートしません。

13.3.15 OBR_getmargincheck

設定されているマージンチェックの倍率を取得します。

```
ER OBR_getmargincheck(
   UH     *pratio
);
```

パラメータ

pratio
設定状態を格納するエリアのアドレス
OBR_MARGIN_WIDE　　:マージンチェック倍率大
OBR_MARGIN_MIDDLE　:マージンチェック倍率中
OBR_MARGIN_NARROW　:マージンチェック倍率小
OBR_MARGIN_MINIMUM　:マージンチェック倍率最小
OBR_MARGIN_NONE　　:マージンチェックなし

戻り値

E_OK　　:正常終了

13.3.16 OBR_setmargincheck

マージンチェックの倍率を設定します。

```
ER OBR_setmargincheck(
   UH     ratio
);
```

パラメータ

pratio
マージンチェックの倍率設定
OBR_MARGIN_WIDE　　:マージンチェック倍率大
OBR_MARGIN_MIDDLE　:マージンチェック倍率中
OBR_MARGIN_NARROW　:マージンチェック倍率小
OBR_MARGIN_MINIMUM　:マージンチェック倍率最小
OBR_MARGIN_NONE　　:マージンチェックなし

戻り値

E_OK　　:正常終了
E_PRM　　:パラメータエラー
E_OBR_PON　　:オープン済

注意

デフォルトは OBR_MARGIN_WIDE です。マージンが狭いコードを読む場合は、小さい倍率を指定してください。

13.3.17 OBR_getfocus

設定されているレーザフォーカスモードを取得します。

```
ER OBR_getfocus(
   UH     *pfocusmode
);
```

パラメータ

pfocusmode
レーザフォーカスモード状態を格納するエリアのアドレス
OBR_FOCUSOFF　　:レーザーフォーカス OFF
OBR_FOCUSON　　:レーザーフォーカス ON

戻り値

E_OK　　:正常終了

13.3.18 OBR_setfocus

レーザフォーカスモードを設定します。

```
ER OBR_setfocus(
   UH     focusmode
);
```

パラメータ

focusmode
レーザフォーカスモード設定
OBR_FOCUSOFF　　:レーザーフォーカス OFF
OBR_FOCUSON　　:レーザーフォーカス ON

戻り値

E_OK　　:正常終了
E_PRM　　:パラメータエラー
E_OBR_PON　　:オープン済

14.HID 通信制御

14.1 使用方法

HID インタフェースの使用方法に関して説明します。
使用方法としては、「HID の経路を選択」→「データ出力」となります。

14.1.1 HID 経路

HID の経路は下記 3 つあり、接続する経路を選択することにより HID を使用可能にします。
⑼ Bluetooth
⑼ USB（クレードル接続）
⑼ USB（ケーブル接続）

経路の選択方法
・ Bluetooth　　BT_SelectProfile(BT_PROFILE_HID) 関数を使用します。
・ USB　　USB_setClientmode(port , USB_CLIENT_HID) 関数を使用します。
　　port　　: USB_PORT_CRADLE（クレードル経由の場合）
　　: USB_PORT_CABLE（ケーブル接続の場合）

14.1.2 データ出力

・ Bluetooth　　BT_HID_Keycode(Send_Code) 関数を使用します。
・ USB　　c_dout(COM8, Send_Code, 8) 関数を使用します。
　　※ HID は COM8 に割り当てられます

⑼ 出力するデータ形式
経路を選択後、Bluetooth または USB(COM8)を OPEN し、相手と接続した後で、HID 規格に従った形式で出力してください。（8 キャラクタ単位で出力します）

例:「1」キーの入力
```
unsigned char Send_Code[] = { 0x00, 0x00, 0x1E, 0x00, 0x00, 0x00, 0x00, 0x00  };
```

例:「abcdef」キーの入力
```
unsigned char Send_Code[] = { 0x00, 0x00, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09  };
```

例:キーが離された
```
unsigned char Send_Code[] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00  };
```

14.1.3 スキャナおよびキー入力データの HID への出力方法

どちらの場合も、一旦、入力されたデータを APL で受け、上記出力形式に変換後、HID として、対象の経路へ出力してください。

14.2 USB HID 関数リファレンス

ファンクション詳細を次ページより示します。

関数	機能概要
c_open	COM のオープン
c_close	COM のクローズ
c_dout	文字列の送信
c_tmdin	タイムアウト監視の受信

14.2.1 c_open

USB HID ポートをオープンします。

```
ER 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
COM8　　USB HID インタフェース

param
0 を指定してください。

busy_ch
0 を指定してください。

nonbusy_ch
0 を指定してください。

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;
```

各値には 0 を指定してください。

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;
```

各値には 0 を指定してください。

戻り値

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

14.2.2 c_close

オープン中の通信ポートをクローズします。

```
ER  c_close(
 H   com_no
);
```

パラメータ

com_no
通信ポート
COM8　　USB HID インタフェース

戻り値

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

14.2.3 c_dout

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  c_dout(
 H   com_no,
 B   *buffer,
 H   length
);
```

パラメータ

com_no
通信ポート
COM8　　USB HID 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

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

14.2.4 c_tmdin

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

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

パラメータ

com_no
通信ポート
COM8　　USB HID インタフェース

buffer
格納バッファアドレス

rcv_time
受信タイムアウト監視値 0～32767 (×7. 8ms)

戻り値

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

15.IrDA 制御

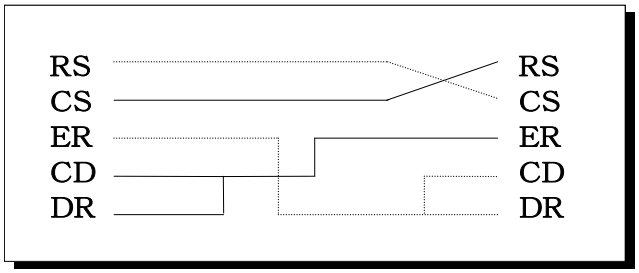
15.1 機能

15.1.1 シリアルポートエミュレーション

信号結線

IrDA ポートを使用して DT-970 同士を接続したとき、9Wire の信号結線は以下に示す通りにエミュレートします。

【信号結線】



15.1.2 ファンクションコール

⑼ **IrCOMM オープン**

IrCOMM（IrDA ポート）をオープンします。
IrDA 部の変数初期化、赤外線デバイス電源 ON、通信用デバイスおよびリソースのロックを行います。
このあと他局とコネクト行いオープンとなります。
DT-970 の IrDA 部のシリアルポートエミュレーションでは局指定を行う必要があります。
1 次局はデータリンクを指示する役割を持ちます。
1 次局は回線上（空間）に接続可能な 2 次局があるときデータリンクの指示を行い、データリンクを確立します。
1 次局と 2 次局のデータリンク確立が行われデータ通信が行える状態（オープン状態）をコネクトと言います。
回線上（空間）に接続可能な 2 次局が存在しないときコネクト待ちとなり、接続可能な 2 次局が見つからなければコネクト待ち時間指定によりタイムアウトして終了します。
2 次局は 1 次局からのデータリンクの指示を受けてデータリンクを確立します。
1 次局からのデータリンクの指示がなく、コネクト待ち時間を経過するとタイムアウトして終了します。
コネクト待ちのとき LB エラー、ブレイクイベントのチェックを行ないます。
待ち時間は秒単位に指定するかまたは、コネクトを行うまで待つかを指定します。
尚、本関数が異常終了した場合、IrCOMM はクローズ状態となります。

⑵ IrCOMM クローズ

IrCOMM（IrDA ポート）をクローズします。
相手局とコネクトを切断するための手続き（通信）を行います。
この処理中に異常が発生することで異常終了となることがありますが、正常にコネクト切断できた場合と同様に赤外線デバイス電源 OFF、通信用デバイスおよびリソースのリリースを行いクローズ状態となります。
尚、相手局からのコネクト切断を正常に受理した状態で本機能が使用された場合は正常終了となります。

(3) データ読み込み

受信データの読み込みを行います。
ユーザ定義のエリアに受信バッファデータの読み込みを行い、読み込んだバイトサイズを返します。
受信バッファデータが無くなるか、ユーザ定義のバッファサイズがフルになるまで読み込みが可能です。
受信バッファが空でもデータ待ち時間が指定されている場合はデータ待ちとなります。
このとき LB エラー、タイムアウト、ブレイクイベントのチェックおよび、パリティ、オーバーラン、フレーミングエラーのチェックを行い、エラー時は直ちに異常終了となります。
データ待ちからは、受信バッファから 1 バイト以上のデータの読み込みが行え、かつ受信バッファに受信データが無くなればユーザ定義のバッファサイズに満たない場合でも終了となります。
また、受信データがある場合でも読み込み後に LB エラー、ブレイクイベントのチェックおよび、パリティ、オーバーラン、フレーミングエラーのチェックを行いエラー時は直ちに異常終了となります。
このため受信データの読み込みが正常に行われていても異常終了となる場合があります。
データ待ち時間の指定は、通信状態設定関数(Ir_State_Set)で行うことができます。
尚、本関数では受信バッファに受信データが存在するとき、コネクト切断によるエラーは、受信バッファのデータが無くなるまで通知しません。
この場合、受信バッファのすべてのデータ読み込みが終了したとき、に異常終了となりますが、受信データはユーザ定義のエリアへ格納されています。
また、相手局からのコネクト切断を待つときは本機能を使用することで可能です。
ユーザアプリケーションが従局的な役割であるときは本機能で主局側からのコネクト切断を待ち、必要に応じて(受信待ちタイムアウトになった場合等)IrCOMM クローズを行うようにして下さい。

(4) データ書き込み

送信データの書き込みを行います。
ユーザ定義のエリアから送信バッファに送信データの書き込みを行い、書き込んだバイトサイズを返します。
送信バッファに送信データの書き込みが行えなくなるか(バッファビジー)、ユーザー定義のバイトサイズまで書き込みを行います。
データ待ち時間が指定されていれば送信バッファに書き込みが行えないときデータ待ちとなり、一度データ待ちとなるとすべてのデータの書き込みが終了するまでの間をデータ待ち時間としてタイマによる監視を行います。
データ待ちの間およびデータ書き込み後に LB エラー、ブレイクイベント、タイムアウトのチェックを行い、エラー時は直ちに異常終了となります。
このため送信データの書き込みが正常に行われていても異常終了となる場合があります。
データ待ち時間の指定は、通信状態設定関数(Ir_State_Set)で行うことができます。

(5) 送信データ数問合せ

送信バッファに残っている未送出のデータ数を問合せます。結果をバイトサイズで返します。
IrDA プロトコル上では送信バッファに書き込まれたデータが送出されるまで、ある程度の時間が掛かります。
本機能でデータが送出されたかを調べることができます。

(6) 受信データ数問合せ

受信バッファより読み込み可能なデータ数を問合せます。結果をバイトサイズで返します。

(7) ER ON
ER 信号を ON にします。IrDA による信号線のエミュレートとなります。
IrDA プロトコル規定の ER 信号 ON を指示するデータフレームを相手局に送信します。
このため、"データ書き込み"機能と同様に送信バッファへの書き込みを行います。
データ待ちの間およびデータ書き込み後に LB エラー、ブレイクイベント、タイムアウトのチェックを行い、エラー時は直ちに異常終了となります。
データ待ち時間の指定は、通信状態設定関数(Ir_State_Set)で行うことができます。

(8) ER OFF
ER 信号を OFF にします。IrDA による信号線のエミュレートとなります。
IrDA プロトコル規定の ER 信号 OFF を指示するデータフレームを相手局に送信します。
このため、"データ書き込み"機能と同様に送信バッファへの書き込みを行います。
データ待ちの間およびデータ書き込み後に LB エラー、ブレイクイベント、タイムアウトのチェックを行い、エラー時は直ちに異常終了となります。
データ待ち時間の指定は、通信状態設定関数(Ir_State_Set)で行うことができます。

(9) RS ON
RS 信号を ON にします。IrDA による信号線のエミュレートとなります。
IrDA プロトコル規定の RS 信号 ON を指示するデータフレームを相手局に送信します。
このため、"データ書き込み"機能と同様に送信バッファへの書き込みを行います。
データ待ちの間およびデータ書き込み後に LB エラー、ブレイクイベント、タイムアウトのチェックを行い、エラー時は直ちに異常終了となります。
データ待ち時間の指定は、通信状態設定関数(Ir_State_Set)で行うことができます。

(10) RS OFF
RS 信号を OFF にします。IrDA による信号線のエミュレートとなります。
IrDA プロトコル規定の RS 信号 OFF を指示するデータフレームを相手局に送信します。
このため、"データ書き込み"機能と同様に送信バッファへの書き込みを行います。
データ待ちの間およびデータ書き込み後に LB エラー、ブレイクイベント、タイムアウトのチェックを行い、エラー時は直ちに異常終了となります。
データ待ち時間の指定は、通信状態設定関数(Ir_State_Set)で行うことができます。

(11) BREAK ON
ブレイク信号を送出します。IrDA による信号のエミュレートとなります。
IrDA プロトコル規定のブレイク信号送出を指示するデータフレームを相手局に送信します。
このため、"データ書き込み"機能と同様に送信バッファへの書き込みを行います。
データ待ちの間およびデータ書き込み後に LB エラー、ブレイクイベント、タイムアウトのチェックを行い、エラー時は直ちに異常終了となります。
データ待ち時間の指定は、通信状態設定関数(Ir_State_Set)で行うことができます。

(12) BREAK OFF
ブレイク信号の送出を停止します。IrDA による信号のエミュレートとなります。
IrDA プロトコル規定のブレイク信号停止を指示するデータフレームを相手局に送信します。
このため、"データ書き込み"機能と同様に送信バッファへの書き込みを行います。
データ待ちの間およびデータ書き込み後に LB エラー、ブレイクイベント、タイムアウトのチェックを行い、エラー時は直ちに異常終了となります。
データ待ち時間の指定は、通信状態設定関数(Ir_State_Set)で行うことができます。

(13) CD 検査
CD 信号の ON/OFF 状態をチェックし、通知します。信号の ON または OFF の指定を行います。
信号待ち時間が指定されているとき指定した信号状態でなければ信号待ちとなります。
信号待ちとなったとき LB エラー、タイムアウト、ブレイクイベントのチェックを行い、エラー時は直ちに異常終了となります。
データ待ち時間の指定は、通信状態設定関数(Ir_State_Set)で行うことができます。

(14) DR 検査
DR 信号の ON/OFF 状態をチェックし、通知します。
信号待ち時間が指定されているとき信号の ON 待ちとなります。
信号待ちとなったのとき、LB エラー、タイムアウト、ブレイクイベントのチェックを行い、エラー時は直ちに異常終了となります。
信号待ち時間の指定は、通信状態設定関数(Ir_State_Set)で行うことができます。

(15) CS 検査
CS 信号の ON/OFF 状態をチェックし、通知します。
信号待ち時間が指定されているとき信号の ON 待ちとなります。
信号待ち時、LB エラー、タイムアウト、ブレイクイベントのチェックを行い、エラー時は直ちに異常終了となります。
信号待ち時間の指定は、通信状態設定関数(Ir_State_Set)で行うことができます。

(16) CI 検査
CI 信号の ON/OFF 状態をチェックし、通知します。

(17) BREAK 検査
ブレイク信号の受信状態をチェックし、通知します。このとき受信状態をクリアします。
尚、ブレイク受信となった場合、フレーミングエラーは発生しません。

(18) エラー値取得
エラー値を取得します。
各関数の異常終了の詳細となるエラー値を返します。このときエラー値をクリアします。

(19) 通信状態設定
IrDA 部の通信状態を設定します。本機能は IrCOMM オープンに先立って行う必要があります。
局は自局が 1 次局か 2 次局であるかを指定します。

- 1 次局
  データリンクを 2 次局に指示します。自局が 1 次局であるときコネクトする相手局は 2 次局となります。

- 2 次局
  1 次局からデータリンクの指示を受けます。2 次局は 1 次局からのデータリンクの指示を受けることでコネクトします。自局が 2 次局であるときコネクトする相手局は 1 次局となります。
  Wire はエミュレートする結線タイプを指定します。結線タイプを Wire と呼び、Wire の指定により機能が異なります。
  指定する各 Wire の機能は次の通りです。

- 3Wire—raw
  実データの送受信のみ行えます。信号線制御、通信エラーの通知（POF エラー）などの機能は持ちません。

- 3Wire
  実データの送受信の他に RS232C の設定、通信エラーの通知（POF エラー）、ブレイク信号等の機能を持ちます。

- 9Wire
  3Wire と信号線制御の機能を持ちます。この Wire が指定されているとき各関数の信号線チェックが有効となります。

- LPT
  3Wire—raw と同等ですが IrLPT クラス名を持つプリンタにコネクトする場合はこの指定を行って下さい。
  また 1 次局に指定する必要があります。
  データ待ち時間は秒単位、待ちなし、無限待ちを指定することができます。
  データ待ちには次の状態があり、各関数でのデータ待ちを行います。

- 送信バッファにデータの書き込みが行えないとき
- 受信バッファに読み込み可能なデータが無いとき
  DR/CS/CD 待ち時間は秒単位、待ちなし、無限待ちを指定することができます。
  信号待ちには次の状態があり、各関数での信号待ちを行います。Wire が 9Wire に指定されているとき有効となります。

- DR 信号が OFF のとき
- CS 信号が OFF のとき
- CD 信号が ON のとき
- CD 信号が OFF のとき
  RS232C の通信設定を行うことができます。Wire が 3Wire または 9Wire に指定されているとき有効となります。

- 通信速度
- データ長
- ストップビット
- パリティビット

（20）自局能力設定

自局能力を設定します。本機能は IrCOMM オープンに先立って使用する必要があります。
設定値は IrDA 規格書に記されている折衝フィールドパラメータです。
パラメータは以下に示す通りです。

- ボーレート
- 最大ターンアラウンドタイム
- フレームデータサイズ
- ウインドウサイズ
- BOF 数
- 最小ターンアラウンドタイム
- リンク開放時間

(21) IrCOMM 強制終了

IrCOMM を強制終了します。
IrCOMM をオープン状態から初期状態（クローズ状態）に設定します。
基本適には IrCOMM クローズと同じ機能をもちますが、通信状態に関係なく直ちに赤外線デバイス電源 OFF、赤外線通信用リソースのリリースを行います。

15.1.3 ファンクションの優先順位

各ファンクションには優先順位があります。優先順位は高い順にプライオリティ 5～1 となっています。
基本的にレベルの高い順に使用します。以下に示す一覧を参考にして下さい。

プライオリティ	ファンクションコール名	備考
5	Ir_State_set, Ir_SetPortConfig	(9) プライオリティ 4 次のファンクションより先に使用して下さい。 (9) リセット直後はデフォルト値となります。 (9) クローズ状態である必要があります。 ・設定値はリセットされるまで有効です。
4		・予約とします。
3	Ir_Err_Get	・プライオリティ 2 次のファンクションと同じ優先順位で使用できます。
2	Ir_Open	・クローズ状態である必要があります。
1	Ir_Close, Ir_Read, Ir_Write, Ir_QueryTx, Ir_QuertRx, Ir_EROn, Ir_EROff, Ir_RSOn, Ir_Rsoff, Ir_BreakOn, Ir_BreakOff, Ir_CheckCD, Ir_CheckDR, Ir_CheckCS, Ir_CheckCI, Ir_CheckBreak, Ir_Init	・オープン状態である必要があります。

15.2 関数リファレンス

ファンクション詳細を次ページより示します。

関数	機能概要
Ir_Open	IrCOMM のオープン
Ir_Close	IrCOMM のクローズ
Ir_Read	データの読込
Ir_Write	データの書込
Ir_QueryTx	送信データ数の問合せ
Ir_QueryRx	受信データ数の問合せ
Ir_EROn	ER 信号の ON
Ir_EROff	ER 信号の OFF
Ir_RSOn	RS 信号の ON
Ir_RSOff	RS 信号の OFF
Ir_BreakOn	ブレーク送信の ON
Ir_BreakOff	ブレーク送信の OFF
Ir_CheckCD	CD の検査
Ir_CheckDR	DR の検査
Ir_CheckCS	CS の検査
Ir_CheckCI	CI の検査
Ir_CheckBreak	BREAK の検査
Ir_Err_Get	エラー値の取得
Ir_State_Set	通信状態の設定
Ir_SetPortConfig	自局能力の設定
Ir_Init	IrCOMM の強制終了
Ir_SetWinMode	Ir モード切り替えの設定

15.2.1 Ir_Open

IrCOMM(赤外線ポート)をオープンします。

- IrDA 部の初期化
- 通信用のデバイスおよびリソースのロック
- 赤外線デバイス電源 ON
- ブレイクイベントのチェック
- LB チェック
- 相手局とのコネクト

```
H Ir_Open(
   H   sec
);
```

パラメータ

sec

コネクト最大待ち時間

1～3600	:待ち時間(秒)
FOREVER	:正常または異常終了するまでコネクト待ちします

戻り値

E_IROK	:正常終了
E_IRNG	:異常終了

15.2.2 Ir_Close

IrCOMM(赤外線ポート)をクローズします。

- 通信用のデバイスおよびリソースのリリース
- 赤外線デバイス電源 OFF
- ブレイクイベントのチェック
- LB チェック
- コネクト切断

```
H Ir_Close();
```

パラメータ

なし

戻り値

E_IROK	:正常終了
E_IRNG	:異常終了

15.2.3 Ir_Read

受信データの読み込みを行います。

- データ受信待ち(受信バッファにデータが無いとき)
- 受信データの読み込み(受信バッファからの読み込み)
- LB チェック
- ブレイクイベントのチェック
- 読み込みデータ数の通知

```
H Ir_Read(
   B      *buff,
   UH     ReadSize,
   UH     *GetSize
);
```

パラメータ

buff
受信データを格納するバッファのポインタ

ReadSize
受信データを格納するバッファのサイズ(バイト数)

GetSize
読み込みデータ数(受信バッファから読み込みできたバイト数)

戻り値

E_IROK	:正常終了
E_IRNG	:異常終了(状態によってはデータの読み込みが行われています)

15.2.4 Ir_Write

送信データの書き込みを行います。

- LB チェック
- データ書き込み待ち(送信バッファへの書き込みが終了するまで)
- 送信データの書き込み(送信バッファへの書き込み)
- ブレイクイベントのチェック
- 書き込みデータ数の通知

```
H Ir_Write(
   B      *buff,
   UH     WriteSize,
   UH     *PutSize
);
```

パラメータ

buff
送信データを格納するバッファのポインタ

WriteSize
送信データ数(送信バッファに書き込むバイト数)

PutSize
書き込みデータ数(送信バッファに書き込みできたバイト数)

戻り値

E_IROK	:正常終了
E_IRNG	:異常終了(状態によってはデータの書き込みが行われています)

15.2.5 Ir_QueryTx

送信バッファに残っている未送出データ数を問合せます。
- 送信バッファ内の未送出のデータ数の通知
- LB チェック
- ブレイクイベントのチェック

```
H Ir_QueryTx(
   H   *SndDataSize
);
```

パラメータ

SndDataSize
未送出データ数(バイト)

戻り値

E_IROK　　:正常終了
E_IRNG　　:異常終了

15.2.6 Ir_QueryRx

受信バッファより読み込み可能なデータ数を問合せます。
- 受信バッファ内の読み込み可能なデータ数の通知
- LB チェック
- ブレイクイベントのチェック

```
H Ir_QueryRx(
   H      *RcvDataSize
)
```

パラメータ

RcvDataSize
読み込み可能なデータ数(バイト)

戻り値

E_IROK　　:正常終了
E_IRNG　　:異常終了

15.2.7 Ir_EROn

ER 信号を ON にします。IrDA プロトコルによる信号線のエミュレートになります。

- LB チェック
- ブレイクイベントのチェック
- 信号制御データの作成
- 送信データの書き込み(制御データ)
- データ書き込み待ち(送信バッファへの書き込みが終了するまで)

```
H Ir_EROn();
```

パラメータ

なし

戻り値

E_IROK	:正常終了
E_IRNG	:異常終了

解説

信号線をエミュレートするため、Ir_State_Set 関数で、9-wire に設定しておく必要があります。

15.2.8 Ir_EROff

ER 信号を OFF にします。IrDA プロトコルによる信号線のエミュレートになります。

- LB チェック
- ブレイクイベントのチェック
- 信号制御データの作成
- 送信データの書き込み(制御データ)
- データ書き込み待ち(送信バッファへの書き込みが終了するまで)

```
H Ir_EROff();
```

パラメータ

なし

戻り値

E_IROK	:正常終了
E_IRNG	:異常終了

解説

信号線をエミュレートするため、Ir_State_Set 関数で、9-wire に設定しておく必要があります。

15.2.9 Ir_RSOn

RS 信号を ON にします。IrDA プロトコルによる信号線のエミュレートになります。

- LB チェック
- ブレイクイベントのチェック
- 信号制御データの作成
- 送信データの書き込み(制御データ)
- データ書き込み待ち(送信バッファへの書き込みが終了するまで)

```
H Ir_RSOn();
```

パラメータ

なし

戻り値

E_IROK	:正常終了
E_IRNG	:異常終了

解説

信号線をエミュレートするため、Ir_State_Set 関数で、9-wire に設定しておく必要があります。

15.2.10 Ir_RSOff

RS 信号を OFF にします。IrDA プロトコルによる信号線のエミュレートになります。

- LB チェック
- ブレイクイベントのチェック
- 信号制御データの作成
- 送信データの書き込み(制御データ)
- データ書き込み待ち(送信バッファへの書き込みが終了するまで)

```
H Ir_RSOff();
```

パラメータ

なし

戻り値

E_IROK	:正常終了
E_IRNG	:異常終了

解説

信号線をエミュレートするため、Ir_State_Set 関数で、9-wire に設定しておく必要があります。

15.2.11 Ir_BreakOn

ブレーク信号を送出します。IrDA プロトコルによる信号のエミュレートになります。

- LB チェック
- ブレイクイベントのチェック
- 信号制御データの作成
- 送信データの書き込み(制御データ)
- データ書き込み待ち(送信バッファへの書き込みが終了するまで)

```
H Ir_BreakOn();
```

パラメータ

なし

戻り値

E_IROK	:正常終了
E_IRNG	:異常終了

解説

信号線をエミュレートするため、Ir_State_Set 関数で、9-wire に設定しておく必要があります。

15.2.12 Ir_BreakOff

ブレーク信号の送出を停止します。IrDA プロトコルによる信号線のエミュレートになります。

- LB チェック
- ブレイクイベントのチェック
- 信号制御データの作成
- 送信データの書き込み(制御データ)
- データ書き込み待ち(送信バッファへの書き込みが終了するまで)

```
H Ir_BreakOff();
```

パラメータ

なし

戻り値

E_IROK	:正常終了
E_IRNG	:異常終了

解説

信号線をエミュレートするため、Ir_State_Set 関数で、9-wire に設定しておく必要があります。

15.2.13 Ir_CheckCD

CD 信号の ON/OFF 状態をチェックします。また信号待ち時間の指定があるとき信号が ON または OFF になるのを待ちます。
ON/OFF 待ちをパラメータで指定することができます。

- 信号 ON または OFF 待ち
- LB チェック
- ブレイクイベントのチェック
- 信号状態の通知

```
H Ir_CheckCD(
   H   line
);
```

パラメータ

line
信号 ON/OFF 待ち指定
ON_WAIT　　:信号が ON になるまで待つ
OFF_WAIT　　:信号が OFF になるまで待つ

戻り値

E_IRLINE_ON　　:信号 ON
E_IRLINE_OFF　　:信号 OFF
E_IRNG　　:異常終了

解説

信号線をエミュレートするため、Ir_State_Set 関数で、9-wire に設定しておく必要があります。

15.2.14 Ir_CheckDR

DR 信号の ON/OFF 状態をチェックします。また信号待ち時間の指定があるとき信号が ON になるのを待ちます。

- 信号 ON 待ち
- LB チェック
- ブレイクイベントのチェック
- 信号状態の通知

```
H Ir_CheckDR();
```

パラメータ

なし

戻り値

E_IRLINE_ON	:信号 ON
E_IRLINE_OFF	:信号 OFF
E_IRNG	:異常終了

解説

信号線をエミュレートするため、Ir_State_Set 関数で、9-wire に設定しておく必要があります。

15.2.15 Ir_CheckCS

CS 信号の ON/OFF 状態をチェックします。また信号待ち時間の指定があるとき信号が ON になるのを待ちます。

- 信号 ON 待ち
- LB チェック
- ブレイクイベントのチェック
- 信号状態の通知

```
H Ir_CheckCS();
```

パラメータ

なし

戻り値

E_IRLINE_ON	:信号 ON
E_IRLINE_OFF	:信号 OFF
E_IRNG	:異常終了

解説

信号線をエミュレートするため、Ir_State_Set 関数で、9-wire に設定しておく必要があります。

15.2.16 Ir_CheckCI

CI 信号の ON/OFF 状態をチェックします。

- 信号状態の通知
- LB チェック
- ブレイクイベントのチェック

```
H Ir_CheckCI();
```

パラメータ

なし

戻り値

E_IRLINE_ON	:信号 ON
E_IRLINE_OFF	:信号 OFF
E_IRNG	:異常終了

解説

信号線をエミュレートするため、Ir_State_Set 関数で、9-wire に設定しておく必要があります。
エラー詳細は、エラー値取得関数(Ir_Err_Get) にて取得して下さい

15.2.17 Ir_CheckBreak

BREAK 信号の受信をチェックします。

- 信号受信状態の通知
- LB チェック
- ブレイクイベントのチェック

```
H Ir_CheckBreak();
```

パラメータ

なし

戻り値

E_IRBRK_ON	:ブレイク信号検出
E_IRBRK_OFF	:ブレイク信号未検出
E_IRNG	:異常終了

15.2.18 Ir_Err_Get

エラー値を取得します。また取得後にエラー値をクリアします。

- エラー値のクリア
- エラー値の通知

```
UW Ir_Err_Get();
```

パラメータ

なし

戻り値

詳細は、解説を参照して下さい

解説

エラー発生要因
次のフォーマットでエラー値について示します。

エラー値	エラーコード名称
詳細 エラーの詳細	

関数名	IrCOMM 状態	主なエラー対処方法
エラーの 発生する関数名	関数異常終了時の IrCOMM オープン状態	IrDA 部の上位が行う発生したエラーに対しての事後処理

エラー値	IRERR_NORESOURCE
詳細 IrDA 部内の資源不足により LASP(コネクトに必要な内部情報)が確保できないと発生します。 IRERR_DISCONNECT エラーの要因として一緒に通知します。 通常このエラーが発生することはありえないのでダンプ等を行い原因の調査をする必要があります。	

関数名	IrCOMM 状態	主なエラー対処方法
Ir_Open	オープンは行わずクローズ状態	・ダンプ等を行い、原因調査をする必要があります。

エラー値	IRERR_NODEVICE
詳細 回線上(空間)にコネクト可能なデバイスがないとき発生します。 IRERR_DISCONNECT エラーの要因として一緒に通知します。 Ir_Open 関数でのコネクト待ちタイムアウトの要因でもあります。	

関数名	IrCOMM 状態	主なエラー対処方法
Ir_Open	オープンは行わずクローズ状態	(9) クレードルの装着を所定の位置に正しく固定して再実行して下さい。 ・デバイス同士を 20cm 以内に接近させて再実行して下さい。

エラー値	IRERR_NOLSAP

詳細

回線上（空間）にコネクト可能なアプリケーションが無いときやIrDAプロトコルの実装が異なる（相手局にIrCOMM層がない）ときに発生します。
IRERR_DISCONNECTエラーの要因として一緒に通知します。
Ir_Open関数でのコネクト待ちタイムアウトの要因でもあります。

