お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

RENESAS

保守／廃止

MOS 集積回路
MOS Integrated Circuit

μPD77533

GPS 信号処理 LSI

μPD77533 は，スナップトラック社（SnapTrack Inc.）のワイヤレス・アシスト GPS（Global Positioning System）を実現する，GPS 端末用 GPS 信号処理 LSI です。

ワイヤレス・アシスト GPS の端末側での測位機能として，擬似距離計算，マルチパス検出等を行い，ネットワーク側のロケーション・サーバとの連携による測位を実現します。

特　　徴

○ワイヤレス・アシスト GPS の測位機能

- 擬似距離計算
- マルチパス検出
- サーバとの連携による高精度，高感度，高速な位置測定

○SRAM（モバイル用途 RAM）インタフェース，ホスト・シリアル・インタフェース，A/D コントロール・インタフェース，RF コントロール・インタフェース内蔵

○電源電圧

- 内部システム電源　　：1.425～1.65 V
- I/O 端子電源　　：2.7～3.6 V

オーダ情報

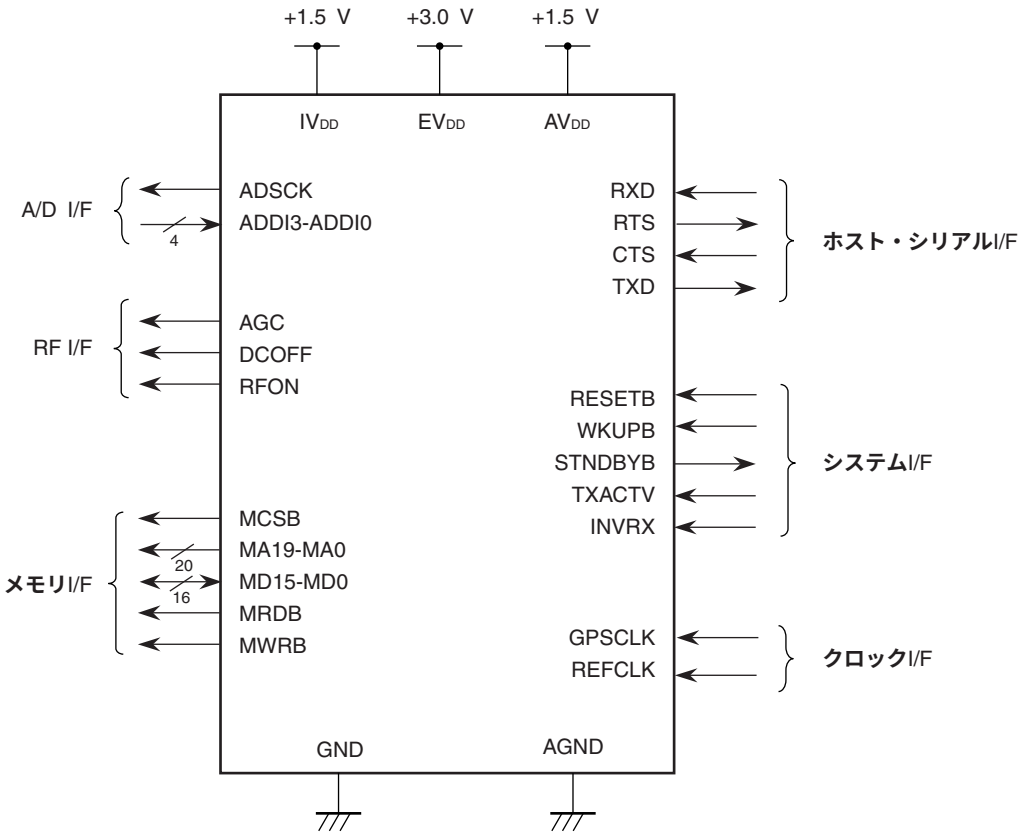
オーダ名称	パッケージ
μPD77533S1-YHC	108 ピン・プラスチック・ファインピッチ BGA（11×11）

本資料の内容は，予告なく変更することがありますので，最新のものであることをご確認の上ご使用ください。

資料番号　S15671JJ4V0DS00（第 4 版）
発行年月　May　2002 NS　CP(K)

本文欄外の★印は，本版で改訂された主な箇所を示しています。

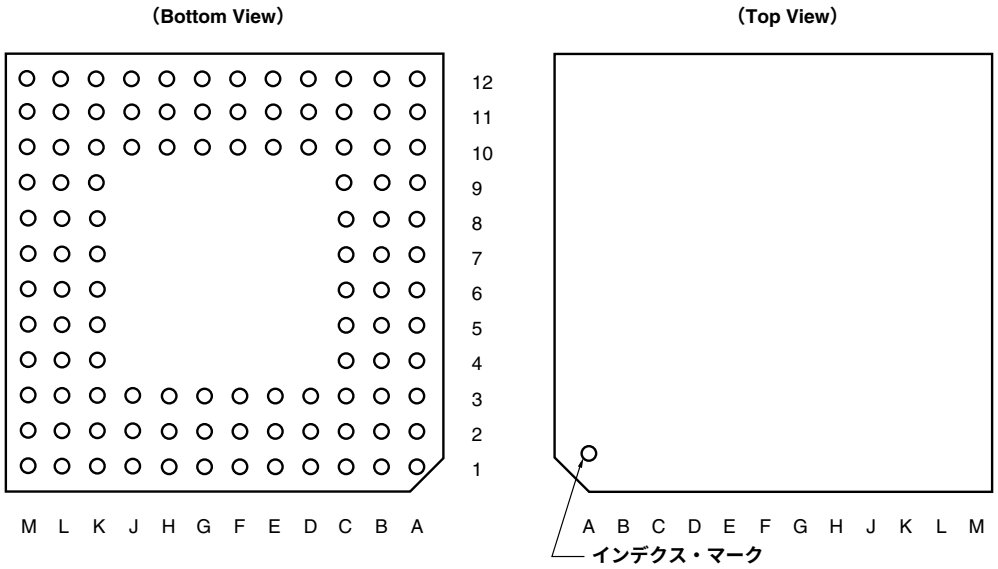
端子構成図


+1.5 V
+3.0 V
+1.5 V
IVDD
EVDD
AVDD
A/D I/F
ADSCK
ADDI3-ADDI0
4
RF I/F
AGC
DCOFF
RFON
メモリI/F
MCSB
MA19-MA0
20
MD15-MD0
16
MRDB
MWRB
RXD
RTS
CTS
TXD
ホスト・シリアルI/F
RESETB
WKUPB
STNDBYB
TXACTV
INVRX
システムI/F
GPSCLK
REFCLK
クロックI/F
GND
AGND

端子接続図（Top View）

μPD77533S1-YHC

・108ピン・プラスチック・ファインピッチBGA（11×11）



端子番号	端子名	端子番号	端子名	端子番号	端子名	端子番号	端子名
A1	NC	C4	GND	G1	MA4	K10	ADSCK
A2	GND	C5	MWRB	G2	MA3	K11	EV_{DD}
A3	MA17	C6	I.C.	G3	EV_{DD}	K12	ADDI3
A4	MA19	C7	MRDB	G10	I.C.	L1	NC
A5	EV_{DD}	C8	STNDBYB	G11	EV_{DD}	L2	MD13
A6	IV_{DD}	C9	WKUPB	G12	NC	L3	MD11
A7	INVRX	C10	GND	H1	MA2	L4	MD9
A8	RESETB	C11	GND	H2	MA0	L5	MD6
A9	GND	C12	REFCLK	H3	MA1	L6	MD3
A10	AGND	D1	MA11	H10	RFON	L7	MD0
A11	NC	D2	MA12	H11	AGC	L8	RXD
A12	NC	D3	MA10	H12	DCOFF	L9	CTS
B1	GND	D10	GND	J1	MD15	L10	TXACTV
B2	MA15	D11	GPSCLK	J2	EV_{DD}	L11	GND
B3	MA16	D12	I.C.	J3	MD12	L12	GND
B4	MA18	E1	MA8	J10	ADDI0	M1	NC
B5	NC	E2	MA9	J11	ADDI2	M2	GND
B6	GND	E3	MA7	J12	ADDI1	M3	MD10
B7	MCSB	E10	I.C.	K1	GND	M4	MD8
B8	NC	E11	I.C.	K2	MD14	M5	MD5
B9	IV_{DD}	E12	I.C.	K3	GND	M6	GND
B10	AV_{DD}	F1	MA6	K4	MD7	M7	MD1
B11	EV_{DD}	F2	GND	K5	EV_{DD}	M8	IV_{DD}
B12	NC	F3	MA5	K6	MD2	M9	RTS
C1	MA13	F10	GND	K7	MD4	M10	TXD
C2	EV_{DD}	F11	I.C. -RL	K8	GND	M11	NC
C3	MA14	F12	I.C.	K9	GND	M12	NC

端子名称

ADDI3-ADDI0:	AD Data Bus
ADSCK:	AD Converter Clock Output
AGC:	Auto Gain Control
AGND:	Ground for PLL Analog System
AV_{DD}:	Power Supply for PLL Analog System
CTS:	Clear To Send
DCOFF:	DC OFF (DC Trimming)
EV_{DD}:	Power Supply for I/O Pins
GND:	Ground
GPSCLK:	GPS Clock
I.C., I.C.-RL:	Internal Connection
INVRX:	RX Invert
IV_{DD}:	Power Supply for Internal System
MA19-MA0:	SnapShot Memory Address Bus
MCSB:	SnapShot Memory Chip Select
MD15-MD0:	SnapShot Memory Bus
MRDB:	SnapShot Memory Read Strobe
MWRB:	SnapShot Memory Write Strobe
NC:	Non-connection
REFCLK:	Reference Clock
RESETB:	Reset
RFON:	RF Power ON
RTS:	Request To Send
RXD:	Host Serial Data Input
STNDBYB:	Standby
TXACTV:	Tx Active
TXD:	Host Serial Data Output
WKUPB:	Wakeup

目　次

1. 端子機能

1. 1 端子機能の説明

・電　源

端子名称	端子番号	入出力	機　能
IV_{DD}	A6, B9, M8	−	コア用電源（1.425～1.65 V）
EV_{DD}	A5, B11, C2, G3, G11, J2 ,K5, K11	−	I/O 端子用電源（2.7～3.6 V）
GND	A2, A9, B1, B6, C4, C10, C11, D10, F2, F10, K1, K3, K8, K9, L11, L12, M2, M6	−	接地
AV_{DD}	B10	−	PLL アナログ用電源（1.425～1.65 V）
AGND	A10	−	PLL アナログ用接地

・クロック･コントロール

端子名称	端子番号	入出力	機　能
GPSCLK	D11	入力	GPS 基準クロック入力（16.384 MHz） μPD77533 を動作させるためのクロック入力です。
REFCLK	C12	入力	周波数調整基準クロック入力 （PDC : 14.4 MHz，PHS : 19.2 MHz など）

・システム・コントロール

端子名称	端子番号	入出力	機　能
RESETB	A8	入力	内部システム・リセット入力 μPD77533 を初期化します。
WKUPB	C9	入力	ストップ・モード解除入力 WAKEUP 割り込み入力：立ち下がり検出
STNDBYB	C8	出力	スタンバイ状態（ストップ or ホールト） 0：スタンバイ状態 1：動作状態
TXACTV	L10	入力	ADDI データ・マスク・イネーブル信号 0：マスクなし 1：マスクあり
INVRX	A7	入力	RXD 論理反転設定信号 ホスト・シリアル入力の RXD の論理反転を設定する信号です。 0：正転（Non-invert） 1：反転（Invert）

・ホスト・シリアル・インタフェース

端子名称	端子番号	入出力	機　能
RXD	L8	入力	調歩同期シリアル入力
RTS	M9	出力	Request To Send
CTS	L9	入力	Clear To Send
TXD	M10	出力	調歩同期シリアル出力

・RF コントロール・インタフェース

端子名称	端子番号	入出力	機　能
AGC	H11	出力 (3S)	オートゲイン・コントロール 表 5-1　AGC, DCOFF の出力値を参照。
DCOFF	H12	出力 (3S)	DC オフセット・コントロール 表 5-1　AGC, DCOFF の出力値を参照。
RFON	H10	出力	RF パワー・コントロール 0：RF OFF 1：RF ON

・A/D コンバータ・コントロール

端子名称	端子番号	入出力	機　能
ADSCK	K10	出力	AD 変換クロック出力
ADDI3- ADDI0	K12, J11, J12 , J10	入力	AD 変換データ入力（4 ビット）

・メモリ・インタフェース

端子名称	端子番号	入出力	機　能
MCSB	B7	出力	SnapShot メモリ・チップ・セレクト ロウ・アクティブです。
MRDB	C7	出力	SnapShot メモリ・リード・ストローブ ロウ・アクティブです。
MWRB	C5	出力	SnapShot メモリ・ライト・ストローブ ロウ・アクティブです。
MA19-MA0	A4, B4, A3, B3, B2, C3, C1, D2, D1, D3, E2, E1, E3, F1, F3, G1, G2, H1, H3, H2	出力	SnapShot メモリ・アドレス（20 ビット）
MD15- MD0	J1, K2, L2, J3, L3, M3, L4, M4, K4, L5, M5, K7, L6, K6, M7, L7	入出力 (3S)	SnapShot メモリ・データ（16 ビット）

備考　表中の入出力欄に，“ 3S ” を付記した端子は，次の状態でハイ・インピーダンスになります。

MD15-MD0:SnapShot メモリ（外部データ・メモリ）非アクセス時

・その他

端子名称	端子番号	入出力	機　能
I.C.	C6, D12, E10, E11, E12, F12, G10	−	内部接続端子 オープンにしてください。
I.C.-RL	F11	−	内部接続端子 抵抗を介して GND に接続してください。
NC	A1, A11, A12, B5, B8, B12, G12, L1, M1, M11, M12	−	ノー・コネクション オープンにしてください。

注意　これらの端子になんらかの信号の印加または読み出しを行ったとき，μPD77533 の正常な動作が保証されません。

1. 2 未使用端子の処理と入出力回路タイプ

1. 2. 1 機能端子の処理

実装時に未使用の端子は，次の表のとおりに扱ってください。

端子	入出力	推奨接続方法
ADSCK	出力	オープンにしてください。
ADDI3-ADDI0	入力	GND に接続してください。
AGC	出力	オープンにしてください。
DCOFF	出力	オープンにしてください。
RFON	出力	オープンにしてください。
MCSB	出力	オープンにしてください。
MA19-MA0	出力	オープンにしてください。
MD15-MD0	入出力	プルアップ抵抗を介して EV_{DD} に接続，またはプルダウン抵抗を介して GND に接続してください。
MRDB	出力	オープンにしてください。
MWRB	出力	オープンにしてください。
RXD	入力	GND に接続してください。
RTS	出力	オープンにしてください。
CTS	入力	GND に接続してください。
TXD	出力	オープンにしてください。
WKUPB	入力	EV_{DD} に接続してください。[注]
STNDBYB	出力	オープンにしてください。
TXACTV	入力	GND に接続してください。
INVRX	入力	GND に接続してください。

注 STOP モードを使用しない場合のみです。

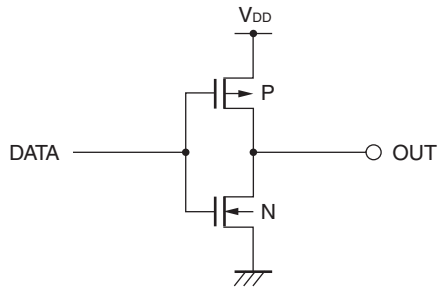
1. 2. 2 非機能端子の処理

端子	入出力	推奨接続方法
I.C.	−	オープンにしてください。
I.C.-RL	−	抵抗を介して GND に接続してください。
NC	−	オープンにしてください。

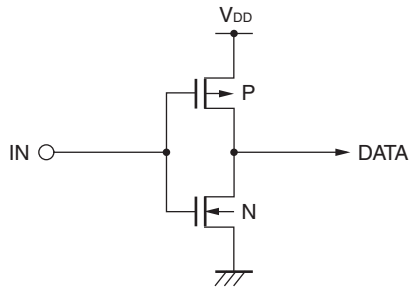
1. 2. 3 端子の入出力回路

端子名	入出力回路タイプ	端子名	入出力回路タイプ
GPSCLK	C	AGC	B
REFCLK	C	DCOFF	B
RESETB	C	RFON	B
WKUPB	C	ADSCK	A
STNDBYB	A	ADDI3-ADDI0	C
TXACTV	D	MCSB	A
INVRX	C	MRDB	A
RXD	D	MWRB	A
RTS	A	MA19-MA0	A
CTS	D	MD15-MD0	B
TXD	A		

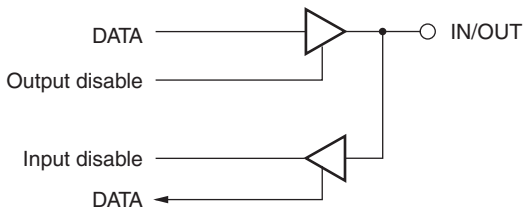
タイプ A



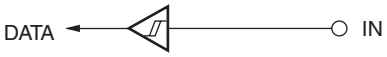
タイプ C



タイプ B



タイプ D



ヒステリシス特性を有するシュミット・トリガです。

2. 機能概要

2. 1 機能およびコマンド一覧

μPD77533 の機能は，外部端子およびコマンドで制御することで実現できます。

表 2-1 μPD77533 の機能と関連する端子／コマンド一覧

機　能		関連する外部端子 （端子番号）	関連するコマンド （CMD-ID）
リセット		RESETB(A8)	Reset(0x04) !ia Request(0x21)
通信エラー制御		—	Error Status(0x3F) Command Timeout(0x13)
測位制御		ADSCK(K10) RFON(H10) AGC(H11) DCOFF(H12)	Flow Point Control(0x7F) !cb Request(0x20)
システム制御	FCC	GPSCLK(D11) REFCLK(C12)	Frequency Calibration Control(0x06) FCC Reference Frequency(0x02) Doppler Search Range(0x57)
	ボー・レート設定	—	Baud Rate Change(0x01)
	RF 電源制御	RFON(H10)	RF Power Control(0x05)
	TXACTV 制御	TXACTV(L10)	TXACTV Control(0x07)
省電力移行／復帰		WKUPB(C9) STNDBYB(C8)	Power Control(0x03) Sleep Timer(0x12)
ディバグ機能		—	SnapShot Memory Check(0x2D) Status Request(0x30) Processing Status Request(0x33) Rest Processing(0x34) D77533 Software Version(0x31)
その他の機能		—	CA Parameter Set(0x40) CB Parameter Set(0x41)

2. 2 HOST シリアル・コマンド

2. 2. 1 コマンド・フォーマット

μPD77533－HOST CPU 間で送受信するシリアル・コマンドのフォーマットを次に示します。コマンド ID（CMD-ID），データ構造などの詳細については，12. パケットとコマンドのフォーマットを参照してください。

図 2-1 コマンド・フォーマット

CMD-ID	LENGTH	DATA	FLAG
8ビット	8ビット	0～255バイト	03H

2. 2. 2 非同期式シリアル

非同期式シリアルの仕様を次に示します。

表 2-2 非同期式シリアル仕様

ボー・レート	2400, 4800, 9600（初期値）, 19200, 38400, 57600, 115200 bps に対応
スタート・ビット	1 ビット
ストップ・ビット	1 ビット
データ・レングス	8 ビット
パリティ	なし
★ フロー制御	RTS（1：送信停止，0：送信要求） CTS（1：送信停止，0：送信要求）

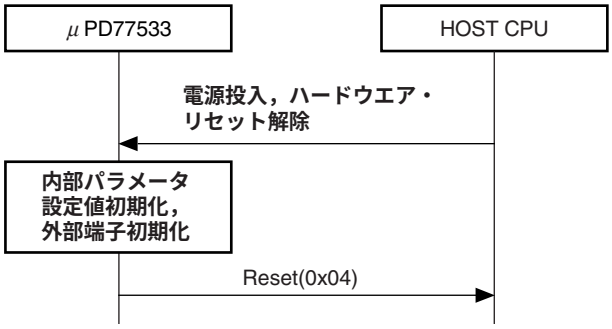
3. リセット

3. 1 リセット（RESETB 端子によるハードウエア・リセット）

μPD77533 は，RESETB 端子によるハードウエア・リセットにより，初期化処理を行います。

注意 電源投入時は，RESETB 端子へのハードウエア・リセット入力と，GPSCLK 端子へのクロック入力を実施する必要があります。

図 3-1 ハードウエア・リセット



3. 1. 1 ハードウエア・リセット時の端子状態

ハードウエア・リセット時（RESETB 入力中）の端子状態を表 3-1 に示します。

表 3-1 ハードウエア・リセット時の端子状態

端子名称	端子番号	入出力	状 態
STNDBYB	C8	出力	ロウ・レベル出力
RTS	M9	出力	ハイ・レベル出力
TXD	M10	出力	ハイ・レベル出力
AGC	H11	出力	ロウ・レベル出力
DCOFF	H12	出力	ロウ・レベル出力
RFON	H10	出力	ロウ・レベル出力
ADSCK	K10	出力	ロウ・レベル出力
MCSB	B7	出力	ハイ・レベル出力
MRDB	C7	出力	ハイ・レベル出力
MWRB	C5	出力	ハイ・レベル出力
MA19-MA0	A4, B4, A3, B3, B2, C3, C1, D2, D1, D3, E2, E1, E3, F1, F3, G1, G2, H1, H3, H2	出力	ロウ・レベル出力（すべて）
MD15-MD0	J1, K2, L2, J3, L3, M3, L4, M4, K4, L5, M5, K7, L6, K6, M7, L7	入出力	ハイ・インピーダンス

3. 1. 2 内部パラメータ初期値

μPD77533 の内部パラメータおよびその初期値を，表 3-2 に示します。

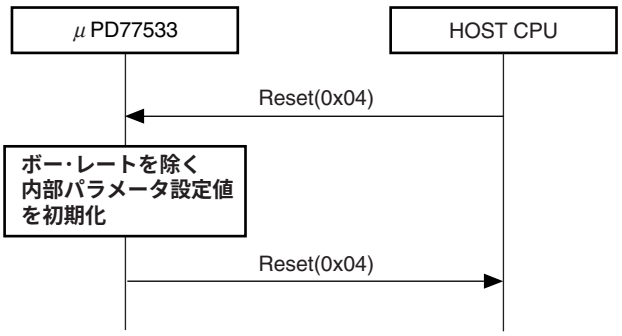
表 3-2 内部パラメータ初期値

パラメータ名	関連コマンド	CMD-ID (EXT-ID)	初期値	初期値の内容
ボー・レート	Baud Rate Change	0x01	2	9600 bps
レファレンス周波数	FCC Reference Frequency	0x02	0xE10000	14.4 MHz。パラメータの値としては，レファレンス周波数に 1.024 を掛けた値を設定します（初期値：14.4 MHz×1.024 = 0xE10000）。
省電力モード	Power Control	0x03	0	省電力モード移行として，RUN へ移行します。
電源	RF Power Control	0x05	0	電源 OFF
FCC 制御	Frequency Calibration Control	0x06	0	FCC 制御なし
TXACTV 制御	TXACTV Control	0x07	0	TXACTV 制御なし
スリープ・モード	Sleep Timer	0x12	0	SleepTimer のタイムアウト後，HALT へ移行します。
SleepTimer タイマ値（Sleep までの時間）			0	Sleep 状態へ移行しません。
キャラクタ・タイムアウトのタイマ値（タイムアウトまでの時間）	Command Timeout	0x13	0	タイムアウト検出しません。
Software バージョン	D77533 Software Version	0x31	1	Ver.1.0
ステータス（μPD77533 の信号処理状態）	Processing Status Request	0x33	—	Idle
処理済みの衛星数	Rest Processing	0x34	0	0 個
処理すべき全衛星数			0	0 個
現在の測位回数			0	0 回
全測位回数			0	0 回
エラー内容	Error Status	0x3F	—	初期値なし
ドップラ・サーチ・レンジ	Doppler Search Range	0x57	0x0168	サーチ・レンジは 360 Hz
フロー・ポイント	Flow Point Control	0x7F(0x00)	すべて 0	フロー・ポイント出力制御をしません。

3. 2 リセット（コマンド“Reset”によるソフトウエア・リセット）

μ PD77533 は，HOST CPU からのコマンド“Reset”(0x04)により，ボー・レートの設定を除く内部パラメータの初期化を行います。

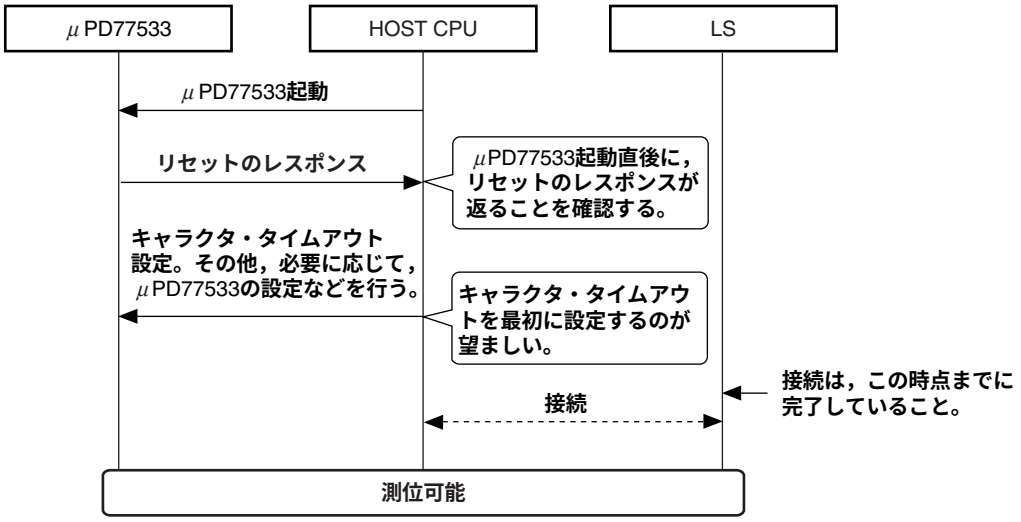
図 3-2 ソフトウエア・リセット



3. 3 起動時に必要な処理

GPS 端末を起動したとき，HOST CPU は次の手順で処理を行う必要があります。

図 3-3 μ PD77533 起動時の HOST CPU 処理手順



4. 通信エラー制御

μPD77533－HOST CPU 間で発生する可能性のある通信エラー（以降コマンド・エラーといいます）として，次の 3 つのエラー・タイプがあります。

コマンド“Error Status” （CMD-ID:0x3F）のデータ・フィールドには，次の 3 つのうちいずれかのエラー・タイプが含まれます。

表 4-1 コマンド・エラーの種類

エラー・タイプ	エラーの定義
フォーマット・エラー	コマンドのフォーマットが誤っている。
タイムアウト・エラー	コマンドのパケット受信が完了しないまま所定時間が経過した。または，HOST CPU からのコマンドに対しμPD77533 から応答がない。
未定義 CMD-ID エラー	CMD-ID として，定義されていない数値が使用されている。

