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

V850/SA1™

32ビット・シングルチップ・マイクロコンピュータ

μPD703014A
μPD703014AY
μPD703014B
μPD703014BY
μPD703015A
μPD703015AY
μPD703015B
μPD703015BY
μPD703017A
μPD703017AY
μPD70F3015B
μPD70F3015BY
μPD70F3017A
μPD70F3017AY

資料番号 U13851JJ2V0AN00（第2版）
発行年月 April 2002 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，NECが出荷梱包に使用している導電性のトレーやマガジン・ケース，または導電性の緩衝材，金属ケースなどを利用し，組み立て工程にはアースを施してください。プラスチック板上に放置したり，端子を触ったりしないでください。

また，MOSデバイスを実装したボードについても同様の扱いをしてください。

②未使用入力の処理（CMOS特有）

注意　CMOSデバイスの入力レベルは固定してください。

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

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

③初期化以前の状態（MOS全般）

注意　電源投入時，MOSデバイスの初期状態は不定です。

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

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

注意：μPD703014AY, 703014BY, 703015AY, 703015BY, 703017AY, 70F3015BY, 70F3017AY**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ください。申告に基づき，以下の特典が受けられます。日本電気株式会社の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 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，外国為替および外国貿易管理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　　：μPD70F3015B, 70F3015BY, 70F3017A, 70F3017AY

ユーザ判定品：μPD703014A, 703014AY, 703014B, 703014BY,
μPD703015A, 703015AY, 703015B, 703015BY, 703017A, 703017AY

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

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

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

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

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

M7　98.8

本版で改訂された主な箇所

箇　所	内　容
全般	μPD703014B, 703014BY, 703015B, 703015BY, 70F3015B, 70F3015BYを追加
p.14	表1－1　V850/SA1の製品一覧を追加
p.15	1. 2　特　徴　最小命令実行時間に記述を追加
p.21	1. 6. 2（2）バス・コントロール・ユニット（BCU）記述を削除
p.27	1. 8　CPUレジスタ・セット　r2の用途変更
p.28	1. 8. 1（1）汎用レジスタ　r2の用途および動作変更
p.43	1. 13　周辺I/Oレジスタ　フラッシュ・メモリ・プログラミング・モード・コントロール・レジスタ（FLPMC）を追加
p.44	1. 13　周辺I/Oレジスタ　注を追加
p.46	2. 1. 1　BV_{DD}端子の接続　を削除

本文欄外の★印は，本版で改訂された主な箇所を示しています。

巻末にアンケート・コーナを設けております。このドキュメントに対するご意見をお気軽にお寄せください。

はじめに

対象者 このアプリケーション・ノートは，V850/SA1（μPD703014A, 703014AY, 703014B, 703014BY, 703015A, 703015AY, 703015B, 703015BY, 703017A, 703017AY, 70F3015B, 70F3015BY, 70F3017A, 70F3017AY）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。

目　的 このアプリケーション・ノートでは，V850/SA1を用いたシステム例として，「V850/SA1トレーニング・ボード（TB-V850/SA1）」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。

- V850/SA1の概要
- バス・インタフェース接続回路例
- 周辺機能の接続例
- 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。

ハードウエア機能の詳細を理解しようとするとき
→別冊のV850/SA1　**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。

命令機能の詳細を理解しようとするとき
→別冊のV850**シリーズ**™　**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：$\overline{xxx}$（端子，信号名称に上線）
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。**

V850/SA1**に関する資料**

資　料　名	資料番号
V850**シリーズ　ユーザーズ・マニュアル　アーキテクチャ編**	U10243J
μPD703014A, 703014AY, 703014B, 703014BY, 703015A, 703015AY, 703015B, 703015BY, 703017A, 703017AY　**データ・シート**	U14526J
μPD70F3017A,70F3017AY　**データ・シート**	U14527J
V850/SA1　**ユーザーズ・マニュアル　ハードウエア編**	U12768J
V850/SA1　**アプリケーション・ノート**	**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-703002-MC（**インサーキット・エミュレータ**）		U11595J
IE-703017-MC-EM1（**インサーキット・エミュレータ・オプション・ボード**）		U12898J
CA850　Ver.2.40**以上　C**コンパイラ・パッケージ	**操作編**	U15024J
	C**言語編**	U15025J
	プロジェクト・マネージャ編	U15026J
	アセンブリ言語編	U15027J
ID850　Ver.2.40　**統合ディバッガ**	**操作編**　Windows®**ベース**	U15181J
SM850　Ver.2.40　**システム・シミュレータ**	**操作編**　Windows**ベース**	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.3.13　**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）

図番号　　タイトル，ページ

図の目次（2/2）

表の目次

第1章　V850/SA1の概要

V850/SA1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 概　　説

V850/SA1は，V850シリーズの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，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は，高いリアルタイム応答性と1クロック・ピッチの基本命令に加え，ディジタル・サーボ制御の応用に最適な命令として，ハードウエア乗算器による乗算命令，飽和演算命令，ビット操作命令などを持っています。また，リアルタイム制御システムとして，超低消費電力を必要とするカムコーダなどのAV機器，携帯電話，PHSなどの携帯通信機器への応用が，きわめて高いコスト・パフォーマンスで実現できます。

次にV850/SA1の製品一覧について示します。

★ 表1－1　V850/SA1の製品一覧

品名	I^2C機能	ROM		RAMサイズ	パッケージ
		種類	サイズ		
μPD703014A	なし	マスクROM	64 Kバイト	4 Kバイト	121ピンFBGA（12×12）
μPD703014AY	あり				
μPD703014B	なし				100ピンLQFP（14×14）
μPD703014BY	あり				
μPD703015A	なし		128 Kバイト		121ピンFBGA（12×12）
μPD703015AY	あり				
μPD703015B	なし				100ピンLQFP（14×14）
μPD703015BY	あり				
μPD703017A	なし		256 Kバイト	8 Kバイト	100ピンLQFP（14×14）/ 121ピンFBGA（12×12）
μPD703017AY	あり				
μPD70F3015B	なし	フラッシュ・メモリ	128 Kバイト	4 Kバイト	100ピンLQFP（14×14）
μPD70F3015BY	あり				
μPD70F3017A	なし		256 Kバイト	8 Kバイト	100ピンLQFP（14×14）/ 121ピンFBGA（12×12）
μPD70F3017AY	あり				

1.2 特　　徴

○命令数　74

★　○最小命令実行時間　50 ns（メイン・クロック（f_{XX}）= 20 MHz動作時）

58.8 ns（メイン・クロック（f_{XX}）= 17 MHz動作時）

30.5 μs（サブクロック（f_{XT}）= 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）：100 ns（20 MHz動作時）

（レジスタ・ハザードが起きない後続の命令を並列に実行可能）

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　16 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：4 Mバイトまで可能

メモリ・ブロック分割機能：2 Mバイト／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

16ビット・データ・バス（アドレス／データ・マルチプレクス）

アドレス・バス：セパレート出力可能

バス・ホールド機能

外部ウエイト機能

★　○内蔵メモリ　μPD703014A, 703014AY, 703014B, 703014BY（ROM：64 Kバイト／RAM：4 Kバイト）

μPD703015A, 703015AY, 703015B, 703015BY（ROM：128 Kバイト／RAM：4 Kバイト）

μPD703017A, 703017AY（ROM：256 Kバイト／RAM：8 Kバイト）

μPD70F3015B, 70F3015BY（フラッシュ・メモリ：128 Kバイト／RAM：4 Kバイト）

μP70F3017A, 70F3017AY（フラッシュ・メモリ：256 Kバイト／RAM：8 Kバイト）

○割り込み／例外

外部割り込み　　：8要因（5要因[注]）

内部割り込み　　：24要因

ソフトウエア例外：32要因

例外トラップ　　：1要因

割り込み優先順位を任意に可変（8レベル）

注　ソフトウエアSTOPモード解除可能な外部割り込み数

○I/Oライン　　合計：85（入力ポート：13　　入出力ポート：72）

○タイマ／カウンタ

16ビット・タイマ：2ch（PWM出力）

8ビット・タイマ：4ch（PWM出力，カスケード接続可能）

○時計用タイマ　　サブクロック／メイン・クロック動作：1ch

○ウォッチドッグ・タイマ：1ch

○シリアル・インタフェース（SIO）

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）

★ （μPD703014AY, 703014BY, 703015AY, 703015BY, 703017AY, 70F3015BY, 70F3017AY のみ）

UART　：1ch

CSI　：1ch

UART/CSI：1ch

I^2C/CSI　：1ch

UART専用ボー・レート・ジェネレータ：2ch

○A/Dコンバータ　10ビット分解能：12ch

○DMAコントローラ　内蔵RAM⇔内蔵周辺I/O間：3ch

○リアルタイム出力ポート　8ビット×1ch　または　4ビット×2ch

○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5段階（スルーレート，サブ動作含む）

○パワー・セーブ機能　HALT/IDLE/ソフトウエアSTOPモード

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

121ピン・プラスチックFBGA（12×12）

○CMOS構造　完全スタティック回路

1.3　応用分野

カムコーダ（DVC含む），メータなどバッテリ駆動機器全般

★ 1.4　オーダ情報

品　名	パッケージ	内蔵ROM
μPD703014AF1-xxx-EA6	121ピン・プラスチックFBGA（12×12）	64 Kバイト（マスクROM）
μPD703014AYF1-xxx-EA6	〃	〃
μPD703014BGC-xxx-8EU	100ピン・プラスチックLQFP（ファインピッチ）（14×14）	〃
μPD703014BYGC-xxx-8EU	〃	〃
μPD703015AF1-xxx-EA6	121ピン・プラスチックFBGA（12×12）	128 Kバイト（マスクROM）
μPD703015AYF1-xxx-EA6	〃	〃
μPD703015BGC-xxx-8EU	100ピン・プラスチックLQFP（ファインピッチ）（14×14）	〃
μPD703015BYGC-xxx-8EU	〃	〃
μPD703017AF1-xxx-EA6	121ピン・プラスチックFBGA（12×12）	256 Kバイト（マスクROM）
μPD703017AGC-xxx-8EU	100ピン・プラスチックLQFP（ファインピッチ）（14×14）	〃
μPD703017AYF1-xxx-EA6	121ピン・プラスチックFBGA（12×12）	〃
μPD703017AYGC-xxx-8EU	100ピン・プラスチックLQFP（ファインピッチ）（14×14）	〃
μPD70F3015BGC-8EU	〃	128 Kバイト（フラッシュ・メモリ）
μPD70F3015BYGC-8EU	〃	〃
μPD70F3017AF1-EA6	121ピン・プラスチックFBGA（12×12）	256 Kバイト（フラッシュ・メモリ）
μPD70F3017AGC-8EU	100ピン・プラスチックLQFP（ファインピッチ）（14×14）	〃
μPD70F3017AYF1-EA6	121ピン・プラスチックFBGA（12×12）	〃
μPD70F3017AYGC-8EU	100ピン・プラスチックLQFP（ファインピッチ）（14×14）	〃

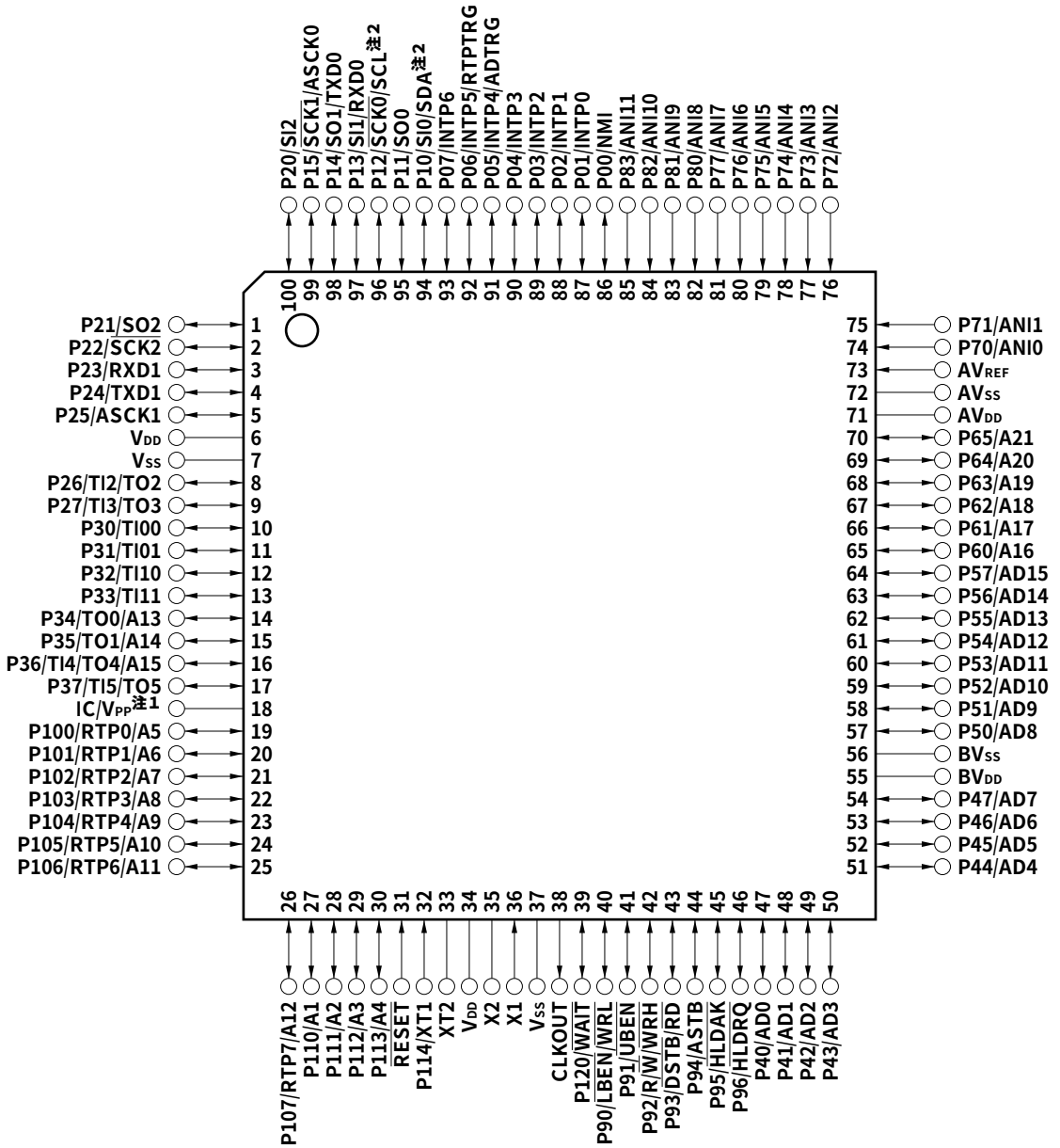
備考1．xxxはROMコード番号です。

2．ROMレス品はありません。

1.5　端子接続図

★　100ピン・プラスチックLQFP（ファインピッチ）（14×14）

- μPD703014BGC-xxx-8EU
- μPD703014BYGC-xxx-8EU
- μPD703015BGC-xxx-8EU
- μPD703015BYGC-xxx-8EU
- μPD703017AGC-xxx-8EU
- μPD703017AYGC-xxx-8EU
- μPD70F3015BGC-8EU
- μPD70F3015BYGC-8EU
- μPD70F3017AGC-8EU
- μPD70F3017AYGC-8EU



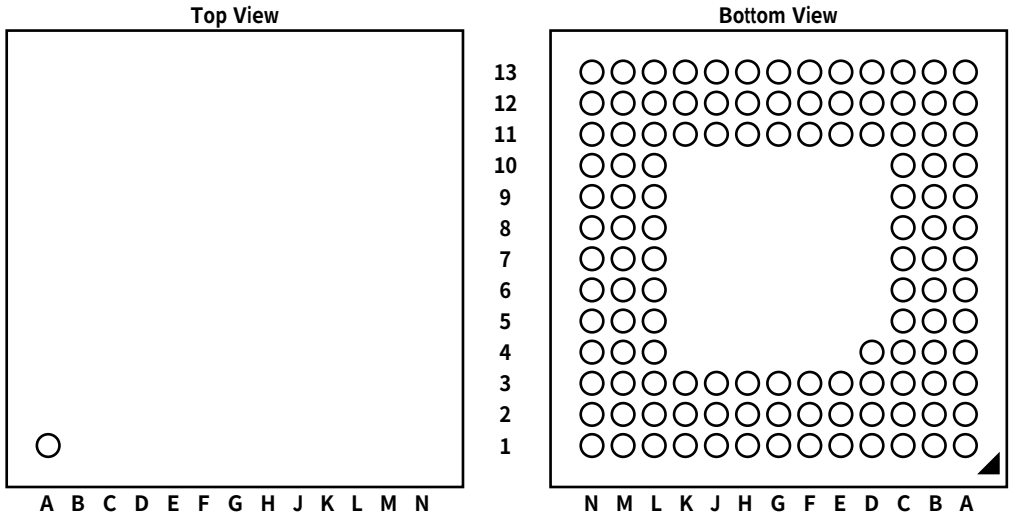
注1．μPD703014B, 703014BY, 703015B, 703015BY, 703017A, 703017AY：IC（VSSに直接接続してください）

μPD70F3015B, 70F3015BY, 70F3017A, 70F3017AY：VPP（通常動作モード時は，VSSに接続してください）

2．SCL, SDA端子は，μPD703014BY, 703015BY, 703017AY, 70F3015BY, 70F3017AYだけに有効です。

★　121ピン・プラスチックFBGA（12×12）

- μPD703014AF1-xxx-EA6
- μPD703014AYF1-xxx-EA6
- μPD703015AF1-xxx-EA6
- μPD703015AYF1-xxx-EA6
- μPD703017AF1-xxx-EA6
- μPD703017AYF1-xxx-EA6
- μPD70F3017AF1-EA6
- μPD70F3017AYF1-EA6



ピン番号	名　称	ピン番号	名　称	ピン番号	名　称	ピン番号	名　称	ピン番号	名　称	ピン番号	名　称
A1	P20	B8	P83	D2	V_{DD}	G11	P60	K13	BV_{DD}	M7	V_{SS}
A2	P15	B9	P80	D3	V_{SS}	G12	P56	L1	P104	M8	V_{SS}
A3	V_{SS}	B10	P75	D11	AV_{DD}	G13	P57	L2	P105	M9	P92
A4	P13	B11	AV_{SS}	D12	AV_{DD}	H1	P34	L3	$\overline{\text{RESET}}$	M10	P95
A5	P11	B12	AV_{SS}	D13	AV_{DD}	H2	P37	L4	V_{DD}	M11	P41
A6	P06	B13	P71	E1	P25	H3	P35	L5	V_{SS}	M12	P45
A7	P03	C1	P22	E2	V_{DD}	H11	P55	L6	X2	M13	P44
A8	P00	C2	P23	E3	P30	H12	P53	L7	P90	N1	P107
A9	P81	C3	V_{SS}	E11	AV_{DD}	H13	P54	L8	P120	N2	P110
A10	P76	C4	P24	E12	P64	J1	IC/V_{PP}注	L9	P93	N3	P112
A11	P73	C5	P07	E13	P65	J2	IC/V_{PP}注	L10	P96	N4	V_{DD}
A12	P72	C6	P04	F1	P26	J3	P100	L11	BV_{SS}	N5	XT1
A13	AV_{SS}	C7	P01	F2	P27	J11	P52	L12	BV_{SS}	N6	V_{SS}
B1	P21	C8	P82	F3	P33	J12	P50	L13	BV_{SS}	N7	V_{SS}
B2	P14	C9	P77	F11	P63	J13	P51	M1	P106	N8	CLKOUT
B3	V_{SS}	C10	P74	F12	P61	K1	P101	M2	P111	N9	P91
B4	P12	C11	AV_{SS}	F13	P62	K2	P102	M3	P113	N10	P94
B5	P10	C12	P70	G1	P31	K3	P103	M4	V_{DD}	N11	P40
B6	P05	C13	AV_{REF}	G2	P32	K11	P46	M5	XT2	N12	P42
B7	P02	D1	V_{DD}	G3	P36	K12	P47	M6	X1	N13	P43

注　μPD703014A, 703014AY, 703015A, 703015AY, 703017A, 703017AY：IC（V_{SS}に直接接続してください）

μPD70F3017A, 70F3017AY：V_{PP}（通常動作モード時は，V_{SS}に接続してください）

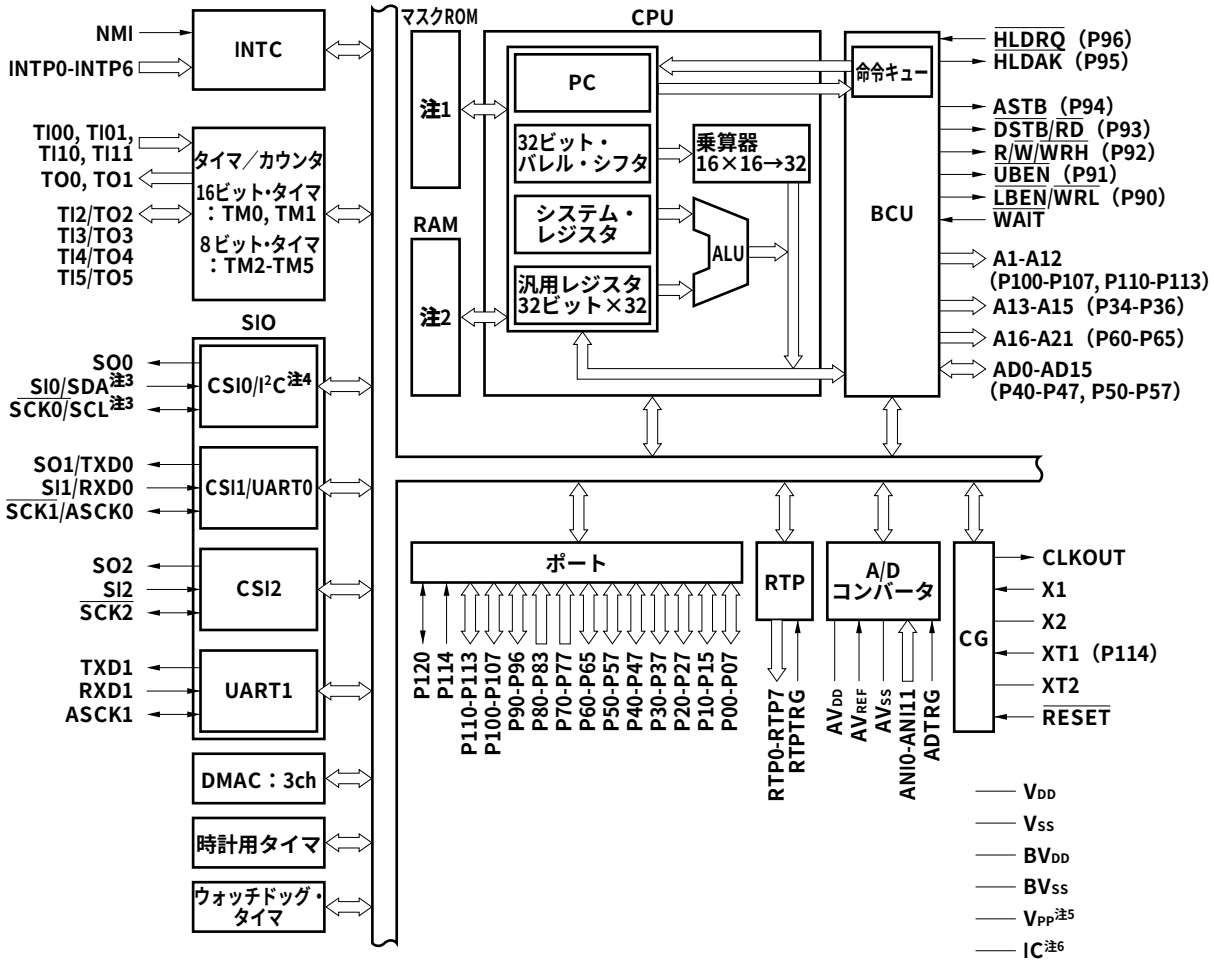
備考1．兼用端子の名称は省略しています。兼用端子は100ピン・プラスチックLQFPと同一です。

ただし，SCL, SDA端子は，μPD703014AY, 703015AY, 703017AY, 70F3017AYのみです。

2．D4端子は，直接V_{SS}に接続してください。

1.6　機能ブロック構成

★　1.6.1　内部ブロック図



注1．μPD703014A, 703014AY, 703014B, 703014BY：64 K**バイト（マスク**ROM）

μPD703015A, 703015AY, 703015B, 703015BY：128 K**バイト（マスク**ROM）

μPD703017A, 703017AY：256 K**バイト（マスク**ROM）

μPD70F3015B, 70F3015BY：128 K**バイト（フラッシュ・メモリ）**

μPD70F3017A, 70F3017AY：256 K**バイト（フラッシュ・メモリ）**

2．μPD703014A, 703014AY, 703014B, 703014BY, 703015A, 703015AY, 703015B, 703015BY, 70F3015B, 70F3015BY：4 K**バイト**

μPD703017A, 703017AY, 70F3017A, 70F3017AY：8 K**バイト**

3．SDA, SCL**端子は**μPD703014AY, 703014BY, 703015AY, 703015BY, 703017AY, 70F3015BY, 70F3017AY**のみ有効**

4．I^2C**機能は**μPD703014AY, 703014BY, 703015AY, 703015BY, 703017AY, 70F3015BY, 70F3017AY **のみ有効**

5．μPD70F3015B, 70F3015BY, 70F3017A, 70F3017AY

6．μPD703014A, 703014AY, 703014B, 703014BY, 703015A, 703015AY, 703015B, 703015BY, 703017A, 703017AY

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），32ビット・バレル・シフタ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

00000000H番地からマッピングされる64 K/128 K/256 Kバイトのマスク ROMまたは128 K/256 Kバイトのフラッシュ・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

FFFFE000H番地からマッピングされる4 Kバイトの RAM，または，FFFFD000H番地からマッピングされる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て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_{XX}）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。5種類（f_{XX}, $f_{XX}/2$, $f_{XX}/4$, $f_{XX}/8$, f_{XT}）のクロックを生成して，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。

(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チャネル内蔵しています。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) 時計用タイマ

サブクロック（32.768 kHz）またはメイン・クロック（16.777 MHz）から時計カウント用の基準時間（0.5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。

(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）が発生します。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）が発生します。

(10) シリアル・インタフェース（SIO）

V850/SA1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）とクロック同期式シリアル・インタフェース（CSI0-CSI2），I^2Cバス・インタフェースをあわせて4チャネル備えています。このうち1チャネルはUARTとCSIの切り替えが可能，別の1チャネルはCSIとI^2Cの切り替えが可能で，2チャネルはUARTとCSIにそれぞれ固定になっています。

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

CSI0-CSI2は，SO0-SO2, SI0-SI2, $\overline{SCK0}$-$\overline{SCK2}$端子によりデータ転送を行います。

★ I^2Cは，SDA, SCL端子によりデータ転送を行います。I^2Cは，μPD703014AY, 703014BY, 703015AY, 703015BY, 703017AY, 70F3015BY, 70F3017AYのみ内蔵しています。

UARTだけ専用ボー・レート・ジェネレータを2チャネル内蔵しています。

(11) A/Dコンバータ

12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) DMAコントローラ

3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と内蔵周辺I/O間でデータを転送します。

(13) リアルタイム出力ポート（RTP）

あらかじめ設定しておいた8ビット・データを，外部トリガ信号またはタイマのコンペア・レジスタの一致信号により出力ラッチに転送する，リアルタイム出力機能です。4ビット×2chとしても使用できます。

(14) 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	8ビット入出力	汎用ポート	NMI，外部割り込み，A/Dコンバータ・トリガ，RTPトリガ
P1	6ビット入出力		シリアル・インタフェース
P2	8ビット入出力		シリアル・インタフェース，タイマ入出力
P3	8ビット入出力		タイマ入出力，外部アドレス・バス
P4	8ビット入出力		外部アドレス／データ・バス
P5	8ビット入出力		
P6	6ビット入出力		外部アドレス・バス
P7	8ビット入力		A/Dコンバータ・アナログ入力
P8	4ビット入力		
P9	7ビット入出力		外部バス・インタフェース制御信号入出力
P10	8ビット入出力		リアルタイム出力ポート，外部アドレス・バス
P11	4ビット入出力， 1ビット入力		外部アドレス・バス，サブクロック入力
P12	1ビット入出力		ウエイト制御

1.7　端子機能

(1) バス・インタフェース端子

端子名称	入出力	PULL	機　　能
A1-A15	出力	なし	外部にメモリを拡張する場合の下位アドレス・バス
A16-A21	出力	なし	外部にメモリを拡張する場合の上位アドレス・バス
AD0-AD15	入出力	なし	外部にメモリを拡張する場合の16ビット・マルチプレクスト・アドレス／データ・バス
ASTB	出力	なし	外部アドレス・ストローブ信号出力
$\overline{\text{DSTB}}$	出力	なし	外部データ・ストローブ信号出力
$\overline{\text{HLDAK}}$	出力	なし	バス・ホールド・アクノリッジ出力
$\overline{\text{HLDRQ}}$	入力	なし	バス・ホールド要求入力
$\overline{\text{LBEN}}$	出力	なし	外部データ・バスの下位バイト・イネーブル信号出力
$\overline{\text{RD}}$	出力	なし	リード・ストローブ信号出力
R/$\overline{\text{W}}$	出力	なし	外部リード／ライト・ステータス出力
$\overline{\text{UBEN}}$	出力	なし	外部データ・バスの上位バイト・イネーブル信号出力
$\overline{\text{WAIT}}$	入力	なし	バス・サイクルにウエイトを挿入する制御信号入力
$\overline{\text{WRH}}$	出力	なし	外部データ・バスの上位バイト・ライト・ストローブ信号出力
$\overline{\text{WRL}}$	出力	なし	外部データ・バスの下位バイト・ライト・ストローブ信号出力

備考　PULL：内蔵プルアップ抵抗

(2) 周辺I/O端子　(1/2)

端子名称	入出力	PULL	機　　能
TI00	入力	あり	TM0の外部カウント・クロック入力／外部キャプチャ・トリガ入力
TI01	入力	あり	TM0の外部キャプチャ・トリガ入力
TI10	入力	あり	TM1の外部カウント・クロック入力／外部キャプチャ・トリガ入力
TI11	入力	あり	TM1の外部キャプチャ・トリガ入力
TI2-TI5	入力	あり	TM2-TM5の外部カウント・クロック入力
TO0-TO5	出力	あり	TM0-TM5のパルス信号出力
INTP0-INTP3	入力	あり	外部割り込み要求入力（アナログ・ノイズ除去）
INTP4-INTP6			外部割り込み要求入力（ディジタル・ノイズ除去）
SI0-SI2	入力	あり	CSI0-CSI2のシリアル受信データ入力（3線式）
SO0-SO2	出力	あり	CSI0-CSI2のシリアル送信データ出力（3線式）
$\overline{\text{SCK0}}$-$\overline{\text{SCK2}}$	入出力	あり	CSI0-CSI2のシリアル・クロック入出力（3線式）
SCL[注]	入出力	あり	I^2Cのシリアル・クロック入出力
SDA[注]	入出力	あり	I^2Cのシリアル送受信データ入出力

★　**注**　μPD703014AY, 703014BY, 703015AY, 703015BY, 703017AY, 70F3015BY, 70F3017AYのみ

備考　PULL：内蔵プルアップ抵抗

(2/2)

端子名称	入出力	PULL	機　　能
ASCK0	入力	あり	UART0のシリアル・ボー・レート・クロック入力
ASCK1	入力	あり	UART1のシリアル・ボー・レート・クロック入力
TXD0	出力	あり	UART0のシリアル送信データ出力
TXD1	出力	あり	UART1のシリアル送信データ出力
RXD0	入力	あり	UART0のシリアル受信データ入力
RXD1	入力	あり	UART1のシリアル受信データ入力
ANI0-ANI11	入力	なし	A/Dコンバータへのアナログ入力
ADTRG	入力	あり	A/Dコンバータ外部トリガ入力
NMI	入力	あり	ノンマスカブル割り込み要求入力（アナログ・ノイズ除去）
RTP0-RTP7	出力	あり	リアルタイム出力ポート
RTPTRG	入力	あり	RTP外部トリガ入力

備考　PULL：内蔵プルアップ抵抗

(3) 電源，クロック，リセット端子

端子名称	入出力	PULL	機　　能
CLKOUT	出力	－	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
X1	入力	なし	メイン・クロック用発振子接続
X2	－		
XT1	入力	なし	サブクロック用発振子接続
XT2	－		
IC[注1]	－	－	内部接続
AV_{REF}	入力	－	A/Dコンバータ用基準電圧入力
AV_{DD}	－	－	A/Dコンバータ用正電源供給
AV_{SS}	－	－	A/Dコンバータ用グランド電位
BV_{DD}	－	－	バス・インタフェース用正電源供給
BV_{SS}	－	－	バス・インタフェース用グランド電位
V_{DD}	－	－	正電源供給端子
V_{SS}	－	－	グランド電位
V_{PP}[注2]	－	－	プログラム書き込み／ベリファイ時の高電圧印加端子

★ **注**1. μPD703014A, 703014AY, 703014B, 703014BY, 703015A, 703015AY, 703015B, 703015BY, 703017A, 703017AY**のみ**

2. μPD70F3015B, 70F3015BY, 70F3017A, 70F3017AY**のみ**

備考　PULL：内蔵プルアップ抵抗

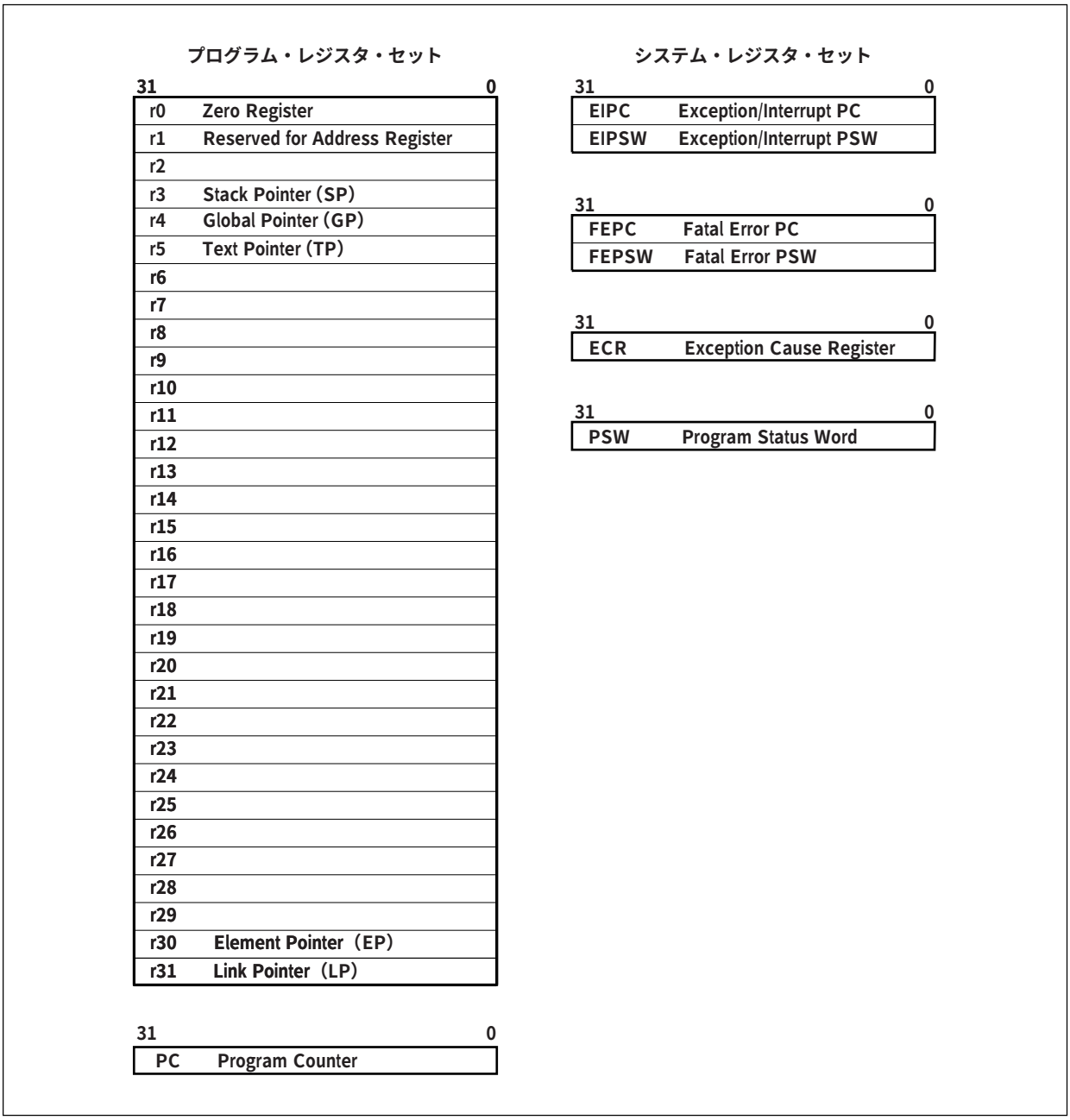
(4) ポート端子

端子名称	入出力	PULL	機　　能
P00-P07	入出力	あり	ポート0 8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
P10-P15	入出力	あり	ポート1 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
P20-P27	入出力	あり	ポート2 8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
P30-P37	入出力	あり	ポート3 8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
P40-P47	入出力	なし	ポート4 8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
P50-P57	入出力	なし	ポート5 8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
P60-P65	入出力	なし	ポート6 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
P70-P77	入力	なし	ポート7 8ビット入力ポート
P80-P83	入力	なし	ポート8 4ビット入力ポート
P90-P96	入出力	なし	ポート9 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
P100-P107	入出力	あり	ポート10 8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
P110-P113	入出力	あり	ポート11 5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
P114	入力	なし	P114は入力モードに固定
P120	入出力	なし	ポート12 1ビット入出力ポート

備考　PULL：内蔵プルアップ抵抗

1. 8　CPUレジスタ・セット

V850/SA1のCPU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ビット幅です。詳細はV850シリーズ　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。



1.8.1　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

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，使用後に元に戻してください。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[注]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

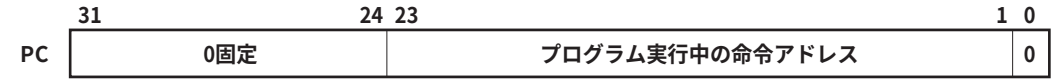
注　プログラム・コードを配置する領域

(2) 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します。下位24ビットが有効で，ビット31-24は0に固定されます。ビット23からビット24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。

リセット時：00000000H



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。

表1－3　システム・レジスタ番号

番号	システム・レジスタ名称	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。
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／割り込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・ステータス・ワードは，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。
6-31	予約		

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て行います。

(1) 割り込み要因レジスタ（ECR）

リセット時：00000000H

	31	16	15	0
ECR	FECC		EICC	

FECC	NMIの例外コード（例外コードはV850/SA1　ユーザーズ・マニュアル　ハードウエア編を参照）
EICC	例外／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）

リセット時：00000020H

	31	8	7	6	5	4	3	2	1	0
PSW	RFU		NP	EP	ID	SAT	CY	OV	S	Z

RFU	予約フィールドです（0に固定されています）。

NP	ノンマスカブル割り込み（NMI）処理状態
0	NMI処理中ではない
1	NMI処理中 NMIが受け付けられるとセット（1）されます。 多重割り込みを禁止します。

EP	例外処理状態
0	例外処理中ではない
1	例外処理中 例外の発生でセット（1）されます。 割り込み要求は受け付けます

ID	マスカブル割り込み処理の指定
0	マスカブル割り込みの受け付けを許可
1	マスカブル割り込みの受け付けを禁止 マスカブル割り込み要求を受け付けるとセット（1）されます。

SAT[注]	飽和演算命令の演算結果の飽和検出
0	飽和していない セット（1）されているときに飽和演算命令を実行した結果が飽和しなくても，クリア（0）はされません。クリア（0）するには，PSWに直接書き込んでください。
1	飽和した

CY	演算結果のキャリーまたはボロー検出
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

OV[注]	演算中のオーバフロー検出
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

S[注]	演算結果の正／負の検出
0	負ではない（正またはゼロであった）
1	負であった

備考　注の説明は次ページに記載しています。

（2/2）

Z	演算結果のゼロの検出
0	ゼロではなかった
1	ゼロであった

注　飽和演算時のOVビットとSビットの内容で，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ビットがセット（1）された場合だけ，SATビットはセット（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

V850/SA1は次の動作モードを備えます。

(1) 通常動作モード（シングルチップ・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，内蔵ROMに書き込まれた命令の処理を開始します。命令によりメモリ拡張モード・レジスタ（MM）を設定すると，外部メモリ領域に外部デバイスを接続できる外部拡張モードになります。

(2) フラッシュ・メモリ・プログラミング・モード

★ このモードは，μPD70F3015B, 70F3015BY, 70F3017A, 70F3017AYだけが備えています。V_{PP}端子にV_{PP}電圧を印加した場合に，内部フラッシュ・メモリの書き込み／消去ができます。

V_{PP}	動作モード
0	通常動作モード
7.8 V	フラッシュ・メモリ・プログラミング・モード
V_{DD}	設定禁止

1. 10　アドレス空間

V850/SA1では，次に示すように各領域を予約しています。

図1－1　メモリ・マップ

注　ここではμPD703017AY, 70F3017AYをあげています。
詳細はV850/SA1　**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。

1. 10. 1　割り込み／例外テーブル

V850/SA1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で，割り込み応答性を高速化しています。

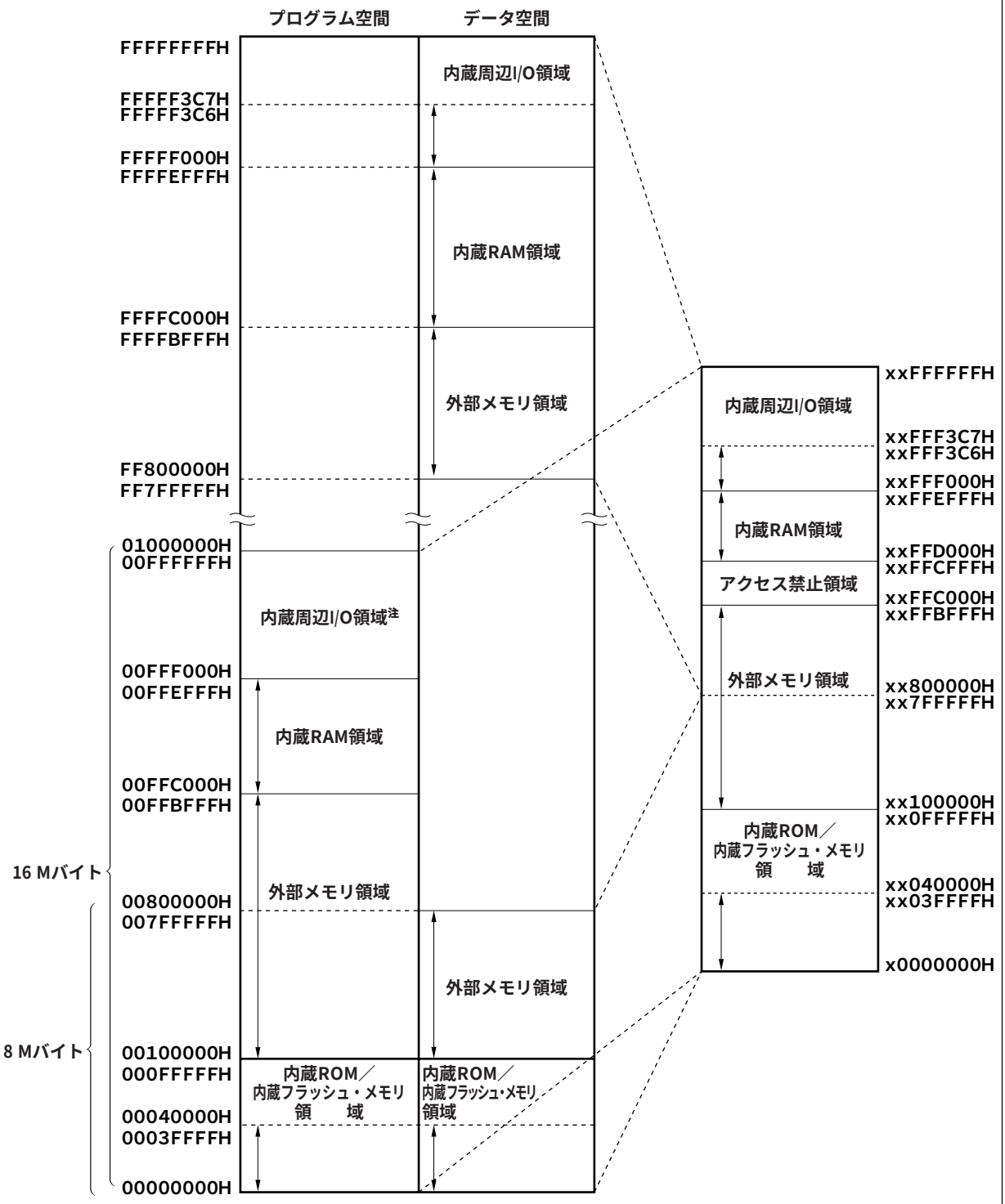
このハンドラ・アドレスの集合は割り込み／例外テーブルと呼ばれ，内蔵ROM／内蔵フラッシュ・メモリに置かれています。割り込み／例外要求が受け付けられると，ハンドラ・アドレスにジャンプし，そのメモリに置かれているプログラムを実行します。次に割り込み／例外要因と対応するアドレスを示します。

