

パナソニックオープンコム端末(KX-G7201N専用)  
ユーザーアプリケーション  
プログラミングガイド

Ver 1.4

本仕様は予告なく変更する場合があります。

2001年7月16日

九州松下電器株式会社  
オープンコムジャパン株式会社

## 目次

<b>1.</b>	<b>概要</b> .....	<b>8</b>
<b>2.</b>	<b>SCのデータ処理</b> .....	<b>9</b>
2.1.	メッセージ通信処理の概要 .....	9
2.2.	衛星とSC間の通信制御 .....	10
2.2.1.	メインシステムで衛星キューを制御するには .....	10
2.2.2.	ユーザーアプリケーションで衛星キューを制御するには .....	11
2.2.3.	メインシステムとユーザーアプリケーション両方でキューを制御するには .....	12
2.3.	DTEとSC間の通信制御 .....	13
2.3.1.	プロトコルモード/バイトモード通信をする場合 .....	13
2.3.2.	ユーザーアプリケーションで通信を制御する場合 .....	14
<b>3.</b>	<b>ユーザーアプリケーションのプログラミング</b> .....	<b>15</b>
3.1.	スタートアップルーチン .....	15
3.2.	RAMの初期化 .....	15
3.3.	ユーザーアプリケーションプログラムへの分岐 .....	15
3.4.	割り込み .....	15
3.5.	スタックエリア .....	15
3.6.	メモリーマップ .....	16
<b>4.</b>	<b>ユーザーアプリケーションの起動方法</b> .....	<b>17</b>
4.1.	SCがパワーオンする時に起動 .....	17
4.2.	自動送信コマンド ( KXB01,B02,B05 ) で指定された条件で起動 .....	17
<b>5.</b>	<b>KMEライブラリ</b> .....	<b>18</b>
5.1.	CONFIGURATION .....	18
5.2.	COMMUNICATION FUNCTION .....	18
5.3.	SYSTEM ANNOUNCE .....	18
5.4.	SC-ORIGINATED MESSAGE .....	19
5.5.	SC-TERMINATED MESSAGE .....	19
5.6.	POWER CONTROL .....	20
5.7.	SERIAL INTERFACE .....	20
5.8.	TIME FUNCTIONS .....	21
5.9.	I/O FUNCTIONS .....	21
5.10.	MEASUREMENT .....	21
5.11.	GPS INFORMATION .....	22
5.12.	STATUS INFORMATION .....	22
5.13.	OTHER FUNCTIONS .....	23
5.14.	KX COMMAND .....	24
<b>6.</b>	<b>各関数の詳細説明</b> .....	<b>26</b>
	set_desired_ncc_id .....	27
	set_pin_code .....	28

set_ncc_search_mode .....	29
all_msg_polling.....	30
s_msg_polling .....	31
globalgram_polling.....	32
chk_failed_polling.....	33
chk_ib_tx.....	34
clear_active_msg .....	36
clear_mha_ref_num_msg .....	37
clear_all_ib_q.....	38
clear_all_ob_q.....	39
req_pos_report_to_ncc.....	40
req_select_next_downlink.....	41
chk_polling_event.....	42
get_polling_info .....	43
req_send_ib_message .....	44
req_send_ib_report .....	46
req_send_ib_globalgram .....	48
req_send_ib_enh_globalgram.....	49
req_send_ib_pos_report.....	50
req_send_ob_message .....	51
get_unget_ob_id_for_user .....	53
remove_ob_q .....	56
get_ob_type.....	57
get_ob_ncc_id.....	58
get_ob_msg_subject_ind.....	59
get_ob_msg_msg_body_type .....	60
get_ob_msg_or_quan .....	61
get_ob_msg_msg_len .....	62
get_ob_msg_msg_body_index .....	63
get_ob_message .....	64
get_ob_user_command .....	65
get_ob_globalgram_ref_num.....	66
get_ob_globalgram_or_ind .....	67
get_ob_globalgram_msg_len .....	68
get_ob_globalgram.....	69
go_sleep_time .....	70
go_wait.....	71
rf_block_power_on.....	72
rf_block_power_off.....	73
check_rf_block_power.....	74
chk_rx_buff .....	75
get_rx_data .....	76
chk_tx_ready.....	77
set_tx_data .....	78
cts_act .....	79
cts_neg .....	80

chk_cd .....	81
chk_rts .....	82
chk_rx_buff_port .....	83
get_rx_data_port .....	84
chk_tx_ready_port .....	85
set_tx_data_port .....	86
get_inc_timer .....	87
set_timer .....	88
check_timer .....	90
get_digital_port .....	91
set_digital_port .....	92
get_adcon_data .....	93
get_adcon_data_10bit .....	94
req_pos_calc_term_from_user .....	95
req_pos_calc_from_user .....	96
chk_pos_calc_result .....	97
get_pos_age .....	98
get_lat_val .....	99
get_lon_val .....	100
set_lat_val .....	101
set_lon_val .....	102
set_pdop_th .....	103
get_pdop_th .....	104
get_x05_str .....	105
get_x06_str .....	107
get_st_status .....	108
get_active_mha_ref_num .....	109
get_sat_no .....	110
get_ncc_quan .....	111
get_ncc_id_prio .....	112
get_num_of_ob_msgs .....	113
get_num_of_ib_msgs .....	114
get_week_time_val .....	115
calc_gps_week .....	116
get_total_sats .....	117
get_check_errs .....	118
exit_user_apl .....	119
break_point .....	120
chk_break_point .....	121
get_application_dbg .....	122
led_on .....	123
led_off .....	124
suspend_ib_msg_tx .....	125
resume_ib_msg_tx .....	126
get_utc_time .....	127
set_utc_time .....	128

get_local_time.....	129
get_convert_time .....	130
calc_distance.....	131
req_apl_sat_predict .....	132
chk_apl_sat_predict_resault .....	133
get_apl_sat_predict_time .....	134
get_sys_info .....	135
get_kme_serial_no.....	136
os_monitor .....	137
print_double.....	138
orb_sprintf1 .....	139
orb_sprintf2 .....	140
orb_sprintf3 .....	141
orb_sprintf4 .....	142
orb_sprintf5 .....	143
orb_sprintf6 .....	144
orb_sprintf7 .....	145
set_kxs01 .....	146
set_kxs02 .....	147
set_kxs03 .....	148
set_kxs04 .....	149
set_kxs05 .....	150
set_kxs06 .....	151
set_kxs07 .....	152
set_kxs08 .....	153
set_kxs14 .....	154
set_kxs15 .....	155
set_kxs16 .....	156
set_kxs17 .....	157
set_kxs18 .....	158
set_kxs23 .....	159
set_kxs24 .....	160
set_kxs25 .....	161
set_kxs26 .....	162
set_kxs27 .....	163
set_kxs28 .....	164
set_kxs29 .....	165
set_kxs30 .....	166
set_kxs31 .....	168
set_kxs32 .....	170
set_kxs33 .....	171
set_kxs34 .....	172
set_kxs35 .....	173
set_kxs36 .....	174
set_kxs37 .....	175
set_kxs39 .....	176

set_kxs40 .....	177
set_kxs41 .....	178
set_kxs42 .....	179
set_kxs43 .....	180
set_kxs43_port.....	181
set_kxs44 .....	183
set_kxs45 .....	184
set_kxs46 .....	185
set_kxs47 .....	186
set_kxs48 .....	187
set_kxs50 .....	188
set_kxs51 .....	189
set_kxs52 .....	190
set_kxs53 .....	191
set_kxs55 .....	192
set_kxs56 .....	193
set_kxs57 .....	194
set_kxs58 .....	195
set_kxs60 .....	196
set_kxs61 .....	197
set_kxs63 .....	198
set_kxs64 .....	199
set_kxs67 .....	200
set_kxs68 .....	201
set_kxs69 .....	202
set_kxs70 .....	203
set_kxs71 .....	204
set_kxs72 .....	205
set_kxs74 .....	206
set_kxs75 .....	207
set_kxs77 .....	208
set_kxs78 .....	209
set_kxs79 .....	210
set_kxs80 .....	211
set_kxs81 .....	212
set_kxs82 .....	213
set_kxs83 .....	214
set_kxs84 .....	215
set_kxs85 .....	216
set_kxs86 .....	217
set_kxs87 .....	219
set_kxs90 .....	220
set_kxs91 .....	221
set_kxs94 .....	222
set_kxs95 .....	223
set_kxs96 .....	224

set_kxd01.....	225
get_kxd02.....	226
set_kxm01.....	227
set_kxb01.....	228
set_kxb02.....	229
set_kxb05.....	230
set_kxb00.....	231
<b>7. プログラミング ヒント .....</b>	<b>232</b>
7.1. GCCへのメッセージ送信 .....	232
7.2. GCCからのメッセージ受信 .....	233
7.3. GCC内のメッセージを取り出す .....	234
<b>8. ユーザーアプリケーションのプログラムのインストール方法 .....</b>	<b>235</b>
8.1. 接続 .....	235
8.2. インストール方法 .....	236
8.3. ユーザーアプリケーションの削除 .....	236
8.4. ユーザーアプリケーションの動作状態の確認 .....	237
<b>9. ユーザーアプリケーションのプログラムのデバッグ方法 .....</b>	<b>238</b>
9.1. メモリーダンプ .....	238
9.2. メモリー書き込み .....	239
9.3. スタック開始アドレスの獲得 .....	239
9.4. ブレークポイント .....	239
9.5. アウトバウンドキューへのメッセージ登録 .....	240
<b>10. ユーザーアプリケーションの開発上の注意点 .....</b>	<b>241</b>
<b>11. KX-G7100/G7101用プログラムとの互換性 .....</b>	<b>242</b>
11.1. UPファイル .....	242
11.2. ソースファイル .....	242
11.3. プログラムサイズ .....	242
11.4. KMEライブラリー内の関数の追廃リスト .....	243
11.4.1. KX-G7100/G7101用ライブラリーから削除された関数 .....	243
11.4.2. KX-G7100/G7101用ライブラリーに追加された関数 .....	244
<b>12. ユーザーアプリケーションの開発環境 .....</b>	<b>245</b>
12.1. ハードウェア環境 .....	245
12.2. ソフトウェア .....	245
12.3. コンパイラーのインストール .....	246
12.4. ディレクトリーの作成とファイルのコピー .....	246
12.5. PCの環境変数の設定 .....	246
<b>13. 変更履歴 .....</b>	<b>247</b>

## 1. 概要

この資料では、パナソニックのオープンコム端末(KX-G7201N 以下SC)ユーザーアプリケーションを作成する上での、開発環境の構築方法、プログラム作成上のノウハウ、KMEライブラリー内の各サブルーチンの説明に付いて記述しています。

SCは、内蔵のパラメータを設定することで、各種の用途に使用できるよう設計されていますが、ユーザーアプリケーションをインストールする事により、より詳細に使用目的にあわせた動作を設定する事ができます。

本SCには、ユーザーアプリケーション用に128KバイトのROMと32KバイトのRAMを装備しています。また、ユーザーアプリケーション用として、SCがもつ情報(例えば、デジタルI/O状況、SCの衛星捕捉状況など)の参照、SCへの測位要求、そしてメッセージの送受信の処理を行う為のサブルーチン群をKMEライブラリーとして提供しています。

SCには、CPUとして日立製作所製H8S/2227シリーズ(HD6432227TE)を使用していますので、ユーザーアプリケーションを作成するにあたっては、専用の開発ツール(Cコンパイラ等)が必要です。[12.ユーザーアプリケーションの開発環境](#)を参照ください。また、この開発ツールについてはオープンコムジャパンにお問い合わせください。

CPU	日立製作所 H8S/2227シリーズ HD6432227TE
ユーザーアプリケーション用ROM	128KB
ユーザーアプリケーション用RAM	32KB
ユーザーアプリケーション用 スタックエリア	3200Byte



## 2. SCのデータ処理

### 2.1. メッセージ通信処理の概要

SCを使用しての衛星とのメッセージ通信動作は、(1)衛星とのデータ送受信、(2)内部でのデータ処理、(3)外部端末(以下DTE)とのデータ入出力にわけることができます。

#### (1)SCと衛星間のメッセージ通信

この部分は、SCのファームウェア(メインシステム)で処理されます。

衛星との通信は、アウトバウンドキュー(衛星からの受信メッセージ)とインバウンドキュー(衛星からの受信メッセージ)をアクセスする事で可能です。インバウンドキューへの書込み(衛星への送信)は、メインシステムおよびユーザーアプリケーションのどちらもアクセス可能です。アウトバウンドキューのメッセージ(衛星からの受信)は、競合を防止するため、アクセス権をKXS69で設定します。

#### (2)SC内部でのデータ処理

ユーザーアプリケーションがインストールされていない場合には、メインシステムが処理を行います。ユーザーアプリケーションが動作する設定の場合には、タイムシェアリングにてユーザーアプリケーションが動作します。

#### (3)SCと外部端末との通信

2チャンネルあるシリアルポートを用いて、DTEとの通信を行います。メインシステムはオーブコム社シリアルインターフェース仕様に準拠した通信方式であるプロトコルモードまたはバイトモードをサポートしています。この通信方式が使用できるポートは2つのシリアルポートの何れか一方のみです。(使用するポートはKXS96で指定します。)また、ユーザーアプリケーションの制御下で通信を行う場合は、ユーザーアプリケーションで通信方式を定義した上でDTEとの通信を行う事になります。

## 2.2. 衛星とSC間の通信制御

### 2.2.1. メインシステムで衛星キューを制御するには

この場合、KXS69= 0、KXS68= 0と設定します。

衛星キューとDTEは、メインシステムの制御下でオーブコム社シリアルインターフェース仕様に準拠した通信方式であるプロトコルモードまたはバイトモードでDTEと通信を行います。また、この設定ではユーザーアプリケーションで作成したメッセージをインバウンドキューにセットして、衛星に送信することは可能ですが、受信メッセージをユーザーアプリケーションで処理することは出来ません。

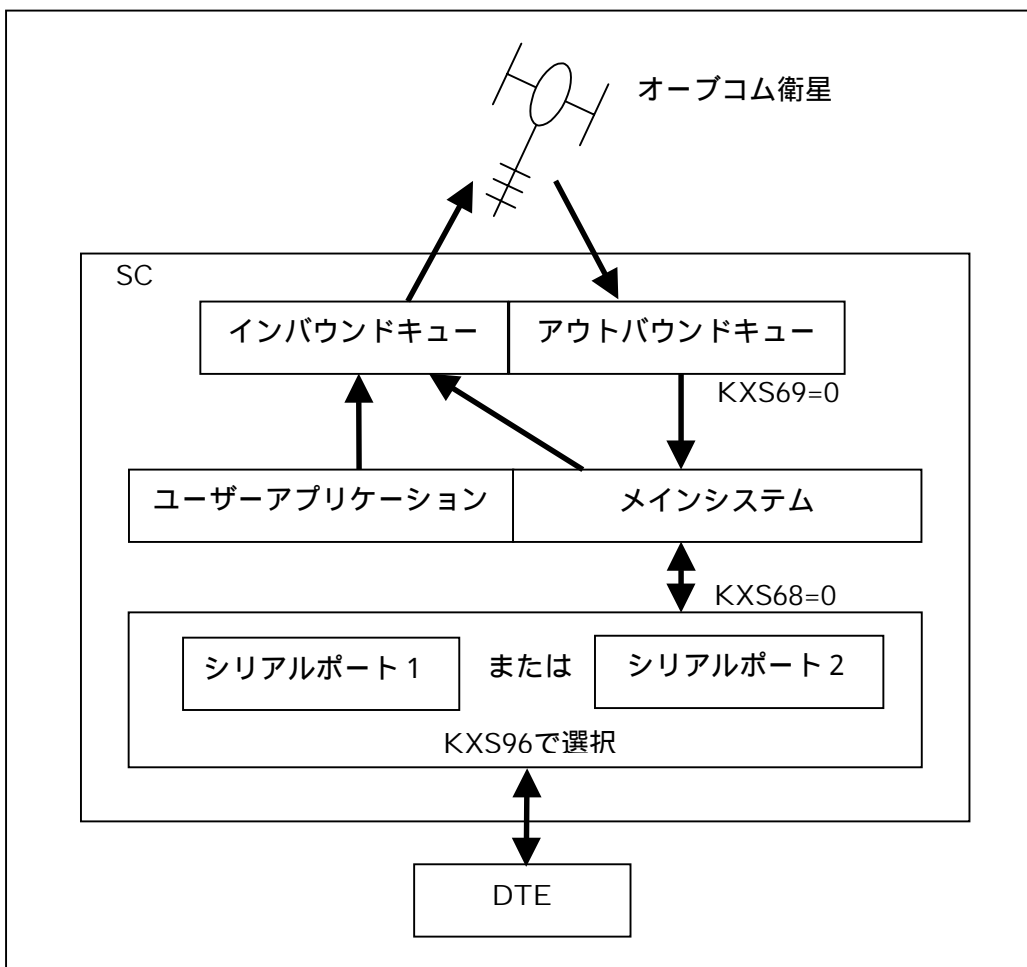


図 2-1

注意:

1. KXS69=0の場合、KXS68=1にするとメインシステムがシリアルインターフェースを制御できませんのでアウトバウンドキュー内のデータを出力できなくなります。このためアウトバウンドキューがオーバーフローする恐れがありますので、KXS69=0を設定する場合には、この設定は出来るだけ避けてください。

### 2.2.2. ユーザーアプリケーションで衛星キューを制御するには

この場合、KXS69= 1と設定します。

衛星からの受信メッセージは、ユーザーアプリケーションで処理を行います。ユーザーアプリケーションの制御下でDTEと通信を行う場合には、KXS68=1と設定しますが、通信方式はユーザーアプリケーション内で定義しなければなりません。

また、KXS68=0と設定した場合には、DTEからはメインシステムを経由してのプロトコルモードまたはバイトモードでのメッセージ送信は可能です。

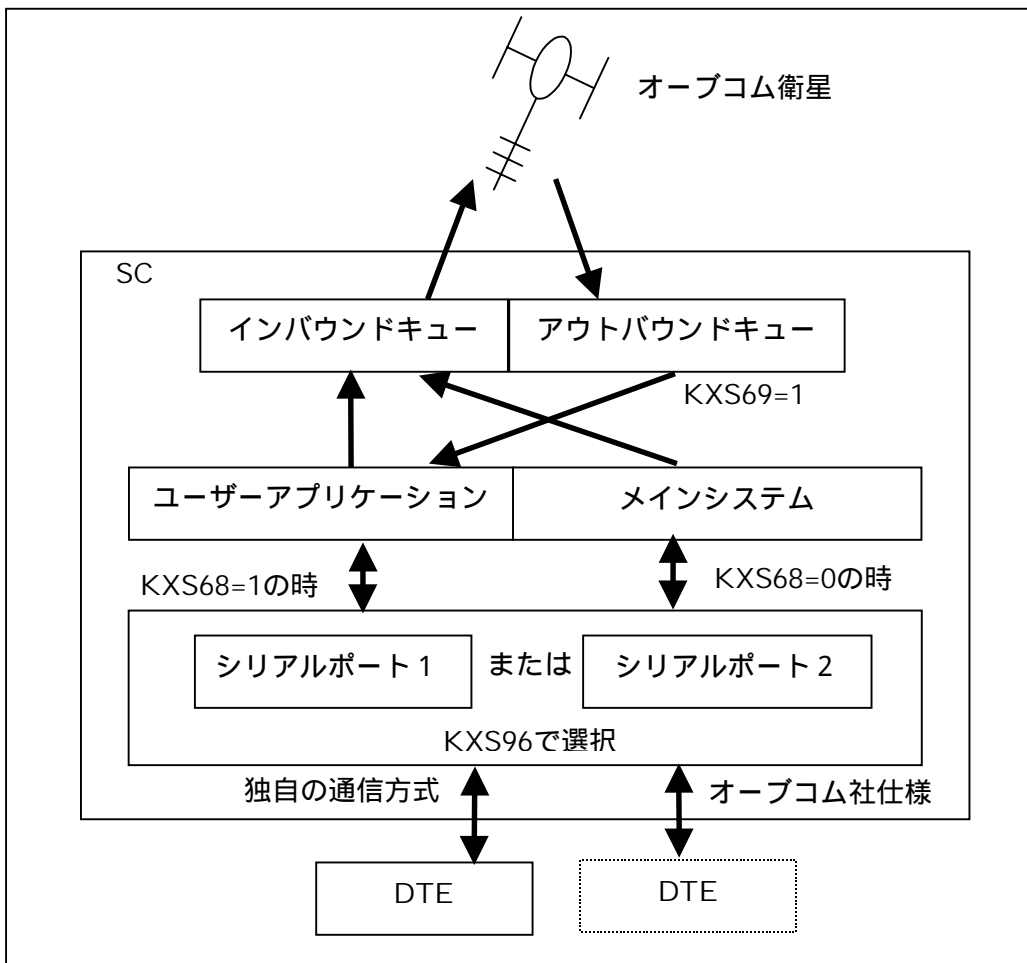


図 2-2

注意:

1. アウトバウンドキュー内の受信メッセージは、ユーザーアプリケーションにて削除処理しなければなりません。

### 2.2.3. メインシステムとユーザーアプリケーション両方でキューを制御するには

KXS69= 2、KXS68= 0にします。

衛星から受信するメッセージにメインシステムとユーザーアプリケーションの双方で処理させたいメッセージが混在している場合に設定します。衛星キューとDTEは、メインシステムの制御下でプロトコルモードまたはバイトモードでDTEと通信を行います。同時にユーザーアプリケーションは、アウトバウンドキュー内の必要なメッセージに対して処理をする事ができます。

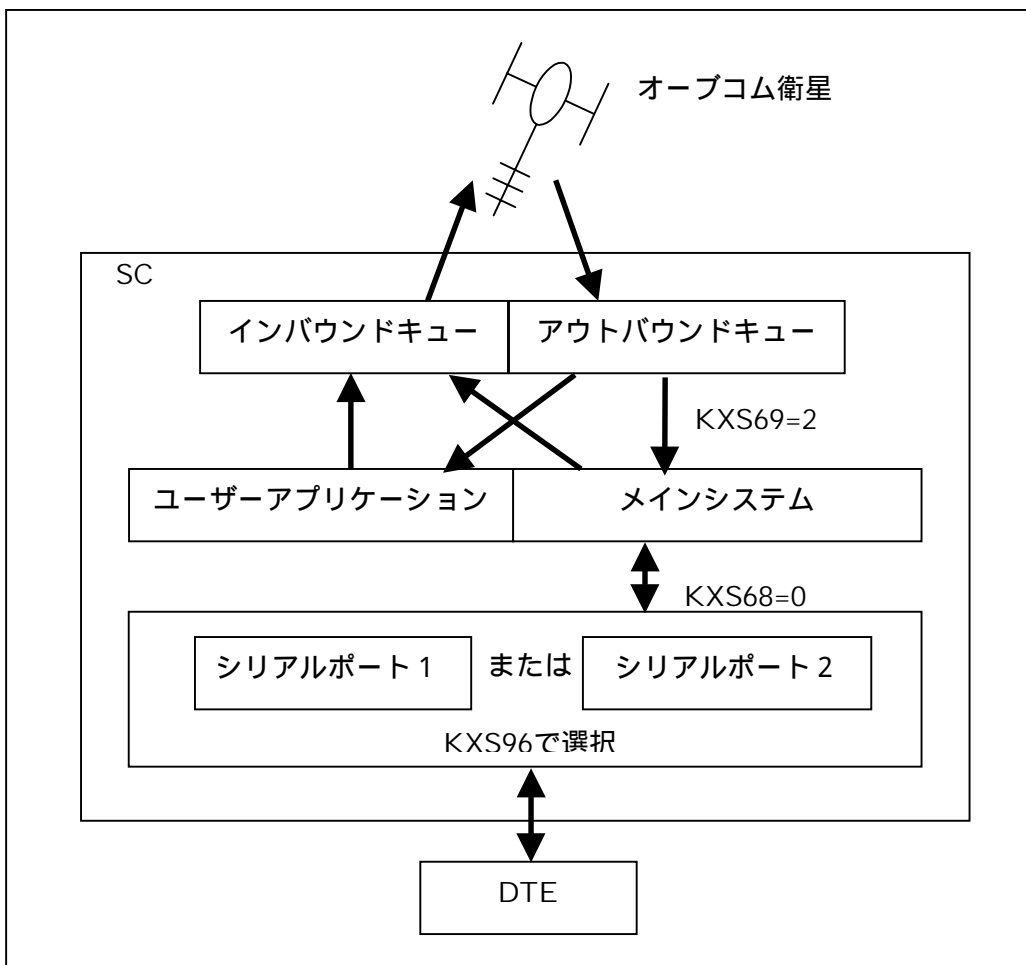


図 2-3

注意:

- (1) アウトバウンドキュー内のメッセージは、メインシステムとユーザーアプリケーションの両方の処理が終わった後で削除されるようにするため、ユーザーアプリケーションは、全ての受信メッセージ毎に削除できる事をメインシステムに知らせなければなりません。メインシステムでは、ユーザーアプリでの処理終了通知を受けて、キューのメッセージを削除します。
- (2) KXS68=1に設定すると、メインシステムが読み出したアウトバウンドキューをDTEに出力できないため、場合によってはアウトバウンドキューがオーバーフローする恐れがあります。この設定は出来るだけ避けてください。

## 2.3. DTEとSC間の通信制御

SCは、シリアルポートを2チャンネル備えています。また、メインシステムは、オーブコム社シリアルインターフェース仕様に準拠した通信方式であるプロトコルモードおよびバイトモードをサポートしていますが、この通信方式が使用できるポートは2つのシリアルポートの何れか一方のみです。使用するシリアルポートはKXS96で選択します。また、ユーザーアプリケーションの制御下で通信を行う場合は、ユーザーアプリケーションで通信方式を定義した上でDTEとの通信を行う事になります。

### 2.3.1. プロトコルモード/バイトモード通信をする場合

この場合、KXS68の設定を0(KXS68=0)にします。プロトコルモードまたはバイトモードは、メインシステムがサポートしていますので、この通信方式でDTEと通信する場合には、メインシステムがシリアルポートを使用するように設定します。通信方式としてプロトコルモードで通信を行う場合は、KXS31=0、バイトモードの場合にはKXS31=1にします。使用するポートはKXS96で選択します。但し、ユーザーアプリケーションでは、SCがサポートしているこの通信方式は使用できません。

本通信方式が設定されている場合においても、2つのシリアルポートの何れからもCTRL+KXORBの入力でコマンドモードに入る事ができます。但し、コマンドモード動作中は、プロトコルモードおよびバイトモードの通信はできません。

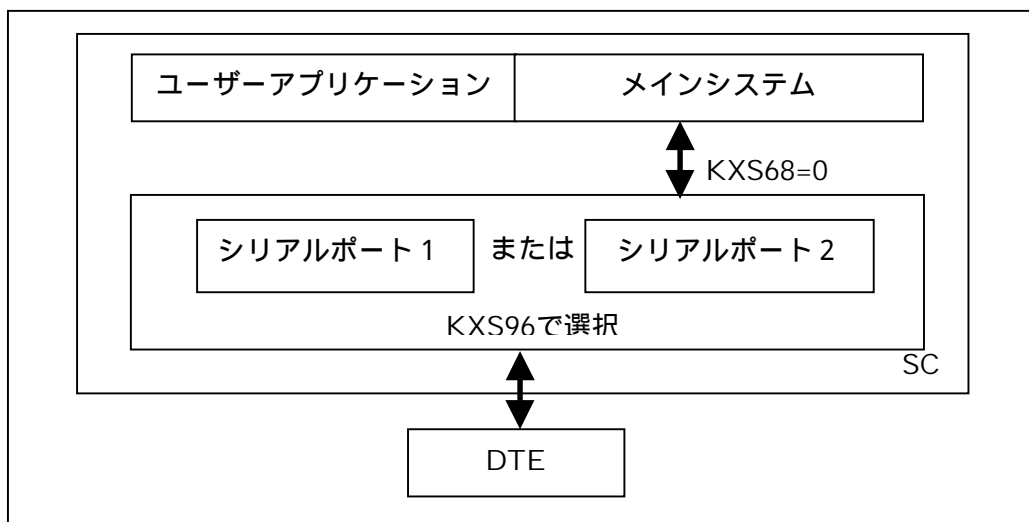


図 2-4

### 2.3.2. ユーザーアプリケーションで通信を制御する場合

この場合、KXS68の設定を1(KXS68=1)にします。ユーザーアプリケーションにシリアルドライバの制御権が亘り、メインシステムではアクセスできません。使用するシリアルポートはKXS96で選択します。

本通信方式が設定されている場合においても、2つのシリアルポートの何れからでもCTRL+KXORBの入力でコマンドモードに入る事ができます。但し、コマンドモード動作中は、プロトコルモードおよびバイトモードの通信はできません。

