CASIO

CASIO DT-930 C Language SDK
GUI Version

デバイス制御ライブラリ
リファレンスマニュアル

DT-930 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. 1.01

ご注意

- このソフトウェアおよびマニュアルの、一部または全部を無断で使用、複製することはできません。
- このソフトウェアおよびマニュアルは、本製品の使用許諾契約書のもとでのみ使用することができます。
- このソフトウェアおよびマニュアルを運用した結果の影響については、一切の責任を負いかねますのでご了承ください。
- このソフトウェアの仕様、およびマニュアルに記載されている事柄は、将来予告なしに変更することがあります。
- このマニュアルの著作権はカシオ計算機株式会社に帰属します。
- 本書中に含まれている画面表示は、実際の画面とは若干異なる場合があります。予めご了承ください。

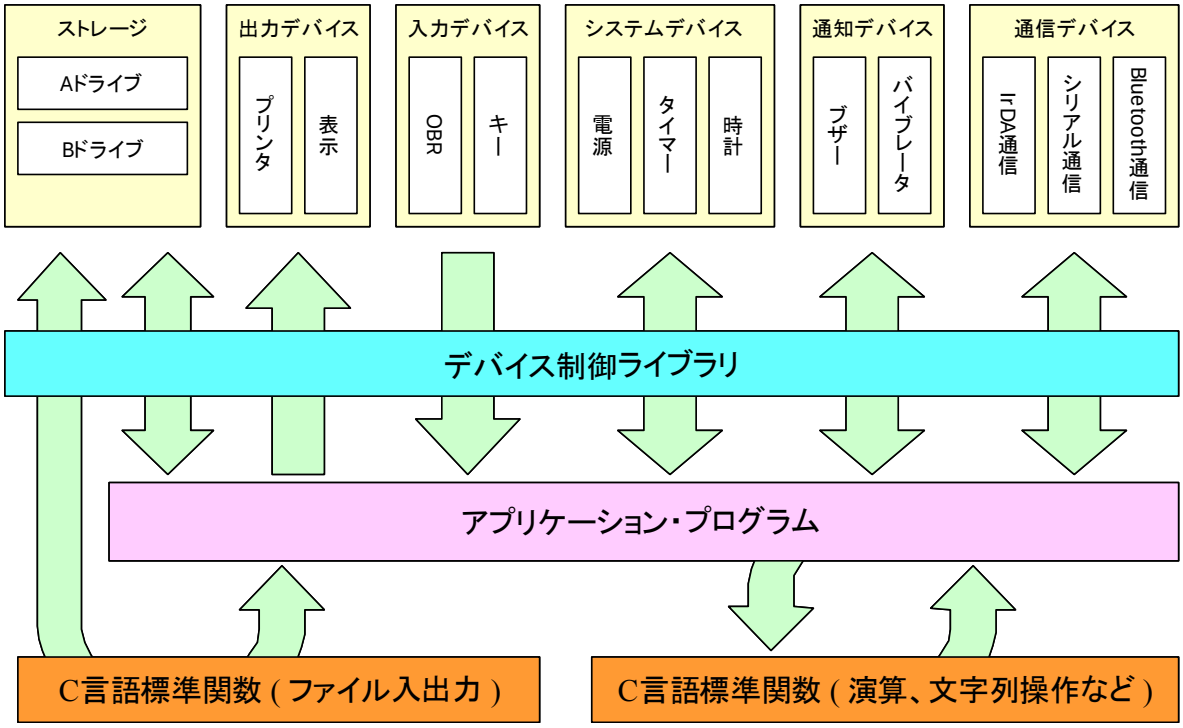
変更履歴

バージョン	変更日付	ページ	内容
0.90	2007/01/22		新規作成
1.00	2007/02/28	4	enum の説明を追加
↑	↑	204	レーザー発光幅の説明を追加
↑	↑	379	FLINK の説明を修正
1.01	2008/12/09	32	M_TBL 構造体を追加

目次

1. はじめに

このマニュアルは DT-930 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。



C 言語の標準関数は、ファイル入出力、およびデバイスに依存しない関数(演算、文字列操作など)を使うことができます。
DT-930 で使用可能な C 言語標準関数は「3. C 言語標準関数(p.58)」を参照してください。

1.1 構成ファイルについて

DT-930 C 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
CU_MULTI.H	通信プロトコル(マルチドロップ用)構造体と関数の定義
CU_DT500.H	通信プロトコル(DT500 用)構造体と関数の定義
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

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_STARA.OBJ	アプリケーション初期化モジュールオブジェクト(DT-700 互換表示 A 用)
AP_STARB.OBJ	アプリケーション初期化モジュールオブジェクト(DT-700 互換表示 B 用)
APINIT.OBJ	アプリケーション使用変数初期化モジュール

ライブラリ

HICIF.LIB	DT-930 関数ライブラリ
SHCLIB.LIB	C 言語標準ライブラリ

1.1.1　基本的な使い方

DT-930 の関数ライブラリを使用するアプリケーションプログラムは、ソースコードに”BIOS1MAC.H”をインクルードし、ライブラリとして”HICIF.LIB”をリンクする必要があります。

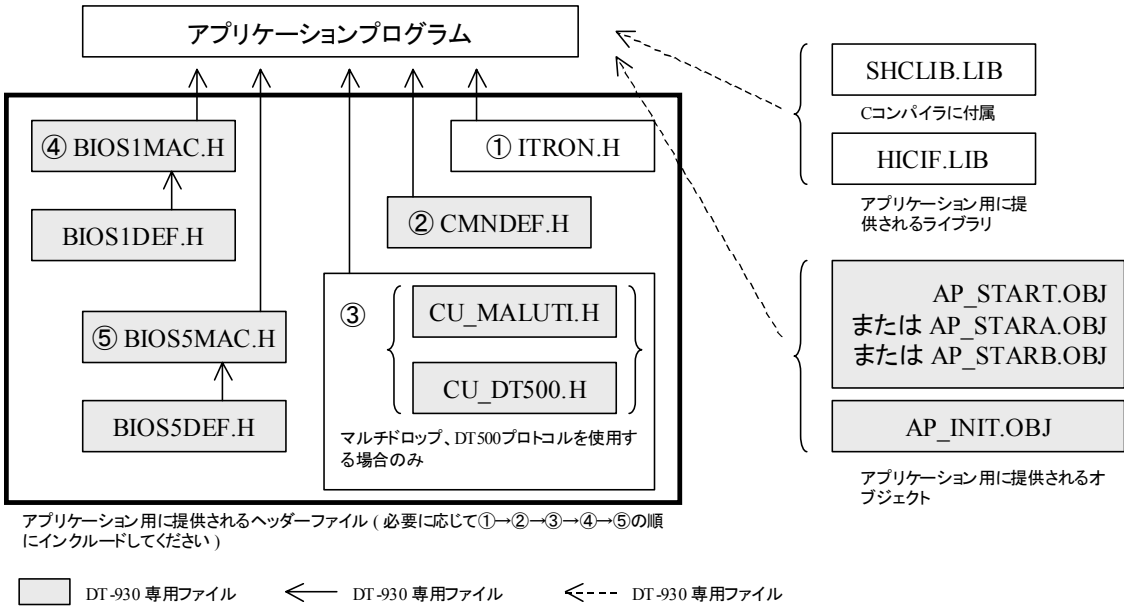
また、C 言語の標準ライブラリ関数を使用する場合には、”SHCLIB.LIB”をリンクしてください。

マルチドロッププロトコルまたは DT500 プロトコルを使用する場合には、専用のヘッダファイルをインクルードしなければなりません。(FLINK プロトコルは標準でサポートしていますので、ヘッダファイルをインクルードする必要はありません)
また、ヘッダファイルは、BIOS1MAC.H より前にインクルードする必要があります。

各プロトコルは一つのソースファイルで同時に使用することはできません。
複数のプロトコルを使用するときは、それぞれオブジェクトに分けてください。

※ 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)

ファイルの関係



2. データ型

DT-930 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

型	サイズ[*1]	境界[*1]	符号	最小値	最大値
char (signed char)	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[*2]	4	4	有	-2^{31}(-2147483648)	2^{31}-1(2147483647)
float	4	4	有	-∞	∞
double long double	8[*3]	8	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

※2 enum 型は int として扱います

※3 -double=float オプションを指定している場合、double 方のサイズは 4 バイトになります

DT-930 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	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)	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	H
ER	エラーコード	W
FN	機能コード	W

2.2 構造体

DT-930 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情報を格納します。

```
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

KEY 関連のシステムデータ情報を格納します。

```
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情報を格納します。

```
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情報を格納します。(DT-700 互換)

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

メンバ

clk_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情報を格納します。(DT-900 互換)

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

メンバ

font_kd
フォント種類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_COMINF_STR 構造体

通信関連のシステムデータ情報を格納します。

```
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_MULTI	:マルチドロップ
PRT_FLINK	:FLINK
PRT_DT500	:DT500

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

2.2.7 DAT_COM_STR 構造体

通信関連のシステムデータ情報を格納します。(IrDA/カシオ IR インタフェース)

```
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

2.2.8 DAT_TIM_STR 構造体

タイマ関連のシステムデータ情報を格納します。

```
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

2.2.9 DAT_SYS_STR 構造体

システム関連のシステムデータ情報を格納します。(DT-700 互換)

```
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

2.2.10 DAT_SYS_STR2 構造体

システム関連のシステムデータ情報を格納します。(DT-900 互換)

```
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

2.2.11 DAT_PRO_STR 構造体

プロトコル関連のシステムデータ情報を格納します。

```
typedef struct sys_pro{
   /* マルチドロップ  */
   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           */
   W      sirial_no;
   W      level_parity;
   W      bht_tmout;
} DAT_PRO_STR;
```

メンバ

non_rec_tmout
通常受信タイムアウトを 0～99 秒の範囲で格納します。

non_retry_ct
通常リトライ回数を 0～99 回の範囲で格納します。

mal_rec_tmout
マルチドロップ受信タイムアウトを 0～9990 秒の範囲で格納します。

ptp_snd_tmout, ptp_rec_tmout, ptp_rec_retry_ct
予約領域。使用しません。

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。

sirial_no

シリアル NO を次の値で格納します。

0　:OFF
1　:ON

level_parity

水平パリティを次の値で格納します。

0　:OFF
1　:ON

bht_tmout

DT500 プロトコルリンク確立タイムアウトを 0～240 秒の範囲で格納します。

参照

dat_system 関数

2.2.12 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
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(0～59)	2 秒単位

attr
ファイルの属性を次の値で格納します。

_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

2.2.13 FIL_SIZE 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_size{
   UW     cnt;
   UW     size;
} FIL_SIZE;
```

メンバ

cnt
ファイルの個数を格納します。

date
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

2.2.14 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
ファイルの属性を格納します。値の詳細は、FIL_FSTAT 構造体を参照してください。

wr_time
ファイルの最終更新時刻を格納します。時刻のフォーマットは、FIL_FSTAT 構造体を参照してください。

wr_date
ファイルの最終更新日付を格納します。日付のフォーマットは、FIL_FSTAT 構造体を参照してください。

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

2.2.15 DIR_TBL 構造体

ファイル格納情報を格納します。

```
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、1980～2079 年
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

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

2.2.16 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　:バーコード読込完了
KEY_IO_EXT　:IO ボックス検出
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

2.2.17 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	:バーコード読込完了
KEY_CLR_EXT	:CLR キー押下
KEY_IO_EXT	:IO ボックス検出
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

※ 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

2.2.18 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

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		DE		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	BS		(	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			+	;	K	[	k	{			ォ	サ	ヒ	ロ		
C	CL	→	,	<	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	゜		

1 文字入力関数では上位にキーコードを返し、エコーバックあり指定のときに表示します。文字列/数値入力では制御コード(BS、LF、その他)のみを処理し、その他は無視します。

1 文字/文字列/数値入力関数では上位にコードを返し、エコーバックあり指定のときに表示します。ただし、数値入力の場合、数値/＋/－/. のみが有効で、その他のキーは無視します。

attr が FFh の場合

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

2.2.19 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;
} 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		

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

2.2.20 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 1980～2079 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

2.2.21 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

2.2.22 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。

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

Itrf[6]

ITF 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

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

※ RSS-14 は 2007 年 2 月に GS1 DataBar Omnidirectional に名称変更されています

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

※ RSS Limited は 2007 年 2 月に GS1 DataBar Limited に名称変更されています

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

※ RSS Expanded は 2007 年 2 月に GS1 DataBar Expanded に名称変更されています

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
※ RSS-14 Stacked は 2007 年 2 月に GS1 DataBar Stacked に名称変更されています

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
※ RSS Expanded Stacked は 2007 年 2 月に GS1 DataBar Expanded Stacked に名称変更されています

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。

参照

OBR_moderd 関数、OBR_modewt 関数

2.2.23 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

2.2.24 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

2.2.25 COM_STS 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{
   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。

参照

c_bfsts 関数

2.2.26 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	:秒単位にデータ読込/書込待ち時間を設定
THROUGH	:データ読込/書込待ちを行なわない
FOREVER	:タイマ指定なしでデータ読込/書込待ちを行なう

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

2.2.27 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

2.2.28 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。この値は読み取り専用です。

参照

BT_GetLocalInfo 関数、BT_SetLocalInfo 関数

2.2.29 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

2.2.30 sys_tty 構造体

マルチドロップ/DT500 プロトコルにおいて通信パラメータを格納します。

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

メンバ

speed

転送速度を格納します。

B_1200	:ボーレート 1200bps の指定
B_2400	:ボーレート 2400bps の指定
B_4800	:ボーレート 4800bps の指定
B_9600	:ボーレート 9600bps の指定
B_19200	:ボーレート 19.2Kbps の指定
B_38400	:ボーレート 38.4Kbps の指定
B_57600	:ボーレート 57.6Kbps の指定
B_115200	:ボーレート 115Kbps の指定

length

データ長を格納します。

CHAR_7	:データ長 7 ビットの指定 (マルチドロップ時は指定できません)
CHAR_8	: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。

マルチドロップ時	意味
PARI_NONE	:パリティなしの設定
PARI_ODD	:奇数パリティの指定
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。

STOP_1	:1 ビットパリティ指定
STOP_2	:2 ビットパリティ指定

参照

cu_open 関数(マルチドロップ)、cu_open 関数(DT500)

2.2.31 CU_FILE_INFO_FORM 構造体

マルチドロッププロトコルにおいて転送ファイル情報を格納します。

```
typedef struct    {
   UB fileName[11];
   UB stat;
} CU_FILE_INFO_FORM;
```

メンバ

fileName

転送ファイル名を格納します。

例) config.hts の場合

C	O	N	F	I	G			H	T	S

stat

転送結果を格納します。

CU_STAT_TRANS	:正常終了
CU_STAT_OPEN_ERR	:転送ファイルのオープンエラー
CU_STAT_READ_ERR	:転送ファイルのリードエラー
CU_STAT_WRITE_ERR	:転送ファイルのライトエラー
CU_STAT_SEND_ERR	:転送ファイルの送信側エラー
CU_STAT_PRE_TRANS	:転送未処理

参照

cu_fileSend 関数、cu_fileSend1 関数

2.2.32 CU_RSPRM 構造体

FLINK プロトコルにおいて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

2.2.33 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

2.2.34 CU_ERRINFO 構造体

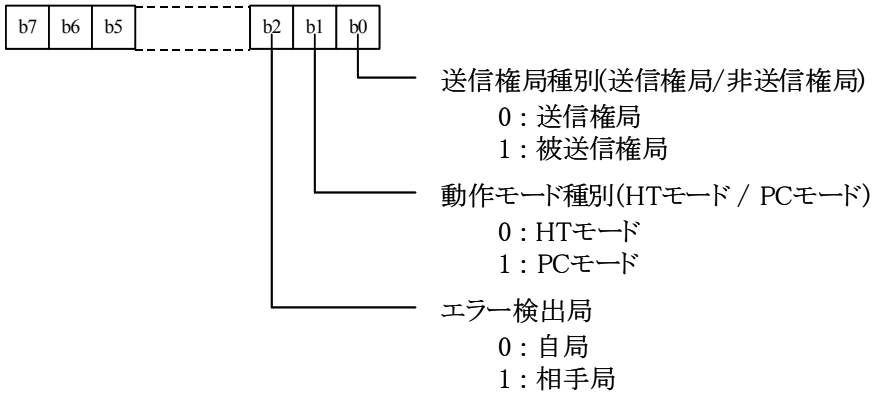
FLINK プロトコルにおけるエラー情報を格納します。

```
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	・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	・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	・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	・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	・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	・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	・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	・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　・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　・CONFIG.HTS 異常 ・ファイルレイアウトを確認
	03	相手局不正　・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

2.2.35 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

2.2.36 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(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

参照

cu_getFileInfo 関数、cu_setFileInfo 関数

2.2.37 CU_DINFO 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 {
   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

2.2.38 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

3. C 言語標準関数

下表は、“ Hシリーズ C 言語”に含まれる C 言語標準関数のうち、DT-930 で使用可能な関数の一覧です。それぞれの関数仕様は、”H シリーズ C 言語マニュアルライブラリ編”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。
低水準関数の詳細は「9.2 関数リファレンス(p.93)」を参照してください。

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
atol	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-930 のシステムデータ管理について説明します。

4.1 システムデータ

システムデータとは DT-930 の基本動作を決めるパラメータであり、dat_system 関数、システムメニュー、およびシステムデータファイルを使って設定することができます。

電源 (DAT_PWR_STR 構造体 参照)

対象	範囲	初期値	初期化のタイミング	
			リセット立上げ	設定ファイル反映
APO 時間(分)	0～59	10	○	○
ABO 時間(分)	10～59	15	○	○
レジューム ON/OFF	RESUME_ON RESUME_OFF	RESUME_ON	○	○

キー (DAT_KEY_STR 構造体 参照)

対象	範囲	初期値	初期化のタイミング	
			リセット立上げ	設定ファイル反映
クリック音 ON/OFF	CLICK_ON CLICK_OFF	CLICK_ON	○	○

表示 (DAT_DSP_STR 構造体、DAT_DSP_STR2 構造体 参照)

対象	範囲	初期値	初期化のタイミング	
			リセット立上げ	設定ファイル反映
フォント MODE	FONT_6SET FONT_8SET FONT_10SET	FONT_8SET	FROM 内容反映	○
フォント種別	FONT_NORMAL FONT_BOLD	FONT_NORMAL	○	○
日本語/英語	JPN_SET ENG_SET	JPN_SET	FROM 内容反映	○
コントラスト	0～15	7	○	○
コントラスト設定差分	-7～7	0	○	○

OBR (DAT_OBR_STR 構造体 参照)

対象	範囲	初期値	初期化のタイミング	
			リセット立上げ	設定ファイル反映
バーコード連続読取り回数	1～9	1	○	○
読取りコード照合回数	1～9	3	○	○
スキャンタイムアウト時間(秒)	1～9	4	○	○

通信 (DAT_COMINF_STR 構造体、DAT_COM_STR 構造体 参照)

対象	範囲	初期値	初期化のタイミング	
			リセット立上げ	設定ファイル反映
プロトコル種類	PRT_MULTI PRT_FLINK PRT_DT500	PRT_FLINK	○	○
通信ポート	IR_PORT	IR_PORT	○	○
通信速度(bps)	B_2400 B_4800 B_9600 B_19200 B_38400 B_57600 B_115200	B_11520	○	○
データ長(bit)	CHAR_7 CHAR_8	CHAR_8	○	○
パリティ	PARI_NON PARI_EVEN PARI_ODD	PARI_NON	○	○
ストップビット(bit)	STOP_1 STOP_2	STOP_1	○	○

タイマ (DAT_TIM_STR 構造体 参照)

対象	範囲	初期値	初期化のタイミング	
			リセット立上げ	設定ファイル反映
音量	BUZZ_OFF BUZZ_LOW BUZZ_MID BUZZ_LOUD	BUZZ_MID	○	○

システム (DAT_SYS_STR 構造体、DAT_SYS_STR2 構造体 参照)

対象	範囲	初期値	初期化のタイミング	
			リセット立上げ	設定ファイル反映
機器 ID(6 桁)			FROM 内容反映	config.id
代理店 ID(6 桁)			-	config.pas
OS バージョン(6 桁)			-	-
PATCH バージョン (6 桁)			-	-
MODEL 情報			IO ポート反映	-
アプリケーション 領域サイズ(KB) ※	128～992	336	RAM ドライブ 情報反映	指定時○
FORMAT タイプ ※	0:FAT モード 1:DT-700 互換 モード	0:FAT モード	FORMAT 情報反映	指定時○

※ この設定は dat_system ではなく、システムメニュー、または設定ファイルから行います。

通信ユーティリティ(マルチドロップ) (DAT_PRO_STR 構造体 参照)

対象	範囲	初期値	初期化のタイミング	
			リセット立上げ	設定ファイル反映
受信タイムアウト(秒)	0～99	3	○	○
リトライ回数(下位)	0～99	3	○	○
リンクタイムアウト (10 ミリ秒)	0～9990	30	○	○

通信ユーティリティ(FLINK) (DAT_PRO_STR 構造体 参照)

対象	範囲	初期値	初期化のタイミング	
			リセット立上げ	設定ファイル反映
セッション確立 タイムアウト(秒)	0～3600	1800	○	○
受信タイムアウト(秒)	0～600	300	○	○
セッション終了 タイムアウト(秒)	0～600	10	○	○

通信ユーティリティ(DT500) (DAT_PRO_STR 構造体 参照)

対象	範囲	初期値	初期化のタイミング	
			リセット立上げ	設定ファイル反映
シリアル NO	0:OFF 1:ON	0:OFF	○	○
水平パリティ	0:OFF 1:ON	0:OFF	○	○
リンクタイムアウト	0～240	0	○	○

4.2 システムデータファイル

システムデータをファイルから設定することが可能です。
このファイルは、次の 3 種類があります。

	ファイル名	形式	内容
機器 ID ファイル	CONFIG.ID	DOS	6 桁の機器 ID
代理店 ID ファイル	CONFIG.PAS	DOS	6 桁の代理店 ID
config ファイル	CONFIG.HTS	DOS	機器 ID/代理店 ID 以外のシステムデータ

4.2.1 config ファイル

次ページに CONFIG.HTS のファイル構造を示します。

CONFIG.HTS のファイル構造

項目			位置	サイズ	設定範囲	既定値
ID			00	10	CONFIG.HTS 固定	CONFIG.HTS
電源		APO 時間	+10	2	00-59(分)	10
		ABO 時間	+12	2	00,10-59(秒)	15
		レジューム	+14	2	00:OFF / 01:ON	01
KEY		クリック音	+16	2	00:OFF / 01:ON	01
OBR		読取回数	+18	2	01-09(回)	01
		照合回数	+20	2	01-09(回)	03
		スキャン時間	+22	2	01-09(秒)	04
表示		MODE	+24	2	00:6dot / 01:8dot / 02:10dot	01
		日/英	+26	2	00:日本語 / 01:英語	00
		種別	+28	2	00:NORMAL / 01:BOLD	00
		コントラスト	+30	2	00-15(段)	7
通信	共通	プロトコル	+32	2	00:マルチドロップ / 01:FLINK / 02:DT500	01
		PORT	+34	2	00:IR	00
	個別	速度(IR)	+36	2	02:2400 / 03:4800 / 04:9600 / 05:19200 / 06:38400 07:57600 / 08:115200(bps)	08
		データ(IR)	+38	2	07:7bit / 08:8bit	08
		パリティ(IR)	+40	2	00:NON / 01:EVEN / 02:ODD	00
		STOP(IR)	+42	2	00:1bit / 01:2bit	00
タイマ		音量	+60	2	00:OFF / 01:小 / 02:中 / 03:大	02
プロトコル		下記参照	+62	14		
ファイルモード		FORMAT	+76	3	F00:FAT ファイルモード / F01:DT700 互換モード	F00
サイズ		アプリケーションサイズ	+79	5	M0128-M0992(Kbyte) 16Kbyte バウンダリで指定	M0336

プロトコル関連:マルチドロップ

項目	位置	サイズ	設定範囲	既定値
受信タイムアウト	+62	2	00-99(秒)	03
リトライ回数	+64	2	00-99(回)	03
リンクタイムアウト	+66	4	0000-9990(10m 秒)	0030
予約領域	+70	2	本機では無効なパラメータです。	00
予約領域	+72	2	本機では無効なパラメータです。	00
予約領域	+74	2	本機では無効なパラメータです。	00

プロトコル関連:FLINK

項目	位置	サイズ	設定範囲	既定値
セッション確立タイムアウト	+62	4	0000-3600(秒)	1800
受信タイムアウト	+66	4	0000-0600(秒)	0300
セッション終了タイムアウト	+70	4	0000-0600(秒)	0010
予約領域	+74	2	00	00

プロトコル関連:DT500

項目	位置	サイズ	設定範囲	既定値
シリアル NO.	+62	2	00:OFF / 01:ON	00
水平パリティ	+64	2	00:OFF / 01:ON	00
リンクタイムアウト	+66	4	0000-0240(秒):実際の設定は 30 秒単位	0000
予約領域	+70	6	000000	000000

4.3 関数リファレンス

DT-930 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 バージョン取得
dat_dealer_chk	代理店 ID のチェック
dat_mem_size	メモリ領域空きサイズの取得

4.3.1 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、設定するかを次の値で指定します。

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　　(DT-700 互換)
SYSD_DSP2　　〃　　(DT-900 互換)
SYSD_COM　　:通信　　(共通)
SYSD_COM0　　:IrDA/カシオ IR インタフェース
SYSD_TIM　　:タイマ
SYSD_SYS　　:システム　　(DT-700 互換)
SYSD_SYS2　　〃　　(DT-900 互換)
SYSD_PRO　　:プロトコル

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_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

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 関数は失敗します。

sys_id パラメータに上記以外の値を指定した場合、dat_system 関数は失敗します。

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、すべてのシステムデータの登録を行ないません。

4.3.2 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

4.3.3 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。

4.3.4 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。

5. 電源管理

ここでは、DT-930 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。詳しくは「イベント通知機能(p.117)」を参照してください。

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(マルチドロップ、FLINK、DT500 プロトコル)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-930 の電源を OFF することができます。

IO BOX 起動
pwr_IoboxBootMode 関数を使用して、IO BOX の接続を検出したときに DT-930 の電源を自動的に ON するように設定できます。

5.1 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_hold_apo	APO 禁止の設定
pwr_off	電源の OFF
pwr_IoboxBootMode	IO ボックスの起動設定

5.1.1 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

5.1.2 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

5.1.3 pwr_IoboxBootMode

DT-930 を IO BOX 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　　: IO BOX に載せたときに、自動的に電源を ON します。

PWR_OFF　　: IO BOX に載せたときに、自動的に電源を ON 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_PRM　　:パラメータエラー

6. ブザー鳴動

DT-930 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は鳴動中ブザーの停止が入っています
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。

6.1 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

6.2 関数リファレンス

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

6.2.1 s_beep

4096Hz、100msec のエラービープ音を鳴らします。

```
void s_beep(void);
```

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。

6.2.2 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したがいます。

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

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 1980 年 1 月 1 日～2079 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_timeget 関数で設定範囲外の時刻を設定使用とした場合、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_dateget	日付の取得
s_dateset	日付の設定
s_timeget	時刻の取得
s_timeset	時刻の設定

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 1980 年 1 月 1 日～2079 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。

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設定使用とした場合、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_timeget 関数

9. ファイル管理

DT-930 では次のドライブに対して、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。

ドライブレター	場所	概要
A	RAM 領域	次の 2 種類のファイルシステムを排他利用可能※。 • MS-DOS 互換 FAT ファイルシステム • DT-700 互換ファイルシステム
B	フラッシュディスク	次のファイルシステムを利用可能。 フラッシュディスク専用ファイルシステム

※ システムメニューを使用して切替え可能です。
※ 切替えた時点で内容を再構築します。(格納情報はすべて消去します)

9.1 ファイルシステム

DT-930 では、次のファイルシステムをサポートします。
- MS-DOS 互換 FAT ファイルシステム
- DT-700 互換ファイルシステム
- フラッシュディスク専用ファイルシステム

MS-DOS 互換 FAT ファイルシステム
FAT システムを用いた MS-DOS 互換ファイルシステムです。
- ファイル領域はリニア空間ではありません。
- ファイル格納情報の取得関数(dat_fdir)を使用して、ファイル領域を直接参照することはできません。

MS-DOS 互換ファイルシステムでの、ファイル領域の構成を以下に示します。

BPB	FAT	ディレクトリ情報	ファイルデータ	チェックサム

BPB	:BIOS Parameter Block
ファイルデータ	:ファイル部データ領域
チェックサム	:チェックサム格納領域(セクタ単位に 4 バイト加算値)

DT-700 互換ファイルシステム
DT-700 と互換動作を行なうファイルシステムです。
- ファイル領域はすべてリニア空間で動作します。
- ファイルの追加書込みの際には、領域の再構成処理を行ないます。
- ファイル格納情報の取得関数(dat_fdir)を使用して、ファイル領域を直接参照することが可能です。

DT-700 互換ファイルシステムでの、ファイル領域構成を以下に示します。

フォーマット判定領域	チェックサム領域	チェックサム判定領域	ファイルデータ

フォーマット判定領域	:フォーマットの有無判定用領域(16 バイト)
チェックサム領域	:ファイルデータ部のチェックサム格納領域 (256 バイト単位に 1 バイト加算値)
チェックサム判定領域	:256 バイト単位のチェックサム格納領域に対応した、 チェックサム判定結果(1 ビット単位で表現)
ファイルデータ	:ファイル部データ領域

フラッシュディスク専用ファイルシステム

本体内蔵のフラッシュメモリを利用した、専用のファイルシステムです。

- 本体の電源がない場合でも、データが消えることはありません。

フラッシュディスク専用ファイルシステムのファイル領域構成を以下に示します。

BPB	ディレクトリ情報	データサイズ	ファイルデータ	チェックサム	……	データサイズ	ファイルデータ	チェックサム

データサイズ	:ファイルデータサイズ
ファイルデータ	:ファイル部データ領域 (最大 64KB でファイルデータを分割)
チェックサム	:ファイルデータ単位に 4 バイト加算値

それぞれのファイルシステムの特徴を以下に示します。

	MS-DOS 互換 FAT	DT-700 互換	フラッシュディスク専用
ドライブの概念	あり RAM ドライブ:A ドライブ	あり ドライブ指定がない場合は、RAM ドライブとしてあつかいます	あり FLASH ドライブ:B ドライブ
ファイル数	ルート:最大 192 個、 ディレクトリ下:無制限 (ファイル領域が許す限り)	最大 192 個	最大 64 個
同時オープン数	16	16	16
ディレクトリ	サポート	未サポート	未サポート
総容量	738KB～1.6MB (デフォルト:1.39MB)	738KB～1.6MB (デフォルト:1.39MB)	12.5MB
セクタサイズ	512	256	概念なし
セクタ/クラスタ	2	1	なし
予約セクタ	1	なし	なし
メディア ディスクリプタ	1	なし	なし
セクタ/FAT	3～5 (デフォルト:5)	概念なし	セクタ/FAT 概念なし
セクタ/トラック	0	0	0
ヘッド番号	0	0	0
総セクタ数	1476～3205 (デフォルト:2788)	5576(2952～6410)	セクタ/FAT 概念なし

9.2 関数リファレンス

DT-930 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_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_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。

9.2.1 fil_mkdir

ディレクトリを作成します。(FAT ファイルモード専用)

```
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

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、fil_mkdir 関数は失敗します。

9.2.2 fil_rmdir

ディレクトリを削除します。(FAT ファイルモード専用)

```
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

すでに最大同時オープン数をオープンしている場合、fil_rmdir 関数は失敗します。

9.2.3 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、fil_remove 関数は失敗します。

9.2.4 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、fil_rename 関数は失敗します。

すでに最大ファイル数のファイルが存在する場合、fil_rename 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。fil_rename 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。

9.2.5 fil_fstat

ファイル番号を指定して、ファイルの属性を取得します。

```
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

9.2.6 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 fil_chsize 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、fil_chsize 関数は失敗します。

9.2.7 fil_getsize

ファイル領域の空きサイズを取得します。

```
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

9.2.8 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

※ アーカイブ/読取り専用属性は、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

9.2.9 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、fil_findnext 関数は失敗します。

参照

FIND_T 構造体、fil_findfirst 関数

9.2.10 fil_filesize

ファイルの個数と総サイズを取得します。

```
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。

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 fil_filesize 関数は正常終了します。

fil_filesize 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 以上)

参照

FIL_SIZE 構造体

9.2.11 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。

fil_filefind 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 以上)
- ファイル未検出
- 指定ファイルが存在しない

9.2.12 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込みます
DAT_FILE_NEXT　　:次のファイル管理テーブルを読込みます

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込位置指定エラー
E_NG　　:ファイル格納情報が存在しない

参照

DIR_TBL 構造体

9.2.13 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。

9.2.14 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、dat_fdel 関数は失敗します。

name パラメータに空白を指定した場合、dat_fdel 関数は失敗します。

9.2.15 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、dat_fdel 関数は失敗します。

9.2.16 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

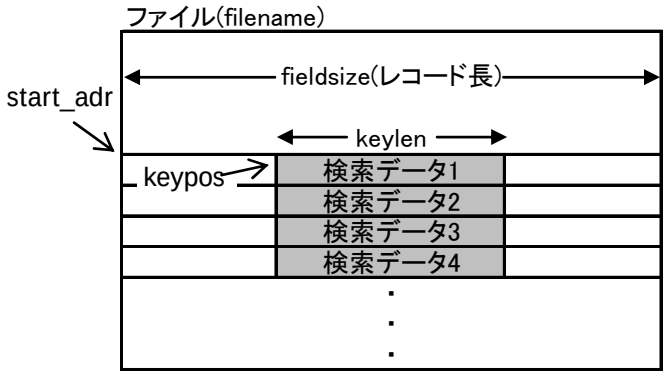
関数が成功すると E_OK が返ります。失敗すると、次のエラーが返ります。

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 dat_F_Search 関数を実行下場合、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

9.2.17 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	:読取り専用でオープンします
O_WRONLY	:書込み専用でオープンします
O_RDWR	:読取り/書込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	:読取り/書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、読取り/書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、open 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。open 関数は失敗します。

9.2.18 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。
close 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。

9.2.19 read

指定ファイル番号に対応したファイルの読出し位置から指定読込領域へ指定データバイト数分格納ファイルデータを読込みます。

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

パラメータ

fileno
読込み対象のファイル番号を指定します。

buf
読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、read 関数は失敗します。

書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。

9.2.20 write

ファイルにデータを書込ます。書込位置は、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
書込み対象のファイル番号を指定します。

buf
書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。

書込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。

9.2.21 lseek

指定ファイルの読込/書込位置をバイト単位で設定します。

```
int 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
読込み/書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　:現在の読込み/書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オフセットが返ります。
失敗すると、次のエラーが返ります。
E_LOWERR　:異常終了

解説

オフセット位置が負になる場合とファイル終端をこえる場合は、現在位置を更新しません。

9.2.22 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 *buf= 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

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、一定時間マシン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)、I/O ボックス接続、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

- タイマイベント
  設定したタイマのタイムアップ

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。

10.1 通知イベントの種類

10.1.1 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。

IO ボックス接続検出イベント
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 IO ボックス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)	シンボル表示	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	IO ボックス接続検出	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。

10.1.2 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	右トリガーキー		不可
	左トリガーキー		

※ マルチファンクションキーに OBR 読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 の読込みは開始されません。

10.1.3 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。

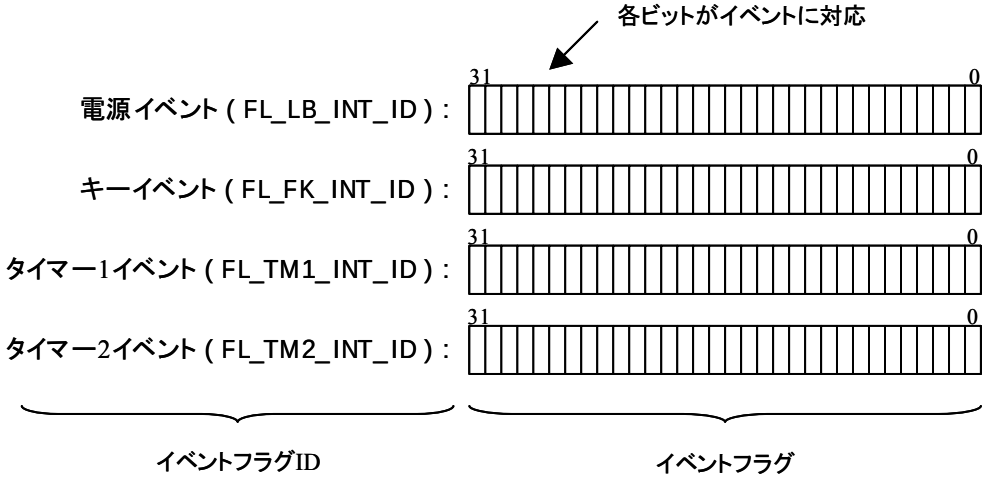
項目	仕様
最小単位	31.25msec
設定時間	1(31.25msec)～115200(1Hour)
誤差	要求時間+(最大)31.25msec
最大登録数	10
タイムアウト時の処理	指定時間経過後、指定のイベントフラグ※によって通知します

※ イベントフラグについては「イベント通知のメカニズム(p.121)」を参照してください。

10.2 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 flg_sts 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	IO ボックス接続検出

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	

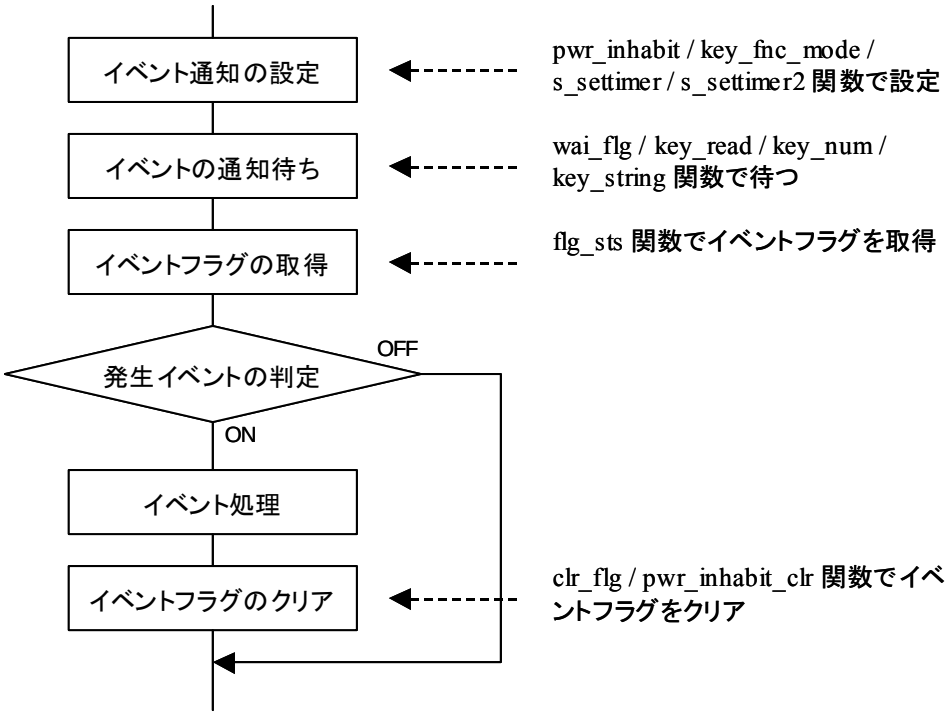
10.3 イベント通知の設定、取得とクリア

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 flg_sts 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	←	flg_sts	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			

10.4 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。



10.5 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
flg_sts	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

10.5.1 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　　:IO ボックス接続検出

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。

10.5.2 pwr_inhabit_clr

通知した電源イベント、および IO ボックス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	:IO ボックス接続検出

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。

10.5.3 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      fun_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、未設定の場合はゼロを返します。

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。※

※ key_fnc_mode 呼出し前に値をセットする必要はありません。

10.5.4 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

10.5.5 s_timerend

s_settimer 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

10.5.6 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_timerend 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

10.5.7 s_timerend2

s_settimer2 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

10.5.8 flg_sts

指定のイベントフラグ ID に対するイベントの通知状態を取得します。

```
ER flg_sts(
   ID     *p_flgpid,
   UW     *p_flgptn,
   ID     flgid
);
```

パラメータ

p_flgpid
システムがプロセス制御のために使います。アプリケーションプログラムは ID 型の領域を確保し、そのアドレスを指定します。

p_flgptn
取得したイベントの通知状態(イベントフラグ)を格納する領域のアドレスを指定します。

flgid
対象のイベントフラグ 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_NOEXS	:フラグ ID 範囲外/予約 ID
E_ILADR	:不正アドレス (p_flgptn が 4 の倍数以外または 0 の場合)

10.5.9 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します。

