

HANDY TERMINAL DT-930

C ライブラリ解説書

このマニュアルは、C ライブラリの
仕様について記載します。



ご注意

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

© 2006 カシオ計算機株式会社

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

目次

1.	概要	1
1.1.	提供ファイルについて	1
1.2.	標準ライブラリ関数	3
1.3.	専用ライブラリ関数	5
1.3.1.	データ管理部関数	5
1.3.2.	表示部関数	5
1.3.3.	キー部関数	6
1.3.4.	O B R 部関数	6
1.3.5.	通信部関数	7
1.3.6.	IrDA 部関数	8
1.3.7.	Bluetooth 部関数	8
1.3.8.	通信ユーティリティ部関数	9
1.3.9.	タイマー部関数	10
1.3.10.	電源部関数	10
1.3.11.	通知モード部関数	10
1.3.12.	共通関数	10
1.3.13.	プリンタ制御関数	10
1.4.	データタイプ	11
2.	データ管理部	12
2.1.	機能	12
2.1.1.	メモリー管理機能	12
2.1.2.	ファイル管理機能	12
2.1.3.	データ管理機能（システムデータ管理）	12
2.1.4.	低水準インタフェース	16
2.1.5.	ファイル情報サーチインタフェース	18
2.1.6.	システムデータファイル	22
2.2.	ファンクション詳細	25
2.2.1.	dat_system	26
2.2.2.	dat_OSVer_Read	30
2.2.3.	dat_dealer_chk	31
2.2.4.	dat_Apload	32
2.2.5.	open	33
2.2.6.	close	35
2.2.7.	read	36
2.2.8.	write	37
2.2.9.	lseek	38
2.2.10.	sbrk	39
2.2.11.	dat_fdir (DT700 互換モード専用)	40
2.2.12.	dat_fsize (DT700 互換モード専用)	42
2.2.13.	dat_fdel (DT700 互換モード専用)	43
2.2.14.	dat_mem_size	44
2.2.15.	fil_mkdir (F A T ファイルモード専用)	45
2.2.16.	fil_rmdir (F A T ファイルモード専用)	46
2.2.17.	fil_remove	47
2.2.18.	fil_rename	48
2.2.19.	fil_fstat	49
2.2.20.	fil_chsize	50
2.2.21.	fil_getsize	51

2.2.22.	fil_findfirst	52
2.2.23.	fil_findnext	54
2.2.24.	fil_filesize	55
2.2.25.	fil_filefind	56
2.2.26.	dat_fname (DT700 互換モード専用)	57
2.2.27.	dat_F_Search (DT700 互換モード専用)	58
3.	表示部	60
3.1.	表示制御	60
3.1.1.	表示画面	60
3.1.2.	表示コード	62
3.1.3.	制御コード表示	65
3.1.4.	E S C 文字列の表示	66
3.1.5.	例外表示制御	67
3.1.6.	スクロール制御	69
3.2.	フォント制御	70
3.2.1.	フォントの種類	70
3.2.2.	フォントデータ構成	71
3.2.3.	修飾文字のフォントデータ制御	78
3.2.4.	ユーザーフォントファイル	81
3.3.	D T 7 0 0 互換表示モード	83
3.3.1.	互換モード A	83
3.3.2.	互換モード B	83
3.4.	ファンクション詳細	84
3.4.1.	lcd_cls	85
3.4.2.	lcd_csr_set	86
3.4.3.	lcd_csr_put	87
3.4.4.	lcd_csr_get	88
3.4.5.	lcd_char	89
3.4.6.	lcd_string	90
3.4.7.	lcd_string2	92
3.4.8.	lcd_userstr	93
3.4.9.	lcd_line	94
3.4.10.	lcd_gajji	95
3.4.11.	lcd_usrfont	96
3.4.12.	lcd_romfont	97
3.4.13.	lcd_led	98
3.4.14.	lcd_el	99
4.	キー部	100
4.1.	機能	100
4.1.1.	キーモード	100
4.1.2.	1 文字入力	100
4.1.3.	文字列入力	101
4.1.4.	数値入力	102
4.1.5.	キーコードの設定	103
4.1.6.	キー通知設定	105
4.1.7.	キー入力有効/無効設定	106
4.1.8.	キーバッファ	107
4.1.9.	バックライト制御	107
4.1.10.	多点押し処理	108
4.1.11.	キーロールオーバー機能	109
4.2.	キーコード	109

4.2.1.	属性	109
4.2.2.	コード	110
4.3.	ファンクション詳細	111
4.3.1.	key_read	112
4.3.2.	key_string	113
4.3.3.	key_num	115
4.3.4.	key_check	117
4.3.5.	key_clear	118
4.3.6.	key_fnc	119
4.3.7.	key_fnc_mode	120
4.3.8.	key_select	121
5.	OBR 部	123
5.1.	基本仕様	123
5.1.1.	レーザースキャナ部	123
5.1.2.	デコード仕様	123
5.2.	機能	126
5.2.1.	1文字／文字列の読み込み	126
5.2.2.	OBR データバッファの状態チェック	126
5.2.3.	OBR データバッファのクリア	126
5.2.4.	格納先バッファの切り替え	126
5.2.5.	モード設定	126
5.2.6.	レーザ発光幅制御	131
5.2.7.	カスタマイズ機能	133
5.3.	ファンクション詳細	136
5.3.1.	OBR_open	137
5.3.2.	OBR_close	138
5.3.3.	OBR_getc	139
5.3.4.	OBR_gets	140
5.3.5.	OBR_flush	141
5.3.6.	OBR_stat	142
5.3.7.	OBR_moderd	143
5.3.8.	OBR_modewt	144
5.3.9.	OBR_chgbuf	148
5.3.10.	OBR_trigmode	149
5.3.11.	OBR_swing	150
5.3.12.	OBR_widenarrow	151
5.3.13.	OBR_getadjust	152
5.3.14.	OBR_setadjust	153
5.3.15.	OBR_getmargincheck	154
5.3.16.	OBR_setmargincheck	155
6.	通信部	156
6.1.	通信仕様	156
6.1.1.	通信インタフェース	156
6.2.	機能	157
6.2.1.	通信ポートのオープン・クローズ・占有	157
6.2.2.	転送データの送信・受信	158
6.2.3.	S I / S O 制御	164
6.2.4.	フロー制御	164
6.2.5.	デリートコード制御	167
6.2.6.	エラーコードバッファリング制御	167
6.2.7.	信号線制御とタイムアウト監視	168

6.2.8.	L B 検出	171
6.2.9.	ブレイク要因検出	172
6.3.	エラー詳細	173
6.3.1.	ファンクションのエラー検出	173
6.3.2.	エラー詳細	174
6.4.	通信関数 補足	183
6.4.1.	受信データの読み込み	183
6.4.2.	L B エラーチェック	184
6.4.3.	中断キーによる処理	186
6.4.4.	カシオ I R ポートの使用	187
6.4.5.	通信関数部制限	191
6.5.	ファンクション詳細	192
6.5.1.	c_open	193
6.5.2.	c_close	195
6.5.3.	c_status	196
6.5.4.	c_hold	198
6.5.5.	c_chkopen	199
6.5.6.	c_dout	200
6.5.7.	c_din	201
6.5.8.	c_tmdin	202
6.5.9.	c_out	203
6.5.10.	c_break	204
6.5.11.	c_trx	205
6.5.12.	c_jobox	206
6.5.13.	c_irout	207
6.5.14.	c_flush	208
6.5.15.	c_bfsts	209
6.5.16.	c_errbfiring	210
6.5.17.	c_rderrsts	211
6.5.18.	c_chghdr	212
6.5.19.	c_timer	213
6.5.20.	c_er	214
6.5.21.	c_rs	215
6.5.22.	c_errs	216
6.5.23.	c_brkevent	217
7.	IrDA 部関数	218
7.1.	機能	218
7.1.1.	シリアルポートエミュレーション	218
7.1.2.	ファンクションコール	219
7.1.3.	ファンクションの優先順位	224
7.2.	ファンクション詳細	225
7.2.1.	Ir_Open	226
7.2.2.	Ir_Close	227
7.2.3.	Ir_Read	228
7.2.4.	Ir_Write	229
7.2.5.	Ir_QueryTx	230
7.2.6.	Ir_QueryRx	231
7.2.7.	Ir_EROn	232
7.2.8.	Ir_EROff	233
7.2.9.	Ir_RSOOn	234
7.2.10.	Ir_RSOff	235

7.2.11.	Ir_BreakOn	236
7.2.12.	Ir_BreakOff	237
7.2.13.	Ir_CheckCD	238
7.2.14.	Ir_CheckDR	239
7.2.15.	Ir_CheckCS	240
7.2.16.	Ir_CheckCI	241
7.2.17.	Ir_CheckBreak	242
7.2.18.	Ir_Err_Get	243
7.2.19.	Ir_State_Set	251
7.2.20.	Ir_SetPortConfig	253
7.2.21.	Ir_Init	255
7.2.22.	Ir_SetWinMode	256
8.	Bluetooth 通信部	257
8.1.	通信仕様	257
8.1.1.	通信インタフェース	257
8.1.2.	通信手順	257
8.2.	機能	258
8.2.1.	関数の状態遷移	258
8.2.2.	関数の実行手順	259
8.3.	エラー詳細	265
8.4.	ファンクション詳細	274
8.4.1.	BT_Start	275
8.4.2.	BT_Stop	276
8.4.3.	BT_GetLocalInfo	277
8.4.4.	BT_SetLocalInfo	278
8.4.5.	BT_Inquiry	279
8.4.6.	BT_GetDevInfo	280
8.4.7.	BT_GetDevName	281
8.4.8.	BT_SetPassKey	282
8.4.9.	BT_SelectDev	283
8.4.10.	BT_Open	284
8.4.11.	BT_Close	285
8.4.12.	BT_Read	286
8.4.13.	BT_Write	287
8.4.14.	BT_QueryRx	288
8.4.15.	BT_SaveDevInfo	289
8.4.16.	BT_LoadDevInfo	290
8.4.17.	BT_Err_Get	291
9.	通信ユーティリティ部関数	292
9.1.	通信インタフェース	292
9.1.1.	使用形態	292
9.1.2.	I Oボックスインタフェース	292
9.1.3.	排他制御	292
9.1.4.	転送ドライブ	293
9.2.	ソフトウェアブロック構成図	293
9.3.	マルチドロップ	294
9.3.1.	通信仕様	294
9.3.2.	ファイル送受信基本機能	297
9.4.	FLINKプロトコル機能	301
9.4.1.	通信仕様	301
9.4.2.	ファイル送受信基本機能	306

9.4.3.	リモート操作機能	313
9.4.4.	ファイルチェック機能関数	316
9.5.	DT500 プロトコル機能	317
9.5.1.	通信仕様	317
9.5.2.	ファイル送受信基本機能	320
9.5.3.	補足	323
9.6.	ファンクション詳細	324
9.7.	共通ファンクション	325
9.7.1.	cu_stopKeySet	326
9.7.2.	cu_setDrive	327
9.8.	マルチドロッププロトコル	328
9.8.1.	cu_open	329
9.8.2.	cu_fileSend	330
9.8.3.	cu_fileSendSet	333
9.8.4.	cu_fileSend1	334
9.8.5.	cu_fileRecv	335
9.8.6.	cu_msgSend	337
9.8.7.	cu_end	338
9.8.8.	cu_close	339
9.8.9.	cu_readErrStat	340
9.8.10.	cu_readDIRjInfo	344
9.9.	FLINK プロトコル	345
9.9.1.	cu_open	346
9.9.2.	cu_fileSend	348
9.9.3.	cu_fileAdd	350
9.9.4.	cu_fileRecv	351
9.9.5.	cu_close	352
9.9.6.	cu_readErrStat	353
9.9.7.	cu_idle	356
9.9.8.	cu_cmdRecv	357
9.9.9.	cu_fileDelete	358
9.9.10.	cu_fileMove	359
9.9.11.	cu_makeDir	360
9.9.12.	cu_getFileInfo	362
9.9.13.	cu_setFileInfo	364
9.9.14.	cu_getDiskInfo	365
9.9.15.	cu_dateTime	366
9.9.16.	cu_getSysInfo	367
9.9.17.	cu_msgSend	368
9.9.18.	cu_beep	369
9.9.19.	cu_setIoboxInfo	370
9.9.20.	cu_fchklog_Create	371
9.9.21.	cu_fchklog_Check	373
9.10.	DT500 プロトコル	375
9.10.1.	cu_open	376
9.10.2.	cu_fileSend	377
9.10.3.	cu_fileRecv	379
9.10.4.	cu_close	381
9.10.5.	cu_readErrStat	382
9.10.6.	cu_SetCode	385
10.	タイマ部	386

10.1.	機能	386
10.1.1.	タイマー部	386
10.1.2.	ブザー音鳴動部	386
10.1.3.	日付時刻制御部	387
10.2.	ファンクション詳細	388
10.2.1.	s_settimer	389
10.2.2.	s_timerend	390
10.2.3.	s_settimer2	391
10.2.4.	s_timerend2	392
10.2.5.	s_beep	393
10.2.6.	s_sound	394
10.2.7.	s_dateset	395
10.2.8.	s_dateget	396
10.2.9.	s_timeset	397
10.2.10.	s_timeget	398
11.	電源	399
11.1.	機能	399
11.1.1.	主電池電圧低下監視／警告	399
11.1.2.	副電池電圧低下監視／警告	399
11.1.3.	自動電源OFF制御 (APO: Auto Power Off)	399
11.1.4.	自動バックライトOFF制御 (ABO: Auto Backlight Off)	399
11.1.5.	低消費電力制御	399
11.1.6.	APO禁止／禁止解除	399
11.1.7.	電源通知モード設定／解除	400
11.1.8.	電源通知イベントクリア	400
11.1.9.	電源OFFコマンド	400
11.1.10.	I O B O X 起動	400
11.1.11.	バイブレータ制御	400
11.2.	ファンクション詳細	401
11.2.1.	pwr_hold_apo	402
11.2.2.	pwr_off	403
11.2.3.	pwr_loboxBootMode	404
11.2.4.	pwr_vibrator	405
12.	通知モード	406
12.1.	通知モードの概念	406
12.2.	通知モード使用時に必要となる関数	406
12.3.	通知モード使用例	407
12.3.1.	LBに対する通知モードの場合	407
12.3.2.	キーに対する通知モードの場合	410
12.4.	ファンクション詳細	412
12.4.1.	flg_sts	413
12.4.2.	clr_flg	414
12.4.3.	wai_flg	415
12.4.4.	pwr_inhabit	416
12.4.5.	pwr_inhabit_clr	417
13.	共通関数	418
13.1.	機能	418
13.1.1.	A B O R T 処理	418
13.1.2.	E X I T 処理	418
13.1.3.	動作環境メニュー起動処理	418

13.1.4.	OB R キャリブレーション起動処理	418
13.2.	ファンクション詳細	419
13.2.1.	abort	420
13.2.2.	exit	421
13.2.3.	wkup_cost	422
13.2.4.	wkup_calib	423
14.	プリンタ制御関数	424
14.1.	対象プリンタ	424
14.2.	使用方法	424
14.3.	通信インターフェース	424
14.4.	コマンドパケット/ステータスパケット仕様	424
14.5.	機能	424
14.5.1.	通信の接続準備	424
14.5.2.	通信のオープン・クローズ	425
14.5.3.	通信の送信・受信	425
14.5.4.	エラー処理	426
14.6.	ファンクション詳細	427
14.6.1.	Bluetooth のオープン	428
14.6.2.	Bluetooth のクローズ	429
14.6.3.	Bluetooth でのコマンド送信	430
14.6.4.	Bluetooth でのステータス受信	431
15.	参考資料	432
15.1.	機能比較	432

1. 概要

1.1. 提供ファイルについて

アプリケーションプログラムを作成する場合、必ず“HICIF. LIB”をリンクして下さい。

本機の関数を使用する場合には、本システムが提供する“BIOS1MAC. H”をアプリケーションプログラム内でインクルードして下さい。

また、C 標準ライブラリの機能を使用する場合には、“SHCLIB. LIB”をリンクして下さい。

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されていますので、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下さい。

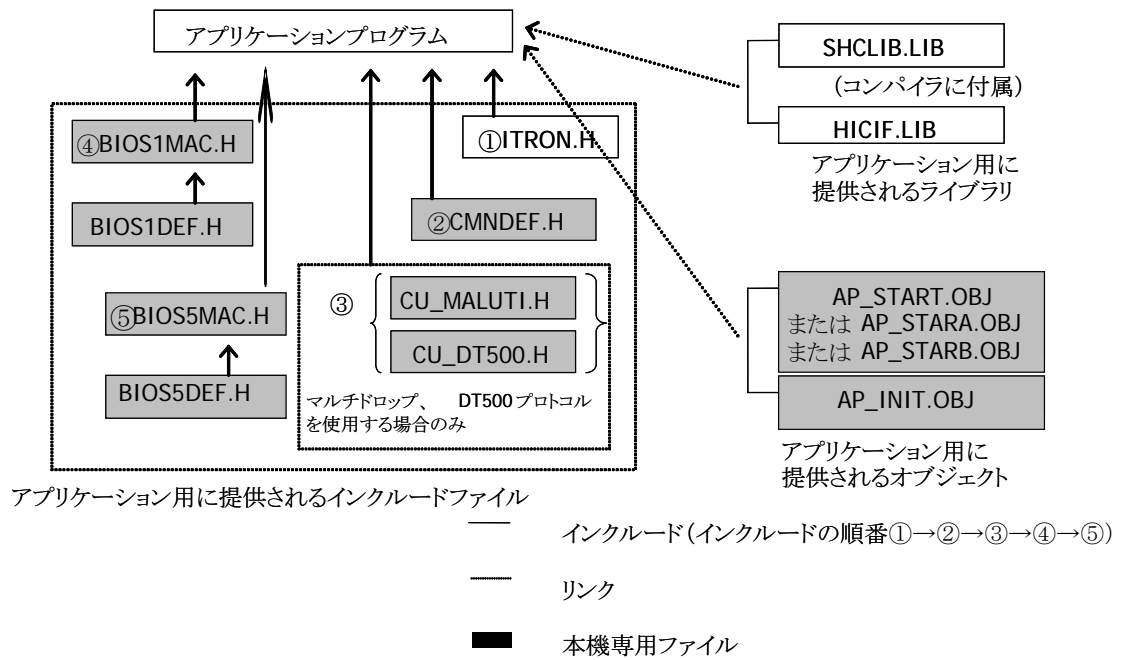
※ アプリケーションは提供された関数群(外部シンボル)の定義ファイルを用いることにより、単独でコンパイル/リンクします。(OS/システムの実体とはリンクしません)

(1)ファイルの内容

- BIOS1DEF. H ファンクションコール用 BIOS ジャンプテーブルの型定義をしたインクルードファイル
- BIOS1MAC. H BIOS のファンクションコールをマクロ定義したインクルードファイル
- SHCLIB. LIB C 言語標準ライブラリ (コンパイラに付属)
- AP_START. OBJ アプリケーション初期化モジュール
- AP_STARA. OBJ アプリケーション初期化モジュール (DT700 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 (DT700 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
- ITRON. H システム用のデータや関数を定義したインクルードファイル
- CMNDEF. H BIOS 用のデータや構造体等を定義したインクルードファイル
- HICIF. LIB C 言語標準ライブラリ
- CU_MULTI.H 通信プロトコル用インクルードファイル(マルチドロップ)
- CU_DT500.H 通信プロトコル用インクルードファイル(DT500)
- BIOS5MAC.H Bluetooth 通信用インクルードファイル

■ : 本機専用ファイル

(2) 各ファイルの関係について



1.2. 標準ライブラリ関数

下記の関数は、本システムで使用可能なC標準ライブラリの一覧です。
インタフェース仕様は、“SHシリーズC言語マニュアルライブラリ編”を参照して下さい。

NO.	ライブラリ名	適用
1	isalnum 関数	英字・10進数字の判定
2	isalpha 関数	英字の判定
3	isctrl 関数	制御文字の判定
4	isdigit 関数	10進数字の判定
5	isgraph 関数	印字文字の判定(空白除く)
6	islower 関数	英小文字の判定
7	isprint 関数	印字文字の判定(空白含む)
8	ispunct 関数	特殊文字の判定
9	isspace 関数	空白類文字の判定
10	isupper 関数	英大文字の判定
11	isxdigit 関数	16進数字の判定
12	tolower 関数	英大文字を英小文字に変換
13	toupper 関数	英小文字を英大文字に変換
14	setjmp 関数	現在実行中の関数の実行環境を指定した記憶域に待避
15	longjmp 関数	setjmp 関数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
16	acos 関数	浮動小数点数の逆余弦の計算
17	asin 関数	浮動小数点数の逆正弦の計算
18	atan 関数	浮動小数点数の逆正接の計算
19	atan2 関	浮動小数点どうしを除算した結果の逆正接の計算
20	cos 関数	浮動小数点数のラジアン値の余弦の計算
21	sin 関数	浮動小数点数のラジアン値の正弦の計算
22	tan 関数	浮動小数点数のラジアン値の正接の計算
23	cosh 関数	浮動小数点数の双曲線余弦の計算
24	sinh 関数	浮動小数点数の双曲線正弦の計算
25	tanh 関数	浮動小数点数の双曲線正接の計算
26	exp 関数	浮動小数点数の指数関数の計算
27	frexp 関数	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
28	ldexp 関数	浮動小数点数と2のべき乗の乗算の計算
29	log 関数	浮動小数点数と2のべき乗の乗算の計算
30	log10 関数	浮動小数点数の10を底とする対数の計算
31	modf 関数	浮動小数点数を整数部分と小数部分に分解
32	pow 関数	浮動小数点数のべき乗の計算
33	sqrt 関数	浮動小数点数の正の平方根の計算
34	ceil 関数	浮動小数点数の小数点以下を切り上げた整数値の取得
35	fabs 関数	浮動小数点数の絶対値の取得
36	floor 関数	浮動小数点数の小数点以下を切り捨てた整数値の取得
37	fmod 関数	浮動小数点どうしを除算した結果の余りの取得
38	va_start マクロ	可変個の引数を参照するため初期値設定
39	va_arg マクロ	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への参照を可能にする
40	va_end マクロ	可変個の引数を持つ関数の引数への参照を終了
41	fclose 関数	ファイルのクローズ
42	fopen 関数	指定ファイル名のファイルのオープン
43	freopen 関数	現在オープンされ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
44	sprintf 関数	データを書式に従って変換し、指定領域に出力
45	scanf 関数	指定領域からデータを入力し、書式に従って変換

NO.	ライブラリ名	適用
46	fread 関数	ファイルから指定領域にデータを入力
47	fwrite 関数	指定領域からデータをファイルに出力
48	fseek 関数	ファイルの現在の読み書き位置を移動
49	ftell 関数	ファイルの現在の読み書き位置を取得
50	rewind 関数	ファイルの現在の読み書き位置をファイル先頭に移動
51	atof 関数	数を表現する文字列を double 型の浮動小数点数に変換
52	atoi 関数	10進数を表現する文字列を int 型の整数値に変換
53	atol 関数	10進数を表現する文字列を long 型の整数値に変換
54	strtod 関数	数を表現する文字列を double 型の浮動小数点数に変換
55	strtol 関数	数を表現する文字列を long 型の整数値に変換
56	srand 関数	rand 関数で生成する疑似乱数列の初期値を設定
57	calloc 関数	記憶域を確保し、確保した全ての領域を0クリア
58	free 関数	指定した記憶域を解放
59	malloc 関数	記憶域を確保
60	realloc 関数	記憶域の大きさを指定した大きさに変更
61	abort 関数	プログラムの異常終了
62	exit 関数	プログラムの正常終了
63	bsearch 関数	2分割検索
64	qsort 関数	ソート
65	abs 関数	int 型整数の絶対値
66	div 関数	int 型整数の除算の商と余り
67	labs 関数	long 型整数の絶対値
68	ldiv 関数	long 型整数の除算の商と余り
69	memcpy 関数	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
70	strcpy 関数	複写元の文字列を複写先の記憶域に NULL も含めて複写
71	strncpy 関数	複写元の文字列を指定文字列分、複写先の記憶域に複写
72	strcat 関数	文字列の後に文字列を連結
73	memcmp 関数	指定された二つの記憶域の比較
74	strcmp 関数	指定された二つの文字列の比較
75	strncmp 関数	指定された二つの文字列を指定文字数分まで比較
76	memchr 関数	指定記憶域で、指定文字が最初現れる位置を検索
77	strchr 関数	指定文字列で、指定文字が最初に現れる位置を検索
78	strcspn 関数	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
79	strpbrk 関数	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
80	strrchr 関数	指定文字列で、指定文字が最後に現れる位置を検索
81	strspn 関数	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
82	strstr 関数	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
83	memset 関数	指定記憶域の先頭から指定文字を指定された文字数分、設定
84	strerror 関数	エラーメッセージを設定
85	strlen 関数	文字列の長さを取得
86	ferror 関数	ファイルがエラー状態であるかを判定
87	clearerr 関数	ファイルのエラー状態をクリア

1.3. 専用ライブラリ関数

1.3.1. データ管理部関数

NO	関数名	機能
	メモリ管理関数	
1	dat_mem_size	メモリ領域の空きサイズの取得
	ファイル管理関数	
2	fil_mkdir	ディレクトリの作成 (FATファイルモード)
3	fil_rmdir	ディレクトリの削除 (FATファイルモード)
4	fil_remove	ファイルの削除
5	fil_rename	ファイル名の変更/移動
6	fil_fstat	ファイルの日時・サイズ・属性の取得
7	fil_chsize	ファイルサイズの変更
8	fil_getsize	ファイル領域空きサイズの取得
9	fil_findfirst	ファイル名の取得
10	fil_findnext	ファイル名の取得(次候補)
11	fil_filesize	ファイルの個数と総サイズの取得
12	fil_filefind	ファイル全パス名の取得
13	dat_fdir	ファイル格納情報の取得(DT700 互換モード)
14	dat_fsize	ファイル空き領域サイズの取得(DT700 互換モード)
15	dat_fdel	ファイルの削除(DT700 互換モード)
16	dat_fname	ファイル名変更(DT700 互換モード)
17	dat_F_Search	ファイルデータの検索(DT700 互換モード)
18	システムデータ管理関数	
19	dat_system	システムデータの設定
20	dat_OSVer_Read	OSバージョン読出し
21	dat_dealer_chk	代理店IDのチェック
22	dat_Apload	APロード&実行
	低水準インタフェース関数	
23	open	ファイルオープン
24	close	ファイルクローズ
25	read	ファイルのリード
26	write	ファイルのライト
27	lseek	ファイルリード/ライト位置の設定
28	sbrk	メモリ領域の割当て

1.3.2. 表示部関数

NO	関数名	機能
1	lcd_cls	画面クリア
2	lcd_csr_set	カーソルタイプ設定
3	lcd_csr_put	カーソル位置設定
4	lcd_csr_get	カーソル位置読出し
5	lcd_char	1文字表示
6	lcd_string	文字列表示
7	lcd_string2	文字列表示2(スクロール抑制)
8	lcd_userstr	ユーザ文字列表示
9	lcd_line	直線描画
10	lcd_gaiji	外字フォント登録
11	lcd_usrfont	ユーザーフォントファイル登録
12	lcd_romfont	ROMフォント設定
13	lcd_led	LEDの制御
14	lcd_el	ELバックライトの制御

1.3.3. キー部関数

NO	関 数 名	機 能
	キー入力ファンクション	
1	key_read	1文字入力
2	key_string	文字列入力
3	key_num	数値入力
4	key_check	キーバッファのステータスチェック
5	key_clear	キーバッファのクリア
	ファンクションキー制御	
6	key_fnc	ファンクションキーコードの設定
7	key_fnc_mode	ファンクションキー通知モード設定
	入力設定	
8	key_select	キー入力モード設定

1.3.4. OBR部関数

NO	関 数 名	機 能
1	OBR_open	OBRオープン
2	OBR_close	OBRクローズ
3	OBR_getc	OBRデータ1文字リード
4	OBR_gets	OBRデータ文字列リード
5	OBR_flush	OBRバッファのクリア
6	OBR_stat	OBRバッファステータスチェック
7	OBR_moderd	OBR動作モードの取得
8	OBR_modewt	OBR動作モードの設定
9	OBR_chgbuf	OBRバッファの切替え
10	OBR_trigmode	トリガーキーによる電源オン設定
11	OBR_swing	レーザー発光幅の設定
12	OBR_widenarrow	レーザー発光幅の微調整
13	OBR_getadjust	バー幅補正モード取得処理
14	OBR_setadjust	バー幅補正モード設定処理
15	OBR_getmargincheck	マージンチェック倍率モード取得処理
16	OBR_setmargincheck	マージンチェック倍率モード設定処理

1.3.5. 通信部関数

NO	関数名	機能
	COM管理機能	
1	c_open	COMオープン
2	c_close	COMクローズ
3	c_status	COMステータスのリード
4	c_hold	COMの占有
5	c_chkopen	COMのオープンチェック
	送受信機能	
6	c_dout	n文字送信
7	c_din	1文字受信
8	c_tmddin	タイムアウト監視受信
9	c_out	1文字送信
10	c_break	ブレイク信号の制御
11	c_txx	送受信の有効/無効
12	c_timer	DR/CS/CDタイムアウト監視値の設定
13	c_er	ER信号の制御
14	c_rs	RS信号の制御
15	c_errs	ER/RS信号の制御
16	c_iobox	IOボックス送信設定
17	c_irout	IOボックス送信
	受信バッファ管理機能	
18	c_flush	受信バッファのクリア
19	c_bfsts	受信バッファステータスのリード
	通信補助機能	
20	c_errbfiring	エラーコードバッファリング制御の設定
21	c_rderrsts	エラーステータスのリード
22	c_chghdr	受信ハンドラ切替え
23	c_brkevent	ブレイク要因の設定

1.3.6. IrDA 部関数

NO	関数名	機能
1	Ir_Open	IrCOMM オープン
2	Ir_Close	IrCOMM クローズ
3	Ir_Read	データ読み込み
4	Ir_Write	データ書き込み
5	Ir_QueryTx	送信データ数問合せ
6	Ir_QueryRx	受信データ数問合せ
7	Ir_EROn	ER信号ON
8	Ir_EROff	ER信号OFF
9	Ir_RSON	RS信号ON
10	Ir_RSOff	RS信号OFF
11	Ir_BreakOn	ブレーク送信ON
12	Ir_BreakOff	ブレーク送信OFF
13	Ir_CheckCD	CD検査
14	Ir_CheckDR	DR検査
15	Ir_CheckCS	CS検査
16	Ir_CheckCI	CI検査
17	Ir_CheckBreak	BREAK検査
18	Ir_Err_Get	エラー値取得
19	Ir_State_Set	通信状態設定
20	Ir_SetPortConfig	自局能力設定
21	Ir_Init	IrCOMM 強制終了
22	Ir_SetWinMode	Ir モード切替設定

1.3.7. Bluetooth 部関数

NO	関数名	機能
1	BT_Start	Bluetooth 通信機能の使用を開始
2	BT_Stop	Bluetooth 通信機能の使用を終了
3	BT_GetLocalInfo	本体のデバイス情報の取得
4	BT_SetLocalInfo	本体のデバイス情報の設定
5	BT_Inquiry	Bluetooth 機器の問い合わせ
6	BT_GetDevInfo	Bluetooth 機器のデバイス情報の取得
7	BT_GetDevName	Bluetooth 機器のデバイス名の取得
8	BT_SetPassKey	Bluetooth パスキーの設定
9	BT_SelectDev	接続する Bluetooth 機器の指定
10	BT_Open	Bluetooth 通信の開始
11	BT_Close	Bluetooth 通信の終了
12	BT_Read	Bluetooth 通信のデータ受信
13	BT_Write	Bluetooth 通信のデータ送信
14	BT_QueryRx	読み込み可能なデータ数の取得
15	BT_SaveDevInfo	Bluetooth 機器のデバイス情報の保存
16	BT_LoadDevInfo	Bluetooth 機器のデバイス情報の読み出し
17	BT_Err_Get	エラー値の取得

1.3.8. 通信ユーティリティ部関数

NO	関数名	機能
	共通ファンクション	
1	cu_stopKeySet	中断キーの設定
2	cu_setDrive	転送ドライブ指定
	マルチドロッププロトコル	
3	cu_open	回線オープン
4	cu_fileSend	ファイル送信
5	cu_fileSendSet	ファイル送信情報設定
6	cu_fileSend1	1ファイル送信
7	cu_fileRecv	ファイル受信
8	cu_msgSend	画面表示メッセージ送信
9	cu_end	通信中断
10	cu_close	回線クローズ
11	cu_readErrStat	エラー詳細情報取得
12	cu_readDirInfo	データリンク拒否情報取得
	FLINK プロトコル	
13	cu_open	回線オープン(初期化)
14	cu_fileSend	ファイル送信
15	cu_fileAdd	ファイル追加
16	cu_fileRecv	ファイル受信
17	cu_close	回線クローズ
18	cu_readErrStat	エラー情報の取得
19	cu_idle	IDLE遷移
20	cu_cmdRecv	コマンド受信待ち
21	cu_fileDelete	ファイル削除
22	cu_fileMove	ファイル移動
23	cu_makeDir	ディレクトリ作成
24	cu_getFileInfo	ファイル情報の取得
25	cu_setFileInfo	ファイル情報の更新
26	cu_getDiskInfo	ディスク情報の取得
27	cu_dateTime	日付時刻の取得および設定
28	cu_getSysInfo	システム情報の取得
29	cu_msgSend	画面表示メッセージの送信
30	cu_beep	ブザー鳴動
31	cu_setIoboxInfo	IOボックス情報設定
32	cu_fchklog_Create	FCHK リストファイルの生成
33	cu_fchklog_Check	FCHK リストファイルのチェック
	DT500 プロトコル	
34	cu_open	回線オープン(初期化)
35	cu_fileSend	ファイル送信
36	cu_fileRecv	ファイル受信
37	cu_close	回線クローズ
38	cu_readErrStat	エラー詳細情報取得
39	cu_SetCode	DT500 プロトコル制御コード拡張設定

1.3.9. タイマー部関数

NO	関数名	機能
	タイマ登録	
1	s_settimer	タイマ1登録
2	s_timerend	タイマ1削除
3	s_settimer2	タイマ2登録
4	s_timerend2	タイマ2削除
	音発生	
5	s_beep	エラービープ音
6	s_sound	サウンド音
	日時設定	
7	s_dateget	日付の取得
8	s_dateset	日付の設定
9	s_timeget	時刻の取得
10	s_timeset	時刻の設定

1.3.10. 電源部関数

NO	関数名	機能
1	pwr_hold_apo	APO禁止設定
2	pwr_off	電源オフ
3	pwr_loboxBootMode	IO ボックス起動設定
4	pwr_vibrator	バイブレーション動作開始

1.3.11. 通知モード部関数

NO	関数名	機能
1	flg_sts	通知フラグ状態取得
2	clr_flg	通知フラグ状態クリア
3	wai_flg	フラグセット待ち
4	pwr_inhabit	通知モード設定
5	pwr_inhabit_clr	電源通知イベントのクリア

1.3.12. 共通関数

NO	関数名	機能
1	abort	ABORT処理
2	exit	EXIT処理
3	wkup_cost	動作環境メニューの起動
4	wkup_calib	OBRキャリブレーション処理起動

1.3.13. プリンタ制御関数

NO	関数名	機能
1	prn_BTinf_open	Bluetooth 通信のオープン
2	prn_BTinf_close	Bluetooth 通信のクローズ
3	prn_BTinf_send	Bluetooth でのコマンド送信
4	prn_BTinf_recvSts	Bluetooth でのステータス受信

1.4. データタイプ

本システムで使用するパラメータのデータタイプとサイズを以下に示します。

typedef	char	B ;	/* 符号付き8ビット整数	*/
typedef	short	H ;	/* 符号付き16ビット整数	*/
typedef	long	W ;	/* 符号付き32ビット整数	*/
typedef	unsigned char	UB ;	/* 符号なし8ビット整数	*/
typedef	unsigned short	UH ;	/* 符号なし16ビット整数	*/
typedef	unsigned long	UW ;	/* 符号なし32ビット整数	*/
typedef	char	VB ;	/* データタイプが一定しない(8ビットサイズ)	*/
typedef	short	VH ;	/* データタイプが一定しない(16ビットサイズ)	*/
typedef	long	VW ;	/* データタイプが一定しない(32ビットサイズ)	*/
typedef	void	*VP;	/* データタイプが一定しないものへのポインタ	*/
typedef	void	(*FP)();	/* プログラム先頭アドレス	*/
typedef	H	ID ;	/* オブジェクトID	*/
typedef	W	ER ;	/* エラーコード	*/
typedef	W	FN ;	/* 機能コード	*/

2. データ管理部

2.1. 機能

2.1.1. メモリー管理機能

要求サイズ分のメモリを、メモリ領域の下位アドレスから連続した領域に割り付けます。割り付けるメモリが、不足または、要求メモリサイズが0の場合はエラーを返します。

表2. 1 サポートしている標準ライブラリ関数の一覧

関数名	処理概要
calloc	記憶域を確保し、確保した全ての領域を0クリアします
free	指定した記憶域を解放します
malloc	記憶域を確保します
realloc	記憶域の大きさを指定した大きさに変更します

2.1.2. ファイル管理機能

本システムのファイル管理機能は、データファイルに関してのみで、標準入出力ファイル(コンソール、プリンタ、ディスクファイル等)のサポートは行いません。

標準ライブラリ関数の詳細は、「SHシリーズC言語マニュアル」ライブラリ編<stdio. h>を参照して下さい。

表2. 2 サポートしている標準ライブラリ関数の一覧

関数名	処理概要
fclose	ファイルをクローズします
fopen	指定ファイル名のファイルをオープンします
freopen	現在オープンされ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します
fread	ファイルから指定領域にデータを入力します
fwrite	指定領域からデータをファイルに出力します
fseek	ファイルの現在の読み書き位置を移動します
ftell	ファイルの現在の読み書き位置を求めます
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します
clearerr	ファイルのエラー状態をクリアします

2.1.3. データ管理機能(システムデータ管理)

システムデータとは、システムメニューで設定可能な動作環境です。

各関連システムデータ毎に一括書込み、および読出しが可能です。また、アプリケーションプログラムでの動作環境の設定が可能です。

表2. 3 サポートしている関数の一覧

関数名	処理概要
dat_system	システムデータの書込み/読込み処理
dat_OSVer_Read	OSバージョン読出し処理
dat_dealer_chk	代理店IDのチェック
dat_Apload	APロード&実行
dat_mem_size	メモリ領域の空きサイズの取得

表2. 4 システムデータ一覧

項目	管理データ	内 容	初期化タイミング		
			リセット立上げ	configファイル反映	
電源	APO	APO 時間:0~59(分)	○	○	
	ABO	ABO 時間:10~59(秒)	○	○	
	レジューム	ON/OFF の設定	○	○	
KEY	クリック音	ON/OFF	○	○	
表示	フォントMODE	6/8/10(dot)	○	○	
	フォント種別	NORMAL/BOLD	○	○	
	日本語/英語	日本語/英語	FROM 内容反映	○	
	コントラスト設定値	0~15	○	○	
	コントラスト手動差分	手動設定による現在設定値との差分	○	—	
OBR	読取り回数	パージコート連続読取回数:1~9(回)	○	○	
	照合回数	読取コート照合回数:1~9(回)	○	○	
	スキャン時間	スキャンタイムアウト時間:1~9(秒)	○	○	
通信	共通	プロトコル	マルチロップ ^o /FLINK/DT500	○	○
		通信 PORT	各種通信における PORT	○	○
	マルチロップ ^o または FLINK または DT500	速度(IrDA/カシオ IR インタフェース)	2400~115200(bps)	○	○
		データ(IrDA/カシオ IR インタフェース)	CHAR_8/7	○	○
		パリティ(IrDA/カシオ IR インタフェース)	EVN/ODD/NON	○	○
		STOP(IrDA/カシオ IR インタフェース)	STOP_1/2	○	○
		速度(シリアルインタフェース)	1200~115200(bps)	○	○
		データ(シリアルインタフェース)	CHAR_8/7	○	○
		パリティ(シリアルインタフェース)	EVN/ODD/NON	○	○
		STOP(シリアルインタフェース)	STOP_1/2	○	○
		速度(PHS インタフェース)	1200~38400(bps)	○	○
		データ(PHS インタフェース)	CHAR_8/7	○	○
		パリティ(PHS インタフェース)	EVN/ODD/NON	○	○
		STOP(PHS インタフェース)	STOP_1/2	○	○
タイマ	音量	OFF/小/中/大	○	○	
システム	機器 ID	機器固有の ID (6 桁の数字)	FROM 内容反映	config.id	
	代理店ID	代理店ID(6 桁の英記号)	—	config.pas	
	OS バージョン	OS バージョン(6 桁)	—	—	
	PATCH バージョン	PATCH バージョン(6 桁)	—	—	
	機器種別	MODEL 情報	IOポート反映	—	
プロトコル	マルチロップ ^o	受信タイムアウト	0~99(秒)	○	○
		リトライ回数	0~99(回)	○	○
		リンクタイムアウト	0~9990(ミリ秒)	○	○
	FLINK	セッション確立タイムアウト	0~3600(秒)	○	○
		受信タイムアウト	0~600	○	○
		セッション終了タイムアウト	0~600	○	○
	DT500	シリアル NO	ON/OFF	○	○
		水平パリティ	ON/OFF	○	○
リンクタイムアウト		0~240	○	○	
サイズ	アプリケーション領域サイズ	アプリケーション領域サイズ	RAMドライブ ^o 情報反映	指定時○	
ファイルモード	FORMAT タイプ	DT700/FAT ファイルシステム	FORMAT 情報反映	指定時○	

表2.5 システムデータ設定範囲

項目	管理データ	サイズ	設定範囲	初期値	備考	
電源	APO	LONG(32 bit)	0~59	10		
	ABO	LONG(32 bit)	10~59	15		
	レジューム	LONG(32 bit)	0 or 1	1	ON:1 OFF:0	
KEY	クリック音	LONG(32 bit)	0 or 1	1	ON:1 OFF:0	
表示	フォント MODE	LONG(32 bit)	0 or 1 or 2	1	6DOT:0 8DOT:1 10DOT:2	
	フォント種別	LONG(32 bit)	0 or 1	0	NORMAL:0 BOLD:1	
	日本語/英語	LONG(32 bit)	0 or 1	0	日本語:0 英語:1	
	コントラスト設定値	LONG(32 bit)	0~15	7		
	コントラスト手動差分	LONG(32 bit)	-7~7	0		
OBR	読取り回数	LONG(32 bit)	1~9	1		
	照合回数	LONG(32 bit)	1~9	3		
	スキャン時間	LONG(32 bit)	1~9	4		
通信	共通	プロトコル	LONG(32 bit)	0 or 1 or 2	1	マルチ:0 FLINK:1 DT500:2
		通信 PORT	LONG(32 bit)	0 or 1 or 2	0	IR:0 14P:1 18P:2
	マルチプロップ または FLINK または DT500	速度	LONG(32 bit)	2400~115200	B_115200	B_2400~B_115200 DEFINE 使用
(カシオ IR インタフェース/IrDA)						
データ		LONG(32 bit)	CHAR_8/7	CHAR_8		
(カシオ IR インタフェース/IrDA)						
パリティ		LONG(32 bit)	EVN/ODD/ NON	PARI_NO N		
(カシオ IR インタフェース/IrDA)						
STOP		LONG(32 bit)	STOP_1/2	STOP_1		
(カシオ IR インタフェース/IrDA)						
速度		LONG(32 bit)	1200~115200	B_115200	B_1200~B_115200 DEFINE 使用	
(シリアルインタフェース)						
データ		LONG(32 bit)	CHAR_8/7	CHAR_8		
(シリアルインタフェース)						
パリティ		LONG(32 bit)	EVN/ODD/ NON	PARI_NO N		
(シリアルインタフェース)						

		STOP	LONG(32 bit)	STOP_1/2	STOP_1	
		(シリアルインタフェース)				
		速度(PHS インタフェース)	LONG(32 bit)	1200～38400	B_38400	B_1200～B_38400 DEFINE 使用
		データ(PHS インタフェース)	LONG(32 bit)	CHAR_8/7	CHAR_8	
		パリティ(PHS インタフェース)	LONG(32 bit)	EVN/ODD/NON	PARL_NO N	
		STOP(PHS インタフェース)	LONG(32 bit)	STOP_1/2	STOP_1	
タイマ		音量	LONG(32 bit)	0 or 1 or 2 or 3	2	OFF:0 LOW:1 MID:2 LOUD:3
システム		機器 ID	BYTE x 6	6桁の数字	未登録	
		代理店ID	BYTE x 6	6桁の英数字	未登録	
		OSバージョン	BYTE x 6	6桁の英数字	OS依存	
		PATCHバージョン	BYTE x 6	6桁の英数字	OS依存	ex: V1.00A
		機器種別	LONG(32 bit)		機種依存	
プロトコル	マルチト ップ	受信タイムアウト	LONG(32 bit)	0～99	3	
		リトライ回数	LONG(32 bit)	0～99	3	
		リンクタイムアウト	LONG(32 bit)	0～9990	30	
	FLINK	セッション確立タイムアウト	LONG(32 bit)	0～3600	1800	
		受信タイムアウト	LONG(32 bit)	0～600	300	
		セッション終了タイムアウト	LONG(32 bit)	0～600	10	
	DT500	シリアル NO	LONG(32 bit)	0 or 1	0	ON:1 OFF:0
		水平パリティ	LONG(32 bit)	0 or 1	0	ON:1 OFF:0
		リンクタイムアウト	LONG(32 bit)	0～240	0	
サイズ		AP 領域サイズ	LONG(32 bit)	128～992	336	KB単位
ファイルモード		FORMAT	LONG(32 bit)	0 or 1		0:FAT モード 1:DT700 モード

※ 網掛け部分は、従来機 DT900 用の設定です。DT930 では機能しません。

2.1.4. 低水準インタフェース

低水準インタフェースでは、ファイルについての処理(状態管理、書込み/読込み等)および、メモリ管理を行います。

以下に低水準で提供する関数についての機能を示します。

(1) openルーチン

引数で指定されたファイルのチェックを行い、正常な場合ハンドルNo. を返します。

以後のコマンド要求に対する妥当性をmodeにより判断します。

modeはOR指定し、読み/書きモード(bit0~2)はいずれかを必ず指定します。

mode ()

3 1 7 6 5 4 3 2 1 0
□□~□□□□□□□□

└─ O_RDONLY: 1=読出し専用にします。

└─ O_WRONLY: 1=書込み専用にします。

└─ O_RDWR: 1=読込み/書込み両用にします。

└─ O_CREAT: 1=ファイルが存在しない場合は、新規ファイルとして作成します。

└─ O_TRUNC: 1=ファイルが存在する場合、ファイル内容を捨て、ファイルサイズを0にします。

└─ O_APPEND: 読込み/書込み位置の設定をします。

0=ファイル先頭に設定

1=ファイル最後に設定

(2) closeルーチン

引数で指定されたハンドルNo. に対応したファイルのクローズ処理を行います。

(3) readルーチン

引数で指定されたハンドルNo. のファイルに対し、現在の読込み位置よりファイルデータを指定バッファへ転送します。転送後の読込み位置は、読出しバイト数だけ先に進みます。

(4) writeルーチン

引数で指定されたハンドルNo. のファイルに対し、書込みデータバッファの内容を現在の書込み位置より書込みを開始します。書込み終了後、ファイル管理テーブルの書込み位置を書込みバイト数だけ先に進めます。

(5) lseekルーチン

引数で指定されたハンドルNo. のファイルに対し、読込み/書込み位置をバイト単位で要求位置に変更します。

要求位置が負または、ファイルサイズをこえる場合はエラーとします。

読込み/書込み位置の設定は、引数により以下のように決定します。

第2引数 offset : 読込み/書込みの位置を示すオフセット(バイト単位)。

第3引数 base : オフセットの起点。

- base が0のとき: ファイルの先頭から offset バイトの位置に設定します。
- base が1のとき: 現在の位置に offset バイトを加えた位置に設定します。
- base が2のとき: ファイルサイズに offset バイトを加えた位置に設定します。

(6) sbrkルーチン

要求されたデータ領域を割り付け、正常に領域が確保できたとき先頭アドレスを返します。

(注意事項)

誤動作の原因となりますので、高水準関数との併用は避けて下さい。

2.1.5. ファイル情報サーチインタフェース

(1) ファイルシステムモード

本機においては、従来からのファイルシステムを踏襲するDT700 互換モードファイルシステムとFATシステムを用い MS-DOS 互換をサポートしているFATファイルシステムの2モードをサポートします。

ただし、これら2つのモードはシステムとして混在するのではなく、排他的に動作します。

また、モード切り替えは、システムメニューにより行なうことができるが、切り替えた段階で以前のモードで作成した

ファイルは領域再構築を行なうため、情報は全て消去されます。

以下に各モード機能概要を示します。

① DT700 互換ファイルシステムモード(Aドライブ)

従来のDT700同様、ファイル領域は全てリニア空間で動作します。従って、ファイルの追加書込みの際は、領域の再構成処理を行います。

また、従来からのファイル格納情報の取得関数(dat_fdir)を用いてのファイル内容の直接参照機能が可能です。

ファイル領域構成、機能詳細をそれぞれ以下に示します。

図2. 1 ファイル領域構成

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バイト単位に持つ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		

表2. 6 DT700互換モード機能詳細

機能項目	内 容
ドライブの概念	あり(ただし、従来のドライブ指定なしの記述の場合は、RAMドライブとして扱います)
ファイル数	最大 192個
同時オープン数	16
ディレクトリ	ディレクトリは従来と同様に未サポート
総容量	1.39MB(738KB-1.6MB 可変)
セクタサイズ	256
セクタ/クラスタ	1
予約セクタ	なし
メディアディスクリプタ	なし
セクタ/FAT	(FATの概念なし)
セクタ/トラック	0
ヘッド番号	0
総セクタ数	5576(2952~6410)

その他、本モードでは、ファイル属性を待ちません。

②FATファイルシステムモード(Aドライブ)

FATシステムを用いた MS-DOS 互換ファイルシステムです。このためファイル領域内の空間はリニア構造にはならず、ファイル領域の直接参照はできません(参照データは、保証されません)。ファイル領域構成、機能詳細をそれぞれ以下に示します。

図2. 2 ファイル領域構成

BPB	FAT	ディレクトリ情報	ファイルデータ	チェックサム
-----	-----	----------	---------	--------

チェックサム領域 : チェックサム格納領域(セクタ単位に4バイト加算値)

ファイルデータ : ファイル部データ領域

表2. 7 FATファイルモード機能詳細

機能項目	内 容
ドライブの概念	あり(RAMドライブ:Aドライブ、AP領域:Cドライブ)
ファイル数	ルート:192個、ディレクトリ配下:無制限(ファイル領域が許す限り)
同時オープン数	16
ディレクトリ	サポート
総容量	738KB~1.6MB (デフォルト:1.39MB)
セクタサイズ	512
セクタ/クラスタ	2
予約セクタ	1
メディアディスクリブタ	1
セクタ/FAT	3~5 (デフォルト:5)
セクタ/トラック	0
ヘッド番号	0
総セクタ数	1476~3205 (デフォルト:2788)

③フラッシュディスク(Bドライブ)

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。本体の電源が無い場合でも、データは常に保存されます。

ファイル領域構成、機能詳細をそれぞれ以下に示します。

図2.3 ファイル領域構成

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(最大約64Kb でファイルデータを分割します)

チェックサム : 分割されたブロック単位に4バイト加算値

表2.8 フラッシュディスク機能詳細

機能項目	内 容
ドライブの概念	あり(FLASHドライブ:Bドライブ)
ファイル数	ルート:64個、ディレクトリ無し
同時オープン数	16
ディレクトリ	未サポート
総容量	12.5MB
セクタサイズ	(セクタの概念なし)
セクタ/クラスタ	なし
予約セクタ	なし
メディアディスクリプタ	なし
セクタ/FAT	(FAT の概念なし)
セクタ/トラック	0
ヘッド番号	0
総セクタ数	(セクタの概念なし)

