

パナソニックオープンコム端末(KX-G7100/7101N)
ユーザーアプリケーション
プログラミングガイド

Ver 3.0

1999年3月25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
KYUSHU MATSUSHITA ELECTRIC CO.,LTD.

目次

1. 概要	4
2. SCのデータ処理	5
2.1. DTEとSC間の通信制御	6
2.2. 衛星とSC間の通信制御	7
3. ユーザーアプリケーションのプログラミング	10
3.1. スタートアップルーチン	10
3.2. レジスタの初期化	10
3.3. RAMの初期化	10
3.4. ユーザーアプリケーションプログラムへの分岐	10
3.5. 割り込み	10
4. ユーザーアプリケーションの起動	11
4.1. SCがパワーオンする時に起動	11
4.2. トラッキングコマンド (KXB01-B03) で指定された条件で起動	11
5. KMEライブラリ	12
5.1. Configuration	12
5.2. Communication Function	12
5.3. System Announce	12
5.4. SC-Originated Message	13
5.5. SC-Terminated Message	13
5.6. Power Control	14
5.7. Serial Interface	14
5.8. Time Functions	14
5.9. I/O Functions	15
5.10. Measurement	15
5.11. GPS Information	15
5.12. Status Information	16
5.13. Other Functions	16
5.14. KX Command	17
6. 各関数の詳細説明	19
7. プログラミング ヒント	216
7.1. GCCへのメッセージ送信	216
7.2. GCCからのメッセージ受信	217
7.3. GCC内のメッセージを取り出す	218
8. メモリーマップ	219
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	221
9.1. メモリーダンプ	221
9.2. メモリー書き込み	222
9.3. スタック開始アドレスの獲得	222
9.4. ブレークポイント	222
9.5. アウトバウンドキューへのメッセージ登録	223
10. 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	224
10.1. 接続	224

10.2.インストール方法.....	225
10.3.ユーザーアプリケーションの削除.....	225
10.4.ユーザーアプリケーションの動作状態の確認.....	226
11.ユーザーアプリケーションの開発上の注意点.....	227
Appendix A. ユーザーアプリケーションの開発環境.....	228
Appendix B. PCの環境設定.....	229

1. 概要

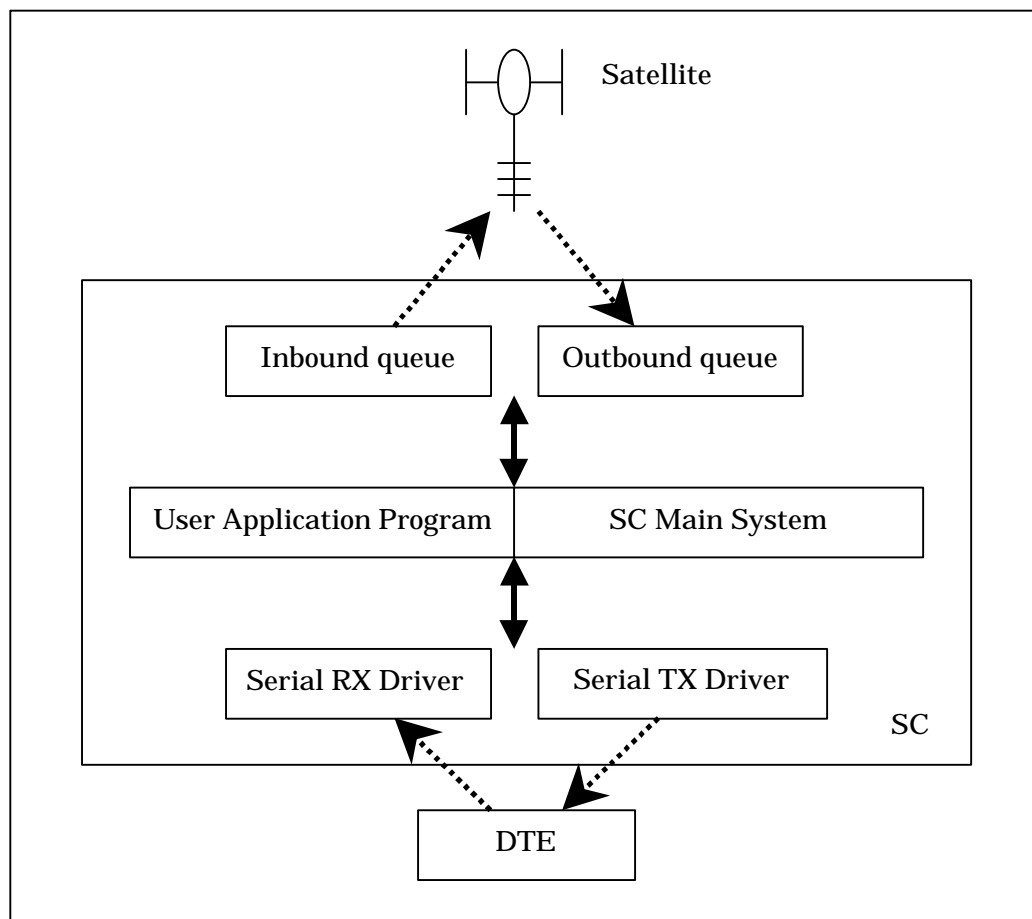
KMEのオープンコム端末(KX-G7100 / G7101 以下SC)は、ユーザーアプリケーション用に128KバイトのROMと4KバイトのRAMを装備しています。ユーザーアプリケーションは、KMEライブラリを使ってSCがもつ情報(例えば、デジタルI/O状況、SCの衛星捕捉状況など)の参照、SCへの測位要求、そしてメッセージの送受信ができます。また、この開発したプログラムは、ローダーを使ってSC内ROMに容易にロードする事ができ、SCの動作を使用環境に合うようにできます。

ユーザーアプリケーションを作成するにあたっては、松下電子工業(株)製MN10200シリーズの開発ツール(Cコンパイラ等)が必要です。この開発ツールについては下記ににお問い合わせください。

(株)ソフィアシステムズ DA事業部 営業部 畠中次長
〒215-8588
川崎市麻生区南黒川6-2
TEL:044-989-7253 FAX:044-989-7014

2. SCのデータ処理

SCは、衛星から受信したメッセージをアウトバウンドキューに、衛星へ送信するメッセージをインバウンドキューに保持しています。また外部端末(以下DTE)との通信は、シリアルドライバを経由して行われ、DTEから受信したデータ用にRXバッファ、DTEへの送信データ用にTXバッファを持ちます。通常これらのキュー及びバッファは、下図の示すようにメインシステムでアクセスされ処理されています。



< 図 2 . 1 >

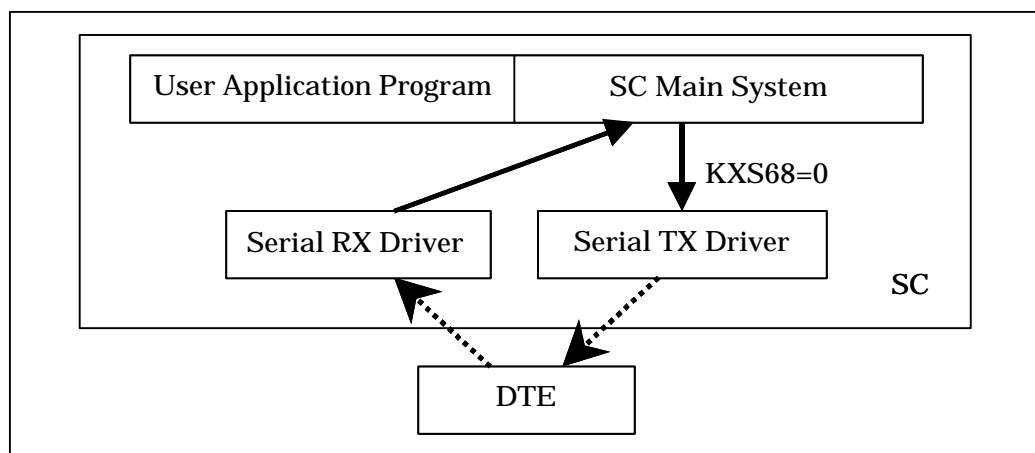
このSCのメインシステムは、常時起動していますので、ユーザーアプリケーションが衛星間通信やDTE間通信を行うような場合は、メインシステムとユーザーアプリケーションでデータの競合がおきないようにSCのデータアクセス権の設定を行う必要があります。

2.1. DTEとSC間の通信制御

SCは、シリアルRXドライバ（DTEからの受信用）とTXドライバ（DTEへの送信用）を持っています。これらの通信ドライバに対してはSCのメインシステムとユーザーアプリケーション双方から制御することが可能です。

2.1.1. メインシステムでシリアルドライバを制御するには

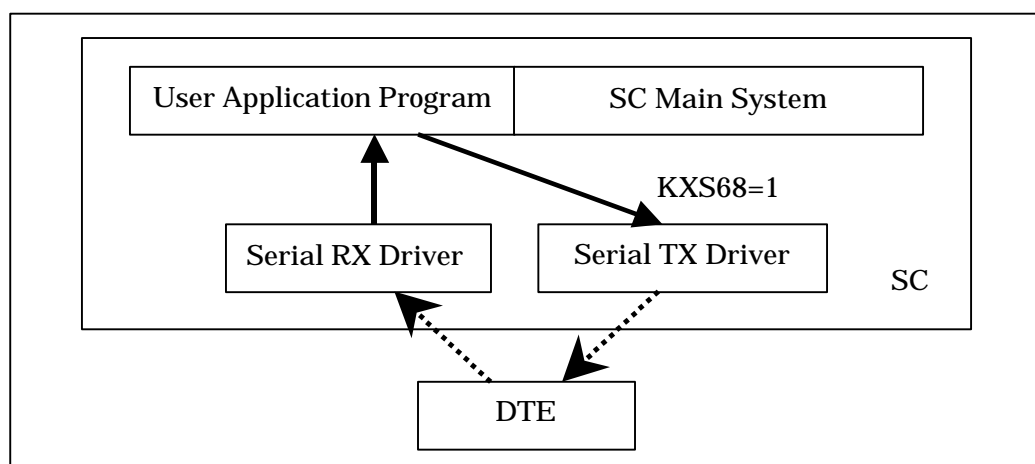
この場合、KXS68の設定を0 (KXS68=0) にします。
ユーザーアプリケーションでDTEとの通信をする必要がない場合、あるいはDTE間の通信はメインシステム（プロトコルモード・バイトモード）で行わせたい場合に設定します。



< 図 2 . 2 >

2.1.2. ユーザーアプリケーションでシリアルドライバを制御するには

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、メインシステムはアクセスできません。



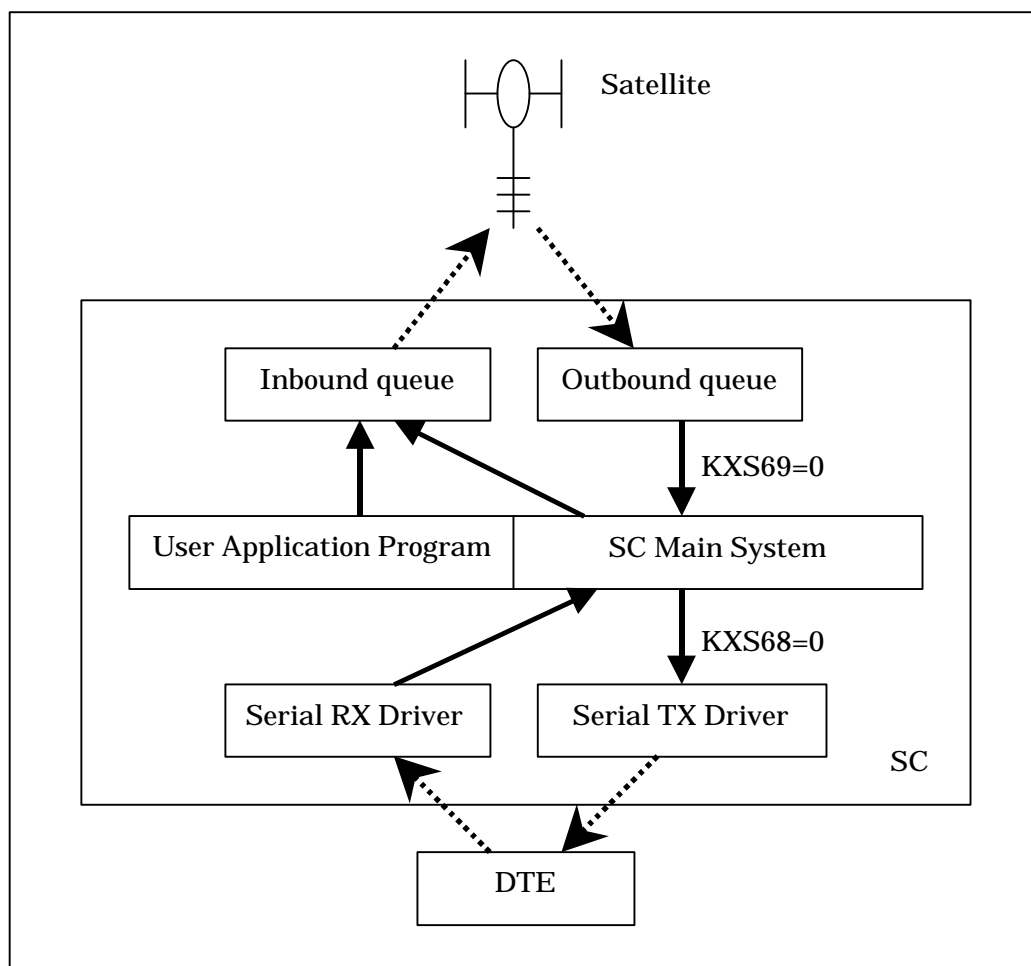
< 図 2 . 3 >

2.2. 衛星とSC間の通信制御

衛星と通信するメッセージは、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になります。インバウンドキューに対してはメインシステム、ユーザーアプリケーション双方からアクセスが可能です。アウトバウンドキューへのアクセスは、メインシステムだけ、あるいはユーザーアプリケーションだけ、そして両方で同じメッセージをアクセスすることもできます。