関数名	IrCOMM状態	主なエラー対処方法
Ir_Open	オープンは行わずクローズ状態	・相手局のプロトコル実装を確認して再実行して下さい。

エラー値	IRERR_LOCK

詳細

通信用のデバイスおよびリソースが既にロックされているときに発生します。
DT-970では通信関数とシステム資源を共有していますので先に起動された方に資源を利用する。
権利があります。このエラーが発生したときは資源が開放されるまで待たなければなりません。
また、OBRとの排他制御を行っていますのでOBR起動中にも当該エラーとなります。

関数名	IrCOMM状態	主なエラー対処方法
Ir_Open	オープンは行わずクローズ状態	(9) デバイス、リソースが開放されてから再実行して下さい ・既にIrCOMM(赤外線ポート)がオープンしています。クローズしてから再実行して下さい

エラー値	IRERR_DISCONNECT

詳細

コネクト手続き中またはコネクト後に相手局からの応答が無くなったとき、相手局からコネクト。
切断されたとき、レジュームON立上げを行ったときに発生します。

関数名	IrCOMM状態	主なエラー対処方法
Ir_Open	オープンは行わずクローズ状態	・通信環境を確認して再実行して下さい。
Ir_Close Ir_Read Ir_Write Ir_EROn Ir_EROff Ir_RSOn Ir_RSOff Ir_BreakOn Ir_BreakOff Ir_QueryTx Ir_QueryRx Ir_CheckCD Ir_CheckDR Ir_CheckCS Ir_CheckCI Ir_CheckBreak Ir_Init	クローズ状態となります。	(9) 相手局と通信不可能な環境にあるのでその原因を取り除いてIrCOMM(赤外線ポート)のオープンを行って下さい。 相手局から一定時間応答がない（回線が外れている）場合が考えられます。

エラー値	IRERR_PARAMETER

詳細

関数のパラメータの入力値に誤りがあるとき発生します。

関数名	IrCOMM 状態	主なエラー対処方法
Ir_Open	カレントの状態に変化はありません。	・入力パラメータを確認して下さい。
Ir_CheckCD		
Ir_State_set		
Ir_SetPortConfig		

エラー値	IRERR_BREAK_EVNT

詳細

キー関数の機能を用いて中断キーのイベント登録が行なわれ、当該キー押下によりブレイクイベントの検出が IrDA 部で行われたときに発生します。

関数名	IrCOMM 状態	主なエラー対処方法
Ir_Open	オープンは行わずクローズ状態	・再実行する場合は Ir_Open から行って下さい。
Ir_Close	クローズ状態となります。	
Ir_Read	オープン状態から変更はありません。	・Ir_Close を行って終了して下さい。
Ir_Write		
Ir_EROn		
Ir_EROff		
Ir_RSOn		
Ir_RSOff		
Ir_BreakOn		
Ir_BreakOff		
Ir_CheckCD		
Ir_CheckDR		
Ir_CheckCS		
Ir_QueryTx		
Ir_QueryRx		
Ir_CheckCI		
Ir_CheckBreak		

エラー値	IRERR_LB0

詳細

LB0 イベントの通知が有効に設定されており、LB0 エラー（主電池なし、電池蓋開き）になったときに発生します。

このエラーはレジューム ON 立上げ時に通知します。レジューム ON 立上げで IrCOMM（赤外線ポート）はクローズ状態となりますので IRERR_DISCONNECT と一緒に通知されます。

関数名	IrCOMM 状態	主なエラー対処方法
Ir_Open	オープンは行わずクローズ状態	(9) 再実行するときは Ir_Open を行って下さい。 ・イベントのクリアを行って下さい。
Ir_Close	クローズ状態となります。	
Ir_Read		
Ir_Write		
Ir_EROn		

Ir_EROff		
Ir_RSOn		
Ir_RSOff		
Ir_BreakOn		
Ir_BreakOff		
Ir_CheckCD		
Ir_CheckDR		
Ir_CheckCS		
Ir_QueryTx		
Ir_QueryRx		
Ir_CheckCI		
Ir_CheckBreak		
Ir_Init		

エラー値	IRERR_LB1	
詳細 LB1 イベントの通知が有効に設定されており、LB1 エラー（主電池電圧低下）になったときに発生します。		
関数名	IrCOMM 状態	主なエラー対処方法
Ir_Open	オープンは行わずクローズ状態	(9) 電池交換を行って下さい。 (9) イベントのクリアを行って下さい。 ・再実行するときは Ir_Open を行って下さい。
Ir_Close	クローズ状態となります。	
Ir_Init		
Ir_Read	オープン状態から変更はありません。	(9) Ir_Close を行って終了して下さい。 (9) 電池交換後に Ir_Open を行って下さい。 ・イベントのクリアを行って下さい。
Ir_Write		
Ir_EROn		
Ir_EROff		
Ir_RSOn		
Ir_RSOff		
Ir_BreakOn		
Ir_BreakOff		
Ir_CheckCD		
Ir_CheckDR		
Ir_CheckCS		
Ir_QueryTx		
Ir_QueryRx		
Ir_CheckCI		
Ir_CheckBreak		

エラー値	IRERR_LB2	
詳細 LB2 イベントの通知が有効に設定されており、LB2 エラー（副電池電圧なし）になったときに発生します。		
関数名	IrCOMM 状態	主なエラー対処方法
Ir_Open	オープンは行わずクローズ状態	(9) 電池交換後を行って下さい。 (9) イベントのクリアを行って下さい。 (9) 再実行するときは Ir_Open を行って下さい。
Ir_Close	クローズ状態となります。	

Ir_Init		
Ir_Read	オープン状態から変更はありません。	(9) Ir_Close を行って終了して下さい。 (9) 電池交換後に Ir_Open を行って下さい。 (9) イベントのクリアを行って下さい。
Ir_Write		
Ir_EROn		
Ir_EROff		
Ir_RSOn		
Ir_RSOff		
Ir_BreakOn		
Ir_BreakOff		
Ir_CheckCD		
Ir_CheckDR		
Ir_CheckCS		
Ir_QueryTx		
Ir_QueryRx		
Ir_CheckCI		
Ir_CheckBreak		

エラー値	IRERR_LB4

詳細

LB4 イベントの通知が有効に設定されており、LB4 エラー（APO 発生）になったときに発生します。
通知が有効に設定されているときは電源 OFF しませんので、アプリケーションが責任を持つ必要があります。

関数名	IrCOMM 状態	主なエラー対処方法
Ir_Open	オープンは行わずクローズ状態	(9) イベントのクリアを行って下さい。 ・電源 OFF して下さい。
Ir_Close	クローズ状態となります。	
Ir_Init		
Ir_Read	オープン状態から変化はありません。	(9) Ir_Close を行って終了して下さい。 (9) イベントのクリアを行って下さい。 ・電源 OFF して下さい。
Ir_Write		
Ir_EROn		
Ir_EROff		
Ir_RSOn		
Ir_RSOff		
Ir_BreakOn		
Ir_BreakOff		
Ir_CheckCD		
Ir_CheckDR		
Ir_CheckCS		
Ir_QueryTx		
Ir_QueryRx		
Ir_CheckCI		
Ir_CheckBreak		

エラー値	IRERR_LB5

詳細

LB5 イベントの通知が有効に設定されており、LB5 エラー（OFF キー押下による電源 OFF）になったときに発生します。

通知が有効に設定されているときは電源 OFF しませんので、アプリケーションが責任を持つ必要があります。

関数名	IrCOMM 状態	主なエラー対処方法
Ir_Open	オープンは行わずクローズ状態	(9) イベントのクリアを行って下さい。 ・電源 OFF して下さい。
Ir_Close	クローズ状態となります。	
Ir_Init		
Ir_Read	オープン状態から変化はありません。	(9) Ir_Close を行って終了して下さい。 (9) イベントのクリアを行って下さい。 ・電源 OFF して下さい。
Ir_Write		
Ir_EROn		
Ir_EROff		
Ir_RSOn		
Ir_RSOff		
Ir_BreakOn		
Ir_BreakOff		
Ir_CheckCD		
Ir_CheckDR		
Ir_CheckCS		
Ir_QueryTx		
Ir_QueryRx		
Ir_CheckCI		
Ir_CheckBreak		

エラー値	IRERR_NOTOPEN

詳細

IrCOMM（赤外線ポート）がオープンされていないときに発生します。

関数名	IrCOMM 状態	主なエラー対処方法
Ir_Close	クローズ状態から変化はありません。	(9) IrCOMM が既にクローズ状態になっています。 Ir_Open を行ってから実行して下さい。
Ir_Read		
Ir_Write		
Ir_QueryTX		
Ir_QueryRx		
Ir_EROn		
Ir_EROff		
Ir_RSOn		
Ir_RSOff		
Ir_BreakOn		
Ir_BreakOff		
Ir_CheckCD		
Ir_CheckDR		
Ir_CheckCS		
Ir_CheckCI		

Ir_CheckBreak		
Ir_Init		

エラー値	IRERR_TIMEOUT	
詳細 Ir_Open 関数で指定したコネクト待ち時間を経過した場合と、Ir_State_Set 関数で指定したデータ待ち時間を経過すると発生します。		
関数名	IrCOMM 状態	主なエラー対処方法
Ir_Open	クローズ状態です	・通信環境を確認して再実行して下さい。
Ir_Read Ir_Write Ir_RSOn Ir_RSOff Ir_EROn Ir_EROff Ir_BreakOn Ir_BreakOff	オープン状態に変化はありません。	・任意の処理を行って下さい。再実行してもかまいません。

エラー値	IRERR_PARITY	
詳細 Ir_State_Set 関数で次の指定のとき RS232C 上（クレードルと PC 間など）でパリティエラーとなったときに発生します。 （9） 3Wire または 9Wire ・パリティあり		
関数名	IrCOMM 状態	主なエラー対処方法
Ir_Read	オープン状態に変化はありません。	・任意の処理を行って下さい。

エラー値	IRERR_OVERRUN	
詳細 Ir_State_Set 関数で次の指定のとき RS232C 上で（クレードルと PC 間など）でオーバーランエラーとなったときに発生します。 ・3Wire または 9Wire		
関数名	IrCOMM 状態	主なエラー対処方法
Ir_Read	オープン状態に変化はありません。	・任意の処理を行って下さい。

エラー値	IRERR_FRAMING	
詳細 Ir_State_Set 関数で次の指定のとき RS232C 上で（クレードルと PC 間など）でフレーミングエラーとなったときに発生します。 ・3Wire または 9Wire		
関数名	IrCOMM 状態	主なエラー対処方法
Ir_Read	オープン状態に変化はありません。	・任意の処理を行って下さい。

エラー値	IRERR_WIRE_TYPE

詳細

Ir_State_Set 関数で 3Wire あるいは 9Wire が指定されている場合しか使用できない関数を使用したとき発生します。

関数名	IrCOMM 状態	主なエラー対処方法
Ir_EROn	オープン状態に変化はありません。	・9Wire 以外が指定されています。用途に合った指定を行って下さい。
Ir_EROff		
Ir_RSOn		
Ir_RSOff		
Ir_BreakOn		
Ir_BreakOff		
Ir_CheckCD		
Ir_CheckDR		
Ir_CheckCS		
Ir_CheckCI		
Ir_CheckBreak	オープン状態に変化はありません。	(9) 3Wire または 9Wire 以外に指定されています。用途に合った指定を行って下さい。
Ir_BreakOn		
Ir_BreakOff		

エラー値	IRERR_CD_TIMEOUT

詳細

Ir_State_Set 関数で指定した信号待ち時間を経過すると発生します。
指定時間内に CD 信号が ON または OFF に変化しませんでした。

関数名	IrCOMM 状態	主なエラー対処方法
Ir_CheckCD	オープン状態に変化はありません。	任意の処理を行って下さい。再実行してもかまいません。

エラー値	IRERR_DR_TIMEOUT

詳細

Ir_State_Set 関数で指定した信号待ち時間を経過すると発生します。
指定時間内に DR 信号が OFF から ON に変化しませんでした。

関数名	IrCOMM 状態	主なエラー対処方法
Ir_CheckDR	オープン状態に変化はありません。	任意の処理を行って下さい。再実行してもかまいません。

エラー値	IRERR_CS_TIMEOUT

詳細

Ir_State_Set 関数で指定した信号待ち時間を経過すると発生します。
指定時間内に CS 信号が OFF から ON に変化しませんでした。

関数名	IrCOMM 状態	主なエラー対処方法
Ir_CheckCS	オープン状態に変化はありません。	任意の処理を行って下さい。再実行してもかまいません。

15.2.19 Ir_State_Set

IrDA 部の通信状態を設定します。

- Wire の指定
- データ読み込み／書き込み(データ待ち)時間の指定
- DR/CS/CD 信号待ち時間の設定(9Wire 指定時のみ有効)
- RS232C の通信仕様の指定(3/9Wire 指定時のみ有効)
- 局の指定

```
H Ir_State_Set(
   struct State_DCB   *state
);
```

パラメータ

state

下記の構造体に値を指定します。

```
struct State_DCB {
   H  station;             : 局
   H  Wire;                : Wire
   H  DataWaitTime;        : データ待ち時間
   H  LineWaitTime;        : DR/CS/CD 信号待ち時間
   H  baudRate;            : RS232C の通信速度
   H  DataLen;             : RS232C のデータ長
   H  StopBit;             : RS232C のストップビット
   H  ParityBit;           : RS232C のパリティビット
};
```

戻り値

E_IROK	:正常終了
E_IRNG	:異常終了

解説

パラメータについての詳しくは次ページを参照して下さい。
オープンに先立って使用して下さい。

参照

State_DCB 構造体

通信状態設定の DCB

項目	定数	詳細	デフォルト
局	PRIMARY	自局を 1 次局に設定	
	SECONDARY	自局を 2 次局に設定	○
Wire	WIRE3RAW	3-wire raw に設定	○
	WIRE3	3-wire に設定	
	WIRE9	9-wire に設定	
	WIRELPT	LPT(3-wire raw)に設定（クラス名を LPT に設定する）	
データ待ち時間	1-600	秒単位にデータ読み込み／書き込み待ち時間を設定	○(300)
	THROUGH	データ読み込み／書き込み待ちを行わない	
	FOREVER	タイマ指定なしでデータ読み込み／書き込み待ちを行う	
DR/CS/CD 信号待ち時間	1-600	秒単位に DR/CS/CD 信号待ち時間を設定	
	THROUGH	DR/CS/CD 信号のチェックを行わない	○
	FOREVER	タイマ指定なしで DR/CS/CD 信号待ちを行う	
RS232C の通信速度	BPS_12	RS232C の通信速度を 1200bps に設定	
	BPS_24	RS232C の通信速度を 2400bps に設定	
	BPS_48	RS232C の通信速度を 4800bps に設定	
	BPS_96	RS232C の通信速度を 9600bps に設定	○
	BPS_192	RS232C の通信速度を 19200bps に設定	
	BPS_384	RS232C の通信速度を 38400bps に設定	
	BPS_576	RS232C の通信速度を 57600bps に設定	
	BPS_1152	RS232C の通信速度を 115200bps に設定	
RS232C のデータ長	LEN_7B	RS232C のデータ長を 7bit に設定	
	LEN_8B	RS232C のデータ長を 8bit に設定	○
RS232C のストップビット	STOP_1B	RS232C のストップビットを 1bit に設定	○
	STOP_2B	RS232C のストップビットを 2bit に設定	
RS232C のパリティビット	PRI_ODD	RS232C のパリティビットを奇数パリティに設定	
	PRI_EVN	RS232C のパリティビットを偶数パリティに設定	
	PRI_NON	RS232C のパリティビットをパリティなしに設定	○

15.2.20 Ir_SetPortConfig

自局能力を設定します。

- 折衝パラメータの設定

```
H Ir_SetPortConfig(
   struct SetPortConfig_DCB   *port
);
```

パラメータ

port

下記の構造体に値を指定します。

```
struct SetPortConfig_DCB {
   UB     irBaud;          : ボーレイト
   UB     MaxTurnTime;     : 最大ターンアラウンドタイム
   UB     FrameSize;       : フレームサイズ
   UB     WindowSize;      : ウインドウサイズ
   UB     BofCount;        : BOF 数
   UB     MinTurnTime;     : 最小ターンアラウンドタイム
   UB     DiscTime;        : リンク開放時間
};
```

戻り値

E_IROK	:正常終了
E_IRNG	:異常終了

解説

パラメータについて詳しくは次ページを参照して下さい。
オープンに先立って使用して下さい。

参照

SetPortConfig_DCB 構造体

自局能力設定の DCB

項目	定数	詳細	デフォルト
IR ボーレート (9) OR で設定して下さい 設定した値が有効となります。	IRBPS_96	IR 接続速度を 9600bps に設定可能	設定有効
	IRBPS_192	IR 接続速度を 19200bps に設定可能	設定有効
	IRBPS_384	IR 接続速度を 38400bps に設定可能	設定有効
	IRBPS_576	IR 接続速度を 57600bps に設定可能	設定有効
	IRBPS_1152	IR 接続速度を 115200bps に設定可能	設定有効
最大ターン アラウンドタイム	TURN_500MS	最大ターンアラウンドタイムを 500ms に設定	○
フレームサイズ	FRAME_512B	フレームサイズを 512byte に設定	○
ウィンドウサイズ	WINDOW_1	ウィンドウサイズを 1 フレームウィンドウに設定	○
BOF 数 (9) IR ボーレイトにより比 例して増減します BOF 数は 115.2k の場合です。	BOF_48	BOF を 48 個追加	○
	BOF_24	BOF を 24 個追加	
	BOF_12	BOF を 12 個追加	
	BOF_5	BOF を 5 個追加	
	BOF_3	BOF を 3 個追加	
	BOF_2	BOF を 2 個追加	
	BOF_1	BOF を 1 個追加	
	BOF_0	BOF を 0 個追加	
最小ターン アラウンドタイム	TURN_5MS	最小ターンアラウンドタイムを 5ms に設定	
リンク開放時間 (9) OR で設定して下さい 設定した値が有効となります。	RELEASE_3S	リンクを開放する時間を 3s に設定可能	設定有効
	RELEASE_8S	リンクを開放する時間を 8s に設定可能	設定有効
	RELEASE_12S	リンクを開放する時間を 12s に設定可能	設定有効
	RELEASE_16S	リンクを開放する時間を 16s に設定可能	設定有効
	RELEASE_20S	リンクを開放する時間を 20s に設定可能	設定有効
	RELEASE_25S	リンクを開放する時間を 25s に設定可能	設定有効
	RELEASE_30S	リンクを開放する時間を 30s に設定可能	設定有効
	RELEASE_40S	リンクを開放する時間を 40s に設定可能	設定有効

15.2.21 Ir_Init

IrCOMM を強制終了します。

- 通信用のデバイスおよびリソースのリリース
- 赤外線デバイス電源 OFF
- LB チェック

```
H Ir_Init();
```

パラメータ

なし

戻り値

E_IROK	:正常終了
E_IRNG	:異常終了

15.2.22 Ir_SetWinMode

使用する IrDA のモードを設定します。
IrDA-USB クレードルを使用する際には、必ず PC 接続モードに設定してください。

```
ER Ir_SetWinMode(
   H  mode
);
```

パラメータ

mode

通信モード／Windows の種類

SET_SIR	:本体間接続モード
SET_WIN2K	:PC 接続モード

戻り値

E_OK	:正常終了
E_NG	:異常終了

16.Bluetooth 制御

16.1 基本機能

16.1.1 通信インターフェース

Bluetooth ライブラリは、ハンディターミナル本体に内蔵している Bluetooth モジュールを利用して、他の Bluetooth 機器との接続および通信を行うためのライブラリです。
サポートする通信プロファイルは、シリアルポートプロファイルです。

Bluetooth 機器間で通信を行う場合、Bluetooth 通信機器はピコネットと呼ばれるワイヤレスネットワークを構成し、その中で各 Bluetooth 機器はマスターまたはスレーブのいずれかのモードで動作します。

DT-970 は Bluetooth プリンタと Bluetooth 通信を行います。その場合は DT-970 本体をマスターモード、Bluetooth プリンタをスレーブモードで使用します。

16.1.2 通信手順

DT-970 本体をマスターモードにして通信する場合の概要を示します。

⑼ 通信する Bluetooth 機器のデバイス情報の登録

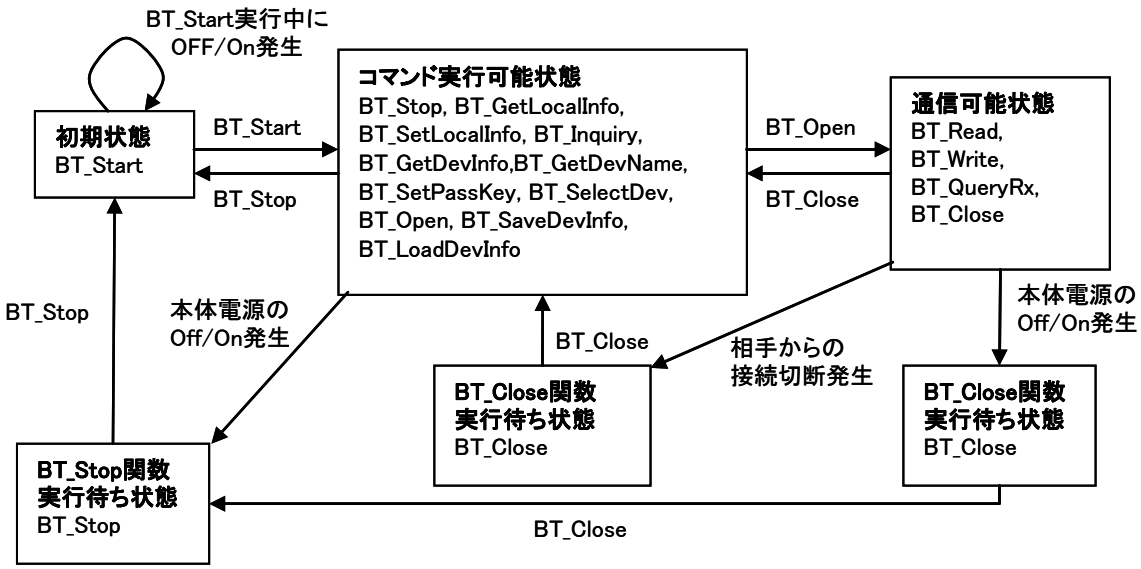
- Bluetooth 機器の Inquiry（問い合わせ）を実行
- Inquiry で発見された Bluetooth 機器の情報を取得
- 取得した Bluetooth 機器情報をファイルに保存

⑵ 通信する Bluetooth 機器の選択および通信の実行

- ファイルから Bluetooth 機器情報を取得
- 通信する Bluetooth 機器を選択
- 選択した Bluetooth 機器との接続および通信を実行

16.2 関数仕様

16.2.1 関数の状態遷移



(1) 初期状態において本体電源の Off/On が発生したとき

BT_Start 関数から関数処理を実行してください。

(2) BT_Start 関数実行中に本体電源の Off/On が発生したとき

BT_Start 関数から関数処理を実行してください。

(3) コマンド実行可能状態において本体電源の Off/On が発生したとき

BT_Stop 関数を実行後、初期状態から関数処理を実行してください。

(4) 通信可能状態において本体電源の Off/On が発生したとき

BT_Close 関数および BT_Stop 関数を実行後、初期状態から関数処理を実行してください。

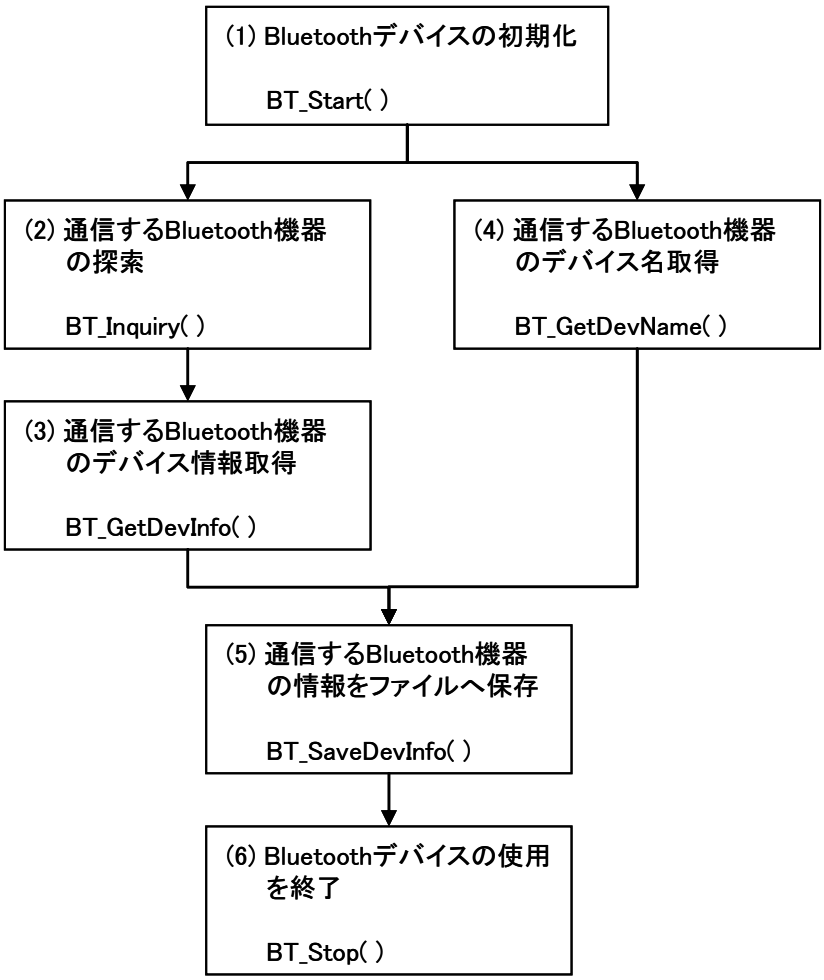
(5) 通信可能状態において通信相手から Bluetooth 接続が切断されたとき

BT_Close 関数を実行後、コマンド実行可能状態から関数処理を実行してください。

※ 上記の状態遷移においては、ユーザが設定したパスキーおよび本体の Bluetooth デバイス名は保存されます。DT-970 本体裏のリセットスイッチを押す等の操作で OS のリセットを実行したとき、パスキーおよび本体の Bluetooth デバイス名は工場出荷時の設定に戻ります。

16.2.2 関数の実行手順

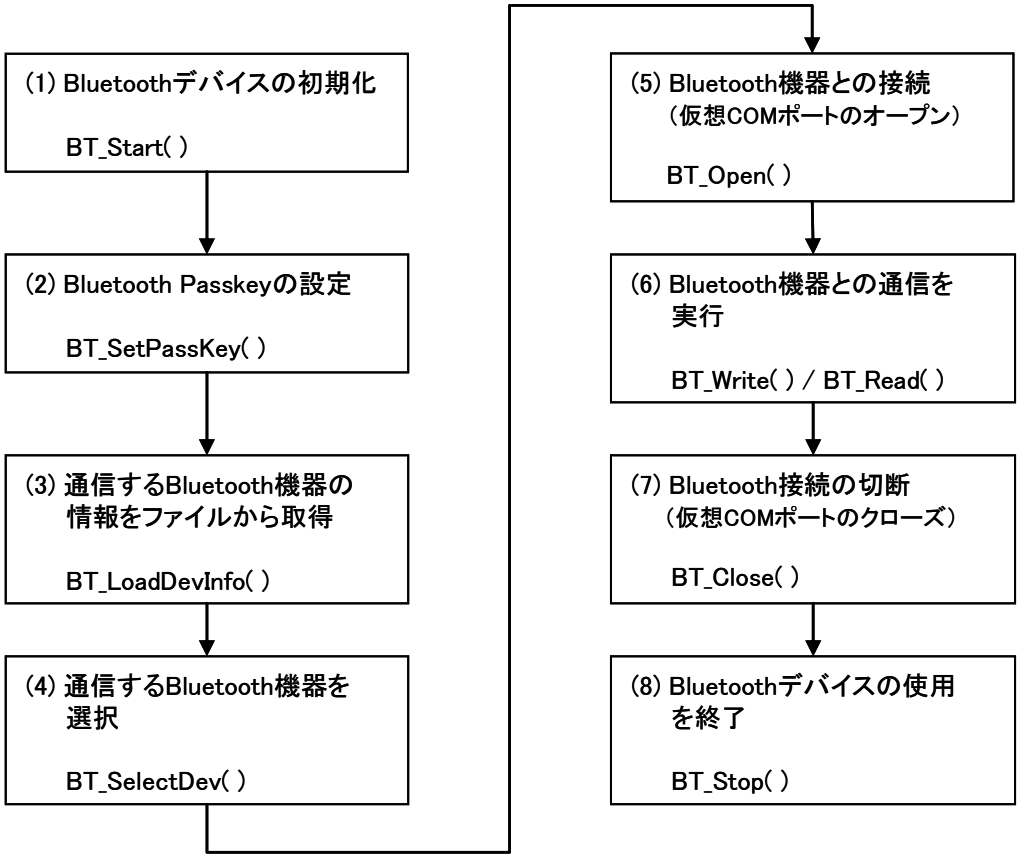
⑼ Bluetooth 機器情報の取得と保存



接続する Bluetooth 機器のアドレスが未知の場合は手順(2)と(3)を、既知の場合は手順(4)を実行してください。

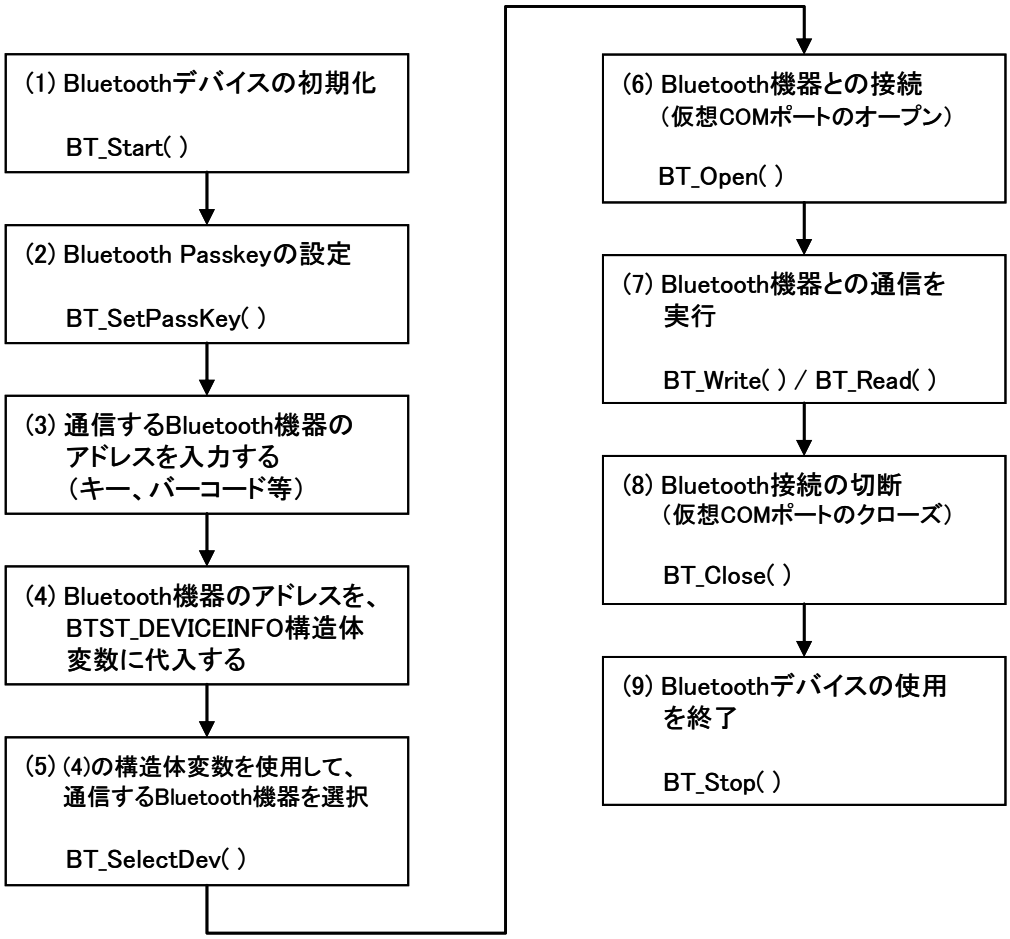
(2) Bluetooth 接続と通信

(1)の手順を実行後、次の手順で Bluetooth 接続と通信を実行することができます。



(3) Bluetooth 接続と通信（BT 機器のアドレスが既知の場合）

接続する Bluetooth 機器のアドレスが既知の場合は、"Bluetooth 機器情報の取得と保存"の手順を省略し、次の手順で Bluetooth 接続と通信を実行することができます。