(2)機能一覧

表2. 8 機能一覧

関数名	内 容	ファイルシステムモード	
		DT700 互換	FAT
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	○	○
dat_del	ファイルの削除	○	×
fil_rename	ファイル名の変更または移動	○	○
dat_fname	ファイル名の変更	○	×
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	ドライブの空き容量の取得	○	○
dat_fsize	ドライブの空き容量の取得	○	×
fil_findfirst	ファイル検索	○	○
fil_findnext	検索ファイルの次候補の読出し	○	○
fil_filesize	ファイルの個数・総サイズの取得	○	○
fil_filefind	ファイルの全パス名の取得	○	○
dat_fdir	ファイル格納情報の取得	○	×
dat_F_Search	ファイルデータの高速検索	○	×
dat_Apload	APローディング&実行	○	○
Hash×××	高速ファイルサーチライブラリ(オブジェクト提供)	×	○

○：機能サポート ×：機能未サポート

備考：

- DT700互換の場合、ドライブおよびディレクトリの概念が無いため、ルートディレクトリのみ有効です。
- DT700互換の場合、ファイル属性を持たないため、ファイル名の取得 (fil_findfirst) では、ファイル属性は検索条件対象外です。
- FATファイルシステムの場合、ファイルデータの高速検索は、Hashライブラリ(オブジェクト提供)で行います。
(DT700互換モードでは、使用しないでください)

2.1.6. 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、以下の3種類があります。

(1) configファイル

ファイル形式: DOSファイル

ファイル名 : CONFIG. HTS

ファイルサイズ : 詳細は次ページ参照。

ファイル内容: 詳細は次ページ参照。

※CONFIG. HTSはテキスト形式の文字列で構成されており、20H以下のコードは無視します。

(2) 機器IDファイル

ファイル形式: DOSファイル

ファイル名 : CONFIG. ID

ファイルサイズ : 6バイト

ファイル内容: ASCIIコードによる機器ID番号(6桁の半角数字)

(3) 代理店IDファイル

ファイル形式: DOSファイル

ファイル名 : CONFIG. PAS

ファイルサイズ : 6バイト

ファイル内容: ASCIIコードによる代理店ID番号(6桁の半角英数字)

CONFIG. HTS形式

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 / 01:シリアルインタフェース / 02:PHS インタフェース	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	
		速度 (シリアルインタフェース)	+44	2	01:1200 / 02:2400 / 03:4800 / 04:9600 / 05:19200 / 06:38400 / 07:57600 / 08:115200(bps)	08	
		データ (シリアルインタフェース)	+46	2	07:7bit / 08:8bit	08	
		パリティ (シリアルインタフェース)	+48	2	00:NON / 01:EVEN / 02:ODD	00	
		STOP (シリアルインタフェース)	+50	2	00:1bit / 01:2bit	00	
		速度 (PHS インタフェース)	+52	2	01:1200 / 02:2400 / 03:4800 / 04:9600 / 05:19200 / 06:38400(bps)	06	
		データ (PHS インタフェース)	+54	2	07:7bit / 08:8bit	08	
		パリティ (PHS インタフェース)	+56	2	00:NON / 01:EVEN / 02:ODD	00	
		STOP (PHS インタフェース)	+58	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

※ 網掛け部分は、従来機 DT900 用の設定です。DT930 では機能しません。

2.2. ファンクション詳細

ファンクション詳細を次ページより示します。

2.2.1. 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を登録または読出します。引数の機能コード、システムデータ識別IDが下記以外の場合は、何もせずにエラー終了とします。登録を行う際には、データ毎の妥当性を確認し不当データの場合、全てデータ登録をせずにエラー終了とします。

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

パラメータ

func

機能コード

SYSD_FNC_READ	読出し
SYSD_FNC_WRITE	登録

sys_id

システムデータ識別 ID

SYSD_PWR	電源	
SYSD_KEY	KEY	
SYSD_OBR	OBR	
SYSD_DSP	表示	(DT700 互換モード)
SYSD_DSP2		(本機オリジナル)
SYSD_COM	通信	(共通)
SYSD_COM0	IrDA/カシオ	IR インタフェース
SYSD_COM1	予約	
SYSD_COM3	予約	
SYSD_TIM	タイマ	
SYSD_SYS	システム	(DT700 互換モード)
SYSD_SYS2		(本機オリジナル)
SYSD_PRO	プロトコル	

sys_dt

登録/読出しデータバッファのポインタ
(構造体の詳細については次項を参照してください)

戻り値

関数結果

E_OK	正常終了
E_PRM	パラメータエラー
E_NG	登録データエラー

補足

システム関連データの BIOS バージョンと機器種別は、変更できません。

dat__systemの設定／読出しのデータバッファ詳細

【電源関連】

```
typedef struct sys_pwr {
    w   apo;          /* APO時間設定 0~59(分) */
    w   abo;          /* ABO時間設定 10~59(秒) */
    w   res_md;       /* レジューム ON/OFF */
                                /* RESUME_ON : レジュームON */
                                /* RESUME_OFF : レジュームOFF */
} DAT_PWR_STR;
```

【KEY関連】

```
typedef struct sys_key {
    w   clk_md;       /* クリック音 ON/OFF */
                                /* CLICK_ON : クリック音ON */
                                /* CLICK_OFF : クリック音OFF */
} DAT_KEY_STR;
```

【OBR関連】

```
typedef struct sys_obr {
    w   rd_ct;        /* 読み取り回数 1~9(回) */
    w   cmp_ct;       /* 照合回数 1~9(回) */
    w   scn_tm;       /* スキャン時間 1~9(秒) */
} DAT_OBR_STR;
```

【表示関連】

```
typedef struct sys_disp {
    w   font_md;      /* 6 / 8 / 10ドットモード */
                                /* 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 */
    w   lang_md;      /* 日本語 / 英語モード */
                                /* JPN_SET : 日本語モード */
                                /* ENG_SET : 英語モード */
} DAT_DSP_STR;

typedef struct sys_disp2 {
    w   font_kd;      /* NORMAL / BOLD */
                                /* FONT_NORMAL : NORMAL */
                                /* FONT_BOLD : BOLD */
    w   cont_md;      /* コントラスト設定値 0~15 */
    w   cont_df;      /* コントラスト差分 (0 固定) 変更不可 */
} DAT_DSP_STR2;
```

【通信関連】

```
typedef struct sys_tty0{
    w    com_proto;    /* プロトコル種別          */
                       /* PRT_MULTI : マルチドロップ */
                       /* PRT_FLINK : F L I N K      */
                       /* PRT_DT500 : D T 5 0 0     */
    w    com_port;    /* 通信ポート              */
                       /* IR_PORT : I Rポート        */
}DAT_COMINF_STR;

typedef struct sys_tty{
    w    speed;       /* 転送速度                */
                       /* B_115200                 */
                       /* B_57600                   */
                       /* B_38400                    */
                       /* B_19200                     */
                       /* B_9600                      */
                       /* B_4800                       */
                       /* B_2400                       */
    w    length;     /* データ長                */
                       /* CHAR_8                     */
                       /* CHAR_7                      */
    w    parity;     /* パリティビット          */
                       /* PARI_NON : なし           */
                       /* PARI_ODD : 奇数           */
                       /* PARI_EVN : 偶数           */
    w    stop_bit;   /* ストップビット          */
                       /* STOP_1                     */
                       /* STOP_2                     */
}DAT_COM_STR;
```

【タイマ関連】

```
typedef struct sys_time{
    w    buzzer;     /* ブザー音量              */
                       /* BUZZ_OFF : 音量OFF        */
                       /* BUZZ_LOW : 音量小         */
                       /* BUZZ_MID : 音量中         */
                       /* BUZZ_LOUD : 音量大        */
}DAT_TIM_STR;
```

【システム関連】

```
typedef struct sys_dat{
    UB  sys_id [7] ; /* 機器 I D (Read only) */
    UB  bios_ver [7] ; /* BIOSバージョン (Read only) */
    UW  mac_type; /* 機器種別 (O固定) (Read only)*/
}DAT_SYS_STR;

typedef struct sys_dat2{
    UB  dlr_id [7] ; /* 代理店 I D (Write only)参照不可 */
    UB  patch_ver [7] ; /* パッチバージョン(Read only) */
}DAT_SYS_STR2;
```

【プロトコル関連】

```
typedef struct sys_pro{
    /* マルチドロップ用 */
    w  non_rec_tmout; /* 通常受信タイムアウト 0~99(秒) */
    w  non_retry_ct; /* 通常リトライ回数 0~99(回) */
    w  mal_rec_tmout; /* マルチドロップ受信タイムアウト 0~9990(秒) */
    w  ptp_snd_tmout; /* 予約領域 */
    w  ptp_rec_tmout; /* 予約領域 */
    w  ptp_rec_retry_ct; /* 予約領域 */

    /* F L I N K */
    w  irda_tmout; /* IrDAセッション確立タイムアウト 0~3600(秒) */
    w  irda_rec_tmout; /* IrDA 受信タイムアウト 0~600(秒) */
    w  dr_tmout; /* IrDA rセッション終了タイムアウト 0~600(秒) */
    w  cs_tmout; /* 予約領域 */
    w  cd_tmout; /* 予約領域 */

    /* DT500 */
    w  s i r i a l _no /* シリアルNo 0:OFF/1:ON */
    w  level_parity /* 水平パリティ 0:OFF/1:ON */
    w  bht_tmout /* DT500 フォトリンク確立タイムアウト時間 */
    /* 0~240(秒) */

}DAT_PRO_STR;
```

2.2.2. dat_OSVer_Read

現在、登録されているFROM OSバージョンを指定バッファ(16バイト)に読出します。
データのフォーマットは、ASCIIコードで以下のようになります。

```
” * . * * * * △ * * . * * . * * △ ”  
バージョンNO 年 月 日  
△:スペース
```

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

パラメータ

rd_buf

OSバージョン格納バッファポインタ(16バイトの領域が必要です)

戻り値

なし

補足

2.2.3. dat_dealer_chk

代理店IDのチェックを行います。
アプリケーションの不正コピー防止用に使用します。

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

パラメータ

dealer_no

代理店ID格納領域のアドレス

戻り値

関数結果

E_OK	ID一致
E_NG	ID不一致

補足

代理店IDは、6桁の半角英数字で構成されています。

2.2.4. dat_Apload

指定されたファイルを読み出し、APファイルとしてAP領域にロードし実行します。

```
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

関数結果

E_NG

異常終了

補足

2.2.5. open

<低水準インタフェース関数>

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。

ファイル操作は、オープン時に返されるファイル番号を指定することにより、modeで指定したファイルモードに従って行われます。

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

パラメータ

name

指定ファイル名の格納先ポインタ

openの指定ファイル名格納先ポインタ指定方法の詳細

ファイルシステムモードにより以下の形式の指定が可能です。

(形式1) nnnnnnnn.mmm

ファイル名 拡張子

(形式2) d:¥ pppppppp ¥ nnnnnnnn.mmm

ドライブ名:パス名¥ ファイル名. 拡張子

注 ファイル名の有効データ(ANKコードのみ)

ファイル名の先頭コードは80H以上にしないで下さい。

	DT700互換モード	FAT ファイルモード
形式1	指定可能	指定可能
形式2	ルートディレクトリのみ指定可能 (パス名は指定不可)	指定可能

mode

ファイルモード

O_RDONLY (B0)	1の時、読み専用
O_WRONLY (B1)	1の時、書き専用
O_RDWR (B2)	1の時、読み／書き両用
O_CREAT (B3)	1の時、ファイル新規作成
O_TRUNC (B4)	1の時、指定ファイルの内容を捨て、サイズを0にします
O_APPEND (B5)	次に読み書きを行うファイル内の位置を設定します (0:ファイルの先頭に設定 / 1:ファイルの最後に設定)

戻り値

正常終了時、オープンしたファイル番号を返します。(0~15)

異常終了時、関数結果を返します

設定可能ファイルモード

FAT モード時 (下記以外はエラーとなります)

BIT	O_RDONLY	O_WRONLY	O_RDWR	O_CREAT	O_TRUNC	O_APPEND	標準関数
000001	○						rb
011010		○		○	○		wb
101010		○		○		○	ab
000100			○				r+b

011100			○	○	○		w+b
101100			○	○		○	a+b

DT-700 モード時

ファイルモード=00h/08h/10h/18h/20h/28h/30h/38h の場合エラーとなります。

関数結果

E_LOWERR

異常終了

補足

2.2.6. close

<低水準インタフェース関数>

指定ファイルをクローズし、ファイルの日付／時刻を登録します。

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

パラメータ

fileno

クローズするファイル番号

戻り値

関数結果

E_OK

正常終了

E_LOWERR

異常終了

補足

2.2.7. read

<低水準インタフェース関数>

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み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。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とします。

読出し位置は、読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サム値のチェックを行い、正しくない場合には、異常終了します。

```
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

buf

読込み領域のポインタ

count

読込みデータの要求バイト数

戻り値

正常終了時は、実際に読込まれたデータバイト数

異常終了時は、関数結果を返します

関数結果

E_LOWERR

異常終了

・チェックサム異常

・ファイル未オープン

補足

書込み専用モードファイルに対し本関数を実行した場合、リターンパラメータ値は E_LOWERR を返します。

2.2.8. write

<低水準インタフェース関数>

ファイルにデータを書込みます。書込み位置は、書込めたデータ数だけ先に進みます。

正常終了した場合は、実際に書込めたデータバイト数を返します。

書込み途中でファイルデータ領域が満杯になった場合も正常終了します。

連続して戻り値が0となるような場合、満杯状態と判断して異常終了します。

```
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

buf

書込み領域のポインタ

count

書込みデータの要求バイト数

戻り値

正常終了時は、実際に書込まれたデータバイト数

異常終了時は、関数結果を返します

関数結果

E_LOWERR

異常終了

・書込み異常

・ファイル未オープン

補足

読み専用モードファイルに対し本関数を実行した場合、リターンパラメータ値は E_LOWERR を返します。

2.2.9. lseek

<低水準インタフェース関数>

指定ファイルの読み／書き位置をバイト単位で設定します。

正常終了した場合は、ファイルの先頭からオフセットを返します。

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

パラメータ

fileno

対象のファイル番号

offset

読み／書き位置の変更先

base で指定された位置からのオフセット値 (バイト単位)

base

0の時、ファイルの先頭を基準とします

1の時、現在の読み／書き位置を基準とします

2の時、ファイルの最後を基準とします

戻り値

正常終了時は、変更した位置情報を返します

ファイルの先頭からオフセットアドレス(バイト単位)

- ・更新した書き込み位置が負の場合、現在位置は更新されません
 - ・更新した書き込み位置がファイルサイズを超えた場合、現在位置は更新されません
- 異常終了時は、リターンコードを返します

関数結果

E_LOWERR

異常終了

- ・ファイル未オープン
- ・更新した読み／書き位置が負
- ・更新した読み／書き位置がファイルサイズを超えた

補足

2.2.10. sbrk

<低水準インタフェース関数>

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(1~16KBバイト)

戻り値

正常終了の場合、割り付けた領域の先頭アドレスを設定

異常終了の場合、リターンコードを設定

関数結果

E_LOWERR

異常終了

- 割り付けるメモリ領域不足
- 要求サイズが0

補足

2.2.11. dat_fdir (DT700 互換モード専用)

第1引数で指定された位置のファイル格納情報を第2引数で指定された領域にファイル格納情報を設定します。

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

パラメータ

id

ファイル管理テーブルの読み込み位置指定

DAT_FILE_TOP ファイル管理テーブルの先頭から読み出します
 DAT_FILE_NEXT 次のファイル管理テーブルを読み出します

buf

ファイル格納情報の読み込み領域

ファイル格納情報のデータ

```
typedef struct fcb {
    B filename[8]; /* ファイル名 */
    B extension[3]; /* 拡張子 */
    W top_adr; /* 先頭アドレス */
    W size; /* ファイルサイズ */
    UW date_tm; /* 日付・時刻 */
    W attribute; /* ファイル属性 */
} DIR_TBL;
```

ファイル名	文字列データの最大8文字＋スペース 前詰めで8文字未満の場合は、NULLコードが格納されます（英文字はすべて大文字です）
拡張子	文字列データの最大3文字＋スペース 前詰めで3文字未満の場合は、NULLコードが格納されます（英文字はすべて大文字です）
先頭アドレス	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
ファイルサイズ	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(バイト単位)
日付／時刻	ファイル書込み時の年月日時分データ D0 ～ D4 = 00～29 秒/2 秒間隔 D5 ～ D10 = 00～59 分 D11 ～ D15 = 00～23 時 D16 ～ D20 = 01～31 日 D21 ～ D24 = 01～12 月 D25 ～ D31 = 00～99/1980～2079 年
ファイル属性	未使用(常に0)

戻り値

関数結果

E_OK	正常終了
E_PRM	ファイル管理テーブルの読み位置指定エラー
E_NG	ファイル格納情報が無い

補足

2.2.12. dat_fsize (DT700 互換モード専用)

ファイル格納領域の未使用領域サイズを取得します。

```
UW dat_fsize(void);
```

パラメータ

なし

戻り値

ファイル格納領域の未使用領域サイズ

補足

2.2.13. dat_fdel (DT700 互換モード専用)

ファイルを削除します。

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

パラメータ

name

指定ファイル名の格納先ポインタ

ファイル名 8文字以下(文字列データ)

拡張子 3文字以下(文字列データ)

例) nnnnnnnn. mmm

ファイル名 拡張子

ANKコード(シフトJISは不可)

". "は省略可能です

空白を指定した時はエラーになります。

戻り値

関数結果

E_OK 正常終了

E_PRM 不当ファイル名

E_NG 指定ファイルなし

補足

2.2.14. dat_mem_size

メモリ領域の未使用領域のサイズを返します。

```
UW dat_mem_size(void);
```

パラメータ

なし

戻り値

メモリの未使用領域サイズ

補足

2.2.15. fil_mkdir (FATファイルモード専用)

新しいディレクトリを作成します。

制限事項: 16ファイル同時オープンされている場合は、異常終了します。

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

パラメータ

path

作成するディレクトリのフルパス名

戻り値

関数結果

E_OK	正常終了
E_NG	異常終了

補足

- 9文字以上11文字以下のディレクトリ名を指定した場合、エラーとはならず以下になります。
(例) A: ¥12345678901 → A: ¥12345678. 901
 - MS-DOS の予約デバイスに相当するファイル名の制限はありません。
 - ディレクトリ名に¥を連続指定してもエラーとはならず、以下のように作成されます。
(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、作成しないで下さい。
サーチ系の関数でサーチできなくなります。
- 上記注意事項は、本システムのファイル系関数全般に該当します。

2.2.16. `fil_rmdir` (FATファイルモード専用)

ディレクトリを削除します。

制限事項:ファイルが16ファイル同時オープンされている場合は、異常終了します。

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

パラメータ

path

削除するディレクトリのフルパス名

戻り値

関数結果

E_OK 正常終了

E_NG 異常終了

補足

2.2.17. fil_remove

ファイルを削除します。

制限事項:ファイルが16ファイル同時オープンされている場合は、異常終了します。

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

パラメータ

pathname

ファイルのパス名 (指定方法は、open 関数参照)

戻り値

関数結果

E_OK	正常終了
E_NG	異常終了
E_PR	パラメータエラー

補足

2.2.18. fil_rename

ファイル名の変更またはファイルの移動を行います。

制限事項: 15ファイル同時オープンされている場合、またはルートディレクトリにファイルが192個存在する場合は、異常終了します。

異なるドライブ間のファイル移動はできません(異常終了します)

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

パラメータ

oldname

現在のファイルのパス名(指定方法は、open 関数参照)

newname

新しいファイルのパス名(指定方法は、open 関数参照)

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

2.2.19. fil_fstat

オープンされているファイルの日時、サイズ、属性を取得します。
DT700互換モードの場合、リターンするファイル属性は常に0です。

```
ER fil_fstat(
    int handle,
    struct stat *buffer
);
```

パラメータ

handle

ファイル番号

buffer

結果格納バッファ

【ストラク構造】

```
typedef struct stat{
    UW    filesize    /* ファイルサイズ */
    UH    date        /* ファイルの日付 */
    UH    time        /* ファイルの時刻 */
    B     attr        /* ファイルの属性 */
                        _A_NORMAL : 読み書き可
                        _A_RDONLY : 読み専用
                        _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
} FIL_FSTAT;
```

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

時刻と日付のフォーマット

	15	1110	5 4	0	
時刻	時(0~23)	分(0~59)	秒(0~29)		秒は2秒単位
	15	9 8	5 4	0	
日付	年(0~99)	月(1~12)	日(1~31)		年は0を1980年とします

2.2.20. fil_chsize

ファイルのサイズを変更します。

元のファイルより大きいサイズが指定された場合は、ファイルの後ろに NULL を付加し、小さいサイズが指定された場合は、先頭から指定サイズまでをファイルサイズとします。

制限事項: 16ファイル同時オープンされている場合は、異常終了します。

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

パラメータ

path

変更対象のファイル名 (全パス名指定)

fsize

変更するファイルのサイズ (バイト単位)

戻り値

関数結果

E_OK 正常終了

E_NG 異常終了

補足

ファイルサイズを一度小さく変更したとき、その部分の内容は保証しません。

オープン中のファイルに対して本関数を実行した場合、その内容は保証しません。

2.2.21. fil_getsize

ファイル領域の空きサイズの取得をします。

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

パラメータ

path
ドライブ名

戻り値

空き領域サイズ
関数結果
E_NG 異常終了
E_PRM パラメータエラー

補足

2.2.22. fil_findfirst

指定された条件でファイルの検索を行い、条件に一致するファイル名を取得します。
 次候補を読み出す時は fil_findnext 関数を使用してください。
 DT700互換モードの場合、第二パラメータのファイル属性は、意味を持ちません。(検索条件対象外)

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

パラメータ

path

検索ファイル名(指定方法は、open 関数参照。形式2のみ有効。ワイルドカード使用可)

attr

検索ファイルの属性(OR指定可)

_A_NORMAL	読み書き可能
_A_VOLID	ボリュームID
_A_RDONLY	読み専用 ※
_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/読み専用属性は、指定の有無に関わらず常に検索対象となります

buffer

結果を格納するバッファ

【ストラク構造】

```
typedef struct find_t{
    B reserved[21] : 予約領域
    B attrib : 検索されたパスについてのファイル属性
    UH wr_time : ファイルを最後に更新した時刻
    UH wr_date : ファイルを最後に更新した日付
    W size : ファイルの大きさ (バイト単位)
    B name[13] : 検索されたファイルもしくはディレクトリの名前
    (パスを含まず文字列の最後は NULL です)
    (英文字は、すべて大文字です)
} FIND_T;
```

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

	15	1110	5 4	0	
時刻	時(0~23)	分(0~59)	秒(0~29)		秒は2秒単位
	15	9 8	5 4	0	
日付	年(0~99)	月(1~12)	日(1~31)		年は0を1980年とします

2.2.23. fil_findnext

fil_findfirst 関数にて、取得されたファイル名の次候補を讀出します。
アーカイブ／読込み専用属性は、指定の有無に関わらず常に検索対象となります。
次候補がない場合は、異常終了とします。

```
ER fil_findnext(  
    struct find_t *buffer  
);
```

パラメータ

buffer

結果を格納するバッファ

【ストラク構造】

```
typedef struct find_t{  
    B reserved[21] : 予約領域  
    B attrib       : 検索されたパスについてのファイル属性  
    UH wr_time    : ファイルを最後に更新した時刻  
    UH wr_date    : ファイルを最後に更新した日付  
    W size        : ファイルの大きさ (バイト単位)  
    B name[13]    : 検索されたファイルもしくはディレクトリの名前  
                  (パスを含まず文字列の最後は NULL です)  
                  (英文字は、すべて大文字です)  
} FIND_T;
```

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

2.2.24. fil_filesize

指定されたファイルの個数と総サイズを取得します。
 また、指定によりサブディレクトリ下の検索も行うことができます。
 指定ファイルが存在しない場合、正常終了(ファイルサイズ/個数=0)します。

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

パラメータ

path

検索ファイル名(指定方法は、open 関数参照。形式2のみ有効。ワイルドカード使用可)

buffer

結果を格納するバッファ

【ストラク構造】

```
typedef struct cnt_and_size{
    UW cnt      : 該当するファイルの個数
    UW size     : ファイルの総サイズ
} FIL_SIZE;
```

find_sw

サブディレクトリ下の検索指定

FIL_SUBDIR_ON	サブディレクトリ下まで検索する
FIL_SUBDIR_OFF	サブディレクトリ下は検索しない

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 (APからCドライブを指定した場合)

補足

【異常終了要因】

- ①ファイル/パス名異常(使用不可コード混在)
- ②パス長異常(128文字以上の指定)
- ③指定ドライブ未フォーマット
- ④指定ドライブ未搭載
- ⑤ドライブ指定外(D以上)

2.2.25. fil_filefind

ファイルの検索を行ないます。

検索結果のファイル名は、パスを含んだ形式で取得されます。

パスの異なる同一名称のファイルが複数存在する場合、検索条件に順次合致していくなかで指定した番目に一致した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

検索するファイル名 (指定方法は、open 関数参照。形式2のみ有効。ワイルドカード使用可)

buffer

ファイル名を格納するバッファ (形式2で返却)

find_sw

サブディレクトリ下の検索指定

FIL_SUBDIR_ON	サブディレクトリ下まで検索する
FIL_SUBDIR_OFF	サブディレクトリ下は検索しない

seq_no

指定番目の番号

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 (APからのCドライブを指定した場合)

補足

【異常終了要因】

- ①ファイル/パス名異常 (使用不可コード混在)
- ②パス長異常 (128文字以上の指定)
- ③指定ドライブ未フォーマット
- ④指定ドライブ未搭載
- ⑤ドライブ指定外 (D以上)
- ⑥ファイル未検出
- ⑦指定ファイルが無い

2.2.26. dat_fname (DT700 互換モード専用)

ファイル名の変更を行ないます。

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

パラメータ

old

変更前のファイル名

例) nnnnnnnn.mmm

ファイル名 拡張子

ANKコード(シフトJIS不可)

”.”は省略可能

new

変更後のファイル名

戻り値

関数結果

E_OK 正常終了

E_NG 異常終了

補足

【ファイル名に使用可能な文字コード】

'0'～'9', 'A'～'Z', 'a'～'z', カタカナ (0xA0～0xDF), スペース, '!', '#', '\$', '%', '&', ''', '(', ')', '-', '^', '_, '@', '{', '}', '_'

注) ファイル名の大文字／小文字は同一視していません。

2.2.27. dat_F_Search (DT700 互換モード専用)

指定された検索指示情報に従いファイルデータを検索します。
OPENされていないファイルに行なうと、異常終了します。(E_NG を関数結果として返します)

```
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

検索要求ファイル名格納先アドレス

start_adr

検索開始相対アドレス

fieldsize

1検索データのデータサイズ

keypos

検索コードの格納先相対アドレス

keylen

検索コードのデータサイズ

code

比較検索コードの格納先アドレス

sdata

検索データの格納先バッファアドレス(出力情報)

fpos

検索データアドレス格納ポインタ(出力情報)

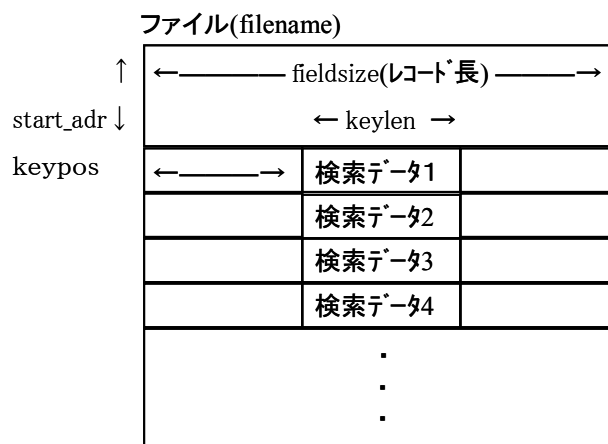
戻り値

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
E_NON	検索データなし

補足

パラメータ補足説明



検索データ(code)

<例: 検索データ3で検索できた場合>

*sdata = 検索データ3

*fpos = 検索データ3のアドレス

3. 表示部

3.1. 表示制御

3.1.1. 表示画面

128×64ドットの画面サイズで、各フォントモードの縮小ANKを基準に、左上を(0,0)のキャラクタ座標系で構成します。

グラフィック座標は、フォントモードに関係なく、左上(0,0)、右下(127,63)になります。

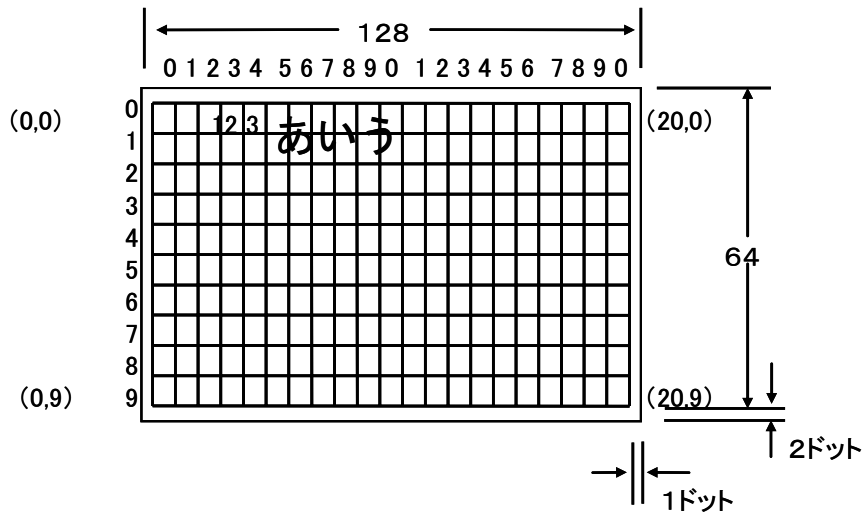
フォントの表示は、キャラクタ座標で行います。

各フォントモードで余ったドット分は、中央寄せで表示します。

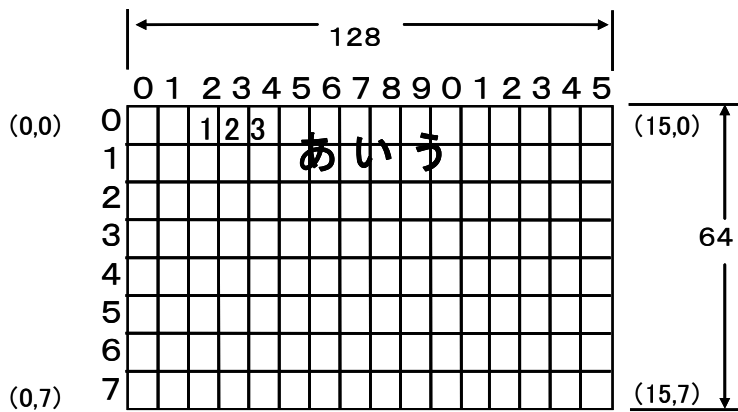
表3.1 表示桁数

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)	21桁×10行	210文字
	標準ANK	(6×12)	21桁× 5行	105文字
	漢字	(12×12)	10桁× 5行	50文字
8ドット	縮小ANK	(8× 8)	16桁× 8行	128文字
	標準ANK	(8×16)	16桁× 4行	64文字
	漢字	(16×16)	8桁× 4行	32文字
10ドット	縮小ANK	(10×10)	12桁× 6行	72文字
	標準ANK	(10×20)	12桁× 3行	36文字
	漢字	(20×20)	6桁× 3行	18文字

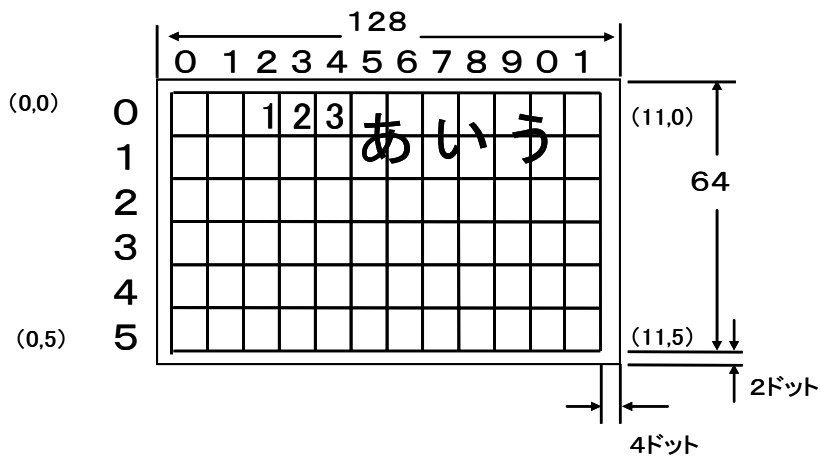
(1) 6ドットモード時



(2) 8ドットモード時



(3) 10ドットモード時



3.1.2. 表示コード

本機は、シフトJISコードを使用します。

コード体系には、制御コードと文字コードがあり、文字コードはさらに ANK と漢字コードに分類されます。

また、漢字コードの一部に外字フォントを登録することができます。

各フォントのビットマップの先頭アドレス等は、フォントテーブルにより管理されており、先頭アドレスを変更することによりユーザフォントを表示させることができます。

(1) ANKコード

表示可能コードは表の網掛け部分(01H~80H、A0H~DFH、FDH~FFH)です。

表3. 2 ANK(半角文字)コード表

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	Null	DEL		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			ア	キ	ヌ	ラ		
8	BS		(8	H	X	h	x			イ	ク	ネ	リ		
9)	9	I	Y	I	y			ウ	ケ	ノ	ル		
A	LF		*	:	J	Z	j	z			エ	コ	ハ	レ		
B		ESC	+	;	K	[k	{			オ	サ	ヒ	ロ		
C		←	,	<	L	¥	l				ヤ	シ	フ	リ		
D	CR	→	-	=	M]	m	}			ユ	ス	ハ	ン		
E			.	>	N	^	n	~			ヨ	セ	ホ	°		
F			/	?	O	_	o				ッ	ソ	マ	°		

網掛け部分が文字として表示されます。

※ ‘¥’コードの表示

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、システムデータ管理提供の関数で変更します。

バックスラッシュのフォントデータ(ビットマップ)は存在しないため、表示関数で独自に持っているデータを使用します。

(2) 漢字／外字コード

1文字／文字列表示を行う場合の表示可能な漢字／外字コード(2バイトコード)は、以下のコードです。

第1水準 : 8140H~84FCH
 : 889FH~989EH
 第2水準 : 989FH~9FFCH
 : E040H~EAFCH
 外字 エリア : EB40H~EBC0H

※ 2バイト目が、7F のコード(例:0xEB7F)は存在しません。

表3.3 コード表(2バイトコード)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00																
10																
20																
30																
40																
50																
60																
70					40			7E	80	9F						FC
80				81	第				1	水	準					
				84	第				1 水 準							
			88													
90					第				2 水 準							
			98													
			9F													
A0																
B0																
C0																
D0					40			7E	80				C0			FC
E0				E0	第				2	水	準					
				EA	第				2 水 準							
			EB													
F0																

外字

(3) 実際の表示文字

1文字表示 (lcd_char)、文字列表示 (lcd_string、lcd_string2) 時に、指定するコードにより実際に表示される文字を以下に示します。

表3. 4 1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	(3.2.3 制御コード表示参照)
	01~09, 0B, 0C 0E~1F, 81~9F E0~FC	ANK スペース	ユーザー登録の文字
	20~7F, A0~DF FD~FF	ANK コード表の文字	ユーザー登録の文字
81~84	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	

表3. 5 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	(3.2.3 制御コード表示参照)
1B		ESC制御	(3.2.4 ESC文字列の表示参照)
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

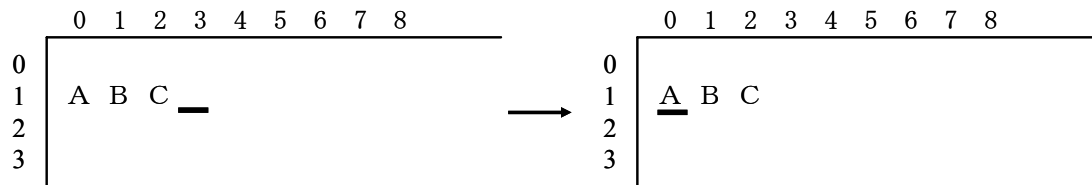
※ ユーザー文字列表示 (lcd_userstr) の場合は、漢字の表示はありません。全て1バイト目を ANK コードとしてみます。81-84、88-9F、E0-EB のコードは ROM フォント時は、ANK スペースを、ユーザーフォント時は、ユーザー登録文字をそれぞれ表示します。

3.1.3. 制御コード表示

1文字表示／文字列表示を行う際、制御コード(0x0a, 0x0d)と認識した場合、以下の表示動作を行います。

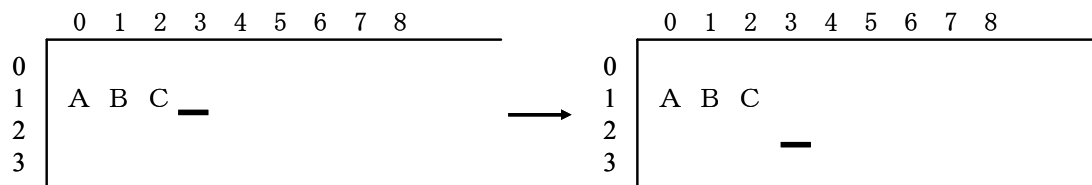
(1) CR(0x0d)の表示

- キャリッジリターン動作を行います

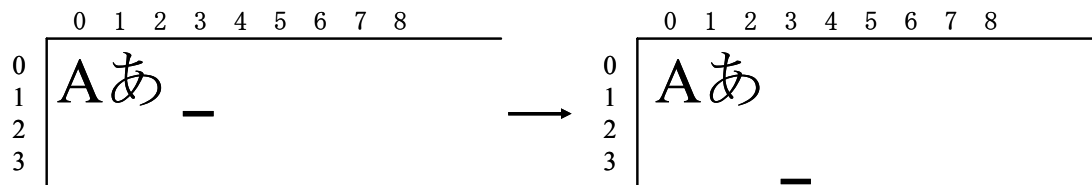


(2) LF(0x0a)の表示

- 縮小ANK文字列中にLFコードが含まれている場合は、1行改行します。

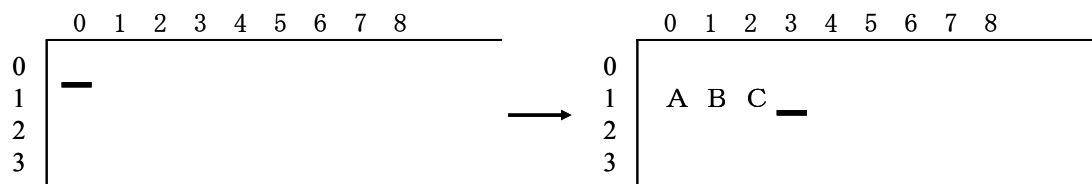


- 標準ANKおよび漢字の文字列中にLFコードが含まれている場合は、2行改行します。

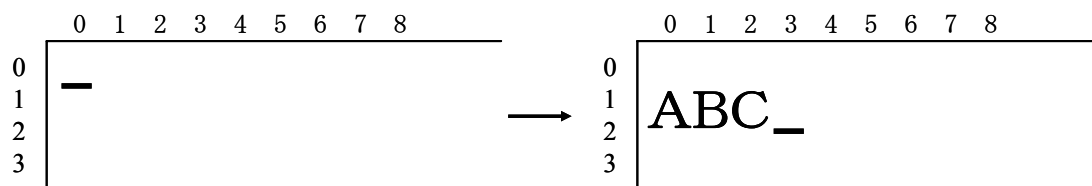


- 文字列表示の先頭がLFコードまたは、1文字表示でLFコードが指定された場合は、入力パラメータのANKモードが縮小の場合は1行改行、標準の場合は2行改行します。

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



3.1.4. ESC文字列の表示

本機は、以下のESCシーケンスをサポートします。

(1)画面クリア

全表示データをスペースクリアします。また、カレントカーソル位置はホームポジション(0行, 0桁)へ移動します。

- 該当指示文字列 : “ ESC [2J ”

(2)カーソル位置設定

指定される座標でカレントカーソル位置を設定します。

- 該当指示文字列 : “ ESC [Pn;PmH ”
“ ESC [Pn;Pmf ”



座標範囲は各フォントモードの縮小ANKを基準にします。

左上端を1行/1桁として指定して下さい。

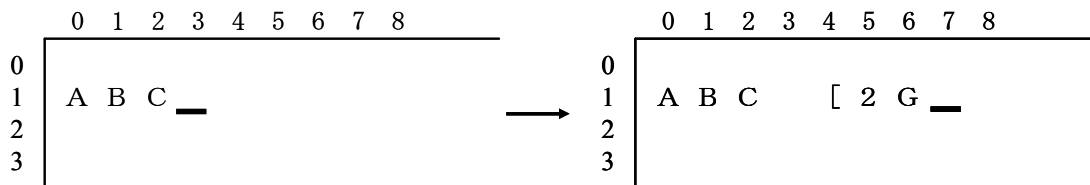
また、範囲外を指定した場合は一番近い行/桁にカーソル位置を設定します。

Pn/Pmは省略可能です。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

フォーマットエラー時は無視して表示します。

例) “ ESC [2J ” が “ ESC [2G ”だった場合



3.1.5. 例外表示制御

1文字／文字列表示を行う際、表示位置によって、既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をする場合と、表示しない場合があります。

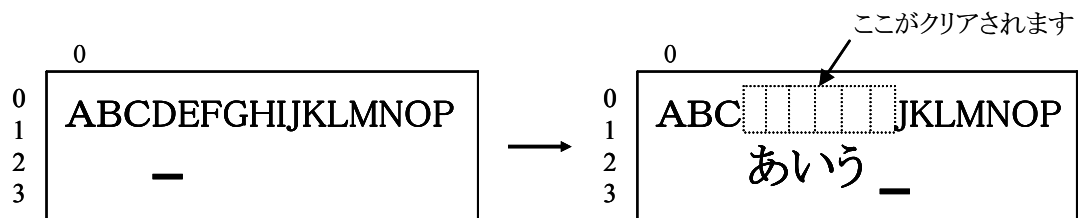
表3.6 例外動作

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	文字列表示を行う時、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

(1) 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。

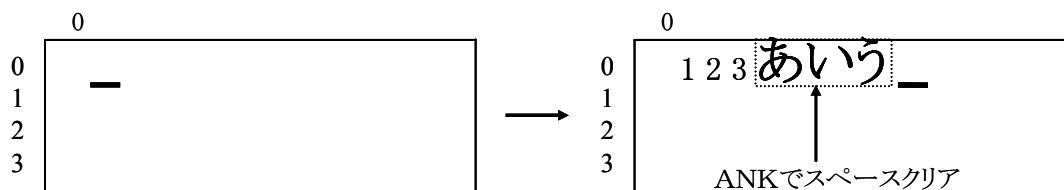
(例) 座標(3, 2)に“あいう”を表示



(2) 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

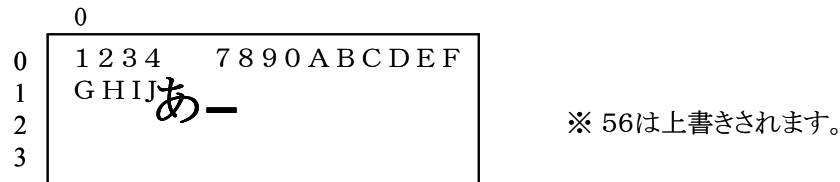


(3) 行端での自動改行制御

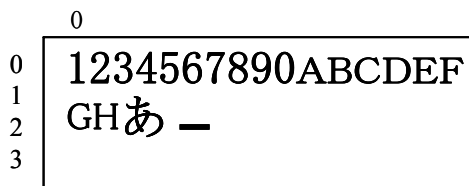
文字列表示を行う時、行端で表示仕切れない場合には先頭文字により、1または2行の改行を自動で行います。

(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



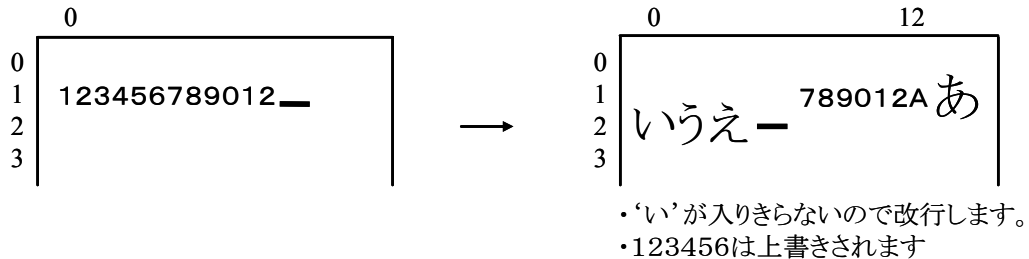
(例2) 先頭文字が標準ANK/漢字の場合、座標(0, 0)に“1234567890ABCDEFあ”表示



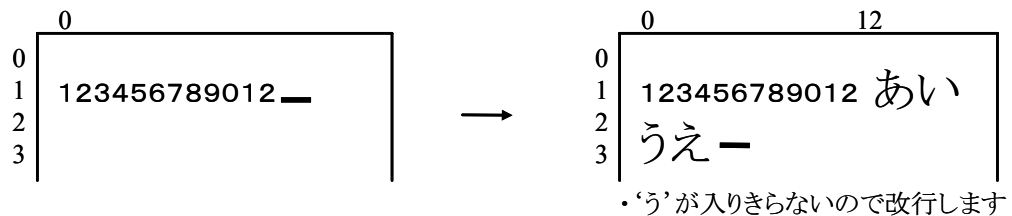
(4) 行端での漢字表示制御

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います。(ただし、改行モードありの時)

(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



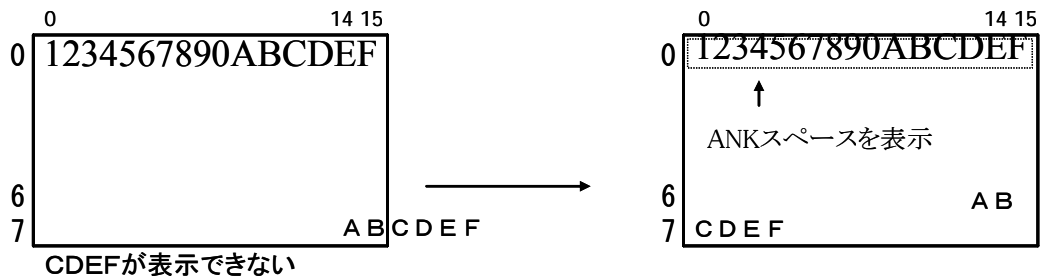
(例2) 先頭文字が標準ANK/漢字の場合、座標(12, 1)に“あいうえ”を表示



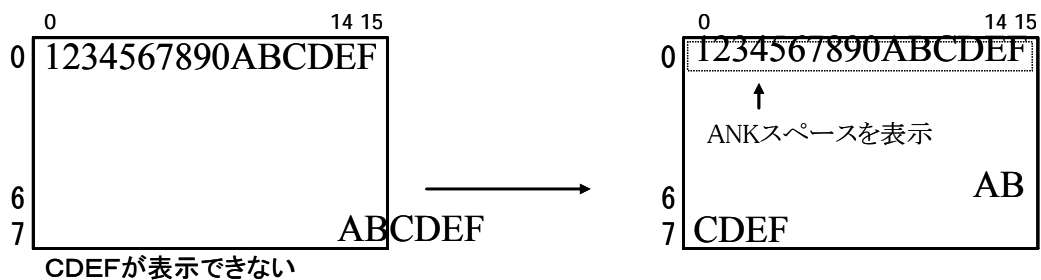
3.1.6. スクロール制御

カレントカーソル位置が最下行で文字列が表示できない場合はスクロール制御を行います。

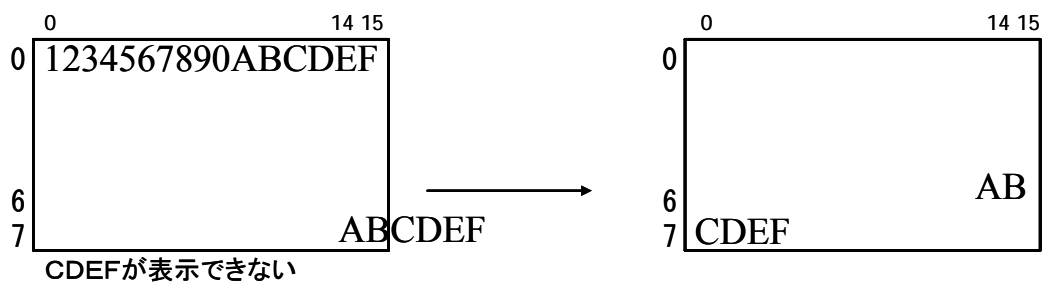
(例1) 最下行の縮小ANK表示でスクロールが発生する場合
座標(14, 7)に縮小ANK“ABCDEF”を表示



(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



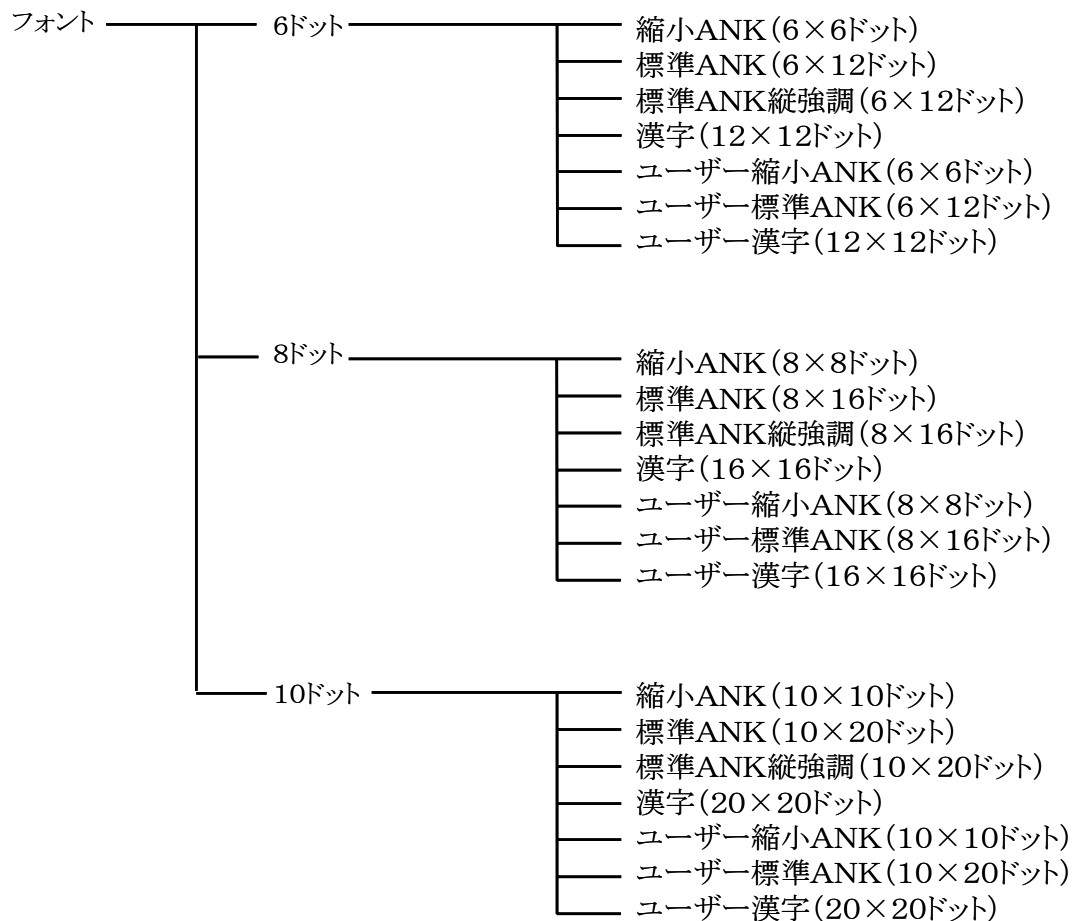
※スクロール抑制

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ときで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いません。表示しきれなかった文字については無視されます。