4. 1 フォーマット・エラー，未定義 CMD-ID エラー

ここでは，フォーマット・エラーおよび未定義 CMD-ID エラーが発生した場合の制御フローについて説明します。タイムアウト・エラーについては，4. 2 **タイムアウト・エラー（コマンド・パケット中断）**および4. 3 **タイムアウト・エラー（μPD77533 無応答）**を参照してください。

注意 Uplink（μPD77533→HOST CPU）で，フォーマット・エラーおよび未定義CMD-IDエラーが発生した場合の制御フローは，HOST CPUからのコマンドがサーバ・コマンドの場合と，ローカル・コマンドの場合とで異なりますので，注意してください。

4. 1. 1 サーバ・コマンドの場合

コマンドがサーバ・コマンドの場合は，1 パケットごとに ACK，NACK を使ってパケットの送受信を確認します。ACK，NACK のフォーマット詳細については，13. ACK，NACK，エラー・ステータスを参照してください。

UpLink でエラーが発生した場合（図 4-1 参照），HOST CPU からμPD77533 へ NACK を送信します。すると，μPD77533 は同じパケットを再度送信します。HOST CPU からμPD77533 へ ACK を送信することにより，μPD77533 は次のパケットの送信が可能になります。

DownLink（HOST CPU→μPD77533）でエラーが発生した場合（図 4-2 参照），μPD77533 から HOST CPU へ NACK を送信します。NACK を受信した HOST CPU は，同じパケットをμPD77533 へ再度送信しなければなりません。μPD77533 から ACK を受信することにより，HOST CPU は次のパケットの送信が可能になります。

図 4-1 サーバ・コマンドのコマンド・エラー制御フロー（UpLink エラー）

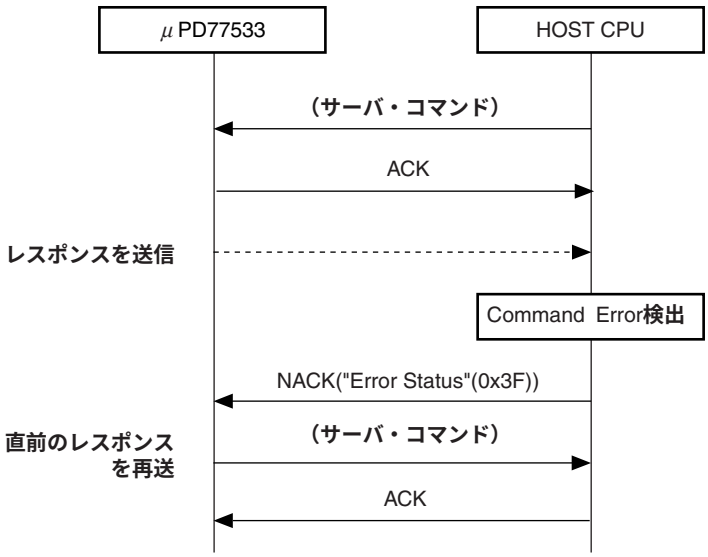
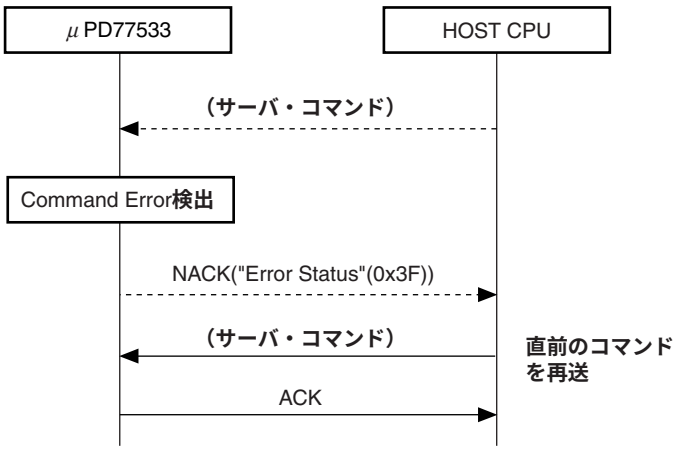


図 4-2 サーバ・コマンドのコマンド・エラー制御フロー（DownLink エラー）



4. 1. 2 **ローカル・コマンドの場合**

コマンドがローカル・コマンドの場合は，ACK，NACK **を使ったパケット送受信の確認を行いません。**

コマンドに対するレスポンス（コマンド実行結果）を，μPD77533 **から** HOST CPU **への** ACK **の代わりとします。レスポンスを返さないコマンド（ボー・レート変更コマンド）に関しては，**ACK **に相当するものはありません。**

UpLink **エラーが発生した場合（図** 4-3 **参照），**HOST CPU **から**μPD77533 **へ** NACK **を送信しても，**μPD77533 **は何も行いません。したがって，**HOST CPU **はこの場合，**NACK **を返さずに，同じパケットを再度送信しなければなりません。**

DownLink **エラーが発生した場合（図** 4-4 **参照），**μPD77533 **から** HOST CPU **へコマンド**“Error Status”（CMD-ID:0x3F）**を送信します。コマンド**“Error Status”（CMD-ID:0x3F）**を受信した** HOST CPU **は，同じパケットを**μPD77533 **へ再度送信しなければなりません。**

図 4-3 **ローカル・コマンドのコマンド・エラー制御フロー（**UpLink **エラー）**

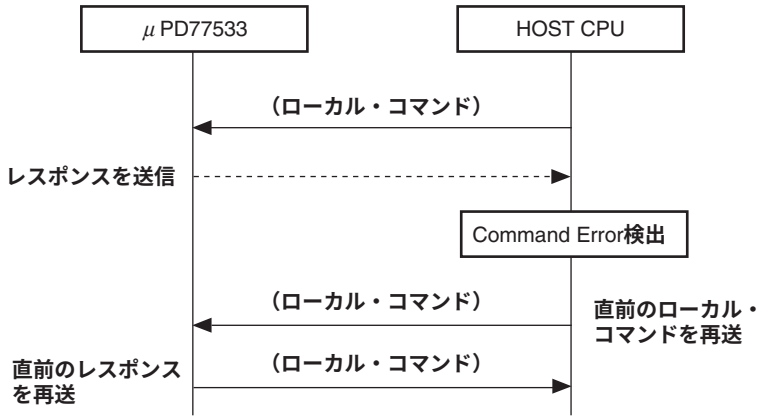
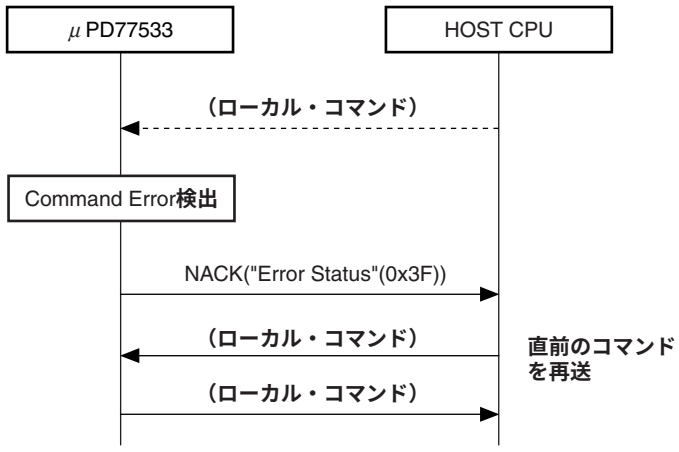


図 4-4 **ローカル・コマンドのコマンド・エラー制御フロー（**DownLink **エラー）**



4. 2 **タイムアウト・エラー（コマンド・パケット中断）**

μPD77533 **は，パケットの受信が完了しない段階で，パケットの最後のデータを受信してから指定時間経過した場合，タイムアウトとなります。**

タイムアウトになったとき，μPD77533 **はタイムアウトの**“Error Status” **（**CMD-ID:0x3F**）を返し，それまでに受信したパケットのデータを破棄します。**

4. 2. 1 キャラクタ・タイムアウトの設定

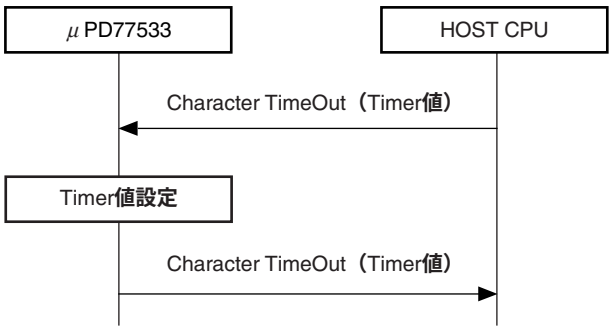
キャラクタ・タイムアウトのタイマ値の初期値は，0（タイムアウト検出しない）です（表 3-2 内部パラメータ初期値参照）。

したがって，μ PD77533（GPS 端末）を起動した直後に，HOST CPU はこのタイマ値を 0 以外に設定しておくことをお勧めします（図 3-3 μ PD77533 起動時の HOST CPU 処理手順参照）。

キャラクタ・タイムアウトのタイマ値を設定するフローを図 4-5 に示します。

タイマ値として設定できる時間は，0～16777215 ms の範囲です。ただし，0 ms のときはタイムアウト処理を行いません。

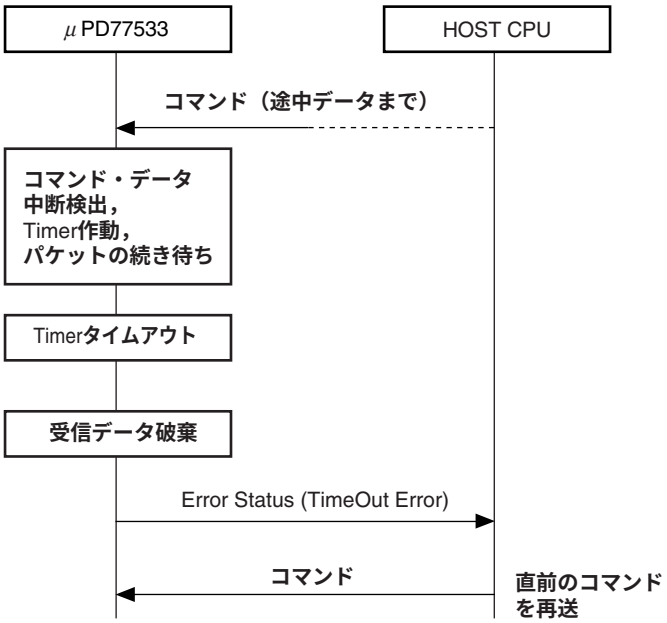
図 4-5 キャラクタ・タイムアウトのタイマ値設定



4. 2. 2 キャラクタ・タイムアウト動作

4.2.1 キャラクタ・タイムアウトの設定による，タイムアウトの動作を図 4-6 に示します。μ PD77533 は，受信途中のコマンド・データを破棄したあと，コマンド“Error Status”（CMD-ID:0x3F）を HOST CPU へ送信します。

図 4-6 キャラクタ・タイムアウト発生



4. 3 タイムアウト・エラー（μPD77533 無応答）

HOST CPU からのコマンドに対して，μPD77533 からのレスポンスがいっさいない場合，HOST CPU は次の制御を行う必要があります。

（1）μPD77533 にキャラクタ・タイムアウトが設定されている場合

μPD77533 でキャラクタ・タイムアウトが発生し，μPD77533 から HOST CPU へコマンド“Error Status”（CMD-ID:0x3F）が送信されます。HOST CPU では“Error Status”を受信後，直前のコマンドを再送します（図 4-7 参照）。

（2）μPD77533 にキャラクタ・タイムアウトが設定されていない場合

μPD77533 からのレスポンスが 100 ms 以上ない場合，HOST CPU はμPD77533 に‘3F’を 1 バイトずつ送信し，μPD77533 からコマンド“Error Status”（CMD-ID:0x3F）を受信するまで待ちます。“Error Status”（CMD-ID:0x3F）受信後，HOST CPU は直前のコマンドを再送します（図 4-8 参照）。

（2）の場合は，μPD77533 が復帰するまで時間がかかる場合が多いので，キャラクタ・タイムアウトはあらかじめ設定しておくことをお勧めします。

図 4-7 μPD77533 無応答時の処理フロー（キャラクタ・タイムアウト設定あり）

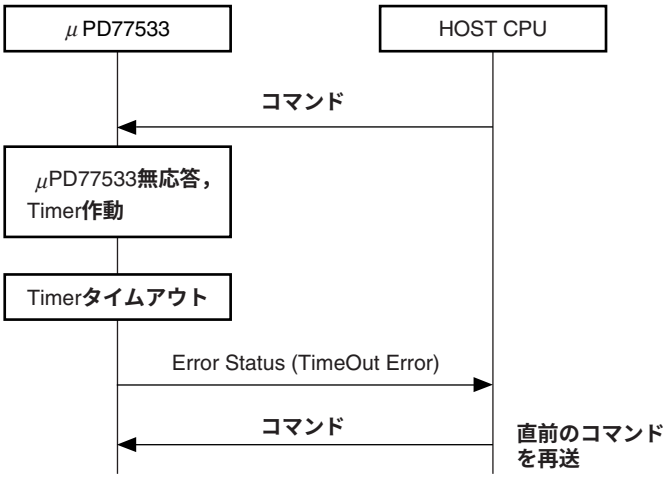
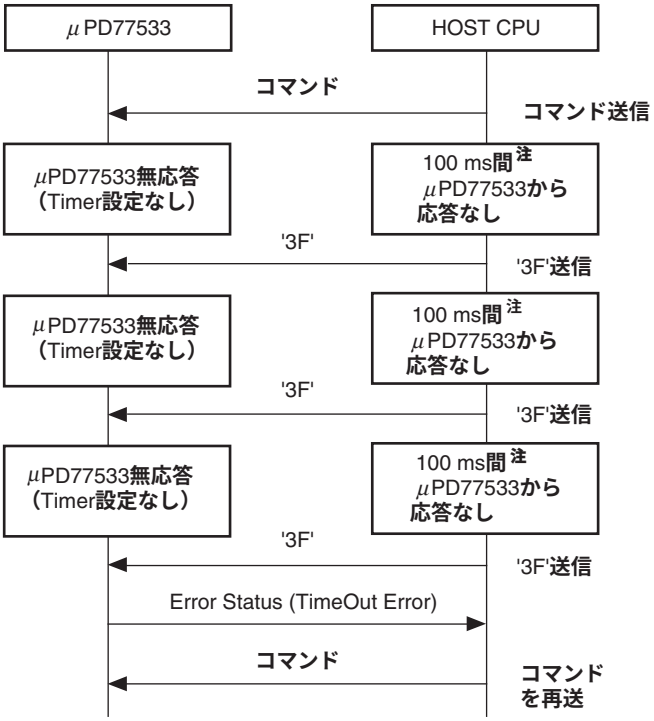


図 4-8 μPD77533 無応答時の処理フロー（キャラクタ・タイムアウト設定なし）



注 μPD77533 が測位処理を行っているときの無応答の判断は 1 s 以上とします（μPD77533 からの応答が 1 s の間なければ無応答と判断して'3F'を送信します）。'3F'送信後の無応答判断は 100 ms 以上とします。

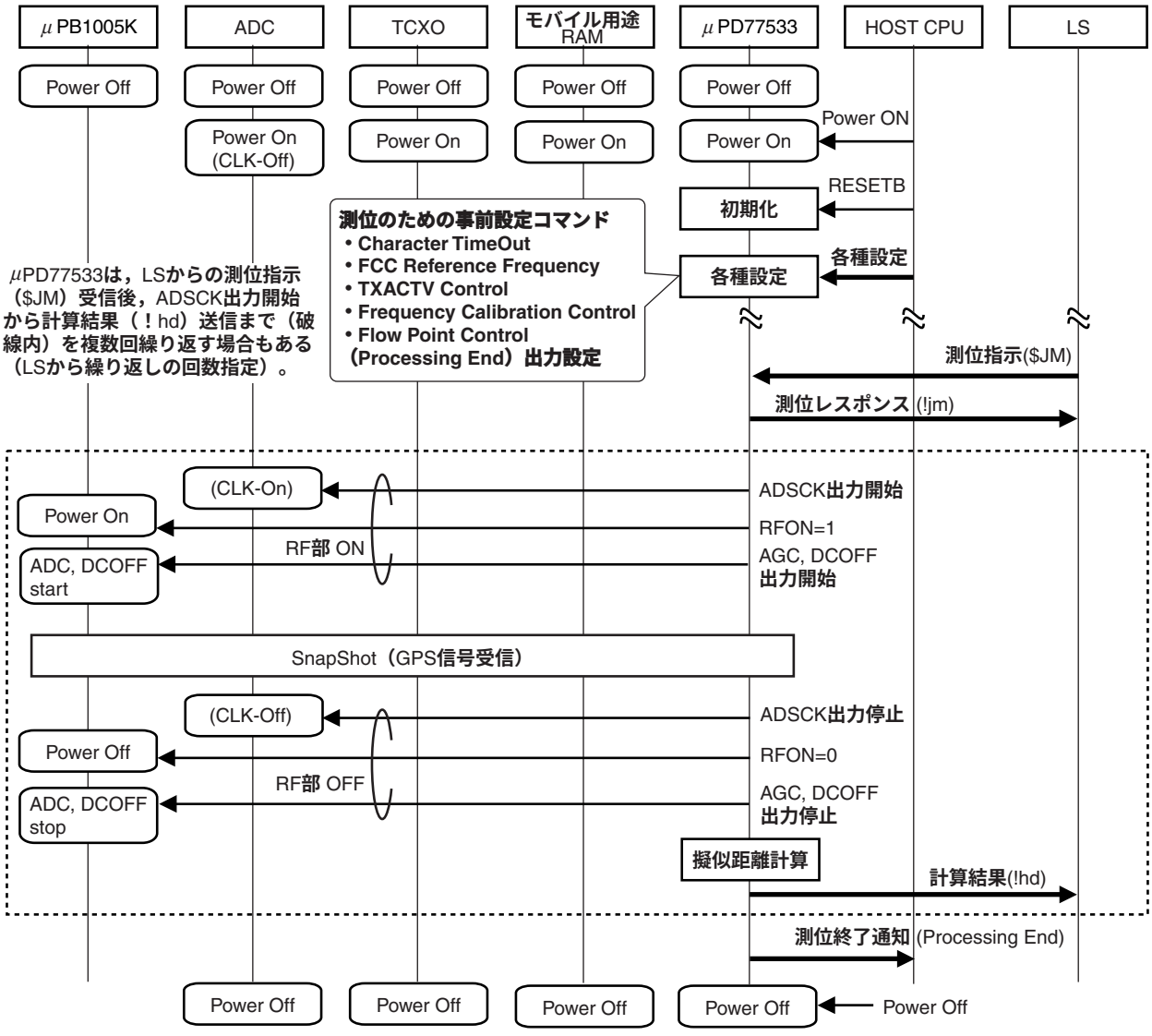
5. 測位制御

測位処理のフロー概要を，図 5-1 に示します。

注意　図中では ACK，NACK **は省略しています。**

ADC **の** Power **制御も，**RFON **端子で制御することができます。ただし，制御の論理に注意してください。**

★ **図** 5-1　**測位制御フロー概要**



備考　◀— **コマンド入力**

◀— **端子入力**

端末側から測位を開始する場合の制御フローは，システムに依存します。

次に，GPS 評価システムにおける，端末測位開始の制御フローの例について説明します。

端末側から測位を開始する場合は，HOST CPU からコマンド“!cb Request” (CMD-ID:0x20)をμPD77533 へ送ります。コマンド“!cb Request” (CMD-ID:0x20)を受信したμPD77533 は，16. 10 !cb Requestに示される“!cb Request”のフローに従って，コマンド“!cb”を LS（ロケーション・サーバ）に送ります。LS は“!cb”を受信したあと，測位を開始します。

LS から測位を要求する場合は，このコマンドを使用する必要はありません。

μPD77533 は，LS（HOST CPU）から測位コマンド“$JM”を受信したあと，図 5-1 に示されるように，次の処理内容を自動的に実行します。

1. RF 部 ON
2. GPS 信号受信
3. RF 部 OFF
4. 擬似距離計算
5. 計算結果通知
6. 測位終了通知

5. 1 測位前に設定しておくべき事項

GPS モジュール起動後，測位開始前にコマンドで設定しておくべき事項について説明します。

5. 1. 1 レファレンス周波数

コマンド：“FCC Reference Frequency”(CMD-ID:0x02)

レファレンス・クロックの周波数の初期値は 14.4 MHz です。14.4 MHz 以外の周波数のレファレンス･クロックを使用する場合は、コマンド“FCC Reference Frequency” (CMD-ID:0x02)で周波数を設定（変更）します。

5. 1. 2 周波数補正機能（FCC）

コマンド：“Frequency Calibration Control”(CMD-ID:0x06)

周波数補正機能（FCC：Frequency Calibration Control）の詳細については，6. 1. 1 Frequency Calibration Control を参照してください。

初期値は，「FCC 制御なし」です。

μPD77533 のローカル･クロック（GPSCLK 入力クロック信号，16.384 MHz）に充分に高い精度が得られない場合，すなわち周波数補正のためにレファレンス・クロック（REFCLK 入力信号）を用いる場合は，コマンド“Frequency Calibration Control”（CMD-ID:0x06）で「FCC 制御あり」に設定（変更）します。

5. 1. 3　TXACTV 制御

コマンド：“TXACTV Control”(CMD-ID:0x07)

TXACTV 制御の詳細については，6. 4　TXACTV 制御を参照してください。

初期値は，「TXACTV 制御：なし」です。

PDC/PHS の RF 出力など，ほかのノイズ源により GPS 受信信号が劣化するような場合，TXACTV 入力端子に PDC/PHS 信号送信中を示す信号を入力し，コマンド“ TXACTV Control”(CMD-ID:0x07)で TXACTV 制御を「TXACTV 制御：あり」に設定（変更）します。これにより，μ PD77533 は，TXACTV 入力信号 = 1（PDC/PHS 送信中）のときには AD コンバータ出力値を 0x0 でマスクします。

5. 1. 4　**ドップラ・サーチ・レンジ**

コマンド：“Doppler Search Range” (CMD-ID:0x57)

ドップラ・サーチ・レンジの設定値は，使用するレファレンス・クロックおよびローカル・クロックの周波数と精度に依存します。算出方法（計算式）については，6. 1. 3　Doppler Search Rangeを参照してください。

5. 1. 5　**フロー・ポイント出力**

コマンド：“Flow Point Control” (CMD-ID:0x7F)

フロー・ポイント出力の初期値は，「フロー・ポイント出力制御をしない」です。

μ PD77533 から HOST CPU に，全測位が終了したことを通知するためには，Processing End のフロー・ポイント出力を ON にする必要があります。

フロー・ポイントの詳細については，17. 1　Flow Point Controlを参照してください。

5. 2　RF 部 ON

測位開始時にμPD77533 は，外部端子 ADSCK，RFON，AGC，DCOFF を自動的に制御して，RF 部を ON にします。具体的には，次を実行します。

- ADSCK 信号出力開始
- RFON = 1（ダウン・コンバータを PowerOn）
- AGC, DCOFF 信号出力開始

5. 2. 1　AGC および DCOFF 出力信号

TXACTV 入力信号，ADDI3-ADDI0 入力信号（AD コンバータからの入力値，4 ビット）の値に対応する，AGC および DCOFF 出力信号値は，表 5-1 のとおりです。

DCOFF は，RF のオフセットを調整するために用いられる，RF 部へのフィードバック信号です。AGC は，RF のゲイン調整に用いられる，RF 部へのフィードバック信号です。

TXACTV が 1（PDC/PHS 送信中）のときは，ADC からの入力値にかかわらず，DCOFF も AGC もハイ・インピーダンスになります。

表 5-1　AGC, DCOFF の出力値

TXACTV （入力信号）	ADDI3-ADDI0 （AD コンバータからの入力値）	DCOFF （出力信号）	AGC （出力信号）
1	すべての場合	ハイ・インピーダンス	ハイ・インピーダンス
0	0b1111	1	0
0	0b1110	1	0
0	0b1101	1	0
0	0b1100	1	0
0	0b1011	1	0
0	0b1010	1	1
0	0b1001	1	1
0	0b1000	ハイ・インピーダンス	1
0	0b0111	0	1
0	0b0110	0	1
0	0b0101	0	0
0	0b0100	0	0
0	0b0011	0	0
0	0b0010	0	0
0	0b0001	0	0
0	0b0000	0	0

5. 3　GPS 信号受信（SnapShot 実行）

GPS 信号の受信を実行します。処理内容の詳細については非公開です。

- SnapShot 実行時間は HOST CPU あるいはサーバ・コマンド“$JM”により指定された時間を設定します。
- SnapShot 実行中は信号処理動作（“ステータス”のパラメータ）が「SnapShot 中」になります。
- SnapShot 終了後は信号処理動作（“ステータス”のパラメータ）が「Idle 中」になります。

備考　ステータスのパラメータについては，表 3-2　内部パラメータ初期値を参照してください。

5. 4　RF 部 OFF

GPS 信号を受信した後，RF 部を OFF にします。具体的には，次を実行します。

- ADSCK 信号出力停止
- RFON = 0（ダウン・コンバータを PowerOff）
- AGC, DCOFF 信号出力停止

5. 5　**擬似距離計算**

受信した GPS 信号を用いて，擬似距離計算を行います。

5. 6　**測位結果通知**

μ PD77533 は，擬似距離計算の処理を終了したあと，サーバ・コマンド“!hd”を発行して，処理結果を HOST CPU 経由で LS に通知します。

5. 7　**測位終了通知**

μ PD77533 は全測位の終了後，“Processing End” (CMD-ID:0x7F)を送信し，すべての測位が終了したことを HOST CPU へ通知します。

