オーブコム端末装置
KX-G7201N用
ユーザーアプリケーション
プログラムミングガイド

Ver 1.5

本仕様は予告なく変更する場合があります。

２００３年1月6日

オーブコムジャパン 株 式 会 社

目次

1. 概要

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

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

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

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

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

2. SCのデータ処理

2.1. メッセージ通信処理の概要

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

(1)ＳＣと衛星間のメッセージ通信

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

(2)SC内部でのデータ処理

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

（３）ＳＣと外部端末との通信

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

2.2. 衛星とSC間の通信制御

2.2.1. メインシステムで衛星キューを制御するには

この場合、KXS69= 0、KXS68= 0と設定します。
衛星キューとDTEは、メインシステムの制御下でオーブコム社シリアルインターフェース仕様に準拠した通信方式であるプロトコルモードまたはバイトモードでDTEと通信を行います。また、この設定ではユーザーアプリケーションで作成したメッセージをインバウンドキューにセットして、衛星に送信することは可能ですが、受信メッセージをユーザーアプリケーションで処理することは出来ません。

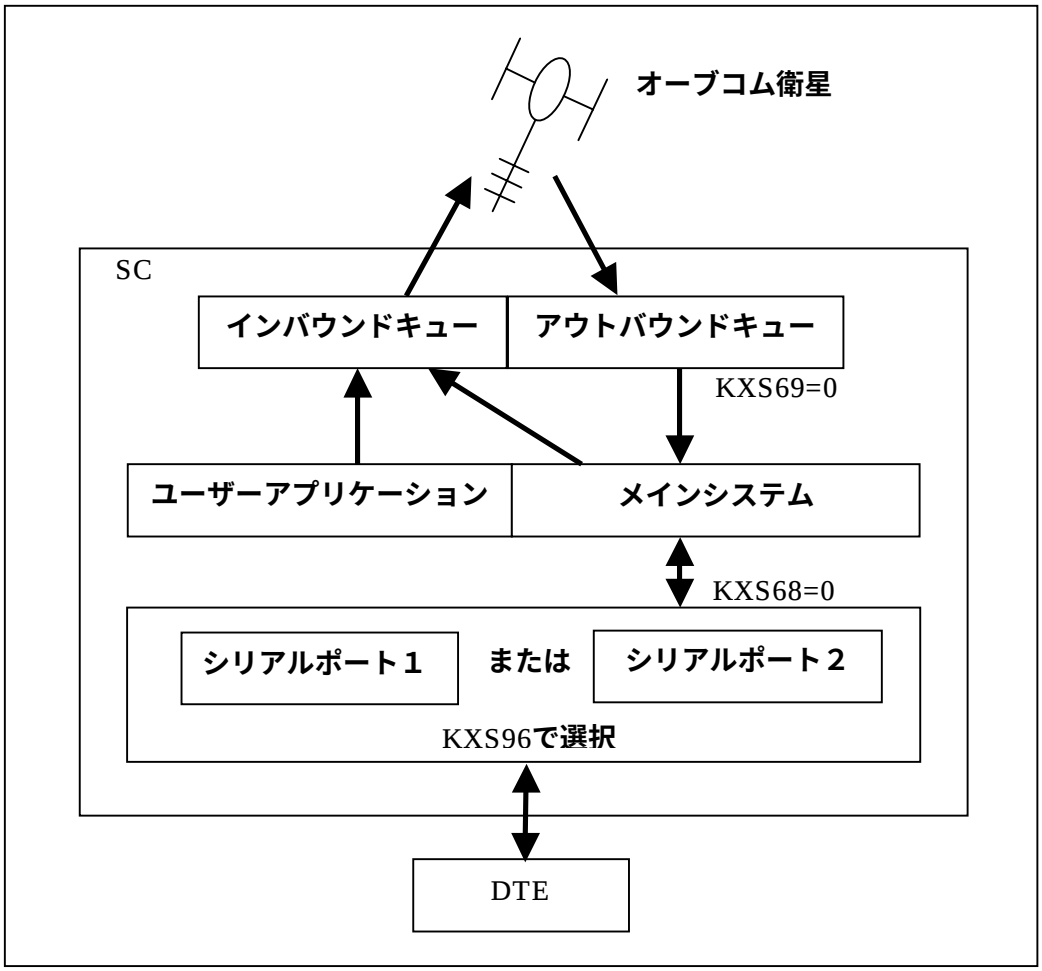


図 2-1

注意：

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

2.2.2. ユーザーアプリケーションで衛星キューを制御するには

この場合、KXS69= 1と設定します。
衛星からの受信メッセージは、ユーザーアプリケーションで処理を行います。ユーザーアプリケーションの制御下でDTEと通信を行う場合には、KXS68=1と設定しますが、通信方式はユーザーアプリケーション内で定義しなければなりません。
また、KXS68=0と設定した場合には、DTEからはメインシステムを経由してのプロトコルモードまたはバイトモードでのメッセージ送信は可能です。

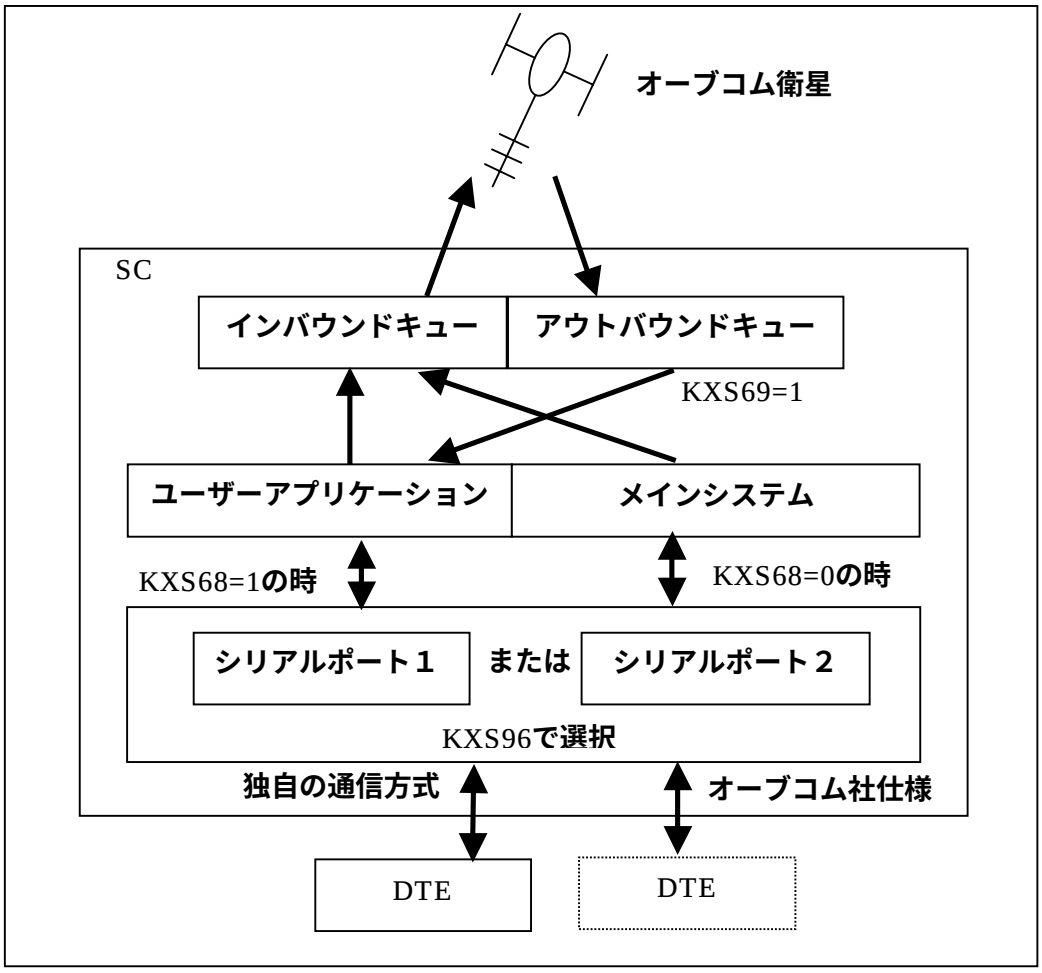


図 2-2

注意：

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

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

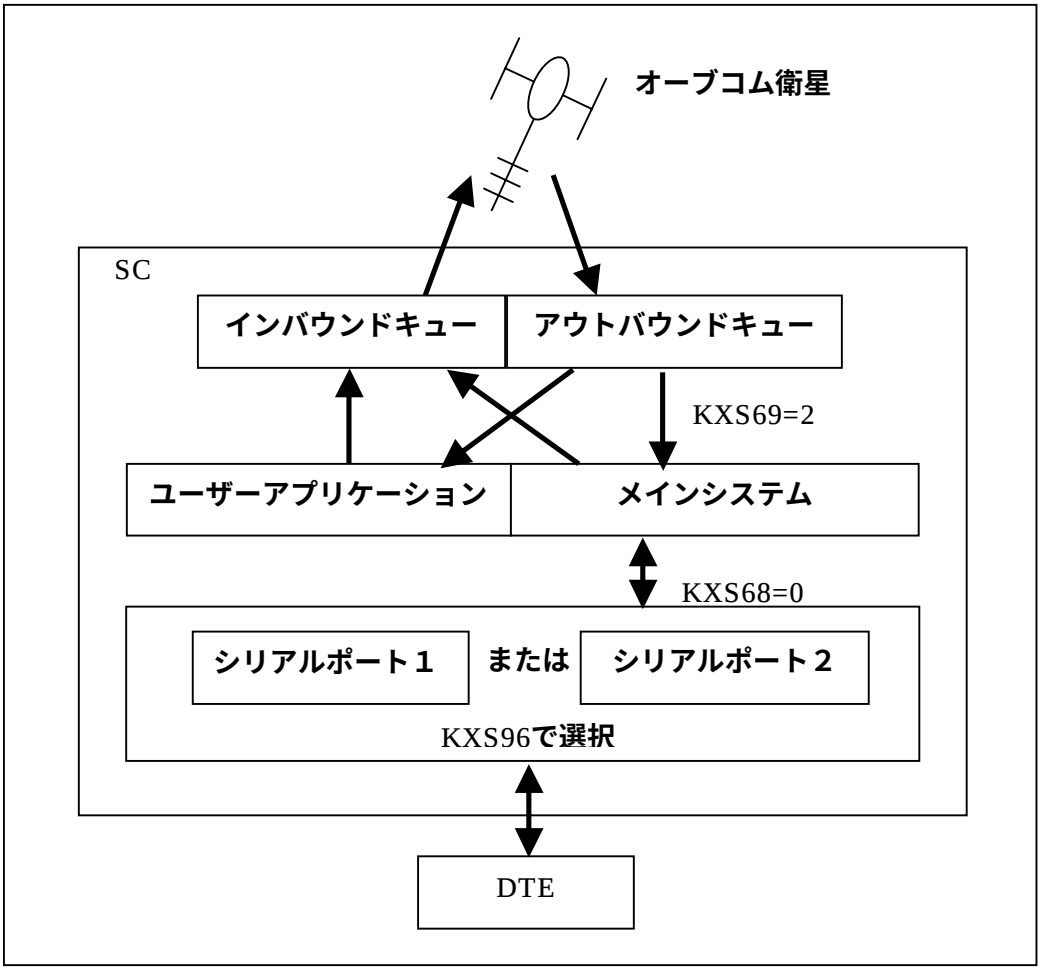


図 2-3

注意：

(1) アウトバウンドキュー内のメッセージは、メインシステムとユーザーアプリケーションの両方の処理が終わった後で削除されるようにするため、ユーザーアプリケーションは、全ての受信メッセージ毎に削除できる事をメインシステムに知らせなければなりません。メインシステムでは、ユーサーアプリでの処理終了通知を受けて、キューのメッセージを削除します。

(2) KXS68=1に設定すると、メインシステムが読み出したアウトバウンドキューをDTEに出力できないため、場合によってはアウトバウンドキューがオーバーフローする恐れがあります。この設定は出来るだけ避けてください。