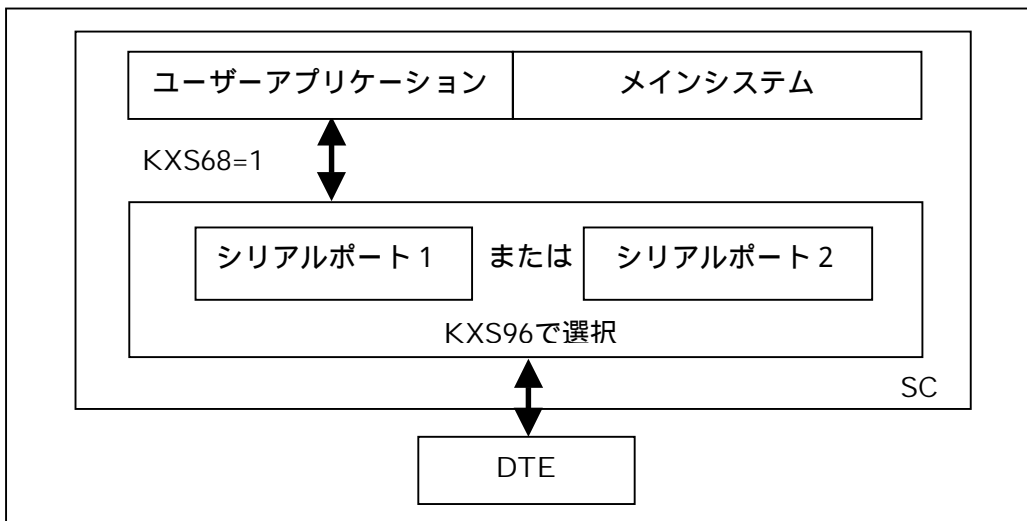


図 2-5

### 3. ユーザーアプリケーションのプログラミング

ユーザーアプリケーションを作成するには、C言語で記述したソースファイルをコンパイル・リンクし、さらにSCにインストールできる形にコンバートするための専用の開発ツールが必要です。更に、SCの情報や資源にアクセスするためのサブルーチン群が入っているライブラリーファイル(KMEライブラリー)が必要です。

#### 3.1. スタートアップルーチン

通常、マイコンのプログラムを記述する場合、スタート時の初期化処理として以下の項目について設定を行わなければいけません。アプリケーションを起動する前にメインシステムで行われます。

- ・ レジスタの初期化
- ・ ユーザーアプリケーションプログラムへの分岐

#### 3.2. RAMの初期化

ユーザーアプリケーションは、バックアップされた32キロバイトのRAMを使用でき、この領域はユーザーアプリケーションをインストール時に0で初期化されます。

#### 3.3. ユーザーアプリケーションプログラムへの分岐

プログラムプログラムへの分岐処理は、メインシステムによって行われますが、ユーザーアプリケーションのmain関数は、“main”という名前にする必要があります。そして起動されたプログラムは、メインシステムによって時分割処理されます。

#### 3.4. 割り込み

ユーザーアプリケーションは、CPUの割り込み機能を使う事ができません。

#### 3.5. スタックエリア

ユーザーアプリケーションのスタック領域はシステムRAM上に割り当てられており、サイズは3200バイトです。スタック領域の先頭アドレスは、コマンド“stack”で確認できます。(9.3.スタック開始アドレスの獲得を参照ください)

ユーザーアプリケーションにおいて、ローカル変数領域を取りすぎるとスタックオーバーフローを起こす可能性があるため、大きなサイズの変数(例:受信バッファ等)は、グローバル変数として宣言することを推奨します。

また、標準関数の“sprintf()”は、スタックを多く消費するため、ユーザーアプリケーションの不具合の原因になる可能性があります。特に浮動小数点変数を変換すると数百バイトのスタックを消費します。標準関数の“sprintf()”に変わる“print\_double()”や“orb\_sprintf1()”等の代用関数を準備していますので使用してください。

### 3.6. メモリーマップ

この端末には、システム用のROM/RAMとユーザーアプリケーション用のROM/RAMがあります。ユーザーアプリケーション用のROMサイズは128KB (0x60000 – 0x7FFFF)、RAMサイズは32KB(0x228000 – 0x22FFFF)です。

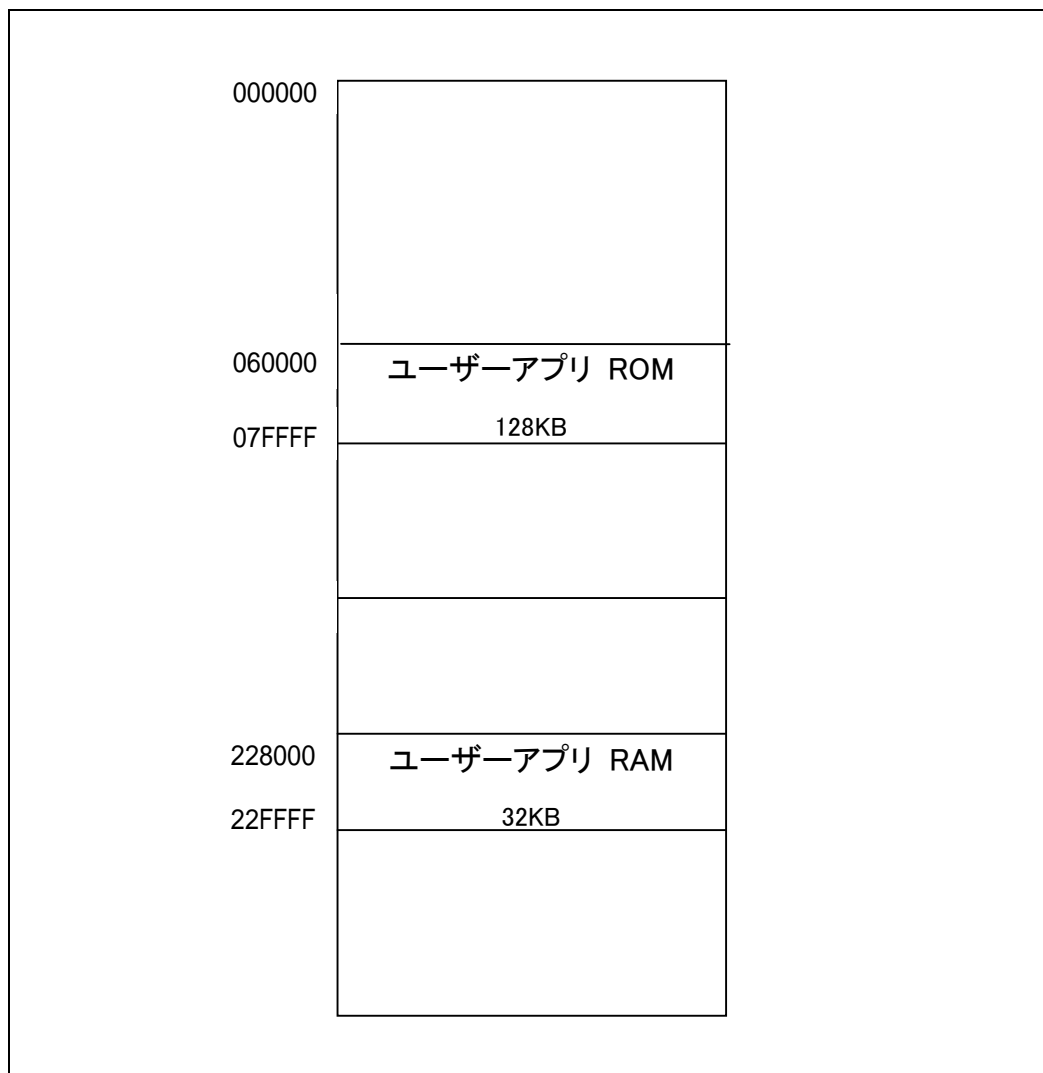


図 3-1



## 4. ユーザーアプリケーションの起動方法

SCにロードされたユーザーアプリケーションを起動させるには、以下の2つの方法があります。

### 4.1. SCがパワーオンする時に起動

SCがパワーオンして数秒後にユーザーアプリケーションが起動します。設定は、KXS67の設定を1(KXS67=1)にします。

### 4.2. 自動送信コマンド（KXB01,B02,B05）で指定された条件で起動

KXBコマンドで指定されたタイミングで指定された条件を満たした時ユーザーアプリケーションを起動します。

## 5. KMEライブラリ

ユーザーアプリケーション用のライブラリーをKMEライブラリーとして提供しています。このライブラリー内の関数をサブルーチンコールすることで、SCの情報の獲得・設定、SCに対しての処理要求等を行う事ができます。このライブラリーで提供している関数の詳細を以下に記載します。

### 5.1. Configuration

SCのコンフィグレーションパラメータを設定します。

関数	説明	頁
set_desired_ncc_id()	接続するGCCを設定する	27
set_pin_code()	PIN CODEを設定する	28
set_ncc_search_mode()	衛星サーチモードを設定する	29

### 5.2. Communication Function

SCのキュー内メッセージの削除やホスト局に対してメッセージの要求を出したりします。

関数	説明	頁
all_msg_polling()	GCC内の全メッセージの送信要求	30
s_msg_polling()	GCC内の150バイト以下の全メッセージ送信要求	31
globalgram_polling()	衛星内のグローバルグラムメッセージ送信要求	32
chk_failed_polling()	ポーリングに失敗した時の原因を獲得する	33
chk_ib_tx()	指定メッセージの送信状況を確認する	34
clear_active_msg()	送受信中メッセージを削除する	36
clear_mha_ref_num_msg()	指定のメッセージをキューから削除する	37
clear_all_ib_q()	インバウンドキュー内メッセージを全て削除する	38
clear_all_ob_q()	アウトバウンドキュー内メッセージを全て削除する	39
req_pos_report_to_ncc()	GCCへポジションレポートの送信を要求する	40
req_select_next_downlink()	捕捉衛星を次のチャンネルの衛星に切り替える	41

### 5.3. System Announce

ホスト局からのポーリングをチェックします

関数	説明	頁
chk_polling_event()	GCCから受信したポーリングがあるか確認する	42
get_polling_info()	ポーリングの応答内容を獲得する	43

## 5.4. SC-Originated Message

GCCにメッセージを送信する為、端末の送信キューにメッセージをセットする。

関数	説明	頁
req_send_ib_message()	送信キューにメッセージをセットする	44
req_send_ib_report()	送信キューにレポートをセットする	46
req_send_ib_globalgram()	送信キューにグローバルメッセージをセットする	48
req_send_ib_enh_globalgram()	送信キューにエンハンスグローバルメッセージをセットする	49
req_send_ib_pos_report()	送信キューにポジションレポートをセットする	50

DTEにメッセージを送信する為、端末の受信キューにメッセージをセットする。

関数	説明	頁
req_send_ob_message()	受信キューにメッセージをセットする	51

## 5.5. SC-Terminated Message

GCCから送信されたメッセージが端末の受信キューに保存されているので、そのメッセージを獲得する。

関数	説明	頁
get_unget_ob_id_for_user()	アウトバウンドキューのデータのポインタを獲得する	53
remove_ob_q()	アウトバウンドキューからデータを削除する	56
get_ob_type()	データ種を獲得する	57
get_ob_ncc_id()	メッセージのGCC IDを獲得する	58
get_ob_msg_subject_ind()	メッセージのサブジェクトを獲得する	59
get_ob_msg_msg_body_type()	メッセージのメッセージタイプを獲得する	60
get_ob_msg_or_quan()	メッセージの送信元を獲得する	61
get_ob_msg_msg_len()	メッセージ長を獲得する	62
get_ob_msg_msg_body_index()	メッセージ本体のインデックスを獲得する	63
get_ob_message ()	メッセージを獲得する	64
get_ob_user_command()	コマンドを獲得する	65
get_ob_globalgram_ref_num()	グローバルメッセージの認識番号を獲得する	66
get_ob_globalgram_or_ind()	グローバルメッセージの送信元を獲得する	67
get_ob_globalgram_msg_len()	グローバルメッセージのメッセージ長を獲得する	68
get_ob_globalgram ()	グローバルメッセージを獲得する	69

## 5.6. Power Control

端末をスリープさせる。

関数	説明	頁
go_sleep_time()	指定時間スリープさせる	70
go_wait ()	指定時間ユーザソフトをサスペンドする	71

ブロック電源を制御する。

関数	説明	頁
rf_block_power_on()	RFブロック電源をONする	72
rf_block_power_off()	RFブロック電源をOFFする	73
chk_rf_block_power()	RFブロックの電源状態を獲得する	74

## 5.7. Serial Interface

DTEに対してデータの送受信を行う。

関数	説明	頁
chk_rx_buff()	メインポートのRS232Cの受信バッファをチェックする	75
get_rx_data()	メインポートのRS232Cの受信バッファからデータを獲得する	76
chk_tx_ready()	RS232Cの送信バッファが満杯かをチェックする	77
set_tx_data()	メインポートのRS232Cの送信バッファにデータをセットする	78
cts_act()	メインポートのCTSポートをアクティブにする	79
cts_neg()	メインポートのCTSポートをネガティブにする	80
chk_cd()	メインポートのCD信号の状態をチェックする	81
chk_rts()	メインポートのRTS信号の状態をチェックする	82
chk_rx_buff_port()	RS232Cの受信バッファをチェックする	83
get_rx_data_port()	RS232Cの受信バッファからデータを獲得する	84
chk_tx_ready_port()	RS232Cの送信バッファが満杯かをチェックする	85
set_tx_data_port()	RS232Cの送信バッファにデータをセットする	86

## 5.8. Time Functions

セットしたタイマをチェックする。

関数	説明	頁
get_inc_timer()	2msecカウンタ値を獲得する	87
set_timer()	タイマをセットする	88
check_timer()	セットしたタイマが経過したか確認する	90

## 5.9. I/O Functions

デジタル入出力ポート各2ポートをコントロールする。

関数	説明	頁
get_digital_port()	デジタルポートの状態を獲得する	91
set_digital_port()	デジタルポートをセットする	92

アナログポートをコントロールする。

関数	説明	頁
get_adcon_data()	アナログポートのA/D値を獲得する	93
get_adcon_data_10bit()	アナログポートのA/D値を10ビットで獲得する	94

## 5.10. Measurement

端末の測位機能をコントロールする。

関数	説明	頁
req_pos_calc_term_from_user()	測位処理を中止する	95
req_pos_calc_from_user()	測位をさせる	96
chk_pos_calc_result()	測位が完了したかチェックする	97
get_pos_age()	前回測位してからの経過時間を獲得する	98
get_lat_val()	測位結果(緯度)を獲得する	99
get_lon_val()	測位結果(経度)を獲得する	100
set_lat_val()	端末の保持している測位情報(緯度)をセットする	101
set_lon_val()	端末の保持している測位情報(経度)をセットする	102
set_pdop_th()	PDOPのしきい値をセットする	103
get_pdop_th()	PDOPのしきい値を獲得する	104

## 5.11. GPS Information

端末がGPSを搭載している時、GPSから各種情報を獲得する。

関数	説明	頁
get_x05_str()	X05 センテンスを獲得する	105
get_x06_str()	X06 センテンスを獲得する	107

## 5.12. Status Information

端末から各種情報を獲得する。

関数	説明	頁
get_st_status()	端末の動作状態を獲得する	108
get_active_mha_ref_num()	送信中メッセージのリファレンス番号を獲得する	109
get_sat_no()	捕捉している衛星番号を獲得する	110
get_ncc_quan()	捕捉している衛星とリンクしているGCCの数を獲得する	111
get_ncc_id_prio()	GCCに送信可能なメッセージプライオリティを獲得する	112
get_num_of_ob_msgs()	アウトバウンドキュー内メッセージの数を獲得する	113
get_num_of_ib_msgs()	インバウンドキュー内メッセージの数を獲得する	114
get_week_time_val()	UTC時00:00:00からの経過秒を獲得する	115
calc_gps_week()	GPS週数を獲得する	116
get_total_sats()	システム内の衛星数を獲得する	117
get_check_errs()	最後のステータスパケット送信後、ダウンリンクチェックサムエラー	118

## 5.13. Other Functions

関数	説明	頁
exit_user_apl()	ユーザーアプリケーションを強制終了する	119
break_point()	デバックのためのブレイクコール	120
chk_break_point()	指定したブレイクポイントの有無状態を獲得する	121
get_application_dbg()	デバック機能ON/OFFを獲得する	122
led_on()	LEDを点灯する	123
led_off()	LEDを消灯する	124
suspend_ib_msg_tx()	衛星への送信を一時中断する.	125
resume_ib_msg_tx()	衛星への送信を再開する	126
get_utc_time()	UTC時を獲得する	127
set_utc_time()	UTC時を設定する	128
get_local_time()	ローカル時を獲得する	129
get_convert_time()	ローカルに変換する	130
calc_distance()	2点間の距離を算出する	131
req_apl_sat_predict()	次の衛星飛来時刻計算を要求する	132
chk_apl_sat_predict_result()	衛星飛来計算が終了したか確認する	133
get_apl_sat_predict_time()	算出した飛来時刻を獲得する	134
get_sys_info()	自己診断結果を獲得する	135
get_kme_serial_no()	SCのシリアル番号を獲得する	136
os_monitor()	SCの動作状態を監視する	137
print_double()	数値の文字列変換	138
orb_sprintf1()	数値の文字列変換	139
orb_sprintf2()	数値の文字列変換	140
orb_sprintf3()	数値の文字列変換	141
orb_sprintf4()	数値の文字列変換	142
orb_sprintf5()	数値の文字列変換	143
orb_sprintf6()	数値の文字列変換	144
orb_sprintf7()	数値の文字列変換	145

ユーザーアプリケーションを終了する場合は必ず"exit\_user\_apl()"をコールしてください。

## 5.14. KX Command

ユーザーアプリケーション用KXコマンド(KXS, KXBコマンド)一覧表。

設定関数	獲得関数	内容	ページ
set_kxs01()	get_kxs01()	登録しているGCC番号	146
set_kxs02()	get_kxs02()	デフォルト ホールト	147
set_kxs03()	get_kxs03()	デフォルト プライオリティ	148
set_kxs04()	get_kxs04()	デフォルト レポート O/R インディケータ	149
set_kxs05()	get_kxs05()	デフォルト メッセージ/グローバルグラム O/R インディケータ	150
set_kxs06()	get_kxs06()	デフォルト ack レベル	151
set_kxs07()	get_kxs07()	デフォルト メッセージ ホディ タイプ	152
set_kxs08()	get_kxs08()	デフォルト サービス タイプ	153
-----	-----	-----	-----
set_kxs14()	get_kxs14()	GCCサーチモード	154
set_kxs15()	get_kxs15()	ダウンリンクチャンネル	155
set_kxs16()	get_kxs16()	ダウンリンクチェックサムエラースレッシュ	156
set_kxs17()	get_kxs17()	ダウンリンクチェックサムエラースレッシュをカウントするフレーム数	157
set_kxs18()	get_kxs18()	連続測位モード	158
-----	-----	-----	-----
set_kxs23()	get_kxs23()	緯度・経度	159
set_kxs24()	get_kxs24()	測位モード	160
set_kxs25()	get_kxs25()	KXBコマンドでのGPS測位情報のフォーマット	161
set_kxs26()	get_kxs26()	DTEに対するホーリングの応答の待ち時間	162
set_kxs27()	get_kxs27()	シリアルプロトコルのACK時間	163
set_kxs28()	get_kxs28()	シリアルプロトコルのリトライ回数	164
set_kxs29()	get_kxs29()	アポートレポート送信モード	165
set_kxs30()	get_kxs30()	アポートレポートの内容	166
set_kxs31()	get_kxs31()	RS232Cの通信モード(プロトコル/バイトモード)	168
set_kxs32()	get_kxs32()	バイトモードトリガー	170
set_kxs33()	get_kxs33()	バイトモードタイムアウト	171
set_kxs34()	get_kxs34()	バイトモード長	172
set_kxs35()	get_kxs35()	バイトモード時の、送信・受信SOM,EOM	173
set_kxs36()	get_kxs36()	バイトモードメッセージタイプ	174
set_kxs37()	get_kxs37()	パワーダウンモード	175
-----	-----	-----	-----
set_kxs39()	get_kxs39()	インアクティブインターバル	176
set_kxs40()	get_kxs40()	パワーセーブモード	177
set_kxs41()	get_kxs41()	インバウンドフロー制御	178
set_kxs42()	get_kxs42()	アウトバウンドフロー制御	179
set_kxs43()	get_kxs43()	メインポートのRS232C通信モード	180
set_kxs43_port()	set_kxs43_port()	RS232C通信モード	181
set_kxs44()	get_kxs44()	RS232C通信モード(半2重、全2重)	183
set_kxs45()	get_kxs45()	インバウンドメッセージトリートメント	184
set_kxs46()	get_kxs46()	アウトバウンドメッセージトリートメント	185
set_kxs47()	get_kxs47()	メッセージリキューオプション	186
set_kxs48()	get_kxs48()	インバウンド・アウトバウンドキューのサイズ	187
-----	-----	-----	-----
set_kxs50()	get_kxs50()	ピンコード	188



set_kxs51()	get_kxs51()	自動グローバルGRAMホーリング	189
set_kxs52()	get_kxs52()	GPS測地系	190
set_kxs53()	get_kxs53()	RTS論値仕様	191
-----	-----	-----	-----
set_kxs55()	get_kxs55()	KXBコマンドで送信するアナログポート番号	192
set_kxs56()	get_kxs56()	KXBコマンド移動距離検知の移動距離	193
set_kxs57()	get_kxs57()	KXBコマンドエリア検知のエリア	194
set_kxs58()	get_kxs58()	KXBコマンド速度検知の速度	195
-----	-----	-----	-----
set_kxs60()	get_kxs60()	KXBコマンドで送信するデータフォーマット	196
set_kxs61()	get_kxs61()	RS232Cドライバパワーセーブモード	197
-----	-----	-----	-----
set_kxs63()	get_kxs63()	タイムアウトウ	198
set_kxs64()	get_kxs64()	KXBコマンドでのクイックパワーダウンモード	199
-----	-----	-----	-----
set_kxs67()	get_kxs67()	端末電源ON時に、ユーザーアプリを起動	200
set_kxs68()	get_kxs68()	RS232C受信データの処理ルート	201
set_kxs69()	get_kxs69()	GCCからの受信データの処理ルート	202
set_kxs70()	get_kxs70()	ユーザーアプリケーションのデバッグモード	203
set_kxs71()	get_kxs71()	GCCからのコマンドリモートセッティングでのセットアップID	204
set_kxs72()	get_kxs72()	ホストからのコマンドリモートセッティングでの応答メッセージ	205
-----	-----	-----	-----
set_kxs74()	get_kxs74()	KXBコマンドによる無条件送信での送信方法	206
set_kxs75()	get_kxs75()	KXBコマンドメッセージのグローバルGRAM自動的変換	207
-----	-----	-----	-----
set_kxs77()	get_kxs77()	バイトモード時、ホストからのメッセージ本体のみを送信	208
set_kxs78()	get_kxs78()	バイトモード時、インバウンドキューが満杯メッセージを送信	209
set_kxs79()	get_kxs79()	KXBコマンドの検知時間	210
set_kxs80()	get_kxs80()	アウトバウンドグローバルGRAMパケットのフォーマット	211
set_kxs81()	get_kxs81()	アウトバウンドメッセージ重複受信防止ガードタイム	212
set_kxs82()	get_kxs82()	アウトバウンドグローバルGRAM重複受信防止ガードタイム	213
set_kxs83()	get_kxs83()	デジタル出力ポートのデフォルト値	214
set_kxs84()	get_kxs84()	ローミングモード	215
set_kxs85()	get_kxs85()	ローミング対象GCC	216
set_kxs86()	get_kxs86()	KXBコマンド動作の曜日指定	217
set_kxs87()	get_kxs87()	GPS測位精度	219
set_kxs90()	get_kxs90()	アナログ入力のAD変換解像度と入力モード	220
set_kxs91()	get_kxs91()	オーブコム衛星の軌道情報のDTEへの出力	221
set_kxs94()	get_kxs94()	デジタル入出力ポートの入出力モード	222
set_kxs95()	get_kxs95()	プロトコルモードのパケットタイプ	223
set_kxs96()	get_kxs96()	オーブコムシリアルリンク通信モードのサポートRS232C選択	224
-----	-----	-----	-----
set_kxd01()	get_kxd01()	デジタルポート出力	225
-----	get_kxd02()	アナログ入力値	226
set_kxm01()	get_kxm01()	固定メッセージ	227
-----	-----	-----	-----
set_kxb01()	get_kxb01()	時刻指定送信 (KXB01コマンド)	228
set_kxb02()	get_kxb02()	インターバル送信 (KXB02コマンド)	229
-----	-----	-----	-----
set_kxb05()	get_kxb05()	I/O状態変化報知	230
set_kxb00()	-----	KXB01 - 05コマンドの設定を解除	231

## 6. 各関数の詳細説明

## set\_desired\_ncc\_id

---

要約 void set\_desired\_ncc\_id ( unsigned char ncc\_id )

unsigned char ncc\_id : GCC 番号( 0 - 255)

解説 端末が登録されているGCC番号を設定する

戻値 なし

補足 マウントモードの KXS01 マウントでも変更できる。

## set\_pin\_code

---

要約           int set\_pin\_code( unsigned long pin\_code );

                unsigned long pin\_code:   ピンコード ( 0 - 9999 )

解説           端末 - ゲートウェイ間通信のパスワードを設定する

戻値           0: 設定成功  
                1: 設定失敗

補足           マウントモードの KXS50 マウントでも変更できる。  
                ピンコードはゲートウェイと端末に設定する。ゲートウェイがピンコード参照モードになっている時は、これら2つのピンコードが合っていないと通信できない。

## set\_ncc\_search\_mode

要約	<pre>int set_ncc_search_mode( int ncc_search_mode );</pre> <p>int ncc_search_mode : サーチモード ( 0 - 4 )</p> <p>0=desired_nccを連続的にダウリンクバンドの中から検索する</p> <p>1=desired_nccを1回探す。もし見つからなければ最初に発見したダウリンクとのロックを維持する。</p> <p>2=最初に発見したダウリンクとのロックを維持する。</p> <p>3=desired_nccを1回探す。もし見つからなければ任意のGCCを含むものの検索を開始、もしなにもなければ、最初に発見したダウリンクとのロックを維持する。</p> <p>4=desired_nccを1回探す。もし見つからなければグローバル衛星かdesired_nccのダウリンクを検索し続ける。</p>
解説	<p>GCCサーチモードを設定する。</p>
戻値	<p>0: 設定成功</p> <p>1: 設定失敗</p>
補足	<p>コマンドモードの KXS14 コマンドでも設定できる。</p>

## all\_msg\_polling

要約	int all_msg_polling( unsigned char ncc_id );  unsigned char ncc_id: GCC 番号
解説	GCC に格納してあるすべての自分宛てのメッセージ / コマンド を要求する
戻値	0: コマンド 正常受信。ポーリング 開始 1: 衛星を捕捉していない等でポーリング できない
補足	この関数は、GCC に格納してあるすべての自分宛てのメッセージ / コマンド を要求するものであり、GCC がこのコマンド を受信するとメッセージ / コマンド の送信を開始する。端末が受信するとアトバウンド キューに格納される。  ポーリング は、端末が GCC とリンクしている衛星を捕捉している時に要求して下さい。

< 例 >

```
#include "kme_lib.h"

void main(void)
{
    while (get_sat_no() == 0) go_wait(10, SEC_UT);

    if (get_ncc_quan() > 0) { /* Check no globalgram */
        if (all_msg_polling(get_kxs01()) == 0) {
            /* SCがポーリング 処理を始めると"st_stauts"が */
            /* 4に変わり、終わると0に戻る */
            while (get_st_status() == 4/* polling */);

            if (chk_failed_polling() == 119) {
                /* ORBCOMM Gatewayには、メッセージ はなかった。 */
            }
        }
    }
}
```

## s\_msg\_polling

要約	int s_msg_polling ( unsigned char ncc_id );  unsigned char ncc_id: GCC 番号
解説	GCC に格納してあるすべての自分宛てのメッセージ (150 バイト以下) / コマンドを要求する。
戻値	0: コマンド 正常受信。ホッピング 開始 1: 衛星を捕捉していない等でホッピング できない
補足	この関数は、GCC に格納してあるすべての自分宛てのメッセージ (150 バイト以下) / コマンドを要求するものであり、GCC がこのコマンドを受信するとメッセージ / コマンドの送信を開始する。端末が受信するとアウトバウンドキューに格納される。  ホッピング は、端末が GCC とリンクしている衛星を捕捉している時に要求して下さい。

< 例 >

```
#include "kme_lib.h"

void main(void)
{
    while (get_sat_no() == 0) go_wait(10, SEC_UT);

    if (get_ncc_quan() > 0) { /* Check no globalgram */
        if (s_msg_polling(get_kxs01()) == 0) {
            /* SCがホッピング 処理を始めると"st_stauts"が */
            /* 4に変わり、終わると0に戻る */
            while (get_st_status() == 4/* polling */);

            if (chk_failed_polling() == 120) {
                /* ORBCOMM Gateway には、150バイト以下のメッセージ */
                /* は、なかった。 */
            }
        }
    }
}
```