ただし，測位前に，コマンド“Flow Point Control” (CMD-ID:0x7F)でフロー・ポイント出力制御の設定を ON にしておくことが必要です。詳細については17. 1　Flow Point Controlを参照してください。

コマンド“Flow Point Control”(CMD-ID:0x7F)の初期値は，「フロー・ポイント出力制御をしない」となっていることに注意してください。

HOST CPU はこのレスポンスを受信したあとに，各周辺モジュール（モバイル用途 RAM など）の PowerOFF などを行うことができます。

5. 8　**測位中の送信禁止コマンド**

μ PD77533 の測位処理中に，HOST が送信してはいけないコマンドを，表 5-2 に示します。

表 5-2　測位中に送信してはいけないコマンド

コマンド・タイプ	CMD-ID	Command
ローカル・コマンド	0x01	Baud Rate Change
	0x05	RF Power Control
	0x06	Frequency Calibration Control
	0x07	TXACTV Control

6. システム制御

6. 1 FCC 機能

FCC（FCC: Frequency Calibration Control）機能とは，精度の低いローカル・クロック（GPSCLK 入力端子への入力クロック信号，16.384 MHz）を，より精度の高いレファレンス・クロック（REFCLK 入力端子への入力信号）で補正する機能です。

μPD77533 は，測位中（SnapShot 中），FCC に用いるレファレンス・クロックが供給されている必要があります。

FCC で補正するローカル・クロックの精度は，約±10 ppm までです。

FCC に関連するコマンドは次の 3 種類です。

- “Frequency Calibration Control” (CMD-ID:0x06)
- “FCC Reference Frequency” (CMD-ID:0x02)
- “Doppler Search Range” (CMD-ID:0x57)

6. 1. 1 Frequency Calibration Control

コマンド“Frequency Calibration Control” (CMD-ID:0x06)により，FCC の有無を設定，参照します。

設定方法の詳細については16. 6 Frequency Calibration Controlを参照してください。

6. 1. 2 FCC Reference Frequency

コマンド“FCC Reference Frequency” (CMD-ID:0x02)により，FCC のためのレファレンス周波数を設定，参照します。

リファレンス・クロックの初期設定は 14.4 MHz です。

設定値は，周波数に 1.024 を掛けた値を，符号なしの 4 バイト・データで設定します（したがって初期値は 0x00E10000（14.4 MHz×1.024）です）。

詳細については16. 2 FCC Reference Frequencyを参照してください。。

6. 1. 3 Doppler Search Range

コマンド“Doppler Search Range” (0x57)により，ドップラ・サーチ・レンジ（受信信号の周波数のサーチ範囲）を設定します。初期値は，360 Hz（0x168）です。

ドップラ・サーチ・レンジとして設定する値は，ローカル・クロックおよびレファレンス・クロックによって決まります。算出方法を次に示します。

パラメータ Doppler Search Range の値（Hz）の最小値は，次式により算出されます。

Doppler Search Range = 2×｜｜DIFr｜－｜DIFi｜｜（Hz）

このとき，

$$DIFr = -1.57542G \times \frac{\left[REFCLKr \times 2 \times \frac{16.384M \times 1.024}{GPSCLKr} + 1\right] - REFCLKi \times 2 \times 1.024}{REFCLKi \times 2 \times 1.024}$$

$$DIFi = -1.57542G \times \frac{\left[REFCLKi \times 2 \times \frac{16.384M \times 1.024}{GPSCLKr}\right] - REFCLKi \times 2 \times 1.024}{REFCLKi \times 2 \times 1.024}$$

REFCLKi ：REFCLK の理論値
REFCLKr ：REFCLK の実際の値（誤差を含む値）
GPSCLKr ：GPSCLK の実際の値（誤差を含む値）

＜例＞
REFCLKi ：14.4 MHz
REFCLKr ：14.400002 MHz
GPSCLKr ：16.383900 MHz
である場合，

$$DIFr = -1.57542G \times \frac{\left[14.400002M \times 2 \times \frac{16.384M \times 1.024}{16.383900M} + 1\right] - 14.4M \times 2 \times 1.024}{14.4M \times 2 \times 1.024} = -9882.7$$

$$DIFi = -1.57542G \times \frac{\left[14.4M \times 2 \times \frac{16.384M \times 1.024}{16.383900M}\right] - 14.4M \times 2 \times 1.024}{14.4M \times 2 \times 1.024} = -9615.6$$

したがって，
Doppler Search Range = 2×｜｜−9882.7｜−｜−9615.6｜｜ = 534.2（Hz）
詳細については16. 20　Doppler Search Rangeを参照してください。

6. 2　ボー・レート設定

コマンド“Baud Rate Change”(CMD-ID:0x01)により，μPD77533−HOST CPU 間のボー・レートの設定を行います。
設定できるボー・レートの値は，2400 bps，4800 bps，9600 bps（初期値），19200 bps，38400 bps，57600 bps，115200 bps の 7 種類です。
μPD77533 がほかのコマンドを受信中にコマンド“Baud Rate Change” (CMD-ID:0x01)を受信した場合，前のコマンドの受信が終了してからボー・レートの設定を行います。

6. 3　RF 電源制御

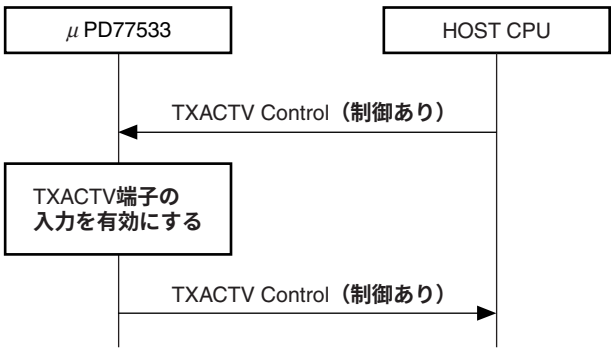
コマンド “RF Power Control”(CMD-ID:0x05)により，RF 部の電源制御を行います。

測位中（GPS 信号受信時）は，μPD77533 が RF 部の電源制御を自律的に行うので，このコマンドによる HOST CPU からの制御は必要ありません。

6. 4　TXACTV 制御

コマンド“TXACTV Control”(CMD-ID:0x07)により，TXACTV 端子入力による A/D 変換値のマスク制御の有無を設定します。

図 6-1　TXACTV 制御



μPD77533 は，PDC/PHS の RF 出力など，ほかのノイズ源により GPS 受信信号が劣化するような場合，A/D コンバータ出力値を 0H でマスクする機能を持っています。初期設定は，A/D 変換値のマスク制御なしに設定されています。

TXACTV 入力端子に，PDC/PHS 信号送信中を示す信号を入力し，TXACTV 制御を「A/D 変換値のマスク制御：あり」にすると，TXACTV 入力信号 = 1（PDC/PHS 送信中）のときは，A/D コンバータの出力値（ADDI3-ADDI0 端子入力値）の内容にかかわらず，μ PD77533 内部では受信信号を‘0b0000’として扱います（すなわち，A/D 変換値をマスクします）。また，そのときの AGC と DCOFF 端子出力は，ハイ・インピーダンスとなります。

TXACTV 信号入力と AGC および DCOFF 信号出力の関係については，5. 2. 1　AGC および DCOFF 出力信号を参照してください。

7. 省電力移行，復帰

μPD77533 は，省電力モードとして，STOP と HALT という 2 種類のモードを持っています。

通常モード（RUN）から STOP または HALT モードへの移行は，コマンド“Power”(CMD-ID:0x03)または“Sleep Timer”(CMD-ID:0x12)により行います。

STOP または HALT モードから通常モード（RUN）への復帰は，WKUPB 端子入力 = 0 により，状態遷移します。HALT からの復帰の場合は，任意のコマンド受信でも状態遷移します。省電力モードへの移行，復帰の概要を，図 7-1 に示します。

HALT と STOP の違いは次のとおりです。

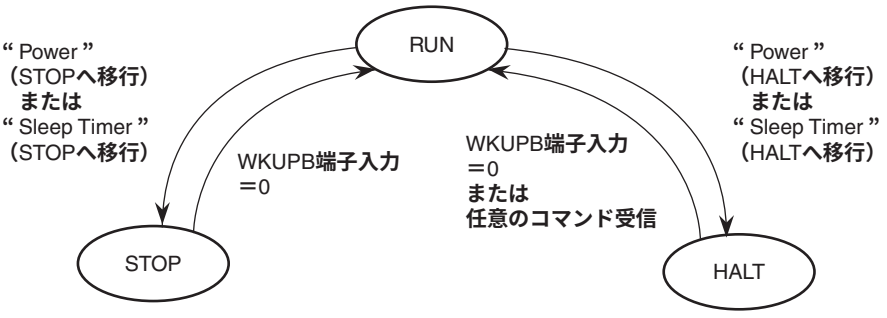
[HALT]

- μPD77533 内部の PLL を停止させず，高速復帰が可能です。
- WKUPB 端子入力 = 0，または任意のコマンド受信で復帰します。

[STOP]

- μPD77533 の内部 PLL を停止し，700 μA（MAX.）での低消費電流となります。
- WKUPB 端子入力 = 0 で通常モード（RUN）へ復帰します。

図 7-1　省電力モードへの移行



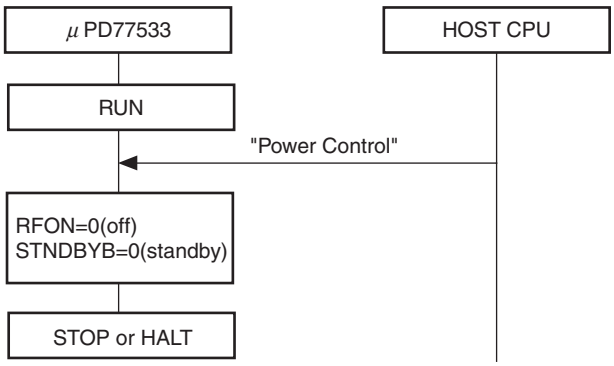
7. 1 “Power”コマンドによる省電力モードへの移行

コマンド“Power” (CMD-ID:0x03)（データ・フィールドに，STOP または HALT を設定）により，μPD77533 は省電力モード（STOP または HALT）へ移行します。省電力モードへの移行フローを図 7-2 に示します。

コマンド“Power” (CMD-ID:0x03)を受信したμPD77533 は，次の処理を行ってから，指定の省電力モードへ移行します。

- RF 部のパワーを OFF（RFON 出力信号 = 0）
- スタンバイ状態へ移行したことを外部へ表示（STNDBYB 出力信号 = 0：スタンバイ状態）

図 7-2 省電力モードへの移行フロー



7. 2 “Sleep Timer”コマンドによる省電力モードへの移行

コマンド“Sleep Timer” (CMD-ID:0x12)で設定されたタイムアウト設定期間，Idle 状態（信号処理停止状態かつシリアル・コマンド送受信なしの状態）が続いた場合，μPD77533 は省電力モード（STOP または HALT）に移行します。移行するモードはあらかじめ“Sleep Timer” (CMD-ID:0x12)で設定します。

ここでは，Sleep Timer の設定および設定通知と動作について説明します。

7. 2. 1 Sleep Timer の設定

Sleep Timer の設定は，コマンド“Sleep Timer” (CMD-ID:0x12)で設定します。

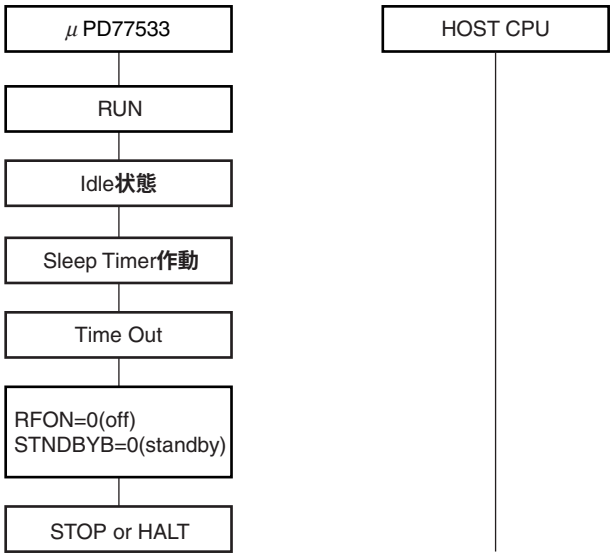
コマンド“Sleep Timer”のデータ・フィールドには，移行するべき省電力モード（STOP または HALT）と，移行までのタイマ値が含まれます。

Sleep Timer のタイムアウト値設定範囲は，0～16777215 (0xFFFFFF)ms です。ただし，0 ms のときは省電力モードへ移行しません。

7. 2. 2 Sleep Timer の動作

7.2.1 で設定された Sleep Timer がタイムアウトしたときの動作を，図 7-3 に示します。

図 7-3 Sleep Timer タイムアウト



備考 Idle 状態：信号処理停止状態かつシリアル・コマンド送受信なしの状態

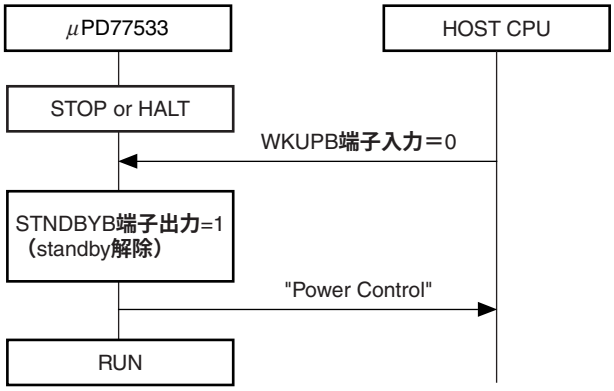
7. 3 省電力モードからの復帰

WKUPB 端子入力 = 0，あるいは任意のコマンド受信により（HALT の場合のみ），μPD77533 は省電力モード（STOP または HALT）から通常モード（RUN）へ復帰します。WKUPB 端子はロウ・アクティブです。

7. 3. 1 WKUPB 端子入力による復帰

WKUPB 端子入力による省電力モードからの復帰手順を，図 7-4 に示します。

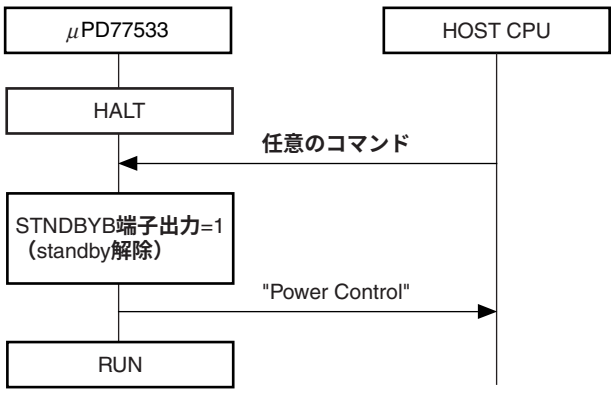
図 7-4 省電力モードからの復帰（WKUPB 端子入力による）



7. 3. 2 コマンド受信による復帰

コマンド受信による，HALT モードからの復帰手順を図 7-5 に示します。HALT モードからの復帰は WKUPB 端子入力制御またはシリアル・コマンド受信によって行います。

図 7-5 省電力モード（HALT）からの復帰（コマンド受信による）



8. ディバグ機能

ディバグ機能として，ローカル・コマンドにより次のことを確認できます。
詳細については，各コマンドの説明を参照してください。

表 8-1 ディバグ機能

項　　目	コマンド	機　　能
SnapShot メモリのチェック	“SnapShot Memory Check” (CMD-ID:0x2D)	SnapShot メモリの全エリアに対して，0x0000，0xFFFF などの値を書き込み，正しく読み出せるかどうかをチェックします。
シリアル通信の動作確認	“Status Request”(CMD-ID:0x30)	μPD77533 の動作確認を行います。 このコマンドに対して，μPD77533 はレスポンスを返すだけです。
信号処理状態の参照	“Processing Status Request” (CMD-ID:0x33)	μ PD77533 の信号処理状態が次のいずれにあるかを参照します。 ・SnapShot 実行中 ・信号処理中 ・Idle 中 μ PD77533 は，測位制御中に，SnapShot 実行中および信号処理中のモードに遷移します。 モード遷移タイミングについては，5. 測位制御を参照してください。
測位処理の経過の参照	“Rest Processing” (CMD-ID:0x34)	測位時に，処理すべき残り衛星数と測位回数を参照します。
μ PD77533 のバージョンの参照	“D77533 Software Version” (CMD-ID:0x31)	μPD77533 のソフトウエア（ファームウエア）のバージョンを参照します。

9. その他の機能

9. 1 端末 ID 通知

端末の ID 番号を LS（ロケーション・サーバ）へ通知するためのコマンドは次の 2 種類です。詳細は16. 18 CA Parameter Set，16. 19 CB Parameter Setを参照してください。

- CA Parameter Set(CMD-ID:0x40) ：
  GPS モジュールの Unit ID とソフトウエア・バージョンの設定または設定通知を行います。

- CB Parameter Set(CMD-ID:0x41)
  μPD77533 が測位開始コマンド（!cb）を発行するときのパラメータの設定または設定通知を行います。

10. コマンド機能概要

μPD77533 を使う際のシステム構成は次のとおりです。ここでは，μPD77533－HOST CPU 間のコマンド・インタフェースについて説明します。

図 10-1 GPS 測位システムの構成図

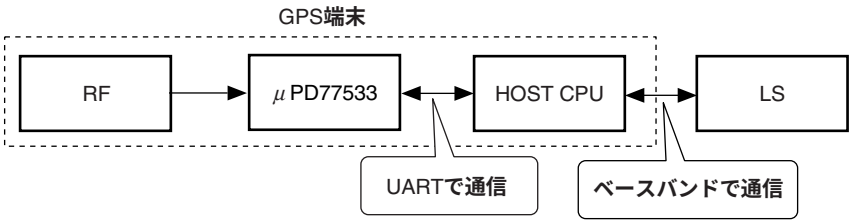


表 10-1 各ブロックの働き

ブロック	機　能
RF	Radio Frequency GPS 衛星からの電波を受信し，μPD77533 に与えます。
μPD77533	DSP コアとロジック回路で構成され，RF からの信号を処理し，LS に送ります。
HOST CPU	μPD77533 の動作を制御します。また，μPD77533－LS 間の通信を媒介します。
LS	ロケーション・サーバ μPD77533 が信号処理した結果に基づき，端末の位置を計算します。

11. コマンドの種類とレスポンス

μ PD77533 は，HOST CPU または LS から受信したコマンド実行して，レスポンスを返します。LS が発行するコマンドをサーバ・コマンド，HOST CPU が発行するコマンドをローカル・コマンドといいます。μPD77533 から HOST CPU または LS に発行する応答はレスポンスといいます。

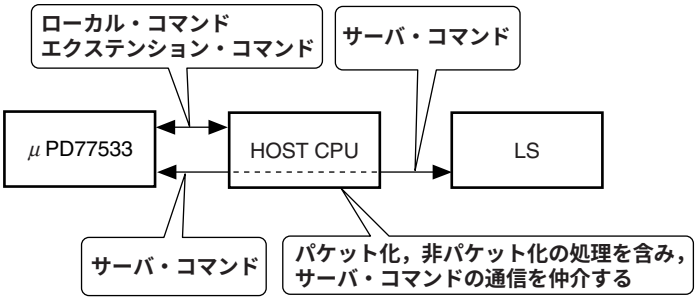
11. 1 コマンド

コマンドには次の 3 種類があります。

表 11-1 コマンドの種類

種　類	発行元	用　途
ローカル・コマンド	HOST CPU	μPD77533 の設定，制御，現在の状態の取得などのために使用します。コマンドは HOST CPU が発行し，μ PD77533 からのレスポンスは HOST CPU が受け取ります。LS は関与しません。
エクステンション・コマンド	HOST CPU	ローカル・コマンドを拡張するために用意しています。コマンドは HOST CPU が発行し，μPD77533 からのレスポンスは HOST CPU が受け取ります。LS は関与しません。
サーバ・コマンド	LS	測位を行うために使用します。コマンドは LS が発行し，μ PD77533 からのレスポンスは LS が受け取ります。HOST CPU はデータ・フォーマットの変換（パケット化，非パケット化）を行い，μ PD77533 と LS 間のコマンド，レスポンスの通信を仲介します。また，一部のサーバ・コマンドについては，HOST CPU が処理する必要があります。

図 11-1 各コマンドと機能ブロックの関係



11. 2 レスポンス

レスポンスには，コマンドへ応答するものと，μPD77533 が自発的に送信するものがあります。

前者の場合，1 つのコマンドに対するレスポンスは 1 つであるとは限りません。複数のレスポンスを返す場合や，レスポンスがないコマンドもあります。

後者の場合，レスポンスを送信するタイミングは不定です。したがって，HOST CPU は常にμPD77533 からのレスポンスを受信できる状態でいなければなりません。

12. パケットとコマンドのフォーマット

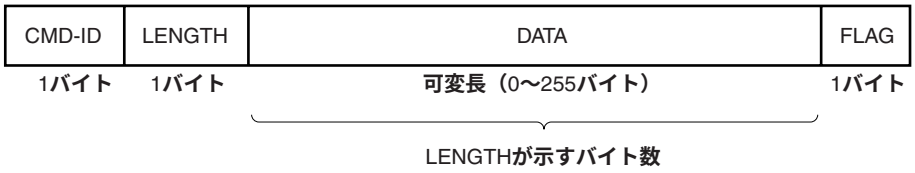
μPD77533－HOST CPU 間のデータ送受信は，パケット形式のデータを使用します。

データは 8 ビットのバイナリ・データ（リトル・エンディアン）です。ASCII キャラクタだけではないので注意してください。

12. 1 パケットのフォーマット

パケットのフォーマットを次に示します。パケットの最大長は 258 バイトです。

図 12-1 パケットのフォーマット



12. 1. 1 CMD-ID フィールド

CMD-ID フィールドは，ローカル・コマンド，サーバ・コマンド，エクステンション・コマンドの種類を示します。

ローカル・コマンドの場合，CMD-ID がそのままローカル・コマンドの識別子になります。

表 12-1 CMD-ID の種類

CMD-ID	種　類
0x00～0x7E	ローカル・コマンド（CMD-ID の値がコマンドの識別子）
0x7F	エクステンション・コマンド
0x80	サーバ・コマンド

12. 1. 2 LENGTH フィールド

DATA フィールドの長さをバイト数で示します。値は 0～255 の範囲です。CMD-ID，LENGTH，FLAG は長さには含みません。

12. 1. 3 DATA フィールド

CMD-ID で指定されたコマンドのデータ，ステータスなどを格納します。

DATA フィールドは LENGTH フィールドで示した長さです。コマンドによっては DATA フィールドがない場合があります。

HOST CPU がサーバ・コマンドをパケット化する際，非パケット化状態でサーバ・コマンドの全長が 255 バイトを越える場合，複数のパケットに分割します。

1 つのデータが 2 バイト以上のサイズを持つ場合，下位バイトから上位バイトの順で格納します（リトル・エンディアン）。

12. 1. 4 FLAG フィールド

パケットの末尾を示すフラグです。2 種類の FLAG があります。

表 12-2 FLAG の種類

FLAG	意　味
0x03	単独，または分割されたパケットのうちの最終パケット
0x02	分割されたパケットのうちの先頭，または途中のパケット

12. 2 パケットの分割について

サーバ・コマンド，またはサーバへのレスポンスの場合，非パケット化の状態での長さが 255 バイトを越える場合があります。この場合，1 パケットにデータが収まらないので，HOST CPU は複数のパケットに分割します。

ローカル・コマンド，エクステンション・コマンドでは，複数のパケットに分割することはできません。コマンドは 1 パケットで完結している必要があります。

図 12-2 単独パケットの例

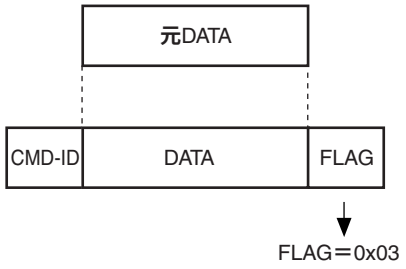
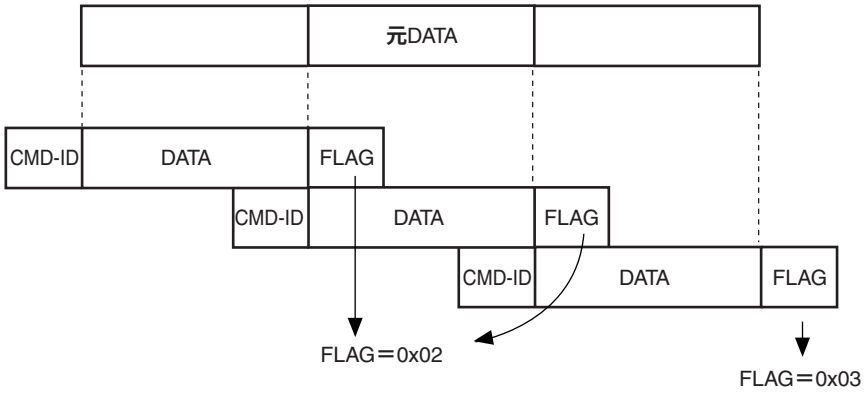