2.3. DTEとSC間の通信制御

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

2.3.1. プロトコルモード／バイトモード通信をする場合

この場合、KXS68の設定を０（KXS68=0）にします。
プロトコルモードまたはバイトモードは、メインシステムがサポートしていますので、この通信方式でDTEと通信する場合には、メインシステムがシリアルポートを使用するように設定します。通信方式としてプロトコルモードで通信を行う場合は、KXS31=0、バイトモードの場合にはKXS31=1にします。使用するポートはKXS96で選択します。但し、ユーザーアプリケーションでは、SCがサポートしているこの通信方式は使用できません。
本通信方式が設定されている場合においても、２つのシリアルポートの何れからもCTRL+KX0RBの入力でコマンドモードに入る事ができます。但し、コマンドモード動作中は、プロトコルモードおよびバイトモードの通信はできません。

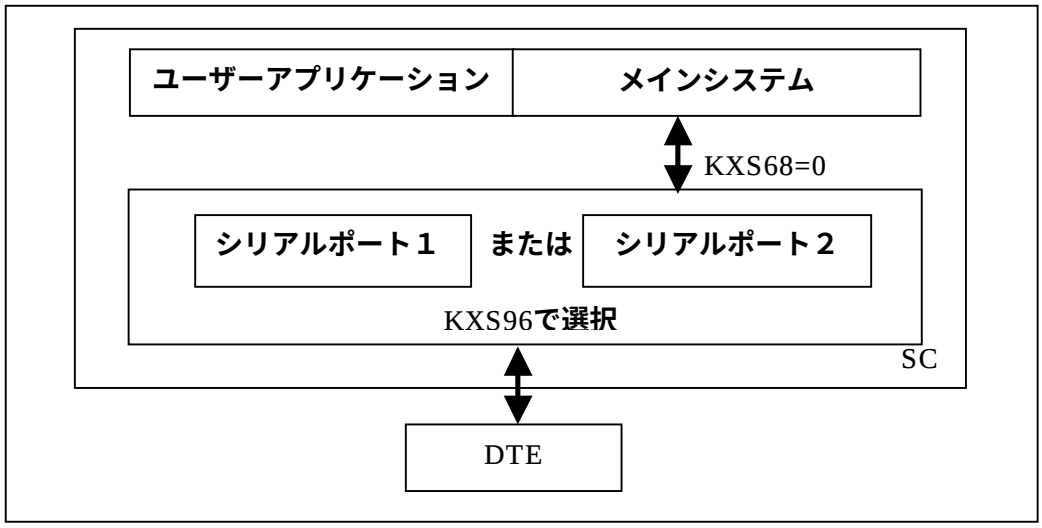


図 2-4

2.3.2. ユーザーアプリケーションで通信を制御する場合

この場合、KXS68の設定を１（KXS68=1）にします。 ユーザーアプリケーションにシリアルドライバの制御権が亘り、メインシステムではアクセスできません。使用するシリアルポートはKXS96で選択します。
本通信方式が設定されている場合においても、２つのシリアルポートの何れからもCTRL+KX0RBの入力でコマンドモードに入る事ができます。但し、コマンドモード動作中は、プロトコルモードおよびバイトモードの通信はできません。

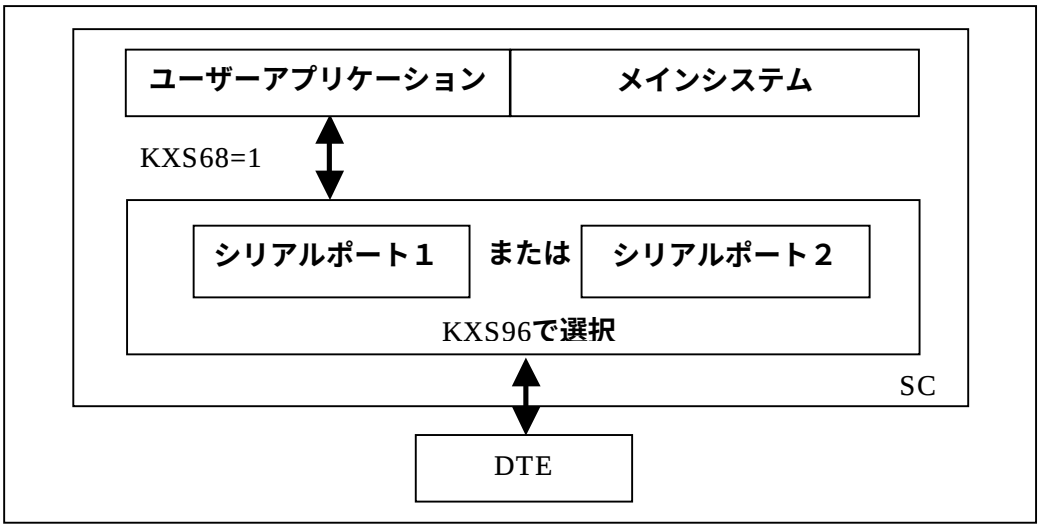


図 2-5

3. ユーザーアプリケーションのプログラミング

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

3.1. スタートアップルーチン

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

- レジスタの初期化
- ユーザーアプリケーションプログラムへの分岐

3.2. RAMの初期化

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

3.3. ユーザーアプリケーションプログラムへの分岐

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

3.4. 割り込み

ユーザーアプリケーションは、CPUの割り込み機能を使う事ができません。

3.5. スタックエリア

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

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

また、標準関数の "spritf()" は、スタックを多く消費するため、ユーザーアプリケーションの不具合の原因になる可能性があります。特に浮動小数点変数を変換すると数百バイトのスタックを消費します。標準関数の"spritf()"に変わる"print_double()"や”orb_sprintf1()”等の代用関数を準備していますので使用してください。

3.6. メモリーマップ

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

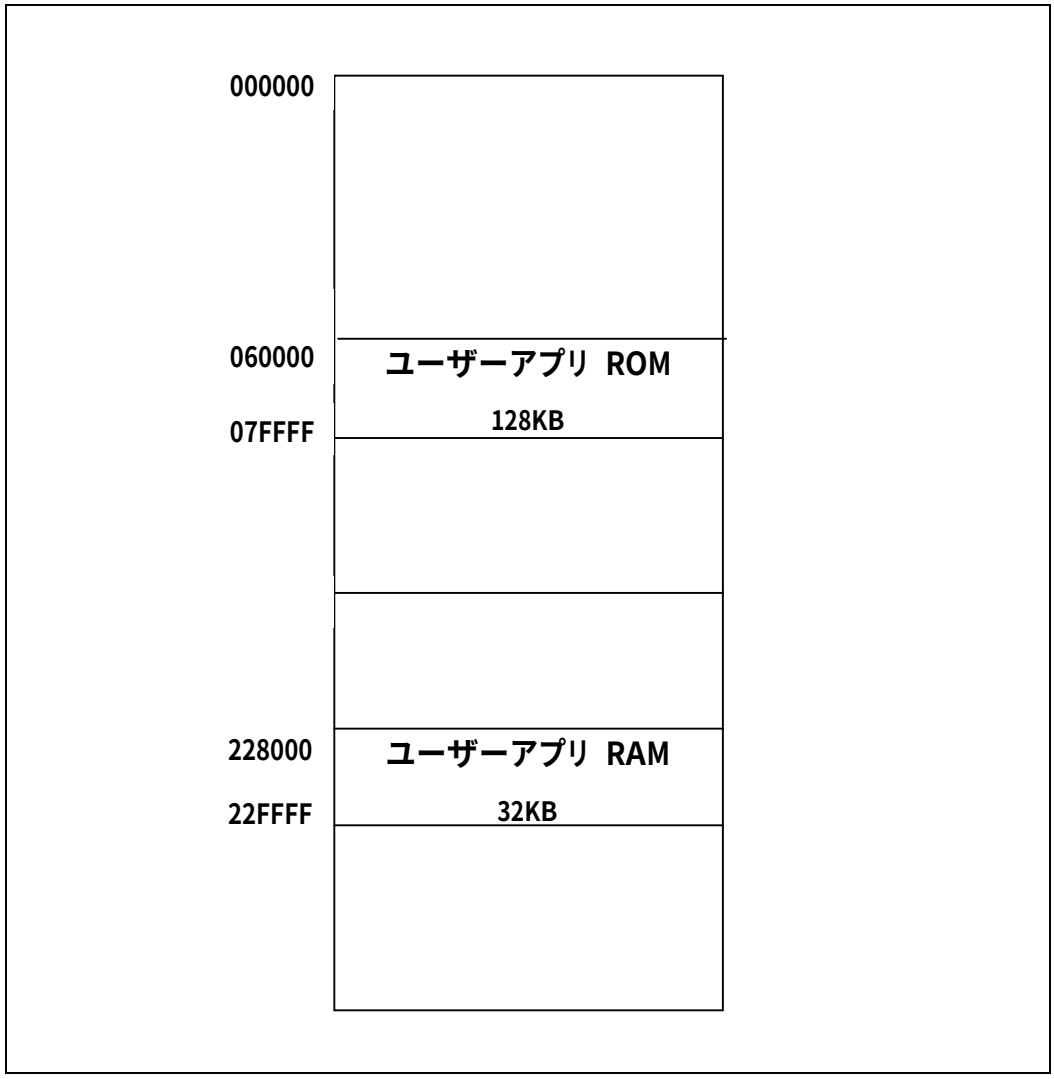


図 3-1

4. ユーザーアプリケーションの起動方法

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

4.1. SCがパワーオンする時に起動

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

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

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

5. ＫＭＥライブラリ

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

5.1. Configuration

SCのコンフィグレーションパラメータを設定します。

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	44
req_send_ib_report()	送信ｷｭｰにﾚﾎﾟｰﾄをｾｯﾄする	46
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	48
req_send_ib_enh_globalgram()	送信ｷｭｰにｴﾝﾊﾝｽﾄｸﾞﾛｰﾊﾞﾙﾒｯｾｰｼﾞをｾｯﾄする	49
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	50

DTEにメッセージを送信する為、端末の受信キューにメッセージをセットする.

関 数	説 明	頁
req_send_ob_message()	受信ｷｭｰにﾒｯｾｰｼﾞをｾｯﾄする	51

5.5. SC-Terminated Message

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

関 数	説 明	頁
get_unget_ob_id_for_user()	ｱｳﾄ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	53
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	56
get_ob_type()	ﾃﾞｰﾀ種を獲得する	57
get_ob_ncc_id()	ﾒｯｾｰｼﾞのGCC IDを獲得する	58
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	59
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	60
get_ob_msg_or_quan()	ﾒｯｾｰｼﾞの送信元を獲得する	61
get_ob_msg_msg_len()	ﾒｯｾｰｼﾞ長を獲得する	62
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	63
get_ob_message ()	ﾒｯｾｰｼﾞを獲得する	64
get_ob_user_command()	ｺﾏﾝﾄﾞを獲得する	65
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	66
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	67
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	68
get_ob_globalgram ()	ｸﾞﾛｰﾊﾞﾙﾒｯｾｰｼﾞを獲得する	69
	.	

