DT-900
C ライブラリ解説書

Rev.2.00

＜メモリ拡張機対応版＞

カシオ計算機株式会社

目 次

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。

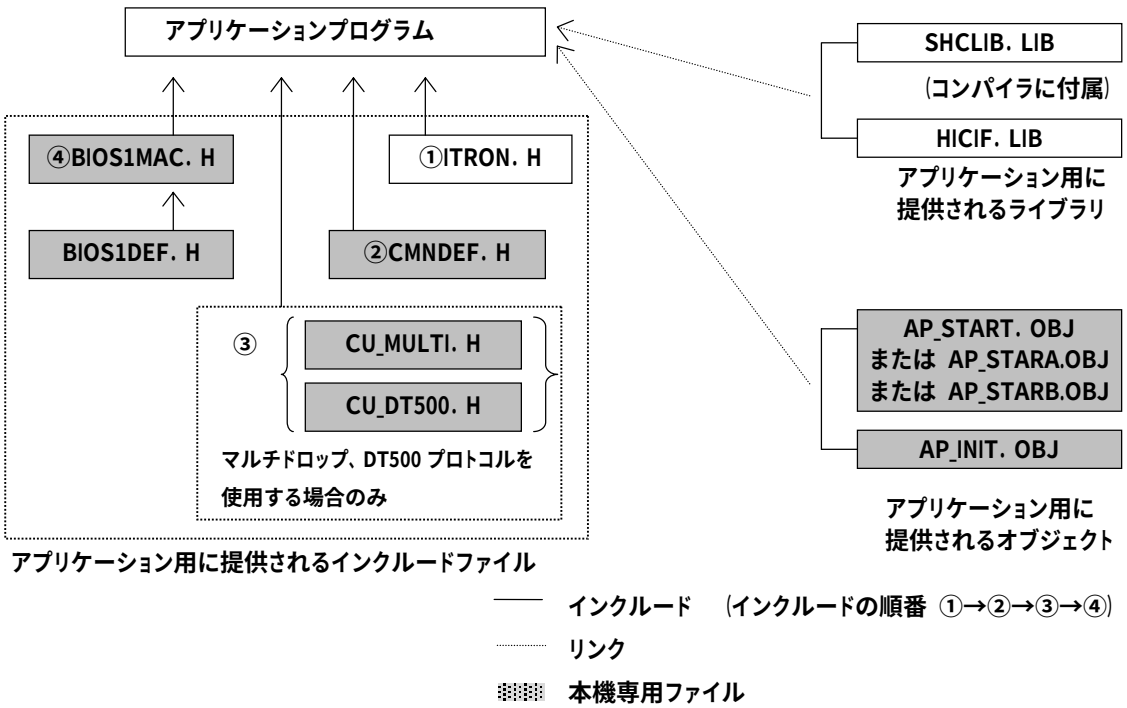
※ アプリケーションは提供された関数群 (外部シンボル) の定義ファイルを用いることにより、単独でコンパイル／リンクします。(OS／システムの実体とはリンクしません)

(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)

： 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	iscntrl 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	浮動小数点数のラディアン値の余弦計算
21	sin 関数	浮動小数点数のラディアン値の正弦計算
22	tan 関数	浮動小数点数のラディアン値の正接計算
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のべき乗の計算
29	log 関数	浮動小数点数と2のべき乗の計算
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	浮動小数点数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	sscanf 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 doublu 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	ページ
1	メモリ管理関数		
	dat_mem_size	メモリ領域の空きサイズの取得	37
2	ファイル管理関数		
	fil_mkdir	ディレクトリの作成（FATファイルモード）	38
	fil_rmdir	ディレクトリの削除（FATファイルモード）	39
	fil_remove	ファイルの削除	40
	fil_rename	ファイル名の変更／移動	41
	fil_fstat	ファイルの日時･サイズ･属性の取得	42
	fil_chsize	ファイルサイズの変更	43
	fil_getsize	ファイル領域空きサイズの取得	44
	fil_findfirst	ファイル名の取得	45
	fil_findnext	ファイル名の取得（次候補）	46
	fil_filesize	ファイルの個数と総サイズの取得	47
	fil_filefind	ファイル全パス名の取得	48
	dat_fdir	ファイル格納情報の取得(DT700 互換ﾓｰﾄﾞ)	33
	dat_fsize	ファイル空き領域サイズの取得(DT700 互換ﾓｰﾄﾞ)	35
	dat_fdel	ファイルの削除(DT700 互換ﾓｰﾄﾞ)	36
	dat_frname	ファイル名変更(DT700 互換ﾓｰﾄﾞ)	49
	dat_F_Search	ファイルデータの検索(DT700 互換ﾓｰﾄﾞ)	50
3	システムデータ管理関数		
	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	23
	dat_dealer_chk	代理店ＩＤのチェック	24
	dat_Aload	ＡＰロード＆実行	25
4	低水準インタフェース関数		
	open	ファイルオープン	26
	close	ファイルクローズ	28
	read	ファイルのリード	29
	write	ファイルのライト	30
	lseek	ファイルリード／ライト位置の設定	31
	sbrk	メモリ領域の割当て	32

1.3.2. 表示部関数

NO	関 数 名	機　　能	ページ
1	lcd_cls	画面クリア	72
2	lcd_csr_set	カーソルタイプ設定	73
3	lcd_csr_put	カーソル位置設定	74
4	lcd_csr_get	カーソル位置読出し	75
5	lcd_char	１文字表示	76
6	lcd_string	文字列表示	77
7	lcd_string2	文字列表示２（スクロール抑制）	78
8	lcd_userstr	ユーザ文字列表示	79
9	lcd_line	直線描画	80
１０	lcd_gaiji	外字フォント登録	81
１１	lcd_usrfont	ユーザーフォントファイル登録	82
１２	lcd_romfont	ＲＯＭフォント設定	83
１３	lcd_led	ＬＥＤの制御	84
１４	lcd_el	ＥＬバックライトの制御	85

1.3.3.　キー部関数

1.3.4.　ＯＢＲ部関数

1.3.5. 通信部関数

NO	関 数 名	機　　能	ページ
1	COM管理機能		
	c_open	COMオープン	167
	c_close	COMクローズ	169
	c_status	COMステータスのリード	170
	c_hold	COMの占有	172
	c_chkopen	COMのオープンチェック	173
	c_wp	WakeUp 信号の設定	193
2	送受信機能		
	c_dout	n文字送信	174
	c_din	1文字受信	175
	c_tmdin	タイムアウト監視受信	176
	c_out	1文字送信	177
	c_break	ブレーク信号の制御	178
	c_txrx	送受信の有効／無効	179
	c_cimode	CI信号立ち上げモード設定	187
	c_timer	DR／CS／CDタイムアウト監視値の設定	188
	c_er	ER 信号の制御	189
	c_rs	RS信号の制御	190
	c_errs	ER／RS信号の制御	191
	c_iobox	IOボックス送信設定	180
	c_irout	IOボックス送信	181
3	受信バッファ管理機能		
	c_flush	受信バッファのクリア	182
	c_bfsts	受信バッファステータスのリード	183
4	通信補助機能		
	c_errbfring	エラーコードバッファリング制御の設定	184
	c_rderrsts	エラーステータスのリード	185
	c_chghdr	受信ハンドラ切替え	186
	c_brkevent	ブレイク要因の設定	192

1.3.6. IrDA 部関数

NO	関 数 名	機　　能	ページ
1	Ir_Open	IrCOMM オープン	200
2	Ir_Close	IrCOMM クローズ	201
3	Ir_Read	データ読込み	202
4	Ir_Write	データ書込み	203
5	Ir_QueryTx	送信データ数問合せ	204
6	Ir_QueryRx	受信データ数問合せ	205
7	Ir_EROn	ER信号ON	206
8	Ir_EROff	ER信号OFF	207
9	Ir_RSOn	RS信号ON	208
10	Ir_RSOff	RS信号OFF	209
11	Ir_BreakOn	ブレーク送信ON	210
12	Ir_BreakOff	ブレーク送信OFF	211
13	Ir_CheckCD	CD検査	212
14	Ir_CheckDR	DR検査	213
15	Ir_CheckCS	CS検査	214
16	Ir_CheckCI	CI検査	215
17	Ir_CheckBreak	BREAK検査	216
18	Ir_Err_Get	エラー値取得	217
19	Ir_State_Set	通信状態設定	225
20	Ir_SetPortConfig	自局能力設定	227
21	Ir_Init	IrCOMM 強制終了	229

1.3.7.　通信ユーティリティ部関数

1.3.8.　タイマー部関数

NO	関 数 名	機　　能	ページ
1	タイマ登録		
	s_settimer	タイマ1登録	316
	s_timerend	タイマ1削除	317
	s_settimer2	タイマ2登録	318
	s_timerend2	タイマ2削除	319
2	音発生		
	s_beep	エラービープ音	320
	s_sound	サウンド音	321
3	日時設定		
	s_dateget	日付の取得	323
	s_dateset	日付の設定	322
	s_timeget	時刻の取得	325
	s_timeset	時刻の設定	324

1.3.9.　電源部関数

NO	関 数 名	機　　能	ページ
1	pwr_hold_apo	APO禁止設定	329
2	pwr_off	電源オフ	330

1.3.10. 通知モード部関数

NO	関 数 名	機　　能	ページ
1	flg_sts	通知フラグ状態取得	338
2	clr_flg	通知フラグ状態クリア	339
3	wai_flg	フラグセット待ち	340
4	pwr_inhabit	通知モード設定	341
5	pwr_inhabit_clr	電源通知イベントのクリア	342

1.3.11. 共通関数

NO	関 数 名	機　　能	ページ
1	abort	ABORT処理	345
2	exit	EXIT処理	346
3	wkup_cost	動作環境メニューの起動	347
4	wkup_calib	OBRキャリブレーション処理起動	348

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 ;       /*  ﾃﾞｰﾀﾀｲﾌﾟが一定しない (8ﾋﾞｯﾄｻｲｽﾞ)            */
typedef    short            VH ;       /*  ﾃﾞｰﾀﾀｲﾌﾟが一定しない (16ﾋﾞｯﾄｻｲｽﾞ)           */
typedef    long             VW ;       /*  ﾃﾞｰﾀﾀｲﾌﾟが一定しない (32ﾋﾞｯﾄｻｲｽﾞ)           */
typedef    void             *VP;       /*  ﾃﾞｰﾀﾀｲﾌﾟが一定しないものへのﾎﾟｲﾝﾀ            */
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クリアーします
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、デイスクファイル等）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 システムデーター覧

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)	○	○
		ﾚｼﾞｭｰﾑ	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	ﾊﾞｰｺｰﾄﾞ連続読取回数：1～9(回)	○	○
		照合回数	読取ｺｰﾄﾞ照合回数：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	ﾏﾙﾁﾄﾞﾛｯﾌﾟ/FLINK/DT500	○	○
		通信PORT	各種通信におけるPORT	○	○
	ﾏﾙﾁﾄﾞﾛｯﾌﾟ 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)	○	○
		ﾃﾞｰﾀ(IrDA/ｶｼｵIRｲﾝﾀﾌｪｰｽ)	CHAR_8/7	○	○
		ﾊﾟﾘﾃｨ(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)	○	○
		ﾃﾞｰﾀ(ｼﾘｱﾙｲﾝﾀﾌｪｰｽ)	CHAR_8/7	○	○
		ﾊﾟﾘﾃｨ(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)	○	○
		ﾃﾞｰﾀ(PHSｲﾝﾀﾌｪｰｽ)	CHAR_8/7	○	○
		ﾊﾟﾘﾃｨ(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ﾊﾞｰｼﾞｮﾝ	OSﾊﾞｰｼﾞｮﾝ(6桁)	―	―
		PATCHﾊﾞｰｼﾞｮﾝ	PATCHﾊﾞｰｼﾞｮﾝ(6桁)	―	―
		機器種別	MODEL情報	IOﾎﾟｰﾄ反映	―
ﾌﾟﾛﾄｺﾙ	ﾏﾙﾁﾄﾞﾛｯﾌﾟ	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	○	○
		水平ﾊﾟﾘﾃｨ	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ﾄﾞﾗｲﾌﾞ情報反映	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(32bit)	0～59	10	
		ABO	LONG(32bit)	10～59	15	
		ﾚｼﾞｭｰﾑ	LONG(32bit)	0 or 1	1	ON:1　OFF:0
KEY		ｸﾘｯｸ音	LONG(32bit)	0 or 1	1	ON:1　OFF:0
表示		ﾌｫﾝﾄMODE	LONG(32bit)	0 or 1 or 2	1	6DOT:0　8DOT:1 10DOT:2
		ﾌｫﾝﾄ種別	LONG(32bit)	0 or 1	0	NORMAL:0　BOLD:1
		日本語/英語	LONG(32bit)	0 or 1	0	日本語:0　英語:1
		ｺﾝﾄﾗｽﾄ設定値	LONG(32bit)	0～15	7	
		ｺﾝﾄﾗｽﾄ手動差分	LONG(32bit)	-7～7	0	
OBR		読取り回数	LONG(32bit)	1～9	1	
		照合回数	LONG(32bit)	1～9	3	
		ｽｷｬﾝ時間	LONG(32bit)	1～9	4	
通信	共通	プロトコル	LONG(32bit)	0 or 1 or 2	1	ﾏﾙﾁ:0　FLINK:1　DT500:2
		通信PORT	LONG(32bit)	0 or 1or 2	0	IR:0　14P:1　18P:2
	ﾏﾙﾁﾄﾞﾛｯﾌﾟ または FLINK または DT500	速度 (ｶｼｵIRｲﾝﾀﾌｪｰｽ/IrDA)	LONG(32bit)	2400～115200	B_115200	B_2400～B_115200　DEFINE使用
		ﾃﾞｰﾀ (ｶｼｵIRｲﾝﾀﾌｪｰｽ/IrDA)	LONG(32bit)	CHAR_8/7	CHAR_8	
		ﾊﾟﾘﾃｨ (ｶｼｵIRｲﾝﾀﾌｪｰｽ/IrDA)	LONG(32bit)	EVN/ODD/NON	PARI_NON	
		STOP (ｶｼｵIRｲﾝﾀﾌｪｰｽ/IrDA)	LONG(32bit)	STOP_1/2	STOP_1	
		速度 (ｼﾘｱﾙｲﾝﾀﾌｪｰｽ)	LONG(32bit)	1200～115200	B_115200	B_1200～B_115200　DEFINE使用
		ﾃﾞｰﾀ (ｼﾘｱﾙｲﾝﾀﾌｪｰｽ)	LONG(32bit)	CHAR_8/7	CHAR_8	
		ﾊﾟﾘﾃｨ (ｼﾘｱﾙｲﾝﾀﾌｪｰｽ)	LONG(32bit)	EVN/ODD/NON	PARI_NON	
		STOP (ｼﾘｱﾙｲﾝﾀﾌｪｰｽ)	LONG(32bit)	STOP_1/2	STOP_1	
		速度(PHSｲﾝﾀﾌｪｰｽ)	LONG(32bit)	1200～38400	B_38400	B_1200～B_38400　DEFINE使用
		ﾃﾞｰﾀ(PHSｲﾝﾀﾌｪｰｽ)	LONG(32bit)	CHAR_8/7	CHAR_8	
		ﾊﾟﾘﾃｨ(PHSｲﾝﾀﾌｪｰｽ)	LONG(32bit)	EVN/ODD/NON	PARI_NON	
		STOP(PHSｲﾝﾀﾌｪｰｽ)	LONG(32bit)	STOP_1/2	STOP_1	
ﾀｲﾏ		音量	LONG(32bit)	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ﾊﾞｰｼﾞｮﾝ	BYTE x 6	6桁の英数字	OS依存	
		PATCHﾊﾞｰｼﾞｮﾝ	BYTE x 6	6桁の英数字	OS依存	ex: V1.00A
		機器種別	LONG(32bit)		機種依存	
ﾌﾟﾛﾄｺﾙ	ﾏﾙﾁﾄﾞﾛｯﾌﾟ	受信ﾀｲﾑｱｳﾄ	LONG(32bit)	0～99	3	
		ﾘﾄﾗｲ回数	LONG(32bit)	0～99	3	
		ﾘﾝｸﾀｲﾑｱｳﾄ	LONG(32bit)	0～9990	30	
	FLINK	ｾｯｼｮﾝ確立ﾀｲﾑｱｳﾄ	LONG(32bit)	0～3600	1800	
		受信ﾀｲﾑｱｳﾄ	LONG(32bit)	0～600	300	
		ｾｯｼｮﾝ終了ﾀｲﾑｱｳﾄ	LONG(32bit)	0～600	10	
	DT500	シリアルNO	LONG(32bit)	0 or 1	0	ON:1　OFF:0
		水平ﾊﾟﾘﾃｨ	LONG(32bit)	0 or 1	0	ON:1　OFF:0
		リンクタイムアウト	LONG(32bit)	0～240	0	
サイズ		AP領域ｻｲｽﾞ	LONG(32bit)	128～992	336	KB単位
ファイルモード		FORMAT	LONG(32bit)	0 or 1		0:FATﾓｰﾄﾞ 1:DT700ﾓｰﾄﾞ

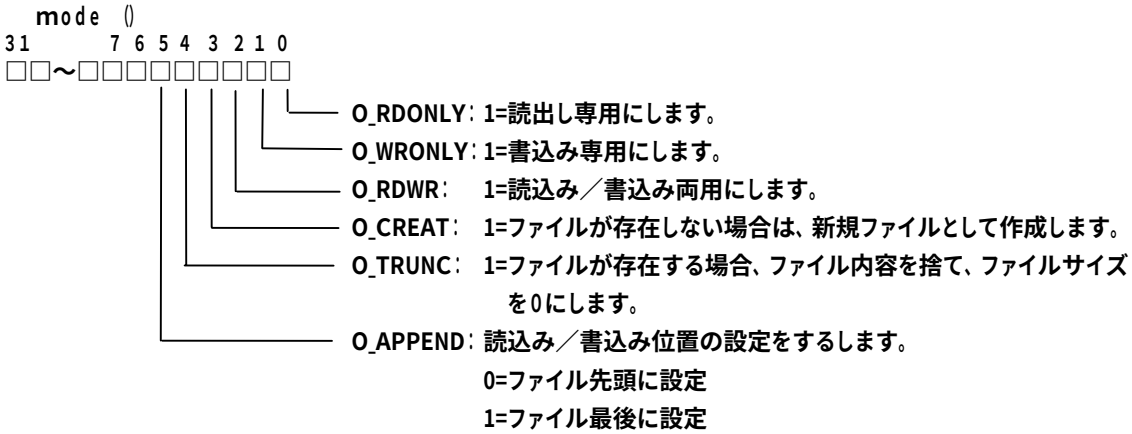
2.1.4.　低水準インタフェース

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。



(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。

①DT700 互換ファイルシステムモード

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
同時ｵｰﾌﾟﾝ数	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

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）
同時ｵｰﾌﾟﾝ数	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プタ	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)

(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ﾄﾞ	
		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_frname	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)

(4) バーコードレベル設定ファイル

ファイル形式　　：DOS ファイル
ファイル名　　：CONFIG.OBR
ファイルサイズ　　：512バイト以下
ファイル内容　　：詳細は 17 ページ参照

CONFIG.HTS形式

項目			位置	ｻｲｽﾞ	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
		ﾚｼﾞｭｰﾑ	+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	ﾌﾟﾛﾄｺﾙ	+32	2	00:ﾏﾙﾁﾄﾞﾛｯﾌﾟ / 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
		ﾃﾞｰﾀ(IR)	+38	2	07:7bit / 08:8bit	08
		ﾊﾟﾘﾃｨ(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
		ﾃﾞｰﾀ (ｼﾘｱﾙｲﾝﾀﾌｪｰｽ)	+46	2	07:7bit / 08:8bit	08
		ﾊﾟﾘﾃｨ (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
		ﾃﾞｰﾀ (PHS ｲﾝﾀﾌｪｰｽ)	+54	2	07:7bit / 08:8bit	08
		ﾊﾟﾘﾃｨ (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
ﾌﾟﾛﾄｺﾙ		下記参照	+62	14		
ﾌｧｲﾙﾓｰﾄﾞ		FORMAT	+76	3	F00:FAT ﾌｧｲﾙﾓｰﾄﾞ / F01:DT700 互換ﾓｰﾄﾞ	F00
ｻｲｽﾞ		ｱﾌﾟﾘｹｰｼｮﾝｻｲｽﾞ	+79	5	M0128-M0992(Kbyte)　16Kbyte ﾊﾞｳﾝﾀﾞﾘで指定	M0336

プロトコル関連：マルチドロップ

項目	位置	ｻｲｽﾞ	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		00
予約領域	+74	2		00

プロトコル関連：FLINK

項目	位置	ｻｲｽﾞ	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	ｻｲｽﾞ	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

CONFIG.OBR

設定ファイル『CONFIG.OBR』にて、NW-7、CODE39 の読み取りレベルの設定が行えます。
下記の書式のファイルを DT-900 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おくと、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 読み取りレベル
                    ;  n = 0(甘い)～3(厳しい)
                    ;  省略時は 2 となります

CODE39LEVEL=n               ; CODE-39 読み取りレベル
                    ;  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。

注 1）本機能は、DT-900 の OS 1.02/PATCH 1.06 以降の環境で有効です。

注 2）設定値により、誤読が発生したり、読取感度が悪くなる場合があります。

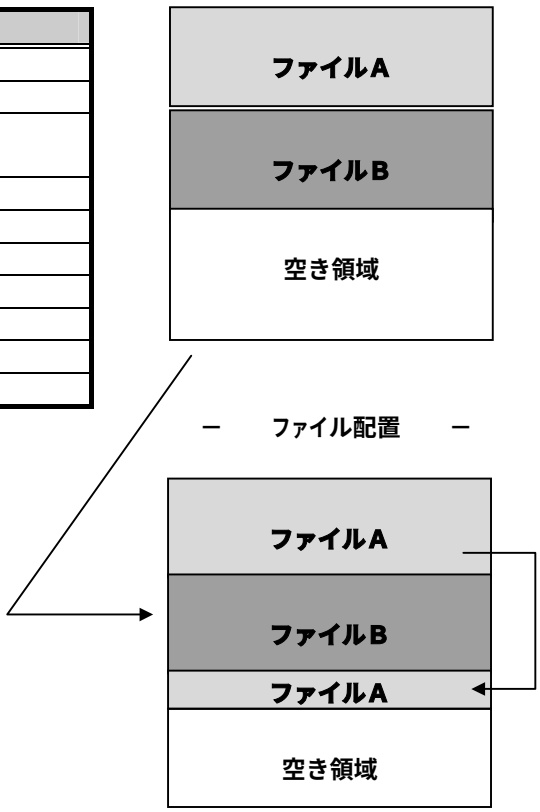
2.1.7.　バックアップ機能 (M50/M51 のみ)

本機に搭載しているＦＲＯＭデバイスを使用したドライブを提供し、データバックアップを実現します。
アクセス方法は通常のファイルとしてアクセス可能で、[Bドライブ] として扱われます。
ただし、書込み回数に制限があるため、書込み方式は追記のみとします。
※本ドライブは、本機に格納されるＫＣＧデータ量により可変となります。

注) M60 及び M61 に関しては、通常のディスクと同様に読み書きが行えるドライブになっています。

フォーマット

項目	内容
総容量	64KB-1408KB(初期値 512KB)
ｾｸﾀｻｲｽﾞ	ｾｸﾀの概念なし
ｾｸﾀ/ｸﾗｽﾀ	ｸﾗｽﾀの概念なし ※1 ﾌｧｲﾙ：最小 8KB
予約ｾｸﾀ	なし
ｾｸﾀ/FAT	なし
ﾃﾞｨﾚｸﾄﾘ	なし
総ｾｸﾀ数	なし
ﾒﾃﾞｨｱﾃﾞｨｽｸﾘﾌﾟﾀ	なし
同時 OPEN	1
総ﾌｧｲﾙ数	16



※ファイル配置
　基本的にはリニアに配置されますが
　上記状態からファイルAを一旦削除し
　前よりも大きなサイズのファイルを
　格納すると、右図のようにチェインします

データの信頼性
- アクセスはレコード単位となり、1回の書込み時に1レコード単位でチェックサムを付加します。

レコード形式

ﾃﾞｰﾀ数(2byte)	ﾃﾞｰﾀ	ｻﾑ(2byte)

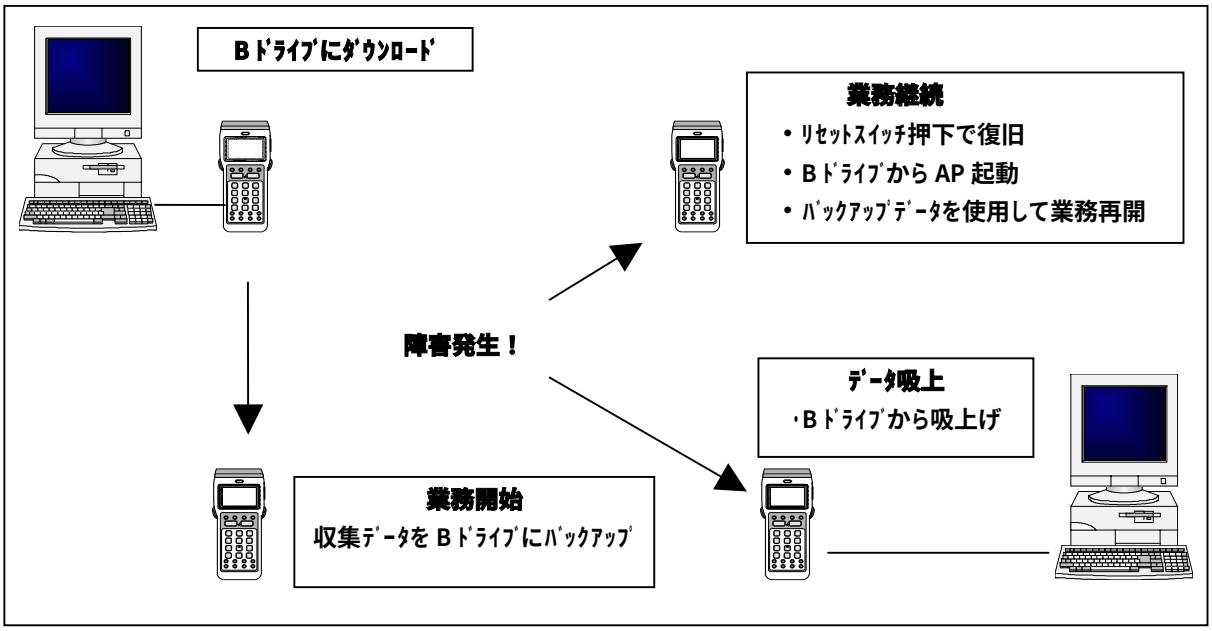
格納プログラム
- アプリケーション
- システムファイル
- マスタファイル

→ 更新されない READ ONLY ファイル
　障害発生後の業務継続を可能とするために、アプリケーションが動作できる最低限のファイルをＦＲＯＭ側に持つことを推奨します。

- バックアップデータ → 更新が必要な追記ファイル

バックアップシステム

FROMドライブは「データバックアップ」を実現するために用意されたもので、以下の想定に基づいています。



データバックアップ方法

- 通常ファイルを扱うのと同じ方法でBドライブにアクセスし、必要なデータを追記方式でバックアップして下さい。

データ吸い上げ方法

- バックアップデータはBドライブのﾌｧｲﾙとして存在しているため

  1：システムメニューのファイル転送による吸い上げ

  2：ファイル吸い上げ用アプリケーションによる吸い上げを行なって下さい。

業務継続方法

- バックアップドライブにアプリケーション／システムファイルを予め格納しておき、そこから起動するように設定しておいて下さい。

注意事項

・バックアップ昨日を使う時は、必ず B ドライブをフォーマットする等によりドライブ内にファイルが存在しない状態から使用して下さい。

※既に存在するファイルに対しての「削除」「クリエイト OPEN」が実行されると処理時間がかかるためアプリケーションの動作に支障をきたすおそれがあります。

・標準関数でのアクセスは禁止です。必ず低水準関数を使用して下さい。

・lseek によるファイルポインタの移動は read 時のみ有効です。write ポインタは常にファイルの最後にあり追記書込みのみをサポートします。

・ファイル転送等により PC 側からファイルを格納した場合、ファイル日付は作成された日時となります。

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。

機能	システムデータの設定	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

```
FN   fnc          :機能コード
                       SYSD_FNC_READ     :読出し
                       SYSD_FNC_WRITE    :登録
ID   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         :シリアルインタフェース
                       SYSD_COM3         :ＰＨＳインタフェース
                       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
*VP  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)
```

[リターンパラメータ]

```
ER   ercd         :リターンコード
```

[リターンコード]

```
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
```

≪備考≫

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

（設定値詳細は、「表２－５　システムデータ設定範囲」を参照してください）

```
[電源関連]
  typedef struct     sys_pwr{
                          w       apo;            /* ＡＰＯ時間設定                    */
                          w       abo;            /* ＡＢＯ時間設定                    */
                          w       res_md;         /* レジューム　ＯＮ／ＯＦＦ          */
                          }DAT_PWR_STR;

[ＫＥＹ関連]
  typedef struct     sys_key{
                          w       clk_md;         /* クリック音　ＯＮ／ＯＦＦ          */
                          }DAT_KEY_STR;

[ＯＢＲ関連]
  typedef struct    sys_obr {
                          w       rd_ct;          /* 読み取り回数                      */
                          w       cmp_ct;         /* 照合回数                          */
                          w       scn_tm;         /* スキャン時間                      */
                          }DAT_OBR_STR;

[表示関連]
  typedef struct    sys_disp {
                          w       font_md;        /* ６／８／１０ドットモード          */
                          w       lang_md;        /* 日本語／英語モード                */
                          }DAT_DSP_STR;

  typedef struct sys_disp2 {
                          w       font_kd;        /* ＮＯＲＭＡＬ／ＢＯＬＤ            */
                          w       cont_md;        /* コントラスト設定値                */
                          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                    */
                          w       com_port;       /* 通信ポート                        */
                          }DAT_COMINF_STR;

  typedef struct     sys_tty {
                          w       speed;          /* 転送速度                          */
                          w       length;         /* データ長                          */
                          w       parity;         /* パリティビット                    */
                          w       stop_bit;       /* ストップビット                    */
                          }DAT_COM_STR;

[タイマ関連]
  typedef struct     sys_time {
                          w       buzzer;         /* ブザー音量                        */
                          }DAT_TIM_STR;

[システム関連]
  typedef struct     sys_dat   {
                          UB      sys_id [7] ;    /* 機器ID  (Read only)               */
                          UB      bios_ver [7] ;  /* BIOS ﾊﾞｰｼﾞｮﾝ  (Read only)        */
                          UW      mac_type;       /* 機器種別　(0固定)　(Read only)    */
                          }DAT_SYS_STR;

  typedef struct     sys_dat2 {
                          UB      dlr_id [7] ;    /* 代理店ID  (Write only)　参照不可  */
                          UB      patch_ver [7] ; /* ﾊﾟｯﾁﾊﾞｰｼﾞｮﾝ(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                    */
                           w        non_retry_ct;       /* 通常ﾘﾄﾗｲ回数                      */
                           w        mal_rec_tmout;      /* ﾏﾙﾁﾄﾞﾛｯﾌﾟ受信ﾀｲﾑｱｳﾄ               */
                           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                          */

                        /* FLINK                 */
                           w        irda_tmout;         /* IrDA ｾｯｼｮﾝ確立ﾀｲﾑｱｳﾄ              */
                           w        irda_rec_tmout;     /* IrDA 受信ﾀｲﾑｱｳﾄ                   */
                           w        dr_tmout;           /* IrDArｾｯｼｮﾝ終了ﾀｲﾑｱｳﾄ              */
                           w        cs_tmout;           /* 予約領域                          */
                           w        cd_tmout;           /* 予約領域                          */

                        /* DT500                 */
                           w        sirial_no           /* シリアルNo                        */
                           w        level_parity        /* 水平パリティ                      */
                           w        bht_tmout           /* DT500 ﾌﾟﾛﾄｺﾙﾘﾝｸ確立ﾀｲﾑｱｳﾄ時間      */
                           }DAT_PRO_STR;
```

機能	OSバージョン読出し	関数名	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

≪C言語インタフェース≫

[コーリングシーケンス]
　void　dat_OSVer_Read(B *rd_buf);

[パラメータ]
　B　*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)

[リターンパラメータ]
　なし

[リターンコード]
　なし

≪備考≫

機能	代理店IDのチェック	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

UB	*dealer_no	:代理店ID格納領域のアドレス

[リターンパラメータ]

ER	ercd	:リターンコード

[リターンコード]

E_OK	:ID一致
E_NG	:ID不一致

≪備考≫
代理店IDは、6桁の半角英数字で構成されています。

機能	APロード＆実行	関数名	dat_Apload

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

≪C言語インタフェース≫

[コーリングシーケンス]

```
ER    ercd =dat_Apload( B *path );
```

[パラメータ]

```
B    *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)
```

[リターンパラメータ]

```
ER    ercd        :リターンコード
```

[リターンコード]

```
E_NG              :異常終了
```

≪備考≫

機能	ファイルオープン	関数名	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。
ファイルの同時オープン可能数は、DT700互換モード、FAT ファイルモード共に16です。
B ドライブは、M50/M51 の場合は1、M60/M61 の場合は 16 です。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

char　*name　　:指定ファイル名の格納先ポインタ(指定方法詳細は次ページ参照)

int　mode　　:ファイルモード
　O_RDONLY（B0）　:1の時、読込み専用
　O_WRONLY（B1）　:1の時、書込み専用
　O_RDWR　（B2）　:1の時、読込み／書込み両用
　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)

[リターンパラメータ]

Int　ercd　　: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、リターンコードを返します

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。

[リターンコード]

E_LOWERR　　:異常終了

≪備考≫

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:\ ppppppp \ nnnnnnnn.mmm
　　ﾄﾞﾗｲﾌﾞ名:ﾊﾟｽ名\　　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

機能	ファイルクローズ	関数名	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。

≪C 言語インタフェース≫

[コーリングシーケンス]
```
int   ercd = close(int fileno) ;
```

[パラメータ]

int	fileno	:クローズするファイル番号

[リターンパラメータ]

int	ercd	:リターンコード

[リターンコード]

E_OK	:正常終了
E_LOWERR	:異常終了

≪備考≫

機能	ファイルのリード	関数名	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。

≪C 言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

int　fileno	:読込み対象のファイル番号
char　*buf	:読込み領域のポインタ
unsigned　int　count	:読込みデータの要求バイト数

[リターンパラメータ]

int　ercd	: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、リターンコードを返します

[リターンコード]

E_LOWERR	:異常終了 ・チェックサム異常 ・ファイル未オープン

≪備考≫
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。

機能	ファイルのライト	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

int　fileno	:書込み対象のファイル番号
char　*buf	:書込み領域のポインタ
unsigned　int　count	:書込みデータの要求バイト数

[リターンパラメータ]

int　ercd	: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、リターンコードを返します

[リターンコード]

E_LOWERR	:異常終了 ・書込み異常 ・ファイル未オープン

≪備考≫
読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。

機能	ファイルリード／ライト位置の設定	関数名	lseek

＜低水準インタフェース関数＞
指定ファイルの読込／書込位置をバイト単位で設定します。
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。

≪C 言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

int　fileno	:対象のファイル番号
long　offset	:読込み／書込み位置の変更先 base で指定された位置からのオフセット値 （バイト単位）
int　base	:0の時、ファイルの先頭を基準とします 1の時、現在の読込み／書込み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

[リターンパラメータ]

int　ercd	: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

[リターンコード]

E_LOWERR	:異常終了 ・ファイル未オープン ・更新した読込み／書込み位置が負 ・更新した読込み／書込み位置がファイルサイズを超えた

≪備考≫

機能	メモリ領域の割当て	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]
　　char　*buf= sbrk(unsigned long size) ;

[パラメータ]
　　unsigned　long　size　　：要求データのサイズ（1～16KBバイト）

[リターンパラメータ]
　　char　*buf　　：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

[リターンコード]
　　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

≪備考≫

機能	ファイル格納情報の取得	関数名	dat_fdir　(DT700 互換ﾓｰﾄﾞ専用)

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。

≪C言語インタフェース≫

[コーリングシーケンス]

ER　ercd = dat_fdir(B id, DIR_TBL *buf);

[パラメータ]

B　id　　:ファイル管理テーブルの読込み位置指定
　　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

DIR_TBL *buf　:ファイル格納情報の読込み領域 (詳細は、次頁参照)

[リターンパラメータ]

ER　ercd　　:リターンコード

[リターンコード]

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

≪備考≫

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;              /* 先頭ｱﾄﾞﾚｽ     */
        W     size;                 /* ﾌｧｲﾙ ｻｲｽﾞ     */
        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）

機能	ファイル空き領域サイズの取得	関数名	dat_fsize　(DT700 互換ﾓｰﾄﾞ専用)

ファイル格納領域の未使用領域サイズを取得します。

≪C 言語インタフェース≫

[コーリングシーケンス]

```
UW    size = dat_fsize(void);
```

[パラメータ]

なし

[リターンパラメータ]

UW　size　　：ファイル格納領域の未使用領域サイズ

[リターンコード]

なし

≪備考≫

機能	ファイルの削除	関数名	dat_fdel　(DT700 互換ﾓｰﾄﾞ専用)

ファイルを削除します。

≪C言語インタフェース≫

[コーリングシーケンス]

ER　ercd = dat_fdel(B *name);

[パラメータ]

B　*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 (文字列ﾃﾞｰﾀ)
　拡張子　　:　3文字以下 (文字列ﾃﾞｰﾀ)

　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。

[リターンパラメータ]

ER　ercd　　:リターンコード

[リターンコード]

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

≪備考≫

機能	メモリ領域の空きサイズの取得	関数名	dat_mem_size

メモリ領域の未使用領域のサイズを返します。

≪C 言語インタフェース≫

[コーリングシーケンス]

```
UW size = dat_mem_size( void );
```

[パラメータ]

なし

[リターンパラメータ]

UW　size　　:メモリの未使用領域サイズ

[リターンコード]

なし

≪備考≫

機能	ディレクトリの作成	関数名	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：Aドライブのファイルが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。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

```
const    char    *path              :作成するディレクトリのフルパス名
```

[リターンパラメータ]

```
ER    ercd        :リターンコード
```

[リターンコード]

```
E_OK              :正常終了
E_NG              :異常終了
```

≪備考≫

・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のようになります。
(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￥(ｽﾍﾟｰｽ)￥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。

機能	ディレクトリの削除	関数名	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: Aドライブのファイルが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。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

```
const   char   *path        :削除するディレクトリのフルパス名
```

[リターンパラメータ]

```
ER    ercd            :リターンコード
```

[リターンコード]

```
E_OK                  :正常終了
E_NG                  :異常終了
```

≪備考≫

機能	ファイルの削除	関数名	fil_remove

ファイルを削除します。

制限事項：ＦＡＴファイルモードで、Ａドライブのファイルが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。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

```
Const   char   *pathname      :ファイルのパス名 (指定方法は、open 関数参照)
```

[リターンパラメータ]

```
ER    ercd       :リターンコード
```

[リターンコード]

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

≪備考≫

機能	ファイル名の変更／移動	関数名	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: ＦＡＴファイルモードで、Ａドライブのファイルが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）

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

const	char	*oldname	:現在のファイルのパス名（指定方法は、open 関数参照）
const	char	*newname	:新しいファイルのパス名（指定方法は、open 関数参照）

[リターンパラメータ]

ER	ercd	:リターンコード

[リターンコード]

E_OK	:正常終了
E_NG	:異常終了
E_PRM	:パラメータエラー

≪備考≫

機能	ファイルの日時･サイズ･属性の取得	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

```
int     handle          :ファイル番号
FIL_FSTAT *buffer       :結果格納バッファ
```

[ストラクト構造]

```
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  : 読込み専用
                                                _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;
```

[リターンパラメータ]

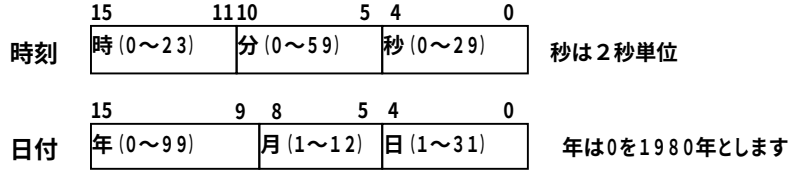
```
ER   ercd       :リターンコード
```

[リターンコード]

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

≪備考≫

時刻と日付のフォーマット



機能	ファイルのサイズの変更	関数名	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：Aドライブのファイルが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。
　　Bドライブは、対象外です（異常終了します）

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

B	*path	：変更対象のファイル名（全パス名指定）
UW	*fsize	：変更するファイルのサイズ（バイト単位）

[リターンパラメータ]

ER	ercd	：リターンコード

[リターンコード]

E_OK	：正常終了
E_NG	：異常終了

≪備考≫

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。

機能	ファイル領域空きサイズの取得	関数名	fil_getsize

ファイル領域の空きサイズの取得をします。

≪C 言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

B	*path	:ドライブ名

[リターンパラメータ]

UW	size	:空き領域サイズ

[リターンコード]

E_NG	:異常終了
E_PRM	:パラメータエラー

≪備考≫

機能	ファイル名の取得	関数名	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)

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

```
B    *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)
UH   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                  :読込み専用 ※
                     _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
FIND_T   *buffer    :結果を格納するバッファ
```

[ストラクト構造]

```
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;
```

[リターンパラメータ]

```
ER   ercd       :リターンコード
```

[リターンコード]

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

≪備考≫

時刻

15	11	10	5	4	0
時 (0～23)		分 (0～59)		秒 (0～29)	

秒は２秒単位

日付

15	9	8	5	4	0
年 (0～99)		月 (1～12)		日 (1～31)	

年は0を1980年とします

機能	ファイル名の取得 (次候補)	関数名	fil_findnext

fil_findfirst 関数にて、取得されたファイル名の次候補を読出します。
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。

≪C 言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

```
FIND_T    *buffer    :結果を格納するバッファ
```

[ストラクト構造]

```
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;
```

[リターンパラメータ]

```
ER    ercd          :リターンコード
```

[リターンコード]

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

≪備考≫

機能	ファイルの個数と総サイズの取得	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

```
B    *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）
FIL_SIZE *buffer    :結果を格納するバッファ
```

[ストラクト構造]

```
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;
UB    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
```

[リターンパラメータ]

```
ER    ercd           :リターンコード
```

[リターンコード]

```
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）
```

≪備考≫

[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）

機能	ファイル全パス名の取得	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

B　*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）	
UB　*buffer	：ファイル名を格納するバッファ（形式2で返却）	
UB　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
UH　seq_no	：指定番目の番号	

[リターンパラメータ]

ER　ercd	：リターンコード

[リターンコード]

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）

≪備考≫

[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

機能	ファイル名変更	関数名	dat_frname (DT700 互換ﾓｰﾄﾞ専用)

ファイル名の変更を行ないます。

≪C言語インタフェース≫

[コーリングシーケンス]

ER　ercd = dat_frname(UB *old ,　UB *new);

[パラメータ]

UB　*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

UB　*new　　:変更後のファイル名

[リターンパラメータ]

ER　ercd　　:リターンコード

[リターンコード]

E_OK　　:正常終了
E_NG　　:異常終了

≪備考≫

[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。

機能	ファイルデータの検索	関数名	dat_F_Search (DT700 互換ﾓｰﾄﾞ専用)

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 をリターンコードとして返します)