手順(3)において入力する Bluetooth 機器のアドレスの形式は、"XX:XX:XX:XX:XX:XX"となります。(X は ASCII の 16 進数で、0～9 および A～F です)
上記は、Bluetooth 機能使用開始から終了まで一連の流れを示しています。実際に業務においてプリンタ等に出力する場合、1 枚プリントする度に上記手順を行うと、時間がかかるため、繰り返し部分について次のような手順で高速化できます。

前処理→BT_Open→通信実行（1 枚目）→通信実行（2 枚目）→ ・・・ → BT_Close→後処理

※ ポイント：時間を置かずにプリントする場合は、一度 BT_Open でプリンタと接続したら、業務終了まで BT 接続を維持したままにしておき、プリントデータのみを順次送信する事で、接続/切断にかかる時間を節約できます

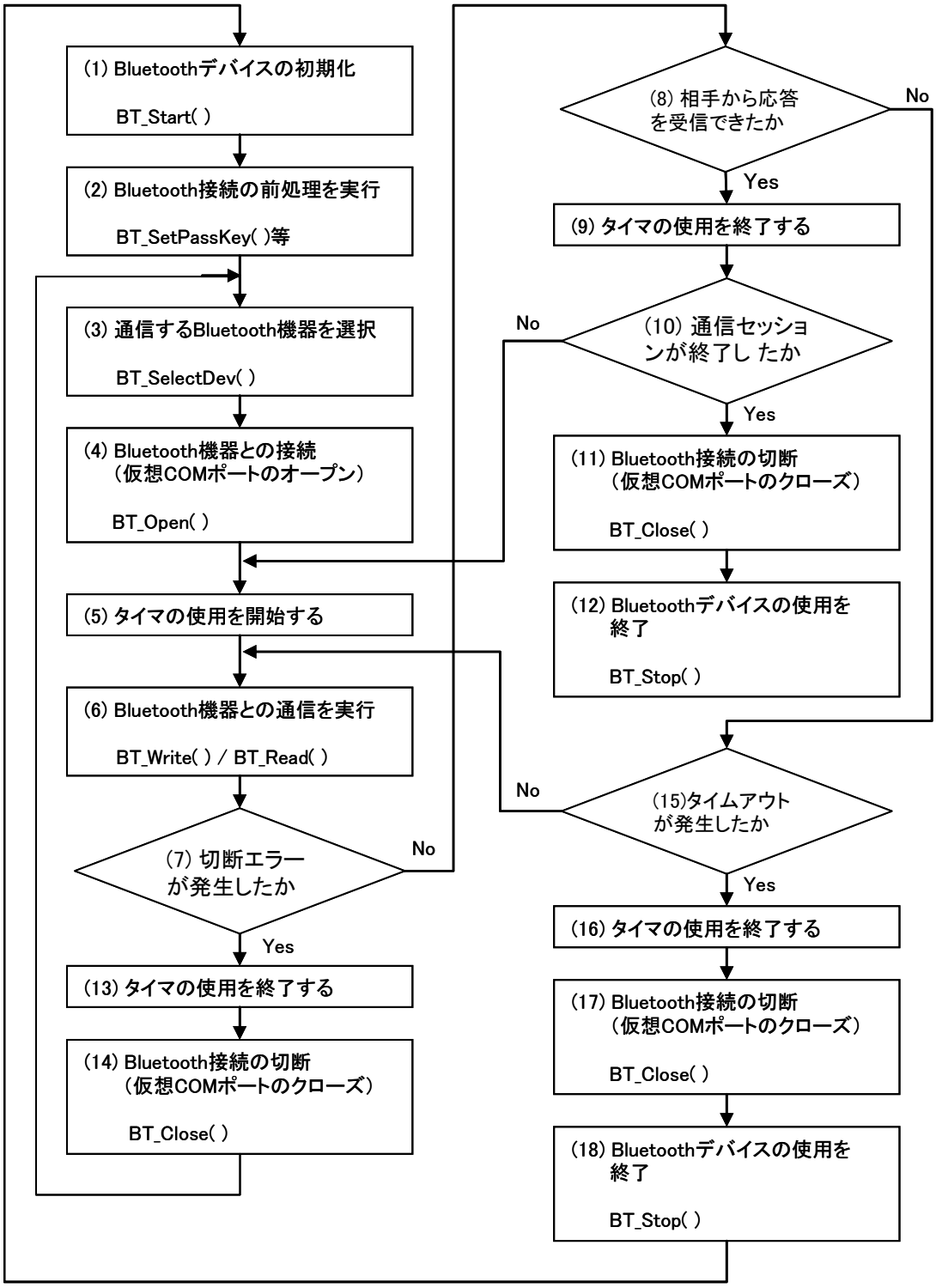
時間が空いてしまう場合は、BT_Open→通信→BT_Close の部分を繰り返す事により BT_Start 関数および BT_Stop 関数にかかる時間を節約できます。ただし、相手から切断された場合、電源 OFF した場合等、関数からエラーが戻ってきた場合や、次項に示すようにプリンタからの応答がない場合には、所定のエラー処理を行ってください。

(4) 接続先の Bluetooth 機器から切断が発生する場合を考慮した通信

通常、接続先の Bluetooth 機器から切断が発生した場合、BT_Write、BT_Read、BT_QueryRx のいずれかの関数を実行したときにエラーが発生します。しかし、Bluetooth ハードウェアの制約上、データ送信中に相手機器から強制切断された場合等、切断が検出できず、切断が発生しても関数でエラーが発生しない場合がまれにあります。この場合、受信データの最終部に INIT の文字列を含むデータが追加されてしまう事があります。

切断が検出できない場合に備え、ユーザアプリケーションにおいて相手機器からの応答データ(ACK など)を監視し、一定時間以上相手機器から応答データが来ない場合は、異常発生としてエラー処理を行うようにして下さい。

接続先の Bluetooth 機器から切断が発生する場合を考慮した関数の実行手順を、次ページに示します。



※ 手順(1)から処理を開始します。手順(2)については、"Bluetooth 接続と通信"を参照してください。
※ 通信先の Bluetooth 機器から切断が発生しない場合は、手順(1)から(12)までが実行されます。
※ 手順(7)において切断エラーを検出した場合、手順(13)と(14)を実行後、手順(3)から処理を再開します。
※ 手順(8)において通信相手から応答(ACK 等)を受信できない場合、通信が切断されているが検出できていない可能性があります。この場合、手順(15)においてタイムアウトが発生したかチェックします。
※ 手順(15)でタイムアウトが発生している場合、通信が切断されているとみなし、手順(16)から(18)までを実行した後、手順(1)から通信処理をやり直します。
※ 使用するタイムアウトの値は、使用する Bluetooth 機器に合わせて、5～10 秒程度に調整してください。

16.3 エラー詳細

エラーステータスはファンクションコールが異常終了したとき、その詳細を示します。

エラー値	BTERR_LOCK
詳細 通信用のデバイスおよびリソースが既にロックされているときに発生します。 Bluetooth ライブラリは他の通信関数とシステム資源を共有していますので先に起動された方に資源を利用する権利があります。このエラーが発生したときは資源が開放されるまで待たなければなりません。	
関数名	主なエラー対処方法
BT_Start	(9) デバイス、リソースが開放されてから再実行して下さい。 ・既に BT_Start 関数が実行されています。BT_Stop 関数を実行してから、BT_Start 関数を再実行して下さい。

エラー値	BTERR_DISCONNECT
詳細 他の Bluetooth 機器と接続していないとき、または他の Bluetooth 機器から接続が切断されたときに発生します。	
関数名	主なエラー対処方法
BT_Close	・他の Bluetooth 機器との接続を確認してください。
BT_Read	
BT_Write	
BT_QueryRx	

エラー値	BTERR_PARAMETER
詳細 関数のパラメータの入力値に誤りがあるとき発生します。	
関数名	主なエラー対処方法
BT_Inquiry	・関数のパラメータの入力値が正しいか確認してください。
BT_GetDevInfo	
BT_GetDevName	
BT_SetPassKey	
BT_SelectDev	
BT_Open	

エラー値	BTERR_BREAK_EVNT
詳細 キー関数の機能を用いて中断キーのイベント登録が行なわれ、当該キー押下によりブレイクイベントの検出がライブラリ関数で行われたときに発生します。	
関数名	主なエラー対処方法
BT_GetLocalInfo	・BT_Stop 関数を実行して終了してください。
BT_SetLocalInfo	
BT_Inquiry	
BT_GetDevInfo	
BT_GetDevName	
BT_SetPassKey	

BT_SelectDev	
BT_Open	
BT_SaveDevInfo	
BT_LoadDevInfo	
BT_Close	・BT_Close 関数および BT_Stop 関数を実行して終了してください。
BT_Read	
BT_Write	
BT_QueryRx	

エラー値	BTERR_LB0
詳細 LB0 イベントの通知が有効に設定されており、LB0 エラー(主電池なし、電池蓋開き)になったときに発生します。	
関数名	主なエラー対処方法
BT_Start	・関数を再実行してください。
BT_GetLocalInfo	(9) BT_Stop 関数を実行して終了してください。 ・処理を再実行する場合は、BT_Start 関数から実行してください。
BT_SetLocalInfo	
BT_Inquiry	
BT_GetDevInfo	
BT_GetDevName	
BT_SetPassKey	
BT_SelectDev	
BT_Open	
BT_SaveDevInfo	
BT_LoadDevInfo	
BT_Close	(9) BT_Close 関数および BT_Stop 関数を実行して終了してください。 ・処理を再実行する場合は、BT_Start 関数から実行してください。
BT_Read	
BT_Write	
BT_QueryRx	

エラー値	BTERR_LB1
詳細 LB1 イベントの通知が有効に設定されており、LB1 エラー(主電池電圧低下)になったときに発生します。	
関数名	主なエラー対処方法
BT_Start	・電池交換後、関数を再実行してください。
BT_GetLocalInfo	(9) BT_Stop 関数を実行して終了してください。 ・電池交換後、BT_Start 関数から処理を実行してください。
BT_SetLocalInfo	
BT_Inquiry	
BT_GetDevInfo	
BT_GetDevName	
BT_SetPassKey	
BT_SelectDev	
BT_Open	
BT_SaveDevInfo	
BT_LoadDevInfo	
BT_Close	(9) BT_Close 関数および BT_Stop 関数を実行して終了してください。 ・電池交換後、BT_Start 関数から処理を実行してください。
BT_Read	

BT_Write	
BT_QueryRx	

エラー値	BTERR_LB2
詳細 LB2 イベントの通知が有効に設定されており、LB2 エラー(副電池電圧なし)になったときに発生します。	
関数名	主なエラー対処方法
BT_Start	・電池交換後、関数を再実行してください。
BT_GetLocalInfo BT_SetLocalInfo BT_Inquiry BT_GetDevInfo BT_GetDevName BT_SetPassKey BT_SelectDev BT_Open BT_SaveDevInfo BT_LoadDevInfo	(9) BT_Stop 関数を実行して終了してください。 ・電池交換後、BT_Start 関数から処理を実行してください。
BT_Close BT_Read BT_Write BT_QueryRx	(9) BT_Close 関数および BT_Stop 関数を実行して終了してください。 ・電池交換後、BT_Start 関数から処理を実行してください。

エラー値	BTERR_LB4
詳細 LB4 イベントの通知が有効に設定されており、LB4 エラー(APO 発生)になったときに発生します。 通知が有効に設定されているときは電源 OFF しませんので、アプリケーションが責任を持つ必要があります。	
関数名	主なエラー対処方法
BT_Start	・電源 ON 後、関数を再実行してください。
BT_GetLocalInfo BT_SetLocalInfo BT_Inquiry BT_GetDevInfo BT_GetDevName BT_SetPassKey BT_SelectDev BT_Open BT_SaveDevInfo BT_LoadDevInfo	(9) 電源 ON 後、BT_Stop 関数を実行して終了してください。 ・処理を再実行する場合は、BT_Start 関数から実行してください。
BT_Close BT_Read BT_Write BT_QueryRx	(9) 電源 ON 後、BT_Close 関数および BT_Stop 関数を実行して終了してください。 ・処理を再実行する場合は、BT_Start 関数から実行してください。

エラー値	BTERR_LB5
詳細 LB5 イベントの通知が有効に設定されており、LB5 エラー(OFF キー押下による電源 OFF)になったときに発生します。 通知が有効に設定されているときは電源 OFF しませんので、アプリケーションが責任を持つ必要があります。	
関数名	主なエラー対処方法
BT_Start	・電源 ON 後、関数を再実行してください。
BT_GetLocalInfo BT_SetLocalInfo BT_Inquiry BT_GetDevInfo BT_GetDevName BT_SetPassKey BT_SelectDev BT_Open BT_SaveDevInfo BT_LoadDevInfo	(9) 電源 ON 後、BT_Stop 関数を実行して終了してください。 ・処理を再実行する場合は、BT_Start 関数から実行してください。
BT_Close BT_Read BT_Write BT_QueryRx	(9) 電源 ON 後、BT_Close 関数および BT_Stop 関数を実行して終了してください。 ・処理を再実行する場合は、BT_Start 関数から実行してください。

エラー値	BTERR_NOTSTART
詳細 BT_Start 関数を実行しないで他の関数を実行したときに発生します。	
関数名	主なエラー対処方法
BT_Stop BT_GetLocalInfo BT_SetLocalInfo BT_Inquiry BT_GetDevInfo BT_GetDevName BT_SetPassKey BT_SelectDev BT_Open BT_Close BT_Read BT_Write BT_QueryRx BT_SaveDevInfo BT_LoadDevInfo	・BT_Start 関数を実行してから、他の関数を実行してください。

エラー値	BTERR_TIMEOUT
詳細 関数を実行してタイムアウトが起きた場合に発生します。	
関数名	主なエラー対処方法
BT_GetLocalInfo	・関数を再実行してください。
BT_SetLocalInfo	
BT_GetDevInfo	
BT_GetDevName	
BT_SetPassKey	
BT_Open	・関数を再実行するか、接続時間を増やしてください。

エラー値	BTERR_PARITY
詳細 Bluetooth ハードウェア上のシリアル通信でパリティエラーとなったときに発生します。	
関数名	主なエラー対処方法
BT_GetLocalInfo	・BT_Stop 関数を実行して終了してください。
BT_SetLocalInfo	
BT_Inquiry	
BT_GetDevInfo	
BT_GetDevName	
BT_SetPassKey	
BT_SelectDev	
BT_Open	
BT_Close	・BT_Close 関数および BT_Stop 関数を実行して終了してください。
BT_Read	
BT_Write	
BT_QueryRx	

エラー値	BTERR_OVERRUN
詳細 Bluetooth ハードウェア上のシリアル通信でオーバーランエラーとなったときに発生します。	
関数名	主なエラー対処方法
BT_GetLocalInfo	・BT_Stop 関数を実行して終了してください。
BT_SetLocalInfo	
BT_Inquiry	
BT_GetDevInfo	
BT_GetDevName	
BT_SetPassKey	
BT_SelectDev	
BT_Open	
BT_Close	・BT_Close 関数および BT_Stop 関数を実行して終了してください。
BT_Read	
BT_Write	
BT_QueryRx	

エラー値	BTERR_FRAMING
詳細 Bluetooth ハードウェア上のシリアル通信でフレーミングエラーとなったときに発生します。	
関数名	主なエラー対処方法
BT_GetLocalInfo	・BT_Stop 関数を実行して終了してください。
BT_SetLocalInfo	
BT_Inquiry	
BT_GetDevInfo	
BT_GetDevName	
BT_SetPassKey	
BT_SelectDev	
BT_Open	
BT_Close	・BT_Close 関数および BT_Stop 関数を実行して終了してください。
BT_Read	
BT_Write	
BT_QueryRx	

エラー値	BTERR_TRANSFER
詳細 Bluetooth ハードウェア上のシリアル通信で予期しない応答が戻ったときに発生します。	
関数名	主なエラー対処方法
BT_GetLocalInfo	・BT_Stop 関数を実行して終了してください。
BT_SetLocalInfo	
BT_Inquiry	
BT_GetDevInfo	
BT_GetDevName	
BT_SetPassKey	
BT_SelectDev	
BT_Open	
BT_Close	・BT_Close 関数および BT_Stop 関数を実行して終了してください。
BT_Read	
BT_Write	
BT_QueryRx	

エラー値	BTERR_FILEOPEN
詳細 Bluetooth デバイス情報保存用のファイルがオープンできなかったときに発生します。	
関数名	主なエラー対処方法
BT_SaveDevInfo	・指定したファイルが存在するか、または他のアプリから使用されていないか確認してください。
BT_LoadDevInfo	

エラー値	BTERR_FILEACCESS
詳細 Bluetooth デバイス情報保存用のファイルの読込または書込ができなかったときに発生します。	
関数名	主なエラー対処方法
BT_SaveDevInfo BT_LoadDevInfo	・指定したファイルの属性と、ファイルが他のアプリから使用されていないか確認してください。

エラー値	BTERR_NAK00
詳細 Bluetooth モジュールから NAK00 エラーが戻ったときに発生します。 Bluetooth モジュールへ送信したコマンドが正しくありません。	
関数名	主なエラー対処方法
BT_GetLocalInfo BT_SetLocalInfo BT_Inquiry BT_GetDevInfo BT_GetDevName BT_SetPassKey BT_SelectDev BT_Open BT_Close	・関数を再実行してください。

エラー値	BTERR_NAK01
詳細 Bluetooth モジュールから NAK01 エラーが戻ったときに発生します。 Bluetooth モジュールへ送信したコマンドのパラメータが正しくありません。	
関数名	主なエラー対処方法
BT_GetLocalInfo BT_SetLocalInfo BT_Inquiry BT_GetDevInfo BT_GetDevName BT_SetPassKey BT_SelectDev BT_Open BT_Close	・関数のパラメータの入力値が正しいか確認してください。

エラー値	BTERR_NAK02
詳細 Bluetooth モジュールから NAK02 エラーが戻ったときに発生します。 Bluetooth 接続実行時の PassKey 交換に失敗しました。	
関数名	主なエラー対処方法
BT_Open	・BT_SetPassKey 関数を実行して正しい PassKey を設定後、BT_Open 関数を再実行してください。

エラー値	BTERR_NAK03
詳細 Bluetooth モジュールから NAK03 エラーが戻ったときに発生します。 Bluetooth デバイスの探索でデバイスが発見されませんでした。	
関数名	主なエラー対処方法
BT_Inquiry	・他の Bluetooth デバイスが探索に応答できる状態であるか(電源が入っているか等)を確認してください。

エラー値	BTERR_NAK04
詳細 Bluetooth モジュールから NAK04 エラーが戻ったときに発生します。 他の Bluetooth デバイスとの接続に失敗しました。	
関数名	主なエラー対処方法
BT_Open	・他の Bluetooth デバイスが接続できる状態であるか(電源が入っているか等)を確認してください。

エラー値	BTERR_NAK05
詳細 Bluetooth モジュールから NAK05 エラーが戻ったときに発生します。 PassKey の内容が正しくありません。	
関数名	主なエラー対処方法
BT_SetPassKey	・設定する PassKey の内容を確認してください

エラー値	BTERR_NAK06
詳細 Bluetooth モジュールから NAK06 エラーが戻ったときに発生します。 Bluetooth モジュールが故障している可能性があります(通常は発生しません) 。	
関数名	主なエラー対処方法
BT_GetLocalInfo BT_SetLocalInfo BT_Inquiry BT_GetDevInfo BT_GetDevName BT_SetPassKey BT_SelectDev BT_Open BT_Close	⑼ 関数を再実行してください。 ・関数を再実行しても同じエラーが発生する場合は、Bluetooth ハードウェアの状態を確認してください。

エラー値	BTERR_NAK08
詳細 Bluetooth モジュールから NAK08 エラーが戻ったときに発生します。 Bluetooth モジュールの状態が正しくありません。	
関数名	主なエラー対処方法
BT_Close	・BT_Stop 関数を実行してください。

エラー値	BTERR_NAK09
詳細 Bluetooth モジュールから NAK09 エラーが戻ったときに発生します。 Bluetooth モジュールの状態が正しくありません。	
関数名	主なエラー対処方法
BT_Close	・BT_Stop 関数を実行してください。

エラー値	BTERR_NAK10
詳細 Bluetooth モジュールから NAK10 エラーが戻ったときに発生します。 Bluetooth モジュールが故障している可能性があります(通常は発生しません) 。	
関数名	主なエラー対処方法
BT_GetLocalInfo BT_SetLocalInfo BT_Inquiry BT_GetDevInfo BT_GetDevName BT_SetPassKey BT_SelectDev BT_Open BT_Close	(9) 関数を再実行してください。 ・関数を再実行しても同じエラーが発生する場合は、Bluetooth ハードウェアの状態を確認してください。

エラー値	BTERR_NAK11
詳細 Bluetooth モジュールから NAK11 エラーが戻ったときに発生します。 設定する Bluetooth デバイス名が長すぎます。	
関数名	主なエラー対処方法
BT_SetLocalInfo	・設定する Bluetooth デバイス名の長さを確認してください。

エラー値	BTERR_NAK12
詳細 Bluetooth モジュールから NAK12 エラーが戻ったときに発生します。 Bluetooth モジュールの状態が正しくありません。	
関数名	主なエラー対処方法
BT_Close	・BT_Stop 関数を実行してください。

エラー値	BTERR_NAK13
詳細 Bluetooth モジュールから NAK13 エラーが戻ったときに発生します。 Bluetooth モジュールが故障している可能性があります(通常は発生しません) 。	
関数名	主なエラー対処方法
BT_GetLocalInfo BT_SetLocalInfo BT_Inquiry BT_GetDevInfo BT_GetDevName	(9) 関数を再実行してください。 ・関数を再実行しても同じエラーが発生する場合は、Bluetooth ハードウェアの状態を確認してください。

BT_SetPassKey	
BT_SelectDev	
BT_Open	
BT_Close	

エラー値	BTERR_NAK14
詳細 Bluetooth モジュールから NAK14 エラーが戻ったときに発生します。 Bluetooth モジュールに送信したコマンドが実行されませんでした。	
関数名	主なエラー対処方法
BT_GetLocalInfo	・関数を再実行してください。
BT_SetLocalInfo	
BT_Inquiry	
BT_GetDevInfo	
BT_GetDevName	
BT_SetPassKey	
BT_SelectDev	
BT_Open	
BT_Close	

エラー値	BTERR_NAKMINUS1
詳細 Bluetooth モジュールから NAK-1 エラーが戻ったときに発生します。 Bluetooth モジュールが故障している可能性があります(通常は発生しません) 。	
関数名	主なエラー対処方法
BT_GetLocalInfo	(9) 関数を再実行してください。 ・関数を再実行しても同じエラーが発生する場合は、Bluetooth ハードウェアの状態を確認してください。
BT_SetLocalInfo	
BT_Inquiry	
BT_GetDevInfo	
BT_GetDevName	
BT_SetPassKey	
BT_SelectDev	
BT_Open	
BT_Close	

16.4 関数リファレンス

ファンクション詳細を次ページより示します。

関数	機能概要
BT_Start	Bluetooth 通信機能の使用を開始
BT_Stop	Bluetooth 通信機能の使用を終了
BT_SelectProfile	使用するプロファイルの指定
BT_GetLocalInfo	本体のデバイス情報の取得
BT_SetLocalInfo	本体のデバイス情報の設定
BT_Inquiry	Bluetooth 機器の問い合わせ
BT_GetDevInfo	Bluetooth 機器のデバイス情報の取得
BT_GetDevName	Bluetooth 機器のデバイス名の取得
BT_SelectDev	接続する Bluetooth 機器の指定
BT_GetPassKey	Bluetooth パスキーの取得
BT_SetPassKey	Bluetooth パスキーの設定
BT_Open	Bluetooth 通信の開始
BT_Close	Bluetooth 通信の終了
BT_Read	Bluetooth 通信のデータ受信
BT_Write	Bluetooth 通信のデータ送信
BT_QueryRx	読込可能なデータ数の取得
BT_SaveDevInfo	Bluetooth 機器のデバイス情報の保存
BT_LoadDevInfo	Bluetooth 機器のデバイス情報の読み出し
BT_Err_Get	エラー値の取得
BT_HID_Keycode	HID プロファイルのキーコードを送信

16.4.1 BT_Start

Bluetooth 通信機能が使用できる状態にします。

```
H BT_Start();
```

パラメータ

なし

戻り値

E_BTOK	:正常終了
E_BTNG	:異常終了

解説

Bluetooth は IrDA と排他で使用します。
このため、本関数を実行すると IrDA 通信はできません。
Bluetooth 通信終了後に IrDA 通信を行う場合は、BT_Stop 関数を実行してください。

16.4.2 BT_Stop

Bluetooth 通信機能の使用を終了します。

```
H BT_Stop();
```

パラメータ

なし

戻り値

E_BTOK	:正常終了
E_BTNG	:異常終了

解説

Bluetooth 通信終了後に IrDA 通信を行う場合、本関数を実行してください。

16.4.3 BT_SelectProfile

使用するプロファイルを指定します。

```
H BT_SelectProfile(
  B   Profile
);
```

パラメータ

Profile

デバイス情報が格納されている構造体変数のポインタ

BT_Start 関数の後に呼び出してください。呼び出されない場合のデフォルトは仮想シリアルです。

BT_PROFILE_SERIAL	:仮想シリアル
BT_PROFILE_DIALUP	:ダイヤルアップ
BT_PROFILE_HID	:HID

戻り値

E_BTOK	:正常終了
E_BTNG	:異常終了

16.4.4 BT_GetLocalInfo

DT-970 本体のデバイス情報を取得します。

```
H BT_GetLocalInfo(
   BT_LOCALINFO    *s_localinfo
);
```

パラメータ

s_localinfo

本体のデバイス情報を格納する構造体変数のポインタ

```
typedef struct {
   B   LocalAddr[18];   : 本体の Bluetooth アドレス
                          形式は"XX:XX:XX:XX:XX:XX"となります
                          X は ASCII の 16 進数(0～9 および A～F)
   B   LocalName[82];   : 本体の Bluetooth デバイス名
   H   LocalClass;      : 本体の Bluetooth デバイスクラス
} BT_LOCALINFO
```

戻り値

E_BTOK	:正常終了
E_BTNG	:異常終了

解説

BT_SetLocalInfo 関数を実行する場合は、先に本関数を実行し、本体のデバイス情報を取得してください。その後、デバイス名のパラメータを変更して、BT_SetLocalInfo 関数を実行してください。
取得可能な Bluetooth デバイス名の長さは最大で 81 文字です。

参照

BT_LOCALINFO 構造体

16.4.5 BT_SetLocalInfo

DT-970 本体のデバイス情報を設定します。

```
H BT_SetLocalInfo(
   BT_LOCALINFO    *s_localinfo
);
```

パラメータ

s_localinfo

本体のデバイス情報を設定する構造体変数のポインタ

```
typedef struct {
   B   LocalAddr[18];   : 本体の Bluetooth アドレス
                          形式は"XX:XX:XX:XX:XX:XX"となります
                          X は ASCII の 16 進数(0～9 および A～F)
                          (値は変更できません)
   B   LocalName[82];   : 本体の Bluetooth デバイス名
   H   LocalClass;      : 本体の Bluetooth デバイスクラス
} BT_LOCALINFO
```

戻り値

E_BTOK	:正常終了
E_BTNG	:異常終了

解説

本関数を実行する前に、BT_GetLocalInfo 関数を実行し、本体のデバイス情報を取得してください。
その後、デバイス名、デバイスクラスのパラメータを変更して、本関数を実行してください。
Bluetooth デバイス名に使用できる文字は ASCII のみです。半角カナや 2 バイト文字は使用できません。設定可能な Bluetooth デバイス名の長さは最大で 81 文字です。
Bluetooth デバイスクラスには、次の値を設定してください。

iOS に HID で接続する場合	:0x0540
iOS 以外の場合	:0

本関数で設定したデバイス情報はグローバル領域に保存され、OS がリセットされるまで保持されます。

参照

BT_LOCALINFO 構造体

16.4.6 BT_Inquiry

DT-970 の周囲にある Bluetooth 機器の探索(Inquiry)を実行します。

```
H BT_Inquiry(
   B   *number,
   H   sec
);
```

パラメータ

number
発見された Bluetooth 機器の数を格納する変数のポインタ

sec
Bluetooth 機器を探索する時間（1～20 秒）

戻り値

E_BTOK　　:正常終了
E_BTNG　　:異常終了

解説

本関数を使用して発見可能な Bluetooth 機器の最大数は 9 です。
本関数を実行中に中断キーが押されるか、電源が OFF された場合は、エラーが発生して関数処理から抜けます。
Bluetooth 機器が発見できなかった場合は、エラーが発生して関数処理から抜けます。

16.4.7 BT_GetDevInfo

DT-970 の周囲にある Bluetooth 機器のデバイス情報を取得します。

```
H BT_GetDevInfo(
   BT_DEVINFO     **s_devinfo,
   B              number
);
```

パラメータ

s_devinfo

デバイス情報が格納されている構造体変数のポインタ

```
typedef struct {
   UW  ErrFlag;          : エラーフラグ
                           (Bluetooth デバイス名が取得できなかった
                           場合にエラーコードがセットされます)
   B   DevAddr[18];      : Bluetooth アドレス
                           形式は"XX:XX:XX:XX:XX:XX"となります
                           X は ASCII の 16 進数(0～9 および A～F)
   B   DevName[82];      : Bluetooth デバイス名
   H   DevClass;         : Bluetooth デバイスクラス
} BT_DEVINFO;
```

number

BT_Inquiry 関数を実行して発見された Bluetooth 機器の数(1～9)

戻り値

E_BTOK	:正常終了
E_BTNG	:異常終了

解説

本関数を使用して取得可能なデバイス情報の最大数は 9 です。
本関数を実行中に中断キーが押されるか、電源が OFF された場合は、エラーが発生して関数処理から抜けます。
本関数は次の例のように引数を指定して実行します。

```
BT_DEVINFO *devinfo;                        /* BT_DEVINFO *  型の変数を用意する  */
ercd = BT_GetDevInfo(&devinfo, number);     /*  変数のポインタを引数として指定する  */
```

参照

BT_DEVINFO 構造体

16.4.8 BT_GetDevName

Bluetooth 機器のデバイスアドレスを指定して、Bluetooth 機器のデバイス名を取得します。

```
H BT_GetDevName(
   BT_DEVINFO     *s_devinfo,
   B              *Address
);
```

パラメータ

s_devinfo

デバイス情報を格納する構造体変数のポインタ

```
typedef struct {
   UW  ErrFlag;          : エラーフラグ
                           (本関数ではセットされません)
   B   DevAddr[18];      : Bluetooth アドレス
                           形式は"XX:XX:XX:XX:XX:XX"となります
                           X は ASCII の 16 進数(0～9 および A～F)
   B   DevName[82];      : Bluetooth デバイス名
   H   DevClass;         : Bluetooth デバイスクラス
                           (本関数ではセットされません)
} BT_DEVINFO;
```

