DT-300
C ライブラリ解説書

Rev1.01

カシオ計算機株式会社

目 次

1 概要 4
1.1 提供ファイルについて 4
1.2 標準ライブラリ関数 2
1.3 専用ライブラリ関数 4
1.3.1 データ管理部 4
1.3.2 表示部関数 4
1.3.3 キー部関数 5
1.3.4 ＯＢＲ部関数 5
1.3.5 通信部関数 5
1.3.6 IrDA 部関数 6
1.3.7 通信ユーティリティ部関数 6
1.3.8 タイマー部関数 7
1.3.9 電源部関数 7
1.3.10 通知モード部関数 8
1.3.11 共通関数 8
1.4 データタイプ 8
2 データ管理部 9
2.1 機能 9
2.1.1 メモリー管理機能 9
2.1.2 ファイル管理機能 9
2.1.3 データ管理機能（システムデータ管理） 11
2.1.4 低水準インタフェース 14
2.1.5 システムデータファイル 15
2.2 ファンクション詳細 17
3 表示部 43
3.1 表示制御 43
3.1.1 表示画面 43
3.1.2 表示コード 45
3.1.3 制御コード表示 48
3.1.4 ＥＳＣ文字列の表示 49
3.1.5 例外表示制御 50
3.1.6 スクロール制御 52
3.2 フォント制御 53
3.2.1 フォントの種類 53
3.2.2 フォントデータ構成 54
3.2.3 修飾文字のフォントデータ制御 58
3.2.4 ユーザーフォントファイル 60
3.3 ファンクション詳細 62
4 キー部 78
4.1 機能 78
4.1.1 キー構成 78
4.1.2 キーモード 78
4.1.3 1 文字入力 79
4.1.4 文字列入力 80
4.1.5 数値入力 81
4.1.6 キーコードの設定 82
4.1.7 キー通知設定 83
4.1.8 キー入力有効／無効設定 84
4.1.9 キーバッファ 85
4.1.10 バックライト制御 85
4.1.11 多点押し処理 85
4.1.12 キーロールオーバー機能 87
4.2 キーコード 87
4.2.1 属性 87

4.2.2 コード ... 88
4.3 ファンクション詳細 ... 89
5 OBR 部 ... 99
5.1 基本仕様 ... 99
5.1.1 レーザースキャナ部 ... 99
5.1.2 デコード仕様 ... 99
5.2 機能 ... 102
5.2.1 1 文字／文字列の読込み ... 102
5.2.2 OBR データバッファの状態チェック ... 102
5.2.3 OBR データバッファのクリア ... 102
5.2.4 格納先バッファの切り替え ... 102
5.2.5 モード設定 ... 103
5.3 ファンクション詳細 ... 108
6 通信部 ... 122
6.1 通信仕様 ... 122
6.1.1 通信インタフェース ... 122
6.2 機能 ... 122
6.2.1 通信ポートのオープン・クローズ・占有 ... 122
6.2.2 転送データの送信・受信 ... 123
6.2.3 ＳＩ／ＳＯ制御 ... 129
6.2.4 フロー制御 ... 130
6.2.5 デリートコード制御 ... 130
6.2.6 エラーコードバッファリング制御 ... 130
6.2.7 信号線制御とタイムアウト監視 ... 131
6.2.8 ＬＢ検出 ... 136
6.2.9 ブレイク要因検出 ... 137
6.3 エラー詳細 ... 138
6.3.1 ファンクションのエラー検出 ... 138
6.3.2 エラー詳細 ... 139
6.4 通信関数 補足 ... 147
6.4.1 受信データの読込み ... 147
6.4.2 ＬＢエラーチェック ... 148
6.4.3 中断キーによる処理 ... 149
6.4.4 通信関数部制限 ... 150
6.5 ファンクション詳細 ... 151
7 IrDA 部 ... 175
7.1 通信仕様 ... 175
7.1.1 通信インタフェース ... 175
7.2 機能 ... 175
7.2.1 シリアルポートエミュレーション ... 175
7.2.2 ファンクションコール ... 175
7.2.3 ファンクションの優先順位 ... 177
7.3 ファンクション詳細 ... 178
8 通信ユーティリティ部 ... 195
8.1 HIOWIN プロトコル ... 195
8.1.1 システム構成 ... 195
8.1.2 関数一覧 ... 195
8.1.3 ファンクション詳細 ... 196
8.2 FLINK プロトコル ... 215
8.2.1 システム構成 ... 215
8.2.2 関数一覧 ... 215
8.2.3 ファンクション詳細 ... 216
9 タイマ部 ... 240
9.1 機能 ... 240
9.1.1 タイマー部 ... 240
9.1.2 ブザー音鳴動部 ... 240
9.1.3 日付時刻制御部 ... 241
9.2 ファンクション詳細 ... 242

10　電源 253
10.1　機能 253
10.1.1　主電池電圧低下監視／警告 253
10.1.2　自動電源ＯＦＦ制御（ＡＰＯ：Ａｕｔｏ Ｐｏｗｅｒ Ｏｆｆ） 253
10.1.3　自動バックライトＯＦＦ制御（ＡＢＯ：Ａｕｔｏ Ｂａｃｋｌｉｇｈｔ Ｏｆｆ） 253
10.1.4　低消費電力制御 253
10.1.5　ＡＰＯ禁止／禁止解除 253
10.1.6　電源通知モード設定／解除 253
10.1.7　電源通知イベントクリア 254
10.1.8　電源ＯＦＦコマンド 254
10.2　ファンクション詳細 255
11　通知モード 259
11.1　通知モードの概念 259
11.2　通知モード使用時に必要となる関数 259
11.3　通知モード使用例 260
11.3.1　ＬＢに対する通知モードの場合 260
11.3.2　キーに対する通知モードの場合 263
11.4　ファンクション詳細 265
12　共通関数 271
12.1　機能 271
12.1.1　ＡＢＯＲＴ処理 271
12.1.2　ＥＸＩＴ処理 271
12.1.3　動作環境メニュー起動処理 271
12.1.4　ソフトウェアリセット処理 271
12.2　ファンクション詳細 272
13　参考資料 277
13.1　機能比較 277

1 概要

1.1 提供ファイルについて

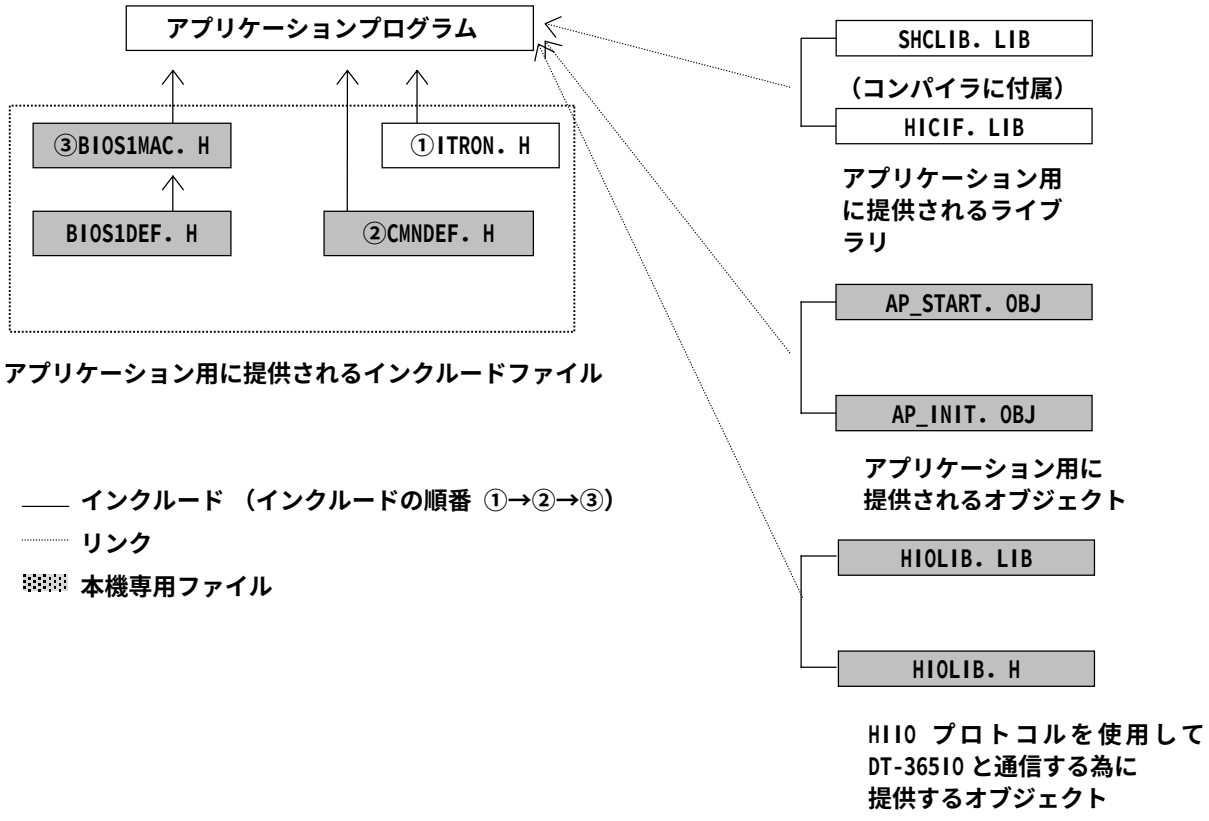
アプリケーションプログラムを作成する場合、必ず“HICIF．LIB“をリンクして下さい。
本機の関数を使用する場合には、本システムが提供する“BIOS1MAC．H“をアプリケーションプログラム内でインクルードして下さい。
また、C 標準ライブラリの機能を使用する場合には、“SHCLIB．LIB“をリンクして下さい。
また、ヘッダファイルは、BIOS1MAC.H より前にインクルードする必要があります。
※アプリケーションは提供された関数群（外部シンボル）の定義ファイルを用いることにより、単独でコンパイル／リンクします。（OS／システムの実体とはリンクしません）

（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_INIT．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
- HIOLIB.LIB ····· HIO プロトコルオブジェクト
- HIOLIB.H ····· HIO プロトコル定義ファイル

： 本機専用ファイル

（2）各ファイルの関係について



1.2 標準ライブラリ関数

下記の関数は、本システムで使用可能なＣ標準ライブラリの一覧です。

インタフェース仕様は、“ ＳＨシリーズＣ言語マニュアルライブラリ編 ” を参照して下さい。

ＮＯ.	ライブラリ名	適用
１	isalnum 関数	英字・１０進数字の判定
２	isalpha 関数	英字の判定
３	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の計算
２１	sin 関数	浮動小数点数のラディアン値の正弦の計算
２２	tan 関数	浮動小数点数のラディアン値の正接の計算
２３	cosh 関数	浮動小数点数の双曲線余弦の計算
２４	sinh 関数	浮動小数点数の双曲線正弦の計算
２５	tanh 関数	浮動小数点数の双曲線正接の計算
２６	exp 関数	浮動小数点数の指数関数の計算
２７	frexp 関数	浮動小数点数を(0.5,1.0)の値として２のべき乗の積に分解
２８	ldexp 関数	浮動小数点数と２のべき乗の乗算の計算
２９	log 関数	浮動小数点数と２のべき乗の乗算の計算
３０	log10 関数	浮動小数点数の１０を底とする対数の計算
３１	modf 関数	浮動小数点数を整数部分と小数部分に分解
３２	pow 関数	浮動小数点数のべき乗の計算
３３	sqrt 関数	浮動小数点数の正の平方根の計算
３４	ceil 関数	浮動小数点数の小数点以下を切り上げた整数値の取得
３５	fabs 関数	浮動小数点数の絶対値の取得
３６	floor 関数	浮動小数点数の小数点以下を切り捨てた整数値の取得
３７	fmod 関数	浮動小数点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

NO.	ライブラリ名	適用
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
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

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	5
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）	32
	fil_rmdir	ディレクトリの削除（ＦＡＴファイルモード）	33
	fil_remove	ファイルの削除	34
	fil_rename	ファイル名の変更／移動	35
	fil_fstat	ファイルの日時・サイズ・属性の取得	36
	fil_chsize	ファイルサイズの変更	37
	fil_getsize	ファイル領域空きサイズの取得	38
	fil_findfirst	ファイル名の取得	39
	fil_findnext	ファイル名の取得（次候補）	40
	fil_filesize	ファイルの個数と総サイズの取得	41
	fil_filefind	ファイル全パス名の取得	42
3	システムデータ管理関数		
	dat_system	システムデータの設定	18
	dat_OSVer_Read	ＯＳバージョン読出し	21
	dat_dealer_chk	代理店ＩＤのチェック	22
	dat_Apload	ＡＰロード＆実行	23
4	低水準インタフェース関数		
	open	ファイルオープン	24
	close	ファイルクローズ	26
	read	ファイルのリード	27
	write	ファイルのライト	28
	lseek	ファイルリード／ライト位置の設定	29
	sbrk	メモリ領域の割当て	30

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	63
2	lcd_csr_set	カーソルタイプ設定	64
3	lcd_csr_put	カーソル位置設定	65
4	lcd_csr_get	カーソル位置読出し	66
5	lcd_char	１文字表示	67
6	lcd_string	文字列表示	68
7	lcd_string2	文字列表示２（スクロール抑制）	69
8	lcd_userstr	ユーザ文字列表示	70
9	lcd_line	直線描画	71
１０	lcd_gaiji	外字フォント登録	72
１１	lcd_usrfont	ユーザーフォントファイル登録	73
１２	lcd_romfont	ＲＯＭフォント設定	74
１３	lcd_led	ＬＥＤの制御	75
１４	lcd_el	ＥＬバックライトの制御	76
１５	lcd_shiftsymbol	Ｓアイコンの表示／非表示	77

1.3.3 キー部関数

NO	関　数　名	機　能	ページ
1	キー入力ファンクション		
	key_read	１文字入力	90
	key_string	文字列入力	91
	key_num	数値入力	92
	key_check	キーバッファのステータスチェック	93
	key_clear	キーバッファのクリア	94
2	ファンクションキー制御		
	key_fnc	ファンクションキーコードの設定	95
	key_fnc_mode	ファンクションキー通知モード設定	96
3	入力設定		
	key_select	キー入力有効／無効設定	97
4	キー入力モード設定		
	key_setInputMode	キー入力モード設定	98

1.3.4 ＯＢＲ部関数

NO	関　数　名	機　　能	ページ
1	OBR_open	ＯＢＲオープン	109
2	OBR_close	ＯＢＲクローズ	110
3	OBR_getc	ＯＢＲデータ１文字リード	111
4	OBR_gets	ＯＢＲデータ文字列リード	112
5	OBR_flush	ＯＢＲバッファのクリア	113
6	OBR_stat	ＯＢＲバッファステータスチェック	114
7	OBR_moderd	ＯＢＲ動作モードの取得	115
8	OBR_modewt	ＯＢＲ動作モードの設定	116
9	OBR_gain	ＯＢＲ読み取り分解能指定	119
１０	OBR_chgbuf	ＯＢＲバッファの切替え	120
１１	OBR_trigmode	トリガーキーによる電源オン設定	121

1.3.5 通信部関数

NO	関　数　名	機　　能	ページ
1	c_open	ＣＯＭのオープン	152
2	c_close	ＣＯＭのクローズ	154
3	c_status	ＣＯＭステータスリード	155
4	c_hold	ＣＯＭの占有	157
5	c_chkopen	ＣＯＭのオープンチェック	158
6	c_dout	ｎ文字送信	159
7	c_din	１文字受信	160
8	c_tmdin	タイムアウト監視受信	161
9	c_out	１文字送信	162
１０	c_break	ブレイク送出の ON/OFF	163
１１	c_txrx	送受信の有効／無効	164
１２	c_flush	受信バッファのクリア	165
１３	c_bfsts	受信バッファステータスのリード	166
１４	c_errbfring	エラーコードバッファリング制御の設定	167
１５	c_rderrsts	標準エラーステータスのリード	168
１６	c_chghdr	受信ハンドラ切替え	169
１７	c_timer	DR/CS/CD タイムアウト監視値の設定	170
１８	c_er	ER 信号の ON/OFF	171
１９	c_rs	RS 信号の ON/OFF	172
２０	c_errs	ER/RS 信号の ON/OFF	173
２１	c_brkevent	ブレイク要因の設定	174

1.3.6 IrDA 部関数

NO	関　数　名	機　　能	ページ
1	Ir_Open	IrCOMM オープン	179
2	Ir_Close	IrCOMM クローズ	180
3	Ir_Read	データ読込み	181
4	Ir_Write	データ書込み	182
5	Ir_QueryTx	送信データ数問合せ	183
6	Ir_QueryRx	受信データ数問合せ	184
7	Ir_Err_Get	エラーステータス取得	185
8	Ir_State_Set	通信状態設定	190
9	Ir_SetPortConfig	自局能力設定	192
１０	Ir_Init	IrCOMM 強制終了	194

1.3.7 通信ユーティリティ部関数

(1)HIOWIN 手順

NO	関　数　名	機　　能	ページ
1	HIO_PortOpen	通信ポートのオープン	197
2	HIO_PortClose	通信ポートのクローズ	198
3	HIO_ReceiveFile	IOBOX からファイルを受信	199
4	HIO_SendFile	IOBOX へファイルを送信	200
5	HIO_DeleteFile	IOBOX のファイルを削除	201
6	HIO_MemoryFormat	IOBOX のメモリ初期化	202
7	HIO_GetSysInfo	IOBOX のシステム情報の取得	203
8	HIO_GetMemoryInfo	IOBOX のメモリ情報の取得	204
9	HIO_GetFileInfo	IOBOX の全ファイル情報の取得	205
１０	HIO_GetLastErrState	最後に発生したエラーの詳細を取得	206
１１	HIO_GetLogData	IOBOX のログデータの取得	208
１２	HIO_ClearLogData	IOBOX のログデータの削除	209
１３	HIO_SetMemoryThresh	IOBOX のキャッシュメモリの上限設定	210
１４	HIO_SetIOBOXID	IOBOX の ID を設定	211
１５	HIO_WriteFirmware	IOBOX のファームウェアの書き換え	212
１６	HIO_ResetIOBOX	IOBOX のリセット	213
１７	HIO_SetDispParam	表示パラメータの設定	214

(2)FLINK 手順

NO	関　数　名	機　　能	ページ
1	cu_open	通信ポートの初期化	217
2	cu_fileSend	ファイル送信	218
3	cu_fileAdd	ファイル追加	219
4	cu_fileRecv	ファイル受信	220
5	cu_close	通信ポートのクローズ	221
6	cu_readErrStat	エラー詳細情報の取得	222
7	cu_idle	ＩＤＬＥ遷移	225
8	cu_cmdRecv	ＰＣモードコマンド待ち	226
9	cu_stopKeySet	中断キーの登録／削除	227
１０	cu_fileDelete	ファイル／ディレクトリ削除	228
１１	cu_fileMove	ファイル移動	229
１２	cu_makeDir	ディレクトリ作成	230
１３	cu_getFileInfo	ファイル情報の取得	231
１４	cu_setFileInfo	ファイル情報の更新	232
１５	cu_getDiskInfo	ディスク情報の取得	233
１６	cu_dateTime	日付時刻の取得／設定	234
１７	cu_getSysInfo	システム情報の取得	235
１８	cu_msgSend	画面表示メッセージの送信	236
１９	cu_beep	ブザー鳴動	237
２０	cu_fchklog_Create	ファイルチェックリスト生成	238
２１	cu_fchklog_Check	ファイルチェックリスト検査	239

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	243
	s_timerend	タイマ１削除	244
	s_settimer2	タイマ２登録	245
	s_timerend2	タイマ２削除	246
2	音発生		
	s_beep	エラービープ音	247
	s_sound	サウンド音	248
3	日時設定		
	s_dateset	日付の設定	249
	s_dateget	日付の取得	250
	s_timeset	時刻の設定	251
	s_timeget	時刻の取得	252

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	256
2	pwr_off	電源オフ	257
3	pwr_ioboxBootMode	IOBOX 起動設定／解除	258

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	266
2	clr_flg	通知フラグ状態クリア	267
3	wai_flg	フラグセット待ち	268
4	pwr_inhabit	電源通知モード設定	269
5	pwr_inhabit_clr	電源通知イベントのクリア	270

1.3.11 共通関数

NO	関　数　名	機　　能	ページ
1	Abort	ＡＢＯＲＴ処理	273
2	exit	ＥＸＩＴ処理	274
3	wkup_cost	動作環境メニューの起動	275
4	sub_reset	ソフトウェアリセット処理	276

1.4 データタイプ

本システムで使用するパラメータのデータタイプとサイズを以下に示します。

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

2 データ管理部

2.1 機能

2.1.1 メモリー管理機能

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

表２．１ サポートしている標準ライブラリ関数の一覧

関　数　名	処　理　概　要
calloc	記憶域を確保し、確保した全ての領域をゼロクリアします
free	指定した記憶域を解放します
malloc	記憶域を確保します
realloc	記憶域の大きさを指定した大きさに変更します

2.1.2 ファイル管理機能

本システムのファイル管理機能は、データファイルに関してのみで、標準入出力ファイル（コンソール、プリンタ、デイスクファイル等）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

(1) Ａドライブ（RAM ドライブ）に関して

本体に搭載されているＲＡＭ空間の内約 1.39ＭＢを、Ａドライブとして使用することができます。このエリアは、MS-DOS 互換の FAT ファイルシステムで構成されており、以下の構造から成っています。

■ドライブ構成

BPB	FAT	DIR	DATA	CheckSUM

表２．３　Ａドライブ構成

機能項目	内　　容
ファイル数	ルート：１９２個、ディレクトリ配下：無制限（ファイル領域が許す限り）
同時オープン数	１６
ディレクトリ	サポート
総容量	７３８ＫＢ～１．６ＭＢ(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（5）
セクタ／トラック	0
ヘッド番号	0
総セクタ数	１４７６～３２０５(２７８８)

※ＦＡＴファイルシステムを用いた MS-DOS 互換ファイルシステムのためファイル領域内の空間はリニア構造にはならず、また、ファイル領域の直接参照はできません。

(2) Ｂドライブ（Flash DISK ドライブ）に関して

本体に搭載されている FlashROM の内約 1.5ＭＢを、Ｂドライブとして使用することができます。このエリアは、専用のファイルシステムで構成されており、以下の構造から成っています。

■ドライブ構成

BPB	DIR	Length	DATA	CheckSUM	・・・・・	Length	DATA	CheckSUM

表２.４　Ｂドライブ構成

機能項目	内　　容
ファイル数	１６個
同時オープン数	１６
ディレクトリ	ルートのみサポート
総容量	１．４２ＭＢ
セクタサイズ	８１７２
セクタ／クラスタ	1
総セクタ数	１８３

※Ａドライブと同様に、ファイル領域内の空間はリニア構造にはならず、また、ファイル領域の直接参照はできません。

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２．５ 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２．６ システムデータ一覧

システムデータ一覧					
項目		管理データ	内　　容	リセットタイミング	
				リセット立上げ	ｃｏｎｆｉｇファイル反映
電源		APO	APO時間：0～59(分)	○	○
		ABO	ABO時間：10～59(秒)	○	○
		レジューム	ON/OFFの設定	○	○
ＫＥＹ		クリック音	ON/OFF	○	○
表示		フォントMODE	6/8/10(dot)	○	○
		フォント種別	NORMAL/BOLD	○	○
		日本語/英語	日本語/英語	FROM内容反映	○
		コントラスト設定値	0～15	○	○
		コントラスト手動差分	手動設定による現在設定値との差分	○	－
OBR		読取り回数	バーコード連続読取回数：1～9(回)	○	○
		照合回数	読取コード照合回数：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		プロトコル	高速IO	○	○
タイマ		音量	OFF/小/中/大	○	○
システム		機器ID	機器固有のID（6桁の数字）	FROM内容反映	config.id
		代理店ＩＤ	代理店ＩＤ（6桁の英記号）	－	config.pas
		OSバージョン	OSバージョン(6桁)	－	－
		PATCHバージョン	PATCHバージョン(6桁)	－	－
		機器種別	MODEL情報	ＩＯポート反映	－
プロトコル	高速IO	タイムアウト	タイムアウト時間：0～9000（ﾐﾘ秒）	○	○
		リトライ回数	リトライ回数：0～99　(回)	○	○
サイズ		ＡＰ領域サイス	アプリケーション領域サイズ	RAMドライブ情報反映	指定時○

表２．７ 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	
		レジューム	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		プロトコル	LONG(32bit)	3	3	高速IO:3
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バージョン	BYTE x 6	6桁の英数字	ＯＳ依存	
		PATCHバージョン	BYTE x 6	6桁の英数字	ＯＳ依存	ex: V1.00A
		機器種別	LONG(32bit)		機種依存	
プロトコル	高速IO	タイムアウト	LONG(32bit)	0～9000	3000	
		リトライ回数	LONG(32bit)	0～99	5	
サイズ		AP領域サイズ	LONG(32bit)	128～992	336	ＫＢ単位

2.1.4 低水準インタフェース

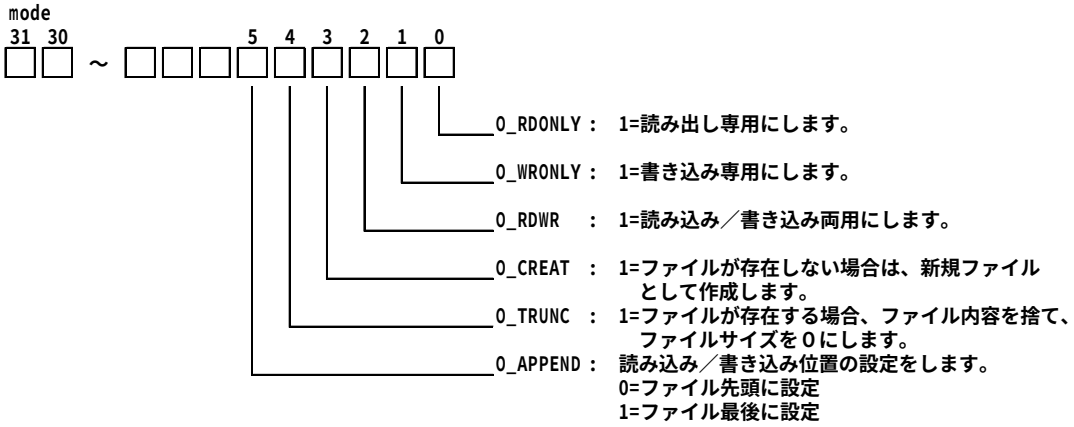
低水準インタフェースでは、ファイルについての処理（状態管理、書込み／読込み等）および、メモリ管理を行います。

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名を引数で指定されたファイルモードでオープンします。正常な場合ハンドル No を返します。ハンドル No は他の低水準関数で指定します。

（注意）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 システムデータファイル

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

（1）ｃｏｎｆｉｇファイル

ファイル形式　　：ＤＯＳファイル
ファイル名　　：ＣＯＮＦＩＧ．ＨＴＳ
ファイルサイズ　：詳細は次ページ参照。
ファイル内容　　：詳細は次ページ参照。
※ＣＯＮＦＩＧ．ＨＴＳはテキスト形式の文字列で構成されており、２０H以下のコードは無視します。

（2）機器ＩＤファイル

ファイル形式　　：ＤＯＳファイル
ファイル名　　：ＣＯＮＦＩＧ．ＩＤ
ファイルサイズ　：６バイト
ファイル内容　　：ＡＳＣＩＩコードによる機器ＩＤ番号（６桁の半角数字）

（3）代理店ＩＤファイル

ファイル形式　　：ＤＯＳファイル
ファイル名　　：ＣＯＮＦＩＧ．ＰＡＳ
ファイルサイズ　：６バイト
ファイル内容　　：ＡＳＣＩＩコードによる代理店ＩＤ番号（６桁の半角英数字）

ＣＯＮＦＩＧ．ＨＴＳ形式

項目			位置	サイズ	設定範囲	既定値
ID			00	10	CONFIG.HTS 固定	CONFIG.HTS
電源		APO 時間	+10	2	00-59(分)	10
		ABO 時間	+12	2	00,10-59(秒)	15
		レジューム	+14	2	00:OFF / 01:ON	01
KEY		クリック音	+16	2	00:OFF / 01:ON	01
OBR		読取回数	+18	2	01-09(回)	01
		照合回数	+20	2	01-09(回)	03
		スキャン時間	+22	2	01-09(秒)	04
表示		MODE	+24	2	00:6dot / 01:8dot / 02:10dot	01
		日/英	+26	2	00:日本語 / 01:英語	00
		種別	+28	2	00:NORMAL / 01:BOLD	00
		コントラスト	+30	2	00-15(段)	07
通信	共通	プロトコル	+32	2	03:高速 IO	03
	個別	速度(IR)	+36	2	08 固定	08
		データ(IR)	+38	2	08 固定	08
		パリティ(IR)	+40	2	00 固定	00
		STOP(IR)	+42	2	00 固定	00
タイマ		音量	+60	2	00:OFF / 01:小 / 02:中 / 03:大	02
プロトコル		下記参照	+62			
サイズ		AP サイズ	+79	5	M0128-M0992(kbyte)	M0336

プロトコル関連：高速ＩＯ

項目	位置	サイズ	設定範囲	既定値
タイムアウト	+62	4	0-9000(ミリ秒)	3000
リトライ回数	+66	4	0-99(回)	0005
リザーブ	+70	6	000000	000000

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（次ページの DAT_DSP_STR を指定時）
　　SYSD_DSP2　　：　　（次ページの DAT_DSP_STR2 を指定時）
　　SYSD_COM　　：通信
　　SYSD_TIM　　：タイマ
　　SYSD_SYS　　：システム（次ページの DAT_SYS_STR を指定時）
　　SYSD_SYS2　　：　　（次ページの DAT_SYS_STR2 を指定時）
　　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２－７ 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／英語判定      */
                              }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      */
                              }DAT_DSP_STR2;
```

【通信関連】

```
typedef struct    sys_tty0    {
                              w                 com_proto;    /* プロトコル種別        */
                              w                 com_port      /* RESERVE               */
                              }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                          */
                              UB    bios_ver[7];        /* BIOS バージョン READ ONLY        */
                              UW    mac_type;           /* 機器種別 READ ONLY               */
                              }DAT_SYS_STR;

typedef struct     sys_dat2   {
                              UB    dir_id[7];          /* 代理店 ID WRITE ONLY             */
                                                        /* 直接参照できません               */
                              UB    patch_ver[7];       /* PATCH バージョン READ ONLY       */
                              }DAT_SYS_STR2;
```

【プロトコル関連】

```
typedef struct     sys_pro    {     /* RESERVE          */
                              w     non_rec_tmount;     /* RESERVE                          */
                              w     non_retry_ct;       /* RESERVE                          */
                              w     mal_rec_tmout;      /* RESERVE                          */
                              w     ptp_snd_tmount;     /* RESERVE                          */
                              w     ptp_rec_tmout;      /* RESERVE                          */
                              w     ptp_rec_retry_ct;   /* RESERVE                          */
                                    /* NEW PALAN        */
                              w     irda_tmout;         /*IrDA セッション確立タイムアウト時間   */
                              w     irda_rec_tmout;     /*IrDA 受信タイムアウト時間
                              w     dr_tmout;           /*IrDA セッション終了タイムアウト時間   */
                                    /* 高速 IO          */
                              w     hio_tmout;          /* タイムアウト                     */
                              w     hio_retry_ct;       /* リトライ回数                     */
                                    /* RESERVE          */
                              w     sirial_no           /* RESERVE                          */
                              w     level_parity;       /* RESERVE                          */
                              w     bht_tmout;          /* RESERVE                          */
                              }DAT_PRO_STR;
```

【AP サイズ関連】

```
typedef struct     sys_mem    {
                              w     AP_size;            /* アプリケーションサイズ           */
                              }DAT_MEM_STR;
```

【ファイルシステムモード関連】

```
typedef struct     sys_file   {
                              w     file_mode;          /* RESERVE                          */
                              }DAT_FIL_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(const UB  *dealer_no ) ;
```

【パラメータ】

```
const  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(const B *path );
```

【パラメータ】

const　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１６で、オープンに失敗した場合は E_LOWERR 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 = 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	：読込み専用
O_WRONLY	：書込み専用
O_RDWR	：読込み／書込み両用
O_CREAT	：ファイル新規作成
O_TRUNC	：指定ファイルの内容を捨て、サイズを０にします
O_APPEND	：次に読み書きを行うファイル内の位置を最後に設定します ※O_APPEND を設定しない場合の次に読み書きを行うファイル内の位置は先頭になります。

※上のファイルモードは以下の組み合わせで OR 指定して使用してください。

設定可能ファイルモード
※下記以外はエラーとなります

標準関数	O_RDONLY	O_WRONLY	O_RDWR	O_CREAT	O_TRUNC	O_APPEND
r b	○					
w b		○		○	○	
a b		○		○		○
r＋b			○			
w＋b			○	○	○	
a＋b			○	○		○

【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

【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

以下の形式の指定が可能です。

（形式１） ｎｎｎｎｎｎｎｎ．mmm
ファイル名　　拡張子

（形式２） ｄ：\ ｐｐｐｐｐｐｐｐ \ ｎｎｎｎｎｎｎｎ．mmm
ドライブ名：パス名\　ファイル名．拡張子

注　ファイル名の有効データ（ＡＮＫコードのみ）
ファイル名の先頭コードは８０Ｈ以上にしないで下さい。

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。ただしリターンパラメータには実際に書き込んだデータ数を返します。
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。
更新した読み込み／書き込み位置が負になったり，ファイルサイズを越える場合は
現在位置の変更をせずに現状ポインタ値を返します。

≪C 言語インタフェース≫

【コーリングシーケンス】

```
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	：ファイルの基準位置
0	：ファイルの先頭
1	：現在の読込み／書込み位置
2	：ファイルの最後

【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） 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。
割り付けるメモリ領域が不足，または要求サイズが０の場合は，エラーとします。
正常終了の場合は，割り付けたメモリの先頭アドレスを，異常終了の場合は，E_LOWERR
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】

```
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	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（バイト単位）

【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

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。
Ｂドライブは対象外です。

≪Ｃ言語インタフェース≫

【コーリングシーケンス】

