オーブコム端末装置
KX-G7100/N，KX-G7101/N
ユーザーアプリケーション
プログラムミングガイド

Ver 4.0

２００３年1月６日

オーブコムジャパン株式会社

目次

1. 概要

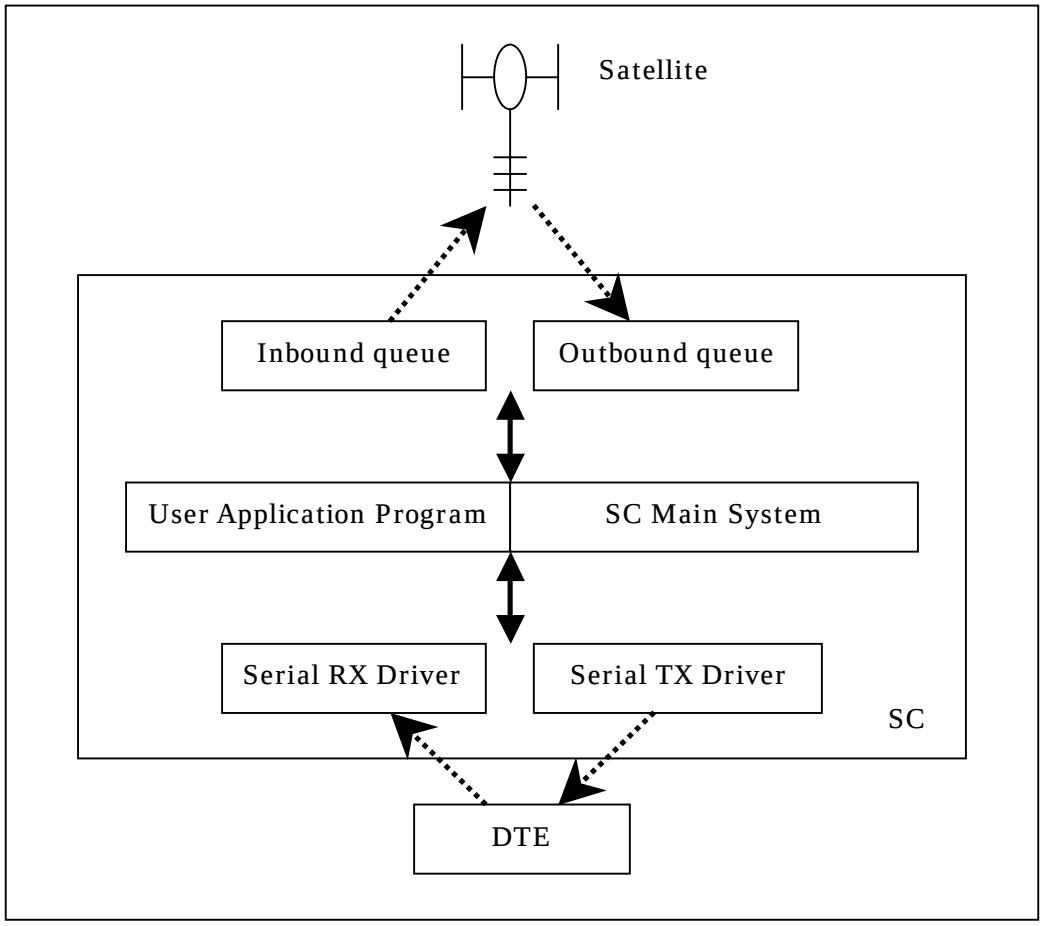
ﾊﾟﾅｿﾆｯｸｺﾐｭﾆｹｰｼｮﾝｽﾞ株式会社製オーブコム端末装置（KX-G7100/N，KX-G7101/N 以下SC）は、ユーザーアプリケーション用に１２８KバイトのROMと４KバイトのRAMを装備しています。ユーザーアプリケーションは、KMEライブラリを使ってSCがもつ情報（例えば、デジタルI/O状況、SCの衛星捕捉状況など）の参照、SCへの測位要求、そしてメッセージの送受信ができます。また、この開発したプログラムは、ローダーを使ってSC内ROMに容易にロードする事ができ、SCの動作を使用環境に合うようにできます。

ユーザアプリケーションを作成するにあたっては、松下電器産業(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＞

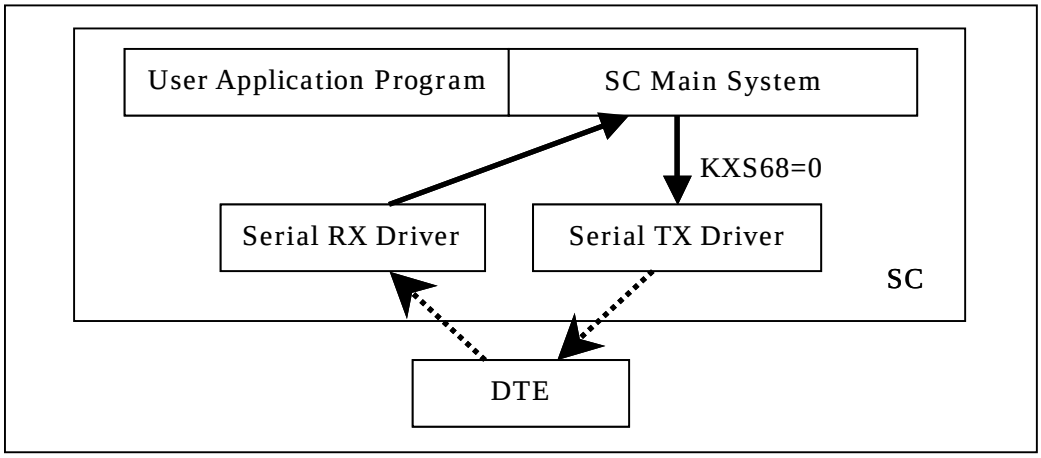
このSCのメインシステムは、常時起動していますので、ユーザーアプリケーションが衛星間通信やDTE間通信を行うような場合は、メインシステムとユーザーアプリケーションでデータの競合がおきないようにSCのデータアクセス権の設定を行う必要があります。

2.1. DTEとSC間の通信制御

SCは、シリアルの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

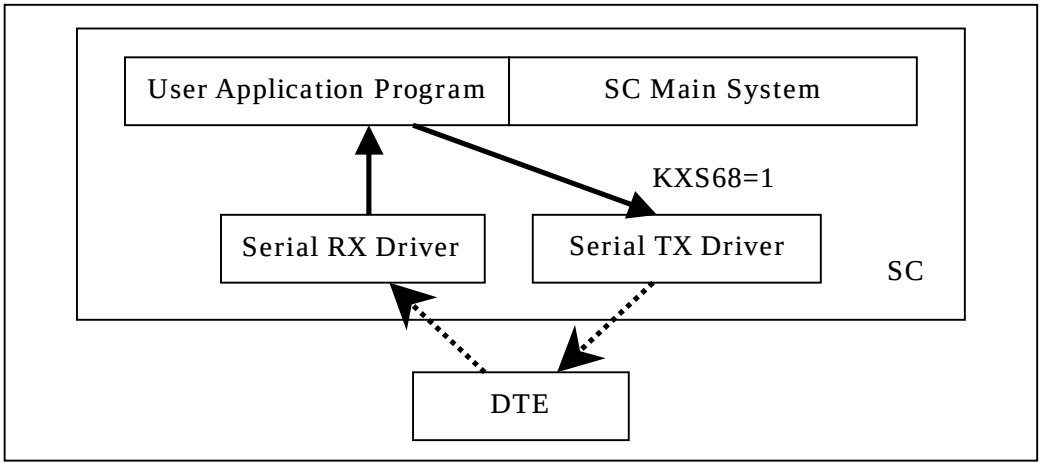
この場合、KXS68の設定を０（KXS68=0）にします。
ユーザーアプリケーションでDTEとの通信をする必要がない場合、あるいはDTE間の通信はメインシステム（プロトコルモード・バイトモード）で行わせたい場合に設定します。



＜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。



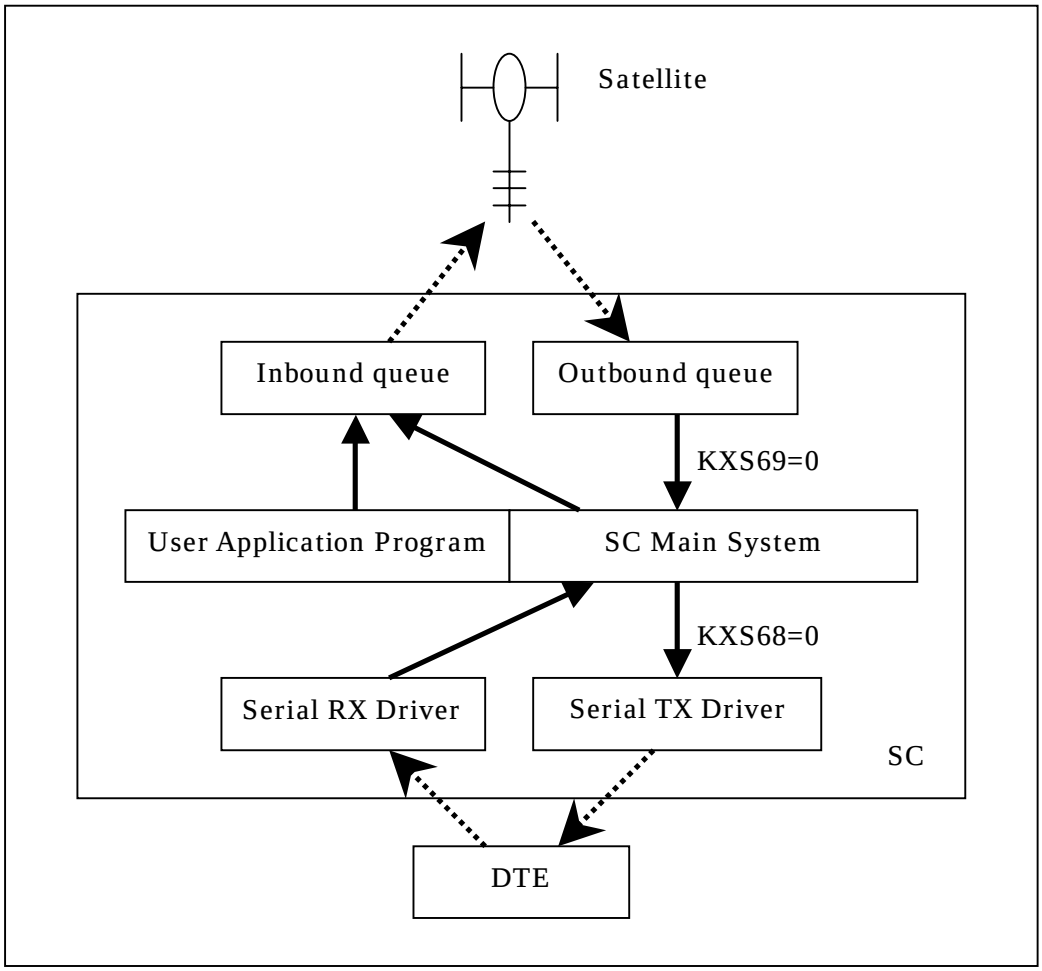
＜図２．３＞

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。



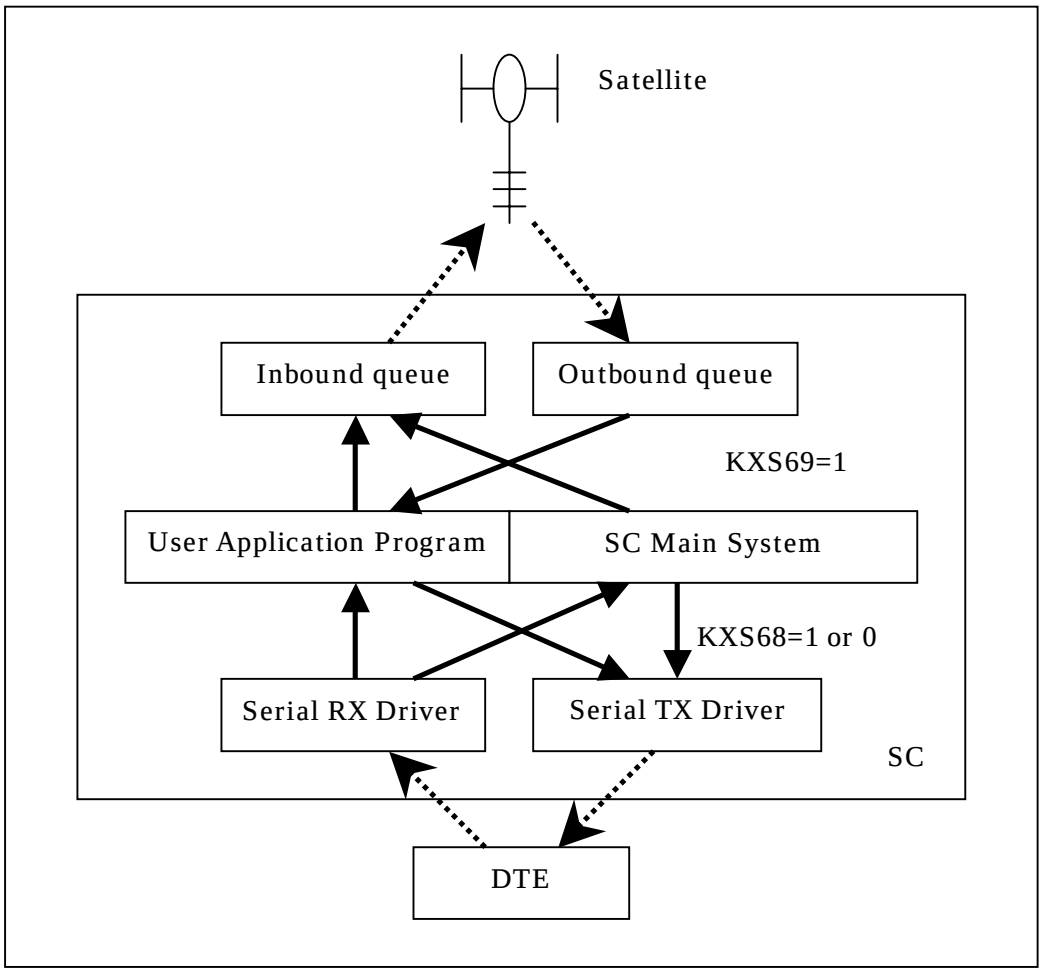
＜図２．４＞

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

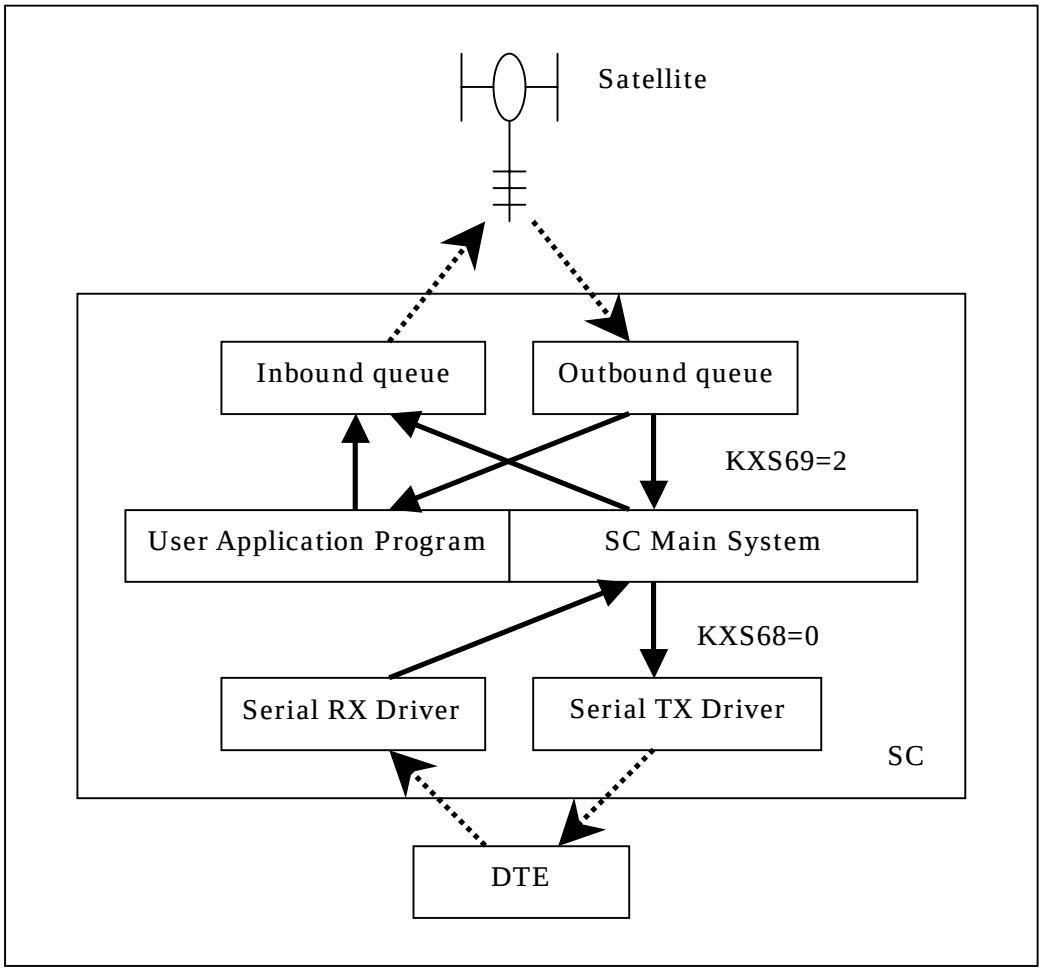
KXS69の設定を１（KXS69=1）にします。
ユーザーアプリケーションが衛星とのメッセージ送受信を行う場合に設定します。
ユーザーアプリケーションは、アウトバウンドの全てのメッセージを獲得し、獲得後、そのメッセージをキューから削除します。メインシステムからアウトバウンドキューに対してはアクセスできません。シリアルドライバに対してはKXS68の設定に従います。(2.1章参照)



＜図２．５＞

2.2.3. メインシステムとﾕｰｻﾞｰｱﾌﾟ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がメインシステムとユーザーアプリケーションそれぞれ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。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することが出来、処理できるメッセージだけを処理します。



＜図２．６＞

注意：

2. KXS69=2の場合メインシステム、ユーザーアプリケーション双方のアウトバウンドキュー処理が終わらないとキューが消去できません。
3. KXS69=2、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。

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器産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が、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. 自動送信、動態管理機能で指定された条件で起動

KXA/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。

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。

5.3. System Announce

ホスト局からのポーリングをチェックします

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ｼﾞをｾｯﾄする	39
req_send_ib_report()	送信ｷｭｰにﾚﾎﾟｰﾄをｾｯﾄする	41
req_send_ib_globalgram()	送信ｷｭｰにｸﾞﾛｰﾊﾞﾙﾒｯｾｰｼﾞをｾｯﾄする	43
req_send_ib_pos_report()	送信ｷｭｰにﾎﾟｼﾞｼｮﾝﾚﾎﾟｰﾄをｾｯﾄする	44

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ｼﾞをｾｯﾄする.	45

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()	ｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰのﾃﾞｰﾀのﾎﾟｲﾝﾀを獲得する	47
remove_ob_q()	ｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰからﾃﾞｰﾀを削除する。	50
get_ob_type()	ﾃﾞｰﾀ種を獲得する	51
get_ob_ncc_id()	ﾒｯｾｰｼﾞのGCC__IDを獲得する	52
get_ob_msg_subject_ind()	ﾒｯｾｰｼﾞのｻﾌﾞｼﾞｪｸﾄを獲得する	53
get_ob_msg_msg_body_type()	ﾒｯｾｰｼﾞのﾒｯｾｰｼﾞﾀｲﾌﾟを獲得する	54
get_ob_msg_or_quan()	ﾒｯｾｰｼﾞの送信元を獲得する	55
get_ob_msg_msg_len()	ﾒｯｾｰｼﾞ長を獲得する	56
get_ob_msg_msg_body_index()	ﾒｯｾｰｼﾞ本体のｲﾝﾃﾞｸｽを獲得する	57
get_ob_message ()	ﾒｯｾｰｼﾞを獲得する	58
get_ob_user_command()	ｺﾏﾝﾄﾞを獲得する	59
get_ob_globalgram_ref_num()	ｸﾞﾛｰﾊﾞﾙﾒｯｾｰｼﾞの認識番号を獲得する	60
get_ob_globalgram_or_ind()	ｸﾞﾛｰﾊﾞﾙﾒｯｾｰｼﾞの送信元を獲得する	61
get_ob_globalgram_msg_len()	ｸﾞﾛｰﾊﾞﾙﾒｯｾｰｼﾞのﾒｯｾｰｼﾞ長を獲得する	62
get_ob_globalgram()	ｸﾞﾛｰﾊﾞﾙﾒｯｾｰｼﾞを獲得する	63

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.	64
go_sleep_next_path()	次に衛星が飛来するまでスリープさせる.	65
go_wait ()	指定時間ユーザソフトをサスペンドする.	66

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	67
rf_block_power_off()	RFブロック電源をOFFする	68
chk_rf_block_power()	RFブロックの電源状態を獲得する	69

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。	70
get_rx_data()	RS232Cの受信バッファからデータを獲得する。	71
chk_tx_ready()	RTSポート状態を獲得する	72
set_tx_data()	RS232Cの送信バッファにデータをセットする	73
cts_act()	CTSポートをアクティブにする	74
cts_neg()	CTSポートをネガティブにする	75
chk_cd()	ﾒｲﾝﾎﾟｰﾄのCD信号の状態をﾁｪｯｸする	76
chk_rts()	ﾒｲﾝﾎﾟｰﾄのRTS信号の状態をﾁｪｯｸする	77

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	78
set_timer()	タイマをセットする	79
check_timer()	セットしたタイマが経過したか確認する	81

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	82
set_digital_port()	デジタルポートをセットする	83

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.	84

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	85
req_pos_calc_from_user()	測位をさせる	86
chk_pos_calc_result()	測位が完了したかチェックする	87
get_pos_quality_ind()	測位品質を獲得する	88
get_pass_quan()	ドプッラー測位で使用したパスの数を獲得する	89
get_pos_age()	前回測位してからの経過時間を獲得する	90
get_lat_val()	測位結果（緯度）を獲得する	91
get_lon_val()	測位結果（経度）を獲得する	92
set_lat_val()	端末の保持している測位情報(緯度)をセットする	93
set_lon_val()	端末の保持している測位情報(経度)をセットする	94
set_pdop_th()	PDOPのしきい値をセットする	95
get_pdop_th()	PDOPのしきい値を獲得する	96

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	97
get_x06_str()	X06 センテンスを獲得する	99