5.6. Power Control

端末をスリープさせる。

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

ブロック電源を制御する。

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

5.7. Serial Interface

DTEに対してデータの送受信を行う。

関 数	説 明	頁
chk_rx_buff()	ﾒｲﾝﾎﾟｰﾄのRS232Cの受信ﾊﾞｯﾌｧをﾁｪｯｸする	75
get_rx_data()	ﾒｲﾝﾎﾟｰﾄのRS232Cの受信ﾊﾞｯﾌｧからﾃﾞｰﾀを獲得する	76
chk_tx_ready()	RS232Cの送信ﾊﾞｯﾌｧが満杯かをﾁｪｯｸする	77
set_tx_data()	ﾒｲﾝﾎﾟｰﾄのRS232Cの送信ﾊﾞｯﾌｧにﾃﾞｰﾀをセットする	78
cts_act()	ﾒｲﾝﾎﾟｰﾄのCTSポートをｱｸﾃｨﾌﾞにする	79
cts_neg()	ﾒｲﾝﾎﾟｰﾄのCTSポートをﾈｶﾞﾃｨﾌﾞにする	80
chk_cd()	ﾒｲﾝﾎﾟｰﾄのCD信号の状態をﾁｪｯｸする	81
chk_rts()	ﾒｲﾝﾎﾟｰﾄのRTS信号の状態をﾁｪｯｸする	82
chk_rx_buff_port()	RS232Cの受信ﾊﾞｯﾌｧをﾁｪｯｸする	83
get_rx_data_port()	RS232Cの受信ﾊﾞｯﾌｧからﾃﾞｰﾀを獲得する	84
chk_tx_ready_port()	RS232Cの送信ﾊﾞｯﾌｧが満杯かをﾁｪｯｸする	85
set_tx_data_port()	RS232Cの送信ﾊﾞｯﾌｧにﾃﾞｰﾀをセットする	86

5.8. Time Functions

セットしたタイマをチェックする。

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。

アナログポートをコントロールする。

5.10. Measurement

端末の測位機能をコントロールする。

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。

5.12. Status Information

端末から各種情報を獲得する。

5.13. Other Functions

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

5.14. KX Command

ユーザーアプリケーション用KXコマンド（KXS，KXBコマンド）一覧表。

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 KXS01 ｺﾏﾝﾄﾞでも変更できる。

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　　0: 設定成功
1: 設定失敗

補足　　ｺﾏﾝﾄﾞﾓｰﾄﾞの KXS50 ｺﾏﾝﾄﾞでも変更できる。
ﾋﾟﾝｺｰﾄﾞはｹﾞｰﾄｳｴｲと端末に設定する。ｹﾞｰﾄｳｴｲがﾋﾟﾝｺｰﾄﾞ参照ﾓｰﾄﾞになっている時は、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ない。

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る。

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する。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る。

　　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する。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る。

　　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　　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。

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ができない

補足　　この関数は、衛星に格納してあるすべての自分宛ての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する。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る。

　　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(NCC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 : ｱﾎﾞｰﾄ要求の発行成功
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中の全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内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　KXS23 に設定してある測位情報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の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
user_data_len : ｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑ長

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_enh_globalgram

要約

```
Int req_send_ib_enh_globalgram( unsigned char NCC_id,
                                unsigned char mha_ref_num,
                                unsigned char priority,
                                unsigned char msg_body_type,
                                unsigned char msg_body_sub_type,
                                unsigned char *rcpnt[7],
                                unsigned char *subject,
                                unsigned char *message,
                                unsigned int message_len)
```

ncc_id : GCC 番号
mha_ref_num : 識別番号
priority : 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

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

ｲﾝﾊﾞｳﾝﾄﾞｴﾝﾊﾝｽ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

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。

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: ｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰに設定完了
　　1: ｱｳﾄ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: ｱｳﾄ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] = ‘0’;
    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;
        }
        remove_ob_q(get_ob_id);
    }
    exit_user_apl();
}
```

remove_ob_q

要約 void remove_ob_q(ob_id rm_ob_id)

ob_id rm_ob_id : ｱｳﾄﾊﾞｳﾝﾄﾞ 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　1: ｱｳﾄﾊﾞｳﾝﾄﾞﾒｯｾｰｼﾞ
2: ｱｳﾄﾊﾞｳﾝﾄﾞｺﾏﾝﾄﾞ
3: 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

ｱｳﾄﾊﾞｳﾝﾄﾞ 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)の時でも、有効です

go_wait

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

unsigned long time: ｻｽﾍﾟﾝﾄﾞ時間
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　ﾒｲﾝﾎﾟｰﾄの 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 4kbyte です。
ﾒｲﾝﾎﾟｰﾄのみに適用されます。

get_rx_data

要約　　int get_rx_data(void);

解説　　ﾒｲﾝﾎﾟｰﾄの 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

0x00-0xff ： 受信ﾃﾞｰﾀ
-1 ： 受信ｴﾗｰ

補足　　RS232C の受信ﾊﾞｯﾌｧｻｲｽﾞは 4k ﾊﾞｲﾄです。
ｻﾌﾞﾎﾟｰﾄは get_rx_data_port 関数を使用してください。

chk_tx_ready

要約 int chk_tx_ready(void);

解説 ﾒｲﾝﾎﾟｰﾄの RS232C の送信ﾊﾞｯﾌｧ（512 ﾊﾞｲﾄ）が満杯かをﾁｪｯｸする

戻値
0: 送信ﾊﾞｯﾌｧが満杯。
1: 送信ﾊﾞｯﾌｧは満杯ではない。

補足
送信ﾊﾞｯﾌｧが満杯の時には、送信ﾃﾞｰﾀをｾｯﾄできません。
送信ﾊﾞｯﾌｧに空きができてから、送信ﾃﾞｰﾀをｾｯﾄしてください。
ｻﾌﾞﾎﾟｰﾄは chk_tx_ready_port 関数を使用してください。

set_tx_data

要約　　unsigned int set_tx_data(unsigned char tx_data);

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

解説　　ﾒｲﾝﾎﾟｰﾄの 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ﾒｲﾝ、ｻﾌﾞ各 512 ﾊﾞｲﾄです。
この関数には送信ﾊﾞｯﾌｧが満杯かのﾁｪｯｸが含まれていませんので、使用する場合には送信ﾊﾞｯﾌｧ状態ﾁｪｯｸ関数 chk_tx_ready()と共に使用して下さい。ｻﾌﾞﾎﾟｰﾄは set_tx_data_port 関数を使用してください。

<例>

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

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

cts_act

要約 void cts_act(void)

解説 ﾒｲﾝﾎﾟｰﾄの 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 ﾒｲﾝﾎﾟｰﾄの 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

chk_rx_buff_port

要約　unsigned int chk_rx_buff_port (unsigned char port);

unsigned char port : port
 0: ｼﾘｱﾙﾎﾟｰﾄ 1 (ﾒｲﾝ)
 1: ｼﾘｱﾙﾎﾟｰﾄ 2 (ｻﾌﾞ)

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　受信ﾊﾞｯﾌｧのｻｲｽﾞ
  0: ｼﾘｱﾙﾎﾟｰﾄ 1 (ﾒｲﾝ) 4k byte
  1: ｼﾘｱﾙﾎﾟｰﾄ 2 (ｻﾌﾞ) 512 byte

get_rx_data_port

要約　　int get_rx_data_port (unsigned char port);

unsigned char port : port
 0: ｼﾘｱﾙﾎﾟｰﾄ 1 (ﾒｲﾝ)
 1: ｼﾘｱﾙﾎﾟｰﾄ 2 (ｻﾌﾞ)

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　　0x00-0xff ： 受信ﾃﾞｰﾀ
-1 ： 受信ｴﾗｰ

補足　　RS232C の受信ﾊﾞｯﾌｧｻｲｽﾞ
  0: ｼﾘｱﾙﾎﾟｰﾄ 1 (ﾒｲﾝ) 4k byte
  1: ｼﾘｱﾙﾎﾟｰﾄ 2 (ｻﾌﾞ) 512 byte

chk_tx_ready_port

要約　　int chk_tx_ready_port (unsigned char port);

unsigned char port : port
 0: ｼﾘｱﾙﾎﾟｰﾄ 1 (ﾒｲﾝ)
 1: ｼﾘｱﾙﾎﾟｰﾄ 2 (ｻﾌﾞ)

解説　　RS232C の送信ﾊﾞｯﾌｧが満杯かをﾁｪｯｸする

戻値　　0: 送信ﾊﾞｯﾌｧが満杯。
1: 送信ﾊﾞｯﾌｧは満杯ではない。

補足　　送信ﾊﾞｯﾌｧが満杯の時には、送信ﾃﾞｰﾀをｾｯﾄできない。
送信ﾊﾞｯﾌｧに空きができてから、送信ﾃﾞｰﾀをｾｯﾄすること。

RS232C の送信ﾊﾞｯﾌｧｻｲｽﾞ
  0: ｼﾘｱﾙﾎﾟｰﾄ 1 (ﾒｲﾝ) 512 byte
  1: ｼﾘｱﾙﾎﾟｰﾄ 2 (ｻﾌﾞ) 512 byte

set_tx_data_port

要約　　unsigned int set_tx_data_port (unsigned char port,
　　unsigned char tx_data);

　　unsigned char port : port
　　0: ｼﾘｱﾙﾎﾟｰﾄ1 (ﾒｲﾝ)
　　1: ｼﾘｱﾙﾎﾟｰﾄ2 (ｻﾌﾞ)
　　unsigned char tx_data : send data　(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
　　0: ｼﾘｱﾙﾎﾟｰﾄ1 (ﾒｲﾝ)　512 byte
　　1: ｼﾘｱﾙﾎﾟｰﾄ2 (ｻﾌﾞ)　512 byte

　　この関数には送信ﾊﾞｯﾌｧが満杯かのﾁｪｯｸが含まれていませんので、使用する場合には送信ﾊﾞｯﾌｧ状態ﾁｪｯｸ関数 chk_tx_ready_port()と共に使用して下さい。

<例>

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

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

get_inc_timer

要約　unsigned int get_inc_timer(void);

解説　OSﾀｲﾏｰのｶｳﾝﾀ値を獲得する。
このｶｳﾝﾀは2 msec毎にｶｳﾝﾄｱｯﾌﾟされる。

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分で測位ができなかったので強制終了 */
              req_pos_calc_term_from_user();
              break;
          }

          /* 10秒毎に測位終了をﾁｪｯｸ */
          go_wait(10, SEC_UT);        /* 10秒ｽﾘｰﾌﾟ */
      }
  }
  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

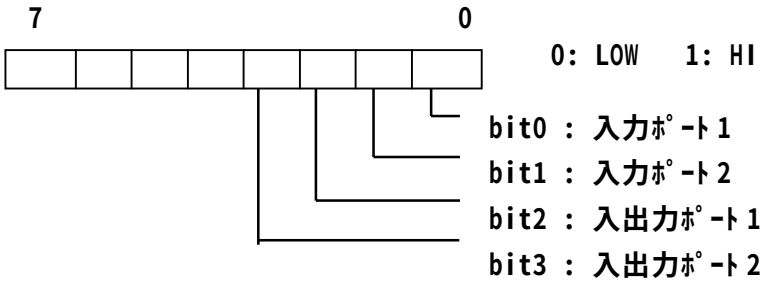
補足

get_digital_port

要約　unsigned int 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



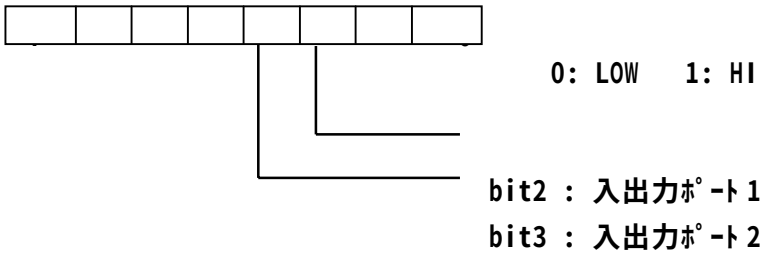
bit8-15：ﾘｻﾞｰﾌﾞ

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 int sel_port, unsigned int data);

unsigned int sel_port: 出力ﾎﾟｰﾄ識別番号
unsigned int 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　　set_kxs94 にて出力モードに設定した場合のみ有効です。

<例>

```
#include “kme_lib.h”

void main(void)
{
  /* 出力ﾎﾟｰﾄ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: ｱﾅﾛｸﾞﾎﾟｰﾄ 3
 4- 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　　旧機種　KX-G710x/N と変換レンジが異なっています。同じ電圧を検知しても旧機種とは変換値が異なりますので注意してください。

KX-G710x/N　 0-3.3V を 0-255 で変換
KX-G7201N　 set_kxs90 で設定した範囲を 0-255 で変換

get_adcon_data_10bit

要約　　unsigned int get_adcon_data_10bit(int ch);

int ch : ｱﾅﾛｸﾞﾎﾟｰﾄ番号
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: ｱﾅﾛｸﾞﾎﾟｰﾄ 3
4- 7: 予備

解説　　ｱﾅﾛｸﾞﾎﾟｰﾄの状態を獲得する(10ﾋﾞｯﾄ)

戻値　　値 (0 -1023)

補足　　set_kxs90 で設定した電圧または電流範囲を 0-1023 で変換します。

req_pos_calc_term_from_user

要約　　void req_pos_calc_term_from_user(void):