3.2. フォント制御

1文字／文字列表示を行うとき、6ドット／8ドット／10ドットフォントの取り扱いは、予め設定されたフォントモードで処理します。
フォントモードの混在はしません。

3.2.1. フォントの種類

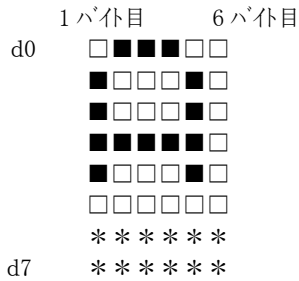


- ※ ユーザーフォントとはユーザーが独自に作成するフォントファイルを示します。また、ユーザーフォント以外はROMフォントです。ROMフォントもユーザーフォントもデータ構成は同じです。「3. 2. 2フォントデータ構成」を参照して下さい。
- ※ ユーザーフォントの表示を行う場合は、「ユーザーフォントファイルの登録」を、ROMフォントへ戻す場合は、「ROMフォント設定」を行って下さい。これによりユーザー／ROMフォントの混在表示が可能です。
- ※ 縦強調フォントを表示する場合は、予め設定を行う必要があります。「データ管理基本仕様書参照」 ユーザーフォント表示の場合は、縦強調指定は無効になります。

3.2.2. フォントデータ構成

(1) 6ドットモードのフォント

①縮小ANKデータ構成(6×6ドット)

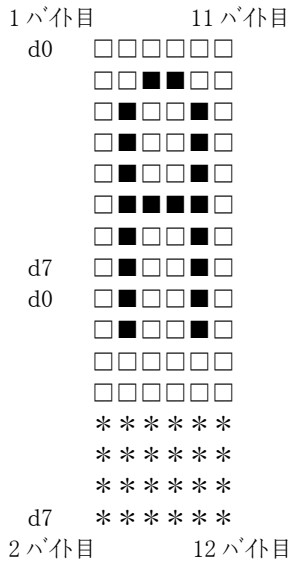


1フォント6バイト構造

1E	09	09	09	1E	00
1	2	3	4	5	6

(バイト目)

②標準ANKデータ構成(6×12ドット)



1フォント12バイト構造

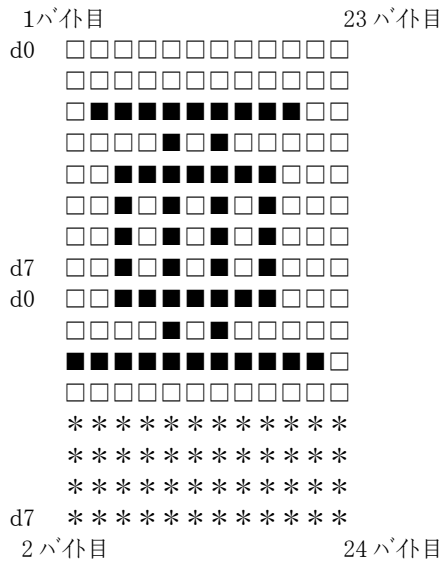
00	00	FC	03	22	00
1	2	3	4	5	6

(バイト目)

22	00	FC	03	00	00
7	8	9	10	11	12

(バイト目)

③標準漢字データ構成(12×12ドット)



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

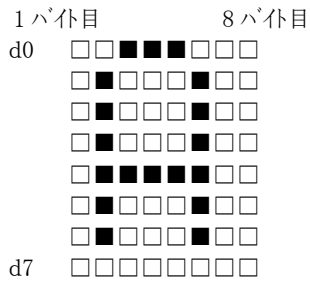
(バイト目)

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

(バイト目)

(2) 8ドットモードのフォント

① 縮小ANKデータ構成 (8×8ドット)

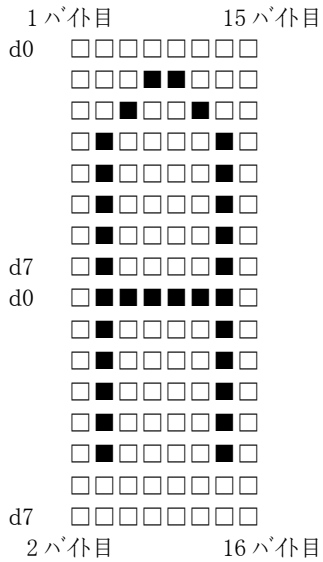


1フォント8バイト構造

00	7E	11	11	11	7E	00	00
1	2	3	4	5	6	7	8

(バイト目)

② 標準ANKデータ構成 (8×16ドット)



1フォント16バイト構造

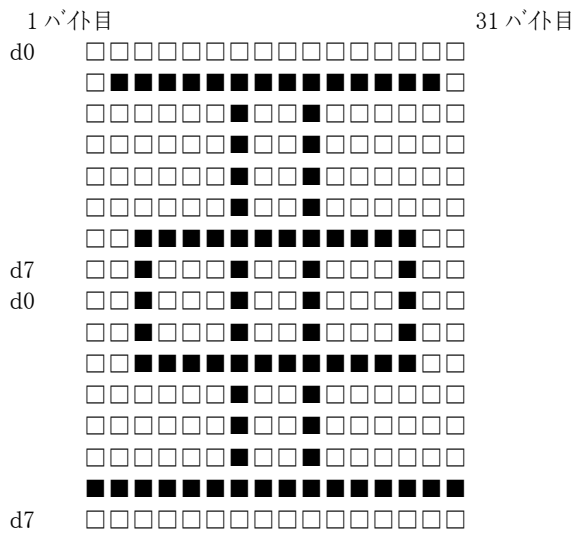
00	00	F8	3F	04	01	02	01
1	2	3	4	5	6	7	8

(バイト目)

02	01	04	01	F8	3F	00	00
9	10	11	12	13	14	15	16

(バイト目)

③標準漢字データ構成(16×16ドット)



1フォント32バイト構造

00	40	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

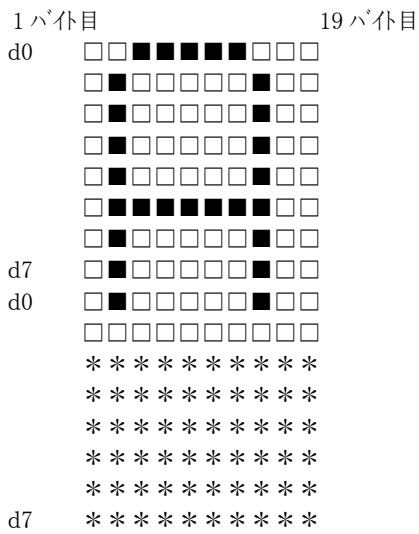
(バイト目)

42	44	FE	7F	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

(バイト目)

(3) 10ドットモードのフォント

①縮小ANKデータ構成(10×10ドット)



2バイト目 20バイト目

1フォント20バイト構造

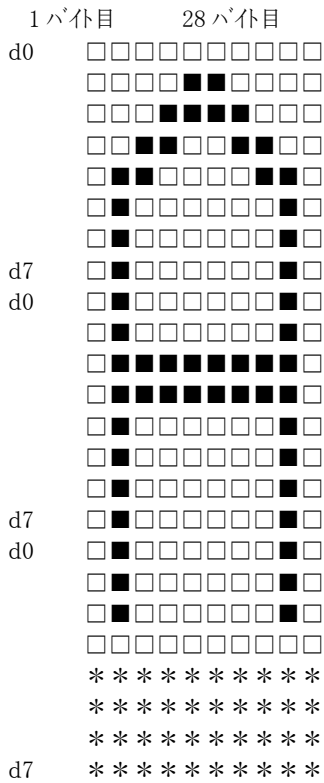
00	00	FE	01	21	00	21	00	21	00
1	2	3	4	5	6	7	8	9	10

(バイト目)

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

(バイト目)

②標準ANKデータ構成(10×20ドット)

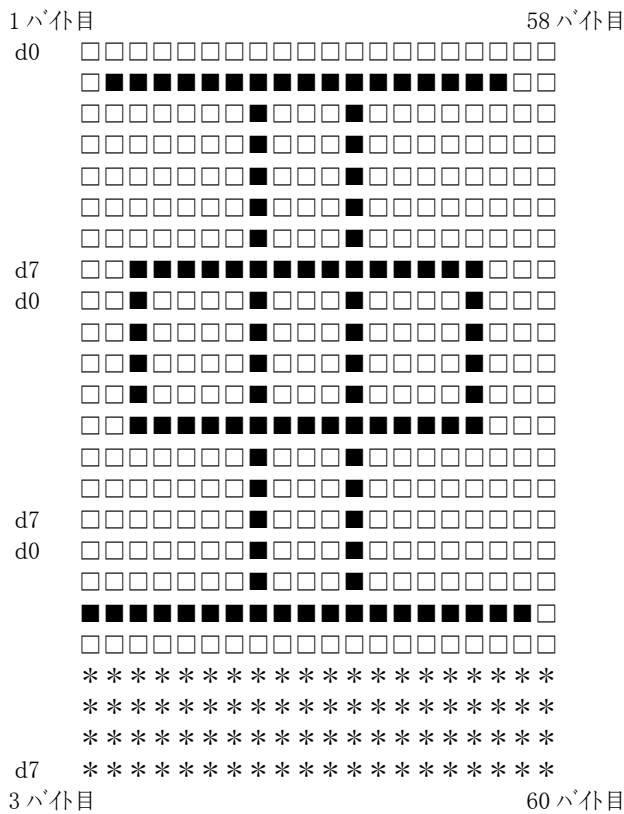


3バイト目 30バイト目

1フォント30バイト構造

00	00	00	F0	FF	07	18	0C	00	0C	(バイト目)
1	2	3	4	5	6	7	8	9	10	
0C	00	06	0C	00	06	0C	00	0C	0C	(バイト目)
11	12	13	14	15	16	17	18	19	20	
00	18	0C	00	F0	FF	07	00	00	00	(バイト目)
21	22	23	24	25	26	27	28	29	30	

③標準漢字データ構成(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

(バイト目)

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

(バイト目)

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

(バイト目)

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

(バイト目)

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

(バイト目)

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

(バイト目)

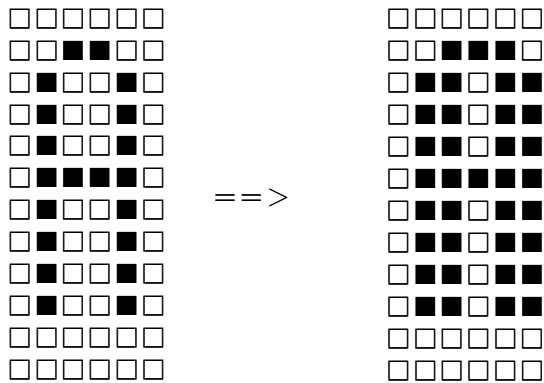
3.2.3. 修飾文字のフォントデータ制御

該当フォントデータから修飾文字の表示をするため、フォントデータを制御します。

- 強調 右へ1ドットずらしORします。
- 反転 ドットを反転します。
- 横倍角 横方向へ2倍にします。

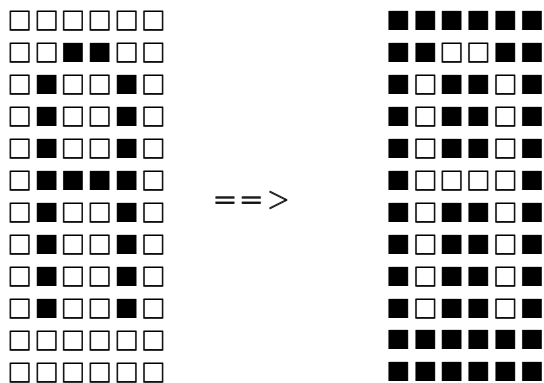
(1) 強調表示

・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。



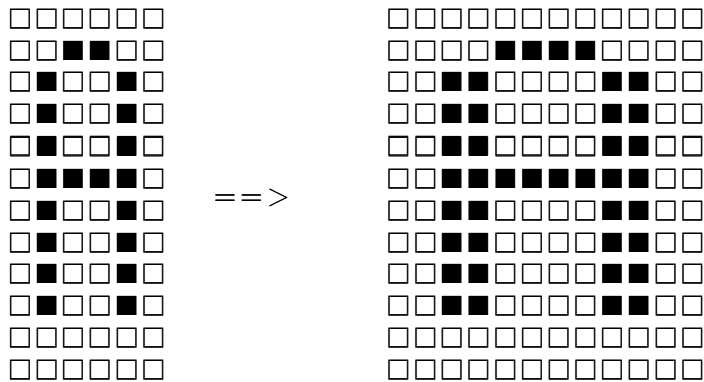
(2) 反転表示

・フォントデータのビットパターンを反転します。



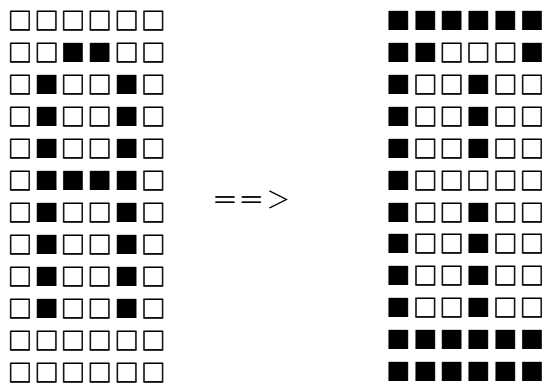
(3) 横倍角表示

- ・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。



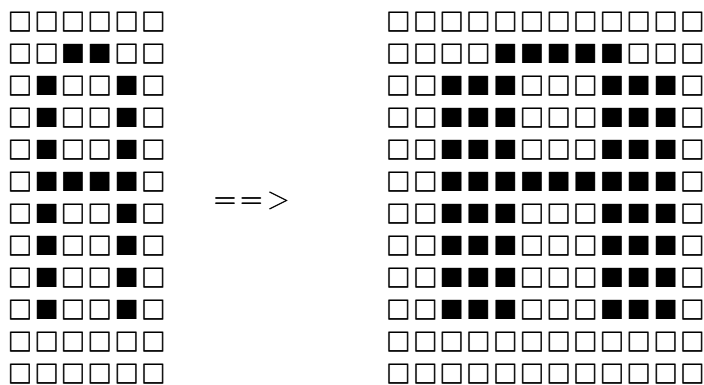
(4) 強調／反転表示

- ・(1)、(2)両方の処理を合わせます。(強調後反転する)



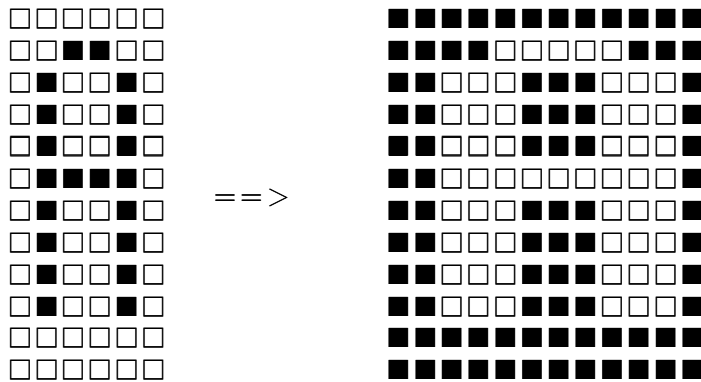
(5) 横倍角／強調表示

- ・(1)、(3)両方の処理を合わせます。(横方向へ2倍してから強調する)

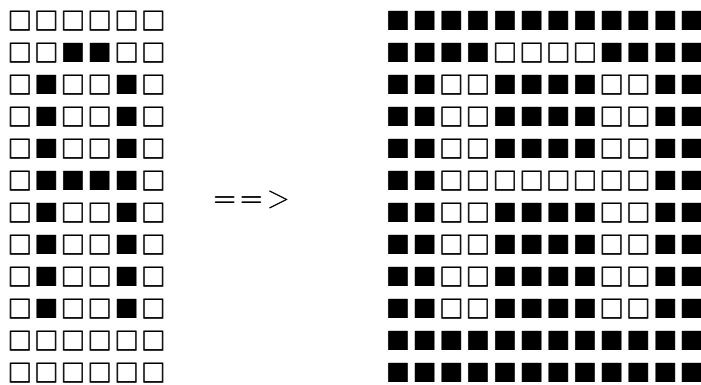


(6) 横倍角／強調／反転表示

- ・(1)、(2)、(3)の処理を合わせます。(横方向へ2倍してから強調後反転する)

**(7) 横倍角／反転表示**

- ・(2)、(3)の処理を合わせます。(横方向へ2倍してから反転する)



3.2.4. ユーザーフォントファイル

ユーザーが独自に作成したフォントを表示させることができます。フォントファイルは大きく分けて2種類あります。

ROM フォントを使用せずユーザー独自のフォントを表示させるためのユーザーフォントファイルおよび、特定のコード(0xEB40～0xEBC0 ただし、0xEB7F は除く)で表示できる外字フォントファイルです。

登録する FONT は、DT-900 と DT-930 で方向が違うので、DT-900 で使用している FONT ファイルを DT-930 に適用する場合は、予めツールで変換をしてから使用して下さい。

(1) フォントファイルの種類

フォントファイル種別	フォント種別	容量
外字フォントファイル	6ドットフォント	24 バイト× 128 文字 = 3,072 バイト
	8ドットフォント	32 バイト× 128 文字 = 4,096 バイト
	10ドットフォント	60 バイト× 128 文字 = 7,680 バイト
ユーザーフォントファイル	6ドット縮小 ANK フォント	6 バイト× 256 文字 = 1,536 バイト
	6ドット標準 ANK フォント	12 バイト× 256 文字 = 3,072 バイト
	6ドット漢字フォント	24 バイト×7,393 文字 = 177,432 バイト
	8ドット縮小 ANK フォント	8 バイト× 256 文字 = 2,048 バイト
	8ドット標準 ANK フォント	16 バイト× 256 文字 = 4,096 バイト
	8ドット漢字フォント	32 バイト×7,393 文字 = 236,576 バイト
	10ドット縮小 ANK フォント	20 バイト× 256 文字 = 5,120 バイト
	10ドット標準 ANK フォント	40 バイト× 256 文字 = 10,240 バイト
	10ドット漢字フォント	60 バイト×7,393 文字 = 443,580 バイト

(2) フォントデータ構成

フォントデータの構成は ROM フォントと同一です。「3.3.2 フォントデータ構成」を参照して下さい。

(3) ファイル構成

- 外字フォントファイル構成

ファイルヘッダ等はありません。
右図の様に続けてフォントイメージを作成して下さい。

ファイル先頭	
EB40hのフォント	
EB41hのフォント	
:	
:	
:	
EB7Ehのフォント	
EB80hのフォント	
:	
:	
EBC0hのフォント	
ファイルEND	

EB7Fhはありません。
詰めて作成して下さい。

- ANKフォントファイル構成

ファイルヘッダ等はありません。
右図の様に続けてフォントイメージを
作成して下さい。

ファイルTOP
00hhのフォント
01hのフォント
⋮
⋮
⋮
⋮
FFhのフォント
ファイルEND

- 漢字フォントファイル構成

ファイルヘッダ等はありません。
右図の様に続けてフォントイメージを
作成して下さい。

ファイルTOP
8140hのフォント
⋮
⋮
84FFhのフォント
889Fhのフォント
⋮
⋮
9FFFh
E040h
EAFFhのフォント
ファイルEND

XX00h~XX3Fhおよび8840h~889Eh
のフォントイメージは入れません。
詰めて作成して下さい。

XX7Fh, XXFDh, XXFEh, XXFFhは指定
しても表示されませんがダミーデータを入れて
おいて下さい。

※ 途中までしかデータが入っていない場合、それ以降のコードが指定された時は、スペースを表示します。

(4) 表示方法

外字フォントは外字切り替え (lcd_gaiji) を呼んでファイルを登録して下さい。登録後、1文字表示 / 文字列表示で 0xEB40~0xEBC0 のコードを指定すると表示されます。

ANKフォントおよび漢字フォントのユーザーフォントは、ユーザーフォントファイル登録 (lcd_usrfont) を呼んでファイルを登録して下さい。登録後、1文字表示 / 文字列表示で表示されます。

※ 処理高速化のため、外字フォントおよびANKフォントは登録時にメモリに展開します。ファイルを更新した場合は、登録し直して下さい。
(漢字フォントは1文字表示 / 文字列表示で漢字表示時にファイルアクセスします。)

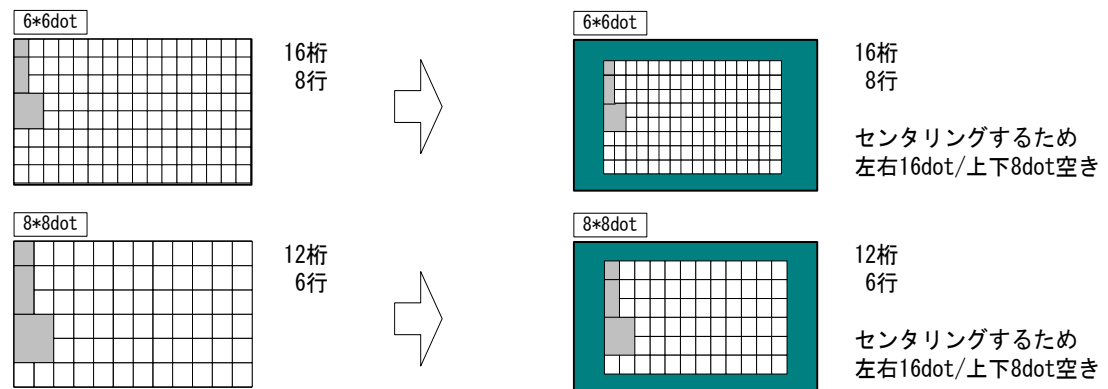
3.3. DT700互換表示モード

本表示関数では、DT700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)

3.3.1. 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)。



3.3.2. 互換モードB

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

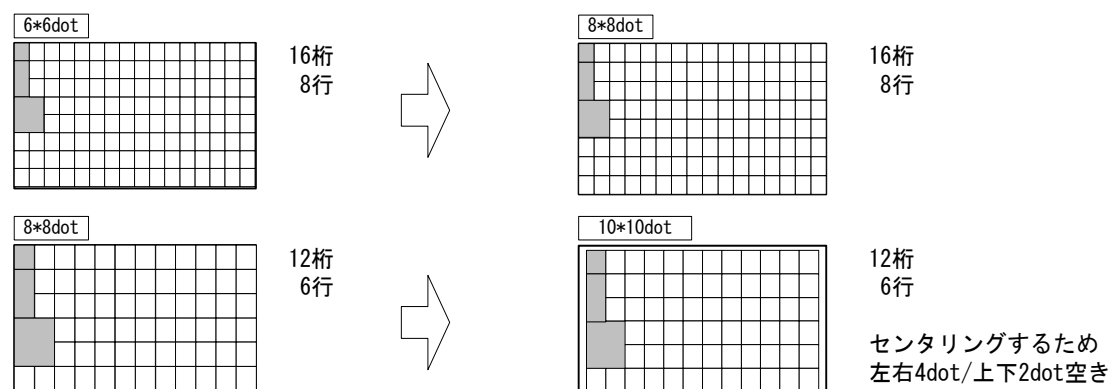


表3.7 モード時の表示桁数

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。

3.4. ファンクション詳細

ファンクション詳細を次ページより示します。

3.4.1. lcd_cls

全表示データをスペースクリアします。
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。

```
ER lcd_cls(void);
```

パラメータ

なし

戻り値

関数結果

E_OK

正常終了

補足

3.4.2. 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。

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。

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

パラメータ

csr_type

カーソル表示タイプ

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

関数結果

E_OK 正常終了

E_PRM パラメータエラー

補足

3.4.3. lcd_csr_put

指定される行・桁でカーソル位置を設定します。

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／桁が最大値を越える場合は、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。

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

パラメータ

csr_line

カーソル行位置

6ドットモード時0～9行を指定

8ドットモード時0～7行を指定

10ドットモード時0～5行を指定

csr_colum

カーソル桁位置

6ドットモード時0～20桁を指定

8ドットモード時0～15桁を指定

10ドットモード時0～11桁を指定

戻り値

関数結果

E_OK

正常終了

補足

3.4.4. lcd_csr_get

カレントカーソル位置およびカーソル表示タイプを返します。

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

パラメータ

csr_line

カーソル行位置のデータポインタ
6ドットモード時は0～9を格納
8ドットモード時は0～7を格納
10ドットモード時は0～5を格納

csr_colum

カーソル桁位置のデータポインタ
6ドットモード時は0～20を格納
8ドットモード時は0～15を格納
10ドットモード時は0～11を格納

csr_type

カーソル表示タイプのデータポインタ
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

関数結果
E_OK 正常終了

補足

3.4.5. lcd_char

カレントカーソル位置に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
 縮小ANKモード LCD_ANK_LIGHT
 標準ANKモード LCD_ANK_STANDARD

disp_attr

表示属性
 通常表示 LCD_ATTR_NORMAL
 反転表示 LCD_ATTR_REVERS
 強調表示 LCD_ATTR_WIDTH
 横倍表示 LCD_ATTR_DOUBLE
 複数の修飾を行う場合はOR指定して下さい。

disp_data

表示データ

lf_mode

改行モード
 改行なし LCD_LF_OFF
 改行あり LCD_LF_ON

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

- ① 00h コード
 NULL(0x00)コードは、終了コードとして扱います。
- ② 0Dh, 0Ah コード
 CR(0x0D), LF(0x0A)の表示動作が可能です。
 ※文字の行端表示で改行ありモードの場合は自動改行して、改行なしモードの場合は改行しません。

3.4.6. lcd_string

カレントカーソル位置から文字列を表示します。

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。

文字列の有効バイト数は1024バイトです。

従ってANKは1024／漢字は512文字が最大表示可能文字数で、以降は無視します。

```
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

縮小ANKモード	LCD_ANK_LIGHT
標準ANKモード	LCD_ANK_STANDARD

disp_attr

表示属性

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

複数の修飾をしたい場合はORで設定して下さい。

disp_data

表示データバッファポインタ

lf_mode

改行モード

改行なし	LCD_LF_OFF
改行あり	LCD_LF_ON

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

- ① 00h コード
NULL (0x00)コードは、終了コードとして扱います。
- ② 0Dh, 0Ah コード
CR (0x0D), LF (0x0A) の表示動作が可能です。

文字の行端表示で改行ありモードの場合は自動改行して、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。)

3.4.7. lcd_string2

通常の文字列表示と同等の処理をしますが、改行ありモード時、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いません。また、最下行にある場合はスクロールを行いません。

```
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	
縮小ANKモード	LCD_ANK_LIGHT
標準ANKモード	LCD_ANK_STANDARD

disp_attr

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

※複数の修飾をしたい場合はORで設定して下さい。

disp_data

表示データバッファポインタ

lf_mode

改行モード	
改行なし	LCD_LF_OFF
改行あり	LCD_LF_ON

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

- ① 制御コード表示(0x01~0x1f)
lcd_stringと同様の表示動作をします。
- ② ESC文字列表示
lcd_stringと同様に複数のESC文字列を対応します。

※ lcd_stringとの相違点

改行ありモードで最下行右端になった場合はスクロールをしないで改行なしモードに切り替わります。

3.4.8. lcd_userstr

カレントカーソル位置から文字列を表示します。

ANKを表示することができます。

(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。

文字列の有効バイト数は 1024 バイトで、以降は無視します。

```
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

縮小ANKモード	LCD_ANK_LIGHT
標準ANKモード	LCD_ANK_STANDARD

disp_attr

表示属性(OR 指定可)

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

disp_data

表示データバッファポインタ

lf_mode

改行モード

改行なし	LCD_LF_OFF
改行あり	LCD_LF_ON

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

① 00h コード

NULL(0x00)コードは、終了コードとして扱います。

② 0Dh, 0Ah コード

CR(0x0D), LF(0x0A)の表示動作が可能です。

※ 漢字の表示はできません。全てのコードをANKとして扱います。

3.4.9. lcd_line

直線を描画します。

画面ドットイメージ(横128、縦64ドット)の開始座標と終了座標で描画します。

引数のドットモードがオン(黒)の場合は表示、オフ(白)の場合は削除します。

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

パラメータ

dot_mode

直線表示モード

直線削除	LCD_LINE_OFF
直線描画	LCD_LINE_ON

start_x

開始X座標

横ドット位置の0～127を指定

start_y

開始Y座標

縦ドット位置の0～63を指定

end_x

終了X座標

横ドット位置の0～127を指定

end_y

終了Y座標

縦ドット位置の0～63を指定

戻り値

関数結果

E_OK	正常終了
------	------

補足

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、線描画できるところまで線を引きます。

3.4.10. 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

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

filename

外字登録ファイル名称

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

本関数がコールされた時点でファイルよりメモリーへ外字フォントデータを展開します。

ファイルを更新した場合は登録し直して下さい。

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。

3.4.11. 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

漢字6ドットフォント	LCD_K6_FILE
漢字8ドットフォント	LCD_K8_FILE
標準ANK6ドットフォント	LCD_AS6_FILE
標準ANK8ドットフォント	LCD_AS8_FILE
縮小ANK6ドットフォント	LCD_AL6_FILE
縮小ANK8ドットフォント	LCD_AL8_FILE
漢字 10ドットフォント	LCD_K10_FILE
標準ANK10ドットフォント	LCD_AS10_FILE
縮小ANK10ドットフォント	LCD_AL10_FILE

filename

ユーザーフォントファイル名称

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

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で当該フォントデータは、ROMフォントデータを表示します。

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画面切り替え(フォントモードの設定)により登録ファイルのフォントが表示可能となります。

登録ファイルが無効にする場合は、ROMフォント設定(lcd_romfont)を実行して下さい。

本関数内でファイルのオープンチェックをします。オープンエラー時にはパラメータエラーを返します。

また、ANKフォントの場合は、メモリーにデータを読み込みます。

3.4.12. lcd_romfont

ユーザーフォントデータ表示からROMフォントデータ表示へ切り替えます。
(ユーザーフォントとROMフォントの混在表示が可能です。)

```
ER lcd_romfont(void);
```

パラメータ

なし

戻り値

関数結果

E_OK

正常終了

補足

3.4.13. lcd_led

読み取りLEDの点灯／消灯を行います。

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

パラメータ

led_mode

LEDモード

LCD_LED_OFF	LED 消灯
LCD_LED_ON	LED 点灯

led_kind

LED点灯種別

LCD_LED_GREEN	緑点灯
LCD_LED_RED	赤点灯
LCD_LED_BLUE	青点灯

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

LEDモードがLED点灯の場合、既に点灯している時はLEDを消灯して点灯します。
LEDモードがLED消灯の場合には、点灯種別は有効ではありません。(点灯中のLEDに対して消灯します)

3.4.14. lcd_el

ELバックライトの点灯／消灯を行います。

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

パラメータ

el_mode

ELモード

LCD_EL_OFF	EL 消灯
LCD_EL_ON	EL 点灯
LCD_EL_HIGH	EL 点灯 (高輝度)

戻り値

関数結果

E_OK	正常終了
E_PRM	パラメータエラー

補足

4. キー部

4.1. 機能

4.1.1. キーモード

本機のキーモードは、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 キーの押下により入力モードの切り替えを行います。

文字入力モード時は、ハードアイコンで **S** シンボルが表示されます。

(1) 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せん。

(2) 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。

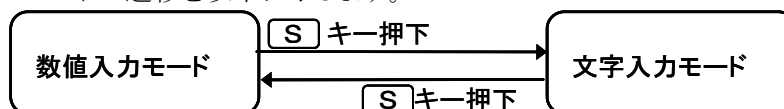
英記号は、めくり入力になっています。

例) ABC

7 …… キーを押すたびに「A → B → C → 7」の順に候補が表示されます。
入力確定は、ENTキー、または、他のキーの入力で確定されます。
ただし、内部処理コードの場合は除きます。

(3) キーモードの遷移

キーモードの遷移を以下に示します。



4.1.2. 1文字入力

アプリケーション指定位置に1文字入力を行います。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ちます。

アスキーコード確定後、アプリケーション指定位置にエコーバック (指定による) を行い戻り値として返却します。

4.1.3. 文字列入力

アプリケーション指定位置から右に指定文字数分を入力領域とし、文字列入力を行います。アプリケーションが指定したバッファにアスキーコードを格納し、確定キーまたは、終了条件になった時点を検出するまで待ちます。

制御コードを入力した場合は、そのコードの処理を行います。

(1)文字列入力編集処理

文字列入力は以下のキー操作により入力文字の編集が行え、これらのキーに関しては文字列格納エリアには格納されません。

また、これらの操作は入力領域中でのみ有効です。

表4. 1 文字列入力編集画面

名 称	デフォルト 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	1234567890 123 1234567890 1234567890 1234 1234567890 !	1234567890 123 1234567890 1234567890 1234 1234567890 !
→	→(F3)	00h	1Ch	カーソルを1文字右へ移動します	1234567890 123 1234567890 1234567890 1234 1234567890 1234	1234567890 123 1234567890 1234567890 1234 1234567890 1234
クリア	クリア(CLR)	00h	0Ch	入力文字を全て削除します	1234567890 123 1234567890 123 1234567890 1234567890 1234	1234567890 123 1234567890 123 1234567890 1234567890 1234
後退	後退(BS)	00h	08h	カーソル前の1文字を削除します	1234567890 123 1234567890 1234567890 1234567890 1234567890 abcd 1234567890 12345 12345	1234567890 12 1234567890 1234567890 1234567890 1234567890 abcd 1234567890 2345 2345
削除	DEL(F4)	00h	10h	カーソル上の文字を削除します	1234567890 123 1234567890 1234567890 1234567890 1234567890 abcd 1234567890 abcd	1234567890 13 1234567890 1234567890 1234567890 1234567890 bcd 1234567890 bcd

4.1.4. 数値入力

アプリケーション指定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入力を行うものです。

アプリケーションが指定したバッファに数値データを格納し、確定キーまたは、終了条件を検出するまで待ちます。

制御コードを入力した場合は、そのコードの処理を行います。

尚、数値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います。

(1)数値入力編集処理

数値入力は以下のキー操作により入力文字の編集が行え、これらのキーに関しては数値列格納エリアには格納されません。

また、これらの操作は入力領域中でのみ有効です。

表4. 2 数値入力編集画面

名 称	デフォルトキー	コード		機 能	動作例	
		属性	コード		入力前	入力後
+	なし	00h	2Bh	「-」(マイナス記号)表示中の場合、「-」を削除します	1234567890 -123	1234567890 123
-	-(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)	1234567890 -123 1234567890 -1 1234567890 - 1234567890 1234567890	1234567890 -123 1234567890 -1 1234567890 - 1234567890 1234567890
.	.(F4)	00h	2Eh	カーソル位置に「.」(ピリオド)記号を 付加します (入力領域がフルの場合、およびすでに 付加された状態の場合、無効になります)	1234567890 123 1234567890 -123 1234567890 . 1234567890 123.4 1234567890 1234567890	1234567890 123. 1234567890 -123. 1234567890 . 1234567890 123.4 1234567890 1234567890
クリア	クリア (CLR)	00h	0Ch	入力文字を全て削除します	1234567890 123 1234567890 . 1234567890 . 12345 1234567890	1234567890 . 1234567890 . 1234567890 . . .
後退	後退(BS)	00h	08h	カーソル上の文字を削除します	1234567890 123 1234567890 . 1234567890 . 123456 1234567890	1234567890 12 1234567890 . 1234567890 . 12345 6123456789
削除	なし	00h	10h	カーソル上の文字を削除します	1234567890 123 1234567890 . 1234567890 . 123456 1234567890	1234567890 12 1234567890 . 1234567890 . 12345 6123456789

4.1.5. キーコードの設定

各設定可能キーに対して、キーコードの設定を行うことができます。

設定可能キーに設定可能なキーコードは属性／コードの2バイトの構成を1データとし、機能および各入力機能による動作内容を示します。

表4.3 設定キーコード

グループ	コード値		機能	1文字入力 (コード返却)	文字列入力 (文字格納)	数値入力 (文字格納)
	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			
		03h	バーコード読み込み開始(※1)			
制御 コード	00h	08h	1文字後退	○	×	×
		0Ah	改行			
		0Ch	入力領域のクリア			
		0Dh	復帰			
		10h	1文字削除			
		1Ch	カーソル右移動			
		1Dh	カーソル左移動			
その他 (ANK)	00h	XXh	文字(※2)	○	○	数字(0~9) +, -, .のみ

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

※2 ANKコードおよび制御コードの設定が可能です。

設定可能範囲(01h~80h, A0h~DFh, FDh~FFh)文字列入力では制御コードは返りません。

表4. 4 キーコード設定可能キー一覧

種別	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	不可		
	テンキー 2			
	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(←)	可能		ハーフコード読み込みを除く全て
	F2(←)			
	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	全て	
	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			

4.1.6. キー通知設定

各設定可能キーに対して、キー通知モード(イベントフラグによる通知)の設定ができます。
通知モードに設定したキーはキーコードの返却および、機能の実行はされません。

表4.5 キー通知モード設定可能キー一覧

種別	キー	設定可能
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	不可
	テンキー 2	
	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(←)	可能
	F2(←)	
	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読み込み機能が登録時に通知キーに設定した場合は、通常キーとして動作します。(OBRの読み込みは開始しません)

4.1.7. キー入力有効／無効設定

各設定可能キーに対してキー入力を無効にすることができます。

表4. 6 キー入力有効／無効設定可能キー一覧

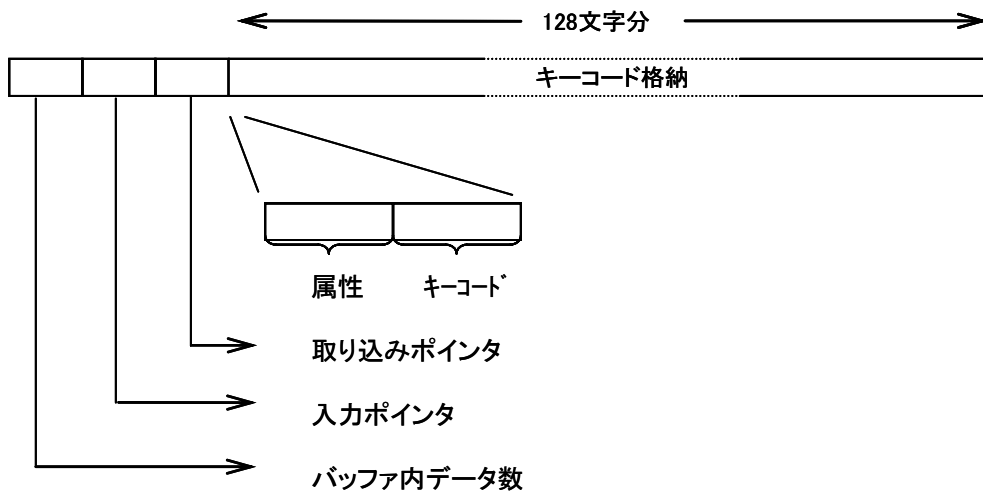
種別	キー	設定可能
ストロークキー	入力モード切替(S)	可能
	後退(BS)	
	クリア(CLR)	
	テンキー 1	
	テンキー 2	
	テンキー 3	
	テンキー 4	
	テンキー 5	
	テンキー 6	
	テンキー 7	
	テンキー 8	
	テンキー 9	
	テンキー 0	
	テンキー .	
	テンキー ENT	
	F1(←)	
	F2(↶)	
	F3(→)	
	F4(DEL)	
	F5(SP)	
F6(▲)		
F7(▼)		
F8(BL)		
マルチファンクションキー	マルチファンクションキー R	可能
	マルチファンクションキー L	
トリガーキー	右トリガーキー	不可
	左トリガーキー	

※ キー入力を無効に設定した場合は、そのキーを押しても入力されません。また、キークリック音もありません。

※ マルチファンクションキーにOBR読み込み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読み込みは開始されません)

4.1.8. キーバッファ

本機のキーバッファは、以下に示すようなリングバッファ構成になっています。



キーバッファは指定した空間に設けることができます。初期化時、キーバッファサイズ／キーバッファ開始アドレスをキー管理テーブルに設定しています。

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、本機では128キー固定としています。

電源Off→On(レジューム立ち上げ)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。

4.1.9. 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。

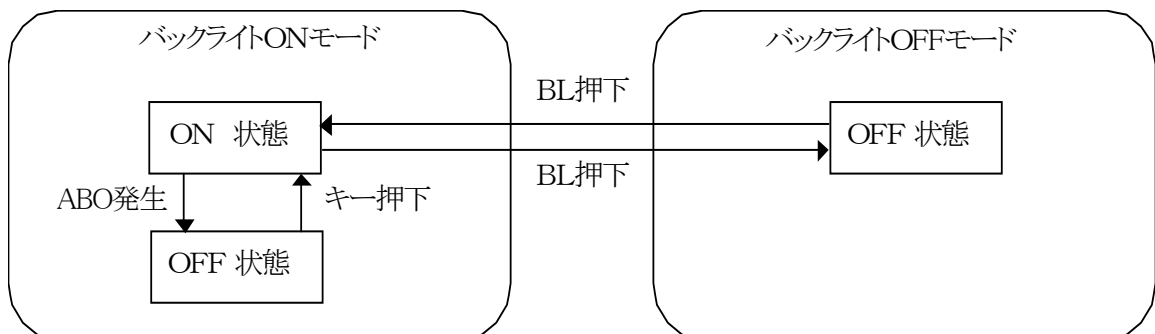


図4. 1 バックライト状態遷移図

4.1.10. 多点押し処理

本機ではキーをOBRキーと通常キーとに分けています。

OBRキーとは、バーコード読み込みを開始するキーコードが設定されているキーのことです。

(トリガーキーR・L、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)

(1) 同時押し処理

- OBRキーどうしの同時押し
バーコード読み込みが開始されます。ただし、OBRオープン中のみ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み込みが開始されます。ただし、OBRが未オープンの場合は、OBRキーは無視され通常キーが入力されます。

表4.7 同時押し時の入力キー

押下キー	動作	備考
OBRキーとOBRキー	バーコード読み込み開始	
OBRキーと通常キー	バーコード読み込み開始	
通常キーと通常キー	未確定	どちらか一方が離された時点で確定
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

(2) 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。全てのOBRキーが離された時点でバーコード読み込みを中止します。
- OBRキー押下後の通常キー
通常キーは無視されます。OBRキーが離された時点で通常キーは入力されます。
- 通常キー押下後のOBRキー押下
バーコード読み込みを開始します。

表4.8 多点押し時の入力キー

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

※ 無効キー：OBRキーであってもOBRが未オープン状態の場合

4.1.11. キーロールオーバー機能

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の入力が行われない)
 ↓
 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の入力が行われない)
 ↓
 ② キー解放 → そのまま(1の入力が行われない)
 ↓
 ② キー押下(押したまま) → そのまま(2の入力が行われない)
 ↓
 ② キー解放 → そのまま(1の入力が行われない)

4.2. キーコード

本関数で使用するキーコードは、下記のような属性／コードの2バイトで構成しています。

上位バイト 属性	下位バイト コード
-------------	--------------

4.2.1. 属性

本関数では、コードを判別するために以下の属性を設けています。

表4.9 属性一覧

属性	内容
00h	次に続くコードがアスキーコードであることを示します
01h	次に続くコードがめくりコードで、1パターン目のアスキーコードに変換されることを示します
02h	次に続くコードがめくりコードで、2パターン目のアスキーコードに変換されることを示します
03h	次に続くコードがめくりコードで、3パターン目のアスキーコードに変換されることを示します
04h	次に続くコードがめくりコードで、4パターン目のアスキーコードに変換されることを示します
FFh	次に続くコードが内部処理コードであることを示します



※ アプリケーションには属性が00hのコードしか返りません。

4.2.2. コード

本システムで使用可能なアスキーコードを以下に示します。

表4. 10 アスキーコード

	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			ア	キ	ヌ	ラ		
8	BS		(8	H	X	h	x			イ	ク	ネ	リ		
9)	9	I	Y	i	y			ウ	ケ	ノ	ル		
A	LF		*	:	J	Z	j	z			エ	コ	ハ	レ		
B			+	;	K	[k	{			オ	サ	ヒ	ロ		
C	CL	→	,	<	L	¥	l				ヤ	シ	フ	ワ		
D	CR	←	-	=	M]	m	}			ユ	ス	ヘ	ン		
E			.	>	N	^	n	~			ヨ	セ	ホ	°		
F			/	?	O	_	o				ツ	ソ	マ	°		

-  1文字入力で上位にキーコードが返り、エコーバックあり指定のとき、表示します。
文字列／数値入力では編集コードのみ処理を行い、その他のキーは無視します。
-  1文字／文字列／数値入力で上位にコードが返り、エコーバックあり指定のとき、表示します。
ただし、数値入力の場合、数値／＋／－／．のみ有効とし、その他のキーは無視します。

4.3. ファンクション詳細

ファンクション詳細を次ページより示します。

4.3.1. key_read

キーバッファから1文字入力を行います。

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

パラメータ

pkey_inp

動作内容の先頭アドレス

戻り値

【ストラク構造】

```
typedef struct st_key_inp {
    UB  ext ; /* リターン条件 (OR 指定)
              /* KEY_INT_EXT : イベント通知キー押下 */
              /* KEY_LB_EXT  : LB発生終了 */
              /* KEY_OBR_EXT  : バーコード読み込み完了 */
              /* KEY_IO_EXT   : IOボックス検出 */
              /* KEY_NON_EXT  : リターン条件なし */
    UB  echo ; /* エコーバック指定
              /* ECHO_ON   : エコーバックあり
              /* ECHO_OFF  : エコーバックなし
    H   font_size ; /* フォントサイズ
              /* LCD_ANK_LIGHT : 縮小ANK
              /* LCD_ANK_STANDARD : 標準ANK
    H   type ; /* 型指定
              /* LCD_ATTR_NORMAL : 通常
              /* LCD_ATTR_REVERS : 反転
              /* LCD_ATTR_WIDTH : 強調
    UH  column_pos ; /* 入力桁座標
    UH  line_pos ; /* 入力行座標
} KEY_INP ;
```

入力ANKコード

E_KEY_INT	イベント通知キー押下検出
E_KEY_LB	LB発生検出
E_KEY_OBR	バーコード読み込み完了検出
E_KEY_IO	IOボックス検出
E_PRM	パラメータエラー

補足

強調反転表示は LCD_ATTR_REVERS と LCD_ATTR_WIDTH をORして指定します。
ECHO_OFF 時、フォントサイズ、型指定、入力行・桁のパラメータチェックは行いません。

4.3.2. key_string

キーバッファから文字列入力を行います。

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

パラメータ

pkey_inps

動作内容の先頭アドレス

string

入力文字列格納エリアアドレス(入力桁数+1の容量が必要)

戻り値

【ストラク構造】

```
typedef struct st_key_inps {
    UB    ext ;          /* リターン条件 (OR 指定)                */
                                /* KEY_INT_EXT          : イベント通知キー押下                */
                                /* KEY_LB_EXT           : L B 発生終了                    */
                                /* KEY_OBR_EXT          : バーコード読み完了                */
                                /* KEY_CLR_EXT          : C L R キー押下                    */
                                /* KEY_IO_EXT           : I O ボックス検出                  */
                                /* KEY_FULL_BEEP        : 入力領域フルで B E E P 音 (※1) */
                                /* KEY_FULL_CHR         : 入力領域フルで処理終了                */
                                /* KEY_NON_EXT          : リターン条件なし                    */
    UB    echo ;        /* エコーバック指定                        */
                                /* ECHO_ON              : エコーバックあり                    */
                                /* ECHO_OFF             : エコーバックなし (※2)                */
    H     font_size ;   /* フォントサイズ                            */
                                /* LCD_ANK_LIGHT        : 縮小 A N K                        */
                                /* LCD_ANK_STANDARD     : 標準 A N K                          */
    H     type ;        /* 型指定 (OR 指定)                            */
                                /* LCD_ATTR_NORMAL      : 通常                                */
                                /* LCD_ATTR_REVERS      : 反転                                */
                                /* LCD_ATTR_WIDTH       : 強調                                */
    UH    len ;         /* 入力文字数 (半角)                            */
    UH    column_pos ;  /* 入力桁座標                                    */
    UH    line_pos ;   /* 入力行座標                                    */
    UH    column_len ; /* 入力文字位置 (半角)                            */
    UH    clr_type ;   /* 数値入力用予約領域 (※3)                            */
} KEY_INPS ;
```

関数結果

E_OK	イベント通知キー押下検出
E_KEY_INT	LB発生検出
E_KEY_LB	バーコード読み込み完了検出
E_KEY_OBR	バーコード読み込み完了検出
E_KEY_CLR	クリアキー押下検出
E_KEY_FUL	入力領域フル終了
E_KEY_IO	IOボックス検出
E_PRM	パラメータエラー

補足

- 1 「入力領域フルでBEEP音」を指定するとBEEP音は鳴りますが終了はしません。
- 2 ECHO_OFF 時、フォントサイズ、型指定、入力行桁、入力文字位置のパラメータチェックは行いません。
- 3 clr_type は数値入力用です。テーブル構造を同一にするために入れてあります。

4.3.3. key_num

キーバッファより文字列入力を行います。数値(0~9)および記号(+, -, .)以外は無視されます。

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

パラメータ

pkey_inps

動作内容の先頭アドレス

string

入力文字列格納エリアアドレス(入力桁数+1の容量が必要)

戻り値

【ストラク構造】

```
typedef struct st_key_inps {
    UB    ext ;          /* リターン条件 (OR 指定)                */
                    /* KEY_INT_EXT          : イベント通知キー押下                */
                    /* KEY_LB_EXT           : LB発生終了                        */
                    /* KEY_OBR_EXT          : バーコード読み込み完了            */
                    /* KEY_CLR_EXT          : CLRキー押下                        */
                    /* KEY_IO_EXT           : IOボックス検出                    */
                    /* KEY_FULL_BEEP        : 入力領域フルでBEEP音(※1)          */
                    /* KEY_FULL_CHR         : 入力領域フルで処理終了            */
                    /* KEY_NON_EXT          : リターン条件なし                */
    UB    echo ;         /* エコーバック指定                        */
                    /* ECHO_ON              : エコーバックあり                    */
                    /* ECHO_OFF             : エコーバックなし(※2)            */
    H     font_size ;    /* フォントサイズ                          */
                    /* LCD_ANK_LIGHT        : 縮小ANK                            */
                    /* LCD_ANK_STANDARD     : 標準ANK                            */
    H     type ;         /* 型指定 (OR 指定)                        */
                    /* LCD_ATTR_NORMAL      : 通常                                */
                    /* LCD_ATTR_REVERS      : 反転                                */
                    /* LCD_ATTR_WIDTH       : 強調                                */
    UH    len ;          /* 入力文字数 (半角)                        */
    UH    column_pos ;   /* 入力桁座標                              */
    UH    line_pos ;     /* 入力行座標                              */
    UH    column_len ;   /* 文字列入力用予約領域(※3)              */
    UH    clr_type ;     /* 初期データ表示後のクリア              */
                    /* KEY_NUM_CLR_ON       : する                                */
                    /* KEY_NUM_CLR_OFF      : しない                              */
} KEY_INPS ;
```

関数結果

E_OK	正常終了
E_KEY_INT	イベント通知キー押下検出
E_KEY_LB	LB発生検出
E_KEY_OBR	バーコード読み込み完了検出
E_KEY_CLR	クリアキー押下検出
E_KEY_FUL	入力領域フル終了
E_KEY_IO	IOボックス検出
E_PRM	パラメータエラー

補足

- 1 「入力領域フルでBEEP音」を指定するとBEEP音は鳴りますが終了はしません。
- 2 ECHO_OFF 時、フォントサイズ、型指定、入力行桁、初期データクリアのパラメータチェックは行いません。
- 3 column_len は文字列入力用です。テーブル構造を同一にするために入れてあります。

4.3.4. 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。
バッファ内にデータが存在しない場合はその旨を通知します。
読み込みポインタは更新されません。

```
ER key_check(void);
```

パラメータ

なし

戻り値

関数結果	
000000xxh	データあり(ANKコード)
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

補足

4.3.5. key_clear

キーバッファをクリアします。