≪C言語インタフェース≫

[コーリングシーケンス]

```
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]

B	*filename	:検索要求ﾌｧｲﾙ名格納先ｱﾄﾞﾚｽ
W	start_adr	:検索開始相対ｱﾄﾞﾚｽ
H	fieldsize	:１検索ﾃﾞｰﾀのﾃﾞｰﾀｻｲｽﾞ
H	keypos	:検索ｺｰﾄﾞの格納先相対ｱﾄﾞﾚｽ
H	keylen	:検索ｺｰﾄﾞのﾃﾞｰﾀｻｲｽﾞ
UB	*code	:比較検索ｺｰﾄﾞの格納先ｱﾄﾞﾚｽ
UB	*sdata	:検索ﾃﾞｰﾀの格納先ﾊﾞｯﾌｧｱﾄﾞﾚｽ (出力情報)
W	*fpos	:検索ﾃﾞｰﾀｱﾄﾞﾚｽ格納ﾎﾟｲﾝﾀ (出力情報)

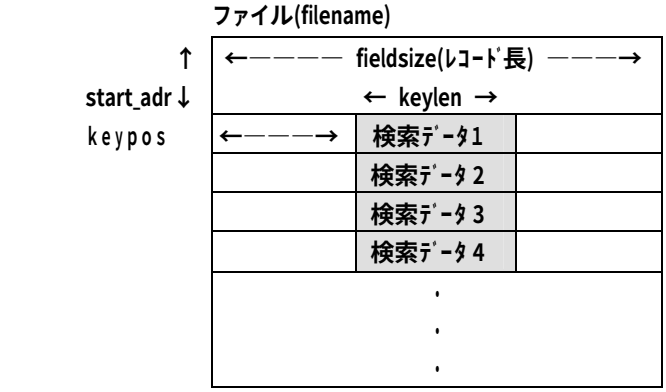
[リターンパラメータ]

ER	ercd	:リターンコード

[リターンコード]

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

≪備考≫　パラメータ補足説明



検索ﾃﾞｰﾀ(code)

＜例：検索ﾃﾞｰﾀ3で検索できた場合＞
*sdata＝検索ﾃﾞｰﾀ3
*fpos　=検索ﾃﾞｰﾀ3のアドレス

3. 表示部

3.1. 表示制御

3.1.1. 表示画面

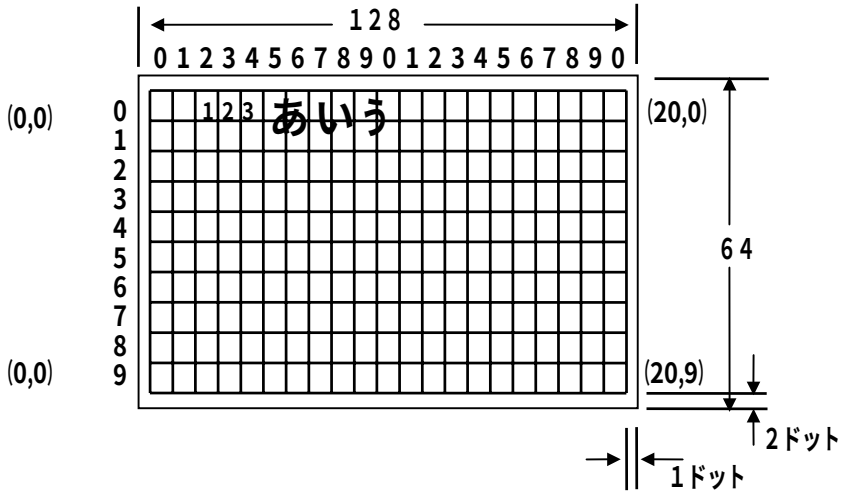
128×64 ドットの画面サイズで、各フォントモードの縮小 ANK を基準に、左上を (0,0)のキャラクタ座標系で構成します。
グラフィック座標は、フォントモードに関係なく、左上(0,0)、右下(127,63)になります。
フォントの表示は、キャラクタ座標で行います。

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。

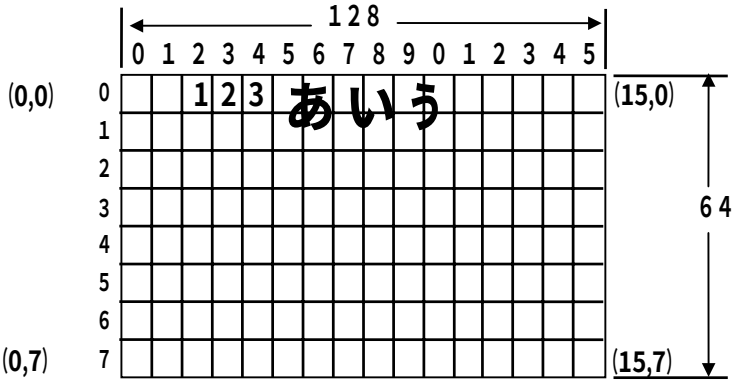
表3．1 表示行桁数

モード	フォント(サイズ)	表示桁数	最大表示文字数
6ドット	縮小ANK (6× 6)	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

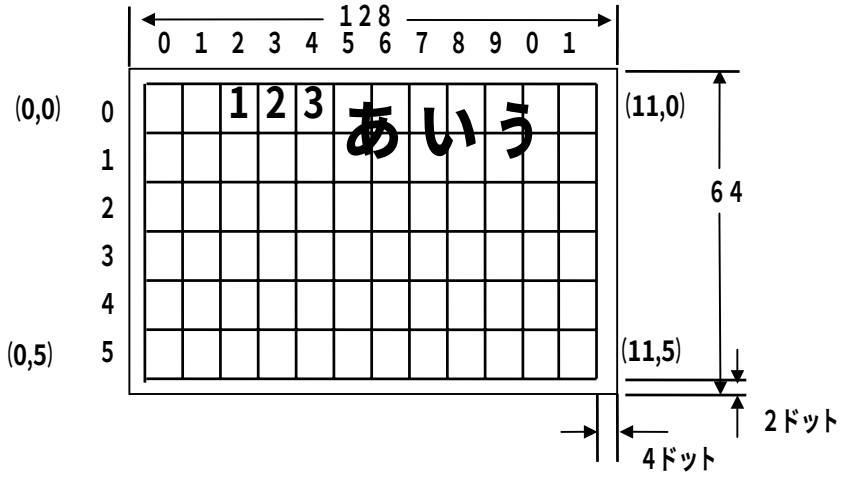
(1) 6ドットモード時



(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			ァ	キ	ヌ	ラ		
8	BS		(	8	H	X	h	x			ィ	ク	ネ	リ		
9			)	9	I	Y	I	y			ゥ	ケ	ノ	ル		
A	LF		*	:	J	Z	j	z			ェ	コ	ハ	レ		
B		ESC	+	;	K	[	k	{			ォ	サ	ヒ	ロ		
C		←	,	<	L	\	l	¦			ャ	シ	フ	ワ		
D	CR	→	-	=	M	]	m	}			ュ	ス	ヘ	ン		
E			.	>	N	^	n	~			ョ	セ	ホ	ﾞ		
F			/	?	O	_	o				ッ	ソ	マ	ﾟ		

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(ﾋﾞｯﾄﾏｯﾌﾟ)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ﾊﾞｲﾄ目が、7F のｺｰﾄﾞ(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 84	第1水準											
				88												
							第			1		水		準		
90																
				98 9F												
							第			2		水		準		
A0																
B0																
C0																
D0					40			7E	80				C0			FC
E0				E0	第2水準											
				EA EB												
							第			2		水		準		
								外	字							
F0																

(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 89～9F E0～EA	40～7E, 80～FC	漢字コード表の文字	ユーザー登録の文字
	00～3F, 7F FD～FF	漢字スペース	8140hのフォント
88	9F～FC	漢字コード表の文字	ユーザー登録の文字
	00～9E, FD～FF	漢字スペース	8140hのフォント
EB	40～7E 80～C0	外字フォント文字　(外字ファイルあり時) 漢字スペース　(外字ファイルなし時)	
	00～3F, 7F C1～FF	漢字スペース	8140hのフォント
上記以外	00～FF	何も表示しません	

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 89～9F E0～EA	00	文字列表示の終了　(画面表示しません)	
	40～7E, 80～FC	漢字コード表の文字	ユーザー登録の文字
	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

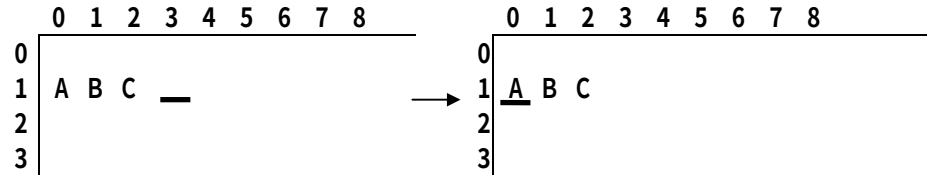
※ ユーザー文字列表示 (lcd_userstr) の場合は、漢字の表示はありません。全て1バイト目を ANK コードとしてみます。81-84、88-9F、E0-EB のコードは ROM フォント時は、ANK スペースを、ユーザーフォント時は、ユーザー登録文字をそれぞれ表示します。

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。

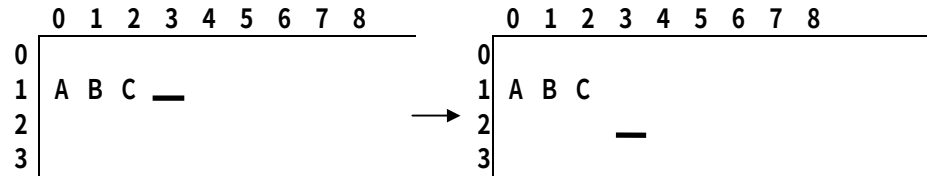
(1)　CR (0x0d) の表示

・ キャリッジリターン動作を行います

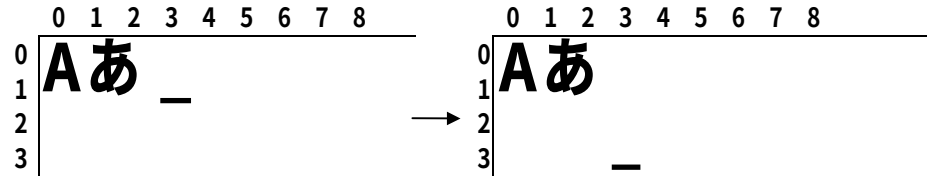


(2)　LF (0x0a) の表示

・ 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。

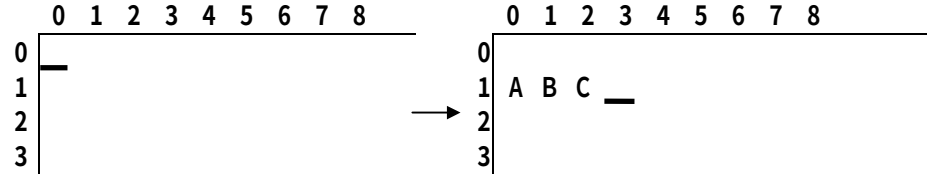


・ 標準ANKおよび漢字の文字列中にLFコードが含まれている場合は、2行改行します。

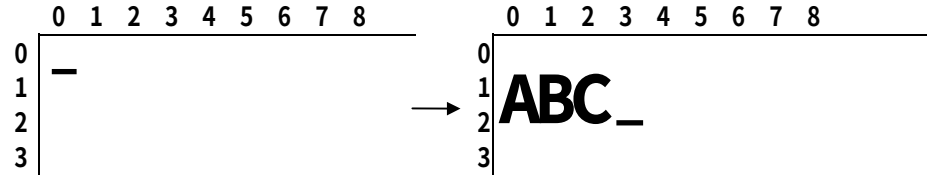


・ 文字列表示の先頭がLFコードまたは、1文字表示でLFコードが指定された場合は、入力パラメータのANKモードが縮小の場合は1行改行、標準の場合は2行改行します。

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 ”

Pm：桁
Pn：行

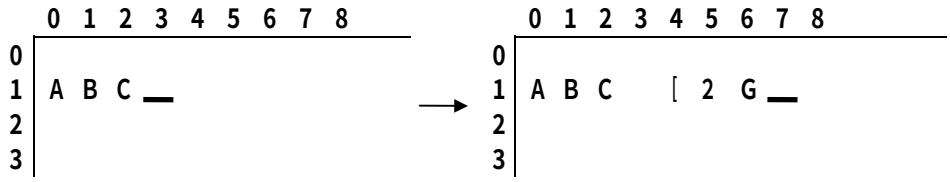
座標範囲は各フォントモードの縮小ＡＮＫを基準にします。

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。

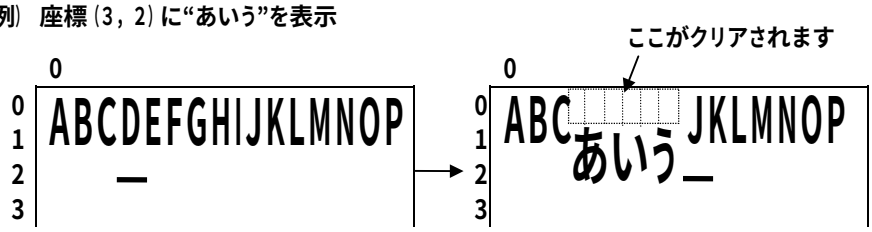
表3．6 例外動作

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。

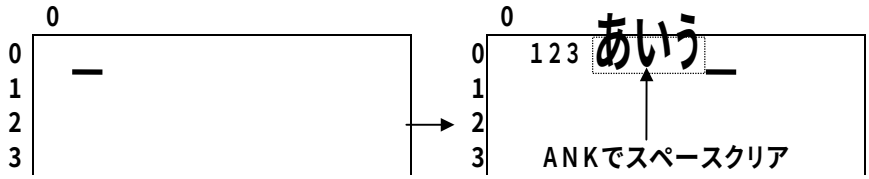
(例)　座標(3，2)に“あいう”を表示



(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。

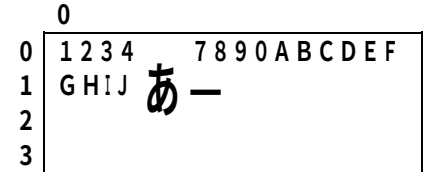
(例)　座標(1，0)に“123あいう”を表示



(3) 行端での自動改行制御

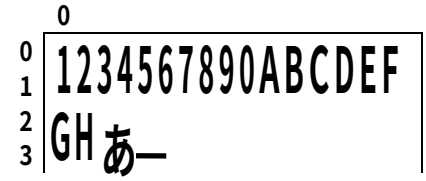
文字列表示を行う時、行端で表示仕切れない場合には先頭文字により、1または2行の改行を自動で行います。
(ただし、改行モードあり指定時)

(例1) 先頭文字が縮小ANKの場合、座標(0, 0)に“1234567890ABCDEFGHIJあ”を表示



※ 56は上書きされます。

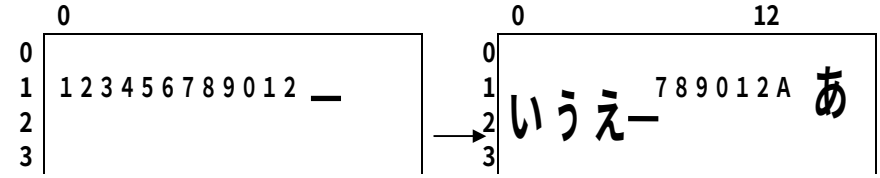
(例2) 先頭文字が標準ANK／漢字の場合、座標(0, 0)に“1234567890ABCDEFGHあ”表示



(4) 行端での漢字表示制御

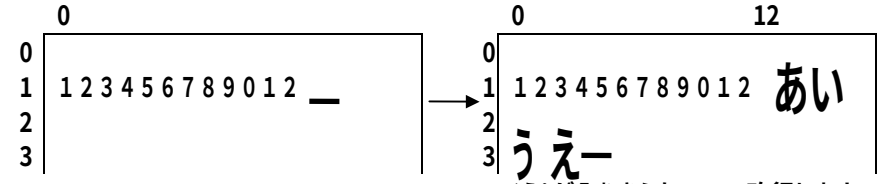
行端で切れ端になる場合には、1または2行の改行を自動で行います。(ただし、改行モードありの時)

(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



・‘い’が入りきらないので改行します。
・123456は上書きされます

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

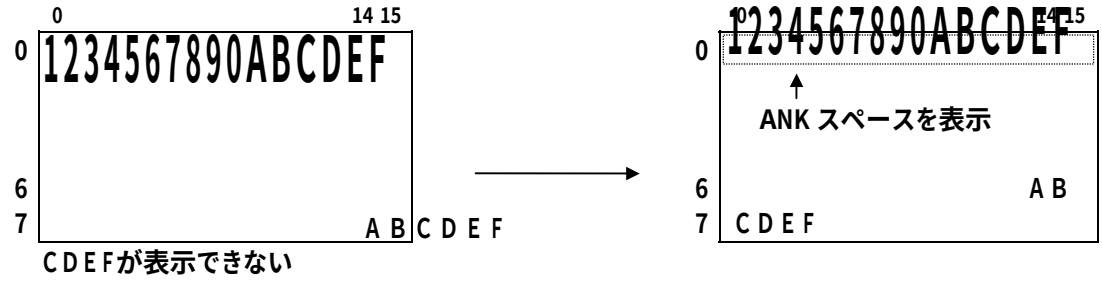


・‘う’が入りきらないので改行します

3.1.6.　スクロール制御

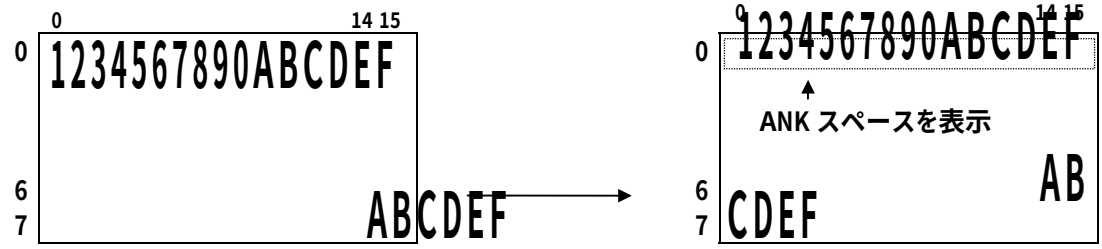
カレントカーソル位置が最下行で文字列が表示できない場合はスクロール制御を行います。

（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



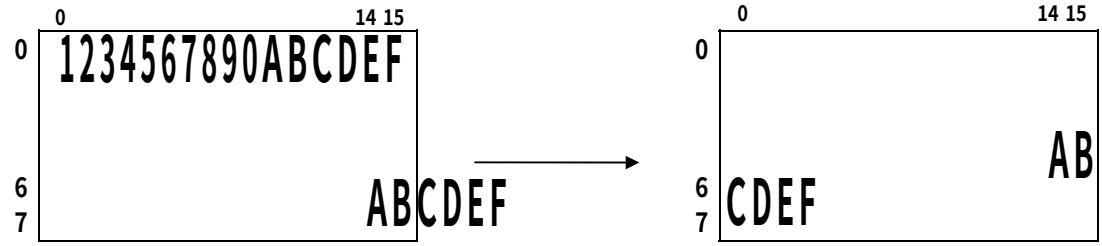
CDEFが表示できない

（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



CDEFが表示できない

（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



CDEFが表示できない

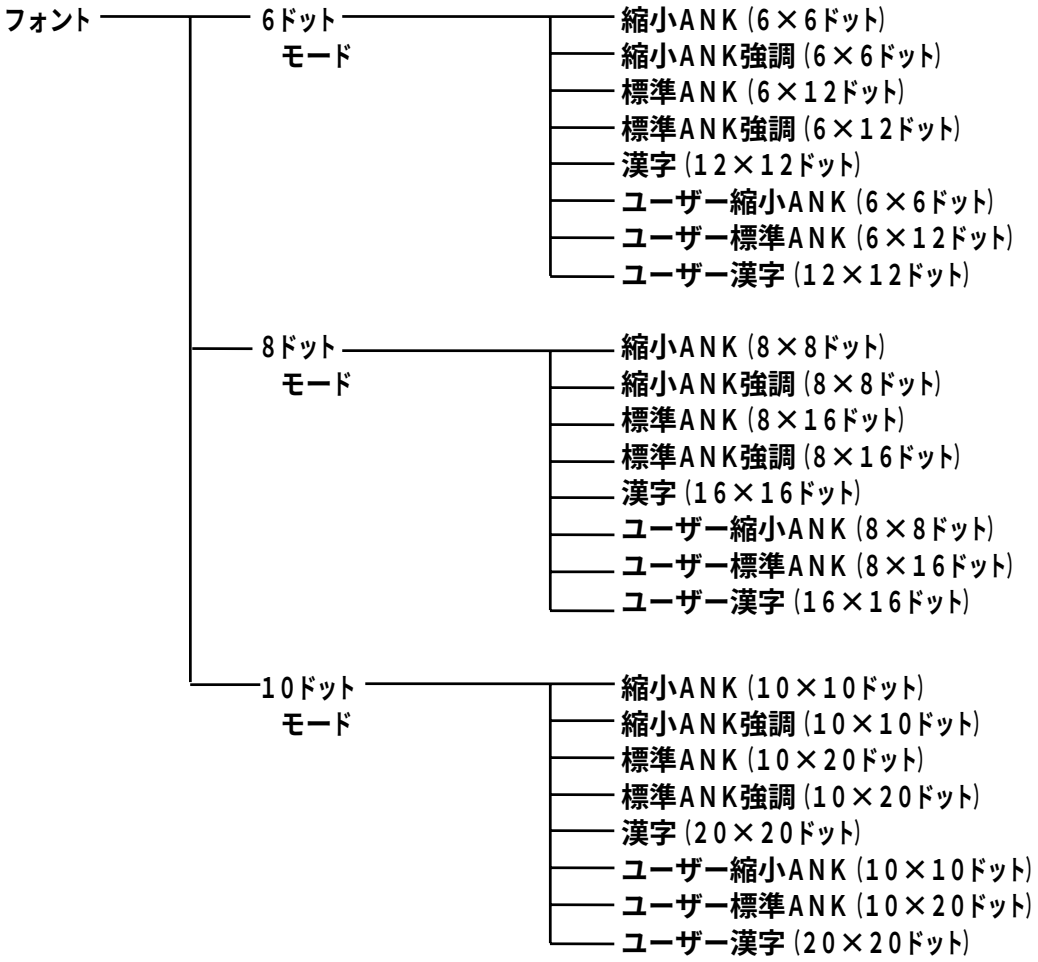
※スクロール抑制

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に依存します。
異なるフォントモードの混在表示はできません。

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。

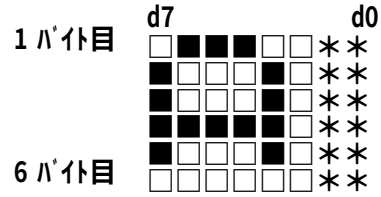
※　ユーザーフォントの表示を行う場合は、「ユーザーフォントファイルの登録」を、ROMフォントへ戻す場合は、「ROMフォント設定」を行って下さい。これによりユーザー／ROMフォントの混在表示が可能です。

※　ユーザーフォント表示の場合は、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)

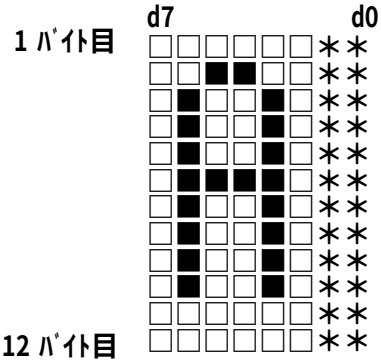


1フォント6バイト構造

70	88	88	F8	88	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

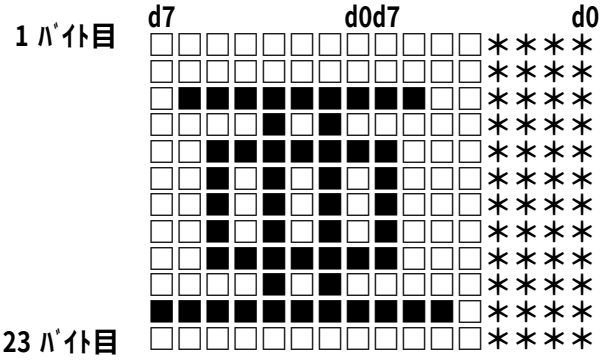
00	30	48	48	48	78
1	2	3	4	5	6

(バイト目)

48	48	48	48	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)



1フォント24バイト構造

00	00	00	00	7F	C0	0A	00	3F	80	2A	80
1	2	3	4	5	6	7	8	9	10	11	12

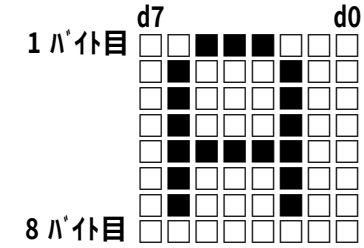
(バイト目)

2A	80	2A	80	3F	80	0A	00	FF	E0	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)

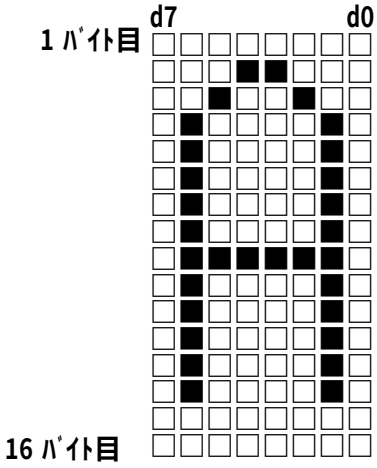


1フォント8バイト構造

38	44	44	44	7C	44	44	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

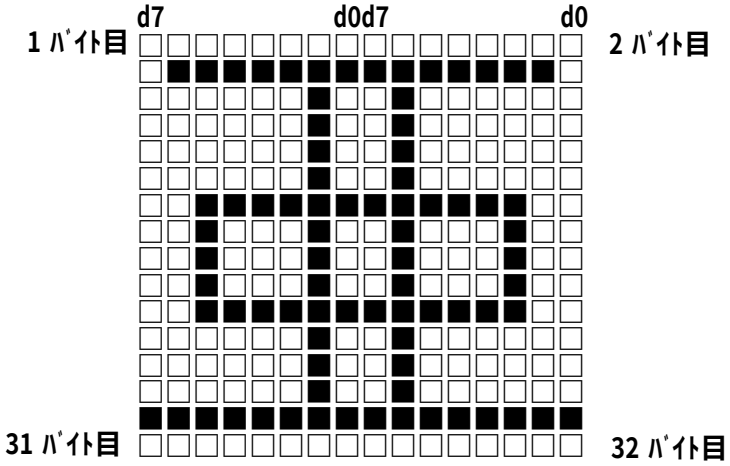
00	18	24	42	42	42	42	42
1	2	3	4	5	6	7	8

(バイト目)

7E	42	42	42	42	42	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	00	7F	FE	02	40	02	40	02	40	02	40	3F	FC	22	44
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

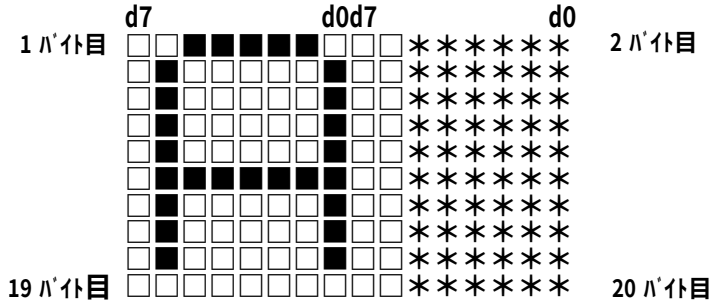
(バイト目)

22	44	22	44	3F	FC	02	40	02	40	02	40	FF	FF	00	00
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)



1フォント20バイト構造

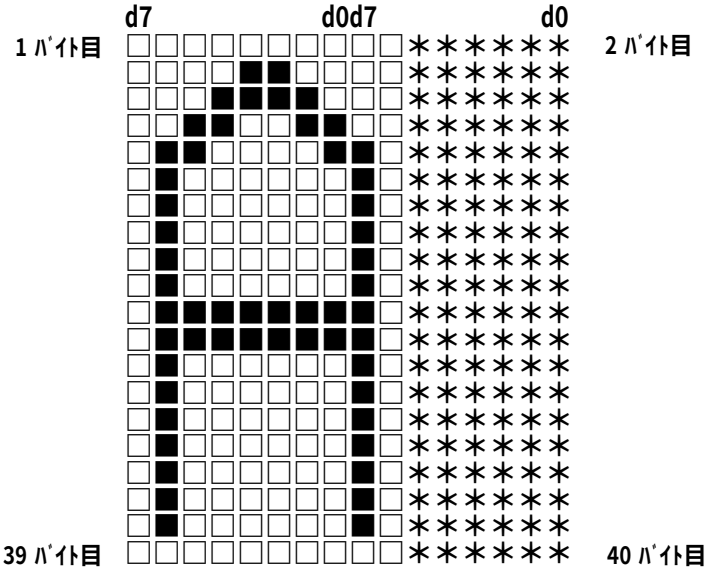
3E	00	41	00	41	00	41	00	41	00
1	2	3	4	5	6	7	8	9	10

(バイト目)

7F	00	41	00	41	00	41	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)



1フォント40バイト構造

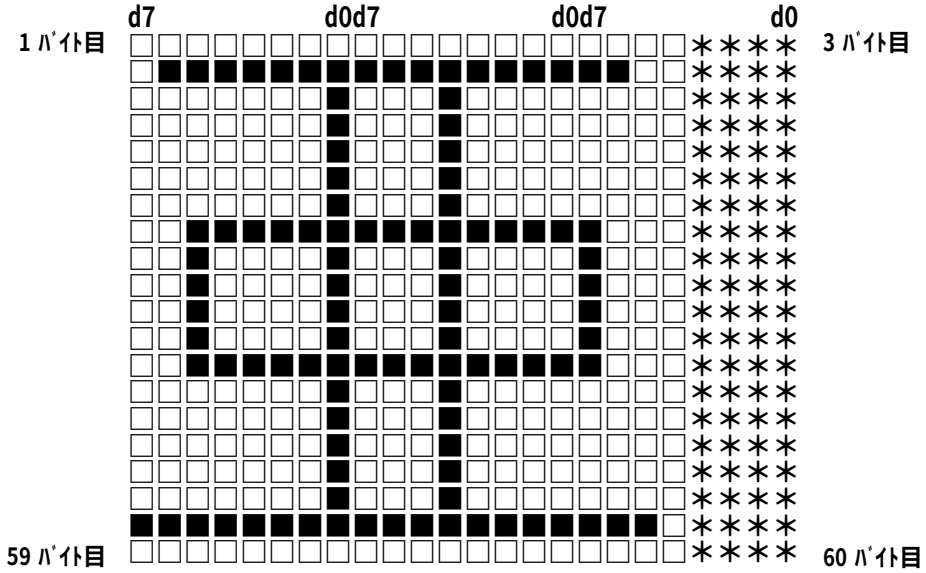
00	00	0C	00	1E	00	33	00	61	80	40	80	40	80	40	80	40	80	40	80
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

(ﾊﾞｲﾄ目)

7F	80	7F	80	40	80	40	80	40	80	40	80	40	80	40	80	40	80	00	00
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40

(ﾊﾞｲﾄ目)

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	00	7F	FF	C0	01	10	00	01	
1	2	3	4	5	6	7	8	9	10	(バイト目)
10	00	01	10	00	01	10	00	01	10	
11	12	13	14	15	16	17	18	19	20	(バイト目)
00	3F	FF	80	21	10	80	21	10	80	
21	22	23	24	25	26	27	28	29	30	(バイト目)
21	10	80	21	10	80	3F	FF	80	01	
31	32	33	34	35	36	37	38	39	40	(バイト目)
10	00	01	10	00	01	10	00	01	10	
41	42	43	44	45	46	47	48	49	50	(バイト目)
00	01	10	00	FF	FF	E0	00	00	00	
51	52	53	54	55	56	57	58	59	60	(バイト目)

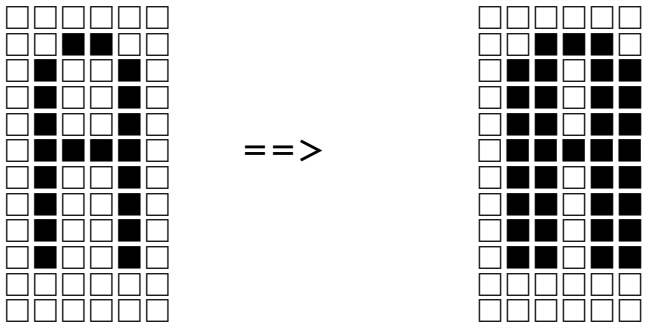
3.2.3.　修飾文字のフォントデータ制御

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。

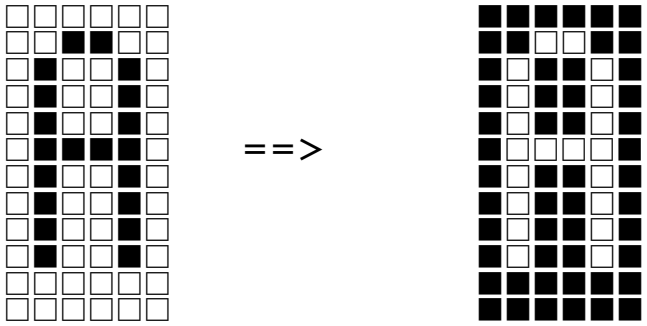
(1) 強調表示

・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。



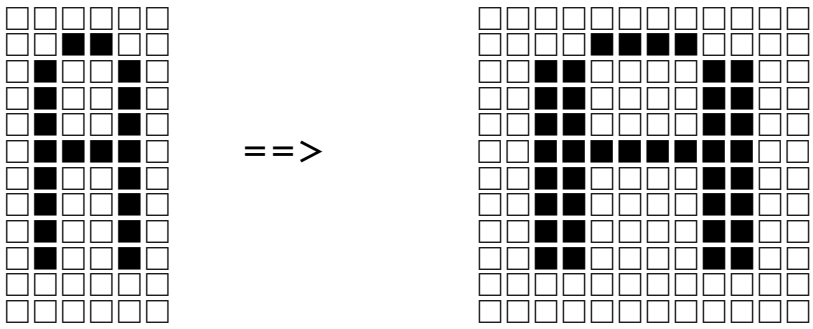
(2) 反転表示

・フォントデータのビットパターンを反転します。



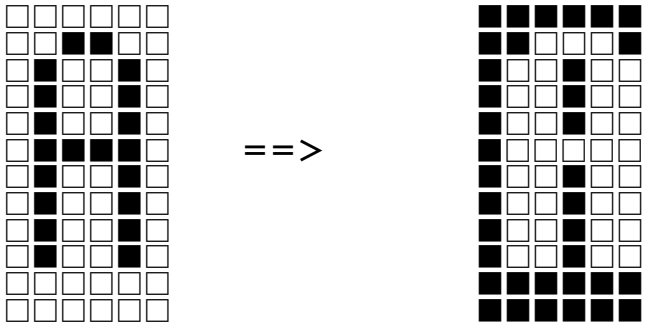
(3) 横倍角表示

・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。



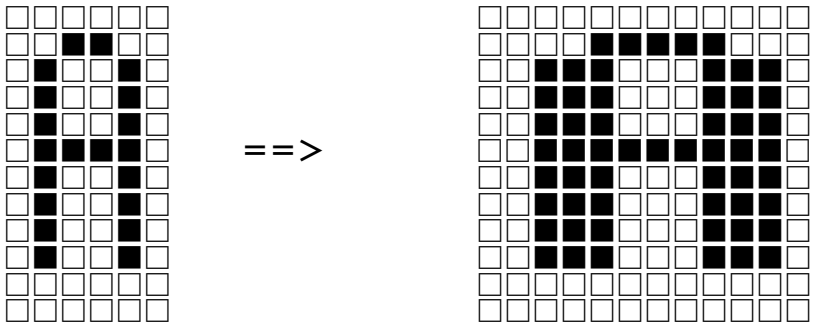
(4) 強調／反転表示

・(1)、(2) 両方の処理を合わせます。(強調後反転する)



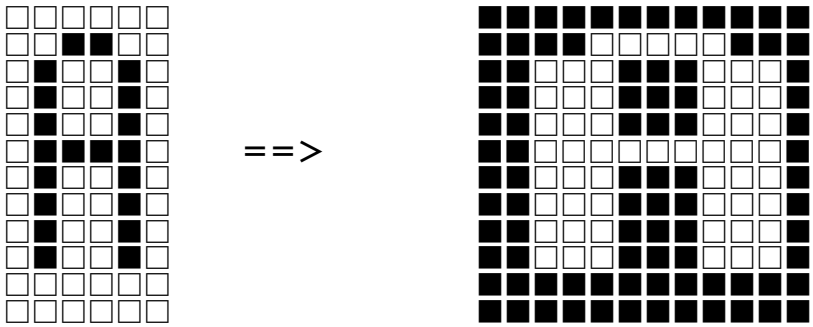
(5) 横倍角／強調表示

・(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)



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。

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

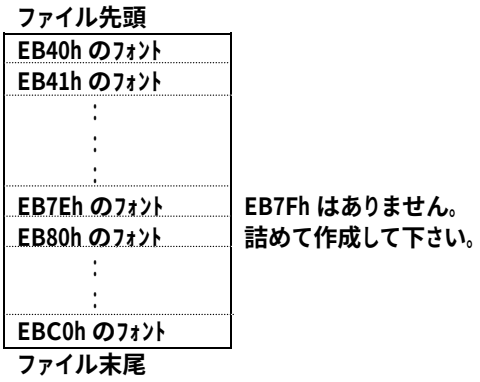
(2) フォントデータ構成

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

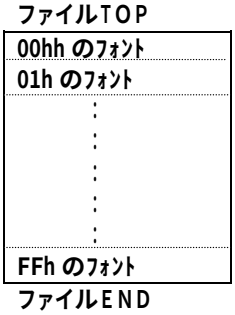
・外字フォントファイル構成

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。



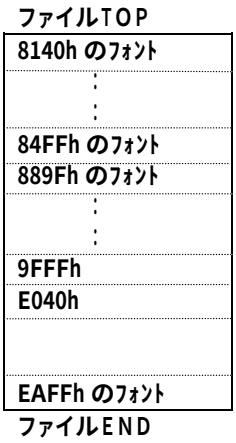
・ＡＮＫフォントファイル構成

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。



・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。



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。

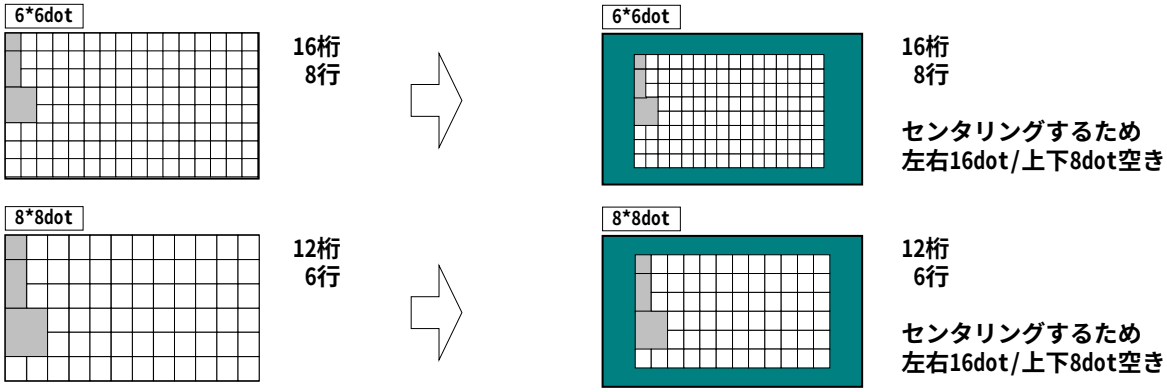
※ 処理高速化のため、外字フォントおよびANKフォントは登録時にメモリに展開します。ファイルを更新した場合は、登録し直して下さい。
(漢字フォントは1文字表示／文字列表示で漢字表示時にファイルアクセスします。)

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)

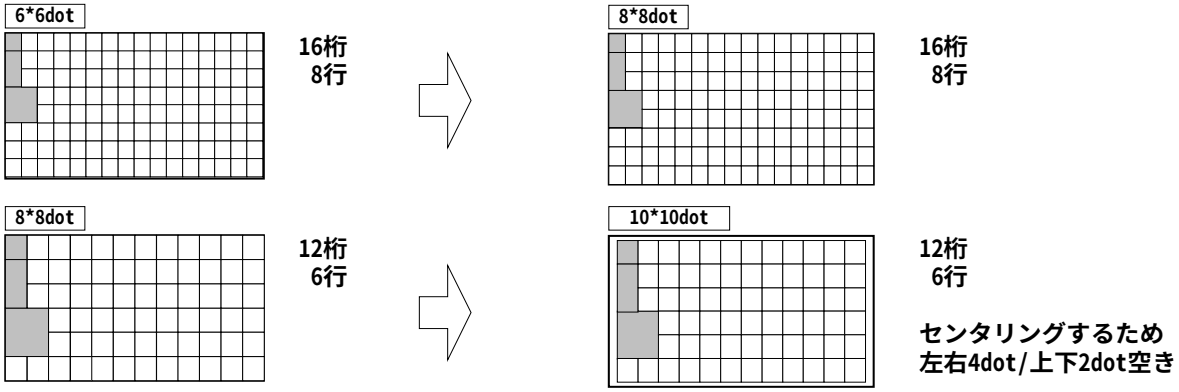


表3．7　モード時の表示行桁数

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

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

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。

機能	画面クリア	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER　ercd = lcd_cls(void);

[パラメータ]
　　なし

[リターンパラメータ]
　　ER　ercd　　：リターンコード

[リターンコード]
　　E_OK　　：正常終了

≪備考≫

機能	カーソルタイプ設定	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER　ercd = lcd_csr_set(H csr_type);

[パラメータ]
　　H　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

[リターンパラメータ]
　　ER　ercd　　：リターンコード

[リターンコード]
　　E_OK　　：正常終了
　　E_PRM　　：パラメータエラー

≪備考≫

機能	カーソル位置設定	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER　ercd = lcd_csr_put(H csr_line,　H csr_colum);

[パラメータ]
　　H　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
　　H　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

[リターンパラメータ]
　　ER　ercd　　：リターンコード

[リターンコード]
　　E_OK　　：正常終了

≪備考≫

機能	カーソル位置読出し	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

H　*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	
H　*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	
H　*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

[リターンパラメータ]

ER　ercd	:リターンコード

[リターンコード]

E_OK	:正常終了

≪備考≫

機能	1文字表示	関数名	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。

≪C言語インタフェース≫

【コーリングシーケンス】

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

【パラメータ】

H　ank_mode	：ＡＮＫモード	
	縮小ＡＮＫモード	：LCD_ANK_LIGHT
	標準ＡＮＫモード	：LCD_ANK_STANDARD
H　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。	
H　disp_data	：表示データ	
H　lf_mode	：改行モード	
	改行なし	：LCD_LF_OFF
	改行あり	：LCD_LF_ON

【リターンパラメータ】

ER　ercd	：リターンコード

【リターンコード】

E_OK	：正常終了
E_PRM	：パラメータエラー

≪備考≫

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。

機能	文字列表示	関数名	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。

≪Ｃ言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

H　ank_mode	：ＡＮＫモード	
	縮小ＡＮＫモード	：LCD_ANK_LIGHT
	標準ＡＮＫモード	：LCD_ANK_STANDARD
H　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。	
UB　*disp_data	：表示データバッファポインタ	
H　lf_mode	：改行モード	
	改行なし	：LCD_LF_OFF
	改行あり	：LCD_LF_ON

[リターンパラメータ]

ER　ercd	：リターンコード

[リターンコード]

E_OK	：正常終了
E_PRM	：パラメータエラー

≪備考≫

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。）

機能	文字列表示2 (スクロール抑制)	関数名	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ないます。また、最下行にある場合はスクロールを行ないます。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

H　ank_mode	:ＡＮＫモード	
	縮小ＡＮＫモード	:LCD_ANK_LIGHT
	標準ＡＮＫモード	:LCD_ANK_STANDARD
H　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。	
UB　*disp_data	:表示データバッファポインタ	
H　lf_mode	:改行モード	
	改行なし	:LCD_LF_OFF
	改行あり	:LCD_LF_ON

[リターンパラメータ]

ER　ercd	:リターンコード

[リターンコード]

E_OK	:正常終了
E_PRM	:パラメータエラー

≪備考≫

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替ります。

機能	ユーザー文字列表示	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

H　ank_mode	：ＡＮＫモード	
	縮小ＡＮＫモード	：LCD_ANK_LIGHT
	標準ＡＮＫモード	：LCD_ANK_STANDARD
H　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
UB　*disp_data	：表示データバッファポインタ	
H　lf_mode	：改行モード	
	改行なし	：LCD_LF_OFF
	改行あり	：LCD_LF_ON

[リターンパラメータ]

ER　ercd	：リターンコード

[リターンコード]

E_OK	：正常終了
E_PRM	：パラメータエラー

≪備考≫

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。

機能	直線描画	関数名	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。

≪C言語インタフェース≫

【コーリングシーケンス】
　　ER　ercd = lcd_line(H dot_mode，　H strart_x，　H start_y，　H end_x，　H end_y);

【パラメータ】
　　H　dot_mode　　：直線表示モード
　　　　直線削除　　：LCD_LINE_OFF
　　　　直線描画　　：LCD_LINE_ON
　　H　start_x　　：開始X座標
　　　　横ドット位置の０～１２７を指定
　　H　start_y　　：開始Y座標
　　　　縦ドット位置の０～６３を指定
　　H　end_x　　：終了X座標
　　　　横ドット位置の０～１２７を指定
　　H end_y　　：終了Y座標
　　　　縦ドット位置の０～６３を指定

【リターンパラメータ】
　　ER　ercd　　：リターンコード

【リターンコード】
　　E_OK　　：正常終了

≪備考≫
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。

機能	外字フォント登録	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER　ercd = lcd_gaiji(H file_mode,　B *filename);

[パラメータ]
　　H　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
　　B　*filename　　：外字登録ファイル名称

[リターンパラメータ]
　　ER　ercd　　：リターンコード

[リターンコード]
　　E_OK　　：正常終了
　　E_PRM　　：パラメータエラー

≪備考≫
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。

機能	ユーザーフォントファイル登録	関数名	lcd_usrfont

ユーザーフォントをシステムに登録します。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

H　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
B　*filename	:ユーザーフォントファイル名称	

[リターンパラメータ]

ER　ercd	:リターンコード

[リターンコード]

E_OK	:正常終了
E_PRM	:パラメータエラー

≪備考≫

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。
登録ファイルを無効にする場合は、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込みます。

機能	ROMフォント設定	関数名	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。）

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER　ercd = lcd_romfont(void);

[パラメータ]
　　なし

[リターンパラメータ]
　　ER　ercd　　:リターンコード

[リターンコード]
　　E_OK　　:正常終了

≪備考≫

機能	ＬＥＤの制御	関数名	lcd_led

読み取りＬＥＤの点灯／消灯を行います。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER　ercd = lcd_led(H led_mode,　H led_kind);

[パラメータ]
　　H　led_mode　　：ＬＥＤモード
　　　　LCD_LED_OFF　　：LED 消灯
　　　　LCD_LED_ON　　：LED 点灯
　　H　led_kind　　：ＬＥＤ点灯種別
　　　　LCD_LED_GREEN　：緑点灯
　　　　LCD_LED_RED　　：赤点灯

[リターンパラメータ]
　　ER　ercd　　：リターンコード

[リターンコード]
　　E_OK　　：正常終了
　　E_PRM　　：パラメータエラー

≪備考≫
　　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。
　　（緑点灯後の赤点灯は赤、赤点灯後の緑点灯は緑になります。）
　　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）
　　ただし、パラメータエラーの対象になりますので、0または1を必ず指定して下さい。

機能	ＥＬバックライトの制御	関数名	lcd_el

ＥＬバックライトの点灯／消灯を行います。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER　ercd = lcd_el(H el_mode);

[パラメータ]
　　H　el_mode　　：ＥＬモード
　　　　LCD_EL_OFF　　：EL 消灯
　　　　LCD_EL_ON　　：EL 点灯

[リターンパラメータ]
　　ER　ercd　　：リターンコード

[リターンコード]
　　E_OK　　：正常終了
　　E_PRM　　：パラメータエラー

≪備考≫

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す。

(2) 文字入力モード

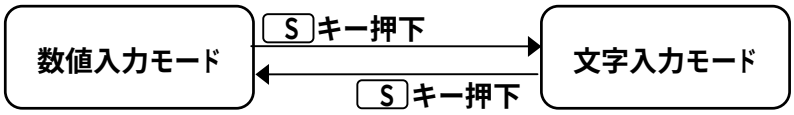
英字(A～Z、SP)、記号(－、$、／、＋、%、:、＊)、数値(0～9、.)の入力が可能です。
英記号は、めくり入力になっています。

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	ﾃﾞﾌｫﾙﾄｷｰ	コード		機　能	動作例	
		属性	ｺｰﾄﾞ		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 1	1234567890 123 1234567890 1234567890 1234 1234567890 1
→	→(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 1234567890 1234567890
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 abcd 1234567890 12345	1234567890 12 1234567890 1234567890 123456789a bcd 1234567890 2345
削除	ＤＥＬ(F4)	00h	10h	カーソル上の文字を削除します	1234567890 123 1234567890 12345678900 1234567890 1234567890 abcd	1234567890 13 1234567890 123456789 1234567890 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	ﾃﾞﾌｫﾙﾄｷｰ	コード		機　能	動作例	
		属性	ｺｰﾄﾞ		入力前	入力後
＋ (ﾌﾟﾗｽ)	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 1 1234567890	1234567890 123 1234567890 1 1234567890
－ (ﾏｲﾅｽ)	-(F3)	00h	2Dh	「-」(ﾏｲﾅｽ記号)を数値の最上位位置に付加します　(入力領域がフルの場合、無効になります)	1234567890 -123 1234567890 1 1234567890 1234567890 1234567890	1234567890 -123 1234567890 -1 1234567890 - 1234567890 1234567890
． (小数点)	.(F4)	00h	2Eh	カーソル位置に「.」(ﾋﾟﾘｵﾄﾞ記号)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 1 1234567890 12345 1234567890	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 1 1234567890 123456 1234567890	1234567890 12 1234567890 1234567890 1234561234
削除	なし	00h	10h	カーソル上の文字を削除します	1234567890 123 1234567890 1 1234567890 123456 1234567890	1234567890 12 1234567890 1234567890 1234561234

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	ｺｰﾄﾞ値		機　能	1文字入力 (ｺｰﾄﾞ返却)	文字列入力 (文字格納)	数値入力 (文字格納)
	属性	ｺｰﾄﾞ				
機能 ｺｰﾄﾞ	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	バーコード読込み開始(※1)			
制御 ｺｰﾄﾞ	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　バーコード読込み開始機能はﾄﾘｶﾞｰｷｰ／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	設定可能ｷｰｺｰﾄﾞ
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	ﾊﾞｰｺｰﾄﾞ読込みを除く全て
	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		
ﾄﾘｶﾞｰｷｰ	右ﾄﾘｶﾞｰｷｰ	不可	
	左ﾄﾘｶﾞｰｷｰ		

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	右ﾄﾘｶﾞｰｷｰ	不可
	左ﾄﾘｶﾞｰｷｰ	

※ マルチファンクションキーにＯＢＲ読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

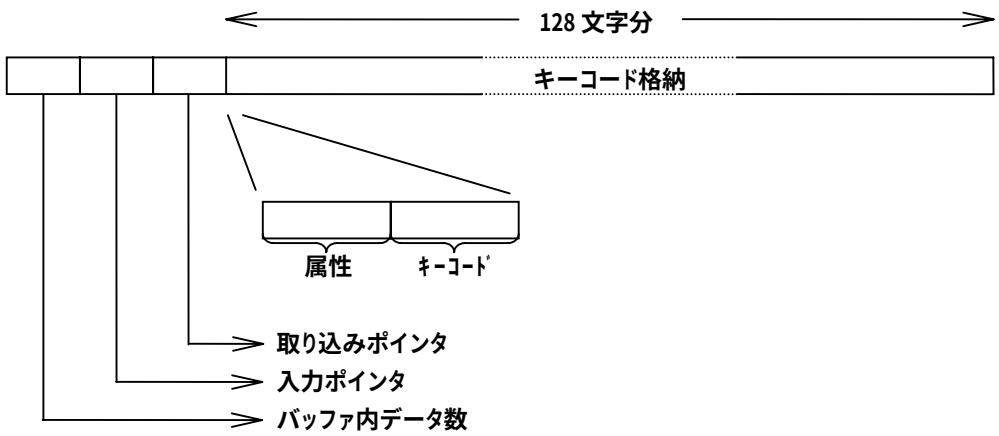
種別	キー	設定可能
ストロークキー	入力モード切替(S)	可能
	後退(BS)	
	クリア(CLR)	
	ﾃﾝｷｰ 1	
	ﾃﾝｷｰ 2	
	ﾃﾝｷｰ 3	
	ﾃﾝｷｰ 4	
	ﾃﾝｷｰ 5	
	ﾃﾝｷｰ 6	
	ﾃﾝｷｰ 7	
	ﾃﾝｷｰ 8	
	ﾃﾝｷｰ 9	
	ﾃﾝｷｰ 0	
	ﾃﾝｷｰ .	
	ﾃﾝｷｰ ENT	
	F1(－)	
	F2(←)	
	F3(→)	
	F4(DEL)	
	F5(SP)	
	F6(▲)	
	F7(▼)	
	F8(BL)	
マルチファンクションキー	ﾏﾙﾁﾌｧﾝｸｼｮﾝｷｰ R	可能
	ﾏﾙﾁﾌｧﾝｸｼｮﾝｷｰ L	
トリガーキー	右ﾄﾘｶﾞｰｷｰ	不可
	左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

※ 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、キークリック音もなりません。

※ マルチファンクションキーにＯＢＲ読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。

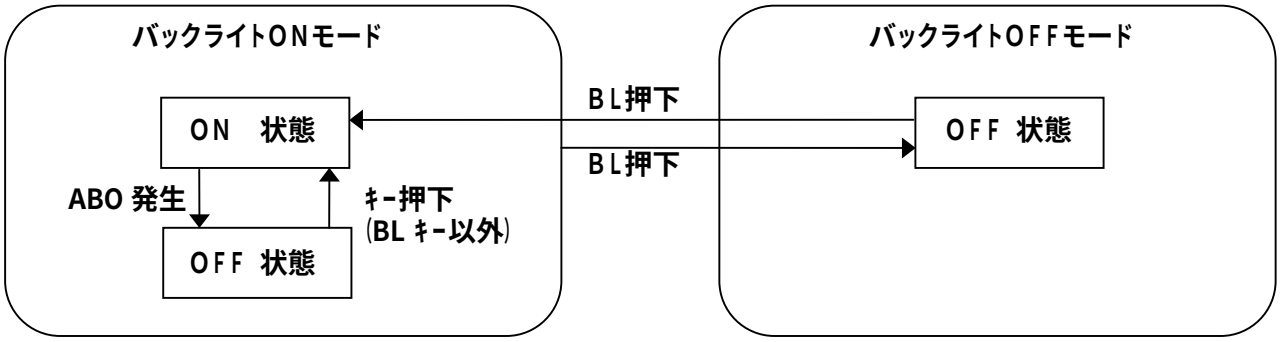


図４．１　バックライト状態遷移図

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	バーコード読込み開始	
OBR ｷｰと通常ｷｰ	バーコード読込み開始	
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込みます。全てのＯＢＲキーが離された時点でバーコード読込みを中止します。

・ 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）	バーコード読込み継続
通常キー	ＯＢＲキー	バーコード読込み開始	
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の入力は行われない)
↓
[1] キー解放　→ 2入力

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

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	次に続くｺｰﾄﾞがｱｽｷｰｺｰﾄﾞであることを示します
01h	次に続くｺｰﾄﾞがめくりｺｰﾄﾞで、1ﾊﾟﾀｰﾝ目のｱｽｷｰｺｰﾄﾞに変換されることを示します
02h	次に続くｺｰﾄﾞがめくりｺｰﾄﾞで、2ﾊﾟﾀｰﾝ目のｱｽｷｰｺｰﾄﾞに変換されることを示します
03h	次に続くｺｰﾄﾞがめくりｺｰﾄﾞで、3ﾊﾟﾀｰﾝ目のｱｽｷｰｺｰﾄﾞに変換されることを示します
04h	次に続くｺｰﾄﾞがめくりｺｰﾄﾞで、4ﾊﾟﾀｰﾝ目のｱｽｷｰｺｰﾄﾞに変換されることを示します
FFh	次に続くｺｰﾄﾞが内部処理ｺｰﾄﾞ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			ァ	キ	ヌ	ラ		
8	BS		(	8	H	X	h	x			ィ	ク	ネ	リ		
9			)	9	I	Y	i	y			ゥ	ケ	ノ	ル		
A	LF		*	:	J	Z	j	z			ェ	コ	ハ	レ		
B			+	;	K	[	k	{			ォ	サ	ヒ	ロ		
C	CL	→	,	<	L	¥	l	|			ャ	シ	フ	ワ		
D	CR	←	−	=	M	]	m	}			ュ	ス	ヘ	ン		
E			.	>	N	^	n	~			ョ	セ	ホ	゛		
F			／	?	O	_	o				ッ	ソ	マ	゜		

1文字入力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。

機能	1文字入力	関数名	key_read

キーバッファから1文字入力を行います。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