表1－4　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
00000000H	RESET
00000010H	NMI
00000020H	INTWDT
00000040H	TRAP0n（n = 0-F）
00000050H	TRAP1n（n = 0-F）
00000060H	ILGOP
00000080H	INTWDTM
00000090H	INTP0
000000A0H	INTP1
000000B0H	INTP2
000000C0H	INTP3
000000D0H	INTP4
000000E0H	INTP5
000000F0H	INTP6
00000100H	INTWTI
00000110H	INTTM00
00000120H	INTTM01
00000130H	INTTM10
00000140H	INTTM11
00000150H	INTTM2
00000160H	INTTM3
00000170H	INTTM4
00000180H	INTTM5
00000190H	INTIIC0[注]/INTCSI0
000001A0H	INTSER0
000001B0H	INTSR0/INTCSI1
000001C0H	INTST0
000001D0H	INTCSI2
000001E0H	INTSER1
000001F0H	INTSR1
00000200H	INTST1
00000210H	INTAD
00000220H	INTDMA0
00000230H	INTDMA1
00000240H	INTDMA2
00000250H	INTWT

★ **注**　μPD703014AY, 703014BY, 703015AY, 703015BY, 703017AY, 70F3015BY, 70F3017AY**のみ有効です。**

図1－2　推奨メモリ・マップ



注　プログラム領域として使用できません。

備考1.　↕　**は推奨使用領域です。**

2. **この図は，**μPD703017A, 703017AY, 70F3017A, 70F3017AY**の場合の推奨メモリ・マップです。**

1. 11　パイプライン

V850/SA1は，1命令を5つのステージで処理します。1つのステージに要する時間は1クロックで，1命令全体の処理としては5クロックかかりますが，5段のパイプラインで5つの命令を並列して処理することにより，ほとんどの命令を1クロックで実行することができます。

IF	ID	EX	MEM	WB

パイプラインの各ステージについて次に示します。

(1) IF（インストラクション・フェッチ）

命令をフェッチして，フェッチ・ポインタをインクリメントします。

(2) ID（インストラクション・デコード）

命令をデコードし，イミーディエト・データの作成，レジスタの読み出しを行います。

(3) EX（ALU，乗算器，バレル・シフタの実行）

デコードした命令を実行します。

(4) MEM（メモリ・アクセス）

対象となるアドレスのメモリをアクセスします。

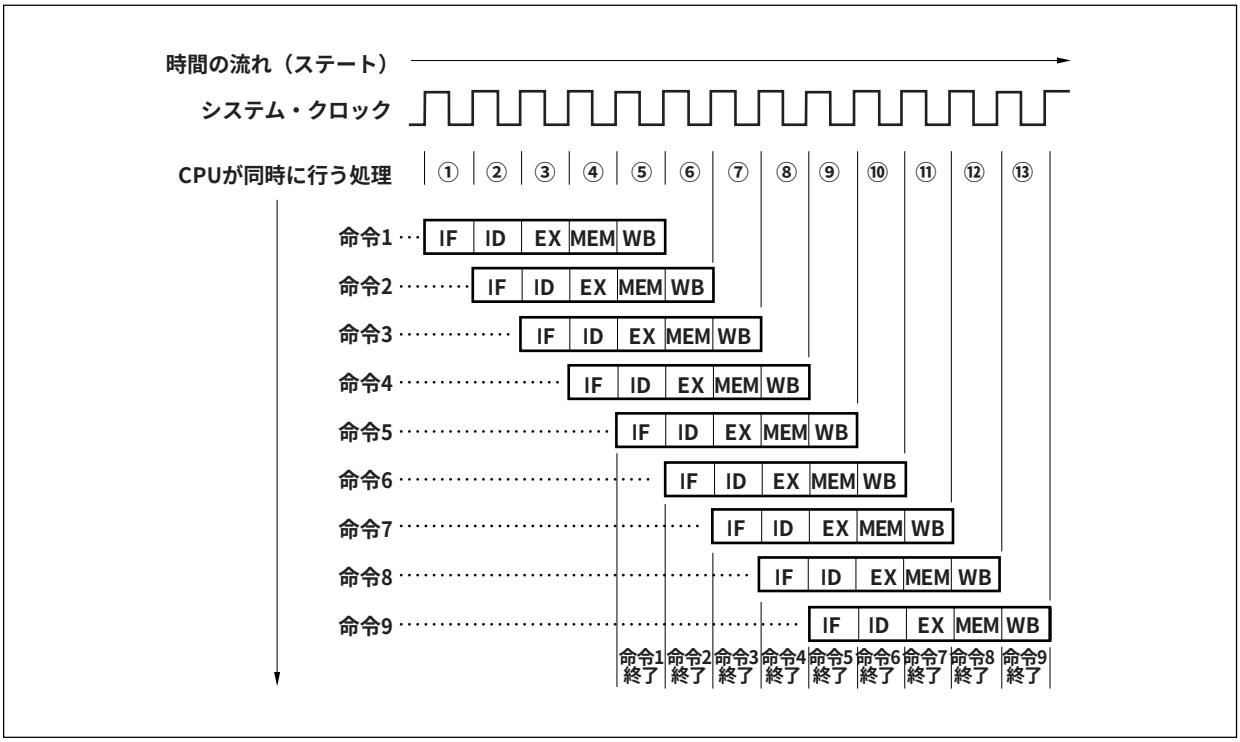
(5) WB（ライトバック）

実行した結果をレジスタに書き込みます。

1. 11. 1　パイプラインの動作

V850/SA1の命令実行手順は，IF（フェッチ）からWB（ライトバック）まで5つのステージで構成されています。各ステージの実行時間は，命令の種類やアクセスの対象となるメモリの種類などによって異なります。パイプラインの動作例として標準的な命令を9つ続けて実行したときのCPUの処理を図1－3に示します。

図1－3　パイプライン動作（標準的な命令を9つ続けて実行した場合）

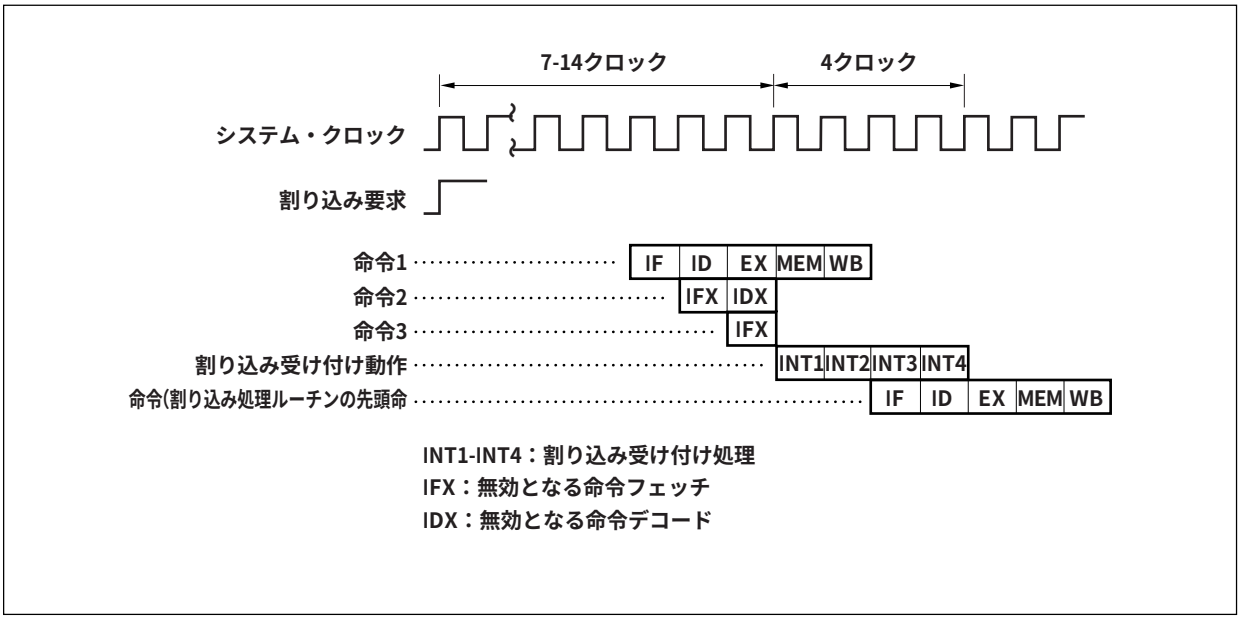


①-⑬はCPUのステートを示します。各ステートでは，命令nのライト・バック，命令n＋1のメモリ・アクセス，命令n＋2の実行，命令n＋3のデコード，命令n＋4のフェッチが同時に行われます。標準的な命令では，フェッチからライトバックまで5クロックかかります。しかし，同時に5命令を処理できるため，標準的な命令は平均1クロックごとに実行できます。

1. 11. 2　割り込み発生時のパイプライン動作

割り込みが発生したときのパイプラインの動作を図1－4に示します。割り込み受け付け動作には4クロックかかります。EXステージまで処理が進んだ命令は割り込み受け付け動作中も継続します。EXステージに満たない命令は一度無効となり，割り込み処理終了後にIFステージから再度，開始されます。

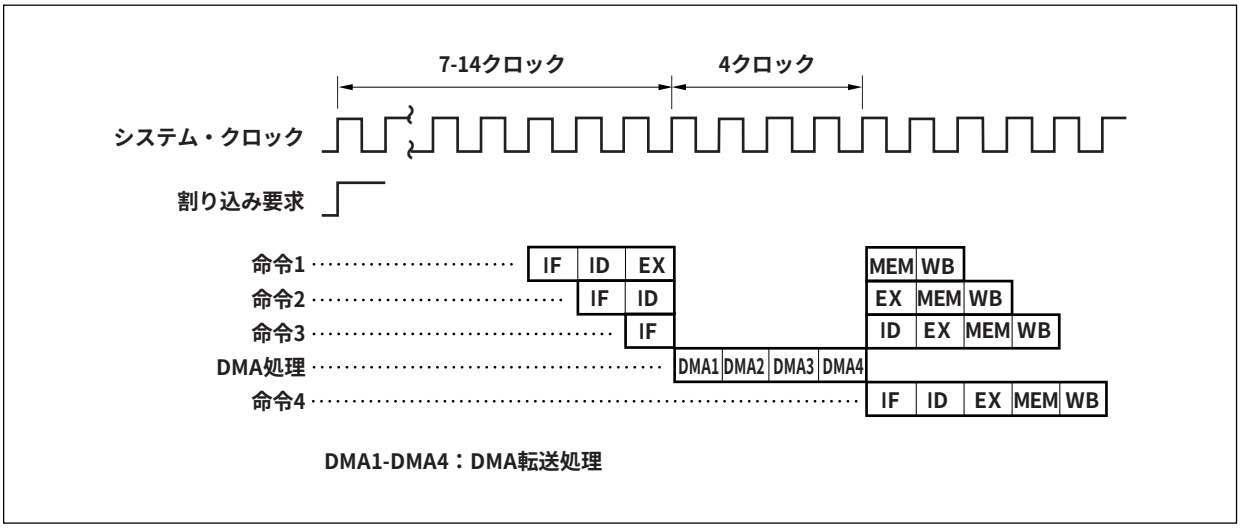
図1－4　パイプライン動作（割り込み要求受け付け時）



1. 11. 3　DMA転送時のパイプライン動作

DMA転送には4クロックかかります。DMA転送中はパイプラインの各ステージが中断して，DMA転送終了後に中断したステージの処理を継続ます。

図1－5　パイプライン動作（DMA転送時）



1. 12　パワー･セーブ機能

1. 12. 1　パワー・セーブ機能の動作モード

パワー・セーブ機能には，次に示すモードがあります。

各モードを組み合わせ，用途によって切り替えて使用すると，効果的な低消費電力システムを実現できます。

(1) HALTモード

このモードでは，クロック発振回路は動作を継続しますが，CPUの動作クロックが停止します。その他の内蔵周辺機能へのクロック供給は継続され，動作を継続します。システムのトータルの消費電力を低減できます。

(2) IDLEモード

クロック発振回路の動作を継続したままで，CPUの動作クロックと内蔵周辺機能の動作クロックを停止させることにより，システム全体を停止させるモードです。ただし，サブクロックは動作を継続し，サブクロックで動作している内蔵周辺機能にクロックを供給します。このモードからの解除時に，発振回路の発振安定時間などを確保する必要がないため，高速に通常動作に移行できます。

(3) ソフトウエアSTOPモード

メイン・クロック発振回路を停止させ，システム全体を停止させるモードです。サブククロックの供給は継続され，サブクロックで動作している内蔵周辺機能は動作を継続します。サブクロックを使用しない場合は，低消費電力状態になります。サブクロックでCPUを動作させている場合は，STOPモードの設定を禁止します。

(4) サブクロック動作

CPUクロックをサブクロック動作に設定し，PCCレジスタのMCKビットをセット（1）することにより，システム全体がサブクロックだけで動作する低消費電力モードになります。

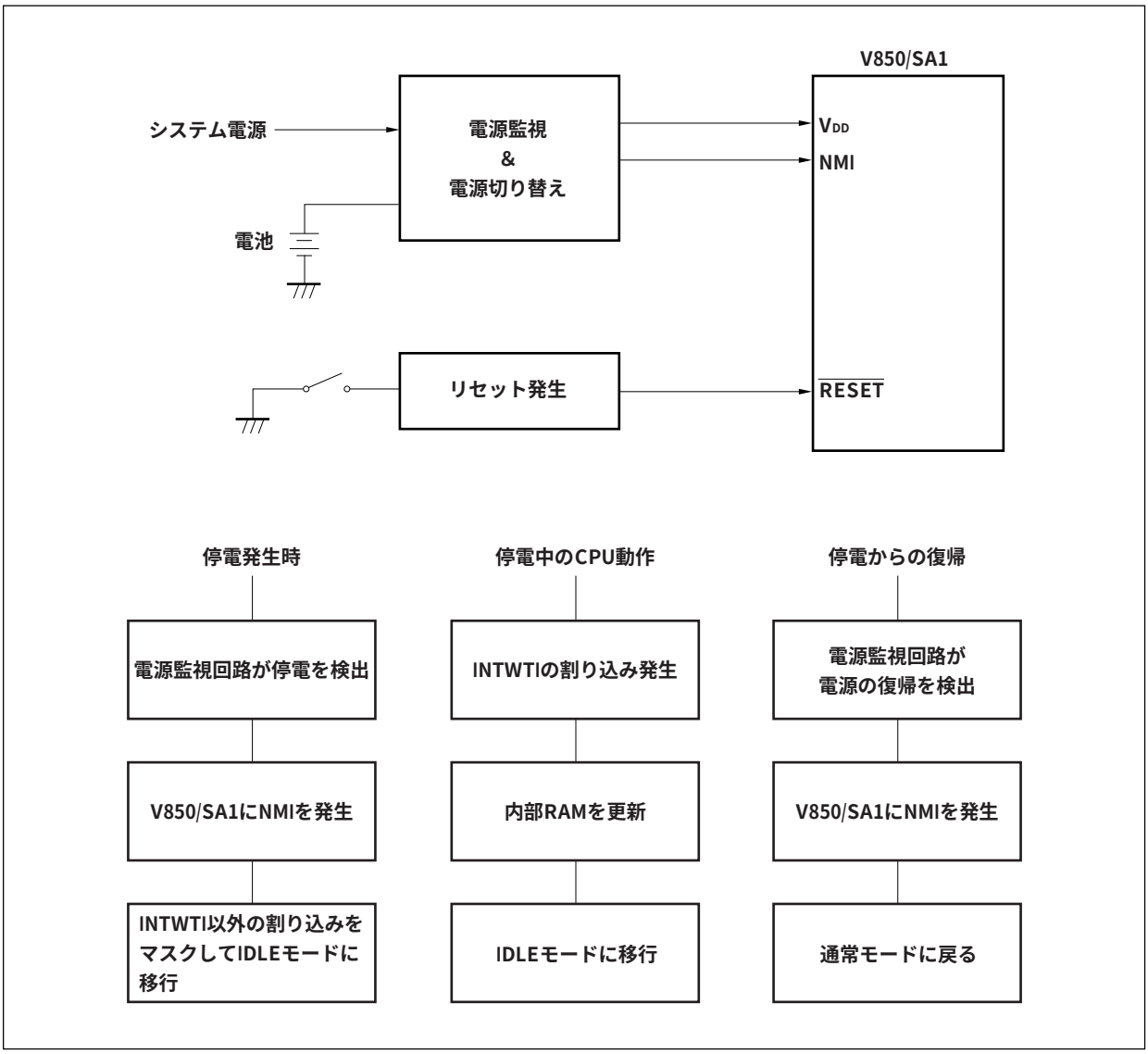
IDLEモードに設定するとCPUの動作クロックと一部の周辺機能（DMAC，BCU）が停止するため，消費電力を低減できます。

パワー・セーブ・モード	モードへの移行	モードの解除
HALTモード	HALT命令の実行	$\overline{\text{RESET}}$, NMI，マスクされていないマスカブル割り込み
IDLEモード	PSCレジスタの設定	$\overline{\text{RESET}}$, NMI，動作可能な内蔵周辺I/Oからのマスクされていないマスカブル割り込み，マスクされていない外部割り込み
ソフトウエアSTOPモード	PSCレジスタの設定	$\overline{\text{RESET}}$, NMI，動作可能な内蔵周辺I/Oからのマスクされていないマスカブル割り込み，マスクされていない外部割り込み
サブクロック動作	PCCレジスタの設定	PCCレジスタの設定

1. 12. 2　パワー・セーブ機能を使用したシステム構成例

パワー・セーブ・モード時にも時計機能を有効にするシステム構成例を示します。図1－6に示す例では，停電の検出と復帰はNMIでV850/SA1に通知し，停電中は電池からV_{DD}が供給されます。V850/SA1の$\overline{\text{RESET}}$端子は別に用意するスイッチに接続し，パワー・オン・リセットは発生させないものとします。また，時刻は内部RAMに格納するものとします（回路例については3. 11　停電検出回路とバックアップ電源を参照してください）。

図1－6　パワー・セーブ機能を使用したシステム構成例



1. 13　周辺I/Oレジスタ

(1/5)

アドレス	機能レジスタ名称	略　号	R/W	操作可能ビット			リセット時
				1	8	16	
FFFFF000H	ポート0	P0	R/W	○	○		00H[注]
FFFFF002H	ポート1	P1	R/W	○	○		00H[注]
FFFFF004H	ポート2	P2	R/W	○	○		00H[注]
FFFFF006H	ポート3	P3	R/W	○	○		00H[注]
FFFFF008H	ポート4	P4	R/W	○	○		00H[注]
FFFFF00AH	ポート5	P5	R/W	○	○		00H[注]
FFFFF00CH	ポート6	P6	R/W	○	○		00H[注]
FFFFF00EH	ポート7	P7	R	○	○		不定
FFFFF010H	ポート8	P8	R	○	○		不定
FFFFF012H	ポート9	P9	R/W	○	○		00H[注]
FFFFF014H	ポート10	P10	R/W	○	○		00H[注]
FFFFF016H	ポート11	P11	R/W	○	○		00H[注]
FFFFF018H	ポート12	P12	R/W	○	○		00H[注]
FFFFF020H	ポート0モード・レジスタ	PM0	R/W	○	○		FFH
FFFFF022H	ポート1モード・レジスタ	PM1	R/W	○	○		3FH
FFFFF024H	ポート2モード・レジスタ	PM2	R/W	○	○		FFH
FFFFF026H	ポート3モード・レジスタ	PM3	R/W	○	○		FFH
FFFFF028H	ポート4モード・レジスタ	PM4	R/W	○	○		FFH
FFFFF02AH	ポート5モード・レジスタ	PM5	R/W	○	○		FFH
FFFFF02CH	ポート6モード・レジスタ	PM6	R/W	○	○		3FH
FFFFF032H	ポート9モード・レジスタ	PM9	R/W	○	○		7FH
FFFFF034H	ポート10モード・レジスタ	PM10	R/W	○	○		FFH
FFFFF036H	ポート11モード・レジスタ	PM11	R/W	○	○		1FH
FFFFF038H	ポート12モード・レジスタ	PM12	R/W	○	○		01H
FFFFF04CH	メモリ拡張モード・レジスタ	MM	R/W	○	○		00H
FFFFF058H	ポート12モード・コントロール・レジスタ	PMC12	R/W	○	○		00H
FFFFF060H	データ・ウエイト・コントロール・レジスタ	DWC	R/W			○	FFFFH
FFFFF062H	バス・サイクル・コントロール・レジスタ	BCC	R/W			○	AAAAH
FFFFF064H	システム制御レジスタ	SYC	R/W	○	○		00H
FFFFF068H	メモリ・アドレス出力モード・レジスタ	MAM	W		○		00H
FFFFF070H	パワー・セーブ・コントロール・レジスタ	PSC	R/W	○	○		C0H
FFFFF074H	プロセッサ・クロック・コントロール・レジスタ	PCC	R/W	○	○		03H
FFFFF078H	システム・ステータス・レジスタ	SYS	R/W	○	○		00H
FFFFF080H	プルアップ抵抗オプション・レジスタ0	PU0	R/W	○	○		00H
FFFFF082H	プルアップ抵抗オプション・レジスタ1	PU1	R/W	○	○		00H
FFFFF084H	プルアップ抵抗オプション・レジスタ2	PU2	R/W	○	○		00H
FFFFF086H	プルアップ抵抗オプション・レジスタ3	PU3	R/W	○	○		00H
FFFFF094H	プルアップ抵抗オプション・レジスタ10	PU10	R/W	○	○		00H
FFFFF096H	プルアップ抵抗オプション・レジスタ11	PU11	R/W	○	○		00H

注　リセットにより入力モードに初期化されるので，リード時は端子レベルを読み出します。00H**に初期化されるのは出力ラッチです。**

(2/5)

アドレス	機能レジスタ名称	略　号	R/W	操作可能ビット			リセット時
				1	8	16	
FFFFF0A2H	ポート1ファンクション・レジスタ	PF1	R/W	○	○		00H
FFFFF0A4H	ポート2ファンクション・レジスタ	PF2	R/W	○	○		00H
FFFFF0B4H	ポート10ファンクション・レジスタ	PF10	R/W	○	○		00H
FFFFF0C0H	立ち上がりエッジ指定レジスタ0	EGP0	R/W	○	○		00H
FFFFF0C2H	立ち下がりエッジ指定レジスタ0	EGN0	R/W	○	○		00H
FFFFF100H	割り込み制御レジスタ	WDTIC	R/W	○	○		47H
FFFFF102H	割り込み制御レジスタ	PIC0	R/W	○	○		47H
FFFFF104H	割り込み制御レジスタ	PIC1	R/W	○	○		47H
FFFFF106H	割り込み制御レジスタ	PIC2	R/W	○	○		47H
FFFFF108H	割り込み制御レジスタ	PIC3	R/W	○	○		47H
FFFFF10AH	割り込み制御レジスタ	PIC4	R/W	○	○		47H
FFFFF10CH	割り込み制御レジスタ	PIC5	R/W	○	○		47H
FFFFF10EH	割り込み制御レジスタ	PIC6	R/W	○	○		47H
FFFFF110H	割り込み制御レジスタ	WTIIC	R/W	○	○		47H
FFFFF112H	割り込み制御レジスタ	TMIC00	R/W	○	○		47H
FFFFF114H	割り込み制御レジスタ	TMIC01	R/W	○	○		47H
FFFFF116H	割り込み制御レジスタ	TMIC10	R/W	○	○		47H
FFFFF118H	割り込み制御レジスタ	TMIC11	R/W	○	○		47H
FFFFF11AH	割り込み制御レジスタ	TMIC2	R/W	○	○		47H
FFFFF11CH	割り込み制御レジスタ	TMIC3	R/W	○	○		47H
FFFFF11EH	割り込み制御レジスタ	TMIC4	R/W	○	○		47H
FFFFF120H	割り込み制御レジスタ	TMIC5	R/W	○	○		47H
FFFFF122H	割り込み制御レジスタ	CSIC0	R/W	○	○		47H
FFFFF124H	割り込み制御レジスタ	SERIC0	R/W	○	○		47H
FFFFF126H	割り込み制御レジスタ	CSIC1	R/W	○	○		47H
FFFFF128H	割り込み制御レジスタ	STIC0	R/W	○	○		47H
FFFFF12AH	割り込み制御レジスタ	CSIC2	R/W	○	○		47H
FFFFF12CH	割り込み制御レジスタ	SERIC1	R/W	○	○		47H
FFFFF12EH	割り込み制御レジスタ	SRIC1	R/W	○	○		47H
FFFFF130H	割り込み制御レジスタ	STIC1	R/W	○	○		47H
FFFFF132H	割り込み制御レジスタ	ADIC	R/W	○	○		47H
FFFFF134H	割り込み制御レジスタ	DMAIC0	R/W	○	○		47H
FFFFF136H	割り込み制御レジスタ	DMAIC1	R/W	○	○		47H
FFFFF138H	割り込み制御レジスタ	DMAIC2	R/W	○	○		47H
FFFFF13AH	割り込み制御レジスタ	WTIC	R/W	○	○		47H
FFFFF166H	インサービス・プライオリティ・レジスタ	ISPR	R	○	○		00H
FFFFF170H	コマンド・レジスタ	PRCMD	**W**		○		不定
FFFFF180H	DMA周辺I/Oアドレス・レジスタ0	DIOA0	R/W			○	不定
FFFFF182H	DMA内蔵RAMアドレス・レジスタ0	DRA0	R/W			○	不定
FFFFF184H	DMAバイト・カウント・レジスタ0	DBC0	R/W		○		不定
FFFFF186H	DMAチャネル・コントロール・レジスタ0	DCHC0	R/W	○	○		00H
FFFFF190H	DMA周辺I/Oアドレス・レジスタ1	DIOA1	R/W			○	不定
FFFFF192H	DMA内蔵RAMアドレス・レジスタ1	DRA1	R/W			○	不定

(3/5)

	アドレス	機能レジスタ名称	略　号	R/W	操作可能ビット			リセット時
					1	8	16	
	FFFFF194H	DMAバイト・カウント・レジスタ1	DBC1	R/W		○		不定
	FFFFF196H	DMAチャネル・コントロール・レジスタ1	DCHC1	R/W	○	○		00H
	FFFFF1A0H	DMA周辺I/Oアドレス・レジスタ2	DIOA2	R/W			○	不定
	FFFFF1A2H	DMA内蔵RAMアドレス・レジスタ2	DRA2	R/W			○	不定
	FFFFF1A4H	DMAバイト・カウント・レジスタ2	DBC2	R/W		○		不定
	FFFFF1A6H	DMAチャネル・コントロール・レジスタ2	DCHC2	R/W	○	○		00H
★	FFFFF1F4H	フラッシュ・メモリ・プログラミング・モード・コントロール・レジスタ	FLPMC	R/W	○	○		注1
	FFFFF200H	16ビット・タイマ・レジスタ0	TM0	R			○	0000H
	FFFFF202H	16ビット・キャプチャ／コンペア・レジスタ00	CR00	注2			○	0000H
	FFFFF204H	16ビット・キャプチャ／コンペア・レジスタ01	CR01	注2			○	0000H
	FFFFF206H	プリスケーラ・モード・レジスタ0	PRM0	R/W		○		00H
	FFFFF208H	16ビット・タイマ・モード・コントロール・レジスタ0	TMC0	R/W	○	○		00H
	FFFFF20AH	キャプチャ／コンペア・コントロール・レジスタ0	CRC0	R/W	○	○		00H
	FFFFF20CH	タイマ出力コントロール・レジスタ0	TOC0	R/W	○	○		00H
	FFFFF20EH	プリスケーラ・モード・レジスタ01	PRM01	R/W		○		00H
	FFFFF210H	16ビット・タイマ・レジスタ1	TM1	R			○	0000H
	FFFFF212H	16ビット・キャプチャ／コンペア・レジスタ10	CR10	注2			○	0000H
	FFFFF214H	16ビット・キャプチャ／コンペア・レジスタ11	CR11	注2			○	0000H
	FFFFF216H	プリスケーラ・モード・レジスタ1	PRM1	R/W		○		00H
	FFFFF218H	16ビット・タイマ・モード・コントロール・レジスタ1	TMC1	R/W	○	○		00H
	FFFFF21AH	キャプチャ／コンペア・コントロール・レジスタ1	CRC1	R/W	○	○		00H
	FFFFF21CH	タイマ出力コントロール・レジスタ1	TOC1	R/W	○	○		00H
	FFFFF21EH	プリスケーラ・モード・レジスタ11	PRM11	R/W		○		00H
	FFFFF240H	8ビット・カウンタ2	TM2	R		○		00H
	FFFFF242H	8ビット・コンペア・レジスタ2	CR20	R/W		○		00H
	FFFFF244H	タイマ・クロック選択レジスタ2	TCL2	R/W		○		00H
★	FFFFF246H	8ビット・タイマ・モード・コントロール・レジスタ2	TMC2	R/W	○	○		04H注3
	FFFFF24AH	16ビット・カウンタ23	TM23	R			○	0000H
	FFFFF24CH	16ビット・コンペア・レジスタ23	CR23	R/W			○	0000H
	FFFFF24EH	タイマ・クロック選択レジスタ21	TCL21	R/W		○		00H
	FFFFF250H	8ビット・カウンタ3	TM3	R		○		00H
	FFFFF252H	8ビット・コンペア・レジスタ3	CR30	R/W		○		00H
	FFFFF254H	タイマ・クロック選択レジスタ3	TCL3	R/W		○		00H
★	FFFFF256H	8ビット・タイマ・モード・コントロール・レジスタ3	TMC3	R/W	○	○		04H注3
	FFFFF25EH	タイマ・クロック選択レジスタ31	TCL31	R/W		○		00H
	FFFFF260H	8ビット・カウンタ4	TM4	R		○		00H

注1. **シングルチップ・モード時**　　：18H**または**38H

フラッシュ・メモリ・プログラミング・モード時：1CH**または**3CH

2. **コンペア・モード時**：R/W

キャプチャ・モード時：R

3. **ハードウエアの状態は**04H**に初期化されますが，リードすると**00H**が読み出されます。**

(4/5)

	アドレス	機能レジスタ名称	略　号	R/W	操作可能ビット			リセット時
					1	8	16	
	FFFFF262H	8ビット・コンペア・レジスタ4	CR40	R/W		○		00H
	FFFFF264H	タイマ・クロック選択レジスタ4	TCL4	R/W		○		00H
★	FFFFF266H	8ビット・タイマ・モード・コントロール・レジスタ4	TMC4	R/W	○	○		04H[注1]
	FFFFF26AH	16ビット・カウンタ45	TM45	R			○	0000H
	FFFFF26CH	16ビット・コンペア・レジスタ45	CR45	R/W			○	0000H
	FFFFF26EH	タイマ・クロック選択レジスタ41	TCL41	R/W		○		00H
	FFFFF270H	8ビット・カウンタ5	TM5	R		○		00H
	FFFFF272H	8ビット・コンペア・レジスタ5	CR50	R/W		○		00H
	FFFFF274H	タイマ・クロック選択レジスタ5	TCL5	R/W		○		00H
★	FFFFF276H	8ビット・タイマ・モード・コントロール・レジスタ5	TMC5	R/W	○	○		04H[注1]
	FFFFF27EH	タイマ・クロック選択レジスタ51	TCL51	R/W		○		00H
	FFFFF2A0H	シリアルI/Oシフト・レジスタ0	SIO0	R/W		○		00H
	FFFFF2A2H	シリアル動作モード・レジスタ0	CSIM0	R/W	○	○		00H
	FFFFF2A4H	シリアル・クロック選択レジスタ0	CSIS0	R/W	○	○		00H
	FFFFF2B0H	シリアルI/Oシフト・レジスタ1	SIO1	R/W		○		00H
	FFFFF2B2H	シリアル動作モード・レジスタ1	CSIM1	R/W	○	○		00H
	FFFFF2B4H	シリアル・クロック選択レジスタ1	CSIS1	R/W	○	○		00H
	FFFFF2C0H	シリアルI/Oシフト・レジスタ2	SIO2	R/W		○		00H
	FFFFF2C2H	シリアル動作モード・レジスタ2	CSIM2	R/W	○	○		00H
	FFFFF2C4H	シリアル・クロック選択レジスタ2	CSIS2	R/W	○	○		00H
	FFFFF300H	アシンクロナス・シリアル・インタフェース・モード・レジスタ0	ASIM0	R/W	○	○		00H
	FFFFF302H	アシンクロナス・シリアル・インタフェース・ステータス・レジスタ0	ASIS0	R	○	○		00H
	FFFFF304H	ボー・レート・ジェネレータ・コントロール・レジスタ0	BRGC0	R/W		○		00H
	FFFFF306H	送信シフト・レジスタ0	TXS0	W		○		FFH
	FFFFF308H	受信バッファ・レジスタ0	RXB0	R		○		FFH
	FFFFF30EH	ボー・レート・ジェネレータ・モード・コントロール・レジスタ0	BRGMC0	R/W		○		00H
	FFFFF310H	アシンクロナス・シリアル・インタフェース・モード・レジスタ1	ASIM1	R/W	○	○		00H
	FFFFF312H	アシンクロナス・シリアル・インタフェース・ステータス・レジスタ1	ASIS1	R	○	○		00H
	FFFFF314H	ボー・レート・ジェネレータ・コントロール・レジスタ1	BRGC1	R/W		○		00H
	FFFFF316H	送信シフト・レジスタ1	TXS1	W		○		FFH
	FFFFF318H	受信バッファ・レジスタ1	RXB1	R		○		FFH
	FFFFF31EH	ボー・レート・ジェネレータ・モード・コントロール・レジスタ1	BRGMC1	R/W		○		00H
	FFFFF320H	ボー・レート・ジェネレータ・モード・コントロール・レジスタ01	BRGMC01	R/W		○		00H
	FFFFF340H	IICコントロール・レジスタ0[注2]	IICC0	R/W	○	○		00H
	FFFFF342H	IIC状態レジスタ0[注2]	IICS0	R	○	○		00H
	FFFFF344H	IICクロック選択レジスタ0[注2]	IICCL0	R/W	○	○		00H
	FFFFF346H	スレーブ・アドレス・レジスタ0[注2]	SVA0	R/W		○		00H
	FFFFF348H	IICシフト・レジスタ0[注2]	IIC0	R/W		○		00H
	FFFFF34AH	IIC機能拡張レジスタ0[注2]	IICX0	R/W	○	○		00H
	FFFFF360H	時計用タイマ・モード・レジスタ	WTM	R/W	○	○		00H

注1.　**ハードウエアの状態は**04H**に初期化されますが，リードすると**00H**が読み出されます。**

★　2.　μPD703014AY, 703014BY, 703015AY, 703015BY, 703017AY, 70F3015BY, 70F3017AY**のみ**

(5/5)

アドレス	機能レジスタ名称	略　号	R/W	操作可能ビット			リセット時
				1	8	16	
FFFFF380H	発振安定時間選択レジスタ	OSTS	R/W		○		04H
FFFFF382H	ウォッチドッグ・タイマ・クロック選択レジスタ	WDCS	R/W		○		00H
FFFFF384H	ウォッチドッグ・タイマ・モード・レジスタ	WDTM	R/W	○	○		00H
FFFFF3A0H	リアルタイム出力バッファ・レジスタL	RTBL	R/W		○		00H
FFFFF3A2H	リアルタイム出力バッファ・レジスタH	RTBH	R/W		○		00H
FFFFF3A4H	リアルタイム出力ポート・モード・レジスタ	RTPM	R/W	○	○		00H
FFFFF3A6H	リアルタイム出力ポート・コントロール・レジスタ	RTPC	R/W	○	○		00H
FFFFF3C0H	A/Dコンバータ・モード・レジスタ	ADM	R/W	○	○		00H
FFFFF3C2H	アナログ入力チャネル指定レジスタ	ADS	R/W	○	○		00H
FFFFF3C4H	A/D変換結果レジスタ	ADCR	R			○	0000H
FFFFF3C6H	A/D変換結果レジスタH（上位8ビット）	ADCRH	R		○		00H

第2章　バス・インタフェース接続回路例

この章ではV850/SA1のバス・インタフェース関連の端子を使用した外部デバイスとの接続について説明します。2. 1　**バス・インタフェース接続回路例**では，接続の注意点と端子機能について説明します。2. 2　PROM**接続回路**以降からは，実際の回路例を示します。

- ○ タイミング生成部やデコーダ部の構成図は論理的説明に重点を置いて，一般的にゲートアレイやPLDを使用する部分でも，AND/OR/フリップフロップ等の部品だけを使用しています。したがって，タイミング生成部やデコーダ部では特に部品の型名は明記していません。実際の回路では，遅延時間とDC特性を考慮して部品を選定してください。
- ○ A1-A15は，V850/SA1のAD0-AD15を分離したアドレス信号か，またはV850/SA1のA1-A15のどちらか一方を使用していることを前提としています（2. 1. 1　**アドレス・バス**参照）。
- ○ 回路例は5 V系デバイスと接続する場合，DC特性を満足できるときでも，レベル変換デバイスを使用しています（ゲート類は除く）。実際の回路では，V850/SA1と接続するデバイスの双方の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もできます。

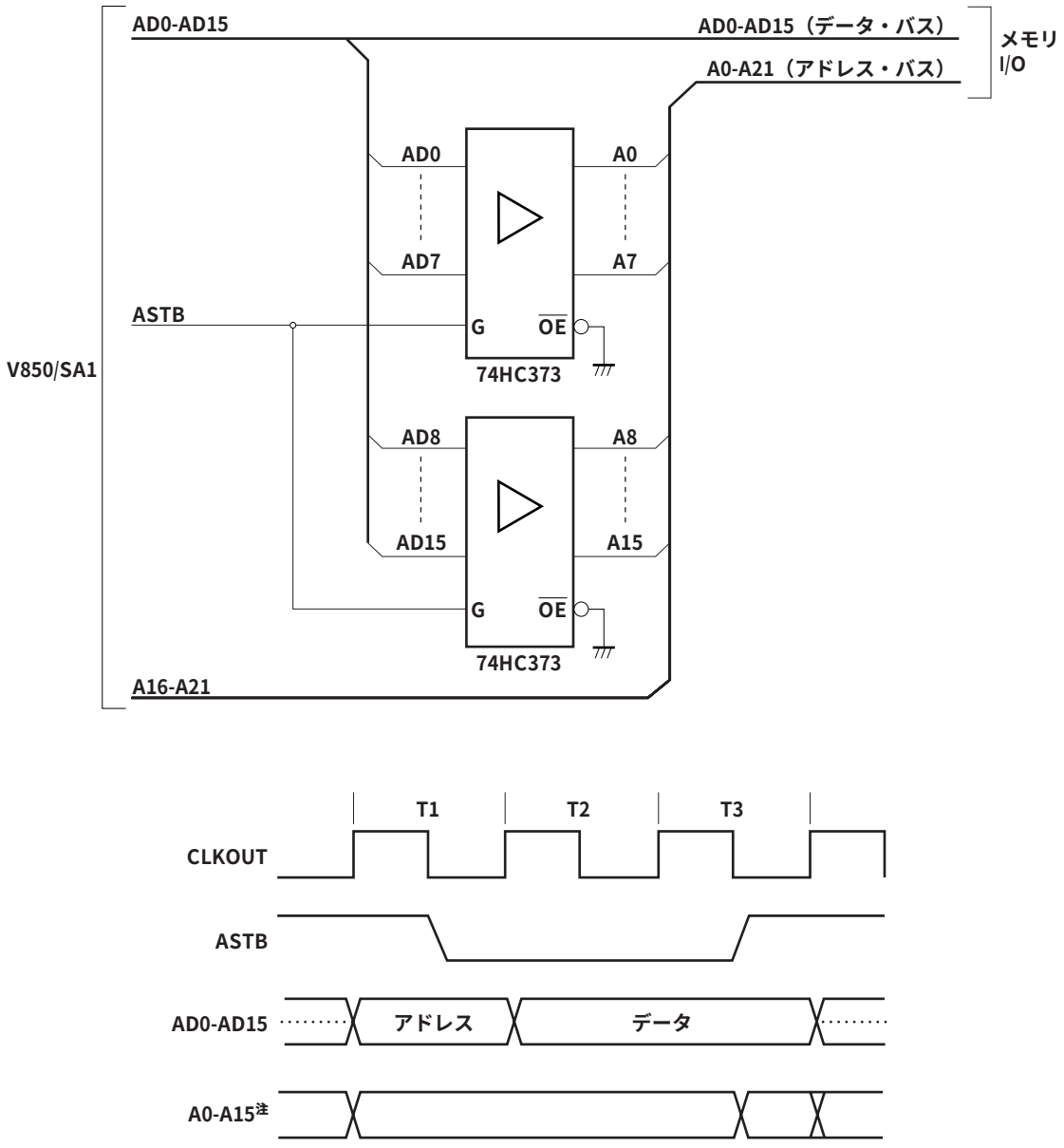
★ 2. 1　バス・インタフェース接続回路例

2. 1. 1　アドレス・バス

V850/SA1は，最大22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，セパレートされたアドレス・バスA1-A15（セパレートされたA0信号は配列していません）を使用する方法があります。

AD0-AD15を使用する方法では，アドレスとデータの分離が必要になります。アドレスはトランス・ペアレント・ラッチのデバイスを使用して，ASTB信号で作成するのが一般的な方法です。また，データ・バスは，AD0-AD15をそのまま使用しています。

図2－1　アドレス・バス／データ・バスの分離回路例



注　A0-A15は，ASTBにより分離された信号です。

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。

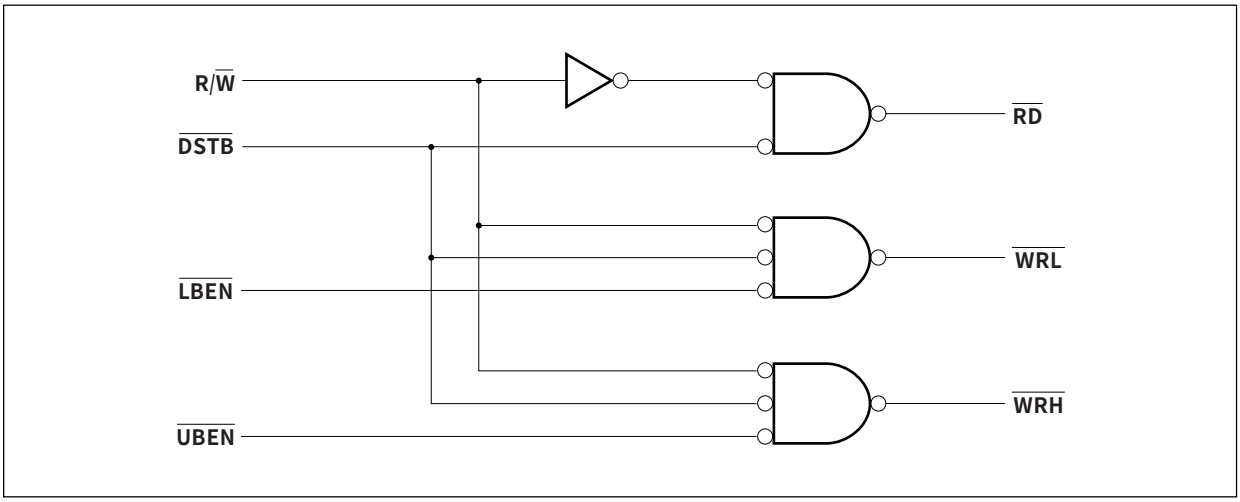
2. 1. 2　リード／ライト制御端子の動作モード

リード／ライト制御端子は次に示す2つのモードがあります。

- $\overline{\text{DSTB}}$, R/$\overline{\text{W}}$, $\overline{\text{LBEN}}$, $\overline{\text{UBEN}}$信号出力モード
- $\overline{\text{RD}}$, $\overline{\text{WRL}}$, $\overline{\text{WRH}}$, $\overline{\text{UBEN}}$信号出力モード