Address

デバイスアドレスを格納している変数へのポインタ
デバイスアドレスの形式は"XX:XX:XX:XX:XX:XX"です
X は ASCII の 16 進数(0～9 および A～F)

戻り値

E_BTOK	:正常終了
E_BTNG	:異常終了

解説

本関数を実行中に中断キーが押されるか、電源が OFF された場合は、エラーが発生して関数処理から抜けます。

参照

BT_DEVINFO 構造体

16.4.9 BT_SelectDev

接続する Bluetooth 機器を指定します。

```
H BT_SelectDev(
   BT_DEVINFO     *s_devinfo,
   B              Security
);
```

パラメータ

s_devinfo

デバイス情報が格納されている構造体変数のポインタ

```
typedef struct {
   UW  ErrFlag;          : エラーフラグ
   B   DevAddr[18];      : Bluetooth アドレス
                           形式は"XX:XX:XX:XX:XX:XX"となります
                           X は ASCII の 16 進数(0～9 および A～F)
   B   DevName[82];      : Bluetooth デバイス名
   H   DevClass;         : Bluetooth デバイスクラス
} BT_DEVINFO;
```

Security

セキュリティモード

BT_SEC_NORMAL	:BT 接続時に認証および暗号化を使用しない
BT_SEC_AUTH	:BT 接続時に認証のみを有効にする
BT_SEC_ENCRYPT	:BT 接続時に認証と暗号化を有効にする
BT_SEC_SSPJST	:BT 接続時に SSP(Just Works)を使用する

戻り値

E_BTOK	:正常終了
E_BTNG	:異常終了

解説

本関数に指定するデバイス情報は、BT_GetDevInfo、 BT_GetDevName、BT_LoadDevInfo のうちのいずれかの関数を使用して、取得しておいてください。
本関数で認証または暗号化を有効に設定する場合は、BT_SetPassKey 関数を実行してパスキーを設定しておく必要があります。
iOS に HID で接続する場合には、BT_SEC_SSPJST にしてください。

参照

BT_DEVINFO 構造体

16.4.10 BT_GetPassKey

Bluetooth パスキーを取得します。

```
H BT_GetPassKey(
  B   *PassKey
);
```

パラメータ

PassKey
パスキーを格納する変数のポインタを指定します。16 バイト以上の領域を確保してください。

戻り値

E_BTOK　　:正常終了
E_BTNG　　:異常終了

16.4.11 BT_SetPassKey

Bluetooth パスキーを設定します。

```
H BT_SetPassKey(
   B   *PassKey
);
```

パラメータ

PassKey
パスキーを格納している変数へのポインタ

戻り値

E_BTOK　　:正常終了
E_BTNG　　:異常終了

解説

パスキー設定に使用できる文字は ASCII のみです。
半角カナや 2 バイト文字は使用できません。
設定可能なパスキーの長さは最大で 16 文字です。
本関数で設定したパスキーはグローバル領域に保存され、OS がリセットされるまで保持されます。

16.4.12 BT_Open

Bluetooth 接続を行い、Bluetooth 通信を開始します。

```
H BT_Open(
   H   sec
);
```

パラメータ

sec
接続タイムアウト(1～3600 秒)

戻り値

E_BTOK	:正常終了
E_BTNG	:異常終了

解説

本関数を実行する前に、BT_SelectDev 関数を実行してください。
BT_SelectDev 関数実行時に認証または暗号化を有効に設定した場合は、本関数を実行する前に BT_SetPassKey 関数を実行して、パスキーを設定しておく必要があります。
本関数を実行中に中断キーが押されるか、電源が OFF された場合は、エラーが発生して関数処理から抜けます。

16.4.13 BT_Close

Bluetooth 接続を切断し、Bluetooth 通信を終了します。

```
H BT_Close();
```

パラメータ

なし

戻り値

E_BTOK	:正常終了
E_BTNG	:異常終了

解説

通信相手の BT 機器側から切断を実行した場合も、必ず本関数を実行してください。

16.4.14 BT_Read

Bluetooth 通信において、受信データの読み込みを実行します。

```
H BT_Read(
   B      *buff,
   UH     ReadSize,
   UH     *GetSize
);
```

パラメータ

buff
受信データを格納するバッファのポインタ

ReadSize
受信データを格納するバッファのサイズ(バイト数)

GetSize
読み込みデータ数(受信バッファから読み込みできたバイト数)

戻り値

E_BTOK	:正常終了
E_BTNG	:異常終了

解説

本関数を実行中に中断キーが押されるか、電源が OFF された場合は、エラーが発生して関数処理から抜けます。

16.4.15 BT_Write

Bluetooth 通信において、送信データの書き込みを実行します。

```
H BT_Write(
   B      *buff,
   UH     WriteSize,
   UH     *PutSize
);
```

パラメータ

buff
送信データを格納するバッファのポインタ

WriteSize
送信データ数（送信バッファに書き込むバイト数）

PutSize
書き込みデータ数（送信バッファに書き込みできたバイト数）

戻り値

E_BTOK	:正常終了
E_BTNG	:異常終了
E_NG	:異常終了（補足を参照してください）

解説

本関数を実行中に中断キーが押されるか、電源が OFF された場合は、エラーが発生して関数処理から抜けます。

補足

本関数を実行中に LB が発生した場合は、E_NG が返ります。
また、pwr_inhabit 関数で電源の通知を有効にした後に、LB が発生した状態で BT_Write 関数を実行すると E_NG が返ります。
それ以外の場合は E_BTNG が返ります。

16.4.16 BT_QueryRx

受信バッファより読み込み可能なデータ数の問い合わせを実行します。

```
H BT_QueryRx(
   H   *RcvDataSize
);
```

パラメータ

RcvDataSize
読み込み可能なデータ数(バイト)

戻り値

E_BTOK　　:正常終了
E_BTNG　　:異常終了

16.4.17 BT_SaveDevInfo

取得した Bluetooth 機器のデバイス情報をファイルに保存します。

```
H BT_SaveDevInfo(
   BT_DEVINFO     *s_devinfo,
   B              *filename
);
```

パラメータ

s_devinfo

デバイス情報を格納している構造体変数のポインタ

```
typedef struct {
   H   ErrFlag;          : エラーフラグ
   B   DevAddr[18];      : Bluetooth アドレス
                           形式は"XX:XX:XX:XX:XX:XX"となります
                           X は ASCII の 16 進数(0～9 および A～F)
   B   DevName[82];      : Bluetooth デバイス名
   H   DevClass;         : Bluetooth デバイスクラス
} BT_DEVINFO;
```

filename

保存先のファイル名を格納している変数のポインタ

戻り値

E_BTOK	:正常終了
E_BTNG	:異常終了

解説

エラーフラグの内容は、ファイルに保存しません。

参照

BT_DEVINFO 構造体

16.4.18 BT_LoadDevInfo

ファイルに保存した Bluetooth 機器のデバイス情報を読み出します。

```
H BT_LoadDevInfo(
   BT_DEVINFO     *s_devinfo,
   B               *filename
);
```

パラメータ

s_devinfo
デバイス情報を格納している構造体変数のポインタ

```
typedef struct {
      H    ErrFlag;          : エラーフラグ
      B    DevAddr[18];      : Bluetooth アドレス
                             形式は"XX:XX:XX:XX:XX:XX"となります
                             X は ASCII の 16 進数(0～9 および A～F)
      B    DevName[82];      : Bluetooth デバイス名
      H    DevClass;         : Bluetooth デバイスクラス
} BT_DEVINFO;
```

filename
保存元のファイル名を格納している変数のポインタ

戻り値

E_BTOK　　:正常終了
E_BTNG　　:異常終了

解説

エラーフラグの内容は、ファイルから読み出されません。

参照

BT_DEVINFO 構造体

16.4.19 BT_Err_Get

エラー値を取得します。また取得後にエラー値をクリアします。

```
UW BT_Err_Get();
```

パラメータ

なし

戻り値

エラーコードを返します。

解説

エラーコードについては、「16.3エラー詳細」を参照してください。

16.4.20 BT_HID_Keycode

HID プロファイルのキーコードを送信します。

```
H BT_HID_Keycode(
  UB  *puckeycode
);
```

パラメータ

puckeycode
相手に送信するキーコードを設定します。

戻り値

E_BTOK　　:正常終了
E_BTNG　　:異常終了

解説

詳細は、「14.1.2 データ出力」を参照してください。

17.LAN 制御

17.1 概要

TCP/IP 通信を行うための関数群で、以下の 3 つに分類されます。

1) lan_xxx　　:IP アドレス選択や設定を行なうための関数群です。
2) lancradle_xxx　:LAN クレードルに IP アドレス設定を行うための関数群です。
3) net_xxx　　:ソケット通信を行なうための関数群および補助関数群です。
　　BSD ソケットに準じます。

まず、lan_xxx 関数にて、IP アドレス等の設定を行い、次に net_xxx 関数で実際のソケット通信を行います。LAN クレードルに IP アドレス設定を行うには、事前に lancradke_xxx 関数を使用して設定を行っておきます。

17.2 関数リファレンス

ファンクション詳細を次ページより示します。

関数	機能概要
lan_setParam_IPmode	DHCP の有効／無効等の IP 選択
lan_setParam_IPaddr	IP アドレスの設定
lan_setParam_subnet	サブネットマスクの設定
lan_setParam_gateway	デフォルトゲートウェイの設定
lan_setParam_DNS	DNS サーバアドレスの設定

関数	機能概要
lancradle_get_IPaddr	LAN クレードルの IP アドレス／サブネットマスク／デフォルトゲートウェイの取得
lancradle_set_IPaddr	LAN クレードルの IP アドレス／サブネットマスク／デフォルトゲートウェイの設定

ソケット API

関数	機能概要
net_socket	ソケットの作成
net_bind	ローカルアドレスの付与
net_connect	相手先との接続
net_listen	接続の待機
net_accept	TCP 接続の受け付け
net_send	データの送信
net_sendto	データの送信（相手先指定）
net_recv	データの受信
net_recvfrom	データの受信（送信元情報取得）
net_shutdown	通信の停止
net_close	ソケットの開放
net_select	イベント待ち
net_getsockopt	ソケットオプションの取得
net_setsockopt	ソケットオプションの設定
net_getsockerr	エラー番号の取得
net_inet_aton	文字列を IPv4 アドレスに変換
net_inet_addr	文字列を IPv4 アドレスに変換
net_inet_ntoa	IPv4 アドレスを文字列に変換

その他の API

関数	機能概要
net_tcpip_wai_rdy	プロトコルスタックの起動待ち
net_ascii_to_ipaddr	IP アドレス文字列を unsigned long 型に変換
net_ipaddr_to_ascii	unsigned long 型の IP アドレスを文字列に変換
net_byte4_to_long	char 型配列の IP アドレスを unsigned long 型の IP アドレスに変換
net_long_to_byte4	unsigned long 型の IP アドレスを unsigned char 型配列の IP アドレスに変換
net_ping_send	ping の送信
net_get_myip_info	自 IP アドレス情報の取得
net_get_ipaddr	DNS を使用してホスト名から IP アドレスを取得

17.2.1 lan_setParam_IPmode

IP アドレスの設定方法を設定します。

```
H lan_setParam_IPmode(
   long   id,
   B      mode
);
```

パラメータ

id
IF を指定します。(0 を指定してください)

mode
IPMODE_MANUAL :ユーザ指定による IP アドレス設定
クレードルに設定された IP アドレスを使用します。
IPMODE_DHCP :DHCP による IP アドレス設定

戻り値

E_OK :正常終了
E_NG :異常終了

説明

LAN クレードルを使用して TCP/IP 通信を行なう場合に使用する IP アドレスは、以下のように決まります。

設定		使用する IP アドレス
lan_setParam_IPmode	dat_system(SYSD_LAN)	
IPMODE_DHCP	LAN_IPADDR_HT	DHCP サーバから取得した IP アドレス
	LAN_IPADDR_CR	
IPMODE_MANUAL	LAN_IPADDR_HT	lan_setParam_xxx にて設定した端末 IP アドレス
	LAN_IPADDR_CR	事前にクレードルに設定しておいた IP アドレス

17.2.2 lan_setParam_IPaddr

IP アドレスを設定します。

```
H lan_setParam_IPaddr(
   long           id,
   unsigned long  IPaddr
);
```

パラメータ

id
IF を指定します。(0 を指定してください)

IPaddr
IP アドレスを指定します。

戻り値

E_OK	:正常終了
E_NG	:異常終了

説明

HT に固定 IP アドレスを設定する場合、本関数を使用したタイミングで新設定が有効になります。

17.2.3 lan_setParam_subnet

サブネットマスクを設定します。

```
H lan_setParam_subnet(
   long           id,
   unsigned long  subnet
);
```

パラメータ

id
IF を指定します。(0 を指定してください)

subnet
サブネットを指定します。

戻り値

E_OK	:正常終了
E_NG	:異常終了

17.2.4 lan_setParam_gateway

デフォルトゲートウェイを設定します。

```
H lan_setParam_gateway(
   long          id,
   unsigned long gateway
);
```

パラメータ

id
IF を指定します。(0 を指定してください)

gateway
デフォルトゲートウェイを指定します。

戻り値

E_OK	:正常終了
E_NG	:異常終了

17.2.5 lan_setParam_DNS

DNS サーバアドレスを設定します。

```
H lan_setParam_DNS(
   long          id,
   unsigned long dns1,
   unsigned long dns2
);
```

パラメータ

id
IF を指定します。(0 を指定してください)

dns1
DNS サーバアドレス(1st)を指定します。

dns2
DNS サーバアドレス(2nd)を指定します。

戻り値

E_OK	:正常終了
E_NG	:異常終了

17.2.6 lancradle_get_IPaddr

LAN クレードルの IP アドレス／サブネットマスク／デフォルトゲートウェイを取得します。

```
ER lancradle_get_IPaddr (
   unsigned long      *IPaddr,
   unsigned long      *subnet,
   unsigned long      *gateway
);
```

パラメータ

IPaddr
LAN クレードルに設定した IP アドレスを格納するポインタを指定します。

subnet
LAN クレードルに設定したサブネットマスクを格納するポインタを指定します。

gateway
LAN クレードルに設定したデフォルトゲートウェイを格納するポインタを指定します。

戻り値

E_OK　　:正常終了
E_NG　　:異常終了

17.2.7 lancradle_set_IPaddr

LAN クレードルの IP アドレス／サブネットマスク／デフォルトゲートウェイを設定します。

```
ER lancradle_get_IPaddr (
   unsigned long      IPaddr,
   unsigned long      subnet,
   unsigned long      gateway
);
```

パラメータ

IPaddr
LAN クレードルに設定する IP アドレスを指定します。

subnet
LAN クレードルに設定するサブネットマスクを指定します。

gateway
LAN クレードルに設定するデフォルトゲートウェイを指定します。

戻り値

E_OK　　:正常終了
E_NG　　:異常終了

17.2.8 net_socket

通信のための端点(ソケット)を作成します。

```
int  net_socket (
  int    addrfamily,
  int    type,
  int    protocol
)
```

パラメータ

addrfamily

アドレスファミリを指定します。

AF_INET　　:IPv4 用ソケット

type

ソケットタイプを指定します。

SOCK_STREAM　　:ストリーム型通信 (TCP 用)

SOCK_DGRAM　　:データグラム型通信 (UDP 用)

protocol

プロトコルを指定します。

0　　:自動判別 (通常はこれを選択してください。)

6　　:TCP

17　　:UDP

戻り値

正常終了した場合は、ソケット番号を返します。

エラーの場合は、-1 を返します。エラーの詳細は、net_getsockerr 関数で取得してください。

説明

TCP/UDP 通信端点となるソケットを作成します。

17.2.9 net_bind

ソケットにローカルアドレスを付与します。

```
int  net_bind (
  int                 socket,
  struct sockaddr     *localaddr,
  int                 addrlen
)
```

パラメータ

socket

ソケット番号を指定します。

localaddr

下記の sockaddr_in 構造体に、自 IP アドレス、自ポート番号をセットし、その構造体のポインタを sockaddr *型にキャストして指定してください。
IP アドレスやポート番号は、ネットワークバイトオーダで指定してください。
IP アドレス、ポート番号に 0 を指定した場合、IP アドレスは自 IP アドレスを、ポート番号は空いている番号を自動割当します。

addrlen

localaddr のサイズを指定します。

戻り値

下記の値を返します。

0	:正常終了
-1	:エラー（詳細は net_getsockerr 関数で取得してください。）

説明

TCP/UDP 通信端点に自アドレスを付与します。

参照

sockaddr_in 構造体、sockaddr 構造体

17.2.10 net_connect

指定された相手先と接続します。

```
int  net_connect (
   int                  socket,
   struct sockaddr      *srvaddr,
   int                  addrlen
)
```

パラメータ

socket

ソケット番号を指定します。

srvaddr

sockaddr_in 構造体に、相手先 IP アドレス、相手先ポート番号をセットし、その構造体のポインタを sockaddr *型にキャストして指定してください。
IP アドレスやポート番号は、ネットワークバイトオーダで指定してください。
sockaddr_in 構造体については、net_bind 関数を参照してください。

addrlen

srvaddr のサイズを指定します。

戻り値

下記の値を返します。

0	:正常終了
-1	:エラー（詳細は net_getsockerr 関数で取得してください。）

説明

引数で指定された相手先と接続します。
UDP ソケットの場合は、以降ここで指定した相手先からのパケットのみ受信するようになります。
net_setsockopt 関数で設定した時間内に接続が完了しなかった場合は、エラーとなります。
net_setsockopt 関数を使用していない場合は、デフォルト値の 1 分となります。

参照

sockaddr_in 構造体、sockaddr 構造体

17.2.11 net_listen

接続を待ちます。

```
int  net_listen (
  int    socket,
  int    backlog
)
```

パラメータ

socket

ソケット番号を指定します。

backlog

接続待ちキューの深さを指定します。
0 から BSDAPI_BACKLOG_MAX までの範囲で指定してください。範囲外の値が指定された場合は、エラーを返さず範囲内の値に補正されます。

ここで設定された数だけ、接続処理が保留されます。例えば 3 に設定した場合、同時に 5 箇所から TCPSYN を受けると以下のように処理されます。

最初の TCPSYN	:接続処理し、後述する net_accept 関数を呼ぶことで接続が確立します。
2-4 番目の 3 つの TCPSYN	:接続処理は行いますが、確立状態に移行せず、接続確立待ち状態として保持されます。
5 番目の TCPSYN	:何も返送しません。その TCPSYN を無視します。

戻り値

正常終了した場合は、ソケット番号を返します。
エラーの場合は、-1 を返します。エラーの詳細は、net_getsockerr 関数で取得してください。

説明

ソケットを TCP 着信待ち状態にします。

17.2.12 net_accept

TCP 接続を受け付けます。

```
int  net_accept (
  int                 socket,
  struct sockaddr     *cliaddr,
  int                 *addrlen
)
```

パラメータ

socket

net_listen 関数で着信待ちにした TCP ソケットのソケット番号を指定します。

cliaddr

TCP 接続元のアドレス情報が必要な場合は、実体を用意した上でこの引数にそのポインタを指定してください。
アドレス情報が不要な場合は 0 を指定してください。

addrlen

cliaddr のサイズを格納した変数のポインタを指定します。
アドレス情報が不要な場合は 0 を指定してください。

戻り値

正常終了した場合は、ソケット番号を返します。
エラーの場合は、-1 を返します。エラーの詳細は、net_getsockerr 関数で取得してください。

説明

net_listen 関数で着信待ちにしたソケットで、TCP 接続を 1 つ受け付けます。
本関数は、デフォルトではタイムアウトしません。net_setsockopt 関数を使用すると、ソケットごとにタイムアウト値を設定することができます。

参照

sockaddr 構造体

17.2.13 net_send

指定済みの相手先にデータを送信します。

```
int  net_send (
  int       socket,
  char      *buf,
  int       buflen,
  int       flags
)
```

パラメータ

socket

ソケット番号を指定します。

buf

送信データ格納領域を指定します。

buflen

送信データの長さを指定します。

flags

UDP ソケットの場合は、0 を指定してください。
TCP ソケットの場合は、下記オプションを指定します。
MSG_OOB　　:TCP 緊急データとして送信します。

戻り値

正常終了した場合は、送信に成功したデータの長さを返します。
エラーの場合は、-1 を返します。エラーの詳細は、net_getsockerr 関数で取得してください。

説明

あらかじめ net_connect 関数で接続した相手先にデータを送信します。
本関数は、ソケット種別により処理が異なります。

- TCP ソケットの場合
  buf で指定されたデータを *buflen* だけ TCP 送信バッファに全てコピーした時点で関数から返ります。
- UDP ソケットの場合
  buf で指定されたデータを 1 つの UDP パケットにコピーした時点で関数から返ります。
  buflen に渡すことができる最大長は 65,000byte までです。ただし、1 つの IP パケットサイズに入りきらない場合は、複数の IP フラグメントパケットを使用して送信されます。
  未 bind 状態でこの関数を呼んだ場合、内部的に自 IP アドレス自動設定、自ポート番号自動割当の設定で bind を行います。

17.2.14 net_sendto

相手先を指定してデータを送信します。

```
int  net_sendto (
  int                 socket,
  char                *buf,
  int                 buflen,
  int                 flags,
  struct sockaddr     *dst,
  int                 dstlen
)
```

パラメータ

socket

ソケット番号を指定します。

buf

送信データ格納領域を指定します。

buflen

送信データの長さを指定します。

flags

UDP ソケットの場合は、0 を指定してください。
TCP ソケットの場合は、下記オプションを指定します。
MSG_OOB　　:TCP 緊急データとして送信します。

dst

UDP ソケットの場合、sockaddr_in 構造体に、相手先 IP アドレス、相手先ポート番号をセットし、その構造体のポインタを sockaddr *型にキャストして指定してください。
IP アドレスやポート番号は、ネットワークバイトオーダで指定してください。
sockaddr_in 構造体については、net_bind 関数を参照してください。
TCP ソケットの場合は、0 を指定してください。

dstlen

UDP ソケットの場合、*dst* のサイズを指定します。
TCP ソケットの場合は、0 を指定してください。

戻り値

正常終了した場合は、送信に成功したデータの長さを返します。
エラーの場合は、-1 を返します。エラーの詳細は、net_getsockerr 関数で取得してください。

説明

引数で指定した相手先に対して、データを送信します。
本関数は、ソケット種別により処理が異なります。

- TCP ソケットの場合
  dst、*dstlen* は無視され、net_connect 関数で接続した相手にデータを送信します。
- UDP ソケットの場合
  net_connect 関数で相手先を指定した場合でも、net_sendto 関数を使用すると相手先に UDP データを送信することができます。

参照

sockaddr_in 構造体、sockaddr 構造体

17.2.15 net_recv

送信元情報を取得せずに、データを受信します。

```
int  net_recv (
  int       socket,
  char      *buf,
  int       buflen,
  int       flags
)
```

パラメータ

socket

ソケット番号を指定します。

buf

受信データ格納領域を指定します。

buflen

最大受信長を指定します。

flags

UDP ソケットの場合は、0 を指定してください。

TCP ソケットの場合は、下記オプションを指定します。

MSG_OOB　　:TCP 緊急データのみを受信します。

戻り値

下記の値を返します。

正の値	:受信したデータ長
0	:相手からの正常切断
-1	:エラー（詳細は net_getsockerr 関数で取得してください。）

説明

net_setsockopt 関数で設定した時間内に送信処理が完了しなかった場合は、エラーとなります。
net_setsockopt 関数を使用していない場合は、デフォルト値の 1 分となります。
本関数は、ソケット種別により処理が異なります。

- TCP ソケットの場合
  受信バッファから *buf* で示される領域へコピーを完了した時点でリターンします。受信バッファに入っているデータが、*buflen* で示される長さより短い場合は、受信バッファが空になるまでデータを取り出し、取り出したデータの長さを返します。また、受信バッファに入っているデータが *buflen* で示される長さ以上の場合は、*buflen* だけ受信して正常終了します。
  受信バッファが空の場合は 1byte でもデータを受信するまで待ち状態になります。
  なお、本 TCP/IP では緊急データの受信時にコールバック関数を呼び出すことができます。これによって、緊急データを受信したときのみ、MSG_OOB をセットした状態で net_recv 関数を呼び出すことができます。

- UDP ソケットの場合
  UDP パケットから *buf* で示される領域へコピーを完了した時点でリターンします。UDP データ長が、*buflen* で示される長さより短い場合は、データを全て取り出し、取り出したデータの長さを返します。UDP データ長が *buflen* で示される長さより長い場合は、*buflen* だけ受信して正常終了します。

17.2.16 net_recvfrom

送信元情報を取得して、データを受信します。

```
int  net_recvfrom (
   int                 socket,
   char                *buf,
   int                 buflen,
   int                 flags,
   struct sockaddr     *from,
   int                 *fromlen
)
```

パラメータ

socket

ソケット番号を指定します。

buf

受信データ格納領域を指定します。

buflen

最大受信長を指定します。

flags

UDP ソケットの場合は、0 を指定してください。
TCP ソケットの場合は、下記オプションを指定します。
MSG_OOB　　:TCP 緊急データのみを受信します。

from

UDP ソケットの場合、sockaddr_in 構造体のポインタを sockaddr*型にキャストして指定してください。
送信元の IP アドレス情報が返ります。
送信元情報が不要な場合は 0 を指定します。
sockaddr_in 構造体については、net_bind 関数を参照してください。
TCP ソケットの場合は、0 を指定してください。

fromlen

UDP ソケットの場合、*from* のサイズを格納した変数のポインタを指定します。
TCP ソケットの場合は、0 を指定してください。

戻り値

下記の値を返します。

正の値	:受信したデータ長
0	:相手からの正常切断
-1	:エラー（詳細は net_getsockerr 関数で取得してください。）

説明

引数で指定した相手に対して、データを受信します。
本関数は、ソケット種別により処理が異なります。詳細は各引数の内容を確認してください。

参照

sockaddr_in 構造体、sockaddr 構造体

17.2.17 net_shutdown

以降の通信を停止します。

```
int  net_shutdown (
  int       socket,
  int       how
)
```

パラメータ

socket

ソケット番号を指定します。

how

停止する通信種別を指定します。

SHUT_RD	:受信を停止する。
SHUT_WR	:送信を停止する。
SHUT_RDWR	:送受信両方を停止する。

戻り値

下記の値を返します。

0	:正常終了
-1	:エラー（詳細は net_getsockerr 関数で取得してください。）

説明

ソケットの通信を停止します。

処理待ちの net_recv 関数、net_recvfrom 関数、net_send 関数、net_sendto 関数からはエラーが返ります。また、本関数を呼んだ以降で net_recv 関数、net_recvfrom 関数、net_send 関数、net_sendto 関数を呼んだ場合もエラーが返ります。

17.2.18 net_close

通信を終了し、ソケットを解放します。

```
int  net_close (
  int    socket
)
```

パラメータ

socket

ソケット番号を指定します。

戻り値

下記の値を返します。

0	:正常終了
-1	:エラー（詳細は net_getsockerr 関数で取得してください。）

説明

ソケットの通信を停止し、ソケットを解放します。
処理待ちの net_recv 関数、net_recvfrom 関数、net_send 関数、net_sendto 関数からはエラーが返ります。
着信待ちキューに入っている TCP 接続は全て破棄されます。
なお、TCP 接続の場合、net_setsockopt 関数で設定した時間まで TCPFIN に対する応答を待ちます。
net_setsockopt 関数を使用していない場合は、デフォルト値の 1 分となります。

17.2.19 net_select

タイムアウト付きでイベントを待ちます。

```
int  net_select (
  int             n,
  fd_set          *readfds,
  fd_set          *writefds,
  fd_set          *exceptfds,
  struct timeval  *timeout
)
```

パラメータ

n

監視するデスクリプタ範囲を指定します。
監視する最大ソケット番号+1 を渡すと効率的です。無駄なループ処理が発生することになりますが、ソケット番号を意識したくない場合は FD_SETSIZE を渡すことも可能です。

readfds

TCP、UDP 受信または TCP 着信をチェックするソケットを指定します。
使用しない場合は 0 を指定してください。

writefds

TCP、UDP 送信可能か否かをチェックするソケットを指定します。
使用しない場合は 0 を指定してください。
なお、UDP ソケットでは必ず送信可能を返します。

exceptfds

使用しません。0 を指定してください。

timeout

タイムアウト値を設定した timeval 構造体を指定します。

戻り値

下記の値を返します。

正の値	:発生イベント数 r
0	:イベントなし。タイムアウト。
-1	:エラー（詳細は net_getsockerr 関数で取得してください。）

説明

受信、送信イベントをタイムアウトつきで待ちます。
この関数を使用することで、複数のソケットの送受信を 1 つのタスクで処理することができます。

fd_set 構造体にアクセスする場合は、下記のマクロを使用してください。

- void FD_ZERO(fd_set **fds*);
  デスクリプタをクリアします。net_select 関数を呼ぶ前に必ずこれでデスクリプタを初期化してください。
- void FD_SET(int *sock*, fd_set **fds*);
  デスクリプタにソケット *sock* を登録します。異なる *sock* で繰り返し呼ぶことで、1 つのデスクリプタに複数のソケットを登録することができます。
- int FD_ISSET(int *sock*, fd_set **fds*);
  デスクリプタに *sock* が登録されているかチェックします。登録されていた場合は 0 以外の値を返します。
- void FD_COPY (fd_set **src_fds*, fd_set **dst_fds*);
  デスクリプタを *src_fds* から *dst_fds* へコピーします。
- void FD_CLR(int *sock*, ctfd_set **fds*);
  デスクリプタに登録されている *sock* を外します。

参照

timeval 構造体、fd_set 構造体

17.2.20 net_getsockopt

ソケットに設定されているオプションを取得します。

```
int  net_getsockopt (
  int       socket,
  int       level,
  int       optname,
  void      *optval,
  int       *optlen
)
```

パラメータ

socket

取得対象のソケット番号を指定します。

level

ソケットレベルを指定します。

SOL_SOCKET　　:ソケットに対するオプションを示します。

optname

取得するオプションの種類を指定します。

SO_TYPE	:ソケットタイプを取得します。 long 型の領域に、SOCK_STREAM か SOCK_DGRAM を返します。
SO_RCVTIMEO	:受信タイムアウト値を取得します。 timeval 構造体で値を返します。タイムアウトなしに設定されている場合は、 *tv_sec*、*tv_usec* ともに 0 が返ります。
SO_SNDTIMEO	:送信タイムアウト値を取得します。 SO_RCVTIMEO と同様に、timeval 構造体で値を返します。
SO_CONTIMEO	:TCP の net_connect 関数のイムアウト値を取得します。 SO_RCVTIMEO と同様に、timeval 構造体で値を返します。
SO_ACCTIMEO	:TCP の net_accept 関数のタイムアウト値を取得します。 SO_RCVTIMEO と同様に、timeval 構造体で値を返します。
SO_CLSTIMEO	:TCP の net_close 関数のタイムアウト値を取得します。 SO_RCVTIMEO と同様に、timeval 構造体で値を返します。
SO_IPID	:このソケットが送受信する IP の I/F ID を取得します。 long 型で値を返します。
SO_URGCB	:net_setsockopt 関数で設定した TCP 緊急データ受信関数ポインタを取得 します。int *型で値を返します。
SO_IPADDR	:IP の I/F ID に対応した自 IP アドレスを返します。 in_addr 構造体で値を返します。

optval

optname で指定したオプションの値を入れる領域です。必要なサイズの領域を用意して、そのポインタを指定してください。

optlen

optval のサイズを格納した変数のポインタを指定します。

戻り値

下記の値を返します。

0	:正常終了
-1	:エラー（詳細は net_getsockerr 関数で取得してください。）

説明

ソケットに設定されている各オプションの値を取得します。

参照

timeval 構造体、in_addr 構造体

17.2.21 net_setsockopt

ソケットにオプションを設定します。

```
int  net_setsockopt (
  int       socket,
  int       level,
  int       optname,
  void      *optval,
  int       optlen
)
```