```
KEY_INP    *pkey_inp          :動作内容の先頭アドレス
```

[リターンパラメータ]

```
ER    ercd                    :リターンコード
```

[ストラクト構造]

```
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          : バーコード読込み完了      */
                                        /*    KEY_IO_EXT           : ＩＯボックス検出          */
                                        /*    KEY_CI_EXT           : ＣＩ信号検出              */
                                        /*    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 ;
```

[リターンコード]

```
                    :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           :バーコード読込み完了検出
E_KEY_IO            :ＩＯボックス検出
E_KEY_CI            :ＣＩ信号検出
E_PRM               :パラメータエラー
```

≪備考≫

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。

機能	文字列入力	関数名	key_string

キーバッファから文字列入力を行います。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

```
KEY_INPS   *pkey_inps          :動作内容の先頭アドレス
UB   *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)
```

[リターンパラメータ]

```
ER   ercd       :リターンコード
```

[ストラクト構造]

```
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          :バーコード読込み完了          */
                                /*    KEY_CLR_EXT          :ＣＬＲキー押下                */
                                /*    KEY_IO_EXT           :ＩＯボックス検出              */
                                /*    KEY_CI_EXT           :ＣＩ信号検出                  */
                                /*    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 ;    /*  入力文字位置 (半角)                                  */
        UH      clr_type ;      /*  数値入力用予約領域 (※3)                              */
                            } KEY_INPS ;
```

[リターンコード]

```
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_KEY_CI      :ＣＩ信号検出
E_PRM         :パラメータエラー
```

≪備考≫

※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。

機能	数値入力	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

```
KEY_INPS   *pkey_inps      :動作内容の先頭アドレス
UB   *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）
```

[リターンパラメータ]

```
ER   ercd                  :リターンコード
```

[ストラクト構造]

```
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          :バーコード読込み完了          */
                           /*     KEY_CLR_EXT          :ＣＬＲキー押下                */
                           /*     KEY_IO_EXT           :ＩＯボックス検出              */
                           /*     KEY_CI_EXT           :ＣＩ信号検出                  */
                           /*     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 ;
```

[リターンコード]

```
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_KEY_CI      :ＣＩ信号検出
E_PRM         :パラメータエラー
```

≪備考≫

※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、ﾌｫﾝﾄｻｲｽﾞ、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。

機能	キーバッファのステータスチェック	関数名	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。
読込みポインタは更新されません。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER　ercd = key_check(void);

[パラメータ]
　　なし

[リターンパラメータ]
　　ER　ercd　　:リターンコード

[リターンコード]
　　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

≪備考≫

機能	キーバッファのクリア	関数名	key_clear

キーバッファをクリアします。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER　ercd = key_clear(void);

[パラメータ]
　　なし

[リターンパラメータ]
　　ER　ercd　　:リターンコード

[リターンコード]
　　E_OK　　:正常終了

≪備考≫

機能	ファンクションキーコードの設定	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

```
UB   func_mode              :動作モード
                               FNC_SET     :設定
                               FNC_GET     :取得
UB   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
KEYFORM   *func_data        :ファンクションキーデータアドレス
```

[リターンパラメータ]

```
ER   ercd        :リターンコード
```

[ストラクト構造]

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

[リターンコード]

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

≪備考≫

機能	ファンクションキー通知モード設定	関数名	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。

≪C言語インタフェース≫

【コーリングシーケンス】

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

【パラメータ】

```
UB   mode          :動作モード
                        FNC_MODE_SET     :設定
                        FNC_MODE_CLR     :解除
                        FNC_MODE_RED     :取得
UB   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
ID   *flgid        :イベントフラグID      (解除時は不要です)
UW   *setptn       :セットするビットパターン (解除時は不要です)
```

【リターンパラメータ】

```
ER   ercd      :リターンコード
```

【リターンコード】

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

≪備考≫

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。

機能	キー入力モード設定	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

```
UB   mode              :動作モード
                          SEL_SET   :  設定
                          SEL_GET   :  取得
                          SEL_RES   :  解除 (全て有効)
KEYSEL *key_sel        :有効無効キーテーブルアドレス
```

[リターンパラメータ]

```
ER   ercd              :リターンコード
```

[ストラクト構造]

```
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 ;     /*  F1 (―)                     */
        UB      func2 ;     /*  F2 (←)                     */
        UB      func3 ;     /*  F3 (→)                     */
        UB      func4 ;     /*  F4 (DEL)                    */
        UB      func5 ;     /*  F5 (SP)                     */
        UB      func6 ;     /*  F6 (▲)                      */
        UB      func7 ;     /*  F7 (▼)                      */
        UB      func8 ;     /*  F8 (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   */
                            } KEYSEL ;
```

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)

[リターンコード]

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

≪備考≫

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	FFMMMMMNNNNNC」	F:ｶﾝﾄﾘｰﾌﾗｸﾞ
	EAN-13	13	FFMMMMMNNNNNC」	M:生産者ｺｰﾄﾞ
	JAN-8	8	FFMMMNC」	N:商品コード
	EAN-8	8	FFMMMNC」	S:ｼｽﾃﾑﾒﾝﾊﾞｰｷｬﾗｸﾀ
	JAN-13 addon+2	15	FFMMMMMNNNNNCAA」	A:addon ﾃﾞｰﾀ
	EAN-13 addon+2	15	FFMMMMMNNNNNCAA」	｣:終了ｺｰﾄﾞ(CR または LF または CR+LF)
	JAN-13 addon+5	18	FFMMMMMNNNNNCAAAAA」	UPC-B を除きﾁｪｯｸﾃﾞｼﾞｯﾄ(mod 10)の計算は
	EAN-13 addon+5	18	FFMMMMMNNNNNCAAAAA」	必ず行われます
	JAN-8 addon+2	10	FFMMMMNCAA」	
	EAN-8 addon+2	10	FFMMMMNCAA」	
	JAN-8 addon+5	13	FFMMMMNCAAAAA」	
	EAN-8 addon+5	13	FFMMMMNCAAAAA」	
	UPC-A	12	0SMMMMMNNNNNC」	
	UPC-B	12	0SMMMMMNNNNNN」	
	UPC-A addon+2	14	0SMMMMMNNNNNCAA」	
	UPC-B addon+2	14	0SMMMMMNNNNNNAA」	
	UPC-A addon+5	17	0SMMMMMNNNNNCAAAAA」	
	UPC-B addon+5	17	0SMMMMMNNNNNNAAAAA」	
	UPC-E	(7),8	0MMNNNMC」	最後の M:0～2
		(7),8	0MMMNN3C」	
		(7),8	0MMMMN4C」	
		(7),8	0MMMMMNC」	最後の N:5～9
	UPC-E addon+2	(9),10	0MMNNNMCAA」	最後の M:0～2
		(9),10	0MMMNN3CAA」	
		(9),10	0MMMMN4CAA」	
		(9),10	0MMMMMNCAA」	最後の N:5～9
	UPC-E addon+5	(12),13	0MMNNNMCAAAAA」	最後の M:0～2
		(12),13	0MMMNN3CAAAAA」	
		(12),13	0MMMMN4CAAAAA」	
		(12),13	0MMMMMNCAAAAA」	最後の N:5～9
	UPC-E(+UPC-A)	6+12	MMNNNMC」SMMM0000NNNC」	6 桁目の M:0～2
		6+12	MMMNN3C」SMMM00000NNC」	
		6+12	MMMMN4C」SMMMM00000NC」	
		6+12	MMMMMNC」SMMMMM0000NC」	6 桁目の N:5～9
CODE-39		3～40	SBBB……BBCS」	A:ﾃﾞｰﾀ,　B:FULL ASCII 変換後ﾃﾞｰﾀ
		3～40	SAAA……AACS」	C:ﾁｪｯｸﾃﾞｼﾞｯﾄ(mod 43)
		1～38	BBB……BBC」	ﾁｪｯｸﾃﾞｼﾞｯﾄなしの場合は､ﾃﾞｰﾀとなります
		1～38	AAA……AAC」	S:ｽﾀｰﾄ･ｽﾄｯﾌﾟｷｬﾗｸﾀ(*)
NW-7		3～40	SDDD……DDDE」	S:ｽﾀｰﾄｺｰﾄﾞ(a,b,c,d のいづれか)
		1～38	DDD……DDD」	E:ｴﾝﾄﾞｺｰﾄﾞ(a,b,c,d のいづれか)
				D:ﾃﾞｰﾀ
Interleaved 2 of 5		2～40	DDD……DDDC」	D:ﾃﾞｰﾀ
				C:ﾁｪｯｸﾃﾞｼﾞｯﾄ(mod 10)
				ﾁｪｯｸﾃﾞｼﾞｯﾄなしの場合は､ﾃﾞｰﾀとなります
				読取桁数は偶数桁のみ
Industrial 2 of 5		2～40	DDD……DDDC」	D:ﾃﾞｰﾀ
				C:ﾁｪｯｸﾃﾞｼﾞｯﾄ(mod 10)
				ﾁｪｯｸﾃﾞｼﾞｯﾄなしの場合は､ﾃﾞｰﾀとなります
				読取桁数は偶数桁のみ
CODE-93		1～40	DDD……DDD」	D:ﾃﾞｰﾀ
CODE-128		1～64	AAA……AAA」	A:ASCII 変換後ﾃﾞｰﾀ,　B:ASCII 変換前ﾃﾞｰﾀ
		1～64	SBBB……BBCE」	C:ﾁｪｯｸﾃﾞｼﾞｯﾄ(mod 47)
				S:ｽﾀｰﾄｺｰﾄﾞ,　E:ｴﾝﾄﾞｺｰﾄﾞ
MSI		1～40	DDD……DDCC」	D:ﾃﾞｰﾀ
				C:ﾁｪｯｸﾃﾞｼﾞｯﾄ(mod 10,mod 11)
				ﾁｪｯｸﾃﾞｼﾞｯﾄなしの場合は､ﾃﾞｰﾀとなります
IATA		1～40	PADDDDDDD……C」	P:ｸｰﾎﾟﾝ NO.
				A:ｴｱﾗｲﾝ NO.
				D:ﾃﾞｰﾀ
				C:ﾁｪｯｸﾃﾞｼﾞｯﾄ(IATA 仕様)
				ﾁｪｯｸﾃﾞｼﾞｯﾄなしの場合はﾃﾞｰﾀとなります

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
	チェックデジッドの計算
	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)

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ないます。

Interleaved 2of5 で奇数桁の指定をした場合、最小桁は、指定＋1の偶数まで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

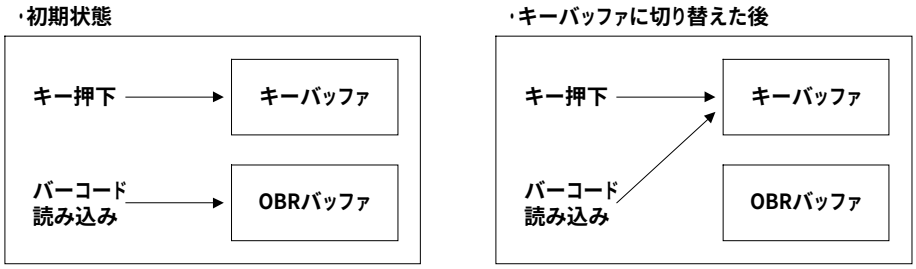
表5．6 設定範囲 (複数のコードを同時に設定した場合)

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、切り替えを行なってから 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	設定内容	初期状態
CODE-39	スタート／ストップキャラクタの出力の有無を設定します	出力有り
	Full ASCII変換の有無を設定します	変換無し
NW-7	スタート／ストップキャラクタの出力の有無を設定します	出力有り
UPC-E	UPC-Aの復元コードの出力の有無を設定します	出力無し
CODE-128	変換前データ／変換後データ(ASCII)のどちらを出力するかを設定します	変換後データ

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チェックデジッドの計算

各コードごとにチェックデジッドの計算を有効／無効にすることができます。
(チェックデジッドの計算: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 制御を向こう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
- チェックデジッド指定時のチェックデジッドエラー
- CODE-39、CODE-93 における Full ASCII 変換エラー

⑮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取りが行われます。

⑯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す。
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

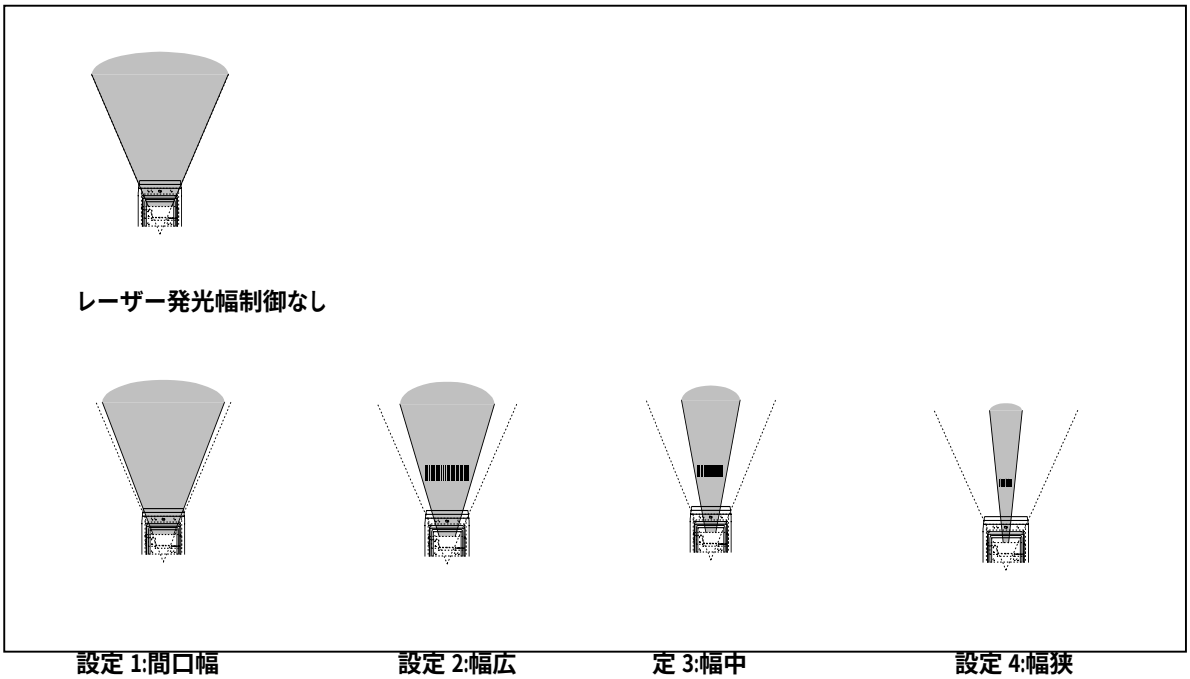
機能	内容	備考
ｷｬﾘﾌﾞﾚｰｼｮﾝ機能	マシン固体差によるレーザー発光幅位置を決定するための機能です	システムメニューまたはアプリケーションから起動可能です(*1)
レーザー発光幅設定機能	4段階のレーザー発光幅を設定するための機能です（レーザー発光幅制御なしの設定も可能）	関数として提供されます
レーザー発光幅微調整機能	現状設定されているレーザー発光幅を微調整し、広める/狭めることが可能です	関数として提供されます

（＊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、ﾊﾞｰｺｰﾄﾞの OPEN／CLOSE 時にも保存されています。

※ﾚｼﾞｭｰﾑ 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。

5.3. 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。

機能	ＯＢＲオープン	関数名	OBR_open

ＯＢＲのオープンを行います。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER　ercd = OBR_open (UW mode);

[パラメータ]
　　UW　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
　　・ﾁｪｯｸﾃﾞｼﾞｯﾄ実行指示
　　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。

[リターンパラメータ]
　　ER　ercd　　:リターンコード

[リターンコード]
　　E_OK　　:正常終了
　　E_OBR_PON　　:オープン済み

≪備考≫

機能	ＯＢＲクローズ	関数名	OBR_close

ＯＢＲの終了処理を行います。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER　ercd = OBR_close (void);

[パラメータ]
　　なし

[リターンパラメータ]
　　ER　ercd　　:リターンコード

[リターンコード]
　　E_OK　　:正常終了
　　E_OBR_POF　　:クローズ済み

≪備考≫

機能	ＯＢＲデータ1文字リード	関数名	OBR_getc

ＯＢＲデータを１文字読込みます。

≪C言語インタフェース≫

[コーリングシーケンス]
　　UB　dat = OBR_getc (UW *rcd);

[パラメータ]
　　UW　*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

[リターンパラメータ]
　　UB　dat　　：ＯＢＲデータ(1文字)

≪備考≫

機能	ＯＢＲデータ文字列リード	関数名	OBR_gets

ＯＢＲデータを文字列で読込みます。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

```
UB   *buff        :ＯＢＲデータ出力先へのポインタ
UW   *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

UB   *lengs       :データバイト数格納先へのポインタ
```

[リターンパラメータ]

```
ER   dat          :リターンコード
```

[リターンコード]

```
E_OK              :正常終了
```

≪備考≫

機能	ＯＢＲバッファのクリア	関数名	OBR_flush

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

≪C言語インタフェース≫

[コーリングシーケンス]
```
ER    ercd = OBR_flush ( void );
```

[パラメータ]
なし

[リターンパラメータ]
```
ER    ercd          :リターンコード
```

[リターンコード]
```
E_OK                :正常終了
```

≪備考≫

機能	ＯＢＲバッファステータスチェック	関数名	OBR_stat

ＯＢＲバッファの状態を読込みます。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER　ercd = OBR_stat (W *leng ,　UB *lcnt);

[パラメータ]
　　W　*leng　　:バッファ内の残りバイト数格納先へのポインタ
　　UB　*lcnt　　:バッファ内の残り段数格納先へのポインタ

[リターンパラメータ]
　　ER　ercd　　:リターンコード

[リターンコード]
　　E_OK　　:正常終了

≪備考≫

機能	ＯＢＲ動作モードの取得	関数名	OBR_moderd

ＯＢＲの動作モードを読込みます。

≪C言語インタフェース≫

[コーリングシーケンス]
```
ER    ercd = OBR_moderd ( M_TBL *modtbl );
```

[パラメータ]
```
M_TBL    *modtbl        :動作モードテーブル格納エリアへのポインタ
```

[リターンパラメータ]
```
ER    ercd              :リターンコード
```

[リターンコード]
```
E_OK                    :正常終了
```

≪備考≫

機能	ＯＢＲ動作モードの設定	関数名	OBR_modewt

ＯＢＲの動作モードを設定します。
①読取り誤動作を防止するため、ＯＢＲオープン中の動作モード設定は禁止します。
また、動作モード設定時、ＯＢＲバッファ内にデータが残っていてはいけません。
②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。）

≪C言語インタフェース≫

[コーリングシーケンス]
```
ER    ercd = OBR_modewt ( M_TBL *modtbl );
```

[パラメータ]

M_TBL	*modtbl	:動作モードテーブル格納エリアへのポインタ

[リターンパラメータ]

ER	ercd	:リターンコード

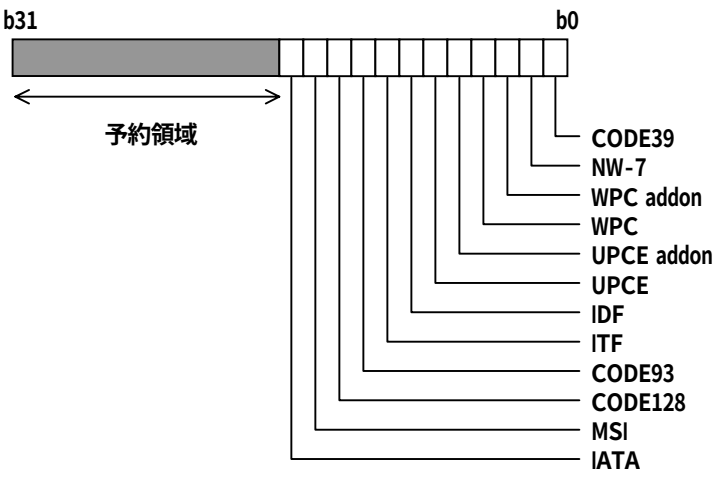
[リターンコード]

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

≪備考≫

動作モードテーブル

<table>
<tr><th>項目</th><th>内容</th><th>初期値</th><th>設定</th><th>参照</th></tr>
<tr><td>読取りコード

(4バイト)</td><td>


＊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。
</td><td>全て
選択
(FFFh)</td><td>○</td><td>○</td></tr>
<tr><td>読取り桁数の設定

出力フォーマット
の設定

チェックデジット
の実行指定

チェックキャラクタの
出力指定

(192バイト)</td><td>
<table>
<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
ﾃﾞｼﾞｯﾄ</th><th>ﾁｪｯｸ
ｷｬﾗｸﾀ</th><th></th></tr>
<tr><td>FFh</td><td>2</td><td>48 *7</td><td>0 *1</td><td>0</td><td>1</td><td>CODE-39</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>-</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>-</td><td>1</td><td>-</td><td>WPC(UPCE以外)</td></tr>
<tr><td>FFh</td><td>9</td><td>12</td><td>-</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 addon</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>Interleaved 2of5</td></tr>
<tr><td>FFh</td><td>3</td><td>40</td><td>-</td><td>1</td><td>-</td><td>CODE-93</td></tr>
<tr><td>FFh</td><td>2</td><td>64</td><td>0 *4</td><td>1</td><td>-</td><td>CODE-128</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></td><td></td><td></td><td></td><td></td><td></td><td>リザーブ</td></tr>
<tr><td>FFh</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td></td></tr>
</table>
- 指定なし (各種コードの固定値)
変更不可 (値のチェックなし)

ﾁｪｯｸﾃﾞｼﾞｯﾄ　0:ﾁｪｯｸ無し　1:ﾁｪｯｸ有り
ﾁｪｯｸｷｬﾗｸﾀ　0:出力無し　1:出力有り

*1 CODE39 出力ﾌｫｰﾏｯﾄ　0:Start/Stopコード有り
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
*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
*3 UPCE 出力ﾌｫｰﾏｯﾄ　0:UPCA への復元ｺｰﾄﾞ出力有り
1:UPCA への復元ｺｰﾄﾞ出力無し
*4 CODE128 出力ﾌｫｰﾏｯﾄ 0:変換後(ASCII)のﾃﾞｰﾀを出力
1:変換前のﾃﾞｰﾀを出力
変換前/後の対応は本仕様書最終頁参照
*5 MSIﾁｪｯｸﾃﾞｼﾞｯﾄ　0:ﾁｪｯｸﾃﾞｼﾞｯﾄ無し
1:1 桁, mod10
2:2 桁, 1st : mod11 2nd : mod 10
3:2 桁, 1st : mod10 2nd : mod 10
*6 IATAﾁｪｯｸﾃﾞｼﾞｯﾄ　0:ﾁｪｯｸﾃﾞｼﾞｯﾄ無し(読取り桁数1-40)
1:最終ｷｬﾗｸﾀ以外を対象(読取り桁数2-40)
2:ｸｰﾎﾟﾝ番号/数値部を対象(読取り桁数15-17)
3:数値部を対象(読取り桁数15-17)
*7 CODE39 読取桁数　OS 1.02/PATCH 1.08以上ならばMAX48桁
それ以前ならばMAX38桁
</td><td>左の表
参照</td><td>○</td><td>○</td></tr>
</table>

項目	内容	初期値	参照	設定
読取り方式の設定 (1バイト)	b7　b0 00h:単発読み 01h:連続読み(ﾄﾘｶﾞｷｰあり)	連続読み (01h)	○	○
ブザー制御の設定 (1バイト)	b7　b0 00h:ブザー制御なし 01h:ブザー制御あり	ブザーあり (01h)	○	○
LED制御の設定 (1バイト)	b7　b0 00h:LED制御なし 01h:LED制御あり 02h:LED制御あり(ｴﾗｰ除く)	LED あり (01h)	○	○
出力バッファ の参照 (1バイト)	b7　b0 00h:OBRバッファに出力 01h:(予約) 02h:KEYバッファに出力(※1)	OBR ﾊﾞｯﾌｧ (00h)	○	×
終了コードの設定 (ﾊﾞｰｺｰﾄﾞの最後尾に 付加するｺｰﾄﾞ) (1バイト)	b7　b0 00h:CR 01h:LF 02h:CR＋LF	CR のみ (00h)	○	○
読取り動作の設定 (1バイト)	b7　b0 00h:通常読み 01h:段数読み	通常読み (00h)	○	○

(※1) 設定は OBR_chgbuf 関数を使用して下さい。

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。

機能	ＯＢＲバッファの切替え	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

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

[リターンパラメータ]

