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

アプリケーション・ノート

V850ES/SA2, V850ES/SA3

32ビット・シングルチップ・マイクロコンピュータ

V850ES/SA2：
μPD703200
μPD703200Y
μPD703201
μPD703201Y
μPD70F3201
μPD70F3201Y

V850ES/SA3：
μPD703204
μPD703204Y
μPD70F3204
μPD70F3204Y

資料番号　U16764JJ1V0AN00（第1版）
発行年月　July 2003 N　CP(K)

〔メ　モ〕

目次要約

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，当社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。

注意： μPD703200Y, 703201Y, 703204Y, 70F3201Y, 70F3204YはI^2Cバス・インタフェース回路を内蔵しています。

I^2Cバス・インタフェースを使用される場合には，カスタム・コードをご発注いただく時に，事前にその旨ご申告下さい。申告に基づき，以下の特典が受けられます。

当社のI^2Cバス対応部品をご購入いただくことにより，これらの部品をI^2Cシステムに使用する実施権がフィリップス社I^2C特許に基づき許諾されることになります。ただし，これらのI^2Cシステムはフィリップス社によって設定されたI^2C標準規格に合致しているものとします。

Purchase of NEC Electronics I^2C components conveys a license under the Philips I^2C Patent Rights to use these components in an I^2C system, provided that the system conforms to the I^2C Standard Specification as defined by Philips.

本製品のうち，外国為替及び外国貿易法の規定により規制貨物等（または役務）に該当するものについては，日本国外に輸出する際に，同法に基づき日本国政府の輸出許可が必要です。

非該当品　：μPD70F3201, 70F3201Y, 70F3104, 70F3204Y

ユーザ判定品：μPD703200, 703200Y, 703201, 703201Y, 703204, 703204Y

- **本資料に記載されている内容は2003年6月現在のもので，今後，予告なく変更することがあります。量産設計の際には最新の個別データ・シート等をご参照ください。**
- **文書による当社の事前の承諾なしに本資料の転載複製を禁じます。当社は，本資料の誤りに関し，一切その責を負いません。**
- **当社は，本資料に記載された当社製品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，一切その責を負いません。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。**
- **本資料に記載された回路，ソフトウエアおよびこれらに関する情報は，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，ソフトウエアおよびこれらに関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，一切その責を負いません。**
- **当社は，当社製品の品質，信頼性の向上に努めておりますが，当社製品の不具合が完全に発生しないことを保証するものではありません。当社製品の不具合により生じた生命，身体および財産に対する損害の危険を最小限度にするために，冗長設計，延焼対策設計，誤動作防止設計等安全設計を行ってください。**
- **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いただく「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ください。**

  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，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

  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。意図されていない用途で当社製品の使用をお客様が希望する場合には，事前に当社販売窓口までお問い合わせください。

（注）

（1）本事項において使用されている「当社」とは，NECエレクトロニクス株式会社およびNEC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う。

（2）本事項において使用されている「当社製品」とは，（1）において定義された当社の開発，製造製品をいう。

M8E　02.11

はじめに

対 象 者　**このアプリケーション・ノートは，**V850ES/SA2, V850ES/SA3**の機能を理解し，それらを使用した応用システムを設計するユーザを対象とします。**
対象製品は次のようになります。

- V850ES/SA2：μPD703200, 703200Y, 703201, 703201Y, 70F3204, 70F3204Y
- V850ES/SA3：μPD703204, 703204Y, 70F3204, 70F3204Y

目　　的　**このアプリケーション・ノートでは**V850ES/SA2, V850ES/SA3**の製品概要，周辺機能接続例，またシステムの例としてバッテリ駆動型データ計測装置の応用回路例を取り上げ，その構成をユーザに理解していただくことを目的としています。**

構　　成　**このアプリケーション・ノートは大きく分けて次の内容で構成しています。**

- **●**V850ES/SA2, V850ES/SA3**の概要**
- **●バス・インタフェース接続回路例**
- **●内部周辺バス・インタフェース接続回路例（付加回路を使用しての接続例）**
- **●アプリケーション例**

読 み 方　**このマニュアルの読者には，電気，論理回路，およびマイクロコンピュータに関する一般知識を必要とします。**

V850ES/SA2, V850ES/SA3**のハードウエア機能および，電気的特性を知りたいとき**
→別冊のV850ES/SA2, V850ES/SA3　**ユーザーズ・マニュアル　ハードウエア編を参照してください。**

V850ES/SA2, V850ES/SA3**の命令機能を知りたいとき**
→別冊のV850ES　**ユーザーズ・マニュアル　アーキテクチャ編を参照してください。**

凡　　例

データ表記の重み	**：左が上位桁，右が下位桁**
アクティブ・ロウの表記	**：**$\overline{\text{xxx}}$**（端子，信号名称に上線）または**
	／xxx**（信号名称の前に“／”記号）**
メモリ・マップのアドレス	**：上部－上位，下部－下位**
注	**：本文中に付けた注の説明**
注意	**：気を付けて読んでいただきたい内容**
備考	**：本文の補足説明**
数の表記	**：2進数** … xxxx**または**xxxxB
	10**進数** … xxxx
	16**進数** … xxxxH
2のべき数を示す接頭語（アドレス空間，メモリ容量）	**：K（キロ）** …2^{10} = 1024
	M（メガ） …2^{20} = 1024^2
	G（ギガ） …2^{30} = 1024^3

関連資料　関連資料は暫定版の場合がありますが，この資料では「暫定」の表示をしておりません。
あらかじめご了承ください。

V850ES/SA2, V850ES/SA3**に関する資料**

資　料　名	資料番号
V850ES　**ユーザーズ・マニュアル　アーキテクチャ編**	U15943J
V850ES/SA2, V850ES/SA3　**ユーザーズ・マニュアル　ハードウエア編**	U15905J
V850ES/SA2, V850ES/SA3　**アプリケーション・ノート　ハードウエア編**	**このマニュアル**

開発ツールに関する資料（ユーザーズ・マニュアル）

資　料　名		資料番号
IE-V850ES-G1（**インサーキット・エミュレータ**）		U16313J
IE-703204-G1-EM1（**インサーキット・エミュレータ・オプション・ボード**）		U16622J
CA850　Ver.2.50　**Cコンパイラ・パッケージ**	**操作編**	U16053J
	C言語編	U16054J
	アセンブリ言語編	U16042J
PM plus Ver.5.10		U16569J
ID850　Ver.2.50　**統合ディバッガ**	**操作編**	U16217J
SM850　Ver.2.50　**システム・シミュレータ**	**操作編**	U15182J
SM850　Ver.2.00**以上**　**システム・シミュレータ**	**外部部品ユーザ・オープン・インタフェース仕様編**	U14873J
RX850　Ver.3.13**以上**　**リアルタイム**OS	**基礎編**	U13430J
	インストレーション編	U13410J
	テクニカル編	U13431J
RX850 Pro　Ver.3.15　**リアルタイム**OS	**基礎編**	U13773J
	インストレーション編	U13774J
	テクニカル編	U13772J
RD850　Ver.3.01　**タスク・ディバッガ**		U13737J
RD850 Pro　Ver.3.01　**タスク・ディバッガ**		U13916J
AZ850　Ver.3.0　**システム・パフォーマンス・アナライザ**		U14410J
PG-FP4（**フラッシュ・メモリ・プログラマ**）		U15260J

目　次

図の目次（1/2）

図の目次（2/2）

図番号　　タイトル，ページ

表の目次

表番号　　タイトル，ページ

第1章　V850ES/SA2, V850ES/SA3の概要

V850ES/SA2, V850ES/SA3は，NECのリアルタイム制御向けシングルチップ・マイクロコンピュータV850シリーズのロウ・パワー・シリーズの1製品です。

1. 1　イントロダクション

V850ES/SA2, V850ES/SA3は，V850ES CPUコアを使用し，ROM/RAM，タイマ／カウンタ，シリアル・インタフェース，A/Dコンバータ，D/Aコンバータ，DMAコントローラなどの周辺機能を内蔵した32ビット・シングルチップ・マイクロコンピュータです。

V850/SA1に対しては，CPUをV850ESに変更し，D/Aコンバータ，ROMコレクションなどの周辺機能を追加しています。

表1－1にV850ES/SA2, V850ES/SA3の製品一覧について示します。

表1－1　V850ES/SA2, V850ES/SA3の製品一覧

製品名		I^2C	ROM		RAM	パッケージ
愛　称	品　名	内蔵	種　類	サイズ	サイズ	
V850ES/SA2	μPD703200	なし	マスクROM	128 Kバイト	8 Kバイト	100ピンTQFP（14×14）
	μPD703200Y	あり				
	μPD703201	なし		256 Kバイト	16 Kバイト	
	μPD703201Y	あり				
	μPD70F3201	なし	フラッシュ・メモリ			
	μPD70F3201Y	あり				
V850ES/SA3	μPD703204	なし	マスクROM	256 Kバイト	16 Kバイト	121ピンFBGA（12×12）
	μPD703204Y	あり				
	μPD70F3204	なし	フラッシュ・メモリ			
	μPD70F3204Y	あり				

1.2 特　　徴

○命令数　83

○最小命令実行時間　50 MHz：メイン・クロック = 20 MHz動作時
（μPD703200, 703201, 703204, 70F3201, 70F3204）
59 MHz：メイン・クロック = 17 MHz動作時
（μPD703200Y, 703201Y, 703204Y, 70F3201Y, 70F3204Y）
サブクロック = 32.768 kHz動作時

○汎用レジスタ　32ビット×32本

○命令セット　符号付き乗算（16×16→32）：1-2クロック
符号付き乗算（32×32→64）：1-5クロック
飽和演算（オーバフロー／アンダフロー検出機能付き）
32ビット・シフト命令：1クロック
ビット操作命令
ロング／ショート形式を持つロード／ストア命令

○メモリ空間　64 Mバイト・リニア・アドレス空間（プログラム／データ共用）
外部拡張：V850ES/SA2は4 Mバイト，V850ES/SA3は16 Mバイトまで可能
（このうち1 Mバイトは内部ROM/RAM空間として使用）
メモリ・ブロック分割機能：2 M, 2 M, 4 M, 8 Mバイト（計4ブロック）
プログラマブル・ウエイト機能
アイドル・ステート挿入機能

○外部バス・インタフェース
マルチプレクス・バス／セパレート・バス出力選択可能
8/16ビット・データ・バス・サイジング機能
4空間のチップ・セレクト機能
ウエイト機能
・プログラマブル・ウエイト機能
・外部ウエイト機能
アイドル・ステート機能
バス・ホールド機能

○内蔵メモリ　RAM　：8 Kバイト（μPD703200, 703200Y）
：16 Kバイト（μPD703201, 703201Y, 70F3201, 70F3201Y, 703204, 703204Y, 70F3204, 70F3204Y）
マスクROM　：128 Kバイト（μPD703200, 703200Y）
256 Kバイト（μPD703201, 703201Y, 703204, 703204Y）
フラッシュ・メモリ　：256 Kバイト（μPD70F3201, 70F3201Y, 70F3204, 70F3204Y）

○割り込み／例外

外部割り込み　：8要因

内部割り込み　：29要因（μPD703200, 703201, 70F3201）

30要因（μPD703200Y, 703201Y, 70F3201Y）

31要因（μPD703204, 70F3204）

32要因（μPD703204Y, 70F3204Y）

ソフトウエア例外　：32要因

例外トラップ　：1要因

○I/Oライン　合計：82（入力ポート：14　入出力ポート：68）（V850ES/SA2）

102（入力ポート：18　入出力ポート：84）（V850ES/SA3）

○タイマ／カウンタ

16ビット・タイマ／イベント・カウンタ：2 ch（PWM出力）

8ビット・タイマ／イベント・カウンタ　：4 ch（カスケード接続可能）

○リアルタイム・カウンタ（時計用）　サブクロック／メイン・クロック動作：1 ch

週，日，時，分，秒の専用ハードウエア・カウンタ搭載

最長4095週までカウント

○ウォッチドッグ・タイマ：1 ch

○シリアル・インタフェース

アシンクロナス・シリアル・インタフェース（UART）

クロック同期式シリアル・インタフェース（CSI）

I^2Cバス・インタフェース（I^2C）

（μPD703200Y, 703201Y, 703204Y, 70F3201Y, 70F3204Yのみ）

CSI /UART　：1 ch

UART　：1 ch

CSI/ I^2C　：1 ch

CSI　：2 ch（V850ES/SA2），3 ch（V850ES/SA3）

○A/Dコンバータ　10ビット分解能：　12 ch（V850ES/SA2）

16 ch（V850ES/SA3）

○D/Aコンバータ　8ビット分解能：　2 ch

○DMAコントローラ：4 ch

○ROMコレクション：4箇所修正可能

○クロック・ジェネレータ　メイン・クロック／サブクロック動作

CPUクロック7段階（f_{XX}, f_{XX}/2, f_{XX}/4, f_{XX}/8, f_{XX}/16, f_{XX}/32, f_{XT}）

○パワー・セーブ機能　HALT/IDLE/STOPモード

○パッケージ　100ピン・プラスチックTQFP（ファインピッチ）（14×14）（V850ES/SA2）

121ピン・プラスチックFBGA（12×12）（V850ES/SA3）

1.3　応用分野

低消費電力を必要とする携帯機器全般

例　DVC，ポータブル・オーディオ

1.4　オーダ情報

1.4.1　V850ES/SA2

品　名	パッケージ	内蔵ROM
μPD703200GC-xxx-YEU	100ピン・プラスチックTQFP（ファインピッチ）（14×14）	マスクROM（128Kバイト）
μPD703200YGC-xxx-YEU	〃	〃
μPD703201GC-xxx-YEU	〃	マスクROM（256Kバイト）
μPD703201YGC-xxx-YEU	〃	〃
μPD70F3201GC-YEU	〃	フラッシュ・メモリ（256Kバイト）
μPD70F3201YGC-YEU	〃	〃

備考1.　xxxはROMコード番号です。

2.　ROMレス品はありません。

1.4.2　V850ES/SA3

品　名	パッケージ	内蔵ROM
μPD703204F1-xxx-EA6	121ピン・プラスチックFBGA（12×12）	マスクROM（256Kバイト）
μPD703204YF1-xxx-EA6	〃	〃
μPD70F3204F1-EA6	〃	フラッシュ・メモリ（256Kバイト）
μPD70F3204YF1-EA6	〃	〃

備考1.　xxxはROMコード番号です。

2.　ROMレス品はありません。

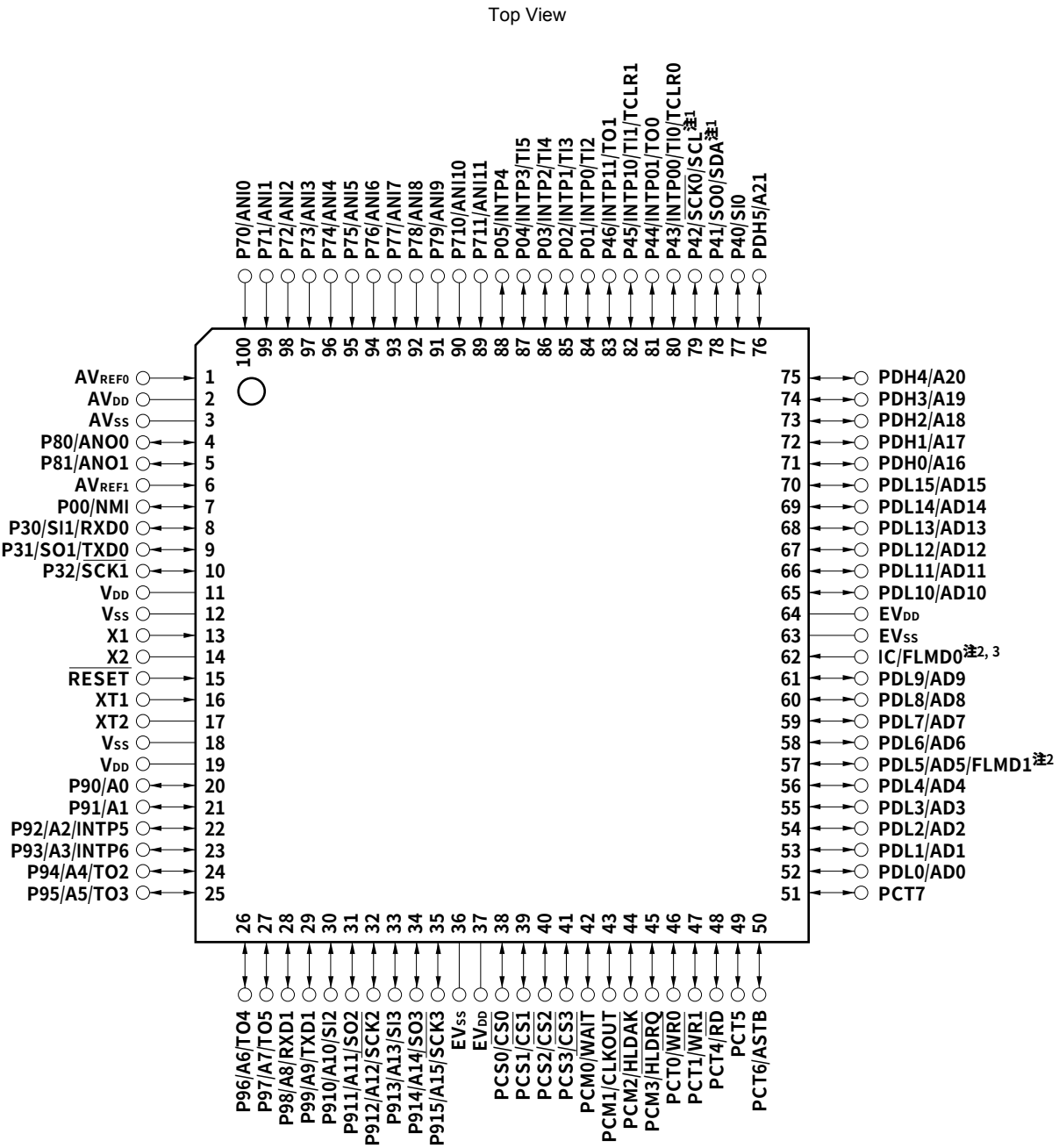
1.5　端子接続図

○V850ES/SA2

100ピン・プラスチックTQFP（ファインピッチ）（14×14）

- μPD703200GC-xxx-YEU
- μPD703201GC-xxx-YEU
- μPD70F3201GC-YEU
- μPD703200YGC-xxx-YEU
- μPD703201YGC-xxx-YEU
- μPD70F3201YGC-YEU

Top View



注1. SCL, SDAはμPD703200Y, 703201Y, 70F3201Yだけに有効です。

2. FLMD0, FLMD1はμPD70F3201, 70F3201Yだけに有効です。

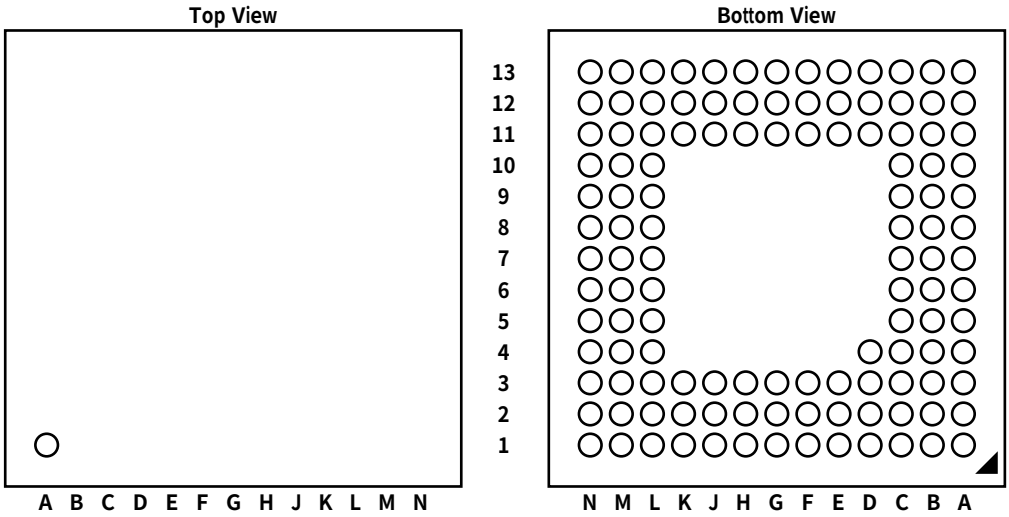
3. IC：VSSに直接接続してください（μPD703200, 703200Y, 703201, 703201Y）。
   FLMD0：通常モード時はVSSに接続してください（μPD70F3201, 70F3201Y）。

○V850ES/SA3

121ピン・プラスチックFBGA（12×12）

μPD703204F1-xxx-EA6　　μPD70F3204F1-EA6

μPD703204YF1-xxx-EA6　　μPD70F3204YF1-EA6



ピン番号	名　　称	ピン番号	名　　称	ピン番号	名　　称
A1	P70/ANI0	B8	PCD3	D2	AV_{REF1}
A2	P71/ANI1	B9	P02/INTP1/TI3	D3	P00/NMI
A3	P73/ANI3	B10	P46/INTP11/TO1	D11	PDH0/A16
A4	P713/ANI13	B11	P42/$\overline{SCK0}$/SCL[注]	D12	PDH2/A18
A5	P76/ANI6	B12	P40/SI0	D13	PDH1/A17
A6	P78/ANI8	B13	PDH4/A20	E1	P30/SI1/RXD0
A7	P711/ANI11	C1	P80/ANO0	E2	P31/SO1/TXD0
A8	P04/INTP3/TI5	C2	AV_{SS}	E3	P32/SCK1
A9	PCD2	C3	P74/ANI4	E11	PDL14/AD14
A10	P45/INTP10/TI1/TCLR1	C4	P714/ANI14	E12	PDH6/A22
A11	P43/INTP00/TI0/TCLR0	C5	P715/ANI15	E13	PDL15/AD15
A12	P41/SO0/SDA[注]	C6	P79/ANI9	F1	V_{SS}
A13	PDH5/A21	C7	P05/INTP4	F2	X1
B1	AV_{DD}	C8	P03/INTP2/TI4	F3	V_{DD}
B2	AV_{REF0}	C9	PCD1	F11	PDL11/AD11
B3	P72/ANI2	C10	P01/INTP0/TI2	F12	PDL13/AD13
B4	P712/ANI12	C11	P44/INTP01/TO0	F13	PDL12/AD12
B5	P75/ANI5	C12	PDH3/A19	G1	$\overline{RESET}$
B6	P77/ANI7	C13	PDH7/A23	G2	XT1
B7	P710/ANI10	D1	P81/ANO1	G3	X2

注　SCL, SDAはμPD703204Y, 70F3204Yだけに有効です。

備考　D4端子は，オープンにしてください。

（2/2）

ピン番号	名　　称	ピン番号	名　　称	ピン番号	名　　称
G11	EV_{SS}	K13	PDL3/AD3	M7	PCS4
G12	PDL10/AD10	L1	P93/A3/INTP6	M8	PCM0/$\overline{WAIT}$
G13	EV_{DD}	L2	P94/A4/TO2	M9	PCM2/$\overline{HLDAK}$
H1	V_{SS}	L3	P911/A11/SO2	M10	PCT3
H2	V_{DD}	L4	P914/A14/SO3	M11	PCT4/$\overline{RD}$
H3	XT2	L5	P915/A15/$\overline{SCK3}$	M12	PCT7
H11	PDL8/AD8	L6	EV_{DD}	M13	PDL0/AD0
H12	IC/FLMD0[注1, 2]	L7	PCS0/$\overline{CS0}$	N1	P96/A6/TO4
H13	PDL9/AD9	L8	PCS2/$\overline{CS2}$	N2	P98/A8/RXD1
J1	P20/SI4	L9	PCM4	N3	P910/A10/SI2
J2	P91/A1	L10	PCT2	N4	P912/A12/$\overline{SCK2}$
J3	P90/A0	L11	PCT0/$\overline{WR0}$	N5	PCS7
J11	PDL5/AD5/FLMD1[注1]	L12	PDL1/AD1	N6	PCS6
J12	PDL7/AD7	L13	PDL2/AD2	N7	PCS1/$\overline{CS1}$
J13	PDL6/AD6	M1	P95/A5/TO3	N8	PCS3/$\overline{CS3}$
K1	P22/$\overline{SCK4}$	M2	P97/A7/TO5	N9	PCM5
K2	P92/A2/INTP5	M3	P99/A9/TXD1	N10	PCM3/$\overline{HLDRQ}$
K3	P21/SO4	M4	P913/A13/SI3	N11	PCT1/$\overline{WR1}$
K11	PCM1/CLKOUT	M5	EV_{SS}	N12	PCT5
K12	PDL4/AD4	M6	PCS5	N13	PCT6/ASTB

注1. FLMD0, FLMD1はμPD70F3204, 70F3204Yだけに有効です。

2. IC：V_{SS}に直接接続してください（μPD703204, 703204Y）。

FLMD0：通常モード時はV_{SS}に接続してください（μPD70F3204, 70F3204Y）。

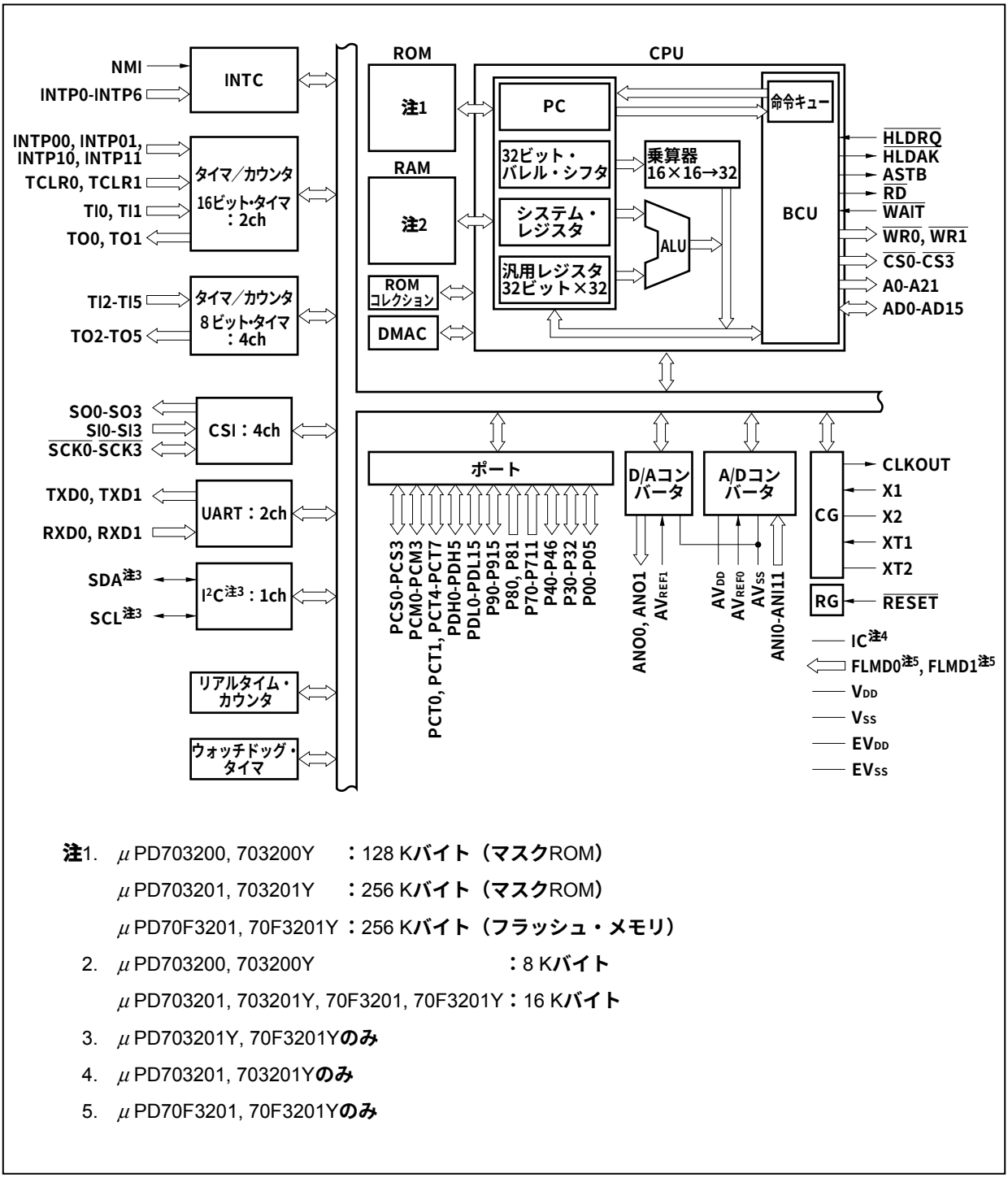
端子名称

A0-A23　：Address Bus
AD0-AD15　：Address/Data Bus
ADTRG　：A/D Trigger Input
ANI0-ANI15　：Analog Input
ANO0, ANO1　：Analog Output
ASTB　：Address Strobe
AV_{DD}　：Analog V_{DD}
AV_{REF0}, AV_{REF1}　：Analog Reference Voltage
AV_{SS}　：Analog V_{SS}
CLKOUT　：Clock Output
$\overline{CS0}$-$\overline{CS3}$　：Chip Select
EV_{DD}　：Power Supply for Port
EV_{SS}　：Ground for Port
FLMD0, FLMD1　：Flash Programming Mode
$\overline{HLDAK}$　：Hold Acknowledge
$\overline{HLDRQ}$　：Hold Request
IC　：Internally Connected
INTP0-INTP6　：External Interrupt Input
INTP00, INTP01, INTP10, INTP11　：Interrupt Request to Timer
NMI　：Non-maskable Interrupt Request
P00-P05　：Port 0
P20-P22　：Port 2
P30-P32　：Port 3
P40-P46　：Port 4
P70-P715　：Port 7
P80, P81　：Port 8
P90-P915　：Port 9
PCD1-PCD3　：Port CD
PCM0-PCM5　：Port CM
PCS0-PCS7　：Port CS
PCT0-PCT7　：Port CT
PDH0-PDH7　：Port DH
PDL0-PDL15　：Port DL
$\overline{RD}$　：Read Strobe
$\overline{RESET}$　：Reset
RXD0, RXD1　：Receive Data
$\overline{SCK0}$-$\overline{SCK4}$　：Serial Clock
SCL　：Serial Clock
SDA　：Serial Data
SI0-SI4　：Serial Input
SO0-SO4　：Serial Output
TCLR0, TCLR1　：Timer Clear Input
TI0-TI5　：Timer Input
TO0-TO5　：Timer Output
TXD0, TXD1　：Transmit Data
V_{DD}　：Power Supply
V_{SS}　：Ground
$\overline{WAIT}$　：Wait
$\overline{WR0}$　：Lower Byte Write Strobe
$\overline{WR1}$　：Upper Byte Write Strobe
X1, X2　：Crystal for Main Clock
XT1, XT2　：Crystal for Subclock

1.6　機能ブロック構成

1.6.1　内部ブロック図

• V850ES/SA2



注1.　μPD703200, 703200Y　　：128 K**バイト（マスク**ROM）

μPD703201, 703201Y　　：256 K**バイト（マスク**ROM）

μPD70F3201, 70F3201Y：256 K**バイト（フラッシュ・メモリ）**

2.　μPD703200, 703200Y　　：8 K**バイト**

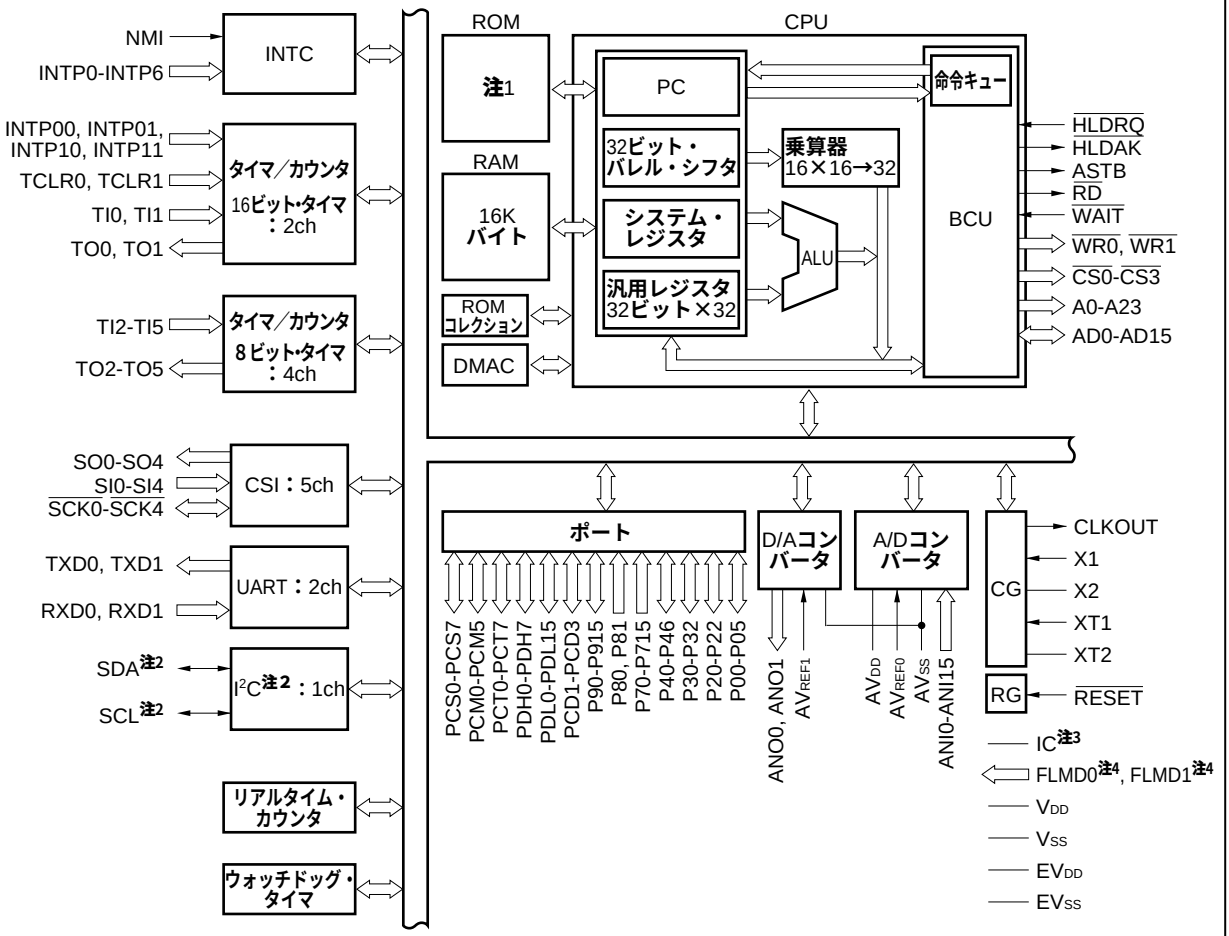
μPD703201, 703201Y, 70F3201, 70F3201Y：16 K**バイト**

3.　μPD703201Y, 70F3201Y**のみ**

4.　μPD703201, 703201Y**のみ**

5.　μPD70F3201, 70F3201Y**のみ**

• V850ES/SA3



注1.　μPD703204, 703204Y　　：256 K**バイト（マスク**ROM）

　　μPD70F3204, 70F3204Y：256 K**バイト（フラッシュ・メモリ）**

2. μPD703204Y, 70F3204Y**のみ**
3. μPD703204, 703204Y**のみ**
4. μPD70F3204, 70F3204Y**のみ**

1.6.2　内部ユニット

(1) CPU

アドレス計算，算術論理演算，データ転送などのほとんどの命令処理を5段パイプライン制御により1クロックで実行できます。

乗算器（16ビット×16ビット→32ビット），バレル・シフタ（32ビット）などの専用ハードウエアを内蔵し，複雑な命令処理の高速化を図っています。

(2) バス・コントロール・ユニット（BCU）

CPUで得られた物理アドレスに基づいて必要な外部バス・サイクルを起動します。外部メモリ領域から命令フェッチするときにCPUからのバス・サイクル起動の要求がない場合は，プリフェッチ・アドレスを生成し，命令コードのプリフェッチを行います。プリフェッチされた命令コードは，内部の命令キューに取り込まれます。

(3) ROM

0000000H-003FFFFH/0000000H-001FFFFH番地にマッピングされる256 K/128 Kバイトのマスク ROMまたはフラッシュ・メモリです。命令フェッチ時にCPUから1クロックでアクセスできます。

(4) RAM

3FFB000H-3FFEFFFH/3FFD000H-3FFEFFFH番地にマッピングされる16 K/8 KバイトのRAMです。データ・アクセス時にCPUから1クロックでアクセスできます。

(5) 割り込みコントローラ（INTC）

内蔵周辺ハードウエア，および外部からのハードウエア割り込み要求（NMI, INTP0-INTP6）を処理します。これらの割り込み要求は，8レベルの割り込み優先順位を指定でき，割り込み要因に対し多重処理制御ができます。

(6) クロック・ジェネレータ（CG）

メイン・クロック（f_X）用とサブクロック（f_{XT}）用の2種類の発振回路を内蔵しています。7種類（f_{XX}, $f_{XX}/2$, $f_{XX}/4$, $f_{XX}/8$, $f_{XX}/16$, $f_{XX}/32$, f_{XT}）のクロックを生成し，そのうちの1つをCPUの動作クロック（f_{CPU}）として供給します。サブクロックは，リアルタイム・カウンタの動作クロックとしてのみ選択できます。

(7) タイマ／カウンタ

16ビットのタイマ／イベント・カウンタを2チャネルと，8ビットのタイマ／イベント・カウンタを4チャネル内蔵し，パルス間隔や周波数の計測，プログラマブルなパルスの出力ができます。

2チャネルの8ビット・タイマ／イベント・カウンタをカスケード接続し，16ビット・タイマとしても使用できます。

(8) リアルタイム・カウンタ（時計用）

サブクロック（32.768 kHz）またはメイン・クロックから時計カウント用の基準時間（1秒）をカウントします。メイン・クロックによるインターバル・タイマとしても同時に使用できます。週，日，時，分，秒の専用ハードウエア・カウンタを持ち，最長4095週までカウントが可能です。

(9) ウォッチドッグ・タイマ

プログラムの暴走，システム異常などを検出するためのウォッチドッグ・タイマを内蔵しています。

インターバル・タイマとしても使用できます。

ウォッチドッグ・タイマとして使用する場合は，オーバフローでノンマスカブル割り込み要求（INTWDT）を発生します。インターバル・タイマとして使用する場合は，オーバフローでマスカブル割り込み要求（INTWDTM）を発生します。

(10) シリアル・インタフェース（SIO）

V850ES/SA2, V850ES/SA3には，シリアル・インタフェースとしてアシンクロナス・シリアル・インタフェース（UART0, UART1），クロック同期式シリアル・インタフェース（V850ES/SA2：CSI0-CSI3, V850ES/SA3：CSI0-CSI4），I^2Cバス・インタフェース（I^2C）を内蔵しており，V850ES/SA2は最大4チャネルを，V850ES/SA3は最大5チャネルを同期に使用できます。このうち1チャネルはUARTとCSIの切り替えができます。もう1チャネルはCSIとI^2Cの切り替えができます。

UART0, UART1は，TXD0, TXD1, RXD0, RXD1端子によりデータ転送を行います。

CSI0-CSI3は，SO0-SO3, SI0-SI3, $\overline{SCK0}$-$\overline{SCK3}$端子によりデータ転送を行います。

CSI4は，SO4, SI4, $\overline{SCK4}$端子によりデータ転送を行います（V850ES/SA3のみ）。

I^2Cは，SDA, SCL端子によりデータ転送を行います。

I^2Cは，μPD703200Y, 703201Y, 703204Y, 70F3201Y, 70F3204Yのみ内蔵しています。

UARTは専用ボー・レート・ジェネレータを内蔵しています。

(11) A/Dコンバータ

V850ES/SA2は12本，V850ES/SA3は16本のアナログ入力端子を持つ高速，高分解能の10ビットA/Dコンバータです。逐次変換方式で変換します。

(12) D/Aコンバータ

8ビット分解能のD/Aコンバータを2チャネル内蔵しています。Rストリング方式です。

(13) DMAコントローラ

4チャネルのDMAコントローラを内蔵しています。内蔵周辺I/Oによる割り込み要求に基づいて，内蔵RAM，内蔵周辺I/O，外部メモリ間でデータを転送します。

(14) ROMコレクション

マスクROM内のプログラムの一部を内蔵RAMのプログラムへ置き換えて実行する機能です。4箇所修正可能です。

（15）ポート

次に示すように，汎用ポートとしての機能と制御端子の機能があります。

ポート	入出力	ポート機能	制御機能
P0	6ビット入出力	汎用ポート	NMI，外部割り込み，タイマ入力
P2[注]	3ビット入出力		シリアル・インタフェース
P3	3ビット入出力		シリアル・インタフェース
P4	7ビット入出力		シリアル・インタフェース，タイマ入出力，タイマ・トリガ
P7	12ビット入力（V850ES/SA2） 16ビット入力（V850ES/SA3）		A/Dコンバータ・アナログ入力
P8	2ビット入力		D/Aコンバータ・アナログ出力
P9	16ビット入出力		外部アドレス・バス，シリアル・インタフェース，タイマ出力，外部割り込み
PCD[注]	3ビット入出力		−
PCM	4ビット入出力（V850ES/SA2） 6ビット入出力（V850ES/SA3）		外部バス・インタフェース
PCS	4ビット入出力（V850ES/SA2） 8ビット入出力（V850ES/SA3）		チップ・セレクト出力
PCT	6ビット入出力（V850ES/SA2） 8ビット入出力（V850ES/SA3）		外部バス・インタフェース
PDH	6ビット入出力（V850ES/SA2） 8ビット入出力（V850ES/SA3）		外部アドレス・バス
PDL	16ビット入出力		外部アドレス／データ・バス

注　V850ES/SA3のみ

1.7　端子機能

1.7.1　バス・インタフェース端子

端子名称	入出力	PULL	機　　能
A0-A15	出力	あり	外部メモリに対するアドレス・バス（セパレート・バス使用時）
A16-A21 【A22, A23】	出力	なし	外部メモリに対するアドレス・バス
AD0-AD15	入出力	なし	外部メモリに対するアドレス・バス（マルチプレクスト・アドレス／データ・バス使用時）
ASTB	出力	なし	外部メモリに対するアドレス・ストローブ信号出力
$\overline{RD}$	出力	なし	外部メモリに対するリード・ストローブ信号出力
$\overline{WR0}$	出力	なし	外部メモリ（下位8ビット）に対するライト・ストローブ信号出力
$\overline{WR1}$	出力	なし	外部メモリ（上位8ビット）に対するライト・ストローブ信号出力
$\overline{HLDAK}$	出力	なし	バス・ホールド・アクノリッジ信号出力
$\overline{HLDRQ}$	入力	なし	バス・ホールド要求信号入力
$\overline{WAIT}$	入力	なし	外部ウエイト信号入力
$\overline{CS0}$-$\overline{CS3}$	出力	なし	チップ・セレクト信号出力

備考　PULL：内蔵プルアップ抵抗

【　】：V850ES/SA3のみの端子

1. 7. 2　内蔵周辺I/O端子

端子名称	入出力	PULL	機　　能
INTP0-INTP6	入力	あり	外部割り込み要求信号入力（マスカブル）
INTP00, INTP01, INTP10, INTP11	入力	あり	キャプチャ・トリガ入力（TM0, TM1）
NMI	入力	あり	外部割り込み要求信号入力（ノンマスカブル）
ANI0-ANI11, 【ANI12-ANI15】	入力	なし	A/Dコンバータ用アナログ電圧入力
ANO0, ANO1	出力	なし	D/Aコンバータ用アナログ電圧出力
RXD0, RXD1	入力	あり	シリアル受信データ入力（UART0, UART1）
$\overline{\text{SCK0}}$-$\overline{\text{SCK3}}$, 【SCK4】	入出力	あり	シリアル・クロック入出力（CSI0-CSI3,【CSI4】）
SCL[注]	入出力	あり	シリアル・クロック入出力（I^2C）
SDA[注]	入出力	あり	シリアル送受信データ入出力（I^2C）
SI0-SI3,【SI4】	入力	あり	シリアル受信データ入力（CSI0-SI3,【CSI4】）
SO0-SO3,【SO4】	出力	あり	シリアル送信データ出力（CSI0-SI3,【CSI4】）
TCLR0, TCLR1	入力	あり	タイマ・クリア入力（TM0,TM1）
TI0-TI5	入力	あり	外部イベント／クロック入力（TM0-TM5）
TO0-TO5	出力	あり	タイマ出力（TM0-TM5）
TXD0, TXD1	出力	あり	シリアル送信データ出力（UART0, UART1）

注　μPD703200Y, 703201Y, 703204Y, 70F3201Y, 70F3204Yのみ

備考　PULL：内蔵プルアップ抵抗

【　】：V850ES/SA3のみの端子

1.7.3　電源，クロック，リセット端子

端子名称	入出力	PULL	機　　能
V_{DD}	−	−	内部用正電源供給端子
V_{SS}	−	−	内部用グランド電位
EV_{DD}	−	−	入出力ポート用正電源供給端子
EV_{SS}	−	−	入出力ポート用グランド電位
AV_{DD}	−	−	A/Dコンバータ用正電源供給端子
AV_{SS}	−	−	A/Dコンバータ用グランド電位
AV_{REF0}	入力	−	A/Dコンバータ用基準電圧入力
AV_{REF1}	入力	−	D/Aコンバータ用基準電圧入力
IC	−	−	内部接続端子（V_{SS}に接続してください）
X1	入力	なし	メイン・クロック用発振子接続
X2	−	なし	メイン・クロック用発振子接続
XT1	入力	なし	サブクロック用発振子接続
XT2	−	なし	サブクロック用発振子接続
CLKOUT	出力	なし	内部システム・クロック出力
$\overline{\text{RESET}}$	入力	−	システム・リセット入力
FLMD0, FLMD1[注]	入力	なし	フラッシュ・メモリ・プログラミング・モード用正電源供給端子

注　μPD70F3201, 70F3201Y, 70F3204, 70F3204Yのみ

備考　PULL：内蔵プルアップ抵抗

1.7.4　ポート端子

(1/3)

端子名称	入出力	内蔵プルアップ抵抗	機　　能	兼用端子
P00	入出力	あり	ポート0 6ビット入出力ポート 1ビット単位で入力／出力の指定が可能	NMI
P01				INTP0/TI2
P02				INTP1/TI3
P03				INTP2/TI4
P04				INTP3/TI5
P05				INTP4
【P20】	入出力	あり	ポート2 3ビット入出力ポート 1ビット単位で入力／出力の指定が可能 1ビット単位でN-chオープン・ドレーンの指定が可能 (P21, P22のみ)	【SI4】
【P21】				【SO4】
【P22】				【$\overline{\text{SCK4}}$】
P30	入出力	あり	ポート3 3ビット入出力ポート 1ビット単位で入力／出力の指定が可能 1ビット単位でN-chオープン・ドレーンの指定が可能 (P31, P32のみ)	SI1/RXD0
P31				SO1/TXD0
P32				$\overline{\text{SCK1}}$
P40	入出力	あり	ポート4 7ビット入出力ポート 1ビット単位で入力／出力の指定が可能 1ビット単位でN-chオープン・ドレーンの指定が可能 (P41, P42のみ)	SI0
P41				SO0/SDA[注]
P42				$\overline{\text{SCK0}}$/SCL[注]
P43				INTP00/TI0/TCLR0
P44				INTP01/TO0
P45				INTP10/TI1/TCLR1
P46				INTP11/TO1
P70	入力	なし	ポート7 12ビット入力ポート（V850ES/SA2） 16ビット入力ポート（V850ES/SA3）	ANI0
P71				ANI1
P72				ANI2
P73				ANI3
P74				ANI4
P75				ANI5
P76				ANI6
P77				ANI7
P78				ANI8
P79				ANI9
P710				ANI10
P711				ANI11
【P712】				【ANI12】
【P713】				【ANI13】
【P714】				【ANI14】
【P715】				【ANI15】

注　μPD703200Y, 703201Y, 703204Y, 70F3201Y, 70F3204Yのみ

備考　【　】内はV850ES/SA3のみの端子

(2/3)

端子名称	入出力	内蔵プルアップ抵抗	機　　能	兼用端子
P80	入力	なし	ポート8	ANO0
P81			2ビット入力ポート	ANO1
P90	入出力	あり	ポート9	A0
P91			16ビット入出力ポート	A1
P92			1ビット単位で入力／出力の指定が可能	A2/INTP5
P93			1ビット単位でN-chオープン・ドレーンの指定が可能	A3/INTP6
P94			（P911, P912, P914, P915のみ）	A4/TO2
P95				A5/TO3
P96				A6/TO4
P97				A7/TO5
P98				A8/RXD1
P99				A9/TXD1
P910				A10/SI2
P911				A11/SO2
P912				A12/$\overline{\text{SCK2}}$
P913				A13/SI3
P914				A14/SO3
P915				A15/$\overline{\text{SCK3}}$
【PCD1】	入出力	なし	ポートCD	−
【PCD2】			3ビット入出力ポート	−
【PCD3】			1ビット単位で入力／出力の指定が可能	−
PCM0	入出力	なし	ポートCM	$\overline{\text{WAIT}}$
PCM1			4ビット入出力ポート（V850ES/SA2）	CLKOUT
PCM2			6ビット入出力ポート（V850ES/SA3）	$\overline{\text{HLDAK}}$
PCM3			1ビット単位で入力／出力の指定が可能	$\overline{\text{HLDRQ}}$
【PCM4】				−
【PCM5】				−
PCS0	入出力	なし	ポート10	$\overline{\text{CS0}}$
PCS1			4ビット入出力ポート（V850ES/SA2）	$\overline{\text{CS1}}$
PCS2			8ビット入出力ポート（V850ES/SA3）	$\overline{\text{CS2}}$
PCS3			1ビット単位で入力／出力の指定が可能	$\overline{\text{CS3}}$
【PCS4】				−
【PCS5】				−
【PCS6】				−
【PCS7】				−
PCT0	入出力	なし	ポートCT	$\overline{\text{WR0}}$
PCT1			6ビット入出力ポート（V850ES/SA2）	$\overline{\text{WR1}}$
【PCT2】			8ビット入出力ポート（V850ES/SA3）	−
【PCT3】			1ビット単位で入力／出力の指定が可能	−
PCT4				$\overline{\text{RD}}$
PCT5				−
PCT6				ASTB
PCT7				−