```
ER clr_flg(
   ID     flgid,
   UW     setptn
);
```

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。

setptn
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_NOEXS　　:フラグ ID 範囲外/予約 ID

注意

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。

10.5.10 wai_flg

指定のイベントフラグ ID に対するイベントが発生し、待機条件を満たすまで待機します。

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

パラメータ

p_flgptn
wfmode に TWF_CLR を指定すると、本関数を抜ける前にイベントの発生状態(イベントフラグ)をクリアします。そのため、クリア前の状態を取得する領域のアドレスを指定します。

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
待機条件を次の値の組み合わせで指定します。

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
TWF_CLR	:関数を抜ける際、すべてのビットをクリアします

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。

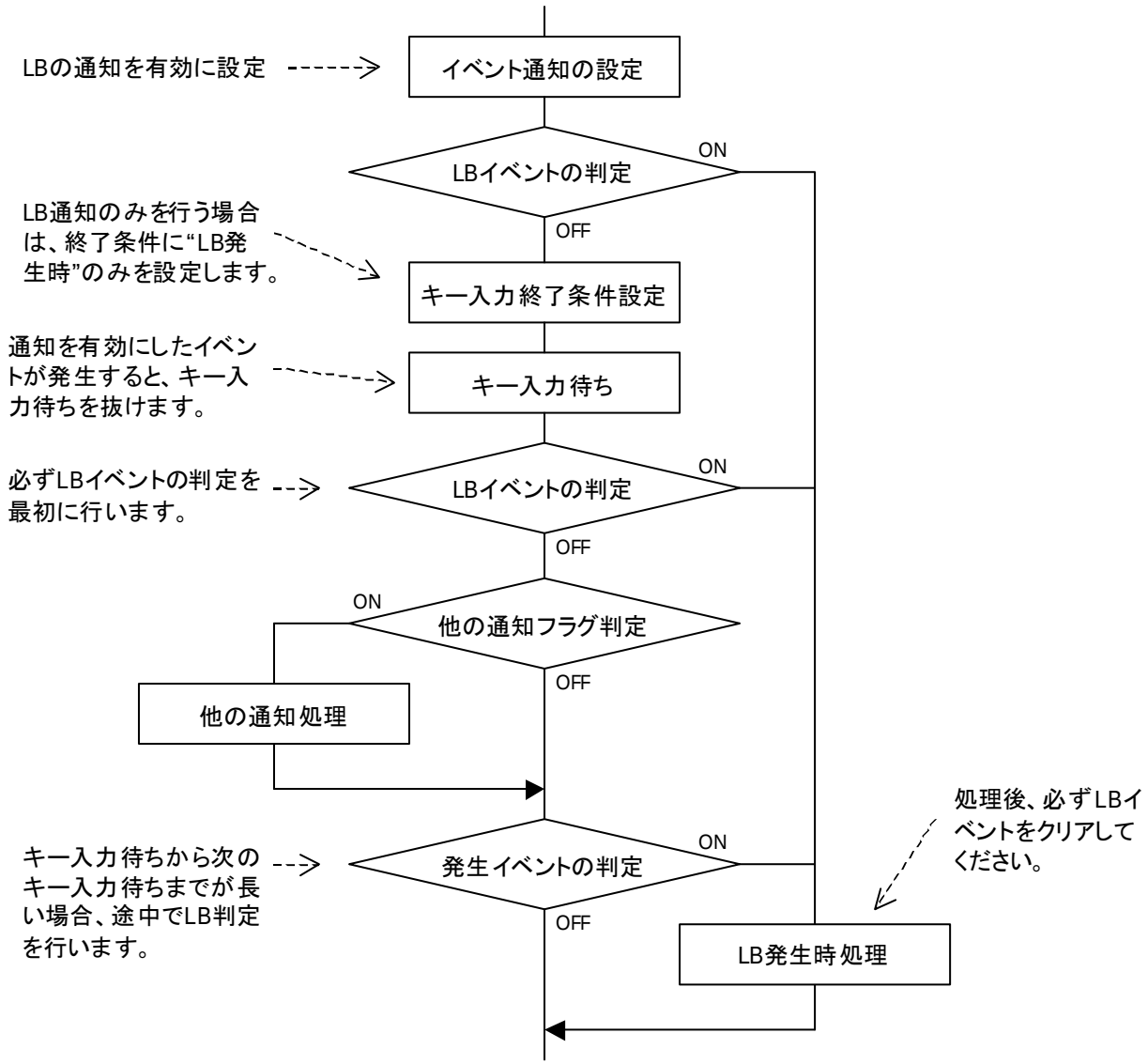
E_ILADR	:不正アドレス (p_flgptn が 4 の倍数以外または 0 の場合)
E_PAR	:パラメータエラー

10.6 サンプルコード

10.6.1 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;
ID          dummy;
UW          ptn, i;
KEY_INP     keyinfo;

/* 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 */
   err =  flg_sts( &dumy, &ptn, FL_LB_INT_ID );

   if( ptn & 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( ptn & 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( ptn & 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

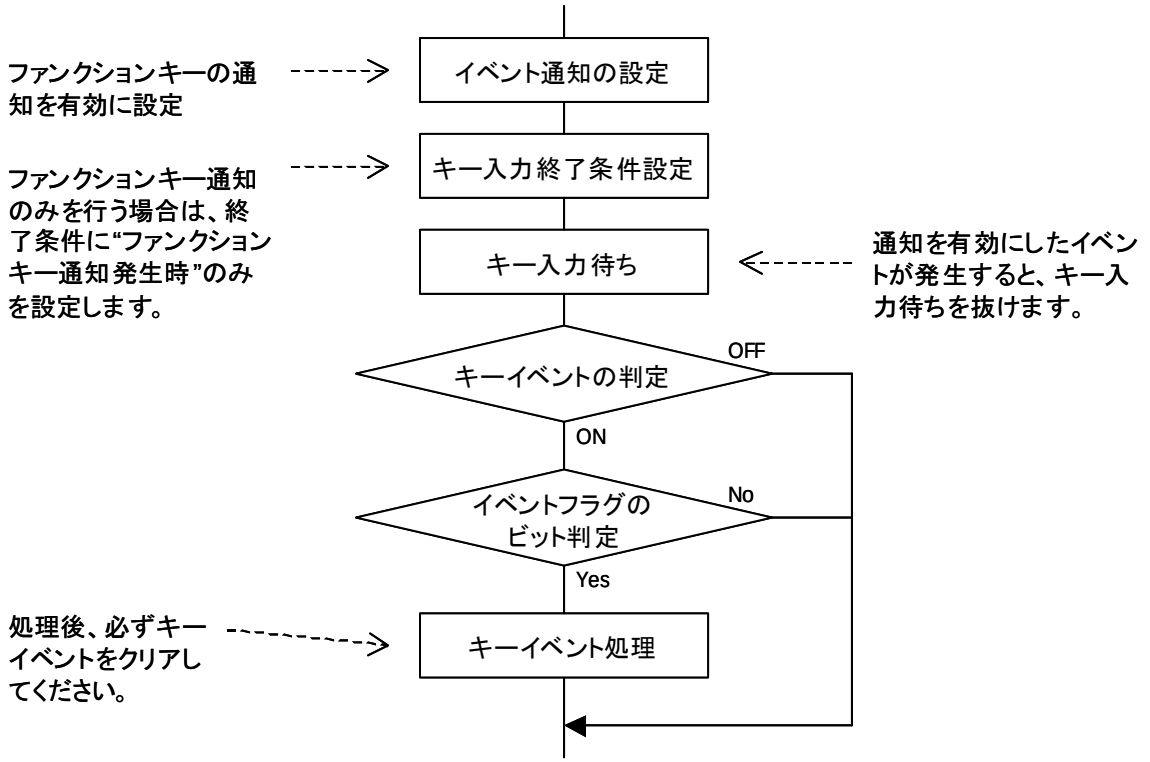
電源イベントに対するイベント通知を使用する場合、下記の点を注意してください。

- 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。

10.6.2 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, i;
KEY_INP     keyinf;
ID          dumy, 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 */
   err = flg_sts( &dumy, &ptn, FL_FK_INT_ID );
   if( ptn & 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( ptn & 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-930 の表示について説明します。
画面サイズは 128×64 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-930 のコード体系を以下に示します。

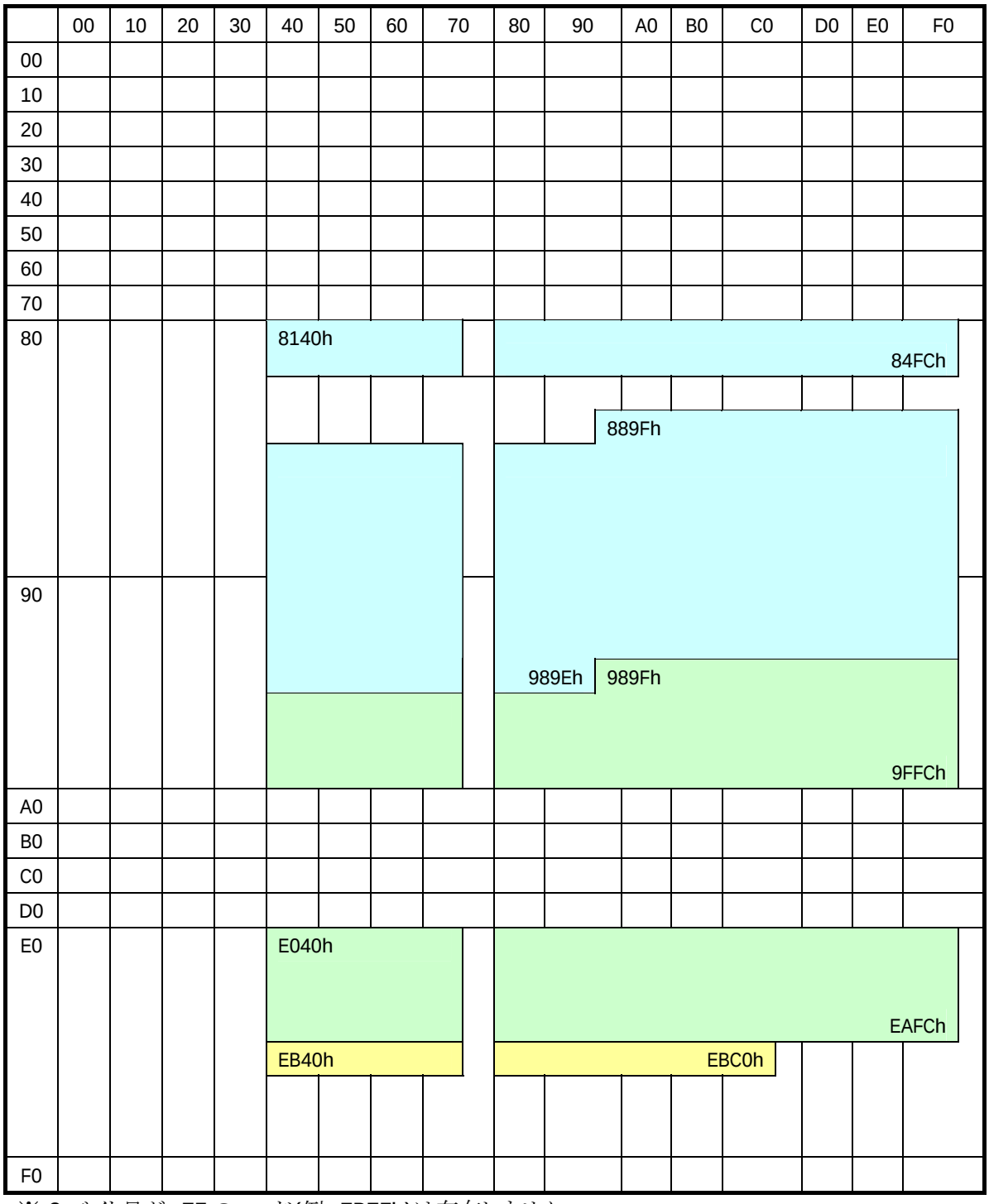
11.1.1 1 バイトコード

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		DE		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	BS		(	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			+	;	K	[	k	{			ォ	サ	ヒ	ロ		
C	CL	→	,	<	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)
ANK	01h～80h、A0h～DFh、FDh～FFh

11.1.2 2 バイトコード



※ 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-930 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-930 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

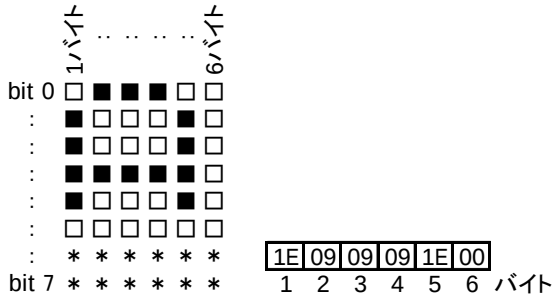
縦強調フォントを表示する場合は、dat_system 関数を使用して設定します。

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

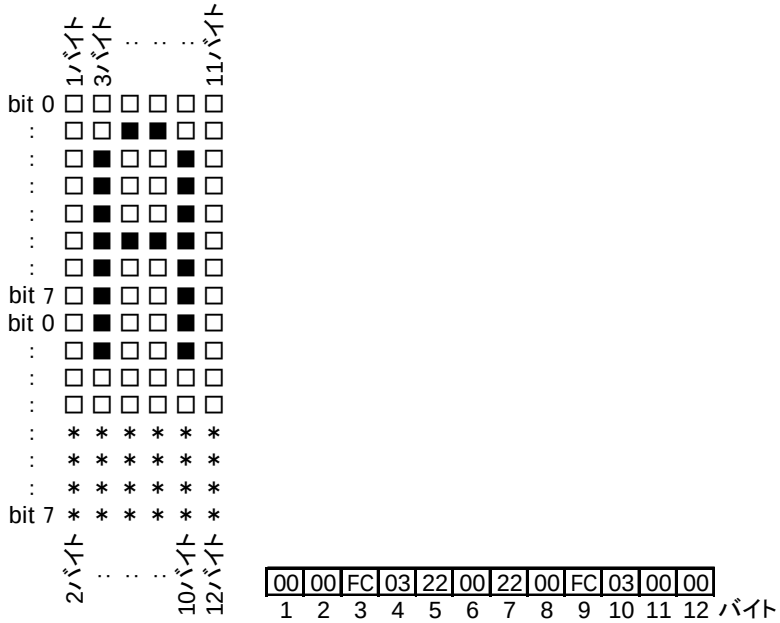
縮小 ANK (6×6 ドット)

1 フォント 6 バイト構造



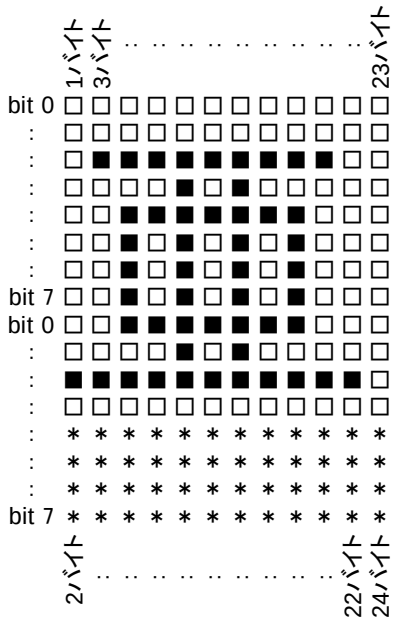
標準 ANK (6×12 ドット)

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



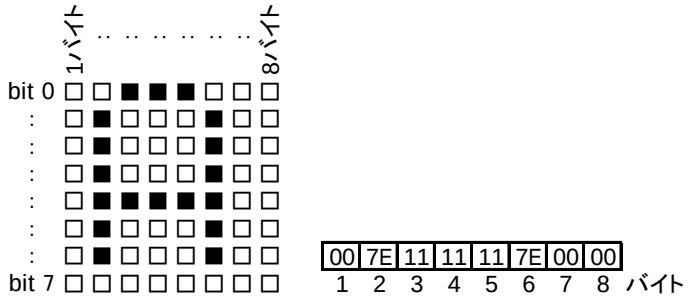
00	04	04	04	F4	05	14	05	FC	07	14	05
1	2	3	4	5	6	7	8	9	10	11	12 バイト

FC	07	14	05	F4	05	04	04	00	04	00	00
13	14	15	16	17	18	19	20	21	22	23	24 バイト

11.3.2 8ドットフォント

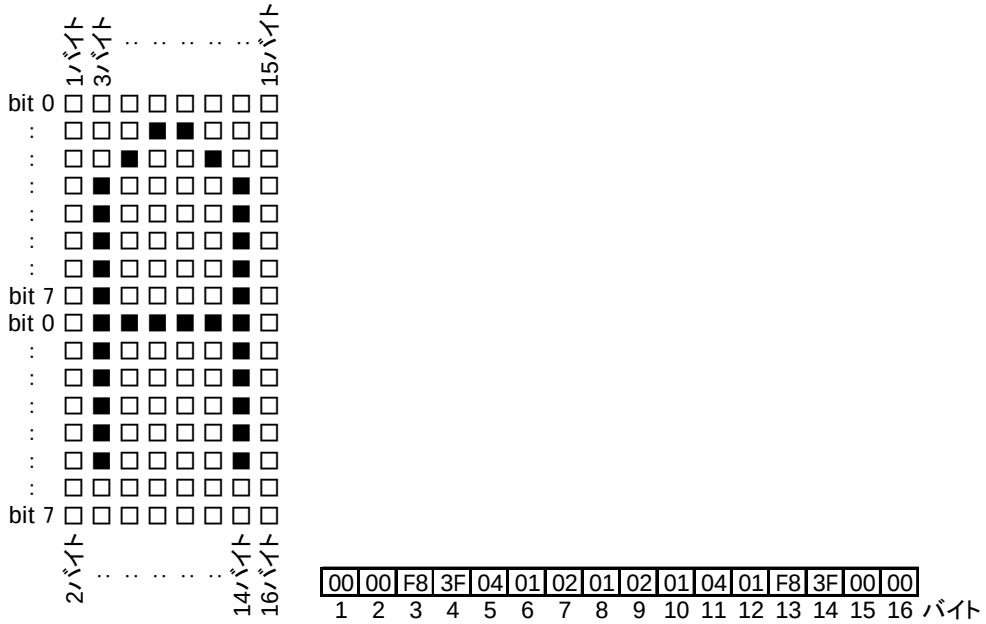
縮小 ANK (8×8 ドット)

1 フォント 8 バイト構造



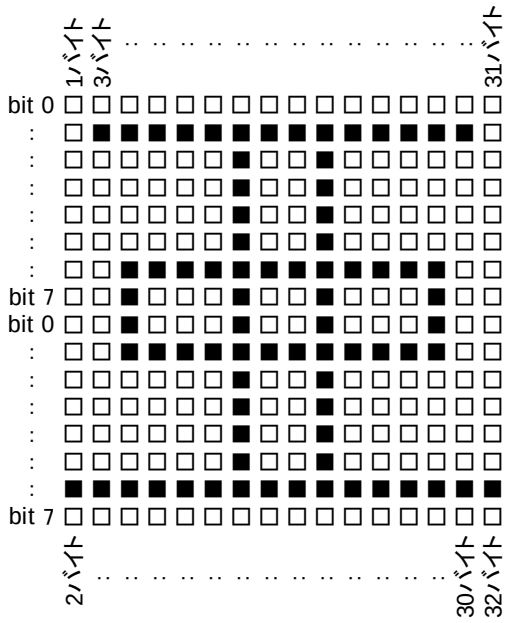
標準 ANK (8×16 ドット)

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



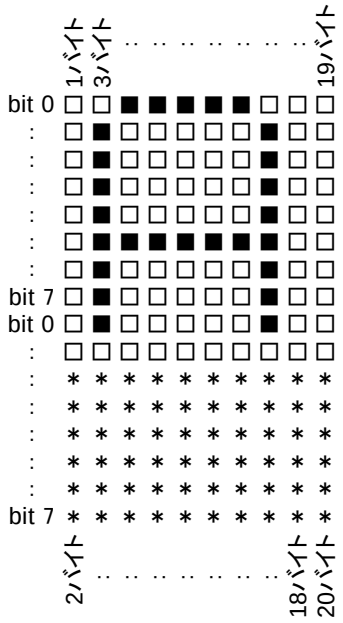
00	42	02	40	C2	47	42	44	42	44	42	44	FE	7F	42	44
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16 バイト

42	44	7F	FE	42	44	42	44	42	44	C2	47	02	40	00	40
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32 バイト

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

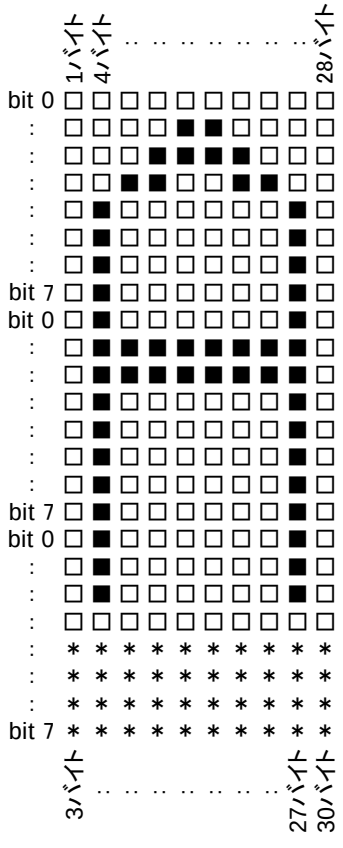


00	00	FE	01	21	00	21	00	21	00
1	2	3	4	5	6	7	8	9	10 バイト

21	00	21	00	FE	01	00	00	00	00
11	12	13	14	15	16	17	18	19	20 バイト

標準 ANK (10×20 ドット)

1 フォント 30 バイト構造



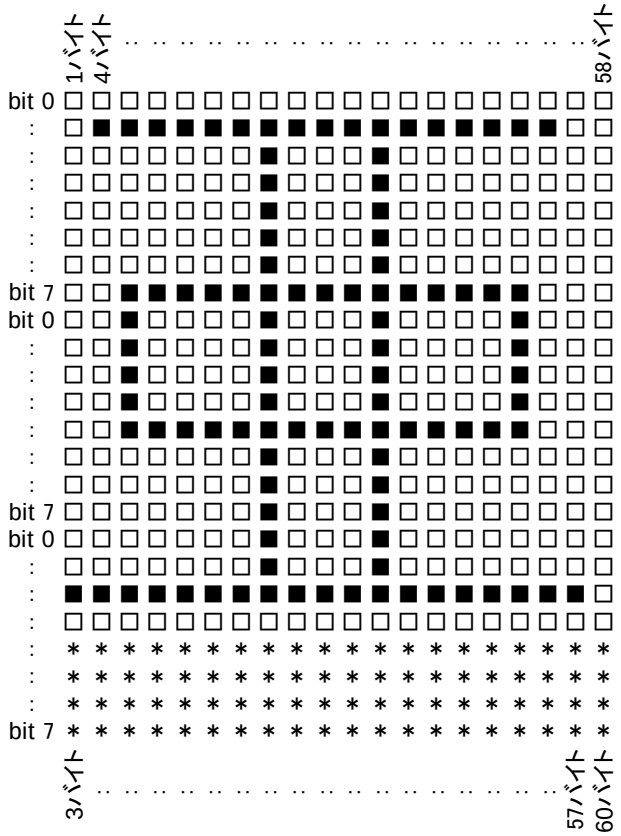
00	00	00	F0	FF	07	18	0C	00	0C
1	2	3	4	5	6	7	8	9	10 バイト

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

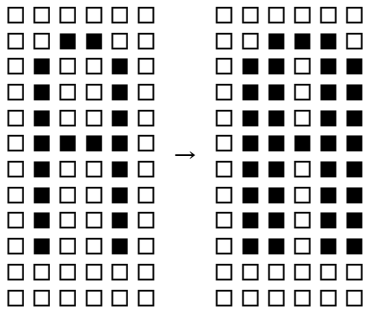
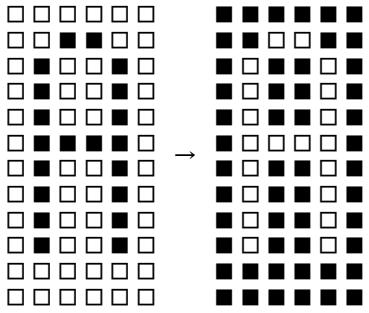
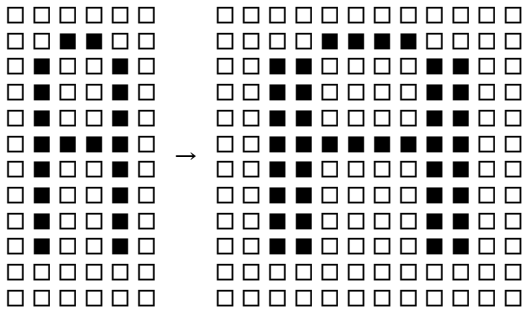
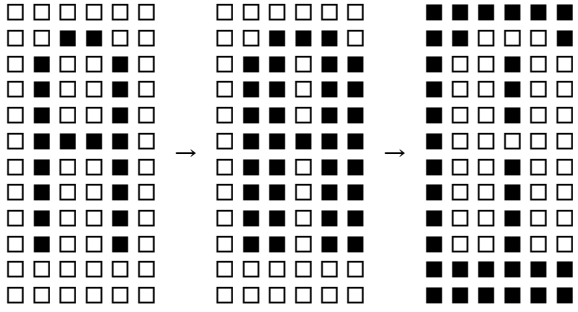
バイト

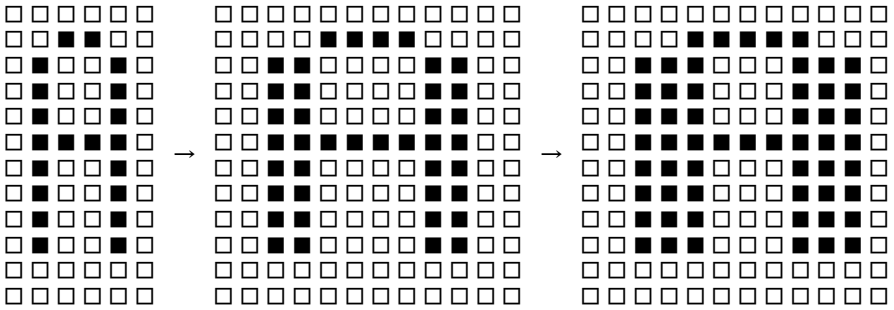
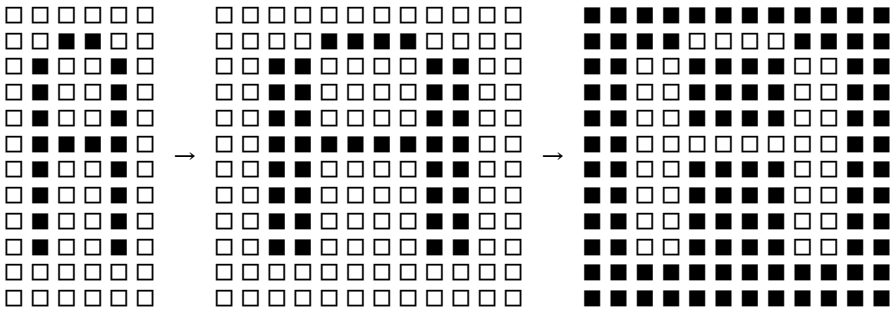
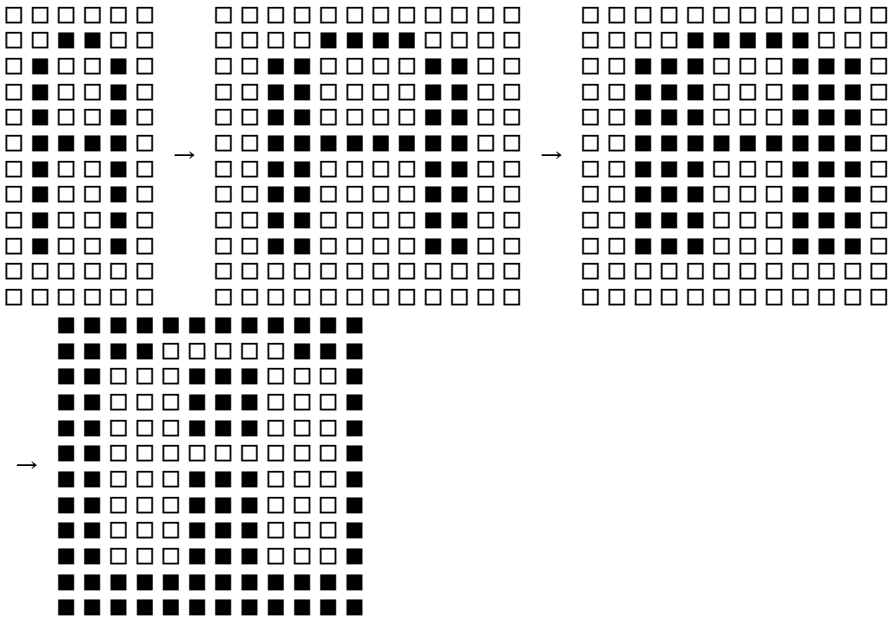
04	02	00	04	00	00	04	00	00	00
51	52	53	54	55	56	57	58	59	60

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	フォントデータのビットパターンを右へずらして、OR します。 
反転	フォントデータのビットパターンを反転します。 
横倍角	フォントデータ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-930 は、従来機種(DT-900)を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-930 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-930 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-930 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-700 互換表示

11.6.1 表示座標系

DT-930 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。

モード	桁数×行数	最大表示文字数
6 ドット	21×10	210
	21×5	105
	10×5	50
8 ドット	16×8	128
	16×4	64
	8×4	32
10 ドット	12×6	72
	12×3	36
	6×3	18

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(127,63)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	制御コード表示(p.159)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	40～7E,80～FC	漢字コード表の文字	ユーザー登録の文字
89～9F,E0～EA	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)

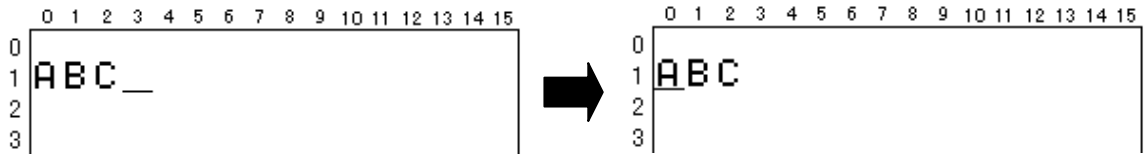
1 バイト目	2 バイト目	ROM フォント時	ユーザーフォント指定時
00		文字列表示の終了	
0A,0D		コントロールコード (制御コード表示(p.159)参照)	
1B		ESC 制御(ESC シーケンス(p.160)参照)	
01～09,0B,0C 0E～1A,1C～1F 81～9F,E0～FC		ANK スペース	ユーザー登録の文字
20～80,A0～DF FD～FF		ANK コード表の文字	ユーザー登録の文字
81～84	00	文字列表示の終了(画面表示しません)	
89～9F	40～7E,80～FC	漢字コード表の文字	ユーザー登録の文字
E0～EA	01～3F,7F,FD～FF	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。

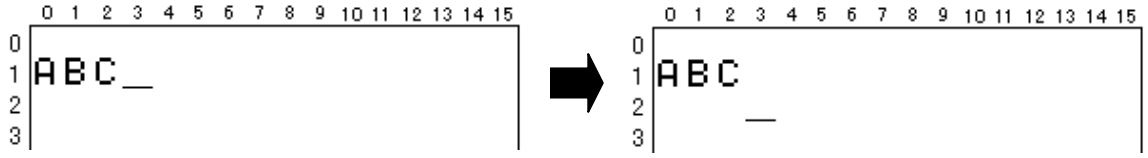
11.6.3 制御コード表示

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。

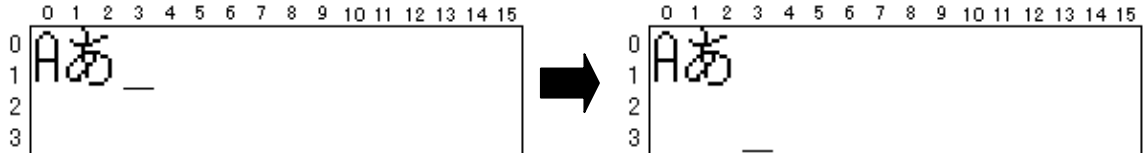
CR (0Dh)
キャリッジリターン動作を行ないます。



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

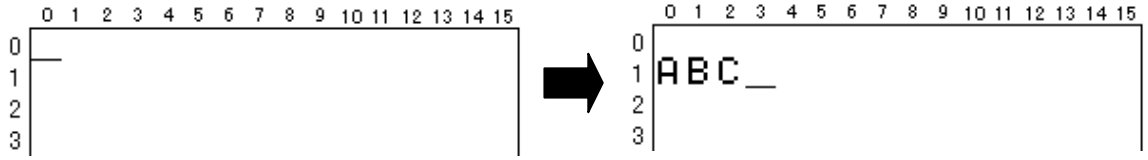


標準 ANK および漢字の文字列中に LF コードが含まれている場合は、2 行改行します。

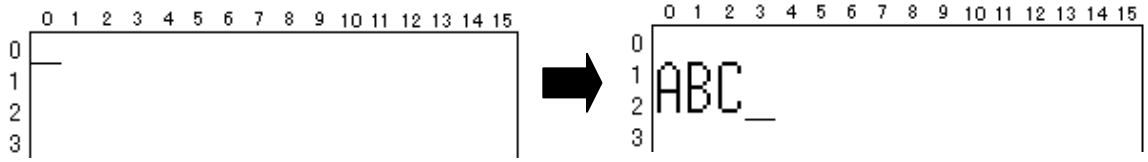


文字列表示の先頭が LF コードまたは、1 文字表示で LF コードの場合、入力パラメータの ANK モードが縮小の場合は 1 行、標準の場合は 2 行改行します。

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。

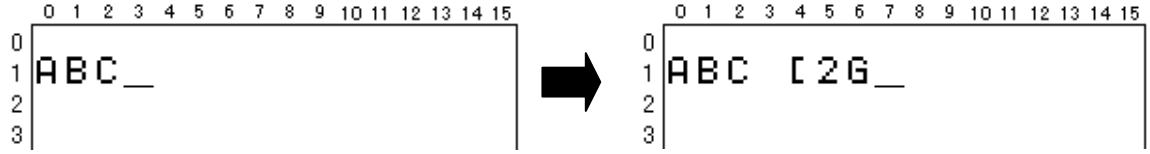
カーソル位置設定（ ESC[Pn;PmH　ESC[Pn;Pmf ）
カレントカーソル位置を移動します。
n：行を指定します。
m：桁を指定します。

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。

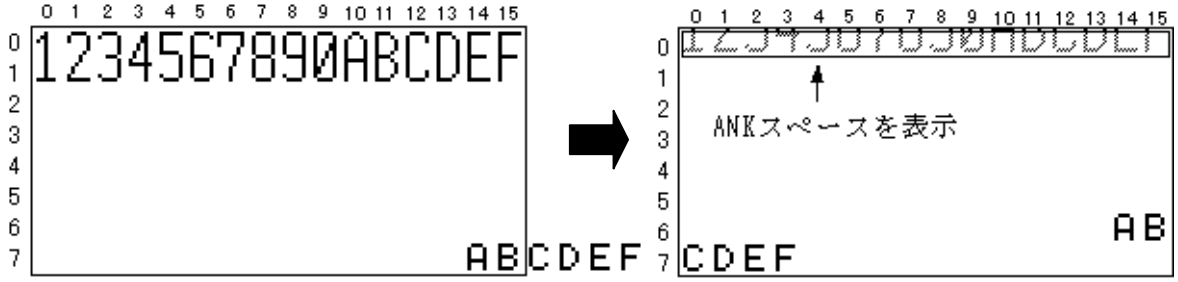
例) “ ESC [2J “ を “ ESC [2G “ と誤って指定した場合



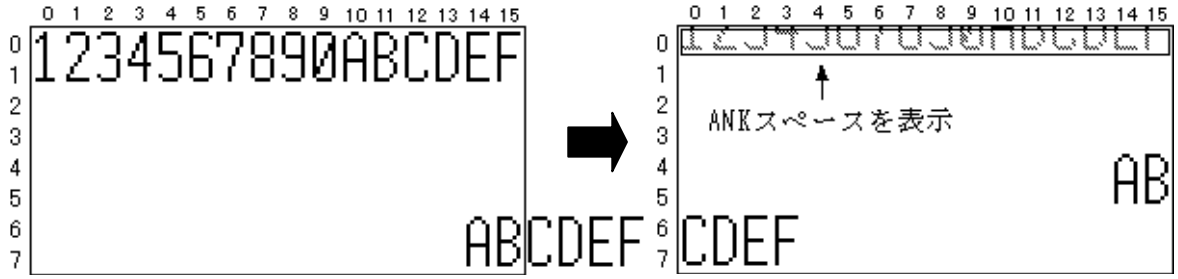
11.6.5 スクロール制御

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、標準フォントなら 1 行、縮小フォントなら2行スクロールします。
縮小フォントの表示でスクロールが発生した場合、最上行が標準フォントの表示なら縮小フォントの ANK スペースになり、縮小フォントなら1行スクロールします。

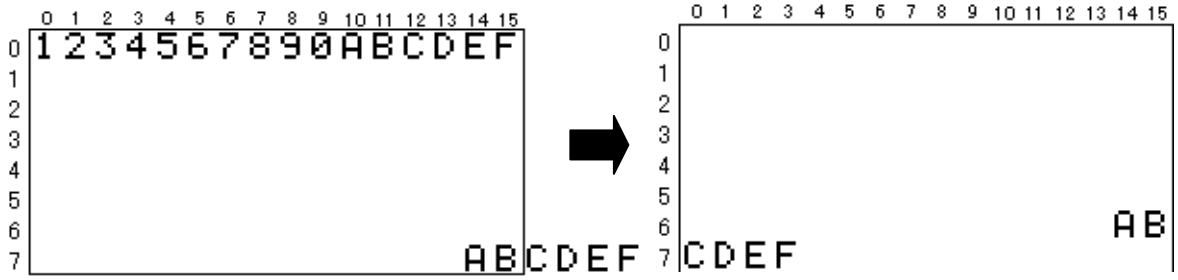
(例 1) 最下行の縮小 ANK 表示でスクロールが発生する場合
座標(14,7)に縮小 ANK”ABCDEF”を表示



(例 2) 最下行-1 の標準 ANK/漢字表示でスクロールが発生する場合
座標(14,6)に標準 ANK”ABCDEF”を表示



(例 3) 最下行の標準
ANK/漢字表示でスクロールが発生する場合
座標(14,7)に標準 ANK”ABCDEF”を表示



※ スクロール抑制
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。

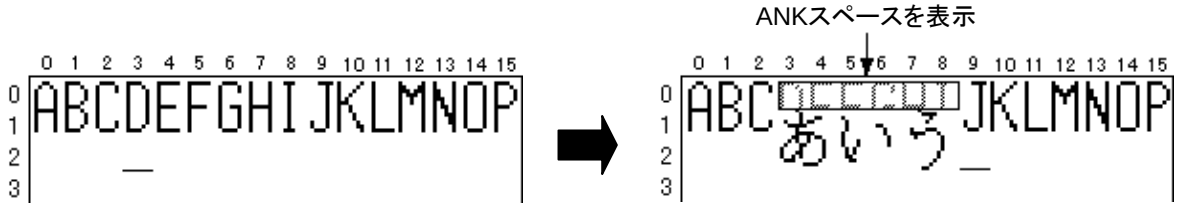
例外動作

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

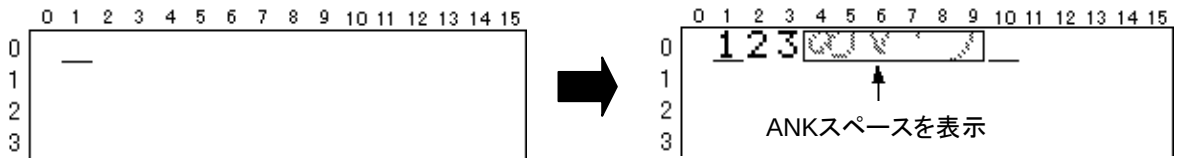


最上位行での標準 ANK/漢字表示

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。

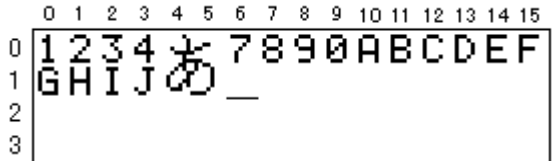
(例) 座標(1,0)に”123 あいう”を表示



行端での自動改行制御

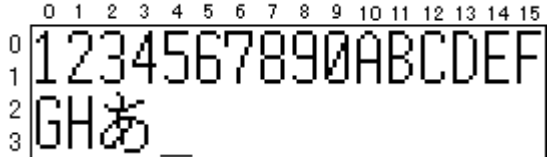
文字列表示を行なうとき、行端で表示仕切れない場合には先頭文字により、1または2行の改行を自動で行ないます。(ただし、改行モードあり指定時)

(例1)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。

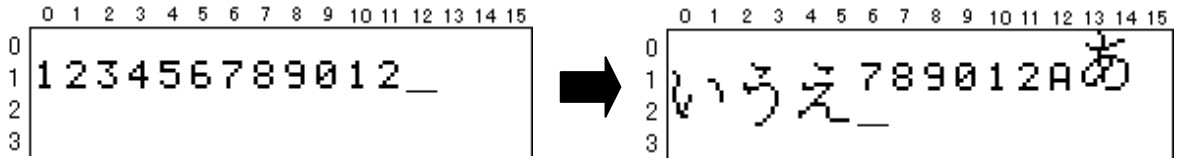
(例2) 先頭文字が標準 ANK/漢字の場合、座標(0,0)に”1234567890ABCDEFGH あ”表示



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)

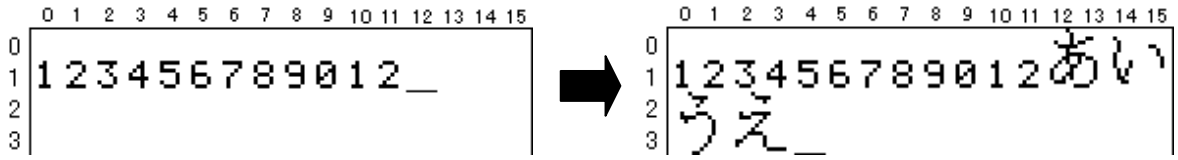
(例1)先頭文字が縮小 ANK の場合、座標(12,1)に“A あいうえ”を表示



※ い’が入りきらないので改行します。

※ 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。

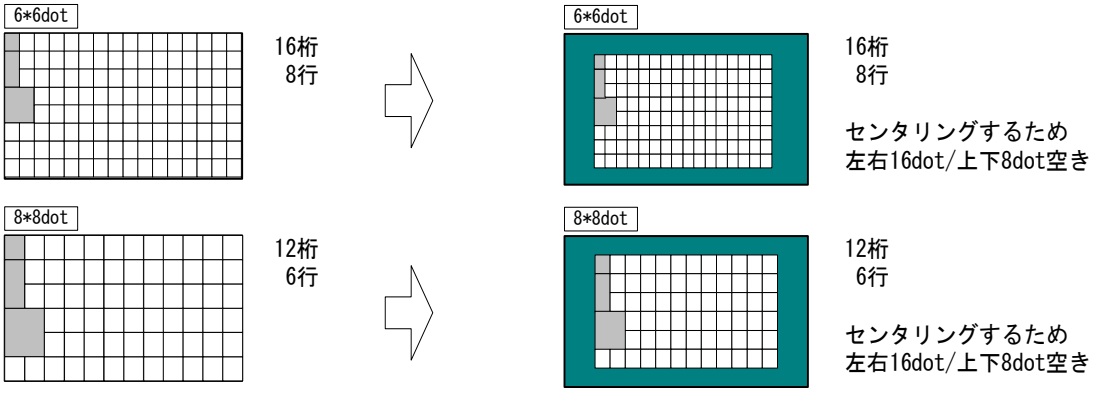
11.6.7 DT-700 互換表示

本表示関数では、DT-700 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。
リンク時に互換モード用のオブジェクトをリンクしてください。(AP_START.OBJ は、リンクしません)

互換モード A

- 128×64dot 内の中央 96×48dot のみを使用して表示します。
- 表示文字が小さくなるのみで、互換表示が可能です。

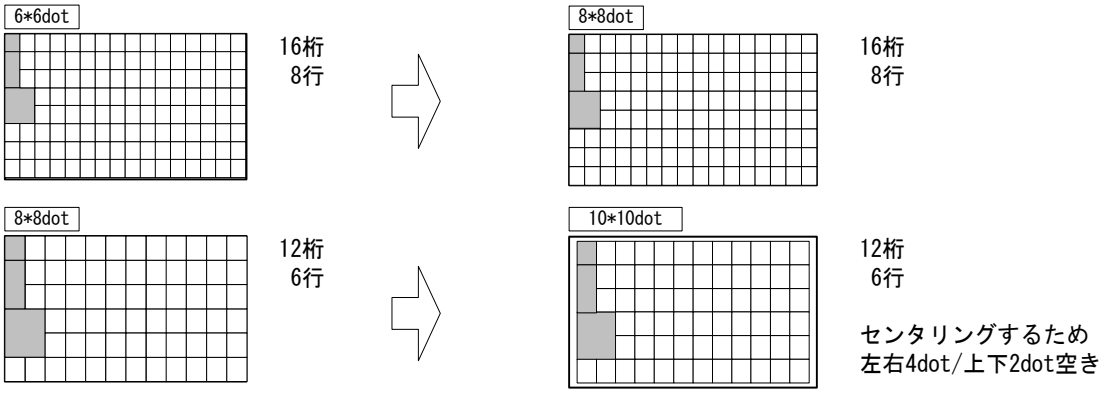
※ ユーザーフォント/外字フォント:ビット並びの変換 TOOL を提供します(PC 側で変換が必要)。



互換モード B

- 6dot 系→8dot 系/8dot 系→10dot 系に内部で自動的に切替えて表示します。
- 表示文字は最適なサイズで表示されますが、次の制約および注意が必要です。

※ ユーザー/外字フォント:フォントサイズ/ビット並びの変換ツールを提供します PC 側で変換が必要)。
※ また、サイズ変換ツールでは最適な文字イメージにはならない場合があります。
※ 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)



モード時の表示行桁数

モード	フォント(サイズ)		表示桁数	最大表示文字数
6ドット	縮小 ANK	(6× 6)	16 桁×8 行	128 文字
	標準 ANK	(6×12)	16 桁×4 行	64 文字
	漢字	(12×12)	8 桁×4 行	32 文字
8ドット	縮小 ANK	(8× 8)	12 桁×6 行	72 文字
	標準 ANK	(8×16)	12 桁×3 行	36 文字
	漢字	(16×16)	6 桁×3 行	18 文字

※ 互換モード設定時、10ドットフォントが設定されている場合は強制的に 8ドットフォントになります。

11.7 関数リファレンス

関数	機能概要
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	外字フォントの登録
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

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

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　　:0～9 行
8ドットモード　　:0～7 行
10ドットモード　　:0～5 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

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。)
引数の文字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。改行コード(CR・LF)は通常の改行処理を行ないます。また、最下行にある場合はスクロールを行ないます。

```
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。

disp_attr
表示属性を指定します。

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。

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替ります。

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。

disp_attr
表示属性を指定します。

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。

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 0～63dot 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 0～63dot 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

lcd_gaiji 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_K6_FILE	:漢字 6 ドットフォント
LCD_K8_FILE	:漢字 8 ドットフォント
LCD_AS6_FILE	:標準 ANK6 ドットフォント
LCD_AS8_FILE	:標準 ANK8 ドットフォント
LCD_AL6_FILE	:縮小 ANK6 ドットフォント
LCD_AL8_FILE	:縮小 ANK8 ドットフォント
LCD_K10_FILE	:漢字 10 ドットフォント
LCD_AS10_FILE	:標準 ANK10 ドットフォント
LCD_AL10_FILE	:縮小 ANK10 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
LCD_EL_HIGH　　:EL 点灯(高輝度)

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

12.キー制御

DT-930 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-930 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 7		キーを押すたびに“A”→”B”→”C”→”7”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)	00h	1Dh	カーソルを 1 文字左へ移動します。	123	123
					1234567890 1234	1234567890 1234
					1	1
→	→(F3)	00h	1Ch	カーソルを 1 文字右へ移動します。	123	123
					1234567890 1234	1234567890 1234
					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	12
					1234567890 abcd	123456789a bcd
					12345	2345
削除	DEL(F4)	00h	10h	カーソル上の 1 文字を削除します。	123	13
					1234567890	123456789
					1234567890 abcd	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	バーコード読込開始※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			
その他(ANK)	00h	XXh	文字※2	○	○	数字(0～9) +,-,.のみ

※ バーコード読込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。

※ 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	－
	後退(BS)			
	クリア(CLR)			
	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		
	F1(-)	F2(←)	可能	バーコード読込を除くすべて
	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	右トリガーキー		不可	－
	左トリガーキー			

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	右トリガーキー		不可
	左トリガーキー		

※ 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)

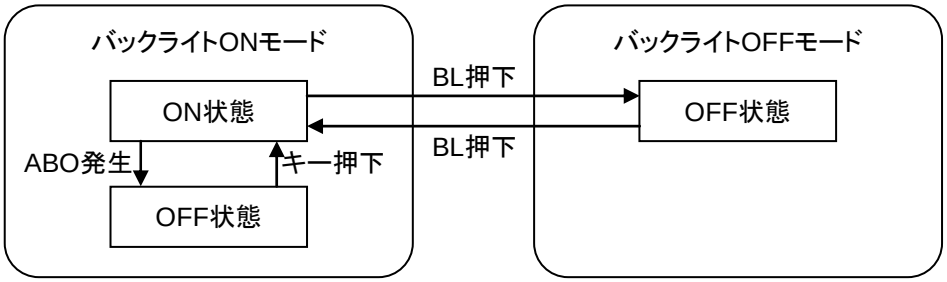
12.4 キーバッファ

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-930 では 128 キー固定としています。
電源 OFF→ON(レジューム立上げ)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-930 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、バーコード読込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、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 キーどうしの同時押し
  バーコード読込が開始されます。ただし、OBR オープン中のみ OBR が未オープン時は、無視されます。
- 通常キーどうしの同時押し
  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込が開始されます。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	バーコード読込開始	
OBR キーと通常キー	バーコード読込開始	
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込を中止します。
- 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
  バーコード読込を開始します。

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	バーコード読込継続
OBR キー	通常キー	無視(OBR キーリリース後確定)	バーコード読込継続
通常キー	OBR キー	バーコード読込開始	
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	:バーコード読込完了検出
E_KEY_IO	:IO ボックス検出
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	:バーコード読込完了検出
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	:IO ボックス検出
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	:バーコード読込完了検出
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	:IO ボックス検出
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出します。キーバッファの読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-930 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
読取り角度	40deg

読取り可能コード

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-B,UPC-E UPC-A addon(+2,+5),UPC-B addon(+2,+5) UPC-E addon(+2,+5)
CODE-39		
NW-7 (CODABAR)		
2of5 (Interleaved/Industrial)		
CODE-93		
CODE-128		
MSI		
IATA		
RSS-14		
RSS Limited		
RSS Expanded		
RSS-14 Stacked		
RSS Expanded Stacked		

読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 を除きチェックデジット(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-B	12	0SMMMMMNNNNNN」	
	UPC-A addon+2	14	0SMMMMMNNNNNCAA」	
	UPC-B addon+2	14	0SMMMMMNNNNNNAA」	
	UPC-A addon+5	17	0SMMMMMNNNNNCAAAAA」	
	UPC-B addon+5	17	0SMMMMMNNNNNNAAAAA」	
	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