```
ER    ercd        :リターンコード
```

[リターンコード]

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

≪備考≫

機能	トリガーキーによる電源オン設定	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER　ercd = OBR_trigmode (UH trigmode);

[パラメータ]
　　UH　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	×	○	×

[リターンパラメータ]
　　ER　ercd　　：リターンコード

[リターンコード]
　　E_OK　　：正常終了
　　E_PRM　　：パラメータエラー

≪備考≫

機能	レーザー発光幅の設定	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER　ercd = OBR_swing (H mode, H ＊pattern);

[パラメータ]
　　H　mode　　:機能選択
　　　　SW_SET　　:設定
　　　　SW_READ　　:取得
　　H　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)

[リターンパラメータ]
　　ER　ercd　　:リターンコード

[リターンコード]
　　E_OK　　:正常終了
　　E_PRM　　:パラメータエラー

≪備考≫
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。

機能	レーザー発光幅の微調整	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER　ercd = OBR_widenarrow(UH mode);

[パラメータ]
　　UH　mode　　:機能選択
　　　　SW_WIDE　　:広くする
　　　　SW_NARROW　　:狭める

[リターンパラメータ]
　　ER　ercd　　:リターンコード

[リターンコード]
　　E_OK　　:正常終了

≪備考≫
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。

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4つの通信ポートが存在します。通信関数部ではシリアルインタフェース、カシオIRインタフェース、PHS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。

通信ﾎﾟｰﾄ	制御形式	ｺﾈｸﾀ	COM No	規格	同期方式	転送速度(bps)	ｷｬﾗｸﾀﾚﾝｸﾞｽ	ﾊﾟﾘﾃｨﾋﾞｯﾄ	ｽﾄｯﾌﾟﾋﾞｯﾄ	信号
カシオ IR ｲﾝﾀﾌｪｰｽ	半二重	IrDA	COM0	ｶｼｵｵﾘｼﾞﾅﾙ	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)
シリアル ｲﾝﾀﾌｪｰｽ	全二重	14Pin	COM1	ｶｼｵｵﾘｼﾞﾅﾙ	調歩	1.2k 2.4k 4.8k 9.6k 19.2 38.4k 57.6k 115.2k	7bit 8bit	NON ODD EVN	1bit 2bit	SD RD ER RS DR CS CD CI
PHS ｲﾝﾀﾌｪｰｽ	全二重	18Pin	COM3	ｶｼｵｵﾘｼﾞﾅﾙ	調歩	1.2k 2.4k 4.8k 9.6k 19.2 38.4k	7bit 8bit	NON ODD EVN	1bit 2bit	SD RD ER RS DR CS CD CI WP
IrDA	半二重	IrDA	ー	IrDA (IrSIR 1.2)	調歩	2.4k 9.6k 19.2 38.4k 57.6k 115.2k	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、シリアル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	シリアルインタフェース	IrDA
ｶｼｵ IR インタフェース		同時不可	同時不可
シリアルインタフェース	同時不可		同時不可
IrDA	同時不可	同時不可	
PHSインタフェース	同時可能	同時可能	同時可能

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

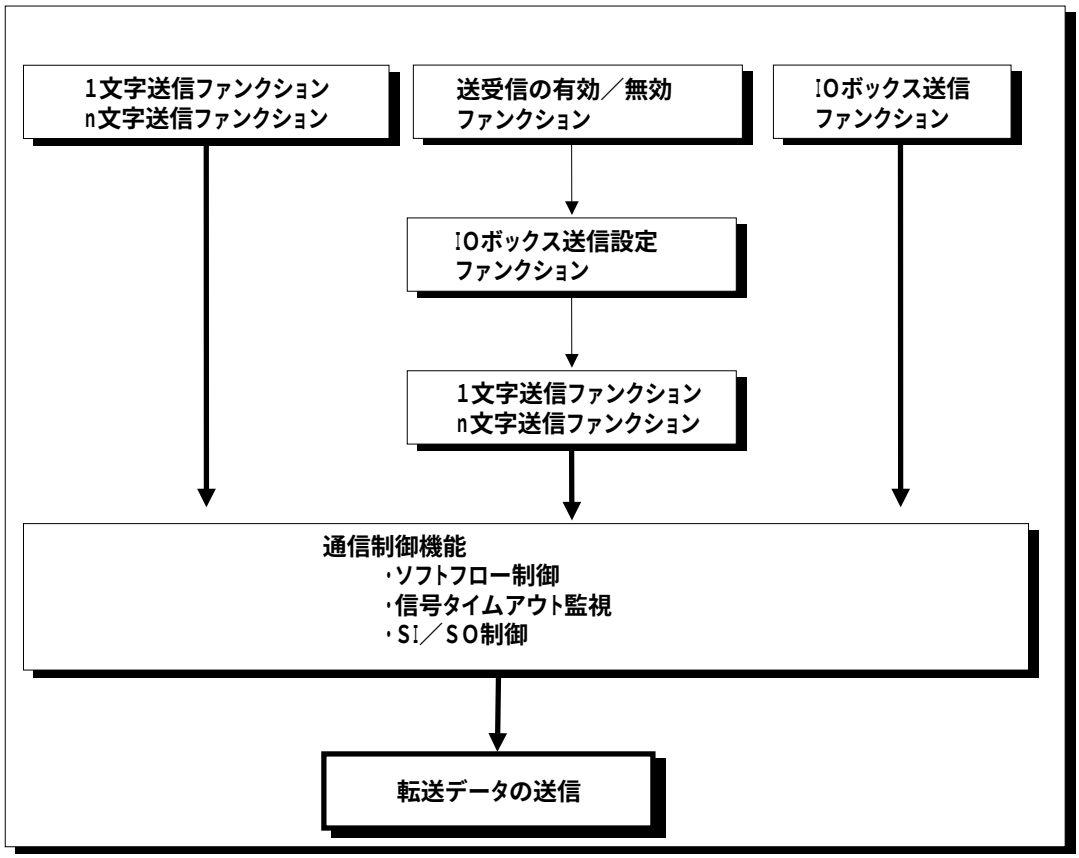
通信ポートのオープンを行うと、通信ポートに電源供給を行い、通信リソースの占有を行います。
これを“ポートオープン”状態とします。ポートオープン状態であるとき、同一（排他関係）の通信ポートをオープンすることはできませんし、本機の通信関数部では通信ポートを“ポート占有状態”にすることができます。ポート占有状態であるとき通信ポートへの電源供給、通信リーソースの占有は行わず、同一（排他関係）の通信ポートのオープンの抑制のみを行います。
例えば、通信関数部以外で通信部が使用する通信リソースを使用してデータ通信を行う場合に通信ポートを占有状態にして、通信部が使用できないようにします。
通信ポートの占有は「COM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

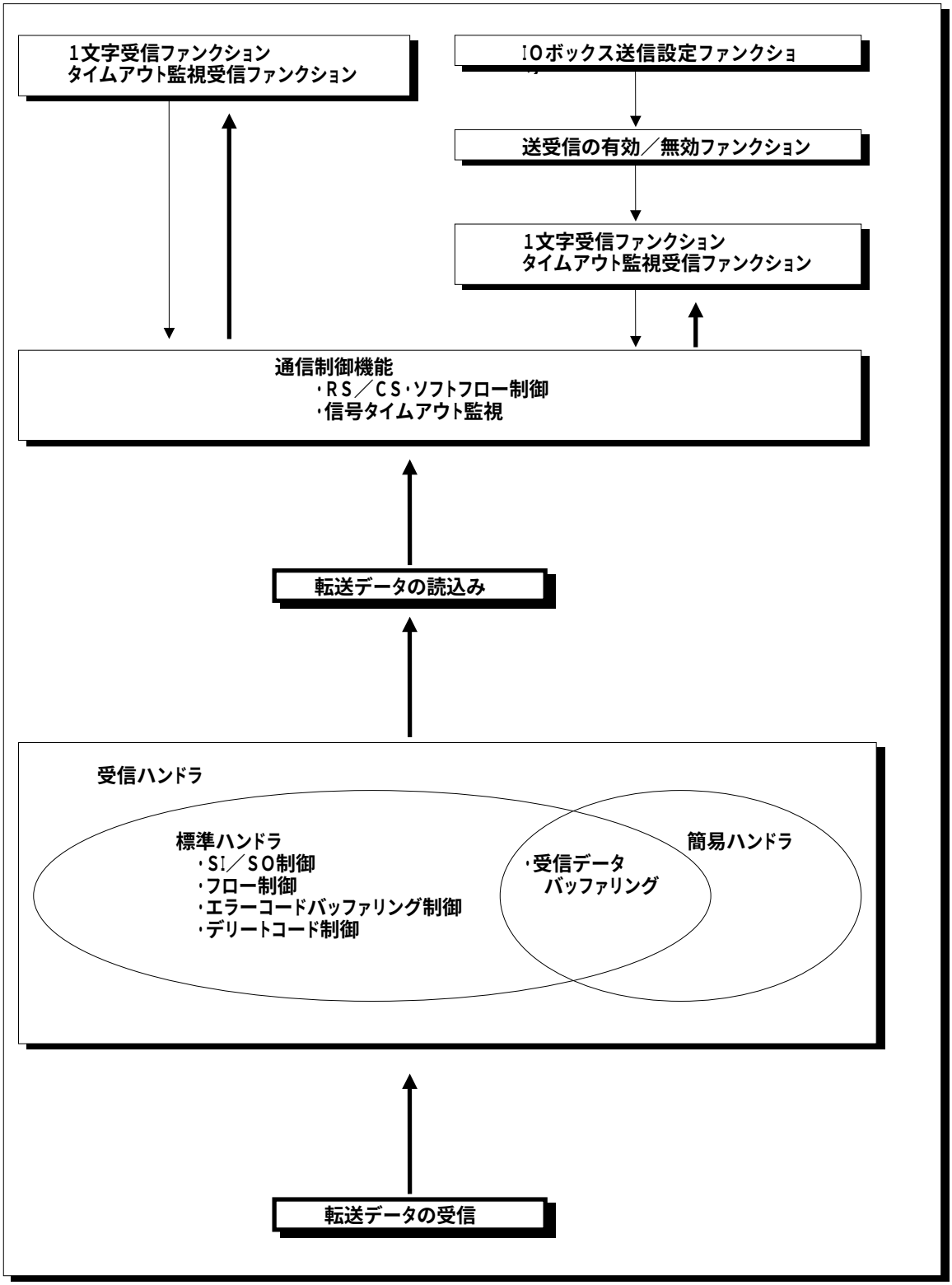
受信ハンドラでは以下の信号線の操作および参照を行います。

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
CI	ON／OFF状態を参照

③受信データの読込み

受信バッファに格納された転送データの読込みを「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。

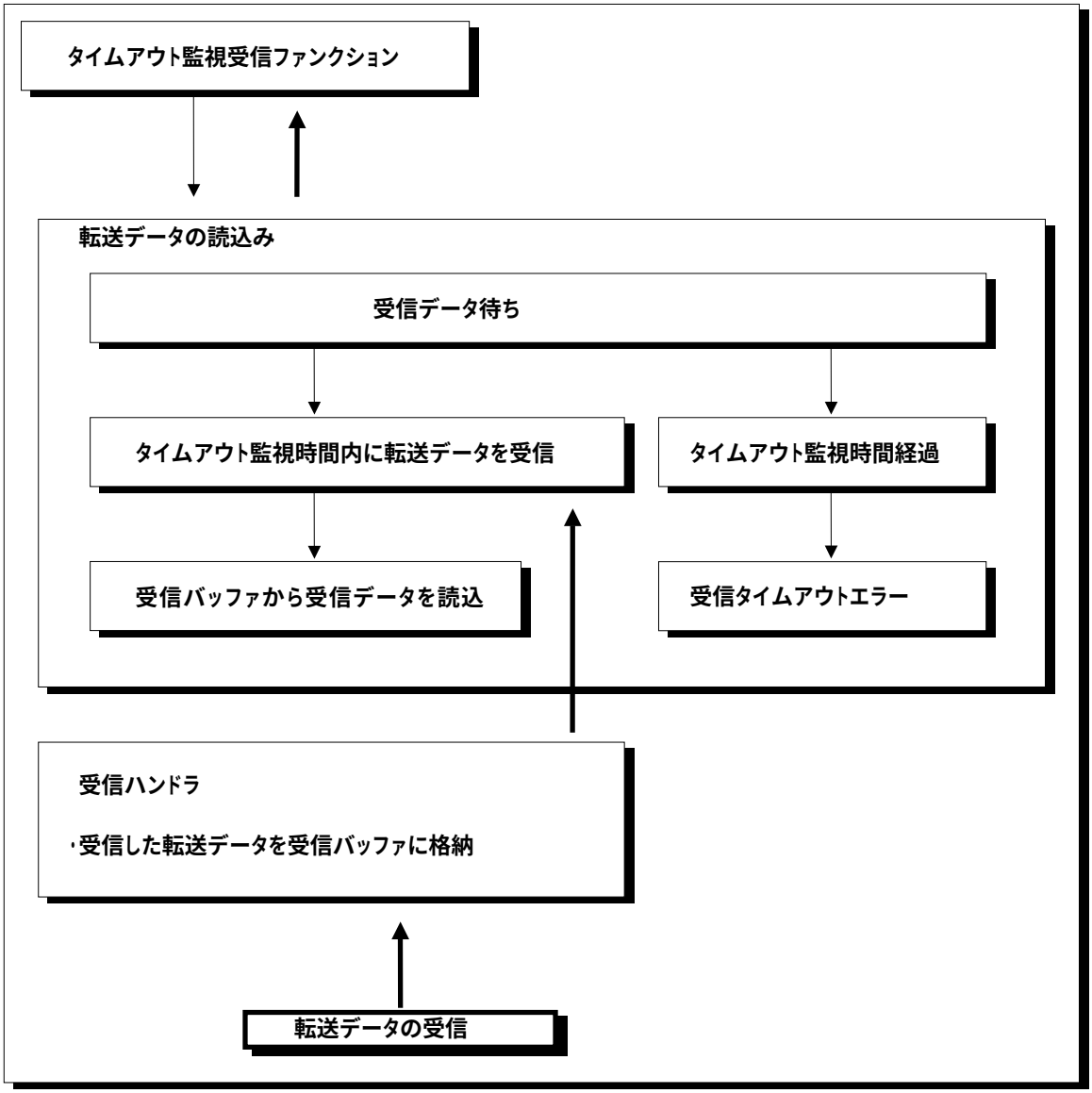
【転送データの受信および受信データの読込みの流れ】



④タイムアウト監視

読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。
「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。

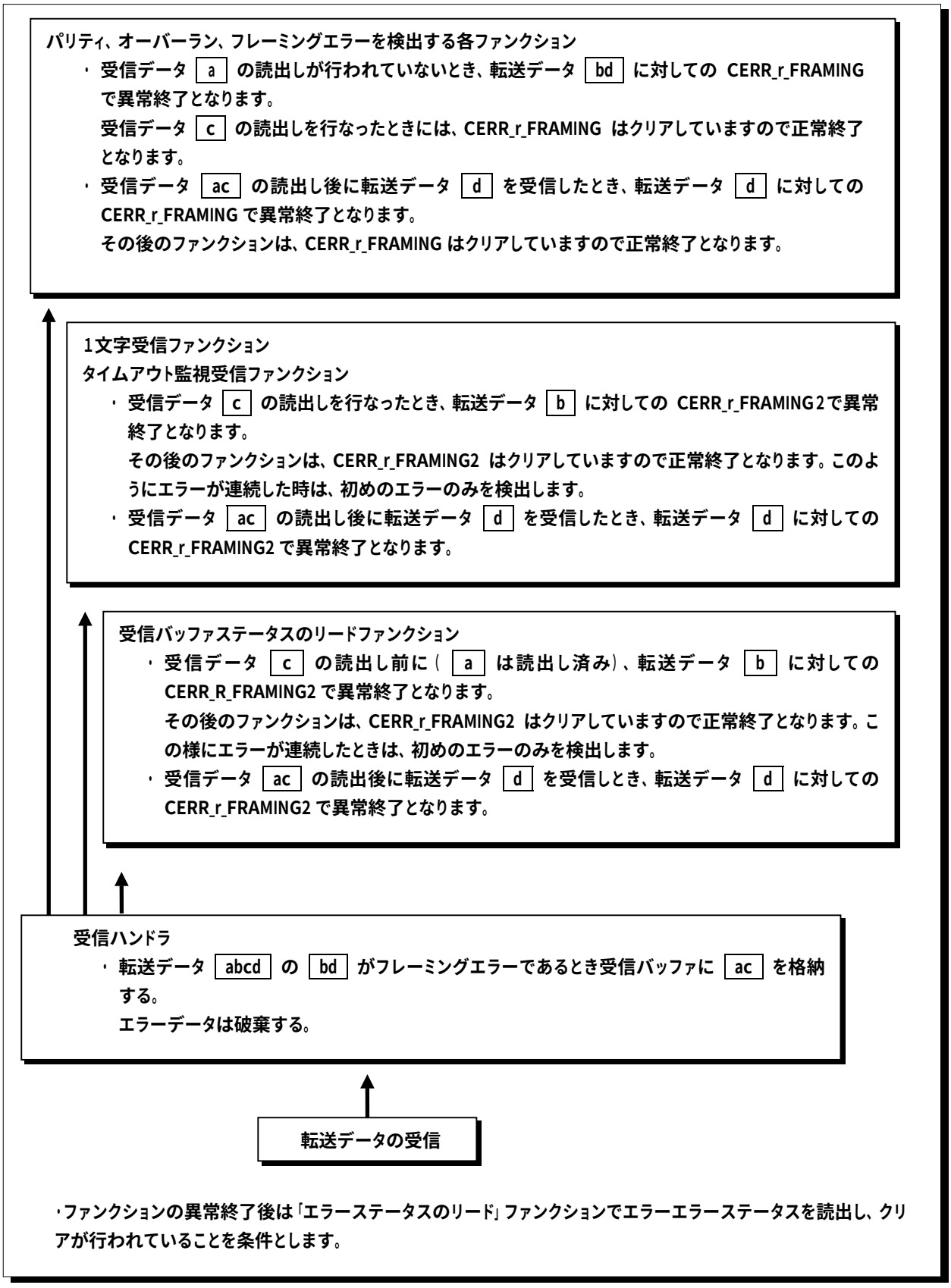
b) CERR__r__PARITY2、OVERRUN2、FRAMING2エラーステータス

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。

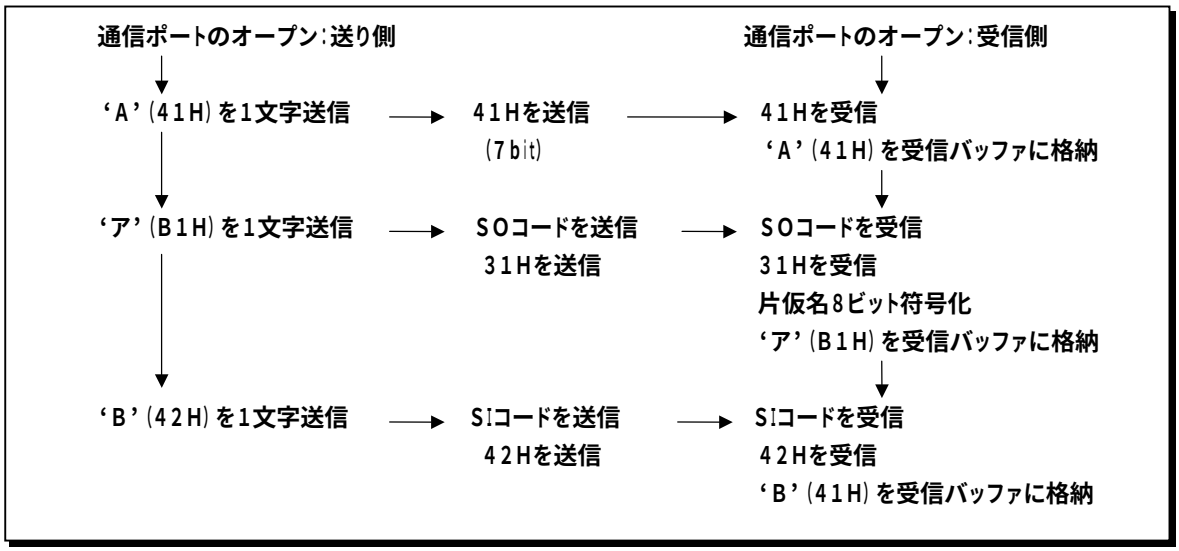
［パリティ、オーバーラン、フレーミングエラーの検出の流れ］



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JIS 8ビット符号化文字（JIS　X　0201参照）を扱う場合に使用します。
JIS 8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

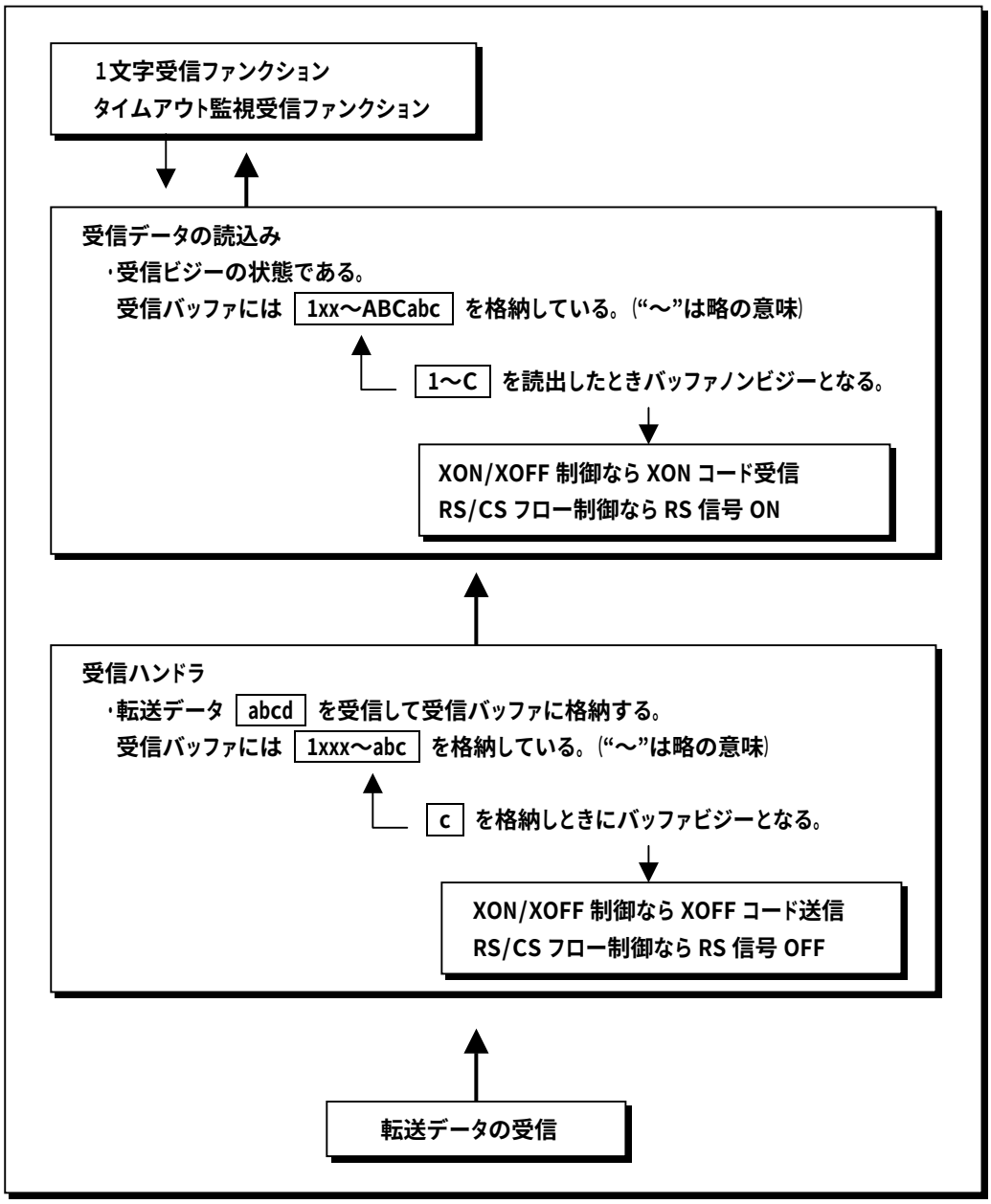
(1) XON/XOFF 制御

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。

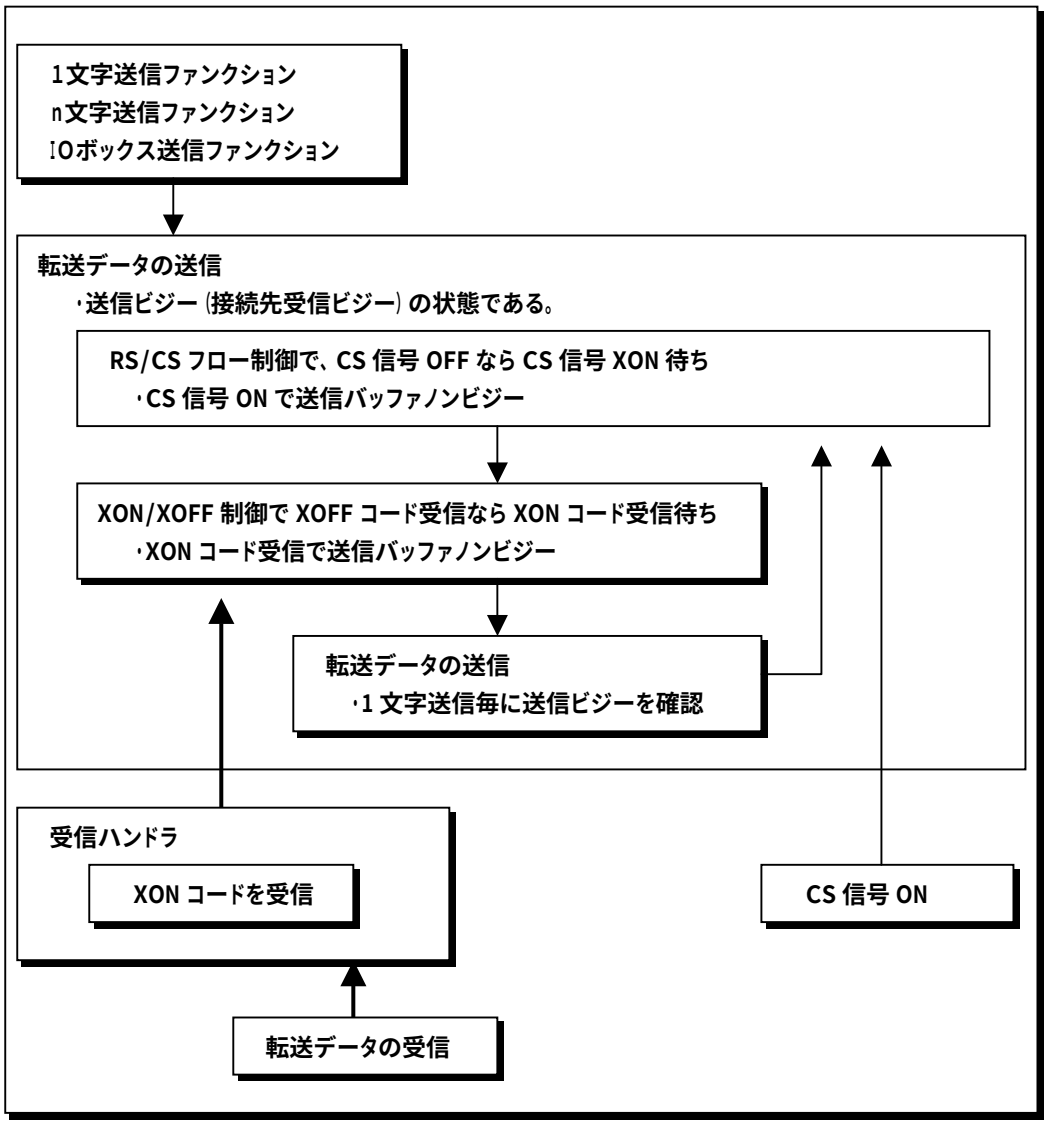
(2) RS／CSフロー制御

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】


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

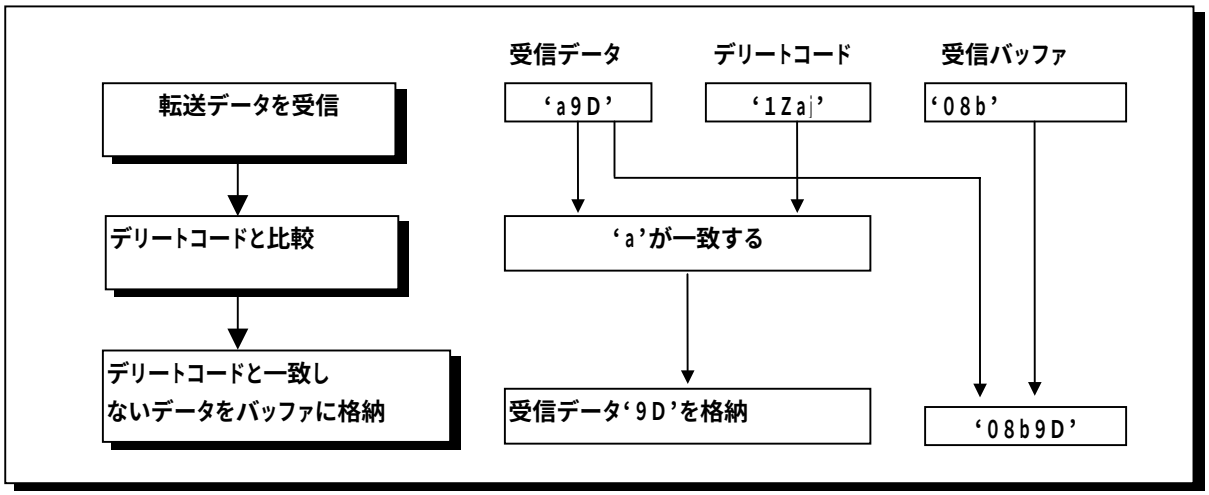
【送信フロー制御の流れ】


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

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。

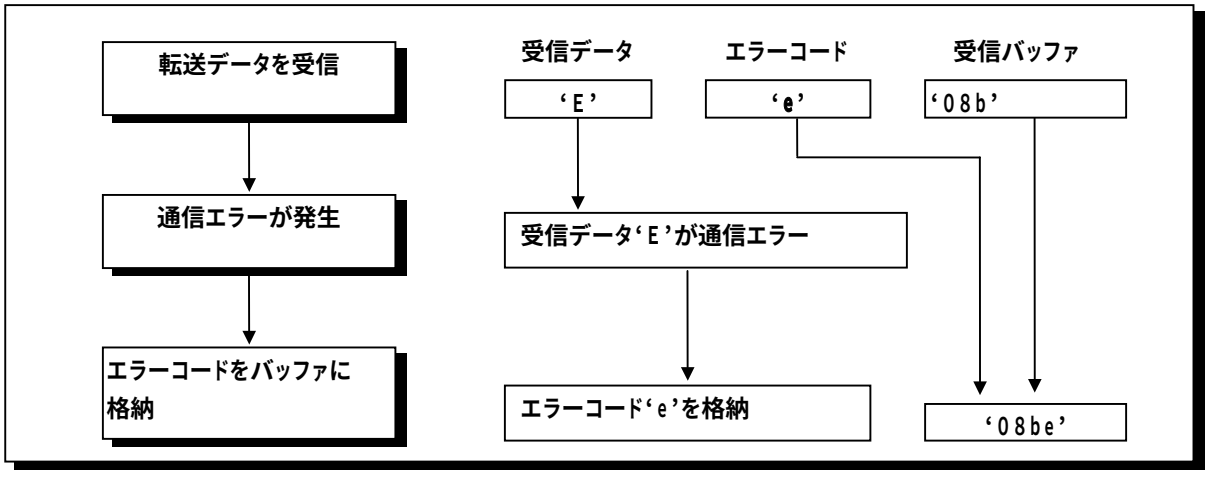
［デリードコードの制御のながれ］



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。

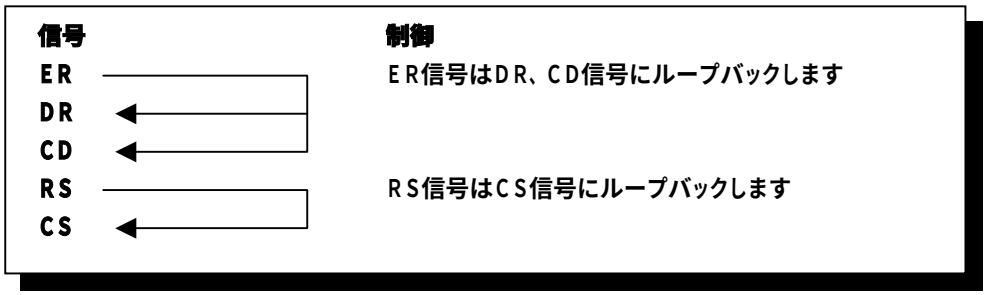
【通信ポートが制御する信号線】

信号線＼通信ポート	カシオIRインタフェース	シリアルインタフェース	PHSインタフェース
ER	あり（仮想）	あり	あり
RS	あり（仮想）	あり	あり
DR	あり（仮想）	あり	あり
CS	あり（仮想）	あり	あり
CD	あり（仮想）	あり	あり
CI	なし	あり	あり
CTRL	あり	なし	なし

b) カシオIRポートの仮想信号制御

以下にカシオIRポートの仮想信号制御方法を示します。

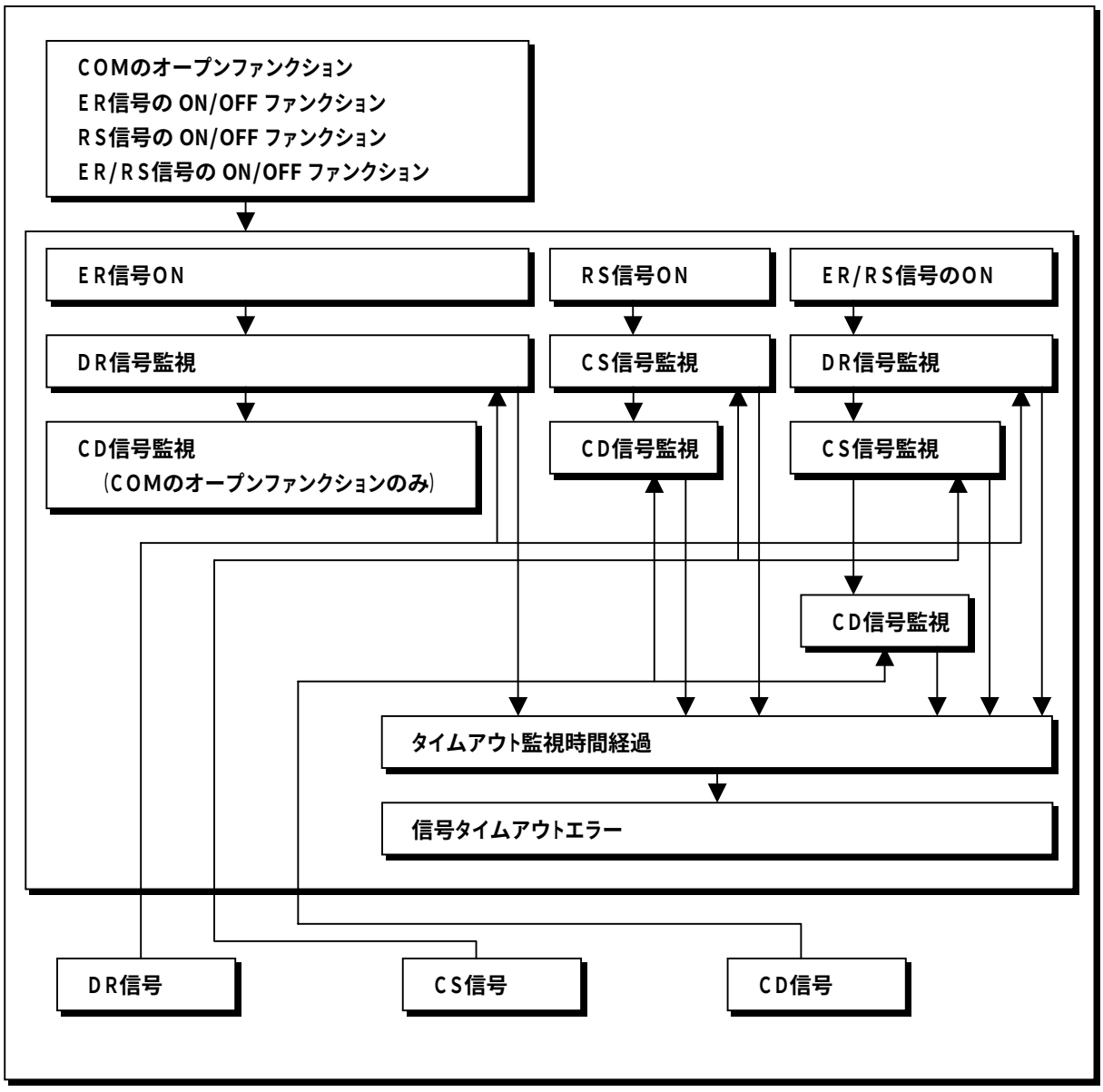
【カシオIRポートの仮想信号制御】



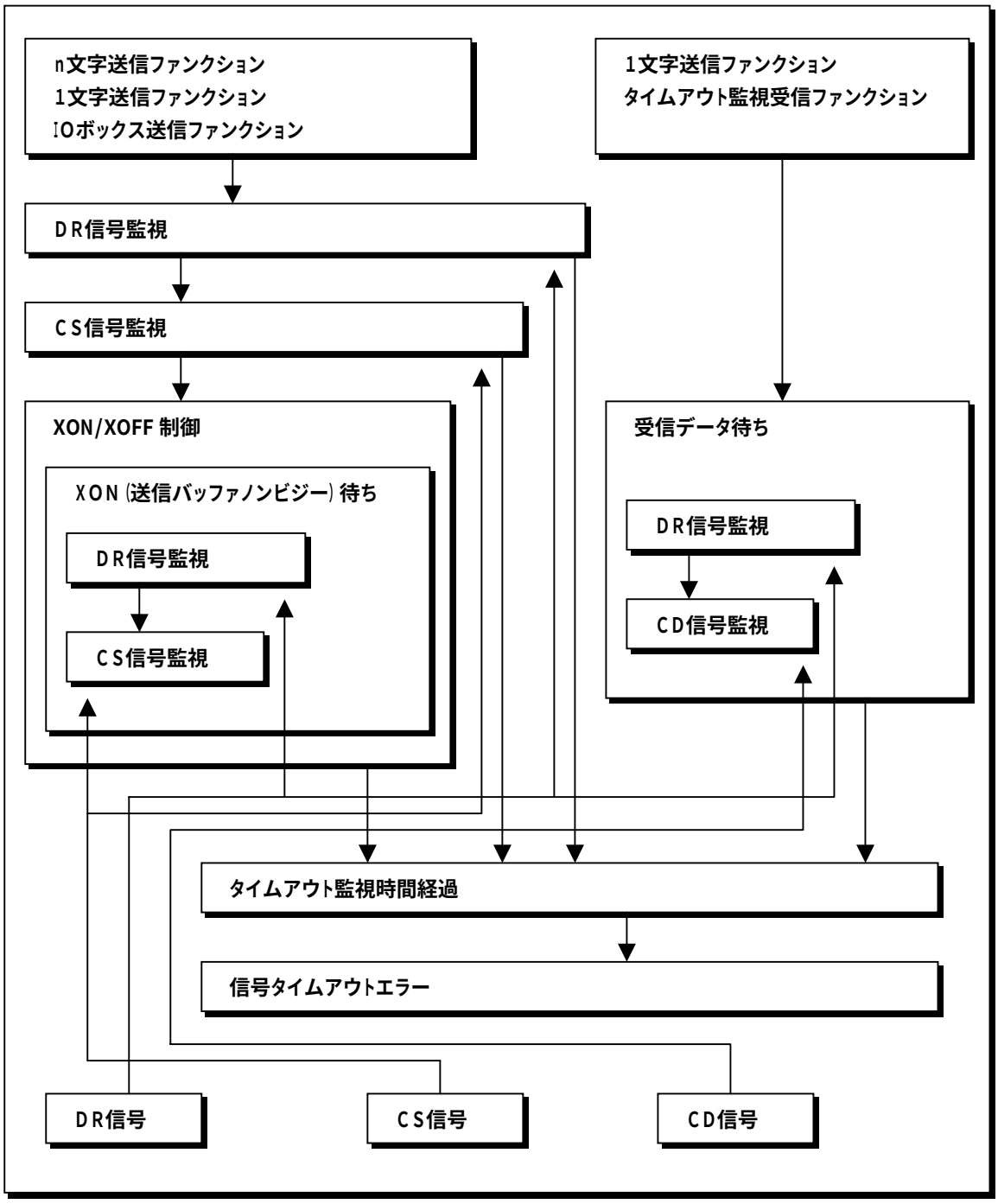
(2) タイムアウト監視

通信ポートのオープン、転送データの送信、受信データの読込みおよび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】



(3) ファンクションコールの信号線制御

以下に信号線を操作および参照するファンクションを示します。

①COMのオープン

信号線	制御
ER	ONまたはOFFに設定
RS	ONまたはOFFに設定
DR	ER信号ON後にON待ち
CS	RS信号ON後にON待ち
CD	ER、RS信号ON後にON待ち
	RS信号のみON後にOFF待ち
CTRL	送信ディセーブルに設定

②COMのクローズ

信号線	制御
ER	OFFに設定
RS	OFFに設定
CTRL	送信ディセーブルに設定

③COMのステータスリード

信号線	制御
DR	ON／OFF状態を参照
CS	ON／OFF状態を参照
CD	ON／OFF状態を参照

④n文字送信

信号線	制御
DR	転送データの送信前にON待ち
	XON/XOFF 制御でビジーのときにON待ち
CS	転送データの送信前にON待ち
	XON/XOFF 制御でビジーのときにON待ち

⑤1文字受信

信号線	制御
RS	バッファノンビジーになったときONに設定
DR	受信バッファに受信データがないときON待ち
CD	受信バッファに受信データがないときON待ち

⑥タイムアウト監視受信

信号線	制御
RS	バッファノンビジーになったときONに設定
DR	受信バッファに受信データがないときON待ち
CD	受信バッファに受信データがないときON待ち

⑦1文字送信

信号線	制御
DR	転送データの送信前にON待ち
	XON/XOFF 制御でビジーのときにON待ち
CS	転送データの送信前にON待ち
	XON/XOFF 制御でビジーのときにON待ち

⑧IOボックス送信設定

信号線	制御
CTRL	送信イネーブルまたは、送信ディセーブルに設定

⑨IOボックス送信

信号線	制御
DR	転送データの送信前にON待ち
CS	転送データの送信前にON待ち
CTRL	転送データの送信前に送信イネーブル、 送信後に送信ディセーブルに設定

⑩ER 信号の ON／OFF

信号線	制御
ER	ONまたはOFFに設定
DR	ER信号ON後にON待ち

⑪RS 信号の ON／OFF

信号線	制御
RS	ONまたはOFFに設定
CS	RS信号ON後にON待ち
CD	RS信号ON前にOFF待ち

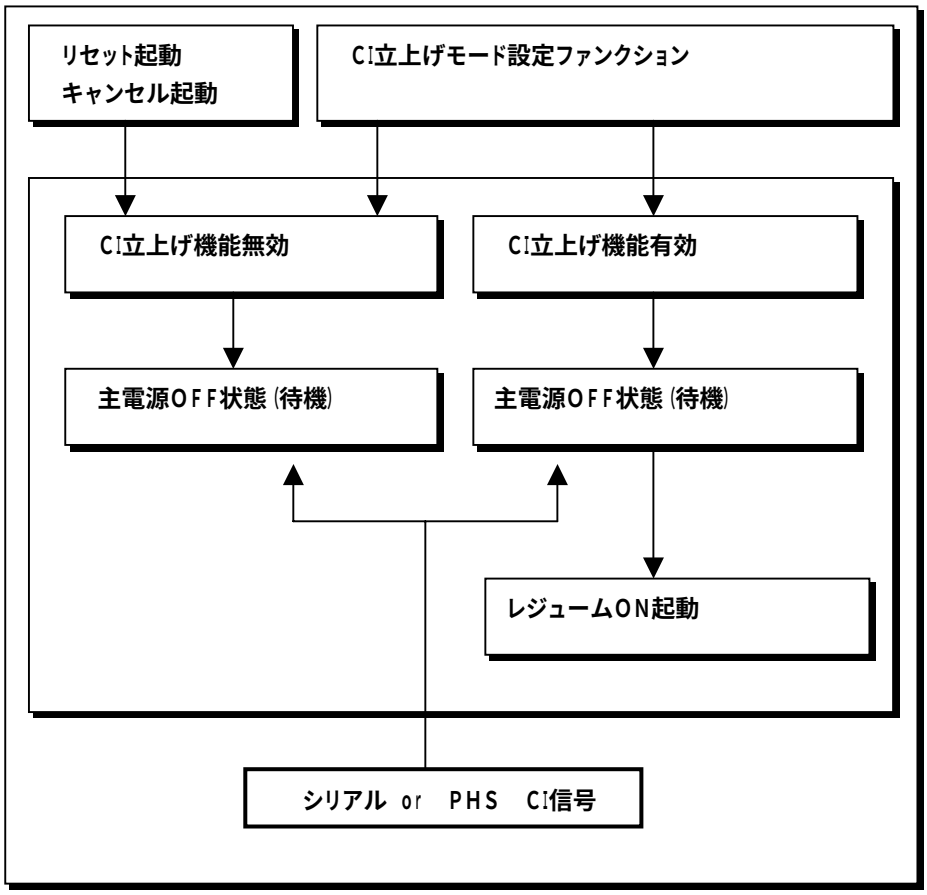
⑫ER／RS 信号の ON／OFF

信号線	制御
RS	ONまたはOFFに設定
ER	ONまたはOFFに設定
CS	RS信号ON後にON待ち
CD	RS信号ON前にON待ち
DR	ER信号ON後にON待ち

6.2.8.　CI信号立上げ

シリアルインタフェース、ＰＨＳポートへのCI信号着信により本機をレジュームＯＮ起動することができます。CI信号立上げを行う条件として、通信ポートのオープン／クローズ状態を指定することができます。
CI信号立上げの設定は「CI立上げモード設定」ファンクションで行います。
尚、本機をリセットおよびキャンセル（レジュームＯＦＦ）起動したときCI信号立上げ機能は解除（行いません）します。

【CI信号立上げの流れ】



6.2.9.　ＬＢ検出

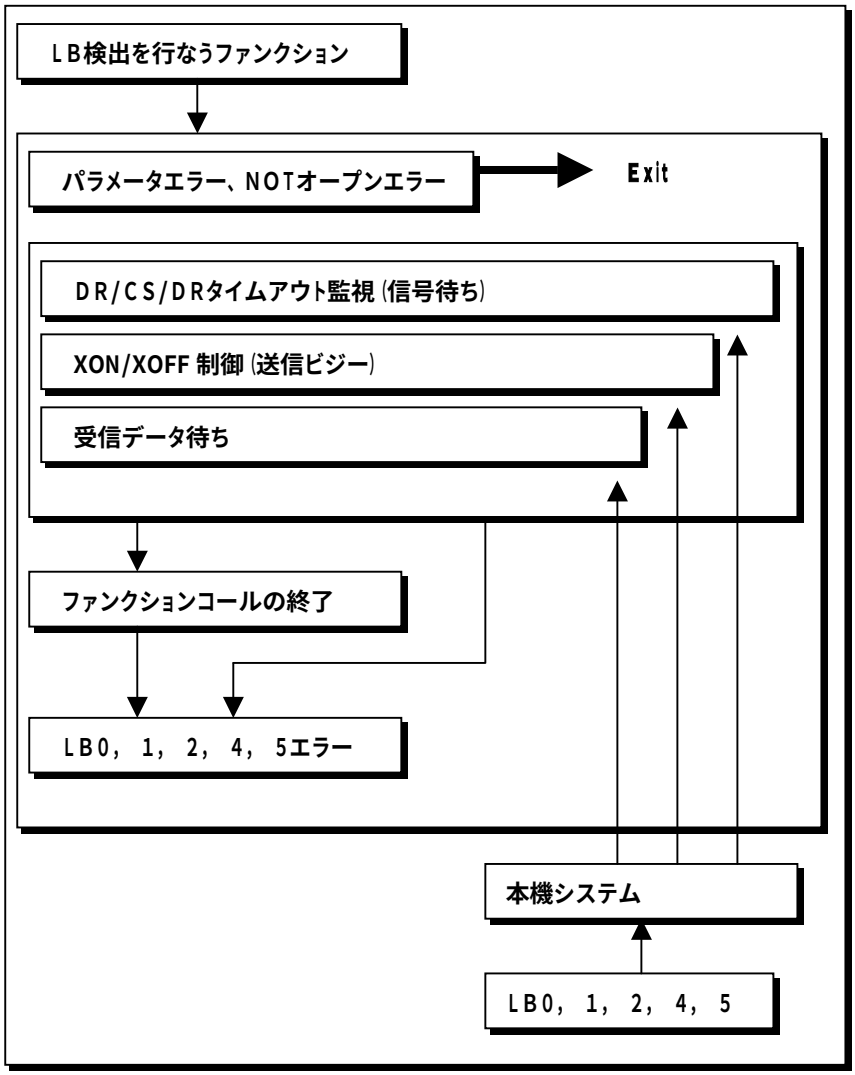
各ファンクションコールでは指定によりローバッテリの検出を行います。
ＬＢを検出するとファンクションは実行中の処理を中断し、異常終了します。
ＬＢの検出を行うには本機のシステムに対してＬＢ検出を行うように設定する必要があります。

(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】



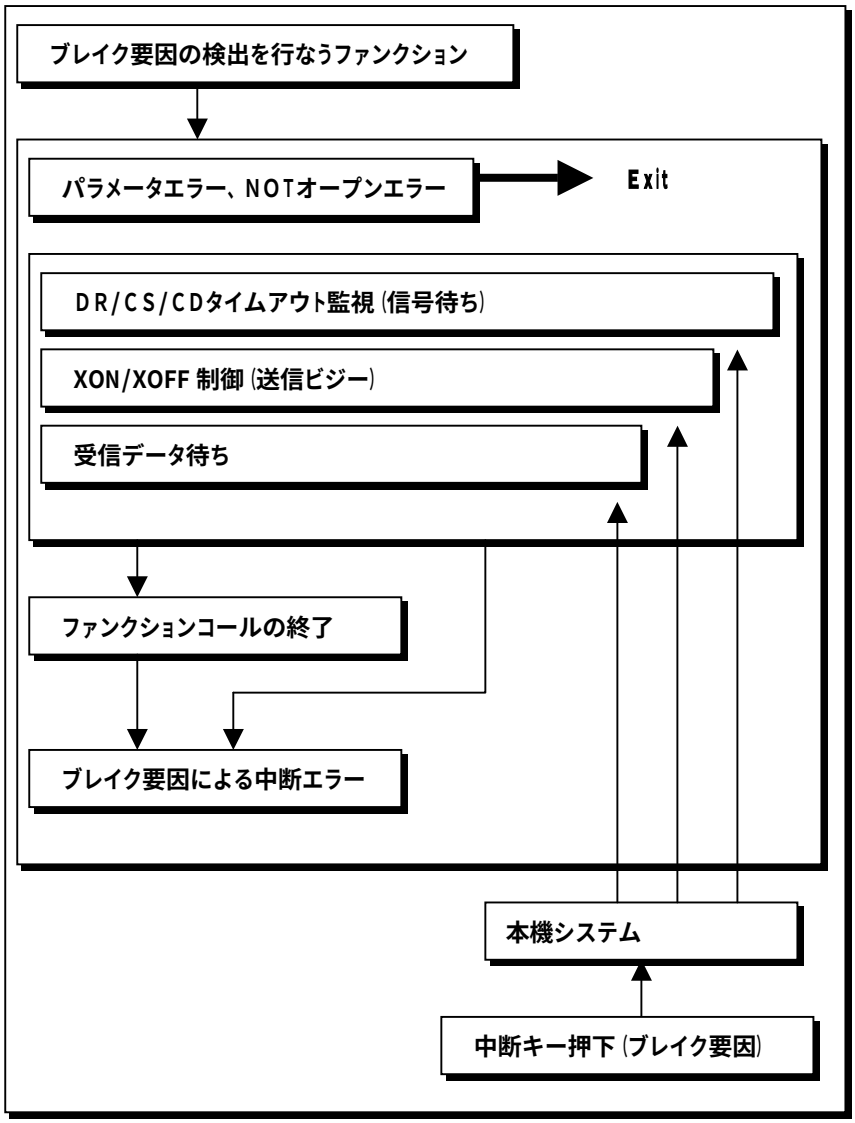
6.2.10. ブレイク要因検出

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】



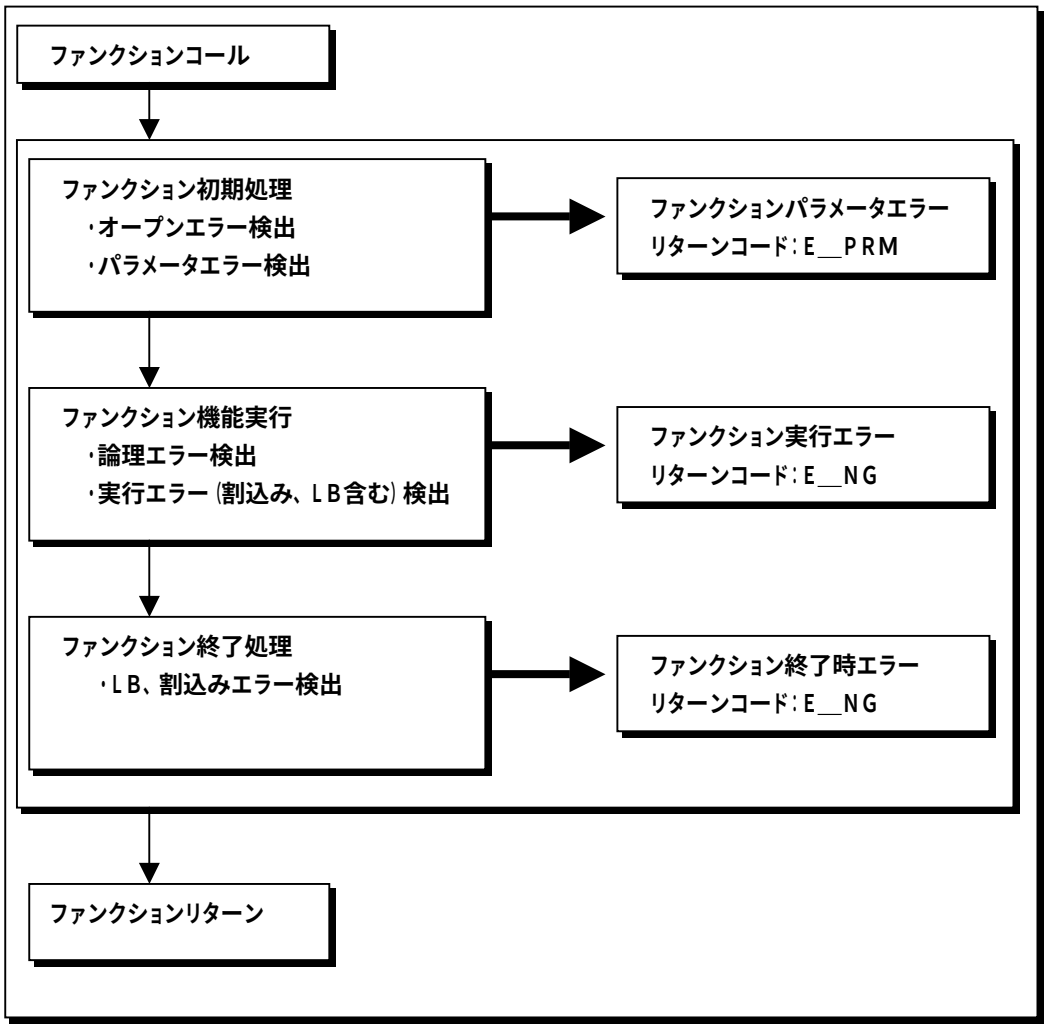
6.3. エラー詳細

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__BUFFUL	バッファフルエラー ・バッファフルエラー検出
	CERR__o__LBx	ローバッテリーエラー参照 ・LBx検出 (x＝0, 1, 2, 4, 5)

(2) ローバッテリー (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 ・「COMの占有」ファンクションで通信ポートは占有中 ・通信ポートはオープン中 ・IrDAポートが使用中 カシオIRポートとIrDAポートはシステムリソースを共用しているため、排他制御を行っている
	CERR__f__DRTIMEOUT	DR信号タイムアウト
	CERR__f__CSTIMEOUT	CS信号タイムアウト
	CERR__f__CDTIMEOUT	CD信号タイムアウト
	CERR__o__BREAK	ブレイク要因による中断 ・信号タイムアウト監視中にブレイク要因検出
	CERR__o__LBx (x＝0、1、2、4、5)	ローバッテリーエラー参照 ・信号タイムアウト監視中にLBx検出
E__PRM	なし	パラメータエラー ・通信ポートの指定が不当
	CERR__f__PARAMETER	パラメータエラー ・受信バッファレングスが範囲外 ・デリートコード数が範囲外 ・パリティビットの指定が不当 ・ストップビットの指定が不当 ・キャラクタレングスの指定が不当 ・ボーレイトの指定が不当 ・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 (P.151)
E__PRM	なし	パラメータエラー ・通信ポートの指定が不当

(6) COMの占有

エラーコード	エラーステータス	要因
E__NG	CERR__f__DEMESNE	占有エラー ・通信ポートは既に占有されている ・通信ポートはオープン中 ・IrDAポートが使用中 カシオIRポートとIrDAポートはシステムリソースを共用しているため、排他制御を行っている
	ファンクション終了時エラー	ファンクション終了時エラー参照 (P.151)
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 リターンコード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 (P.151)
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 (P.151)
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	ブレーク要因による中断 ・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 (P.151) ・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 (P.151)
E__PRM	CERR__f__PARAMETER	パラメータエラー ・受信タイムアウト監視値が範囲外
	なし	パラメータエラー ・通信ポートの指定が不当

(11) 1文字送信

エラーコード	エラーステータス	要因
E__NG	CERR__f__NORECOVER	致命的エラー ・「送受信の有効／無効」ファンクションで送信無効に設定されている ・カシオIRポート使用時に送信有効でない状態 →「COMのオープン」、「IOボックス送信」ファンクションの実行後 ・「ブレイク送出のON／OFF」 ファンクションでブレイクon中
	CERR__f__DRTIMEOUT	DR信号タイムアウト
	CERR__f__CSTIMEOUT	CS信号タイムアウト
	CERR__o__BREAK	ブレイク要因による中断 ・信号タイムアウト監視中にブレイク要因検出
	CERR__f__NOTOPEN	NOT OPENエラー ・通信ポートはオープンされていない
	CERR__o__LBx (x＝0、1、2、4、5)	ローバッテリーエラー参照 (P.151) ・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 (P.151)
E__PRM	なし	パラメータエラー ・通信ポートの指定が不当

(12) ブレーク送出の ON／OFF

エラーコード	エラーステータス	要因
E__NG	CERR__f__NOTOPEN	NOT OPENエラー ・通信ポートはオープンされていない
	ファンクション終了時エラー	ファンクション終了時エラー参照 (P.151)
E__PRM	なし	パラメータエラー ・通信ポートの指定が不当
	CERR__f__PARAMETER	パラメータエラー ・ブレークON／OFFの指定が不当

(13) 送受信の有効／無効

エラーコード	エラーステータス	要因
E__NG	CERR__f__NOTOPEN	NOT OPENエラー ・通信ポートはオープンされていない
	ファンクション終了時エラー	ファンクション終了時エラー参照 (P.151)
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 (P.151)
E__PRM	なし	パラメータエラー ・通信ポートの指定が不当
	CERR__f__PARAMETER	パラメータエラー ・送信状態設定の指定が不当

(15) IOボックス送信

エラーコード	エラーステータス	要因
E__NG	CERR__f__NORECOVER	致命的エラー ・「ブレイク送出のON／OFF」 ファンクションでブレイク ON 中
	CERR__f__DRTIMEOUT	DR信号タイムアウト
	CERR__f__CSTIMEOUT	CS信号タイムアウト
	CERR__o__BREAK	ブレイク要因による中断 ・各信号タイムアウト監視中にブレイク要因検出
	CERR__f__NOTOPEN	NOT OPENエラー ・通信ポートはオープンされていない
	CERR__o__LBx (x＝0、1、2、4、5)	ローバッテリーエラー参照 (P.151) ・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 (P.151)
E__PRM	なし	パラメータエラー ・通信ポートの指定が不当
	CERR__f__PARAMETER	パラメータエラー ・送信レングスが範囲外

(16) 受信バッファのクリア

エラーコード	エラーステータス	要因
E__NG	CERR__f__NOTOPEN	NOT OPENエラー ・通信ポートはオープンされていない
	ファンクション終了時エラー	ファンクション終了時エラー参照 (P.151)
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 (P.151)
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 (P.151)
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) CI立上げモード設定

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 ・通信ポートの指定が不当
	CERR__f__PARAMETER	パラメータエラー ・立上げモードの指定が不当

(21) 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 (P.151)
	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

(22) 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 (P.151) ・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 (P.151)
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) 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 (P.151) ・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 (P.151)
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) 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 (P.151) ・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 (P.151)
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

(25) ブレイク要因の設定

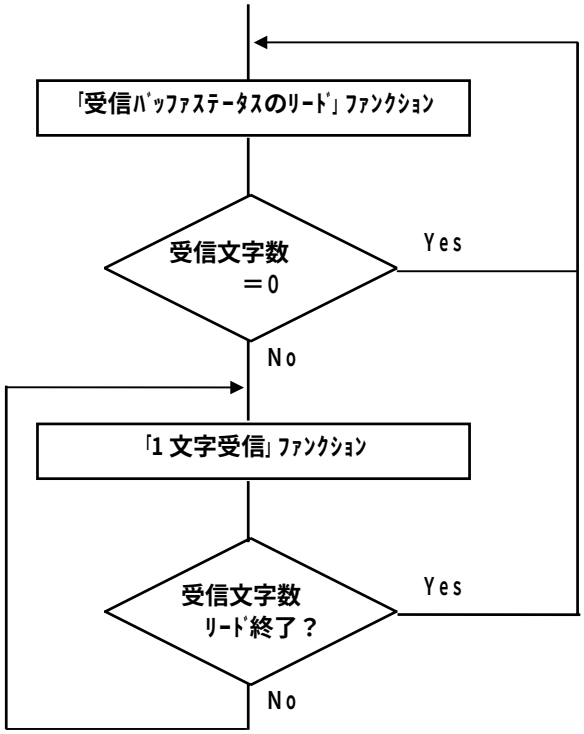
エラーコード	エラーステータス	要因
E__PRM	なし	パラメータエラー ・ブレイク要因通知の指定が不当 ・ファンクションキーの指定が不当

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込む例を示します。



①受信文字数を取得
　エラーを随時チェックする

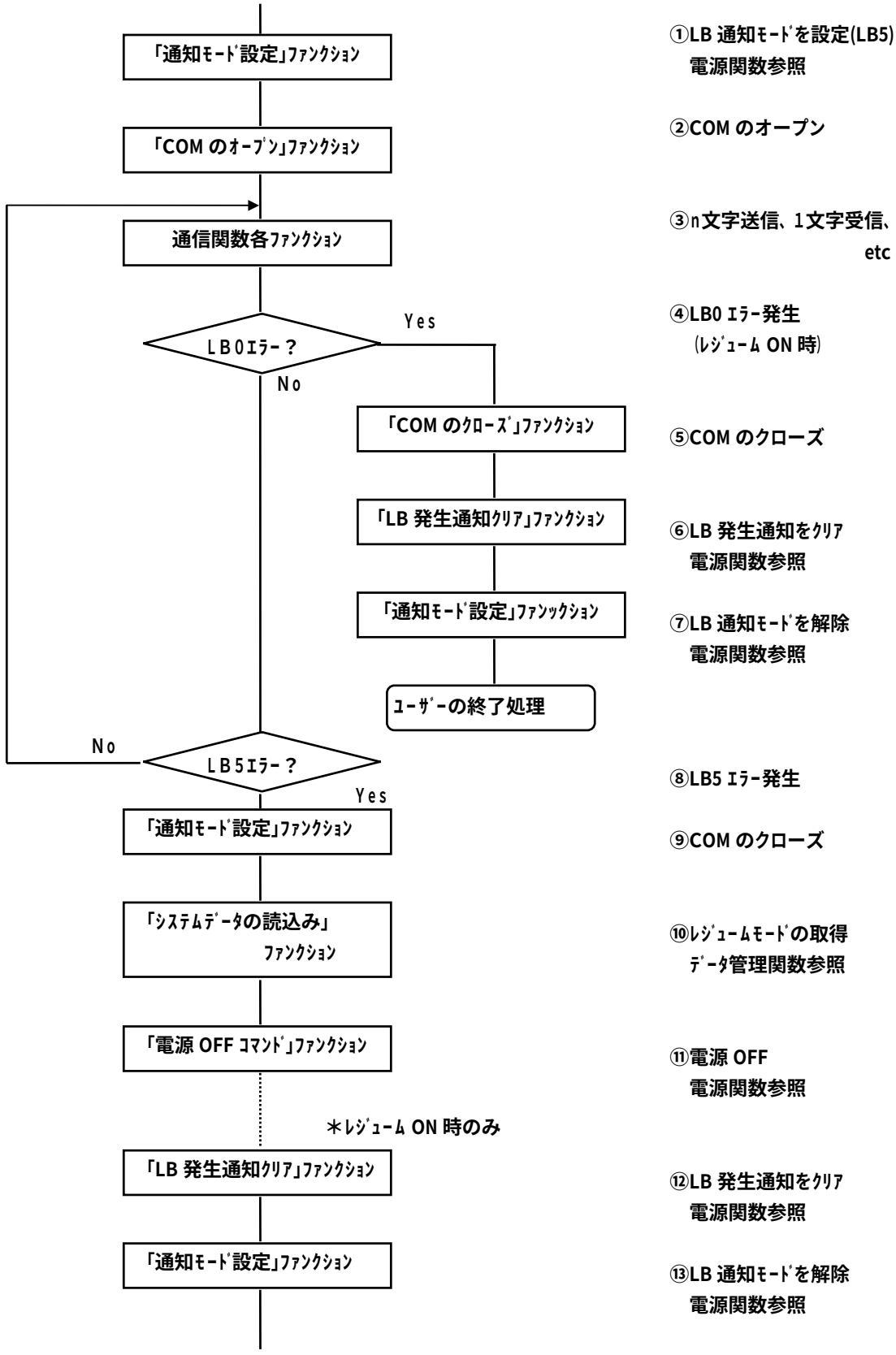
②受信文字が無ければ繰り返し

③受信ﾊﾞｯﾌｧから受信ﾃﾞｰﾀをﾘｰﾄﾞ

④受信文字を全てﾘｰﾄﾞする
　エラーを随時チェックする

6.4.2.　ＬＢエラーチェック

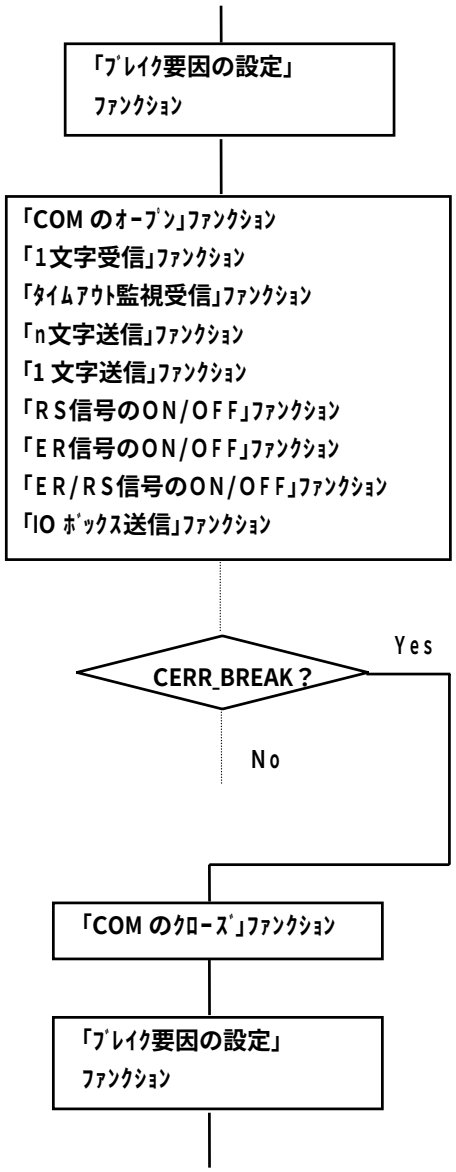
ローバーッテリ（ＬＢ）エラーのチェックを通信中の電源ＯＦＦキー押下（ＬＢ５）および主電池なし（ＬＢ０）発生時の処理例を以下に示します。レジュームＯＮによる通信の再開等ができない場合などを考慮して行います。



＊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、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いけません。

a) CTRL信号の切替え時のターンアラウンドタイム

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

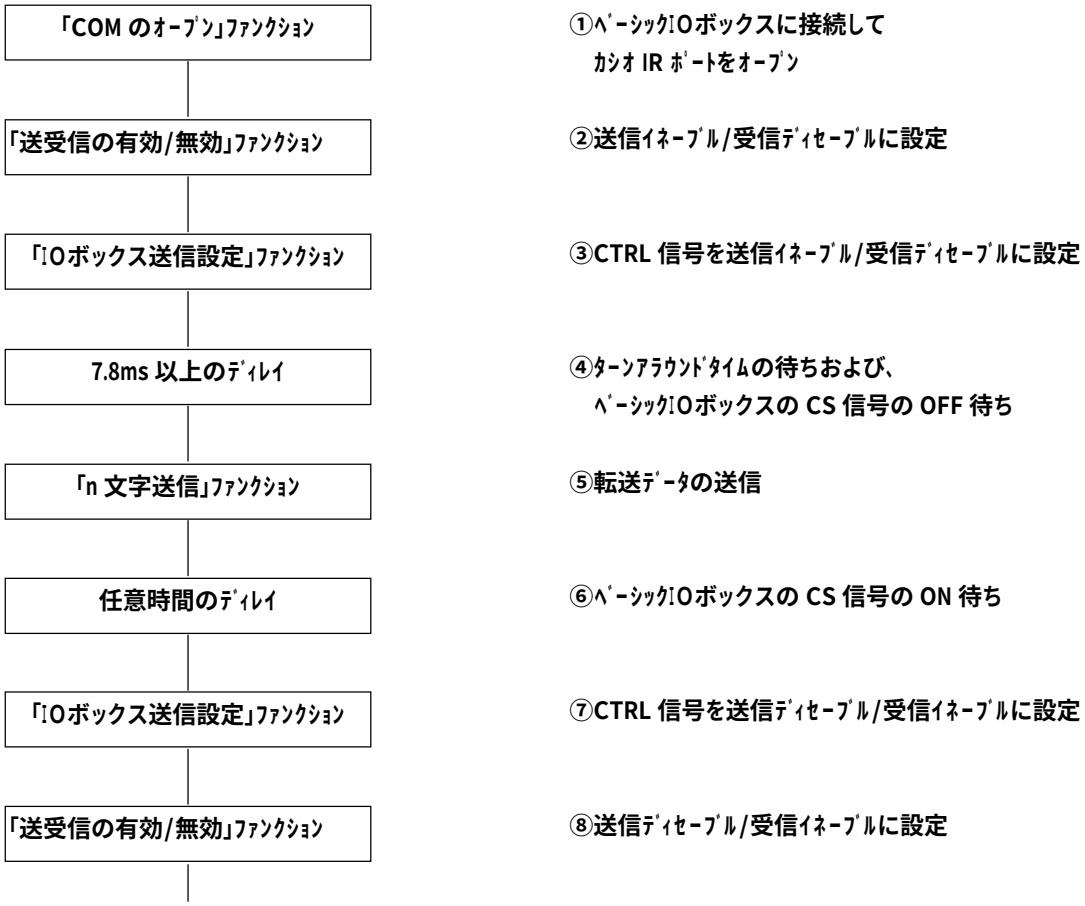
b) CTRL信号とベーシックIOボックスのCS信号 (RS-232Cインタフェース側) の状態

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。

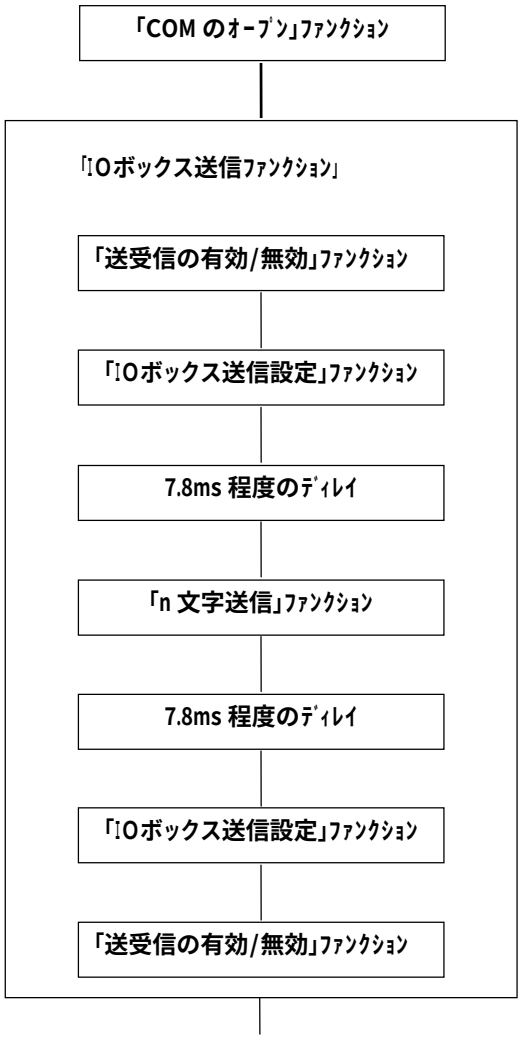
CTRL信号が送信ディセーブル、ベーシックIOボックスのCS信号がOFFであるとき本機にフレーミングエラー (ノイズが発生する) が発生する場合があります。

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ﾍﾞｰｼｯｸIOボックスに接続して
カシオ IR ﾎﾟｰﾄをｵｰﾌﾟﾝ

②送信ｲﾈｰﾌﾞﾙ/受信ﾃﾞｨｾｰﾌﾞﾙに設定

③CTRL 信号を送信ｲﾈｰﾌﾞﾙ/受信ﾃﾞｨｾｰﾌﾞﾙに設定

④ﾀｰﾝｱﾗｳﾝﾄﾞﾀｲﾑの待ちおよび、
ﾍﾞｰｼｯｸIOボックスの CS 信号の OFF 待ち

⑤転送ﾃﾞｰﾀの送信

⑥ﾍﾞｰｼｯｸIOボックスの CS 信号の ON 待ち

⑦CTRL 信号を送信ﾃﾞｨｾｰﾌﾞﾙ/受信ｲﾈｰﾌﾞﾙに設定

⑧送信ﾃﾞｨｾｰﾌﾞﾙ/受信ｲﾈｰﾌﾞﾙに設定

(2) CERR__f__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	DATA	FCS	EOF

→

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

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

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

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

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

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

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

次ページへ続く

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

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

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

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

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

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

⑯	OFF	OFF				送受信の有効/無効

＊　送信ｲﾈｰﾌﾞﾙ/受信ｲﾈｰﾌﾞﾙに設定します。

⑰	OFF	OFF				受信ﾊﾞｯﾌｧのｸﾘｱ

＊　受信バッファとｴﾗｰｽﾃｰﾀｽをｸﾘｱします。

⑱	ON	OFF				IBOX 送信設定

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

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	･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	･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でしないで下さい ･通信関数内部受信バッファ使用する場合は使用でしないで下さい
3	RS／CSフロー制御	･カシオIRポートを使用する場合は使用しないで下さい ･通信関数内部受信バッファ使用する場合は使用でしないで下さい
4	SI／SO制御	･XON/XOFF 制御、デリートコード制御、エラーコードバッファリング制御で使用する制御コードとSI／SOコードが重複しないように設定して下さい 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	･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、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先頭の文字がエラーコードバッファリング制御の対象となります
6	外部要因エラーの検出	･LB0，1，2，4，5およびブレイク要因は、システムがセットするイベントフラグの参照により、検出します。 従って検出を行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	CI信号立上げ	･CI信号はONからOFFに変化したときに検出します
8	CERR__r__xxxx2 エラーステータス	･カシオIR、シリアルポートが使用するシリアルコントローラのレシーバはFIFOバッファになっています。通信エラーが発生したときにレシーバ内のデータのいずれかがエラーであることを示しています。エラーステータスはレシーバから先頭の文字を読出したときに設定しています。これにより「1文字受信」、「タイムアウト監視受信」ファンクションでエラーとなったきに読出した文字と実際にエラーとなった文字に最大プラス15文字（レシーバが16バイト）の誤差があります
9	転送速度	･複数ポートオープン時は転送速度の合計を115．2kbps以下に設定して下さい
10	ブレイク信号	･カシオIRポートではブレイク信号の送出および検出をするこができません ブレイク信号を送信した場合は1バイト長のデータのスペース（ストップビットまで）となります（送出停止までの間、連続して送出されます） ブレイク信号を受信した場合はフレーミングエラーとなります
11	受信バッファのクリア	･フロー制御中に受信バッファビジーであるとき「受信バッファのクリア」ファンクションで受信バッファのクリアを行った場合、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。

機能	COMオープン	関数名	c_open

カシオIR、シリアル、PHSポートの各通信ポートをオープンします。
カシオIRポートを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）に、それ以外の通信ポートは全二重（送受信イネーブル）の状態でオープンします。
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
- SI／SO制御の設定
- DR／CS/CD信号タイムアウト監視
- 送受信の有効
- フロー制御の設定
- デリートコード設定
- 受信割込みの許可
- 受信バッファの設定
- 通信ポートの排他制御
- 通信形式の設定
- ER／RS信号設定

≪C言語インタフェース≫

[コーリングシーケンス]