解説　　測位を中止する

戻値　　なし

補足　　測位が起動してない場合には、無効となる。

<例>

```
#include “kme_lib.h”

void main(void)
{
    if (req_pos_calc_from_user() == 1) {        /* 測位開始要求 */
        /* 測位開始失敗 */
    }
    else {
        /* 測位開始 */
        if (set_timer(1, 15, MIN_UT) == 0) {    /* ﾀｲﾏｰ設定 */
            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いたらこの要求は無視されるが、戻り値は０が返る。

chk_pos_calc_result

要約　　int chk_pos_calc_result(void);

解説　　測位が終了したかをﾁｪｯｸする。

戻値　　0: 測位中
1: 測位終了

補足

get_pos_age

要約　unsigned int get_pos_age(void)

解説　測位結果の経過時間（古さ）を獲得する。

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

get_pdop_th

要約　int get_pdop_th(void)

解説　GPS位置情報をフィルタする時のPDOPのしきい値を獲得する。

戻値　PDOP 値（１～１０）

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

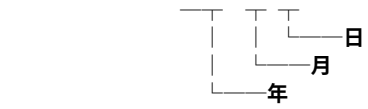
補足　　x05 ｾﾝﾃﾝｽは、GPS から出力される測位情報である。

< 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）>

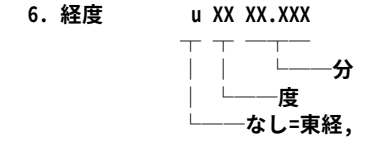
bit4: 衛星数不足
Bit5: 演算不可能
bit6: その他異常
bit7: 初期異常

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



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

x06 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、GPS から出力される受信可能衛星情報である。

```
< X06ｾﾝﾃﾝｽﾌｫｰﾏｯﾄ >
 "$PKMEX06,XX[,XX,XX,XX.....XX,XX]*cc<CR><LF>"    *)最後はNULL。
                ┬  ┬ ┬ ┬
                │  │ │ └── 衛星仰角(0 - 90) *1)
                │  │ └──── 衛星方位各(0 - 360) ( 0=北, 180=南)
                │  └──────  衛星番号
                └── 観測可能衛星数
     *cc: Checksum *2)

      *1) 全ての可視衛星の衛星番号・仰角・方位角

      *2) ﾁｪｯｸｻﾑ：ﾁｪｯｸｻﾑ値は、ｾﾝﾃﾝｽ中の ’$’ と ’*’ とで挟まれたｷｬﾗｸﾀﾃﾞｰﾀを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 mha ﾚﾌｪﾚﾝｽ番号を獲得する

戻値　　0x00 - 0xfe: 現在送信中の mha ﾚﾌｪﾚﾝｽ番号
　　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_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：設定完了
０：設定ｴﾗｰ。KXS70=0 になっている
-1：設定ｴﾗｰ。ﾌﾞﾚｰｸﾎﾟｲﾝﾄ番号に０が設定されている

補足　“break_point” は、255 個まで設定でき、それぞれ ON/OFF 設定ができます。
“break 23,0” : ﾌﾞﾚｰｸﾎﾟｲﾝﾄ 23 を無効にする
“break 23,1” : ﾌﾞﾚｰｸﾎﾟｲﾝﾄ 23 を有効にする

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）の時に有効となります。

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 1: ﾌﾞﾚｰｸﾎﾟｲﾝﾄが設定されている
0: 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;
```

解説　　UTC 時刻を獲得する

戻値　　なし

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

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

set_utc_time

要約　　void set_utc_time(TIME_INFO *time);

TIME_INFO *time: UTC 時刻

```
struct time_buff {
unsigned char  year;       /* 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;
```

解説　　UTC 時刻を設定する

戻値　　なし

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

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

get_local_time

要約　　void get_local_time(TIME_INFO *local_time, int time_zone)

TIME_INFO *time: ﾛｰｶﾙ時刻

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

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

解説　　ﾛｰｶﾙ時刻を獲得する

戻値　　なし

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

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

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_zone　　:　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す。 本関数使用の際は軌道計算用ﾗｲﾌﾞﾗﾘが必要です。

<例.>

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

    req_apl_sat_predict( 60 * 60 );

    while ( chk_apl_sat_predict_result() == 1 );

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

    exit_user_apl();    /* ﾕｰ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　　本関数使用には軌道計算用ﾗｲﾌﾞﾗﾘが必要です。

get_apl_sat_predict_time

要約　int get_apl_sat_predict_time(TIME_INFO *time)

TIME_INFO *time: 衛星飛来時刻

解説　衛星飛来時刻を獲得する

戻値　0: 衛星飛来時刻を獲得できなかった
1: 衛星飛来時刻を獲得できた

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

get_sys_info

要約　　char get_sys_info(int obj)

int obj : 番号
 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

 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
 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　　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
```

os_monitor

要約 void os_monitor(int *os_work)

unsigned int *os_work : work buffer

解説 何もしない。

戻値 なし

補足 本関数は 710x シリーズの同名関数のダミーとして準備しており、710x シリーズのソースをそのまま使用する場合、os_monitor 関数を削除しないで良い様にしています。

<参考:710x シリーズアプリにおける本関数の動作>
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。

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　　端末はKXS14の設定に従いこのGCCとﾘﾝｸしている衛星を捕捉しようとする。

get_kxs01

要約　　unsigned char get_kxs01(void)

解説　　ＧＣＣ番号 (0 - 255)を獲得する

戻値　　ＧＣＣ番号(0 - 255).

補足

例:

```
#include “kmelib.h”              /* ﾍｯ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 ﾎﾟｰﾙﾄﾞ設定値を獲得する

戻値　　ﾃﾞﾌｫﾙﾄ polled 設定値を獲得する
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 ﾌﾟﾗｲｵﾘﾃｨ設定値を獲得する

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) 設定値を獲得する

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*/
    or_ind = 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) 設定値を獲得する

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 */
    or_ind = 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) 設定値を獲得する

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_kxs06();     /* 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) 設定値を獲得する

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_kxs07(&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) 設定値を獲得する

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_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ｻｰﾁﾓｰﾄﾞ設定値を獲得する

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);                /* NCCｻｰﾁﾓｰﾄﾞを設定する */
    search_mode = get_kxs14();   /* NCC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　　優先的に探索する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　　優先的に探索するﾀﾞｳﾝﾘﾝｸﾁｬﾝﾈﾙ（24ﾁｬﾝﾈﾙ）設定値を獲得する

戻値　　なし

補足

例：

```
#include “kme_lib.h”              /* ﾍｯ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設定値を獲得する

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の設定値を獲得する

戻値　　ﾌﾚｰﾑ数(1 - 16).

補足

例：

```
#include “kme_lib.h”              /* ﾍｯ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　　連続測位ﾓｰﾄﾞ設定値を獲得する

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 */
    mode = get_kxs18();    /* 連続測位ﾓｰﾄﾞを獲得する */
}
```

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　　測位ﾓｰﾄﾞ設定値を設定する

戻値　　 0: 設定成功
 1: 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)
 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　　KXB ｺﾏﾝﾄﾞでの GPS 測位情報のﾌｫｰﾏｯﾄを設定する

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

補足　　ﾌｫｰﾏｯﾄ詳細はｺﾏﾝﾄﾞﾘｽﾄ仕様書を参照。

get_kxs25

要約　　int get_kxs25(void)

解説　　KXBｺﾏﾝﾄﾞでのGPS測位情報ﾌｫｰﾏｯﾄ設定値を獲得する

戻値　　ﾌｫｰﾏｯﾄ
 0: LAT/LON
 1: NMEA

補足

例：

```
#include “kme_lib.h”              /* ﾍｯ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設定値を獲得する

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　　unsigned char get_kxs27(void)

解説　　ｼﾘｱﾙﾌﾟﾛﾄｺﾙのACK時間設定値を獲得する

戻値　　時間 (1 - 30 秒)

補足

例：

```
#include “kme_lib.h”              /* ﾍｯ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が返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設定値を獲得する

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設定値を獲得する

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(0 - 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 ncc_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)
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の通信ﾓｰﾄ設定値を獲得する

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_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_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設定値を獲得する

戻値　　ﾄﾘｶﾞｰ(0, 1).

補足

set_kxs33

要約　　int set_kxs33(unsigned long value)

　　unsigned long value : 時間 (1 - 604800) [秒]

解説　　ﾊ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 long get_kxs33(void)

解説　　ﾊﾞｲﾄﾓｰﾄﾞﾀｲﾑｱｳﾄ設定値を獲得する

戻値　　時間 (1 - 604800 秒)

補足

set_kxs34

要約　　int set_kxs34(unsigned int value)

　　unsigned char value : 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設定値を獲得する

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設定値を獲得する

戻値　　なし

補足

set_kxs36

要約　　int set_kxs36(unsigned char value)

unsigned char value : type (0 - 2)
0: ﾒｯｾｰｼﾞ
1: ﾚﾎﾟｰﾄ
2: 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設定値を獲得する

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設定値を獲得する

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_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_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

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設定値を獲得する

戻値　　時間 (0 - 86400 秒)

補足

set_kxs40

要約　　int set_kxs40(unsigned char value)

unsigned char value : ﾓｰﾄﾞ
 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設定値を獲得する

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の設定を獲得する

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設定値を獲得する

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。
ｻﾌﾞﾎﾟｰﾄは set_kxs43_port 関数を使用してください。

get_kxs43

要約　　void get_kxs43(SERIAL_PAR *serial)

SERIAL_PAR *serial
unsigned char baud : 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 通信ﾓｰﾄﾞ設定値を獲得する

戻値　　なし

補足　　ｻﾌﾞﾎﾟｰﾄはget_kxs43_port関数を使用してください。

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_kxs43_port

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

unsigned char port : port
 0: ｼﾘｱﾙﾎﾟｰﾄ 1 (ﾒｲﾝ)
 1: ｼﾘｱﾙﾎﾟｰﾄ 2 (ｻﾌﾞ)

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_port

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

unsigned char port : port
 0: ｼﾘｱﾙﾎﾟｰﾄ 1 (ﾒｲﾝ)
 1: ｼﾘｱﾙﾎﾟｰﾄ 2 (ｻﾌﾞ)

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 通信ﾓｰﾄﾞ設定値を獲得する

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_port(0,&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通信ﾓｰﾄﾞ設定値を獲得する