## globalgram\_polling

要約	int globalgram_polling ( unsigned char ncc_id );  unsigned char ncc_id: GCC番号
解説	衛星に格納してあるすべての自分宛てのグローバルグラムを要求する。
戻値	0: コマンド正常受信。ポーリング開始 1: 衛星を捕捉していない等でポーリングができない
補足	この関数は、衛星に格納してあるすべての自分宛てのグローバルグラムを要求するものであり、衛星がこのコマンドを受信するとグローバルグラムの送信を開始する。端末が受信するとアウトバウンドキューに格納される。  ポーリングは、端末がグローバルグラム衛星を捕捉している時に要求して下さい。

<例>

```
#include "kme_lib.h"

void main(void)
{
    while (get_sat_no() == 0) go_wait(10, SEC_UT);

    if (get_ncc_quan() == 0) { /* Check globalgram */
        if (globalgram_polling(get_kxs01()) == 0) {
            /* SCがポーリング処理を始めると"st_stauts"が */
            /* 6に変わり、終わると0に戻る */
            while (get_st_status() == 6/* polling */);

            if (chk_failed_polling() == 121) {
                /*衛星には、グローバルメッセージがなかった。 */
            }
        }
    }
}
```



## chk\_failed\_polling

要約           int chk\_failed\_polling( void );

解説           端末がホップリング失敗した時の原因を獲得する

戻値           ダイアグノースティック

- 1 = まだ結果が出ていない
- 0 = 認識されない発信者 / 受信者の名前
- 1 = あいまいな発信者 / 受信者の名前
- 2 = X.400 MTA の輻輳
- 3 = ループ検知
- 4 = 受領者が利用できない
- 5 = 転送タイムアウト
- 6 = メッセージタイプがサポートされていない
- 7 = 中身が長すぎる
- 8 = 実際的でない変換
- 9 = 禁止された変換
- 10 = 変換が登録されていない
- 11 = 無効パラメータ
- 12-100 = システム予約
- 101 = 加入者端末IDが登録されていない
- 102 = PINコードが無効
- 103 = 要求されたGCCが衛星ダイアグノースティックの中に見つからない。
- 104 = 不十分なメッセージ優先度(GCCが輻輳中かもしれない)
- 105 = 衛星が応答しない(アップリンクが輻輳中かも知れない)
- 106 = SCのアクセス制限
- 107 = 加入者端末の登録期限終了
- 108 = インバンドメッセージが既にGCC内に存在する
- 109 = インバンドメッセージ番号エラー
- 110 = 不揮発性メモリにメッセージをセーブしている時に、GCC内にエラーが生じた
- 111 = GCC内のデータベースにエラーが生じた
- 112 = 追加の診断インフォメーションがない
- 113 = インバンド転送の再送回数が最大値を越えた
- 114 = グローバルブロード送信は禁止されている
- 115 = 衛星が見つからない
- 116 = ポジションレポートが現在有効でないが、計算を開始している。
- 117 = 測位機能がない。
- 118 = グローバルブロードのサイズ超過
- 119 = GCC内にアウトバンドメッセージ/コマンドがない。
- 120 = GCC内に150バイト以下のメッセージ/コマンドがない。
- 121 = 衛星内にグローバルブロードがない。
- 122 = 要求されたメッセージは削除された。
- 123 = 軌道要素をストアしていない。
- 124 = レジストレーションの要求は受信された。しばらく待て。
- 125 = レジストレーションの要求は認められた。
- 126 = レジストレーションの要求は却下された。
- 127 = グローバルブロードの最大値(16)まで現在の衛星に格納されている。

補足

## chk\_ib\_tx

### 要約

```
int chk_ib_tx( unsigned char mha_ref_num,
              unsigned char *status )
```

unsigned char mha\_ref\_num: メッセージ識別番号

unsigned char \*status : メッセージ転送状態

- 0 = 転送失敗
- 1 = 転送不能
- 2 = 変換されなかった
- 3 = システム予約
- 4 = システム予約
- 5 = システム予約
- 6 = システム予約
- 7 = システム予約
- 8 = システム予約
- 9 = システム予約
- 10 = 送信失敗したが、メッセージはキューイングされ、再び試みられる。
- 11 = Communication Commandの応答
- 12 = statusが未知
- 13 = 今は受領者からACKがない - しばらくお待ちください
- 14 = ユーザーにより中断されたメッセージ
- 15 = 示されたACKの発信者によりメッセージが受信された

### 解説

メッセージ番号指定のメッセージ転送状態を要求する

### 戻値

- 1 = 送信成功
- 0 = 認識されない発信者 / 受信者の名前
- 1 = あいまいな発信者 / 受信者の名前
- 2 = X.400 MTA の輻輳
- 3 = ループ検知
- 4 = 受領者が利用できない
- 5 = 転送タイムアウト
- 6 = メッセージタイプがサポートされていない
- 7 = 中身が長すぎる
- 8 = 実際的でない変換
- 9 = 禁止された変換
- 10 = 変換が登録されていない
- 11 = 無効パラメータ
- 12-100 = システム予約
- 101 = 加入者端末IDが登録されていない
- 102 = PINコードが無効
- 103 = 要求されたGCCが衛星ネットワークの中に見つからない。
- 104 = 不十分なメッセージ優先度(NCCが輻輳中かもしれない)
- 105 = 衛星が応答しない(アップリンクが輻輳中かも知れない)
- 106 = SCのアクセス制限
- 107 = 加入者端末の登録期限終了
- 108 = インバウンドメッセージが既にGCC内に存在する
- 109 = インバウンドメッセージ番号エラー
- 110 = 不揮発性メモリにメッセージをセーブしている時に、GCC内にエラーが生じた
- 111 = GCC内のデータバスにエラーが生じた

- 112 = 追加の診断情報が無い
- 113 = インパウンド転送の再送回数が最大値を越えた
- 114 = グローバルログ送信は禁止されている
- 115 = 衛星が見つからない
- 116 = ポジションポートが現在有効でないが、計算を開始している。
- 117 = 測位機能がない。
- 118 = グローバルログのサイズ超過
- 119 = GCC内にアウトバウンドメッセージ/コマンドがない。
- 120 = GCC内に150バイト以下のメッセージ/コマンドがない。
- 121 = 衛星内にグローバルログがない。
- 122 = 要求されたメッセージは削除された。
- 123 = 軌道要素をストアしていない。
- 124 = レジストレーションの要求は受信された。しばらく待て。
- 125 = レジストレーションの要求は認められた。
- 126 = レジストレーションの要求は却下された。
- 127 = グローバルログの最大値(16)まで現在の衛星に格納されている。

補足

## clear\_active\_msg

---

要約	int clear_active_msg(void);
解説	メッセージ送信・受信（端末 - GCC 間）をリセットする
戻値	NULL : リセット要求の発行成功 それ以外: 通信中ではない
補足	リセット要求の発行が成功しても、タイミングによってはメッセージがホストに配信されることがあります。

## clear\_mha\_ref\_num\_msg

要約	<pre>ib_id clear_mha_ref_num_msg( unsigned char mha_ref_num );</pre> <p>unsigned char mha_ref_num: メッセージ識別番号</p>
解説	メッセージ識別番号により識別されるインバウンドメッセージを削除する
戻値	<p>NULL: メッセージ識別番号により識別されるインバウンドメッセージが、インバウンドキューにない</p> <p>それ以外: 削除成功、あるいはそのメッセージは現在送信中。 送信中メッセージに関してはアホート要求が発行される。</p>
補足	送信中メッセージに関してはアホート要求の発行されるが、タイミングによってはメッセージがホストに配信されることがあります。

## clear\_all\_ib\_q

---

要約	ib_id clear_all_ib_q( void )
解説	送信中メッセージ以外のインバウンドキュー中の全メッセージをクリア。
戻値	non-NULL: 送信中メッセージ（送信中メッセージ以外は削除された） NULL : インバウンドキューの全てのメッセージがクリアされた。
補足	送信中メッセージのクリアは、“clear_active_msg ( )” 関数コールする

## clear\_all\_ob\_q

---

要約           ob\_id clear\_all\_ob\_q( void );

解説           受信中メッセージを除いたアウトバウンドキュー内の全てのメッセージをクリア

戻値           non-NULL : 受信中メッセージ（受信中メッセージ以外は削除された）  
              NULL     : 全てのメッセージが削除された

補足





## req\_select\_next\_downlink

---

要約	<code>void req_select_next_downlink( void );</code>
解説	システム内の次のダウンリンクを強制受信
戻値	なし
補足	端末内に記憶しているダウンリンクチャネルリストの中で、現在受信しているチャネルの次のチャネルの受信を試みる。

## chk\_polling\_event

---

要約	int chk_polling_event( void )
解説	GCC からのポインタがあるかをチェックする
戻値	1以外: なし 1 : あり
補足	この関数は、KXS68=0 の時無効です。

## get\_polling\_info

---

要約 void get\_polling\_info( unsigned char \*poll\_obj,  
unsigned char \*ncc\_id,  
unsigned char \*or\_ind )

unsigned char \*poll\_obj

- 0: 1メッセージ
- 1: 1メッセージあるいはポート
- 2: 1ポート
- 3: 1測位ポート

unsigned char \*ncc\_id : GCC番号

unsigned char \*or\_ind : O/R インディケータ

解説 GCCからのポーリング内容を獲得する

戻値 なし

補足 この関数は、KXS68=0の時無効です。

## req\_send\_ib\_message

要約

```
int req_send_ib_message( unsigned char ncc_id,
                        unsigned char polled,
                        unsigned char ack_level,
                        unsigned char priority,
                        unsigned char msg_body_type,
                        unsigned char msg_body_sub_type,
                        unsigned char mha_ref_num,
                        unsigned char *rcpnt[7],
                        unsigned char *subject,
                        unsigned char *message,
                        unsigned int message_len)
```

ncc\_id : GCC 番号  
 polled : ポールト  
 ack\_level : ACK レベル  
 priority : プライオリティ  
 msg\_body\_type : メッセージ ボディ タイプ  
 msg\_body\_sub\_type : メッセージ ボディ サブ タイプ  
 mha\_ref\_num : メッセージ 識別番号  
 rcpnt[7] : O/R インディケータ、O/R アドレス  
 subject : サブジェクト  
 message : メッセージ 本体  
 message\_len : メッセージ 長

解説                   インバウンドメッセージをインバウンドキューに設定する

戻値                   0: インバウンドキューに設定完了  
                       1: インバウンドキューに設定失敗。メッセージ長が長すぎてキューに入らない。  
                       2: インバウンドキューに設定失敗。メモリが獲得できない。

補足                   メッセージ ボディ サブ タイプは、メッセージ ボディ タイプが 0、15 の時だけ有効。  
                       サブジェクトがない時は、NULL を設定すること。

<例>

```
#include <stdio.h>
#include <string.h>
#include "kmlib.h"

void main(void)
{
    unsigned char    subject[20];
    unsigned char    *recipient[7], recipient0[10];
    unsigned char    message[30];
    unsigned char    mha_ref_num;

    /* 受信者を設定 */
```

```

recipient0[0] = 0x01;
recipient[0] = recipient0;
recipient[1] = (unsigned char *)0;
recipient[2] = (unsigned char *)0;
recipient[3] = (unsigned char *)0;
recipient[4] = (unsigned char *)0;
recipient[5] = (unsigned char *)0;
recipient[6] = (unsigned char *)0;

/* メッセージ作成 */
strcpy(message, "HELLO");

/* メッセージ識別番号設定 */
mha_ref_num = 0;

/* 送信メッセージをインバウンドキューに設定 */
subject[0] = '¥0';
if (req_send_ib_message(get_kxs01(), /* GCC 番号 */
                        0, /* すぐに送信 */
                        get_kxs06(), /* ACKレベル */
                        get_kxs03(), /* プライオリティ */
                        14, /* メッセージホデタイプ : ハイリ */
                        0, /* メッセージホデタイプ (タミ) */
                        mha_ref_num, /* メッセージ識別番号 */
                        recipient, /* 受信者 */
                        subject, /* サブジェクトなし */
                        message, /* メッセージ本体 */
                        strlen(message) /* メッセージ長 */
                        == MESSAGE_QUEUED) { /* 設定成功? */
    ; /* 設定成功 */
} else {
    ; /* 設定失敗 */
}
exit_user_apl(); /* ユーザーアプリケーション終了 */
}

```

## req\_send\_ib\_report

要約

```
int req_send_ib_report(unsigned char ncc_id,
                      unsigned char polled,
                      unsigned char serv_type,
                      unsigned char or_ind,
                      unsigned char mha_ref_num,
                      unsigned char *user_data )
```

ncc\_id : GCC 番号  
 polled : ポールト  
 serv\_type : サービスタイプ  
 mha\_ref\_num : 識別番号  
 user\_data[6] : レポート本体

解説

インバウンドレポートをインバウンドキューに設定する

戻値

0: インバウンドキューに設定完了  
 2: インバウンドキューに設定失敗。メモリが獲得できない。

補足

user\_data (レポート本体) のバイト数は6バイト。

&lt;例&gt;

```

#include "kme_lib.h"

void main(void)
{
    unsigned char user_data[6];
    unsigned char mha_ref_num;

    /* ホット本体設定 */
    user_data[0] = 'H';
    user_data[1] = 'E';
    user_data[2] = 'L';
    user_data[3] = 'L';
    user_data[4] = 'O';
    user_data[5] = '!';

    /* 識別番号設定 */
    mha_ref_num = 0;

    /* インポートをインポートキーに設定する */
    if (req_send_ib_report( get_kxs01(), /* GCC番号 */
                          0, /* すぐに送信 */
                          get_kxs08(), /* サブスタブ */
                          get_kxs04(), /* O/Rインデクター */
                          mha_ref_num, /* 識別番号 */
                          user_data ) /* メッセージ本体 */
        == MESSAGE_QUEUED) { /* 設定成功? */
        ; /* 設定成功 */
    } else {
        ; /* 設定失敗 */
    }
    exit_user_apl(); /* ユーザーアプリケーション終了 */
}

```

req\_send\_ib\_globalgram

要約

```
int req_send_ib_globalgram(unsigned char ncc_id,
                            unsigned char mha_ref_num,
                            unsigned char or_ind,
                            unsigned char *user_data,
                            unsigned int user_data_len)
```

ncc\_id : GCC 番号  
mha\_ref\_num : 識別番号  
or\_ind : O/R インデキータ  
user\_data : グローバルグラム本体  
user\_data\_len : グローバルグラム長  
インバウンドグローバルグラムをインバウンドキューに設定する

解説

戻値

- 0: インバウンドキューに設定完了
- 1: インバウンドキューに設定失敗。長すぎてキューに入らない。
- 2: インバウンドキューに設定失敗。メモリが獲得できない。

補足

user\_data (グローバルグラム本体) のバイト数は229バイト以下。



## req\_send\_ib\_enh\_globalgram

要約 Int req\_send\_ib\_enh\_globalgram( unsigned char NCC\_id,  
 unsigned char mha\_ref\_num,  
 unsigned char priority,  
 unsigned char msg\_body\_type,  
 unsigned char msg\_body\_sub\_type,  
 unsigned char \*rcpnt[7],  
 unsigned char \*subject,  
 unsigned char \*message,  
 unsigned int message\_len)

ncc\_id : GCC 番号

mha\_ref\_num : 識別番号

priority : プライオリティ

msg\_body\_type : メッセージボディタイプ

msg\_body\_sub\_type: メッセージボディサブタイプ

rcpnt[7] : O/R インデキータ、O/R アドレス

subject : サブジェクト

message : メッセージ本体

message\_len : メッセージ長

解説 インバウンドエンハンスドグローバルグラムをインバウンドキューに設定する

戻値 0: インバウンドキューに設定完了  
 1: インバウンドキューに設定失敗。長すぎてキューに入らない。  
 2: インバウンドキューに設定失敗。メモリが獲得できない。

補足 メッセージボディサブタイプは、メッセージボディタイプが 0、15 の時だけ有効。  
 サブジェクトがない時は、NULLを設定すること。

## req\_send\_ib\_pos\_report

---

要約           int req\_send\_ib\_pos\_report(unsigned char ncc\_id,  
  unsigned char polled,  
  unsigned char serv\_type,  
  unsigned char or\_ind,  
  unsigned char mha\_ref\_num,  
  unsigned char \*lat  
  unsigned char \*lon )

ncc\_id : GCC 番号

polled : ホールト

serv\_type : サービスタイプ

mha\_ref\_num : 識別番号

\*lat: 緯度 - 0: 北極, 0xffffffff: 南極

\*lon: 経度 - 0: グリニッジ子午線, 東に向かって増加

### 解説

インバウンド測位レポートをインバウンドキューに設定する

### 戻値

0: インバウンドキューに設定完了

2: インバウンドキューに設定失敗。メモリーが獲得できない。

### 補足

lat/lonは3バイト。

## req\_send\_ob\_message

```

要約      int req_send_ob_message(unsigned char ncc_id,
                                unsigned char msg_body_type,
                                unsigned char msg_body_sub_type,
                                unsigned char *rcpnt[7],
                                unsigned char *subject,
                                unsigned char *message,
                                unsigned int message_len)

```

ncc\_id : GCC 番号  
msg\_body\_type : メッセージボディタイプ  
msg\_body\_sub\_type : メッセージボディサブタイプ  
rcpnt[7] : O/R インディケータ、O/R アドレス  
subject : サブジェクト  
message : メッセージ本体  
message\_len : メッセージ長

解説 アウトバウンドメッセージをアウトバウンドキューに設定する

戻値

- 0: アウトバウンドキューに設定完了
- 1: アウトバウンドキューに設定失敗。長すぎてキューに入らない。
- 2: アウトバウンドキューに設定失敗。メモリが獲得できない。

補足 メッセージボディサブタイプは、メッセージボディタイプが 0、15 の時だけ有効。  
サブジェクトがない時は、NULL を設定すること。

&lt;例&gt;

```

void main(void)
{
    unsigned char    subject[20];
    unsigned char    *recipient[7], recipient0[10];
    unsigned char    message[30];

    /* 受信者を設定 */
    recipient0[0] = 0x01;
    recipient[0] = recipient0;
    recipient[1] = (unsigned char *)0;
    recipient[2] = (unsigned char *)0;
    recipient[3] = (unsigned char *)0;
    recipient[4] = (unsigned char *)0;
    recipient[5] = (unsigned char *)0;
    recipient[6] = (unsigned char *)0;

    /* メッセージ本体を設定 */
    message[0] = 'H';
    message[1] = 'E';
    message[2] = 'L';
    message[3] = 'L';
    message[4] = 'O';
    message[5] = '¥0';

    /* メッセージをアウトバウンドキューに設定する */
    subject[0] = '¥0';
    if (req_send_ob_message( get_kxs01(),      /* GCC番号 */
                            0,                /* メッセージホテタイプ : TEXT */
                            5,                /* メッセージホテサブタイプ */
                            recipient,        /* 受信者 */
                            subject,          /* サブジェクト */
                            message,         /* メッセージ本体 */
                            strlen(message) ) /* メッセージ長 */
        == MESSAGE_QUEUED) { /* 設定成功? */
        ; /* 設定成功 */
    } else {
        ; /* 設定失敗 */
    }
    exit_user_apl(); /* ユーザーアプリケーション終了 */
}

```

## get\_unget\_ob\_id\_for\_user

---

要約                   ob\_id get\_unget\_ob\_id\_for\_user( void);

解説                   アウトバウンド ID を獲得する

戻値                   non-NULL: アウトバウンド ID  
NULL.                 : アウトバウンド キューにデータがない

補足                   アウトバウンド ID は、アウトバウンド キューにあるデータのポインタ。

本関数をコールするとアウトバウンド キューを送信中状態とします。本関数を使用した後は必ず remove\_ob\_q() をコールして送信中状態とキューの解除をしてください。

<例>

```
#include "kme_lib.h"

void main(void)
{
    ob_id            get_ob_id;
    unsigned char    ncc_id;
    unsigned char    subject_ind;
    unsigned char    msg_body_type, msg_body_sub_type;
    unsigned char    data, or_quan, or_ind;
    unsigned int     msg_len, index, si;
    unsigned char    message[230], subject[81];
    unsigned char    ref_num;

    /* アウトバウンド キューにデータがあるかをチェック */
    while ((get_ob_id = get_unget_ob_id_for_user()) != 0) {
        switch (get_ob_type(get_ob_id)) {
            case OB_MSG: /* アウトバウンド メッセージ */
                /* 1. GCC番号                                                 */
                /* 2. サブジェクト                                             */
                /* 3. メッセージポインタ                                     */
                /* 4. 受信者 + 送信者                                         */
                /* 5. メッセージ長                                             */
                /* 6. メッセージ                                                 */
                ncc_id = get_ob_ncc_id(get_ob_id);
                subject_ind = get_ob_msg_subject_ind( get_ob_id );

                msg_body_type = get_ob_msg_msg_body_type( get_ob_id );
                or_quan = get_ob_msg_or_quan( get_ob_id );
                msg_len = get_ob_msg_msg_len( get_ob_id );
                si = 0;
            }
    }
}
```

```

while ( or_quan ) {
    data = get_ob_message( get_ob_id, si++ );
    switch ( data & 0xf0 ) {
    case 0x00 :
    case 0x80 :
    case 0x90 :
        or_quan--;
        break;
    default :
        break;
    }
}

if (subject_ind == 1) {
    for (index = 0; (data = get_ob_message( get_ob_id, si++ )) != 0; ) {
        if (index < sizeof(subject) -1 ) {
            subject[index++] = data;
        }
    }
    subject[index] = '\0';
}
if ((msg_body_type == 0) || (msg_body_type == 15)) {
    msg_body_sub_type = get_ob_message( get_ob_id, si++ );
}

for (index = 0; si < msg_len; index++) {
    message[index] = get_ob_message( get_ob_id, si++ );
    if (index >= sizeof(message) - 1) break;
}
message[index] = '\0';
break;

case OB_USER : /* アウトバウンド ユーザー-コマンド */
    /* 1. GCC番号 */
    /* 2. テータ本体 (5 bytes) */
    ncc_id = get_ob_ncc_id(get_ob_id);
    msg_len = 5;
    for (index = 0; index < 5; index++) {
        message[index] = get_ob_user_command( get_ob_id, index );
    }
    message[5] = '\0';
    break;

case OB_GGRAM : /* アウトバウンド グローバルグラム */
    /* 1. GCC番号 */
    /* 2. メッセージホテタイ */

```

```
/* 3. 識別番号 */
/* 4. メッセージ長 */
/* 5. グローバルグラム本体 */
ncc_id = get_ob_ncc_id(get_ob_id);
ref_num = get_ob_globalgram_ref_num( get_ob_id );
or_ind = get_ob_globalgram_or_ind( get_ob_id );
msg_len = get_ob_globalgram_msg_len( get_ob_id );

for (index = 0; index < msg_len; index++) {
    message[index] = get_ob_globalgram( get_ob_id, index);
    if (index >= sizeof(message) - 1) break;
}
message[index] = '¥0';
break;

default :
    index = msg_len = 0;
    break;
}
remove_ob_q(get_ob_id);
}
exit_user_apl();
}
```

## remove\_ob\_q

---

要約	void remove_ob_q( ob_id rm_ob_id)  ob_id rm_ob_id : アウトバウンド ID
解説	アウトバウンドキューからアウトバウンド ID 指定のメッセージを削除する
戻値	なし
補足	メッセージをキューから削除するのに 2 重削除を行わないでください。 削除したメッセージに対して再度この関数をコールするとプログラムが正常に動作しなくなります。



## get\_ob\_type

---

要約	unsigned char get_ob_type( ob_id get_ob_id );  ob_id get_ob_id : アウトバウンド ID
解説	アウトバウンド キーにあるデータの種別を取得
戻値	1: アウトバウンド メッセージ 2: アウトバウンド コメント 3: アウトバウンド グローバル グラム
補足	アウトバウンド ID は、関数“get_unget_ob_id_for_user”により獲得する

## get\_ob\_ncc\_id

---

要約	unsigned char get_ob_ncc_id( ob_id get_ob_id );  ob_id get_ob_id : アウト ID
解説	アウト ID 指定のデータの GCC 番号を取得する
戻値	GCC 番号 (0 - 255)
補足	アウト ID は、関数“get_unget_ob_id_for_user”により獲得する

## get\_ob\_msg\_subject\_ind

---

要約	unsigned char get_ob_msg_subject_ind( ob_id get_ob_id);  ob_id get_ob_id : アウトバウンド ID
解説	サブジェクトの有無を獲得する。
戻値	0: サブジェクト無し 1: サブジェクト有り
補足	アウトバウンド ID は、関数“get_unget_ob_id_for_user”により獲得する

## get\_ob\_msg\_msg\_body\_type

---

要約            unsigned char get\_ob\_msg\_msg\_body\_type( ob\_id get\_ob\_id);

ob\_id get\_ob\_id : アウト ID

解説            アウト ID 指定のメッセージのメッセージボディタイプを取得する

戻値            メッセージボディタイプ

補足            アウト ID は、関数“get\_unget\_ob\_id\_for\_user”により獲得する

## get\_ob\_msg\_or\_quan

要約	<code>unsigned char get_ob_msg_or_quan( ob_id get_ob_id);</code>  ob_id get_ob_id : アウトバウンド ID
解説	アウトバウンド ID 指定のメッセージ受信者の数を獲得する
戻値	受信者の数
補足	アウトバウンド ID は、関数“get_unget_ob_id_for_user”により獲得する。

## get\_ob\_msg\_msg\_len

---

要約	<code>unsigned int get_ob_msg_msg_len( ob_id get_ob_id);</code>  <code>ob_id get_ob_id</code> : アウトバウンド ID
解説	アウトバウンド ID 指定のメッセージのメッセージ長を取得する
戻値	メッセージ長
補足	アウトバウンド ID は、関数“get_unget_ob_id_for_user”により獲得する



## get\_ob\_message

要約	<pre>unsigned char get_ob_message ( ob_id get_ob_id,                                unsigned int msg_index);</pre> <p>ob_id get_ob_id : アウトバウンド ID                  unsigned int msg_index : インデックス</p>
解説	アウトバウンド ID 指定のメッセージのメッセージ本体を取得する
戻値	メッセージ本体の 1 データ ( 0x00 - 0xff )
補足	アウトバウンド ID は、関数“get_unget_ob_id_for_user”により獲得する



## get\_ob\_user\_command

要約	<pre>unsigned char get_ob_user_command( ob_id get_ob_id                                    unsigned int data_index);</pre> <p>ob_id get_ob_id : アウト ID data_index : インデックス</p>
解説	アウト ID 指定のコマンドの本体を取得する
戻値	コマンド本体の 1 データ
補足	アウト ID は、関数“get_unget_ob_id_for_user”により獲得する

## get\_ob\_globalgram\_ref\_num

---

要約	unsigned char get_ob_globalgram_ref_num( ob_id get_ob_id);  ob_id get_ob_id : オブジェクト ID
解説	オブジェクト ID 指定のグローバルグラムの識別番号を取得する
戻値	識別番号
補足	オブジェクト ID は、関数“get_unget_ob_id_for_user”により獲得する

## get\_ob\_globalgram\_or\_ind

要約                    unsigned char get\_ob\_globalgram\_or\_ind( ob\_id get\_ob\_id);

ob\_id get\_ob\_id : オブジェクト ID

解説                    オブジェクト ID 指定のグローバルグラムの O/R インデキータを獲得する

戻値                    O/R インデキータ

補足                    オブジェクト ID は、関数“get\_unget\_ob\_id\_for\_user”により獲得する

## get\_ob\_globalgram\_msg\_len

要約            unsigned int get\_ob\_globalgram\_msg\_len( ob\_id get\_ob\_ib);

ob\_id get\_ob\_id : オブジェクト ID

解説            オブジェクト ID 指定のグローバルグラムのデータ長を取得する

戻値            データ長

補足            オブジェクト ID は、関数“get\_unget\_ob\_id\_for\_user”により獲得する

## get\_ob\_globalgram

---

要約	<pre>unsigned char get_ob_globalgram( ob_id get_ob_id,                                 unsigned int data_index);</pre> <p>ob_id get_ob_id : オブジェクト ID                  unsigned int data_index: インデックス</p>
解説	オブジェクト ID 指定のグローバルグラムの本体を獲得する
戻値	グローバルグラム本体の 1 バイト ( 0x00 - 0xff)
補足	オブジェクト ID は、関数“get_unget_ob_id_for_user”により獲得する

## go\_sleep\_time

---

要約           void go\_sleep\_time( unsigned long int sleep\_time );

                  unsigned long int sleep\_time :時間 ( 0 - 4294967295[秒])

解説           一定時間ハワタウさせる

戻値           なし

補足           スリープモードがオ(KXS37=0)の時でも、有効です

## go\_wait

要約	<pre>void go_wait( unsigned long time, char time_unit );</pre> <p>unsigned long time: スリープ時間  char time_unit : 単位  0: ミリセカント  1: 秒  2: 分</p>
解説	指定時間サスペンドする。
戻値	なし
補足	これは、ユーザーアプリケーションの実行権を他タスクに渡す関数であり、パワーダウンではありません。

## rf\_block\_power\_on

---

要約           void rf\_block\_power\_on( void );

解説           無線系の受信部分の電源を入れる。

戻値           なし。

補足



## rf\_block\_power\_off

---

要約           void rf\_block\_power\_off( void );

解説           無線系の受信部分の電源を切る。

戻値           なし。

補足

## check\_rf\_block\_power

---

要約           int check\_rf\_block\_power( void );

解説           無線系の受信部分の電源状態を獲得する

戻値           0: 電源ON  
              1: 電源OFF

補足

## chk\_rx\_buff

---

要約                    unsigned int chk\_rx\_buff( void );

解説                    メインポートの RS232C インターフェイスから受信されたデータ数を獲得する

戻値                    受信データ数

補足

## get\_rx\_data

---

要約	int get_rx_data( void );
解説	メインポートの RS232C インターフェースから受信されたデータを 1 バイト取り出す
戻値	0x00-0xff : 受信データ -1 : 受信エラー
補足	RS232C の受信バッファサイズは 4k バイトです。 サブポートは get_rx_data_port 関数を使用してください。

## chk\_tx\_ready

---

要約	int chk_tx_ready( void );
解説	メインポートの RS232C の送信バッファ ( 256 バイト ) が満杯かをチェックする
戻値	0: 送信バッファが満杯。 1: 送信バッファは満杯ではない。
補足	送信バッファが満杯の時には、送信データをセツできません。 送信バッファに空きができてから、送信データをセツしてください。 サブポートは chk_tx_ready_port 関数を使用してください。

## set\_tx\_data

---

要約	unsigned int set_tx_data( unsigned char tx_data );  unsigned char tx_data : 送信データ ( 0x00 - 0xff )
解説	メインポートの RS232C への送信データを設定する
戻値	インデックス。送信バッファ先頭からの番号
補足	RS232C の送信バッファサイズはメイン、サブ各 512 バイトです。 この関数には送信バッファが満杯かのチェックが含まれていませんので、使用する場合には送信バッファ状態チェック関数 chk_tx_ready() と共に使用して下さい。サブポートは set_tx_data_port 関数を使用して下さい。

<例>

```
void tx_string_to_DTE(char *str)
{
    char *pStr;

    for (pStr = str; *pStr != '\0'); {
        while (chk_tx_ready() != 1);          /* 送信バッファは満杯か ? */
        set_tx_data((unsigned char)(*pStr)); /* 1文字送信 */
        pStr++;
    }
}
```

## cts\_act

---

要約           void cts\_act( void )

解説           メインポートの CTS 信号をアクティブにする

戻値           なし

補足

## cts\_neg

---

要約            void cts\_neg( void )

解説           メインポートの CTS 信号を無効にする。

戻値           なし

補足



## chk\_cd

---

要約           int   chk\_cd( void )

解説           メインポートの CD 信号の状態をチェックする.