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。

5.13. Other Functions

orb_sprintf7()	数値の文字列変換	139

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。

ﾗｲﾌﾞﾗﾘｰ関数		内容	ﾍﾟｰｼﾞ
設定関数	獲得関数		
set_kxs01()	get_kxs01()	ﾃﾞﾌｫﾙﾄ GCC番号	140
set_kxs02()	get_kxs02()	ﾃﾞﾌｫﾙﾄ ﾎﾟｰﾙﾄﾞ	141
set_kxs03()	get_kxs03()	ﾃﾞﾌｫﾙﾄ ﾌﾟﾗｲｵﾘﾃｨ	142
set_kxs04()	get_kxs04()	ﾃﾞﾌｫﾙﾄ ﾚﾎﾟｰﾄ O/R ｲﾝﾃﾞｨｹｰﾀ	143
set_kxs05()	get_kxs05()	ﾃﾞﾌｫﾙﾄ ﾒｯｾｰｼﾞ/ｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑ O/R ｲﾝﾃﾞｨｹｰﾀ	144
set_kxs06()	get_kxs06()	ﾃﾞﾌｫﾙﾄ ack ﾚﾍﾞﾙ	145
set_kxs07()	get_kxs07()	ﾃﾞﾌｫﾙﾄ ﾒｯｾｰｼﾞ ﾎﾞﾃﾞｨ ﾀｲﾌﾟ	146
set_kxs08()	get_kxs08()	ﾃﾞﾌｫﾙﾄ ｻｰﾋﾞｽ ﾀｲﾌﾟ	147
---------	---------		
set_kxs10()	get_kxs10()	ﾎﾟｰﾘﾝｸﾞ複数ﾚﾎﾟｰﾄのｲﾝﾀｰﾊﾞﾙ	148
set_kxs11()	get_kxs11()	ﾎﾟｰﾘﾝｸﾞ複数ﾚﾎﾟｰﾄの数	149
set_kxs12()	get_kxs12()	ﾎﾟｰﾘﾝｸﾞ複数測位ﾚﾎﾟｰﾄのｲﾝﾀｰﾊﾞﾙ	150
set_kxs13()	get_kxs13()	ﾎﾟｰﾘﾝｸﾞ複数測位 ﾚﾎﾟｰﾄの数	151
set_kxs14()	get_kxs14()	GCCｻｰﾁﾓｰﾄﾞ	152
set_kxs15()	get_kxs15()	ﾀﾞｳﾝﾘﾝｸﾁｬﾝﾈﾙ	153
set_kxs16()	get_kxs16()	ﾀﾞｳﾝﾘﾝｸﾁｪｯｸｻﾑｴﾗｰｽﾚｯｼｭ	154
set_kxs17()	get_kxs17()	ﾀﾞ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	155
set_kxs18()	get_kxs18()	連続測位ﾓｰﾄﾞ	156
set_kxs19()	get_kxs19()	ﾄﾞｯﾌﾟﾗｰ測位時のｴﾌｪﾒﾘｰ数とその収集間隔	157
set_kxs20()	get_kxs20()	測位結果の有効時間	158
set_kxs21()	get_kxs21()	測位結果の最小有効ｸｫﾘﾃｨｲﾝﾃﾞｨｹｰﾀ	159
set_kxs22()	get_kxs22()	軌道要素の有効時間orbit elements age	160
set_kxs23()	get_kxs23()	緯度・経度	161
set_kxs24()	get_kxs24()	測位ﾓｰﾄﾞ	162
set_kxs25()	get_kxs25()	KXA/BｺﾏﾝﾄﾞでのGPS測位情報のﾌｫｰﾏｯﾄ	163
set_kxs26()	get_kxs26()	DTEに対するﾎﾟｰﾘﾝｸﾞの応答の待ち時間	164
set_kxs27()	get_kxs27()	ｼﾘｱﾙﾌﾟﾛﾄｺﾙのACK時間	165
set_kxs28()	get_kxs28()	ｼﾘｱﾙﾌﾟﾛﾄｺﾙのﾘﾄﾗｲ回数	166
set_kxs29()	get_kxs29()	ｱﾎﾞｰﾄﾚﾎﾟｰﾄ送信ﾓｰﾄﾞ	167
set_kxs30()	get_kxs30()	ｱﾎﾞｰﾄﾚﾎﾟｰﾄの内容	168
set_kxs31()	get_kxs31()	RS232Cの通信ﾓｰﾄﾞ （ﾌﾟﾛﾄｺﾙ/ﾊﾞｲﾄﾓｰﾄﾞ）	170
set_kxs32()	get_kxs32()	ﾊﾞｲﾄﾓｰﾄﾞﾄﾘｶﾞｰ	172
set_kxs33()	get_kxs33()	ﾊﾞｲﾄﾓｰﾄﾞﾀｲﾑｱｳﾄ	173
set_kxs34()	get_kxs34()	ﾊﾞｲﾄﾓｰﾄﾞ長	174
set_kxs35()	get_kxs35()	ﾊﾞｲﾄﾓｰﾄﾞ時の、送信・受信SOM,EOM	175
set_kxs36()	get_kxs36()	ﾊﾞｲﾄﾓｰﾄﾞﾒｯｾｰｼﾞﾀｲﾌﾟ	176
set_kxs37()	get_kxs37()	ﾊﾟﾜｰﾀﾞｳﾝﾓｰﾄﾞ	177
set_kxs38()	get_kxs38()	ﾊﾟﾜｰﾀﾞｳﾝﾐﾆﾏﾑｲﾝﾀｰﾊﾞﾙ	178
set_kxs39()	get_kxs39()	ｲﾝｱｸﾃｨﾌﾞｲﾝﾀｰﾊﾞﾙ	179
set_kxs40()	get_kxs40()	ﾊﾟﾜｰｾｰﾌﾞﾓｰﾄﾞ	180
set_kxs41()	get_kxs41()	ｲﾝﾊﾞｳﾝﾄﾞﾌﾛｰ制御	181
set_kxs42()	get_kxs42()	ｱｳﾄﾊﾞｳﾝﾄﾞﾌﾛｰ制御	182
set_kxs43()	get_kxs43()	RS232C通信ﾓｰﾄﾞ （ﾎﾞｰﾚｰﾄ/ﾊﾟﾘﾃｨ/etc）	183
set_kxs44()	get_kxs44()	RS232C通信ﾓｰﾄﾞ （半２重、全２重）	184
set_kxs45()	get_kxs45()	ｲﾝﾊﾞｳﾝﾄﾞﾒｯｾｰｼﾞﾄﾘｰﾄﾒﾝﾄ	185

set_kxa05()	get_kxa05()	入力ﾎﾟｰﾄ変化で送信（KXA05ｺﾏﾝﾄﾞ）	237
set_kxb01()	get_kxb01()	時刻指定送信（KXB01ｺﾏﾝﾄﾞ）	238
set_kxb02()	get_kxb02()	ｲﾝﾀｰﾊﾞﾙ送信（KXB02ｺﾏﾝﾄﾞ）	239
set_kxb03()	get_kxb03()	衛星飛来送信（KXB03ｺﾏﾝﾄﾞ）	240
set_kxa00()	---------	KXA01 - 05ｺﾏﾝﾄﾞの設定を解除	241
set_kxb00()	---------	KXB01 - 03ｺﾏﾝﾄﾞの設定を解除	241

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 ｺﾏﾝﾄﾞﾓｰﾄﾞの KXSO1 ｺﾏﾝﾄﾞでも変更できます。

set_pin_code

要約 int set_pin_code(unsigned long pin_code);

unsigned long pin_code: ﾋﾟﾝｺｰﾄﾞ (0 - 9999)

解説 端末ｰｹﾞｰﾄｳｴｲ間通信のﾊﾟｽﾜｰﾄﾞを設定する

戻値 なし

補足 ｺﾏﾝﾄﾞﾓｰﾄﾞの KXS50 ｺﾏﾝﾄﾞでも変更できます。
ﾋﾟﾝｺｰﾄﾞは GCC と端末に設定すします。GCC にてﾋﾟﾝｺｰﾄﾞを参照するようになっている時は、これら２つのﾋﾟﾝｺｰﾄﾞが合っていないと通信できません。

set_ncc_search_mode

要約 `int set_ncc_search_mode( int ncc_search_mode );`

`int ncc_search_mode : ｻｰﾁﾓｰﾄﾞ ( 0 - 4 )`

- 0 : 希望GCCを連続的にﾀﾞｳﾝﾘﾝｸﾊﾞﾝﾄﾞの中から検索する
- 1 : 希望GCCを１回探す。もし 見つからなければ最初に発見したﾀﾞｳﾝﾘﾝｸとのﾛｯｸを維持する。
- 2 : 最初に発見したﾀﾞｳﾝﾘﾝｸとのﾛｯｸを維持する。
- 3 : 希望GCC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、最初に発見したﾀﾞｳﾝﾘﾝｸとのﾛｯｸを維持する。
- 4 : 希望GCCを１回探す。もし見つからなければｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑ衛星かkxs01に設定されているGCCのﾀﾞｳﾝﾘﾝｸを検索し続ける。

解説 GCCｻｰﾁﾓｰﾄﾞを設定する。

戻値
0 : 設定成功
1 : 設定失敗

補足 ｺﾏﾝﾄﾞﾓｰﾄﾞの KXS14 ｺﾏﾝﾄﾞでも設定できます。希望 GCC は set_kxs01 または set_desired_ncc_id で設定します

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ｼﾞ/ｺﾏﾝﾄﾞを要求する

戻値　　0 : ｺﾏﾝﾄﾞ正常受信。ﾎﾟｰﾘﾝｸﾞ開始
　　1 : 衛星を捕捉してない等でﾎﾟｰﾘﾝｸﾞできない

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ｼﾞ/ｺﾏﾝﾄﾞを要求するものであり、GCC がこのｺﾏﾝﾄﾞを受信するとﾒｯｾｰｼﾞ/ｺﾏﾝﾄﾞの送信を開始します。端末が受信するとｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰに格納されます。

　　ﾎﾟｰﾘﾝｸﾞは、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がﾎﾟｰﾘﾝｸﾞ処理を始めると”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、ﾒｯｾｰｼﾞはなかった。 */
            }
        }
    }
}
```

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ｼﾞ(150ﾊﾞｲﾄ以下)/ｺﾏﾝﾄﾞを要求する。

戻値　　0 : ｺﾏﾝﾄﾞ正常受信。ﾎﾟｰﾘﾝｸﾞ開始
1 : 衛星を捕捉していない等でﾎﾟｰﾘﾝｸﾞできない

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ｼﾞ（150ﾊﾞｲﾄ以下）/ｺﾏﾝﾄﾞを要求するものであり、GCC がこのｺﾏﾝﾄﾞを受信するとﾒｯｾｰｼﾞ/ｺﾏﾝﾄﾞの送信を開始します。端末が受信するとｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰに格納されます。

ﾎﾟｰﾘﾝｸﾞは、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がﾎﾟｰﾘﾝｸﾞ処理を始めると”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ﾊﾞｲﾄ以下のﾒｯｾｰｼﾞ*/
                /* 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　　GCC に格納してあるすべての自分宛てのｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑを要求する。

戻値　　0 : ｺﾏﾝﾄﾞ正常受信。ﾎﾟｰﾘﾝｸﾞ開始
　　1 : 衛星を捕捉してない等でﾎﾟｰﾘﾝｸﾞできない

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ｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑを要求するものであり、衛星がこのｺﾏﾝﾄﾞを受信するとｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑの送信を開始します。端末が受信するとｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰに格納されます。

　　ﾎﾟｰﾘﾝｸﾞは、端末がｸﾞﾛｰﾊﾞﾙｸﾞ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がﾎﾟｰﾘﾝｸﾞ処理を始めると”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、ｸﾞﾛｰﾊﾞﾙﾒｯｾｰｼﾞがなかった。 .*/
            }
        }
    }
}
```

chk_failed_polling

要約　　int chk_failed_polling(void);

解説　　端末がﾎﾟｰﾘﾝｸﾞ失敗した時の原因を獲得する

戻値　　ﾀﾞｲｱｸﾞｺｰﾄﾞ

-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 = ﾙｰﾌﾟ検知
4 = 受領者が利用できない
5 = 転送ﾀｲﾑｱｳﾄ
6 = ﾒｯｾｰｼﾞﾀｲﾌﾟがｻﾎﾟｰﾄされていない
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 = 無効ﾊﾟﾗﾒｰﾀ
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ｺｰﾄﾞが無効
103 = 要求されたGCCが衛星ﾀﾞｳﾝﾘﾝｸの中に見つからない。
104 = 不十分なﾒｯｾｰｼﾞﾌﾟﾗｲｵﾘﾃｨ(GCCが輻輳中かもしれない)
105 = 衛星が応答しない(ｱｯﾌﾟ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 = ｲﾝﾊﾞｳﾝﾄﾞﾒｯｾｰｼﾞが既にGCC内に存在する
109 = ｲﾝﾊﾞｳﾝﾄﾞﾒｯｾｰｼﾞﾅﾝﾊﾞｰｴﾗｰ
110 = 不揮発性ﾒﾓﾘにﾒｯｾｰｼﾞをｾｰﾌﾞしている時に、GCC内にｴﾗｰが生じた
111 = GCC内のﾃﾞｰﾀﾍﾞ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 = ｲﾝﾊﾞｳﾝﾄﾞ転送の再送回数が最大値を越えた
114 = ｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑ送信は禁止されている
115 = 衛星が見つからない
116 = ﾎﾟｼﾞｼｮﾝﾚﾎﾟ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 = ｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑのｻｲｽﾞ超過
119 = GCC内にｱｳﾄﾊﾞｳﾝﾄﾞﾒｯｾｰｼﾞ/ｺﾏﾝﾄﾞがない。
120 = GCC内に150ﾊﾞｲﾄ以下のﾒｯｾｰｼﾞ/ｺﾏﾝﾄﾞがない。
121 = 衛星内にｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑがない。
122 = 要求されたﾒｯｾｰｼﾞは削除された。
123 = 軌道要素をｽﾄｱしていない。
124 = ﾚｼﾞｽﾄﾚｰｼｮﾝの要求は受信された。しばらく待て。
125 = ﾚｼﾞｽﾄﾚｰｼｮﾝの要求は認められた。
126 = ﾚｼﾞｽﾄﾚｰｼｮﾝの要求は却下された。
127 = ｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑの最大値(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: ﾒｯｾｰｼﾞ識別番号
unsigned char *status　　: ﾒｯｾｰｼﾞ転送状態
　0 : 送信待ち
　1 : 衛星からのACK受信
　2 : GCCからのACK受信

解説　　ﾒｯｾｰｼﾞ番号指定のﾒｯｾｰｼﾞ転送状態を要求する

戻値　　0 : ACK/NACKが受信されていない

以下はACK/NACKが受信された場合
　1 : あいまいな発信者／受信者の名前
　2 : X.400 MTA の輻輳
　3 : ﾙｰﾌﾟ検知
　4 : 受領者が利用できない
　5 : 転送ﾀｲﾑｱｳﾄ
　6 : ﾒｯｾｰｼﾞﾀｲﾌﾟがｻﾎﾟｰﾄされていない
　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 : 無効ﾊﾟﾗﾒｰﾀ
　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ｺｰﾄﾞが無効
　103 : 要求されたGCCが衛星ﾀﾞｳﾝﾘﾝｸの中に見つからない。
　104 : 不十分なﾒｯｾｰｼﾞﾌﾟﾗｲｵﾘﾃｨ(GCCが輻輳中かもしれない)
　105 : 衛星が応答しない(ｱｯﾌﾟ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 : ｲﾝﾊﾞｳﾝﾄﾞﾒｯｾｰｼﾞが既にGCC内に存在する
　109 : ｲﾝﾊﾞｳﾝﾄﾞﾒｯｾｰｼﾞﾅﾝﾊﾞｰｴﾗｰ
　110 : 不揮発性ﾒﾓﾘにﾒｯｾｰｼﾞをｾｰﾌﾞしている時に、GCC内にｴﾗｰが生じた
　111 : GCC内のﾃﾞｰﾀﾍﾞ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 : ｲﾝﾊﾞｳﾝﾄﾞ転送の再送回数が最大値を越えた
　114 : ｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑ送信は禁止されている
　115 : 衛星が見つからない
　116 : ﾎﾟｼﾞｼｮﾝﾚﾎﾟ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 : ｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑのｻｲｽﾞ超過
　119 : GCC内にｱｳﾄﾊﾞｳﾝﾄﾞﾒｯｾｰｼﾞ/ｺﾏﾝﾄﾞがない。
　120 : GCC内に150ﾊﾞｲﾄ以下のﾒｯｾｰｼﾞ/ｺﾏﾝﾄﾞがない。

121 : 衛星内にｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑがない。
122 : 要求されたﾒｯｾｰｼﾞは削除された。
123 : 軌道要素をｽﾄｱしていない。
124 : ﾚｼﾞｽﾄﾚｰｼｮﾝの要求は受信された。しばらく待て。
125 : ﾚｼﾞｽﾄﾚｰｼｮﾝの要求は認められた。
126 : ﾚｼﾞｽﾄﾚｰｼｮﾝの要求は却下された。
127 : ｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑの最大値(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 ﾒｯｾｰｼﾞ送信・受信（端末－GCC 間）をｱﾎﾞｰﾄする

戻値 NULL ： ｱﾎﾞｰﾄ成功
それ以外: 通信中ではない

補足 ｱﾎﾞｰﾄ要求の発行が成功しても、ﾀｲﾐﾝｸﾞによってはﾒｯｾｰｼﾞがﾎｽﾄに配信されることがあります。

clear_mha_ref_num_msg

要約　　ib_id clear_mha_ref_num_msg(unsigned char mha_ref_num);

unsigned char mha_ref_num: ﾒｯｾｰｼﾞ識別番号

解説　　ﾒｯｾｰｼﾞ識別番号により識別されるｲﾝﾊﾞｳﾝﾄﾞﾒｯｾｰｼﾞを削除する

戻値　　NULL : ﾒｯｾｰｼﾞ識別番号により識別されるｲﾝﾊﾞｳﾝﾄﾞﾒｯｾｰｼﾞが
ｲﾝﾊﾞｳﾝﾄﾞｷｭｰにない。
それ以外 : 削除成功、あるいはそのﾒｯｾｰｼﾞは現在送信中
送信中ﾒｯｾｰｼﾞに関してはｱﾎﾞｰﾄ要求が発行される。

補足　　送信中ﾒｯｾｰｼﾞに関してはｱﾎﾞｰﾄ要求の発行されるが、ﾀｲﾐﾝｸﾞによってはﾒｯｾｰｼﾞ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 送信中ﾒｯｾｰｼﾞ以外のｲﾝﾊﾞｳﾝﾄﾞｷｭｰの中の全てのﾒｯｾｰｼﾞをｸﾘｱ。

戻値 non-NULL : 送信中ﾒｯｾｰｼﾞのﾎﾟｲﾝﾀ（送信中ﾒｯｾｰｼﾞ以外は削除された）
NULL : ｲﾝﾊﾞｳﾝﾄﾞｷｭｰの全てのﾒｯｾｰｼﾞがｸﾘｱされた。

補足 送信中ﾒｯｾｰｼﾞのｸﾘｱは、“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　受信中ﾒｯｾｰｼﾞを除いたｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰの中の全てのﾒｯｾｰｼﾞをｸﾘｱ

戻値　non-NULL : 受信中ﾒｯｾｰｼﾞのﾎﾟｲﾝﾀ（受信中ﾒｯｾｰｼﾞ以外は削除された）
NULL　　: 全てのﾒｯｾｰｼﾞが削除された

補足

req_pos_report_to_ncc

要約

```
int req_pos_report_to_ncc( unsigned char or_ind,
                           unsigned char ncc_id );

unsigned char or_ind : O/R ｲﾝﾃﾞｨｹｰﾀ
unsigned char ncc_id : GCC 番号
```

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

-1: NG　衛星が無い為、要求できない
0: OK
1: NG　ｲﾝﾊﾞｳﾝﾄﾞｷｭｰが満杯

補足

1(NG)の時は、ｲﾝﾊﾞｳﾝﾄﾞｷｭｰのﾒｯｾｰｼﾞが送信されてｷｭｰが空き領域ができてから再度この関数を呼んでください。

req_select_next_downlink

要約 void req_select_next_downlink(void);

解説 ｼｽﾃﾑ内の次のﾀﾞ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ﾀﾞｳﾝﾘﾝｸﾁｬﾝﾈﾙﾘｽﾄの中で、現在受信しているﾁｬﾝﾈﾙの次のﾁｬﾝﾈﾙの受信を試みる。

chk_polling_event

要約　　int chk_polling_event(void)

解説　　GCC からのﾎﾟｰﾘﾝｸﾞ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ﾒｯｾｰｼﾞ
 1:　１ﾒｯｾｰｼﾞあるいはﾚﾎﾟｰﾄ
 2:　１ﾚﾎﾟｰﾄ
 3:　１測位ﾚﾎﾟｰﾄ

unsigned char *ncc_id : GCC番号
unsigned char *or_ind : O/R ｲﾝﾃﾞｨｹｰﾀ

解説　　GCC からのﾎﾟｰﾘﾝｸﾞ内容を獲得する

戻値　　なし

補足　　この関数は、KXS68=0 の時無効です。
poll_obj(3)は KXS24=0 のときのみ得ることができます。

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 : ﾎﾟｰﾙﾄﾞ
ack_level : ACK ﾚﾍﾞﾙ
priority : ﾌﾟﾗｲｵﾘﾃｨ
msg_body_type : ﾒｯｾｰｼﾞ ﾎﾞﾃﾞｨ ﾀｲﾌﾟ
msg_body_sub_type: ﾒｯｾｰｼﾞ ﾎﾞﾃﾞｨ ｻﾌﾞ ﾀｲﾌﾟ
mha_ref_num : ﾒｯｾｰｼﾞ識別番号
rcpnt[7] : O/R ｲﾝﾃﾞｨｹｰﾀ、O/R ｱﾄﾞﾚｽ
subject : ｻﾌﾞｼﾞｪｸﾄ
message : ﾒｯｾｰｼﾞ本体
message_len : ﾒｯｾｰｼﾞ長

解説　　ｲﾝﾊﾞｳﾝﾄﾞﾒｯｾｰｼﾞをｲﾝﾊﾞｳﾝﾄﾞｷｭｰに設定する

戻値　　0 : ｲﾝﾊﾞｳﾝﾄﾞｷｭｰに設定完了
1 : ｲﾝﾊﾞｳﾝﾄﾞｷｭｰに設定失敗。ﾒｯｾｰｼﾞ長が長すぎてｷｭｰに入らない。
2 : ｲﾝﾊﾞｳﾝﾄﾞｷｭｰに設定失敗。ﾒﾓﾘｰが獲得できない。

補足　　ﾒｯｾｰｼﾞ ﾎﾞﾃﾞｨ ｻﾌﾞ ﾀｲﾌﾟは、ﾒｯｾｰｼﾞ ﾎﾞﾃﾞｨ ﾀｲﾌﾟが 0、15 の時だけ有効。
ｻﾌﾞｼﾞｪｸﾄがない時は、NULL を設定すること。

<例>

```
#include <stdio.h>
#include <string.h>
#include “kmelib.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;

    /* ﾒｯｾｰｼﾞ作成 */
    strcpy(message, “ HELLO ”);

    /* ﾒｯｾｰｼﾞ識別番号設定 */
    mha_ref_num = 0;

    /* 送信ﾒｯｾｰｼﾞをｲﾝﾊﾞｳﾝﾄﾞｷｭｰに設定 */
    subject[0] = '\0';
    if (req_send_ib_message(get_kxs01(),    /* GCC 番号 */
                         0,          /*すぐに送信 */
                        get_kxs06(),         /* ACKﾚﾍﾞﾙ */
                        get_kxs03(),         /* ﾌﾟﾗｲｵﾘﾃｨ */
                        14,          /* ﾒｯｾｰｼﾞﾎﾞﾃﾞｨﾀｲﾌﾟ：ﾊﾞｲﾅﾘｰ */
                         0,          /* ﾒｯｾｰｼﾞﾎﾞﾃﾞｨｻﾌﾞﾀｲﾌﾟ (ﾀﾞﾐｰ) */
              mha_ref_num,           /* ﾒｯｾｰｼﾞ識別番号 */
                   recipient,        /* 受信者 */
                    subject,         /* ｻﾌﾞｼﾞｪｸﾄなし */
                   message,          /* ﾒｯｾｰｼﾞ本体 */
             strlen(message) )       /* ﾒｯｾｰｼﾞ長 */
        == MESSAGE_QUEUED) {/* 設定成功？ */
        ;   /* 設定成功 */
    } else {
        ;   /* 設定失敗 */
    }
    exit_user_apl();/* ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝ終了 */
}
```

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 : ﾎﾟｰﾙﾄﾞ
serv_type : ｻｰﾋﾞｽﾀｲﾌﾟ
mha_ref_num : 識別番号
user_data[6] : ﾚﾎﾟｰﾄ本体

解説

ｲﾝﾊﾞｳﾝﾄﾞﾚﾎﾟｰﾄをｲﾝﾊﾞｳﾝﾄﾞｷｭｰに設定する

戻値

0 : ｲﾝﾊﾞｳﾝﾄﾞｷｭｰに設定完了
2 : ｲﾝﾊﾞｳﾝﾄﾞｷｭｰに設定失敗。ﾒﾓﾘｰが獲得できない。

補足

user_data（ﾚﾎﾟｰﾄ本体）のﾊﾞｲﾄ数は6ﾊﾞｲﾄ。

<例>

```
#include “kme_lib.h”

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

    /* ﾚﾎﾟｰﾄ本体設定 */
    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;

    /* ｲﾝﾊﾞｳﾝﾄﾞﾚﾎﾟｰﾄをｲﾝﾊﾞｳﾝﾄﾞｷｭｰに設定する */
    if (req_send_ib_report( get_kxs01(), /* GCC番号 */
                            0,        /* すぐに送信 */
                      get_kxs08(),          /* ｻｰﾋﾞｽﾀｲﾌﾟ  */
                      get_kxs04(),          /* O/Rｲﾝﾃﾞｨｹｰﾀ  */
                    mha_ref_num,   /* 識別番号 */
                    user_data )    /* ﾒｯｾｰｼﾞ本体 */
        == MESSAGE_QUEUED) { /* 設定成功？ */
        ;   /* 設定成功 */
    } else {
        ;   /* 設定失敗 */
    }
    exit_user_apl();        /* ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝ終了 */
}
```

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 ｲﾝﾃﾞｨｹｰﾀ
user_data : ｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑ本体

解説

ｲﾝﾊﾞｳﾝﾄﾞｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑをｲﾝﾊﾞｳﾝﾄﾞｷｭｰに設定する

戻値

0 : ｲﾝﾊﾞｳﾝﾄﾞｷｭｰに設定完了
1 : ｲﾝﾊﾞｳﾝﾄﾞｷｭｰに設定失敗。長すぎてｷｭｰに入らない。
2 : ｲﾝﾊﾞｳﾝﾄﾞｷｭｰに設定失敗。ﾒﾓﾘｰが獲得できない。

補足

user_data（ｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑ本体）のﾊﾞｲﾄ数は229ﾊﾞｲﾄ以下。

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 : ﾎﾟｰﾙﾄﾞ
serv_type : ｻｰﾋﾞｽﾀｲﾌﾟ
mha_ref_num : 識別番号
*lat: 緯度 - 0: 北極, 0xffffff: 南極
*lon: 経度 - 0: ｸﾞﾘﾆｯｼﾞ子午線,東に向かって増加

解説

ｲﾝﾊﾞｳﾝﾄﾞ測位ﾚﾎﾟｰﾄをｲﾝﾊﾞｳﾝﾄﾞｷｭｰに設定する

戻値

0 : ｲﾝﾊﾞｳﾝﾄﾞｷｭｰに設定完了
2 : ｲﾝﾊﾞｳﾝﾄﾞｷｭｰに設定失敗。ﾒﾓﾘｰが獲得できない。

補足

lat/lonは3ﾊﾞｲﾄ。

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 : ﾒｯｾｰｼﾞ ﾎﾞﾃﾞｨ ﾀｲﾌﾟ
msg_body_sub_type: ﾒｯｾｰｼﾞ ﾎﾞﾃﾞｨ ｻﾌﾞ ﾀｲﾌﾟ
rcpnt[7] : O/R ｲﾝﾃﾞｨｹｰﾀ、O/R ｱﾄﾞﾚｽ
subject : ｻﾌﾞｼﾞｪｸﾄ
message : ﾒｯｾｰｼﾞ本体
message_len : ﾒｯｾｰｼﾞ長