```
ER  ercd = fil_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￥(スペース)￥ＡＢＣ
・先頭コードが８０ｈ以上のディレクトリ・ファイル名は、作成しないで下さい。
サーチ系の関数でサーチできなくなります。

上記注意事項は、本システムのファイル系関数全般に該当します。

機能	ディレクトリの削除	関数名	fil_rmdir

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。
Ｂドライブは対象外です。

≪Ｃ言語インタフェース≫

【コーリングシーケンス】

```
ER  ercd = fil_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：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：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。

≪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, FIL_FSTAT  *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

	15	11	10	5	4	0	
時刻	時（0～２３）		分（0～５９）		秒（0～２９）		秒は２秒単位

	15	9	8	5	4	0	
日付	年（0～９９）		月（1～１２）		日（1～３１）		年は0を 1980 年とします

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 = 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      ：正常終了時は空き領域サイズ（バイト単位）を返します
              ：エラー終了時はリターンコードを返します
```

【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。

≪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,  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≫

時刻と日付のフォーマット

	15 … 11	10 … 5	4 … 0	
時刻	時（０～２３）	分（０～５９）	秒（０～２９）	秒は２秒単位

	15 … 9	8 … 5	4 … 0	
日付	年（０～９９）	月（１～１２）	日（１～３１）	年は０を 1980 年とします

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(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】

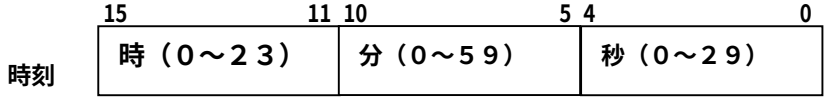
ER ercd ：リターンコード

【リターンコード】

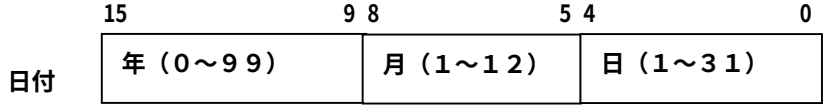
E_OK	：正常終了
E_NG	：異常終了
E_PRM	：パラメータエラー

≪備考≫

時刻と日付のフォーマット



秒は２秒単位



年は0を 1980 年とします

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,  FIL_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（アプリケーションから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） ファイル名は、英文字は全て大文字になります。(a:\abc→A:\ABC)
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（アプリケーションからの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

3 表示部

3.1 表示制御

3.1.1 表示画面

画面サイズは 112×64 ドットです。用途に応じて２つの座標系があります。

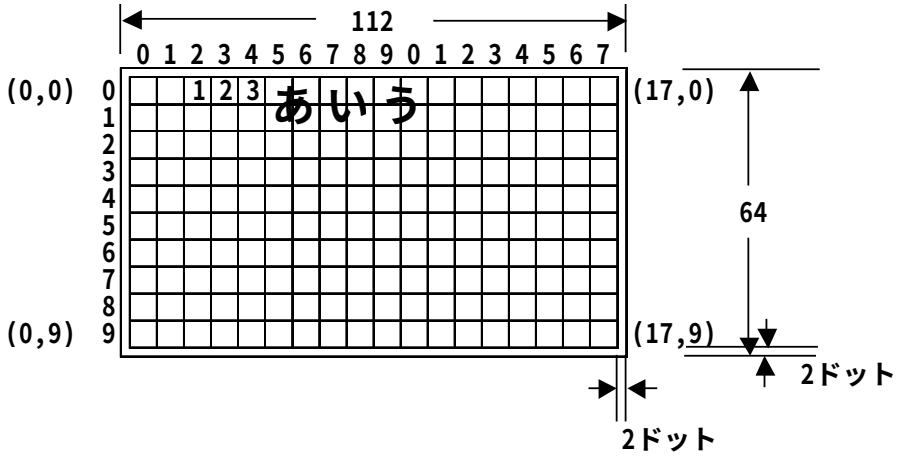
1 つ目はキャラクタ座標系です。キャラクタ座標系では各フォントモードの縮小 ANK を基準に左上を (0,0)であらわします。フォントの表示やカーソル指定に使われます。
2 つ目はグラフィック座標系です。グラフィック座標系は、フォントモードに関係なく、左上(0,0)、右下(111,63)であらわします。線の表示に使われます。

表３．１ 表示行桁数

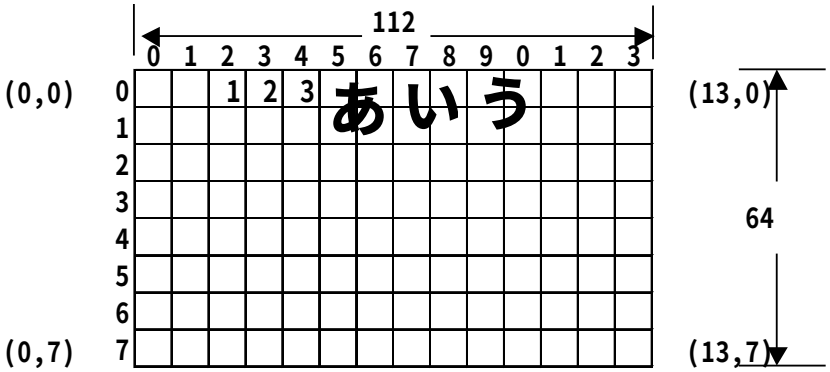
モード	フォント（サイズ）	表示桁数	最大表示文字数
６ドット	縮小ＡＮＫ　（ 6× 6）	１８桁×１０行	１８０文字
	標準ＡＮＫ　（ 6×12）	１８桁×　５行	９０文字
	漢字　（12×12）	９桁×　５行	４５文字
８ドット	縮小ＡＮＫ　（ 8× 8）	１４桁×　８行	１１２文字
	標準ＡＮＫ　（ 8×16）	１４桁×　４行	５６文字
	漢字　（16×16）	７桁×　４行	２８文字
１０ドット	縮小ＡＮＫ　（10×10）	１１桁×　６行	６６文字
	標準ＡＮＫ　（10×20）	１１桁×　３行	３３文字
	漢字　（20×20）	５桁×　３行	１５文字

※各フォントモードでのキャラクタ座標系をあらわしています。

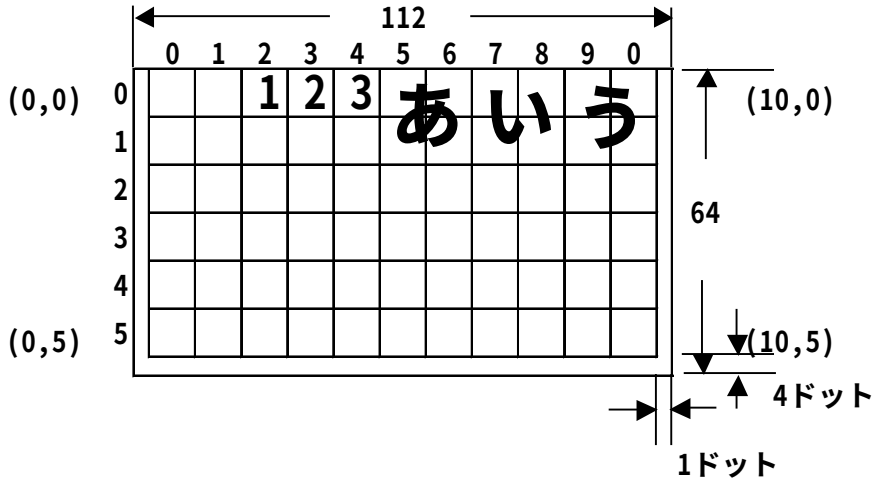
[6 ドットモード]



[8 ドットモード]



[10 ドットモード]



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			0	@	P	`	p				-	タ	ミ		
1			!	1	A	Q	a	q			。	ア	チ	ム		
2			”	2	B	R	b	r			「	イ	ツ	メ		
3			#	3	C	S	c	s			」	ウ	テ	モ		
4			$	4	D	T	d	t			、	エ	ト	ヤ		
5			%	5	E	U	e	u			・	オ	ナ	ユ		
6			&	6	F	V	f	v			ヲ	カ	ニ	ヨ		
7			’	7	G	W	g	w			ァ	キ	ヌ	ラ		
8			(	8	H	X	h	x			ィ	ク	ネ	リ		
9			)	9	I	Y	i	y			ゥ	ケ	ノ	ル		
A	LF		*	:	J	Z	j	z			ェ	コ	ハ	レ		
B			+	;	K	[	k	{			ォ	サ	ヒ	ロ		
C			,	<	L	\	l	|			ャ	シ	フ	ワ		
D	CR		-	=	M	]	m	}			ュ	ス	ヘ	ン		
E			.	>	N	^	n	~			ョ	セ	ホ	゛		
F			/	?	O	_	o				ッ	ソ	マ	゜		

網掛け部分が文字として表示されます。

※ ‘￥’コードの表示
5Ch を表示する場合、日本語モード時は‘￥’を英語モード時は‘＼’を表示します。
日本語／英語モードは、「動作環境メニュー」または、システムデータ管理提供の関数で変更します。
バックスラッシュのフォントデータ（ビットマップ）は存在しないため、表示関数で独自に持っているデータを使用します。

（2）漢字／外字コード

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

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

※　２バイト目が、7F のコード(例:0xEB7F)は存在しません。

表３．３ コード表（２バイトコード）

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

指定コード		表示される文字	
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	制御コード	（3.1.3 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）

指定コード		表示される文字	
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		制御コード　（3.1.3 制御コード表示参照）	
1B		ＥＳＣ制御　（3.1.4 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

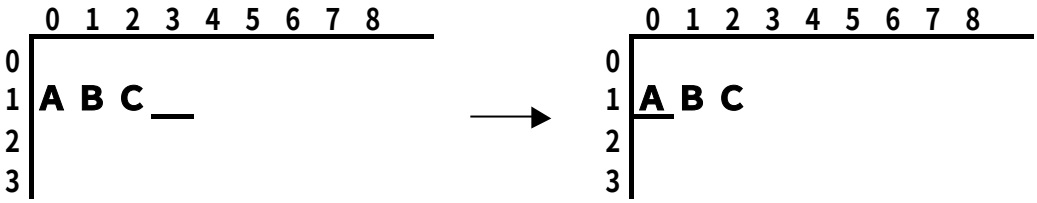
※ ユーザー文字列表示（lcd_userstr）の場合は、漢字の表示はありません。全て１バイト目を ANK コードとしてみます。81-84、88-9F、E0-EB のコードは ROM フォント時は、ANK スペースを、ユーザーフォント時は、ユーザー登録文字をそれぞれ表示します。

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。

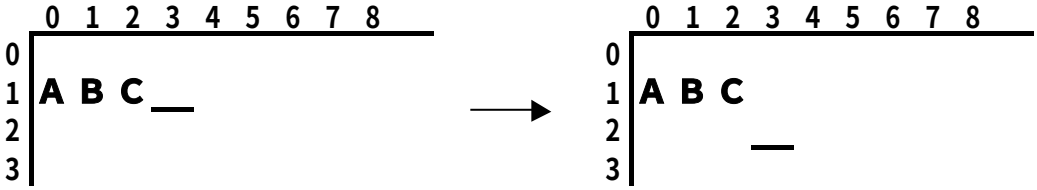
（1）　ＣＲ（０ｘ０ｄ）の表示

・ キャリッジリターン動作を行います

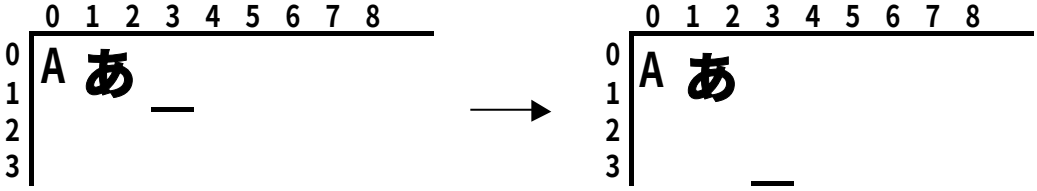


（2）　ＬＦ（０ｘ０ａ）の表示

・ 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。

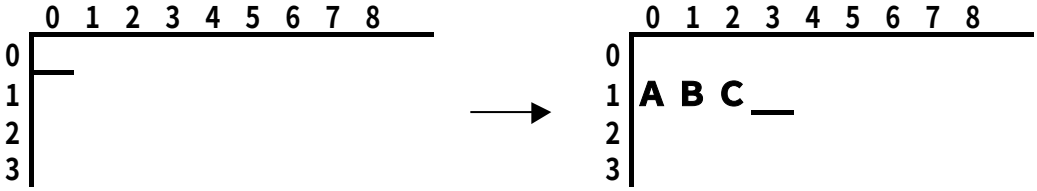


・ 標準ＡＮＫおよび漢字の文字列中にＬＦコードが含まれている場合は、２行改行します。

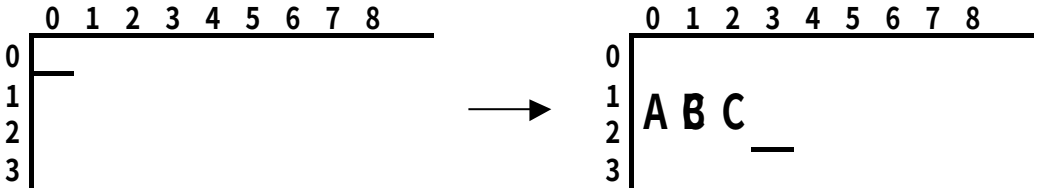


・ 文字列表示の先頭がＬＦコードまたは、１文字表示でＬＦコードが指定された場合は、入力パラメータのＡＮＫモードが縮小の場合は１行改行、標準の場合は２行改行します。

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

文字列表示を行う際、ESC シーケンスを認識した場合、以下の表示動作を行います。

（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 ”

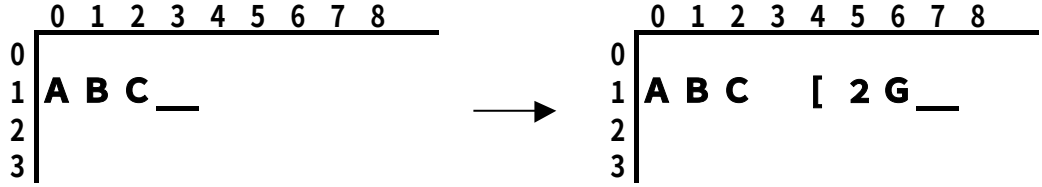
桁
行

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に例外動作が発生した場合は以下のような表示制御をします。

表３．６ 例外動作

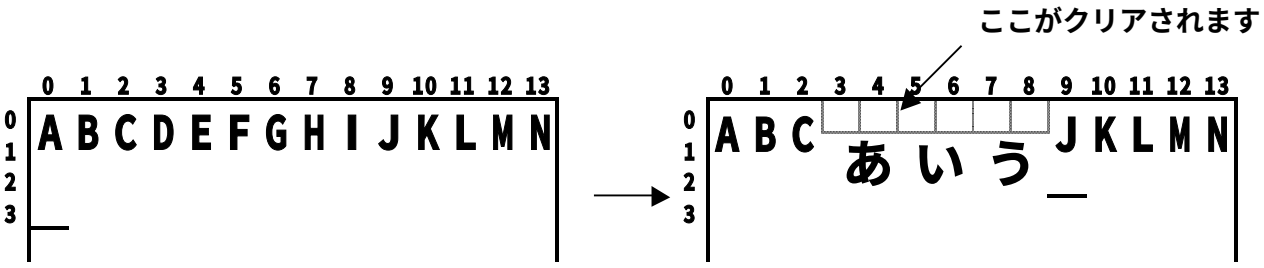
動作	内容
表示データの重複	表示データと重なる部分の文字は、重なった文字全体をスペースでクリアし、その上に新たな文字を表示します
最上位行での標準ＡＮＫ／漢字表示	画面からはみ出すのでその位置にスペースを表示します
行端での自動改行制御	文字列表示を行う時、行端で表示仕切れない場合かつ改行ありモードの時は、自動改行します
行端での漢字表示	行端で切れ端になる場合は、改行モードありの時、自動改行します

（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

※ABC は標準 ANK で、８ドット表示です。

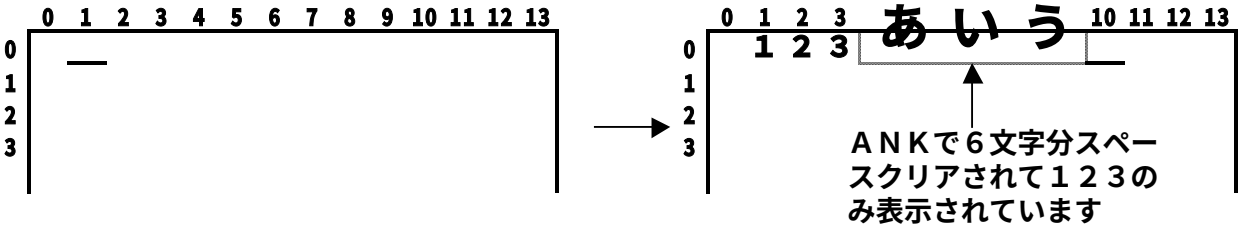


（2）最上位行での標準ＡＮＫ／漢字表示

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

※１２３は縮小 ANK で、８ドット表示です。

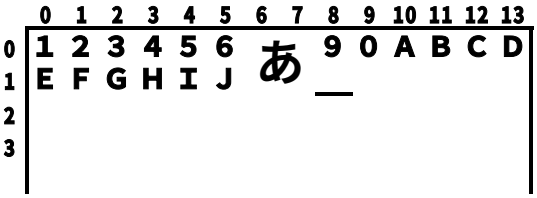


（３）行端での自動改行制御

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）

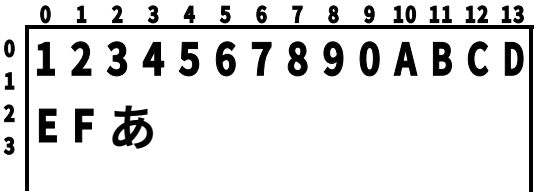
（例１）先頭文字が縮小ＡＮＫの場合、座標（０，０）に“１２３４５６７８９０ＡＢＣＤＥＦＧＨＩＪあ”を表示

※１２３…はあ縮小 ANK で、８ドット表示です。



（例２）先頭文字が標準ＡＮＫ／漢字の場合、座標（０，１）に“１２３４５６７８９０ＡＢＣＤＥＦあ”を表示

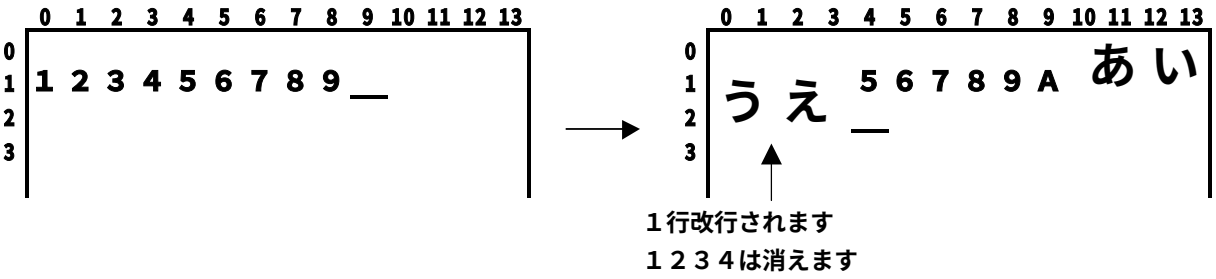
※１２３…は標準 ANK で８ドット表示です。



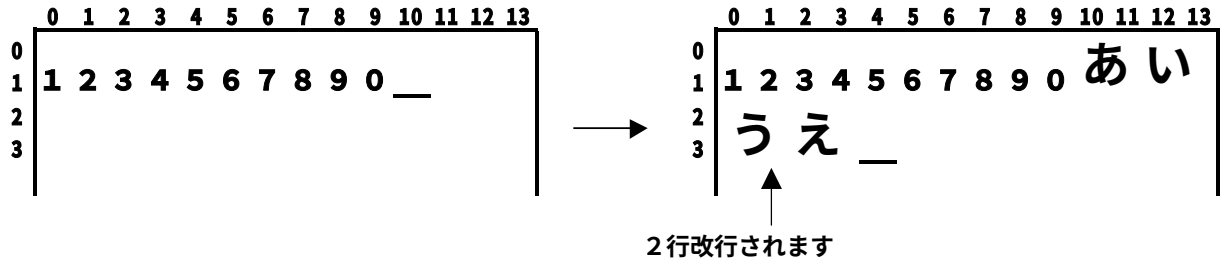
（４）行端での漢字表示制御

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（９、１）に“Ａあいうえ”を表示



（例２） 先頭文字が標準ＡＮＫ／漢字の場合、座標（１０、１）に“ あいうえ ”を表示

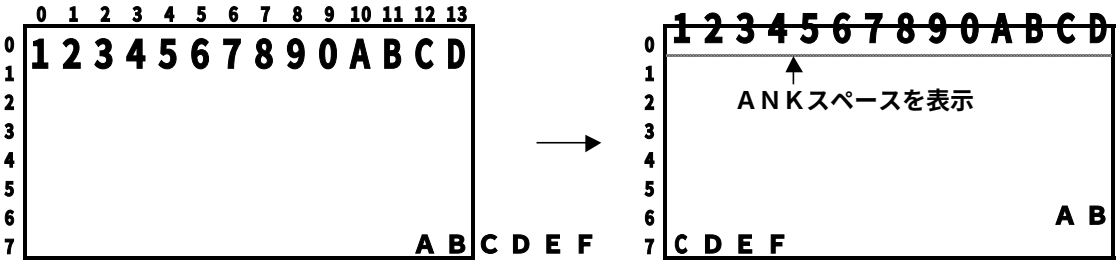


3.1.6　スクロール制御

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

（例１）　最下行の縮小ＡＮＫ表示でスクロールが発生する場合

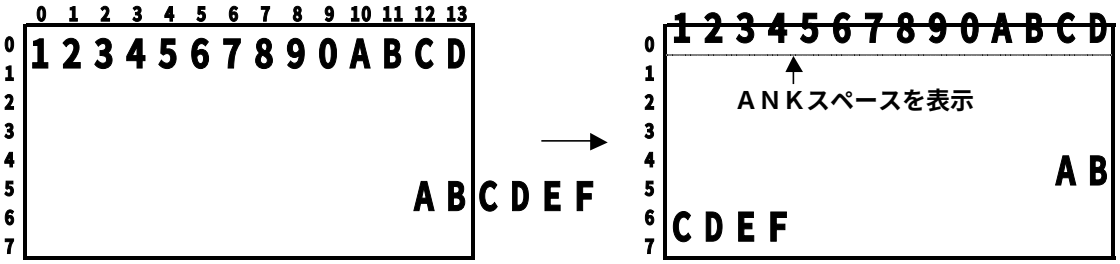
座標（１２，７）に縮小ＡＮＫ“ＡＢＣＤＥＦ”を表示



※１行スクロールします

（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

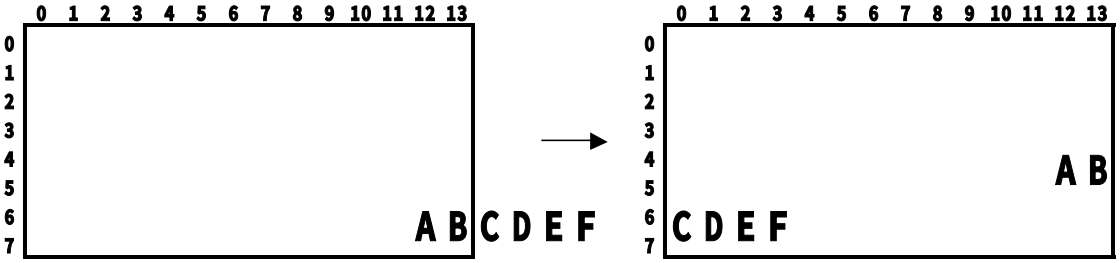
座標（１２，６）に標準ＡＮＫ“ＡＢＣＤＥＦ”を表示



※１行スクロールします

（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（１２，７）に標準ＡＮＫ“ＡＢＣＤＥＦ”を表示



※２行スクロールします

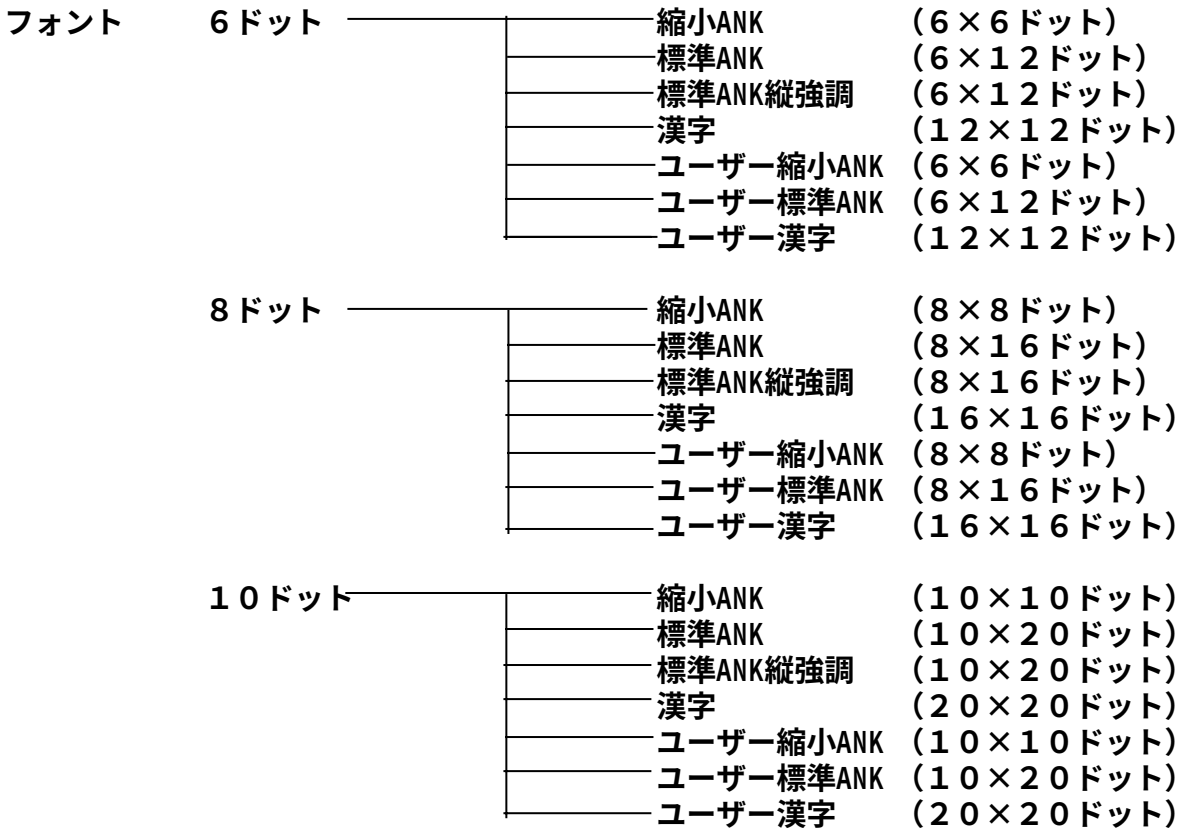
※スクロール抑制

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。以下のフォント種類は本機にフォントファイルがあるものを示します。

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。

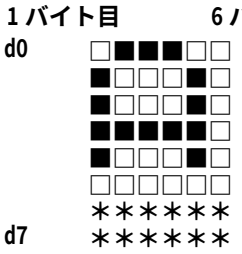
※　ユーザーフォントの表示を行う場合は、「ユーザーフォントファイルの登録」を、ＲＯＭフォントへ戻す場合は、「ＲＯＭフォント設定」を行って下さい。これによりユーザー／ＲＯＭフォントの混在表示が可能です。

※　ユーザーフォント表示の場合は、縦強調ファイルがないため（上のフォントの種類を参照）縦強調指定は無効になります。ただし、修飾文字の強調は指定できます。

3.2.2 フォントデータ構成

６ドットモードのフォント

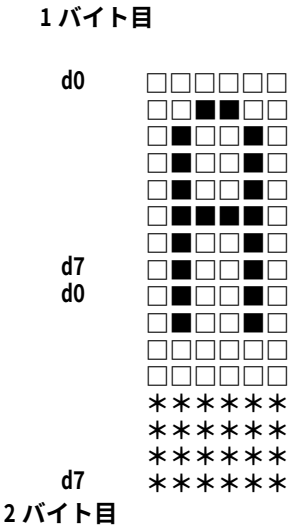
①　縮小ＡＮＫデータ構成（６×６ドット）



１フォント６バイト構造

1E	09	09	09	1E	00	
1	2	3	4	5	6	（バイト目）

②標準ＡＮＫデータ構成（６×１２ドット）

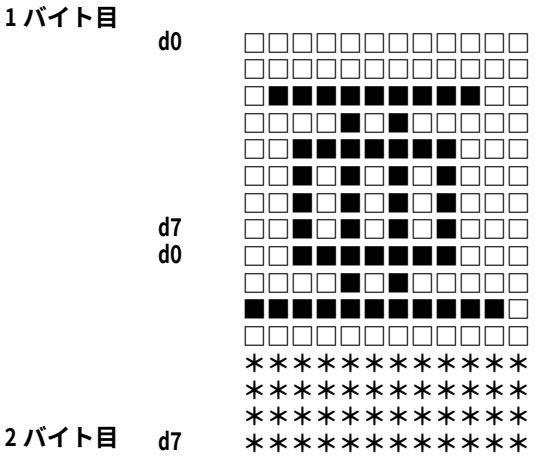


１フォント１２バイト構造

00	00	FC	03	22	00	
1	2	3	4	5	6	（バイト目）

22	00	FC	03	00	00	
7	8	9	10	11	12	（バイト目）

③標準漢字データ構成（１２×１２ドット）



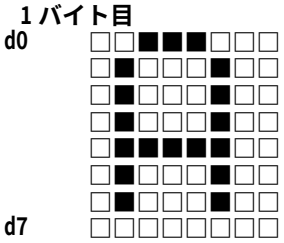
１フォント２４バイト構造

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

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

（2）８ドットモードのフォント

①縮小ＡＮＫデータ構成（８×８ドット）

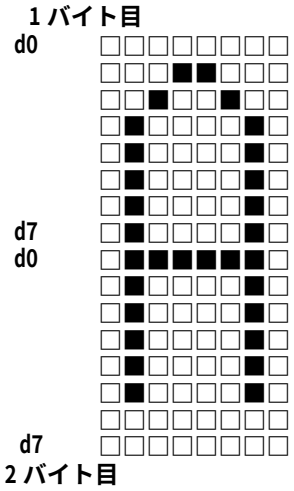


１フォント８バイト構造

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

（バイト目）

②標準ＡＮＫデータ構成（８×１６ドット）



１フォント１６バイト構造

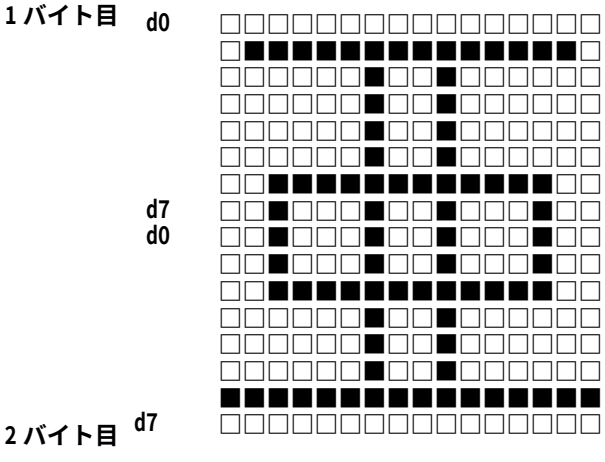
00	00	F8	3F	04	01	02	01
1	2	3	4	5	6	7	8

（バイト目）

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

（バイト目）

③標準漢字データ構成（１６×１６ドット）



１フォント３２バイト構造

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

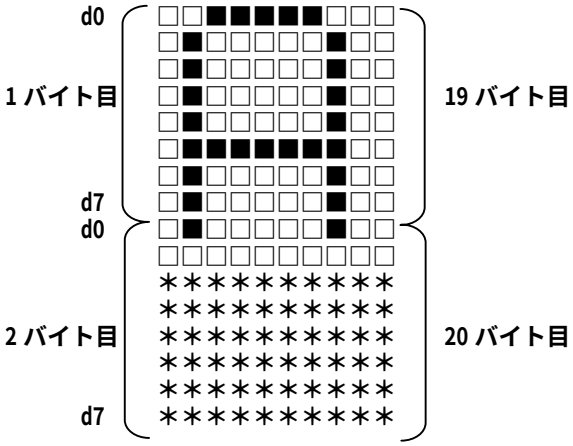
（バイト目）

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

（バイト目）

（3）１０ドットモードのフォント

①縮小ＡＮＫデータ構成（１０×１０ドット）



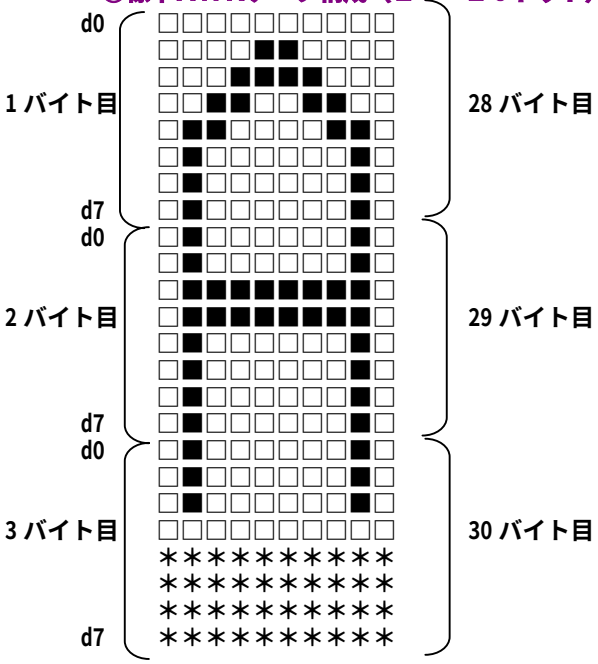
00	00	FE	01	21	00	21	00	21	00
1	2	3	4	5	6	7	8	9	10