戻値　　0: 半２重
1: 全２重
2: 受信のみ

補足

例:

```
#include “kme_lib.h”              /* ﾍｯ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設定値を獲得する

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設定値を獲得する

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設定値を獲得する

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　　int 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ｻｲｽﾞ設定値を獲得する

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_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　　ﾋﾟﾝｺｰﾄﾞ設定値を獲得する

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設定値を獲得する

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測地系設定値を獲得する

戻値　　測地系 (0 - 100).

補足

set_kxs53

要約　　int set_kxs53(unsigned char value)

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

解説　　RTS 論理仕様を設定する

戻値　　0: 設定成功
1: 設定失敗（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足

get_kxs53

要約　　unsigned char get_kxs53(void)

解説　　RTS論理仕様設定値を獲得する

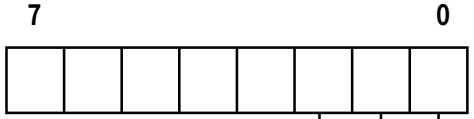
戻値　　RTS論理
0: LOW：ｱｸﾃｨﾌﾞ（ORBCOMM 仕様）
1: HI：ｱｸﾃｨﾌﾞ（一般仕様）

補足

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



bit0 : ｱﾅﾛｸﾞﾎﾟｰﾄ1
bit1 : ｱﾅﾛｸﾞﾎﾟｰﾄ2
bit2 : ｱﾅﾛｸﾞﾎﾟｰﾄ3
bit3-7:予備

解説　　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　　KXBｺﾏﾝﾄﾞで送信するｱﾅﾛｸﾞﾎﾟｰﾄ番号設定値を獲得する

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設定値を獲得する

戻値　　なし

補足

set_kxs57

要約　　int set_kxs57(double lat, double lon,
　　double radius, unsigned char unit)

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

解説　　KXB 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設定値を獲得する

戻値　　なし

補足

set_kxs58

要約　　int set_kxs58(unsigned char speed, unsigned char unit)

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

解説　　KXB 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設定値を獲得する

戻値　　なし

補足

set_kxs60

要約　　int set_kxs60(unsigned char value)

unsigned char value : ﾀｲﾌﾟ (0-5)
0: ﾃｷｽﾄ形式
1: ﾊﾞｲﾅﾘｰ形式(非短縮)
2: ﾊﾞｲﾅﾘｰ形式(短縮)
3: 位置は測位ﾚﾎﾟｰﾄで送信、その他はﾃｷｽﾄ(0 と同じ)
4: 位置は測位ﾚﾎﾟｰﾄで送信、その他はﾊﾞｲﾅﾘ(1 と同じ)
5: 位置は測位ﾚﾎﾟｰﾄで送信、その他はﾊﾞｲﾅﾘ短縮(2 と同じ)

解説　　KXB 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)
2: 排他ｴﾗｰ:KXS60(2-5) ／ KXS74(1)

補足
測位ﾚﾎﾟｰﾄはｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑ自動変換は行われない。
次の設定は無効
・KXS74(1)とKXS60(2-5) 一括送信と短縮ﾓｰﾄﾞの併用不可
・KXB位置情報送信時の相手ｱﾄﾞﾚｽ(0/R)4以上とKXS60(3-5)

get_kxs60

要約　　unsigned char get_kxs60(void)

解説　　KXBｺﾏﾝﾄﾞで送信するﾃﾞｰﾀﾌｫｰﾏｯﾄ設定値を獲得する

戻値　　ﾀｲﾌﾟ (0-5)
0: ﾃｷｽﾄ形式
1: ﾊﾞｲﾅﾘｰ形式(非短縮)
2: ﾊﾞｲﾅﾘｰ形式(短縮)
3: 位置は測位ﾚﾎﾟｰﾄで送信、その他はﾃｷｽﾄ(0 と同じ)
4: 位置は測位ﾚﾎﾟｰﾄで送信、その他はﾊﾞｲﾅﾘ(1 と同じ)
5: 位置は測位ﾚﾎﾟｰﾄで送信、その他はﾊﾞｲﾅﾘ短縮(2 と同じ)

補足

set_kxs61

要約　　int set_kxs61(unsigned char value)

unsigned char value: ﾓｰﾄﾞ (0-3)
0: 電力節減ﾓｰﾄﾞOFF
1: ﾒｲﾝﾎﾟｰﾄのみ電力節減ﾓｰﾄﾞON
2: ｻﾌﾞﾎﾟｰﾄのみ電力節減ﾓｰﾄﾞON
3: ﾒｲﾝ,ｻﾌﾞﾎﾟｰﾄ双方電力節減ﾓｰﾄﾞ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設定値を獲得する

戻値　　ﾓｰﾄﾞ (0-3)
0: 電力節減モードOFF
1: ﾒｲﾝﾎﾟｰﾄのみ電力節減モードON
2: ｻﾌﾞﾎﾟｰﾄのみ電力節減モードON
3: ﾒｲﾝ,ｻﾌﾞﾎﾟｰﾄ双方電力節減モードON

補足

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 get_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設定値を獲得する

戻値　　なし

補足

set_kxs64

要約　　int set_kxs64(unsigned char value)

unsigned char value : ｸｲｯｸ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設定値を獲得する

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_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時の、ﾕｰｻﾞｰｱﾌﾟﾘ起動設定値を獲得する

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設定値を獲得する

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設定値を獲得する

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設定値を獲得する

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設定値を獲得する

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の応答ﾚｽﾎﾟﾝｽ設定値を獲得する

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_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　　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　　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　　KXB 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　　KXBｺﾏﾝﾄﾞで生成されるﾒｯｾｰｼﾞの自動ｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑ変換ﾓｰﾄﾞ設定値を獲得する

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_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の設定値を獲得する

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設定値を獲得する

戻値　　検知時間 (5 - 20分)

補足

set_kxs80

要約　　int set_kxs80(int sw)

int sw : ｱｳﾄ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) : Not work
  ‘1’(0x31) : Work

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 : Sunday - Monday
  ‘0’(0x30) : Not work
  ‘1’(0x31) : Work

補足

Example:

```
#include “kme_lib.h”

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

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

  set_kxs86(day_info);