解説

ｱｳﾄﾊﾞｳﾝﾄﾞﾒｯｾｰｼﾞをｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰに設定する

戻値

0 : ｲﾝﾊﾞｳﾝﾄﾞｷｭｰに設定完了
1 : ｲﾝﾊﾞｳﾝﾄﾞｷｭｰに設定失敗。長すぎてｷｭｰに入らない。
2 : ｲﾝﾊﾞｳﾝﾄﾞｷｭｰに設定失敗。ﾒﾓﾘｰが獲得できない。

補足

ﾒｯｾｰｼﾞ ﾎﾞﾃﾞｨ ｻﾌﾞ ﾀｲﾌﾟは、ﾒｯｾｰｼﾞ ﾎﾞﾃﾞｨ ﾀｲﾌﾟが 0、15 の時だけ有効。
ｻﾌﾞｼﾞｪｸﾄがない時は、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;

        /* ﾒｯｾｰｼﾞ本体を設定 */
        message[0] = ‘H’;
        message[1] = ‘E’;
        message[2] = ‘L’;
        message[3] = ‘L’;
        message[4] = ‘O’;
        message[5] = '\0';

        /* ﾒｯｾｰｼﾞをｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰに設定する */
        subject[0] = '\0';
        if (req_send_ob_message( get_kxs01(),    /* GCC番号 */
                                 0,           /* ﾒｯｾｰｼﾞﾎﾞﾃﾞｨﾀｲﾌﾟ：TEXT */
                                 5,           /* ﾒｯｾｰｼﾞﾎﾞﾃﾞｨｻﾌﾞﾀｲﾌﾟ */
                           recipient,                 /* 受信者 */
                             subject,          /* ｻﾌﾞｼﾞｪｸﾄ */
                             message,          /* ﾒｯｾｰｼﾞ本体 */
                             strlen(message) )   /* ﾒｯｾｰｼﾞ長 */
                        == MESSAGE_QUEUED) {      /* 設定成功？ */
            ;   /* 設定成功 */
        } else {
            ;   /* 設定失敗 */
        }
        exit_user_apl();        /* ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝ終了 */
    }
```

get_unget_ob_id_for_user

要約　　ob_id get_unget_ob_id_for_user(void);

解説　　ｱｳﾄﾊﾞｳﾝﾄﾞ ID を獲得する

戻値　　non-NULL: ｱｳﾄﾊﾞｳﾝﾄﾞ ID
NULL.　: ｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰにﾃﾞｰﾀがない

補足　　ｱｳﾄﾊﾞｳﾝﾄﾞ ID は、ｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰにあるﾃﾞｰﾀのﾎﾟｲﾝﾀ。

本関数をコールするとｱｳﾄﾊﾞｳﾝﾄﾞ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;

  /* ｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰにﾃﾞ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 :  /* ｱｳﾄﾊﾞｳﾝﾄﾞﾒｯｾｰｼﾞ */
          /*  1. GCC番号                                  */
          /*  2. ｻﾌﾞｼﾞｪｸﾄ                                  */
          /*  3. ﾒｯｾｰｼﾞﾎﾞﾃﾞｨﾀｲﾌﾟ                           */
          /*  4. 受信者 + 送信者                          */
          /*  5. ﾒｯｾｰｼﾞ長                                 */
          /*  6. ﾒｯｾｰｼﾞ                                   */
          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 :  /* ｱｳﾄﾊﾞｳﾝﾄﾞﾕｰｻﾞｰｺﾏﾝﾄﾞ */
    /*  1. GCC番号      */
    /*  2. ﾃﾞｰﾀ本体 (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 : /* ｱｳﾄﾊﾞｳﾝﾄﾞｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑ */
            /*  1. GCC番号                                  */
            /*  2. ﾒｯｾｰｼﾞ ﾎﾞﾃﾞｨ ﾀｲﾌﾟ                          */
            /*  3. 識別番号                                 */
            /*  4. ﾒｯｾｰｼﾞ長                                 */
            /*  5. ｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑ本体                          */
            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 : ｱｳﾄﾊﾞｳﾝﾄﾞ ID

解説 ｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰからｱｳﾄﾊﾞｳﾝﾄﾞ ID 指定のﾒｯｾｰｼﾞを削除する

戻値 なし

補足 ﾒｯｾｰｼﾞ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。
削除したﾒｯｾｰｼﾞに対して再度この関数をｺｰﾙするとﾌﾟﾛｸﾞﾗﾑが
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 : ｱｳﾄﾊﾞｳﾝﾄﾞ ID

解説 ｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰにあるﾃﾞｰﾀの種類を獲得

戻値
0) ｱｳﾄﾊﾞｳﾝﾄﾞﾒｯｾｰｼﾞ
1) ｱｳﾄﾊﾞｳﾝﾄﾞｺﾏﾝﾄﾞ
2) ｱｳﾄﾊﾞｳﾝﾄﾞｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑ

補足 ｱｳﾄﾊﾞｳﾝﾄﾞ 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 : ｱｳﾄﾊﾞｳﾝﾄﾞ ID

解説 ｱｳﾄﾊﾞｳﾝﾄﾞ ID 指定のﾃﾞｰﾀの GCC 番号を獲得する

戻値 GCC 番号 (0 - 255)

補足 ｱｳﾄﾊﾞｳﾝﾄﾞ 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 : ｱｳﾄﾊﾞｳﾝﾄﾞ ID

解説 ｻﾌﾞｼﾞｪｸﾄの有無を獲得する。

戻値 0 : ｻﾌﾞｼﾞｪｸﾄ無し
1 : ｻﾌﾞｼﾞｪｸﾄ有り

補足 ｱｳﾄﾊﾞｳﾝﾄﾞ 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 : ｱｳﾄﾊﾞｳﾝﾄﾞ ID

解説 ｱｳﾄﾊﾞｳﾝﾄﾞ ID 指定のﾒｯｾｰｼﾞのﾒｯｾｰｼﾞﾎﾞﾃﾞｨﾀｲﾌﾟを獲得する

戻値 ﾒｯｾｰｼﾞﾎﾞﾃﾞｨﾀｲﾌﾟ

補足 ｱｳﾄﾊﾞｳﾝﾄﾞ ID は、関数“get_unget_ob_id_for_user”により獲得する

get_ob_msg_or_quan

要約 unsingned char get_ob_msg_or_quan(ob_id get_ob_id);

ob_id get_ob_id : ｱｳﾄﾊﾞｳﾝﾄﾞ ID

解説 ｱｳﾄﾊﾞｳﾝﾄﾞ ID 指定のﾒｯｾｰｼﾞの受信者の数を獲得する

戻値 受信者の数

補足 ｱｳﾄﾊﾞｳﾝﾄﾞ ID は、関数“ get_unget_ob_id_for_user ”により獲得する.

get_ob_msg_msg_len

要約 unsigned int get_ob_msg_msg_len(ob_id get_ob_id);

ob_id get_ob_id : ｱｳﾄﾊﾞｳﾝﾄﾞ ID

解説 ｱｳﾄﾊﾞｳﾝﾄﾞ ID 指定のﾒｯｾｰｼﾞのﾒｯｾｰｼﾞ長を獲得する

戻値 ﾒｯｾｰｼﾞ長

補足 ｱｳﾄﾊﾞｳﾝﾄﾞ 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 : ｱｳﾄﾊﾞｳﾝﾄﾞ ID
int *msg_len　　: ﾒｯｾｰｼﾞ長.

解説　　ｱｳﾄﾊﾞｳﾝﾄﾞ ID 指定のﾒｯｾｰｼﾞのﾒｯｾｰｼﾞ本体を指すｲﾝﾃﾞｯｸｽを獲得する

戻値　　ﾒｯｾｰｼﾞ本体を指すｲﾝﾃﾞｯｸｽ

補足　　ｱｳﾄﾊﾞｳﾄﾞ ID は、関数“ get_unget_ob_id_for_user ”により獲得する。ﾒｯｾｰｼﾞﾎﾟﾃﾞｨﾀｲﾌﾟが 0 または 15 の時はﾒｯｾｰｼﾞ本体の最初のﾃﾞｰﾀはｻﾌﾞﾎﾞﾃﾞｨﾀｲﾌﾟとなる。

get_ob_message

要約　　unsigned char get_ob_message (ob_id get_ob_id,
　　unsigned int msg_index);

　　ob_id get_ob_id　　: ｱｳﾄﾊﾞｳﾝﾄﾞ ID
　　unsigned int msg_index : ｲﾝﾃﾞｯｸｽ

解説　　ｱｳﾄﾊﾞｳﾝﾄﾞ ID 指定のﾒｯｾｰｼﾞのﾒｯｾｰｼﾞ本体を獲得する

戻値　　ﾒｯｾｰｼﾞ本体の１ﾃﾞｰﾀ (0x00 - 0xff)

補足　　ｱｳﾄﾊﾞｳﾝﾄﾞ ID は、関数“ get_unget_ob_id_for_user ”により獲得する

get_ob_user_command

要約　unsigned char get_ob_user_command(ob_id get_ob_id
unsigned int data_index);

ob_id get_ob_id : ｱｳﾄﾊﾞｳﾝﾄﾞ ID
data_index : ｲﾝﾃﾞｯｸｽ

解説　ｱｳﾄﾊﾞｳﾝﾄﾞ ID 指定のｺﾏﾝﾄﾞの本体を獲得する

戻値　ｺﾏﾝﾄﾞ本体の 1 ﾃﾞｰﾀ

補足　ｱｳﾄﾊﾞｳﾝﾄﾞ 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 : ｱｳﾄﾊﾞｳﾝﾄﾞ ID

解説 ｱｳﾄﾊﾞｳﾝﾄﾞ ID 指定のｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑの識別番号を獲得する

戻値 GCC より割り当てられる識別番号(0-255)

補足 ｱｳﾄﾊﾞｳﾝﾄﾞ 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 : ｱｳﾄﾊﾞｳﾝﾄﾞ ID

解説 ｱｳﾄﾊﾞｳﾝﾄﾞ ID 指定のｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑの O/R ｲﾝﾃﾞｨｹｰﾀを獲得する

戻値 O/R ｲﾝﾃﾞｨｹｰﾀ

補足 ｱｳﾄﾊﾞｳﾝﾄﾞ 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 : ｱｳﾄﾊﾞｳﾝﾄﾞ ID

解説　　ｱｳﾄﾊﾞｳﾝﾄﾞ ID 指定のｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑのﾃﾞｰﾀ長を獲得する

戻値　　ﾃﾞｰﾀ長(0 - 182)

補足　　ｱｳﾄﾊﾞｳﾝﾄﾞ ID は、関数“get_unget_ob_id_for_user”により獲得する

get_ob_globalgram

要約　　unsigned char get_ob_globalgram(ob_id get_ob_id,
　　unsigned int data_index);

　　ob_id get_ob_id　　: ｱｳﾄﾊﾞｳﾝﾄﾞ ID
　　unsigned int data_index: ｲﾝﾃﾞｯｸｽ

解説　　ｱｳﾄﾊﾞｳﾝﾄﾞ ID 指定のｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑの本体を獲得する

戻値　　ｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑ本体の 1 ﾊﾞｲﾄ (0x00 - 0xff)

補足　　ｱｳﾄﾊﾞｳﾝﾄﾞ 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　　一定時間ﾊﾟﾜｰﾀﾞｳﾝさせる

戻値　　なし

補足　　ｽﾘｰﾌﾟﾓｰﾄﾞがｵﾌ(KXS37=0)の時は、無効です

go_sleep_next_path

要約 void go_sleep_next_path(unsigned long int sleep_time);

unsigned long int sleep_time : 最低ﾊﾟﾜｰﾀﾞｳﾝ時間
(0 － 4294967295[分])

解説 少なくとも最低ﾊﾟﾜｰﾀﾞｳﾝ時間で示される秒数分だけﾊﾟﾜｰﾀﾞｳﾝする。それから次の衛星のﾊﾟｽで起床する。

戻値 なし

補足 ｽﾘｰﾌﾟﾓｰﾄﾞがｵﾌ(KXS37=0)の時は、無効です

go_wait

要約 void go_wait(unsigned long time, char time_unit);

unsigned long time: サスペンド時間
char time_unit : 単位
0 : ﾐﾘｾｶﾝﾄﾞ
1 : 秒
2 : 分

解説 ﾕｰｻﾞｰｱﾌﾟﾘの実行を指定時間ｻｽﾍﾟﾝﾄﾞする。

戻値 なし

補足 これは、ﾕｰｻﾞｰｱﾌﾟﾘﾀｽｸの実行権を他ﾀｽｸに渡す関数であり、ﾊﾟﾜｰﾀﾞ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 : 電源ｵﾌ
1 : 電源ｵﾝ

補足

chk_rx_buff

要約 unsigned int chk_rx_buff(void);

解説 RS232C ｲﾝﾀｰﾌｪｰｽから受信されたﾃﾞｰﾀ数を獲得する

戻値 受信ﾃﾞｰﾀ数

補足 受信ﾊﾞｯﾌｧｻｲｽﾞは 256byte です。

get_rx_data

要約 int chk_rx_buff(void);

解説 RS232C ｲﾝﾀｰﾌｪｰｽから受信されたﾃﾞｰﾀを１ﾊﾞｲﾄ取り出す。

戻値 受信ﾃﾞｰﾀ: 0x00 - 0xff
ｴﾗｰ : -1

補足 RS232C の受信ﾊﾞｯﾌｧｻｲｽﾞは 256 ﾊﾞｲﾄです。

chk_tx_ready

要約　　int chk_tx_ready(void);

解説　　RS232C の送信ﾊﾞｯﾌｧが使用可能かﾁｪｯｸする

戻値　　0 : 送信不可。ﾊﾞｯﾌｧが満杯。
1 : 送信可能。ﾊﾞｯﾌｧは使用可能。

補足　　送信ﾊﾞｯﾌｧが満杯の時は送信ﾃﾞｰﾀをｾｯﾄできません。

set_tx_data

要約　unsigned int set_tx_data(unsigned char tx_data);

unsigned char tx_data : 送信ﾃﾞｰﾀ　(0x00　- 0xff)

解説　RS232C への送信ﾃﾞｰﾀを設定する

戻値　ｲﾝﾃﾞｯｸｽ。送信ﾊﾞｯﾌｧ先頭からの番号

補足　RS232C の送信ﾊﾞｯﾌｧｻｲｽﾞは 256 ﾊﾞｲﾄです。
この関数には送信ﾊﾞｯﾌｧが使用可能かのﾁｪｯｸは含まれていませんので、
chk_tx_ready()関数と共に使用してください。

cts_act

要約 void cts_act(void)

解説 CTS 信号をｱｸﾃｨﾌﾞにする

戻値 なし

補足 KXS68=1 のときのみ有効です。

cts_neg

要約 void cts_neg(void)

解説 CTS 信号をﾉﾝｱｸﾃｨﾌﾞにする.

戻値 なし

補足 KXS68=1 のときのみ有効です。

chk_cd

要約 int chk_cd(void)

解説 ﾒｲﾝﾎﾟｰﾄの 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　　ﾒｲﾝﾎﾟｰﾄの 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

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毎にｶｳﾝﾄｱｯﾌﾟされる。

戻値　　ｶｳﾝﾀ値　(0 - 0xffff)

補足

set_timer

要約　　int set_timer(char no,
　　unsigned long time,
　　char time_unit);

　　char no　　: ﾀｲﾏｰ番号 (1 - 10)
　　unsigned long time: ﾀｲﾏｰ値
　　char time_unit　: 単位
　　0:ﾐﾘｾｶﾝﾄﾞ
　　1:秒
　　2:分

解説　　ﾀｲﾏｰを設定する

戻値　　0 : 設定失敗
　　1 : 設定成功

補足　　ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝ用に 10 個のﾀｲﾏｰが用意されている。
　　時間単位毎の最大値は以下である。
　　ﾐﾘｾｶﾝﾄﾞ : 4294967295 ﾐﾘｾｶﾝﾄﾞ
　　秒　　: 274877906 秒
　　分　　: 4581298 分

<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) {     /* ﾀｲﾏｰ1設定 */
          exit_user_apl();                     /* ﾕｰｻﾞｰｱﾌﾟﾘ終了 */
      }

      while (chk_pos_calc_result() == 0) {     /* 測位終了？ */
          if (check_timer(1) == 0) {           /* ﾀｲﾑｱｯﾌﾟ？ */
              /* ﾀｲﾑｱｯﾌﾟ */
              /* 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秒ｽﾘｰﾌﾟ */
      }
  }
  exit_user_apl();    /* ﾕｰｻﾞｰｱﾌﾟﾘ終了*/
}
```

check_timer

要約　　int check_timer(char no);

char no:　ﾀｲﾏｰ番号　(1 - 10)

解説　　指定ﾀｲﾏｰがﾀｲﾑｱｯﾌﾟしたかﾁｪｯｸする

戻値　　0 : ﾀｲﾑｱｯﾌﾟした
1 : ﾀｲﾑｱｯﾌﾟしてない

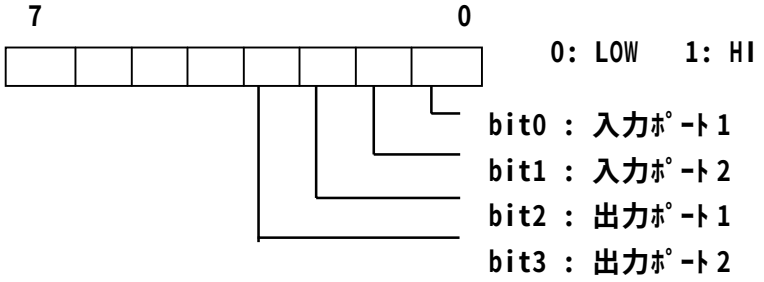
補足

get_digital_port

要約　　unsigned char get_digital_port(void);

解説　　入力ﾎﾟｰﾄ、出力ﾎﾟｰﾄの状態を獲得する

戻値　　ﾎﾟｰﾄ状態

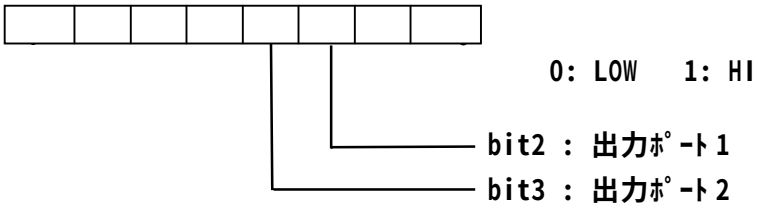


補足　　電源 ON 時、出力ﾎﾟｰﾄは set_kxs83 の設定に従います。

set_digital_port

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

unsigned char sel_port : 出力ﾎﾟｰﾄ識別番号
unsigned char data　　: 出力ﾃﾞｰﾀ (0 : LOW　or 1: HI)



解説　　出力ﾎﾟｰﾄを設定する

戻値　　なし

補足

<例>

```
#include “kme_lib.h”

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

  /* 出力ﾎﾟｰﾄ1をHI に設定 */
  Set_digital_port(0x04, 0x04);

  /* 出力ﾎﾟｰﾄ2をLOW に設定 */
  Set_digital_port(0x08, 0x00);

  /* 出力ﾎﾟｰﾄ2をHI に設定 */
  Set_digital_port(0x08, 0x08);

  exit_user_apl();          /* ﾕｰｻﾞｰｱﾌﾟﾘを終了する */
}
```

get_adcon_data

要約　　unsigned char get_adcon_data(int ch);

int ch : ｱﾅﾛｸﾞﾎﾟｰﾄ番号
  0 : RSSI
  1 : ｱﾅﾛｸﾞﾎﾟｰﾄ1
  2 : ｱﾅﾛｸﾞﾎﾟｰﾄ2
  3- 7 : 予備

解説　　ｱﾅﾛｸﾞﾎﾟｰﾄの状態を獲得する

戻値　　値 (0 -255)

補足　　0 - 3.3V を 0 - 255 の範囲で変換します。

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) {    /* ﾀｲﾏｰ設定 */
            exit_user_apl();    /* ﾕｰｻﾞｰｱﾌﾟﾘ終了 */
        }

        while (chk_pos_calc_result() == 0) { /* 測位終了？ */
            /* 測位中 */
            if (check_timer(1) == 0) {
                /* ﾀｲﾑｵｰﾊﾞｰ */
                /* 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();                    /* ﾕｰｻﾞｰｱﾌﾟﾘ終了 */
}
```

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ＫＸＳ20 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 "ｸｵﾘﾃｨｲﾝﾃﾞｨｹｰﾀ" を獲得する。

戻値 測位精度
0 ： GPSによる測位結果
3 ：ﾄﾞｯﾌﾟﾗｰ測位による測位結果
15 ：ﾄﾞｯﾌﾟﾗｰ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　　ﾄﾞｯﾌﾟﾗｰ測位に使用したｵｰﾌﾞｺﾑ衛星の数を獲得する。

戻値　　ｵｰﾌﾞｺﾑ衛星数

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

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（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足　　この関数は、set_kxs87()と同じ処理を行います。
PDOP を小さくするほど精度は向上しますが、測位時間が長くなります。

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（１～１０）

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 ﾊﾞｲﾄの領域が必要。

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

2. UTC 年月日XXXX XX XX

3. 測位ｽﾃｰﾀｽ (1ﾊﾞｲﾄ)

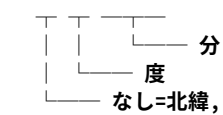
< 警告（測位有効）>
bit0: 受信感度悪い
Bit1: HDOP悪い
Bit2: 測位精度悪い
Bit3: 位置保留

< 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



6. 経度　　u XX XX.XXX
　　┬ ┬ ┬┬┬
　　│ │ └┴┴ 分
　　│ └┴┴ 度
　　└┴┴ なし=東経, '-'=西経
7. 標高　　-9999 ～ 9999 [m]
8. ｼﾞｵｲﾄﾞ高　-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. 測位ﾓｰﾄﾞ　ｵｰﾄﾓｰﾄﾞ(A).
13. 測位次元　2= 2次元測位.　3=3次元測位
14. ﾁｬﾝﾈﾙ数　8
15. 衛星情報(2ﾊﾞｲﾄ).
　　bit0-7 : 衛星番号
　　bit8-11: 受信ﾚﾍﾞﾙ
　　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) ｼﾞｵｲﾄﾞ高： 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、ｾﾝﾃﾝｽ中の '$' と '*' とで挟まれたｷｬﾗｸﾀﾃﾞｰﾀを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

x05 ｾﾝﾃﾝｽを獲得する為には、170 ﾊﾞｲﾄの領域が必要。

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、受信可能な GPS 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ﾃﾞｰﾀを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 動作状態

- 0 = ｱｲﾄﾞﾙ
- 1 = ｲﾝﾊﾞｳﾝﾄﾞﾒｯｾｰｼﾞ送信中
- 2 = ｲﾝﾊﾞｳﾝﾄﾞﾚﾎﾟｰﾄ送信中
- 3 = ｲﾝﾊﾞｳﾝﾄﾞｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑ送信中
- 4 = ｱｳﾄﾊﾞｳﾝﾄﾞﾒｯｾｰｼﾞ受信中
- 5 = ｱｳﾄﾊﾞｳﾝﾄﾞｺﾏﾝﾄﾞ受信中
- 6 = ｱｳﾄﾊﾞｳﾝﾄﾞｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑ受信中
- 7 = ｾﾙﾌﾃｽﾄ中
- 8 = ﾛｰｶﾙﾙｰﾌﾟﾊﾞｯｸ中
- 9 = ﾘﾓｰﾄﾙｰﾌﾟﾊﾞｯｸﾃｽﾄ中

補足 GCC に対してﾎﾟｰﾘﾝｸﾞ(ﾒｯｾｰｼﾞ/ｺﾏﾝﾄﾞの送信要求)をしている時は、
4 になります。
衛星に対してﾎﾟｰﾘﾝｸﾞ(ｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑの送信要求)をしている時は、
6 になります。

get_active_mha_ref_num

要約　unsigned char get_active_mha_ref_num(void);

解説　現在送信中の mhal ﾘﾌｧﾚﾝｽ番号を獲得する

戻値　0x00 - 0xfe　：現在送信中の mha ﾘﾌｧﾚﾝｽ番号
0xff　：現在送信していない

補足　mha ﾘﾌｧﾚﾝｽ番号はﾒｯｾｰｼﾞ作成時に指定しますが、本関数を使用する場合
mha ﾘﾌｧﾚﾝｽ(0xff)は戻り値の判定のために使用しないでください。

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 現在ﾘﾝｸしている GCC の数を獲得する

戻値 現在ﾘﾝｸしている 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　: ｲﾝﾃﾞｯｸｽ。0 から数える
ncc_id : GCC ID
min_pri : 受信可能なﾒｯｾｰｼﾞのﾌﾟﾗｲｵﾘﾃｨの最小値

解説

GCC ID とその受信可能なﾒｯｾｰｼﾞのﾌﾟﾗｲｵﾘﾃｨ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、ｲﾝﾃﾞｯｸｽを+1 していってこの関数を呼ぶ。

get_num_of_ob_msgs

要約 int get_num_of_ob_msgs(void);

解説 ｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰに格納しているﾒｯｾｰｼﾞの数を獲得する

戻値 ﾒｯｾｰｼﾞの数

補足

get_num_of_ib_msgs

要約 int get_num_of_ib_msgs(void);

解説 ｲﾝﾊﾞｳﾝﾄﾞｷｭｰに格納しているﾒｯｾｰｼﾞの数を獲得する

戻値 ﾒｯｾｰｼﾞの数

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 ｵｰﾋﾞｺﾑ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 衛星ﾀﾞｳﾝﾘﾝｸの受信ｴﾗｰ数を獲得する

戻値 ｴﾗｰ数

補足 エラー数は KXST 入力または本関数 CALL 時に 0 にクリアされる。

exit_user_apl

要約 void exit_user_apl(void);

解説 ﾕｰｻﾞｰｱﾌﾟﾘを終了する

戻値 なし

補足

break_point

要約 int break_point(unsigned char brk_num)

unsigned char brk_num: ﾌﾞﾚｰｸﾎﾟｲﾝﾄ番号 (1 - 255).

解説 この関数はﾕｰｻﾞｰｱﾌﾟﾘﾌﾟﾛｸﾞﾗﾑをﾃﾞﾊﾞｯｸﾞする時に使用される。
ﾕｰｻﾞｰは、あらかじめﾕｰｻﾞｰｱﾌﾟﾘﾌﾟﾛｸﾞﾗﾑの中に“ break_point ”関数を挿入しておく必要がある。

戻値
-1: 設定ｴﾗｰ　ﾌﾞﾚｲｸﾎﾟｲﾝﾄ番号が 0 になっている
0 : 設定ｴﾗｰ　KXS70=0 となっている
1 : 設定完了

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 ” : ﾌﾞﾚｰｸﾎﾟｲﾝﾄ 23 を有効にする
“ break 23 1 ” : ﾌﾞﾚｰｸﾎﾟｲﾝﾄ 23 を無効にする

ﾌﾞﾚｰｸした後、 ”ctrl+g ”ｺﾏﾝﾄﾞにより再開する。

これらの機能は、ﾃﾞﾊﾞｯｸﾞ機能が ON（KXS70=1-9）の時に有効となる。

chk_break_point

要約 int chk_break_point(unsigned char num)

unsigned char num : ﾌﾞﾚｰｸﾎﾟｲﾝﾄ番号 (1 - 255).

解説 ﾌﾞﾚｰｸﾎﾟｲﾝﾄ設定状態を獲得する

戻値 0 : ﾌﾞﾚｰｸﾎﾟｲﾝﾄが設定されている
1 : ﾌﾞﾚｰｸﾎﾟｲﾝﾄが設定されてない

補足

get_application_dbg

要約 unsigned char get_application_dbg(void)

解説 ﾕｰｻﾞｰｱﾌﾟﾘﾃﾞﾊﾞｯｸﾞ機能（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 ｲﾝﾊﾞｳﾝﾄﾞｷｭｰにﾒｯｾｰｼﾞ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ｼﾞ送信を再開する

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;            /* 00-99  */
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 : UTC 時刻