（バイト目）

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

（バイト目）

②標準ＡＮＫデータ構成（１０×２０ドット）



１フォント３０バイト構造

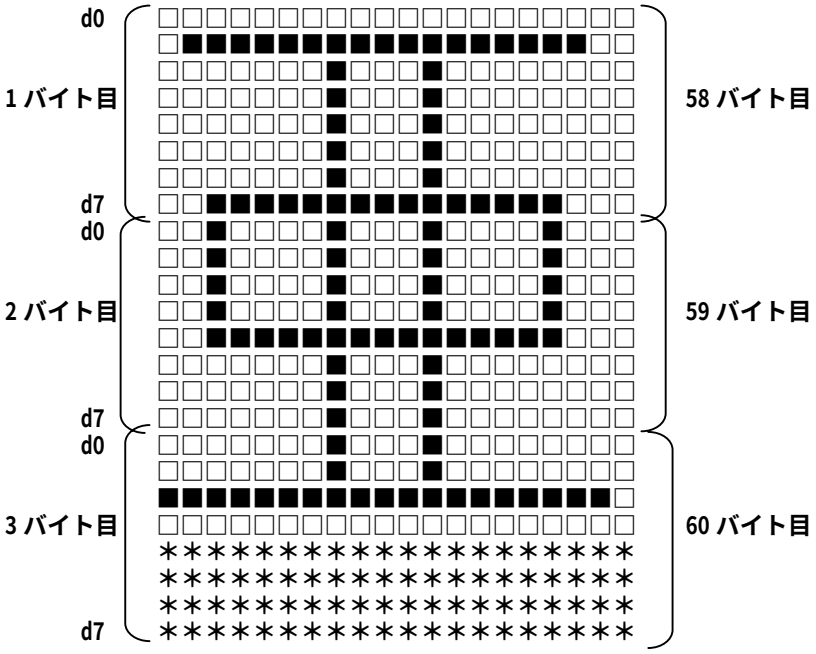
00	00	00	F0	FF	07	18	0C	00	0C	0C	00	06	0C	00	06	0C	00	0C	0C
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

（バイト目）

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

（バイト目）

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

（バイト目）

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

（バイト目）

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

（バイト目）

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

（バイト目）

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

（バイト目）

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

（バイト目）

3.2.3 修飾文字のフォントデータ制御

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

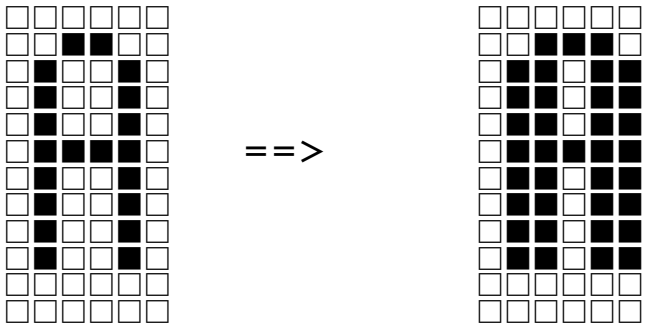
強調　・・・・・・・・・・・・・・・・　右へ１ドットずらしＯＲします。

反転　・・・・・・・・・・・・・・・・　ドットを反転します。

横倍角　・・・・・・・・・・・・・・・・　横方向へ２倍にします。

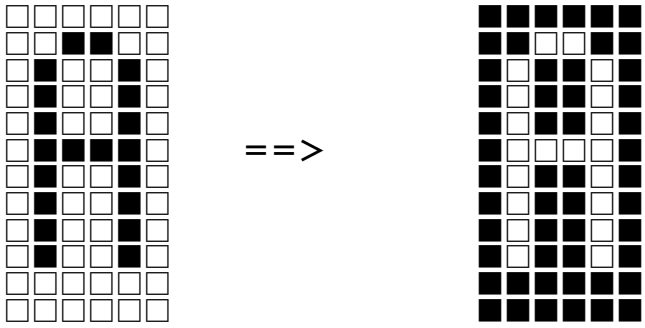
（1） 強調表示

・フォントデータのビットパターンを右へずらして、ＯＲします。



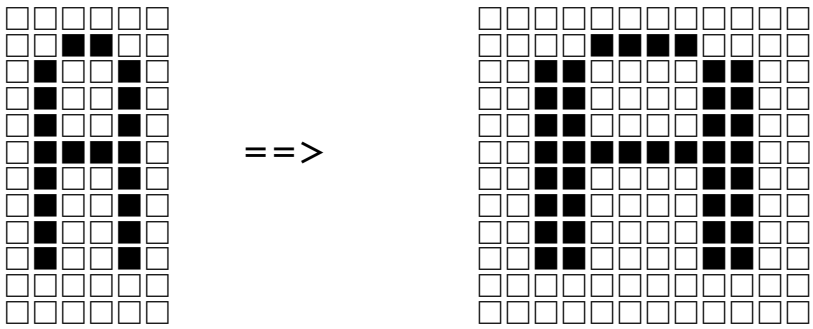
（2） 反転表示

・フォントデータのビットパターンを反転します。



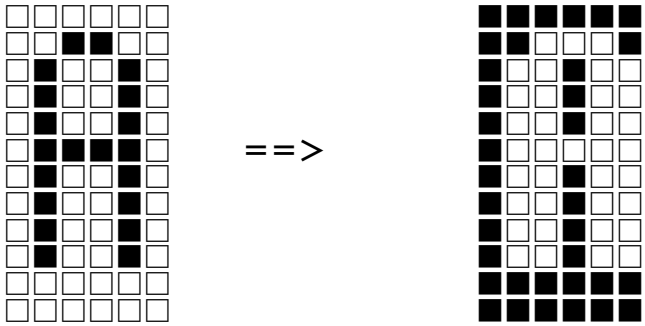
（3） 横倍角表示

・フォントデータのビットパターンを横方向へ２倍にします。



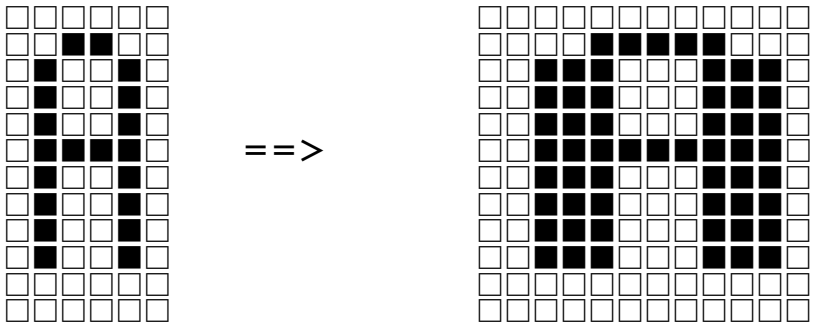
（4） 強調／反転表示

・（1）、（2）両方の処理を合わせます。（強調後反転する）



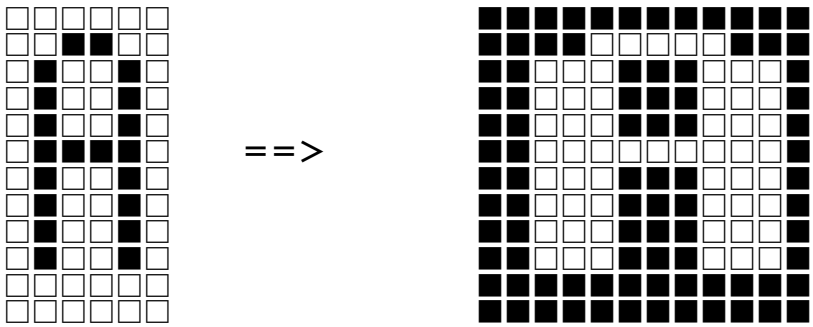
（5） 横倍角／強調表示

・（1）、（3）両方の処理を合わせます。（横方向へ２倍してから強調する）



（6） 横倍角／強調／反転表示

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



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

表３．７　フォントファイルの種類

フォントファイル種別	フォント種別	容量
外字フォントファイル	6 ドットフォント	24 バイト× 128 文字 ＝ 3,072 バイト
	8 ドットフォント	32 バイト× 128 文字 ＝ 4,096 バイト
	10 ドットフォント	60 バイト× 128 文字 ＝ 7,680 バイト
ユーザーフォント	6 ドット縮小 ANK フォント	6 バイト× 256 文字 ＝ 1,536 バイト
ファイル	6 ドット標準 ANK フォント	12 バイト× 256 文字 ＝ 3,072 バイト
	6 ドット漢字フォント	24 バイト×7,393 文字＝177,432 バイト
	8 ドット縮小 ANK フォント	8 バイト× 256 文字 ＝ 2,048 バイト
	8 ドット標準 ANK フォント	16 バイト× 256 文字 ＝ 4,096 バイト
	8 ドット漢字フォント	32 バイト×7,393 文字＝236,576 バイト
	10 ドット縮小 ANK フォント	20 バイト× 256 文字 ＝ 5,120 バイト
	10 ドット標準 ANK フォント	30 バイト× 256 文字 ＝ 7,680 バイト
	10 ドット漢字フォント	60 バイト×7,393 文字＝443,580 バイト

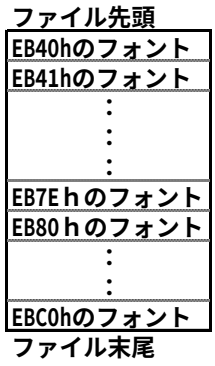
（2） フォントデータ構成

フォントデータの構成は ROM フォントと同一です。「3.2.2 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。



EB7Fhはありません。
詰めて作成して下さい。

・ANK フォントファイル構成

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

ファイル先頭
00hのフォント
01hのフォント
：
：
：
：
FFｈのフォント
ファイル末尾

・漢字フォントファイル構成

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

ファイル先頭	
8140hのフォント	
：	XX00h～XX3Fhおよび
：	8840h～889Ehのフォント
84FFhのフォント	イメージは入れません。
889Fｈのフォント	詰めて作成してください。
：	
：	XX7Fh、XXFDh、XXFEh、XXFFh
9FFFhのフォント	は指定しても表示されませんが
E040hのフォント	ダミーデータを入れておいて
：	下さい。
：	
EAFFhのフォント	
ファイル末尾	

※ 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 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０～１７桁を指定
                    ８ドットモード時０～１３桁を指定
                   １０ドットモード時０～１０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０～１７を格納
                      ８ドットモード時は０～１３を格納
                     １０ドットモード時は０～１０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。	
UH 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 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ＡＮＫ１０２４文字／漢字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。
③ESC 文字列表示
　ESC 文字の動作が可能です。
※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
                   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。

※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。
グラフィック座標系（横１１２、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の場合は表示、オフ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０～１１１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０～１１１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

ユーザフォントを表示するために、ユーザーフォントファイル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
                   縮小ＡＮＫ６ドットフォント      ：LCD_AL6_FILE
                   標準ＡＮＫ６ドットフォント      ：LCD_AS6_FILE
                   漢字６ドットフォント            ：LCD_K6_FILE
                   縮小ＡＮＫ８ドットフォント      ：LCD_AL8_FILE
                   標準ＡＮＫ８ドットフォント      ：LCD_AS8_FILE
                   漢字８ドットフォント            ：LCD_K8_FILE
                   縮小ＡＮＫ１０ドットフォント    ：LCD_AL10_FILE
                   標準ＡＮＫ１０ドットフォント    ：LCD_AS10_FILE
                   漢字１０ドットフォント          ：LCD_K10_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。
例　）ファイル種別　　LCD_AL10_FILE
　　フォントモード　8 ドット
　※表示されるのは 8 ドットの ROM フォントで、10 ドットに切り替えるとユーザーフォントが表示されます。
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、緑点灯または赤点灯（LCD_LED_GREEN または LCD_LED_RED）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≫

機能	Ｓアイコン ＯＮ／ＯＦＦ設定	関数名	lcd_shiftsymbol

Ｓ　アイコン（キー入力モードシンボル）の点灯／消灯をします。

≪Ｃ言語インタフェース≫

【コーリングシーケンス】

```
ER    ercd=lcd_shiftsymbol(UB mode);
```

【パラメータ】

```
UB    mode   ：点灯／消灯モード
             LCD_S_ON     ：点灯
             LCD_S_OFF    ：消灯
```

【リターンパラメータ】

```
ER    erced  ：リターンコード
```

【リターンコード】

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

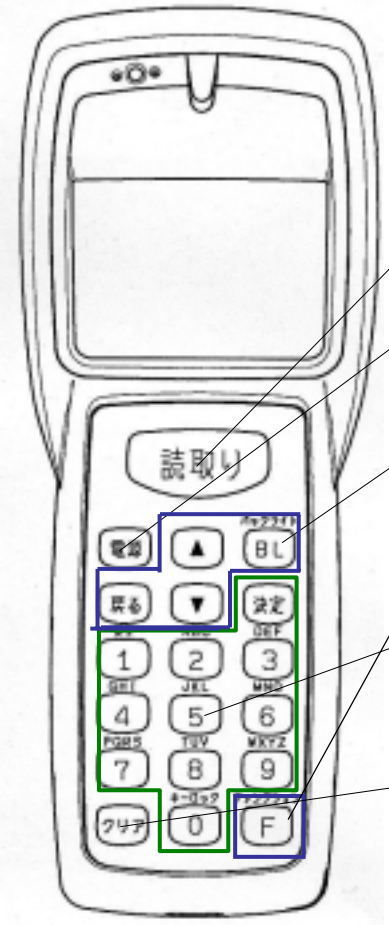
≪備考≫

4 キー部

4.1 機能

4.1.1 キー構成

本機のキー構成は以下のようになっています。



■読取りキー
バーコード読取り開始

■電源キー
本体の電源ｏｎ／ｏｆｆ

■ファンクションキー
任意の機能設定可能　※１：　参照

■テンキー

■クリアキー

※１　ファンクションキーはデフォルトでは各キーに対して以下の機能で動作します。ただし、任意の機能を設定することができます。そのため、解説書では F1～F５の名称でこれらのキーを呼びます。

表４．１　デフォルト機能と名称

キー	デフォルト機能	機能割付け時の名称
▲	コントラスト UP	Ｆ１
バックライト	バックライト ON／OFF	Ｆ２
戻る	バックスペース	Ｆ３
▼	コントラスト DOWN	Ｆ４
F	スペース	Ｆ５

4.1.2 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。
※文字入力は関数(key_setInputMode)をコールすることで遷移します。

(1)数値入力モード

０～９の数値、‘ SP’、入力の確定キーの入力が可能です。

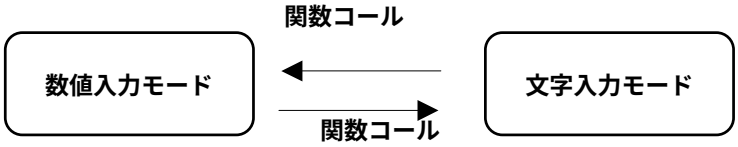
(2)文字入力モード

英字（Ａ～Ｚ）、数値（０～９）の入力が可能です。英字は、めくり文字になっています。

例）　ABC

2　・・・・・・・　キーを押すたびに「Ａ → Ｂ → Ｃ」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。



4.1.3 １文字入力

アプリケーション指定位置に１文字入力を行うものです。アスキーコードが確定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。

4.1.4 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、文字列入力を行うものです。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で戻ります。
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４．２　文字列入力編集処理

名　称	デフォルトキー	コード		機　　能	動作例	
		属性	コード		入力前	入力後
←	なし ※1	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
→	なし ※1	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
戻る	戻る	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	なし ※1	00h	10h	カーソル上の文字を削除します。	1234567890 123 1234567890 1234567890 1234567890 1234567890 abcd	1234567890 13 1234567890 123456789 1234567890 1234567890 bcd

※1：　この機能を使用する場合はコードをファンクションキーに割り当てて下さい。

4.1.5 数値入力

（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４．３　数値入力編集処理

名　称	デフォルトキー	コード		機　　能	動作例	
		属性	ｺｰﾄﾞ		入力前	入力後
＋ (プラス)	なし ※1	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)	なし ※1	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 -123 1234567890 -1 1234567890 - 1234567890 1234567890
． (小数点)	なし ※1	00h	2Eh	カーソル位置に「.」(ピリオド記号)を 付加します。（入力領域がフルの場合、及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 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
戻る	戻る	00h	08h	カーソル上の文字を削除します。	1234567890 123 1234567890 1 1234567890 12345 1234567890	1234567890 12 1234567890 1234567890 12345 123456789
削除	なし ※1	00h	10h	カーソル上の文字を削除します。	1234567890 123 1234567890 1 1234567890 12345 1234567890	1234567890 12 1234567890 1234567890 123456 123456789

※1：　この機能を使用する場合はコードをファンクションキーに割り当てて下さい。

4.1.6 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４．４ 設定キーコード

グループ	コード値		機　　能	１文字入力 (コード返却)	文字列入力 (文字格納)	数値入力 (文字格納)
	属性	コード				
機能コード	ＦＦｈ	00h	コントラストを１段濃くします	×	×	×
		01h	コントラストを１段淡くします			
		02h	バックライトＯＮ／ＯＦＦ切替			
制御コード	００ｈ	08h	１文字後退	○	×	×
		0Ah	改行			
		0Ch	入力領域のクリア			
		0Dh	復帰			
		10h	１文字削除			
		1Ch	カーソル右移動			
		1Dh	カーソル左移動			
その他(ANK)	００ｈ	XXh	文字（※２）	○	○	数字(0～9) +,-,.のみ

※１　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４．５　設定可能キー一覧

種別	キー	設定可能	設定可能キーコード
ストロークキー	クリア(CLR)	不可	
	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　ENT		
	F1	可能	全て
	F2		
	F3		
	F4		
	F5		
読取りキー	読取りキー	可能	全て

4.1.7 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４．６ 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	クリア(CLR)	不可
	テンキー　1	可能
	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 ENT	
	F1	
	F2	
	F3	
	F4	
	F5	
読取りキー	読取りキー	不可

4.1.8 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。

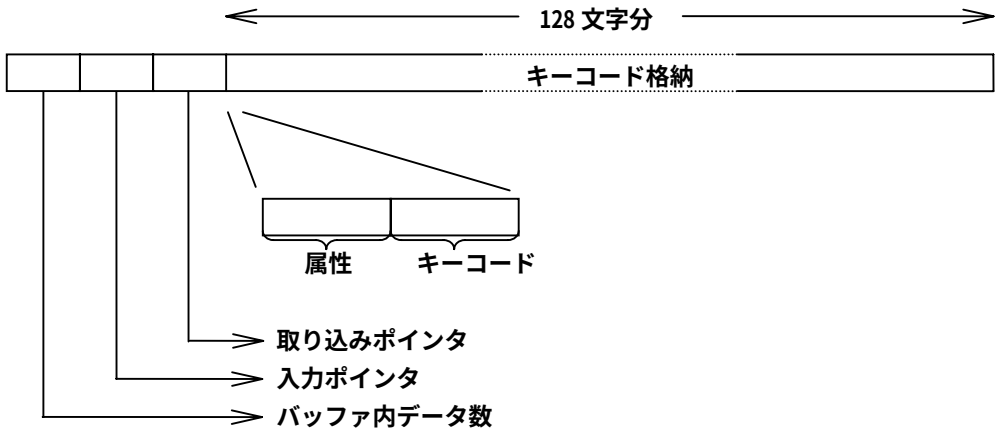
表４．７ キー入力有効／無効設定可能キー一覧

種別	キー	設定可能
ストロークキー	クリア(CLR)	可能
	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　ENT	
	F1(▲)	
	F2(BL)	
	F3(BS)	
	F4(▼)	
	F5(SP)	
読み取りキー	読み取りキー	

※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、キークリック音も鳴りません。

4.1.9 キーバッファ

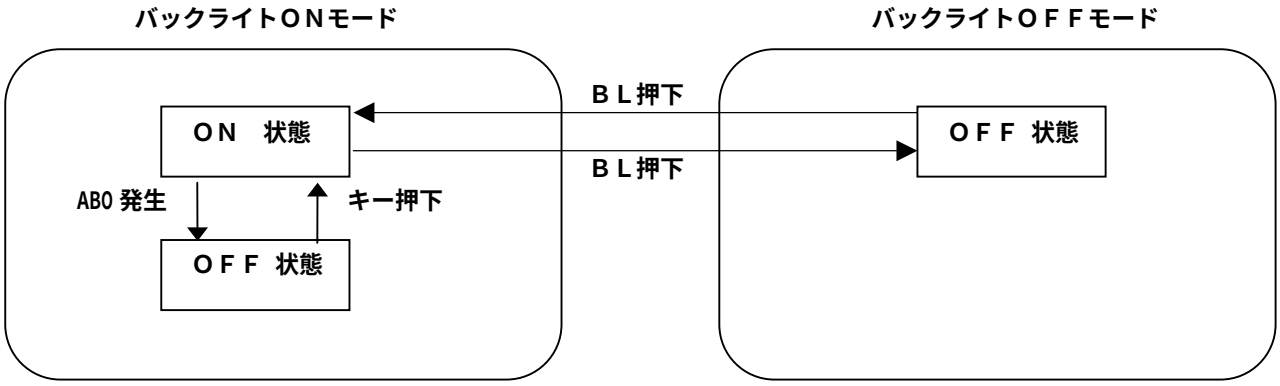
本機のキーバッファは、以下に示すようなリングバッファ構成になっています。



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。

4.1.10 バックライト制御

バックライトはＦ２（ＢＬ）キーでＯＮ／ＯＦＦ出来ます。バックライトＯＮ後は、一定時間キー操作が行われない場合はＡＢＯ（Auto Backlight Off）されます。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されます。再度、Ｆ２（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。



4.1.11 多点押し処理

本機ではキーを読取りキーと通常キー（読取りキー以外のキー）に分けています。読取り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

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）

読取りキーと通常キー

読取りキーが優先されバーコード読み込みが開始されます。但し、ＯＢＲが未オープンの場合は読取りキーは無視され通常キーが入力されます。

表４．８ 同時押し時の入力キー

押下キー	動作	備考
読取りキーと通常キー	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 ：読取りキーであってもＯＢＲが未オープン状態の場合

（２）多点押し（順次押下）

- 読取りキー押下後の通常キー
  通常キーは無視されます。読取りキーが離された時点で通常キーは入力されます。
- 通常キー押下後の読取りキー押下
  バーコード読込みを開始します。

表４．９ 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
読取りキー	通常キー	無視	バーコード読込み継続
通常キー	読取りキー	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 ： 読取りキーであってもＯＢＲが未オープン状態の場合

4.1.12　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４．１０ 属性一覧

属　性	内　　容
００ｈ	次に続くコードがアスキーコードであることを示します
ＦＦｈ	次に続くコードが内部処理コードであることを示します

※アプリケーションには属性が００ｈのコードしか返りません。

4.2.2 コード

本システムで使用可能なアスキーコードを以下に示します。

表４．１１ 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

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_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】

```
000000xxh     ：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_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_TMR     ：キー待ちタイムアウト発生
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

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）
```

※初期値がない場合は文字列格納エリアの先頭に NULL を入れてください。

【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_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_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）
```

※初期値がない場合は文字列格納エリアの先頭に NULL を入れてください。

【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_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                    */
                          /*  エコーバック指定                                                  */
    UB     echo ;         /*    ECHO_ON                    ：エコーバックあり                    */
                          /*    ECHO_OFF                   ：エコーバックなし（※２）            */
                          /*  フォントサイズ                                                    */
    H      font_size ;    /*    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_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、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_5      ：ファンクションキー１～５
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_5      ：ファンクションキー１～５

ID  *flgid        ：イベントフラグＩＤ      （解除時は不要です）
                   FL_FK_INT_ID を設定して下さい。
UW  *setptn       ：セットするビットパターン（解除時は不要です）
                   FL_FK_INT_FNC1      :F1
                   FL_FK_INT_FNC2      :F2
                   FL_FK_INT_FNC3      :F3
                   FL_FK_INT_FNC4      :F4
                   FL_FK_INT_FNC5      :F5
```

【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。

使用例は「10.3.2　キーに対する通知モード」を参照してください。

機能	キー入力有効・無効設定	関数名	key_select

各設定可能キーに対して、キーの有効・無効のキー入力モードを設定します。

≪C言語インタフェース≫

【コーリングシーケンス】

```
ER  ercd = 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 ;        /* RESERVE                  */
      UB    bs ;       /* RESERVE                  */
      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 ;      /* RESERVE                  */
      UB    ent ;      /* リターン                 */
      UB    func1 ;    /* Ｆ１(▲)                 */
      UB    func2 ;    /* Ｆ２（BL）              */
      UB    func3 ;    /* Ｆ３（BS）              */
      UB    func4 ;    /* Ｆ４（▼）              */
      UB    func5 ;    /* Ｆ５（SP）              */
      UB    func6 ;    /* RESERVE                  */
      UB    func7 ;    /* RESERVE                  */
      UB    func8 ;    /* RESERVE                  */
      UB    mltr ;     /* 読取りキー               */
      UB    mltl ;     /* RESERVE                  */
                       } 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≫

※設定可能キーについては「4.1.8 キー入力有効／無効」を参照して下さい。

機能	キー入力モードの設定／解除	関数名	key_setInputMode

数値入力モードと文字入力モードの切替えを行います。
S アイコンの点灯は本関数ではされません。表示機能の関数(lcd_shiftsymbol)でアイコンが点灯します。

≪C言語インタフェース≫

【コーリングシーケンス】

```
ER    ercd = key_setInputMode( UB mode );
```

【パラメータ】

UB	mode	：動作モード	
		NUM_INPUT_SET	：数値入力モード
		STRING_INPUT_SET	：文字入力モード

【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_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	54±5deg
読み取り角度	44deg

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／ＥＡＮ－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:カントリーフラグ
	EAN-13	13	FFMMMMMNNNNNC」	M:生産者コード
	JAN-8	8	FFMMMNC」	N:商品コード
	EAN-8	8	FFMMMNC」	S:システムメンバーキャラクタ
	JAN-13 addon+2	15	FFMMMMMNNNNNCAA」	A:addon データ
	EAN-13 addon+2	15	FFMMMMMNNNNNCAA」	」:終了コード(CR または LF または CR+LF)
	JAN-13 addon+5	18	FFMMMMMNNNNNCAAAAA」	UPC-B を除きチェックデジット(mod 10)の計算
	EAN-13 addon+5	18	FFMMMMMNNNNNCAAAAA」	は必ず行われます
	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	OSMMMMMNNNNNC」	
	UPC-B	12	OSMMMMMNNNNNN」	
	UPC-A addon+2	14	OSMMMMMNNNNNCAA」	
	UPC-B addon+2	14	OSMMMMMNNNNNNAA」	
	UPC-A addon+5	17	OSMMMMMNNNNNCAAAAA」	
	UPC-B addon+5	17	OSMMMMMNNNNNNAAAAA」	
	UPC-E	(7),8	OMMNNNMC」	最後の M:0～2
		(7),8	OMMMNN3C」	
		(7),8	OMMMMN4C」	
		(7),8	OMMMMMNC」	最後の N:5～9
	UPC-E addon+2	(9),10	OMMNNNMCAA」	最後の M:0～2
		(9),10	OMMMNN3CAA」	
		(9),10	OMMMMN4CAA」	
		(9),10	OMMMMMNCAA」	最後の N:5～9
	UPC-E addon+5	(12),13	OMMNNNMCAAAAA」	最後の M:0～2
		(12),13	OMMMNN3CAAAAA」	
		(12),13	OMMMMN4CAAAAA」	
		(12),13	OMMMMMNCAAAAA」	最後の N:5～9
CODE-39		3～40	SBBB･･･････････････BBCS」	A:ASCII 変換後データ,
		3～40	SAAA･･･････････････AACS」	B:ASCII 変換前データ
		1～38	BBB･･･････････････BBC」	C:チェックデジット(mod 43)
		1～38	AAA･･･････････････AAC」	チェックデジットなしの場合は、データとなります
				S:スタート･ストップキャラクタ(*)
NW-7		3～40	SDDD･･･････････････DDDE」	S:スタートコード(a,b,c,d のいづれか)
		1～38	DDD･･･････････････DDD」	E:エンドコード(a,b,c,d のいづれか)
				D:データ
Interleaved 2 of 5		2～40	DDD･･･････････････DDDC」	D:データ
				C:チェックデジット(mod 10)
				チェックデジットなしの場合は、データとなります
				読取桁数は偶数桁のみ
Industrial 2 of 5		2～40	DDD･･･････････････DDDC」	D:データ
				C:チェックデジット(mod 10)
				チェックデジットなしの場合は、データとなります
				読取桁数は偶数桁のみ
CODE-93		1～40	AAA･･･････････････AAA」	A:ASCII 変換後データ,　B:ASCII 変換前データ
				C:チェックデジット(mod 47)
				S:スタートコード,　E:エンドコード
CODE-128		1～64	AAA･･･････････････AAA」	A:ASCII 変換後データ,
		1～64	SBBB･･･････････････BBCE」	B:ASCII 変換前データ
		1～64	FAAA･･･････････････AAA」	C:チェックデジット(mod 47)
		1～64	GAAA･･･････････････AAA」	S:スタートコード,E:エンドコード,
				F:コード ID(“)C1”,EAN-128 のみ)
				G:GS(1Dh,EAN-128 のみ)

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。

MSI		1～40	DDD･･･････････････････DDCC」	D:データ C:チェックデジット(mod 10,mod 11) チェックデジットなしの場合は、データとなります
IATA		1～40	DDDDDDDDDD･･･････････C」 PADDDDDDDDDDDDDDC」	D:データ C:チェックデジット(IATA 仕様) チェックデジット無しの場合はデータとなります P:クーポン NO. A:エアライン NO. D:データ C:チェックデジット

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

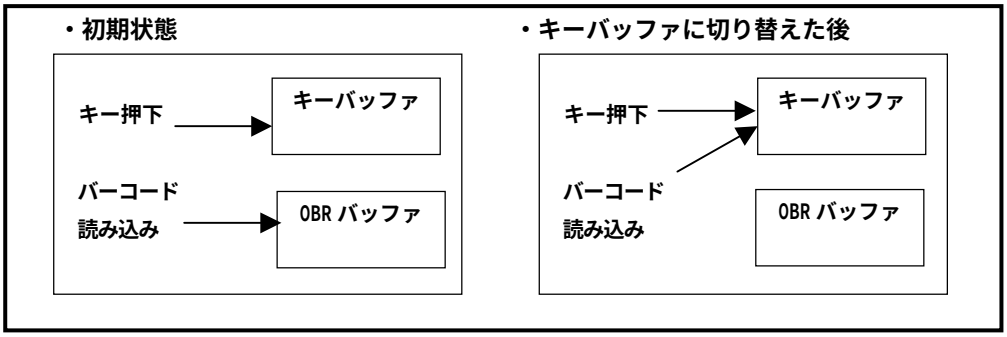
OBR バッファのデータ格納状態をチェックし、バッファ内の残りバイト数と残り段数を通知します。

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。



各バッファの使用方法を記載します。

バッファ名	使用方法
OBR バッファ	バーコードを読み取るとコード種別・データバイト数とともにデコードデータをバッファに保存します。 同時に、キー入力関数で読み取り完了通知を指定している場合は通知します。バッファ内のデータはクリアするまで保存されるので、自由なタイミングで読み出しが可能です。また、バッファ内のデータをすべて読み出して不要になったときは、バッファをクリアしてください。
キーバッファ	通常のキー入力と同様にデコードデータをキーバッファに出力します。 OBRバッファから切り替えた時は、OBRバッファを一旦クリアしてください。

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		備考
バーコード関連	読み取りコード	
	読み取り桁数	
	出力フォーマット	
	チェックデジットの計算	
	チェックキャラクタの出力	
出力関連	出力先バッファ	
	終了コード	
読み取り関連	読み取り方法	
	読み取り動作	
	ブザー制御	
	LED制御	
	読み取り回数	データ管理部で設定
	照合回数	データ管理部で設定
	スキャン時間	データ管理部で設定
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

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

表５．５　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(EAN-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	単一および任意の複数コードが選択可能です	（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。

各コードの設定可能範囲は次の通りです。

表５．６　読取り桁数

WPC	:	桁数は固定 （設定不可能）
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）
Industrial 2of5	:	2～40桁
Interleaved 2of5	:	4～40桁
CODE-93	:	1～40桁
CODE-128	:	1～64桁
MSI	:	1～40桁
IATA	:	1～40桁

制限事項：

- Interleaved 2of5 は仕様上、コードデータは必ず偶数桁です。

  奇数桁の指定をした場合，最小桁は(指定桁+1)の桁数まで，最大桁は(指定桁-1)の桁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。

- 誤読防止のため、CODE-39 および NW-7 の 1 桁，Interleaved 2of5 の 2 桁に関しては，コードを限定した場合のみ設定可能です。

  ＜ 設定範囲 ＞　１コード限定のとき

  表５．７　読取り桁数（１コード限定時）

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）
Interleaved 2of5	:	2～40桁

③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　出力フォーマット

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	出力有り
	Full ASCII 変換の有無を設定	変換なし
NW-7	スタート/ストップキャラクタの出力の有無を設定	出力有り
CODE-128	変換前データ/変換後データ(ASCII)のどちらかを出力するかを設定	変換後データ
	EAN-128 出力の有無を設定	出力無し
	コード ID 出力の有無を設定(EAN-128 のみ)	出力無し
	先頭 Fncl を GS に変換して出力するかを設定(EAN-128 のみ)	出力無し

④チェックデジッドの計算

各コードの計算方式による結果とチェックキャラクタを照合し，一致したときのみデコードを完了する設定を有効／無効にすることができます。（NW-7 は除く）

⑤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を有効無効に設定することができます。

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/UPC-E addon	出力有り
Industrial 2of5	出力有り
Interleaved 2of5	出力有り
MSI	出力有り

⑥出力バッファ参照

デコードデータを出力する，現在のバッファ設定を参照します。

⑦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

⑧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	読取り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	読取り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。 また、二度読み防止のため同一ラベルを連続して読むことは出来ません。	・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 ・読取りキー離し

⑨読み取り動作

読み取り方法で連続読みを指定している場合、次の動作切り替えが可能です。
通常読み：オープン後、クローズするまで連続して読み取りが行われます。
段数読み：オープン後、指定された回数分の読み取りが行われます。
（読み取ったコードが既に OBR バッファに格納されているときはカウントされません）

⑩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。
※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。

⑪LED 制御

1 コードごとの読み取り完了を LED の点灯によって通知することができます。また LED 制御を無効にすることも可能です。モードにより以下のように LED は制御は制御されます。

表５．１１　LED 制御一覧

読取り状態＼モード	0	1	2
読取り正常	変化なし	状態１	状態１
読取り以上	変化なし	状態２	変化なし

状態１　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。
状態２　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 における Full ASCII 変換エラー

⑫読み取り回数

連続読みの場合の読み取りコード数を設定できます。
（設定した回数分のコードを読み取り完了すると，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。

⑬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。

⑭スキャン時間

読取りキーを押下した後の読み取り可能時間を設定できます。
（設定した時間を経過すると，自動的に読み取り待機状態となります）
1～9 秒まで設定することが可能です。

⑮立ち上げモード

読取りキーによる電源オンを有効／無効にすることができます。

表５．１２ 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	立ち上げ可	立ち上げ可
CLOSE 状態	立ち上げ不可	立ち上げ可	立ち上げ不可

（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。

5.3 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 OR で指定）

・使用コード

OBR_CD39	：CODE39
OBR_NW_7	：NW-7
OBR_WPCA	：WPC(UPC-E 以外)addon
OBR_WPC	：WPC(UPC-E 以外)
OBR_UPEA	：UPC-E addon
OBR_UPE	：UPC-E
OBR_IDF	：Industrial 2of5
OBR_ITF	：Interleaved 2of5
OBR_CD93	：CODE93
OBR_CD128	：CODE128
OBR_MSI	：MSI
OBR_IATA	：IATA

・チェックデジット実行指示

OBR_CHK_ON

・チェックキャラクタ出力指示

OBR_OUT_ON

※１ 現在設定されている動作モードでオープンする場合は、オープンモードを“０”にします。
※２ オープンモードが“０”以外の場合、動作モードは初期化されます。
（使用コードのチェックデジット／チェックキャラクタ以外の項目は初期値になります）
※３ 本関数で設定できない項目は、動作モードの設定関数で設定してください。
読取り回数、照合回数、読取り時間は、システムデータ管理関数で設定してください。

【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
E_CMN_EXC	：排他エラー

≪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