2.2.1. メインシステムで衛星キューを制御するには

このような設定にするには、KXS69=0、KXS68=0にします。ユーザーアプリケーションからも衛星へ独自メッセージの送信は可能が、受信は出来ません。



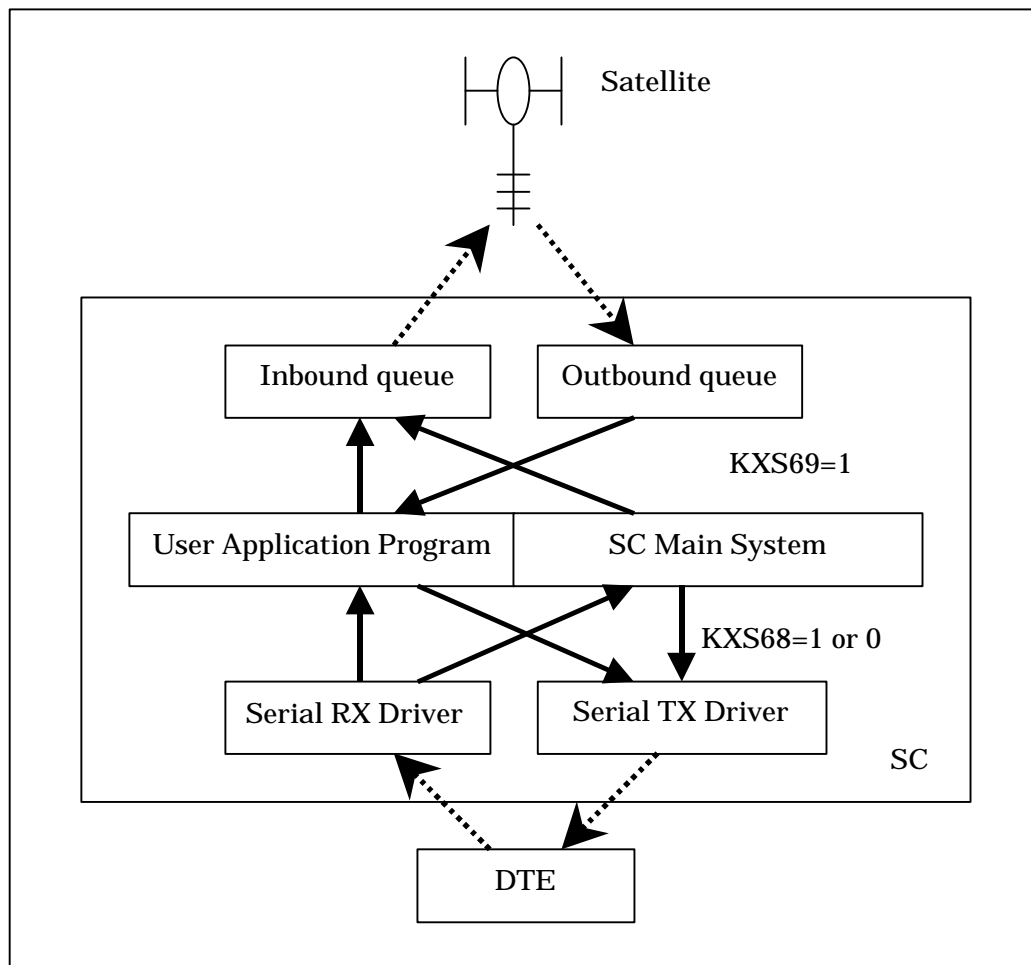
< 図 2 . 4 >

注意:

1. KXS69=0、KXS68=1にするとメインシステム側のアウトバウンドキュー処理ができないため、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。

2.2.2. 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

KXS69の設定を1 (KXS69=1) にします。
 ユーザーアプリケーションが衛星とのメッセージ送受信を行う場合に設定します。
 ユーザーアプリケーションは、アウトバウンドの全てのメッセージを獲得し、獲得後、そのメッセージをキューから削除します。メインシステムからアウトバウンドキューに対してはアクセスできません。シリアルドライバに対してはKXS68の設定に従います。(2.1章参照)



< 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のプログラミングは、最初に記載しているようにプログラムをコンパイル・リンクをするために松下電子工業製のソフト開発ツールが必要です。またプログラムのコーディングはC言語を使って開発してください。

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開発する場合、スタートアップ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。これらの処理は、全てメインシステムで行われます。

- ・レジスタの初期化
- ・RAM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. レジスタの初期化

ユーザーアプリケーションで使用するスタックは、ユーザーアプリケーションが起動する直前にメインシステムによって初期化されます。

3.3. RAMの初期化

ユーザーアプリケーションは、4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。また、このRAMについては、通常ドップラ測位処理で使用しているワークをユーザーアプリケーションに開放することで、ドップラ測位は使えなくなりますが14キロバイトまで増やす事ができます。設定は、KXS66の設定を1(KXS66=1)にします。

3.4. 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。

3.5. 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。

4. ユーザーアプリケーションの起動

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. トラッキングコマンド (KXB01-B03) 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は、SCの情報の獲得・設定、SCに対していろいろな処理要求をするためライブラリを使えます。このライブラリは、SC内モジュールをサブルーチンコールしています。

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NCCを設定する	17
set_pin_code()	PIN CODEを設定する	18
set_ncc_search_mode()	衛星サーチモードを設定する。	19

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()	NCC内の全てのメッセージの送信要求	20
s_msg_polling()	NCC内の全ての150バイト以下のメッセージの送信要求	21
globalgram_polling()	衛星内のグローバルメッセージ送信要求	22
chk_failed_polling()	ポーリングに失敗した時の原因を獲得する	23
chk_ib_tx()	指定メッセージの送信状況を確認する	24
clear_active_msg()	送受信中メッセージを削除する	26
clear_mha_ref_num_msg()	指定のメッセージをキューから削除する	27
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	28
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	29
req_pos_report_to_ncc()	NCCへポジショナルレポートの送信を要求する	30
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	31

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()	NCCから受信したポーリングがあるか確認する	32
get_polling_info()	ポーリングの応答内容を獲得する	33

5.4. SC-Originated Message

NCC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	34
req_send_ib_report()	送信キューにレポートをセットする	35
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	36
req_send_ib_enh_globalgram()	送信キューにエンハンスドグローバルメッセージをセットする	38
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	39

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	40

5.5. SC-Terminated Message

NCC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	42
remove_ob_q()	アウトバウンドキューからデータを削除する。	45
get_ob_type()	データ種を獲得する	46
get_ob_ncc_id()	メッセージのNCC_IDを獲得する	47
get_ob_msg_subject_ind()	メッセージのサブジェクトを獲得する	48
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	49
get_ob_msg_or_quan()	メッセージの送信元を獲得する	50
get_ob_msg_msg_len()	メッセージ長を獲得する	51
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	52
get_ob_message ()	メッセージを獲得する	53
get_ob_user_command()	コマンドを獲得する	54
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	55
get_ob_globalgram_or_ind()	グローバルメッセージの送信元を獲得する	56
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	57
get_ob_globalgram_msg_body()	グローバルメッセージを獲得する	58

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。	59
go_sleep_next_path()	次に衛星が飛来するまでスリープさせる。	60
go_wait ()	指定時間ユーザソフトをサスペンドする。	61

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	62
rf_block_power_off()	RFブロック電源をOFFする	63
chk_rf_block_power()	RFブロックの電源状態を獲得する	64

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()	RS232Cの受信バッファをチェックする。	65
get_rx_data()	RS232Cの受信バッファからデータを獲得する。	66
chk_tx_ready()	RTSポート状態を獲得する	67
set_tx_data()	RS232Cの送信バッファにデータをセットする	68
cts_act()	CTSポートをアクティブにする	69
cts_neg()	CTSポートをネガティブにする	70

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	71
set_timer()	タイマをセットする	72
check_timer()	セットしたタイマが経過したか確認する	74

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	75
set_digital_port()	デジタルポートをセットする	76

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。	77

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	78
req_pos_calc_from_user()	測位をさせる	79
chk_pos_calc_result()	測位が完了したかチェックする	80
get_pos_quality_ind()	測位品質を獲得する	81
get_pass_quan()	ドブツラー測位で使用したパスの数を獲得する	82
get_pos_age()	前回測位してからの経過時間を獲得する	83
get_lat_val()	測位結果(緯度)を獲得する	84
get_lon_val()	測位結果(経度)を獲得する	85
set_lat_val()	端末の保持している測位情報(緯度)をセットする	86
set_lon_val()	端末の保持している測位情報(経度)をセットする	87

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	88
get_x06_str()	X06 センテンスを獲得する	90

これらの関数は、GPS搭載機でのみ有効です。

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。	91
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	92
get_sat_no()	捕捉している衛星番号を獲得する	93
get_ncc_quan()	捕捉している衛星とリンクしているNCCの数を獲得する	94
get_ncc_id_prio()	NCCに送信可能なメッセージプライオリティを獲得する	95
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	96
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	97
get_week_time_val()	UTC時00:00:00からの経過秒を獲得する	98
calc_gps_week()	GPS週数を獲得する	99
get_total_sats()	システム内の衛星数を獲得する	100
get_stored_sats()	軌道要素を格納している衛星数を獲得する	101
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	102

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	103
break_point()	デバックのためのブレイクコール	104
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	105
get_application_dbg()	デバック機能ON / OFFを獲得する	106
led_on()	LEDを点灯する	107
led_off()	LEDを消灯する	108
suspend_ib_msg_tx()	衛星への送信を一時中断する。	109
resume_ib_msg_tx()	衛星への送信を再開する	110
get_utc_time()	UTC時を獲得する	111
set_utc_time()	UTC時を設定する	112
get_local_time()	ローカル時を獲得する	113
calc_distance()	2点間の距離を算出する	114
req_apl_sat_predict()	次の衛星飛来時刻計算を要求する	115
chk_apl_sat_predict_result()	衛星飛来計算が終了したか確認する	116
get_apl_sat_predict_time()	算出した飛来時刻を獲得する	117
get_sys_info()	自己診断結果を獲得する	118

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, KXA, KXBコマンド)一覧表。

タイプ別関数		内容	ページ
設定関数	獲得関数		
set_kxs01()	get_kxs01()	登録しているNCC(GCC)番号	119
set_kxs02()	get_kxs02()	デフォルトポート	120
set_kxs03()	get_kxs03()	デフォルトプライオリティ	121
set_kxs04()	get_kxs04()	デフォルトポートO/Rインデクサ	122
set_kxs05()	get_kxs05()	デフォルトメッセージ/グローバルプログラムO/Rインデクサ	123
set_kxs06()	get_kxs06()	デフォルトackレベル	124
set_kxs07()	get_kxs07()	デフォルトメッセージフォーマット	125
set_kxs08()	get_kxs08()	デフォルトサービスタイプ	126
-----	-----		
set_kxs10()	get_kxs10()	ホッピング複数ポートのインターバル	127
set_kxs11()	get_kxs11()	ホッピング複数ポートの数	128
set_kxs12()	get_kxs12()	ホッピング複数測位ポートのインターバル	129
set_kxs13()	get_kxs13()	ホッピング複数測位ポートの数	130
set_kxs14()	get_kxs14()	NCCサーチモード	131
set_kxs15()	get_kxs15()	データリンクチャンネル	132
set_kxs16()	get_kxs16()	データリンクチェックサムエラースレッシュ	133
set_kxs17()	get_kxs17()	データ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	134
set_kxs18()	get_kxs18()	連続測位モード	135
set_kxs19()	get_kxs19()	ドップラー測定時のエラー数とその収集間隔	136
set_kxs20()	get_kxs20()	測位結果の有効時間	137
set_kxs21()	get_kxs21()	測位結果の最小有効クォリティインデクサ	138
set_kxs22()	get_kxs22()	軌道要素の有効時間orbit elements age	139
set_kxs23()	get_kxs23()	緯度・経度	140
set_kxs24()	get_kxs24()	測位モード	141
set_kxs25()	get_kxs25()	KXA/BコマンドでのGPS測位情報のフォーマット	142
set_kxs26()	get_kxs26()	DTEに対するホッピングの応答の待ち時間	143
set_kxs27()	get_kxs27()	シリアルポートのACK時間	144
set_kxs28()	get_kxs28()	シリアルポートのリトライ回数	145
set_kxs29()	get_kxs29()	ポートポート送信モード	146
set_kxs30()	get_kxs30()	ポートポートの内容	148
set_kxs31()	get_kxs31()	RS232Cの通信モード(ポートポート/ポート)	149
set_kxs32()	get_kxs32()	ポートトリガ	151
set_kxs33()	get_kxs33()	ポートタイムアウト	152
set_kxs34()	get_kxs34()	ポート長	153
set_kxs35()	get_kxs35()	ポート時の、送信・受信SOM, EOM	154
set_kxs36()	get_kxs36()	ポートメッセージタイプ	155
set_kxs37()	get_kxs37()	パワーダウンモード	156
set_kxs38()	get_kxs38()	パワーダウン最小インターバル	157
set_kxs39()	get_kxs39()	インアクティブインターバル	158
set_kxs40()	get_kxs40()	パワーセーブモード	159
set_kxs41()	get_kxs41()	インバウンドフロー制御	160
set_kxs42()	get_kxs42()	アウトバウンドフロー制御	161
-----	get_kxs43()	RS232C通信モード(ポート/ポート/etc)	162
set_kxs44()	get_kxs44()	RS232C通信モード(半2重、全2重)	163
set_kxs45()	get_kxs45()	インバウンドメッセージトリートメント	164
set_kxs46()	get_kxs46()	アウトバウンドメッセージトリートメント	165
set_kxs47()	get_kxs47()	メッセージリキューブション	166
-----	get_kxs48()	インバウンド・アウトバウンドキューのサイズ	167
set_kxs49()	get_kxs49()	UTC時間補正值	168
set_kxs50()	get_kxs50()	ピンコード	169