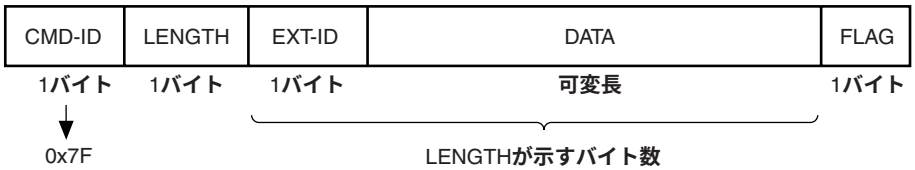
図 12-3 複数パケットへの分割の例



12. 3 エクステンション・コマンドのフォーマット

エクステンション・コマンド（CMD-ID = 0x7F）は，EXT-ID フィールドを追加し，EXT-ID によりコマンドを識別します。

図 12-4 エクステンション・コマンドのフォーマット

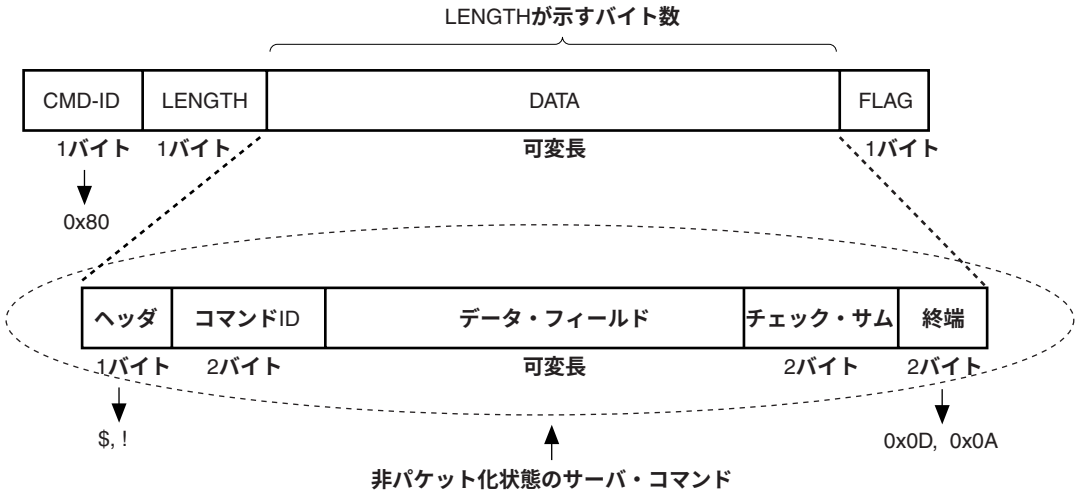


12. 4 サーバ・コマンドのフォーマット

次に示すフォーマットに従って，サーバ・コマンドをパケットの DATA フィールドに格納します。

LS が送受信する時点でのサーバ・コマンドは，パケット化されていません。したがって，HOST CPU はサーバ・コマンドをパケット化する必要があります。

図 12-5 サーバ・コマンドのフォーマット



12. 4. 1 ヘッダ

サーバ・コマンド，サーバへのレスポンスは，ASCII 文字がヘッダとなります。ヘッダは，コマンド，レスポンスの先頭を示します。

表 12-3 ヘッダ

種　類	ASCII 文字
コマンド	$ (0x24)
レスポンス	! (0x21)

12. 4. 2 コマンド ID

コマンド，レスポンスの種類を識別するための ID は，2 バイトの ASCII 文字（2 文字）で構成されます。

表 12-4 コマンド ID

種　類	ASCII 文字
コマンド	A～Z の大文字 2 文字の組み合わせ
レスポンス	a～z の小文字 2 文字の組み合わせ

12. 4. 3 データ・フィールド

各コマンドおよびレスポンスのデータ，ステータスなどを格納します。

12. 4. 4 チェック・サム

チェック・サムは，CMD-ID の 1 バイト目から，チェック・サムの直前までのすべてのバイトの和の下位 16 ビットを使用します。

12. 4. 5 **終端**

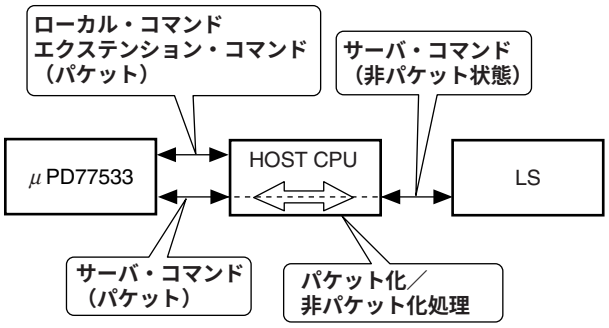
すべてのコマンドおよびレスポンスは<CR><LF>**で終わります。**

<CR> = 0x0D, <LF> = 0x0A

12. 5 サーバ・コマンドのパケット化，非パケット化

LS **が送信する時点では，サーバ・コマンドは**μ PD77533－HOST CPU **間インタフェース用のパケットになっていないので，**HOST CPU **がパケット化を行います。また，**μPD77533 **が** LS **に送るデータはパケット化されているので，**HOST CPU **は非パケット化してから** LS **に送ります。**

図 12-6 **サーバ・コマンドのパケット化，非パケット化**



13. ACK，NACK，エラー・ステータス

μPD77533－HOST CPU **間の通信において，サーバ・コマンド，エクステンション・コマンドは，**ACK，NACK **によりコマンド，レスポンスの送受信を確認します。**

表 13-1 ACK, NACK **の用途**

種　類	内　容
ACK	**コマンドを受理したことを示します。**
NACK **（エラー・ステータス）**	**何らかの理由により，コマンドを受理できなかったことを示します。**

13. 1 ACK，NACK，エラー・ステータスのフォーマット

13. 1. 1 ACK

ACK には，サーバ・コマンドに対する ACK（ACK フォーマット 1），エクステンション・コマンドに対する ACK（ACK フォーマット 2）の 2 種類のフォーマットがあります。

μPD77533 は ACK フォーマット 1，2 のどちらを受信しても，区別なく ACK として扱います。HOST CPU は状況に関わりなく，どちらの ACK を用いてもかまいません。

μPD77533 が HOST CPU に返す ACK に関しては，サーバ・コマンドを受け付けたことに対する ACK は ACK フォーマット 1 で，エクステンション・コマンドを受け付けたことに対する ACK は ACK フォーマット 2 で返します。

表 13-2 ACK フォーマット 1 （サーバ・コマンドに対する ACK）

フィールド	値	バイト数	内　容
CMD-ID	0x80	1	
LENGTH	0x00	1	
FLAG	0x03	1	

表 13-3 ACK フォーマット 2 （エクステンション・コマンドに対する ACK）

フィールド	値	バイト数	内　容
CMD-ID	0x3F	1	
LENGTH	0x01	1	
DATA	0x00	1	0x00 で ACK を示します。
FLAG	0x03	1	

13. 1. 2 NACK，エラー・ステータス

表 13-4 NACK, エラー・ステータスのフォーマット

フィールド	値	バイト数	内　容
CMD-ID	0x3F	1	
LENGTH	0x01	1	
DATA	ErrStat	1	0x00：ACK（OK） 0x01：フォーマット・エラー 0x02：タイムアウト・エラー 0x04：未定義 CMD-ID エラー Else：未定義
FLAG	0x03	1	

13. 2 コマンド，レスポンス，ACK，NACK のフロー

13. 2. 1 ローカル・コマンドのときの ACK，NACK

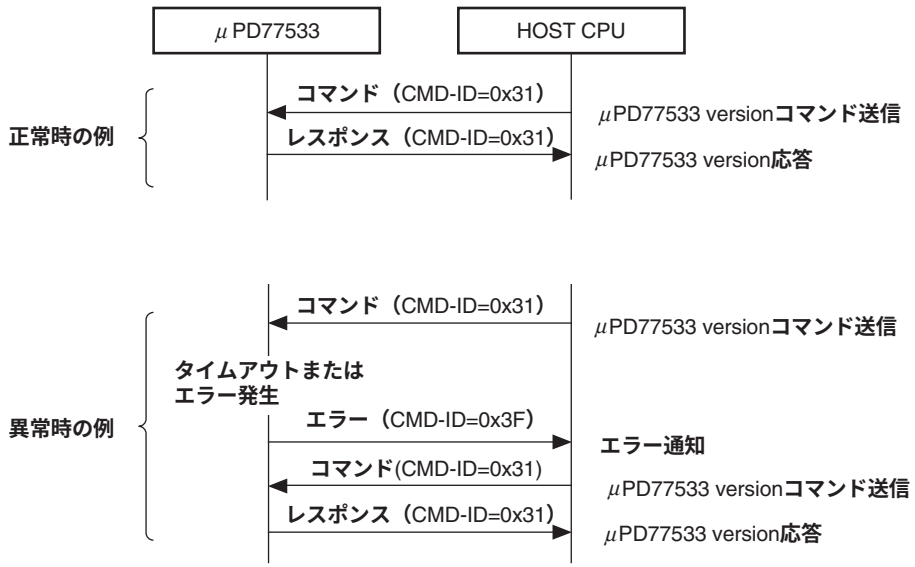
コマンドがローカル・コマンドのときは，ACK，NACK を使ったパケット送受信の確認は行いません。

コマンドに対するレスポンス（コマンド実行結果）を，μPD77533 から HOST CPU への ACK の代わりとします。

レスポンスを返さないコマンド（ボー・レート変更コマンド）に関しては，ACK に相当するものはありません。

HOST CPU からμPD77533 へ NACK を送信しても，μPD77533 はパケットの再送を行いません。

図 13-1 ローカル・コマンドのフロー



13. 2. 2 エクステンション・コマンドのときの ACK，NACK

エクステンション・コマンドでは，ACK，NACK を使うものと使わないものがあります。

HOST CPU からμPD77533 へ NACK を送信すると，同じパケットを再度送信します。HOST CPU からμPD77533 へ ACK を送信することにより，μPD77533 は次のパケットの送信が可能になります。

μPD77533 から HOST CPU にフロー・ポイントの出力が行われたら，HOST CPU は必ず ACK または NACK を送らなければなりません。μPD77533 は ACK または NACK を受信するまで，いっさいのパケットの出力を停止します。

表 13-5 エクステンション・コマンドの ACK，NACK

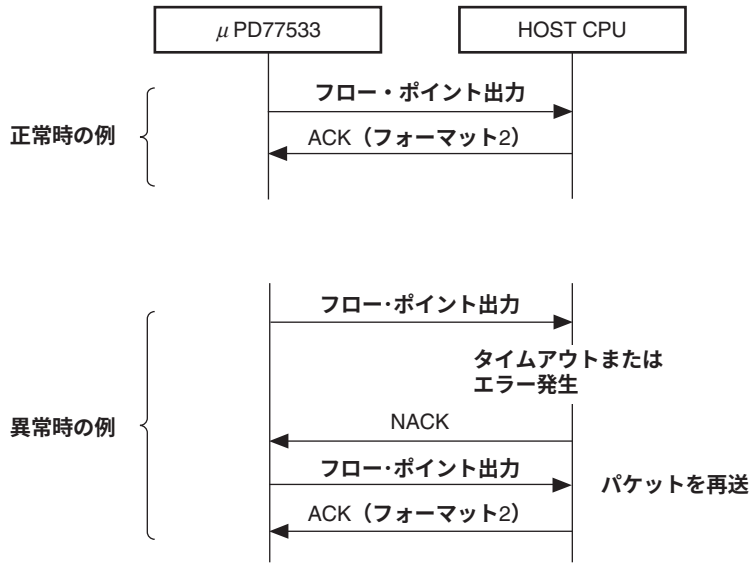
コマンドの種類	EXT-ID	ACK，NACK 制御
フロー・ポイントの設定コマンド	0x00	しない
フロー・ポイントの出力	0x00 以外	する

(1) フロー・ポイントの設定コマンドの場合

図 13-1 ローカル・コマンドのフローを参照してください。

(2) フロー・ポイントの出力の場合

図 13-2 フロー・ポイントのフロー



13. 2. 3 サーバ・コマンドのときの ACK，NACK

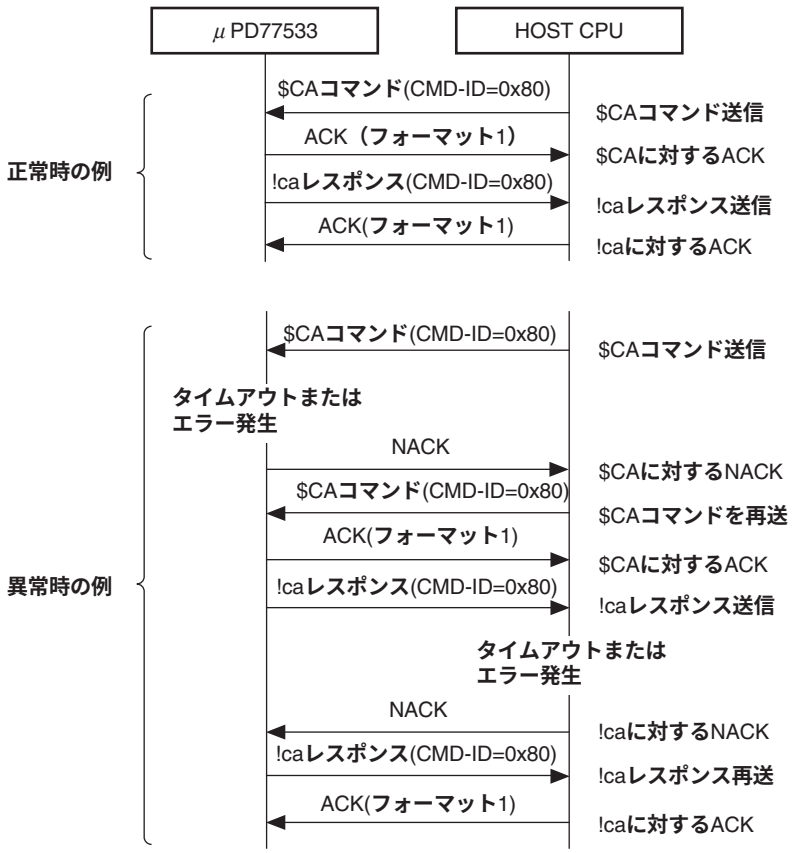
コマンドがサーバ・コマンドのときは，1 パケットごとに ACK，NACK を使ってパケットの送受信を確認します。

HOST CPU から μPD77533 へ NACK を送信すると，同じパケットを再度送信します。HOST CPU から μPD77533 へ ACK を送信することにより，μPD77533 は次のパケットの送信が可能になります。

μPD77533 から HOST CPU にレスポンスの出力が行われたら，HOST CPU は必ず ACK または NACK を送らなければなりません。

μPD77533 は ACK または NACK を受信するまで，いっさいのパケットの出力を停止します。

図 13-3 サーバ・コマンドのフロー



14. ハードウエア・フロー制御

μPD77533－HOST CPU 間の通信は，CTS-RTS を使用してハードウエア・フロー制御を行います。これにより，バッファのオーバフローによるデータ落ちを防ぐことができます。

14. 1 ハードウエア・フロー制御を使用しない場合

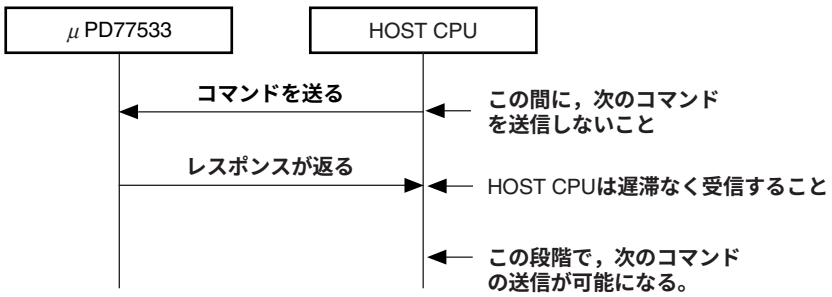
ハードウエア・フロー制御を使用しない場合，バッファ・オーバフローによるデータ落ちが発生する可能性があります。これを防ぐために，HOST CPU は次のような処理手順で行ってください。

14. 1. 1　**ローカル・コマンド**

ローカル・コマンドの場合は，コマンドのレスポンスが返ってくるまで，次のコマンドを送信しないでください。HOST CPU **は** 1 **パケット分のバッファを用意し，**μPD77533 **からのレスポンスを遅滞なく受信しなければなりません。**

μPD77533 **が信号処理中のときは，レスポンスの返らないローカル・コマンドを送信しないでください。信号処理中はコマンドを受信してから実行するまでに若干時間がかかるため，コマンドの実行完了を知ることができないからです。**

図 14-1　**ローカル・コマンドのフロー（ハードウエア・フロー制御なしの場合）**

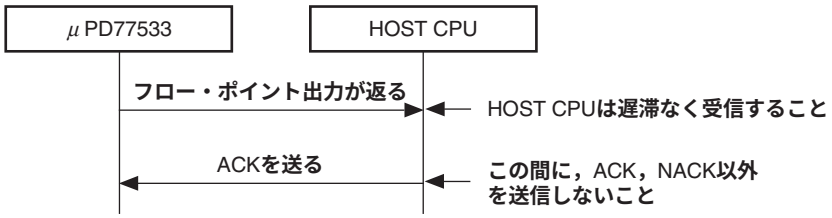


14. 1. 2　**エクステンション・コマンド**

フロー・ポイント設定コマンドの場合は，14.1.1　**ローカル・コマンドと同様に処理してください。**

フロー・ポイントの出力の場合，ACK**，**NACK **でパケット送受信の管理が可能であるため，**HOST CPU **側に** 1 **パケット分のバッファを用意し，**μPD77533 **からのパケットを遅滞なく受信できれば問題はありません。**

図 14-2　**フロー・ポイント出力のフロー（ハードウエア・フロー制御なしの場合）**



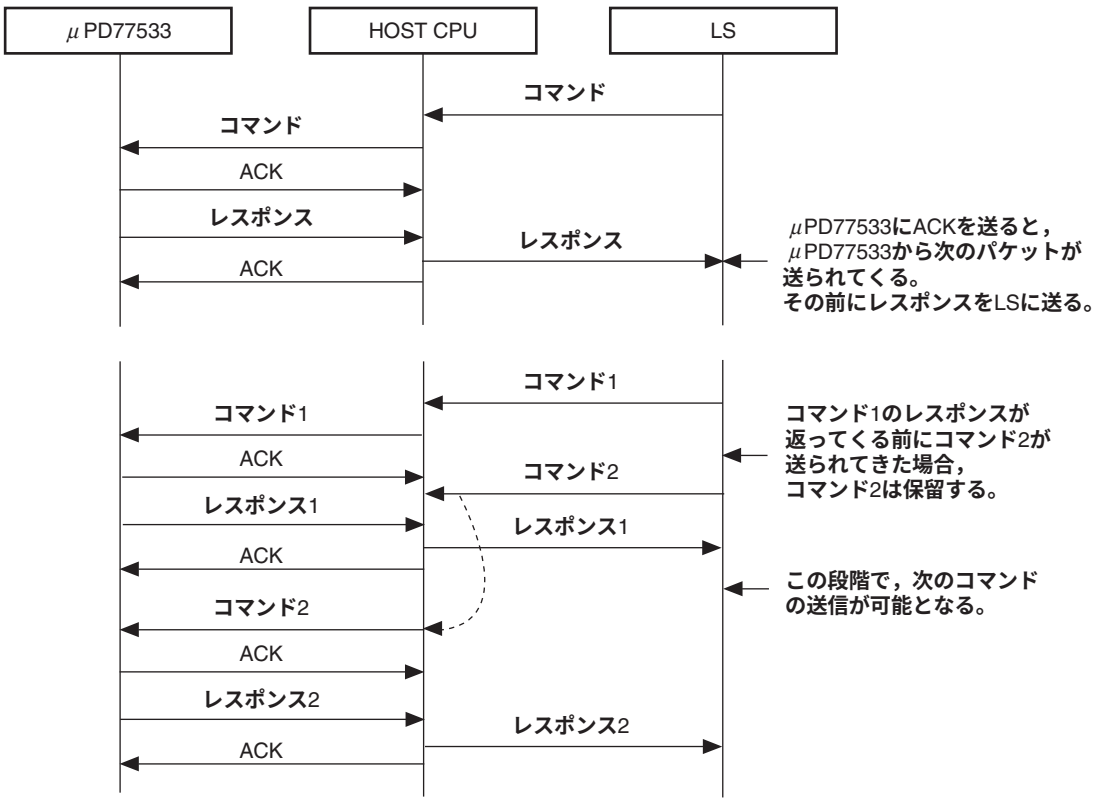
14. 1. 3 サーバ・コマンド

サーバ・コマンドをμPD77533 に送信する場合，コマンドのレスポンスが返ってくるまで，次のコマンドを送信しないでください。

LS から複数のコマンドが連続で送られてくることがありますが，その場合は LS からのコマンドを HOST CPU 側でプールし，μPD77533 に送るコマンドは常に 1 つずつになるようにしてください。

サーバ・コマンドに対するレスポンスは，ACK，NACK でパケット送受信の管理が可能であるため，HOST CPU 側に 1 パケット分のバッファを用意し，μPD77533 からのパケットを遅滞なく受信できれば問題はありません。

図 14-3 サーバ・コマンドのフロー（ハードウエア・フロー制御なしの場合）



15. サーバ・コマンド

μPD77533 で対応するサーバ・コマンドは，次のとおりです。

表 15-1 サーバ・コマンド一覧

コマンド名	コマンド ID	用　途
Module ID/Version	CA，ca	UnitID，ソフトウエアのバージョン取得
Module Processing Status Request	BB，bb	LSI，測位の状態の問い合わせ
Module Reset	IA，ia	GPS モジュールのリセット
Send Processed Result	hd	測位結果の通知
Multi Fix	JM，jm	測位指示
Ping for Coarse Time Synchronization	PJ，pj	粗時間同期。測位セッションの最初のコマンド
Set Adaptive Parameters	AC，ac	測位に関する各種パラメータの設定
Set SnapShot Duration	DA，da	μPD77533 の機能検証用に使用されるコマンド[注1]（SnapShot 時間の設定）
Take SnapShot	EA，ea	μPD77533 の機能検証用に使用されるコマンド[注1]（SnapShot の実行）
Send SnapShot Data	FA，FC，fa，fb，fc	μPD77533 の機能検証用に使用されるコマンド[注1]（SnapShot データ（1 ms Frame）の取得）
Set Baud Rate	BD，bd	HOST CPU－LS 間の接続速度の設定[注2]
Request for Service	CB，cb	端末発呼の測位開始の通知

注 1. これらのコマンドは，通常の測位をする場合には必要ありません。

2. このコマンドは，HOST CPU が処理するコマンドです。μPD77533 がコマンド処理する必要はありません。

15. 1 HOST CPU でのサーバ・コマンドの扱い方

サーバ・コマンドに関しては，HOST CPU は一部のコマンドを除きμPD77533－LS 間の通信の仲介だけを行います。具体的には，次の処理を行います。

- LS から送られてきたサーバ・コマンドをパケット化し，μPD77533 に転送する。
- μPD77533 から送られてきたレスポンスを非パケット化し，LS に転送する。
- ボー・レート変更のサーバ・コマンドを解析し，ボー・レート変更処理を行い，レスポンスを返す。
- μPD77533－HOST CPU 間の通信で ACK，NACK の処理を行う。

これを実現するために，HOST CPU はサーバ・コマンド，レスポンスの解析を行う必要があります。

15. 1. 1　**サーバ・コマンドの解析**

LS から送られてきたサーバ・コマンドは，次のように解析されます。

（1）コマンドのヘッダが「$」文字であることを確認する。

（2）コマンド ID が有効であることを確認する。チェック・サムの計算を開始する。

（3）表 15-2　データ・フィールドのバイト数の求め方にしたがって，データ・フィールドのバイト数を計算する。

（4）（3）で計算したバイト数分を読み込み，チェック・サムの計算を行う。

（5）チェック・サムが計算値と一致するか確認する。

（6）終端が<CR><LF>か確認する。

これらの（1）～（6）がすべて OK であれば，それを 1 コマンドとみなします。

ボー・レート変更コマンド以外なら，パケット化してμPD77533 に転送します。

いずれかの段階でエラーが起きた場合は，1 バイト破棄して（1）から繰り返します。

表 15-2　データ・フィールドのバイト数の求め方

コマンド ID	データ・フィールドのバイト数の求め方
CA	0 バイト固定
BB	1 バイト固定
IA	1 バイト固定
JM	データ・フィールドの最初の 2 バイトに格納されている値−7 バイト
PJ	1 バイト固定
AC	データ・フィールドの最初の 2 バイトに格納されている値−7 バイト
DA	5 バイト固定
EA	0 バイト固定
FA	4 バイト固定
FC	0 バイト固定
BD	1 バイト固定
CB	1 バイト固定
ca	9 バイト固定
bb	1 バイト固定
ia	1 バイト固定
hd	データ・フィールドの最初の 2 バイトに格納されている値−7 バイト
jm	1 バイト固定
pj	4 バイト固定
ac	データ・フィールドの最初の 2 バイトに格納されている値−7 バイト
da	1 バイト固定
ea	1 バイト固定
fa	3 バイト固定
fb	直前の!fa の結果コードが 0 の場合：2050 バイト固定 直前の!fa の結果コードが 1 の場合：1026 バイト固定
fc	1 バイト固定
bd	2 バイト固定
cb	10 バイト固定

15. 1. 2 **ボー・レート変更コマンド（**“Set Baud Rate”**）への対応**

ボー・レート変更コマンドの場合，次のように処理します。

（1）コマンドに対するレスポンスを作成し，LS **に送信する。**

（2）レスポンスの送信が完了するのを待つ。

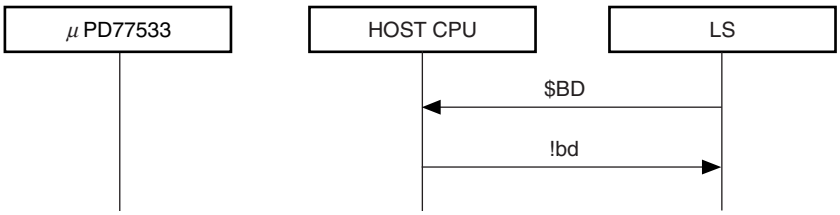
（3）コマンドにしたがって，ボー・レートを変更する。

15. 2 Set Baud Rate

HOST CPU－LS **間のボー・レートを変更します。**

このコマンドは HOST CPU **が処理しなければなりません。**μPD77533 **はこのコマンドを処理しません。**

図 15-1 Set Baud Rate **のフロー**



15. 2. 1 $BD

表 15-3 $BD **のフォーマット（非パケット状態）**

セクション		ビット数	形式	内　容
ヘッダ		8	－	$
コマンド ID		16	－	BD
データ	ボー・レート	8	unsigned	**表** 15-5 **ボー・レート一覧を参照。**
チェック・サム		16	－	チェック・サム
終端		16	－	<0x0D><0x0A>

15. 2. 2 !bd

表 15-4 !bd **のフォーマット（非パケット状態）**

セクション		ビット数	形式	内　容
ヘッダ		8	－	!
コマンド ID		16	－	bd
データ	結果コード	8	unsigned	0 = OK Else = エラー
	ボー・レート	8	unsigned	**表** 15-5 **ボー・レート一覧を参照。**
チェック・サム		16	－	チェック・サム
終端		16	－	<0x0D><0x0A>