パラメータ

socket

設定対象のソケット番号を指定します。

level

ソケットレベルを指定します。

SOL_SOCKET　　:ソケットに対するオプションを示します。

optname

設定するオプションの種類を指定します。

SO_RCVTIMEO　　:受信タイムアウト値を設定します。
timeval 構造体で値を渡します。タイムアウトなしに設定する場合は、*tv_sec*、*tv_usec* ともに 0 を指定してください。

SO_SNDTIMEO　　:送信タイムアウト値を設定します。
SO_RCVTIMEO と同様に、timeval 構造体で値を渡します。

SO_CONTIMEO　　:TCP の net_connect 関数のタイムアウト値を設定します。
SO_RCVTIMEO と同様に、timeval 構造体で値を渡します。

SO_ACCTIMEO　　:TCP の net_accept 関数のタイムアウト値を設定します。
SO_RCVTIMEO と同様に、timeval 構造体で値を渡します。

SO_CLSTIMEO　　:TCP の net_close 関数のタイムアウト値を設定します。
SO_RCVTIMEO と同様に、timeval 構造体で値を渡します。

SO_IPID　　:このソケットが送受信する IP の I/F ID を設定します。
long 型で値を渡します。

SO_URGCB　　:TCP 緊急データ受信関数ポインタを int *型で設定します。
TCP 緊急データを受信したときに、コールバック関数として任意の関数を呼び出すことができます。その関数から net_recv 関数で TCP 緊急データを読み出すことができます。コールバック関数の形式については、「17.2.26 コールバック関数」を参照してください。

optval

optname で指定したオプションの値を入れる領域です。必要なサイズの領域を用意して、そのポインタを指定してください。

optlen

optval のサイズを指定します。

戻り値

下記の値を返します。

0	:正常終了
-1	:エラー（詳細は net_getsockerr 関数で取得してください。）

説明

Bluetooth DUN プロファイルを使用したダイアルアップ（PPP）通信を行なう場合には、以下のようにソケットで使用する通信インタフェースを設定してください。設定を行なわないと、デフォルトでは LAN クレードル経由の通信になります。

```
int sock;
long interfaceid;

// ソケット生成
sock = net_socket(AF_INET, SOCK_STREAM, 0);
// インタフェースを PPP に設定
interfaceid = PPP_IF_ID;
net_setsockopt(sock, SOL_SOCKET, SO_IPID, &interfaceid, sizeof(interfaceid) );
```

なお、明示的に LAN クレードル経由の通信を指定する場合には ID として、CRADLE_IF_IP を使用します。

参照

timeval 構造体

17.2.22 net_getsockerr

発生したエラー内容を返します。

```
int  net_getsockerr ()
```

パラメータ

ありません。

戻り値

正常終了した場合は、エラー番号を返します。
エラーの場合は、-1 を返します。

EIO	:プロトコルスタック内部エラー
EBADF	:ソケット番号不正
EFAULT	:引数のポインタが不正
EINVAL	:引数が不正
EMFILE	:ソケット数が最大値を超えた
EPIPE	:通信エラー
EDESTADDRREQ	:相手先アドレス不正
ENOPROTOOPT	:オプション内容不正
EPROTONOSUPPORT	:ソケットタイプとプロトコルが不一致
EOPNOTSUPP	:ソケットに対する操作が不正
EAFNOSUPPORT	:アドレスファミリ不正
EADDRINUSE	:アドレスが既に使われている
EADDRNOTAVAIL	:そのアドレスはその機器で使用できない
ECONNRESET	:相手から RESET を受信した
ENOBUFS	:静的パケットバッファが確保できない
EISCONN	:既に接続している
ENOTCONN	:接続していない
ESHUTDOWN	:シャットダウンされた
ETIMEDOUT	:API がタイムアウトした
ECONNREFUSED	:接続が拒否された

説明

各 API 内でエラーが発生した場合、エラー番号を内部的に記録して-1 を返します。エラーは各 API を呼び出したタスクごとに記録しています。

17.2.23 net_inet_aton

文字列を IPv4 アドレスに変換します。

```
int  net_inet_aton (
  const char        *cp,
  struct in_addr    *inp
)
```

パラメータ

cp

文字列へのポインタを指定します。
ドットノーテーション形式(xxx.xxx.xxx.xxx)で指定してください。

inp

変換された IPv4 アドレスを格納する領域を指定します。
IPv4 アドレスは、ネットワークバイトオーダで記録されます。

戻り値

下記の値を返します。

1	:変換成功
0	:変換失敗

参照

in_addr 構造体

17.2.24 net_inet_addr

文字列を IPv4 アドレスに変換します。

```
in_addr_t  net_inet_addr (
  const char     *cp
)
```

パラメータ

cp

文字列へのポインタを指定します。
ドットノーテーション形式(xxx.xxx.xxx.xxx)で指定してください。

戻り値

下記の値を返します。

0 以外	:IPv4 アドレス(ネットワークバイトオーダ)
0	:変換失敗

説明

cp に「0.0.0.0」を渡した場合、成功しても返り値が 0 になるため、成功か失敗か判断できません。ご注意ください。

17.2.25 net_inet_ntoa

IPv4 アドレスを文字列に変換します。

```
char*  net_inet_ntoa (
  struct in_addr    *in
)
```

パラメータ

in

IPv4 アドレスを設定した in_addr 構造体を指定します。

戻り値

変換された IP アドレスの文字列をドットノーテーション形式(XXX.XXX.XXX.XXX)で返します。

説明

ドットノーテーション形式の IP アドレス文字列に変換して文字列へのポインタを返します。
変換された IP アドレス文字列は「XXX.XXX.XXX.XXX」の形です。区切り記号にはピリオドを使用します。
なお、戻り値として返す文字列領域は全モジュールで共通です。複数のタスクから同時に呼び出すと正しい結果を得られない可能性があります。その場合は呼び出し側で排他制御してください。

参照

in_addr 構造体

17.2.26 コールバック関数

TCP 緊急データ受信時のコールバック関数です。

```
int  callback (
  int    socket,
  int    len
)
```

パラメータ

socket

TCP 緊急データを受信したソケット番号です。

len

受信した TCP 緊急データ長です。

戻り値

変換された IP アドレスの文字列をドットノーテーション形式(XXX.XXX.XXX.XXX)で返します。

説明

本 API では、TCP 緊急データ受信をコールバックで知らせることができます。TCP のコールバック関数を使用する際は、net_setsockopt 関数でソケットにコールバックに使用する関数のアドレスを設定してください。その際、コールバック関数に指定する関数の名称は任意です。

本コールバック関数内で net_recv 関数を呼ぶと、TCP 緊急データを受信することができます。これによって、通常データをタスクで受信している間に緊急データを受信した場合に特別な処理を行うことができます。
なお、本コールバック関数内で net_recv 関数を呼ばなかった場合、受信した緊急データは破棄されます。

17.2.27 net_tcpip_wai_rdy

プロトコルスタックの起動を待ちます。

```
ER  net_tcpip_wai_rdy (
   TMO     tmout,
   long    argnum,
   ...
)
```

パラメータ

tmout

タイムアウト値(tick 単位)、または下記の値を指定します。

TMO_FEVR　　:永久待ち

argnum

起動を待つ I/F の数を指定します。

0 を指定した場合は、「存在する I/F のどれか」という意味になります。

...

起動を待つ I/F ID を並べます。

CRADLE_IF_ID　　:LAN クレードル

PPP_IF_ID　　:Bluetooth ダイアルアップ

戻り値

下記の値を返します。

E_OK　　:正常終了

上記以外　　:異常終了

説明

プロトコルスタックの起動を待ちます。プロトコル起動直後に TCP 接続を開始したり、UDP パケットを送信する場合に使用します。

マルチ IP に対応しており、I/F ごとに起動を待つことが可能です。

例)

・I/F を ID 0 の一つしか使用しない場合

tcpip_wai_rdy(TMO_FEVR, 0);または tcpip_wai_rdy(TMO_FEVR, 1, 0);

⑼ I/F を複数使用する場合、ID 2 の起動を待つ場合

tcpip_wai_rdy(TMO_FEVR, 1, 2);

⑼ I/F を複数使用する場合、ID 0 と ID 3 の両方の起動を待つ場合

tcpip_wai_rdy(TMO_FEVR, 2, 0, 3);

17.2.28 net_ascii_to_ipaddr

IP アドレス文字列を unsigned long 型に変換します。

```
unsigned long  net_ascii_to_ipaddr (
  char   *s
)
```

パラメータ

s

IP アドレスの文字列を指定します。
文字列は、ドットノーテーション形式(XXX.XXX.XXX.XXX)で指定してください。

戻り値

下記の値を返します。

0 以外	:変換された IP アドレス
0	:変換失敗

説明

s で指定された IP アドレス文字列を unsigned long 型の IP アドレスに変換します。
なお、「0.0.0.0」を渡した場合、成功しても返り値が 0 になるため、成功か失敗か判断できません。

17.2.29 net_ipaddr_to_ascii

unsigned long 型の IP アドレスを文字列に変換します。

```
int  net_ipaddr_to_ascii (
  char           *s,
  unsigned long  ipaddr
)
```

パラメータ

s

変換された IP アドレスの文字列を格納する変数のポインタを指定します。
文字列の格納領域は最低 16byte 用意してください。

ipaddr

変換する IP アドレスを指定します。

戻り値

変換された IP アドレスの文字列長を返します。

説明

ipaddr で指定された unsigned long 型の IP アドレスを、ドットノーテーション形式(XXX.XXX.XXX.XXX)の IP アドレス文字列に変換し、*s* で指定された領域に格納します。

17.2.30 net_byte4_to_long

char 型配列の IP アドレスを unsigned long 型の IP アドレスに変換します。

```
unsigned long  net_byte4_to_long (
  const unsigned char    *b
)
```

パラメータ

b

変換する char 型配列の先頭アドレスを指定します。

戻り値

変換された IP アドレスを返します。

説明

*b*で指定された unsigned char 型配列の IP アドレスを、unsigned long 型の IP アドレスに変換します。

17.2.31 net_long_to_byte4

unsigned long 型の IP アドレスを unsigned char 型配列の IP アドレスに変換します。

```
void  net_long_to_byte4 (
  unsigned char  *b,
  unsigned long  d
)
```

パラメータ

b

変換された IP アドレスを格納する char 型配列の先頭アドレスを指定します。
格納領域は最低 4byte 用意してください。

d

変換する IP アドレスを指定します。

説明

*d*で指定された unsigned long 型の IP アドレスを、unsigned char 型配列の IP アドレスに変換します。

17.2.32 net_ping_send

ping を送信します。

```
ER  net_ping_send (
  unsigned long  ipaddr_ul,
  long           ipid,
  void           (*callback)(unsigned long),
  TMO            tmout
)
```

パラメータ

ipaddr_ul

宛先の IP アドレスを、unsigned long 型で指定します。

ipid

パケットを送信する I/F の ID を指定します。

CRADLE_IF_ID	:LAN クレードル
PPP_IF_ID	:Bluetooth ダイアルアップ

callback

応答がきたときに呼ばれる関数のポインタを指定します。

tmout

タイムアウト値(tick 単位)を指定します。

戻り値

下記の値を返します。

E_OK	:応答あり
それ以外	:応答なし

説明

ipaddr_ul で指定した宛先に対して ping を送信します。

応答があった場合、*callback* で指定された関数を呼びます。以下の形式の関数を用意し、そのポインタを *callback* に渡してください。

```
void (*callback)(unsigned long);
```

このときの引数は応答を送信した相手先の IP アドレスです。

なお、本関数は応答のあるなしにかかわらず、*tmout* で指定された時間だけ応答を待ちます。ブロードキャスト宛の ping を送信した場合は、*tmout* で指定された時間まで複数回 *callback* が呼ばれることがあります。

17.2.33 net_get_myip_info

現在起動中の自 IP アドレスに関する情報を取得します。

```
ER  net_get_myip_info (
   long           ipid,
   T_MYIP_PARAM   *param
)
```

パラメータ

ipid

情報を取得する I/F の ID を指定します。

CRADLE_IF_ID　:LAN クレードル
PPP_IF_ID　:Bluetooth ダイアルアップ

param

IP アドレスに関する情報を格納する T_MYIP_PARAM 構造体へ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
それ以外　:パラメータエラー

説明

ipid で指定された I/F が持つ自 IP アドレスに関する情報を取得します。

参照

T_MYIP_PARAM 構造体

17.2.34 net_get_ipaddr

DNS を使用して、ホスト名から IP アドレスを取得します。

```
ER  net_get_ipaddr (
    long            ipid,
    const char      *host,
    T_IPV4EP        *ip
)
```

パラメータ

ipid

情報を取得する I/F の ID を指定します。

CRADLE_IF_ID	:LAN クレードル
PPP_IF_ID	:Bluetooth ダイアルアップ

host

アドレス解決をしたいホスト名を指定します。

ip

解決した IP アドレスの情報を格納する T_IPV4EP 構造体を指定します。

戻り値

下記の値を返します。

E_OK	:正常終了
E_NOEXS	:指定したホストは存在しない
E_OBJ	:サーバからエラーパケットを受信
E_TMOUT	:タイムアウト(サーバが存在しない場合等)
E_NOID	:メモリ不足
その他	:その他のエラー

説明

DNS サーバへアドレス問い合わせパケットを送信し、受信結果を返す関数です。
アドレス解決に成功した場合、*ip->ipaddr* に結果を格納し E_OK を返します。
ホスト名がドット区切りの IPv4 アドレス(例: "192.168.0.1")の場合は、UW 型に変換した値を *ip->ipaddr* に格納します。この場合は DNS サーバへの問い合わせは行いません。

参照

T_IPV4EP 構造体

18.通信ユーティリティ制御

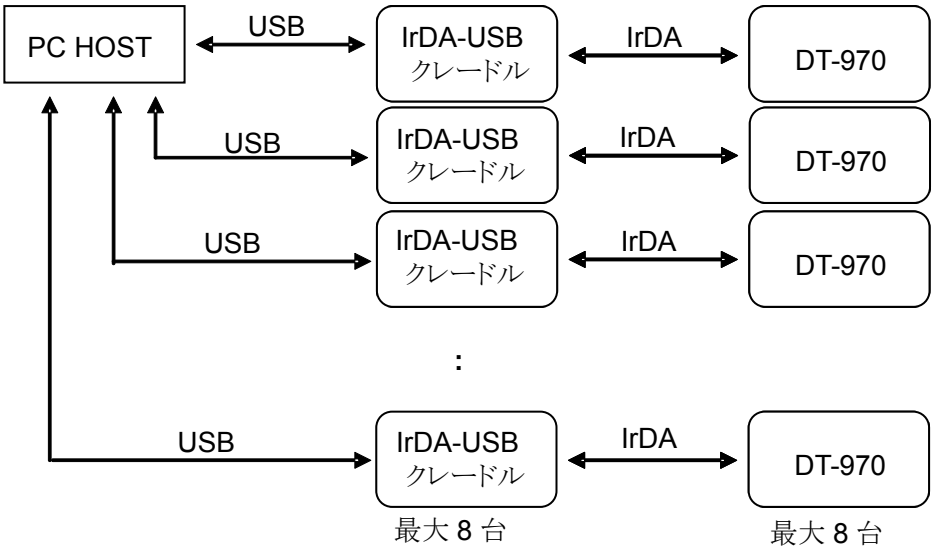
18.1 FLINK プロトコル機能

18.1.1 通信仕様

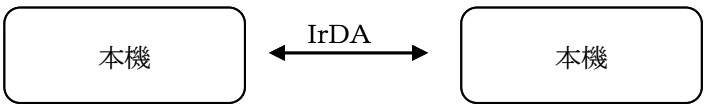
⑼ **通信構成**

本プロトコルのファイル転送機能は次の通りです。

1. IrDA-USB クレードル経由でのホスト通信
   IrDA-USB クレードルによるファイル転送ができます。



2. 赤外線による本体間通信
   赤外線通信による本体間でのファイル転送ができます。



(2) 通信パラメータ

	関数 (基本送受信関数 リモート操作関数)	 システムメニュー用通信機能 (同報通信、本体間通信(子機作成)を含む)
共通パラメータ		
セッション確立待ちタイムアウト時間 (※1)	システム環境設定(データ管理部)より取得します (0 ～ 3600 秒)	
受信待ちタイムアウト時間 (※2)	システム環境設定(データ管理部)より取得します (0 ～ 600 秒)	
セッション終了待ちタイムアウト時間 (※3)	システム環境設定(データ管理部)より取得します (0 ～ 600 秒)	
COM0(赤外線 IrDA)		
通信速度	選択可 (2400～115.2K)	選択可 (システムメニューで選択)

※ 1 セッション確立待ちタイムアウト時間
回線オープン時、セッション確立するまでの待ち時間を監視します。
設定値 0 はタイムアウトなしです。

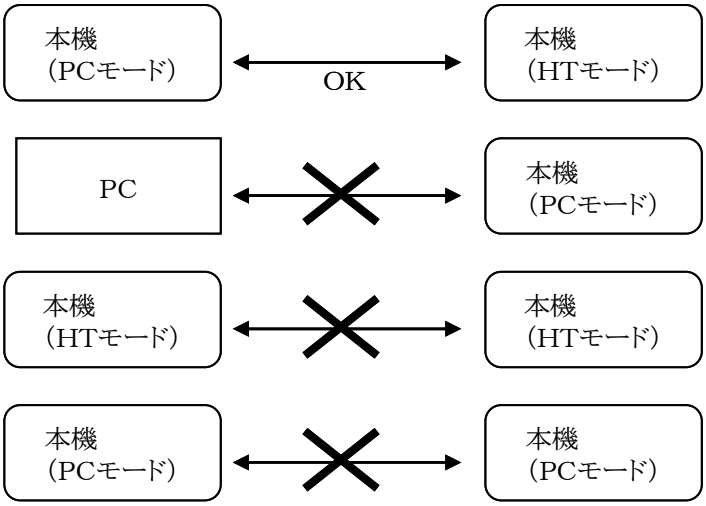
※ 2 受信待ちタイムアウト時間
コマンド／レスポンス受信待ち時間を監視します。
設定値 0 はタイムアウトなしです。

※ 3 セッション終了待ちタイムアウト時間
終了指示コマンド送信側が、相手局のセッション終了を確認するまで待ち時間を監視します。
設定値 0 はタイムアウトなしです。

(3) 動作モード

通信ユーティリティでは接続構成の違いにより次のモードをサポートします。
これらのモードはオープン時に選択します。

1. HT モード
   セッション(※1)確立後、コマンドを送信する権利(以後、送信権とする)を持つモードです。
   PC-DT-970 間通信時および本体-本体間通信時に、どちらか一方の DT-970 が選択されます。
2. PC モード
   セッション確立後、DT-970 からのコマンド待ちとなる、擬似 PC モードです。
   本体-本体間通信時に、どちらか一方の DT-970 が選択されます。



(4) コマンド送信権

1. HT モード
   セッション確立後、HT 側は送信権を有し、PC(PC モードを含みます。以降、PC とします。)に各コマンドを送信することにより、各機能を実現します。
   送信権を PC 側に譲渡する場合は IDLE 通知コマンドを送信します。
   その後、DT-970 は PC からのコマンド受信待ち状態となります。
   IDLE 通知コマンド送信の際、PC のスクリプトファイル実行の指定が可能です。
   ただし、PC モードの DT-970 にはスクリプトファイル実行機能はありません。
2. PC モード
   PC はセッション確立後に DT-970 からのコマンド受信待ちとなり、以後受信したコマンドに従い処理を実行します。
   DT-970 から IDLE 通知コマンドを受信した場合、PC に送信権が移ります。
   尚、PC からの IDLE 遷移は行えません。

※ セッションとは回線オープン時に、相手局確認等のネゴシエーションをさします。
※ 赤外線通信(IrDA)では相手局検出、転送速度の決定等を行います。

(5) 処理概要

以下に各関数内の処理概要およびエラー発生時の処理を示します。
エラー発生時は、直ちに通信を終了します。
この場合、送信権の有無に関らず、先にエラーを検出した側がエラー情報(カテゴリコード・エラー詳細コード)を終了指示コマンドに設定し、相手局へ送信します。
相手局は、受信した終了指示コマンドのエラー情報により、異常終了を検出します。
(エラー処理は、自局内でエラーを検出した場合と、相手局からのエラー終了指示コマンドを受信した場合の両方を指します。)

関数	送信権局の処理	被送信権局の処理	エラー発生時の処理
ファイル送信	コマンド送信後、指定ファイルを順次送信します	指定ファイルを順次受信します	受信中ファイルの削除を行います
ファイル受信	コマンド送信後、指定ファイルを順次受信します	指定ファイルを順次送信します	
ファイル追加	コマンド送信後、指定ファイルを送信します	転送ファイルをテンポラリファイル(FL.ADD)に受信後、ファイルを追加します。追加完了後、テンポラリファイルを削除します	テンポラリファイルを削除します
ファイル削除	コマンドを送信します	指定ファイル／ディレクトリを削除します	削除したファイル／ディレクトリの復旧は行いません
ファイル移動	コマンドを送信します	指定ファイルを移動します	移動後の削除ファイルは復旧しません
ディレクトリ作成	コマンドを送信します	指定ディレクトリを作成します	作成したディレクトリは削除しません
日付時刻の取得	コマンドを送信後、日付時刻情報を受信します	システム日付時刻情報を送信します	―――
日付時刻の設定	コマンドを送信します	日付時刻をシステムに設定します	設定後の日付時刻は復旧しません
ファイル情報の取得	コマンドを送信後、ファイル情報受信します	指定ファイル情報を送信します	―――
ファイル情報の設定	コマンドを送信します	指定ファイルの情報を変更します	設定後のファイル情報は復旧しません
ディスク情報の取得	コマンドを送信後、ディスク情報を受信します	指定ディスク情報を送信します	―――
システム情報の取得	セッション確立時に相手局情報を取得するため、通信は行いません	セッション確立時に相手局情報を取得するため、通信は行いません	―――
画面メッセージ表示	コマンドを送信します	受信データを画面左上より表示します	表示後のメッセージはクリアしません
ブザー鳴動	コマンドを送信します	3 秒間ブザーを鳴らします	―――
IDLE 通知	(HT モードのみ) コマンドを送信後、コマンド受信待ちとなります	(PC モードのみ) コマンド送信権を取得します	―――
終了指示	コマンド送信後、通信を終了します	通信を終了します	―――

18.1.2 ファイル送受信基本機能

複数ファイルの送信および受信を行うための基本機能を提供します。

⑼ 通信基本関数

ファイル送受信関数およびリモート操作関数を使用する際に必要となる基本関数です。

⑼ 通信ポートの初期化

[回線ポートの指定]
回線ポートの初期化を行います。回線ポートは COM0：赤外線(IrDA) とします。
回線ポートによるパラメータおよび設定値を以下に示します。

COM0（赤外線）	
最高速度	CU_B9600(9600bps) / CU_B19K(19.2kbps) / CU_B38K(38.4kbps) / CU_B57K(57.6kbps) / CU_B115K(115.2kbps)

[HT モード／PC モード指定]
PC と通信を行う場合、DT-970 は通常モード（HT モード）でオープンする必要があります。
DT-970 と通信を行う場合、一方が PC モード、もう一方が HT モードでオープンする必要があります。
[APO 禁止設定]
回線オープン時、APO 状態をセーブし、APO 禁止設定を行います。
オープンエラー時は、APO 禁止設定は行われません。
オープン後は、回線クローズでのみ APO の復旧を行うので、エラー発生後は回線をクローズする必要があります。

② 通信ポートのクローズ

通信終了および回線ポートのクローズを行います。通信を終了するために相手局に終了指示コマンドを送信します。
ただし、既に相手局より終了指示コマンドを受信していた場合は、終了指示コマンドの送信は行いません。
終了指示コマンドにはエラー情報（カテゴリコード・エラー詳細コード）を設定します。
回線オープン時にセーブした APO 設定を復旧します。

③ 中断キーの登録／削除

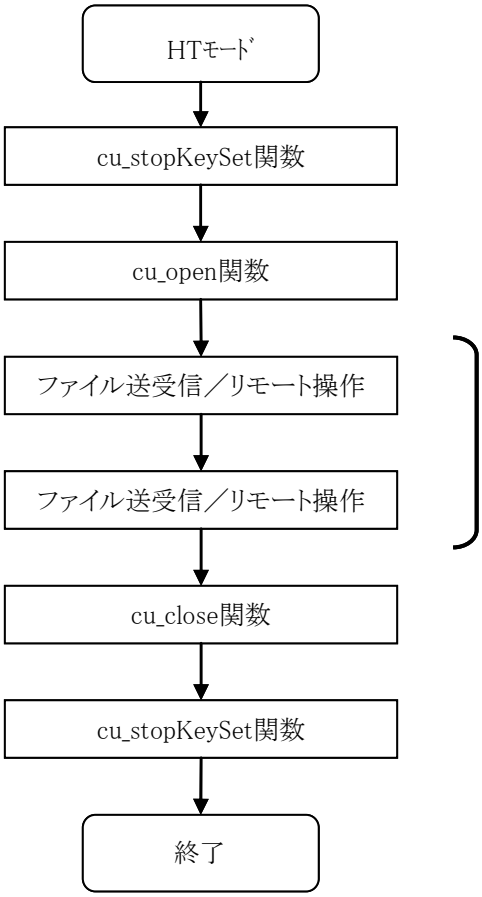
中断キーの選択登録／削除を行います。
登録は回線オープンの前に、削除はクローズの後に行う事が必要です。

④ エラー詳細情報の取得

エラー情報の取得を行います。
発生エラーコード、相手局からの終了指示コマンドのエラー情報（カテゴリコード・エラー詳細コード）等の取得が可能です。

HT コマンド送信による通信

[HT モード基本フロー]



中断キーの登録

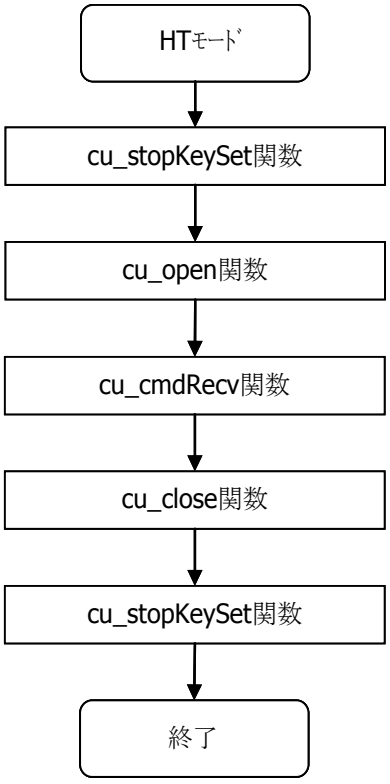
通信ポートのオープン／セッション確立

ファイル送受信／リモート操作関数
を順次実行します。

セッション解放／回線クローズ
（終了指示コマンド送信）

中断キーの削除

[PC モード基本フロー]



中断キーの登録

通信ポートのオープン／セッション確立

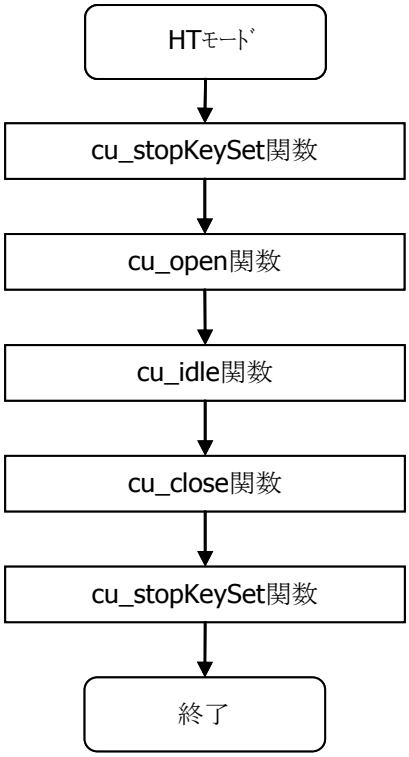
HTからのコマンド待ち
（HTからの終了指示、またはIDLE通知を
受信するまで、受信コマンドを順次実行します。）

セッション確立／回線クローズ

中断キーの削除

PC コマンド送信による通信

[HT モード基本フロー]



中断キーの登録

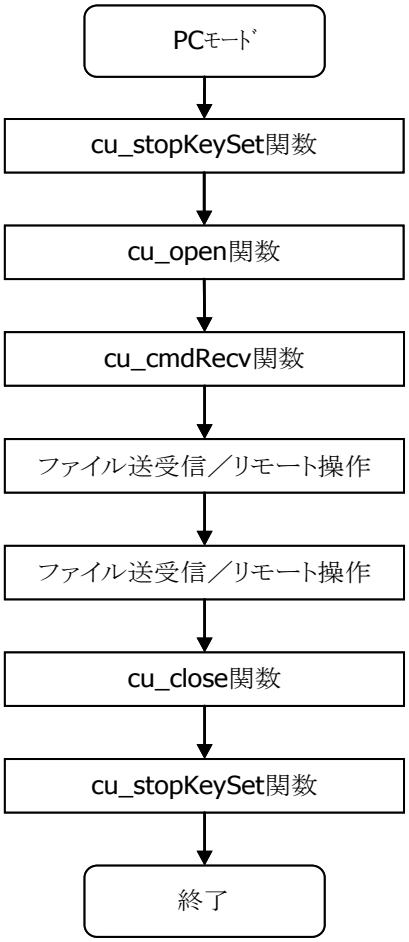
通信ポートのオープン／セッション確立

PCからのコマンド待ち
（PCからの終了指示を受信するまで
受信コマンドを順次実行します。）

セッション解放／回線クローズ

中断キーの削除

[PC モード基本フロー]



中断キーの登録

通信ポートのオープン／セッション確立

HTからのコマンド待ち
（HTからのIDLE通知受信待ち。）

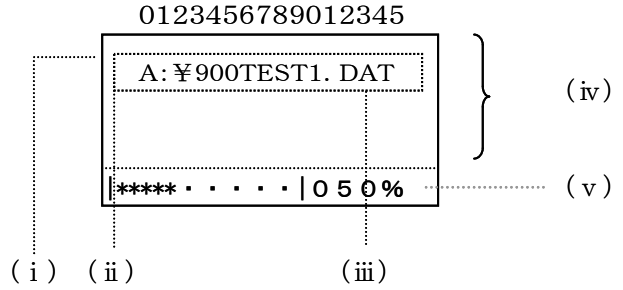
ファイル送受信／リモート操作関数
を順次実行します。

セッション解放／回線クローズ

中断キーの削除

(2) ファイル送受信関数

相手局とのファイル転送(送信、追加、受信)を行うための関数です。
送信権局はファイル送信、追加および受信コマンドを使用して、相手局とのファイル転送を実現します。
被送信権局は、IDLE 状態(HT モード)、PC モードコマンド待ち状態(PC モード)にて、相手からのコマンドを受け付けます。
尚、ファイル転送時、HT の画面上に進捗グラフを表示することが可能です。
グラフ表示フォーマットは次の通りです
(16 ドットフォント 8＊16 での表示例)



(ⅰ) ファイル名表示先頭行(graphPos)
転送ファイル名を表示する行の先頭を指定します。
(ⅱ) ファイル名表示先頭カラム(graphCol)
転送ファイル名を表示する桁の先頭を指定します。
(ⅲ) ファイル名表示フラグ(graphName)
転送ファイル名を全パス表示するか、ファイル名のみ表示するかを指定します。
(ⅳ) ファイル名表示行数(graphLine)
転送ファイルパス名を表示するエリア行数を指定します。
ファイル名表示フラグが、全パス表示の場合のみ有効です。
尚、パス名がファイル名エリア行を越える場合はファイル名のみ表示します。
(ⅴ) 進捗グラフおよびパーセンテージ表示。
ファイル名表示エリアの次の行に表示されます。(1 行固定)
10%単位で"・"が"＊"に変わります。