```
ER   ercd = 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]

次ページに記載

[リターンパラメータ]

ER　ercd	:リターンコード

[リターンコード]

E_OK	:正常終了
E_NG	:異常終了
E_PRM	:パラメータエラー

≪備考≫

- 「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のタイマ監視は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。

【パラメータ】

H　com_no　　：通信ポート
　　COM0　　：カシオIRインタフェース
　　COM1　　：シリアルインタフェース
　　COM2　　：予約領域（指定するとパラメータエラーになります）
　　COM3　　：ＰＨＳインタフェース

UW　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．

B　busy_ch　　：ＸＯＦＦコード（受信不可時のコード）
B　nonbusy_ch　：ＸＯＮコード　（受信可能時のコード）
B　*buff　　：受信バッファアドレス
H　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_TBL *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_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;
```

機能	COMクローズ	関数名	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

≪C言語インタフェース≫

[コーリングシーケンス]
ER　ercd = c_close(H com_no);

[パラメータ]
H　com_no　：通信ポート
COM0　：カシオ IR インタフェース
COM1　：シリアルインタフェース
COM2　：予約
COM3　：ＰＨＳインタフェース

[リターンパラメータ]
ER　ercd　：リターンコード

[リターンコード]
E_OK　：正常終了
E_NG　：異常終了
E_PRM　：パラメータエラー

≪備考≫

機能	COMステータスのリード	関数名	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

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

```
H    com_no      :通信ポート
                    COM0      :カシオ IR インタフェース
                    COM1      :シリアルインタフェース
                    COM2      :予約
                    COM3      :ＰＨＳインタフェース
```

[リターンパラメータ]

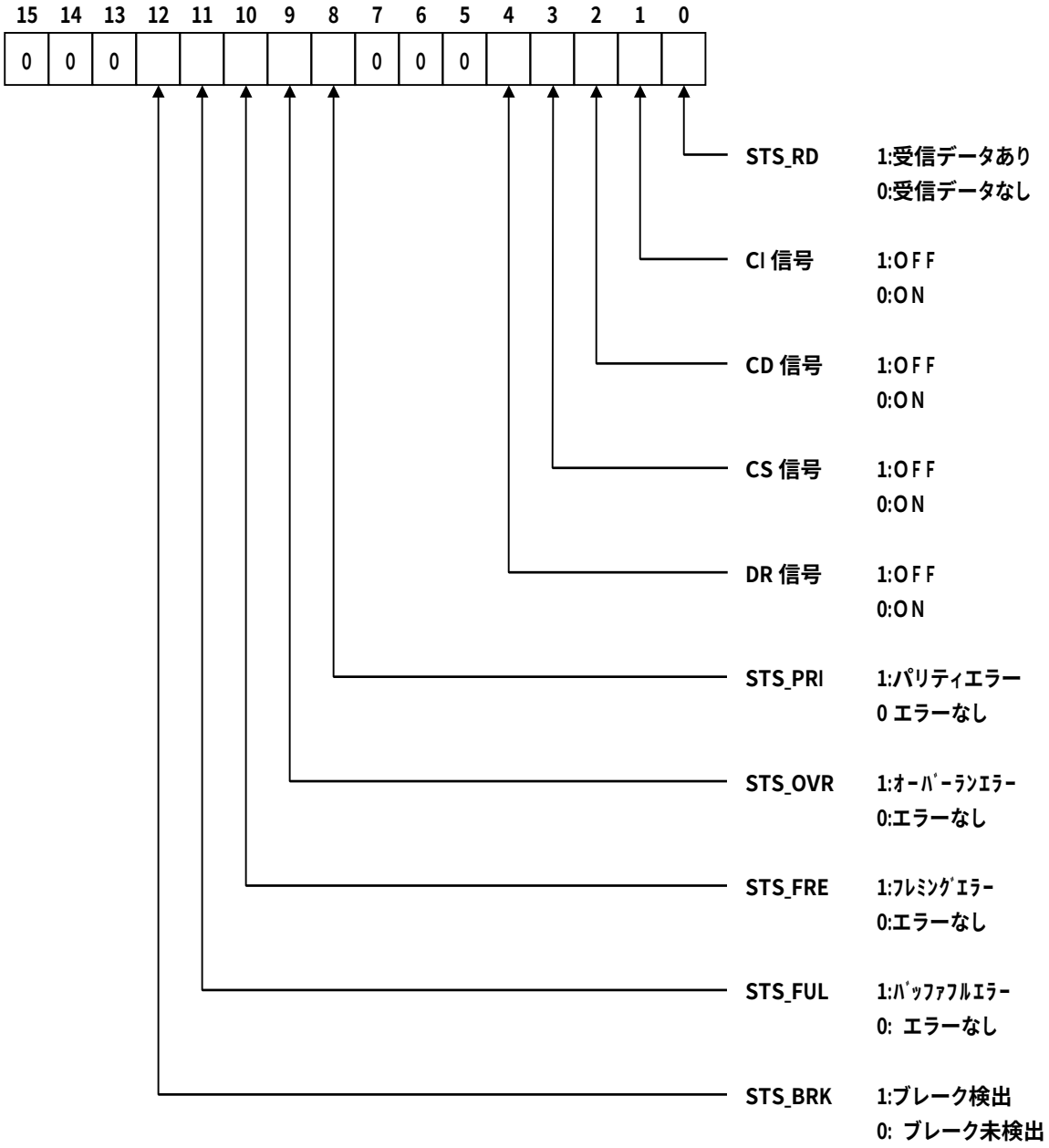
```
ER    ercd        :リターンコード
```

[リターンコード]

```
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）
```

≪備考≫

COMステータス


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

機能	COMの占有	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER　ercd = c_hold(H com_no,　B mode);

[パラメータ]
　　H　com_no　　:通信ポート
　　　　COM0　　:カシオ IR インタフェース
　　　　COM1　　:シリアルインタフェース
　　　　COM2　　:予約
　　　　COM3　　:ＰＨＳインタフェース
　　B　mode　　:占有設定
　　　　HOLD_ON　　:占有する
　　　　HOLD_OFF　　:占有解除

[リターンパラメータ]
　　ER　ercd　　:リターンコード

[リターンコード]
　　E_OK　　:正常終了
　　E_NG　　:異常終了
　　E_PRM　　:パラメータエラー

≪備考≫

機能	COMのオープンチェック	関数名	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ポートの使用中はカシオIR、シリアルポートは占有状態になります。）

≪C言語インタフェース≫

[コーリングシーケンス]

```
ER   ercd = c_chkopen( void );
```

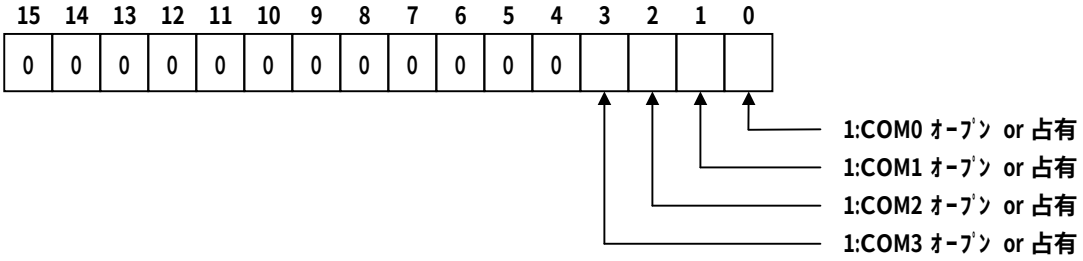
[パラメータ]

なし

[リターンパラメータ]

ER　ercd　　:リターンコード(ｵｰﾌﾟﾝ or 占有通知)

[リターンコード]



≪備考≫

機能	n文字送信	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER　ercd = c_dout(H com_no,　B *buffer,　H length);

[パラメータ]
　　H　com_no　　：通信ポート
　　　　COM0　：カシオ IR インタフェース
　　　　COM1　：シリアルインタフェース
　　　　COM2　：予約
　　　　COM3　：ＰＨＳインタフェース
　　B　*buffer　　：送信バッファアドレス
　　H　length　　：送信文字数（バイト数）

[リターンパラメータ]
　　ER　ercd　　：リターンコード

[リターンコード]
　　E_OK　　：正常終了
　　E_NG　　：異常終了
　　E_PRM　　：パラメータエラー

≪備考≫
・「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。

機能	1文字受信	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER　ercd = c_din(H com_no,　B *buffer);

[パラメータ]
　　H　com_no　　:通信ポート
　　　　COM0　　:カシオ IR インタフェース
　　　　COM1　　:シリアルインタフェース
　　　　COM2　　:予約
　　　　COM3　　:ＰＨＳインタフェース
　　B　*buffer　　:格納バッファアドレス

[リターンパラメータ]
　　ER　ercd　　:リターンコード

[リターンコード]
　　E_OK　　:正常終了
　　E_NG　　:異常終了
　　E_PRM　　:パラメータエラー

≪備考≫
・「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。

機能	タイムアウト監視受信	関数名	c_tmdin

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

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

```
H   com_no      :通信ポート
                   COM0     :カシオ IR インタフェース
                   COM1     :シリアルインタフェース
                   COM2     :予約
                   COM3     :ＰＨＳインタフェース
B   *buffer     :格納バッファアドレス
H   rcv_time    :受信タイムアウト監視値  0～32767 (×7．8ms)
```

[リターンパラメータ]

```
ER   ercd       :リターンコード
```

[リターンコード]

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

≪備考≫

・「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。

機能	1文字送信	関数名	c_out

データを1文字（バイト）送信します。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

```
H   com_no      :通信ポート
                   COM0     :カシオ IR インタフェース
                   COM1     :シリアルインタフェース
                   COM2     :予約
                   COM3     :ＰＨＳインタフェース
B   snddata     :送信文字 (1 ﾊﾞｲﾄ)
```

[リターンパラメータ]

```
ER   ercd       :リターンコード
```

[リターンコード]

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

≪備考≫

・「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。

機能	ブレーク信号の制御	関数名	c_break

ブレーク信号の送出または、送出停止を行います。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

```
H   com_no      :通信ポート
                   COM0     :カシオ IR インタフェース
                   COM1     :シリアルインタフェース
                   COM2     :予約
                   COM3     :ＰＨＳインタフェース
B   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
```

[リターンパラメータ]

```
ER   ercd       :リターンコード
```

[リターンコード]

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

≪備考≫

機能	送受信の有効／無効	関数名	c_txrx

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

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

H　com_no　：通信ポート
- COM0　：カシオIRインタフェース
- COM1　：シリアルインタフェース
- COM2　：予約
- COM3　：ＰＨＳインタフェース

B　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

[リターンパラメータ]

ER ercd　：リターンコード

[リターンコード]

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

≪備考≫

機能	IOボックス送信設定	関数名	c_iobox

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

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

```
H  com_no     : 通信ポート
                  COM0          :カシオIRインタフェース
B  mode       : 送信状態設定
                  C_IOBOXENB    :送信に設定(送信ｲﾈｰﾌﾞﾙ/受信ﾃﾞｨｾｰﾌﾞﾙ)
                  C_IOBOXDSB    :送信を解除(送信ﾃﾞｨｾｰﾌﾞﾙ/受信ｲﾈｰﾌﾞﾙ)
```

[リターンパラメータ]

```
ER ercd       :リターンコード
```

[リターンコード]

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

≪備考≫

機能	IOボックス送信	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

```
H  com_no     :通信ポート
                  COM0          :カシオ IR インタフェース
B  *buffer    :送信バッファアドレス
H  length     :送信文字数（バイト数）
```

[リターンパラメータ]

```
ER ercd       :リターンコード
```

[リターンコード]

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

≪備考≫

・「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。

機能	受信バッファのクリア	関数名	c_flush

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

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

```
H   com_no      :通信ポート
                    COM0        :カシオ IR インタフェース
                    COM1        :シリアルインタフェース
                    COM2        :予約
                    COM3        :ＰＨＳインタフェース
```

[リターンパラメータ]

```
ER   ercd       :リターンコード
```

[リターンコード]

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

≪備考≫

機能	受信バッファステータスのリード	関数名	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出し可能なデータ数 (受信文字数：バイト単位)
・受信バッファの先頭に格納されているデータの文字コード (次読出し文字)
・受信バッファに格納できるデータ数 (受信可能文字数：バイト単位)

≪C言語インタフェース≫

[コーリングシーケンス]
```
ER ercd = c_bfsts( H com_no,  COM_STS *bfsts );
```

[パラメータ]
```
H  com_no          :通信ポート
                       COM0    :カシオ IR インタフェース
                       COM1    :シリアルインタフェース
                       COM2    :予約
                       COM3    :ＰＨＳインタフェース
COM_STS *bfsts     :受信バッファステータス
```

[リターンパラメータ]
```
ER ercd        :リターンコード
```

[ストラクト構造]
```
typedef  struct {
                    H  char_no          :受信文字数
                    H  rest_no          :受信可能残り文字数
                    UB char_cod         :先頭文字コード
                  } COM_STS ;
```

[リターンコード]
```
E_OK           :正常終了
E_NG           :異常終了
E_PRM          :パラメータエラー
```

≪備考≫

機能	エラーコードバッファリング制御の設定	関数名	c_errbfring

リターンコード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。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

```
H   com_no      :通信ポート
                    COM0            :カシオ IR インタフェース
                    COM1            :シリアルインタフェース
                    COM2            :予約
                    COM3            :ＰＨＳインタフェース
B   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
UB c_errcd      :エラーコード（任意の１バイトコード）
```

[リターンパラメータ]

```
ER ercd         :リターンコード
```

[リターンコード]

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

≪備考≫

機能	エラーステータスのリード	関数名	c_rderrsts

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

≪C 言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

```
H  com_no          :通信ポート
                      COM0    :カシオ IR インタフェース
                      COM1    :シリアルインタフェース
                      COM2    :予約
                      COM3    :ＰＨＳインタフェース
UW *com_status     :エラーステータス
```

[リターンパラメータ]

```
ER ercd            :リターンコード
```

[リターンコード]

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

≪備考≫
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。

機能	受信ハンドラ切替え	関数名	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

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

```
H   com_no      :通信ポート
                    COM0            :カシオ IR インタフェース
                    COM1            :シリアルインタフェース
                    COM2            :予約
                    COM3            :PHSインタフェース
B   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
```

[リターンパラメータ]

```
ER ercd         :リターンコード
```

[リターンコード]

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

≪備考≫

機能	CI信号立ち上げモード設定	関数名	c_cimode

CI信号による立ち上げモード設定を行います。
本ファンクションで立上げモードに指定にすると、指定した通信ポートのCI信号からの割込みにより本機を起動（レジューム ON立上げ）します。
本ファンクションは通信ポートのオープンまたはクローズ状態で使用することができます。
本機をリセットおよびキャンセル（レジュームOFF）で起動した場合はモード0となります。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER ercd = c_cimode(　H com_no,　UH cimode);

[パラメータ]
　　H　com_no　　:通信ポート
　　　　COM1　　:シリアルインタフェース
　　UH cimode　　:CI信号立ち上げモード設定
　　　　C_CI0　　:モード0
　　　　C_CI1　　:モード1
　　　　C_CI2　　:モード2

モード / 状態	0	1	2
COMオープン	×	○	○
COMクローズ	×	○	×

○:立ち上げ可能
×:立ち上げ不可能

[リターンパラメータ]
　　ER ercd　　:リターンコード

[リターンコード]
　　E_OK　　:正常終了
　　E_PRM　　:パラメータエラー

≪備考≫

機能	ＤＲ／ＣＳ／ＣＤタイムアウト監視値の設定	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

H　com_no	：通信ポート	
	COM0	：カシオ IR インタフェース
	COM1	：シリアルインタフェース
	COM2	：予約
	COM3	：ＰＨＳインタフェース
H　cs_time	：ＣＳタイムアウト監視値設定(0～32767)×7.8ms	
H　dr_time	：ＤＲタイムアウト監視値設定(0～32767)×7.8ms	
H　cd_time	：ＣＤタイムアウト監視値設定(0～32767)×7.8ms	

[リターンパラメータ]

ER ercd	：リターンコード

[リターンコード]

E_OK	：正常終了
E_NG	：異常終了
E_PRM	：パラメータエラー

≪備考≫

機能	ER 信号の制御	関数名	c_er

ＥＲ信号線の ON／OFF を行います。

≪C 言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

```
H   com_no      :通信ポート
                    COM0        :カシオ IR インタフェース
                    COM1        :シリアルインタフェース
                    COM2        :予約
                    COM3        :ＰＨＳインタフェース
B   er_mode     :ＥＲ信号線の設定
                    ERS_ON      :ＥＲ信号ＯＮ
                    ERS_OFF     :ＥＲ信号ＯＦＦ
```

[リターンパラメータ]

```
ER ercd         :リターンコード
```

[リターンコード]

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

≪備考≫

・「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。

機能	ＲＳ信号の制御	関数名	c_rs

ＲＳ信号線の ON／OFF を行います。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

```
H   com_no      :通信ポート
                    COM0        :カシオ IR インタフェース
                    COM1        :シリアルインタフェース
                    COM2        :予約
                    COM3        :ＰＨＳインタフェース
B   mode        :RS 信号線の設定
                    RS_ON       :ＲＳ信号ＯＮ
                    RS_OFF      :ＲＳ信号ＯＦＦ
```

[リターンパラメータ]

```
ER ercd         :リターンコード
```

[リターンコード]

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

≪備考≫

・「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。

機能	ＥＲ／ＲＳ信号の制御	関数名	c_errs

ＥＲ／ＲＳ信号線のＯＮ／ＯＦＦを行います。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

```
H   com_no      :通信ポート
                    COM0        :カシオ IR インタフェース
                    COM1        :シリアルインタフェース
                    COM2        :予約
                    COM3        :ＰＨＳインタフェース
B   mode        :ER/RS 信号線の設定
                    ERRS_ON     :ＥＲ／ＲＳ信号ＯＮ
                    ERRS_OFF    :ＥＲ／ＲＳ信号ＯＦＦ
```

[リターンパラメータ]

```
ER ercd         :リターンコード
```

[リターンコード]

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

≪備考≫

・「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。

機能	ブレイク要因の設定	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER ercd = c_brkevent(UH event_mode,　UB func_mode);

[パラメータ]
　　UH　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
　　UB　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

[リターンパラメータ]
　　ER ercd　　:リターンコード

[リターンコード]
　　E_OK　　:正常終了
　　E_PRM　　:パラメータエラー

≪備考≫
・「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。

機能	WakeUp信号の設定	関数名	c_wp

WakeUp信号をHIまたはLOWに設定します。
WakeUp信号はPHSポートに接続するPHSアダプタ専用の出力信号です。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER ercd = c_wp(H com_no, UB wp_mode);

[パラメータ]
　　H　com_no　　：通信ポート
　　　　COM3　　：PHSインタフェース
　　UB　er_mode　　：WAKEUP信号線の設定
　　　　WP_HI　　：WAKEUP信号HI
　　　　WP_LOW　　：WAKEUP信号LOW

[リターンパラメータ]
　　ER ercd　　：リターンコード

[リターンコード]
　　E_OK　　：正常終了
　　E_NG　　：異常終了（WAKEUP信号線の設定誤り）
　　E_PRM　　：パラメータエラー（通信ポート誤り）

≪備考≫
異常終了のときエラーステータスはパラメータエラーになります。
本ファンクションはファンクション終了処理を行います。

7. IrDA 部関数

7.1. 機能

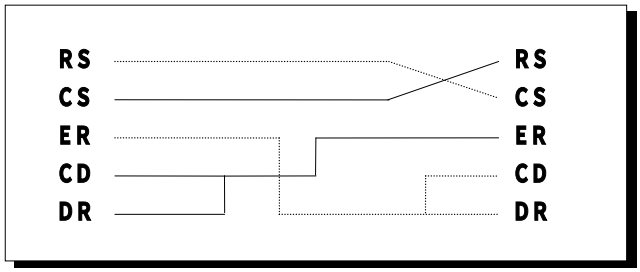
7.1.1. シリアルポートエミュレーション

IrDA部はシリアルポートエミュレーションエンティティとして存在します。
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ト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ないます。
待ち時間は秒単位に指定するかまたは、コネクトを行うまで待つかを指定します。
尚、本関数が異常終了した場合はIrCOMM (カシオ IR インタフェース) はクローズ状態となります。

(2) IrCOMMクローズ

IrCOMM (IrDAポート) をクローズします。
相手局とコネクトを切断するための手続き (通信) を行います。
この処理中に異常が発生することで異常終了となることがありますが、正常にコネクト切断できた場合と同様に赤外線デバイス電源OFF、通信用デバイスおよびリソースのリリースを行いクローズ状態となります。
尚、相手局からのコネクト切断を正常に受理した状態で本機能が使用された場合は正常終了となります。

(3) データ読込み

受信データの読込みを行います。
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込みが正常に行われていても異常終了となる場合があります。
データ待ち時間の指定は、通信状態設定関数 (Ir_State_Set) で行うことができます。
尚、、本関数では受信バッファに受信データが存在するとき、コネクト切断によるエラーは、受信バッファのデータが無くなるまで通知しません。
この場合、受信バッファの全てのデータ読込みが終了したとき、に異常終了となりますが、受信データはユーザ定義のエリアへ格納されています。
また、相手局からのコネクト切断を待つときは本機能を使用することで可能です。
ユーザアプリケーションが従局的な役割であるときは本機能で主局側からのコネクト切断を待ちを行い、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) データ書込み

送信データの書込みを行います。
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込みが行えなくなるか (バッファビジ-) 、ユーザー定義のバイトサイズまで書込みを行います。
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、通知ます。信号のＯＮまたはＯＦＦの指定を行います。
信号待ち時間が指定されているとき指定した信号状態でなければ信号待ちとなります。
信号待ちとなったときＬＢエラー、タイムアウト、ブレイクイベントのチェックを行い、エラー時は直ちに異常終了となります。
データ待ち時間の指定は、通信状態設定関数 (Ir_State_Set) で行うことができます。

(14) DR検査

ＤＲ信号のＯＮ／ＯＦＦ状態をチェックし、通知します。
信号待ち時間が指定されているとき信号のＯＮ待ちとなります。
信号待ちとなったのとき、ＬＢエラー、タイムアウト、ブレイクイベントのチェックを行い、エラー時は直ちに異常終了となります。
信号待ち時間の指定は、通信状態設定関数 (Ir_State_Set) で行うことができます。

(15) CS検査

CS信号のON／OFF状態をチェックし、通知します。
信号待ち時間が指定されているとき信号のON待ちとなります。
信号待ち時、LBエラー、タイムアウト、ブレイクイベントのチェックを行い、エラー時は直ちに異常終了となります。
信号待ち時間の指定は、通信状態設定関数(Ir_State_Set)で行うことができます。

(16) CI検査

CI信号のON／OFF状態をチェックし、通知します。

(17) BREAK検査

ブレイク信号の受信状態をチェックし、通知します。このとき受信状態をクリアします。
尚、ブレイク受信となった場合、フレーミングエラーは発生しません。

(18) エラー値取得

エラー値を取得します。
各関数の異常終了の詳細となるエラー値を返します。このときエラー値をクリアします。

(19) 通信状態設定

IrDA部の通信状態を設定します。本機能はIrCOMMオープンに先立って行う必要があります。
局は自局が1次局か2次局であるかを指定します。

- 1次局
  データリンクを2次局に指示します。自局が1次局であるときコネクトする相手局は2次局となります。
- 2次局
  1次局からデータリンクの指示を受けます。2次局は1次局からのデータリンクの指示を受けることでコネクトします。
  自局が2次局であるときコネクトする相手局は1次局となります。

Wireはエミュレートする結線タイプを指定します。結線タイプをWireと呼び、Wireの指定により機能が異なります。
指定する各Wireの機能は以下の通りです。

- 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コネクトする場合はこの指定を行って下さい。
  また1次局に指定する必要があります。

データ待ち時間は秒単位、待ちなし、無限待ちを指定することができます。
データ待ちには以下の状態があり、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。

- ボーレイト
- 最大ターンアラウンドタイム
- フレームデータサイズ
- ウインドウサイズ
- BOF数
- 最小ターンアラウンドタイム
- リンク開放時間

(21) IrCOMM強制終了

IrCOMMを強制終了します。
IrCOMMをオープン状態から初期状態 (クローズ状態) に設定します。
基本適にはIrCOMMクローズと同じ機能をもちますが、通信状態に関係なく直ちに赤外線デバイス電源OFF、赤外線通信用リソースのリリースを行います。

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。

ﾌﾟﾗｲｵﾘﾃｨ	ﾌｧﾝｸｼｮﾝｺｰﾙ名	備考
5	Ir_State_set, Ir_SetPortConfig	・ﾌﾟﾗｲｵﾘﾃｨ 4 以下のﾌｧﾝｸｼｮﾝより先に使用して下さい ・ﾘｾｯﾄ直後はﾃﾞﾌｫﾙﾄ値となります ・ｸﾛｰｽﾞ状態である必要があります ・設定値はﾘｾｯﾄされるまで有効です
4		・予約とします
3	Ir_Err_Get	・ﾌﾟﾗｲｵﾘﾃｨ 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	・ｸﾛｰｽﾞ状態である必要があります
1	Ir_Close, Ir_Read, Ir_Write, Ir_QueryTx, Ir_QuertRx, Ir_EROn, Ir_EROff, Ir_RSOn, Ir_Rsoff, Ir_BreakOn, Ir_BreakOff, Ir_CheckCD, Ir_CheckDR, Ir_CheckCS, Ir_CheckCI, Ir_CheckBreak, Ir_Init	・ｵｰﾌﾟ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。

機能	IrCOMMオープン	関数名	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

≪C言語インタフェース≫

[コーリングシーケンス]

```
H　ercd = Ir_Open( H sec );
```

[パラメータ]

H　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

[リターンパラメータ]

H　ercd　　:リターンコード
　　(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)

[リターンコード]

E_IROK　　:正常終了
E_IRNG　　:異常終了

≪備考≫

機能	IrCOMMクローズ	関数名	Ir_Close

IrCOMM (赤外線ポート) をクローズします。

- 通信用のデバイスおよびリソースのリリース
- 赤外線デバイス電源OFF
- ブレイクイベントのチェック
- LBチェック
- コネクト切断

≪C言語インタフェース≫

[コーリングシーケンス]

```
H    ercd = Ir_Close( void );
```

[パラメータ]

なし

[リターンパラメータ]

H　ercd　　:リターンコード
（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）

[リターンコード]

E_IROK　　:正常終了
E_IRNG　　:異常終了

≪備考≫

機能	データ読込み	関数名	Ir_Read

受信データの読込みを行います。

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

≪C 言語インタフェース≫

[コーリングシーケンス]

```
H    ercd = Ir_Read( B *buff、 UH ReadSize、 UH *GetSize );
```

[パラメータ]

B　*buff	:受信データを格納するバッファのポインタ
UH ReadSize	:受信データを格納するバッファのサイズ(バイト数)
UH *GetSize	:読込みデータ数 (受信バッファから読込みできたバイト数)

[リターンパラメータ]

H　ercd	:リターンコード (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)

[リターンコード]

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

≪備考≫

機能	データ書込み	関数名	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

≪C 言語インタフェース≫

[コーリングシーケンス]

```
H    ercd = Ir_Write( B *buff、 UH WriteSize、 UH *PutSize );
```

[パラメータ]

B　*buff	:送信データを格納するバッファのポインタ
UH WriteSize	:送信データ数 (送信バッファに書込むバイト数)
UH *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)

[リターンパラメータ]

H　ercd	:リターンコード (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)

[リターンコード]

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

≪備考≫

機能	送信データ数問合せ	関数名	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

≪C 言語インタフェース≫

[コーリングシーケンス]
　　H　ercd = Ir_QueryTx(H *SndDataSize);

[パラメータ]
　　H　*SndDataSize　　:未送出データ数（バイト）

[リターンパラメータ]
　　H　ercd　　:リターンコード
　　（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）

[リターンコード]
　　E_IROK　　:正常終了
　　E_IRNG　　:異常終了

≪備考≫

機能	受信データ数問合せ	関数名	Ir_QueryRx

受信バッファより読込み可能なデータ数を問合せます。
- ・受信バッファ内の読込み可能なデータ数の通知
- ・ＬＢチェック
- ・ブレイクイベントのチェック

≪C 言語インタフェース≫

[コーリングシーケンス]

```
H   ercd = Ir_QueryRx( H *RcvDataSize )
```

[パラメータ]

```
H   *RcvDataSize     :読込み可能なデータ数（バイト）
```

[リターンパラメータ]

```
H   ercd             :リターンコード
                      （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）
```

[リターンコード]

```
E_IROK               :正常終了
E_IRNG               :異常終了
```

≪備考≫

機能	ＥＲ信号ＯＮ	関数名	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)

≪Ｃ言語インタフェース≫

[コーリングシーケンス]

```
H    ercd = Ir_EROn( void );
```

[パラメータ]

なし

[リターンパラメータ]

H　ercd　　:リターンコード
　　(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)

[リターンコード]

E_IROK　　:正常終了
E_IRNG　　:異常終了

≪備考≫

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。

機能	ＥＲ信号ＯＦＦ	関数名	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)

≪Ｃ言語インタフェース≫

[コーリングシーケンス]
```
H    ercd = Ir_EROff( void );
```

[パラメータ]
なし

[リターンパラメータ]
H　ercd　　:リターンコード
　　(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)

[リターンコード]
E_IROK　　:　正常終了
E_IRNG　　:　異常終了

≪備考≫
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。

機能	RS信号ON	関数名	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)

≪C言語インタフェース≫

[コーリングシーケンス]

```
H    ercd = Ir_RSOn( void );
```

[パラメータ]

なし

[リターンパラメータ]

H　ercd　　:リターンコード
　　(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)

[リターンコード]

E_IROK　　:正常終了
E_IRNG　　:異常終了

≪備考≫

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。

機能	ＲＳ信号ＯＦＦ	関数名	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)

≪Ｃ言語インタフェース≫

[コーリングシーケンス]

```
H    ercd = Ir_RSOff( void );
```

[パラメータ]

なし

[リターンパラメータ]

H　ercd	:リターンコード (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)

[リターンコード]

E_IROK	:正常終了
E_IRNG	:異常終了

≪備考≫

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。

機能	ブレーク送信ON	関数名	Ir_BreakOn

ブレーク信号を送出します。IrDAプロトコルによる信号のエミュレートになります。
- ・LBチェック
- ・ブレイクイベントのチェック
- ・信号制御データの作成
- ・送信データの書込み (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)

≪C言語インタフェース≫

[コーリングシーケンス]

```
H    ercd = Ir_BreakOn( void );
```

[パラメータ]

なし

[リターンパラメータ]

H　ercd　　:リターンコード
　　(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)

[リターンコード]

E_IROK　　:正常終了
E_IRNG　　:異常終了

≪備考≫

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。

機能	ブレーク送信ＯＦＦ	関数名	Ir_BreakOff

ブレーク信号の送出を停止します。IrＤＡプロトコルによる信号線のエミュレートになります。
- ・ＬＢチェック
- ・ブレイクイベントのチェック
- ・信号制御データの作成
- ・送信データの書込み (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)

≪Ｃ言語インタフェース≫

[コーリングシーケンス]

```
H    ercd = Ir_BreakOff( void );
```

[パラメータ]

なし

[リターンパラメータ]

H　ercd　　:リターンコード
　　(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)

[リターンコード]

E_IROK　　:正常終了
E_IRNG　　:異常終了

≪備考≫

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。

機能	CD検査	関数名	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

≪C言語インタフェース≫

[コーリングシーケンス]

```
H    ercd = Ir_CheckCD( H line );
```

[パラメータ]

```
H    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
```

[リターンパラメータ]

```
H    ercd          :リターンコード
                     (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)
```

[リターンコード]

```
E_IRLINE_ON    :信号ON
E_IRLINE_OFF   :信号OFF
E_IRNG         :異常終了
```

≪備考≫

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。

機能	DR検査	関数名	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

≪C言語インタフェース≫

[コーリングシーケンス]

```
H    ercd = Ir_CheckDR( void );
```

[パラメータ]

なし

[リターンパラメータ]

H　ercd　　:リターンコード
　　（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）

[リターンコード]

E_IRLINE_ON　:信号ON
E_IRLINE_OFF　:信号OFF
E_IRNG　　:異常終了

≪備考≫

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。

機能	CS検査	関数名	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

≪C言語インタフェース≫

[コーリングシーケンス]

```
H    ercd = Ir_CheckCS( void );
```

[パラメータ]

なし

[リターンパラメータ]

H　ercd　　:リターンコード
(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)

[リターンコード]

E_IRLINE_ON　:信号ON
E_IRLINE_OFF　:信号OFF
E_IRNG　　:異常終了

≪備考≫

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。

機能	CI検査	関数名	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

≪C言語インタフェース≫

[コーリングシーケンス]

```
H    ercd = Ir_CheckCI( void );
```

[パラメータ]

なし

[リターンパラメータ]

H　ercd　　:リターンコード
（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）

[リターンコード]

E_IRLINE_ON　:信号ON
E_IRLINE_OFF　:信号OFF
E_IRNG　　:異常終了

≪備考≫

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。

機能	ＢＲＥＡＫ検査	関数名	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

≪Ｃ言語インタフェース≫

[コーリングシーケンス]
　　H　ercd = Ir_CheckBreak(void);

[パラメータ]
　　なし

[リターンパラメータ]
　　H　ercd　　：リターンコード
　　（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）

[リターンコード]
　　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

≪備考≫

機能	エラー値取得	関数名	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

≪C 言語インタフェース≫

[コーリングシーケンス]
　　UW　wercd = Ir_Err_Get(void);

[パラメータ]
　　なし

[リターンパラメータ]
　　UW　wercd　　:リターンコード

[リターンコード]
　　詳細は、次ページを参照して下さい

≪備考≫

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 ｵｰﾌﾟﾝ状態	IrDA部の上位が行う発生したエラーに対しての事後処理

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

エラー値	IRERR_NODEVICE	
詳細 回線上(空間)にコネクト可能なデバイスがないとき発生します IRERR_DISCONNECTエラーの要因として一緒に通知します Ir_Open関数でのコネクト待ちタイムアウトの要因でもあります		
関数名	IrCOMM 状態	主なエラー対処方法
Ir_Open	ｵｰﾌﾟﾝは行わずｸﾛｰｽﾞ状態	･IOボックスの装着を所定の位置に正しく固定して再実行して下さい ･デバイス同士を20cm以内に接近させて再実行して下さい

エラー値	IRERR_NOLSAP	
詳細 回線上(空間)にコネクト可能なアプリケーションが無いときやIrDAプロトコルの実装が異なる(相手局にIrCOMM層がない)ときに発生します IRERR_DISCONNECTエラーの要因として一緒に通知します Ir_Open関数でのコネクト待ちタイムアウトの要因でもあります		
関数名	IrCOMM 状態	主なエラー対処方法
Ir_Open	ｵｰﾌﾟﾝは行わずｸﾛｰｽﾞ状態	･相手局のプロトコル実装を確認して再実行して下さい

エラー値	IRERR_LOCK	
詳細 通信用のデバイスおよびリソースが既にロックされているときに発生します 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	ｵｰﾌﾟﾝは行わずｸﾛｰｽﾞ状態	･デバイス、リソースが開放されてから再実行して下さい ･既に IrCOMM(赤外線ﾎﾟｰﾄ)がｵｰﾌﾟﾝしています。ｸﾛｰｽﾞしてから 再実行して下さい

エラー値	IRERR_DISCONNECT

詳細
コネクト手続き中またはコネクト後に相手局からの応答が無くなったとき、相手局からコネクト切断されたとき、レジュームON立上げを行ったときに発生します

関数名	IrCOMM 状態	主なエラー対処方法
Ir_Open	ｵｰﾌﾟﾝは行わずｸﾛｰｽﾞ状態	・通信環境を確認して再実行して下さい
Ir_Close	ｸﾛｰｽﾞ状態となります	・相手局と通信不可能な環境にあるのでその原因を取り除いてIrCOMM(赤外線ﾎﾟｰﾄ)のｵｰﾌﾟﾝを行って下さい 相手局から一定時間応答がない（回線が外れている）場合が考えられます
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	・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	ｵｰﾌﾟﾝは行わずｸﾛｰｽﾞ状態	・再実行する場合は Ir_Open から行って下さい
Ir_Close	ｸﾛｰｽﾞ状態となります	
Ir_Read	ｵｰﾌﾟﾝ状態から変更はありません	・Ir_Close を行って終了して下さい
Ir_Write		
Ir_EROn		
Ir_EROff		
Ir_RSOn		
Ir_RSOff		
Ir_BreakOn		
Ir_BreakOff		
Ir_CheckCD		
Ir_CheckDR		
Ir_CheckCS		
Ir_QueryTx		
Ir_QueryRx		
Ir_CheckCI		
Ir_CheckBreak		

エラー値	IRERR_LB0

詳細
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

関数名	IrCOMM 状態	主なエラー対処方法
Ir_Open	ｵｰﾌﾟﾝは行わずｸﾛｰｽﾞ状態	・再実行するときは Ir_Open を行って下さい ・イベントのクリアを行って下さい
Ir_Close	ｸﾛｰｽﾞ状態となります	
Ir_Read		
Ir_Write		
Ir_EROn		
Ir_EROff		
Ir_RSOn		
Ir_RSOff		
Ir_BreakOn		
Ir_BreakOff		
Ir_CheckCD		
Ir_CheckDR		
Ir_CheckCS		
Ir_QueryTx		
Ir_QueryRx		
Ir_CheckCI		
Ir_CheckBreak		
Ir_Init		

エラー値	IRERR_LB1

詳細
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	IrCOMM 状態	主なエラー対処方法
Ir_Open	ｵｰﾌﾟﾝは行わずｸﾛｰｽﾞ状態	・電池交換を行って下さい ・イベントのクリアを行って下さい ・再実行するときは Ir_Open を行って下さい
Ir_Close	ｸﾛｰｽﾞ状態となります	
Ir_Init		
Ir_Read	ｵｰﾌﾟﾝ状態から変更はありません	・Ir_Close を行って終了して下さい ・電池交換交換後に Ir_Open を行って下さい ・イベントのクリアを行って下さい
Ir_Write		
Ir_EROn		
Ir_EROff		
Ir_RSOn		
Ir_RSOff		
Ir_BreakOn		
Ir_BreakOff		
Ir_CheckCD		
Ir_CheckDR		
Ir_CheckCS		
Ir_QueryTx		
Ir_QueryRx		
Ir_CheckCI		
Ir_CheckBreak		

エラー値	IRERR_LB2

詳細
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	ｵｰﾌﾟﾝは行わずｸﾛｰｽﾞ状態	・電池交換後を行って下さい ・イベントのクリアを行って下さい ・再実行するときは Ir_Open を行って下さい
Ir_Close	ｸﾛｰｽﾞ状態となります	
Ir_Init		
Ir_Read	ｵｰﾌﾟﾝ状態から変更はありません	・Ir_Close を行って終了して下さい ・電池交換後に Ir_Open を行って下さい ・イベントのクリアを行って下さい
Ir_Write		
Ir_EROn		
Ir_EROff		
Ir_RSOn		
Ir_RSOff		
Ir_BreakOn		
Ir_BreakOff		
Ir_CheckCD		
Ir_CheckDR		
Ir_CheckCS		
Ir_QueryTx		
Ir_QueryRx		
Ir_CheckCl		
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	ｵｰﾌﾟﾝは行わずｸﾛｰｽﾞ状態	・イベントのクリアを行って下さい ・電源ＯＦＦして下さい
Ir_Close	ｸﾛｰｽﾞ状態となります	
Ir_Init		
Ir_Read	ｵｰﾌﾟﾝ状態から変化はありません	・Ir_Close を行って終了して下さい ・イベントのクリアを行って下さい ・電源ＯＦＦして下さい
Ir_Write		
Ir_EROn		
Ir_EROff		
Ir_RSOn		
Ir_RSOff		
Ir_BreakOn		
Ir_BreakOff		
Ir_CheckCD		
Ir_CheckDR		
Ir_CheckCS		
Ir_QueryTx		
Ir_QueryRx		
Ir_CheckCl		
Ir_CheckBreak		

エラー値	IRERR__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	IrCOMM 状態	主なエラー対処方法
Ir_Open	ｵｰﾌﾟﾝは行わずｸﾛｰｽﾞ状態	・イベントのクリアを行って下さい ・電源ＯＦＦして下さい
Ir_Close	ｸﾛｰｽﾞ状態となります	
Ir_Init		
Ir_Read	ｵｰﾌﾟﾝ状態から変化はありません	・Ir_Close を行って終了して下さい ・イベントのクリアを行って下さい ・電源ＯＦＦして下さい
Ir_Write		
Ir_EROn		
Ir_EROff		
Ir_RSOn		
Ir_RSOff		
Ir_BreakOn		
Ir_BreakOff		
Ir_CheckCD		
Ir_CheckDR		
Ir_CheckCS		
Ir_QueryTx		
Ir_QueryRx		
Ir_CheckCI		
Ir_CheckBreak		

エラー値	IRERR__NOTOPEN

詳細
IrＣＯＭＭ (赤外線ポート) がオープンされていないときに発生します

関数名	IrCOMM 状態	主なエラー対処方法
Ir_Close	ｸﾛｰｽﾞ状態から変化はありません	・IrCOMM が既にｸﾛｰｽﾞ状態になっています Ir_Open を行ってから実行して下さい
Ir_Read		
Ir_Write		
Ir_QueryTX		
Ir_QueryRx		
Ir_EROn		
Ir_EROff		
Ir_RSOn		
Ir_RSOff		
Ir_BreakOn		
Ir_BreakOff		
Ir_CheckCD		
Ir_CheckDR		
Ir_CheckCS		
Ir_CheckCI		
Ir_CheckBreak		
Ir_Init		

エラー値	IRERR_TIMEOUT

詳細
Ir_Opne関数で指定したコネクト待ち時間を経過した場合と、Ir_State_Set関数で指定したデータ待ち時間を経過すると発生します

関数名	IrCOMM 状態	主なエラー対処方法
Ir_Open	ｸﾛｰｽﾞ状態です	・通信環境を確認して再実行して下さい
Ir_Read	ｵｰﾌﾟﾝ状態に変化はありません	・任意の処理を行って下さい。再実行してもかまいません
Ir_Write		
Ir_RSOn		
Ir_RSOff		
Ir_EROn		
Ir_EROff		
Ir_BreakOn		
Ir_BreakOff		

エラー値	IRERR_PARITY

詳細
Ir_State_Set関数で以下の指定のとき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
・3Wireまたは9Wire
・パリティあり

関数名	IrCOMM 状態	主なエラー対処方法
Ir_Read	ｵｰﾌﾟﾝ状態に変化はありません	・任意の処理を行って下さい

エラー値	IRERR_OVERRUN

詳細
Ir_State_Set関数で以下の指定のとき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
・3Wireまたは9Wire

関数名	IrCOMM 状態	主なエラー対処方法
Ir_Read	ｵｰﾌﾟﾝ状態に変化はありません	・任意の処理を行って下さい

エラー値	IRERR_FRAMING

詳細
Ir_State_Set関数で以下の指定のとき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
・3Wireまたは9Wire

関数名	IrCOMM 状態	主なエラー対処方法
Ir_Read	ｵｰﾌﾟﾝ状態に変化はありません	・任意の処理を行って下さい

エラー値	IRERR_WIRE_TYPE

詳細
Ir_State_Set関数で3Wireあるいは9Wireが指定されている場合しか使用できない関数を使用したとき発生します

関数名	IrCOMM 状態	主なエラー対処方法
Ir_EROn	ｵｰﾌﾟ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	ｵｰﾌﾟﾝ状態に変化は 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	ｵｰﾌﾟﾝ状態に変化はありません	・任意の処理を行って下さい。再実行してもかまいません

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	ｵｰﾌﾟﾝ状態に変化はありません	・任意の処理を行って下さい。再実行してもかまいません

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	ｵｰﾌﾟﾝ状態に変化はありません	・任意の処理を行って下さい。再実行してもかまいません

機能	通信状態設定	関数名	Ir_State_Set

IrDA部の通信状態を設定します。

- Wireの指定
- データ読込み／書込み（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

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