```
ER key_clear(void);
```

パラメータ

なし

戻り値

関数結果

E_OK 正常終了

補足

4.3.6. 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に対しキーコードの設定をします。
また、現在設定されているキーコードの取得を行います。

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

パラメータ

func_mode

動作モード

FNC_SET	設定
FNC_GET	取得

func_num

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

func_data

ファンクションキーデータアドレス

戻り値

【ストラク構造】

```
typedef struct stKeyCode{
    UB attr ; /* 属性 */
    UB code ; /* コード */
} KEYFORM ;
```

関数結果

E_OK	正常終了
E_PRM	パラメータエラー

補足

4.3.7. key_fnc_mode

各ファンクションキーに対して通知モード(イベントフラグによる通知)の設定/解除を行います。

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

パラメータ

mode

動作モード

FNC_MODE_SET	設定
FNC_MODE_CLR	解除
FNC_MODE_RED	取得

func_num

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

flgid

イベントフラグID (解除時は不要です)

setptn

セットするビットパターン(解除時は不要です)

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

取得時にキー通知設定されていないキーは、イベントフラグIDが NULL でリターンします。アプリケーション用にイベントフラグIDおよびセットするビットパターンが推奨されています。

4.3.8. key_select

各設定可能キーに対して、キー入力モードの設定および削除を行います。

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

パラメータ

mode

動作モード

SEL_SET	設定
SEL_GET	取得
SEL_RES	解除(全て有効)

key_sel

有効無効キーテーブルアドレス

戻り値

【ストラク構造】

```
typedef struct stKeySel {
    UB s ; /* 入力モード切替 (S) */
    UB bs ; /* 後退 (BS) */
    UB clr ; /* クリア (CLR) */
    UB ten1 ; /* テンキー 1 */
    UB ten2 ; /* テンキー 2 */
    UB ten3 ; /* テンキー 3 */
    UB ten4 ; /* テンキー 4 */
    UB ten5 ; /* テンキー 5 */
    UB ten6 ; /* テンキー 6 */
    UB ten7 ; /* テンキー 7 */
    UB ten8 ; /* テンキー 8 */
    UB ten9 ; /* テンキー 9 */
    UB ten0 ; /* テンキー 0 */
    UB ten ; /* 小数点 */
    UB ent ; /* リターン */
    UB func1 ; /* F 1 (一) */
    UB func2 ; /* F 2 (←) */
    UB func3 ; /* F 3 (→) */
    UB func4 ; /* F 4 (DEL) */
    UB func5 ; /* F 5 (SP) */
    UB func6 ; /* F 6 (▲) */
    UB func7 ; /* F 7 (▼) */
    UB func8 ; /* F 8 (BL) */
    UB mltr ; /* マルチファンクションキー R */
    UB mltl ; /* マルチファンクションキー L */
} KEYSSEL ;
```

左記のエリアに以下のデータを格納します。

キー入力有効 (KEY_MODE_ENA)

キー入力無効 (KEY_MODE_DIS)

関数結果

E_OK

E_PRM

正常終了

パラメータエラー

補足

5. OBR 部

5.1. 基本仕様

この章では本製品のレーザースキャナ部および、OBR の基本仕様について記載します。

5.1.1. レーザースキャナ部

表5.1 レーザースキャナ性能

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

5.1.2. デコード仕様

表5.2 読み取り可能コード

WPC	JAN規格	JIS X0501 JAN-13, JAN-8 JAN-13 addon(+2, +5), JAN-8 addon(+2, +5)
	EAN規格	General Specification for the Article Symbol Marking EAN-13, EAN-8 EAN-13 addon(+2, +5), EAN-8 addon(+2, +5)
	UPC規格	UPC Symbol Specification Jan 1986 UPC-A, UPC-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		

表5. 3 読み取り桁数と出力フォーマット

バーコード種類	規格	読取桁数	出力フォーマット	備考
WPC	JAN-13	13	FFMMMMNNNNNCJ	F:カントリーフラグ
	EAN-13	13	FFMMMMNNNNNCJ	M:生産者コード
	JAN-8	8	FFMMMNCCJ	N:商品コード
	EAN-8	8	FFMMMNCCJ	S:システムメンバーキャラクタ
	JAN-13 addon+2	15	FFMMMMNNNNNCAAJ	A:addon データ
	EAN-13 addon+2	15	FFMMMMNNNNNCAAJ	J:終了コード(CR または LF または CR+LF)
	JAN-13 addon+5	18	FFMMMMNNNNNCAAAAAJ	UPC-B を除きチェックデジット(mod 10)の計算は必ず行われます
	EAN-13 addon+5	18	FFMMMMNNNNNCAAAAAJ	
	JAN-8 addon+2	10	FFMMMNCAAJ	
	EAN-8 addon+2	10	FFMMMNCAAJ	
	JAN-8 addon+5	13	FFMMMNCAAAAAJ	
	EAN-8 addon+5	13	FFMMMNCAAAAAJ	
	UPC-A	12	0SMMMMNNNNNCJ	
	UPC-B	12	0SMMMMNNNNNCJ	
	UPC-A addon+2	14	0SMMMMNNNNNCAAJ	
	UPC-B addon+2	14	0SMMMMNNNNNCAAJ	
	UPC-A addon+5	17	0SMMMMNNNNNCAAAAAJ	
	UPC-B addon+5	17	0SMMMMNNNNNCAAAAAJ	
	UPC-E	(7),8	0MMNNMCJ	最後の M:0~2
		(7),8	0MMNN3CJ	
		(7),8	0MMNN4CJ	
		(7),8	0MMMMNCJ	最後の N:5~9
	UPC-E addon+2	(9),10	0MMNNMCAAJ	最後の M:0~2
		(9),10	0MMNN3CAAJ	
		(9),10	0MMNN4CAAJ	
		(9),10	0MMMMNCAAJ	最後の N:5~9
	UPC-E addon+5	(12),13	0MMNNMCAAAAAJ	最後の M:0~2
		(12),13	0MMNN3CAAAAAJ	
		(12),13	0MMNN4CAAAAAJ	
		(12),13	0MMMMNCAAAAAJ	最後の N:5~9
	UPC-E(+UPC-A)	6+12	MMNNMCJSM0000NNNCJ	6 桁目の M:0~2
		6+12	MMNN3CJSM0000NNNCJ	
	6+12	MMNN4CJSM0000NNNCJ		
	6+12	MMMMNCCJSM0000NNNCJ	6 桁目の N:5~9	
JAN-13	14	OFFMMMMNNNNNCJ	GTIN	
EAN-13	14	OFFMMMMNNNNNCJ	GTIN	
JAN-8	14	OOOOOOFMMNCCJ	GTIN	
EAN-8	14	OOOOOOFMMNCCJ	GTIN	
UPC-A	14	OOSMMMMNNNNNCJ	GTIN	
UPC-E	14	OOOOO0MMNNMCJ	GTIN 最後の M: 0~2	
	14	OOOOO0MMNN3CJ	GTIN	
	14	OOOOO0MMNN4CJ	GTIN	
	14	OOOOO0MMMMNCCJ	GTIN 最後の M: 0~2	
CODE-39		3~50 *	SBBB.....BBCSJ	A:データ, B:FULL ASCII 変換後データ
		3~50 *	SAAA.....AACSJ	C:チェックデジット(mod 43)
		1~48 *	BBB.....BBCJ	チェックデジットなしの場合は、データとなります
		1~48 *	AAA.....AACJ	S:スタート・ストップキャラクタ
NW-7		3~40	SDDD.....DDDEJ	S:スタートコード(a,b,c,d のいずれか)
		1~38	DDD.....DDDJ	E:エンドコード(a,b,c,d のいずれか) D:データ
Interleaved 2 of 5		2~40	DDD.....DDDCJ	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は、データとなります 読取桁数は偶数桁のみ
Industrial 2 of 5		2~40	DDD.....DDDCJ	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は、データとなります 読取桁数は偶数桁のみ
CODE-93		1~40	DDD.....DDDJ	D:データ
CODE-128		1~64	AAA.....AAAJ	A:ASCII 変換後データ, B:ASCII 変換前データ
		1~64	SBBB.....BBCEJ	C:チェックデジット(mod 47) S:スタートコード, E:エンドコード

MSI		1~40	DDD.....DDCC」	D:データ C:チェックデジット(mod 10,mod 11) チェックデジットなしの場合は、データとなります
IATA		1~40	PADDDDDDD.....C」	P:カーボン 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はデータとなります

※ 読取桁数が、カッコの桁の場合は、出力フォーマットに「C」は付加されません

5.2. 機能

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

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

5.2.1. 1文字／文字列の読込み

(1) 1文字リード

OBR バッファから1文字を読出します。

(2) 文字列リード

OBR バッファから1ラベル(コード)分読出します。

5.2.2. OBR データバッファの状態チェック

OBR バッファのデータ格納状態をチェックし、バッファ内の残りバイト数と残り段数を通知します。

5.2.3. OBR データバッファのクリア

OBR バッファのクリアを行います。

5.2.4. 格納先バッファの切り替え

バーコードデータの出力先をキーバッファに切りかえる事により、読取ったデータをキー入力と同時に扱うことができます。

初期状態は、OBR バッファを設定しています。

(モード設定参照)

5.2.5. モード設定

(1) 項目設定一覧

表5.4 設定項目一覧

項目	
バーコード関連	読み取りコード種別
	読み取り桁数
出力関連	出力先バッファ
	出力フォーマット
	終了コード
	チェックキャラクタの出力
読み取り関連	読み取り方法
	スキャン時間
	読み取り回数
	照合回数
	チェックデジットの計算
	同一ラベルの二度読み防止
	ブザー制御
	LED制御
読み取り動作	
電源立ち上げ関連	立ち上げモード

(2) 項目設定詳細

以降に各項目の詳細について記載します。

① 読み取りコード

特定のコードしか読み取らない場合、デコード処理の処理時間の関係から特定のコードのみを設定しておくことを推奨します。

表5. 4 読取りコード

設定条件	コード	備考
自動機別 (全てのコードを選択した時)	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	↑ デコードの優先順位 高い ↓ デコードの優先順位 低い
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	複数の設定が可能です (複数コードを設定した場合の 優先順位は上段の通りです)

※ 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より EAN128 出力を指定してください。

②読み取り桁数

コードごとに読み取り桁数の有効範囲の指定が可能です。

制限事項

誤読防止のため複数のコードが同時に設定された場合、CODE-39、NW-7、Interleaved 2of5 に関しては、有効範囲の変更を行いません。

Interleaved 2of5 で奇数桁の指定をした場合、最小桁は、指定+1の偶数までが読み取り可能となり、最大桁は、指定-1の偶数までが読み取り可能となります。従って、最大最小桁に同一の奇数を指定した場合、何も読み取れなくなります。

CODE-39 の 1 桁、NW-7 の 1 桁、Interleaved 2of5 の 2 桁を読み取りたい場合は、コード限定の読み取りを指定します。

< 設定範囲 >

表5.5 設定範囲

WPC	:	桁数は固定（設定不可能）
CODE-39	:	1～38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（スタート/ストップキャラクタを含みません）
Industrial 2of5	:	2～40桁
Interleaved 2of5	:	2～40桁
CODE-93	:	1～40桁
CODE-128	:	1～64桁
MSI	:	1～40桁
IATA	:	1～40桁（チェックデジットありの場合は、2～40桁）

< 設定範囲 > 複数のコードを同時に設定した場合

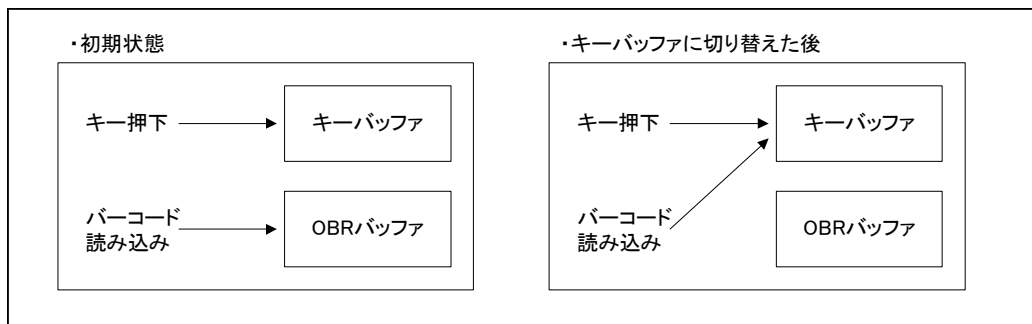
表5.6 設定範囲（複数のコードを同時に設定した場合）

CODE-39	:	2～38桁（スタート/ストップキャラクタを含みません）
NW-7	:	2～38桁（スタート/ストップキャラクタを含みません）
Interleaved 2of5	:	4～40桁

③出力バッファ

データの出力先をキーバッファに切り替えることにより、読み取ったデータをキー入力と同等に扱うことができます。初期状態は、OBR バッファを設定しています。

図5.1 出力バッファの切り替え



切り替え時点で切り替え元バッファ内に残っているデータの取り扱いを、次表に記載します。

表5.7 データの取り扱い

バッファ名	内容
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、切り替えを行ってからOBRバッファのクリアを指示してください
キーバッファ	保存します

④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。

表5.8 バーコード一覧

バーコードの種類	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	変更不可
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 2 of 5	2~40桁	同左	偶数桁のみ設定して下さい
Interleaved 2 of 5	2~40桁	4~40桁	偶数桁のみ設定して下さい
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	

⑤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

⑥チェックキャラクタの出力

次に記載するバーコードの種類は、チェックキャラクタの出力が可能です。

表5.9 バーコード一覧

バーコードの種類	初期状態
CODE-39	出力有り
UPC-E	出力有り
Industrial 2of5	出力有り
Interleaved 2of5	出力有り

⑦読み取り方法

読み取り方法は次の項目から選択が可能です。

表5.10 読み取り方法

読み取り方法	説明	読み取り終了条件
単発読み	トリガキーを押下すると読み取り可能状態となり、読み取り完了後待機状態となります	・スキャン時間経過 ・読み取り完了
連続読み (トリガキー有り)	トリガキーを押下している間常に読み取り可能状態となります	・前コード読み取り完了後、スキャン時間が経過 ・指定読み取り回数分の読み取り完了 ・トリガキー離し

⑧スキャン時間

トリガキーを押下した後の読み取り可能時間を「動作環境メニュー」または、データ管理部が提供する関数で設定できます。(設定した時間を経過すると、自動的に読み取り待機状態となります)
1~9秒まで設定することが可能です。

⑨読み取り回数

連続読みの場合の読み取り可能回数を「動作環境メニュー」または、データ管理部が提供する関数で設定できます。(設定した回数分読み取りを完了すると、自動的に読み取り待機状態となります)

1~9回まで設定することが可能です。

⑩照合回数

読み取ったデータに対する信頼性を強化するための照合回数を「動作環境メニュー」または、データ管理部が提供する関数で設定できます。(照合回数をもとに内部で設定され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。

⑪チェックデジットの計算

各コードごとにチェックデジットの計算を有効/無効にすることができます。

(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。)

初期値:有効

⑫同一ラベルの二度読み防止

連続読みにて読み取りを行なっている場合、二度読み防止のため同一ラベルを連続して読むことはできません。

⑬ブザー制御

1コードごとの読み取り完了をブザー音によって通知することができます。

また、ブザー制御を無効にすることも可能です。

注意 ブザーの音量は「環境設定メニュー」または、データ管理部が提供する関数によって設定することができます。そのため音量がオフになっている場合、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。

⑭LED 制御

1コードごとの読み取り完了をLEDの点灯によって通知することができます。

またLED制御を無効にすることも可能です。

制御内容: 読み取りコードが正常な場合、LEDを一定秒間緑色に点灯したのち消灯します。

読み取りコードが異常な場合、LEDを一定秒間赤色に点灯したのち消灯します。

読み取りコードが異常になる要因

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

⑮バイブレータ制御

1コード毎の読み取り完了を、バイブレータの振動によって通知することができます。

また、バイブレータ制御を無効にもできます。

制御内容: 読み取りコードが正常な場合、バイブレータを振動させます。

⑯読み取り動作

通常読み: オープン後、クローズするまで連続して読み取りが行われます。

段数読み: オープン後、指定された回数分の読み取りが行われます。

⑰ 立ち上げモード

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 するかしないのことで、

OBR では次に記載する各モードによりこの立ち上げモード選択できます。

表5. 11 立ち上げモード一覧

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

(3) 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。

5.2.6. レーザー発光幅制御

従来のレーザースキャナにおいて遠距離のスキャンを行なうとレーザーの走査幅が広がり、読取りたいバーコード以外にも、複数のバーコードをスキャンしてしまう事があります。

レーザー発光幅制御は、読取りたいバーコードの幅にあわせてレーザーの照射を制御することにより、確実にバーコードの読取りを行なうことができます。

レーザー発光幅制御を行なうために次の関数があります。

表5. 12 レーザー発光幅制御機能一覧

機能	内容	備考
キャリブレーション機能	マシン固体差によるレーザー発光幅位置を決定するための機能です	システムメニューまたはアプリケーションから起動可能です(*1)
レーザー発光幅設定機能	4段階のレーザー発光幅を設定するための機能です(レーザー発光幅制御なしの設定も可能)	関数として提供されます
レーザー発光幅微調整機能	現状設定されているレーザー発光幅を微調整し、広める/狭めることが可能です	関数として提供されます

(*1) : 共通関数を参照して下さい。

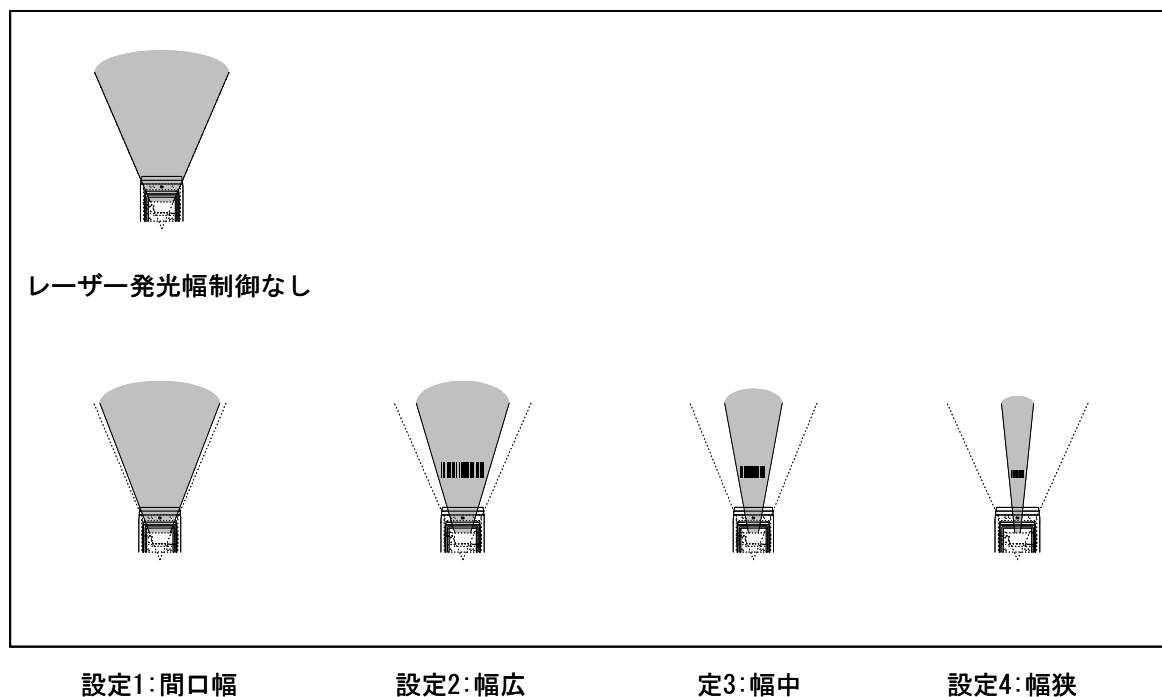
(1) キャリブレーション

マシンの固定差によるレーザー発光幅位置(レーザーの発光幅)を調節するための機能です。

システムメニューや関数(共通関数)で行なうことができます。

通知モードが設定されている状態で、キャリブレーションを実行した場合、通知イベントが発生してもキャリブレーション処理から戻れません。(電源 OFF キー/APO を通知モードにしている場合は、キャリブレーション実行中にそのイベントが発生しても電源 OFF されません)

(2)レーザー発光幅設定機能



- ※ 設定したレーザー発光幅の段階は、バーコードの OPEN/CLOSE 時にも保存されています。
 ※ レジューム OFF/リセットにより、「間口幅」に設定されます。

表5. 13 レーザー発光幅

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°

(3)レーザー発光幅微調整機能

現状設定されているレーザー発光幅を微調整し、±5段階に広める／狭めることができます。リセットするとクリアされます。

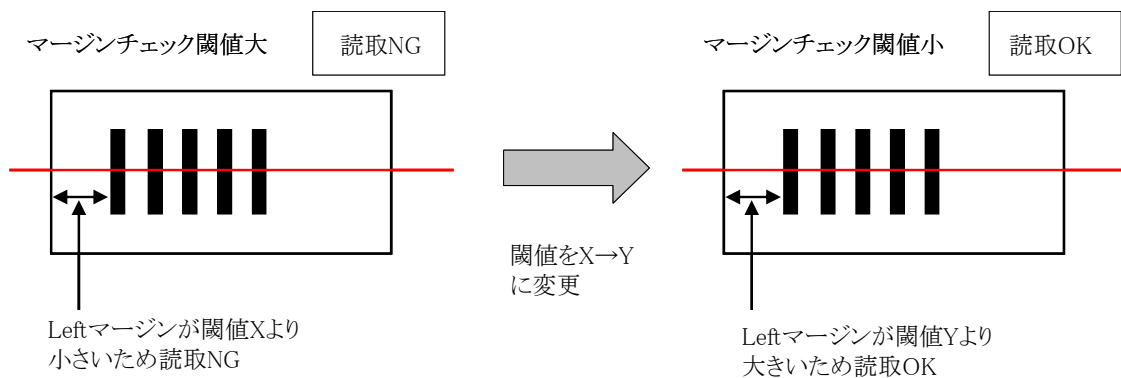
5.2.7. カスタマイズ機能

世の中に存在する全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 させることは困難であり、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、特定ユーザ向けのデコーダを開発しているケースが多々あります。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み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います

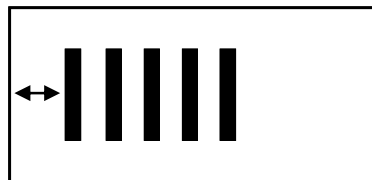
(1) Right/Left マージンの閾値変更

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うことにより読み取りが行えるようになります。

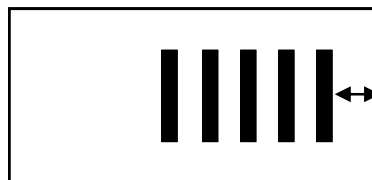


マージンチェック閾値を変更すると、以下のようなバーコードの読取が行えるようになります。

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

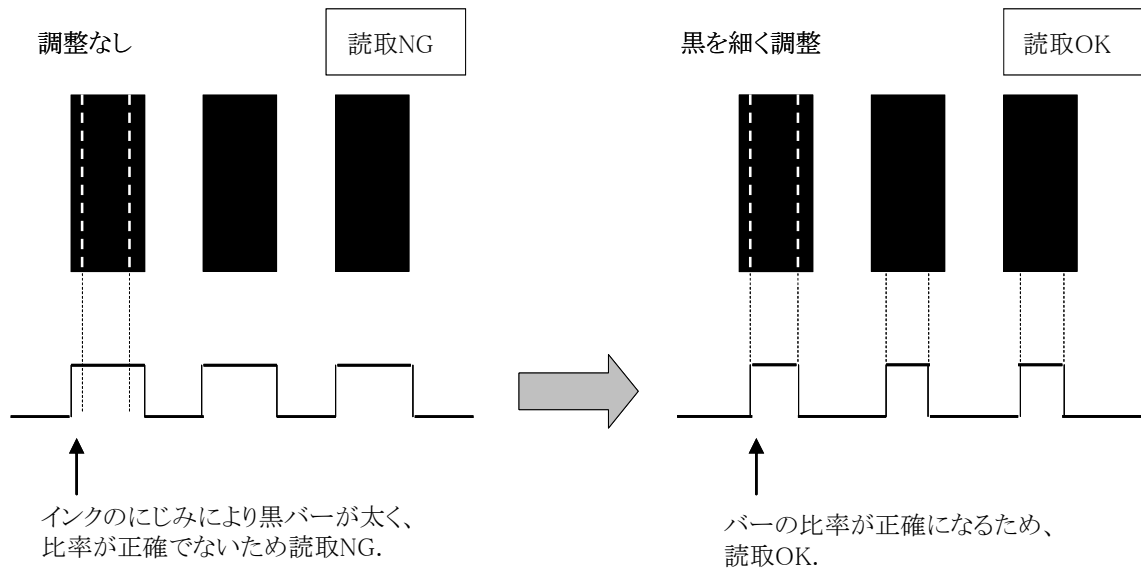


Rightマージン・Leftマージンの両方が狭いバーコード



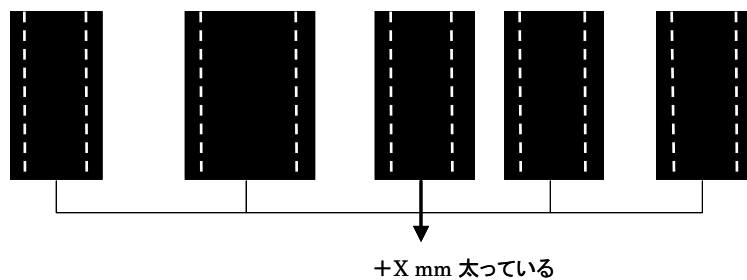
(2) バーの太り・細り補正值変更

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・細りの調整を行うことにより読取が行えるようになります。この調整は、全ての黒バーまたは、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。

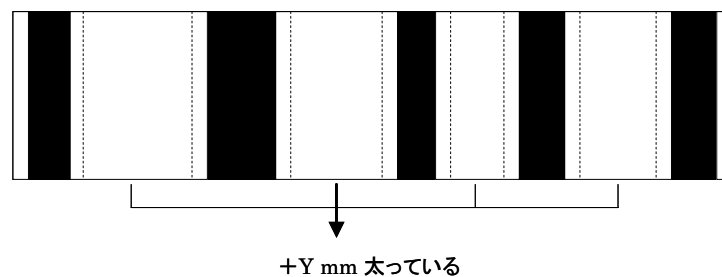


太り・細り補正值を変更すると、以下のようなバーコードの読取が行えるようになります。

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



(3) 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。

5.3. ファンクション詳細

ファンクション詳細を次ページより示します。

5.3.1. OBR_open

OBRのオープンを行います。

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

パラメータ

mode

オープンモード(以下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
- ・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

※1 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。

※2 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)

※3 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

関数結果

- | | |
|-----------|--------|
| E_OK | 正常終了 |
| E_OBR_PON | オープン済み |

補足

5.3.2. OBR_close

OBRの終了処理を行います。

```
ER OBR_close(void);
```

パラメータ

なし

戻り値

関数結果

E_OK

E_OBR_POF

正常終了

クローズ済み

補足

5.3.3. OBR_getc

OBRデータを1文字読みます。

```
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

戻り値

OBRデータ(1文字)

補足

5.3.4. OBR_gets

OBRデータを文字列で読み込みます。

```
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

lengs

データバイト数格納先へのポインタ

戻り値

関数結果

E_OK 正常終了

補足

5.3.5. OBR_flush

OBRバッファのクリアを行いません。
(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(void);
```