バーコード	規格	桁数	出力フォーマット	備考
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) 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
		3～50*	SAAA・・・・・AACS」	
		1～48*	BBB・・・・・・BBC」	
		1～48*	AAA・・・・・・AAC」	
NW-7		3～40	SDDD・・・・・DDDE」	S: スタートコード (A,B,C,D のいづれか) E: エンドコード (A,B,C,D のいづれか) 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		1～40	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		1～40	DDD・・・・・・DDCC」	D: データ C: チェックデジット (mod10,mod11) チェックデジットなしの場合、 データとなります。
IATA		1～40	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

バーコード	規格	桁数	出力フォーマット	備考

RSS-14 Stacked		16 14	01DDDDDDDDDDDDC」 DDDDDDDDDDDDC」	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

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

開始処理は読取りコードの設定を行なうことも可能です。読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 文字の読込み
OBR_getc 関数を使用して、OBR バッファから 1 文字を読出します。

文字列の読込み
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、読取りデータをキー入力と同時にあつかうことができます。
初期状態は、OBR バッファを設定しています。

※ 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。

読取り可能コード種別

特定のコードしか読取る必要がない場合は、必要なコードのみを読取り対象に設定します。
必要なコードのみを読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 Interleaved 2of5 CODE-93 CODE-128 MSI IATA 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 を読取る場合は、読取可能コードに 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。

読取り桁数

読み取り対象のコードごとに、読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	1～40 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	1～40 桁	同左	
IATA	1～40 桁	同左	チェックデジット有の場合は、2～40
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	

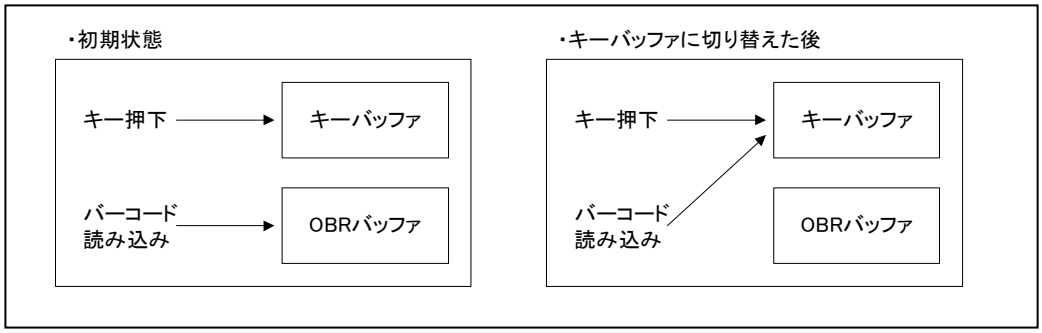
※ 複数のコードを同時に設定した場合、CODE-39、 NW-7、 Interleaved 2of5 は、誤読防止のため桁数の設定可能範囲が変化します。(カッコつきの範囲)

※ 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 桁を読取る場合は、コード限定の読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 OCR バッファとキーバッファに切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、読取りデータをキー入力と同時にあつかうことができます。
初期状態は、OBR バッファを設定しています。



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ます。
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。

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。

- 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

読取り方法

バーコードの読取り方法を、“単発読み”/“連続読み”から選択することができます。

読取り方法	内容	読取り終了条件
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、読取り完了後に待機状態となります。	スキャン時間経過 読取り完了
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、常に読取り可能状態となります。	前コード読取り完了後にスキャン時間経過 指定読みと理解数分の読取り完了 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、dat_system 関数を使用して設定します。

読取り回数

連続読みの場合の読取り可能回数を設定できます。
設定した回数分読取りを完了すると、自動的に読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。
読取り回数は、dat_system 関数を使用して設定します。

照合回数
読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、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。
初期値状態は“有効”です。

同一ラベルの二度読み防止
読取り回数を2回以上に設定し、連続読みにて読取りを行なっている場合、1回の読取り中（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”環境設定メニュー”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。
- 読取りコードが異常な場合、LED を一定秒間赤色に点灯したのち消灯します。

LED 制御を無効にすることも可能です。

読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。
- 読取りコードが異常な場合、バイブレータを振動させません。

読取り動作

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取りが行なわれます。

立上げモード

ここでいう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

※ 注意事項

- バーコード読取りを行なっている最中に、動作モード設定による誤動作を防止するために、オープン中の動作モード設定は無効となります。
- 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しますが引き続きパラメータ設定の処理を行ないます。また、パラメータ中にエラーがあった場合、パラメータエラーを返します。

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、読取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-930 では、読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。

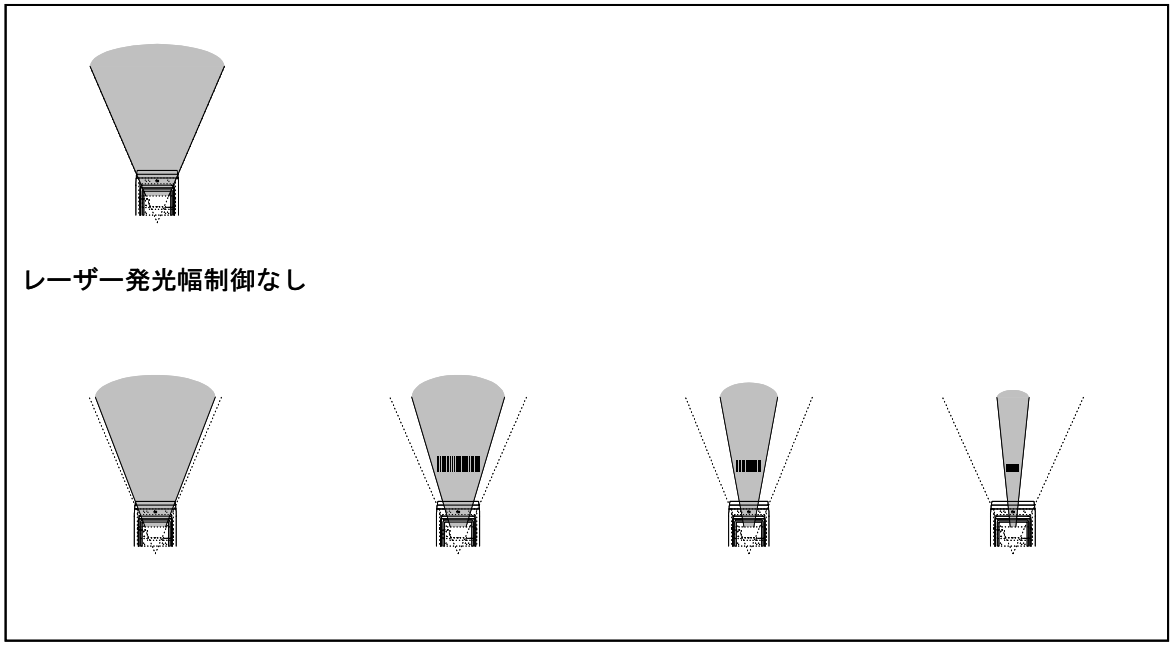
レーザー発光幅制御の機能を以下に示します。

機能	内容
キャリブレーション	DT-930 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。
レーザー発光幅微調整	レーザー発光幅を微調整します。

キャリブレーション （ wkup_calib 関数 ）
DT-930 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"に設定されます。

レーザー発光幅制御量	レーザー発光幅
間口幅	40°
幅広	32°
幅中	24°
幅狭	16°

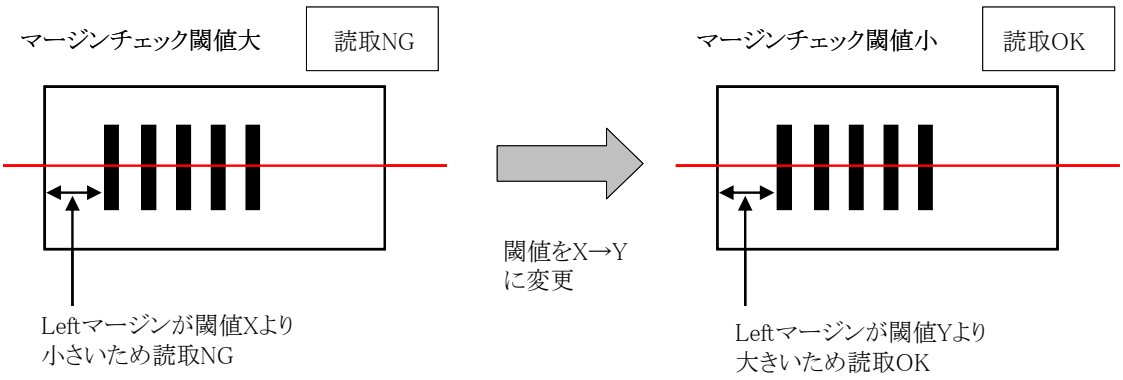
レーザー発光幅微調整機能 （ OBR_widenarrow 関数 ）
レーザー発光幅を微調整します
レーザー発光幅を±5 段階で調整することができます。
微調整の値は、DT-930 をリセットした時点でクリアします。

13.2.3 カスタマイズ機能

世の中に存在するすべてのバーコードを確実に読むことができるデコード制御を FIX させることは事実上不可能です。実際、読取環境や、印刷物の状態等に応じて、個別対策を実施しています。

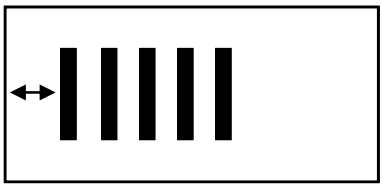
本機能は、この個別対策による読取性能の向上を効率よく、実施するための機能です。
通常のデコードロジックの読取り性能を保持するために、まずは通常のデコーダによりデコードを行ない、デコードできなかった場合はカスタマイズしたデコーダによりデコードを行ないます

Right/Left マージンの閾値変更 （ OBR_getmargincheck 関数、OBR_setmargincheck 関数 ）
バーコードが枠に囲まれて印刷されている場合、Right マージンまたは Left マージンが十分に確保されておらず、読取ができない場合があります。このような場合、Right/Left マージンの閾値の変更を行なうことにより読取りが行なえるようになります。

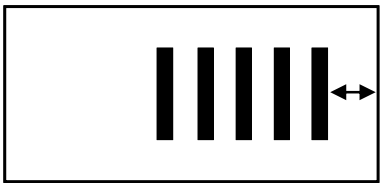


マージンチェック閾値を変更すると、次のようなバーコードの読取が行なえるようになります。

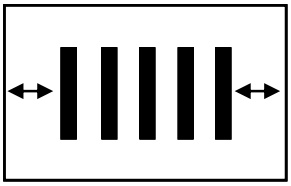
Leftマージンが狭いバーコード



Rightマージンが狭いバーコード

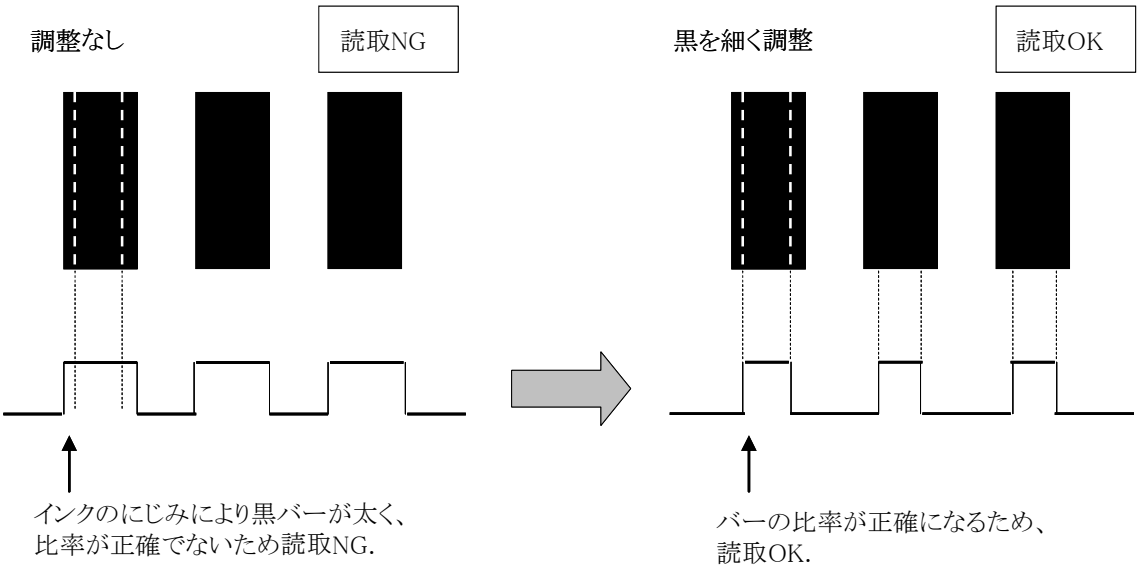


Rightマージン・Leftマージンの
両方が狭いバーコード



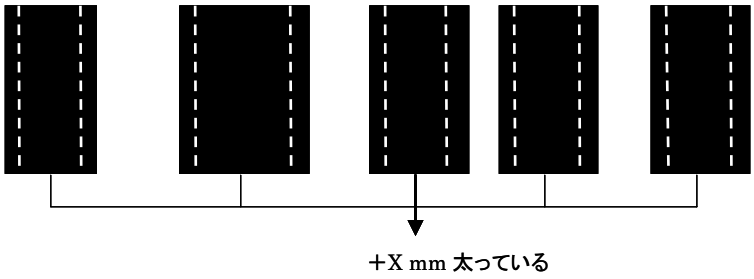
バーの太り・細り補正値変更 （ OBR_getadjust 関数、OBR_setmargincheck 関数 ）

バーコードの印刷において、インクがにじんで黒バーが太く印刷されたり、かすれて白バーが細く印刷されたりするような場合、バーの比率が正確でなくなってしまうため読取できない場合があります。このような場合、バーの太り・細りの調整を行なうことにより読取が行なえるようになります。この調整は、すべての黒バーまたは、白バーに対して同じ幅ずつ細らせて行なうため、全体が同等に太っているバーコードまたは、全体が同等に細っているバーコードに対して有効となります。

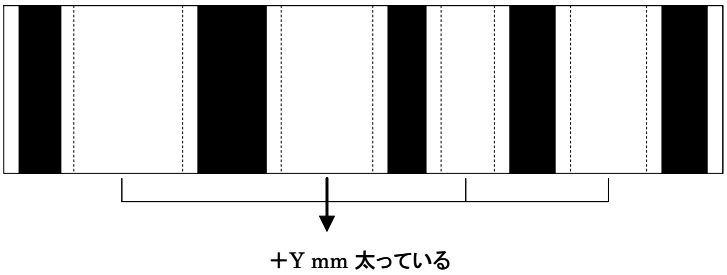


太り・細り補正値を変更すると、次のようなバーコードの読取が行なえるようになります。

黒バーが全体的に同等に太ったバーコード



白バーが全体的に同等に太ったバーコード



読取りレベル設定

CODE39･NW7 の読取りレベルを設定することができます。

NW-7、CODE39 の読取りレベルの設定は、設定ファイル『CONFIG.OBR』にて行なうようにします。下記の書式のファイルを DT-930 の Aドライブもしくは Bドライブのルートディレクトリに格納してください。アプリケーション起動時に設定が反映されます。

CONFIG.OBR の書式

```
; DT-900 CONFIG.OBR
; Copyright(C) 2000 CASIO COMPUTER CO.,LTD. All rights reserved.

NW7LEVEL=n        ;  NW-7 読取りレベル
                  ;  n = 0(甘い)～3(厳しい)
                  ;  省略時は 2 となります

CODE39LEVEL=n     ;  CODE-39 読取りレベル
                  ;  n = 0(甘い)～3(厳しい)
                  ;  省略時は 2 となります
```

※ コメント以外は半角英数字を指定してください。
※ 大文字、小文字の区別はありません。
※ Aドライブおよび Bドライブのルートディレクトリに CONFIG.OBR がない場合は、デフォルト値(NW-7、CODE39 ともにレベル 2)が設定されます。
※ AドライブとBドライブの両方にCONFIG.OBRがある場合は、Aドライブの同ファイルが有効となります。
※ 無効な設定値が指定された場合は、デフォルト値が設定されます。
※ スペース、TAB、空白行は無視されます。
※ セミコロン";"の後はコメントとなります。
※ 同じ指定が 2 箇所以上ある場合は、後に指定されたものが有効になります。
※ CONFIG.OBR のファイルサイズは 512 バイト以下にしてください。

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
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

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	:Industrial 2of5
OBR_ITF	:Interleaved 2of5
OBR_CD93	:CODE93
OBR_CD128	:CODE128
OBR_MSI	:MSI
OBR_IATA	:IATA
OBR_RSS14	:RSS-14
OBR_RSSLTD	:RSS Limited
OBR_RSSEXP	:RSS Expanded
OBR_RSS14S	:RSS-14 Stacked
OBR_RSSEXPS	: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、動作モードは初期化されます。
(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値になります)
※ 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。
読取り回数、照合回数、読取り時間は、システムデータ管理関数で設定してください。