各モードは，システム制御レジスタ（SYC）で設定します。各端子の論理的関係を次に示します。

図2－2　リード／ライト制御端子の論理的関係

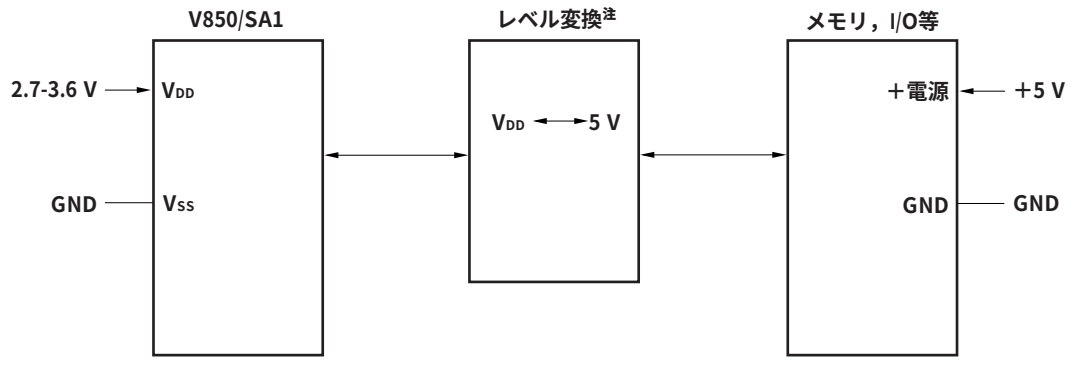


2. 1. 3　5 V系デバイスの接続

V850/SA1と5 Vシステム（デバイス）を接続する場合，V850/SA1のDC特性を守るためにレベル変換用デバイスを使用します。

接続するデバイスがV850/SA1のDC特性を満足できる場合は，レベル変換用デバイスは必要ありません。

図2－3　5 V系デバイスの接続



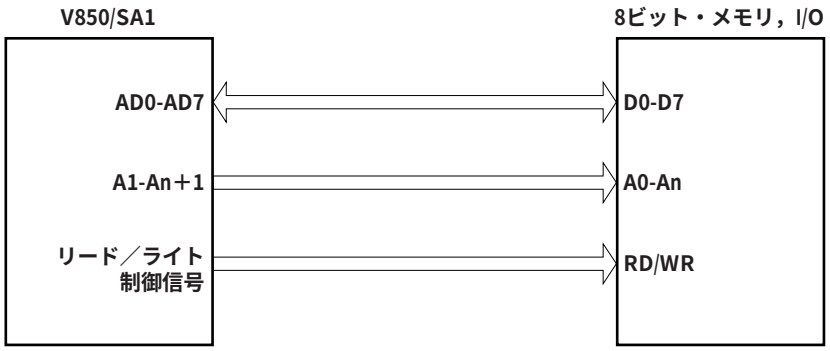
注　レベル変換用として使用できるデバイスは，LCXシリーズ，LVXシリーズ，FCTシリーズなどがあります。ICメーカから各種デバイスが販売されいます。

2. 1. 4　8ビット・バス幅ユニットと16ビット・バス幅ユニットの接続

V850/SA1のデータ・バス幅は16ビット固定ですが，データ空間（LD命令，ST命令，ビット操作命令でアクセスする空間）はバイト・アクセスができるため8ビット・バス幅でも，16ビット・バス幅でも構成できます。プログラム空間はすべて16ビット・アクセスとなりますので，16ビット・バス幅で構成する必要があります。アドレス・バスは，いずれの場合もA1信号から配列します。

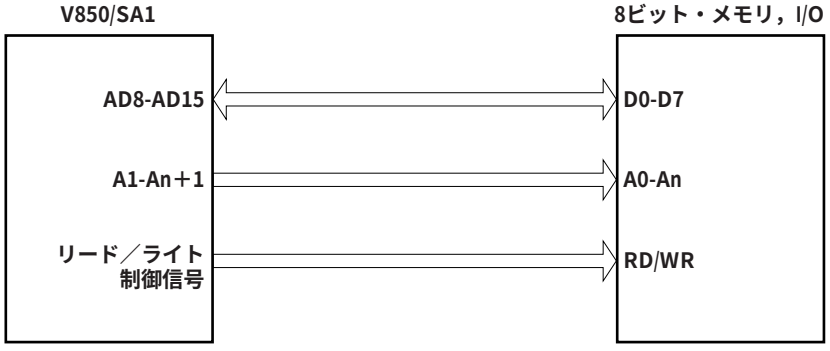
8ビット・バス幅ユニットと16ビット・バス幅ユニットの接続を次に示します。

図2－4　データ空間に配列する8ビット・バス幅ユニット（AD0-AD7に接続時）



注意　偶数アドレスに対するバイト・アクセスだけ有効になります。

図2－5　データ空間に配列する8ビット・バス幅ユニット（AD8-AD15に接続時）



注意　奇数アドレスに対するバイト・アクセスだけ有効になります。

図2－6　16ビット・バス幅ユニットの接続

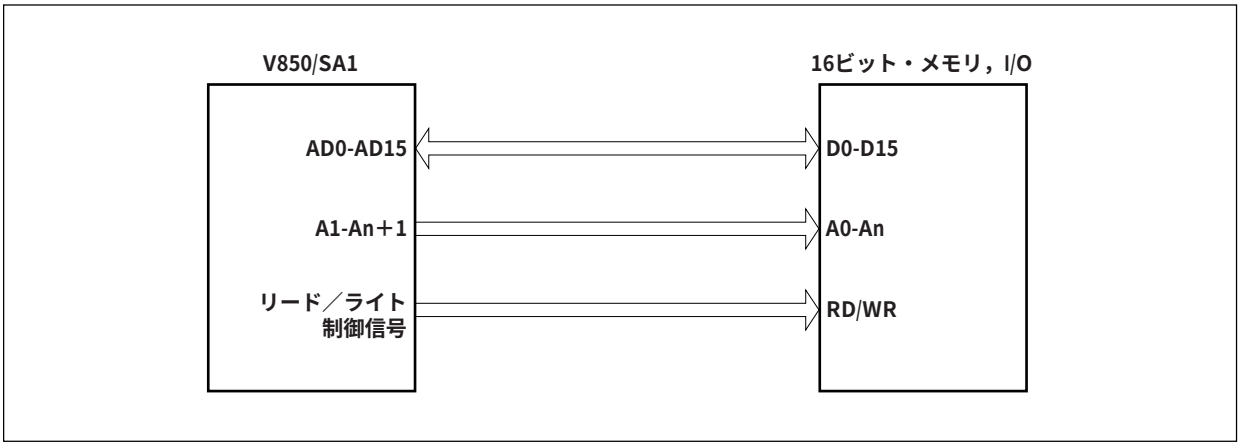
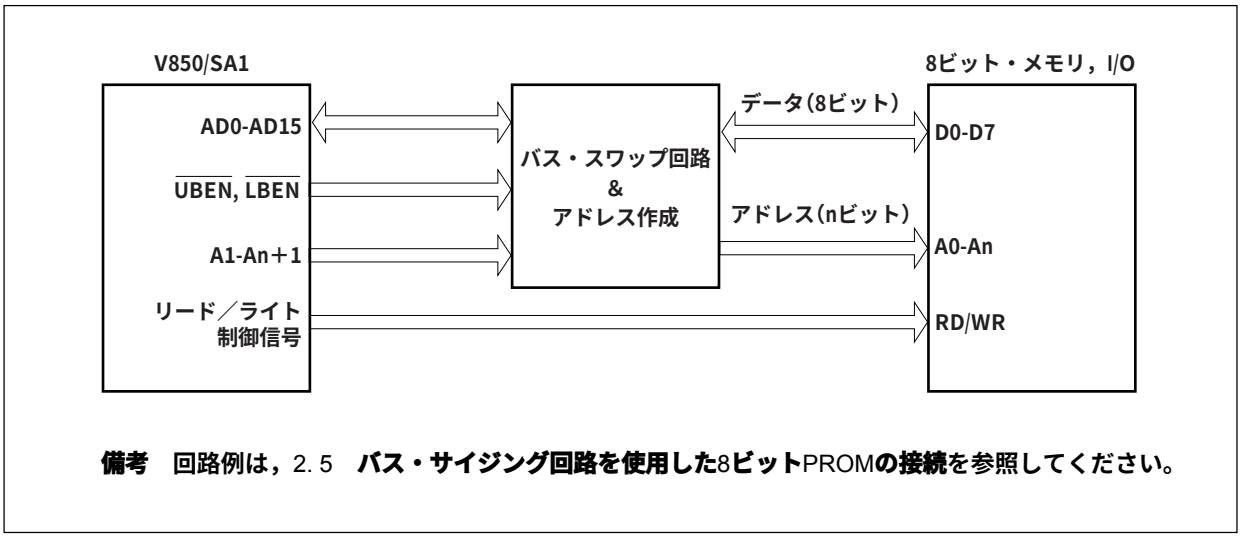


図2－7　プログラム空間／16ビット・アクセスを必要とする空間に配列する8ビット・バス幅ユニットの接続



備考　回路例は，2. 5　バス・サイジング回路を使用した8ビットPROMの接続を参照してください。

2. 2　PROM接続回路

27C1024（PROM，64 K×16ビット）を1つ使用して，128 Kバイトの外部ROM空間をV850/SA1に接続する例を次に示します。

【回路構成】

- ○ **システム・クロック：**16 MHz
- ○ **接続デバイス　　：**27C1024×1
- ○ **占有空間　　：外部メモリ空間の**00000H-1FFFFH（0番地からの128 Kバイト空間）

【DWC, BCC**の設定】**

- ○ **ウエイト設定　　：**0ウエイト
- ○ **アイドル・ステート　　：挿入しない**
- ○ **リード／ライト制御端子のモード：**$\overline{\text{DSTB}}$, R/$\overline{\text{W}}$, $\overline{\text{LBEN}}$, $\overline{\text{UBEN}}$信号出力

【回路方式】

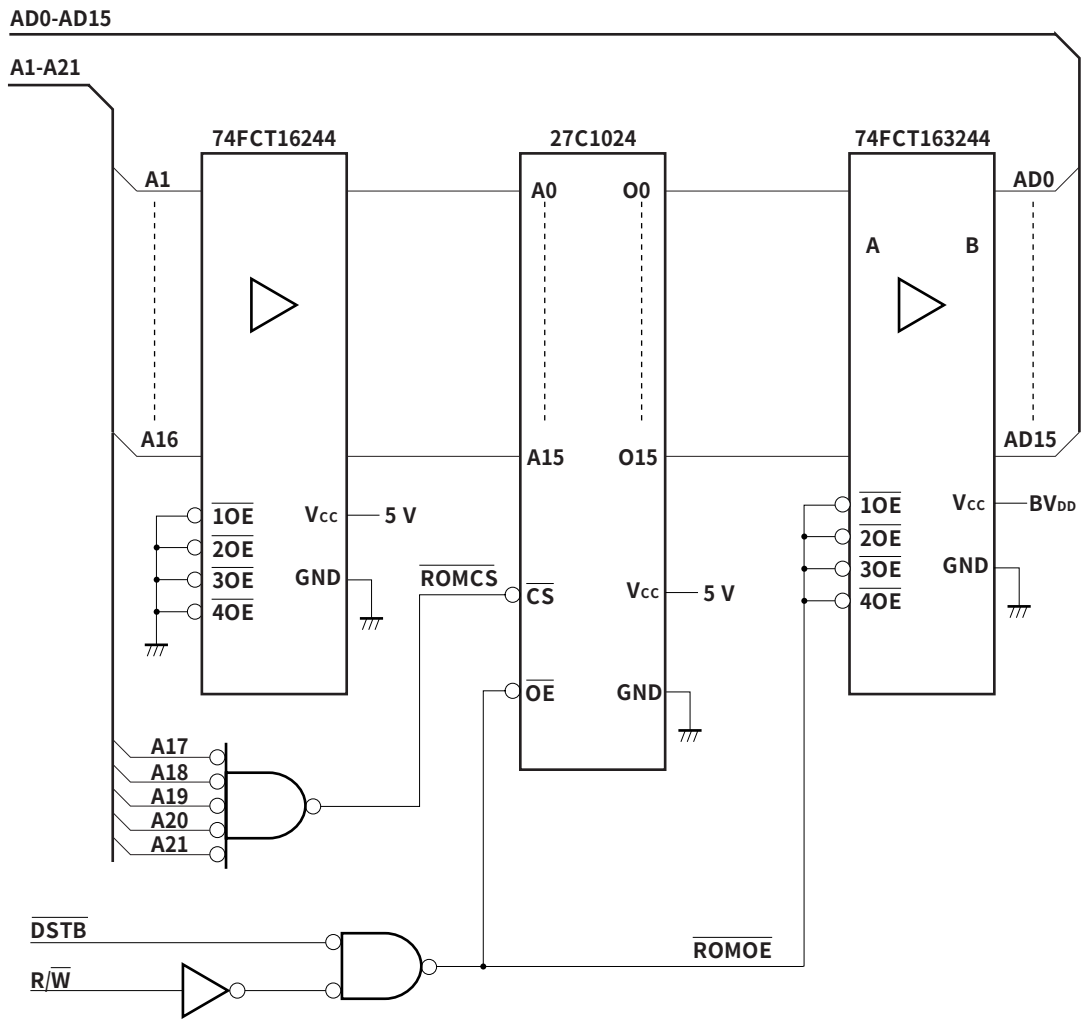
- ○ 27C1024は5 V製品なので，アドレス・バス，データ・バスの接続にレベル変換用デバイスを使用します。

アドレス・バス：74FCT16244
データ・バス　：74FCT163244

注意　$\overline{\text{CS}}$の生成に使用するデコーダとデータ・バッファの$\overline{\text{OE}}$制御は，V850/SA1と27C1024のDC特性を守れる部品を使用するものとします。

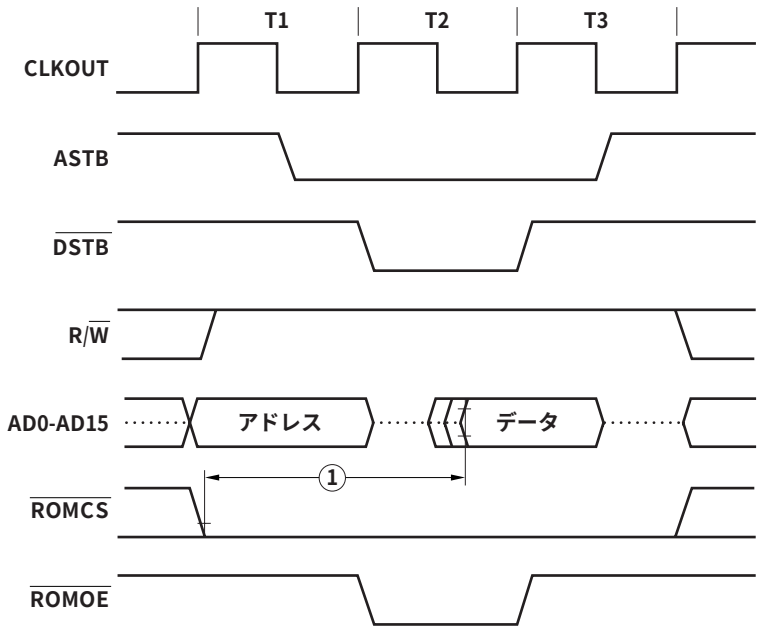
- ○ ROMの$\overline{\text{CS}}$信号（$\overline{\text{ROMCS}}$）はアドレス上位5ビットをデコードして生成します。
- ○ 74FCT163244の$\overline{\text{OE}}$信号（$\overline{\text{ROMOE}}$）は，R/$\overline{\text{W}}$信号と$\overline{\text{DSTB}}$信号で生成します（$\overline{\text{UBEN}}$, $\overline{\text{LBEN}}$信号は使用しません）。
- ○ $\overline{\text{WAIT}}$端子の制御は行いません（ハイ・レベルに固定）。

図2－8　PROM接続回路例



備考　27C1024：64 Kアドレス×16ビット（1 Mビット）

図2－9　PROMリードの動作



① 27C1024の$\overline{CS}$信号アクティブからのデータ出力遅延時間：85 ns（MAX.）

備考1．$\overline{ROMCS}$，$\overline{ROMOE}$は，この回路で生成される信号です。

2．破線はハイ・インピーダンスを示します。

2. 3　SRAM接続回路

V850/SA1にSRAMを接続する例を次に示します。

2. 3. 1　SRAM接続回路例1（8ビットSRAMの接続）では，8ビットSRAMを2つ接続して16ビット・バス空間にする例を示します。

2. 3. 2　SRAM接続回路例2（16ビットSRAMの接続）では，16ビットSRAMを1つ接続する例を示します。

2. 3. 1　SRAM接続回路例1（8ビットSRAMの接続）

μPD431000A（SRAM，128 K×8ビット）を2つ使用して，256 Kバイトの外部RAM空間をV850/SA1に接続する例を示します。

【回路構成】

- ○ V850/SA1の内部システム・クロック：16 MHz
- ○ 接続デバイス　：μPD431000A×2
- ○ 占有空間　：外部メモリ空間の040000H-07FFFFH
  （40000H番地からの256 Kバイト空間）

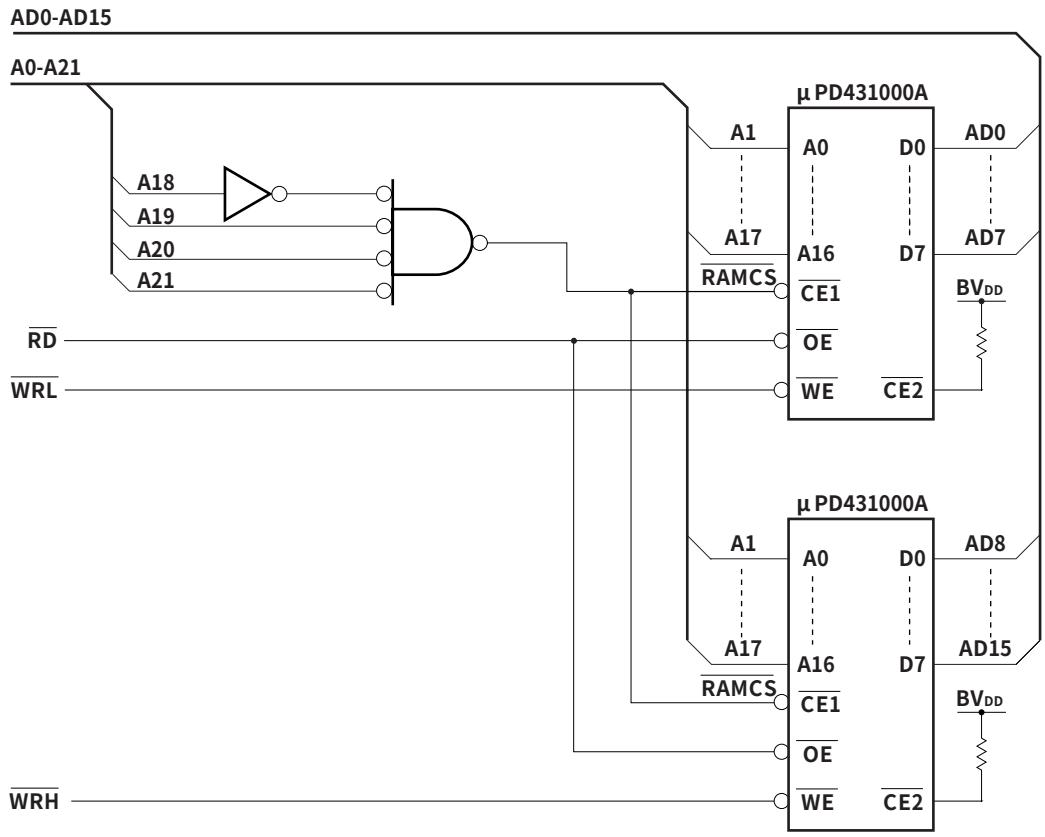
【DWC, BCCの設定】

- ○ ウエイト設定　：2ウエイト
- ○ アイドル・ステート　：挿入しない
- ○ リード／ライト制御端子のモード：$\overline{RD}$, $\overline{WRL}$, $\overline{WRH}$, $\overline{UBEN}$信号出力

【回路方式】

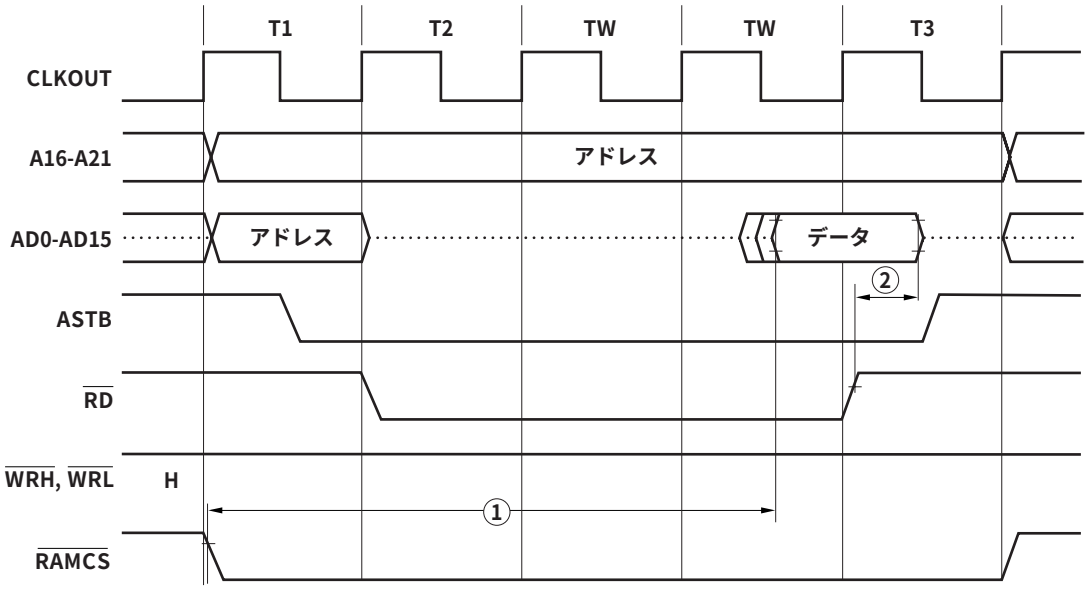
- ○ SRAMの$\overline{CS}$信号（$\overline{RAMCS}$）は，アドレス上位4ビットをデコードして生成。
- ○ SRAMの$\overline{OE}$端子は，$\overline{RD}$を接続。
- ○ AD0-AD7に接続したSRAMの$\overline{WE}$端子は$\overline{WRL}$を接続。
  AD8-AD15に接続したSRAMの$\overline{WE}$端子は$\overline{WRH}$を接続。
- ○ $\overline{WAIT}$端子の制御は行いません。

図2－10　μPD431000A接続回路例



備考　μPD431000A：128 Kアドレス×8ビット（1 Mビット）

図2－11　リード動作（μPD431000A）



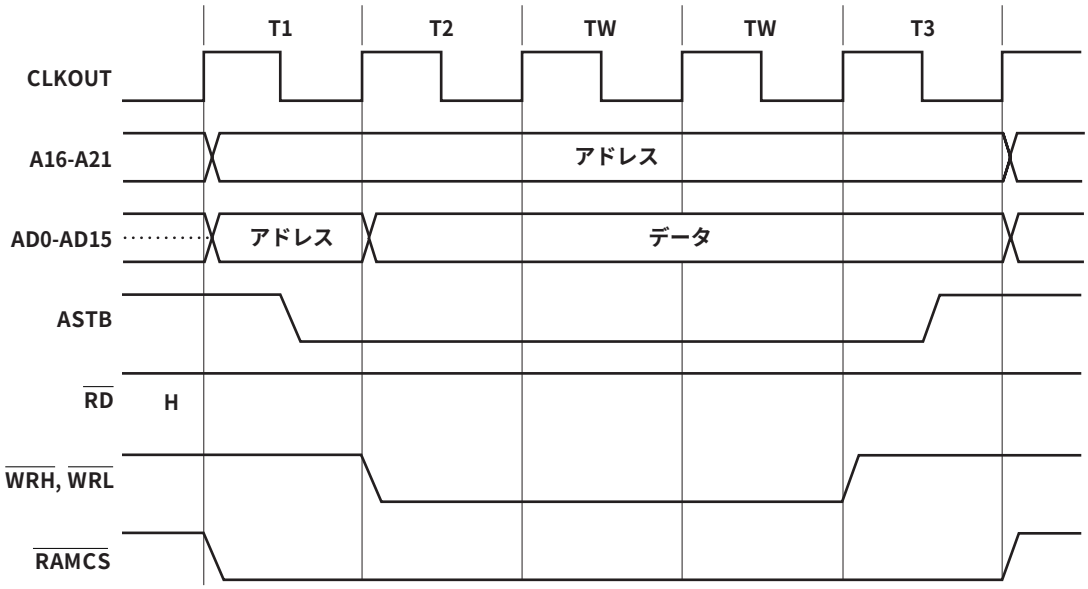
① $\overline{CS}$ **アクティブからの出力遅延時間：**150 ns（MAX.）

② $\overline{OE}$ **インアクティブからの出力フローティング遅延時間：**50 ns（MAX.）

備考1．$\overline{RAMCS}$**は，この回路で生成される信号です。**

2．**破線はハイ・インピーダンスを示します。**

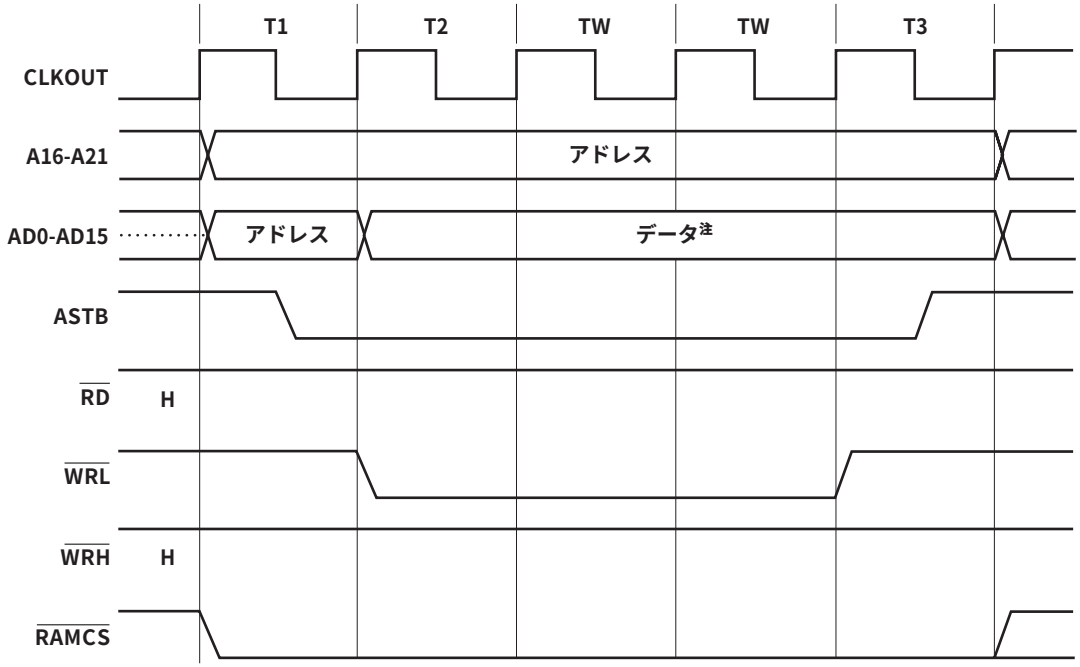
図2－12　ライト動作（μPD431000A）



備考1．$\overline{RAMCS}$**は，この回路で生成される信号です。**

2．**破線はハイ・インピーダンスを示します。**

図2－13　偶数アドレス・バイト・ライト（μPD431000A）

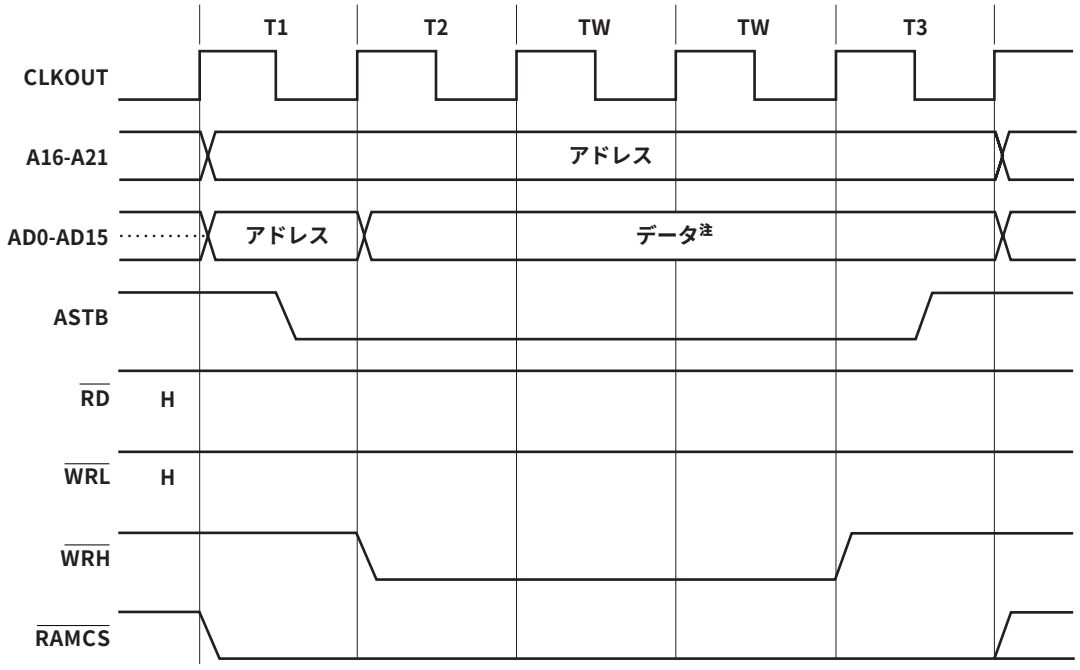


注　AD8-AD15は，無効データを出力します。

備考1．RAMCSは，この回路で生成される信号です。

2．破線はハイ・インピーダンスを示します。

図2－14　奇数アドレス・バイト・ライト（μPD431000A）



注　AD0-AD7は，無効データを出力します。

備考1．RAMCSは，この回路で生成される信号です。

2．破線はハイ・インピーダンスを示します。

2. 3. 2　SRAM接続回路例2（16ビットSRAMの接続）

μPD431016LE（SRAM，64 K×16ビット）を1つ使用して，128 Kバイトの外部RAM空間をV850/SA1に接続する例を示します。

【回路構成】

- ○ V850/SA1の内部システム・クロック：16 MHz
- ○ 接続デバイス　　：μPD431016LE×1
- ○ 占有空間　　：外部メモリ空間の040000H-07FFFFH
  （40000H番地からの256 Kバイト空間）

注意　60000H-7FFFFHは，40000H-5FFFFHのイメージになります。

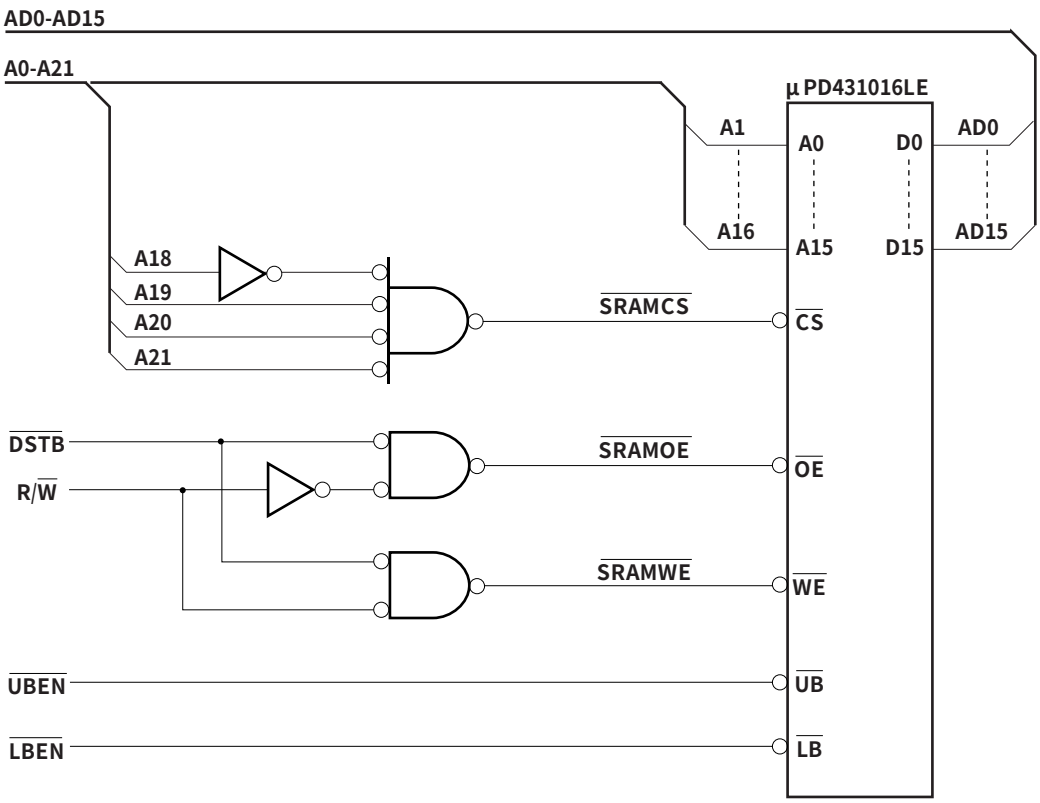
【DWC, BCCの設定】

- ○ ウエイト設定　　：0ウエイト
- ○ アイドル・ステート　　：挿入しない
- ○ リード／ライト制御端子のモード：$\overline{\text{DSTB}}$, R/$\overline{\text{W}}$, $\overline{\text{LBEN}}$, $\overline{\text{UBEN}}$信号出力

【回路方式】

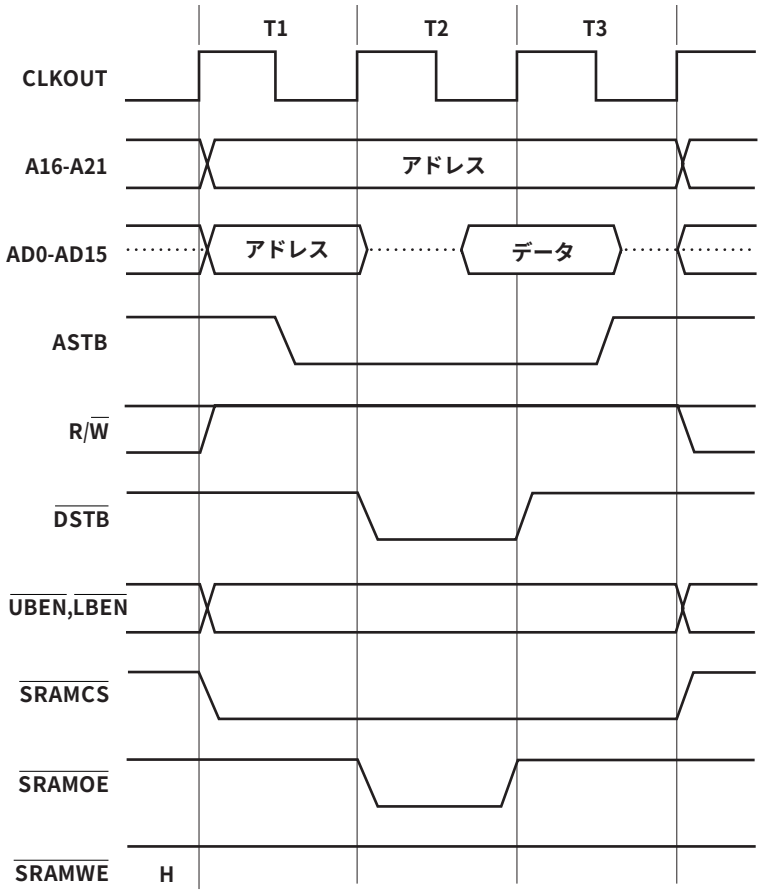
- ○ SRAMの$\overline{\text{CS}}$信号（$\overline{\text{SRAMCS}}$）は，アドレス上位4ビットをデコードして生成。
- ○ SRAMの$\overline{\text{OE}}$, $\overline{\text{WE}}$信号（$\overline{\text{SRAMOE}}$, $\overline{\text{SRAMWE}}$）は，R/$\overline{\text{W}}$, $\overline{\text{DSTB}}$信号で生成。
- ○ SRAMの$\overline{\text{UB}}$, $\overline{\text{LB}}$端子は，$\overline{\text{UBEN}}$, $\overline{\text{LBEN}}$端子をそれぞれ接続。
- ○ $\overline{\text{WAIT}}$端子の制御は行いません。

図2－15　μPD431016LE接続回路例



備考　μPD431016LE：64 Kアドレス×16ビット（1 Mビット）

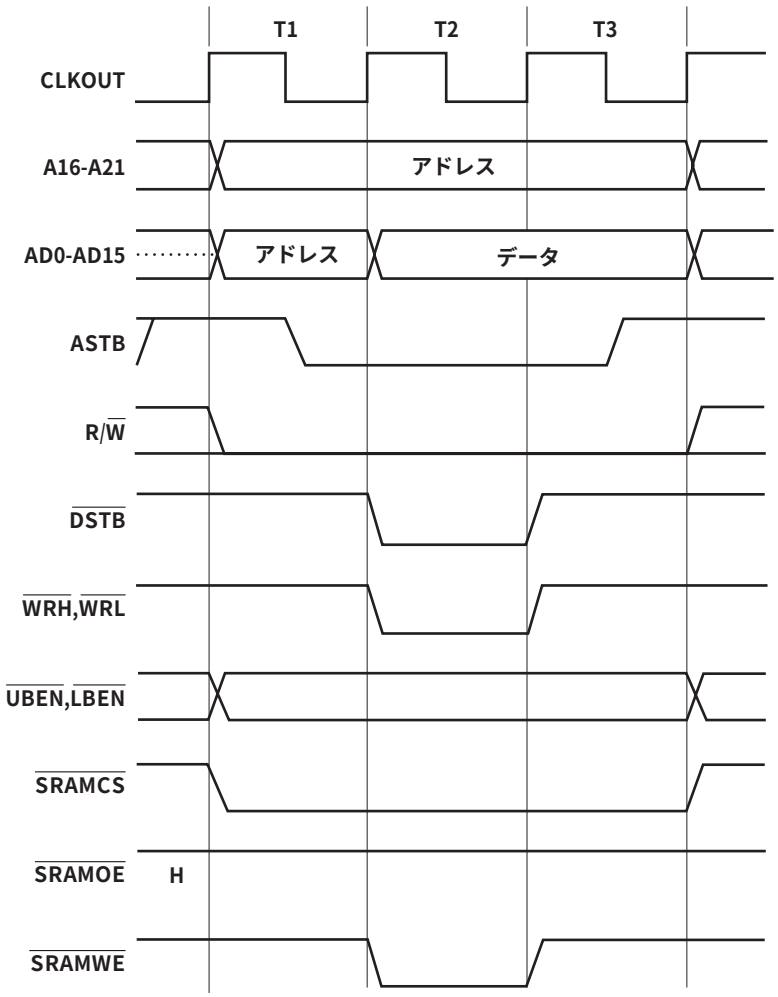
図2－16　リード動作（μPD431016LE）



備考1．$\overline{SRAMCS}$，$\overline{SRAMOE}$，$\overline{SRAMWE}$ **は，この回路で生成される信号です。**

2．**破線はハイ・インピーダンスを示します。**

図2－17　ライト動作（μPD431016LE）



備考1．SRAMCS，SRAMOE，SRAMWE **は，この回路で生成される信号です。**

2．**破線はハイ・インピーダンスを示します。**

2. 4　DRAM接続回路

μPD42S16160L（DRAM，1 M×16ビット）を1つ使用して，2Mバイトの外部RAM空間をV850/SA1に接続する例を示します。

DRAM接続回路例1（2. 4. 1参照）とDRAM接続回路例2（2. 4. 2参照）では，リフレッシュ回路が異なります。DRAMのリフレッシュ方式はいずれの場合もCBR方式で同じですが，DRAM接続回路例1ではリフレッシュ期間中，V850/SA1のバスはホールド状態になります。DRAM接続回路例2では外部バス・アービタによりバスの調停を行います。

2. 4. 1　DRAM接続回路例1（$\overline{\text{HLDRQ}}$端子を使用したリフレッシュ）

【回路構成】

- ○ V850/SA1の内部システム・クロック：16 MHz
- ○ 接続デバイス　：μPD42S16160L×1
- ○ 占有空間　：外部メモリ空間の200000H-3FFFFFH
- ○ サポートするサイクル　：リード・サイクル／アーリ・ライト・サイクル／CBRリフレッシュ・サイクル

【DWC, BCCの設定】

- ○ ウエイト設定　：1ウエイト
- ○ アイドル・ステート　：挿入しない
- ○ リード／ライト制御端子のモード：$\overline{\text{DSTB}}$, R/$\overline{\text{W}}$, $\overline{\text{LBEN}}$, $\overline{\text{UBEN}}$信号出力

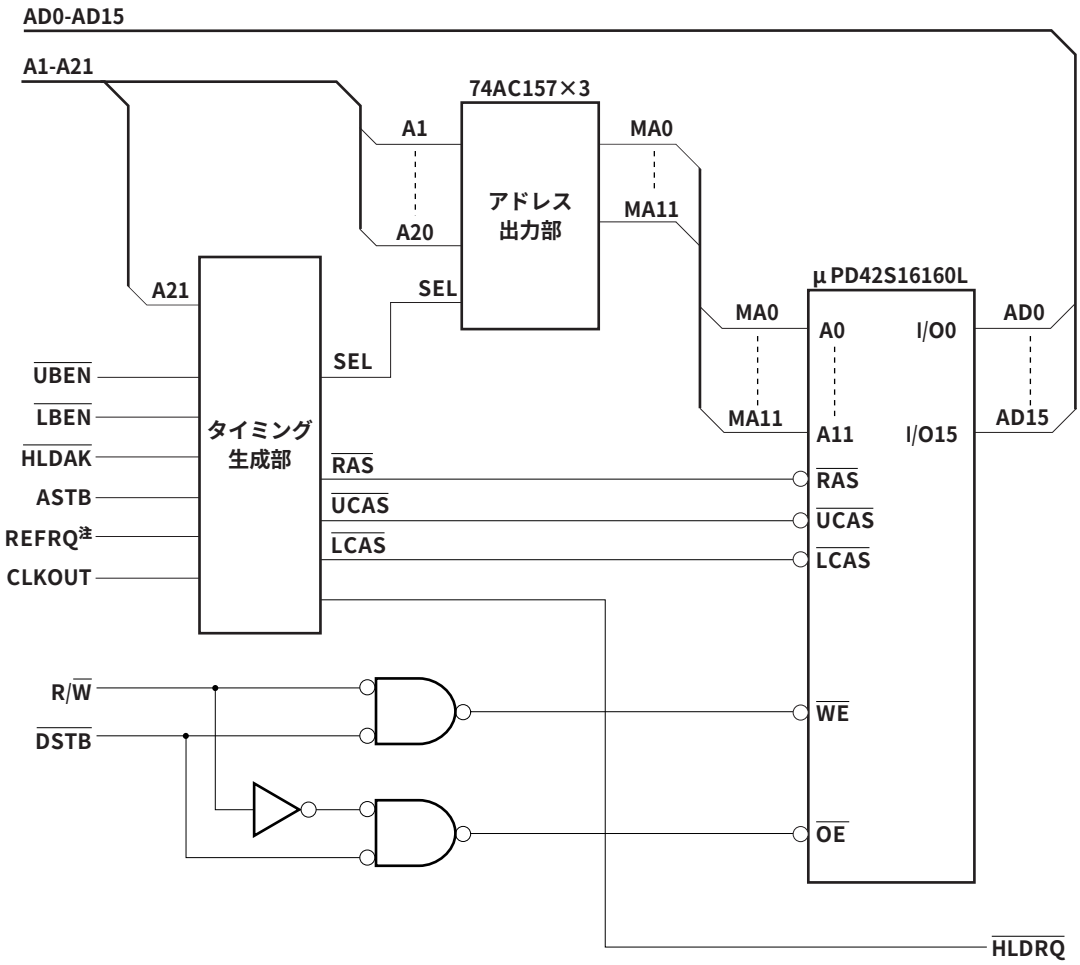
【回路方式】

- ○ T2クロックの立ち上がりエッジでV850/SA1のA21をモニタして，アドレスが一致したらリード・サイクル／ライト・サイクルのタイミングをCLKOUT信号で作成。
- ○ $\overline{\text{WAIT}}$端子の制御は行いません（ハイ・レベルに固定）。
- ○ 外部カウンタで31.25 μsに1回[注]のリフレッシュ要求を作成して，次の手順でリフレッシュを行います。

注　μPD42S16160Lのリフレッシュ・サイクルは4096サイクル／128 μsなので，リフレッシュ・インターバルは，128000÷4096 = 31.25(μs)になります。したがって，31.25 μs以下のリフレッシュ・インターバルが必要になります。
16 MHzの1周期は62.5 nsなので，31.25 μs÷62.5 ns = 500になります。したがって，CLKOUT500回に1回のリフレッシュを行えば良いことになります。