パラメータ

なし

戻り値

関数結果

E_OK

正常終了

補足

5.3.6. OBR_stat

OBRバッファの状態を読み込みます。

```
ER OBR_stat(  
    W * leng ,  
    UB * lcnt  
);
```

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

関数結果

E_OK

正常終了

補足

5.3.7. OBR_moderd

OBRの動作モードを読み込みます。

```
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

関数結果

E_OK

正常終了

補足

5.3.8. 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

関数結果

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

補足

本関数は、OBR_moderd 関数で現在設定を読み込んで、それに対して変更をかける様にしてください。

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)	<div style="display: flex; justify-content: space-between; align-items: center;"> b31 b0 </div> <p style="margin-left: 150px;">予約領域</p> <ul style="list-style-type: none"> CODE39 NW-7 WPC addon WPC UPCE addon UPCE IDF ITF CODE93 CODE128 MSI IATA <p>* 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。</p>	全て選択 (FFFh)	○	○

項目	内容	初期値	参照	設定																																																																																																		
読取り桁数の設定		左の表参照	○	○																																																																																																		
出力フォーマットの設定	<table border="1"> <thead> <tr> <th>リザーブ</th> <th>Min</th> <th>Max</th> <th>出力フォーマット</th> <th>チェックデジット</th> <th>チェックキャラクタ</th> <th></th> </tr> </thead> <tbody> <tr> <td>FFh</td> <td>2</td> <td>38</td> <td>0*1</td> <td>0</td> <td>1</td> <td>CODE39</td> </tr> <tr> <td>FFh</td> <td>2</td> <td>38</td> <td>0*2</td> <td>-</td> <td>-</td> <td>NW-7</td> </tr> <tr> <td>FFh</td> <td>10</td> <td>18</td> <td>0*7</td> <td>1</td> <td>-</td> <td>WPC(UPCE以外)addon</td> </tr> <tr> <td>FFh</td> <td>8</td> <td>13</td> <td>0*8</td> <td>1</td> <td>-</td> <td>WPC(UPCE以外)</td> </tr> <tr> <td>FFh</td> <td>9</td> <td>12</td> <td>0*9</td> <td>1</td> <td>1</td> <td>UPCE addon</td> </tr> <tr> <td>FFh</td> <td>7</td> <td>7</td> <td>1*3</td> <td>1</td> <td>1</td> <td>UPCE</td> </tr> <tr> <td>FFh</td> <td>2</td> <td>40</td> <td>-</td> <td>1</td> <td>1</td> <td>Industrial 2of5</td> </tr> <tr> <td>FFh</td> <td>4</td> <td>40</td> <td>-</td> <td>1</td> <td>1</td> <td>ITF</td> </tr> <tr> <td>FFh</td> <td>3</td> <td>40</td> <td>-</td> <td>1</td> <td>-</td> <td>CODE93</td> </tr> <tr> <td>FFh</td> <td>2</td> <td>64</td> <td>0*4</td> <td>1</td> <td>-</td> <td>CODE128</td> </tr> <tr> <td>FFh</td> <td>1</td> <td>40</td> <td>-</td> <td>1*5</td> <td>1</td> <td>MSI</td> </tr> <tr> <td>FFh</td> <td>1</td> <td>40</td> <td>-</td> <td>0*6</td> <td>-</td> <td>IATA</td> </tr> <tr> <td>FFh</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>リザーブ</td> </tr> </tbody> </table>	リザーブ	Min	Max	出力フォーマット	チェックデジット	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	1	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	-	-	-	-	-	リザーブ			
リザーブ	Min	Max	出力フォーマット	チェックデジット	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	1	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	-	-	-	-	-	リザーブ																																																																																																
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)	<p>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)</p> <p>チェックデジット 0:チェック無し 1:チェック有り</p> <p>チェックキャラクタ 0:出力無し 1:出力有り</p> <p>*1 CODE39出力フォーマット 0:Start/Stopコード有り 1:Start/Stopコード無し 2:Full ASCII Start/Stopコード有り 3:Full ASCII Start/Stopコード無し</p> <p>*2 NW-7出力フォーマット 0:Start/Stopコード有り 1:Start/Stopコード無し</p> <p>*3 UPCE出力フォーマット 0:UPCAへの復元コード出力有り 1:UPCAへの復元コード出力無し 2:GTINフォーマットで出力 3:UPCAへの復元コード出力無し 4:先頭の0を削除して出力</p> <p>*4 CODE128出力フォーマット 0:CODE128(変換後)又はEAN128(変換後)出力 1:CODE128(変換前)又はEAN128(変換前)出力 2:EAN128(変換後)出力(CODE128は読取り不可) 3:EAN128(変換前)出力(CODE128は読取り不可) 4:CODE128又はEAN128(変換後+認識コード)出力 6:EAN128(変換後+認識コード)出力(CODE128は読取り不可) 8:CODE128(変換後)又は EAN128(GS変換)出力 A:EAN128(GS変換)出力(CODE128は読取り不可) 10:CODE128(変換前)又は EAN128(変換後)出力 20:CODE128(変換後)又は EAN128(変換前)出力</p> <p>*5 MSIチェックデジット 0:チェックデジット無し 1:1桁,mod10 2:2桁,1st:mod11 2nd:mod10 3:2桁,1st:mod10 2nd:mod10</p> <p>*6 IATAチェックデジット 0:チェックデジット無し(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:キーボード番号/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)</p> <p>*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</p> <p>*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</p> <p>*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</p>																																																																																																					

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 <div style="border: 1px solid black; width: 40px; height: 15px; display: inline-block;"></div> 00h:単発読み 01h:連続読み(トリガキーあり) 02h:連続読み(トリガキーなし)(非公開) 03h:連続読み(トリガキー切替)(非公開)	連続読み (01h)	○	○
ブザー制御の設定 (1バイト)	b7 b0 <div style="border: 1px solid black; width: 40px; height: 15px; display: inline-block;"></div> 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 <div style="border: 1px solid black; width: 40px; height: 15px; display: inline-block;"></div> 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 <div style="border: 1px solid black; width: 40px; height: 15px; display: inline-block;"></div> 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 (ハーフコードの最後尾 に付加するコード) (1バイト)	b7 b0 <div style="border: 1px solid black; width: 40px; height: 15px; display: inline-block;"></div> 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 <div style="border: 1px solid black; width: 40px; height: 15px; display: inline-block;"></div> 00h:通常読み 01h:段数読み	通常 読み (00h)	○	○

(※1)設定は OBR_chgbuf 関数を使用して下さい。

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。

5.3.9. 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_BUF0BR

OBRバッファ

OBR_STOFF

KEYバッファ

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

5.3.10. OBR_trigmode

トリガキーによる電源オン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

状態 \ モード	0	1	2
オープン	×	○	○
クローズ	×	○	×

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

5.3.11. 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

関数結果

E_OK	正常終了
E_PRM	パラメータエラー

補足

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。

5.3.12. 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

関数結果

E_OK	正常終了
------	------

補足

設定した微調整の値は、レーザー発光幅設定を行なったり、リセットを行なうとクリアされます。

5.3.13. 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

関数結果

E_OK	正常終了
------	------

補足

5.3.14. 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

関数結果

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

補足

5.3.15. 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

関数結果

E_OK	正常終了
------	------

補足

5.3.16. 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

関数結果

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

補足

6. 通信部

6.1. 通信仕様

6.1.1. 通信インタフェース

(1) 通信ポート

本機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

※1 ベーシックIOボックス制御信号

(2) 排他制御

本機のカシオ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	

(2) 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っています。(詳細は、6.2.7.信号線制御とタイムアウト を参照してください。)

6.2. 機能

6.2.1. 通信ポートのオープン・クローズ・占有

(1) オープン

本機の通信関数部を使用してデータ通信を行うためには、通信ポートのオープンを行う必要があります。

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い、通信関数部が提供する様々なデータ通信機能を用いてデータ通信を行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います。

(2) クローズ

本機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い、通信関数部が提供するデータ通信機能を用いてデータ通信を行うことを不能にします。

本機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って下さい。

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います。

(3) 占有

通信ポートのオープンを行うと、通信ポートに電源供給を行い、通信リソースの占有を行います。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、本機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います。

6.2.2. 転送データの送信・受信

(1) 送 信

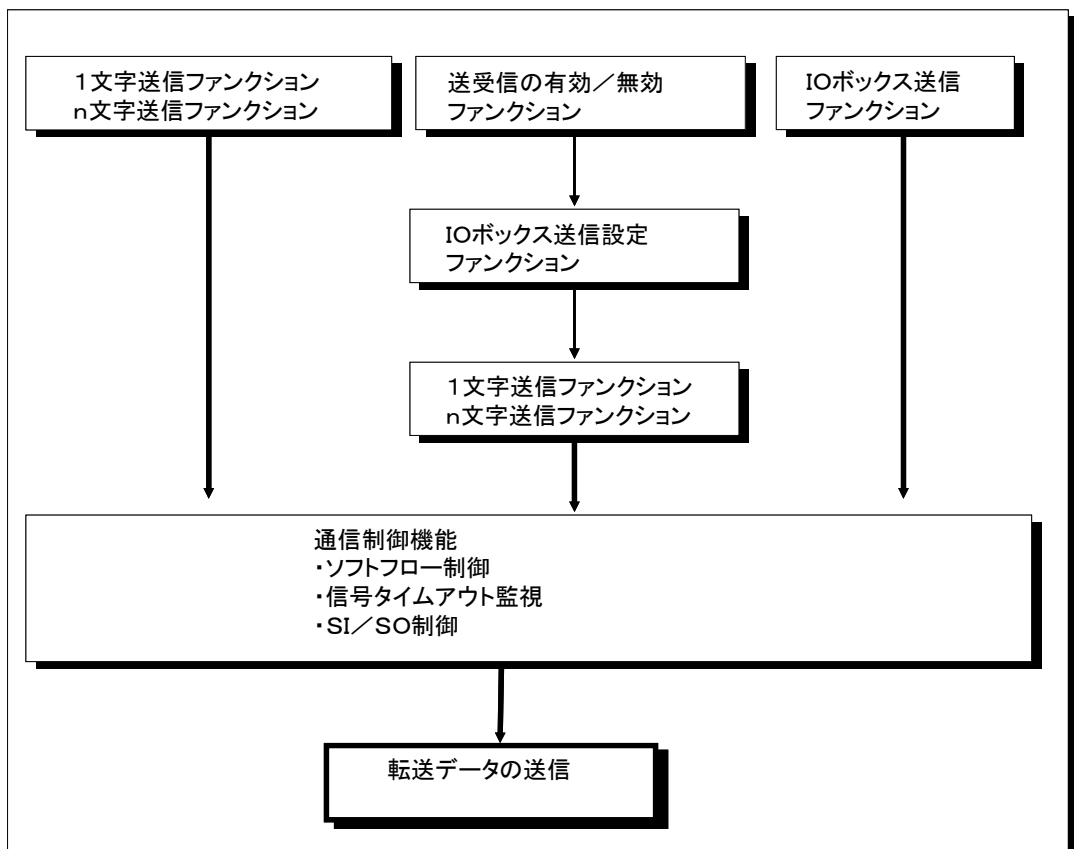
通信ポートをオープンすることで転送データの送信を行うことができます。

送信する転送データは1文字、n文字単位で扱うことができ、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います。

各ファンクションの使用例については「通信関数部補足」を参照して下さい。

【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み込みを行うことができます。

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)の他に以下の機能を持ちます。

- ・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、本機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

受信ハンドラでは以下の信号線の操作および参照を行います。

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のときOFFに設定

③受信データの読み込み

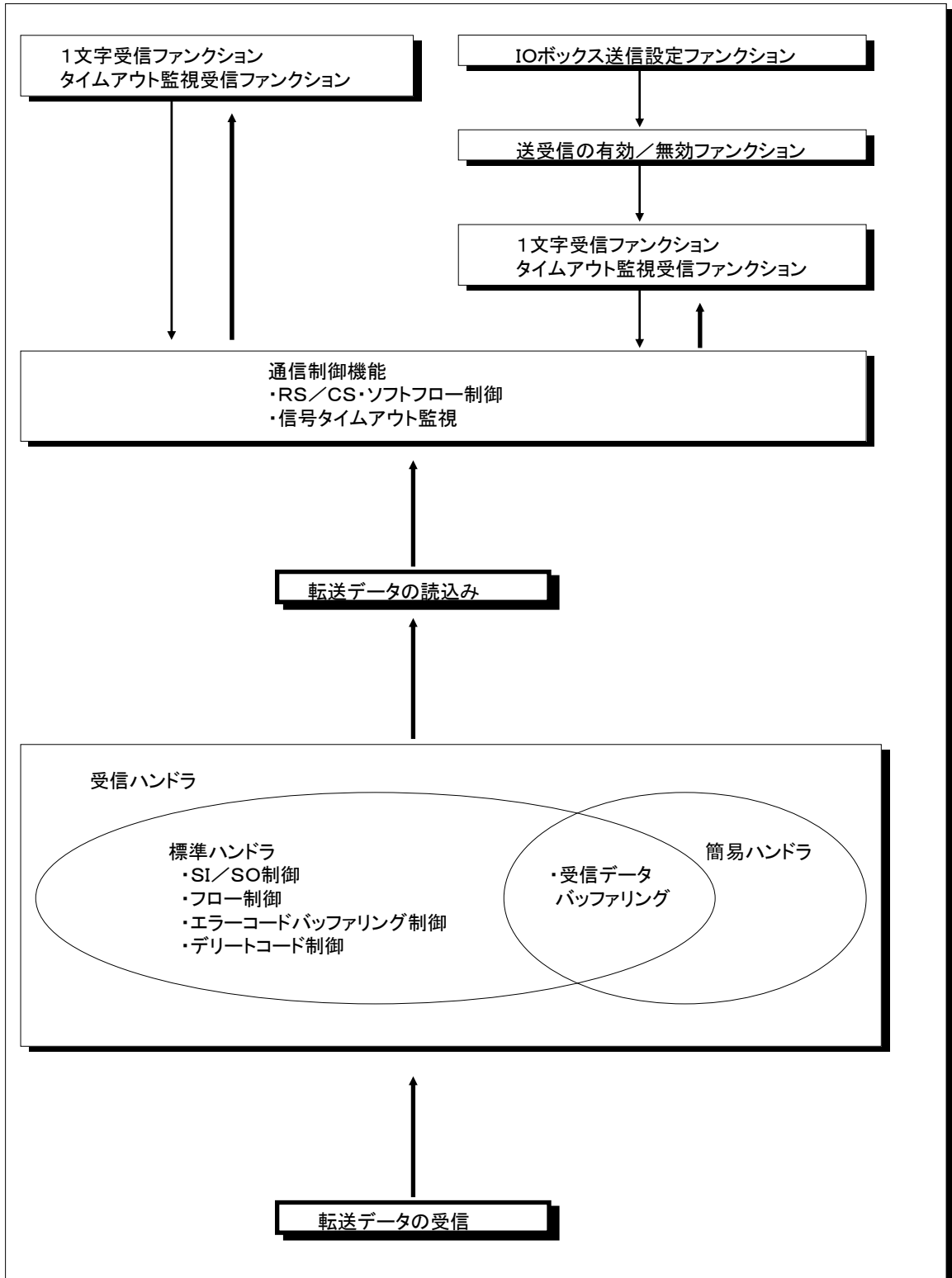
受信バッファに格納された転送データの読み込みを「1文字受信」、「タイムアウト監視受信」で行うことができます。

これらのファンクションでは読み込み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み込み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います。

各ファンクションの使用例については「通信関数部補足」を参照して下さい。

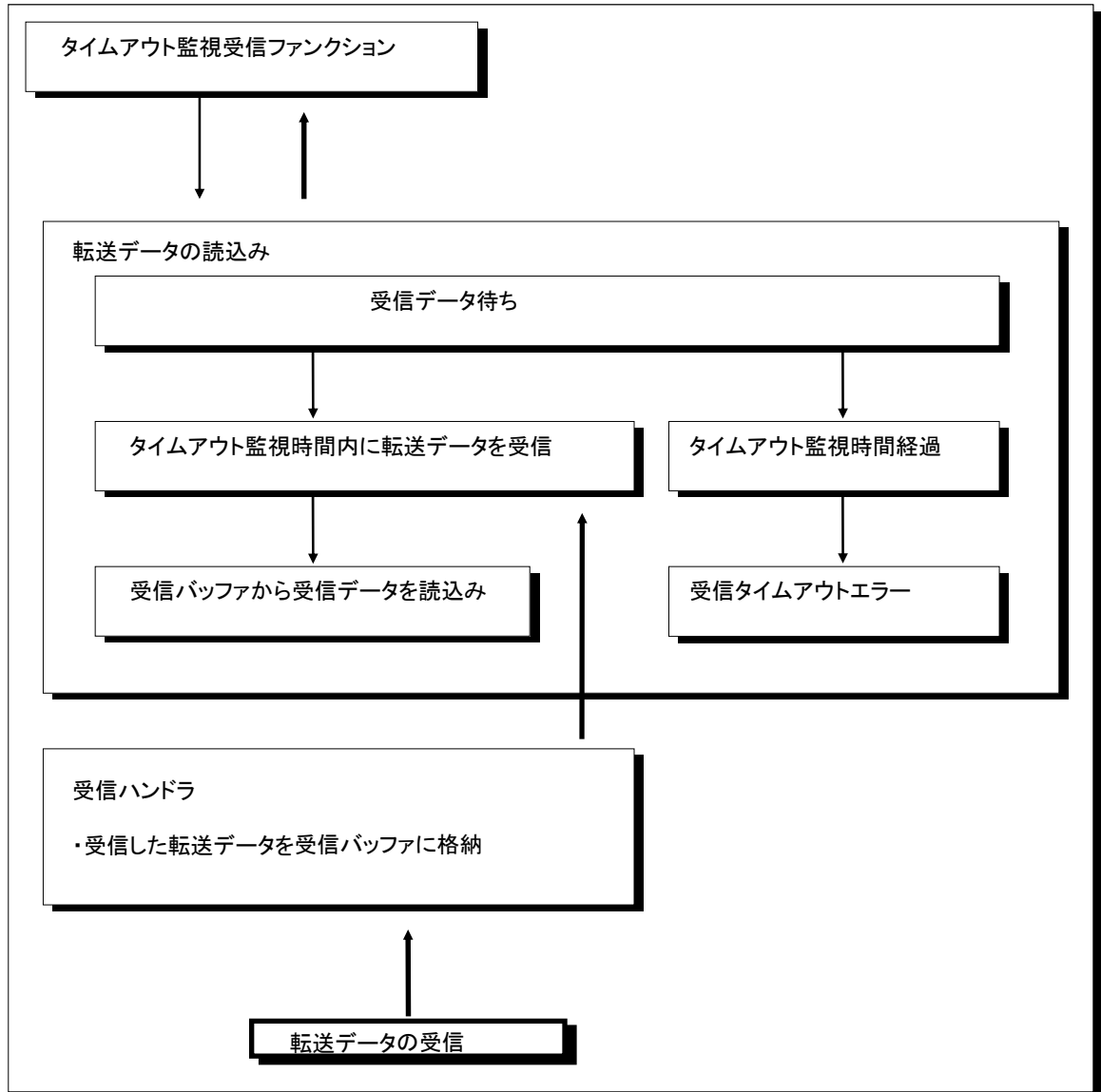
【転送データの受信および受信データの読み込みの流れ】



④タイムアウト監視

読み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ー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】



⑤パリティ、オーバーラン、フレーミングエラー

パリティ、オーバーラン、フレーミングエラーにはそれぞれ2種類のエラーステータスが在ります。これらは転送データの受信が要因で発生するエラーであり、通信関数部の受信割込みハンドラでエラーを検出して設定しますが、ファンクションコールがそれらを検出して異常終了とする制御が異なります。

a) CERR_r_PARITY、OVERRUN、FRAMINGエラーステータス

エラーステータスは受信ハンドラでエラーステータスの設定を行った後にパリティ、オーバーラン、フレーミングエラーの検出を行うファンクションコール(実行中の場合あり)で異常終了となります。

b) CERR_r_PARITY2、OVERRUN2、FRAMING2エラーステータス

これらのエラーステータスは「1文字受信」、「タイムアウト監視受信」および「受信バッファステータスのリード」ファンクションコールで異常終了となります。

ファンクションのエラー検出は、受信ハンドラでの転送データの受信とエラーの検出から時系列に行います。

また、「COMステータスのリード」ファンクションでこれらのエラーステータスのクリアは行いません。

【パリティ、オーバーラン、フレーミングエラーの検出の流れ】

パリティ、オーバーラン、フレーミングエラーを検出する各ファンクション

- ・受信データ a の読出しが行われていないとき、転送データ bd に対しての CERR_r_FRAMING で異常終了となります。
受信データ c の読出しを行なったときには、CERR_r_FRAMING はクリアしていますので正常終了となります。
- ・受信データ ac の読出し後に転送データ d を受信したとき、転送データ d に対しての CERR_r_FRAMING で異常終了となります。
その後のファンクションは、CERR_r_FRAMING はクリアしていますので正常終了となります。

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 c の読出しを行なったとき、転送データ b に対しての CERR_r_FRAMING2 で異常終了となります。
その後のファンクションは、CERR_r_FRAMING2 はクリアしていますので正常終了となります。
このようにエラーが連続した時は、初めのエラーのみを検出します。
- ・受信データ ac の読出し後に転送データ d を受信したとき、転送データ d に対しての CERR_r_FRAMING2 で異常終了となります。

受信バッファステータスのリードファンクション

- ・受信データ c の読出し前に (a は読出し済み)、転送データ b に対しての CERR_r_FRAMING2 で異常終了となります。
その後のファンクションは、CERR_r_FRAMING2 はクリアしていますので正常終了となります。この様にエラーが連続したときは、初めのエラーのみを検出します。
- ・受信データ ac の読出し後に転送データ d を受信したとき、転送データ d に対しての CERR_r_FRAMING2 で異常終了となります。

受信ハンドラ

- ・転送データ abcd の bd がフレーミングエラーであるとき受信バッファに ac を格納する。
エラーデータは破棄する。

転送データの受信

- ・ファンクションの異常終了後は「エラーステータスのリード」ファンクションでエラーエラーステータスを読み出し、クリアが行われていることを条件とします。

6.2.3. SI/SO制御

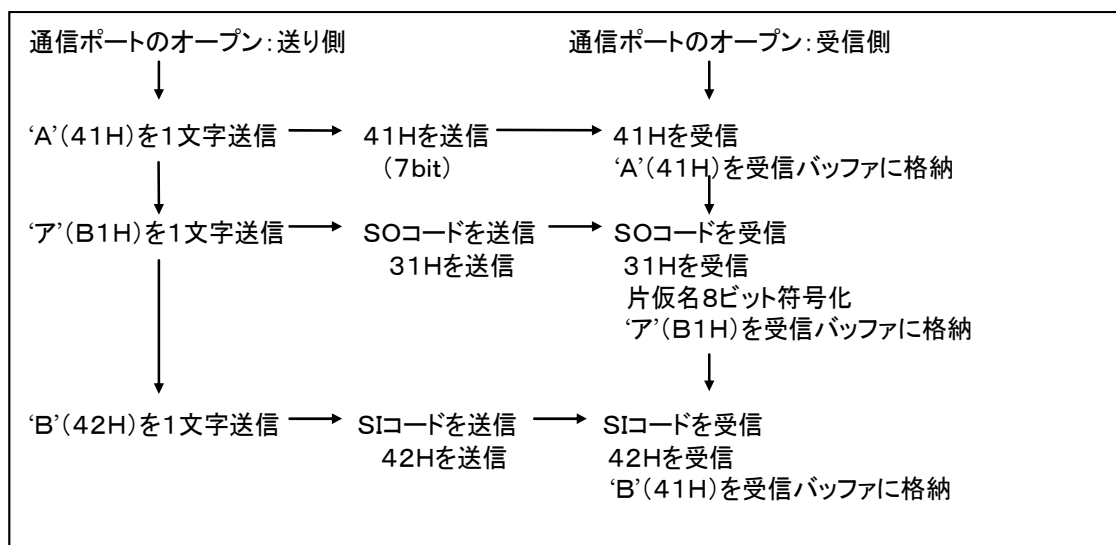
キャラクタレングスが7ビットであるときにJIS8ビット符号化文字(JIS X 0201参照)を扱う場合に使用します。

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以下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います。

【SI/SO制御の流れ】



6.2.4. 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下さい。

(2) RS/CSフロー制御

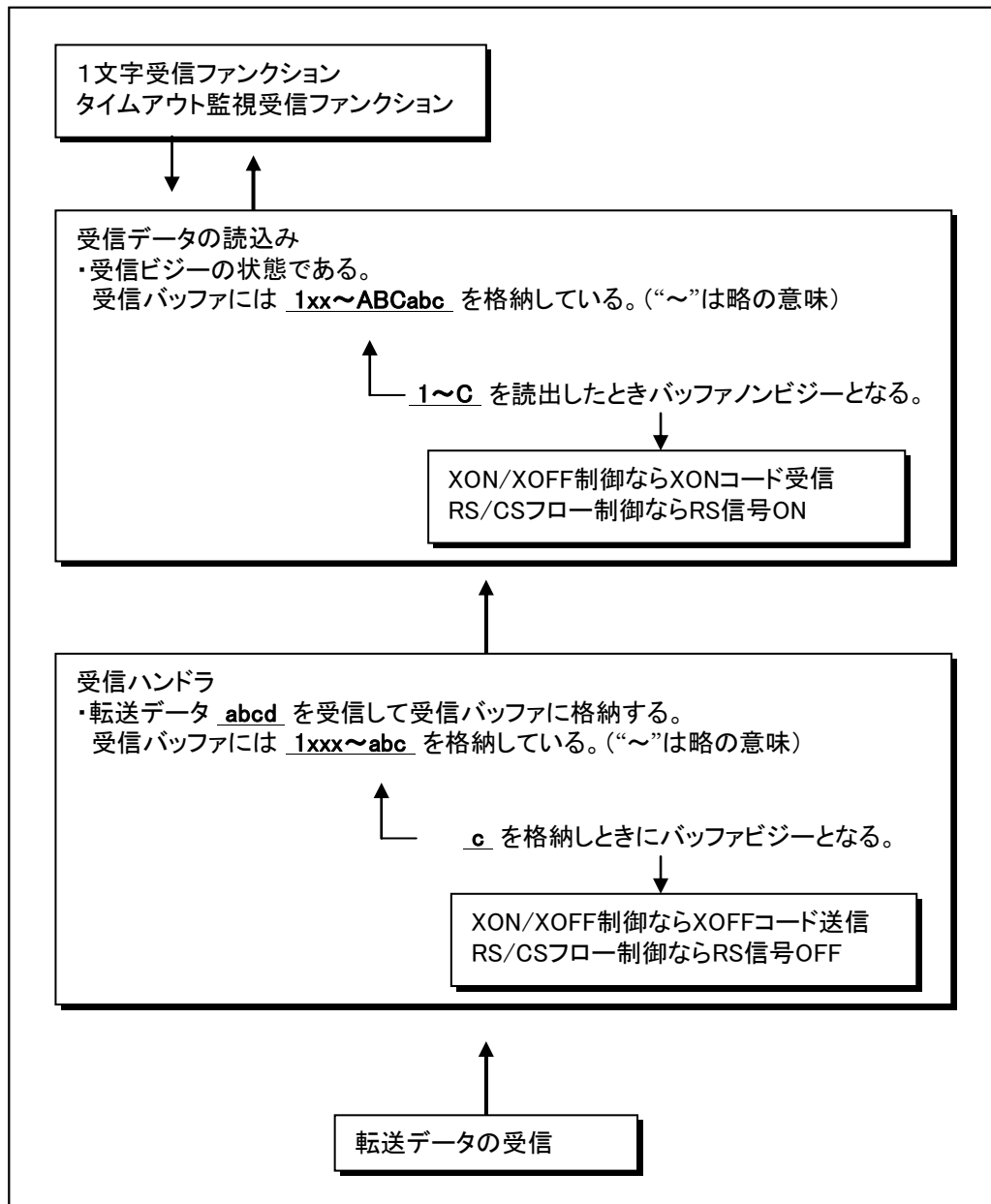
XON/XOFF 制御は受信バッファ全体の領域が67バイト以上であるとき機能します。

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。

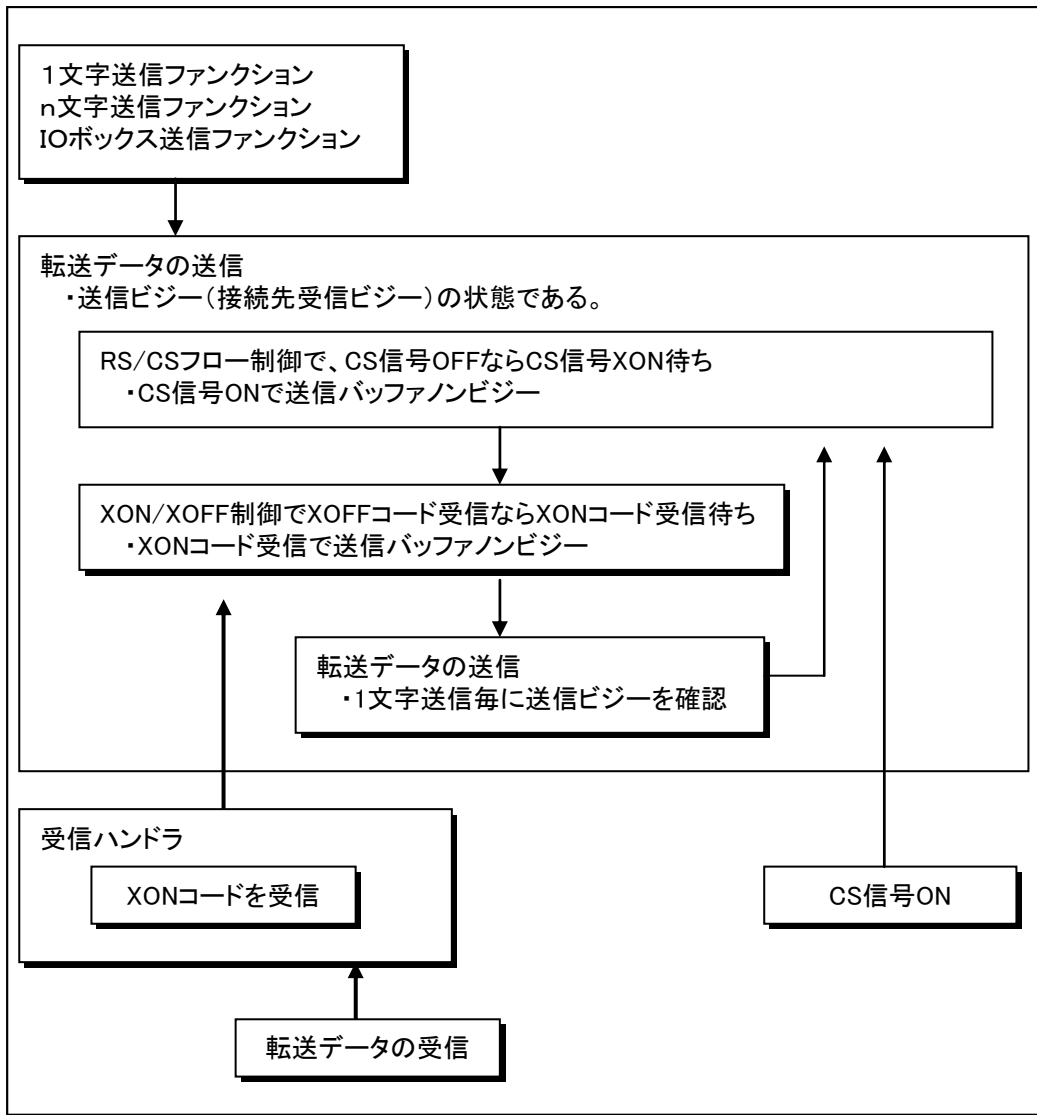
受信バッファに格納されている受信データが32バイト未満になるとバッファノンビジーとなり、RS

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であれば行い、CS信号がOFFであれば行わず、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下さい。
 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信号の監視を行うように指定して下さい。

【受信フロー制御の流れ】



【送信フロー制御の流れ】



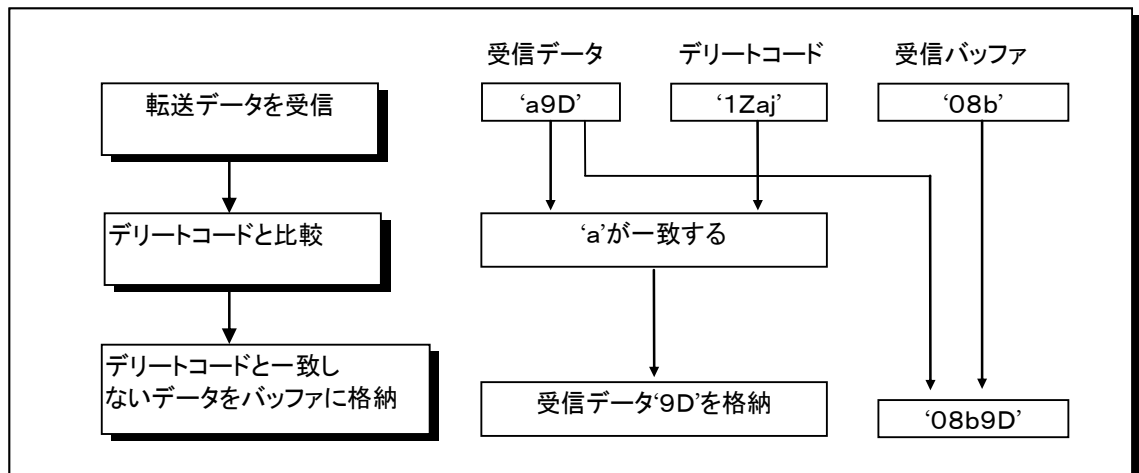
6.2.5. デリートコード制御

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います。

【デリートコードの制御の流れ】



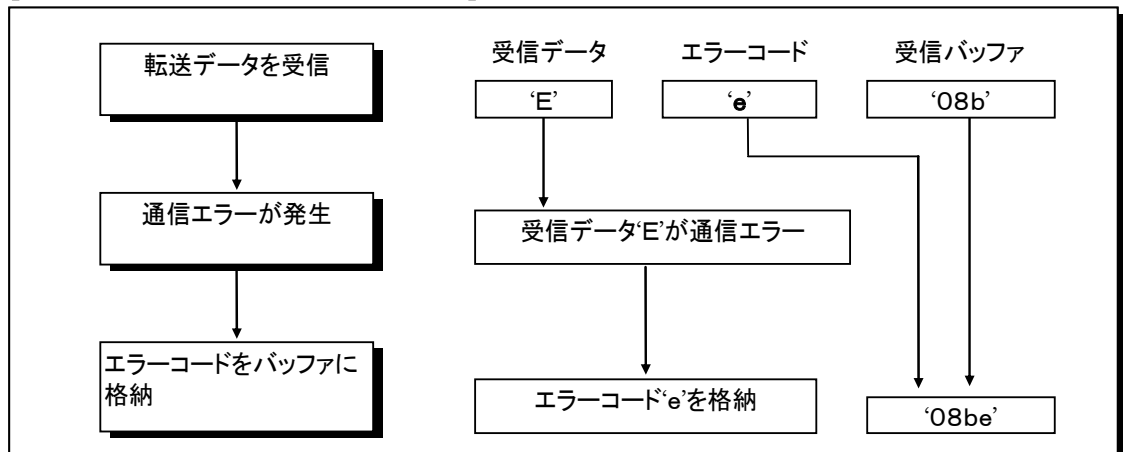
6.2.6. エラーコードバッファリング制御

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います。

【エラーコードバッファリング制御の流れ】



6.2.7. 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。

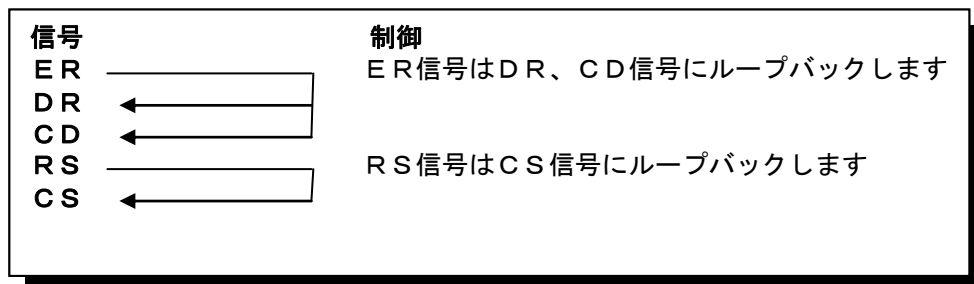
【通信ポートが制御する信号線】

信号線	通信ポート	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

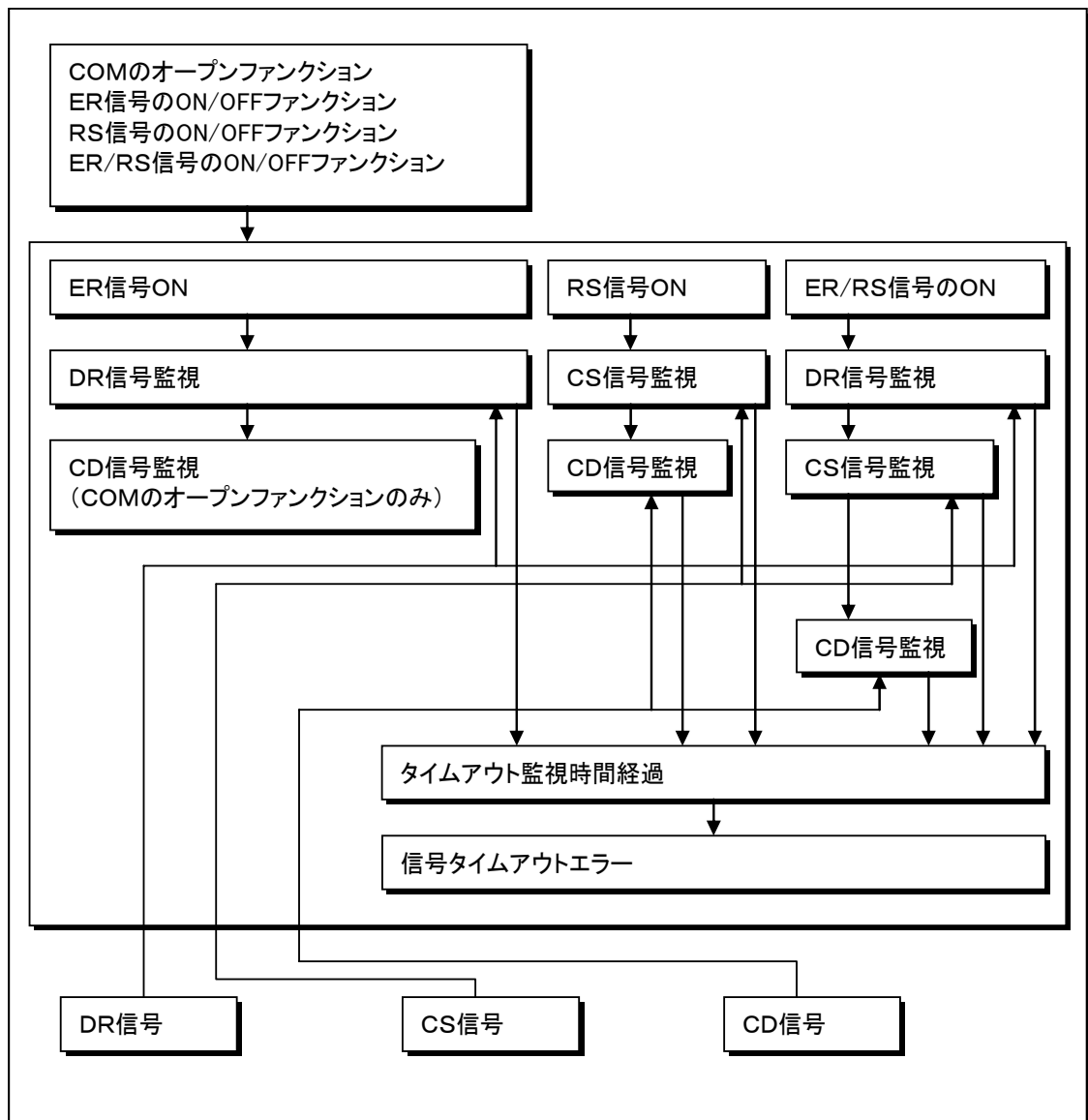
通信ポートのオープン、転送データの送信、受信データの読み込みおよびER/RS信号のONを行うとき、DR/CS/CD信号の ONまたはOFF状態の監視(遷移待ち)を行います。

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。

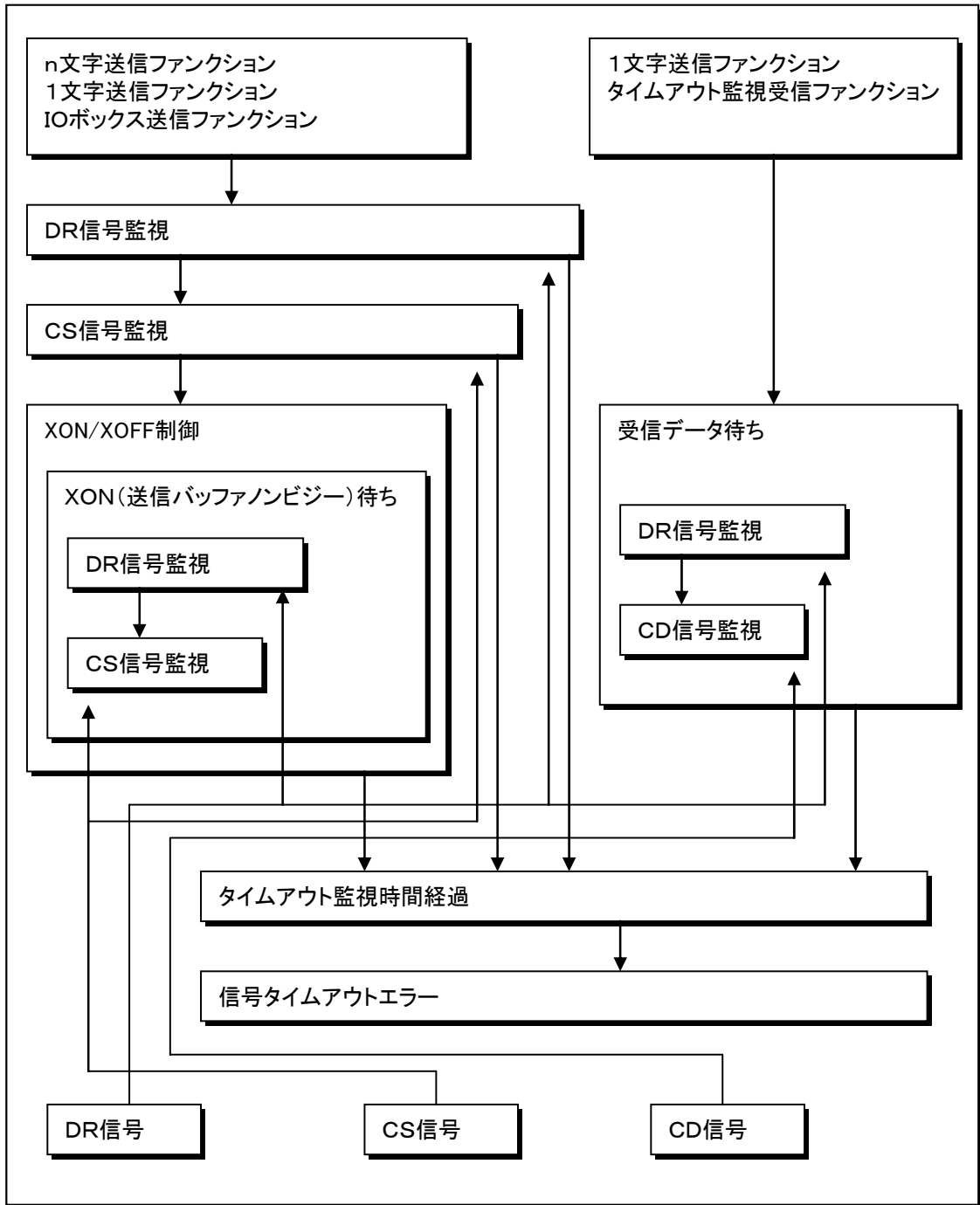
また、タイムアウト監視値の設定値により信号の監視を行わないようにすることができます。

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のタイムアウト監視の流れ】



6.2.8. 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ーの検出を行います。
 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の検出を行うには本機のシステムに対してLB検出を行うように設定する必要があります。

(1) LBの検出を行うファンクション

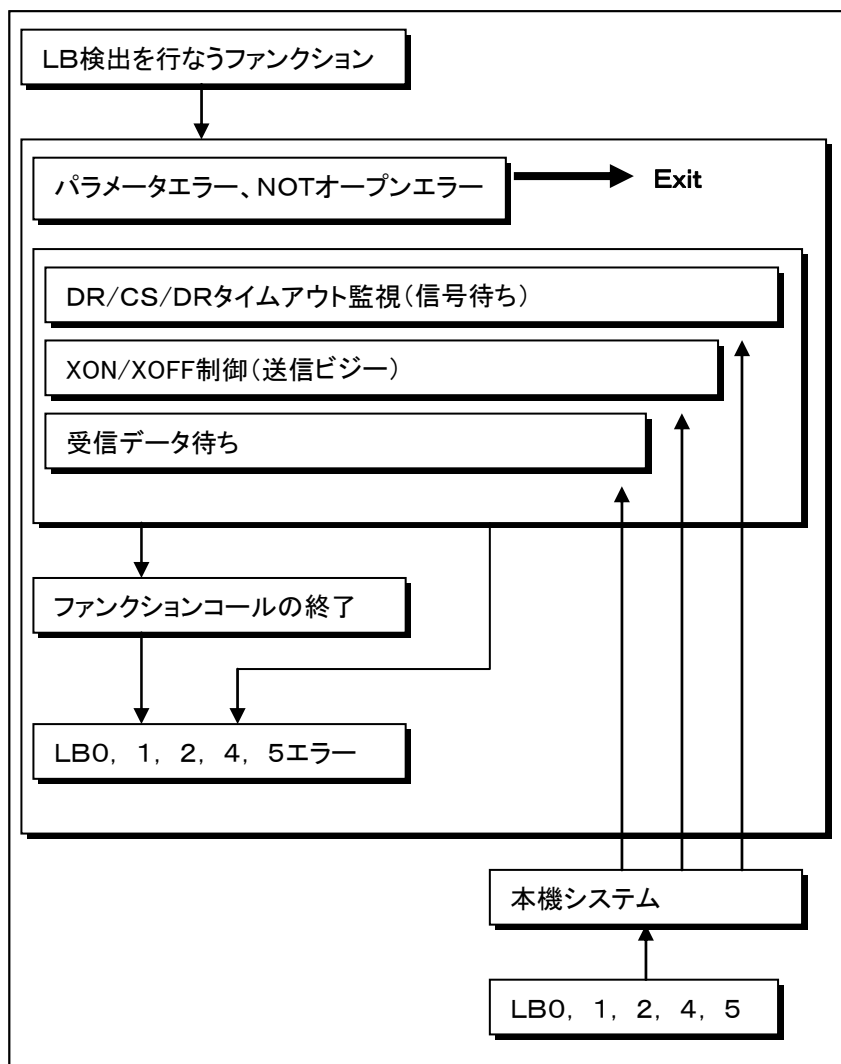
LBの検出はフロー制御、信号タイムアウト監視、受信データ待ちおよび各ファンクションコールの処理終了時に行います。

エラーステータスは「CERR_o_LBx」を通知します。

ローバッテリーには以下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】



6.2.9. 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。

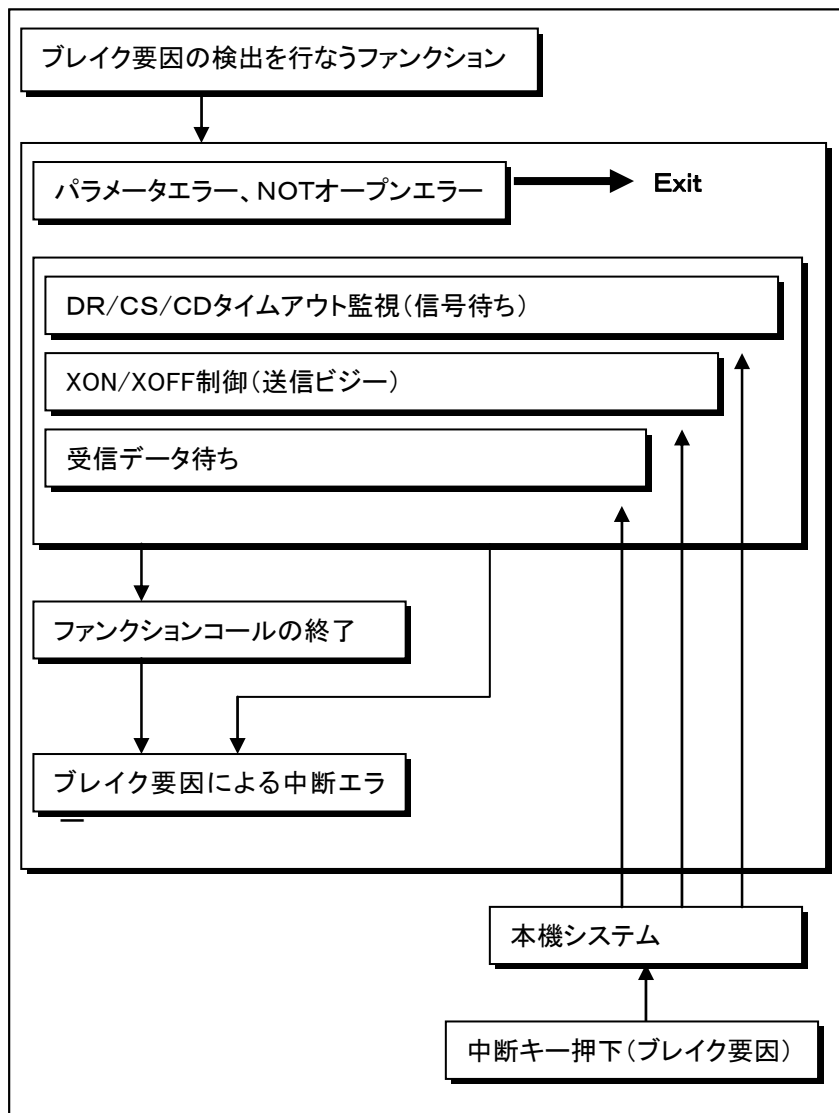
ブレイク要因の検出を行うには本機のシステムに対してブレイク要因の検出を行うように設定する必要があります。ブレイク要因の検出は「ブレイク要因の設定」ファンクションで設定することができます。

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います。ブレイク要因の検出を行うファンクションについては「6.4 エラー詳細」で各ファンクションのエラーステータスを参照して下さい。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】



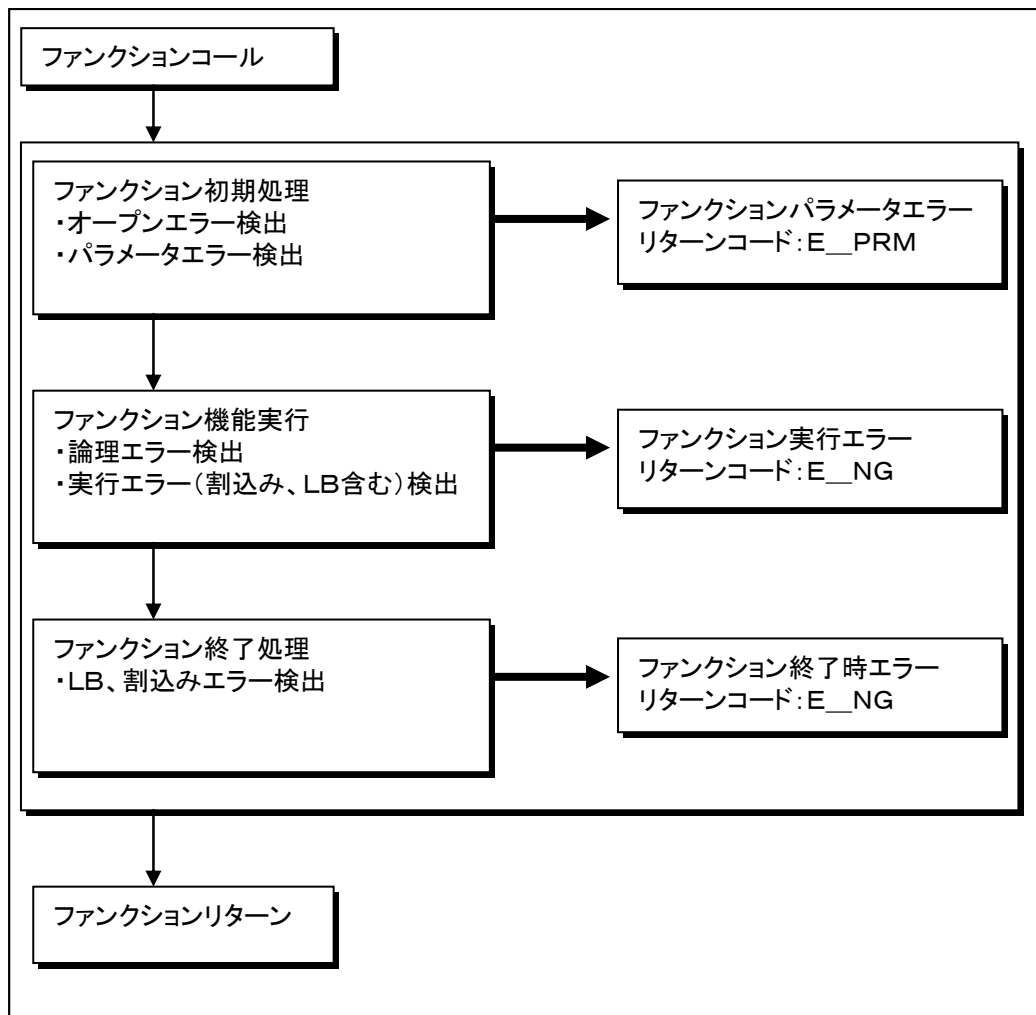
6.3. エラー詳細

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。

6.3.1. 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います。
ファンクションのリー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下さい。

【ファンクションのエラー検出の流れ】



6.3.2. 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_BUFFFUL	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) ローバッテリー(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 <ul style="list-style-type: none"> ・「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 <ul style="list-style-type: none"> ・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 <ul style="list-style-type: none"> ・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 <ul style="list-style-type: none"> ・通信ポートの指定が不当
	CERR__f__PARAMETER	パラメータエラー <ul style="list-style-type: none"> ・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 ・ボーレイトの指定が不当 ・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有しているため、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) ファンクション終了時エラー	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) ファンクション終了時エラー	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레이크送しの 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레이크ON/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레이크送しのON/OFF」 ファンクションでブ레이크 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레이크要因による中断 ・各信号タイムアウト監視中にブ레이크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 ・送信レンジが範囲外

(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

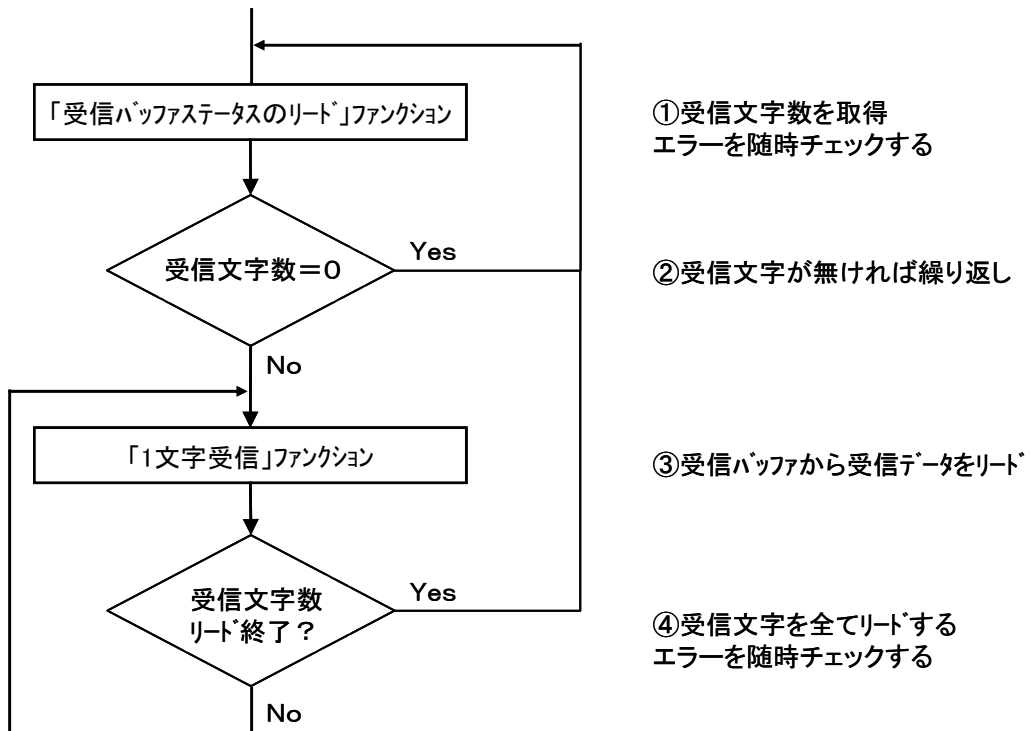
エラーコード	エラーステータス	要因
E_PRM	なし	パラメータエラー ・ブレイク要因通知の指定が不当 ・ファンクションキーの指定が不当

6.4. 通信関数 補足

通信関数が提供する機能について補足します。

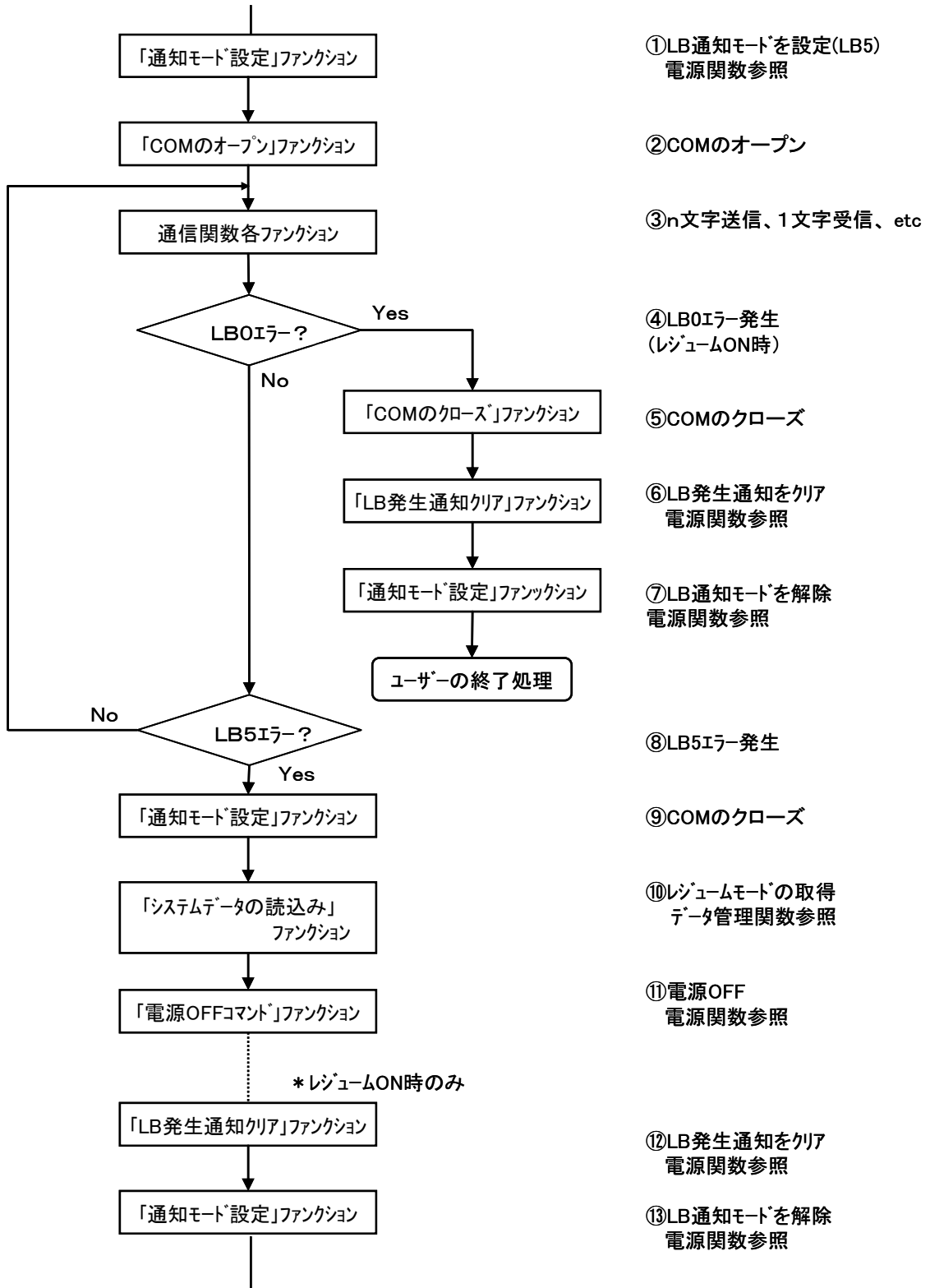
6.4.1. 受信データの読み込み

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み込む例を示します。



6.4.2. LBエラーチェック

ローバATTERY (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います。

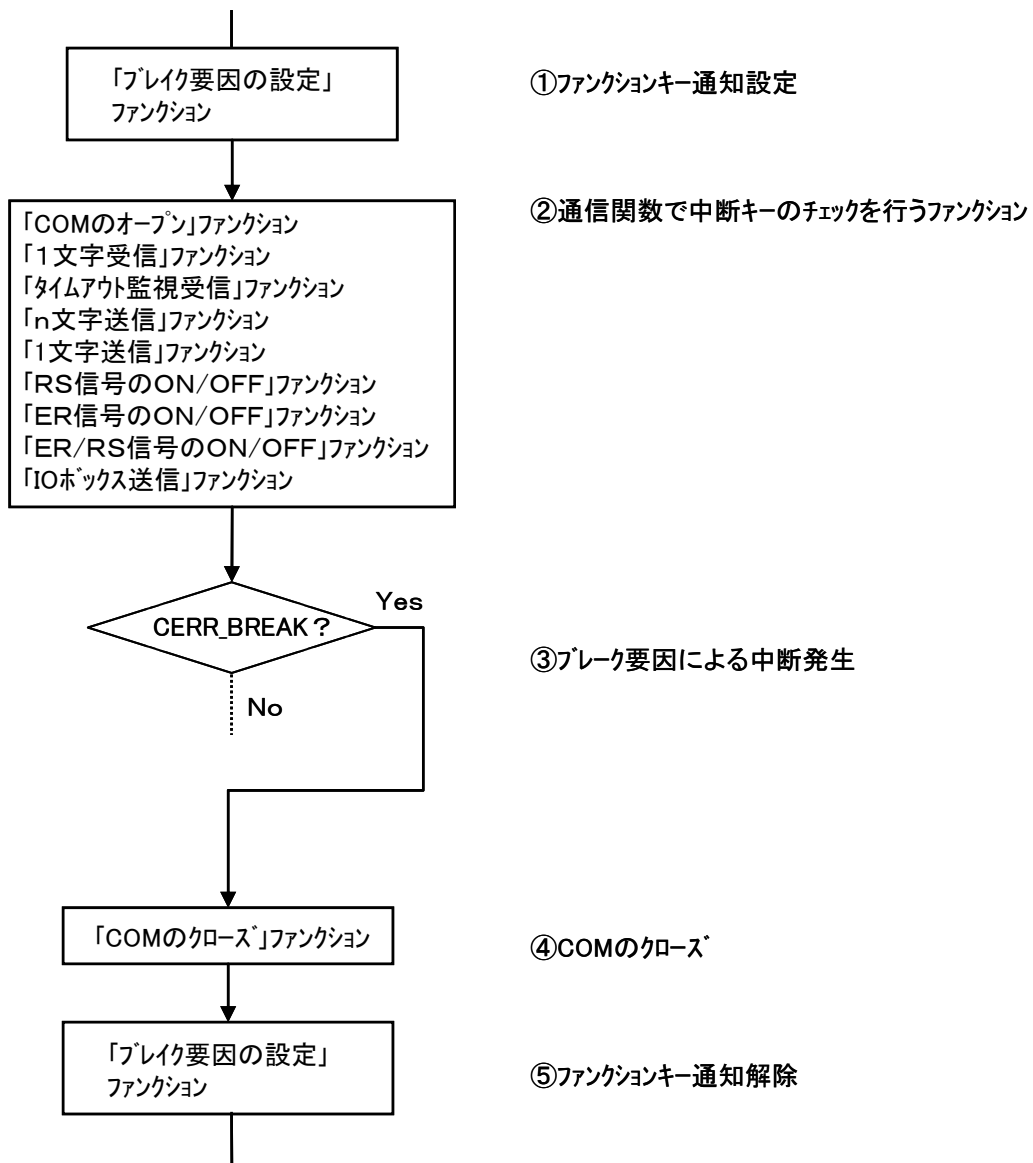


*LBエラー以外のチェック、処理は随時行うこと。

通信中にAPO (LB4) 発生の禁止は、「APO 禁止設定／解除」(電源関数参照)で行う必要があります。

6.4.3. 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、処理は随時行う必要があります。

6.4.4. 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う必要があります。以下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

半二重制御を行う場合、以下の2つ点を配慮しなくてはなりません。

a) CTRL信号の切替え時のターンアラウンドタイム

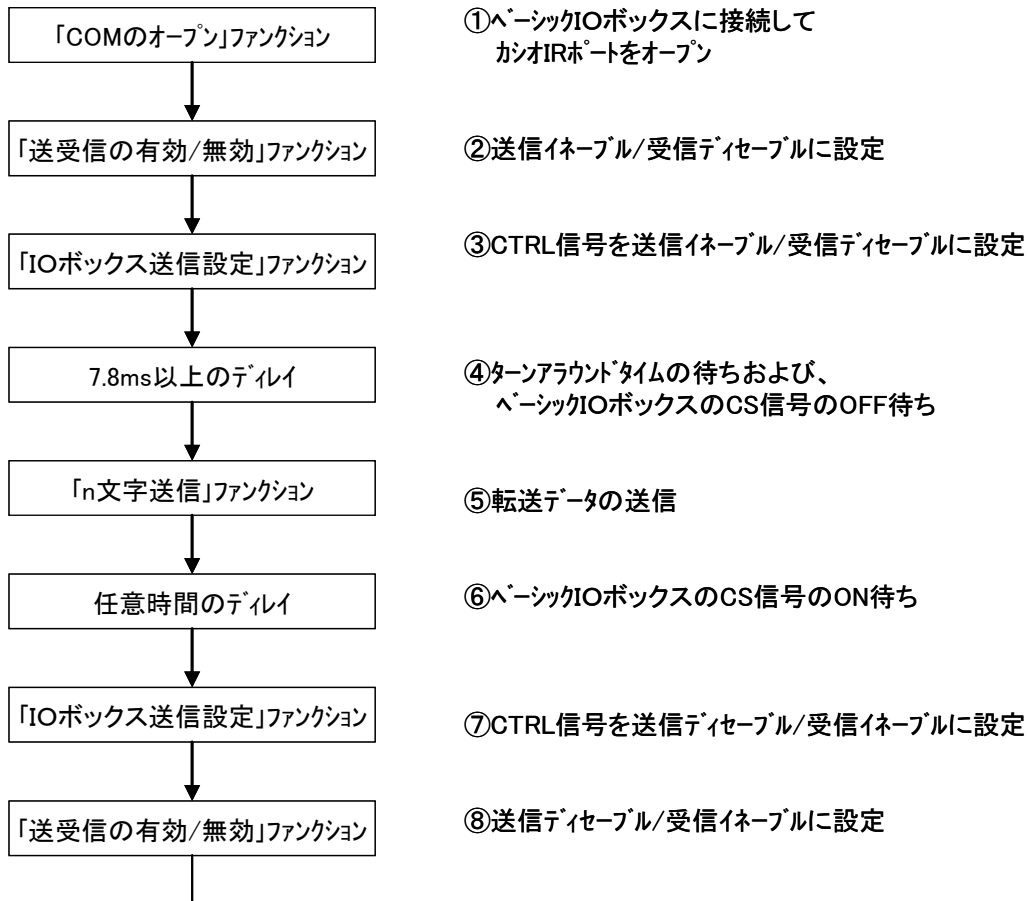
CTRL信号を切替えてデータ転送を開始するまでの間に7.8ms以上のターンアラウンドタイムが必要です。

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

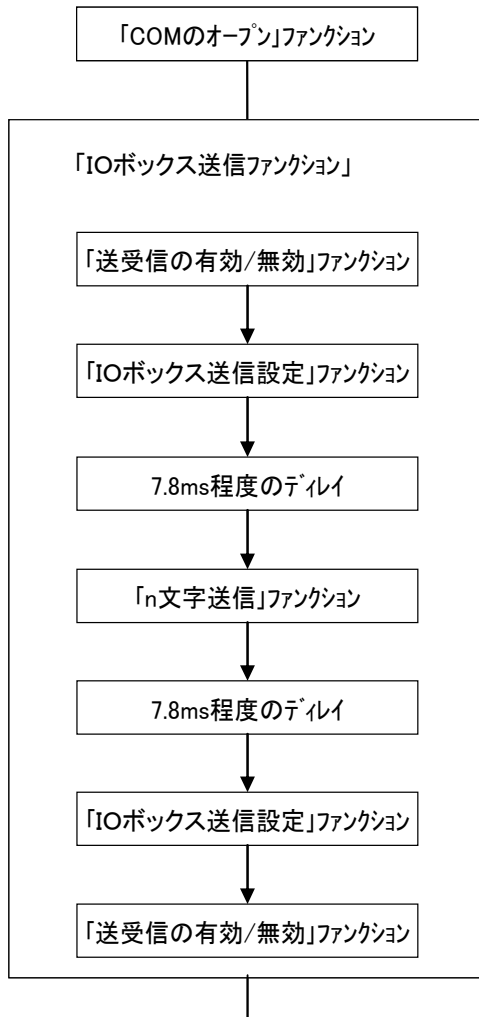
本機が転送データの送信を行うときCTRL信号が送信イネーブル、ベーシックIOボックスのCS信号はOFFでなければなりません。また、本機が転送データの受信を行うときCTRL信号が送信ディセーブル、ベーシックIOボックスのCS信号はOFFでなければなりません。CTRL信号が送信ディセーブル、ベーシックIOボックスのCS信号がOFFであるとき本機にフレーミングエラー(ノイズが発生する)が発生する場合があります。

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】



【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

先に述べた通り、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。

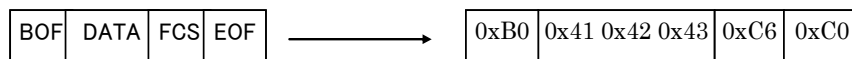
しかし、前述の“(1)ベーシックIOボックスとの接続”に示す例のようにソフトウェアでCTRL信号とベーシックIOボックスのCS信号が共にOFFにならない状態を作り出すことは容易でないと考えられます。従ってカシオIRポートで無手受信のデータ通信を行うことは不向きであり、データ通信を行うユーザエンティティの間で何等かの通信手順が必要になります。

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:フレームヘッダ
 DATA:転送データ
 FCS:フレームチェックシケンス
 EOF:フレームフッタ

	CTRL	CS	転送データ	受信バッファ	エラー	ファンクションコール
①	ON	OFF		<input type="text"/>		
②	OFF	OFF	0xFF	<input type="text"/>	フレーミングエラー	IOボックス送信設定
* CTRLとCSがOFFとなりフレーミングエラーとなった。エラー(ノイズ)のデータは破棄されます。						
③	OFF	ON	0xB0 0x41 0x42	<input type="text" value="0xB0 0x41 0x42"/>		
④	OFF	ON		<input type="text" value="0xB0 0x41 0x42"/>		1文字受信
⑤	OFF	ON		<input type="text" value="0xB0 0x41 0x42"/>		エラーステータスのリード
* エラーステータスがCERR_f_FRAMING or CERR_f_FRAMING2。読出したデータ'0xB0'は正常なデータです。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(ヘッダ)であるのでエラーはフレーム本体には影響を与えない。受信データの読み込みを継続します。						
⑥	OFF	ON		<input type="text" value="0x41 0x42"/>		1文字受信
⑦	OFF	ON		<input type="text" value="0x42"/>		1文字受信

次ページへ続く

	CTRL	CS	転送データ	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	フレーミング*	
* CTRLとCSがOFFとなりフレーミングエラーとなった。エラーのデ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ド
* エラーステータスがCERR_f_FRAMING。CERR_f_FRAMING2ではないので'0x42'はフレーミングエラーでない。受信データの読み込みを続けます。						
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'を読み出しフレームが完成したので受信データの読み込みはここで終了します。						
⑯	OFF	OFF				送受信の有効/無効
* 送信イネーブル/受信イネーブルに設定します。						
⑰	OFF	OFF				受信バッファのクリア
* 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
* 送信イネーブル/受信イネーブルに設定し、次は転送データの送信を行います。						

6.4.5. 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。

表6. 1 制限・注意事項一覧

項番	機能	内容
1	デリートコード制御	<ul style="list-style-type: none"> ・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下さい
2	XON/XOFF 制御	<ul style="list-style-type: none"> ・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下さい ・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しないで下さい ・通信関数内部受信バッファを使用する場合は使用しないで下さい
3	RS/CSフロー制御	<ul style="list-style-type: none"> ・カシオIRポートを使用する場合は使用しないで下さい ・通信関数内部受信バッファを使用する場合は使用しないで下さい
4	SI/SO制御	<ul style="list-style-type: none"> ・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下さい ・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	<ul style="list-style-type: none"> ・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下さい ・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	<ul style="list-style-type: none"> ・LBO, 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。従って検出を行う場合はシステムに対してイベントセットを行うように設定して下さい
7	CERR_r_xxxx2 エラーステータス	<ul style="list-style-type: none"> ・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读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ときに读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	<ul style="list-style-type: none"> ・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とができません ・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	<ul style="list-style-type: none"> ・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った場合、受信ビジーは解除しません

6.5. ファンクション詳細

ファンクション詳細を次ページより示します。

6.5.1. 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)の状態オープンします。

本ファンクションは以下の処理および設定を行います。

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

```
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

関数結果

E_OK	正常終了
E_NG	異常終了
E_PRM	パラメータエラー

補足

- ・「DR/CS/CDタイムアウト監視値の設定」により信号線の監視を行います。
- ・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わないで下さい。
- ・ER信号OFF、RS信号ONに設定したときCS信号ON、CD信号OFFのタイム監視はCSタイムアウト監視値で行います。
- ・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。

6.5.2. 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うことはできません。

本ファンクションは以下の処理を行います。

- 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

補足

6.5.3. 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以下のものがあります。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) のオン/オフ
- ・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

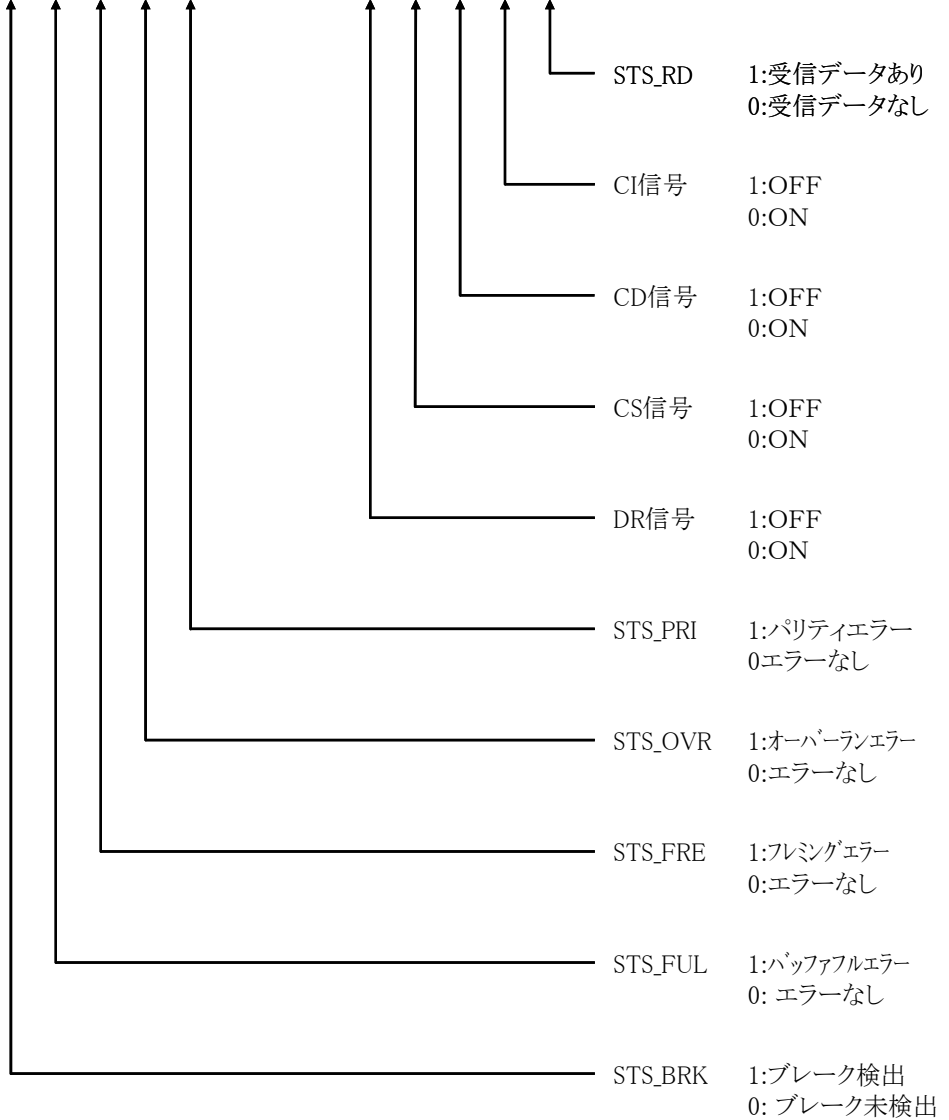
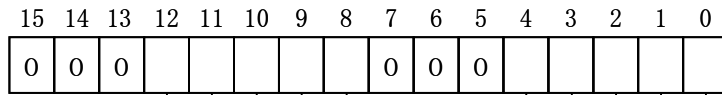
戻り値