尚、グラフ表示モード(graphMode)は次の 3 モードをいいます。

- 転送全体を 100%として表示します。
- 1 ファイルを 100%として表示します。
- 表示しません。

⑼ ファイル送信（cu_fileSend）

複数ファイルの送信を一括して行います。
送信先に指定ディレクトリが存在しない場合は、自動的に作成されます。
送信ファイルに対して次のオプションを選択することができます。

(a) リードオンリーファイル強制ライトオプション
　送信ファイルが、既に受信側にリードオンリーファイルとして存在していた場合、強制的にライトすることができます。この指定が無い場合にリードオンリーファイルへのライトを行うと、エラーとなります。
(b) 再帰呼び出し指定オプション
　送信ファイルパス名で指定されたディレクトリ傘下のすべてのファイルが転送対象となります。
　指定ディレクトリ傘下にサブディレクトリが存在した場合はサブディレクトリ名を付加してファイルの送信を行います。

（例）

[送信ファイル名]　[送信先ディレクトリ名]
"A:¥SEND¥AAA.DAT"　　"B:¥RECV¥"

（送信側ディレクトリ構成）

```
A:¥--SEND¥--SUB1¥---AAA.DAT       B:¥---RECV¥---SUB1¥---AAA.DAT
          |-----SUB2¥----BBB.DAT              |----AAA.DAT
          |-----AAA.DAT
          |-----BBB.DAT
```

© ワイルドカードの使用
　送信ファイル名にはワイルドカード(*,?)を使用することができます。

② ファイル追加（cu_fileAdd）

HT 側ファイルを相手局側ファイルにアペンドすることができます。
相手局側に指定されたアペンドファイルが存在しない場合は、新規作成となります。
複数ファイルおよびワイルドカードの指定はできません。

③ ファイル受信（cu_fileRecv）

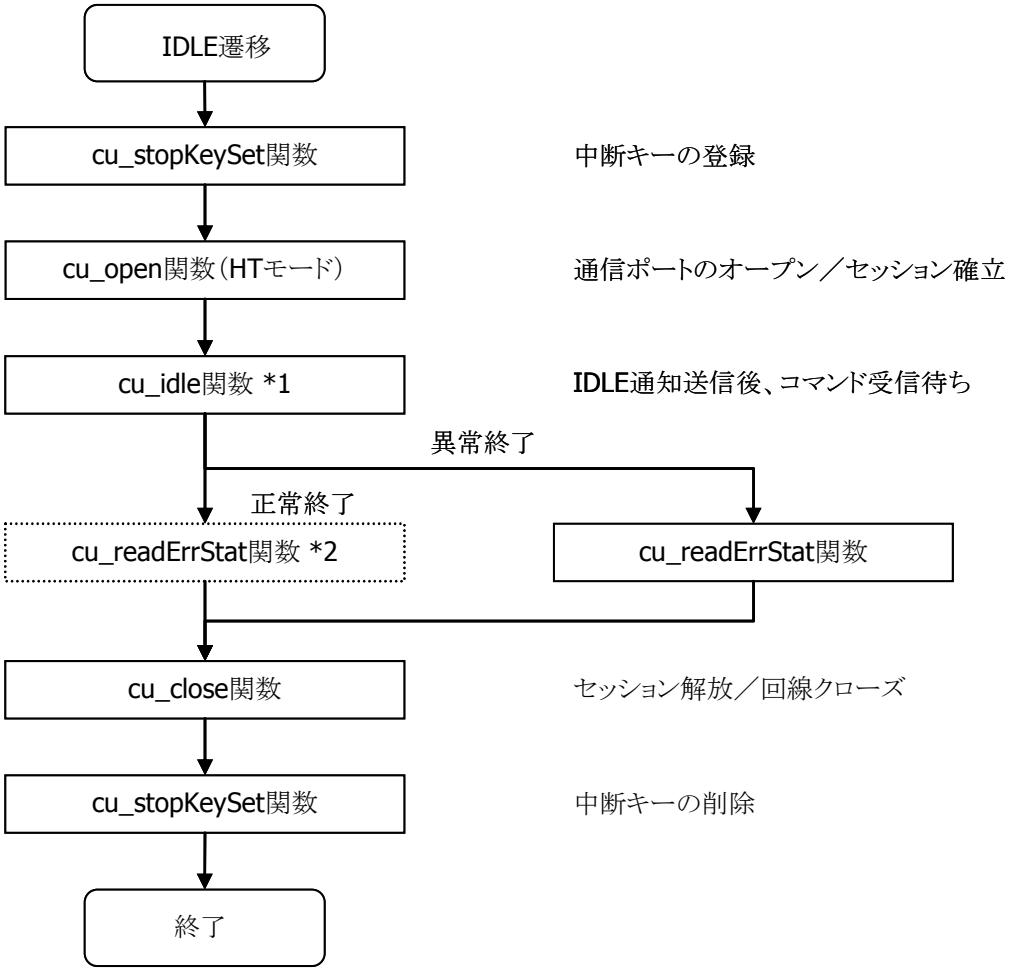
複数ファイルの受信を一括して行うことができます。
受信ファイルに対して次のオプションを選択することができます。

(a) リードオンリーファイル強制ライトオプション
　受信ファイルが、既に受信側にリードオンリーファイルとして存在していた場合、強制的にライトすることができます。この指定が無い場合にリードオンリーファイルへのライトを行うと、エラーとなります。
(b) 再帰呼び出し指定オプション
　受信ファイルパス名で指定されたディレクトリ傘下のすべてのファイルが転送対象となります。
　指定ディレクトリ傘下にサブディレクトリが存在した場合はサブディレクトリ名を付加してファイルの受信を行います。
© ワイルドカードの使用
　受信ファイル名にはワイルドカード(*,?)を使用することができます。

④ IDLE 遷移（cu_idle）

相手局側に送信権を渡し、コマンド待ち状態となります。
終了指示コマンドを受信またはエラー発生まで、受信したコマンドを順次実行します。
尚、オプションとして PC のスクリプトファイルの実行を指示することができます。
エラー発生時は直ちに処理を中止し、エラー処理を行った後、異常終了を返します。

[IDLE 遷移基本フロー]

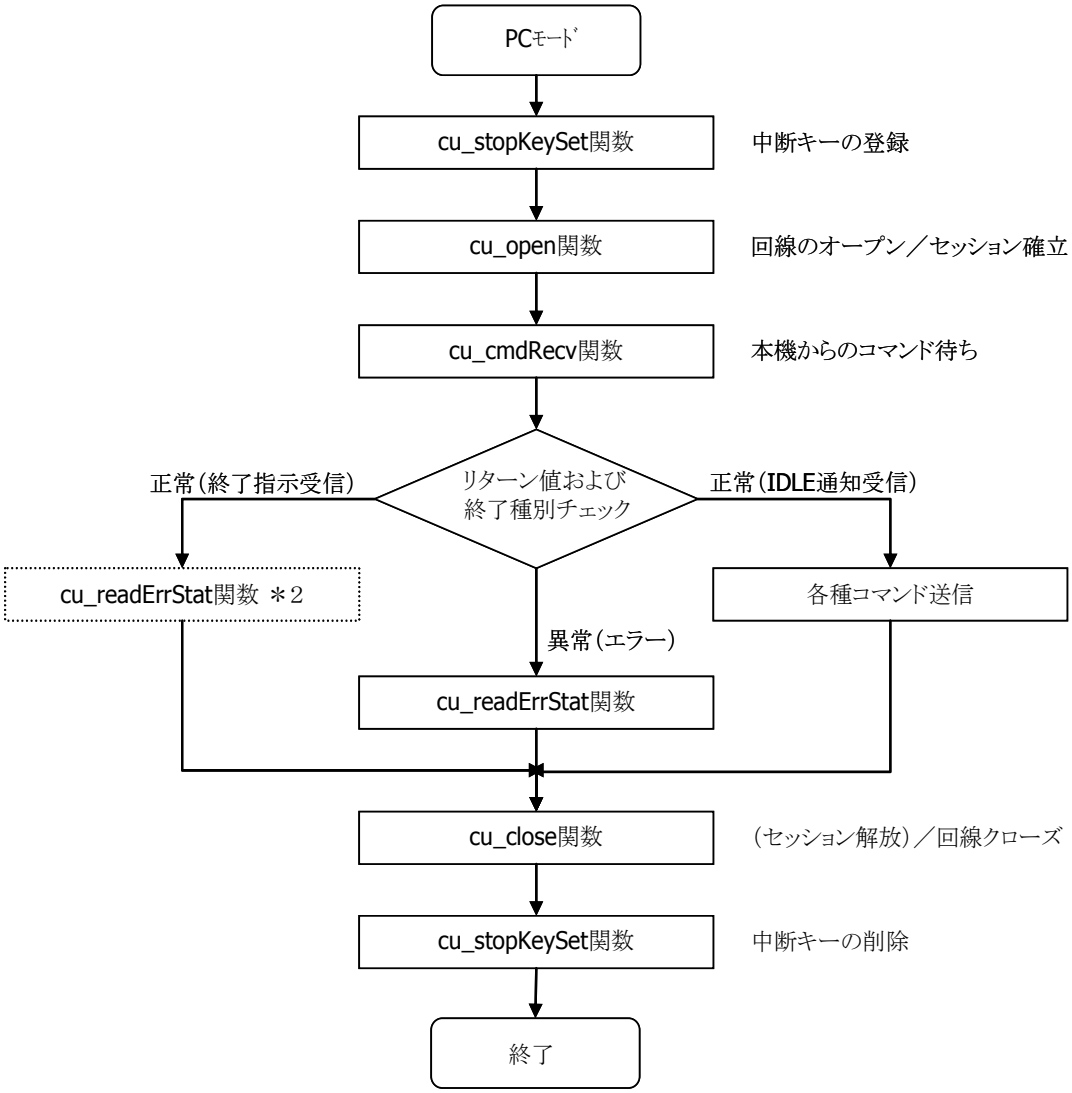


※ 1 相手局からの終了指示コマンド受信またはエラー発生まで、受信コマンドを順次実行します。
※ 2 必要に応じて相手局からの終了指示コマンド詳細情報（フォーマット指示、リセット指示等）の取得が可能です。

⑤ PC モードコマンド待ち（cu_cmdRecv）

DT-970 からのコマンド受信待ち状態となる。PC モードでのみ使用可能です。
セッション確立直後は、HT 側に送信権があるため、PC モードではオープン直後にこの関数を用いてコマンド待ちとなる必要があります。
終了指示コマンドか、IDLE 通知コマンドを受信またはエラー発生まで、受信したコマンドを順次実行します。
エラー発生時は直ちに処理を中止し、エラー処理を行った後、異常終了を返します。

[PC モード時の基本フロー]



※ 1 DT-970 から終了指示コマンドまたは IDLE 通知コマンドを受信するか、エラーが発生するまで、受信コマンドを順次実行します。
※ 2 必要に応じて相手局からの終了指示コマンド詳細情報（フォーマット指示、リセット指示等）の取得が可能です。

18.1.3 リモート操作機能

相手局側のファイル操作、環境情報の取得／設定を行います。
ファイル送受信関数と同様、回線オープン中のみ有効です。

⑼ ファイル操作関数

相手局のファイル／ディレクトリ情報の取得および設定、相手局上でのファイル操作を行うための関数です。

⑼ ファイル／ディレクトリ削除（cu_fileDelete）

相手局側ファイルおよびディレクトリの削除を行います。
複数ファイル指定、ワイルドカード使用が可能です。
指定ファイルが存在しない場合でもエラーになりません。

② ファイル移動（cu_fileMove）

相手局側ファイルの同一ドライブ内での移動またはファイル名の変更を行います。
複数ファイル指定、ワイルドカード使用はできません。
移動先ディレクトリが存在しない場合は自動的に作成します。
移動元と移動先のドライブ名が異なる場合はエラーとなります。

③ ディレクトリ作成（cu_makeDir）

相手局側ディレクトリ作成を行います。
複数ファイル指定、ワイルドカード使用はできません。
タイムスタンプ、属性の設定が可能です。

④ ファイル情報の取得（cu_getFileInfo）

相手局のファイル情報（タイムスタンプ、サイズ、属性）の取得を行います。
ワイルドカード使用が可能です。

⑤ ファイル情報の更新（cu_setFileInfo）

相手局のファイル情報（タイムスタンプ、サイズ、属性）の更新を行います。

(2) 相手局環境情報取得／設定関数

相手局のシステム環境情報の取得および設定を行うための関数です。

(9) 日付時刻の取得／設定 (cu_dateTime)

相手局のシステム日付時刻の取得／設定を行います。

② ディスク情報の取得 (cu_getDiskInfo)

相手局側ディスク情報の取得を行います。
ディスク情報の項目は次の通りです。

- ディスク総容量
- ディスク空き容量
- ディスク状態(フォーマット済み／未フォーマット／ディスクなし)

③ システム情報の取得 (cu_getSysInfo)

相手局側のシステム情報の取得を行います。
システム情報の項目は次の通りです。
セッション ID(通信時のセッション番号)
プロトコルバージョン(ファイル転送プロトコルのバージョン番号)
相手局機種コード(DT-970／PC(AT 互換機)／PC(98 シリーズ))
OS モデル情報(DT-970 モデル種別／PC の OS 種別)
尚、上記情報は回線オープン時のセッション確立直後に相手局より取得します。

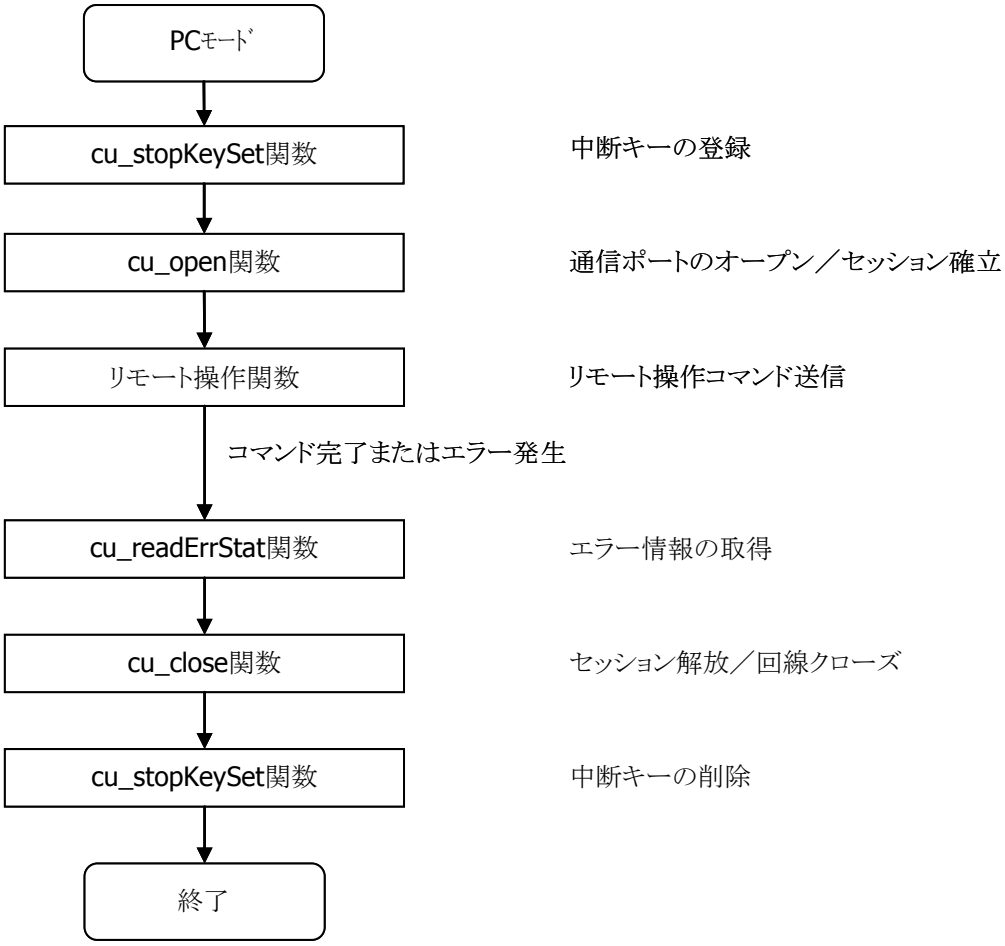
④ 画面表示メッセージの送信 (cu_msgSend)

相手局へ画面表示用のメッセージを送信します。
相手局 HT はメッセージを画面に表示します。

⑤ ブザー鳴動 (cu_beep)

相手局へブザー鳴動コマンドを送信します。
相手局 DT-970 は 3 秒間ブザーを鳴らします。

[リモート操作関数の基本フロー]



18.1.4 ファイルチェック機能

⑼ ファイルチェック概要

ファイルチェック(FCHK)は、インストール直後にインストールが正しく完了したことを確認するための機能です。
ファイル送信局が、FCHK リストを生成し、ファイルと共に受信局側へ送信します。
ファイル受信局側は、FCHK リストのチェックを行う事で、受信したファイルの整合性を確認できます。

(2) FCHK リストファイル生成

転送対象ファイルの FCHK リストファイルを生成します。
FCHK リストファイルには次の項目が設定されます。

- 対象ファイル総数
- 各ファイル名（受信先フルパス名）、サイズ、タイムスタンプ
- 対象ファイルのチェックサム
- FCHK リストファイルのチェックサム

(3) FCHK リストファイルチェック

FCHK リストファイル内の各項目と実ファイルの比較チェックを行います。

18.2 関数リファレンス

ファンクション詳細を次ページより示します。

関数	機能概要
cu_open	回線のオープン(初期化)
cu_fileSend	ファイルの送信
cu_fileAdd	ファイルの追加
cu_fileRecv	ファイルの受信
cu_close	回線のクローズ
cu_readErrStat	エラー情報の取得
cu_idle	IDLE 遷移
cu_cmdRecv	コマンドの受信待ち
cu_fileDelete	ファイルの削除
cu_fileMove	ファイルの移動
cu_makeDir	ディレクトリの作成
cu_getFileInfo	ファイル情報の取得
cu_setFileInfo	ファイル情報の更新
cu_getDiskInfo	ディスク情報の取得
cu_dateTime	日付時刻の取得および設定
cu_getSysInfo	システム情報の取得
cu_msgSend	画面表示メッセージの送信
cu_beep	ブザーの鳴動
cu_setIoboxInfo	クレードル情報の設定
cu_fchklog_Create	FCHK リストファイルの生成
cu_fchklog_Check	FCHK リストファイルのチェック
cu_stopKeySet	中断キーの設定

注意事項

- ファイル名およびディレクトリ名は、特に指定がない場合は絶対パスで指定します。
- ファイル名領域は、終端子に"0x00"を指定します。
- ディレクトリ名領域は、終端子に"¥0x00"を指定します。
- ファイル名を複数指定する時には、連結子として"::"を使用します。
- ファイル名領域およびディレクトリ名領域の最大長は、終端子を含んで、1 ファイルで 256 バイト、複数指定時で 1024 バイトです。

18.2.1 cu_open

【通信ユーティリティ ： FLINK プロトコル】
通信ポートの初期化およびセッションの確立を行ないます。
セッション確立までは、タイムアウト時間まで待ちます。
相手局システム情報の取得を行ないます。

```
ER cu_open(
  H             comNo,
  H             irSpeed,
  CU_RSPRM      *rsPrm,
  H             mode
)
```

パラメータ

comNo
通信ポート

COM0	IrDA
COM9	USB
LAN	LAN クレードル

irSpeed
赤外通信最高速度を指定します。但し、Ir_SetWinMode 関数で PC 接続モードに切り替えてから呼び出された場合、この引数を参照しません。

CU_B9600	9600 bps
CU_B19K	19.2kbps
CU_B38K	38.4 kbps
CU_B57K	57.6 kbps
CU_B115K	115.2 kbps

rsPrm
comNo に COM9 を指定した場合は NULL を指定してください。
COM0 を指定した場合は、CU_RSPRM 構造体に各パラメータを格納して設定してください。
LAN を指定した場合は、CU_LANPRM 構造体に各パラメータを格納し、CU_RSPRM へキャストして設定してください。

mode
局モード

CU_MODE_HT	HT モード
CU_MODE_PC	PC モード（擬似 PC として動作を行います）

戻り値

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

補足

HT 対 HT で通信を行う場合は、一方の HT を HT モード、もう一方の HT を PC モードでオープンする必要があります。

18.2.2 cu_fileSend

【通信ユーティリティ ： FLINK プロトコル】
指定された複数ファイルを一括して送信します。
転送先ディレクトリが存在しない場合は自動的に生成します。
パラメータの指定により、画面に送信処理の進捗を示すグラフを表示できます。

```
ER cu_fileSend(
  H             comNo,
  H             mode,
  B             *fName,
  B             *dir,
  H             protect,
  CU_GRAPHSET   *graphSet
)
```

パラメータ

comNo
通信ポート
COM0　IrDA
COM9　USB
LAN　LAN クレードル

mode
転送モード（通常転送か再帰呼び出し転送かを指定します）
CU_TRANS_NORMAL　通常転送
CU_TRANS_RECURSIVE　再帰呼び出し

fName
送信ファイル名エリア（複数指定およびワイルドカード可）

dir
送信先ディレクトリ名エリア（複数指定およびワイルドカード不可）

protect
強制上書きフラグ（受信側に同一ファイルが書込禁止モードで存在した場合、属性変更して書き込みを行うかを指定します）
CU_PROTECT_VALID　強制書込しません
CU_PROTECT_INVALID　強制書込します

graphSet

グラフ表示情報

```
typedef struct {
  H  graphMode   : グラフ表示モード
                     CU_GRAPH_ON_1          : 転送全体を 100%として表示
                     CU_GRAPH_ON_2          : 1 ファイルを 100%として表示
                     CU_GRAPH_OFF           : 表示しない
                     (CU_GRAPH_OFF 設定時は次のパラメータは参照しません)
  H  graphPos    : ファイル名表示先頭行(0～11)
  H  graphCol    : ファイル名表示先頭桁(0～25)
  H  graphName   : ファイル名表示フラグ(全パス表示かファイル名のみかを指定します)
                     CU_GRAPH_NM_PATH       : 全パス表示
                     CU_GRAPH_NM_FILE       : ファイル名のみ
  H  graphLine   : ファイル名エリア行数(1～12)
} CU_GRAPHSET;
```

戻り値

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

参照

CU_GRAPHSET 構造体

18.2.3 cu_fileAdd

【通信ユーティリティ ： FLINK プロトコル】
指定されたファイルを相手局側の既存ファイルにアペンドします。
送信元、追加先ファイル名とも複数ファイルの指定および、ワイルドカードの指定はできません。
追加先ファイル名が相手局側に存在しない場合、新規にファイルを作成します。
パラメータの指定により、画面に追加処理の進捗を示すグラフを表示できます。

```
ER cu_fileAdd(
  H             comNo,
  B             *sfName,
  B             *rfName,
  CU_GRAPHSET   *graphSet
)
```

パラメータ

comNo
通信ポート

COM0	IrDA
COM9	USB
LAN	LAN クレードル

sfName
送信元ファイル名エリア（複数指定およびワイルドカード不可）

rfName
追加先ファイル名エリア（複数指定およびワイルドカード不可）

graphSet
グラフ表示情報（cu_fileSend 関数参照）

戻り値

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

参照

CU_GRAPHSET 構造体

18.2.4 cu_fileRecv

【通信ユーティリティ ： FLINK プロトコル】
指定された複数ファイルを一括して受信します。
受信先ディレクトリが存在しない場合は自動的に生成します。
パラメータの指定により、画面に受信処理の進捗を示すグラフを表示できます。

```
ER cu_fileRecv(
  H             comNo,
  H             mode,
  B             *fName,
  B             *dir,
  H             protect,
  CU_GRAPHSET   *graphSet
)
```

パラメータ

comNo
通信ポート

COM0	IrDA
COM9	USB
LAN	LAN クレードル

mode
転送モード（通常転送か再帰呼び出し転送かを指定する。）

CU_TRANS_NORMAL	通常転送
CU_TRANS_RECURSIVE	再帰呼び出し

fName
受信ファイル名エリア（複数指定およびワイルドカード可）

dir
受信先ディレクトリ名エリア（複数指定およびワイルドカード不可）

protect
強制上書きフラグ（受信側に同一ファイルが書込禁止モードで存在した場合、属性変更して書き込みを行うかを指定します）

CU_PROTECT_VALID	強制書込しません
CU_PROTECT_INVALID	強制書込します

graphSet
グラフ表示情報（cu_fileSend 関数参照）

戻り値

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

18.2.5 cu_close

【通信ユーティリティ ： FLINK プロトコル】
セッションの開放および回線ポートのクローズを行います。
終了指示コマンドを相手に送信することにより、セッションを開放します。
その際、送信権モード時に限り、相手局に対して終了時の動作指示コマンドを送信することができます。
ただし、既にエラーが発生した場合している場合は送信されません。

```
ER cu_close(
 H   comNo,
 H   endKind
)
```

パラメータ

comNo
通信ポート

COM0	IrDA
COM9	USB
LAN	LAN クレードル

endKind
相手局への終了指示（送信権局モード時のみ有効）

CU_CLOSE_NORMAL	通常終了
CU_CLOSE_RESET	リセット指示
CU_CLOSE_FORMAT_A	Aドライブフォーマット指示
CU_CLOSE_FORMAT_B	Bドライブフォーマット指示
CU_CLOSE_PWROFF	電源 OFF 指示

戻り値

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

18.2.6 cu_readErrStat

【通信ユーティリティ ： FLINK プロトコル】
当ファイル／コマンド送信受信関数でのエラー情報を取得します。
また、相手局からの終了指示コマンド受信時、カテゴリコード・エラー詳細コードを取得します。
取得後、エラー情報はクリアされます。

```
ER cu_readErrStat(
  H           comNo,
  CU_ERRINFO  *errInfo
)
```

パラメータ

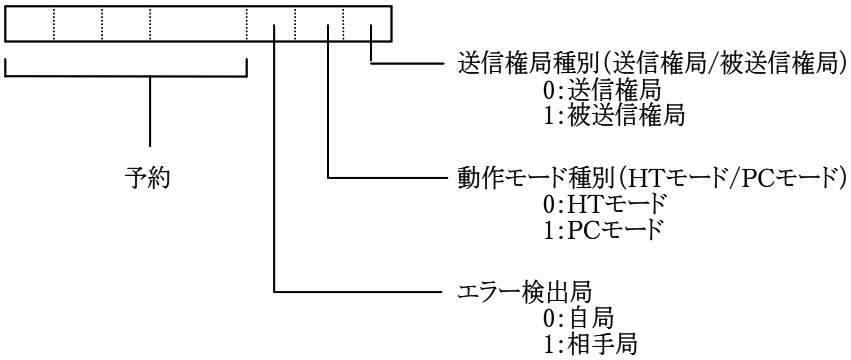
comNo
通信ポート

COM0	IrDA
COM9	USB
LAN	LAN クレードル

errInfo
エラー情報設定エリア

戻り値

```
typedef struct {
    UB  kind        : エラー種別     (下記参照)
    UB  command     : コマンド種別   (次ページ参照)
    UB  category    : カテゴリ       (次ページ参照)
    UB  detail      : エラー詳細     (次ページ参照)
    UW  biosStat    : システム領域エラーエリア
                        (comNo が COM0 時は IrDA 部関数、COM1 時は通信関数のエラー
                        が設定されます)
} CU_ERRINFO;
```



E_OK	正常終了
E_PRM	パラメータエラー

参照

CU_ERRINFO 構造体

解説

エラー情報の取得　コマンド種別・エラー状態　一覧

コマンド種別		
値	シンボル	意味
00	CU_CMD_NON	該当コマンドなし
01	CU_CMD_FSEND_TINFO	ファイル転送情報コマンド
02	CU_CMD_FSEND_FINFO	ファイル情報コマンド
03	CU_CMD_FRECV_TREQ	ファイル受信要求コマンド
04	CU_CMD_FADD	ファイル追加コマンド
05	CU_CMD_FDATA	ファイルデータコマンド
06	CU_CMD_FDEL	ファイル削除コマンド
07	CU_CMD_FMOV	ファイル移動コマンド
08	CU_CMD_MAKEDIR	ディレクトリ作成コマンド
09	CU_CMD_TIME_SET	日付時刻設定コマンド
0A	CU_CMD_TIME_GET	日付時刻取得コマンド
0B	CU_CMD_DISP	メッセージ表示コマンド
0C	CU_CMD_BEEP	ブザー鳴動コマンド
0D	CU_CMD_FINFO_GET	ファイル情報取得コマンド
0E	CU_CMD_FINFO_SET	ファイル情報設定コマンド
0F	CU_CMD_DINFO_GET	ディスク情報取得コマンド
10	CU_CMD_SYS_GET	システム情報取得コマンド
11	CU_CMD_IDLE	IDLE 通知コマンド
12	CU_CMD_END	終了指示コマンド

カテゴリコード・エラー詳細コード
カテゴリとエラー詳細コードの組み合わせによりエラー状態を表す

値		意味
カテゴリ	詳細	
正常終了状態		
00	00	正常終了
DC～F5	00	フォーマット指示コマンド(A～Z)
F6	00	電源 OFF 終了通知
F7	00	リセット指定終了通知
F8	00	中断キーによる終了通知
F9～FF	-	予約領域
プロトコルエラー		
01	00	受信フレームファンクションコード未定義エラー
	01	受信フレームサブファンクションコード未定義エラー
	03	受信フレームチェックサムエラー
	04	シーケンスエラー
	05	シーケンス番号エラー
	07	受信フレーム内情報パラメータエラー
	08	受信タイムアウト
	10	コマンドレングスエラー

ファイルエラー[プロトコル論理]			
04	00	リードオンリィファイルアクセスエラー	
ユーティリティエラー			
10	00	回線オープンエラー	(9) 回線がオープンされていない ・オープン時にエラーが発生していないか確認
	01	使用関数フェーズエラー	(9) 関数の使い方に誤りがある ・動作モード/送信権局モードを確認
	02	使用関数パラメータエラー	(9) 関数パラメータに誤りがある ・指定パラメータを確認
	03	指定ファイル未検出エラー	(9) 指定されたファイルが存在しない ・指定ファイルを確認
	04	相手局未検出	(9) セッション確立待ちタイムアウト ・通信設定、回線経路を確認
	05	システム日付設定エラー	・指定日付を確認
	06	システム時刻設定エラー	・指定時刻を確認
	07	タイマ使用エラー	(9) タイマが登録できなかった ・AP で使用しているタイマ数を確認
	08	CPU クロック切替えエラー	・CPU 切替え禁止状態でないか確認
	09	致命的エラー	(9) IrDA、通信関数からのエラー ・ローバッテリの発生等が考えられる
	0A	通信中回線断エラー	(9) 通信中に回線が切断された ・回線経路を確認
	0B	ドライブ容量不足	・指定ドライブの容量が足りない
ファイルエラー[ファイル関数]			
11	00	クリエートエラー	
	01	オープンエラー	
	02	リードエラー	
	03	ライトエラー	
	04	シークエラー	
	05	ファイル削除エラー	
	06	ディレクトリ削除エラー	
	07	ファイル名変更移動エラー	
	08	タイムスタンプ設定エラー	
	09	タイムスタンプ取得エラー	
	0A	ファイル属性設定エラー	
	0B	ファイル属性取得エラー	
	0C	ディレクトリ作成エラー	
	0D	ファイルサイズ変更エラー	