① 外部カウンタでリフレッシュ要求を発生。
② V850/SA1の$\overline{\text{HLDRQ}}$端子をアクティブにします。
③ V850/SA1の$\overline{\text{HLDAK}}$端子がアクティブになり，ホールド状態になります。
④ CBRのリフレッシュ・サイクルを1回行います（タイミングはCLKOUT信号で作成）。
⑤ V850/SA1の$\overline{\text{HLDRQ}}$端子をインアクティブにします。
⑥ V850/SA1の$\overline{\text{HLDAK}}$端子がインアクティブになり，ホールド状態が解除されます。

図2－18　μPD42S16160L接続回路例（$\overline{\text{HLDRQ}}$端子を使用したリフレッシュ）



注　REFRQ信号は，CLKOUT信号に同期した1クロック分のハイ・レベルをリフレッシュ・タイミングごとに1回発生するものとします。

備考　μPD42S16160L：1 M×16ビット（16 Mビット）

図2－19　タイミング生成部（DRAM接続回路例1）

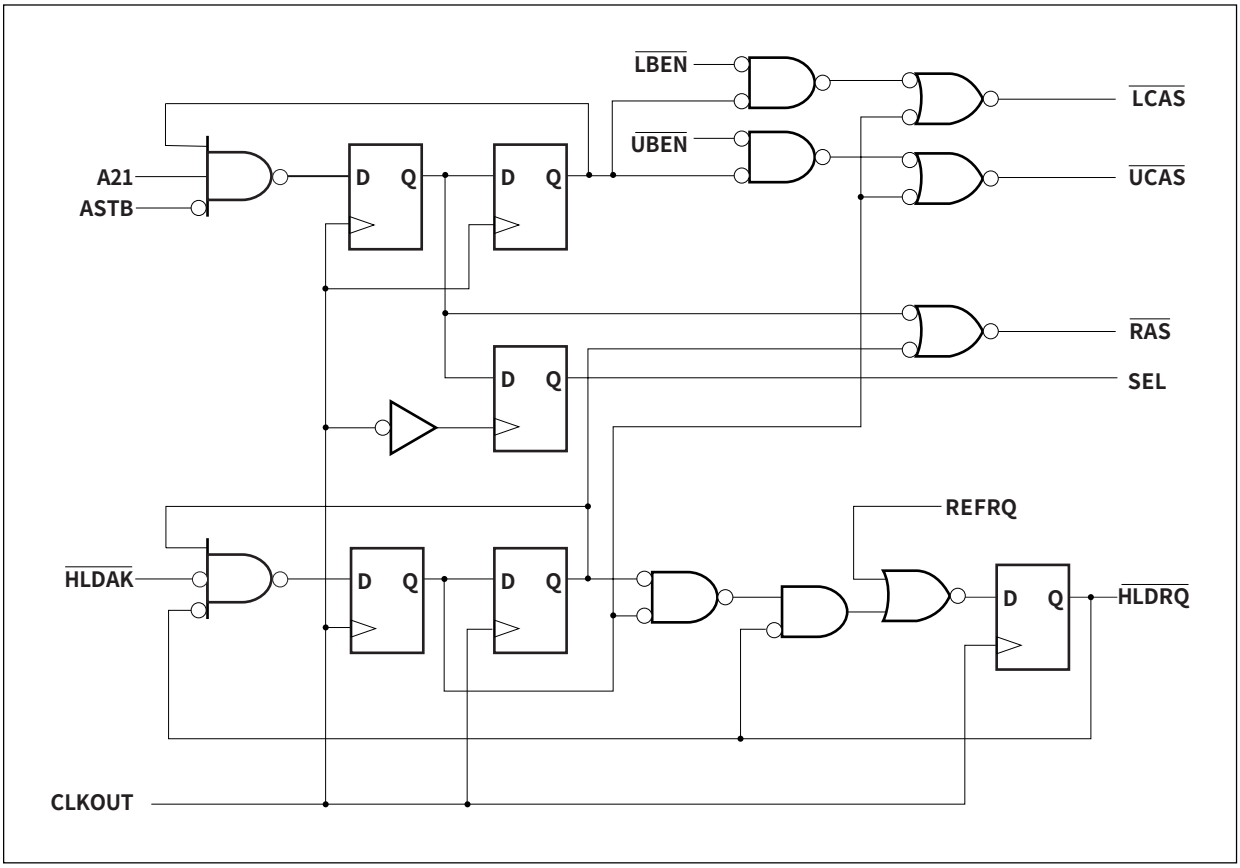


図2－20　アドレス出力部

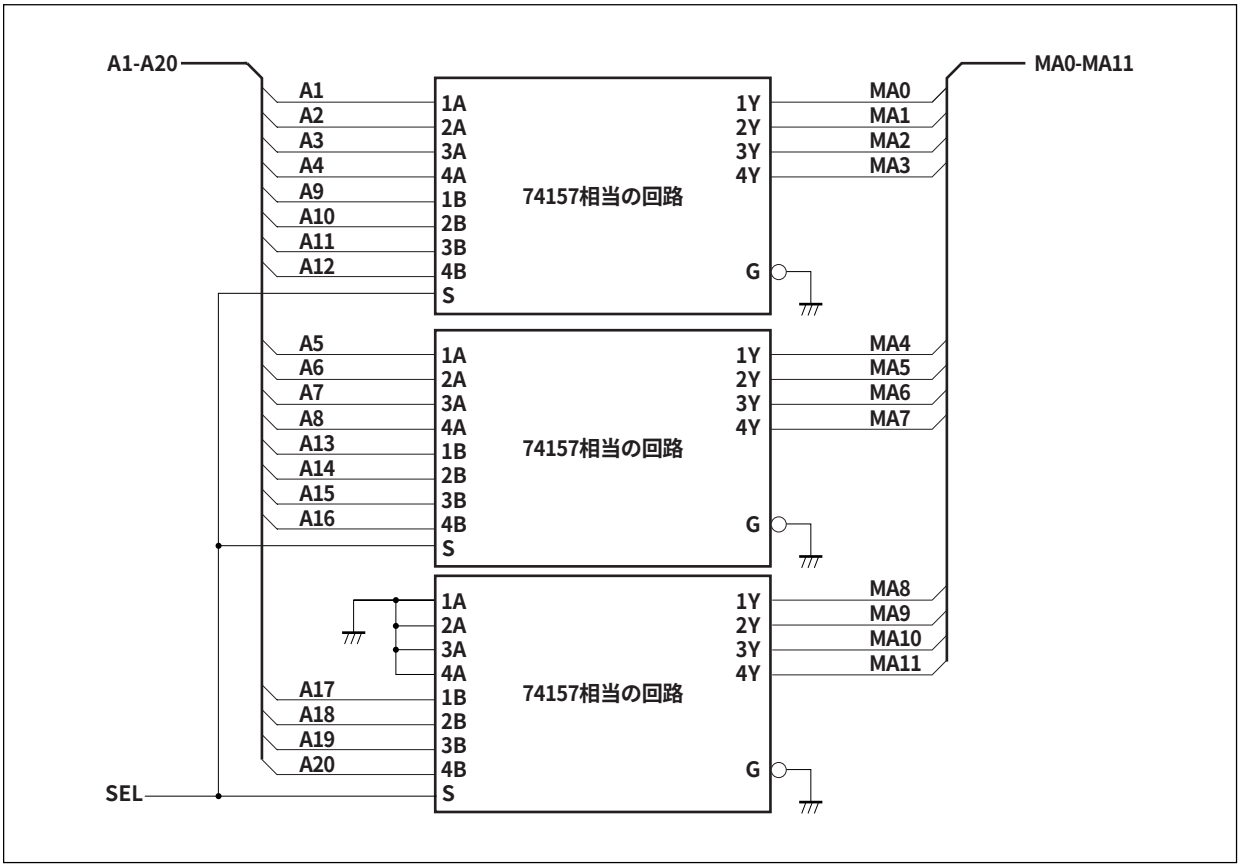
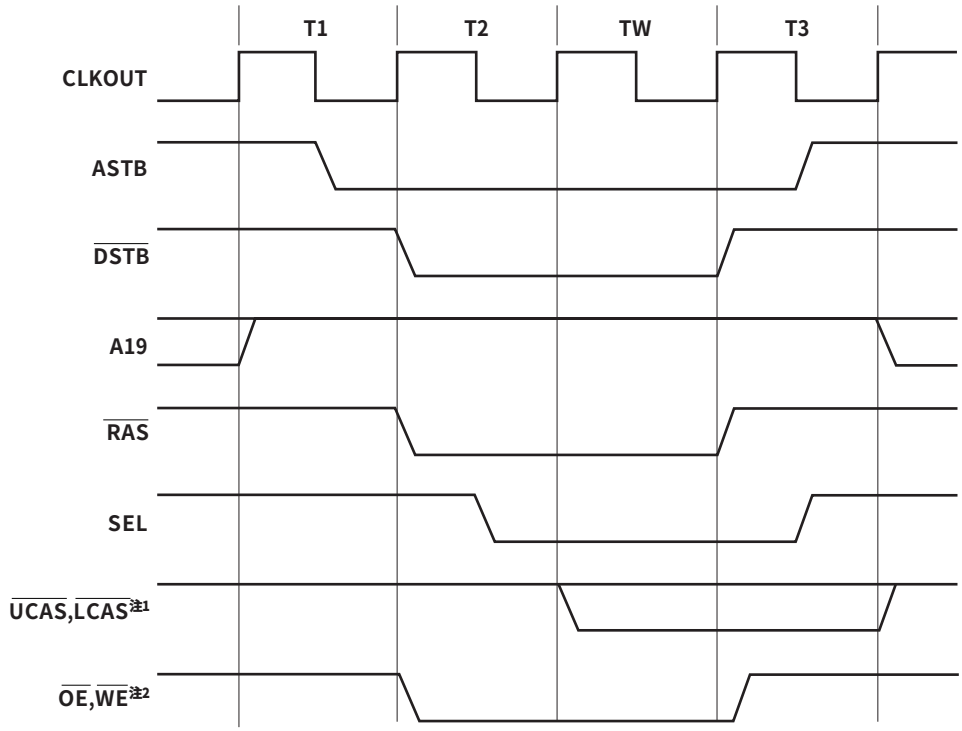


図2－21　リード／ライト・タイミング（μPD42S16160L）



注1．$\overline{\text{LBEN}}$ **がアクティブ時は**$\overline{\text{LCAS}}$ **がアクティブになります。**

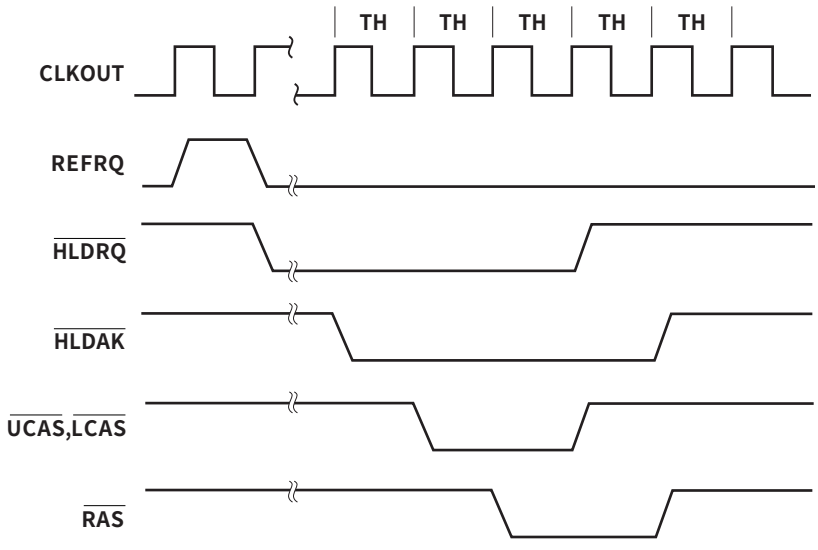
$\overline{\text{UBEN}}$ **がアクティブ時は**$\overline{\text{UCAS}}$ **がアクティブになります。**

$\overline{\text{LBEN}}$，$\overline{\text{UBEN}}$ **がアクティブ時は**$\overline{\text{LCAS}}$，$\overline{\text{UCAS}}$ **がアクティブになります。**

2．リード時は$\overline{\text{OE}}$ **がアクティブになります。**

ライト時は$\overline{\text{WE}}$ **がアクティブになります。**

図2－22　リフレッシュ・タイミング（DRAM接続回路1）



2. 4. 2　DRAM接続回路例2（$\overline{\text{WAIT}}$端子を使用したリフレッシュ・アービタ）

【回路構成】

- ○ V850/SA1の内部システム・クロック：16 MHz
- ○ 接続デバイス　：μPD42S16160L×1
- ○ 占有空間　：外部メモリ空間の200000H-3FFFFFH
- ○ サポートするサイクル　：リード・サイクル／アーリ・ライト・サイクル／CBRリフレッシュ・サイクル

【DWC, BCCの設定】

- ○ ウエイト設定　：0ウエイト（$\overline{\text{WAIT}}$端子で制御）
- ○ アイドル・ステート　：挿入しない
- ○ リード／ライト制御端子のモード：$\overline{\text{DSTB}}$, R/$\overline{\text{W}}$, $\overline{\text{LBEN}}$, $\overline{\text{UBEN}}$信号出力

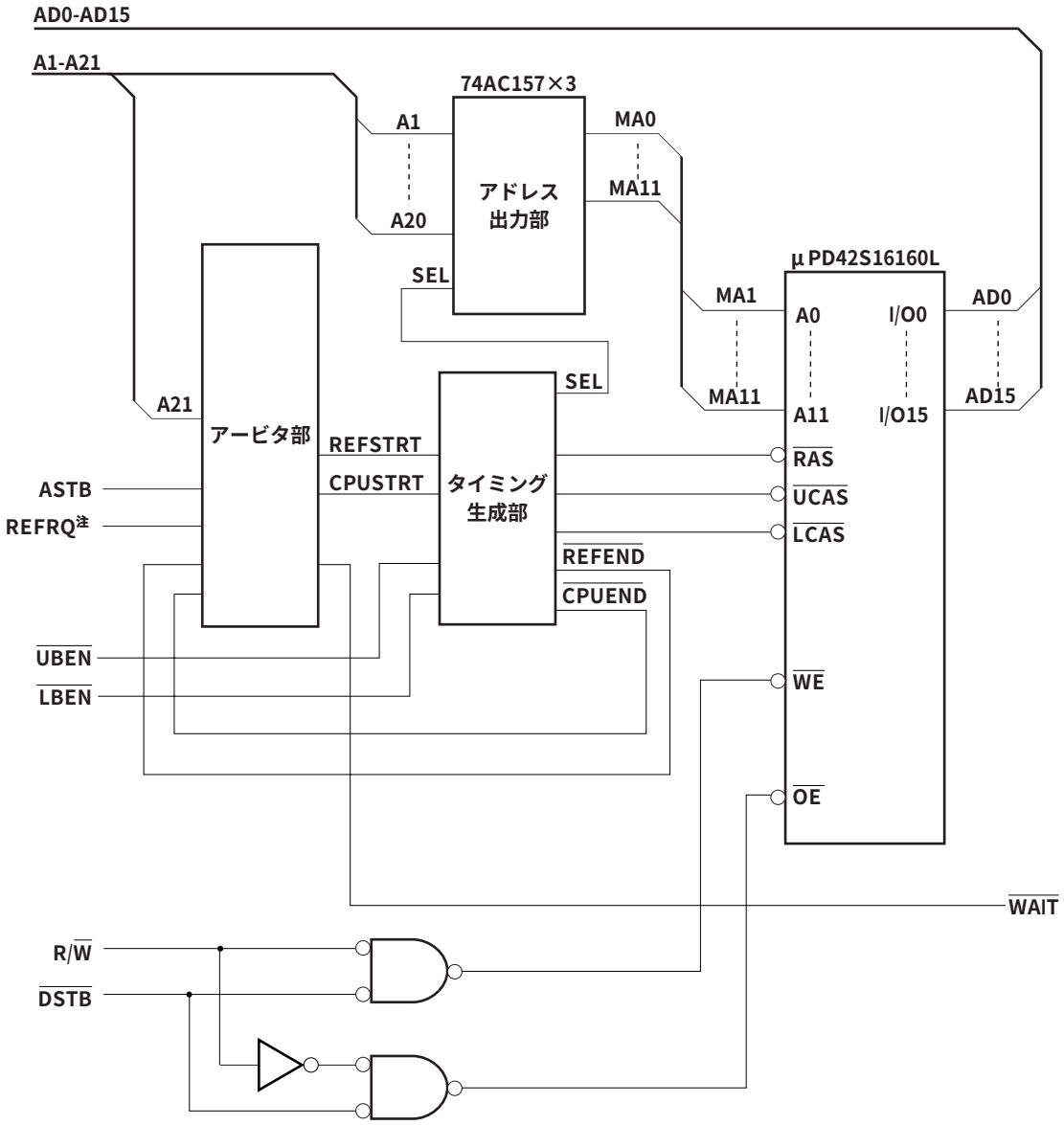
【回路方式】

- ○ T2クロックの立ち上がりエッジでV850/SA1のA21をモニタして，アドレスが一致したらリード・サイクル／ライト・サイクルのタイミングをCLKOUT信号で作成。
- ○ V850/SA1のアクセス要求とリフレッシュ要求の調停を行うアービタ回路を作成。
- ○ V850/SA1のアクセス要求とリフレッシュ要求が同時に発生した場合，リフレッシュ・サイクルを優先。
- ○ リフレッシュ動作中にV850/SA1からアクセス要求があった場合，ハードウエアによるウエイトを挿入して，リフレッシュ動作が終了してからリード･サイクルまたはライト・サイクルを駆動。
- ○ 外部カウンタで31.25 μsに1回[注]のリフレッシュ要求を作成。

注　μPD42S16160Lのリフレッシュ・サイクルは4096サイクル／128 μsなので，リフレッシュ・インターバルは，128000÷4096 = 31.25(μs)になります。したがって，31.25 μs以下のリフレッシュ・インターバルが必要になります。

16 MHzの1周期は62.5 nsなので，31.25 μs÷62.5 ns = 500になります。したがって，CLKOUT500回に1回のリフレッシュを行えば良いことになります。

図2－23　μPD42S16160L接続回路例（$\overline{\text{WAIT}}$端子を使用したリフレッシュ・アービタ）



注　REFRQ信号は，CLKOUT信号に同期した1クロック分のハイ・レベルをリフレッシュ・タイミングごとに1回発生するものとします。

備考　μPD42S16160L：1 M×16ビット（16 Mビット）

図2－24　アービタ部

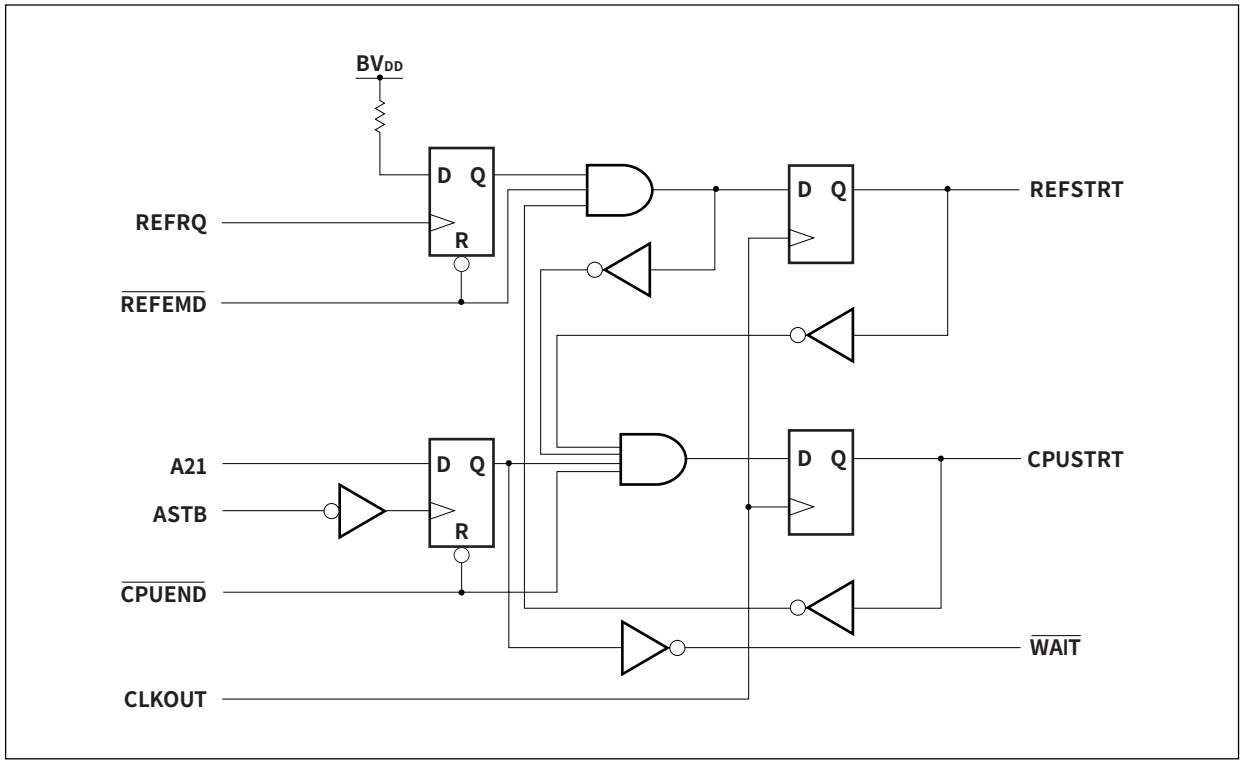


図2－25　タイミング生成部（DRAM接続回路例2）

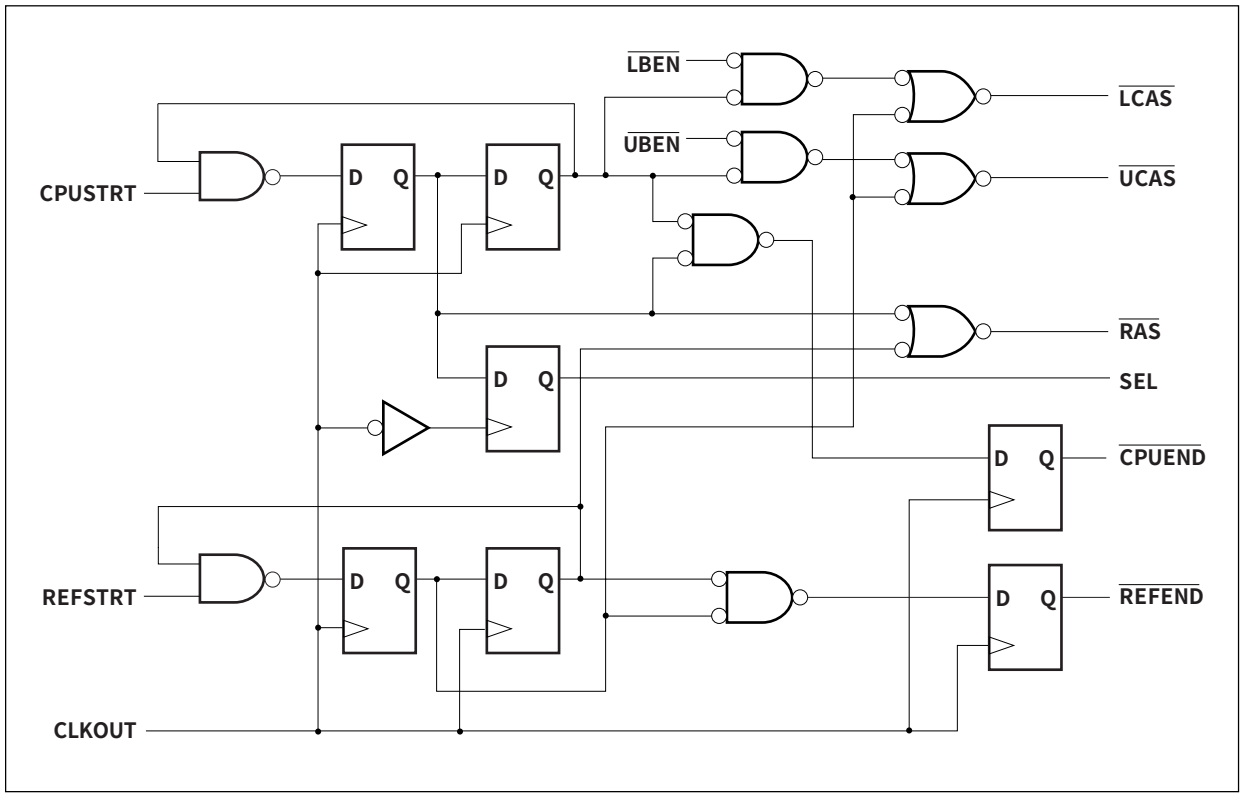
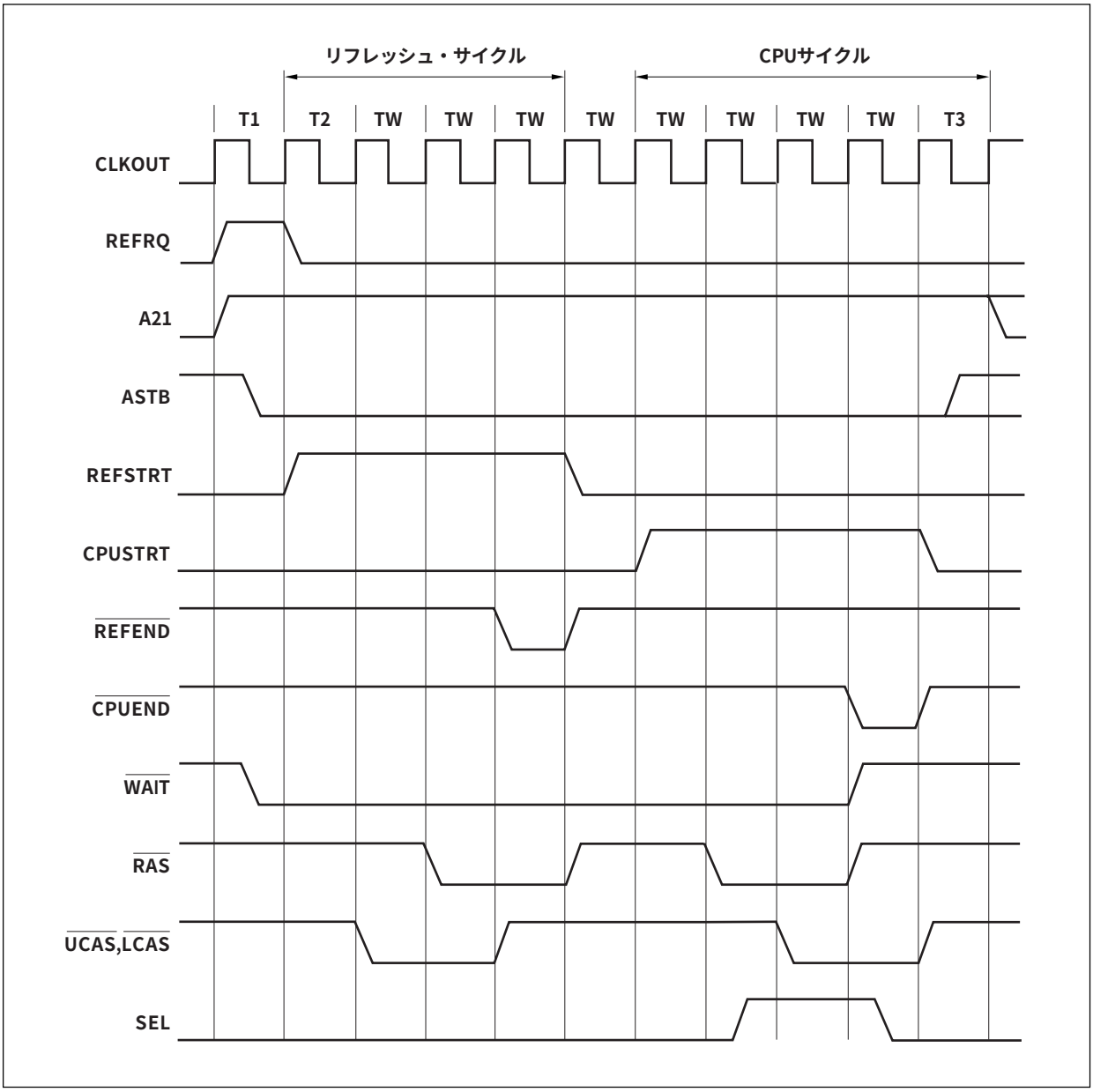


図2－26　アービタ部の動作（リフレッシュ要求とV850/SA1の要求が同時に発生した場合）



2. 5　バス・サイジング回路を使用した8ビットPROMの接続

外部のROM空間をV850/SA1に接続する例を示します。

この回路例では，8ビット・アクセス／16ビット・アクセスの両方に対応しています。16ビット・リード動作では，ハードウエアでPROMに対して2回のリード・アクセスを行い，16ビット・データをそろえてV850/SA1に転送します。ウエイト制御はハードウエアで行います。

【回路構成】

○ V850/SA1の内部システム・クロック：16 MHz

○ 接続デバイス　　：HN27C4001G×1

○ 占有空間　　：外部メモリ空間の000000H-07FFFFH

【DWC, BCCの設定】

○ ウエイト設定　　：0ウエイト（$\overline{\text{WAIT}}$端子で制御）

8ビット・リード時　：1ウエイト（ハードウエアにより挿入）

16ビット・リード時：3ウエイト（ハードウエアにより挿入）

○ アイドル・ステート　　：挿入しない

○ リード／ライト制御端子のモード：$\overline{\text{RD}}$, $\overline{\text{WRL}}$, $\overline{\text{WRH}}$, $\overline{\text{UBEN}}$信号出力

【回路方式】

○ ROMの$\overline{\text{CS}}$信号（$\overline{\text{ROMCS}}$）は，アドレス上位3ビットをデコードして生成。

○ A0（AD0を分離して作成した信号），$\overline{\text{UBEN}}$をデコードして，V850/SA1のアクセス・タイプを判別。

A0	$\overline{\text{UBEN}}$	アクセス・タイプ
ロウ・レベル	ロウ・レベル	16ビット・アクセス
ロウ・レベル	ハイ・レベル	偶数アドレス・バイト・アクセス
ハイ・レベル	ロウ・レベル	奇数アドレス・バイト・アクセス

○ 8ビットのバッファを1つ使用して，16ビット・リード時に下位バイトをラッチ。

○ ROMのA0信号およびラッチ信号とV850/SA1の$\overline{\text{WAIT}}$信号は，CLKOUT信号で作成。

○ ROMの入出力信号は，レベル変換デバイスを使用して接続。

ROMのA0-A18, $\overline{\text{CS}}$, $\overline{\text{OE}}$端子：74FCT244×3

ROMのO0-O7　　：74FCT3373（V850/SA1のAD0-AD7に接続）×1

：74FCT3244（V850/SA1のAD8-AD15に接続）×1

図2－27　バス・サイジングを使用した8ビットPROMの接続回路例

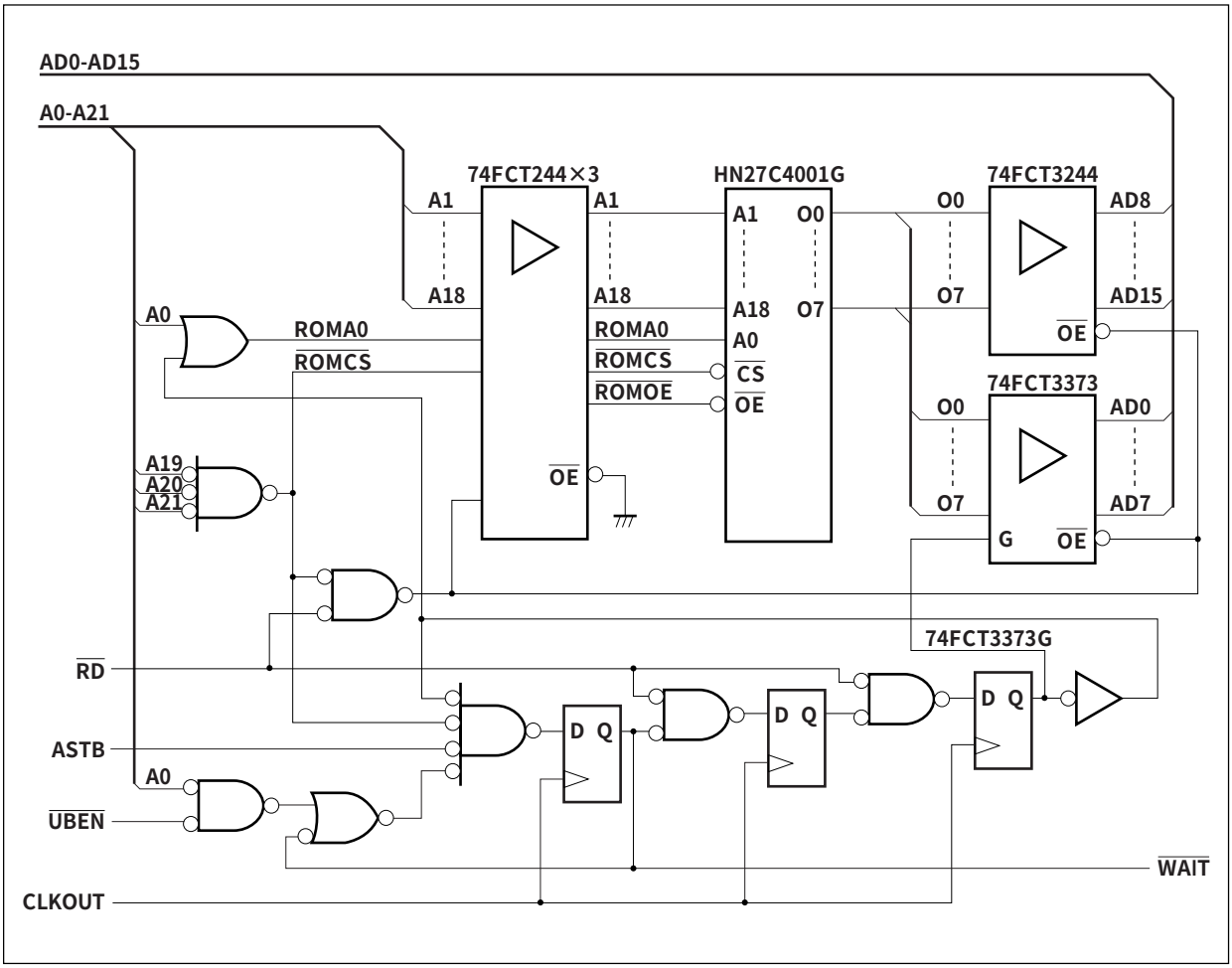


図2－28　バイト・アクセス

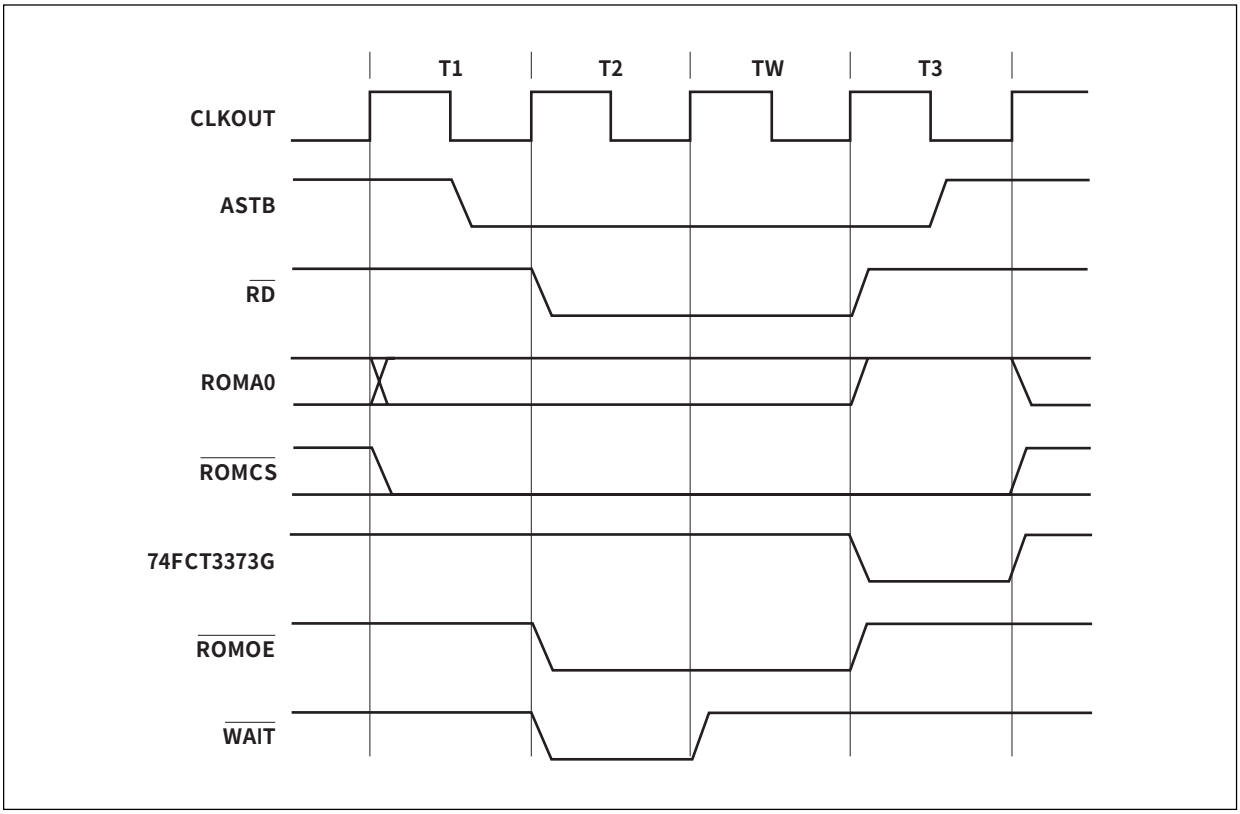
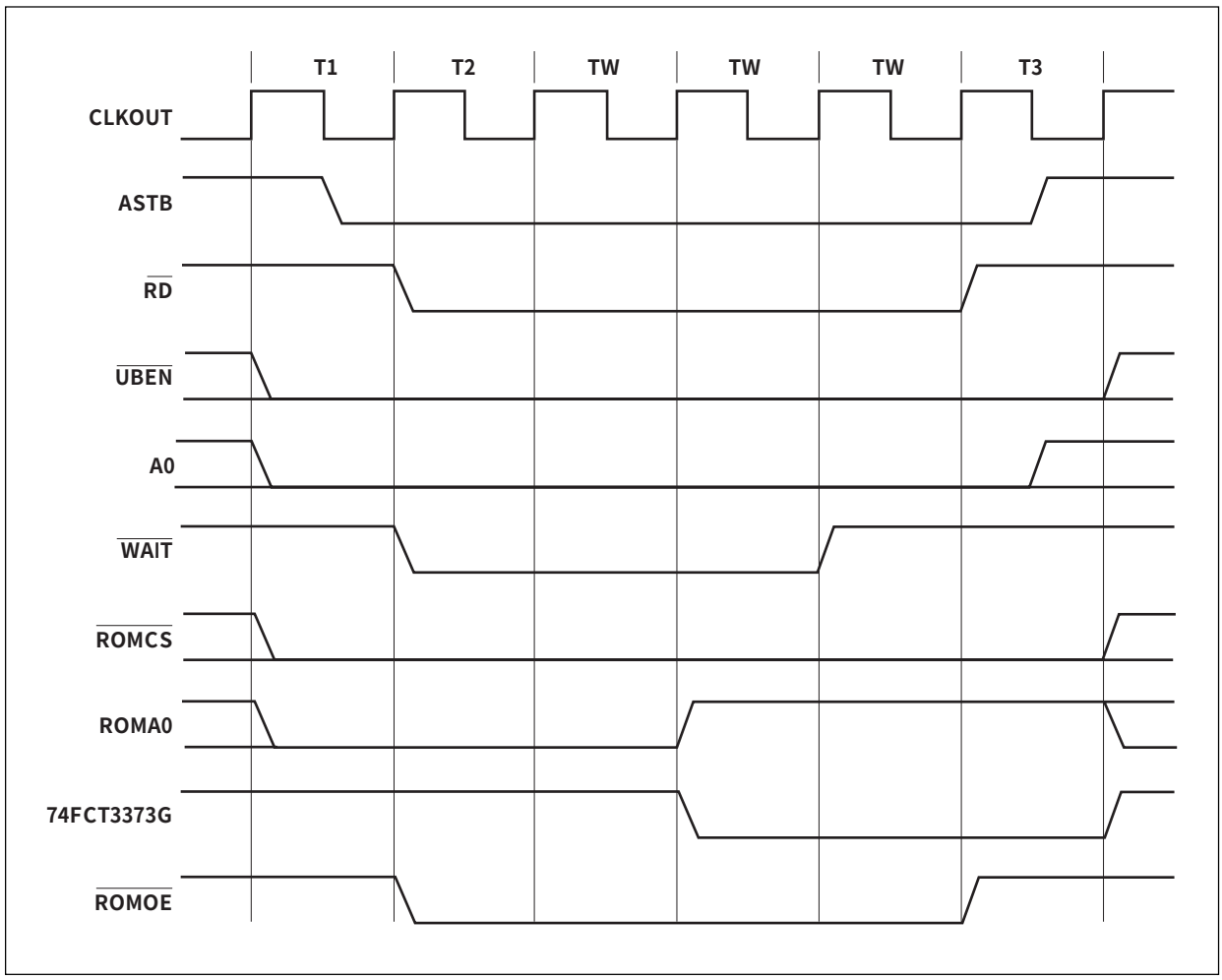


図2－29　ワード／ハーフワード・アクセス



第3章　周辺機能の接続例

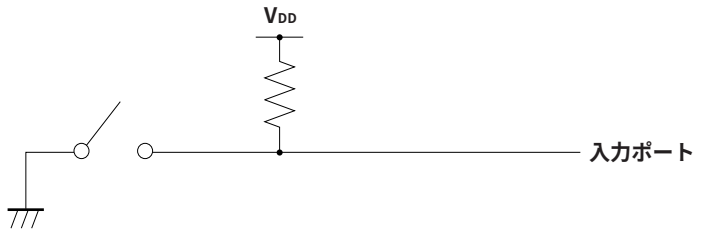
この章では，V850/SA1の内部周辺I/Oの接続例を示します。V850/SA1のV_{DD}，BV_{DD}は，3.3 Vを供給するものとします。

この章の回路例では，第2章と同様に5 V系デバイスと接続する場合，レベル変換デバイスを使用しています。実際の回路では，V850/SA1と接続するデバイスの双方の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。