関数結果

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)

補足

COMステータス



6.5.4. 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

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

6.5.5. 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することができます。

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。)

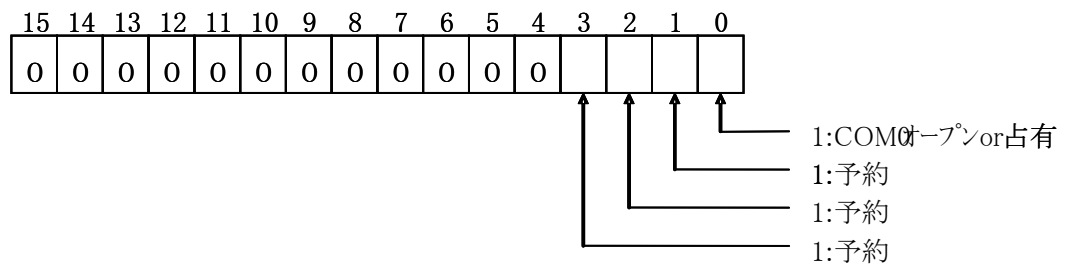
```
ER c_chkopen(void);
```

パラメータ

なし

戻り値

関数結果(オープン or 占有通知)



補足

6.5.6. 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

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

- ・「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。

6.5.7. 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

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

- ・「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。

6.5.8. c_tmdin

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

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

パラメータ

com_no

通信ポート

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

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

- ・「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。

6.5.9. 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

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

- ・「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。

6.5.10. 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

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

6.5.11. c_trxr

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_trxr(  
    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

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

6.5.12. 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。

本機から送信するときは送信イネーブル、本機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(送信イネーブル/受信ディセーブル)

C_IOBOXDSB

送信を解除(送信ディセーブル/受信イネーブル)

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

6.5.13. 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

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

- ・「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。

6.5.14. c_flush

受信バッファ内と通信コントローラのレジスタ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

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

6.5.15. 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以下のものがあります。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み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(次読み出し文字)
- 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

【ストラク構造】

```
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 ;
```

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

6.5.16. c_errbfiring

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_errbfiring(
    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

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

6.5.17. c_rderrsts

エラーステータスを読み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み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

関数結果

E_OK	正常終了
E_PRM	パラメータエラー

補足

エラーステータス詳細は、「6. 4. 2 エラー詳細」を参照して下さい。

6.5.18. 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、以下の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

関数結果

E_OK	正常終了
E_PRM	パラメータエラー

補足

6.5.19. 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であるときは監視は行いません。

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

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

6.5.20. 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

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

- ・「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。

6.5.21. 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

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

- ・「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。

6.5.22. 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

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

- ・「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。

6.5.23. 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

補足

- ・「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。

7. IrDA 部関数

7.1. 機能

7.1.1. 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。

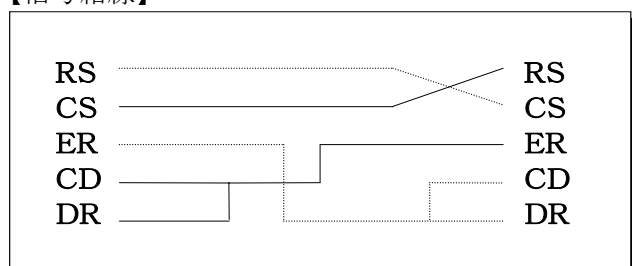
FIRモードは、USB IO ボックスで使用可能なモードです。本章で説明する Ir_xxx 関数では動作しませんので、FIRを使用する際は必ず、Ir モード設定関数により FIR に切替後、通信ユーティリティ関数 cu_xxx を使用してください。

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。

(1)信号結線

IrDAポートを使用して本機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】



7.1.2. 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を行いオープンとなります。

本機の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いません。

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) データ読み込み

受信データの読み込みを行います。

ユーザ定義のエリアに受信バッファデータの読み込みを行い、読込んだバイトサイズを返します。受信バッファデータが無くなるか、ユーザ定義のバッファサイズがフルになるまで読み込みが可能です。

受信バッファが空でもデータ待ち時間が指定されている場合はデータ待ちとなります。

このときLBエラー、タイムアウト、ブレイクイベントのチェックおよび、パリティ、オーバーラン、フレーミングエラーのチェックを行い、エラー時は直ちに異常終了となります。

データ待ちからは、受信バッファから1バイト以上のデータの読み込みが行え、かつ受信バッファに受信データが無くなればユーザ定義のバッファサイズに満たない場合でも終了となります。

また、受信データがある場合でも読み込み後にLBエラー、ブレイクイベントのチェックおよび、パリティ、オーバーラン、フレーミングエラーのチェックを行いエラー時は直ちに異常終了となります。このため受信データの読み込みが正常に行われていても異常終了となる場合があります。

データ待ち時間の指定は、通信状態設定関数(Ir_State_Set)で行うことができます。

尚、本関数では受信バッファに受信データが存在するとき、コネク特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、受信バッファの全てのデータ読み込みが終了したとき、に異常終了となりますが、受信データはユーザ定義のエリアへ格納されています。

また、相手局からのコネク特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特切断を待ちを行い、必要に応じて(受信待ちタイムアウトになった場合等)IrCOMMクローズを行うようにして下さい。

(4) データ書き込み

送信データの書き込みを行います。

ユーザ定義のエリアから送信バッファに送信データの書き込みを行い、書込んだバイトサイズを返します。

送信バッファに送信データの書き込みが行えなくなるか(バッファビジー)、ユーザー定義のバイトサイズまで書き込みを行います。

データ待ち時間が指定されていれば送信バッファに書き込みが行えないときデータ待ちとなり、一度データ待ちとなると全てのデータの書き込みが終了するまでの間をデータ待ち時間としてタイマによる監視を行います。

データ待ちの間およびデータ書き込み後にLBエラー、ブレイクイベント、タイムアウトのチェックを行い、エラー時は直ちに異常終了となります。

このため送信データの書き込みが正常に行われていても異常終了となる場合があります。

データ待ち時間の指定は、通信状態設定関数(Ir_State_Set)で行うことができます。

(5)送信データ数問合せ

送信バッファに残っている未送出のデータ数を問合せます。結果をバイトサイズで返します。
IrDAプロトコル上では送信バッファに書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み可能なデータ数を問合せます。結果をバイトサイズで返します。

(7)ER ON

ER信号をONにします。IrDAによる信号線のエミュレートとなります。
IrDAプロトコル規定のER信号ONを指示するデータフレームを相手局に送信します。
このため、「データ書込み」機能と同様に送信バッファへの書込みを行います。
データ待ちの間およびデータ書込み後にLBエラー、ブレイクイベント、タイムアウトのチェックを行い、エラー時は直ちに異常終了となります。
データ待ち時間の指定は、通信状態設定関数(Ir_State_Set)で行うことができます。

(8)ER OFF

ER信号をOFFにします。IrDAによる信号線のエミュレートとなります。
IrDAプロトコル規定のER信号OFFを指示するデータフレームを相手局に送信します。
このため、「データ書込み」機能と同様に送信バッファへの書込みを行います。
データ待ちの間およびデータ書込み後にLBエラー、ブレイクイベント、タイムアウトのチェックを行い、エラー時は直ちに異常終了となります。
データ待ち時間の指定は、通信状態設定関数(Ir_State_Set)で行うことができます。

(9)RS ON

RS信号をONにします。IrDAによる信号線のエミュレートとなります。
IrDAプロトコル規定のRS信号ONを指示するデータフレームを相手局に送信します。
このため、「データ書込み」機能と同様に送信バッファへの書込みを行います。
データ待ちの間およびデータ書込み後にLBエラー、ブレイクイベント、タイムアウトのチェックを行い、エラー時は直ちに異常終了となります。
データ待ち時間の指定は、通信状態設定関数(Ir_State_Set)で行うことができます。

(10)RS OFF

RS信号をOFFにします。IrDAによる信号線のエミュレートとなります。
IrDAプロトコル規定のRS信号OFFを指示するデータフレームを相手局に送信します。
このため、「データ書込み」機能と同様に送信バッファへの書込みを行います。
データ待ちの間およびデータ書込み後にLBエラー、ブレイクイベント、タイムアウトのチェックを行い、エラー時は直ちに異常終了となります。
データ待ち時間の指定は、通信状態設定関数(Ir_State_Set)で行うことができます。

(11)BREAK ON

ブレイク信号を送出します。IrDAによる信号のエミュレートとなります。
IrDAプロトコル規定のブレイク信号送出を指示するデータフレームを相手局に送信します。
このため、「データ書込み」機能と同様に送信バッファへの書込みを行います。
データ待ちの間およびデータ書込み後にLBエラー、ブレイクイベント、タイムアウトのチェックを行い、エラー時は直ちに異常終了となります。
データ待ち時間の指定は、通信状態設定関数(Ir_State_Set)で行うことができます。

(12)BREAK OFF

ブレイク信号の送出を停止します。IrDAによる信号のエミュレートとなります。
IrDAプロトコル規定のブレイク信号停止を指示するデータフレームを相手局に送信します。
このため、「データ書込み」機能と同様に送信バッファへの書込みを行います。
データ待ちの間およびデータ書込み後にLBエラー、ブレイクイベント、タイムアウトのチェックを行い、エラー時は直ちに異常終了となります。
データ待ち時間の指定は、通信状態設定関数(Ir_State_Set)で行うことができます。

(13)CD検査

CD信号のON/OFF状態をチェックし、通知します。信号のONまたはOFFの指定を行います。
信号待ち時間が指定されているとき指定した信号状態でなければ信号待ちとなります。
信号待ちとなったときLBエラー、タイムアウト、ブレイクイベントのチェックを行い、エラー時は直ちに異常終了となります。
データ待ち時間の指定は、通信状態設定関数(Ir_State_Set)で行うことができます。

(14)DR検査

DR信号のON/OFF状態をチェックし、通知します。
信号待ち時間が指定されているとき信号のON待ちとなります。
信号待ちとなったとき、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次局であるとき接続する相手局は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接続します。自局が2次局であるとき接続する相手局は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の機能は以下の通りです。

- 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接続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。

データ待ちには以下の状態があり、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みが行えないとき

- 受信バッファに読み込み可能なデータが無いとき

DR/CS/CD待ち時間は秒単位、待ちなし、無限待ちを指定することができます。

信号待ちには以下の状態があり、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
- ウィンドウサイズ
- 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。

7.1.3. 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	<ul style="list-style-type: none"> • プライオリティ 4 以下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		<ul style="list-style-type: none"> • 予約とします
3	Ir_Err_Get	<ul style="list-style-type: none"> • プライオリティ 2 以下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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • オープン状態である必要があります

7.2. ファンクション詳細

ファンクション詳細を次ページより示します。

7.2.1. 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

(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)

関数結果

E_IROK 正常終了

E_IRNG 異常終了

補足

7.2.2. 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
- ・コネクタ切断

```
H Ir_Close(void);
```

パラメータ

なし

戻り値

(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)

関数結果

E_IROK	正常終了
E_IRNG	異常終了

補足

7.2.3. Ir_Read

受信データの読み込みを行います。

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

```
H Ir_Read(  
    B *buff,  
    UH ReadSize,  
    UH *GetSize  
);
```

パラメータ

buff

受信データを格納するバッファのポインタ

ReadSize

受信データを格納するバッファのサイズ(バイト数)

GetSize

読み込みデータ数(受信バッファから読み込みできたバイト数)

戻り値

(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)

関数結果

E_IROK	正常終了
E_IRNG	異常終了 状態によってはデータの読み込みが行われています

補足

7.2.4. Ir_Write

送信データの書込みを行います。

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

```
H Ir_Write(  
    B *buff,  
    UH WriteSize,  
    UH *PutSize  
);
```

パラメータ

buff

送信データを格納するバッファのポインタ

WriteSize

送信データ数(送信バッファに書込むバイト数)

PutSize

書込みデータ数(送信バッファに書込みできたバイト数)

戻り値

(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)

関数結果

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みが行われています

補足

7.2.5. 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

(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)

関数結果

E_IROK	正常終了
E_IRNG	異常終了

補足

7.2.6. Ir_QueryRx

受信バッファより読み可能なデータ数を問合せます。

- 受信バッファ内の読み可能なデータ数の通知
- LBチェック
- ブレイクイベントのチェック

```
H Ir_QueryRx(  
    H *RcvDataSize  
)
```

パラメータ

RcvDataSize

読み可能なデータ数(バイト)

戻り値

(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)

関数結果

E_IROK	正常終了
E_IRNG	異常終了

補足

7.2.7. 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
- 送信データの書込み(制御データ)
- データ書込み待ち(送信バッファへの書込みが終了するまで)

```
H Ir_EROn(void);
```

パラメータ

なし

戻り値

(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)

関数結果

E_IROK	正常終了
E_IRNG	異常終了

補足

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。

7.2.8. 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
- 送信データの書込み(制御データ)
- データ書込み待ち(送信バッファへの書込みが終了するまで)

```
H Ir_EROff(void);
```

パラメータ

なし

戻り値

(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)

関数結果

E_IROK	正常終了
E_IRNG	異常終了

補足

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。

7.2.9. 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
- 送信データの書込み(制御データ)
- データ書込み待ち(送信バッファへの書込みが終了するまで)

```
H Ir_RSON(void);
```

パラメータ

なし

戻り値

(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)

関数結果

E_IROK	正常終了
E_IRNG	異常終了

補足

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。

7.2.10. 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
- 送信データの書込み(制御データ)
- データ書込み待ち(送信バッファへの書込みが終了するまで)

```
H Ir_RSOff(void);
```

パラメータ

なし

戻り値

(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)

関数結果

E_IROK	正常終了
E_IRNG	異常終了

補足

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。

7.2.11. Ir_BreakOn

ブレイ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
- 送信データの書込み(制御データ)
- データ書込み待ち(送信バッファへの書込みが終了するまで)

```
H Ir_BreakOn(void);
```

パラメータ

なし

戻り値

(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)

関数結果

E_IROK	正常終了
E_IRNG	異常終了

補足

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。

7.2.12. Ir_BreakOff

ブレイ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
- 送信データの書込み(制御データ)
- データ書込み待ち(送信バッファへの書込みが終了するまで)

```
H Ir_BreakOff(void);
```

パラメータ

なし

戻り値

(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)

関数結果

E_IROK	正常終了
E_IRNG	異常終了

補足

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。

7.2.13. 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

(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)

関数結果

E_IRLINE_ON	信号ON
E_IRLINE_OFF	信号OFF
E_IRNG	異常終了

補足

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。

7.2.14. 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(void);
```

パラメータ

なし

戻り値

(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)

関数結果

E_IRLINE_ON	信号ON
E_IRLINE_OFF	信号OFF
E_IRNG	異常終了

補足

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。

7.2.15. 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の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(void);
```

パラメータ

なし

戻り値

(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)

関数結果

E_IRLINE_ON	信号ON
E_IRLINE_OFF	信号OFF
E_IRNG	異常終了

補足

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。

7.2.16. 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 (void);
```

パラメータ

なし

戻り値

(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)

関数結果

E_IRLINE_ON	信号ON
E_IRLINE_OFF	信号OFF
E_IRNG	異常終了

補足

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。

7.2.17. 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(void);
```

パラメータ

なし

戻り値

(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)

関数結果

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

補足

7.2.18. 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(void);
```

パラメータ

なし

戻り値

関数結果

詳細は、次ページを参照して下さい

補足

エラー発生要因

以下のフォーマットでエラー値について示します。

エラー値	エラーコード名称	
詳細	エラーの詳細	
関数名	IrCOMM 状態	主なエラー対処方法
エラーの発生する関数名	関数異常終了時のIrCOMM オープン状態	IrDA部の上位が行う発生したエラーに対する事後処理

エラー値	IRERR__NORESOURCE	
詳細	IrDA部内の資源不足によりLASP(コネクに必要な内部情報)が確保できないと発生します IRERR__DISCONNECTエラーの要因として一緒に通知します 通常このエラーが発生することはありえないのでダンプ等を行い原因の調査をする必要があります	
関数名	IrCOMM 状態	主なエラー対処方法
Ir_Open	オープンを行わずクローズ状態	・ダンプ等を行い、原因調査をする必要があります

エラー値	IRERR__NODEVICE	
詳細	回線上(空間)にコネク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関数でのコネク待ちタイムアウトの要因でもあります	
関数名	IrCOMM 状態	主なエラー対処方法
Ir_Open	オープンを行わずクローズ状態	・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関数でのコネク待ちタイムアウトの要因でもあります	
関数名	IrCOMM 状態	主なエラー対処方法
Ir_Open	オープンを行わずクローズ状態	・相手局のプロトコル実装を確認して再実行して下さい

エラー値	IRERR__LOCK	
詳細	通信用のデバイスおよびリソースが既にロックされているときに発生します 本機では通信関数とシステム資源を共有していますので先に起動された方に資源を利用する権利があります。このエラーが発生したときは資源が開放されるまで待たなければなりません また、OBRとの排他制御を行っていますのでOBR起動中にも当該エラーとなります	
関数名	IrCOMM 状態	主なエラー対処方法
Ir_Open	オープンを行わずクローズ状態	・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)がオープンしています。クローズしてから再実行して下さい

エラー値	IRERR__DISCONNECT	
詳細	コネクト手続き中またはコネクト後に相手局からの応答が無くなったとき、相手局からコネクト切断されたとき、レジュームON立上げを行ったときに発生します	
関数名	IrCOMM 状態	主なエラー対処方法
Ir_Open	オープンを行わずクローズ状態	<ul style="list-style-type: none"> 通信環境を確認して再実行して下さい 相手局と通信不可能な環境にあるのでその原因を取り除いて IrCOMM(赤外線ポート)のオープンを行って下さい 相手局から一定時間応答がない(回線が外れている)場合が考えられます
Ir_Close	クローズ状態となります	
Ir_Read		
Ir_Write		
Ir_EROn		
Ir_EROff		
Ir_RSON		
Ir_RSOff		
Ir_BreakOn		
Ir_BreakOff		
Ir_QueryTx		
Ir_QueryRx		
Ir_CheckCD		
Ir_CheckDR		
Ir_CheckCS		
Ir_CheckCI		
Ir_CheckBreak		
Ir_Init		

エラー値	IRERR__PARAMETER	
詳細	関数のパラメータの入力値に誤りがあるとき発生します	
関数名	IrCOMM 状態	主なエラー対処方法
Ir_Open	カレントの状態に変化はありません	<ul style="list-style-type: none"> 入力パラメータを確認して下さい
Ir_CheckCD		
Ir_State_set		
Ir_SetPortConfig		

エラー値	IRERR__BREAK_EVNT	
詳細	キー関数の機能を用いて中断キーのイベント登録が行なわれ、当該キー押下によりブレイクイベントの検出がIrDA部で行われたときに発生します	
関数名	IrCOMM 状態	主なエラー対処方法
Ir_Open	オープンを行わずクローズ状態	<ul style="list-style-type: none"> 再実行する場合は Ir_Open から行って下さい
Ir_Close	クローズ状態となります	
Ir_Read	オープン状態から変更はありません	<ul style="list-style-type: none"> 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	
関数名	IrCOMM 状態	主なエラー対処方法
詳細 電源関数の機能を用いてLB0の通知モードに設定されており、LB0エラー(主電池なし、電池蓋開き)と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		
Ir_Open	オープンを行わずクローズ状態	<ul style="list-style-type: none"> 再実行するときは Ir_Open を行って下さい イベントのクリアを行って下さい
Ir_Close	クローズ状態となります	
Ir_Read		
Ir_Write		
Ir_EROn		
Ir_EROff		
Ir_RSOOn		
Ir_RSOff		
Ir_BreakOn		
Ir_BreakOff		
Ir_CheckCD		
Ir_CheckDR		
Ir_CheckCS		
Ir_QueryTx		
Ir_QueryRx		
Ir_CheckCI		
Ir_CheckBreak		
Ir_Init		

エラー値	IRERR__LB1	
関数名	IrCOMM 状態	主なエラー対処方法
詳細 電源関数の機能を用いてLB1の通知モードに設定されており、LB1エラー(主電池電圧低下)となったときに発生します		
Ir_Open	オープンを行わずクローズ状態	<ul style="list-style-type: none"> 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
Ir_Close	クローズ状態となります	
Ir_Init		
Ir_Read	オープン状態から変更はありません	<ul style="list-style-type: none"> Ir_Close を行って終了して下さい 電池交換交換後に 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_RSOOn		
Ir_RSOff		
Ir_BreakOn		
Ir_BreakOff		
Ir_CheckCD		
Ir_CheckDR		
Ir_CheckCS		
Ir_QueryTx		
Ir_QueryRx		
Ir_CheckCI		
Ir_CheckBreak		

エラー値	IRERR__LB2	
詳細	電源関数の機能を用いてLB2の通知モードに設定されており、LB2エラー(副電池電圧なし)となったときに発生します	
関数名	IrCOMM 状態	主なエラー対処方法
Ir_Open	オープンを行わずクローズ状態	<ul style="list-style-type: none"> 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	クローズ状態となります	
Ir_Init		
Ir_Read	オープン状態から変更はありません	<ul style="list-style-type: none"> 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	電源関数の機能を用いてLB4の通知モードに設定されており、LB4エラー(APO 発生)となったときに発生します 通知モードに設定されているときは電源OFFしませんので、アプリケーションが責任を持つ必要があります	
関数名	IrCOMM 状態	主なエラー対処方法
Ir_Open	オープンを行わずクローズ状態	<ul style="list-style-type: none"> イベントのクリアを行って下さい 電源OFFして下さい
Ir_Close	クローズ状態となります	
Ir_Init		
Ir_Read	オープン状態から変化はありません	<ul style="list-style-type: none"> 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	
関数名	IrCOMM 状態	主なエラー対処方法
詳細 電源関数の機能を用いてLB5の通知モードに設定されており、LB5エラー(OFFキー押下による電源OFF)となったときに発生します 通知モード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		
Ir_Open	オープンを行わずクローズ状態	<ul style="list-style-type: none"> • イベントのクリアを行って下さい • 電源OFFして下さい
Ir_Close	クローズ状態となります	
Ir_Init		
Ir_Read	オープン状態から変化はありません	<ul style="list-style-type: none"> • 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_RSOOn		
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	
関数名	IrCOMM 状態	主なエラー対処方法
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		
Ir_Close	クローズ状態から変化はありません	<ul style="list-style-type: none"> • IrCOMM が既にクローズ状態になっています • Ir_Open を行ってから実行して下さい
Ir_Read		
Ir_Write		
Ir_QueryTX		
Ir_QueryRx		
Ir_EROn		
Ir_EROff		
Ir_RSOOn		
Ir_RSOff		
Ir_BreakOn		
Ir_BreakOff		
Ir_CheckCD		
Ir_CheckDR		
Ir_CheckCS		
Ir_CheckCI		
Ir_CheckBreak		
Ir_Init		

エラー値	IRERR__TIMEOUT	
詳細 Ir__Opne関数で指定したコネク待ち時間を経過した場合と、Ir__State__Set関数で指定したデータ待ち時間を経過すると発生します		
関数名	IrCOMM 状態	主なエラー対処方法
Ir_Open	クローズ状態です	・通信環境を確認して再実行して下さい
Ir_Read	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
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関数で以下の指定のとき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 <ul style="list-style-type: none"> ・3Wireまたは9Wire ・パリティあり 		
関数名	IrCOMM 状態	主なエラー対処方法
Ir_Read	オープン状態に変化はありません	・任意の処理を行って下さい

エラー値	IRERR__OVERRUN	
詳細 Ir__State__Set関数で以下の指定のとき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 <ul style="list-style-type: none"> ・3Wireまたは9Wire 		
関数名	IrCOMM 状態	主なエラー対処方法
Ir_Read	オープン状態に変化はありません	・任意の処理を行って下さい

エラー値	IRERR__FRAMING	
詳細 Ir__State__Set関数で以下の指定のとき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 <ul style="list-style-type: none"> ・3Wireまたは9Wire 		
関数名	IrCOMM 状態	主なエラー対処方法
Ir_Read	オープン状態に変化はありません	・任意の処理を行って下さい

エラー値	IRERR__WIRE__TYPE	
詳細 Ir_State_Set関数で3Wireあるいは9Wireが指定されている場合しか使用できない関数を使用したとき発生します		
関数名	IrCOMM 状態	主なエラー対処方法
Ir_EROn	オープン状態に変化はありません	・9Wire以外が指定されています。用途に合った指定を行って下さい
Ir_EROff		
Ir_RSON		
Ir_RSOFF		
Ir_BreakOn		
Ir_BreakOff		
Ir_CheckCD		
Ir_CheckDR		
Ir_CheckCS		
Ir_CheckCI		
Ir_CheckBreak	オープン状態に変化はありません	・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

7.2.19. Ir_State_Set

IrDA部の通信状態を設定します。

- Wireの指定
- データ読み込み／書き込み(データ待ち)時間の指定
- DR／CS／CD信号待ち時間の設定(9Wire指定時のみ有効)
- RS232Cの通信仕様の指定(3／9Wire指定時のみ有効)
- 局の指定

```
H Ir_State_Set(  
    struct *State_DCB  
);
```

パラメータ

State_DCB

【ストラク構造】

```
struct State_DCB {  
    H station;           : 局  
    H Wire;              : W i r e  
    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

(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)

関数結果

E_IROK	正常終了
E_IRNG	異常終了

補足

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	秒単位にデータ読み込み/書き込み待ち時間を設定	○(300)
	THROUGH	データ読み込み/書き込み待ちを行わない	
	FOREVER	タイマ指定なしでデータ読み込み/書き込み待ちを行う	
DR/CS/CD 信号待ち時間	1-600	秒単位に DR/CS/CD 信号待ち時間を設定	
	THROUGH	DR/CS/CD 信号のチェックを行わない	○
	FOREVER	タイマ指定なしで DR/CS/CD 信号待ちを行う	
RS232C の通信速度	BPS_12	RS232C の通信速度を 1200bps に設定	
	BPS_24	RS232C の通信速度を 2400bps に設定	
	BPS_48	RS232C の通信速度を 4800bps に設定	
	BPS_96	RS232C の通信速度を 9600bps に設定	○
	BPS_192	RS232C の通信速度を 19200bps に設定	
	BPS_384	RS232C の通信速度を 38400bps に設定	
	BPS_576	RS232C の通信速度を 57600bps に設定	
	BPS_1152	RS232C の通信速度を 115200bps に設定	
RS232C のデータ長	LEN_7B	RS232C のデータ長を 7bit に設定	
	LEN_8B	RS232C のデータ長を 8bit に設定	○
RS232C のストップビット	STOP_1B	RS232C のストップビットを 1bit に設定	○
	STOP_2B	RS232C のストップビットを 2bit に設定	
RS232C のパリティビット	PRI_ODD	RS232C のパリティビットを奇数パリティに設定	
	PRI_EVN	RS232C のパリティビットを偶数パリティに設定	
	PRI_NON	RS232C のパリティビットをパリティなしに設定	○

7.2.20. 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

【ストラク構造】

```
struct SetPortConfig_DCB {  
    UB irBaud;           : ボーレート  
    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;     : ウィンドウサイズ  
    UB BofCount;        : B O F 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

(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)

関数結果

E_IROK	正常終了
E_IRNG	異常終了

補足

- 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 ボーレート ・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ドタイム	TURN_500MS	最大ターンアラウンドタイムを 500ms に設定	○
フレームサイズ	FRAME_1024B	フレームサイズを 1024byte に設定	○
ウィンドウサイズ	WINDOW_4	ウィンドウサイズを 4 フレームウィンドウに設定	○
BOF 数 ・IR ボーレートにより比例して増減します BOF 数は 115.2k の場合です	BOF_48	BOF を 48 個追加	○
	BOF_24	BOF を 24 個追加	
	BOF_12	BOF を 12 個追加	
	BOF_5	BOF を 5 個追加	
	BOF_3	BOF を 3 個追加	
	BOF_2	BOF を 2 個追加	
	BOF_1	BOF を 1 個追加	
	BOF_0	BOF を 0 個追加	
最小ターンアラウンドタイム	TURN_5MS	最小ターンアラウンドタイムを 5ms に設定	
	TURN_1MS	最小ターンアラウンドタイムを 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

7.2.21. 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(void);
```

パラメータ

なし

戻り値

(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)

関数結果

E_IROK	正常終了
E_IRNG	異常終了

補足

7.2.22. 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

関数結果

E_OK 正常終了

E_NG 異常終了

補足

8. Bluetooth 通信部

8.1. 通信仕様

8.1.1. 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

8.1.2. 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

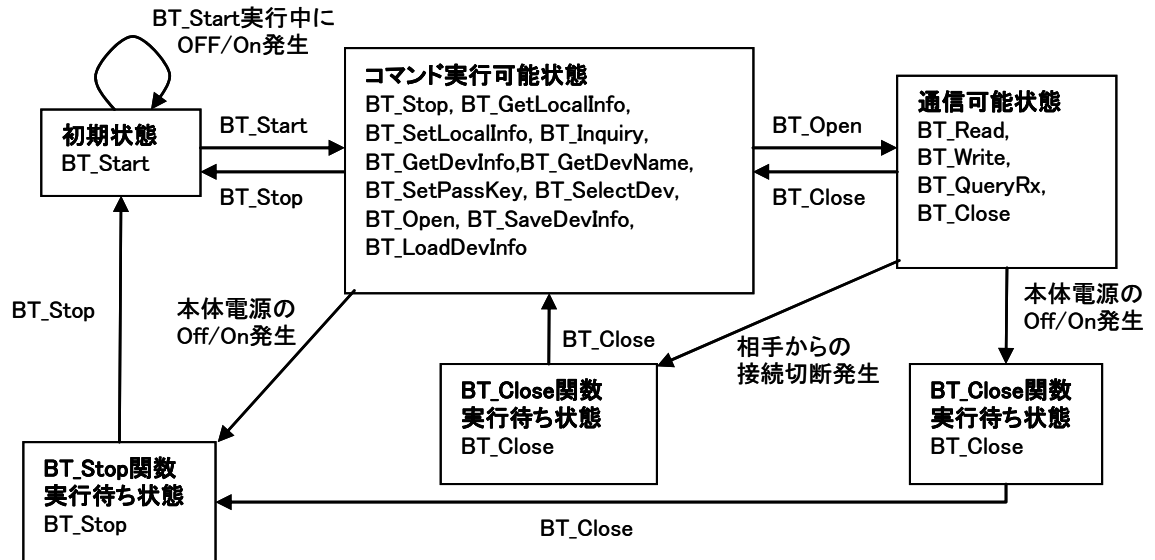
- Bluetooth 機器の Inquiry (問い合わせ) を実行
- Inquiry で発見された Bluetooth 機器の情報を取得
- 取得した Bluetooth 機器情報をファイルに保存

(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

8.2. 機能

8.2.1. 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。

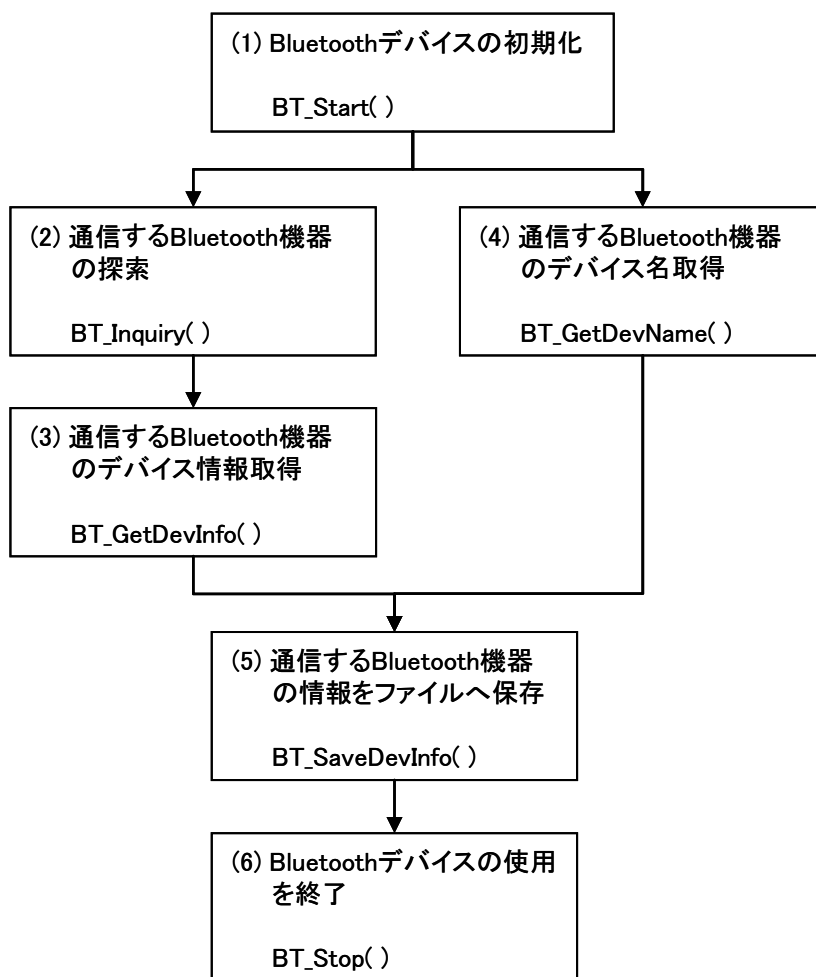
(5) 通信可能状態において通信相手から Bluetooth 接続が切断されたとき

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。

8.2.2. 関数の実行手順

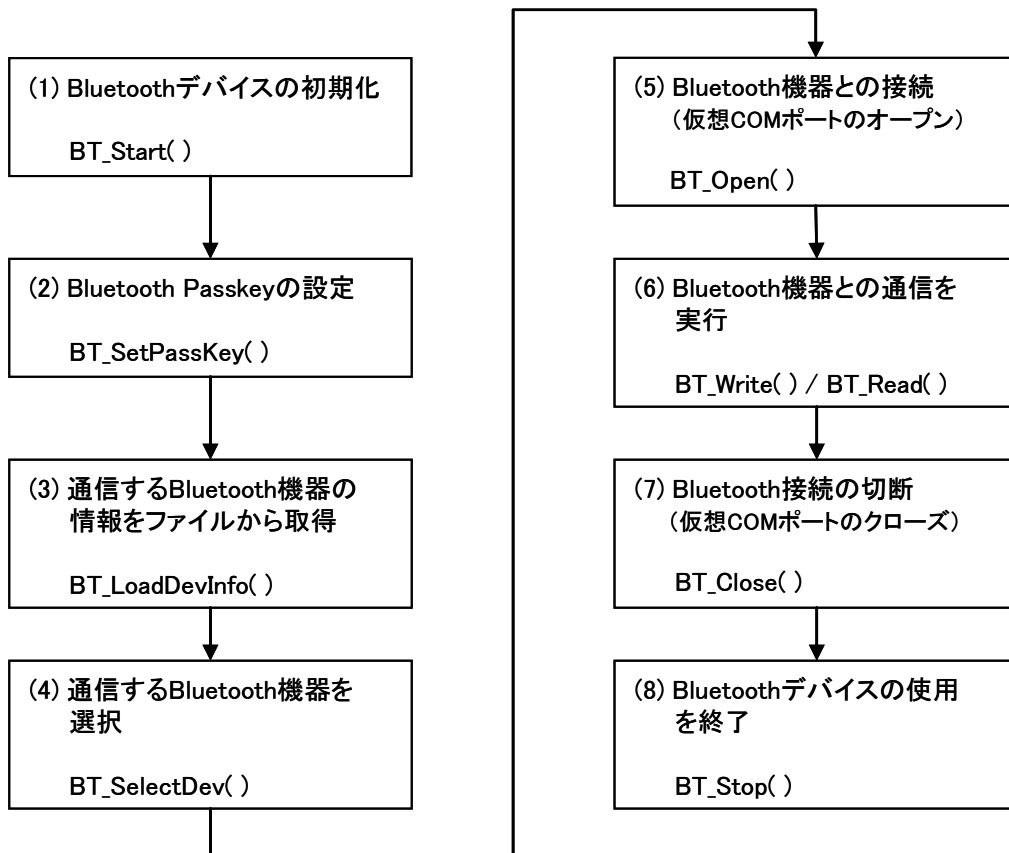
(1) Bluetooth 機器情報の取得と保存



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。

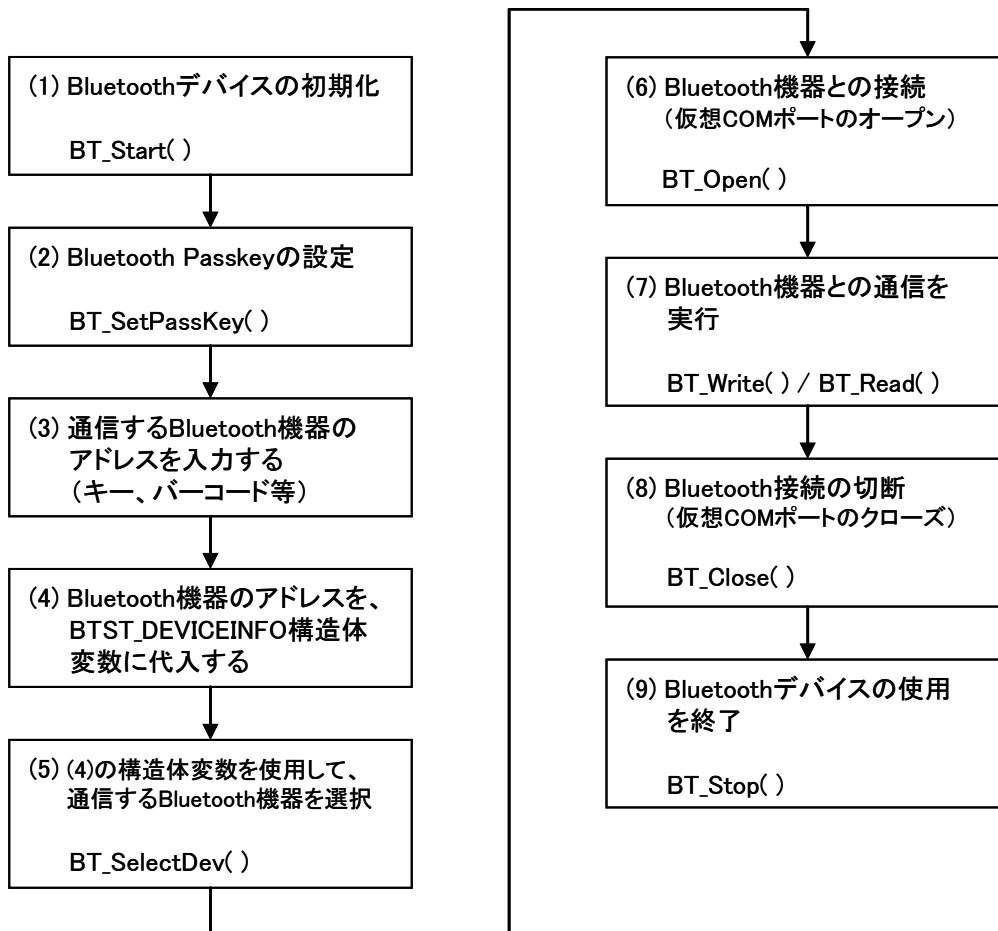
(2) Bluetooth 接続と通信

(1)の手順を実行後、以下の手順で 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、以下の手順で 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へ出力する場合、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以下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

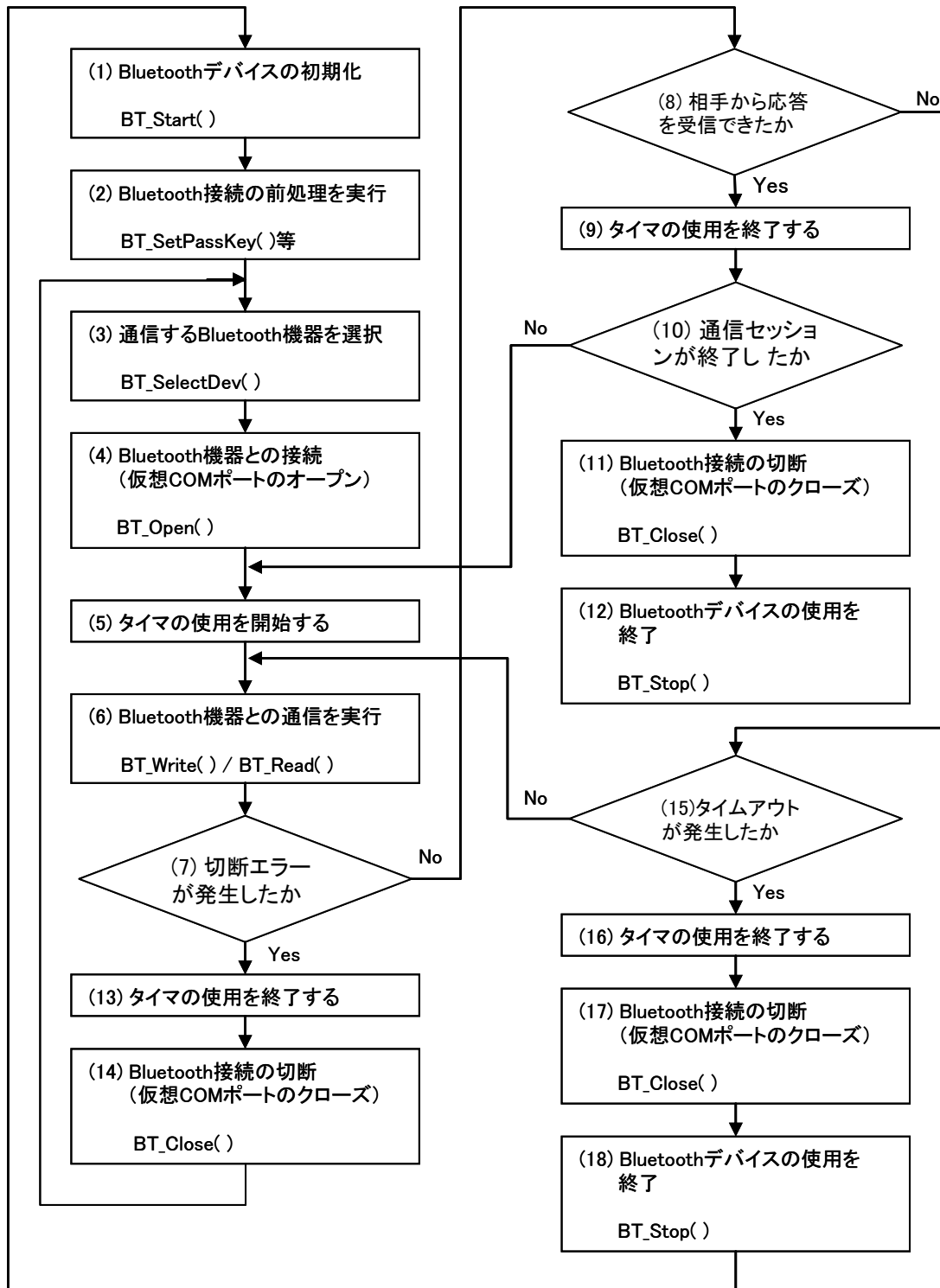
※ 時間が空いてしまう場合は、BT_Open→通信→BT_Close の部分を繰り返す事により BT_Start()および BT_Stop()にかかる時間を節約できます。ただし、相手から切断された場合、電源 OFF した場合等、関数からエラーが戻ってきた場合や、次項に示すようにプリンタからの応答がない場合には、所定のエラー処理を行ってください。

(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。

8.3. 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	<p>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</p> <p>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</p>
関数名	主なエラー対処方法
BT_Start	<ul style="list-style-type: none"> ・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	<p>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</p>
関数名	主なエラー対処方法
BT_Close	<ul style="list-style-type: none"> ・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	<p>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</p>
関数名	主なエラー対処方法
BT_Inquiry	<ul style="list-style-type: none"> ・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	<p>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</p>	
関数名	主なエラー対処方法	
BT_GetLocalInfo	<ul style="list-style-type: none"> ・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		<ul style="list-style-type: none"> ・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	電源関数の機能を用いてLB0の通知モードに設定されており、LB0エラー(主電池なし、電池蓋開き)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_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
BT_SetLocalInfo	・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_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	・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_Write	
BT_QueryRx	

エラー値	BTERR_LB1
詳細	電源関数の機能を用いてLB1の通知モードに設定されており、LB1エラー(主電池電圧低下)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_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
BT_SetLocalInfo	・電池交換後、BT_Start 関数から処理を実行してください
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	・電池交換後、BT_Start 関数から処理を実行してください
BT_Write	
BT_QueryRx	

エラー値	BTERR_LB2
詳細	電源関数の機能を用いてLB2の通知モードに設定されており、LB2エラー(副電池電圧なし)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_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
BT_SetLocalInfo	・電池交換後、BT_Start 関数から処理を実行してください
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	・電池交換後、BT_Start 関数から処理を実行してください
BT_Write	
BT_QueryRx	

エラー値	BTERR_LB4
詳細	電源関数の機能を用いてLB4の通知モードに設定されており、LB4エラー(APO 発生)となったときに発生します 通知モード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
BT_SetLocalInfo	・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_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
BT_Read	・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_Write	
BT_QueryRx	

エラー値	BTERR__LB5
詳細	電源関数の機能を用いてLB5の通知モードに設定されており、LB5エラー (OFF キー押下による電源 OFF) となったときに発生します 通知モード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
BT_SetLocalInfo	・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_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
BT_Read	・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_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_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み込みまたは書き込み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	・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
BT_LoadDevInfo	

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	・関数を再実行してください
BT_SetLocalInfo	
BT_Inquiry	
BT_GetDevInfo	
BT_GetDevName	
BT_SetPassKey	
BT_SelectDev	
BT_Open	
BT_Close	

エラー値	BTERR__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	<ul style="list-style-type: none"> 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_SetLocalInfo	
BT_Inquiry	
BT_GetDevInfo	
BT_GetDevName	
BT_SetPassKey	
BT_SelectDev	
BT_Open	
BT_Close	

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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 他の Bluetooth デバイスが探索に回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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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__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	<ul style="list-style-type: none"> 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	・関数を再実行してください
BT_SetLocalInfo	・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_Inquiry	
BT_GetDevInfo	
BT_GetDevName	
BT_SetPassKey	
BT_SelectDev	
BT_Open	
BT_Close	

エラー値	BTERR__NAK14
詳細	Bluetooth モジュールから NAK14 エラーが戻ったときに発生します Bluetooth モジュールに送信したコマンドが実行されませんでした
関数名	主なエラー対処方法
BT_GetLocalInfo	・関数を再実行してください
BT_SetLocalInfo	
BT_Inquiry	
BT_GetDevInfo	
BT_GetDevName	
BT_SetPassKey	
BT_SelectDev	
BT_Open	
BT_Close	

エラー値	BTERR__NAKMINUS1
詳細	Bluetooth モジュールから NAK-1 エラーが戻ったときに発生します Bluetooth モジュールが故障している可能性があります(通常は発生しません)
関数名	主なエラー対処方法
BT_GetLocalInfo	・関数を再実行してください
BT_SetLocalInfo	・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_Inquiry	
BT_GetDevInfo	
BT_GetDevName	
BT_SetPassKey	
BT_SelectDev	
BT_Open	
BT_Close	

8.4. ファンクション詳細

ファンクション詳細を次ページより示します。

8.4.1. BT_Start

Bluetooth 通信機能が使用できる状態にします。

```
H BT_Start(void);
```

パラメータ

xxx_xx
なし

戻り値

(エラー詳細は、エラー値取得関数 (BT_Err_Get) にて取得してください)

関数結果

E_BTOK	正常終了
E_BTNG	異常終了

補足

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。

8.4.2. BT_Stop

Bluetooth 通信機能の使用を終了します。

```
H BT_Stop(void);
```

パラメータ

なし

戻り値

(エラー詳細は、エラー値取得関数 (BT_Err_Get) にて取得してください)

関数結果

E_BTOK	正常終了
E_BTNG	異常終了

補足

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。

8.4.3. 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

(エラー詳細は、エラー値取得関数 (BT_Err_Get) にて取得してください)

関数結果

E_BTOK	正常終了
E_BTNG	異常終了

補足

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。

8.4.4. 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

(エラー詳細は、エラー値取得関数 (BT_Err_Get) にて取得してください)

関数結果

E_BTOK	正常終了
E_BTNG	異常終了

補足

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。

8.4.5. 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

(エラー詳細は、エラー値取得関数 (BT_Err_Get) にて取得してください)

関数結果

E_BTOK	正常終了
E_BTNG	異常終了

補足

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たり、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。

8.4.6. 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

(エラー詳細は、エラー値取得関数 (BT_Err_Get) にて取得してください)

関数結果

E_BTOK	正常終了
E_BTNG	異常終了

補足

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たり、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。

本関数は以下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 */
```

8.4.7. 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

(エラー詳細は、エラー値取得関数 (BT_Err_Get) にて取得してください)

関数結果

E_BTOK 正常終了
E_BTNG 異常終了

補足

本関数を実行中に中断キーが押されたり、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。

8.4.8. 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

(エラー詳細は、エラー値取得関数 (BT_Err_Get) にて取得してください)

関数結果

E_BTOK 正常終了

E_BTNG 異常終了

補足

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。

8.4.9. 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

(エラー詳細は、エラー値取得関数 (BT_Err_Get) にて取得してください)

関数結果

E_BTOK	正常終了
E_BTNG	異常終了

補足

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。

8.4.10. 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

(エラー詳細は、エラー値取得関数 (BT_Err_Get) にて取得してください)

関数結果

E_BTOK	正常終了
E_BTNG	異常終了

補足

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たり、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。

8.4.11. BT_Close

Bluetooth 接続を切断し、Bluetooth 通信を終了します。

```
H BT_Close(void);
```

パラメータ

なし

戻り値

(エラー詳細は、エラー値取得関数 (BT_Err_Get) にて取得してください)

関数結果

E_BTOK	正常終了
E_BTNG	異常終了

補足

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。

8.4.12. BT_Read

Bluetooth 通信において、受信データの読み込みを実行します。

```
H BT_Read(  
    B *buff,  
    UH ReadSize,  
    UH *GetSize  
);
```

パラメータ

buff

受信データを格納するバッファのポインタ

ReadSize

受信データを格納するバッファのサイズ(バイト数)

GetSize

読み込みデータ数(受信バッファから読み込みできたバイト数)

戻り値

(エラー詳細は、エラー値取得関数 (BT_Err_Get) にて取得してください)

関数結果

E_BTOK	正常終了
E_BTNG	異常終了

補足

本関数を実行中に中断キーが押されたり、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)。

8.4.13. BT_Write

Bluetooth 通信において、送信データの書き込みを実行します。