15. 2. 3 ボー・レート一覧

表 15-5 ボー・レート一覧

番　　号	ボー・レート
0	2400 bps
1	4800 bps
2	9600 bps
3	19200 bps
4	38400 bps
5	57600 bps
Else	115200 bps

備考 μPD77533－HOST CPU 間のボー・レートは，ローカル・コマンド16. 1 Baud Rate Changeで変更します。

16. ローカル・コマンド

μPD77533 プログラムで対応するローカル・コマンドを，次に示します。

表 16-1 ローカル・コマンド一覧

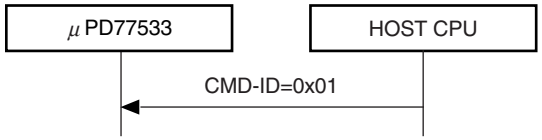
コマンド	CMD-ID	用　　途
Baud Rate Change	0x01	μPD77533－HOST CPU 間のボー・レートの設定
FCC Reference Frequency	0x02	FCC レファレンス周波数の設定，参照
Power Control	0x03	省電力の設定
Reset	0x04	μPD77533 をリセット
RF Power Control	0x05	RF 電源の設定
Frequency Calibration Control	0x06	FCC 制御の設定，参照
TXACTV Control	0x07	TXACVT 制御の設定，参照
Sleep Timer	0x12	スリープ・タイマの設定，参照
Characer Timeout	0x13	キャラクタ・タイムアウトの設定，参照
!cb Request	0x20	!cb レスポンスの発行要求
!ia Request	0x21	!ia レスポンスの発行要求
SnapShot Memory Check	0x2D	SnapShot メモリのチェック
Status Request	0x30	シリアル通信の動作確認
D77533 Software Version	0x31	μPD77533 のバージョンの参照
Processing Status Request	0x33	信号処理状態の参照
Rest Processing	0x34	測位処理の経過の参照
Error Status	0x3F	エラーの通知
CA Parameter Set	0x40	!ca レスポンスのパラメータの設定
CB Parameter Set	0x41	!cb レスポンスのパラメータの設定
Doppler Search Range	0x57	Doppler Search 範囲の拡大

16. 1　Baud Rate Change

[機　　能]

μPD77533－HOST CPU **間のボー・レートを変更する。**

図 16-1　Baud Rate Change **のフロー**



[コマンド]

表 16-2　Baud Rate Change **のフォーマット**

フィールド		バイト数	形　　式	内　　容
CMD-ID		1	－	0x01
LENGTH		1	－	1
DATA	ボー・レート	1	unsigned	0 = 2400 bps, 1 = 4800 bps, 2 = 9600 bps (Default), 3 = 19200 bps, 4 = 38400 bps, 5 = 57600 bps, Else = 115200 bps
FLAG		1	－	0x03

[レスポンス]

μPD77533 **はレスポンスを返しません。**

[備　　考]

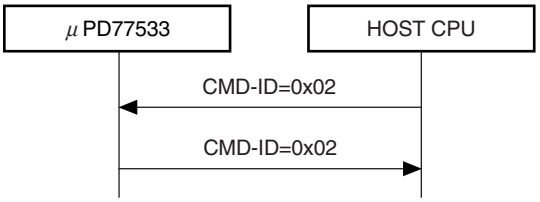
μPD77533 **はこのコマンドが正常に実行できた場合，レスポンスを返しません。**

16. 2　FCC Reference Frequency

[機　　能]

周波数補正のためのレファレンス周波数を設定，参照します。

図 16-2　FCC Reference Frequency **のフロー**



[コマンド]

表 16-3　FCC Reference Frequency **のフォーマット**

フィールド		バイト数	形式	内　　容
CMD-ID		1	−	0x02
LENGTH		1	−	4 = **レファレンス周波数の設定。** 0 = **現在のレファレンス周波数を参照。**
DATA	**レファレンス周波数** (LENGTH = 4 **のとき**)	4	unsigned	**レファレンス周波数** **設定する値は，周波数に** 1.024 **を掛けた値であること。** Default = 0xE10000（14.4 MHz×1.024）
FLAG		1	−	0x03

[レスポンス]

設定した値，または現在の値を表 16-3**のフォーマットで返します。**

[関連コマンド]

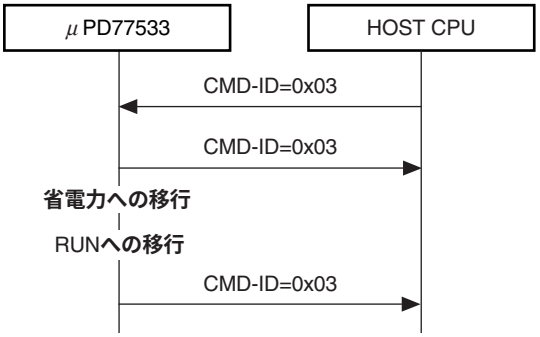
16. 6　Frequency Calibration Control

16. 3　Power Control

[機　　能]

省電力モードへの移行を行います。

図 16-3　Power Control **のフロー**



[コマンド]

表 16-4　Power Control **のフォーマット**

フィールド		バイト数	形式	内　　容
CMD-ID		1	−	0x03
LENGTH		1	−	1
DATA	モード	1	unsigned	0 = RUN へ移行（Default） 1 = HALT へ移行 Else = STOP へ移行
FLAG		1	−	0x03

[レスポンス]

移行する状態を表 16-4**のフォーマットで返します。**

[備　　考]

Sleep Timer **で自動的に省電力へ移行するときは，レスポンスを返しません。**

省電力からの復帰時も Power Control **のレスポンスを** DATA = 0x00 **で返します。**

16. 4　Reset

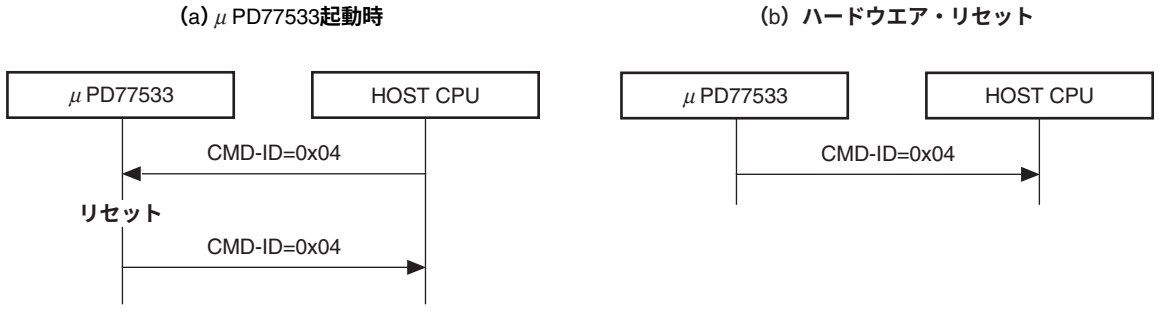
[機　　能]

μPD77533 **をリセットします。**

ボー・レート以外のすべての設定を初期化します。

μPD77533 **起動時，ハードウエアのリセット時にも** Reset **のレスポンスを返します。**

図 16-4　Reset **のフロー**



[コマンド]

表 16-5　Reset **のフォーマット**

フィールド	バイト数	形式	内　　容
CMD-ID	1	−	0x04
LENGTH	1	−	0
FLAG	1	−	0x03

[レスポンス]

移行する状態を表 16-5**のフォーマットで返します。**

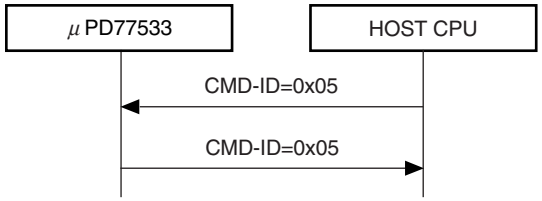
16. 5　RF Power Control

[機　　能]

RF **部の電源の** ON/OFF **を行います。**

通常の測位においては，RF **部の電源** ON/OFF **はロジックが行うため，ソフトウエアで制御する必要はありません。**

図 16-5　RF Power Control **のフロー**



[コマンド]

表 16-6　RF Power Control **のフォーマット**

フィールド		バイト数	形式	内容
CMD-ID		1	−	0x05
LENGTH		1	−	1
DATA	**電源**	1	unsigned	0 = **電源** OFF（Default），Else = **電源** ON
FLAG		1	−	0x03

[レスポンス]

設定した値を表 16-6**のフォーマットで返します。**

16. 6　Frequency Calibration Control

[機　　能]

FCC **制御の有無を設定，参照します。**

図 16-6　Frequency Calibration Control **のフロー**

[コマンド]

表 16-7　Frequency Calibration Control **のフォーマット**

フィールド		**バイト数**	**形式**	**内　　容**
CMD-ID		1	−	0x06
LENGTH		1	−	1 = FCC **制御の有無を設定** 0 = **現在の** FCC **制御の有無を参照**
DATA	FCC **制御** （LENGTH = 1 **のとき**）	1	unsigned	0 = FCC **制御なし**（Default） Else = FCC **制御あり**
FLAG		1	−	0x03

[レスポンス]

設定した値，または現在の値を表 16-7**のフォーマットで返します。**

[関連コマンド]

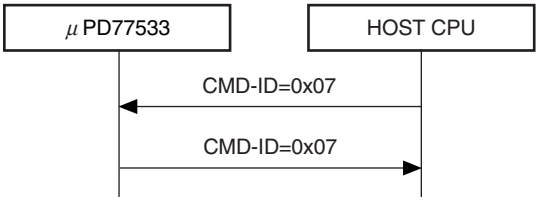
16. 2　FCC Reference Frequency

16. 7　TXACTV Control

[機　　能]

TXACTV **制御の有無を設定，参照します。**

図 16-7　TXACTV Control **のフロー**



[コマンド]

表 16-8　TXACTV Control **のフォーマット**

フィールド		バイト数	形式	内　　容
CMD-ID		1	−	0x07
LENGTH		1	−	1 = TXACTV **制御の有無を設定** 0 = **現在の** TXACTV **制御の有無を参照**
DATA	TXACTV **制御** **（**LENGTH = 1 **のとき）**	1	unsigned	0 = TXACTV **制御なし**　**（**Default**）** Else = TXACTV **制御あり**
FLAG		1	−	0x03

[レスポンス]

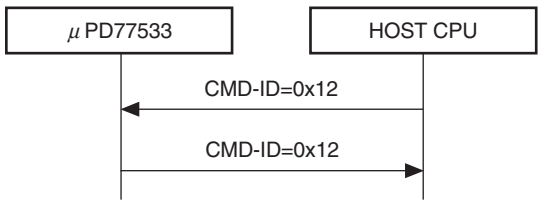
設定した値，または現在の値を表 16-8**のフォーマットで返します。**

16. 8　Sleep Timer

[機　　能]

Sleep Timer **を設定，参照します。**

図 16-8　Sleep Timer **のフロー**



[コマンド]

表 16-9　Sleep Timer **のフォーマット**

フィールド		バイト数	形式	内　　容
CMD-ID		1	−	0x12
LENGTH		1	−	4 = SleepTimer **を設定する** 0 = **現在の** SleepTimer **を参照する**
DATA	**モード** **（**LENGTH = 4 **のとき）**	1	unsigned	0 = HALT **へ移行** Else = STOP **へ移行**
	Sleep **までの時間** **（**LENGTH = 4 **のとき）**	3	unsigned	Sleep **へ移行するまでの時間。**ms **単位。** 0 = Sleep **へ移行しない（**Default**）**
FLAG		1	−	0x03

[レスポンス]

設定した値，または現在の値を表 16-9**のフォーマットで返します。**

[備　　考]

μ PD77533 **は測位，コマンドの総受信などを行わない状態のまま，指定時間経過した場合** Sleep **に移行します。**

HALT **からは，コマンドの受信，**WKUPB **信号，リセット，タイマ割り込みなどで復帰します。**

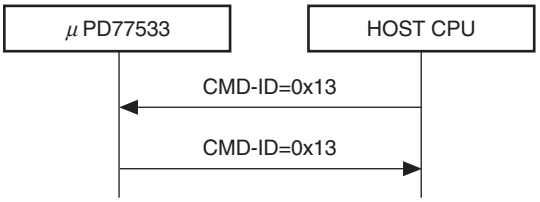
STOP **からは，**WKUPB **信号で** RUN **状態に復帰し，リセットで全初期化，再起動します。**

16. 9　Character Timeout

[機　　能]

Character **のタイムアウト時間を設定，参照します。**

図 16-9　Character Timeout **のフロー**



[コマンド]

表 16-10　Character Timeout **のフォーマット**

フィールド		バイト数	形式	内　　容
CMD-ID		1	－	0x13
LENGTH		1	－	3 = **タイムアウト時間を設定する** 0 = **現在のタイムアウト時間を参照する**
DATA	**タイムアウトまでの時間**（LENGTH = 3 **のとき**）	3	unsigned	**タイムアウトになるまでの時間。**ms **単位。** 0 = **タイムアウト検出しない**（Default）
FLAG		1	－	0x03

[レスポンス]

設定した値，または現在の値を表 16-10**のフォーマットで返します。**

[備　　考]

μPD77533 **はパケットの受信が完了しない段階で，パケットの最後のデータを受信してから指定時間経過した場合，タイムアウトになります。**

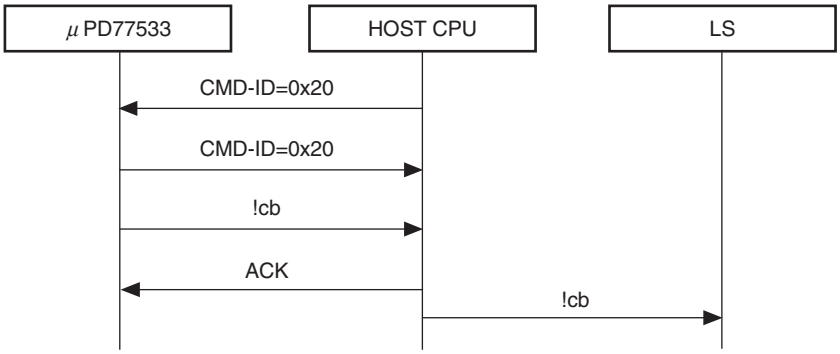
タイムアウトになったとき，μPD77533 **はタイムアウトのエラー・ステータスを返し，それまでに受信したパケットのデータを破棄します。**

16. 10　!cb Request

[機　　能]

!cb **レスポンスの発行を要求します。**

図 16-10　!cb Request **のフロー**



[コマンド]

表 16-11　!cb Request **のフォーマット**

フィールド	バイト数	形式	内　　容
CMD-ID	1	−	0x20
LENGTH	1	−	0
FLAG	1	−	0x03

[レスポンス]

表 16-11**のフォーマット・レスポンスを返します。**

μPD77533 **はレスポンスに続いて，**!cb **を返します。**

[備　　考]

端末側から測位の要求を行う場合，このコマンドを使うことができます。コマンド"!cb Request"**を受信した** μPD77533 **は，**!cb **レスポンスを** LS **に送ります。**LS **は**!cb **を受信したあと，測位を開始します。**

LS **から測位を要求する場合は，このコマンドを使用する必要はありません。**

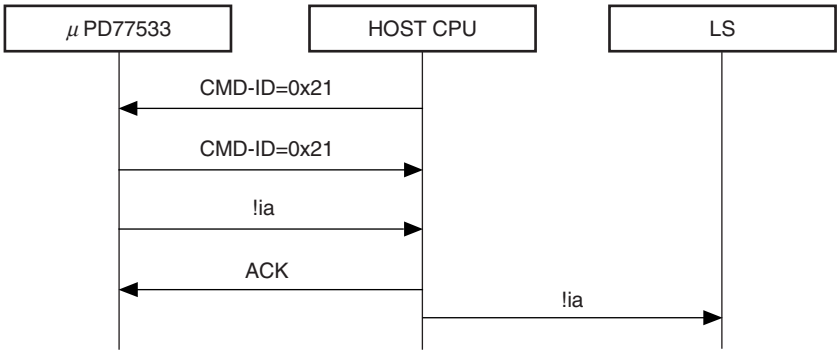
なお，端末側から測位を開始する場合の実際の制御フローはシステムに依存します。

16. 11　!ia Request

[機　　能]

!ia **レスポンスの発行を要求します。**

図 16-11　!ia Request **のフロー**



[コマンド]

表 16-12　!ia Request **のフォーマット**

フィールド	バイト数	形式	内　　容
CMD-ID	1	−	0x21
LENGTH	1	−	0
FLAG	1	−	0x03

[レスポンス]

表 16-12**のフォーマット・レスポンスを返します。**

レスポンスに続いて，!ia **レスポンスを返します。**!ia **の結果コードは** 0x09 **です。**

[備　　考]

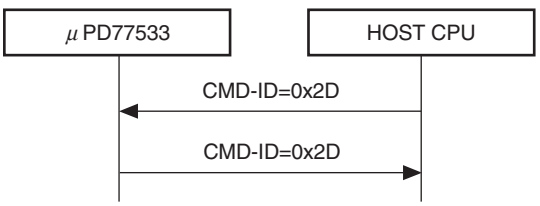
このコマンドの使用方法は，システムに依存します。

16. 12　SnapShot Memory Check

[機　　能]

SnapShot **メモリのチェックを行います。**

図 16-12　SnapShot Memory Check **のフロー**



[コマンド]

表 16-13　SnapShot Memory Check **のコマンド・フォーマット**

フィールド	バイト数	形式	内　容
CMD-ID	1	−	0x2D
LENGTH	1	−	0
FLAG	1	−	0x03

[レスポンス]

SnapShot **メモリにエラーがないときは，表** 16-13**のフォーマットでレスポンスを返します。エラーがあるときは，次のフォーマットでエラーを返します。**

表 16-14　SnapShot Memory Check **のレスポンス・フォーマット（エラー時）**

フィールド		バイト数	形式	内　容
CMD-ID		1	−	0x2D
LENGTH		1	−	7
DATA	**アドレス**	3	unsigned	**エラーが起きた** SnapShot **メモリのアドレス**（0x00000〜0xFFFFF）
	書き込み値	2	−	SnapShot **メモリに書き込んだ値**
	読み出し値	2	−	SnapShot **メモリから読み出した値**
FLAG		1	−	0x03

[備　考]

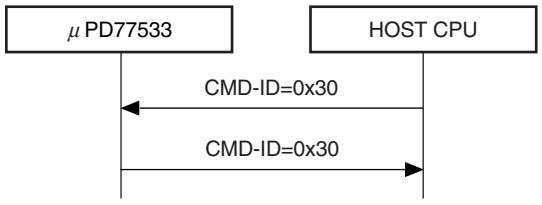
SnapShot **メモリの全エリアに対して，**0x0000，0xAAAA，0xFFFF，0x5555 **の値を書き込み，正しく読み出せるかをチェックします。**

16. 13　Status Request

[機　能]

μPD77533 **の動作確認を行います。**

図 16-13　Status Request **のフロー**



[コマンド]

表 16-15　Status Request **のコマンド・フォーマット**

フィールド	バイト数	形式	内　容
CMD-ID	1	−	0x30
LENGTH	1	−	0
FLAG	1	−	0x03

[レスポンス]

表 16-15**のフォーマットでレスポンスを返します。**

[備　　考]

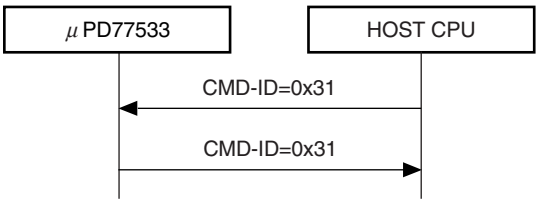
このコマンドはレスポンスを返すだけです。ほかに何も行いません。

16. 14　D77533 Software Version

[機　　能]

μPD77533 **のソフトウエアのバージョンを参照します。**

図 16-14　D77533 Software Version **のフロー**



[コマンド]

表 16-16　D77533 Software Version **のコマンド・フォーマット**

フィールド	バイト数	形式	内　　容
CMD-ID	1	−	0x31
LENGTH	1	−	0
FLAG	1	−	0x03

[レスポンス]

表 16-17　D77533 Software Version **のレスポンス・フォーマット**

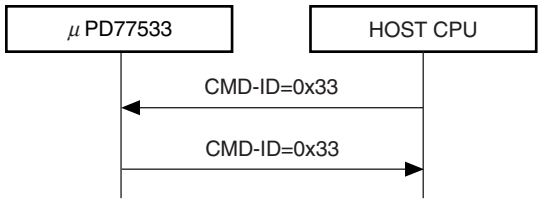
フィールド		バイト数	形式	内　　容
CMD-ID		1	−	0x31
LENGTH		1	−	2
DATA	バージョン	1	−	**上位** 4 **ビット：** 10 **の位** **下位** 4 **ビット：** 1 **の位**
		1	−	**上位** 4 **ビット：** 0.1 **の位** **下位** 4 **ビット：** 0.01 **の位**
FLAG		1	−	0x03

16. 15　Processing Status Request

[機　　能]

μPD77533 **の信号処理状態を参照します。**

図 16-15　Processing Status Request **のフロー**



[コマンド]

表 16-18　Processing Status Request **のコマンド・フォーマット**

フィールド	バイト数	形式	内　　容
CMD-ID	1	−	0x33
LENGTH	1	−	0
FLAG	1	−	0x03

[レスポンス]

表 16-19　Processing Status Request **のレスポンス・フォーマット**

フィールド		バイト数	形式	内　　容
CMD-ID		1	−	0x33
LENGTH		1	−	1
DATA	**ステータス**	1	unsigned	0x00 = SnapShot **実行中** 0x10 = **信号処理中** Else = Idle **中**
FLAG		1	−	0x03

16. 16　Rest Processing

[機　　能]

測位時の処理すべき残り衛星数，測位回数を参照します。

図 16-16　Rest Processing **のフロー**

[コマンド]

表 16-20　Rest Processing **のコマンド・フォーマット**

フィールド	バイト数	形式	内　　容
CMD-ID	1	−	0x34
LENGTH	1	−	0
FLAG	1	−	0x03

[レスポンス]

表 16-21　Rest Processing **のレスポンス・フォーマット**

フィールド		バイト数	形式	内　　容
CMD-ID		1	−	0x34
LENGTH		1	−	4
DATA	**処理済みの衛星数**	1	unsigned	1〜32
	処理すべき全衛星数	1	unsigned	1〜32
	現在の測位回数	1	unsigned	0〜
	全測位回数	1	unsigned	0 = **制限なし** 1〜 = **回数**
FLAG		1	−	0x03

16. 17　Error Status

[機　　能]

エラーを通知します。

[コマンド]

表 16-22　Error Status **のフォーマット**

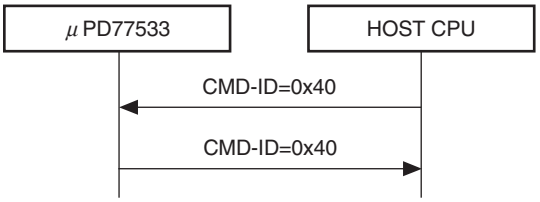
フィールド	バイト数	形式	内　　容
CMD-ID	1	−	0x3F
LENGTH	1	−	1
DATA	1	unsigned	0 = OK（ACK） 1 = **フォーマット・エラー** 2 = **タイムアウト・エラー** 4 = **未定義** CMD-ID **エラー** Else = **未定義**
FLAG	1	−	0x03

16. 18　CA Parameter Set

[機　　能]

!ca **のパラメータを設定，参照します。**

図 16-17　CA Parameter Set **のフロー**



[コマンド]

表 16-23　CA Parameter Set **のフォーマット**

フィールド		バイト数	形式	内　　容
CMD-ID		1	－	0x40
LENGTH		1	－	8 = !ca **のパラメータを設定する** 0 = **現在の**!ca **のパラメータを参照する**
DATA	UnitID （LENGTH = 8 **のとき**）	1	－	UnitID（7:0）
		1	－	UnitID（15:8）
		1	－	UnitID（23:16）
		1	－	UnitID（31:24）
	バージョン （LENGTH = 8 **のとき**）	1	－	**バージョン**（7:0）
		1	－	**バージョン**（15:8）
		1	－	**バージョン**（23:16）
		1	－	**バージョン**（31:24）
FLAG		1	－	0x03

[レスポンス]

設定した値，または現在の値を表 16-23**のフォーマットで返します。**

[備　　考]

UnitID **およびバージョンの値はシステムに依存します**（Custmer Default）。

16. 19　CB Parameter Set

[機　　能]

!cb **のパラメータを設定，参照します。**

図 16-18　CB Parameter Set **のフロー**

[コマンド]

表 16-24　CB Parameter Set **のフォーマット**

フィールド		バイト数	形式	内　　容
CMD-ID		1	－	0x41
LENGTH		1	－	10 = !cb **のパラメータを設定する** 0 = **現在の**!cb **のパラメータを参照する**
DATA	**タイプ** **（**LENGTH = 10 **のとき）**	1	－	**タイプ**（7:0）
		1	－	**タイプ**（15:8）
	UnitID **（**LENGTH = 10 **のとき）**	1	－	UnitID（7:0）
		1	－	UnitID（15:8）
		1	－	UnitID（23:16）
		1	－	UnitID（31:24）
	バージョン **（**LENGTH = 10 **のとき）**	1	－	**バージョン**（7:0）
		1	－	**バージョン**（15:8）
		1	－	**バージョン**（23:16）
		1	－	**バージョン**（31:24）
FLAG		1	－	0x03

[レスポンス]

設定した値，または現在の値を表 16-24**のフォーマットで返します。**

[備　　考]

タイプ，UnitID **およびバージョンの値は，システムに依存します（**Customer Default)。

16. 20　Doppler Search Range

[機　　能]

Doppler Search **範囲のパラメータを設定，参照します。**

図 16-19　Doppler Search Range **のフロー**

[コマンド]

表 16-25　Doppler Search Range **のフォーマット**

フィールド		バイト数	形式	内　　容
CMD-ID		1	－	0x57
LENGTH		1	－	2 = **パラメータを設定する** 0 = **現在のパラメータを参照する**
DATA	Doppler Search Range	2	－	Default Doppler Search Range **初期値：**0x0168
FLAG		1	－	0x03

[レスポンス]

設定した値，または現在の値を表 16-25**のフォーマットで返します。**

[備　　考]