set_kxs51()	get_kxs51()	自動グローバルラムダリンク	170
set_kxs52()	get_kxs52()	GPS測地系	171
set_kxs53()	get_kxs53()	RTS論値仕様	172
-----	-----		173
set_kxs55()	get_kxs55()	KXBコマンドで送信するアラームポート番号	174
set_kxs56()	get_kxs56()	KXB コマンド 移動距離検知の移動距離	175
set_kxs57()	get_kxs57()	KXB コマンド エリア検知のエリア	176
set_kxs58()	get_kxs58()	KXB コマンド 速度検知の速度	177
set_kxs59()	get_kxs59()	送信履歴自動送信モード	178
set_kxs60()	get_kxs60()	KXA/Bコマンドで送信するデータフォーマット	179
set_kxs61()	get_kxs61()	RS232 ドライバレベルセーブモード	180
-----	-----		181
set_kxs63()	get_kxs63()	タイムアウト	182
set_kxs64()	get_kxs64()	KXB コマンドでのクイックリワードコメント	183
set_kxs65()	get_kxs65()	連続測位、DTEからの測位後の測位レポート送信トップ	184
-----	get_kxs66()	ラ測位に使用するRAM領域開放	185
set_kxs67()	get_kxs67()	端末電源ON時に、ユーザーアプリを起動	186
set_kxs68()	get_kxs68()	RS232C受信データの処理ルート	187
set_kxs69()	get_kxs69()	GCCからの受信データの処理ルート	188
set_kxs70()	get_kxs70()	ユーザーアプリケーションのバグモード	189
set_kxs71()	get_kxs71()	GCCからのコマンドリモートセットアップ ID	190
set_kxs72()	get_kxs72()	ホストからのコマンドリモートセットアップでその応答メッセージ	191
set_kxs73()	get_kxs73()	DGPS用のFM周波数	192
set_kxs74()	get_kxs74()	KXA/Bコマンドによる無条件送信での送信方法	193
set_kxs75()	get_kxs75()	KXA/Bコマンドでのメッセージのグローバルラムダ自動的変換	194
set_kxs76()	get_kxs76()	衛星飛来時刻を計算する時の衛星の最低仰角	195
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	196
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	197
set_kxs79()	get_kxs79()	KXBコマンドの検知時間	198
set_kxs80()	get_kxs80()	アウトバウンドグローバルラムダセットのフォーマット	199
set_kxd01()	get_kxd01()	デジタルポート出力	200
-----	get_kxd02()	アラーム入力値	201
set_kxp01()	get_kxp01()	入力ポートと出力ポートのリンク	202
set_kxm01()	get_kxm01()	固定メッセージ	203
set_kxa01()	get_kxa01()	時刻指定送信(KXA01コマンド)	203
set_kxa02()	get_kxa02()	インターバル送信(KXA02コマンド)	204
set_kxa03()	get_kxa03()	衛星飛来送信(KXA03コマンド)	205
set_kxa04()	get_kxa04()	バイトモード設定(KXA04コマンド)	206
set_kxa05()	get_kxa05()	入力ポート変化で送信(KXA05コマンド)	207
set_kxb01()	get_kxb01()	時刻指定送信(KXB01コマンド)	208
set_kxb02()	get_kxb02()	インターバル送信(KXB02コマンド)	209
set_kxb03()	get_kxb03()	衛星飛来送信(KXB03コマンド)	210
set_kxa00()	-----	KXA01 - 05コマンドの設定を解除	211
set_kxb00()	-----	KXB01 - 03 コマンドの設定を解除	212

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 : NCC(GCC)番号(0 - 255)
解説	端末が登録されているNCC(GCC)番号を設定する
戻値	なし
補足	コマンドモードの KXS01 コマンドで変更できる。

set_pin_code

要約	<code>void set_pin_code(unsigned long pin_code);</code> <code>unsigned long pin_code: ピンコード (0 - 0x0fffffff)</code>
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	なし
補足	コマンドモードの KXS50 コマンドで変更できる。 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-フト (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=derired_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=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	NCC (GCC) サ-フトを設定する。
戻値	<p>0) 設定失敗</p> <p>1) 設定成功</p>
補足	コマンドモードの KXS14 コマンドで設定できる。

all_msg_polling

要約	int all_msg_polling(unsigned char ncc_id); unsigned char ncc_id: NCC(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。ホ-リング 開始 1: 衛星を捕捉していない等でホ-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ウトバウンドキューに格納される。 ホ-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がホ-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: NCC(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-リング 開始 1: 衛星を捕捉していない等でホ-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 (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-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 (s_msg_polling(get_kxs01()) == 0) {
            /* SCがホ-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() == 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: NCC(GCC)番号
解説	GCC 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。ホ-リンク 開始 1: 衛星を捕捉していない等でホ-リンク 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。 ホ-リンクは、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がホ-リンク 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 端末がポ-リング失敗した時の原因を獲得する
戻値 タイグコード

- 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-ブ 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-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NCCが衛星データベースの中に見つからない。
- 104 = 不十分なメッセージプライオリティ(NCCが輻輳中かもしれない)
- 105 = 衛星が応答しない(アプリケーション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NCC内に存在する
- 109 = インバンドメッセージナンバーエラー
- 110 = 不揮発性メモリにメッセージをセーブしている時に、NCC内にエラーが生じた
- 111 = NCC内のデータベースにエラーが生じた
- 112 = 追加の診断情報が無い
- 113 = インバンド転送の再送回数が最大値を越えた
- 114 = グローバルグラム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 = グローバルグラムのサイズ超過
- 119 = NCC内にアウトバンドメッセージ/コメントがない。
- 120 = NCC内に150バイト以下のメッセージ/コメントがない。
- 121 = 衛星内にグローバルグラムがない。
- 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 = グローバルグラムの最大値(16)まで現在の衛星に格納されている。

補足

KYUSHU MATSUSHITA ELECTRIC CO.,LTD.

- 108 = インバウンドメッセージが既にNCC内に存在する
- 109 = インバウンドメッセージナンバーエラー
- 110 = 不揮発性メモリにメッセージをセーブしている時に、NCC内にエラーが生じた
- 111 = NCC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 = インバウンド転送の再送回数が最大値を越えた
- 114 = グローバルプログラム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 = グローバルプログラムのサイズ超過
- 119 = NCC内にアウトバウンドメッセージ/コマンドがない。
- 120 = NCC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プログラムがない。
- 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 = グローバルプログラムの最大値(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）をクリアする

戻値 NULL : クリア成功
 それ以外: 通信中ではない

補足

clear_mha_ref_num_msg

要約	<code>ib_id clear_mha_ref_num_msg(unsigned char mha_ref_num);</code> unsigned char mha_ref_num: メッセージ識別番号
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	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、あるいはそのメッセージは現在送信中。 それ以外: 削除成功
補足	送信中メッセージを削除したければ、“clear_active_msg”関数を使用する。

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の中の全ての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の中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_pos_report_to_ncc

要約	<pre>int req_pos_report_to_ncc(unsigned char or_ind, unsigned char ncc_id);</pre> <p>unsigned char or_ind : O/R インデキータ unsigned char ncc_id : NCC(GCC) 番号</p>
解説	測位を開始して、終了後その測位結果をホストへ送信する
戻値	<p>-1: NG 衛星が無い為、要求できない 0 : NG インパウトキーが満杯 1 : OK</p>
補足	0(NG)の時は、インパウトキーのメッセージが送信されてキーが空き領域ができてから再度この関数を呼ぶ。

req_select_next_downlink

要約 void req_select_next_downlink(void);

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にロックしようとする

chk_polling_event

要約 int chk_polling_event(void)

解説 NCC (GCC) からのホ-リング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。


```

/* 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ポートキューに設定 */
subject[0] = '¥0';
if (req_send_ib_message(get_kxs010, /* NCC(GCC) 番号 */
                        0, /* すぐに送信 */
                        get_kxs060, /* ACKレベル */
                        get_kxs030, /* プライオリティ */
                        14, /* メッセージホテタイプ : ハイリ- */
                        0, /* メッセージホテサブタイプ (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(); /* 1-ザ-ア-リ-ク-シ-ョ-ン-終-了 */
}

```


<例>

```

#include "kme_lib.h"

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

    /* レポート本体設定 */
    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;

    /* インポートレポートをインポートキューに設定する */
    if (req_send_ib_report( get_kxs010, /* NCC(GCC) 番号 */
                          0, /* すぐに送信 */
                          get_kxs08(), /* サービスタップ */
                          get_kxs04(), /* O/Rインテイクータ */
                          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	<pre>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)</pre> <p>ncc_id : NCC (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</p>
解説	インバウンドグローバルグラムをインバウンドキーに設定する
戻値	<p>0 : インバウンドキーに設定完了 1 : インバウンドキーに設定失敗。長すぎてキーに入らない。 2 : インバウンドキーに設定失敗。メモリが獲得できない。</p>
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_ob_message

要約	<pre>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)</pre> <p>ncc_id : NCC (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</p>
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	<p>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。</p>
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>

```

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-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-ジ° をアットハ° ウド° キュ-に設定する */
    subject[0] = '\0';
    if (req_send_ob_message( get_kxs010,          /* NCC(GCC ) 番号 */
                            0,                    /* メッセ-ジ° ホ° テ° タイ° : TEXT */
                            5,                    /* メッセ-ジ° ホ° テ° イサ° タイ° */
                            recipient,           /* 受信者 */
                            subject,             /* サ° ジ° ェ° ト° */
                            message,            /* メッセ-ジ° 本体 */
                            strlen(message) ) /* メッセ-ジ° 長 */
        == MESSAGE_QUEUED) { /* 設定成功? */
        ; /* 設定成功 */
    } else {
        ; /* 設定失敗 */
    }
    exit_user_apl(); /* 1-ザ-アプ リケ-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。

<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. NCC(GCC) 番号 */
                /* 2. サブシフト */
                /* 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7 ウソ7 1-ザ7 -コソソ7 */
    /* 1. NCC(GCC) 番号 */
    /* 2. 7 -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 グローバル グラム */
    /* 1. NCC(GCC) 番号 */
    /* 2. メッセージ ホテイ タイプ */
    /* 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;
}
}
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 ID
解説	アウト キーからアウト 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 ID

解説 アウト キーにあるデータの種別を取得

戻値 0) アウト メッセージ
 1) アウト コメント
 2) アウト グローバル プログラム

補足 アウト ID は、関数“get_unget_ob_id_for_user”により獲得する

get_ob_ncc_id

要約	<code>unsigned char get_ob_ncc_id(ob_id get_ob_id);</code> <code>ob_id get_ob_id</code> : アウト ID
解説	アウト ID 指定のデータの NCC(GCC) 番号を取得する
戻値	NCC(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	<code>int get_ob_msg_subject_ind(ob_id get_ob_id);</code> <code>ob_id get_ob_id</code> : アカウント ID
解説	サブジェクトの有無を取得する。
戻値	0: サブジェクト無し 1: サブジェクト有り
補足	アカウント 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> <code>ob_id get_ob_id</code> : アウト ID
解説	アウト ID 指定のメッセージの受信者の数を獲得する
戻値	受信者の数
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_msg_len

要約	<code>int get_ob_msg_msg_len(ob_id get_ob_id);</code> <code>ob_id get_ob_id</code> : アウト ID
解説	アウト ID 指定のメッセージのメッセージ長を獲得する
戻値	メッセージ長
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_msg_body_index

要約 unsigned int get_ob_msg_msg_body_index(ob_id get_ob_id, int
 *msg_len);

ob_id get_ob_id: アウト ID
int *msg_len: メッセージ長.

解説 アウト ID 指定のメッセージのメッセージ本体を指すインデックスを取得する

戻値 メッセージ本体を指すインデックス

補足 アウト ID は、関数“get_unget_ob_id_for_user”により獲得する

get_ob_message

要約	<code>unsigned char get_ob_message (ob_id get_ob_id, unsigned int msg_index);</code>
	<code>ob_id get_ob_id</code> : アウト ID <code>unsigned int msg_index</code> : インデックス
解説	アウト 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バイト (0x00 - 0xff)
補足	アウト ID は、関数“get_unget_ob_id_for_user”により獲得する

get_ob_user_command

要約	<pre>int 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獲得する
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 指定のグローバルラムの識別番号を獲得する
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獲得する

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

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-ダ-ウさせる
戻値	なし
補足	スリープモードが0(KXS37=0)の時は、無効です

go_sleep_next_path

要約	<code>void go_sleep_next_path(unsigned long int sleep_time);</code> <code>unsigned long int sleep_time</code> : 最低パワーダウン時間 (0 - 4294967295[分])
解説	少なくとも最低パワーダウン時間で示される秒数分だけパワーダウンする。それから次の衛星のパスで起床する。
戻値	なし
補足	スリープモードがOFF(KXS37=0)の時は、無効です

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: ミリセカド 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、1-ザ-アプリタスクの実行権を他タスクに渡す関数であり、パワーダウンとは違う。

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	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	unsigned int chk_rx_buff(void);
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	受信データ (0x00 - 0xff)
補足	RS232C の受信バッファサイズは 256 バイト。

chk_tx_ready

要約 int chk_tx_ready(void);

解説 RS232C インターフェイスの CTS 信号をチェックする

戻値 0 = 送信不可。CTS が忙。
 1 = 送信可能。CTS が忙。

補足

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 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 256 バイト

cts_act

要約	void cts_act(void)
解説	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 CTS 信号をノアケイブにする.