```
H BT_Write(  
    B *buff,  
    UH WriteSize,  
    UH *PutSize  
);
```

パラメータ

buff

送信データを格納するバッファのポインタ

WriteSize

送信データ数(送信バッファに書込むバイト数)

PutSize

書き込みデータ数(送信バッファに書き込みできたバイト数)

戻り値

(エラー詳細は、エラー値取得関数 (BT_Err_Get) にて取得してください)

関数結果

E_BTOK	正常終了
E_BTNG	異常終了

補足

本関数を実行中に中断キーが押されたり、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)。

8.4.14. BT_QueryRx

受信バッファより読み込み可能なデータ数の問い合わせを実行します。

```
H BT_QueryRx(  
    H *RcvDataSize  
);
```

パラメータ

RcvDataSize

読み込み可能なデータ数(バイト)

戻り値

(エラー詳細は、エラー値取得関数 (BT_Err_Get) にて取得してください)

関数結果

E_BTOK 正常終了

E_BTNG 異常終了

補足

8.4.15. 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

(エラー詳細は、エラー値取得関数 (BT_Err_Get) にて取得してください)

関数結果

E_BTOK 正常終了

E_BTNG 異常終了

補足

エラーフラグの内容は、ファイルに保存しません。

8.4.16. 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

(エラー詳細は、エラー値取得関数 (BT_Err_Get) にて取得してください)

関数結果

E_BTOK 正常終了

E_BTNG 異常終了

補足

エラーフラグの内容は、ファイルから読み出されません。

8.4.17. 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(void);
```

パラメータ

なし

戻り値

関数結果

補足

エラーコードについては、8.3 エラー詳細を参照してください。

9. 通信ユーティリティ部関数

- 関数名は、プロトコル毎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。
- FIR モードで使用する場合は、予め Ir モード設定関数により FIR に切替後以下に説明する 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。

9.1. 通信インタフェース

9.1.1. 使用形態

本機では、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以下の通りです。

プロトコル	転送方向 (本機を基準)	対象ファイル	COM0 (赤外線) 接続機器			備考
			I/O BOX *1	本機	IRDA アダプタ	
マルチドロップ	受信	アプリケーション	◎	×	×	*2
		データファイル	◎	×	×	
	送信	アプリケーション	×	×	×	
		データファイル	◎	×	×	
FLINK	受信	アプリケーション	◎	○	○	
		データファイル	◎	○	○	
	送信	アプリケーション	◎	○	○	
		データファイル	◎	○	○	
DT500	受信	アプリケーション	◎	×	×	*3
		データファイル	◎	×	×	*3
	送信	アプリケーション	◎	×	×	*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。

9.1.2. 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	本機用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

9.1.3. 排他制御

複数プロトコルは同時使用できません。

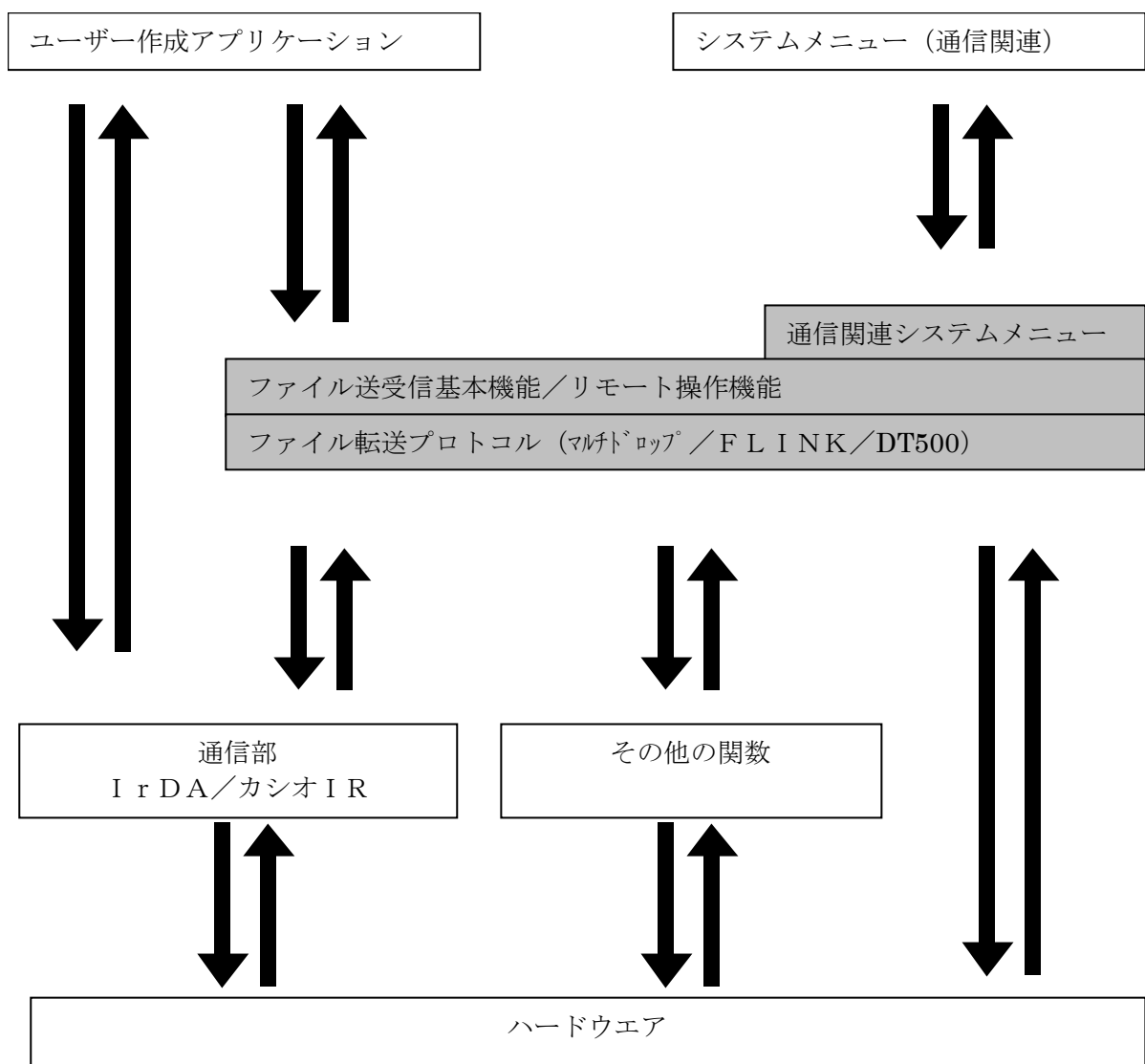
9.1.4. 転送ドライブ

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。

9.2. ソフトウェアブロック構成図

通信ユーティリティを中心としたソフトウェ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、Ir DA通信、その他の関数を使用する事で作成できます。

図8.1 ソフトウェアブロック構成



9.3. マルチドロップ

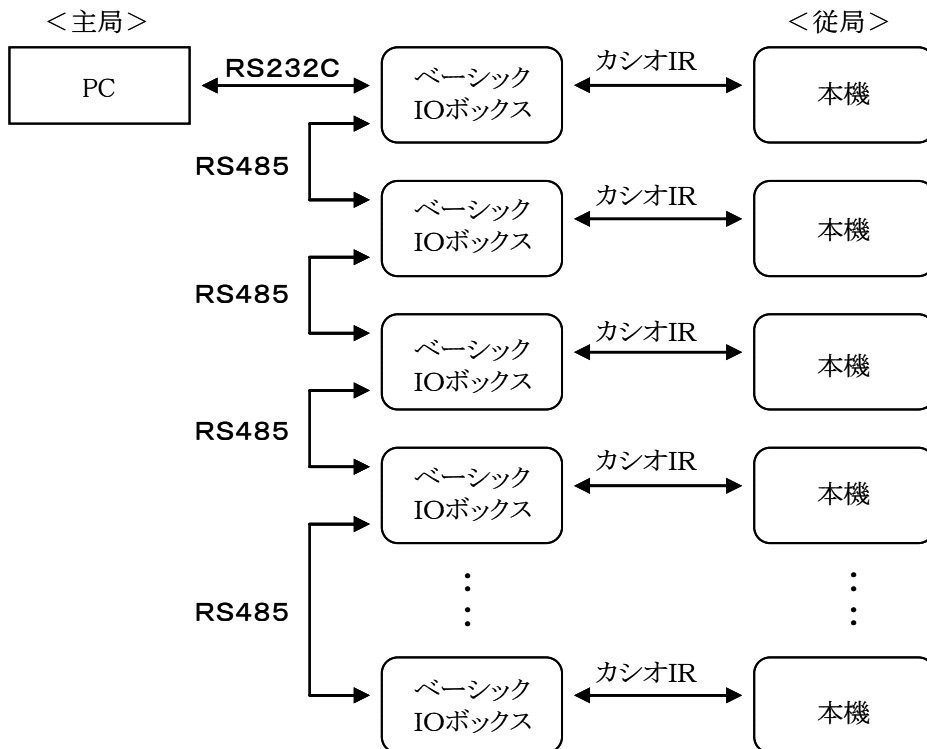
9.3.1. 通信仕様

(1) 通信構成

本プロトコルでは主局がホストPCに、従局が本機になります。
本プロトコルでは、以下のファイル転送機能を提供します。

① 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
共通パラメータ			
データ長	8ビット固定		
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ビット/2ビット)	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
ポーリング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)

主局のファイルに従局へ送信する。以下の特徴があります。

- 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み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。*1
- 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。

※ 1 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はなく、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)

従局のファイルの主局へ送信します。

- 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み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、送信側と受信側にて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。

9.3.2. 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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、本機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)
```

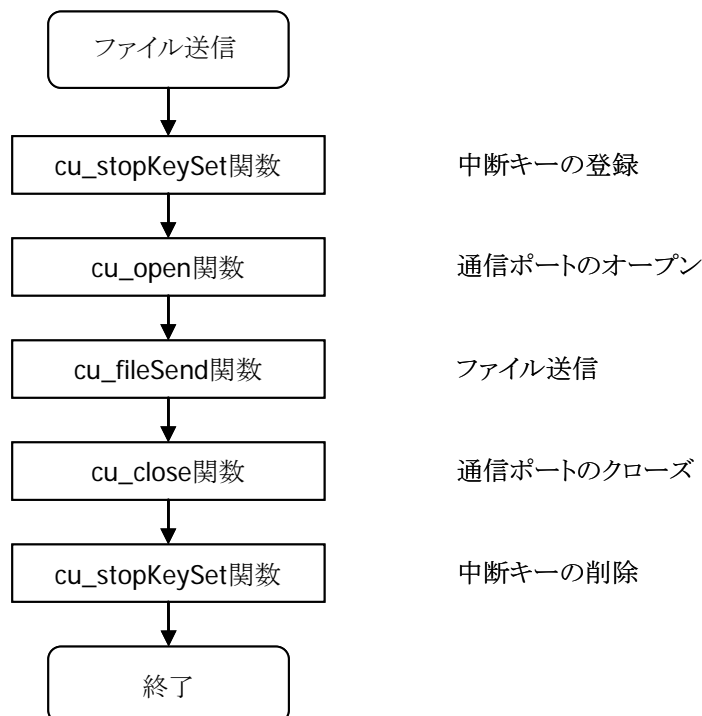
*実際の表示位置は画面モードにより異なります。

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]

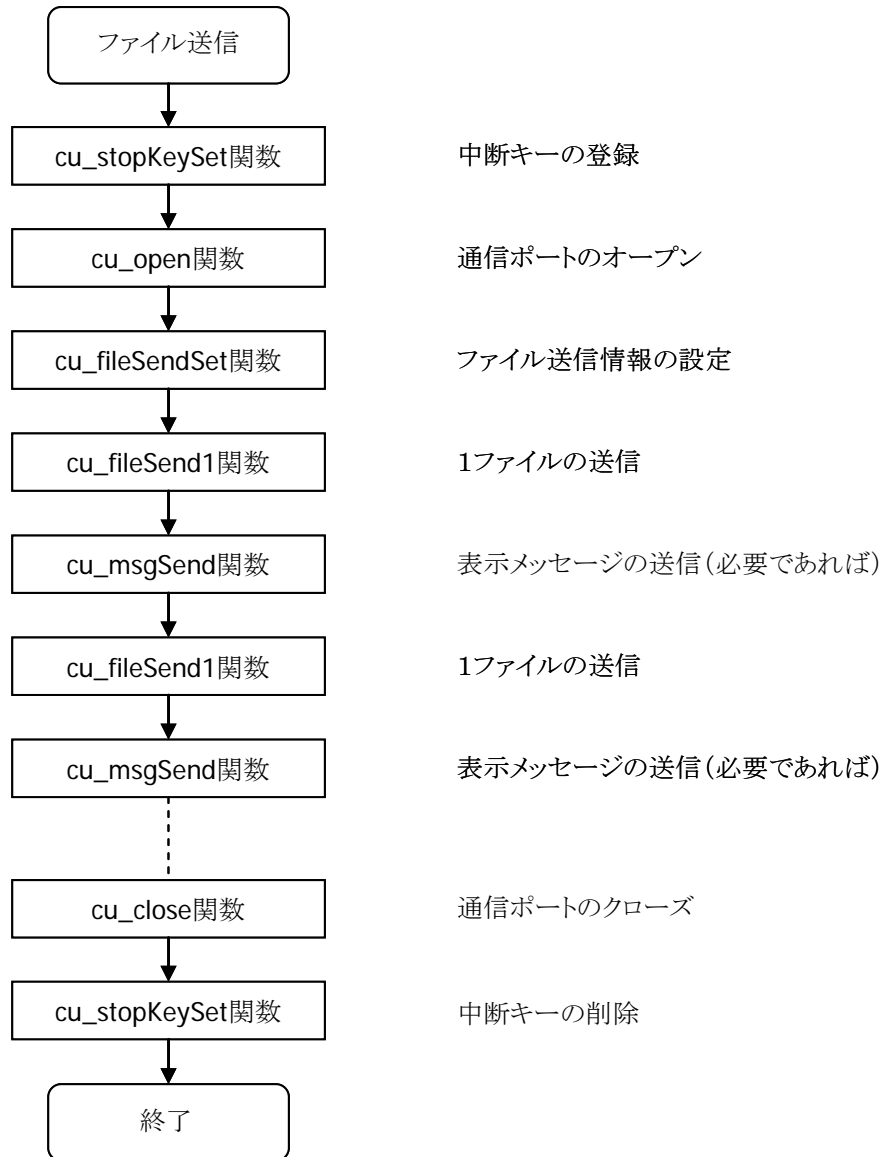
ファイル送信はファイル送信関数にて一括して送信できます。
この場合、通信中に画面表示メッセージを送信することはできません、ファイル一括送信の手順は以下の通りです。



[1ファイル送信関数によるファイル送信]

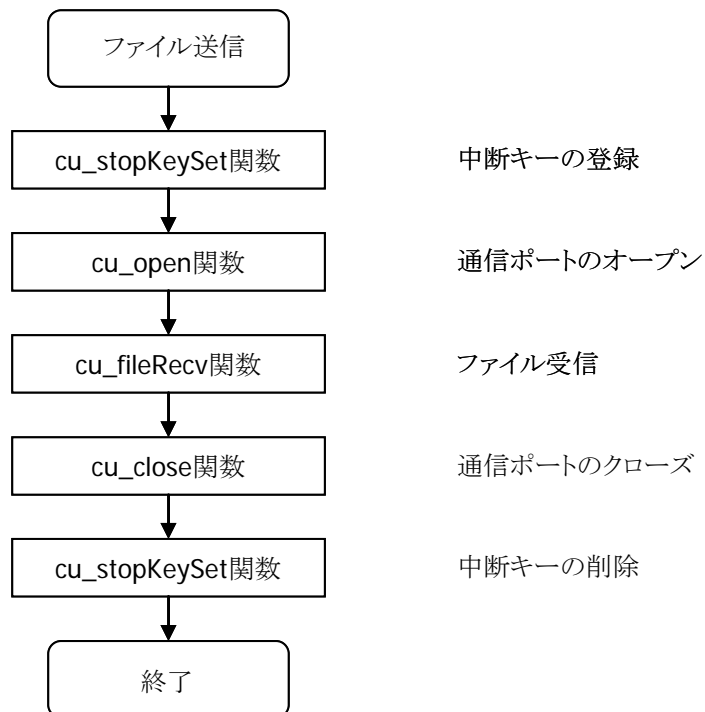
ファイル送信中に画面表示メッセージを送信する場合や通信を中断する場合には、ファイル送信情報の設定および1ファイル送信関数を用いて1ファイルずつ送信するようにし、その中で画面表示メッセージの送信や通信の中断を行います。

画面表示メッセージ送信の手順は以下の通りです。



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以下の通りです。



9.4. FLINKプロトコル機能

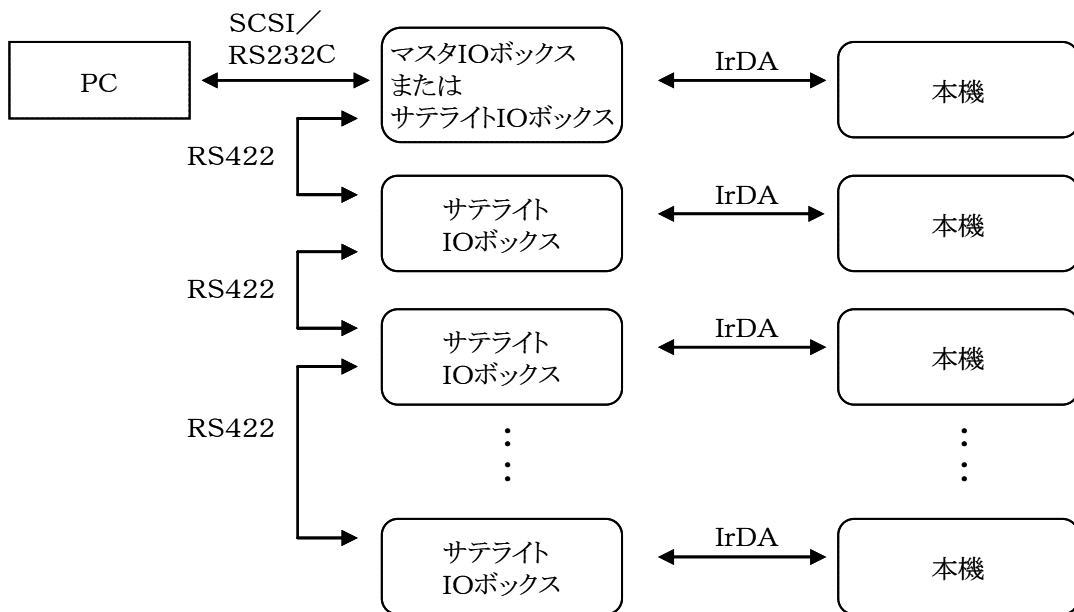
9.4.1. 通信仕様

(1)通信構成

本プロトコルのファイル転送機能は以下の通りです。

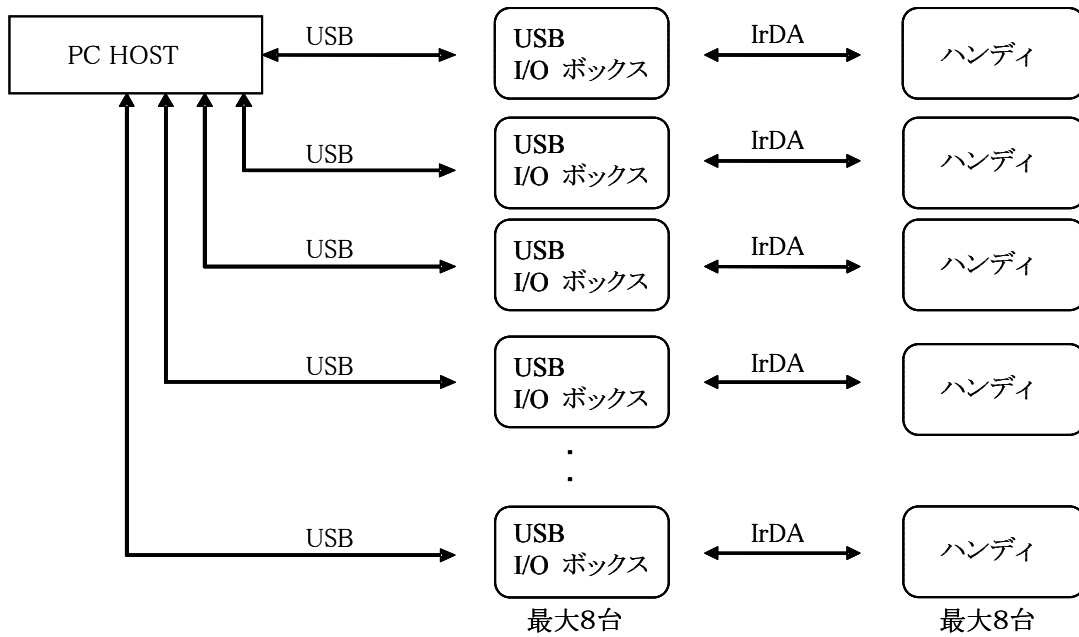
①サテライトIOボックス経由でのホスト通信

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。



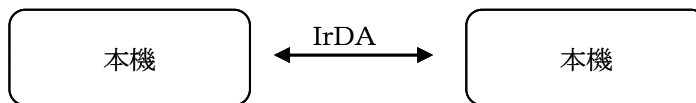
②USB IO ボックス経由でのホスト通信

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)
共通パラメータ		
セッション確立待ちタイムアウト時間(*1)	システム環境設定(データ管理部)より取得します (0 ~ 3600 秒)	
受信待ちタイムアウト時間(*2)	システム環境設定(データ管理部)より取得します (0 ~ 600 秒)	
セッション終了待ちタイムアウト時間(*3)	システム環境設定(データ管理部)より取得します (0 ~ 600 秒)	
COMO(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

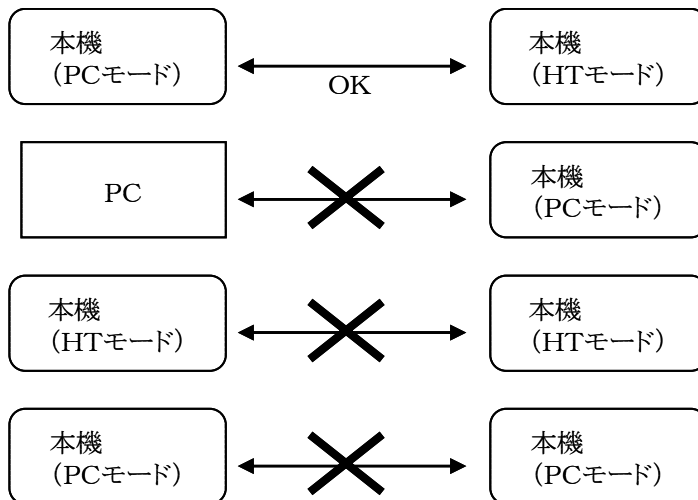
通信ユーティリティでは接続構成の違いにより以下のモードをサポートします。
これらのモードはオープン時に選択します。

①HT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-本機間通信時および本体-本体間通信時に、どちらか一方の本機が選択されます。

②PCモード

- セッション確立後、本機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、どちらか一方の本機が選択されます。



(4)コマンド送信権

①HT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、本機は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モードの本機にはスクリプトファイル実行機能はありません。

②PCモード

PCはセッション確立後に本機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。

本機から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。

※ 1 セッションとは回線オープン時に、相手局確認等のネゴシエーションをさします。

赤外線通信(IrDA)では相手局検出、転送速度の決定等を行います。

(5) 処理概要

以下に各関数内の処理概要およびエラー発生時の処理を示します。

エラー発生時は、直ちに通信を終了します。

この場合、送信権の有無に関らず、先にエラーを検出した側がエラー情報(カテゴリコード・エラー詳細コード)を終了指示コマンドに設定し、相手局へ送信します。

相手局は、受信した終了指示コマンドのエラー情報により、異常終了を検出します。

(エラー処理は、自局内でエラーを検出した場合と、相手局からのエラー終了指示コマンドを受信した場合の両方を指します。)

関数	送信権局の処理	被送信権局の処理	エラー発生時の処理
ファイル送信	コマンド送信後、指定ファイルを順次送信します	指定ファイルを順次受信します	受信中ファイルの削除を行います
ファイル受信	コマンド送信後、指定ファイルを順次受信します	指定ファイルを順次送信します	
ファイル追加	コマンド送信後、指定ファイルを送信します	転送ファイルをテンポラリファイル(FL.ADD)に受信後、ファイルを追加します。追加完了後、テンポラリファイルを削除します	テンポラリファイルを削除します
ファイル削除	コマンドを送信します	指定ファイル/ディレクトリを削除します	削除したファイル/ディレクトリの復旧は行いません
ファイル移動	コマンドを送信します	指定ファイルを移動します	移動後の削除ファイルは復旧しません
ディレクトリ作成	コマンドを送信します	指定ディレクトリを作成します	作成したディレクトリは削除しません
日付時刻の取得	コマンドを送信後、日付時刻情報を受信します	システム日付時刻情報を送信します	—————
日付時刻の設定	コマンドを送信します	日付時刻をシステムに設定します	設定後の日付時刻は復旧しません
ファイル情報の取得	コマンドを送信後、ファイル情報を受信します	指定ファイル情報を送信します	—————
ファイル情報の設定	コマンドを送信します	指定ファイルの情報を変更します	設定後のファイル情報は復旧しません
ディスク情報の取得	コマンドを送信後、ディスク情報を受信します	指定ディスク情報を送信します	—————
システム情報の取得	セッション確立時に相手局情報を取得するため、通信は行いません	セッション確立時に相手局情報を取得するため、通信は行いません	—————
画面メッセージ表示	コマンドを送信します	受信データを画面左上より表示します	表示後のメッセージはクリアしません
ブザー鳴動	コマンドを送信します	3秒間ブザーを鳴らします	—————
IDLE通知	(HTモードのみ) コマンドを送信後、コマンド受信待ちとなります	(PCモードのみ) コマンド送信権を取得します	—————
終了指示	コマンド送信後、通信を終了します	通信を終了します	—————

9.4.2. 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、本機は通常モード (HTモード) でオープンする必要があります。
- 本機と通信を行う場合、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

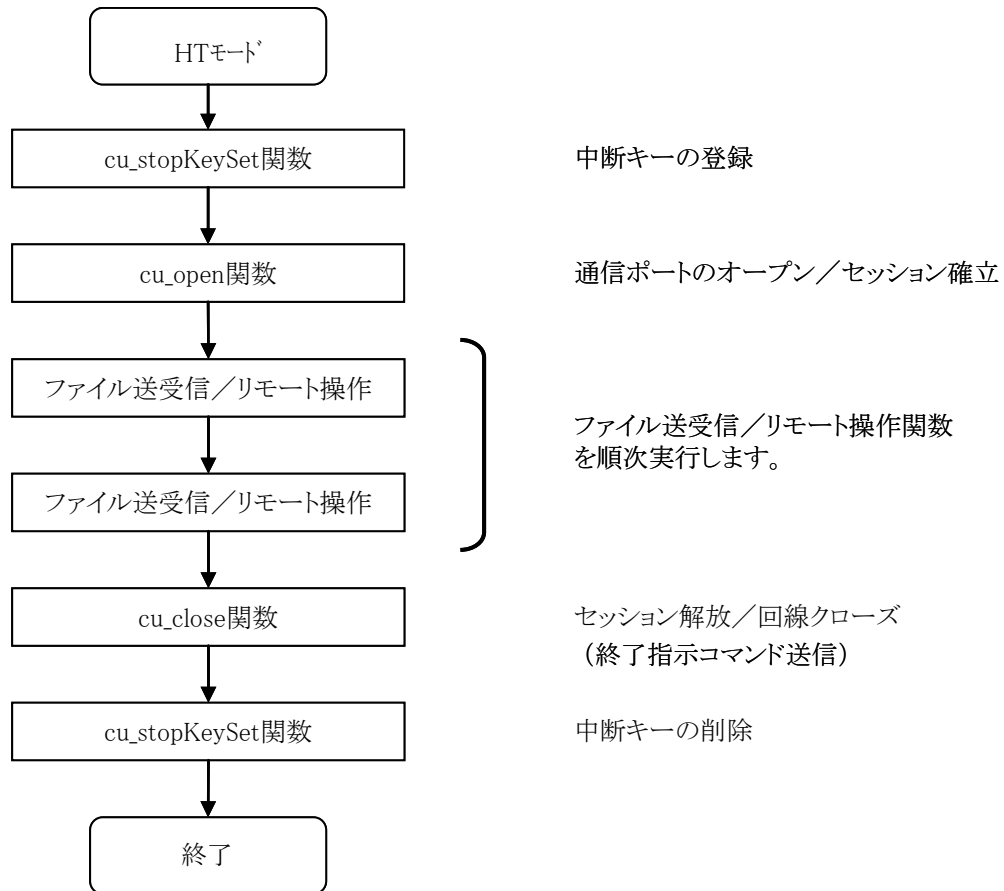
- 中断キーの選択登録／削除を行います。
登録は回線オープンの前に、削除はクローズの後に行う必要があります。

④ エラー詳細情報の取得

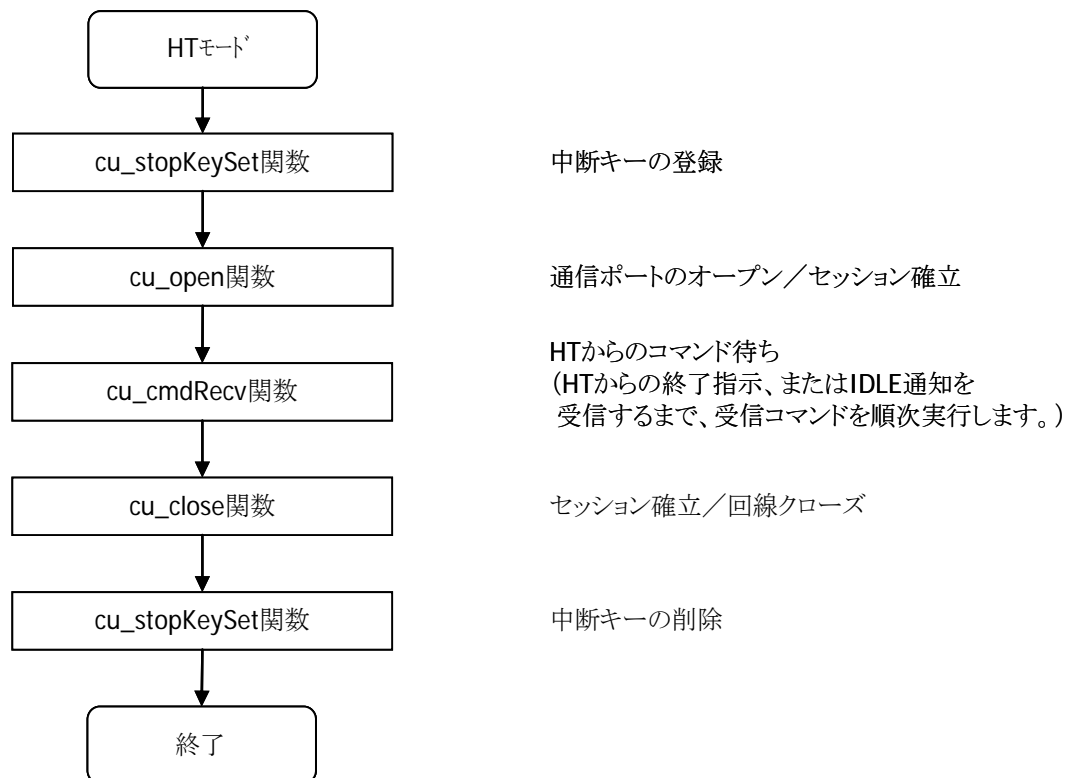
- エラー情報の取得を行います。
発生エラーコード、相手局からの終了指示コマンドのエラー情報 (カテゴリコード・エラー詳細コード) 等の取得が可能です。

HTコマンド送信による通信

[HTモード基本フロー]

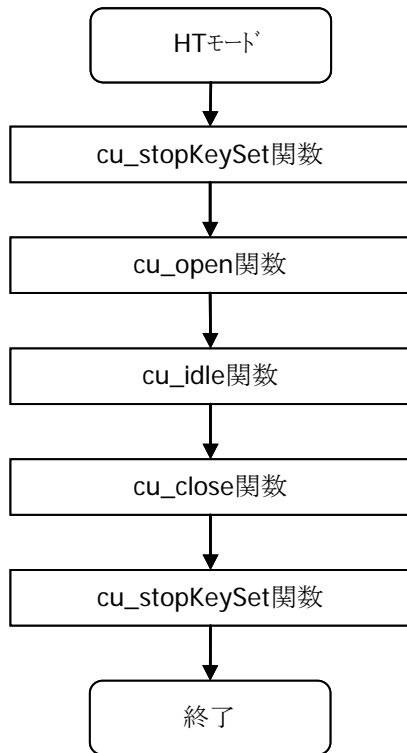


[PCモード基本フロー]



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

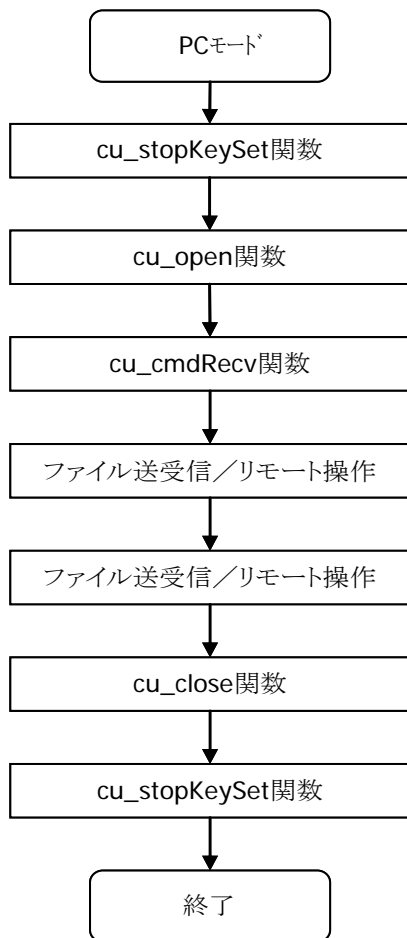
通信ポートのオープン/セッション確立

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。

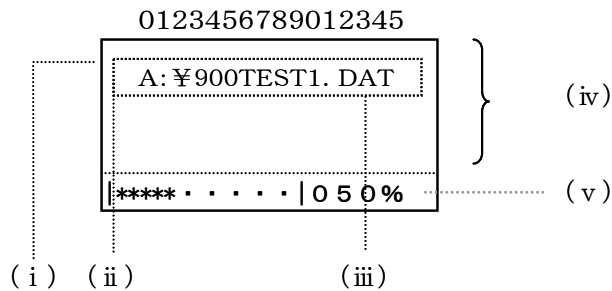
送信権局はファイル送信、追加および受信コマンドを使用して、相手局とのファイル転送を実現します。

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コラム(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)は以下の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以下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全てのファイルが転送対象となります。指定ディレクトリ傘下にサブディレクトリが存在した場合はサブディレクトリ名を付加してファイルの送信を行います。

(例)

[送信ファイル名] [送信先ディレクトリ名]
 “A:¥SEND¥AAA.DAT” ”B:¥RECV¥”

(送信側ディレクトリ構成)

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

(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以下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全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。

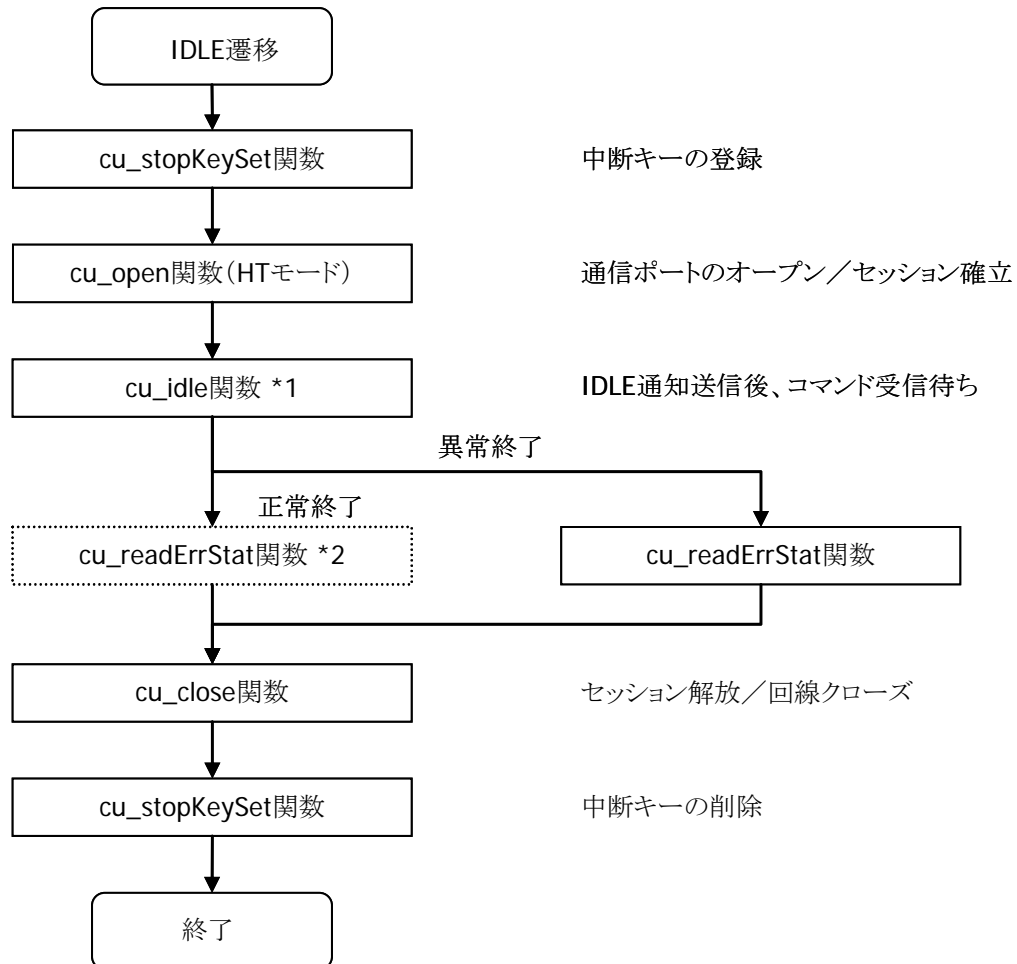
(c)ワイルドカードの使用

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。

④IDLE 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。

⑤PCモードコマンド待ち (cu_cmdRecv)

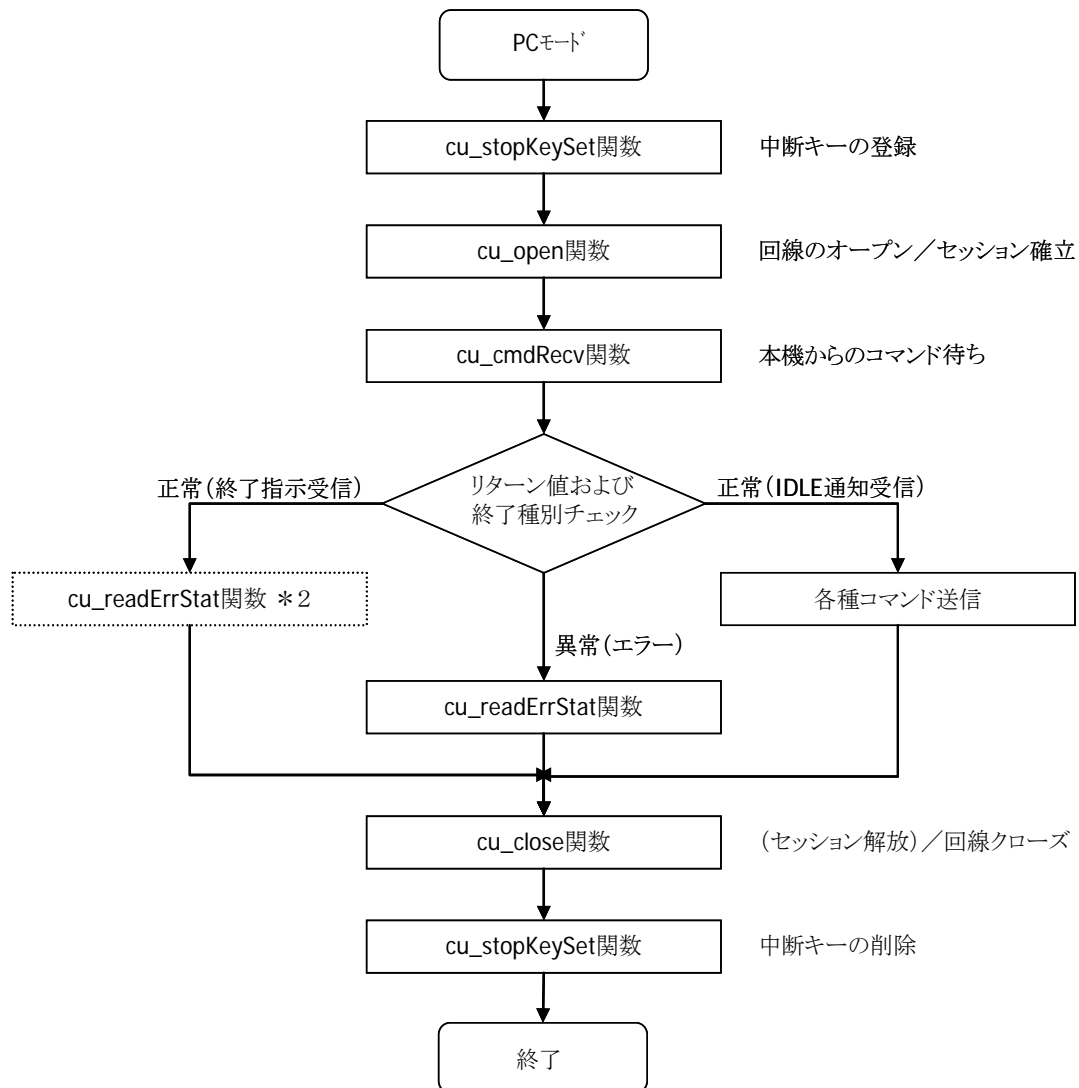
本機からのコマンド受信待ち状態となる。PCモードでのみ使用可能です。

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・・・本機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。

9.4.3. 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以下の通りです。

- ・ 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以下の通りです。

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 (本機／PC (AT互換機)／PC (98シリーズ))

OSモデル情報 (本機モデル種別／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。

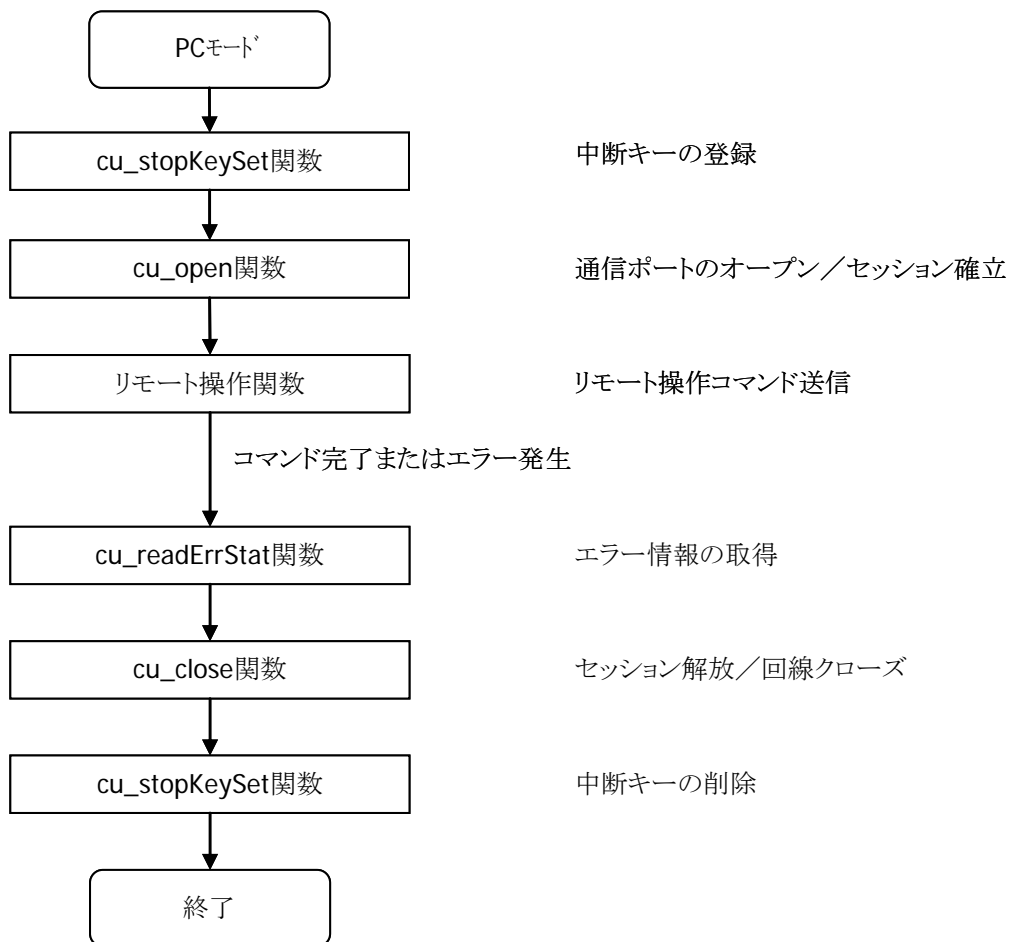
相手局HTはメッセージを画面に表示します。

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。

相手局本機は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]



9.4.4. 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以下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。

9.5. DT500 プロトコル機能

9.5.1. 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。

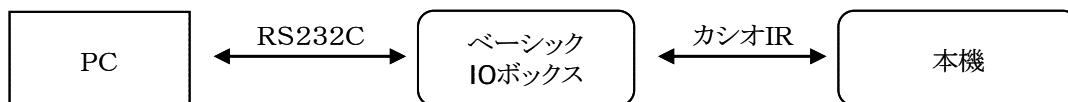
本プロトコルでは、以下の構成でのファイル転送機能を提供します。

・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)

(主局/従局)



(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。

※ 1 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。

①DT500 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。
ただし、システムメニューによるバイナリ送信は行なえません。
- ・ 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。(※1)

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”)

- ・ 本機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。(※1)

③DT-900 システム転送用変換ファイル(拡張子、“DTF”)

- ・ 本機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。
- (※1)DT500 では DT-BASIC 形式ファイルへの変換等を行ないませんが、本機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。以下の特徴があります。

- ・ 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。以下の特徴があります。

- ・ 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。

9.5.2. 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。

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

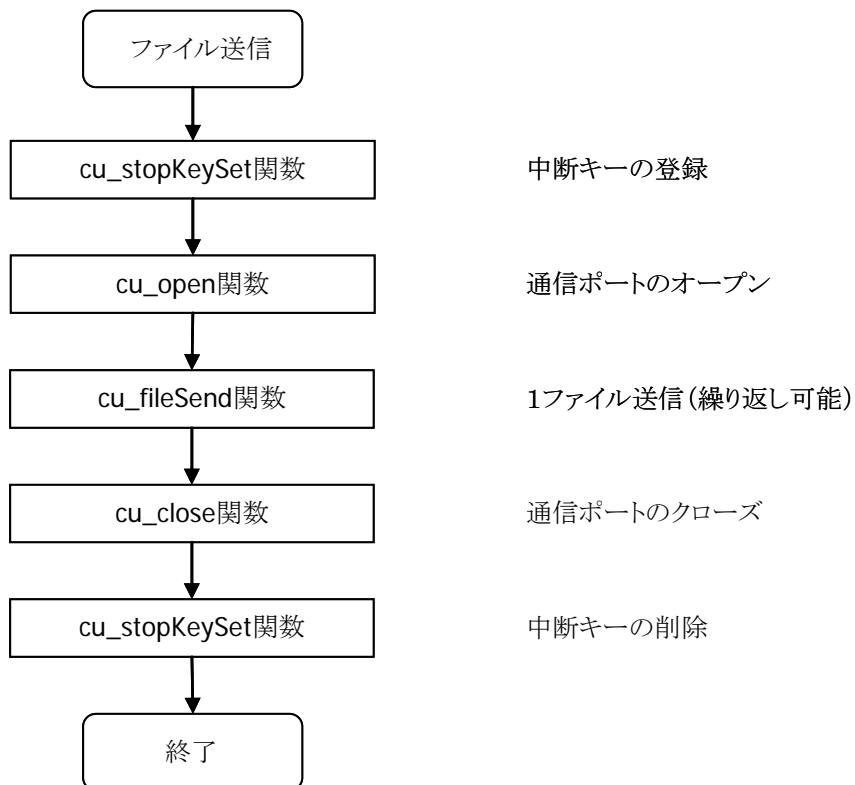
PCとのファイル転送(送信、受信)を行うための関数です。
尚、ファイル転送時、HTの画面上に進捗グラフを表示することができます。
グラフ表示フォーマットは次の通りです。

```
0123456789012345
CONFIG. HTS
      XXX%
|*****|. . . . .|
```

← 転送ファイル名(グラフ表示指定行)
← 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)

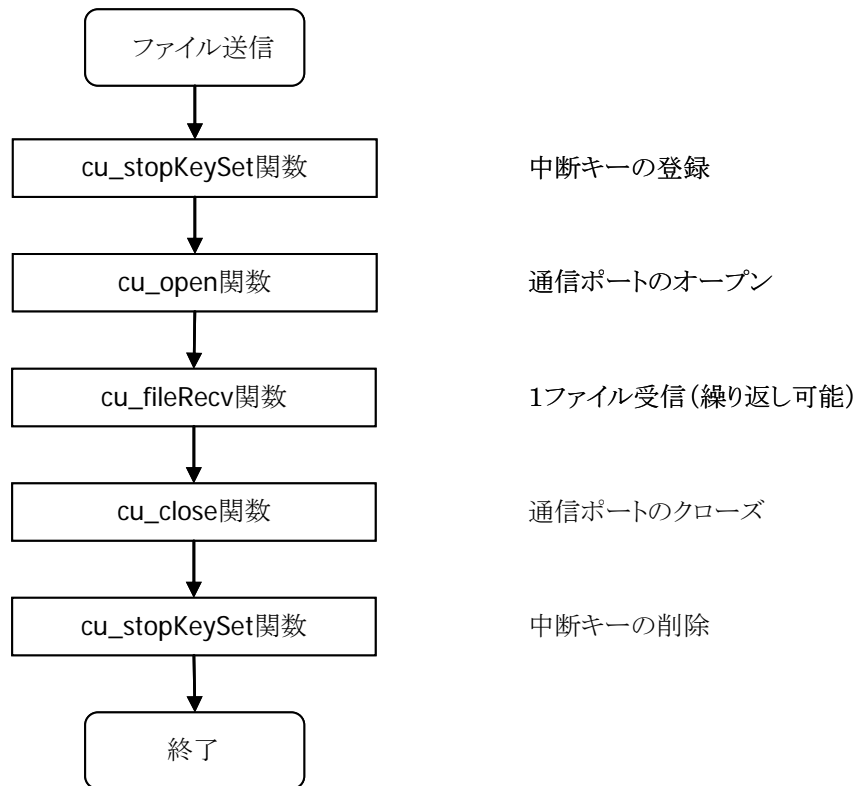
①ファイル送信

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以下の通り。



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以下の通りです。



(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。

①SOH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。

②STX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。

③ETX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。

9.5.3. 補足

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

- ・ 本機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

- ・ 本機では 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わず、そのまま格納します。

③本機システム関連ファイル

- ・ 本機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本機へ送信する時、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本機へ送信する時、指定するフィールド長は以下の通りです。

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。

9.6. ファンクション詳細

ファンクション詳細を次ページより示します。

9.7. 共通ファンクション

9.7.1. 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

関数結果

E_OK	正常終了
E_PRM	パラメータエラー

補足

9.7.2. 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

関数結果

E_OK 正常終了

E_PRM パラメータエラー

補足

9.8. マルチドロッププロトコル

9.8.1. 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           /* パリティビット */
                    : PARI_ODD
                    : PARI_EVN
    W stop_bit:   : STOP_1             /* ストップビット */
                    : 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

補足

9.8.2. 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

COMO 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、以下の設定が必要です。

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;
```