```
struct {
      UW    Code;         /* 読み取り可能コード                    */
      UB    Cd39[6];      /* Code39                                */
      UB    Nw7[6];       /* Nw-7                                  */
      UB    Wpcea[6];     /* Wpc(Upc-e 以外)addon                  */
      UB    Wpce[6];      /* Wpc(Upc-e 以外)                       */
      UB    Upcea[6];     /* Upc-e addon                           */
      UB    Upce[6];      /* Upc-e                                 */
      UB    Idsf[6];      /* Industrial 2of5                       */
      UB    Itrf[6];      /* Interleaved 2of5                      */
      UB    Cd93[6];      /* Code93                                */
      UB    Cd128[6];     /* Code128                               */
      UB    Msi[6];       /* MSI                                   */
      UB    Iata[6];      /* IATA                                  */
      UB    Resv[20][6];  /* 拡張用                                */
                          /* 配列[0]リザーブ                       */
                          /*     [1]読み取り桁数(Min)              */
                          /*     [2]読み取り桁数(Max)              */
                          /*     [3]出力フォーマット               */
                          /*     [4]チェックデジット実行指示       */
                          /*     [5]チェックキャラクタ 出力指示    */
      UB    Type;         /* 読み取り方式の設定                    */
      UB    Gain;         /* Reserved                              */
      UB    Buzc;         /* ブザーコントロールの設定              */
      UB    Ledc;         /* LED コントロールの設定                */
      UB    Bufc;         /* 出力バッファの設定                    */
      UB    Endc;         /* 終了コードの設定                      */
      UB    Mode;         /* 通常読み/段数読み指定                 */
      UB    Dumy;         /* 拡張用                                */
} M_TBL
```

※得られるデータについての詳細は OBR_modewt 関数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

≪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 （構造体は OBR_moderd 関数を参照）

【リターンパラメータ】

ER ercd	：リターンコード

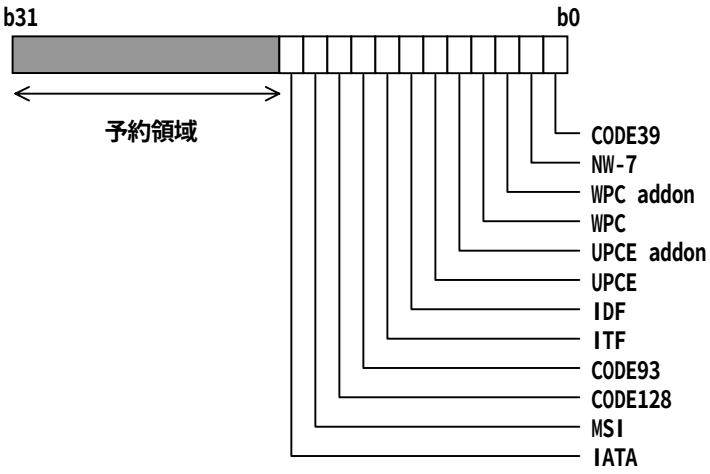
【リターンコード】

E_OK	：正常終了
E_PRM	：パラメータエラー
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
M_TBL 構造体の Code に右図を参照してセットするデータを作り設定してください。

（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>M_TBL 構造体に各コードごと以下の項目を設定します。

・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

設定値は左表を参照してください。

（１９２バイト）</td><td>
対応する配列の要素→ [0] [1] [2] [3] [4] [5]
コード種別↓ リザーブ / Min / Max / 出力フォーマット / チェックデジット / チェックキャラクタ
CODE-39 FFh 2 38 0*1 0 1
NW-7 FFh 2 38 0*2 - -
WPC(UPCE以外)addon FFh 10 18 - 1 -
WPC(UPCE以外) FFh 8 13 - 1 -
UPCE addon FFh 9 12 - 1 1
UPCE FFh 7 7 - 1 1
Industrial 2 of 5 FFh 2 40 - 1 1
Interleaved 2 of 5 FFh 4 40 - 1 1
CODE-93 FFh 3 40 - 1 -
CODE-128 FFh 2 64 0*3 1 -
MSI FFh 1 40 - 1*4 1
IATA FFh 1 40 - 0*5 -
リザーブ FFh - - - - -

－ 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）

チェックデジット OBR_CHKDOFF:計算無し OBR_CHKDON:計算有り
チェックキャラクタ OBR_CHKKOFF:出力無し OBR_CHKKON:出力有り

*1 CODE39出力フォーマット OBR_39SON(00h):Start/Stopコード有り
OBR_39SOFF(01h):Start/Stopコード無し
OBR-39ASON(02h):Full ASCII Start/Stopコード有り
OBR_39ASOFF(03h):Full ASCII Start/Stopコード無し

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

*3 CODE128出力フォーマット OBR_128AON(00h):変換後(ASCII)のデータを出力
OBR_128AOFF(01h):変換前のデータを出力
OBR_128EAN(02h):EAN-128出力有り
OBR_128AIM(06h):EAN-128コードID出力有り
OBR_128FNC(0Ah):EAN-128Fnc→GS出力有り
変換前/後の対応は本仕様書最終頁参照

*4 MSIチェックデジット OBR_CHKDOFF(00h):チェックデジット無し
OBR_CHKDON(01h):１桁、mod10
OBR_CDMSEV(02h):2桁、1st:mod11 2nd:mod10
OBR_CDMSTN(03h):2桁、1st:mod10 2nd:mod10

*5 IATAチェックデジット OBR_CHKDOFF(00h):チェックデジット無し（読取り桁数1-40）
OBR_CHKDON(01h):最終キャラクタ以外を対象（読取り桁数2-40）
OBR_CDMSEV(02h):クーポン番号/整数部を対象(読取り桁数15-17)
OBR_CDMSTN(03h):整数部を対象（読取り桁数15-17)
</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 M_TBL 構造体の Type に左の値を設定します。（１バイト）	OBR_TYPE0(00h)：単発読み OBR_TYPE1(01h)：連続読み(読み取りキー有り)	連続読み(01h)	○	○
ブザー制御の設定 M_TBL 構造体の Buzc に左の値を設定します。（１バイト）	OBR_BUZOFF(00h)：ブザー制御無し OBR_BUZON(01h) ：ブザー制御有り	ブザーあり(01h)	○	○
ＬＥＤ制御の設定 M_TBL 構造体の Ledc に左の値を設定します。（１バイト）	OBR_LEDOFF(00h) ：ＬＥＤ制御無し OBR_LEDON(01h) ：ＬＥＤ制御有り OBR_LEDEROF(02h)：ＬＥＤ制御有り(エラー除く)	ＬＥＤあり(01h)	○	○
出力バッファの参照 M_TBL 構造体の Bufc に左の値を設定します。（１バイト）	OBR_BUFOBR(00h)：ＯＢＲバッファに出力 OBR_STOFF(02h) ：ＫＥＹバッファに出力	ＯＢＲバッファ(00h)	○	×
終了コードの設定(バーコードの最後尾に付加するコード) M_TBL 構造体の Endc に左の値を設定します。（１バイト）	OBR_ENDCR(00h)：ＣＲ OBR_ENDLF(01h)：ＬＦ OBR_ENDCL(02h)：ＣＲ＋ＬＦ	ＣＲのみ(00h)	○	○
読取り動作の設定 M_TBL 構造体の Mode に左の値を設定します。（１バイト）	OBR_NORM(00h)：通常読み OBR_DANN(01h)：段数読み	通常読み(00h)	○	○

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の読取り分解能指定	関数名	OBR_gain

本機ではレーザモジュールの読取り分解能の切替え機能はありません。
本関数は互換用として残してあるため、呼ばれても E_OK を返すのみで機能はありません。

≪Ｃ言語インタフェース≫

【コーリングシーケンス】