戻値 なし

補足

get_inc_timer

要約	<code>unsigned int get_inc_timer(void);</code>
解説	OSタイマ-のカウント値を獲得する。 このカウントは2 msec毎にカウントアップされる。
戻値	カウント値 (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	<p>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。</p> <p>ミリセカント : 4294967295 ミリセカント 秒 : 274877906 秒 分 : 4581298 分</p>

<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();                     /* 1-ザ-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();                             /* 1-ザ-アプリ終了*/
}

```

check_timer

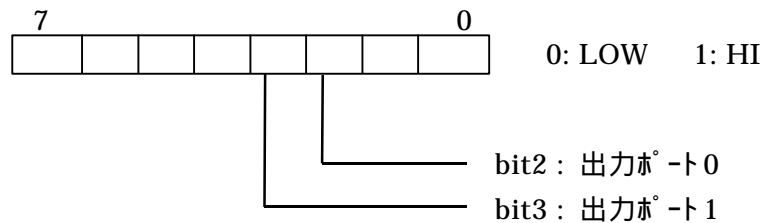
要約	int check_timer(char no); char no: タイマ-番号 (1 - 10)
解説	指定タイマ-がタイムアップしたかチェックする
戻値	0 = タイムアップした 1 = タイムアップしてない
補足	

set_digital_port

要約 void set_digital_port(unsigned char sel_port, unsigned char data);

unsigned char sel_port: 出力ポート識別番号

unsigned char 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

<例>

```
#include "kme_lib.h"

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

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

    /* 出力ポート1をLOW に設定 */
    Set_digital_port(0x08, 0x00);

    /* 出力ポート1をHI に設定 */
    Set_digital_port(0x08, 0x08);

    exit_user_apl();          /* 1-ザ-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° ホ-ト番号

0= RSSI

1=アナログ° ホ-ト 1

2=アナログ° ホ-ト 2

3- 7= 予備

解説 アナログ° ホ-トの状態を獲得する

戻値 値 (0 -255)

補足

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(); /* 1-ザ-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(); /* 1-ザ-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	測位が完了しても K X S 20 の設定時間経過したらこの関数の戻り値は 0 になります。

get_pos_quality_ind

要約	unsigned char get_pos_quality_ind(void)
解説	測位精度を示す "クオリティインデキータ" を獲得する。
戻値	測位精度 0 : GPSによる測位結果 3 : ドップラ測位による測位結果 15 : ドップラ測位による測位結果 (3 より測位精度が落ちる)
補足	

get_pass_quan

要約 unsigned char get_pass_quan(void)

解説 トップラ測位に使用した衛星軌道の数を獲得する。

戻値 軌道数

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）を獲得する。
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

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.

get_x06_str

要約 void get_x06_str(char *str)

char str[170] : x06 センテンス

x05 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 から出力される測位情報である。

< 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"> 0 = アイドル 1 = インバウンドメッセージ送信中 2 = インバウンドレポート送信中 3 = インバウンドグローバルグラム送信中 4 = アウトバウンドメッセージ受信中 5 = アウトバウンドコマンド受信中 6 = アウトバウンドグローバルグラム受信中 7 = テキスト中 8 = ローカルブッキング中 9 = リモートブッキングテスト中
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 (グローバルグラムの送信要求) 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 レジスタ番号を取得する

戻値 0x00 - 0xfe : 現在送信中の mha レジスタ番号
 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 : 衛星を捕捉していない
補足	

get_ncc_quan

要約 unsigned char get_ncc_quan (void);

解説 現在リンクしている NCC (GCC) の数を獲得する

戻値 現在リンクしている NCC (GCC) の数

補足

get_ncc_id_prio

要約	<pre>int get_ncc_id_prio(unsigned char index, unsigned char *ncc_id, unsigned char *min_prio);</pre> <p>index : インデックス。0 から数える ncc_id : NCC(GCC) ID min_prio : 受信可能なメッセージの優先度の最小値</p>
解説	NCC(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 NCC(GCC)数だけ、インデックスを+1 してこの関数を呼ぶ。

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 0:00 からの経過秒を獲得する

戻値 秒 : 0 - 7*24*60*60 [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_stored_sats

要約 unsigned char get_stored_sats(void)

解説 端末が格納している衛星軌道要素の数を獲得する

戻値 衛星軌道要素の数

補足

get_check_errs

要約 unsigned char get_check_errs(void)

解説 衛星データ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	void break_point(unsigned char brk_num) unsigned char brk_num: ブレークポイント番号 (1 - 255).
解説	この関数はユーザーアプリのプログラムをデバッグする時に使用される。 ユーザーは、あらかじめユーザーアプリのプログラムの中に“break_point”関数を挿入しておく必要がある。
戻値	なし
補足	“break_point” は、255 個まで設定でき、それぞれ ON/OFF 設定ができる。 “break 23 0” : ブレークポイント 23 を有効にする “break 23 1” : ブレークポイント 23 を無効にする ブレークした後、“ctrl+g”コマンドにより再開する。 これらの機能は、デバッグ機能が ON (KXS70=1) の時に有効となる。

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 0 : ブレークポイントが設定されている
 1 : 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: 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	端末が衛星へ送信している間は NCC(GCC)からのメッセージを受信できない。インバウンドキューに多数のメッセージが格納されている場合には端末は連続で送信を行い、長時間受信できなくなるケースがある。これを回避する為に、この関数をコールして一時端末からの送信を中止して、メッセージ受信ができるようにする。

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

```
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;
```

年情報は、80 以上が 19xx を、80 未満が 20xx を示す。

解説 UTC 時刻を獲得する

戻値 なし

補足 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

```
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;
```

年情報は、80 以上が 19xx を、80 未満が 20xx を示す。

解説 UTC 時刻を設定する

戻値 なし

補足 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-加時間
 int time_zome : タイムゾーン (分) (-690 - 720)

解説 日-加時間を獲得する

戻値 なし

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。

<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();    /* 1-ザ-アプリ終了 */
}
```

chk_apl_sat_predict_resault

要約 int chk_apl_sat_predict_resault(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

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。

get_sys_info

要約 char get_sys_info(int obj)

int obj : 番号

0	ROMバージョン
3	インバウンド/アウトバウンドキュ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
6	送信機 - (送信部) のチェック
7	受信機 - (受信部) のチェック
8	衛星通信のデジ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取得する

戻値 0: OK
 1: NG

補足

set_kxs01

要約	int set_kxs01(unsigned char value) unsigned char value:N C C (G C C) 番号(0 - 255)
解説	N C C (G C C) 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	N C C (G C C) 番号 (0 - 255)を獲得する
戻値	N C C (G C C) 番号(0 - 255).
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 ncc_id;

    set_kxs01(43);           /* N C C ( G C C ) 番号を設定する */
    ncc_id = get_kxs01();    /* N C C ( G C C ) 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、NCC 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	デフォルトポートを獲得する
戻値	デフォルト polled 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、NCC 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	デフォルト優先度を獲得する
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を獲得する */
}
```

set_kxs04

要約	int set_kxs04(unsigned char value) unsigned char value : デ フォルト レボ-ト O/R イデ イケ-タ
解説	デ フォルト レボ-ト O/R イデ イケ-タ(0 -3)を設定する
戻値	0) 設定成功 1) 設定失敗 (ハ 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 イデ イケ-タ(0 -3)を獲得する
戻値	デ フォルト レボ-ト O/R イデ イケ-タ(0 -3)
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_kxs04(0);            /* デ フォルト レボ-ト O/R イデ イケ-タ(0 -3)を設定する*/
    priority = get_kxs04(); /* デ フォルト レボ-ト O/R イデ イケ-タ(0 -3)を獲得する*/
}
```

set_kxs05

要約	int set_kxs05(unsigned char value) unsigned char value : メッセ-ジ O/R インデ-イタ(0 - 15)
解説	デフォルト メッセ-ジ O/R インデ-イタ(0 - 15)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_kxs05

要約	unsigned char get_kxs05(void)
解説	デフォルト メッセ-ジ O/R インデ-イタ(0 - 15)を獲得する
戻値	メッセ-ジ O/R インデ-イタ(0 - 15)
補足	

例 :

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

void main(void)
{
    unsigned char or_ind;

    set_kxs05(0);           /* デフォルト メッセ-ジ O/R インデ-イタを設定する */
    priority = get_kxs05(); /* デフォルト メッセ-ジ O/R インデ-イタを獲得する */
}
```

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)を獲得する

戻値 Ack レベル (0 - 4)

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 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 を獲得する */
}
```

set_kxs07

要約 int set_kxs07(unsigned char type, unsigned char object)

unsigned char type : テ フォルトメッセ-ジ ホ テ イタイ (0 -15)

unsigned char object: サブ タイ (0 - 4)

解説 テ フォルトメッセ-ジ ホ テ イタイ (0 -15)を 設定する

戻値 0) 設定成功
1) 設定失敗 (ハ ラメ-タIラ-)

補足

get_kxs07

要約 void get_kxs07(unsigned char *type, unsigned char *object)

unsigned char *type : テ フォルトメッセ-ジ ホ テ イタイ (0 -15)

unsigned char *object: サブ タイ (0 - 4)

解説 テ フォルトメッセ-ジ ホ テ イタイ (0 -15)を 獲得する

戻値 なし

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 type, sub_type;

    set_kxs07(0, 5);          /* テ フォルトメッセ-ジ ホ テ イタイ を 設定する */
    set_kxs07(14, 0/*dummy*/); /* テ フォルトメッセ-ジ ホ テ イタイ を 設定する */
    get_kxs05(&type, &sub_type); /* テ フォルトメッセ-ジ ホ テ イタイ を 獲得する */
}
```


set_kxs08

要約	int set_kxs08(unsigned char value) unsigned char value : サービススタイ (0 - 4, 10 - 14)
解説	デフォルトサービススタイ (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	サービススタイ (0 - 4, 10 - 14)を獲得する
戻値	サービススタイ (0 - 4, 10 - 14)
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 service;

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

set_kxs10

要約	int set_kxs10(unsigned int value) unsigned char value : インターバル(0 – 65535)[分]
解説	レポートインターバルを設定する
戻値	0) 設定成功 1) 設定失敗 (パラメータエラー)
補足	レポートインターバルは、GCCからのオプションコマンド (複数オプションレポート) を受信した時に参照される。

get_kxs10

要約 unsigned int get_kxs10(void)

解説 レポートインターバルを獲得する

戻値 レポートインターバル(0 – 65535)[分]

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 int interval;

    set_kxs10(5);              /* レポートインターバルを設定する */
    interval = get_kxs10();    /* レポートインターバルを獲得する */
}
```

set_kxs11

要約	int set_kxs11(unsigned char value) unsigned char value: レポート数 (0 - 255)
解説	複数レポートリンク時のレポート数を設定する
戻値	0) 設定成功 1) 設定失敗 (パラメータエラー)
補足	レポート数は、GCCからのレポートリンクコメント (複数レポートリンクレポート) を受信した時に参照される。

get_kxs11

要約	unsigned char get_kxs11(void)
解説	複数レポートリンク時のレポート数を獲得する
戻値	レポート数 (0 - 255).
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 rpt_num;

    set_kxs11(1);              /* 複数レポート数を設定する */
    rpt_num = get_kxs11();     /* 複数レポート数を獲得する */
}
```

set_kxs12

要約	int set_kxs12(unsigned int value) unsigned char value : Interval (0 - 65535) [分]
解説	測位ポートインターバルを設定する
戻値	0) 設定成功 1) 設定失敗 (パラメータエラー)
補足	測位ポートインターバルは、GCCからのポートリンクコマンド (複数ポートリンク 測位ポート) を受信した時に参照される。

get_kxs12

要約 unsigned int get_kxs12(void)

解説 測位ポートインターバルを獲得する

戻値 測位ポートインターバル(0 - 65535 分)

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 int interval;

    set_kxs12(10);           /* 測位ポートインターバルを設定する */
    interval = get_kxs12(); /* 測位ポートインターバルを獲得する */
}
```

set_kxs13

要約	int set_kxs13(unsigned char value) unsigned char value : 測位レポート数(0 - 255)
解説	複数ポートリンク時の測位レポート数を設定する
戻値	0) 設定成功 1) 設定失敗 (パラメータエラー)
補足	測位レポート数は、GCC からのポートリンクコメント (複数ポートリンク 測位レポート) を受信した時に参照される。

get_kxs13

要約	unsigned char get_kxs13(void)
解説	複数ポートリンク時の測位レポート数を獲得する
戻値	測位レポート数(0 - 255)
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 pos_rpt_num;

    set_kxs13(0);            /* 複数ポートリンク時の測位レポート数を設定する */
    pos_rpt_num = get_kxs13(); /* 複数ポートリンク時の測位レポート数を獲得する */
}

```

set_kxs14

要約	int set_kxs14(unsigned char value)
	<p>unsigned char value : NCCサ-フト (0 - 4)</p> <p>0=kxs01に設定されているNCC (GCC)を連続的にダウリツバンドの中から検索する</p> <p>1= kxs01に設定されているNCC (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ツバンドのロックを維持する。</p> <p>2=最初に発見したダウリツバンドのロックを維持する。</p> <p>3= kxs01に設定されているNCC (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ツバンドのロックを維持する。</p> <p>4= kxs01に設定されているNCC (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NCC (GCC)のダウリツバンドを検索し続ける。</p>
解説	NCCサ-フトを設定する
戻値	0) 設定成功 1) 設定失敗 (パラメータ)
補足	

get_kxs14

要約	unsigned char get_kxs14(void)
解説	NCCサ-フトを獲得する
戻値	NCCサ-フト (0 - 4).
補足	
例 :	<pre>#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-フトを設定する */ search_mode = get_kxs14(); /* NCCサ-フトを獲得する */ }</pre>

set_kxs15

要約 int set_kxs15(unsigned char block, unsigned int channel)

unsigned char block : 番号 (0 - 23)

unsigned int ch : ダ° ウリンクチャンネル (0 - 399)

解説 ダ° ウ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 (パラメタエラー)

補足 SCが衛星ダ° ウリンクアクイシ° ヨ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° ウリンクチャンネル (24 チャンネル)

解説 ダ° ウリンクチャンネル (24チャンネル) 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° ト° する° .° */

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

    set_kxs15(0, 80);                   /* ダ° ウリンクチャンネル° を° 設定° する° */
    get_kxs15(channel);                 /* ダ° ウリンクチャンネル° (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	最大チツクサムイレ-数を設定する
戻値	0) 設定成功 1) 設定失敗 (パラメ-タイレ-) 2) 排他イレ-:KXS17(1)の時は KXS16(51 - 100)にできない
補足	異なったダウリンクの獲得を試みる前に連続する指定グループ (KXS17) で許されるダウリンクチツクサムイレ-の最大数

get_kxs16

要約 unsigned char get_kxs16(void)

解説 最大チツクサムイレ-数を獲得する

戻値 最大チツクサムイレ-数(1 - 100).