備考　【　】内はV850ES/SA3のみの端子

（3/3）

端子名称	入出力	内蔵プルアップ抵抗	機　　能	兼用端子
PDH0	入出力	なし	ポートDH 6ビット入出力ポート（V850ES/SA2） 8ビット入出力ポート（V850ES/SA3） 1ビット単位で入力／出力の指定が可能	A16
PDH1				A17
PDH2				A18
PDH3				A19
PDH4				A20
PDH5				A21
【PDH6】				【A22】
【PDH7】				【A23】
PDL0	入出力	なし	ポートDL 16ビット入出力ポート 1ビット単位で入力／出力の指定が可能	AD0
PDL1				AD1
PDL2				AD2
PDL3				AD3
PDL4				AD4
PDL5				AD5/FLMD1[注]
PDL6				AD6
PDL7				AD7
PDL8				AD8
PDL9				AD9
PDL10				AD10
PDL11				AD11
PDL12				AD12
PDL13				AD13
PDL14				AD14
PDL15				AD15

注　μPD70F3201, 70F3201Y, 70F3204, 70F3204Yのみ

備考　【　】内はV850ES/SA3のみの端子

1. 8　CPUレジスタ・セット

V850ES/SA2, V850ES/SA3のレジスタは，汎用のプログラム・レジスタ・セットと，専用のシステム・レジスタ・セットの2種類に分類できます。すべてのレジスタは32ビット幅となっています。

詳細はV850ES　**ユーザーズ・マニュアル　アーキテクチャ編**を参照してください。

（1）プログラム・レジスタ・セット

31		0
r0	（ゼロ・レジスタ）	
r1	（アセンブラ予約レジスタ）	
r2		
r3	（スタック・ポインタ（SP））	
r4	（グローバル・ポインタ（GP））	
r5	（テキスト・ポインタ（TP））	
r6		
r7		
r8		
r9		
r10		
r11		
r12		
r13		
r14		
r15		
r16		
r17		
r18		
r19		
r20		
r21		
r22		
r23		
r24		
r25		
r26		
r27		
r28		
r29		
r30	（エレメント・ポインタ（EP））	
r31	（リンク・ポインタ（LP））	

31		0
PC	（プログラム・カウンタ）	

（2）システム・レジスタ・セット

31		0
EIPC	（割り込み時状態退避レジスタ）	
EIPSW	（割り込み時状態退避レジスタ）	

FEPC	（NMI時状態退避レジスタ）
FEPSW	（NMI時状態退避レジスタ）

ECR	（割り込み要因レジスタ）

PSW	（プログラム・ステータス・ワード）

CTPC	（CALLT実行時状態退避レジスタ）
CTPSW	（CALLT実行時状態退避レジスタ）

DBPC	（例外／ディバグ・トラップ時状態退避レジスタ）
DBPSW	（例外／ディバグ・トラップ時状態退避レジスタ）

CTBP	（CALLTベース・ポインタ）

1.8.1　プログラム・レジスタ・セット

プログラム・レジスタには，汎用レジスタとプログラム・カウンタがあります。

(1) 汎用レジスタ（r0-r31）

汎用レジスタとして，r0-r31の32本が用意されています。これらのレジスタは，どれでもデータ変数またはアドレス変数として利用できます。

ただし，r0とr30は命令により暗黙的に使用しますので，これらのレジスタを使用する際には注意が必要です。また，r1, r3-r5, r31は，アセンブラとCコンパイラが暗黙的に使用しますので，これらのレジスタを使用する際にはレジスタの内容を破壊しないように退避してから使用し，使用後に元に戻す必要があります。r2は，リアルタイムOSが使用する場合があります。使用するリアルタイムOSがr2を使用していない場合は，変数用レジスタとしてr2を使用できます。

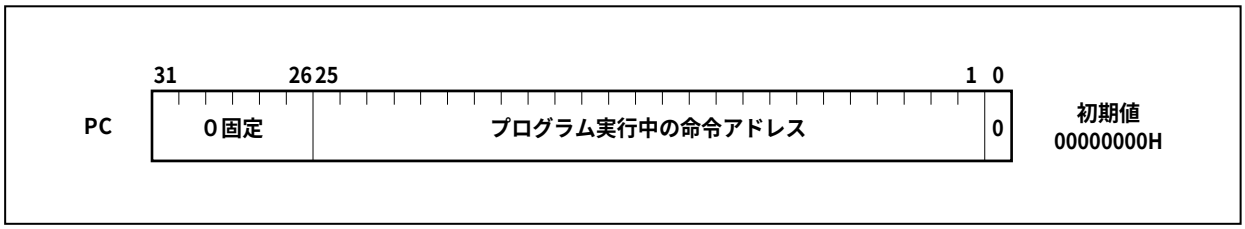
表1－2　プログラム・レジスター覧

名　称	用　途	動　作
r0	ゼロ・レジスタ	常に0を保持
r1	アセンブラ予約レジスタ	32ビット・イミーディエト作成用のワーキング・レジスタとして使用
r2	アドレス／データ変数用レジスタ（使用するリアルタイムOSがr2を使用していない場合）	
r3	スタック・ポインタ	関数コール時のスタック・フレーム生成時に使用
r4	グローバル・ポインタ	データ領域のグローバル変数をアクセスするときに使用
r5	テキスト・ポインタ	テキスト領域（プログラム・コードを配置する領域）の先頭を指すレジスタとして使用
r6-r29	アドレス／データ変数用レジスタ	
r30	エレメント・ポインタ	メモリをアクセスするときのベース・ポインタとして使用
r31	リンク・ポインタ	コンパイラが関数コールをするときに使用
PC	プログラム・カウンタ	プログラム実行中の命令アドレスを保持

(2) プログラム・カウンタ（PC）

プログラム実行中の命令アドレスを保持します。下位26ビットが有効で，ビット31-26は0に固定されます。ビット25からビット26へのキャリーがあっても無視します。

また，ビット0は0に固定されており，奇数番地への分岐はできません。



1. 8. 2　システム・レジスタ・セット

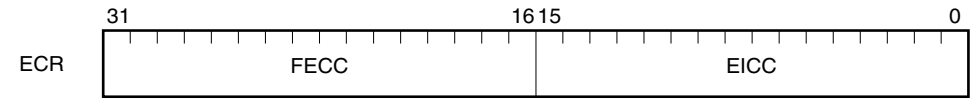
システム・レジスタは，CPUの状態制御，割り込み情報保持などを行います。

番　号	名　称	用　途	動　作
0	EIPC	割り込み時状態退避レジスタ	例外または割り込みが発生した場合に，PCとPSWを退避するレジスタです。このレジスタは1組しかないため，多重割り込みを許可する場合は，プログラムでこのレジスタの内容を退避してください。
1	EIPSW		
2	FEPC	NMI時状態退避レジスタ	NMIが発生した場合に，PCとPSWを退避するレジスタです。このレジスタは1組しかないため，多重割り込みを許可する場合は，プログラムでこのレジスタの内容を退避してください。
3	FEPSW		
4	ECR	割り込み要因レジスタ	例外，マスカブル割り込み，NMIが発生した場合に，その要因を保持するレジスタです。このレジスタの上位16ビットは“FECC“と呼ばれ，NMIの例外コードがセットされます。下位16ビットは”EICC”と呼ばれ，例外／割りこみの例外コードがセットされます。
5	PSW	プログラム・ステータス・ワード	プログラムの状態（命令実行結果）やCPUの状態を示すフラグの集合です。
16	CTPC	CALLT実行時状態退避レジスタ	CALLT命令実行した場合のPCとPSWを退避するレジスタです。
17	CTPSW		
18	DBPC	例外／ディバグ・トラップ時状態退避レジスタ	例外／ディバグ・トラップが発生した場合のPCとPSWを退避するレジスタです。
19	DBPSW		
20	CTBP	CALLTベース・ポインタ	CALLT命令実行時のターゲット・アドレス生成に使用するレジスタです。
6-15 21-31	－	予約	－

備考　これらのシステム・レジスタへのリード／ライトは，システム・レジスタ・ロード／ストア命令（LDSR命令／STSR命令）で示すシステム・レジスタ番号を指定します。

(1) 割り込み要因レジスタ（ECR）

割り込み要因レジスタ（ECR）は，例外や割り込みが発生した場合に，その要因を保持するレジスタです。ECRが保持する値は，割り込み要因ごとにコード化された例外コードです。なお，このレジスタは読み出し専用のため，LDSR命令を使ってこのレジスタにデータを書き込むことはできません。



初期値
00000000H

ビット位置	ビット名	意　味
31-16	FECC	ノンマスカブル割り込み（NMI）の例外コード
15-0	EICC	例外，マスカブル割り込みの例外コード

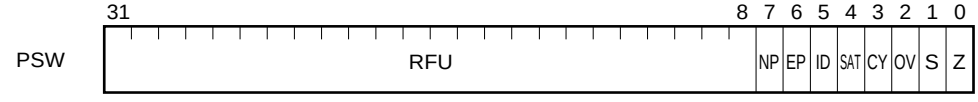
(2) プログラム・ステータス・ワード（PSW）

プログラム・ステータス・ワード（PSW）は，プログラムの状態（命令実行の結果）やCPUの状態を示すフラグの集合です。

LDSR命令を使用してこのレジスタの各ビットの内容を変更した場合は，LDSR命令実行終了直後から変更内容が有効となります。ただし，IDフラグをセット“1”する場合，LDSR命令実行中から割り込み要求の受け付けを禁止します。

なお，ビット31-8は，将来の機能拡張のために予約されています（0に固定）。

(1/2)



初期値
00000020H

ビット位置	フラグ名	意　味
31-8	RFU	予約フィールドです。“0”に固定されています。
7	NP	ノンマスカブル割り込み（NMI）処理中であることを示します。NMI要求が受け付けられるとセット“1”され，多重割り込みを禁止します。 0：NMI処理中でない。 1：NMI処理中である。
6	EP	例外処理中であることを示します。例外の発生でセット“1”されます。なお，このビットがセットされても割り込み要求は受け付けます。 0：例外処理中でない。 1：例外処理中である。
5	ID	マスカブル割り込み要求を受け付ける状態かどうかを示します。 0：割り込み可（EI） 1：割り込み不可（DI）

(2/2)

ビット位置	フラグ名	意　味
4	SAT[注]	飽和演算命令の演算結果がオーバフローし，演算結果が飽和していることを示します。累積フラグのため，飽和演算命令で演算結果が飽和するとセット“1”され，以降の命令の演算結果が飽和しなくてもクリア“0”されません。クリア“0”する場合は，LDSR命令により行います。なお，算術演算命令の実行では，セット“1”もクリア“0”も行いません。 0：飽和していない。 1：飽和している。
3	CY	演算結果にキャリー，またはボローがあったかどうかを示します。 0：キャリー，またはボローは発生していない。 1：キャリー，またはボローが発生した。
2	OV[注]	演算中にオーバフローが発生したかどうかを示します。 0：オーバフローは発生していない。 1：オーバフローが発生した。
1	S[注]	演算の結果が負かどうかを示します。 0：演算の結果は，正または0であった。 1：演算の結果は負であった。
0	Z	演算の結果が0かどうかを示します。 0：演算の結果は0でなかった。 1：演算の結果は0であった。

注　飽和演算時の OV フラグと S フラグの内容で飽和処理した演算結果が決まります。また，飽和演算時に OV フラグがセット（1）された場合だけ，SAT フラグはセット（1）されます。

演算結果の状態	フラグの状態			飽和処理をした演算結果
	SAT	OV	S	
正の最大値を越えた	1	1	0	7FFFFFFFH
負の最大値を越えた	1	1	1	80000000H
正（最大値を越えない）	演算前の値を保持	0	0	演算結果そのもの
負（最大値を越えない）			1	

1.9　動作モード

V850ES/SA2, V850ES/SA3は次に示す動作モードを備えます。

(1) 通常動作モード

システム・リセット解除後，バス・インタフェース関連の各端子はポート・モードになり，内蔵ROMのリセット・エントリ・アドレスに分岐し，命令処理を開始します。ソフトウエアでPMCDH, PMCDL, PMCCM, PMCCS, PMCCTレジスタをコントロール・モードに設定することにより，外部メモリ領域に外部デバイスを接続できます。

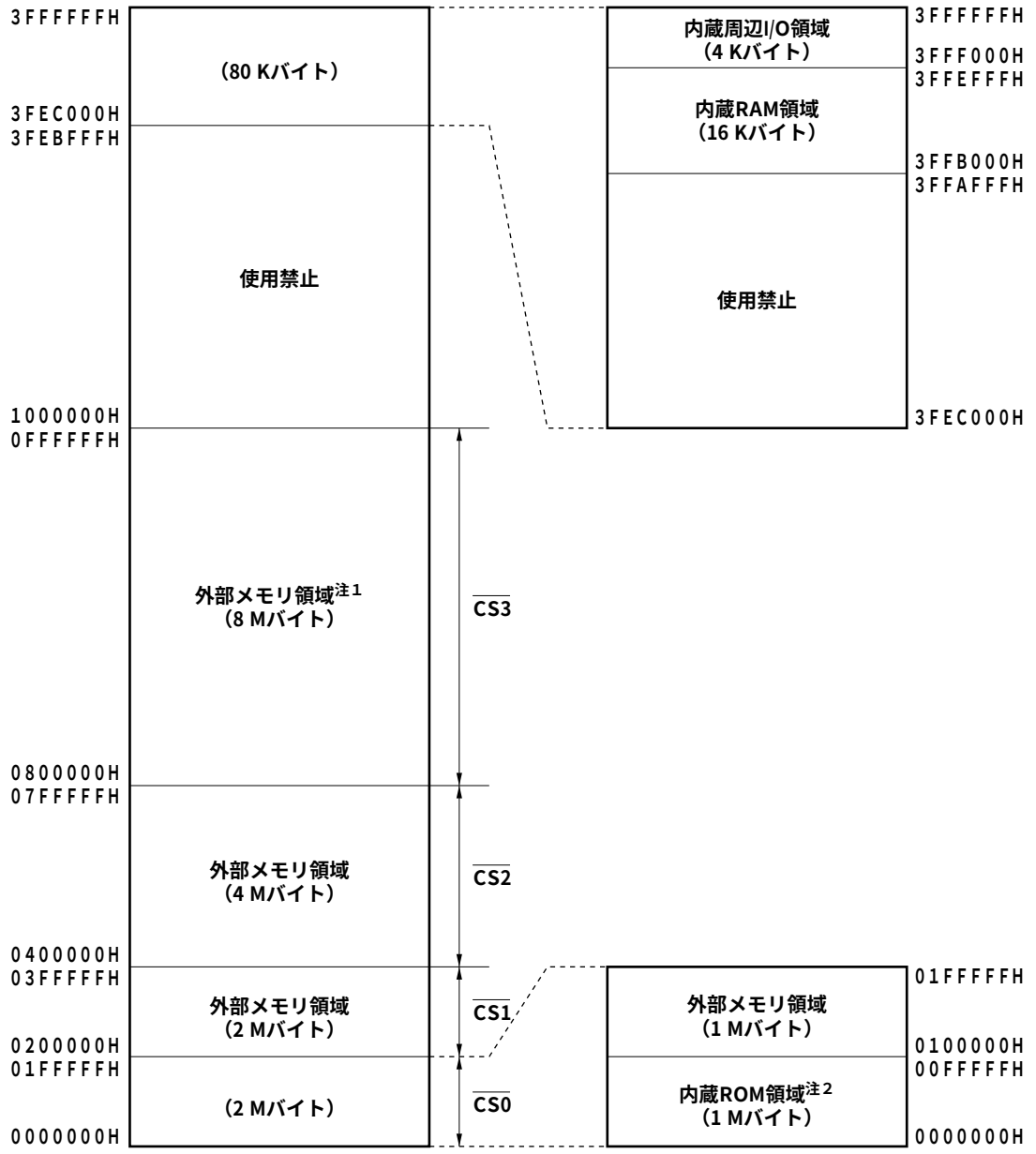
(2) フラッシュ・メモリ・プログラミング・モード（μPD70F3201, 70F3201Y, 70F3204, 70F3204Y**）**

このモードを指定すると，フラッシュ・ライタによる内蔵フラッシュ・メモリへのプログラム動作が可能になります。

1. 10　アドレス空間

V850ES/SA2, V850ES/SA3では，図1－1に示すように各領域を予約しています。

図1－1　データ・メモリ・マップ（物理アドレス）



注1. V850ES/SA2では0800000H-0BFFFFFHの4 Mバイト空間となります（0C00000H-0FFFFFFHは，0800000H-0BFFFFFHのイメージです）。

2. 0000000H-00FFFFFH番地へのフェッチ・アクセスおよび，リード・アクセスは内蔵ROM領域に対して行われますが，データ・ライト・アクセス時は，外部メモリ領域として行われます。

1. 10. 1　割り込み／例外テーブル

V850ES/SA2, V850ES/SA3は，割り込み／例外に対応したハンドラ・アドレスを固定化することにより，割り込み応答性を高速化しています。

このハンドラ・アドレスの集合を割り込み／例外テーブルと呼び，内蔵ROM領域に置かれています。割り込み／例外要求が受け付けられると，ハンドラ・アドレスにジャンプし，そのメモリに置かれているプログラムを実行します。表1－3に割り込み／例外要因と，対応するアドレスを示します。

表1－3　割り込み／例外テーブル

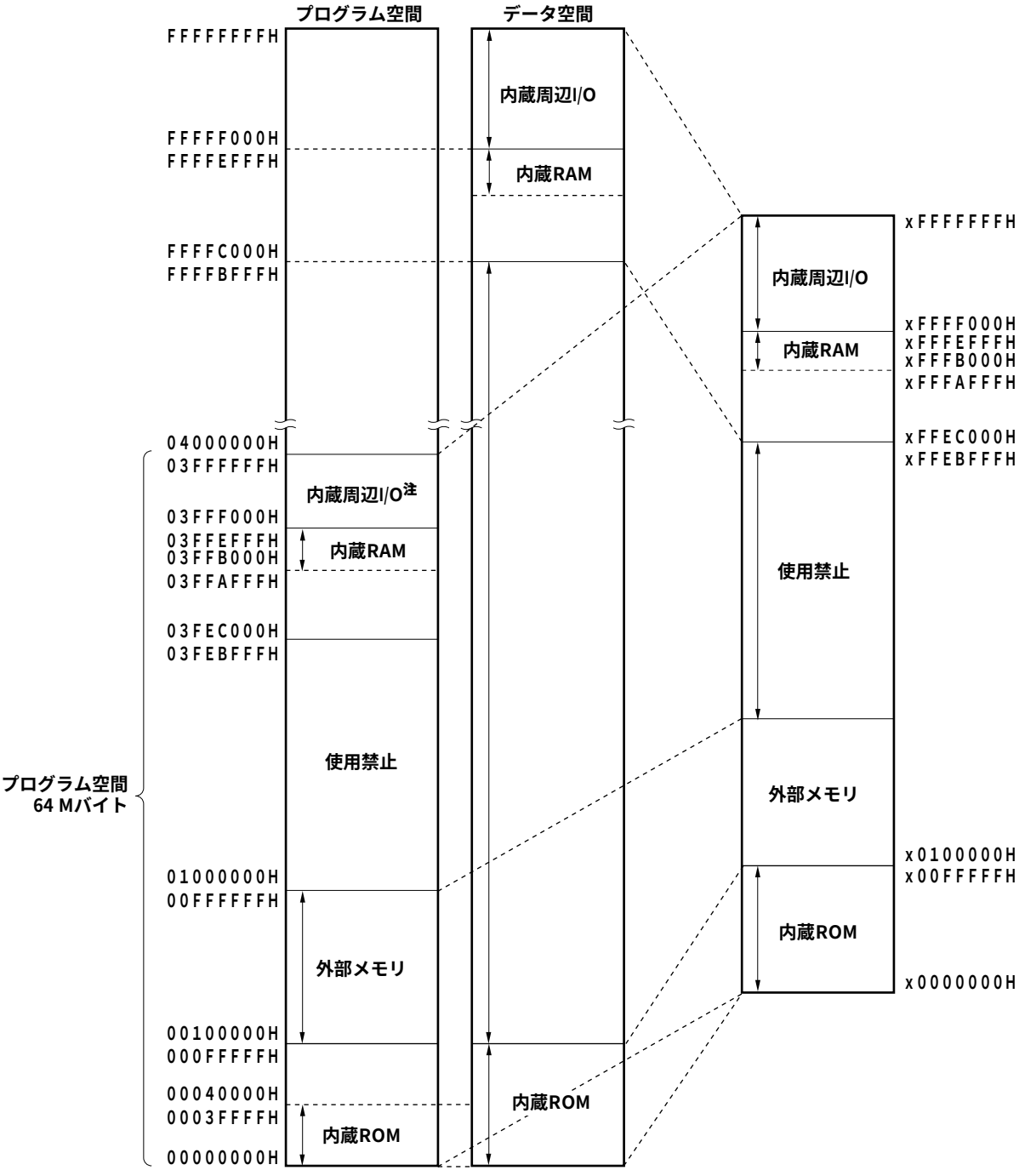
割り込み／例外テーブルの先頭アドレス	割り込み／例外要因	割り込み／例外テーブルの先頭アドレス	割り込み／例外要因
00000000H	RESET	00000180H	INTTM3
00000010H	NMI	00000190H	INTTM4
00000020H	INTWDT	000001A0H	INTTM5
00000040H	TRAP0n（n = 0-F）	000001B0H	INTCSI0
00000050H	TRAP1n（n = 0-F）	000001C0H	INTIIC[注1]
00000060H	ILGOP/DBG0	000001D0H	INTCSI1
00000080H	INTWDTM	000001E0H	INTSRE0
00000090H	INTP0	000001F0H	INTSR0
000000A0H	INTP1	00000200H	INTST0
000000B0H	INTP2	00000210H	INTCSI2
000000C0H	INTP3	00000220H	INTSRE1
000000D0H	INTP4	00000230H	INTSR1
000000E0H	INTP5	00000240H	INTST1
000000F0H	INTP6	00000250H	INTCSI3
00000100H	INTRTC	00000260H	INTCSI4[注2]
00000110H	INTCC00	00000270H	INTAD
00000120H	INTCC01	00000280H	INTDMA0
00000130H	INTOVF0	00000290H	INTDMA1
00000140H	INTCC10	000002A0H	INTDMA2
00000150H	INTCC11	000002B0H	INTDMA3
00000160H	INTOVF1	000002C0H	INTROV
00000170H	INTTM2	000002D0H	INTBRG

注1.　μPD703200Y, 703201Y, 703204Y, 70F3201Y, 70F3204Y**のみ**

2.　V850ES/SA3**のみ**

1. 10. 2　推奨メモリ・マップ

図1－2　推奨メモリ・マップ



注　この領域はアクセス禁止です。この領域の内蔵周辺I/OをアクセスするときにはFFFF000H-FFFFFFFH**番地を指定してください。**

備考1.　↕ **は推奨使用領域です。**

2. **この図は，**μPD703204**の場合の推奨メモリ・マップです。**

1. 11　周辺I/Oレジスタ

(1/8)

アドレス	機能レジスタ名称	略号	R/W	操作可能ビット			初期値
				1	8	16	
FFFFF004H	ポート・レジスタDL	PDL	R/W			○	不定
FFFFF004H	ポート・レジスタDLL	PDLL		○	○		
FFFFF005H	ポート・レジスタDLH	PDLH		○	○		
FFFFF006H	ポート・レジスタDH	PDH		○	○		
FFFFF008H	ポート・レジスタCS	PCS		○	○		
FFFFF00AH	ポート・レジスタCT	PCT		○	○		
FFFFF00CH	ポート・レジスタCM	PCM		○	○		
FFFFF00EH	ポート・レジスタCD[注]	PCD		○	○		
FFFFF024H	ポート・モード・レジスタDL	PMDL				○	FFFFH
FFFFF024H	ポート・モード・レジスタDLL	PMDLL		○	○		FFH
FFFFF025H	ポート・モード・レジスタDLH	PMDLH		○	○		
FFFFF026H	ポート・モード・レジスタDH	PMDH		○	○		
FFFFF028H	ポート・モード・レジスタCS	PMCS		○	○		
FFFFF02AH	ポート・モード・レジスタCT	PMCT		○	○		
FFFFF02CH	ポート・モード・レジスタCM	PMCM		○	○		
FFFFF02EH	ポート・モード・レジスタCD[注]	PMCD		○	○		
FFFFF044H	ポート・モード・コントロール・レジスタDL	PMCDL				○	0000H
FFFFF044H	ポート・モード・コントロール・レジスタDLL	PMCDLL		○	○		00H
FFFFF045H	ポート・モード・コントロール・レジスタDLH	PMCDLH		○	○		
FFFFF046H	ポート・モード・コントロール・レジスタDH	PMCDH		○	○		
FFFFF048H	ポート・モード・コントロール・レジスタCS	PMCCS		○	○		
FFFFF04AH	ポート・モード・コントロール・レジスタCT	PMCCT		○	○		
FFFFF04CH	ポート・モード・コントロール・レジスタCM	PMCCM		○	○		
FFFFF066H	バス・サイズ・コンフィギュレーション・レジスタ	BSC				○	5555H
FFFFF06EH	システム・ウエイト・コントロール・レジスタ	VSWC			○		77H
FFFFF080H	DMAソース・アドレス・レジスタ0L	DSA0L				○	不定
FFFFF082H	DMAソース・アドレス・レジスタ0H	DSA0H				○	
FFFFF084H	DMAディスティネーション・アドレス・レジスタ0L	DDA0L				○	
FFFFF086H	DMAディスティネーション・アドレス・レジスタ0H	DDA0H				○	
FFFFF088H	DMAソース・アドレス・レジスタ1L	DSA1L				○	
FFFFF08AH	DMAソース・アドレス・レジスタ1H	DSA1H				○	
FFFFF08CH	DMAディスティネーション・アドレス・レジスタ1L	DDA1L				○	
FFFFF08EH	DMAディスティネーション・アドレス・レジスタ1H	DDA1H				○	
FFFFF090H	DMAソース・アドレス・レジスタ2L	DSA2L				○	
FFFFF092H	DMAソース・アドレス・レジスタ2H	DSA2H				○	
FFFFF094H	DMAディスティネーション・アドレス・レジスタ2L	DDA2L				○	
FFFFF096H	DMAディスティネーション・アドレス・レジスタ2H	DDA2H				○	
FFFFF098H	DMAソース・アドレス・レジスタ3L	DSA3L				○	
FFFFF09AH	DMAソース・アドレス・レジスタ3H	DSA3H				○	
FFFFF09CH	DMAディスティネーション・アドレス・レジスタ3L	DDA3L				○	

注　V850ES/SA3のみ。

(2/8)

アドレス	機能レジスタ名称	略号	R/W	操作可能ビット			初期値
				1	8	16	
FFFFF09EH	DMAディスティネーション・アドレス・レジスタ3H	DDA3H	R/W			○	不定
FFFFF0C0H	DMA転送カウント・レジスタ0	DBC0				○	
FFFFF0C2H	DMA転送カウント・レジスタ1	DBC1				○	
FFFFF0C4H	DMA転送カウント・レジスタ2	DBC2				○	
FFFFF0C6H	DMA転送カウント・レジスタ3	DBC3				○	
FFFFF0D0H	DMAアドレシング・コントロール・レジスタ0	DADC0				○	0000H
FFFFF0D2H	DMAアドレシング・コントロール・レジスタ1	DADC1				○	
FFFFF0D4H	DMAアドレシング・コントロール・レジスタ2	DADC2				○	
FFFFF0D6H	DMAアドレシング・コントロール・レジスタ3	DADC3				○	
FFFFF0E0H	DMAチャネル・コントロール・レジスタ0	DCHC0		○	○		00H
FFFFF0E2H	DMAチャネル・コントロール・レジスタ1	DCHC1		○	○		
FFFFF0E4H	DMAチャネル・コントロール・レジスタ2	DCHC2		○	○		
FFFFF0E6H	DMAチャネル・コントロール・レジスタ3	DCHC3		○	○		
FFFFF100H	割り込みマスク・レジスタ0	IMR0				○	FFFFH
FFFFF100H	割り込みマスク・レジスタ0L	IMR0L		○	○		FFH
FFFFF101H	割り込みマスク・レジスタ0H	IMR0H		○	○		
FFFFF102H	割り込みマスク・レジスタ1	IMR1				○	FFFFH
FFFFF102H	割り込みマスク・レジスタ1L	IMR1L		○	○		FFH
FFFFF103H	割り込みマスク・レジスタ1H	IMR1H		○	○		
FFFFF104H	割り込みマスク・レジスタ2	IMR2				○	FFFFH
FFFFF104H	割り込みマスク・レジスタ2L	IMR2L		○	○		FFH
FFFFF110H	割り込み制御レジスタ	WDTIC		○	○		47H
FFFFF112H	割り込み制御レジスタ	PIC0		○	○		
FFFFF114H	割り込み制御レジスタ	PIC1		○	○		
FFFFF116H	割り込み制御レジスタ	PIC2		○	○		
FFFFF118H	割り込み制御レジスタ	PIC3		○	○		
FFFFF11AH	割り込み制御レジスタ	PIC4		○	○		
FFFFF11CH	割り込み制御レジスタ	PIC5		○	○		
FFFFF11EH	割り込み制御レジスタ	PIC6		○	○		
FFFFF120H	割り込み制御レジスタ	RTCIC		○	○		
FFFFF122H	割り込み制御レジスタ	CCIC00		○	○		
FFFFF124H	割り込み制御レジスタ	CCIC01		○	○		
FFFFF126H	割り込み制御レジスタ	OVFIC0		○	○		
FFFFF128H	割り込み制御レジスタ	CCIC10		○	○		
FFFFF12AH	割り込み制御レジスタ	CCIC11		○	○		
FFFFF12CH	割り込み制御レジスタ	OVFIC1		○	○		
FFFFF12EH	割り込み制御レジスタ	TMIC2		○	○		
FFFFF130H	割り込み制御レジスタ	TMIC3		○	○		
FFFFF132H	割り込み制御レジスタ	TMIC4		○	○		
FFFFF134H	割り込み制御レジスタ	TMIC5		○	○		
FFFFF136H	割り込み制御レジスタ	CSIIC0		○	○		

(3/8)

アドレス	機能レジスタ名称	略号	R/W	操作可能ビット			初期値
				1	8	16	
FFFFF138H	割り込み制御レジスタ[注1]	IICIC	R/W	○	○		47H
FFFFF13AH	割り込み制御レジスタ	CSIIC1		○	○		
FFFFF13CH	割り込み制御レジスタ	SREIC0		○	○		
FFFFF13EH	割り込み制御レジスタ	SRIC0		○	○		
FFFFF140H	割り込み制御レジスタ	STIC0		○	○		
FFFFF142H	割り込み制御レジスタ	CSIIC2		○	○		
FFFFF144H	割り込み制御レジスタ	SREIC1		○	○		
FFFFF146H	割り込み制御レジスタ	SRIC1		○	○		
FFFFF148H	割り込み制御レジスタ	STIC1		○	○		
FFFFF14AH	割り込み制御レジスタ	CSIIC3		○	○		
FFFFF14CH	割り込み制御レジスタ[注2]	CSIIC4		○	○		
FFFFF14EH	割り込み制御レジスタ	ADIC		○	○		
FFFFF150H	割り込み制御レジスタ	DMAIC0		○	○		
FFFFF152H	割り込み制御レジスタ	DMAIC1		○	○		
FFFFF154H	割り込み制御レジスタ	DMAIC2		○	○		
FFFFF156H	割り込み制御レジスタ	DMAIC3		○	○		
FFFFF158H	割り込み制御レジスタ	ROVIC		○	○		
FFFFF15AH	割り込み制御レジスタ	BRGIC		○	○		
FFFFF1FAH	インサービス・プライオリティ・レジスタ	ISPR	R	○	○		00H
FFFFF1FCH	コマンド・レジスタ	PRCMD	W		○		不定
FFFFF1FEH	パワー・セーブ・コントロール・レジスタ	PSC	R/W	○	○		00H
FFFFF200H	A/Dコンバータ・モード・レジスタ	ADM		○	○		
FFFFF201H	アナログ入力チャネル指定レジスタ	ADS			○		
FFFFF202H	パワー・フェイル比較モード・レジスタ	PFM		○	○		
FFFFF203H	パワー・フェイル比較しきい値レジスタ	PFT			○		
FFFFF204H	A/D変換結果レジスタ	ADCR	R			○	不定
FFFFF204H	A/D変換結果レジスタL	ADCRL					
FFFFF205H	A/D変換結果レジスタH	ADCRH			○		
FFFFF280H	D/Aコンバータ変換値設定レジスタ0	DACS0	R/W		○		00H
FFFFF282H	D/Aコンバータ変換値設定レジスタ1	DACS1			○		
FFFFF288H	D/Aコンバータ・モード・レジスタ	DAM		○	○		
FFFFF400H	ポート・レジスタ0	P0		○	○		不定
FFFFF404H	ポート・レジスタ2[注2]	P2		○	○		
FFFFF406H	ポート・レジスタ3	P3		○	○		
FFFFF408H	ポート・レジスタ4	P4		○	○		
FFFFF40EH	ポート・レジスタ7	P7	R			○	
FFFFF40EH	ポート・レジスタ7L	P7L		○	○		
FFFFF40FH	ポート・レジスタ7H	P7H		○	○		
FFFFF410H	ポート・レジスタ8	P8		○	○		

注1. μPD703200Y, 703201Y, 703204Y, 70F3201Y, 70F3204Y**のみ。**

2. V850ES/SA3**のみ。**

(4/8)

アドレス	機能レジスタ名称	略号	R/W	操作可能ビット			初期値
				1	8	16	
FFFFF412H	ポート・レジスタ9	P9	R/W			○	不定
FFFFF412H	ポート・レジスタ9L	P9L		○	○		
FFFFF413H	ポート・レジスタ9H	P9H		○	○		
FFFFF420H	ポート・モード・レジスタ0	PM0		○	○		FFH
FFFFF424H	ポート・モード・レジスタ2[注]	PM2		○	○		
FFFFF426H	ポート・モード・レジスタ3	PM3		○	○		
FFFFF428H	ポート・モード・レジスタ4	PM4		○	○		
FFFFF432H	ポート・モード・レジスタ9	PM9				○	FFFFH
FFFFF432H	ポート・モード・レジスタ9L	PM9L		○	○		FFH
FFFFF433H	ポート・モード・レジスタ9H	PM9H		○	○		
FFFFF440H	ポート・モード・コントロール・レジスタ0	PMC0		○	○		00H
FFFFF444H	ポート・モード・コントロール・レジスタ2[注]	PMC2		○	○		
FFFFF446H	ポート・モード・コントロール・レジスタ3	PMC3		○	○		
FFFFF448H	ポート・モード・コントロール・レジスタ4	PMC4		○	○		
FFFFF452H	ポート・モード・コントロール・レジスタ9	PMC9				○	0000H
FFFFF452H	ポート・モード・コントロール・レジスタ9L	PMC9L		○	○		00H
FFFFF453H	ポート・モード・コントロール・レジスタ9H	PMC9H		○	○		
FFFFF466H	ポート・ファンクション・コントロール・レジスタ3	PFC3		○	○		
FFFFF468H	ポート・ファンクション・コントロール・レジスタ4	PFC4		○	○		
FFFFF472H	ポート・ファンクション・コントロール・レジスタ9	PFC9				○	0000H
FFFFF472H	ポート・ファンクション・コントロール・レジスタ9L	PFC9L		○	○		00H
FFFFF473H	ポート・ファンクション・コントロール・レジスタ9H	PFC9H		○	○		
FFFFF484H	データ・ウエイト制御レジスタ0	DWC0				○	7777H
FFFFF488H	アドレス・ウエイト制御レジスタ	AWC				○	FFFFH
FFFFF48AH	バス・サイクル制御レジスタ	BCC				○	AAAAH
FFFFF600H	タイマ0	TM0	R			○	0000H
FFFFF602H	キャプチャ／コンペア・レジスタ00	CC00	R/W			○	
FFFFF604H	キャプチャ／コンペア・レジスタ01	CC01				○	
FFFFF606H	タイマ・コントロール・レジスタ00	TMC00		○	○		00H
FFFFF608H	タイマ・コントロール・レジスタ01	TMC01		○	○		20H
FFFFF609H	有効エッジ選択レジスタ0	SES0			○		00H
FFFFF610H	タイマ1	TM1	R			○	0000H
FFFFF612H	キャプチャ／コンペア・レジスタ10	CC10	R/W			○	
FFFFF614H	キャプチャ／コンペア・レジスタ11	CC11				○	
FFFFF616H	タイマ・コントロール・レジスタ10	TMC10		○	○		00H
FFFFF618H	タイマ・コントロール・レジスタ11	TMC11		○	○		20H
FFFFF619H	有効エッジ選択レジスタ1	SES1			○		00H

注　V850ES/SA3のみ。

(5/8)

アドレス	機能レジスタ名称	略号	R/W	操作可能ビット			初期値
				1	8	16	
FFFFF640H	タイマ・カウンタ23	TM23	R			○	0000H
FFFFF640H	タイマ・カウンタ2	TM2			○		00H
FFFFF641H	タイマ・カウンタ3	TM3			○		
FFFFF642H	コンペア・レジスタ23	CR23	R/W			○	0000H
FFFFF642H	コンペア・レジスタ2	CR2			○		00H
FFFFF643H	コンペア・レジスタ3	CR3			○		
FFFFF644H	タイマ・クロック選択レジスタ23	TCL23				○	0000H
FFFFF644H	タイマ・クロック選択レジスタ2	TCL2			○		00H
FFFFF645H	タイマ・クロック選択レジスタ3	TCL3			○		
FFFFF646H	タイマ・モード・コントロール・レジスタ23	TMC23				○	0000H
FFFFF646H	タイマ・モード・コントロール・レジスタ2	TMC2		○	○		00H
FFFFF647H	タイマ・モード・コントロール・レジスタ3	TMC3		○	○		
FFFFF650H	タイマ・カウンタ45	TM45	R			○	0000H
FFFFF650H	タイマ・カウンタ4	TM4			○		00H
FFFFF651H	タイマ・カウンタ5	TM5			○		
FFFFF652H	コンペア・レジスタ45	CR45	R/W			○	0000H
FFFFF652H	コンペア・レジスタ4	CR4			○		00H
FFFFF653H	コンペア・レジスタ5	CR5			○		
FFFFF654H	タイマ・クロック選択レジスタ45	TCL45				○	0000H
FFFFF654H	タイマ・クロック選択レジスタ4	TCL4			○		00H
FFFFF655H	タイマ・クロック選択レジスタ5	TCL5			○		
FFFFF656H	タイマ・モード・コントロール・レジスタ45	TMC45				○	0000H
FFFFF656H	タイマ・モード・コントロール・レジスタ4	TMC4		○	○		00H
FFFFF657H	タイマ・モード・コントロール・レジスタ5	TMC5		○	○		
FFFFF6C0H	発振安定時間選択レジスタ	OSTS			○		04H
FFFFF6C1H	ウォッチドッグ・タイマ・時間選択レジスタ	WDCS			○		00H
FFFFF6C2H	ウォッチドッグ・タイマ・モード・レジスタ	WDTM		○	○		
FFFFF6E0H	RTC動作制御レジスタ	RTCC				○	808XH
FFFFF6E0H	RTC動作制御レジスタ0	RTCC0		○	○		80H
FFFFF6E1H	RTC動作制御レジスタ1	RTCC1		○	○		8XH
FFFFF6E2H	サブカウント・レジスタ	SUBC	R			○	XXXXH
FFFFF6E2H	サブカウント・レジスタL	SUBCL			○		XXH
FFFFF6E3H	サブカウント・レジスタH	SUBCH			○		
FFFFF6E4H	分秒カウント・レジスタ	SECMIN				○	XXXXH
FFFFF6E4H	秒カウント・レジスタ	SEC			○		XXH
FFFFF6E5H	分カウント・レジスタ	MIN			○		
FFFFF6E6H	日時カウント・レジスタ	HOURDAY				○	0XXXH
FFFFF6E6H	時カウント・レジスタ	HOUR			○		XXH
FFFFF6E7H	日カウント・レジスタ	DAY			○		0XH
FFFFF6E8H	週カウント・レジスタ	WEEK				○	0XXXH
FFFFF6E8H	週カウント・レジスタL	WEEKL			○		XXH
FFFFF6E9H	週カウント・レジスタH	WEEKH			○		0XH

(6/8)

アドレス	機能レジスタ名称	略号	R/W	操作可能ビット				初期値
				1	8	16	32	
FFFFF6EAH	分秒カウント設定レジスタ	SECMINB	W			○		0000H
FFFFF6EAH	秒カウント設定レジスタ	SECB			○			00H
FFFFF6EBH	分カウント設定レジスタ	MINB			○			
FFFFF6ECH	日時カウント設定レジスタ	HOURDAYB				○		0000H
FFFFF6ECH	時カウント設定レジスタ	HOURB			○			00H
FFFFF6EDH	日カウント設定レジスタ	DAYB			○			
FFFFF6EEH	週カウント設定レジスタ	WEEKB				○		0000H
FFFFF6EEH	週カウント設定レジスタL	WEEKBL			○			00H
FFFFF6EFH	週カウント設定レジスタH	WEEKBH			○			
FFFFF802H	システム・ステータス・レジスタ	SYS	R/W	○	○			
FFFFF810H	DMAトリガ要因レジスタ0	DTFR0		○	○			00H
FFFFF812H	DMAトリガ要因レジスタ1	DTFR1		○	○			
FFFFF814H	DMAトリガ要因レジスタ2	DTFR2		○	○			
FFFFF816H	DMAトリガ要因レジスタ3	DTFR3		○	○			
FFFFF820H	パワー・セーブ・モード・レジスタ	PSMR		○	○			
FFFFF828H	プロセッサ・クロック・コントロール・レジスタ	PCC		○	○			03H
FFFFF840H	コレクション・アドレス・レジスタ0	CORAD0					○	00000000H
FFFFF840H	コレクション・アドレス・レジスタ0L	CORAD0L				○		0000H
FFFFF842H	コレクション・アドレス・レジスタ0H	CORAD0H				○		
FFFFF844H	コレクション・アドレス・レジスタ1	CORAD1					○	00000000H
FFFFF844H	コレクション・アドレス・レジスタ1L	CORAD1L				○		0000H
FFFFF846H	コレクション・アドレス・レジスタ1H	CORAD1H				○		
FFFFF848H	コレクション・アドレス・レジスタ2	CORAD2					○	00000000H
FFFFF848H	コレクション・アドレス・レジスタ2L	CORAD2L				○		0000H
FFFFF84AH	コレクション・アドレス・レジスタ2H	CORAD2H				○		
FFFFF84CH	コレクション・アドレス・レジスタ3	CORAD3					○	00000000H
FFFFF84CH	コレクション・アドレス・レジスタ3L	CORAD3L				○		0000H
FFFFF84EH	コレクション・アドレス・レジスタ3H	CORAD3H				○		
FFFFF880H	コレクション・コントロール・レジスタ	CORCN		○	○			00H
FFFFF8B0H	プリスケーラ・モード・レジスタ	PRSM			○			
FFFFF8B1H	プリスケーラ・コンペア・レジスタ	PRSCM			○			
FFFFFA00H	UART0動作モード・レジスタ	ASIM0		○	○			01H
FFFFFA02H	受信バッファ・レジスタ0	RXB0	R		○			FFH
FFFFFA03H	UART0受信エラー・ステータス・レジスタ	ASIS0			○			00H
FFFFFA04H	送信バッファ・レジスタ0	TXB0	R/W		○			FFH
FFFFFA05H	UART0送信ステータス・レジスタ	ASIF0	R	○	○			00H
FFFFFA06H	クロック選択レジスタ0	CKSR0	R/W		○			
FFFFFA07H	ボー・レート・ジェネレータ・コンペア・レジスタ0	BRGC0			○			FFH
FFFFFA10H	UART1動作モード・レジスタ	ASIM1		○	○			01H
FFFFFA12H	受信バッファ・レジスタ1	RXB1	R		○			FFH
FFFFFA13H	UART1受信エラー・ステータス・レジスタ	ASIS1			○			00H

(7/8)

アドレス	機能レジスタ名称	略号	R/W	操作可能ビット			初期値
				1	8	16	
FFFFFA14H	送信バッファ・レジスタ1	TXB1	R/W		○		FFH
FFFFFA15H	UART1送信ステータス・レジスタ	ASIF1	R	○	○		00H
FFFFFA16H	クロック選択レジスタ1	CKSR1	R/W		○		
FFFFFA17H	ボー・レート・ジェネレータ・コンペア・レジスタ1	BRGC1			○		FFH
FFFFFC00H	外部割り込み立ち下がりエッジ指定レジスタ0	INTF0		○	○		00H
FFFFFC12H	外部割り込み立ち下がりエッジ指定レジスタ9	INTF9				○	0000H
FFFFFC12H	外部割り込み立ち下がりエッジ指定レジスタ9L	INTF9L		○	○		00H
FFFFFC20H	外部割り込み立ち上がりエッジ指定レジスタ0	INTR0		○	○		
FFFFFC32H	外部割り込み立ち上がりエッジ指定レジスタ9	INTR9				○	0000H
FFFFFC32H	外部割り込み立ち上がりエッジ指定レジスタ9L	INTR9L		○	○		00H
FFFFFC40H	プルアップ抵抗オプション・レジスタ0	PU0		○	○		
FFFFFC44H	プルアップ抵抗オプション・レジスタ2[注]	PU2		○	○		
FFFFFC46H	プルアップ抵抗オプション・レジスタ3	PU3		○	○		
FFFFFC48H	プルアップ抵抗オプション・レジスタ4	PU4		○	○		
FFFFFC52H	プルアップ抵抗オプション・レジスタ9	PU9				○	0000H
FFFFFC52H	プルアップ抵抗オプション・レジスタ9L	PU9L		○	○		00H
FFFFFC53H	プルアップ抵抗オプション・レジスタ9H	PU9H		○	○		
FFFFFC64H	ポート・ファンクション・レジスタ2[注]	PF2		○	○		
FFFFFC66H	ポート・ファンクション・レジスタ3	PF3		○	○		
FFFFFC68H	ポート・ファンクション・レジスタ4	PF4		○	○		
FFFFFC72H	ポート・ファンクション・レジスタ9	PF9				○	0000H
FFFFFC73H	ポート・ファンクション・レジスタ9H	PF9H		○	○		00H
FFFFFD00H	クロック同期式シリアル・インタフェース・モード・レジスタ0	CSIM0		○	○		
FFFFFD01H	クロック同期式シリアル・インタフェース・クロック選択レジスタ0	CSIC0			○		
FFFFFD02H	シリアルI/Oシフト・レジスタ0	SIO0	R		○		
FFFFFD03H	受信専用シリアルI/Oシフト・レジスタ0	SIOE0			○		
FFFFFD04H	クロック同期式シリアル・インタフェース送信バッファ・レジスタ0	SOTB0	R/W		○		
FFFFFD10H	クロック同期式シリアル・インタフェース・モード・レジスタ1	CSIM1		○	○		
FFFFFD11H	クロック同期式シリアル・インタフェース・クロック選択レジスタ1	CSIC1			○		
FFFFFD12H	シリアルI/Oシフト・レジスタ1	SIO1	R		○		
FFFFFD13H	受信専用シリアルI/Oシフト・レジスタ1	SIOE1			○		
FFFFFD14H	クロック同期式シリアル・インタフェース送信バッファ・レジスタ1	SOTB1	R/W		○		
FFFFFD20H	クロック同期式シリアル・インタフェース・モード・レジスタ2	CSIM2		○	○		
FFFFFD21H	クロック同期式シリアル・インタフェース・クロック選択レジスタ2	CSIC2			○		
FFFFFD22H	シリアルI/Oシフト・レジスタ2	SIO2	R		○		
FFFFFD23H	受信専用シリアルI/Oシフト・レジスタ2	SIOE2			○		
FFFFFD24H	クロック同期式シリアル・インタフェース送信バッファ・レジスタ2	SOTB2	R/W		○		
FFFFFD30H	クロック同期式シリアル・インタフェース・モード・レジスタ3	CSIM3		○	○		
FFFFFD31H	クロック同期式シリアル・インタフェース・クロック選択レジスタ3	CSIC3			○		
FFFFFD32H	シリアルI/Oシフト・レジスタ3	SIO3	R		○		
FFFFFD33H	受信専用シリアルI/Oシフト・レジスタ3	SIOE3			○		

注　V850ES/SA3のみ。

(8/8)

アドレス	機能レジスタ名称	略号	R/W	操作可能ビット			初期値
				1	8	16	
FFFFFD34H	クロック同期式シリアル・インタフェース送信バッファ・レジスタ3	SOTB3	R/W		○		00H
FFFFFD40H	クロック同期式シリアル・インタフェース・モード・レジスタ4注1	CSIM4		○	○		
FFFFFD41H	クロック同期式シリアル・インタフェース・クロック選択レジスタ4注1	CSIC4			○		
FFFFFD42H	シリアルI/Oシフト・レジスタ4注1	SIO4	R		○		
FFFFFD43H	受信専用シリアルI/Oシフト・レジスタ4注1	SIOE4			○		
FFFFFD44H	クロック同期式シリアル・インタフェース送信バッファ・レジスタ4注1	SOTB4	R/W		○		
FFFFFD80H	IICシフト・レジスタ注2	IIC			○		
FFFFFD82H	IICコントロール・レジスタ注2	IICC		○	○		
FFFFFD83H	スレーブ・アドレス・レジスタ注2	SVA			○		
FFFFFD84H	IICクロック選択レジスタ注2	IICCL		○	○		
FFFFFD85H	IIC機能拡張レジスタ注2	IICX		○	○		
FFFFFD86H	IIC状態レジスタ注2	IICS	R	○	○		
FFFFF4BEH	外部バス・インタフェース・モード・コントロール・レジスタ	EXIMC	R/W	○	○		

注1. V850ES/SA3**のみ。**

2. μPD703200Y, 703201Y, 703204Y, 70F3201Y, 70F3204Y**のみ。**

1. 12　スタンバイ機能

スタンバイ機能には，内部システム・クロックをメイン・クロックで動作させる通常動作状態でHALT, IDLE，ソフトウエアSTOPの3モードがあります。そして，内部システム・クロックをサブクロックで動作させる状態でサブクロック動作，サブIDLEの2モードがあります。