精度が 0.1 ppm **のローカル周波数（**GPCLK **入力クロック）を使用する場合，パラメータ** Doppler Search Range **の設定値は初期値のままです。**

パラメータ Doppler Search Range **の設定値を初期値以外に変更する場合の計算方法は，**6. 1. 3　Doppler Search Range**で説明します。**

17. エクステンション・コマンド

μPD77533 で対応するエクステンション・コマンドを，次に示します。

★ 表 17-1 エクステンション・コマンド一覧

コマンド	EXT-ID	用　途
Flow Point Control	0x00	フロー・ポイントの設定，参照
Processing End	0x06	全測位処理終了のフロー・ポイント出力

17. 1 Flow Point Control

[機　能]

フロー・ポイント出力の有無を設定します。

このコマンドの実行後，ACK は待ちません。

図 17-1 Flow Point Control のフロー

[コマンド]

★ 表 17-2 Flow Point Control のコマンド・フォーマット

フィールド		バイト数	形式	内　容
CMD-ID		1	−	0x7F
LENGTH		1	−	3
DATA	EXT-ID	1	−	0x00
	フロー・ポイント	1	−	0x00：Processing End を出力しない（Default） 0x10：Processing End を出力する
		1	−	0x00
FLAG		1	−	0x03

[レスポンス]

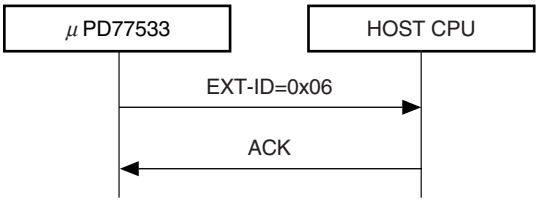
設定した値を表 17-2のフォーマットで返します。

17. 2　Processing End

[機　　能]

全測位の終了を通知します。

図 17-2　Processing End **のフロー**



[コマンド]

フロー・ポイントの出力には，コマンドはありません。

[レスポンス]

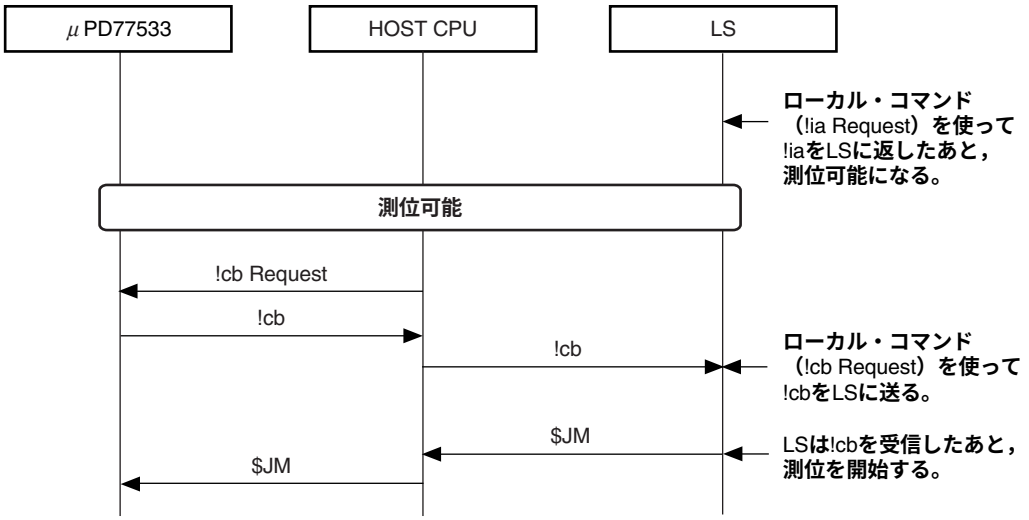
表 17-3　Processing End **のコマンド・フォーマット**

フィールド		バイト数	形式	内　　容
CMD-ID		1	−	0x7F
LENGTH		1	−	1
DATA	EXT-ID	1	−	0x06
FLAG		1	−	0x03

18. 端末から測位開始するときの HOST CPU の処理

HOST CPU と LS を直結するシステムにおいて，GPS 端末から測位トリガをかけて測位を開始するとき，HOST CPU は次の手順で処理を行う必要があります。

図 18-1　端末から測位開始するときの HOST CPU の処理手順



備考　GPS 端末を起動したときの HOST CPU 処理については，3. 3　**起動時に必要な処理**を参照してください。

19. 電気的特性

絶対最大定格

項　目		略　号	条　件	定　格	単位
電源電圧	内部電源電圧	IV_{DD}	コア用	−0.5〜+2.6	V
		AV_{DD}	PLL アナログ用	−0.5〜+2.6	V
	外部電源電圧	EV_{DD}	I/O 端子用	−0.5〜+4.6	V
入力／出力電圧		V_I /V_O		−0.5〜+4.1, $V_I < V_{DD}$+0.5	V
動作周囲温度		T_A		−20〜+85	°C
保存温度		T_{stg}		−65〜+150	°C

注意　各項目のうち 1 項目でも，また一瞬でも絶対最大定格を越えると，製品の品質を損なう恐れがあります。つまり絶対最大定格とは，製品に物理的な損傷を与えかねない定格値です。必ずこの定格値を越えない状態で，製品をご使用ください。

推奨動作範囲

項　目		略　号	条　件	MIN.	TYP.	MAX.	単位
動作電圧	内部電圧	IV_{DD}	コア用	1.425	1.5	1.65	V
		AV_{DD}	PLL アナログ用	1.425	1.5	1.65	V
	外部電圧	EV_{DD}	I/O 端子用	2.7	−	3.6	V
入力電圧		V_I		0	−	EV_{DD}	V

容量

項　目	略　号	条　件	MIN.	TYP.	MAX.	単位
入力容量	C_{IN}	f = 1 MHz， 測定端子以外は 0 V	−	−	10	pF
出力容量	C_{OUT}		−	−	10	pF
入出力容量	C_I /C_O		−	−	10	pF

DC特性（T_A＝－20～＋85℃，IV_{DD}，AV_{DD}＝1.425～1.65 V，EV_{DD}＝2.7～3.6 V）

項　目	略　号	条　件	MIN.	TYP.	MAX.	単位
ハイ・レベル入力電圧	V_{IHN}	シュミット入力端子以外	0.7 EV_{DD}	－	EV_{DD}	V
	V_{IHS}	シュミット入力端子	0.8 EV_{DD}	－	EV_{DD}	V
	V_{IHC}	クロック入力端子 GPSCLK, REFCLK	0.6 EV_{DD}	－	EV_{DD}	V
ロウ・レベル入力電圧	V_{IL}	クロック入力端子以外	0	－	0.2 EV_{DD}	V
	V_{ILC}	クロック入力端子 GPSCLK, REFCLK	0	－	0.4 EV_{DD}	V
ハイ・レベル出力電圧	V_{OH}	I_{OH} = －2.0 mA	0.7 EV_{DD}	－	－	V
		I_{OH} = －100 μA	0.8 EV_{DD}	－	－	
ロウ・レベル出力電圧	V_{OL}	I_{OL} = 2.0 mA	－	－	0.2 EV_{DD}	V
ハイ・レベル入力リーク電流	I_{LH}	V_I = EV_{DD}	0	－	10	μA
ロウ・レベル入力リーク電流	I_{LL}	V_I = EV_{DD}	－10	－	0	μA
プルアップ端子電流	I_{PUI}	0V<V_I<EV_{DD}	－250	－	0	μA
プルダウン端子電流	I_{PDI}	0V<V_I<EV_{DD}	0	－	250	μA
通常電源電流	I_{EVDD}	動作モード， T_{cyc} = 12.2 ns,	－	2 注1	－	mA
	I_{IVDD}	IV_{DD}, AV_{DD} = 1.5 V,	－	34 注1	38	mA
	I_{AVDD}	EV_{DD} =3.0 V	－	1 注1	2	mA
HALT モード時電源電流	I_{EVDDH}	HALT モード，T_{cyc} = 12.2 ns,	－	－	－	mA
	I_{IVDDH}	IV_{DD}, AV_{DD} = 1.5 V,	－	－	5 注2	mA
	I_{AVDDH}	EV_{DD} = 3.0 V	－	－		
STOP モード時電源電流	I_{EVDDS}	STOP モード，T_A = 25℃,	－	－	700 注3	μA
	I_{IVDDS}	IV_{DD}, AV_{DD} = 1.5 V,	－	－		
	I_{AVDDS}	EV_{DD} = 3.0 V	－	－		

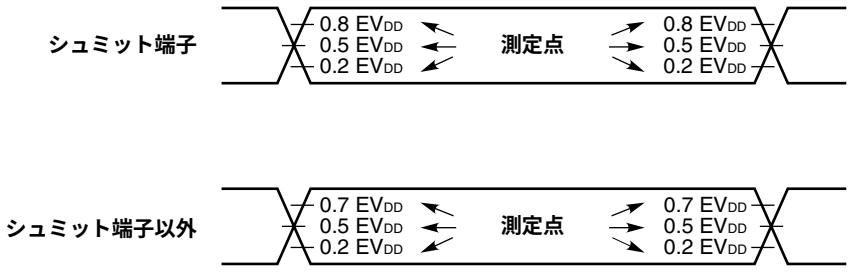
注 1. 参考値です。

2. I_{IVDDH} と I_{AVDDH} の合計値です。

3. I_{EVDDS}, I_{IVDDS}, I_{AVDDS} の合計値です。

AC特性

・テスト波形

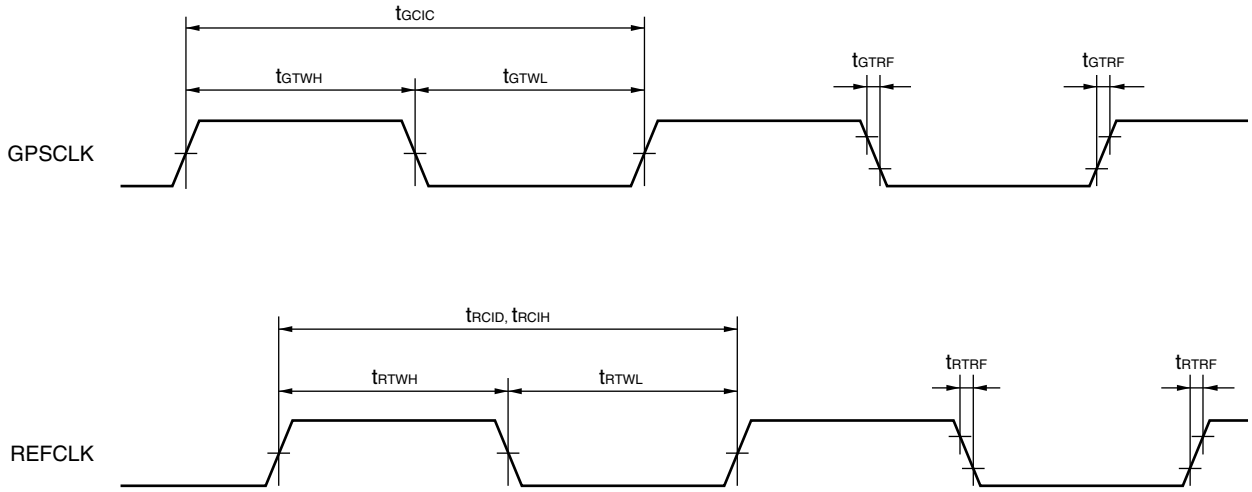


クロック・インタフェース

・クロック

項　　目	略　　目	条　　件	MIN.	TYP.	MAX.	単位
GPS クロック入力周期	t_{GCIC}	f = 16.384 MHz	61	−	−	ns
GPS クロック・ハイ・レベル幅	t_{GTWH}		t_{GCIC}÷2−3	−	−	ns
GPS クロック・ロウ・レベル幅	t_{GTWL}		t_{GCIC}÷2−3	−	−	ns
GPS クロック立ち上がり／立ち下がり時間	t_{GTRF}		−	−	5	ns
REF クロック入力周期	t_{RCID}	PDC レファレンス	−	14.4	−	MHz
	t_{RCIH}	PHS レファレンス	−	19.2	−	MHz
REF クロック・ハイ・レベル幅	t_{RTWH}		t_{RCID}/t_{RCIH}÷2−3	−	−	ns
REF クロック・ロウ・レベル幅	t_{RTWL}		t_{RCID}/t_{RCIH}÷2−3	−	−	ns
REF クロック立ち上がり／立ち下がり時間	t_{RTRF}		−	−	5	ns

・クロック・タイミング



システム・インタフェース

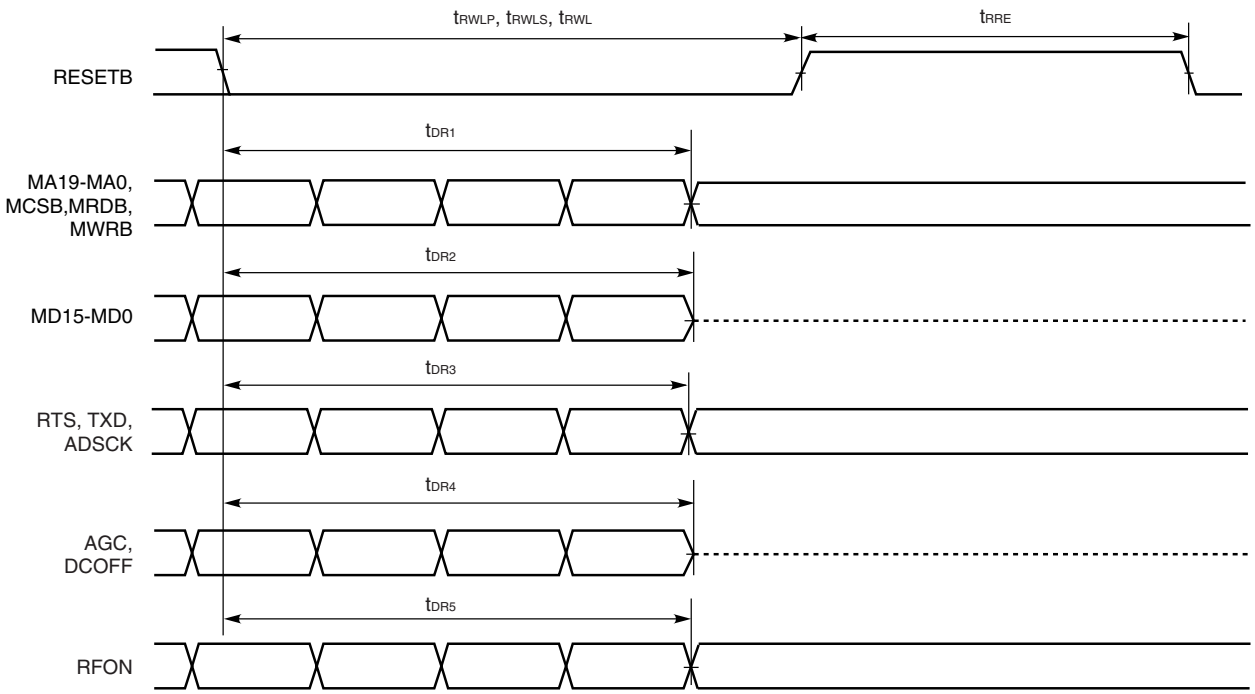
・リセット（RESETB）

項　目	略　号	条　件	MIN.	TYP.	MAX.	単位
RESETB ロウ・レベル幅	t_{RWLP}	電源投入時[注1]	73.2	−	−	ns
	t_{RWLS}	STOP モード時	73.2	−	−	ns
	t_{RWL}	通常モード時	73.2	−	−	ns
		HALT モード時	1170			
RESETB リカバリ時間	t_{RRE}	通常モード時	73.2	−	−	ns
		HALT モード時	1170			
出力初期化時間 1[注2]（対 RESETB↓）	t_{DR1}	MA19-MA0, MCSB, MRDB, MWRB	−	−	t_{GCIC}+30	ns
出力初期化時間 2[注2]（対 RESETB↓）	t_{DR2}	MD15-MD0	−	−	t_{GCIC}+30	ns
出力初期化時間 3[注2]（対 RESETB↓）	t_{DR3}	RTS, TXD, ADSCK	−	−	t_{GCIC}+30	ns
出力初期化時間 4[注2]（対 RESETB↓）	t_{DR4}	AGC, DCOFF	−	−	t_{GCIC}+30	ns
出力初期化時間 5[注2]（対 RESETB↓）	t_{DR5}	RFON	−	−	t_{GCIC}+30	ns

注 1．電源投入時は，EV_{DD}，IV_{DD}，AV_{DD} のすべてが保証動作電圧に達した時点を測定の起点とします。

2．RESET 投入から上記に記載される出力初期化時間までの間における出力端子信号は不定です。また，電源投入時，STOP モード時からの復帰は内部クロックが定常発振する時点を測定の起点とします。

・リセット・タイミング


tRWLP, tRWLS, tRWL
tRRE
RESETB
tDR1
MA19-MA0,
MCSB,MRDB,
MWRB
tDR2
MD15-MD0
tDR3
RTS, TXD,
ADSCK
tDR4
AGC,
DCOFF
tDR5
RFON

ウエイクアップ，Txアクティブ

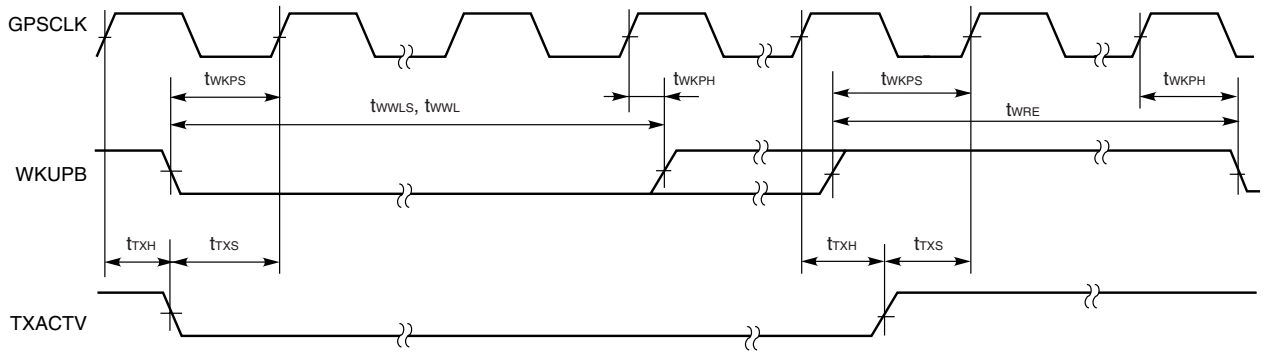
・ウエイクアップ（WKUPB）

項　目	略　号	条　件	MIN.	TYP.	MAX.	単位
WKUPB 設定時間（対 GPSCLK↑）	t_{WKPS}		0	−	−	ns
WKUPB 保持時間（対 GPSCLK↑）	t_{WKPH}		15.3	−	−	ns
WKUPB ロウ・レベル幅	t_{WWLS}	STOP モード時	73.2	−	−	ns
	t_{WWL}	通常モード時	73.2	−	−	ns
		HALT モード時	1170			
WKUPB リカバリ時間	t_{WRE}	通常時	73.2	−	−	ns
		HALT 時	1170	−	−	

・TXアクティブ

項　目	略　号	条　件	MIN.	TYP.	MAX.	単位
TXACTV 設定時間（対 GPSCLK↑）	t_{TXS}		0	−	−	ns
TXACTV 保持時間（対 GPSCLK↑）	t_{TXH}		t_{GCIC}	−	−	ns

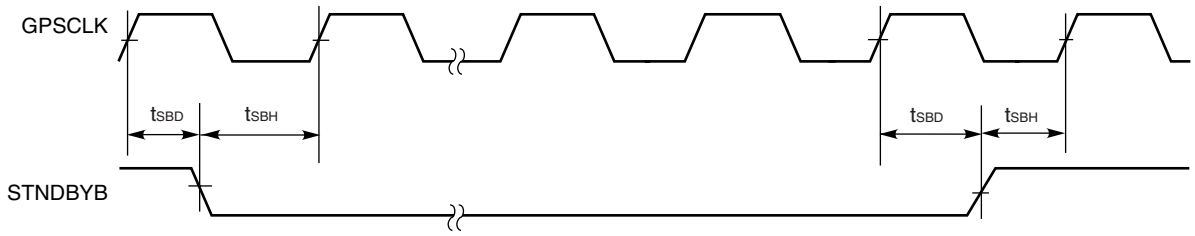
・ウエイクアップ，TXアクティブ・タイミング



・スタンバイ

項　目	略　号	条　件	MIN.	TYP.	MAX.	単位
STNDBYB 遅延時間（対 GPSCLK↑）	t_{SBD}		−	−	20	ns
STANDBYB 保持時間（対 GPSCLK↑）	t_{SBH}		0	−	−	ns

・スタンバイ・タイミング

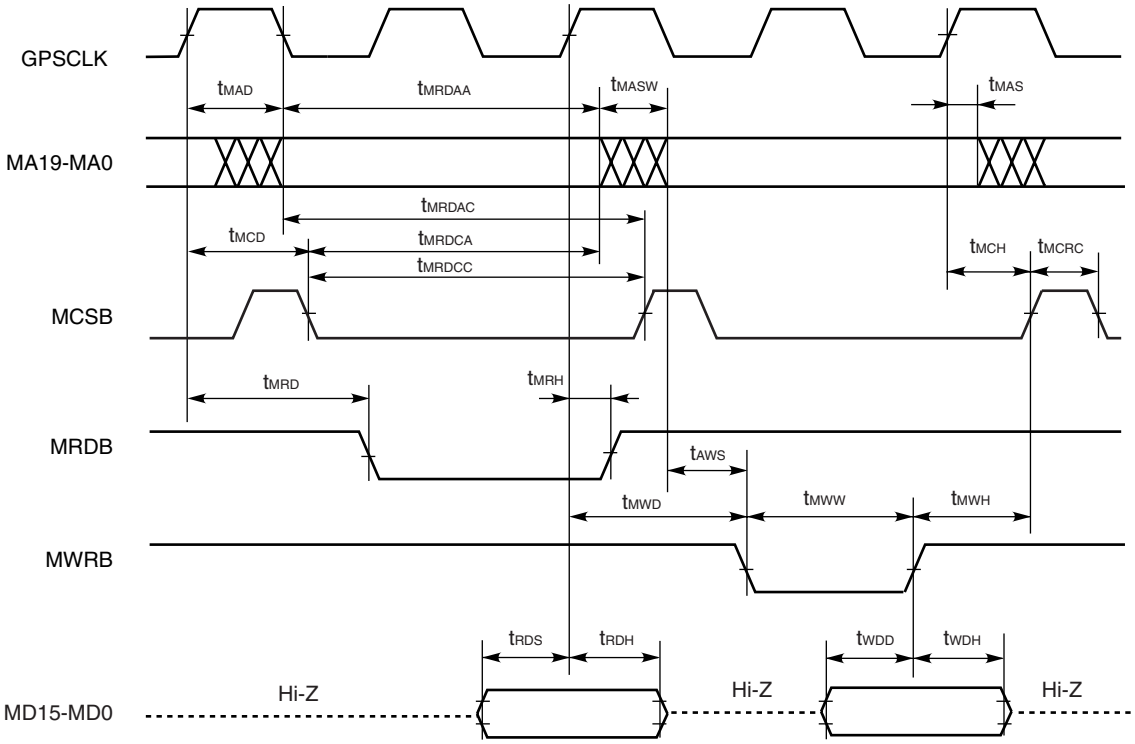


メモリ・インタフェース

・モバイル用途 RAM

項　目	略　号	条　件	MIN.	TYP.	MAX.	単位
アクセス・サイクル時間（MA to MA）	t_{MRDAA}		122	−	1000	ns
アクセス・サイクル時間（MA to MCSB↑）	t_{MRDAC}		122	−	1000	ns
アクセス・サイクル時間（MCSB ↓ to MA）	t_{MRDCA}		122	−	1000	ns
アクセス・サイクル時間（MCSB ↓ to MCSB↑）	t_{MRDCC}		122	−	1000	ns
アドレス（MA）遅延時間	t_{MAD}		−	−	10	ns
アドレス（MA）保持時間	t_{MAS}		0	−	−	ns
アドレス（MA）スキュー時間	t_{MASW}		−	−	10	ns
チップ・セレクト遅延時間	t_{MCD}		−	−	10	ns
チップ・セレクト保持時間	t_{MCH}		0	−	−	ns
チップ・セレクト・リカバリ時間	t_{MCRC}		10	−	−	ns
リード・ストローブ（MRDB）遅延時間	t_{MRD}		−	−	10	ns
リード・ストローブ（MRDB）保持時間	t_{MRH}		0	−	−	ns
ライト・ストローブ（MWRB）遅延時間	t_{MWD}	t_{AWS} を満足させる	−	−	10	ns
ライト・ストローブ（MWRB）保持時間	t_{MWH}		10	−	−	ns
ライト・ストローブ（MWRB）幅	t_{MWW}		30	−	−	ns
アドレス設定時間（対 MWRB↓）	t_{AWS}		0	−	−	ns
リード・データ（MD）設定時間	t_{RDS}		20	−	−	ns
リード・データ（MD）保持時間	t_{RDH}		0	−	−	ns
ライト・データ（MD）遅延時間	t_{WDD}		−	−	20	ns
ライト・データ（MD）保持時間	t_{WDH}		0	−	−	ns