戻値           0: ネガティブ  
              1: アクティブ

補足

## chk\_rts

---

要約           int   chk\_rts(void)

解説           メインポートの RTS 信号の状態をチェックする。

戻値           0: ネガティブ  
              1: アクティブ

補足

## chk\_rx\_buff\_port

要約            unsigned int chk\_rx\_buff\_port ( unsigned char port );

unsigned char port : port

0: シリアルポート1 (メイン)

1: シリアルポート2 (サブ)

解説            RS232C インターフェイスから受信されたデータ数を獲得する

戻値            受信データ数

補足            受信バッファのサイズ

0: シリアルポート1 (メイン)    4k byte

1: シリアルポート2 (サブ)    512 byte

## get\_rx\_data\_port

---

要約	int get_rx_data_port ( unsigned char port);  unsigned char port : port 0: シリアルポート1 (メイン) 1: シリアルポート2 (サブ)
解説	RS232C インターフェイスから受信されたデータを1バイト取り出す
戻値	0x00-0xff : 受信データ -1 : 受信エラー
補足	RS232C の受信バッファサイズ 0: シリアルポート1 (メイン) 4k byte 1: シリアルポート2 (サブ) 512 byte

## chk\_tx\_ready\_port

要約 int chk\_tx\_ready\_port ( unsigned char port );

unsigned char port : port

0: シリアルポート1 (メイン)

1: シリアルポート2 (サブ)

解説 RS232C の送信バッファが満杯かをチェックする

戻値 0: 送信バッファが満杯。

1: 送信バッファは満杯ではない。

補足 送信バッファが満杯の時には、送信データをセットできない。  
送信バッファに空きができてから、送信データをセットすること。

RS232C の送信バッファサイズ

0: シリアルポート1 (メイン) 512 byte

1: シリアルポート2 (サブ) 512 byte

## set\_tx\_data\_port

要約	<pre> unsigned int set_tx_data_port ( unsigned char port,                                 unsigned char tx_data );  unsigned char port : port   0: シリアルポート1 (メイン)   1: シリアルポート2 (サブ) unsigned char tx_data : send data   ( 0x00 - 0xff )             </pre>
解説	RS232C への送信データを設定する
戻値	インデックス。送信バッファ先頭からの番号
補足	<p>RS232C の送信バッファサイズ</p> <p>0: シリアルポート1 (メイン) 512 byte</p> <p>1: シリアルポート2 (サブ) 512 byte</p> <p>この関数には送信バッファが満杯かのチェックが含まれていませんので、使用する場合には送信バッファ状態チェック関数 <code>chk_tx_ready_port()</code> と共に使用して下さい。</p>

<例>

```

void tx_string_to_port1(char *str)
{
    char *pStr;

    for (pStr = str; *pStr != '\0'); {
        while (chk_tx_ready_port(0) != 1); /* 送信バッファは満杯か ? */
        set_tx_data_port(0, (unsigned char)(*pStr)); /* 1文字送信 */
        pStr++;
    }
}
            
```

## get\_inc\_timer

---

要約            unsigned int get\_inc\_timer( void );

解説            OSタイマーのカウンタ値を獲得する。  
                 このカウンタは2 msec毎にカウンタアップされる。

戻値            カウンタ値 ( 0 - 0xffff )

補足

## set\_timer

---

要約	<pre>int set_timer( char no,                unsigned long time,                char time_unit );</pre> <p>char no : タイマ-番号 ( 1 - 10 )                  unsigned long time : タイマ-値                  char time_unit : 単位                  0: ミリセカント                  1: 秒                  2: 分</p>
解説	タイマ-を設定する
戻値	0: 設定失敗 1: 設定成功
補足	ユーザーアプリケーション用に 10 個のタイマ-が用意されている。 時間単位毎の最大値は以下である。 ミリセカント : 4294967295 ミリセカント 秒 : 274877906 秒 分 : 4581298 分



&lt;例&gt;

```

#include "kme_lib.h"

void main(void)
{
    if (req_pos_calc_from_user() == 1) {          /* 測位開始要求 */
        /* 測位開始に失敗 */
    } else {
        /* 測位開始 */
        if (set_timer(1, 15, MIN_UT) == 0) {      /* タイマ-1設定 */
            exit_user_apl();                      /* ユーザー-アプリ終了 */
        }

        while (chk_pos_calc_result() == 0) {      /* 測位終了? */
            if (check_timer(1) == 0) {           /* タイムアップ? */
                /* タイムアップ */
                /* 15分で測位ができなかったので強制終了 */
                req_pos_calc_term_from_user();
                break;
            }

            /* 10秒毎に測位終了をチェック */
            go_wait(10, SEC_UT);                 /* 10秒スリープ */
        }
    }
    exit_user_apl(); /* ユーザー-アプリ終了 */
}

```

## check\_timer

---

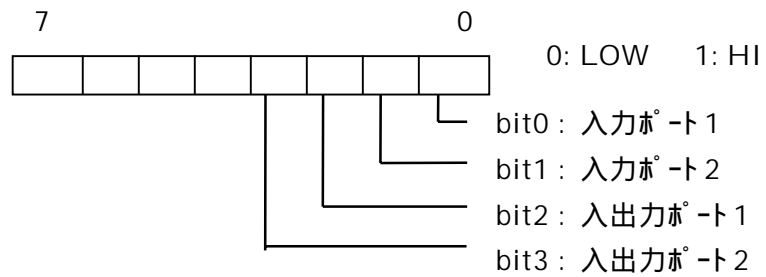
要約	int check_timer( char no );  char no: タイマ番号 ( 1 - 10 )
解説	指定タイマがタイムアップしたかチェックする
戻値	0: タイムアップした 1: タイムアップしていない
補足	

## get\_digital\_port

要約 unsigned int get\_digital\_port( void );

解説 入力ポート、出力ポートの状態を獲得する

戻値 ポート状態



bit8-15 : リザーブ

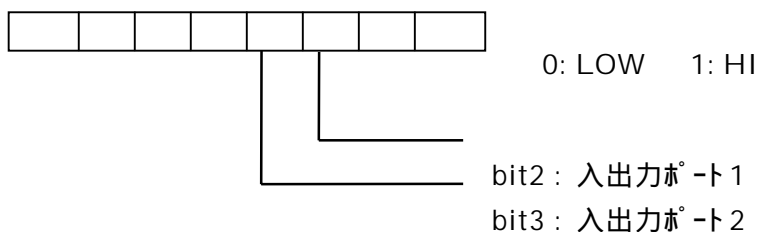
補足 電源 ON 時、出力ポートは set\_kxs83 の設定に従います。

## set\_digital\_port

要約 void set\_digital\_port( unsigned int sel\_port, unsigned int data );

unsigned int sel\_port: 出力ポート識別番号

unsigned int data: 出力データ (0 : LOW or 1: HI )



解説 出力ポートを設定する

戻値 なし

補足 set\_kxs94 にて出力モードに設定した場合のみ有効です。

<例>

```
#include "kme_lib.h"

void main(void)
{
    /* 出力ポート1をLOW に設定 */
    Set_digital_port(0x04, 0x00);

    /* 出力ポート1をHI に設定 */
    Set_digital_port(0x04, 0x04);

    /* 出力ポート2をLOW に設定 */
    Set_digital_port(0x08, 0x00);

    /* 出力ポート2をHI に設定 */
    Set_digital_port(0x08, 0x08);

    exit_user_apl();          /* ユーザーアプリを終了する */
}
```

## get\_adcon\_data

要約            unsigned char get\_adcon\_data( int ch );

int ch : アナログポート番号

0: RSSI

1: アナログポート1

2: アナログポート2

3: アナログポート3

4- 7: 予備

解説            アナログポートの状態を獲得する

戻値            値 ( 0 -255)

補足            旧機種 KX-G710x/N と変換レンジが異なります。同じ電圧を検知しても旧機種とは変換値が異なりますので注意してください。

KX-G710x/N    0-3.3V を 0-255 で変換

KX-G7201N    set\_kxs90 で設定した範囲を 0-255 で変換

## get\_adcon\_data\_10bit

---

要約            unsigned int get\_adcon\_data\_10bit( int ch );

int ch : アナログポート番号

0: RSSI

1: アナログポート1

2: アナログポート2

3: アナログポート3

4- 7: 予備

解説            アナログポートの状態を獲得する(10ビット)

戻値            値 ( 0 -1023)

補足            set\_kxs90 で設定した電圧または電流範囲を 0-1023 で変換します。

## req\_pos\_calc\_term\_from\_user

要約 void req\_pos\_calc\_term\_from\_user( void );

解説 測位を中止する

戻値 なし

補足 測位が起動してない場合には、無効となる。

<例>

```
#include "kme_lib.h"

void main(void)
{
    if (req_pos_calc_from_user() == 1) { /* 測位開始要求 */
        /* 測位開始失敗 */
    }
    else {
        /* 測位開始 */
        if (set_timer(1, 15, MIN_UT) == 0) { /* タイマ設定 */
            exit_user_apl(); /* ユーザーアプリ終了 */
        }

        while (chk_pos_calc_result() == 0) { /* 測位終了? */
            /* 測位中 */
            if (check_timer(1) == 0) {
                /* タイムアウト */
                /* 15分経過したら測位終了 */
                req_pos_calc_term_from_user();
                break;
            }
            /* 測位終了を10秒毎にチェック */
            go_wait(10, SEC_UT); /* 10秒待機 */
        }
    }
    exit_user_apl(); /* ユーザーアプリ終了 */
}
```

## req\_pos\_calc\_from\_user

---

要約           int req\_pos\_calc\_from\_user( void );

解説           測位を開始する。

戻値           0: 測位開始  
              1: 測位開始に失敗

補足           既に測位が動作していたらこの要求は無視されるが、戻り値は0が返る。



## chk\_pos\_calc\_result

---

要約           int chk\_pos\_calc\_result( void );

解説           測位が終了したかをチェックする。

戻値           0: 測位中  
              1: 測位終了

補足

## get\_pos\_age

---

要約            unsigned int get\_pos\_age( void )

解説            測位結果の経過時間（古さ）を取得する。

戻値            時間 (0 - 65535) [分]

補足

## get\_lat\_val

---

要約            `double get_lat_val( void );`

解説            緯度を獲得する

戻値            緯度(-90.0 - +90.0)[度]

補足

## get\_lon\_val

---

要約            `double get_lon_val( void );`

解説            経度を獲得する

戻値            経度(-180.0 - +180.0) [度]

補足

## set\_lat\_val

---

要約           int set\_lat\_val(double lat );

double lat : 緯度(-90.0    -    +90.0) [度]

解説           緯度を設定する.

戻値           0: 設定エラー  
              1: 設定完了

補足

## set\_lon\_val

---

要約           int set\_lon\_val(double lon);

double lon : 経度(-180.0   -   +180.0) [度]

解説           経度を設定する

戻値           0: 設定エラー  
              1: 設定完了

補足

## set\_pdop\_th

要約 int set\_pdop\_th( int pdop\_thresh)

int pdop\_thresh : PDOP 値 ( 1 ~ 10 )

解説 GPSの測位精度は、測位に使用した3つないし4つの衛星の散ばり具合に左右される傾向がある。PDOPはその衛星の散ばり具合を表す値で、衛星が広範囲に散ばっているとこの値は小さくなり、精度のよい位置が得られやすくなる。  
端末は、GPSの算出した位置情報と同時に得られるPDOP値が設定値以下の時だけ、位置情報の獲得処理を行う。そのPDOPのしきい値を設定する。

戻値 0: 設定成功  
1: 設定失敗 (パラメータ)

補足 この関数は、set\_kxs87()と同じ処理を行います。

pdop\_th を 5 以下に設定すると 2 次元測位解を採用しません。精度は上がりますが測位時間が長くなることがあります。(ファーム Ver1.20 以降)

## get\_pdop\_th

---

要約           int get\_pdop\_th( void)

解説           GPS位置情報をフィルタする時のPDOPのしきい値を獲得する。

戻値           PDOP 値 ( 1 ~ 1 0 )

補足           この関数は、  
get\_kxs87()と同じ処理を行う。



## get\_x05\_str

要約 void get\_x05\_str( char \*str )

char str[210] : x05 センテンス

x05 センテンスを獲得する為には、210 バイトの領域が必要。

解説 x05 センテンスを獲得する

戻値 なし

補足 x05 センテンスは、GPS から出力される測位情報である。

< X05センテンスのフォーマット >

"\$PKMEX05,x1,x2,x3,x4,x5,x6,,,,,x7,x8,x9,x10,x11,x12,x13,x14,x15, x16,x17,  
x18,x19,,, \*cc<CR><LF>" \*) 最後は NULL 。

1. UTC 時分秒 XX XX XX

秒  
分  
時間

2. UTC 年月日XXXX XX XX

日  
月  
年

3. 測位ステータス (1バイト)

< 警告 (測位有効) >

bit0: 受信感度悪い

Bit1: HDOP悪い

Bit2: 測位精度悪い

Bit3: 位置保留

< 致命的 (測位無効) >

bit4: 衛星数不足

Bit5: 演算不可能

bit6: その他異常

bit7: 初期異常

4. 測位品質

0= 未測位

1= 通常測位

2= DGPS測位

5. 緯度 u XX XXX.XXX

分  
度  
なし=北緯, '-'=南緯

6. 経度 u XX XX.XXX

分  
度  
なし=東経, '-'=西経

7. 標高 -9999 - 9999 [m]

8. シフト高 -9999 - 9999 [m] \*1)

9. 移動速度 -999.9 - 999.9 [Km/h]

- 10. 移動方位 0.0 - 359.9
- 11. 移動仰角 -90.0 - 90.0 [True]
- 12. 測位モード オートモード (A).
- 13. 測位次元 2=2次元測位. 3=3次元測位
- 14. チャネル数 8
- 15. 衛星情報(2バイト).
  - bit0-7: 衛星番号
  - bit8-11: 受信レベル
  - bit12 : 衛星補足
  - bit13 : 衛星使用可能
  - bit14 : 測位に使用
  - bit15 : Don't care
- 16. VDOP
- 17. PDOP
- 18. HDOP
- 19. Uere 単位はメートル。

\*cc チェックサム \*2)

\*1) ジョイト高: WGS84楕円体と海水面との差

2) チェックサム: チェックサム値は、センサ中の '\$' と '\*' とで挟まれたキャラクターを XOR した値

<例>

\$PKMEX05,093020.81,19970909,00,1,3334.566,13025.376,,,,0,29,0.8,336.3,20.9,A,  
3,8,7501,0019,0011,7617,761E,731A,7605,7409,1.2,2.3,1.9,53,,,\*2B

## get\_x06\_str

要約 void get\_x06\_str( char \*str )

char str[170] : x06 センテンス

x06 センテンスを獲得する為には、170 バイトの領域が必要。

解説 x06 センテンスを獲得する

戻値 なし

補足 x06 センテンスは、GPS から出力される受信可能衛星情報である。

< X06センテンスフォーマット >

"\$PKMEX06,XX[,XX,XX,XX,...XX,XX]\*cc<CR><LF>" \*)最後はNULL。

衛星仰角(0 - 90) \*1)

衛星方位各(0 - 360) ( 0=北, 180=南)

衛星番号

観測可能衛星数

\*cc: Checksum \*2)

\*1) 全ての可視衛星の衛星番号・仰角・方位角

\*2) チェックサム: チェックサム値は、センテンス中の '\$' と '\*' とで挟まれたキャラクター - タを XOR した値

<例>

\$PKMEX06,8,23,182,68,05,98,56,30,167,43,09,45,37,01,304,29,26,98,12,  
25,245,8,17,196,6\*6B

## get\_st\_status

要約	unsigned char get_st_status( void );
解説	端末動作状態を獲得する
戻値	<p>動作状態</p> <ul style="list-style-type: none"> <li>0: アイドル</li> <li>1: インバウンドメッセージ送信中</li> <li>2: インバウンドレポート送信中</li> <li>3: インバウンドグローバルRAM送信中</li> <li>4: アウトバウンドメッセージ受信中</li> <li>5: アウトバウンドコマンド受信中</li> <li>6: アウトバウンドグローバルRAM受信中</li> <li>7: テキスト中</li> <li>8: ローカルブッキング中</li> <li>9: リモートブッキング中</li> </ul>
補足	<p>HOST に対してホッピング (メッセージ / コマンド の送信要求) をしている時は、4 になります。</p> <p>衛星に対してホッピング (グローバルRAM の送信要求) をしている時は、6 になります。</p>

## get\_active\_mha\_ref\_num

---

要約            unsigned char get\_active\_mha\_ref\_num( void );

解説            現在送信中の mha レベル番号を獲得する

戻値            0x00 - 0xfe: 現在送信中の mha レベル番号  
                 0xff: 現在送信していない

補足

## get\_sat\_no

---

要約            unsigned char get\_sat\_no ( void );

解説            現在捕捉している衛星番号を獲得する

戻値            0 以外: 衛星番号  
                 0:        衛星を捕捉してない

補足

## get\_ncc\_quan

---

要約            unsigned char get\_ncc\_quan ( void );

解説            現在リクしている GCC の数を獲得する

戻値            現在リクしている GCC の数

補足

## get\_ncc\_id\_prio

---

要約

```
int get_ncc_id_prio( unsigned char index,  
                   unsigned char *ncc_id,  
                   unsigned char *min_pri );
```

index : インデックス。0 から数える

ncc\_id : GCC ID

min\_pri : 受信可能なメッセージの優先度の最小値

解説

GCC ID とその受信可能なメッセージの優先度の最小値を獲得する

戻値

なし

補足

get\_ncc\_quan() で獲得した GCC 数だけ、インデックスを+1 していったこの関数を呼ぶ。



## get\_num\_of\_ob\_msgs

---

要約           int get\_num\_of\_ob\_msgs( void );

解説           アウトバウンドキューに格納しているメッセージの数を獲得する

戻値           メッセージの数

補足

## get\_num\_of\_ib\_msgs

---

要約           int get\_num\_of\_ib\_msgs( void );

解説           インバウンドキューに格納しているメッセージの数を獲得する

戻値           メッセージの数

補足

## get\_week\_time\_val

---

要約            long get\_week\_time\_val( void )

解説            日曜の午前 0:00UTC からの経過秒を獲得する

戻値            秒 : 0 - 604799[秒]

補足

## calc\_gps\_week

---

要約	unsigned int calc_gps_week( void )
解説	GPS 週を獲得する
戻値	GPS 週
補足	1980 年 1 月 6 日を week#0 として開始。

## get\_total\_sats

---

要約            unsigned char get\_total\_sats( void )

解説            オートシステムの衛星の総数を獲得する。

戻値            衛星数

補足

## get\_check\_errs

---

要約            unsigned char get\_check\_errs( void )

解説            衛星がリンクの受信エラー数を獲得する

戻値            エラー数

補足

## exit\_user\_apl

---

要約           void exit\_user\_apl( void );

解説           ユーザーアプリを終了する

戻値           なし

補足

## break\_point

---

要約	int break_point( unsigned char brk_num )  unsigned char brk_num: ブレークポイント番号 (1 - 255).
解説	この関数はユーザ-アプリケーションプログラムをデバッグする時に使用される。 ユーザ-は、あらかじめユーザ-アプリケーションプログラムの中に“break_point”関数を挿入しておく必要がある。
戻値	1 : 設定完了 0 : 設定エラー。KXS70=0 になっている -1 : 設定エラー。ブレークポイント番号に 0 が設定されている
補足	“break_point” は、255 個まで設定でき、それぞれ ON/OFF 設定ができます。 “break 23,0” : ブレークポイント 23 を無効にする “break 23,1” : ブレークポイント 23 を有効にする  ブレークした後、“ctrl+g”コメントにより再開する。  これらの機能は、デバッグ機能が ON ( KXS70=1-9 ) の時に有効となります。



## chk\_break\_point

---

要約           int chk\_break\_point(unsigned char num)

unsigned char num : ブレークポイント番号 (1 - 255).

解説           ブレークポイント設定状態を獲得する

戻値           1: ブレークポイントが設定されている  
              0: ブレークポイントが設定されていない

補足

## get\_application\_dbg

---

要約            unsigned char get\_application\_dbg( void )

解説            ユーザーアプリケーション機能 ( KXS70 ) の設定状態を獲得する

戻値            0: OFF  
                 1-9: ON

補足

## led\_on

---

要約            void led\_on(void)

解説            LEDを点灯する

戻値            なし

補足

## led\_off

---

要約            void led\_off(void)

解説            LED を消灯する

戻値            なし

補足

## suspend\_ib\_msg\_tx

---

要約           void suspend\_ib\_msg\_tx( void )

解説           衛星への送信を一時中止する。

戻値           なし

補足           インバウンドキューにメッセージが格納されている場合に何らかの理由で送信を一時留保したい場合に使用します。この関数により送信キューの検索を行わなくなります。

## resume\_ib\_msg\_tx

---

要約           void resume\_ib\_msg\_tx( void )

解説           suspend\_ib\_msg\_tx()で一時中止していたメッセージ送信を再開する

戻値           なし

補足

## get\_utc\_time

要約            void get\_utc\_time( TIME\_INFO \*time );

TIME\_INFO \*time: UTC 時刻

```
struct time_buff {
    unsigned char  year;          /* 19xx */
    unsigned char  month;        /* 01-12 */
    unsigned char  day;          /* 01-31 */
    unsigned char  hour;         /* 00-23 */
    unsigned char  min;          /* 00-59 */
    unsigned char  sec;          /* 00-59 */
    unsigned char  week;         /* 00-06 */
};
typedef struct time_buff  TIME_INFO;
```

解説            UTC 時刻を獲得する

戻値            なし

補足            年情報は、80 以上が 19xx を、80 未満が 20xx を示す。

週 : 0: 日曜  
 1: 月曜  
 2: 火曜  
 3: 水曜  
 4: 木曜  
 5: 金曜  
 6: 土曜

## set\_utc\_time

要約           void set\_utc\_time( TIME\_INFO \*time );

TIME\_INFO \*time: UTC 時刻

```
struct time_buff {
    unsigned char  year;          /* 20xx */
    unsigned char  month;        /* 01-12 */
    unsigned char  day;          /* 01-31 */
    unsigned char  hour;         /* 00-23 */
    unsigned char  min;          /* 00-59 */
    unsigned char  sec;          /* 00-59 */
    unsigned char  week;         /* 00-06 */
};
typedef struct time_buff  TIME_INFO;
```

解説           UTC 時刻を設定する

戻値           なし

補足           年情報は、80 以上が 19xx を、80 未満が 20xx を示す。

週 : 0: 日曜  
 1: 月曜  
 2: 火曜  
 3: 水曜  
 4: 木曜  
 5: 金曜  
 6: 土曜



## get\_local\_time

要約 void get\_local\_time( TIME\_INFO \*local\_time, int time\_zone )

TIME\_INFO \*time: 日-加時刻

```
struct time_buff {
    unsigned char  year;          /* 20xx */
    unsigned char  month;        /* 01-12 */
    unsigned char  day;          /* 01-31 */
    unsigned char  hour;         /* 00-23 */
    unsigned char  min;          /* 00-59 */
    unsigned char  sec;          /* 00-59 */
    unsigned char  week;         /* 00-06 */
};
typedef struct time_buff  TIME_INFO;
```

int time\_zone : タイムゾーン (分) (-690 - 720)

解説 日-加時刻を取得する

戻値 なし

補足 年情報は、80 以上が 19xx を、80 未満が 20xx を示す。

週 : 0: 日曜  
 1: 月曜  
 2: 火曜  
 3: 水曜  
 4: 木曜  
 5: 金曜  
 6: 土曜

## get\_convert\_time

要約 void get\_convert\_time( TIME\_INFO \*time, int min\_time )

TIME\_INFO \*time: ローカル時刻

```
struct time_buff {
    unsigned char year;      /* 20xx */
    unsigned char month;    /* 01-12 */
    unsigned char day;      /* 01-31 */
    unsigned char hour;     /* 00-23 */
    unsigned char min;      /* 00-59 */
    unsigned char sec;      /* 00-59 */
    unsigned char week;     /* 00-06 */
};
typedef struct time_buff TIME_INFO;
```

int time\_zone : タイムゾーン (分) (-690 - 720)

解説 バッファ内の時刻データを、ローカル時間に変換する

戻値 なし

補足 年情報は、80 以上が 19xx を、80 未満が 20xx を示す。

週 : 0: 日曜  
1: 月曜  
2: 火曜  
3: 水曜  
4: 木曜  
5: 金曜  
6: 土曜

## calc\_distance

---

要約 `double calc_distance(double lat0, double lon0, double lat1, double lon1)`

`double lat0` : 位置 0 (緯度)

`double lon0` : 位置 0 (経度)

`double lat1` : 位置 1 (緯度)

`double lon1` : 位置 1 (経度)

解説           2点間の距離を獲得する

戻値           距離 (メートル)

補足

## req\_apl\_sat\_predict

要約           void req\_apl\_sat\_predict( unsigned long offset\_time )

unsigned long offset\_time : 最低時間 (1 – 4294967294)[秒]

解説           最低時間後のオフセット衛星飛来時刻の計算を開始する

戻値           なし

補足           “offset\_time” が 3600 秒のとき、端末は 1 時間以降の衛星飛来時刻を返す。 本関数使用の際は軌道計算用ライブラリが必要です。

<例.>

```
void main(void) {
    TIME_INFO arrival_time;
    unsigned long sv_pass_time;

    req_apl_sat_predict( 60 * 60 );

    while ( chk_apl_sat_predict_result() == 1 );

    if ( get_apl_sat_predict_time( &arrival_time ) ) {
        /* 軌道計算できた */
    }
    else {
        /* 軌道計算できなかった */
    }

    exit_user_apl();    /* ユーザーアプリ終了 */
}
```

## chk\_apl\_sat\_predict\_result

---

要約	int chk_apl_sat_predict_result( void )
解説	軌道計算が終了したかどうかをチェックする
戻値	0: 終了 1: 計算中
補足	本関数使用には軌道計算用ライブラリが必要です。

## get\_apl\_sat\_predict\_time

要約           int get\_apl\_sat\_predict\_time( TIME\_INFO \*time )

TIME\_INFO \*time: 衛星飛来時刻

解説           衛星飛来時刻を獲得する

戻値           0: 衛星飛来時刻を獲得できなかった  
              1: 衛星飛来時刻を獲得できた

補足           軌道計算には、端末位置 (KXS23) が設定されていて、さらに端末が衛星から軌道要素を受信していることが必要。  
              本関数使用の際は軌道計算用ライブラリが必要です。