```
struct time_buff {
unsigned char  year;            /* 00-99  */
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: ﾛｰｶﾙ時間

```
struct time_buff {
unsigned char  year;            /* 00-99  */
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  */
};
```

int time_zone : ﾀｲﾑｿﾞｰﾝ (分) (-690 - 720)

解説 ﾛｰｶﾙ時間を獲得する

戻値 なし

補足

get_convert_time

要約　　void get_convertl_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_zome　　:　ﾀｲﾑｿﾞｰﾝ (分)　(-690 - 720)

解説　　ﾊﾞｯﾌｧ内の時刻ﾃﾞｰﾀを、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　　最低時間後のｵｰﾌﾞｺﾑ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します。

<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();    /* ﾕｰｻﾞｰｱﾌﾟﾘ終了 */
}
```

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

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です。

get_sys_info

要約　　char get_sys_info(int obj)

int obj : 番号

0	ROM ﾊﾞｰｼﾞｮﾝ
3	ｲﾝﾊﾞｳﾝﾄﾞ/ｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰのﾁｪｯｸｻﾑのﾁｪｯｸ
4	ASIC ﾘｰﾄﾞ/ﾗｲﾄﾁｪｯｸ
5	IC ﾘｰﾄﾞ/ﾗｲﾄﾁｪｯｸ
6	ｼﾝｾｻｲｻﾞｰ（送信部）のﾁｪｯｸ
7	ｼﾝｾｻｲｻﾞｰ（受信部）のﾁｪｯｸ
8	衛星通信のﾃﾞｼﾞﾀﾙ部のﾙｰﾌﾟﾊﾞｯｸﾃｽﾄ
9	GPS ROM ﾊﾞｰｼﾞｮﾝ
10	端末 ID のﾁｪｯｸｻﾑのﾁｪｯｸ
11	ｺﾝﾌｨｸﾞﾚｰｼｮﾝﾊﾟ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ﾊﾞｰｼﾞｮﾝ、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 文字列ﾊﾞｯﾌｧのｱﾄﾞﾚｽ（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　　格納ﾊﾞｯﾌｧは、少なくとも 12 ﾊﾞｲﾄ必要。

<ex.>

```
void main(void)
{
    char serial_no[12];

    get_kme_serial_no(serial_no);

    rs_tx_str(serial_no);
}
```

実行結果

0ACDE801001

set_no_wait_for_power_save

要約　int set_no_wait_for_power_save(int flag)

int flag
  0 : ﾊﾟﾜｰｾｰﾌﾞ時にｳｪｲﾄを挿入する
  1 : ﾊﾟﾜｰｾｰﾌﾞ時にｳｪｲﾄを挿入しない

解説　OSは、ﾊﾟﾜｰｾｰﾌﾞ時の消費電力を抑える為、CPUにｳｪｲﾄを入れている.。このｳｪｲﾄ挿入の為、ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝの稼働率が低下する。
この関数は､ﾊﾟﾜｰｾｰﾌﾞ時に自動的に挿入されるｳｴｲﾄを挿入しないようにする。

戻値　０：設定成功　１：設定失敗

補足　この設定値は、ﾊﾞｯｸｱｯﾌﾟされないので、毎回ﾊﾟﾜｰｵﾝ時に設定してください。

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 ***/
#pragma _section B=USER
int os_work;;
#pragma _section B

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 prisicion)

　　double value:変換する数値
　　unsigned char *string:文字列格納ﾊﾞｯﾌｧ
　　int width:全変換桁数
　　int prisicion: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ﾎﾟｲﾝﾀ

補足　　<note>
　　標準関数の "spritf()" は、ｽﾀｯｸを消費するので、しばしばﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝの不具合の原因になっている事があります。特に浮動少数点変数を変換すると数百ﾊﾞｲﾄのｽﾀｯｸを消費します。
　　できるだけ標準関数の"spritf()"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 *pforamt, int data0)

　　char *pbuff　　展開するﾊﾞｯﾌｧのｱﾄﾞﾚｽ
　　char *pformat　変換する書式
　　int　data0　　変換するﾃﾞｰﾀ

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 *pforamt, int data0,
　　int data1)

　　char *pbuff　　展開するﾊﾞｯﾌｧのｱﾄﾞﾚｽ
　　char *pformat　変換する書式
　　int　data0　　変換するﾃﾞｰﾀ
　　int　data1　　変換するﾃﾞｰﾀ

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_sprintf3

要約　　void orb_sprintf3(char *pbuff, char *pforamt, int data0,
　　int data1,
　　int data2)

　　char *pbuff　　展開するﾊﾞｯﾌｧのｱﾄﾞﾚｽ
　　char *pformat　変換する書式
　　int　data0　　変換するﾃﾞｰﾀ0
　　int　data1　　変換するﾃﾞｰﾀ1
　　int　data2　　変換するﾃﾞｰﾀ2

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:%03d, %03d, %03d\r\n", data, data+10, data+20) */
    orb_sprintf3(work, "DATA:%03d, %03d, %03d\r\n", data, data+10, data+20 );
    rs_tx_str(work);
}
```

実行結果
DATA:010,020,030

orb_sprintf4

要約　　void orb_sprintf4(char *pbuff, char *pforamt, int data0,
　　int data1,
　　int data2,
　　int data3)

　　char *pbuff　展開するﾊﾞｯﾌｧのｱﾄﾞﾚｽ
　　char *pformat　変換する書式
　　int　data0　変換するﾃﾞｰﾀ0
　　int　data1　変換するﾃﾞｰﾀ1
　　int　data2　変換するﾃﾞｰﾀ2
　　int　data3　変換するﾃﾞｰﾀ3

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\r\n", data, data+10, data+20, data+30) */
    orb_sprintf4(work, "DATA:%d, %d, %d, %d\r\n", data, data+10, data+20, data+30);
    rs_tx_str(work);
}
```

実行結果
DATA:10, 20, 30, 40

orb_sprintf5

要約　　void orb_sprintf5(char *pbuff, char *pforamt, int data0,
　　int data1,
　　int data2,
　　int data3,
　　int data4,
　　int data5)

	char *pbuff	展開するﾊﾞｯﾌｧのｱﾄﾞﾚｽ
	char *pformat	変換する書式
	int data0	変換するﾃﾞｰﾀ0
	int data1	変換するﾃﾞｰﾀ1
	int data2	変換するﾃﾞｰﾀ2
	int data3	変換するﾃﾞｰﾀ3
	int data4	変換するﾃﾞｰﾀ4

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\r\n", data, data+10, data+20, data+30,
data+40) */
    orb_sprintf5(work, "DATA:%d, %d, %d, %d, %d\r\n", data, data+10, data+20, data+30,
data+40);

    rs_tx_str(work);
}
```

実行結果
DATA:10, 20, 30, 40, 50

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	展開するﾊﾞｯﾌｧのｱﾄﾞﾚｽ
	char *pformat	変換する書式
	int data0	変換するﾃﾞｰﾀ0
	int data1	変換するﾃﾞｰﾀ1
	int data2	変換するﾃﾞｰﾀ2
	int data3	変換するﾃﾞｰﾀ3
	int data4	変換するﾃﾞｰﾀ4
	int data5	変換するﾃﾞｰﾀ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 *pforamt, int data0,
　　int data1,
　　int data2,
　　int data3,
　　int data4,
　　int data5,
　　int data6)

	char *pbuff	展開するﾊﾞｯﾌｧのｱﾄﾞﾚｽ
	char *pformat	変換する書式
	int data0	変換するﾃﾞｰﾀ0
	int data1	変換するﾃﾞｰﾀ1
	int data2	変換するﾃﾞｰﾀ2
	int data3	変換するﾃﾞｰﾀ3
	int data4	変換するﾃﾞｰﾀ4
	int data5	変換するﾃﾞｰﾀ5
	int data6	変換するﾃﾞｰﾀ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 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足　　ﾃﾞﾌｫﾙﾄGCC ID兼KXS14における希望GCCを設定する。

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”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */

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 : ｲﾝﾊﾞｳﾝﾄﾞﾒｯｾｰｼﾞをすぐ送信
1 : ﾒｯｾｰｼﾞ/ﾚﾎﾟｰﾄは SC の中でｷｭｰｲﾝｸﾞされ、GCC からﾎﾟｰﾘﾝｸﾞされるのを待つ

解説　　ﾃﾞﾌｫﾙﾄﾎﾟｰﾙﾄﾞを設定する

戻値　　0 : 設定成功
1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足

get_kxs02

要約　　unsigned char get_kxs02(void)

解説　　ﾃﾞﾌｫﾙﾄ ﾎﾟｰﾙﾄﾞを獲得する

戻値　　ﾃﾞﾌｫﾙﾄ polled を獲得する
0 : ｲﾝﾊﾞｳﾝﾄﾞﾒｯｾｰｼﾞをすぐ送信
1 : ﾒｯｾｰｼﾞ/ﾚﾎﾟｰﾄは SC の中でｷｭｰｲﾝｸﾞされ、GCC からﾎﾟｰﾘﾝｸﾞされるのを待つ

補足

例：

```
#include “kmelib.h”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */

void main(void)
{
    unsigned char polled;

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

set_kxs03

要約　　int set_kxs03(unsigned char value)

unsigned char value (0 - 3)
0 : 不急（最も低いﾌﾟﾗｲｵﾘﾃｨ）
1 : 通常
2 : 至急
3 : 速達（ｲﾝﾊﾞｳﾝﾄﾞのみ、最も高いﾌﾟﾗｲｵﾘﾃｨ）

解説　　ﾃﾞﾌｫﾙﾄﾌﾟﾗｲｵﾘﾃｨを設定する

戻値　　0 : 設定成功
1 : 設定失敗（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足

get_kxs03

要約　　unsigned char get_kxs03(void)

解説　　ﾃﾞﾌｫﾙﾄ ﾌﾟﾗｲｵﾘﾃｨを獲得する

戻値　　ﾃﾞﾌｫﾙﾄ ﾌﾟﾗｲｵﾘﾃｨ
0 : 不急（最も低いﾌﾟﾗｲｵﾘﾃｨ）
1 : 通常
2 : 至急
3 : 速達（ｲﾝﾊﾞｳﾝﾄﾞのみ、最も高いﾌﾟﾗｲｵﾘﾃｨ）

補足

例:

```
#include “kmelib.h”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */

void main(void)
{
    unsigned char priority;

    set_kxs03(0);               /* ﾃﾞﾌｫﾙﾄ priorityを設定する */
    priority = get_kxs03();       /* ﾃﾞﾌｫﾙﾄpriorityを獲得する */
}
```

set_kxs04

要約　　int set_kxs04(unsigned char value)

　　unsigned char value : ﾃﾞﾌｫﾙﾄ ﾚﾎﾟｰﾄ O/R ｲﾝﾃﾞｨｹｰﾀ

解説　　ﾃﾞﾌｫﾙﾄﾚﾎﾟｰﾄ O/R ｲﾝﾃﾞｨｹｰﾀ(0 -3)を設定する

戻値　　0 : 設定成功
　　1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足

get_kxs04

要約　　unsigned char get_kxs04(void)

解説　　ﾃﾞﾌｫﾙﾄ ﾚﾎﾟｰﾄO/R ｲﾝﾃﾞｨｹｰﾀ(0 -3)を獲得する

戻値　　ﾃﾞﾌｫﾙﾄ ﾚﾎﾟｰﾄ O/R ｲﾝﾃﾞｨｹｰﾀ(0 -3)

補足

例:

```
#include “kmelib.h”          /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */

void main(void)
{
    unsigned char or_ind;

    set_kxs04(0);          /* ﾃﾞﾌｫﾙﾄ ﾚﾎﾟｰﾄO/R ｲﾝﾃﾞｨｹｰﾀ(0 -3)を設定する*/
    priority = get_kxs04();  /* ﾃﾞﾌｫﾙﾄ ﾚﾎﾟｰﾄO/R ｲﾝﾃﾞｨｹｰﾀ(0 -3)を獲得する*/
}
```

set_kxs05

要約　　int set_kxs05(unsigned char value)

　　unsigned char value : ﾒｯｾｰｼﾞ O/R ｲﾝﾃﾞｨｹｰﾀ(0 - 15)

解説　　ﾃﾞﾌｫﾙﾄ ﾒｯｾｰｼﾞ O/R ｲﾝﾃﾞｨｹｰﾀ(0 - 15)を設定する

戻値　　0 : 設定成功
　　1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足

get_kxs05

要約　　unsigned char get_kxs05(void)

解説　　ﾃﾞﾌｫﾙﾄ ﾒｯｾｰｼﾞ O/R ｲﾝﾃﾞｨｹｰﾀ(0 - 15)を獲得する

戻値　　ﾒｯｾｰｼﾞ O/R ｲﾝﾃﾞｨｹｰﾀ(0 - 15)

補足

例：

```
#include “kmelib.h”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */

void main(void)
{
    unsigned char or_ind;

    set_kxs05(0);             /* ﾃﾞﾌｫﾙﾄ ﾒｯｾｰｼﾞ O/R ｲﾝﾃﾞｨｹｰﾀを設定する */
    priority = get_kxs05();   /* ﾃﾞﾌｫﾙﾄ ﾒｯｾｰｼﾞ O/R ｲﾝﾃﾞｨｹｰﾀを獲得する */
}
```

set_kxs06

要約　　int set_kxs06(unsigned char value)

　　unsigned char value : Ack ﾚﾍﾞﾙ (0 - 4)

解説　　ﾃﾞﾌｫﾙﾄ Ack ﾚﾍﾞﾙ (0 - 4)を設定する

戻値　　0 : 設定成功
　　1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足

get_kxs06

要約　　unsigned char get_kxs06(void)

解説　　ﾃﾞﾌｫﾙﾄAck ﾚﾍﾞﾙ (0 - 4)を獲得する

戻値　　Ack ﾚﾍﾞﾙ (0 - 4)

補足

例：

```
#include “kmelib.h”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */

void main(void)
{
    unsigned char ack_level;

    set_kxs06(0);               /* ﾃﾞﾌｫﾙﾄAck ﾚﾍﾞﾙを設定する */
    ack_level = get_kxs05();     /* ﾃﾞﾌｫﾙﾄAck ﾚﾍﾞﾙ を獲得する */
}
```

set_kxs07

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

　　unsigned char type　: ﾃﾞﾌｫﾙﾄﾒｯｾｰｼﾞﾎﾞﾃﾞｨﾀｲﾌﾟ (0 -15)
　　unsigned char object: ｻﾌﾞﾀｲﾌﾟ(0 - 4)

解説　　ﾃﾞﾌｫﾙﾄﾒｯｾｰｼﾞﾎﾞﾃﾞｨﾀｲﾌﾟ (0 -15)を設定する

戻値　　0 : 設定成功
　　1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足

get_kxs07

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

　　unsigned char *type　: ﾃﾞﾌｫﾙﾄﾒｯｾｰｼﾞﾎﾞﾃﾞｨﾀｲﾌﾟ (0 -15)
　　unsigned char *object: ｻﾌﾞﾀｲﾌﾟ(0 - 4)

解説　　ﾃﾞﾌｫﾙﾄﾒｯｾｰｼﾞﾎﾞﾃﾞｨﾀｲﾌﾟ (0 -15)を獲得する

戻値　　なし

補足

例:

```
#include “kmelib.h”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */

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

    set_kxs07(0, 5);               /* ﾃﾞﾌｫﾙﾄﾒｯｾｰｼﾞﾎﾞﾃﾞｨﾀｲﾌﾟを設定する */
    set_kxs07(14, 0/*dummy*/);     /* ﾃﾞﾌｫﾙﾄﾒｯｾｰｼﾞﾎﾞﾃﾞｨﾀｲﾌﾟを設定する */
    get_kxs05(&type, &sub_type);   /* ﾃﾞﾌｫﾙﾄﾒｯｾｰｼﾞﾎﾞﾃﾞｨﾀｲﾌﾟを獲得する */
}
```

set_kxs08

要約　　int set_kxs08(unsigned char value)

　　unsigned char value : ｻｰﾋﾞｽﾀｲﾌﾟ (0 - 4, 10 - 14)

解説　　ﾃﾞﾌｫﾙﾄｻｰﾋﾞｽﾀｲﾌﾟ (0 - 4, 10 - 14)を設定する

戻値　　0 : 設定成功
　　1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足

get_kxs08

要約　　unsigned char get_kxs08(void)

解説　　ｻｰﾋﾞｽﾀｲﾌﾟ (0 - 4, 10 - 14)を獲得する

戻値　　ｻｰﾋﾞｽﾀｲﾌﾟ (0 - 4, 10 - 14)

補足

例：

```
#include “kmelib.h”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */

void main(void)
{
    unsigned char service;

    set_kxs08(2);                  /* ﾃﾞﾌｫﾙﾄｻｰﾋﾞｽﾀｲﾌﾟを設定する */
    service = get_kxs08();          /* ﾃﾞﾌｫﾙﾄ ｻｰﾋﾞｽﾀｲﾌﾟを獲得する */
}
```

set_kxs10

要約　　int set_kxs10(unsigned int value)

　　unsigned char value : ｲﾝﾀｰﾊﾞﾙ(0 - 65535)[分]

解説　　ﾚﾎﾟｰﾄｲﾝﾀｰﾊﾞﾙを設定する

戻値　　0 : 設定成功
　　1 : 設定失敗（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足　　ﾚﾎﾟｰﾄｲﾝﾀｰﾊﾞﾙは、GCCからのﾎﾟｰﾘﾝｸﾞｺﾏﾝﾄﾞ（複数ﾎﾟｰﾘﾝｸﾞﾚﾎﾟｰﾄ）を受信した時に参照される。

get_kxs10

要約　　unsigned int get_kxs10(void)

解説　　ﾚﾎﾟｰﾄｲﾝﾀｰﾊﾞﾙを獲得する

戻値　　ﾚﾎﾟｰﾄｲﾝﾀｰﾊﾞﾙ(0 - 65535)[分]

補足

例：

```
#include “kmelib.h”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */

void main(void)
{
    unsigned int interval;

    set_kxs10(5);                  /* ﾚﾎﾟｰﾄｲﾝﾀｰﾊﾞﾙを設定する */
    interval = get_kxs10();         /* ﾚﾎﾟｰﾄｲﾝﾀｰﾊﾞﾙを獲得する  */
}
```

set_kxs11

要約　　int set_kxs11(unsigned char value)

　　unsigned char value: ﾚﾎﾟｰﾄ数 (0 - 255)

解説　　複数ﾎﾟｰﾘﾝｸﾞ時のﾚﾎﾟｰﾄ数を設定する

戻値　　0 : 設定成功
　　1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足　　ﾚﾎﾟｰﾄ数は、GCCからのﾎﾟｰﾘﾝｸﾞｺﾏﾝﾄﾞ (複数ﾎﾟｰﾘﾝｸﾞﾚﾎﾟｰﾄ) を受信した時に参照される。

get_kxs11

要約　　unsigned char get_kxs11(void)

解説　　複数ﾎﾟｰﾘﾝｸﾞ時のﾚﾎﾟｰﾄ数を獲得する

戻値　　ﾚﾎﾟｰﾄ数 (0 - 255).

補足

例：

```
#include “kmelib.h”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */

void main(void)
{
    unsigned char rpt_num;

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

set_kxs12

要約　　int set_kxs12(unsigned int value)

　　unsigned char value : Interval (0 - 65535) [分]

解説　　測位ﾚﾎﾟｰﾄｲﾝﾀｰﾊﾞﾙを設定する

戻値　　0 : 設定成功
　　1 : 設定失敗（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足　　測位ﾚﾎﾟｰﾄｲﾝﾀｰﾊﾞﾙは、GCCからのﾎﾟｰﾘﾝｸﾞｺﾏﾝﾄﾞ（複数ﾎﾟｰﾘﾝｸﾞ測位ﾚﾎﾟｰﾄ）を受信した時に参照される。

get_kxs12

要約　　unsigned int get_kxs12(void)

解説　　測位ﾚﾎﾟｰﾄｲﾝﾀｰﾊﾞﾙを獲得する

戻値　　測位ﾚﾎﾟｰﾄｲﾝﾀｰﾊﾞﾙ(0 - 65535 分)

補足

例：

```
#include “kmelib.h”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */

void main(void)
{
    unsigned int interval;

    set_kxs12(10);                /* 測位ﾚﾎﾟｰﾄｲﾝﾀｰﾊﾞﾙを設定する */
    interval = get_kxs12();         /* 測位ﾚﾎﾟｰﾄｲﾝﾀｰﾊﾞﾙを獲得する */
}
```

set_kxs13

要約　　int set_kxs13(unsigned char value)

　　unsigned char value : 測位ﾚﾎﾟｰﾄ数(0 - 255)

解説　　複数ﾎﾟｰﾘﾝｸﾞ時の測位ﾚﾎﾟｰﾄ数を設定する

戻値　　0 : 設定成功
　　1 : 設定失敗（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足　　測位ﾚﾎﾟｰﾄ数は、GCC からのﾎﾟｰﾘﾝｸﾞｺﾏﾝﾄﾞ（複数ﾎﾟｰﾘﾝｸﾞ測位ﾚﾎﾟｰﾄ）を受信した時に参照される。

get_kxs13

要約　　unsigned char get_kxs13(void)

解説　　複数ﾎﾟｰﾘﾝｸﾞ時の測位ﾚﾎﾟｰﾄ数を獲得する

戻値　　測位ﾚﾎﾟｰﾄ数(0 - 255)

補足

例：

```
#include “kmelib.h”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */

void main(void)
{
    unsigned char pos_rpt_num;

    set_kxs13(0);              /* 複数ﾎﾟｰﾘﾝｸﾞ時の測位ﾚﾎﾟｰﾄ数を設定する */
    pos_rpt_num = get_kxs13();  /* 複数ﾎﾟｰﾘﾝｸﾞ時の測位ﾚﾎﾟｰﾄ数を獲得する */
}
```

set_kxs14

要約　　int set_kxs14(unsigned char value)

unsigned char value : GCCｻｰﾁﾓｰﾄﾞ (0 - 4)
- 0 : 希望GCCを連続的にﾀﾞｳﾝﾘﾝｸﾊﾞﾝﾄﾞの中から検索する
- 1 : 希望GCCを１回探す。もし 見つからなければ最初に発見したﾀﾞｳﾝﾘﾝｸとのﾛｯｸを維持する。
- 2 : 最初に発見したﾀﾞｳﾝﾘﾝｸとのﾛｯｸを維持する。
- 3 : 希望GCC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、最初に発見したﾀﾞｳﾝﾘﾝｸとのﾛｯｸを維持する。
- 4 : 希望GCCを１回探す。もし見つからなければｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑ衛星かkxs01に設定されているGCCのﾀﾞｳﾝﾘﾝｸを検索し続ける。

解説　　GCC ｻｰﾁﾓｰﾄﾞを設定する

戻値　　0 : 設定成功
1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足　　希望 GCC は set_kxs01 または set_desired_ncc_id で設定します

get_kxs14

要約　　unsigned char get_kxs14(void)

解説　　GCCｻｰﾁﾓｰﾄﾞを獲得する

戻値　　GCC ｻｰﾁﾓｰﾄﾞ(0 - 4).