補足

例 :

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

void main(void)
{
    unsigned char threshold;

    set_kxs16(10);             /* 最大チツクサムイレ-数を設定する */
    threshold = get_kxs16();   /* 最大チツクサムイレ-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-タエラー)
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-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 */
    count = get_kxs18();                        /* 連続測位モードを獲得する */
}
```

set_kxs19

要約 int set_kxs19(unsigned char rate, unsigned char point)

unsigned char rate : I/Fメ-収集間隔 (4,8,12,16秒)

unsigned char point: I/Fメ-数

50 - 150 : 4 秒間隔の時

25 - 75 : 8秒間隔の時

20 - 50 : 12秒間隔の時

20 - 35 : 16 秒間隔の時

解説 トッラ-測位時のI/Fメ-数とその収集間隔を設定する

戻値 0) 設定成功
 1) 設定失敗 (パラメタ-)
 2) 排他エラー

補足

get_kxs19

要約 void get_kxs19(unsigned char *rate, unsigned char *point)

unsigned char *rate : I/Fメ-収集間隔

unsigned char *point: I/Fメ-数

解説 トッラ-測位時のI/Fメ-数とその収集間隔を獲得する

戻値 なし

補足

例 :

```
#include "kme_lib.h"                   /* ヘッダ-ファイルをインクル-ドする。*/
void main(void)
{
    unsigned char rate, point;
    set_kxs19(4, 50);                   /* I/Fメ-数とその収集間隔を設定する */
    get_kxs19(&rate, &point);         /* I/Fメ-数とその収集間隔を獲得する */
}
```

set_kxs20

要約	int set_kxs20(unsigned long value) unsigned long value : 1イ (0 – 65535) [分]
解説	測位結果の有効時間を設定する
戻値	0) 設定成功 1) 設定失敗 (パラメータエラー)
補足	GCC からホ-リング コマンド (測位) を受信した時、前回の測位結果がこの有効時間内に入っていれば、まず前回の測位結果を送信して、再度測位を開始して、測位終了後にその結果を送信する。

get_kxs20

要約	unsigned long get_kxs20(void)
解説	測位結果の有効時間を獲得する
戻値	測位結果の有効時間(0 – 65535 分)
補足	

例 :

```
#include "kme_lib.h"           /* ヘッダ-ファイルをインクル-ドする. */
void main(void)
{
    unsigned long age;

    set_kxs20(10);             /* 測位結果の有効時間を設定する */
    age = get_kxs20();        /* 測位結果の有効時間を獲得する */
}
```

set_kxs21

要約	int set_kxs21(unsigned char value) unsigned char value: クォリティインディケータ (0 - 15)
解説	測位結果の最小有効クォリティインディケータを設定する
戻値	0) 設定成功 1) 設定失敗 (パラメータエラー)
補足	GCC にオプションポートを送る際、測位結果の最小有効クォリティインディケータ

get_kxs21

要約	unsigned char get_kxs21(void)
解説	測位結果の最小有効クォリティインディケータを獲得する
戻値	クォリティインディケータ(0 - 15)
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 indicator;

    set_kxs21(3);              /* クォリティインディケータを設定する */
    indicator = get_kxs21();   /* クォリティインディケータを獲得する */
}
```

set_kxs22

要約	int set_kxs22(int value) int value: 有効時間 (12 - 8760) [時間]
解説	軌道要素の有効時間を設定する
戻値	0) 設定成功 1) 設定失敗 (パラメータエラー)
補足	

get_kxs22

要約	int get_kxs22(void)
解説	軌道要素の有効時間を獲得する
戻値	有効時間(12 - 8760) [時間]
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)
{
    int age;

    set_kxs22(3);               /* 軌道要素の有効時間を設定する */
    age = get_kxs22();         /* 軌道要素の有効時間を獲得する */
}
```

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 測位モードを獲得する

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();                 /* 測位モードを獲得する */
}
```


set_kxs25

要約	int set_kxs25(int value) unsigned char value : type of sent message 0= LAT/LON 1= NMEA
解説	KXA/B モジュールでの 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	.

get_kxs25

要約	int get_kxs25(void)
解説	KXA/BモジュールでのGPS測位情報のフォーマットを獲得する
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) [分]
解説	DTE に対するホ-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-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-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-リング の応答の待ち時間を獲得する
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-リング の応答の待ち時間を設定する */
    timeout = get_kxs26();     /* ホ-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) [分]
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	void resume_ib_msg_tx(void)
解説	シリアルポートのACK時間を獲得する
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が帰ってこなかったらリンクをアボ-トする (0=アボ-トしない)

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-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-トルボ-トを送信しない
 1= アホ-トルボ-トを送信する

解説 アホ-トルボ-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-トルボ-ト送信モードを獲得する

戻値 0= アホ-トルボ-トを送信しない
 1= アホ-トルボ-トを送信する

補足

set_kxs30

要約	<pre>int set_kxs30(unsigned char gcc_id, unsigned char polled, unsigned char srv_type, unsigned char or_ind, char *inf) unsigned char gcc_id: NCC(GCC)番号 (0-255) unsigned char polled: ホ-ルト (0, 1) unsigned char srv_type: サ-ビ-スタ-イ- (0-4, 10-14) unsigned char or_ind: O/Rイ-テ-イ-ク- (0-3) unsigned char *inf: テ-ク- (0x00-0xff) * 6 バ-イ-ト</pre>
解説	ア-ホ-トル-ホ-ルト-の内容を-設定-する
戻値	0) 設定成功 1) 設定失敗 (バ-ラ-メ-タ-イ-ラ-) 2) 排他イ-ラ-
補足	

get_kxs30

要約	<pre>void get_kxs30(unsigned char *ncc_id, unsigned char *polled, unsigned char *srv_type, unsigned char *or_ind, char *inf) unsigned char ncc_id: NCC(GCC)番号 (0-255) unsigned char polled: ホ-ルト (0, 1) unsigned char srv_type: サ-ビ-スタ-イ- (0-4, 10-14) unsigned char or_ind: O/Rイ-テ-イ-ク- (0-3) char *inf: テ-ク- (0x00-0xff) * 6 バ-イ-ト</pre>
解説	ア-ホ-トル-ホ-ルト-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"); /* ポリトポトを設定する */
    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 (パラメータエラー)

補足

get_kxs31

要約 unsigned char get_kxs31(void)

解説 RS232Cの通信モードを獲得する

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)
{
    /* プロトコルモードに設定 */
    set_kxa000();             /* KXA設定を解除 */
    set_kxb000();             /* KXB設定を解除 */
    set_kxs31(0);             /* プロトコルモードを設定する */
}

```

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;

    /* ハートモードに設定 */
    set_kxa000();             /* KXA設定を解除 */
    set_kxb000();             /* KXB設定を解除 */
    set_kxs31(1);             /* ハートモードに設定 */

    /* ハートモードのパラメータを設定 */
    set_kxs32(1);             /* ハートモードトリガ-を設定する */
    set_kxs33(1);             /* ハートモードタイムアウトを設定する */
    set_kxs34(200);           /* ハートモード長を設定する */
    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);             /* ハートモードのメッセージタイプを設定する */
}

```

set_kxs32

要約	int set_kxs32(unsigned char value)
	<p>unsigned char value : トリガ - (0, 1)</p> <p>0=最初のデータを受信してハイレート長受信あるいはハイレートタイムアウト秒経過したら送信開始</p> <p>1=最初のデータを受信してハイレート長受信あるいはハイレートタイムアウト秒経過するまでにハイレート RX_SOM/RX_EOMを受信したら送信開始</p>
解説	ハイレートトリガ-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_kxs32

要約	unsigned char get_kxs32(void)
解説	ハイレートトリガ-を獲得する
戻値	トリガ-(0, 1).
補足	

set_kxs33

要約	int set_kxs33(unsigned int value) unsigned char value : 時間 (1 – 3600) [分]
解説	ハイト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 int get_kxs33(void)
解説	ハイトトタイムアウトを獲得する
戻値	時間 (1 – 3600 秒)
補足	

set_kxs34

要約	int set_kxs34(unsigned int value) unsigned char value : バイト数 (1 - インバイトキューサイズ)
解説	バイト長を設定する
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	バイト長を獲得する
戻値	長さ (1 - インバイトキューサイズ).
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_som : (0x00 - 0xff)

解説 ハイトト時の、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 ハイトト時の、送信・受信SOM,EOM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-ジ 1=レポ-ト 2=グロ-バ-ル-ク-ラム
解説	ハ-イト-モード-メッセ-ジ-タイプ-を設定する
戻値	0) 設定成功 1) 設定失敗 (ハ-ラ-メ-タ-エラー-)
補足	ハ-イト-モード-の-デ-ータ-を-レポ-ト-メッセ-ジ- / デ-ータ-ク-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-イト-モード-メッセ-ジ-タイプ-を-獲得-する
戻値	メッセ-ジ-タイプ (0 - 2) 0=インバ-ウンド-メッセ-ジ 1=インバ-ウンド-レポ-ト 2=インバ-ウンド-グロ-バ-ル-ク-ラム
補足	

set_kxs37

要約 int set_kxs37(unsigned char value)

unsigned char value : 冪ト°
 0=OFF
 1=ON

解説 ハ° ワ-タ° ウン冪ト° を設定する

戻値 0) 設定成功
 1) 設定失敗 (ハ° ラメ-タ-エラ-)

補足

get_kxs37

要約 unsigned char get_kxs37(void)

解説 ハ° ワ-タ° ウン冪ト° を獲得する

戻値 冪ト° (0:OFF, 1:ON)

補足

例:

```
#include "kme_lib.h"                    /* ヘッダ° -ファイル° を° インクル° ト° する° */

void main(void)
{
    /* set that the power down mode is on. */
    set_kxs37(1);                        /* ハ° ワ-タ° ウン冪ト° を° 設定° する° */

    set_kxs38(30);                       /* ハ° ワ-タ° ウン° ミ° ム° インタ° ル° 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° ティ° ブ° インタ° ル° を° 設定° する° */
}
```

set_kxs38

要約	int set_kxs38(unsigned long value) unsigned char value : 時間 (0 – 535600) [分]
解説	パワーダウンミムインターバルを設定する
戻値	0) 設定成功 1) 設定失敗 (パラメータエラー)
補足	パワーダウンミムインターバルは、衛星とのリンクが切れて次の衛星飛来時刻までパワーダウンする時に参照される。パワーダウンミムインターバル時間以降の最初の衛星飛来時刻までパワーダウンする。 衛星軌道要素を受信していて、緯度・経度が設定されていて、パワーダウンミムインターバルが0でない時に起動計算を行う。これ以外は、インアクティブインターバルだけパワーダウンする。

get_kxs38

要約	unsigned char get_kxs38(void)
解説	パワーダウンミムインターバルを獲得する
戻値	時間 (0 – 535600 分)
補足	

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 (パラメータエラー)
補足	set_kxs38 参照

get_kxs39

要約	unsigned long get_kxs39(void)
解説	インアクティブ インターバルを獲得する
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 だけ無線受信系グループの電源を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) / 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	インバウンドフロー制御を設定を獲得する
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を獲得する
戻値	0= RTSアクティブで送信中止. 1=フロー制御なし
補足	

set_kxs43 はサポートされていません。

get_kxs43

要約 void get_kxs43(SERIAL_PAR *serial)

SERIAL_PAR *serial

unsigned char baud : ボ-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 通信ポートを獲得する

戻値 なし

補足

例:

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

void main(void)
{
    SERIAL_PAR seral;

    get_kxs43(&serial);      /* RS232C通信ポートを獲得する */
}
```


set_kxs45

要約	int set_kxs45(unsigned char value)
	<p>unsigned char value : 0-1 (0, 1)</p> <p>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</p> <p>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</p>
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	<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_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を獲得する
戻値	<p>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</p> <p>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</p>
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 (パラメータエラー)
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てもキューから削除しない 1= GCCへのメッセージ送信失敗したらキューから削除する
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を獲得する
戻値	0=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 1= GCCへのメッセージ送信失敗したらキューから削除する
補足	

set_kxs48はサポートしていません。

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のサイズを獲得する

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_kxs49

要約	int set_kxs49(unsigned char value) unsigned char value : 時間 (0 – 20) [秒]
解説	UTC 時間補正値を設定する
戻値	0) 設定成功 1) 設定失敗 (パラメータエラー)
補足	

get_kxs49

要約	unsigned char get_kxs49(void)
解説	UTC時間補正値を獲得する
戻値	時間 (0 – 20 秒)
補足	

set_kxs50

要約	int set_kxs50(unsigned long value) unsigned char value : コード (0 - 4294967295)
解説	ピコードを設定する
戻値	0) 設定成功 1) 設定失敗 (パラメータエラー)
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	ピコードを獲得する
戻値	コード (0 - 4294967295) .
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ランプ-リンクをかける
解説	自動グローバルランプ-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	自動グローバルランプ-リンクを獲得する
戻値	0=何もしない 1=衛星に対しグローバルランプ-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 論値仕様を設定する
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論値仕様を獲得する
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	

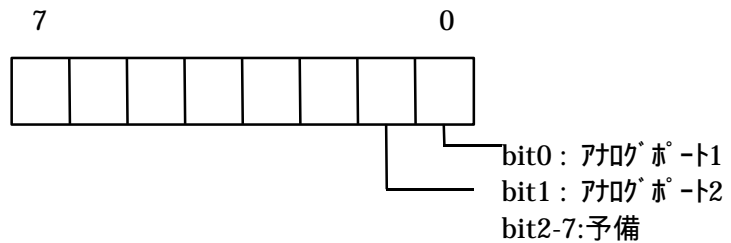
“set_kxs54” はサポートされていません。

“get_kxs54” はサポートされていません。

set_kxs55

要約 int set_kxs55(char value)

char value: 送信するアノグポート番号



解説 KXBポートで送信するアノ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ポートで送信するアノグポート番号を獲得する

戻値 アノグポート番号

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 コント`速度検知の速度を設定する

戻値 0) 設定成功
 1) 設定失敗 (パラメータエラー)

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 コント`速度検知の速度を獲得する

戻値 なし

補足

set_kxs59

要約	int set_kxs59(unsigned char value)
	unsigned char value : 戻り値 (0, 1) 0= 何もしない 1= 送信履歴を DTE に自動で送信する
解説	送信履歴自動送信戻り値を設定する
戻値	0) 設定成功 1) 設定失敗 (パラメータエラー)
補足	

get_kxs59

要約	unsigned char get_kxs59(void)
解説	送信履歴自動送信戻り値を獲得する
戻値	戻り値 (0, 1) 0= 何もしない 1= 送信履歴を DTE に自動で送信する
補足	

set_kxs60

要約	int set_kxs60(unsigned char value)
	unsigned char value : タイプ (0, 1) 0= テキスト形式 1= ハイリ-形式
解説	KXA/B 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)
補足	

get_kxs60

要約	unsigned char get_kxs60(void)
解説	KXA/Bコマンドで送信するデータフォーマットを獲得する
戻値	タイプ (0, 1) 0= テキスト形式 1= ハイリ-形式
補足	

set_kxs61

要約	int set_kxs61(unsigned char value) unsigned char value: 0-1 (0, 1) 0=OFF 1=ON
解説	RS232 ドライバ-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	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を獲得する
戻値	0-1 (0, 1) 0=OFF 1=ON .
補足	

“set_kxs62” はサポートされていません。

“get_kxs62” はサポートされていません。

set_kxs63

要約 int set_kxs63(TIME_WINDOW *time_win)

 TIME_WINDOW *time_win
 unsigned char on_off: 0-1 (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 タイムアウト° が設定されると、開始 - 終了時間以外は端末はパワーダウンする。

get_kxs63

要約 int set_kxs63(TIME_WINDOW *time_win)

 TIME_WINDOW *time_win
 unsigned char on_off: 0-1 (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 タイムアウト° 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 = モード切 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を設定するを獲得する
戻値	クイックワータウンモード (0,1) 0 = モード切 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_kxs65

要約	int set_kxs65(unsigned char value)
	unsigned char value : 0-1 (0,1) 0= 何も送信しない 1= 測位レポートを送信する
解説	連続測位、DTE からの測位要求で、測位終了後に GCC に測位レポートを送信するかどうかの設定
戻値	0) 設定成功 1) 設定失敗 (パラメータエラー)
補足	

get_kxs65

要約	unsigned char get_kxs65(void)
解説	連続測位、DTEからの測位要求で、測位終了後にGCCに測位レポートを送信するかどうかの設定を獲得する
戻値	0-1 (0,1) 0= 何も送信しない 1= 測位レポートを送信する
補足	

set_kxs66はサポートされていません。

get_kxs66

要約	unsigned char get_kxs66(void)
解説	トッパラ測位に使用するRAM領域をユーザーアプリに開放するかどうかを獲得する
戻値	0: 開放しない 1: 開放する
補足	

set_kxs67

要約	int set_kxs67(unsigned char value)
	unsigned char value : 戻り値 (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時に、ユーザーアプリを起動するかどうかを設定を獲得する
戻値	戻り値 (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 - 2) 0:メインシステムで処理 1:1-ザ-アプリケーションで処理 2: DGPS タスクで処理
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 (パラメータエラー)
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を獲得する
戻値	タイプ (0 - 2). 0:メインシステムで処理 1:1-ザ-アプリケーションで処理 2:DGPS タスクで処理
補足	

set_kxs69

要約	int set_kxs69(unsigned char value)
	unsigned char value : タイプ (0 - 2) 0:メインシステムで処理 1:1-ザ-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(1or2)
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を獲得する
戻値	タイプ (0 - 2) 0:メインシステムで処理 1:1-ザ-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= テンパックモード OFF

1= テンパックモード ON

解説 ユーザーアプリケーションのテンパ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パックモードを獲得する

戻値 モード
 0= テンパックモード OFF
 1= テンパックモード 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 からのコメント リモ-トセッティング でのセッティング ID 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_kxs71

要約	char *get_kxs71(void)
解説	リモ-トセッティング でのセッティング ID を獲得する
戻値	セッティング 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	ホストからのコメント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からのコメントリセットでその応答メッセージを返すかどうかを獲得する
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_kxs73

要約	int set_kxs73(unsigned char block, double c) unsigned char block : 番号(0 - 4) double channel: FM 周波数 (76.1 - 89.9, 0.0 : 7)
解説	DGPS 用の FM 周波数を設定する
戻値	0) 設定成功 1) 設定失敗 (パラメータエラー)
補足	5周波数設定可能。

get_kxs73

要約	int get_kxs73(double *array) double *array : FM 周波数 (76.1 - 89.9, 0.0:7)
解説	DGPS用のFM周波数を獲得する
戻値	なし
補足	

例:

```
#include "kme_lib.h"

void main(void)
{
    double search_ch[5];