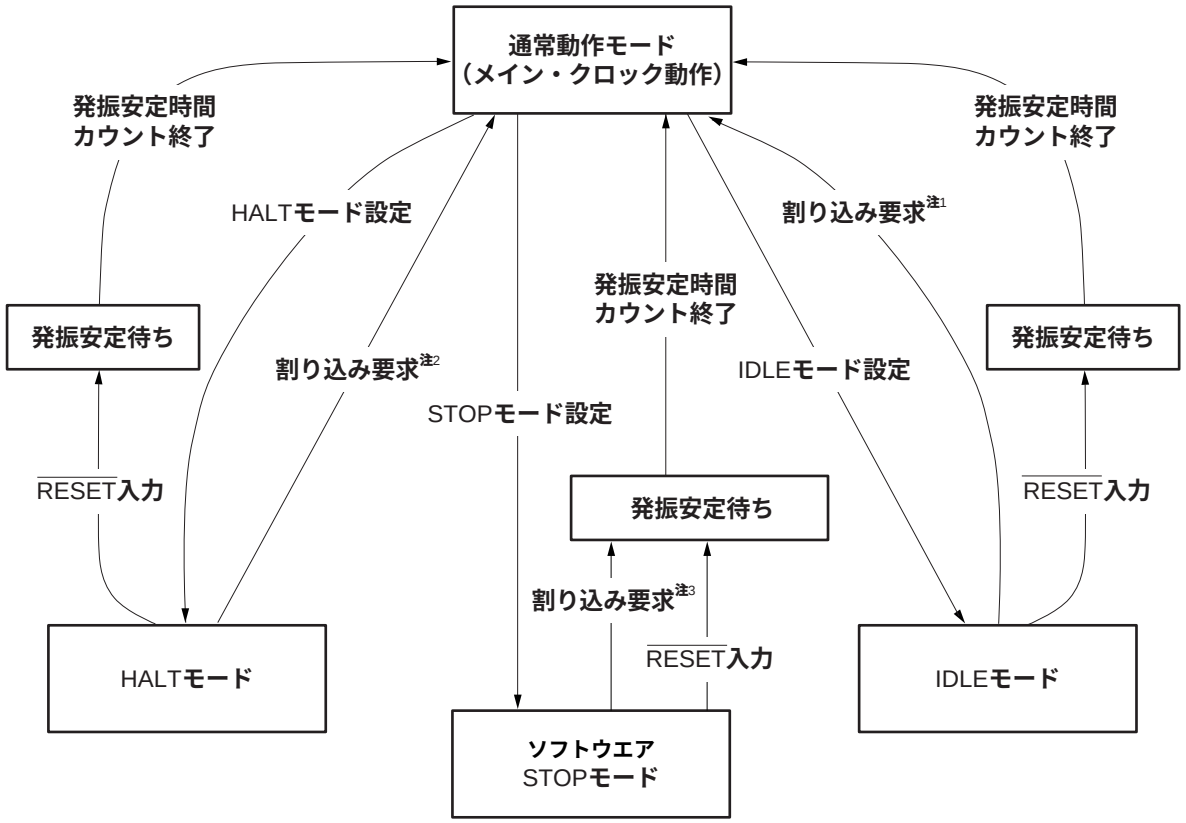
これらのモードの組み合わせ，用途によって切り替えて使用することで，効果的な低消費電力システムを実現できます。

次に各モードの機能概要と通常動作，サブクロック動作における状態遷移図を示します。

表1－4　スタンバイ機能のモード一覧

モード	機能概要
HALTモード	CPUの動作クロックのみを停止させるモード
IDLEモード	発振回路以外のチップ内部の動作をすべて停止させるモード
ソフトウエアSTOPモード	サブクロック発振回路以外のチップ内部の動作をすべて停止させるモード
サブクロック動作モード	内部システム・クロックをサブクロックで動作させるモード
サブIDLEモード	サブクロック動作モード時，発振回路以外のチップ内部の動作をすべて停止させるモード

図1－3　状態遷移図

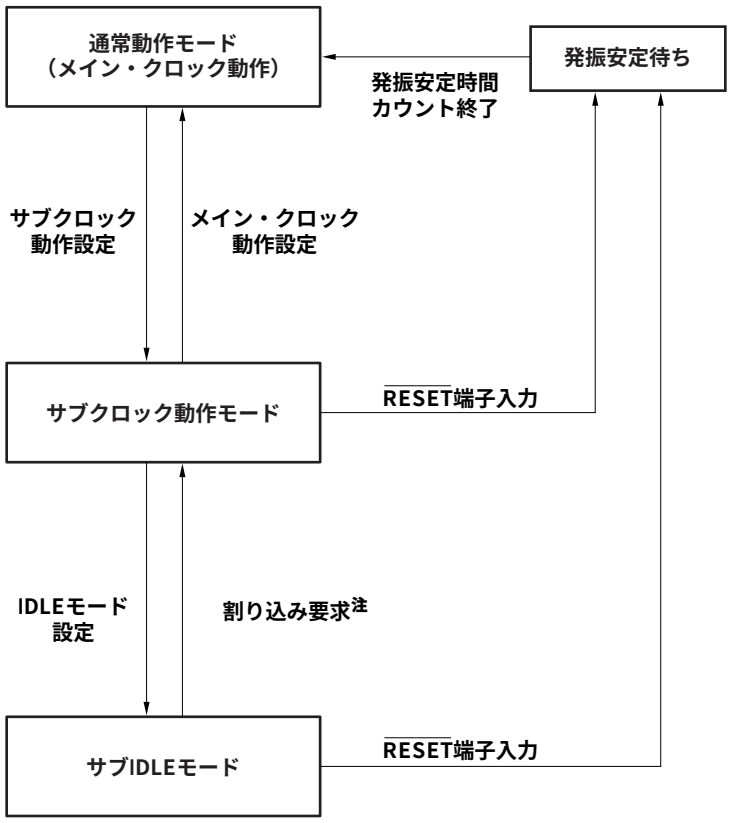


注1. ノンマスカブル割り込み要求（NMI端子入力），マスクされていない外部割り込み要求（INTP0-INTP6端子入力），IDLEモード中に動作可能な周辺機能のマスクされていない内部割り込み要求

2. ノンマスカブル割り込み要求，マスクされていないマスカブル割り込み要求，WDTオーバフローによるリセット入力

3. ノンマスカブル割り込み要求（NMI端子入力），マスクされていない外部割り込み要求（INTP0-INTP6端子入力），ソフトウエアSTOPモード中に動作可能な周辺機能のマスクされていない内部割り込み要求

図1－4　状態遷移図（サブクロック動作時）



注　ノンマスカブル割り込み要求（NMI端子入力），マスクされていない外部割り込み要求（INTP0-INTP6端子入力），サブIDLEモード中動作可能な周辺機能のマスクされていない内部割り込み要求

第2章　バス・インタフェース接続回路例

V850ES/SA2, V850ES/SA3の外部バス・インタフェース機能を使用した，外部デバイスとの接続について説明します。

- 接続例におけるタイミング生成部，デコーダ部は，一般的にはゲートアレイやPLDなどを使用しますが，ここでは論理的説明に重点をおきAND/ORフリップ・フロップなどを使用しています。したがって，実際の回路では部品の遅延時間，DC特性を考慮して設計してください。
- A0-A15は，V850ES/SA2, V850ES/SA3のAD0-AD15から分離生成（マルチプレクス・バス・モード）したアドレス信号，またはセパレート・モードでのA0-A15のどちらかを使用していることとします（2. 1. 1　**アドレス／データ・バスの分離**を参照）。
- 接続例においては，3.3 V系デバイスと接続する場合，DC特性を満足できるものでもレベル変換デバイスを使用しています。実際の回路では，V850ES/SA2, V850ES/SA3と対象デバイスの双方のDC特性が満足できれば，レベル変換デバイスを省略することができます。
- 接続例における信号遅延計算では，V850ES/SA2, V850ES/SA3とメモリ部品のみを計算対象として，ウエイト，アイドルの挿入設定をしています。レベル変換部品やプリント基板配線の遅延時間は含んでいません。実際の回路では，これらの遅延を考慮したうえで設定を行ってください。

2. 1　外部バス・インタフェース

2. 1. 1　アドレス／データ・バスの分離

V850ES/SA2は最大22ビット，V850ES/SA3は最大24ビットのアドレス信号が使用できます。このうち下位16ビットは，データ・バスとマルチプレクスされているAD0-AD15を使用する方法と，データ・バスとセパレートされたアドレス・バス（A0-A15）を使用する方法があります。前者をマルチプレクス・バス・モード，後者をセパレート・バス・モードといいます。

2つのモードの切り替えは，EXIMCレジスタで設定します。

リセット時：00H　R/W　**アドレス：**FFFFFFBEH

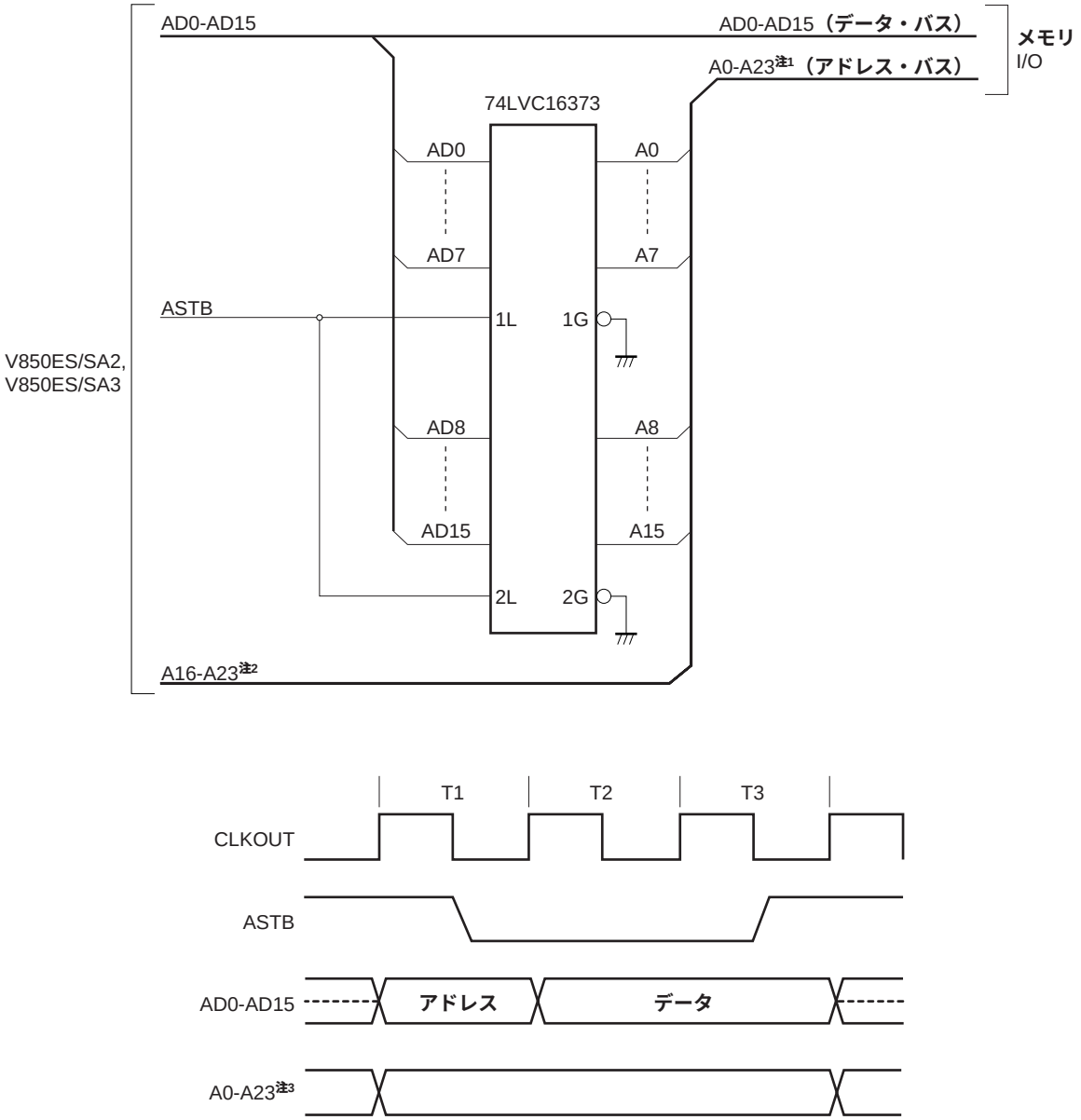
	7	6	5	4	3	2	1	0
EXIMC	0	0	0	0	0	0	0	SMSEL

SMSEL	モード切り替え
0	マルチプレクス・バス・モード
1	セパレート・バス・モード

(1) マルチプレクス・バス・モード

アドレスとデータを外部回路により分離する必要があります。アドレス信号は透過ラッチ機能を持つデバイスを使って，ASTB信号でラッチし作成するのが一般的な方法です。データ・バスはAD0-AD15をそのまま使用します。例を図2－1に示します。

図2－1　アドレス・バス／データ・バスの分離回路例



注1. A0-A15は，ASTBにより分離された信号です。

2. V850ES/SA2は，A0-A21までの信号です。

3. V850ES/SA2は，A16-A21までの信号です。

備考1. タイミング・チャートは0ウエイト時の状態です。

2. 破線はハイ・インピーダンスを示します。

(2) セパレート・バス・モード

アドレス・バス（A0-A15）に対応する端子がほかの機能と兼用されているので，次のレジスタの設定を行います。

(a) ポート・ファンクション・コントロール・レジスタ9（PFC9）の設定

セパレート・アドレス・バス出力を行うか，シリアルI/O，タイマ入出力を行うかを設定します。

PFC9レジスタの全ビットに“0”を設定して，セパレート・アドレス・バス出力の設定をします。

リセット時：0000H　R/W　**アドレス：**FFFFF472H, FFFFF473H

	15	14	13	12	11	10	9	8
PFC9	PFC910	PFC910	PFC910	PFC910	PFC910	PFC910	PFC99	PFC98

7	6	5	4	3	2	1	0
PFC97	PFC96	PFC95	PFC94	PFC93	PFC92	0	0

PFC9 = 0000000000000000 B

(b) ポート・モード・コントロール・レジスタ9（PMC9）の設定

入出力モードにするか，セパレート・アドレス・バス出力，シリアルI/O，タイマ入出力モードにするかを設定します。

PMC9レジスタの全ビットに“1”を設定して，セパレート・アドレス・バス出力の設定をします。

リセット時：0000H　R/W　**アドレス：**FFFFF452H, FFFFF453H

	15	14	13	12	11	10	9	8
PMC9	PMC915	PMC914	PMC913	PMC912	PMC911	PMC910	PMC99	PMC98

7	6	5	4	3	2	1	0
PMC97	PMC96	PMC95	PMC94	PMC93	PMC92	PMC91	PMC90

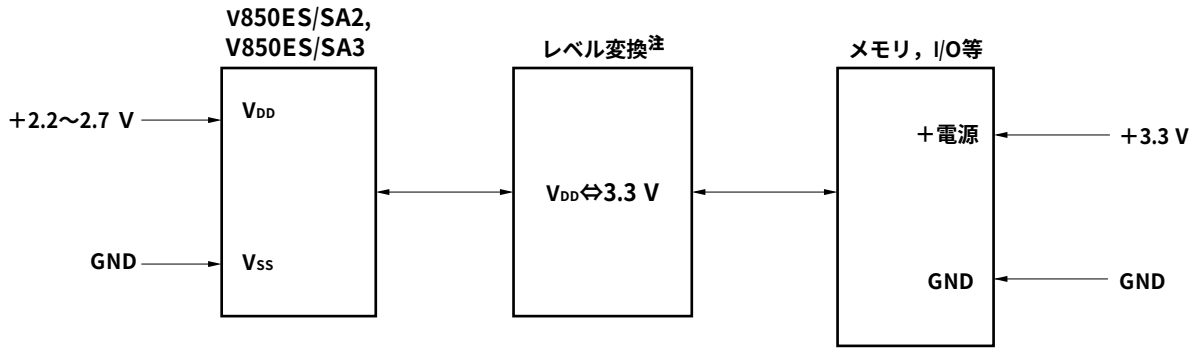
PMC9 = 1111111111111111B

注意　PFC9レジスタ設定後にPMC9レジスタに，16ビット一括で設定してください。

2. 1. 2　3.3 V系デバイスの接続

V850ES/SA2, V850ES/SA3と3.3Vシステム（デバイス）を接続する場合は，それぞれのDC特性を守るためにレベル変換デバイスを挿入します。

図2－2　3.3 V系デバイスの接続



注　レベル変換用として使用できるデバイスは，LCXシリーズ，LVXシリーズ，FCTシリーズなどがあります。ICメーカから各種デバイスが販売されています。

2. 1. 3　バス・サイズの設定と接続例

外部メモリ領域は，8ビットまたは16ビットのバス・サイズを，$\overline{\text{CSn}}$で選択される領域ごとにBSCレジスタで設定します。

リセット時：5555H　R/W　アドレス：FFFFF066H

	15	14	13	12	11	10	9	8
BSC	0	1	0	1	0	1	0	1

	7	6	5	4	3	2	1	0
	0	BS30	0	BS20	0	BS10	0	BS00
$\overline{\text{CSn}}$信号		$\overline{\text{CS3}}$		$\overline{\text{CS2}}$		$\overline{\text{CS1}}$		$\overline{\text{CS0}}$

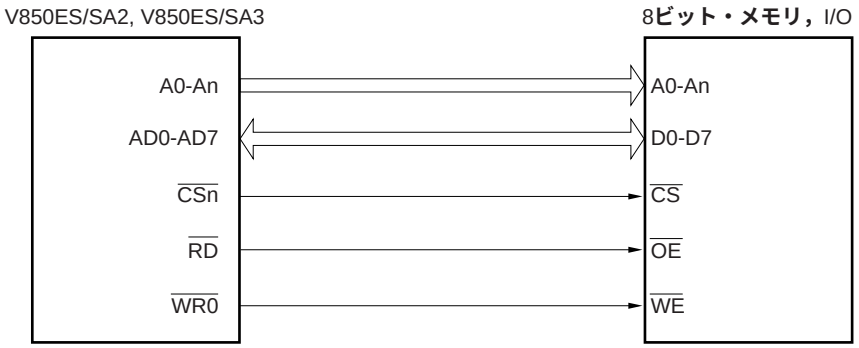
BSn0	CSn空間のデータ・バス幅（n = 0-3）
0	8ビット
1	16ビット

注意　ビット14, 12, 10, 8**には必ず“**1**”を設定し，ビット**15, 13, 11, 9, 7, 5, 3, 1**には必ず“**0**”を設定してください**

（1）8ビット・バス・サイズの接続

8ビット・メモリ，I/Oユニットのデータ信号は外部アドレス・データAD0-AD7に接続します。

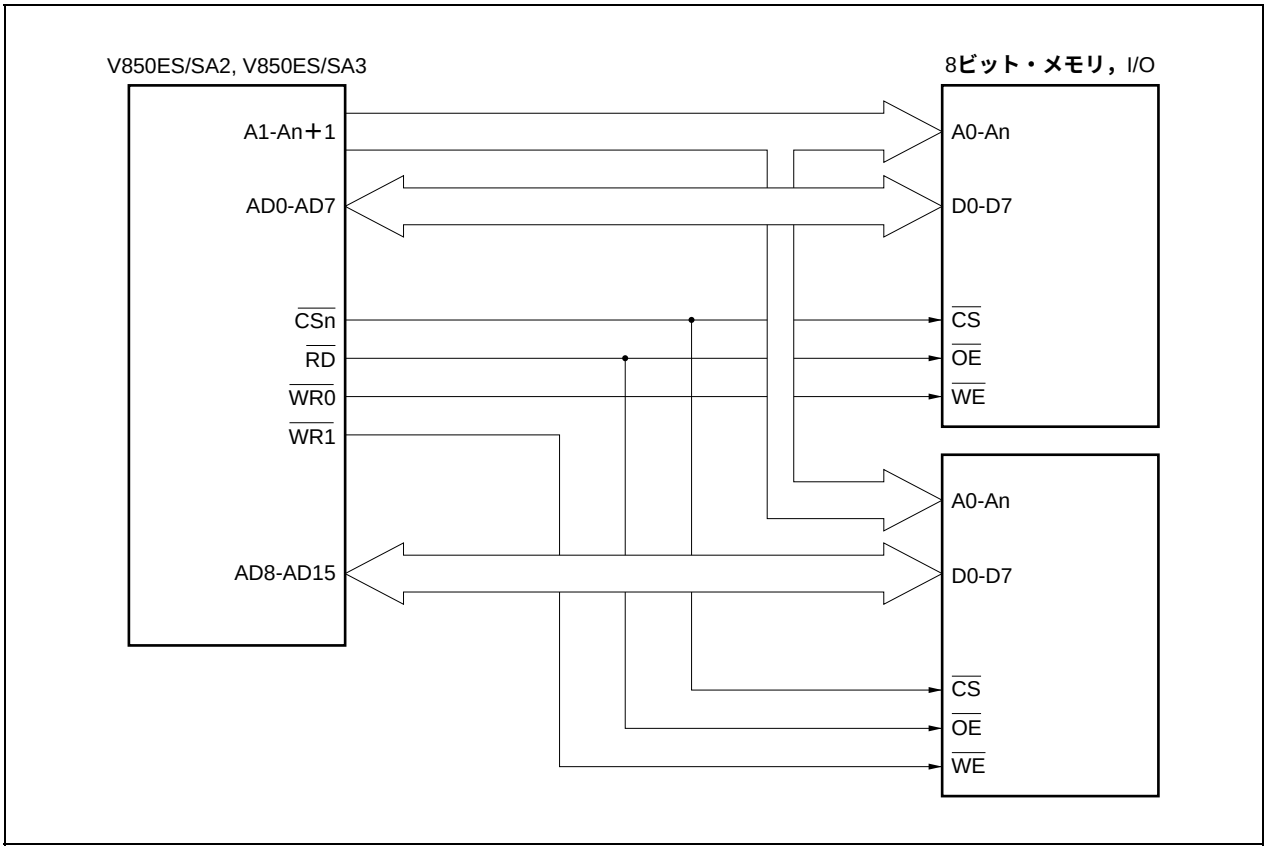
図2－3　8ビット・バス使用時の8ビット・バス幅ユニットとの接続



(2) 16ビット・バス・サイズの接続

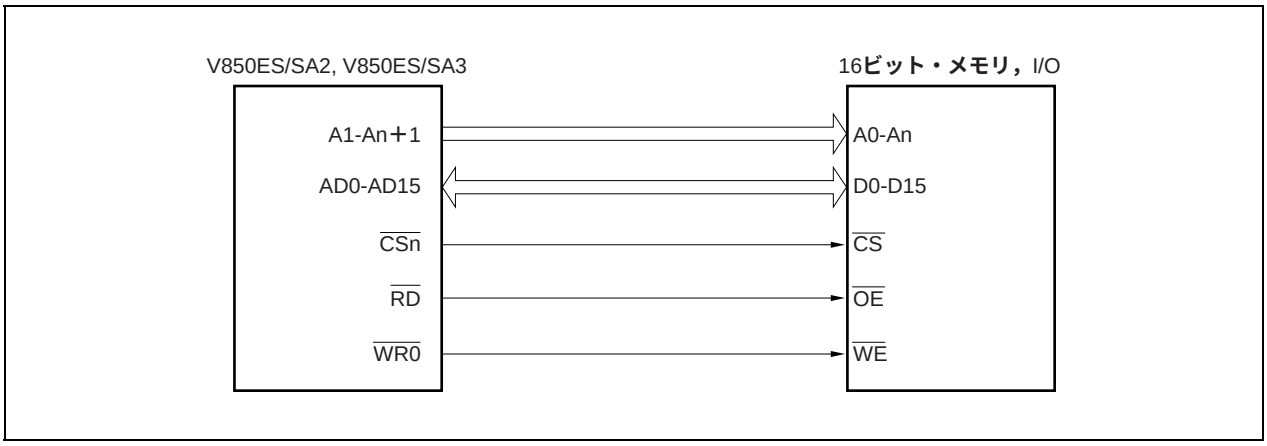
8ビット・メモリ，I/Oユニットを16ビット・バスに接続するのは次のようになります。

図2－4　16ビット・バス使用時の8ビット・バス幅ユニットとの接続



16ビット・メモリ，I/Oユニットを16ビット・バスに接続するのは次のようになります。

図2－5　16ビット・バス使用時の16ビット・バス幅ユニットとの接続



2. 2　8ビットSRAMとの接続

SRAM（μPD434008AL：512 K×8ビット）の接続例を示します。

メモリ・アドレス0x200000から配置することとし，CS信号は$\overline{CS1}$を使います。

2. 2. 1　8ビット・バス幅例

SRAM（μPD434008AL：512 K×8ビット）を1個使用して，512 Kバイトの外部メモリ空間を接続する例を示します。

【回路構成】

・システム・クロック　：20 MHz

・接続デバイス　　：μPD434008ALxx-A20（1個）

・使用CS信号　　：$\overline{CS1}$

占有メモリ空間：0x00000-0x27FFFF

【接続の考え方と注意点】

SRAMのデータ・バスはV850ES/SA2, V850ES/SA3のAD0-AD7に，アドレス・バスはV850ES/SA2, V850ES/SA3のA0-A18に接続します。$\overline{CS}$は$\overline{CS1}$に，$\overline{OE}$は$\overline{RD}$に，$\overline{WR}$は$\overline{WR0}$にそれぞれ接続します。

【レジスタの設定】

・バス・サイズの設定

BS10ビットに“0”を設定し，$\overline{CS1}$領域を8ビット・サイズに設定します。

リセット時：5555H　R/W　アドレス：FFFFF066H

	15	14	13	12	11	10	9	8
BSC	0	1	0	1	0	1	0	1
	7	6	5	4	3	2	1	0
	0	BS30	0	BS20	0	BS10	0	BS00
$\overline{CSn}$信号		$\overline{CS3}$		$\overline{CS2}$		$\overline{CS1}$		$\overline{CS0}$

BSC = 010101010x0x000xB

注意　ビット14, 12, 10, 8には必ず“1”を，ビット15, 13, 11, 9, 7, 5, 3, 1には必ず“0”を設定してください。

・データ・ウエイトの設定

DW12-DW10ビットに“000”を設定し，$\overline{CS1}$領域を0ウエイトに設定します。

リセット時：7777H　R/W　**アドレス：**FFFFF484H

	15	14	13	12	11	10	9	8
DWC0	0	DW32	DW31	DW30	0	DW22	DW21	DW20
$\overline{CSn}$信号		$\overline{CS3}$				$\overline{CS2}$		

	7	6	5	4	3	2	1	0
	0	DW12	DW11	DW10	0	DW02	DW01	DW00
$\overline{CSn}$信号		$\overline{CS1}$				$\overline{CS0}$		

DWC0 = 0xxx0xxx00000xxxB

注意　ビット15, 11, 7, 3**には必ず“**0**”を設定してください。**

・アドレス・ウエイトの設定

AHW1, ASW1ビットに“0”を設定し，$\overline{CS1}$領域にアドレス・ホールド・ウエイト，アドレス・セットアップ・ウエイトの双方を“挿入しない”に設定します。

リセット時：FFFFH　R/W　**アドレス：**FFFFF488H

	15	14	13	12	11	10	9	8
AWC	1	1	1	1	1	1	1	1

	7	6	5	4	3	2	1	0
	AHW3	ASW3	AHW2	ASW2	AHW1	ASW1	AHW0	ASW0
$\overline{CSn}$信号		$\overline{CS3}$		$\overline{CS2}$		$\overline{CS1}$		$\overline{CS0}$

AWC = 11111111xxxx00xxB

注意　ビット15-8**には必ず“**1**”を設定してください。**

・アイドル・ステートの設定

BC11ビットに“0”を設定し，$\overline{CS1}$領域にアイドル・ステートを“挿入しない”に設定します。

リセット時：AAAAH　R/W　アドレス：FFFFF48AH

	15	14	13	12	11	10	9	8
BCC	1	0	1	0	1	0	1	0

	7	6	5	4	3	2	1	0
	BC31	0	BC21	0	BC11	0	BC01	0
$\overline{CSn}$信号	$\overline{CS3}$		$\overline{CS2}$		$\overline{CS1}$		$\overline{CS0}$	

BCC = 10101010x0x000x0B

注意　ビット14, 12, 10, 8, 6, 4, 2, 0**には必ず“**0**”を，ビット**15, 13, 11, 9**には必ず“**1**”を設定してください。**

図2－6　μPD434008AL**の**8**ビット・バス幅接続例**

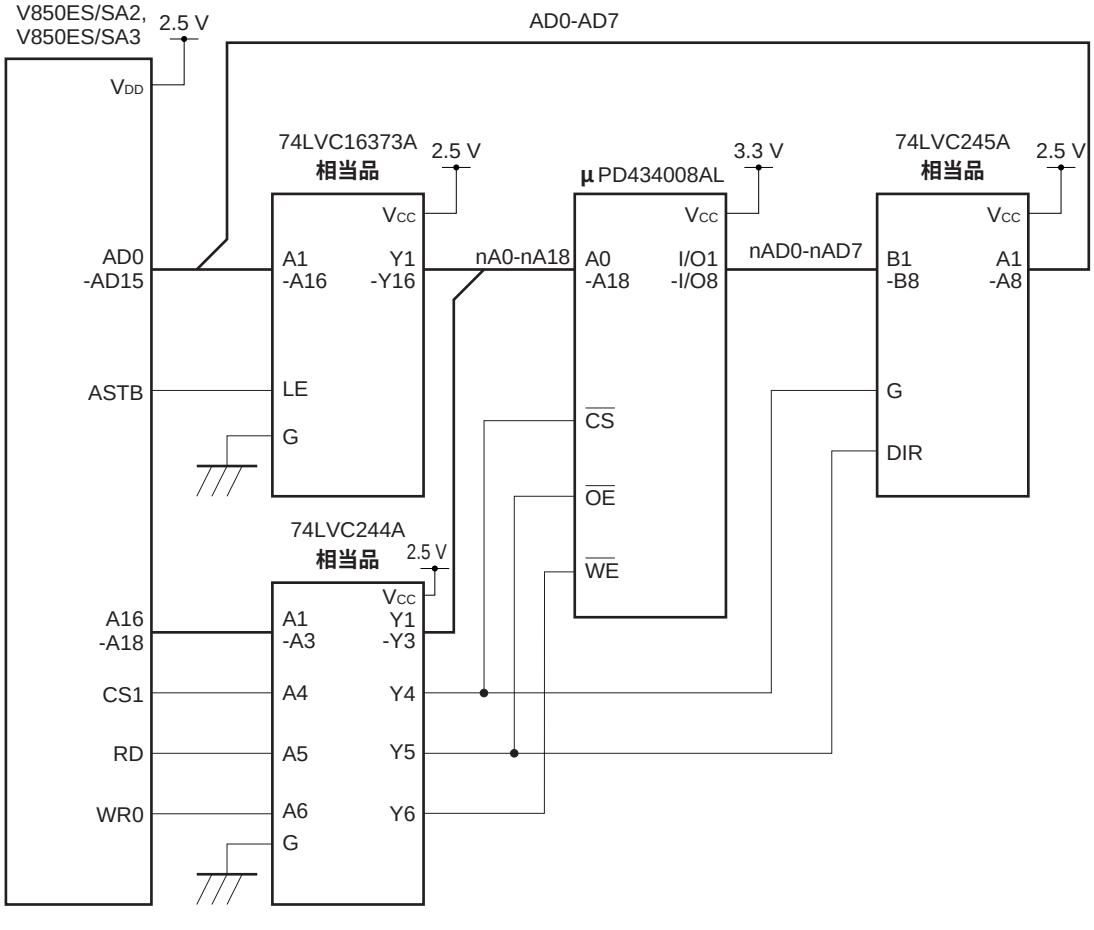
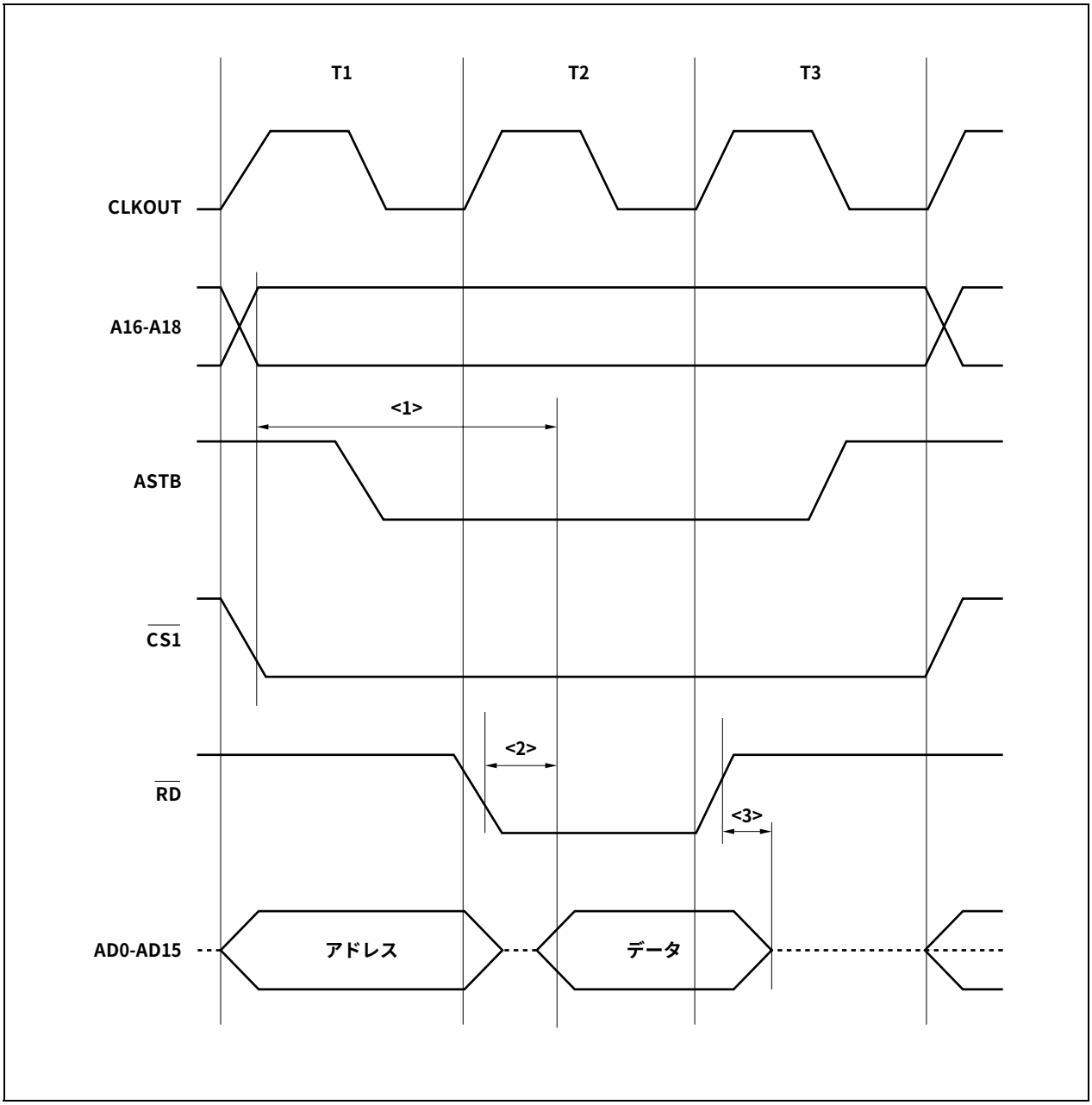


図2－7　μPD434008AL のリード動作



<1> **このメモリのアドレス，$\overline{CE}$がアクティブになってからのデータ出力遅延時間：**20 ns

V850ES/SA2, V850ES/SA3**の電気的特性（**CLKOUT**非同期／マルチプレクス・バス・モード時）は**

t_{SAID}（ns）（MAX.）=（T1＋T2＋TW×m）－25　（m = 0-7）

=（T1＋T2＋TW×0）－25 = 75〔＞ 20 ns〕

<2> **このメモリの$\overline{OE}$がアクティブになってからのデータ出力遅延時間：**10 ns

V850ES/SA2, V850ES/SA3**の電気的特性（**CLKOUT**非同期／マルチプレクス・バス・モード時）は**

t_{SRID}（ns）（MAX.）=（T2＋TW×m）－25　（m = 0-7）

=（T2＋TW×0）－25 = 25〔＞ 10 ns〕

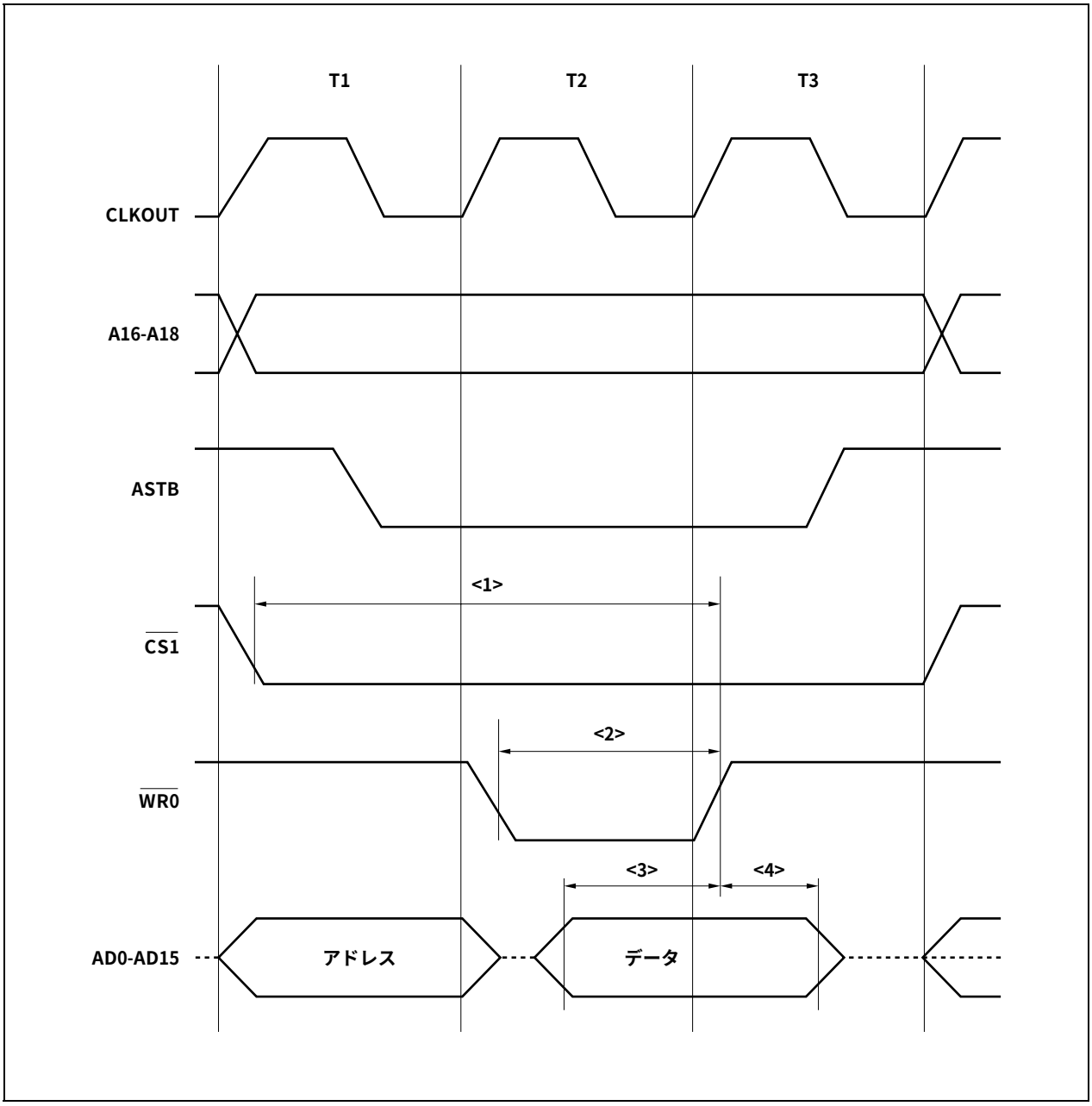
<3> **このメモリの$\overline{OE}$がインアクティブになってからのデータ出力遅延時間：**8 ns

V850ES/SA2, V850ES/SA3**の電気的特性（**CLKOUT**非同期／マルチプレクス・バス・モード時）は**

t_{DRDA}（ns）（MIN.）=（T3＋Ti×m）－15　（m = 0, 1）

=（T3＋Ti×0）－15 = 35〔＞ 8 ns〕

図2－8　μPD434008ALのライト動作



<1> **このメモリの$\overline{CS}$がアクティブになってからデータ・セットまでの時間：**12 ns

$\overline{CS}$がアクティブになったときから$\overline{WR0}$がインアクティブになるまでの時間はT1＋T2**になります。したがって**

T1＋T2 = 100〔＞12ns〕

<2> **このメモリの$\overline{WR0}$に与えるアクティブ・パルス幅：**12 ns

V850ES/SA2, V850ES/SA3**の電気的特性（**CLKOUT**非同期／マルチプレクス・バス・モード時）は**

t_{WRDWRL}（ns）（MIN.）= （T2＋TW×m）－15　（m = 0-7）

= （T2＋TW×0）－15 = 35〔 ＞ 12 ns〕

<3> **このメモリのデータ・セットアップ時間：**9 ns

V850ES/SA2, V850ES/SA3**の電気的特性（**CLKOUT**非同期／マルチプレクス・バス・モード時）は**

t_{SODWR}（ns）（MIN.）= （T2＋TW×m）－20　（m = 0-7）

= （T2＋TW×0）－20 = 30〔 ＞ 9 ns〕

<4> **このメモリのデータ・ホールド時間：**0 ns

V850ES/SA2, V850ES/SA3**の電気的特性（**CLKOUT**非同期／マルチプレクス・バス・モード時）は**

t_{HWROD}（ns）（MIN.）= T3－15 = 35〔 ＞ 0 ns〕

2. 2. 2　16ビット・バス幅例

SRAM（μPD434008AL：512K×8ビット）を2個使用して，1,024 Kバイトの外部メモリ空間を接続する例を示します。

【回路構成】

・システム・クロック：20 MHz
・接続デバイス　　：μPD434008ALxx-A20（2個）
・使用CS信号　　：$\overline{CS1}$
　　占有メモリ空間：0x200000-0x2FFFFF

【接続の考え方と注意点】

偶数バイト側SRAMのデータ・バスはV850ES/SA2, V850ES/SA3のAD0-AD7に，アドレス・バスはV850ES/SA2, V850ES/SA3のA1-A19に接続します。$\overline{CS}$は$\overline{CS1}$に，$\overline{OE}$は$\overline{RD}$に，$\overline{WR}$は$\overline{WR0}$にそれぞれ接続します。

奇数バイト側SRAMのデータ・バスはV850ES/SA2, V850ES/SA3のAD8-AD15に，アドレス・バスはV850ES/SA2, V850ES/SA3のA1-A19に接続します。$\overline{CS}$は$\overline{CS1}$に，$\overline{OE}$は$\overline{RD}$に，$\overline{WR}$は$\overline{WR1}$にそれぞれ接続します。

【レジスタの設定】

バス・サイズの設定以外は2. 2. 1　8ビット・バス幅例と同一設定です。

・バス・サイズの設定：

BS10ビットに“1”を設定し，$\overline{CS1}$領域を16ビット・サイズに設定します。

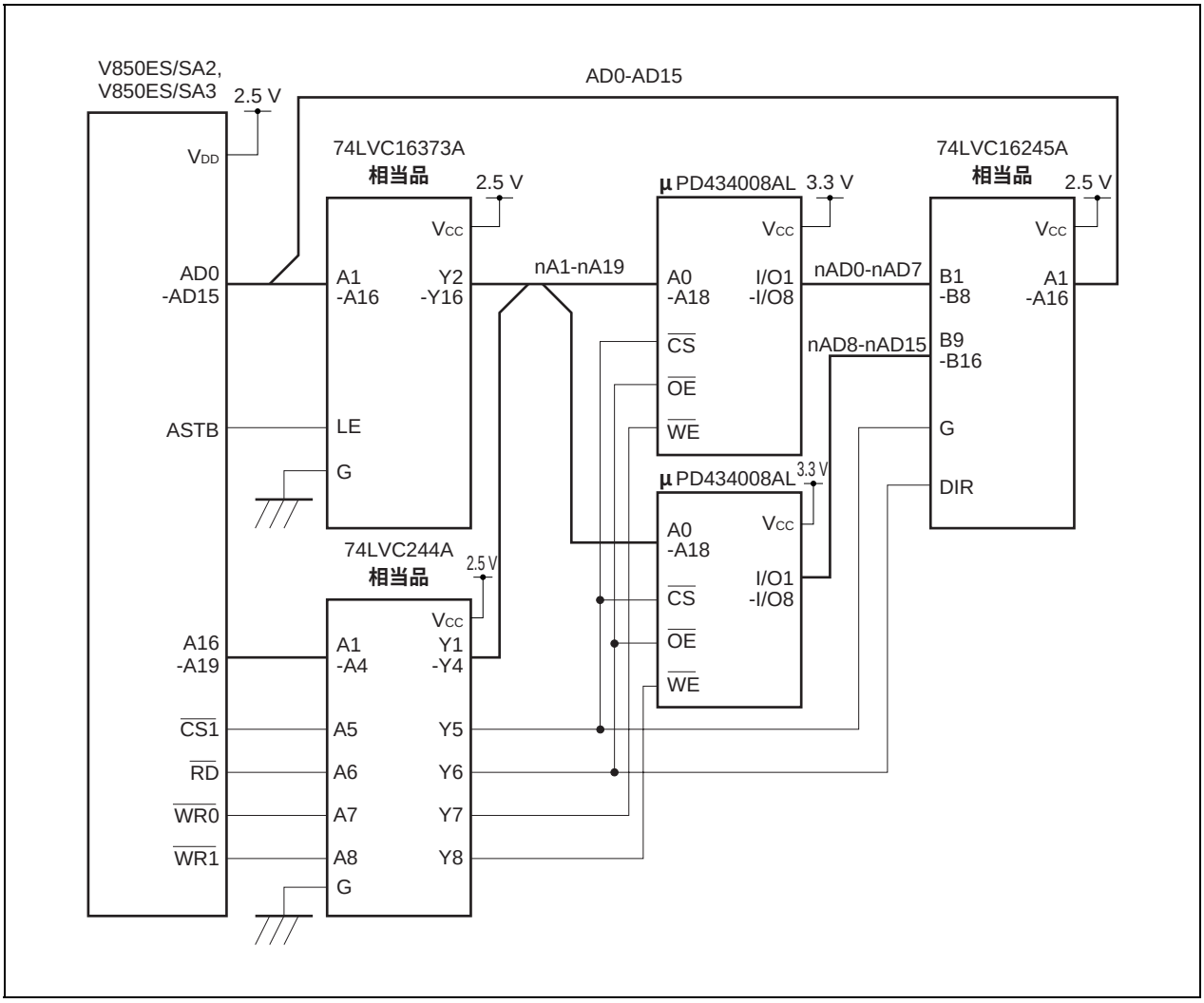
リセット時：5555H　R/W　アドレス：FFFFF066H

	15	14	13	12	11	10	9	8
BSC	0	1	0	1	0	1	0	1
	7	6	5	4	3	2	1	0
	0	BS30	0	BS20	0	BS10	0	BS00
$\overline{CSn}$信号		$\overline{CS3}$		$\overline{CS2}$		$\overline{CS1}$		$\overline{CS0}$

BSC = 010101010x0x010xB

注意　ビット14, 12, 10, 8には必ず“1”を，ビット15, 13, 11, 9, 7, 5, 3, 1には必ず“0”を設定してください。

図2－9　μPD434008ALの16ビット・バス幅接続例



2. 3　16ビットSRAMとの接続

SRAM（μPD434016AL：256K×16ビット）を1個使用して，512 Kバイトの外部メモリ空間を接続する例を示します。メモリ・アドレス0×200000から配置することとし，CS信号は$\overline{CS1}$を使います。

備考　16ビットSRAMのタイミング・チャートは図2－7を参照してください。

【回路構成】

- **システム・クロック**：20 MHz
- **接続デバイス**　：μPD434016ALxx-20（1個）
- **使用CS信号**　：$\overline{CS1}$