補足

例：

```
#include “kmelib.h”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */
void main(void)
{
    unsigned char search_mode;

    set_kxs14(1);                /* GCCｻｰﾁﾓｰﾄﾞを設定する */
    search_mode = get_kxs13();   /* GCCｻｰﾁﾓｰﾄﾞを獲得する */
}
```

set_kxs15

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

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

解説　　ﾀﾞ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 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足　　SCが衛星ﾀﾞｳﾝﾘﾝｸｱｸｳｨｼﾞ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 :ﾀﾞｳﾝﾘﾝｸﾁｬﾝﾈﾙ (24 ﾁｬﾝﾈﾙ)

解説　　ﾀﾞｳﾝﾘﾝｸﾁｬﾝﾈﾙ (24ﾁｬﾝﾈﾙ) を獲得する

戻値　　なし

補足

例：

```
#include “kme_lib.h”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */

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

    set_kxs15(0, 80);              /* ﾀﾞｳﾝﾘﾝｸﾁｬﾝﾈﾙを設定する */
    get_kxs15(channel);           /* ﾀﾞ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 : ｽﾚｯｼｭ (1 - 255)

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 (ﾊﾟﾗﾒｰﾀｴﾗｰ)
　　2 : 排他ｴﾗｰ:KXS17(1)の時は KXS16(51 - 100)にできない

補足　　異なったﾀﾞｳﾝﾘﾝｸの獲得を試みる前に連続する指定ﾌﾚｰﾑ (KXS17) で許されるﾀﾞｳﾝ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を獲得する

戻値　　最大ﾁｪｯｸｻﾑｴﾗｰ数(1 - 255).

補足

例：

```
#include “kme_lib.h”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */

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 (ﾊﾟﾗﾒｰﾀｴﾗｰ)
　　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”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */

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 : ﾓｰﾄﾞ
 0 : OFF
 1 : ON

解説　　連続測位ﾓｰﾄﾞを設定する

戻値　　0 : 設定成功
1 : 設定失敗（ﾊﾟﾗﾒｰﾀｴﾗｰ）
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　　連続測位ﾓｰﾄﾞを獲得する

戻値　　ﾓｰﾄﾞ
0 : 連続測位ﾓｰﾄﾞOFF
1 : 連続測位ﾓｰﾄﾞ ON

補足

例：

```
#include “kme_lib.h”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */

void main(void)
{
    unsigned char mode;

    set_kxs18(0);           /* 連続測位ﾓｰﾄﾞをOFFに設定する */
    count = get_kxs18();    /* 連続測位ﾓｰﾄﾞを獲得する */
}
```

set_kxs19

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

unsigned char rate : ｴﾌｪﾒﾘｰ収集間隔 (4,8,12,16秒)
unsigned char point: ｴﾌｪﾒﾘｰ数
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　　ﾄﾞｯﾌﾟﾗｰ測位時のｴﾌｪﾒﾘｰ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（ﾊﾟﾗﾒｰﾀｴﾗｰ）
2 : 排他ｴﾗｰ

補足

get_kxs19

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

unsigned char *rate : ｴﾌｪﾒﾘｰ収集間隔
unsigned char *point: ｴﾌｪﾒﾘｰ数

解説　　ﾄﾞｯﾌﾟﾗｰ測位時のｴﾌｪﾒﾘｰ数とその収集間隔を獲得する

戻値　　なし

補足

例：

```
#include “kme_lib.h”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */
void main(void)
{
    unsigned char rate, point;
    set_kxs19(4, 50);            /* ｴﾌｪﾒﾘｰ数とその収集間隔を設定する */
    get_kxs19(&rate, &point);    /* ｴﾌｪﾒﾘｰ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 : ｴｲｼﾞ (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（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足　　GCC からﾎﾟｰﾘﾝｸﾞｺﾏﾝﾄﾞ（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”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */
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: ｸｫﾘﾃｨｲﾝﾃﾞｨｹｰﾀ (0 - 15)

解説　　測位結果の最小有効ｸｫﾘﾃｨｲﾝﾃﾞｨｹｰﾀを設定する

戻値　　0 : 設定成功
　　1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足

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ﾃﾞｨｹｰﾀを獲得する

戻値　　ｸｫﾘﾃｨｲﾝﾃﾞｨｹｰﾀ(0 - 15)

補足

例：

```
#include “kme_lib.h”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */
void main(void)
{
    unsigned char indicator;

    set_kxs21(3);                /* ｸｫﾘﾃｨｲﾝﾃﾞｨｹｰﾀを設定する */
    indicator = get_kxs21();      /* ｸｫﾘﾃｨｲﾝﾃﾞ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（ﾊﾟﾗﾒｰﾀｴﾗｰ）

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”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */
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（ﾊﾟﾗﾒｰﾀｴﾗｰ）
　　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”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */

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 : ﾓｰﾄﾞ
　　0 : 測位機能OFF
　　1 : 測位機能 ON

解説　　測位ﾓｰﾄﾞを設定する

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

補足

get_kxs24

要約　　unsigned char get_kxs24(void)

解説　　測位ﾓｰﾄﾞを獲得する

戻値　　測位ﾓｰﾄﾞ
　　0: OFF
　　1: ON

補足

例：

```
#include “kme_lib.h”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */

void main(void)
{
    unsigned char mode;

    set_kxs24(1);               /* 測位機能を有効にする */
    mode = get_kxs24();         /* 測位ﾓｰﾄﾞを獲得する */
}
```

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　　KXA/B ｺﾏﾝﾄﾞでの GPS 測位情報のﾌｫｰﾏｯﾄ(高度)を設定する

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

補足　　.

get_kxs25

要約　　int get_kxs25(void)

解説　　KXA/Bｺﾏﾝﾄﾞでの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”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */

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 に対するﾎﾟｰﾘﾝｸﾞ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（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足　　DTEにﾎﾟｰﾘﾝｸﾞｺﾏﾝﾄﾞ（ｼｽﾃﾑｱﾅｳﾝｽﾒﾝﾄ）を送信した後、この時間内の応答をﾎﾟｰﾘﾝｸﾞの応答と判断する。

get_kxs26

要約　　unsigned char get_kxs26(void)

解説　　DTEに対するﾎﾟｰﾘﾝｸﾞ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”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */

void main(void)
{
    unsigned char timeout;

    set_kxs26(5);               /* ﾎﾟｰﾘﾝｸﾞの応答の待ち時間を設定する */
    timeout = get_kxs26();       /* ﾎﾟｰﾘﾝｸﾞ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　　ｼﾘｱﾙﾌﾟﾛﾄｺﾙの ACK 時間を設定する

戻値　　0 : 設定成功
　　1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足　　ﾊﾟｹｯﾄの最後のﾊﾞｲﾄを送った後、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　　ｼﾘｱﾙﾌﾟﾛﾄｺﾙのACK時間を獲得する

戻値　　時間 (1 - 30 秒)
補足

例：

```
#include “kme_lib.h”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */

void main(void)
{
    unsigned char timeout;

    set_kxs27(5);                /* ｼﾘｱﾙﾌﾟﾛﾄｺﾙのACK時間を設定する */
    timeout = get_kxs27();       /* ｼﾘｱﾙﾌﾟﾛﾄｺﾙの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　　ｼﾘｱﾙﾌﾟﾛﾄｺﾙのﾘﾄﾗｲ回数を設定する

戻値　　0 : 設定成功
　　1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

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ﾎﾞｰﾄする
　　(0=ｱﾎﾞｰﾄしない)

get_kxs28

要約　　unsigned char get_kxs28(void)

解説　　ｼﾘｱﾙﾌﾟﾛﾄｺﾙ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”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */

void main(void)
{
    unsigned char count;

    set_kxs28(5);                /* ｼﾘｱﾙﾌﾟﾛﾄｺﾙのﾘﾄﾗｲ回数を設定する */
    count = get_kxs28();         /* ｼﾘｱﾙﾌﾟﾛﾄｺﾙ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 : ﾓｰﾄﾞ
　　0 : ｱﾎﾞｰﾄﾚﾎﾟｰﾄを送信しない
　　1 : ｱﾎﾞｰﾄﾚﾎﾟｰﾄを送信する

解説　　ｱﾎﾞｰﾄﾚﾎﾟｰﾄ送信ﾓｰﾄﾞを設定する

戻値　　0 : 設定成功
　　1 : 設定失敗（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足

get_kxs29

要約　　unsigned char get_kxs29(void)

解説　　ｱﾎﾞｰﾄﾚﾎﾟｰﾄ送信ﾓｰﾄﾞを獲得する

戻値　　0 : ｱﾎﾞｰﾄﾚﾎﾟｰﾄを送信しない
　　1 : ｱﾎﾞｰﾄﾚﾎﾟｰﾄを送信する

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　　: ﾎﾟｰﾙﾄﾞ (0, 1)
　　unsigned char srv_type　: ｻｰﾋﾞｽﾀｲﾌﾟ (0 - 4, 10 - 14)
　　unsigned char or_ind　　: O/Rｲﾝﾃﾞｨｹｰﾀ(1 - 3)
　　unsigned char *inf　　: ﾃﾞｰﾀ(0x00 - 0xff) * 6ﾊﾞｲﾄ

解説　　ｱﾎﾞｰﾄﾚﾎﾟｰﾄの内容を設定する

戻値　　0 : 設定成功
　　1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)
　　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 gcc_id　　: GCC番号 (0 - 255)
　　unsigned char polled　　: ﾎﾟｰﾙﾄﾞ (0, 1)
　　unsigned char srv_type　: ｻｰﾋﾞｽﾀｲﾌﾟ (0 - 4, 10 - 14)
　　unsigned char or_ind　　: O/Rｲﾝﾃﾞｨｹｰﾀ(1 - 3)
　　unsigned char *inf　　: ﾃﾞｰﾀ(0x00 - 0xff) * 6ﾊﾞｲﾄ

解説　　ｱﾎﾞｰﾄﾚﾎﾟｰﾄの内容を獲得する

戻値

補足

例：

```
#include “kme_lib.h”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */

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

    set_kxs30(1, 0, 2, 1, “012345”);   /* ｱﾎﾞｰﾄﾚﾎﾟｰﾄを設定する */
    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: ﾓｰﾄﾞ (0, 1)
  0 : ﾌﾟﾛﾄｺﾙﾓｰﾄﾞ
  1 : ﾊﾞｲﾄﾓｰﾄﾞ

解説　　RS232C の通信ﾓｰﾄﾞを設定する

戻値　　0 : 設定成功
1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足

get_kxs31

要約　　unsigned char get_kxs31(void)

解説　　RS232Cの通信ﾓｰﾄﾞを獲得する

戻値　　ﾓｰﾄﾞ (0:ﾌﾟﾛﾄｺﾙ, 1:ﾊﾞｲﾄ)

補足

例1:

```
#include “kme_lib.h”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */

void main(void)
{
    /* ﾌﾟﾛﾄｺﾙﾓｰﾄﾞに設定 */
    set_kxa00();      /* KXA設定を解除 */
    set_kxb00();      /* KXB設定を解除 */
    set_kxs31(0);     /* ﾌﾟﾛﾄｺﾙﾓｰﾄﾞを設定する */
}
```

例2:

```
#include “kme_lib.h”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */

void main(void)
{
BMODE_TRGGER_CODE bmode_code;

    /* ﾊﾞｲﾄﾓｰﾄﾞに設定 */
    set_kxa00();      /* KXA設定を解除 */
    set_kxb00();      /* KXB設定を解除 */
    set_kxs31(1);     /* ﾊﾞｲﾄﾓｰﾄﾞに設定 */

    /* ﾊﾞｲﾄﾓｰﾄﾞのﾊﾟﾗﾒｰﾀを設定 */
    set_kxs32(1);         /* ﾊﾞｲﾄﾓｰﾄﾞﾄﾘｶﾞｰを設定する */
    set_kxs33(1);         /* ﾊﾞｲﾄﾓｰﾄﾞﾀｲﾑｱｳﾄを設定する */
    set_kxs34(200);       /* ﾊﾞｲﾄﾓｰﾄﾞ長を設定する */
    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);         /* ﾊﾞｲﾄﾓｰﾄﾞのﾒｯｾｰｼﾞﾀｲﾌﾟを設定する */
}
```

set_kxs32

要約　　int set_kxs32(unsigned char value)

unsigned char value : ﾄﾘｶﾞｰ (0, 1)
0 : 最初のﾃﾞｰﾀを受信してﾊﾞｲﾄﾓｰﾄﾞ長受信あるいはﾊﾞｲﾄﾓｰﾄﾞﾀｲﾑｱｳﾄ
秒経過したら送信開始
1 : ﾊﾞｲﾄﾓｰﾄﾞRX_SOM/RX_EOMを受信したら送信開始

解説　　ﾊﾞｲﾄﾓｰﾄﾞﾄﾘｶﾞｰを設定する

戻値　　0 : 設定成功
1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足

get_kxs32

要約　　unsigned char get_kxs32(void)

解説　　ﾊﾞｲﾄﾓｰﾄﾞﾄﾘｶﾞｰを獲得する

戻値　　ﾄﾘｶﾞｰ(0, 1).

補足

set_kxs33

要約　　int set_kxs33(unsigned int value)

　　unsigned char value : 時間 (1 - 604800) [分]

解説　　ﾊﾞｲﾄﾓｰﾄﾞﾀｲﾑｱｳﾄを設定する

戻値　　0 : 設定成功
　　1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足　　set_kxs32 参照

get_kxs33

要約　　unsigned int get_kxs33(void)

解説　　ﾊﾞｲﾄﾓｰﾄﾞﾀｲﾑｱｳﾄを獲得する

戻値　　時間 (1 - 604800 秒)

補足

set_kxs34

要約　　int set_kxs34(unsigned int value)

　　unsigned char value : ﾊﾞｲﾄ数 (1 - ｲﾝﾊﾞｳﾝﾄﾞｷｭｰｻｲｽﾞ)

解説　　ﾊﾞｲﾄﾓｰﾄﾞ長を設定する

戻値　　0 : 設定成功
　　1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)
　　2 : 排他ｴﾗｰ: KXS48

補足　　set_kxs32 参照

get_kxs34

要約　　unsigned int get_kxs34(void)

解説　　ﾊﾞｲﾄﾓｰﾄﾞ長を獲得する

戻値　　長さ (1 - ｲﾝﾊﾞｳﾝﾄﾞｷｭｰｻｲｽﾞ).

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　　ﾊﾞｲﾄﾓｰﾄﾞ時の、送信・受信 SOM,EOM を設定する

戻値　　0 : 設定成功
1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)
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　　ﾊﾞｲﾄﾓｰﾄﾞ時の、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 : ｲﾝﾊﾞｳﾝﾄﾞﾒｯｾｰｼﾞ
- 1 : ｲﾝﾊﾞｳﾝﾄﾞﾚﾎﾟｰﾄ
- 2 : ｲﾝﾊﾞｳﾝﾄﾞｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑ

解説　　ﾊﾞｲﾄﾓｰﾄﾞﾒｯｾｰｼﾞﾀｲﾌﾟを設定する

戻値　　0 : 設定成功
1 : 設定失敗（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足　　ﾊﾞｲﾄﾓｰﾄﾞのﾃﾞｰﾀをﾚﾎﾟｰﾄ/ﾒｯｾｰｼﾞ/ﾃﾞｰﾀｸﾞ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　　ﾊﾞｲﾄﾓｰﾄﾞﾒｯｾｰｼﾞﾀｲﾌﾟを獲得する

戻値　　ﾒｯｾｰｼﾞﾀｲﾌﾟ (0 - 2)
- 0 : ｲﾝﾊﾞｳﾝﾄﾞﾒｯｾｰｼﾞ
- 1 : ｲﾝﾊﾞｳﾝﾄﾞﾚﾎﾟｰﾄ
- 2 : ｲﾝﾊﾞｳﾝﾄﾞｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑ

補足

set_kxs37

要約　　int set_kxs37(unsigned char value)

unsigned char value : ﾓｰﾄﾞ
  0 : OFF
  1 : ON

解説　　ﾊﾟﾜｰﾀﾞｳﾝﾓｰﾄﾞを設定する

戻値　　0 : 設定成功
1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足

get_kxs37

要約　　unsigned char get_kxs37(void)

解説　　ﾊﾟﾜｰﾀﾞｳﾝﾓｰﾄﾞを獲得する

戻値　　ﾓｰﾄﾞ
0 : OFF
1 : ON

補足

例:

```
#include “kme_lib.h”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */

void main(void)
{
    /* set that the power down mode is on. */
    set_kxs37(1);                 /* ﾊﾟﾜｰﾀﾞｳﾝﾓｰﾄﾞを設定する */

    set_kxs38(30);                /* ﾊﾟﾜｰﾀﾞｳﾝﾐﾆﾏﾑｲﾝﾀｰﾊﾞﾙを設定する */
    set_kxs23(34.3333, -45.6666);  /* 緯度・経度を設定する */

    set_kxs39(300);               /* ｲﾝｱｸﾃｨﾌﾞｲﾝﾀｰﾊﾞﾙを設定する*/
}
```

set_kxs38

要約 int set_kxs38(unsigned long value)

unsigned char value : 時間 (0 - 535600) [分]

解説 ﾊﾟﾜｰﾀﾞｳﾝﾐﾆﾏﾑｲﾝﾀｰﾊﾞﾙを設定する

戻値 0 : 設定成功
1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足 ﾊﾟﾜｰﾀﾞｳﾝﾐﾆﾏﾑｲﾝﾀｰﾊﾞﾙは、衛星とのﾘﾝｸが切れて次の衛星飛来時刻までﾊﾟﾜｰﾀﾞｳﾝする時に参照される。ﾊﾟﾜｰﾀﾞｳﾝﾐﾆﾏﾑｲﾝﾀｰﾊﾞﾙ時間以降の最初の衛星飛来時刻までﾊﾟﾜｰﾀﾞｳﾝする。
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、ﾊﾟﾜｰﾀﾞｳﾝﾐﾆﾏﾑｲﾝﾀｰﾊﾞﾙが0でない時に起動計算を行う。これ以外は、ｲﾝｱｸﾃｨﾌﾞｲﾝﾀｰﾊﾞﾙだけﾊﾟﾜｰﾀﾞｳﾝする。

get_kxs38

要約 unsigned char get_kxs38(void)

解説 ﾊﾟﾜｰﾀﾞｳﾝﾐﾆﾏﾑｲﾝﾀｰﾊﾞﾙを獲得する

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 ｲﾝｱｸﾃｨﾌﾞｲﾝﾀｰﾊﾞﾙを設定する

戻値 0 : 設定成功
1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足 set_kxs38 参照

get_kxs39

要約 unsigned long get_kxs39(void)

解説 ｲﾝｱｸﾃｨﾌﾞｲﾝﾀｰﾊﾞﾙを獲得する

戻値 時間 (0 - 86400 秒)

補足

set_kxs40

要約　　int set_kxs40(unsigned char value)

unsigned char value : ﾓｰﾄﾞ
  0 : OFF
  1 : ON

解説　　ﾊﾟﾜｰｾｰﾌﾞﾓｰﾄﾞを設定する

戻値　　0 : 設定成功
1 : 設定失敗（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足　　端末下りIDに対応するﾌﾚｰﾑとﾌﾚｰﾑ0だけ無線受信系ﾌﾞﾛｯｸの電源をONする

get_kxs40

要約　　unsigned char get_kxs40(void)

解説　　ﾊﾟﾜｰｾｰﾌﾞﾓｰﾄﾞを獲得する

戻値　　ﾓｰﾄﾞ
 0 : OFF
 1 : ON

補足

例:

```
#include “kme_lib.h”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */

void main(void)
{
    unsigned char mode;

    set_kxs40(1);                /* ﾊﾟﾜｰｾｰﾌﾞﾓｰﾄﾞを設定する */
    mode = get_kxs40();          /* ﾊﾟﾜｰｾｰﾌﾞﾓｰﾄﾞを獲得する */
}
```

set_kxs41

要約　　int set_kxs41(unsigned char value)

unsigned char value: ﾓｰﾄﾞ
0 : CTSが非ｱｸﾃｨﾌﾞで送信中止
1 : ﾌﾛｰ制御無し

解説　　ｲﾝﾊﾞｳﾝﾄﾞﾌﾛｰ制御を設定する

戻値　　0 : 設定成功
1 : 設定失敗（ﾊﾟﾗﾒｰﾀｴﾗｰ）
2 : 排他ｴﾗｰ : KXS44(0) ／KXS41(1).

補足　　DTEからﾊﾟｹｯﾄが送られてくるのを中止させるためには、CTSを非ｱｸﾃｨﾌﾞにする

get_kxs41

要約　　unsigned char get_kxs41(void)

解説　　ｲﾝﾊﾞｳﾝﾄﾞﾌﾛｰ制御を設定を獲得する

戻値　　0 : CTSが非ｱｸﾃｨﾌﾞで送信中止
1 : ﾌﾛｰ制御無し

補足

set_kxs42

要約　　int set_kxs42(unsigned char value)

unsigned char value: ﾓｰﾄﾞ
0 : RTSｱｸﾃｨﾌﾞで送信中止
1 : ﾌﾛｰ制御なし

解説　　ｱｳﾄﾊﾞｳﾝﾄﾞﾌﾛｰ制御を設定する

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

補足　　RTSがｱｸﾃｨﾌﾞの時、端末はDTEへ送信しない

get_kxs42

要約　　unsigned char get_kxs42(void)

解説　　ｱｳﾄﾊﾞｳﾝﾄﾞﾌﾛｰ制御を獲得する

戻値　　0 : RTSｱｸﾃｨﾌﾞで送信中止.
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          : ﾎﾞｰﾚｰﾄ (0 :300, 1: 600, 2:1200,
                                3:2400, 4:4800, 5:9600bps)
  unsigned char parity        : ﾊﾟﾘﾃｨ (0:Even, 1:Odd, 2:None)
  unsigned char stop_bit      : ｽﾄｯﾌﾟﾋﾞｯﾄ (1:1bit, 2:2bit)
  unsigned char data_bit      : ﾃﾞｰﾀ長 (7:7bit, 8:8bit)
```

解説　　ﾒｲﾝﾎﾟｰﾄの RS232C 通信ﾓｰﾄﾞを設定する

戻値　　0: 設定成功
1: 設定失敗（ﾊﾟﾗﾒｰﾀｴﾗｰ）

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

要約　　void get_kxs43(SERIAL_PAR *serial)

```
SERIAL_PAR *serial
  unsigned char baud          : ﾎﾞｰﾚｰﾄ (0 :300, 1: 600, 2:1200,
                                3:2400, 4:4800, 5:9600bps)
  unsigned char parity        : ﾊﾟﾘﾃｨ (0:Even, 1:Odd, 2:None)
  unsigned char stop_bit      : ｽﾄｯﾌﾟﾋﾞｯﾄ (1:1bit, 2:2bit)
  unsigned char data_bit      : ﾃﾞｰﾀ長 (7:7bit, 8:8bit)
```

解説　　RS232C 通信ﾓｰﾄﾞを獲得する

戻値　　なし

補足

例:

```
#include “kme_lib.h”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */

void main(void)
{
     SERIAL_PAR seral;

    get_kxs43(&serial);          /* RS232C通信ﾓｰﾄﾞを獲得する */
}
```

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 通信ﾓｰﾄﾞを設定する

戻値　　0 : 設定成功
1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)
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通信ﾓｰﾄﾞを獲得する

戻値　　0 : 半２重
1 : 全２重
2 : 受信のみ

補足

例:

```
#include “kme_lib.h”              /* ﾍｯﾀﾞｰﾌｧｲﾙをｲﾝｸﾙｰﾄﾞする. */

void main(void)
{
    set_kxs44(1);           /* RS232C通信ﾓｰﾄﾞを設定する */
    set_kxs41(0);           /* ｲﾝﾊﾞｳﾝﾄﾞﾌﾛｰ制御ﾓｰﾄﾞを設定する */
    set_kxs42(0);           /* ｱｳﾄﾊﾞｳﾝﾄﾞﾌﾛｰ制御ﾓｰﾄﾞを設定する*/
}
```

set_kxs45

要約　　int set_kxs45(unsigned char value)

unsigned char value : ﾓｰﾄﾞ (0, 1)
0 : ｲﾝﾊﾞｳﾝﾄﾞｷｭｰが満杯時、DTEからの新しいﾒｯｾｰｼﾞは受信されない
1 : ｲﾝﾊﾞｳﾝﾄﾞｷｭｰが満杯時、DTE から新しいﾒｯｾｰｼﾞが受信されると
ｲﾝﾊﾞｳﾝﾄﾞｷｭｰの古いﾒｯｾｰｼﾞから削除されて、ｷｭｰに格納される

解説　　ｲﾝﾊﾞｳﾝﾄﾞﾒｯｾｰｼﾞﾄﾘｰﾄﾒﾝﾄを設定する

戻値　　0 : 設定成功
1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足

get_kxs45

要約　　unsigned char get_kxs45(void)

解説　　ｲﾝﾊﾞｳﾝﾄﾞﾒｯｾｰｼﾞﾄﾘｰﾄﾒﾝﾄを獲得する

戻値　　0 : ｲﾝﾊﾞｳﾝﾄﾞｷｭｰが満杯時、DTEからの新しいﾒｯｾｰｼﾞは受信されない
1 : ｲﾝﾊﾞｳﾝﾄﾞｷｭｰが満杯時、DTE から新しいﾒｯｾｰｼﾞが受信されると
ｲﾝﾊﾞｳﾝﾄﾞｷｭｰの古いﾒｯｾｰｼﾞから削除されて、ｷｭｰに格納される

補足

set_kxs46

要約　　int set_kxs46(unsigned char value)

unsigned char value : ﾓｰﾄﾞ (0, 1)
0 : ｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰが満杯時、GCCからの新しいﾒｯｾｰｼﾞは受信されない
1 : ｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰが満杯時、GCC から新しいﾒｯｾｰｼﾞが受信されると
ｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰの古いﾒｯｾｰｼﾞから削除されて、ｷｭｰに格納される

解説　　ｱｳﾄﾊﾞｳﾝﾄﾞﾒｯｾｰｼﾞﾄﾘｰﾄﾒﾝﾄを設定する

戻値　　0 : 設定成功
1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足

get_kxs46

要約　　Unsigned char get_kxs46(void)

解説　　ｱｳﾄﾊﾞｳﾝﾄﾞﾒｯｾｰｼﾞﾄﾘｰﾄﾒﾝﾄを獲得する

戻値　　0 : ｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰが満杯時、GCCからの新しいﾒｯｾｰｼﾞは受信されない
1 : ｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰが満杯時、GCC から新しいﾒｯｾｰｼﾞが受信されると
ｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰの古いﾒｯｾｰｼﾞから削除されて、ｷｭｰに格納される

補足

set_kxs47

要約　　int set_kxs47(int value)

int value : ﾓｰﾄﾞ (0, 1)
0 : GCCへのﾒｯｾｰｼﾞ送信失敗したらｷｭｰから削除する
1 : GCC へのﾒｯｾｰｼﾞ送信失敗してもｷｭｰから削除しない

解説　　ﾒｯｾｰｼﾞﾘｷｭｰｵﾌﾟｼｮﾝを設定する

戻値　　0 : 設定成功
1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足

get_kxs47

要約　　int get_kxs47(void)

解説　　ﾒｯｾｰｼﾞﾘｷｭｰｵﾌﾟｼｮﾝを獲得する

戻値　　0 : GCCへのﾒｯｾｰｼﾞ送信失敗したらｷｭｰから削除する
1 : GCCへのﾒｯｾｰｼﾞ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:

	ｲﾝﾊﾞｳﾝﾄﾞｷｭｰ	ｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰ
1:	1K	7K
2:	2K	6K
3:	3K	5K
4:	4K	4K
5:	5K	3K
6:	6K	2K
7:	7K	1K

解説　　ｲﾝﾊﾞｳﾝﾄﾞ・ｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰのｻｲｽﾞを設定する

戻値　　0: 設定成功
1: 設定失敗（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足　　ｲﾝﾊﾞｳﾝﾄﾞ・ｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰ合わせて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　　ｲﾝﾊﾞｳﾝﾄﾞ・ｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰのｻｲｽﾞを獲得する

戻値

	ｲﾝﾊﾞｳﾝﾄﾞｷｭｰ	ｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰ
1:	1K	7K
2:	2K	6K
3:	3K	5K
4:	4K	4K
5:	5K	3K
6:	6K	2K
7:	7K	1K

補足　　ｲﾝﾊﾞｳﾝﾄﾞ・ｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰ合わせて8Kbyteあり、それを上記のように分割して使用する。

set_kxs49，get_kxs49はｻﾎﾟｰﾄしていません。

set_kxs50

要約　　int set_kxs50(unsigned long value)

　　unsigned char value : ｺｰﾄﾞ (0 - 9999)

解説　　ﾋﾟﾝｺｰﾄﾞを設定する

戻値　　0 : 設定成功
　　1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足　　ﾋﾟﾝｺｰﾄﾞとは、端末－GCC 間通信におけるﾊﾟｽﾜｰﾄﾞである

get_kxs50

要約　　unsigned long get_kxs50(void)

解説　　ﾋﾟﾝｺｰﾄﾞを獲得する

戻値　　ｺｰﾄﾞ (0 - 9999)

補足

set_kxs51

要約　　int set_kxs51(unsigned char value)

unsigned char value: ﾓｰﾄﾞ (0, 1)
0 : 何もしない
1: 衛星に対しｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑﾎﾟｰﾘﾝｸﾞをかける

解説　　自動ｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑﾎﾟｰﾘﾝｸﾞを設定する

戻値　　0 : 設定成功
1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足

get_kxs51

要約　　unsigned char get_kxs51(void)

解説　　自動ｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑﾎﾟｰﾘﾝｸﾞを獲得する

戻値　　0 : 何もしない
1 : 衛星に対しｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑﾎﾟｰﾘﾝｸﾞをかける

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 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

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でｱｸﾃｨﾌﾞ（ORBCOMM 仕様）
　　1 : Hi でｱｸﾃｨﾌﾞ（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（ﾊﾟﾗﾒｰﾀｴﾗｰ）

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でｱｸﾃｨﾌﾞ（ORBCOMM 仕様）
　　1 : HI でｱｸﾃｨﾌﾞ（一般仕様）

補足

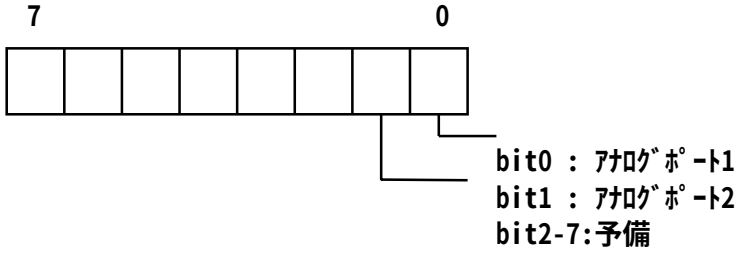
“set_kxs54” はｻﾎﾟｰﾄされてません。

“get_kxs54” はｻﾎﾟｰﾄされてません。

set_kxs55

要約　　int set_kxs55(char value)

char value: 送信するｱﾅﾛｸﾞﾎﾟｰﾄ番号



解説　　KXA/KXB ｺﾏﾝﾄﾞで送信するｱﾅﾛｸﾞﾎﾟｰﾄ番号を設定する

戻値　　0 : 設定成功
1 : 設定失敗（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足

get_kxs55

要約　　char get_kxs55(void)

解説　　KXA/KXBｺﾏﾝﾄﾞで送信するｱﾅﾛｸﾞﾎﾟｰﾄ番号を獲得する

戻値　　ｱﾅﾛｸﾞﾎﾟｰﾄ番号

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 ｺﾏﾝﾄﾞ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（ﾊﾟﾗﾒｰﾀｴﾗｰ）

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 ｺﾏﾝﾄﾞ移動距離検知の移動距離を獲得する

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 ｺﾏﾝﾄﾞ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 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

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 ｺﾏﾝﾄﾞｴﾘｱ検知のｴﾘｱを獲得する

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 ｺﾏﾝﾄﾞ速度検知の速度を設定する

戻値　　0 : 設定成功
1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

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 ｺﾏﾝﾄﾞ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 : ﾓｰﾄﾞ (0 , 1)
0 : 送信しない
1 : 履歴ﾊﾞｯﾌｧﾌﾙ時送信する

解説　　送信履歴自動送信ﾓｰﾄﾞを設定する

戻値　　0 : 設定成功
1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足

get_kxs59

要約　　unsigned char get_kxs59(void)

解説　　送信履歴自動送信ﾓｰﾄﾞを獲得する

戻値　　ﾓｰﾄﾞ (0 , 1)
0 : 送信しない
1 : 履歴ﾊﾞｯﾌｧﾌﾙ時送信する

補足

set_kxs60

要約　　int set_kxs60(unsigned char value)

unsigned char value : ﾀｲﾌﾟ (0, 1)
0 : ﾃｷｽﾄ形式
1 : ﾊﾞｲﾅﾘｰ形式
2 : ﾊﾞｲﾅﾘｰ形式(短縮)

解説　　KXA/B ｺﾏﾝﾄﾞで送信するﾃﾞ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 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足

get_kxs60

要約　　unsigned char get_kxs60(void)

解説　　KXA/Bｺﾏﾝﾄﾞで送信するﾃﾞｰﾀﾌｫｰﾏｯﾄを獲得する

戻値　　ﾀｲﾌﾟ (0, 1)
0 : ﾃｷｽﾄ形式
1 : ﾊﾞｲﾅﾘｰ形式
2 : ﾊﾞｲﾅﾘｰ形式(短縮)

補足

set_kxs61

要約　　int set_kxs61(unsigned char value)

unsigned char value: ﾓｰﾄﾞ (0, 1)
0 : 電力節減ﾓｰﾄﾞOFF
1 : 電力節減ﾓｰﾄﾞON

解説　　RS232 ﾄﾞﾗｲﾊﾞｰﾊﾟﾜｰｾｰﾌﾞﾓｰﾄﾞを設定する

戻値　　0 : 設定成功
1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足　　ﾊﾟﾜｰｾｰﾌﾞﾓｰﾄﾞONにすると、端末がDTEと通信してないときは、RS232 ﾄﾞﾗｲﾊﾞｰは自動的にﾊﾟﾜｰｾｰﾌﾞﾓｰﾄﾞになる

get_kxs61

要約　　unsigned char get_kxs61(void)

解説　　RS232 ﾄﾞﾗｲﾊﾞｰﾊﾟﾜｰｾｰﾌﾞﾓｰﾄﾞを獲得する

戻値　　ﾓｰﾄﾞ (0, 1)
0 : 電力節減ﾓｰﾄﾞOFF
1 : 電力節減ﾓｰﾄﾞON

補足

“set_kxs62” はｻﾎﾟｰﾄされてません。

“get_kxs62” はｻﾎﾟｰﾄされてません。

set_kxs63

要約　　int set_kxs63(TIME_WINDOW *time_win)

```
TIME_WINDOW *time_win
   unsigned char on_off : ﾓｰﾄﾞ(0, 1)
       0 : OFF
       1 : ON
   char lh              : ﾀｲﾑｿﾞｰﾝ ( -11 - + 12) [時間]
   unsigned char lm     : ﾀｲﾑｿﾞｰﾝ(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　　ﾀｲﾑｳｲﾝﾄﾞｳを設定する

戻値　　0 : 設定成功
1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足　　ﾀｲﾑｳｲﾝﾄﾞｳが設定されると、開始－終了時間以外は端末はﾊﾟﾜｰﾀﾞｳﾝする。