3. 1　押しボタン・スイッチの接続（ポート機能）

入力ポートに押しボタン・スイッチを接続する例を次に示します。

図3－1　ソフトウエアでチャタリング除去を行う例

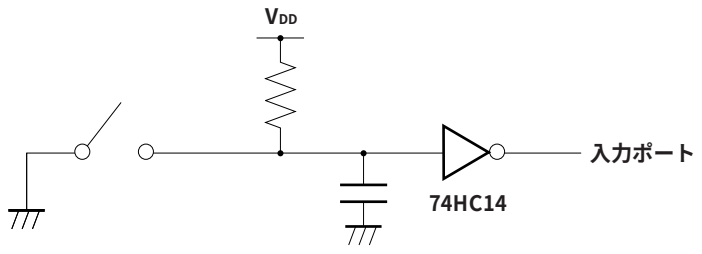


備考1．スイッチがオンで“0”，オフで“1”が入力されます。

2．スイッチのチャタリングは，ソフトウエアで考慮してください。

3．内部プルアップが可能なポートでは，外付けプルアップ抵抗を削除できます。

図3－2　ハードウエアでチャタリング除去を行う例



備考　スイッチがオンで“1”，オフで“0”が入力されます。

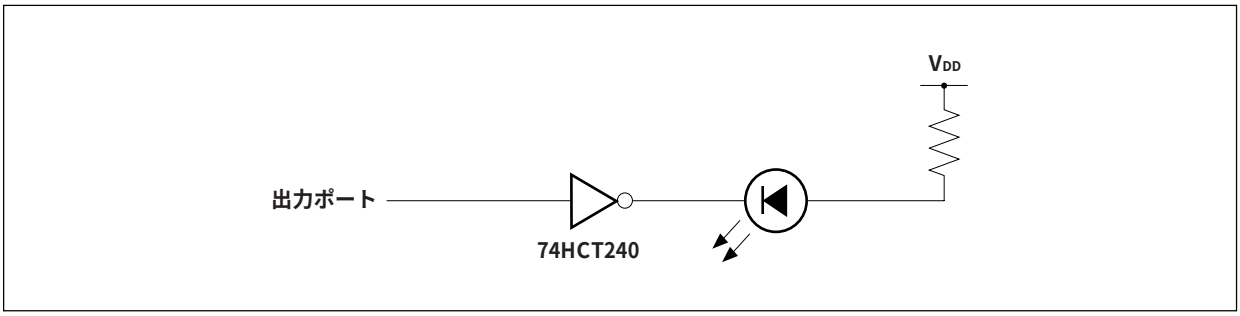
3. 2　LEDの接続（ポート機能）

出力ポートにLEDを接続する例を次に示します。

・ポート出力が“1”のとき，LEDが点灯します。

・ポート出力が“0”のとき，LEDが消灯します。

図3－3　出力ポートにLEDを接続する例



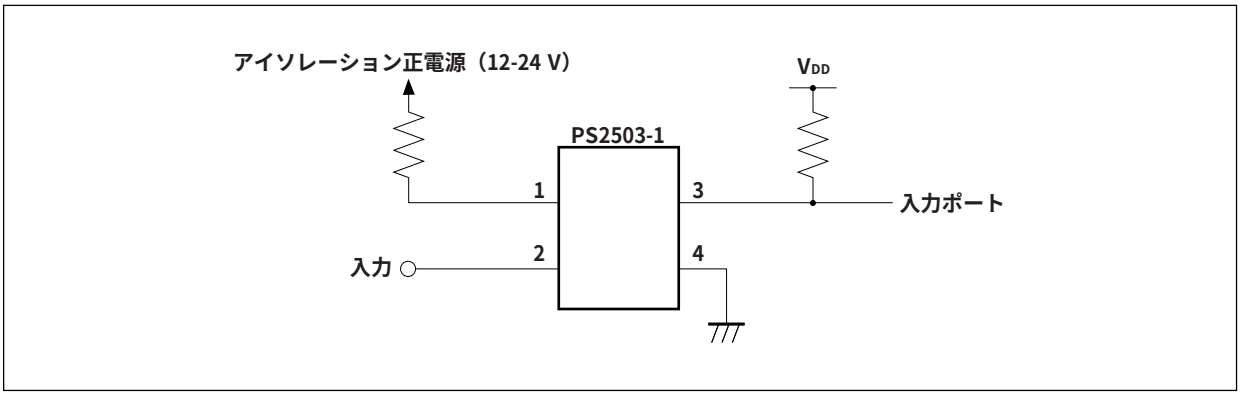
3. 3　アイソレーション・ディジタル入力（ポート機能）

フォトカプラを使用して，入力ポートをアイソレーションする例を次に示します。

・ポート入力がロウ・レベルのとき，“0”が入力されます。

・ポート入力がハイ・レベルのとき，“1”が入力されます。

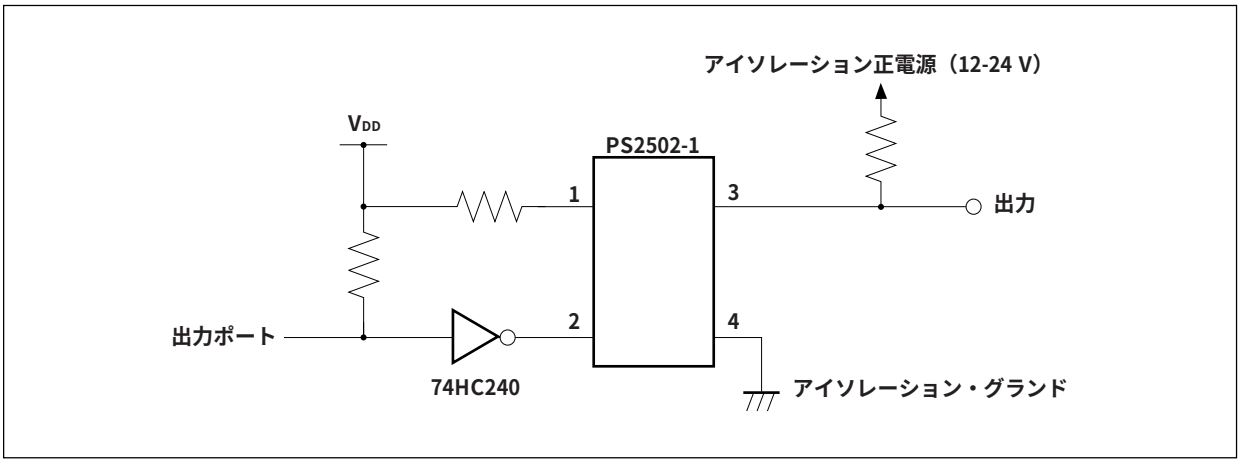
図3－4　入力ポートをアイソレーションする例



3. 4　アイソレーション・ディジタル出力（ポート機能）

フォトカプラを使用して，出力ポートをアイソレーションする例を次に示します。

図3－5　出力ポートをアイソレーションする例

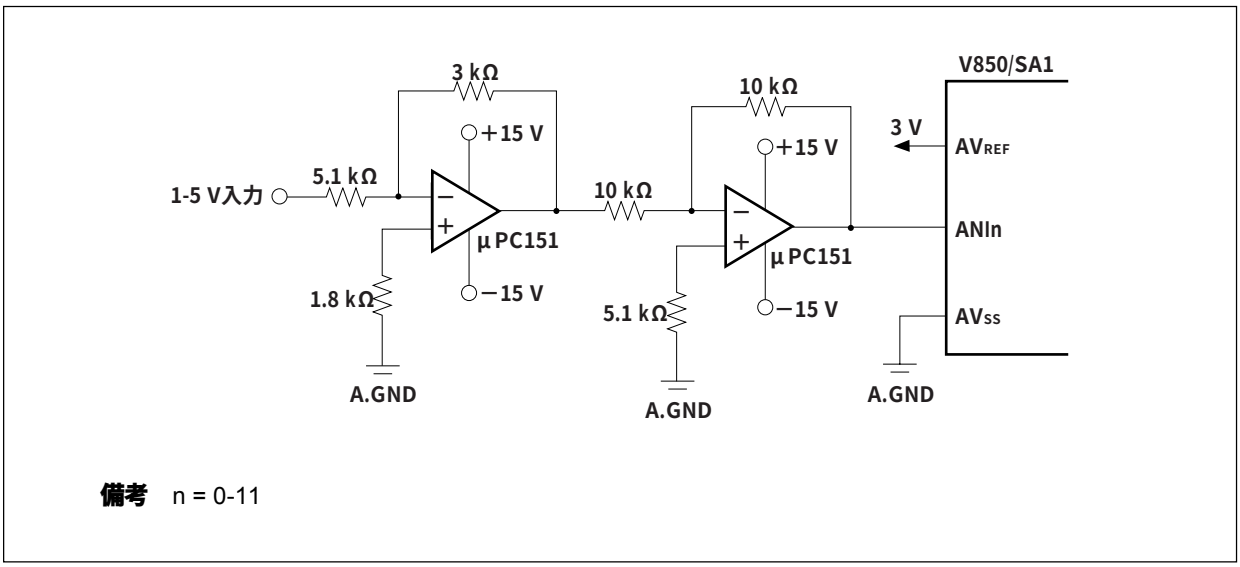


3. 5　1-5 Vタイプ・アナログ入力（A/Dコンバータ）

アナログ電圧1-5 V入力の接続例を次に示します。

- 入力電圧が1 Vのとき，0.59 Vが入力されます。
- 入力電圧が5 Vのとき，2.94 Vが入力されます。

図3－6　アナログ電圧1-5 V入力の接続例



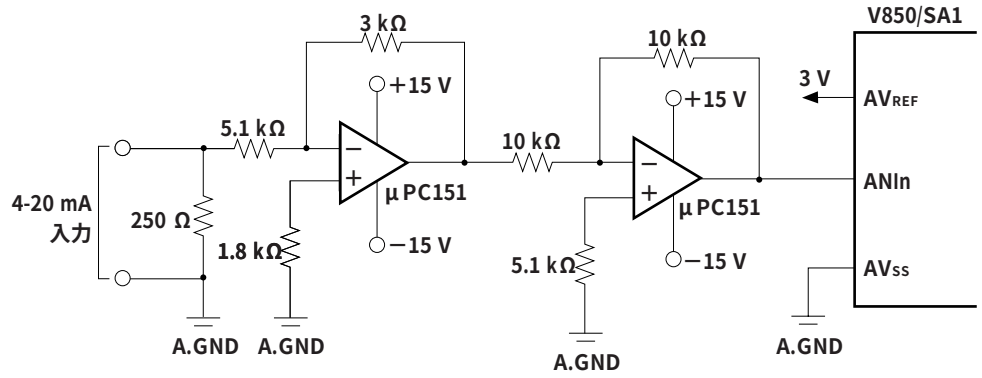
備考　n = 0-11

3. 6　4-20 mAタイプ・アナログ入力（A/Dコンバータ）

4-20 mA方式のアナログ入力の接続例を次に示します。

- 入力値が4 mAのとき，0.59 Vが入力されます。
- 入力値が20 mAのとき，2.94 Vが入力されます。

図3ー7　4-20 mA方式のアナログ入力の接続例



備考　n = 0-11

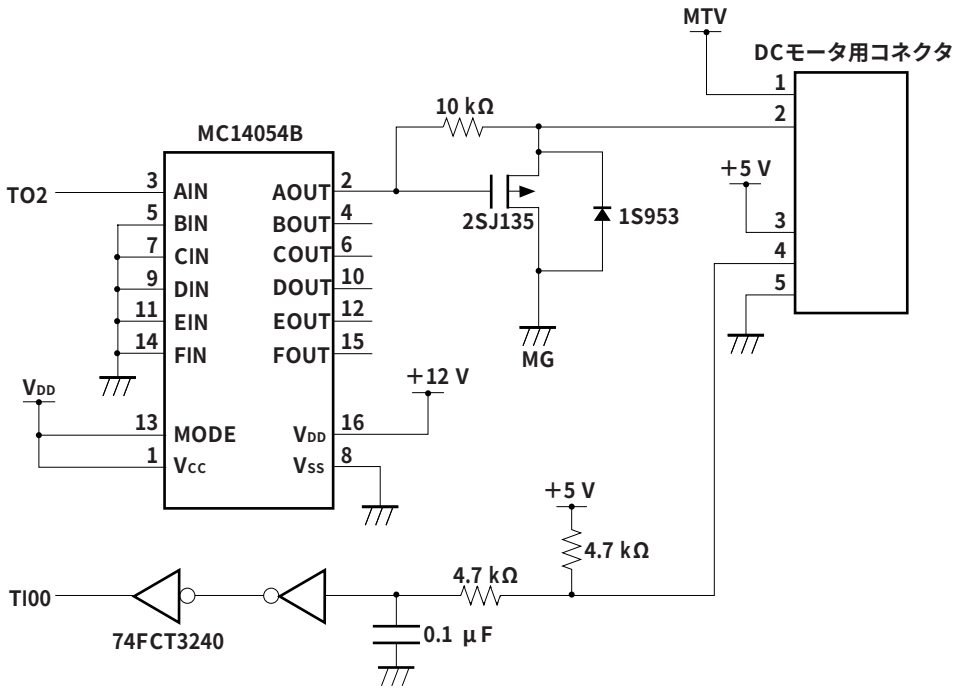
3. 7　DCモータの接続（タイマ／カウンタ機能）

TM2の出力（TO2）をDCモータの制御に使用する例を次に示します。

TM2をPWM出力に設定して，TO2の出力パルスを変化させてDCモータの回転速度を制御します。

DCモータからの回転エンコーダ・パルス出力を，TI00に接続することによってDCモータの回転速度をモニタできます。

図3－8　TM2の出力（TO2）をDCモータの制御に使用する例



備考　MTVとMGは，ほかの電源と分離されています。

DCモータは，DME33BM37Gを接続します。

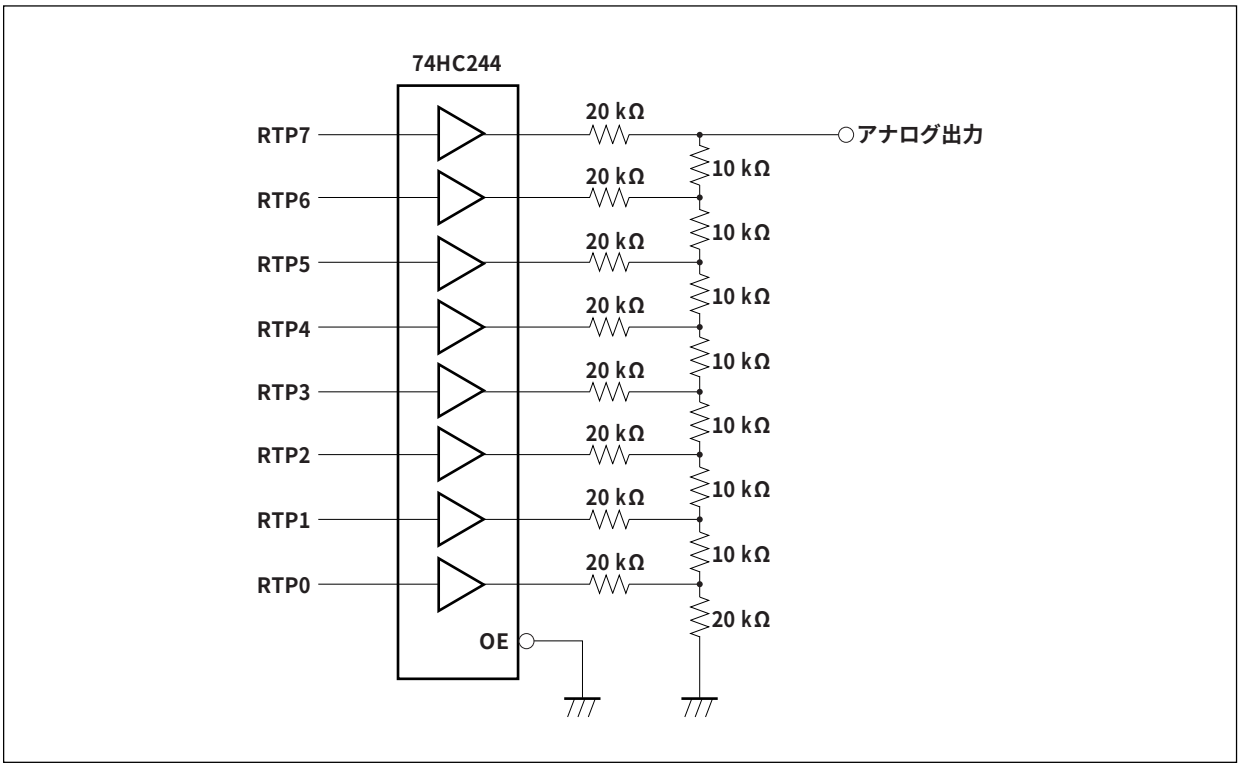
表3－1　DCモータ用コネクタ

番　号	機　　能
1	モータ用電源（＋12 V）
2	モータ用グランド
3	パルス・ジェネレータ用電源
4	パルス・ジェネレータ出力
5	パルス・ジェネレータ用グランド

3. 8　R-2R回路によるアナログ出力（RTP）

RTP0-RTP7の出力を使用して，R-2R回路による8ビットD/Aコンバータを構成する例を次に示します。
RTP7がMSB, RTP0をLSBに対応して，FFHでフルスケール出力となります。

図3－9　R-2R回路による8ビットD/Aコンバータを構成する例



3. 9　RS-232-Cインタフェースの接続（UART）

UART1を調歩同期式RS-232-Cインタフェースとして接続する例を次に示します。

送受信クロックは，内部クロックを分周して使用します。

図3－10　UART1を調歩同期式RS-232-Cインタフェースとして接続する例

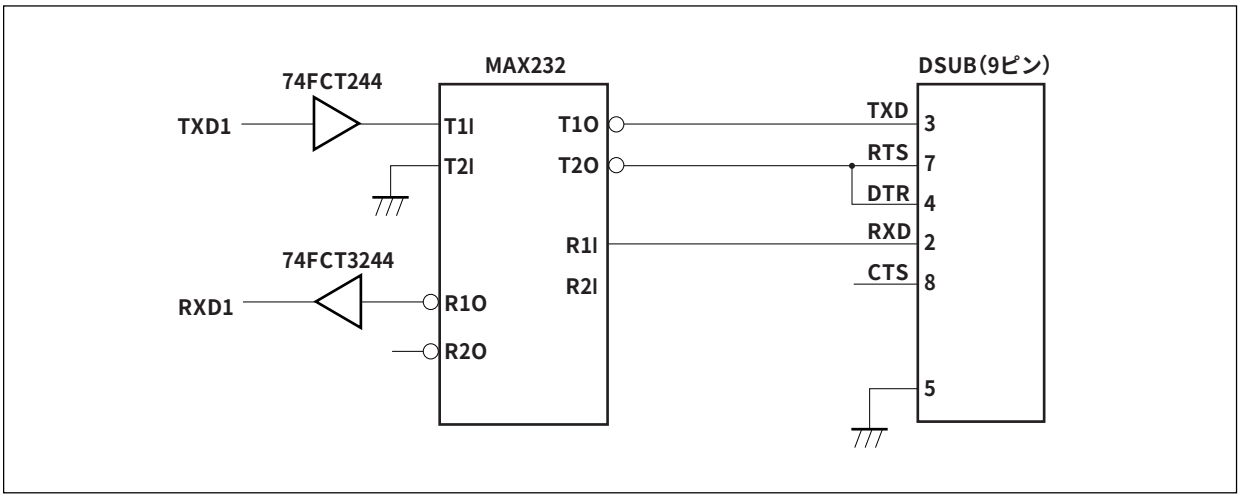


表3－2　RS-232-Cインタフェース・コネクタ

番　号	信号名	意　　味
1	NC	未接続
2	RXD	受信データ
3	TXD	送信データ
4	DTR	ターミナル・レディ
5	SG	信号グランド
6	NC	未接続
7	RTS	送信要求
8	CTS	送信可
9	NC	未接続

備考　RTS, DTRはアクティブ・レベルに固定されています。

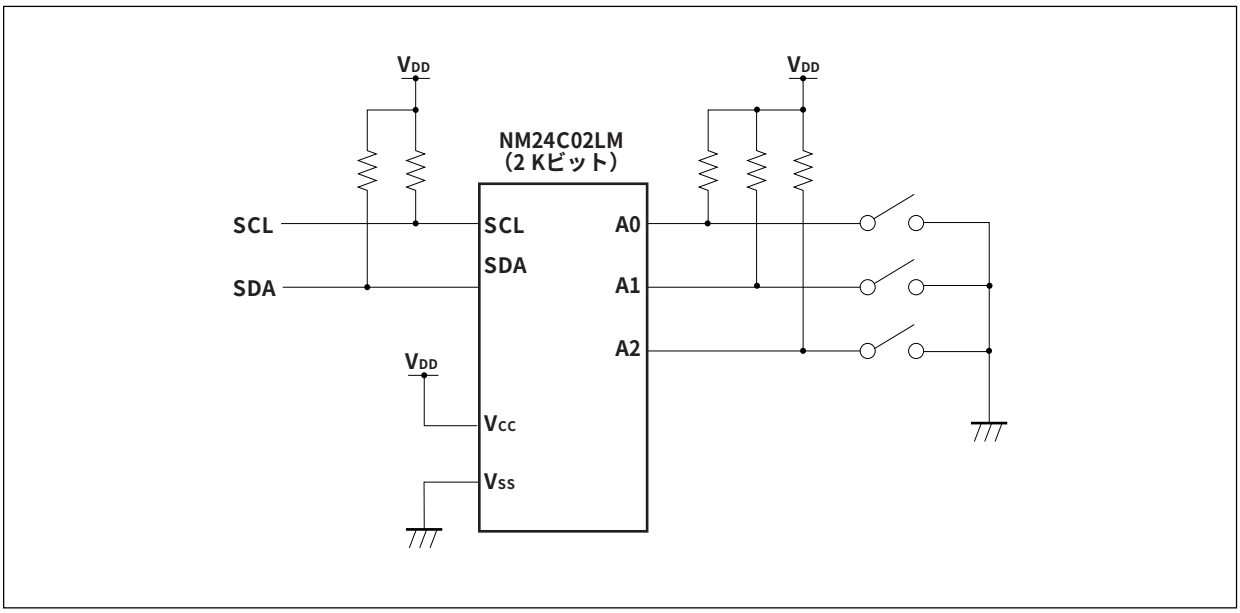
CTSは接続されていません。

3. 10　シリアルEEPROM^{TM}の接続（I^{2}C）

I^{2}Cバス・インタフェースをシリアル・タイプのEEPROM（NM24C02LM）に接続する例を次に示します。

NM24C02LMのデバイス・アドレス（A0-A2）は，スイッチで設定します。SCL, SDAは，V850/SA1のSCL, SDAと直結します。

図3－11　I^{2}Cバス・インタフェースをシリアル・タイプのEEPROM（NM24C02LM）に接続する例



3. 11　停電検出回路とバックアップ電源

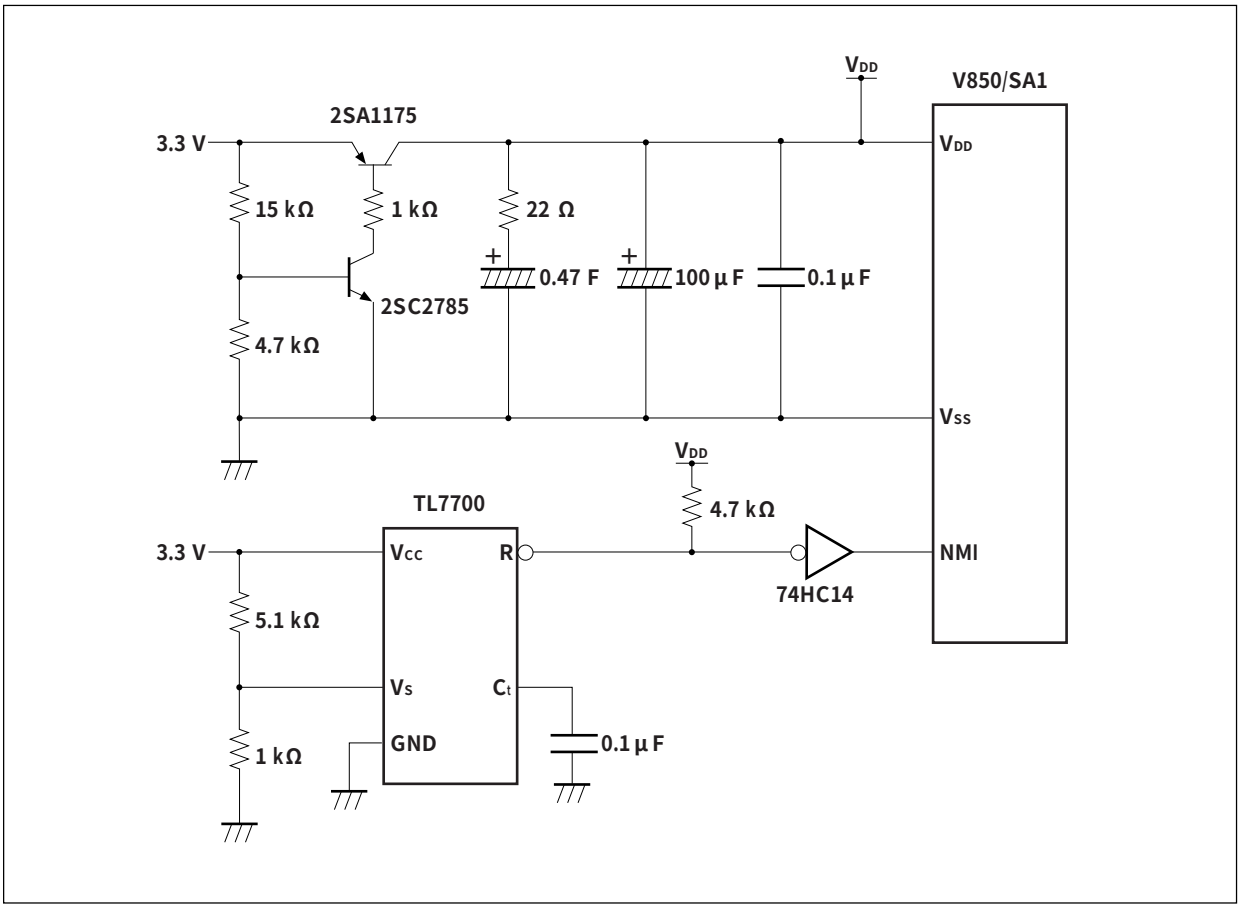
電源電圧の監視回路と停電時のバックアップ例を示します。

この回路例では，電源電圧が3 Vに降下したときはNMI端子がハイ・レベルになり，3 V以上に復帰したときはNMI端子がロウ・レベルになります。

ノンマスカブル割り込みは，立ち上がりでも，立ち下がりでも発生するように設計します。ノンマスカブル割り込みが発生したとき，NMI端子の状態を確認するようにプログラムすれば，より安全なシステムになります。

停電時は，コンデンサ（0.47 F）からV_{DD}が供給されます。

図3－12　電源電圧の監視回路と停電時のバックアップ例



第4章　アプリケーション例

TB-V850/SA1は，V850/SA1の評価，学習を目的として開発されたトレーニング・ボードです。TB-V850/SA1の機能を次に示します。

4. 1　TB-V850/SA1の機能

4. 1. 1　TB-V850/SA1の概要

（1）CPU供給クロック

メイン・クロック：16 MHz

サブクロック　　：32.768 kHz

（2）CPU供給電源

システム電源（＋5 V）とCPU供給電源（＋3.3 V）（V_{DD}, BV_{DD}）を別々に供給

（3）バス・インタフェースの接続

メモリ：EPROM（128 Kバイト）

SRAM（128 Kバイト）

DRAM（2 Mバイト）

I/O　　：LCD（16文字×2行）

（4）周辺機能の接続

INTC　　：NMIスイッチ

INTスイッチ

DCモータからのエンコード・パルス

タイマ・カウンタ：DCモータ×1

UART　　：RS-232-Cインタフェース

専用フラッシュ・ライタ・インタフェース

I^2Cバス　　：EEPROM（NM24C02LM）

A/Dコンバータ　：可変抵抗による0-3 V変化の電圧入力

RTP　　：R-2R回路を経由してのスピーカ出力

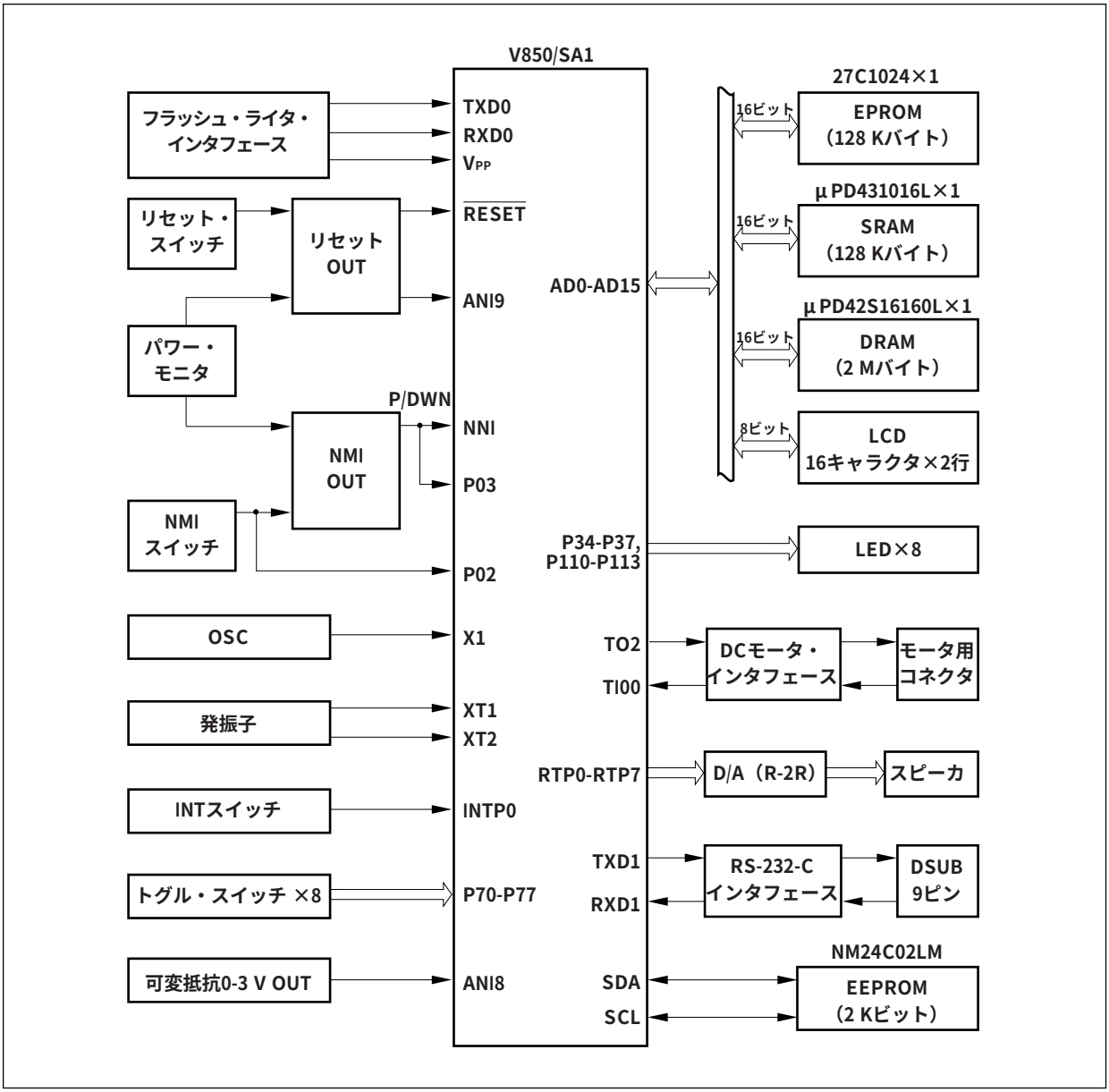
ポート機能　　：出力ポート（LED×8個）

入力ポート（トグル・スイッチ×8個）

4. 1. 2　TB-V850/SA1の構成

TB-V850/SA1の構成を次に示します。

図4－1　TB-V850/SA1の構成



4.1.3　バス・インタフェースの接続

V850/SA1の外部バス4 Mバイト空間に，PROM, SRAM, DRAM, LCDが接続されます。

ボード上のアドレスはAD0-AD15を分離して作成します。制御信号は$\overline{\text{DSTB}}$, R/$\overline{\text{W}}$, $\overline{\text{LBEN}}$, $\overline{\text{UBEN}}$を使用します。

PROM, SRAMのウエイト設定は，プログラマブル・ウエイトを使用します。

DRAM, LCDは，$\overline{\text{WAIT}}$端子の制御によるハードウエア・ウエイトが挿入されます。アイドル・ステートの挿入は必要ありません。

EPROM：xx000000H番地からの128 Kバイト分がROM空間になります。

27C1024（1 Mビット（64 K×16ビット））を1個実装しています。システム・クロックが16 MHz時には，1ウエイト分のプログラマブル・ウエイトが必要になります。

SRAM　：xx020000H番地からの128 Kバイト分がSRAM空間になります。

μPD431016L（1 Mビット（64 K×16ビット））を1個実装しています。この空間は，0ウエイトでアクセスできます。

DRAM　：xx200000H番地からの2 Mバイト分がDRAM空間になります。

μPD42S16160L（16 Mビット（1 M×16ビット））を1個実装しています。この空間へのアクセスでは，ハードウエアにより2ウエイトが挿入されます。リフレッシュ・サイクルは，このユニット上のハードウエアがCBRリフレッシュ・サイクルを駆動します。リフレッシュ動作中のV850/SA1のDRAMへのアクセスは，リフレッシュ・サイクルが終了するまで，このボード上のアービタ回路により，ウエイト・サイクルが挿入されます。リフレッシュ要求とV850/SA1のアクセスが同時に発生した場合は，リフレッシュ・サイクルが優先されます。

LCD　：L1682（16文字×2行のLCD）を1個実装しています。

コマンド・ステータスのアクセスは，xx140000H番地に対するバイト・アクセスとなり，データの書き込みはxx140002H番地に対するバイト・アクセスになります。LCDへのアクセスでは，ハードウエアにより7ウエイトが挿入されます。プログラマブル・ウエイトは必要ありません。

4.1.4　周辺機能の接続

V850/SA1の内蔵周辺機能で，各種スイッチ，LED, DCモータのユニットを制御します。

表4－1　周辺機能の接続

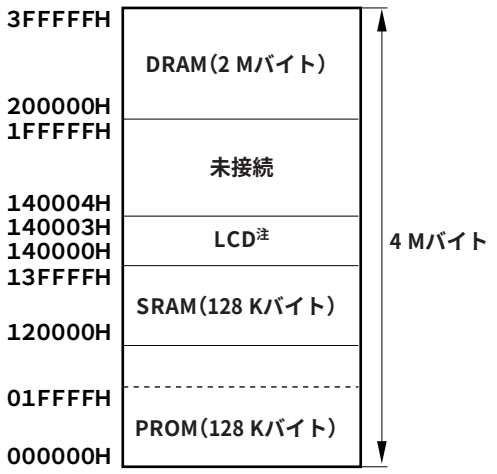
CPUユニット名	接続端子名称	接続ユニット
ポート機能	P70-P77	汎用トグル・スイッチ×8個
	P34-P37, P110-P113	汎用LED×8個
ICU	NMI	NMIスイッチ
	INTP0	INTスイッチ
UART	TXD0, RXD0	フラッシュ・ライタ・インタフェース
	TXD1, RXD1	RS-232-Cインタフェース
I^2C	SDA, SCL	NM24C02LM[注]（EEPROM, 2 Kビット）
タイマ／カウンタ	TO2	DCモータ速度制御
	TI00	DCモータ回転エンコード・パルス
RTP	RTP0-RTP7	スピーカ
A/D	ANI8	可変抵抗（0-3 V入力）
	ANI9	システム電源（＋5 V）状態のモニタ

注　NM24C02LMのスレーブ・アドレスは，ハードウエアで0に固定されています。

4.1.5　外部メモリの配列

V850/SA1のバス・インタフェース（4 Mバイト空間）に接続されるメモリの配列を示します。V850/SA1の各メモリへのアクセスについては，4.3.1　メモリ・マップを参照してください。

図4－2　バス・インタフェース（4 Mバイト空間）



注　LCDは，偶数アドレスに対するバイト・アクセスが有効です。

4. 2　TB-V850/SA1の仕様一覧

表4－2　TB-V850/SA1の仕様一覧

項　　目	仕様概要
V850/SA1	メイン・クロック：16 MHz 水晶振動子をX1端子に接続（X2端子はオープン） サブクロック：32.768 kHz 水晶振動子をXT1端子, XT2端子に接続
EPROM	容　　量　：128 Kバイト 使用ROM　：27C1024相当品×1個 ウエイト数：1ウエイト動作（16 MHz時）
SRAM	容　　量　：256 Kバイト 使用SRAM　：μPD431016L×1個 ウエイト数：0ウエイト動作（16 MHz時）
DRAM	容　　量　：512 Kバイト 使用DRAM　：μPD42S16160L×1個 ウエイト数：2ウエイト動作（ハードウエアで制御） リフレッシュ・サイクル競合時は，ハードウエアにより最大6ウエイトが挿入されます。
LCD	文 字 数　：16文字×2行 型　　名　：L1682 ウエイト数：7ウエイト（ハードウエアで制御）
シリアルEEPROM	NM24C02LM×1個 V850/SA1内蔵I^2Cを使用
スイッチ	トグル・スイッチ×8個 押しボタン・スイッチ×3個 NMIスイッチ INTスイッチ リセット・スイッチ
LED	汎用ランプ（赤色）×8個 電源用ランプ（緑色）×1個
DCモータ	型　　名：DME33S37G18 パルス・ジェネレータ出力（12パルス／回転）は，TI00端子に接続
アナログ出力	ボリュームによる電圧レベルをANI8端子に入力 電圧レベル：0-3 V可変
スピーカ	RTP出力をR-2R回路でD/A変換して接続
RS-232-Cインタフェース	V850/SA1内蔵UART1を使用
フラッシュ・ライタ・インタフェース	V850/SA1内蔵UART0を使用

4. 3　内部レジスタの設定

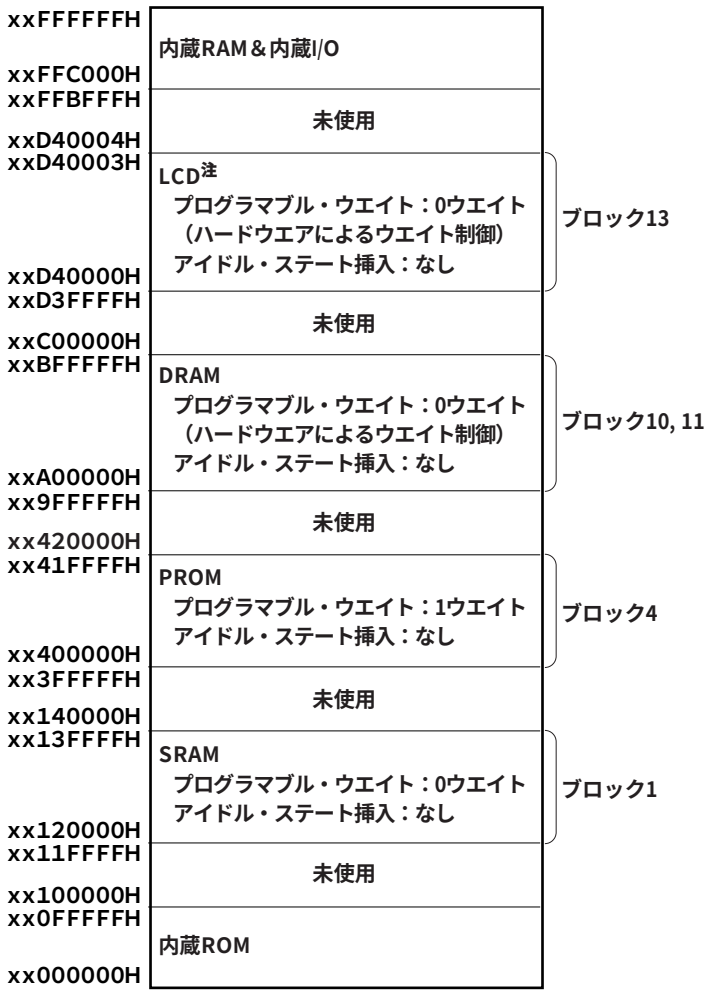
TB-V850/SA1のハードウエア構成により，設定が決定される内部レジスタについて説明します。その他のレジスタについては，アプリケーション・プログラムの内容で設定が決定されます。

詳細については，4. 4　**アプリケーション・プログラム例**を参照してください。

4. 3. 1　メモリ・マップ

V850/SA1のウエイト挿入機能を有効に活用するために，TB-V850/SA1では図4－3で示すメモリ・マップを使用しています。

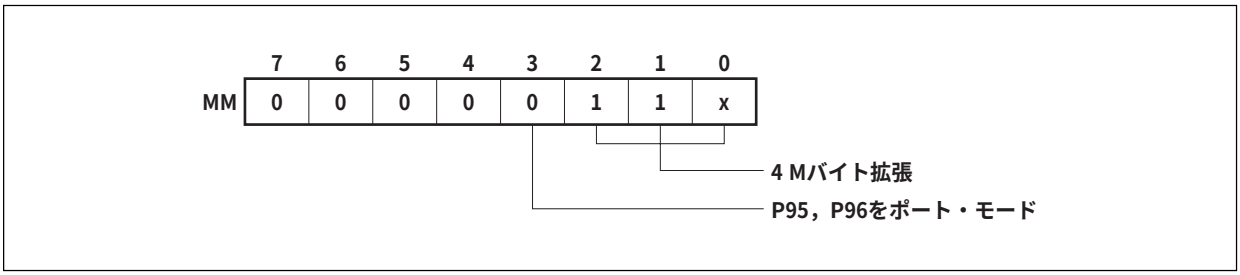
図4－3　**メモリ・マップ**



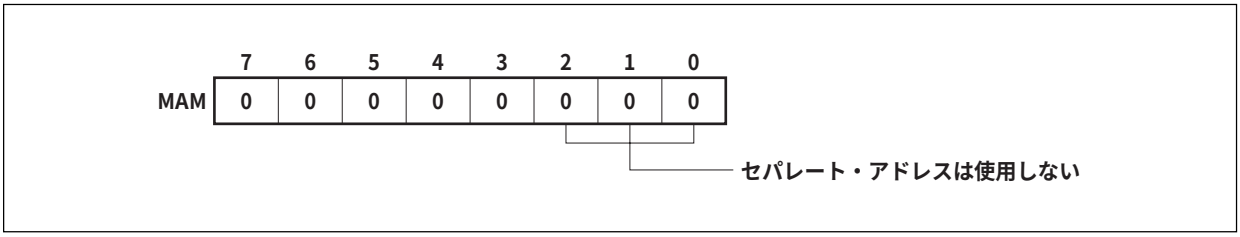
注　LCDは，偶数アドレスに対するバイト・アクセスが有効です。

4. 3. 2　バス・インタフェース関連レジスタの設定

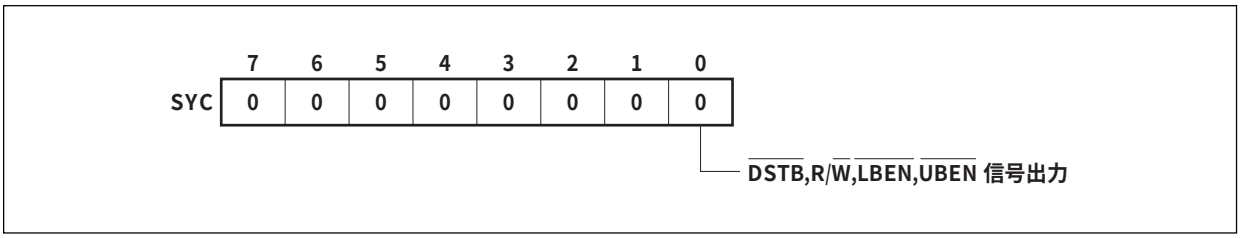
(1) メモリ拡張モード・レジスタ（MM）の設定



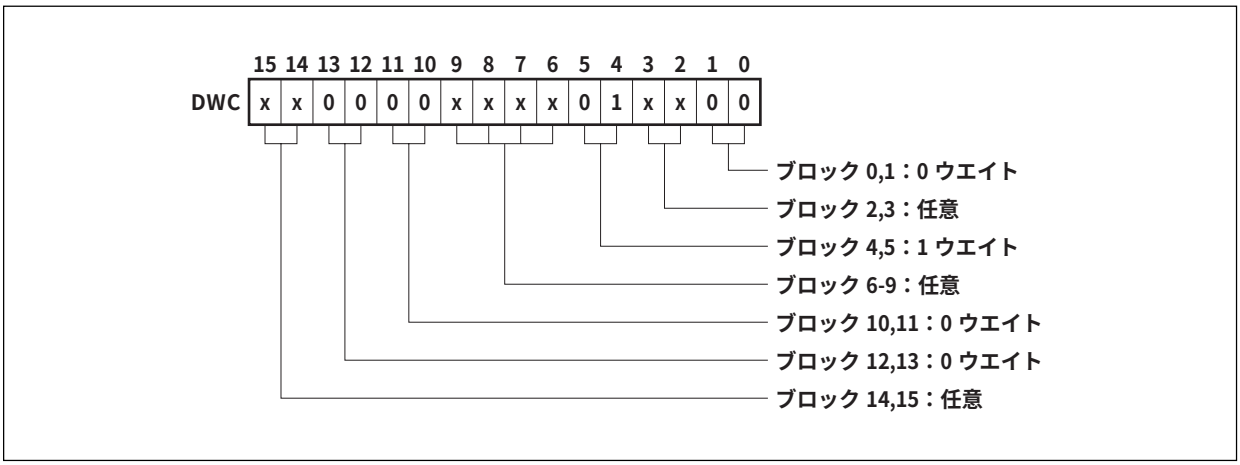
(2) メモリ・アドレス出力モード・レジスタ（MAM）の設定



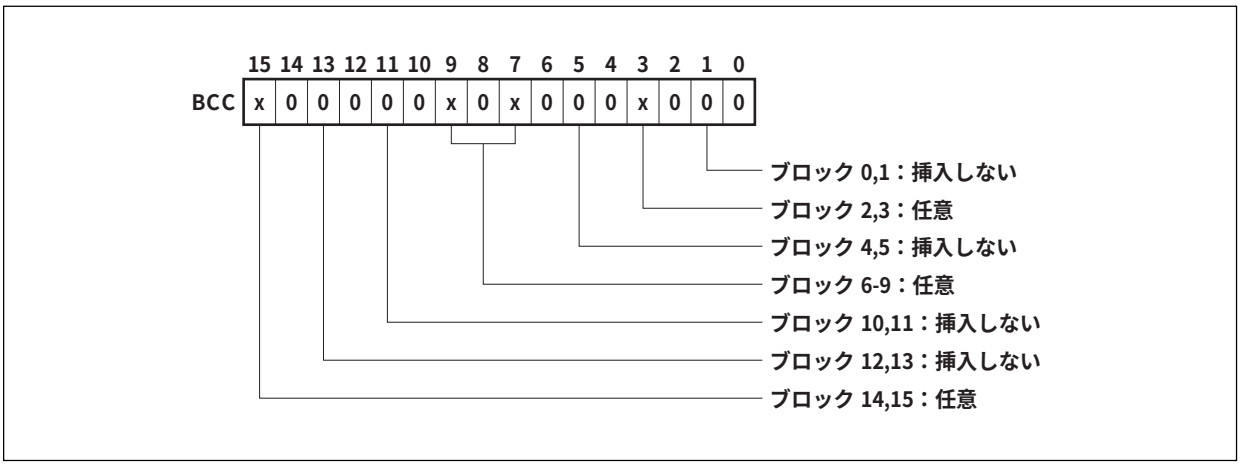
(3) システム制御レジスタ（SYC）の設定



(4) データ・ウエイト・コントロール・レジスタ（DWC）の設定



(5) バス・サイクル・コントロール・レジスタ（BCC）の設定



4.3.3　ポート機能

TB-V850/SA1でのポート機能の設定を次に示します。

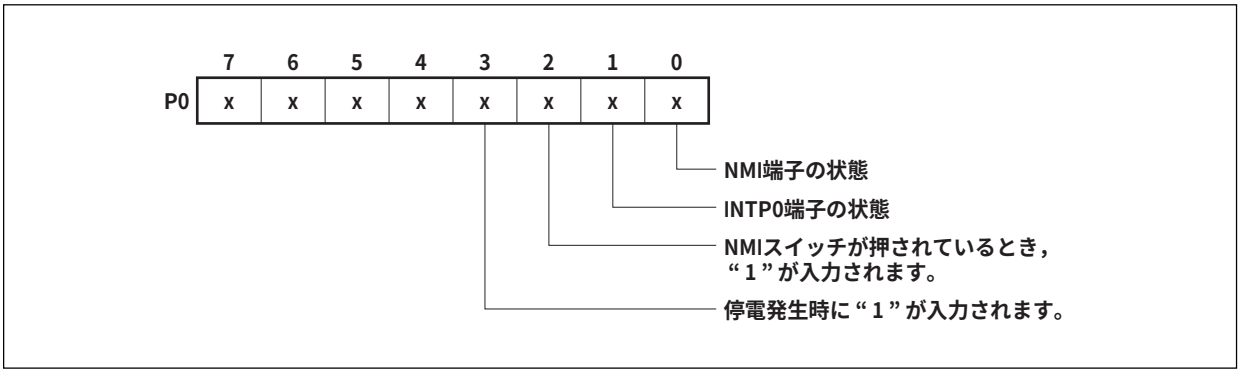
各ポートごとにTB-V850/SA1での使用方法とレジスタの設定を示します。

(1) ポート0

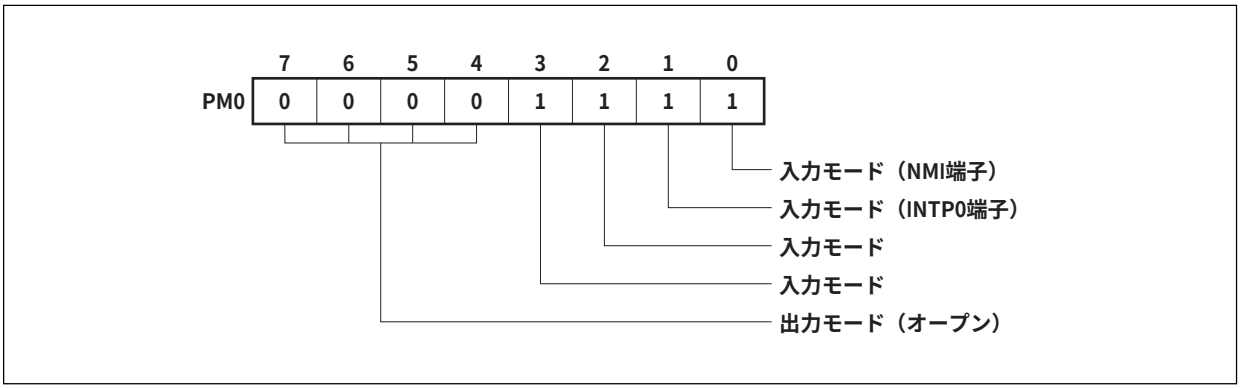
(a) ポート0の使用方法

ポート名称	端子の使用	備　　考
P00	NMI	立ち上がり，立ち下がりの両方で割り込みを発生
P01	INTP0	立ち上がりで割り込みを発生
P02	入力ポート	NMIスイッチの状態モニタ
P03	入力ポート	電源監視ユニットの状態モニタ
P04-P07	未使用（オープン）	出力モードに設定