```
struct   *State_DCB
```

[ストラクト構造]

```
struct State_DCB      {
                        H   station;          :局
                        H   Wire;             :Wire
                        H   DataWaitTime;     :データ待ち時間
                        H   LineWaitTime;     :DR／CS／CD信号待ち時間
                        H   baudRate;         :RS232Cの通信速度
                        H   DataLen;          :RS232Cのデータ長
                        H   StopBit;          :RS232Cのストップビット
                        H   ParityBit;        :RS232Cのパリティビット
                        };
```

[リターンパラメータ]

```
H   ercd        :リターンコード
                 （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）
```

[リターンコード]

```
E_IROK          :正常終了
E_IRNG          :異常終了
```

≪備考≫

- 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	秒単位にﾃﾞｰﾀ読込み/書込み待ち時間を設定	○(300)
	THROUGH	ﾃﾞｰﾀ読込み/書込み待ちを行わない	
	FOREVER	ﾀｲﾏ指定なしでﾃﾞｰﾀ読込み/書込み待ちを行う	
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 のﾃﾞｰﾀ長	LEN_7B	RS232C のﾃﾞｰﾀ長を 7bit に設定	
	LEN_8B	RS232C のﾃﾞｰﾀ長を 8bit に設定	○
RS232C のｽﾄｯﾌﾟﾋﾞｯﾄ	STOP_1B	RS232C のｽﾄｯﾌﾟﾋﾞｯﾄを 1bit に設定	○
	STOP_2B	RS232C のｽﾄｯﾌﾟﾋﾞｯﾄを 2bit に設定	
RS232C のﾊﾟﾘﾃｨﾋﾞｯﾄ	PRI_ODD	RS232C のﾊﾟﾘﾃｨﾋﾞｯﾄを奇数ﾊﾟﾘﾃｨに設定	
	PRI_EVN	RS232C のﾊﾟﾘﾃｨﾋﾞｯﾄを偶数ﾊﾟﾘﾃｨに設定	
	PRI_NON	RS232C のﾊﾟﾘﾃｨﾋﾞｯﾄをﾊﾟﾘﾃｨなしに設定	○

機能	自局能力設定	関数名	Ir_SetPortConfig

自局能力を設定します。
・折衝パラメータの設定

≪C 言語インタフェース≫

[コーリングシーケンス]
　　H　ercd = Ir_SetPortConfig(struct *SetPortConfig_DCB);

[パラメータ]
　　struct　*SetPortConfig_DCB

[ストラクト構造]
```
　　struct SetPortConfig_DCB {
                                    UB   irBaud;          :ボーレイト
                                    UB   MaxTurnTime;     :最大ターンアラウンドタイム
                                    UB   FrameSize;       :フレームサイズ
                                    UB   WindowSize;      :ウインドウサイズ
                                    UB   BofCount;        :ＢＯＦ数
                                    UB   MinTurnTime;     :最小ターンアラウンドタイム
                                    UB   DiscTime;        :リンク開放時間
                                    };
```

[リターンパラメータ]
　　H　ercd　　:リターンコード
　　（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）

[リターンコード]
　　E_IROK　　:正常終了
　　E_IRNG　　:異常終了

≪備考≫
・パラメータについて詳しくは次ページを参照して下さい
・オープンに先立って使用して下さい

自局能力設定のＤＣＢ

項目	定数	詳細	デフォルト
IR ﾎﾞｰﾚｰﾄ ・OR で設定して下さい 設定した値が有効となります	IRBPS_24	IR 接続速度を 2400bps に設定可能	設定有効
	IRBPS_96	IR 接続速度を 9600bps に設定可能	設定有効
	IRBPS_192	IR 接続速度を 19200bps に設定可能	設定有効
	IRBPS_384	IR 接続速度を 38400bps に設定可能	設定有効
	IRBPS_576	IR 接続速度を 57600bps に設定可能	設定有効
	IRBPS_1152	IR 接続速度を 115200bps に設定可能	設定有効
最大ﾀｰﾝｱﾗｳﾝﾄﾞﾀｲﾑ	TURN_500MS	最大ﾀｰﾝｱﾗｳﾝﾄﾞﾀｲﾑを 500ms に設定	○
ﾌﾚｰﾑｻｲｽﾞ	FRAME_1024B	ﾌﾚｰﾑｻｲｽﾞを 1024byte に設定	○
ｳｲﾝﾄﾞｳｻｲｽﾞ	WINDOW_4	ｳｲﾝﾄﾞｳｻｲｽﾞを 4 ﾌﾚｰﾑｳｲﾝﾄﾞｳに設定	○
BOF 数 ・IR ﾎﾞｰﾚｲﾄにより比例して増減します BOF 数は 115．2k の場合です	BOF_48	BOF を 48 個追加	○
	BOF_24	BOF を 24 個追加	
	BOF_12	BOF を 12 個追加	
	BOF_5	BOF を 5 個追加	
	BOF_3	BOF を 3 個追加	
	BOF_2	BOF を 2 個追加	
	BOF_1	BOF を 1 個追加	
	BOF_0	BOF を 0 個追加	
最小ﾀｰﾝｱﾗｳﾝﾄﾞﾀｲﾑ	TURN_5MS	最小ﾀｰﾝｱﾗｳﾝﾄﾞﾀｲﾑを 5ms に設定	
	TURN_1MS	最小ﾀｰﾝｱﾗｳﾝﾄﾞﾀｲﾑを 1ms に設定	
ﾘﾝｸ開放時間 ・OR で設定して下さい 設定した値が有効となります	RELEASE_3S	ﾘﾝｸを開放する時間を 3s に設定可能	設定有効
	RELEASE_8S	ﾘﾝｸを開放する時間を 8s に設定可能	設定有効
	RELEASE_12S	ﾘﾝｸを開放する時間を 12s に設定可能	設定有効
	RELEASE_16S	ﾘﾝｸを開放する時間を 16s に設定可能	設定有効
	RELEASE_20S	ﾘﾝｸを開放する時間を 20s に設定可能	設定有効
	RELEASE_25S	ﾘﾝｸを開放する時間を 25s に設定可能	設定有効
	RELEASE_30S	ﾘﾝｸを開放する時間を 30s に設定可能	設定有効
	RELEASE_40S	ﾘﾝｸを開放する時間を 40s に設定可能	設定有効

機能	IrCOMM強制終了	関数名	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

≪C言語インタフェース≫

[コーリングシーケンス]

```
H    ercd = Ir_Init( void );
```

[パラメータ]

なし

[リターンパラメータ]

H　ercd　　:リターンコード
（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）

[リターンコード]

E_IROK　　:正常終了
E_IRNG　　:異常終了

≪備考≫

8. 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。
・ 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。

8.1. 通信インタフェース

8.1.1. 使用形態

本機では、2つの回線ポート(赤外線／シリアルインタフェース)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			COM1(ｼﾘｱﾙ)接続機器		備考
			I／O BOX *1	本機	IRDA ｱﾀﾞﾌﾟﾀ	PC	本機	
マルチドロップ	受信	ｱﾌﾟﾘｹｰｼｮﾝ	◎	×	×	○	×	＊2
		ﾃﾞｰﾀﾌｧｲﾙ	◎	×	×	○	×	
	送信	ｱﾌﾟﾘｹｰｼｮﾝ	×	×	×	×	×	
		ﾃﾞｰﾀﾌｧｲﾙ	◎	×	×	○	×	
FLINK	受信	ｱﾌﾟﾘｹｰｼｮﾝ	◎	○	○	○	○	
		ﾃﾞｰﾀﾌｧｲﾙ	◎	○	○	○	○	
	送信	ｱﾌﾟﾘｹｰｼｮﾝ	◎	○	○	○	○	
		ﾃﾞｰﾀﾌｧｲﾙ	◎	○	○	○	○	
DT500	受信	ｱﾌﾟﾘｹｰｼｮﾝ	◎	×	×	○	×	＊3
		ﾃﾞｰﾀﾌｧｲﾙ	◎	×	×	○	×	＊3
	送信	ｱﾌﾟﾘｹｰｼｮﾝ	◎	×	×	○	×	＊3
		ﾃﾞｰﾀﾌｧｲﾙ	◎	×	×	○	×	＊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。

8.1.2. 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		
	マスタIO(IrDA)	サテライトIO(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

8.1.3. 排他制御

複数プロトコルは同時使用できません。

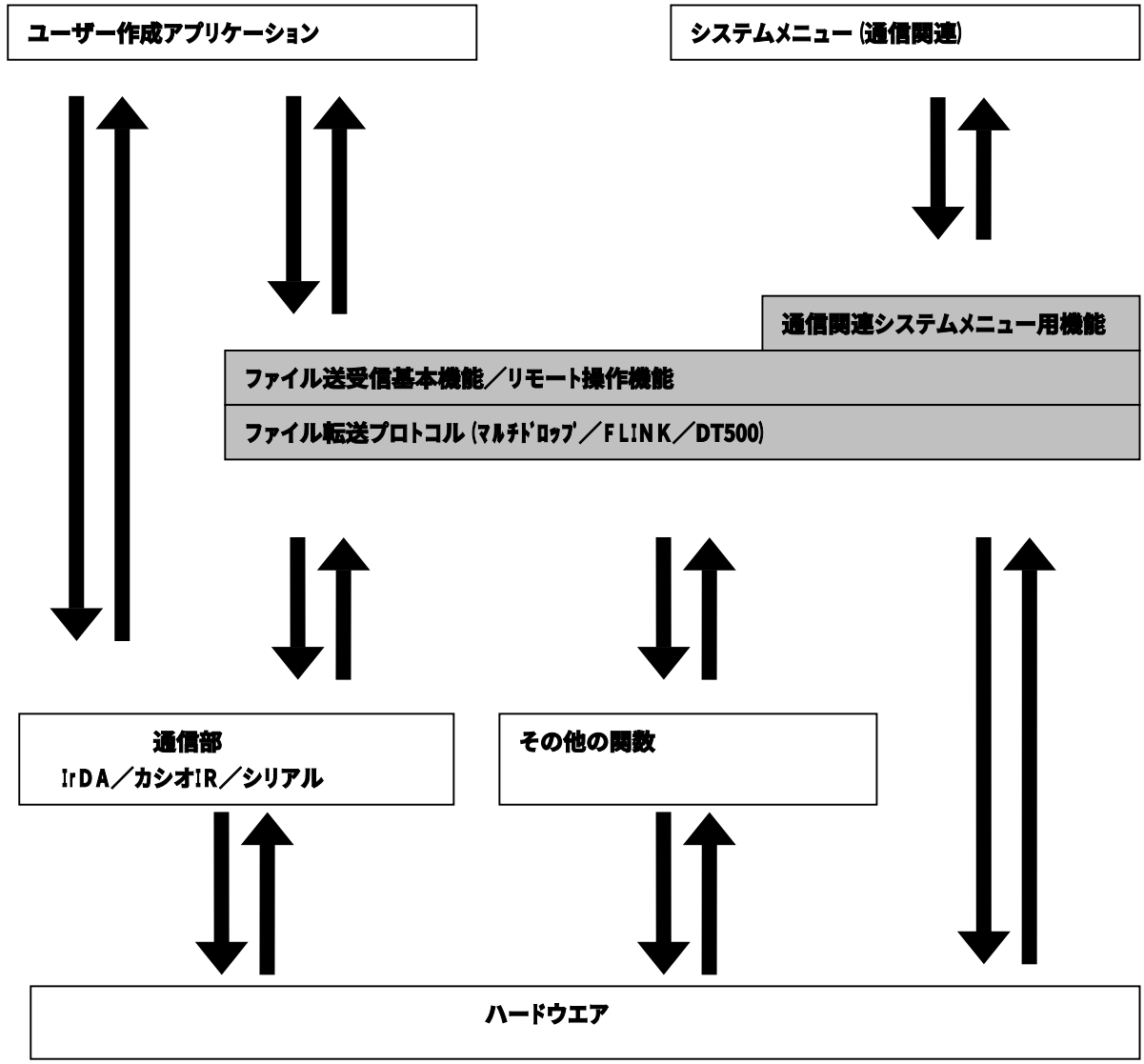
8.1.4. 転送ドライブ

各プロトコルでは、Aドライブ(RAMドライブ)、Bドライブ(バックアップ用ドライブ)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。

8.2.　ソフトウエアブロック構成図

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

図8．1 ソフトウエアブロック構成



8.3. マルチドロップ

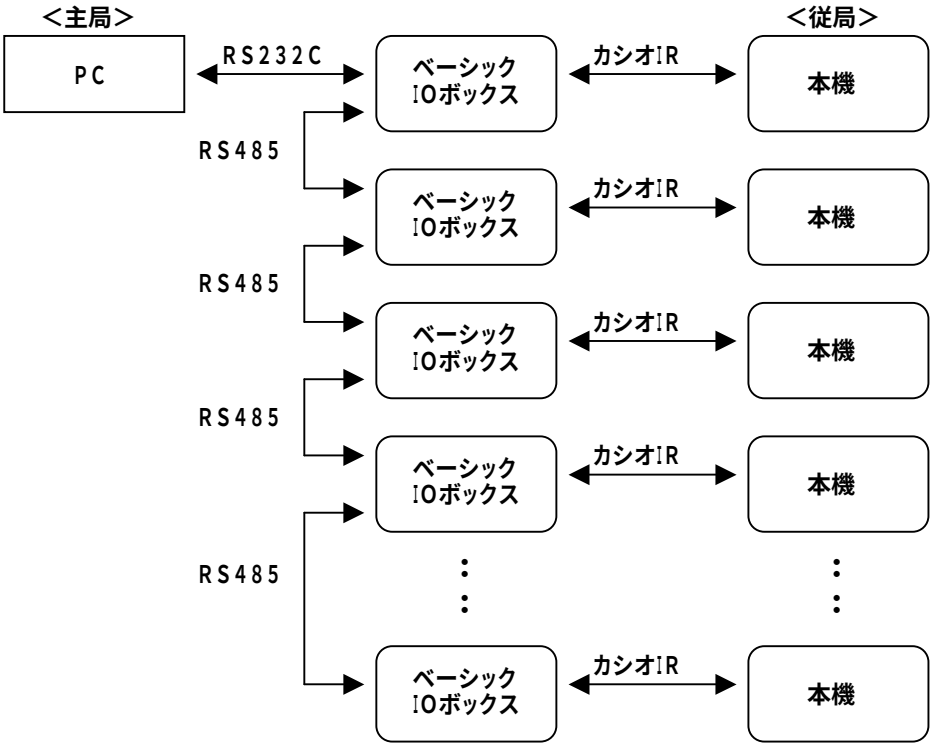
8.3.1. 通信仕様

(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専用ケーブルでのシリアルインタフェース通信

(2) 通信パラメータ

	関数	システムメニュー用通信機能	備考
共通ﾊﾟﾗﾒｰﾀ			
データ長	8ﾋﾞｯﾄ固定		
パリティビット	選択可(奇数/偶数/なし)	ｼｽﾃﾑ環境設定 (データ管理部) より取得	
ストップビット	選択可(1ﾋﾞｯﾄ/2ﾋﾞｯﾄ)	ｼｽﾃﾑ環境設定 (データ管理部) より取得	
通常受信ﾀｲﾑｱｳﾄ ＊1	ｼｽﾃﾑ環境設定 (データ管理部) より取得 (00 - 99 秒)		推奨値 3秒
通常ﾘﾄﾗｲ回数 ＊2	ｼｽﾃﾑ環境設定 (データ管理部) より取得 (00 - 99 回)		推奨値 3回
ﾎﾟｰﾘﾝｸﾞ受信中ﾀｲﾑｱｳﾄ ＊3	3秒固定		＊5
ﾃﾞ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	
COM1 (ｼﾘｱﾙ)	選択可(1200～115200bps)	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		

通信速度に関して

マルチドロップとして提供されているダウンロードプログラムは最大19200bps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MAX115200bpsをサポートします。

＊1 通常受信タイムアウト
- 通常の通信中における受信のタイムアウト時間。

＊2 通常リトライ回数
- 通常の通信中における送受信のリトライ回数。

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

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

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

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

(3) ファイル受信 (主局から従局へのファイル転送)

主局のファイルを従局へ送信する。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)

従局のファイルを主局へ送信します。

- 複数ファイルを1回のデータリンクにて送信できます。尚、受信したファイルの書込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、送信側と受信側でに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。

8.3.2.　ファイル送受信基本機能

複数ファイルの送信および受信を行うための基本機能を提供します。

(1) 機能一覧

機能	内容
通信ポートの初期化	・回線ポートの初期化を行います 回線ポートは COM0：カシオIRインタフェース　COM1：シリアルインタフェースとします ・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
通信ポートのクローズ	・回線ポートのクローズを行います ・回線オープン時にセーブした APO 設定を復旧します
中断キーの登録／削除	・中断キーの選択登録／削除を行います
エラー詳細情報の取得	・エラー情報の取得を行います 通信ユーティリティエラー、関数エラーの取得が可能です
データリンク拒否情報の取得	・主局からデータリンクの拒否が行われた場合（エラー詳細情報のエラー状態がデータリンクエラー（主局拒否）の場合）、拒否理由値の取得が可能です

(2) ファイル送受信関数

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

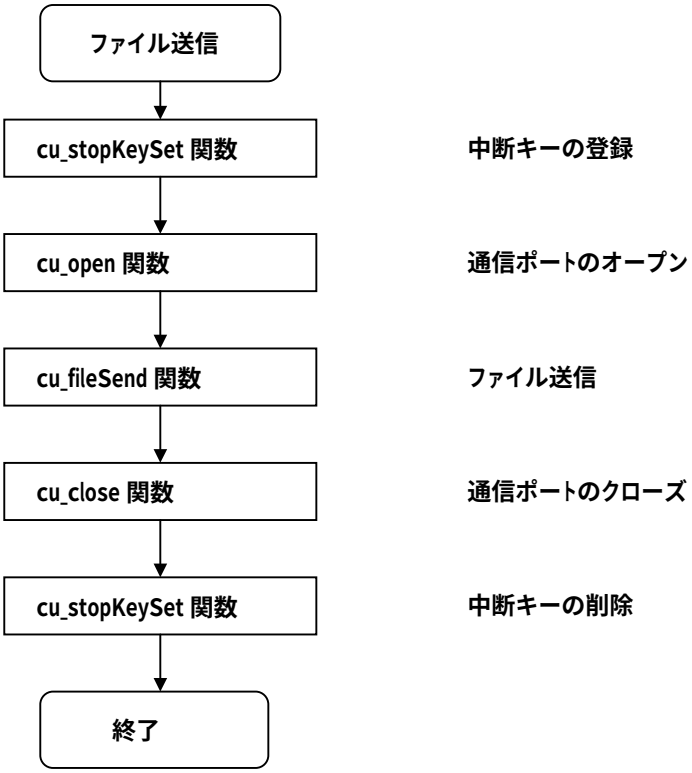
＊実際の表示位置は画面モードにより異なります。

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。

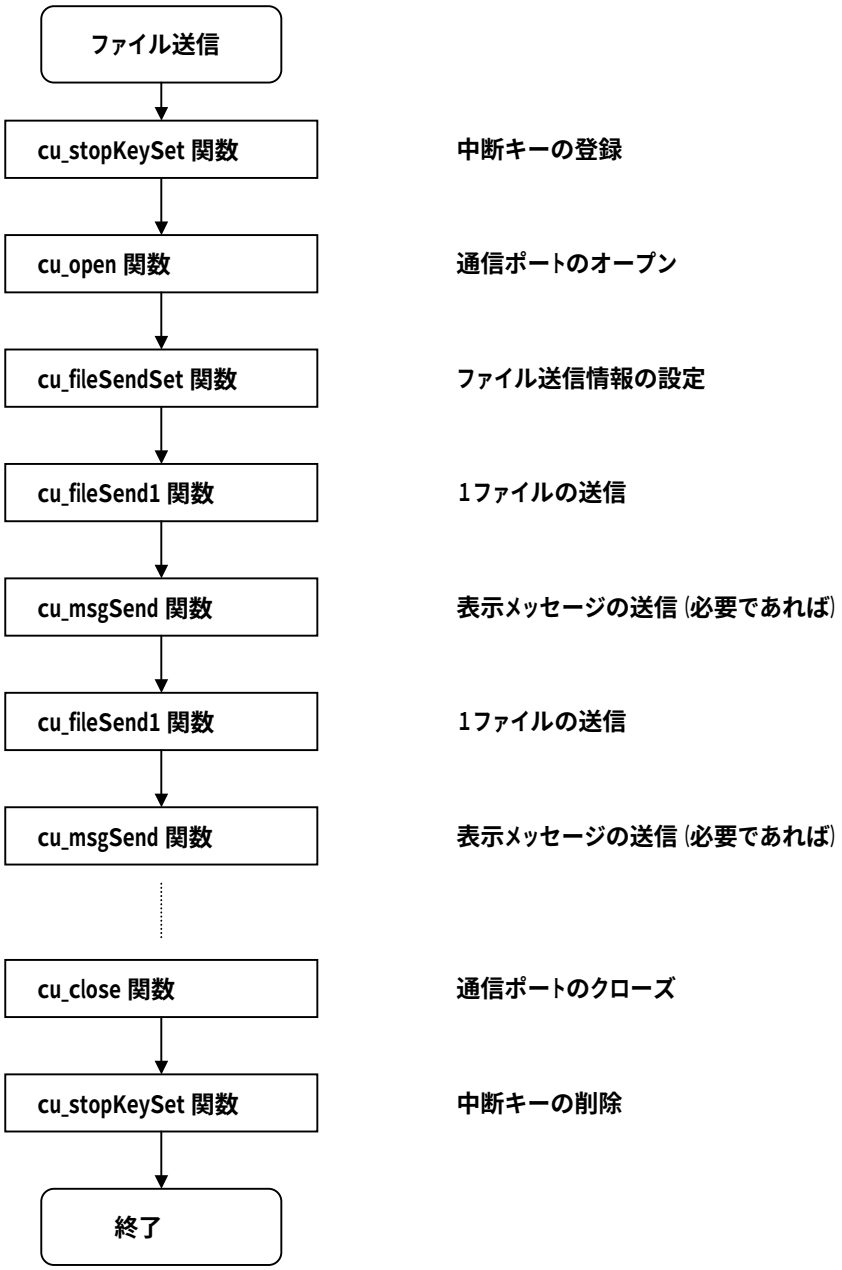
[ファイル送信関数によるファイル送信]

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。



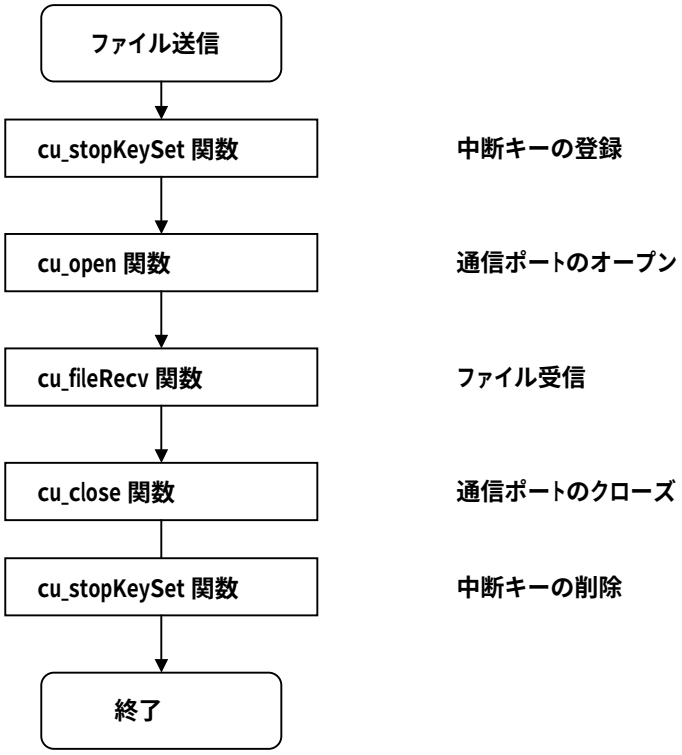
［1ファイル送信関数によるファイル送信］

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。



8.4. FLINKプロトコル機能

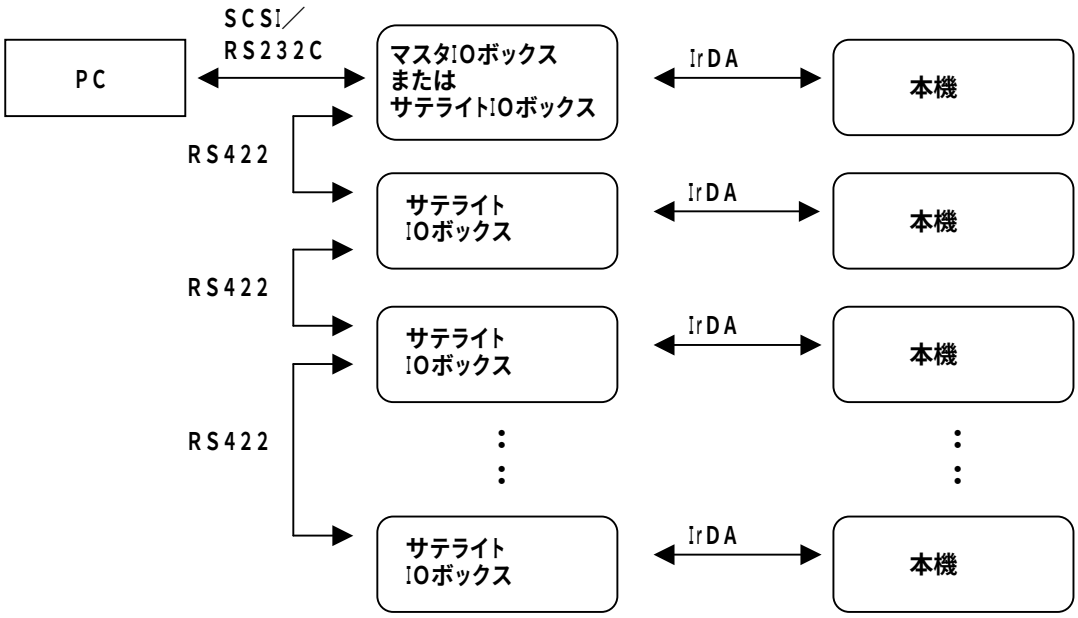
8.4.1. 通信仕様

(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。

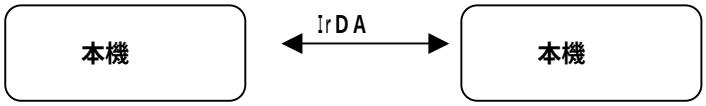
①マスタ・サテライトIOボックス経由でのホスト通信

マスタIOボックス・サテライトIOボックスの連鎖接続によるファイル転送ができます。
尚、ベーシックIOボックスによる通信は行えません。



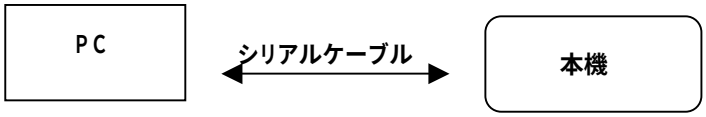
②赤外線による本体間通信

赤外線通信による本体間でのファイル転送ができます。



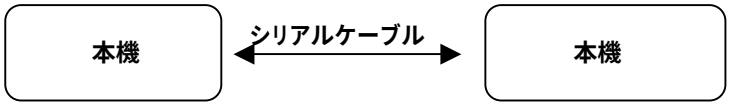
③シリアルインタフェース (専用ケーブル) によるホスト通信

専用ケーブルによるホスト PC とのファイル転送ができます。



④シリアルインタフェース (専用ケーブル) による本体間通信

赤外線通信による本体間でのファイル転送ができます。



(2) 通信パラメータ

	関数 (基本送受信関数 ﾘﾓｰﾄ操作関数)	システムメニュー用通信機能 (同報通信、本体間通信(子機作成)を含む)
共通ﾊﾟﾗﾒｰﾀ		
ｾｯｼｮﾝ確立待ちﾀｲﾑｱｳﾄ時間(＊1)	ｼｽﾃﾑ環境設定(データ管理部)より取得します (0 ～ 3600 秒)	
受信待ちﾀｲﾑｱｳﾄ時間(＊2)	ｼｽﾃﾑ環境設定(データ管理部)より取得します (0 ～ 600 秒)	
ｾｯｼｮﾝ終了待ちﾀｲﾑｱｳﾄ時間(＊3)	ｼｽﾃﾑ環境設定(データ管理部)より取得します (0 ～ 600 秒)	
COM0 (赤外線 IrDA Ver1.0)		
通信速度	選択可 (2400～115.4Kbps)	選択可 (ｼｽﾃﾑﾒﾆｭｰで選択)
COM1 (シリアル)		
通信速度	選択可 (1200～115.2Kbps)	選択可 (ｼｽﾃﾑﾒﾆｭｰで選択)
データ長	8ﾋﾞｯﾄ固定	8ﾋﾞｯﾄ固定
パリティビット	選択可 (奇数/偶数/なし)	選択可 (ｼｽﾃﾑﾒﾆｭｰで選択)
ストップビット	選択可 (1ﾋﾞｯﾄ/2ﾋﾞｯﾄ)	選択可 (ｼｽﾃﾑﾒﾆｭｰで選択)
フロー制御	ＲＳ／ＣＳフロー制御固定	

＊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

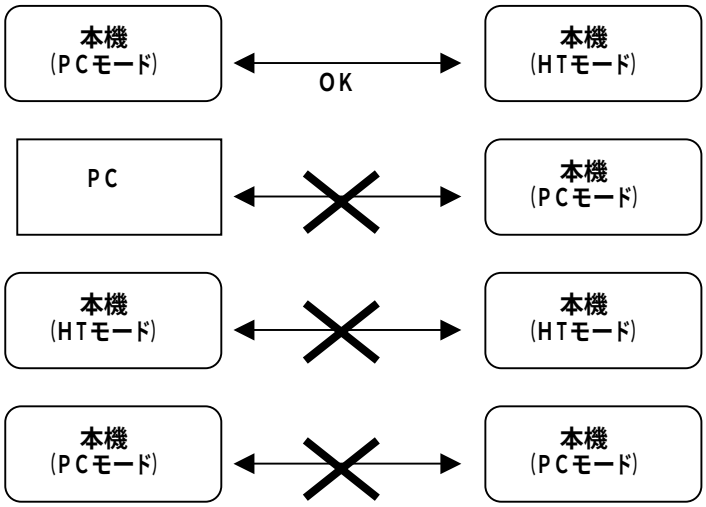
通信ユーティリティでは接続構成の違いにより以下のモードをサポートします。
これらのモードはオープン時に選択します。

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。
　　尚、１０ピン（ＲＳ２３２Ｃ）ではＤＲ　ＯＮにてセッションの確立と見なします。

(5) 処理概要

以下に各関数内の処理概要およびエラー発生時の処理を示します。
エラー発生時は、直ちに通信を終了します。
この場合、送信権の有無に関らず、先にエラーを検出した側がエラー情報（カテゴリコード・エラー詳細コード）を終了指示コマンドに設定し、相手局へ送信します。
相手局は、受信した終了指示コマンドのエラー情報により、異常終了を検出します。
（エラー処理は、自局内でエラーを検出した場合と、相手局からのエラー終了指示コマンドを受信した場合の両方を指します。）

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

8.4.2.　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)　COM1：シリアルインタフェースとします。
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)
COM1 (10 ピン)	
通信速度	CU_B1200(1200bps)／CU_B2400(2400bps)／CU_B4800(4800bps) CU_B9600(9600bps)／CU_B19K(19.2kbps)／CU_B38K(38.4kbps) CU_B57K(57.6kbps)／CU_B115K(115.2kbps)
データ長	CU_CHAR8(データ長 8 ビット)
パリティ	CU_PRI_ODD(奇数)／CU_PRI_EVN(偶数)／CU_PRI_NON(ﾊﾟﾘﾃｨなし)
ストップビット	CU_STOP1(ストップビット 1)／CU_STOP2(ストップビット 2)

[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

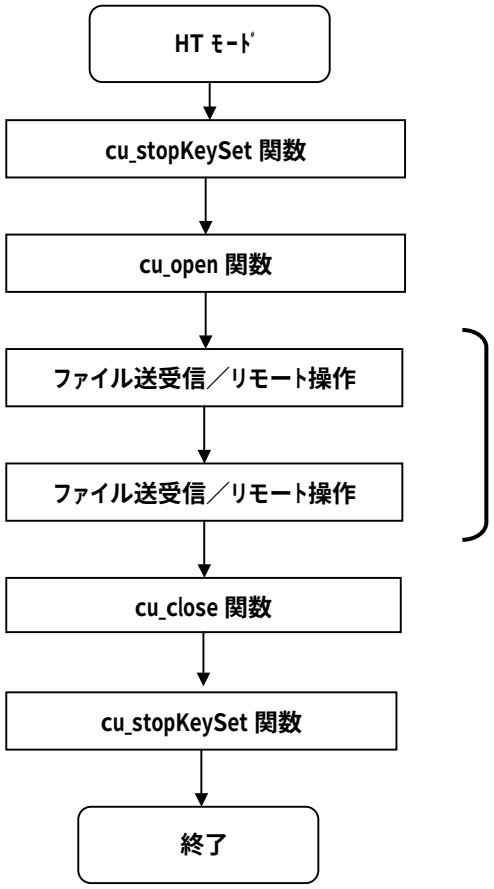
- 中断キーの選択登録／削除を行います。
  登録は回線オープンの前に、削除はクローズの後に行う事が必要です。

④エラー詳細情報の取得

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

HTコマンド送信による通信

［HTモード基本フロー］



中断キーの登録

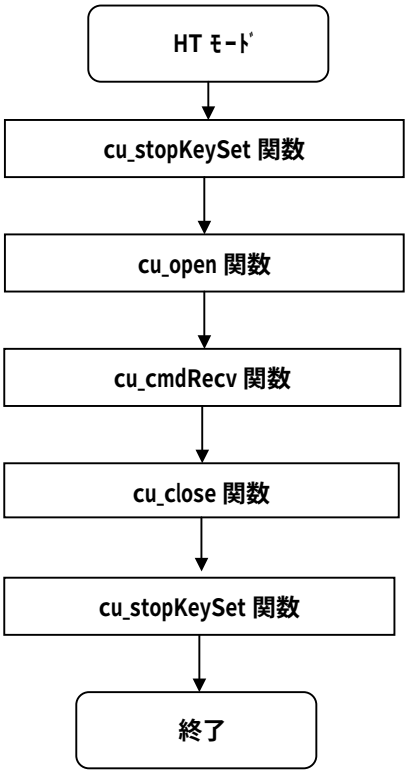
通信ポートのオープン／セッション確立

ファイル送受信／リモート操作関数
を順次実行します。

セッション解放／回線クローズ
（終了指示コマンド送信）

中断キーの削除

［PCモード基本フロー］



中断キーの登録

通信ポートのオープン／セッション確立

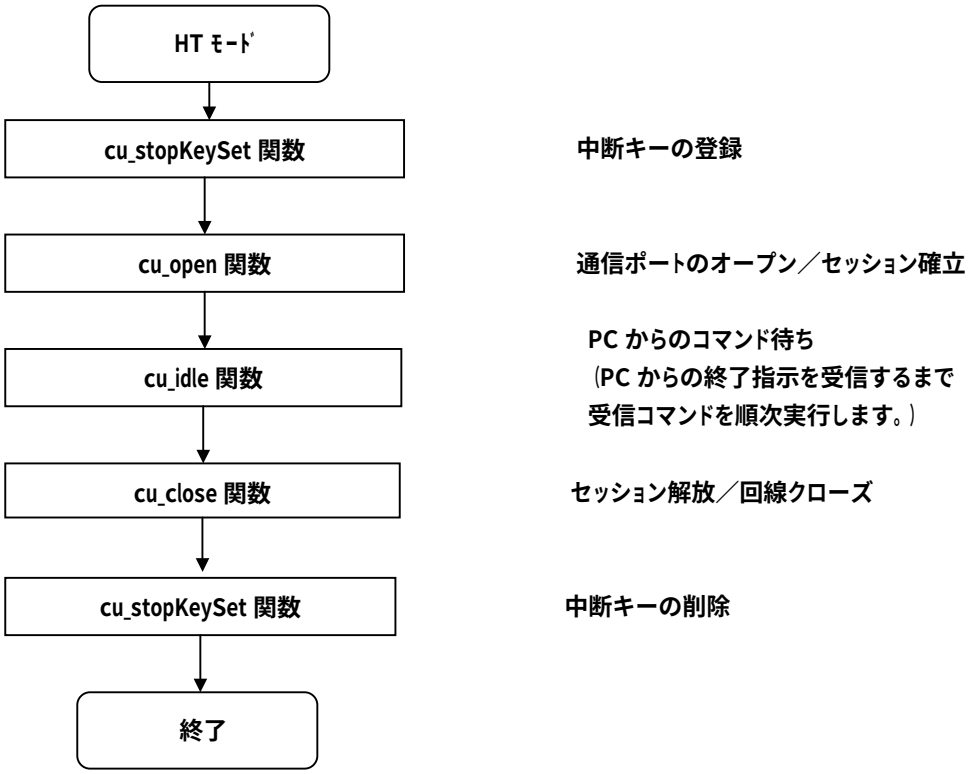
HT からのコマンド待ち
（HT からの終了指示、または IDLE 通知を
受信するまで、受信コマンドを順次実行します。）

セッション確立／回線クローズ

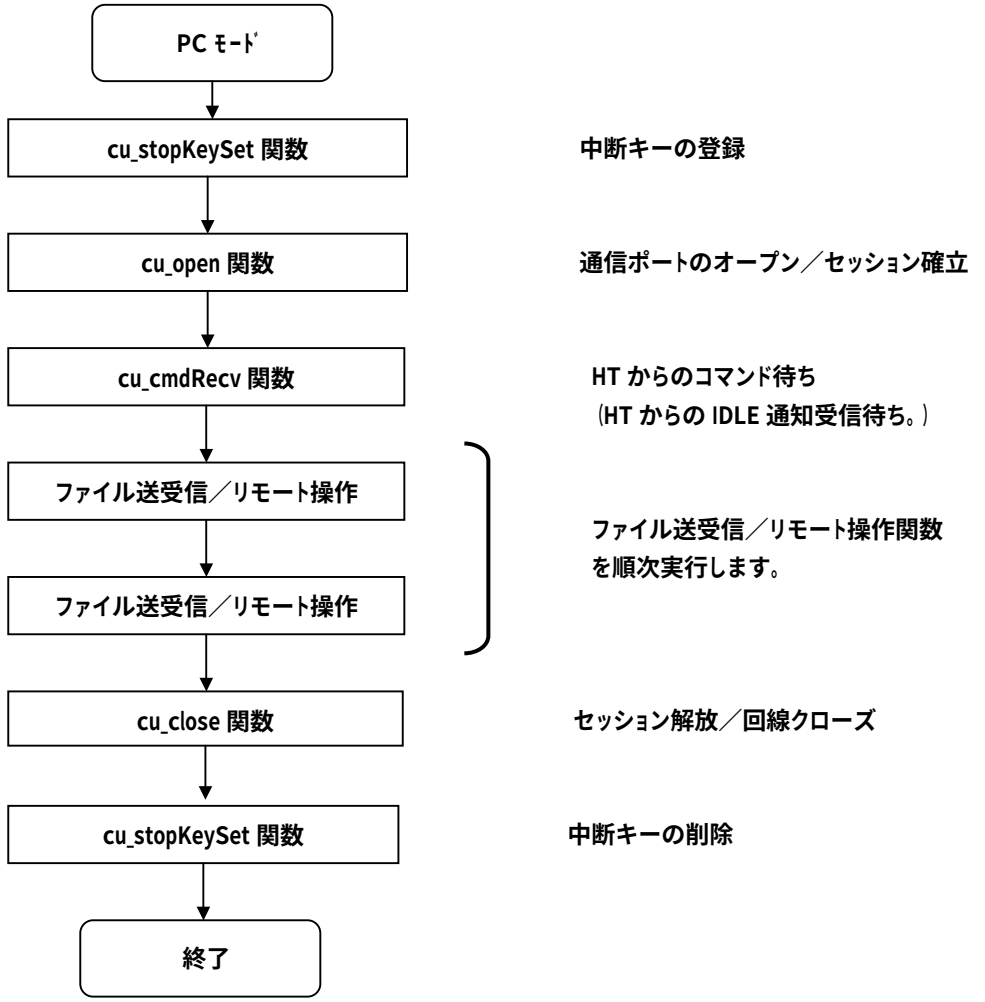
中断キーの削除

PCコマンド送信による通信

[HTモード基本フロー]



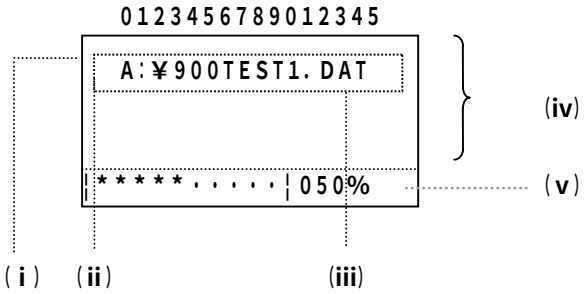
[PCモード基本フロー]



(2) ファイル送受信関数

相手局とのファイル転送 (送信、追加、受信) を行うための関数です。
送信権局はファイル送信、追加および受信コマンドを使用して、相手局とのファイル転送を実現します。
被送信権局は、IDLE 状態 (HT モード)、PC モードコマンド待ち状態 (PC モード) にて、相手からのコマンドを受け付けます。
尚、ファイル転送時、HT の画面上に進捗グラフを表示することが可能です。
グラフ表示フォーマットは次の通りです

(16ドットフォント　8＊16での表示例)



(i) ファイル名表示先頭行(graphPos)
　　転送ファイル名を表示する行の先頭を指定します。
(ii) ファイル名表示先頭カラム(graphCol)
　　転送ファイル名を表示する桁の先頭を指定します。
(iii) ファイル名表示フラグ(graphName)
　　転送ファイル名を全パス表示するか、ファイル名のみ表示するかを指定します。
(iv) ファイル名表示行数(graphLine)
　　転送ファイルパス名を表示するエリア行数を指定します。
　　ファイル名表示フラグが、全パス表示の場合のみ有効です。
　　尚、パス名がファイル名エリア行を越える場合はファイル名のみ表示します。
(v) 進捗グラフおよびパーセンテージ表示。
　　ファイル名表示エリアの次の行に表示されます。(1行固定)
　　10%単位で“,”が“＊”に変わります。

尚、グラフ表示モード(graphMode)は以下の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]	[送信先ﾃﾞｨﾚｸﾄﾘ名]
“A:\SEND\AAA.DAT”	”B:\RECV\”

(送信側ﾃﾞｨﾚｸﾄﾘ構成)