get_kxs63

要約　　void set_kxs63(TIME_WINDOW *time_win)

```
TIME_WINDOW *time_win
   unsigned char on_off : ﾓｰﾄﾞ(0, 1)
       0 : OFF
       1 : ON
   char lh              : ﾀｲﾑｿﾞｰﾝ ( -11 - + 12) [時間]
   unsigned char lm     : ﾀｲﾑｿﾞｰﾝ(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　　ﾀｲﾑｳｲﾝﾄﾞｳを獲得する

戻値　　なし

補足

set_kxs64

要約　　int set_kxs64(unsigned char value)

unsigned char value : ｸｲｯｸﾊﾟﾜｰﾀﾞｳﾝﾓｰﾄﾞ(0,1)
0 : ﾓｰﾄﾞｵﾌ
1 : ﾃﾞｰﾀ送信が成功して、送信待ちﾃﾞｰﾀがなければ即ﾊﾟﾜｰﾀﾞｳﾝ

解説　　KXB ｺﾏﾝﾄﾞでのｸｲｯｸﾊﾟﾜｰﾀﾞｳﾝﾓｰﾄを設定する

戻値　　0 : 設定成功
1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足　　.

get_kxs64

要約　　unsigned char get_kxs64(void)

解説　　KXB ｺﾏﾝﾄﾞでのｸｲｯｸﾊﾟﾜｰﾀﾞｳﾝﾓｰﾄを設定するを獲得する

戻値　　ｸｲｯｸﾊﾟﾜｰﾀﾞｳﾝﾓｰﾄﾞ(0,1)
0 : ﾓｰﾄﾞｵﾌ
1 : ﾃﾞｰﾀ送信が成功して、送信待ちﾃﾞｰﾀがなければ即ﾊﾟﾜｰﾀﾞｳﾝ

補足

set_kxs65

要約　　int set_kxs65(unsigned char value)

unsigned char value : ﾓｰﾄﾞ (0,1)
0 : 何も送信しない
1 : 測位ﾚﾎﾟｰﾄを送信する

解説　　連続測位、DTE からの測位要求で、測位終了後に GCC に測位ﾚﾎﾟ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（ﾊﾟﾗﾒｰﾀｴﾗｰ）

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に測位ﾚﾎﾟｰﾄを送信するかどうかの設定を獲得する

戻値　　ﾓｰﾄﾞ (0,1)
0 : 何も送信しない
1 : 測位ﾚﾎﾟｰﾄを送信する

補足

set_kxs66

要約　　int set_kxs66(unsigned char value)

unsigned char value : ﾓｰﾄﾞ (0,1)
  0 : 開放しない
  1 : 開放する

解説　　ﾄﾞｯﾌﾟﾗｰ測位に使用する RAM 領域をﾕｰｻﾞｰｱﾌﾟﾘに開放するかどうかを設定する

戻値　　0 : 設定成功
1 : 設定失敗（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足

get_kxs66

要約　　unsigned char get_kxs66(void)

解説　　ﾄﾞｯﾌﾟﾗｰ測位に使用するRAM領域をﾕｰｻﾞｰｱﾌﾟ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 : ﾓｰﾄﾞ (0,1)
0 : 端末電源ON時に、ﾕｰｻﾞｰｱﾌﾟﾘを起動しない
1 : 端末電源 ON 時に、ﾕｰｻﾞｰｱﾌﾟﾘを起動する

解説　　端末電源 ON 時に、ﾕｰｻﾞｰｱﾌﾟ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 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足

get_kxs67

要約　　unsigned char get_kxs67(void)

解説　　端末電源ON時に、ﾕｰｻﾞｰｱﾌﾟﾘを起動するかどうかを設定を獲得する

戻値　　ﾓｰﾄﾞ (0,1)
0 : 端末電源ON時に、ﾕｰｻﾞｰｱﾌﾟﾘを起動しない
1 : 端末電源 ON 時に、ﾕｰｻﾞｰｱﾌﾟﾘを起動する

補足

set_kxs68

要約　　int set_kxs68(unsigned char value)

unsigned char value : ﾀｲﾌﾟ (0 - 1)
0 : ﾒｲﾝｼｽﾃﾑで処理
1 : ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝで処理

解説　　RS232C 受信ﾃﾞ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（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足

get_kxs68

要約　　unsigned char get_kxs68(void)

解説　　RS232C受信ﾃﾞｰﾀの処理ﾙｰﾄを獲得する

戻値　　ﾀｲﾌﾟ (0 - 1).
0 : ﾒｲﾝｼｽﾃﾑで処理
1 : ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝで処理

補足

set_kxs69

要約　　int set_kxs69(unsigned char value)

unsigned char value : ﾀｲﾌﾟ (0 - 2)
0 : ﾒｲﾝｼｽﾃﾑで処理
1 : ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝで処理
2 : 両方で処理

解説　　GCC からの受信ﾃﾞ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 (ﾊﾟﾗﾒｰﾀｴﾗｰ)
2 : 排他ｴﾗｰ: KXS69(1or2) ／KXS68(1)

補足

get_kxs69

要約　　unsigned char get_kxs69(void)

解説　　GCCからの受信ﾃﾞｰﾀの処理ﾙｰﾄを獲得する

戻値　　ﾀｲﾌﾟ (0 - 2)
0 : ﾒｲﾝｼｽﾃﾑで処理
1 : ﾕｰｻﾞｰｱﾌﾟ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 : ﾓｰﾄﾞ
  0 : ﾃﾞﾊﾞｯｸﾞﾓｰﾄﾞOFF
  1-9 : ﾃﾞﾊﾞｯｸﾞﾓｰﾄﾞ ON

解説　　ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝのﾃﾞﾊﾞｯｸﾞﾓｰﾄﾞを設定する

戻値　　0 : 設定成功
1 : 設定失敗（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足

get_kxs70

要約　　unsigned char get_kxs70(void)

解説　　ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝのﾃﾞﾊﾞｯｸﾞﾓｰﾄﾞを獲得する

戻値　　ﾓｰﾄﾞ
  0 : ﾃﾞﾊﾞｯｸﾞﾓｰﾄﾞOFF
  1-9 : ﾃﾞﾊﾞｯｸﾞﾓｰﾄﾞ 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 からのｺﾏﾝﾄﾞﾘﾓｰﾄｾｯﾃｨﾝｸﾞでのｾｯﾄｱｯﾌﾟ ID を設定する

戻値 0 : 設定成功
1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足

get_kxs71

要約 char *get_kxs71(void)

解説 ﾘﾓｰﾄｾｯﾃｨﾝｸﾞでのｾｯﾄｱｯﾌﾟ IDを獲得する

戻値 ｾｯﾄｱｯﾌﾟ ID へのﾎﾟｲﾝﾀ

補足

set_kxs72

要約　　int set_kxs72(unsigned char value)

unsigned char value : ﾓｰﾄﾞ (0, 1)
0 : 応答ﾒｯｾｰｼﾞなし
1 : 応答ﾒｯｾｰｼﾞをﾎｽﾄに返す

解説　　ﾎｽﾄからのｺﾏﾝﾄﾞﾘﾓｰﾄｾｯﾃｨﾝｸﾞでその応答ﾒｯｾｰｼﾞを返すかどうかを設定する

戻値　　0 : 設定成功
1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足

get_kxs72

要約　　unsigned char get_kxs72(void)

解説　　GCCからのｺﾏﾝﾄﾞﾘﾓｰﾄｾｯﾃｨﾝｸﾞでその応答ﾒｯｾｰｼﾞを返すかどうかを獲得する

戻値　　ﾓｰﾄﾞ (0, 1)
0 : 応答ﾒｯｾｰｼﾞなし
1 : 応答ﾒｯｾｰｼﾞをﾎｽﾄに返す

補足

“set_kxs73” はｻﾎﾟｰﾄされてません。

“get_kxs73” はｻﾎﾟｰﾄされてません。

set_kxs74

要約　　int set_kxs74(unsigned char value)

unsigned char value : ﾓｰﾄﾞ
0 : 個別に送信
1 : まとめて送信

解説　　KXA あるいは KXB ｺﾏﾝﾄﾞによる無条件送信で位置情報を含む 2 つ以上のﾃﾞ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（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足　　0: I/O ﾎﾟｰﾄやﾊﾞｯﾃﾘｰ情報など即送信できるﾃﾞｰﾀは先に送信し、位置/速 度/方向情報など送信するまでに時間のかかるﾃﾞｰﾀは後で送信する。

1: 送信ﾃﾞｰﾀｻｲｽﾞを減らすため、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 ﾎﾟｰﾄやﾊﾞ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 ﾎﾟｰﾄやﾊﾞ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ｺﾏﾝﾄﾞによる無条件送信で位置情報を含む2つ以上のﾃﾞｰﾀを送信する場合の送信方法を獲得する

戻値　　ﾓｰﾄﾞ
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 : ﾓｰﾄﾞ
  0 : 自動でﾒｯｾｰｼﾞ/ｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑに変換する
  1 : 自動変換しない

解説　　KXA/B ｺﾏﾝﾄﾞで生成されるﾒｯｾｰｼﾞを衛星がｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑ衛星ならｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑに自動的に変換するﾓｰﾄﾞを設定する

戻値　　0 : 設定成功
1 : 設定失敗（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足　　.

get_kxs75

要約　　unsigned char get_kxs75(void)

解説　　KXA/Bｺﾏﾝﾄﾞで生成されるﾒｯｾｰｼﾞを衛星がｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑ衛星ならｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑに自動的に変換するﾓｰﾄﾞを獲得する

戻値　　ﾓｰﾄﾞ
  0 : 自動でﾒｯｾｰｼﾞ/ｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑに変換する
  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（ﾊﾟﾗﾒｰﾀｴﾗｰ）

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 : ﾓｰﾄﾞ
0 : 通常（O/Rｱﾄﾞﾚｽ、ｻﾌﾞｼﾞｪｸﾄ、ﾒｯｾｰｼﾞ本体すべてを送信）
1 : ﾒｯｾｰｼﾞ本体のみを DTE に送信

解説　　ﾊﾞｲﾄﾓｰﾄﾞ時、ﾎｽﾄからの受信ﾃﾞｰﾀのﾒｯｾｰｼﾞ本体のみをDTEに送信するﾓｰﾄﾞに設定する

戻値　　0 : 設定成功
1 : 設定失敗（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足

get_kxs77

要約　　unsigned char get_kxs77(void)

解説　　ﾊﾞｲﾄﾓｰﾄﾞ時、ﾎｽﾄからの受信ﾃﾞｰﾀのﾒｯｾｰｼﾞ本体のみをDTEに送信するﾓｰﾄﾞを獲得する

戻値　　ﾓｰﾄﾞ
0 : 通常（O/Rｱﾄﾞﾚｽ、ｻﾌﾞｼﾞｪｸﾄ、ﾒｯｾｰｼﾞ本体すべてを送信）
1 : ﾒｯｾｰｼﾞ本体のみを DTE に送信

補足

set_kxs78

要約　　int set_kxs78(unsigned char value)

unsigned char value : ﾓｰﾄﾞ
0 : 何も送信しない
1 : ｲﾝﾊﾞｳﾝﾄﾞﾊﾞｯﾌｧ満杯ﾒｯｾｰｼﾞを送信する

解説　　ﾊﾞｲﾄﾓｰﾄﾞ時、ｲﾝﾊﾞｳﾝﾄﾞｷｭｰが満杯で指定ﾊﾞｲﾄ数受信できない時にﾊﾞｯﾌｧﾌﾙﾒｯｾｰｼﾞを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（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足

get_kxs78

要約　　unsigned char get_kxs78(void)

解説　　ﾊﾞｲﾄﾓｰﾄﾞ時、ｲﾝﾊﾞｳﾝﾄﾞｷｭｰが満杯で指定ﾊﾞｲﾄ数受信できない時にﾊﾞｯﾌｧﾌﾙﾒｯｾｰｼﾞを DTE に送信する設定を獲得する

戻値　　ﾓｰﾄﾞ
0 : 何も送信しない
1 : ｲﾝﾊﾞｳﾝﾄﾞﾊﾞｯﾌｧ満杯ﾒｯｾｰｼﾞを送信する

補足

set_kxs79

要約 int set_kxs79(int time)

int range: KXB 検知時間 （5 - 20）[分]

解説 KXBｺﾏﾝﾄﾞの検知時間を設定する

戻値 0 : 設定成功
1 : 設定失敗（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足

get_kxs79

要約 int get_kxs79(void)

解説 KXB ｺﾏﾝﾄﾞ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 : ｱｳﾄﾊﾞｳﾝﾄﾞｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑﾊﾟｹｯﾄのﾌｫｰﾏｯﾄ
0 : ﾉｰﾏﾙｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑ
1 : (使用不可)

解説　　ｱｳﾄﾊﾞｳﾝﾄﾞｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑﾊﾟ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（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足　　(本ｺﾏﾝﾄﾞはｴﾝﾊﾝｽﾄｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑへの将来対応でしたが、ﾈｯﾄﾜｰｸ側計画変更により使用できません)

get_kxs80

要約　　int get_kxs80(void)

解説　　ｱｳﾄﾊﾞｳﾝﾄﾞｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑﾊﾟｹｯﾄのﾌｫｰﾏｯﾄを獲得する

戻値　　ｱｳﾄﾊﾞｳﾝﾄﾞｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑﾊﾟｹｯﾄのﾌｫｰﾏｯﾄ
0 : ﾉｰﾏﾙｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑ
1 : (使用不可)

補足

set_kxs81

要約　　int set_kxs81(unsigned long int time)

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

解説　　ｱｳﾄﾊﾞｳﾝﾄﾞﾒｯｾｰｼﾞの重複受信防止のｶﾞｰﾄﾞﾀｲﾏｰを設定する

戻値　　0: 設定成功
1: 設定失敗（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足

get_kxs81

要約　　unsigned long int get_kxs81(void)

解説　　ｱｳﾄﾊﾞｳﾝﾄﾞﾒｯｾｰｼﾞの重複受信防止のｶﾞｰﾄﾞ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　　ｱｳﾄﾊﾞｳﾝﾄﾞｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑの重複受信防止のｶﾞｰﾄﾞﾀｲﾏｰを設定する

戻値　　0: 設定成功
1: 設定失敗（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足

get_kxs82

要約　　int get_kxs82(void)

解説　　ｱｳﾄﾊﾞｳﾝﾄﾞｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑの重複受信防止のｶﾞｰﾄﾞ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　　ﾃﾞｼﾞﾀﾙ出力ﾎﾟｰﾄのﾃﾞﾌｫﾙﾄ値を設定する

戻値　　0: 設定成功
1: 設定失敗（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足

get_kxs83

要約　　int get_kxs83(void)

解説　　ﾃﾞｼﾞﾀﾙ出力ﾎﾟｰﾄのﾃﾞ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 moe: roaming mode
　　0: ﾛｰﾐﾝｸﾞ OFF(KXS01 で設定した GCC とのみ通信可)
　　1: ﾛｰﾐﾝｸﾞ ON (KXS85 で設定した GCC と通信可)
　　2: ﾛｰﾐﾝｸﾞ ON (全 GCC と通信可)

解説　　ﾛｰﾐﾝｸﾞﾓｰﾄﾞを設定する

戻値　　0: 設定成功
　　1: 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足

get_kxs84

要約　　int get_kxs84(void)

解説　　ﾛｰﾐﾝｸﾞﾓｰﾄﾞの設定値を獲得する

戻値　　roming mode
　　0: ﾛｰﾐﾝｸﾞ OFF(KXS01 で設定した GCC とのみ通信可)
　　1: ﾛｰﾐﾝｸﾞ ON (KXS85 で設定した GCC と通信可)
　　2: ﾛｰﾐﾝｸﾞ 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　　ﾛｰﾐﾝｸﾞ対象GCCを設定する

戻値　　0: 設定成功
　　1: 設定失敗（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足

get_kxs85

要約　　unsigned char get_kxs85(int ind)
　　int ind: block No(0 - 11)

解説　　ﾛｰﾐﾝｸﾞ対象 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 : 日曜 - 土曜
 ‘0’(0x30) : 動作しない
 ‘1’(0x31) : 動作する

解説　　KXBｺﾏﾝﾄﾞ動作の曜日指定を行う

戻値　　0: 設定成功
 1: 設定失敗（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足

get_kxs86

要約　　char *get_kxs86(void)

解説　　KXB ｺﾏﾝﾄﾞ動作の曜日指定値を獲得する

戻値　　pointer of day information.
<day information>
0 - 6 : 日曜 - 土曜
 ‘0’(0x30) : 動作しない
 ‘1’(0x31) : 動作する

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（１～１０）

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（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足　　set_pdop_th()と同じ処理を行います。
PDOP を小さくするほど精度は向上しますが、測位時間が長くなります。

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（１～１０）

補足　　get_pdop_th()と同じ処理を行います。

set_kxs88

要約　　int set_kxs88(int mode)

int mode : ﾘｾｯﾄ方法
0 : ｿﾌﾄｳｪｱまたはﾊｰﾄﾞｳｪｱﾘｾｯﾄいずれも行わない
1 : ｿﾌﾄｳｪｱﾘｾｯﾄのみ実施
2 : ﾊｰﾄﾞｳｪｱﾘｾｯﾄのみ実施
3 : ｿﾌﾄｳｪｱ、ﾊｰﾄﾞｳｪｱﾘｾｯﾄ双方実施

解説　　自動ﾘｾｯﾄﾓｰﾄﾞを設定する。

戻値　　0: 設定成功
1: 設定失敗（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足　　ｿﾌﾄｳｪｱﾘｾｯﾄ
24 時間以上連続動作で午前 2 時 27 分 30 秒(ローカル時間)を経過した時点で実施。

ﾊｰﾄﾞｳｪｱﾘｾｯﾄ
上記ｿﾌﾄｳｪｱﾘｾｯﾄ条件かつ24時間以上衛星をまったく受信していない場合、短時間のｽﾘｰﾌﾟをかけ、ﾊｰﾄﾞ資源を含めたﾘｾｯﾄを実施します。この場合ﾎﾟｰﾄの値が一時的に変化することがあります。

get_kxs88

要約　　int get_kxs88(void)

解説　　自動ﾘｾｯﾄ方法を獲得する。

戻値　　ﾘｾｯﾄ方法
0 : ｿﾌﾄｳｪｱまたはﾊｰﾄﾞｳｪｱﾘｾｯﾄいずれも行わない
1 : ｿﾌﾄｳｪｱﾘｾｯﾄのみ実施
2 : ﾊｰﾄﾞｳｪｱﾘｾｯﾄのみ実施
3 : ｿﾌﾄｳｪｱ、ﾊｰﾄﾞｳｪｱﾘｾｯﾄ双方実施

補足

set_kxd01

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

unsigned char m: ﾎﾟｰﾄ (0:出力ﾎﾟｰﾄ1, 1:出力ﾎﾟｰﾄ2)
unsigned char hi_low: ﾚﾍﾞﾙ(0:LOW, 1:HI)

解説　　ﾃﾞｼﾞﾀﾙﾎﾟｰﾄ出力を設定する

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

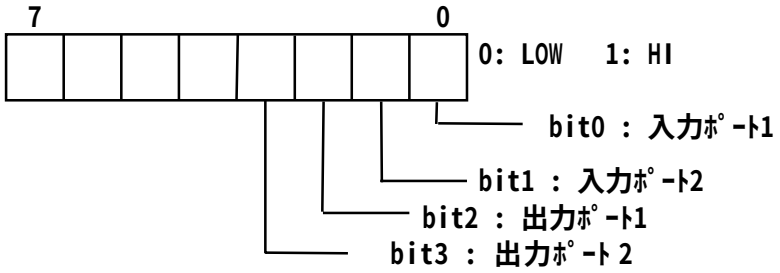
補足

get_kxd01

要約　　unsigned char get_kxd01(void)

解説　　ﾃﾞｼﾞﾀﾙﾎﾟｰﾄ出力設定を獲得する

戻値



補足

set_kxd02 はｻﾎﾟｰﾄされてません。

get_kxd02

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

unsigned char ch_no : ｱﾅﾛｸﾞﾎﾟｰﾄ番号 (0 - 7)
0: RSSI
1: ｱﾅﾛｸﾞﾎﾟｰﾄ 1
2: ｱﾅﾛｸﾞﾎﾟｰﾄ 2
3 - 7: 予備

unsigned char *ch　: ｱﾅﾛｸﾞﾃﾞｰﾀ(0 - 255)

解説　　ｱﾅﾛｸﾞ入力値を獲得する

戻値　　0: 設定成功
1: ｱﾅﾛｸﾞﾎﾟｰﾄ番号入力ｴﾗｰ

補足

set_kxp01

要約　　int set_kxp01(unsigned char value)

unsigned char value : ﾓｰﾄﾞ (0 - 2)
  0 : なし
  1 : 入力ﾎﾟｰﾄの状態を出力ﾎﾟｰﾄにそのまま出力
  2 : 入力ﾎﾟｰﾄの状態を反転させて出力ﾎﾟｰﾄに出力

解説　　入力ﾎﾟｰﾄと出力ﾎﾟｰﾄのﾘﾝｸを設定する

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

補足

get_kxp01

要約　　unsigned char get_kxp01(void)

解説　　入力ﾎﾟｰﾄと出力ﾎﾟｰﾄのﾘﾝｸ設定を獲得する

戻値　　ﾓｰﾄﾞ (0 - 2)
  0 : なし
  1 : 入力ﾎﾟｰﾄの状態を出力ﾎﾟｰﾄにそのまま出力
  2 : 入力ﾎﾟｰﾄの状態を反転させて出力ﾎﾟｰﾄに出力

補足

set_kxm01

要約　　int set_kxs01(unsigned char *str)

　　char *str : 固定ﾒｯｾｰｼﾞへのﾎﾟｲﾝﾀ (最大 200 ﾊﾞｲﾄ)

解説　　固定ﾒｯｾｰｼﾞを設定する

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

補足　　.

get_kxm01

要約　　char *get_kxm01(void)

解説　　固定ﾒｯｾｰｼﾞを獲得する

戻値　　固定ﾒｯｾｰｼﾞへのﾎﾟｲﾝﾀ (最大 200 ﾊﾞｲﾄ)

補足

set_kxa01

要約　　int set_kxa01(KXA01_CMND *pkxa01_cmd, int n)

KXA01_CMND *pkxa01_cmd
　int lh: ﾀｲﾑｿﾞｰﾝ (-11 - +12) [時間]
　int lM: ﾀｲﾑｿﾞｰﾝ (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ｲﾝﾃﾞｨｹｰﾀ (1 - 8)
　char d: 送信ﾃﾞｰﾀ

7	6	5	4	3	2	1	0

　bit0: 測位情報
　bit1: ﾃﾞｼﾞﾀﾙﾎﾟｰﾄ情報
　bit2: 固定ﾒｯｾｰｼﾞ
　bit3: GCCへのﾎﾟｰﾘﾝｸﾞ
　bit4: DTEへのﾎﾟｰﾘﾝｸﾞ
　bit5: 予備
　bit6: ﾕｰｻﾞｰｱﾌﾟﾘ起動
　bit7: ｱﾅﾛｸﾞﾎﾟｰﾄ情報

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 (ﾊﾟﾗﾒｰﾀｴﾗｰ)
　　2 : 排他ｴﾗｰ

補足

get_kxa01

要約　　int get_kxa01(KXA01_CMND *pkxa01_cmnd, int n)

KXA01_CMND *pkxa01_cmd
int n: 時刻番号 (1 - 6)

解説　　時刻指定送信(KXA01ｺﾏﾝﾄﾞ)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: ﾀｲﾑｿﾞｰﾝ (-11 - +12) [時間]
 int lM: ﾀｲﾑｿﾞｰﾝ (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: ｲﾝﾀｰﾊﾞﾙ (1 - 10080)[分]
 int a: 0/Rｲﾝﾃﾞｨｹｰﾀ (1 - 3)
 char d: 送信ﾃﾞｰﾀ

7	6	5	4	3	2	1	0

bit0: 測位情報
bit1: ﾃﾞｼﾞﾀﾙﾎﾟｰﾄ情報
bit2: 固定ﾒｯｾｰｼﾞ
bit3: GCCへのﾎﾟｰﾘﾝｸﾞ
bit4: DTEへのﾎﾟｰﾘﾝｸﾞ
bit5: 予備
bit6: ﾕｰｻﾞｰｱﾌﾟﾘ起動
bit7: ｱﾅﾛｸﾞﾎﾟｰﾄ情報

解説　　ｲﾝﾀｰﾊﾞﾙ送信を設定する

戻値　　0 : 設定成功
1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)
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　　ｲﾝﾀｰﾊﾞﾙ送信 (KXA02ｺﾏﾝﾄﾞ) 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: ｲﾝﾀｰﾊﾞﾙ (1 - 1440)[分]
  int a: 0/Rｲﾝﾃﾞｨｹｰﾀ (1 -8)
  char d: 送信ﾃﾞｰﾀ

7	6	5	4	3	2	1	0

bit0: 測位情報
bit1: ﾃﾞｼﾞﾀﾙﾎﾟｰﾄ情報
bit2: 固定ﾒｯｾｰｼﾞ
bit3: GCCへのﾎﾟｰﾘﾝｸﾞ
bit4: DTEへのﾎﾟｰﾘﾝｸﾞ
bit5: 予備
bit6: ﾕｰｻﾞｰｱﾌﾟﾘ起動
bit7: ｱﾅﾛｸﾞﾎﾟｰﾄ情報

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 (ﾊﾟﾗﾒｰﾀｴﾗｰ)
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ｺﾏﾝﾄﾞ) 設定を獲得する

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: ﾄﾘｶﾞｰ (0:OFF, 1:LOW -> HI, 2:HI->LOW, 3:HI<->LOW)
 int a: 0/Rｲﾝﾃﾞｨｹｰﾀ (1 - 8)
 char d: 送信ﾃﾞｰﾀ

7	6	5	4	3	2	1	0

bit0: 測位情報
bit1: ﾃﾞｼﾞﾀﾙﾎﾟｰﾄ情報
bit2: 固定ﾒｯｾｰｼﾞ
bit3: GCCへのﾎﾟｰﾘﾝｸﾞ
bit4: DTEへのﾎﾟｰﾘﾝｸﾞ
bit5: 予備
bit6: ﾕｰｻﾞｰｱﾌﾟﾘ起動
bit7: ｱﾅﾛｸﾞﾎﾟｰﾄ情報

int p: ﾃﾞｼﾞﾀﾙ入力ﾎﾟｰﾄ番号(0, 1)

解説　　入力ﾎﾟｰﾄ変化で送信を設定する

戻値　　0 : 設定成功
1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)
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 : ﾃﾞｼﾞﾀﾙ入力ﾎﾟｰﾄ番号 (0, 1)

解説　　入力ﾎﾟｰﾄ変化で送信 (KXA05ｺﾏﾝﾄﾞ) 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: ﾀｲﾑｿﾞｰﾝ (-11 - +12) [時間]
int lM: ﾀｲﾑｿﾞｰﾝ (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: 検知ｺｰﾄﾞ
int a: 0/Rｲﾝﾃﾞｨｹｰﾀ (1 - 8)
char d: 送信ﾃﾞｰﾀ

7	6	5	4	3	2	1	0

bit0: 測位情報
bit1: ﾃﾞｼﾞﾀﾙﾎﾟｰﾄ情報
bit2: 固定ﾒｯｾｰｼﾞ
bit3: GCCへのﾎﾟｰﾘﾝｸﾞ
bit4: DTEへのﾎﾟｰﾘﾝｸﾞ
bit5: 予備
bit6: ﾕｰｻﾞｰｱﾌﾟﾘ起動
bit7: ｱﾅﾛｸﾞﾎﾟｰﾄ情報

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 (ﾊﾟﾗﾒｰﾀｴﾗｰ)
2 : 排他ｴﾗｰ

補足

get_kxb01

要約　　int get_kxb01(KXB01_CMND *pkxb01_cmnd, int n)

KXB01_CMND *pkxb01_cmnd
int n: 時刻番号 (1 - 6)

解説　　時刻指定送信 (KXB01ｺﾏﾝﾄﾞ) 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: ﾀｲﾑｿﾞｰﾝ (-11 - +12) [時間]
- int lM: ﾀｲﾑｿﾞｰﾝ (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: ｲﾝﾀｰﾊﾞﾙ (1 - 10080)[分]
- int c: 検知ｺｰﾄﾞ
- int a: O/Rｲﾝﾃﾞｨｹｰﾀ (1 - 8)
- char d: 送信ﾃﾞｰﾀ

7	6	5	4	3	2	1	0

bit0: 測位情報
bit1: ﾃﾞｼﾞﾀﾙﾎﾟｰﾄ情報
bit2: 固定ﾒｯｾｰｼﾞ
bit3: GCCへのﾎﾟｰﾘﾝｸﾞ
bit4: DTEへのﾎﾟｰﾘﾝｸﾞ
bit5: 予備
bit6: ﾕｰｻﾞｰｱﾌﾟﾘ起動
bit7: ｱﾅﾛｸﾞﾎﾟｰﾄ情報

解説　　ｲﾝﾀｰﾊﾞﾙ送信を設定する

戻値　　0 : 設定成功
1 : 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)
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　　ｲﾝﾀｰﾊﾞﾙ送信 (KXB02ｺﾏﾝﾄﾞ) 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: ｲﾝﾀｰﾊﾞﾙ (1 - 1440)[分]
int c: 検知ｺｰﾄﾞ
int a: O/Rｲﾝﾃﾞｨｹｰﾀ (1 - 8)
char d: 送信ﾃﾞｰﾀ

7	6	5	4	3	2	1	0

bit0: 測位情報
bit1: ﾃﾞｼﾞﾀﾙﾎﾟｰﾄ情報
bit2: 固定ﾒｯｾｰｼﾞ
bit3: GCCへのﾎﾟｰﾘﾝｸﾞ
bit4: DTEへのﾎﾟｰﾘﾝｸﾞ
bit5: 予備
bit6: ﾕｰｻﾞｰｱﾌﾟﾘ起動
bit7: ｱﾅﾛｸﾞﾎﾟｰﾄ情報

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 (ﾊﾟﾗﾒｰﾀｴﾗｰ)
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ｺﾏﾝﾄﾞ) 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 - 03，05 ｺﾏﾝﾄﾞの設定を解除する

戻値　　なし

補足

set_kxb00

要約　　void set_kxb00(void)

解説　　KXB01 - 03，05 ｺﾏﾝﾄﾞの設定を解除する

戻値　　なし

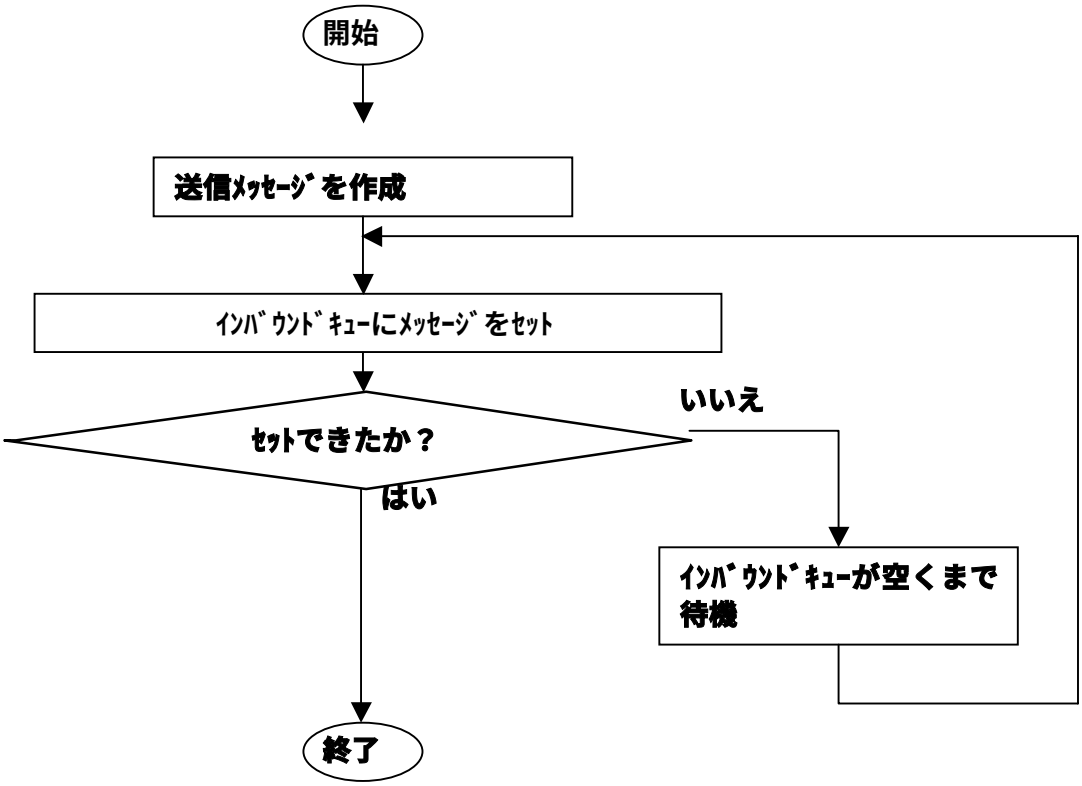
補足

7. ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ ﾋﾝﾄ

7.1. GCCへのﾒｯｾｰｼﾞ送信

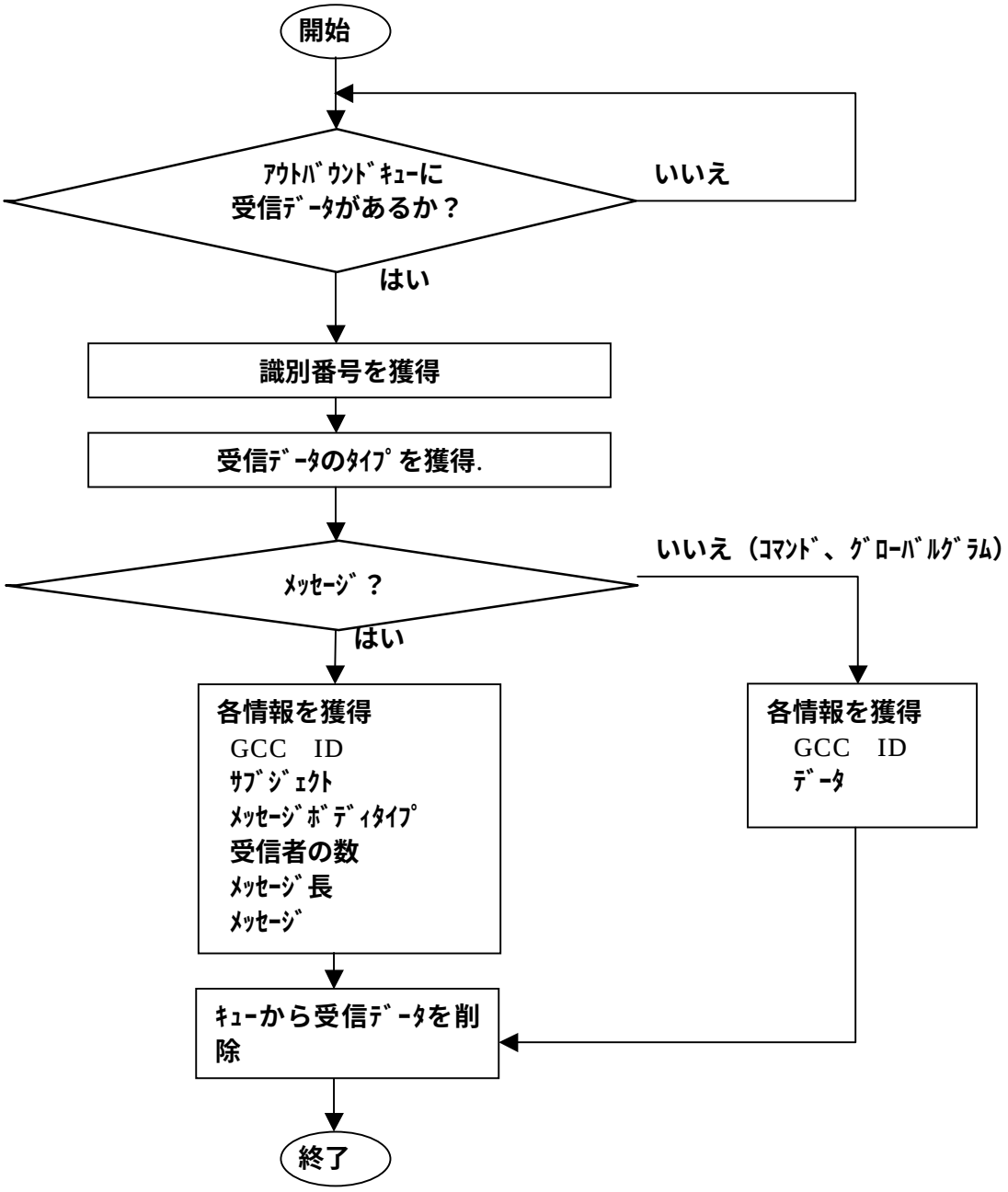
1. 送信ﾒｯｾｰｼﾞを作成する
2. ﾒｯｾｰｼﾞをｲﾝﾊﾞｳﾝﾄﾞｷｭｰにｾｯﾄする為に、“ req_send_ib_message() ” をｺｰﾙする。
3. ﾒｯｾｰｼﾞがｲﾝﾊﾞｳﾝﾄﾞｷｭｰにｾｯﾄされたかをﾁｪｯｸする。

   ｾｯﾄできなかった場合は、ｲﾝﾊﾞｳﾝﾄﾞｷｭｰから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ﾊﾞｳﾝﾄﾞｷｭｰにｾｯﾄされたﾒｯｾｰｼﾞが自動的にGCCへ送信される。



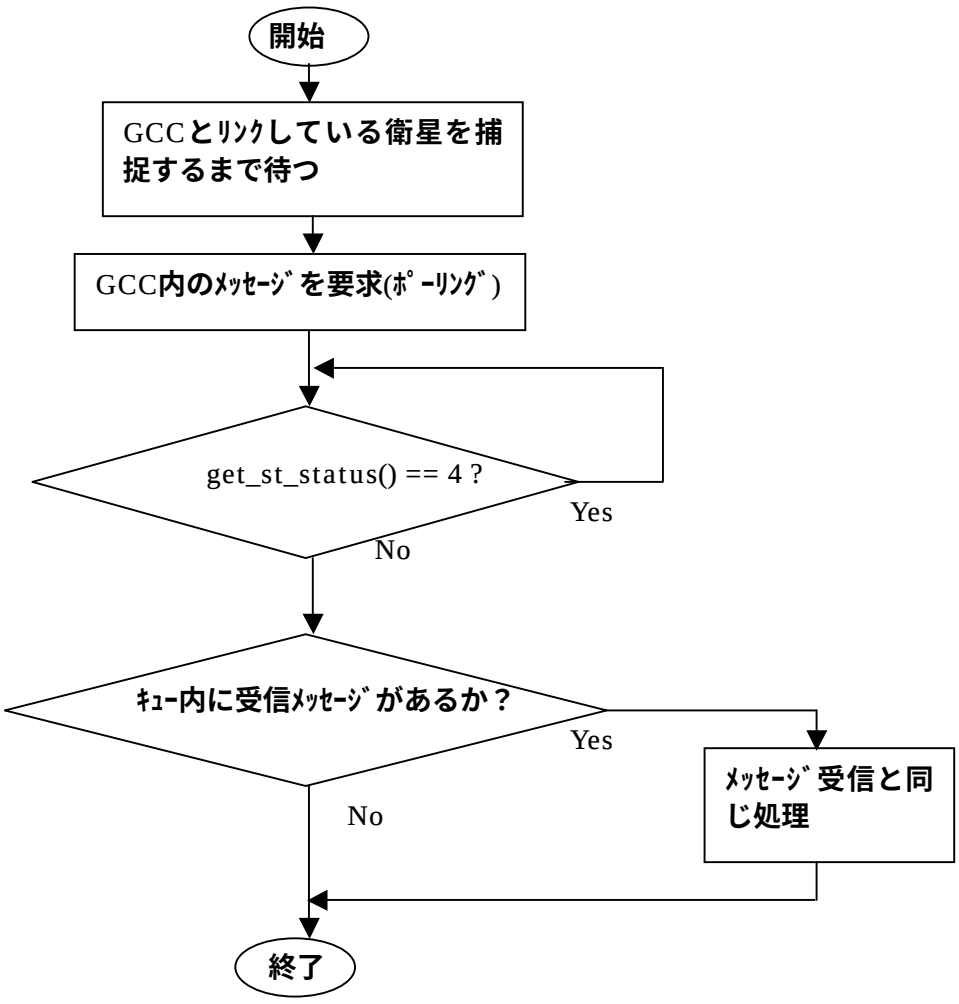
7.2. GCCからのﾒｯｾｰｼﾞ受信

1. “get_unget_ob_id_for_user”をｺｰﾙして、ｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰに受信ﾃﾞｰﾀがあるかをﾁｪｯｸする。
2. もし受信ﾃﾞｰﾀがあれば、そのﾀｲﾌﾟ（ﾒｯｾｰｼﾞ/ｺﾏﾝﾄﾞ）に応じた処理をする。
3. ｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰからﾃﾞｰﾀを消す



7.3. GCC内のﾒｯｾｰｼﾞを取り出す

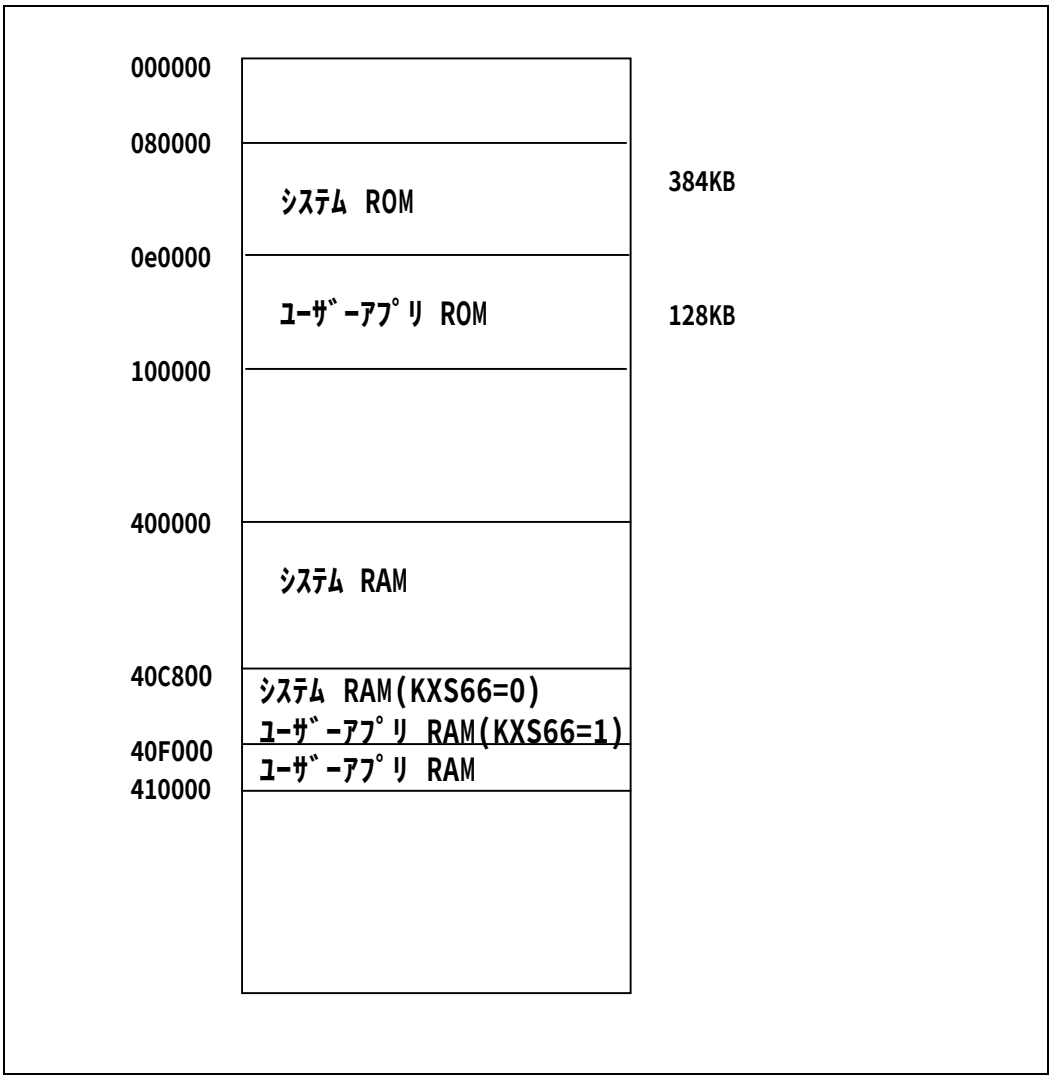
1. GCCとﾘﾝｸしている衛星を捕捉しているか確認する。
2. “all_msg_polling”をｺｰﾙして、GCC内のﾒｯｾｰｼﾞを要求する。
3. GCCからの応答を待つ。 この時SCは、応答待ちしている間“get_st_status()”の戻り値は４になる。
4. ﾒｯｾｰｼﾞがあれば、GCCから送信され、そのﾒｯｾｰｼﾞは自動的にｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰに格納される。
5. “get_st_status()”の戻り値が0に変わるのをまって“get_unget_ob_id_for_user”をｺｰﾙし、ｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰに受信ﾃﾞｰﾀがあるかをﾁｪｯｸする。
6. もし受信ﾃﾞｰﾀがあれば、そのﾀｲﾌﾟ（ﾒｯｾｰｼﾞ/ｺﾏﾝﾄﾞ）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とﾕｰｻﾞｰｱﾌﾟﾘ用のROM/RAMがある。
ﾕｰｻﾞｰｱﾌﾟﾘ用の ROM ｻｲｽﾞは 128KB (0xe0000 - 0xfffff)、RAM ｻｲｽﾞは KXS66 ｺﾏﾝﾄﾞで選択可能であり、4KB(0x40f000 - 0x40ffff)、あるいは 14KB(0x40c800 - 0x40ffff)である。
但し、RAM ｻｲｽﾞが 16KB の時にはﾄﾞｯﾌﾟﾗｰ測位は使用できなくなる。

このﾕｰｻﾞｰｱﾌﾟﾘ用の RAM 領域は、ﾕｰｻﾞｰｱﾌﾟﾘﾌﾟﾛｸﾞﾗﾑをｲﾝｽﾄｰﾙする時に 0 ｸﾘｱされる。



従って、ﾘﾝｸする際には以下のように設定しなくてはいけない。