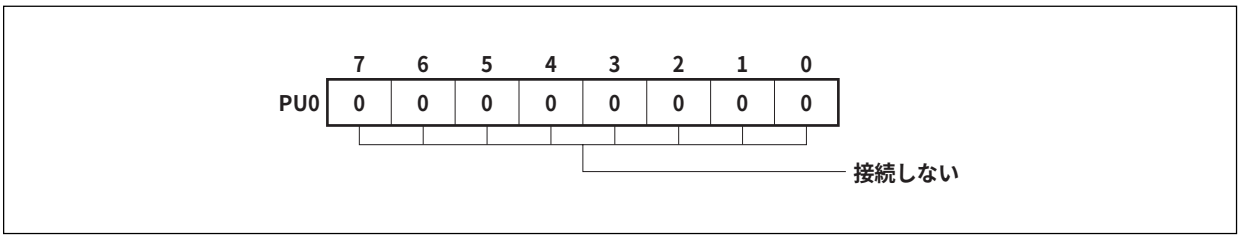
(b) ポート0（P0）のリード時の状態



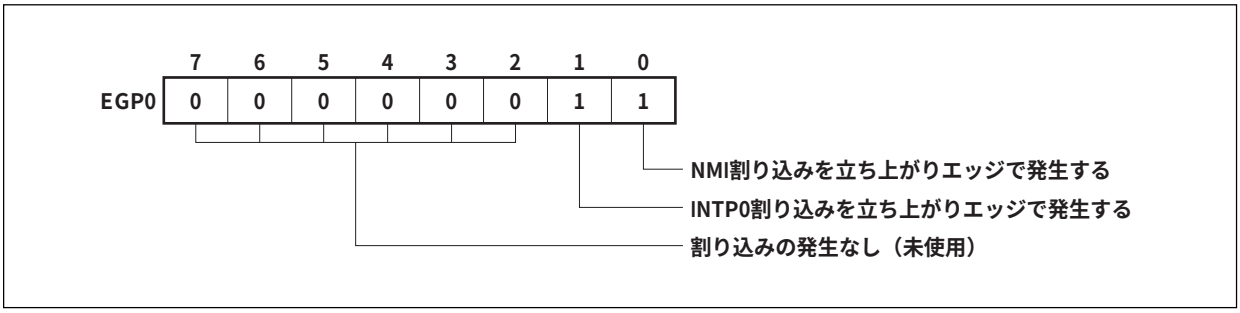
(c) ポート0モード・レジスタ（PM0）の設定



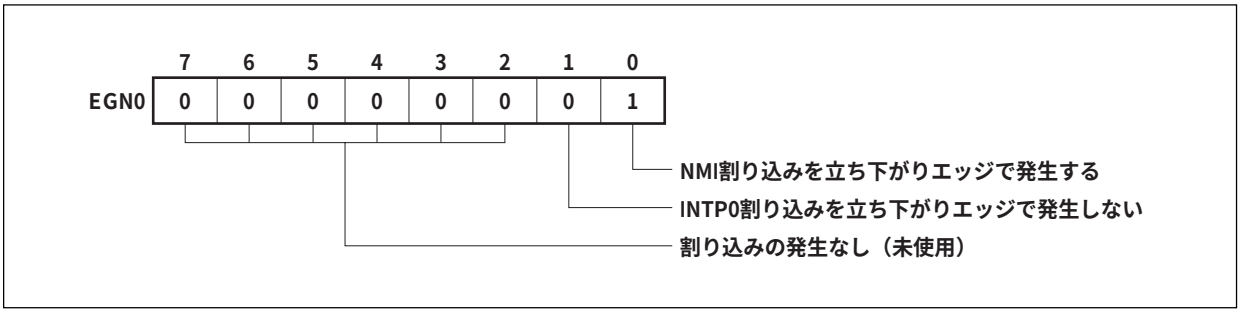
(d) プルアップ抵抗オプション・レジスタ0（PU0）の設定



(e) 立ち上がりエッジ指定レジスタ（EGP0）の設定



(f) 立ち下がりエッジ指定レジスタ（EGN0）の設定

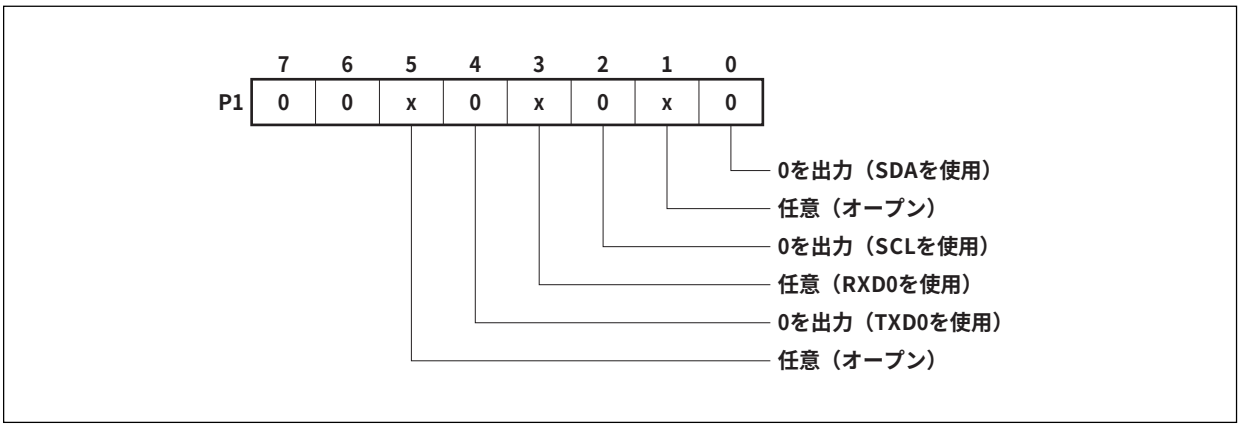


(2) ポート1

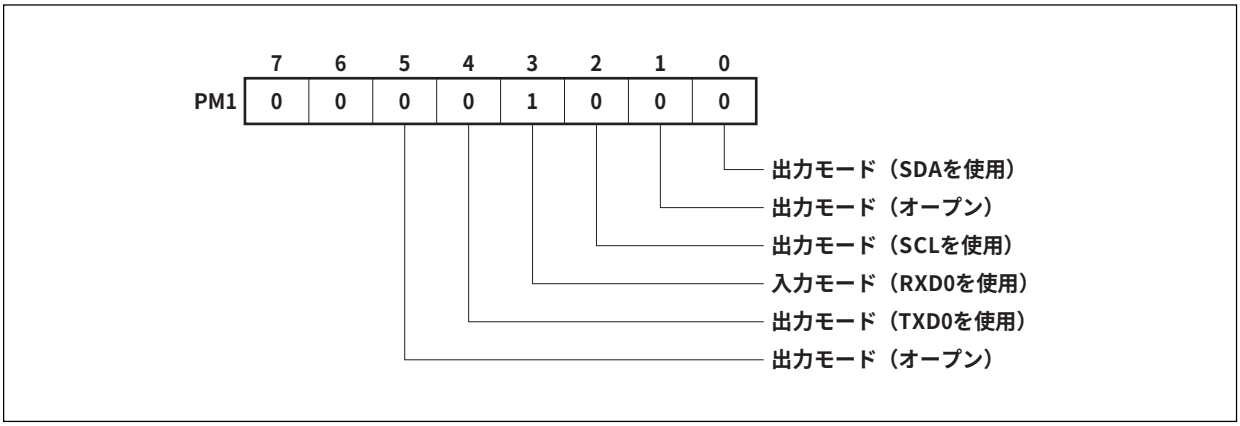
(a) ポート1の使用方法

ポート名称	端子の使用	備　　考
P10	SDA	出力モードでP10を0に設定
P11	未使用（オープン）	出力モードに設定
P12	SCL	出力モードでP12を0に設定
P13	RXD0	入力モードに設定
P14	TXD0	出力モードでP14を0に設定
P15	未使用（オープン）	出力モードに設定

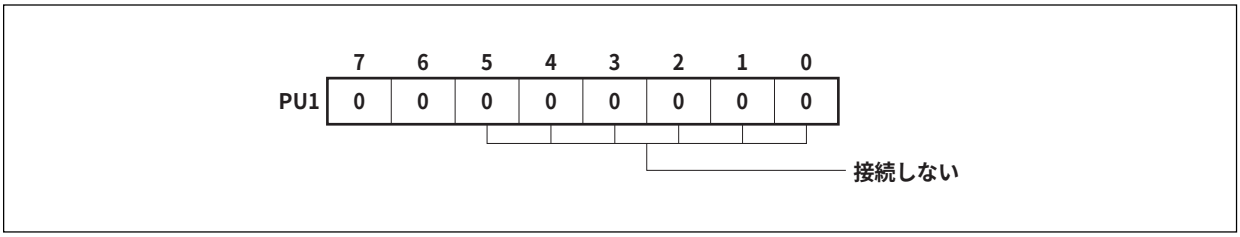
(b) ポート1（P1）の設定



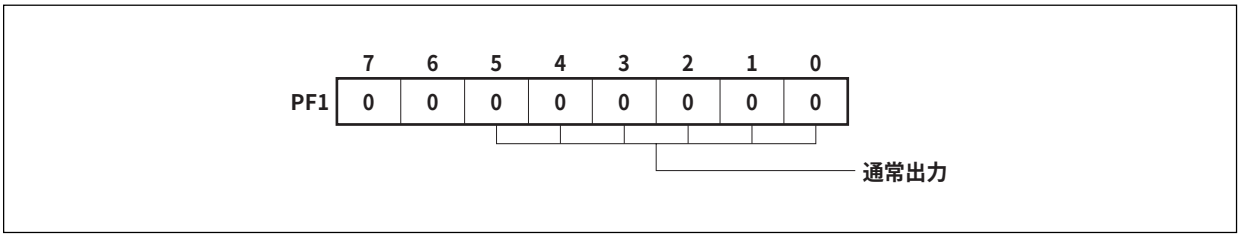
(c) ポート1モード・レジスタ（PM1）の設定



(d) プルアップ抵抗オプション・レジスタ1（PU1）の設定



(e) ポート1ファンクション・レジスタ（PF1）の設定

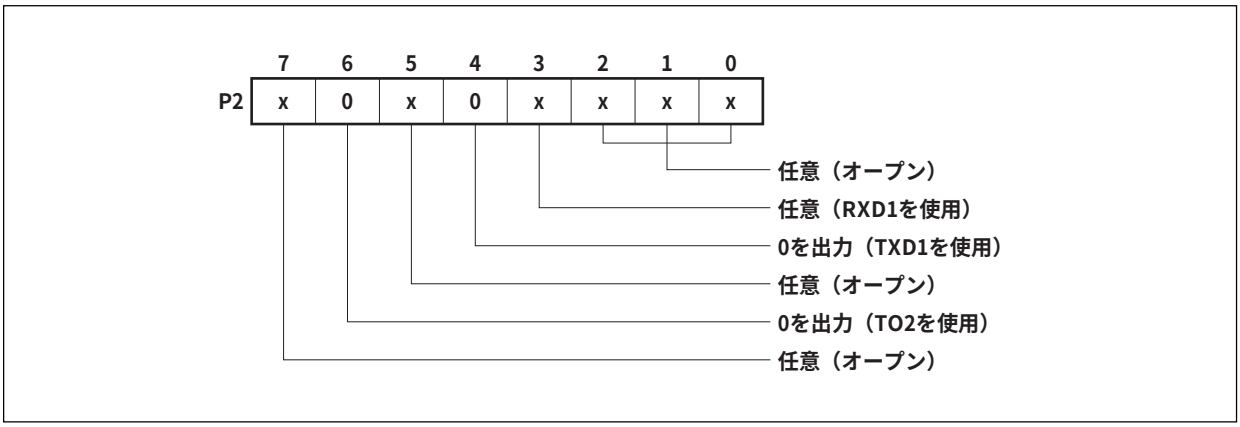


(3) ポート2

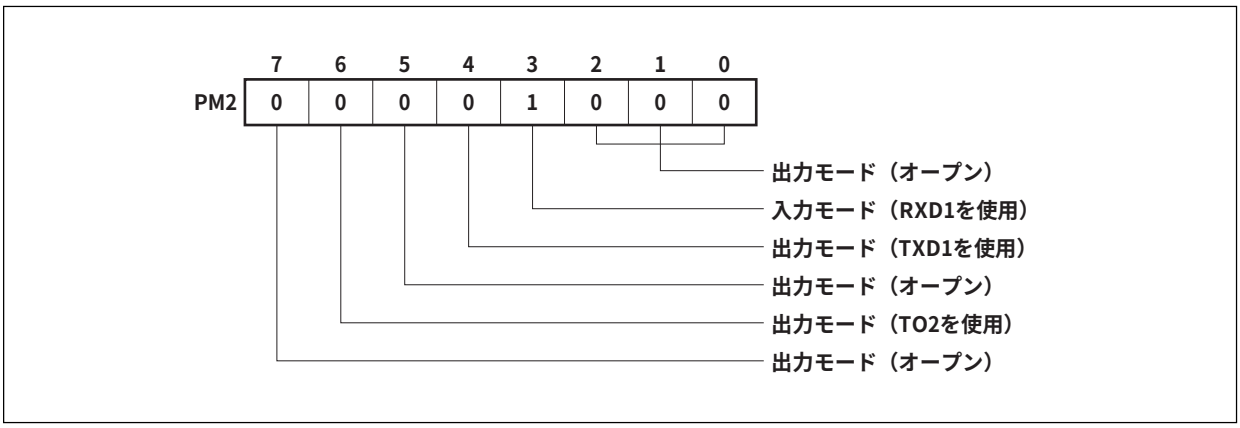
(a) ポート2の使用方法

ポート名称	端子の使用	備　　考
P20-P22, P25, P27	未使用（オープン）	出力モードに設定
P23	RXD1	入力モードに設定
P24	TXD1	出力モードでP24を0に設定
P26	TO2	出力モードでP26を0に設定

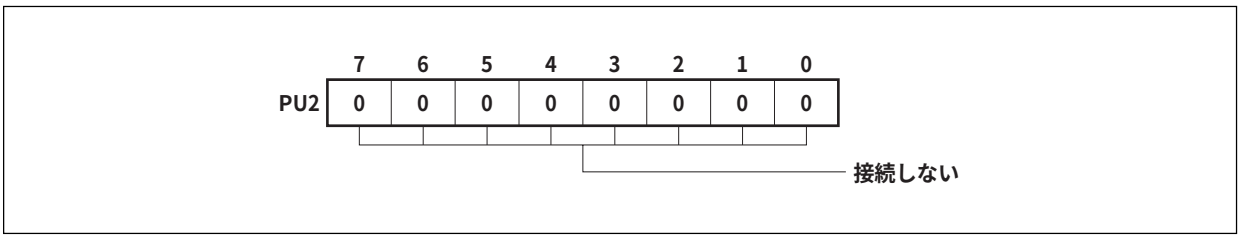
(b) ポート2（P2）の設定



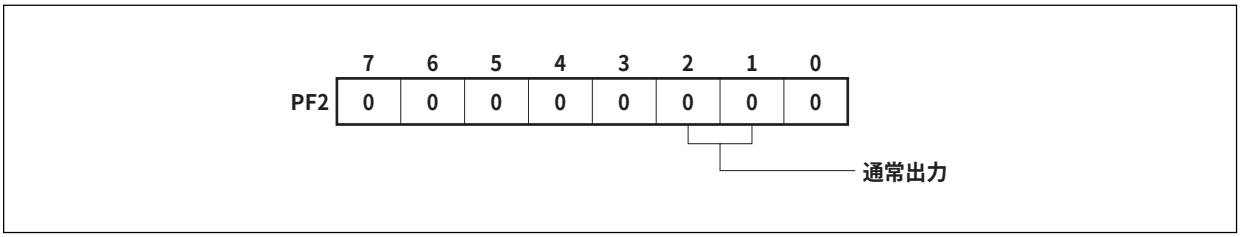
(c) ポート2モード・レジスタ（PM2）の設定



(d) プルアップ抵抗オプション・レジスタ2（PU2）の設定



(e) ポート2ファンクション・レジスタ（PF2）の設定

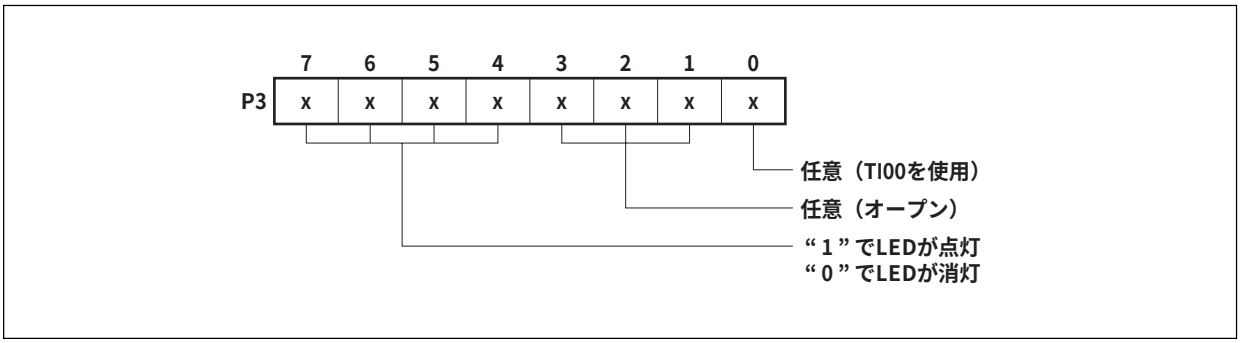


(4) ポート3

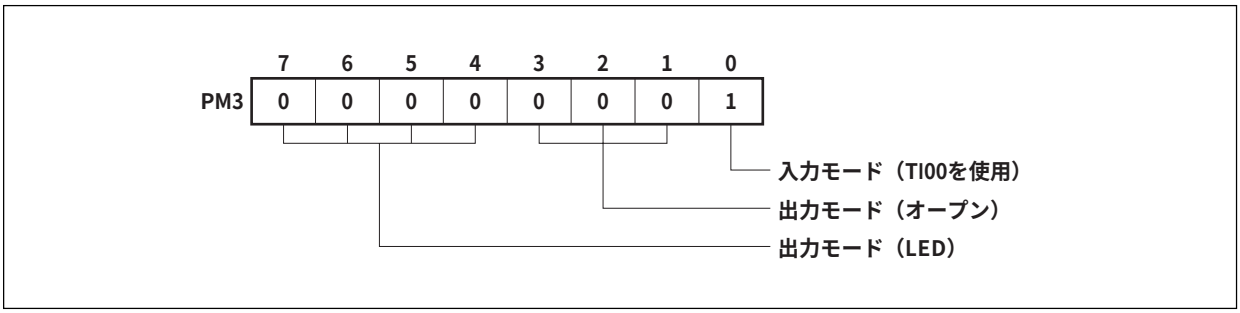
(a) ポート3の使用方法

ポート名称	端子の使用	備　　考
P30	TI00	入力モードに設定
P31-P33	未使用（オープン）	出力モードに設定
P34-P37	出力ポート	LED出力ポート

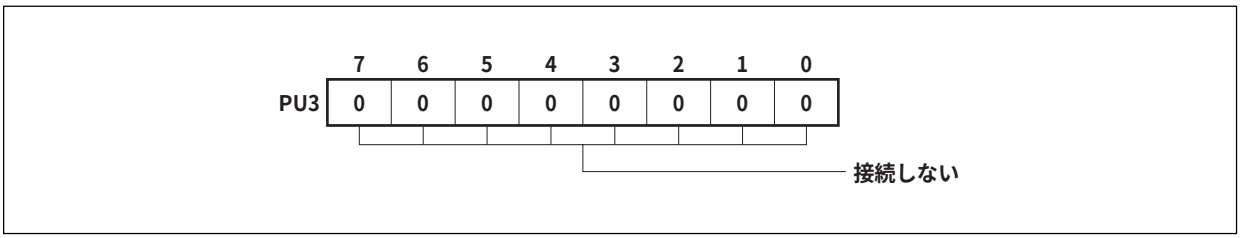
(b) ポート3（P3）の設定



(c) ポート3モード・レジスタ（PM3）の設定



(d) プルアップ抵抗オプション・レジスタ3（PU3）の設定



(5) ポート4，ポート5

(a) ポート4，ポート5の使用方法

メモリ拡張モード・レジスタ（MM）で外部拡張モードを指定した場合は，ポート4（P4），ポート5（P5），ポート4モード・レジスタ（PM4），ポート5モード・レジスタ（PM5）の設定は意味を持ちません。

ポート名称	端子の使用	備　　考
P40-P47, P50-P57	AD0-AD15	MMレジスタで外部拡張モードに指定

(6) ポート6

(a) ポート6の使用方法

メモリ拡張モード・レジスタ（MM）で4 Mバイト拡張モードを指定した場合は，ポート6（P6），ポート6モード・レジスタ（PM6）の設定は意味を持ちません。

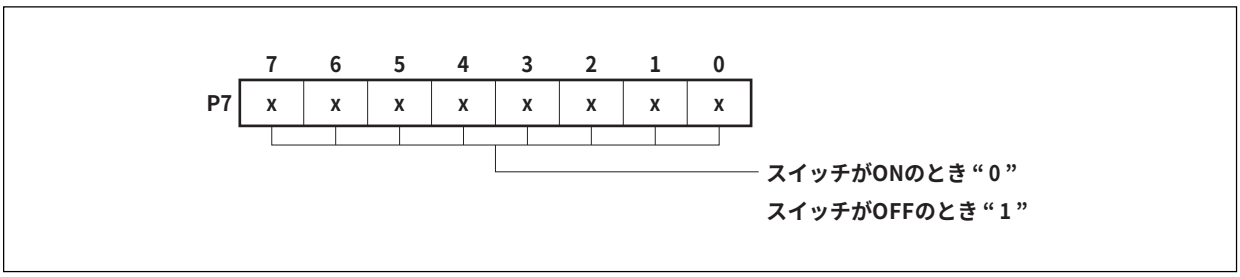
ポート名称	端子の使用	備　　考
P60-P65	A16-A21	MMレジスタで4 Mバイト拡張モードに指定

(7) ポート7

(a) ポート7の使用方法

ポート名称	端子の使用	備　　考
P70-P77	入力ポート	スイッチ入力ポート

(b) ポート7（P7）のリード時の状態

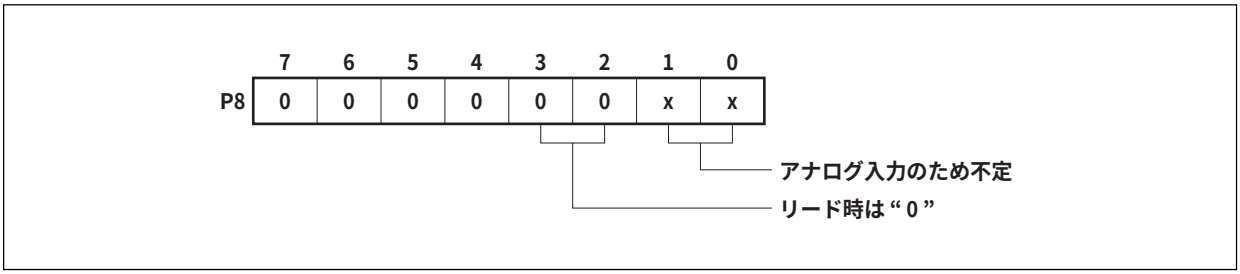


(8) ポート8

(a) ポート8の使用方法

ポート名称	端子の使用	備　　考
P80	ANI8	可変抵抗によるアナログ入力
P81	ANI9	システム電源のモニタ
P82, P83	未使用（AVssに接続）	

(b) ポート8（P8）のリード時の状態



(9) ポート9

(a) ポート9の使用方法

ポート9（P9），ポート9モード・レジスタ（PM9）の設定は意味を持ちません。

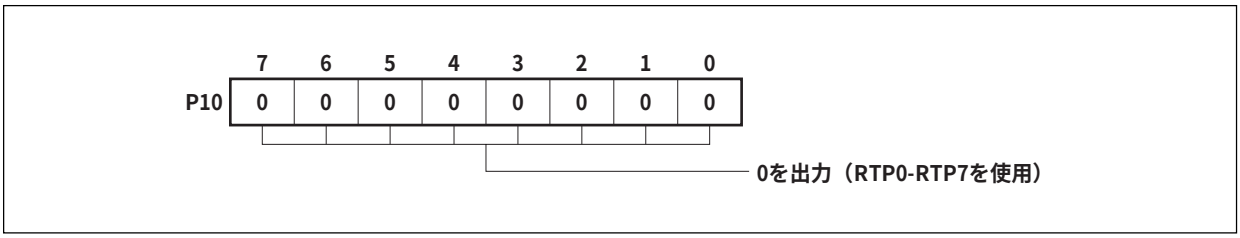
ポート名称	端子の使用	備　　考
P90	$\overline{\text{LBEN}}$	MMレジスタでとSYCレジスタで指定
P91	$\overline{\text{UBEN}}$	
P92	R/$\overline{\text{W}}$	
P93	$\overline{\text{DSTB}}$	
P94	ASTB	
P95	$\overline{\text{HLDAK}}$	
P96	$\overline{\text{HLDRQ}}$	

(10) ポート10

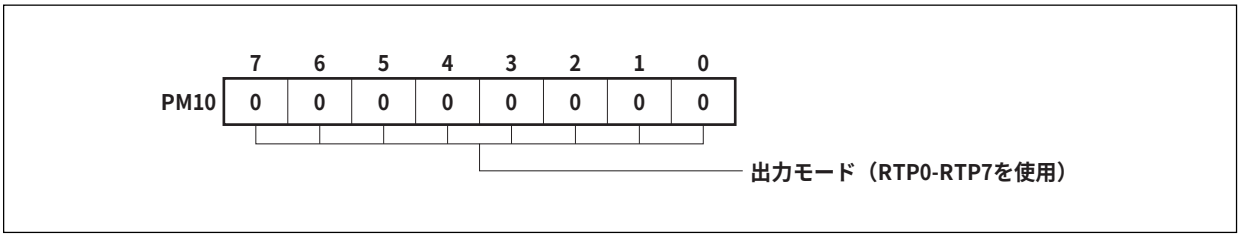
(a) ポート10の使用方法

ポート名称	端子の使用	備　　考
P100-P107	RTP0-RTP7	出力モードで0に設定

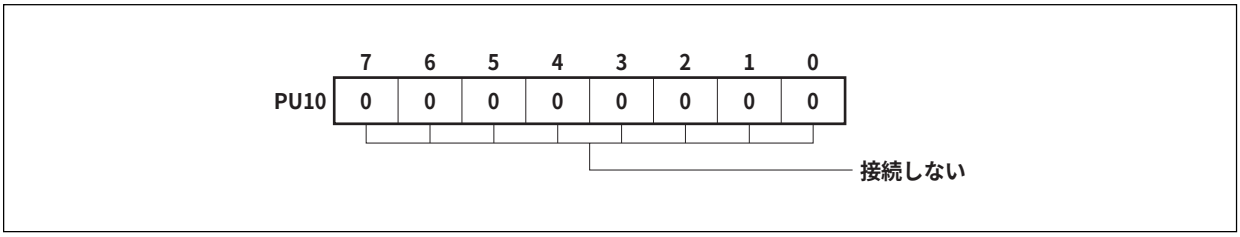
(b) ポート10（P10）の設定



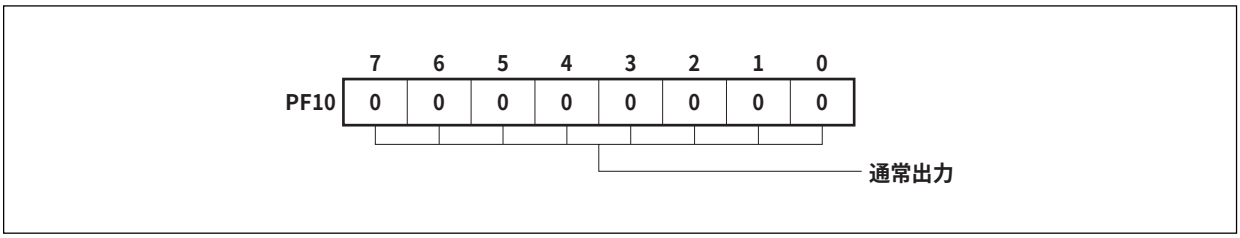
(c) ポート10モード・レジスタ（PM10）の設定



(d) プルアップ抵抗オプション・レジスタ10（PU10）の設定



(e) ポート10ファンクション・レジスタ（PF10）の設定

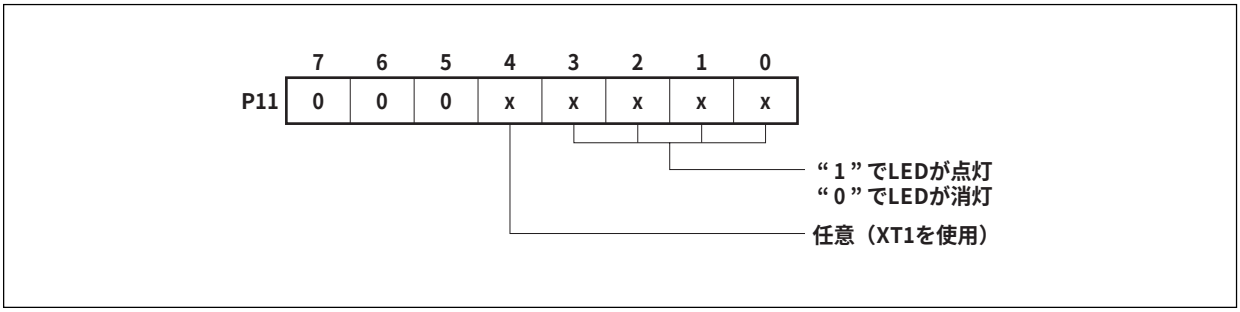


（11）ポート11

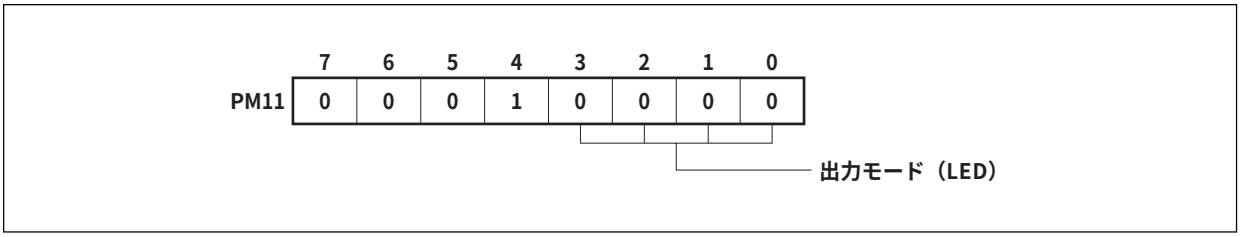
（a）ポート11の使用方法

ポート名称	端子の使用	備　　考
P110-P113	出力ポート	LED出力ポート
P114	XT1	入力モードに固定

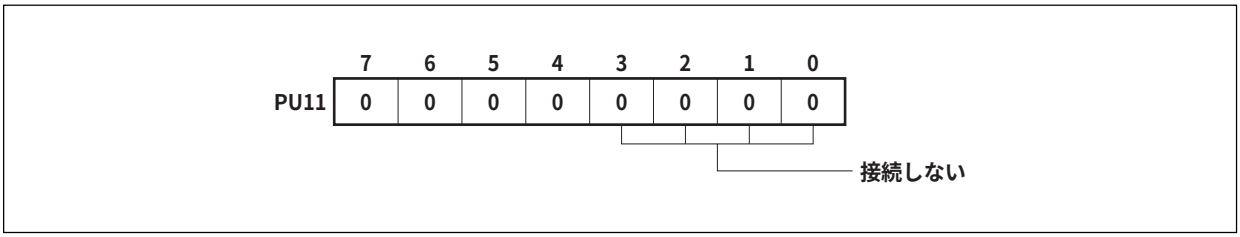
（b）ポート11（P11）の設定



（c）ポート11モード・レジスタ（PM11）の設定



（d）プルアップ抵抗オプション・レジスタ11（PU11）の設定

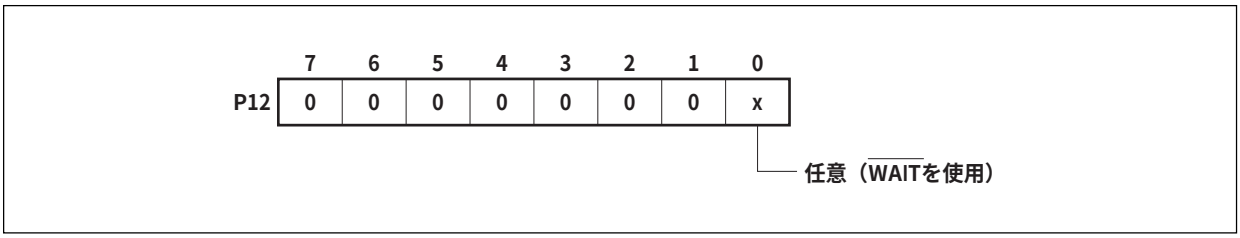


（12）ポート12

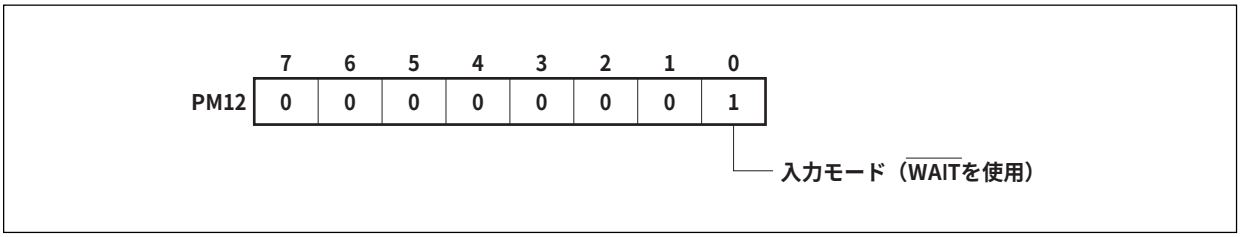
（a）ポート12の使用方法

ポート名称	端子の使用	備　　考
P120	$\overline{\text{WAIT}}$	入力モードに指定

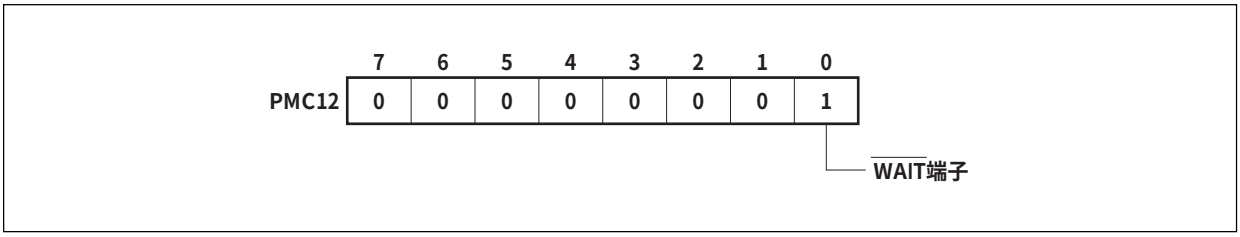
（b）ポート12（P12）の設定



（c）ポート12モード・レジスタ（PM12）の設定



（d）ポート12モード・コントロール・レジスタ（PMC12）の設定



4.4　アプリケーション・プログラム例

TB-V850/SA1の動作プログラム例を次に示します。プログラム例は，アセンブリ言語およびC言語で記述しています（4. 4. 10, 4. 4. 11はアセンブリ言語のみ）。

なお，アセンブリ言語のプログラムでは，アセンブラ初期設定プログラムとリンクされていることを前提としています。

C言語のプログラムでは，Cスタートアップ・モジュールおよびCリセット処理とリンクされていることを前提としています。

CPUの供給クロックは，メイン・クロック（16 MHz），サブクロック（32.768 kHz）です。

(1) アセンブラ初期設定プログラム

```
###### リセット割り込み処理.
.text
        .section        "RESET"
        jr      init

.text
############ I/Oレジスタの初期設定を行う  ##############
init:
        mov     0x06,   r11
        st.b    r11,    MM[r0]                  -- メモリ拡張モード・レジスタの設定
        mov     0x00,   r11
        st.b    r11,    MAM[r0]                 -- メモリ・アドレス出力モード・レジスタの設定
        mov     0x00,   r11
        st.b    r11,    SYC[r0]                 -- システム制御レジスタの設定
        movea   0x10,   r0, r11
        st.h    r11,    DWC[r0]                 -- データ・ウエイト・コントロール・レジスタの設定
        mov     0x00,   r11
        st.h    r11,    BCC[r0]                 -- バス・サイクル・コントロール・レジスタの設定
        mov     0x0f,   r11
        st.b    r11,    PM0[r0]                 -- ポート0モード・レジスタの設定
        mov     0x00,   r11
        st.b    r11,    PU0[r0]                 -- プルアップ抵抗オプション・レジスタ0の設定
        mov     0x03,   r11
        st.b    r11,    EGP0[r0]                -- 立ち上がりエッジ指定レジスタの設定
        mov     0x01,   r11
        st.b    r11,    EGN0[r0]                -- 立ち下がりエッジ指定レジスタの設定
        mov     0x08,   r11
        st.b    r11,    PM1[r0]                 -- ポート1モード・レジスタの設定
        mov     0x00,   r11
        st.b    r11,    P1[r0]                  -- ポート1レジスタの設定
        mov     0x00,   r11
        st.b    r11,    PU1[r0]                 -- プルアップ抵抗オプション・レジスタ1の設定
        mov     0x00,   r11
        st.b    r11,    PF1[r0]                 -- ポート1ファンクション・レジスタの設定
        mov     0x08,   r11
        st.b    r11,    PM2[r0]                 -- ポート2モード・レジスタの設定
        mov     0x00,   r11
        st.b    r11,    P2[r0]                  -- ポート2レジスタの設定
        mov     0x00,   r11
        st.b    r11,    PU2[r0]                 -- プルアップ抵抗オプション・レジスタ2の設定
        mov     0x00,   r11
        st.b    r11,    PF2[r0]                 -- ポート2ファンクション・レジスタの設定
        mov     0x01,   r11
        st.b    r11,    PM3[r0]                 -- ポート3モード・レジスタの設定
        mov     0x00,   r11
        st.b    r11,    P3[r0]                  -- ポート3レジスタの設定
        mov     0x00,   r11
        st.b    r11,    PU3[r0]                 -- プルアップ抵抗オプション・レジスタ3の設定
        mov     0x00,   r11
        st.b    r11,    PM10[r0]                -- ポート10モード・レジスタの設定
```

```
mov     0x00,   r11
st.b    r11,    P10[r0]                 -- ポート10レジスタの設定
mov     0x00,   r11
st.b    r11,    PU10[r0]                -- プルアップ抵抗オプション・レジスタ10の設定
mov     0x00,   r11
st.b    r11,    PF10[r0]                -- ポート10ファンクション・レジスタの設定
mov     0x10,   r11
st.b    r11,    PM11[r0]                -- ポート11モード・レジスタの設定
mov     0x00,   r11
st.b    r11,    P11[r0]                 -- ポート11レジスタの設定
mov     0x00,   r11
st.b    r11,    PU11[r0]                -- プルアップ抵抗オプション・レジスタ11の設定
mov     0x01,   r11
st.b    r11,    PM12[r0]                -- ポート12モード・レジスタの設定
mov     0x01,   r11
st.b    r11,    PMC12[r0]               -- ポート12モード・コントロール・レジスタの設定

stsr    5,      r12
mov     r12,    r13
or      0x20,   r13
mov     0x00,   r11
ldsr    r13,    5
st.b    r0,     PRCMD[r0]               -- PRCMDへの書き込み
## プロセッサ・コントロール・レジスタの設定
st.b    r11,    PCC[r0]                 -- CPUクロックをfxxに選択
ldsr    r12,    5
nop                                     -- ダミー命令(5命令)
nop
nop
nop
nop
jr      start
```