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(void);
```

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 文字読込ます。

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

パラメータ

rcd
読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
OBR_RSSLTD	:RSS Limited
OBR_RSSEXP	:RSS Expanded
OBR_RSS14S	:RSS-14 Stacked
OBR_RSSEXPS	: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
BR_MSI	:MSI
OBR_IATA	:IATA
OBR_RSS14	:RSS-14
OBR_RSSLTD	:RSS Limited
OBR_RSSEXP	:RSS Expanded
OBR_RSS14S	:RSS-14 Stacked
OBR_RSSEXPS	: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

13.3.8 OBR_modewt

OBR の動作モードを設定します。
読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

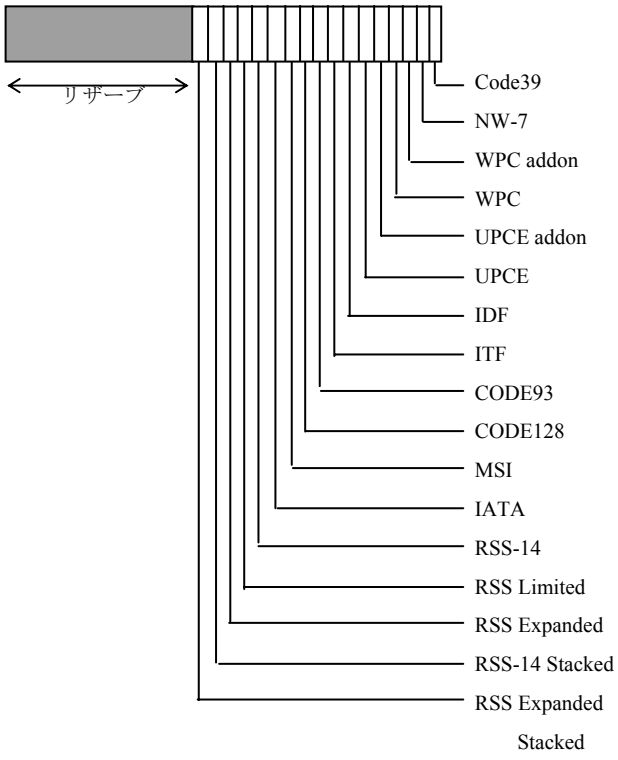
戻り値

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 のパラメータに"0"を指定してください。
OBR_open("0");

動作モードテーブル

項目	内容	初期値	設定	参照
読取り コード (4 バイト)	b31 b0  ＊1 読取るコードに該当するビットをＯＮ（1）にします。 ＊2 読取り性能を向上させるため、必要な読取りコードのみ設定することを推奨します。	すべて選択 (FFFh)	○	○

項目	内容	初期値	参照	設定
読取り桁数の設定 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*7	1	–	WPC(UPCE以外)addon
FFh	8	13	0*8	1	–	WPC(UPCE以外)
FFh	9	12	0*9	1	1	UPCE addon
FFh	7	7	1*3	1	0	UPCE
FFh	2	40	–	1	1	Industrial 2of5
FFh	4	40	–	1	1	ITF
FFh	3	40	–	1	–	CODE93
FFh	2	64	0*4	1	–	CODE128
FFh	1	40	–	1*5	1	MSI
FFh	1	40	–	0*6	–	IATA
FFh	14	14	0*10	1	1	RSS-14
FFh	14	14	0*11	1	1	RSS Limited
FFh	1	74	–	1	1	RSS Expanded
FFh	14	14	0*12	1	1	RSS-14 Stacked
FFh	1	74	–	1	1	RSS Expanded Stacked
↓ ﾘｻﾞｰﾌﾞ						
FFh	–	–	–	–	–	

– 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)

ﾁｪｯｸﾃﾞｼﾞｯﾄ 0:ﾁｪｯｸ無し 1:ﾁｪｯｸ有り
ﾁｪｯｸｷｬﾗｸﾀ 0:出力無し 1:出力有り
*1 CODE39出力ﾌｫｰﾏｯﾄ 0:Start/Stopｺｰﾄﾞ有り
1:Start/Stopｺｰﾄﾞ無し
2:Full ASCII Start/Stopｺｰﾄﾞ有り
3:Full ASCII Start/Stopｺｰﾄﾞ無し
*2 NW-7出力ﾌｫｰﾏｯﾄ 0:Start/Stopｺｰﾄﾞ有り
1:Start/Stopｺｰﾄﾞ無し
*3 UPCE出力ﾌｫｰﾏｯﾄ 0:UPCAへの復元ｺｰﾄﾞ出力有り
1:UPCAへの復元ｺｰﾄﾞ出力無し
2:GTINﾌｫｰﾏｯﾄで出力
3:UPCAへの復元ｺｰﾄﾞ出力無し
4:先頭の0を削除して出力
*4 CODE128出力ﾌｫｰﾏｯﾄ 0:変換後(ASCII)のデータを出力
1:変換前のデータを出力
2:CODE128(×)EAN128(変換後)出力
3:CODE128(×)EAN128(変換前)出力
4:CODE128/EAN128(変換後+認識ｺｰﾄﾞ)出力
6:CODE128(×)EAN128(変換後+認識ｺｰﾄﾞ)出力
8:CODE128(変換後)EAN128(GS変換)出力
A:CODE128(×)EAN128(GS変換)出力
10:CODE128(変換前)EAN128(変換後)出力
20:CODE128(変換後)EAN128(変換前)出力
変換前/後の対応は、本仕様書最終頁参照
*5 MSIﾁｪｯｸﾃﾞｼﾞｯﾄ 0:ﾁｪｯｸﾃﾞｼﾞｯﾄ無し
1:1桁,mod10
2:2桁,1st:mod11 2nd:mod10
3:2桁,1st:mod10 2nd:mod10
*6 IATAﾁｪｯｸﾃﾞｼﾞｯﾄ 0:ﾁｪｯｸﾃﾞｼﾞｯﾄ無し(読取り桁数1-40)
1:最終ｷｬﾗｸﾀ以外を対象(読取り桁数2-40)
2:ｸｰﾎﾟﾝ番号/数値部を対象(読取り桁数15-17)
3:数値部を対象(読取り桁数15-17)
*7 WPCaddon出力ﾌｫｰﾏｯﾄ 0:UPCAの先頭に0を付加して出力
1:UPCAの先頭に0を削除して出力
*8 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
*9 UPCEaddon出力ﾌｫｰﾏｯﾄ 0:先頭の0を付加して出力
4:先頭の0を削除して出力
*10 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
*11 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
*12 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

項目	内容	初期値	参照	設定
読取り方式の設定 (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)	○	○

※1 設定は OBR_chgbuf 関数を使用してください。

動作モードテーブル内で、記載されていない数値を設定した場合、パラメータエラーにします。

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

4 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_SET0 :間口幅
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"間口幅"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。

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

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

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

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
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

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.シリアル通信制御

管理

関数	機能概要
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

送受信

関数	機能概要
c_dout	n 文字の送信
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_er	ER 信号の制御
c_rs	RS 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

受信バッファ管理

関数	機能概要
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

通信補助

関数	機能概要
c_errbfring	エラー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	受信ハンドラの切替え
c_brkevent	ブレイク要因の設定

14.1 通信仕様

14.1.1 通信インタフェース

通信ポート

DT-930 には赤外線通信ポートが存在します。通信関数部では、赤外線ポートを使用した、カシオオリジナル IR インタフェースに対する機能を提供します。
IrDA 部では IrDA ポートに対する機能を提供します。

通信ポート	制御形式	コネクタ	COM No	規格	同期方式	転送速度 (bps)	キャラクタレングス	パリティビット	ストップビット	信号
カシオ IR インタフェース	半二重	IrDA	COM0	カシオオリジナル	調歩	2.4k 9.6k 19.2 38.4k 57.6k 115.2k	7bit 8bit	NON ODD EVN	1bit 2bit	SD RD CTRL (※1)
IrDA	半二重	IrDA	-	IrDA (IrSIR 1.2)	調歩フレーム	2.4k 9.6k 19.2 38.4k 57.6k 115.2k 4M	8bit	NON	1bit	SD RD

※ ベーシック IO ボックス制御信号

排他制御

DT-930 のカシオ IR インタフェース、および IrDA ポートは同時に使用することはできません。

排他相互関係	カシオ IR インタフェース	IrDA
カシオ IR インタフェース		同時不可
IrDA	同時不可	

カシオオリジナル IR インタフェース

カシオオリジナル IR インタフェースは、SD/RD の信号線のみを使用した通信ポートです。シリアルインタフェースの仕様に合わせて、仮想的な信号線制御を行なっています。(詳細は、「14.2.7 信号線制御とタイムアウト監視 (p.248)」を参照してください。

14.2 機能

14.2.1 通信ポートのオープン・クローズ・占有

(1) オープン

DT-930 の通信関数部を使用してデータ通信を行なうためには、通信ポートのオープンを行なう必要があります。
通信ポートをオープンすることで通信ポートに電源供給し、通信リソースの占有を行ない、通信関数部が提供するさまざまなデータ通信機能を用いてデータ通信を行なうことを可能にします。
通信ポートのオープンは"COM のオープン"ファンクションで行ないます。

(2) クローズ

DT-930 の通信関数部を使用してのデータ通信が終了したときは、通信ポートのクローズを行なう必要があります。
通信ポートをクローズすることで通信ポートの電源供給を停止し、通信リソースの解放を行ない、通信関数が提供するデータ通信機能を用いてデータ通信を行なうことを不能にします。
DT-930 は通信ポートのオープン中の間だけ通信ポートへの電源供給を行ない、また通信リソースを占有しています。
通信が終了したときには速やかに通信ポートのクローズを行なってください。
通信ポートのクローズは"COM のクローズ"ファンクションで行ないます。

(3) 占有

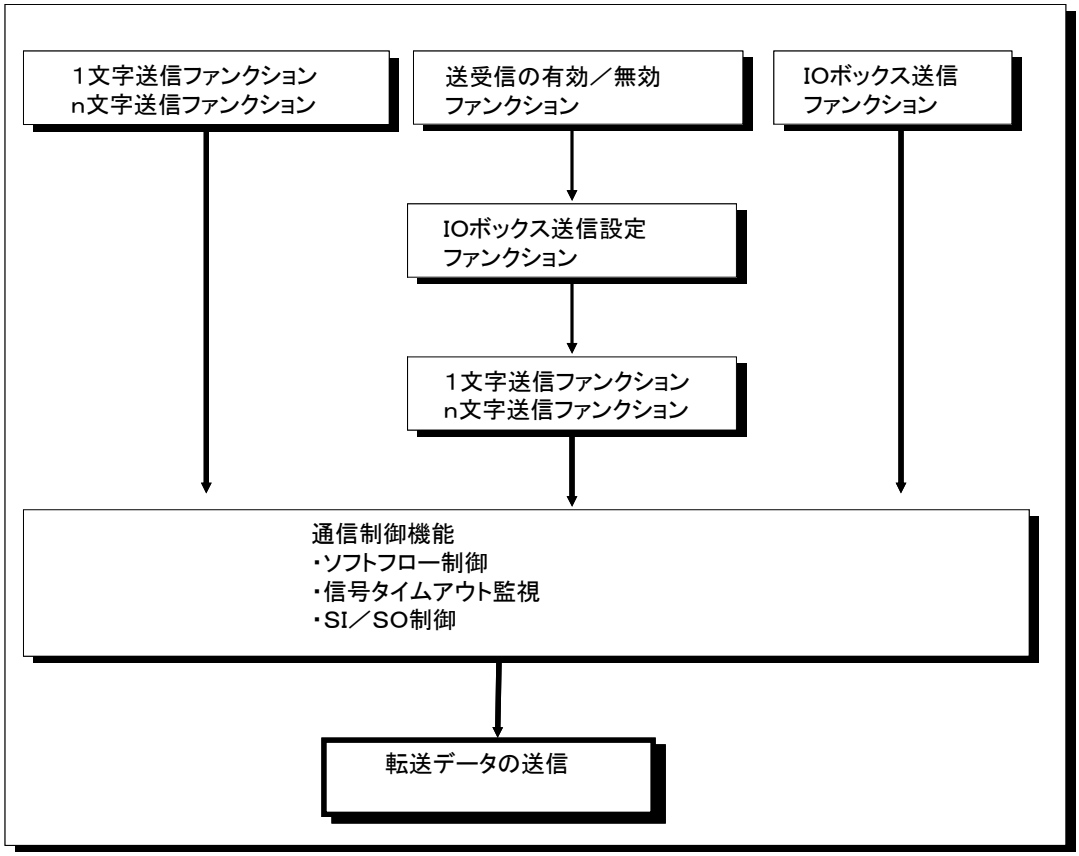
通信ポートのオープンを行なうと、通信ポートに電源供給を行ない、通信リソースの占有を行ないます。これを"ポートオープン"状態とします。ポートオープン状態であるとき、同一(排他関係)の通信ポートをオープンすることはできませんし、DT-930 の通信関数部では通信ポートを"ポート占有状態"にすることができます。ポート占有状態であるとき通信ポートへの電源供給、通信リソースの占有は行なわず、同一(排他関係)の通信ポートのオープンの抑制のみを行ないます。
例えば、通信関数部以外で通信部が使用する通信リソースを使用してデータ通信を行なう場合に通信ポートを占有状態にして、通信部が使用できないようにします。
通信ポートの占有は"COM の占有"ファンクションで行ないます。

14.2.2　転送データの送信・受信

(1) 送信

通信ポートをオープンすることで転送データの送信を行なうことができます。
送信する転送データは 1 文字、n 文字単位であつかうことができ、SI/SO 制御、フロー制御の通信制御機能を使用することができます。またカシオ IR ポート用の半二重制御による転送データの送信を行なうことができます。
転送データの送信は"n 文字送信"、"1 文字送信"ファンクションで行ないます。またカシオ IR ポートを使用する場合はこれらのファンクションの他に"IO ボックス送信"、"IO ボックス送信設定"、"送受信の有効/無効"ファンクションを使用して行ないます。

【転送データの送信の流れ】



(2) 受信

通信ポートをオープンすることで転送データの受信を行なうことができ、SI/SO 制御、フロー制御の通信制御機能を使用することができます。またカシオ IR ポート専用の半二重制御による転送データの受信および受信データの読込を行なうことができます。

1. 受信バッファの設定

転送データの受信を行なうためには転送データを受信して格納する領域と文字数(byte)を設定します。
通信関数部は設定された領域への転送データの格納、読出しを FIFO 形式(この領域を受信バッファと呼び、格納したデータを受信データと呼ぶ)で処理します。
転送データを受信したとき、この受信バッファに空きなければ受信バッファオーバーフローエラーとなります。
文字数を 0 に設定したとき通信部の内部領域を使用します。この場合はバッファフロー制御を行なうことはできません。
受信バッファの設定は"COM のオープン"ファンクションで行ないます。

2. 受信ハンドラ

転送データの受信は、割込みにより通信関数部の受信ハンドラが行ないます。
通信関数部には標準ハンドラと簡易ハンドラの 2 つの受信ハンドラ部を持ち、指定によりどちらかの受信ハンドラを使用します。
受信ハンドラの設定は"受信ハンドラ切替え"ファンクションで行ないます。

a) 標準ハンドラ

標準ハンドラは転送データの受信機能(受信データバッファリングと呼ぶ)の他に次の機能を持ちます。

- SI/SO 制御
- バッファフロー制御
- デリートコード制御
- エラーコードバッファリング制御

尚、DT-930 の初期化において標準ハンドラに設定します。

b) 簡易ハンドラ

簡易ハンドラは受信割込み処理を短縮することができます。簡易ハンドラは転送データの受信機能(受信データバッファリングと呼ぶ)のみを持ちます。

c) 受信ハンドラが制御する信号線

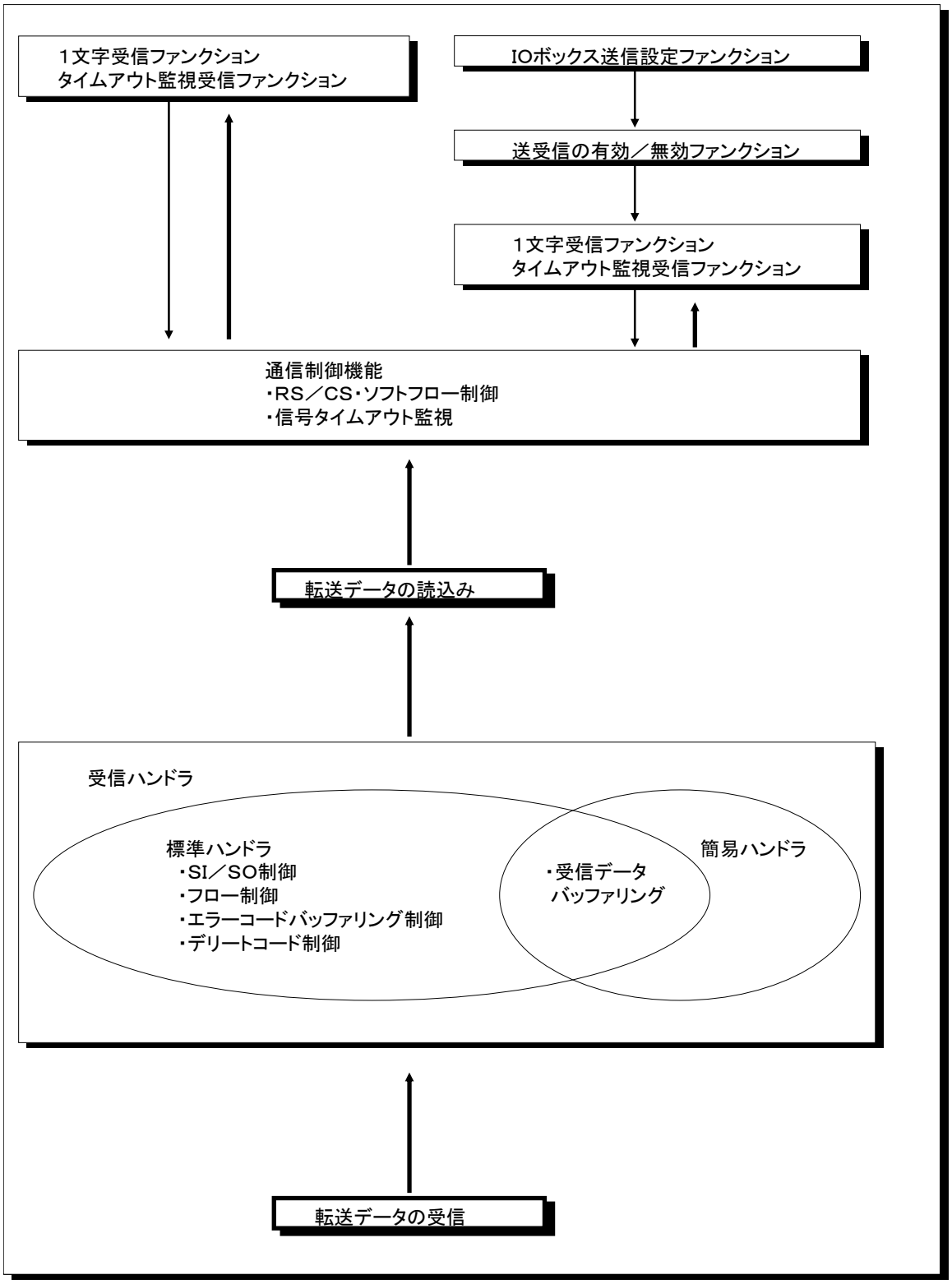
受信ハンドラでは次の信号線の操作および参照を行ないます。

信号線	制御
RS	標準ハンドラ使用時、バッファビジーになったとき OFF に設定

3. 受信データの読込

受信バッファに格納された転送データの読込を"1 文字受信"、"タイムアウト監視受信"で行なうことができます。
これらのファンクションでは読込可能な受信データが受信バッファに存在しないとき、受信データを待ちます。
また、カシオ IR ポート専用の半二重制御による転送データの受信および受信データの読込をこれらのファンクションの他に"IO ボックス送信設定"、"送受信の有効/無効"ファンクションを使用して行ないます。

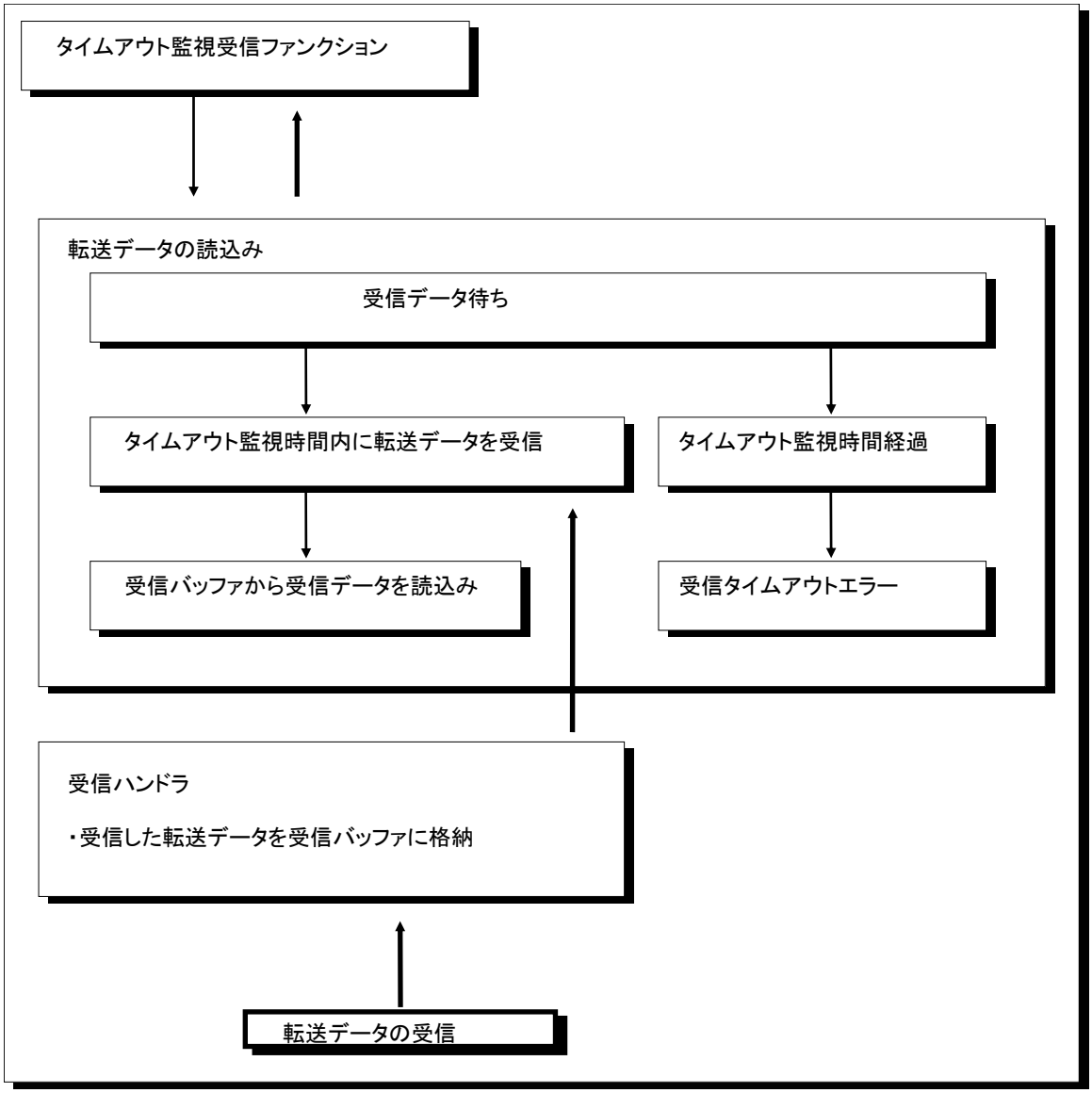
【転送データの受信および受信データの読込の流れ】



4. タイムアウト監視

読込可能な受信データが受信バッファに存在しないとき、受信データを待ちます。
“タイムアウト監視受信”ファンクションでは受信データ待ちにタイマを設定することができます。
受信データ待ちのままタイムアウト監視時間経過すると受信タイムアウトエラーになります。

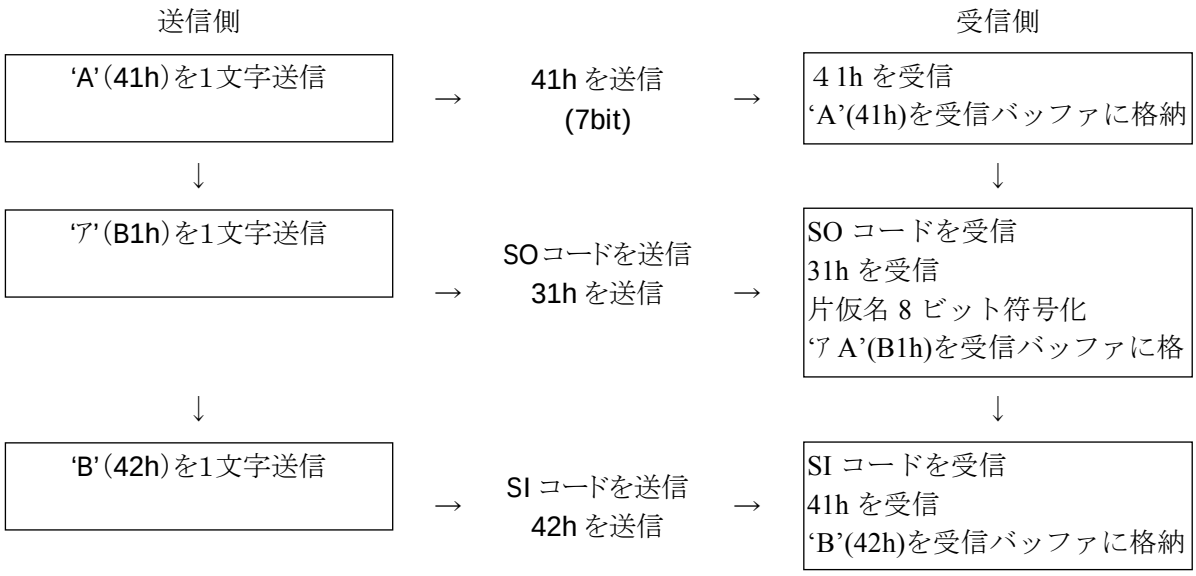
【タイムアウト監視の流れ】



14.2.3 SI/SO 制御

キャラクタレングスが 7 ビットであるときに JIS8 ビット符号化文字(JIS X 0201 参照)をあつかう場合に使用します。
JIS8 ビット符号化文字コードの片仮名 8 ビット符号(A0H～DFH)を送信するとき SO コードの送信を行ないます。
SO コードを受信したとき、その後に受信した転送データを片仮名 8 ビット符号に変換して受信バッファに格納します。
SO コード送信後に 9FH 次の文字コードを送信するとき、SI コードを送信してから文字コードの送信を行ないます。

【SI/SO 制御の流れ】



14.2.4 フロー制御

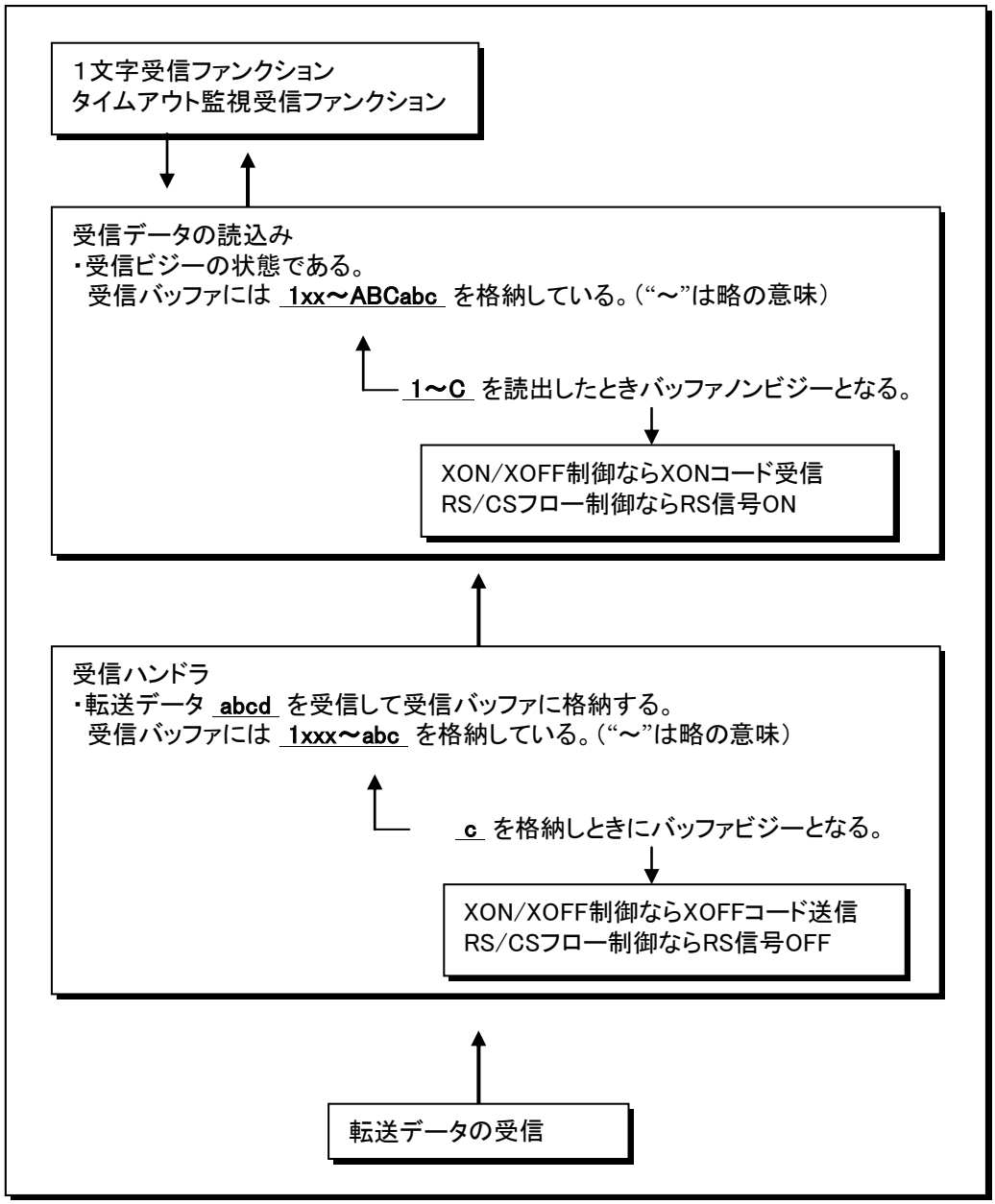
(1) XON/XOFF 制御

XON/XOFF 制御は受信バッファ全体の領域が 67 バイト以上であるとき機能します。
受信バッファの空きが 32 バイト以下になるとバッファビジーとなり、XOFF コードを送信して接続先からの転送データの送信を一時停止するように要求します。
受信バッファに格納されている受信データが 32 バイト未満になるとバッファノンビジーとなり、XON コードを送信して接続先からの転送データの送信再開を要求します。
接続先から XOFF コードが送られてきたとき接続先がバッファビジーとなり、転送データの送信を一時中止して XON コード受信待ちとなります。
接続先から XON コードが送られてきたとき接続先がバッファノンビジーとなり、転送データの送信を再開します。
本機能を使用する場合は"受信ハンドラの切替え"ファンクションで標準ハンドラに指定してください。

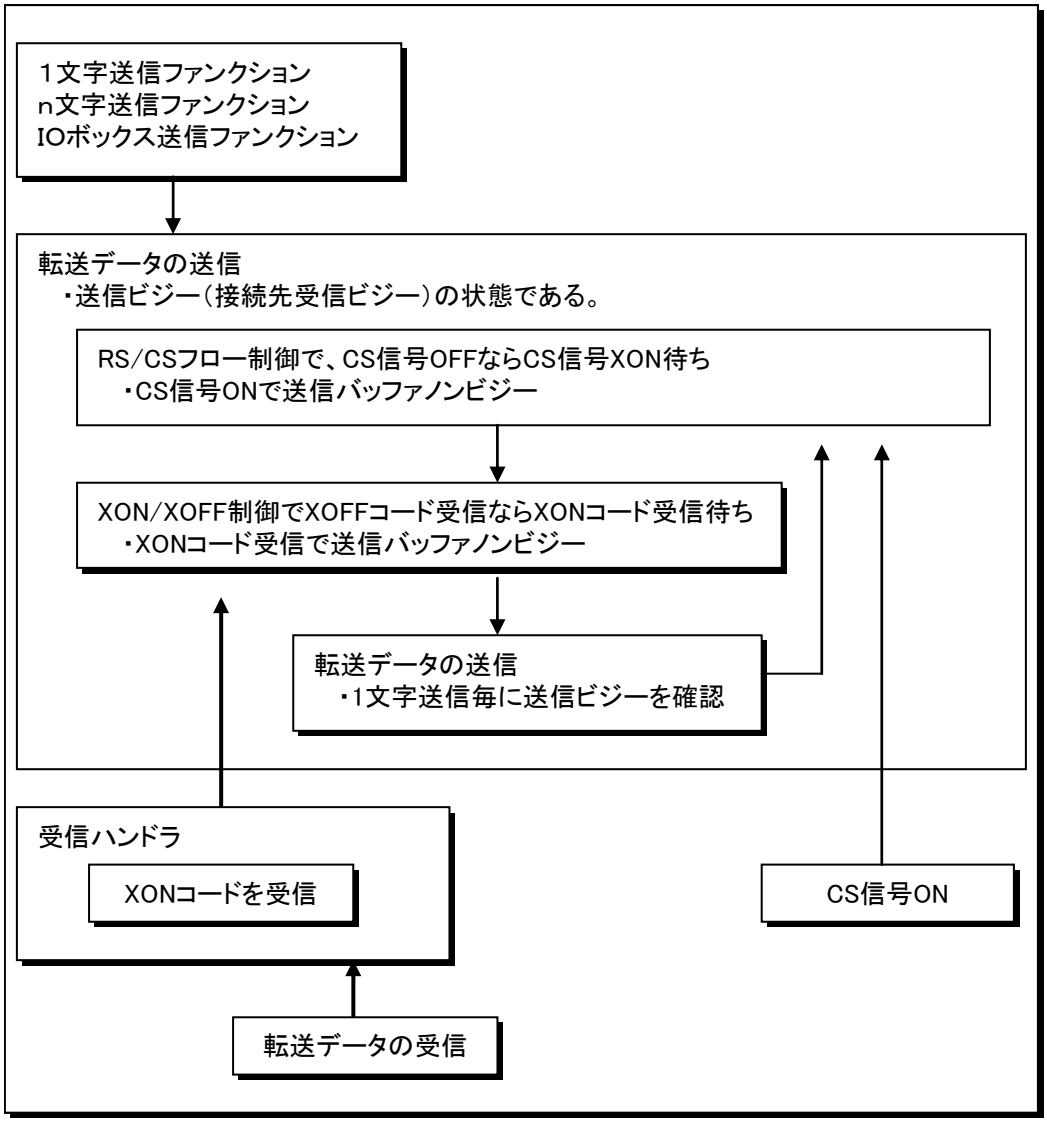
(2) RS/CS フロー制御

XON/XOFF 制御は受信バッファ全体の領域が 67 バイト以上であるとき機能します。
受信バッファの空きが 32 バイト以下になるとバッファビジーとなり、RS 信号を OFF にして接続先からの転送データの送信を一時停止するように要求します。
受信バッファに格納されている受信データが 32 バイト未満になるとバッファノンビジーとなり、RS 信号を ON にして接続先からの転送データの送信再開を要求します。
転送データの送信は CS 信号が ON であれば行ない、CS 信号が OFF であれば行なわず、ON 待ちとなります。
本機能を使用する場合は"受信ハンドラの切替え"ファンクションで標準ハンドラに指定してください。
また"COM のオープン"、"DR/CS/CD タイムアウト監視値の設定"ファンクションで CS 信号の監視を行なうように指定してください。

【受信フロー制御の流れ】


1文字受信ファンクション
タイムアウト監視受信ファンクション
受信データの読込み
・受信ビジーの状態である。
受信バッファには 1xx～ABCabc を格納している。(“～”は略の意味)
1～C を読出したときバッファノンビジーとなる。
XON/XOFF制御ならXONコード受信
RS/CSフロー制御ならRS信号ON
受信ハンドラ
・転送データ abcd を受信して受信バッファに格納する。
受信バッファには 1xxx～abc を格納している。(“～”は略の意味)
c を格納しときにバッファビジーとなる。
XON/XOFF制御ならXOFFコード送信
RS/CSフロー制御ならRS信号OFF
転送データの受信

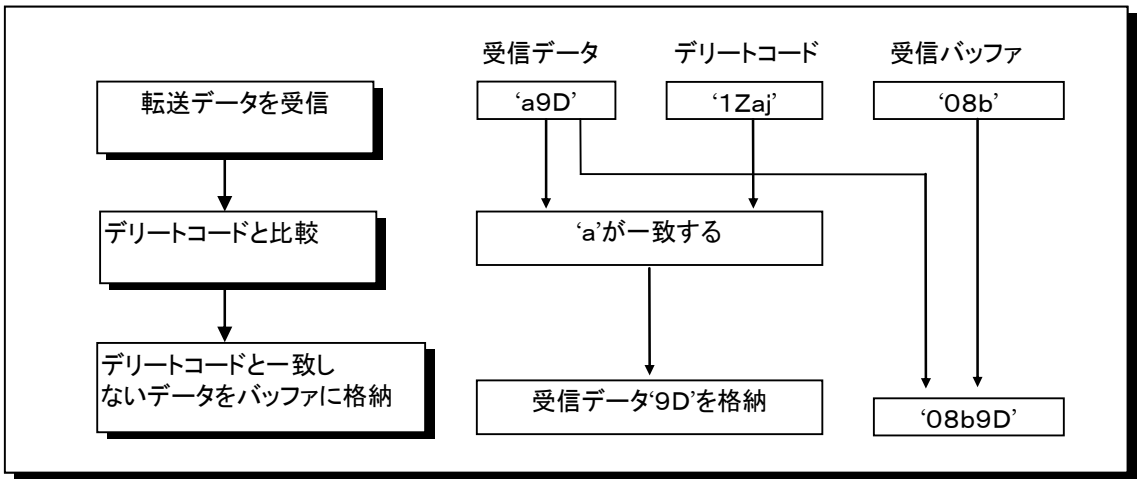
【送信フロー制御の流れ】


１文字送信ファンクション
n文字送信ファンクション
IOボックス送信ファンクション
転送データの送信
・送信ビジー（接続先受信ビジー）の状態である。
RS/CSフロー制御で、CS信号OFFならCS信号XON待ち
・CS信号ONで送信バッファノンビジー
XON/XOFF制御でXOFFコード受信ならXONコード受信待ち
・XONコード受信で送信バッファノンビジー
転送データの送信
・1文字送信毎に送信ビジーを確認
受信ハンドラ
XONコードを受信
CS信号ON
転送データの受信

14.2.5 デリートコード制御

デリートコードと受信文字コードが一致したとき、そのデータを破棄して受信バッファへの格納を行ないません。
デリートコードは 4 つまで指定できます。
デリートコード制御の設定は"COM のオープン"ファンクションで行ないます。

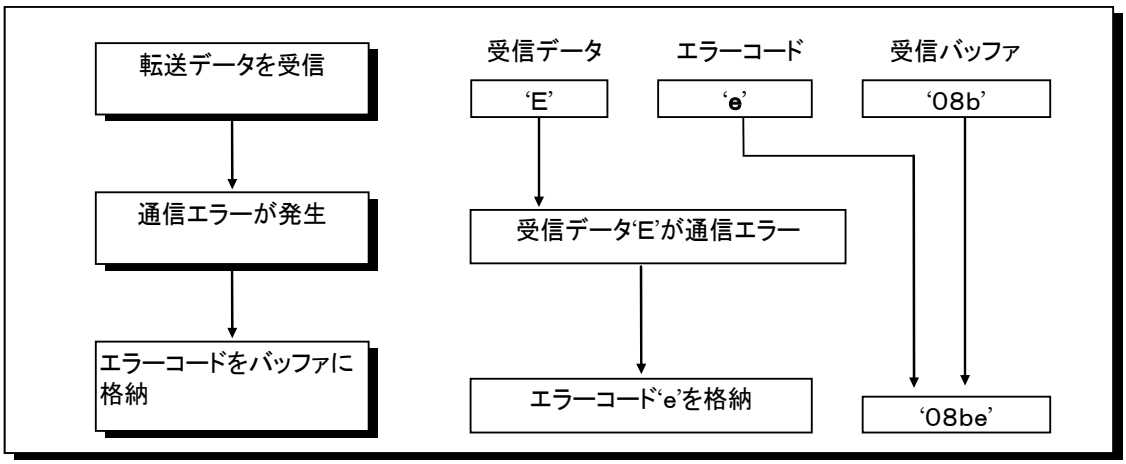
【デリードコードの制御の流れ】



14.2.6 エラーコードバッファリング制御

通信エラー(パリティ、オーバーラン、フレーミング)が発生したとき、指定のコードを受信バッファへ格納します。
通信エラーとなった受信データは受信バッファに格納しません。
エラーコードバッファリング制御の設定は"エラーコードバッファリング制御の設定"ファンクションで行ないます。

【エラーコードバッファリング制御の流れ】



14.2.7 信号線制御とタイムアウト監視

(1) 信号線

制御する信号線(SD、RD は除く)は通信ポートによって異なります。
また、カシオ IR ポートには仮想制御(物理的に信号線は存在しない)する信号線があります。

a) 通信ポートが制御する信号線

以下に各通信ポートが制御する信号線を示します。

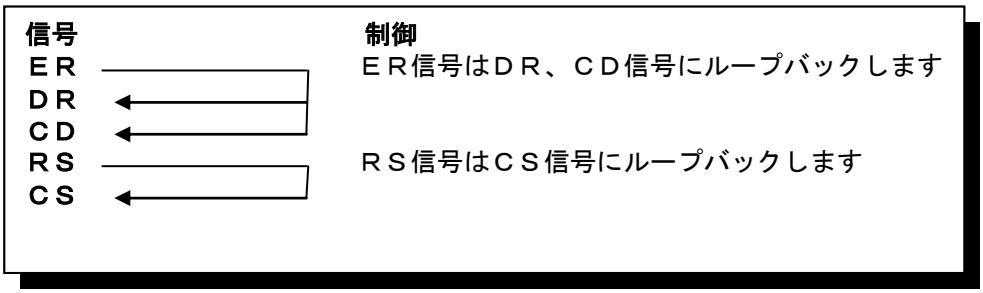
【通信ポートが制御する信号線】

通信ポート信号線	カシオ IR インタフェース
ER	あり(仮想)
RS	あり(仮想)
DR	あり(仮想)
CS	あり(仮想)
CD	あり(仮想)
CI	なし
CTRL	あり

b) カシオ IR ポートの仮想信号制御

以下にカシオ IR ポートの仮想信号制御方法を示します。

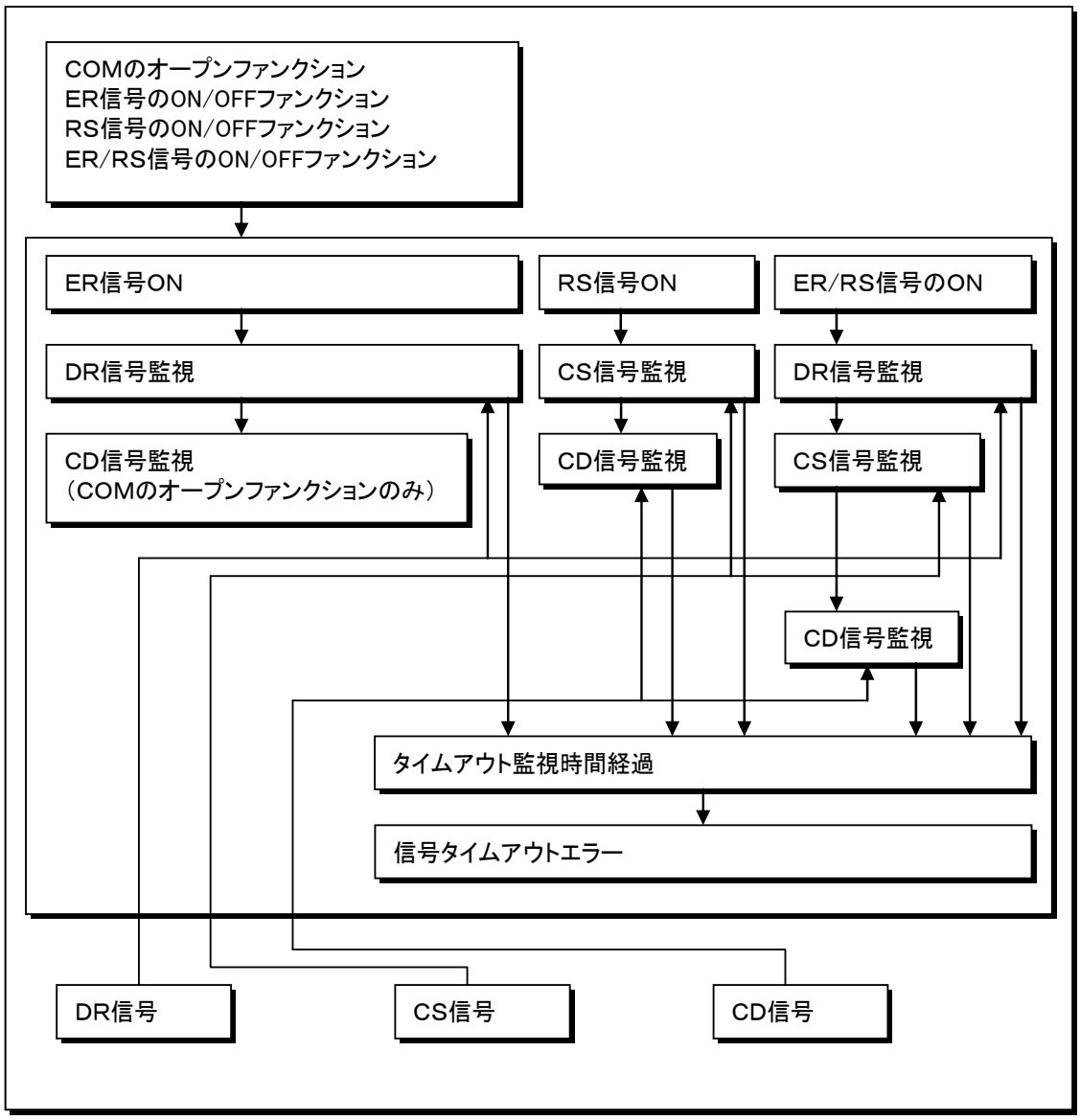
【カシオ IR ポートの仮想信号制御】



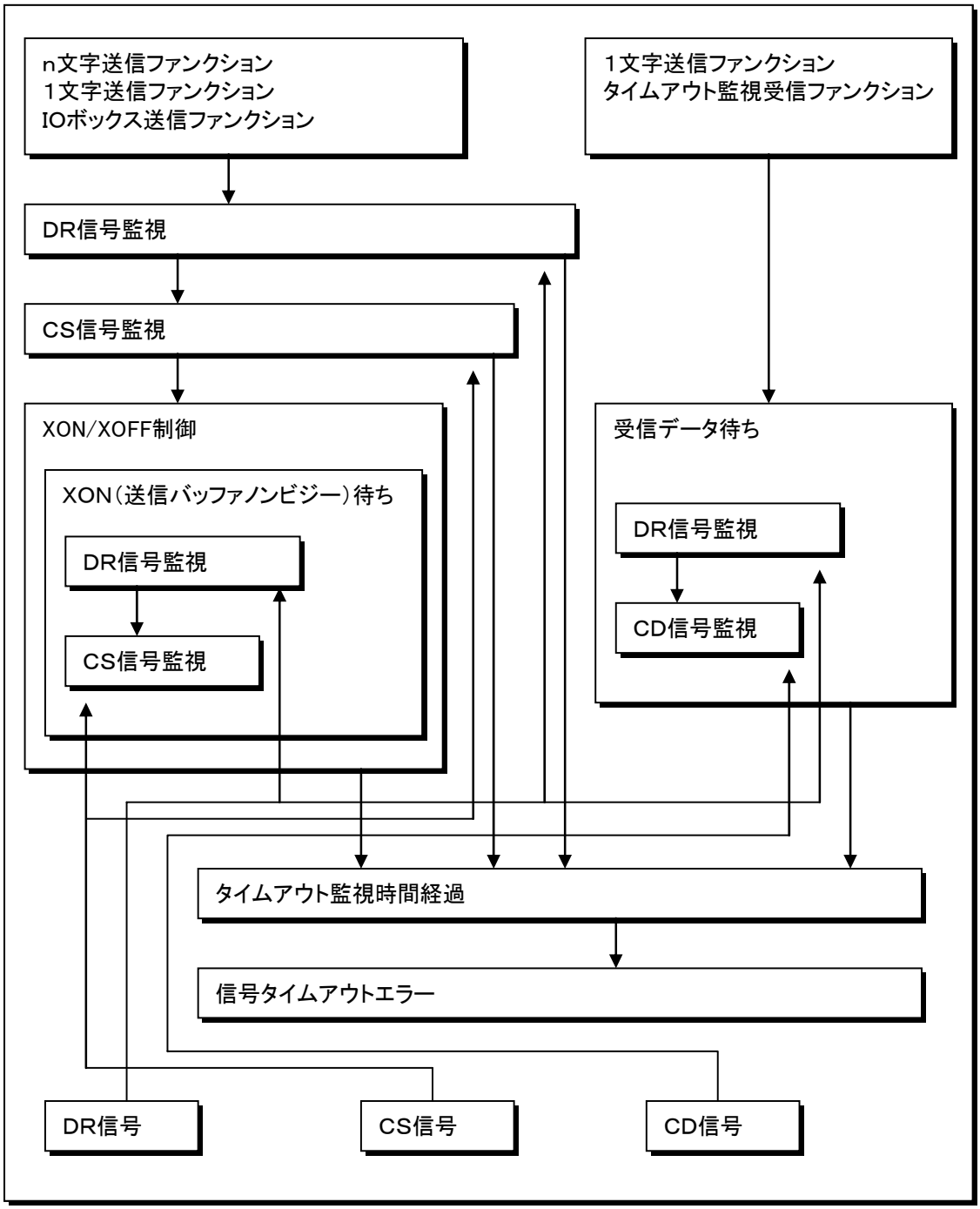
(2) タイムアウト監視

通信ポートのオープン、転送データの送信、受信データの読込および ER/RS 信号の ON を行なうとき、DR/CS/CD 信号の ON または OFF 状態の監視(遷移待ち)を行ないます。
DR/CS/CD 信号が規定の状態(ON または OFF)でないとき監視を行ない、タイムアウト監視値の時間だけ経過すると信号タイムアウトエラーとなります。
また、タイムアウト監視値の設定値により信号の監視を行なわないようにすることができます。
信号を監視するには"COM のオープン"、"DR/CS/CD タイムアウト監視値の設定"ファンクションでタイムアウト監視値を指定します。

【ER/RS 信号 ON のタイムアウト監視の流れ】



【転送データ送受信のタイムアウト監視の流れ】



14.2.8 LB 検出

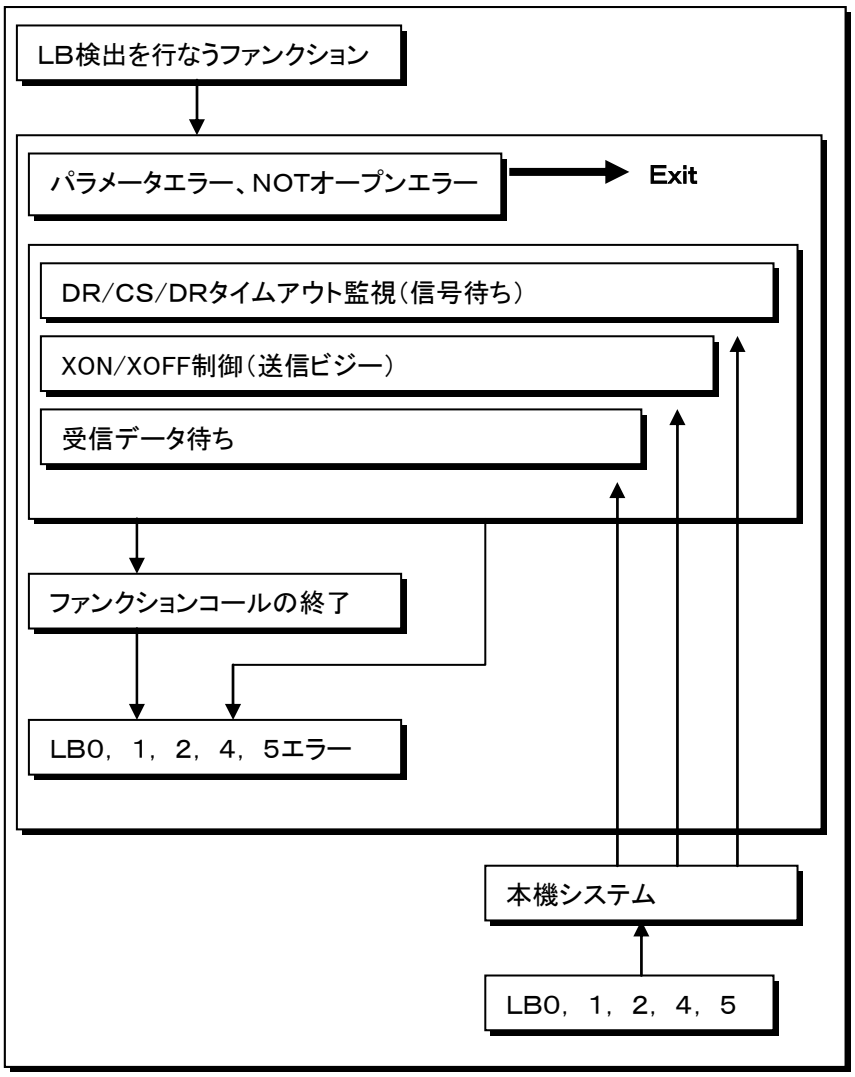
各ファンクションコールでは指定によりローバッテリの検出を行ないます。
LB を検出するとファンクションは実行中の処理を中断し、異常終了します。
LB の検出を行なうには DT-930 のシステムに対して LB 検出を行なうように設定する必要があります。

(1) LB の検出を行なうファンクション

LB の検出はフロー制御、信号タイムアウト監視、受信データ待ちおよび各ファンクションコールの処理終了時に行ないます。
エラーステータスは"CERR_o_LBx"を通知します。
ローバッテリには次のような種類があります。

LB 名称	内容
LB0	・主電池なし ・電池蓋外し
LB1	・主電池警告
LB2	・副電池警告
LB4	・APO
LB5	・電源 OFF

【LB 検出の流れ】



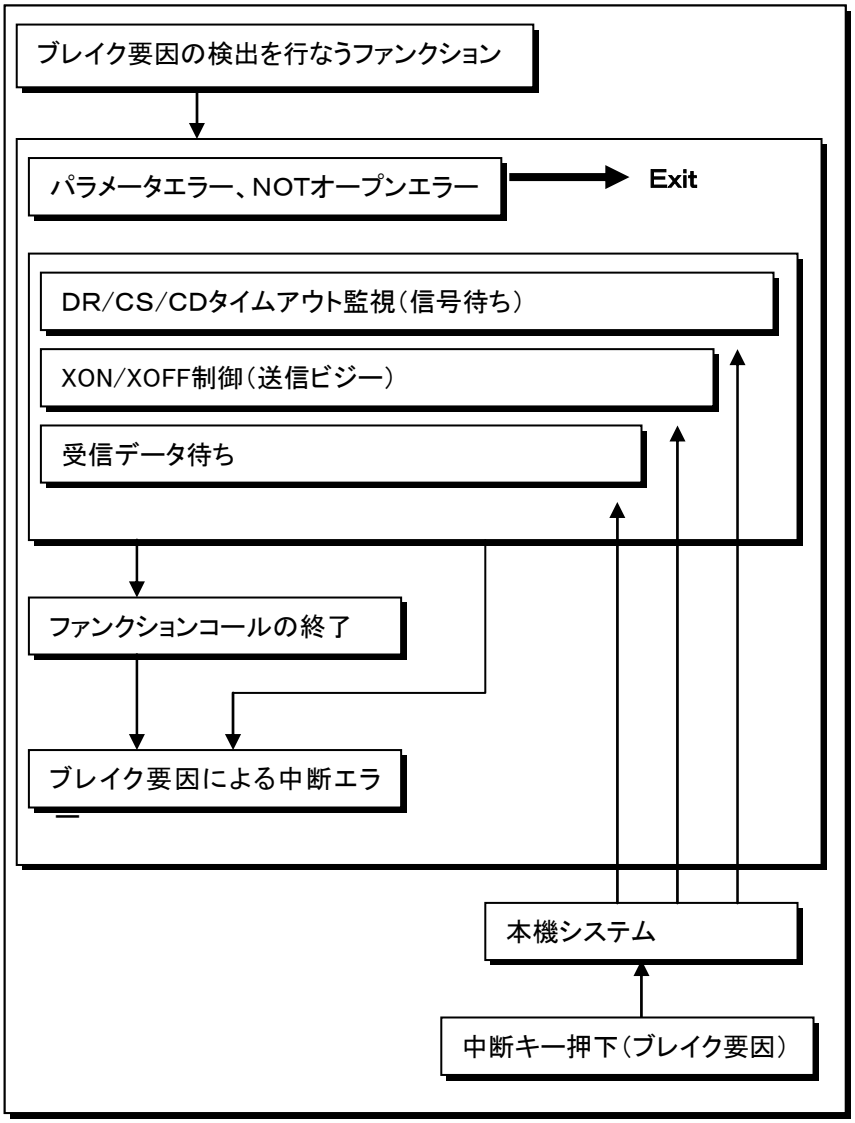
14.2.9 ブレイク要因検出

各ファンクションコールでは指定によりブレイク要因の検出を行ないます。ブレイク要因を検出するとファンクションは実行中の処理を中断し、異常終了します。
ブレイク要因の検出を行なうには DT-930 のシステムに対してブレイク要因の検出を行なうように設定する必要があります。ブレイク要因の検出は"ブレイク要因の設定"ファンクションで設定することができます。
ブレイク要因は通信ポートのオープン直前(信号タイムアウト監視を行う前に)、ブレイク要因の検出後に一度クリアします。

(1) ブレイク要因の検出を行なうファンクション

ブレイク要因の検出はフロー制御、信号タイムアウト監視、受信データ待ちのときに行ないます。
ブレイク要因の検出を行なうファンクションについては「14.3 エラー詳細(p.253)」で各ファンクションのエラーステータスを参照してください。エラーステータスの"CERR_o_BREAK"を通知します。

【ブレイク要因検出の流れ】



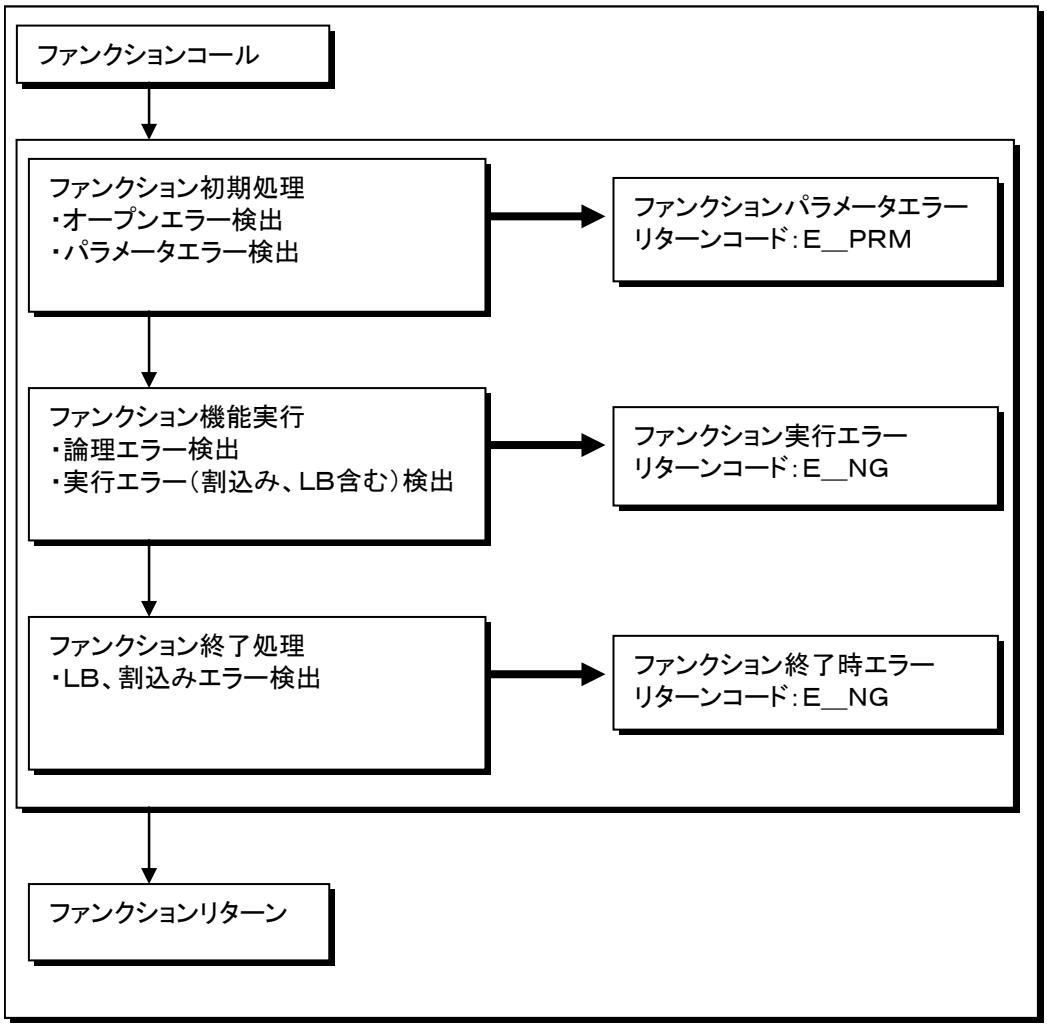
14.3 エラー詳細

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。
“エラーステータスのリード”ファンクションでエラーステータスを取得することができます。

14.3.1 ファンクションのエラー検出

エラーステータスを設定するファンクションではパラメータのエラー、ファンクションの実行エラーおよび実行終了時に割込みおよび外部要因のエラー検出を行ないます。
ファンクションのリターンコードが異常終了であるとき、ファンクションの処理は行なわれている場合があります。例えば”1 文字送信”ファンクションコールが終了時エラーの検出で異常終了となったとき、転送データ送信は正しく行なわれていることが考えられます。
基本的にエラーが発生した場合は、データ通信の信頼性が損なわれる状況にありますので通信を中断してエラーの原因を取り除きプロトコル等のデータ通信をやり直すようにしてください。

【ファンクションのエラー検出の流れ】



14.3.2 エラー詳細

各ファンクションコールが異常終了したときのエラーステータスを以下に示します。

(1) ファンクション終了時エラー

各ファンクションの実行終了に検出する割込みおよび外部要因のエラーステータスです。

エラーコード	エラーステータス	要因
E_NG	CERR_r_PARITY	パリティエラー ・パリティエラー検出
	CERR_r_OVERRUN	オーバーランエラー ・オーバーランエラー検出
	CERR_r_FLAMING (CERR_r_FRAMING)	フレーミングエラー ・フレーミングエラー検出
	CERR_r_BUFFUL	バッファフルエラー ・バッファフルエラー検出
	CERR_o_LBx	ローバッテリーエラー参照 ・LBx 検出(x＝0,1,2,4,5)

(2) ローバッテリ(LB)エラー

各ファンクションで検出するローバッテリー(外部要因)のエラーステータスです。

エラーコード	エラーステータス	要因
E_NG	CERR_o_LB0	本体主電池なし(LB0) ・LB0 検出
	CERR_o_LB1	本体主電池電圧低下(LB1) ・LB1 検出
	CERR_o_LB2	副電池電圧なし(LB2) ・LB2 検出
	CERR_o_LB4	APO による電源 OFF(LB4) ・LB4 検出
	CERR_o_LB5	OFF キー押下による電源 OFF(LB5) ・LB5 検出

(3) COM のオープン

エラーコード	エラーステータス	要因
E_NG	CERR_f_DEMESNE	占有エラー ・"COM の占有"ファンクションで通信ポートは占有中 ・通信ポートはオープン中 ・IrDA ポートが使用中 カシオ IR ポートと IrDA ポートはシステムリソースを共用しているため、排他制御を行っている
	CERR_f_DRTIMEOUT	DR 信号タイムアウト
	CERR_f_CSTIMEOUT	CS 信号タイムアウト
	CERR_f_CDTIMEOUT	CD 信号タイムアウト
	CERR_o_BREAK	ブレイク要因による中断 ・信号タイムアウト監視中にブレイク要因検出
	CERR_o_LBx (x=0、1、2、4、5)	ローバッテリーエラー参照 ・信号タイムアウト監視中に LBx 検出
E_PRM	なし	パラメータエラー ・通信ポートの指定が不当
	CERR_f_PARAMETER	パラメータエラー ・受信バッファレングスが範囲外 ・デリートコード数が範囲外 ・パリティビットの指定が不当 ・ストップビットの指定が不当 ・キャラクタレングスの指定が不当 ・ボーレートの指定が不当 ・各信号タイムアウト値が範囲外

(4) COM のクローズ

エラーコード	エラーステータス	要因
E_NG	CERR_f_NOTOPEN	NOT OPEN エラー ・通信ポートはオープンされていない
E_PRM	なし	パラメータエラー ・通信ポートの指定が不当

(5) COM のステータスリード

エラーコード	エラーステータス	要因
E_NG	CERR_f_NOTOPEN	NOT OPEN エラー ・通信ポートはオープンされていない
	CERR_o_LBx (x=0、1、2、4、5)	ローバッテリーエラー参照
E_PRM	なし	パラメータエラー ・通信ポートの指定が不当

(6) COM の占有

エラーコード	エラーステータス	要因
E_NG	CERR_f_DEMESNE	占有エラー ・通信ポートは既に占有されている ・通信ポートはオープン中 ・IrDA ポートが使用中 カシオ IR ポートと IrDA ポートはシステムリソースを共用しているため、排他制御を行っている
	ファンクション終了時エラー	ファンクション終了時エラー参照
E_PRM	なし	パラメータエラー ・通信ポートの指定が不当 ・占有/解除指定が無効
	CERR_f_PARAMETER	パラメータエラー ・占有していないポートを占有解除した

(7) COM のオープンチェック

エラーコード	エラーステータス	要因
通信ポートの オープン状態	なし	発生するエラーはありません 関数結果には通信ポートのオープン状態を返します

(8) n 文字送信

エラーコード	エラーステータス	要因
E_NG	CERR_f_NORECOVER	致命的エラー ・"送受信の有効/無効"ファンクションで送信無効に設定されている ・カシオ IR ポート使用時に送信有効でない状態 → "COM のオープン"、"IO ボックス送信"ファンクションの実行後 ・"ブレイク送出の ON/OFF" ファンクションでブレイク ON 中
	CERR_f_DRTIMEOUT	DR 信号タイムアウト
	CERR_f_CSTIMEOUT	CS 信号タイムアウト
	CERR_o_BREAK	ブレイク要因による中断 ・信号タイムアウト監視中にブレイク要因検出
	CERR_f_NOTOPEN	NOT OPEN エラー ・通信ポートはオープンされていない
	CERR_o_LBx (x=0、1、2、4、5)	ローバッテリーエラー参照 ・信号タイムアウト監視中に LBx 検出
	ファンクション終了時エラー	ファンクション終了時エラー参照
E_PRM	なし	パラメータエラー ・通信ポートの指定が不当
	CERR_f_PARAMETER	パラメータエラー ・送信レングスが範囲外

(9) 1 文字受信

エラーコード	エラーステータス	要因
E_NG	CERR_f_NORECOVER	致命的エラー ・受信データ待ちのとき、 “送受信の有効/無効”ファンクションで受信無効に設定されている
	CERR_f_DRTIMEOUT	DR 信号タイムアウト
	CERR_f_CDTIMEOUT	CD 信号タイムアウト
	CERR_r_PARITY	パリティエラー ・受信データ待ちのときパリティエラー
	CERR_r_OVERRUN	オーバーランエラー ・受信データ待ちのときオーバーランエラー
	CERR_r_FLAMING (CERR_r_FRAMING)	フレーミングエラー ・受信データ待ちのときフレーミングエラー
	CERR_r_PARITY2	パリティエラー
	CERR_r_OVERRUN2	オーバーランエラー
	CERR_r_FLAMING2 (CERR_r_FRAMING2)	フレーミングエラー
	CERR_r_BUFFULL	バッファフルエラー ・受信データ待ちのときバッファフルエラー
	CERR_o_BREAK	ブレイク要因による中断 ・受信データ待ち、信号タイムアウト監視中にブレイク要因検出
	CERR_f_NOTOPEN	NOT OPEN エラー ・通信ポートはオープンされていない
	CERR_o_LBx (x=0、1、2、4、5)	ローバッテリーエラー参照 ・受信データ待ち、各信号タイムアウト監視中に LBx 検出
	ファンクション終了時エラー	ファンクション終了時エラー参照
E_PRM	なし	パラメータエラー ・通信ポートの指定が不当

(10) タイムアウト監視受信

エラーコード	エラーステータス	要因
E_NG	CERR_f_NORECOVER	致命的エラー ・受信データ待ちのとき “送受信の有効/無効”ファンクションで受信無効に設定されている
	CERR_f_DRTIMEOUT	DR 信号タイムアウト
	CERR_f_CDTIMEOUT	CD 信号タイムアウト
	CERR_f_RCVTOUT	受信タイムアウト
	CERR_r_PARITY	パリティエラー ・受信データ待ちのときパリティエラー
	CERR_r_OVERRUN	オーバーランエラー ・受信データ待ちのときオーバーランエラー
	CERR_r_FLAMING (CERR_r_FRAMING)	フレーミングエラー ・受信データ待ちのときフレーミングエラー
	CERR_r_PARITY2	パリティエラー
	CERR_r_OVERRUN2	オーバーランエラー
	CERR_r_FLAMING2 (CERR_r_FRAMING2)	フレーミングエラー
	CERR_r_BUFFULL	バッファフルエラー ・受信データ待ちのときバッファフルエラー
	CERR_o_BREAK	ブレイク要因による中断 ・受信データ待ち、信号タイムアウト監視中にブレイク要因検出
	CERR_f_NOTOPEN	NOT OPEN エラー ・通信ポートはオープンされていない
	CERR_o_LBx (x=0、1、2、4、5)	ローバッテリーエラー参照 ・受信データ待ち、各信号タイムアウト監視中に LBx 検出
	ファンクション終了時エラー	ファンクション終了時エラー参照
E_PRM	CERR_f_PARAMETER	パラメータエラー ・受信タイムアウト監視値が範囲外
	なし	パラメータエラー ・通信ポートの指定が不当

(11) 1 文字送信

エラーコード	エラーステータス	要因
E_NG	CERR_f_NORECOVER	致命的エラー ・"送受信の有効/無効"ファンクションで送信無効に設定されている ・カシオ IR ポート使用時に送信有効でない状態 → "COM のオープン"、"IO ボックス送信"ファンクションの実行後 ・"ブレイク送出の ON/OFF"ファンクションでブレイク ON 中
	CERR_f_DRTIMEOUT	DR 信号タイムアウト
	CERR_f_CSTIMEOUT	CS 信号タイムアウト
	CERR_o_BREAK	ブレイク要因による中断 ・信号タイムアウト監視中にブレイク要因検出
	CERR_f_NOTOPEN	NOT OPEN エラー ・通信ポートはオープンされていない
	CERR_o_LBx (x=0、1、2、4、5)	ローバッテリーエラー参照 ・信号タイムアウト監視中に LBx 検出
	ファンクション終了時エラー	ファンクション終了時エラー参照
E_PRM	なし	パラメータエラー ・通信ポートの指定が不当

(12) ブレイク送出の ON/OFF

エラーコード	エラーステータス	要因
E_NG	CERR_f_NOTOPEN	NOT OPEN エラー ・通信ポートはオープンされていない
	ファンクション終了時エラー	ファンクション終了時エラー参照
E_PRM	なし	パラメータエラー ・通信ポートの指定が不当
	CERR_f_PARAMETER	パラメータエラー ・ブレイク ON/OFF の指定が不当

(13) 送受信の有効/無効

エラーコード	エラーステータス	要因
E_NG	CERR_f_NOTOPEN	NOT OPEN エラー ・通信ポートはオープンされていない
	ファンクション終了時エラー	ファンクション終了時エラー参照)
E_PRM	なし	パラメータエラー ・通信ポートの指定が不当
	CERR_f_PARAMETER	パラメータエラー ・送受信の有効/無効の指定が不当

(14) IO ボックス送信設定

エラーコード	エラーステータス	要因
E_NG	CERR_NOTOPEN	NOT OPEN エラー ・通信ポートはオープンされていない
	ファンクション終了時エラー	ファンクション終了時エラー参照
E_PRM	なし	パラメータエラー ・通信ポートの指定が不当
	CERR_f_PARAMETER	パラメータエラー ・送信状態設定の指定が不当

(15) IO ボックス送信

エラーコード	エラーステータス	要因
E_NG	CERR_f_NORECOVER	致命的エラー ・"ブレイク送出の ON/OFF" ファンクションでブレイク ON 中
	CERR_f_DRTIMEOUT	DR 信号タイムアウト
	CERR_f_CSTIMEOUT	CS 信号タイムアウト
	CERR_o_BREAK	ブレイク要因による中断 ・各信号タイムアウト監視中にブレイク要因検出
	CERR_f_NOTOPEN	NOT OPEN エラー ・通信ポートはオープンされていない
	CERR_o_LBx (x=0、1、2、4、5)	ローバッテリーエラー参照 ・信号タイムアウト監視中に LBx 検出
	ファンクション終了時エラー	ファンクション終了時エラー参照
E_PRM	なし	パラメータエラー ・通信ポートの指定が不当
	CERR_f_PARAMETER	パラメータエラー ・送信レングスが範囲外

(16) 受信バッファのクリア

エラーコード	エラーステータス	要因
E_NG	CERR_f_NOTOPEN	NOT OPEN エラー ・通信ポートはオープンされていない
	ファンクション終了時エラー	ファンクション終了時エラー参照
E_PRM	なし	パラメータエラー ・通信ポートの指定が不当

(17) 受信バッファステータスのリード

エラーコード	エラーステータス	要因
E_NG	CERR_f_NOTOPEN	NOT OPEN エラー ・通信ポートはオープンされていない
	CERR_r_PARITY2	パリティエラー
	CERR_r_OVERRUN2	オーバーランエラー
	CERR_r_FLAMING2 (CERR_r_FRAMING2)	フレーミングエラー
	ファンクション終了時エラー	ファンクション終了時エラー参照
E_PRM	なし	パラメータエラー ・通信ポートの指定が不当

(18) エラーコードバッファリング制御の設定

エラーコード	エラーステータス	要因
E_NG	ファンクション終了時エラー	ファンクション終了時エラー参照
E_PRM	なし	パラメータエラー ・通信ポートの指定が不当
	CERR_f_PARAMETER	パラメータエラー ・バッファリング制御の指定が不当

(19)受信ハンドラ切替え

エラーコード	エラーステータス	要因
E_PRM	なし	パラメータエラー ・通信ポートの指定が不当
	CERR_f_PARAMETER	パラメータエラー ・ハンドラの指定が不当

(20) DR/CS/CD タイムアウト監視値の設定

エラーコード	エラーステータス	要因
E_NG	ファンクション終了時エラー	ファンクション終了時エラー参照
	CERR_f_NOTOPEN	NOT OPEN エラー ・通信ポートはオープンされていない
E_PRM	なし	パラメータエラー ・通信ポートの指定が不当
	CERR_f_PARAMETER	パラメータエラー ・タイムアウト監視値の指定が範囲外

(21) ER 信号の ON/OFF

エラーコード	エラーステータス	要因
E_NG	CERR_f_DRTIMEOUT	DR 信号タイムアウト
	CERR_f_NOTOPEN	NOT OPEN エラー ・通信ポートはオープンされていない
	CERR_o_BREAK	ブレイク要因による中断 ・信号タイムアウト監視中にブレイク要因検出
	CERR_o_LBx (x=0、1、2、4、5)	ローバッテリーエラー参照 ・信号タイムアウト監視中に LBx 検出
	ファンクション終了時エラー	ファンクション終了時エラー参照
E_PRM	なし	パラメータエラー ・通信ポートの指定が不当
	CERR_f_PARAMETER	パラメータエラー ・信号 ON/OFF 指定が不当

(22) RS 信号の ON/OFF

エラーコード	エラーステータス	要因
E_NG	CERR_f_CSTIMEOUT	CS 信号タイムアウト
	CERR_f_CDTIMEOUT	CD 信号タイムアウト
	CERR_f_NOTOPEN	NOT OPEN エラー ・通信ポートはオープンされていない
	CERR_o_BREAK	ブレイク要因による中断 ・信号タイムアウト監視中にブレイク要因検出
	CERR_o_LBx (x=0、1、2、4、5)	ローバッテリーエラー参照 ・信号タイムアウト監視中に LBx 検出
	ファンクション終了時エラー	ファンクション終了時エラー参照
E_PRM	なし	パラメータエラー ・通信ポートの指定が不当
	CERR_f_PARAMETER	パラメータエラー ・信号 ON/OFF 指定が不当

(23) ER/RS 信号の ON/OFF

エラーコード	エラーステータス	要因
E_NG	CERR_f_DRTIMEOUT	DR 信号タイムアウト
	CERR_f_CSTIMEOUT	CS 信号タイムアウト
	CERR_f_CDTIMEOUT	CD 信号タイムアウト
	CERR_f_NOTOPEN	NOT OPEN エラー ・通信ポートはオープンされていない
	CERR_o_BREAK	ブレイク要因による中断 ・信号タイムアウト監視中にブレイク要因検出
	CERR_o_LBx (x=0、1、2、4、5)	ローバッテリーエラー参照 ・信号タイムアウト監視中に LBx 検出
	ファンクション終了時エラー	ファンクション終了時エラー参照
E_PRM	なし	パラメータエラー ・通信ポートの指定が不当
	CERR_f_PARAMETER	パラメータエラー ・信号 ON/OFF 指定が不当

(24) ブレイク要因の設定

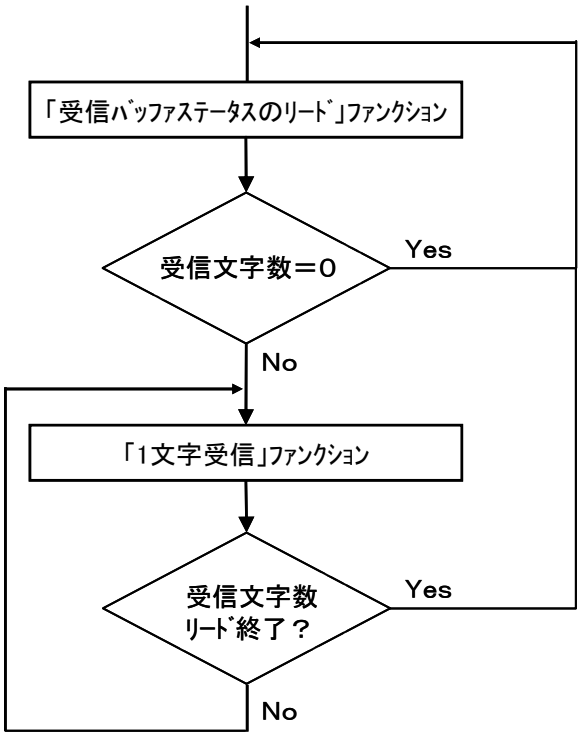
エラーコード	エラーステータス	要因
E_PRM	なし	パラメータエラー ・ブレイク要因通知の指定が不当 ・ファンクションキーの指定が不当

14.4 通信関数 解説

通信関数が提供する機能について解説します。

14.4.1 受信データの読込

受信データを"受信バッファステータスのリード"ファンクションを使用して受信バッファから受信文字数だけ読込む例を示します。



①受信文字数を取得
エラーを随時チェックする

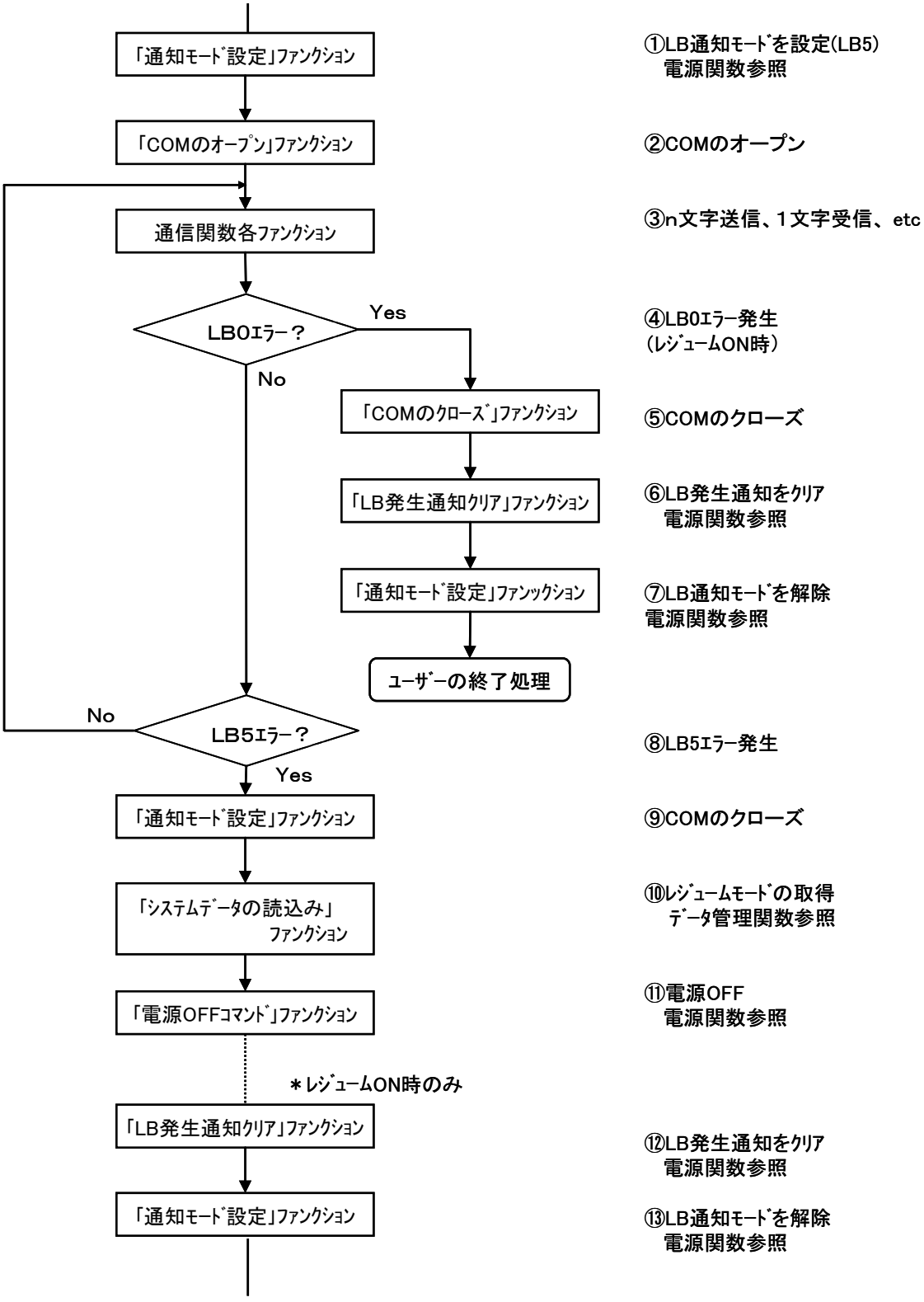
②受信文字が無ければ繰り返し

③受信ﾊﾞｯﾌｧから受信ﾃﾞｰﾀをﾘｰﾄﾞ

④受信文字を全てﾘｰﾄﾞする
エラーを随時チェックする

14.4.2 LB エラーチェック

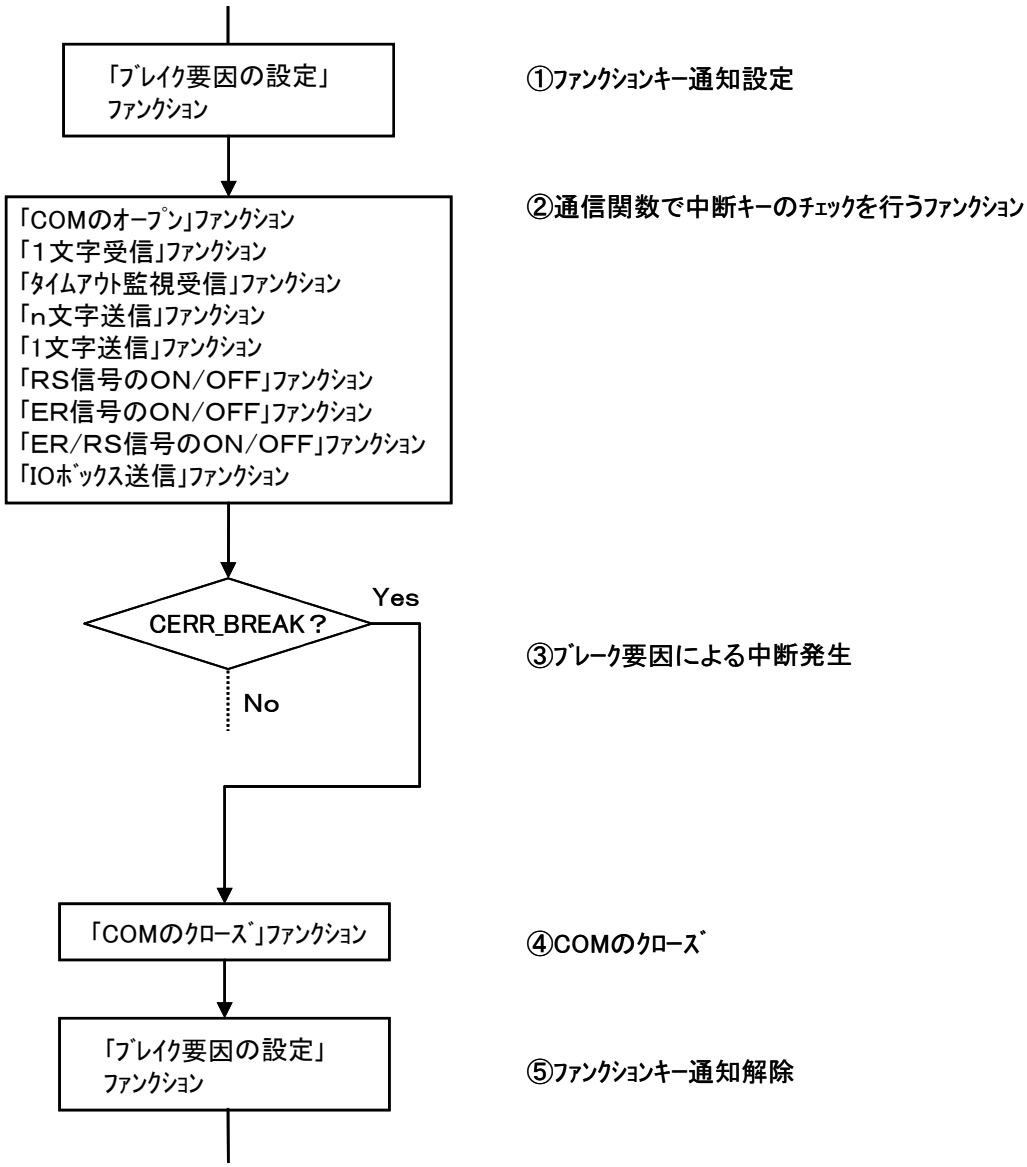
ローバッテリ(LB)エラーのチェックを通信中の電源 OFF キー押下(LB5)および主電池なし(LB0)発生時の処理例を以下に示します。レジューム ON による通信の再開等ができない場合などを考慮して行ないます。



※ LB エラー以外のチェック、処理は随時行なうこと。
通信中に APO(LB4)発生の禁止は、”APO 禁止設定/解除”(pwr_hold_apo 参照)で行なう必要があります。

14.4.3 中断キーによる処理

通信関数内部での受信データ待ちの強制解除のための中断キー処理例を以下に示します。



※ 他のエラーチェック、処理は随時行なう必要があります。

14.4.4 カシオ IR ポートの使用

(1) ベーシック IO ボックスとの接続

カシオ IR ポートはベーシック IO ボックスと接続して使用します。このとき半二重制御を行なう必要があり、次のファンクションコールを使用します。

- 送受信の有効/無効
- IO ボックス送信設定
- IO ボックス送信

半二重制御を行なう場合、次の 2 つ点を配慮しなくてはいけません。

a) CTRL 信号の切替え時のターンアラウンドタイム

CTRL 信号を切替えてデータ転送を開始するまでの間に 7.8ms 以上のターンラウンドタイムが必要です。

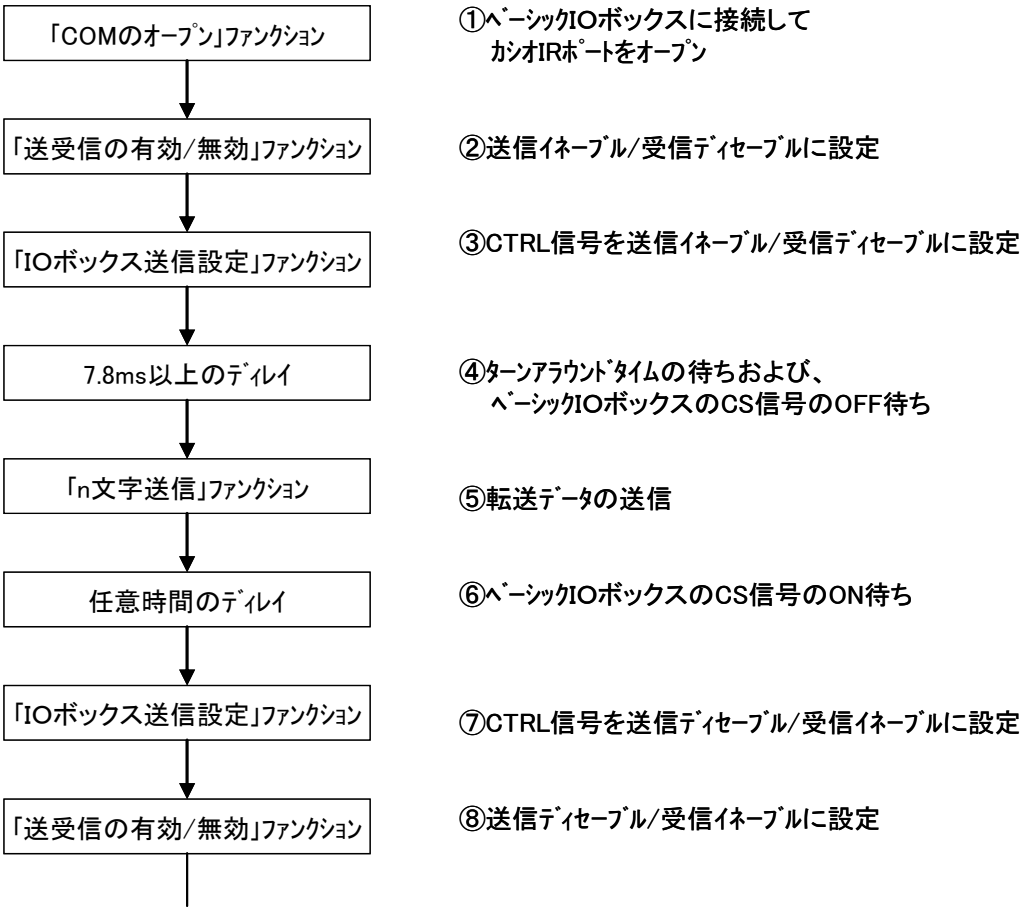
b) CTRL 信号とベーシック IO ボックスの CS 信号(RS-232C インタフェース側)の状態

DT-930 が転送データの送信を行なうとき CTRL 信号が送信イネーブル、ベーシック IO ボックスの CS 信号は OFF でなければなりません。また、DT-930 が転送データの受信を行なうとき CTRL 信号が送信ディセーブル、ベーシック IO ボックスの CS 信号は OFF でなければなりません。

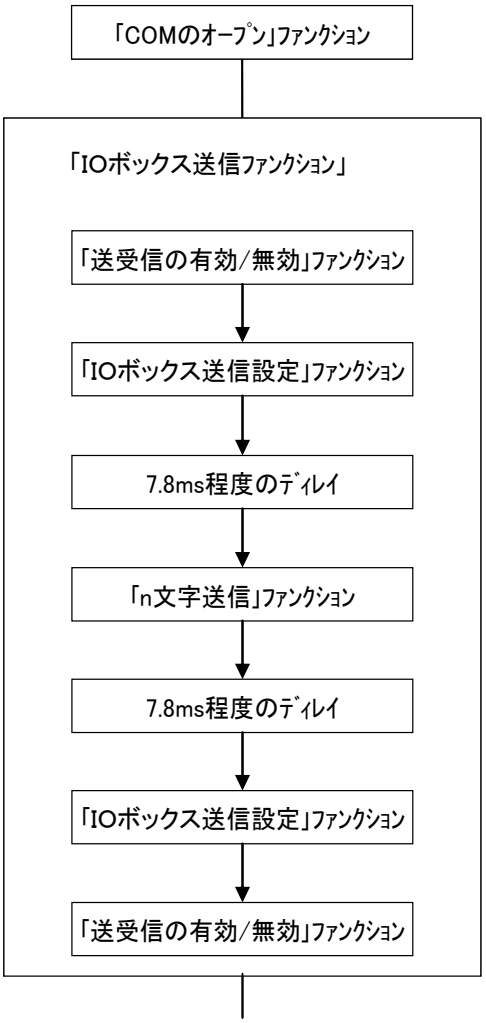
CTRL 信号が送信ディセーブル、ベーシック IO ボックスの CS 信号が OFF であるとき DT-930 にフレーミングエラー(ノイズが発生する)が発生する場合があります。

尚、例 2 に示す"IO ボックス送信"ファンクションを使用する場合、接続先のユーザーエンティティは半二重制御のタイミング(例 2 の④、⑥)を"IO ボックス送信"ファンクションに合わせる必要があります。

【転送データの送信例 1】


「COMのオープン」ファンクション
①ベーシックIOボックスに接続して
カシオIRポートをオープン
「送受信の有効/無効」ファンクション
②送信イネーブル/受信ディセーブルに設定
「IOボックス送信設定」ファンクション
③CTRL信号を送信イネーブル/受信ディセーブルに設定
7.8ms以上のディレイ
④ターンアラウンドタイムの待ちおよび、
ベーシックIOボックスのCS信号のOFF待ち
「n文字送信」ファンクション
⑤転送データの送信
任意時間のディレイ
⑥ベーシックIOボックスのCS信号のON待ち
「IOボックス送信設定」ファンクション
⑦CTRL信号を送信ディセーブル/受信イネーブルに設定
「送受信の有効/無効」ファンクション
⑧送信ディセーブル/受信イネーブルに設定

【転送データの送信例 2】



①ベーシックIOボックスに接続して
カシオIRポートをオープン

②送信イネーブル/受信ディセーブルに設定

③CTRL信号を送信イネーブル/受信ディセーブルに設定

④ターンアラウンドタイムの待ちおよび、
ベーシックIOボックスのCS信号のOFF待ち

⑤転送データの送信

⑥ベーシックIOボックスのCS信号のON待ち

⑦CTRL信号を送信ディセーブル/受信イネーブルに設定

⑧送信ディセーブル/受信イネーブルに設定

(2) CERR_r_xxxx2 エラーステータスのチェック

先に述べたとおり、CTRL 信号とベーシック IO ボックスの RS232C の CS 信号が共に OFF であるときフレーミングエラーが発生する場合があります。
しかし、前述の"(1)ベーシック IO ボックスとの接続"に示す例のようにソフトウェアで CTRL 信号とベーシック IO ボックスの CS 信号が共に OFF にならない状態を作り出すことは容易でないと考えられます。従ってカシオ IR ポートで無手受信のデータ通信を行なうことは不向きであり、データ通信を行なうユーザーエンティティの間で何等かの通信手順が必要になります。
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エラーステータスの CERR_f_PARITY2、FRAMING2、OVERRUN2 で行なうと、"1 文字受信"、"タイムアウト監視受信"ファンクションで受信データとエラーを時系列で得ることができます。
以下にエラーチェックの例を示します。

【エラーチェックの方法】

・通信手順(プロトコル)を用いてデータ転送を行う。フレームの形式は以下の通りです。
・本機が受信側、ベーシックIOボックスが送信側とします。
・カシオIRが使用するシリアルコントローラのレシーバはFIFOバッファなのでフレームのヘッダ、フッタは余分に16文字付加すべきです。

BOF	DATA	FCS	EOF

→

0xB0	0x41 0x42 0x43	0xC6	0xC0

BOF:ﾌﾚｰﾑﾍｯﾀﾞ
DATA:転送ﾃﾞｰﾀ
FCS:ﾌﾚｰﾑﾁｪｯｸｼｰｹﾝｽ
EOF:ﾌﾚｰﾑﾌｯﾀ

	CTRL	CS	転送ﾃﾞｰﾀ	受信バッファ	ｴﾗｰ	ﾌｧﾝｸｼｮﾝｺｰﾙ
①	ON	OFF				
②	OFF	OFF	0xFF		ﾌﾚｰﾐﾝｸﾞｴﾗｰ	IOボックス送信設定

＊ CTRLとCSがOFFとなりﾌﾚｰﾐﾝｸﾞｴﾗｰとなった。ｴﾗｰ(ﾉｲｽﾞ)のﾃﾞｰﾀは破棄されます。

③	OFF	ON	0xB0 0x41 0x42	0xB0 0x41 0x42		
④	OFF	ON		0xB0 0x41 0x42		1文字受信
⑤	OFF	ON		0xB0 0x41 0x42		ｴﾗｰｽﾃｰﾀｽのﾘｰﾄﾞ

＊ ｴﾗｰｽﾃｰﾀｽがCERR_f_FRAMING or CERR_f_FRAMING2。読出したﾃﾞｰﾀ'0xBO'は正常なﾃﾞｰﾀです。
CERR_f_FRAMING2なので'0xB0'より前にｴﾗｰとなったことが分かる。'0xB0'はﾌﾚｰﾑ(ﾍｯﾀﾞ)であるのでｴﾗｰはﾌﾚｰﾑ本体のには影響を与えない。受信ﾃﾞｰﾀの読込みを継続します。

⑥	OFF	ON		0x41 0x42		1文字受信
⑦	OFF	ON		0x42		1文字受信

次ページへ続く

	CTRL	CS	転送ﾃﾞｰﾀ	受信バッファ	ｴﾗｰ	ﾌｧﾝｸｼｮﾝｺｰﾙ
⑧	OFF	ON	0x43 0xC6 0xC0	0x42 0x43 0xC6 0xC0		
⑨	OFF	OFF	0xFF	0x42 0x43 0xC6 0xC0	ﾌﾚｰﾐﾝｸﾞ	

＊ CTRLとCSがOFFとなりﾌﾚｰﾐﾝｸﾞｴﾗｰとなった。ｴﾗｰのﾃﾞｰﾀは破棄されます。

⑩	OFF	OFF		0x43 0xC6 0xC0	1文字受信
⑪	OFF	OFF		0x43 0xC6 0xC0	ｴﾗｰｽﾃｰﾀｽのﾘｰﾄﾞ

＊ ｴﾗｰｽﾃｰﾀｽがCERR_f_FRAMING。CERR_f_FRAMING2ではないので’0x42’はﾌﾚｰﾐﾝｸﾞｴﾗｰでない。
受信ﾃﾞｰﾀの読込みを続けます。

⑫	OFF	OFF		0x44 0xC6 0xC0	1文字受信
⑬	OFF	OFF		0xC6 0xC0	1文字受信
⑭	OFF	OFF		0xC0	1文字受信
⑮	OFF	OFF			1文字受信

＊ ﾌﾚｰﾑのﾌｯﾀである’0xC0’を読出しﾌﾚｰﾑが完成したので受信ﾃﾞｰﾀの読込みはここで終了します。

⑯	OFF	OFF			送受信の有効/無効

＊ 送信ｲﾈｰﾌﾞﾙ/受信ｲﾈｰﾌﾞﾙに設定します。

⑰	OFF	OFF			受信ﾊﾞｯﾌｧのｸﾘｱ

＊ 受信バッファとｴﾗｰｽﾃｰﾀｽをｸﾘｱします。

⑱	ON	OFF			IBOX送信設定

＊ 送信ｲﾈｰﾌﾞﾙ/受信ｲﾈｰﾌﾞﾙに設定し、次は転送ﾃﾞｰﾀの送信を行います。

14.4.5 通信関数部制限

通信関数部の制限項目と注意事項を以下に示します。

制限・注意事項一覧

	機能	内容
1	デリートコード制御	・SI/SO 制御、XON/XOFF 制御、エラーコードバッファリング制御で使用する制御コードとデリートコードが重複しないように設定してください
2	XON/XOFF 制御	・SI/SO 制御、デリートコード制御、エラーコードバッファリング制御で使用する制御コードと XON/XOFF コードが重複しないように設定してください XON コードの既定値は 0x11、XOFF コードの既定値は 0x13 です ・”受信バッファのクリア”ファンクションを使用すると通信コントローラのレシーバに格納されている制御コードもクリアされる可能性があります ・カシオ IR ポートを使用する場合は使用でしないでください ・通信関数内部受信バッファ使用する場合は使用でしないでください
3	RS/CS フロー制御	・カシオ IR ポートを使用する場合は使用しないでください ・通信関数内部受信バッファ使用する場合は使用でしないでください
4	SI/SO 制御	・XON/XOFF 制御、デリートコード制御、エラーコードバッファリング制御で使用する制御コードと SI/SO コードが重複しないように設定してください SI コードの既定値は 0x0F、SO コードの既定値は 0x0E です
5	エラーコードバッファリング 制御	・XON/XOFF 制御、デリートコード制御、SI/SO 制御で使用する制御コードとエラーコードが重複しないように設定してください ・カシオ IR、が使用するシリアルコントローラのレシーバは FIFO バッファになっています。通信エラーが発生したときにレシーバ内の先頭の文字がエラーコードバッファリング制御の対象となります
6	外部要因エラーの検出	・LB0,1,2,4,5 およびブレイク要因は、システムがセットするイベントフラグの参照により、検出します。 従って検出を行なう場合はシステムに対してイベントセットを行なうように設定してください
7	CERR_r_xxxx2 エラーステータス	・カシオ IR、シリアルポートが使用するシリアルコントローラのレシーバは FIFO バッファになっています。通信エラーが発生したときにレシーバ内のデータのいずれかがエラーであることを示しています。エラーステータスはレシーバから先頭の文字を読出したときに設定しています。これにより”1 文字受信”、”タイムアウト監視受信”ファンクションでエラーとなったきに読出した文字と実際にエラーとなった文字に最大プラス 15 文字(レシーバが 16 バイト)の誤差があります
8	ブレイク信号	・カシオ IR ポートではブレイク信号の送出および検出をするこができません ブレイク信号を送信した場合は 1 バイト長のデータのスペース(ストップビットまで)となります(送出停止までの間、連続して送出されます) ブレイク信号を受信した場合はフレーミングエラーとなります
9	受信バッファのクリア	・フロー制御中に受信バッファビジーであるとき”受信バッファのクリア”ファンクションで受信バッファのクリアを行なった場合、受信ビジーは解除しません

14.5 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。

14.5.1 c_open

カシオ IR、の通信ポートをオープンします。
オープンしたときは半二重(受信イネーブル、送信ディセーブル)の状態でオープンします。
本ファンクションは次の処理および設定を行ないます。

- 通信ポート電源の ON
- SI/SO 制御の設定
- DR/CS/CD 信号タイムアウト監視
- 送受信の有効
- フロー制御の設定
- デリートコード設定
- 受信割込みの許可
- 受信バッファの設定
- 通信ポートの排他制御
- 通信形式の設定
- ER/RS 信号設定