システムメニュー通信エラー			
20	00	フォーマット実行エラー	(9) フォーマット中にエラー発生 ・再フォーマットする
	01	環境設定ファイル未存在エラー	・CONFIG.HTS ファイルが存在しない
	02	環境設定ファイル更新エラー	(9) CONFIG.HTS 異常 ・ファイルレイアウトを確認
	03	相手局不正	(9) 想定している相手局ではない ・相手局を確認
	04	指定ドライブなし	・子機作成時、送信側指定ドライブが受信側に存在しない
システム異常エラー			
0F	0x	FTP 部内部エラー	
	1x	通信ユーティリティ内部エラー	

18.2.7 cu_idle

【通信ユーティリティ ： FLINK プロトコル】
IDLE 通知送信後、相手局からのコマンド受信待ち状態となります。HT モード時のみ使用可能です。
以後、相手局から受信したコマンドは順次実行していきます。
終了指示コマンドを受信するか、エラーが発生するまで処理を終了しません。
ファイル送信、追加および受信の際、進捗グラフを表示することができます。

```
ER cu_idle(
  H             comNo,
  B             *script,
  CU_GRAPHSET   *graphSet
)
```

パラメータ

comNo
通信ポート

COM0	IrDA
COM9	USB
LAN	LAN クレードル

script
スクリプトファイル名エリア [ファイル名のみ。終端子 0x00 を含め最大 13 バイト]
複数指定およびワイルドカードは不可です。未設定時は NULL を設定します。

graphSet
グラフ表示情報（cu_fileSend 関数参照）
ファイル送信、追加、受信の場合のみ表示します。

戻り値

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

参照

CU_GRAPHSET 構造体

18.2.8 cu_cmdRecv

【通信ユーティリティ ： FLINK プロトコル】
HT からのコマンド受信待ち状態となります。PC モード時のみ使用可能です。
以後、HT から受信したコマンドは順次実行されます。
IDLE 通知コマンド、終了指示コマンドを受信するか、エラーが発生するまで処理を終了しません。
ファイル送信、追加および受信の際、進捗グラフを表示することができます。

```
ER cu_cmdRecv(
  H             comNo,
  H             *endKind,
  B             *script,
  CU_GRAPHSET   *graphSet
)
```

パラメータ

[入力]
comNo
通信ポート

COM0	IrDA
COM9	USB
LAN	LAN クレードル

graphSet
グラフ表示情報(cu_fileSend 関数参照)
ファイル送信、追加、受信の場合のみ表示します。

[出力]
endKind
終了種別フラグ設定エリア(正常終了時のみ有効)

CU_RECV_END	終了指示受信
CU_RECV_IDLE	IDLE 通知受信

script
スクリプトファイル名エリア(IDLE 通知コマンド受信時に設定されます。)
[ファイル名のみ。終端子 0x00 を含め最大 13 バイト]

戻り値

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

参照

CU_GRAPHSET 構造体

18.2.9 cu_fileDelete

【通信ユーティリティ ： FLINK プロトコル】
相手局側のファイル／ディレクトリを削除します。複数ファイル／ディレクトリの削除が可能です。
指定ファイルが存在しない場合は正常終了します。

```
ER cu_fileDelete(
  H    comNo,
  B    *fName
)
```

パラメータ

comNo
通信ポート

COM0	IrDA
COM9	USB
LAN	LAN クレードル

fName
削除するファイル／ディレクトリ名エリア（複数指定およびワイルドカード可）

戻り値

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

18.2.10 cu_fileMove

【通信ユーティリティ ： FLINK プロトコル】
相手局側のファイルを同一ディスク内で移動します。
移動先ディレクトリが存在しない場合は自動生成します。
移動元ディレクトリと移動先ディレクトリが同一でファイル名のみ異なる場合は、ファイル名の変更になります。
移動元と移動先のドライブ名が異なる場合はエラーになります。

```
ER cu_fileMove(
  H   comNo,
  B   *sfName,
  B   *dfName
)
```

パラメータ

comNo
通信ポート

COM0	IrDA
COM9	USB
LAN	LAN クレードル

sfName
移動元ファイル名エリア（複数指定およびワイルドカード不可）

dfName
移動先ファイル名エリア（複数指定およびワイルドカード不可）

戻り値

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

18.2.11 cu_makeDir

【通信ユーティリティ ： FLINK プロトコル】
側のディスクにディレクトリを作成します。

```
ER cu_makeDir(
  H             comNo,
  B             *mDir,
  CU_DATETIME   *datetime,
  B             atr
)
```

パラメータ

comNo
通信ポート

COM0	IrDA
COM9	USB
LAN	LAN クレードル

mDir
作成ディレクトリ名エリア（複数指定およびワイルドカード不可）

datetime
日付時刻エリア

```
typedef struct {
    UB  day;        /* 日(1-31)     */
    UB  month;      /* 月(1-12)     */
    UH  year;       /* 年(1980-2079)    */
    UB  sec;        /* 秒(0-59)     */
    UB  min;        /* 分(0-59)     */
    UB  hour;       /* 時(0-23)     */
} CU_DATETIME;
```

日付時刻を指定しない場合は year に FFFFH を day、month、sec、min、hour に FFH を設定して下さい

atr
属性（OR 指定により複数指定可）

_A_NORMAL	通常ファイル(R/W)
_A_HIDDEN	不可視ファイル
_A_RDONLY	読出し専用ファイル
_A_SYSTEM	システムファイル
_A_SUBDIR	ディレクトリ
_A_ARCH	アーカイブ

(9) _A_SUBDIR は自動的に OR されます。

戻り値

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

参照

CU_DATETIME 構造体

18.2.12 cu_getFileInfo

【通信ユーティリティ ： FLINK プロトコル】
相手局側の指定ファイル情報(ファイルサイズ・タイムスタンプ・属性)の取得を行います。
検索ファイル名と一致するファイルの情報がファイル情報エリアに設定されます。
ワイルドカード指定時は 1 回目に"最初の取得"、2 回目以降に"次情報取得"を指定します。
ワイルドカード指定時は、この関数を連続的に呼ぶ必要があります。
他の通信関数を使用すると、次情報取得は行えません。

```
ER cu_getFileInfo(
  H           comNo,
  H           mode,
  B           *fName,
  CU_FINFO     *fInfo
)
```

パラメータ

comNo
通信ポート

COM0	IrDA
COM9	USB
LAN	LAN クレードル

mode
最初/次フラグ

CU_GET_FIRST	最初の取得(1 ファイル指定またはワイルドカード指定時の 1 回目)
CU_GET_NEXT	次情報取得(ワイルドカード指定時の 2 回目以降)

fName
検索ファイル名エリア
(ワイルドカード指定可。複数指定不可。"次情報取得"では参照しません。)

fInfo
ファイル情報エリア(検索したファイルの情報が設定されます。)
該当ファイルが存在しない場合にはファイル情報エリアの各パラメータに 0x00 が設定されます。

```
typedef struct {
   B            name[256]       : 検索されたファイル名(ﾌﾙﾊﾟｽ名)
   CU_DATETIME  datetime;       : 日付時刻エリア(cu_dateTime 関数参照)
   W            size;           : サイズ
   B            atr;            : 属性(OR 指定により設定される)
                                    _A_NORMAL     : 通常ファイル(R/W)
                                    _A_HIDDEN     : 不可視ファイル
                                    _A_RDONLY     : 読出し専用ファイル
                                    _A_SYSTEM     : システムファイル
                                    _A_SUBDIR     : ディレクトリ
                                    _A_ARCH       : アーカイブ
} CU_FINFO;
```

戻り値

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

参照

CU_FINFO 構造体

18.2.13 cu_setFileInfo

【通信ユーティリティ ： FLINK プロトコル】
相手局側の指定ファイル情報(タイムスタンプ・属性・サイズ)の更新を行います。
ファイル情報エリアの内容をファイル名エリアと一致するファイルに設定します。

```
ER cu_setFileInfo(
  H           comNo,
  CU_FINFO    *fInfo
)
```

パラメータ

comNo
通信ポート

COM0	IrDA
COM9	USB
LAN	LAN クレードル

fInfo
ファイル情報設定エリア

```
typedef struct {
    B             name[256]       : 設定するファイル名(ﾌﾙﾊﾟｽ名)
                                    (複数指定不可・ワイルドカード指定不可)
    CU_DATETIME   datetime;       : 日付時刻エリア(cu_dateTime 関数参照)
                                    (変更しない場合は cu_dateTime 関数と同様)
    UW            size;           : サイズ(0 指定時は変更しません)
    B             atr;            : 属性  (OR 指定により設定)
                                      _A_NORMAL    : 通常ファイル(R/W)
                                      _A_HIDDEN    : 不可視ファイル
                                      _A_RDONLY    : 読出し専用ファイル
                                      _A_SYSTEM    : システムファイル
                                      _A_SUBDIR    : ディレクトリ
                                      _A_ARCH      : アーカイブ
} CU_FINFO;
```

戻り値

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

参照

CU_FINFO 構造体

18.2.14 cu_getDiskInfo

【通信ユーティリティ ： FLINK プロトコル】
相手局側の指定ドライブ情報の取得を行います。
指定ドライブの情報がドライブ情報エリアへ設定されます。

```
ER cu_getDiskInfo(
  H           comNo,
  B           drive,
  CU_DINFO    *dInfo
)
```

パラメータ

[入力]
comNo
通信ポート

COM0	IrDA
COM9	USB
LAN	LAN クレードル

drive
ドライブ名エリア　'A'～'Z'の何れか。

dInfo
ドライブ情報エリアアドレス(検索したドライブの情報が設定されます。)

```
typedef struct {
    UW    size;         /* ディスク容量     */
    UW    freex;        /* ディスク空き容量 */
    UB    status;       /* ディスク状態     */
                          CU_DINFO_NORMAL   ：ディスクあり(フォーマット済)
                          CU_DINFO_NOFMT    ：ディスクあり(未フォーマット)
                          CU_DINFO_NODISK   ：ディスクなし
} CU_DINFO;
```

戻り値

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

参照

CU_DINFO 構造体

18.2.15 cu_dateTime

【通信ユーティリティ ： FLINK プロトコル】
相手局側の日付時刻の取得および設定を行います。
取得の場合は、日付時刻エリアへ相手局のシステム日付時刻が設定されます。
設定の場合は、日付時刻エリアの値を相手局のシステム日付時刻に設定します。

```
ER cu_dateTime(
  H             comNo,
  H             mode,
  CU_DATETIME   *dateTime
)
```

パラメータ

comNo
通信ポート

COM0	IrDA
COM9	USB
LAN	LAN クレードル

mode
取得/設定フラグ

CU_GET_MODE	取得
CU_SET_MODE	設定

dateTime
設定日付時刻エリアアドレス
取得日付時刻エリアアドレス(取得した日付時刻が設定されます)

```
typedef struct {
    UB  day;            /* 日(1～31)          */
    UB  month;          /* 月(1～12)          */
    UH  year;           /* 年(1980～2079)     */
    UB  sec;            /* 秒(0～59)          */
    UB  min;            /* 分(0～59)          */
    UB  hour;           /* 時(0～23)          */
} CU_DATETIME;
```

日付のみの設定の場合は sec,min,hour にすべて FFH を設定して下さい。
時刻のみの設定の場合は day,month,year,それぞれ FFH,FFH,FFFFH を設定して下さい。

戻り値

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

参照

CU_DATETIME 構造体

18.2.16 cu_getSysInfo

【通信ユーティリティ ： FLINK プロトコル】
相手局側のシステム情報を取得します。
相手局が PC の場合は接続セッション番号も返します。(相手局が HT の場合は 0 固定)
尚、これらの情報はオープンのセッション時に既に取得しているため、通信は行わず、情報のみを返します。

```
ER cu_getSysInfo(
  H           comNo,
  CU_SYSINFO  *sysInfo
)
```

パラメータ

comNo
COM NO.

COM0	IrDA
COM9	USB
LAN	LAN クレードル

sysInfo
取得システム情報エリア(検索されたシステム情報が設定されます)

```
typedef struct {
    UH    id;             : セッション ID(PC との接続以外は 0 固定)
    UB    ftpver;         : FTP バージョン
    UB    code[3];        : 機種コード
                            "710"     : HT
                            その他    : PC または他機種
    UB    model;          : モデル情報(00h 固定)
} CU_SYSINFO;
```

戻り値

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

参照

CU_SYSINFO 構造体

18.2.17 cu_msgSend

【通信ユーティリティ ： FLINK プロトコル】
相手局側に表示するメッセージを送信します。

```
ER cu_msgSend(
  H    comNo,
  B    *msg
)
```

パラメータ

comNo
通信ポート

COM0	IrDA
COM9	USB
LAN	LAN クレードル

msg
表示メッセージ格納エリア（終端は NULL を設定）

戻り値

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

18.2.18 cu_beep

【通信ユーティリティ ： FLINK プロトコル】
相手局側のブザーを鳴らします。

```
ER cu_beep(
  H  comNo
)
```

パラメータ

comNo
通信ポート

COM0	IrDA
COM9	USB
LAN	LAN クレードル

戻り値

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

18.2.19 cu_setIoboxInfo

【通信ユーティリティ ： FLINK プロトコル】
LAN クレードルに対し情報設定を行います。
設定情報は、cu_open 関数実行時にクレードルに送信され、クレードルからの応答情報は取得情報アドレスに格納されます。
本関数は、cu_open 関数実行前に実行して下さい。
cu_open 関数実行後は、*setLen*=0 に設定して本関数を実行して下さい。(設定クリア)

```
ER cu_setIoboxInfo(
  H       setLen,
  UB      *setInfo,
  H       *getLen,
  UB      *getInfo
)
```

パラメータ

setLen
設定情報のレングス(0～1024:0 設定時はクレードル通信を行いません)

setInfo
設定情報アドレス

getLen
(設定時)取得情報エリアレングス(0～1024:0 設定時は、取得情報は設定されません)
(cu_open 関数実行時)取得情報レングス

getInfo
取得情報アドレス(cu_open 関数実行時に設定されます)

戻り値

E_OK	正常終了
E_PRM	パラメータエラー

解説

クレードルが TCP/IP 未対応の場合、設定情報は送信されません。
cu_open では、クレードル通信後、ホスト(PC)とのコネクションを実行します。

18.2.20 cu_fchklog_Create

【通信ユーティリティ ： FLINK プロトコル】
指定複数ファイルの FCHK リストファイル(FCHK.LOG)を生成します。
FCHK リストファイルには、指定されたファイルに対する次の情報が生成されます。

(1) ファイルのパス名（転送先ディレクトリ名を含む）
(2) 作成日付
(3) 作成時間
(4) ファイルサイズ
(5) 指定された全ファイルのチェックサムデータ
(6) FCHK リストファイル自身のチェックサムデータ

パラメータの指定により、画面に FCHK リストファイルの生成処理の進捗を示すグラフを表示できます。

```
ER cu_fchklog_Create(
  H             mode,
  B             *fName,
  B             *dir,
  B             *listDir,
  H             append,
  CU_GRAPHSET   *graphSet
)
```

パラメータ

mode
ファイル指定モード（再帰呼び出しを行うかどうかを指定します。）

CU_TRANS_NORMAL	再帰呼出し無
CU_TRANS_RECURSIVE	再帰呼出し有

fName
転送元ファイル名（複数指定およびワイルとカード指定可）

dir
転送先ディレクトリ名（複数指定およびワイルドカード指定不可）

listDir
FCHK リスト生成ディレクトリ名（複数指定およびワイルドカード指定不可）

append
アペンドオプション（既存の FCHK リストファイルに対する追加を指定します。）

CU_FCHK_CREATE	新規作成
CU_FCHK_APPEND	既存ファイルに追加

graphSet

グラフ表示情報

```
typedef struct {
  H  graphMode   : グラフ表示モード
                     CU_GRAPH_ON_1    : リストファイル生成全体を 100%として表示します。
                     CU_GRAPH_OFF     : 表示しません。
  (CU_GRAPH_OFF 設定時は次のパラメータは、参照しません。)
  H  graphPos    : ファイル名表示先頭行  (0～11)
  H  graphCol    : ファイル名表示先頭桁  (0～25)
  H  graphName   : ファイル名表示フラグ
                     (全パス表示かファイル名のみかを指定します。)
                     CU_GRAPH_NM_PATH     : 全パス表示
                     CU_GRAPH_NM_FILE     : ファイル名のみ
  H  graphLine   : ファイル名エリア行数(1～12)
} CU_GRAPHSET;
```

戻り値

E_OK	正常終了
FCHK_NG01	指定したパス名が見つからない
FCHK_NG02	リストファイル作成エラー
FCHK_NG03	FCHK.LOG が見つかりません
FCHK_NG0D	パラメータエラー

参照

CU_GRAPHSET 構造体

18.2.21 cu_fchklog_Check

【通信ユーティリティ ： FLINK プロトコル】
指定されたディレクトリの FCHK リストファイル(FCHK.LOG)の内容と FCHK リストファイル内のファイル情報を比較照合します。
比較照合するファイル情報は、次の情報です。

(1) 作成日付
(2) 作成時間
(3) ファイルサイズ
(4) 全ファイルのチェックサム
(5) FCHK リストファイル自身のチェックサムデータ

パラメータの指定により、画面に FCHK リストファイルの比較処理の進捗を示すグラフを表示できます。

```
ER cu_fchklog_Check(
  B           *listDir,
  CU_GRAPHSET *graphSet
)
```

パラメータ

listDir
FCHK リストファイルが存在するディレクトリ名（複数指定およびワイルドカード指定不可）

graphSet
グラフ表示情報

```
typedef struct {
    H    graphMode   : グラフ表示モード
                         CU_GRAPH_ON_1    : リストファイル照合全体を 100%として表示
                         CU_GRAPH_OFF     : 表示しません
                         （CU_GRAPH_OFF 設定時は次のパラメータは、参照しません）
    H    graphPos    : ファイル名表示先頭行  (0～11)
    H    graphCol    : ファイル名表示先頭桁  (0～25)
    H    graphName   : ファイル名表示フラグ  （全パス表示かファイル名のみかを指定）
                         CU_GRAPH_NM_PATH     : 全パス表示
                         CU_GRAPH_NM_FILE     : ファイル名のみ
    H    graphLine   : ファイル名ｴﾘｱ行数 (1～12)
} CU_GRAPHSET;
```

戻り値

E_OK	正常終了
FCHK_NG03	FCHK.LOG が見つかりません
FCHK_NG04	リストファイルの内容不一致(パス名の不一致)
FCHK_NG05	リストファイルの内容不一致(ファイルサイズの不一致)
FCHK_NG06	リストファイルの内容不一致(日付/時刻の不一致)
FCHK_NG07	リストファイルの内容不一致(全ファイルチェックサムデータの不一致)
FCHK_NG08	リストファイルの内容不一致(リストチェックサムデータの不一致)
FCHK_NG0B	リストファイル読込時エラー
FCHK_NG0D	パラメータエラー

18.2.22 cu_stopKeySet

通信を中断するキーを登録／復旧（戻す）を行います。
設定できるキーは F1～F8 のみであり、COM0,1 で共通設定となります。

```
ER cu_stopKeySet(
  UB   keyId
)
```

パラメータ

keyId
設定する中断キーの指定

CU_FNC_1	:F1
CU_FNC_2	:F2
CU_FNC_3	:F3
CU_FNC_4	:F4
CU_FNC_5	:F5
CU_FNC_6	:F6
CU_FNC_7	:F7
CU_FNC_8	:F8
CU_FNC_NON	:設定なし

戻り値

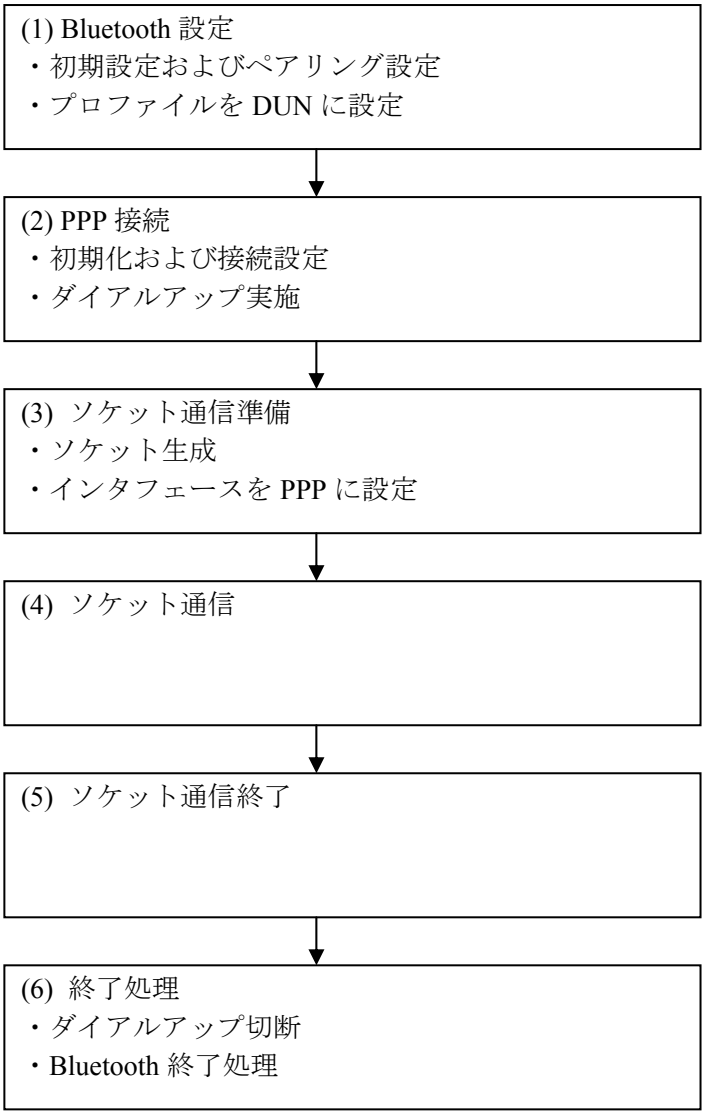
E_OK	:正常終了
E_PRM	:パラメータエラー

19.PPP 制御

19.1 機能

PPP 機能を使用して、Bluetooth DUN プロファイルを使用したダイアルアップ（PPP）通信を行う事が出来ます。

手順は以下の通りです。



⑼ Bluetooth 設定

```
BT_Start();          // BT 電源 ON
BT_SelectProfile(BT_PROFILE_DUN);          // DUN プロファイルを選択
```

電話との Bluetooth 接続設定を実施（BT_SetPassKey()、BT_SelectDev()など）

(2) PPP 接続

```
ppp_init();          // プログラム中1回のみ
//ダイアル時の追加コマンドを携帯電話仕様および APN 仕様によって設定
ppp_setParam_MdmInit1("E0");
ppp_setParam_MdmInit2"+CGDCONT=1,¥"IP¥", ¥"mopera.net¥"");

strcpy(call_prm.number, "*99***1#");
                    // ダイアル時の電話番号。携帯電話仕様および APN 仕様によって設定。
strcpy(call_prm.user, "plus");                    // ダイアルアップ認証用ユーザ名
strcpy(call_prm.password, "softbank");          // ダイアルアップ認証用パスワード

ppp_cctrl_call(PPP_IF_ID, TRUE, ppp_callback1, &call_prm);    // ダイアルアップ実施
```

(3) ソケット通信準備

```
sock = net_socket(AF_INET, SOCK_STREAM, 0);
//インタフェースを PPP に設定
interfaceid = PPP_IF_ID;
net_setsockopt(sock, SOL_SOCKET, SO_IPID, &interfaceid, sizeof(interfaceid) );
```

(4) ソケット通信

net_connect()など

(5) ソケット通信終了

net_close()など

(6) 終了処理

```
ppp_cctrl_disconnect(PPP_IF_ID, TRUE, ppp_callback2);          //ダイアルアップ切断
dly_tsk( 1000 );
BT_Stop();          //BT 電源 OFF
```

19.2 関数リファレンス

ファンクション詳細を次ページより示します。

関数	機能概要
ppp_init	PPP モジュールの初期化
ppp_setParam_MdmInit1	モデム初期化 AT コマンド 1 の設定
ppp_setParam_MdmInit2	モデム初期化 AT コマンド 2 の設定
ppp_cctrl_call	指定した相手先に発信し、PPP 接続
ppp_cctrl_disconnect	通信中の PPP 接続の切断

19.2.1 ppp_init

PPP モジュールの初期化を行います。

```
ER ppp_init()
```

パラメータ

ありません。

戻り値

E_OK	正常終了
E_NG	異常終了

説明

アプリケーション内で 1 回のみコールしてください。

19.2.2 ppp_setParam_MdmInit1

モデム初期化 AT コマンド 1 を設定します。

```
ER ppp_setParam_MdmInit1(
  char    *atcmd
)
```

パラメータ

atcmd

AT コマンド文字列を指定してください。

モデムに対して送信する初期化 AT コマンド文字列を設定します。

少なくともモデムが以下の状態になるように AT コマンドを選択してください。

DTR 信号オフ時の制御	回線切断
応答コード形式	単語方式(英単語応答)
エコーバック	なし
フロー制御方式	ハードウェアフロー制御
DCD 信号の制御	DCD 信号のは回線接続中オン

戻り値

E_OK	正常終了
E_NG	異常終了

19.2.3 ppp_setParam_MdmInit2

モデム初期化 AT コマンド 2 を設定します。

```
ER ppp_setParam_MdmInit2(
  char    *atcmd
)
```

パラメータ

atcmd

ppp_setParam_MdmInit1 関数で指定した AT コマンドを送信し、E_OK が返ると、次にここで指定した AT コマンドが送信されます。

ppp_setParam_MdmInit1 関数で指定した AT コマンドだけで初期化が終わる場合は、本関数を呼び出す必要はありません。

戻り値

E_OK	正常終了
E_NG	異常終了

19.2.4 ppp_cctrl_call

指定されたパラメータで、PPP 接続を行います。

```
ER ppp_cctrl_call(
  long                      ipid,
  int                       sync,
  void                      (*callback)(long, long, long),
  T_CCTRL_SETUP_REQ_PRM     *param
)
```

パラメータ

ipid
IP の IF ID を指定します。(PPP_IF_ID を指定してください)

sync
同期呼び出しの場合は TRUE を指定してください。
それ以外の場合は FALSE を指定してください。

callback
コールバック関数のポインタ

param
発信パラメータへのポインタ

```
typedef struct T_CCTRL_SETUP_REQ_PRM{
   char user[CCTRL_NAME_LEN+1];       // PPP ユーザ ID 文字列
   char password[CCTRL_NAME_LEN+1];   // PPP パスワード文字列
   char number[CCTRL_NAME_LEN+1];     // 電話番号文字列
}
```

戻り値

E_OK	正常終了
E_NG	異常終了

解説

指定した相手先に対して発信し、PPP 接続を行います。
sync と *callback* の組み合わせにより、以下のように動作が異なります。

sync が TRUE で *callback* が NULL のとき
PPP 接続が完了するかエラーが発生するまで本関数から返りません。
PPP 通信中に異常切断が発生しても、ユーザには通知しません。

sync が TRUE で *callback* がコールバック関数のポインタのとき
PPP 接続が完了するかエラーが発生するまで本関数から返りません。
PPP 通信中に異常切断が発生すると、コールバック関数を呼び出します。

sync が FALSE で *callback* が NULL のとき
エラーを返します。

sync が FALSE で *callback* がコールバック関数のポインタのとき
本関数から即座に制御が返ります。
コールバック関数が呼ばれることにより結果が通知されます。
PPP 通信中に異常切断が発生しても、コールバック関数を呼び出します。

19.2.5 ppp_cctrl_disconnect

通信中の PPP 接続を切断します。

```
ER ppp_cctrl_disconnect(
  long     ipid,
  int      sync,
  void     (*callback)(long, long, long)
)
```

パラメータ

ipid
IP の IF ID を指定します。(PPP_IF_ID を指定してください)

sync
同期呼び出しの場合は TRUE を指定してください。
それ以外の場合は FALSE を指定してください。

callback
コールバック関数のポインタ

戻り値

E_OK	正常終了
E_NG	異常終了

解説

通信中の PPP 接続を切断します。
sync と *callback* の組み合わせにより、以下のように動作が異なります。

sync が TRUE で *callback* が NULL のとき
PPP 切断が完了するかエラーが発生するまで本関数から返りません。

sync が TRUE で *callback* がコールバック関数のポインタのとき
PPP 接続が完了するかエラーが発生するまで本関数から返りません。
本関数から返ると同時にコールバック関数を呼び出します。

sync が FALSE で *callback* が NULL のとき
エラーを返します。

sync が FALSE で *callback* がコールバック関数のポインタのとき
本関数から即座に制御が返ります。
コールバック関数が呼ばれることにより結果が通知されます。

20.USB 制御

20.1 関数リファレンス

ファンクション詳細を次ページより示します。

関数	機能概要
USB_setClientMode	USB クライアントの切り替え
USB_getClientMode	USB クライアントの状態取得
USB_setMSCDrive	マスストレージドライブの設定
USB_getMSCDrive	マスストレージドライブの状態取得

20.1.1 USB_setClientMode

USB クライアント接続時の機能を切り替えます。

```
ER USB_setClientMode(
  int     port,
  int     type
);
```

パラメータ

port

接続先のポートを指定します。

USB_PORT_CRADLE	:クレードル接続
USB_PORT_CABLE	:ケーブル接続

type

接続の種類を指定します。

USB_CLIENT_MSC	:マスストレージ(PC からドライブとして見える)
USB_CLIENT_COM	:COM(PC から COM として見える)
USB_CLIENT_HID	:HID 接続

戻り値

E_OK	:正常終了
E_NG	:異常終了
E_PRM	:パラメータエラー

20.1.2 USB_getClientMode

USB クライアントの接続状態を取得します。

```
ER USB_getClientMode(
  int     port,
  int     *type
);
```

パラメータ

port

接続先のポートを指定します。

USB_PORT_CRADLE	:クレードル接続
USB_PORT_CABLE	:ケーブル接続

type

接続の状態を格納する変数のポインタを指定します。

下記の値が返ります。

USB_CLIENT_MSC	:マスストレージ(PC からドライブとして見える)
USB_CLIENT_COM	:COM(PC から COM として見える)
USB_CLIENT_HID	:HID 接続
USB_CLIENT_NON	:未設定
USB_CLIENT_UNKNOWN	:不明

戻り値

E_OK	:正常終了
E_NG	:異常終了
E_PRM	:パラメータエラー

20.1.3 USB_setMSCDrive

指定のドライブをマスストレージ(PC からドライブとして見える)ドライブとして設定します。

```
ER USB_setMSCDrive(
  int     drive
);
```

パラメータ

drive

マスストレージとして設定するドライブを指定します。

USB_MSC_A	:A ドライブ
USB_MSC_B	:B ドライブ
USB_MSC_D	:D ドライブ

戻り値

E_OK	:正常終了
E_NG	:異常終了
E_PRM	:パラメータエラー

20.1.4 USB_getMSCDrive

どのドライブがマスストレージ(PC からドライブとして見える)として設定されているかを取得します。

```
ER USB_getMSCDrive(
  int     *drive
);
```

パラメータ

drive

マスストレージの設定状態を格納する変数のポインタを指定します。

下記の値が返ります。

USB_MSC_A	:A ドライブ
USB_MSC_B	:B ドライブ
USB_MSC_D	:D ドライブ
USB_MSC_NON	:未設定

戻り値

E_OK	正常終了
E_NG	異常終了
E_PRM	:パラメータエラー

21.共通関数

21.1 機能

共通関数は、アプリケーションの実行／終了／各種設定を次の機能によりサポートします。

21.1.1 アプリケーションのロードと実行

DT-970 では ASTART.HTS に登録されたアプリケーションプログラムをロードして実行します。

```
A:¥AP.LOD
```

：Aドライブに置かれたAP.LODを起動するAPSTART.HTSの例

起動されたアプリケーションプログラムは、dat_Apload 関数を使って別のプログラムをロードして実行することができます。但し、呼び出し元のアプリケーションプログラムは dat_Apload 関数がロードするアプリケーションプログラムによって上書きされるため、一般的な関数呼び出しのように制御が呼び出し元に戻ることはありません。