(2) Cスタートアップ・モジュール

```
#   Copyright (C) NEC Corporation 1994,1995,1996
#   All rights reserved by NEC Corporation. This program must be used solely
#   for the purpose for which it was furnished by NEC Corporation. No part
#   of this program may be reproduced or disclosed to others, in any form,
#   without the prior written permission of NEC Corporation.

#       @(#)crtN850.s   1.13 96/05/31 17:10:34

#===============================================================================
# NAME
#    crtN850.s -  start up module for ca850
#
# DESCRIPTIONS:
#      This assembly program is a sample of start-up module for ca850.
#    If you modified this program, you must assemble this file, and
#    locate a given directory.
#
#    Unless -G is specified, sections are located as the following.
#
#                            |             :             |
#                            |             :             |
#                   tp->-    +---------------------------+ __start     __tp_TEXT
#                            |          start up         |
#                            |---------------------------|
#             text section   |                           |
#                            |        user program       |
#                            |                           |
#                            |---------------------------|
#                            |          library          |
#                        -   +---------------------------+ __argc
#                            |              0            |
#                            |---------------------------| __argv
#             data section   |            #.L16          |
#                            |---------------------------| .L16
#                            |      0x0,0x0,0x0,0x0      |
#                        -   +---------------------------+
#                            |                           |
#            sdata section   |                           |
#                            |                           |
#                   gp->-    +---------------------------+ __ssbss
#                            |                           |
#             sbss section   |                           |
#                            |                           |
#                            +---------------------------+ __stack   __esbss    __sbss
#                            |        stack area         |
#              bss section   |                           |
#                            |       0x10000 bytes       |
#                   sp->-    +---------------------------+ __stack + STACKSIZE   __ebss
#                            |             :             |
#                            |             :             |
#                        -   +---------------------------+
#           sedata section   |                           |
#                            |                           |
#                        -   +---------------------------+ __ssebss
#            sebss section   |                           |
#                        -   +---------------------------+ __esebss
#                            |             :             |
#                   ep->-    +---------------------------+ __ep_DATA
#           tidata section   |                           |
#                        -   +---------------------------+
#           sidata section   |                           |
#                        -   +---------------------------+
#                            |             :             |
#                            |             :             |
#
#===============================================================================
```

```
#-------------------------------------------------------------------------
#       register mode
#-------------------------------------------------------------------------
        .option reg_mode 5 5

#-------------------------------------------------------------------------
#       special symbols
#-------------------------------------------------------------------------
        .extern __tp_TEXT, 4
        .extern __gp_DATA, 4
        .extern __ep_DATA, 4
        .extern __ssbss, 4
        .extern __esbss, 4
        .extern __sbss, 4
        .extern __ebss, 4
        .extern __ssebss, 4
        .extern __esebss, 4


#-------------------------------------------------------------------------
#        C program main function
#-------------------------------------------------------------------------
        .extern _main


#-------------------------------------------------------------------------
#       for argv
#-------------------------------------------------------------------------
        .data
        .size   __argc, 4
        .align  4
__argc:
        .word   0
        .size   __argv, 4
__argv:
        .word   #.L16
.L16:
        .byte   0
        .byte   0
        .byte   0
        .byte   0


#-------------------------------------------------------------------------
#       dummy data declaration for creating sbss section
#-------------------------------------------------------------------------
        .sbss
        .lcomm  __sbss_dummy, 0, 0


#-------------------------------------------------------------------------
#       dummy data declaration for creating sebss section
#-------------------------------------------------------------------------
        .sebss
        .lcomm  __sebss_dummy, 0, 0


#-------------------------------------------------------------------------
#       system stack
#-------------------------------------------------------------------------
        .set    STACKSIZE, 0x4000
        .bss
        .lcomm  __stack, STACKSIZE, 4
```

```
#------------------------------------------------------------------------------
#       start up
#               pointers:  tp - text pointer
#                          gp - global pointer
#                          sp - stack pointer
#                          ep - element pointer
#               mask reg:  r20 - 0xff
#                          r21 - 0xffff
#       exit status is set to r10
#------------------------------------------------------------------------------
        .text
        .align  4

        .globl  __start
        .globl  _exit
        .globl  __exit
__start:
        mov     #__tp_TEXT, tp                  -- set tp register
        mov     #__gp_DATA, gp                  -- set gp register offset
        add     tp, gp                          -- set gp register
        mov     #__stack+STACKSIZE, sp          -- set sp register
        mov     #__ep_DATA, ep                  -- set ep register
#
        .option nowarning
        movea   0xff,   r0, r20                 -- set mask register
        movea   0xffff, r0, r21
        andi    0xffff, r21, r21                -- set mask register
        .option warning
#
        mov     0x06,   r11
        st.b    r11,    MM[r0]                  -- メモリ拡張モード・レジスタの設定
        mov     0x00,   r11
        st.b    r11,    MAM[r0]                 -- メモリ・アドレス出力モード・レジスタの設定
        mov     0x00,   r11
        st.b    r11,    SYC[r0]                 -- システム制御レジスタの設定
        movea   0x10,   r0, r11
        st.h    r11,    DWC[r0]                 -- データ・ウエイト・コントロール・レジスタの設定
        mov     0x00,   r11
        st.h    r11,    BCC[r0]                 -- バス・サイクル・コントロール・レジスタの設定
        mov     0x0f,   r11
        st.b    r11,    PM0[r0]                 -- ポート0モード・レジスタの設定
        mov     0x00,   r11
        st.b    r11,    PU0[r0]                 -- プルアップ抵抗オプション・レジスタ0の設定
        mov     0x03,   r11
        st.b    r11,    EGP0[r0]                -- 立ち上がりエッジ指定レジスタの設定
        mov     0x01,   r11
        st.b    r11,    EGN0[r0]                -- 立ち下がりエッジ指定レジスタの設定
        mov     0x08,   r11
        st.b    r11,    PM1[r0]                 -- ポート1モード・レジスタの設定
        mov     0x00,   r11
        st.b    r11,    P1[r0]                  -- ポート1レジスタの設定
        mov     0x00,   r11
        st.b    r11,    PU1[r0]                 -- プルアップ抵抗オプション・レジスタ1の設定
        mov     0x00,   r11
        st.b    r11,    PF1[r0]                 -- ポート1ファンクション・レジスタの設定
        mov     0x08,   r11
        st.b    r11,    PM2[r0]                 -- ポート2モード・レジスタの設定
        mov     0x00,   r11
        st.b    r11,    P2[r0]                  -- ポート2レジスタの設定
        mov     0x00,   r11
        st.b    r11,    PU2[r0]                 -- プルアップ抵抗オプション・レジスタ2の設定
        mov     0x00,   r11
        st.b    r11,    PF2[r0]                 -- ポート2ファンクション・レジスタの設定
        mov     0x01,   r11
        st.b    r11,    PM3[r0]                 -- ポート3モード・レジスタの設定
        mov     0x00,   r11
        st.b    r11,    P3[r0]                  -- ポート3レジスタの設定
        mov     0x00,   r11
```

```
        st.b    r11,    PU3[r0]                 -- プルアップ抵抗オプション・レジスタ3の設定
        mov     0x00,   r11
        st.b    r11,    PM10[r0]                -- ポート10モード・レジスタの設定
        mov     0x00,   r11
        st.b    r11,    P10[r0]                 -- ポート10レジスタの設定
        mov     0x00,   r11
        st.b    r11,    PU10[r0]                -- プルアップ抵抗オプション・レジスタ10の設定
        mov     0x00,   r11
        st.b    r11,    PF10[r0]                -- ポート10ファンクション・レジスタの設定
        mov     0x10,   r11
        st.b    r11,    PM11[r0]                -- ポート11モード・レジスタの設定
        mov     0x00,   r11
        st.b    r11,    P11[r0]                 -- ポート11レジスタの設定
        mov     0x00,   r11
        st.b    r11,    PU11[r0]                -- プルアップ抵抗オプション・レジスタ11の設定
        mov     0x01,   r11
        st.b    r11,    PM12[r0]                -- ポート12モード・レジスタの設定
        mov     0x01,   r11
        st.b    r11,    PMC12[r0]               -- ポート12モード・コントロール・レジスタの設定

        stsr    5,      r12
        mov     r12,    r13
        or      0x20,   r13
        mov     0x00,   r11
        ldsr    r13,    5
        st.b    r0,     PRCMD[r0]               -- PRCMDへの書き込み
        ## プロセッサ・コントロール・レジスタの設定
        st.b    r11,    PCC[r0]                 -- CPUクロックをfxxに選択
        ldsr    r12,    5
        nop                                     -- ダミー命令(5命令)
        nop
        nop
        nop
        nop
#
        mov     #__ssbss, r13                   -- clear sbss section
        mov     #__esbss, r12
        cmp     r12, r13
        jnl     .L11
.L12:
        st.w    r0, [r13]
        add     4, r13
        cmp     r12, r13
        jl      .L12
.L11:
#
        mov     #__sbss, r13                    -- clear bss section
        mov     #__ebss, r12
        cmp     r12, r13
        jnl     .L14
.L15:
        st.w    r0, [r13]
        add     4, r13
        cmp     r12, r13
        jl      .L15
.L14:
#
        mov     #__ssebss, r13                  -- clear sebss section
        mov     #__esebss, r12
        cmp     r12, r13
        jnl     .L17
.L18:
        st.w    r0, [r13]
        add     4, r13
        cmp     r12, r13
        jl      .L18
.L17:
#
        ld.w    $__argc, r6                     -- set argc
```

```
        movea   $__argv, gp, r7                 -- set argv
        jarl    _main, lp                       -- call main function
__exit:
        halt                                    -- end of program
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
__startend:
#                                                               #
#------------------- end of start up module -------------------#
#
```

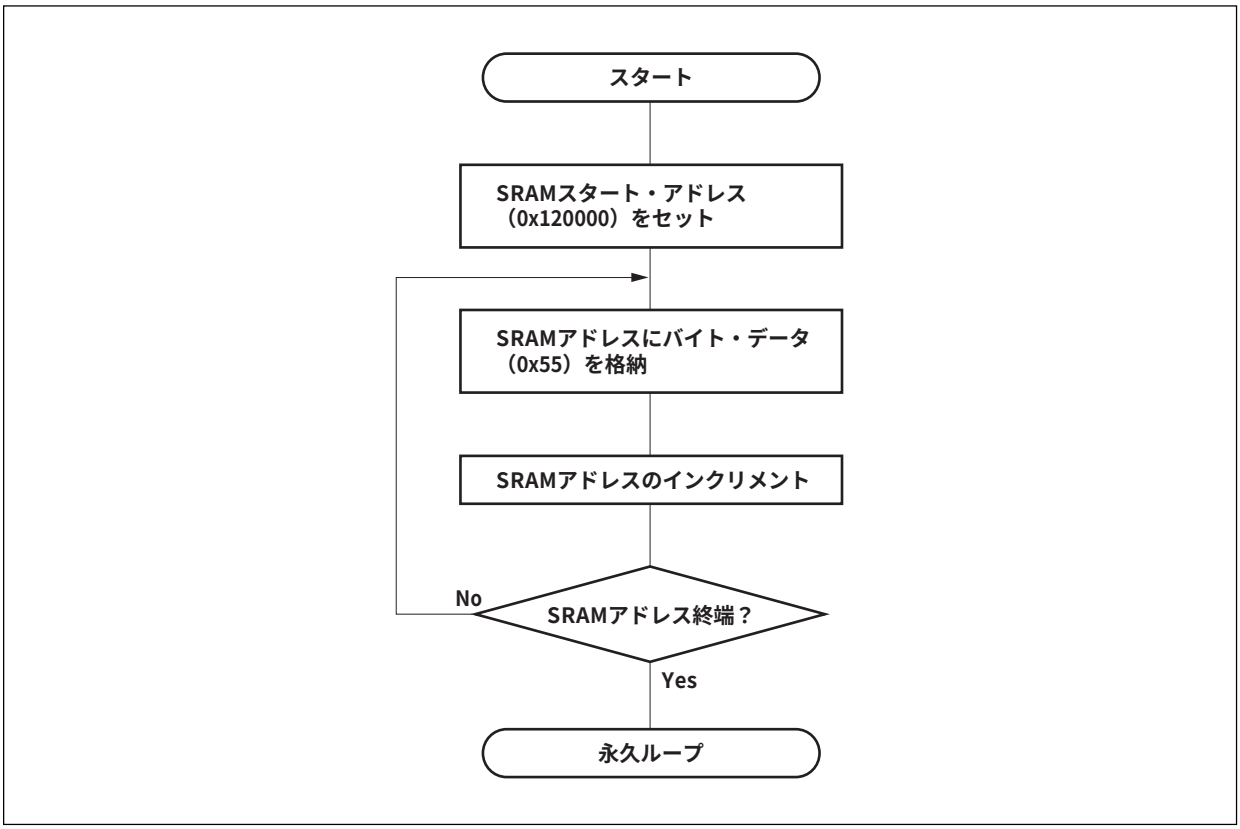
(3) Cリセット処理

```
        .option reg_mode 5 5
        .extern __start

.text
        .section        "RESET"
        di
        jr      __start
```

4.4.1　外部メモリのフィル・プログラム

外部SRAM領域（0x120000-0x13FFFF）に対して，1バイトずつ固定データ（0x55）をライトしていきます。

（1）フロー・チャート



(2) アセンブラ・プログラム

```
.globl  start

.text
.set    sram_addr,        0x120000              -- SRAMアドレス
.set    sram_addr_end,    0x140000              -- SRAMアドレス・エンド

        .align 4
start:
        movea   sram_addr & 0x00ffff, r0, r10
        movhi   sram_addr >> 16, r10, r10       -- SRAMアドレス
        movea   0x55,             r0, r11       -- 55:=フィル･データ
write_loop:
        st.b    r11,              0[r10]        -- データのライト
        add     1,                r10           -- 書き込みアドレス・インクリメント
        cmp     sram_addr_end,    r10           -- SRAM アドレス・エンド？
        bnz     write_loop
fv_loop:
        br      fv_loop
```

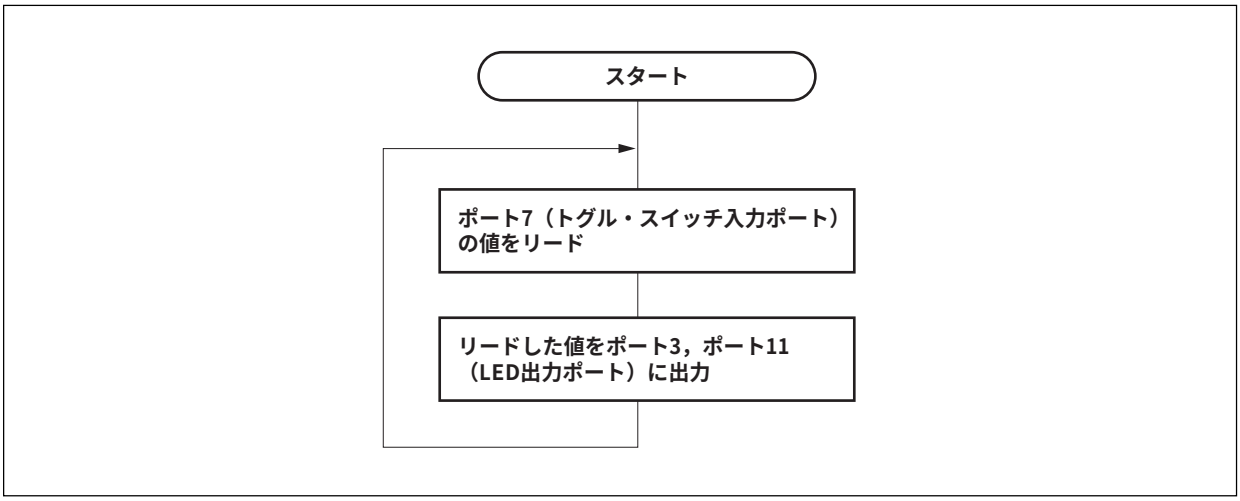
(3) Cプログラム

```
#pragma ioreg                                   /* 周辺I/Oレジスタ名を使用可能にする */

/* ------------------------------------------------------
 *  メイン処理
 *------------------------------------------------------
 */
void main(void)
{
    char *p ;                                   /* SRAMアドレス・ポインタ */

    p = (char *)0x120000 ;                      /* SRAMアドレス先頭ポインタ */

    for ( ; p < (char *)0x140000 ; p++ ){       /* SRAMアドレスの終端までループする */
        *p = 0x55 ;                             /* 55でメモリ・ライト */
    }
    /* 永久ループ */
    for ( ; ; ) {
    }
}
```

4.4.2　スイッチ入力とLEDランプの点灯

ポート7（トグル・スイッチ）の状態をリードして，リード・データをポート3，ポート11（LED）にライトします。

(1) フロー・チャート



(2) アセンブラ・プログラム

```
.globl  start

.text
        .align 4
start:
        ld.b    P7[r0],   r10                   -- スイッチ入力ポートのリード
        mov     r10,      r11                   -- r11:=r10
        and     0x0f,     r11                   -- ポート7の下位4ビットを
        shl     4,        r11                   -- ポート3の上位4ビットに出力
        st.b    r11,      P3[r0]                -- LED点灯
        and     0xf0,     r10                   -- ポート7の上位4ビットを
        shr     4,        r10                   -- ポート11の下位4ビットに出力
        st.b    r10,      P11[r0]               -- LED点灯
        br      start
```

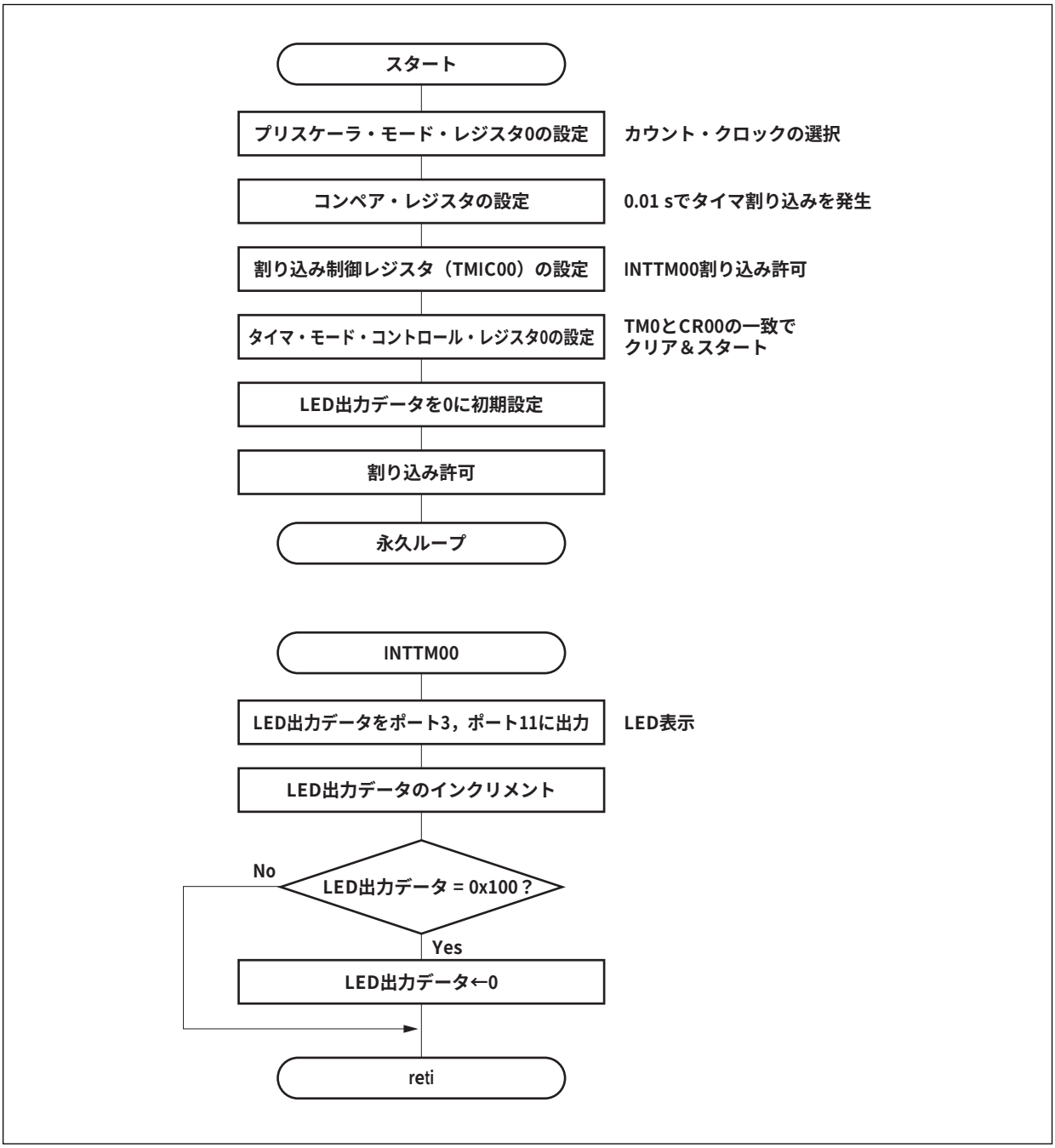
(3) Cプログラム

```
#pragma ioreg                                   /* 周辺I/Oレジスタ名を使用可能にする */

/* ------------------------------------------------------
 *  メイン処理
 *------------------------------------------------------
 */
void main(void)
{
    char isw ;                                  /* スイッチ入力ポート */
    char p3out ;                                /* ポート3出力値  */
    char p11out ;                               /* ポート11出力値  */

    /* 永久ループ */
    for ( ; ; ) {
        isw = P7 ;                              /* スイッチ入力ポート・リード */
        /* iswの下位4ビットを4ビット左シフトしポート3に出力 */
        P3 = ( isw & 0x0f ) << 4 ;
        /* iswの上位4ビットを4ビット右シフトしポート11に出力 */
        P11 = ( isw & 0xf0 ) >> 4 ;
    }
}
```

4.4.3　タイマ割り込みによるLEDランプの点灯

タイマ割り込みにより，LEDランプをインクリメント点灯します。タイマ割り込み間隔は，10 msに設定します。

(1) フロー・チャート



(2) アセンブラ・プログラム

```
.globl  start

## 割り込みハンドラのセット
.section "INTTM00"                              -- INTTM00割り込みハンドラ
        jr      _int_tm00                       -- 割り込み処理へ

.text
        .align 4
start:
        mov     0x01,       r10
        ## カウント・クロック:=fxx/16(1MHz:fxx=16MHz)
        st.b    r10,     PRM0[r0]               -- プリスケーラ・モード・レジスタ0の設定
        movea   0x2710,  r0, r10                -- 1MHz*0x2710=0.01sec
        st.h    r10,     CR00[r0]               -- コンペア・レジスタの設定
        mov     0x07,    r10
        st.b    r10,     TMIC00[r0]             -- INTTM00 割り込み許可
        mov     0x0c,    r10
        ## TM0とCR00の一致でクリア&スタート
        st.b    r10,     TMC0[r0]               -- タイマ・モード・コントロール・レジスタ0の設定
        mov     r0,      r10                    -- 表示データ初期化
        ei
fv_loop:
        br      fv_loop                         -- 永久ループ

_int_tm00:
        mov     r10,     r11                    -- r11=r10
        mov     r10,     r12                    -- r12=r10
        and     0x0f,    r11                    -- 表示データの下位4ビットを
        shl     4,       r11                    -- ポート3の上位4ビットに出力
        st.b    r11,     P3[r0]                 -- LED点灯
        and     0xf0,    r12                    -- 表示データの上位4ビットを
        shr     4,       r12                    -- ポート11の下位4ビットに出力
        st.b    r12,     P11[r0]                -- LED点灯
        add     1,       r10                    -- r10を1つインクリメント
        cmp     0x100,   r10                    -- r10が0x100になったら
        bnz     _int_tm00_not0                  --
        mov     r0,      r10                    -- r10を0に戻す
_int_tm00_not0:
        reti                                    -- 割り込みルーチンから復帰
```

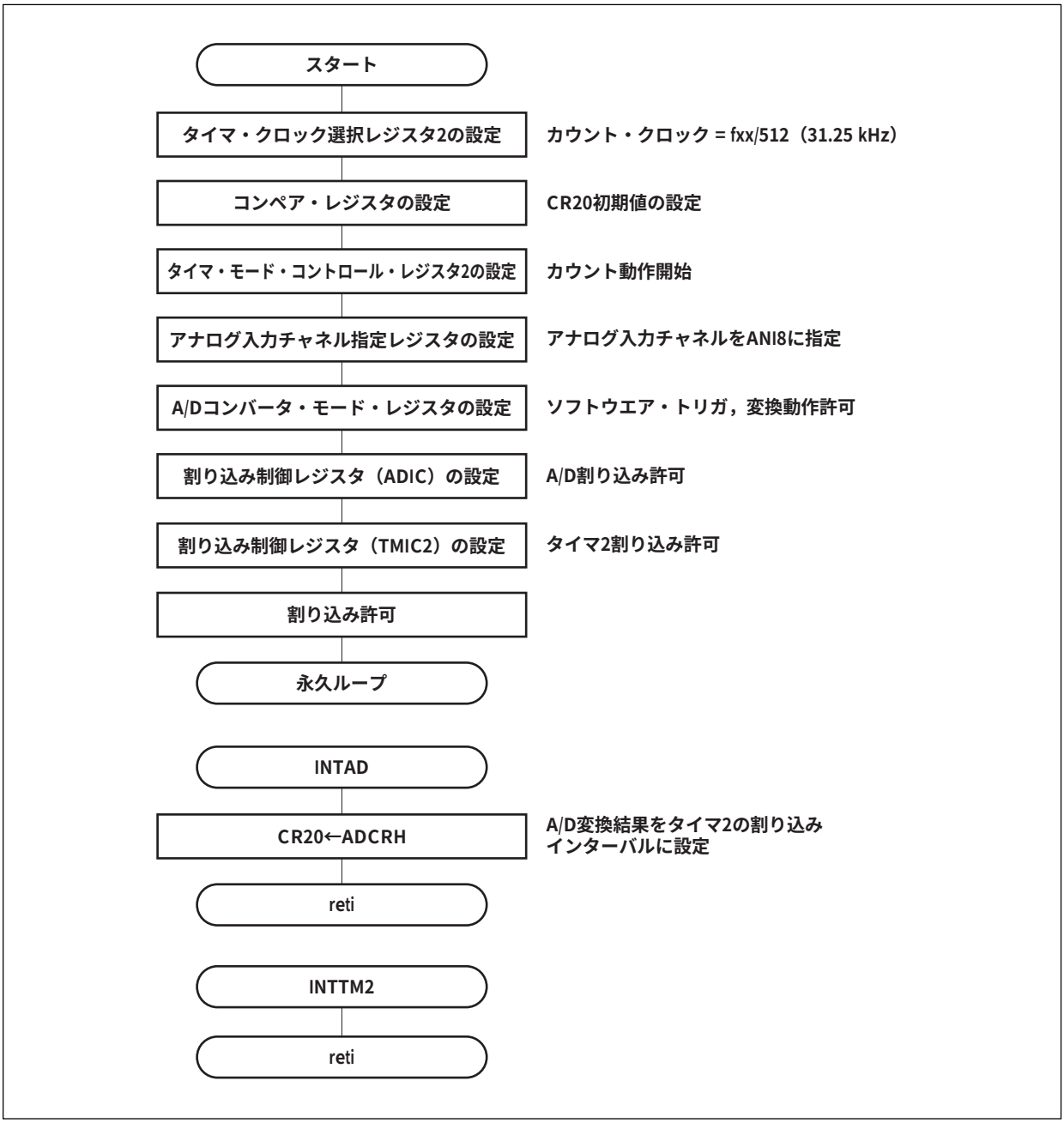
(3) Cプログラム

```
#pragma ioreg                                   /* 周辺I/0レジスタ名を使用可能にする */

/* ------------------------------------------------------
 *  グローバル変数
 *------------------------------------------------------
 */
int dsp_data ;                                  /* LED表示データ */

/* ------------------------------------------------------
 *  メイン処理
 *------------------------------------------------------
 */
void main(void)
{
    PRM0 = 0x01 ;                               /* カウント・クロック:=fxx/16(1MHz:fxx=16MHz) */
    /* コンペア・レジスタの設定 */
    CR00 = 0x2710 ;                             /* 1MHz*0x2710=0.01sec */
    TMIC00 = 0x07 ;                             /* INTTM00 割り込み許可 */
    /* タイマ・モード・コントロール・レジスタの設定 */
    TMC0 = 0x0c ;                               /* TM0とCR00の一致でクリア&スタート */
    dsp_data = 0 ;                              /* 表示データ初期化 */
    __EI();                                     /* 割り込み許可 */
    /* 永久ループ */
    for ( ; ; ) {
    }
}

/* ------------------------------------------------------
 *  タイマ4割り込み関数
 *------------------------------------------------------
 */
#pragma interrupt INTTM00 _int_tm00
void _int_tm00(void)
{
    /* dataの下位4ビットを4ビット左シフトしポート3に出力 */
    P3 = ( dsp_data & 0x0f ) << 4 ;
    /* dataの上位4ビットを4ビット右シフトしポート11に出力 */
    P11 = ( dsp_data & 0xf0 ) >> 4 ;

    /* dsp_dataが0xfe以下ならインクリメントし他なら0に戻す */
    if ( dsp_data < 0xfe ) {
        dsp_data++ ;
    }else{
        dsp_data = 0 ;
    }
}
```

4.4.4　アナログ入力のサンプリングとDCモータの速度制御

A/D機能により，可変抵抗の値をリードします。リードしたデータをタイマ2のコンペア・レジスタに設定します。タイマ2はPWM出力として動作させているので，モータの速度制御になります。

(1) フロー・チャート



(2) アセンブラ・プログラム

```
.globl  start

## 割り込みハンドラのセット
.section "INTAD"                                -- INTAD割り込みハンドラ
        jr      _int_ad                         -- 割り込み処理へ
.section "INTTM2"                               -- INTTM2割り込みハンドラ
        jr      _int_tm2                        -- 割り込み処理へ

.text
        .align 4
start:
        mov     0x07,           r10
        ## カウント・クロック:=fxx/512(31.25kHz)
        st.b    r10,            TCL2[r0]        -- タイマ・クロック選択レジスタ2の設定
        movea   0xff,           r0, r10
        st.b    r10,            CR20[r0]        -- コンペア・レジスタの設定(初期値)
        movea   0xC3,           r0, r10
        ## カウント動作開始,タイマ出力許可,PWMモード
        st.b    r10,            TMC2[r0]        -- タイマ・モード・コントロール・レジスタ2の設定
        mov     0x08,           r10
        st.b    r10,            ADS[r0]         -- アナログ入力チャネルをANI8に指定
        movea   0x88,           r0, r10
        st.b    r10,            ADM[r0]         -- ソフトウエア・トリガ,変換動作許可
        mov     0x07,           r10
        st.b    r10,            ADIC[r0]        -- A/D割り込み許可
        mov     0x07,           r10
        st.b    r10,            TMIC2[r0]       -- タイマ2割り込み許可
        ei
fv_loop:
        br      fv_loop                         -- 永久ループ

_int_ad:                                        -- A/D完了割り込み処理
        ld.b    ADCRH[r0],      r10             -- アナログ入力値をリードし
        st.b    r10,            CR20[r0]        -- タイマ2の割り込みインターバルに設定
        reti                                    -- 割り込みルーチンから復帰

_int_tm2:                                       -- タイマ2割り込み処理
        reti                                    -- 割り込みルーチンから復帰
```

(3) Cプログラム

```
#pragma ioreg                                   /* 周辺I/0レジスタ名を使用可能にする */

/* ------------------------------------------------------
 *  メイン処理
 *------------------------------------------------------
 */
void main(void)
{
    /* タイマ･クロック選択レジスタの設定 */
    TCL2 = 0x07 ;                               /* カウント・クロック:=fxx/512(31.25kHz) */
    CR20 = 0xff ;                               /* コンペア・レジスタの設定(初期値) */
    /* タイマ・モード・コントロール・レジスタの設定 */
    TMC2 = 0xc3 ;                               /* カウント動作開始,タイマ出力許可,PWMモード */
    ADS  = 0x08 ;                               /* アナログ入力チャネルをANI8に指定 */
    ADM  = 0x88 ;                               /* ソフトウエア・トリガ,変換動作許可 */
    ADIC = 0x07 ;                               /* A/D割り込み許可 */
    TMIC2 = 0x07 ;                              /* タイマ2割り込み許可 */
    __EI();                                     /* 割り込み許可 */
    /* 永久ループ */
    for ( ; ; ) {
    }
}

/* ------------------------------------------------------
 *  A/D完了割り込み関数
 *------------------------------------------------------
 */
#pragma interrupt INTAD _int_ad
void _int_ad(void)
{
    /* アナログ入力値をリードしタイマ2の割り込みインターバルに設定 */
    CR20 = ADCRH ;
}
/* ------------------------------------------------------
 *  タイマ2割り込み処理
 *------------------------------------------------------
 */
#pragma interrupt INTTM2 _int_tm2
void _int_tm2(void)
{
}
```

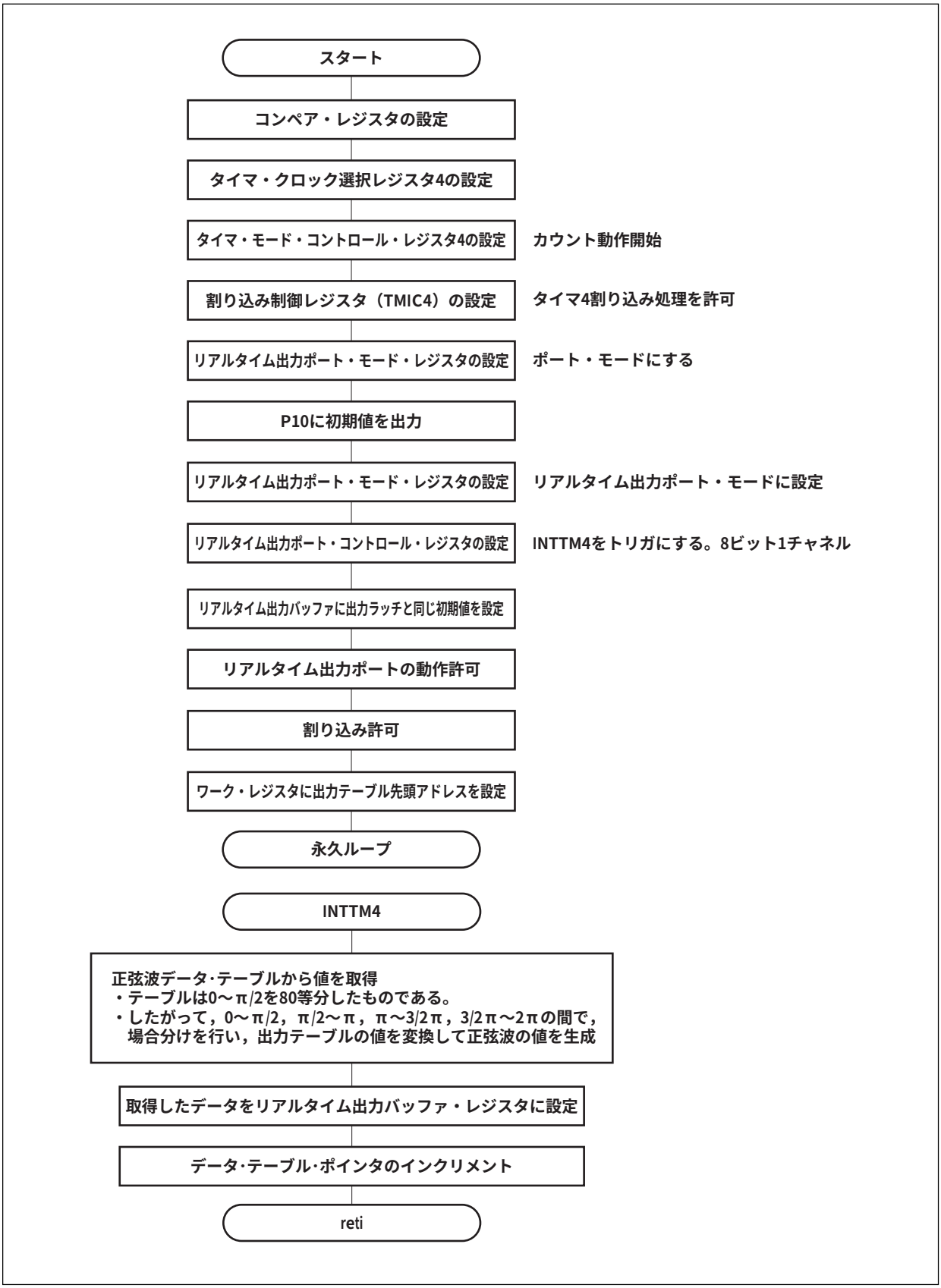
4.4.5　RTPを使用したD/Aによる正弦波出力

リアルタイム出力ポートの出力トリガにはINTTM4を指定します。

正弦波データは，アセンブリ言語ではデータ・テーブルとしてあらかじめ用意します。C言語ではsinf()関数を使用してテーブルを作成します。

正弦波の振動数は，アセンブリ言語では520 Hz，C言語では200 Hzとします。リアルタイム出力ポートから出力されたデータはD/A機能により，アナログ値となりスピーカから音として出力されます。

(1) フロー・チャート（アセンブリ言語）

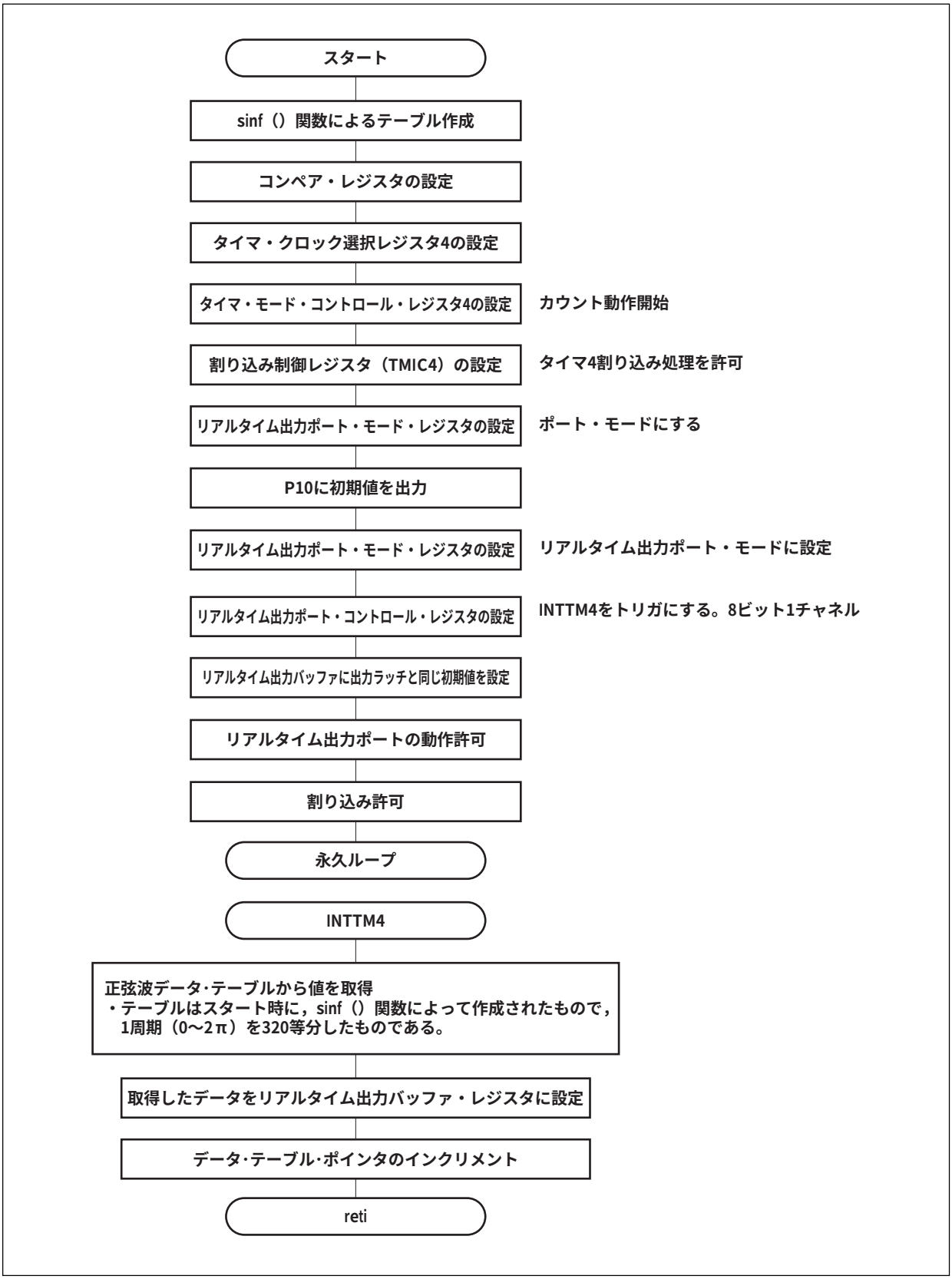


(2) アセンブラ・プログラム

```
# ＜使用レジスタ＞
#   r6                  ：r1退避用レジスタ
#   r10                 ：メイン処理作業用
#   r11                 ：リアルタイム出力値
#   r12                 ：_sin_tbアドレス
#   r13                 ：Δx [Δxの範囲は0から319]
#   r21,r22,r23         ：タイマ4割り込み処理作業用

.globl  start

## 割り込みハンドラのセット
.section "INTTM4"                           -- INTTM4割り込みハンドラ
        jr      _int_tm4                    -- 割り込み処理へ

.text
        .align 4
start:
        ## タイマ4の設定
        ## 4MHz/(24*320)=520Hzの音を出力
        movea   24,         r0, r10
        st.b    r10,        CR40[r0]        -- コンペア･レジスタの設定
        mov     2,          r10
        ## fxx/4 ( 4MHzを選択 )
        st.b    r10,        TCL4[r0]        -- タイマ･クロック選択レジスタ4の設定
        movea   0x80,       r0, r10
        st.b    r10,        TMC4[r0]        -- カウント動作開始
        mov     0x07,       r10
        st.b    r10,        TMIC4[r0]       -- タイマ4割り込み処理を許可
        ## リアルタイム出力の設定
        st.b    r0,         RTPM[r0]        -- ポート・モードにする
        st.b    r0,         P10[r0]         -- 初期値を出力ラッチに設定する
        movea   0xff,       r0, r10
        st.b    r10,        RTPM[r0]        -- リアルタイム出力ポート・モードに設定
        movea   0x20,       r0, r10
        st.b    r10,        RTPC[r0]        -- INTTM4をトリガとする
                                            -- 8ビット１チャネル
        st.b    r0,         RTBL[r0]        -- 出力ラッチと同じ初期値をリアルタイム
                                            -- 出力バッファ・レジスタに設定する
        set1    7,          RTPC[r0]        -- リアルタイム出力ポートの動作許可
        ei                                  -- 割り込みの許可
init_reg:
        movhi   hi1(#_sin_tb), r0, r12      -- r12:=_sin_tbアドレス
        movea   lo(#_sin_tb), r12, r12
        mov     r0,         r13

fv_loop:
        br      fv_loop                     -- 永久ループ

_int_tm4:
        mov     r1,          r6             -- r1の退避
tm4_chk_dx:
        movea   0x50,  r0,  r21             -- r21:=0x50(80)
        cmp     r21,        r13             -- もし r13 >= 80 のとき
        bp      tm4_chk_u160                -- ジャンプ
tm4_u80:
        mov     r12,        r21             -- r21:=_sin_tbアドレス
        add     r13,        r21             -- r21:=_sin_tbアドレス+Δx
        ld.b    0[r21],     r11             -- r11:=リアルタイム出力値
        br      tm4_out
tm4_chk_u160:
        movea   0xA0,  r0,  r21             -- r21:=0xA0(160)
        cmp     r21,        r13             -- もし r13 >= 160 のとき
        bp      tm4_chk_u240                -- ジャンプ
tm4_u160:
        mov     r12,        r21             -- r21:=_sin_tbアドレス
        movea   0x9F,  r0,  r22             -- r22:=0x9F(159)
```

```
        add     r22,        r21                 -- r21=r21+159
        sub     r13,        r21                 -- r21=r21-Δx
        ld.b    0[r21],     r11                 -- r11:=リアルタイム出力値
        br      tm4_out
tm4_chk_u240:
        movea   0xF0,  r0,  r21                 -- r21:=0xF0(240)
        cmp     r21,        r13                 -- もし r13 >= 240 のとき
        bp      tm4_u320                        -- ジャンプ
tm4_u240:
        mov     r12,        r21                 -- r21:=_sin_tbアドレス
        add     r13,        r21                 -- r21=r21+Δx
        movea   0xA0,  r0,  r22                 -- r22:=0xA0(160)
        sub     r22,        r21                 -- r21=r21-160
        ld.b    0[r21],     r22                 -- メモリからロード
        and     0xff,       r22
        movea   0xFF,  r0,  r11                 -- リアルタイム出力値
        sub     r22,        r11                 -- 0x80からテーブル値を減算する
        br      tm4_out
tm4_u320:
        mov     r12,        r21                 -- r21:=_sin_tbアドレス
        movea   0x13F, r0,  r22                 -- r22=0x13F(320)
        add     r22,        r21                 -- r21=r21+319
        sub     r13,        r21                 -- r21=r21-Δx
        ld.b    0[r21],     r22                 -- メモリからロード
        and     0xff,       r22
        movea   0xFF,  r0,  r11                 -- リアルタイム出力値
        sub     r22,        r11                 -- 0x80からテーブル値を減算する
        br      tm4_out
tm4_out:
        st.b    r11,        RTBH[r0]            -- リアルタイム出力バッファ・レジスタに出力値を設定
tm4_inc_dx:
        add     1,          r13                 -- Δxインクリメント
        movea   0x140, r0,  r21                 -- r21:=0x140(320)
        cmp     r13,        r21                 -- Δxが320になっていなければ
        bnz     tm4_exit                        -- ジャンプ
tm4_dx_eq0:
        mov     r0,         r13                 -- Δxを0に戻す
tm4_exit:
        mov     r6,         r1                  -- r1の復帰
        reti

_sin_tb:                                        -- 正弦波テーブル
        ## sin(x)*0x80  x = 0 to π/2 [ step=(π/2)/80 ]
        .byte   128,130,133,135,138,140,143,145
        .byte   148,150,152,155,157,160,162,165
        .byte   167,169,172,174,176,179,181,183
        .byte   186,188,190,192,194,197,199,201
        .byte   203,205,207,209,211,213,214,216
        .byte   218,220,221,223,225,226,228,230
        .byte   231,233,234,235,237,238,239,240
        .byte   242,243,244,245,246,247,248,248
        .byte   249,250,251,251,252,253,253,254
        .byte   254,254,255,255,255,255,255,255
```