    /* 日本でのDGPS測位モードを設定する */
    set_kxs68(2);          /* RS232C受信データの処理モードを設定する */
    set_kxs73(0, 76.5);   /* DGPS用のFM周波数を設定する */
    get_kxs73(search_ch); /* DGPS用のFM周波数を獲得する */
}
```

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	KXA あるいは 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 (パラメータエラー)
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	KXA あるいは 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フレームに変換する 1= 自動変換しない
解説	KXA/Bコマンドで生成されるメッセージを衛星がグローバルフレーム衛星ならグローバルフレー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	KXA/Bコマンドで生成されるメッセージを衛星がグローバルフレーム衛星ならグローバルフレームに自動的に変換するモードを獲得する
戻値	モード 0= 自動でメッセージがグローバルフレームに変換する 1= 自動変換しない
補足	

set_kxs76

要約	int set_kxs76(unsigned int value) unsigned int value : 仰角(0 - 45)[度]
解説	端末で衛星飛来時刻を計算する時の衛星の最低仰角を設定する
戻値	0) 設定成功 1) 設定失敗 (パラメータエラー)
補足	

get_kxs76

要約	unsigned int get_kxs76(void)
解説	衛星の最低仰角を獲得する
戻値	仰角(0 - 45 度)
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 ー外、メッセ ー 本体すべてを送信) 1= メッセ ー 本体のみを DTE に送信
解説	ハ ー ト時、ホストからの受信デー タのメッセ ー 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 (ハ ー ータエラー)
補足	

get_kxs77

要約	unsigned char get_kxs77(void)
解説	ハ ー ト時、ホストからの受信デー タのメッセ ー 本体のみをDTEに送信するモードを獲得する
戻値	モード 0= 通常 (O/Rアドレス、サブシ ー外、メッセ ー 本体すべてを送信) 1= メッセ ー 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コソト° 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-タエラー)

補足

get_kxs79

要約 int get_kxs79(void)

解説 KXB コソト° の検知時間を獲得する

戻値 検知時間 (5 – 20分)

補足

set_kxs80

要約	int set_kxs78(int sw) int sw : アウトウントグローバルランプセットのフォーマット 0= ノーマルグローバルランプ 1= エンハストグローバルランプ
解説	アウトウントグローバルランプセ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	グローバルランプセットには、ノーマルグローバルランプとエンハストグローバルランプがある。エンハストグローバルランプセットの時でも、KXS79設定によりメッセージ本体のみを取り出せるようにするため。

get_kxs80

要約	int get_kxs80(void)
解説	アウトウントグローバルランプセットのフォーマットを獲得する
戻値	アウトウントグローバルランプセットのフォーマット 0= ノーマルグローバルランプ 1= エンハストグローバルランプ
補足	

set_kxd01

要約 `int set_kxd01(unsigned char m, unsigned char hi_low)`

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

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

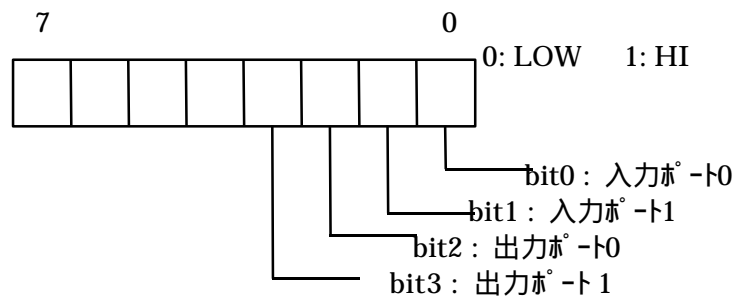
補足

get_kxd01

要約 `unsigned char get_kxd01(void)`

解説 デジタルポート出力設定を獲得する

戻値



補足

set_kxd02 はサポートされていません。

get_kxd02

要約 int get_kxd02(unsigned char ch_no, unsigned char *ch)

unsigned char ch_no : ポートの番号 (1, 2).

unsigned char *ch : ポートのデータ(0 - 255)

解説 ポートの入力値を取得する

戻値 0: 設定成功
 1: ポート番号入力エラー

補足

set_kxp01

要約 int set_kxp01(unsigned char value)

unsigned char value : モード (0 - 2)

0= なし

1= 入力ポートの状態を出力ポートにそのまま出力

2= 入力ポートの状態を反転させて出力ポートに出力

解説 入力ポートと出力ポートのリンクを設定する

戻値 0) 設定成功

1) 設定失敗 (パラメータエラー)

2) 排他エラー

補足

get_kxp01

要約 unsigned char get_kxp01(void)

解説 入力ポートと出力ポートのリンク設定を獲得する

戻値 モード (0 - 2)

0= なし

1= 入力ポートの状態を出力ポートにそのまま出力

2= 入力ポートの状態を反転させて出力ポートに出力

補足

set_kxm01

要約	int set_kxs01(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	固定メッセージを獲得する
戻値	固定メッセージへのポインタ (最大 200 バイト)
補足	

set_kxa01

要約 int set_kxa01(KXA01_CMND *pkxa01_cmd, int n)

int t: 番号 (1 - 6)
 KXA01_CMND *pkxa01_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 a: O/Rインテイクタ (1 - 3)
 char d: 送信データ

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

bit0: 測位情報
 bit1: ポート情報
 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
 bit5-7: 予備

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_kxa01

要約 int get_kxa01(KXA01_CMND *pkxa01_cmd, int n)

int t: Time Number (1 - 6)
 KXA01_CMND *pkxa01_cmd

解説 時刻指定送信(KXA01コマンド)設定を獲得する

戻値 0: 未設定
 1: 設定

補足

set_kxa02

要約 int set_kxa02(KXA02_CMND *pkxa02_cmnd)

KXA02_CMND *pkxa02_cmnd
 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 i: インタバル (1 - 10080)[分]
 int a: O/Rインデ-クタ (1 - 3)
 char d: 送信データ

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

bit0: 測位情報
 bit1: ホ-ト情報
 bit2: 固定メッセージ
 bit3: GCCへのホ-リング
 bit4: DTEへのホ-リング
 bit5-7: 予備

解説 インタバル送信を設定する

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

補足

get_kxa02

要約 int get_kxa02(KXA02_CMND *pkxa02_cmnd)

KXA02_CMND *pkxa02_cmnd

解説 インタバル送信 (KXA02コマンド) 設定を獲得する

戻値 0: 未設定
 1: 設定

補足

set_kxa03

要約 int set_kxa03(KXA03_CMND *pkxa03_cmnd)

KXA03_CMND *pkxa03_cmnd

int lm: インタバル (1 - 1440)[分]

int a: O/Rインテイク (1 - 3)

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へのホートリンク

bit4: DTEへのホートリンク

bit5-7: 予備

解説 衛星飛来送信を設定する

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_kxa03

要約 int get_kxa03(KXA03_CMND *pkxa03_cmnd)

KXA03_CMND *pkxa03_cmnd

解説 衛星飛来送信 (KXA03コマンド) 設定を獲得する

戻値
 0: 未設定
 1: 設定

補足

set_kxa04

要約 int set_kxa04(KXA04_CMND *pkxa04_cmnd)

KXA04_CMND *pkxa04_cmnd
 int s: ON/OFF (0:ON, 1:OFF)
 int l: インタ-バル (1 - 1440)[分]
 int to: タイムアウト (1 - 3600)[秒]
 int a: O/Rインテ-イク-タ (1 - 3)

解説 ハ° イト-ト° 設定を設定する

戻値 0) 設定成功
 1) 設定失敗 (ハ° ラメ-タエラー)
 2) 排他エラー

補足

get_kxa04

要約 int get_kxa04(KXA04_CMND *pkxa04_cmnd)

KXA04_CMND *pkxa04_cmnd

解説 ハ° イト-ト° 設定 (KXA04コマンド°) を獲得する

戻値 0: 未設定
 1: 設定

補足

set_kxa05

要約 int set_kxa05(KXA05_CMND *pkxa05_cmnd, int p)

KXA05_CMND *pkxa05_cmnd
 int tr: トリガ - (0:OFF, 1:LOW -> HI, 2:HI->LOW, 3:HI<->LOW)
 int a: O/Rインデイクタ (1 - 3)
 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へのホートリンク
 bit4: DTEへのホートリンク
 bit5-7: 予備

int p: デジタル入力ポート番号(0, 1)

解説 入力ポート変化で送信を設定する

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_kxa05

要約 int get_kxa05(KXA05_CMND *pkxa05_cmnd, int n)

KXA05_CMND *pkxa05_cmnd
 int n: デジタル入力ポート番号 (0, 1)

解説 入力ポート変化で送信 (KXA05コメント) を獲得する

戻値 0: 未設定
 1: 設定

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 - 3)
 char d: 送信データ

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

bit0: 測位情報
 bit1: ポート情報
 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: 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_cmnd)

KXB02_CMND *pkxb02_cmnd
 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 i: インターバル (1 - 10080)[分]
 int c: 検知コード
 int a: O/Rインテイクタ (1 - 3)
 char d: 送信データ

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

bit0: 測位情報
 bit1: ポート情報
 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: 1-ザ-アプリ起動
 bit7: 予備

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_cmnd)

KXB02_CMND *pkxb02_cmnd

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_kxb03

要約 int set_kxb03(KXB03_CMND *pkxb03_cmnd)

KXB03_CMND *pkxb03_cmnd
 int lm: インタバル (1 - 1440)[分]
 int c: 検知コード
 int a: O/Rインテイクタ (1 - 3)
 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: 予備

解説 衛星飛来送信を設定する

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_kxb03

要約 int get_kxb03(KXB03_CMND *pkxb03_cmnd)

KXB03_CMND *pkxb03_cmnd

解説 衛星飛来送信 (KXB03フラグ) を獲得する

戻値 0: 未設定
 1: 設定

補足

set_kxa00

要約 void set_kxa00(void)

解説 KXA01 - 05 マント° の設定を解除する

戻値 なし

補足

set_kxb00

要約 void set_kxb00(void)

解説 KXB01 - 03 マント° の設定を解除する

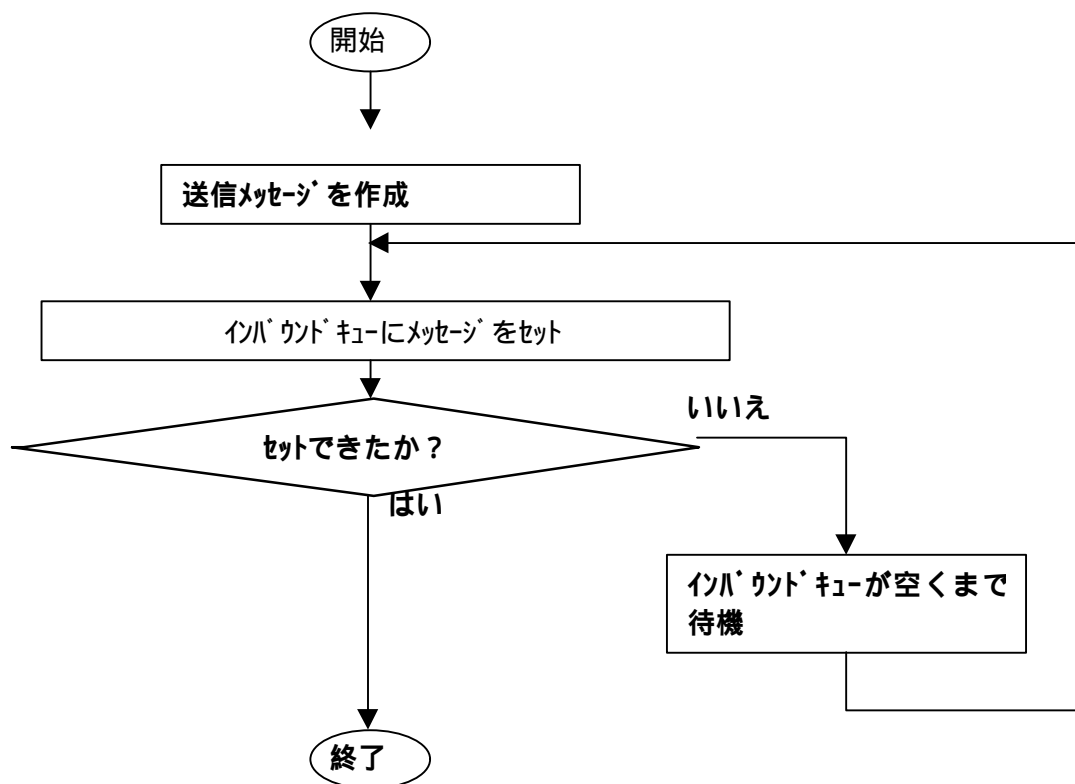
戻値 なし

補足

7. プログラミング ヒント

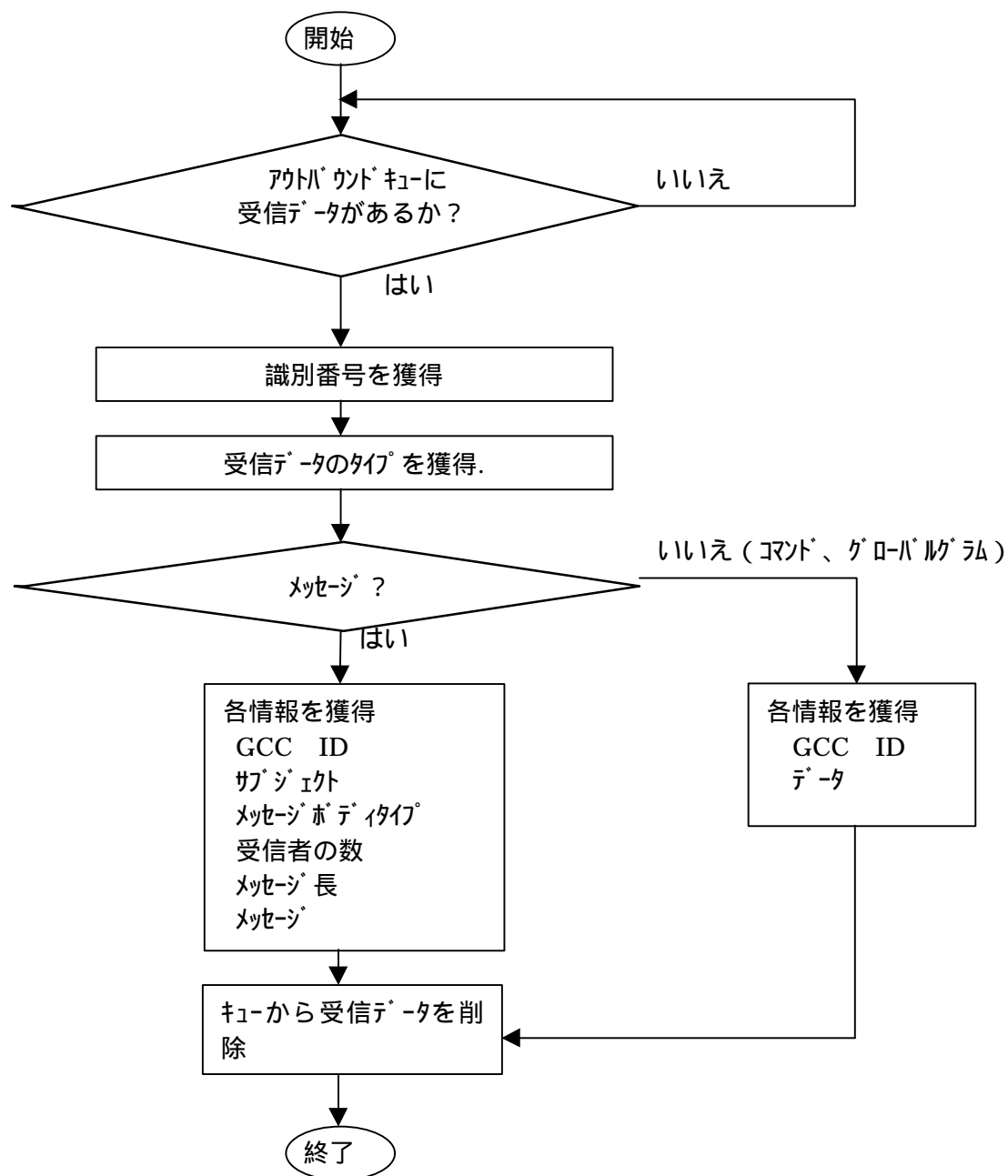
7.1. GCCへのメッセージ送信

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ためには、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確認する。
 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。



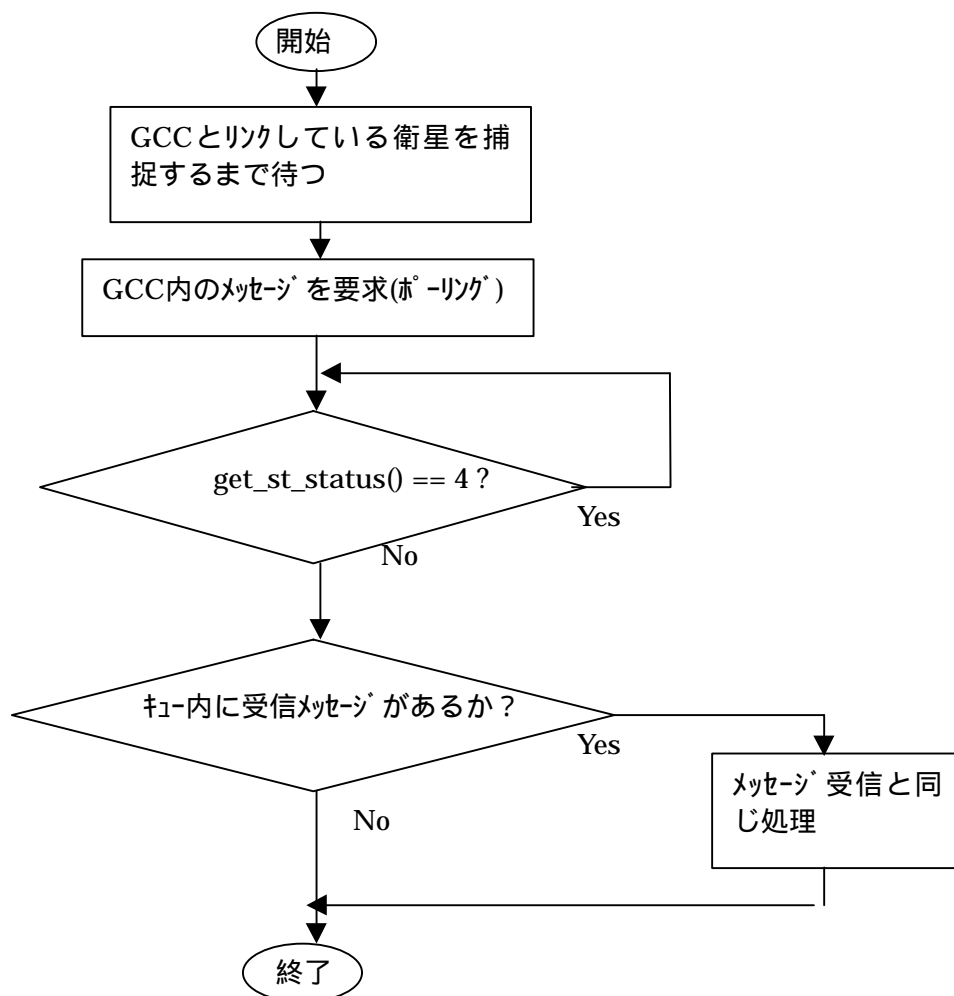
7.2. GCCからのメッセージ受信

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チェックする。
2. もし受信データがあれば、そのタイプ(メッセージコマンド)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

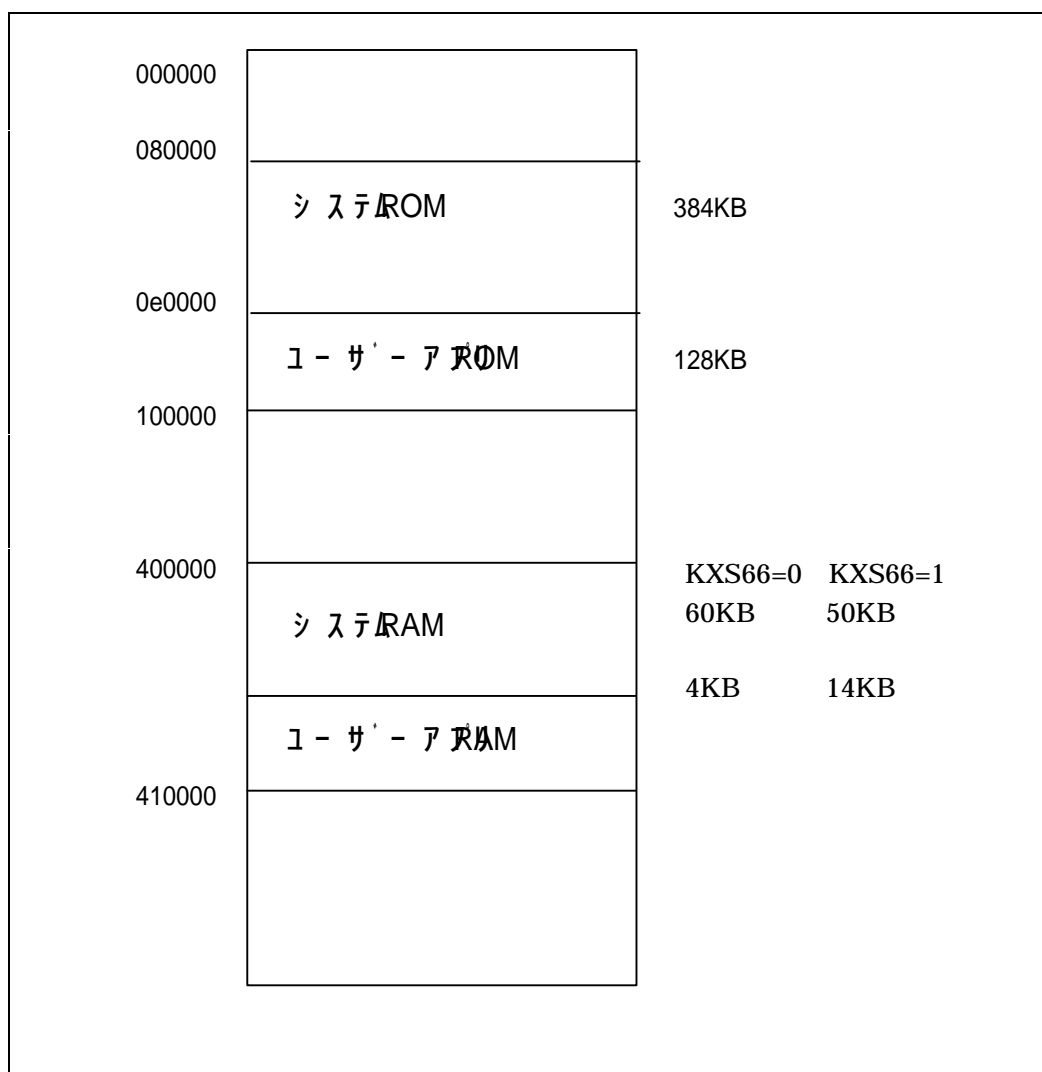
1. GCCとリンクしている衛星を捕捉しているか確認する。
2. “all_msg_polling”をコールしてGCC内のメッセージを要求する。
3. GCCからの応答を待つ。この時SCは、応答待ちしている間“get_st_status()”の戻り値は4になる。
4. メッセージがあればGCCから送信され、そのメッセージは自動的にアウトバウンドキューに格納される。
5. “get_st_status()”の戻り値が0に変わるのをまって“get_unget_ob_id_for_user”をコールし、アウトバウンドキューに受信データがあるかをチェックする。
6. もし受信データがあれば、そのタイズ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



8. メモリーマップ

システムには、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がある。
 ユーザーアプリケーションROMのサイズは8KB (0xe0010 - 0x10000)、RAMのサイズはKXS66 コマンドで選択可能であり、4KB(0x40f000 - 0x41000)、あるいは16KB(0x40c800 - 0x41000)である。
 但し、RAMのサイズが4KBの時にはドッブラー測定使用できなくなる。

このユーザーアプリケーションRAM領域は、ユーザーアプリケーションプログラムをインストールする時に0クリアされる。



従って、リンクするには以下のように設定しなくてはならない。

T_TEXT_START=e0000	スタートアップ関数の先頭アドレス
T_TEXT=e0010	ユーザーアプリケーションの先頭アドレス
T_GBSS_USER=40f000	ユーザーアプリケーションの先頭アドレス
	*4KBの時、 40f000
	16KBの時、 40c800

ユーザーアプリケーションのスタック領域はRAM上に割り当てられおり、サイズは500バイトである。スタック領域の先頭アドレスは“stack”で確認できる。

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おこす可能性があるため、大きいサイズの変数(例えば受信バッファ等)は、グローバル変数として宣言したほうが好ましい。

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、ユーザーはそのプログラムをデバッグすることができる。そのためには、端末とパソコンをRS232Cケーブルで接続して、パソコンで通信の例えば“ハイパーターミナル”を起動する必要があり、そうすれば端末からいろいろなデバッグ情報を獲得できる。

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る。

1. メモリダンプ
2. メモリ書き込み
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時に有効となる。デバッグモードにするためにはKXS70コマンドを使用する。

デバッグモード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点灯する。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る。

9.1. メモリダンプ

メモリ内容を6進で表示する。

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む。

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する。ユーザーアプリのスタック領域は上位割け出でられおり、サイズは500バイトである。

入力コマンド : stack

<例>

> stack
 4020d8

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る。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る。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る。

<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 - 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]

ブレークした後に、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

あたかもGCCから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る。メッセージ本体のほかに、サブシフトインディケータアドレスが設定できる。

<入力コマンド>