```
A:\----SEND\----SUB1\------AAA.DAT       B:\---RECV\---SUB1\-----AAA.DAT
          ¦-------SUB2\------BBB.DAT                ¦------AAA.DAT
          ¦-------AAA.DAT
          ¦-------BBB.DAT
```

(c)ワイルドカードの使用
送信ファイル名にはワイルドカード(*,?)を使用することができ来ます。

②ファイル追加 (cu_fileAdd)

HT側ファイルを相手局側ファイルにアペンドすることがでできます。
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。

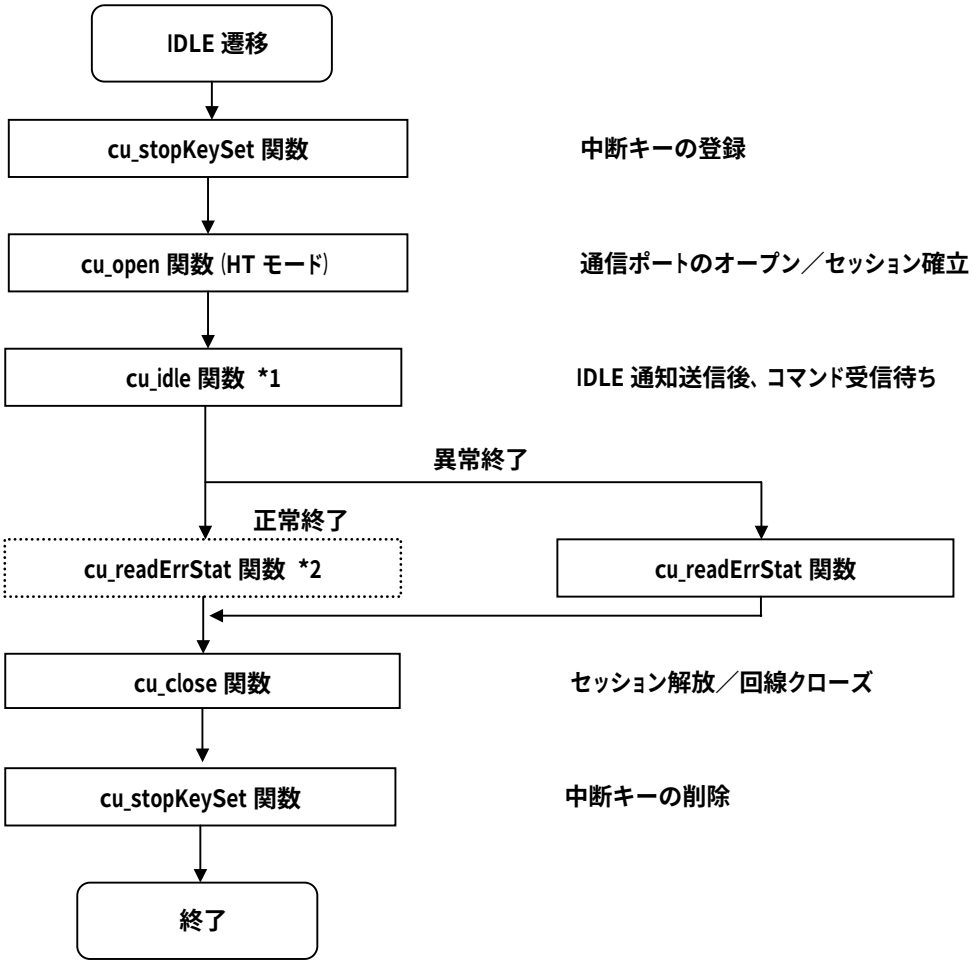
(b)再帰呼び出し指定オプション
受信ファイルパス名で指定されたディレクトリ傘下の全てのファイルが転送対象となります。
指定ディレクトリ傘下にサブディレクトリが存在した場合はサブディレクトリ名を付加してファイルの受信を行います。

(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］

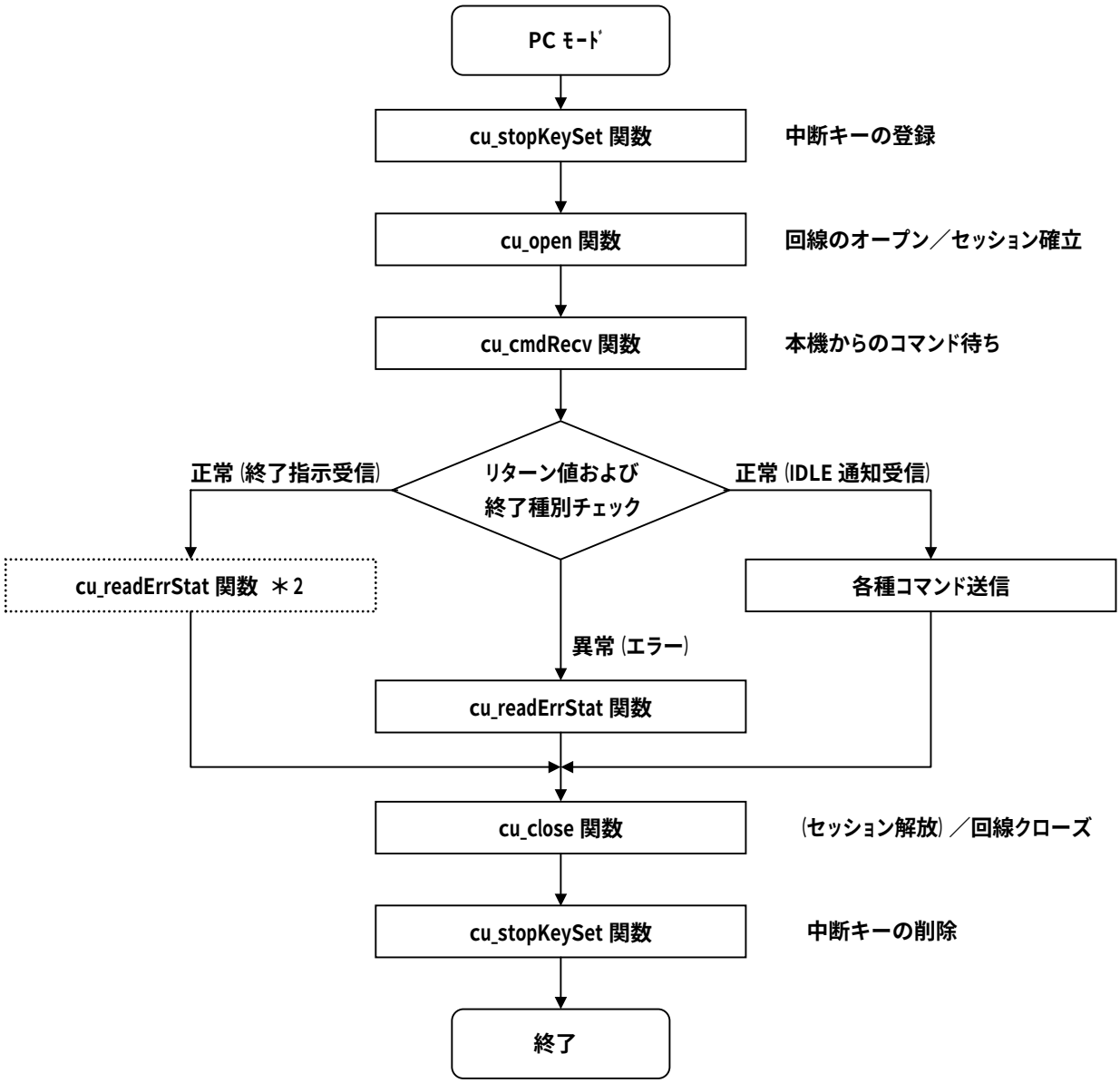


＊1･･･相手局からの終了指示コマンド受信またはエラー発生まで、受信コマンドを順次実行します。
＊2･･･必要に応じて相手局からの終了指示コマンド詳細情報（ﾌｫｰﾏｯﾄ指示、ﾘｾｯﾄ指示等）の取得が可能です。

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。

8.4.3.　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。

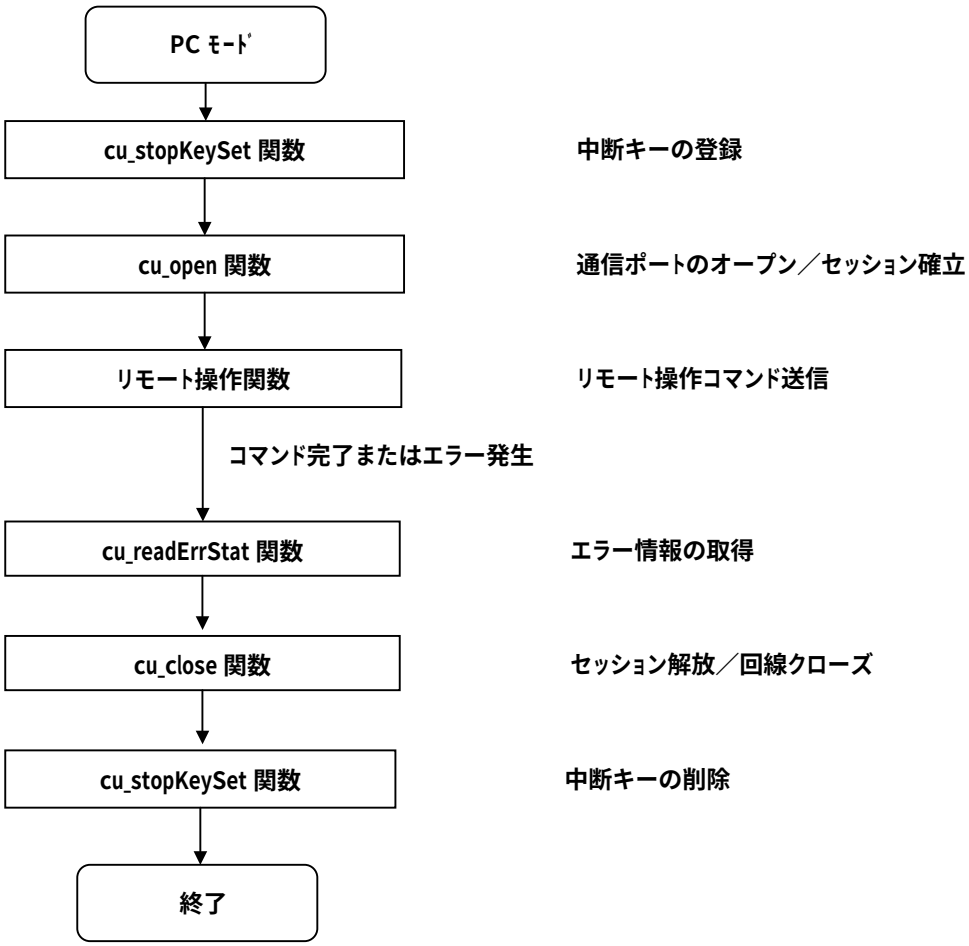
相手局ＨＴはメッセージを画面に表示します。

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］



8.4.4.　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。

8.5. DT500 プロトコル機能

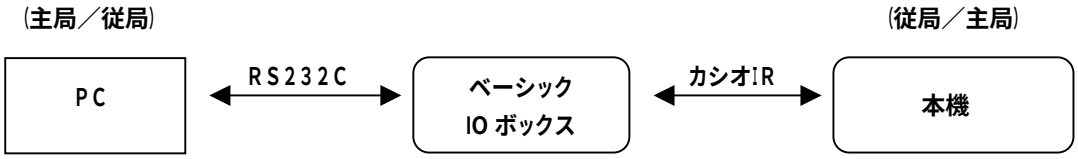
8.5.1. 通信仕様

(1) 通信構成

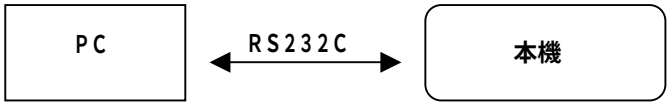
本プロトコルは DT500 プロトコルをサポートした機器（PC）との1対1のファイル転送を想定しています。
DT500 プロトコルでは、ファイル送信側を主局、受信側を従局と呼び、ファイルの転送方向により主局・従局は入れ替わります。
本プロトコルでは、以下の構成でのファイル転送機能を提供します。

①IO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・マスタIOボックスは使用できません）



②直結ケーブルでのシリアル通信



(2) 通信パラメータ

	関数	システムメニュー用通信機能
共通ﾊﾟﾗﾒｰﾀ		
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／IrDA Ver1.0)		
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
COM1 (10ﾋﾟﾝ)		
通信速度	選択可 (1200-115200bps)	ｼｽﾃﾑ環境設定 (データ管理部) より取得
データ長	選択可 (7ﾋﾞｯﾄ/8ﾋﾞｯﾄ)	ｼｽﾃﾑ環境設定 (データ管理部) より取得
パリティビット	選択可 (奇数/偶数/なし)	ｼｽﾃﾑ環境設定 (データ管理部) より取得
ストップビット	選択可 (1ﾋﾞｯﾄ/2ﾋﾞｯﾄ)	ｼｽﾃﾑ環境設定 (データ管理部) より取得
RS/CS ﾌﾛｰ制御	なし	

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すが、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。

8.5.2.　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およびシリアルインタフェースです。
- 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。

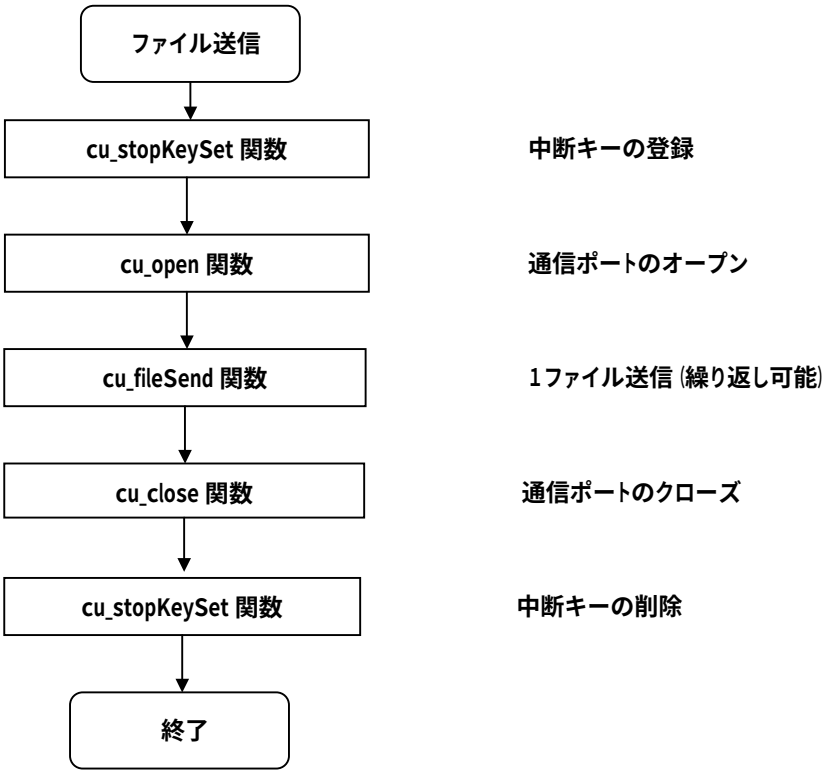
尚、ファイル転送時、HTの画面上に進捗グラフを表示することができます。
グラフ表示フォーマットは次の通りです。

```
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)

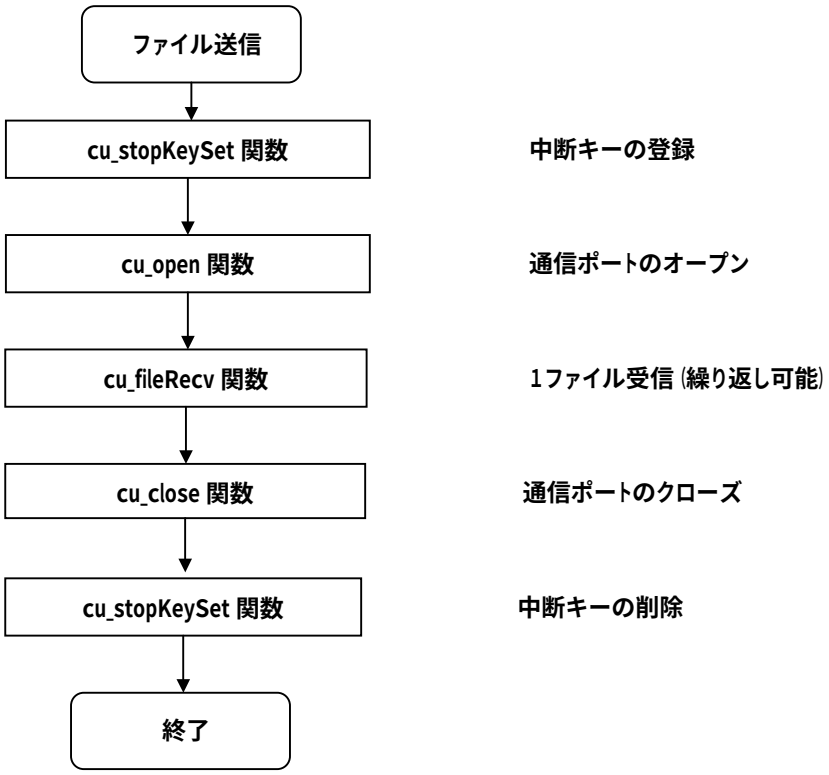
①ファイル送信

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。

8.5.3.　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なわずそのまま格納します。
- フィールド末尾のスペースは削除しません。
- システムメニュー上からのバイナリ転送による送信はできません。

②ユーザプログラムファイル

- 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なわず、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。

8.6. 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。

8.6.1. 共通ファンクション

機能	中断キーの設定	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER ercd = cu_stopKeySet(UB keyId)

[パラメータ]
　　UB　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

[リターンパラメータ]
　　ER　ercd　　:リターンコード

[リターンコード]
　　E_OK　　:正常終了
　　E_PRM　　:パラメータエラー

≪備考≫

機能	転送ドライブ指定	関数名	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)

≪Ｃ言語インタフェース≫

[コーリングシーケンス]
　　ER ercd = cu_setDrive(UB drive)

[パラメータ]
　　UB　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

[リターンパラメータ]
　　ER　ercd　　:リターンコード

[リターンコード]
　　E_OK　　:正常終了
　　E_PRM　　:パラメータエラー

≪備考≫

8.6.2.　マルチドロッププロトコル

機能	回線オープン	関数名	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。

≪C 言語インタフェース≫

[コーリングシーケンス]
ER ercd = cu_open(H comNo， UB connectMode， struct sys_tty *param)

[パラメータ]
H　comNo　　：通信ポート
　　COM0　　：カシオ IR インタフェース
　　COM1　　：シリアルインタフェース
UB connectMode　　：接続モード
　　CU_CNCT_MULT　　：マルチドロップ接続
struct sys_tty *param　　：通信パラメータのポインタ

[ストラクト構造]
```
struct sys_tty     {
                    W   speed:          :B_1200～B_115200   /* 転送速度        */
                    W   length;         :CHAR_8             /* ﾃﾞｰﾀ長固定      */
                    W   parity;         :PARI_NON           /* ﾊﾟﾘﾃｨﾋﾞｯﾄ       */
                                        :PARI_ODD
                                        :PARI_EVN
                    W   stop_bit;       :STOP_1             /* ｽﾄｯﾌﾟﾋﾞｯﾄ       */
                                        :STOP_2
                   } *param;
```

[リターンパラメータ]
ER　ercd　　：リターンコード

[リターンコード]
E_OK　　：正常終了
E_NG　　：異常終了
E_PRM　　：パラメータエラー

≪備考≫

機能	ファイル送信	関数名	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。

C言語インタフェース

[コーリングシーケンス]
```
ER ercd = cu_fileSend(H comNo,  UB priority,  UB fileKind,  UH filecount,
                      CU_FILE_INFO_FORM *fileInfo,  UB graphFlag,  UB graphPos)
```

[パラメータ]
```
H   comNo                    :通信ポート
                                 COM0    :カシオ IR インタフェース
                                 COM1    :シリアルインタフェース
UB priority                  :優先順位 (0～255)
UB 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
UH fileCount                 :送信ファイルの数 (1～65535)
CU_FILE_INFO_FORM *fileInfo  :転送ファイル情報の配列 (fileCount 分) の先頭アドレス
UB 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
UB graphPos                  :グラフ表示行 (0～7, graphFlag が CU_GRAPH_OFF 時は参照しません)
```

[リターンパラメータ]
```
ER   ercd                    :リターンコード
```

[ストラクト構造]
```
typedef struct{
        UB fileName[11];   :転送ファイル名格納領域
                              例) CONFIG. HTS の場合
```

C	O	N	F	I	G			H	T	S

```
        UB stat;           :転送結果格納領域 (次ページを参照して下さい)
      }CU_FILE_INFO_FORM;
```

[リターンコード]
```
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
```

≪備考≫
・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

機能	ファイル送信情報設定	関数名	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。

≪C 言語インタフェース≫

[コーリングシーケンス]
ER ercd = cu_fileSendSet(H comNo， UB priority， UB fileKind， UH fileCount, W totalTransSize)

[パラメータ]
H　comNo　　:通信ポート
　COM0　　:カシオ IR インタフェース
　COM1　　:シリアルインタフェース
UB priority　　:優先順位　　(0～255)
UB 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
UH fileCount　　:送信ファイルの数　(1～65535)
W　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

[リターンパラメータ]
ER　ercd　　:リターンコード

[リターンコード]
E_OK　　:正常終了
E_NG　　:異常終了
E_PRM　　:パラメータエラー

≪備考≫
・当関数の使用前に cu_open 関数を実行する必要があります。

機能	1ファイル送信	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]
ER ercd = cu_fileSend1(H comNo， CU_FILE_INFO_FORM *fileInfo， UB graphFlag， UB graphPos)

[パラメータ]
H　comNo　：通信ポート
　COM0　：カシオ IR インタフェース
　COM1　：シリアルインタフェース
CU_FILE_INFO_FORM *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)
UB 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
UB graphPos　：グラフ表示行（0～7，graphFlag が CU_GRAPH_OFF 時は参照しません）

[リターンパラメータ]
ER　ercd　：リターンコード

[リターンコード]
E_OK　：正常終了
E_NG　：異常終了
E_PRM　：パラメータエラー

≪備考≫
・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。

機能	ファイル受信	関数名	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。

≪C 言語インタフェース≫

[コーリングシーケンス]

```
ER ercd = cu_fileRecv(H comNo,  UB priority,  UB fileKind,  UH, *fileCount,
            CU_FILE_INFO_FORM *fileInfo,  UB graphFlag,  UB graphPos)
```

[パラメータ]

H　comNo	:通信ポート
	COM0　:カシオ IR インタフェース
	COM1　:シリアルインタフェース
UB 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)
UB 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。
UH *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
CU_FILE_INFO_FORM *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)
UB 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
UB graphPos	:グラフ表示行 (0～7, graphFlag が CU_GRAPH_OFF 時は参照しません)

[リターンパラメータ]

ER　ercd	:リターンコード

[リターンコード]

E_OK	:正常終了
E_NG	:異常終了
E_PRM	:パラメータエラー

≪備考≫
・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。

機能	画面表示メッセージ送信	関数名	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。

≪C 言語インタフェース≫

[コーリングシーケンス]
　　ER ercd = cu_msgSend(H comNo,　UB *data)

[パラメータ]
　　H　comNo　　:通信ポート
　　　　COM0　　:カシオ IR インタフェース
　　　　COM1　　:シリアルインタフェース
　　UB *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）

[リターンパラメータ]
　　ER　ercd　　:リターンコード

[リターンコード]
　　E_OK　　:正常終了
　　E_NG　　:異常終了
　　E_PRM　　:パラメータエラー

≪備考≫
・当関数を使用する前に必ず cu_fileSend1 関数を使用して下さい。

機能	通信中断	関数名	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。

≪C 言語インタフェース≫

[コーリングシーケンス]
　　ER ercd = cu_end(H comNo)

[パラメータ]
　　H　comNo　　:通信ポート
　　　　COM0　　:カシオ IR インタフェース
　　　　COM1　　:シリアルインタフェース

[リターンパラメータ]
　　ER　ercd　　: リターンコード

[リターンコード]
　　E_OK　　:正常終了
　　E_NG　　:異常終了
　　E_PRM　　:パラメータエラー

≪備考≫
・当関数を使用する前に必ず cu_fileSend1 関数を使用して下さい。

機能	回線クローズ	関数名	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。

≪C 言語インタフェース≫

[コーリングシーケンス]
　　ER ercd = cu_close(H comNo)

[パラメータ]
　　H　comNo　　:通信ポート
　　　　COM0　　:カシオ IR インタフェース
　　　　COM1　　:シリアルインタフェース

[リターンパラメータ]
　　ER　ercd　　:リターンコード

[リターンコード]
　　E_OK　　:正常終了
　　E_NG　　:異常終了
　　E_PRM　　:パラメータエラー

≪備考≫

機能	エラー詳細情報取得	関数名	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。

≪C 言語インタフェース≫

[コーリングシーケンス]
　　ER ercd = cu_readErrStat(H comNo,　UW *cuStat,　UW *biosStat)

[パラメータ]
　　H　comNo　　：通信ポート
　　　　COM0　　：カシオ IR インタフェース
　　　　COM1　　：シリアルインタフェース
　　UW *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)
　　UW *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)

[リターンパラメータ]
　　ER　ercd　　：リターンコード

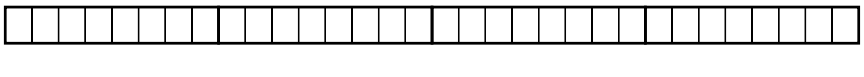
[リターンコード]
　　E_OK　　：正常終了
　　E_PRM　　：パラメータエラー

≪備考≫

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。



未使用

エラー状態
0～255にてエラーの状態を表します。
エラー状態一覧を参照のし下さい。

コマンド種別
0～255にてエラー発生時のコマンド番号を表します。
コマンド種別一覧を参照して下さい。

動作モード
0、1にてエラー発生時の動作モードを表します。
(コマンド種別が CU_CMD_NON の場合には無意味)。
動作モード一覧を参照して下さい。

フェーズ
0、1にてエラー発生時のフェーズを表します。
フェーズ一覧を参照して下さい。

フェーズ一覧

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

動作モード一覧

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

エラー詳細情報 (2)

コマンド種別一覧

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

エラー状態一覧

値	シンボル	意味
00	CU_ERR_NON	エラー発生なし
01	CU_ERR_PHASE	該当関数の使用フェーズ誤り (通信は継続)
02	CU_ERR_FILE	ファイル I／Oエラー
03	CU_ERR_CLOCK	CPU のクロック切り替えエラー
04	CU_ERR_DL	データリンクエラー
05	CU_ERR_DL_RJ	データリンクエラー (主局拒否)
06	CU_ERR_TIMER	タイマー使用エラー
07	CU_ERR_NAK	ＮＡＫコード受信でのリトライオーバー
08	CU_ERR_BCC	ＢＣＣエラーでのリトライオーバー
09	CU_ERR_SEQ	コマンドシーケンス番号誤り
0A	CU_ERR_CONT	コマンド内容異常
0B	CU_ERR_EOT	ＥＯＴコード受信による中断
0C	CU_ERR_RCVTO	受信タイムアウト
0D	CU_ERR_CRP	コマンド／レスポンスのフェーズエラー
0E	CU_ERR_CMD	期待されないコマンドの受信
0F	CU_ERR_FILE_NO	受信ファイル数エラー
10	CU_ERR_SEND	送信エラー
11	CU_ERR_STOP	中断キー押下
12	CU_ERR_NODATA	受信データなし (相手局不在)
13	CU_ERR_DL_OVER	マルチドロップ再データリンクオーバー
14	CU_ERR_UNKNOWN_CMD	未知のコマンドの受信
15	CU_ERR_PRM	通信ＵＴ関数パラメータエラー
16	CU_ERR_RECV	受信エラー
17	CU_ERR_AP	受信ＡＰのエラー (メモリアドレス等)
18	(未使用)	(未使用)
:	:	:
40	CU_ERR_NOT_SUPPORT	未サポート
41	(未使用)	(未使用)
:	:	:
FE	CU_ERR_ABNORMAL	通信ＵＴロジックエラー
FF	CU_ERR_OTHER	上記以外のエラー 通信関数部システムエラーです

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

エラー詳細情報 (3)

2. 通信関数部システムエラー

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

3. システムメニューエラー

システムメニューでのみ発生し得るエラー状態を以下に示します。

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

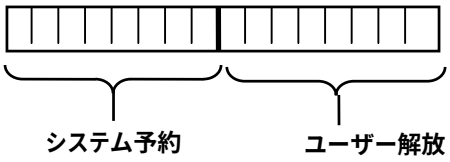
機能	データリンク拒否情報取得	関数名	cu_readDlRjInfo

[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。

≪C 言語インタフェース≫

[コーリングシーケンス]
　　ER ercd = cu_readDlRjInfo(H comNo,　UH *rjInfo)

[パラメータ]
　　H　comNo　　:通信ポート
　　　　COM0　　:カシオ IR インタフェース
　　　　COM1　　:シリアルインタフェース
　　UH *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) ＞



[リターンパラメータ]
　　ER　ercd　　:リターンコード

[リターンコード]
　　E_OK　　:正常終了
　　E_PRM　　:パラメータエラー

≪備考≫
・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。

8.6.3.　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。

機能	回線オープン (初期化)	関数名	cu_open

[通信ユーティリティ ： FLINK プロトコル]
通信ポートの初期化およびセッションの確立を行ないます。
セッション確立までは、タイムアウト時間まで待ちます。
相手局ｼｽﾃﾑ情報の取得を行ないます。

≪C 言語インタフェース≫

[コーリングシーケンス]
```
ER ercd = cu_open(H comNo， H irSpeed， CU_RSPRM *rsPrm， H mode)
```

[パラメータ]
```
H　comNo          :通信ポート
                       COM0          :カシオ IR インタフェース
                       COM1          :シリアルインタフェース
H　irSpeed        :赤外通信最高速度 (comNo が COM1 使用時は参照しません) CU_B2400～CU_B115K
CU_RSPRM *rsPrm   :10 ﾋﾟﾝ通信ﾊﾟﾗﾒｰﾀ (comNo が COM0 使用時は参照しません)
H　mode           :局モード
                       CU_MODE_HT    ： HT ﾓｰﾄﾞ
                       CU_MODE_PC    ： PC ﾓｰﾄﾞ (擬似ＰＣとして動作を行います)
```

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

[リターンパラメータ]
```
ER　ercd          :リターンコード
```

[リターンコード]
```
E_OK              :正常終了
E_NG              :異常終了
E_PRM             :パラメータエラー
```

≪備考≫
・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。

機能	ファイル送信	関数名	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。

≪C 言語インタフェース≫

[コーリングシーケンス]
ER ercd = cu_fileSend(H comNo, H mode, B *fName, B *dir, H protect,　CU_GRAPHSET *graphSet)

[パラメータ]
H　comNo　:通信ポート
　COM0　:カシオ IR インタフェース
　COM1　:シリアルインタフェース
H　mode　:転送ﾓｰﾄﾞ(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
B　*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)
B　*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)
H　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
CU_GRAPHSET *graphSet　： グラフ表示情報

[ストラクト構造]
typedef struct{
　H graphMode　:グラフ表示モード
　　CU_GRAPH_ON_1　:転送全体を 100%として表示
　　CU_GRAPH_ON_2　:1ファイルを 100%として表示
　　CU_GRAPH_OFF　:表示しない
　(CU_GRAPH_OFF 設定時は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;

[リターンパラメータ]
ER　ercd　:リターンコード

[リターンコード]
E_OK　:正常終了
E_NG　:異常終了
E_PRM　:パラメータエラー

≪備考≫

機能	ファイル追加	関数名	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。

≪C 言語インタフェース≫

[コーリングシーケンス]
ER ercd = cu_fileAdd(H comNo, B *sfName, B *rfName, CU_GRAPHSET *graphSet)

[パラメータ]

H comNo	:通信ポート	
	COM0	:カシオ IR インタフェース
	COM1	:シリアルインタフェース
B *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)
B *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)
CU_GRAPHSET *graphSet	:グラフ表示情報	(cu_fileSend 関数参照)

[リターンパラメータ]

ER ercd	:リターンコード

[リターンコード]

E_OK	:正常終了
E_NG	:異常終了
E_PRM	:パラメータエラー

≪備考≫

機能	ファイル受信	関数名	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。

≪C 言語インタフェース≫

[コーリングシーケンス]
ER ercd = cu_fileRecv(H comNo, H mode， B *fName， B *dir， H protect， CU_GRAPHSET *graphSet)

[パラメータ]

H　comNo	：通信ポート	
	COM0	：カシオ IR インタフェース
	COM1	：シリアルインタフェース
H　mode	：転送ﾓｰﾄﾞ (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
B　*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)	
B　*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)	
H　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
CU_GRAPHSET *graphSet	：グラフ表示情報 (cu_fileSend 関数参照)	

[リターンパラメータ]

ER　ercd	：リターンコード

[リターンコード]

E_OK	：正常終了
E_NG	：異常終了
E_PRM	：パラメータエラー

≪備考≫

機能	回線クローズ	関数名	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。

≪C 言語インタフェース≫

[コーリングシーケンス]
　　ER ercd = cu_close(H comNo,　H endKind)

[パラメータ]
　　H　comNo　　:通信ポート
　　　　COM0　　:カシオ IR インタフェース
　　　　COM1　　:シリアルインタフェース
　　H　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

[リターンパラメータ]
　　ER　ercd　　:リターンコード

[リターンコード]
　　E_OK　　:正常終了
　　E_NG　　:異常終了
　　E_PRM　　:パラメータエラー

≪備考≫

機能	エラー情報の取得	関数名	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。

≪C 言語インタフェース≫

[コーリングシーケンス]
　　ER ercd = cu_readErrStat(H comNo,　CU_ERRINFO *errInfo)

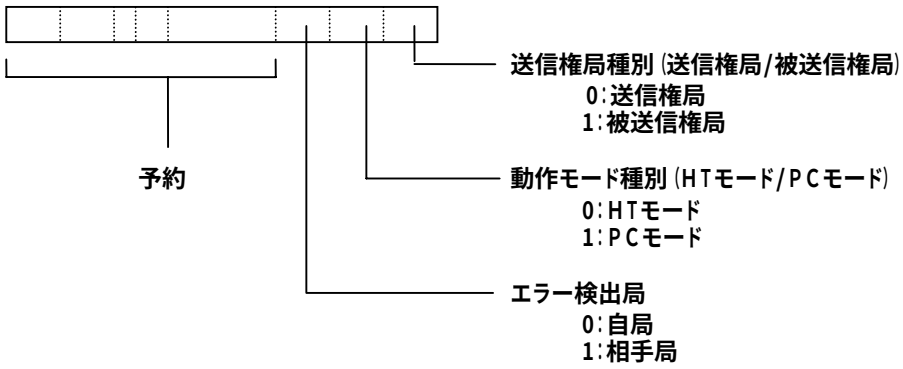
[パラメータ]
　　H　comNo　　：通信ポート
　　　　COM0　　：カシオ IR インタフェース
　　　　COM1　　：シリアルインタフェース

　　CU_ERRINFO *errInfo　　：エラー情報設定エリア

[リターンパラメータ]
　　ER　ercd　　：リターンコード

[ストラクト構造]
　　typedef struct{
　　　　UB kind　　：エラー種別　　(下記参照)
　　　　UB command　　：コマンド種別　　(次ページ参照)
　　　　UB category　　：カテゴリ　　(次ページ参照)
　　　　UB detail　　：エラー詳細　　(次ページ参照)
　　　　UW biosStat　　：システム領域エラーエリア

　　　　(comNo が COM0 時は IrDA 部関数、COM1 時は通信関数のエラーが設定されます)
　　} CU_ERRINFO;



s
[リターンコード]
　　E_OK　　：正常終了
　　E_PRM　　：パラメータエラー

≪備考≫

エラー情報の取得　コマンド種別･エラー状態　一覧

コマンド種別		
値	シンボル	意味
00	CU_CMD_NON	該当コマンドなし
01	CU_CMD_FSEND_TINFO	ファイル転送情報コマンド
02	CU_CMD_FSEND_FINFO	ファイル情報コマンド
03	CU_CMD_FRECV_TREQ	ファイル受信要求コマンド
04	CU_CMD_FADD	ファイル追加コマンド
05	CU_CMD_FDATA	ファイルデータコマンド
06	CU_CMD_FDEL	ファイル削除コマンド
07	CU_CMD_FMOV	ファイル移動コマンド
08	CU_CMD_MAKEDIR	ディレクトリ作成コマンド
09	CU_CMD_TIME_SET	日付時刻設定コマンド
0A	CU_CMD_TIME_GET	日付時刻取得コマンド
0B	CU_CMD_DISP	メッセージ表示コマンド
0C	CU_CMD_BEEP	ブザー鳴動コマンド
0D	CU_CMD_FINFO_GET	ファイル情報取得コマンド
0E	CU_CMD_FINFO_SET	ファイル情報設定コマンド
0F	CU_CMD_DINFO_GET	ディスク情報取得コマンド
10	CU_CMD_SYS_GET	システム情報取得コマンド
11	CU_CMD_IDLE	IDLE通知コマンド
12	CU_CMD_END	終了指示コマンド

カテゴリコード･エラー詳細コード カテゴリとエラー詳細コードの組み合わせによりエラー状態を表す		
値		意味
カテゴリ	詳細	
正常終了状態		
00	00	正常終了
DC～F5	00	フォーマット指示コマンド(A～Z)
F6	00	電源 OFF 終了通知
F7	00	リセット指定終了通知
F8	00	中断キーによる終了通知
F9～FF	－	予約領域
プロトコルエラー		
01	00	受信フレームファンクションコード未定義エラー
	01	受信フレームサブファンクションコード未定義エラー
	03	受信フレームチェックサムエラー
	04	シーケンスエラー
	05	シーケンス番号エラー
	07	受信フレーム内情報パラメータエラー
	08	受信タイムアウト
	10	コマンドレングスエラー
ファイルエラー[プロトコル論理]		
04	00	リードオンリィファイルアクセスエラー

ユーティリティエラー			
10	00	回線オープンエラー	・回線がオープンされていない ・オープン時にエラーが発生していないか確認
	01	使用関数フェーズエラー	・関数の使い方に誤りがある ・動作モード／送信権局ﾓｰﾄﾞを確認
	02	使用関数パラメータエラー	・関数パラメータに誤りがある ・指定パラメータを確認
	03	指定ファイル未検出エラー	・指定されたファイルが存在しない ・指定ファイルを確認
	04	相手局未検出	・セッション確立待ちタイムアウト ・通信設定、回線経路を確認
	05	システム日付設定エラー	・指定日付を確認
	06	システム時刻設定エラー	・指定時刻を確認
	07	タイマー使用エラー	・タイマーが登録できなかった ・APで使用しているタイマ数を確認
	08	CPUクロック切替えエラー	・CPU切替え禁止状態でないか確認
	09	致命的エラー	・IrDA、通信関数からのエラー ・ローバッテリの発生等が考えられる
	0A	通信中回線断エラー	・通信中に回線が切断された ・回線経路を確認
	0B	ドライブ容量不足	・指定ドライブの容量が足りない
ファイルエラー［ファイル関数］			
11	00	クリエートエラー	
	01	オープンエラー	
	02	リードエラー	
	03	ライトエラー	
	04	シークエラー	
	05	ファイル削除エラー	
	06	ディレクトリ削除エラー	
	07	ファイル名変更移動エラー	
	08	タイムスタンプ設定エラー	
	09	タイムスタンプ取得エラー	
	0A	ファイル属性設定エラー	
	0B	ファイル属性取得エラー	
	0C	ディレクトリ作成エラー	
	0D	ファイルサイズ変更エラー	
システムメニュー通信エラー			
20	00	フォーマット実行エラー	・フォーマット中にエラー発生 ・再フォーマットする
	01	環境設定ファイル未存在エラー	・CONFIG. HTS ファイルが存在しない
	02	環境設定ファイル更新エラー	・CONFIG. HTS 異常 ・ファイルレイアウトを確認
	03	相手局不正	・想定している相手局ではない ・相手局を確認
	04	指定ドライブなし	・子機作成時、送信側指定ﾄﾞﾗｲﾌﾞ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	

機能	IDLE遷移	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]
ER ercd = cu_idle(H comNo， B *script， CU_GRAPHSET *graphSet)

[パラメータ]
H　comNo　：通信ポート
COM0　：カシオ IR インタフェース
COM1　：シリアルインタフェース
B　*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 ﾊﾞｲﾄ]
(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)
CU_GRAPHSET *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)

[リターンパラメータ]
ER　ercd　：リターンコード

[リターンコード]
E_OK　：正常終了
E_NG　：異常終了
E_PRM　：パラメータエラー

≪備考≫

機能	コマンド受信待ち	関数名	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。

≪C 言語インタフェース≫

[コーリングシーケンス]
　　ER ercd = cu_cmdRecv(H comNo,　H *endKind,　B *script,　CU_GRAPHSET *graphSet)

[パラメータ]
　[入力]
　　H　comNo　　:通信ポート
　　　　COM0　　:カシオ IR インタフェース
　　　　COM1　　:シリアルインタフェース
　　CU_GRAPHSET *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]
　　H　*endKind　　:終了種別ﾌﾗｸﾞ設定ｴﾘｱ (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
　　B　*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 ﾊﾞｲﾄ]

[リターンパラメータ]
　　ER　ercd　　:リターンコード

[リターンコード]
　　E_OK　　:正常終了
　　E_NG　　:異常終了
　　E_PRM　　:パラメータエラー

≪備考≫

機能	ファイル削除	関数名	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／ﾃﾞｨﾚｸﾄﾘを削除します。複数ファイル／ﾃﾞ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。

≪C 言語インタフェース≫

[コーリングシーケンス]
　　ER ercd = cu_fileDelete(H comNo,　B *fName)

[パラメータ]
　　H　comNo　　:通信ポート
　　　　COM0　　:カシオ IR インタフェース
　　　　COM1　　:シリアルインタフェース
　　B　*fName　　:削除するファイル／ﾃﾞｨﾚｸﾄﾘ名エリア (複数指定およびワイルドカード可)

[リターンパラメータ]
　　ER　ercd　　:リターンコード

[リターンコード]
　　E_OK　　:正常終了
　　E_NG　　:異常終了
　　E_PRM　　:パラメータエラー

≪備考≫

機能	ファイル移動	関数名	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。
移動元と移動先のﾄﾞﾗｲﾌﾞ名が異なる場合はエラーになります。

≪C 言語インタフェース≫

[コーリングシーケンス]
　　ER ercd = cu_fileMove(H comNo,　B *sfName,　B *dfName)

[パラメータ]
　　H　comNo　　：通信ポート
　　　　COM0　　：カシオ IR インタフェース
　　　　COM1　　：シリアルインタフェース
　　B　*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)
　　B　*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)

[リターンパラメータ]
　　ER　ercd　　：リターンコード

[リターンコード]
　　E_OK　　：正常終了
　　E_NG　　：異常終了
　　E_PRM　　：パラメータエラー

≪備考≫

機能	ディレクトリ作成	関数名	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。

≪C 言語インタフェース≫

[コーリングシーケンス]
　　ER ercd = cu_makeDir(H comNo， B *mDir， CU_DATETIME *datetime， B atr)

[パラメータ]
　　H　comNo　　：通信ポート
　　　　COM0　　：カシオ IR インタフェース
　　　　COM1　　：シリアルインタフェース
　　B　*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)
　　CU_DATETIME *datetime　　：日付時刻エリア (下記参照)
　　B　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)

[リターンパラメータ]
　　ER　ercd　　：リターンコード

[ストラクト構造]
```
typedef struct{
    UB   day;          /* 日(1-31)        */
    UB   month;        /* 月(1-12)        */
    UH   year;         /* 年(1980-2079)   */
    UB   sec;          /* 秒(0-59)        */
    UB   min;          /* 分(0-59)        */
    UB   hour;         /* 時(0-23)        */
} CU_DATETIME;
```
　　*日付時刻を指定しない場合は year に FFFFH を　day、month、sec、min、hour に FFH を設定して下さい

[リターンコード]
　　E_OK　　：正常終了
　　E_NG　　：異常終了
　　E_PRM　　：パラメータエラー

≪備考≫

機能	ファイル情報の取得	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER ercd = cu_getFileInfo(H comNo， H mode， B *fName， CU_FINFO *fInfo)

[パラメータ]
　　H　comNo　　:通信ポート
　　　　COM0　　:カシオ IR インタフェース
　　　　COM1　　:シリアルインタフェース
　　H　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)
　　B　*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。)
　　CU_FINFO *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。

[リターンパラメータ]
　　ER　ercd　　:リターンコード

[ストラクト構造]
　　typedef struct{
　　　　B name[256]　　:検索されたファイル名 (ﾌﾙﾊﾟｽ名)
　　　　CU_DATETIME datetime;　　:日付時刻エリア (cu_dateTime 関数参照)
　　　　W size;　　:サイズ
　　　　B atr;　　:属性(OR 指定により設定される)
　　　　　　_A_NORMAL　　:通常ファイル(R/W)
　　　　　　_A_HIDDEN　　:不可視ファイル
　　　　　　_A_RDONLY　　:読出し専用ファイル
　　　　　　_A_SYSTEM　　:システムファイル
　　　　　　_A_SUBDIR　　:ディレクトリ
　　　　　　_A_ARCH　　:アーカイブ
　　} CU_FINFO;

[リターンコード]
　　E_OK　　:正常終了
　　E_NG　　:異常終了
　　E_PRM　　:パラメータエラー

≪備考≫

機能	ファイル情報の更新	関数名	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・ｻｲｽﾞ) 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。

≪C 言語インタフェース≫

[コーリングシーケンス]
　　ER ercd = cu_setFileInfo(H comNo,　CU_FINFO *finfo)

[パラメータ]
　　H　comNo　　：通信ポート
　　　　COM0　　：カシオ IR インタフェース
　　　　COM1　　：シリアルインタフェース
　　CU_FINFO *finfo　　：ファイル情報設定エリア

[リターンパラメータ]
　　ER　ercd　　：リターンコード

[ストラクト構造]
```
typedef struct{
    B name[256]                 ：設定するファイル名 (ﾌﾙﾊﾟｽ名)
                                    (複数指定不可・ワイルドカード指定不可)
    CU_DATETIME datetime;       ：日付時刻エリア (cu_dateTime 関数参照)
                                    (変更しない場合は cu_dateTime 関数と同様)
    UW size;                    ：サイズ (0 指定時は変更しません)
    B atr;                      ：属性　(OR 指定により設定)
                                    _A_NORMAL　：通常ファイル(R/W)
                                    _A_HIDDEN　：不可視ファイル
                                    _A_RDONLY　：読出し専用ファイル
                                    _A_SYSTEM　：システムファイル
                                    _A_SUBDIR　：ディレクトリ
                                    _A_ARCH　　：アーカイブ
} CU_FINFO;
```

[リターンコード]
　　E_OK　　：正常終了
　　E_NG　　：異常終了
　　E_PRM　　：パラメータエラー

≪備考≫

機能	ディスク情報の取得	関数名	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。

≪C 言語インタフェース≫

[コーリングシーケンス]
ER ercd = cu_getDiskInfo(H comNo， B drive， CU_DINFO *dInfo)

[パラメータ]
[入力]
H comNo :通信ポート
COM0 :カシオ IR インタフェース
COM1 :シリアルインタフェース
B drive :ドライブ名エリア 'A'～'Z'の何れか。
CU_DINFO *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)

[リターンパラメータ]
ER ercd :リターンコード

[ストラクト構造]
```
typedef struct{
                    UW size;            /* ディスク容量      */
                    UW freex;           /* ディスク空き容量  */
                    UB status;          /* ディスク状態      */
                                            CU_DINFO_NORMAL   :ﾃﾞｨｽｸあり (ﾌｫｰﾏｯﾄ済)
                                            CU_DINFO_NOFMT    :ﾃﾞｨｽｸあり (未ﾌｫｰﾏｯﾄ)
                                            CU_DINFO_NODISK   :ﾃﾞｨｽｸなし
                } CU_DINFO;
```

[リターンコード]
E_OK :正常終了
E_NG :異常終了
E_PRM :パラメータエラー

≪備考≫

機能	日付時刻の取得および設定	関数名	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。

≪C 言語インタフェース≫

[コーリングシーケンス]
```
ER ercd = cu_dateTime(H comNo,  H mode,  CU_DATETIME *dateTime)
```

[パラメータ]
```
H   comNo                  :通信ポート
                               COM0          :カシオ IR インタフェース
                               COM1          :シリアルインタフェース
H   mode                   :取得／設定フラグ
                               CU_GET_MODE   :取得
                               CU_SET_MODE   :設定
CU_DATETIME *dateTime      :設定日付時刻エリアアドレス
CU_DATETIME *dateTime      :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)
```

[リターンパラメータ]
```
ER  ercd                   :リターンコード
```

[ストラクト構造]
```
typedef struct{
                    UB   day;          /* 日(1～31)        */
                    UB   month;        /* 月(1～12)        */
                    UH   year;         /* 年(1980～2079)   */
                    UB   sec;          /* 秒(0～59)        */
                    UB   min;          /* 分(0～59)        */
                    UB   hour;         /* 時(0～23)        */
                  } CU_DATETIME;
                * 日付のみの設定の場合は sec, min, hour に全て 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。
```

[リターンコード]
```
E_OK                       :正常終了
E_NG                       :異常終了
E_PRM                      :パラメータエラー
```

≪備考≫

機能	システム情報の取得	関数名	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ﾌﾟﾝのｾｯｼｮﾝ時に既に取得しているため、通信は行わず、情報のみを返します。

≪C 言語インタフェース≫

[コーリングシーケンス]
　　ER ercd = cu_getSysInfo(H comNo,　CU_SYSINFO *sysInfo)

[パラメータ]
　　H　comNo　　：COM Ｎｏ.
　　　　COM0　　：カシオ IR インタフェース
　　　　COM1　　：シリアルインタフェース
　　CU_SYSINFO *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)

[リターンパラメータ]
　　ER　ercd　　：リターンコード

[ストラクト構造]
　　typedef struct{
　　　UH id;　　：ｾｯｼｮﾝ ID (PC との接続以外は 0 固定)
　　　UB ftpver;　　：FTP ﾊﾞｰｼﾞｮﾝ
　　　UB code[3];　　：機種ｺｰﾄﾞ
　　　　”710”　　：HT
　　　　その他　　：PC または他機種
　　　UB model;　　：ﾓﾃﾞﾙ情報 (００h固定)
　　} CU_SYSINFO;

[リターンコード]

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

≪備考≫

機能	画面表示メッセージの送信	関数名	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。

≪C 言語インタフェース≫

[コーリングシーケンス]
　　ER ercd = cu_msgSend(H comNo,　B *msg)

[パラメータ]
　　H　comNo　　:通信ポート
　　　　COM0　　:カシオ IR インタフェース
　　　　COM1　　:シリアルインタフェース
　　B　*msg　　:表示メッセージ格納エリア (終端は NULL を設定)

[リターンパラメータ]
　　ER ercd:リターンコード

[リターンコード]
　　E_OK　　:正常終了
　　E_NG　　:異常終了
　　E_PRM　　:パラメータエラー

≪備考≫

機能	ブザー鳴動	関数名	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。

≪C 言語インタフェース≫

[コーリングシーケンス]
　　ER ercd = cu_beep(H comNo)

[パラメータ]
　　H　comNo　　:通信ポート
　　　　COM0　　:カシオ IR インタフェース
　　　　COM1　　:シリアルインタフェース

[リターンパラメータ]
　　ER　ercd　　:リターンコード

[リターンコード]
　　E_OK　　:正常終了
　　E_NG　　:異常終了
　　E_PRM　　:パラメータエラー

≪備考≫

機能	IOボックス情報設定	関数名	cu_setIoboxInfo

[通信ユーティリティ ： FLINK プロトコル]
マスターIOボックス(TCP／IP)に対し情報設定を行います。
設定情報は、cu_open 関数実行時にIOボックスに送信され、IOボックスからの応答情報は取得情報アドレスに格納されます。
本関数は、cu_open 関数実行前に実行して下さい。
cu_open 関数実行後は、setLen=0に設定して本関数を実行して下さい。(設定クリア)

≪C言語インタフェース≫

[コーリングシーケンス]
```
ER ercd = cu_setIoboxInfo( H setLen, UB *setInfo, H *getLen, UB *getInfo)
```

[パラメータ]
H　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)
UB　*setInfo　：設定情報アドレス
H　*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
UB　*getInfo　：取得情報アドレス(cu_open 実行時に設定されます)