}
```

set_kxs87

要約　　int set_kxs87(int pdop_thresh)

　　int pdop_thresh : PDOP 値（１～１０）

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()と同じ処理を行う。

　　set_kxs87 を 5 以下に設定すると 2 次元測位解を採用しません。精度は上がりますが測位時間が長くなることがあります。(ﾌｧｰﾑ Ver1.20 以降)

get_kxs87

要約　　int get_kxs87(void)

解説　　GPS位置情報をフィルタする時のPDOPのしきい値を獲得する。

戻値　　PDOP 値（１～１０）

補足　　get_pdop_th()と同じ処理を行う。

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_kxs90

要約　　int set_kxs90(unsigned char value, unsigned char mode)

unsigned char value : resolution of analog value
 0: 8bit
 1: 10bit

unsigned char mode : input mode
 0: 0-5V
 1: 0-15V
 2: 4-20mA

解説　　ｱﾅﾛｸﾞ入力のAD変換解像度と入力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_kxs90

要約　　void get_kxs90(unsigned char *value, unsigned char *mode)

unsigned char *value : resolution of analog value
0= 8bit
1= 10bit

unsigned char *mode : input mode
 0: 0-5V
 1: 0-15V
 2: 4-20mA

解説　　ｱﾅﾛｸﾞ入力の AD 変換解像度と入力ﾓｰﾄﾞ設定値を獲得する

戻値　　なし

補足

set_kxs91

要約　　int set_kxs91(unsigned char value)

int value :
0: 出力しない
1: 出力する

解説　　オーブコム衛星の軌道情報を受信した場合、DTEへの出力設定

戻値　　0: 設定成功
1: 設定失敗（ﾊﾟﾗﾒｰﾀｴﾗｰ）

補足

get_kxs91

要約　　unsigned char get_kxs91(void)

解説　　オーブコム衛星の軌道情報の DTE 出力設定値を獲得する

戻値　　0: 出力しない
1: 出力する

補足

set_kxs94

要約　　int set_kxs94(unsigned char port_no, unsigned char direction)

unsigned char port_no : digital I/O port number (0-1)
 0: DIO1
 1: DIO2

unsigned char direction : port direction
 0: 入力ﾎﾟｰﾄ
 1: 出力ﾎﾟｰﾄ

解説　　ﾃﾞｼﾞﾀﾙ入出力ﾎﾟｰﾄの入出力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_kxs94

要約　　int get_kxs94(void)

解説　　ﾃﾞｼﾞﾀﾙ入出力ﾎﾟｰﾄの入出力ﾓｰﾄﾞ設定値を獲得する

戻値　　bit0 : DIO1　(0:入力ﾎﾟｰﾄ、1:出力ﾎﾟｰﾄ)
bit1 : DIO2

補足

set_kxs95

要約　　int set_kxs95(unsigned char sw)

int sw : Selection of
0= 新ｼﾘｱﾙｲﾝﾀｰﾌｪｲｽ(Ver.6.0以降)仕様書に準拠するStatus Code
1= 旧ｼﾘｱﾙｲﾝﾀｰﾌｪｲｽ(Ver.5.01 以前)仕様書に準拠する Status Code

解説　　ﾌﾟﾛﾄｺﾙﾓｰﾄﾞのLL ACK ｽﾃｰﾀｽｺｰﾄﾞを選択する

戻値　　0: 設定成功
1: 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足　　新ｼﾘｱﾙｲﾝﾀｰﾌｪｲｽ(Ver.6.0以降)仕様書に準拠する場合
ｽﾃｰﾀｽｺｰﾄﾞ(0-7)
0:エラーなし
1:バッファが利用不可‐30 秒待って再び送信
2:パケット拒否‐チェックサム不正
3:パケット拒否‐パラメータ不正
4:パケット拒否‐キューの容量オーバー
5:パケット拒否‐フォーマット不正
6:パケット拒否‐異常パケットタイプ
7:パケット拒否‐パケットシーケンス番号の重複

旧ｼﾘｱﾙｲﾝﾀｰﾌｪｲｽ(Ver.5.01以前)仕様書に準拠する場合
ｽﾃｰﾀｽｺｰﾄﾞ(0-1)
0:エラーなし
1:バッファが利用不可‐30秒待って再び送信

get_kxs95

要約　　unsigned char get_kxs95(void)

解説　　プロトコルモードの LL ACK ｽﾃｰﾀｽｺｰﾄﾞ指定値を獲得する

戻値　　0= 新ｼﾘｱﾙｲﾝﾀｰﾌｪｲｽ(Ver.6.0以降)仕様書に準拠するStatus Code
1= 旧ｼﾘｱﾙｲﾝﾀｰﾌｪｲｽ(Ver.5.01 以前)仕様書に準拠する Status Code

補足

set_kxs96

要約　　int set_kxs96(unsigned char port)

unsigned char port : port
0: ｼﾘｱﾙﾎﾟｰﾄ1 (ﾒｲﾝ)
1: ｼﾘｱﾙﾎﾟｰﾄ2 (ｻﾌﾞ)

解説　　ｵｰﾌﾞｺﾑｼﾘｱﾙﾘﾝｸ通信ﾓｰﾄﾞをｻﾎﾟｰﾄするRS232Cﾎﾟｰﾄを指定する

戻値　　0: 設定成功
1: 設定失敗 (ﾊﾟﾗﾒｰﾀｴﾗｰ)

補足

get_kxs96

要約　　int get_kxs96(void)

解説　　ｵｰﾌﾞｺﾑｼﾘｱﾙﾘﾝｸ通信ﾓｰﾄﾞをｻﾎﾟｰﾄする RS232C ﾎﾟｰﾄの設定値を獲得する

戻値　　0: ｼﾘｱﾙﾎﾟｰﾄ1 (ﾒｲﾝ)
1: ｼﾘｱﾙﾎﾟｰﾄ2 (ｻﾌﾞ)

補足

set_kxd01

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

unsigned char m: ﾎﾟｰﾄ (0:入出力ﾎﾟｰﾄ1, 1:入出力ﾎﾟｰﾄ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

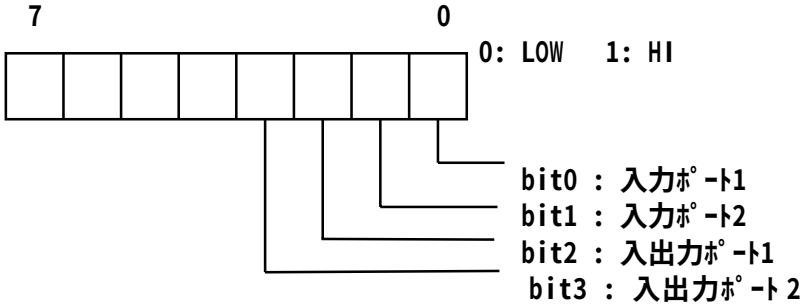
補足　　出力設定は set_kxs94 にて入出力ﾎﾟｰﾄを出力ﾓｰﾄﾞにしていないと無効です。

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　　本関数にて入力ﾎﾟｰﾄの情報も取得できます。

set_kxd02 はｻﾎﾟｰﾄされていません。

get_kxd02

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

unsigned char ch_no : 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: ｱﾅﾛｸﾞﾎﾟｰﾄ 3
 4- 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_kxm01

要約　　int set_kxm01(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　　固定ﾒｯｾｰｼﾞ設定値を獲得する

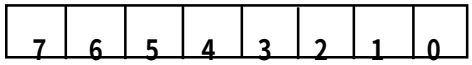
戻値　　固定ﾒｯｾｰｼﾞへのﾎﾟｲﾝﾀ (最大 200 ﾊﾞｲﾄ)

補足

set_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: O/Rｲﾝﾃﾞｨｹｰﾀ (1 - 8)
char d: 送信ﾃﾞｰﾀ



bit0: 測位情報
bit1: 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: 予備

int n: 時刻番号 (1 - 6)

解説　　時刻指定送信を設定する

戻値　　0: 設定成功
1: 設定失敗 (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 (0 - 44640)[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: 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入力状態

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

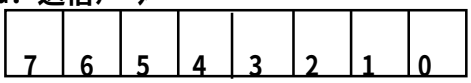
戻値　　0: 未設定
1: 設定

補足

set_kxb05

要約　　int set_kxb05(KXB05_CMND *pkxb05_cmnd, int p)

KXB05_CMND *pkxb05_cmnd
 int tr: Trigger (0:OFF, 1:LOW -> HI, 2:HI->LOW, 3:HI<->LOW)
 int a: O/Rｲﾝﾃﾞｨｹｰﾀ(1 - 8)
 char d: 送信ﾃﾞｰﾀ



bit0: 測位情報
bit1: 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状態

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

解説　　I/O 状態変化報知を設定する

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

補足

get_kxb05

要約　　int get_kxb05(KXB05_CMND *pkxb05_cmnd, int n)

KXB05_CMND *pkxb05_cmnd
int n : Digital input port No. (0, 1)

解説　　I/O状態変化報知の設定値を獲得する

戻値　　0: 未設定
1: 設定

補足

set_kxb00

要約　　void set_kxb00(void)

解説　　KXB01,02,05 ｺﾏﾝﾄﾞの設定を解除する

戻値　　なし

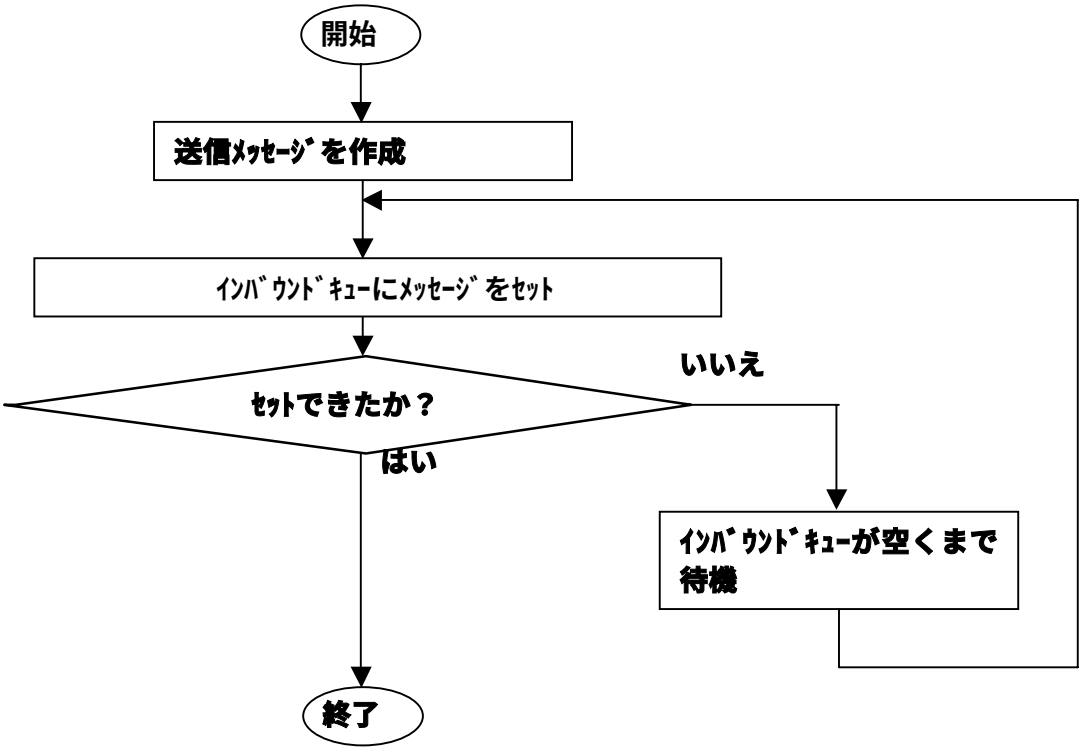
補足

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

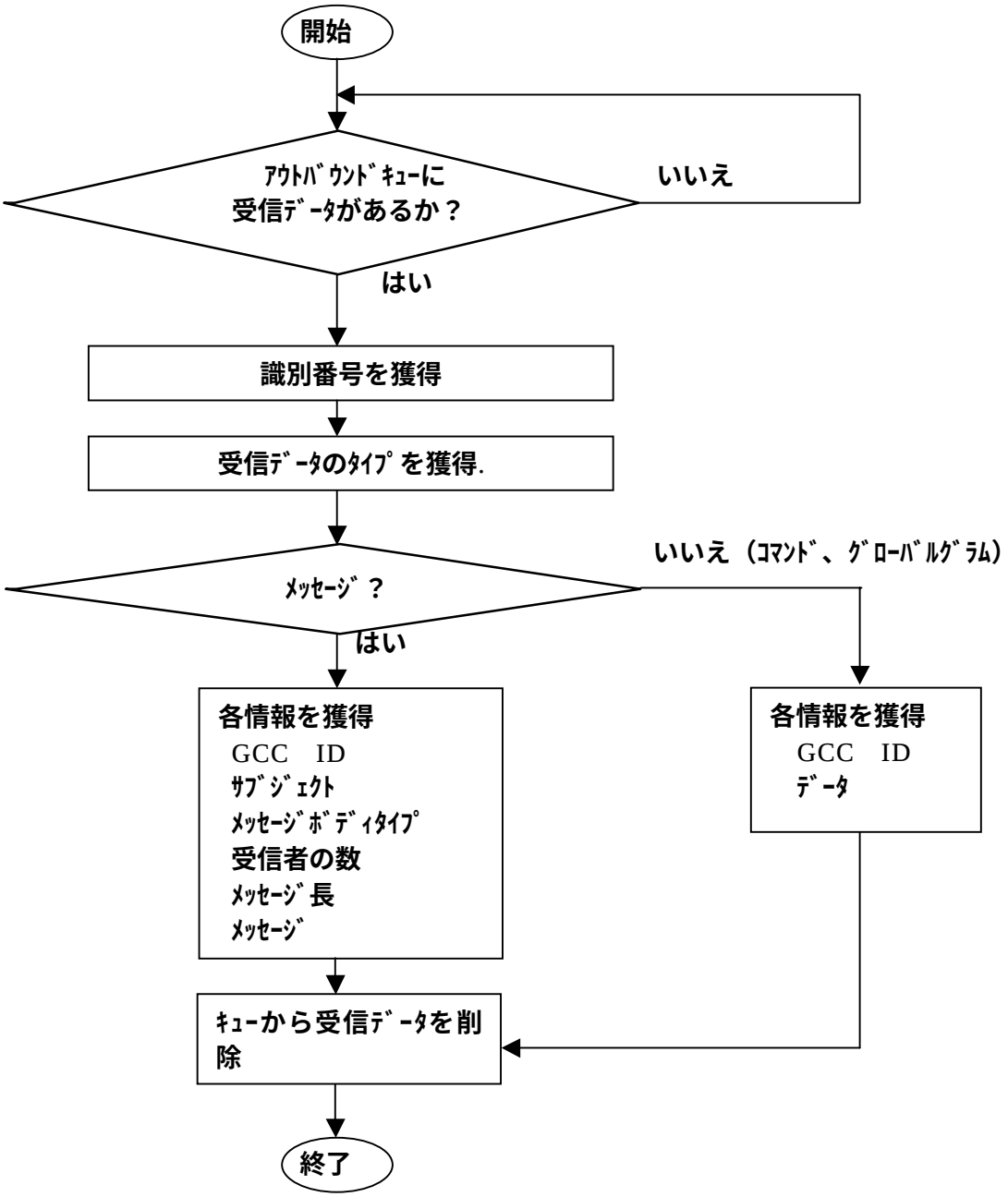
1. 送信ﾒｯｾｰｼﾞを作成する
2. ﾒｯｾｰｼﾞをｲﾝﾊﾞｳﾝﾄﾞｷｭｰにｾｯﾄする為に、“ req_send_ib_message() ” をｺｰﾙする。
3. ﾒｯｾｰｼﾞがｲﾝﾊﾞｳﾝﾄﾞｷｭｰにｾｯﾄされたかをﾁｪｯｸする。

   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。



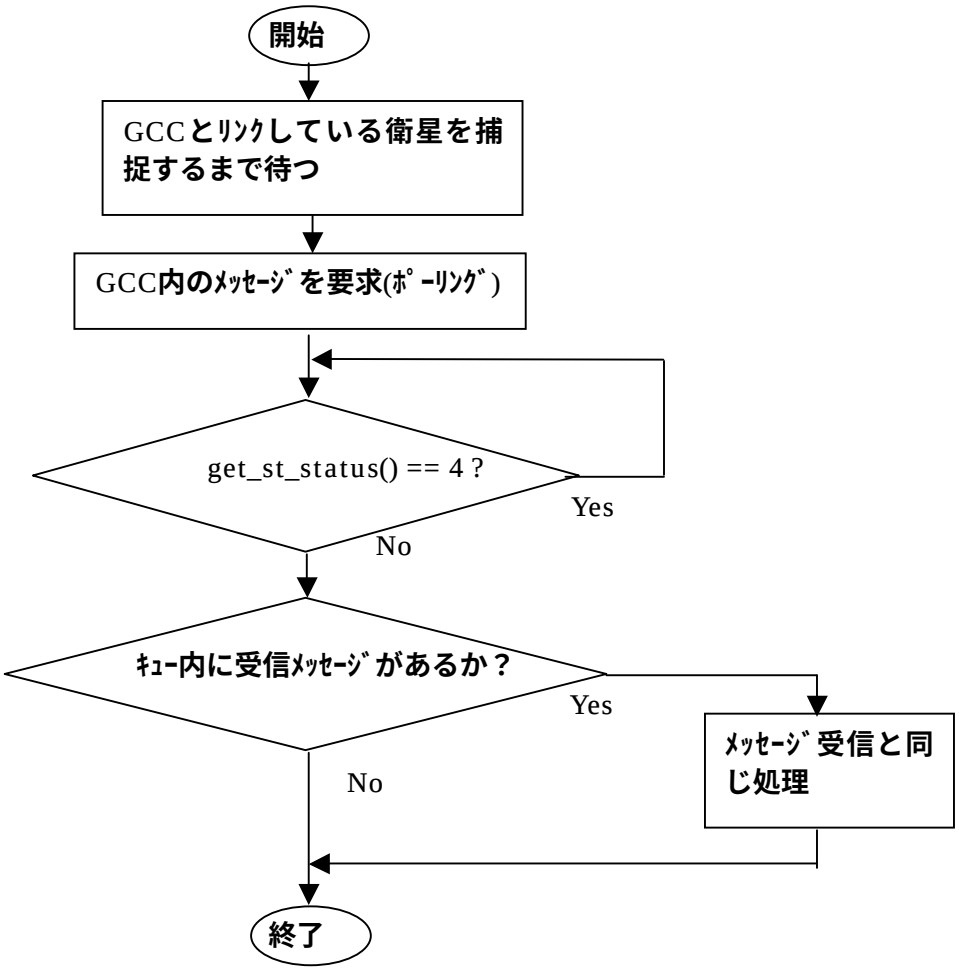
7.2. GCCからのﾒｯｾｰｼﾞ受信

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



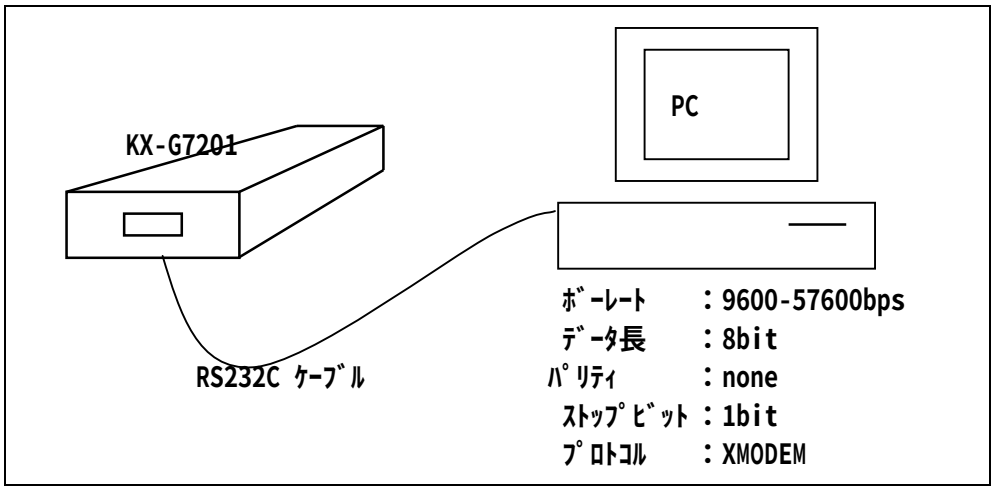
7.3. GCC内のﾒｯｾｰｼﾞを取り出す

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。

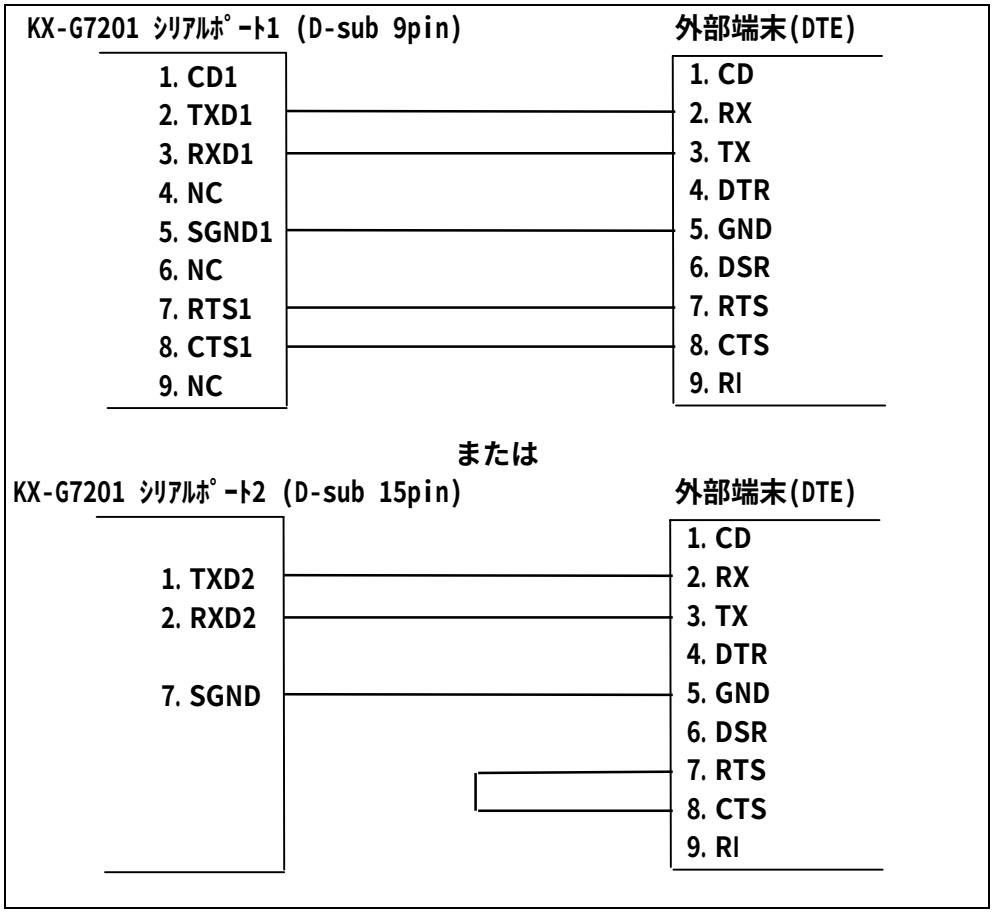


8. ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｸﾞﾗﾑのｲﾝｽﾄｰﾙ方法

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(0x060000)ｲﾝｽﾄｰﾙします。



8.1. 接続



8.2. インストール方法

１．　端末とﾊﾟｿｺﾝをRS232Cｹｰﾌﾞﾙで接続し、ﾊｲﾊﾟｰﾀｰﾐﾅﾙを起動してください。端末には、シリアルポートが２チャンネルあります。何れのポートを用いてもインストールが可能です。但し、チャンネル２については通信速度57,600bpsは使用できません。

２．　CTRL +KXORBと入力してｺﾏﾝﾄﾞﾓｰﾄﾞに入ります。

３．　CTRL + UALDRと入力してｲﾝｽﾄｰﾙﾓｰﾄﾞに入ります。
PCに以下のﾒｯｾｰｼﾞが表示されます。
> [User Application Software]:*** Installation Mode ***
> Select SC's RS232C bps (1:9600/ 2:19200/ 3:38400/ 4:57600/ 5:Exit)”

４．　ｲﾝｽﾄｰﾙの通信速度を選択し、番号(1-4)を入力してください。
PCに以下のﾒｯｾｰｼﾞが表示されます。
> Match the modem bps to this, then reconnect.

５．　ﾊｲﾊﾟｰﾀｰﾐﾅﾙの通信速度を選択した通信速度に変更し、ﾊｲﾊﾟｰﾀｰﾐﾅﾙを再接続してください。PCに以下のﾒｯｾｰｼﾞが表示されます。
> [User Application Software]:*** Installation Mode ***
（この操作は、通信速度選択後１分以内に、完了してください。）

６．　PCからﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｸﾞﾗﾑを送信します。
ﾊｲﾊﾟｰﾀｰﾐﾅﾙの転送(T)メニューから、ﾌｧｲﾙの送信(S)を選び、ｲﾝｽﾄｰﾙするﾌｧｲﾙ（*.up）を指定し、XMODEMまたは1K XMODEMプロトコルで送信します。
ｲﾝｽﾄｰﾙ終了後、PCに以下のﾒｯｾｰｼﾞが表示されｲﾝｽﾄｰﾙﾌｧｲﾙのﾁｪｯｸを行います。
> [User Application Software]:Now, Checking User Application Software….
ｲﾝｽﾄｰﾙのﾁｪｯｸが完了したら以下のﾒｯｾｰｼﾞが表示されます。
> Installation was Completed. Checksum OK.
Reset power switch”

７．　端末の電源を入れ直してください。

8.3. ユーザーアプリケーションの削除

１．　端末とﾊﾟｿｺﾝをRS232Cｹｰﾌﾞﾙで接続し、ﾊｲﾊﾟｰﾀｰﾐﾅﾙを起動します。
２．　CTRL +KXORBと入力してｺﾏﾝﾄﾞﾓｰﾄﾞに入ります。
３．　CTRL + UADLTと入力します。
PCに以下のﾒｯｾｰｼﾞが表示されます。
“User application area Initialize OK ? <Y/N>”
４．“Y“を押していください。
“Initialized : OK”
“Reset power switch”
５．　端末の電源を入れ直してください。

8.4. ユーザーアプリケーションの動作状態の確認

１． 端末とﾊﾟｿｺﾝをRS232Cｹｰﾌﾞﾙで接続し、ﾊｲﾊﾟｰﾀｰﾐﾅﾙを起動します。
２． CTRL +KXORBと入力してｺﾏﾝﾄﾞﾓｰﾄﾞに入ります。
３． KXCHKと入力すると、以下のような自己診断結果が表示されます。