・**モバイル用途**RAM**インタフェース・タイミング**

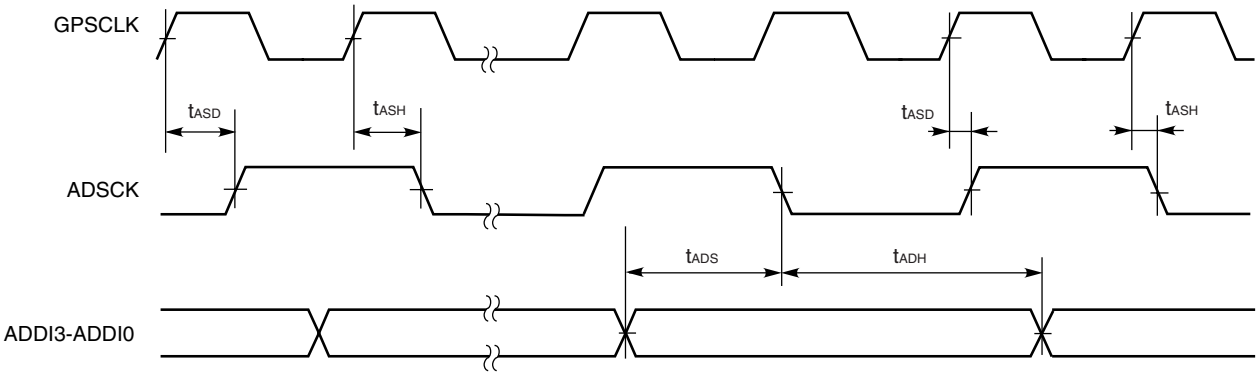


A/D**コントロール・インタフェース**

・AD**クロック，**AD**データ**

項　目	略　号	条　件	MIN.	TYP.	MAX.	単位
ADSCK **遅延時間（対** GPSCLK↑**）**	t_{ASD}		−	−	20	ns
ADSCK **保持時間（対** GPSCLK↑**）**	t_{ASH}		0	−	−	ns
ADDI **設定時間（対** ADSCK↓**）**	t_{ADS}		20	−	−	ns
ADDI **保持時間（対** ADSCK↓**）**	t_{ADH}		5	−	−	ns

・AD**クロック，**AD**データ・タイミング**

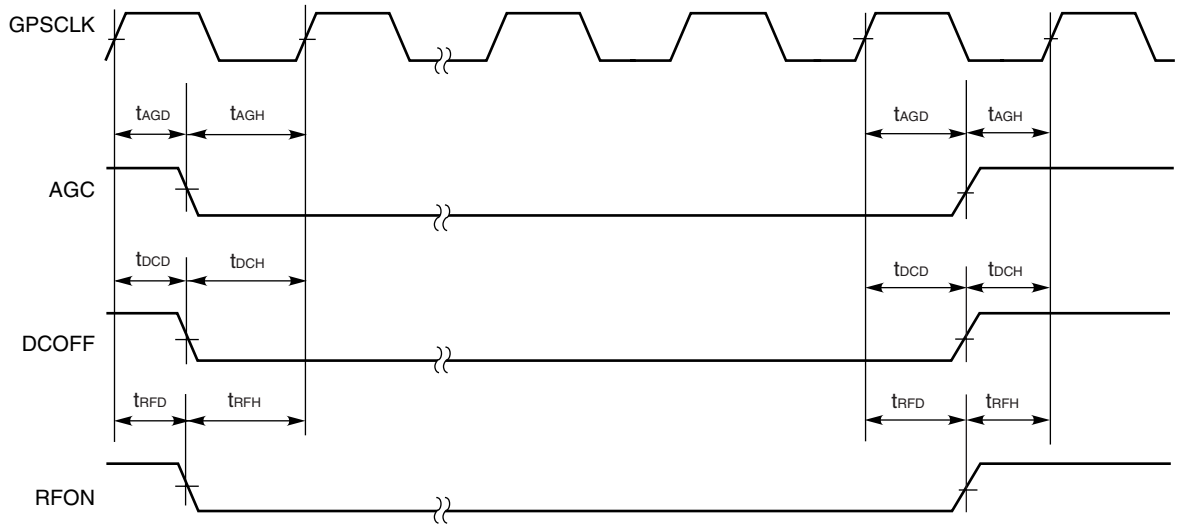


RF**コントロール・インタフェース**

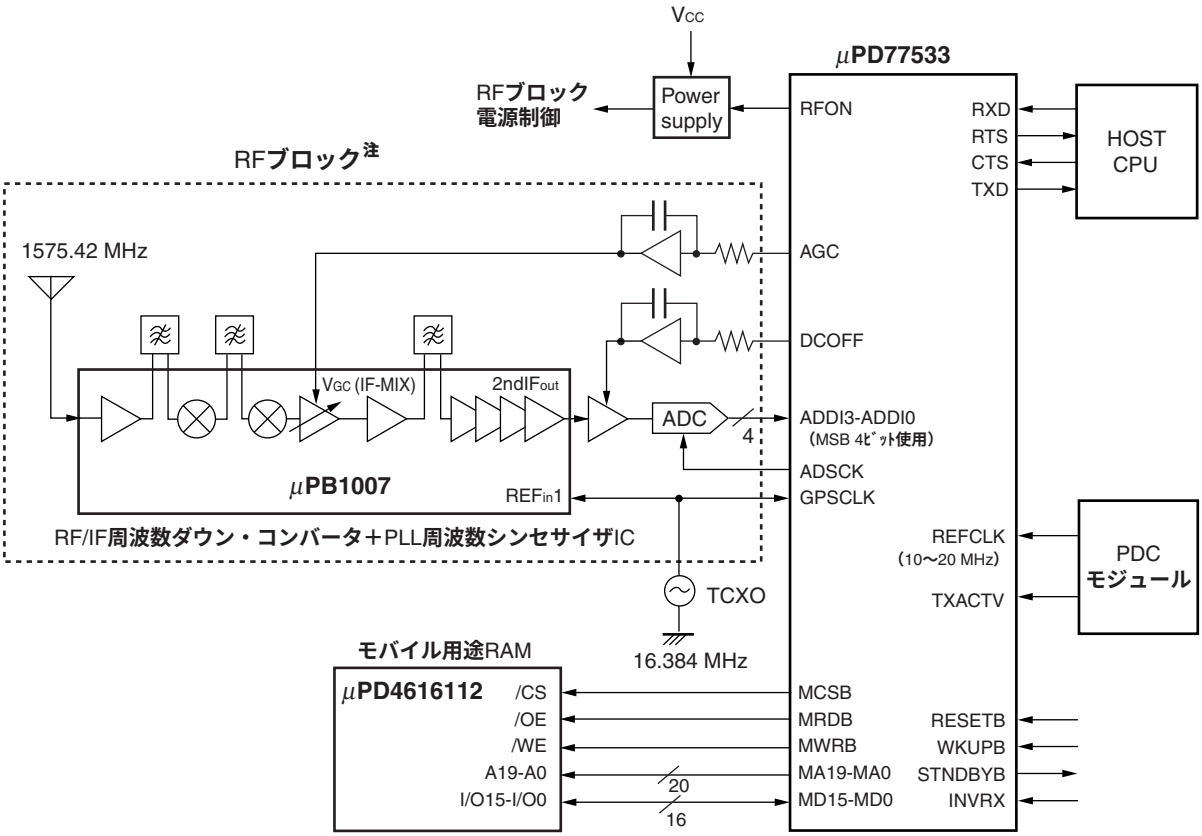
• AGC, DCOFF, RFON

項　目	略　号	条　件	MIN.	TYP.	MAX.	単位
AGC **遅延時間（対** GPSCLK↑**）**	t_{AGD}		−	−	20	ns
AGC **保持時間（対** GPSCLK↑**）**	t_{AGH}		0	−	−	ns
DCOFF **遅延時間(対** GPSCLK↑**）**	t_{DCD}		−	−	20	ns
DCOFF **保持時間(対** GPSCLK↑**）**	t_{DCH}		0	−	−	ns
RFON **遅延時間（対** GPSCLK↑**）**	t_{RFD}		−	−	50	ns
RFON **保持時間（対** GPSCLK↑**）**	t_{RFH}		0	−	−	ns

• AGC, DCOFF, RFON**タイミング**



20. 応用回路例



注意 μPD77533 **のシステム・ゲインは**, 60.1dB **です。**

注 μPD77533 **への** GPS **信号入力部（**RF **ブロック）に関する推奨動作環境を次に示します。**

項　目	数　値	
Local Oscillator Stability	REFCLK **への入力クロック精度**	＜± 0.1 ppm
	GPSCLK **への入力クロック精度**	**未使用**
A/D **コンバータの解像度**	4 **ビット以上**	
A/D **コンバータの変換速度**	8.192 MHz	
システム NF	＜2 dB	
帯域内スプリアス	**最終** IF **信号の中心周波数**± 1 MHz **帯域内において，**IF **のノイズ・レベルに対して，スプリアスのトータル・パワーが**– 20 dBc **未満となること。**	
平均アンテナ・ゲイン	– 6 dBi **以上（リニア）**	

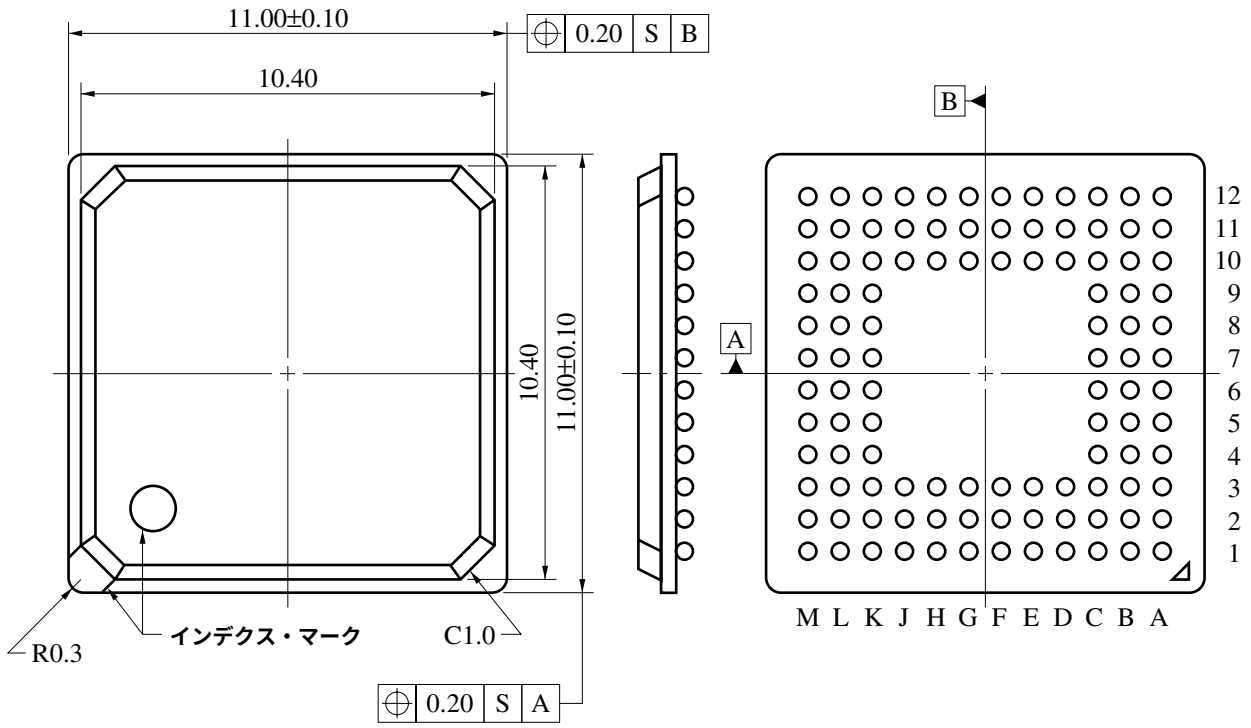
Local Oscillator Stability **の推奨精度が得られない場合は，**REFCLK **による** FCC **機能を使用する方法があります（**6. 1　FCC **機能を参照)。ただし，その場合は測位時間および感度の劣化が生じます。**
次に例を示します（この例では，測位時間は最大約 30 **秒，感度劣化はほとんどありません)。**

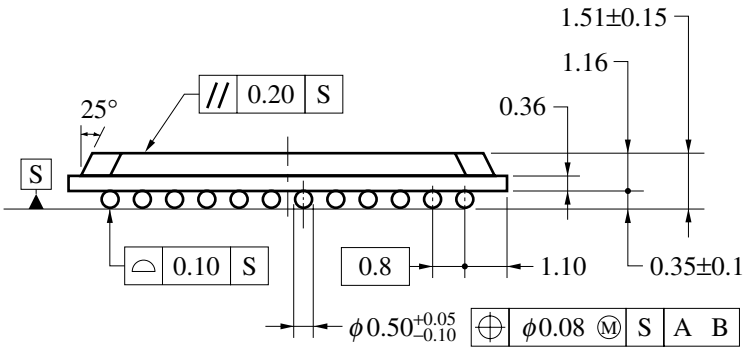
REFCLK **への入力クロックの精度：**　＜± 0.3 ppm

GPSCLK **への入力クロックの精度：**　＜± 10 ppm

21. 外 形 図

108ピン・プラスチック FBGA（11x11）外形図（単位：mm）





S108S1-80-YHC-1

22. 半田付け推奨条件

この製品の半田付け実装は，次の推奨条件で実施してください。

なお，推奨条件以外の半田付け方式および半田付け条件については，当社販売員にご相談ください。

半田付け推奨条件の技術的内容については下記を参照してください。

「半導体実装マニュアル」（http://www.nec.co.jp/pkg/ja/jissou/index.html）

μPD77533S1-YHC: 108 ピン・プラスチック・ファインピッチ BGA（11×11）

半田付け方式	半田付け条件	推奨条件記号
赤外線リフロ	パッケージ・ピーク温度：260 ℃，時間：60 秒以内（220 ℃以上） 回数：2 回以内，制限日数：3 日間[注]（以降は 125 ℃プリベーク 10～72 時間必要） ＜留意事項＞ 耐熱トレイ以外（マガジン，テーピング，非耐熱トレイ）は，包装状態でのベーキングができません。	IR60-103-2