```
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
COM0 :カシオ IR インタフェース
COM1 :予約
COM2 :予約
COM3 :予約

param

通信形式パラメータ(各パラメータの論理和で指定)

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

※ カシオ IR インタフェースの場合は予約領域になります.

busy_ch

XOFF コード(受信不可時のコード)

nonbusy_ch

XON コード (受信可能時のコード)

buff

受信バッファアドレス

buf_l

受信バッファレングス

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

tim_out

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

del_cod

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

戻り値

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

解説

”DR/CS/CD タイムアウト監視値の設定”により信号線の監視を行ないます。
カシオ IR ポートをオープンする場合はフロー制御の指定は行なわないでください。
ER 信号 OFF、RS 信号 ON に設定したとき CS 信号 ON、CD 信号 OFF のタイマ監視は CS タイムアウト監視値で行ないます。
カシオ IR ポートオープン時は CTRL 信号を送信ディセーブル/受信イネーブルに設定します。

14.5.2 c_close

オープン中の通信ポートをクローズし、通信ポートの使用を禁止します。クローズした通信ポートを使用してデータ通信を行なうことはできません。
本ファンクションは次の処理を行ないます。

- 通信ポートの電源 OFF
- 通信ポートの排他解除
- 送受信の無効
- 各信号線の OFF
- 受信割込みの禁止

```
ER c_close(
  H     com_no
);
```

パラメータ

com_no
通信ポート

COM0	:カシオ IR インタフェース
COM1	:予約
COM2	:予約
COM3	:予約

戻り値

関数結果

E_OK	:正常終了
E_NG	:異常終了
E_PRM	:パラメータエラー

14.5.3 c_status

通信ポートのステータスを読出します。ステータスのアトリビュートには次のものがあります。
通信エラー、受信バッファオーバーフローおよびブレイク信号は、割込み要因のエラーステータスは本ファンクションによりクリアします。

- 信号線(DR,CD,CS)の ON/OFF
- 受信バッファオーバーフロー
- 通信エラー(パリティ,オーバーラン,フレーミング)
- ブレイク信号の受信
- 受信バッファに格納されたデータ(受信データ)の有無

```
ER c_status(
  H  com_no
);
```

パラメータ

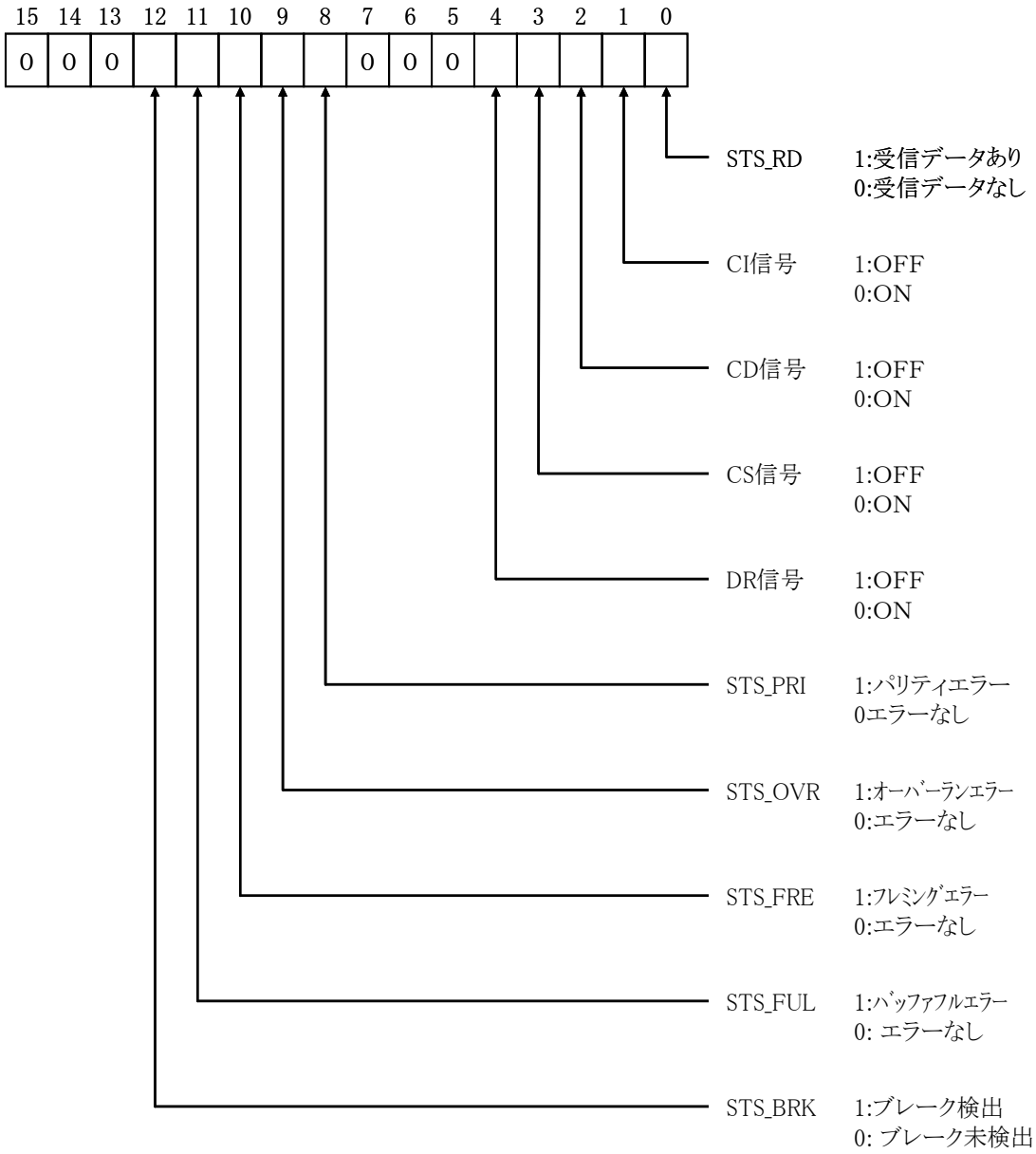
com_no
通信ポート

COM0	:カシオ IR インタフェース
COM1	:予約
COM2	:予約
COM3	:予約

戻り値

E_NG	:異常終了
E_PRM	:パラメータエラー
上記以外	:COM ステータス(次ページ参照)

COM ステータス


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

14.5.4 c_hold

通信ポートを占有します。
占有されている通信ポートをオープンすることはできません。

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

パラメータ

com_no
通信ポート
COM0 :カシオ IR インタフェース
COM1 :予約
COM2 :予約
COM3 :予約

mode
占有設定
HOLD_ON :占有する
HOLD_OFF :占有解除

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

14.5.5 c_chkopen

通信ポートのオープン状態を読出します。各通信ポートがオープン/クローズ中であるかチェックできます。
また、”COM の占有”ファンクションおよび”IrCOMM オープン”ファンクションにより占有状態にある通信ポートを知ることができます。
(IrDA ポートの使用中は占有状態になります。)

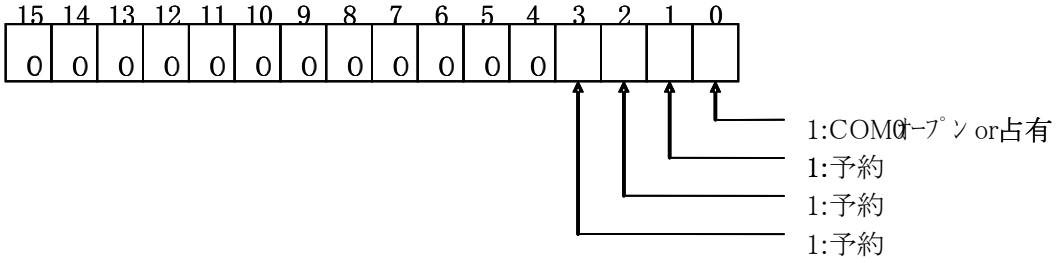
```
ER c_chkopen();
```

パラメータ

なし

戻り値

関数結果(ｵｰﾌﾟﾝ or 占有通知)



14.5.6 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

COM0	:カシオ IR インタフェース
COM1	:予約
COM2	:予約
COM3	:予約

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

解説

”COM のオープン”ファンクションによりフロー制御を行ないます。
”COM のオープン”ファンクションにより SI/SO 制御を行ないます。
”COM のオープン”ファンクションにより信号線の監視を行ないます。
”DR/CS/CD タイムアウト監視値の設定”ファンクションにより信号線の監視を行ないます。

14.5.7 c_din

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

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

パラメータ

com_no
通信ポート

COM0	:カシオ IR インタフェース
COM1	:予約
COM2	:予約
COM3	:予約

buffer
格納バッファアドレス

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

解説

”COM のオープン”ファンクションによりフロー制御を行ないます。
”COM のオープン”ファンクションにより信号線の監視を行ないます。
”DR/CS/CD タイムアウト監視値の設定”ファンクションにより信号線の監視を行ないます。

14.5.8 c_tmdin

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

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

パラメータ

com_no
通信ポート

COM0	:カシオ IR インタフェース
COM1	:予約
COM2	:予約
COM3	:予約

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

解説

”COM のオープン”ファンクションによりフロー制御を行ないます。
”COM のオープン”ファンクションにより信号線の監視を行ないます。
”DR/CS/CD タイムアウト監視値の設定”ファンクションにより信号線の監視を行ないます。

14.5.9 c_out

データを 1 文字(バイト)送信します。

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

パラメータ

com_no
通信ポート

COM0	:カシオ IR インタフェース
COM1	:予約
COM2	:予約
COM3	:予約

snddata
送信文字 (1 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

解説

”COM のオープン”ファンクションによりフロー制御を行います。
”COM のオープン”ファンクションにより SI/SO 制御を行います。
”COM のオープン”ファンクションにより信号線の監視を行います。
”DR/CS/CD タイムアウト監視値の設定”ファンクションにより信号線の監視を行います。

14.5.10 c_break

ブレーク信号の送出または、送出停止を行います。

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

パラメータ

com_no
通信ポート

COM0	:カシオ IR インタフェース
COM1	:予約
COM2	:予約
COM3	:予約

mode
ブレーク信号制御

BRK_ON	:ブレーク信号を送出する
BRK_OFF	:ブレーク信号を停止する

戻り値

E_OK	:正常終了
E_NG	:異常終了
E_PRM	:パラメータエラー

14.5.11 c_txrx

通信コントローラの送受信動作を有効(イネーブル)または無効(ディセーブル)に設定します。
送信動作を無効に設定した場合、データの送信を行うことができなくなります。
また、受信動作を無効に設定した場合、データの受信を行うことができなくなります。

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

パラメータ

com_no
通信ポート

COM0	:カシオ IR インタフェース
COM1	:予約
COM2	:予約
COM3	:予約

mode
送受信の有効/無効

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

戻り値

E_OK	:正常終了
E_NG	:異常終了
E_PRM	:パラメータエラー

14.5.12 c_iobox

CTRL 信号を制御してベーシック IO ボックスの送受信イネーブルを設定します。ベージック IO ボックスへの接続はカシオ IR ポートのみ可能です。カシオ IR ポートは半二重制御でデータ転送を行います。
本ファンクションを使用してベーシック IO ボックスの送信イネーブル(受信ディセーブル)と受信イネーブル(送信ディセーブル)を排他的に設定します。
DT-930 から送信するときは送信イネーブル、DT-930 が受信するときに受信イネーブルに指定します。

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

パラメータ

com_no
通信ポート
COM0　　:カシオ IR インタフェース

mode
送信状態設定
C_IOBOXENB　　:送信に設定(送信ｲﾈｰﾌﾞﾙ/受信ﾃﾞｨｾｰﾌﾞﾙ)
C_IOBOXDSB　　:送信を解除(送信ﾃﾞｨｾｰﾌﾞﾙ/受信ｲﾈｰﾌﾞﾙ)

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

14.5.13 c_irout

ベーシック IO ボックスへデータの送信を行います。
ベーシック IO ボックスへの接続はカシオ IR ポートのみ可能です。
カシオ IR ポートは半二重制御でデータ転送を行います。
本ファンクションは、CTRL 信号を制御して送信イネーブル(受信ディセーブル)に設定し、データの送信を行います。
データの送信が終了すると受信イネーブル(送信ディセーブル)の状態に設定します。
本ファンクションは"n 文字送信"、"送受信の有効/無効"、"IOボックス送信設定"の各ファンクションの機能を 1 つにまとめた機能を持ちます。従って、本ファンクションの正常終了後の CTRL 信号は OFF となり、送受信の有効/無効の設定は送信ディセーブル、送受信イネーブルとなります。

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

パラメータ

com_no
通信ポート
COM0　　:カシオ IR インタフェース

buffer
送信バッファアドレス

length
送信文字数(バイト数)

戻り値

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

解説

"COM のオープン"ファンクションにより SI/SO 制御を行います。
"COM のオープン"ファンクションにより信号線の監視を行います。
"DR/CS/CD タイムアウト監視値の設定"ファンクションにより信号線の監視を行います。

14.5.14 c_flush

受信バッファ内と通信コントローラのレシーバ内の転送データを破棄し、初期化します。
また"エラーステータスのリード"、"COM ステータスのリード"ファンクションで通知するステータス(パリティ、フレーミング、オーバーラン、バッファフルエラー)をクリアします。ただし、これらのステータスは、ファンクション終了時の異常終了チェックで検出可能です。
本ファンクションを XON/XOFF 制御指定時に使用すると XON/XOFF 制御コードが失われる可能性がありますので、XON/XOFF 制御指定時は使用しないで下さい。

```
ER c_flush(
  H   com_no
);
```

パラメータ

com_no
通信ポート

COM0	:カシオ IR インタフェース
COM1	:予約
COM2	:予約
COM3	:予約

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

14.5.15 c_bfsts

受信バッファのステータスをリードします。ステータスのアトリビューには次のものがあります。
尚、NOT オープンエラー、パラメータエラー以外で異常終了となったとき、ステータスのリードを行います。

- 受信バッファに格納されている読出し可能なデータ数(受信文字数:バイト単位)
- 受信バッファの先頭に格納されているデータの文字コード(次読出し文字)
- 受信バッファに格納できるデータ数(受信可能文字数:バイト単位)

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

パラメータ

com_no
通信ポート

COM0	:カシオ IR インタフェース
COM1	:予約
COM2	:予約
COM3	:予約

bfsts
受信バッファステータス

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

解説

【ストラクト構造】

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

14.5.16 c_errbfring

関数結果バッファリング制御の設定を行います。
通信エラー(パリティ、オーバーラン、フレーミング)が発生したとき、指定のコードを受信バッファへ格納します。通信エラーとなった受信データは受信バッファに格納しません。
システムデフォルト値は、"エラーコードバッファリング制御しない"となっています。

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

パラメータ

com_no
通信ポート

COM0	:カシオ IR インタフェース
COM1	:予約
COM2	:予約
COM3	:予約

mode
設定/解除

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

c_errcd
エラーコード(任意の 1 バイトコード)

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

14.5.17 c_rderrsts

エラーステータスを読出しおよびクリアを行います。
各ファンクションの関数結果が異常終了であるとき本ファンクションでエラーステータスを読出し詳細を調べることができます。
エラーステータスは複数の場合があります。

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

パラメータ

com_no
通信ポート

COM0	:カシオ IR インタフェース
COM1	:予約
COM2	:予約
COM3	:予約

com_status
エラーステータス

戻り値

E_OK	:正常終了
E_PRM	:パラメータエラー

解説

エラーステータス詳細は、「14.3 エラー詳細(p.253)」を参照して下さい。

14.5.18 c_chghdr

受信ハンドラ(受信割込み処理)を標準または、簡易ハンドラに切替えを行います。
標準ハンドラでは、次の 5 つの項目の処理を行います。簡易ハンドラは受信データバッファリング処理のみを行い、標準ハンドラより割込み処理時間を短縮します。

- バッファバッファフロー制御
- SI/SO 制御
- デリートコード制御
- エラーコードバッファリング制御
- 受信データバッファリング処理

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

パラメータ

com_no
通信ポート

COM0	:カシオ IR インタフェース
COM1	:予約
COM2	:予約
COM3	:予約

mode
受信ハンドラ切替え

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

戻り値

E_OK	:正常終了
E_PRM	:パラメータエラー

14.5.19 c_timer

DR/CS/CD 信号の監視を転送データの送信や受信した転送データの読出しなどのときに行う指定をします。
各ファンクションで信号の ON または OFF をタイムアウト監視値の時間だけ待ち、タイムアウト監視値の時間を経過するとタイムアウトエラーとなります。
タイムアウト値が 0 であるときは監視を行いません。
本ファンクションは"COM のオープン"ファンクションの DR/CS/CD 信号タイムアウト監視設定と同じ機能を持ちます。

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

パラメータ

com_no
通信ポート

COM0	:カシオ IR インタフェース
COM1	:予約
COM2	:予約
COM3	:予約

cs_time
CS タイムアウト監視値設定(0～32767)×7.8ms

dr_time
DR タイムアウト監視値設定(0～32767)×7.8ms

cd_time
CD タイムアウト監視値設定(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

14.5.20 c_er

ER 信号線の ON/OFF を行います。

```
ER c_er(
   H   com_no,
   B   er_mode
);
```

パラメータ

com_no
通信ポート

COM0	:カシオ IR インタフェース
COM1	:予約
COM2	:予約
COM3	:予約

er_mode
ER 信号線の設定

ERS_ON	:ER 信号 ON
ERS_OFF	:ER 信号 OFF

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

解説

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

14.5.21 c_rs

RS 信号線の ON/OFF を行います。

```
ER c_rs(
   H   com_no,
   B   rs_mode
);
```

パラメータ

com_no
通信ポート

COM0	:カシオ IR インタフェース
COM1	:予約
COM2	:予約
COM3	:予約

mode
RS 信号線の設定

RS_ON	:RS 信号 ON
RS_OFF	:RS 信号 OFF

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

解説

”COM のオープン”ファンクションにより信号線の監視を行います。
”DR/CS/CD タイムアウト監視値の設定”ファンクションにより信号線の監視を行います。
RS 信号 ON に設定したとき CS 信号 ON、CD 信号 OFF タイマ監視は CS タイムアウト監視値で行います。