```
>KXCHK

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

“USER APPL”の表示結果は、ﾕｰ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”と表示します。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“NONE ACTIVE”を表示します。ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝﾗｲﾌﾞﾗﾘのﾊﾞｰｼﾞｮﾝ番号は< >の中に表示されます。

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、作成者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ます。端末とﾊﾟｿｺﾝをRS232Cｹｰﾌﾞﾙで接続して、通信ｿﾌﾄ（例えば “ﾊｲﾊﾟｰﾀｰﾐﾅﾙ”)を起動するにより端末からいろいろなﾃﾞﾊﾞｯｸﾞ情報を獲得できます。

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ます。

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の時に有効となります。ﾃﾞﾊﾞｯｸﾞﾓｰﾄﾞをONにする為にはKXS70ｺﾏﾝﾄﾞを使用してください。.
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点灯します。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ます。

9.1. メモリーダンプ

ﾒﾓﾘｰ内容を16進で表示します。

入力ｺﾏﾝﾄﾞ：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みます。

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します。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3200ﾊﾞｲﾄです。.

入力ｺﾏﾝﾄﾞ：stack

<例> **> stack**

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

9.4. ブレークポイント

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ます。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ります。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/無効に設定できます。

<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

GCCからのﾒｯｾｰｼをﾞ受信したように、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ます。
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ます。

1) 通常のOBﾒｯｾｰｼﾞ

<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

[ﾒｯｾｰｼﾞ] ：ﾒｯｾｰｼﾞ本体(最大256ﾊﾞｲﾄ)

<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
```

2) ﾕｰｻﾞｰｺﾏﾝﾄﾞ

<入力ｺﾏﾝﾄﾞ>

obreg ,, CMD,< H>[data1] [data2] [data3] [data4] [data5]<CR><LF>

[data1-5] ：ﾊﾞｲﾅﾘｰﾃﾞｰﾀ。

<例> ﾃﾞｰﾀ 0x41,0x42,0x43,0x44,0x45

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

3) ｱｳﾄﾊﾞｳﾝﾄﾞｸﾞﾛｰﾊﾞﾙｸﾞﾗﾑ

<入力ｺﾏﾝﾄﾞ>

obreg [ﾚｼﾋﾟｴﾝﾄ1],,GLB,<T>[ﾒｯｾｰｼﾞ]<CR><LF>

<例> 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,,GLB,<T>12345<CR><LF>
```

10. 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、228000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。
  ”sprintf() ” の代替関数として、 print_double(), orb_sprintf1(),,, orb_sprintf7()を用意しています。

- 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、PCの通信ﾓｰﾄﾞを以下の様に設定した後、“ 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。また、端末の通信モードもDefault設定に戻っています。

  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

11. KX-G7100/G7101用プログラムとの互換性

11.1. UPファイル

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

11.2. ソースファイル

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

(1) セクションの記述方法が異なる。
(例：KX-G720x)

```
#pragma section USR
#pragma section
```

(例：KX-G71xx)

```
#pragma _section B=USER
#pragma _section B
```

(2) Endian(データのメモリへの格納方法)が異なる。
(メモリーを直接参照するようなプログラムでは修正が必要。)

(3) KME-library
一部の関数に付いて、廃止・変更・追加が行われています。

(4) 定数・関数名の文字数制限
KX-G7201Nでは、33文字以降はコンパイル時に無視されます。

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

11.3. プログラムサイズ

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

11.4. KMEライブラリー内の関数の追廃リスト

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

(1) Power Control

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

(2) Measurement

Library Function	Description
get_pos_quality_ind() get_pass_quan()	測位品質を獲得する ドプッラー測位で使用したパスの数を獲得する

(3) Status Information

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

(4) Other Functions

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

(5) KX Command

Library Function		Description
set_kxs10()	get_kxs10()	ﾎﾟｰﾘﾝｸﾞ複数ﾚﾎﾟｰﾄのｲﾝﾀｰﾊﾞﾙ
set_kxs11()	get_kxs11()	ﾎﾟｰﾘﾝｸﾞ複数ﾚﾎﾟｰﾄの数
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
set_kxs13()	get_kxs13()	ﾎﾟｰﾘﾝｸﾞ複数測位 ﾚﾎﾟｰﾄの数
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
set_kxs20()	get_kxs20()	測位結果の有効時間
set_kxs21()	get_kxs21()	測位結果の最小有効ｸｫﾘﾃｨｲﾝﾃﾞｨｹｰﾀ
set_kxs22()	get_kxs22()	軌道要素の有効時間
set_kxs38()	get_kxs38()	ﾊﾟﾜｰﾀﾞｳﾝﾐﾆﾏﾑｲﾝﾀｰﾊﾞﾙ
set_kxs49()	get_kxs49()	UTC時間補正値
set_kxs59()	get_kxs59()	送信履歴自動送信モード
set_kxs65()	get_kxs65()	測位ﾚﾎﾟｰﾄ送信
set_kxs66()	get_kxs66()	ドップラー測位可否
set_kxs73()	get_kxs73()	DGPS用のFM周波数
set_kxs76()	get_kxs76()	衛星飛来時刻を計算する時の衛星の最低仰角
set_kxp01()	get_kxp01()	入力ﾎﾟｰﾄと出力ﾎﾟｰﾄのﾘﾝｸ
set_kxa01()	get_kxa01()	時刻指定送信(KXA01ｺﾏﾝﾄﾞ)
set_kxa02()	get_kxa02()	ｲﾝﾀｰﾊﾞﾙ送信（KXA02ｺﾏﾝﾄﾞ）
set_kxa03()	get_kxa03()	衛星飛来送信（KXA03ｺﾏﾝﾄﾞ）
set_kxa04()	get_kxa04()	ﾊﾞｲﾄﾓｰﾄﾞ設定（KXA04ｺﾏﾝﾄﾞ）
set_kxa05()	get_kxa05()	入力ﾎﾟｰﾄ変化で送信（KXA05ｺﾏﾝﾄﾞ）
set_kxb03()	get_kxb03()	衛星飛来送信