```
obreg [レシビエ1],[レシビエ2],[サブシフト],<T or H>[メッセージ]<R><LF>
      [レシビエ2]           :O/RインディケータアドレスO/Rインディケータの時には
                           先頭
                           に '@'を付加する。
      [サブシフト]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
      [メッセージ]        :メッセージ本体
```

* “obreg” コマンドは、最大56バイトです。

<例>

- サブシフト: TEST1
レシビエO/Rインディケータ
メッセージ: "12345"

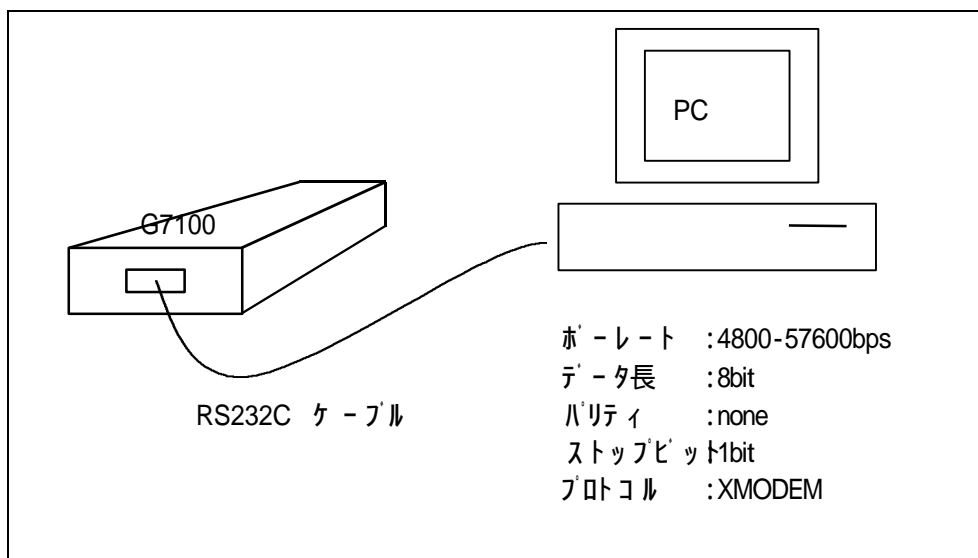
```
obreg @1,,TEST1,<T>12345
```

- サブシフト: TEST2
レシビエO/Rインディケータ-O/Rアドレス“SUM”
メッセージ: "0x12,0x34,0x56,0xab"

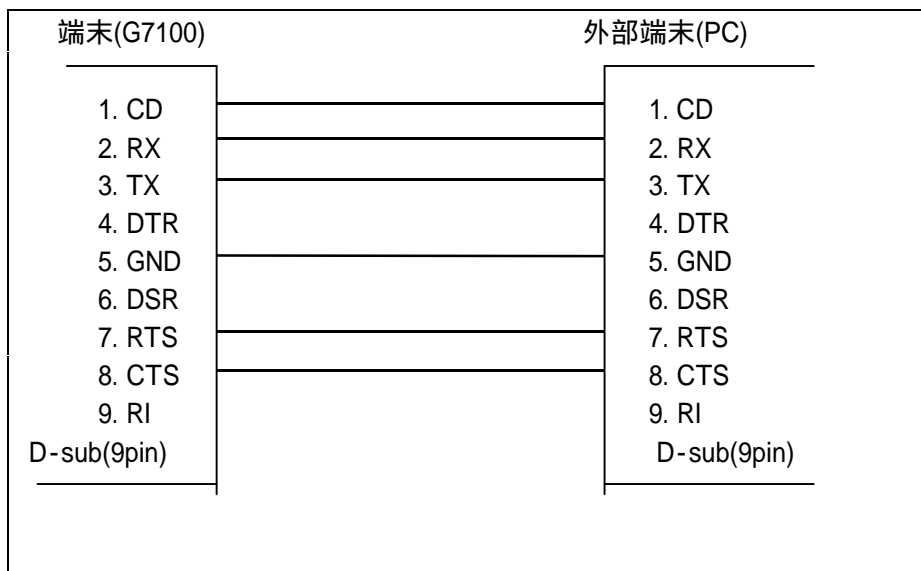
```
obreg SUM,@1,TEST2,<H>123456ab
```

10. 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(0x0D0000)インストールする。



10.1. 接続



10.2. 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する
2. CTRL + KXORBと入力してコマンドモードに入る
3. CTRL + UALDRと入力してインストールモードに入る
PCに以下のメッセージが表示される
“*** User Application Software Installation Mode *** “
“Select SC's RS232C bps (0:Continue/ 1:9600/ 2:19200/ 3:38400/
4:57600/ 5:Exit)”
4. 番号(0-5)を入力してボーレートを選択する
PCに以下のメッセージが表示される
“Match the modem bps to this, then reconnect.”
“Start Sending User Application Software within 10sec.”
5. PCからユーザーアプリケーションプログラムを送信する。
インストール終了後Cに以下のメッセージが表示される
“Installation of User Application Software was Completed. Reset
power switch”
6. 端末の電源を入れ直す。

10.3. 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する
2. CTRL + KXORBと入力してコマンドモードに入る
3. CTRL + UADLTと入力する
PCに以下のメッセージが表示される
“User application area Initialize OK ? <Y/N>”
4. “Y”を押す
“Initialized : OK”
“Reset power switch”
5. 端末の電源を入れ直す

10.4. 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S232Cケーブルで接続し、ハイパーターミナルを起動する。
2. CTRL +KXORBと入力してコマンドモードに入る。
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る。

```

>KXCHK

EEPROM      : OK
RAM         : OK
LOOP        : OK
ASIC        : OK
RTC         : OK
SYNTHE      : OK
ROM Ver.    : X2C1A-012

1. USER APPL : 37EA(ACTIVE)
.....

```

“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ェックサムと動作状態を表示し、インストールされていない場合は“NONE”と表示する。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“NO ACTIVE”を表示する。

11. ユーザーアプリケーションのプログラム開発上の注意点

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、40f000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“sprintf()”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。“sprintf()”をコールした時スタックが溢れ、プログラムの暴走する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“sprintf()”の使用を避けて下さい。
- ユーザーアプリケーションプログラムが暴走した場合、**RETR**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。このときのPCの通信モードは以下にしてから入力してください。

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

Appendix A. ユーザーアプリケーションの開発環境

- **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	NEC PC9801、PC-ATおよびその互換機
CPU	80386以上
メモリ	5.6MB 以上
ハードディスク	3MB以上

- **ソフトウェア**

松下電子工業製（MEC）の以下のソフトウェア/ドキュメントが必要です。

MN10200 シリーズ C コンパイラ
MN10200 シリーズ アセンブラ
MN10200 シリーズ リンカ
MN10200 シリーズ ライブラリ
ファイル変換ユーティリティ
ユーザーマニュアル

九州松下電器製（KME）の以下のソフトウェア/ドキュメントが必要です。

KME ライブラリ
ユーザーアプリケーションのインストール
サンプルプログラム

Appendix B. PCの環境設定

1. 環境変数の設定

コンパイルが動作できるように、環境変数ATHの内容を書き替えて下さい。また、中間ファイルを作成するディレクトリとして環境変数IPを設定して下さい。

2. CONFIG.SYSの書き換え

CONFIG.SYSが、あらかじめ FILES、BUFFERS指定を持たない場合、次のように書き加えてください。また、すでに存在する場合、確認して必要があれば変更してください。

```
FILES = 20
BUFFERS = 20
```

3. 環境設定ファイル

環境設定ファイル(C102L.rc)では以下の5項目が設定できます。もしこのファイルがなければ作成して下さい。

- (1) ソースファイル内で絶対パスが指定されていないインクルードファイルの標準ディレクトリ
- (2) ドライバからリンクを起動した際に自動的にリンクされるライブラリファイル
- (3) リンクで自動的にリンクされるライブラリファイルの標準ディレクトリ
- (4) テンポラリファイルを生成する標準ディレクトリ
- (5) ドライバが起動する各コマンドの標準ディレクトリ

<例>

```
temp      c:¥cc102l
path      c:¥cc102l¥bin
include   c:¥cc102l¥include
libdir    c:¥cc102l¥lib
stdlib    cc102l.lib
```

環境設定ファイル(C102L.rc)は “cc102l¥bin “に置きます。

変更履歴 (Ver2.0 Ver3.0)

- ・ 章番号について英語版と統一のため全面打ち直し
- ・ 2章全体について、図及び記載内容改定

変更履歴 (Ver1.0 Ver2.0)

変更頁	変更内容
3	開発ツールの入手先を変更 松下電子工業(株) ソフィアシステムズ(株)
11	go_sleep_next_path()関数名の記述間違いを修正 (誤) "pass" (正) "path"
20	all_msg_polling()のプログラミング例追記
21	s_msg_polling() のプログラミング例追記
22	globalgram_polling() のプログラミング例追記
23	chk_failed_polling()の戻り値(-1)の説明追記
30	req_pos_report_to_ncc()の戻り値(-1)の説明追記
31	req_select_next_downlink()の戻り値の記述間違いを修正 (誤) 0:... 1:... (正)戻り値無し
32	chk_polling_event()の戻り値の説明変更 (誤) 0:無し 1:あり (正) 1以外:無し 1: あり
43	get_unget_ob_id_for_user()のプログラミング例の間違いを修正
45	remove_ob_q()の補足説明を追記
48	get_ob_msg_subject_ind()の戻り値の説明追記
59	go_sleep_time()の補足説明を追記
60	go_sleep_next_path()の補足説明を追記
66	get_rx_data() の補足説明を追記
76	set_digital_port() のプログラミング例の間違いを修正
79	req_pos_calc_from_user()の戻り値の記述間違いを修正 (誤) 0:測位失敗 1: 測位開始 (正) 0: 測位開始 1: 測位失敗
91	get_st_status() の補足説明を追記
118	get_sys_info()の関数定義の間違いを修正 (誤) int get_sys_info() (正) char get_sys_info()
215	GCCへのホリスティック処理のフローチャートを修正
219	ブレークポイントによるサスペンド状態からのレジュームコマンドの記述を修正 (誤) "go" (正) "g"
224	ユーザーアプリケーションのプログラミング 開発上の注意点に説明を追加