14.5.22 c_errs

ER/RS 信号線の ON/OFF を行います。

```
ER c_errs(
   H   com_no,
   B   errs_mode
);
```

パラメータ

com_no
通信ポート

COM0	:カシオ IR インタフェース
COM1	:予約
COM2	:予約
COM3	:予約

mode
ER/RS 信号線の設定

ERRS_ON	:ER/RS 信号 ON
ERRS_OFF	:ER/RS 信号 OFF

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

解説

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

14.5.23 c_brkevent

ブレイク要因イベント通知の設定または解除を行います。

```
ER c_brkevent(
   UH      event_mode,
   UB      func_mode
);
```

パラメータ

event_mode
ブレイクイベント通知
BRK_EVENT_ON　　:ブレイク要因の通知を行う
BRK_EVENT_OFF　　:ブレイク要因の通知を行わない

func_mode
ファンクションキー番号
FNC_1～FNC_8　　:ファンクションキー1～8
MLT_R,MLT_L　　:マルチファンクションキーR,L

戻り値

関数結果
E_OK　　:正常終了
E_PRM　　:パラメータエラー

解説

”COM のオープン”ファンクション内でブレイク要因はクリアします。
ブレイク要因の検出を行うファンクションでブレイク要因検出後にブレイク要因はクリアします。

15.IrDA 制御

15.1 機能

IrDA 制御

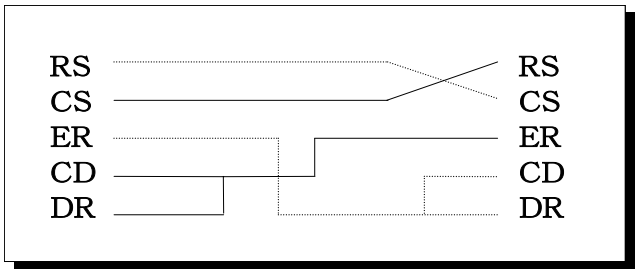
関数	機能概要
Ir_Open	IrCOMM のオープン
Ir_Close	IrCOMM のクローズ
Ir_Read	データの読込
Ir_Write	データの書込
Ir_QueryTx	送信データ数の問合せ
Ir_QueryRx	受信データ数の問合せ
Ir_EROn	ER 信号の ON
Ir_EROff	ER 信号の OFF
Ir_RSOn	RS 信号の ON
Ir_RSOff	RS 信号の OFF
Ir_BreakOn	ブレーク送信の ON
Ir_BreakOff	ブレーク送信の OFF
Ir_CheckCD	CD の検査
Ir_CheckDR	DR の検査
Ir_CheckCS	CS の検査
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

15.1.1 シリアルポートエミュレーション

IrDA 部は SIR(最大 115.2Kbps)と FIR(最大 4Mbps)を搭載します。本章で説明する"シリアルポートエミュレーション"は SIR モード時に動作可能です。
FIR モードは、USB IO ボックスで使用可能なモードです。本章で説明する Ir_xxx 関数では動作しませんので、FIR を使用する際は必ず、Ir モード設定関数(Ir_SetWinMode 関数)により FIR に切替後、通信ユーティリティ関数 cu_xxx を使用してください。
ユーザーエンティティは IrDA(プロトコル)のフレーム形式を意識することなくデータ通信を行うことができます。

信号結線
IrDA ポートを使用して DT-930 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

(1) 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-930 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は IrCOMM(カシオ IR インタフェース)はクローズ状態となります。

(2) 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) データ読込
受信データの読込を行います。
ユーザ定義のエリアに受信バッファデータの読込を行い、読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込が可能です。
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込が行え、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込後に 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込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込が終了したとき、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) データ書込
送信データの書込を行います。
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込を行い、書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込が行えなくなるか(バッファビジ-)、ユーザー定義のバイトサイズまで書込を行います。
データ待ち時間が指定されていれば送信バッファに書込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込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込後に 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込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込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、"データ書込"機能と同様に送信バッファへの書込を行います。
データ待ちの間およびデータ書込後に 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、"データ書込"機能と同様に送信バッファへの書込を行います。
データ待ちの間およびデータ書込後に 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、"データ書込"機能と同様に送信バッファへの書込を行います。
データ待ちの間およびデータ書込後に 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、"データ書込"機能と同様に送信バッファへの書込を行います。
データ待ちの間およびデータ書込後に 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、"データ書込"機能と同様に送信バッファへの書込を行います。
データ待ちの間およびデータ書込後に 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、"データ書込"機能と同様に送信バッファへの書込を行います。
データ待ちの間およびデータ書込後に 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込が行えないとき
- 受信バッファに読込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	・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 ・リセット直後はデフォルト値となります ・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。

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

受信データの読込を行います。

- データ受信待ち(受信バッファにデータが無いとき)
- 受信データの読込(受信バッファからの読込)
- LB チェック
- ブレイクイベントのチェック
- 読込データ数の通知

```
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
読込データ数(受信バッファから読込できたバイト数)

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込が行われています)

15.2.4 Ir_Write

送信データの書込を行います。

- LB チェック
- データ書込待ち(送信バッファへの書込が終了するまで)
- 送信データの書込(送信バッファへの書込)
- ブレイクイベントのチェック
- 書込データ数の通知

```
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(送信バッファに書込むバイト数)

PutSize
書込データ数(送信バッファに書込できたバイト数)

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込が行われています)

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込可能なデータ数を問合せます。

- 受信バッファ内の読込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
読込可能なデータ数(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
- 送信データの書込(制御データ)
- データ書込待ち(送信バッファへの書込が終了するまで)

```
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
- 送信データの書込(制御データ)
- データ書込待ち(送信バッファへの書込が終了するまで)

```
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
- 送信データの書込(制御データ)
- データ書込待ち(送信バッファへの書込が終了するまで)

```
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
- 送信データの書込(制御データ)
- データ書込待ち(送信バッファへの書込が終了するまで)

```
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
- 送信データの書込(制御データ)
- データ書込待ち(送信バッファへの書込が終了するまで)

```
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
- 送信データの書込(制御データ)
- データ書込待ち(送信バッファへの書込が終了するまで)

```
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、次ページを参照して下さい

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 ｵｰﾌﾟﾝ状態	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	ｵｰﾌﾟﾝは行わずｸﾛｰｽﾞ状態	・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	ｵｰﾌﾟﾝは行わずｸﾛｰｽﾞ状態	・IO ボックス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	ｵｰﾌﾟﾝは行わずｸﾛｰｽﾞ状態	・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-930 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	ｵｰﾌﾟﾝは行わずｸﾛｰｽﾞ状態	・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)がｵｰﾌﾟﾝしています。ｸﾛｰｽﾞ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	ｵｰﾌﾟﾝは行わずｸﾛｰｽﾞ状態	・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	ｸﾛｰｽﾞ状態となります	・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)のｵｰﾌﾟﾝを行って下さい 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 Ir_CheckCD Ir_State_set Ir_SetPortConfig	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

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	ｵｰﾌﾟﾝは行わずｸﾛｰｽﾞ状態	・再実行する場合は Ir_Open から行って下さい
Ir_Close	ｸﾛｰｽﾞ状態となります	
Ir_Read	ｵｰﾌﾟ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	ｵｰﾌﾟﾝは行わずｸﾛｰｽﾞ状態	・再実行するときは Ir_Open を行って下さい
Ir_Close	ｸﾛｰｽﾞ状態となります	・イベントのクリアを行って下さい
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		

<table>
<tr><td>エラー値</td><td colspan="2">IRERR_LB1</td></tr>
<tr><td colspan="3">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</td></tr>
<tr><td>関数名</td><td>IrCOMM 状態</td><td>主なエラー対処方法</td></tr>
<tr><td>Ir_Open</td><td rowspan="3">ｵｰﾌﾟﾝは行わずｸﾛｰｽﾞ状態
ｸﾛｰｽﾞ状態となります</td><td rowspan="3">・電池交換を行って下さい
・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</td></tr>
<tr><td>Ir_Close</td></tr>
<tr><td>Ir_Init</td></tr>
<tr><td>Ir_Read</td><td rowspan="16">ｵｰﾌﾟﾝ状態から変更はありません</td><td rowspan="16">・Ir_Close を行って終了して下さい
・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</td></tr>
<tr><td>Ir_Write</td></tr>
<tr><td>Ir_EROn</td></tr>
<tr><td>Ir_EROff</td></tr>
<tr><td>Ir_RSOn</td></tr>
<tr><td>Ir_RSOff</td></tr>
<tr><td>Ir_BreakOn</td></tr>
<tr><td>Ir_BreakOff</td></tr>
<tr><td>Ir_CheckCD</td></tr>
<tr><td>Ir_CheckDR</td></tr>
<tr><td>Ir_CheckCS</td></tr>
<tr><td>Ir_QueryTx</td></tr>
<tr><td>Ir_QueryRx</td></tr>
<tr><td>Ir_CheckCI</td></tr>
<tr><td>Ir_CheckBreak</td></tr>
</table>

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	ｵｰﾌﾟﾝは行わずｸﾛｰｽﾞ状態	・電池交換後を行って下さい ・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	ｸﾛｰｽﾞ状態となります	
Ir_Init		
Ir_Read	ｵｰﾌﾟ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 ・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_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	ｵｰﾌﾟﾝは行わずｸﾛｰｽﾞ状態	・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	ｸﾛｰｽﾞ状態となります	
Ir_Init		
Ir_Read	ｵｰﾌﾟﾝ状態から変化はありません	・Ir_Close を行って終了して下さい ・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	ｵｰﾌﾟﾝは行わずｸﾛｰｽﾞ状態	・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	ｸﾛｰｽﾞ状態となります	
Ir_Init		
Ir_Read	ｵｰﾌﾟﾝ状態から変化はありません	・Ir_Close を行って終了して下さい ・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 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	ｸﾛｰｽﾞ状態から変化はありません	・IrCOMM が既にｸﾛｰｽﾞ状態になっています Ir_Open を行ってから実行して下さい

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	ｸﾛｰｽﾞ状態です	・通信環境を確認して再実行して下さい
Ir_Read	ｵｰﾌﾟ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
Ir_Write		
Ir_RSOn		
Ir_RSOff		
Ir_EROn		
Ir_EROff		
Ir_BreakOn		
Ir_BreakOff		

エラー値	IRERR_PARITY

詳細

Ir_State_Set 関数で次の指定のとき RS232C 上(IO ボックスと PC 間など)でパリティエラーとなったときに発生します

・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	ｵｰﾌﾟ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(IO ボックスと 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	ｵｰﾌﾟ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(IO ボックスと 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	ｵｰﾌﾟ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	・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
- データ読込/書込(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
);
```

パラメータ

State_DCB

【ストラクト構造】

```
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

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	秒単位にデータ読込/書込待ち時間を設定	○(300)
	THROUGH	データ読込/書込待ちを行わない	
	FOREVER	タイマ指定なしでデータ読込/書込待ちを行う	
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 のｽﾄｯﾌﾟﾋﾞｯﾄ	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	
RS232C のパリティﾋﾞｯﾄ	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	○

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
);
```

パラメータ

SetPortConfig_DCB

【ストラクト構造】

```
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

自局能力設定の DCB

項目	定数	詳細	デフォルト
IR ﾎﾞｰﾚｰﾄ ・OR で設定して下さい 設定した値が有効と なります	IRBPS_24	IR 接続速度を 2400bps に設定可能	設定有効
	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ﾄﾞﾀｲﾑ	TURN_500MS	最大ﾀｰﾝｱﾗｳﾝﾄﾞﾀｲﾑを 500ms に設定	○
ﾌﾚｰﾑｻｲｽﾞ	FRAME_1024B	ﾌﾚｰﾑｻｲｽﾞを 1024byte に設定	○
ｳｲﾝﾄﾞｳｻｲｽﾞ	WINDOW_4	ｳｲﾝﾄﾞｳｻｲｽﾞを 4 ﾌﾚｰﾑｳｲﾝﾄﾞｳに設定	○
BOF 数 ・IR ﾎﾞ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ﾄﾞﾀｲﾑ	TURN_5MS	最小ﾀｰﾝｱﾗｳﾝﾄﾞﾀｲﾑを 5ms に設定	
	TURN_1MS	最小ﾀｰﾝｱﾗｳﾝﾄﾞﾀｲﾑを 1ms に設定	
リンク開放時間 ・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

解説

15.2.22 Ir_SetWinMode

使用する IrDA のモード(SIR/FIR)を設定します。
USB IO ボックスを使用する際には、必ず FIR 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 :SIR モード
SET_WIN98 :FIR モード/USB IO ボックス使用時(Windows98 用)
SET_WIN2K :FIR モード/USB IO ボックス使用時(Windows98 以外)

戻り値

E_OK :正常終了
E_NG :異常終了

16.Bluetooth 制御

通信仕様

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_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_SetPassKey	Bluetooth パスキーの設定
BT_SelectDev	接続する Bluetooth 機器の指定
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

16.1.1 通信インタフェース

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-930 は Bluetooth プリンタと Bluetooth 通信を行います。その場合は DT-930 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-930 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。

(1) 通信する Bluetooth 機器のデバイス情報の登録

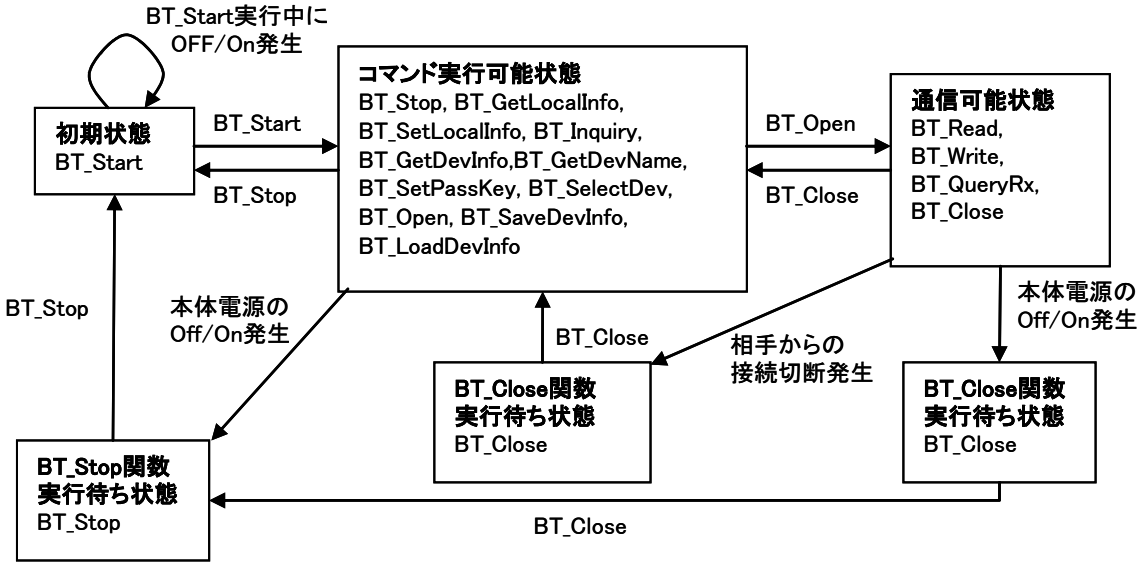
- Bluetooth 機器の Inquiry(問い合わせ)を実行
- Inquiry で発見された Bluetooth 機器の情報を取得
- 取得した Bluetooth 機器情報をファイルに保存

(2) 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 機能

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。

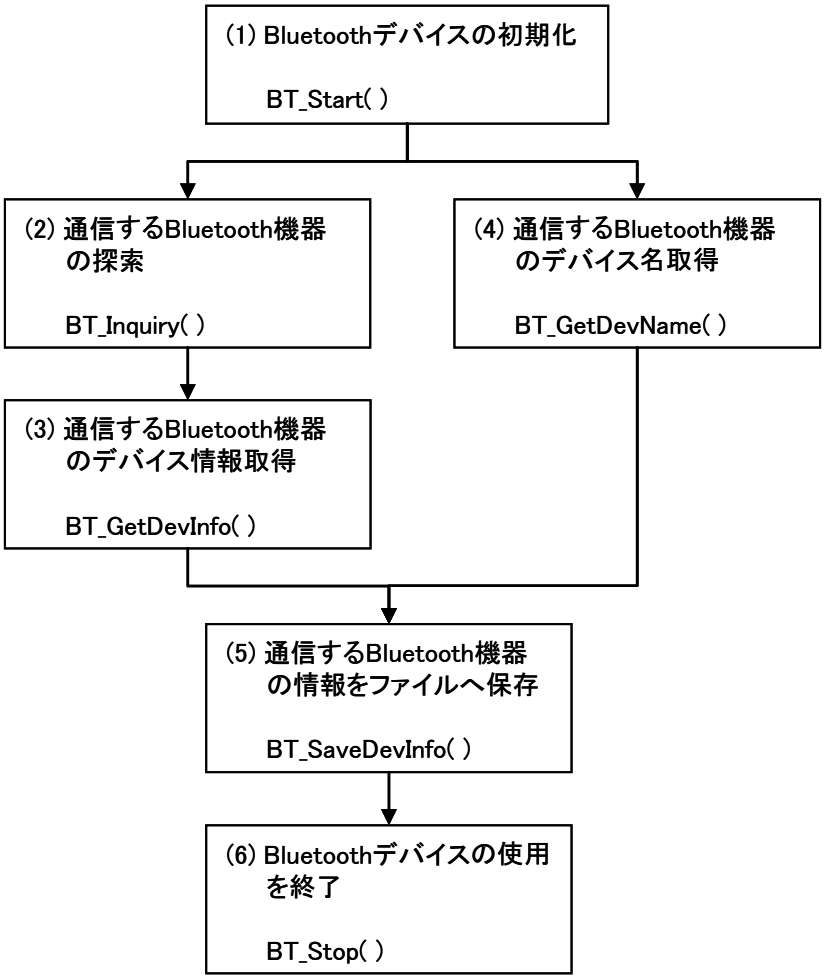
(4) 通信可能状態において本体電源の Off/On が発生したとき
BT_Close 関数および BT_Stop 関数を実行後、初期状態から関数処理を実行してください。

(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-930 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

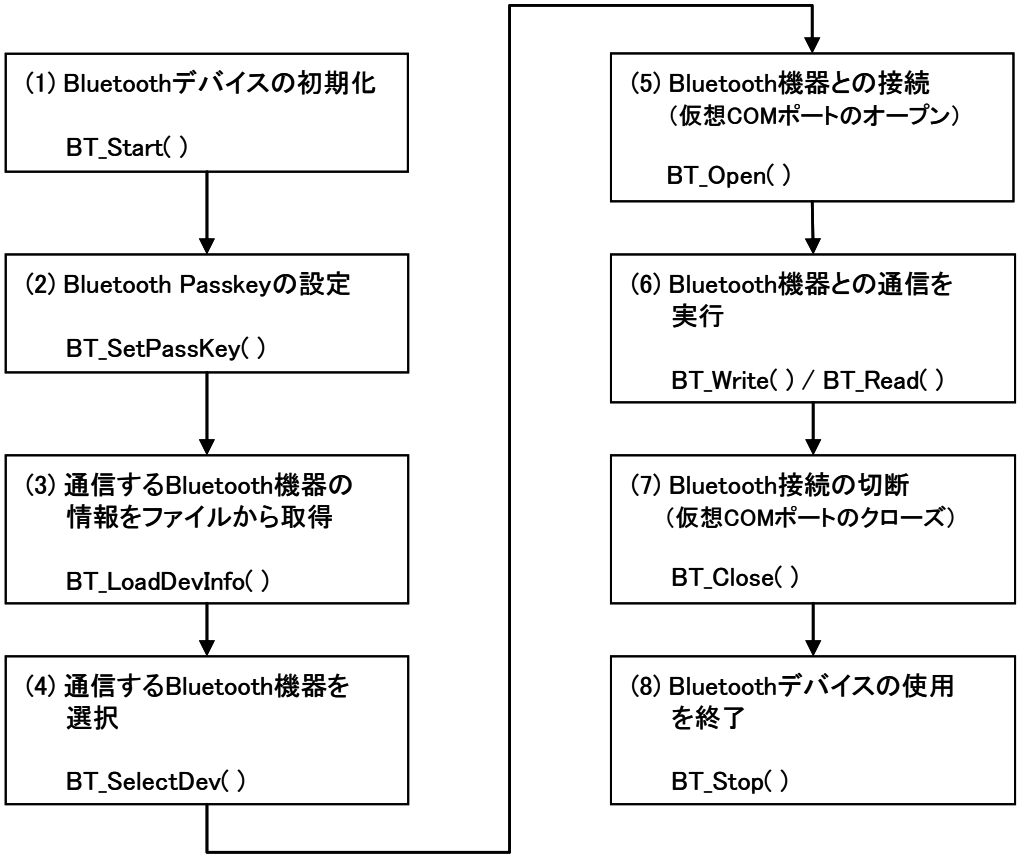
(1) Bluetooth 機器情報の取得と保存



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。

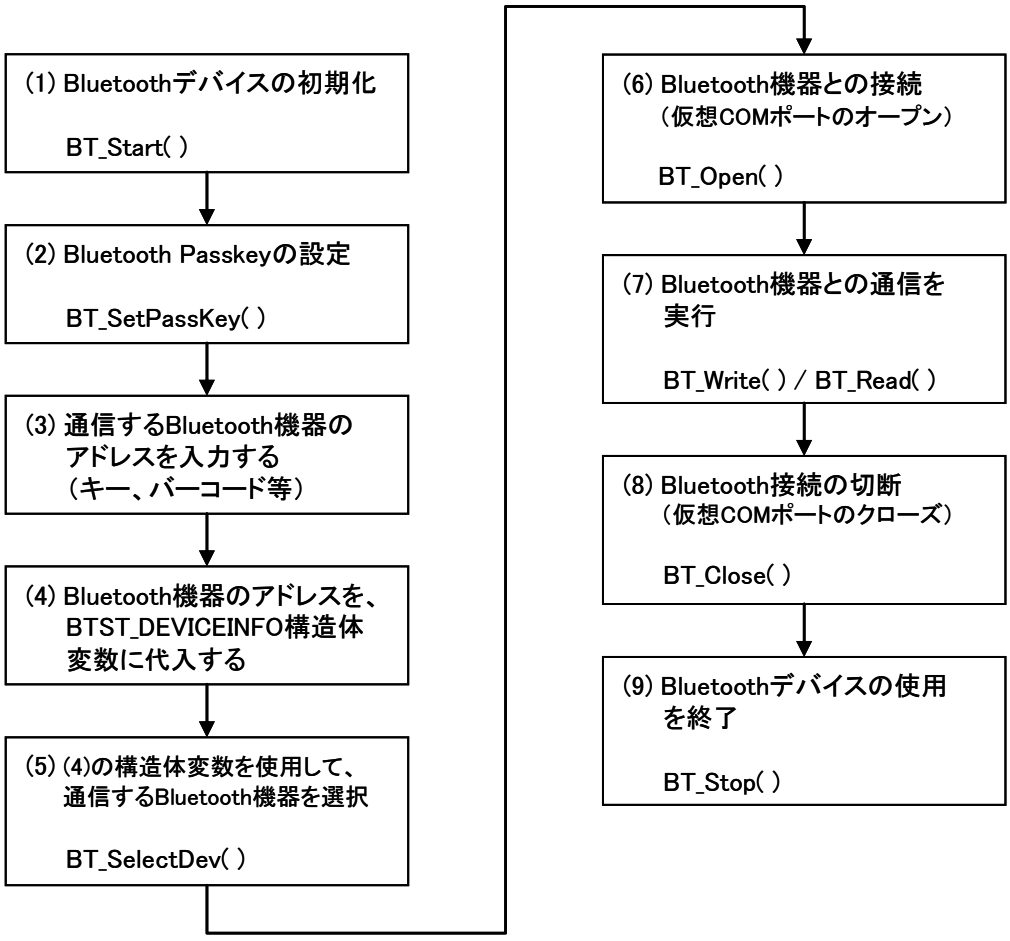
(2) Bluetooth 接続と通信

(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

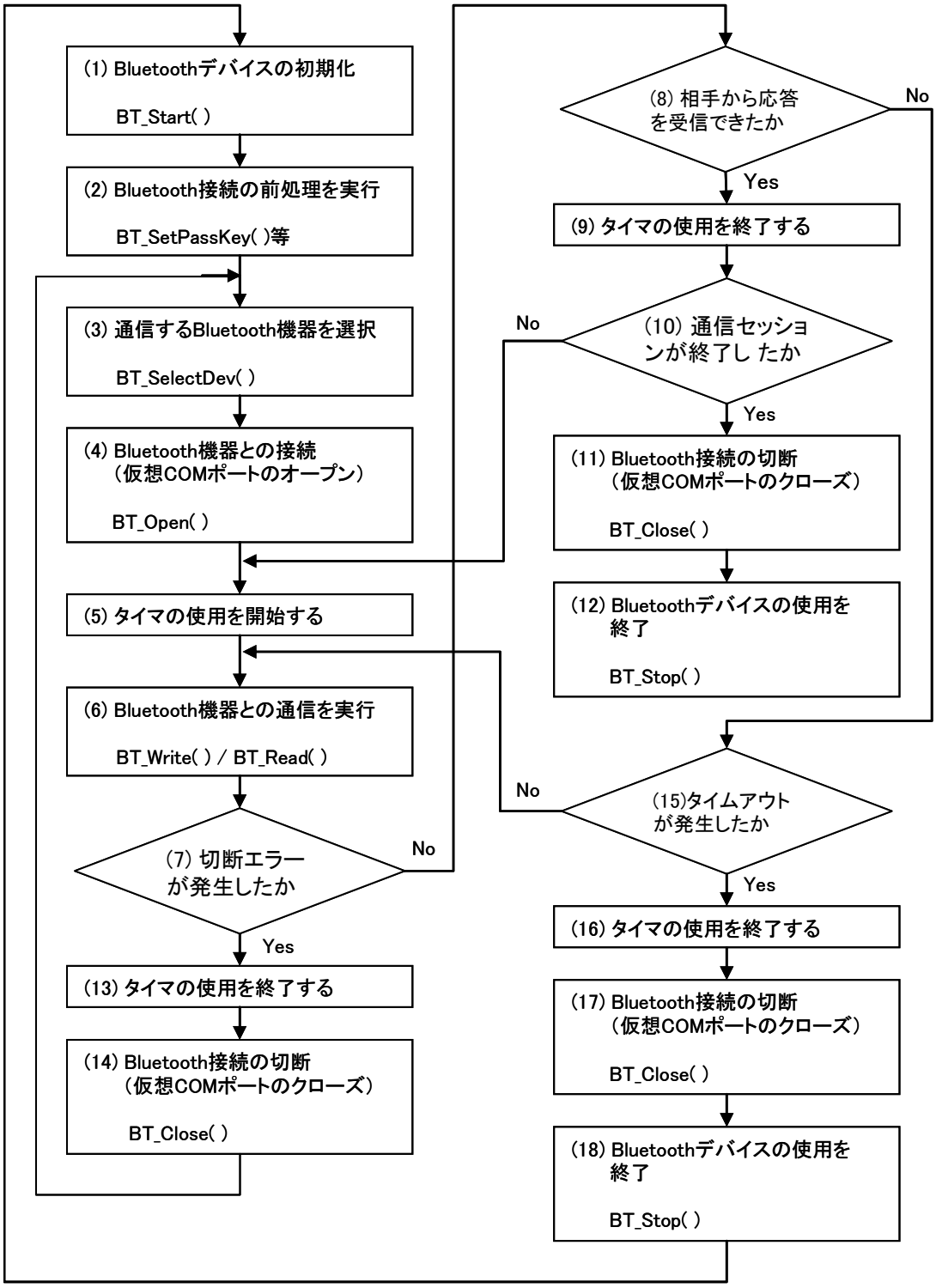
※ 時間が空いてしまう場合は、BT_Open→通信→BT_Close の部分を繰り返す事により BT_Start()および BT_Stop()にかかる時間を節約できます。ただし、相手から切断された場合、電源 OFF した場合等、関数からエラーが戻ってきた場合や、次項に示すようにプリンタからの応答がない場合には、所定のエラー処理を行ってください。

(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	・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	・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	・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	・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	・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_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	・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_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	・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_SetLocalInfo	
BT_Inquiry	
BT_GetDevInfo	
BT_GetDevName	
BT_SetPassKey	
BT_SelectDev	
BT_Open	
BT_SaveDevInfo	
BT_LoadDevInfo	
BT_Close	・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
BT_Read	
BT_Write	
BT_QueryRx	

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	・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_SetLocalInfo	
BT_Inquiry	
BT_GetDevInfo	
BT_GetDevName	
BT_SetPassKey	
BT_SelectDev	
BT_Open	
BT_SaveDevInfo	
BT_LoadDevInfo	
BT_Close	・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
BT_Read	
BT_Write	
BT_QueryRx	

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_Start 関数を実行してから、他の関数を実行してください
BT_GetLocalInfo	
BT_SetLocalInfo	
BT_Inquiry	
BT_GetDevInfo	
BT_GetDevName	
BT_SetPassKey	
BT_SelectDev	
BT_Open	
BT_Close	
BT_Read	
BT_Write	
BT_QueryRx	
BT_SaveDevInfo	
BT_LoadDevInfo	

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	・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	

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	・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 BT_SetPassKey BT_SelectDev BT_Open BT_Close	・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_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 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_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 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 ・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

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。

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_GetLocalInfo

DT-930 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
【ストラクト構造】

```
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。

16.4.4 BT_SetLocalInfo

DT-930 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
【ストラクト構造】

```
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
                         (値は変更できません)
} 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。
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。

16.4.5 BT_Inquiry

DT-930 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。

16.4.6 BT_GetDevInfo

DT-930 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

number
BT_Inquiry 関数を実行して発見された Bluetooth 機器の数(1～9)

【ストラクト構造】

```
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;
```

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  */
```

16.4.7 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

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)

【ストラクト構造】

```
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;
```

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。

16.4.8 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。

16.4.9 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

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

【ストラクト構造】

```
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;
```

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。

16.4.10 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。

16.4.11 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。

16.4.12 BT_Read

Bluetooth 通信において、受信データの読込を実行します。

```
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
読込データ数(受信バッファから読込できたバイト数)

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。
タイムアウトの仕様については検討中です(必要があればタイムアウト設定用の別関数を追加)。

16.4.13 BT_Write

Bluetooth 通信において、送信データの書込を実行します。

```
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(送信バッファに書込むバイト数)

PutSize
書込データ数(送信バッファに書込できたバイト数)

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。
タイムアウトの仕様については検討中です(必要があればタイムアウト設定用の別関数を追加)。

16.4.14 BT_QueryRx

受信バッファより読込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
読込可能なデータ数(バイト)

戻り値

E_BTOK　　:正常終了
E_BTNG　　:異常終了

16.4.15 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

filename
保存先のファイル名を格納している変数のポインタ

【ストラクト構造】

```
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;
```

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。

16.4.16 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

filename
保存元のファイル名を格納している変数のポインタ

【ストラクト構造】