11.4.2. KX-G7100/G7101用ライブラリーに追加された関数

(1)Serial Interface

Library Function	Description
chk_rx_buff_port()	RS232Cの受信ﾊﾞｯﾌｧをﾁｪｯｸする
get_rx_data_port ()	RS232Cの受信ﾊﾞｯﾌｧからﾃﾞｰﾀを獲得する
chk_tx_ready_port ()	RS232Cの送信ﾊﾞｯﾌｧが満杯かをﾁｪｯｸする
set_tx_data_port ()	RS232Cの送信ﾊﾞｯﾌｧにﾃﾞｰﾀをｾｯﾄする
cts_act_port ()	CTSﾎﾟｰﾄをｱｸﾃｨﾌﾞにする
cts_neg_port ()	CTSﾎﾟｰﾄをﾈｶﾞﾃｨﾌﾞにする
chk_cd_port ()	ｼﾘｱﾙﾎﾟｰﾄのCD信号の状態をﾁｪｯｸする
chk_rts_port ()	ｼﾘｱﾙﾎﾟｰﾄのRTS信号の状態をﾁｪｯｸする

(2) I/O Functions

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

(3) KX Command

Library Function		Description
set_kxs43_port()	get_kxs43_port()	RS232C通信ﾓｰﾄﾞ
set_kxs90()	get_kxs90()	ｱﾅﾛｸﾞ入力のAD変換解像度と入力ﾓｰﾄﾞ
set_kxs91()	get_kxs91()	ｵｰﾌﾞｺﾑ衛星の軌道情報のDTEへの出力
set_kxs94()	get_kxs94()	ﾃﾞｼﾞﾀﾙ入出力ﾎﾟｰﾄの入出力ﾓｰﾄﾞ
set_kxs95()	get_kxs95()	ﾌﾟﾛﾄｺﾙﾓｰﾄﾞのﾊﾟｹｯﾄﾀｲﾌﾟ
set_kxs96()	get_kxs96()	ｵｰﾌﾞｺﾑｼﾘｱﾙﾘﾝｸ通信ﾓｰﾄﾞのｻﾎﾟｰﾄRS232C選択
set_kxb05()	get_kxb05()	I/O状態変化報知

12. ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝの開発環境

12.1. ハードウェア環境

パソコンは以下の環境が必要です。

機種	PC-ATおよびその互換機
CPU	80386以上
ﾒﾓﾘｰ	5.6MB 以上
ﾊｰﾄﾞﾃﾞｨｽｸ	35MB以上

12.2. ソフトウェア

以下のソフトウエアおよびドキュメントが必要です。.

1. H8S,H8/300ｼﾘｰｽﾞC/C++ｺﾝﾊﾟｲﾗｰ Ver.2.0E PS008CAS3-MWR（日立製）
   (CD-ROMで供給され、電子文書版のﾏﾆｭｱﾙが含まれています。)
2. kme_lib.h - KMEﾗｲﾌﾞﾗﾘｰﾍｯﾀﾞｰﾌｧｲﾙ
3. kmelib.obj - KMEﾗｲﾌﾞﾗﾘｰ
4. user_main.obj -- ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝ ｽﾀｰﾄｱｯﾌﾟﾗｲﾌﾞﾗﾘｰ
5. ms2bin2.exe -- ﾌｧｲﾙﾀｲﾌﾟ変換用ﾕｰﾃｨﾘﾃｨﾌﾟﾛｸﾞﾗﾑ
6. bin2boot2.exe -- ﾌｧｲﾙﾀｲﾌﾟ変換用ﾕｰﾃｨﾘﾃｨﾌﾟﾛｸﾞﾗﾑ
7. ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｸﾞﾗﾐﾝｸﾞｶﾞｲﾄﾞ(本ﾏﾆｭｱﾙ)

また、サンプルプログラムを準備しています。

8. sample1.c -- ｻﾝﾌﾟﾙﾌﾟﾛｸﾞﾗﾑ
9. sample1.bat -- ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝ作成用ﾊﾞｯﾁﾌｧｲﾙ
10. sample1.lnk -- ﾘﾝｸﾌｧｲﾙ
11. compile.cmd -- ｺﾝﾊﾟｲﾙｻﾌﾞｺﾏﾝﾄﾞﾌｧｲﾙ
12. Kankyo.bat -- 環境変数設定ﾊﾞｯﾁﾌｧｲﾙ

13. ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝｻﾝﾌﾟﾙﾌﾟﾛｸﾞﾗﾑｶﾞｲﾄﾞ

12.3. コンパイラーのインストール

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

12.4. ディレクトリーの作成とファイルのコピー

\himのｻﾌﾞﾃﾞｨﾚｸﾄﾘｰとして、tmpおよびuserﾃﾞｨﾚｸﾄﾘを作成します。
tmpﾃﾞｨﾚｸﾄﾘは、ｺﾝﾊﾟｲﾙ中に一時的に作成される中間ﾌｧｲﾙが使用します。
userﾃﾞｨﾚｸﾄﾘは、ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝの開発、ｺﾝﾊﾟｲﾙ・ﾘﾝｸ作業用です。

次のﾌｧｲﾙは、c:\him\Toolchains\Hitachi\ch38\V20c\Binにｺﾋﾟｰします。

- ms2bin2.exe -- ﾌｧｲﾙﾀｲﾌﾟ変換用ﾕｰﾃｨﾘﾃｨﾌﾟﾛｸﾞﾗﾑ
- bin2boot2.exe -- ﾌｧｲﾙﾀｲﾌﾟ変換用ﾕｰﾃｨﾘﾃｨﾌﾟﾛｸﾞﾗﾑ

次のﾌｧｲﾙは、c:\him\userにｺﾋﾟｰします。

- kme_lib.h - KMEﾗｲﾌﾞﾗﾘｰﾍｯﾀﾞｰﾌｧｲﾙ
- kmelib.obj - KMEﾗｲﾌﾞﾗﾘｰ
- user_main.obj -- ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝ ｽﾀｰﾄｱｯﾌﾟﾗｲﾌﾞﾗﾘｰ

ｻﾝﾌﾟﾙﾌﾟﾛｸﾞﾗﾑは必要に応じてc:\him\userにｺﾋﾟｰします。

- sample1.c -- ｻﾝﾌﾟﾙﾌﾟﾛｸﾞﾗﾑ
- sample1.bat -- ﾕｰｻﾞｰｱﾌﾟﾘｹｰｼｮﾝ作成用ﾊﾞｯﾁﾌｧｲﾙ
- sample1.lnk -- ﾘﾝｸﾌｧｲﾙ
- compile.cmd -- ｺﾝﾊﾟｲﾙｻﾌﾞｺﾏﾝﾄﾞﾌｧｲﾙ

12.5. PCの環境変数の設定

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

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

13. 変更履歴

Ver1.0
- KX-G7201用新規作成

Ver1.1 (2000.9.1)
- シリアルポートの名称をチャンネル１、チャンネル２に統一
- コンパイラーの品番・内容物の情報を修正、コンパイラーのバージョン指定を追加
- Set_kxs68/ get_kxs68のパラメータ記述を修正
- Set_kxs83/ get_kxs83のパラメータ記述を修正

Ver1.21 (2000.12.26)
- set_kxs20/ get_kxs20を削除
- set_kxs59/ get_kxs59を削除
- set_kxb03/ get_kxb03を削除
- set_tx_data_portの引数の順序を訂正
- orb_sprintf0を削除
- コンパイラーのバージョンを2.0から2.0Eへ変更
- メモリーマップ関連アドレスの修正

Ver1.3 (2001. 2.23)
- set_tx_data, set_tx_data_portの説明追加
- set_timerの戻値訂正
- get_ob_typeの戻値訂正
- set_pin_codeの要約、戻値訂正
- set_ncc_search_modeの戻値訂正
- globalgram_pollingの説明訂正
- clear_active_msgの戻値、補足追加
- clear_mha_ref_num_msgの戻値、補足訂正
- req_pos_report_to_nccの解説、戻値訂正
- req_send_ib_globalgramの要約追加
- req_send_ib_enh_globalgramの要約訂正
- req_send_ob_messageの戻値訂正
- get_ob_msg_subject_indの要約訂正
- get_ob_msg_msg_lenの要約訂正
- get_ob_user_commandの要約訂正
- get_ob_globalgramの関数名、要約訂正
- go_sleep_timeの補足訂正
- get_rx_dataの戻値説明追加
- get_rx_data_portの要約、戻値訂正
- get_digital_portの要約、戻値訂正
- set_digital_portの要約訂正
- get_adcon_data_10bitの要約訂正
- chk_pos_calc_resultの補足削除

- set_pdop_thの補足訂正
- get_pdop_thの補足訂正
- break_pointの要約、戻値、補足訂正
- chk_break_pointの戻値訂正
- req_apl_sat_predictの補足追加
- chk_apl_sat_predict_resultの要約、補足訂正
- get_apl_sat_predict_timeの補足追加
- get_sys_infoの要約訂正
- set_no_wait_for_power_saveの削除
- get_kxs25の要約訂正
- get_kxs27の要約訂正
- set_kxs33の要約訂正
- get_kxs33の要約、戻値訂正
- get_kxs46の要約訂正
- set_kxs50の要約訂正
- get_kxs50の戻値訂正
- set_kxs60の要約、戻値、補足訂正
- set_kxs61の要約訂正
- get_kxs61の要約訂正
- set_kxs69の戻値訂正
- set_kxs81の要約訂正
- get_kxs81の要約訂正
- get_kxd02の要約訂正

Ver1.4 (2001. 7.16)

- get_unget_ob_id_for_userの補足追加、サンプルプログラム修正
- get_adcon_dataの補足追加
- get_rx_dataの補足修正
- set_tx_dataの補足修正
- cts_actの補足削除
- cts_negの補足削除
- chk_rx_buff_portの補足修正
- get_rx_buff_portの補足修正
- chk_tx_ready_portの補足修正
- set_tx_data_portの補足修正
- cts_act_port, cts_neg_port, chk_cd_port, chk_rts_port関数説明削除
- get_digital_portの戻り値図、補足修正
- set_digital_portの要約、補足修正
- get_adcon_dataの要約修正
- get_adcon_data_10bitの要約修正
- set_pdop_thの補足追加
- set_kxs47の要約修正
- set_kxs55の要約修正
- set_kxs87の補足追加
- get_kxd02の要約修正
- ｱｳﾄﾊﾞｳﾝﾄﾞｷｭｰへのﾒｯｾｰｼﾞﾏﾆｭｱﾙ登録の説明追加

Ver1.5 (2003.1.6)

- chk_ib_txの引数、戻り値の説明修正
- go_waitの説明修正
- chk_tx_readyの送信ﾊﾞｯﾌｧｻｲｽﾞ修正
- cts_act，cts_negの補足説明追加
- get_active_mha_ref_numの補足説明追加
- get_chk_errorsの補足説明追加
- chk_apl_sat_predict_resultの関数名誤記修正
- get_sys_infoの引数修正
- os_monitorの説明修正
- get_kxs06，get_kxs07のﾌﾟﾛｸﾞﾗﾑ例の誤記修正
- set_kxs16，get_kxs16の引数・戻り値範囲修正
- set_kxs30，get_kxs30の引数範囲修正
- set_kxs32の引数説明修正
- set_kxs48の関数型修正
- get_kxs63の関数型修正
- set_kxs80，get_kxs80の説明修正
- set_kxs84，get_kxs84の説明修正
- set_kxs86，get_kxs86の説明修正
- set_kxs88，get_kxs88の説明追加