```
ER     ercd = OBR_gain( UB gain );
```

【パラメータ】

```
UB     gain
```

【リターンコード】

```
ER     ercd          ：エラーコード
```

【エラーコード】

```
E_OK          0x00000000          ：正常終了
```

≪備考≫

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	読取りキーによる電源オン設定	関数名	OBR_trigmode

読取り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≫

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２つの通信ポートが存在します。通信関数部ではカシオＩＲインタフェースに対する機能を提供します。
ＩｒＤＡ部（次章）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
IrDA	半二重	IrDA	－	IrDA (IrSIR 1.1)	調歩	2.4k 9.6k 19.2 38.4k 57.6k 115.2k 4M	8bit	NON	1bit	SD RD

（2）排他制御

本機のカシオＩＲインタフェースとＩｒＤＡポートは同時に使用することはできません。

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

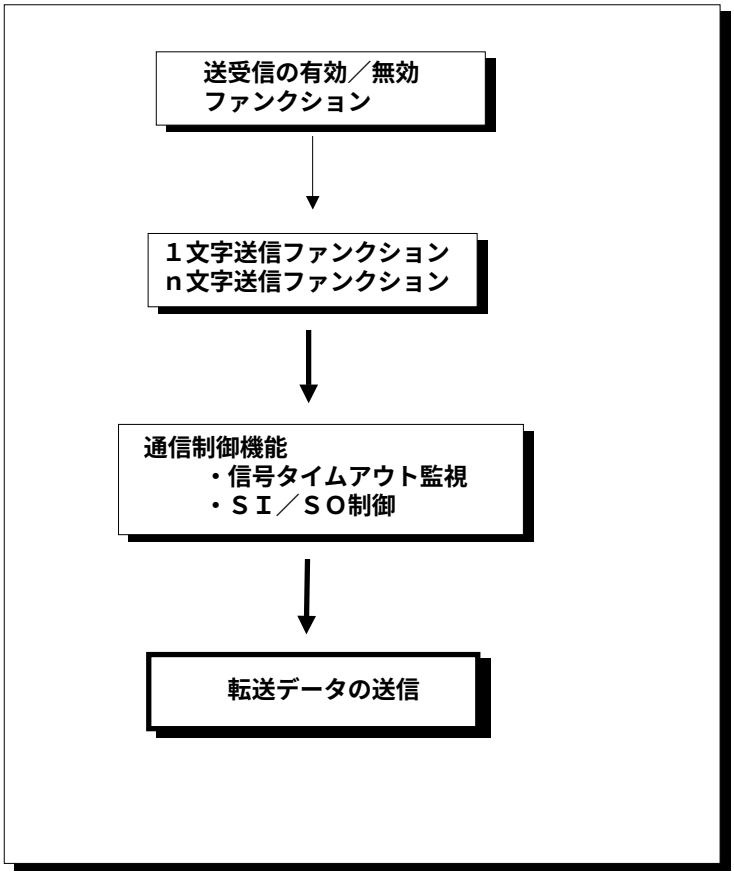
通信ポートのオープンを行うと、通信ポートに電源供給を行い、通信リソースの占有を行います。
これを“ポートオープン”状態とします。ポートオープン状態であるとき、同一（排他関係）の通信ポートをオープンするこはできませんし、本機の通信関数部では通信ポートを“ポート占有状態”にすることができます。
ポート占有状態であるとき通信ポートへの電源供給、通信リーソースの占有は行わず、同一（排他関係）の通信ポートのオープンの抑制のみを行います。
例えば、通信関数部以外で通信部が使用する通信リソースを使用してデータ通信を行う場合に通信ポートを占有状態にして、通信部が使用できないようにします。
通信ポートの占有は「ＣＯＭ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

【転送データの送信の流れ】



（2）受　　信

通信ポートをオープンすることで転送データの受信を行うことができ、ＳＩ／ＳＯ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。
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
- ・デリートコード制御
- ・エラーコードバッファリング制御

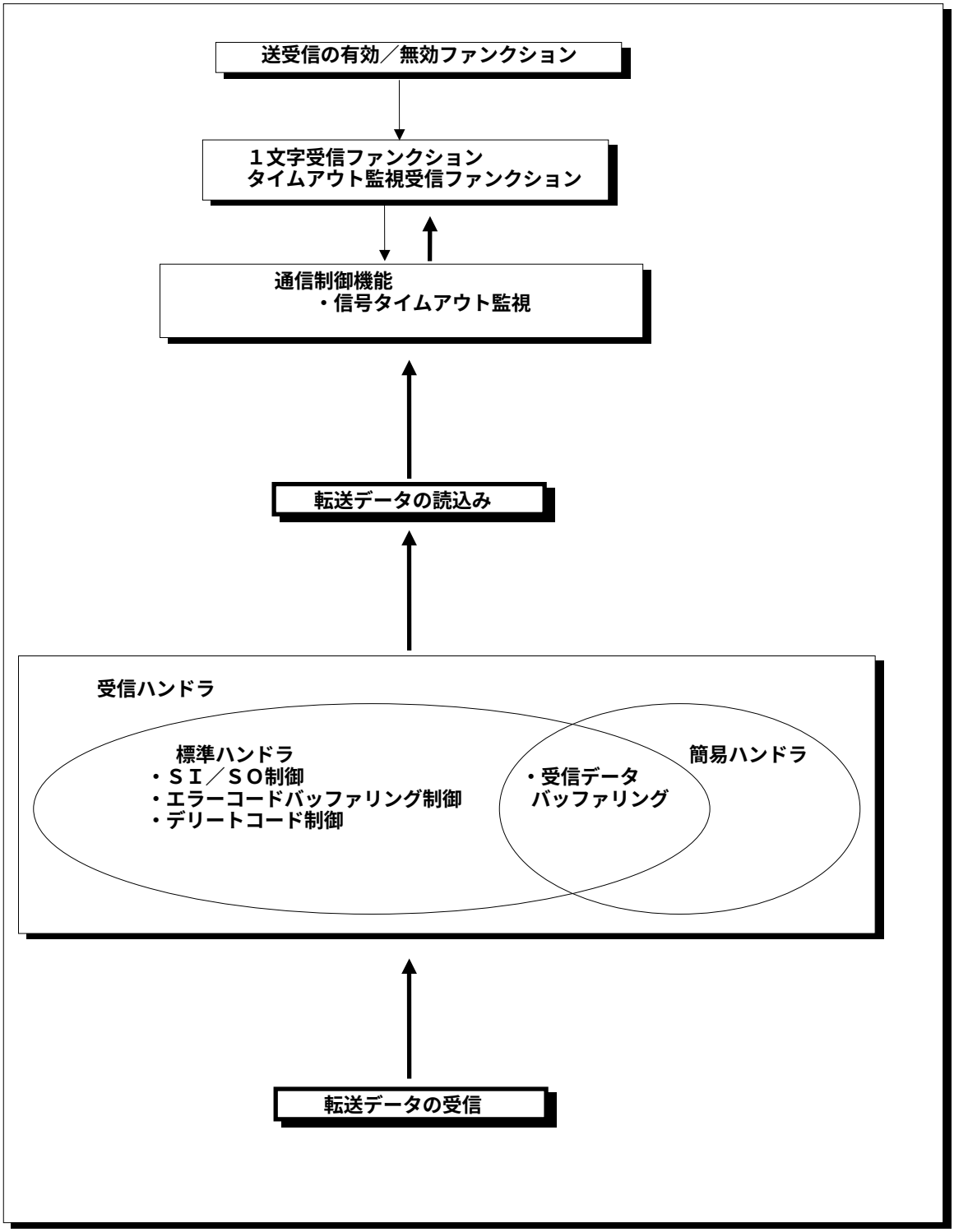
尚、本機の初期化において標準ハンドラに設定します。

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。

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「送受信の有効／無効」ファンクションを使用して行います。

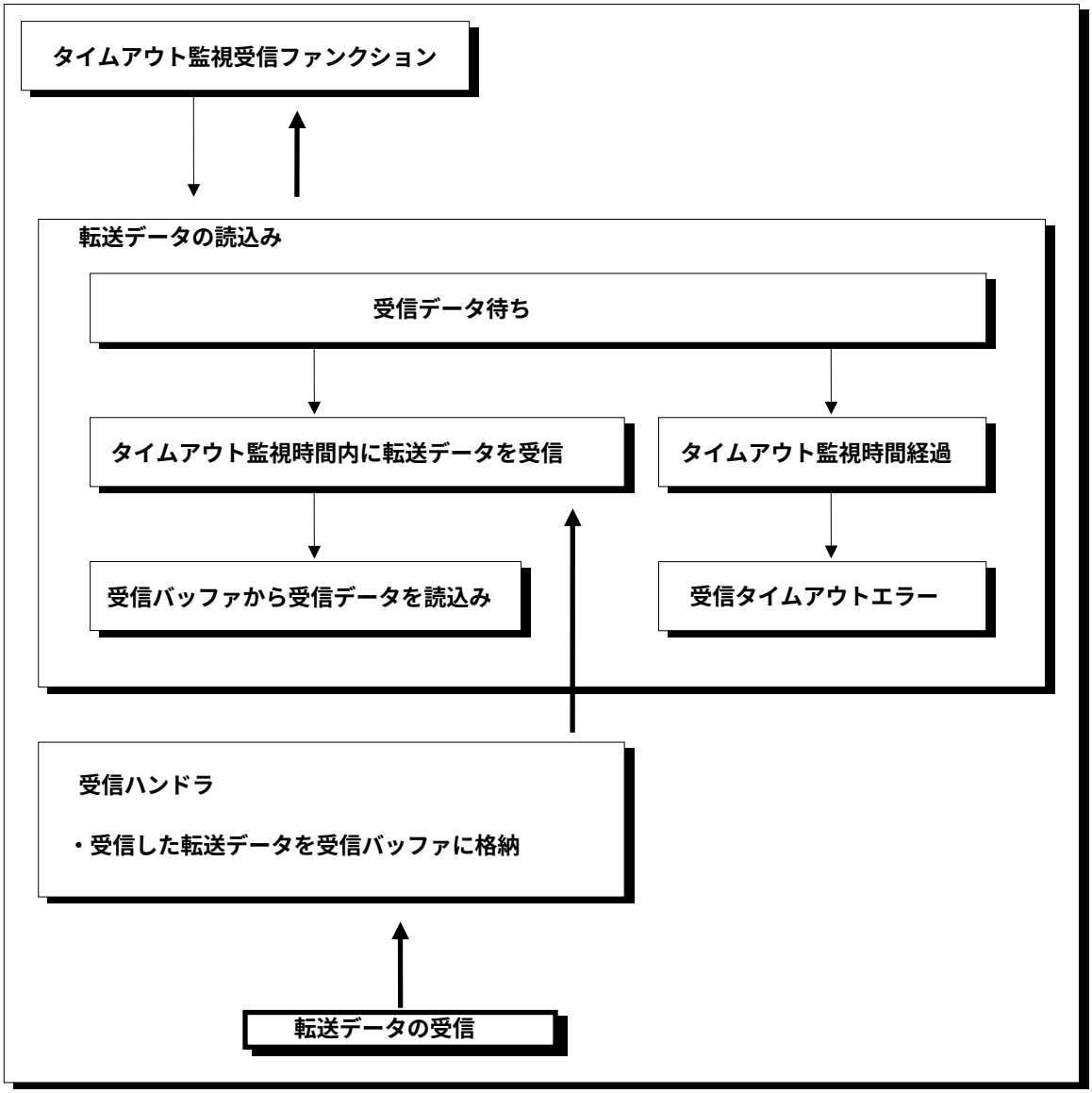
【転送データの受信および受信データの読込みの流れ】



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。

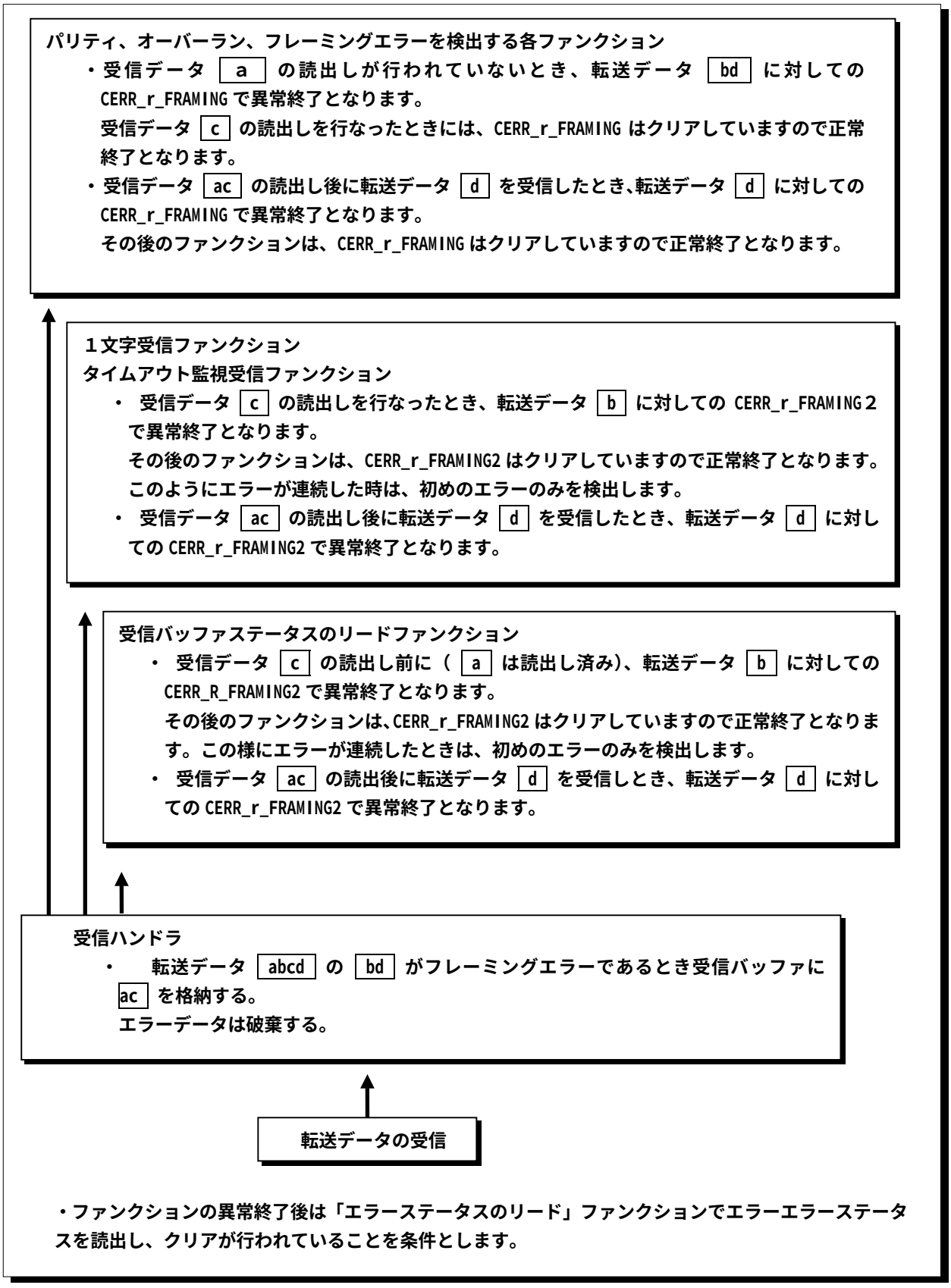
ｂ）ＣＥＲＲ＿ｒ＿ＰＡＲＩＴＹ２、ＯＶＥＲＲＵＮ２、ＦＲＡＭＩＮＧ２エラーステータス

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。

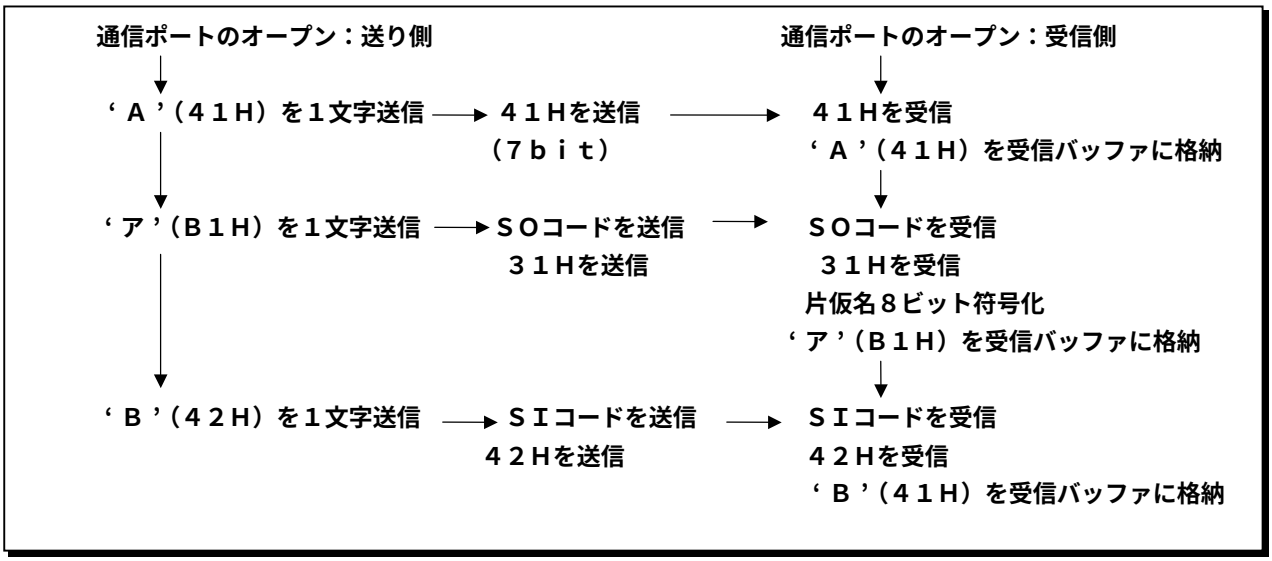
【パリティ、オーバーラン、フレーミングエラーの検出の流れ】



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ＪＩＳ８ビット符号化文字（ＪＩＳ　Ｘ　０２０１参照）を扱う場合に使用します。
ＪＩＳ８ビット符号化文字コードの片仮名８ビット符号（Ａ０Ｈ～ＤＦＨ）を送信するときＳＯコードの送信を行います。
ＳＯコードを受信したとき、その後に受信した転送データを片仮名８ビット符号に変換して受信バッファに格納します。
ＳＯコード送信後に９ＦＨ以下の文字コードを送信するとき、ＳＩコードを送信してから文字コードの送信を行います。

【ＳＩ／ＳＯ制御の流れ】



6.2.4　フロー制御

（1）XON/XOFF 制御

カシオ IR インターフェースは半二重のため、XON/XOFF 制御は行えません。
COM オープンの関数で指定しないで下さい。

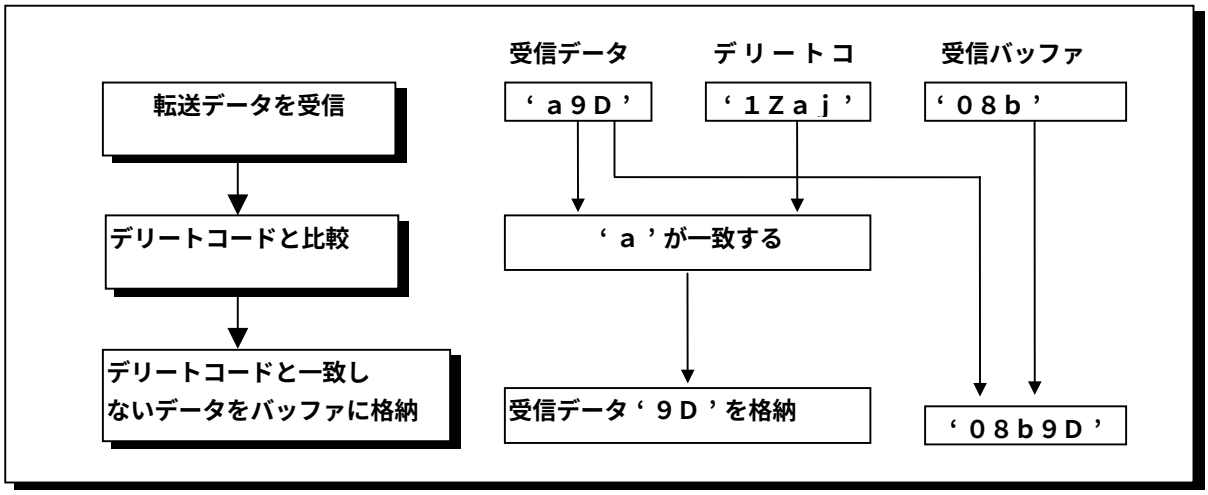
（2）ＲＳ／ＣＳフロー制御

カシオ IR インターフェースは SD、RD 信号のみのため、RS/CS フロー制御は行えません。
ただし、仮想の RS 信号は持っており、これが ON でないとデータの送信が行えないため、データ送信時には、これを ON にして下さい。

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。

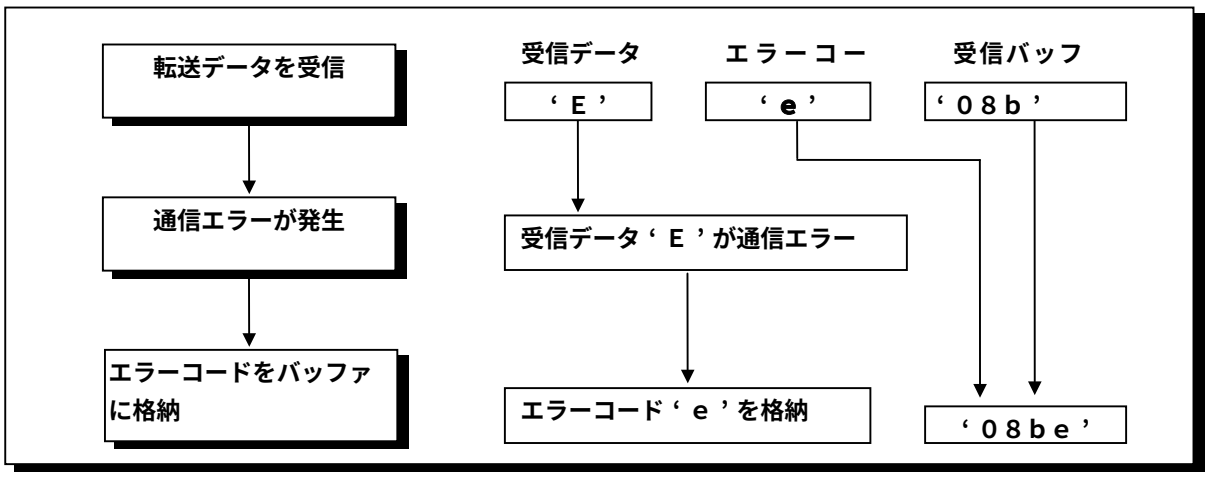
【デリードコードの制御のながれ】



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

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

以下にカシオＩＲポートが制御する信号線を示します。

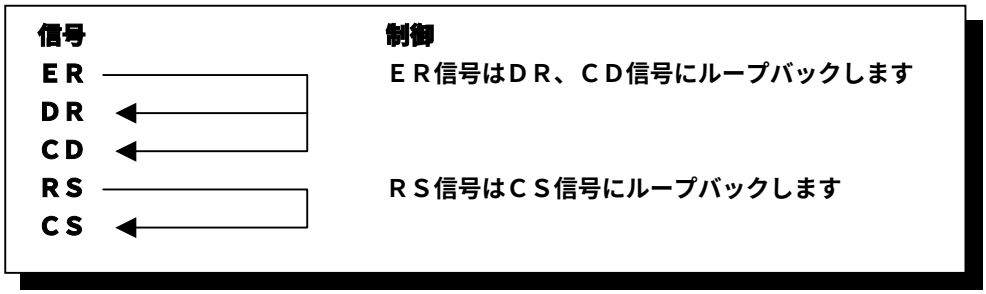
【通信ポートが制御する信号線】

ＥＲ	あり（仮想）
ＲＳ	あり（仮想）
ＤＲ	あり（仮想）
ＣＳ	あり（仮想）
ＣＤ	あり（仮想）
ＣＩ	なし
ＣＴＲＬ	なし

ｂ）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。

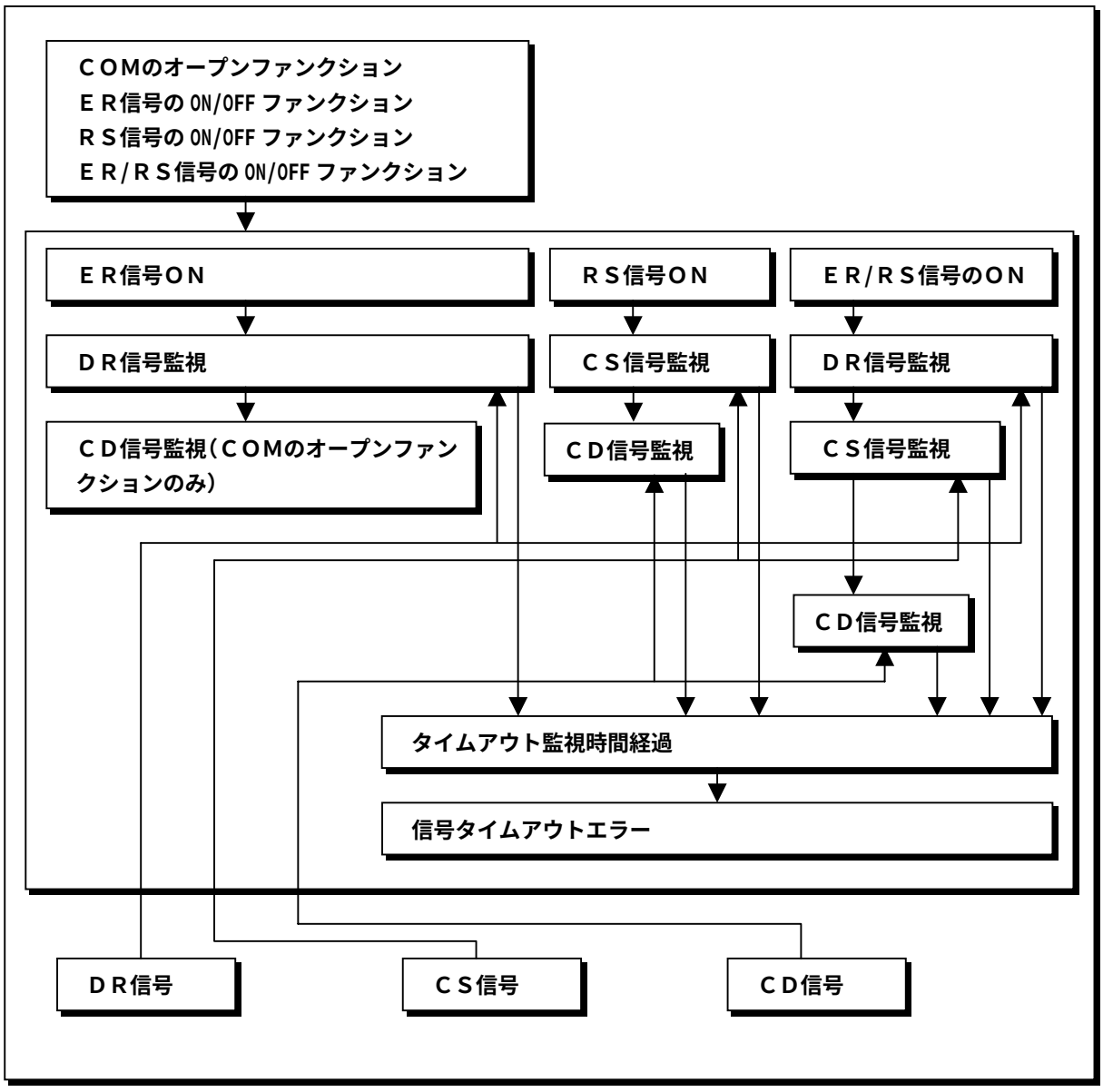
【カシオＩＲポートの仮想信号制御】



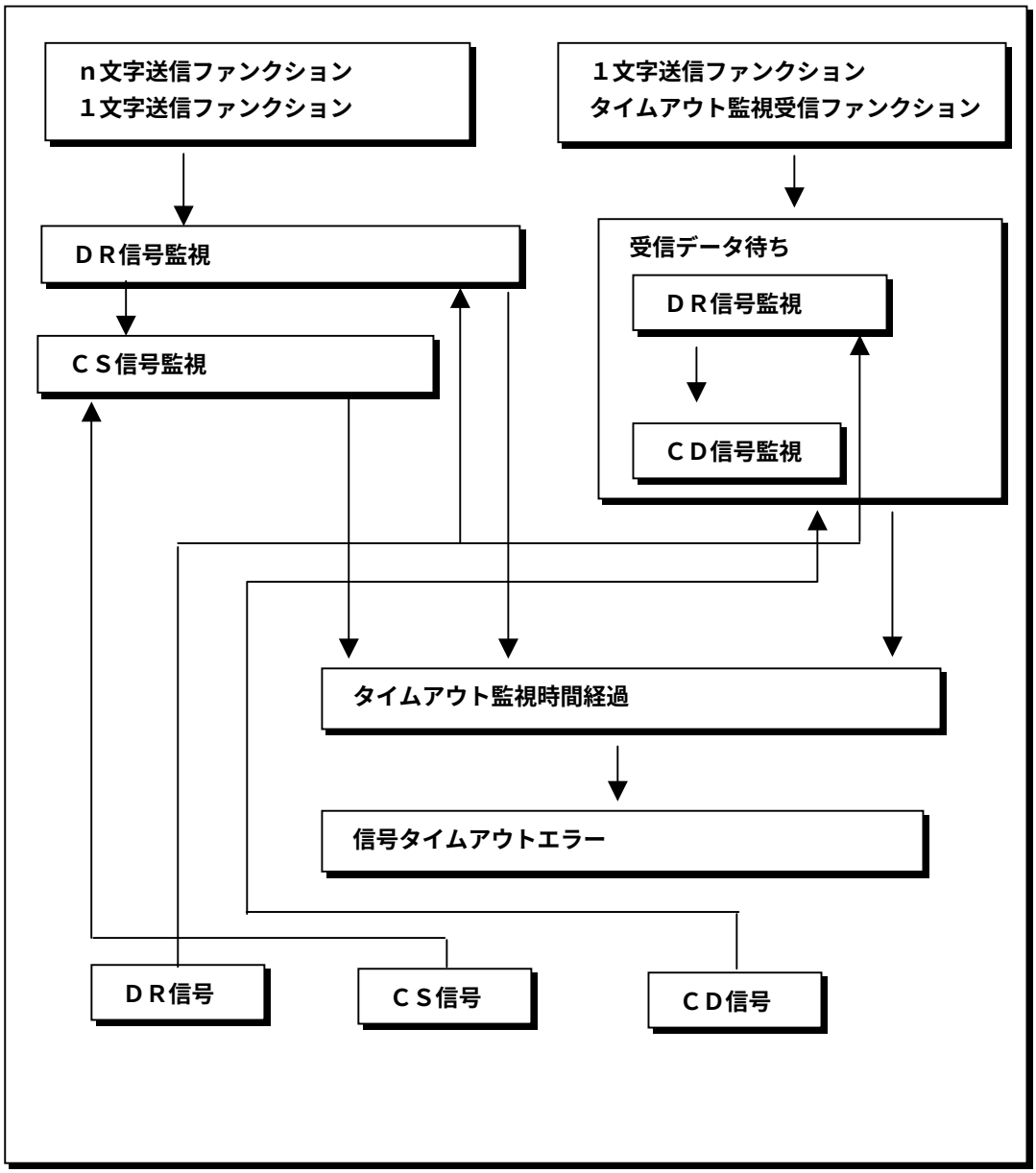
（2）タイムアウト監視

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

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
ＣＳ	転送データの送信前にＯＮ待ち

⑧ER 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

⑨RS 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

⑩ER／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 ＬＢ検出

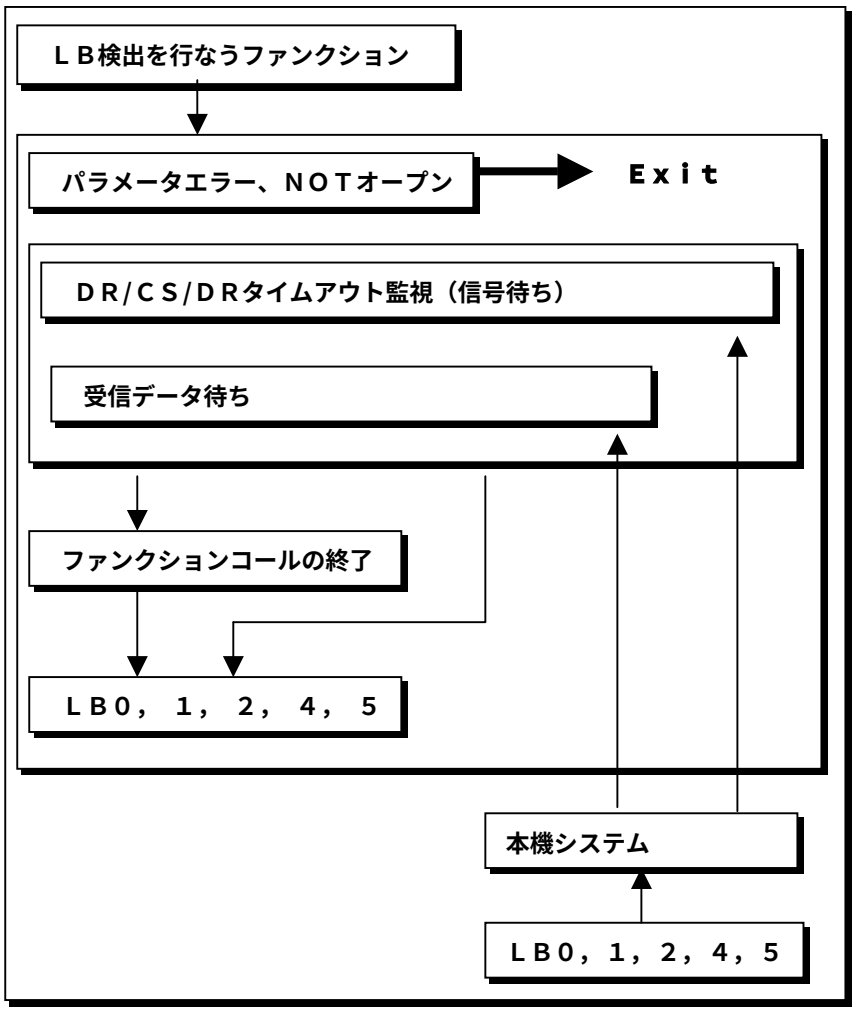
各ファンクションコールでは指定によりローバッテリの検出を行います。
ＬＢを検出するとファンクションは実行中の処理を中断し、異常終了します。
ＬＢの検出を行うには本機のシステムに対してＬＢ検出を行うように設定する必要があります。

（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】



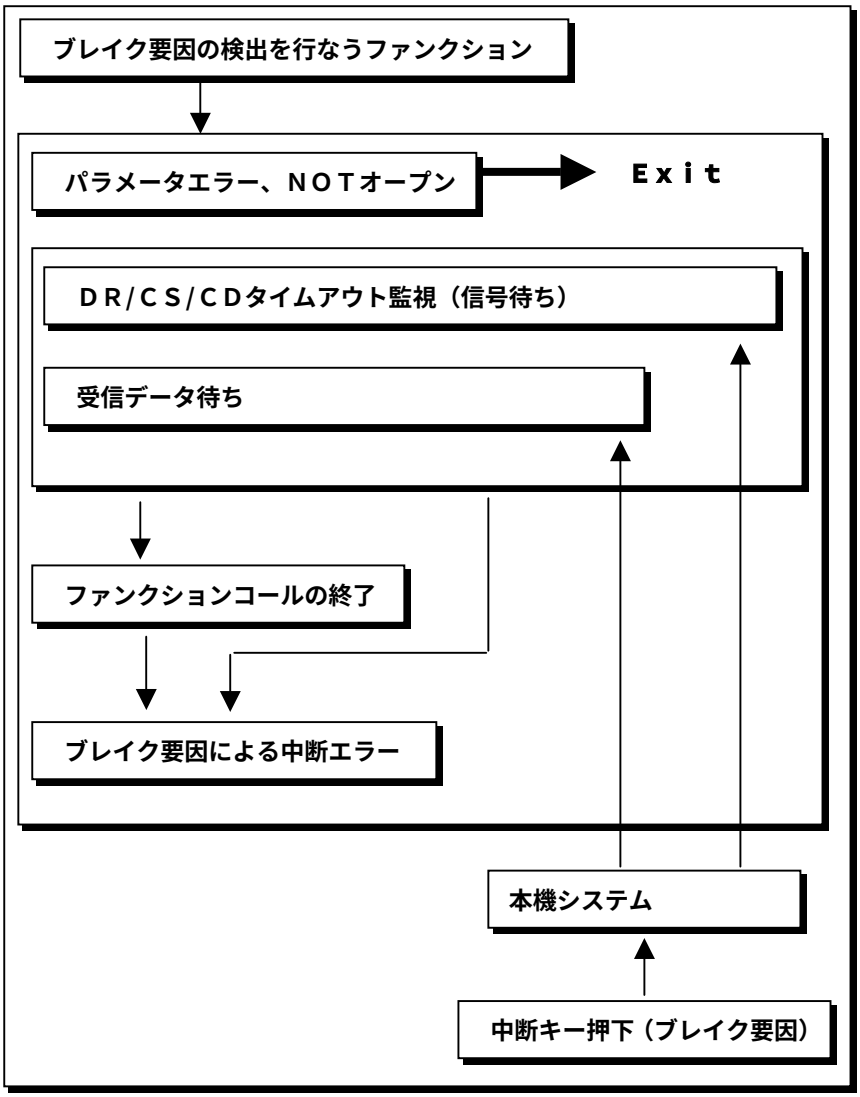
6.2.9 ブレイク要因検出

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

（1）ブレイク要因の検出を行うファンクション

ブレイク要因の検出はフロー制御、信号タイムアウト監視、受信データ待ちのときに行います。
ブレイク要因の検出を行うファンクションについては「６．４ エラー詳細」で各ファンクションのエラーステータスを参照して下さい。エラーステータスの「ＣＥＲＲ＿ｏ＿ＢＲＥＡＫ」を通知します。

【ブレイク要因検出の流れ】



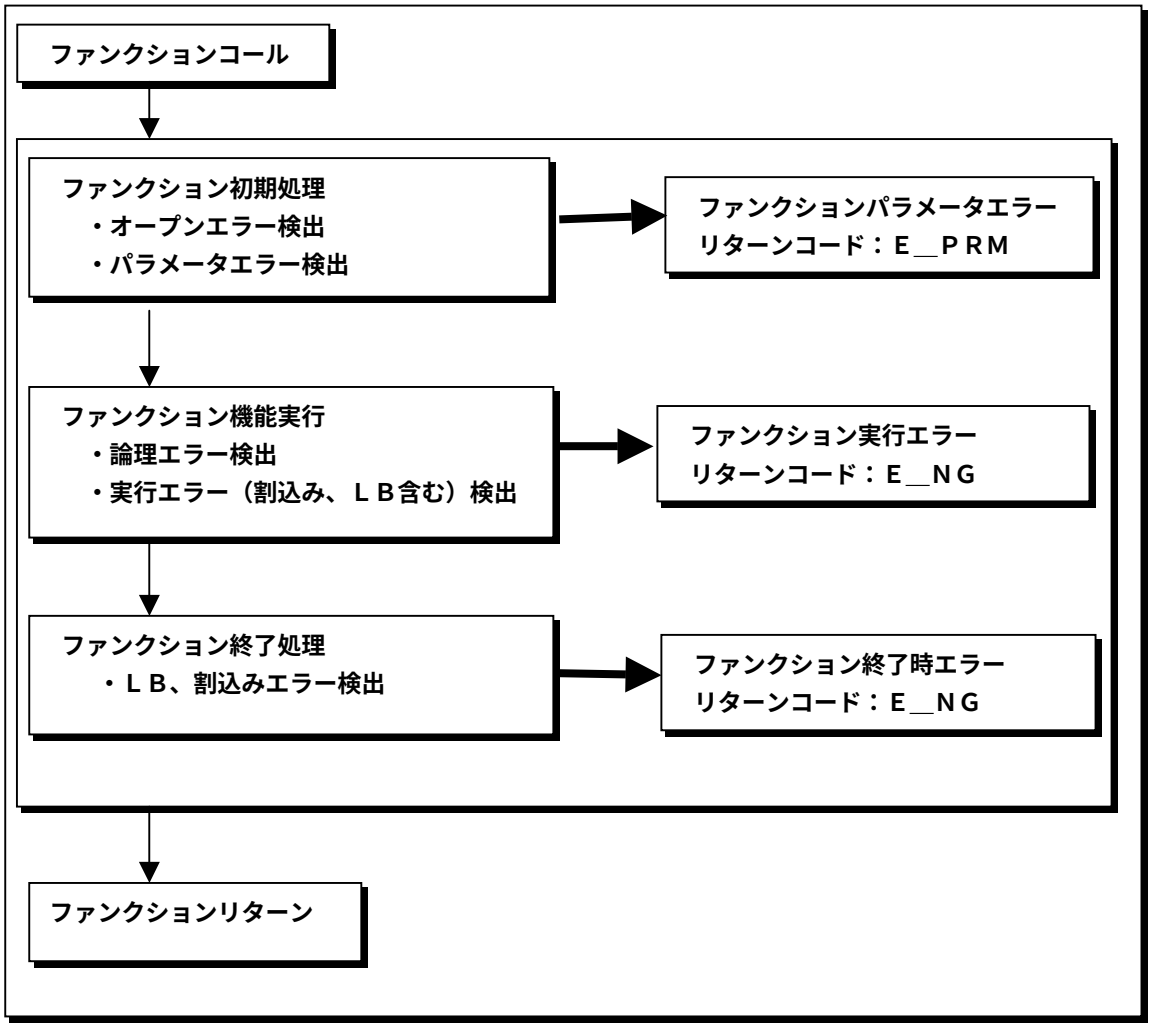
6.3 エラー詳細

エラーステータスはファンクションコールが異常終了したとき、その詳細を示します。
「エラーステータスのリード」ファンクションでエラーステータスを取得することができます。

6.3.1 ファンクションのエラー検出

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

【ファンクションのエラー検出の流れ】



6.3.2 エラー詳細

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

（1）ファンクション終了時エラー

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

エラーコード	エラーステータス	要因
E_NG	CERR_r_PARITY	パリティエラー ・パリティエラー検出
	CERR_r_OVERRUN	オーバーランエラー ・オーバーランエラー検出
	CERR_r_FLAMING （CERR_r_FRAMING）	フレーミングエラー ・フレーミングエラー検出
	CERR_r_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）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

（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

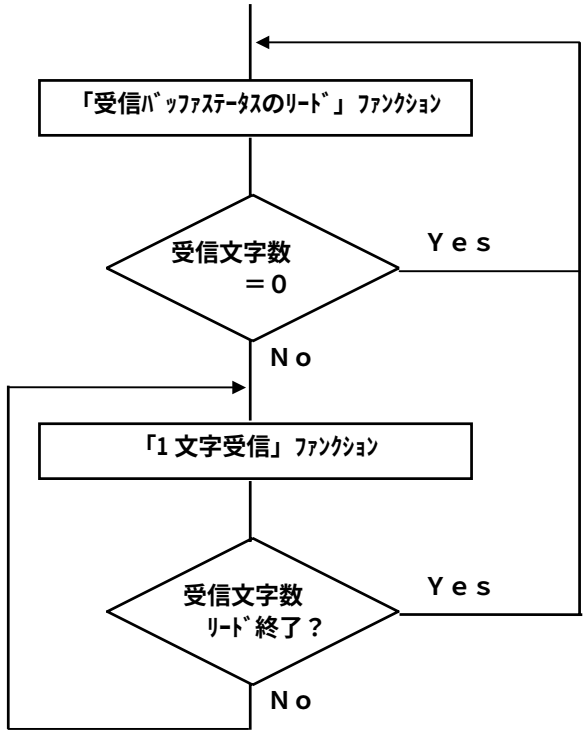
エラーコード	エラーステータス	要因
E＿PRM	なし	パラメータエラー ・ブレイク要因通知の指定が不当 ・ファンクションキーの指定が不当

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

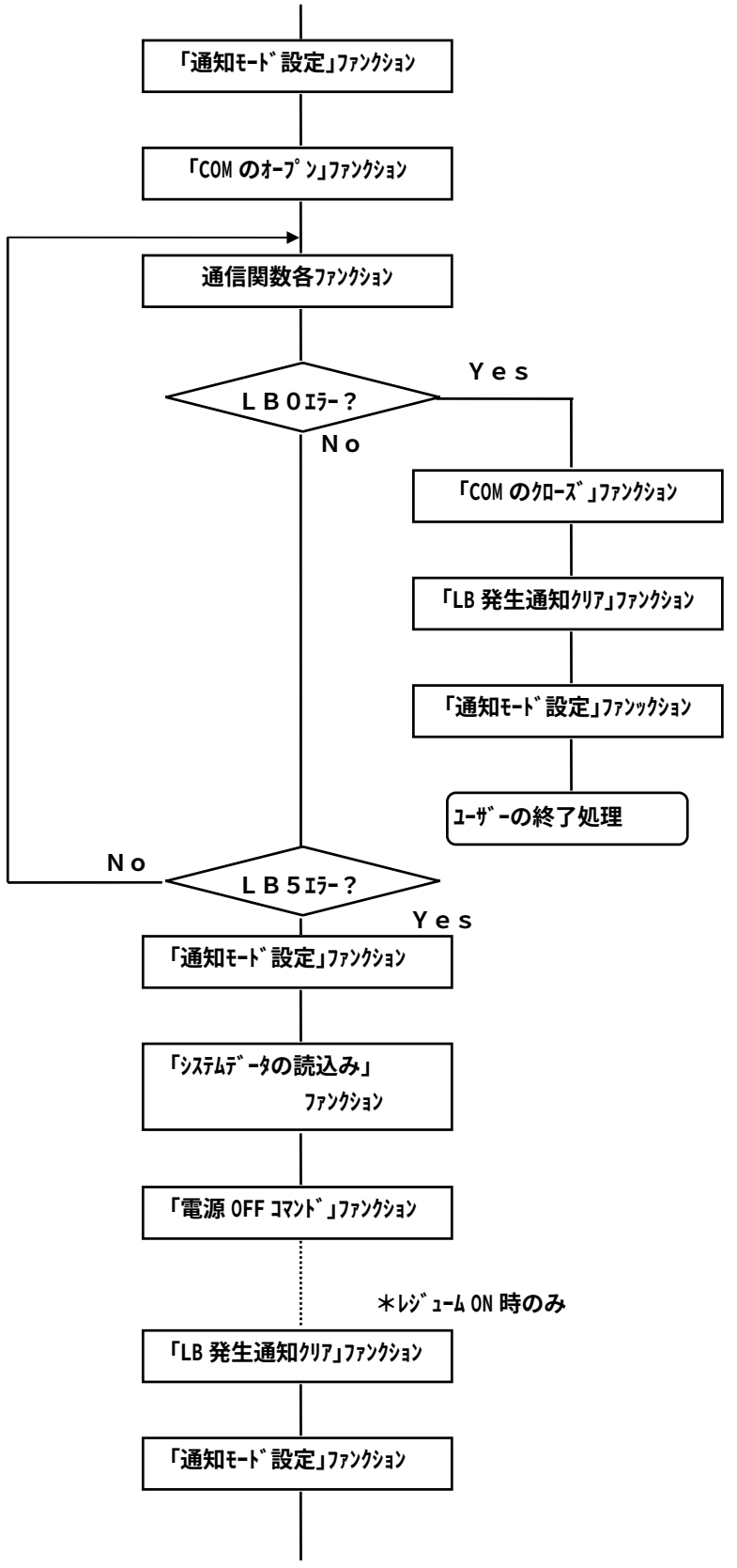
②受信文字が無ければ繰り返し

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

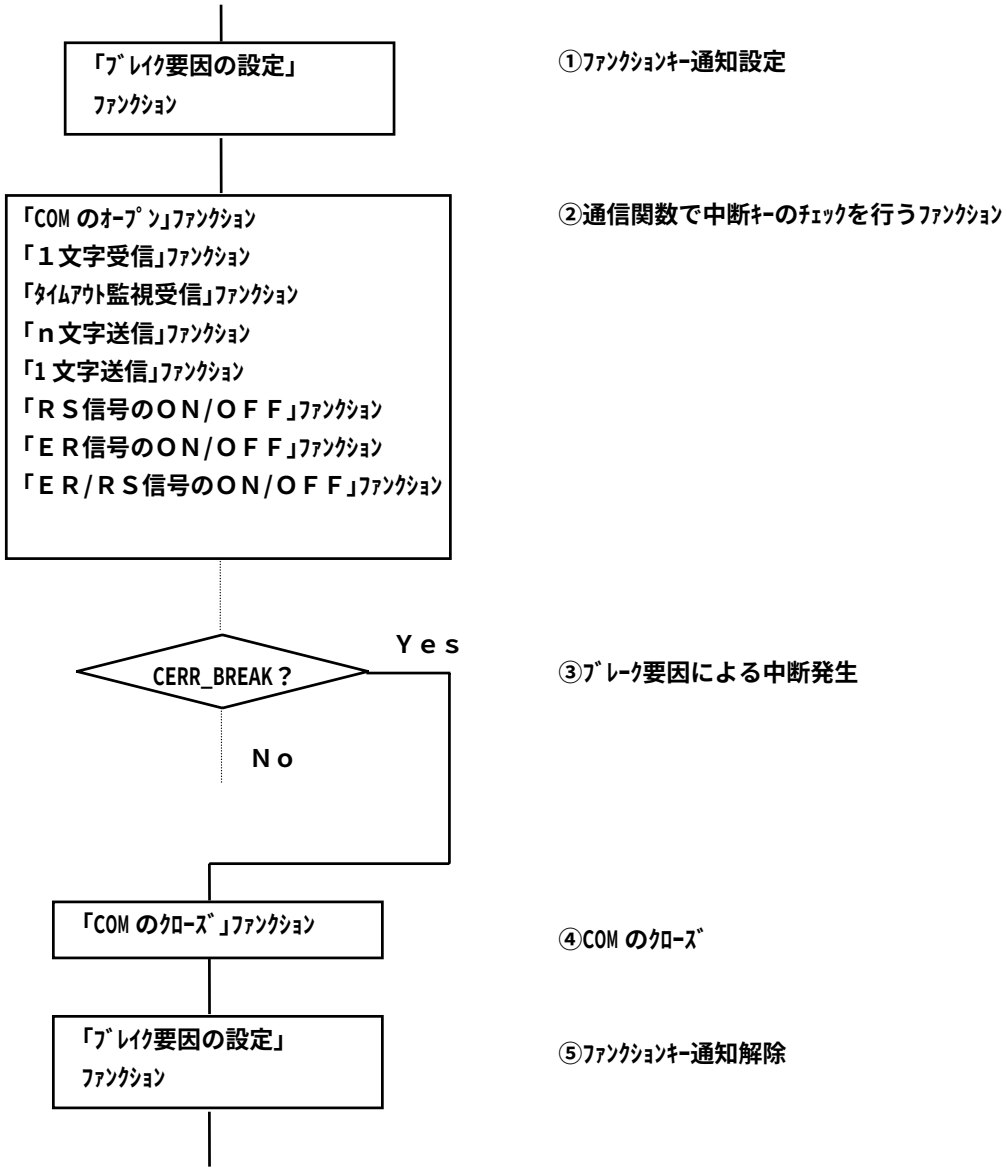
ローバーッテリ（ＬＢ）エラーのチェックを通信中の電源ＯＦＦキー押下（ＬＢ５）および主電池なし（ＬＢ０）発生時の処理例を以下に示します。レジュームＯＮによる通信の再開等ができない場合などを考慮して行います。



＊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 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	・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
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
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
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
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のオープン	関数名	c_open

３ＷｉｒｅＩＲポートをオープンします。
３ＷｉｒｅＩＲポートをオープンすると半二重（受信イネーブル、送信ディセーブル）の状態でオープンします。
本ファンクションが正常終了でないとき通信ポートはオープンしません。
本ファンクションは以下の処理及び設定を行います。

- 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

【戻り値】

ER	ercd	：エラーコード

【エラーコード】

E_OK	00000000h	：正常終了
E_NG	FFFFFFFFh	：異常終了
E_PRM	FFFFFFEh	：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」は意味がありません。
  RS を ON にすると CS はすぐに ON され、ER を ON にすると DR/CD がすぐに ON します。
- 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。

【パラメータ】

H com_no：COM No.
　　COM0　　：3WireIR
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
　　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
　　（以下のモードは指定不可です）
　　~~XON_XOFF　：DC1/DC3 での XON/XOFF ｺﾝﾄﾛｰﾙ~~
　　~~BUSY_CHAR：指定コードでの XON/XOFF ｺﾝﾄﾛｰﾙ~~
　　~~RS_CS　　：RS/CS による RS/CS ﾌﾛｰｺﾝﾄﾛｰﾙ~~
　　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
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のクローズ	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　：**ＣＯＭ　ＮＯ.**
　COM0　**：３ＷｉｒｅＩＲ**

【戻り値】

ER　ercd　：**エラーコード**

【エラーコード】

E_OK	00000000h	**：正常終了**
E_NG	FFFFFFFFh	**：異常終了**
E_PRM	FFFFFFFEh	**：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、受信キューオーバーフロー及びブレイク信号は割込み要因の
エラーステータスは本ファンクションによりクリアします。

- ・信号線（ＤＲ、ＣＤ、ＣＳ）のＯＮ／ＯＦＦ
- ・受信バッファオーバーフロー
- ・通信エラー（パリティ、オーバーラン、フレーミング）
- ・ブレイク信号の受信
- ・受信キュー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	：ＣＯＭ　ＮＯ.
		COM0　　：３ＷｉｒｅＩＲ

【戻り値】

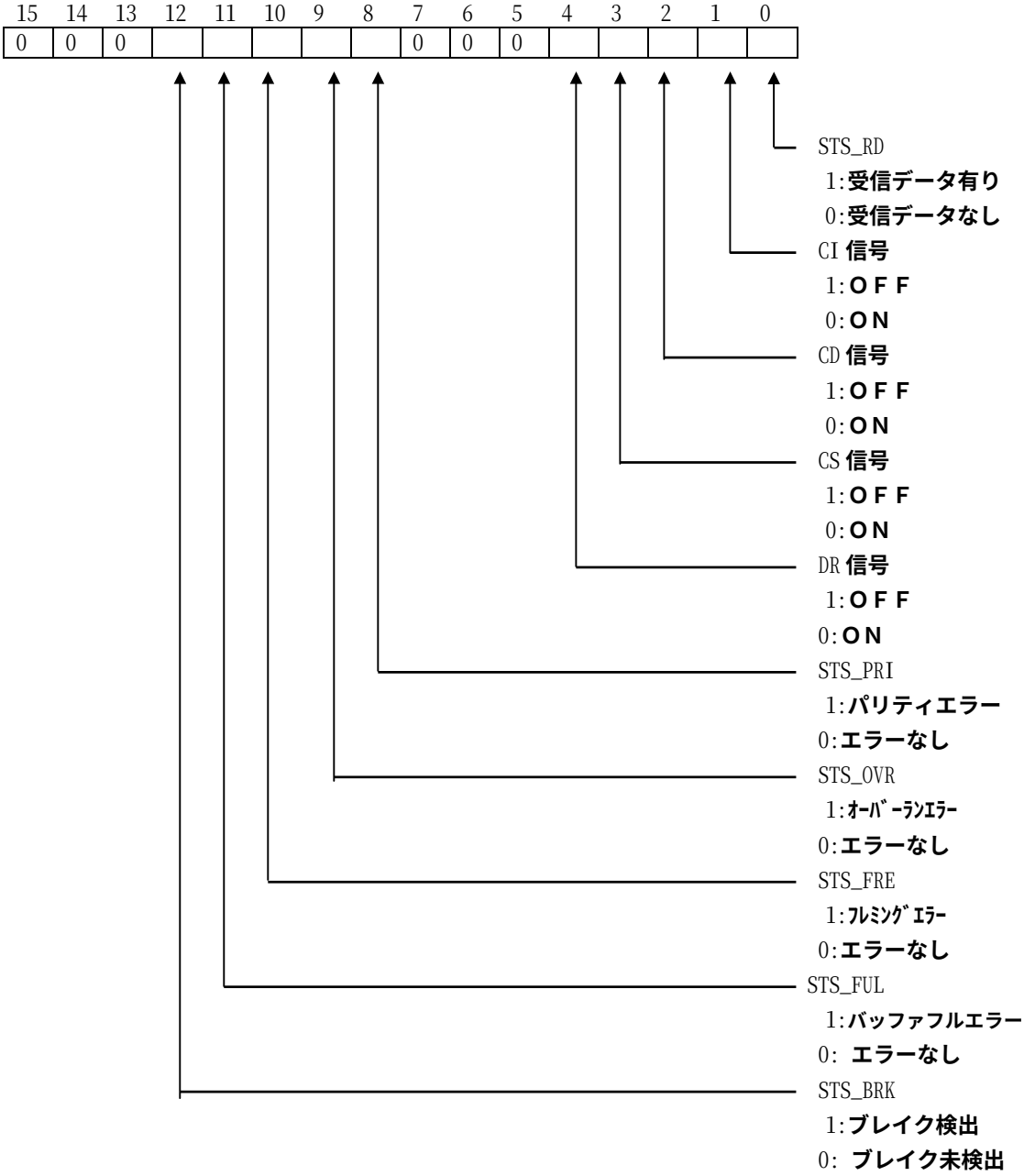
ER	ercd	：エラーコード

【エラーコード】

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

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:受信データ有り
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:バッファフルエラー
0: エラーなし
STS_BRK
1:ブレイク検出
0: ブレイク未検出

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	: COM NO.	
		COM0	：3WireIR
B	mode	: 占有状態設定	
		HOLD_ON	：占有
		HOLD_OFF	：占有解除

【戻り値】

ER	ercd	: エラーコード

【エラーコード】

E_OK	00000000h	: 正常終了
E_NG	FFFFFFFFh	: 異常終了
E_PRM	FFFFFFFEh	: 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み出します。各通信ポートがオープン／クローズ中であるかを知ることができます。
また、「ＣＯＭの占有」ファンクション及び「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３Ｗｉｒｅ ＩＲが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 );
```