```
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;
```

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。

16.4.17 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

解説

エラーコードについては、「16.3 エラー詳細(p.355)」を参照してください。

17.通信ユーティリティ制御

マルチドロップ/FLINK/DT500 プロトコル共通

関数	機能概要
cu_stopKeySet	中断キーの設定
cu_setDrive	転送ドライブの指定

マルチドロッププロトコル

関数	機能概要
cu_open	回線のオープン
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_readDlRjInfo	データリンク拒否情報の取得

FLINK プロトコル

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	IO ボックス情報の設定
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

DT500 プロトコル

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_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

関数名は、プロトコルごとに同一のものが存在しますが、(cu_open、cu_fileSend、cu_fileRecv 等)関数機能・インタフェースはプロトコルによって異なります。
FLINK プロトコルを FIR モードで使用する場合は、Ir モード設定関数(Ir_SetWinMode 関数)で FIR モードに切替えてから、以下に説明する cu_xxx 関数を使用してください。
使用するプロトコル別に提供されるヘッダーファイルをインクルードする必要があります。

17.1 通信インタフェース

17.1.1 使用形態

DT-930 では、赤外線による回線ポートが存在します。
プロトコル別による使用形態は次の通りです。

プロトコル	転送方向 (DT-930 を基準)	対象ファイル	COM0(赤外線) 接続機器			備考
			I/O BOX *1	DT-930	IRDA ｱﾀﾞﾌﾟﾀ	
マルチドロップ	受信	ｱﾌﾟﾘｹｰｼｮﾝ	◎	×	×	＊2
		データファイル	◎	×	×	
	送信	ｱﾌﾟﾘｹｰｼｮﾝ	×	×	×	
		データファイル	◎	×	×	
FLINK	受信	ｱﾌﾟﾘｹｰｼｮﾝ	◎	○	○	
		データファイル	◎	○	○	
	送信	ｱﾌﾟﾘｹｰｼｮﾝ	◎	○	○	
		データファイル	◎	○	○	
DT500	受信	ｱﾌﾟﾘｹｰｼｮﾝ	◎	×	×	＊3
		データファイル	◎	×	×	＊3
	送信	ｱﾌﾟﾘｹｰｼｮﾝ	◎	×	×	＊3
		データファイル	◎	×	×	＊3

◎使用可能(推奨)　○使用可能　×使用不可

※ 1 IO ボックスインタフェースについては次の章を参照して下さい。
※ 2 マルチドロップでは AP 上から AP の受信は行えません。
※ 3 DT500 では、規定フォーマットのテキストファイルのみ転送可能です。
※　それ以外のファイルは規定フォーマットへの変換が必要です。

17.1.2 IO ボックスインタフェース

通信ユーティリティでは、各プロトコルとも赤外線コネクタによる通信をサポートしていますが、使用する IO ボックスは次の組み合わせになります。

プロトコル	DT-930 用 IO ボックス	
	サテライト/USBIO(IrDA)	ベーシック IO(カシオ IR)
マルチドロップ	×	○
FLINK	○	×
DT500	×	○

○ 使用可能　× 使用不可

17.1.3 排他制御

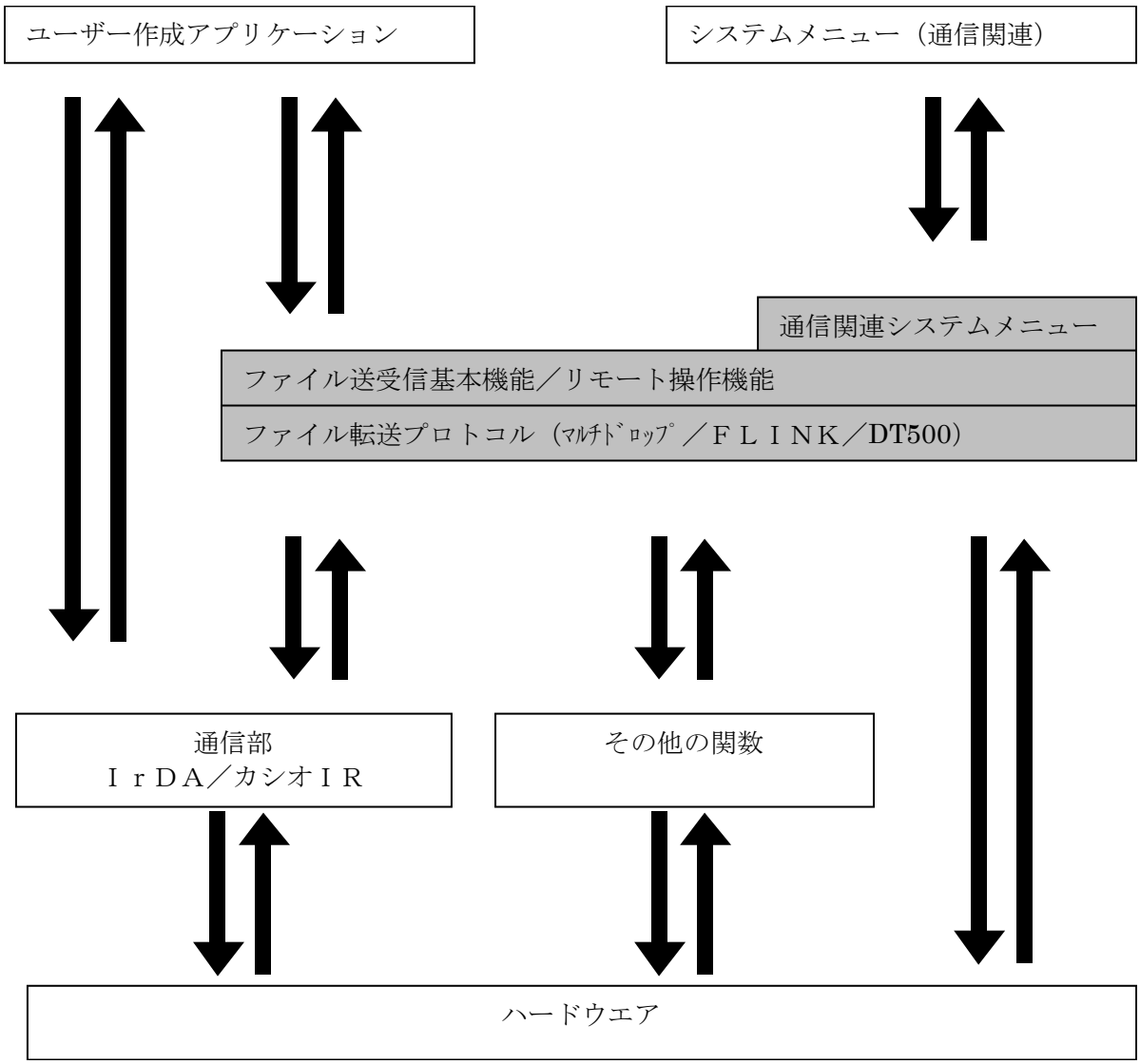
複数プロトコルは同時使用できません。

17.1.4 転送ドライブ

各プロトコルでは、Aドライブ(RAMドライブ)、Bドライブともアクセスが可能です。
ただし、マルチドロップおよびDT500プロトコルに関しては、送信受信を行う前に転送ドライブの指定が必要です。

17.2 ソフトウエアブロック構成図

通信ユーティリティを中心としたソフトウエアブロック構成を示します。(網掛け部分が当通信ユーティリティを表します)
ユーザー作成アプリケーションは当通信ユーティリティのファイル送受信基本機能を用いるか、IrDA 通信、その他の関数を使用する事で作成できます。



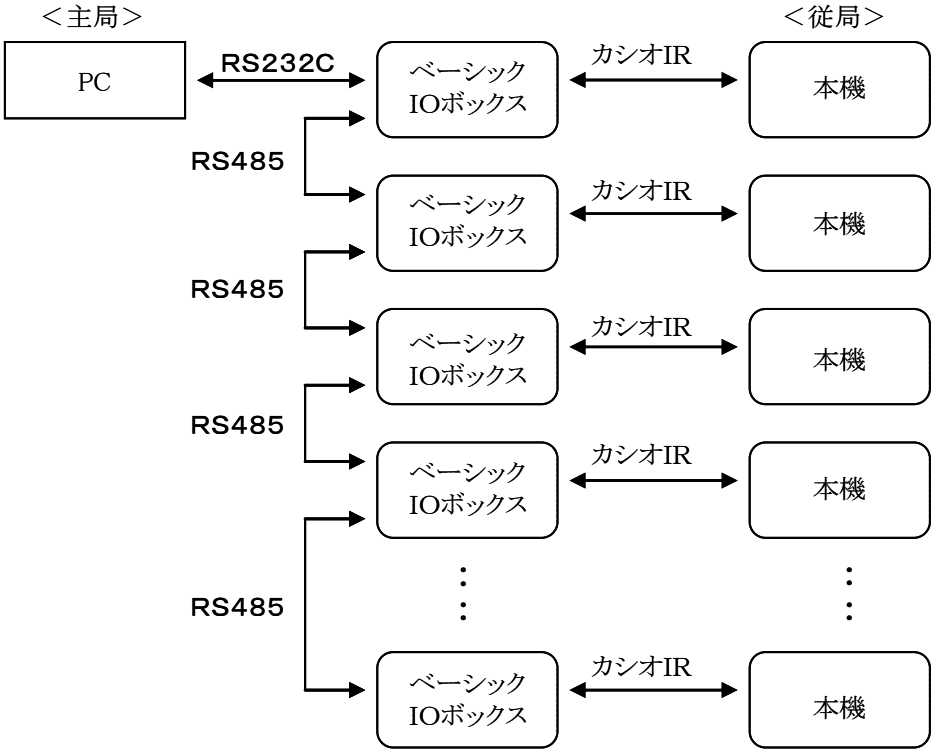
17.3 マルチドロップ

17.3.1 通信仕様

(1) 通信構成

本プロトコルでは主局がホスト PC に、従局が DT-930 になります。
本プロトコルでは、次のファイル転送機能を提供します。

【ベーシック IO ボックス経由での赤外線通信】
ベーシック IO ボックスの連鎖接続によるファイル転送が可能です。
尚、マスタ,サテライト IO ボックスによる通信は行えません。



(2) 通信パラメータ

	関数	システムメニュー用通信機能	備考
共通ﾊﾟﾗﾒｰﾀ			
データ長	8ﾋﾞｯﾄ固定		
パリティビット	選択可(奇数/偶数/なし)	システム環境設定(データ管理部)より取得	
ストップビット	選択可(1ﾋﾞｯﾄ/2ﾋﾞｯﾄ)	システム環境設定(データ管理部)より取得	
通常受信タイムアウト ＊1	システム環境設定(データ管理部)より取得 (00 - 99 秒)		推奨値 3 秒
通常リトライ回数 ＊2	システム環境設定(データ管理部)より取得 (00 - 99 回)		推奨値 3 回
ﾎﾟｰﾘﾝｸﾞ受信中タイムアウト ＊3	3 秒固定		＊5
データリンク待ちタイムアウト ＊4	10 分固定		
通信速度			
COM0(カシオ IR)	選択可(2400～115200bps)	システム環境設定(データ管理部)より取得	
フロー制御	なし		

※ 1 通常受信タイムアウト
通常の通信中における受信のタイムアウト時間。

※ 2 通常リトライ回数
通常の通信中における送受信のリトライ回数。

※ 3 ポーリング受信中タイムアウト
ポーリングに対する処理コマンド送信後、レスポンス待ちタイムアウト時間。

※ 4 データリンク待ちタイムアウト
相手からのポーリング待ちでのタイムアウト時間。

※ 5 ポーリングに要する時間は通信速度や回線品質により異なります。
PC 側ポーリングタイムアウト設定値(HOSTPC.INI)は、下表の値を参考として動作状況により若干多めに指定します。

通信速度	ポーリングタイムアウト設定時間(最小値)
38400 bps 以上	30 ms
19200 bps	30 ms
9600 bps	40 ms
4800 bps	40 ms
2400 bps	50 ms
1200 bps	60 ms

(3) ファイル受信(主局から従局へのファイル転送)

主局のファイルを従局へ送信する。次の特徴があります。

- 従局は端末 ID にて識別され、特定の従局にファイルを送信することができます。
- 複数ファイルを 1 回のデータリンクにて送信できます。
  尚、受信したファイルの書込処理で異常が発生した場合には、従局での通信は異常終了します。
- テキスト/バイナリファイルのどちらも送信可能です。
- 従局は拡張子が".MOT"であるファイルを受信すると、ファイルはアプリケーションプログラムであると解釈し、アプリケーションプログラム領域にメモリ展開します。
- 従局はファイル名が"CONFIG.HTS"であるファイルを受信すると、システム環境ファイルであると解釈し、受信後その内容をシステム環境に反映させます。※
- 受信する局に表示するメッセージデータを転送できます。

※ 確立されているデータリンクが解放される直前に反映します。
例えば、1 回のデータリンクでファイル"CONFIG.HTS"に引き続き、"EXAMPLE.DAT"を転送する間は変化せず、両ファイルの転送後に、新しく転送された"CONFIG.HTS"の内容が反映されます。

(4) ファイル送信(従局から主局へのファイル転送)

従局のファイルを主局へ送信します。

- 複数ファイルを 1 回のデータリンクにて送信できます。尚、受信したファイルの書込処理で異常が発生した場合には、従局での通信は異常終了します。
- テキスト/バイナリデータのどちらも送信できます。ただし、アプリケーション領域に存在するアプリケーションプログラムの送信は行えません。
- 受信する局に表示するメッセージデータを転送できます。

(5) ファイル種別

ファイル転送を行う際には、利用者側でファイル種別を指定することができます、送信側と受信側でファイルの整合性をプロトコルレベルでチェックできます。
ファイル送受信においては従局側でファイル種別を指定します。
主局側にて、そのファイル種別を参照し、その値により転送/未転送を決めることが可能となります。

17.3.2 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。

(1) 機能一覧

機能	内容
通信ポートの初期化	・回線ポートの初期化を行います 回線ポートは COM0:カシオ IR インタフェース ・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が、オープン後は回線クローズでのみ APO の復旧を行いますので、エラー発生後は回線をクローズする必要があります
通信ポートのクローズ	・回線ポートのクローズを行います ・回線オープン時にセーブした APO 設定を復旧します
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
エラー詳細情報の取得	・エラー情報の取得を行います 通信ユーティリティエラー、関数エラーの取得が可能です
データリンク拒否情報の取得	・主局からデータリンクの拒否が行われた場合(エラー詳細情報のエラー状態がデータリンクエラー(主局拒否)の場合)、拒否理由値の取得が可能です

(2) ファイル送受信関数

主局とのファイル転送(送信、受信)を行うための関数です。
尚、ファイル転送時、DT-930 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。

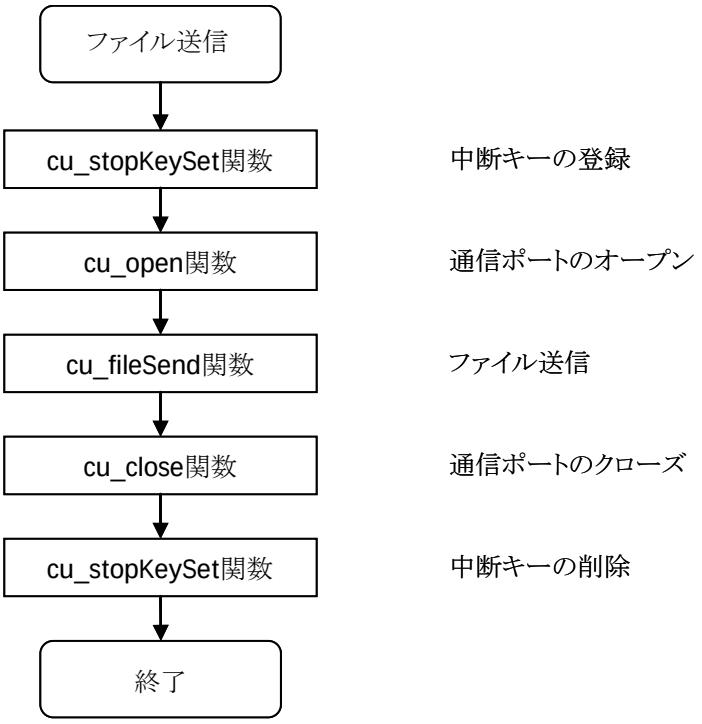
```
0123456789012345
CONFIG. HTS          ← 転送ファイル名(グラフ表示指定行)
      XXX%           ← 進捗のパーセンテージ表示
|*****・・・・・|     ← 進捗のグラフ表示(10%単位で"・"が"＊"に変わります)
```

※ 実際の表示位置は画面モードにより異なります。

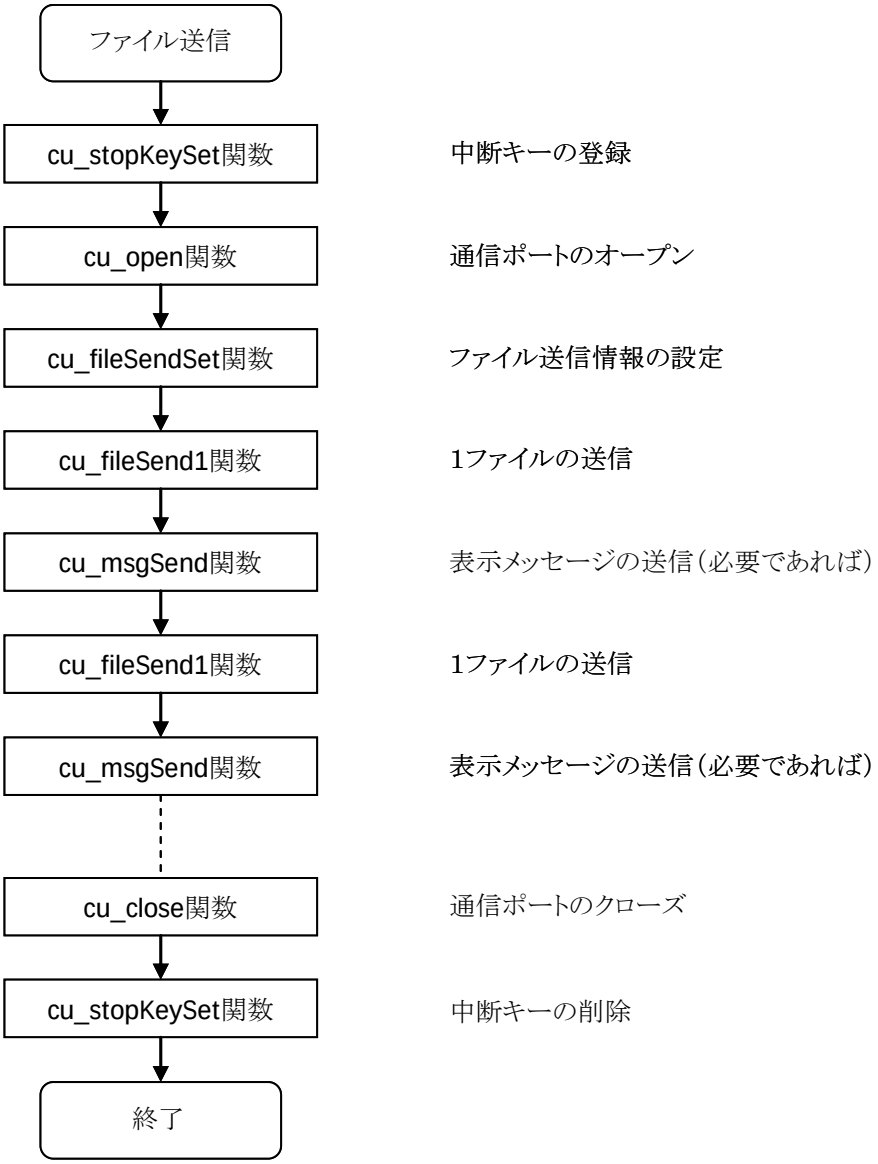
1. ファイル送信関数

従局から主局へファイルの転送を行います。
ファイル送信には、ファイル送信関数により、一括して送信する方法とファイル送信情報の設定を行った後、1 ファイル送信関数により 1 ファイルずつ送信する方法とがあります。

[ファイル送信関数によるファイル送信]
ファイル送信はファイル送信関数にて一括して送信できます。
この場合、通信中に画面表示メッセージを送信することはできません、ファイル一括送信の手順は次の通りです。

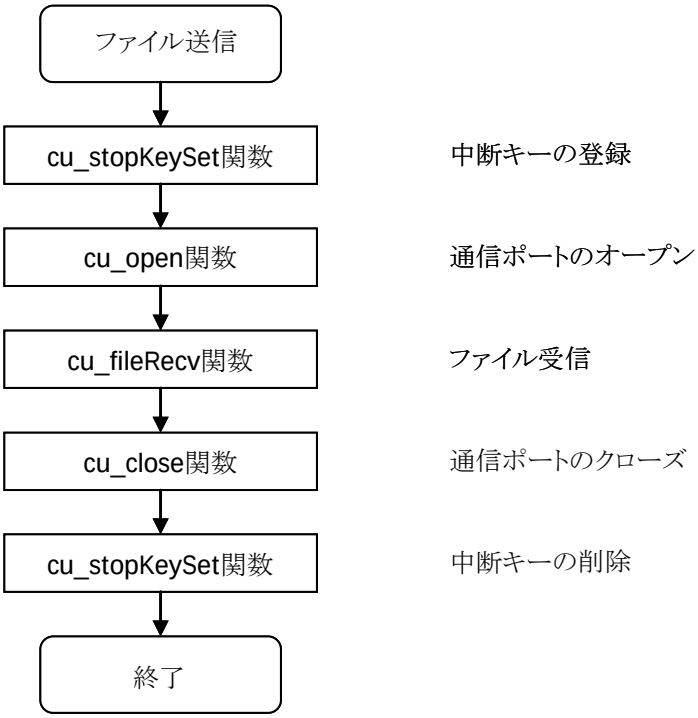


[1 ファイル送信関数によるファイル送信]
ファイル送信中に画面表示メッセージを送信する場合や通信を中断する場合には、ファイル送信情報の設定および 1 ファイル送信関数を用いて 1 ファイルずつ送信するようにし、その中で画面表示メッセージの送信や通信の中断を行います。
画面表示メッセージ送信の手順は次の通りです。



2. ファイル受信関数

主局から従局へファイルの転送を行います。
ファイル受信はファイル受信関数にて、複数ファイルを一括して受信できます。
ファイル受信の手順は次の通りです。



17.4 FLINK プロトコル機能

17.4.1 通信仕様

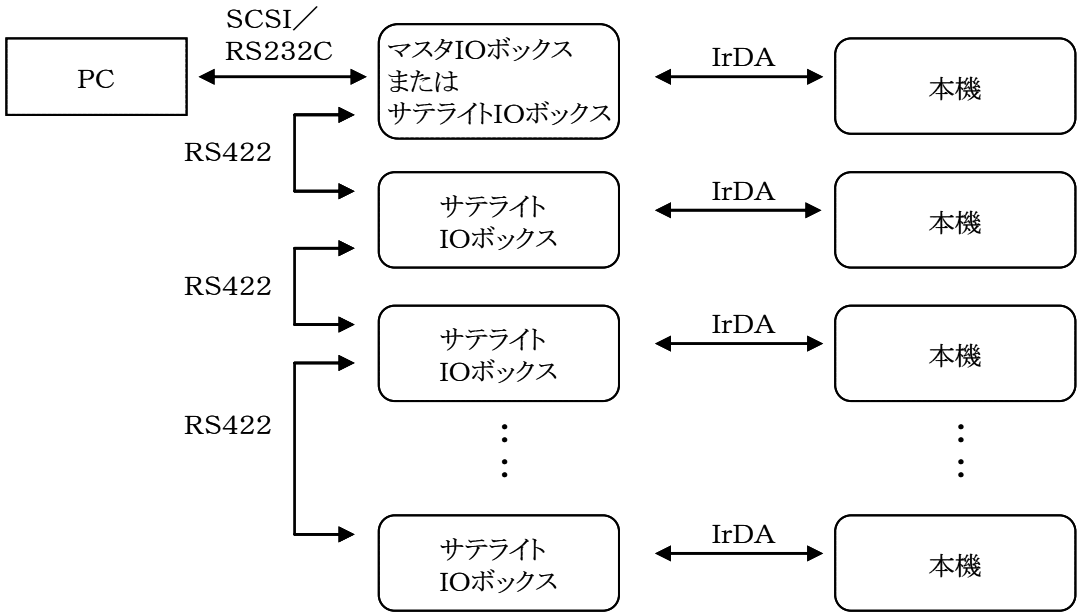
(1) 通信構成

本プロトコルのファイル転送機能は次の通りです。

1. サテライト IO ボックス経由でのホスト通信

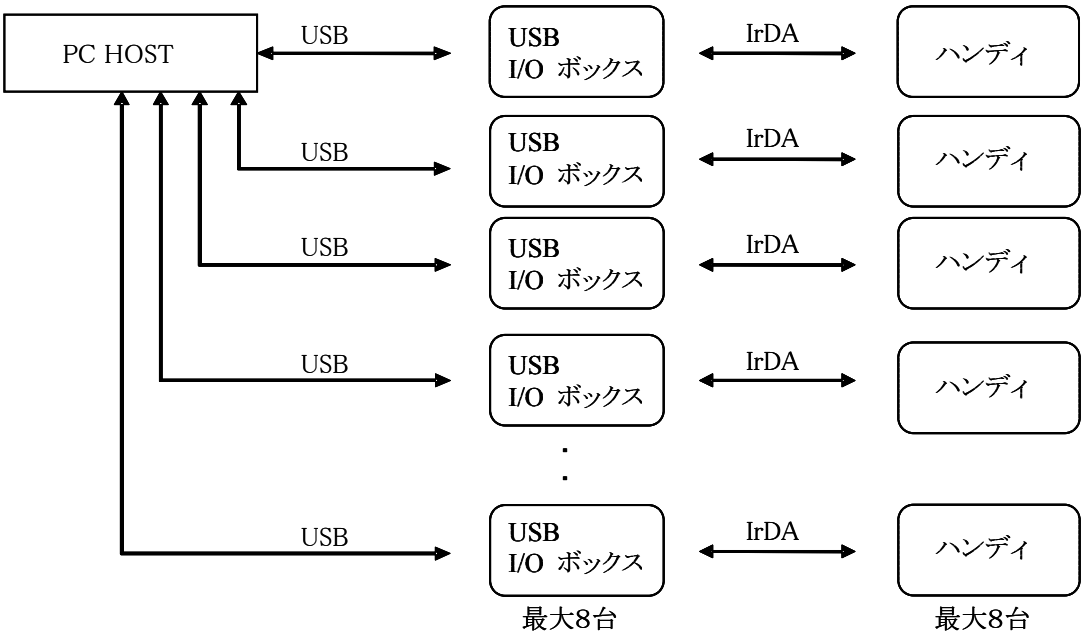
   サテライト IO ボックスの連鎖接続によるファイル転送ができます。

   尚、ベーシック IO ボックスによる通信は行えません。



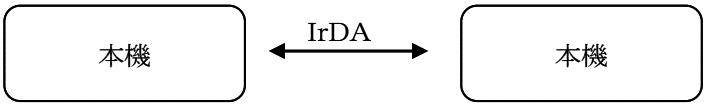
2. USB IO ボックス経由でのホスト通信

USB IO ボックスによるファイル転送ができます。



3. 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)
共通ﾊﾟﾗﾒｰﾀ		
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、4Mbps)	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。

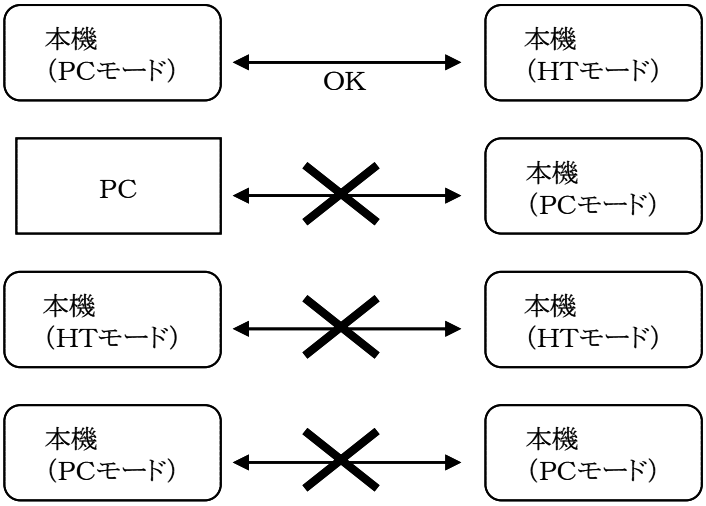
※ 2 受信待ちタイムアウト時間
コマンド/レスポンス受信待ち時間を監視します。
設定値 0 はタイムアウトなしです。

※ 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-930 間通信時および本体-本体間通信時に、どちらか一方の DT-930 が選択されます。
2. PC モード
   セッション確立後、DT-930 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-930 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-930 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-930 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-930 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-930 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ﾎﾟﾗﾘﾌｧｲﾙ(FL.ADD)に受信後、ファイルを追加します。追加完了後、ﾃﾝﾎﾟﾗﾘﾌｧｲﾙを削除します	ﾃﾝﾎﾟ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/ﾃﾞｨﾚｸﾄﾘを削除します	削除したファイル/ﾃﾞ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	作成したﾃﾞ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	ｺﾏﾝﾄﾞを送信後、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	表示後のﾒｯｾｰｼﾞはクリアしません
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	ｺﾏﾝﾄﾞ送信後、通信を終了します	通信を終了します	―――

17.4.2 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。

(1) 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。

① 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_B2400(2400bps)/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-930 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-930 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

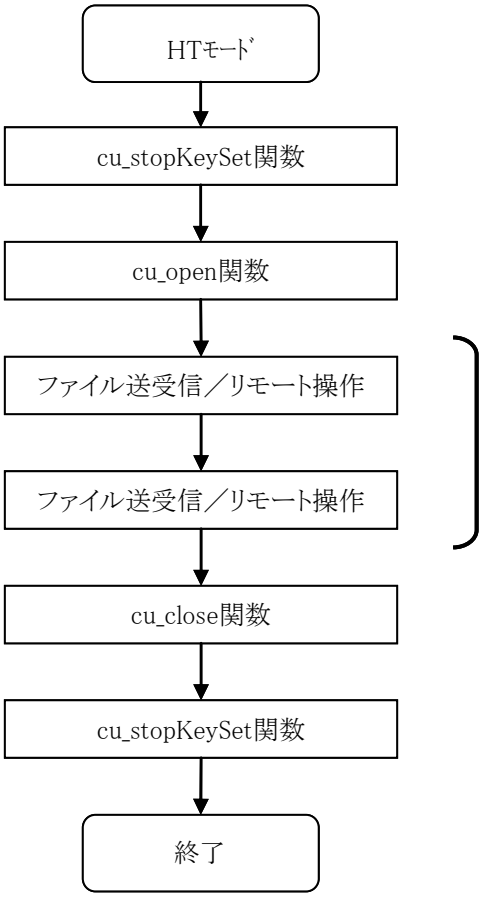
中断キーの選択登録/削除を行います。
登録は回線オープンの前に、削除はクローズの後に行う事が必要です。

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

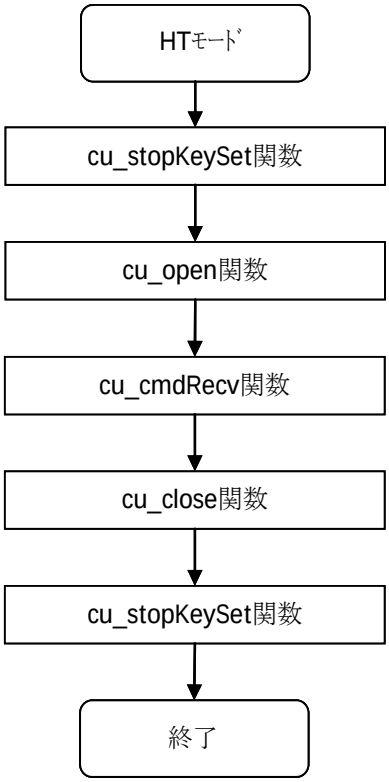
通信ポートのオープン／セッション確立

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

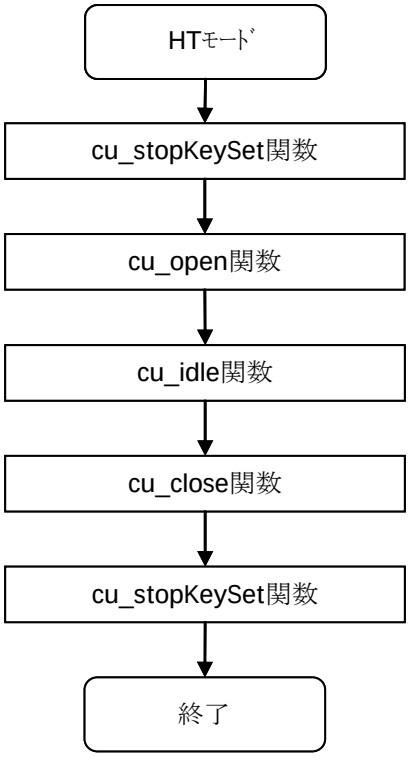
HTからのコマンド待ち
（HTからの終了指示、またはIDLE通知を
受信するまで、受信コマンドを順次実行します。）

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

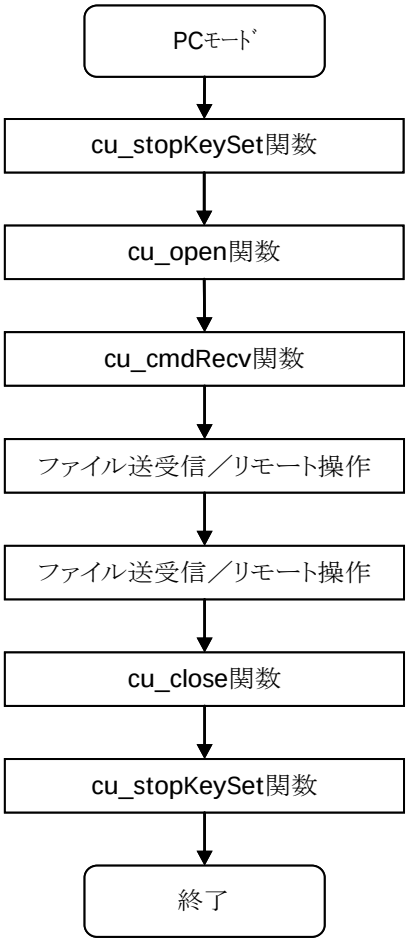
通信ポートのオープン／セッション確立

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。）

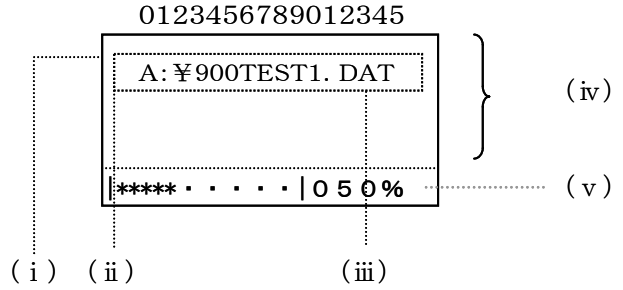
ファイル送受信／リモート操作関数
を順次実行します。

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。

① 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]	[送信先ﾃﾞｨﾚｸﾄﾘ名]
“A:¥SEND¥AAA.DAT”	“B:¥RECV¥”

(送信側ﾃﾞｨﾚｸﾄﾘ構成)

```
A:¥--SEND¥--SUB1¥---AAA.DAT        B:¥---RECV¥---SUB1¥---AAA.DAT
          |-----SUB2¥----BBB.DAT                |----AAA.DAT
          |-----AAA.DAT
          |-----BBB.DAT
```

(c)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来ます。

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)

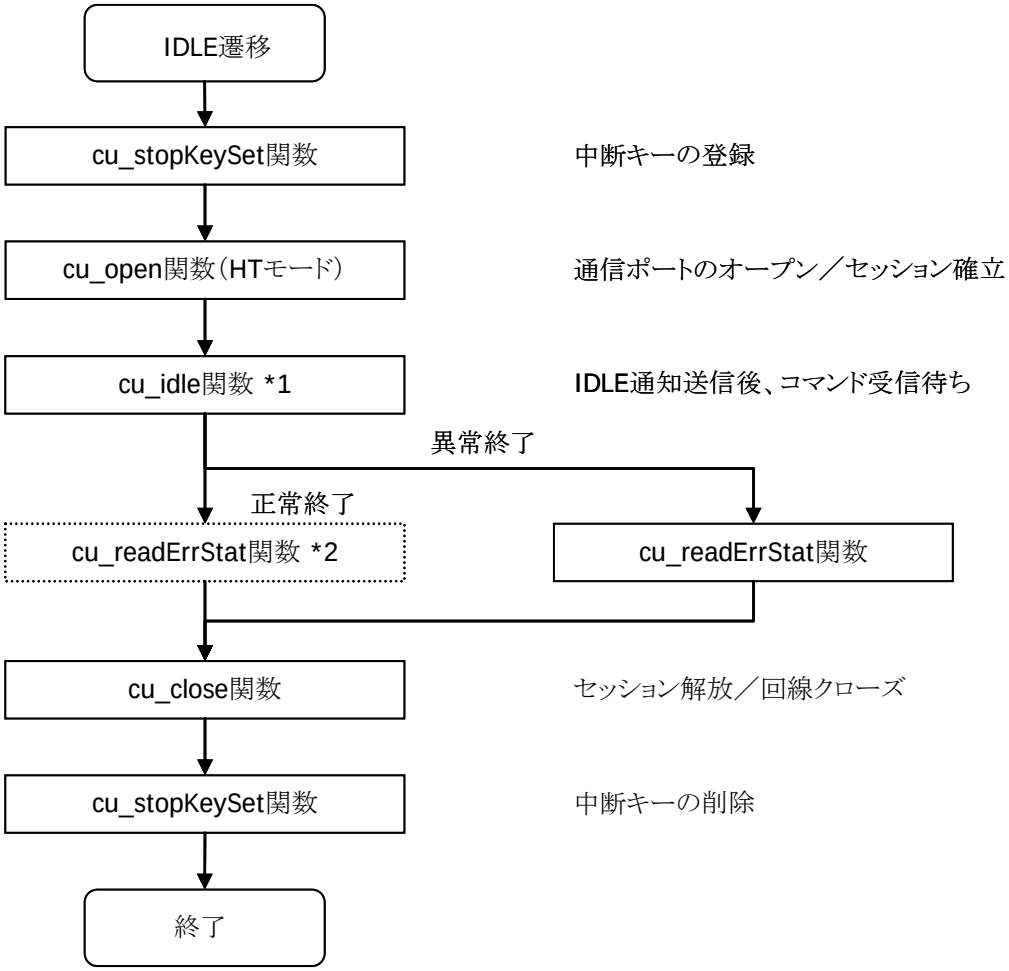
複数ファイルの受信を一括して行うことができます。
受信ファイルに対して次のオプションを選択することができます。

(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。
(c)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]

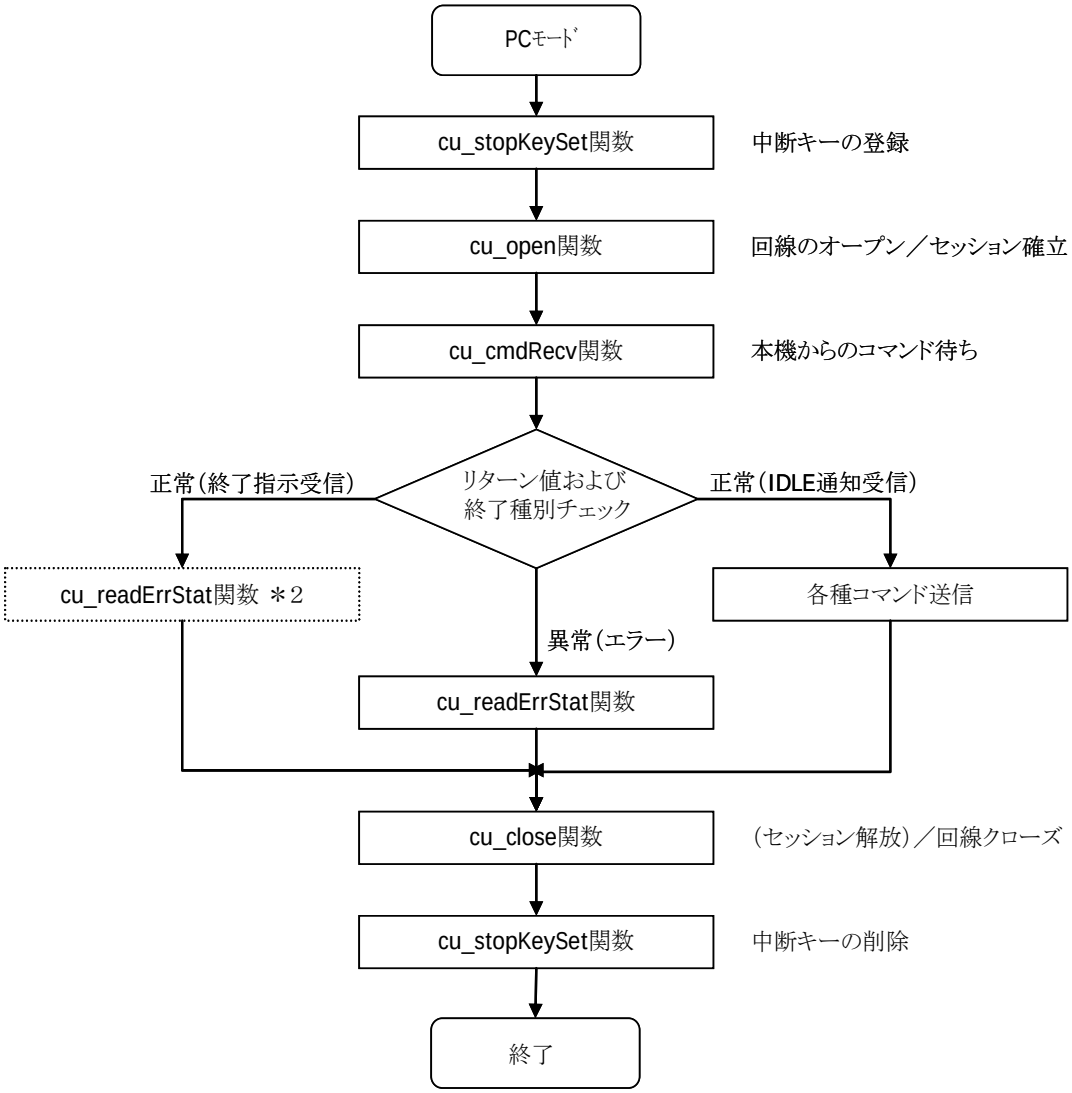


※ 1 相手局からの終了指示コマンド受信またはエラー発生まで、受信コマンドを順次実行します。
※ 2 必要に応じて相手局からの終了指示コマンド詳細情報(ﾌｫｰﾏｯﾄ指示、ﾘｾｯﾄ指示等)の取得が可能です。

⑤ PC モードコマンド待ち(cu_cmdRecv)

DT-930 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-930 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。

17.4.3 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。

(1) 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。

① 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。

① 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-930/PC(AT 互換機)/PC(98 シリーズ))
OS モデル情報(DT-930 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。

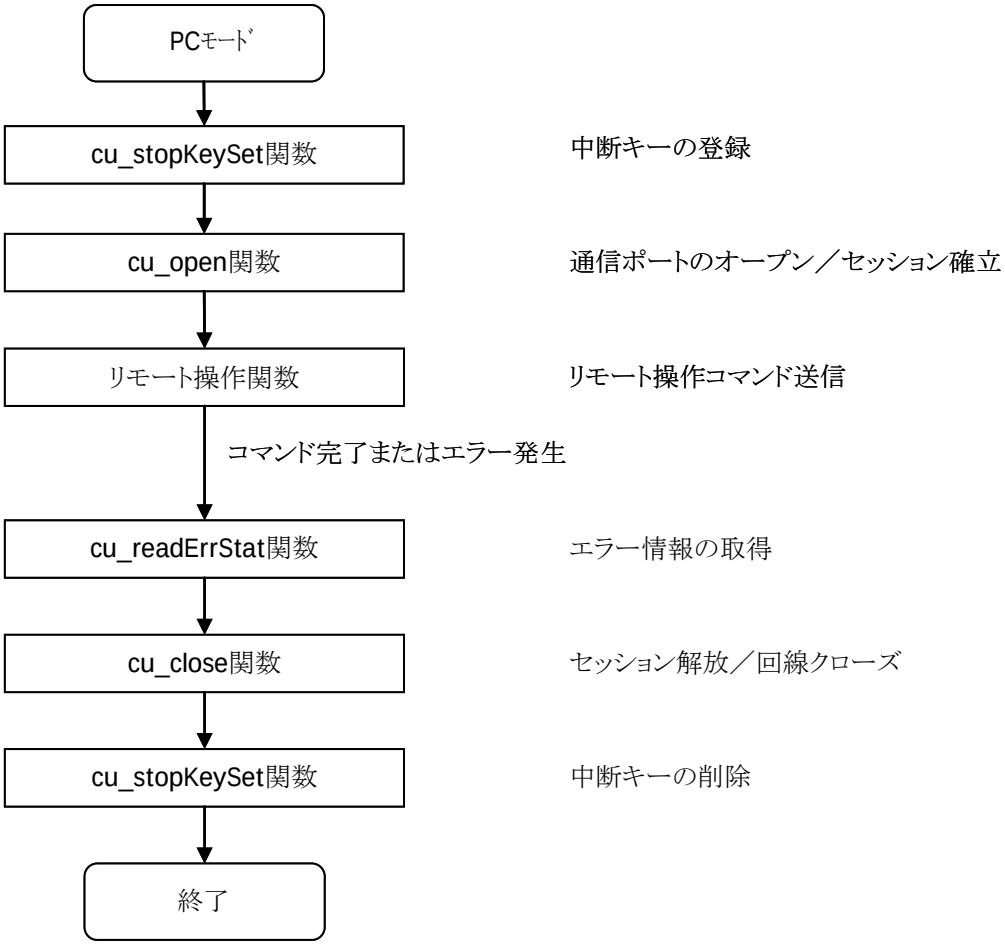
④ 画面表示メッセージの送信(cu_msgSend)

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-930 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]



17.4.4 ファイルチェック機能関数

(1) 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。

17.5 DT500 プロトコル機能

17.5.1 通信仕様

(1) 通信構成

本プロトコルは DT500 プロトコルをサポートした機器(PC)との 1 対 1 のファイル転送を想定しています。
DT500 プロトコルでは、ファイル送信側を主局、受信側を従局と呼び、ファイルの転送方向により主局・従局は入れ替わります。
本プロトコルでは、次の構成でのファイル転送機能を提供します。

IO ボックス経由での赤外線通信
IO ボックスを使用して 1 対 1 でのファイル転送が可能です。(連鎖接続はできません)
IO ボックスはベーシック IO のみ使用可能です。(サテライト IO ボックスは使用できません)

(2) 通信パラメータ

	関数	システムメニュー用通信機能
共通パラメータ		
リンク確立時タイムアウト時間(＊1)	システム環境設定(データ管理部)より取得 (30 ～ 240 秒)	
COM0(赤外線　カシオ IR)		
最大通信速度	選択可 (2400-115200bps)	システム環境設定(データ管理部)より取得

通信速度に関して
DT500 プロトコルとして提供されているダウンロードプログラムは最大 38400bps の対応ですが、プロトコルが公開されているため、ユーザ独自のユーティリティ作成による運用を想定し、本関数レベルで最大 115200bps をサポートします。

リンク確立時タイムアウト時間
・ファイル受信のタイムアウト時間およびファイル送信時の ENQ 再送回数。

設定値	ファイル受信タイムアウト値	ファイル送信 ENQ 再送回数
1	30 秒	10 回
2	60 秒	20 回
3	90 秒	30 回
4	120 秒	40 回
5	150 秒	50 回
6	180 秒	60 回
7	210 秒	70 回
8	240 秒	80 回
9	タイムアウトなし	再送オーバーなし

(3) 転送ファイル

本プロトコルでは、DT500 プロトコル用のテキストファイルの転送を行います。

① DT500 ファイル形式データファイル

- フィールド長は 1～254 バイトまで有効です。
- PC 側 windows 版転送ユーティリティのバイナリ転送でも転送可能です。
- PC から受信したファイルをそのままの形式で指定ドライブに格納します。(※)

②DT500 ファイル形式ユーザプログラムファイル(拡張子、"PD3"、"EX3"、"FN3")

- DT-930 でプログラムとして実行はできませんが、ファイルとして転送が可能です。
- PC から受信したファイルをそのままの形式で指定ドライブに格納します。(※)

③DT-900 システム転送用変換ファイル(拡張子、"DTF")

- DT-930 用アプリケーション・パッチファイルを本プロトコル転送用に変換したファイル。
- システムメニューでのみ転送が可能です。
- PC 側 windows 版転送ユーティリティのバイナリ転送でも転送ができます。

※ DT500 では DT-BASIC 形式ファイルへの変換等を行ないますが、DT-930 では一切行ないません。

(4) プロトコルオプション

① シリアル番号

シリアル番号は 5 桁の 10 進数で、伝送ブロックの先頭にあるテキスト制御文字のすぐ後につけられます。
5 桁に満たないときは、上位桁に 0(ゼロ)がはいります。

② 水平パリティチェック

ブロックチェックキャラクタ(BCC)は、伝送ブロックの末尾にあるターミネータのすぐ後につけられます。
水平パリティは、伝送ブロックにヘッダ(SOH や STX)を除いた部分について計算されます。

(5) ダウンロード(PC から HT へのファイル送信)

ホスト PC のファイルを HT へ送信します。次の特徴があります。

- 非透過モードのため、規定フォーマットのテキストファイルのみ転送可能です。
- 受信データファイルは、指定ドライブに格納されます。
- HT 上に進捗グラフを表示することが可能です。

(6)アップロード(HT から PC へのファイル送信)

HT のファイルを PC へ送信する。次の特徴があります。

- 非透過モードにより、規定フォーマットのテキストファイルのみ転送可能です。
- HT 上に進捗グラフを表示することが可能です。

17.5.2 ファイル送受信基本機能

(1) 通信基本関数

当機能は DT500 プロトコルでの通信基本関数を提供します。

① 通信ポートの初期化
回線ポートの初期化を行う。回線ポートはカシオオリジナル IR です。
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 禁止設定が行われませんが、オープン後は回線クローズでのみ APO の復旧を行います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
回線ポートのクローズを行います。
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。

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、通信関数状態等の取得が可能です。

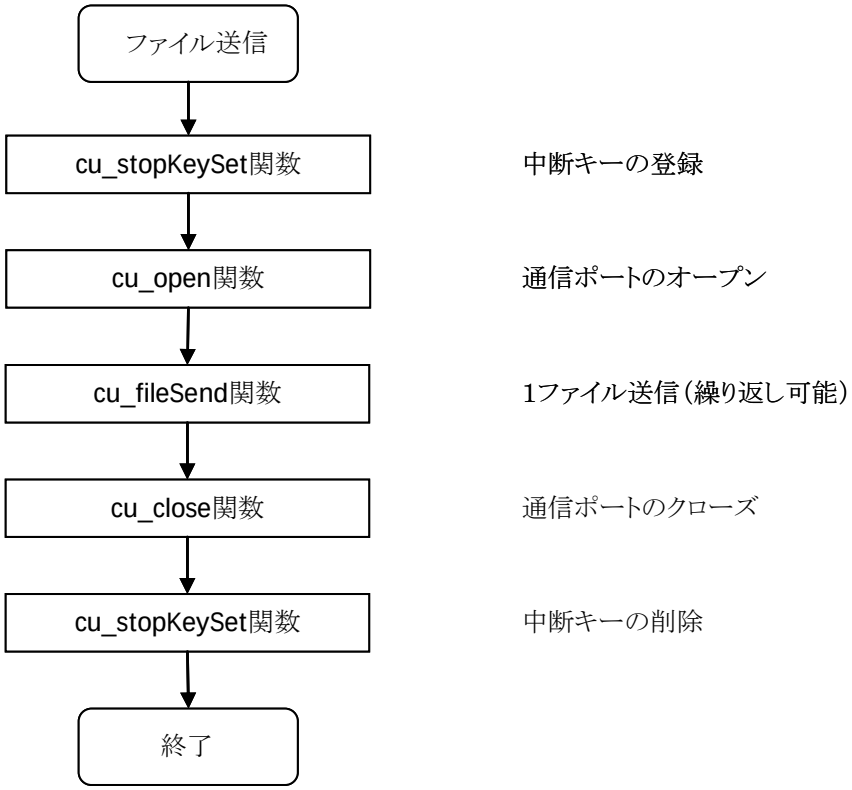
(2) ファイル送受信関数

PC とのファイル転送(送信、受信)を行うための関数です。
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できます。
グラフ表示フォーマットは次の通りです。

```
 0123456789012345
┌─────────────────┐
│CONFIG. HTS      │ ← 転送ファイル名（グラフ表示指定行）
│          XXX%   │ ← 進捗のパーセンテージ表示
│|*****・・・・・|│ ← 進捗のグラフ表示（10％単位で“・”が“＊”に変わります）
└─────────────────┘
```

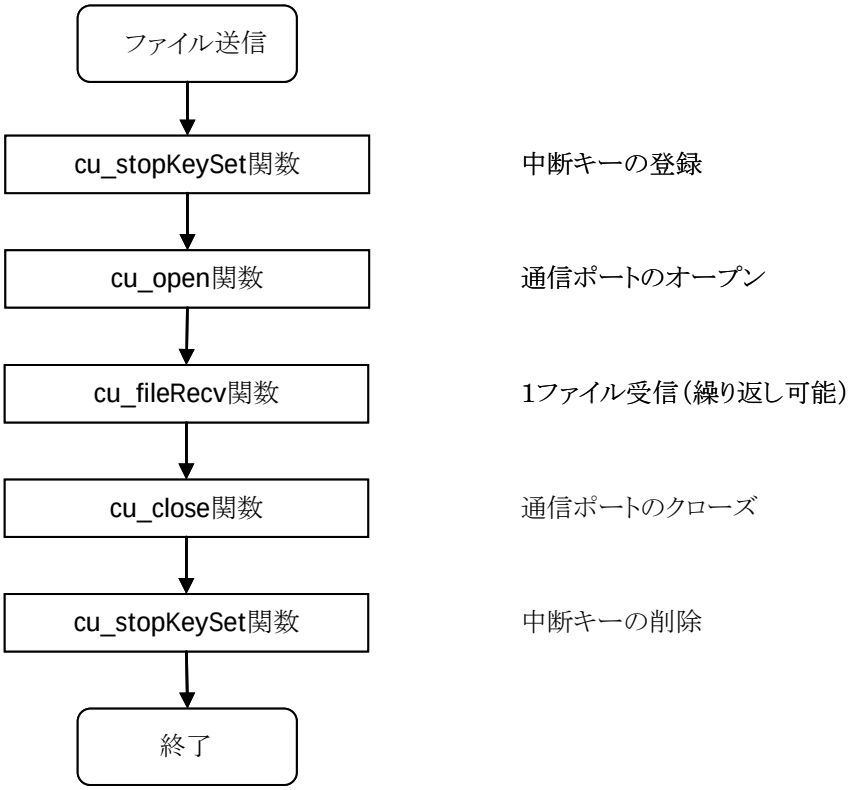
① ファイル送信

HT から PC へ 1 ファイルの転送を行います。
ファイル送信は、ファイル送信関数により、1 ファイルを送信します。
ファイル送信の手順は次の通り。



②ファイル受信

PC から HT へ 1 ファイルの転送を行います。
ファイル受信はファイル受信関数にて、1 ファイルを受信できます。
ファイル受信の手順は次の通りです。



(3) 制御コード関数

拡張機能として、プロトコル制御コードの変更/参照機能を提供します。

① SOH 変更/参照

“ヘッディングテキストの開始”を示す SOH(デフォルト 01h)コードを変更することができます。
現在設定されている SOH コード値を取得することができます。

② STX 変更/参照

“データテキストの開始”を示す STX(デフォルト 02h)コードを変更することができます。
現在設定されている STX コード値を取得することができます。

③ ETX 変更/参照

“テキストの終結”を示す ETX(デフォルト 03h)コードを変更することができます。
現在設定されている ETX コード値を取得することができます。
尚、パソコン用転送ユーティリティは、デフォルト以外はサポートしていません。

17.5.3 解説

DT500 プロトコルを使用する時の注意点を以下に示します。

(1) DT500 との相違点

① データファイル

DT-930 上に同一のファイルが存在した時は上書きします。
受信データは BASIC 形式ファイルへの変換は行なわずそのまま格納します。
フィールド末尾のスペースは削除しません。
システムメニュー上からのバイナリ転送による送信はできません。

②ユーザプログラムファイル

DT-930 では DT500 プログラムは使用できません。
受信データは HEX 形式ファイルへの変換は行なわず、そのまま格納します。

③DT-930 システム関連ファイル

DT-930 アプリケーションファイル・パッチファイル等のシステム関連ファイルは、DT500 プロトコルで転送する時、ファイル変換を行なう必要があります。

(2)AP インストール時の留意点

①アプリケーションファイル・パッチファイル

データコンバータ(dtfilcnv.exe)により、転送用変換ファイル(*.DTF)を作成します。
転送ユーティリティを使用して DT-930 へ送信する時、指定するフィールド長はコンバータで指定したフィールド長を使用します。
アプリケーションファイルは指定ドライブへ転送されるので、ASTART.HTS で指定する事が必用です。

②システム関連ファイル(CONFIG.HTS、CONFIG.ID、CONFIG.PAS、ASTART.HTS)

ファイルの末尾に CR・LF を追加します。ただし、ファイルサイズは 254 バイト以下である必要があります。
転送ユーティリティを使用して DT-930 へ送信する時、指定するフィールド長は次の通りです。

最大フィールド長
windows 版転送ユーティリティ　:254 バイト
DOS 版転送ユーティリティ　　:99 バイト

(a)CR・LF(2 バイト)以外のデータサイズが最大フィールド長に収まる場合
　　CR・LF 以外のデータサイズをフィールド長として指定します。

(b)CR・LF 以外のデータサイズが最大フィールド長を超える場合
　　最大フィールド長以内のサイズにブロックに分割して指定します。

17.6 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。

17.7 共通ファンクション

17.7.1 cu_stopKeySet

【通信ユーティリティ ： 共通ファンクション】
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

17.7.2 cu_setDrive

【通信ユーティリティ ： 共通ファンクション】
マルチドロップおよび DT500 プロトコルにて、ファイル送信・ファイル受信の転送ドライブを指定します。
(FLINK では無効です)

```
ER cu_setDrive(
  UB   drive
)
```

パラメータ

drive
送信・受信ファイルのドライブ
CU_DRIVE_A　　:A ドライブ
CU_DRIVE_B　　:B ドライブ

戻り値

E_OK　　:正常終了
E_PRM　　:パラメータエラー

17.8 マルチドロッププロトコル

17.8.1 cu_open

【通信ユーティリティ ： マルチドロッププロトコル】
通信ポートの初期化およびセッションの確立を行います。

```
ER cu_open(
  H comNo,
  UB connectMode,
  struct sys_tty *param
)
```

パラメータ

comNo
通信ポート
COM0　　カシオ IR インタフェース

connectMode
接続モード
CU_CNCT_MULT　マルチドロップ接続

param
通信パラメータのポインタ
【ストラクト構造】

```
struct sys_tty  {
   W  speed:       : B_1200～B_115200  /* 転送速度      */
   W  length;  : CHAR_8                    /* データ長固定      */
   W  parity;  : PARI_NON              /* パリティﾋﾞｯﾄ     */
               : PARI_ODD
               : PARI_EVN
   W  stop_bit;    : STOP_1                    /* ｽﾄｯﾌﾟﾋﾞｯﾄ        */
               : STOP_2
} *param;
```

戻り値

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

17.8.2 cu_fileSend

【通信ユーティリティ ： マルチドロッププロトコル】
指定された複数ファイルを一括して送信します。
送信の結果は、送信ファイル情報エリアの stat に格納されます。
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,
  UB priority,
  UB fileKind,
  UH filecount,
  CU_FILE_INFO_FORM *fileInfo,
  UB graphFlag,
  UB graphPos
)
```

パラメータ

comNo
通信ポート
COM0　　カシオ IR インタフェース

priority
優先順位(0～255)

fileKind
ファイル種別
カシオ提供の通信ユーティリティを使用する場合、次の設定が必要です。
01h　　*.LOD 以外のファイルを送信
02h　　*.LOD ファイル
03h　　全ファイル

fileCount
送信ファイルの数(1～65535)

fileInfo
転送ファイル情報の配列(fileCount 分)の先頭アドレス

graphFlag
グラフ表示フラグ
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　表示しません

graphPos
グラフ表示行(0～7,graphFlag が CU_GRAPH_OFF 時は参照しません)