## get\_sys\_info

要約 char get\_sys\_info( int obj )

int obj : 番号  
 0: ROMバージョン  
 3: インパウンド/アウトパウンドキーのチェックサムチェック  
 4: ASIC リード/ライトチェック  
 5: IC リード/ライトチェック  
  
 8: 衛星通信のデジタ外部のループバックテスト  
 9: GPS ROMバージョン  
 10: 端末 ID のチェックサムのチェック  
 11: コンフィグレーションパラメータのチェックサムのチェック  
 12: GPS ステータス

解説 ROMバージョン、自己診断テスト結果を獲得する

戻値 0: OK  
 1: NG

補足

## get\_kme\_serial\_no

要約                   void get\_kme\_serial\_no( char \*serial\_no )

char \*serial\_no 文字列バッファのアドレス ( 獲得するシリアル番号は 11 桁 )

解説                   端末の製造番号を獲得する。

戻値                   なし

補足                   格納バッファは、少なくとも 12 バイト必要。

```
<ex.>
void main(void)
{
    char serial_no[12];

    get_kme_serial_no(serial_no);

    rs_tx_str(serial_no);
}
```

実行結果

0ACDE801001



## os\_monitor

---

要約	void os_monitor(int *os_work)  unsigned int *os_work : work buffer
解説	端末本体ジョブのヘルス状態を監視します。もし本体ジョブ内に不具合(ジョブ不起動等)が発生した場合、本処理により修復します。
戻値	なし
補足	ユーザーアプリで 1 分程度の間隔で本関数を起動する事を推奨します。  グローバルエリア内にワーク領域"os_work"を指定してください。

```
<ex.>
/** Global Data */

int os_work;;

void main(void)
{
    os_work = 0; /* initialize of work */

    os_monitor(&os_work); /* make the calling one per one minute */
    /* user original management */
    .....
    os_monitor(&os_work); /* make the calling one per one minute */
    /* user original management */
    .....
}

```

## print\_double

要約           char \*print\_double( char \*string,  
  double value,  
  int width,  
  int prisdiction)

double value:変換する数値  
unsigned char \*string:文字列格納バッファ  
int width:全変換桁数  
int prisdiction:小数部桁数

解説           浮動小数点を文字列に変換する

戻値           変換された文字列のポインタ

補足           <note>  
標準関数の "sprintf()" は、スタックを消費するので、しばしばユーザーアプリケーションの不具合の原因になっていることがあります。特に浮動小数点変数を変換すると数百バイトのスタックを消費します。  
できるだけ標準関数の"sprintf()"をさけて"print\_double()"等の代用関数を使用してください。

```
<ex.>
void main(void)
{
    char work1[20], work2[20];
    double value1, value2;

    value1 = 123.456;
    value2 = 456.789;

    /* sprintf(work1, "%f", value1); */
    print_double(work1, value1, 7, 3);
    rs_tx_str(work1);

    /* sprintf(work, "%f, %f", value1, value2); */
    print_double(work1, value1, 7, 1);
    print_double(work2, value2, 7, 1);
    strcat(work1, work2);
    rs_tx_str(work1);
}
```

実行結果  
123.456  
123.5 456.8

## orb\_sprintf1

要約 void orb\_sprintf1(char \*pbuff, char \*pformat, int data0)

char \*pbuff 展開するバッファのアドレス  
 char \*pformat 変換する書式  
 int data0 変換するデータ

解説 1つのint型変数を指定書式に変換する

戻値 なし

補足 変換指定文字  
 d:符号付き 10進数で読み込みます  
 u:符号なし 10進数で読み込みます  
 x:16進整数で読み込みます

```
<ex.>
void main(void)
{
    char work[20];
    int data = 10;

    /* sprintf(work, "DATA:%d¥r¥n", data) */
    orb_sprintf1(work, "DATA: %d¥r¥n", data);

    rs_tx_str(work);
}
```

Result:  
 DATA:10

## orb\_sprintf2

要約 void orb\_sprintf2(char \*pbuff, char \*pformat, int data0,  
int data1)

char \*pbuff 展開するバッファのアドレス  
 char \*pformat 変換する書式  
 int data0 変換するデータ  
 int data1 変換するデータ

解説 2つのint型変数を指定書式に変換する

戻値 なし

補足 変換指定文字  
 d:符号付き 10進数で読み込みます  
 u:符号なし 10進数で読み込みます  
 x:16進整数で読み込みます

```
<ex.>
void main(void)
{
    char work[20];
    int data = 10;

    /* sprintf(work, "DATA:0x%02x, 0x%02x\r\n", data, data+10) */
    orb_sprintf2(work, "DATA:0x%02x, 0x%02x\r\n", data, data+10);

    rs_tx_str(work);
}
```

実行結果  
 DATA:0x0a,0s0x14







## orb\_sprintf6

```

要約 void orb_sprintf6(char *pbuff, char *pforamt, int data0
                                           int data1,
                                           int data2,
                                           int data3,
                                           int data4,
                                           int data5)
    
```

```

char *pbuff    展開するバッファのアドレス
char *pformat  変換する書式
int data0     変換するデータ0
int data1     変換するデータ1
int data2     変換するデータ2
int data3     変換するデータ3
int data4     変換するデータ4
int data5     変換するデータ5
    
```

解説 int型変数を指定書式に変換する

戻値 なし

補足 変換指定文字  
d:符号付き 10進数で読み込みます  
u:符号なし 10進数で読み込みます  
x:16進整数で読み込みます

```

<ex.>
void main(void)
{
    char work[20];
    int data = 10;

    /* sprintf(work, "DATA:0x%02x, 0x%02x, 0x%02x, 0x%02x, 0x%02x, 0x%02x¥r¥n",
                  data, data+10, data+20, data+30, data+40, data+50) */
    orb_sprintf6(work, "DATA:0x%02x, 0x%02x, 0x%02x, 0x%02x, 0x%02x, 0x%02x¥r¥n",
                  data, data+10, data+20, data+30, data+40, data+50);

    rs_tx_str(work);
}
    
```

実行結果  
DATA:0x0A, 0x14, 0x1E, 0x28, 0x32, 0x3C



## orb\_sprintf7

---

要約 void orb\_sprintf7(char \*pbuff, char \*pformat, int data0,  
int data1,  
int data2,  
int data3,  
int data4,  
int data5,  
int data6)

char \*pbuff 展開するバッファのアドレス  
char \*pformat 変換する書式  
int data0 変換するデータ 0  
int data1 変換するデータ 1  
int data2 変換するデータ 2  
int data3 変換するデータ 3  
int data4 変換するデータ 4  
int data5 変換するデータ 5  
int data6 変換するデータ 6

解説 int型変数を指定書式に変換する

戻値 なし

補足 変換指定文字  
d:符号付き 10 進数で読み込みます  
u:符号なし 10 進数で読み込みます  
x:16 進整数で読み込みます

```
<ex.>
void main(void)
{
    char work[20];
    int data = 10;

    /* sprintf(work, "DATA:%d, %d, %d, %d, %d, %d, %d\r\n",
        data, data+10, data+20, data+30, data+40, data+50, data+60) */
    orb_sprintf7(work, "DATA:%d, %d, %d, %d, %d, %d, %d\r\n",
        data, data+10, data+20, data+30, data+40, data+50, data+60);
    rs_tx_str(work);
}
```

実行結果  
DATA:10, 20, 30, 40, 50, 60, 70

## set\_kxs01

---

要約	int set_kxs01(unsigned char value)  unsigned char value:GCC 番号(0 - 255)
解説	GCC 番号を設定する
戻値	0: 設定成功 1: 設定失敗 (パラメータエラー)
補足	端末はこのgccとリンクしている衛星を捕捉しようとする。

## get\_kxs01

---

要約	unsigned char get_kxs01(void)
解説	GCC 番号 (0 - 255)を獲得する
戻値	GCC 番号(0 - 255).
補足	

例:

```
#include "kmelib.h"           /* ヘッダファイルを含める。 */

void main(void)
{
    unsigned char ncc_id;

    set_kxs01(43);           /* GCC 番号を設定する */
    ncc_id = get_kxs01();    /* GCC 番号を獲得する */
}
```

## set\_kxs02

要約	int set_kxs02(unsigned char value)  unsigned char value (0, 1) 0: インバウンドメッセージをすぐ送信 1: メッセージレポートは SC の中でキューイングされ、GCC からポーリングされるのを待つ
解説	デフォルトポートを設定する
戻値	0: 設定成功 1: 設定失敗 (パラメータエラー)
補足	

## get\_kxs02

要約	unsigned char get_kxs02(void)
解説	デフォルトポート設定値を獲得する
戻値	デフォルト polled 設定値を獲得する 0: インバウンドメッセージをすぐ送信 1: メッセージレポートは SC の中でキューイングされ、GCC からポーリングされるのを待つ
補足	

例:

```
#include "kmlib.h"                /* ヘッダファイルをインクルードする。*/

void main(void)
{
    unsigned char polled;

    set_kxs02(0);                  /* デフォルト polledを設定する */
    polled = get_kxs02();         /* デフォルト polledを獲得する */
}
```

## set\_kxs03

---

要約           int set\_kxs03(unsigned char value)

                unsigned char value (0 - 3)  
                 0: 不急 (最も低い優先度)  
                 1: 通常  
                 2: 至急  
                 3: 速達 (インバウンドのみ、最も高い優先度)

解説           デフォルト優先度を設定する

戻値           0: 設定成功  
                 1: 設定失敗 (パラメータエラー)

補足

## get\_kxs03

---

要約           unsigned char get\_kxs03(void)

解説           デフォルト優先度設定値を取得する

戻値           デフォルト優先度  
                 0: 不急 (最も低い優先度)  
                 1: 通常  
                 2: 至急  
                 3: 速達 (インバウンドのみ、最も高い優先度)

補足

例:

```
#include "kmlib.h"                                   /* ヘッダファイルをインクルードする。*/

void main(void)
{
    unsigned char priority;

    set_kxs03(0);                                   /* デフォルト priorityを設定する */
    priority = get_kxs03();                       /* デフォルトpriorityを取得する */
}
```

## set\_kxs04

---

要約	int set_kxs04(unsigned char value)  unsigned char value : デフォルトレボ-トO/R インデ-クター
解説	デフォルトレボ-トO/R インデ-クター(0 -3)を設定する
戻値	0: 設定成功 1: 設定失敗 (パラメ-タエラー)
補足	

## get\_kxs04

---

要約	unsigned char get_kxs04(void)
解説	デフォルトレボ-トO/R インデ-クター(0 -3) 設定値を獲得する
戻値	デフォルトレボ-トO/R インデ-クター(0 -3)
補足	

例:

```
#include "kmelib.h"           /* ヘッダ-ファイルをインクル-ドする. */

void main(void)
{
    unsigned char or_ind;

    set_kxs04(0);             /* デフォルトレボ-トO/R インデ-クター(0 -3)を設定する*/
    or_ind = get_kxs04();     /* デフォルトレボ-トO/R インデ-クター(0 -3)を獲得する*/
}
```

## set\_kxs05

---

要約	int set_kxs05(unsigned char value)  unsigned char value : メッセージ O/R インデキータ(0 - 15)
解説	デフォルトメッセージ O/R インデキータ(0 - 15)を設定する
戻値	0: 設定成功 1: 設定失敗 (パラメータエラー)
補足	

## get\_kxs05

---

要約	unsigned char get_kxs05(void)
解説	デフォルトメッセージ O/R インデキータ(0 - 15) 設定値を獲得する
戻値	メッセージ O/R インデキータ(0 - 15)
補足	

例 :

```
#include "kmelib.h"           /* ヘッダファイルをインクルードする。*/

void main(void)
{
    unsigned char or_ind;

    set_kxs05(0);             /* デフォルトメッセージ O/R インデキータを設定する */
    or_ind = get_kxs05();     /* デフォルトメッセージ O/R インデキータを獲得する */
}
```

## set\_kxs06

---

要約           int set\_kxs06(unsigned char value)  
                   unsigned char value : Ack レベル (0 - 4)

解説           デフォルトAck レベル (0 - 4)を設定する

戻値           0: 設定成功  
                   1: 設定失敗 (パラメータエラー)

補足

## get\_kxs06

---

要約           unsigned char get\_kxs06(void)

解説           デフォルトAck レベル (0 - 4) 設定値を取得する

戻値           Ack レベル (0 - 4)

補足

例 :

```
#include "kmlib.h"                   /* ヘッダファイルをインクルードする。*/

void main(void)
{
    unsigned char ack_level;

    set_kxs06(0);                   /* デフォルトAck レベルを設定する */
    ack_level = get_kxs05();       /* デフォルトAck レベル を取得する */
}
```

## set\_kxs07

要約           int set\_kxs07(unsigned char type, unsigned char object)

unsigned char type : デフォルトメッセージのタイプ (0 -15)

unsigned char object: サブタイプ (0 - 4)

解説           デフォルトメッセージのタイプ (0 -15)を設定する

戻値           0: 設定成功  
              1: 設定失敗 (パラメータエラー)

補足

## get\_kxs07

要約           void get\_kxs07(unsigned char \*type, unsigned char \*object)

unsigned char \*type : デフォルトメッセージのタイプ (0 -15)

unsigned char \*object: サブタイプ (0 - 4)

解説           デフォルトメッセージのタイプ (0 -15) 設定値を獲得する

戻値           なし

補足

例:

```
#include "kmlib.h"                               /* ヘッダファイルをインクルードする. */

void main(void)
{
    unsigned char type, sub_type;

    set_kxs07(0, 5);                             /* デフォルトメッセージのタイプを設定する */
    set_kxs07(14, 0/*dummy*/);                /* デフォルトメッセージのタイプを設定する */
    get_kxs05(&type, &sub_type);              /* デフォルトメッセージのタイプを獲得する */
}
```



## set\_kxs08

---

要約	int set_kxs08(unsigned char value)  unsigned char value : サービスタイプ (0 - 4, 10 - 14)
解説	デフォルトサービスタイプ (0 - 4, 10 - 14)を設定する
戻値	0: 設定成功 1: 設定失敗 (パラメータエラー)
補足	

## get\_kxs08

---

要約	unsigned char get_kxs08(void)
解説	サービスタイプ (0 - 4, 10 - 14) 設定値を獲得する
戻値	サービスタイプ (0 - 4, 10 - 14)
補足	

例 :

```
#include "kmelib.h"           /* ヘッダファイルをインクルードする。*/

void main(void)
{
    unsigned char service;

    set_kxs08(2);              /* デフォルトサービスタイプを設定する */
    service = get_kxs08();     /* デフォルト サービスタイプを獲得する */
}
```

## set\_kxs14

要約	<pre>int set_kxs14(unsigned char value)</pre> <p>unsigned char value : GCCサーチモード (0 - 4)</p> <p>0: kxs01に設定されているGCCを連続的にダウリンクバインドの中から検索する</p> <p>1: kxs01に設定されているGCCを1回探す。もし見つからなければ最初に発見したダウリンクとのロックを維持する。</p> <p>2: 最初に発見したダウリンクとのロックを維持する。</p> <p>3: kxs01に設定されているGCCを1回探す。もし見つからなければ任意のNCCを含むものの検索を開始、もしなにもなければ、最初に発見したダウリンクとのロックを維持する。</p> <p>4: kxs01に設定されているGCCを1回探す。もし見つからなければグローバル衛星がkxs01に設定されているGCCのダウリンクを検索し続ける。</p>
解説	GCCサーチモードを設定する
戻値	<p>0: 設定成功</p> <p>1: 設定失敗 (パラメータエラー)</p>
補足	

## get\_kxs14

要約	<pre>unsigned char get_kxs14(void)</pre>
解説	GCCサーチモード設定値を獲得する
戻値	GCCサーチモード (0 - 4).
補足	

例 :

```
#include "kmlib.h"           /* ヘッダファイルをインクルードする。 */
void main(void)
{
    unsigned char search_mode;

    set_kxs14(1);             /* NCCサーチモードを設定する */
    search_mode = get_kxs14(); /* NCCサーチモードを獲得する */
}
```

## set\_kxs15

---

要約           int set\_kxs15(unsigned char block, unsigned int channel)

unsigned char block : 番号 (0 - 23)  
 unsigned int ch     : ダウンリンクチャンネル (0 - 399)

解説           優先的に探索するダウンリンクチャンネルを設定する

戻値           0: 設定成功  
               1: 設定失敗 (パラメータエラー)

補足           SCが衛星ダウンリンクアキュイジションサーチを開始する24チャンネル

## get\_kxs15

---

要約           void get\_kxs15(unsigned int \*array)

unsigned int \*array : ダウンリンクチャンネル (24チャンネル)

解説           優先的に探索するダウンリンクチャンネル (24チャンネル) 設定値を獲得する

戻値           なし

補足

例 :

```
#include "kme_lib.h"                               /* ヘッダファイルを含める。*/

void main(void)
{
    unsigned int  channel[24];

    set_kxs15(0, 80);                              /* ダウンリンクチャンネルを設定する */
    get_kxs15(channel);                            /* ダウンリンクチャンネル (24チャンネル) を獲得する */
}
```

## set\_kxs16

---

要約	int set_kxs16(unsigned char value)  unsigned char value : スレッシユ ( 1 - 100 )
解説	最大チェックサムエラー数を設定する
戻値	0: 設定成功 1: 設定失敗 ( パラメータエラー ) 2: 排他エラー: KXS17(1)の時は KXS16(51 - 100)にできない
補足	異なったデータリンクの獲得を試みる前に連続する指定フレーム ( KXS17 ) で許されるデータリンクチェックサムエラーの最大数

## get\_kxs16

---

要約	unsigned char get_kxs16(void)
解説	最大チェックサムエラー数設定値を獲得する
戻値	最大チェックサムエラー数 ( 1 - 100 ).
補足	

例 :

```
#include "kme_lib.h"                /* ヘッダファイルをインクルードする. */

void main(void)
{
    unsigned char threshold;

    set_kxs16(10);                    /* 最大チェックサムエラー数を設定する */
    threshold = get_kxs16();          /* 最大チェックサムエラー数を獲得する */
}
```

## set\_kxs17

---

要約	int set_kxs17(unsigned char value)  unsigned char value : フレーム数 ( 1 - 16 )
解説	チェックサムエラーをカウントするフレーム数を設定する
戻値	0: 設定成功 1: 設定失敗 ( パラメータエラー ) 2: 排他エラー : KXS17(1)の時は KXS16(51 - 100)にできない
補足	連続するフレーム ( KXS17 ) でチェックサムエラー ( KXS16 ) の検証をする。

## get\_kxs17

---

要約	unsigned char get_kxs17(void)
解説	チェックサムエラーをカウントするフレーム数の設定値を獲得する
戻値	フレーム数 ( 1 - 16 ).
補足	

例 :

```
#include "kme_lib.h"           /* ヘッダファイルをインクルードする. */

void main(void)
{
    unsigned char count;

    set_kxs17(2);              /* フレーム数を設定する */
    count = get_kxs17();      /* フレーム数を獲得する */
}
```

## set\_kxs18

---

要約           int set\_kxs18(unsigned char value)

unsigned char value : モード  
0: OFF  
1: ON

解説           連続測位モードを設定する

戻値           0: 設定成功  
1: 設定失敗 (パラメータ)  
2: 排他エラー (KXS24(0) の時 KXS18(1)にできない)

補足

## get\_kxs18

---

要約           unsigned char get\_kxs18(void)

解説           連続測位モード設定値を獲得する

戻値           モード  
0: 連続測位モード OFF  
1: 連続測位モード ON

補足

例 :

```
#include "kme_lib.h"                           /* ヘッダファイルをインクルードする。*/

void main(void)
{
    unsigned char mode;

    set_kxs18(0);                           /* 連続測位モードをOFFに設定する */
    mode = get_kxs18();                   /* 連続測位モードを獲得する */
}
```

## set\_kxs23

---

```
要約      int set_kxs23(double lat, double lon)

          double lat : 緯度 (-90.0000 - +90.0000)[度]
          double lon : 経度(-180.0000 - +180.0000) [度]
```

```
解説      緯度・経度を設定する
```

```
戻値      0: 設定成功
          1: 設定失敗 (パラメータエラー)
          2: 排他エラー
```

```
補足
```

## get\_kxs23

---

```
要約      void get_kxs23(double *lat, double *lon)

          double *lat : 緯度 (-90.0000 - +90.0000) [度]
          double *lon : 経度(-180.0000 - +180.0000) [度]
```

```
解説      緯度・経度を獲得する
```

```
戻値      なし
```

```
補足
```

```
例 :
```

```
#include "kme_lib.h"                /* ヘッダファイルをインクルードする. */

void main(void)
{
    double lat, lon;;

    set_kxs23(33.344, 134.333); /* 緯度・経度を設定する */
    get_kxs22(&lat, &lon);    /* 緯度・経度を獲得する */
}
```

## set\_kxs24

要約 int set\_kxs24(unsigned char value)

unsigned char value : モード

0: 測位機能OFF

1: 測位機能 ON

解説 測位モード設定値を設定する

戻値 0: 設定成功  
1: 設定失敗 (パラメータエラー)  
2: 排他エラー : KXS18(1) / KXS24(0)

補足

## get\_kxs24

要約 unsigned char get\_kxs24(void)

解説 測位モードを取得する

戻値 測位モード  
0: OFF  
1: ON

補足

例 :

```
#include "kme_lib.h"                /* ヘッダファイルをインクルードする。*/

void main(void)
{
    unsigned char mode;

    set_kxs24(1);                    /* 測位機能を有効にする */
    mode = get_kxs24();              /* 測位モードを取得する */
}
```



## set\_kxs25

要約	int set_kxs25(int value)
	unsigned char value : type of sent message 0: 緯度経度情報 1: 緯度経度情報 + 付加情報 (高度、速度、方位等)
解説	KXB コマンドでの GPS 測位情報のフォーマットを設定する
戻値	0: 設定成功 1: 設定失敗 (パラメータエラー) 2: 排他エラー
補足	フォーマット詳細はコマンドリスト仕様書を参照。

## get\_kxs25

要約	int get_kxs25(void)
解説	KXBコマンドでのGPS測位情報フォーマット設定値を獲得する
戻値	フォーマット 0: LAT/LON 1: NMEA
補足	

例 :

```
#include "kme_lib.h"                /* ヘッダファイルをインクルードする。 */

void main(void)
{
    unsigned char mode;

    set_kxs25(0);                    /* GPS測位情報のフォーマットを設定する */
    mode = get_kxs25();              /* GPS測位情報のフォーマットを獲得する */
}
```

## set\_kxs26

---

要約	int set_kxs26(unsigned char value)
	unsigned char value : 時間( 2 – 30) [秒]
解説	DTE に対するホーリングの応答の待ち時間を設定する
戻値	0: 設定成功 1: 設定失敗 (パラメータエラー)
補足	DTE にホーリングコマンド (システムアカウント) を送信した後、この時間内の応答をホーリングの応答と判断する。

## get\_kxs26

---

要約	unsigned char get_kxs26(void)
解説	DTE に対するホーリングの応答の待ち時間設定値を獲得する
戻値	時間 ( 2 – 30 秒 )
補足	

例 :

```
#include "kme_lib.h"           /* ヘッダファイルをインクルードする。 */

void main(void)
{
    unsigned char timeout;

    set_kxs26(5);               /* ホーリングの応答の待ち時間を設定する */
    timeout = get_kxs26();     /* ホーリングの応答の待ち時間を獲得する */
}
```

## set\_kxs27

---

要約	int set_kxs27(unsigned char value)  unsigned char value : 時間 ( 1 – 30) [秒]
解説	シリアルポートの ACK 時間を設定する
戻値	0: 設定成功 1: 設定失敗 (パラメータエラー)
補足	パケットの最後のバイトを送った後、この時間だけACK/NACKを待ち再送する

## get\_kxs27

---

要約	unsigned char get_kxs27(void)
解説	シリアルポートのACK時間設定値を獲得する
戻値	時間 ( 1 – 30 秒 )
補足	

例 :

```
#include "kme_lib.h"           /* ヘッダファイルをインクルードする。*/

void main(void)
{
    unsigned char timeout;

    set_kxs27(5);               /* シリアルポートのACK時間を設定する */
    timeout = get_kxs27();     /* シリアルポートのACK時間を獲得する */
}
```

## set\_kxs28

---

要約	int set_kxs28(unsigned char value)  unsigned char value : 回数 ( 0 - 255 )
解説	シリアルポートのリトライ回数を設定する
戻値	0: 設定成功 1: 設定失敗 (パラメータエラー)
補足	この回数だけリトライして有効なACKが返ってこなかったらリンクを諦めてする (0=諦めない)

## get\_kxs28

---

要約	unsigned char get_kxs28(void)
解説	シリアルポートのリトライ回数設定値を獲得する
戻値	回数 ( 0 - 255 )
補足	

例 :

```
#include "kme_lib.h"                /* ヘッダファイルをインクルードする. */

void main(void)
{
    unsigned char count;

    set_kxs28(5);                    /* シリアルポートのリトライ回数を設定する */
    count = get_kxs28();             /* シリアルポートのリトライ回数を獲得する */
}
```

## set\_kxs29

---

要約           int set\_kxs29(unsigned char value)

unsigned char value : モード  
 0: アホ-トルホ-トを送信しない  
 1: アホ-トルホ-トを送信する

解説           アホ-トルホ-ト送信モードを設定する

戻値           0: 設定成功  
 1: 設定失敗 (パラメータエラー)

補足

## get\_kxs29

---

要約           unsigned char get\_kxs29(void)

解説           アホ-トルホ-ト送信モード設定値を獲得する

戻値           0: アホ-トルホ-トを送信しない  
 1: アホ-トルホ-トを送信する

補足

## set\_kxs30

---

### 要約

```
int set_kxs30(unsigned char gcc_id,
              unsigned char polled,
              unsigned char srv_type,
              unsigned char or_ind,
              char *inf)
```

unsigned char gcc\_id: GCC番号 (0 - 255 )  
 unsigned char polled: ホールト (0, 1)  
 unsigned char srv\_type: サービスタイプ (0 - 4, 10 - 14)  
 unsigned char or\_ind: O/Rインディケータ(0 - 3)  
 unsigned char \*inf: テータ(0x00 - 0xff) \* 6 バイト

### 解説

アトリビュートの内容を設定する

### 戻値

0: 設定成功  
 1: 設定失敗 (パラメータエラー)  
 2: 排他エラー

### 補足

## get\_kxs30

---

### 要約

```
void get_kxs30(unsigned char *ncc_id,
               unsigned char *polled,
               unsigned char *srv_type,
               unsigned char *or_ind,
               char *inf)
```

unsigned char ncc\_id: GCC番号(0 - 255 )  
 unsigned char polled: ホールト (0, 1)  
 unsigned char srv\_type: サービスタイプ (0 - 4, 10 - 14)  
 unsigned char or\_ind: O/Rインディケータ(0 - 3)  
 char \*inf: テータ(0x00 - 0xff) \* 6 バイト

### 解説

アトリビュートの内容を獲得する

### 戻値

### 補足

例：

```
#include "kme_lib.h"          /* ヘッダファイルをインクルードする。*/

void main(void)
{
    unsigned char ncc_id, polled, srv_type, or_ind;
    char inf[6];

    set_kxs30(1, 0, 2, 1, "012345"); /* ホストを設定する */
    set_kxs29(1);

    get_kxs30(&ncc_id, &polled, &srv_type, &or_ind, inf);
}
```

## set\_kxs31

---

要約           int set\_kxs31(unsigned char value)

unsigned char value: モード (0, 1)

0: プロトコルモード

1: ハイモード

解説           RS232C の通信モードを設定する

戻値           0: 設定成功

1: 設定失敗 (パラメーター)

補足

## get\_kxs31

---

要約           unsigned char get\_kxs31(void)

解説           RS232Cの通信モード設定値を獲得する

戻値           モード (0:プロトコル, 1:ハイ)

補足



例1:

```
#include "kme_lib.h"          /* ヘッダファイルをインクルードする。*/

void main(void)
{
    /* プロットに設定 */
    set_kxb00();      /* KXB設定を解除 */
    set_kxs31(0);     /* プロットを設定する */
}
```

例2:

```
#include "kme_lib.h"          /* ヘッダファイルをインクルードする。*/

void main(void)
{
    BMODE_TRGGER_CODE bmode_code;

    /* プロットに設定 */
    set_kxb00();      /* KXB設定を解除 */
    set_kxs31(1);     /* プロットを設定 */

    /* プロットのパラメータを設定 */
    set_kxs32(1);     /* プロットトリガを設定する */
    set_kxs33(1);     /* プロットタイムアウトを設定する */
    set_kxs34(200);   /* プロット長を設定する */
    bmode_code.tx_som = 2;
    bmode_code.tx_eom = 3;
    bmode_code.rx_som = 2;
    bmode_code.rx_eom = 3;
    set_kxs35(&bmode_code); /* TX_SOM, TX_EOM, RX_SOM, RX_EOMを設定
する */
    set_kxs36(0);     /* プロットのメッセージタイプを設定する */
}
```

## set\_kxs32

---

要約           int set\_kxs32(unsigned char value)

unsigned char value : トリガ - (0, 1)

0: 最初のデータを受信してバイト長受信あるいはバイトタイムアウト秒経過したら送信開始

1: 最初のデータを受信してバイト長受信あるいはバイトタイムアウト秒経過するまでにバイト RX\_SOM/RX\_EOMを受信したら送信開始

解説           バイトトリガ-を設定する

戻値           0: 設定成功

1: 設定失敗 (パラメータエラー)

補足

## get\_kxs32

---

要約           unsigned char get\_kxs32(void)

解説           バイトトリガ-設定値を獲得する

戻値           トリガ-(0, 1).