  占有メモリ空間0x200000-0x27FFFF

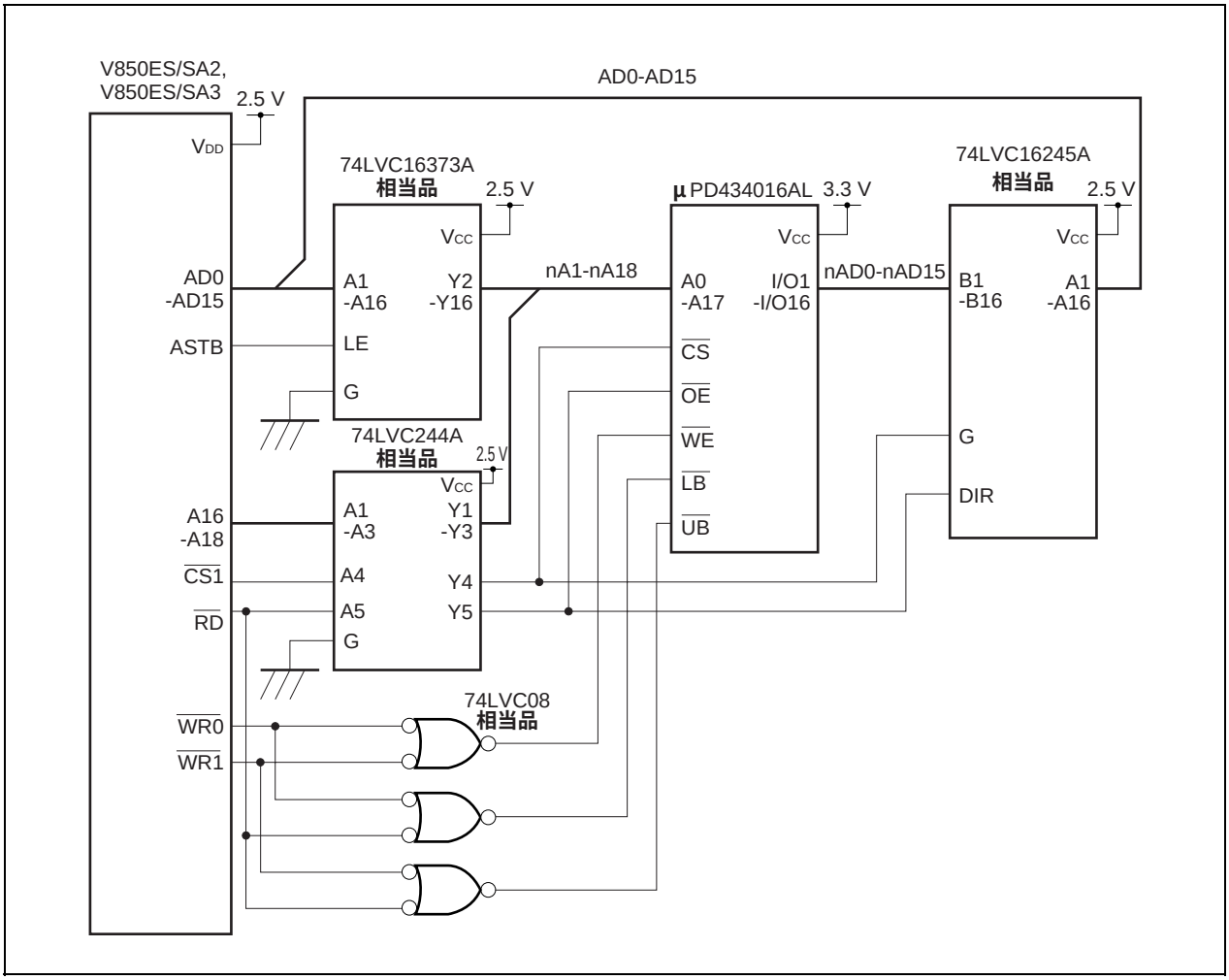
【接続の考え方と注意点】

SRAMのデータ・バスはV850ES/SA2, V850ES/SA3のAD0-AD15に，アドレス・バスはV850ES/SA2, V850ES/SA3のA1-A18に接続します。$\overline{CS}$は$\overline{CS1}$に，$\overline{OE}$は$\overline{RD}$に，$\overline{WR}$は$\overline{WR0}$と$\overline{WR1}$から生成しそれぞれ接続します。$\overline{LB}$と$\overline{UB}$は$\overline{RD}$と$\overline{WR0}$, $\overline{WR1}$から生成し接続します。

【レジスタの設定】

2. 2. 2　16ビット・バス幅例を参照してください

図2－10　μPD434016ALの16ビット・バス幅接続例



2. 4　フラッシュ・メモリとの接続

フラッシュ・メモリ（μPD29F032202AL：2 M×16ビット）を1個使用して，4 Mバイトの外部メモリ空間を接続する例を示します。

メモリ・アドレス0x400000から配置することとし，CS信号は$\overline{CS2}$を使います。

このメモリはワード・モード（16ビット・データ幅）の設定とします。

【回路構成】

・システム・クロック　：20 MHz

・接続デバイス　　：μPD29F032202AL（1個）

・使用CS信号　　：$\overline{CS2}$

占有メモリ空間：0x400000-0x7FFFFF

【接続の考え方と注意点】

フラッシュ・メモリのデータ・バスはV850ES/SA2, V850ES/SA3のAD0-AD15に，アドレス・バスはV850ES/SA2, V850ES/SA3のA1-A21に接続します。$\overline{CE}$端子は$\overline{CS2}$に，$\overline{OE}$端子は$\overline{RD}$に，$\overline{WE}$端子は$\overline{WR0}$と$\overline{WR1}$から生成しそれぞれ接続します。$\overline{BYTE}$端子をV_{DD}へ，プルアップしワード・モードとします。$\overline{RESET}$端子はパワーオン・リセット信号などシステム・リセットの信号に接続します。

備考　RY/$\overline{BY}$をポート入力などに接続することによりライト／イレース時の完了状態を，プログラム上から監視ができます。

【レジスタの設定】

・バス・サイズの設定（BSC）

BS20ビットに“1”を設定し，$\overline{CS2}$領域を16ビット・サイズに設定します。

リセット時：5555H　R/W　アドレス：FFFFF066H

	15	14	13	12	11	10	9	8
BSC	0	1	0	1	0	1	0	1

	7	6	5	4	3	2	1	0
	0	BS30	0	BS20	0	BS10	0	BS00
$\overline{CSn}$信号		$\overline{CS3}$		$\overline{CS2}$		$\overline{CS1}$		$\overline{CS0}$

BSC = 010101010x010x0xB

注意　ビット14, 12, 10, 8には必ず“1”を，ビット15, 13, 11, 9, 7, 5, 3, 1には必ず“0”を設定してください。

・データ・ウエイトの設定（DWC0）

DW22-DW20ビットに“001”を設定し，$\overline{CS2}$領域を1ウエイト挿入に設定します。

リセット時：7777H　R/W　アドレス：FFFFF484H

	15	14	13	12	11	10	9	8
DWC0	0	DW32	DW31	DW30	0	DW22	DW21	DW20
$\overline{CSn}$信号		$\overline{CS3}$				$\overline{CS2}$		

	7	6	5	4	3	2	1	0
	0	DW12	DW11	DW10	0	DW02	DW01	DW00
$\overline{CSn}$信号		$\overline{CS1}$				$\overline{CS0}$		

DWC0 = 0xxx00010xxx0xxxB

注意　ビット15, 11, 7, 3には必ず“0”を設定してください。

・アドレス・ウエイトの設定（AWC）

AHW2, ASW2ビットに“0”を設定し，$\overline{CS2}$領域にアドレス・ホールド・ウエイト，アドレス・セットアップ・ウエイトの双方を“挿入しない”に設定します。

リセット時：FFFFH　R/W　アドレス：FFFFF488H

	15	14	13	12	11	10	9	8
AWC	1	1	1	1	1	1	1	1

	7	6	5	4	3	2	1	0
	AHW3	ASW3	AHW2	ASW2	AHW1	ASW1	AHW0	ASW0
$\overline{CSn}$信号		$\overline{CS3}$		$\overline{CS2}$		$\overline{CS1}$		$\overline{CS0}$

AWC = 11111111xx00xxxxB

注意　ビット15-8には必ず“1”を設定してください。

・アイドル・ステートの設定（BCC）

BC21ビットに“0”を設定し，$\overline{CS2}$領域にアイドル・ステートを“挿入しない“に設定します。

リセット時：AAAAH　R/W　アドレス：FFFFF48AH

	15	14	13	12	11	10	9	8
BCC	1	0	1	0	1	0	1	0

	7	6	5	4	3	2	1	0
	BC31	0	BC21	0	BC11	0	BC01	0
$\overline{CSn}$信号	$\overline{CS3}$		$\overline{CS2}$		$\overline{CS1}$		$\overline{CS0}$	

BCC = 10101010x000x0x0B

注意　ビット14, 12, 10, 8, 6, 4, 2, 0**には必ず“**0**”を，ビット**15, 13, 11, 9**には必ず“**1**”を設定してください。**

図2－11　**μPD29F032202ALの16ビット・バス幅接続例**

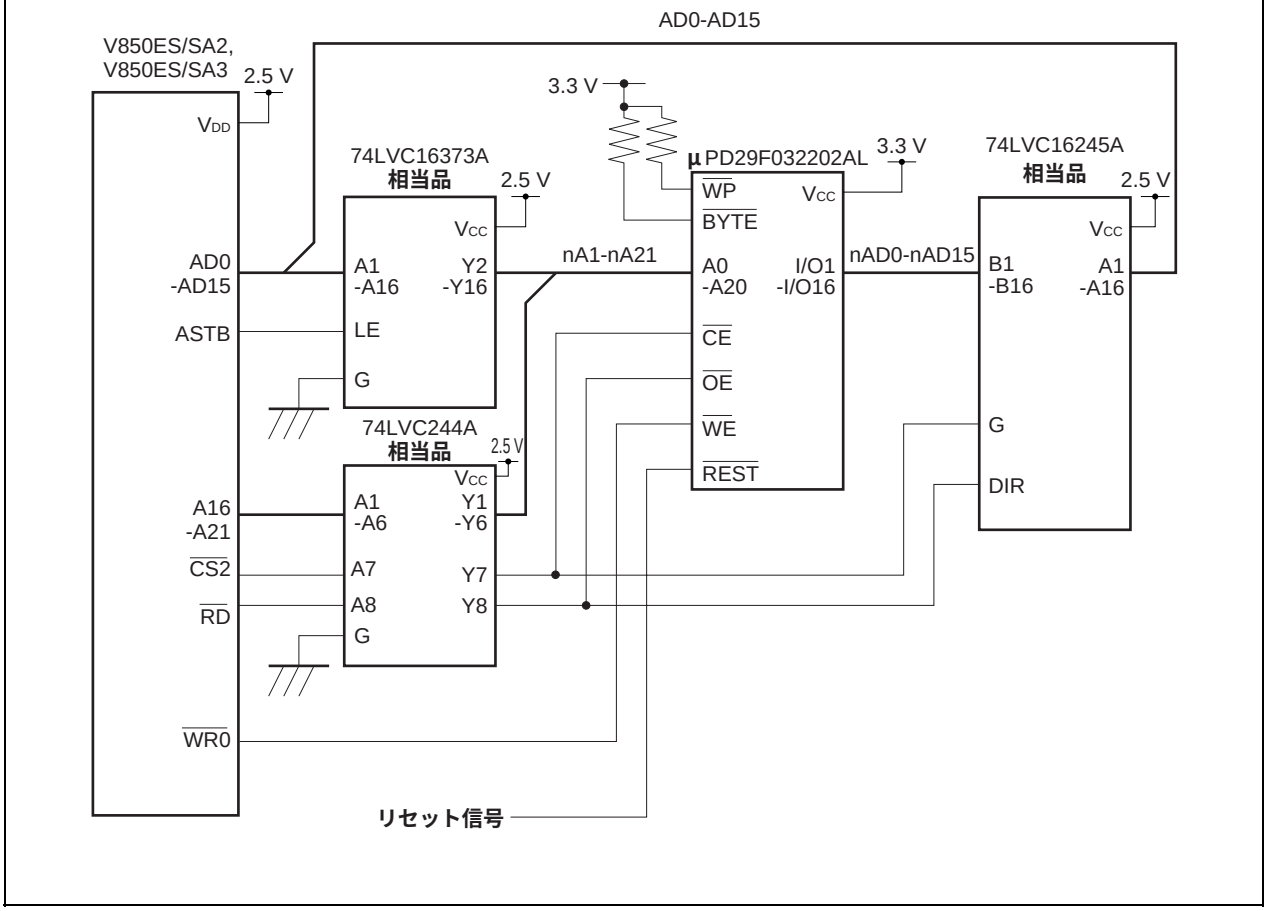
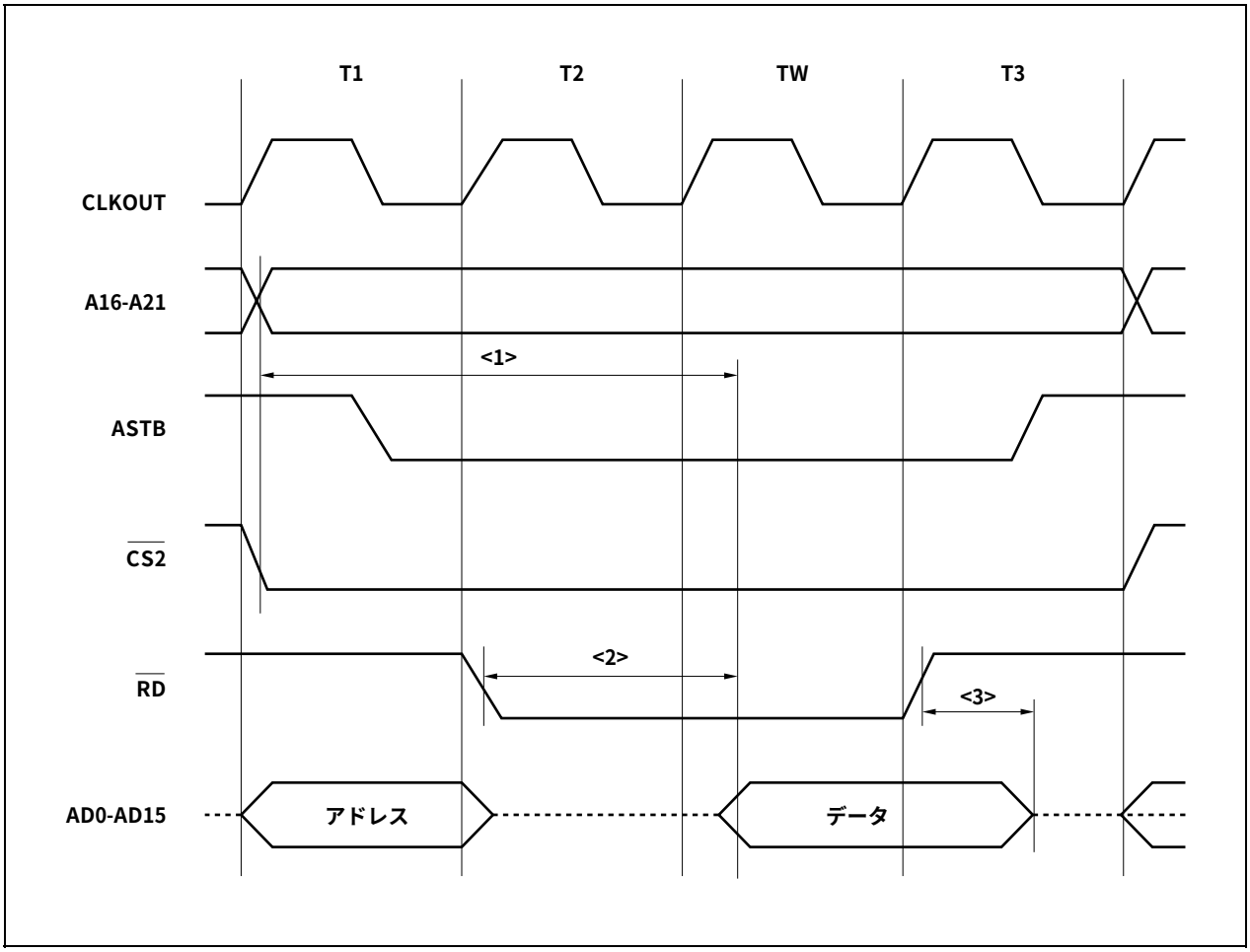


図2－12　μPD29F032202ALのリード動作



<1> **このメモリのアドレス，$\overline{CE}$がアクティブになってからのデータ出力遅延時間：**85 ns

V850ES/SA2，V850ES/SA3**の電気的特性（**CLKOUT**非同期／マルチプレクス・バス・モード時）は**

t_{SAID}（ns）（MAX.）=（T1＋T2＋TW×m）－25　（m = 0-7）

=（T1＋T2＋TW×1）－25 = 125〔 ＞ 85 ns〕

<2> **このメモリの$\overline{OE}$がアクティブになってからのデータ出力遅延時間：**40　ns

V850ES/SA2，V850ES/SA3**の電気的特性（**CLKOUT**非同期／マルチプレクス・バス・モード時）は**

t_{SRID}（ns）（MAX.）=（T2＋TW×m）－25　（m = 0-7）

=（T2＋TW×1）－25 = 75〔 ＞ 40 ns〕

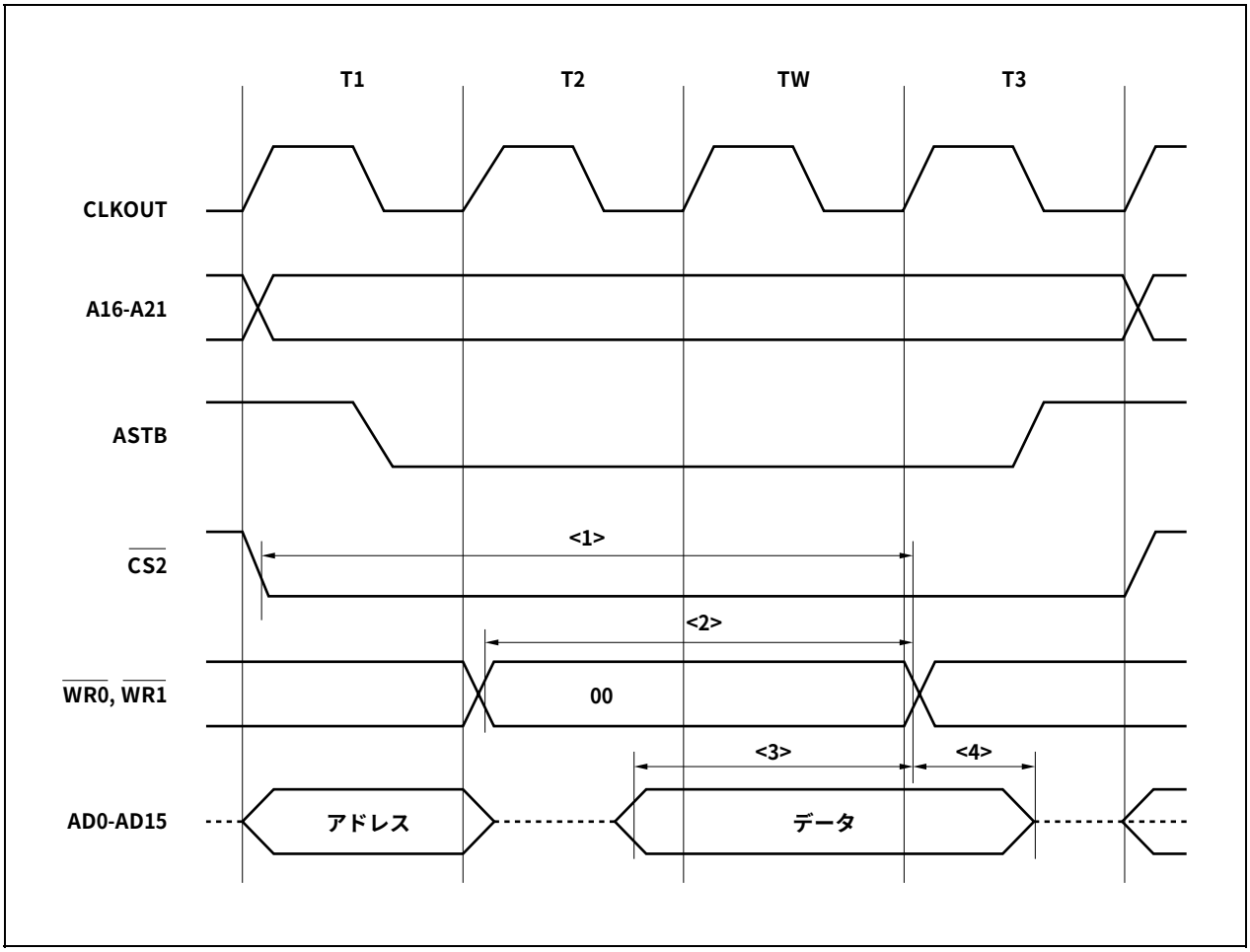
<3> **このメモリの$\overline{OE}$がインアクティブになってからのデータ出力遅延時間：**30ns

V850ES/SA2，V850ES/SA3**の電気的特性（**CLKOUT**非同期／マルチプレクス・バス・モード時）は**

t_{DRDA}（ns）（MAX.）=（T3＋Ti×m）－15　（m = 0, 1）

=（T3＋Ti×0）－15 = 35〔 ＞ 30 ns〕

図2－13　μPD29F032202ALのライト動作



<1> **このメモリの$\overline{CE}$がアクティブになってからデータ・セットまでの時間：**35 ns

$\overline{CE}$がアクティブになったときから$\overline{WR}$がインアクティブになるまでの時間はT1＋T2**になります。したがって**

T1＋T2 = 100〔 ＞ 35 ns〕

<2> **このメモリの$\overline{WR}$に与えるアクティブ・パルス幅：**35 ns

V850ES/SA2, V850ES/SA3**の電気的特性（**CLKOUT**非同期／マルチプレクス・バス・モード時）は**

t_{WRDWRL}（ns）（MIN.）= （T2＋TW×m）－15　（m = 0-7）

= （T2＋TW×1）－15 = 85〔 ＞ 35 ns〕

<3>**このメモリのデータ・セットアップ時間：**35 ns

V850ES/SA2, V850ES/SA3**の電気的特性（**CLKOUT**非同期／マルチプレクス・バス・モード時）は**

t_{SODWR}（ns）（MIN.）= （T2＋TW×m）－20　（m = 0-7）

= （T2＋TW×1）－20 = 80〔 ＞ 35 ns〕

<4> **このメモリのデータ・ホールド時間：**0 ns

V850ES/SA2, V850ES/SA3**の電気的特性（**CLKOUT**非同期／マルチプレクス・バス・モード時）は**

t_{HWROD}（ns）（MIN.）= T3 – 15 = 35〔 ＞ 0 ns〕

2. 5　PROMとの接続

PROM（M27C1024：64 K×16ビット）を1個使用して，128 Kバイトの外部メモリ空間を接続する例を示します。

メモリ・アドレス0x400000から配置することとし，CS信号は$\overline{CS2}$を使います。

このメモリはワード・モード（16ビット・データ幅）の設定とします。

【回路構成】

・システム・クロック　：20MHz

・接続デバイス　　：M27C1024-10（1個）

・使用CS信号　　：$\overline{CS2}$

占有メモリ空間：0x400000-0x7FFFFF

【接続の考え方と注意点】

PROMのデータ・バスはV850ES/SA2, V850ES/SA3のAD0-AD15に，アドレス・バスはV850ES/SA2, V850ES/SA3のA1-A16に接続します。$\overline{CS}$端子は$\overline{CS2}$に，$\overline{OE}$端子は$\overline{RD}$にそれぞれ接続します。$\overline{Program}$端子をV_{CC}へプルアップし書き込み禁止とします。V_{PP}端子はV_{CC}に接続します。

【レジスタの設定】

バス・サイズの設定（BSC）

BS20ビットに“1”を設定し，$\overline{CS2}$領域を16ビット・サイズに設定します。

リセット時：5555H　R/W　アドレス：FFFFF066H

	15	14	13	12	11	10	9	8
BSC	0	1	0	1	0	1	0	1
	7	6	5	4	3	2	1	0
	0	BS30	0	BS20	0	BS10	0	BS00
$\overline{CSn}$信号		$\overline{CS3}$		$\overline{CS2}$		$\overline{CS1}$		$\overline{CS0}$

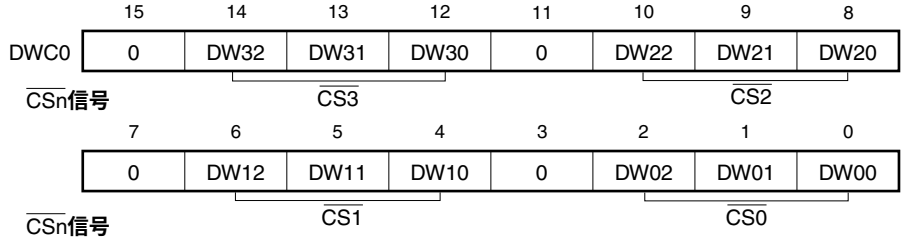
BSC = 010101010x010x0xB

注意　ビット14, 12, 10, 8**には必ず“**1**”を，ビット**15, 13, 11, 9, 7, 5, 3, 1**には必ず“**0**”を設定してください。**

・データ・ウエイトの設定（DWC0）

DW22-DW20**ビットに“** 001 **”を設定し，**$\overline{\text{CS2}}$**領域を**1**ウエイト挿入に設定します。**

リセット時：7777H　R/W　**アドレス：**FFFFF484H



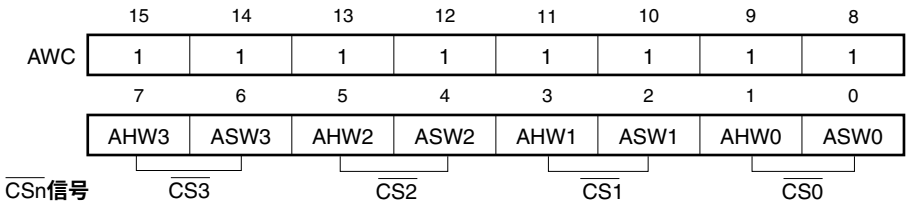
DWC0 = 0xxx00010xxx0xxxB

注意　ビット15, 11, 7, 3**には必ず“** 0 **”を設定してください。**

アドレス・ウエイトの設定（AWC）

AHW2，ASW2**ビットに“** 0 **”を設定し，**$\overline{\text{CS2}}$**領域にアドレス・ホールド・ウエイト，アドレス・セットアップ・ウエイトの双方を“ 挿入しない ”に設定します。**

リセット時：FFFFH　R/W　**アドレス：**FFFFF488H



AWC = 11111111xx00xxxxB

注意　ビット15-8**には必ず“** 1 **”を設定してください。**

・アイドル・ステートの設定（BCC）

BC21ビットに“0”を設定し，$\overline{CS2}$領域にアイドル・ステートを“挿入しない”に設定します。

リセット時：AAAAH　R/W　アドレス：FFFFF48AH

	15	14	13	12	11	10	9	8
BCC	1	0	1	0	1	0	1	0

	7	6	5	4	3	2	1	0
	BC31	0	BC21	0	BC11	0	BC01	0
$\overline{CSn}$信号	$\overline{CS3}$		$\overline{CS2}$		$\overline{CS1}$		$\overline{CS0}$	

BCC = 10101010x000x0x0B

注意　ビット14, 12, 10, 8, 6, 4, 2, 0**には必ず“**0**”を，ビット**15, 13, 11, 9**には必ず“**1**”を設定してください。**

図2－14　M27C1024の16ビット・バス幅接続例

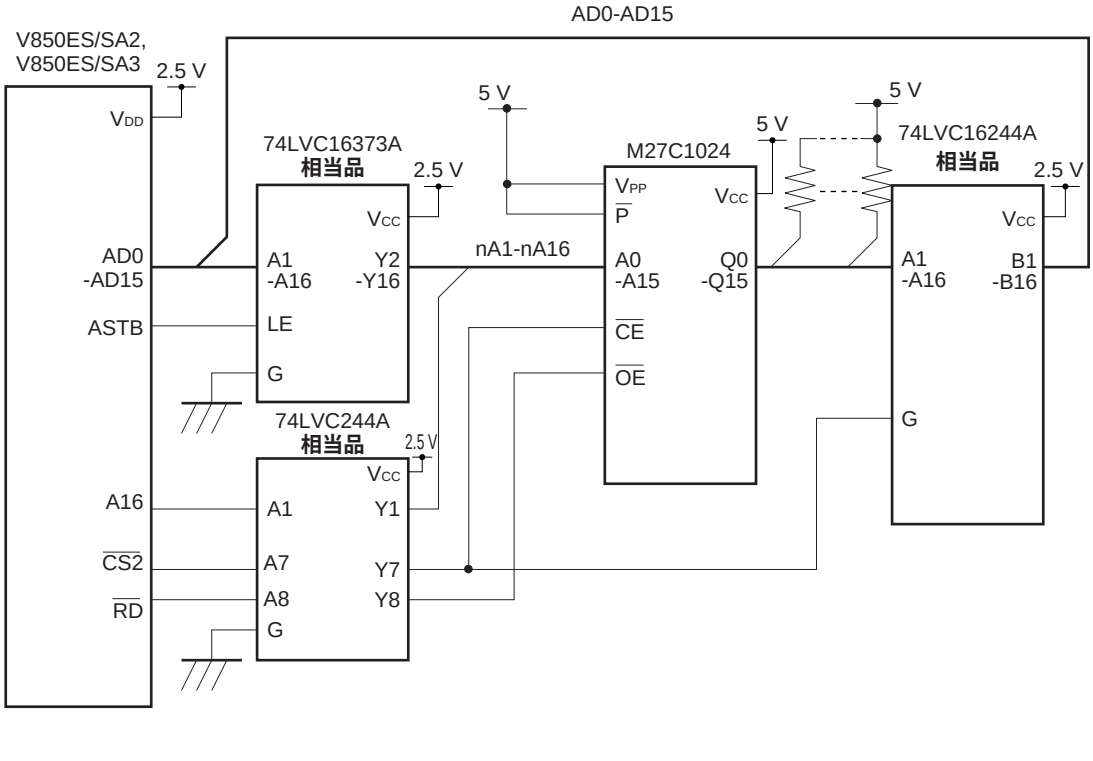
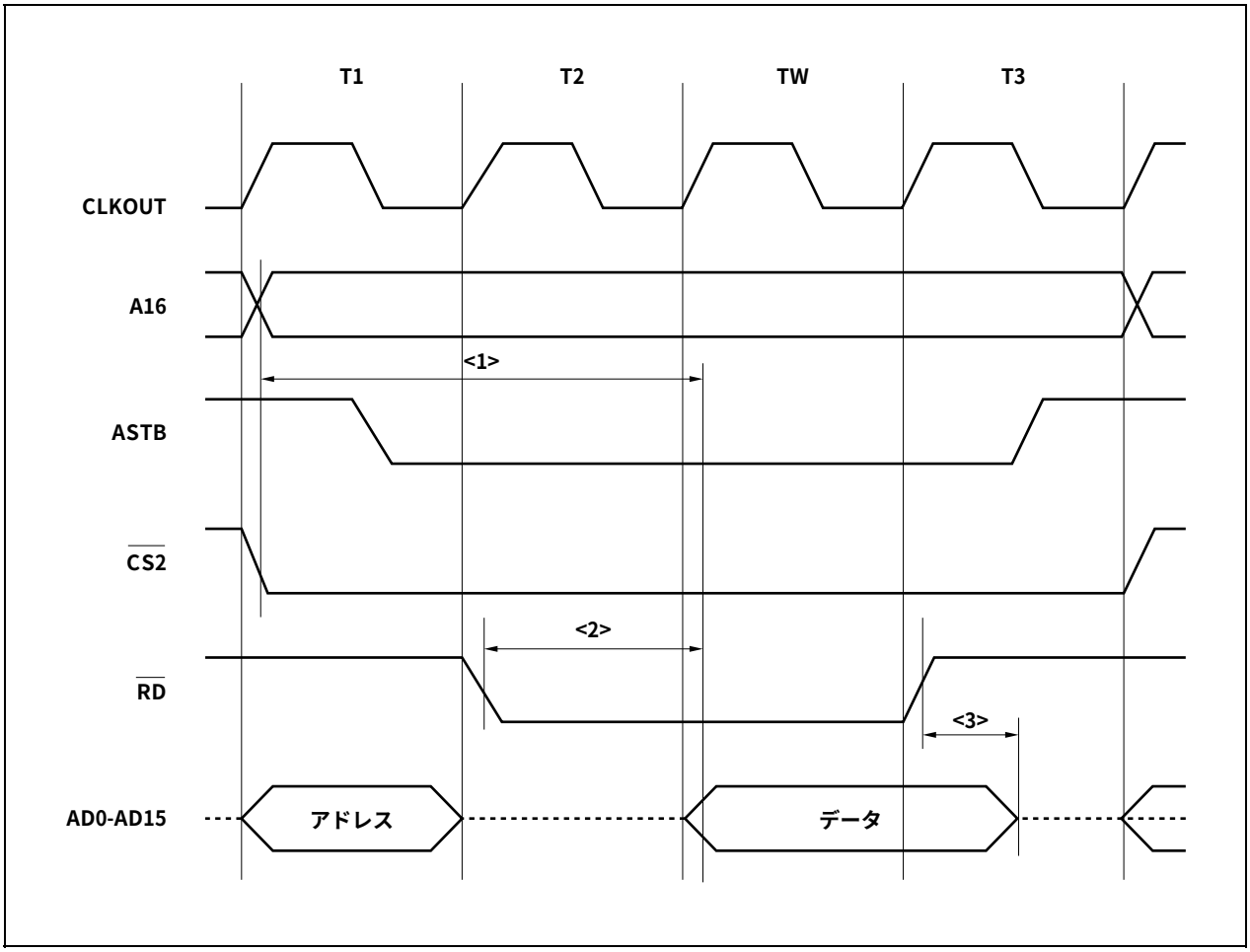


図2－15　M27C1024-10リード動作



<1> **このメモリのアドレス，$\overline{CE}$がアクティブになってからのデータ出力遅延時間：**100 ns

V850ES/SA2，V850ES/SA3**の電気的特性（**CLKOUT**非同期／マルチプレクス・バス・モード時）は**

t_{SAID}（ns）（MAX.）=（T1＋T2＋TW×m）－25　（m = 0-7）

=（T1＋T2＋TW×1）－25 = 125〔 ＞ 100 ns〕

<2> **このメモリの$\overline{OE}$がアクティブになってからのデータ出力遅延時間：**50 ns

V850ES/SA2，V850ES/SA3**の電気的特性（**CLKOUT**非同期／マルチプレクス・バス・モード時）は**

t_{SRID}（ns）（MAX.）=（T2＋TW×m）－25　（m = 0-7）

=（T2＋TW×1）－25 = 75〔 ＞ 50 ns〕

<3> **このメモリの$\overline{OE}$がインアクティブになってからのデータ出力遅延時間：**30 ns

V850ES/SA2，V850ES/SA3**の電気的特性（**CLKOUT**非同期／マルチプレクス・バス・モード時）は**

t_{DRDA}（ns）（MIN.）=（T3＋Ti×m）－15　（m = 0-1）

=（T3＋Ti×0）－15 = 35〔 ＞ 30 ns〕

第3章　内蔵周辺機能の接続回路例

3. 1　押しボタン・スイッチの接続（ポート機能）

3. 1. 1　一般的なスイッチ入力でポート0を使用する例

P01/INTP0/TI2端子に押ボタン・スイッチを接続，一般的なスイッチ入力の設定で使用します。

内部プルアップ抵抗を利用します。

スイッチがオンで“0”，オフで“1”が入力されます。

チャタリングはソフトウエアで除去することとします。

【レジスタの設定】

・ポート・モード・コントロール・レジスタ0（PMC0）の設定

PMC01ビットに“0”を設定し，入出力ポートに設定します。

リセット時：00H　R/W　**アドレス：**FFFFF440H

	7	6	5	4	3	2	1	0
PMC0	0	0	PMC05	PMC04	PMC03	PMC02	PMC01	PMC00

PMC0 = 00xxxx0xB

・ポート・モード・レジスタ0（PM0）の設定

PM01ビットに“1”を設定し，入力モードに設定します。

リセット時：FFH　R/W　**アドレス：**FFFFF420H

	7	6	5	4	3	2	1	0
PM0	1	1	PM05	PM04	PM03	PM02	PM01	PM00

PM0 = 11xxxx1xB

・プルアップ抵抗オプション・レジスタ0（PU0）の設定

PU01ビットに“1”を設定し，内蔵プルアップ抵抗を接続する設定にします。

リセット時：00H　R/W　アドレス：FFFFFC40H

	7	6	5	4	3	2	1	0
PU0	0	0	PU05	PU04	PU03	PU02	PU01	PU00

PU0 = 00xxxx1xB

図3－1　スイッチ接続例（1）

3. 1. 2　外部割り込みスイッチ入力でポート0を使用する例

P01/INTP0/TI2端子に押ボタン・スイッチを接続，スイッチを押し下げた時点で割り込み要因の発生を行わせます。

チャタリングはハードウエアで除去します。

【レジスタの設定】

・ポート・モード・コントロール・レジスタ0（PMC0）の設定

PMC01ビットに“1”を設定し，外部割り込みまたはタイマ入力を設定します。

リセット時：00H　R/W　アドレス：FFFFF440H

	7	6	5	4	3	2	1	0
PMC0	0	0	PMC05	PMC04	PMC03	PMC02	PMC01	PMC00

PMC0 = 00xxxx1xB

・外部割り込み立ち下がりエッジ指定レジスタ0（INTF0）の設定

INTF01ビットに“1”を設定し，立ち下がりエッジ検出を有効に設定します。

リセット時：00H　R/W　アドレス：FFFFFC00H

	7	6	5	4	3	2	1	0
INTF0	0	0	INTF05	INTF04	INTF03	INTF02	INTR01	INTR00

INTF0 = 00xxxx1xB

・外部割り込み立ち上がりエッジ指定レジスタ0（INTR0）の設定

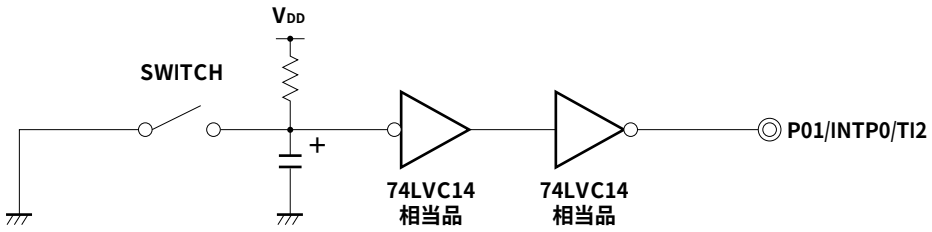
INTR01ビットに“0”を設定し，立ち上がりエッジ検出を無効に設定します。

リセット時：00H　R/W　アドレス：FFFFFC20H

	7	6	5	4	3	2	1	0
INTR0	0	0	INTR05	INTR04	INTR03	INTR02	INTR01	INTR00

INTR0 = 00xxxx0xB

図3－2　スイッチ接続例（2）



3. 2　LEDの接続（ポート機能）

P21端子にLEDを接続し，点灯，消灯による状態表示をさせます。

ポート出力が“0”のとき，LEDは点灯します。

ポート出力が“1”のとき，LEDは消灯します。

【レジスタの設定】

・ポート・モード・コントロール・レジスタ2（PMC2）の設定

PMC21ビットに“0”を設定し，入出力ポートに設定します。

リセット時：00H　R/W　アドレス：FFFFF444H

	7	6	5	4	3	2	1	0
PMC2	0	0	0	0	0	PMC22	PMC21	PMC20

PMC2 = 00000x0xB

・ポート・モード・レジスタ2（PM2）の設定

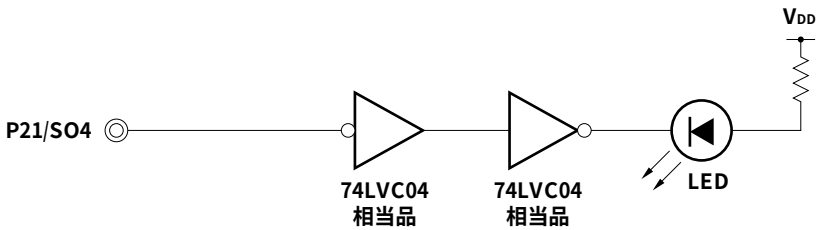
PM21ビットに“0”を設定し，出力モードに設定します。

リセット時：FFH　R/W　アドレス：FFFFF424H

	7	6	5	4	3	2	1	0
PM2	1	1	1	1	1	PM22	PM21	PM20

PM2 = 11111x0xB

図3－3　LED接続例



3. 3　小型DCモータの接続（ポート機能）

P21, P22の2つのポートを使用して，小型DCモータの正逆回転，停止を行います。

モータ・ドライバ（LB1638相当品）は，2つの入力レベルの組み合わせにより，モータに対し表3－1の駆動を行います。

表3－1　モータの駆動

IN1	IN2	モード
H	L	正　転
L	H	逆　転
H	H	ブレーキ
L	L	待　機

2つの出力ポート信号をモータ・ドライバに入力することで，プログラムによりモータの駆動を制御できます。また，一定周期内でハイ・レベル，ロウ・レベルの時間を変える（たとえば正転の場合はIN1のハイ・レベルの時間を増減させる）ことによって，モータ回転速度の制御もできます。

【レジスタの設定】

・ポート・モード・コントロール・レジスタ2（PMC2）の設定

PMC21, PMC22ビットに“0”を設定し，入出力ポートに設定します。

リセット時：00H　R/W　アドレス：FFFFF444H

	7	6	5	4	3	2	1	0
PMC2	0	0	0	0	0	PMC22	PMC21	PMC20

PMC2 = 0000000xB

・ポート・モード・レジスタ2（PM2）の設定

PM21, PM22ビットに“0”を設定し，出力モードに設定します。

リセット時：FFH　R/W　アドレス：FFFFF424H

	7	6	5	4	3	2	1	0
PM2	1	1	1	1	1	PM22	PM21	PM20

PM2 = 1111100xB

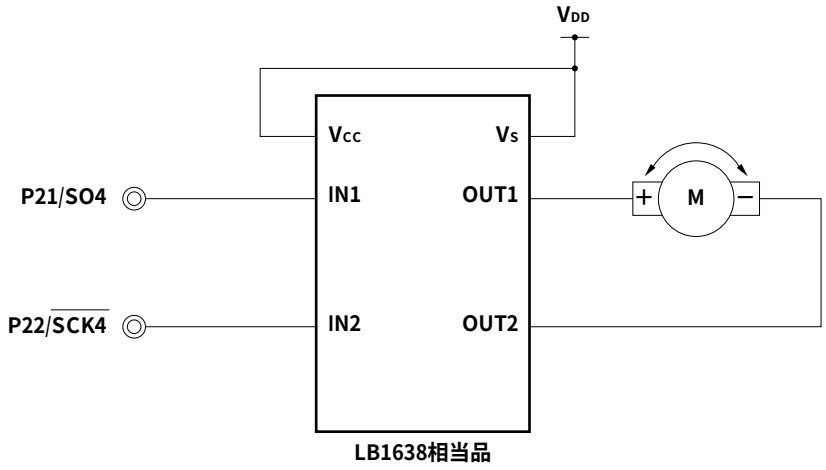
・ポート・レジスタ2（P2）の設定（正転の場合）

リセット時：不定　　R/W　　アドレス：FFFFF404H

	7	6	5	4	3	2	1	0
P2	0	0	0	0	0	P22	P21	P20

P2 = 0000001xB

図3－4　小型DCモータの接続例



3. 4　RS-232-Cインタフェースの接続（シリアル・インタフェース機能）

UART1を使用してRS-232-Cインタフェース接続をします。

【接続の考え方と注意点】

P99/A9/TXD1端子をデータの送信に，P98/A8/RXD1端子をデータの受信に設定します。これを，レベル変換素子を経由して専用素子に接続します。DTR信号はアクティブ・レベルに固定します。送受信クロックは専用のボー・レート・ジェネレータ1（BRG1）により設定しておきます。

【レジスタの設定】

・ポート・ファンクション・コントロール・レジスタ9（PFC9）の設定：

PFC99, PFC98ビットに“11”を設定し，TXD1出力とRXD1入力を設定します。

リセット時：0000H　R/W　**アドレス：**FFFFF472H, FFFFF473H

	15	14	13	12	11	10	9	8
PFC9	PFC910	PFC910	PFC910	PFC910	PFC910	PFC910	PFC99	PFC98

7	6	5	4	3	2	1	0
PFC97	PFC96	PFC95	PFC94	PFC93	PFC92	0	0

PFC9 = xxxxxx11xxxxxx00B

・ポート・モード・コントロール・レジスタ9（PMC9）の設定

PMC99, PMC98ビットに“11”を設定し，セパレート・アドレス・バス出力またはシリアルI/O，タイマ入出力を設定します。

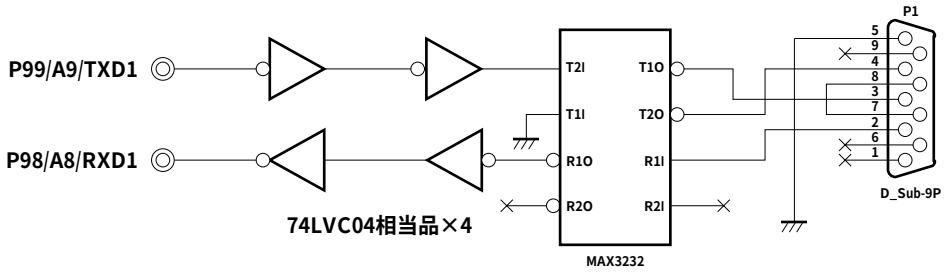
リセット時：0000H　R/W　**アドレス：**FFFFF452H, FFFFF453H

	15	14	13	12	11	10	9	8
PMC9	PMC915	PMC914	PMC913	PMC912	PMC911	PMC910	PMC99	PMC98

7	6	5	4	3	2	1	0
PMC97	PMC96	PMC95	PMC94	PMC93	PMC92	PMC91	PMC90

PMC9 = xxxxxx11xxxxxxxxB

図3ー5　RS-232-C接続例



ピン番号	信号名	内　　容
1	CD	未接続
2	RXD	受信データ
3	TXD	送信データ
4	DTR	データ・ターミナル・レディ
5	SG	信号グランド
6	DSR	未接続
7	RTC	送信要求
8	CTS	送信可
9	CI	未接続