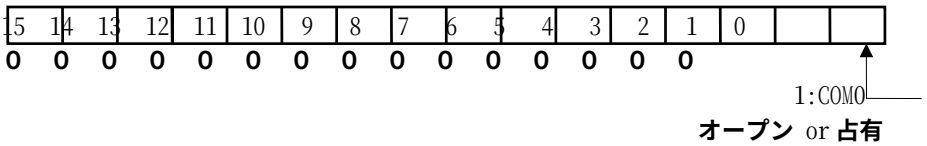
【パラメータ】

なし

【戻り値】

ER ercd : エラーコード(オープン or 占有通知)

【エラーコード】



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転送データの送信を指定の文字（ｂｙｔｅ）数だけ行います。
指定の送信文字数が０である場合はＮＵＬＬ文字をデリミッタとして送信（ＮＵＬＬ文字は送信しません）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	: ＣＯＭ　ＮＯ. COM0　：３ＷｉｒｅＩＲ
B	*buffer	: 送信バッファアドレス
H	length	: 送信文字数（バイト数）

【戻り値】

ER ercd	: エラーコード

【エラーコード】

E_OK	00000000h	: 正常終了
E_NG	FFFFFFFFh	: 異常終了
E_PRM	FFFFFFFEh	: 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。

機能	１文字受信	関数名	c_din

受信キューに格納された転送データを格納バッファへ１文字（ｂｙｔｅ）読出します。
また、読出し可能な転送データが存在しないときは、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	: **ＣＯＭ　ＮＯ.**
		COM0　　**：３ＷｉｒｅＩＲ**
B	*buffer	: **格納バッファアドレス**

【戻り値】

ER ercd	: **エラーコード**

【エラーコード】

E_OK	00000000h	: **正常終了**
E_NG	FFFFFFFFh	: **異常終了**
E_PRM	FFFFFFEh	: **パラメータエラー**

備考

機能	タイムアウト監視受信	関数名	c_tmdin

受信キューに格納された転送データ格納バッファへ１文字読出します。
受信キューに転送データ読出し可能な転送データ存在しないときは受信データ待ちとなり、受信タイムアウト監視値だけ受信データを待ちます。タイムアウト監視値に０を指定すると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	: ＣＯＭ　ＮＯ． COM0　　：３Ｗｉｒｅ ＩＲ
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単位）

【戻り値】

ER ercd　　: エラーコード

【エラーコード】

E_OK	00000000h	: 正常終了
E_NG	FFFFFFFFh	: 異常終了
E_PRM	FFFFFFFEh	: パラメータエラー

備考

機能	１文字送信	関数名	c_out

ユーザ指定の領域に格納された転送データ（送信文字）の送信を指定の１文字（ｂｙｔｅ）だけ行います。

Ｃ言語インタフェース

【コーリングシーケンス】

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

【パラメータ】

H	com_no	:	ＣＯＭ　ＮＯ.
			COM0　　：３ＷｉｒｅＩＲ
B	snddata	:	送信文字（１バイト）

【戻り値】

ER ercd	:	エラーコード

【エラーコード】

E_OK	00000000h	:	正常終了
E_NG	FFFFFFFFh	:	異常終了
E_PRM	FFFFFFFEh	:	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。

機能	ブレイク送出のｏｎ／ｏｆｆ	関数名	c_break

ブレイ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	：	ＣＯＭ　ＮＯ．
			COM0　　：３Ｗｉｒｅ ＩＲ
B	mode	：	ブレイクのｏｎ／ｏｆｆ
			BRK_ON　　：ブレイク信号を送出する
			BRK_OFF　　：ブレイク信号を停止する

【戻り値】

ER ercd　　：エラーコード

【エラーコード】

E_OK	00000000h	：正常終了
E_NG	FFFFFFFFh	：異常終了
E_PRM	FFFFFFEh	：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）／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とき転送データの送信を行うことが出来なくなります。
また、受信動作を無効に設定すると転送データの受信を行うことが出来なくなります。

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 : COM NO.
- COM0 ：3WireIR

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

【戻り値】

ER ercd : エラーコード

【エラーコード】

E_OK	00000000h	: 正常終了
E_NG	FFFFFFFFh	: 異常終了
E_PRM	FFFFFFFEh	: パラメータエラー

備考

機能	受信バッファのクリア	関数名	c_flush

受信キュー内と通信コントローラのレシーバ内の転送データを破棄し、初期化します。
また「エラーステータスのリード」、「COMステータスのリード」ファンクションで通知するステータスのアトリビュートであるパリティ，フレーミング，オーバーラン，バッファフルエラーを保持（発生）していればこれをクリアします。
ただし、クリア後のファンクション終了時エラーのチェックでこれらのエラーの検出を行います。

C言語インタフェース

【コーリングシーケンス】

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

【パラメータ】

H com_no	: COM NO.	
	COM0	：3WireIR

【戻り値】

ER ercd	: エラーコード

【エラーコード】

E_OK	00000000h	: 正常終了
E_NG	FFFFFFFFh	: 異常終了
E_PRM	FFFFFFFEh	: 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

受信キュー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。
なお、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。

・受信キューに格納されている読出し可能な転送データ数
（受信文字数：ｂｙｔｅ単位）
・受信キューの先頭に格納されている転送データの文字コード（次読出し文字）
・受信キューに格納できる転送データ数（受信可能文字数：ｂｙｔｅ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          : ＣＯＭ　ＮＯ.
                            COM0        ：3ＷｉｒｅＩＲ
COM_STS *bfsts          : 受信バッファステータス
        typedef  struct {
                 H      char_no  ,      ：受信文字数
                 H      rest_no  ,      ：受信可能残り文字数
                 UB     char_cod        ：先頭文字コード
        } COM_STS ;
```

【戻り値】

```
ER ercd       : エラーコード
```

【エラーコード】

```
E_OK    00000000h   : 正常終了
E_NG    FFFFFFFFh   : 異常終了
E_PRM   FFFFFFFEh   : 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

エラー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キューへ格納します。通信エラーとなった受信データは受信キューに格納しません。
「ＣＯＭのオープン」ファンクションでは既定の処理が本機能を制御しない設定になっています。

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	: ＣＯＭ　ＮＯ.	
		COM0	：３ＷｉｒｅＩＲ
B	mode	: 設定／解除	
		ERRCD_ON	：制御する
		ERRCD_OFF	：制御しない
UB	c_errcd	: エラーコード（任意の１バイトコード）	

【戻り値】

ER ercd	: エラーコード

【エラーコード】

E_OK	00000000h	: 正常終了
E_NG	FFFFFFFFh	: 異常終了
E_PRM	FFFFFFEh	: 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及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マスクして参照して下さい。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	: COM NO.
		COM0 ：3WireIR
UW	*com_status	: エラーステータス

【戻り値】

ER ercd	: エラーコード

【エラーコード】

E_OK	00000000h	: 正常終了
E_PRM	FFFFFFFEh	: パラメータエラー

備考

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	: ＣＯＭ　ＮＯ.	
		COM0	：３ＷｉｒｅＩＲ
B	mode	: 設定／解除	
		STAND_HDR	：標準受信ハンドラ設定
		HIGH_HDR	：簡易受信ハンドラ設定

【戻り値】

ER ercd	: エラーコード	

【エラーコード】

E_OK	00000000h	: 正常終了
E_PRM	FFFFFFFEh	: 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時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、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	:	ＣＯＭ　ＮＯ． COM0　　：３Ｗｉｒｅ ＩＲ
H	cs_time	:	ＣＳタイムアウト監視値設定
H	dr_time	:	ＤＲタイムアウト監視値設定
H	cd_time	:	ＣＤタイムアウト監視値設定 ０～３２７６７（７．８ｍｓ）

【戻り値】

ER ercd	:	エラーコード

【エラーコード】

E_OK	00000000h	:	正常終了
E_NG	FFFFFFFFh	:	異常終了
E_PRM	FFFFFFFEh	:	パラメータエラー

備考

機能	ＥＲ信号のＯＮ／ＯＦＦ	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	：	ＣＯＭ　ＮＯ．
			COM0　　：３ＷｉｒｅＩＲ
B	er_mode	：	ＥＲ信号線の設定
			ERS_ON　　：ＥＲ信号ＯＮ
			ERS_OFF　　：ＥＲ信号ＯＦＦ

【戻り値】

ER ercd	：	エラーコード

【エラーコード】

E_OK	00000000h	：	正常終了
E_NG	FFFFFFFFh	：	異常終了
E_PRM	FFFFFFFEh	：	パラメータエラー

備考

機能	ＲＳ信号のＯＮ／ＯＦＦ	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	: ＣＯＭ　ＮＯ.	
		COM0	：３ＷｉｒｅＩＲ
B	mode	: 信号線の設定	
		RS_ON	：ＲＳ信号ＯＮ
		RS_OFF	：ＲＳ信号ＯＦＦ

【戻り値】

ER ercd	: エラーコード	

【エラーコード】

E_OK	00000000h	: 正常終了
E_NG	FFFFFFFFh	: 異常終了
E_PRM	FFFFFFFEh	: パラメータエラー

備考

機能	ＥＲ／ＲＳ信号のＯＮ／ＯＦＦ	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	:	**ＣＯＭ　ＮＯ.**
			COM0　　**：３ＷｉｒｅＩＲ**
B	mode	:	**信号線の設定**
			ERRS_ON　　**：ＥＲ／ＲＳ信号ＯＮ**
			ERRS_OFF　　**：ＥＲ／ＲＳ信号ＯＦＦ**

【戻り値】

ER ercd　　: **エラーコード**

【エラーコード】

E_OK	00000000h	: **正常終了**
E_NG	FFFFFFFFh	: **異常終了**
E_PRM	FFFFFFFEh	: **パラメータエラー**

備考

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又は解除を行います。

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
			MLTR_R,MLTR_L：マルチファンクションキーＲ、Ｌ

【戻り値】

ER ercd	:	エラーコード

【エラーコード】

E_OK	00000000h	:	正常終了
E_PRM	FFFFFFFEh	:	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。

7 IrDA 部

7.1 通信仕様

7.1.1 通信インタフェース

（1）通信ポート

ＩｒＤＡポートに対する機能を記載します。

表６．１　通信ポートと機能一覧

通信ポート	制御形式	規格	同期方式	転送速度(bps)	キャラクタレングス	パリティビット	ストップビット	信号
IrDA	半二重	IrDA (IrSIR 1.1)	調歩同期式	9.6k 19.2k 38.4k 57.6k 115.2k	8bit	NON	1bit	SD RD
			同期式	4M	―	―	―	

※―は対応する項目がないことをあらわします。

7.2 機能

7.2.1 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。

7.2.2 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。
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（バッファビジー）、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、ある程度の時間がかかります。
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）エラーステータス取得

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

（8）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。

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

（9）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

（10）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。

7.2.3 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。

表６．２　ファンクション優先順位

プライオリティ	ファンクションコール名	備考
5	Ir_State_set, Ir_SetPortConfig	・プライオリティ 4 以下のファンクションより先に使用して下さい ・リセット直後はデフォルト値となります ・クローズ状態である必要があります ・設定値はリセットされるまで有効です
4	reserve	・予約とします
3	Ir_Err_Get	・プライオリティ 2 以下のファンクションと同じ優先順位で使用できます
2	Ir_Open	・クローズ状態である必要があります
1	Ir_Close, Ir_Read, Ir_Write, Ir_QueryTx,Ir_QuertRx, Ir_Init	・オープン状態である必要があります

7.3 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	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　　・通信用のデバイス及びリソースのロック
・赤外線デバイス電源ＯＮ　　・ブレー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　：秒単位に待ち時間を設定
　FOREVER　：正常または異常終了するまでコネクト待ちを行う

【リターンコード】

H　ercd　：終了情報
　E_IROK　：正常終了
　E_IRNG　：異常終了

【エラーステータス】

IRERR_NORESOURCE　：資源不足
IRERR_NODEVICE　：接続可能デバイスなし
IRERR_NOLSAP　：接続先サービスなし
IRERR_DISCONNECT　：コネクト失敗、切断
IRERR_TIMEOUT　：送信又は受信タイムアウト
IRERR_LOCK　：通信デバイスロック
IRERR_PARAMETER　：パラメータエラー
IRERR_LB0　：LB0
IRERR_LB1　：LB1
IRERR_LB4　：LB4
IRERR_LB5　：LB5
IRERR_BREAK_EVNT　：ブレークイベント発生

≪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。
・通信用のデバイス及びリソースのリリース
・赤外線デバイス電源ＯＦＦ
・ブレー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

【リターンコード】

H	ercd	：終了情報
	E_IROK	：正常終了
	E_IRNG	：異常終了

【エラーステータス】

IRERR_DISCONNECT	：コネクト失敗、切断
IRERR_NOTOPEN	：未オープン
IRERR_LB0	：LB0
IRERR_LB1	：LB1
IRERR_LB4	：LB4
IRERR_LB5	：LB5
IRERR_BREAK_EVNT	：ブレークイベント発生

≪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
・ブレー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）

【リターンコード】

H	ercd	：終了情報
	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。）

【エラーステータス】

IRERR_DISCONNECT	：コネクト失敗、切断
IRERR_TIMEOUT	：送信又は受信タイムアウト
IRERR_NOTOPEN	：未オープン
IRERR_LB0	：LB0
IRERR_LB1	：LB1
IRERR_LB4	：LB4
IRERR_LB5	：LB5
IRERR_BREAK_EVNT	：ブレークイベント発生

≪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）
・ブレー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）

【リターンコード】

H	ercd	：終了情報
	E_IROK	：正常終了
	E_IRNG	：異常終了
		（状態によってはデータの書き込みが行われています。）

【エラーステータス】

IRERR_DISCONNECT	：コネクト失敗、切断
IRERR_TIMEOUT	：送信又は受信タイムアウト
IRERR_NOTOPEN	：未オープン
IRERR_LB0	：LB0
IRERR_LB1	：LB1
IRERR_LB4	：LB4
IRERR_LB5	：LB5
IRERR_BREAK_EVNT	：ブレークイベント発生

≪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
・ブレー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）
```

【リターンコード】

```
H     ercd          ：終了情報
                    E_IROK        ：正常終了
                    E_IRNG        ：異常終了
```

【エラーステータス】

```
IRERR_DISCONNECT              ：コネクト失敗、切断
IRERR_NOTOPEN                 ：未オープン
IRERR_LB0                     ：LB0
IRERR_LB1                     ：LB1
IRERR_LB4                     ：LB4
IRERR_LB5                     ：LB5
IRERR_BREAK_EVNT              ：ブレークイベント発生
```

≪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
・ブレー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）
```

【リターンコード】

```
H     ercd              ：終了情報
                E_IROK      ：正常終了
                E_IRNG      ：異常終了
```

【エラーステータス】

```
IRERR_DISCONNECT          ：コネクト失敗、切断
IRERR_NOTOPEN             ：未オープン
IRERR_LB0                 ：LB0
IRERR_LB1                 ：LB1
IRERR_LB4                 ：LB4
IRERR_LB5                 ：LB5
IRERR_BREAK_EVNT          ：ブレークイベント発生
```

≪備考≫

機能	エラーステータス取得	関数名	Ir_Err_Get

エラーステータスを取得します。また取得後にエラーステータスをクリアします。
・エラーステータスのクリア
・エラーステータスの通知

≪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

【リターンコード】

UW　wercd　：エラーステータス

【エラーステータス】

IRERR_NORESOURCE	：資源不足
IRERR_NODEVICE	：接続可能デバイス
IRERR_NOLSAP	：接続先サービスなし
IRERR_DISCONNECT	：コネクト失敗、切断
IRERR_TIMEOUT	：送信又は受信タイムアウト
IRERR_LOCK	：通信デバイスロック
IRERR_NOTOPEN	：未オープン
IRERR_PARAMETER	：パラメータエラー
IRERR_LB0	：LB0
IRERR_LB1	：LB1
IRERR_LB4	：LB4
IRERR_LB5	：LB5
IRERR_WIRE_TYPE	：Wire 指定が不正
IRERR_BREAK_EVNT	：ブレークイベント発生

≪備考≫

エラーステータスについて詳しくは次ページを参照して下さい。

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 ステータス	エラーステータス名称	
詳細 エラーの詳細		
関数名	IrCOMM 状態	主なエラー対処方法
エラーの 発生する関数名	関数異常終了時の IrCOMM オープン状態	ＩｒＤＡ部の上位が行う発生したエラーに対しての事後処理

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

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

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

エラーステータス	ＩＲＥＲＲ＿ＤＩＳＣＯＮＮＥＣＴ

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

関数名	IrCOMM 状態	主なエラー対処方法
Ir_Open	オープンは行わずクローズ状態	・通信環境を確認して再実行して下さい
Ir_Close	クローズ状態となります	・相手局と通信不可能な環境にあるのでその原因を取り除いてIrCOMM(赤外線ポート)のオープンを行って下さい 相手局から一定時間応答がない（回線が外れている）場合が考えられます
Ir_Read		
Ir_Write		
Ir_QueryTx		
Ir_QueryRx		
Ir_Init		

エラーステータス	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_State_set		
Ir_SetPortConfig		

エラーステータス	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ークイベントの検出がＩｒＤＡ部で行われたときに発生します

関数名	IrCOMM 状態	主なエラー対処方法
Ir_Open	オープンは行わずクローズ状態	・再実行する場合は Ir_Open から行って下さい
Ir_Close	クローズ状態となります	
Ir_Read	オープン状態から変更はありません	・Ir_Close を行って終了して下さい
Ir_Write		
Ir_QueryTx		
Ir_QueryRx		

エラーステータス	ＩＲＥＲＲ＿ＬＢＯ

詳細
電源関数の機能を用いてＬＢＯの通知モードに設定されており、ＬＢＯ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	オープンは行わずクローズ状態	・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	クローズ状態となります	
Ir_Read		
Ir_Write		
Ir_QueryTx		
Ir_QueryRx		
Ir_Init		

エラーステータス	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	オープンは行わずクローズ状態	・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	クローズ状態となります	
Ir_Init		
Ir_Read	オープン状態から変更はありません。	・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_QueryTx		
Ir_QueryRx		

エラーステータス	ＩＲＥＲＲ＿ＬＢ４

詳細
電源関数の機能を用いてＬＢ４の通知モードに設定されており、ＬＢ４エラー(APO 発生)となったときに発生します
通知モードに設定されているときは電源ＯＦＦしませんので、アプリケーションが責任を持つ必要があります

関数名	IrCOMM 状態	主なエラー対処方法
Ir_Open	オープンは行わずクローズ状態	・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	クローズ状態となります	
Ir_Init		
Ir_Read	オープン状態から変化はありません	・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_QueryTx		
Ir_QueryRx		

エラーステータス	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	オープンは行わずクローズ状態	・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	クローズ状態となります	
Ir_Init		
Ir_Read	オープン状態から変化はありません	・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_QueryTx		
Ir_QueryRx		

エラーステータス	ＩＲＥＲＲ＿ＮＯＴＯＰＥＮ

詳細
ＩｒＣＯＭＭ（赤外線ポート）がオープンされていないときに発生します

関数名	IrCOMM 状態	主なエラー対処方法
Ir_Close	クローズ状態から変化はありません	・IrCOMM が既にクローズ状態になっています Ir_Open を行ってから実行して下さい
Ir_Read		
Ir_Write		
Ir_QueryTX		
Ir_QueryRx		
Ir_Init		

エラーステータス	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	クローズ状態です	・通信環境を確認して再実行して下さい
Ir_Read	オープン状態に変化はありません	・任意の処理を行って下さい。再実行してもかまいません
Ir_Write		

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
・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    ：通信状態設定構造体
```

【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
};
```

【リターンコード】

```
H     ercd          ：終了情報
                      E_IROK     ：正常終了
                      E_IRNG     ：異常終了
```

【エラーステータス】

```
IRERR_PARAMETER                  ：パラメータエラー
```

≪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	備考
局	PRIMARY	自局を１次局に設定	デフォルト
	SECONDARY	自局を２次局に設定	
Wire	WIRE3	3-wire に設定	デフォルト
データ待ち時間	1-600	秒単位にデータ読込み/書込み待ち時間を設定	デフォルト(300)
	THROUGH	データ読込み/書込み待ちを行わない	
	FOREVER	タイマ指定なしでデータ読込み/書込み待ちを行う	

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        ：自局能力設定構造体
```

【ストラクト構造】

```
struct SetPortConfig_DCB {
        UB     irBaud;           ：ボーレート
        UB     MaxTurnTime;      ：最大ターンアラウンドタイム
        UB     FrameSize;        ：フレームサイズ
        UB     WindowSize;       ：ウインドウサイズ
        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
};
```

【リターンコード】

```
H      ercd          ：終了情報
                       E_IROK     ：正常終了
                       E_IRNG     ：異常終了
```

【エラーステータス】

```
IRERR_PARAMETER                  ：パラメータエラー
```

≪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	備考
IR ボーレート ・OR で設定して下さい。設定した値が有効となります。	IRBPS_96	IR 接続速度を 9600bps に設定可能	デフォルトは設定有効
	IRBPS_192	IR 接続速度を 19200bps に設定可能	デフォルトは設定有効
	IRBPS_384	IR 接続速度を 38400bps に設定可能	デフォルトは設定有効
	IRBPS_576	IR 接続速度を 57600bps に設定可能	デフォルトは設定有効
	IRBPS_1152	IR 接続速度を 115200bps に設定可能	デフォルトは設定有効
	IRBPS_4M	IR 接続速度を 4Mbps に設定可能	デフォルトは設定有効
最大ターンアラウンドタイム	TURN_500MS	最大ターンアラウンドタイムを 500ms に設定	デフォルト
フレームサイズ	FRAME_2048B	フレームサイズを 2048byte に設定	デフォルト
ウインドウサイズ	WINDOW_1	ウインドウサイズを１フレームウインドウに設定	デフォルト
BOF 数 ・BOF 数は 115.2k の場合で､他のスピードでは自動調整されます	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	デフォルト
	BOF_0	BOF を 0 個追加	
最小ターンアラウンドタイム	TURN_1MS	最小ターンアラウンドタイムを 1ms に設定	デフォルト
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	デフォルトは設定有効
	RELEASE_8S	リンクを開放する時間を 8s に設定可能	デフォルトは設定有効
	RELEASE_12S	リンクを開放する時間を 12s に設定可能	デフォルトは設定有効
	RELEASE_16S	リンクを開放する時間を 16s に設定可能	デフォルトは設定有効
	RELEASE_20S	リンクを開放する時間を 20s に設定可能	デフォルトは設定有効
	RELEASE_25S	リンクを開放する時間を 25s に設定可能	デフォルトは設定有効
	RELEASE_30S	リンクを開放する時間を 30s に設定可能	デフォルトは設定有効
	RELEASE_40S	リンクを開放する時間を 40s に設定可能	デフォルトは設定有効

機能	ＩｒＣＯＭＭ強制終了	関数名	Ir_Init

ＩｒＣＯＭＭを強制終了します。
・通信用のデバイス及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

【リターンコード】

H	ercd	：終了情報
	E_IROK	：正常終了
	E_IRNG	：異常終了

【エラーステータス】

IRERR_DISCONNECT	：コネクト失敗、切断
IRERR_NOTOPEN	：未オープン
IRERR_LB0	：LB0
IRERR_LB1	：LB1
IRERR_LB4	：LB4
IRERR_LB5	：LB5

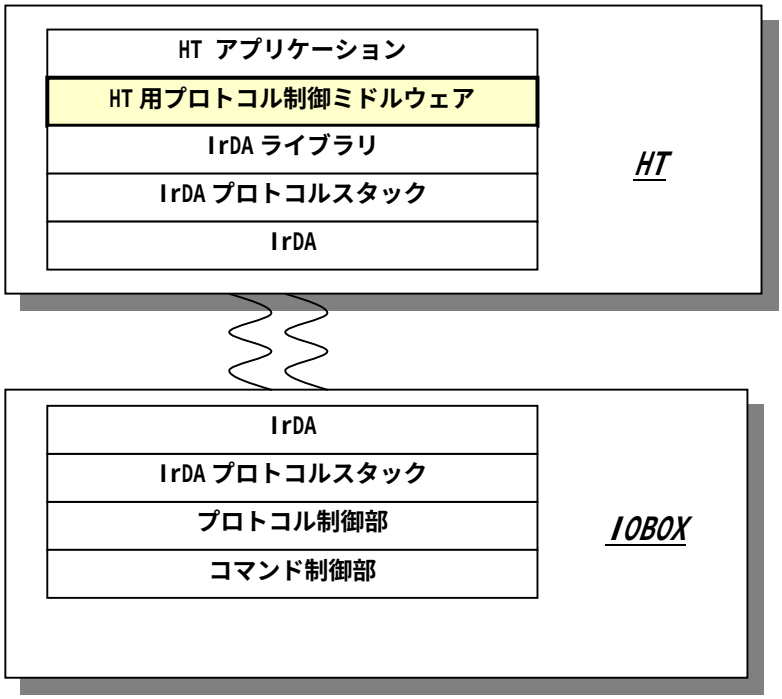
≪備考≫

8 通信ユーティリティ部

8.1 HIOWIN プロトコル

8.1.1 システム構成

HIOWIN プロトコルは HT（以後は本機を HT と呼びます）から IOBOX へ通信するために使用するものです。
HT－IOBOX 間の構成と HT 用ミドルウェア（本章で提供する関数）の位置付けを下記に示します。



8.1.2 関数一覧

表７．１　HIOWIN プロトコル関数一覧

通信ポートのオープン／クローズ	
HIO_PortOpen	通信ポートのオープン
HIO_PortClose	通信ポートのクローズ
ファイルの送受信／削除／メモリ初期化	
HIO_ReceiveFile	IOBOX からファイルを受信
HIO_SendFile	IOBOX へファイルを送信
HIO_DeleteFile	IOBOX のファイルの削除
HIO_MemoryFormat	IOBOX のメモリ初期化
各種情報の取得	
HIO_GetSysInfo	IOBOX のシステム情報の取得
HIO_GetMemoryInfo	IOBOX のメモリ情報の取得
HIO_GetFileInfo	IOBOX の全ファイル情報の取得（INDEX／機器 ID／ファイル名を指定）
HIO_GetLastErrState	最後に発生したエラーの詳細を取得します。
設定／保守	
HIO_GetLogData	IOBOX のログデータの取得
HIO_ClearLogData	IOBOX のログデータの削除
HIO_SetMemoryThresh	IOBOX のキャッシュメモリの上限設定
HIO_SetIOBOXID	IOBOX の ID を設定
HIO_WriteFirmware	IOBOX のファームウェアの書換
HIO_ResetIOBOX	IOBOX のリセット
HIO_SetDispParam	表示パラメータの設定

8.1.3 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	通信ポートのオープン	関数名	HIO_PortOpen

指定された通信ポート(IrDA)をオープンし、デフォルトのパラメータで初期化します。

≪C言語インタフェース≫

【コーリングシーケンス】

```
ER    ercd = HIO_PortOpen (H ConnectTime);
```

【パラメータ】

H　ConnectTime　：コネクト最大待ち時間　(1～3600 秒)
※ただし FOREVER を指定すると、正常または異常終了するまでコネクト待ちを行います。

【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　：通信ポートオープン成功
E_PRM　：パラメータエラー
E_NG　：異常終了(通信ポートオープン失敗)

≪備考≫

機能	通信ポートのクローズ	関数名	HIO_PortClose

オープンした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 = HIO_PortClose(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	:通信ポートクローズ成功
E_NG	:通信ポートクローズ失敗

≪備考≫

機能	IOBOX からファイルを受信	関数名	HIO_ReceiveFile

IOBOX から指定したファイルを受信します。受信したファイルは指定したフォルダに保存されます。
送信元（IOBOX）のファイルは削除しません。関数を実行する前に HIO_GetFileInfo 関数を実行して、
ファイル情報を取得して下さい。

≪C 言語インタフェース≫

【コーリングシーケンス】

```
ER    ercd = HIO_ReceiveFile(IOFILEINFO *Iofileinfo, B *Folder, H Option, B disp);
```

【パラメータ】

```
struct _iofileinfo {            /*  IOBOX のファイル情報を格納する構造体です   */
      UH    Index;              データ受信の順番
      UB    Id[16];             機器 ID
      B     FileName[8];        ファイル名
      B     FileExt[3];         ファイル拡張子
      UW    FileSize            ファイルサイズ（単位は Byte）
      UH    FileDay;            ファイルの日付
      UH    FileTime;           ファイルの時刻
      UB    Attr;               ファイル属性
} IOFILEINFO;

B     *Folder     :受信先のフォルダ名
                  フォルダ名はフルパスで'\'まで指定します。

H     Option      :オプションフラグ（メモリ種類の指定）
                  IO_MEMORY_RAM     :RAM から受信
                  IO_MEMORY_FROM    :FROM から受信

 B     disp       : 受信進捗バー切替
                  0                 :非表示
                  1                 :表示
```

【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          :  ファイル受信成功
E_PRM         :  パラメータエラー
E_NG          :  異常終了
```

≪備考≫

受信するフォルダに同一ファイル名のファイルがある場合は上書きを行います。
また、受信時は受信するファイルのファイルサイズ分の空き領域が必要になります。

機能	IOBOX へファイルを送信	関数名	HIO_SendFile

HT から IOBOX へ指定したファイルを送信します。送信したファイルのタイムスタンプおよび属性は、送信元のファイルと同一になります。ファイル構造体の INDEX メンバには 0x00 を設定します。関数を実行すると、ファイル構造体に IOBOX に送信されたファイルの情報が、格納されます。エラーによるファイルの再送信を行う場合には、ファイル構造体の情報を変更せずにもう一度関数を実行してください。(INDEX メンバの変更不要)。

≪C 言語インタフェース≫

【コーリングシーケンス】

```
ER ercd = HIO_SendFile (B *Filename, IOFILEINFO *Iofileinfo, H Option, B disp);
```

【パラメータ】

B *Filename　　:HT から送信するファイル名
※ファイル名はフルパスでファイルの拡張子まで指定してください。

```
struct  _iofileinfo{
UH                  Index;          データ受信の順番
UB                  Id[16];         機器 ID
B                   FileName[8];    ファイル名
B                   FileExt[3];     ファイル拡張子
UW                  FileSize;       ファイルサイズ（単位は Byte）
UH                  FileDay;        ファイルの日付
UH                  FileTime;       ファイルの時刻
UB                  Attr;           ファイル属性
}IOFILEINFO;
```

H　Option　　:オプションフラグ（メモリ種類の指定）
IO_MEMORY_RAM　:RAM へ送信
IO_MEMORY_FROM　:FROM へ送信

B　disp　　: 送信進捗バー切替
0　　:非表示
1　　:表示

【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　　:　ファイル送信成功
E_PRM　　:　パラメータエラー
E_NG　　:　異常終了

≪備考≫

機能	IOBOX のファイルの削除	関数名	HIO_DeleteFile

IOBOX のファイルを削除します。関数を実行する前に HIO_GetFileInfo 関数を実行して、ファイル情報を取得しておく必要があります。

≪C 言語インタフェース≫

【コーリングシーケンス】

```
ER     ercd = HIO_DeleteFile(IOFILEINFO *Iofileinfo, H Option);
```

【パラメータ】

```
struct  _iofileinfo{      /*  IOBOX のファイル情報を格納する構造体です   */
UH                  Index;          データ受信の順番
UB                  Id[16];         機器 ID
B                   FileName[8];    ファイル名
B                   FileExt[3];     ファイル拡張子
UW                  FileSize;       ファイルサイズ（単位は Byte）
UH                  FileDay;        ファイルの日付
UH                  FileTime;       ファイルの時刻
UB                  Attr;           ファイル属性
}IOFILEINFO;
H      Option:オプションフラグ（メモリ種類の指定）
                    IO_MEMORY_RAM       :RAM にあるファイルを削除
                    IO_MEMORY_FROM      :FROM にあるファイルを削除
```

【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_PRM           :  パラメータエラー
E_NG            :  異常終了
```

≪備考≫

機能	IOBOX のメモリ初期化	関数名	HIO_MemoryFormat

IOBOX のメモリをフォーマットします。

≪C 言語インタフェース≫

【コーリングシーケンス】

```
ER    ercd = HIO_MemoryFormat(H Option);
```

【パラメータ】

H	Option	:オプションフラグ（メモリ種類の指定）
	IO_MEMORY_RAM	:RAM をフォーマット
	IO_MEMORY_FROM	:FROM をフォーマット

【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_PRM	: パラメータエラー
E_NG	: 異常終了

≪備考≫

機能	IOBOX のシステム情報の取得	関数名	HIO_GetSysInfo

IOBOX のシステム情報を取得します。

≪C 言語インタフェース≫

【コーリングシーケンス】

```
ER    ercd =HIO_GetSysInfo(IOSYSINFO *Iosysinfo);
```

【パラメータ】

```
struct  _iosysinfo{   /*     システム情報を格納する構造体です       */
H                     IoboxId;          IOBOX の ID 番号
H                     IoboxMode;        IOBOX の動作モード
                                        0x00:マスター
                                        0x01:スレーブ
H                     IoboxEndStatus;   終端状態
                                        0x00:非終端
                                        0x01:終端
H                     HTStatus;         HT の状態
                                        0x00:HT 非装着
                                        0x01:HT 装着
UB                    IoboxFirmVer[7];  ファームウェアバージョン
UB                    IoboxSerial[16];  製造番号
}IOSYSINFO;
```

【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          :  システム情報取得成功
E_NG          :  異常終了
```

≪備考≫

※ファームウェア異常時に本関数を実行するとファームウェアバージョンの末尾が”B”となります。

機能	IOBOX のメモリ情報の取得	関数名	HIO_GetMemoryInfo

IOBOX のメモリ情報を取得します。

≪C 言語インタフェース≫

【コーリングシーケンス】

```
ER     ercd = HIO_GetMemoryInfo(IOMEMINFO *Iomeminfo);
```

【パラメータ】

```
struct  _iomemoryinfo{/*    IOBOX のメモリ情報を格納する構造体です        */
H                       RAMState;       RAM の状態
                                        0x00:フォーマット済み
H                       RAMFileNum;     RAM に保存されているファイル数
UW                      RAMFreeSize;    RAM の残りメモリ容量（単位は Byte）
H                       RAMThresh;      RAM のメモリ上限値（クラスタ単位）
H                       FROMState;      FROM の状態
                                        0x00:フォーマット済み
                                        0xFF:未フォーマット
H                       FROMFileNum;    FROM に保存されているファイル数
UW                      FROMFreeSize;   FROM の残りメモリ容量（単位は Byte）
H                       FROMThresh;     FROM のメモリ上限値（クラスタ単位）
}IOMEMINFO;
```