[リターンパラメータ]
ER　ercd　：リターンコード

[リターンコード]
E_OK　：正常終了
E_PRM　：パラメータエラー

≪備考≫
・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。

機能	FCHK リストファイルの生成	関数名	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)ﾌｧｲﾙのﾊﾟｽ名(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。

≪C 言語インタフェース≫

[コーリングシーケンス]
ER ercd = cu_fchklog_Create(H mode， B *fName， B *dir， B *listDir， H append, CU_GRAPHSET *graphSet)

[パラメータ]
H　mode　:ファイル指定ﾓｰﾄﾞ　(再帰呼び出しを行うかどうかを指定します。)
CU_TRANS_NORMAL　:再帰呼出し無
CU_TRANS_RECURSIVE　:再帰呼出し有
B　*fName　:転送元ファイル名　(複数指定およびワイルとカード指定可)
B　*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)
B　*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)
H　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
CU_GRAPHSET *graphSet　:グラフ表示情報

[リターンパラメータ]
ER　ercd　:リターンコード

[ストラクト構造]
typedef struct{
H graphMode　:グラフ表示モード
CU_GRAPH_ON_1　:リストファイル生成全体を 100%として表示します。
CU_GRAPH_OFF　:表示しません。
(CU_GRAPH_OFF 設定時は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;

[リターンコード]
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

≪備考≫

機能	FCHK リストファイルのチェック	関数名	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。

≪C 言語インタフェース≫

[コーリングシーケンス]
ER ercd = cu_fchklog_Check(B *listDir， CU_GRAPHSET *graphSet)

[パラメータ]
B *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)
CU_GRAPHSET *graphSet　：グラフ表示情報

[リターンパラメータ]
ER　ercd　：リターンコード

[ストラクト構造]
typedef struct{
H graphMode　：グラフ表示ﾓｰﾄﾞ
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　：ファイル名ｴﾘｱ行数　(1～12)
} CU_GRAPHSET;

[リターンコード]
E_OK　：正常終了
FCHK_NG03　：FCHK．LOG が見つかりません
FCHK_NG04　：リストファイルの内容不一致(パス名の不一致)
FCHK_NG05　：リストファイルの内容不一致(ファイルサイズの不一致)
FCHK_NG06　：リストファイルの内容不一致(日付/時刻の不一致)
FCHK_NG07　：リストファイルの内容不一致(全ファイルチェックサムデータの不一致)
FCHK_NG08　：リストファイルの内容不一致(リストチェックサムデータの不一致)
FCHK_NG0B　：リストファイル読込み時エラー
FCHK_NG0D　：パラメータエラー

≪備考≫

8.6.4.　DT500 プロトコル

機能	回線オープン(初期化)	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER ercd = cu_open(H comNo,　struct sys_tty *param)

[パラメータ]
　　H　comNo　　:通信ポート
　　　　COM0　　:カシオIRインタフェース
　　　　COM1　　:シリアルインタフェース
　　struct sys_tty *param　　:通信パラメータエリアアドレス

[リターンパラメータ]
　　ER　ercd　　:リターンコード

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

[リターンコード]
　　E_OK　　:正常終了
　　E_NG　　:異常終了
　　E_PRM　　:パラメータエラー

≪備考≫

機能	ファイル送信	関数名	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。

≪C 言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

```
H   comNo                 :通信ポート
                              COM0          :カシオ IR インタフェース
                              COM1          :シリアルインタフェース
B   *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")
B   fieldCount            :送信ファイルのフィールド数　(1～16)
UB *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    fieldCol
```

10	10	8	4	20

```
CU_DT_OPT *option         :プロトコルオプションエリアアドレス
CU_GRAPHSET *graphSet     :グラフ表示情報エリアアドレス
```

[リターンパラメータ]

```
ER  ercd                  :リターンコード
```

[ストラクト構造]

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

[リターンコード]

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

≪備考≫
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。

機能	ファイル受信	関数名	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。

≪C 言語インタフェース≫

[コーリングシーケンス]
```
ER ercd = cu_fileRecv(H comNo, B *fName, B *fieldCount, UB *fieldCol, CU_DT_OPT *option )
```

[パラメータ]
```
H   comNo                 :通信ポート
                              COM0          :カシオ IR インタフェース
                              COM1          :シリアルインタフェース
B   *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)
B   *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
UB *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)
CU_DT_OPT *option         :プロトコルオプションエリアアドレス
```

[リターンパラメータ]
```
ER  ercd                  :リターンコード
```

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

[リターンコード]
```
E_OK                      :正常終了
E_NG                      :異常終了
E_PRM                     :パラメータエラー
```

≪備考≫

機能	回線クローズ	関数名	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。

≪C 言語インタフェース≫

[コーリングシーケンス]
　　ER ercd = cu_close(H comNo)

[パラメータ]
　　H　comNo　　:通信ポート
　　　　COM0　　:カシオ IR インタフェース
　　　　COM1　　:シリアルインタフェース

[リターンパラメータ]
　　ER　ercd　　:リターンコード

[リターンコード]
　　E_OK　　:正常終了
　　E_NG　　:異常終了
　　E_PRM　　:パラメータエラー

≪備考≫

機能	エラー詳細情報取得	関数名	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。

≪C 言語インタフェース≫

[コーリングシーケンス]
　　ER ercd = cu_readErrStat(H comNo,　UW *cuStat,　UW *biosStat)

[パラメータ]
　　H　comNo　　:通信ポート
　　　　COM0　　:カシオ IR インタフェース
　　　　COM1　　:シリアルインタフェース
　　UW *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)
　　UW *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)

[リターンパラメータ]
　　ER　ercd　　:リターンコード

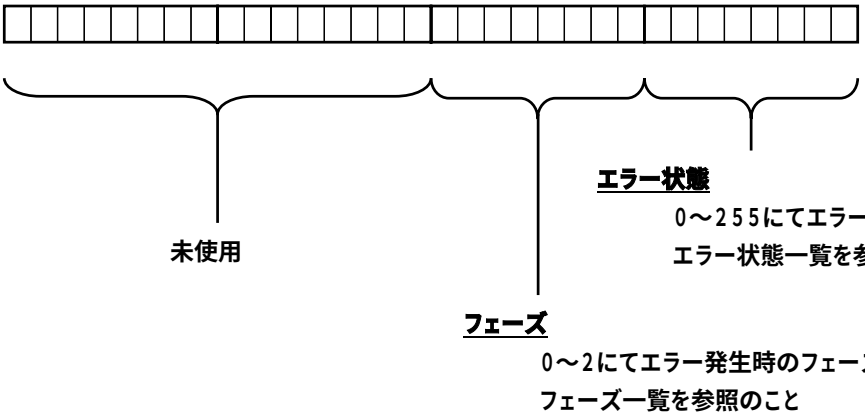
[リターンコード]
　　E_OK　　:正常終了
　　E_PRM　　:パラメータエラー

≪備考≫

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

機能	DT500 プロトコル制御コード拡張設定	関数名	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。

≪C言語インタフェース≫

【コーリングシーケンス】
　　ER ercd = cu_setSOH(UB kind , UB mode, UB *code)

【パラメータ】
　　UB　　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）
　　UB　　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
　　UB　　*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。

【リターンパラメータ】
　　ER　　ercd　　：リターンコード

【リターンコード】
　　E_OK　　：正常終了
　　E_PRM　　：パラメータエラー

≪備考≫

9. タイマ部

9.1. 機能

9.1.1. 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。

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

9.1.2. 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す
周波数	2048Hz	4096Hz	0 128～4096Hz
長さ	50msec	100msec	1～160（×25msec） 0（停止）
その他	システム専用	アプリケーション使用可能	アプリケーション使用可能

9.1.3.　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。

9.2. 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。

機能	タイマー1登録	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER　ercd = s_settimer(ID flgid,　UW setptn,　UW tmcnt);

[パラメータ]
　　ID　flgid　　:イベントフラグID
　　　　FL_TM1_INT_ID を設定してください。
　　UW　setptn　　:ビットパターン
　　　　FL_TM1_INT_RTC0 ～ FL_TM1_INT_RTC31
　　UW　tmcnt　　:タイマーカウント
　　　　1～3600　　(1 カウント = 1 秒)

[リターンパラメータ]
　　ER　ercd　　:リターンコード

[リターンコード]
　　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

≪備考≫
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。

機能	タイマー1削除	関数名	s_timerend

登録済みタイマー1を削除します。

≪C言語インタフェース≫

[コーリングシーケンス]

```
ER    ercd = s_timerend( ER del_id );
```

[パラメータ]

```
ER    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。
```

[リターンパラメータ]

```
ER    ercd        :リターンコード
```

[リターンコード]

```
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
```

≪備考≫

機能	タイマー2登録	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER　ercd = s_settimer2(ID flgid，　UW setptn，　UW tmcnt);

[パラメータ]
　　ID　flgid　　：イベントフラグID
　　　　FL_TM2_INT_ID を設定してください。

　　UW　setptn　　：ビットパターン
　　　　FL_TM2_INT_ITU0 ～ FL_TM2_INT_ITU31

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

[リターンパラメータ]
　　ER　ercd　　：リターンコード

[リターンコード]
　　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

≪備考≫
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。

機能	タイマー2削除	関数名	s_timerend2

登録済みタイマー2を削除します。

≪C言語インタフェース≫

[コーリングシーケンス]

```
ER    ercd = s_timerend2( ER del_id );
```

[パラメータ]

```
ER    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。
```

[リターンパラメータ]

```
ER    ercd            :リターンコード
```

[リターンコード]

```
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
```

≪備考≫

機能	エラービープ音	関数名	s_beep

エラービープ音を鳴らします。
・周波数 = 4096Hz
・音長　 = 100msec

≪C言語インタフェース≫

[コーリングシーケンス]
　　void s_beep(void);

[パラメータ]
　　なし

[リターンパラメータ]
　　なし

[リターンコード]
　　なし

≪備考≫
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。

機能	サウンド音	関数名	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)

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER　ercd = s_sound(UW freq,　UW leng);

[パラメータ]
UW　freq　　:周波数 (0, 128～4096 Hz)
UW　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。

[リターンパラメータ]
　　ER　ercd　　:リターンコード

[リターンコード]
　　E_OK　　:正常終了
　　E_PRM　　:パラメータエラー

≪備考≫
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。

機能	日付の設定	関数名	s_dateset

日付を設定します。

≪C言語インタフェース≫

【コーリングシーケンス】
```
ER    ercd = s_dateset( DAY_DAT *day_dat );
```

【パラメータ】
```
DAY_DAT    *day_dat          :日付格納エリアアドレス
```

【ストラクト構造】
```
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;
```

【リターンパラメータ】
```
ER    ercd          :リターンコード
```

【リターンコード】
```
E_OK              :正常終了
E_PRM             :パラメータエラー
```

≪備考≫

機能	日付の取得	関数名	s_dateget

現在の日付を取得します。

≪C言語インタフェース≫

【コーリングシーケンス】
```
ER    ercd = s_dateget( DAY_DAT *day_dat );
```

【パラメータ】
```
DAY_DAT    *day_dat    :日付格納エリアアドレス
```

【ストラクト構造】
```
typedef    struct day_tabl {
                                UH     year;          :西暦 (1980 ～ 2079)
                                UB     month;         :月 (1 ～ 12)
                                UB     day;           :日 (1 ～ 31)
                           } DAY_DAT;
```

【リターンパラメータ】
```
ER    ercd        :リターンコード
```

【リターンコード】
```
E_OK              :正常終了
E_PRM             :パラメータエラー
```

≪備考≫
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。

機能	時刻の設定	関数名	s_timeset

時刻を設定します。

≪C言語インタフェース≫

【コーリングシーケンス】

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

【パラメータ】

```
TIM_DAT    *tim_dat          :時刻格納エリアアドレス
```

【ストラクト構造】

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

【リターンパラメータ】

```
ER    ercd        :リターンコード
```

【リターンコード】

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

≪備考≫

機能	時刻の取得	関数名	s_timeget

現在の時刻を取得します。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER　ercd = s_timeget(TIM_DAT *tim_dat);

[パラメータ]
　　TIM_DAT　*tim_dat　　:時刻格納エリアアドレス

[ストラクト構造]
```
　　typedef　struct tim_tabl {
                    UB    hour;      :時 (0 ～ 23)
                    UB    mint;      :分 (0 ～ 59)
                    UB    sec;       :秒 (0 ～ 59)
                } TIM_DAT;
```

[リターンパラメータ]
　　ER　ercd　　:リターンコード

[リターンコード]
　　E_OK　　:正常終了
　　E_PRM　　:パラメータエラー

≪備考≫
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。

10. 電源

10.1.　機能

10.1.1. 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。

10.1.2. 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。

10.1.3. 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。

10.1.4. 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、キー入力で再びＯＮします。

10.1.5. 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。

10.1.6. 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。

10.1.7. 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	
7	ＣＩ信号検出	何もしない	イベント設定	ＣＩ信号受信時	※２

※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。
※２　ＣＩ信号制御は、シリアル／ＰＨＳにあります。

10.1.8. 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。

10.1.9. 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。

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。

機能	ＡＰＯ禁止設定	関数名	pwr_hold_apo

ＡＰＯ禁止／禁止解除の設定を行います。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER　ercd = pwr_hold_apo(UH OnOff,　UW BitPtrn);

[パラメータ]
　　UH　OnOff　　：ＡＰＯ禁止設定
　　　　PWR_ON　　：禁止設定
　　　　PWR_OFF　：禁止解除
　　UW　BitPtrn　　：ビットパターン要因
　　　　FL_INV_APO_USR を設定して下さい。

[リターンパラメータ]
　　ER　ercd　　：リターンコード

[リターンコード]
　　E_OK　　：正常終了
　　E_PRM　　：パラメータエラー

≪備考≫

機能	電源オフ	関数名	pwr_off

電源をオフにします。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER　ercd = pwr_off(UH OnOff);

[パラメータ]
　　UH　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。

[リターンパラメータ]
　　ER　ercd　　:リターンコード

[リターンコード]
　　E_OK　　:正常終了
　　E_PRM　　:パラメータエラー

備考

11. 通知モード

11.1. 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。

11.2.　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。

11.3. 通知モード使用例

11.3.1. 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。

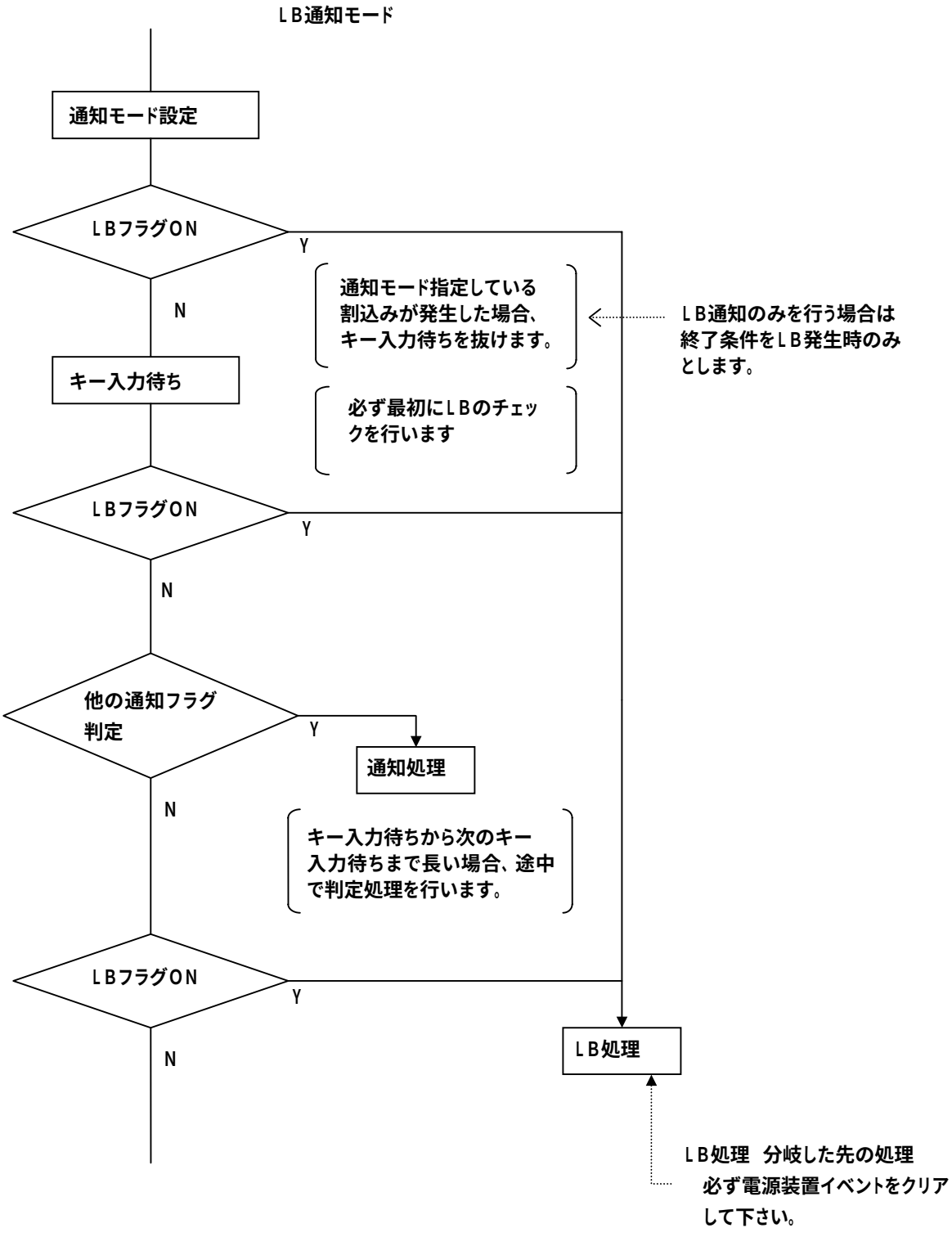
表11．1　通知モードビットパターン

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_CI_INT_SIF	CI検出用ビットパターン （ｼﾘｱﾙｲﾝﾀﾌｪｰｽ）
FL_CI_INT_PHS	CI検出用ビットパターン (PHS ｲﾝﾀﾌｪｰｽ)
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。



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          dumy
        UW          ptn, i;
        KEY_INP     keyinf;
                        ・
pwr_inhabit( PWR_ON, FL_LB_INT_ID,                                  通知モード設定
             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( &dumy, &ptn, FL_LB_INT_ID );             フラグ状態取得
            if(ptn & FL_LB_INT_LB0)
            {
                pwr_inhabit_clr( FL_LB_INT_ID, FL_LB_INT_LB0 );
                sub_lb0();                                          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
            }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;                    リターン条件セット
            keyinf.type = LCD_ATTR_NORMAL;
            keyinf.column_pos = 0;
            keyinf.line_pos = 0;
            retcd = key_read(&keyinf);                              (LB による脱出)
        }
        pwr_inhabit( PWR_OFF, FL_LB_INT_ID,                         通知モード解除
             FL_LB_INT_LB0|FL_LB_INT_LB1|FL_LB_INT_LB2);
                        ・
                        ・
                        ・
void    sub_lb0( void )                                             各 LB に対応する処理
{
                        ・
                        ・
    return;
}

void    sub_lb1( void )
{
                        ・
                        ・
    return;
}

void    sub_lb2( void )
{
                        ・
                        ・
    return;
}
```

11.3.2.　キーに対する通知モードの場合

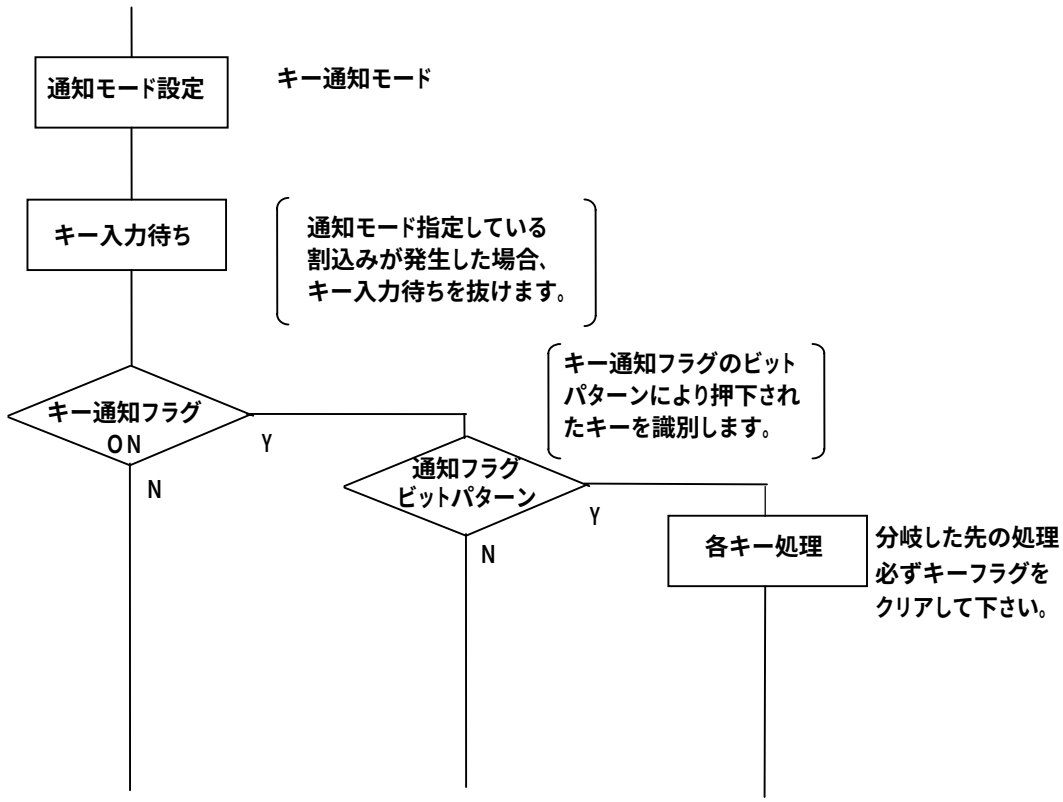
キーに対する通知モードを使用する場合、下記の点に注意して下さい。

- 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              dumy, fid;
                        ・
                        ・
        fid = FL_FK_INT_ID;                                          通知モード設定
        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 );
                        ・
                        ・
                        ・                                           リターン条件設定
        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 );                                 (KEY による脱出)
        if(retcd == E_KEY_INT)
        {
            err = flg_sts( &dumy, &ptn, FL_FK_INT_ID );              通知フラグ状態をクリア後
            if(ptn & FL_FK_INT_FNC1)                                 各 KEY に対応した処理へ
            {                                                        の分岐
                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;                                         通知モード解除
        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 )
    {                                                                各 KEY に対応する処理
                        ・
                        ・
          return;
    }
    void        sub_fnc2( void )
    {
                        ・
                        ・
        return;
    }
```

11.4. 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。

機能	通知フラグ状態取得	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

ID	*p_flgpid	:ワーク領域の先頭アドレス
UW	*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
ID	flgid	:フラグID

[リターンパラメータ]

ER	ercd	:リターンコード
ID	*p_flgpid	:ワークデータ
UW	*p_flgptn	:フラグのビットパターンを格納する領域の先頭アドレス

[リターンコード]

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

≪備考≫

機能	通知フラグ状態クリア	関数名	clr_flg

指定 ID の通知フラグの指定ビットをクリアします。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER ercd = clr_flg(ID flgid, UW setptn);

[パラメータ]
　　ID　flgid　　:フラグID
　　UW　setptn　　:クリアするビットパターン

[リターンパラメータ]
　　ER　ercd　　:リターンコード

[リターンコード]
　　E_OK　　:正常終了
　　E_NOEXS　　:フラグ ID 範囲外／予約 ID

≪備考≫

機能	フラグセット待ち	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]

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

[パラメータ]

UW	*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
ID	flgid	:フラグID
UW	waipth	:待ちビットパターン
UW	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)

[リターンパラメータ]

ER	ercd	:リターンコード
UW	*p_flgptn	:待ち解除時のビットパターンを格納する領域の先頭アドレス

[リターンコード]

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

≪備考≫

機能	通知モード設定	関数名	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。

≪C言語インタフェース≫

[コーリングシーケンス]

```
ER   ercd = pwr_inhabit( UH OnOff、 ID EventFlg、 UW BitPtrn );
```

[パラメータ]

UH　OnOff	:通知モード	
	PWR_ON	:通知モード設定
	PWR_OFF	:通知モード解除
ID　EventFlg	:イベントフラグID	
UW　BitPtrn	:ビットパターン	
	FL_LB_INT_LB0 - FL_LB_INT_LB5	:LB検出
	FL_CI_INT_SIF	:CI検出　(シリアルインタフェース)
	FL_CI_INT_PHS	:CI検出　(PHSインタフェース)
	FL_SET_INT_IO	:IO ボックス検出

[リターンパラメータ]

ER　ercd	:リターンコード

[リターンコード]

E_OK	:正常終了
E_PRM	:パラメータエラー

≪備考≫

機能	電源通知イベントのクリア	関数名	pwr_inhabit_clr

電源通知イベントフラグをクリアします。

≪C言語インタフェース≫

[コーリングシーケンス]
　　ER　ercd = pwr_inhabit_clr(ID EventFlg、 UW BitPtrn);

[パラメータ]
　　ID　EventFlg　　:イベントフラグID
　　UW　BitPtrn　　:ビットパターン
　　　　FL_LB_INT_LB0 - FL_LB_INT_LB5　　:LB検出
　　　　FL_CI_INT_SIF　　:CI検出　(シリアルインタフェース)
　　　　FL_CI_INT_PHS　　:CI検出　(PHSインタフェース)
　　　　FL_SET_INT_IO　　:IO ボックス検出

[リターンパラメータ]
　　ER　ercd　　:リターンコード

[リターンコード]
　　E_OK　　:正常終了
　　E_PRM　　:パラメータエラー

≪備考≫

12. 共通関数

12.1. 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。

12.1.1. 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、INIT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。

12.1.2. 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リターンコード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。

12.1.3. 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。

12.1.4. 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。

12.2. 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。

機能	ＡＢＯＲＴ処理	関数名	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≫

[コーリングシーケンス]
void = abort(int user_code);

[パラメータ]
int　user_code　：表示させたい任意のコード

[リターンパラメータ]
なし

≪備考≫
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。

機能	EXIT処理	関数名	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リターンコードの待避
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≫

[コーリングシーケンス]
void = exit(int rtn_code);

[パラメータ]
int　rtn_code　：ユーザアプリケーションのリターンコード(固定エリアに保存されます)

[リターンパラメータ]
なし

≪備考≫

機能	動作環境メニューの起動	関数名	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。

≪C 言語インタフェース≫

[コーリングシーケンス]
　　void = wkup_cost(void);

[パラメータ]
　　なし

[リターンパラメータ]
　　なし

≪備考≫

機能	ＯＢＲキャリブレーション処理起動	関数名	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≫

[コーリングシーケンス]
　　void = wkup_calib(void)

[パラメータ]
　　なし

[リターンパラメータ]
　　なし

≪備考≫

13.参考資料

13.1. 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	DT-700	DT-750	DT-800	本機
lcd_csr_set	ｶｰｿﾙﾀｲﾌﾟ設定	○	○	○	○
lcd_csr_put	ｶｰｿﾙ位置設定	○	○	○	○
lcd_csr_get	ｶｰｿﾙ位置読出し	○	○	○	○
lcd_char	1文字表示	○	○	○	○
lcd_string	文字列表示	○	○	○	○
lcd_line	直線描画	○	○	○	○
lcd_cls	画面ｸﾘｱ	○	○	○	○
lcd_led	LED の制御	○	○	○	○
lcd_gaiji	外字ﾌｫﾝﾄ登録	○	○	○	○
lcd_string2	文字列表示 2(ｽｸﾛｰﾙ抑制)	○	○	○	○
lcd_usrfont	ﾕｰｻﾞﾌｫﾝﾄﾌｧｲﾙ登録	○	○	○	○
lcd_romfont	ROM ﾌｫﾝﾄ設定	○	○	○	○
lcd_userstr	ﾕｰｻﾞ文字列表示	○	○	○	○
lcd_active_set	ﾕｰｻﾞ描画ﾍﾟｰｼﾞ設定			○	
lcd_visual_set	ﾕｰｻﾞ表示ﾍﾟｰｼﾞ設定			○	
lcd_active_get	ﾕｰｻﾞ描画ﾍﾟｰｼﾞ読出し			○	
lcd_visual_get	ﾕｰｻﾞ表示ﾍﾟｰｼﾞ読出し			○	
lcd_cls_page	指定ﾍﾟｰｼﾞｸﾘｱ			○	
lcd_grchar	ｸﾞﾗﾌｨｯｸ文字表示			○	
lcd_box	BOX 描画/削除			○	
lcd_el	EL ﾊﾞｯｸﾗｲﾄの制御			○	○

(3) キー部

関数名	機能	DT-700	DT-750	DT-800	本機
key_read	1文字入力	○	○	○	○
key_string	文字列入力	○	○	○	○
key_num	数値入力	○	○	○	○
key_check	ｷｰﾊﾞｯﾌｧのｽﾃｰﾀｽﾁｪｯｸ	○	○	○	○
key_clear	ｷｰﾊﾞｯﾌｧのｸﾘｱ	○	○	○	○
key_fnc	ﾌｧﾝｸｼｮﾝｷｰｺｰﾄﾞの設定	○	○	○	○
key_fnc_mode	ﾌｧﾝｸｼｮﾝｷｰ通知ﾓｰﾄﾞ設定	○	○	○	○
key_select	ｷｰ入力ﾓｰﾄﾞ設定		○	○	○
key_patchg	ｷｰ入力有効無効設定/解除		○		
key_maketime	切替えｷｰ確定時間設定		○		
key_pad_set	ｷｰﾊﾟｯﾄﾞﾌｧｲﾙ登録			○	
key_touch	ﾕｰｻﾞﾀｯﾁｷｰ設定削除			○	
key_point	ﾀｯﾁ座標取得			○	
key_pad	ｷｰﾊﾟｯﾄﾞ切替/状態取得			○	
key_pad_entry	ｷｰﾊﾟｯﾄﾞ遷移設定/状態取得			○	

(4) 通信部

関数名	機能	DT-700	DT-750	DT-800	本機
c_open	COM のｵｰﾌﾟﾝ	○	○	○	○
c_close	COM のｸﾛｰｽﾞ	○	○	○	○
c_status	COM ｽﾃｰﾀのｽﾘｰﾄﾞ	○	○	○	○
c_hold	COM の占有	○	○	○	○
c_chkopen	COM のｵｰﾌﾟﾝﾁｪｯｸ	○	○	○	○
c_wp	WakeUp 信号の設定				○
c_dout	n 文字送信	○	○	○	○
c_din	1 文字受信	○	○	○	○
c_tmdin	ﾀｲﾑｱｳﾄ監視受信	○	○	○	○
c_mdout	ﾒﾓﾘﾌﾞﾛｯｸ送信	○	○		
c_mdin	ﾒﾓﾘﾌﾞﾛｯｸ受信	○	○		
c_out	1 文字送信	○	○	○	○
c_break	ﾌﾞﾚｰｸ信号の制御	○	○	○	○
c_txrx	送受信の有効/無効	○	○	○	○
c_iobox	IO ﾎﾞｯｸｽ送信設定	○	○		○
c_irout	IO ﾎﾞｯｸｽ送信	○	○		○
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	受信ﾊﾞｯﾌｧのｸﾘｱ	○	○	○	○
c_bfsts	受信ﾊﾞｯﾌｧｽﾃｰﾀｽのﾘｰﾄﾞ	○	○	○	○
c_errbfring	ﾘﾀｰﾝｺｰﾄﾞﾊﾞｯﾌｧﾘﾝｸﾞ制御の設定	○	○	○	○
c_rderrsts	ｴﾗｰｽﾃｰﾀｽのﾘｰﾄﾞ	○	○	○	○
c_chghdr	受信ﾊﾝﾄﾞﾗ切替え	○	○	○	○
c_cimode	CI 信号立ち上げﾓｰﾄﾞ設定				○
c_brkevent	ﾌﾞﾚｲｸ要因の設定				○

(5) 電源部

関数名	機能	DT-700	DT-750	DT-800	本機
pwr_inhabit	通知ﾓｰﾄﾞ設定	○	○	○	○
pwr_inhabit_clr	電源通知ｲﾍﾞﾝﾄのｸﾘｱ	○	○	○	○
pwr_hold_apo	APO 禁止設定	○	○	○	○
pwr_off	電源ｵﾌ	○	○	○	○

(6) 通知部

関数名	機能	DT-700	DT-750	DT-800	本機
flg_sts	通知ﾌﾗｸﾞ状態取得	○	○	○	○
clr_flg	通知ﾌﾗｸﾞ状態ｸﾘｱ	○	○	○	○
wai_flg	ﾌﾗｸﾞｾｯﾄ待ち	○	○	○	○

(7) バーコード部

関数名	機能	DT-700	DT-750	DT-800	本機
OBR_open	OBR ｵｰﾌﾟﾝ	○	○	○	○
OBR_close	OBR ｸﾛｰｽﾞ	○	○	○	○
OBR_getc	OBR ﾃﾞｰﾀ 1 文字ﾘｰﾄﾞ	○	○	○	○
OBR_gets	OBR ﾃﾞｰﾀ文字列ﾘｰﾄﾞ	○	○	○	○
OBR_stat	OBR ﾊﾞｯﾌｧｽﾃｰﾀｽﾁｪｯｸ	○	○	○	○
OBR_flush	OBR ﾊﾞｯﾌｧのｸﾘｱ	○	○	○	○
OBR_moderd	OBR 動作ﾓｰﾄﾞの取得	○	○	○	○
OBR_modewt	OBR 動作ﾓｰﾄﾞ設定	○	○	○	○
OBR_chgbuf	OBR ﾊﾞｯﾌｧの切替え	○	○	○	○
OBR_gain	発光ｹﾞｲﾝ切替え		○	○	
OBR_trigmode	ﾄﾘｶﾞｰｷｰによる電源ｵﾝ設定		○	○	○
OBR_swing	レーザー発光幅の設定／参照				○
OBR_widenarrow	レーザー発光幅の微調整				○

(8) バーコード動作モード

機能	DT-700	DT-750	DT-800	本機
読取り可能ｺｰﾄﾞ	○	○	○	○
読取り桁数	○	○	○	○
出力ﾌｫｰﾏｯﾄ	○	○	○	○
ﾁｪｯｸﾃﾞｼﾞｯﾄの実行	○	○	○	○
ﾁｪｯｸｷｬﾗｸﾀの出力	○	○	○	○
読取り方式(単発)	○	○	○	○
読取り方式(連続:TRG 有)	○	○	○	○
読取り方式(連続:切替)	○	○	○	○
読取り方式(連続:TRG 無)	○	○	○	○
ｹﾞｲﾝｺﾝﾄﾛｰﾙ	○	○	×	×
ﾌﾞｻﾞｰ制御	○	○	○	○
LED 制御	○	○	○	○
出力ﾊﾞｯﾌｧ	○	○	○	○
終了ｺｰﾄﾞ	○	○	○	○
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	DT-700	DT-750	DT-800	本機
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	ｴﾗｰﾋﾞｰﾌﾟ音	○	○	○	○
s_beep2	ｴﾗｰﾋﾞｰﾌﾟ音 2 (赤ＬＥＤ点灯)		○	○	
s_sound	ｻｳﾝﾄﾞ音 1	○	○	○	○
s_dateget	日付の取得	○	○	○	○
s_dateset	日付の設定	○	○	○	○
s_timeget	時間の取得	○	○	○	○
s_timeset	時間の設定	○	○	○	○

(10) データ管理部

関数名	機能	DT-700	DT-750	DT-800	本機
dat_mem_size	ﾒﾓﾘ領域の空きｻｲｽﾞの取得	○	○	○	○
dat_system	ｼｽﾃﾑﾃﾞｰﾀの設定	○	○	○	○
dat_OSVer_Read	OS ﾊﾞｰｼﾞｮﾝ読出し		○	○	○
dat_dealer_chk	代理店 ID のﾁｪｯｸ			○	○
dat_Apload (注 2)	AP ﾛｰﾄﾞ&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	管理ﾃﾞｰﾀ	DT-700	DT-750	DT-800	本機
電源関連	APO 時間	○	○	○	○
	ABO 時間	○	○	○	○
	ﾚｼﾞｭｰﾑ 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)	○	○	○	○
	ﾃﾞｰﾀ(IR)	○	○	予約領域	○
	ﾊﾟﾘﾃｨ(IR)	○	○	予約領域	○
	STOP(IR)	○	○	予約領域	○
	速度(RF/ｼﾘｱﾙ)	○	○	予約領域	○
	ﾃﾞｰﾀ(RF/ｼﾘｱﾙ)	○	○	予約領域	○
	ﾊﾟﾘﾃｨ(RF/ｼﾘｱﾙ)	○	○	予約領域	○
	STOP(RF/ｼﾘｱﾙ)	○	○	予約領域	○
	速度(10P)	○	○	○	
	ﾃﾞｰﾀ(10P)	○	○	予約領域	
	ﾊﾟﾘﾃｨ(10P)	○	○	予約領域	
	STOP(10P)	○	○	予約領域	
	ﾃﾞﾌｫﾙﾄ通信 PORT			○	
	速度(PHS)				○
	ﾃﾞｰﾀ(PHS)				○
	ﾊﾟﾘﾃｨ(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 ﾊﾞｰｼﾞｮﾝ	○	○	○	○
	PATCH ﾊﾞｰｼﾞｮﾝ			○	○
	機器種別	○	○	○	○
ﾈｯﾄﾜｰｸ関連	IP ｱﾄﾞﾚｽ(SS 無線)			○	
	ﾏｽｸ値(SS 無線)			○	

(前頁つづき)

項目	管理ﾃﾞｰﾀ	DT-700	DT-750	DT-800	本機
ﾌﾟﾛﾄｺﾙ関連	通常受信ﾀｲﾑｱｳﾄ	○	○		○
	通常ﾘﾄﾗｲ回数	○	○		○
	ﾏﾙﾁﾃﾞｰﾀﾘﾝｸ受信ﾀｲﾑｱｳﾄ	○			○
	対向送信ﾃﾞｰﾀﾘﾝｸ受信ﾀｲﾑｱｳﾄ	○			
	対向受信ﾃﾞｰﾀﾘﾝｸ受信ﾀｲﾑｱｳﾄ	○			
	対向受信ﾃﾞｰﾀﾘﾝｸﾘﾄﾗｲ回数	○			
	ﾃﾞｰﾀﾘﾝｸ受信ﾀｲﾑｱｳﾄ		○		
	受信ﾃﾞｰﾀなしﾀｲﾑｱｳﾄ		○		
	再ﾃﾞｰﾀﾘﾝｸ可能回数		○		
	ｾｯｼｮﾝ確立ﾀｲﾑｱｳﾄ			○	○
	受信ﾀｲﾑｱｳﾄ			○	○
	DR ﾀｲﾑｱｳﾄ(10PIN)			○	
	CS ﾀｲﾑｱｳﾄ(10PIN)			○	
	CD ﾀｲﾑｱｳﾄ(10PIN)			○	
	ｼﾘｱﾙ NO				○
	水平ﾊﾟﾘﾃｨ				○
	ﾘﾝｸﾀｲﾑｱｳﾄ				○
ﾒﾓﾘ関連	ｱﾌﾟﾘｹｰｼｮﾝ SIZE			○	○
ﾌｧｲﾙﾓｰﾄﾞ	FORMAT				○

(12) ファイル部

関数名	機能	DT-700	DT-750	DT-800	本機
dat_fsize	ﾌｧｲﾙ空き領域ｻｲｽﾞの取得	○	○		○
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	ﾌｧｲﾙﾃﾞｰﾀの検索	注1	○		○
dat_fsize_chg	ﾌｧｲﾙｻｲｽﾞ変更		○		
open	ﾌｧｲﾙｵｰﾌﾟﾝ	○	○	○	○
close	ﾌｧｲﾙｸﾛｰｽﾞ	○	○	○	○
read	ﾌｧｲﾙのﾘｰﾄﾞ	○	○	○	○
write	ﾌｧｲﾙのﾗｲﾄ	○	○	○	○
lseek	ﾌｧｲﾙﾘｰﾄﾞ/ﾗｲﾄ位置の設定	○	○	○	○
sbrk	ﾒﾓﾘ領域の割当て	○	○	○	○
fil_mkdir	ﾃﾞｨﾚｸﾄﾘの作成			○	○
fil_rmdir	ﾃﾞｨﾚｸﾄﾘの削除			○	○
fil_remove	ﾌｧｲﾙの削除			○	○
fil_rename	ﾌｧｲﾙ名の変更/移動			○	○
fil_fstat	ﾌｧｲﾙの日時･ｻｲｽﾞ･属性の取得			○	○
fil_chsize	ﾌｧｲﾙのｻｲｽﾞの変更			○	○
fil_getsize	ﾌｧｲﾙ領域空きｻｲｽﾞの取得			○	○
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ｽﾞの取得				○
fil_filefind	ﾌｧｲﾙ全ﾊﾟｽ名の取得				○

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	DT-700	DT-750	DT-800	本機
abort	ABORT 処理	○	○	○	○
exit	EXIT 処理	○	○	○	○
wkup_cost	動作環境ﾒﾆｭｰの起動	○	○	○	○
wkup_calib	ｷｬﾘﾌﾞﾚｰｼｮﾝ起動			○	○
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	DT-700	DT-750	DT-800	本機
cu_open	通信ﾎﾟｰﾄ初期化	○	○	○	○
cu_stopKeySet	中断ｷｰの登録/削除	○	○	○	○
cu_fileSend	ﾌｧｲﾙ送信	○	○	○	○
cu_fileSendSet	ﾌｧｲﾙ送信情報設定	○	○		○
cu_fileSend1	1 ﾌｧｲﾙ送信	○	○		○
cu_fileRecv	ﾌｧｲﾙ受信	○	○	○	○
cu_msgSend	画面表示ﾒｯｾｰｼﾞ送信	○	○	○	○
cu_end	通信中断	○	○		○
cu_close	回線ｸﾛｰｽﾞ	○	○	○	○
cu_readErrStat	ｴﾗｰ詳細情報取得	○	○	○	○
cu_readDIRjInfo	ﾃﾞｰﾀﾘﾝｸ拒否情報取得	○	○		○
cu_downloadSet	ﾀﾞｳﾝﾛｰﾄﾞ動作設定		○		
cu_fileAdd	ﾌｧｲﾙの追加			○	○
cu_idle	IDLE 遷移			○	○
cu_cmdRecv	ｺﾏﾝﾄﾞ受信待ち			○	○
cu_fileDelete	ﾌｧｲﾙ削除			○	○
cu_fileMove	ﾌｧｲﾙ移動			○	○
cu_makeDir	ﾃﾞｨﾚｸﾄﾘ作成			○	○
cu_dateTime	日付時刻の取得および設定			○	○
cu_getFileInfo	ﾌｧｲﾙ情報の取得			○	○
cu_setFileInfo	ﾌｧｲﾙ情報の更新			○	○
cu_getDiskInfo	ﾃﾞｨｽｸ情報の取得			○	○
cu_beep	ﾌﾞｻﾞｰ鳴動			○	○
cu_getSysInfo	ｼｽﾃﾑ情報の取得			○	○
cu_apRecvSet	AP ｲﾝｽﾄｰﾙ設定			○	
cu_fchklog_Create	FCHK ﾘｽﾄﾌｧｲﾙの生成			○	○
cu_fchklog_Check	FCHK ﾘｽﾄﾌｧｲﾙのﾁｪｯｸ			○	○
cu_setDrive	転送ﾄﾞﾗｲﾌﾞ指定				○
cu_msgSend	画面表示ﾒｯｾｰｼﾞ送信				○
cu_SetCode	DT500 ﾌﾟﾛﾄｺﾙ制御ｺｰﾄﾞ拡張設定				○

(15) IrDA部

関数名	機能	DT-700	DT-750	DT-800	本機
Ir_Open	IrCOMM ｵｰﾌﾟﾝ			○	○
Ir_Close	IrCOMM ｸﾛｰｽﾞ			○	○
Ir_Read	ﾃﾞｰﾀ読込み			○	○
Ir_Write	ﾃﾞｰﾀ書込み			○	○
Ir_QueryTx	送信ﾃﾞｰﾀ数問合せ			○	○
Ir_QueryRx	受信ﾃﾞｰﾀ数問合せ			○	○
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	ﾊﾟﾗﾒｰﾀ設定			○	

(16) メモリーバックアップ

関数名	機能	DT-700	DT-750	DT-800	本機
mbu_format	FROM 領域の確保		○		
mbu_open	ﾌｧｲﾙｵｰﾌﾟﾝ		○		
mbu_close	ﾌｧｲﾙｸﾛｰｽﾞ		○		
mbu_write	ﾌｧｲﾙ書込み		○		
mbu_read	ﾌｧｲﾙ読出し		○		
mbu_clear	FROM ｸﾘｱ		○		
mbu_sts	ﾒﾓﾘﾊﾞｯｸｱｯﾌﾟ状態の取得		○		

(17) プリンタ

関数名	機能	DT-700	DT-750	DT-800	本機
prn_inf_open	赤外ｵｰﾌﾟﾝ		○		
prn_inf_close	赤外ｸﾛｰｽﾞ		○		
prn_inf_send	赤外のｺﾏﾝﾄﾞ送信		○		
prn_inf_status	赤外のｽﾃｰﾀｽﾘｰﾄﾞ		○		
prn_wir_open	無線ｵｰﾌﾟﾝ		○		
prn_wir_close	無線ｸﾛｰｽﾞ		○		
prn_wir_send	無線のｺﾏﾝﾄﾞ送信		○		
prn_pktlen	無線のﾊﾟｹｯﾄ長登録		○		
prn_crc_calc	CRC 算出		○		

(18) システムメニュー

項目	設定内容	DT-700	DT-750	DT-800	本機
TOP 項目選択	AP 起動	○	○	○	○
	動作環境ﾒﾆｭｰ起動	○	○	○	○
	日付時刻設定	○	○	○	○
	転送	○	○	○	○
	FROM ﾊﾞｯｸｱｯﾌﾟ		○		
	ｷｬﾘﾌﾞﾚｰｼｮﾝ起動			○	
	OS ﾊﾞｰｼﾞｮﾝ表示	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			○	○
	通信ﾎﾟｰﾄ設定			○	○
	通信速度設定			○	○
	ﾌﾟﾛﾄｺﾙ				○
ﾒﾓﾘ転送	本体受信		○		
	本体送信		○		
	通信ﾎﾟｰﾄ設定		○		
	通信速度設定		○		
	ﾁｪｯｸｻﾑ		○		
子機作成	本体送信			○	○
	本体受信			○	○
	転送ﾄﾞﾗｲﾌﾞ			○	○
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	○	○		
	通信ﾎﾟｰﾄ設定		○		
	通信速度設定		○		
	ﾌｧｲﾙ送信			○	○
	ﾌｧｲﾙ受信			○	○
	ﾄﾞﾗｲﾌﾞﾌｫｰﾏｯﾄ			○	○
	ﾒﾓﾘｻｲｽﾞ変更			○	○
	ﾌｧｲﾙﾓｰﾄﾞ				○
ﾌｧｲﾙ転送	ﾎｽﾄ受信	○	○		
	ﾎｽﾄ送信	○	○		

(19) 動作環境メニュー

項目	設定内容	DT-700	DT-750	DT-800	本機
TOP 項目選択	環境	○	○	○	○
	表示ﾓｰﾄﾞ	○	○	○	○
	通信ｾｯﾄ	○	○		
	ﾊﾞｰｺｰﾄﾞ	○	○	○	○
	ID セット	○	○	○	○
環境	APO 時間	○	○	○	○
	ABO 時間	○	○	○	○
	ｷｰｸﾘｯｸ ON/OFF	○	○	○	○
	ﾌﾞｻﾞｰ音量	○	○	○	○
	自動ｺﾝﾄﾗｽﾄ ON/OFF			○	
	警告ﾒｯｾｰｼﾞ ON/OFF			○	
表示ﾓｰﾄﾞ	ﾌｫﾝﾄﾓｰﾄﾞ	○	○	○	○
	ﾒｯｾｰｼﾞ	○	○	○	○
ﾌｫﾝﾄﾓｰﾄﾞ	ｻｲｽﾞ(12/16/24)			○	
	ｻｲｽﾞ(6/8/10)				○
	ﾀｲﾌﾟ(標準/強調)			○	○
通信ｾｯﾄ	通信ﾎﾟｰﾄ	○	○		
	通信速度	○	○		
	ﾃﾞｰﾀ長	○	○		
	ﾊﾟﾘﾃｨ	○	○		
	ｽﾄｯﾌﾟﾋﾞｯﾄ	○	○		
ﾊﾞｰｺｰﾄﾞ	読取り回数	○	○	○	○
	照合回数	○	○	○	○
	ﾀｲﾑｱｳﾄ	○	○	○	○
	読取り禁止時間		○		
	ｷｬﾘﾌﾞﾚｰｼｮﾝ				○
ID セット	機器 ID	○	○	○	○
	代理店 ID			○	○

最終ページ