注　ドライパック開封後の保管日数で，保管条件は 25 ℃，65 %RH 以下。

付録　サンプル・プログラム

本プログラムは，HOST CPU での処理内容を理解するためのサンプルです。実際の動作プログラム内容は，システムに依存します。

```
/*============================================================================
*
*                          ホスト CPU サンプルプログラム
*
============================================================================
*                    Copyright(C) 2001 NEC Corporation.
============================================================================
* このプログラムは、ホスト CPU が行うべき処理の例を記述した物です。
* プログラムを実際に動作させるには、システムにあわせて内容を変更する
* 必要があります。
============================================================================*/


/*============*/
/* 定数の定義 */
/*============*/

/* BOOL 値 */
#define TRUE                        1       /* TRUE */
#define FALSE                       0       /* FALSE */

/* ステートマシンで使用するステート値 */
#define STATE_INIT              0       /* 初期化 */
#define STATE_WAIT_RESET        1       /* Reset レスポンス待ち */
#define STATE_SEND_TIMEOUT      2       /* Character Timeout コマンド送信 */
#define STATE_WAIT_TIMEOUT      3       /* Character Timeout レスポンス待ち */
#define STATE_SEND_FCC_FREQ     4       /* FCC Frequency コマンド送信 */
#define STATE_WAIT_FCC_FREQ     5       /* FCC Frequency レスポンス待ち */
#define STATE_SEND_TXACVT       6       /* TXACVT コマンド送信 */
#define STATE_WAIT_TXACVT       7       /* TXACVT レスポンス待ち */
#define STATE_SEND_FCC_CTRL     8       /* FCC Control コマンド送信 */
#define STATE_WAIT_FCC_CTRL     9       /* FCC Control レスポンス待ち */
#define STATE_SEND_FLOW_CTRL    10      /* Flow Point Control コマンド送信 */
#define STATE_WAIT_FLOW_CTRL    11      /* Flow Point Control レスポンス待ち */
#define STATE_INIT_DONE         12      /* 初期化終了 */
#define STATE_DONE              13      /* 終了 */

/* シリアルバッファ制御値 */
#define SERIAL_GM               0       /* シリアルライン GM 側 */
#define SERIAL_LS               1       /* シリアルライン LS 側 */

#define SERIAL_IN               0       /* シリアル入力 */
#define SERIAL_OUT              1       /* シリアル出力 */

/* パケット */
#define PACKET_DATA_LEN         255     /* パケットのデータ部分の最大長 */
#define PACKET_INFO_LEN         3       /* パケット情報部分の長さ */
#define PACKET_MAX_LEN      (PACKET_DATA_LEN+PACKET_INFO_LEN)   /* パケットの最大長 */
#define PACKET_FLG              0x03    /* FLG(続きなし) */
#define PACKET_FLG_CONT         0x02    /* FLG(続あり) */

/* バッファサイズ */
#define MAXBLOCK                PACKET_MAX_LEN  /* シリアル受信ブロックサイズ */
```

```
/* LS コマンド */
#define CMD_CA              0x4341  /* "CA" */
#define CMD_BB              0x4242  /* "BB" */
#define CMD_IA              0x4941  /* "IA" */
#define CMD_JM              0x4A4D  /* "JM" */
#define CMD_PJ              0x504A  /* "PJ" */
#define CMD_AC              0x4143  /* "AC" */
#define CMD_DA              0x4441  /* "DA" */
#define CMD_EA              0x4541  /* "EA" */
#define CMD_FA              0x4641  /* "FA" */
#define CMD_FC              0x4643  /* "FC" */
#define CMD_BD              0x4244  /* "BD" */
#define CMD_CB              0x4342  /* "CB" */

#define CMD_ID_LS           0x80    /* LS コマンド */
#define CMD_ID_EXTENTION    0x7F    /* 拡張コマンド */

#define LS_CMD_MAX_LEN      330     /* LS コマンドの最大長 */


/*====================*/
/* ストラクチャの定義 */
/*====================*/

/* COM 情報 */
/* シリアルでの送受信データは、COM 情報ワークに格納する。 */
typedef struct {
    unsigned char   BinData[2][PACKET_MAX_LEN]; /* 送受信したデータ（バイナリ） */
    int             CurrentLen[2];      /* BinData に格納されたデータの長さ */
    int             EstimateLen[2];     /* 想定されるパケットの全長 */
    int             FixedFlag[2];       /* パケット完了フラグ */
        /* 上のパケット完了フラグは、SERIAL_IN ではパケット全体を受信した     */
        /* ときに TRUE となり、SERIAL_OUT では TRUE をセットした時点で送信が開始 */
        /* される。                                                          */
    int             AckNackWait;        /* ACK/NACK 待ちフラグ */
    int             LScmdLen;           /* LS からのコマンドの長さ */
} COMMINFO;


/* レスポンス情報 */
/* GM からのレスポンス、LS からのコマンドを受信したときは、レスポンス情報 */
/* ワークに格納する。                                                    */
typedef struct {
    /* GM 側 */
    unsigned int    GmResCnt;           /* GM 側レスポンス番号 */
    int             GmResFixed;         /* GM 側レスポンス受信完了フラグ */
    int             GmResLen;           /* GM 側レスポンスの長さ */
    unsigned char   GmResData[PACKET_MAX_LEN];  /* GM 側レスポンスデータ */
    /* LS 側 */
    unsigned int    LsResCnt;           /* LS 側レスポンス番号 */
    int             LsResFixed;         /* LS 側レスポンス受信完了フラグ */
    int             LsResLen;           /* LS 側レスポンスの長さ */
    unsigned char   LsResData[LS_CMD_MAX_LEN];  /* LS 側レスポンスデータ */
} RESINFO;


/*==================*/
/* 関数プロトタイプ */
/*==================*/

static int CompareData(const unsigned char *Data1, int Len1, const unsigned char *Data2, int Len2);
```

```
static int RecvDataFromGm(unsigned char *Buf);
static int RecvDataFromLS(unsigned char *Buf);
static void SendDataToGm(const unsigned char *Data, int Len);
static void SendDataToLs(const unsigned char *Data, int Len);
static void HostCPUThreadLoop(void);
static void GmSerialIn(void);
static void CopyGmRes(void);
static void CheckGmRes(void);
static void DiscardLastPacket(void);
static void ResendLastPacket(void);
static void SendLSAck(void);
static void SendLocalAck(void);
static void SendNack(void);
static void CopyGmToLs(void);
static void GmSerialOut(void);
static void LsSerialIn(void);
static void CopyLsRes(void);
static void CopyLsToGm(void);
static int GetLsCmdLen(unsigned char *Data);
static void LsSerialOut(void);


/*======================*/
/* スタティックなワーク */
/*======================*/

static COMMINFO GmComm;     /* GM側 COM情報 */
static COMMINFO LsComm;     /* LS側 COM情報 */
static RESINFO ResInfo;     /* レスポンス情報 */

/* GMコマンドのデータ列 */
/* ACK(LSコマンド) */
static const unsigned char ServerACK[] = { CMD_ID_LS, 0x00, PACKET_FLG };
/* ACK(ローカルコマンド) */
static const unsigned char LocalACK[] = { 0x3F, 0x01, 0x00, PACKET_FLG };
/* NACK */
static const unsigned char BothNACK[] = { 0x3F, 0x01 };
/* NACK(エラー) */
static const unsigned char ErrorNACK[] = { 0x3F, 0x01, 0x01, PACKET_FLG };
/* Reset レスポンス */
static const unsigned char ResetRsp[] = { 0x04, 0x00, PACKET_FLG };
/* Character Timeout コマンド／レスポンス */
static const unsigned char TimeoutCmdRsp[] = { 0x13, 0x03, 0xE8, 0x03, 0x00, PACKET_FLG };
/* FCC Frequency コマンド／レスポンス */
static const unsigned char FccFreqCmdRsp[] = { 0x02, 0x04, 0x00, 0x00, 0xE1, 0x00, PACKET_FLG };
/* TXACVT コマンド／レスポンス */
static const unsigned char TxacvtCmdRsp[] = { 0x07, 0x01, 0x01, PACKET_FLG };
/* FCC Control コマンド／レスポンス */
static const unsigned char FccCtrlCmdRsp[] = { 0x06, 0x01, 0x01, PACKET_FLG };
/* Flow Point Control コマンド／レスポンス */
static const unsigned char FlowPointCtrlCmdRsp[] = { 0x7F, 0x03, 0x00, 0x10, 0x00, PACKET_FLG };



/*-------------------------------------------------------------------------
*  Function
*      メインプログラム
*
*  Parameter
*      なし
*
```

```
*   Return Value
*       なし
------------------------------------------------------------------------------ */
int main(void)
{
    int State;
    unsigned char Buf[LS_CMD_MAX_LEN];
    int Len;

    /* ステートマシン */
    /* 初期化－Reset 待ち－各初期設定 の一連の処理を行うためのステートマシン */
    State = STATE_INIT;
    while(State != STATE_DONE)
    {
        switch(State)
        {
            /******************/
            /* 初期化ステート    */
            /******************/
            case STATE_INIT :
                /* 内部的なワークを初期化 */
                GmComm.CurrentLen[SERIAL_OUT]     = 0;
                GmComm.CurrentLen[SERIAL_IN]      = 0;
                GmComm.EstimateLen[SERIAL_OUT]    = 0;
                GmComm.EstimateLen[SERIAL_IN]     = 0;
                GmComm.FixedFlag[SERIAL_OUT]      = FALSE;
                GmComm.FixedFlag[SERIAL_IN]       = FALSE;
                GmComm.AckNackWait                = FALSE;
                GmComm.LScmdLen                   = 0;

                LsComm.CurrentLen[SERIAL_OUT]     = 0;
                LsComm.CurrentLen[SERIAL_IN]      = 0;
                LsComm.EstimateLen[SERIAL_OUT]    = 0;
                LsComm.EstimateLen[SERIAL_IN]     = 0;
                LsComm.FixedFlag[SERIAL_OUT]      = FALSE;
                LsComm.FixedFlag[SERIAL_IN]       = FALSE;
                LsComm.AckNackWait                = FALSE;
                LsComm.LScmdLen                   = 0;

                ResInfo.GmResCnt      = 0;
                ResInfo.GmResFixed    = FALSE;
                ResInfo.GmResLen      = 0;
                ResInfo.LsResCnt      = 0;
                ResInfo.LsResFixed    = FALSE;
                ResInfo.LsResLen      = 0;

                /* GM 側、LS 側のシリアル通信をオープンする */
                OpenSerial(SERIAL_GM);
                OpenSerial(SERIAL_LS);

                /* ホスト CPU 処理のスレッドを起動する */
                CreateThread(HostCPUThreadLoop);

                State = STATE_WAIT_RESET;
                break;

            /******************************/
            /* Reset レスポンス待ちステート     */
            /******************************/
            /* Reset レスポンスを待ち、来たら次の処理へ。 */
            case STATE_WAIT_RESET :
```

```
    Len = RecvDataFromGm(Buf);
    if(CompareData(Buf, Len, ResetRsp, sizeof(ResetRsp)) == 0)
        State = STATE_SEND_TIMEOUT;
    break;

/*****************************************/
/* Character Timeout コマンドの送信ステート     */
/*****************************************/
/* Character Timeout コマンドをを GM へ送る。（タイムアウト時間を 1 秒に） */
case STATE_SEND_TIMEOUT :
    SendDataToGm(TimeoutCmdRsp, sizeof(TimeoutCmdRsp));
    State = STATE_WAIT_TIMEOUT;
    break;

/* Character Timeout レスポンス待ち */
/* Character Timeout レスポンスを待ち、来たら次の処理へ。 */
case STATE_WAIT_TIMEOUT :
    Len = RecvDataFromGm(Buf);
    if(CompareData(Buf, Len, TimeoutCmdRsp, sizeof(TimeoutCmdRsp)) == 0)
        State = STATE_SEND_FCC_FREQ;
    break;

/*************************************/
/* FCC Frequency コマンドの送信ステート     */
/*************************************/
/* FCC Frequency コマンドをを GM へ送る。（周波数を 14.4MHz に） */
case STATE_SEND_FCC_FREQ :
    SendDataToGm(FccFreqCmdRsp, sizeof(FccFreqCmdRsp));
    State = STATE_WAIT_FCC_FREQ;
    break;

/* FCC Frequency レスポンス待ち */
/* FCC Frequency レスポンスを待ち、来たら次の処理へ。 */
case STATE_WAIT_FCC_FREQ :
    Len = RecvDataFromGm(Buf);
    if(CompareData(Buf, Len, FccFreqCmdRsp, sizeof(FccFreqCmdRsp)) == 0)
        State = STATE_SEND_TXACVT;
    break;

/*******************************/
/* TXACVT コマンドの送信ステート     */
/*******************************/
/* TXACVT コマンドをを GM へ送る。（TXACVT 制御を ON に） */
case STATE_SEND_TXACVT :
    SendDataToGm(TxacvtCmdRsp, sizeof(TxacvtCmdRsp));
    State = STATE_WAIT_TXACVT;
    break;

/* TXACVT レスポンス待ち */
/* TXACVT レスポンスを待ち、来たら次の処理へ。 */
case STATE_WAIT_TXACVT :
    Len = RecvDataFromGm(Buf);
    if(CompareData(Buf, Len, TxacvtCmdRsp, sizeof(TxacvtCmdRsp)) == 0)
        State = STATE_SEND_FCC_CTRL;
    break;

/***********************************/
/* FCC Control コマンドの送信ステート     */
/***********************************/
/* FCC Control コマンドをを GM へ送る。（FCC 制御を ON に） */
case STATE_SEND_FCC_CTRL :
```

```
            SendDataToGm(FccCtrlCmdRsp, sizeof(FccCtrlCmdRsp));
            State = STATE_WAIT_FCC_CTRL;
            break;

        /* FCC Control レスポンス待ち */
        /* FCC Control レスポンスを待ち、来たら次の処理へ。 */
        case STATE_WAIT_FCC_CTRL :
            Len = RecvDataFromGm(Buf);
            if(CompareData(Buf, Len, FccCtrlCmdRsp, sizeof(FccCtrlCmdRsp)) == 0)
                State = STATE_SEND_FLOW_CTRL;
            break;

        /*****************************************/
        /* Flow Point Control コマンドの送信ステート */
        /*****************************************/
        /* Flow Point Control コマンドをを GM へ送る。(Processing End 出力を ON に) */
        case STATE_SEND_FLOW_CTRL :
            SendDataToGm(FlowPointCtrlCmdRsp, sizeof(FlowPointCtrlCmdRsp));
            State = STATE_WAIT_FLOW_CTRL;
            break;

        /* Flow Point Control レスポンス待ち */
        /* Flow Point Control レスポンスを待ち、来たら次の処理へ。 */
        case STATE_WAIT_FLOW_CTRL :
            Len = RecvDataFromGm(Buf);
            if(CompareData(Buf, Len, FlowPointCtrlCmdRsp, sizeof(FlowPointCtrlCmdRsp)) == 0)
                State = STATE_INIT_DONE;
            break;

        /*****************************************/
        /* GM の初期化終了。測位などの処理を行う。      */
        /*****************************************/
        case STATE_INIT_DONE :
            /* 初期化後に必要な処理があれば、ここで行う。 */
            break;
    }
  }
  return(0);
}


/*---------------------------------------------------------------------------
*   Function
*       データを比較する
*
*   Parameter
*       const unsigned char *Data1 : データ1
*       int Len1 : データ1のバイト数
*       const unsigned char *Data2 : データ2
*       int Len2 : データ2のバイト数
*
*   Return Value
*       int : 結果 (0=一致)
---------------------------------------------------------------------------- */
static int CompareData(const unsigned char *Data1, int Len1, const unsigned char *Data2, int Len2)
{
  int Sts;

  Sts = Len1 - Len2;
  if(Sts == 0)
    Sts = memcmp(Data1, Data2, Len1);
```

```
    return(Sts);
}


/*-----------------------------------------------------------------------------
*   Function
*       GM からデータを受け取る
*
*   Parameter
*       unsigned char *Buf : データを受け取るバッファ
*
*   Return Value
*       int : データのバイト数 (0=データはなかった)
----------------------------------------------------------------------------- */
static int RecvDataFromGm(unsigned char *Buf)
{
    int Len;

    Len = 0;
    if(ResInfo.GmResFixed)      /* レスポンスがあれば */
    {
        /* バッファにコピーする */
        memcpy(Buf, ResInfo.GmResData, ResInfo.GmResLen);
        Len = ResInfo.GmResLen;
        ResInfo.GmResFixed = FALSE;
    }
    return(Len);
}


/*-----------------------------------------------------------------------------
*   Function
*       LS からデータを受け取る
*
*   Parameter
*       unsigned char *Buf : データを受け取るバッファ
*
*   Return Value
*       int : データのバイト数 (0=データはなかった)
----------------------------------------------------------------------------- */
static int RecvDataFromLS(unsigned char *Buf)
{
    int Len;

    Len = 0;
    if(ResInfo.LsResFixed)      /* レスポンスがあれば */
    {
         /* バッファにコピーする */
        memcpy(Buf, ResInfo.LsResData, ResInfo.LsResLen);
        Len = ResInfo.LsResLen;
        ResInfo.LsResFixed = FALSE;
    }
    return(Len);
}


/*-----------------------------------------------------------------------------
*   Function
*       GM にデータを送る
*
*   Parameter
```

```
*      unsigned char *Data : 送るデータ
*      int Len : データのバイト数
*
*   Return Value
*      なし
------------------------------------------------------------------------- */
static void SendDataToGm(const unsigned char *Data, int Len)
{
    /* 送信するデータを COM 情報にコピー */
    memcpy(GmComm.BinData[SERIAL_OUT], Data, Len);
    GmComm.CurrentLen[SERIAL_OUT] = Len;
    GmComm.FixedFlag[SERIAL_OUT] = TRUE;    /* このフラグで送信開始 */
    return;
}


/*-------------------------------------------------------------------------
*   Function
*      LS にデータを送る
*
*   Parameter
*      unsigned char *Data : 送るデータ
*      int Len : データのバイト数
*
*   Return Value
*      なし
------------------------------------------------------------------------- */
static void SendDataToLs(const unsigned char *Data, int Len)
{
    /* 送信するデータを COM 情報にコピー */
    memcpy(LsComm.BinData[SERIAL_OUT], Data, Len);
    LsComm.CurrentLen[SERIAL_OUT] = Len;
    LsComm.FixedFlag[SERIAL_OUT] = TRUE;    /* このフラグで送信開始 */
    return;
}


/*-------------------------------------------------------------------------
*   Function
*      ホスト CPU 処理部のスレッド
*
*   Parameter
*      なし
*
*   Return Value
*      なし
*
*   Note
*      この部分はメインプログラムの処理に影響されないよう、別のスレッドと
*      して処理している。
------------------------------------------------------------------------- */
static void HostCPUThreadLoop(void)
{
    for(;;)
    {
        /* GM からデータを受信し、GmComm に格納する */
        GmSerialIn();

        /* GM からの ACK/NACK を待っている状態でなければ */
        /* LS からデータを受信し、LsComm に格納する */
        if(GmComm.AckNackWait == FALSE)
```

```
            LsSerialIn();

        /* GmComm に格納された送信データを送信する */
        GmSerialOut();

        /* LsComm に格納された送信データを送信する */
        LsSerialOut();
    }
}


/*-------------------------------------------------------------------------
*   Function
*       DSP からの入力
*
*   Parameter
*       なし
*
*   Return Value
*       なし
------------------------------------------------------------------------- */
static void GmSerialIn(void)
{
    int Len;
    unsigned char Buf[MAXBLOCK];
    int i;

    /* GM からのレスポンスを読み込む */
    if((Len = RecvFromSerial(SERIAL_GM, (char*)Buf, MAXBLOCK)) > 0)
    {
        /* レスポンス内容を解析 */
        for(i = 0; i < Len; i++)
        {
            GmComm.BinData[SERIAL_IN][GmComm.CurrentLen[SERIAL_IN]] = Buf[i];
            GmComm.CurrentLen[SERIAL_IN] += 1;

            if(GmComm.CurrentLen[SERIAL_IN] == 1)   /* １バイト目か */
            {
                 /* CMD-ID を読み込んだ */
                GmComm.EstimateLen[SERIAL_IN] = -1;
            }
            else if(GmComm.CurrentLen[SERIAL_IN] == 2)  /* ２バイト目か */
            {
                /* LENGTH を読み込んだ */
                GmComm.EstimateLen[SERIAL_IN] = Buf[i]+PACKET_INFO_LEN;
            }
            else
            {
                if(GmComm.CurrentLen[SERIAL_IN] == GmComm.EstimateLen[SERIAL_IN])
                {
                    /* パケット全体を読み込んだ */
                    GmComm.FixedFlag[SERIAL_IN] = TRUE;

                    /* GM からのレスポンスをチェック */
                    CopyGmRes();
                    CheckGmRes();

                    GmComm.FixedFlag[SERIAL_IN] = FALSE;
                    GmComm.CurrentLen[SERIAL_IN] = 0;
                }
            }
```

```
        }
    }
    return;
}


/*---------------------------------------------------------------------------
*   Function
*       GM からのレスポンスをレスポンス情報にコピー
*
*   Parameter
*       なし
*
*   Return Value
*       なし
--------------------------------------------------------------------------- */
static void CopyGmRes(void)
{
    ResInfo.GmResCnt += 1;
    ResInfo.GmResLen = GmComm.CurrentLen[SERIAL_IN];
    memcpy(ResInfo.GmResData, GmComm.BinData[SERIAL_IN], ResInfo.GmResLen);
    ResInfo.GmResFixed = TRUE;
    return;
}


/*---------------------------------------------------------------------------
*   Function
*       GM からのレスポンスをチェック
*
*   Parameter
*       なし
*
*   Return Value
*       なし
--------------------------------------------------------------------------- */
static void CheckGmRes(void)
{
    unsigned char Flag;
    int Len;

    Len = GmComm.CurrentLen[SERIAL_IN];
    Flag = GmComm.BinData[SERIAL_IN][Len-1];
    if((Flag == PACKET_FLG) || (Flag == PACKET_FLG_CONT))
    {
        if(((Len == 3) && (memcmp(GmComm.BinData[SERIAL_IN], ServerACK, sizeof(ServerACK)) == 0)) ||
           ((Len == 4) && (memcmp(GmComm.BinData[SERIAL_IN], LocalACK, sizeof(LocalACK)) == 0)))
        {
            /* レスポンスは ACK */
            if(GmComm.AckNackWait)
                DiscardLastPacket();
#if 0
            else
                ACK/NACK 待ちでないときに、不必要な ACK を GM から受信した
#endif
        }
        else if((Len == 4) && (memcmp(GmComm.BinData[SERIAL_IN], BothNACK, sizeof(BothNACK)) == 0))
        {
            /* レスポンスは NACK */
            if(GmComm.AckNackWait)
                ResendLastPacket();
```

```
        }
        else if(GmComm.BinData[SERIAL_IN][0] == CMD_ID_LS)
        {
            /* レスポンスは LS コマンド */
            SendLSAck();
            CopyGmToLs();
        }
        else if((GmComm.BinData[SERIAL_IN][0] == CMD_ID_EXTENTION) &&
                (GmComm.BinData[SERIAL_IN][2] != 0x00))
        {
            /* レスポンスはフローポイントの設定 */
            SendLocalAck();
        }
        else
        {
            /* レスポンスはローカルコマンド */

            /* 何か処理する事があればここへ追加 */
        }
    }
    else
    {
        /* FLAG が正しくない */
        /* サーバコマンドのパケットの場合は NACK を送り、再送を要求する */
        if(GmComm.BinData[SERIAL_IN][0] == CMD_ID_LS)
            SendNack();
    }
    return;
}


/*---------------------------------------------------------------------------
*   Function
*       最後に送出したパケットを破棄する
*
*   Parameter
*       なし
*
*   Return Value
*       なし
---------------------------------------------------------------------------- */
static void DiscardLastPacket(void)
{
    GmComm.AckNackWait = FALSE;
    GmComm.CurrentLen[SERIAL_OUT] = 0;
    return;
}


/*---------------------------------------------------------------------------
*   Function
*       最後に送出したパケットを再送する
*
*   Parameter
*       なし
*
*   Return Value
*       なし
---------------------------------------------------------------------------- */
static void ResendLastPacket(void)
{
```

```
    GmComm.FixedFlag[SERIAL_OUT] = TRUE;
    GmComm.AckNackWait = FALSE;
    GmSerialOut();
}


/*---------------------------------------------------------------------------
*   Function
*       サーバコマンドの ACK(80 00 03)を送る
*
*   Parameter
*       なし
*
*   Return Value
*       なし
--------------------------------------------------------------------------- */
static void SendLSAck(void)
{
    memcpy(GmComm.BinData[SERIAL_OUT], ServerACK, sizeof(ServerACK));
    GmComm.CurrentLen[SERIAL_OUT] = sizeof(ServerACK);
    GmComm.FixedFlag[SERIAL_OUT] = TRUE;
    GmSerialOut();
}


/*---------------------------------------------------------------------------
*   Function
*       ローカルコマンドの ACK(3f 01 00 03)を送る
*
*   Parameter
*       なし
*
*   Return Value
*       なし
--------------------------------------------------------------------------- */
static void SendLocalAck(void)
{
    memcpy(GmComm.BinData[SERIAL_OUT], LocalACK, sizeof(LocalACK));
    GmComm.CurrentLen[SERIAL_OUT] = sizeof(LocalACK);
    GmComm.FixedFlag[SERIAL_OUT] = TRUE;
    GmSerialOut();
}


/*---------------------------------------------------------------------------
*   Function
*       NACK を送る
*
*   Parameter
*       なし
*
*   Return Value
*       なし
--------------------------------------------------------------------------- */
static void SendNack(void)
{
    memcpy(GmComm.BinData[SERIAL_OUT], ErrorNACK, sizeof(ErrorNACK));
    GmComm.CurrentLen[SERIAL_OUT] = sizeof(ErrorNACK);
    GmComm.FixedFlag[SERIAL_OUT] = TRUE;
    GmSerialOut();
}
```

```
/*---------------------------------------------------------------------------
*   Function
*       GM からのレスポンスを LS に送る
*
*   Parameter
*       なし
*
*   Return Value
*       なし
--------------------------------------------------------------------------- */
static void CopyGmToLs(void)
{
    int Len;

    /* パケットの中身だけを LS に送る */
    Len = GmComm.CurrentLen[SERIAL_IN] - PACKET_INFO_LEN;
    memcpy(LsComm.BinData[SERIAL_OUT], GmComm.BinData[SERIAL_IN] + 2, Len);
    LsComm.CurrentLen[SERIAL_OUT] = Len;
    LsComm.FixedFlag[SERIAL_OUT] = TRUE;
    LsSerialOut();
    return;
}


/*---------------------------------------------------------------------------
*   Function
*       GM への出力
*
*   Parameter
*       なし
*
*   Return Value
*       なし
--------------------------------------------------------------------------- */
static void GmSerialOut(void)
{
    if(GmComm.FixedFlag[SERIAL_OUT])
    {
        /* GM に出力する */
        SendToSerial(SERIAL_GM, (char*)GmComm.BinData[SERIAL_OUT], GmComm.CurrentLen[SERIAL_OUT]);
        GmComm.FixedFlag[SERIAL_OUT] = FALSE;

        /* LS コマンドなら ACK/NACK を待つようにセット */
        if((GmComm.BinData[SERIAL_OUT][0] == CMD_ID_LS) &&
           ((GmComm.CurrentLen[SERIAL_OUT] != 3) ||
            (memcmp(GmComm.BinData[SERIAL_OUT], ServerACK, sizeof(ServerACK)) != 0)))
        {
            GmComm.AckNackWait = TRUE;
        }
        else
            GmComm.CurrentLen[SERIAL_OUT] = 0;
    }
    return;
}


/*---------------------------------------------------------------------------
*   Function
*       LS からの入力
```

```
*
*   Parameter
*       なし
*
*   Return Value
*       なし
-------------------------------------------------------------------------- */
static void LsSerialIn(void)
{
    int Len;

    if(LsComm.EstimateLen[SERIAL_IN] > 0)
    {
        /* 前に読み込んだデータが使われずに残っている */
        /* LS からの複数のコマンド（複数のパケットに分ける必要がある）が */
        /* 連続して送られてきたとき、最初のコマンドはすぐに使うが、 */
        /* ２つ目以降のコマンドは残ったままここに来る */
        CopyLsToGm();

        if(LsComm.CurrentLen[SERIAL_IN] == LsComm.EstimateLen[SERIAL_IN])
            LsComm.EstimateLen[SERIAL_IN] = 0;
    }
    else if((Len = RecvFromSerial(SERIAL_LS, (char*)LsComm.BinData[SERIAL_IN], PACKET_DATA_LEN)) > 0)
    {
        LsComm.CurrentLen[SERIAL_IN] = Len;
        LsComm.EstimateLen[SERIAL_IN] = 0;

        CopyLsToGm();

        if(LsComm.CurrentLen[SERIAL_IN] == LsComm.EstimateLen[SERIAL_IN])
            LsComm.EstimateLen[SERIAL_IN] = 0;
    }
    return;
}


/*--------------------------------------------------------------------------
*   Function
*       LS からのコマンドをレスポンスバッファにコピー
*
*   Parameter
*       なし
*
*   Return Value
*       なし
-------------------------------------------------------------------------- */
static void CopyLsRes(void)
{
    ResInfo.LsResCnt += 1;
    ResInfo.LsResLen = LsComm.CurrentLen[SERIAL_IN];
    memcpy(ResInfo.LsResData, LsComm.BinData[SERIAL_IN], ResInfo.LsResLen);
    ResInfo.LsResFixed = TRUE;
    return;
}


/*--------------------------------------------------------------------------
*   Function
*       LS からのコマンドを GM に出力
*
*   Parameter
```

```
*      なし
*
*   Return Value
*      なし
------------------------------------------------------------------------------ */
static void CopyLsToGm(void)
{
    int i;
    unsigned char Data;
    static unsigned char LScmdData[5];

    for(i = LsComm.EstimateLen[SERIAL_IN]; i < LsComm.CurrentLen[SERIAL_IN]; i++)
    {
        Data = LsComm.BinData[SERIAL_IN][i];

        if(GmComm.CurrentLen[SERIAL_OUT] == 0)
        {
            /* 最初のデータ */
            /* パケット化する */
            GmComm.BinData[SERIAL_OUT][0] = CMD_ID_LS;  /* CMD-ID=0x80, LENGTH は後でセット */
            GmComm.BinData[SERIAL_OUT][2] = Data;
            GmComm.CurrentLen[SERIAL_OUT] = PACKET_INFO_LEN;
        }
        else
        {
            /* データをセット */
            GmComm.BinData[SERIAL_OUT][GmComm.CurrentLen[SERIAL_OUT]] = Data;
            GmComm.CurrentLen[SERIAL_OUT] += 1;
        }

        if(LsComm.LScmdLen < 5)
            LScmdData[LsComm.LScmdLen] = Data;
        LsComm.LScmdLen += 1;
        if(LsComm.LScmdLen == 5)
        {
            LsComm.EstimateLen[SERIAL_OUT] = GetLsCmdLen(LScmdData);
            if(LsComm.EstimateLen[SERIAL_OUT] == 0)
                GmComm.CurrentLen[SERIAL_OUT] = 0;
        }

        if(LsComm.LScmdLen == LsComm.EstimateLen[SERIAL_OUT])
        {
            /* 最後のパケット */
            /* FLAG=0x03 で送る */
            GmComm.BinData[SERIAL_OUT][GmComm.CurrentLen[SERIAL_OUT]] = PACKET_FLG;
            GmComm.CurrentLen[SERIAL_OUT] += 1;
            GmComm.BinData[SERIAL_OUT][1] = GmComm.CurrentLen[SERIAL_OUT] - PACKET_INFO_LEN;
            GmComm.FixedFlag[SERIAL_OUT] = TRUE;

            CopyLsRes();

            LsComm.LScmdLen = 0;
            LsComm.EstimateLen[SERIAL_OUT] = 0;
            i++;
            break;
        }
    }

    if(GmComm.FixedFlag[SERIAL_OUT] == FALSE)
    {
        /* 途中のパケット */
```

```
        /* FLAG=0x02 で送る */
        GmComm.BinData[SERIAL_OUT][GmComm.CurrentLen[SERIAL_OUT]] = PACKET_FLG_CONT;
        GmComm.CurrentLen[SERIAL_OUT] += 1;
        GmComm.BinData[SERIAL_OUT][1] = GmComm.CurrentLen[SERIAL_OUT] - PACKET_INFO_LEN;
        GmComm.FixedFlag[SERIAL_OUT] = TRUE;
    }

    GmSerialOut();

    LsComm.EstimateLen[SERIAL_IN] = i;
    return;
}


/*---------------------------------------------------------------------------
*   Function
*       LS コマンドの長さの期待値を求める
*
*   Parameter
*       unsigned char *Data : LS コマンドのデータ列
*
*   Return Value
*       int : バイト数
--------------------------------------------------------------------------- */
static int GetLsCmdLen(unsigned char *Data)
{
    short Cmd;
    int Len;

    Len = 0;
    Cmd = (*(Data + 1) << 8) | *(Data + 2);
    switch(Cmd)
    {
        case CMD_CA :
        case CMD_EA :
        case CMD_FC :
            Len = 0 + 7;    /* DATA フィールドは 0 バイト固定 */
            break;

        case CMD_BB :
        case CMD_IA :
        case CMD_PJ :
        case CMD_BD :
        case CMD_CB :
            Len = 1 + 7;    /* DATA フィールドは 1 バイト固定 */
            break;

        case CMD_FA :
            Len = 4 + 7;    /* DATA フィールドは 4 バイト固定 */
            break;

        case CMD_DA :
            Len = 5 + 7;    /* DATA フィールドは 5 バイト固定 */
            break;

        case CMD_JM :
        case CMD_AC :
            Len = (*(Data + 4) << 8) | *(Data + 3);
            break;
    }
    return(Len);
```

```
}


/*---------------------------------------------------------------------------
*   Function
*       LS への出力
*
*   Parameter
*       なし
*
*   Return Value
*       なし
--------------------------------------------------------------------------- */
static void LsSerialOut(void)
{
    if(LsComm.FixedFlag[SERIAL_OUT])
    {
        SendToSerial(SERIAL_LS, (char*)LsComm.BinData[SERIAL_OUT], LsComm.CurrentLen[SERIAL_OUT]);
        LsComm.FixedFlag[SERIAL_OUT] = FALSE;
    }
    return;
}
```

CMOSデバイスの一般的注意事項

①静電気対策（MOS全般）

注意　MOSデバイス取り扱いの際は静電気防止を心がけてください。

MOSデバイスは強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には，NECが出荷梱包に使用している導電性のトレーやマガジン・ケース，または導電性の緩衝材，金属ケースなどを利用し，組み立て工程にはアースを施してください。プラスチック板上に放置したり，端子を触ったりしないでください。

また，MOSデバイスを実装したボードについても同様の扱いをしてください。

②未使用入力の処理（CMOS特有）

注意　CMOSデバイスの入力レベルは固定してください。

バイポーラやNMOSのデバイスと異なり，CMOSデバイスの入力に何も接続しない状態で動作させると，ノイズなどに起因する中間レベル入力が生じ，内部で貫通電流が流れて誤動作を引き起こす恐れがあります。プルアップかプルダウンによって入力レベルを固定してください。また，未使用端子が出力となる可能性（タイミングは規定しません）を考慮すると，個別に抵抗を介してV_{DD}またはGNDに接続することが有効です。

資料中に「未使用端子の処理」について記載のある製品については，その内容を守ってください。

③初期化以前の状態（MOS全般）

注意　電源投入時，MOSデバイスの初期状態は不定です。

分子レベルのイオン注入量等で特性が決定するため，初期状態は製造工程の管理外です。電源投入時の端子の出力状態や入出力設定，レジスタ内容などは保証しておりません。ただし，リセット動作やモード設定で定義している項目については，これらの動作ののちに保証の対象となります。

リセット機能を持つデバイスの電源投入後は，まずリセット動作を実行してください。

- 本資料の内容は予告なく変更することがありますので，最新のものであることをご確認の上ご使用ください。
- 文書による当社の承諾なしに本資料の転載複製を禁じます。
- 本資料に記載された製品の使用もしくは本資料に記載の情報の使用に際して，当社は当社もしくは第三者の知的財産権その他の権利に対する保証または実施権の許諾を行うものではありません。上記使用に起因する第三者所有の権利にかかわる問題が発生した場合，当社はその責を負うものではありませんのでご了承ください。
- 本資料に記載された回路，ソフトウエア，及びこれらに付随する情報は，半導体製品の動作例，応用例を説明するためのものです。従って，これら回路・ソフトウエア・情報をお客様の機器に使用される場合には，お客様の責任において機器設計をしてください。これらの使用に起因するお客様もしくは第三者の損害に対して，当社は一切その責を負いません。
- 当社は品質，信頼性の向上に努めていますが，半導体製品はある確率で故障が発生します。当社半導体製品の故障により結果として，人身事故，火災事故，社会的な損害等を生じさせない冗長設計，延焼対策設計，誤動作防止設計等安全設計に十分ご注意願います。
- 当社は，当社製品の品質水準を「標準水準」，「特別水準」およびお客様に品質保証プログラムを指定して頂く「特定水準」に分類しております。また，各品質水準は以下に示す用途に製品が使われることを意図しておりますので，当社製品の品質水準をご確認の上ご使用願います。

  標準水準：コンピュータ，OA機器，通信機器，計測機器，AV機器，家電，工作機械，パーソナル機器，産業用ロボット

  特別水準：輸送機器（自動車，列車，船舶等），交通用信号機器，防災／防犯装置，各種安全装置，生命維持を直接の目的としない医療機器

  特定水準：航空機器，航空宇宙機器，海底中継機器，原子力制御システム，生命維持のための医療機器，生命維持のための装置またはシステム等

当社製品のデータ・シート／データ・ブック等の資料で，特に品質水準の表示がない場合は標準水準製品であることを表します。当社製品を上記の「標準水準」の用途以外でご使用をお考えのお客様は，必ず事前に当社販売窓口までご相談頂きますようお願い致します。

M7 98.8

お問い合わせ先

【技術的なお問い合わせ先】

ＮＥＣ半導体テクニカルホットライン
（電話：午前 9:00～12:00，午後 1:00～5:00）

電　話　：044−435−9494
FAX　：044−435−9608
E-mail　：info@lsi.nec.co.jp

【営業関係お問い合わせ先】

第一販売事業部
東　京　(03)3798−6106, 6107, 6108
大　阪　(06)6945−3178, 3200, 3208, 3212
広　島　(082)242−5504
仙　台　(022)267−8740

第二販売事業部
東　京　(03)3798−6110, 6111, 6112
立　川　(042)526−5981, 6167
松　本　(0263)35−1662
静　岡　(054)254−4794
金　沢　(076)232−7303
松　山　(089)945−4149

第三販売事業部
東　京　(03)3798−6151, 6155, 6586, 1622, 1623, 6156
水　戸　(029)226−1702
前　橋　(027)243−6060
鳥　取　(0857)27−5313
名古屋　(052)222−2170, 2190
福　岡　(092)261−2806

【資料の請求先】

上記営業関係お問い合わせ先またはNEC特約店へお申しつけください。

【NECエレクトロンデバイス　ホームページ】

NECエレクトロンデバイスの情報がインターネットでご覧になれます。　URL（アドレス）　http://www.ic.nec.co.jp/

C02.1