※ RAMState／FROMState は、電源 ON 時にデータが正しく保持されているか確認した結果を示します。保持されていなかった場合は、HIO_MemoryFormat を使用して RAM／FROM をフォーマットしなければなりません。

※ クラスタ単位については HIO_SetMemoryThresh 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            :  メモリ情報取得成功
E_NG            :  異常終了
```

≪備考≫

機能	IOBOX の全ファイル情報の取得	関数名	HIO_GetFileInfo

IOBOX にあるファイルのうち、該当するファイル INDEX 番号、機器 ID、ファイル名のファイル情報を
取得します。各パラメータの AND を取ることにより、該当するファイルがあるかどうかの判断をします。
一致するファイルがない場合には０が返ります。ファイル INDEX 番号、機器 ID、ファイル名のパラ
メータに特殊な値（NULL）を指定することにより、そのパラメータの全条件がファイル取得条件となります。

全ファイル情報の取得 → Fileindex = 0, Fileid = NULL, Filename=NULL

≪Ｃ言語インタフェース≫

【コーリングシーケンス】

```
ER    ercd = HIO_GetFileInfo(H Fileindex,B *Fileid, B *Filename, IOFILEINFO *Iofileinfo, H Option);
```

【パラメータ】

```
H     Fileindex          :IOBOX のファイル INDEX 番号（０～１２８）
B     *Fileid            :HT/ホスト PC の接続 ID
B     *Filename          :ファイル名
                    ※拡張子まで指定してください。
                    ※パスは含めないでください。
 struct  _iofileinfo{
 UH                 Index;          データ受信の順番
 UB                 Id[16];         機器 ID
 B                  FileName[8];    ファイル名
 B                  FileExt[3];     ファイル拡張子
 UW                 FileSize;       ファイルサイズ（単位は Byte）
 UH                 FileDay;        ファイルの日付
 UH                 FileTime;       ファイルの時刻
 UB                 Attr;           ファイル属性
 }IOFILEINFO;
H     Option:オプションフラグ（メモリ種類の指定）
                    IO_MEMORY_RAM     :RAM のファイル情報を取得
                    IO_MEMORY_FROM    :FROM のファイル情報を取得
```

【リターンパラメータ】

```
ER    ercd        ：取得したファイル情報の数（正常時）
                  ：リターンコード（エラー時）
```

【リターンコード】

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

≪備考≫

IOBOX からは、最大１２８ファイル分のファイル情報が返ります。Iofileinfo には IOFILEINFO128 個
分の領域を確保し、指定してください。

機能	最後のエラー詳細	関数名	HIO_GetLastErrState

最後に発生したエラーの詳細を取得します。

≪C言語インタフェース≫

【コーリングシーケンス】

```
void  HIO_GetLastErrState(IOERRSTATE *ioerrstate);
```

【パラメータ】

```
struct _ioerrstate{
            UW IrLibErr         IrLibrary 関数で発生したエラーの詳細（※）
            UW ProtocolErr      プロトコル制御で発生したエラーの詳細（※）
}IOERRSTATE
```

※エラー詳細は次ページを参照してください。

【リターンパラメータ】

なし

【リターンコード】

なし

≪備考≫

エラー詳細

IrLibErr		(備考)
IRERR_NOTOPEN	ポートオープンエラー	IrDA ライブラリのエラーコードが返ります。
IRERR_NORESOURCE	資源不足	
IRERR_NODEVICE	接続可能デバイスエラー	
IRERR_NOLSAP	接続先サービスエラー	
IRERR_DISCONNECT	コネクトの失敗、または切断	
IRERR_LOCK	通信デバイスロック	
IRERR_LB0	LB0 発生	
IRERR_LB1	LB1 発生	
IRERR_LB4	LB4 発生	
IRERR_LB5	LB5 発生	
IRERR_TIMEOUT	送信、または受信タイムアウト	
IRERR_BREAK_EVNT	ブレークイベント発生	
IRERR_PARAMETER	パラメータエラー	
IRERR_PARITY	パリティエラー	
IRERR_OVERRUN	オーバーランエラー	
IRERR_FRAMING	フレーミングエラー	

プロトコル制御エラー		(備考)
IOERR_NOCOMM	該当コマンド無し	
IOERR_COMMERR	コマンドエラー(他のコマンド処理中)	
IOERR_PACKETERR	パケット数エラー	
IOERR_SUMERR	パケットのチェックサムエラー	
IOERR_DATAERR	パケットのデータエラー	
IOERR_TIMEOUT	パケットの受信タイムアウト	
IOERR_MEMFULL	メモリフルエラー	
IOERR_NOFILE	該当ファイル無し	
IOERR_NOLOG	ログデータ無し	
IOERR_FILEINUSE	ファイル使用中	
IOERR_FILEOPEN	ファイルオープン中	
IOERR_FILEWRITE	ファイル書込みエラー	
IOERR_MEMINUSE	メモリ使用中	
IOERR_NONFORMAT	メモリ未フォーマット	
IOERR_SRCNODATA	コピー元データ無し	
IOERR_SRCNONFORMAT	コピー元メモリ未フォーマット	
IOERR_DSTNONFORMAT	コピー先メモリ未フォーマット	
IOERR_NONID	ID 未設定	
IOERR_FILESUM	ファイル異常	
IOERR_FWUPDATE	F/W アップデートエラー	
IOERR_MEMFORMATING	フォーマット中	
IOERR_HT_OPEN	ファイルオープンエラー	
IOERR_HT_CLOSE	ファイルクローズエラー	
IOERR_HT_WRITE	ファイルへの書き込みエラー	
IOERR_HT_READ	ファイルからの読み込みエラー	
IOERR_SUMCHECK	作成パケットのチェックエラー	
IOERR_QUERYTIME	パケットの読み込みタイムアウト	
IOERR_NOTRECVPACK	異常パケット受信	
IOERR_HT_MAKEDIR	ディレクトリ作成エラー	
IOERR_FILE_DEL	ファイル削除エラー	

機能	IOBOX のログデータの取得	関数名	HIO_GetLogData

IOBOX からログデータを取得します。

≪C 言語インタフェース≫

【コーリングシーケンス】

```
ER    ercd = HIO_GetLogData(B  *Filename, B disp);
```

【パラメータ】

B　*Filename　:ログを保存するファイル名
※ファイル名はフルパスでファイルの拡張子まで指定してください。

B　disp　: 受信進捗バー切替
0　：非表示
1　：表示

【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	:	ログデータ取得終了
E_PRM	:	パラメータエラー
E_NG	:	異常終了

≪備考≫

※受信するフォルダに同一ファイル名のファイルがある場合は上書きを行います。
受信時は受信するログデータ分の空き領域（最大８K バイト）が必要になります。

機能	IOBOX のログデータの削除	関数名	HIO_ClearLogData

IOBOX のログデータを消去します。

≪C 言語インタフェース≫

【コーリングシーケンス】

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

【パラメータ】

なし

【リターンパラメータ】

ercd　　:リターンコード

【リターンコード】

E_OK	:	ログデータ消去成功
E_NG	:	異常終了

≪備考≫

機能	IOBOX のキャッシュメモリの上限設定	関数名	HIO_SetMemoryThresh

IOBOX のキャッシュメモリの上限を設定します。

≪C 言語インタフェース≫

【コーリングシーケンス】

```
ER     ercd = HIO_SetMemoryThresh(H MemorySize, H Option);
```

【パラメータ】

H	MemorySize	:設定するメモリ上限の大きさ（クラスタ単位で指定します。） RAM:　０～２０１１（ただし０は上限値の初期化を行う際に設定します。） FROM:　０～２３９　（ただし０は上限値の初期化を行う際に設定します。） (※)
H	Option	:オプションフラグ（メモリ種類の指定） IO_MEMORY_RAM　　:RAM のメモリ上限を設定 IO_MEMORY_FROM　　:FROM のメモリ上限を設定

【リターンパラメータ】

ercd	:リターンコード

【リターンコード】

E_OK	:	キャッシュメモリ上限設定成功
E_PRM	:	パラメータエラー
E_NG	:	異常終了

≪備考≫

※　クラスタ単位は RAM,FROM で異なります。
　１クラスタはそれぞれ　　RAM:１０２４バイト
　　ROM:８１８６バイト です。

※　範囲外の値を設定するとリターンコードが E_PRM または E_NG の場合、IOERRSTATE 構造体には
　プロトコルエラー（IOERR_DATAERR）が設定されます。

機能	IOBOX の ID を設定	関数名	HIO_SetIOBOXID

IOBOX の ID 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 = HIO_SetIOBOXID(H    Ioboxid);
```

【パラメータ】

H	Ioboxid	:IOBOX に設定する ID の値（１～２５３）

【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	:	IOBOXID 設定成功
E_PRM	:	パラメータエラー
E_NG	:	異常終了

≪備考≫

※FROM のフォーマットを行っても ID は消去されません。
　消去する場合は ID として 0 を設定してください.

機能	IOBOX のファームウェアの書き換え	関数名	HIO_WriteFirmware

HT からファイルを送信して、IOBOX のファームウェアを更新します。

≪C 言語インタフェース≫

【コーリングシーケンス】

```
ER    HIO_WriteFirmware(B *FirmwareFilename, B disp);
```

【パラメータ】

```
B     *FirmwareFilename        :ファームウェアのファイル名
            ※ファイル名はフルパスでファイルの拡張子まで指定してください。

B     disp       : 送信進捗バー切替
                   0                    :非表示
                   1                    :表示
```

【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          :  ファームウェア更新成功
E_PRM         :  パラメータエラー
E_NG          :  異常終了
```

≪備考≫

HIO_WriteFirmware を実行すると IOBOX はリセットするため、HT との通信(IrDA)が切断されます。
HIO_WriteFirmware 後は HIO_PortClose を実行して下さい。再び通信を行う場合(書換後の Version 確認等を含む)は、HIO_PortOpen を実行して下さい。

機能	IOBOX のリセット	関数名	HIO_ResetIOBOX

IOBOX をリセットします。

≪C 言語インタフェース≫

【コーリングシーケンス】
ER ercd = HIO_ResetIOBOX(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 : リセット成功
E_NG : 異常終了

≪備考≫
HIO_ResetIOBOX を実行すると IOBOX はリセットするため、HT との通信(IrDA)が切断されます。
HIO_Reset 後は HIO_PortClose を実行して下さい。再び通信を行う場合は、HIO_PortOpen を実行して下さい。

機能	表示パラメータの設定	関数名	HIO_SetDispParam

通信中のファイル名と通信状態進捗バーを表示するための位置パラメータを設定します。
(HIO_SendFile，HIO_ReceiveFile，HIO_GetLogData，HIO_WriteFirmware 関数実行時に表示されます)
HIO_PortOpen を使用する前に設定してください。

≪C 言語インタフェース≫

【コーリングシーケンス】

```
ER    ercd = HIO_SetDispParam(H DispXParam, H DispYParam, H DispBarLine);
```

【パラメータ】

H	DispXParam	:ファイル名の X 軸方向の表示位置（桁数で指定）
H	DispYParam	:ファイル名の Y 軸方向の表示位置（行数で指定）
H	DispBarLine	:進捗バーの表示行

【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】

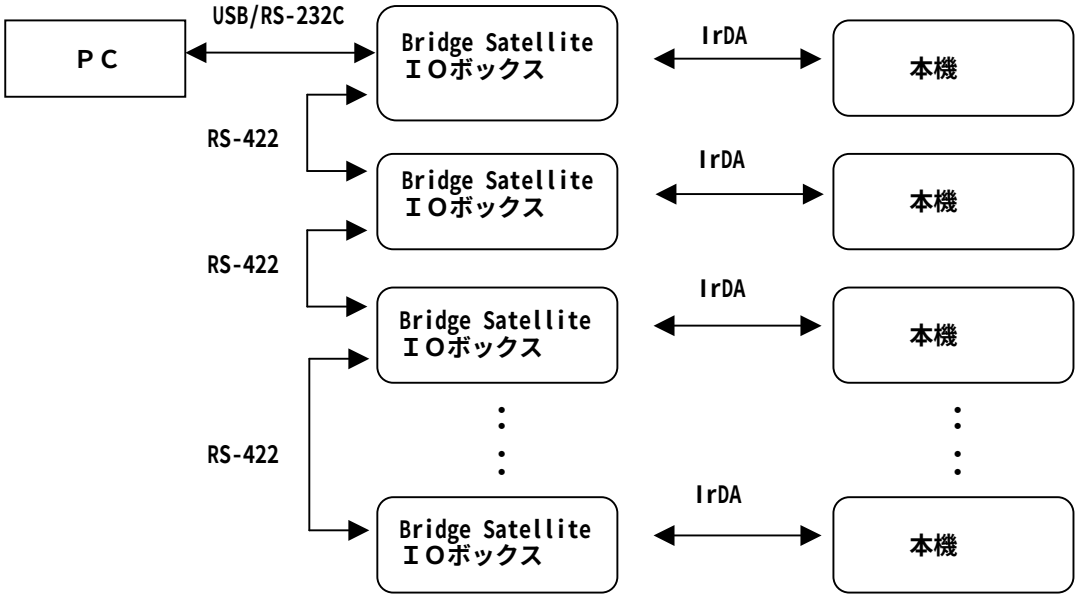
E_OK	:	位置設定の成功
E_PRM	:	パラメータエラー
E_NG	:	異常終了

≪備考≫

8.2 FLINK プロトコル

8.2.1 システム構成

FLINK プロトコルは HT（以後は本機を HT と呼びます）とホストを通信するために使用するものです。
I/O ボックスは DT-3641O（Bridge Satellite I/O）を使用します。



8.2.2 関数一覧

表７．２　FLINK プロトコル関数一覧

ファイル送受信基本機能関数		
	cu_open	通信ポートの初期化
	cu_fileSend	ファイル送信
	cu_fileAdd	ファイル追加
	cu_fileRecv	ファイル受信
	cu_close	通信ポートのクローズ
	cu_readErrStat	エラー詳細情報の取得
	cu_idle	ＩＤＬＥ遷移
	cu_cmdRecv	ＰＣモードコマンド待ち
	cu_stopKeySet	中断キーの登録／削除
	cu_setDrive	転送ドライブ指定
リモート操作機能関数		
	cu_fileDelete	ファイル／ディレクトリ削除
	cu_fileMove	ファイル移動
	cu_makeDir	ディレクトリ作成
	cu_getFileInfo	ファイル情報の取得
	cu_setFileInfo	ファイル情報の更新
	cu_getDiskInfo	ディスク情報の取得
	cu_dateTime	日付時刻の取得／設定
	cu_getSysInfo	システム情報の取得
	cu_msgSend	画面表示メッセージの送信
	cu_beep	ブザー鳴動
ファイルチェック関数		
	cu_fchklog_Create	ファイルチェックリスト生成
	cu_fchklog_Check	ファイルチェックリスト検査

8.2.3 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	回線オープン(初期化)	関数名	cu_open

通信ポートの初期化及びセッションの確立を行う。
セッション確立までは、タイムアウト時間まで待つ。
相手局ｼｽﾃﾑ情報の取得を行う。

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)
```

[入力パラメータ]

```
H          comNo   ：COM 番号
                       COM0：ポート０
H          irSpeed       ：赤外通信最高速度（自動設定なので指定は無効）
CU_RSPRM       *rsPrm    ：10 ﾋﾟﾝ通信パラメータ（COM0 しかないので無効）
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として動作を行う。）
```

```
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,CU_PARI_ODD,CU_PARI_EVN /*パリティビット*/
   H stop_bit;  CU_STOP1,CU_STOP2                    /*ストップビット*/
}CU_RSPRM;
```

[リターンパラメータ]

```
ER         ercd    ：処理ステータス
                     E_OK：正常　E_NG：異常　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る。

[関 連 関 数]

cu_close

機 能	ファイルの送信	関数名	cu_fileSend

指定された複数ファイルを一括して送信する。
転送先ディレクトリが存在しない場合は自動的に生成する。
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る。

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)
```

[入力パラメータ]

```
H   comNo  ：COM 番号   COM0：ポート0
H   mode   ：転送モード（通常転送か再帰呼出し転送かを指定する。）
             CU_TRANS_NORMAL：通常転送  CU_TRANS_RECURSIVE：再帰呼出し
B   *fName：送信ファイル名エリア（複数指定及びワイルドカード可）
B   *dir   ：送信先ディレクトリ名エリア（複数指定及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。）
             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
```

```
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ない)
  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;
```

[リターンパラメータ]

```
ER        ercd          ：処理ステータス
                        E_OK：正常  E_NG：異常  E_PRM：パラメータエラー
```

備　考

[関 連 関 数]

cu_open,cu_close

機　能	ファイルの追加	関数名	cu_fileAdd

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する。
送信元、追加先ファイル名とも複数ファイル指定,ワイルドカード指定は不可。
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する。
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る。

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)
```

[入力パラメータ]

H	comNo	：COM 番号 COM0：ポート 0
B	*sfName	：送信元ファイル名エリア （複数指定及びワイルドカード不可）
B	*rfName	：追加先ファイル名エリア （複数指定及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	：処理ステータス E_OK：正常　E_NG：異常　E_PRM：パラメータエラー

備　考

[関 連 関 数]

cu_open,cu_close

機　能	ファイルの受信	関数名	cu_fileRecv

指定された複数ファイルを一括して受信する。
受信先ディレクトリが存在しない場合は自動的に生成する。
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る。

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)
```

[入力パラメータ]

```
H   comNo  ：COM 番号  COM0：ポート 0
H   mode   ：転送モード（通常転送か再帰呼出し転送かを指定する。）
             CU_TRANS_NORMAL：通常転送  CU_TRANS_RECURSIVE：再帰呼出し
B         *fName：受信ファイル名エリア（複数指定及びワイルドカード可）
B   *dir   ：受信先ディレクトリ名エリア（複数指定及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。）
             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（cu_fileSend 関数参照）
```

[リターンパラメータ]

```
ER        ercd        ：処理ステータス
                      E_OK：正常  E_NG：異常  E_PRM：パラメータエラー
```

備　考

[関 連 関 数]

cu_open,cu_close

機　能	回線クローズ	関数名	cu_close

セッションの開放及び回線ポートのクローズを行う。
終了指示コマンドを相手に送信することにより、セッションを開放する。
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る。
但し、既にエラーが発生した場合している場合は送信されない。

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)
```

[入力パラメータ]

H	comNo	：COM 番号　COM0：ポート 0
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 ﾄﾞﾗｲﾌﾞﾌｫｰﾏｯﾄ指示
		CU_CLOSE_FORMAT_B：B ﾄﾞﾗｲﾌﾞﾌｫｰﾏｯﾄ指示
		CU_CLOSE_PWROFF　：電源 OFF 指示

[リターンパラメータ]

ER	ercd	：処理ステータス
		E_OK：正常　E_NG：異常　E_PRM：パラメータエラー

備　考

[関 連 関 数]

cu_open

機　能	エラー情報の取得	関数名	cu_readErrStat

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する。
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する。
取得後、エラー情報はクリアされる。

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)
```

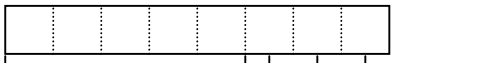
[入力パラメータ]

```
H          comNo          ：COM 番号
                             COM0：ポート０
```

[出力パラメータ]

```
CU_ERRINFO    *errInfo          ：エラー情報

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      ：BIOS ｴﾗｰエリア　（IrDA 部 BIOS エラーが設定される。）
}CU_ERRINFO;
```



予約

送信権局種別（送信権局/被送信権局）
　0：送信権局
　1：被送信権局
動作モード種別（ＨＴモード/ＰＣモード）
　0：ＨＴモード
　1：ＰＣモード
エラー検出局
　0：自局
　1：相手局

[リターンパラメータ]

```
ER         ercd          ：処理ステータス
                        E_OK：正常　E_PRM：パラメータエラー
```

備　考

[関 連 関 数]

cu_open,cu_fileSend,cu_fileRecv,cu_fileAdd,cu_close,cu_idle,cu_cmdRecv,
cu_fileDelete,cu_fileMove,cu_makeDir,cu_dateTime,cu_getFileInfo,cu_setFileInfo,
cu_getDiskInfo,cu_msgSend,cu_beep,cu_getSysInfo

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
００ｈ	CU_CMD_NON	該当コマンドなし
０１ｈ	CU_CMD_FSEND_TINFO	ファイル転送情報コマンド
０２ｈ	CU_CMD_FSEND_FINFO	ファイル情報コマンド
０３ｈ	CU_CMD_FRECV_TREQ	ファイル受信要求コマンド
０４ｈ	CU_CMD_FADD	ファイル追加コマンド
０５ｈ	CU_CMD_FDATA	ファイルデータコマンド
０６ｈ	CU_CMD_FDEL	ファイル削除コマンド
０７ｈ	CU_CMD_FMOV	ファイル移動コマンド
０８ｈ	CU_CMD_MAKEDIR	ディレクトリ作成コマンド
０９ｈ	CU_CMD_TIME_SET	日付時刻設定コマンド
０Ａｈ	CU_CMD_TIME_GET	日付時刻取得コマンド
０Ｂｈ	CU_CMD_DISP	メッセージ表示コマンド
０Ｃｈ	CU_CMD_BEEP	ブザー鳴動コマンド
０Ｄｈ	CU_CMD_FINFO_GET	ファイル情報取得コマンド
０Ｅｈ	CU_CMD_FINFO_SET	ファイル情報設定コマンド
０Ｆｈ	CU_CMD_DINFO_GET	ディスク情報取得コマンド
１０ｈ	CU_CMD_SYS_GET	システム情報取得コマンド
１１ｈ	CU_CMD_IDLE	ＩＤＬＥ通知コマンド
１２ｈ	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（カテゴリコード００ｈ、ＤＣｈ～ＦＦｈ）		
００ｈ	００ｈ	正常終了
ＤＣｈ～Ｆ５ｈ	００ｈ	ﾌｫｰﾏｯﾄ指示コマンド（Ａ～Ｚ）
Ｆ６ｈ	００ｈ	電源 OFF 終了通知
Ｆ７ｈ	００ｈ	リセット指定終了通知
Ｆ８ｈ	００ｈ	中断キーによる終了通知
Ｆ９ｈ～ＦＦｈ	－	リザーブ
プロトコルエラー（カテゴリコード０１ｈ）		
０１ｈ	００ｈ	受信ﾌﾚｰﾑﾌｧﾝｸｼｮﾝｺｰﾄﾞ未定義エラー
	０１ｈ	受信ﾌﾚｰﾑｻﾌﾞﾌｧﾝｸｼｮﾝｺｰﾄﾞ未定義エラー
	０３ｈ	受信ﾌﾚｰﾑﾁｪｯｸｻﾑｴﾗｰ
	０４ｈ	ｼｰｹﾝｽｴﾗｰ
	０５ｈ	ｼｰｹﾝｽ番号エラー
	０７ｈ	受信ﾌﾚｰﾑ内情報パラメータエラー
	０８ｈ	受信タイムアウト
	１０ｈ	ｺﾏﾝﾄﾞﾚﾝｸﾞｽｴﾗｰ
ファイルエラー［プロトコル論理］（カテゴリコード０４ｈ）		
０４ｈ	００ｈ	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（カテゴリコード１０ｈ）		
１０ｈ	００ｈ	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）
	０１ｈ	使用関数フェーズエラー　（関数の使い方に誤りがある。動作モード／送信権局モードを確認）
	０２ｈ	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）
	０３ｈ	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）
	０４ｈ	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）
	０５ｈ	システム日付設定エラー　（指定日付を確認）
	０６ｈ	システム時刻設定エラー　（指定時刻を確認）
	０７ｈ	タイマ使用エラー　（タイマが登録できなかったＡＰで使用しているタイマ数を確認）
	０８ｈ	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）
	０９ｈ	致命的エラー　（IrDA BIOS からのエラー。ＬＢの発生等が考えられる。詳細は各 BIOS 仕様を参照）
	０Ａｈ	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）
	０Ｂｈ	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ＢＩＯＳ］（カテゴリコード１１ｈ）		
１１ｈ	００ｈ	クリエートエラー
	０１ｈ	オープンエラー
	０２ｈ	リードエラー
	０３ｈ	ライトエラー
	０４ｈ	シークエラー
	０５ｈ	ファイル削除エラー
	０６ｈ	ディレクトリ削除エラー
	０７ｈ	ファイル名変更移動エラー
	０８ｈ	タイムスタンプ設定エラー
	０９ｈ	タイムスタンプ取得エラー
	０Ａｈ	ファイル属性設定エラー
	０Ｂｈ	ファイル属性取得エラー
	０Ｃｈ	ディレクトリ作成エラー
	０Ｄｈ	ﾌｧｲﾙｻｲｽﾞ変更エラー
システムメニュー通信エラー（カテゴリコード２０ｈ）		
２０ｈ	００ｈ	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）
	０１ｈ	環境設定ファイル未存在エラー（CONFIG.HTS ファイルがない）
	０２ｈ	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の確認)
	０３ｈ	相手局不正(想定している相手局ではない。相手局を確認)
	０４ｈ	指定ドライブなし(子機作成時、送信側指定ドライブが受信側に存在しない)
システム異常エラー（カテゴリコード０Ｆｈ）		
０Ｆｈ	０ｘｈ	ＦＴＰ部内部エラー
	１ｘｈ	通信ユーティリティ内部エラー

機　能	IDLE遷移	関数名	cu_idle

IDLE 通知送信後、相手局からのコマンド受信待ち状態となる。HTモード時のみ使用可能。
以後、相手局から受信したコマンドは順次実行していく。
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ない。
ファイル送信、追加及び受信の際、進捗グラフを表示することができる。

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)
```

[入力パラメータ]

```
H    comNo          ：COM 番号          COM0：ポート 0
B    *script              ：ｽｸﾘﾌﾟ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]
                    (複数指定及びワイルドカード不可。未設定時は NULL を設定)
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する。)
```

[リターンパラメータ]

```
ER          ercd          ：処理ステータス
                          E_OK：正常　E_NG：異常　E_PRM：パラメータエラー
```

備　考

[関 連 関 数]

cu_open,cu_close

機　能	コマンド受信待ち	関数名	cu_cmdRecv

HT からのコマンド受信待ち状態となる。ＰＣモード時のみ使用可能。
以後、HT から受信したコマンドは順次実行していく。
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
終了しない。
ファイル送信、追加及び受信の際、進捗グラフを表示することができる。

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)
```

[入力パラメータ]

```
H           comNo          ：COM 番号          COM0：ポート０
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する。）
```

[出力パラメータ]

```
H           *endKind       ：終了種別フラグ設定エリア（正常終了時のみ有効）
                              CU_RECV_END：終了指示受信
                              CU_RECV_IDLE：ＩＤＬＥ通知受信
B           *script              ：ｽｸﾘﾌﾟﾄﾌｧｲﾙ名エリア
                          （ＩＤＬＥ通知コマンド受信時に設定される。）
                                 [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           ：処理ステータス
                          E_OK：正常　E_NG：異常　E_PRM：パラメータエラー
```

備　考

[関 連 関 数]

cu_open,cu_close

機能	中断キーの設定	関数名	cu_stopKeySet

通信を中断するキーを登録／復旧（戻す）を行います。
設定できるキーはＦ１～Ｆ５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_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(BL)
                  CU_FNC_3      ：Ｆ３(戻る)
                  CU_FNC_4      ：Ｆ４(▼)
                  CU_FNC_5      ：Ｆ５(F)
                  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	ファイル削除	関数名	cu_fileDelete

相手局側のファイル／ディレクトリを削除する。複数ファイル／ディレクトリの削除が可能。
指定ファイルが存在しない場合は正常終了する。

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)
```

[入力パラメータ]

H	comNo	：COM 番号　　COM0：ポート 0
B	*fName	：削除するファイル／ディレクトリ名エリア （複数指定及びワイルドカード可）

[リターンパラメータ]

ER	ercd	：処理ステータス E_OK：正常　E_NG：異常　E_PRM：パラメータエラー

備　考

[関 連 関 数]

cu_open,cu_close

機　能	ファイル移動	関数名	cu_fileMove

相手局側のファイルを同一ディスク内で移動する。
移動先ディレクトリが存在しない場合は自動生成する。
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、ファイル名の変更となる。
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る。

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)
```

[入力パラメータ]

H	comNo	：COM 番号　　COM0：ポート 0
B	*sfName	：移動元ファイル名エリア（複数指定及びワイルドカード不可）
B	*dfName	：移動先ファイル名エリア（複数指定及びワイルドカード不可）

[リターンパラメータ]

ER	ercd	：処理ステータス E_OK：正常　E_NG：異常　E_PRM：パラメータエラー

備　考

[関 連 関 数]

cu_open,cu_close

機　能	ディレクトリ作成	関数名	cu_makeDir

相手局側のディスクにディレクトリを作成する。

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)
```

[入力パラメータ]

```
H           comNo          ：COM 番号        COM0：ポート 0
B           *mDir          ：作成ディレクトリ名エリア
                            (複数指定及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：ｱｰｶｲﾌﾞ
                                   （_A_SUBDIR は自動的に OR される）
```

```
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 を設定すること。

[リターンパラメータ]

```
ER          ercd           ：処理ステータス
                           E_OK：正常　E_NG：異常　E_PRM：パラメータエラー
```

備　考

[関 連 関 数]

cu_open,cu_close

機　能	ファイル情報の取得	関数名	cu_getFileInfo

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）の取得を行う。
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る。
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」を指定する。
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る。
他の通信関数を使用すると、次情報取得は行えない。

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)
```

[入力パラメータ]

```
H    comNo  ：COM 番号        COM0：ポート0
H    mode   ：最初／次フラグ
            CU_GET_FIRST：最初の取得
             （１ファイル指定又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」では参照しない。）
```

[出力パラメータ]

```
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る）
                  ＊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 が設定される。
```

```
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：ｱｰｶｲﾌﾞ
}CU_FINFO;
```

[リターンパラメータ]

```
ER        ercd          ：処理ステータス
                        E_OK：正常   E_NG：異常   E_PRM：パラメータエラー
```

備　考

[関 連 関 数]

cu_open,cu_close

機　能	ファイル情報の更新	関数名	cu_setFileInfo

相手局側の指定ファイル情報（タイムスタンプ・属性・サイズ）の更新を行う。
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する。

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)
```

[入力パラメータ]

```
H          comNo  ：COM 番号        COM0：ポート０
CU_FINFO   *fInfo ：ファイル情報設定エリア
```

```
typedef struct{
    B          name[256]   ：設定するファイル名（ﾌﾙﾊﾟｽ名）
                           （複数指定不可・ワイルドカード指定不可）
    CU_DATETIME    datetime;   ：日付時刻エリア（cu_dateTime 関数参照）
                           （変更しない場合は cu_dateTime 関数と同様）
    UW         size;       ：サイズ　（0 指定時は変更しない）
    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：ｱｰｶｲﾌﾞ
    }CU_FINFO;
```

[リターンパラメータ]

```
ER         ercd          ：処理ステータス
                         E_OK：正常　E_NG：異常　E_PRM：パラメータエラー
```

備　考

[関 連 関 数]

cu_open,cu_close

機　能	ディスク情報の取得	関数名	cu_getDiskInfo

相手局側の指定ドライブ情報の取得を行う。
指定ドライブの情報がドライブ情報エリアへ設定される。

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)
```

[入力パラメータ]

H	comNo	：COM 番号　　COM0：ポート 0
B	drive	：ドライブ名エリア 'A'～'Z'の何れか。

[出力パラメータ]

CU_DINFO　*dInfo：ドライブ情報エリアアドレス

```
typedef struct{
      UW    size;        /*ディスク容量*/
      UW    freex;       /*ディスク空き容量*/
      UB    status;            /*ディスク状態*/
                  CU_DINFO_NORMAL：ディスク有り（ﾌｫｰﾏｯﾄ済）
                  CU_DINFO_NOFMT ：ディスク有り（未ﾌｫｰﾏｯﾄ）
                  CU_DINFO_NODISK：ディスク無し
}CU_DINFO;
```

[リターンパラメータ]

ER	ercd	：処理ステータス E_OK：正常　E_NG：異常　E_PRM：パラメータエラー

備　考

[関 連 関 数]

cu_open,cu_close

機　能	日付時刻の取得／設定	関数名	cu_dateTime

相手局側の日付時刻の取得及び設定を行う。
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る。
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する。

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)
```

[入力パラメータ]

```
H           comNo          ：COM 番号    COM0：ポート 0
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
```

[出力パラメータ]

```
CU_DATETIME *dateTime     ：取得日付時刻エリアアドレス
```

```
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 を設定すること。
*時刻のみの設定の場合は day,month,year,それぞれ FFH,FFH,FFFFH を設定すること。

[リターンパラメータ]

```
ER          ercd           ：処理ステータス
                           E_OK：正常   E_NG：異常   E_PRM：パラメータエラー
```

備　考

[関 連 関 数]

cu_open,cu_close

機　能	システム情報の取得	関数名	cu_getSysInfo

相手局側のシステム情報を取得する。
相手局がＰＣの場合は接続セッション番号も返す。(相手局がＨＴの場合は 0 固定)
なお、これらの情報はオープンのセッション時に既に取得しているため、通信は行わず、
情報のみを返す。

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)
```

[入力パラメータ]

H	comNo	：COM 番号	COM0：ポート０

[出力パラメータ]

CU_SYSINFO *sysInfo　：取得システム情報エリア

```
typedef     struct{
      UH    id;    ：セッション ID（PC との接続以外は 0 固定）
      UB    ftpver;      ：FTP バージョン
      UB    code[3];：機種コード
                        ”710”：HT　その他：PC 又は他機種
      UB    model;：ﾓﾃﾞﾙ情報（値は 04H 固定）
}CU_SYSINFO;
```