```
T_TEXT_START=e0000   ←  ｽﾀｰﾄｱｯﾌﾟ関数の先頭ｱﾄﾞﾚｽ
T_TEXT=e0010         ←  ﾕｰｻﾞｰｱﾌﾟﾘﾌﾟﾛｸﾞﾗﾑの先頭ｱﾄﾞﾚｽ
T_GBSS_USER=40f000   ←  ﾕｰｻﾞｰｱﾌﾟﾘ用 RAM の先頭ｱﾄﾞﾚｽ
                            *4KB の時、 40f000
                            16KB の時、 40c800
```

ﾕｰｻﾞｰｱﾌﾟﾘのｽﾀｯｸ領域は、ｼｽﾃﾑRAM上に割り当てられおり、ｻｲｽﾞは1500ﾊﾞｲﾄで
ある。ｽﾀｯｸ領域の先頭ｱﾄﾞﾚｽは “ stack ” で確認できる。

ﾕｰｻﾞｰｱﾌﾟﾘﾌﾟﾛｸﾞﾗﾑ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ﾊﾞｰﾌﾛｰをおこす可能性があるので、大きいｻｲｽﾞの変数（例えば受信ﾊﾞｯﾌｧ等）は、ｸﾞﾛｰﾊﾞﾙ変数として宣言したほうが好ましい。

9. ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｸﾞﾗﾑのﾃﾞﾊﾞｯｸﾞ方法

ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｸﾞﾗﾑ作成後、ﾕｰｻﾞｰはそのﾌﾟﾛｸﾞﾗﾑをﾃﾞﾊﾞｯｸﾞすることができる。 その為には、端末とﾊﾟｿｺﾝをRS232Cｹｰﾌﾞﾙで接続して、ﾊﾟｿｺﾝで通信ｿﾌﾄ（例えば “ﾊｲﾊﾟｰﾀｰﾐﾅﾙ”)を起動する必要があり、そうすれば端末からいろいろなﾃﾞﾊﾞｯｸﾞ情報を獲得できる。

ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｸﾞﾗﾑのﾃﾞﾊﾞｯｸﾞ機能として以下の機能が用意されている。

1. ﾒﾓﾘｰﾀﾞﾝﾌﾟ
2. ﾒﾓﾘｰ書き込み
3. ｽﾀｯｸ開始ｱﾄﾞﾚｽの獲得
4. ﾌﾞﾚｰｸﾎﾟｲﾝﾄ
5. ｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰへのﾒｯｾｰｼﾞの登録

これらのﾃﾞﾊﾞｯｸﾞ機能はﾃﾞﾊﾞｯｸﾞﾓｰﾄﾞがONの時に有効となる。ﾃﾞﾊﾞｯｸﾞﾓｰﾄﾞをONにする為にはKXS70ｺﾏﾝﾄﾞを使用する。.
ﾃﾞﾊﾞｯｸﾞﾓｰﾄﾞON時にﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｸﾞﾗﾑが起動されると、LEDが500 msec点灯する。これによりﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｸﾞﾗﾑが起動されたかどうかの確認ができる。

9.1. メモリーダンプ

ﾒﾓﾘｰ内容を16進で表示する。

入力ｺﾏﾝﾄﾞ：mdump <ｱﾄﾞﾚｽ>,<範囲>

<例>

```
      <CTRL> + “KXORB”      ：ｺﾏﾝﾄﾞﾓｰﾄﾞに入る
  >   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領域にﾃﾞｰﾀを書き込む。

入力ｺﾏﾝﾄﾞ：mem <ｱﾄﾞﾚｽ>,<ﾃﾞｰﾀ>

<例>

<CTRL> + “KXORB” ：ｺﾏﾝﾄﾞﾓｰﾄﾞに入る
➢ **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ﾄﾞﾚｽを獲得する。ﾕｰｻﾞｰｱﾌﾟﾘのｽﾀｯｸ領域は、ｼｽﾃﾑRAM上に割り当てられおり、ｻｲｽﾞは1500ﾊﾞｲﾄである。.

入力ｺﾏﾝﾄﾞ：stack

<例>

➢ **stack**
4020d8

9.4. ブレークポイント

ﾌﾞﾚｰｸﾎﾟｲﾝﾄ機能によりﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｸﾞ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、事前にﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｸﾞﾗﾑのｿｰｽｺｰﾄﾞに “break_point” 関数を挿入しておく必要がある。このﾌﾞﾚｰｸﾎﾟｲﾝﾄは、ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｸﾞﾗﾑをﾛｰﾄﾞした後にDTEからの “break” ｺﾏﾝﾄﾞにより、有効/無効に設定できる。

<ﾌﾞﾚｰｸﾎﾟｲﾝﾄ関数>
ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｸﾞﾗﾑのｿｰｽｺｰﾄﾞに挿入。
KME 関数：break_point(bp)
bp : ﾌﾞﾚｰｸﾎﾟｲﾝﾄ番号 (1 - 256)

<ﾌﾞﾚｰｸﾎﾟｲﾝﾄの設定>
入力ｺﾏﾝﾄﾞ： “break”
ﾌﾞﾚｰｸﾎﾟｲﾝﾄを有効にする：break <bp>,0
ﾌﾞﾚｰｸﾎﾟｲﾝﾄを無効にする：break <bp>,1

すべてのﾌﾞﾚｰｸﾎﾟｲﾝﾄを有効にする：break 0,0
すべてのﾌﾞﾚｰｸﾎﾟｲﾝﾄを無効にする：break 0,1

[note]
ﾌﾞﾚｰｸした後に、再開させる為には以下のｺﾏﾝﾄﾞを入力する。
入力ｺﾏﾝﾄﾞ：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からﾒｯｾｰｼﾞを受信したように、ｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰへﾒｯｾｰｼﾞをｾｯﾄすることができる。
ﾒｯｾｰｼﾞ本体のほかに、ｻﾌﾞｼﾞｪｸﾄ、O/Rｲﾝﾃﾞｨｹｰﾀ、O/Rｱﾄﾞﾚｽが設定できる。

<入力ｺﾏﾝﾄﾞ>
obreg [ﾚｼﾋﾟｴﾝﾄ1],[ﾚｼﾋﾟｴﾝﾄ2], [ｻﾌﾞｼﾞｪｸﾄ],<T or H>[ﾒｯｾｰｼﾞ]<CR><LF>
[ﾚｼﾋﾟｴﾝﾄ1,2] ：O/Rｲﾝﾃﾞｨｹｰﾀ、O/Rｱﾄﾞﾚｽ。O/Rｲﾝﾃﾞｨｹｰﾀの時には先頭に ‘@’を付加する。
[ｻﾌﾞｼﾞｪｸﾄ] ：最大10文字
<T or H> ：[ﾒｯｾｰｼﾞ].の種類。 “<T>” ：ﾃｷｽﾄ、“<H>”：ﾊﾞｲﾅﾘｰ
[ﾒｯｾｰｼﾞ] ：ﾒｯｾｰｼﾞ本体

* “obreg”ｺﾏﾝﾄﾞは、最大256ﾊﾞｲﾄです。

<例>

- ｻﾌﾞｼﾞｪｸﾄ：TEST1、
  ﾚｼﾋﾟｴﾝﾄ：O/Rｲﾝﾃﾞｨｹｰﾀ1、
  ﾒｯｾｰｼﾞ ： ”12345”

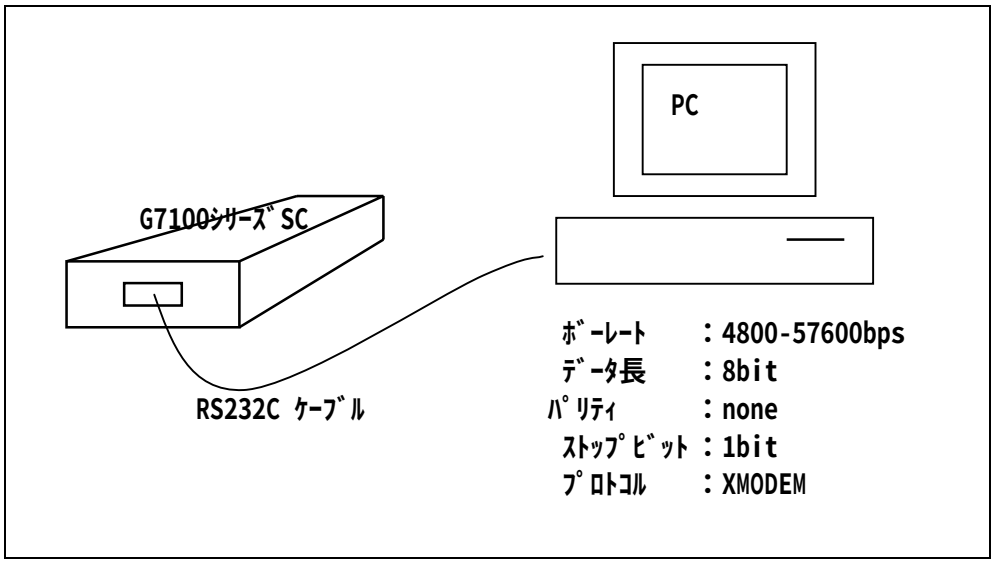
```
obreg @1,,TEST1,<T>12345
```

- ｻﾌﾞｼﾞｪｸﾄ：TEST2、
  ﾚｼﾋﾟｴﾝﾄ：O/Rｲﾝﾃﾞｨｹｰﾀ1、O/Rｱﾄﾞﾚｽ：“SUM”、
  ﾒｯｾｰｼﾞ ： ”0x12,0x34,0x56,0xab”

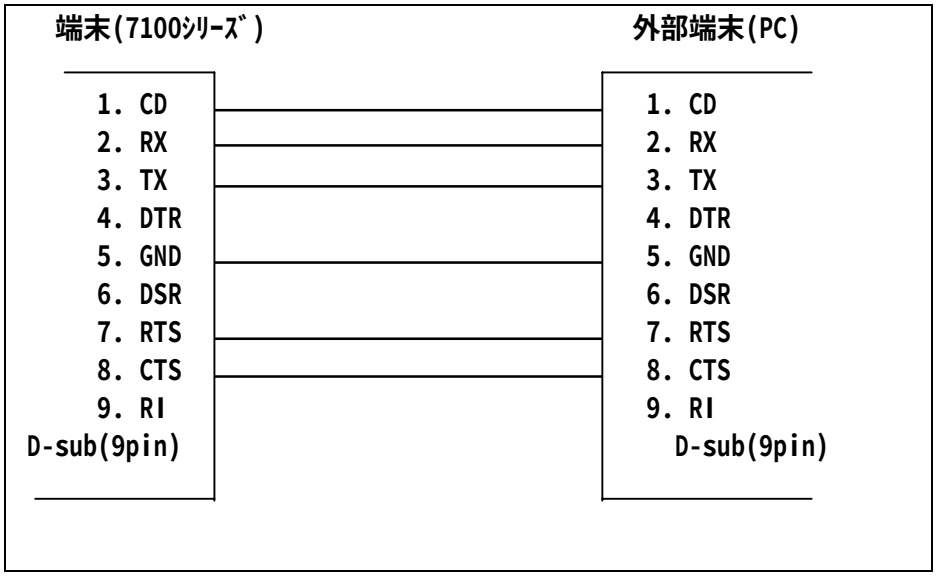
```
obreg SUM,@1,TEST2,<H>123456ab
```

10. ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｸﾞﾗﾑのｲﾝｽﾄｰﾙ方法

ﾊﾟｿｺﾝの通信ｿﾌﾄ（例えばﾊｲﾊﾟｰﾀｰﾐﾅﾙ）を使用してﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｸﾞﾗﾑをﾕｰｻﾞｰ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．端末とﾊﾟｿｺﾝをRS232Cｹｰﾌﾞﾙで接続し、ﾊｲﾊﾟｰﾀｰﾐﾅﾙを起動する
２．CTRL +KXORBと入力してｺﾏﾝﾄﾞﾓｰﾄﾞに入る
３．CTRL + UALDRと入力してｲﾝｽﾄｰﾙﾓｰﾄﾞに入る
PCに以下のﾒｯｾｰｼﾞが表示される
“*** 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)を入力してﾎﾞｰﾚｰﾄを選択する
PCに以下のﾒｯｾｰｼﾞが表示される
“Match the modem bps to this, then reconnect.”
“Start Sending User Application Software within 10sec.”
５．PCからﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｸﾞﾗﾑを送信する
ｲﾝｽﾄｰﾙ終了後、PCに以下のﾒｯｾｰｼﾞが表示される
“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．端末とﾊﾟｿｺﾝをRS232Cｹｰﾌﾞﾙで接続し、ﾊｲﾊﾟｰﾀｰﾐﾅﾙを起動する
２．CTRL +KXORBと入力してｺﾏﾝﾄﾞﾓｰﾄﾞに入る
３．CTRL + UADLTと入力する
PCに以下のﾒｯｾｰｼﾞが表示される
“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． 端末とﾊﾟｿｺﾝをRS232Cｹｰﾌﾞﾙで接続し、ﾊｲﾊﾟｰﾀｰﾐﾅﾙを起動する。
２． CTRL +KXORBと入力してｺﾏﾝﾄﾞﾓｰﾄﾞに入る。
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
USER APPL       : 37EA(ACTIVE)
.......
.......
```

“USER APPL”の表示結果は、ﾕｰｻﾞｰｱﾌﾟ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ければ”NONE”と表示する。またﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝの動作状態は、ﾕｰｻﾞｰｱﾌﾟ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. ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｸﾞﾗﾑ開発上の注意点

ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｸﾞﾗﾑを開発する時には次の点に注意してください。

- 大きいｻｲｽﾞのRAM領域を使用する際にはｸﾞﾛｰﾊﾞﾙ宣言してください。ﾛｰｶﾙ宣言するとｽﾀｯｸｵｰﾊﾞ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.
- ﾃﾞﾊﾞｯｸﾞする時には、KXS67=1、 KXS37=0に設定してください
- 標準ﾗｲﾌﾞﾗﾘの“sprintf()”で 複数の“double” や “float”型の変数を”char”型に変換する場合、約300ﾊﾞ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ｵﾊﾞｰﾌﾛｰし、ﾌﾟﾛｸﾞ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。
- ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｸﾞﾗﾑが暴走した場合には、“CTRL+R”を入力しながらﾘｾｯﾄﾎﾞﾀﾝを押してください。そうすれば、ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｸﾞﾗﾑは起動されません。このときのPCの通信ﾓｰﾄﾞは以下にしてから入力してください。

  ﾎﾞｰﾚｰﾄ ：4800bps
  ﾊﾟﾘﾃｨ ：なし
  ｽﾄｯﾌﾟﾋﾞｯﾄ ：1bit
  ﾃﾞｰﾀ長 ：8bit

Appendix A. ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝの開発環境

- **ハードウェア環境**

  ﾊﾟ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
ﾒﾓﾘｰ	5.6MB 以上
ﾊｰﾄﾞﾃﾞｨｽｸ	3MB以上

- **ソフトウェア**

  松下電器産業(株)製の以下のｿﾌﾄｳｴｱ/ﾄﾞｷｭﾒﾝﾄが必要です。.

  MN10200 ｼﾘｰｽﾞ C ｺﾝﾊﾟｲﾗｰ
  MN10200 ｼﾘｰｽﾞ ｱｾﾝﾌﾞﾗ
  MN10200 ｼﾘｰｽﾞ ﾘﾝｶｰ
  MN10200 ｼﾘｰｽﾞ ﾗｲﾌﾞﾗﾘ
  ﾌｧｲﾙ変換ﾕｰﾃｨﾘﾃｨ
  ﾕｰｻﾞｰﾏﾆｭｱﾙ

  オーブコムジャパン(株)供給の以下のｿﾌﾄｳｴｱ/ﾄﾞｷｭﾒﾝﾄが必要です。

  KME ﾗｲﾌﾞﾗﾘ
  ﾕｰｻﾞｰ ｱﾌﾟﾘｹｰｼｮﾝ ﾌﾟﾛｸﾞﾗﾑ ｶﾞｲﾄﾞ
  ｻﾝﾌﾟﾙ ﾌﾟﾛｸﾞﾗﾑ

Appendix B. PCの環境設定

１．環境変数の設定

ｺﾝﾊﾟｲﾗが動作できるように、環境変数PATH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ﾃﾞｨﾚｸﾄﾘとして環境変数TMP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(CC102L.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) Cｿｰｽﾌｧｲﾙ内で絶対ﾊﾟｽが指定されていないｲﾝｸﾙｰﾄﾞﾌｧｲﾙの標準ﾃﾞｨﾚｸﾄﾘ
(2) ﾄﾞﾗｲﾊﾞからﾘﾝｶを起動した際に自動的にﾘﾝｸされるﾗｲﾌﾞﾗﾘﾌｧｲﾙ
(3) ﾘﾝｶで自動的にﾘﾝｸされるﾗｲﾌﾞﾗﾘﾌｧｲﾙの標準ﾃﾞｨﾚｸﾄﾘ
(4) ﾃﾝﾎﾟﾗﾘﾌｧｲﾙを生成する標準ﾃﾞｨﾚｸﾄﾘ
(5) ﾄﾞﾗｲﾊﾞが起動する各ｺﾏﾝﾄﾞの標準ﾃﾞｨﾚｸﾄﾘ

<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(CC102L.rc)は “ cc102l\bin “ に置きます。

変更履歴 （Ver3.0→ Ver4.0）

5．KMEﾗｲﾌﾞﾗﾘ　6章の関数追加、削除に伴い表を一部変更

6．各関数の詳細説明

set_pin_code	関数型、引数範囲の誤り訂正
set_ncc_search_mode	引数の説明変更、戻り値の誤り訂正
chk_ib_tx	引数、戻値の記載変更
req_pos_report_to_ncc	戻り値の訂正
get_ob_msg_subject_ind	関数型の訂正
get_ob_msg_msg_len	関数型の訂正
get_ob_user_command	関数型の訂正
get_ob_global	関数名の変更
chk_tx_ready	説明修正
chk_cd	関数説明追加
chk_rts	関数説明追加
set_pdop_th	関数説明追加
get_pdop_th	関数説明追加
break_point	関数型の訂正、戻り値説明の追加
get_application_dbg	戻り値の訂正
get_convert_time	関数説明追加
chk_apl_sat_predict_result	関数名誤記訂正
get_kme_serial_no()	関数説明追加
set_no_eait_for_power_save()	関数説明追加
os_monitor()	関数説明追加
print_double()	関数説明追加
orb_sprintf1()	関数説明追加
⋮	
orb_sprintf7()	関数説明追加
set_kxs16, get_kxs16	引数、戻り値範囲訂正
set_kxs26	引数単位訂正
set_kxs27	引数単位訂正
set_kxs30,get_kxs30	引数、戻り値範囲訂正
set_kxs33,get_kxs33	引数、戻り値範囲訂正
set_kxs43	関数説明追加
set_kxs47,get_kxs47	引数、戻り値範囲訂正
set_kxs48	関数説明追加
set_kxs49,get_kxs49	削除
set_kxs50,get_kxs50	引数、戻り値範囲訂正
set_kxs60,get_kxs60	引数、戻り値範囲訂正
get_kxs63,set_kxs63	関数型訂正
set_kxs66	関数説明追加
set_kxs68,get_kxs68	引数、戻り値範囲訂正
set_kxs70,get_kxs70	引数、戻り値範囲訂正
set_kxs73,get_kxs73	削除
get_kxs80,set_kxs80	説明訂正

get_kxs80,set_kxs80	説明追加
get_kxs88,set_kxs88	説明追加
get_kxd02	引数範囲訂正
set_kxa01,set_kxa02,set_kxa03	引数説明変更
set_kxa04,get_kxs04	削除
set_kxa05	引数説明変更
set_kxb01,set_kxb02,set_kxb03	引数説明変更

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()のﾌﾟﾛｸﾞﾗﾑ例追記
21	s_msg_polling() のﾌﾟﾛｸﾞﾗﾑ例追記
22	globalgram_polling() のﾌﾟﾛｸﾞﾗﾑ例追記
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()のﾌﾟﾛｸﾞﾗﾑ例の間違いを修正
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() のﾌﾟﾛｸﾞﾗﾑ例の間違いを修正
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へのﾎﾟｰﾘﾝｸﾞ処理のﾌﾛｰﾁｬｰﾄを修正
219	ﾌﾞﾚｰｸﾎﾟｲﾝﾄによるｻｽﾍﾟﾝﾄﾞ状態からのﾚｼﾞｭｰﾑｺﾏﾝﾄﾞの記述を修正 (誤) “go” → (正)“g”
224	ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｸﾞﾗﾐﾝｸﾞ開発上の注意点に説明を追加