関数結果

E_OK 正常終了 (stat は全て CU_STAT_TRANS)
E_NG 異常終了 E_PRM: パラメータエラー

補足

- ・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

9.8.3. 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、以下の設定が必要です。

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

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

・当関数の使用前に cu_open 関数を実行する必要があります。

9.8.4. 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

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。

9.8.5. 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 (必須ですが本機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

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

- ・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。

9.8.6. 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

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。

9.8.7. 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。

9.8.8. 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

9.8.9. 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_PRM

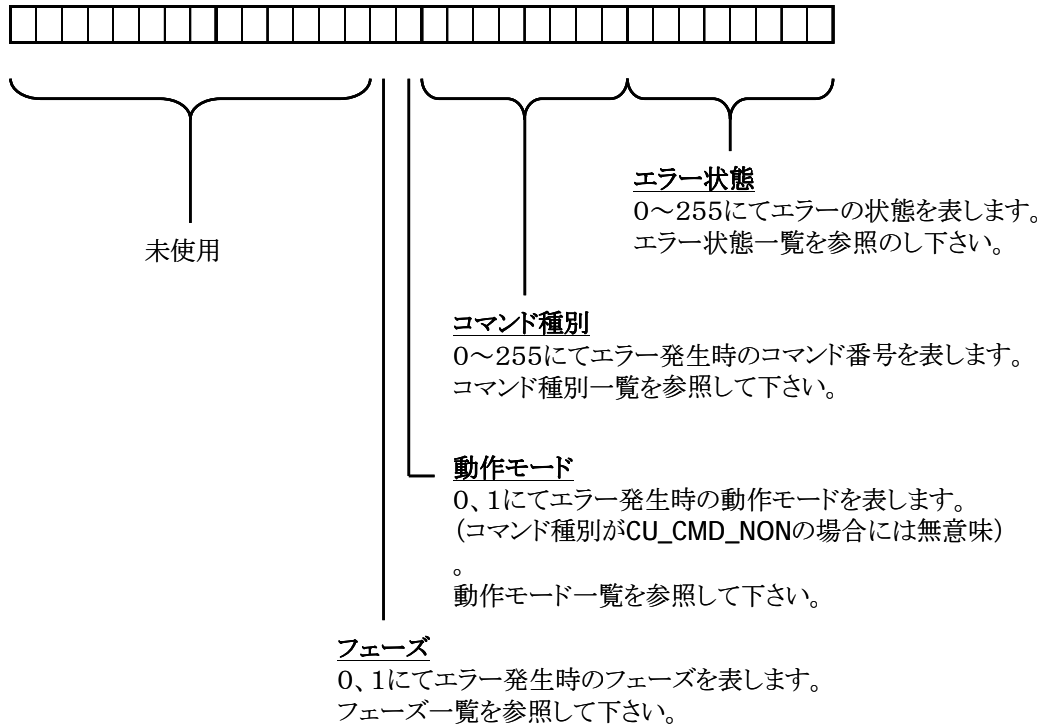
パラメータエラー

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_FL_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

9.8.10. 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

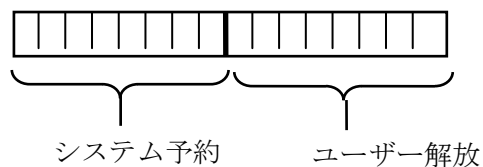
通信ポート

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。
- ・それ以外の場合には返却値の値は不定となります。

9.9. 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。

9.9.1. 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いません。

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いません。

```
ER cu_open (
    H comNo,
    H irSpeed,
    CU_RSPRM *rsPrm,
    H mode
)
```

パラメータ

comNo

通信ポート

COMO カシオ IR インタフェース

irSpeed

赤外通信最高速度 CU_B2400～CU_B115K

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         /* パリティビット なし */
                          : CU_PARI_ODD         /*        奇数        */
                          : CU_PARI_EVN         /*        偶数        */
    H stop_bit:            : CU_STOP1            /* ストップビット 1    */
                          : CU_STOP2            /*        2        */
} CU_RSPRM;
```

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

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。

9.9.2. 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、属性変更して書込みを行うかを指定します)

CU_PROTECT_VALID 強制書込みしない
CU_PROTECT_INVALID する

graphSet

グラフ表示情報

【ストラク構造】

```
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 設定時は以下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

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

9.9.3. 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

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

9.9.4. 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、属性変更して書込みを行うかを指定します)

CU_PROTECT_VALID 強制書込みしません

CU_PROTECT_INVALID 強制書込みします

graphSet

グラフ表示情報 (cu_fileSend 関数参照)

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

9.9.5. 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

COMO 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

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

9.9.6. 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

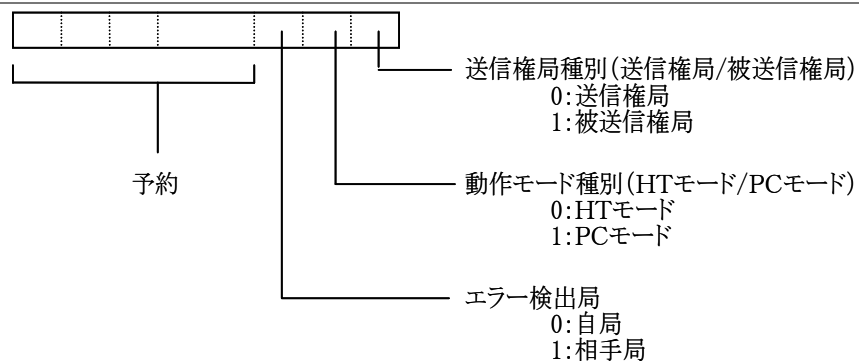
errInfo

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;
```



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	コマンドレンゲスエラー
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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 指定日付を確認
	06	システム時刻設定エラー	<ul style="list-style-type: none"> 指定時刻を確認
	07	タイマー使用エラー	<ul style="list-style-type: none"> タイマー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	<ul style="list-style-type: none"> CPU切替え禁止状態でないか確認
	09	致命的エラー	<ul style="list-style-type: none"> IrDA、通信関数からのエラー ローバッテリーの発生等が考えられる
	0A	通信中回線断エラー	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> CONFIG. HTS 異常 ファイルレイアウトを確認
	03	相手局不正	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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	

9.9.7. 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

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

9.9.8. 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

COMO 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

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

9.9.9. 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

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

9.9.10. 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

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

9.9.11. 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

関数結果

E_OK	正常終了
E_NG	異常終了
E_PRN	パラメータエラー

補足

9.9.12. 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

【ストラク構造】

```
typedef struct {
    B name[256]           : 検索されたファイル名 (フルパス名)
    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   : 読み出し専用ファイル
                          _A_SYSTEM    : システムファイル
                          _A_SUBDIR   : ディレクトリ
                          _A_ARCH     : アーカイブ
} CU_FINFO;
```

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

9.9.13. 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

【ストラク構造】

```
typedef struct {
    B name[256]                   : 設定するファイル名 (フルパス名)
                                   (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;
```

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

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

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

9.9.15. 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 に全て 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。

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

9.9.16. 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;       : モデル情報 (00h 固定)
} CU_SYSINFO;
```

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

9.9.17. 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

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

9.9.18. 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

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

9.9.19. cu_setloboxInfo

【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_setloboxInfo(  
    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

関数結果

E_OK	正常終了
E_PRM	パラメータエラー

補足

- 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。

9.9.20. 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以下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	再帰呼び出し無
CU_TRANS_RECURSIVE	再帰呼び出し有

fName

転送元ファイル名(複数指定およびワイルドカード指定可)

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 設定時は以下のパラメータは、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;
```

関数結果

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

補足

9.9.21. 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、以下の情報です。

(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 設定時は以下のパラメータは、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      : ファイル名行数 (1~12)
} CU_GRAPHSET;
```

関数結果

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	リストファイル読み込み時エラー
FCHK_NG0D	パラメータエラー

補足

9.10.DT500 プロトコル

9.10.1. 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

【ストラク構造】

```
struct sys_tty {
    W speed;           : B_1200~B_115200 /* 転送速度           */
    W length;          : CHAR_8, CHAR_7 /* データ長固定       */
    W parity;          : PARI_NON       /* パリティビット なし */
                      : PARI_ODD       /*                   奇数 */
                      : PARI_EVN       /*                   偶数 */
    W stop_bit;        : STOP_1        /* ストップビット    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

9.10.2. 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。

9.10.3. 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

9.10.4. 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

9.10.5. 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

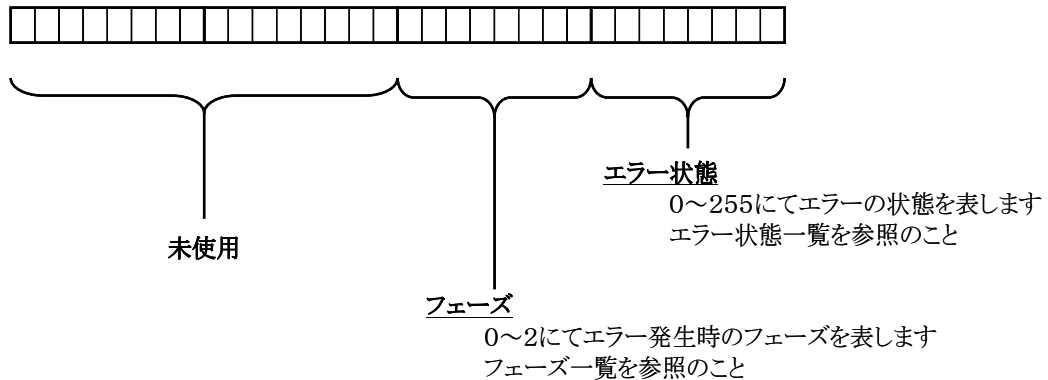
E_OK 正常終了
E_PRM パラメータエラー

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

9.10.6. 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

10. タイマ部

10.1. 機能

10.1.1. タイマー部

(1) タイマー1

1秒単位のインターバルタイマーです。

本タイマーは、タイムアウト時指定されているイベントフラグをONにします。

表9. 1 タイマー概要

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、指定されたイベントフラグをONにします

(2) タイマー2

31. 25msec単位のインターバルタイマーです。

本タイマーは、タイムアウト時指定されているイベントフラグをONにします。

表9. 2 タイマー概要

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

10.1.2. ブザー音鳴動部

ブザー音鳴動機能ではキークリック音、エラービープ音、サウンド音の3種類の音鳴動を提供します。

また、音鳴動が同時に発生した場合は、以下の優先順位に従います。

キークリック音 < サウンド音 < エラービープ音

- ブザー鳴動中に同レベルまたはそれより優先順位の低いブザー要求が行われた場合、そのブザー要求は無効になります。
ただし、サウンド音鳴動中にサウンド音要求(同じ優先順位のサウンド音)が行われた場合、1鳴動分だけバッファリングを行います。また、バッファリング中にサウンド音鳴動があった場合はその要求をウェイトします。
- ブザー鳴動中にそれより優先順位の高いブザー要求が行われた場合、ブザー鳴動を停止しブザー要求が行われたブザーを鳴動します。
また、サウンド音鳴動中にバッファリングまたはウェイトしていた場合に優先順位の高いブザー音鳴動要求があった場合は、バッファのクリアまたはウェイト無効にて復帰を行い、その要求のブザー音を鳴動します。

表9.3 ブザー音鳴動部機能一覧

機能	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 本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せん
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

10.1.3. 日付時刻制御部

(1) 日付設定／取得

現在の日付をバイナリデータで設定します。日付の設定範囲は1980年～2079年で、閏年にのみ2月29日設定が可能です。閏年以外の年に2月29日設定が行われた場合や、設定範囲外のデータで設定が行われた場合は、異常データとして日付設定は行われません。
また、現在日付をバイナリデータとして取得できます。

(2) 時刻設定／取得

現在の時刻をバイナリデータで設定できます。時刻の設定範囲は00:00:00～23:59:59で、設定範囲外のデータで設定が行われた場合、異常データとして時刻設定は行われません。
また、現在時刻をバイナリデータとして取得します。

10.2.ファンクション詳細

ファンクション詳細を次ページより示します。

10.2.1. s_settimer

1秒間隔のインターバルタイマーをセットします。
指定時間経過後に通知(イベントフラグ設定)します。
登録可能件数は10件です。それ以上の登録は異常終了となります。

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

パラメータ

flgid

イベントフラグID
FL_TM1_INT_ID を設定してください。

setptn

ビットパターン
FL_TM1_INT_RTC0 ~ FL_TM1_INT_RTC31

tmcnt

タイマーカウント
1~3600 (1 カウント = 1 秒)

戻り値

関数結果	
00h~09h	タイマー登録ID
E_PRM	パラメータエラー
E_TID_OVER	登録数オーバー

補足

本タイマーは最大+1秒の誤差が生じます。
また、タイマーが不要になった場合は必ず `s_timerend` 関数で、タイマーを削除して下さい。

10.2.2. s_timerend

登録済みタイマー1を削除します。

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

パラメータ

del_id

タイマー登録ID(00h~09h)

登録時に得られたIDを指定して下さい。

戻り値

関数結果

E_OK	正常終了
E_PRM	パラメータエラー
E_TID_NON	未登録タイマの削除

補足

10.2.3. s_settimer2

31. 25 ミリ秒間隔のインターバルタイマーをセットします。
指定時間経過後に通知(イベントフラグ設定)します。
登録可能件数は10件です。それ以上の登録は異常終了となります。

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

パラメータ

flgid

イベントフラグID
FL_TM2_INT_ID を設定してください。

setptn

ビットパターン
FL_TM2_INT_ITU0 ~ FL_TM2_INT_ITU31

tmcnt

タイマーカウント
1~115200 (1 カウント = 31. 25 ミリ秒)

戻り値

関数結果	
00h~09h	タイマー登録ID
E_PRM	パラメータエラー
E_TID_OVER	登録数オーバー

補足

本タイマーは最大+31. 25 ミリ秒の誤差が生じます。
また、タイマーが不要になった場合は必ず s_timerend2 関数で、タイマーを削除して下さい。

10.2.4. s_timerend2

登録済みタイマー2を削除します。

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

パラメータ

del_id

タイマー登録ID(00h~09h)

登録時に取得したIDを指定して下さい。

戻り値

関数結果

E_OK	正常終了
E_PRM	パラメータエラー
E_TID_NON	未登録タイマー削除

補足

10.2.5. s_beep

エラービープ音を鳴らします。

- ・周波数 = 4096Hz
- ・音長 = 100msec

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

パラメータ

なし

戻り値

なし

補足

エラービープ音要求時に現在エラービープ音が鳴動中の場合、エラービープ音要求は無視されます。

その他の音鳴動時は、音鳴動停止を行った後でエラービープ音の鳴動を開始します。

音量は、システムデータ管理で定義した値に従います。

10.2.6. s_sound

任意の周波数／音長にてサウンド音を鳴らします。

- ・周波数 = 0(無音), 128Hz ~ 4096Hz
- ・音長 = 0(停止), 1 ~ 160 (×25msec)

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

パラメータ

freq

周波数(0, 128~4096 Hz)

leng

音長(0, 1 ~ 160 ×25msec)

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

関数結果

E_OK	正常終了
E_PRM	パラメータエラー

補足

サウンド音鳴動中に本関数を実行した場合、1鳴動分のみバッファリングされます。
バッファリング中に本関数を実行した場合、鳴動待ちバッファが空くまでウェイトします。
サウンド鳴動中にエラービープ音鳴動関数が実行された場合、バッファリングはクリアされま
す。

音量はシステムデータ管理で設定した値に従います。

本関数は、バッファフル時以外即時復帰し、サウンド音鳴動を行います。

10.2.7. s_dateset

日付を設定します。

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

パラメータ

day_dat

日付格納エリアアドレス

【ストラク構造】

```
typedef struct day_tabl {  
    UH year:      : 西暦 (1980 ~ 2079)  
    UB month:     : 月 (1 ~ 12)  
    UB day:       : 日 (1 ~ 31) ただし、月よって変わります。  
} DAY_DAT;
```

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

10.2.8. 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

日付格納エリアアドレス

【ストラク構造】

```
typedef struct day_tabl {  
    UH year;      : 西暦 (1980 ~ 2079)  
    UB month;    : 月 (1 ~ 12)  
    UB day;      : 日 (1 ~ 31)  
} DAY_DAT;
```

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

本機能はメモリ内に格納されている日付データをそのまま取得しており、日付データ内容のチェックは行っていません。

10.2.9. s_timeset

時刻を設定します。

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

パラメータ

tim_dat

時刻格納エリアアドレス

【ストラク構造】

```
typedef struct tim_tabl {  
    UB  hour;    : 時 (0 ~ 23)  
    UB  mint;    : 分 (0 ~ 59)  
    UB  sec;     : 秒 (0 ~ 59)  
} TIM_DAT;
```

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

10.2.10. 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

時刻格納エリアアドレス

【ストラク構造】

```
typedef struct tim_tabl {  
    UB  hour;    : 時 (0 ~ 23)  
    UB  mint;    : 分 (0 ~ 59)  
    UB  sec;     : 秒 (0 ~ 59)  
} TIM_DAT;
```

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

本機能はメモリ内に格納された時刻データをそのまま取得しており、時刻データ内容のチェックは行っていません。

11. 電源

11.1. 機能

11.1.1. 主電池電圧低下監視／警告

本機は、主電池なし状態になる前に一定電圧以下になると警告を出します。
この警告は、一定電圧以下の状態が続いた場合に、表示右下にあるシンボルを点灯させます。
電圧が一定電圧以上に復帰するとシンボルを消灯します。
通知モード設定することで、主電源電圧低下 (LB1) 確定時APに通知することも可能です。

11.1.2. 副電池電圧低下監視／警告

本機は、副電池が一定電圧以下になると警告を出します。
この警告は、一定電圧以下の状態を検出した場合に、表示右下にあるシンボルを点灯させます。
電圧が一定電圧以上に復帰するとシンボルを消灯します。
通知モード設定することで、主電源電圧低下 (LB2) 確定時APに通知することも可能です。

11.1.3. 自動電源OFF制御 (APO: Auto Power Off)

APOとは、システムで設定した時間無操作状態が続いた場合、自動的に電源をOFFする機能です。
設定時間は、1分単位で1～59分の間で設定できます。
通知設定が行われている場合は、設定時に指定されたイベントフラグに特定ビット (FL_LB_INT_LB4) を立て電源OFFは行いません。
APOで電源OFFした場合の次回の電源ONは、システム設定のレジュームON／OFFの設定に関わらずレジュームON起動となります。

11.1.4. 自動バックライトOFF制御 (ABO: Auto Backlight Off)

ABOとは、システムで設定した時間無操作状態が続いた場合、自動的にバックライトをOFFする機能です。
設定時間は、1秒単位で10～59秒の間で設定できます。
ABOでOFFしたバックライトは、キー入力ですべてONします。

11.1.5. 低消費電力制御

キー待ち状態 (key_read/key_string/key_num をAPで呼ぶ) になった場合、CPUをSLEEP状態にし消費電力を抑える制御を行います。
従って、アプリケーションプログラムで処理のない場面では、キー待ち関数でウェイトすることを推奨します。

11.1.6. APO禁止／禁止解除

アプリケーションプログラムでAPOされたくない場面がある場合、APOを禁止することができます。

11.1.7. 電源通知モード設定／解除

通知モードが指定された時は、指定されているイベントを設定します。
以下の項目の通知が可能です。

No	通知項目	通常処理	通知モード処理	通知タイミング	備考
1	電源OFFキー (LB5)	電源OFF処理	電源OFFしない イベント設定	発生時	
2	主電池なしまたは 電池蓋外し (LB0)	電源OFF処理	電源OFF処理 イベント設定	次回立上げ時	
3	APO (LB4)	電源OFF処理	電源OFFしない イベント設定	発生時	
4	主電池警告 (LB1)	シンボル表示	シンボル表示 イベント設定	発生時	※1
5	副電池警告 (LB2)	シンボル表示	シンボル表示 イベント設定	発生時	※1
6	IOボックス接続	何もしない	イベント設定	発生時	

※1 通知設定がされていても警告状態から復帰した場合、設定したイベントを消します。

11.1.8. 電源通知イベントクリア

電源通知モード設定で設定されたイベントが通知された後、そのイベントをクリアする場合に使用します。

通知されたイベントを本関数でクリアしない場合、キー待ちなどの動作が正常に行えません。

11.1.9. 電源OFFコマンド

本関数をアプリケーションプログラムから呼ぶことで電源OFF処理を行います。

11.1.10. IO BOX起動

IO BOX起動関数を使用することにより、IO BOX接続時に、自動的に本体の電源をONすることができます。

11.1.11. バイブレータ制御

バイブレータ制御関数を仕様することにより、本体内蔵のバイブレータを振動させることができます。

11.2.ファンクション詳細

ファンクション詳細を次ページより示します。

11.2.1. pwr_hold_apo

APO禁止／禁止解除の設定を行います。

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

パラメータ

OnOff

APO禁止設定

PWR_ON 禁止設定

PWR_OFF 禁止解除

BitPtrn

ビットパターン要因

FL_INV_APO_USR を設定して下さい。

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

11.2.2. pwr_off

電源をオフにします。

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

パラメータ

OnOff

電源オフ設定

PWR_ON	次回電源オン時、レジュームオンモードで起動します。
PWR_OFF	次回電源オン時、レジュームオフモードで起動します。

戻り値

関数結果

E_OK	正常終了
E_PRM	パラメータエラー

備考

11.2.3. pwr_IoboxBootMode

本体をIO BOXに乗せたときに、自動的に電源をオンするかどうかを設定します。

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

パラメータ

OnOff

電源オフ設定

PWR_ON	IO BOX に乗せた時、電源オンします。
PWR_OFF	IO BOX に載せた時、電源オンしません。

戻り値

関数結果

E_OK	正常終了
E_PRM	パラメータエラー

備考

11.2.4. pwr_vibrator

バイブレータの振動を制御します。

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

パラメータ

OnOff

電源オフ設定

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

関数結果

E_OK	正常終了
E_PRM	パラメータエラー

備考

12. 通知モード

12.1.通知モードの概念

通知モードは、LB、キーファンクション、タイマの発生に対して状態を確認する機能です。
通知モードを指定しない場合は、各イベントの処理はシステムで管理します。
通知モードはイベントの発生をフラグで通知するだけなので、イベントに対応する処理を行いたい場所に、イベントに対応するフラグを判断して対応処理に分岐する処理を組み込んで下さい。

12.2. 通知モード使用時に必要となる関数

(1)フラグ状態取得関数 (flg_sts)

通知モードで使用するフラグの状態を取得するための関数です。
本関数を使用することにより、通知イベントが発生したかの情報を取得することができます。

(2)フラグ状態クリア関数 (clr_flg)

ファンクション通知モードで使用するフラグの指定ビットをクリアするための関数です。
本関数は通知イベントが発生して対応する処理を実行するときに、処理を行う通知イベント状態をクリアするために使用します。

(3)フラグセット待ち関数 (wai_flg)

タイマ待ち等で、指定されたイベントが発生するまで処理を待ち状態にするために使用する関数です。

(4)通知モード設定／解除 (pwr_inhabit)

通知モードが指定された時は、APに通知します。
通知モードが設定されている時とされていない(通常処理)時では処理が異なります。
(詳細は、『ソフトウェア解説書』を参照して下さい)

(5)電源通知フラグ状態クリア関数 (pwr_inhabit_clr)

電源通知モードで使用するフラグの指定ビットをクリアするための関数です。
本関数はLB通知イベントが発生して対応する処理を実行するときに、処理を行う通知イベント状態をクリアするために使用します。

12.3. 通知モード使用例

12.3.1. LBに対する通知モードの場合

LBに対する通知モードを使用する場合、下記の点を注意して下さい。

- 複数のイベントに対して通知モードを使用する場合、必ず最優先でLB通知用の処理を行って下さい。
- 通知モードに対応する処理へ分岐する処理は、キー入力待ちの前後で行って下さい。
- 通知モードに対応する処理へ分岐する処理の間隔が長い場合、対応処理に分岐する処理を途中に組み込んで下さい。
- キー入力待ちの終了条件にLB発生時を加えて下さい。
- 通知モードに使用するフラグは FL_LB_INT_ID を使用して下さい。
- 通知イベントに対応する処理に分岐する場合は、必ず対応する通知イベントをクリアして下さい。
- 通知モードで設定するビットパターンは下記の値を組み合わせて使用して下さい。

表11.1 通知モードビットパターン

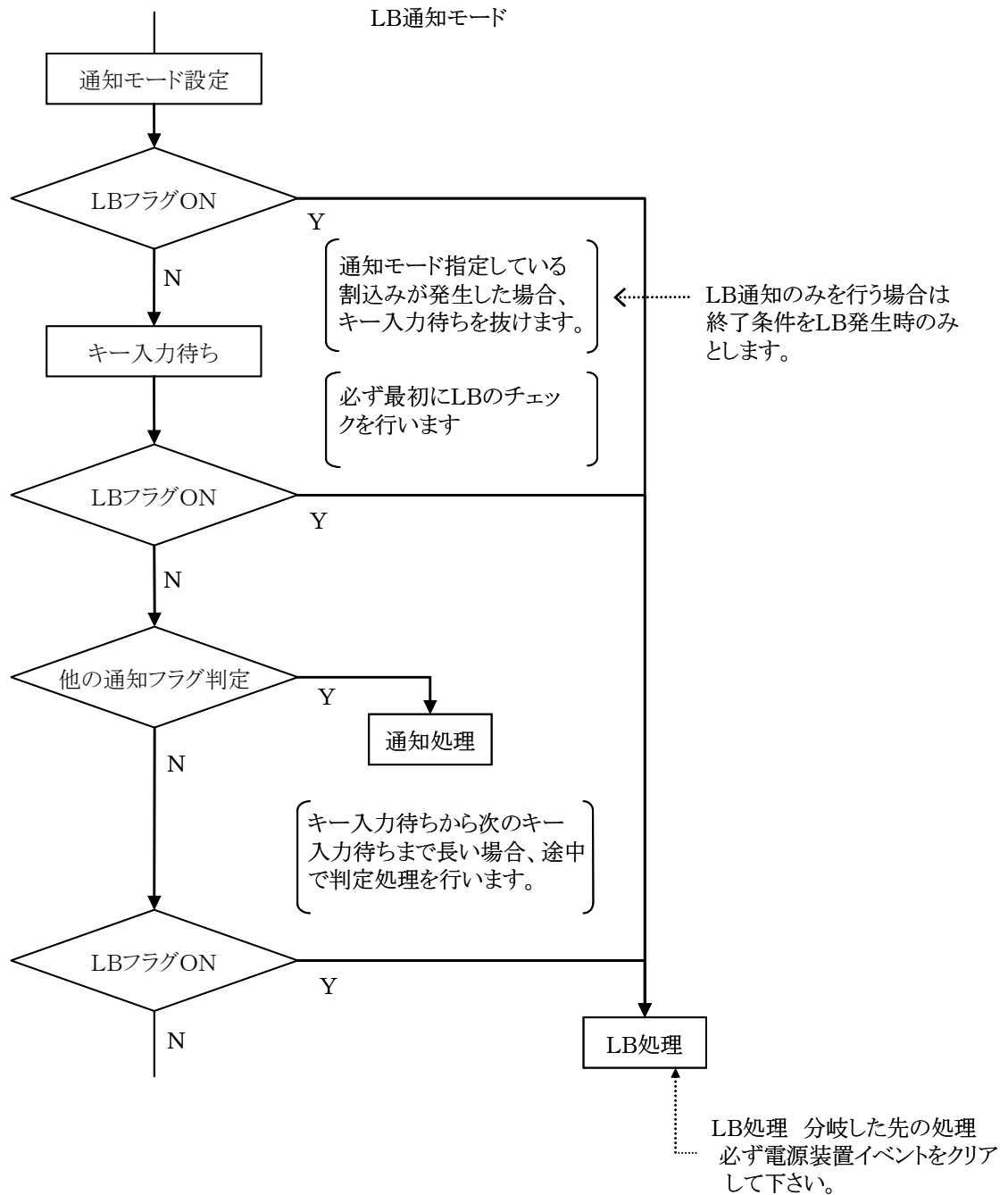
名称	説明
FL_LB_INT_LB0	LB0設定用ビットパターン
FL_LB_INT_LB1	LB1設定用ビットパターン
FL_LB_INT_LB2	LB2設定用ビットパターン
FL_LB_INT_LB4	LB4設定用ビットパターン
FL_LB_INT_LB5	LB5設定用ビットパターン

表11.2 通知モード以外のビットパターン

名称	説明
FL_SET_INT_IO	IOボックス検出用ビットパターン

以下にLB通知モードを使用する場合のチャートを記します。

LB通知モード



以下にLB0、1、2に対する通知モードの使用例を記します。

```

ER    err, retcd;
ID    dummy
UW    ptn, i;
KEY_INP keyinf;
.
.
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)
{
err = flg_sts( &dummy, &ptn, FL_LB_INT_ID );
if(ptn & FL_LB_INT_LB0)
{
pwr_inhabit_clr( FL_LB_INT_ID, FL_LB_INT_LB0 );
sub_lb0();

}
else if(ptn & FL_LB_INT_LB1)
{
pwr_inhabit_clr( FL_LB_INT_ID, FL_LB_INT_LB1 );
sub_lb1();
}
else if(ptn & FL_LB_INT_LB2)
{
pwr_inhabit_clr( FL_LB_INT_ID, FL_LB_INT_LB2 );
sub_lb2();
}
keyinf.ext = KEY_LB_EXT;
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);
}
pwr_inhabit( PWR_OFF, FL_LB_INT_ID,
            FL_LB_INT_LB0|FL_LB_INT_LB1|FL_LB_INT_LB2);
.
.
.
void sub_lb0( void )
{
.
.
return;
}

void sub_lb1( void )
{
.
.
return;
}

void sub_lb2( void )
{
.
.
return;
}

```

通知モード設定

フラグ状態取得

電源通知イベントのクリア後
各LBに対応した処理への分岐

リターン条件セット

(LBによる脱出)

通知モード解除

各LBに対応する処理

12.3.2. キーに対する通知モードの場合

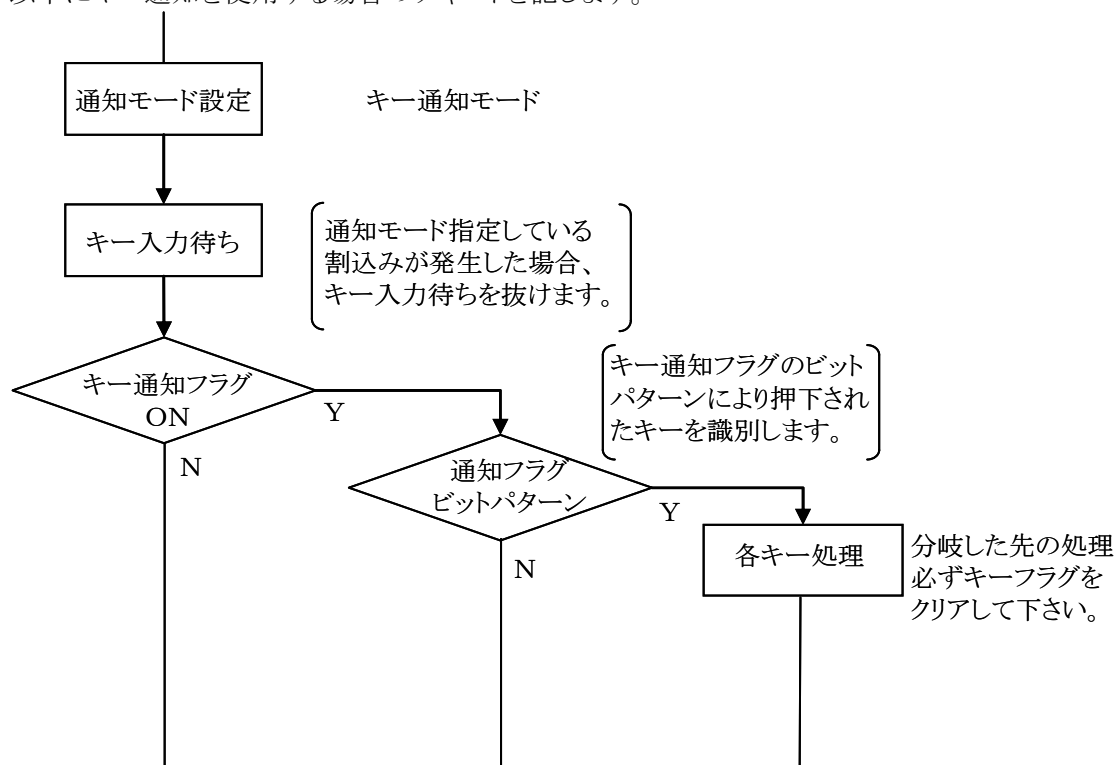
キーに対する通知モードを使用する場合、下記の点に注意して下さい。

- ・ 通知モードに対応する処理へ分岐する処理は、キー入力待ちの後で行って下さい。
- ・ 通知モードで押下されたキーの識別は、キー通知フラグのビットパターンで判別して下さい。
- ・ キー入力待ちの終了条件にイベント通知キー押下を加えて下さい。
- ・ 通知モードに使用するフラグは FL_FK_INT_ID を使用して下さい。
- ・ 通知イベントに対応する処理に分岐する場合は、必ず対応する通知イベントフラグをクリアして下さい。
- ・ 通知モードで設定するビットパターンは、下記の値を組み合わせて使用して下さい。

表11.3 ファンクション設定用ビットパターン

名称	説明
FL_FK_INT_FNC1	ファンクション 1 設定用ビットパターン
FL_FK_INT_FNC2	ファンクション 2 設定用ビットパターン
FL_FK_INT_FNC3	ファンクション 3 設定用ビットパターン
FL_FK_INT_FNC4	ファンクション 4 設定用ビットパターン
FL_FK_INT_FNC5	ファンクション 5 設定用ビットパターン
FL_FK_INT_FNC6	ファンクション 6 設定用ビットパターン
FL_FK_INT_FNC7	ファンクション 7 設定用ビットパターン
FL_FK_INT_FNC8	ファンクション 8 設定用ビットパターン

以下にキー通知を使用する場合のチャートを記します。



以下にファンクションキー1、2に対する通知モードの使用例を記します。

```

ER      err, retcd;
UW      ptn, i;
KEY_INP keyinf;
ID      dummy, fid;
      .
      .
      .
fid = FL_FK_INT_ID;
ptn = FL_FK_INT_FNC1;
err = key_fnc_mode( FNC_MODE_SET, FNC_1, &fid, &ptn );
fid = FL_FK_INT_ID;
ptn = FL_FK_INT_FNC2;
err = key_fnc_mode( FNC_MODE_SET, FNC_2, &fid, &ptn );
      .
      .
      .
keyinf.ext = KEY_INT_EXT;
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)
{
    err = flg_sts( &dummy, &ptn, FL_FK_INT_ID );
    if(ptn & FL_FK_INT_FNC1)
    {
        clr_flg( FL_FK_INT_ID, ~FL_FK_INT_FNC1 );
        sub_fnc1();
    }else if(ptn & FL_FK_INT_FNC2)
    {
        clr_flg( FL_FK_INT_ID, ~FL_FK_INT_FNC2 );
        sub_fnc2();
    }
}
fid=FL_FK_INT_ID;
ptn= FL_FK_INT_FNC1;
err= key_fnc_mode( FNC_MODE_CLR, FNC_1, &fid, &ptn );
fid= FL_FK_INT_ID;
ptn= FL_FK_INT_FNC2;
err = key_fnc_mode( FNC_MODE_CLR, FNC_2, &fid, &ptn );
      .
      .
      .
void sub_fnc1( void )
{
      .
      .
      return;
}
void sub_fnc2( void )
{
      .
      .
      return;
}

```

通知モード設定

リターン条件設定

(KEYによる脱出)

通知フラグ状態をクリア後
各KEYに対応した処理への
分岐

通知モード解除

各KEYに対応する処理

12.4.ファンクション詳細

ファンクション詳細を次ページより示します。

12.4.1. flg_sts

指定 ID の通知フラグの各種状態を参照し、対象フラグの現在の値を返します。

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

パラメータ

p_flgpid

ワーク領域の先頭アドレス

p_flgptn

フラグのビットパターンを返す領域の先頭アドレス

flgid

フラグID

戻り値

関数結果

E_OK

正常終了

E_NOEXS

フラグ ID 範囲外 / 予約 ID

E_ILADR

不正アドレス

補足

12.4.2. clr_flg

指定 ID の通知フラグの指定ビットをクリアします。

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

パラメータ

flgid
フラグID

setptn
クリアするビットパターン

戻り値

関数結果	
E_OK	正常終了
E_NOEXS	フラグ ID 範囲外 / 予約 ID

補足

12.4.3. wai_flg

指定 ID のフラグがセットされるのを、指定待ち条件に従って待ちます。

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

パラメータ

p_flgptn

待ち解除時のビットパターンを返す領域の先頭アドレス

flgid

フラグID

waipth

待ちビットパターン

wfmode

待ちモード

$wfmode = (TWF_ANDW \ || \ TW_ORW) \ |[TWF_CLF]$

TWF_ANDW AND 待ち

TWF_ORW OR 待ち

TWF_CLR クリア指定（条件が満足されてタスク待ち解除になるとイベントフラグの全部のビットが0にクリアされます）

戻り値

関数結果

E_OK 正常終了

E_RLWAI 待ち状態強制解除（本システムでは起り得ません）

E_QOVR キューイングのオーバフロー（本システムでは起り得ません）

E_CTX コンテキストエラー（本システムでは起り得ません）

E_NOEXS フラグ ID 範囲外／予約 ID（本システムでは起り得ません）

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

補足

12.4.4. pwr_inhabit

通知モードの設定および解除を行います。

通知要因が発生した時、該当ビットがセットされている時のみ通知されます。

2回目以降本関数をコールする場合、異なるイベントフラグを指定するとイベントフラグ名は変更されます。

```
ER pwr_inhabit(  
    UH OnOff,  
    ID EventFlg,  
    UW BitPtrn  
);
```

パラメータ

OnOff

通知モード

PWR_ON	通知モード設定
PWR_OFF	通知モード解除

EventFlg

イベントフラグID

BitPtrn

ビットパターン

FL_LB_INT_LB0 - FL_LB_INT_LB5	LB検出
FL_SET_INT_IO	I/O ボックス検出

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

12.4.5. pwr_inhabit_clr

電源通知イベントフラグをクリアします。

```
ER pwr_inhabit_clr(  
    ID EventFlg,  
    UW BitPtrn  
);
```

パラメータ

EventFlg

イベントフラグID

BitPtrn

ビットパターン

FL_LB_INT_LB0 - FL_LB_INT_LB5

LB検出

FL_SET_INT_IO

IO ボックス検出

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

13. 共通関数

13.1. 機能

共通関数は、アプリケーションの終了／各種設定を以下の機能によりサポートします。

13.1.1. ABORT処理

本関数が CALL された場合、以下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、以下の状態になります。

- 全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。
- 次回電源オン時は、レジュームOFFモードになります。
- 全てのファイルをクローズします。
- LCD 以外の全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。

13.1.2. EXIT処理

以下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。
- 全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。
- 全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。

13.1.3. 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。

13.1.4. 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。

13.2.ファンクション詳細

ファンクション詳細を次ページより示します。

13.2.1. abort

以下の処理を行ない、アボート画面を表示し電源キー押下待ちになります。(DT700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)
- 《C言語インタフェース》

```
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

補足

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。

13.2.2. exit

以下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。
- 《C言語インタフェース》

```
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

補足

13.2.3. 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(void);
```

パラメータ

なし

戻り値

なし

補足

13.2.4. 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。
《C言語インタフェース》

```
wkup_calib(void)
```

パラメータ

なし

戻り値

なし

補足

14. プリンタ制御関数

14.1. 対象プリンタ

以下の2種類のプリンタを制御するための関数をサポートします。

- ・プチラバン(SATO)
- ・B-SP2D(東芝テック)

14.2. 使用方法

拡張機能なので、SDK 内にあるライブラリとヘッダをリンクしてください。

Prn_bt.h	プリンタ制御関数定義ヘッダ
Prn_bt.obj	プリンタ制御関数ライブラリ

14.3. 通信インターフェース

通信インターフェースは、Bluetooth です。

通信部は、Bluetooth ライブラリを使用します。

14.4. コマンドパケット／ステータスパケット仕様

詳細な仕様は、各々のプリンタの解説書をご覧ください。

14.5. 機能

14.5.1. 通信の接続準備

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

(1) 周囲の Bluetooth 機器の検索および設定

以下の手順で処理を行ってください。

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

※但し、B-SP2D は電源 ON 直後から1分間の間しか Inquiry を受け付けません。

※Inquiry で探索できるのは最大9件なので、周りにたくさん Bluetooth 機器が有る場合は対象の機器が出てこない可能性があります。

(2) 固定の MAC アドレスで設定

以下の手順で処理を行ってください。

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

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

14.5.2. 通信のオープン・クローズ

(1) オープン

Bluetooth 接続処理の中で、BT_Start()と BT_Open()に時間がかかるため、関数の構成としてこれら2つの関数は、分けて実行するようにしています。オープン関数の中には、BT_Open()が含まれています。従って、オープン関数を呼ぶ前には、必ず BT_Start() (最短 1.5 秒) を呼ぶ必要があります。

オープン関数の中では、以下の処理を行なっています。

- ①通信する Bluetooth 機器の情報をファイルから取得 (BT_LoadDevInfo)
- ②通信する Bluetooth 機器を選択 (BT_SelectDev)
- ③Bluetooth 機器との接続 (BT_Open 最短 3 秒)

(2) クローズ

オープン処理で BT_Start()と BT_Open()を分離しているので、クローズ処理もそれに対応して、BT_Close()と BT_Stop()を分離しています。完全に終了させるためには、クローズ後に BT_Stop() を呼んでください。

クローズ関数の中では、以下の処理を行なっています。

- ①Bluetooth 機器との切断 (BT_Close)

14.5.3. 通信の送信・受信

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

(1) データ送信

指定された送信データをプリンタに送ります。

送信関数の中では、以下の処理を行なっています。

- ①指定文字数分のデータを送信 (BT_Write)

(2) ステータス受信

STX で始まる文字列を指定文字数分 (STX を含む) 受信します。

受信関数の中では、以下の処理を行なっています。

- ①指定文字数分のデータを受信 (BT_Read)
- ②受信タイムアウトをチェック

14.5.4. エラー処理

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

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

※1:リトライしてもつながらない場合は、保守に連絡してください。

14.6.ファンクション詳細

ファンクション詳細を次ページより示します。

14.6.1. Bluetooth のオープン

Bluetooth を占有します。
本関数を呼ぶ前に必ず BT_Start を呼んでください。

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

パラメータ

**buff*

受信バッファアドレス(最低でも52byteのエリアが必要です)

tout

接続待ちタイムアウト時間(1~3600 秒)

len

受信バッファレングス

**fname*

デバイス情報格納ファイル名

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
E_PRN_PON	オープン済み

補足

戻り値に E_NG が返って来た場合、BT_Err_Get で詳細なエラーステータスを確認してください。

14.6.2. Bluetooth のクローズ

Bluetooth の占有を解除します。
完全に終了させるためには、本関数呼出し後、BT_Stop を呼んでください。

```
ER prn_BTinf_close (void);
```

パラメータ

なし

戻り値

関数結果

E_OK	正常終了
E_NG	異常終了
E_PRN_POFF	未オープン

補足

戻り値に E_NG が返って来た場合、BT_Err_Get で詳細なエラーステータスを確認してください。

14.6.3. Bluetooth でのコマンド送信

プリンタに対して指定レングス分各種コマンドを送信します。

```
ER prn_BTinf_send (  
    B *command_ptr  
    UH len  
);
```

パラメータ

** command_ptr*
送信するコマンドのアドレス

len
コマンドレングス

戻り値

関数結果

E_OK	正常終了
E_NG	異常終了
E_PRN_POFF	未オープン

補足

戻り値に E_NG が返って来た場合、BT_Err_Get で詳細なエラーステータスを確認してください。

14.6.4. Bluetooth でのステータス受信

プリンタから送られてくる、プリンタステータスを受信するために使用します。
prn_BTinf_open()で指定したバッファにデータを受信します。
指定するレングスは、STX からの受信予定データ数を指定します。

```
ER prn_BTinf_send (  
    UH len,  
    H tout  
);
```

パラメータ

len

受信予定データ数 (STX を含むデータ数)

tout

受信待ちタイムアウト時間 (1~3600 秒)

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
E_PRN_POFF	未オープン
E_PRN_TIMEOUT	受信タイムアウト

補足

戻り値に E_NG が返って来た場合、BT_Err_Get で詳細なエラーステータスを確認してください。

15. 参考資料

15.1. 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	カーソルタイプ設定	○	○	○	○	○
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	ユーザフォントファイル登録	○	○	○	○	○
lcd_romfont	ROMフォント設定	○	○	○	○	○
lcd_userstr	ユーザ文字列表示	○	○	○	○	○
lcd_active_set	ユーザ描画ページ設定			○		
lcd_visual_set	ユーザ表示ページ設定			○		
lcd_active_get	ユーザ描画ページ読出し			○		
lcd_visual_get	ユーザ表示ページ読出し			○		
lcd_cls_page	指定ページクリア			○		
lcd_grchar	グラフィック文字表示			○		
lcd_box	BOX 描画/削除			○		
lcd_el	EL バックライトの制御			○	○	○

(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	キーハットの状態チェック	○	○	○	○	○
key_clear	キーハットの状態クリア	○	○	○	○	○
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	キーパッドファイル登録			○		
key_touch	ユーザタッチキー設定削除			○		
key_point	タッチ座標取得			○		
key_pad	キーパッド切替/状態取得			○		
key_pad_entry	キーパッド遷移設定/状態取得			○		

(4) 通信部

関数名	機能	DT700	DT750	DT800	DT900	DT930
c_open	COM のオープン	○	○	○	○	○
c_close	COM のクローズ	○	○	○	○	○
c_status	COM ステータスのリード	○	○	○	○	○
c_hold	COM の占有	○	○	○	○	○
c_chkopen	COM のオープンチェック	○	○	○	○	○
c_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	メモブロック送信	○	○			
c_mdin	メモブロック受信	○	○			
c_out	1 文字送信	○	○	○	○	○
c_break	ブレイク信号の制御	○	○	○	○	○
c_txx	送受信の有効/無効	○	○	○	○	○
c_jobox	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ド	○	○	○	○	○
c_errbfring	リターンコードバッファリング制御の設定	○	○	○	○	○
c_rdersts	エラーステータスのリード	○	○	○	○	○
c_chghdr	受信ハンドラ切替え	○	○	○	○	○
c_cimode	CI 信号立ち上げモード設定				○	
c_brkevent	ブレイク要因の設定				○	○

(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	電源通知イベントのクリア	○	○	○	○	○
pwr_hold_apo	APO 禁止設定	○	○	○	○	○
pwr_off	電源オフ	○	○	○	○	○
pwr_loboxBootMode	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	通知フラグ状態取得	○	○	○	○	○
clr_flg	通知フラグ状態クリア	○	○	○	○	○
wai_flg	フラグセット待ち	○	○	○	○	○

(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 文字リード	○	○	○	○	○
OBR_gets	OBR データ文字列リード	○	○	○	○	○
OBR_stat	OBR ハッファステータスチェック	○	○	○	○	○
OBR_flush	OBR ハッファのクリア	○	○	○	○	○
OBR_moderd	OBR 動作モードの取得	○	○	○	○	○
OBR_modewt	OBR 動作モード設定	○	○	○	○	○
OBR_chgbuf	OBR ハッファの切替え	○	○	○	○	○
OBR_gain	発光ゲイン切替え		○	○		
OBR_trigmode	トリガ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	マージンチェック倍率モード取得					○
OBR_setmargincheck	マージ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	○	○	○	○	○
チェックデジットの実行	○	○	○	○	○
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)	○	○	○	○	○
ゲインコントロール	○	○	×	×	×
プサージ制御	○	○	○	○	○
LED 制御	○	○	○	○	○
出力ハッファ	○	○	○	○	○
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	タイマー 1 登録	○	○	○	○	○
s_timerend	タイマー 1 削除	○	○	○	○	○
s_settimer2	タイマー 2 登録	○	○	○	○	○
s_timerend2	タイマー 2 削除	○	○	○	○	○
s_beep	エラービープ音	○	○	○	○	○
s_beep2	エラービープ音2(赤LED点灯)		○	○		
s_sound	サウンド音 1	○	○	○	○	○
s_dateget	日付の取得	○	○	○	○	○
s_dateset	日付の設定	○	○	○	○	○
s_timeget	時間の取得	○	○	○	○	○
s_timeset	時間の設定	○	○	○	○	○

(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(注 2)	APロード&実行			RAM ○	○	○

注 2 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)		○	予約領域	○	○
	パリティ(IR)	○	○	予約領域	○	○
	STOP(IR)	○	○	予約領域	○	○
	速度(RF/シリアル)	○	○	予約領域	○	
	データ(RF/シリアル)		○	予約領域	○	
	パリティ(RF/シリアル)	○	○	予約領域	○	
	STOP(RF/シリアル)	○	○	予約領域	○	
	速度(10P)	○	○	○		
	データ(10P)		○	予約領域		
	パリティ(10P)	○	○	予約領域		
	STOP(10P)	○	○	予約領域		
	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	○	○	予約領域		
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)			○		
メモリ関連	アプリケーション SIZE			○	○	○
	ファイルモード*	FORMAT			○	○
	シリアル NO				○	○
	水平パリティ				○	○
リンクタイムアウト				○	○	

(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	注1	○		○	○
dat_fsize_chg	ファイルサイズ変更		○			
open	ファイルオープン	○	○	○	○	○
close	ファイルクローズ	○	○	○	○	○
read	ファイルのリード	○	○	○	○	○
write	ファイルのライト	○	○	○	○	○
lseek	ファイルリード/ライト位置の設定	○	○	○	○	○
sbrk	メモリ領域の割当て	○	○	○	○	○
fil_mkdir	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	ファイル全パス名の取得				○	○

注1 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	画面表示メッセージ送信	○	○	○	○	○
cu_end	通信中断	○	○		○	○
cu_close	回線クロス	○	○	○	○	○
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	ダウンロード動作設定		○			
cu_fileAdd	ファイルの追加			○	○	○
cu_idle	IDLE 遷移			○	○	○
cu_cmdRecv	コマンド受信待ち			○	○	○
cu_fileDelete	ファイル削除			○	○	○
cu_fileMove	ファイル移動			○	○	○
cu_makeDir	ディレクトリ作成			○	○	○
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	ディスク情報の取得			○	○	○
cu_beep	ブザー鳴動			○	○	○
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	画面表示メッセージ送信				○	○
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	データ読み込み			○	○	○
Ir_Write	データ書込み			○	○	○
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	読み込み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	ファイルオープン		○			
Mbu_close	ファイルクローズ		○			
Mbu_write	ファイル書込み		○			
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	赤外のコマンド送信		○			
Prn_inf_status	赤外のステータスリード		○			
Prn_wir_open	無線オープン		○			
Prn_wir_close	無線クローズ		○			
Prn_wir_send	無線のコマンド送信		○			
Prn_pktlen	無線のバケット長登録		○			
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 バックアップ ^o		○			
	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	○	○	○	○	○
	同朋インストール		○	○	○	○
	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 ^o			○	○	○
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			○	○	○
	メモサイズ ^o 変更			○	○	○
	ファイルモード ^o				○	○
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*	フロントモード*	○	○	○	○	○
	メッセージ*	○	○	○	○	○
フロントモード*	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*	読取り回数	○	○	○	○	○
	照合回数	○	○	○	○	○
	タイムアウト	○	○	○	○	○
	読取り禁止時間		○			
	キャリブレーション				○	○
ID セット	機器 ID	○	○	○	○	○
	代理店 ID			○	○	○

DT-930
Cライブラリ解説書
平成18年2月 Rev1.0発行

カシオ計算機株式会社