[リターンパラメータ]

ER	ercd	：処理ステータス
		E_OK：正常　E_NG：異常　E_PRM：パラメータエラー

備　考

[関 連 関 数]

cu_open,cu_close

機　能	画面表示メッセージの送信	関数名	cu_msgSend

相手局側に表示するメッセージを送信する。

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)
```

[入力パラメータ]

H	comNo	：COM 番号 COM0：ポート０　COM1：ポート１
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	：処理ステータス E_OK：正常　E_NG：異常　E_PRM：パラメータエラー

備　考

[関 連 関 数]

cu_open,cu_close

機　能	ブザー鳴動	関数名	cu_beep

相手局側のブザーを鳴らす。

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)
```

[入力パラメータ]

H	comNo	：COM 番号	COM0：ポート 0

[リターンパラメータ]

ER	ercd	：処理ステータス E_OK：正常　E_NG：異常　E_PRM：パラメータエラー

備　考

[関 連 関 数]

cu_open,cu_close

機　能	FCHK ﾘｽﾄﾌｧｲﾙの生成	関数名	cu_fchklog_Create

指定複数ファイルの FCHK ﾘｽﾄﾌｧｲﾙ(FCHK.LOG)を生成する。
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る。
(1)ファイルのパス名(転送先ディレクトリ名を含む),(2)作成日付,(3)作成時間,
(4)ﾌｧｲﾙｻｲｽﾞ,(5)指定された全ファイルのﾁｪｯｸｻﾑﾃﾞｰﾀ,
(6)FCHK ﾘｽﾄﾌｧｲﾙ自身のﾁｪｯｸｻﾑﾃﾞｰﾀ
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る。

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)
```

[入力パラメータ]

```
H   mode   ：ファイル指定モード(再帰呼出しを行うかどうかを指定する。)
           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(複数指定及びワイルドカード指定可)
B   *dir   ：転送先ディレクトリ名（複数指定及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（複数指定及びワイルドカード指定不可）
H   append ：ｱﾍﾟﾝﾄﾞｵﾌﾟｼｮﾝ（既存の FCHK ﾘｽﾄﾌｧｲﾙに対する追加を指定）
             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
```

```
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%として表示する
               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)
  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;
```

[リターンパラメータ]

```
ER        ercd   ：処理ステータス
                   E_OK：正常
                        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 が見つからない(追加)
                        FCHK_NG0D：パラメータエラー
```

備　考

[関 連 関 数]

cu_fchklog_Check

機　能	FCHK ﾘｽﾄﾌｧｲﾙのチェック	関数名	cu_fchklog_Check

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する。
比較照合するファイル情報は、以下の情報である。
(1) 作成日付,(2)作成時間,(3)ﾌｧｲﾙｻｲｽﾞ,(4)全ファイルのﾁｪｯｸｻﾑﾃﾞｰﾀ,
(5)FCHK ﾘｽﾄﾌｧｲﾙ自身のﾁｪｯｸｻﾑﾃﾞｰﾀ
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
表示できる。

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)
```

[入力パラメータ]

```
B           *listDir      ：FCHK ﾘｽﾄﾌｧｲﾙが存在するディレクトリ名
                          （複数指定及びワイルドカード指定不可）
CU_GRAPHSET       *graphSet     ：グラフ表示情報
```

```
typedef struct{
  H  graphMode   ：グラフ表示モード
               CU_GRAPH_ON_1：ﾘｽﾄﾌｧｲﾙ照合全体を 100%として表示する
               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)
  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;
```

[リターンパラメータ]

```
ER          ercd   ：処理ステータス
                 E_OK：正常
                       FCHK_NG03：FCHK.LOG が見つからない
                       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(ﾌｧｲﾙｻｲｽﾞの不一致)
                       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ﾃﾞｰﾀの不一致)
                       FCHK_NG08：ﾘｽﾄﾌｧｲﾙの内容不一致(ﾘｽﾄﾁｪｯｸｻﾑﾃﾞｰﾀの不一致)
                       FCHK_NG0B：ﾘｽﾄﾌｧｲﾙ読み込み時エラー
                       FCHK_NG0D：パラメータエラー
```

備　考	

[関 連 関 数]

cu_fchklog_Create

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８．１ 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。

表８．２ 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(内２つはシステムで使用)
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

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を無効にして復帰を行い、その要求のブザー音を鳴動します。

表８．３ 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	2600Hz	3000Hz	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        ：タイマ登録 ID またはリターンコード
```

【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）
                登録時に戻り値として得られた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(内２つはシステムで使用)です。それ以上の登録は
登録数オーバーとなります。

≪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　　：タイマ登録 ID またはリターンコード

【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 = 3000Hz
・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

本機には満タン、レベル１、レベル２、ＬＢ１の４段階の電池状態があります。通知モードを設定することでＬＢ1状態をアプリケーションプログラムに通知することも可能です。

10.1.2 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。

10.1.3 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。

10.1.4 低消費電力制御

キー待ち状態（key_read/key_string/key_num をアプリケーションで呼ぶ）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。

10.1.5 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。

10.1.6 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。

表９．１　電源通知モードの動作

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	ＩＯボックス接続	何もしない	イベント設定	発生時	

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

10.1.7 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。

10.1.8 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

機能	ＩＯＢＯＸ起動設定／解除	関数名	pwr_IoboxBootMode

ＩＯＢＯＸ装着で電源ＯＮをするかしないかの設定をおこないます。

≪Ｃ言語インタフェース≫

【コーリングシーケンス】

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

【パラメータ】

UH　OnOff ： 通知モード
　　PWR_ON　　：ＩＯＢＯＸ起動設定
　　PWR_OFF　　：ＩＯＢＯＸ起動解除

【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_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、アプリケーションに通知します。
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で使用して下さい。

表１０．１　通知モードビットパターン

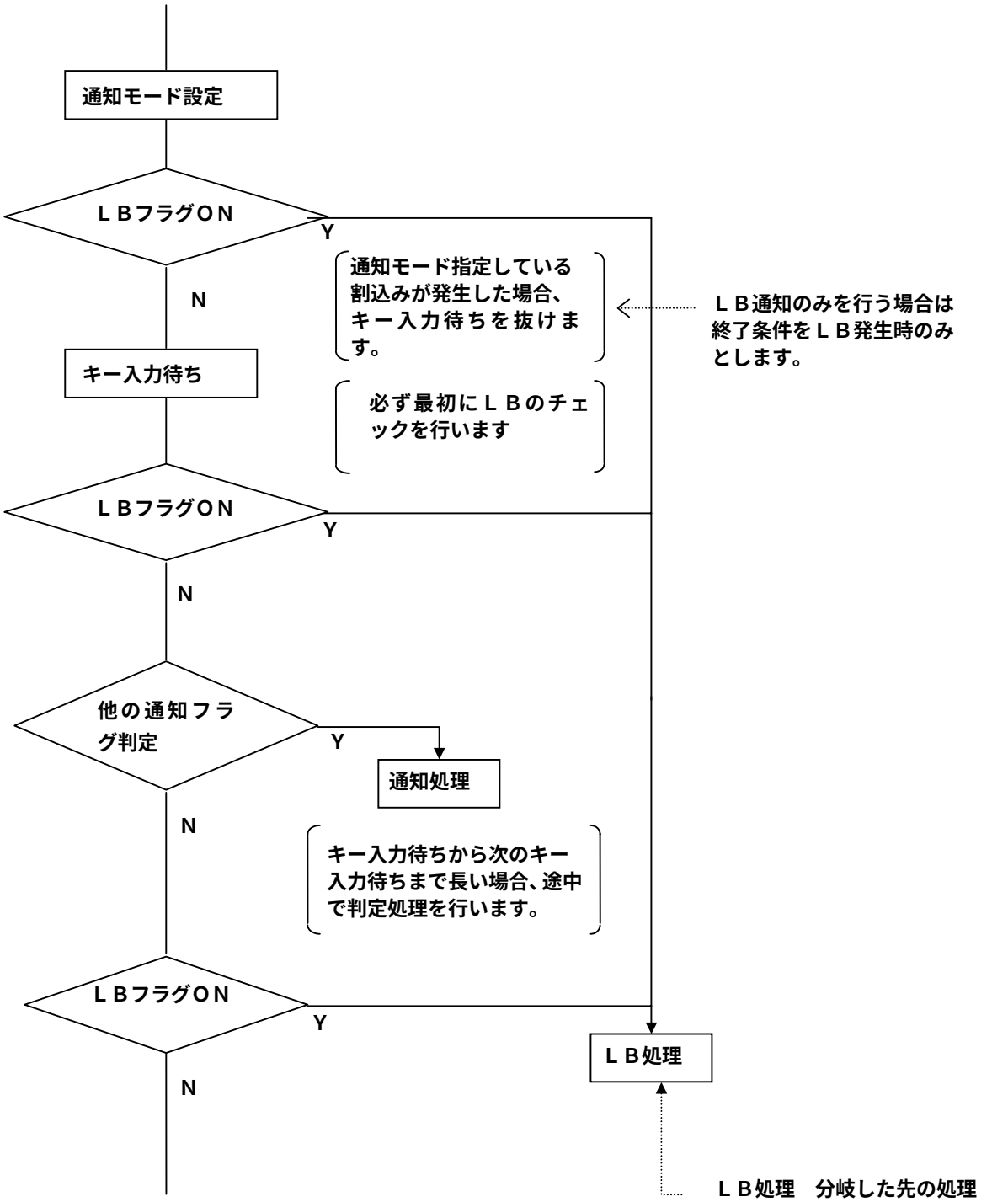
名　称	説　明
FL_LB_INT_LB0	ＬＢ０設定用ビットパターン
FL_LB_INT_LB1	ＬＢ１設定用ビットパター
FL_LB_INT_LB4	ＬＢ４設定用ビットパターン
FL_LB_INT_LB5	ＬＢ５設定用ビットパターン

表１０．２　通知モード以外のビットパターン

名　称	説　明
FL_SET_INT_IO	ＩＯボックス検出用ビットパターン

以下にＬＢ通知モードを使用する場合のチャートを記します。

ＬＢ通知モード



以下にＬＢ０、１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,FL_LB_INT_LB0|FL_LB_INT_LB1);   通知モード設定
             •
             •
             •
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 );           電源通知イベント
      sub_lb0();                                                のクリア後各 LB に対応した
   }else if(ptn & FL_LB_INT_LB1)                                処理への分岐
   {
      pwr_inhabit_clr( FL_LB_INT_ID, FL_LB_INT_LB1 );
      sub_lb1();
   }
   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);
             •
             •
             •
void    sub_lb0( void )
{                                                               各 LB に対応する処理
                  •
                  •
    return;
}

void    sub_lb1( void )
{
                  •
                  •
    return;
}
```

11.3.2　　キーに対する通知モードの場合

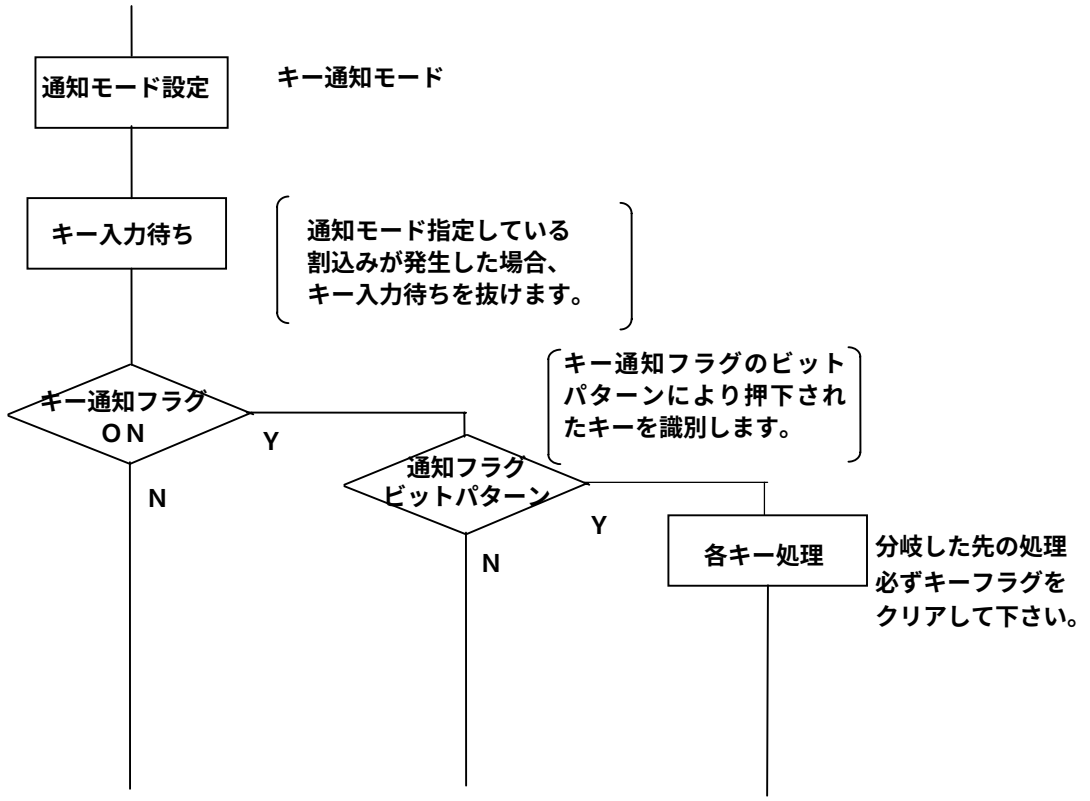
キーに対する通知モードを使用する場合、下記の点に注意して下さい。

- 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１０．３ 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

以下にキー通知を使用する場合のチャートを記します。



以下にファンクションキー１、２に対する通知モードの使用例を記します。

```
ER          err, retcd;
UW          ptn, i;
KEY_INP     keyinf;
ID          dummy, fid;
            …
fid = FL_FK_INT_ID;
ptn = FL_FK_INT_FNC1;
err = key_fnc_mode( FNC_MODE_SET, FNC_1, &fid, &ptn );
fid = FL_FK_INT_ID;
ptn = FL_FK_INT_FNC2;
err = key_fnc_mode( FNC_MODE_SET, FNC_2, &fid, &ptn );
            …
keyinf.ext = KEY_INT_EXT;
keyinf.echo = ECHO_OFF;
keyinf.font_size = LCD_ANK_STANDARD;
keyinf.type = LCD_ATTR_NORMAL;
keyinf.column_pos = 0;
keyinf.line_pos = 0;
retcd = key_read( &keyinf );
if(retcd == E_KEY_INT)
{
   err = flg_sts( &dummy, &ptn, FL_FK_INT_ID );
   if(ptn & FL_FK_INT_FNC1)
   {
      clr_flg( FL_FK_INT_ID, ～FL_FK_INT_FNC1 );
      sub_fnc1();
   }else if(ptn & FL_FK_INT_FNC2)
   {
      clr_flg( FL_FK_INT_ID, ～FL_FK_INT_FNC2 );
      sub_fnc2();
   }
}
fid=FL_FK_INT_ID;
ptn= FL_FK_INT_FNC1;
err= key_fnc_mode( FNC_MODE_CLR, FNC_1, &fid, &ptn );
fid= FL_FK_INT_ID;
ptn= FL_FK_INT_FNC2;
err = key_fnc_mode( FNC_MODE_CLR, FNC_2, &fid, &ptn );
             …
void    sub_fnc1( void )
{
      …
    return;
}
void    sub_fnc2( void )
{
      …
   return;
}
```

通知モード設定

リターン条件設定

（KEY による脱出）

通知フラグ状態をクリア後
各 KEY に対応した処理へ
の分岐

通知モード解除

各 KEY に対応する処理

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

【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

【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	電源通知モード設定	関数名	pwr_inhabit

電源の通知モード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
                    FL_LB_INT_ID を設定してください
UW  BitPtrn      ：ビットパターン
                  FL_LB_INT_LB0 - FL_LB_INT_LB5      ：ＬＢ検出
                  FL_SET_INT_IO                      ：IO ボックス検出
```

【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 FL_LB_INT_ID を設定してください
UW	BitPtrn	：ビットパターン FL_LB_INT_LB0 - FL_LB_INT_LB5 ：ＬＢ検出 FL_SET_INT_IO ：IO ボックス検出

【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 ソフトウェアリセット処理

アプリケーションの指定で、以下の３つの動作をします。

① 再起動処理後、システムメニューを起動します。
② 再起動処理後、OS 受信待ちになります。
③ 再起動処理後、アプリケーションを起動します。

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。
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

【リターンコード】
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

【リターンコード】
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

【リターンコード】

なし

≪備考≫

機能	ソフトウェアリセット処理	関数名	sub_reset

アプリケーションから本関数を呼ぶと本機が再起動します。
リセット時指定するパラメータにより以下の動作をします。

・再起動後システムメニューが起動します。
・再起動後 OS 受信待ちになります。
・再起動後アプリケーションが起動します。

≪C 言語インタフェース≫

【コーリングシーケンス】

```
void = sub_reset ( VH reset_mode);
```

【パラメータ】

VH reset_mode　　：リセット種別
　　SUB_SOFT_RESET　　：再起動後システムメニューを起動
　　SUB_OS_REPLACE　　：再起動後 OS 受信待ち
　　SUB_RST_APEXEC　　：再起動後アプリケーションを起動

【リターンパラメータ】

なし

【リターンコード】

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-800	DT-900	本機
lcd_csr_set	カーソルタイプ設定	○	○	○	○
lcd_csr_put	カーソル位置設定	○	○	○	○
lcd_csr_get	カーソル位置読出し	○	○	○	○
lcd_char	１文字表示	○	○	○	○
lcd_string	文字列表示	○	○	○	○
lcd_line	直線描画	○	○	○	○
lcd_cls	画面クリア	○	○	○	○
lcd_led	LED の制御	○	○	○	○
lcd_gaiji	外字フォント登録	○	○	○	○
lcd_string2	文字列表示 2(スクロール抑制)	○	○	○	○
lcd_usrfont	ユーザーフォントファイル登録	○	○	○	○
lcd_romfont	ROM フォント設定	○	○	○	○
lcd_userstr	ユーザ文字列表示	○	○	○	○
lcd_active_set	ユーザ描画ページ設定		○		
lcd_visual_set	ユーザ表示ページ設定		○		
lcd_active_get	ユーザ描画ページ読出し		○		
lcd_visual_get	ユーザ表示ページ読出し		○		
lcd_cls_page	指定ページクリア		○		
lcd_grchar	グラフィック文字表示		○		
lcd_box	BOX 描画/削除		○		
lcd_el	EL バックライトの制御		○	○	○
lcd_shiftsymbol	S アイコンの点灯／消灯				○

（3）キー部

関数名	機能	DT-700	DT-800	DT-900	本機
key_read	１文字入力	○	○	○	○
key_string	文字列入力	○	○	○	○
key_num	数値入力	○	○	○	○
key_check	キーバッファのステータスチェック	○	○	○	○
key_clear	キーバッファのクリア	○	○	○	○
key_fnc	ファンクションコードの設定	○	○	○	○
key_fnc_mode	ファンクションキー通知モード設定	○	○	○	○
key_select	キー入力モード設定		○	○	○
key_pad_set	キーパッドファイル登録		○		
key_touch	ユーザタッチキー設定削除		○		
key_point	タッチ座標取得		○		
key_pad	キーパッド切替/状態取得		○		
key_pad_entry	キーパッド遷移設定/状態取得		○		
key_setInputMode	キー入力モード切替				○

（4）通信部

関数名	機能	DT-700	DT-800	DT-900	本機
c_open	COM のオープン	○	○	○	
c_close	COM のクローズ	○	○	○	
c_status	COM ステータスのリード	○	○	○	
c_hold	COM の占有	○	○	○	
c_chkopen	COM のオープンチェック	○	○	○	
c_wp	WakeUp 信号の設定			○	
c_dout	n 文字送信	○	○	○	
c_din	1 文字受信	○	○	○	
c_tmdin	タイムアウト監視受信	○	○	○	
c_mdout	メモリブロック送信	○			
c_mdin	メモリブロック受信	○			
c_out	1 文字送信	○	○	○	
c_break	ブレーク信号の制御	○	○	○	
c_txrx	送受信の有効/無効	○	○	○	
c_iobox	IO ボックス送信設定	○		○	
c_irout	IO ボックス送信	○		○	
c_timer	DR/CS/CD タイムアウト監視値設定	○	○	○	
c_rs	RS 信号の制御	○	○	○	
c_er	ER 信号の制御	○	○	○	
c_errs	ER/RS 信号の制御	○	○	○	
c_flush	受信バッファのクリア	○	○	○	
c_bfsts	受信バッファステータスのリード	○	○	○	
c_errbfring	リターンコードバッファリング制御の設定	○	○	○	
c_rderrsts	エラーステータスのリード	○	○	○	
c_chghdr	受信ハンドラ切替え	○	○	○	

前ページから続き

関数名	機能	DT-700	DT-800	DT-900	本機
c_cimode	CI 信号立ち上げモード設定			○	
c_brkevent	ブレイク要因の設定			○	
Ir_Open	IrCOMM オープン		○	○	○
Ir_Close	IrCOMM クローズ		○	○	○
Ir_Read	データ読込み		○	○	○
Ir_Write	データ書込み		○	○	○
Ir_QueryTx	送信データ数問合せ		○	○	○
Ir_QueryRx	受信データ数問合せ		○	○	○
Ir_Err_Get	エラー値取得		○	○	○
Ir_State_Set	通信状態設定		○	○	○
Ir_SetPortConfig	自局能力設定		○	○	○
Ir_Init	IrCOMM 強制終了		○	○	○
Ir_SetParame	パラメータ設定		○		

（5）電源部

関数名	機能	DT-700	DT-800	DT-900	本機
pwr_inhabit	通知モード設定	○	○	○	○
pwr_inhabit_clr	電源通知イベントのクリア	○	○	○	○
pwr_hold_apo	APO 禁止設定	○	○	○	○
pwr_off	電源オフ	○	○	○	○
pwr_ioboxBootMode	IOBOX 起動設定／解除				○

（6）通知部

関数名	機能	DT-700	DT-800	DT-900	本機
flg_sts	通知フラグ状態取得	○	○	○	○
clr_flg	通知フラグ状態クリア	○	○	○	○
wai_flg	フラグセット待ち	○	○	○	○

（7）バーコード部

関数名	機能	DT-700	DT-800	DT-900	本機
OBR_open	OBR オープン	○	○	○	○
OBR_close	OBR クローズ	○	○	○	○
OBR_getc	OBR データ１文字リード	○	○	○	○
OBR_gets	OBR データ文字列リード	○	○	○	○
OBR_stat	OBR バッファステータスチェック	○	○	○	○
OBR_flush	OBR バッファのクリア	○	○	○	○
OBR_moderd	OBR 動作モードの取得	○	○	○	○
OBR_modewt	OBR 動作モード設定	○	○	○	○
OBR_chgbuf	OBR バッファの切替え	○	○	○	○
OBR_gain	発光ゲイン切替え				
OBR_trigmode	トリガーキーによる電源オン設定		○	○	○
OBR_swing	レーザー発光幅の設定／参照			○	
OBR_widenarrow	レーザー発光幅の微調整			○	

（8）バーコード動作モード

機能	DT-700	DT-800	DT-900	本機
読取り可能コード	○	○	○	○
読取り桁数	○	○	○	○
出力フォーマット	○	○	○	○
チェックデジットの実行	○	○	○	○
チェックキャラクタの出力	○	○	○	○
読取り方式(単発)	○	○	○	○
読取り方式(連続:TRG 有)	○	○	○	○
ゲインコントロール	○			
ブザー制御	○	○	○	○
LED 制御	○	○	○	○
出力バッファ	○	○	○	○
終了コード	○	○	○	○
読取り動作	○	○	○	○
２つコード認識		○		
レーザー発光幅制御			○	

（９）タイマ部

関数名	機能	DT-700	DT-800	DT-900	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	エラービープ音	○	○	○	○
s_beep2	エラービープ音２（赤ＬＥＤ点灯）		○		
s_sound	サウンド音 1	○	○	○	○
s_dateget	日付の取得	○	○	○	○
s_dateset	日付の設定	○	○	○	○
s_timeget	時間の取得	○	○	○	○
s_timeset	時間の設定	○	○	○	○

（１０）データ管理部

関数名	機能	DT-700	DT-800	DT-900	本機
dat_mem_size	メモリ領域の空きサイズの取得	○	○	○	○
dat_system	システムデータの設定	○	○	○	○
dat_OSVer_Read	OS バージョン読出し		○	○	○
dat_dealer_chk	代理店 ID のチェック		○	○	○
dat_Apload（注 2）	AP ロード&実行		ＲＡＭ○	○	○

注 2　ＡまたはＢドライブに存在するアプリケーションから他のアプリケーションを実行します。
複数のアプリケーションを同時に動作させる関数ではありません。
「ＲＡＭライブラリ」として提供します。

（１１）システムデータ

項目	管理データ	DT-700	DT-800	DT-900	本機
電源関連	APO 時間	○	○	○	○
	ABO 時間	○	○	○	○
	レジューム ON/OFF	○	○	○	○
	自動コントラスト調整 ON/OFF		○		
KEY 関連	クリック音 ON/OFF	○	○	○	○
表示関連	フォント MODE	○	○	○	○
	フォント種別(通常/強調)		○	○	○
	日本語/英語	○	○	○	○
	コントラスト値	○	○	○	○
	コントラスト差分		○	○	○
	LB 表示 MODE		○		
通信関連	速度(IR)	○	○	○	
	データ(IR)	○	予約領域	○	
	パリティ(IR)	○	予約領域	○	
	STOP(IR)	○	予約領域	○	
	速度(RF/シリアル)	○	予約領域	○	
	データ(RF/シリアル)	○	予約領域	○	
	パリティ(RF/シリアル)	○	予約領域	○	
	STOP(RF/シリアル)	○	予約領域	○	
	速度(10P)	○	○		
	データ(10P)	○	予約領域		
	パリティ(10P)	○	予約領域		
	STOP(10P)	○	予約領域		
	デフォルト通信 PORT		○		
	速度(PHS)			○	
	データ(PHS)			○	
	パリティ(PHS)			○	
	STOP(PHS)			○	
OBR 関連	読取り回数	○	○	○	○
	照合回数	○	○	○	○
	スキャン時間	○	○	○	○
	読取り禁止時間	○	予約領域		
タイマ関連	音量	○	○	○	○
システム関連	機器 ID	○	○	○	○
	代理店 NO		○	○	○
	BIOS バージョン	○	○	○	○
	PATCH バージョン		○	○	○
	機器種別	○	○	○	○
ネットワーク関連	IP アドレス(SS 無線)		○		
	マスク値(SS 無線)		○		

（前頁つづき）

項目	管理データ	DT-700	DT-800	Dt-900	本機
プロトコル関連	通常受信タイムアウト（マルチドロップ）	○		○	
	通常リトライ回数（マルチドロップ）	○		○	
	マルチデータリンク受信タイムアウト（マルチドロップ）	○		○	
	対向送信データリンク受信タイムアウト（マルチドロップ）	○			
	対向受信データリンク受信タイムアウト（マルチドロップ）	○			
	対向受信データリンクリトライ回数（マルチドロップ）	○			
	データリンク受信タイムアウト(FLINK)				○
	受信データなしタイムアウト(FLINK)				○
	再データリンク可能回数(FLINK)				○
	セッション確立タイムアウト		○	○	
	受信タイムアウト		○	○	
	タイムアウト時間（HIO）				○
	リトライ時間（HIO）				○
	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(BHT)			○	
	水平パリティ(BHT)			○	
	リンクタイムアウト(BHT)			○	
メモリ関連	アプリケーション SIZE		○	○	○
ファイルモード	FORMAT			○	

（１２）ファイル部

関数名	機能	DT-700	DT-800	DT-900	本機
dat_fsize	ファイル空き領域サイズの取得	○		○	
dat_fdir	ファイル格納情報の取得	○		○	
dat_fdel	ファイルの削除	○		○	
dat_F_Search	ファイルデータの検索	注１		○	
dat_fsize_chg	ファイルサイズ変更				
open	ファイルオープン	○	○	○	○
close	ファイルクローズ	○	○	○	○
read	ファイルのリード	○	○	○	○
write	ファイルのライト	○	○	○	○
lseek	ファイルリード/ライト位置の設定	○	○	○	○
sbrk	メモリ領域の割当て	○	○	○	○
fil_mkdir	ディレクトリの作成		○	○	○
fil_rmdir	ディレクトリの削除		○	○	○
fil_closeX	ファイルのクローズ（日付指定）				○
fil_remove	ファイルの削除		○	○	○
fil_rename	ファイル名の変更/移動		○	○	○
fil_renameX	ファイル名の変更/移動（日付指定）				○
fil_fstat	ファイルの日時･サイズ･属性の取得		○	○	○
fil_chsize	ファイルのサイズの変更		○	○	○
fil_getsize	ファイル領域空きサイズの取得		○	○	○
fil_findfirst	ファイル名の取得		○	○	○
fil_findnext	ファイル名の取得(次候補)		○	○	○
fil_filesize	ファイルの個数と総サイズの取得			○	○
fil_filefind	ファイル全パス名の取得			○	○

注１　dat_sub.obj とリンクすることで対応

（１３）共通関数

関数名	機能	DT-700	DT-800	DT-900	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	キャリブレーション起動		○	○	
wkup_ss	SS 無線ユーティリティ起動		○		
sub_reset	ソフトウェアリセット処理				○

（１４）通信ユーティリティ

関数名	機能	DT-700	DT-800	DT-900	本機
cu_open	通信ポート初期化	○	○	○	
cu_stopKeySet	中断キーの登録/削除	○	○	○	
cu_fileSend	ファイル送信	○	○	○	
cu_fileSendSet	ファイル送信情報設定	○		○	
cu_fileSend1	1 ファイル送信	○		○	
cu_fileRecv	ファイル受信	○	○	○	
cu_msgSend	画面表示メッセージ送信	○	○	○	
cu_end	通信中断	○		○	
cu_close	回線クローズ	○	○	○	
cu_readErrStat	エラー詳細情報取得	○	○	○	
cu_readDIRjInfo	データリンク拒否情報取得	○		○	
cu_fileAdd	ファイルの追加		○	○	
cu_idle	IDLE 遷移		○	○	
cu_cmdRecv	コマンド受信待ち		○	○	
cu_fileDelete	ファイル削除		○	○	
cu_fileMove	ファイル移動		○	○	
cu_makeDir	ディレクトリ作成		○	○	
cu_dateTime	日付時刻の取得および設定		○	○	
cu_getFileInfo	ファイル情報の取得		○	○	
cu_setFileInfo	ファイル情報の更新		○	○	
cu_getDiskInfo	ディスク情報の取得		○	○	
cu_beep	ブザー鳴動		○	○	
cu_getSysInfo	システム情報の取得		○	○	
cu_apRecvSet	AP インストール設定		○		
cu_fchklog_Create	FCHK リストファイルの生成		○	○	
cu_fchklog_Check	FCHK リストファイルのチェック		○	○	
cu_setDrive	転送ドライブ指定			○	
cu_msgSend	画面表示メッセージ送信			○	
cu_SetCode	DT500 プロトコル制御コード拡張設定			○	
HIO_PortOpen	通信ポートのオープン				○
HIO_PortClose	通信ポートのクローズ				○
HIO_ReceiveFile	IOBOX からファイル受信				○
HIO_SendFile	IOBOX へファイル送信				○
HIO_DeleteFile	IOBOX のファイル削除				○
HIO_MemoryFormat	IOBOX のメモリ初期化				○
HIO_GetSysInfo	IOBOX のシステム情報の取得				○
HIO_GetMemoryInfo	IOBOX のメモリ情報の取得				○
HIO_GetFileInfo	IOBOX の全ファイル情報の取得				○

前ページからの続き

項目	設定内容	DT-700	DT-800	DT-900	本機
HIO_GetLastErrState	最後に発生したエラーの取得				○
HIO_GetLogData	IOBOX のログデータの取得				○
HIO_ClearLogData	IOBOX のログデータの削除				○
HIO_SetMemoryThresh	IOBOX のキャッシュメモリの上限設定				○
HIO_SetIOBOXID	IOBOX の ID を設定				○
HIO_WriteFirmware	IOBOX のファームウェアの書き換え				○
HIO_ResetIOBOX	IOBOX のリセット				○
HIO_SetDispParam	表示パラメータの設定				○

（１５）システムメニュー

項目	設定内容	DT-700	DT-800	DT-900	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	○	○	○	○
	キャリブレーション起動		○		
	OS バージョン表示	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		○	○	
	AP インストール		○	○	○
	子機作成		○	○	○
	通信ポート設定		○	○	
	通信速度設定		○	○	
	プロトコル			○	○
子機作成	本体送信		○	○	○
	本体受信		○	○	○
	転送ドライブ		○	○	○
ユーティリティ	AP インストール	○			
	ファイル転送	○		○	○
	メモリ初期化	○			
	ファイル送信		○		
	ファイル受信		○		
	ドライブフォーマット		○	○	○
	メモリサイズ変更		○	○	○
	ファイルモード			○	
ファイル転送	ホスト受信	○			
	ホスト送信	○			

（１６）動作環境メニュー

項目	設定内容	DT-700	DT-800	DT-900	本機
TOP 項目選択	環境	○	○	○	○
	表示モード	○	○	○	○
	通信セット	○			
	バーコード	○	○	○	○
	ID セット	○	○	○	○
環境	APO 時間	○	○	○	○
	ABO 時間	○	○	○	○
	キークリック ON/OFF	○	○	○	○
	ブザー音量	○	○	○	○
	自動コントラスト ON/OFF		○		
	警告メッセージ ON/OFF		○		
表示モード	フォントモード	○	○	○	○
	メッセージ	○	○	○	○
フォントモード	サイズ(12/16/24)		○		
	サイズ(6/8/10)			○	○
	タイプ(標準/強調)		○	○	○
通信セット	通信ポート	○			
	通信速度	○			
	データ長	○			
	パリティ	○			
	ストップビット	○			
バーコード	読取り回数	○	○	○	○
	照合回数	○	○	○	○
	タイムアウト	○	○	○	○
	キャリブレーション			○	
ID セット	機器 ID	○	○	○	○
	代理店 ID		○	○	○

最終ページ