3. 5　アナログ入力の接続（A/Dコンバータ機能）

P70/ANI0端子をアナログ電圧入力で使用します。

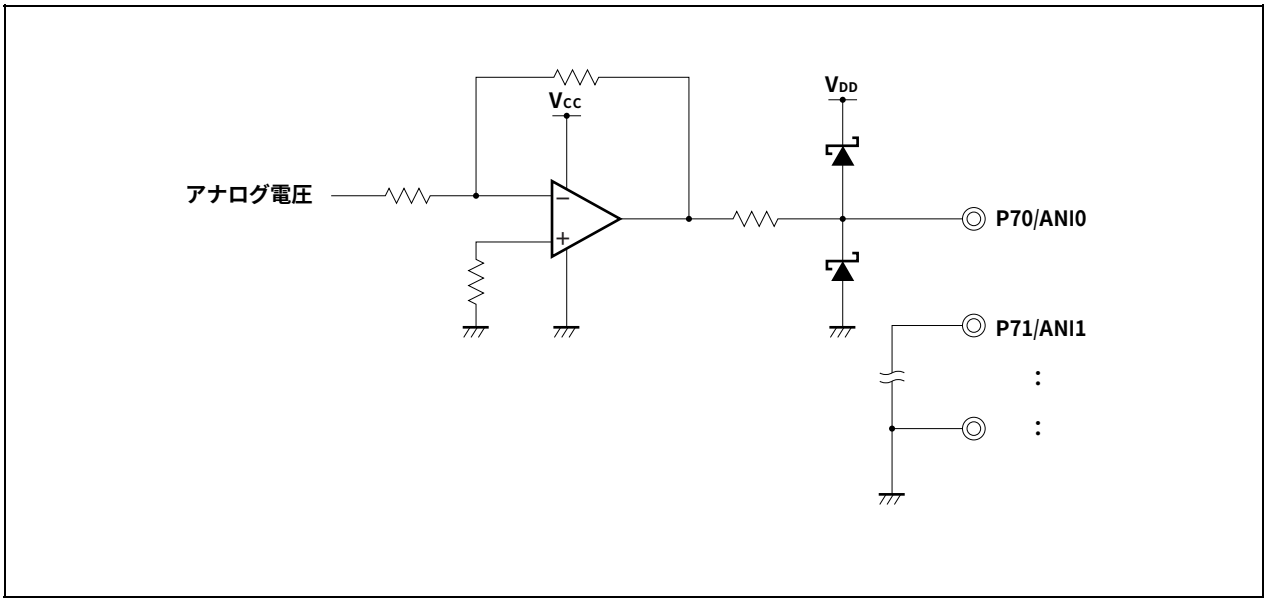
【接続の考え方と注意点】

AVDD, AVREF0にはVDDと同電位を供給します。またAVSSはグランドに接続します。

アナログ入力電圧はAVREF0電圧以上，AVSS電圧以下にならない範囲で供給します。

ポート7（P7）はすべての端子が入力専用です。未使用端子はグランドにクランプします。ポートに対するレジスタの設定はありません。

図3－6　アナログ入力接続例



3. 6　マイクロフォン入力の接続（A/Dコンバータ機能）

P70/ANI0端子をアナログ電圧入力で使用します。

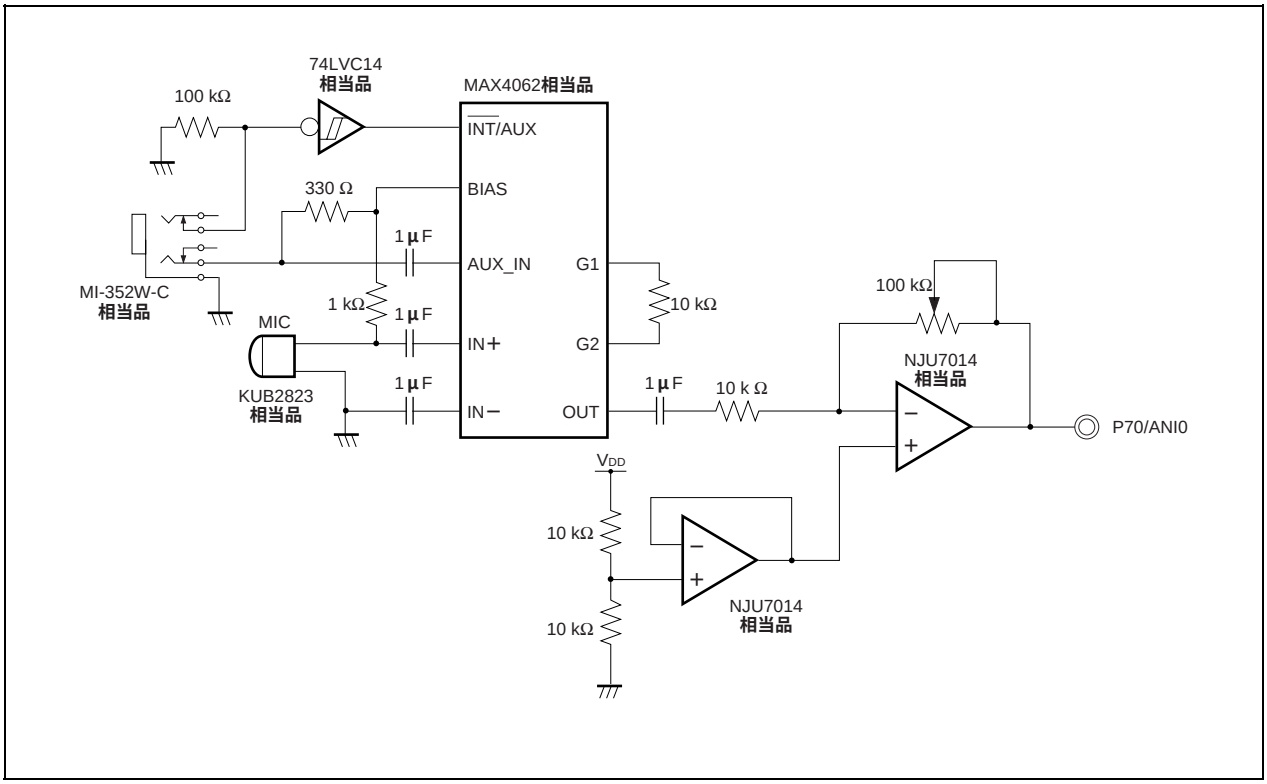
内蔵のマイクロフォンとマイク入力ジャックを備えます。

この入力をアンプで増幅します。次段では，最大入力レベルがA/Dコンバータの10ビット分解能のフルスケールに近くなるように，増幅度の調整をおこないます。

【接続の考え方と注意点】

3. 5　**アナログ入力の接続（A/Dコンバータ機能）**を参照してください。

図3－7　マイクロフォン入力の接続



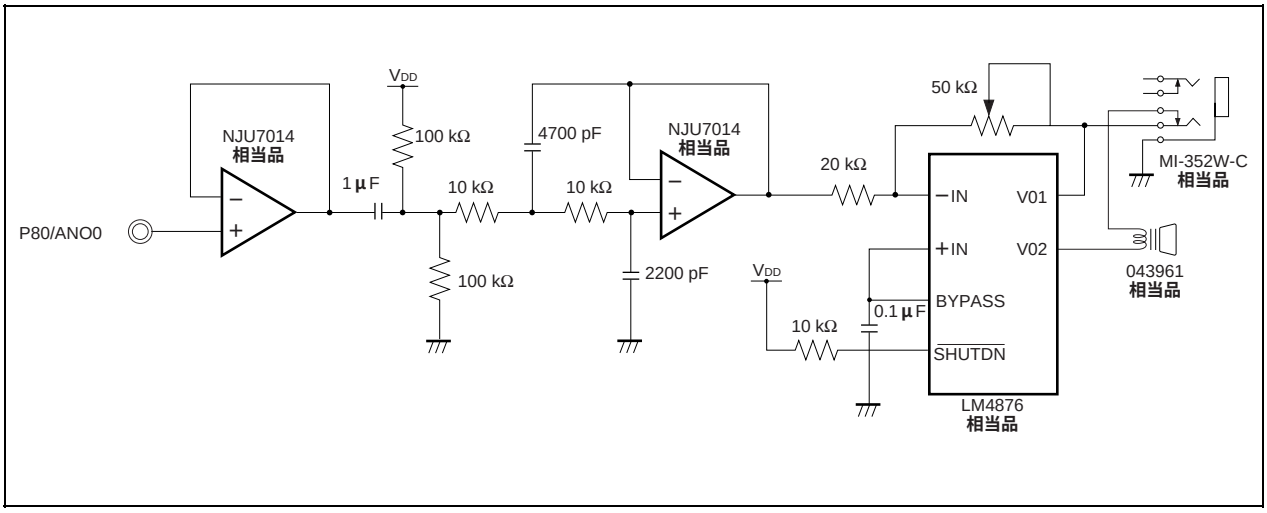
3. 7　スピーカの接続（D/Aコンバータ機能）

P80/ANO0のアナログ出力電圧を増幅し，スピーカを駆動します。

スピーカへの出力ラインは内蔵スピーカ用のコネクタと外部スピーカ・ジャックを備えています。外部スピーカが差し込まれると直接接続されていたスピーカは切断されます。

ポートに対するレジスタの設定はありません。

図3－8　スピーカの接続



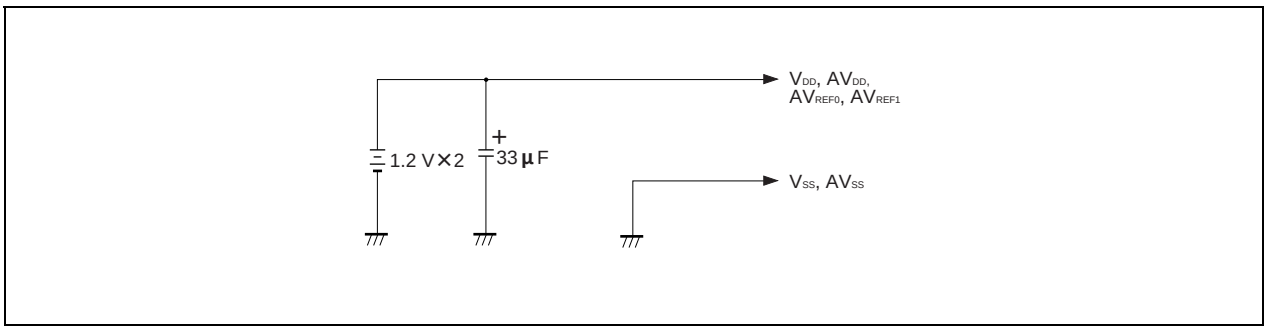
3. 8　電源の接続

3. 8. 1　バッテリ電源を直接供給する例

ニッケル水素電池（1.2 V）2本により2.4 V電圧を供給します。

A/Dコンバータ機能，D/Aコンバータ機能の変換電圧レベルを最大にする（V_{DD}と同一にする）場合および機能を利用しない場合は次のように接続します。

図3－9　電源供給



3. 8. 2　バッテリ電圧を昇圧して供給する例

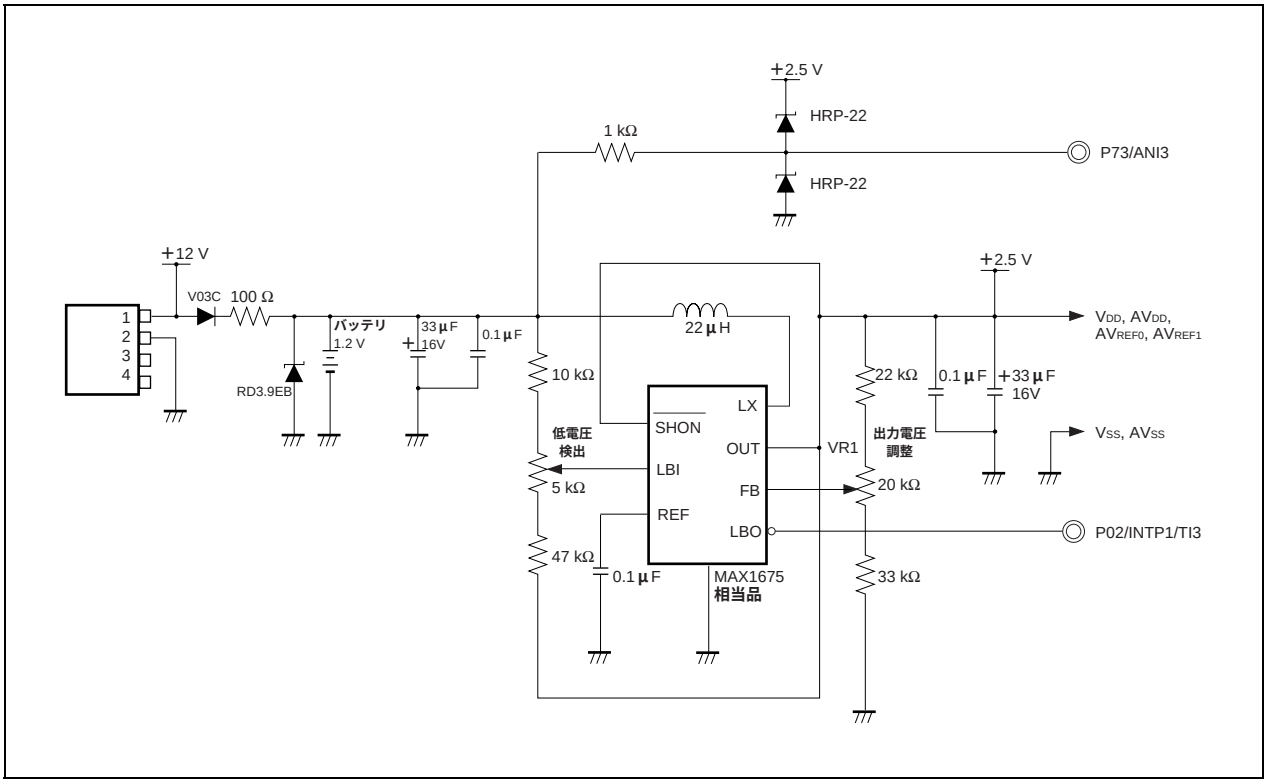
ニッケル水素電池（1.2 V）1本から2.5 Vへ昇圧し供給します。

ステップ・アップDC/DCコンバータ（MAX1675相当品）により昇圧し，出力電圧値はVR1で調整します。

バッテリは外部から供給されるDC 12 Vにより充電可能になっています。

また充電中に誤ってバッテリを取り外し，各部品に過電圧を印加させないために，バッテリに対して並列にツェナー・ダイオードを付加します。

図3－10　電圧昇圧例



3. 8. 3　バッテリ電圧低下検出と外部割り込みの例

ステップ・アップDC/DCコンバータ（MAX1675相当品）の検出機能を利用し，この出力を外部割り込み入力端子（P02/INTP1/TI3）に接続します。

検出電圧値はVR2で調整します（現回路では1 Vに調整）。

電圧が検出値以下になると低電圧検出割り込み（LOWBATT）はロウ・レベルに変化します。

このときポートのレジスタ設定を次のようにすることにより，検出時点で外部割り込み要因を発生させて，プログラム処理を行うことができます。

接続例は図3－10を参照してください。

【レジスタの設定】

・ポート・モード・コントロール・レジスタ0（PMC0）の設定

PMC02ビットに“1”を設定し，外部割り込みまたはタイマ入力を設定します。

リセット時：00H　R/W　アドレス：FFFFF440H

	7	6	5	4	3	2	1	0
PMC0	0	0	PMC05	PMC04	PMC03	PMC02	PMC01	PMC00

PMC0 = 00xxx1xxB

・外部割り込み立ち下がりエッジ指定レジスタ0（INTF0）

INTF02ビットに“1”を設定し，立ち下がりエッジ検出を有効に設定します。

リセット時：00H　R/W　アドレス：FFFFFC00H

	7	6	5	4	3	2	1	0
INTF0	0	0	INTF05	INTF04	INTF03	INTF02	INTR01	INTR00

INTF0 = 00xxx1xxB

外部割り込み立ち上がりエッジ指定レジスタ0（INTR0）

INTR02ビットに“0”を設定し，立ち上がりエッジ検出を無効に設定します。

リセット時：00H　R/W　アドレス：FFFFFC20H

	7	6	5	4	3	2	1	0
INTR0	0	0	INTR05	INTR04	INTR03	INTR02	INTR01	INTR00

INTR0 = 00xxx0xxB

3. 8. 4　バッテリ電圧の監視（A/Dコンバータ機能）の例

V_{DD} = AV_{DD} = AV_{REF0} = 2.5 Vに回路上で設定されています。また「バッテリ電圧＜AV_{REF0}」なのでバッテリ電圧は分圧をせず，抵抗を介して，直接アナログ電圧入力します。ただし，バッテリ未実装時の充電電圧印加への対策として，ダイオードでV_{DD}, GNDへクランプしておきます。

A/Dコンバータは10ビット分解能です。2.5 Vの入力に対する変換値は3FFH（ADCRレジスタ：FFC0H）になるので，1ビットは約2.44 mVになります。したがってバッテリ電圧（1.2 V）は1EBH（ADCRレジスタ：7ACH）に変換されます。

接続例は図3－10を参照してください。

第4章　アプリケーション例

V850ES/SA2, V850ES/SA3の特徴の1つである，低電圧／低消費電力性能に着目し，バッテリ駆動型データ計測システムを応用例として取り上げます。

また，ROM化後のプログラム・ミスを修正するROMコレクション機能を適用して，サンプル・プログラムを記載しています。

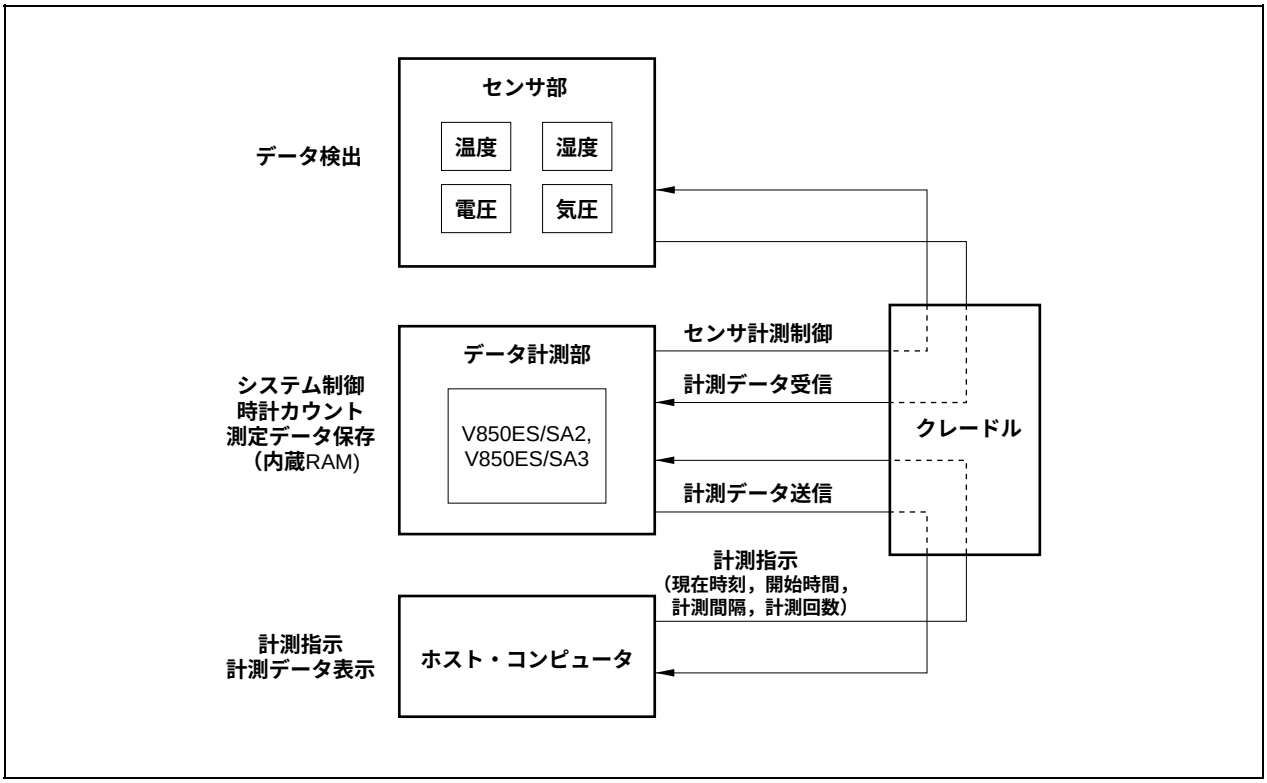
4.1　仕様概要

システムはV850ES/SA2で構成されたデータ計測部，センサ部，ホスト・コンピュータと計測装置とのデータ経由を行うクレードル部で構成されています。

温度，湿度，気圧センサの情報を，設定された時間間隔で内蔵RAMに蓄積します。バッテリ消費を抑えるため，計測時はサブクロック動作モード，待機時はサブIDLEモードで動作させます。指定回数の計測が終了すると，測定したデータの保存のために，メイン・クロックを停止してさらに消費電力を抑えます。また，バッテリ電圧が低下した場合も以後の計測を中止し，同様の状態に入ります。

計測時刻，間隔，回数の設定と収集データの取り出しは，データ計測のクレードルを介してホスト・コンピュータとRS-232-Cによって行っています。

備考　μPD70F3201Yを例としていますが，ほかのデバイスでも端子を変更することなく使用可能です。



4. 2　ハードウエア

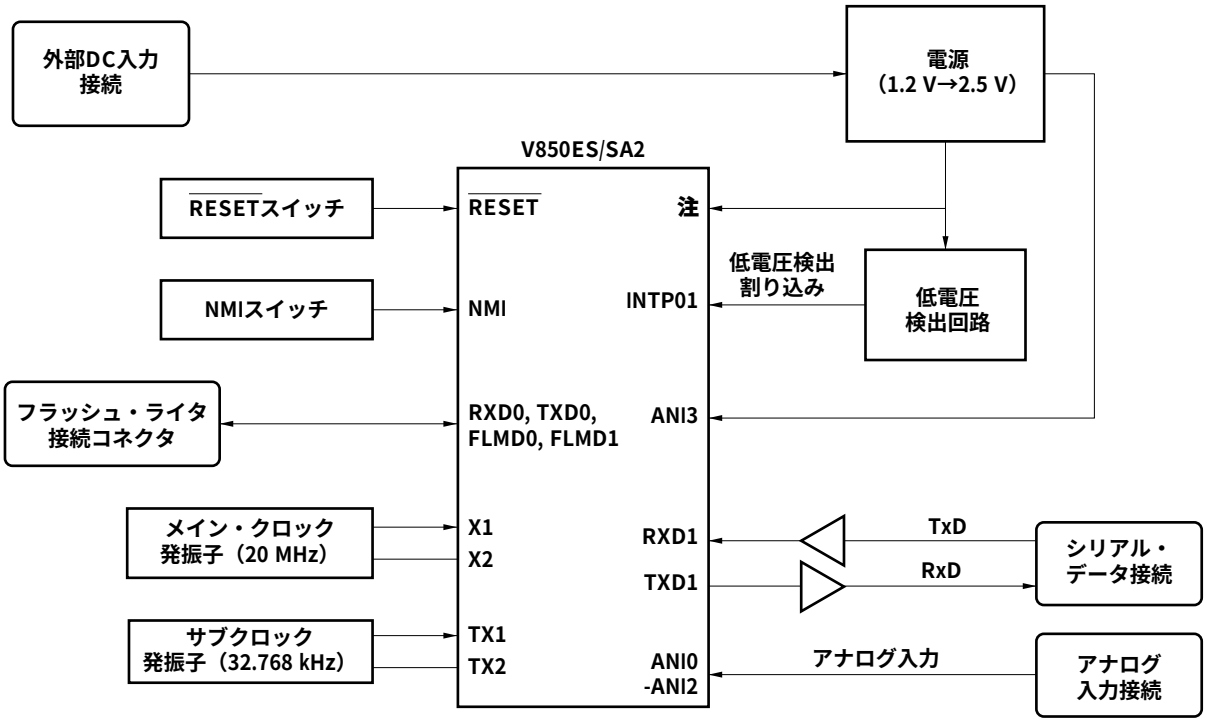
4. 2. 1　ハードウエア構成

データ計測，クレードル，センサの3つのブロックで構成されています。

（1）データ計測

- CPU　：V850ES/SA2（μPD70F3201Y）
- 電源　：DC 1.2 Vニッケル水素電池（1本）からDC 2.5Vへ昇圧
- 電源入力　：DC 12 Vニッケル水素電池充電用
- 水晶振動子　：20 MHz（メイン・クロック用）
  32.768 kHz（サブクロック用）
- 低電圧検出　：バッテリ電圧1.0 V以下で低電圧検出割り込み（LOW_BATT）
- スイッチ　：$\overline{\text{RESET}}$, NMI
- 外部入出力信号　：UART1
  ANI0-ANI2
- その他　：フラッシュ・ライタ接続コネクタ

図4－1　データ計測構成



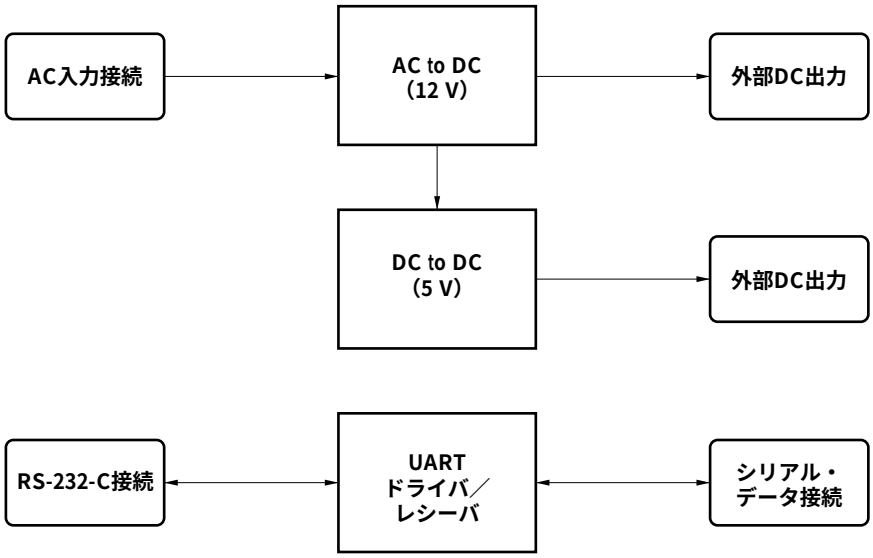
注　V_{DD}, EV_{DD}, AV_{DD}, AV_{REF0}, AV_{REF1}, V_{SS}, EV_{SS}, AV_{SS}

備考　回路図については，**図4－40　データ計測**を参照してください。

(2) クレードル

・電源入力　　：AC 100 V
・電源出力　　：DC 12 V（データ計測用）
　　DC 5 V（センサ用）
・外部入出力信号　：UART1（データ計測およびホストと接続）

図4－2　クレードル構成

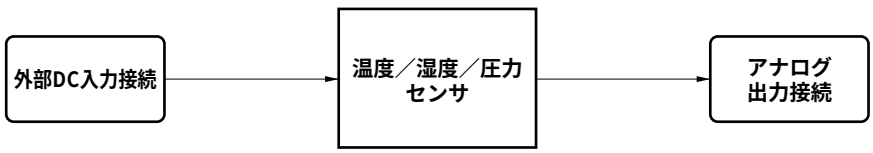


備考　回路図については，図4－41　クレードルを参照してください。

(3) センサ

・電源入力　　：DC 5 V
・外部出力信号　　：ANI0-ANI2（温度，湿度，圧力センサ用）

図4－3　センサ構成



備考　回路図については，図4－42　センサを参照してください。

4. 3　ソフトウエア

4. 3. 1　開発ツール

掲載してあるプログラム例は，すべてNECエレクトロニクス製の開発環境を使用して作成しています。

（1）Cコンパイラ・パッケージ（CA850）

Cコンパイラ・パッケージに関するユーザーズ・マニュアルは次の４つです。

- CA850 Ver.2.50 Cコンパイラ・パッケージ　操作編（U16053J）
- CA850 Ver.2.50 Cコンパイラ・パッケージ　C言語編（U16054J）
- CA850 Ver.2.50 Cコンパイラ・パッケージ　アセンブリ言語編（U16042J）
- PM plus Ver.2.50（U16055J）

備考1. C言語中の#pragma拡張記述やアセンブリ言語中の周辺内蔵I/Oレジスタ名略号記述，擬似命令記述などはNECエレクトロニクス製Cコンパイラ・パッケージ（CA850）固有のものです。

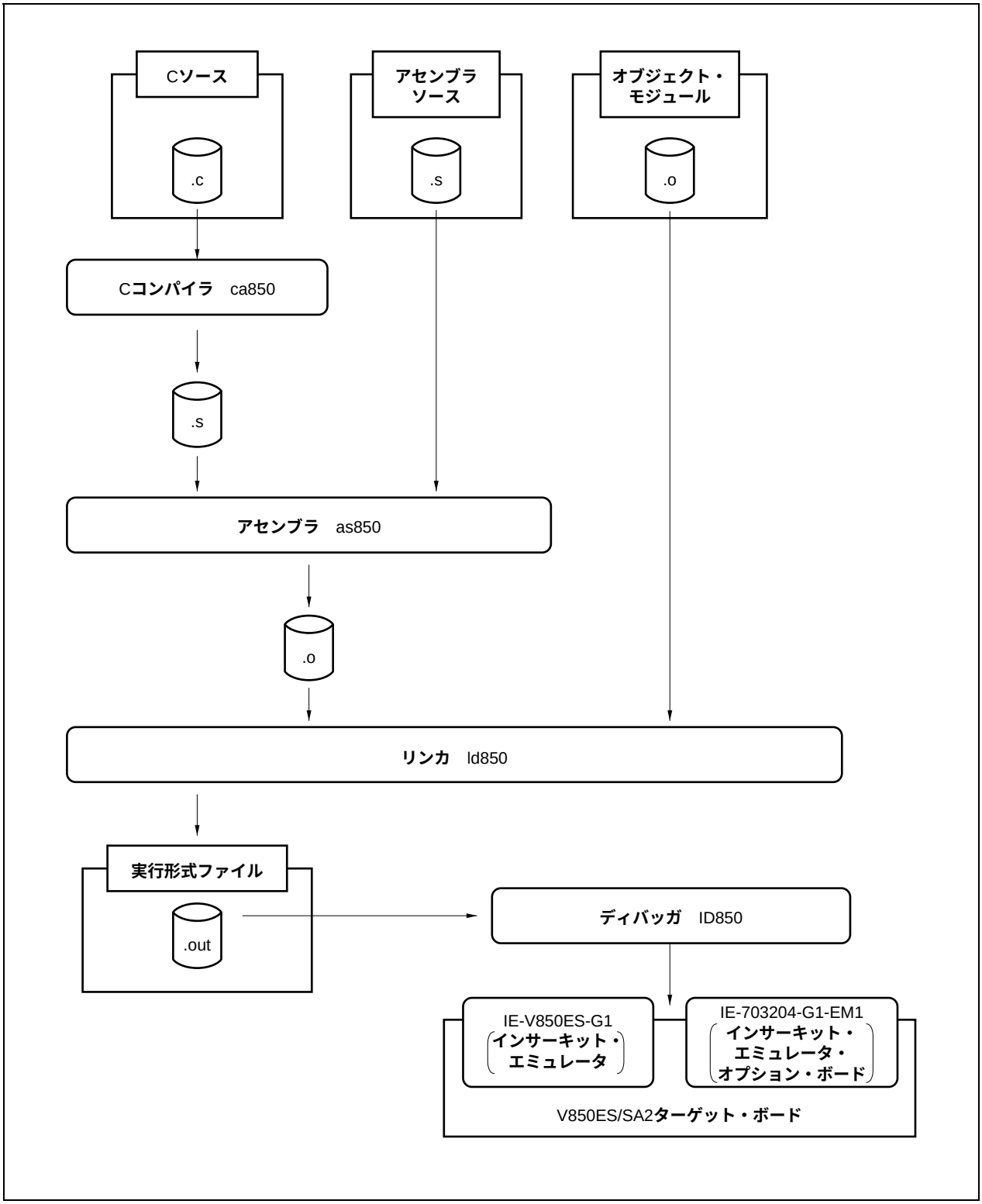
2. 他社製開発ツールを使用する場合，それらの記述がそのまま使えるとは限りませんので，十分に注意してください。

（2）V850ファミリ用Cソース・ディバッガ（ID850）

V850ファミリ用Cソース・ディバッガに関するユーザーズ・マニュアルです。

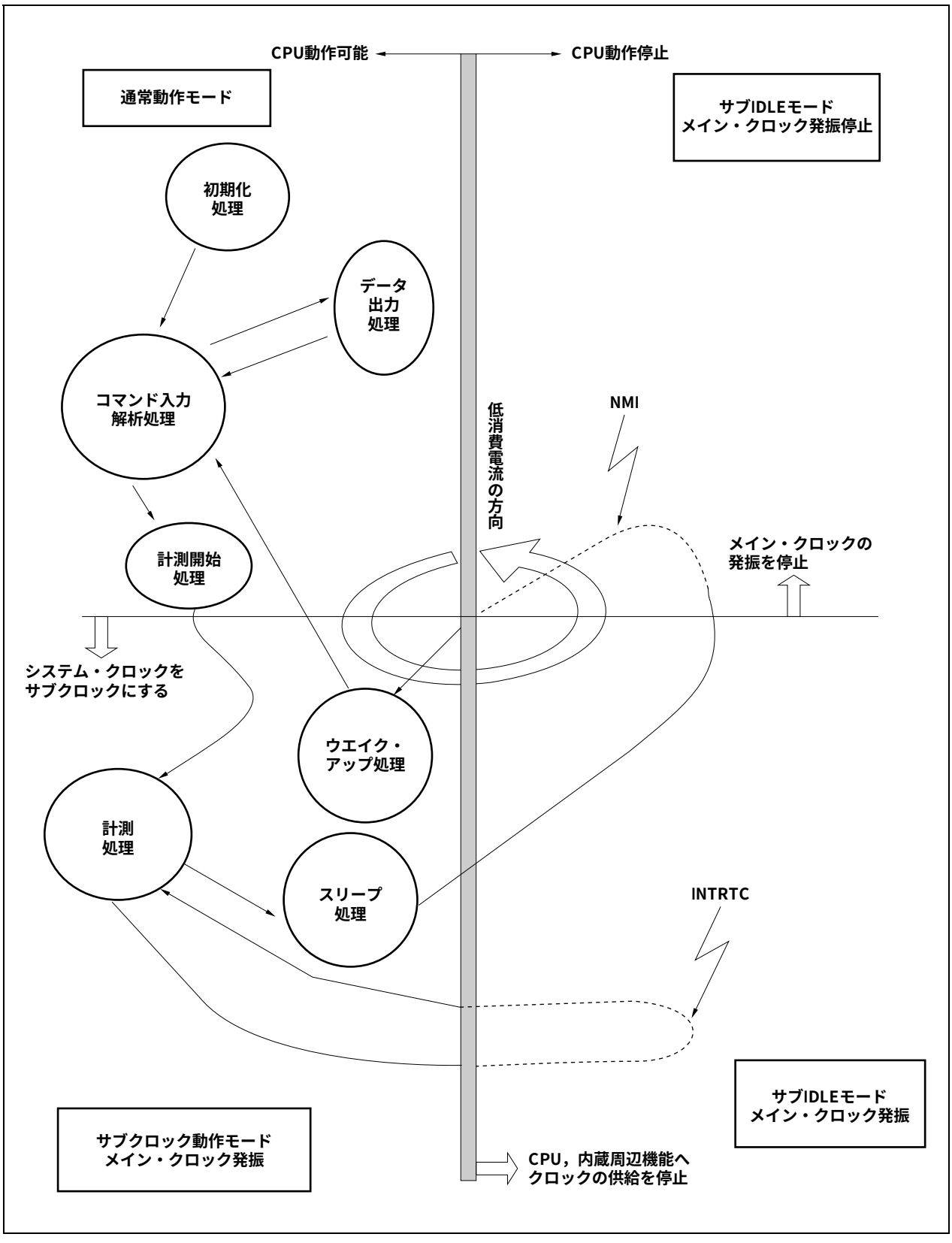
- ID850 Ver.2.50　統合ディバッガ　操作編（U16217J）

図4－4　開発ツールの構成



4. 3. 2　プログラム構成とCPU動作モード

図4－5　プログラム構成とCPU動作モード



(1) 構造の説明

(i) 図4－5の各処理はメイン・プログラムでシーケンシャルに処理されます。

(ii) 割り込み処理は次のようになります。

- nmi（スイッチ入力）処理で停止していたCPUを動作させ，ウエイク・アップ処理に入ります。
- int_rtc（リアルタイム・カウンタ割り込み）処理で停止していたCPUを動作させ，計測処理に入ります。
- int_p01（INTP1外部割り込み）処理で低電圧検出フラグをセット（1）します。

(2) CPU動作モード

図4－5の4つの動作モードがあります。

図の左側はCPUが動作しています。右側は停止状態です。

4つの領域は左上の通常動作モードから反時計まわりに低消費電流になります。

各モードにおけるCPUおよび内蔵周辺機能の動作状態を次に示します。

表4－1　サブクロック動作モード時の動作状態

項目 ＼ サブクロック動作モードの設定		動作状態	
		メイン・クロック発振時	メイン・クロック停止時
サブクロック発振回路		発振可能	
CPU		動作可能	
DMA		動作可能	
割り込みコントローラ		動作可能	
ROMコレクション		動作可能	
16ビット・タイマ／イベント・カウンタ（TM0, TM1）		動作停止	
8ビット・タイマ／イベント・カウンタ（TM2-TM5）		動作停止	
リアルタイム・カウンタ		動作可能	カウント・クロックにf_{XT}選択時に動作可能
ウォッチドッグ・タイマ		動作停止	
シリアル・インタフェース	CSI0-CSI4	動作可能	動作クロックに$\overline{SCKn}$入力クロック選択時，動作可能（n = 0-4）
	I^2C注	動作可能	動作停止
	UART0, UART1	動作可能	動作停止
A/Dコンバータ		動作可能	動作停止
D/Aコンバータ		動作可能	動作停止
外部バス・インタフェース		動作可能	
ポート		設定可能	
内部データ		設定可能	

注　I^2CはμPD703200Y, 703201Y, 703204Y, 70F3201Y, 70F3204Yのみ。

表4－2　サブIDLEモード時の動作状態

サブIDLEモードの設定 / 項　目		動作状態	
		メイン・クロック発振時	メイン・クロック停止時
サブクロック発振回路		発振可能	
CPU		動作停止	
DMA		動作停止	
割り込みコントローラ		動作停止	
ROMコレクション		動作停止	
16ビット・タイマ／イベント・カウンタ（TM0, TM1）		動作停止	
8ビット・タイマ／イベント・カウンタ（TM2-TM5）		動作停止	
リアルタイム・カウンタ		動作可能	カウント・クロックにf_{XT}選択時に動作可能
ウォッチドッグ・タイマ		動作停止	
シリアル・インタフェース	CSI0-CSI4	動作クロックに$\overline{SCKn}$入力クロック選択時，動作可能（n = 0-4）	
	I^2C注	動作停止	
	UART0, UART1	動作停止	
A/Dコンバータ		動作停止	
D/Aコンバータ		動作停止	
外部バス・インタフェース		IDLEモード時と同じ	
ポート		サブIDLEモード設定前の状態を保持	
内部データ		CPUのレジスタ，ステータス，データ，内蔵RAMなどの内部データはすべてサブIDLEモード設定前の状態を保持	

注　I^2CはμPD703200Y, 703201Y, 703204Y, 70F3201Y, 70F3204Yのみ。

V850ES/SA2のCPU動作モードにおいて，実行されるこのプログラムの各処理と動作電流比を表4－3にあらわします。

表4－3　各動作と電流比の関係

V850ES/SA2動作モード		システム動作	通常動作を基準にした各モードの動作電流比イメージ[注]
通常動作		初期化処理，コマンド入力解析処理，計測開始処理，データ出力処理	1
HALT		－	1/2
IDLE		－	1/10
STOP	サブクロック動作	－	1/4000
	サブクロック停止	－	1/15000
サブクロック動作		計測処理 スリープ処理 ウエイク・アップ処理	1/400
サブIDLE	メイン・クロック動作	計測待機（INTRTC待ち）	1/10
	メイン・クロック停止	起床待機（NMI待ち）	1/15000

注　マスクROM搭載品

4. 3. 3　入出力データ形式とコマンド

入出力データ形式はすべてHEX-ASCII形式とします。

図4－6　入出力データ形式

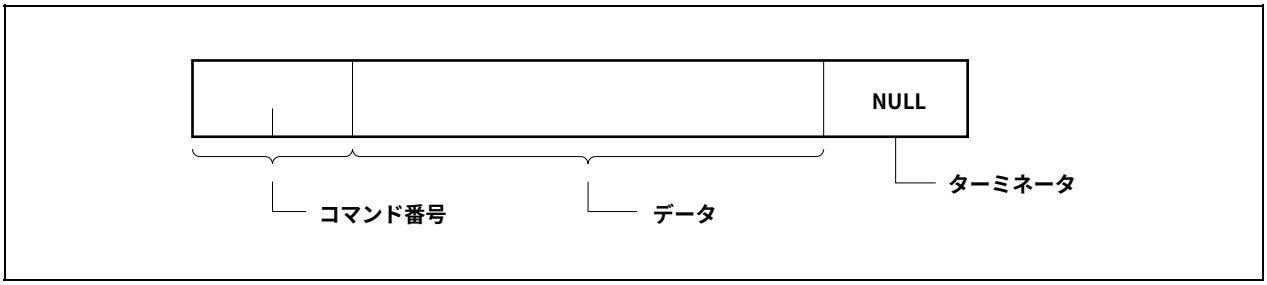


図4－7　計測コマンド

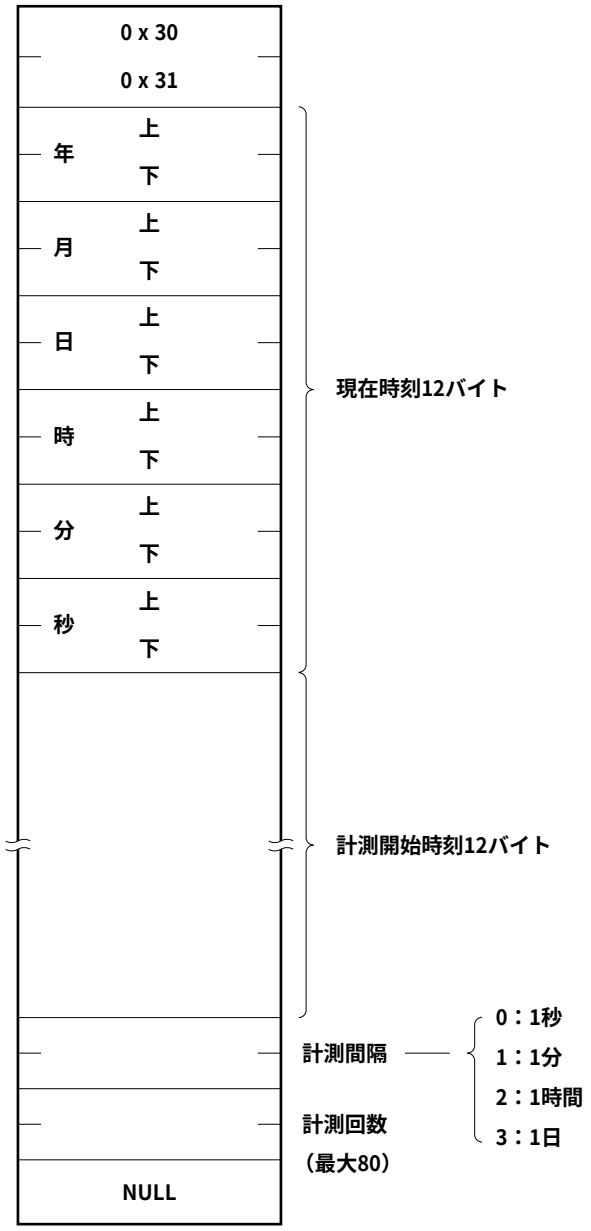

0 x 30
0 x 31
年
月
日
時
分
秒
上
下
現在時刻12バイト
計測開始時刻12バイト
計測間隔
計測回数
（最大80）
0：1秒
1：1分
2：1時間
3：1日
NULL

図4－8　計測データ出力コマンド

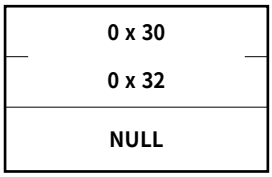
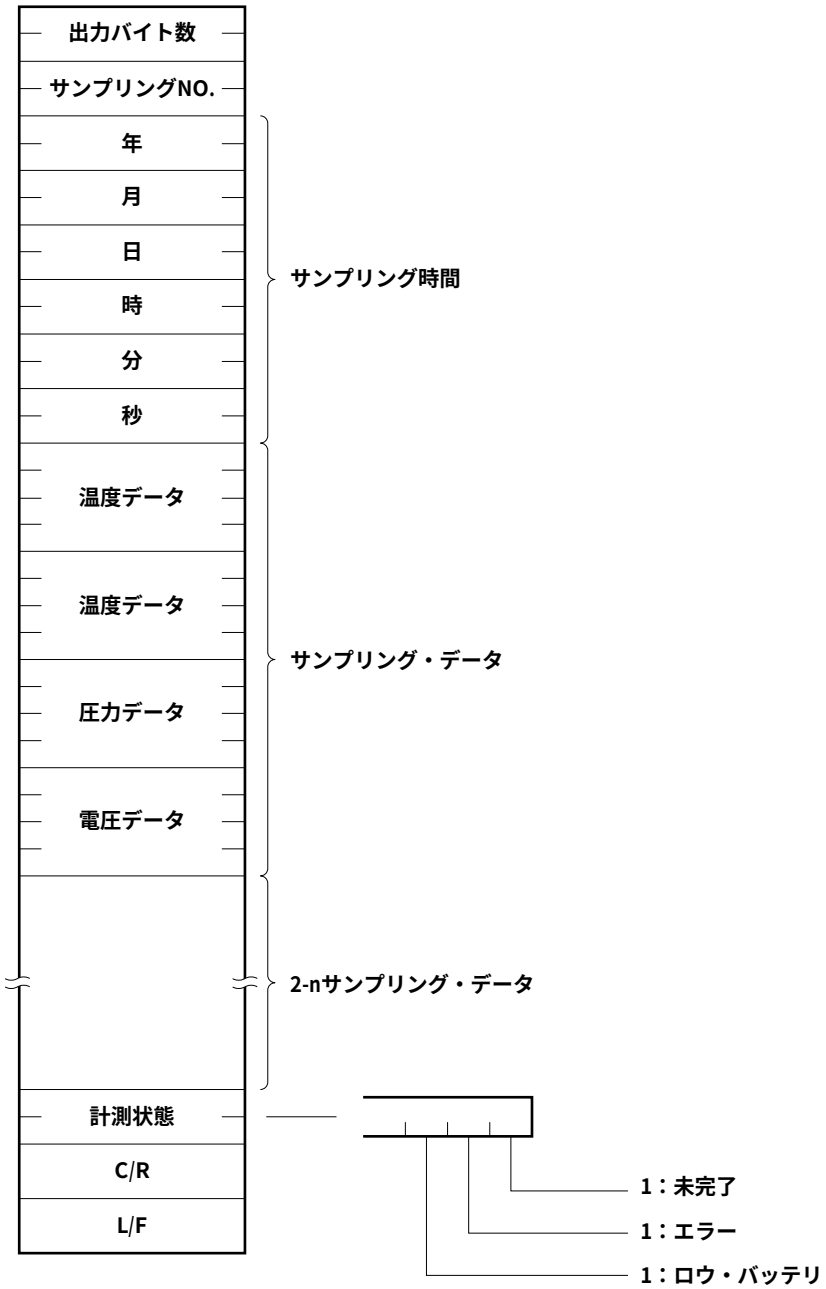