戻り値

【ストラクト構造】

```
typedef struct    {
   UB fileName[11];     ：転送ファイル名格納領域
                        例） CONFIG.HTS の場合
                                  C | O | N | F | I | G |  |  | H | T | S
   UB stat;             ：転送結果格納領域（次ページを参照して下さい）
}CU_FILE_INFO_FORM;
```

E_OK	正常終了(stat はすべて CU_STAT_TRANS)
E_NG	異常終了　E_PRM:パラメータエラー

解説

当関数の使用前に cu_open 関数、使用後に cu_close 関数を実行する必要があります。

転送ファイル情報:転送結果格納領域の設定値一覧

値	シンボル	意味
0	CU_STAT_TRANS	正常終了
1	CU_STAT_OPEN_ERR	転送ファイルのオープンエラー
2	CU_STAT_READ_ERR	転送ファイルのリードエラー
3	CU_STAT_WRITE_ERR	転送ファイルのライトエラー
4	CU_STAT_SEND_ERR	転送ファイルの送信側エラー
5	(未使用)	(未使用)
	:	:
255	CU_STAT_PRE_TRANS	転送未処理

17.8.3 cu_fileSendSet

【通信ユーティリティ ： マルチドロッププロトコル】
ファイルの送信に先立ち、送信情報の設定/送信を行います。

```
ER cu_fileSendSet(
  H comNo,
  UB   priority,
  UB   fileKind,
  UH   fileCount,
  W totalTransSize
)
```

パラメータ

comNo
通信ポート
COM0　　カシオ IR インタフェース

priority
優先順位(0～255)

fileKind
ファイル種別
カシオ提供の通信ユーティリティを使用する場合、次の設定が必要です。
01h　　*.LOD 以外のファイルを送信
02h　　*.LOD ファイル
03h　　全ファイル

fileCount
送信ファイルの数(1～65535)

totalTransSize
総転送サイズ　　不定/不明の場合には FFFFFFFFh 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

解説

当関数の使用前に cu_open 関数を実行する必要があります。

17.8.4 cu_fileSend1

【通信ユーティリティ ： マルチドロッププロトコル】
1 ファイルの送信を行います。
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_fileSend1(
  H                  comNo,
  CU_FILE_INFO_FORM  *fileInfo,
  UB                 graphFlag,
  UB                 graphPos
)
```

パラメータ

comNo
通信ポート

COM0	カシオ IR インタフェース

fileInfo
転送ファイル情報のアドレス
(転送ファイル情報はファイル送信関数を参照して下さい)

graphFlag
グラフ表示フラグ

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	表示しません

graphPos
グラフ表示行(0～7,graphFlag が CU_GRAPH_OFF 時は参照しません)

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

解説

当関数の実行に先立ち 1 度、cu_fileSendSet 関数を実行する必要があります。
cu_fileSendSet 関数で総転送サイズを不定/不明とした場合、グラフ表示フラグに関わらず、グラフ表示は行いません。
途中で cu_end 関数を実行するか、相手から中断されない限り、cu_fileSendSet 関数で指定したファイル数分、当関数を実行する必要があります。
全ファイル送信後は、cu_close 関数を実行する必要があります。

17.8.5 cu_fileRecv

【通信ユーティリティ ： マルチドロッププロトコル】
相手より送信される複数ファイルを一括して受信します。
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,
  UB                    priority,
  UB                    fileKind,
  UH                    *fileCount,
  CU_FILE_INFO_FORM     *fileInfo,
  UB                    graphFlag,
  UB                    graphPos
)
```

パラメータ

comNo
通信ポート
COM0　　カシオ IR インタフェース

priority
優先順位(0～255)0～FFh (必須ですが DT-930 では使用していません)

fileKind
ファイル種別
カシオ提供の通信ユーティリティを使用する場合は、03h に設定して下さい。

fileCount
(呼び出し時)fileInfo から始まる CU_FILE_INFO_FORM の配列数
(戻り時)受信したファイルの数

fileInfo
転送ファイル情報の配列(fileCount 分)の先頭アドレス
(転送ファイル情報はファイル送信関数を参照)

graphFlag
グラフ表示フラグ
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　　表示しません

graphPos
グラフ表示行(0～7,graphFlag が CU_GRAPH_OFF 時は参照しません)

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

解説

相手から総転送サイズを不定/不明としてファイル送信された場合には、グラフ表示フラグに関わらず、グラフ表示は行いません。
当関数の使用前に cu_open 関数、使用後に cu_close 関数を実行する必要があります。

17.8.6 cu_msgSend

【通信ユーティリティ ： マルチドロッププロトコル】
画面表示メッセージの送信を行います。

```
ER cu_msgSend(
  H comNo,
  UB   *data
)
```

パラメータ

comNo
通信ポート

COM0	カシオ IR インタフェース

data
メッセージ格納アドレス(メッセージの最後は NULL コードでターミネートする必要があります)

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

解説

当関数を使用する前に必ず cu_fileSend1 関数を使用して下さい。

17.8.7 cu_end

【通信ユーティリティ ： マルチドロッププロトコル】
通信を中断します。

```
ER cu_end(
  H comNo
)
```

パラメータ

comNo
通信ポート
COM0　　カシオ IR インタフェース

戻り値

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

補足

当関数を使用する前に必ず cu_fileSend1 関数を使用して下さい。

17.8.8 cu_close

【通信ユーティリティ ： マルチドロッププロトコル】
通信ポートをクローズします。

```
ER cu_close(
  H comNo
)
```

パラメータ

comNo
通信ポート

COM0	カシオ IR インタフェース

戻り値

関数結果

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

補足

17.8.9 cu_readErrStat

【通信ユーティリティ ： マルチドロッププロトコル】
当ファイル送受信関数でのエラー詳細情報を取得します。
取得後、エラー詳細情報はクリアされます。

```
ER cu_readErrStat(
  H comNo,
  UW *cuStat,
  UW *biosStat
)
```

パラメータ

comNo
通信ポート

COM0	カシオ IR インタフェース

cuStat
通信ユーティリティエラーの情報設定エリアアドレス
（次ページの「エラー詳細情報」参照）

biosStat
エラー発生時の通信関数部システムエラー詳細情報
（次ページの「エラー詳細情報」参照）

戻り値

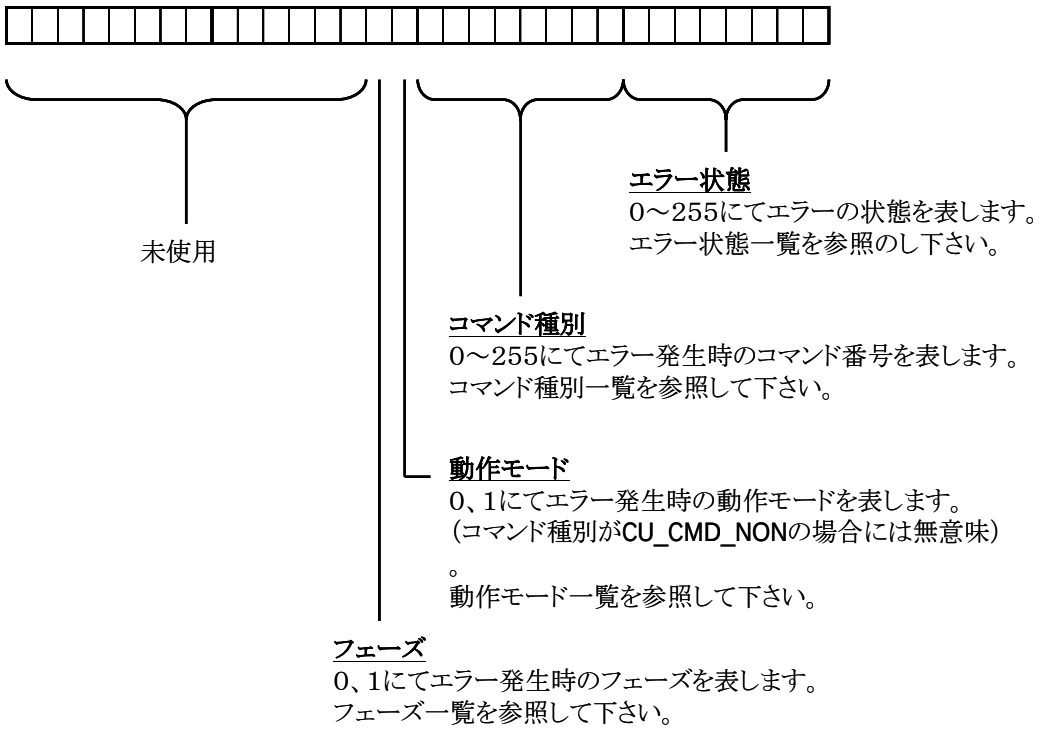
関数結果

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

補足

エラー詳細情報(1)

1. 通信ユーティリティエラー
   通信ユーティリティとしてのエラーを返します。以下のビット構成で通知されます。



フェーズ一覧

値	シンボル	意味
0	CU_PHASE_DATALINK	データリンク以前
1	CU_PHASE_INFO_TRANS	情報転送以降

動作モード一覧

値	シンボル	意味
0	CU_MODE_RES	レスポンス受信時
1	CU_MODE_CMD	コマンド受信時

エラー詳細情報（2）

コマンド種別一覧

値	シンボル	意味
00	CU_CMD_NON	該当コマンドなし
01	CU_CMD_SYN	同期コード
02	CU_CMD_POL	ポーリングコマンド
03	CU_CMD_EOT	EOTコード
04	（未使用）	（未使用）
:	:	:
21	CU_CMD_FS_REQ	ファイル送信要求コマンド
22	CU_CMD_FR_REQ	ファイル受信要求コマンド
23	CU_CMD_FS_START_REQ	ファイル送信開始要求コマンド
24	CU_CMD_FI_NOTICE	ファイル転送情報通知コマンド
25	CU_CMD_FC_TRANS	ファイル内容転送コマンド
26	（未使用）	（未使用）
:	:	:
F1	CU_CMD_MT	画面表示メッセージ送信コマンド
F2	CU_CMD_DL_OFF	データリンク切断コマンド
F3	（未使用）	（未使用）
:	:	:
FF	（未使用）	（未使用）

エラー状態一覧

値	シンボル	意味
00	CU_ERR_NON	エラー発生なし
01	CU_ERR_PHASE	該当関数の使用フェーズ誤り（通信は継続）
02	CU_ERR_FILE	ファイルI／Oエラー
03	CU_ERR_CLOCK	CPU のクロック切り替えエラー
04	CU_ERR_DL	データリンクエラー
05	CU_ERR_DL_RJ	データリンクエラー（主局拒否）
06	CU_ERR_TIMER	タイマー使用エラー
07	CU_ERR_NAK	NAKコード受信でのリトライオーバー
08	CU_ERR_BCC	BCCエラーでのリトライオーバー
09	CU_ERR_SEQ	コマンドシーケンス番号誤り
0A	CU_ERR_CONT	コマンド内容異常
0B	CU_ERR_EOT	EOTコード受信による中断
0C	CU_ERR_RCVTO	受信タイムアウト
0D	CU_ERR_CRP	コマンド／レスポンスのフェーズエラー
0E	CU_ERR_CMD	期待されないコマンドの受信
0F	CU_ERR_FILE_NO	受信ファイル数エラー
10	CU_ERR_SEND	送信エラー
11	CU_ERR_STOP	中断キー押下
12	CU_ERR_NODATA	受信データなし（相手局不在）
13	CU_ERR_DL_OVER	マルチドロップ再データリンクオーバー
14	CU_ERR_UNKNOWN_CMD	未知のコマンドの受信

15	CU_ERR_PRM	通信UT関数パラメータエラー
16	CU_ERR_RECV	受信エラー
17	CU_ERR_AP	受信APのエラー（メモリアドレス等）
18	（未使用）	（未使用）
:	:	:
40	CU_ERR_NOT_SUPPORT	未サポート
41	（未使用）	（未使用）
:	:	:
FE	CU_ERR_ABNORMAL	通信UTロジックエラー
FF	CU_ERR_OTHER	上記以外のエラー 通信関数部システムエラーです

注意： ファイル IO エラーと通信関連のエラーが発生した場合には通信関連のエラーを優先して返却します。

エラー詳細情報（3）

2. 通信関数部システムエラー
   通信プロトコルエラーが発生した時点での、通信関数部システムエラー詳細情報を返します。

3. システムメニューエラー
   システムメニューでのみ発生し得るエラー状態を以下に示します。

値	シンボル	意味
80	－	指定ドライブなし
81	－	（未使用）
82	－	（未使用）
83	－	システム情報取得NG
84	－	システム環境ファイル異常
85	－	送信ファイルなし
89	－	システムメニュー内部エラー

17.8.10 cu_readDIRjInfo

【通信ユーティリティ ： マルチドロッププロトコル】
エラー詳細情報のエラー状態で CU_ERR_DL_RJ が返却された場合の拒否理由値を取得します。

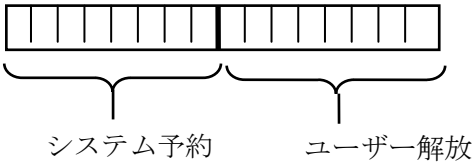
```
ER cu_readDIRjInfo(
  H comNo,
  UH *rjInfo
)
```

パラメータ

comNo
通信ポート

COM0	カシオ IR インタフェース

rjInfo
切断理由値(ビット対応)を設定するアドレス
＜ビット割り付け(該当するものは1を設定)＞



戻り値

関数結果

E_OK	正常終了
E_PRM	パラメータエラー

補足

エラー詳細情報のエラー状態で CU_ERR_DL_RJ が返却された直後に取得する必要があります。それ以外の場合、戻り値は不定となります。

17.9 FLINK プロトコル

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。

17.9.1 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　　カシオ IR インタフェース

irSpeed
赤外通信最高速度　CU_B2400～CU_B115K。但し、Ir_SetWinMode 関数で FIR 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。

rsPrm
予約

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)

【ストラクト構造】

```
typedef struct    {
   H   speed:      : CU_B1200～CU_B115K    /* 転送速度          */
   H   length;         : CU_CHAR8              /* データ長          */
   H   parity;         : CU_PARI_NON           /* パリティﾋﾞｯﾄ なし     */
                   : CU_PARI_ODD           /*       奇数        */
                   : CU_PARI_EVN           /*       偶数        */
   H   stop_bit;   : CU_STOP1              /* ｽﾄｯﾌﾟﾋﾞｯﾄ    1       */
                   : CU_STOP2              /*           2       */
 } CU_RSPRM;
```

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

解説

HT 対 HT で通信を行う場合は、一方の HT が HT モード、もう一方の HT が PC モードでオープンする必要があります。

17.9.2 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　　カシオ IR インタフェース

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、
属性変更して書込を行うかを指定します)
CU_PROTECT_VALID　　強制書込しない
CU_PROTECT_INVALID　する

graphSet
グラフ表示情報

【ストラクト構造】

```
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

17.9.3 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　　カシオ IR インタフェース

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

17.9.4 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　　カシオ IR インタフェース

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、
属性変更して書込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

17.9.5 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	カシオ IR インタフェース

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

17.9.6 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

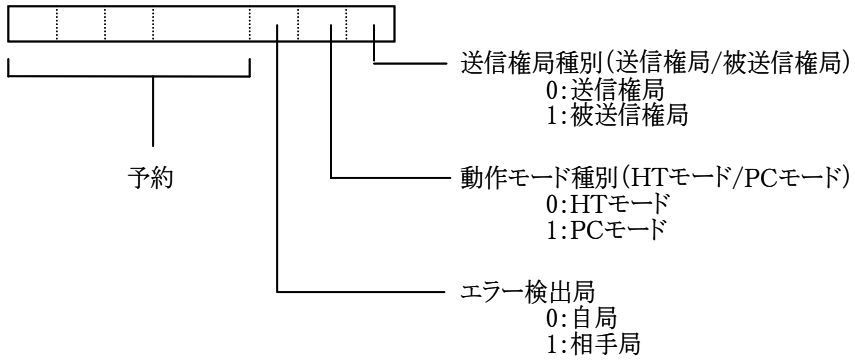
comNo
通信ポート
COM0　　カシオ IR インタフェース

errInfo
エラー情報設定エリア

戻り値

【ストラクト構造】

```
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

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	・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	・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	・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	・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	・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	・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	・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	・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	・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	・CONFIG.HTS 異常 ・ファイルレイアウトを確認
	03	相手局不正	・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

17.9.7 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　　カシオ IR インタフェース

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

17.9.8 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　　カシオ IR インタフェース

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

17.9.9 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　　カシオ IR インタフェース

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

17.9.10 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　　カシオ IR インタフェース

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

17.9.11 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　　カシオ IR インタフェース

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(下記参照)

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
(_A_SUBDIR は自動的に OR されます)

戻り値

【ストラクト構造】

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

17.9.12 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　　カシオ IR インタフェース

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。

戻り値

【ストラクト構造】

```
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;
```

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

17.9.13 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　　カシオ IR インタフェース

fInfo
ファイル情報設定エリア

戻り値

【ストラクト構造】

```
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;
```

E_OK　　正常終了
E_NG　　異常終了
E_PRM　　パラメータエラー

17.9.14 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　　カシオ IR インタフェース

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)

戻り値

【ストラクト構造】

```
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  :ﾃﾞｨｽｸあり(ﾌｫｰﾏｯﾄ済)
                          CU_DINFO_NOFMT       :ﾃﾞｨｽｸあり(未ﾌｫｰﾏｯﾄ)
                          CU_DINFO_NODISK  :ﾃﾞｨｽｸなし
} CU_DINFO;
```

E_OK　　正常終了
E_NG　　異常終了
E_PRM　　パラメータエラー

17.9.15 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　　カシオ IR インタフェース

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)

戻り値

【ストラクト構造】

```
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。

E_OK　　正常終了
E_NG　　異常終了
E_PRM　　パラメータエラー

17.9.16 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　　カシオ IR インタフェース

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)

戻り値

【ストラクト構造】

```
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;       ：ﾓﾃﾞﾙ情報(00h 固定)
} CU_SYSINFO;
```

E_OK　　正常終了
E_NG　　異常終了
E_PRM　　パラメータエラー

17.9.17 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　　カシオ IR インタフェース

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

17.9.18 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	カシオ IR 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

17.9.19 cu_setIoboxInfo

【通信ユーティリティ ： FLINK プロトコル】
マスターIO ボックス(TCP/IP)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 IO ボックスに送信され、IO ボックス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 IO ボックス通信を行いません)

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 実行時)取得情報レングス

getInfo
取得情報アドレス(cu_open 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

IO ボックスが TCP/IP 未対応の場合、設定情報は送信されません。
cu_open では、IO ボックス通信後、ホスト(PC)とのコネクションを実行します。

17.9.20 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

戻り値

【ストラクト構造】

```
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;
```

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

17.9.21 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

戻り値

【ストラクト構造】

```
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;
```

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

17.10 DT500 プロトコル

17.10.1 cu_open

【通信ユーティリティ ： DT500 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います。

```
ER cu_open(
  H comNo,
  struct sys_tty *param
)
```

パラメータ

comNo
通信ポート

COM0	カシオIRインタフェース

param
通信パラメータエリアアドレス

戻り値

【ストラクト構造】

```
struct sys_tty {
      W speed;       : B_1200～B_115200 /* 転送速度            */
      W length;      : CHAR_8, CHAR_7   /* ﾃﾞｰﾀ長固定          */
      W parity;      : PARI_NON         /* ﾊﾟﾘﾃｨﾋﾞｯﾄ  なし    */
                     : PARI_ODD         /*             奇数    */
                     : PARI_EVN         /*             偶数    */
      W stop_bit;    : STOP_1           /* ｽﾄｯﾌﾟﾋﾞｯﾄ  1       */
                     : STOP_2           /*             2       */
      };
```

関数結果

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

補足

17.10.2 cu_fileSend

【通信ユーティリティ ： DT500 プロトコル】
指定された1ファイルを送信します。
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。
転送ファイルは、転送ドライブ指定（cu_setDrive）で指定されたドライブ上に存在する必要があります。

```
ER cu_fileSend(
  H comNo,
  B *fName,
  B fieldCount,
  UB *fieldCol,
  CU_DT_OPT *option
)
```

パラメータ

comNo
通信ポート
COM0　　カシオ IR インタフェース

fName
送信ファイル名エリアアドレス　（ワイルドカード不可）
ファイル名のみを指定します（例：“DTFILE01.DAT”）

fieldCount
送信ファイルのフィールド数 （1～16）

fieldCol
送信するファイルの各フィールド桁数
（1バイト, 1～254）の配列　アドレス（fieldCount が配列の長さ）
例）フィールド数5, 各フィールド桁数が, 10, 10, 8, 4, 20 の場合

fieldCount = 5 fieldCo

10	10	8	4	20

option
プロトコルオプションエリアアドレス

graphSet
グラフ表示情報エリアアドレス

戻り値

【ストラクト構造】

```
typedef struct {
      B  serial      : シリアル番号
                           CU_ON          : 付加する
                           CU_OFF         : 付加しない
      B  bcc         : 水平パリティチェック
                           CU_ON          : 付加する
                           CU_OFF         : 付加しない
      B  timeOut     : タイムアウト時間設定  1～9
      UB graphMode   : グラフ表示モード
                           CU_GRAPH_ON    : １ファイルを 100%として表示
                           CU_GRAPH_OFF   : 表示しない
    (CU_GRAPH_OFF 設定時は以下のパラメータは参照しない)
      UB graphPos    : ファイル名表示先頭行(0～7)
      }CU_DT_OPT
```

関数結果

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

補足

フィールド数および各フィールド桁数は(各フィールドの総バイト数＋フィールド数)＜255の条件を満たす必要があります。

17.10.3 cu_fileRecv

【通信ユーティリティ ： DT500 プロトコル】
ファイルを受信します。
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。
転送ファイルは、転送ドライブ指定（cu_setDrive）で設定されたドライブ上に受信されます。

```
ER cu_fileRecv(
  H comNo,
  B *fName,
  B *fieldCount,
  UB *fieldCol,
  CU_DT_OPT *option
)
```

パラメータ

comNo
通信ポート
COM0　　カシオ IR インタフェース

fName
受信するファイル名エリアアドレス　　（ファイル名省略は不可）

fieldCount
（呼出時）fieldColから始まる配列の長さ
（戻り時）受信したファイルのフィールド数

fieldCom
受信ファイルの各フィールド桁数（1バイト, 1～254）の配列アドレス
（fieldCount が配列の長さ）

option
プロトコルオプションエリアアドレス

戻り値

【ストラクト構造】

```
typedef struct {
      B  serial      : シリアル番号
                          CU_ON          : 付加する
                          CU_OFF         : 付加しない
      B  bcc         : 水平パリティチェック
                          CU_ON          : 付加する
                          CU_OFF         : 付加しない
      B  timeOut     : タイムアウト時間設定  1～9
      UB graphMode   : グラフ表示モード
                          CU_GRAPH_ON    : １ファイルを 100%として表示
                          CU_GRAPH_OFF   : 表示しない
    (CU_GRAPH_OFF 設定時は以下のパラメータは参照しません)
      UB graphPos    : ファイル名表示先頭行(0～7)
      }CU_DT_OPT
```

関数結果

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

補足

17.10.4 cu_close

【通信ユーティリティ ： DT500 プロトコル】
通信ポートをクローズします。

```
ER cu_close(
  H comNo
)
```

パラメータ

comNo
通信ポート

COM0	カシオ IR インタフェース

戻り値

関数結果

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

補足

17.10.5 cu_readErrStat

【通信ユーティリティ ： DT500 プロトコル】
当ファイル送受信関数でのエラー詳細情報を取得します。取得後、エラー詳細情報はクリアされます。

```
ER cu_readErrStat(
  H comNo,
  UW *cuStat,
  UW *biosStat
)
```

パラメータ

comNo
通信ポート

COM0	カシオ IR インタフェース

cuStat
通信ユーティリティエラーの情報設定エリアアドレス
（次ページの「エラー詳細情報」を参照して下さい）

biosStat
エラー発生時の通信関数部システムエラー詳細情報
（次ページの「エラー詳細情報」を参照して下さい）

戻り値

関数結果

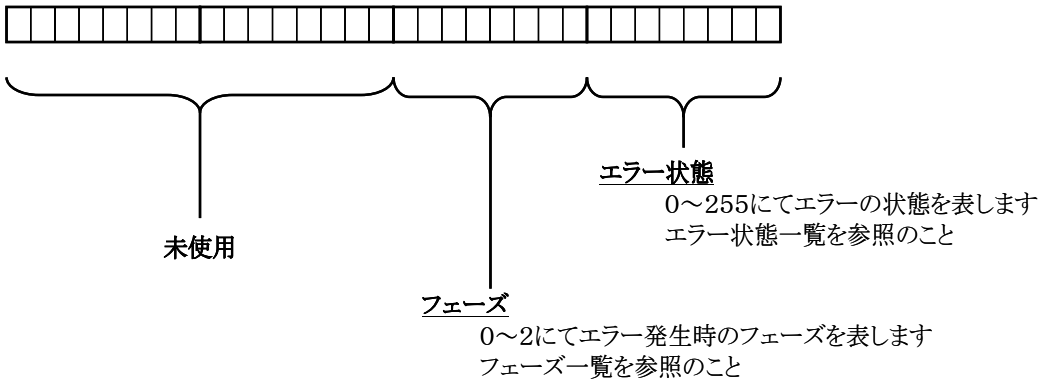
E_OK	正常終了
E_PRM	パラメータエラー

補足

エラー詳細情報

1．通信ユーティリティエラー

通信ユーティリティとしてのエラーを返す。以下のビット構成で通知されます。



フェーズ一覧

値	シンボル	意味
0	CU_DT_PHASE_DATALINK	データリンク確立中
1	CU_DT_PHASE_TRANS	データ伝送中
2	CU_DT_PHASE__END	データリンク終結中

エラー状態一覧

値	意味
00	エラー発生なし。
02	設定ファイル不正
05	パラメータエラー
07	ファイルライトエラー（ディスクフルエラー）
32	ファイルタイプ不正
33	受信テキストフォーマット不正
34	回線オープンエラー
35	ファイルが見つからない
37	ファイルオープンエラー
38	ヘディングテキストファイル名異常
3B	レコード数オーバー
46	通信エラー
47	中断キー
50	通信タイムアウトエラー
51	EOT受信エラー
52	NAK受信リトライオーバー
53	ENQ受信リトライオーバー
54	受信BCCエラー
55	受信シリアル番号エラー
56	受信ヘッダー情報エラー
60	ファイルIOエラー
70～73	内部エラー

2. 通信関数部システムエラー
通信エラーが発生した時点での、通信関数部システムエラー詳細情報を返します。

3. システムメニューエラー
システムメニューでのみ発生し得るエラー状態を以下に示します。

値	意味
80	システム環境ファイル異常
81	指定ドライブなし
82	フォーマットエラー
83	送信ファイル未検出
84	システム情報取得NG

17.10.6 cu_SetCode

【通信ユーティリティ ： DT500 プロトコル】
DT500プロトコルの制御ヘッダコード・ターミネータコード（SOH・STX・ETX）の値を変更または読出します。

```
ER cu_setSOH(
  UB kind ,
  UB mode,
  UB *code
)
```

パラメータ

kind
参照／変更する制御コードを指定する。

CU_DT_SOH	SOHコード（デフォルト 01H）
CU_DT_STX	STXコード（デフォルト 02H）
CU_DT_ETX	ETXコード（デフォルト 03H）

mode
設定／読出しフラグ

CU_DT_GET	コード読出し
CU_DT_SET	コード設定

code
設定／読出しコード格納アドレス

コード読出し時	現在のコードを返します。
コード設定時	入力コードを設定します。最後にヌルを設定してください。

・コード値の有効バイト数は1バイトです。
00H は出力しないことを表します。ただし、ETX は 00H 設定はできません。

戻り値

関数結果

E_OK	正常終了
E_PRM	パラメータエラー

補足

18.共通関数

18.1 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。

18.1.1 アプリケーションのロードと実行

DT-930 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。

18.1.2 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。

18.1.3 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。

18.1.4 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。

18.1.5 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。

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処理の起動

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。

18.2.1 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_NG	異常終了

18.2.2 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)
①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。

18.2.3 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。
①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

18.2.4 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

18.2.5 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

19. 参考資料

19.1 機能比較

(1)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
lcd_csr_set	ｶｰｿﾙﾀｲﾌﾟ設定	○	○	○	○	○
lcd_csr_put	ｶｰｿﾙ位置設定	○	○	○	○	○
lcd_csr_get	ｶｰｿﾙ位置読出し	○	○	○	○	○
lcd_char	1 文字表示	○	○	○	○	○
lcd_string	文字列表示	○	○	○	○	○
lcd_line	直線描画	○	○	○	○	○
lcd_cls	画面クリア	○	○	○	○	○
lcd_led	LED の制御	○	○	○	○	○
lcd_gaiji	外字フォント登録	○	○	○	○	○
lcd_string2	文字列表示 2(ｽｸﾛｰﾙ抑制)	○	○	○	○	○
lcd_usrfont	ﾕｰｻﾞﾌｫﾝﾄﾌｧｲﾙ登録	○	○	○	○	○
lcd_romfont	ROM フォント設定	○	○	○	○	○
lcd_userstr	ﾕｰｻﾞ文字列表示	○	○	○	○	○
lcd_active_set	ﾕｰｻﾞ描画ページ設定			○		
lcd_visual_set	ﾕｰｻﾞ表示ページ設定			○		
lcd_active_get	ﾕｰｻﾞ描画ページ読出し			○		
lcd_visual_get	ﾕｰｻﾞ表示ページ読出し			○		
lcd_cls_page	指定ﾍﾟｰｼﾞｸﾘｱ			○		
lcd_grchar	ｸﾞﾗﾌｨｯｸ文字表示			○		
lcd_box	BOX 描画/削除			○		
lcd_el	EL ﾊﾞｯｸﾗｲﾄの制御			○	○	○

(3)キー部

関数名	機能	DT700	DT750	DT800	DT900	DT930
key_read	1 文字入力	○	○	○	○	○
key_string	文字列入力	○	○	○	○	○
key_num	数値入力	○	○	○	○	○
key_check	ｷｰﾊﾞｯﾌｧのｽﾃｰﾀｽﾁｪｯｸ	○	○	○	○	○
key_clear	ｷｰﾊﾞｯﾌｧのｸﾘｱ	○	○	○	○	○
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_select	キー入力モード設定		○	○	○	○
key_patchg	キー入力有効無効設定/解除		○			
key_maketime	切替えキー確定時間設定		○			
key_pad_set	ｷｰﾊﾟｯﾄﾞﾌｧｲﾙ登録			○		
key_touch	ﾕｰｻﾞﾀｯﾁｷｰ設定削除			○		
key_point	ﾀｯﾁ座標取得			○		
key_pad	ｷｰﾊﾟｯﾄﾞ切替/状態取得			○		
key_pad_entry	ｷｰﾊﾟｯﾄﾞ遷移設定/状態取得			○		

(4)通信部

関数名	機能	DT700	DT750	DT800	DT900	DT930
c_open	COM のオープン	○	○	○	○	○
c_close	COM のクローズ	○	○	○	○	○
c_status	COM ｽﾃｰﾀのｽﾘｰﾄﾞ	○	○	○	○	○
c_hold	COM の占有	○	○	○	○	○
c_chkopen	COM のｵｰﾌﾟﾝﾁｪｯｸ	○	○	○	○	○
c_wp	WakeUp 信号の設定				○	
c_dout	n 文字送信	○	○	○	○	○
c_din	1 文字受信	○	○	○	○	○
c_tmdin	タイムアウト監視受信	○	○	○	○	○
c_mdout	ﾒﾓﾘﾌﾞﾛｯｸ送信	○	○			
c_mdin	ﾒﾓﾘﾌﾞﾛｯｸ受信	○	○			
c_out	1 文字送信	○	○	○	○	○
c_break	ブレーク信号の制御	○	○	○	○	○
c_txrx	送受信の有効/無効	○	○	○	○	○
c_iobox	IO ボックス送信設定	○	○		○	○
c_irout	IO ボックス送信	○	○		○	○
c_timer	DR/CS/CD タイムアウト監視値設定	○		○	○	○
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_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ﾄﾞ	○	○	○	○	○
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ﾄﾞ	○	○	○	○	○
c_chghdr	受信ﾊﾝﾄﾞﾗ切替え	○	○	○	○	○
c_cimode	CI 信号立ち上げモード設定				○	
c_brkevent	ブレーク要因の設定				○	○

(5)電源部

関数名	機能	DT700	DT750	DT800	DT900	DT930
pwr_inhabit	通知モード設定	○	○	○	○	○
pwr_inhabit_clr	電源通知ｲﾍﾞﾝﾄのクリア	○	○	○	○	○
pwr_hold_apo	APO 禁止設定	○	○	○	○	○
pwr_off	電源ｵﾌ	○	○	○	○	○
pwr_IoboxBootMode	IO ボックス起動設定					○
pwr_vibrator	バイブレータ動作開始					○

(6)通知部

関数名	機能	DT700	DT750	DT800	DT900	DT930
flg_sts	通知ﾌﾗｸﾞ状態取得	○	○	○	○	○
clr_flg	通知ﾌﾗｸﾞ状態クリア	○	○	○	○	○
wai_flg	ﾌﾗｸﾞｾｯﾄ待ち	○	○	○	○	○

(7)バーコード部

関数名	機能	DT700	DT750	DT800	DT900	DT930
OBR_open	OBR オープン	○	○	○	○	○
OBR_close	OBR クローズ	○	○	○	○	○
OBR_getc	OBR データ 1 文字ﾘｰﾄﾞ	○	○	○	○	○
OBR_gets	OBR データ文字列ﾘｰﾄﾞ	○	○	○	○	○
OBR_stat	OBR バッファステータスチェック	○	○	○	○	○
OBR_flush	OBR バッファのクリア	○	○	○	○	○
OBR_moderd	OBR 動作モードの取得	○	○	○	○	○
OBR_modewt	OBR 動作モード設定	○	○	○	○	○
OBR_chgbuf	OBR バッファの切替え	○	○	○	○	○
OBR_gain	発光ｹﾞｲﾝ切替え		○	○		
OBR_trigmode	ﾄﾘｶﾞｰキーによる電源ｵﾝ設定		○	○	○	○
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	バー幅補正モード取得					○
OBR_setadjust	バー幅補正モード設定					○
OBR_getmargincheck	ﾏｰｼﾞﾝﾁｪｯｸ倍率モード取得					○
OBR_setmargincheck	ﾏｰｼﾞﾝﾁｪｯｸ倍率モード設定					○

(8)バーコード動作モード

機能	DT700	DT750	DT800	DT900	DT930
読取り可能コード	○	○	○	○	○
読取り桁数	○	○	○	○	○
出力ﾌｫｰﾏｯﾄ	○	○	○	○	○
ﾁｪｯｸﾃﾞｼﾞｯﾄの実行	○	○	○	○	○
ﾁｪｯｸｷｬﾗｸﾀの出力	○	○	○	○	○
読取り方式(単発)	○	○	○	○	○
読取り方式(連続:TRG 有)	○	○	○	○	○
読取り方式(連続:切替)	○	○	○	○	○
読取り方式(連続:TRG 無)	○	○	○	○	○
ｹﾞｲﾝｺﾝﾄﾛｰﾙ	○	○	×	×	×
ﾌﾞｻﾞｰ制御	○	○	○	○	○
LED 制御	○	○	○	○	○
出力バッファ	○	○	○	○	○
終了コード	○	○	○	○	○
読取り動作	○	○	○	○	○
2 つコード認識		○	○		
レーザー発光幅制御				○	○

(9)タイマ部

関数名	機能	DT700	DT750	DT800	DT900	DT930
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	ｴﾗｰﾋﾞｰﾌﾟ音	○	○	○	○	○
s_beep2	ｴﾗｰﾋﾞｰﾌﾟ音 2(赤 LED 点灯)		○	○		
s_sound	ｻｳﾝﾄﾞ音 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	○	○	○	○	○

(10)データ管理部

関数名	機能	DT700	DT750	DT800	DT900	DT930
dat_mem_size	ﾒﾓﾘ領域の空きサイズの取得	○	○	○	○	○
dat_system	システムデータの設定	○	○	○	○	○
dat_OSVer_Read	OS バージョン読出し		○	○	○	○
dat_dealer_chk	代理店 ID のﾁｪｯｸ			○	○	○
dat_Apload(※)	AP ロード&実行			RAM○	○	○

※ A または B ドライブに存在する AP から他の AP を実行します。
複数の AP 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。

(11)システムデータ

項目	管理データ	DT700	DT750	DT800	DT900	DT930
電源関連	APO 時間	○	○	○	○	○
	ABO 時間	○	○	○	○	○
	レジューム ON/OFF	○	○	○	○	○
	自動コントラスト調整 ON/OFF			○		
KEY 関連	クリック音 ON/OFF	○	○	○	○	○
表示関連	フォント MODE	○	○	○	○	○
	フォント種別(通常/強調)			○	○	○
	日本語/英語	○	○	○	○	○
	コントラスト値	○	○	○	○	○
	コントラスト差分			○	○	○
	LB 表示 MODE			○		
通信関連	速度(IR)	○	○	○	○	○
	データ(IR)		○	予約	○	○
	パリティ(IR)	○	○	予約	○	○
	STOP(IR)	○	○	予約	○	○
	速度(RF/シリアル)	○	○	予約	○	
	データ(RF/シリアル)		○	予約	○	
	パリティ(RF/シリアル)	○	○	予約	○	
	STOP(RF/シリアル)	○	○	予約	○	
	速度(10P)	○	○	○		
	データ(10P)		○	予約		
	パリティ(10P)	○	○	予約		
	STOP(10P)	○	○	予約		
	デフォルト通信 PORT			○		
	速度(PHS)				○	
	データ(PHS)					
	パリティ(PHS)				○	
	STOP(PHS)				○	
OBR 関連	読取り回数	○	○	○	○	○
	照合回数		○	○	○	○
	スキャン時間	○	○	○	○	○
	読取り禁止時間	○	○	予約		
タイマ関連	音量	○	○	○	○	○
システム関連	機器 ID	○	○	○	○	○
	代理店 NO			○	○	○
	BIOS バージョン	○	○	○	○	○
	PATCH バージョン			○	○	○
	機器種別	○	○	○	○	○
ネットワーク関連	IP アドレス(SS 無線)			○		
	マスク値(SS 無線)			○		

(前頁つづき)

項目	管理データ	DT700	DT750	DT800	DT900	DT930
プロトコル関連	通常受信タイムアウト	○	○		○	○
	通常リトライ回数	○	○		○	○
	マルチデータリンク受信タイムアウト	○			○	○
	対向送信データリンク受信タイムアウト	○				
	対向受信データリンク受信タイムアウト	○				
	対向受信データリンクリトライ回数	○				
	データリンク受信タイムアウト		○			
	受信データなしタイムアウト		○			
	再データリンク可能回数		○			
	セッション確立タイムアウト			○	○	○
	受信タイムアウト			○	○	○
	DR タイムアウト(10PIN)			○		
	CS タイムアウト(10PIN)			○		
	CD タイムアウト(10PIN)			○		
	シリアル NO				○	○
	水平パリティ				○	○
	リンクタイムアウト				○	○
メモリ関連	アプリケーション SIZE			○	○	○
ファイルモード	FORMAT				○	○

(12)ファイル部

関数名	機能	DT700	DT750	DT800	DT900	DT930
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_F_Search	ファイルデータの検索	※	○		○	○
dat_fsize_chg	ファイルサイズ変更		○			
open	ファイルｵｰﾌﾟﾝ	○	○	○	○	○
close	ファイルｸﾛｰｽﾞ	○	○	○	○	○
read	ファイルのﾘｰﾄﾞ	○	○	○	○	○
write	ファイルのﾗｲﾄ	○	○	○	○	○
lseek	ファイルﾘｰﾄﾞ/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の取得				○	○
fil_filefind	ファイル全ﾊﾟｽ名の取得				○	○

※ dat_sub.obj とﾘﾝｸすることで対応

(13)共通関数

関数名	機能	DT700	DT750	DT800	DT900	DT930
abort	ABORT 処理	○	○	○	○	○
exit	EXIT 処理	○	○	○	○	○
wkup_cost	動作環境メニューの起動	○	○	○	○	○
wkup_calib	キャリブレーション起動			○	○	○
wkup_ss	SS 無線ユーティリティ起動			○		

(14)通信ユーティリティ

関数名	機能	DT700	DT750	DT800	DT900	DT930
cu_open	通信ポート初期化	○	○	○	○	○
cu_stopKeySet	中断キーの登録/削除	○	○	○	○	○
cu_fileSend	ファイル送信	○	○	○	○	○
cu_fileSendSet	ファイル送信情報設定	○	○		○	○
cu_fileSend1	1 ファイル送信	○	○		○	○
cu_fileRecv	ファイル受信	○	○	○	○	○
cu_msgSend	画面表示ﾒｯｾｰｼﾞ送信	○	○	○	○	○
cu_end	通信中断	○	○		○	○
cu_close	回線クローズ	○	○	○	○	○
cu_readErrStat	エラー詳細情報取得	○	○	○	○	○
cu_readDIRjInfo	データリンク拒否情報取得	○	○		○	○
cu_downloadSet	ﾀﾞｳﾝﾛｰﾄﾞ動作設定		○			
cu_fileAdd	ファイルの追加			○	○	○
cu_idle	IDLE 遷移			○	○	○
cu_cmdRecv	ｺﾏﾝﾄﾞ受信待ち			○	○	○
cu_fileDelete	ファイル削除			○	○	○
cu_fileMove	ファイル移動			○	○	○
cu_makeDir	ﾃﾞｨﾚｸﾄﾘ作成			○	○	○
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	ﾃﾞｨｽｸ情報の取得			○	○	○
cu_beep	ﾌﾞｻﾞｰ鳴動			○	○	○
cu_getSysInfo	システム情報の取得			○	○	○
cu_apRecvSet	AP ｲﾝｽﾄｰﾙ設定			○		
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_setDrive	転送ドライブ指定				○	○
cu_msgSend	画面表示ﾒｯｾｰｼﾞ送信				○	○
cu_SetCode	DT500 プロトコル制御コード拡張設定				○	○

(15)IrDA 部

関数名	機能	DT700	DT750	DT800	DT900	DT930
Ir_Open	IrCOMM オープン			○	○	○
Ir_Close	IrCOMM クローズ			○	○	○
Ir_Read	データ読込			○	○	○
Ir_Write	データ書込			○	○	○
Ir_QueryTx	送信データ数問合せ			○	○	○
Ir_QueryRx	受信データ数問合せ			○	○	○
Ir_EROn	ER ON			○	○	○
Ir_EROff	ER OFF			○	○	○
Ir_RSOn	RS ON			○	○	○
Ir_RSOff	RS OFF			○	○	○
Ir_BreakOn	BREAK ON			○	○	○
Ir_BreakOff	BREAK OFF			○	○	○
Ir_CheckCD	CD 検査			○	○	○
Ir_CheckDR	DR 検査			○	○	○
Ir_CheckCS	CS 検査			○	○	○
Ir_CheckCI	CI 検査			○	○	○
Ir_CheckBreak	BREAK 検査			○	○	○
Ir_Err_Get	エラー値取得			○	○	○
Ir_State_Set	通信状態設定			○	○	○
Ir_SetPortConfig	自局能力設定			○	○	○
Ir_Init	IrCOMM 強制終了			○	○	○
Ir_SetParame	パラメータ設定			○		
Ir_SetWinMode	Ir モード切替設定					○

(16)Bluetooth 部

関数名	機能	DT700	DT750	DT800	DT900	DT930
BT_Start	Bluetooth 通信開始					○
BT_Stop	Bluetooth 通信終了					○
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 デバイス情報取得					○
BT_GetDevName	Bluetooth デバイス名取得					○
BT_SetPassKey	Bluetooth パスキーの設定					○
BT_SelectDev	接続する Bluetooth 機器の指定					○
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 デバイス情報保存					○
BT_LoadDevInfo	Bluetooth デバイス情報読出し					○
BT_Err_Get	エラー値の取得					○

(17)メモリーバックアップ

関数名	機能	DT700	DT750	DT800	DT900	DT930
Mbu_format	FROM 領域の確保		○			
Mbu_open	ファイルｵｰﾌﾟﾝ		○			
Mbu_close	ファイルｸﾛｰｽﾞ		○			
Mbu_write	ファイル書込		○			
Mbu_read	ファイル読出し		○			
Mbu_clear	FROM クリア		○			
Mbu_sts	メモリーバックアップ状態の取得		○			

(18)プリンタ

関数名	機能	DT700	DT750	DT800	DT900	DT930
Prn_inf_open	赤外オープン		○			
Prn_inf_close	赤外クローズ		○			
Prn_inf_send	赤外のｺﾏﾝﾄﾞ送信		○			
Prn_inf_status	赤外のｽﾃｰﾀｽﾘｰﾄﾞ		○			
Prn_wir_open	無線オープン		○			
Prn_wir_close	無線クローズ		○			
Prn_wir_send	無線のｺﾏﾝﾄﾞ送信		○			
Prn_pktlen	無線のﾊﾟｹｯﾄ長登録		○			
Prn_crc_calc	CRC 算出		○			

(19)システムメニュー

項目	設定内容	DT700	DT750	DT800	DT900	DT930
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 ﾊﾞｯｸｱｯﾌﾟ		○			
	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	○	○	○	○	○
	同期ｲﾝｽﾄｰﾙ		○	○	○	○
	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			○	○	○
	ﾒﾓﾘｻｲｽﾞ変更			○	○	○
	ファイルモード				○	○
ファイル転送	ﾎｽﾄ受信	○	○			
	ﾎｽﾄ送信	○	○			

(20)動作環境メニュー

項目	設定内容	DT700	DT750	DT800	DT900	DT930
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	○	○	○	○	○
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	○	○			
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				○	○
ID セット	機器 ID	○	○	○	○	○
	代理店 ID			○	○	○

カシオ計算機お問い合わせ窓口

製品に関する最新情報

●製品サポートサイト（カシオペア・ハンディターミナル）

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

製品の取扱い方法のお問い合わせ

●情報機器コールセンター

0570-022066
市内通話料金でご利用いただけます。
携帯電話・PHS 等をご利用の場合、**048-233-7241**

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）