21.1.2 ABORT 処理

本関数が CALL された場合、次の画面を表示し電源キー押下待ちになります。

```
User ABORT

User    :XXXXXXXX
ERR     :XXXXXXXX
KIND    :XXXXXXXX
CODE    :XXXXXXXX
```

ABORT 画面表示中は、次の状態になります。

- すべての通知モードは解除されます。
- 電源キー、RESET スイッチ以外は入力できません。
- 次回電源 ON 時は、レジューム OFF モードになります。
- すべてのファイルをクローズします。
- LCD 以外のすべてのデバイスの電源を OFF にします。
- 本画面表示中は、APO は行いません。

21.1.3 EXIT 処理

次の手順によりユーザアプリケーションを停止させ、システムメニューを起動します。

- アプリケーションの関数結果を共通ワークエリアに待避します。
- すべてのファイルをクローズします。
- ユーザアプリケーションを終了させます。
- すべての通知モードを解除します。
- ファンクションキー等の状態をデフォルト状態に戻します。

21.1.4 動作環境メニュー起動処理

アプリケーションから動作環境メニューを起動し、各種動作設定を行ないます。
動作環境メニューは、終了した段階でアプリケーションへ処理が戻ります。

21.1.5 OBR キャリブレーション起動処理

レーザー発光幅制御に伴う OBR のキャリブレーション処理を起動します。

21.2 関数リファレンス

ファンクション詳細を次ページより示します。

その他共通

関数	機能概要
dat_Apload	アプリケーションのロードと実行
abort	ABORT 処理
exit	EXIT 処理
wkup_cost	動作環境メニューの起動
wkup_calib	OBR キャリブレーションの起動

21.2.1 dat_Apload

指定されたプログラムファイルをアプリケーション領域にロードして実行します。

```
ER dat_Apload(
  B   *path
);
```

パラメータ

path
指定ファイル名の格納先ポインタ(指定方法詳細は open 関数参照)

戻り値

E_OK	正常終了
E_NG	異常終了

21.2.2 abort

次の処理を行ない、アボート画面を表示し電源キー押下待ちになります。(DT-700 インタフェース非互換)
(9) 全ファイルの強制クローズ
②全通知モードの解除
③デバイス電源 OFF(LCD 以外)

```
abort(
  int  user_code
);
```

パラメータ

user_code
表示させたい任意のコード

戻り値

なし

解説

次回電源キーによる立ち上げは、「レジューム OFF」になります。

21.2.3 exit

次の処理を行いユーザアプリケーションを終了し、システムメニューに戻ります。
(9) 関数結果の待避
②全ファイルの強制クローズ
③全通知モードの解除
④ファンクションキー等、システム状態をデフォルトに戻す。

```
exit(
  int  rtn_code
);
```

パラメータ

rtn_code
ユーザアプリケーションの関数結果(固定エリアに保存されます)

戻り値

なし

21.2.4 wkup_cost

アプリケーションを WAIT させ、動作環境メニュータスクを起動します。

```
wkup_cost();
```

パラメータ

なし

戻り値

なし

21.2.5 wkup_calib

レーザー発光幅制御に伴う OBR キャリブレーション処理を起動します。

```
wkup_calib();
```

パラメータ

なし

戻り値

なし

22. 参考資料

22.1 メニュー項目

(9) システムメニュー

項目	設定内容	DT700	DT750	DT800	DT900	DT930	DT970
TOP 項目選択	AP 起動	○	○	○	○	○	○
	動作環境メニュー起動	○	○	○	○	○	○
	日付時刻設定	○	○	○	○	○	○
	転送	○	○	○	○	○	○
	FROM バックアップ		○				
	キャリブレーション起動			○			○
	OS バージョン表示	立上時	立上時	○	○	○	○
	SS 無線ユーティリティ起動			○			
転送	本体受信	○					
	本体送信	○					
	1 ショットインストール	○	○				
	ユーティリティ	○	○	○	○	○	○
	同報インストール		○	○	○	○	
	メモリ転送		○				
	AP インストール			○	○	○	○
	子機作成			○	○	○	○
	通信ポート設定			○	○	○	
	通信速度設定			○	○	○	
	プロトコル				○	○	
メモリ転送	本体受信		○				
	本体送信		○				
	通信ポート設定		○				
	通信速度設定		○				
	チェックサム		○				
子機作成	本体送信			○	○	○	○
	本体受信			○	○	○	○
	転送ドライブ			○	○	○	○
ユーティリティ	AP インストール	○	○				
	ファイル転送	○	○				
	メモリ初期化	○	○				
	通信ポート設定		○				
	通信速度設定		○				
	ファイル送信			○	○	○	○
	ファイル受信			○	○	○	○
	ドライブフォーマット			○	○	○	○
	メモリサイズ変更			○	○	○	○
	ファイルモード				○	○	○

ファイル転送	ホスト受信	○	○				
	ホスト送信	○	○				

(2) 動作環境メニュー

項目	設定内容	DT700	DT750	DT800	DT900	DT930	DT970
TOP 項目選択	環境	○	○	○	○	○	○
	表示モード	○	○	○	○	○	○
	通信セット	○	○				
	バーコード	○	○	○	○	○	○
	ID セット	○	○	○	○	○	○
環境	APO 時間	○	○	○	○	○	○
	ABO 時間	○	○	○	○	○	○
	キークリック ON/OFF	○	○	○	○	○	○
	ブザー音量	○	○	○	○	○	○
	自動コントラスト ON/OFF			○			
	警告メッセージ ON/OFF			○			
表示モード	フォントモード	○	○	○	○	○	○
	メッセージ	○	○	○	○	○	○
	拡大表示						○
	行間モード						○
	タスクバー表示/非表示						○
フォントモード	サイズ(12/16/24)			○			
	サイズ(6/8/10)				○	○	○
	タイプ(標準/強調)			○	○	○	○
通信セット	通信ポート	○	○				
	通信速度	○	○				
	データ長	○	○				
	パリティ	○	○				
	ストップビット	○	○				
	Bluetooth						○
	LAN						○
	USB						○
バーコード	読み取り回数	○	○	○	○	○	○
	照合回数	○	○	○	○	○	○
	タイムアウト	○	○	○	○	○	○
	読み取り禁止時間		○				
	キャリブレーション				○	○	○
電池	設定自動表示						○
	切り替え						○
ID セット	機器 ID	○	○	○	○	○	○
	代理店 ID			○	○	○	○

22.2 関数一覧

(9) 標準ライブラリ

従来機と同等の標準ライブラリをサポートします。

標準ライブラリの提供に必要な低水準レベルの機能をフルサポートします。

(2) 表示部

関数名	機能	DT700	DT750	DT800	DT900	DT930	DT970
lcd_cls	画面のクリア	○	○	○	○	○	○
lcd_csr_set	カーソルタイプの設定	○	○	○	○	○	○
lcd_csr_put	カーソル位置の設定	○	○	○	○	○	○
lcd_csr_get	カーソル位置の読出し	○	○	○	○	○	○
lcd_char	1 文字の表示	○	○	○	○	○	○
lcd_string	文字列の表示	○	○	○	○	○	○
lcd_string2	文字列の表示(スクロール抑制)	○	○	○	○	○	○
lcd_userstr	ユーザー文字列の表示	○	○	○	○	○	○
lcd_line	直線の描画	○	○	○	○	○	○
lcd_gaiji	外字フォント登録	○	○	○	○	○	○
lcd_usrfont	ユーザーフォントファイルの登録	○	○	○	○	○	○
lcd_romfont	ROM フォントの設定	○	○	○	○	○	○
lcd_led	LED の制御	○	○	○	○	○	○
lcd_el	EL バックライトの制御			○	○	○	○
lcd_active_set	ユーザー描画ページの設定			○			
lcd_visual_set	ユーザー表示ページの設定			○			
lcd_active_get	ユーザー描画ページの読み出し			○			
lcd_visual_get	ユーザー表示ページの読み出し			○			
lcd_cls_page	指定ページのクリア			○			
lcd_grchar	グラフィック文字の表示			○			
lcd_box	BOX 描画/削除			○			
lcd_expand_set	拡大表示の設定						○
lcd_expand_get	拡大表示設定状態の取得						○
lcd_expand_pos_set	拡大表示の表示開始位置設定						○
lcd_expand_pos_get	拡大表示の表示開始位置取得						○

(3) キー部

関数名	機能	DT700	DT750	DT800	DT900	DT930	DT970
key_select	キー入力モードの設定		○	○	○	○	○
key_read	1 文字の入力	○	○	○	○	○	○
key_string	文字列の入力	○	○	○	○	○	○
key_num	数値の入力	○	○	○	○	○	○
key_check	キーバッファのステータスチェック	○	○	○	○	○	○
key_clear	キーバッファのクリア	○	○	○	○	○	○
key_fnc	ファンクションキーコードの設定	○	○	○	○	○	○
key_fnc_mode	ファンクションキー通知モード設定	○	○	○	○	○	○
key_patchg	キー入力有効無効設定／解除		○				
key_maketime	切り替えキー確定時間設定		○				
key_pad_set	キーパッドファイル登録			○			
key_touch	ユーザータッチキーの設定削除			○			
key_point	タッチ座標取得			○			
key_pad	キーパッド切り替え／状態取得			○			
key_pad_entry	キーパッド遷移設定／状態取得			○			

(4) 通信部

関数名	機能	DT700	DT750	DT800	DT900	DT930	DT970
c_open	COM のオープン	○	○	○	○	○	※
c_close	COM のクローズ	○	○	○	○	○	※
c_status	COM ステータスのリード	○	○	○	○	○	
c_hold	COM の占有	○	○	○	○	○	
c_chkopen	COM のオープンチェック	○	○	○	○	○	
c_dout	文字列の送信	○	○	○	○	○	※
c_din	1 文字の受信	○	○	○	○	○	
c_tmdin	タイムアウト監視の受信	○	○	○	○	○	※
c_out	1 文字の送信	○	○	○	○	○	
c_break	ブレイク信号の制御	○	○	○	○	○	
c_txrx	送受信の有効/無効	○	○	○	○	○	
c_timer	DR/CS/CD タイムアウト監視値の設定	○		○	○	○	
c_rs	RS 信号の制御	○		○	○	○	
c_er	ER 信号の制御	○		○	○	○	
c_errs	ER/RS 信号の制御	○		○	○	○	
c_iobox	IO ボックス送信の設定	○	○		○	○	
c_irout	IO ボックス送信	○	○		○	○	
c_flush	受信バッファのクリア	○	○	○	○	○	
c_bfsts	受信バッファステータスのリード	○	○	○	○	○	
c_errbfring	リターンコードバッファリング制御の設定	○	○	○	○	○	

c_rderrsts	エラーステータスのリード	○	○	○	○	○	
c_chghdr	受信ハンドラの切り替え	○	○	○	○	○	
c_brkevent	ブレーク要因の設定				○	○	
c_mdout	メモリブロックの送信	○	○				
c_mdin	メモリブロックの受信	○	○				
c_wp	WakeUp 信号の設定				○		
c_cimode	CI 信号立ち上げモード設定				○		

※ USB HID 通信の機能のみを提供します。従来の赤外線通信等のインターフェースはありません。

(5) 電源部

関数名	機能	DT700	DT750	DT800	DT900	DT930	DT970
pwr_inhabit	通知モード設定	○	○	○	○	○	○
pwr_inhabit_clr	電源通知イベントのクリア	○	○	○	○	○	○
pwr_hold_apo	APO 禁止の設定	○	○	○	○	○	○
pwr_off	電源の OFF	○	○	○	○	○	○
pwr_IoboxBootMode	クレードルの起動設定					○	○
pwr_vibrator	バイブレータの動作開始					○	○

(6) イベント通知部

関数名	機能	DT700	DT750	DT800	DT900	DT930	DT970
flg_sts	イベントフラグの取得	○	○	○	○	○	
ref_flg	イベントフラグの取得						○
clr_flg	イベントフラグのクリア	○	○	○	○	○	○
wai_flg	イベント発生待ち	○	○	○	○	○	○

(7) バーコード部

関数名	機能	DT700	DT750	DT800	DT900	DT930	DT970
OBR_open	OBR オープン	○	○	○	○	○	○
OBR_close	OBR クローズ	○	○	○	○	○	○
OBR_getc	OBR データ 1 文字のリード	○	○	○	○	○	○
OBR_gets	OBR データ文字列のリード	○	○	○	○	○	○
OBR_flush	OBR バッファのクリア	○	○	○	○	○	○
OBR_stat	OBR バッファステータスのチェック	○	○	○	○	○	○
OBR_moderd	OBR 動作モードの取得	○	○	○	○	○	○
OBR_modewt	OBR 動作モードの設定	○	○	○	○	○	○
OBR_chgbuf	OBR バッファの切り替え	○	○	○	○	○	○
OBR_trigmode	トリガーキーによる電源 ON 設定		○	○	○	○	○

OBR_swing	レーザー発光幅の設定／参照				○	○	○
OBR_widenarrow	レーザー発光幅の微調整				○	○	※
OBR_getadjust	バー幅補正モードの取得					○	※
OBR_setadjust	バー幅補正モードの設定					○	※
OBR_getmargincheck	マージンチェック倍率モードの取得					○	○
OBR_setmargincheck	マージンチェック倍率モードの設定					○	○
OBR_gain	発光ゲインの切り替え		○	○			
OBR_getfocus	レーザフォーカスモードの取得						○
OBR_setfocus	レーザフォーカスモードの設定						○

※ DT-970 では本関数はサポートしません。

(8) タイマ部

関数名	機能	DT700	DT750	DT800	DT900	DT930	DT970
s_settimer	タイマ 1 登録	○	○	○	○	○	○
s_timerend	タイマ 1 削除	○	○	○	○	○	○
s_settimer2	タイマ 2 登録	○	○	○	○	○	○
s_timerend2	タイマ 2 削除	○	○	○	○	○	○
s_beep	エラービープ音	○	○	○	○	○	○
s_beep2	エラービープ音 2（赤 LED 点灯）		○	○			
s_sound	サウンド音 1	○	○	○	○	○	○
s_dateget	日付の取得	○	○	○	○	○	○
s_dateset	日付の設定	○	○	○	○	○	○
s_timeget	時間の取得	○	○	○	○	○	○
s_timeset	時間の設定	○	○	○	○	○	○

(9) データ管理部

関数名	機能	DT700	DT750	DT800	DT900	DT930	DT970
dat_system	システムデータの設定/取得	○	○	○	○	○	○
dat_OSVer_Read	OS バージョンの取得		○	○	○	○	○
dat_dealer_chk	代理店 ID のチェック			○	○	○	○
dat_mem_size	メモリ領域の空きサイズの取得	○	○	○	○	○	○
dat_get_device_id	シリアル番号の取得						○
dat_Aplоad (※)	アプリケーションのロードと実行			RAM ○	○	○	○

※ A または B ドライブに存在するアプリケーションから他のアプリケーションを実行します。
複数のアプリケーションを同時に動作させる関数ではありません。"RAM ライブラリ"として提供します。

(10) ファイル部

関数名	機能	DT700	DT750	DT800	DT900	DT930	DT970
dat_fsize	ファイル空き領域サイズの取得	○	○		○	○	○
dat_fdir	ファイル格納情報の取得	○	○		○	○	○
dat_fdel	ファイルの削除	○	○		○	○	○
dat_frname	ファイル名の変更					○	○
dat_F_Search	ファイルデータの検索	※	○		○	○	○
dat_fsize_chg	ファイルサイズの変更		○				
open	ファイルのオープン	○	○	○	○	○	○
close	ファイルのクローズ	○	○	○	○	○	○
read	ファイルのリード	○	○	○	○	○	○
write	ファイルのライト	○	○	○	○	○	○
lseek	ファイルリード／ライト位置の設定	○	○	○	○	○	○
sbrk	メモリ領域の割当て	○	○	○	○	○	○
fil_mkdir	ﾃﾞｨﾚｸﾄﾘの作成			○	○	○	○
fil_rmdir	ﾃﾞｨﾚｸﾄﾘの削除			○	○	○	○
fil_remove	ファイルの削除			○	○	○	○
fil_rename	ファイル名の変更／移動			○	○	○	○
fil_fstat	ファイルの日時･サイズ･属性の取得			○	○	○	○
fil_chsize	ファイルのサイズの変更			○	○	○	○
fil_getsize	ファイル領域空きサイズ(4GB 以内)の取得			○	○	○	○
fil_getsize2	ファイル領域空きサイズの取得						○
fil_findfirst	ファイル名の取得			○	○	○	○
fil_findnext	ファイル名の取得(次候補)			○	○	○	○
fil_filesize	ファイルの個数と総サイズ(4GB 以内)の取得				○	○	○
fil_filesize2	ファイルの個数と総サイズの取得						○
fil_filefind	ファイル全パス名の取得				○	○	○

※ dat_sub.obj とリンクすることで対応

(11) 共通関数

関数名	機能	DT700	DT750	DT800	DT900	DT930	DT970
abort	ABORT 処理	○	○	○	○	○	○
exit	EXIT 処理	○	○	○	○	○	○
wkup_cost	動作環境メニューの起動	○	○	○	○	○	○
wkup_calib	OBR キャリブレーションの起動			○	○	○	○
wkup_ss	SS 無線ユーティリティ起動			○			

(12) 通信ユーティリティ

マルチドロップ／FLINK／DT500 プロトコル共通

関数名	機能	DT700	DT750	DT800	DT900	DT930	DT970
cu_stopKeySet	中断キーの設定	○	○	○	○	○	○
cu_setDrive	転送ドライブ指定				○	○	

マルチドロッププロトコル

関数名	機能	DT700	DT750	DT800	DT900	DT930	DT970
cu_open	通信ポート初期化	○	○	○	○	○	
cu_fileSend	ファイルの送信	○	○	○	○	○	
cu_fileSendSet	ファイル送信情報の設定	○	○		○	○	
cu_fileSend1	1 ファイルの送信	○	○		○	○	
cu_fileRecv	ファイルの受信	○	○	○	○	○	
cu_msgSend	画面表示メッセージの送信	○	○	○	○	○	
cu_end	通信の中断	○	○		○	○	
cu_close	回線のクローズ	○	○	○	○	○	
cu_readErrStat	エラー詳細情報の取得	○	○	○	○	○	
cu_readDIRjInfo	データリンク拒否情報の取得	○	○		○	○	

FLINK プロトコル

関数名	機能	DT700	DT750	DT800	DT900	DT930	DT970
cu_open	通信ポート初期化	○	○	○	○	○	○
cu_fileSend	ファイルの送信	○	○	○	○	○	○
cu_fileAdd	ファイルの追加			○	○	○	○
cu_fileRecv	ファイルの受信	○	○	○	○	○	○
cu_close	回線のクローズ	○	○	○	○	○	○
cu_readErrStat	エラー詳細情報の取得	○	○	○	○	○	○
cu_idle	IDLE 遷移			○	○	○	○
cu_cmdRecv	コマンドの受信待ち			○	○	○	○
cu_fileDelete	ファイルの削除			○	○	○	○
cu_fileMove	ファイルの移動			○	○	○	○
cu_makeDir	ディレクトリの作成			○	○	○	○
cu_dateTime	日付時刻の取得および設定			○	○	○	○
cu_getFileInfo	ファイル情報の取得			○	○	○	○
cu_setFileInfo	ファイル情報の更新			○	○	○	○
cu_getDiskInfo	ディスク情報の取得			○	○	○	○
cu_getSysInfo	システム情報の取得			○	○	○	○
cu_msgSend	画面表示メッセージの送信				○	○	○
cu_beep	ブザーの鳴動			○	○	○	○

cu_setIoboxInfo	クレードル情報の設定					○	○
cu_fchklog_Create	FCHK リストファイルの生成			○	○	○	○
cu_fchklog_Check	FCHK リストファイルのチェック			○	○	○	○
cu_apRecvSet	アプリケーションインストールの設定			○			

DT500 プロトコル

関数名	機能	DT700	DT750	DT800	DT900	DT930	DT970
cu_open	通信ポート初期化	○	○	○	○	○	
cu_fileSend	ファイルの送信	○	○	○	○	○	
cu_fileRecv	ファイルの受信	○	○	○	○	○	
cu_close	回線のクローズ	○	○	○	○	○	
cu_readErrStat	エラー詳細情報の取得	○	○	○	○	○	
cu_SetCode	DT500 プロトコル制御コードの拡張設定				○	○	

(13) IrDA 部

関数名	機能	DT700	DT750	DT800	DT900	DT930	DT970
Ir_Open	IrCOMM のオープン			○	○	○	○
Ir_Close	IrCOMM のクローズ			○	○	○	○
Ir_Read	データの読み込み			○	○	○	○
Ir_Write	データの書み込み			○	○	○	○
Ir_QueryTx	送信データ数の問合せ			○	○	○	○
Ir_QueryRx	受信データ数の問合せ			○	○	○	○
Ir_EROn	ER 信号の ON			○	○	○	○
Ir_EROff	ER 信号の OFF			○	○	○	○
Ir_RSOn	RS 信号の ON			○	○	○	○
Ir_RSOff	RS 信号の OFF			○	○	○	○
Ir_BreakOn	ブレイク送信の ON			○	○	○	○
Ir_BreakOff	ブレイク送信の OFF			○	○	○	○
Ir_CheckCD	CD の検査			○	○	○	○
Ir_CheckDR	DR の検査			○	○	○	○
Ir_CheckCS	CS の検査			○	○	○	○
Ir_CheckCI	CI の検査			○	○	○	○
Ir_CheckBreak	BREAK の検査			○	○	○	○
Ir_Err_Get	エラー値の取得			○	○	○	○
Ir_State_Set	通信状態の設定			○	○	○	○
Ir_SetPortConfig	自局能力の設定			○	○	○	○
Ir_Init	IrCOMM の強制終了			○	○	○	○
Ir_SetWinMode	Ir モードの切り替え設定					○	○
Ir_SetParame	パラメータの設定			○			

(14) Bluetooth 部

関数名	機能	DT700	DT750	DT800	DT900	DT930	DT970
BT_Start	Bluetooth 通信の開始					○	○
BT_Stop	Bluetooth 通信の終了					○	○
BT_SelectProfile	使用するプロファイルの指定						○
BT_GetLocalInfo	本体のデバイス情報の取得					○	○
BT_SetLocalInfo	本体のデバイス情報の設定					○	○
BT_Inquiry	Bluetooth 機器の問い合わせ					○	○
BT_GetDevInfo	Bluetooth デバイス情報の取得					○	○
BT_GetDevName	Bluetooth デバイス名の取得					○	○
BT_SelectDev	接続する Bluetooth 機器の指定					○	○
BT_GetPassKey	Bluetooth パスキーの取得						○
BT_SetPassKey	Bluetooth パスキーの設定					○	○
BT_Open	Bluetooth 通信の開始					○	○
BT_Close	Bluetooth 通信の終了					○	○
BT_Read	Bluetooth 通信のデータの受信					○	○
BT_Write	Bluetooth 通信のデータの送信					○	○
BT_QueryRx	読み込み可能なデータ数の取得					○	○
BT_SaveDevInfo	Bluetooth デバイス情報の保存					○	○
BT_LoadDevInfo	Bluetooth デバイス情報の読出し					○	○
BT_Err_Get	エラー値の取得					○	○
BT_HID_Keycode	HID プロファイルのキーコード送信						○

(15) LAN 制御部

関数名	機能	DT930	DT970
lan_setParam_IPmode	DHCP の有効/無効の切り替え		○
lan_setParam_IPaddr	IP アドレスの設定		○
lan_setParam_subnet	サブネットマスクの設定		○
lan_setParam_gateway	デフォルトゲートウェイの設定		○
lan_setParam_DNS	DNS サーバアドレスの設定		○
lancradle_get_IPaddr	LAN クレードルの IP 情報の取得		○
lancradle_set_IPaddr	LAN クレードルの IP 情報設定		○
net_socket	ソケットの作成		○
net_bind	ローカルアドレスの付与		○
net_connect	相手先との接続		○
net_listen	接続の待機		○
net_accept	TCP 接続の受け付け		○
net_send	データの送信		○
net_sendto	データの送信(相手先指定)		○
net_recv	データの受信		○
net_recvfrom	データの受信(送信元情報取得)		○

net_shutdown	通信の停止		○
net_close	ソケットの開放		○
net_select	イベント待ち		○
net_getsockopt	ソケットオプションの取得		○
net_setsockopt	ソケットオプションの設定		○
net_getsockerr	エラー番号の取得		○
net_inet_aton	文字列を IPv4 アドレスに変換		○
net_inet_addr	文字列を IPv4 アドレスに変換		○
net_inet_ntoa	IPv4 アドレスを文字列に変換		○
net_tcpip_wai_rdy	プロトコルスタックの起動待ち		○
net_ascii_to_ipaddr	IP アドレス文字列を unsigned long 型に変換		○
net_ipaddr_to_ascii	unsigned long 型の IP アドレスを文字列に変換		○
net_byte4_to_long	char 型配列の IP アドレスを unsigned long 型の IP アドレスに変換		○
net_long_to_byte4	unsigned long 型の IP アドレスを unsigned char 型配列の IP アドレスに変換		○
net_ping_send	ping の送信		○
net_get_myip_info	自 IP アドレス情報の取得		○
net_get_ipaddr	DNS を使用してホスト名から IP アドレスを取得		○

(16) メモリーバックアップ

関数名	機能	DT700	DT750	DT800	DT900	DT930	DT970
Mbu_format	FROM 領域の確保		○				
Mbu_open	ファイルのオープン		○				
Mbu_close	ファイルのクローズ		○				
Mbu_write	ファイルの書き込み		○				
Mbu_read	ファイルの読み出し		○				
Mbu_clear	FROM クリア		○				
Mbu_sts	メモリーバックアップ状態の取得		○				

(17) プリンタ

関数名	機能	DT700	DT750	DT800	DT900	DT930	DT970
Prn_inf_open	赤外線通信のオープン		○				
Prn_inf_close	赤外ク通信のローズ		○				
Prn_inf_send	赤外線でのコマンド送信		○				
Prn_inf_status	赤外線でのステータスリード		○				
Prn_wir_open	無線のオープン		○				
Prn_wir_close	無線のクローズ		○				
Prn_wir_send	無線のコマンド送信		○				
Prn_pktlen	無線のパケット長登録		○				
Prn_crc_calc	CRC 算出		○				

(18) PPP 制御

関数名	機能	DT700	DT750	DT800	DT900	DT930	DT970
ppp_init	PPP モジュールの初期化						○
ppp_setParam_MdmInit1	モデム初期化 AT コマンド 1 の設定						○
ppp_setParam_MdmInit2	モデム初期化 AT コマンド 2 の設定						○
ppp_cctrl_call	指定した相手先に発信し、PPP 接続						○
ppp_cctrl_disconnect	通信中の PPP 接続の切断						○

(19) USB 制御

関数名	機能	DT700	DT750	DT800	DT900	DT930	DT970
USB_setClientMode	USB クライアントの切り替え						○
USB_getClientMode	USB クライアントの状態取得						○
USB_setMSCDrive	マスストレージドライブの設定						○
USB_getMSCDrive	マスストレージドライブの状態取得						○

22.3 構造体一覧

(1) dat_system で使用する構造体

構造体名	格納情報	DT930	DT970
DAT_PWR_STR	電源関連のシステム情報	○	○
DAT_KEY_STR	キー関連のシステムデータ情報	○	○
DAT_OBR_STR	OBR 関連のシステムデータ情報	○	○
DAT_DSP_STR	表示関連のシステムデータ情報(DT-700 互換)	○	○
DAT_DSP_STR2	表示関連のシステムデータ情報(DT-900 互換)	○	○
DAT_DSP_STR3	画面表示に関する情報		○
DAT_COMINF_STR	通信関連のシステムデータ情報	○	○
DAT_COM_STR	通信関連のシステムデータ情報(IrDA/カシオ IR)	○	○
DAT_TIM_STR	タイマ関連のシステムデータ情報	○	○
DAT_PRO_STR	プロトコル関連のシステムデータ情報	○	○
DAT_SYS_STR	システム関連のシステムデータ情報(DT-700 互換)	○	○
DAT_SYS_STR2	システム関連のシステムデータ情報(DT-900 互換)	○	○
DAT_BAT_STR	電池に関する情報		○
DAT_BTH_STR	Bluetooth に関する情報		○
DAT_LAN_STR	LAN に関する情報		○
DAT_HIP_STR	DT-970 の IP アドレスに関する情報		○
DAT_CIP_STR	LAN クレードルの IP アドレスに関する情報		○
DAT_USB_STR	USB に関する情報		○
DAT_HST_STR	接続先に関する情報		○

(2) ファイル管理関数で使用する構造体

構造体名	格納情報	DT930	DT970
FIL_FSTAT	ファイル属性情報	○	○
FIL_SIZE	ファイルの個数と総サイズ(4GB 以内)	○	○
FIL_SIZE2	ファイルの個数と総サイズ		○
FIND_T	ファイル検索の結果	○	○
DIR_TBL	ファイル格納情報	○	○

(3) キー制御関数で使用する構造体

構造体名	格納情報	DT930	DT970
KEY_INP	1 文字入力情報	○	○
KEY_INPS	文字列入力/数値入力情報	○	○
KEYFORM	キーコードデータ	○	○
KEYSEL	有効無効キーテーブル	○	○

(4) 日時設定関数で使用する構造体

構造体名	格納情報	DT930	DT970
DAY_DAT	日付データ	○	○
TIM_DAT	時刻データ	○	○

(5) OBR 関数で使用する構造体

構造体名	格納情報	DT930	DT970
M_TBL	OBR の動作モード	○	○

(6) シリアル通信関数で使用する構造体

構造体名	格納情報	DT930	DT970
TIM_TBL	シリアル通信制御の監視タイムアウト値	○	○
DEL_TBL	シリアル通信制御のデリートコード情報	○	○
COM_STS	シリアル通信制御における受信バッファのステータス	○	○

(7) IrDA 関数で使用する構造体

構造体名	格納情報	DT930	DT970
State_DCB	IrDA 制御の通信パラメータ	○	○
SetPortConfig_DCB	IrDA 制御における自局能力情報	○	○

(8) Bluetooth 関数で使用する構造体

構造体名	格納情報	DT930	DT970
BT_LOCALINFO	Bluetooth 制御における DT-970 のデバイス情報	○	○
BT_DEVINFO	Bluetooth 機器のデバイス情報	○	○

(9) 通信ユーティリティ制御関数で使用する構造体

構造体名	格納情報	DT930	DT970
CU_RSPRM	FLINK プロトコルにおける通信パラメータ	○	○
CU_LANPRM	LAN 通信におけるパラメータ		○
CU_GRAPHSET	FLINK プロトコルにおける進捗グラフ表示情報	○	○
CU_ERRINFO	FLINK プロトコルにおけるエラー情報	○	○
CU_DATETIME	FLINK プロトコルにおける日付時刻情報	○	○
CU_FINFO	FLINK プロトコルにおけるファイル情報	○	○
CU_DINFO	FLINK プロトコルにおけるディスク情報	○	○
CU_SYSINFO	FLINK プロトコルにおける相手局のシステム情報	○	○

(10) LAN 制御関数で使用する構造体

構造体名	格納情報	DT930	DT970
sockaddr	アドレス情報		○
sockaddr_in	IPv4 アドレス情報		○
in_addr	IPv4 アドレス		○
timeval	時間情報		○
fd_set	ソケット情報		○
T_MYIP_PARAM	IP アドレス情報（net_get_myip_info 関数用）		○
T_IPV4EP	IP アドレス情報（net_get_ipaddr 関数用）		○

(11) その他構造体

構造体名	格納情報	DT930	DT970
T_RFLG	イベントの詳細情報。		○

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/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)