図4－9　計測データ出力形式（すべてHEX-ASCII）



4. 3. 4　フロー・チャート

図4－10　メイン・ルーチン

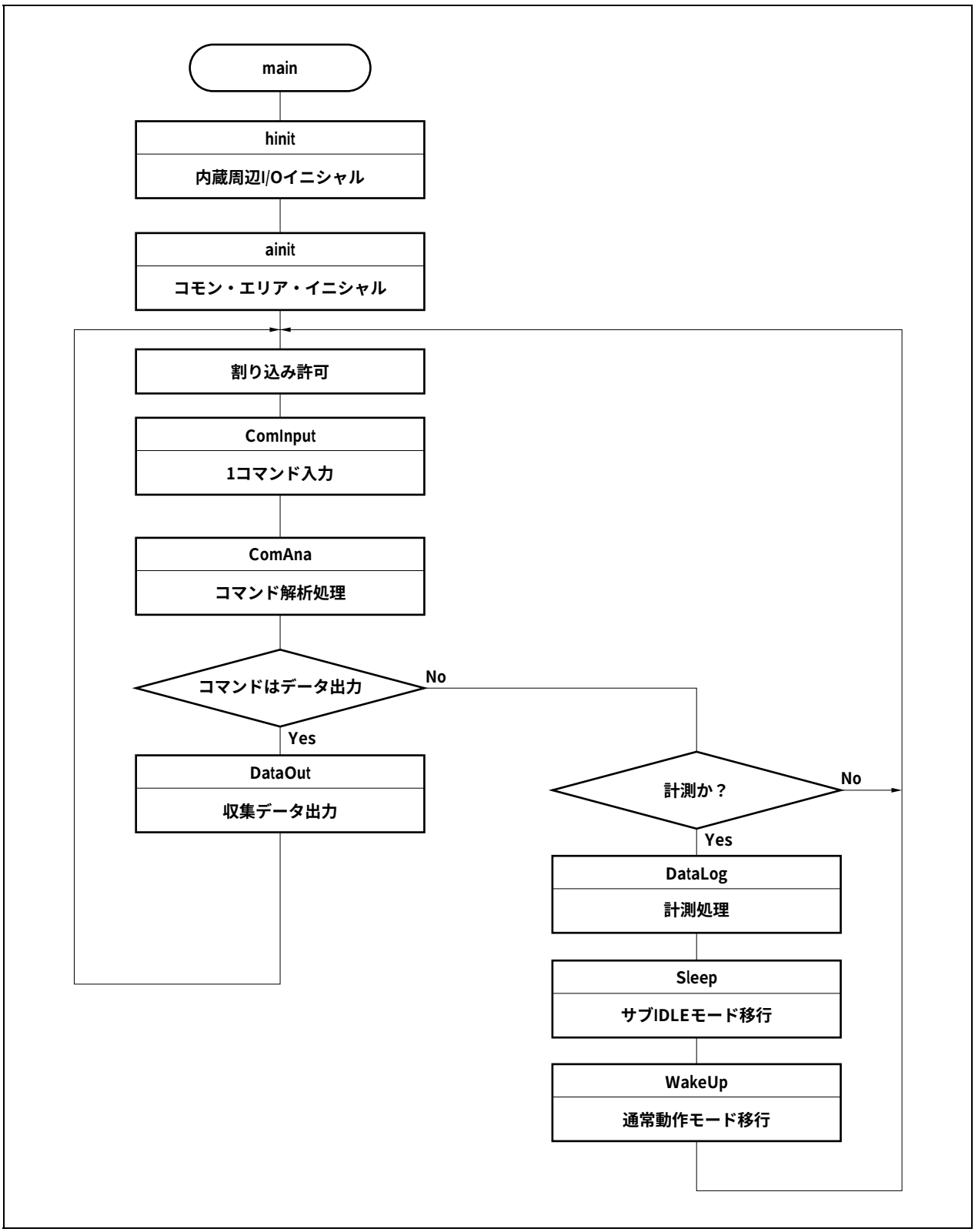


図4－11　NMI割り込み処理

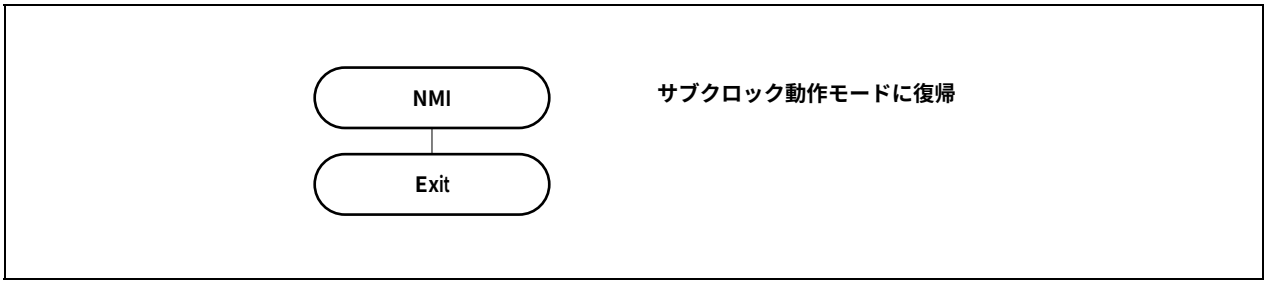


図4－12　リアルタイム・カウンタ割り込み処理

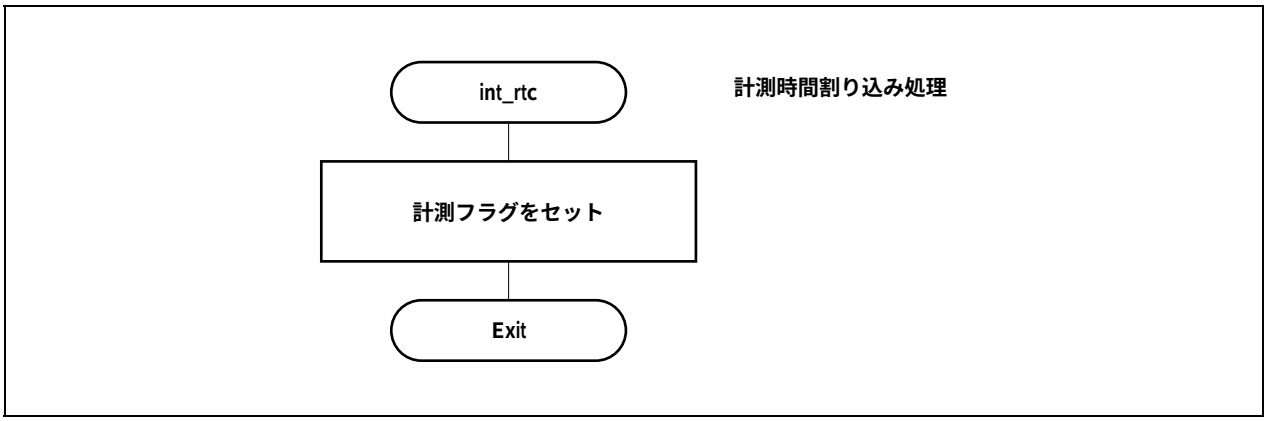


図4－13　INTP01外部割り込み処理

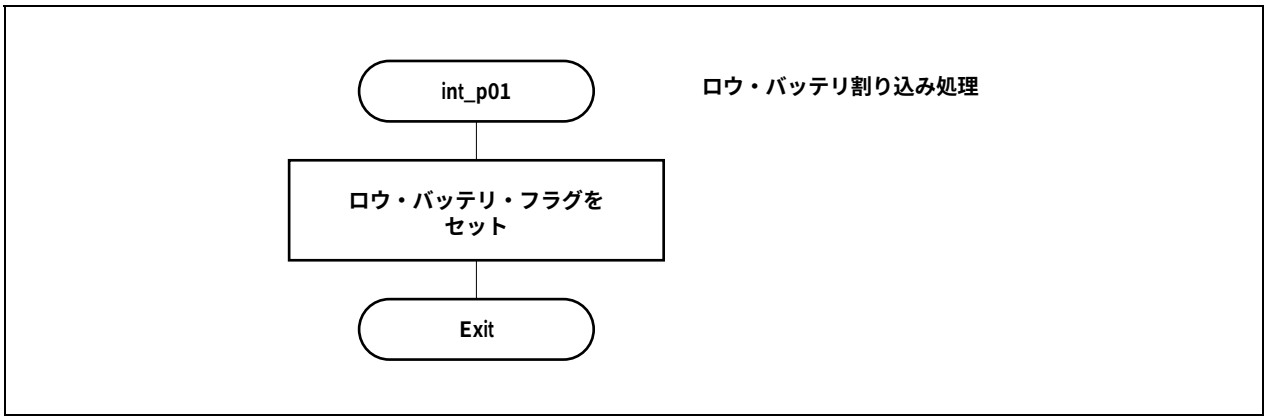


図4－14　内蔵周辺I/Oイニシャルライズ処理

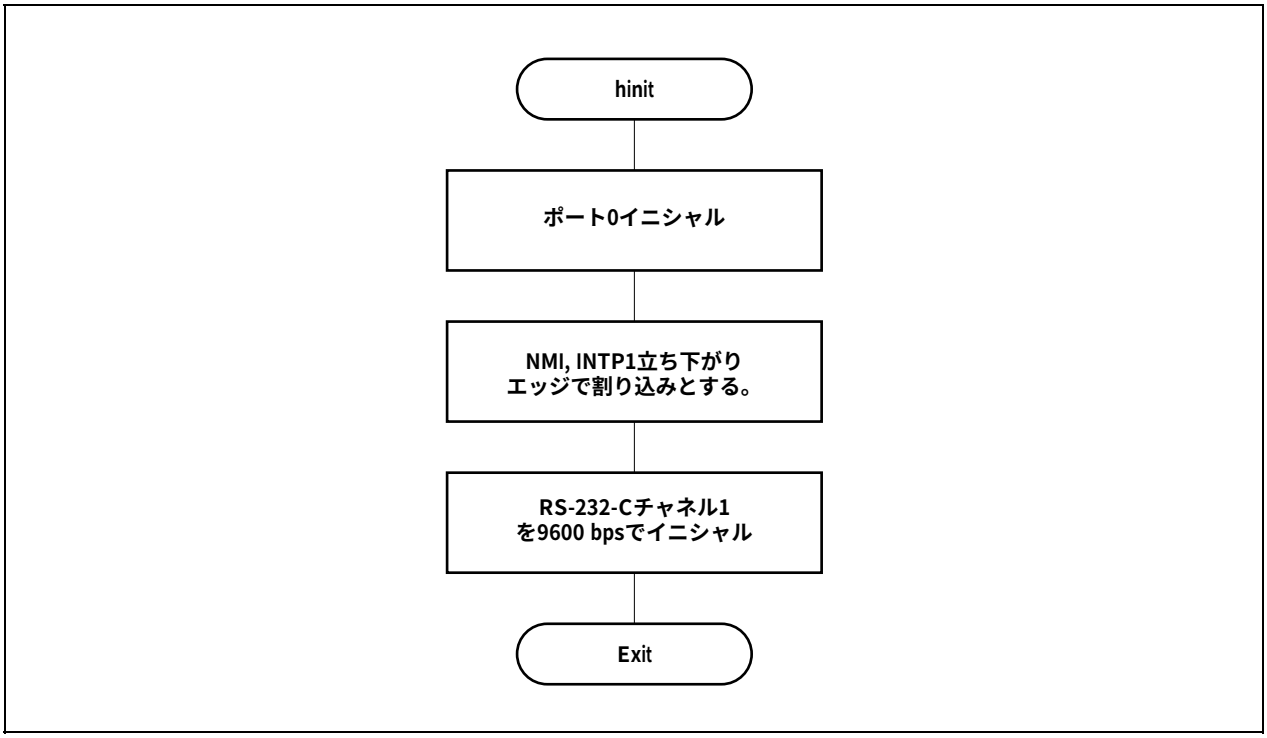


図4－15　コモン・エリア・イニシャライズ処理

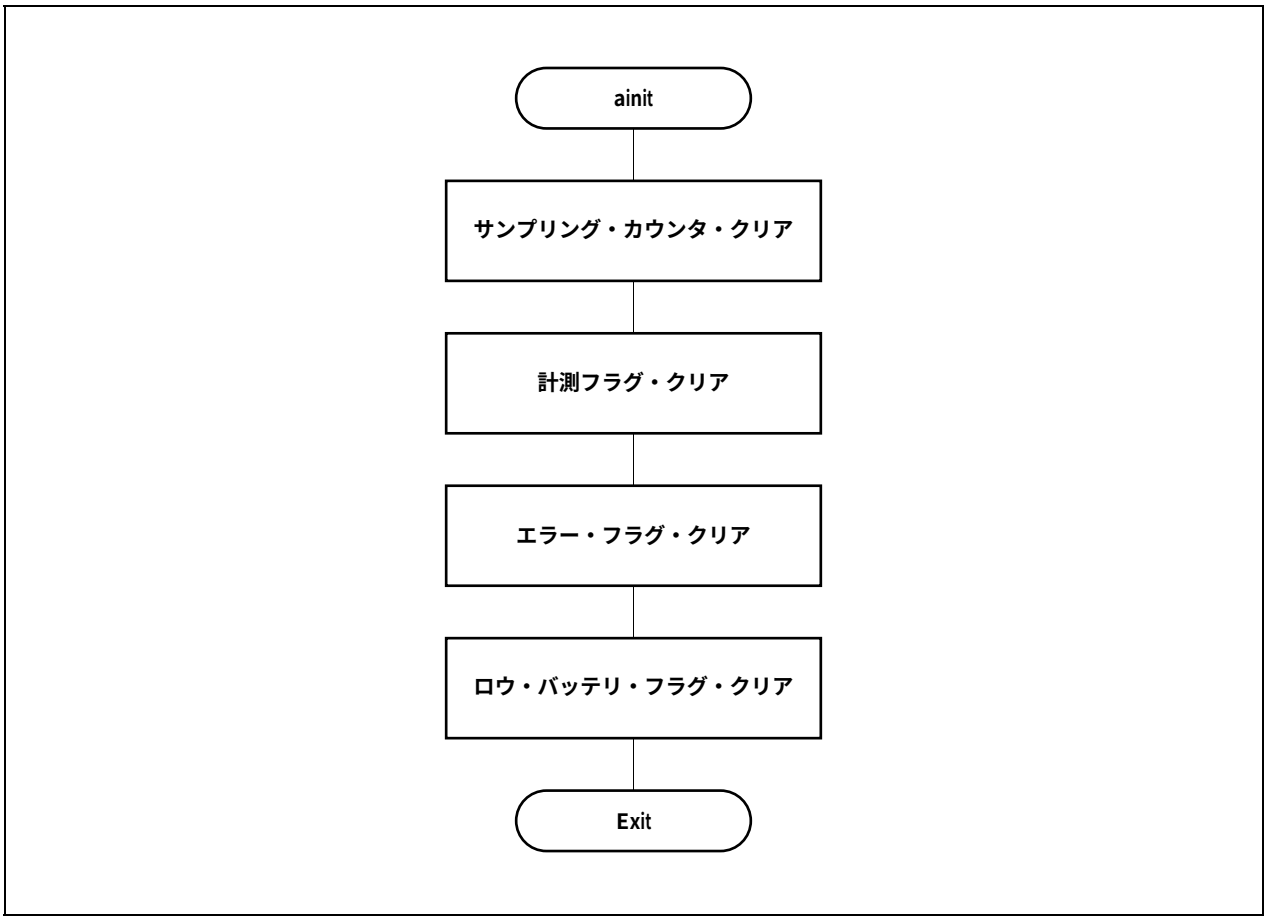


図4－16　コマンド入力処理

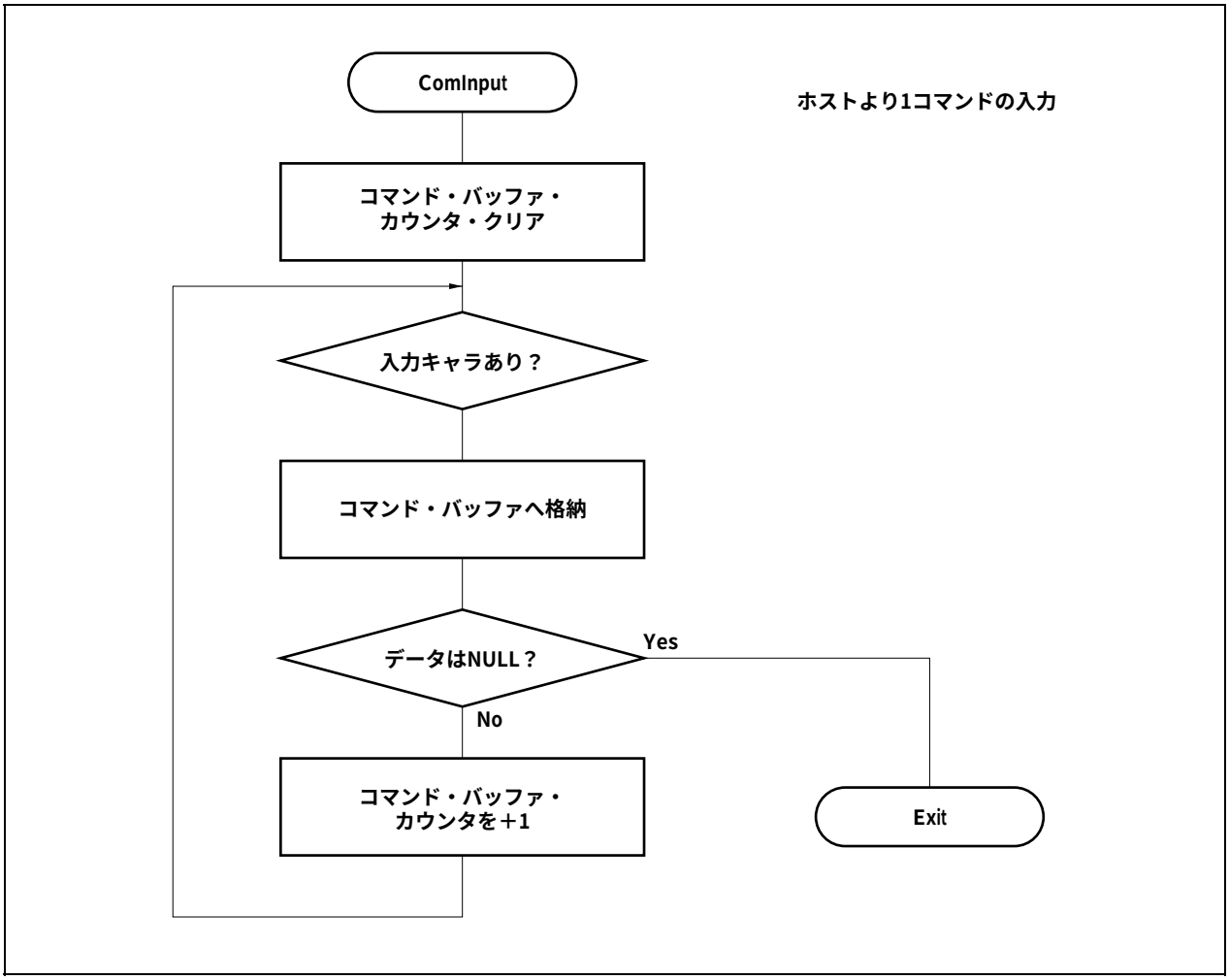


図4－17　コマンド解析処理（1/3）

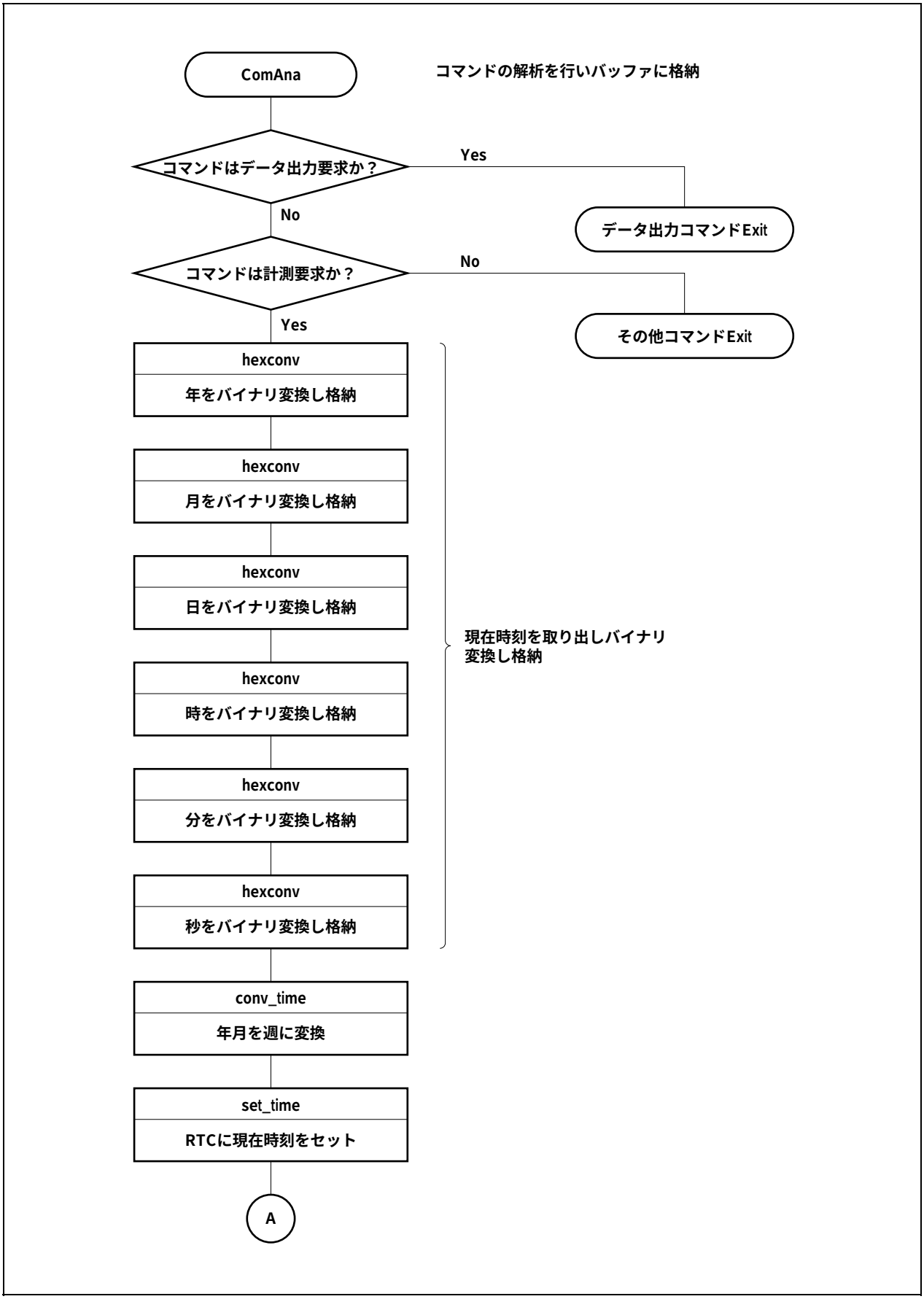


図4－17　コマンド解析処理（2/3）

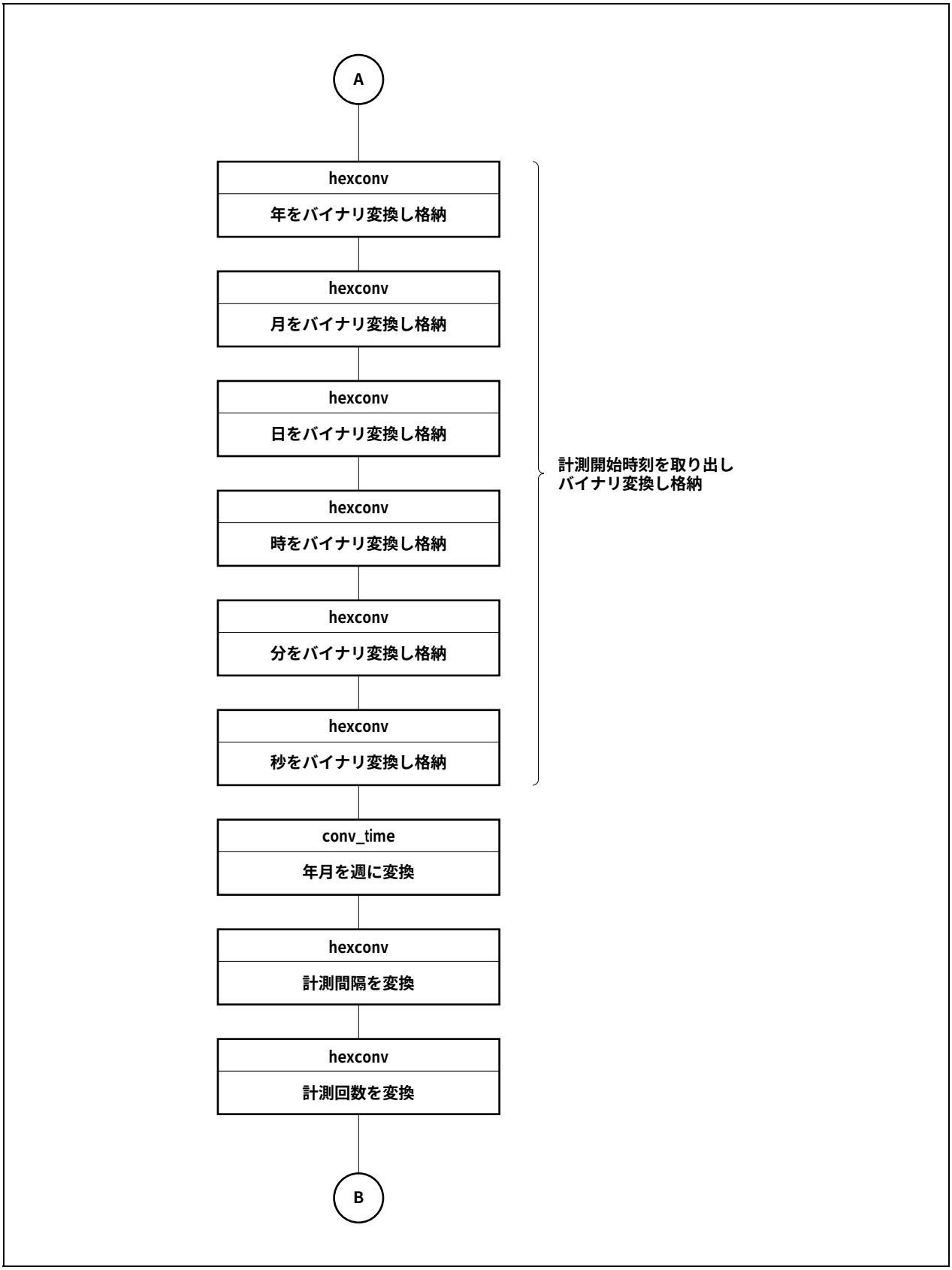


図4－17　コマンド解析処理（3/3）

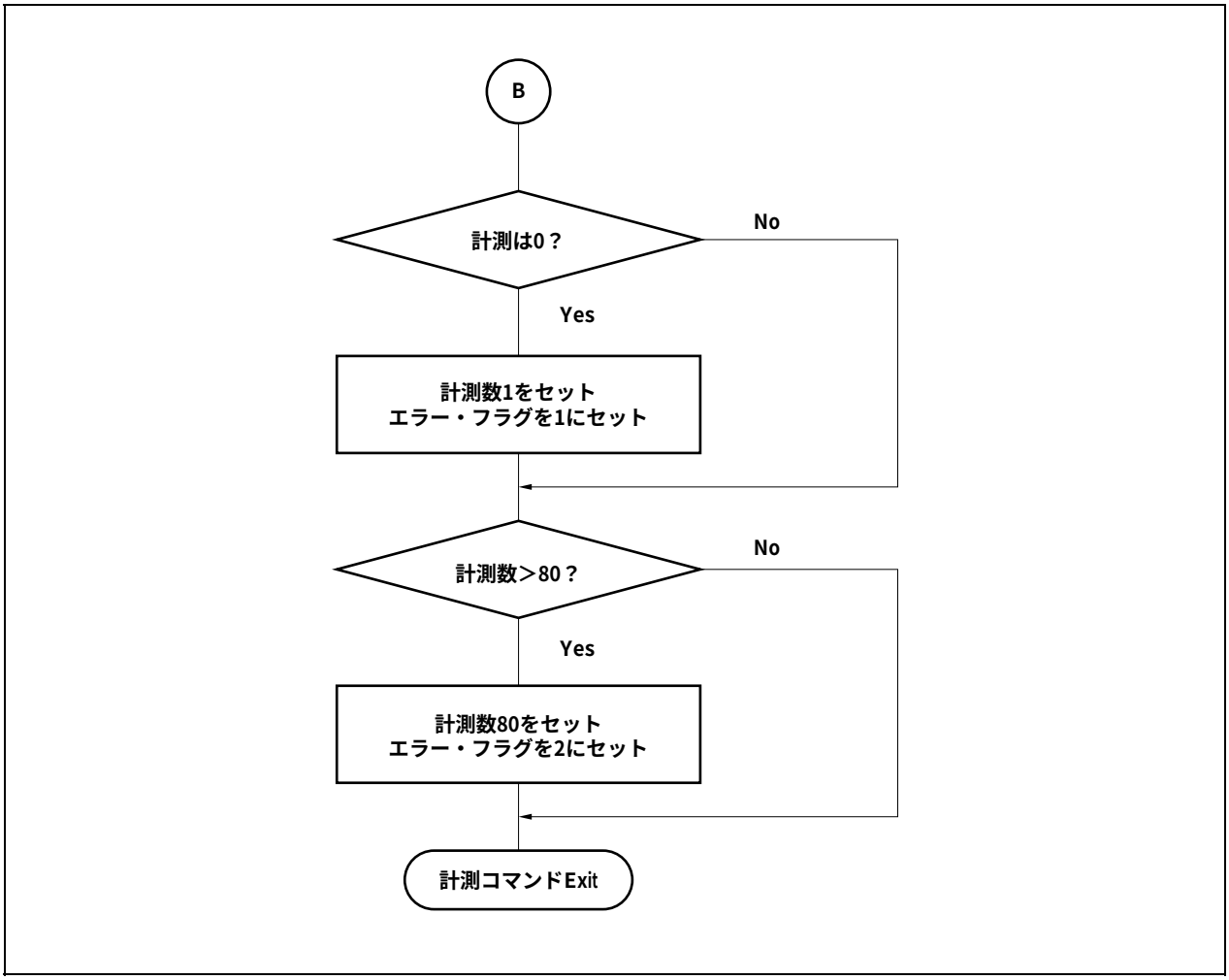


図4－18　計測データ出力処理

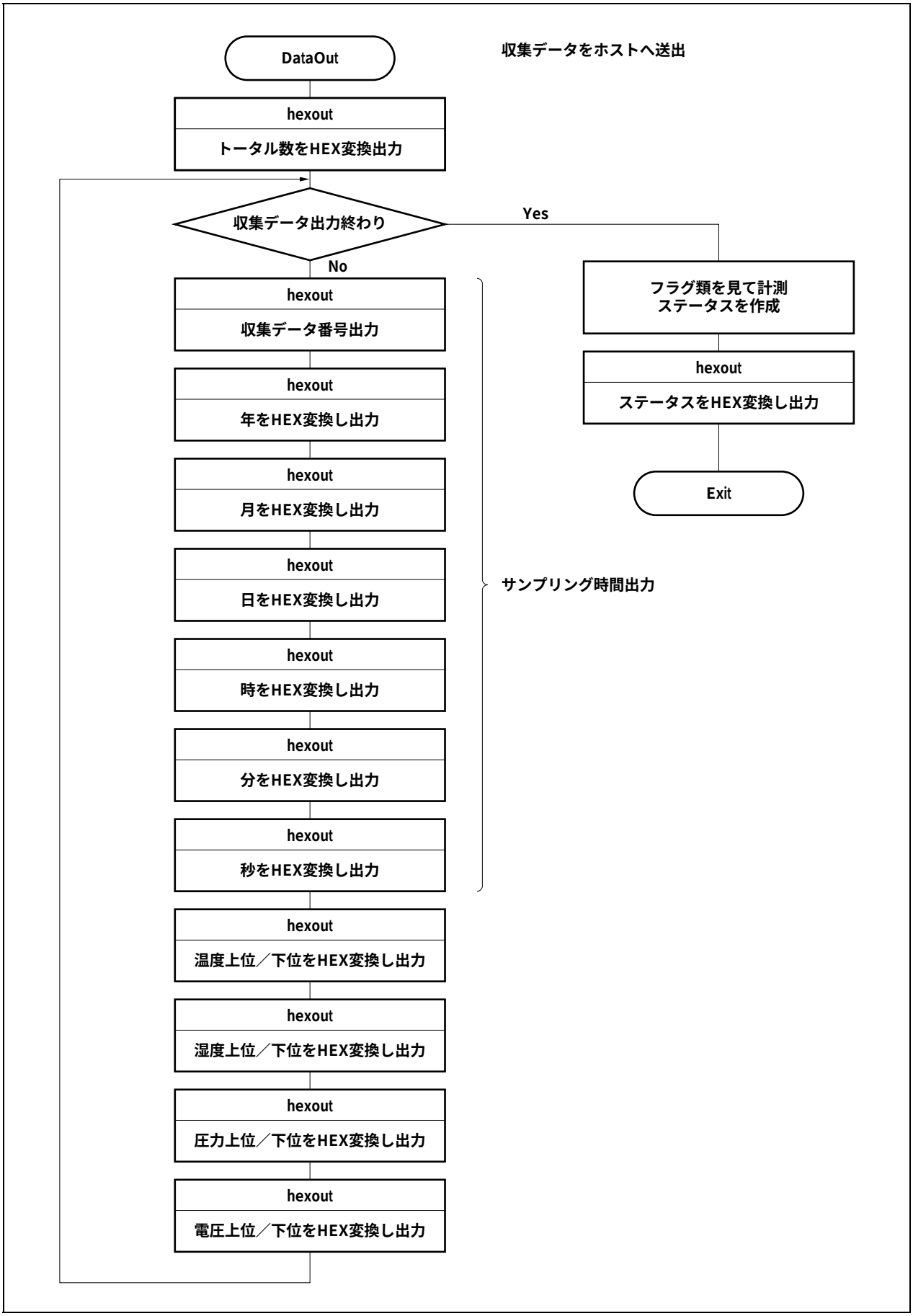


図4－19　計測処理

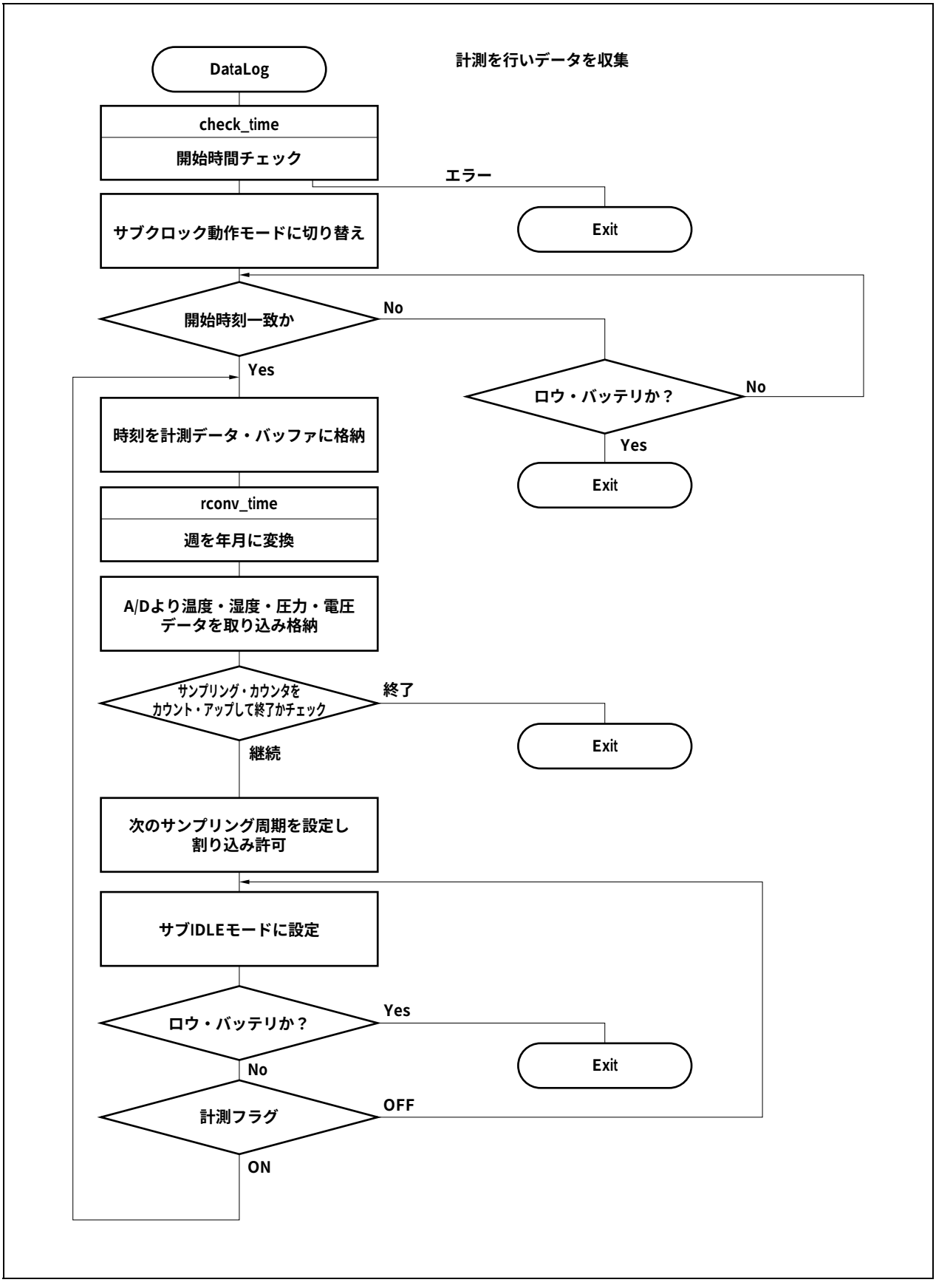

DataLog
計測を行いデータを収集
check_time
開始時間チェック
エラー
Exit
サブクロック動作モードに切り替え
開始時刻一致か
No
Yes
ロウ・バッテリか？
No
Yes
Exit
時刻を計測データ・バッファに格納
rconv_time
週を年月に変換
A/Dより温度・湿度・圧力・電圧
データを取り込み格納
サンプリング・カウンタを
カウント・アップして終了かチェック
終了
Exit
継続
次のサンプリング周期を設定し
割り込み許可
サブIDLEモードに設定
ロウ・バッテリか？
Yes
Exit
No
計測フラグ
OFF
ON

図4－20　**スリープ処理**

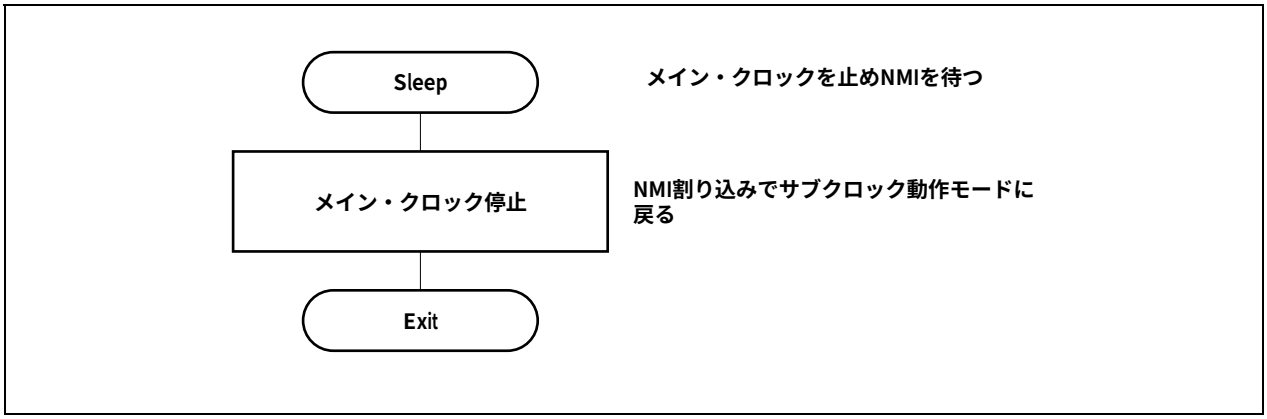


図4－21　**ウエイク・アップ処理**

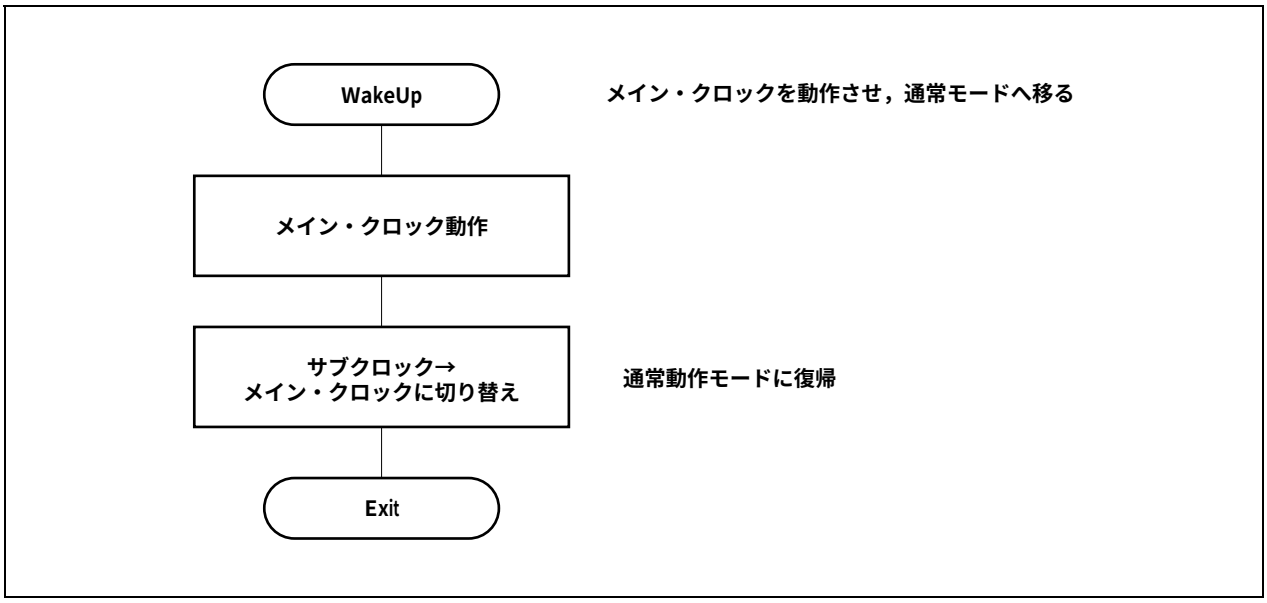


図4－22　HEX-ASCII Binary 変換処理

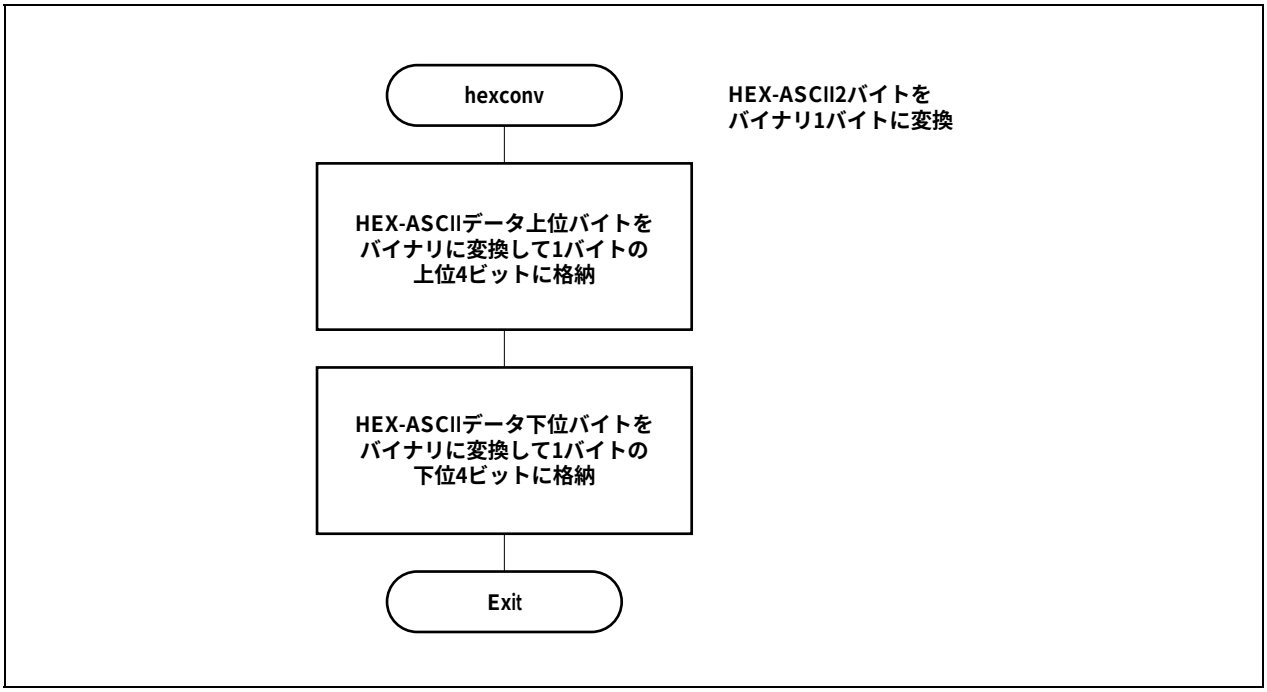


図4－23　Binary HEX-ASCII変換と出力処置

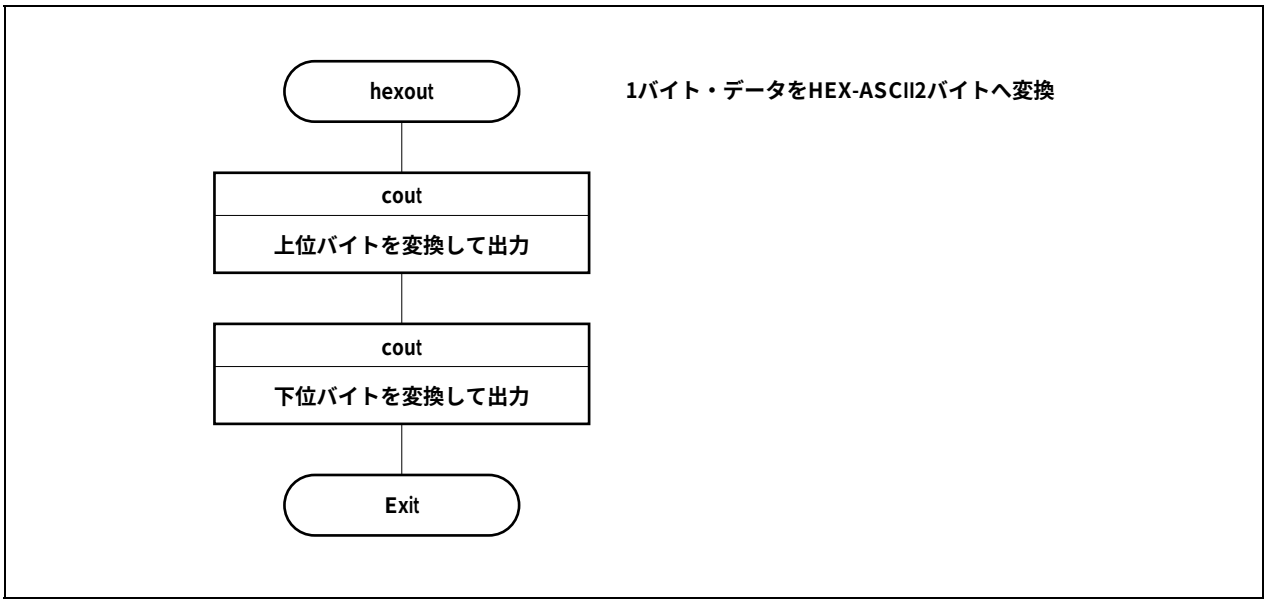


図4－24　RS-232-C出力処理

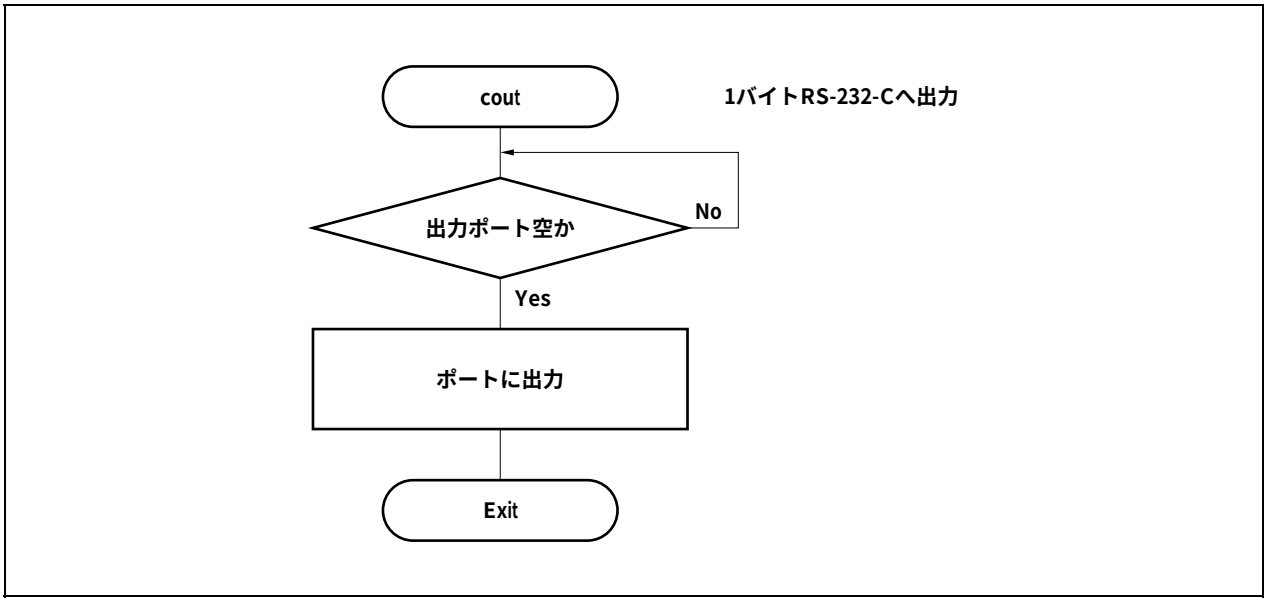


図4－25　リアルタイム・カウンタ時刻設定処理

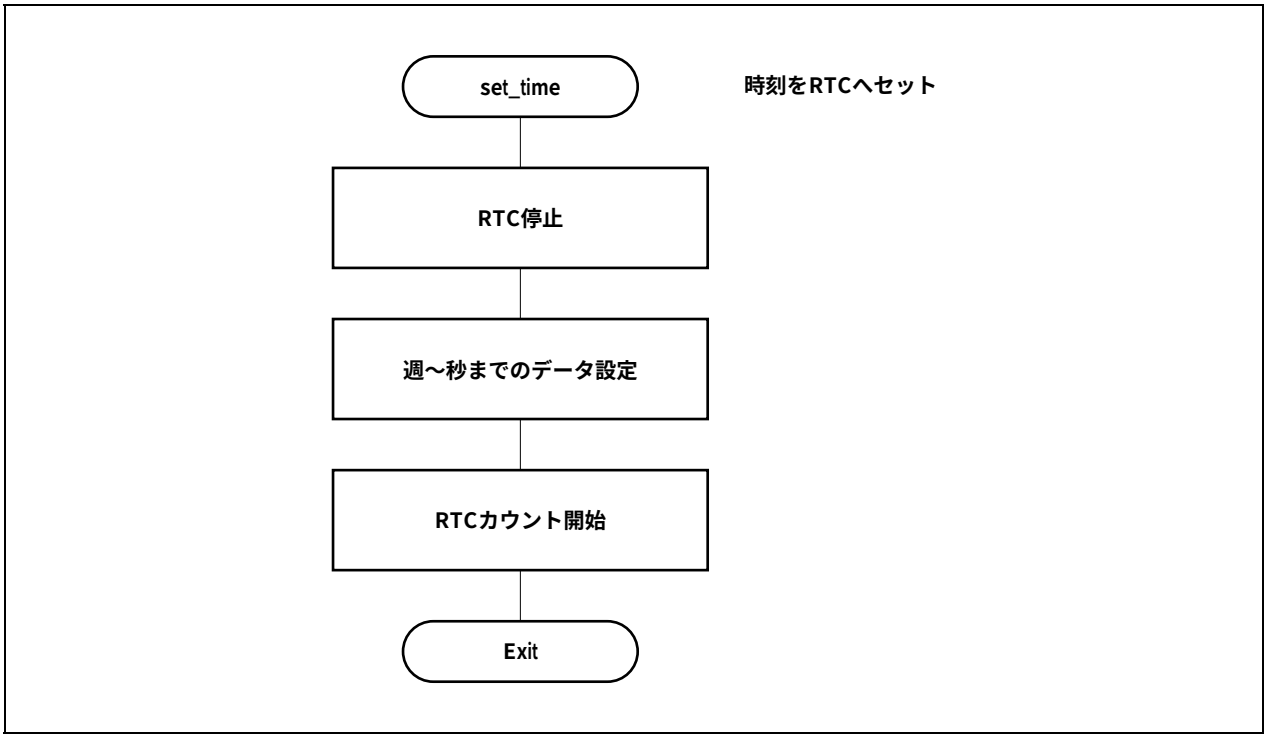


図4－26　年月データ週変換処理

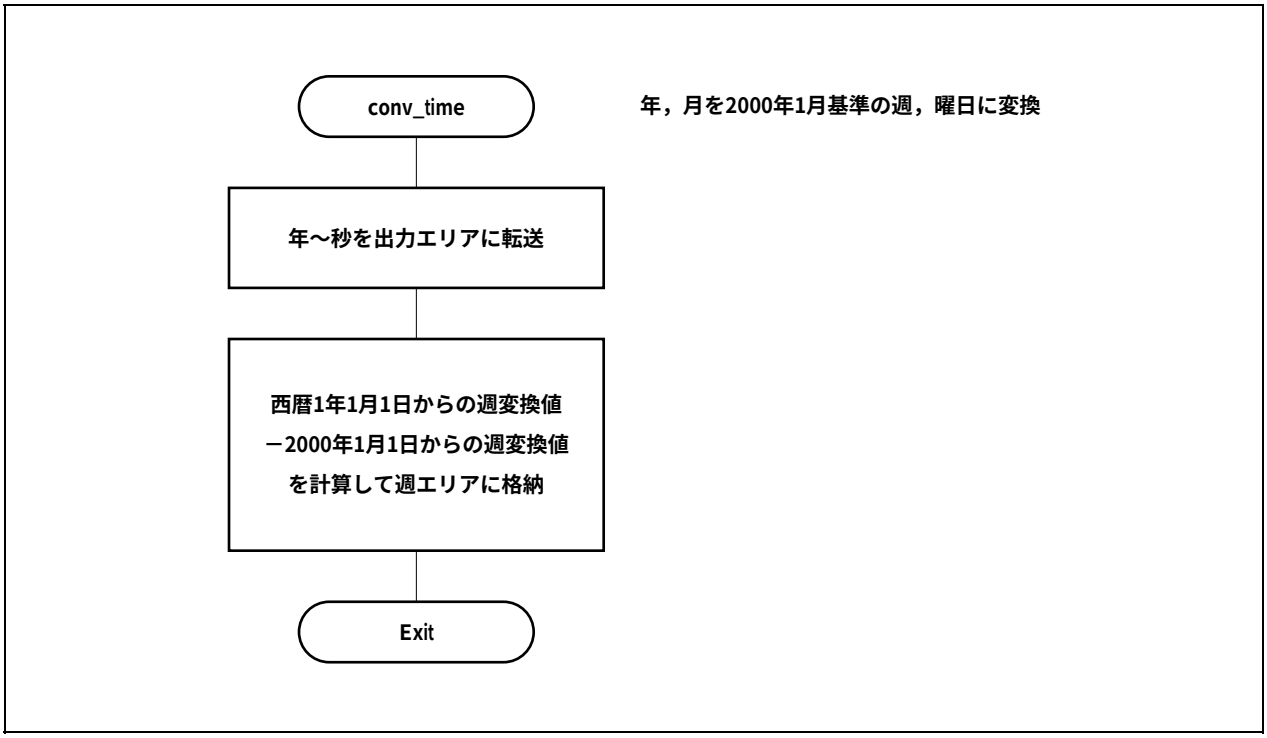


図4－27　週データ年月変換処理

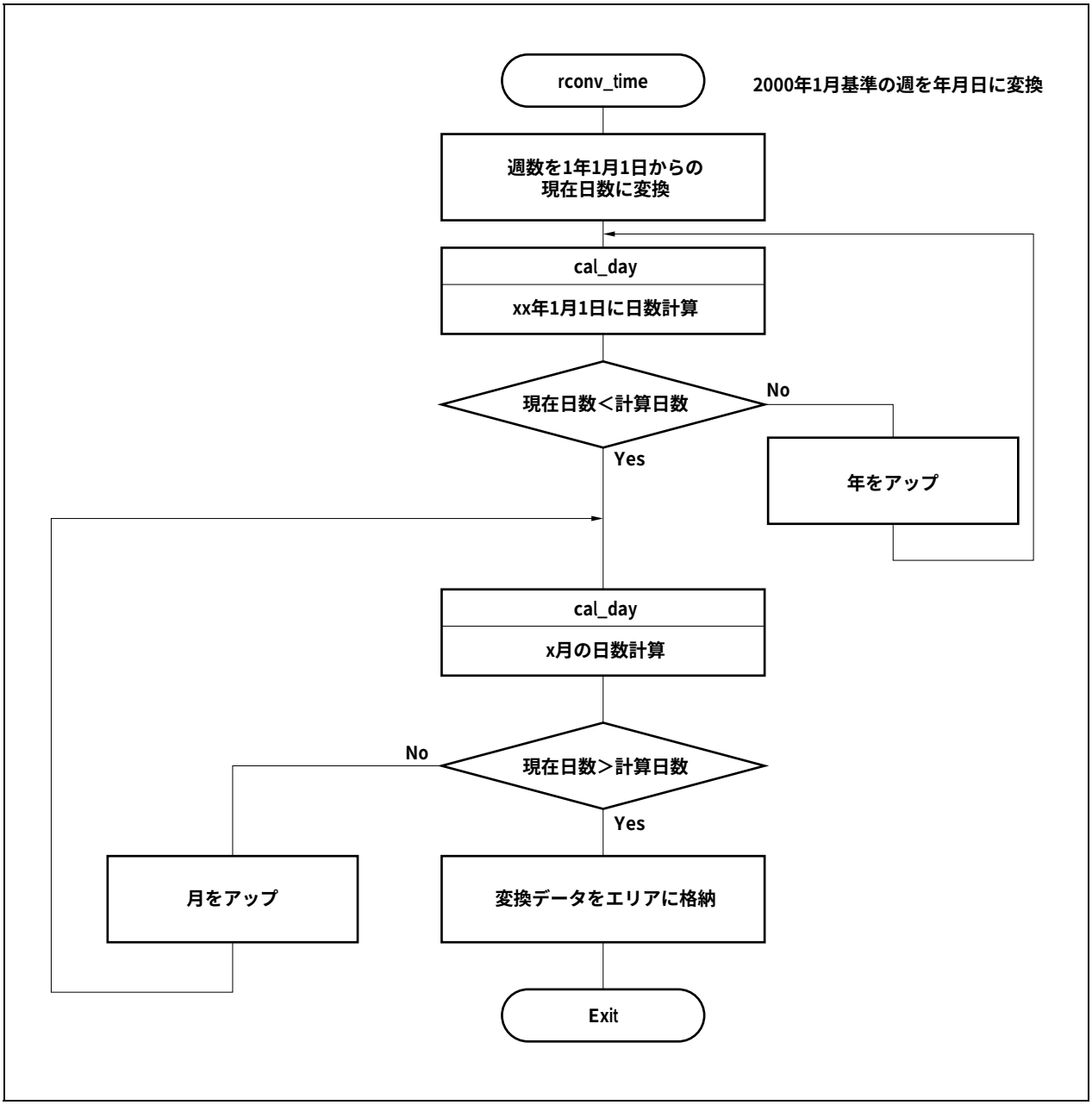


図4－28　計測開始時刻処理

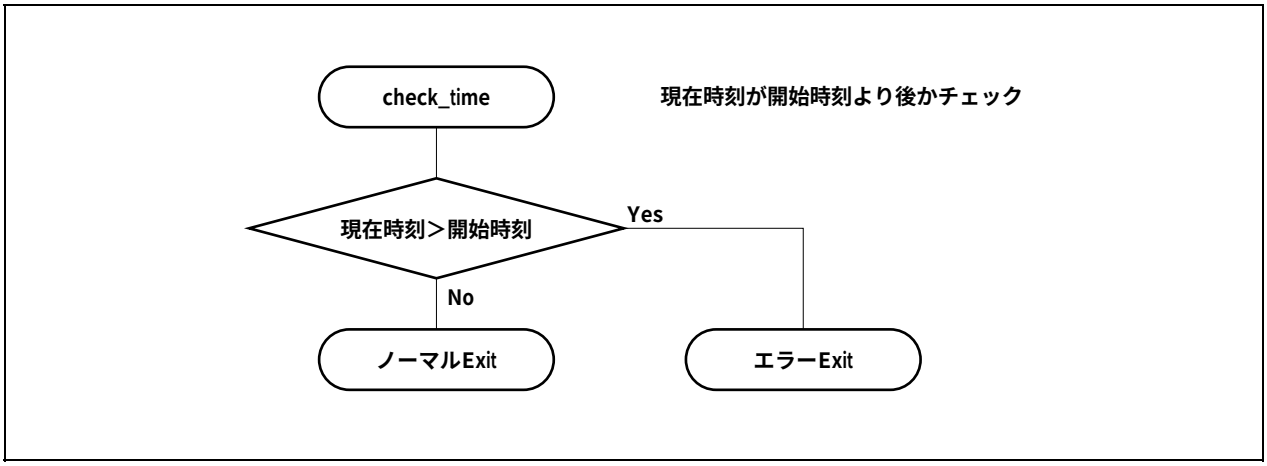
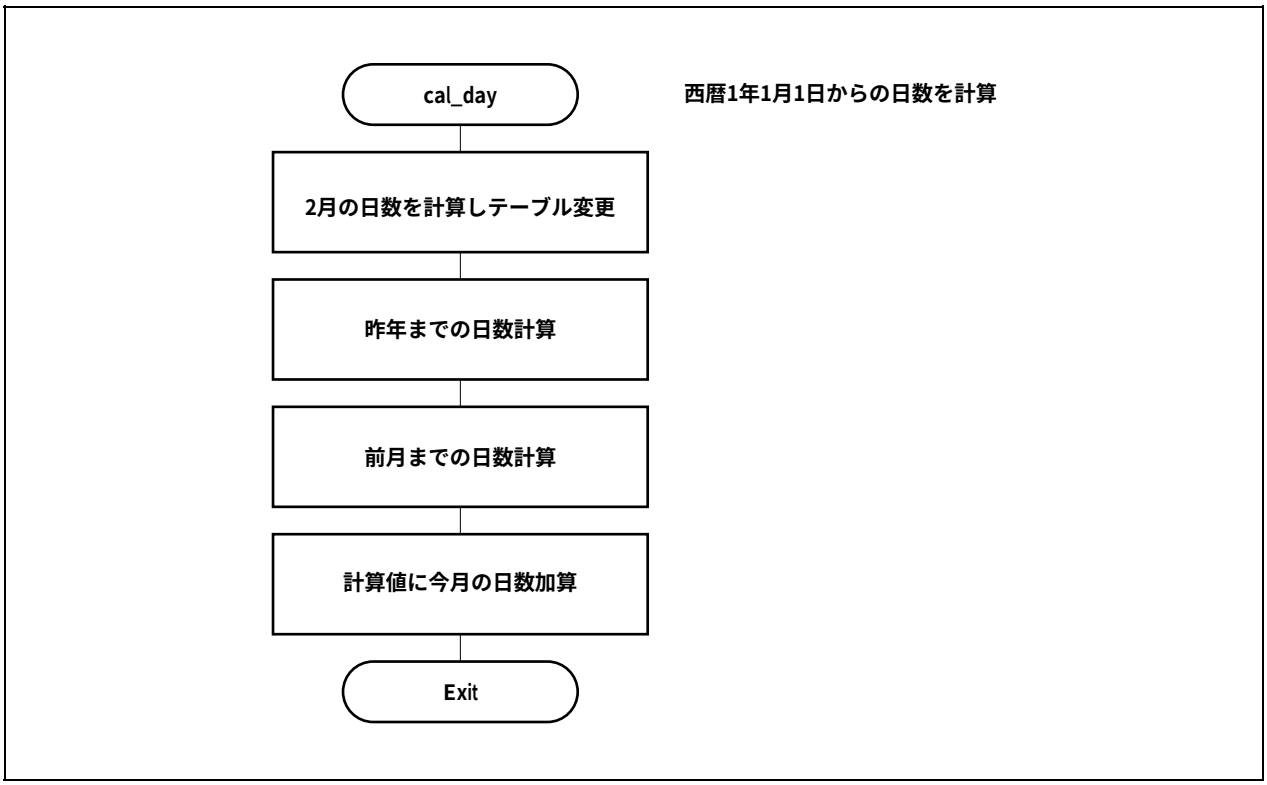