補足

## set\_kxs33

---

要約	int set_kxs33(unsigned long value)  unsigned long value : 時間 (1 – 604800) [秒]
解説	ハートタイムアウトを設定する
戻値	0: 設定成功 1: 設定失敗 (パラメータエラー)
補足	set_kxs32 参照

## get\_kxs33

---

要約	unsigned long get_kxs33(void)
解説	ハートタイムアウト設定値を獲得する
戻値	時間 (1 – 604800 秒)
補足	

## set\_kxs34

---

要約	int set_kxs34(unsigned int value)  unsigned char value : パイロット長 (1 - インパルス幅)
解説	パイロット長を設定する
戻値	0: 設定成功 1: 設定失敗 (パラメータエラー) 2: 排他エラー: KXS48
補足	set_kxs32 参照

## get\_kxs34

---

要約	unsigned int get_kxs34(void)
解説	パイロット長設定値を取得する
戻値	長さ (1 - インパルス幅).
補足	

## set\_kxs35

要約 int set\_kxs35(BMODE\_TRGGER\_CODE \*bmode\_code)

BMODE\_TRGGER\_CODE \*bmode\_code  
 unsigned char tx\_som : (0x00 - 0xff)  
 unsigned char tx\_eom : (0x00 - 0xff)  
 unsigned char rx\_som : (0x00 - 0xff)  
 unsigned char rx\_eom : (0x00 - 0xff)

解説 ハート時の、送信・受信 SOM,EOM を設定する

戻値 0: 設定成功  
 1: 設定失敗 (パラメータエラー)  
 2: 排他エラー

補足 SOMとEOMは同じ値に設定できない

set\_kxs32 参照

## get\_kxs35

要約 void get\_kxs35(BMODE\_TRGGER\_CODE \*bmode\_code)

BMODE\_TRGGER\_CODE \*bmode\_code  
 unsigned char tx\_som : (0x00 - 0xff)  
 unsigned char tx\_eom : (0x00 - 0xff)  
 unsigned char rx\_som : (0x00 - 0xff)  
 unsigned char rx\_eom : (0x00 - 0xff)

解説 ハート時の、送信・受信SOM,EOM設定値を獲得する

戻値 なし

補足

## set\_kxs36

---

要約	int set_kxs36(unsigned char value)
	unsigned char value : type (0 - 2) 0: メッセージ 1: レポート 2: グローバルプログラム
解説	ハイモートメッセージタイプを設定する
戻値	0: 設定成功 1: 設定失敗 (パラメータエラー)
補足	ハイモートのデータをレポート/メッセージ/データグラムのどれで送信するかの設定

## get\_kxs36

---

要約	unsigned char get_kxs36(void)
解説	ハイモートメッセージタイプ設定値を獲得する
戻値	メッセージタイプ (0 - 2) 0: インバウンドメッセージ 1: インバウンドレポート 2: インバウンドグローバルプログラム
補足	

## set\_kxs37

要約           int set\_kxs37(unsigned char value)

unsigned char value : モード  
0: OFF  
1: ON

解説           パワーダウンモードを設定する

戻値           0: 設定成功  
1: 設定失敗 (パラメータエラー)

補足

## get\_kxs37

要約           unsigned char get\_kxs37(void)

解説           パワーダウンモード設定値を獲得する

戻値           モード (0:OFF, 1:ON)

補足

例:

```
#include "kme_lib.h"                           /* ヘッダファイルをインクルードする。*/

void main(void)
{
    /* set that the power down mode is on. */
    set_kxs37(1);                               /* パワーダウンモードを設定する */

    set_kxs23(34.3333, -45.6666); /* 緯度・経度を設定する */

    set_kxs39(300);                            /* インアクティブインターバルを設定する*/
}
```

## set\_kxs39

---

要約           int set\_kxs39(unsigned long value)  
                   unsigned char value : 時間 (0 – 86400) [秒]

解説            インアクティブ インターバルを設定する

戻値            0: 設定成功  
                   1: 設定失敗 (パラメーター)

補足

## get\_kxs39

---

要約            unsigned long get\_kxs39(void)

解説            インアクティブ インターバル設定値を獲得する

戻値            時間 (0 – 86400 秒)

補足



## set\_kxs40

---

要約	int set_kxs40(unsigned char value)  unsigned char value : モード 0: OFF 1: ON
解説	パワーセーブモードを設定する
戻値	0: 設定成功 1: 設定失敗 (パラメータエラー)
補足	端末下りIDに対応するグループとグループ0 だけ無線受信系ブロックの電源をONする

## get\_kxs40

---

要約 unsigned char get\_kxs40(void)

解説 パワーセーブモード 設定値を獲得する

戻値 モード  
0: OFF  
1: ON

補足

例:

```
#include "kme_lib.h"           /* ヘッダファイルをインクルードする. */

void main(void)
{
    unsigned char mode;

    set_kxs40(1);              /* パワーセーブモードを設定する */
    mode = get_kxs40();        /* パワーセーブモードを獲得する */
}
```

## set\_kxs41

---

要約	int set_kxs41(unsigned char value)  unsigned char value: モード 0: CTSが非アクティブで送信中止 1: フロー制御無し
解説	インポートフロー制御を設定する
戻値	0: 設定成功 1: 設定失敗 (パラメータエラー) 2: 排他エラー : KXS44(0) / KXS41(1).
補足	DTEからパケットが送られてくるのを中止させるためには、CTSを非アクティブにする

## get\_kxs41

---

要約	unsigned char get_kxs41(void)
解説	インポートフロー制御の設定を取得する
戻値	0: CTSが非アクティブで送信中止 1: フロー制御無し
補足	

## set\_kxs42

---

要約	int set_kxs42(unsigned char value)
	unsigned char value: モード 0: RTSがアクティブで送信中止 1: フロー制御なし
解説	アウトバウンドフロー制御を設定する
戻値	0: 設定成功 1: 設定失敗 (パラメータエラー) 2: 排他エラー: KXS44(0) / KXS42(1).
補足	RTSがアクティブの時、端末はDTEへ送信しない

## get\_kxs42

---

要約	unsigned char get_kxs42(void)
解説	アウトバウンドフロー制御設定値を獲得する
戻値	0: RTSがアクティブで送信中止. 1: フロー制御なし
補足	

## set\_kxs43

---

要約	int set_kxs43(SERIAL_PAR *serial)  SERIAL_PAR *serial unsigned char baud : ポート (0 :300, 1: 600, 2:1200, 3:2400, 4:4800, 5:9600bps) unsigned char parity:パリティ(0:Even, 1:Odd, 2:None) unsigned char stop_bit:ストップビット(1:1bit, 2:2bit) unsigned char data_bit:データ長(7:7bit, 8:8bit)
解説	メインポートの RS232C 通信モードを設定する
戻値	0: 設定成功 1: 設定失敗 (パラメータエラー)
補足	RS232Cのパラメータの選択後、有効になるまで数秒間お待ちください。 サブポートは set_kxs43_port 関数を使用してください。

## get\_kxs43

---

要約	void get_kxs43(SERIAL_PAR *serial)  SERIAL_PAR *serial unsigned char baud : ポート (0 :300, 1: 600, 2:1200, 3:2400, 4:4800, 5:9600bps) unsigned char parity:パリティ(0:Even, 1:Odd, 2:None) unsigned char stop_bit:ストップビット(1:1bit, 2:2bit) unsigned char data_bit:データ長(7:7bit, 8:8bit)
解説	メインポートの RS232C 通信モード設定値を獲得する
戻値	なし
補足	サブポートは get_kxs43_port 関数を使用してください。

例:

```
#include "kme_lib.h"                /* ヘッダファイルをインクルードする。*/

void main(void)
{
    SERIAL_PAR seral;
    get_kxs43(&seral);              /* RS232C通信モードを獲得する */
}
```

## set\_kxs43\_port

```

要約          int set_kxs43(unsigned char port, SERIAL_PAR *serial)

              unsigned char port : port
              0: シリアルポート1 (メイン)
              1: シリアルポート2 (サブ)

              SERIAL_PAR *serial
              unsigned char baud :     ポートレート (0 :300, 1: 600, 2:1200,
                                          3:2400, 4:4800, 5:9600bps)
              unsigned char parity:    パリティ(0:Even, 1:Odd, 2:None)
              unsigned char stop_bit:  ストップビット(1:1bit, 2:2bit)
              unsigned char data_bit:  データ長(7:7bit, 8:8bit)

解説          RS232C 通信モードを設定する

戻値          0: 設定成功
              1: 設定失敗 (パラメータエラー)

補足          RS232Cのパラメータの選択後、有効になるまで数秒間お待ちください。
    
```

## get\_kxs43\_port

```

要約          void get_kxs43(unsigned char port, SERIAL_PAR *serial)

              unsigned char port : port
              0: シリアルポート1 (メイン)
              1: シリアルポート2 (サブ)

              SERIAL_PAR *serial
              unsigned char baud :     ポートレート (0 :300, 1: 600, 2:1200,
                                          3:2400, 4:4800, 5:9600bps)
              unsigned char parity:    パリティ(0:Even, 1:Odd, 2:None)
              unsigned char stop_bit:  ストップビット(1:1bit, 2:2bit)
              unsigned char data_bit:  データ長(7:7bit, 8:8bit)

解説          RS232C 通信モード設定値を獲得する

戻値          なし

補足
    
```

例:

```
#include "kme_lib.h"           /* ヘッダファイルをインクルードする。*/
```

```
void main(void)
```

```
{  
    SERIAL_PAR seral;  
    get_kxs43_port(0,&serial);          /* RS232C通信ポートを獲得する */  
}
```

## set\_kxs44

要約 int set\_kxs44(unsigned char value)

unsigned char value(0-2)

- 0: 半2重
- 1: 全2重
- 2: 受信のみ

解説 RS232C 通信モードを設定する

戻値 0: 設定成功  
 1: 設定失敗 (パラメータエラー)  
 2: 排他エラー: KXS41(1) / KXS44(0),  
 KXS42(1) / KXS44(0)

補足

## get\_kxs44

要約 unsigned char get\_kxs44(void)

解説 RS232C通信モード設定値を獲得する

戻値 0: 半2重  
 1: 全2重  
 2: 受信のみ

補足

例:

```
#include "kme_lib.h"                /* ヘッダファイルをインクルードする。*/

void main(void)
{
    set_kxs44(1);    /* RS232C通信モードを設定する */
    set_kxs41(0);    /* インバウンドフロー制御モードを設定する */
    set_kxs42(0);    /* アウトバウンドフロー制御モードを設定する */
}
```

## set\_kxs45

---

要約 int set\_kxs45(unsigned char value)

unsigned char value : モード (0, 1)

- 0: インバウンドキューが満杯時、DTEからの新しいメッセージは受信されない
- 1: インバウンドキューが満杯時、DTE から新しいメッセージが受信されると、インバウンドキューの古いメッセージから削除されて、キューに格納される

解説 インバウンドメッセージトリートメントを設定する

戻値 0: 設定成功  
1: 設定失敗 (パラメータエラー)

補足

## get\_kxs45

---

要約 unsigned char get\_kxs45(void)

解説 インバウンドメッセージトリートメント設定値を獲得する

戻値 0: インバウンドキューが満杯時、DTEからの新しいメッセージは受信されない  
1: インバウンドキューが満杯時、DTE から新しいメッセージが受信されると、インバウンドキューの古いメッセージから削除されて、キューに格納される

補足



## set\_kxs46

---

要約 int set\_kxs46(unsigned char value)

unsigned char value : モード (0, 1)

- 0: アウトバウンドキューが満杯時、GCCからの新しいメッセージは受信されない
- 1: アウトバウンドキューが満杯時、GCC から新しいメッセージが受信されると、アウトバウンドキューの古いメッセージから削除されて、キューに格納される

解説 アウトバウンドメッセージトリートメントを設定する

戻値 0: 設定成功  
1: 設定失敗 (パラメーター)

補足

## get\_kxs46

---

要約 unsigned char get\_kxs46(void)

解説 アウトバウンドメッセージトリートメント設定値を獲得する

戻値 0: アウトバウンドキューが満杯時、GCCからの新しいメッセージは受信されない  
1: アウトバウンドキューが満杯時、GCC から新しいメッセージが受信されると、アウトバウンドキューの古いメッセージから削除されて、キューに格納される

補足

## set\_kxs47

---

要約	int set_kxs47(int value)
	int value : モード (0, 1) 0: GCCへのメッセージ送信失敗したらキューから削除する 1: GCCへのメッセージ送信失敗してもキューから削除しない
解説	メッセージリキューオプションを設定する
戻値	0: 設定成功 1: 設定失敗 (パラメータエラー)
補足	

## get\_kxs47

---

要約	int get_kxs47(void)
解説	メッセージリキューオプション設定値を取得する
戻値	0: GCCへのメッセージ送信失敗したらキューから削除する 1: GCCへのメッセージ送信失敗してもキューから削除しない
補足	

## set\_kxs48

---

要約 unsigned char set\_kxs48(int size)

int size:

	インバウンドキュー	アウトバウンドキュー
1:	1K	7K
2:	2K	6K
3:	3K	5K
4:	4K	4K
5:	5K	3K
6:	6K	2K
7:	7K	1K

解説 インバウンド・アウトバウンドキューのサイズを設定する

戻値 0: 設定成功  
1: 設定失敗 (パラメーター)

補足 インバウンド・アウトバウンドキュー合わせて8Kbyteあり、それを上記のように分割して使用する。

## get\_kxs48

---

要約 unsigned char get\_kxs48(void)

解説 インバウンド・アウトバウンドキューサイズ設定値を獲得する

戻値	インバウンドキュー	アウトバウンドキュー
1:	1K	7K
2:	2K	6K
3:	3K	5K
4:	4K	4K
5:	5K	3K
6:	6K	2K
7:	7K	1K

補足 インバウンド・アウトバウンドキュー合わせて8Kbyteあり、それを上記のように分割して使用する。

## set\_kxs50

---

要約	int set_kxs50(unsigned long value)  unsigned char value : コード (0 – 9999)
解説	ピコードを設定する
戻値	0: 設定成功 1: 設定失敗 (パラメーター)
補足	ピコードとは、端末 - GCC 間通信におけるパスワードである

## get\_kxs50

---

要約	unsigned long get_kxs50(void)
解説	ピコード設定値を獲得する
戻値	コード (0 – 9999)
補足	

## set\_kxs51

---

要約           int set\_kxs51(unsigned char value)

                  unsigned char value: モード (0, 1)  
                   0: 何もしない  
                   1: 衛星に対しグローバルリンクをかける

解説           自動グローバルリンクを設定する

戻値           0: 設定成功  
                   1: 設定失敗 (パラメータエラー)

補足

## get\_kxs51

---

要約           unsigned char get\_kxs51(void)

解説           自動グローバルリンク設定値を獲得する

戻値           0: 何もしない  
                   1: 衛星に対しグローバルリンクをかける

補足

## set\_kxs52

---

要約           int set\_kxs52(unsigned char value)  
  
                  unsigned char value : 測地系 ( 0 - 100 )

解説           GPS 測地系を設定する

戻値           0: 設定成功  
                  1: 設定失敗 (パラメータエラー)

補足

## get\_kxs52

---

要約           unsigned char get\_kxs52(void)

解説           GPS測地系設定値を獲得する

戻値           測地系 ( 0 - 100 ).

補足

## set\_kxs53

---

要約           int set\_kxs53(unsigned char value)

                  unsigned char value : RTS論理  
                  0: LOW : アクティブ (ORBCOMM 仕様)  
                  1: HI : アクティブ (一般仕様)

解説           RTS 論理仕様を設定する

戻値           0: 設定成功  
                  1: 設定失敗 (パラメータエラー)

補足

## get\_kxs53

---

要約           unsigned char get\_kxs53(void)

解説           RTS論理仕様設定値を獲得する

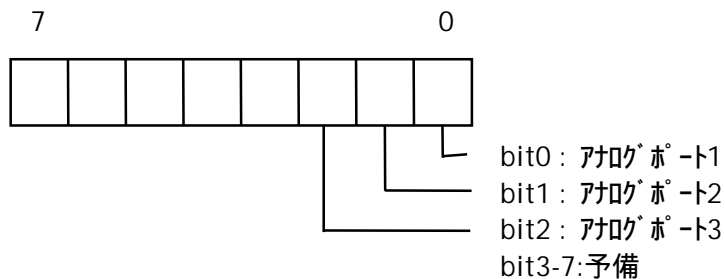
戻値           RTS論理  
                  0: LOW : アクティブ (ORBCOMM 仕様)  
                  1: HI : アクティブ (一般仕様)

補足

## set\_kxs55

要約 int set\_kxs55(char value)

char value: 送信するアナログポート番号



解説 KXB モジュールで送信するアナログポート番号を設定する

戻値 0: 設定成功  
 1: 設定失敗 (パラメータエラー)

補足

## get\_kxs55

要約 char get\_kxs55(void)

解説 KXBモジュールで送信するアナログポート番号設定値を獲得する

戻値 アナログポート番号

補足



## set\_kxs56

---

要約           int set\_kxs56(double distance, unsigned char unit)

int distance : 移動距離(0.1 – 5000)

unsigned char unit: 単位(0:Km, 1:mile, 2:NM)

解説           KXB コマンド 移動距離検知の移動距離を設定する

戻値           0: 設定成功  
1: 設定失敗 (パラメータエラー)

補足

## get\_kxs56

---

要約           void get\_kxs56(double \*distance, unsigned char \*unit)

int \*distance : 移動距離(0.1 - 5000)

unsigned char \*unit: 単位(0:Km, 1:mile, 2:NM)

解説           KXB コマンド 移動距離検知の移動距離設定値を獲得する

戻値           なし

補足

## set\_kxs57

要約 int set\_kxs57(double lat, double lon,  
double radius, unsigned char unit)

double lat: 緯度 (-90.0000 - +90.0000)  
double lon: 経度(-180.0000 - +180.0000)  
int radius: 半径(0.1 - 5000)  
unsigned char unit: 単位(0:Km, 1:mile, 2:NM)

解説 KXB マウントエリア検知のエリアを設定する

戻値 0: 設定成功  
1: 設定失敗 (パラメータエラー)

補足

## get\_kxs57

要約 void get\_kxs57(double \*lat, double \*lon, double  
\*radius, unsigned char \*unit)

double \*lat: 緯度 (-90.0000 - +90.0000)  
double \*lon: 経度(-180.0000 - +180.0000)  
int \*radius: 半径(0.1 - 5000)  
unsigned char \*unit: 単位(0:Km, 1:mile, 2:NM)

解説 KXB マウントエリア検知のエリア設定値を獲得する

戻値 なし

補足

## set\_kxs58

要約 int set\_kxs58(unsigned char speed, unsigned char unit)

int speed: 速度(1 - 255)  
 unsigned char unit: 単位(0 -2)  
 0: Km/h  
 1: mile/h  
 2: knots

解説 KXB モント 速度検知の速度を設定する

戻値 0: 設定成功  
 1: 設定失敗 (パラメータ)

補足

## get\_kxs58

要約 void get\_kxs58(unsigned char \*speed, unsigned char \*unit)

int \*speed: 速度(1 - 255)  
 unsigned char \*unit : 単位(0:Km/h, 1:mile/h, Knots)

解説 KXB モント 速度検知の速度設定値を獲得する

戻値 なし

補足

## set\_kxs60

要約 int set\_kxs60(unsigned char value)

unsigned char value : タイプ (0-5)

- 0: テキスト形式
- 1: ハーフイリ形式(非短縮)
- 2: ハーフイリ形式(短縮)
- 3: 位置は測位ポートで送信、その他はテキスト(0と同じ)
- 4: 位置は測位ポートで送信、その他はハーフイリ(1と同じ)
- 5: 位置は測位ポートで送信、その他はハーフイリ短縮(2と同じ)

解説 KXBコマンドで送信するデータフォーマットを設定する

戻値 0: 設定成功  
 1: 設定失敗 (パラメータエラー)  
 2: 排他エラー: KXS60(2-5) / KXS74(1)

補足 測位ポートはグローバルプログラム自動変換は行われない。  
 次の設定は無効

- ・ KXS74(1)とKXS60(2-5) 一括送信と短縮モードの併用不可
- ・ KXB位置情報送信時の相手アドレス(O/R)4以上とKXS60(3-5)

## get\_kxs60

要約 unsigned char get\_kxs60(void)

解説 KXBコマンドで送信するデータフォーマット設定値を獲得する

戻値 タイプ (0-5)

- 0: テキスト形式
- 1: ハーフイリ形式(非短縮)
- 2: ハーフイリ形式(短縮)
- 3: 位置は測位ポートで送信、その他はテキスト(0と同じ)
- 4: 位置は測位ポートで送信、その他はハーフイリ(1と同じ)
- 5: 位置は測位ポートで送信、その他はハーフイリ短縮(2と同じ)

補足

## set\_kxs61

要約	int set_kxs61(unsigned char value)
	<p>unsigned char value: モード (0-3)</p> <p>0: 電力節減モード OFF</p> <p>1: メインポートのみ電力節減モード ON</p> <p>2: サポートのみ電力節減モード ON</p> <p>3: メイン,サポート双方電力節減モード ON</p>
解説	RS232 ドライバ-パワーセーブモードを設定する
戻値	<p>0: 設定成功</p> <p>1: 設定失敗 (パラメタエラー)</p>
補足	パワーセーブモード ONにすると、端末がDTEと通信していないときは、RS232 ドライバ-は自動的にパワーセーブモードになる

## get\_kxs61

要約	unsigned char get_kxs61(void)
解説	RS232 ドライバ-パワーセーブモード設定値を獲得する
戻値	<p>モード (0-3)</p> <p>0: 電力節減モードOFF</p> <p>1: メインポートのみ電力節減モードON</p> <p>2: サポートのみ電力節減モードON</p> <p>3: メイン,サポート双方電力節減モードON</p>
補足	

## set\_kxs63

---

要約 int set\_kxs63(TIME\_WINDOW \*time\_win)

TIME\_WINDOW \*time\_win  
 unsigned char on\_off : オフ (0, 1)  
 0= OFF  
 1= ON  
 char lh: タイムゾーン (-11 - + 12) [時間]  
 unsigned char lm : タイムゾーン(0, 30) [分]  
 unsigned char th1 :開始時間(0 - 23) [時間]  
 unsigned char tm1:開始時間(0 - 59) [分]  
 unsigned char th2:終了時間(0 - 23) [時間]  
 unsigned char tm2:終了時間(0 - 59) [分]

解説 タイムゾーンを設定する

戻値 0: 設定成功  
 1: 設定失敗 (パラメータエラー)

補足 タイムゾーンが設定されると、開始 - 終了時間以外は端末はパワーオフする。

## get\_kxs63

---

要約 int get\_kxs63(TIME\_WINDOW \*time\_win)

TIME\_WINDOW \*time\_win  
 unsigned char on\_off : オフ (0, 1)  
 0= OFF  
 1= ON  
 char lh: タイムゾーン (-11 - + 12) [時間]  
 unsigned char lm : タイムゾーン(0, 30) [分]  
 unsigned char th1 :開始時間(0 - 23) [時間]  
 unsigned char tm1:開始時間(0 - 59) [分]  
 unsigned char th2:終了時間(0 - 23) [時間]  
 unsigned char tm2:終了時間(0 - 59) [分]

解説 タイムゾーン設定値を獲得する

戻値 なし

補足

## set\_kxs64

---

要約	int set_kxs64(unsigned char value)
	unsigned char value : クイックワータウンモード (0,1) 0: モードオ 1: データ送信が成功して、送信待ちデータがなければ即ハワータウン
解説	KXB コマンドでのクイックワータウンモードを設定する
戻値	0: 設定成功 1: 設定失敗 (パラメータエラー)
補足	

## get\_kxs64

---

要約	unsigned char get_kxs64(void)
解説	KXB コマンドでのクイックワータウンモード設定値を獲得する
戻値	クイックワータウンモード (0,1) 0: モードオ 1: データ送信が成功して、送信待ちデータがなければ即ハワータウン
補足	

## set\_kxs67

---

要約 int set\_kxs67(unsigned char value)

unsigned char value : モード (0,1)

0: 端末電源ON時に、ユーザーアプリを起動しない

1: 端末電源 ON 時に、ユーザーアプリを起動する

解説 端末電源 ON 時に、ユーザーアプリを起動するかどうかを設定する

戻値 0: 設定成功  
1: 設定失敗 (パラメータエラー)

補足

## get\_kxs67

---

要約 unsigned char get\_kxs67(void)

解説 端末電源ON時の、ユーザーアプリ起動設定値を獲得する

戻値 モード (0,1)

0: 端末電源ON時に、ユーザーアプリを起動しない

1: 端末電源 ON 時に、ユーザーアプリを起動する

補足



## set\_kxs68

---

要約           int set\_kxs68(unsigned char value)

unsigned char value : タイプ (0 – 1)

0: メインシステムで処理

1: ユーザーアプリケーションで処理

解説           RS232C 受信データの処理ルートを設定する

戻値           0: 設定成功

1: 設定失敗 (パラメーター)

補足

## get\_kxs68

---

要約           unsigned char get\_kxs68(void)

解説           RS232C 受信データの処理ルート設定値を獲得する

戻値           タイプ (0 – 1).