図4－29　年月日処理



4. 3. 5　プログラム・リスト

(1) 定数定義

```
#include   <stddef.h>
#pragma    ioreg                          //内蔵周辺I/Oレジスタ定義
////////////////////////////////////////////////////////////////////////////////
//       定数定義
////////////////////////////////////////////////////////////////////////////////
#define MEASURE    1                      //計測コマンド番号
#define DATAOUT    2                      //データ出力コマンド番号
typedef struct time {                     //時刻データ構成
    short  year;
    short  month;
    short  week;
    short  day;
    short  hour;
    short  minute;
    short  second;
} TIME;
```

(2) コモン・エリア

```
017  ////////////////////////////////////////////////////////////////////////////
018  //コモン・エリア
019  ////////////////////////////////////////////////////////////////////////////
020  char   in_buf[40];                          //コマンド・バッファ
021  short  in_co;                               //コマンド・バッファ・カウンタ
022  short  sample_co;                           //サンプリング・カウンタ
023  int    measure_interval;                    //計測間隔
024  short  measure_co;                          //計測数
025  short  measure_flag;                        //計測フラグ
026  short  low_flag;                            //ロウ・バッテリ・フラグ
027  short  error_flag;                          //計測データ・オーバ
028  TIME   now_time;                            //現在時刻
029  TIME   start_time;                          //計測開始時刻
030  struct log {                                //計測データ・バッファ
031      TIME  time;
032      short  adi[4];
033  } log_buf[80];
```

(3) 関　数

```
034  ////////////////////////////////////////////////////////////////////////////
035  //     関数
036  ////////////////////////////////////////////////////////////////////////////
037  void   hinit( void );
038  void   ainit( void );
039  void   ComInput( void );
040  int    ComAna( void );
041  void   DataOut( void );
042  void   DataLog( void );
043  void   Sleep( void );
044  void   WakeUp( void );
045  void   hexconv( unsigned char *, unsigned char * );
046  void   cout( unsigned char );
047  void   hexout( unsigned char );
048  void   set_time( TIME * );
049  void   conv_time( TIME *, TIME * );
050  void   rconv_time( TIME *, TIME * );
051  int    check_time( TIME *, TIME * );
052  int    cal_day( TIME * );
```

(4) メイン・ルーチン

```
053  ////////////////////////////////////////////////////////////////////////////
054  //      メイン・ルーチン
055  ////////////////////////////////////////////////////////////////////////////
056  main()
057  {
058  short   Cno;
059  /*  */
060      hinit();                                //周辺I/Oイニシャル
061      ainit();                                //コモン・エリア・イニシャル
062          __EI();                             //割り込み許可
063
064      while( 1 ) {
065          ComInput();                         //1コマンド入力
066          Cno = ComAna();                     //コマンド解析処理
067          if  ( Cno == DATAOUT ) {
068              DataOut();                      //収集データ出力
069          } else if  ( Cno == MEASURE ) {
070              DataLog();                      //計測処理
071              Sleep();                        //サブIDLEモード移行
072              WakeUp();                       //通常動作モード移行
073          }
074      }
075  }
```

(5) NMI割り込み処理

```
076  ////////////////////////////////////////////////////////////////////////////
077  //  名称       : NMI
078  //  形式       : void nmi()
079  //
080  //  機能説明    : サブクロック動作モードに復帰
081  //  引き数説明  : なし
082  //  戻り値      : なし
083  ////////////////////////////////////////////////////////////////////////////
084  #pragma interrupt NMI nmi
085  __interrupt
086  void   nmi( void )
087  {
088
089
090  }
```

(6) リアルタイム・カウンタ割り込み処理

```
091  ////////////////////////////////////////////////////////////////////////////
092  //  名称       : int_rtc
093  //  形式       : void int_rtc()
094  //
095  //  機能説明    : 計測時間割り込み処理
096  //  引き数説明  : なし
097  //  戻り値      : なし
098  ////////////////////////////////////////////////////////////////////////////
099  #pragma interrupt INTRTC int_rtc
100  __interrupt
101  void     int_rtc( void )
102  {
103       measure_flag = 1;                    //計測フラグをセット
104  }
```

（7）INTP01外部割込み処理

```
105  ////////////////////////////////////////////////////////////////////////////
106  //  名称       : int_p01
107  //  形式       : void int_p01()
108  //
109  //  機能説明    : ロウ・バッテリ割り込み処理
110  //  引き数説明  : なし
111  //  戻り値      : なし
112  ////////////////////////////////////////////////////////////////////////////
113  #pragma interrupt INTP1 int_p01
114  __interrupt
115  void     int_p01( void )
116  {
117       low_flag = 1;                          //ロウ・バッテリ・フラグをセット
118  }
```

(8) 内蔵周辺I/Oイニシャライズ処理

```
119  ////////////////////////////////////////////////////////////////////////////
120  //  名称       : hinit
121  //  形式       : void hinit()
122  //
123  //  機能説明   : 内蔵周辺I/Oの初期化を行う。
124  //  引き数説明 : なし
125  //  戻り値     : なし
126  ////////////////////////////////////////////////////////////////////////////
127  void    hinit( void )
128  {
129  int     *po;
130  /*      */
131
132      PMC0  = 0x05;
133      PM0   = 0xff;
134      PU0   = 0x3f;
135      INTF0 = 0x05;
136      INTR0 = 0x00;                          //NMI,INTP1立ち下がりエッジ割り込み
137      PIC1  = 0x03;                          //割り込み許可
138
139      CKSR1 = 0x03;
140      BRGC1 = 130;                           //9600 bps
141      ASIM1 = 0xe5;                          //8ビット  NONパリティ  ストップ1
142      PFC9  = 0x0300;                        //UART1設定
143      PMC9  = 0x0300;
144
145  }
```

(9) コモン・エリア・イニシャライズ処理

```
146  ////////////////////////////////////////////////////////////////////////////
147  //  名称       : ainit
148  //  形式       : void ainit()
149  //
150  //  機能説明   : 使用コモン・エリアの初期化を行う。
151  //  引き数説明 : なし
152  //  戻り値     : なし
153  ////////////////////////////////////////////////////////////////////////////
154  void    ainit( void )
155  {
156      sample_co = 0;                      //サンプリング・カウンタ・クリア
157      measure_interval = 0;               //計測間隔レジスタ・クリア
158      measure_co = 0;                     //計測カウンタ・クリア
159      measure_flag = 0;                   //計測フラグ・クリア
160      error_flag = 0;                     //エラー・フラグ・クリア
161      low_flag = 0;                       //ロウ・バッテリ・フラグ・クリア
162  }
```

（10）コマンド入力処理

```
163  ////////////////////////////////////////////////////////////////////////////
164  //  名称       : ComInput
165  //  形式       : void ComInput()
166  //
167  //  機能説明   : ホストより1コマンドの入力を行う。
168  //  引き数説明 : なし
169  //  戻り値     : なし
170  ////////////////////////////////////////////////////////////////////////////
171  void    ComInput( void )
172  {
173     in_co = 0;                              //コマンド・バッファ・カウンタ・クリア
174     while( 1 ) {
175
176        while ( ~SRIC1 & 0x80 );             //UART1受信完了フラグが立つまで待つ
177        in_buf[in_co] = RXB1;                //UART1受信バッファの内容を
178                                             //コマンド・バッファへ格納
179        SRIC1 = (SRIC1 & 0x7F);              //受信完了フラグをクリア
180        if  ( in_buf[in_co] == NULL) break; //格納した内容がNULLであれば抜ける
181        in_co++;                             //コマンド・バッファ・カウンタを+1する
182     }
183  }
```

（11）コマンド解析処理

```
184  ////////////////////////////////////////////////////////////////////////////
185  //  名称       : ComAna
186  //  形式       : int ComAna()
187  //
188  //  機能説明   : コマンドの解析を行いバッファに格納する。
189  //  引き数説明 : なし
190  //  戻り値     : コマンド番号
191  ////////////////////////////////////////////////////////////////////////////
192  int    ComAna( void )
193  {
194  int wk;
195  /*  */
196      wk = in_buf[1];                          //コマンド・バッファ1番地の内容を取り出す
197      wk -= 0x30;                              //その内容から30Hを引く
198      if  ( wk == DATAOUT ){                   //その結果が2であれば戻り値2を帰し抜ける
199          return DATAOUT;
200      }
201      if  ( wk != MEASURE ){                   //その結果が1でなければ戻り値0を帰し抜ける
202          return 0;
203      }
204          // 現在時刻を取り出しバイナリ変換し格納
205      hexconv( &now_time.year, &in_buf[2] );
206      hexconv( &now_time.month, &in_buf[4] );
207      hexconv( &now_time.day, &in_buf[6] );
208      hexconv( &now_time.hour, &in_buf[8] );
209      hexconv( &now_time.minute, &in_buf[10] );
210      hexconv( &now_time.second, &in_buf[12] );
211      conv_time( &now_time, &now_time );     // 年月を週に変換
212      set_time( &now_time );
213          // 計測開始時刻を取り出しバイナリ変換し格納
214      hexconv( &start_time.year, &in_buf[14] );
215      hexconv( &start_time.month, &in_buf[16] );
216      hexconv( &start_time.day, &in_buf[18] );
217      hexconv( &start_time.hour, &in_buf[20] );
218      hexconv( &start_time.minute, &in_buf[22] );
219      hexconv( &start_time.second, &in_buf[24] );
220      conv_time( &start_time, &start_time ); // 年月を週に変換
221          // 計測間隔をバイナリ変換し格納
222      hexconv( &measure_interval, &in_buf[26] );
223          // 計測数をバイナリ変換し格納
224      hexconv( &measure_co, &in_buf[28] );
```

```
225     if  ( measure_co == 0 ) {
226        measure_co = 1;                    //計測数が0なら1をセット
227        error_flag = 1;                    //エラー・フラグを1にセット
228     } else if  ( measure_co > 80 ) {
229        measure_co = 80;                   //計測数が80以上なら80をセット
230        error_flag = 2;                    //エラー・フラグを2にセット
231     }
232     return MEASURE;
233  }
```

（12）計測データ出力処理

```
234  ////////////////////////////////////////////////////////////////////////////
235  //  名称       : DataOut
236  //  形式       : void DataOut()
237  //
238  //  機能説明   : 収集データをホストへ送出する。
239  //  引き数説明 : なし
240  //  戻り値     : なし
241  ////////////////////////////////////////////////////////////////////////////
242  void   DataOut( void )
243  {
244  int    lco, status;
245  /*  */
246     hexout( sample_co * 22 + 2 );                //トータル数出力
247       for( lco=0; lco < measure_co ; lco++ ) {   //収集データの終わりまで出力
248         hexout( lco );                           //サンプリング番号出力
249         hexout( log_buf[lco].time.year );        //年をHEX変換し出力
250         hexout( log_buf[lco].time.month );       //月をHEX変換し出力
251         hexout( log_buf[lco].time.day );         //日をHEX変換し出力
252         hexout( log_buf[lco].time.hour );        //時をHEX変換し出力
253         hexout( log_buf[lco].time.minute );      //分をHEX変換し出力
254         hexout( log_buf[lco].time.second );      //秒をHEX変換し出力
255
256         hexout( log_buf[lco].adi[0] >> 8 );      //温度データをHEX変換し出力
257         hexout( log_buf[lco].adi[0] );           //温度データをHEX変換し出力
258         hexout( log_buf[lco].adi[1] >> 8 );      //湿度データをHEX変換し出力
259         hexout( log_buf[lco].adi[1] );           //湿度データをHEX変換し出力
260         hexout( log_buf[lco].adi[2] >> 8 );      //圧力データをHEX変換し出力
261         hexout( log_buf[lco].adi[2] );           //圧力データをHEX変換し出力
262         hexout( log_buf[lco].adi[3] >> 8 );      //バッテリ電圧データをHEX変換し出力
263         hexout( log_buf[lco].adi[3] );           //バッテリ電圧データをHEX変換し出力
264
265     }
266     status = 0;                                  //ステータスに0をセット
267     if ( measure_co != sample_co ){              //サンプリング数と計測数が等しくない場合は
268     status |= 0x01;                              //ステータスのビット0に1をセットする
269     }
270     if ( error_flag ){                           //エラー・フラグがセットされている場合は
271        status |= 0x02;                           //ステータスのビット1に1をセット
272     }
273     if ( low_flag  ){                            //ロウ・バッテリ・フラグがセットされている場合は
274        status |= 0x04;                           //ステータスのビット2に1をセット
```

```
275     }
276     hexout( status );                          //計測状態ステータスをHEXに変換して出力
277     cout( 0x0d ); cout( 0x0a );                // C/R L/F 出力
278 }
```

(13) 計測処理

```
279 ////////////////////////////////////////////////////////////////////////////
280 //  名称       : DataLog
281 //  形式       : void DataLog()
282 //
283 //  機能説明   : 計測を行いデータを収集する。
284 //  引き数説明 : なし
285 //  戻り値     : なし
286 ////////////////////////////////////////////////////////////////////////////
287 void   DataLog( void )
288 {
289 int i;
290 /*  */
291        // 開始時間待ち
292     if  ( check_time( &now_time, &start_time ) ) {
293        return;                                  //計測開始時刻に達してないので抜ける
294     }
295     #pragma asm                                 --サブクロック動作モードへ切り替え
296        ld.b    PCC[r0],r10                      --PCCレジスタの内容を読み出す
297        or      0x08,r10                         --ビット4を1にセットする
298        st.b    r10,PRCMD[r0]                    --特定レジスタに対するプログラム
299        st.b    r10,PCC[r0]                      --PCCレジスタにセット
300        nop                                      --特定レジスタに対するプログラム
301        nop                                      --特定レジスタに対するプログラム
302        nop                                      --特定レジスタに対するプログラム
303        nop                                      --特定レジスタに対するプログラム
304        nop                                      --特定レジスタに対するプログラム
305     l100:                                       --サブクロック動作モードに移行したか
306                                                 --確認する
307        ld.b    PCC[r0],r10                      --PCCレジスタの内容を読み出す
308        and     0x10,r10                         --ビット4が1になるまでループする
309        bz      l100
310     #pragma endasm                              --サブクロック動作モード移行完了
311
312     while( WEEK != start_time.week ) {          //現在週が開始週と一致するまで
313        if  ( low_flag ){                        //ロウ・バッテリ・フラグを確認しながら待つ
314            return;
315        }
316     }
317     while( DAY  != start_time.day ) {           //現在日が開始日と一致するまで
318        if  ( low_flag ){                        //ロウ・バッテリ・フラグを確認しながら待つ
319            return;
```

```
320         }
321     }
322     while( HOUR != start_time.hour ) {          //現在時間が開始時間と一致するまで
323         if  ( low_flag ){                       //ロウ・バッテリ・フラグを確認しながら待つ
324             return;
325         }
326     }
327     while( MIN  != start_time.minute ) {        //現在分が開始分と一致するまで
328         if  ( low_flag ){                       //ロウ・バッテリ・フラグを確認しながら待つ
329             return;
330         }
331     }
332     while( SEC != start_time.second ) {         //現在秒が開始秒と一致するまで
333         if  ( low_flag ){                       //ロウ・バッテリ・フラグを確認しながら待つ
334             return;
335         }
336     }
337
338     while( 1 ) {
339             // 計測時刻を計測データ・バッファに格納
340         log_buf[sample_co].time.week = WEEK;
341         log_buf[sample_co].time.day  = DAY;
342         log_buf[sample_co].time.hour = HOUR;
343         log_buf[sample_co].time.minute = MIN;
344         log_buf[sample_co].time.second = SEC;
345         rconv_time( &log_buf[sample_co].time.year, &log_buf[sample_co].time.year );
346                                                 // 週を年月に変換
347
348             // データ収集
349         for( i = 0; i < 4; i++ ) {              //温度→湿度→圧力→バッテリ電圧の
350                                                 //4データを順に取込む
351             ADM = 0x00;                         //A/D変換動作停止
352             ADS = i;                            //アナログ入力ポートを指定
353             ADM = 0x80;                         //A/D変換動作許可
354             while( 1 ) {
355                 if  ( low_flag ){               //ロウ・バッテリ・フラグを確認する
356                     return;
357                 }
358                 if  ( ADIC & 0x80 ){            //A/D変換完了
359                     break;
360                 }
361             }
362             log_buf[sample_co].adi[i] = ADCR;   //A/D変換レジスタ内容を
```

```
363                                          //計測データ・バッファへ格納
364             ADIC = (ADIC & 0x7f);
365         }
366
367         sample_co++;                     //サンプリング・カウンタを+1
368         if  ( sample_co == measure_co ){ //サンプリング・カウンタ値が計測数になったら
369             RTCC1 = 0x80;                //RTCのインターバル動作停止
370             RTCIC = 0x43;                //RTCの割り込み禁止
371             PIC1 = 0x43;                 //INTP1割り込み禁止
372             return;
373             }                            // 計測終了
374             // 次のログ時間待ち
375         measure_flag = 0;                //計測フラグ・クリア
376         if  ( measure_interval == 0 ) {  //計測間隔が0なら
377             RTCC1 = 0xb8 ;               //毎秒割り込みをセット
378         } else if  ( measure_interval == 1 ) {  //計測間隔が1なら
379             RTCC1 = 0xc0 ;               //毎分割り込みをセット
380         } else if  ( measure_interval == 2 ) {  //計測間隔が2なら
381             RTCC1 = 0xc8 ;               //毎時割り込みをセット
382         } else {                         //それ以外なら
383             RTCC1 = 0xd0 ;               //毎日割り込みをセット
384         }
385             RTCIC = 0x03;                //RTCの割り込み許可
386
387         while( 1 ) {
388         // サブIDLEモードへ
389             #pragma asm
390                 st.b    r0,PSMR          --IDLEモードを指定
391                 mov     0x02,r10
392                 st.b    r10,PRCMD[r0]
393                 st.b    r10,PSC[r0]      --サブIDLEモード有効
394                 nop
395                 nop
396                 nop
397                 nop
398                 nop
399             #pragma endasm               --サブIDLEモードへ移行
400
401             if  ( low_flag ){            //ロウ・バッテリ・フラグがセットされていれば抜ける
402                 return;
403             }
404             if  ( measure_flag ){        //計測フラグがセットされていれば
405                 break;                   //計測時刻を計測データ・バッファに格納の処理へ
```

```
406             }
407         }
408     }
409 }
```

（14）スリープ処理

```
410  ////////////////////////////////////////////////////////////////////////////
411  //  名称       : Sleep
412  //  形式       : void Sleep()
413  //
414  //  機能説明   : メイン・クロックを止めNMIを待つ。
415  //  引き数説明 : なし
416  //  戻り値     : なし
417  ////////////////////////////////////////////////////////////////////////////
418  void   Sleep( void )
419  {
420          // メイン・クロック停止
421      #pragma asm
422          ld.b    PCC[r0],r10
423          or      0x40,r10
424          st.b    r10,PRCMD[r0]
425          st.b    r10,PCC[r0]                     --メイン・クロック停止指定
426          nop
427          nop
428          nop
429          nop
430          nop
431      #pragma endasm                              --メイン・クロック停止完了
432          // サブIDLEモード
433      #pragma asm
434          st.b    r0,PSMR                         --IDLEモードを指定
435          mov     0x02,r10
436          st.b    r10,PRCMD[r0]
437          st.b    r10,PSC[r0]                     --サブIDLEモード有効
438          nop
439          nop
440          nop
441          nop
442          nop
443      #pragma endasm
444  }
```

(15) ウエイク・アップ処理

```
445  ////////////////////////////////////////////////////////////////////////////
446  //  名称       : WakeUp
447  //  形式       : void WakeUp()
448  //
449  //  機能説明   : メイン・クロックを動作させ，通常モードへ移る。
450  //  引き数説明 : なし
451  //  戻り値     : なし
452  ////////////////////////////////////////////////////////////////////////////
453  void   WakeUp( void )
454  {
455
456                  //メイン・クロック動作，メイン・クロック選択
457      #pragma asm
458         ld.b    PCC[r0],r10
459         and    0xbf,r10
460         st.b    r10,PRCMD[r0]
461         st.b    r10,PCC[r0]                          --メイン・クロック動作を指定
462         nop
463         nop
464         nop
465         nop
466         nop
467      #pragma endasm
468
469      #pragma asm
470         ld.b    PCC[r0],r10
471         and     0xf7,r10
472         st.b    r10,PRCMD[r0]
473         st.b    r10,PCC[r0]                        --メイン・クロックを指定
474         nop
475         nop
476         nop
477         nop
478         nop
479      #pragma endasm
480
481      while(PCC & 0x10);                             //メイン・クロック動作を確認
482
483  }
```

（16）HEX-ASCII　Binary　変換処理

```
484  ////////////////////////////////////////////////////////////////////////////
485  //  名称       : hexconv
486  //  形式       : void hexconv( unsigned char *out, unsigned *data )
487  //             out : 変換結果格納ポインタ
488  //             data: HEX-ASCII格納ポインタ
489  //  機能説明   : HEX-ASCII2バイトをバイナリ1バイトに変換する。
490  //  引き数説明 : なし
491  //  戻り値     : なし
492  ////////////////////////////////////////////////////////////////////////////
493  void   hexconv( unsigned char *out, unsigned char *data )
494  {
495      if  ( *data > 0x40 ) {                      //最初のASCIIバイトが0x40以上なら
496          *out = ( *data++ - 0x41 + 10 ) << 4;    //バイナリ値0xA～0xFを1バイトの上位
497                                                  //4ビットに格納
498      } else {
499          *out = ( *data++ - 0x30 ) << 4;         //最初のASCIIバイトからバイナリ値
500                                                  //0x0～0x9を格納
501      }
502      if  ( *data > 0x40 ) {                      //次のASCIIバイトが0x40以上なら
503          *out += ( *data - 0x41 + 10 );          //バイナリ値0xA～0xFを１バイトの
504                                                  //下位4ビットに格納
505      } else {
506          *out += ( *data - 0x30 );               //次のASCIIバイトからバイナリ値
507                                                  //0x0～0x9を格納
508      }
509  }
```

（17）Binary HEX-ASCII 変換と出力処置

```
510  ////////////////////////////////////////////////////////////////////////////
511  //  名称       : hexout
512  //  形式       : void hexout( unsigned char data )
513  //             data: 出力データ
514  //  機能説明   : 1バイト・データをHEX-ASCII2バイトへ変換する。
515  //  引き数説明 : なし
516  //  戻り値     : なし
517  ////////////////////////////////////////////////////////////////////////////
518  void   hexout( unsigned char data )
519  {
520  short   wk;
521  /*  */
522     wk = ( data >> 4 ) & 0x0f;                  //バイト・データの上位4ビットを抽出
523     if  ( wk < 10 ) {
524        cout( wk + 0x30 );                       //10より小なら0x30を加える
525     } else {
526        cout( wk - 10 + 0x41 );                  //でなければ10を引いて0x41を加える
527     }
528     wk = data & 0x0f;                           //バイト・データの下位4ビットを抽出する
529     if  ( wk < 10 ) {
530        cout( wk + 0x30 );                       //10より小なら0x30を加える
531     } else {
532        cout( wk - 10 + 0x41 );                  //でなければ10を引いて0x41を加える
533     }
534  }
```

（18）RS-232-C出力処理

```
535  ////////////////////////////////////////////////////////////////////////////
536  //  名称       : cout
537  //  形式       : void cout( unsigned char data )
538  //             data: 出力データ
539  //  機能説明   : 1バイトRS-232-Cへ出力する。
540  //  引き数説明 : なし
541  //  戻り値     : なし
542  ////////////////////////////////////////////////////////////////////////////
543  void   cout( unsigned char data )
544  {
545      while( ASIF1 & 0x02 ) ;           //UART1の送信レジスタが空になるまでループ
546      TXB1 = data ;                     //(空なので)送信データを送信バッファ・レジスタに格納
547  }
```

（19）リアルタイム・カウンタ時刻設定処理

```
548  ////////////////////////////////////////////////////////////////////////////
549  //  名称       : set_time
550  //  形式       : void set_time( TIME *time )
551  //             time: 設定データポインタ
552  //  機能説明   : 時刻をRTCへセットする。
553  //  引き数説明 : なし
554  //  戻り値     : なし
555  ////////////////////////////////////////////////////////////////////////////
556  void   set_time( TIME *time_data )
557  {
558      RTCC0 = 0x00;                              //リアルタイム・カウンタカウント停止
559      WEEKB = time_data->week;                   //現在時刻の中の週データをWEEKBにセット
560      DAYB  = time_data->day;                    //現在時刻の中の曜日データをDAYBにセット
561      HOURB = time_data->hour;                   //現在時刻の中の時データをHOURBにセット
562      MINB  = time_data->minute;                 //現在時刻の中の分データをMINBにセット
563      SECB  = time_data->second;                 //現在時刻の中の秒データをSECBにセット
564
565      RTCC0 = 0x80;
566      RTCC1 = 0x80;                              //リアルタイム・カウンタカウント開始
567  }
```

(20) 年月データ週変換処理

```
568  ////////////////////////////////////////////////////////////////////////////
569  //  名称       : conv_time
570  //  形式       : void conv_time( TIME *time_data1, TIME *time_data2 )
571  //             time_data1: 変換後時刻データポインタ
572  //             time_data2: 変換前時刻データポインタ
573  //  機能説明   : 年および月を2000年1月基準の週,曜日に変換
574  //  引き数説明 : なし
575  //  戻り値     : なし
576  ////////////////////////////////////////////////////////////////////////////
577  void   conv_time( TIME *time_data1, TIME *time_data2 )
578  {
579  TIME   twk = { 00, 1, 0, 1, 0, 0, 0 };          //2000年，1月，第0週，第1曜日，0時，0分，0秒を
580                                                  //基準値とする
581  /*  */
582      time_data1->year = time_data2->year;         //変換前の年データを格納
583      time_data1->month = time_data2->month;       //変換前の月データを格納
584      time_data1->day = time_data2->day;           //変換前の曜日データを格納
585      time_data1->hour = time_data2->hour;         //変換前の時データを格納
586      time_data1->minute = time_data2->minute;     //変換前の分データを格納
587      time_data1->second = time_data2->second;     //変換前の秒データを格納
588      time_data1->week = ( cal_day( time_data1 ) - cal_day( &twk ) ) / 7;
589                                                  //西暦1年1月1日から2000年1月1日
590                                                  //までの日数の残りから週数を計算して格納
591      time_data1->day = ( cal_day( time_data1 ) - cal_day( &twk ) ) % 7;
592                                                  //その余りで曜日を計算して格納
593  }
```

(21) 週データ年月変換処理

```
594  ////////////////////////////////////////////////////////////////////////////
595  //  名称       : rconv_time
596  //  形式       : void rconv_time( TIME *time_data1, TIME *time_data2 )
597  //             time_data1: 変換後時刻データ・ポインタ
598  //             time_data2: 変換前時刻データ・ポインタ
599  //  機能説明   : 2000年1月基準の週を年月日に変換
600  //  引き数説明 : なし
601  //  戻り値     : なし
602  ////////////////////////////////////////////////////////////////////////////
603  void   rconv_time( TIME *time_data1, TIME *time_data2 )
604  {
605  TIME   twk = { 00, 1, 0, 1, 0, 0, 0 };        //2000年，1月，第0週，第1曜日，0時，0分，0秒を
606                                                 //基準値とする
607  int    now_day, day, i;
608  /*  */
609     now_day = ( time_data2->week * 7 ) + time_data2->day + cal_day( &twk );
610                                                 //週データに7を掛け，曜日データを加え，
611                                                 //基準値を加え現在日を算出
612     for ( i = 0; i < 99; i++ ) {               //2000年から2098年までの変換をする
613     twk.year = i + 1;                          //基準値に1年を加える
614     day = cal_day( &twk );                     //その日数をdayに格納
615     if  ( now_day < day ) break;               //現在日数よりdayが大きくなったら計算を抜ける
616     }
617     twk.year = i;                              //iの内容を年にする
618
619     for ( i = 0; i < 12; i++ ) {               //1月から12月までの変換をする
620     twk.month = i + 1;                         //基準値に1月を加える
621     day = cal_day( &twk );                     //その日数をdayに格納
622     if  ( now_day < day ) break;               //現在日数よりdayが大きくなったら計算を抜ける
623     }
624     twk.month = i;                             //iの内容を月にする
625
626     time_data1->year = twk.year;               //年変換後データを格納
627     time_data1->month = twk.month;             //月変換後データを格納
628     time_data1->day = twk.day;                 //日変換後データを格納
629     time_data1->hour = time_data2->hour;       //時変換後データを格納
630     time_data1->minute = time_data2->minute;   //分変換後データを格納
631     time_data1->second = time_data2->second;   //秒変換後データを格納
632  }
```

(22) 計測開始時刻処理

```
633  ////////////////////////////////////////////////////////////////////////////
634  //  名称       : check_time
635  //  形式       : int check_time( TIME *ntime, TIME *stime )
636  //             ntime: 現在時刻データ・ポインタ
637  //             stime: 開始時刻データ・ポインタ
638  //  機能説明   : 開始時刻が現在時刻より後かチェックを行う。
639  //  引数説明   : なし
640  //  戻り値     : 0：ノーマル  1：エラー
641  ////////////////////////////////////////////////////////////////////////////
642  int check_time( TIME *ntime, TIME *stime )
643  {
644      if  ( ntime->week > stime->week ) return 1;    //現在週が開始週より大きければ1を戻す
645      if  ( ntime->week != stime->week ) return 0;   //現在週が開始週より大きくなければ0を戻す
646      if  ( ntime->day > stime->day ) return 1;      //現在日が開始日より大きければ1を戻す
647      if  ( ntime->day != stime->day ) return 0;     //現在日が開始日より大きくなければ0を戻す
648      if  ( ntime->hour > stime->hour ) return 1;    //現在時が開始時より大きければ1を戻す
649      if  ( ntime->hour != stime->hour ) return 0;   //現在時が開始時より大きくなければ0を戻す
650      if  ( ntime->minute > stime->minute ) return 1;//現在分が開始分より大きければ1を戻す
651      if  ( ntime->minute != stime->minute ) return 0;//現在分が開始分より大きくなければ0を戻す
652      if  ( ntime->second >= stime->second ) return 1;//現在秒が開始秒より大きければ1を戻す
653      return 0;
654  }
```

(23) 年月日処理

```
655  ////////////////////////////////////////////////////////////////////////////
656  //  名称       : cal_day
657  //  形式       : int cal_day( TIME *timep )
658  //             timep: 時刻データ・ポインタ
659  //  機能説明   : 西暦1年1月1日からの日数を計算する。
660  //  引き数説明 : なし
661  //  戻り値     : 計算した日数
662  ////////////////////////////////////////////////////////////////////////////
663  int    cal_day( TIME *timep )
664  {
665  int    mon[12] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
666  int    yy, y1, y2, y3;
667  int    ur, dy, i;
668  /*  */
669      yy = timep->year + 2000;                //年データに2000を加え格納
670      if  ( ((yy % 4) == 0) && ((yy % 100) != 0) || ((yy % 400) == 0 ) ) {
671                                              //うるう年は，4で割り切れる年から100で
672  //割り切れる年を除き，400で割り切れる年を加えた年
673          mon[1] = 29;                        //2月の日数を29にする
674      } else {
675          mon[1] = 28;                        //2月の日数を28にする
676      }
677
678      y1 = (yy - 1) / 4;                      //前年データを4で割る
679      y2 = (yy - 1) / 100;                    //前年データを100で割る
680      y3 = (yy - 1) / 400;                    //前年データを400で割る
681      ur = y1 - y2 + y3;                      //うるう年の回数を格納
682
683      dy = (yy - 1) * 365 + ur + timep->day;  //西暦1年から前年末までの日数を算出し格納
684      for(i=0;i<(timep->month - 1); i++ ) {   //本年1月1日から前月末までの日数を計算し格納
685          dy = dy + mon[i];                   //当日の日数を加える格納
686      }
687      return dy;                              //日数を戻す
688  }
```

4. 4　ROMコレクション機能サンプル・プログラム

4. 4. 1　仕様概要

マスク化されたROMで発見された命令バグを修正可能な，ROMコレクション機能を，前記プログラムに使用するサンプル例を示します。

（1）構成

（a）修正ポイント

リアル・タイム・カウンタ割り込み処理（int_rtc），外部割り込み処理（int_p01）」の2箇所

（b）修正プログラム[注1]

内蔵RAMに修正プログラムを格納し，修正プログラムに変更してプログラム実行

（c）ROMコレクション設定情報

ROMコレクション機能の許可／禁止設定，修正アドレス，戻り先アドレス[注2]をROMコレクション・テーブルとして内蔵RAMに格納

注1. 実際のシステムでは修正プログラム，ROMコレクション設定情報は外部メモリなどからダウンロードされるイメージになります。このため，このサンプル・プログラム例ではROMコレクション許可 = 0，禁止 = 1と負論理に設定しています。

2. このサンプル・プログラムでは，ROMコレクション機能の戻り先アドレスは，発生アドレスの直後（DBPCに退避されたアドレス）になっていますが，実際のシステムでは修正ポイントを事前に想定した処理単位等でのプログラム代替になり，復帰命令（DBRET命令）実行前に戻り先アドレスの書き換えを実行します。

サンプル・プログラムはデータ計測システムをベースにして，処理内に追加変更されたもの，および新規に作成した処理についてフロー・チャートとプログラム・リストを記載しています。

ROMコレクションの動作とプログラムの流れを次に示します。

図4－30　ROMコレクションの動作とプログラムの流れ

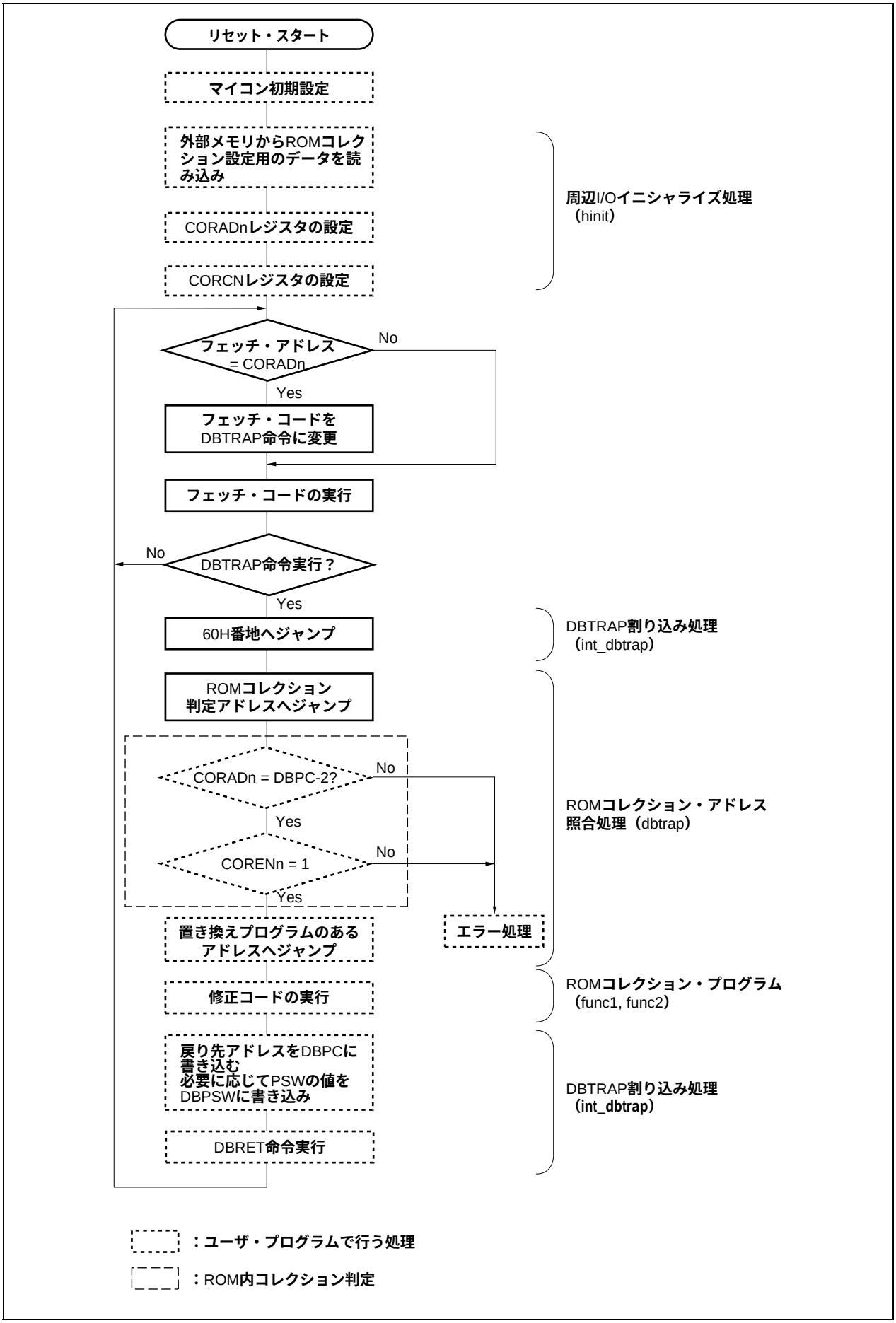
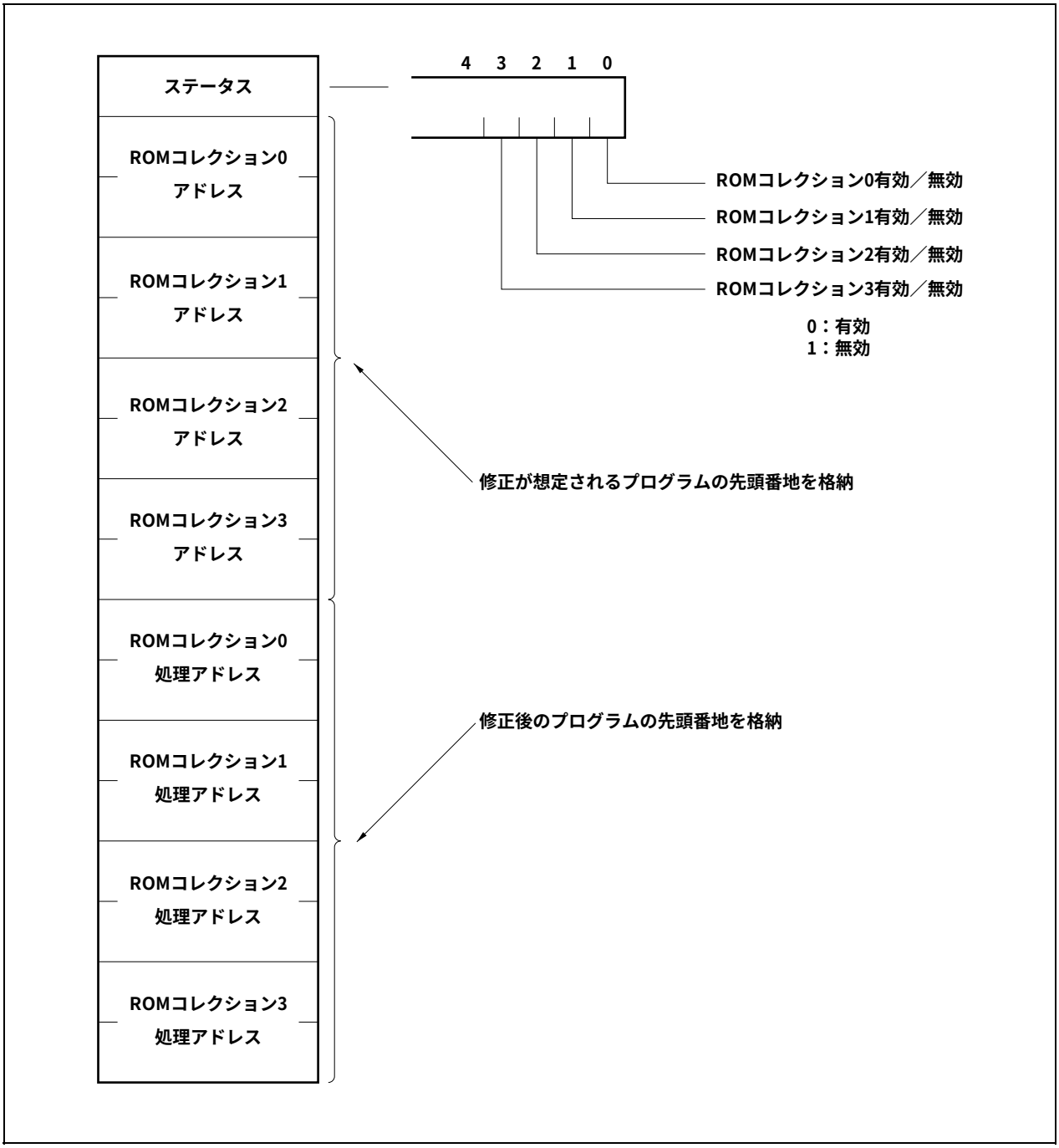


図4－31　ROMコレクション・テーブル



4.4.2　フロー・チャート

図4－32　ROMコレクション割り込み処理

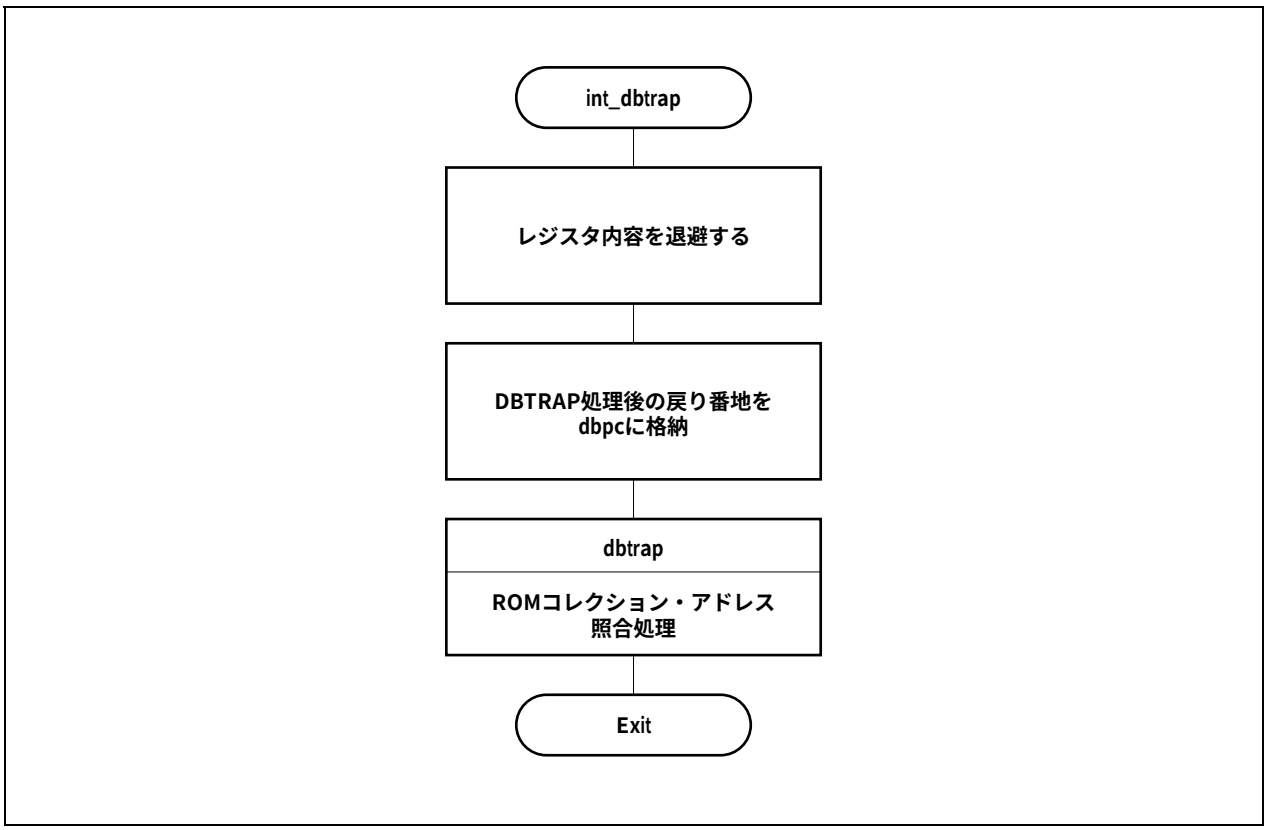


図4－33　ROMコレクション・アドレス照合処理

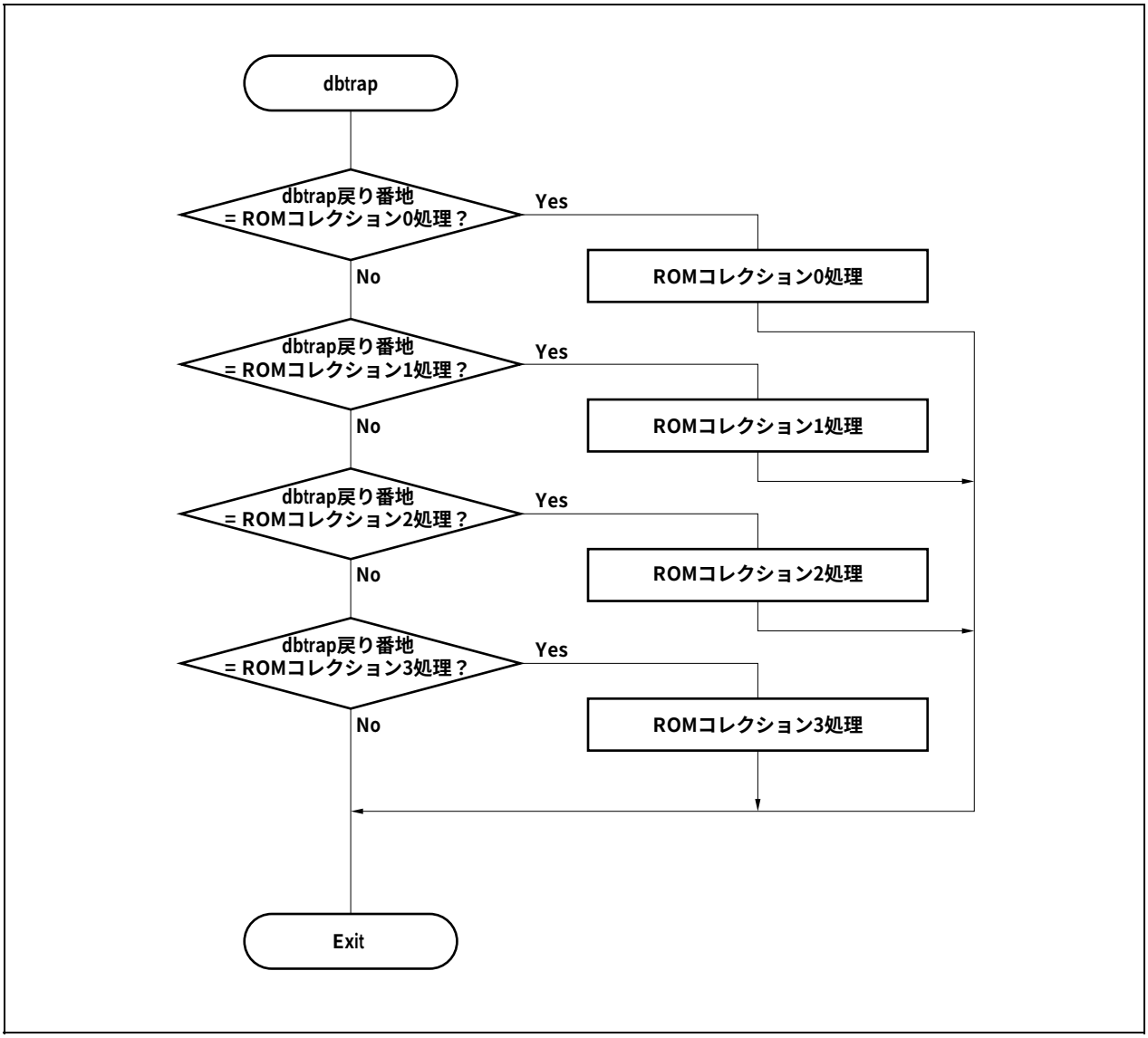


図4－34　リアルタイム・カウンタ割り込み処理

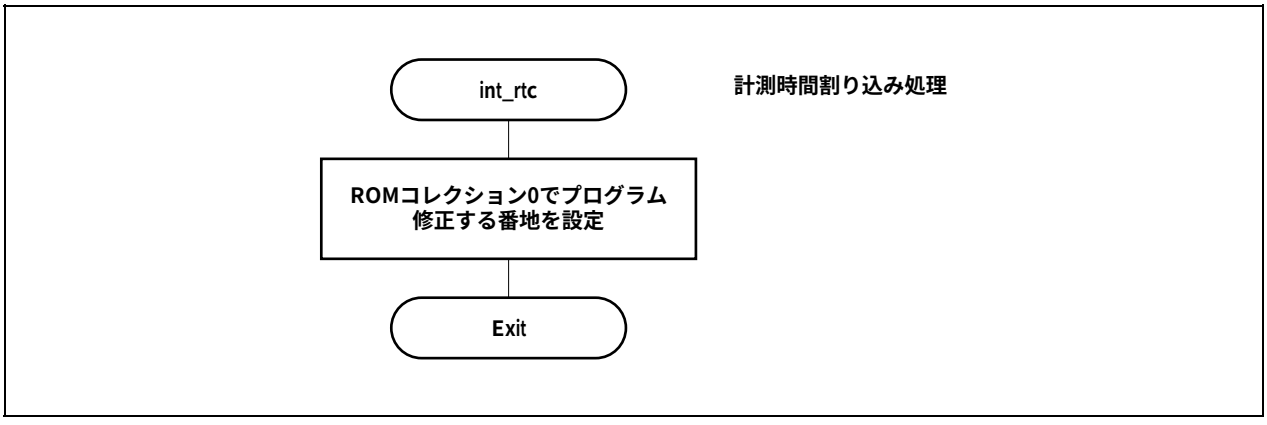


図4－35　INTP01外部割り込み処理

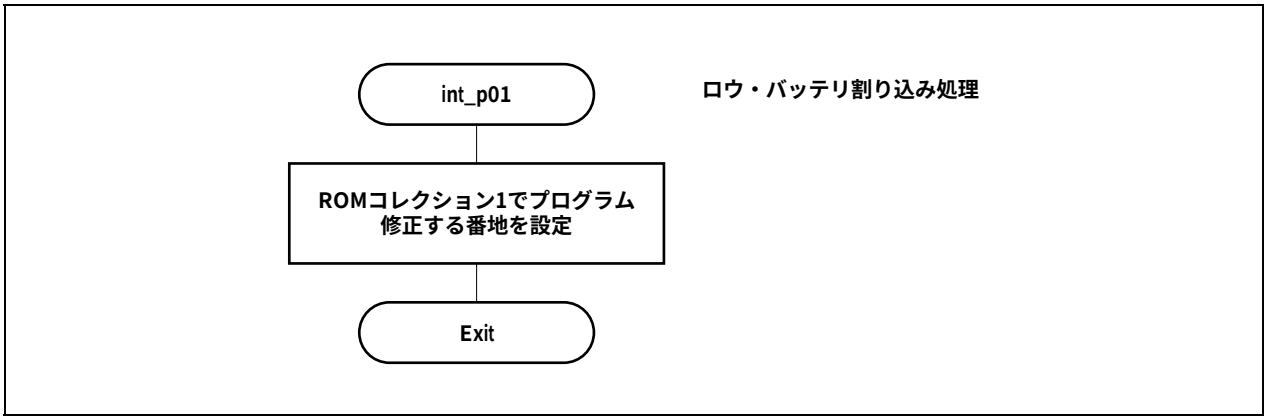


図4－36　内蔵周辺I/Oイニシャライズ処理

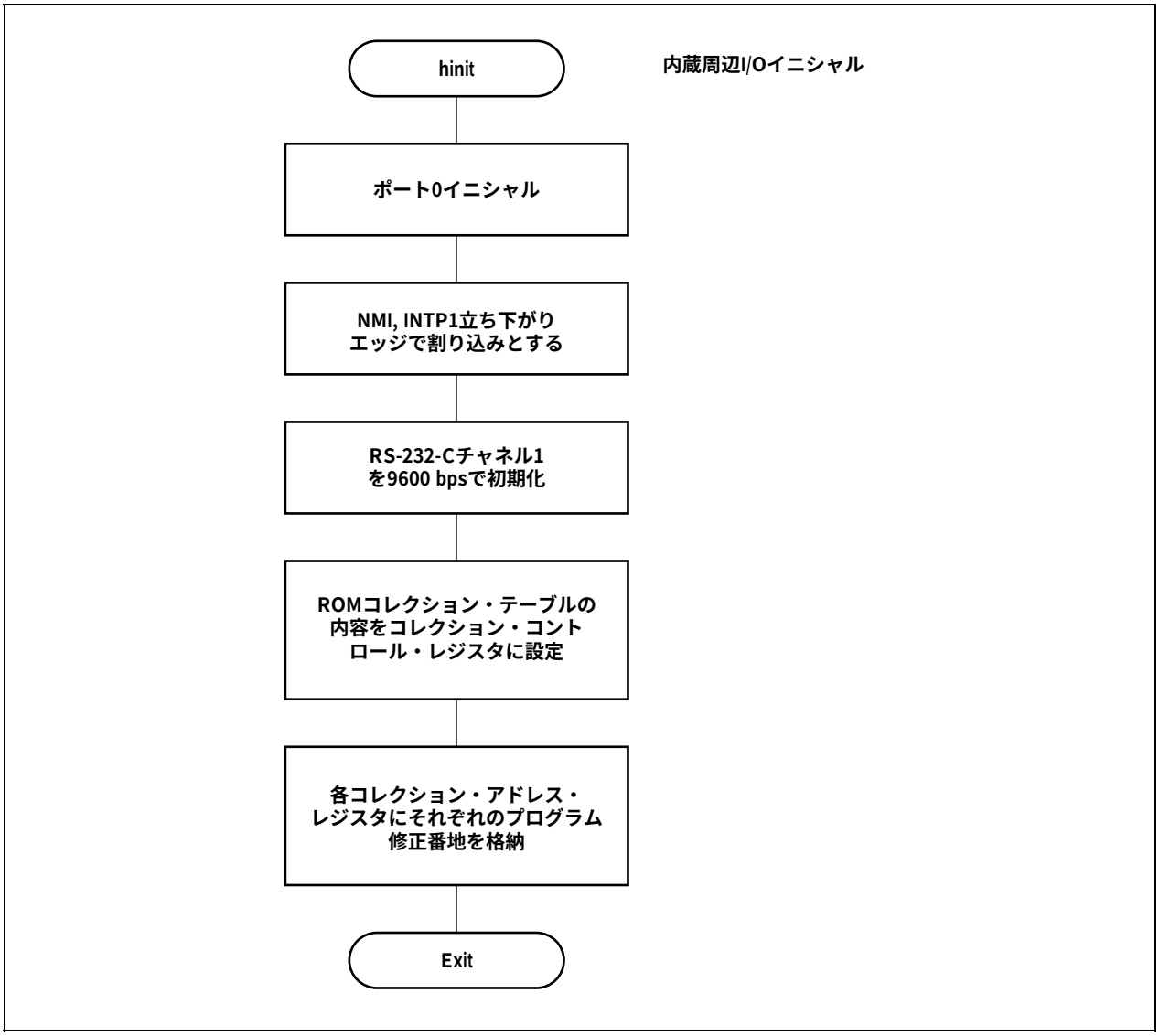


図4－37　コモン・エリア・イニシャライズ処理

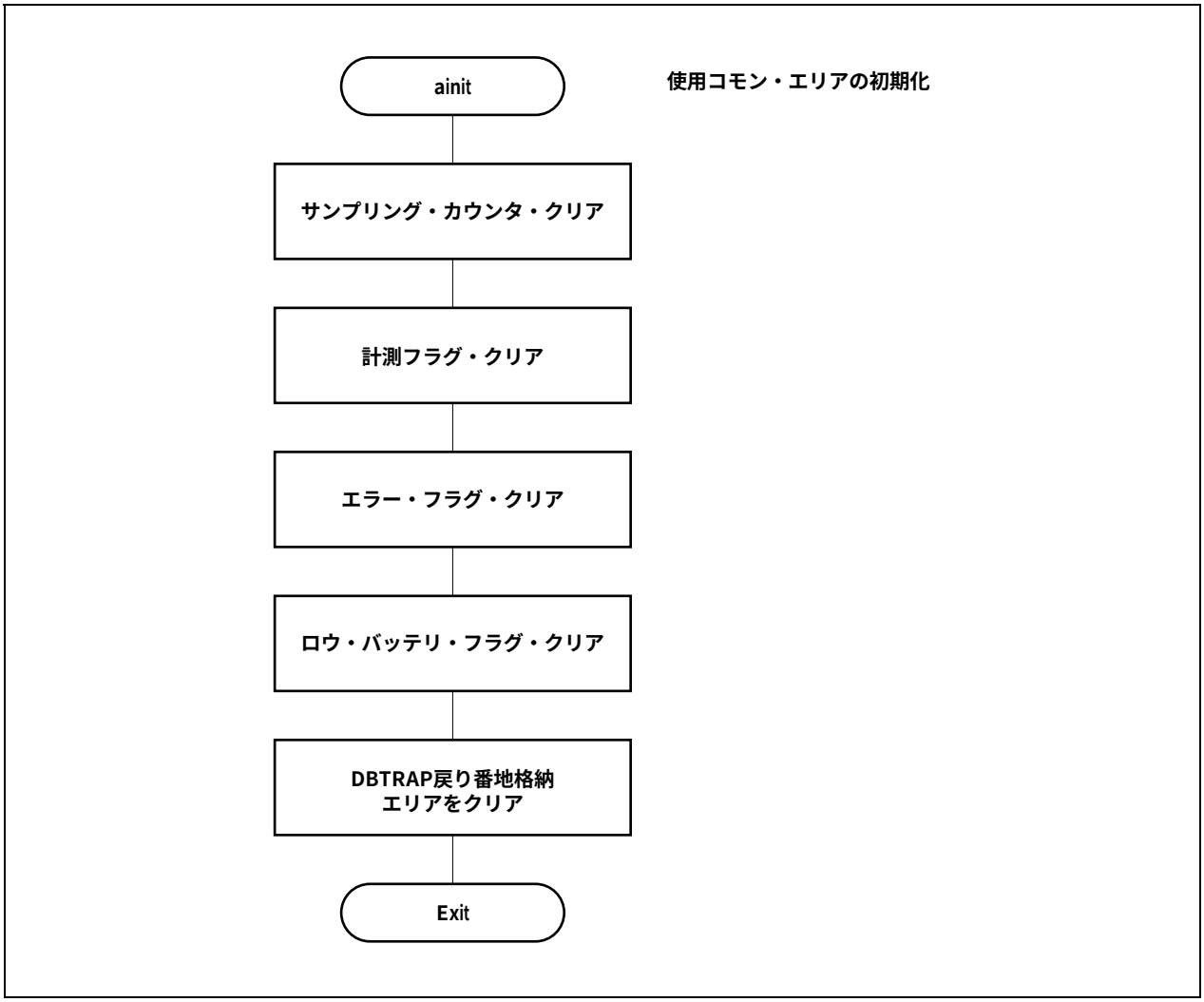


図4－38　ROM**コレクション・プログラム**1

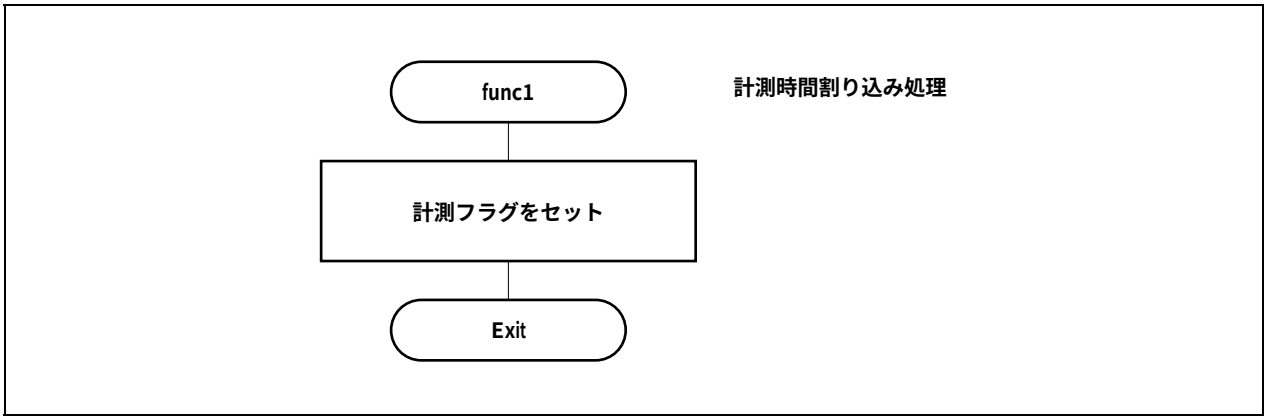
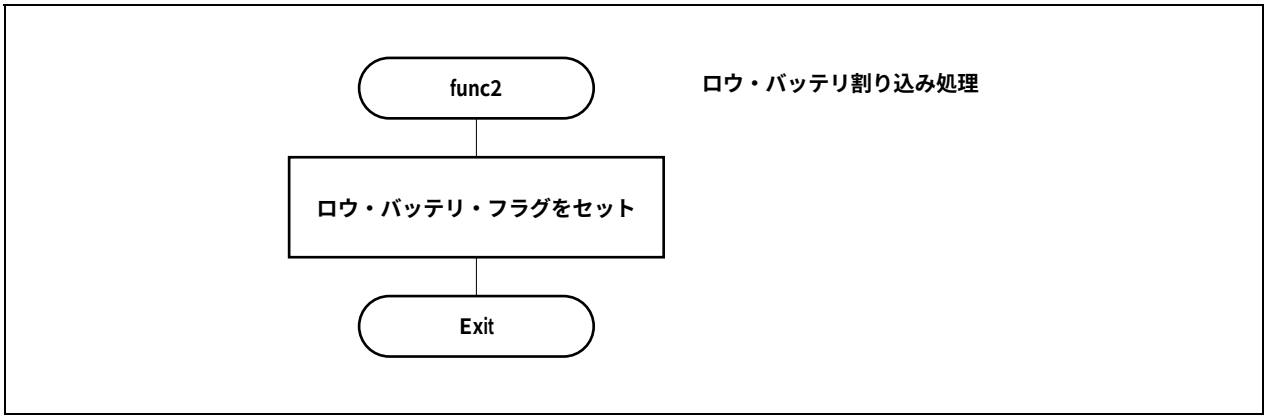


図4－39　ROM**コレクション・プログラム**2



4.4.3　プログラム・リスト

(1) 定数定義

```
689  ////////////////////////////////////////////////////////////////////////////
690  //  定数定義
691  ////////////////////////////////////////////////////////////////////////////
692  #define MEASURE    1                        //計測コマンド番号
693  #define DATAOUT    2                        //データ出力コマンド番号
694  typedef struct time {                       //時刻データ構成
695      short   year;
696      short   month;
697      short   week;
698      short   day;
699      short   hour;
700      short   minute;
701      short   second;
702  } TIME;
703
704  #define    ROMCORE_ADD  0x2000;             //ROMコレクション・テーブル先頭番地
```

(2) コモン・エリア

```
705  ////////////////////////////////////////////////////////////////////////////
706  //  コモン・エリア
707  ////////////////////////////////////////////////////////////////////////////
708  char    in_buf[40];                          //コマンド・バッファ
709  short   in_co;                               //コマンド・バッファ・カウンタ
710  short   sample_co;                           //サンプリング・カウンタ
711  int     measure_interval;                    //計測間隔
712  short   measure_co;                          //計測数
713  short   measure_flag;                        //計測フラグ
714  short   low_flag;                            //ロウ・バッテリ・フラグ
715  short   error_flag;                          //計測データ・オーバ
716  TIME    now_time;                            //現在時刻
717  TIME    start_time;                          //計測開始時刻
718  struct  log {                                //計測データ・バッファ
719      TIME    time;
720      short   adi[4];
721  } log_buf[80];
722
723  int     dbtrap;                              //DBTRAP戻り番地格納エリア
```

(3) DBTRAP割り込み処理

```
724  ////////////////////////////////////////////////////////////////////////////
725  //  名称       : int_dbtrap
726  //  形式       : void int_dbtrap()
727  //  機能説明   : ROMコレクション割り込み処理
728  //  引き数説明 : なし
729  //  戻り値     : なし
730  ////////////////////////////////////////////////////////////////////////////
731  #pragma interrupt DBG0 int_dbtrap
732
733  void    int_dummy( void )
734  {
735     #pragma asm
736     .globl     _int_dbtrap
737     _int_bdtrap:
738        prepare 1p,  112                        --r31レジスタ内容の退避
739        st.w    r1,   108[sp]                   --以下各レジスタ内容の退避
740        st.w    r6,   104[sp]
741        st.w    r7,   100[sp]
742        st.w    r8,   96[sp]
743        st.w    r9,   92[sp]
744        st.w    r10,  88[sp]
745        st.w    r11,  84[sp]
746        st.w    r12,  80[sp]
747        st.w    r13,  76[sp]
748        st.w    r14,  72[sp]
749        st.w    r15,  68[sp]
750        st.w    r16,  64[sp]
751        st.w    r17,  60[sp]
752        st.w    r18,  56[sp]
753        st.w    r19,  52[sp]
754        st.w    r20,  48[sp]
755        st.w    r21,  44[sp]
756        st.w    r22,  40[sp]
757        st.w    r23,  36[sp]
758        st.w    r24,  32[sp]
759        st.w    r25,  28[sp]
760        st.w    r26,  24[sp]
761        st.w    r27,  20[sp]
762        st.w    r28,  16[sp]
763        st.w    r29,  12[sp]
764        st.w    r30,    8[sp]
```

```
765         stsr    16,   r1                    --システムレジスタ16の内容を読み出し
766         st.w    r1,   4[sp]                 --退避する
767         stsr    17,   r1                    --システムレジスタ17の内容を読み出し
768         st.w    r1,   0[sp]                 --退避する
769         stsr    18,   r10                   --システムレジスタ18の内容を読み出し
770         st.w    r10,$_dbpc                  --dbpcに格納(DBTRAP戻り番地を格納)
771         jarl    _dbtrap,1p                  --ROMコレクション・アドレス照合処理
772                                             --にとび処理後戻る
773         ld.w    0[sp],   r1                 --退避した内容を各レジスタに復帰
774         ldsr    r1,      17
775         ld.w    4[sp],   r1
776         ldsr    r1,      16
777         ld.w    8[sp],   r30
778         ld.w    12[sp],  r29
779         ld.w    16[sp],  r28
780         ld.w    20[sp],  r27
781         ld.w    24[sp],  r26
782         ld.w    28[sp],  r25
783         ld.w    32[sp],  r24
784         ld.w    36[sp],  r23
785         ld.w    40[sp],  r22
786         ld.w    44[sp],  r21
787         ld.w    48[sp],  r20
788         ld.w    52[sp],  r19
789         ld.w    56[sp],  r18
790         ld.w    60[sp],  r17
791         ld.w    64[sp],  r16
792         ld.w    68[sp],  r15
793         ld.w    72[sp],  r14
794         ld.w    76[sp],  r13
795         ld.w    80[sp],  r12
796         ld.w    84[sp],  r11
797         ld.w    88[sp],  r10
798         ld.w    92[sp],  r9
799         ld.w    96[sp],  r8
800         ld.w    100[sp], r7
801         ld.w    104[sp], r6
802         ld.w    108[sp], r1
803         dispose 112,     lp
804
805         dbret                               --DBTRAP割り込み処理を抜け
806                                             --DBPC番地へ戻る
807     #pragma endasm
```

```
808   }
```

(4) ROMコレクション・アドレス照合処理

```
809  ////////////////////////////////////////////////////////////////////////////
810  //  名称       : dbtrap
811  //  形式       : void dbtrap()
812  //  機能説明   : ROMコレクション・アドレス照合処理
813  //  引き数説明 : なし
814  //  戻り値     : なし
815  ////////////////////////////////////////////////////////////////////////////
816  void     dbtrap( void )
817  {
818  int      *po;
819  void     (*func)();
820  /*       */
821      po = (int *)ROMCORE_ADD;                    //ROMコレクション・テーブル先頭番地を
822                                                  //格納
823      if  ( (*(po+1) + 2 ) == dbpc ) {            //ROMコレクション0アドレスに
824                                                  //+2した番地はdbtrap処理戻り番地か？
825              func = *(po+5);                     //ROMコレクション0処理アドレスを
826                                                  //読み出す
827              (*func)();                          //ROMコレクションを実行する
828      } else if ( (*(po+2) + 2) == dbpc ) {       //ROMコレクション1アドレスに
829                                                  //+2した番地はdbtrap処理戻り番地か？
830              func = *(po+6);                     //ROMコレクション1処理アドレスを
831                                                  //読み出す
832              (*func)();                          //ROMコレクションを実行する
833      } else if ( (*(po+3) + 2) == dbpc ) {       //ROMコレクション2アドレスに
834                                                  //+2した番地はdbtrap処理戻り番地か？
835              func = *(po+7);                     //ROMコレクション2処理アドレスを
836                                                  //読み出す
837              (*func)();                          //ROMコレクションを実行する
838      } else {                                    //ROMコレクション3アドレスに
839                                                  //+2した番地はdbtrap処理戻り番地か？
840              func = *(po+8);                     //ROMコレクション3処理アドレスを
841                                                  //読み出す
842              (*func)();                          //ROMコレクションを実行する
843      }
844  }
```

(5) リアルタイム・カウンタ割り込み処理

```
845  ////////////////////////////////////////////////////////////////////////////
846  //  名称       : int_rtc
847  //  形式       : void int_rtc()
848  //  機能説明   : 計測時間割り込み処理
849  //               (DBTRAP割り込みを発生させ，修正プログラムに処理を代替する)
850  //  引き数説明 : なし
851  //  戻り値     : なし
852  ////////////////////////////////////////////////////////////////////////////
853  #pragma interrupt INTRTC int_rtc
854  __interrupt
855  void    int_rtc( void )
856  {
857
858      #pragma asm
859          .globl  _label1
860      _label1:                                    --ROMコレクション0，修正想定先頭番地
861                                                  --DBTRAP割り込み発生番地
862                  nop                             --ROMコレクション0，プログラム修正後の
863                                                  --戻り番地
864      #pragma endasm;
865  }
```

(6) INTP01外部割り込み処理

```
866  ////////////////////////////////////////////////////////////////////////////
867  //  名称       : int_p01
868  //  形式       : void int_p01()
869  //  機能説明   : ロウ・バッテリ割り込み処理
870  //               (DBTRAP割り込みを発生させ，修正プログラムに処理を代替する)
871  //  引き数説明 : なし
872  //  戻り値     : なし
873  ////////////////////////////////////////////////////////////////////////////
874  #pragma interrupt INTP1 int_p01
875  __interrupt
876  void    int_p01( void )
877  {
878
879      #pragma asm
880           .globl   _label2
881      _label2:                                  --ROMコレクション1，修正想定先頭番地
882                                                --DBTRAP割り込み発生番地
883      nop                                       --ROMコレクション1，プログラム修正後の
884                                                --戻り番地
885      #pragma endasm;
886  }
```

（7）内蔵周辺I/Oイニシャライズ処理

```
887  ////////////////////////////////////////////////////////////////////////////
888  //  名称       : hinit
889  //  形式       : void hinit()
890  //  機能説明   : 内蔵周辺I/Oのイニシャルを行う。
891  //  引き数説明 : なし
892  //  戻り値     : なし
893  ////////////////////////////////////////////////////////////////////////////
894  void    hinit( void )
895  {
896  int *po;
897  /* */
898
899      PMC0  = 0x05;
900      PM0   = 0xff;
901      PU0   = 0x3f;
902      INTF0 = 0x05;
903      INTR0 = 0x00;                          //NMI,INTP1立ち下がりエッジ割り込み
904      PIC1  = 0x03;                          //割り込み許可
905
906      CKSR1 = 0x03;
907      BRGC1 = 130;                           //9600 bps
908      ASIM1 = 0xe5;                          //8ビット　NONパリティ　ストップ１
909      PFC9  = 0x0300;                        //UART1設定
910      PMC9  = 0x0300;
911
912      // ROMコレクション設定
913  po = (int *)ROMCORE_ADD;                   //ROMコレクション・テーブル先頭番地を格納
914  CORCN  = ~*po++;                           //ROMコレクション有効無効設定を
915                                             //レジスタにセット
916  CORAD0 = *po++;                            //ROMコレクション0アドレス設定を
917                                             //レジスタにセット(修正想定先頭番地)
918  CORAD1 = *po++;                            //ROMコレクション1アドレス設定を
919                                             //レジスタにセット(修正想定先頭番地)
920  CORAD2 = *po++;                            //ROMコレクション2アドレス設定を
921                                             //レジスタにセット(修正想定先頭番地)
922  CORAD3 = *po;                              //ROMコレクション3アドレス設定を
923                                             //レジスタにセット(修正想定先頭番地)
924  }
```

(8) コモン・エリア・イニシャライズ処理

```
925  ////////////////////////////////////////////////////////////////////////////
926  //  名称       : ainit
927  //  形式       : void ainit()
928  //
929  //  機能説明    : 使用コモン・エリアの初期化
930  //  引き数説明  : なし
931  //  戻り値      : なし
932  ////////////////////////////////////////////////////////////////////////////
933  void    ainit( void )
934  {
935      sample_co = 0;                          //サンプリング・カウンタ・クリア
936      measure_interval = 0;                   //計測間隔レジスタ・クリア
937      measure_co = 0;                         //計測カウンタ・クリア
938      measure_flag = 0;                       //計測フラグ・クリア
939      error_flag = 0;                         //エラー・フラグ・クリア
940      low_flag = 0;                           //ロウ・バッテリ・フラグ・クリア
941
942      dbtrap=0;                               //DBTRAP戻り番地格納エリア・クリア
943  }
```

(9) ROMコレクション・テーブル

```
944  ////////////////////////////////////////////////////////////////////////////
945  // ROMコレクション・テーブル
946  ////////////////////////////////////////////////////////////////////////////
947     #pragma asm
948  .section"rtext",text
949  .globl   _label1
950  .globl   _label2
951
952  .word    0xfc                              --ROMコレクション・ステータス
953
954  .word    _label1                           --ROM0コレクション・アドレス
955  .word    _label2                           --ROM1コレクション・アドレス
956  .word    0                                 --ROM2コレクション・アドレス
957  .word    0                                 --ROM3コレクション・アドレス
958
959  .word    _func1                            --ROM0コレクション処理アドレス
960  .word    _func2                            --ROM1コレクション処理アドレス
961  .word    0                                 --ROM2コレクション処理アドレス
962  .word    0                                 --ROM3コレクション処理アドレス
963     #pragma endasm
```

（10）ROMコレクション・プログラム1

```
964  ////////////////////////////////////////////////////////////////////////////
965  //  名称       :func1
966  //  形式       : void func1()
967  //  機能説明   : ROMコレクション・プログラム１
968  //               （int_rtc処理の修正プログラム）
969  //  引き数説明 : なし
970  //  戻り値     : なし
971  ////////////////////////////////////////////////////////////////////////////
972  void func1()
973  {
974     measure_flag = 1;                        //計測フラグをセット
975}
```

（11）ROMコレクション・プログラム2

```
976  ////////////////////////////////////////////////////////////////////////////
977  //  名称       :func2
978  //  形式       : void func2()
979  //  機能説明   : ROMコレクション・プログラム２
980  //               （int_p01処理の修正プログラム）
981  //  引き数説明 : なし
982  //  戻り値     : なし
983  ////////////////////////////////////////////////////////////////////////////
984  void                                        func2()
985  {
986     low_flag = 1;                            //ロウ・バッテリ・フラグをセット
987  }
```

〔メ　モ〕

図4－40　データ計測

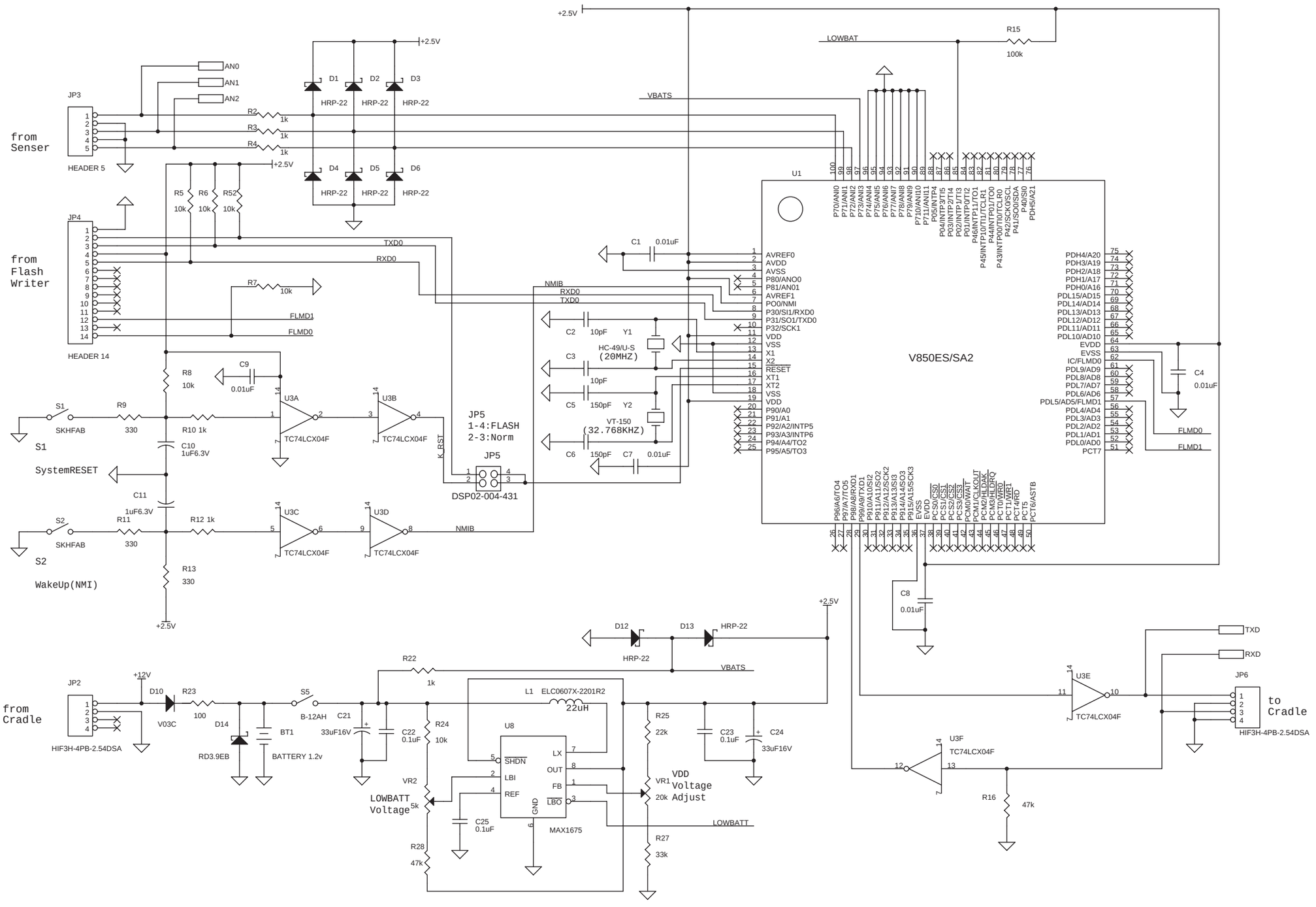

from Senser
JP3
HEADER 5
AN0
AN1
AN2
R2 1k
R3 1k
R4 1k
D1 D2 D3
D4 D5 D6
HRP-22
+2.5V
R5 10k
R6 10k
R52 10k
from Flash Writer
JP4
HEADER 14
TXD0
RXD0
FLMD1
FLMD0
R7 10k
R8 10k
C9 0.01uF
S1
SKHFAB
R9 330
R10 1k
C10 1uF6.3V
SystemRESET
C11 1uF6.3V
S2
R11 330
R12 1k
WakeUp(NMI)
R13 330
U3A
U3B
U3C
U3D
TC74LCX04F
K_RST
JP5
1-4:FLASH
2-3:Norm
DSP02-004-431
NMIB
+2.5V
LOWBAT
R15 100k
VBATS
U1
V850ES/SA2
C1 0.01uF
C2 10pF
C3 10pF
C5 150pF
C6 150pF
C7 0.01uF
Y1
HC-49/U-S
(20MHZ)
Y2
VT-150
(32.768KHZ)
AVREF0
AVDD
AVSS
P80/ANO0
P81/AN01
AVREF1
PO0/NMI
P30/SI1/RXD0
P31/SO1/TXD0
P32/SCK1
VDD
VSS
X1
X2
RESET
XT1
XT2
VSS
VDD
P90/A0
P91/A1
P92/A2/INTP5
P93/A3/INTP6
P94/A4/TO2
P95/A5/TO3
P70/ANI0
P71/ANI1
P72/ANI2
P73/ANI3
P74/ANI4
P75/ANI5
P76/ANI6
P77/ANI7
P78/ANI8
P79/ANI9
P710/ANI10
P711/ANI11
P05/INTP4
P04/INTP3/TI5
P03/INTP2/TI4
P02/INTP1/TI3
P01/INTP0/TI2
P46/INTP11/TIO1
P45/INTP10/TI1/TCLR1
P44/INTP01/TO0
P43/INTP00/TI0/TCLR0
P42/SCK0/SCL
P41/SO0/SDA
P40/SI0
PDH5/A21
PDH4/A20
PDH3/A19
PDH2/A18
PDH1/A17
PDH0/A16
PDL15/AD15
PDL14/AD14
PDL13/AD13
PDL12/AD12
PDL11/AD11
PDL10/AD10
EVDD
EVSS
IC/FLMD0
PDL9/AD9
PDL8/AD8
PDL7/AD7
PDL6/AD6
PDL5/AD5/FLMD1
PDL4/AD4
PDL3/AD3
PDL2/AD2
PDL1/AD1
PDL0/AD0
PCT7
C4 0.01uF
P96/A6/TO4
P97/A7/TO5
P98/A8/RXD1
P99/A9/TXD1
P910/A10/SI2
P911/A11/SO2
P912/A12/SCK2
P913/A13/SI3
P914/A14/SO3
P915/A15/SCK3
EVSS
EVDD
PCS0/CS0
PCS1/CS1
PCS2/CS2
PCS3/CS3
PCM0/WAIT
PCM1/CLKOUT
PCM2/HLDAK
PCM3/HLDRQ
PCT0/WR0
PCT1/WR1
PCT4/RD
PCT5
PCT6/ASTB
C8 0.01uF
D12 D13 HRP-22
+12V
JP2
HIF3H-4PB-2.54DSA
D10 V03C
R23 100
D14 RD3.9EB
BT1 BATTERY 1.2v
S5 B-12AH
C21 33uF16V
C22 0.1uF
R22 1k
L1 ELC0607X-2201R2
22uH
U8
MAX1675
SHDN
LBI
REF
LX
OUT
FB
LBO
GND
R24 10k
VR2 LOWBATT Voltage 5k
C25 0.1uF
R28 47k
R25 22k
VR1 20k
VDD Voltage Adjust
C23 0.1uF
C24 33uF16V
R27 33k
LOWBATT
U3E
U3F
TXD
RXD
JP6
to Cradle
HIF3H-4PB-2.54DSA
R16 47k

図4－41　クレードル

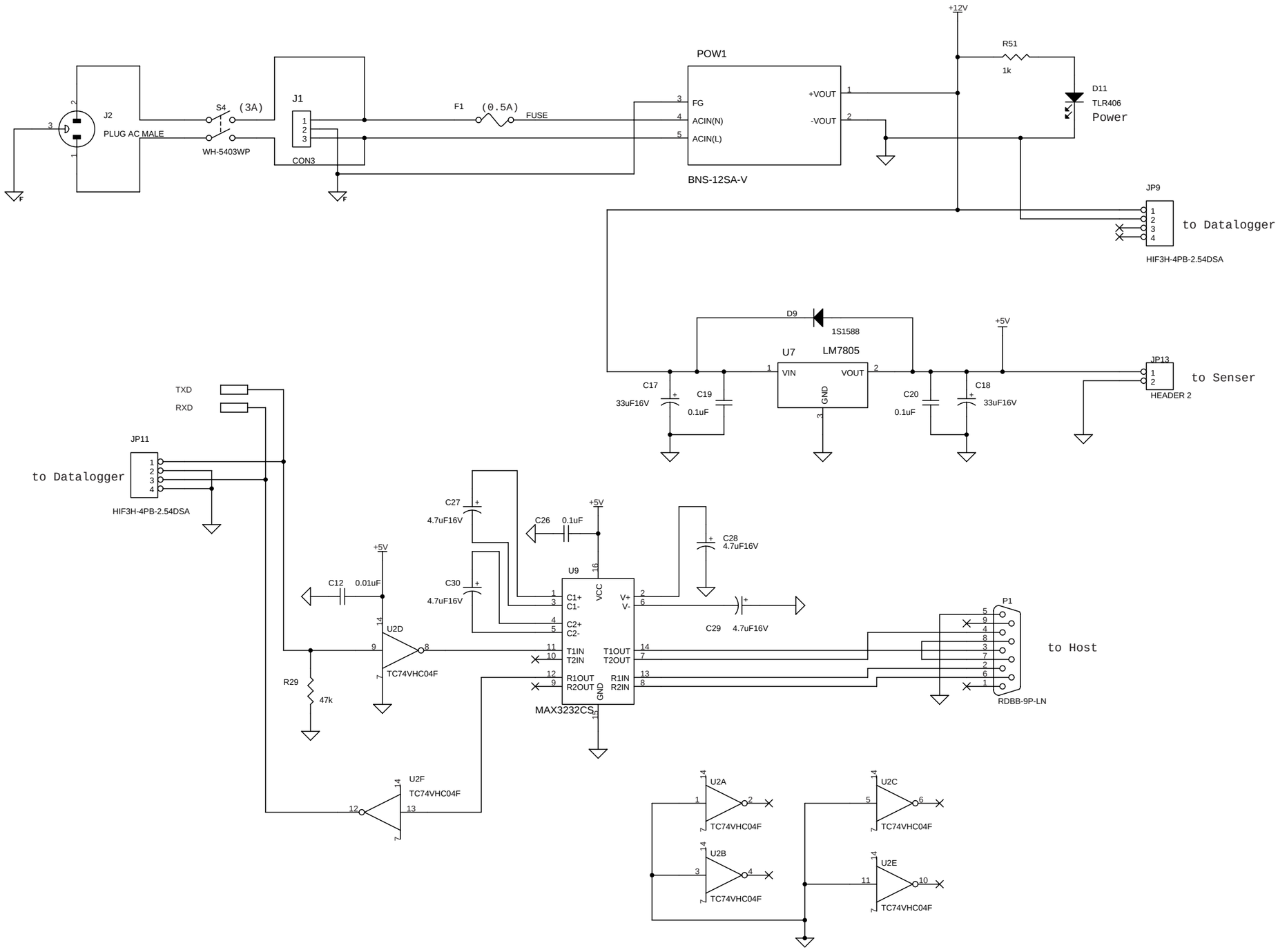
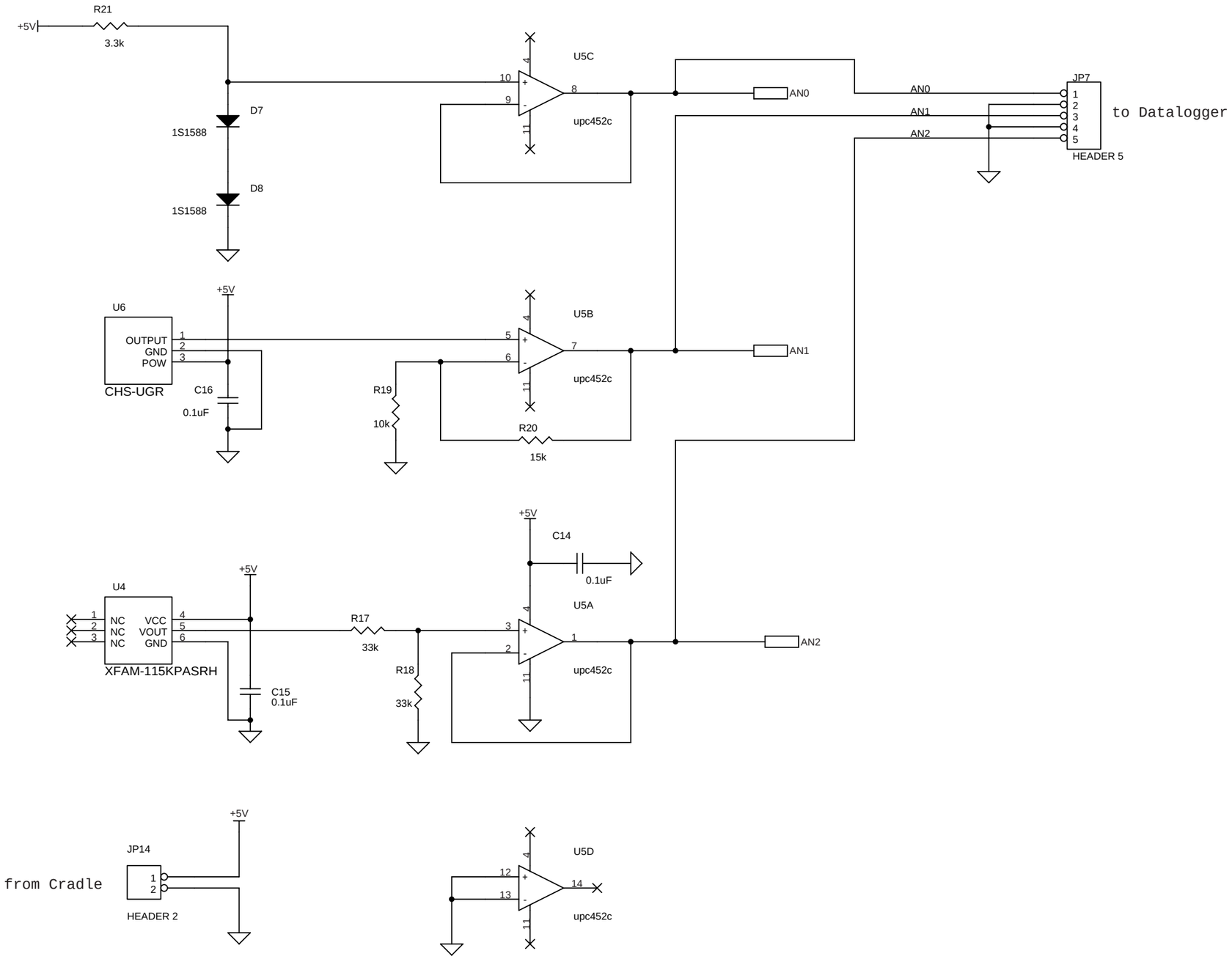

+12V
R51
1k
D11
TLR406
Power
POW1
+VOUT
-VOUT
FG
ACIN(N)
ACIN(L)
BNS-12SA-V
J2
PLUG AC MALE
S4 (3A)
WH-5403WP
J1
CON3
F1 (0.5A) FUSE
JP9
to Datalogger
HIF3H-4PB-2.54DSA
D9
1S1588
U7
LM7805
VIN
VOUT
GND
C17
33uF16V
C19
0.1uF
C20
0.1uF
C18
33uF16V
+5V
JP13
to Senser
HEADER 2
TXD
RXD
JP11
to Datalogger
HIF3H-4PB-2.54DSA
C27
4.7uF16V
C26 0.1uF
C28
4.7uF16V
C30
4.7uF16V
C29 4.7uF16V
U9
VCC
GND
C1+
C1-
C2+
C2-
V+
V-
T1IN
T2IN
R1OUT
R2OUT
T1OUT
T2OUT
R1IN
R2IN
MAX3232CS
C12 0.01uF
U2D
TC74VHC04F
R29
47k
U2F
TC74VHC04F
U2A
TC74VHC04F
U2B
TC74VHC04F
U2C
TC74VHC04F
U2E
TC74VHC04F
P1
RDBB-9P-LN
to Host

図4－42　センサ


R21
+5V
3.3k
D7
1S1588
D8
1S1588
U5C
upc452c
AN0
JP7
AN0
AN1
AN2
to Datalogger
HEADER 5
U6
OUTPUT
GND
POW
CHS-UGR
C16
0.1uF
U5B
upc452c
AN1
R19
10k
R20
15k
C14
0.1uF
U5A
upc452c
AN2
U4
NC
NC
NC
VCC
VOUT
GND
XFAM-115KPASRH
C15
0.1uF
R17
33k
R18
33k
from Cradle
JP14
HEADER 2
U5D
upc452c

〔メ　モ〕

【発　行】

NECエレクトロニクス株式会社

〒211-8668　神奈川県川崎市中原区下沼部1753

電話（代表）：044(435)5111

お問い合わせ先

【ホームページ】

NECエレクトロニク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.necel.co.jp/

【営業関係，技術関係お問い合わせ先】

半導体ホットライン

（電話：午前 9:00～12:00，午後 1:00～5:00）

電　話　：044−435−9494

E-mail　：info@lsi.nec.co.jp

【資料請求先】

NECエレクトロニクスのホームページよりダウンロードいただくか，NECエレクトロニクス特約店へお申し付けください。