0: メインシステムで処理

1: ユーザーアプリケーションで処理

補足

## set\_kxs69

---

要約	int set_kxs69(unsigned char value)
	unsigned char value : タイプ (0 – 2) 0: システムで処理 1: ユーザーアプリケーションで処理 2: 両方で処理
解説	GCC からの受信データの処理ルートを設定する
戻値	0: 設定成功 1: 設定失敗 (パラメータエラー) 2: 排他エラー: KXS69(1or2) / KXS68(1)
補足	

## get\_kxs69

---

要約	unsigned char get_kxs69(void)
解説	GCCからの受信データの処理ルート設定値を取得する
戻値	タイプ (0 – 2) 0: システムで処理 1: ユーザーアプリケーションで処理 2: 両方で処理
補足	

## set\_kxs70

---

要約                   int set\_kxs70(unsigned char value)

unsigned char value : モード  
 0: テンタックモード OFF  
 1-9: テンタックモード ON

解説                   ユーザーアプリケーションのテンタックモードを設定する

戻値                   0: 設定成功  
 1: 設定失敗 (パラメータエラー)

補足

## get\_kxs70

---

要約                   unsigned char get\_kxs70(void)

解説                   ユーザーアプリケーションのテンタックモード設定値を取得する

戻値                   モード  
 0: テンタックモード OFF  
 1-9: テンタックモード ON

補足

## set\_kxs71

---

要約           int set\_kxs71(char \*str)

unsigned char \*str: (0-9, A-Z) \* 4 文字

解説           GCC からのコマンドリモートセッティングでのセットアップ ID を設定する

戻値           0: 設定成功  
               1: 設定失敗 (パラメータエラー)

補足

## get\_kxs71

---

要約           char \*get\_kxs71(void)

解説           リモートセッティングでのセットアップ ID 設定値を獲得する

戻値           セットアップ ID へのポインタ

補足

## set\_kxs72

---

要約           int set\_kxs72(unsigned char value)

unsigned char value : モード (0, 1)

0: 応答メッセージなし

1: 応答メッセージをホストに返す

解説           ホストからのコマンドリセットでその応答メッセージを返すかどうかを設定する

戻値           0: 設定成功  
1: 設定失敗 (パラメータエラー)

補足

## get\_kxs72

---

要約           unsigned char get\_kxs72(void)

解説           GCCからのコマンドリセットでの応答レスポンス設定値を獲得する

戻値           モード (0, 1)  
0: 応答メッセージなし  
1: 応答メッセージをホストに返す

補足

## set\_kxs74

要約	int set_kxs74(unsigned char value)
	unsigned char value : モード 0: 個別に送信 1: まとめて送信
解説	KXB コマンドによる無条件送信で位置情報を含む 2 つ以上のデータを送信する場合の送信方法を設定する
戻値	0: 設定成功 1: 設定失敗 (パラメータ)
補足	0: I/O ポートやバッテリー情報など即送信できるデータは先に送信し、位置/速度/方向情報など送信するまでに時間のかかるデータは後で送信する。 1: 送信データサイズを減らすため、送信するのに時間のかかる位置/速度/方向情報などの時間に合わせて I/O ポートやバッテリー情報をまとめて送信する。もし時間内に測位ができなければ、I/O ポートやバッテリー情報だけを送信する。

## get\_kxs74

要約	void get_kxs74(void)
解説	KXBコマンドによる無条件送信で位置情報を含む2つ以上のデータを送信する場合の送信方法を獲得する
戻値	モード 0: 個別に送信 1: まとめて送信
補足	

## set\_kxs75

---

要約 int set\_kxs75(unsigned char value)

unsigned char value : モード  
 0: 自動でメッセージがグローバルラムに変換する  
 1: 自動変換しない

解説 KXB コマンドで生成されるメッセージを衛星がグローバルラム衛星ならグローバルラムに自動的に変換するモードを設定する

戻値 0: 設定成功  
 1: 設定失敗 (パラメータエラー)

補足

## get\_kxs75

---

要約 unsigned char get\_kxs75(void)

解説 KXBコマンドで生成されるメッセージの自動グローバルラム変換モード設定値を獲得する

戻値 モード  
 0: 自動でメッセージがグローバルラムに変換する  
 1: 自動変換しない

補足

## set\_kxs77

---

要約           int set\_kxs77(unsigned char value)

unsigned char value : モード  
 0: 通常 (O/Rアドレス、サブシフト、メッセージ本体すべてを送信)  
 1: メッセージ本体のみを DTE に送信

解説           バイトモード時、ホストからの受信データのメッセージ本体のみをDTEに送信するモードに設定する

戻値           0: 設定成功  
 1: 設定失敗 (パラメータエラー)

補足

## get\_kxs77

---

要約           unsigned char get\_kxs77(void)

解説           バイトモード時、ホストからの受信データのメッセージ本体のみをDTEに送信するモードの設定値を取得する

戻値           モード  
 0: 通常 (O/Rアドレス、サブシフト、メッセージ本体すべてを送信)  
 1: メッセージ本体のみを DTE に送信

補足



## set\_kxs78

---

要約           int set\_kxs78(unsigned char value)

unsigned char value : モード  
 0: 何も送信しない  
 1: インバウンドバッファ満杯メッセージを送信する

解説           バイトモード時、インバウンドキューが満杯で指定バイト数受信できない時にバッファフルメッセージをDTEに送信するように設定する

戻値           0: 設定成功  
 1: 設定失敗 (パラメータエラー)

補足

## get\_kxs78

---

要約           unsigned char get\_kxs78(void)

解説           バイトモード時、インバウンドキューが満杯で指定バイト数受信できない時にバッファフルメッセージをDTEに送信する設定を獲得する

戻値           モード  
 0: 何も送信しない  
 1: インバウンドバッファ満杯メッセージを送信する

補足

## set\_kxs79

---

要約           int set\_kxs79(int time)  
  
                  int range: KXB 検知時間 (5 – 20) [分]

解説            KXBモジュールの検知時間を設定する

戻値            0: 設定成功  
                  1: 設定失敗 (パラメータエラー)

補足

## get\_kxs79

---

要約            int get\_kxs79(void)

解説            KXB モジュールの検知時間設定値を獲得する

戻値            検知時間 (5 – 20分)

補足

## set\_kxs80

---

要約	int set_kxs80(int sw)
	int sw : アウトバウンドグローバルラミングケットのフォーマット 0: ノーマルグローバルラミング 1: エンハストグローバルラミング
解説	アウトバウンドグローバルラミングケットのフォーマットを設定する
戻値	0: 設定成功 1: 設定失敗 (パラメータエラー)
補足	グローバルラミングケットには、ノーマルグローバルラミングとエンハストグローバルラミングがあります。エンハストグローバルラミングケットの時にメッセージ本体のみ取り出す場合は、本設定を1にすると共にKXS77=1と設定してください。

## get\_kxs80

---

要約	int get_kxs80(void)
解説	アウトバウンドグローバルラミングケットのフォーマットを獲得する
戻値	アウトバウンドグローバルラミングケットのフォーマット 0= ノーマルグローバルラミング 1= エンハストグローバルラミング
補足	

## set\_kxs81

---

要約 int set\_kxs81( unsigned long int time )

unsigned long int time: available time  
0 – 3932100[秒]

解説 アウトメッセージの重複受信防止のガードタイマーを設定する

戻値 0: 設定成功  
1: 設定失敗 (パラメーター)

補足

## get\_kxs81

---

要約 unsigned long int get\_kxs81( void )

解説 アウトメッセージの重複受信防止のガードタイマー値設定値を獲得する

戻値 Guard time (0 – 3932100[秒])

補足

## set\_kxs82

---

要約            int set\_kxs82( int time )

int time: guard time  
0 – 15300[秒]

解説            アウトバンドグローバルラムの重複受信防止のガードタイマーを設定する

戻値            0: 設定成功  
1: 設定失敗 (パラメーター)

補足

## get\_kxs82

---

要約            int get\_kxs82( void )

解説            アウトバンドグローバルラムの重複受信防止のガードタイマー値設定値を獲得する

戻値            Guard time (0 – 15300 [秒])

補足

## set\_kxs83

---

要約           int set\_kxs83( int logic )

                  int logic : level of output digital port at reset  
                  0: LOW  
                  1: HIGH  
                  2: HOLD(電源断直前の状態)

解説           デジタル出力ポートのデフォルト値を設定する

戻値           0: 設定成功  
                  1: 設定失敗 (パラメータエラー)

補足

## get\_kxs83

---

要約           int get\_kxs83( void )

解説           デジタル出力ポートのデフォルト設定値を獲得する

戻値           0: LOW  
                  1: HIGH  
                  2: HOLD

補足

## set\_kxs84

要約	int set_kxs84( int mode )  int mode: roaming mode 0: ロミング OFF(KXS01 で設定した GCC とのみ通信可) 1: ロミング ON (KXS01 および KXS85 で設定した GCC と通信可) 2: ロミング ON (全 GCC と通信可)
解説	ロミング モード を設定する
戻値	0: 設定成功 1: 設定失敗 (パラメータエラー)
補足	

## get\_kxs84

要約	int get_kxs84( void )
解説	ロミング モード の設定値を獲得する
戻値	roaming mode 0: ロミング OFF(KXS01 で設定した GCC とのみ通信可) 1: ロミング ON (KXS01 および KXS85 で設定した GCC と通信可) 2: ロミング ON (全 GCC と通信可)
補足	

## set\_kxs85

---

要約           int set\_kxs85( int ind, unsigned char gcc )

int ind: block No(0 –11)  
 unsigned char gcc: GCC\_ID(0 – 255)

解説           □-ミグ 対象GCCを設定する

戻値           0: 設定成功  
 1: 設定失敗 (パラメータエラー)

補足

## get\_kxs85

---

要約           unsigned char get\_kxs85( int ind )  
 int ind: block No(0 –11)

解説           □-ミグ 対象 GCC の設定を獲得する

戻値           GCC ID(0 – 255)

補足



## set\_kxs86

---

要約           int set\_kxs86( char \*array )

char \*array : pointer of day information.  
 <day information>  
 0 – 6 : Sunday – Monday  
       '0'(0x30) : Not work  
       '1'(0x31) : Work

解説           KXBマウント動作の曜日指定を行う

戻値           0: 設定成功  
               1: 設定失敗 (パラメータエラー)

補足

## get\_kxs86

---

要約           char \*get\_kxs86( void )

解説           KXBマウント動作の曜日指定値を獲得する

戻値           pointer of day information.  
               <day information>  
               0 – 6 : Sunday – Monday  
               '0'(0x30) : Not work  
               '1'(0x31) : Work

補足

Example:

```
#include "kme_lib.h"

void main(void)
{
    char day_info[8];

    /* Set the worked day to Monday and Wednesday. */
    day_info[0] = '0'; /* Sunday */
    day_info[1] = '1'; /* Monday */
    day_info[2] = '0'; /* Tuesday */
    day_info[3] = '1'; /* Wednesday */
    day_info[4] = '0'; /* Thursday */
    day_info[5] = '0'; /* Friday */
    day_info[6] = '0'; /* Saturday */
    day_info[7] = '\0'; /* Null stop */

    set_kxs86(day_info);
}
```

## set\_kxs87

要約	int set_kxs87( int pdop_thresh)  int pdop_thresh : PDOP 値 ( 1 ~ 1 0 )
解説	GPSの測位精度は、測位に使用した3つないし4つの衛星の散ばり具合に左右される傾向がある。PDOPはその衛星の散ばり具合を表す値で、衛星が広範囲に散ばっているとこの値は小さくなり、精度のよい位置が得られやすくなる。 端末は、GPSの算出した位置情報と同時に得られるPDOP値が設定値以下の時だけ、位置情報の獲得処理を行う。そのPDOPのしきい値を設定する。
戻値	0: 設定成功 1: 設定失敗 (パラメータエラー)
補足	set_pdop_th()と同じ処理を行う。  set_kxs87 を 5 以下に設定すると 2 次元測位解を採用しません。精度は上がりますが測位時間が長くなることがあります。(ファーム Ver1.20 以降)

## get\_kxs87

要約	int get_kxs87( void)
解説	GPS位置情報をフィルタする時のPDOPのしきい値を獲得する。
戻値	PDOP 値 ( 1 ~ 1 0 )
補足	get_pdop_th()と同じ処理を行う。

## set\_kxs90

要約 int set\_kxs90(unsigned char value, unsigned char mode)

unsigned char value : resolution of analog value

0: 8bit

1: 10bit

unsigned char mode : input mode

0: 0-5V

1: 0-15V

2: 4-20mA

解説 アナログ入力のAD変換解像度と入力モードを設定する

戻値 0: 設定成功  
1: 設定失敗 (パラメータエラー)

補足

## get\_kxs90

要約 void get\_kxs90(unsigned char \*value, unsigned char \*mode)

unsigned char \*value : resolution of analog value

0= 8bit

1= 10bit

unsigned char \*mode : input mode

0: 0-5V

1: 0-15V

2: 4-20mA

解説 アナログ入力のAD変換解像度と入力モード設定値を取得する

戻値 なし

補足

## set\_kxs91

---

要約           int set\_kxs91(unsigned char value)

int value :  
 0: 出力しない  
 1: 出力する

解説           オーブコム衛星の軌道情報を受信した場合、DTEへの出力設定

戻値           0: 設定成功  
 1: 設定失敗 (パラメータエラー)

補足

## get\_kxs91

---

要約           unsigned char get\_kxs91(void)

解説           オーブコム衛星の軌道情報の DTE 出力設定値を獲得する

戻値           0: 出力しない  
 1: 出力する

補足

## set\_kxs94

---

要約           int set\_kxs94(unsigned char port\_no, unsigned char direction)

unsigned char port\_no : digital I/O port number (0-1)

0: DIO1

1: DIO2

unsigned char direction : port direction

0: 入力ポート

1: 出力ポート

解説           デジタル入出力ポートの入出力モードを設定する

戻値           0: 設定成功

1: 設定失敗 (パラメータエラー)

補足

## get\_kxs94

---

要約           int get\_kxs94(void)

解説           デジタル入出力ポートの入出力モード設定値を獲得する

戻値           bit0 : DIO1   (0:入力ポート、1:出力ポート)

bit1 : DIO2

補足

## set\_kxs95

要約	int set_kxs95(unsigned char sw)
	<p>int sw : Selection of</p> <p>0= 新シリアルインターフェイス(Ver.6.0以降)仕様書に準拠するStatus Code</p> <p>1= 旧シリアルインターフェイス(Ver.5.01 以前)仕様書に準拠する Status Code</p>
解説	プロトコルモードのLL ACK ステータスコードを選択する
戻値	<p>0: 設定成功</p> <p>1: 設定失敗 (パラメータエラー)</p>
補足	<p>新シリアルインターフェイス(Ver.6.0以降)仕様書に準拠する場合 ステータスコード (0-7)</p> <ul style="list-style-type: none"> <li>0:エラーなし</li> <li>1:バッファが利用不可 30 秒待って再び送信</li> <li>2:パケット拒否 チェックサム不正</li> <li>3:パケット拒否 パラメータ不正</li> <li>4:パケット拒否 キューの容量オーバー</li> <li>5:パケット拒否 フォーマット不正</li> <li>6:パケット拒否 異常パケットタイプ</li> <li>7:パケット拒否 パケットシーケンス番号の重複</li> </ul> <p>旧シリアルインターフェイス(Ver.5.01以前)仕様書に準拠する場合 ステータスコード (0-1)</p> <ul style="list-style-type: none"> <li>0:エラーなし</li> <li>1:バッファが利用不可 30秒待って再び送信</li> </ul>

## get\_kxs95

要約	unsigned char get_kxs95(void)
解説	プロトコルモードのLL ACK ステータスコード指定値を獲得する
戻値	<p>0= 新シリアルインターフェイス(Ver.6.0以降)仕様書に準拠するStatus Code</p> <p>1= 旧シリアルインターフェイス(Ver.5.01 以前)仕様書に準拠する Status Code</p>
補足	

## set\_kxs96

---

要約           int set\_kxs96(unsigned char port)

unsigned char port : port

0: シリアルポート1 (メイン)

1: シリアルポート2 (サブ)

解説           オブジェクトシリアル通信モードをポートするRS232Cポートを指定する

戻値           0: 設定成功  
1: 設定失敗 (パラメータエラー)

補足

## get\_kxs96

---

要約           int get\_kxs96(void)

解説           オブジェクトシリアル通信モードをポートする RS232C ポートの設定値を獲得する

戻値           0: シリアルポート1 (メイン)  
1: シリアルポート2 (サブ)

補足



## set\_kxd01

要約 int set\_kxd01(unsigned char m, unsigned char hi\_low)

unsigned char m: ポート (0:入出力ポート1, 1:入出力ポート2)

unsigned char hi\_low: レベル(0:LOW, 1:HI)

解説 デジタルポート出力を設定する

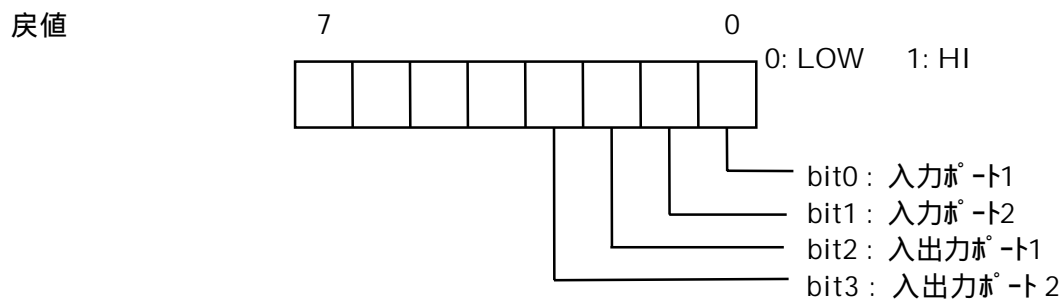
戻値 0: 設定成功  
1: 設定失敗 (パラメータエラー)  
2: 排他エラー

補足 出力設定は set\_kxs94 にて入出力ポートを出力モードにしていないと無効です。

## get\_kxd01

要約 unsigned char get\_kxd01(void)

解説 デジタルポート出力設定を取得する



補足 本関数にて入力ポートの情報も取得できます。

set\_kxd02 は呼ばれていません。

## get\_kxd02

要約 int get\_kxd02(unsigned char ch\_no, unsigned char \*ch)

unsigned char ch\_no : アンテナ番号 (0-7).

0: RSSI

1: アンテナ1

2: アンテナ2

3: アンテナ3

4- 7: 予備

unsigned char \*ch : アンテナ電圧(0 - 255)

解説 アンテナ入力値を取得する

戻値 0: 設定成功

1: アンテナ番号入力エラー

補足

## set\_kxm01

---

要約           int set\_kxm01(unsigned char \*str)

char \*str : 固定メッセージへのポインタ (最大 200 バイト)

解説           固定メッセージを設定する

戻値           0: 設定成功  
               1: 設定失敗 (パラメータエラー)  
               2: 排他エラー

補足

## get\_kxm01

---

要約           char \*get\_kxm01(void)

解説           固定メッセージ設定値を取得する

戻値           固定メッセージへのポインタ (最大 200 バイト)

補足

## set\_kxb01

要約 int set\_kxb01(KXB01\_CMND \*pkxb01\_cmd, int n)

KXB01\_CMND \*pkxb01\_cmd  
 int lh: タイムゾーン (-11 - +12) [時間]  
 int lm: タイムゾーン (00 or 30) [分]  
 int hh: 起動時間 (00 - 23) [時間]  
 int mm: 起動時間 (00 - 59) [分]  
 int c: 検知コード  
 int a: O/Rインテイクタ (1 - 8)  
 char d: 送信データ

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

bit0: 測位情報  
 bit1: ホート情報  
 bit2: 固定メッセージ  
 bit3: GCCへのホーリング  
 bit4: DTEへのホーリング  
 bit5: 予備  
 bit6: ユーザーアプリ起動  
 bit7: 予備

int n: 時刻番号 (1 - 6)

解説 時刻指定送信を設定する

戻値 0: 設定成功  
 1: 設定失敗 (パラメータエラー)  
 2: 排他エラー

補足

## get\_kxb01

要約 int get\_kxb01(KXB01\_CMND \*pkxb01\_cmd, int n)

KXB01\_CMND \*pkxb01\_cmd  
 int n: 時刻番号 (1 - 6)

解説 時刻指定送信 (KXB01コマンド) を獲得する

戻値 0: 未設定  
 1: 設定

補足

## set\_kxb02

要約 int set\_kxb02(KXB02\_CMND \*pkxb02\_cmdnd)

KXB02\_CMND \*pkxb02\_cmdnd  
 int lh: タイムゾーン (-11 - +12) [時間]  
 int IM: タイムゾーン (00 or 30) [分]  
 int hh: 起動時間 (00 - 23) [時間]  
 int mm: 起動時間 (00 - 59) [分]  
 int i: インターバル (0 - 44640)[分]  
 int c: 検知コード  
 int a: O/Rインデキータ (1 - 8)  
 char d: 送信データ

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

bit0: 測位情報  
 bit1: ホート情報  
 bit2: 固定メッセージ  
 bit3: GCCへのポートリンク  
 bit4: DTEへのポートリンク  
 bit5: 予備  
 bit6: ユーザーアプリ起動  
 bit7: アナログポート入力状態

解説 インターバル送信を設定する

戻値 0: 設定成功  
 1: 設定失敗 (パラメータエラー)  
 2: 排他エラー

補足

## get\_kxb02

要約 int get\_kxb02(KXB02\_CMND \*pkxb02\_cmdnd)

KXB02\_CMND \*pkxb02\_cmdnd

解説 インターバル送信 (KXB02コマンド) を獲得する

戻値 0: 未設定  
 1: 設定

補足

## set\_kxb05

要約 int set\_kxb05( KXB05\_CMND \*pkxb05\_cmnd, int p)

KXB05\_CMND \*pkxb05\_cmnd

int tr: Trigger (0:OFF, 1:LOW -> HI, 2:HI->LOW, 3:HI<->LOW)

int a: O/Rインデキータ(1 - 8)

char d: 送信データ

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

bit0: 測位情報

bit1: ホット情報

bit2: 固定メッセージ

bit3: GCCへのホットリンク

bit4: DTEへのホットリンク

bit5: 予備

bit6: ユーザーアプリ起動

bit7: アナログホット状態

int p: Digital IN Port No. (0, 1)

解説 I/O 状態変化報知を設定する

戻値 0: 設定成功  
1: 設定失敗 (パラメータエラー)  
2: 排他エラー

補足

## get\_kxb05

要約 int get\_kxb05( KXB05\_CMND \*pkxb05\_cmnd, int n)

KXB05\_CMND \*pkxb05\_cmnd

int n : Digital input port No. (0, 1)

解説 I/O状態変化報知の設定値を獲得する

戻値 0: 未設定  
1: 設定

補足

## set\_kxb00

---

要約           void set\_kxb00( void )

解説           KXB01,02,05 コマンドの設定を解除する

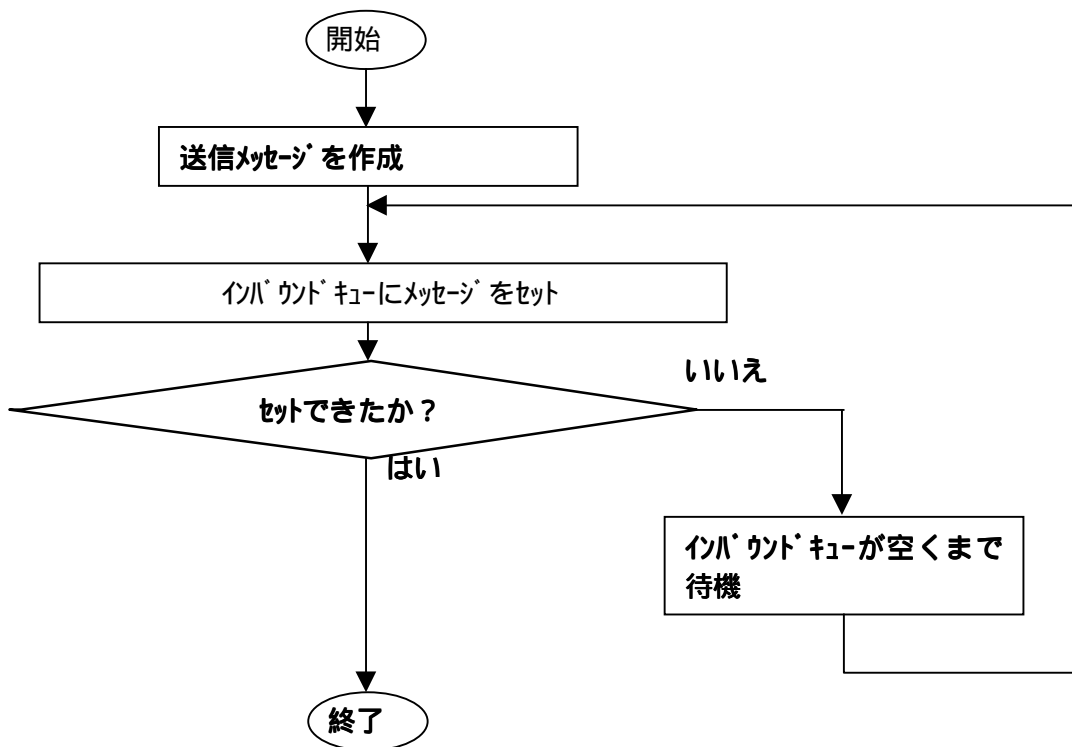
戻値           なし

補足

## 7. プログラミング ヒント

### 7.1. GCCへのメッセージ送信

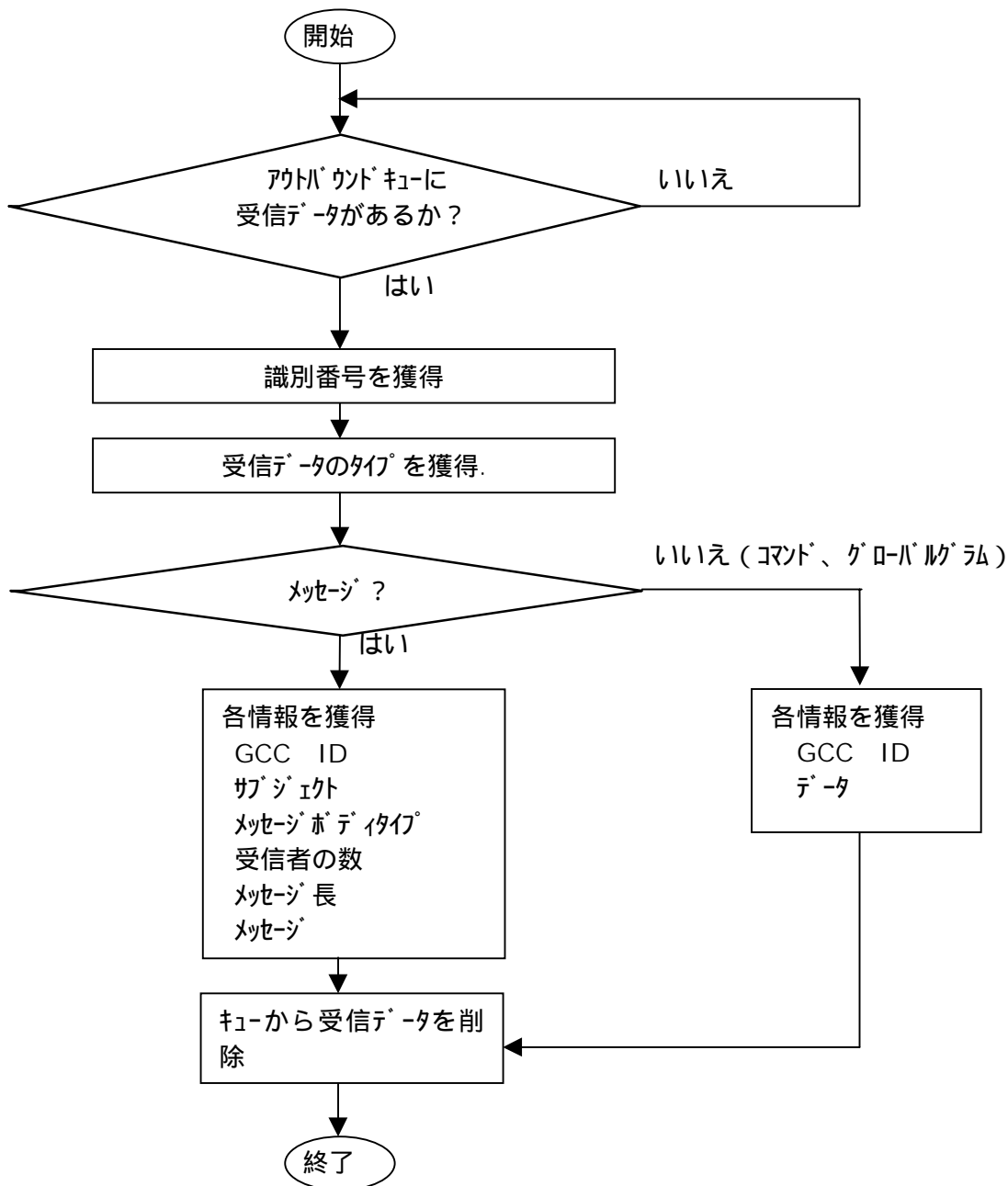
1. 送信メッセージを作成する
2. メッセージをインバウンドキューにセットする為に、“req\_send\_ib\_message()” をコールする。
3. メッセージがインバウンドキューにセットされたかをチェックする。  
 セットできなかった場合は、インバウンドキューからGCCへ送信されて領域が空くまで待機しないといけない。その後、再度 “req\_send\_ib\_message()”をコールする。
4. 衛星とリンクするとインバウンドキューにセットされたメッセージが自動的にGCCへ送信される。





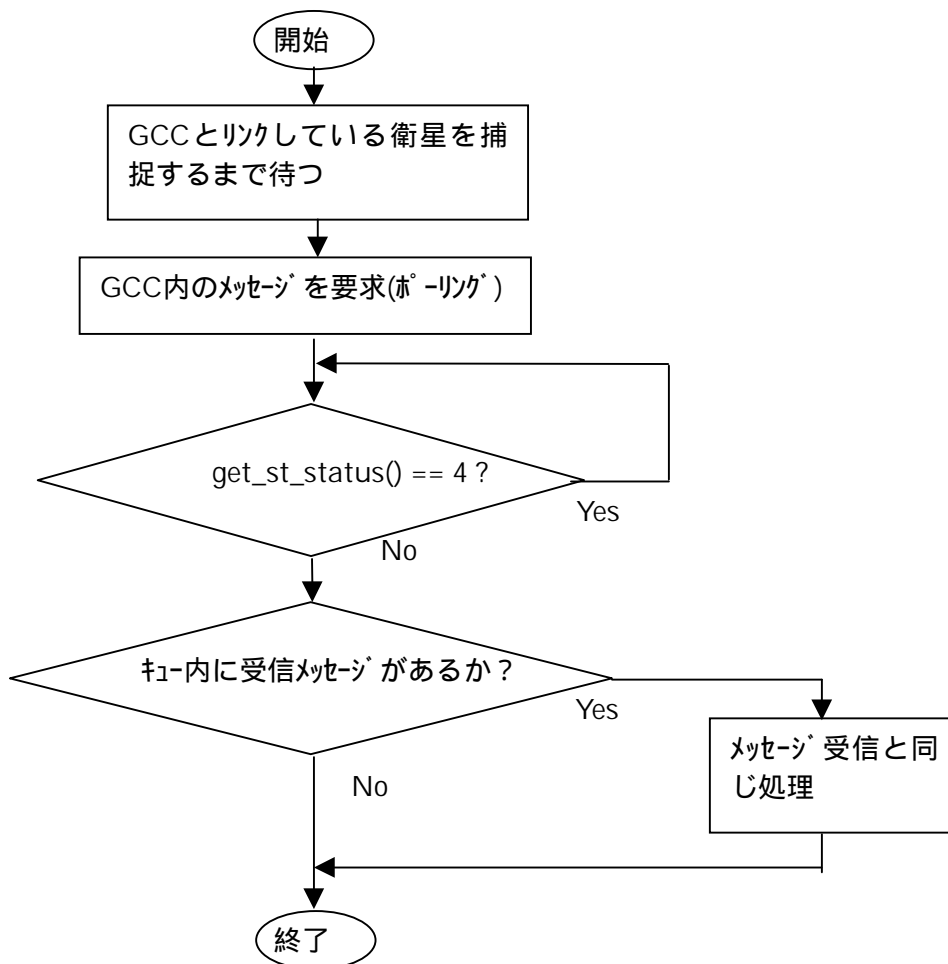
## 7.2. GCCからのメッセージ受信

1. "get\_unget\_ob\_id\_for\_user"をコールして、アウトバウンドキューに受信データがあるかを確認する。
2. もし受信データがあれば、そのタイプ(メッセージ/コメント)に応じた処理をする。
3. アウトバウンドキューからデータを消す



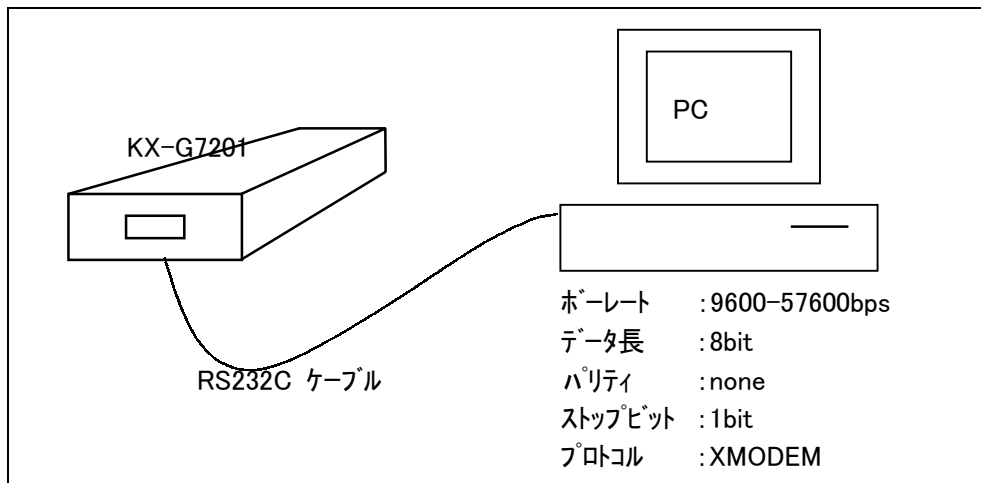
### 7.3. GCC内のメッセージを取り出す

1. GCCとリンクしている衛星を捕捉しているか確認する。
2. "all\_msg\_polling"をコールして、GCC内のメッセージを要求する。
3. GCCからの応答を待つ。この時SCは、応答待ちしている間"get\_st\_status()"の戻り値は4になる。
4. メッセージがあれば、GCCから送信され、そのメッセージは自動的にアウトバウンドキューに格納される。
5. "get\_st\_status()"の戻り値が0に変わるのをまって"get\_unget\_ob\_id\_for\_user" をコールし、アウトバウンドキューに受信データがあるかをチェックする。
6. もし受信データがあれば、そのタイプ(メッセージ/コメント)に応じた処理をする。

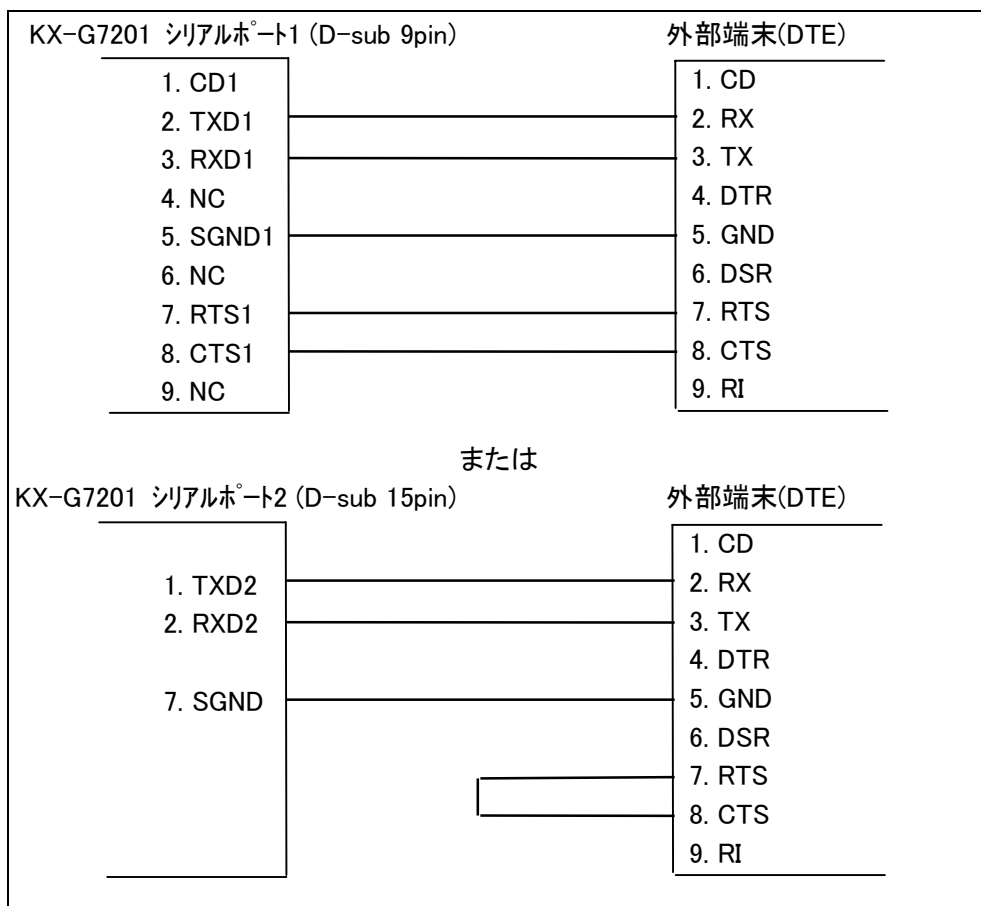


## 8. ユーザーアプリケーションプログラムのインストール方法

パソコンの通信ソフト(例えばハイパーターミナル)を使用してユーザーアプリケーションプログラムをユーザーROM領域に(0x060000)インストールします。



### 8.1. 接続



## 8.2. インストール方法

1. 端末とパソコンをRS232Cケーブルで接続し、ハイパーターミナルを起動してください。  
 端末には、シリアルポートが2チャンネルあります。何れのポートを用いてもインストールが可能です。但し、チャンネル2については通信速度57,600bpsは使用できません。
2. CTRL +KXORBと入力してコマンドモードに入ります。
3. CTRL + UALDRと入力してインストールモードに入ります。  
 PCに以下のメッセージが表示されます。  
 > [User Application Software]:\*\*\* Installation Mode \*\*\*  
 > Select SC's RS232C bps (1:9600/ 2:19200/ 3:38400/ 4:57600/ 5:Exit)"
4. インストールの通信速度を選択し、番号(1-4)を入力してください。  
 PCに以下のメッセージが表示されます。  
 > Match the modem bps to this, then reconnect.
5. ハイパーターミナルの通信速度を選択した通信速度に変更し、ハイパーターミナルを再接続してください。PCに以下のメッセージが表示されます。  
 > [User Application Software]:\*\*\* Installation Mode \*\*\*  
 (この操作は、通信速度選択後1分以内に、完了してください。)
6. PCからユーザーアプリケーションプログラムを送信します。  
 ハイパーターミナルの転送(T)メニューから、ファイルの送信(S)を選び、インストールするファイル(\*.up)を指定し、XMODEMまたは1K XMODEMプロトコルで送信します。  
 インストール終了後、PCに以下のメッセージが表示されインストールファイルのチェックを行います。  
 > [User Application Software]:Now, Checking User Application Software....  
 インストールのチェックが完了したら以下のメッセージが表示されます。  
 > Installation was Completed. Checksum OK.  
 Reset power switch"
7. 端末の電源を入れ直してください。

## 8.3. ユーザーアプリケーションの削除

1. 端末とパソコンをRS232Cケーブルで接続し、ハイパーターミナルを起動します。
2. CTRL +KXORBと入力してコマンドモードに入ります。
3. CTRL + UADLTと入力します。  
 PCに以下のメッセージが表示されます。  
 "User application area Initialize OK ? <Y/N>"
4. "Y"を押してください。  
 "Initialized : OK"  
 "Reset power switch"
5. 端末の電源を入れ直してください。

## 8.4. ユーザーアプリケーションの動作状態の確認

1. 端末とパソコンをRS232Cケーブルで接続し、ハイパーターミナルを起動します。
2. CTRL +KXORBと入力してコマンドモードに入ります。
3. KXCHKと入力すると、以下のような自己診断結果が表示されます。

```

>KXCHK

ST_ID      : OK
RAM        : OK
:          :
:          :
USER APPL  : 37EA (ACTIVE) <Ver 1.1>

```

“ USER APPL ”の表示結果は、ユーザーアプリケーションがインストールされていればそのチップIDと動作状態を表示し、インストールされていなければ “ NONE ” と表示します。またユーザーアプリケーションの動作状態は、ユーザーアプリケーションが動作中であれば “ ACTIVE ”、動作中でなければ “ NONE ACTIVE ” を表示します。ユーザーアプリケーションライブラリのバージョン番号は < > の中に表示されます。

## 9. ユーザーアプリケーションプログラムのデバッグ方法

ユーザーアプリケーションプログラム作成後、作成者はそのプログラムをデバッグすることができます。端末とパソコンをRS232Cケーブルで接続して、通信ソフト(例えば“ハイパーターミナル”)を起動するにより端末からいろいろなデバッグ情報を獲得できます。

ユーザーアプリケーションプログラムのデバッグ機能として以下の機能が用意されています。

1. メモリーダンプ
2. メモリー書き込み
3. スタック開始アドレスの獲得
4. ブレークポイント
5. アウトバウンドキューへのメッセージの登録

これらのデバッグ機能はデバッグモードがONの時に有効となります。デバッグモードをONにする為にはKXS70コマンドを使用してください。

デバッグモードON時にユーザーアプリケーションプログラムが起動されると、LEDが500 msec点灯します。これによりユーザーアプリケーションプログラムが起動されたかどうかの確認ができます。

### 9.1. メモリーダンプ

メモリー内容を16進で表示します。

入力コマンド: `mdump <アドレス>,<範囲>`

<例>

- ```
<CTRL> + "KXORB"      :コマンドモードに入る
> mdump #40f000,34      :#40f000～ #40f034までのメモリー内容表示

> 32 4F FD 3D 66 AA 00 58 24 AA 2F 5D D7 E5 FA AA
   11 12 3A 8F FA 5A 7C BB 20 2C B1 4E A6 A0 F7 DF
   34 5F
```

## 9.2. メモリー書き込み

RAM領域にデータを書き込みます。

入力コマンド: mem <アドレス>, <データ>

<例>

<CTRL> + "KXORB" : コマンドモードに入る  
> mem #40f000,ff : #40f000 番地に 0xff を書く

## 9.3. スタック開始アドレスの獲得

スタック領域の先頭アドレスを獲得します。ユーザーアプリのスタック領域は、システムRAM上に割り当てられおり、サイズは3200バイトです。

入力コマンド: stack

<例>

> stack  
4020d8 : スタックエリアは0x4020d8から0x402d58となります。

## 9.4. ブレークポイント

ブレークポイント機能によりユーザーアプリケーションプログラムの実行を一時停止させることができます。この機能を使用する為には、事前にユーザーアプリケーションプログラムのソースコードに "break\_point" 関数を挿入しておく必要があります。このブレークポイントは、ユーザーアプリケーションプログラムをロードした後、DTEからの "break" コマンドにより、有効/無効に設定できます。

<ブレークポイント関数>

ユーザーアプリケーションプログラムのソースコードに挿入。

KME 関数: break\_point( bp)

bp: ブレークポイント番号 (1 - 256)

<ブレークポイントの設定>

入力コマンド: "break"

ブレークポイントを有効にする: break <bp>,0

ブレークポイントを無効にする: break <bp>,1

すべてのブレークポイントを有効にする: break 0,0

すべてのブレークポイントを無効にする: break 0,1

[note]

ブレークした後、再開させる為には以下のコマンドを入力する。

入力コマンド: CTRL+"g"

## 9.5. アウトバウンドキューへのメッセージ登録

GCCからのメッセージを受信したように、アウトバウンドキューへメッセージをセットすることができます。メッセージ本体のほかに、サブジェクト、O/Rインディケータ、O/Rアドレスが設定できます。

### 1) 通常のOBメッセージ

<入力コマンド>

```
obreg [レシピエント1],[レシピエント2], [サブジェクト],<T or H>[メッセージ]<CR><LF>
      [レシピエント1,2]      : O/Rインディケータ、O/Rアドレス。O/Rインディケータの時には先頭に '@'を付加する。
      [サブジェクト]         : 最大10文字
      <T or H>                : [メッセージ]の種類。" <T>" : テキスト、" <H>" : バイナリー
      [メッセージ]          : メッセージ本体(最大256バイト)
```

<例>

- サブジェクト: TEST1、  
レシピエント: O/Rインディケータ1、  
メッセージ: "12345"

```
obreg @1,,TEST1,<T>12345
```

- サブジェクト: TEST2、  
レシピエント: O/Rインディケータ1、O/Rアドレス: "SUM"、  
メッセージ: "0x12,0x34,0x56,0xab"

```
obreg SUM,@1,TEST2,<H>123456ab
```

### 2) ユーザーコマンド

<入力コマンド>

```
obreg ,, CMD,<H>[data1] [data2] [data3] [data4] [data5]<CR><LF>
      [data1-5]              : バイナリーデータ。
```

<例>      データ 0x41,0x42,0x43,0x44,0x45

```
obreg ,,CMD,<H>4142434445<CR><LF>
```

### 3) アウトバウンドグローバルグラム

<入力コマンド>

```
obreg [レシピエント1],,GLB,<T>[メッセージ]<CR><LF>
```

<例>      レシピエント: O/Rインディケータ1、  
            メッセージ: "12345"

```
obreg @1,,GLB,<T>12345<CR><LF>
```



## 10. ユーザーアプリケーションプログラムの開発上の注意点

ユーザーアプリケーションプログラムを開発する時には次の点に注意してください。

- 大きいサイズのRAM領域を使用する際にはグローバル宣言してください。ローカル宣言するとスタックオーバーフローする可能性があります。
- RAM領域は、228000番地以降にアロケーションしてください。
- デバッグする時には、KXS67=1、KXS37=0に設定してください。
- 標準ライブラリの“printf()”で複数の“double”や“float”型の変数を“char”型に変換する場合、約300バイト以上のスタックを使用します。“printf()”をコールした時スタックが溢れ、プログラムの暴走の原因になっている事がありますので、もしコールしている付近でリセットしている形跡があれば“printf()”の使用を避けて下さい。“printf()”の代替関数として、print\_double(), orb\_printf1(),,,, orb\_printf7()を用意しています。
- ユーザーアプリケーションプログラムが暴走した場合には、PCの通信モードを以下の様に設定した後、“CTRL+R”を入力しながらリセットボタンを押してください。そうすれば、ユーザーアプリケーションプログラムは起動されません。また、端末の通信モードもDefault設定に戻っています。
 

|         |           |
|---------|-----------|
| ボーレート   | : 4800bps |
| パリティ    | : なし      |
| ストップビット | : 1bit    |
| データ長    | : 8bit    |

## 11. KX-G7100/G7101用プログラムとの互換性

### 11.1. UPファイル

UPファイルには、互換性はありません。  
KX-G7201専用の開発ツールにて再度UPファイルを作成する必要があります。

### 11.2. ソースファイル

C言語で記述されたソースファイルの大部分は互換性があると思われませんが、動作についての完全な互換性を保証するものではありません。  
ソースファイルは、それぞれのCPUやコンパイラ固有の仕様が有りますので、次の部分の記述に付いては、コンパイル前に訂正する必要があります。  
また、作成されたプログラムに関しては、再度品質の確認をお願いいたします。

(1) セクションの記述方法が異なる。

(例：KX-G720x)  
#pragma section USR  
#pragma section

(例：KX-G71xx)  
#pragma \_section B=USER  
#pragma \_section B

(2) Endian(データのメモリへの格納方法)が異なる。

(メモリーを直接参照するようなプログラムでは修正が必要。)

(3) KME-library

一部の関数に付いて、廃止・変更・追加が行われています。

(4) 定数・関数名の文字数制限

KX-G7201Nでは、33文字以降はコンパイル時に無視されます。

(5)有効小数桁

KX-G7201Nでは、小数点以下17桁以上を定義していた場合はコンパイルエラーとなります。

### 11.3. プログラムサイズ

プログラムの内容によって異なるため、一概に比較できませんがKX-G7100/G7101用とKX-G7201N用のプログラムサイズの差は概ね数%程度と思われます。

## 11.4. KMEライブラリー内の関数の追廃リスト

### 11.4.1. KX-G7100/G7101用ライブラリーから削除された関数

#### (1) Power Control

| Library Function     | Description        |
|----------------------|--------------------|
| go_sleep_next_path() | 次に衛星が飛来するまでスリープさせる |

#### (2) Measurement

| Library Function      | Description           |
|-----------------------|-----------------------|
| get_pos_quality_ind() | 測位品質を獲得する             |
| get_pass_quan()       | ドップラー測位で使用したパスの数を獲得する |

#### (3) Status Information

| Library Function  | Description         |
|-------------------|---------------------|
| get_stored_sats() | 軌道要素を格納している衛星数を獲得する |

#### (4) Other Functions

| Library Function | Description |
|------------------|-------------|
| orb_sprintf0()   | 数値の文字列変換    |

#### (5) KX Command

| Library Function | Description                        |
|------------------|------------------------------------|
| set_kxs10()      | get_kxs10() ホーリング複数ポートのインターバル      |
| set_kxs11()      | get_kxs11() ホーリング複数ポートの数           |
| set_kxs12()      | get_kxs12() ホーリング複数測位ポートのインターバル    |
| set_kxs13()      | get_kxs13() ホーリング複数測位ポートの数         |
| set_kxs19()      | get_kxs19() ドップラー測位時のフェリ-数とその収集間隔  |
| set_kxs20()      | get_kxs20() 測位結果の有効時間              |
| set_kxs21()      | get_kxs21() 測位結果の最小有効コリインデ-クター     |
| set_kxs22()      | get_kxs22() 軌道要素の有効時間              |
| set_kxs38()      | get_kxs38() パワ-データ通信インターバル         |
| set_kxs49()      | get_kxs49() UTC時間補正值               |
| set_kxs59()      | get_kxs59() 送信履歴自動送信モード            |
| set_kxs65()      | get_kxs65() 測位ポート送信                |
| set_kxs66()      | get_kxs66() ドップラー測位可否              |
| set_kxs73()      | get_kxs73() DGPS用のFM周波数            |
| set_kxs76()      | get_kxs76() 衛星飛来時刻を計算する時の衛星の最低仰角   |
| set_kxp01()      | get_kxp01() 入力ポートと出力ポートのリク         |
| set_kxa01()      | get_kxa01() 時刻指定送信 (KXA01コマンド)     |
| set_kxa02()      | get_kxa02() インターバル送信 (KXA02コマンド)   |
| set_kxa03()      | get_kxa03() 衛星飛来送信 (KXA03コマンド)     |
| set_kxa04()      | get_kxa04() パ-トE-ト設定 (KXA04コマンド)   |
| set_kxa05()      | get_kxa05() 入力ポート変化で送信 (KXA05コマンド) |
| set_kxb03()      | get_kxb03() 衛星飛来送信                 |

## 11.4.2. KX-G7100/G7101用ライブラリーに追加された関数

### (1) Serial Interface

| Library Function    | Description              |
|---------------------|--------------------------|
| chk_rx_buff_port()  | RS232Cの受信バッファをチェックする     |
| get_rx_data_port()  | RS232Cの受信バッファからデータを獲得する  |
| chk_tx_ready_port() | RS232Cの送信バッファが満杯かをチェックする |
| set_tx_data_port()  | RS232Cの送信バッファにデータをセットする  |
| cts_act_port()      | CTSポートをアクティブにする          |
| cts_neg_port()      | CTSポートをネガティブにする          |
| chk_cd_port()       | シリアルポートのCD信号の状態をチェックする   |
| chk_rts_port()      | シリアルポートのRTS信号の状態をチェックする  |

### (2) I/O Functions

| Library Function       | Description             |
|------------------------|-------------------------|
| get_adcon_data_10bit() | アナログポートのA/D値を10ビットで獲得する |

### (3) KX Command

| Library Function                     | Description                    |
|--------------------------------------|--------------------------------|
| set_kxs43_port()    get_kxs43_port() | RS232C通信モード                    |
| set_kxs90()        get_kxs90()       | アナログ入力のAD変換解像度と入力モード           |
| set_kxs91()        get_kxs91()       | オーブコム衛星の軌道情報のDTEへの出力           |
| set_kxs94()        get_kxs94()       | デジタル入出力ポートの入出力モード              |
| set_kxs95()        get_kxs95()       | プロトコルモードの packetsize           |
| set_kxs96()        get_kxs96()       | オーブコムシリアルリンク通信モードのサポートRS232C選択 |
| set_kxb05()        get_kxb05()       | I/O状態変化報知                      |

## 12. ユーザーアプリケーションの開発環境

### 12.1. ハードウェア環境

パソコンは以下の環境が必要です。

|         |               |
|---------|---------------|
| 機種      | PC-ATおよびその互換機 |
| CPU     | 80386以上       |
| メモリー    | 5.6MB 以上      |
| ハードディスク | 35MB以上        |

### 12.2. ソフトウェア

以下のソフトウェアおよびドキュメントが必要です。

1. H8S, H8/300シリーズ C/C++コンパイラ Ver. 2.0E PS008CAS3-MWR (日立製)  
(CD-ROMで供給され、電子文書版のマニュアルが含まれています。)
2. kme\_lib.h - KMEライブラリヘッダファイル
3. kme.lib.obj - KMEライブラリ
4. user\_main.obj -- ユーザーアプリケーション スタートアップ ライブラリ
5. ms2bin2.exe -- ファイルタイプ変換ユーティリティプログラム
6. bin2boot2.exe -- ファイルタイプ変換ユーティリティプログラム
7. ユーザーアプリケーションプログラムのインストール (本マニュアル)

また、サンプルプログラムを準備しています。

8. sample1.c -- サンプルプログラム
9. sample1.bat -- ユーザーアプリケーション作成用バッチファイル
10. sample1.lnk -- リンクファイル
11. compile.cmd -- コンパイルサブコマンドファイル
12. Kankyo.bat -- 環境変数設定バッチファイル
13. ユーザーアプリケーションサンプルプログラムのインストール

### 12.3. コンパイラーのインストール

コンパイラーのインストール手順に従ってコンパイラー類をインストールしてください。  
本マニュアルでは、コンパイラー類はC:\¥him以下のディレクトリにインストールされたものとして説明を行っています。

### 12.4. ディレクトリーの作成とファイルのコピー

¥himのサブディレクトリとして、tmpおよびuserディレクトリを作成します。  
tmpディレクトリは、コンパイル中に一時的に作成される中間ファイルが使用します。  
userディレクトリは、ユーザーアプリケーションの開発、コンパイル・リンク作業用です。

次のファイルは、c:\¥him¥Toolchains¥Hitachi¥ch38¥V20c¥Binにコピーします。

```
ms2bin2.exe -- ファイルタイプ変換用ユーティリティプログラム
bin2boot2.exe -- ファイルタイプ変換用ユーティリティプログラム
```

次のファイルは、c:\¥him¥userにコピーします。

```
kme_lib.h -- KMEライブラリヘッダファイル
kme_lib.obj -- KMEライブラリ
user_main.obj -- ユーザーアプリケーション スタートアップ ライブラリ
```

サンプルプログラムは必要に応じてc:\¥him¥userにコピーします。

```
sample1.c -- サンプルプログラム
sample1.bat -- ユーザーアプリケーション作成用バッチファイル
sample1.lnk -- リンクファイル
compile.cmd -- コンパイルサブコマンドファイル
```

### 12.5. PCの環境変数の設定

コンパイラが動作できるように、Kankyo.batの内容を参考に以下環境変数を書き替えて下さい。

```
set CH38=c:\¥him¥Toolchains¥Hitachi¥ch38¥V20c¥Include
set CH38TMP=c:\¥him¥tmp
set HLNK_LIBRARY1=c:\¥him¥Toolchains¥Hitachi¥ch38¥V20c¥Lib¥c8s26as.lib
path=%path%;c:\¥him¥Toolchains¥Hitachi¥ch38¥V20c¥Bin
path=%path%;c:\¥him¥Toolchains¥Hitachi¥Ink¥V53b
path=%path%;c:\¥him¥Toolchains¥Hitachi¥cnvs¥V15b
path=%path%;c:\¥him¥Toolchains¥Hitachi¥optlnk38¥V10b
```

## 13. 変更履歴

### Ver1.0

- ・ KX-G7201用新規作成

### Ver1.1 (2000.9.1)

- ・ シリアルポートの名称をチャンネル1、チャンネル2に統一
- ・ コンパイラーの品番・内容物の情報を修正、コンパイラーのバージョン指定を追加
- ・ Set\_kxs68/get\_kxs68のパラメータ記述を修正
- ・ Set\_kxs83/get\_kxs83のパラメータ記述を修正

### Ver1.21 (2000.12.26)

- ・ set\_kxs20/get\_kxs20を削除
- ・ set\_kxs59/get\_kxs59を削除
- ・ set\_kxb03/get\_kxb03を削除
- ・ set\_tx\_data\_portの引数の順序を訂正
- ・ orb\_sprintf0を削除
- ・ コンパイラーのバージョンを2.0から2.0Eへ変更
- ・ メモリーマップ関連アドレスの修正

### Ver1.3 (2001. 2.23)

- ・ set\_tx\_data, set\_tx\_data\_portの説明追加
- ・ set\_timerの戻値訂正
- ・ get\_ob\_typeの戻値訂正
- ・ set\_pin\_codeの要約、戻値訂正
- ・ set\_ncc\_search\_modeの戻値訂正
- ・ globalgram\_pollingの説明訂正
- ・ clear\_active\_msgの戻値、補足追加
- ・ clear\_mha\_ref\_num\_msgの戻値、補足訂正
- ・ req\_pos\_report\_to\_nccの解説、戻値訂正
- ・ req\_send\_ib\_globalgramの要約追加
- ・ req\_send\_ib\_enh\_globalgramの要約訂正
- ・ req\_send\_ob\_messageの戻値訂正
- ・ get\_ob\_msg\_subject\_indの要約訂正
- ・ get\_ob\_msg\_msg\_lenの要約訂正
- ・ get\_ob\_user\_commandの要約訂正
- ・ get\_ob\_globalgramの関数名、要約訂正
- ・ go\_sleep\_timeの補足訂正
- ・ get\_rx\_dataの戻値説明追加
- ・ get\_rx\_data\_portの要約、戻値訂正
- ・ get\_digital\_portの要約、戻値訂正
- ・ set\_digital\_portの要約訂正
- ・ get\_adcon\_data\_10bitの要約訂正
- ・ chk\_pos\_calc\_resultの補足削除

- set\_pdop\_thの補足訂正
- get\_pdop\_thの補足訂正
- break\_pointの要約、戻値、補足訂正
- chk\_break\_pointの戻値訂正
- req\_apl\_sat\_predictの補足追加
- chk\_apl\_sat\_predict\_resultの要約、補足訂正
- get\_apl\_sat\_predict\_timeの補足追加
- get\_sys\_infoの要約訂正
- set\_no\_wait\_for\_power\_saveの削除
- get\_kxs25の要約訂正
- get\_kxs27の要約訂正
- set\_kxs33の要約訂正
- get\_kxs33の要約、戻値訂正
- get\_kxs46の要約訂正
- set\_kxs50の要約訂正
- get\_kxs50の戻値訂正
- set\_kxs60の要約、戻値、補足訂正
- set\_kxs61の要約訂正
- get\_kxs61の要約訂正
- set\_kxs69の戻値訂正
- set\_kxs81の要約訂正
- get\_kxs81の要約訂正
- get\_kxd02の要約訂正

#### Ver1.4 (2001. 7.16)

- get\_unget\_ob\_id\_for\_userの補足追加、サンプルプログラム修正
- get\_adcon\_dataの補足追加
- get\_rx\_dataの補足修正
- set\_tx\_dataの補足修正
- cts\_actの補足削除
- cts\_negの補足削除
- chk\_rx\_buff\_portの補足修正
- get\_rx\_buff\_portの補足修正
- chk\_tx\_ready\_portの補足修正
- set\_tx\_data\_portの補足修正
- cts\_act\_port, cts\_neg\_port, chk\_cd\_port, chk\_rts\_port関数説明削除
- get\_digital\_portの戻り値図、補足修正
- set\_digital\_portの要約、補足修正
- get\_adcon\_dataの要約修正
- get\_adcon\_data\_10bitの要約修正
- set\_pdop\_thの補足追加
- set\_kxs47の要約修正
- set\_kxs55の要約修正
- set\_kxs87の補足追加
- get\_kxd02の要約修正
- 印刷アウトキーへのメッセージマニュアル登録の説明追加