(3) フロー・チャート（C言語）



(4) Cプログラム

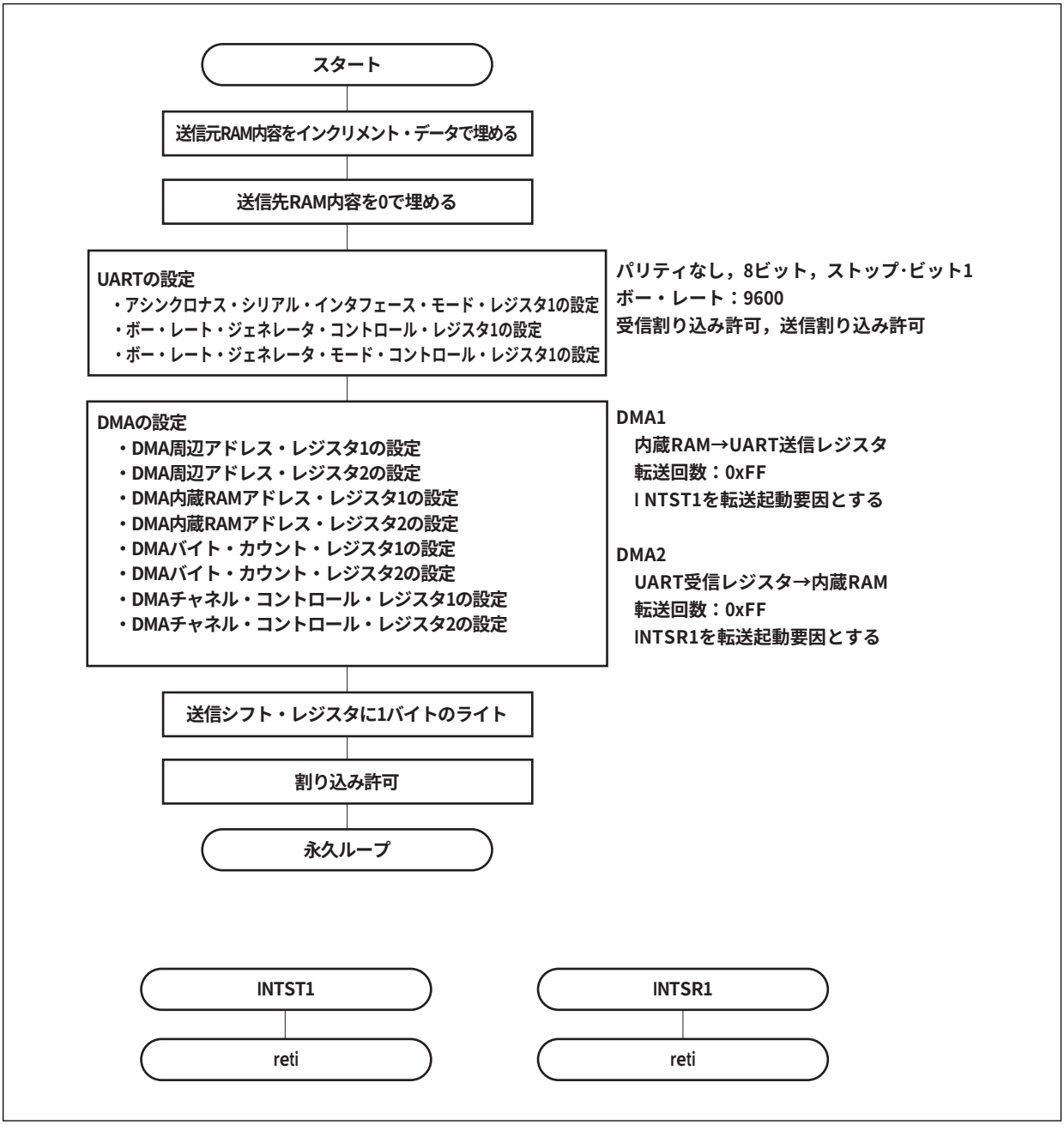
```
#include <math.h>
#include <float.h>
#pragma ioreg                                   /* 周辺I/0レジスタ名を使用可能にする */

int st  ;                                       /* ステップ間隔  */
int sin_buf[320] ;                              /* 出力値配列 */

/* ------------------------------------------------------
 *  グローバル変数初期化処理
 *------------------------------------------------------
 */
void sta_init()
{
    double pai ;                                /* π        */
    int data ;
    int st0  ;

    /* 変数st初期化 */
    st = 0 ;

    /* 変数sin_buf[]初期化 */
    pai = 3.14159 ;
    for ( st0 = 0 ; st0 < 320 ; st0++ ) {
        /* 320個の正弦波データをテーブル化する */
        data = ( int ) ( sinf( ( pai * 2 * st0 ) / 320 )  * 0x80 ) ;
        data += 0x80;
        if ( data == 0x80 *2 ) {
            data = 0xff ;
        }
        sin_buf[st0] = data ;                   /* テーブルにデータをセット */
    }
}


/* ------------------------------------------------------
 *  メイン処理
 *------------------------------------------------------
 */
void main(void)
{
    sta_init();                                 /* static変数の初期化 */
    /* タイマ4の設定
        4MHz/(62*320)=200Hzの音を出力 */
    CR40 = 62 ;                                 /* コンペア･レジスタの設定 */
    /* fxx/4 ( 4MHzを選択 ) */
    TCL4 = 2 ;                                  /* タイマ･クロック選択レジスタ4の設定 */
    TMC4 = 0x80 ;                               /* カウント動作開始 */
    TMIC4 = 0x07 ;                              /* タイマ4割り込み処理を許可 */
    /* リアルタイム出力の設定 */
    RTPM = 0 ;                                  /* ポート・モードにする */
    P10 = 0 ;                                   /* 初期値を出力ラッチに設定する */
    RTPM = 0xff ;                               /* リアルタイム出力ポート・モードに設定 */
    RTPC = 0x20 ;                               /* INTTM4をトリガとする */
                                                /* 8ビット１チャネル */
    RTBL = 0 ;                                  /* 出力ラッチと同じ初期値をリアルタイム */
                                                /* 出力バッファ・レジスタに設定する */
    RTPC.7 = 1 ;                                /* リアルタイム出力ポートの動作許可 */
    __EI() ;                                    /* 割り込みの許可 */

    /* 永久ループ */
    for ( ; ; ) {
    }
}
```

```
/* ------------------------------------------------------
 *  タイマ4割り込み関数
 *------------------------------------------------------
 */
#pragma interrupt INTTM4 _int_tm4
void _int_tm4(void)
{
    RTBH = sin_buf[st] ;                    /* リアルタイム出力バッファ・レジスタに出力値を設定 */
    st++ ;                                  /* ステップのインクリメント */
    if ( st == 320 ) {                      /* ステップが320になったら */
        st = 0 ;                            /* 0に戻す                 */
    }
}
```

4.4.6　DMA転送（UART←→内部RAM）プログラム1

DMAを2チャネル使用します。1チャネルは，INTST1を転送起動要因として，内蔵RAMからUART送信レジスタへの転送を行います。もう1チャネルは，INTSR1を転送起動要因として，UART受信レジスタから内蔵RAMへの転送を行います。なお，UARTは送信データがそのまま受信できるように折り返しの設定で動作させます。

(1) フロー・チャート



(2) アセンブラ・プログラム

```
.globl  start

## 割り込みハンドラのセット
.section "INTST1"                           -- INTST1割り込みハンドラ
        jr      _int_st1                    -- 割り込み処理へ
.section "INTSR1"                           -- INTSR1割り込みハンドラ
        jr      _int_sr1                    -- 割り込み処理へ

.text
        .align 4
start:
        ## 送信元RAM内容をインクリメント・データで埋める
        movea   0xe000      r0, r11         -- 0xffffe000:=UART送信データ格納アドレス
        mov     0,          r12             -- 送信データ初期値
ram_inc:
        st.b    r12,        0[r11]          -- 送信データの格納
        add     1,          r11             -- 送信データ格納アドレスのインクリメント
        add     1,          r12             -- 送信データのインクリメント
        cmp     0x100,      r12             -- r12が0x100になるまで
        bnz     ram_inc                     -- ループする
        ## 送信先ram内容を0で埋める
        movea   0xe100,     r0, r11         -- 0xffffe100:=UART受信データ格納アドレス
        mov     r0,         r12             -- r12:=ループ・カウンタ
ram_0:
        st.b    r0,         0[r11]          -- 受信先アドレスに0を設定
        add     1,          r11             -- 受信データ格納アドレスのインクリメント
        add     1,          r12             -- ループ・カウンタのインクリメント
        cmp     0x100,      r12             -- r12が0x100になるまで
        bnz     ram_0                       -- ループする

        ## UART設定
        movea   0xc8,       r0, r10
        st.b    r10,        ASIM1[r0]       -- パリティなし，8ビット，ストップ・ビット1
        movea   0xd0,       r0, r10
        st.b    r10,        BRGC1[r0]       -- ボー・レート:=9600
        mov     0x03,       r10
        st.b    r10,        BRGMC1[r0]      -- ボー・レート:=9600
        mov     0x07,       r10
        st.b    r10,        SRIC1           -- 受信割り込み許可
        st.b    r10,        STIC1           -- 送信割り込み許可
        ## DMA設定
        movea   0x0316,     r0, r10
        st.h    r10,        DIOA1[r0]       -- 送信レジスタのアドレス
        movea   0x0318,     r0, r10
        st.h    r10,        DIOA2[r0]       -- 受信レジスタのアドレス
        mov     0x0000,     r10
        st.h    r10,        DRA1[r0]        -- 内蔵RAM（送信元）アドレス
        movea   0x0100,     r0, r10
        st.h    r10,        DRA2[r0]        -- 内蔵RAM（送信先）アドレス
        movea   0xff,       r0, r10
        st.b    r10,        DBC1[r0]        -- DMAバイト・カウント・レジスタ1の設定
        movea   0xff,       r0, r10
        st.b    r10,        DBC2[r0]        -- DMAバイト・カウント・レジスタ2の設定
        mov     0x09,       r10
        st.b    r10,        DCHC1[r0]       -- DMA1の転送起動要因:=INTST1,転送許可
        mov     0x05,       r10
        st.b    r10,        DCHC2[r0]       -- DMA2の転送起動要因:=INTSR1,転送許可
        ## 送信シフト・レジスタに1バイトのライト
        st.b    r0,         TXS1[r0]
        ei
fv_loop:
        br      fv_loop                     -- 永久ループ

_int_st1:                                   -- 送信完了割り込み処理
        reti                                -- 割り込みルーチンから復帰
_int_sr1:                                   -- 受信完了割り込み処理
        reti                                -- 割り込みルーチンから復帰
```

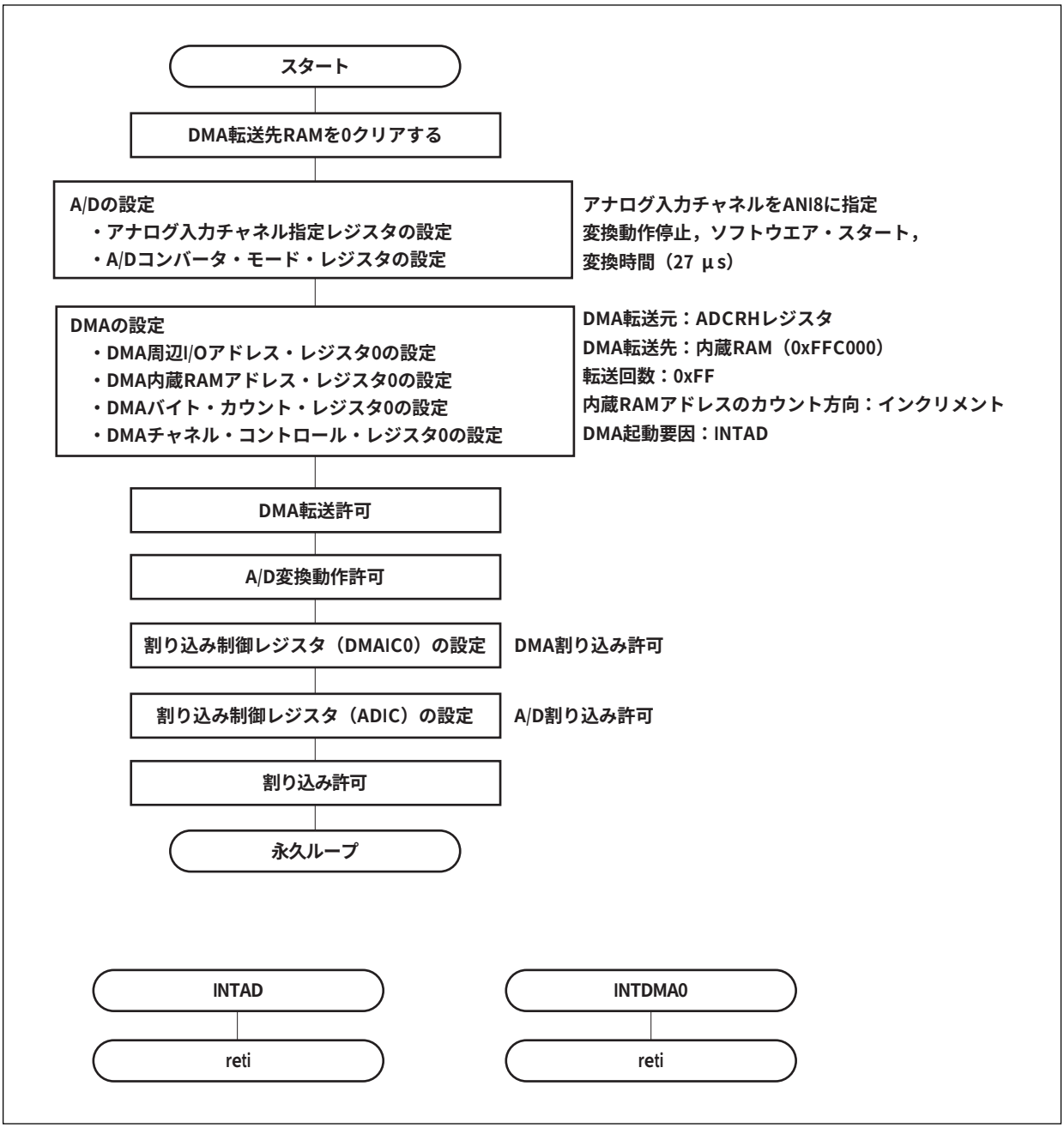
(3) Cプログラム

```
#pragma ioreg                                   /* 周辺I/0レジスタ名を使用可能にする */

/* ------------------------------------------------------
 *  メイン処理
 *------------------------------------------------------
 */
void main(void)
{
    char *p ;
    int   i ;

    /* 送信元RAM内容をインクリメント･データで埋める，p = UART送信データ格納アドレス */
    for ( p = (char *)0xffffe000, i = 0  ; p < (char *)0xffffe100 ; p++, i++ ) {
        *p = i ;                                /* 送信データの格納 */
    }

    /* 送信先RAM内容を0で埋める，p = UART受信データ格納アドレス */
    for ( p = (char *)0xffffe100 ; p < (char *)0xffffe200 ; p++ ) {
        *p = 0 ;                                /* 受信先アドレスに0を設定 */
    }

    /* UART設定 */
    ASIM1  = 0xc8 ;                             /* パリティなし,8ビット,ストップ･ビット1 */
    BRGC1  = 0xd0 ;                             /* ボー・レート:=9600 */
    BRGMC1 = 0x03 ;                             /* ボー・レート:=9600 */
    SRIC1  = 0x07 ;                             /* 受信割り込み許可 */
    STIC1  = 0x07 ;                             /* 送信割り込み許可 */

    /* DMA設定 */
    DIOA1 = 0x0316 ;                            /* 送信レジスタのアドレス */
    DIOA2 = 0x0318 ;                            /* 受信レジスタのアドレス */
    DRA1  = 0x0000 ;                            /* 内蔵RAM（送信元）アドレス */
    DRA2  = 0x0100 ;                            /* 内蔵RAM（送信先）アドレス */
    DBC1  = 0xff ;                              /* DMAバイト・カウント・レジスタ1の設定 */
    DBC2  = 0xff ;                              /* DMAバイト・カウント・レジスタ2の設定 */
    DCHC1 = 0x09 ;                              /* DMA1の転送起動要因:=INTST1,転送許可 */
    DCHC2 = 0x05 ;                              /* DMA2の転送起動要因:=INTSR1,転送許可 */

    /* 送信シフト・レジスタに1バイトのライト */
    TXS1 = 0 ;

    __EI();                                     /* 割り込み許可 */
    /* 永久ループ */
    for ( ; ; ) {
    }
}

/* ------------------------------------------------------
 *  送信完了割り込み処理
 *------------------------------------------------------
 */
#pragma interrupt INTST1 _int_st1
void _int_st1(void)
{
}
/* ------------------------------------------------------
 *  受信完了割り込み処理
 *------------------------------------------------------
 */
#pragma interrupt INTSR1 _int_sr1
void _int_sr1(void)
{
}
```

4.4.7　DMA転送（A/D→内部RAM）プログラム2

INTADをDMA転送起動要因として，可変抵抗値を内蔵RAMに格納します。

（1）フロー・チャート



(2) アセンブラ・プログラム

```
.globl  start
.set    ram_addr,      0xffffe000                  -- DMA転送先RAMアドレス
.set    ram_addr_end,  0xffffe100                  -- DMA転送先RAMアドレス終端

## 割り込みハンドラのセット
.section "INTAD"                                   -- INTAD割り込みハンドラ
        jr      _int_ad                            -- 割り込み処理へ
.section "INTDMA0"                                 -- INTDMA0割り込みハンドラ
        jr      _int_dma0                          -- 割り込み処理へ

.text
        .align 4
start:
        ## DMA転送先RAMを0クリアする
        movea   ram_addr & 0x0000ffff, r0, r11     -- r11:=DMA転送先RAMアドレス
ram_init_set:
        st.b    r0,            0[r11]              -- DMA転送先RAMアドレスに0をセット
        add     1,             r11                 -- DMA転送先RAMアドレス・インクリメント
        cmp     ram_addr_end,  r11                 -- もしDMA転送先RAMアドレス終端
        bnz     ram_init_set                       -- でなければループする
        ## A/D設定
        mov     0x08,          r10
        st.b    r10,           ADS[r0]             -- アナログ入力チャネルをANI8に指定
        mov     0x01,          r10
        ## 変換動作停止,ソフトウエア・スタート,変換時間(27us)
        st.b    r10,           ADM[r0]             -- A/Dコンバータ・モード・レジスタの設定
        ## DMA設定
        movea   0x03c6,        r0, r10
        ## DMA転送元をADCRHレジスタに設定
        st.h    r10,           DIOA0[r0]           -- DMA周辺I/Oアドレス・レジスタ0のを設定
        ## DMA転送先を内蔵RAM(番地:0xFFE000)に設定
        st.h    r0,            DRA0[r0]            -- DMA内蔵RAMアドレス・レジスタ0の設定
        movea   0xff,          r0, r10
        st.b    r10,           DBC0[r0]            -- DMAバイト・カウント・レジスタ0の設定
        movea   0x14,          r0, r10
        ## 内蔵RAMアドレスのカウント方向:インクリメント,DMA起動要因:INTAD,転送方向:周辺I/O→内蔵RAM
        ## 8ビット転送,DMA転送禁止
        st.b    r10,           DCHC0[r0]           -- DMAチャネル・コントロール・レジスタ0の設定
        set1    0,             DCHC0[r0]           -- DMA転送許可
        set1    7,             ADM[r0]             -- 変換動作許可
        mov     0x07,          r10
        st.b    r10,           DMAIC0[r0]          -- DMA転送終了割り込み許可
        st.b    r10,           ADIC[r0]            -- A/D完了割り込み許可
        ei
fv_loop:
        br      fv_loop
_int_ad:                                           -- A/D完了割り込み処理
        reti                                       -- 割り込みルーチンから復帰
_int_dma0:                                         -- DMA完了割り込み処理
        reti                                       -- 割り込みルーチンから復帰
```

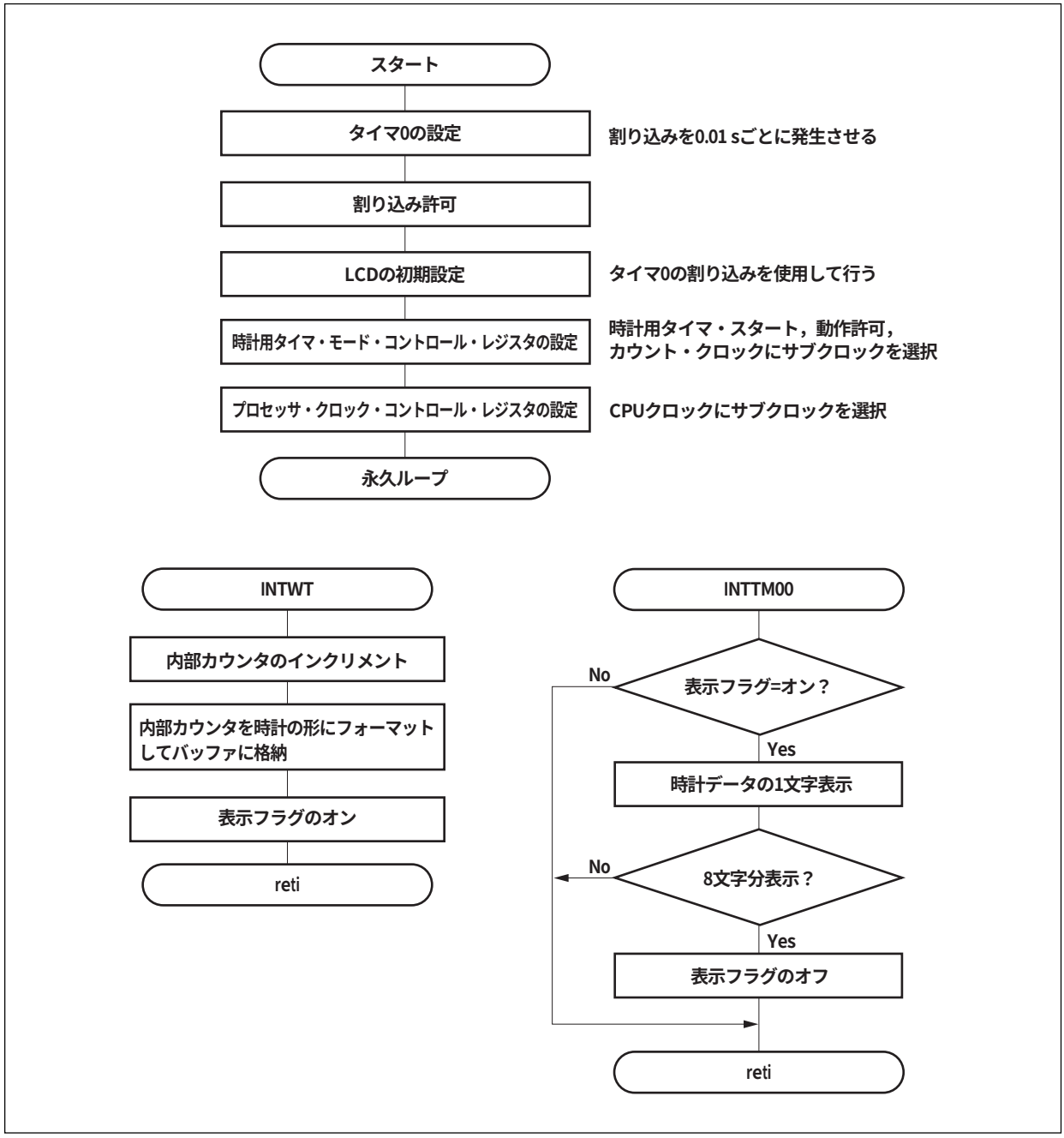
(3) Cプログラム

```
#pragma ioreg                                   /* 周辺I/0レジスタ名を使用可能にする */

/* ------------------------------------------------------
 *  メイン処理
 *------------------------------------------------------
 */
void main(void)
{
    char *p ;
    /* DMA転送先RAMを0クリアする, p = DMA転送先RAMアドレス */
    for ( p = (char *)0xffffe000 ; p < (char *)0xffffe100 ; p++ ) {
        *p = 0 ;
    }
    /* A/D設定 */
    ADS = 0x08 ;                                /* アナログ入力チャネルをANI8に指定 */
    /* A/Dコンバータ・モード・レジスタの設定 */
    ADM = 0x01 ;                                /* 変換動作停止,ソフトウエア・スタート,変換時間(27us) */

    /* DMA設定 */
    /* DMA周辺I/0レジスタのアドレスを設定 */
    DIOA0 = 0x03c6 ;                            /* DMA転送元をADCRHレジスタに設定 */
    /* DMA内蔵RAMアドレス・レジスタの設定 */
    DRA0  = 0 ;                                 /* DMA転送先を内蔵RAM(番地:0xFFE000)に設定 */
    DBC0  = 0xff ;                              /* DMAバイト・カウント・レジスタ0の設定 */
    /* DMAチャネル・コントロール・レジスタの設定 */
    DCHC0 = 0x14 ;                              /* 内蔵RAMアドレスのカウント方向:インクリメント, */
                                                /* DMA起動要因:INTAD,転送方向:周辺I/0→内蔵RAM */
                                                /* 8ビット転送,DMA転送禁止 */
    DCHC0.0 = 1 ;                               /* DMA転送許可 */
    ADM.7   = 1 ;                               /* 変換動作許可 */

    DMAIC0  = 0x07 ;                            /* DMA転送終了割り込み許可 */
    ADIC    = 0x07 ;                            /* A/D完了割り込み許可 */

    __EI();                                     /* 割り込み許可 */
    /* 永久ループ */
    for ( ; ; ) {
    }
}

/* ------------------------------------------------------
 *  A/D完了割り込み処理
 *------------------------------------------------------
 */
#pragma interrupt INTAD _int_ad
void _int_ad(void)
{
}
/* ------------------------------------------------------
 *  DMA0完了割り込み処理
 *------------------------------------------------------
 */
#pragma interrupt INTDMA0 _int_dma0
void _int_dma0(void)
{
}
```

4.4.8　サブクロック動作への切り替えと時計表示プログラム

CPUクロックにサブクロックを選択します。時計用タイマのカウント･クロックにもサブクロックを選択します。時計用タイマ割り込みにより，時計データをインクリメントします。実際は，タイマ0割り込みにより，LCDに表示されます。

(1) フロー・チャート



(2) アセンブラ・プログラム

```
# ＜使用レジスタ＞
#   r6                  ：r1退避用レジスタ
#   r11,r12,r13,r14,r15：メイン処理作業用
#   r16,r17,r18,r19,r20：タイマ0割り込み処理作業用
#   r21,r22,r23,r24,r25：時計用タイマ割り込み処理作業用
#   r26,r27,r28,r29     ：bf_chk ルーチン・ワーク・レジスタ

.globl  start

.set    tm_buf,    0x120000                  -- 時計表示データ(xx:xx:xx)(8バイト)
.set    col_ctr,   0x120008                  -- 時計表示データの表示カラム数(1バイト)
.set    flg_wt,    0x120009                  -- 1秒経過フラグ(1バイト)
.set    wt_num,    0x120010                  -- 時計データ(4バイト)
.set    disp_flg,  0x120014                  -- 時計LCD表示状態フラグ(1で表示)
.set    tm0_num,   0x120018                  -- タイマ0カウンタ(4バイト)
.set    mem_last,  0x12001d                  -- ワーク変数領域最終アドレス
.set    lcd_addr,  0x140000                  -- LCDアドレス

## 割り込みハンドラのセット
.section "INTWT"                             -- INTWT割り込みハンドラ
        jr      _int_wt                      -- 割り込み処理へ
.section "INTTM00"                           -- INTTM00割り込みハンドラ
        jr      _int_tm00                    -- 割り込み処理へ

.text
        .align 4
start:
mem_init:
        movhi   hi1(tm_buf), r0, r11         -- r11:=ワーク･メモリの先頭アドレス
        movea   lo(tm_buf), r11, r11
        movhi   hi1(mem_last), r0, r12       -- r12:=ワーク変数領域最終アドレス
        movea   lo(mem_last), r12, r12
mem_init_lp:
        st.b    r0,           0[r11]         -- ワーク変数の0セット
        addi    1,    r11,    r11            -- 0クリアするアドレスのインクリメント
        cmp     r11,          r12            -- 0クリアするアドレスが最終アドレスでなければ
        bnz     mem_init_lp                  -- ループする
ins_3a:
        movhi   hi1(tm_buf), r0, r11         -- r11:=ワーク･メモリの先頭アドレス
        movea   lo(tm_buf), r11, r11
        movea   0x3a,   r0,   r13            -- 時間データに
        st.b    r13,          2[r11]         -- ':'=0x3Aをセットする
        st.b    r13,          5[r11]         -- ':'=0x3Aをセットする

init_tm0:
        movea   0x01,   r0,   r11            -- r11:=1
        -- fxx/16(fxx=16MHz):=カウント･クロックを1MHzに指定
        st.b    r11,          PRM0[r0]       -- プリスケーラ・モード・レジスタ0の設定
        st.b    r0,           TOC0[r0]       -- タイマ出力コントロール・レジスタ0の設定
        movea   0x64, r0,   r11              -- r11:1MHz*0x64=0.1ms
        -- 割り込み間隔を0.1msに設定
        st.h    r11,          CR00[r0]
        -- CR00をコンペア・レジスタとして動作
        st.b    r0,           CRC0[r0]       -- キャプチャ/コンペア・コントロール・レジスタ0の設定
        mov     0x0c,         r11
        -- TM0とCR00の一致でクリア&スタート
        st.b    r11,          TMC0[r0]       -- タイマ・モード・コントロール・レジスタ0の設定
        mov     0x07,         r11
        st.b    r11,          TMIC00[r0]     -- タイマ割り込み許可(レベル7)
        ei
init_lcd:
        movhi   hi1(lcd_addr), r0, r11       -- r11:=LCD アドレス
        movea   lo(lcd_addr), r11, r11
        movhi   hi1(tm0_num), r0, r12        -- r12:=タイマ0カウンタ・アドレス
        movea   lo(tm0_num), r12, r12
        movea   0x30,   r0,   r13            -- r13:=0x30
```

```
        st.b    r13,          0[r11]            -- ファンクション・セット 8ビット 1行 5*7
        st.w    r0,           0[r12]            -- タイマ0カウンタ=0
        movea   0x29,   r0,   r13               -- r13:=0x29=41
wait_1:
        ld.w    0[r12],       r14               -- r14:=タイマ0カウンタ
        cmp     r14,          r13               -- タイマ0カウンタが41に達したかのチェック
        bp      wait_1                          -- 4.1ms以上待つ
        st.b    r13,          0[r11]            -- ファンクション・セット 8ビット 1行 5*7
        st.w    r0,           0[r12]            -- タイマ0カウンタ=0
        movea   0x2,    r0,   r13               -- r13:=0x2
wait_2:
        ld.w    0[r12],       r14               -- r14:=タイマ0カウンタ
        cmp     r14,          r13               -- タイマ0カウンタが2に達したかのチェック
        bp      wait_2                          -- 100us以上待つ
        st.b    r13,          0[r11]            -- ファンクション・セット 8ビット 1行 5*7
        jarl    bf_chk,       r29               -- LCDビジィ・チェック
        movea   0x38,   r0,   r13
        st.b    r13,          0[r11]            -- ファンクション・セット 8ビット 2行 5*7
        jarl    bf_chk,       r29               -- LCDビジィ・チェック
        mov     0x01,         r13
        st.b    r13,          0[r11]            -- 表示クリア,カーソルをホーム位置へ
        jarl    bf_chk,       r29               -- LCDビジィ・チェック
        mov     0x06,         r13
        st.b    r13,          0[r11]            -- カーソルはインクリメント，表示シフトしない
        jarl    bf_chk,       r29               -- LCDビジィ・チェック
        mov     0x0C,         r13
        st.b    r13,          0[r11]            -- 表示をON，カーソルをOFF

init_wt:
        ## 時計用タイマ・モード・コントロール・レジスタの設定
        movea   0x83,   r0,   r11
        st.b    r11,          WTM[r0]           -- 時計用タイマ・スタート,動作許可
                                                -- カウント・クロックをサブクロックに選択
        movea   0x06,   r0,   r11               -- r10:=0x06
        ## 時計割り込みのレベルはタイマ0割り込みレベルより，
        ## 高いものに設定する
        st.b    r11,          WTIC[r0]          -- 時計用タイマ割り込み許可(レベル6)
set_sub_clock:
        ## PCCレジスタは特定レジスタなので,特定のシーケンスにより変更する
        movea   0x04,   r0,   r13
        stsr    5,            r11               -- プログラム・ステータス・ワードのロード
        mov     r11,          r12               -- r12=r11
        or      0x80,         r11               -- PSWのNPビット=1
        ldsr    r11,          5                 -- NPビット=1にしたPSWを書き込む
        st.b    r0,           PRCMD[r0]         -- PRCMDへの書き込み
        ## CPUクロックをサブクロックに選択
        st.b    r13,          PCC[r0]           -- プロセッサ・クロック・コントロール・レジスタへの設定
        ldsr    r12,          5                 -- PSWを元に戻す
        nop                                     -- ダミー命令(5命令)
        nop
        nop
        nop
        nop
fv_loop:
        br      fv_loop                         -- 永久ループ

## LCDビジィ・チェック・ルーチン
bf_chk:
        movhi   hi1(lcd_addr), r0, r27          -- r27:=LCD アドレス
        movea   lo(lcd_addr), r27, r27
        movea   0x80,   r0,   r26               -- r26:=0x80
bf_chk0:
        ld.b    0[r27],       r28
        andi    0x80,   r28,  r28
        cmp     r26,          r28
        bz      bf_chk0
        jmp     [r29]

_int_wt:
```

```
        mov     r1,             r6              -- r1の退避
wt_chk:
        movhi   hi1(flg_wt), r0, r21            -- r21:=1秒経過フラグ
        movea   lo(flg_wt), r21, r21
        not1    0,              0[r21]          -- flg_wtを反転
        tst1    0,              0[r21]          -- flg_wtが1だったら
        bnz     wt_exit                         -- 何もせずスキップする
wt_data_set:
        movhi   hi1(wt_num), r0, r21            -- r21:=時計データ
        movea   lo(wt_num), r21, r21
        ld.w    0[r21],         r22             -- wt_numレジスタにロード
        addi    1,      r22,    r22             -- wt_numインクリメント
        st.w    r22,            0[r21]          -- wt_numメモリにストア
        ## r22 を時計の形にフォーマットしてメモリに格納
        movhi   hi1(tm_buf), r0, r21            -- r21:=時計表示データ
        movea   lo(tm_buf), r21, r21
        mov     r22,            r7              -- r7=r22
        movea   60*60*10, r0, r6
        andi    0xffff,   r6, r6                -- r6=60*60*10
        jarl    ___ div, lp                     -- r6=r7/(60*60*10)
        add     0x30,           r6              -- r6=0x30+r6
        st.b    r6,             0[r21]          -- x*10時間
        mov     r22,            r23             -- r23=r22
        movea   60*60,  r0,     r6              -- r6=60*60
        divh    r6,             r23             -- r23=r23/(60*60)
        mov     r23,            r24             -- r24=r23
        mov     10,             r7              -- r7=r10
        divh    r7,             r24             -- r24=r24/10
        mulh    10,             r24             -- r24=r24*10
        sub     r24,            r23             -- r23=r23-r24
        add     0x30,           r23             -- r23=0x30+r23
        st.b    r23,            1[r21]          -- x時間
        mov     r22,            r23             -- r23=r22
        divh    r6,             r23             -- r23=r23/(60*60)
        mulh    60*60,          r23             -- r23=r23*(60*60)
        mov     r22,            r24             -- r24=r22
        sub     r23,            r24             -- r24=r24-r23
        movea   60,     r0,     r6              -- r6=60
        divh    r6,             r24             -- r24=r24/60
        mov     r24,            r23             -- r23=r24
        divh    r7,             r23             -- r23=r23/10
        add     0x30,           r23             -- r23=0x30+r23
        st.b    r23,            3[r21]          -- x*10分
        mov     r24,            r23             -- r23=r24
        divh    r7,             r24             -- r24=r24/10
        mulh    10,             r24             -- r24=r24*10
        sub     r24,            r23             -- r23=r23-r24
        add     0x30,           r23             -- r23=0x30+r23
        st.b    r23,            4[r21]          -- x分
        mov     r22,            r23             -- r23=r22
        divh    r6,             r23             -- r23=r23/60
        mulh    60,             r23             -- r23=r23*60
        mov     r22,            r24             -- r24=r22
        sub     r23,            r24             -- r24=r24-r23
        mov     r24,            r23             -- r23=r24
        divh    r7,             r23             -- r23=r23/10
        add     0x30,           r23             -- r23=0x30+r23
        st.b    r23,            6[r21]          -- x*10秒
        mov     r24,            r23             -- r23=r24
        divh    r7,             r24             -- r24=r24/10
        mulh    10,             r24             -- r24=r24*10
        sub     r24,            r23             -- r23=r23-r24
        add     0x30,           r23             -- r23=0x30+r23
        st.b    r23,            7[r21]          -- x秒
        movhi   hi1(disp_flg), r0, r21          -- r21:=時計LCD表示状態フラグ
        movea   lo(disp_flg), r21, r21
        set1    0,              0[r21]          -- disp_flgのオン
wt_exit:
        mov     r6,             r1              -- r1の復帰
```

```
        reti

_int_tm00:
        mov     r1,             r6              -- r1の退避
tm_countup:
        movhi   hi1(tm0_num), r0, r16           -- r16:=タイマ0カウンタ
        movea   lo(tm0_num), r16, r16
        ld.w    0[r16],         r17             -- tm0_numをレジスタにロード
        addi    1,      r17,    r17             -- tm0_numインクリメント
        st.w    r17,            0[r16]          -- tm0_numをメモリにストア
tm_chk_flg:
        movhi   hi1(disp_flg), r0, r16          -- r16:=時計LCD表示状態フラグ
        movea   lo(disp_flg), r16, r16
        tst1    0,              0[r16]          -- disp_flgのチェック
        bz      tm_exit                         -- disp_flgが0だったらジャンプ
tm_disp_lcd:
        movhi   hi1(col_ctr), r0, r16           -- r16:=時計表示データの表示カラム数
        movea   lo(col_ctr), r16, r16
        ld.b    0[r16],         r17             -- col_ctrをレジスタにロード
        movea   0x07,   r0,     r18             -- r18:=7
        cmp     r17,            r18             -- col_ctrが7以上だったら
        bn      tm_cursol_home                  -- ジャンプ

        movhi   hi1(tm_buf), r0, r18            -- r17:=時計表示データ
        movea   lo(tm_buf), r18, r18
        add     r17,            r18
        ld.b    0[r18],         r20             -- r20:=表示キャラクタ

        movhi   hi1(lcd_addr), r0, r19          -- r19:=LCD アドレス
        movea   lo(lcd_addr), r19, r19
        st.b    r20,            2[r19]
        addi    1,      r17,    r17             -- col_ctrインクリメント
        st.b    r17,            0[r16]          -- col_ctrのメモリへのストア
        br      tm_exit                         -- 終了処理へ
tm_cursol_home:
        movhi   hi1(lcd_addr), r0, r19          -- r19:=LCD アドレス
        movea   lo(lcd_addr), r19, r19
        movea   0x02,   r0,     r20
        st.b    r20,            0[r19]          -- LCDホーム・ポジション
        movhi   hi1(disp_flg), r0, r16          -- r16:=時計LCD表示状態フラグ
        movea   lo(disp_flg), r16, r16
        clr1    0,              0[r16]          -- disp_flgのオフ
        movhi   hi1(col_ctr), r0, r16           -- r16:=時計表示データの表示カラム数
        movea   lo(col_ctr), r16, r16
        st.b    r0,             0[r16]          -- col_ctr = 0
        br      tm_exit                         -- 終了処理へ
tm_exit:
        mov     r6,             r1              -- r1の復帰
        reti
```

(3) Cプログラム

```
#pragma ioreg                                   /* 周辺I/0レジスタ名を使用可能にする */

/* ------------------------------------------------------
 *  グローバル変数
 *------------------------------------------------------
 */
int flg_wt  ;                                   /* 1秒経過フラグ */
int wt_num ;                                    /* 時間データ */
char tm_buf[100] ;                              /* フォーマットされた時間データ */
int disp_flg = 1 ;                              /* 表示フラグ */
int tm0_num ;                                   /* タイマ0カウンタ */
int col_ctr ;                                   /* 表示カラム位置 */
char *lcd_addr ;                                /* LCDアドレス */

/* ------------------------------------------------------
 *  LCDビジィ・チェック・ルーチン
 *------------------------------------------------------
 */
void bf_chk(void)
{
    while ( ((*lcd_addr) & 0x80) == 0x80 ){
    }
}

/* ------------------------------------------------------
 *  メイン処理
 *------------------------------------------------------
 */
void main(void)
{
    /* グローバル変数初期化 */
    flg_wt = 0 ;                                /* 1秒経過フラグ */
    wt_num = 0 ;                                /* 時間データ */
    disp_flg = 0 ;                              /* 表示フラグ */
    tm0_num = 0 ;                               /* タイマ0カウンタ */
    col_ctr = 0;                                /* 表示カラム位置 */
    lcd_addr = (char *)0x140000;                /* LCDアドレス */

    /* プリスケーラ･モード･レジスタの設定 */
    PRM0 = 0x01 ;                               /* fxx/16(fxx=16MHz):=カウント･クロックを1MHzに指定 */
    TOC0 = 0 ;                                  /* タイマ出力コントロール･レジスタの設定 */
    CR00 = 100 ;                                /* 割り込み間隔を0.1msに設定 */
    /* キャプチャ・コンペア･コントロール・レジスタの設定 */
    CRC0 = 0 ;                                  /* CR00をコンペア･レジスタとして動作 */
    /* タイマ・モード・コントロール・レジスタの設定 */
    TMC0 = 0x0c ;                               /* TM0とCR00の一致でクリア&スタート */
    /* タイマ割り込み許可(レベル7) */
    TMIC00 = 0x07 ;                             /* タイマ割り込み許可(レベル7) */
    __EI();                                     /* 割り込み許可 */

    *lcd_addr = 0x30 ;                          /* ファンクション・セット 8ビット 1行 5*7 */
    for ( tm0_num = 0 ; tm0_num < 41 ; ) {      /* 4.1ms以上待つ */
    }
    *lcd_addr = 0x30 ;                          /* ファンクション・セット 8ビット 1行 5*7 */
    for ( tm0_num = 0 ; tm0_num < 2 ; ) {       /* 100us以上待つ */
    }
    *lcd_addr = 0x30 ;                          /* ファンクション・セット 8ビット 1行 5*7 */
    bf_chk();
    *lcd_addr = 0x38 ;                          /* ファンクション・セット 8ビット 2行 5*7 */
    bf_chk();
    *lcd_addr = 0x01 ;                          /* 表示クリア,カーソルをホーム位置へ */
    bf_chk();
    *lcd_addr = 0x06 ;                          /* カーソルはインクリメント，表示シフトしない */
    bf_chk();
    *lcd_addr = 0x0C ;                          /* 表示をON，カーソルをOFF */
```

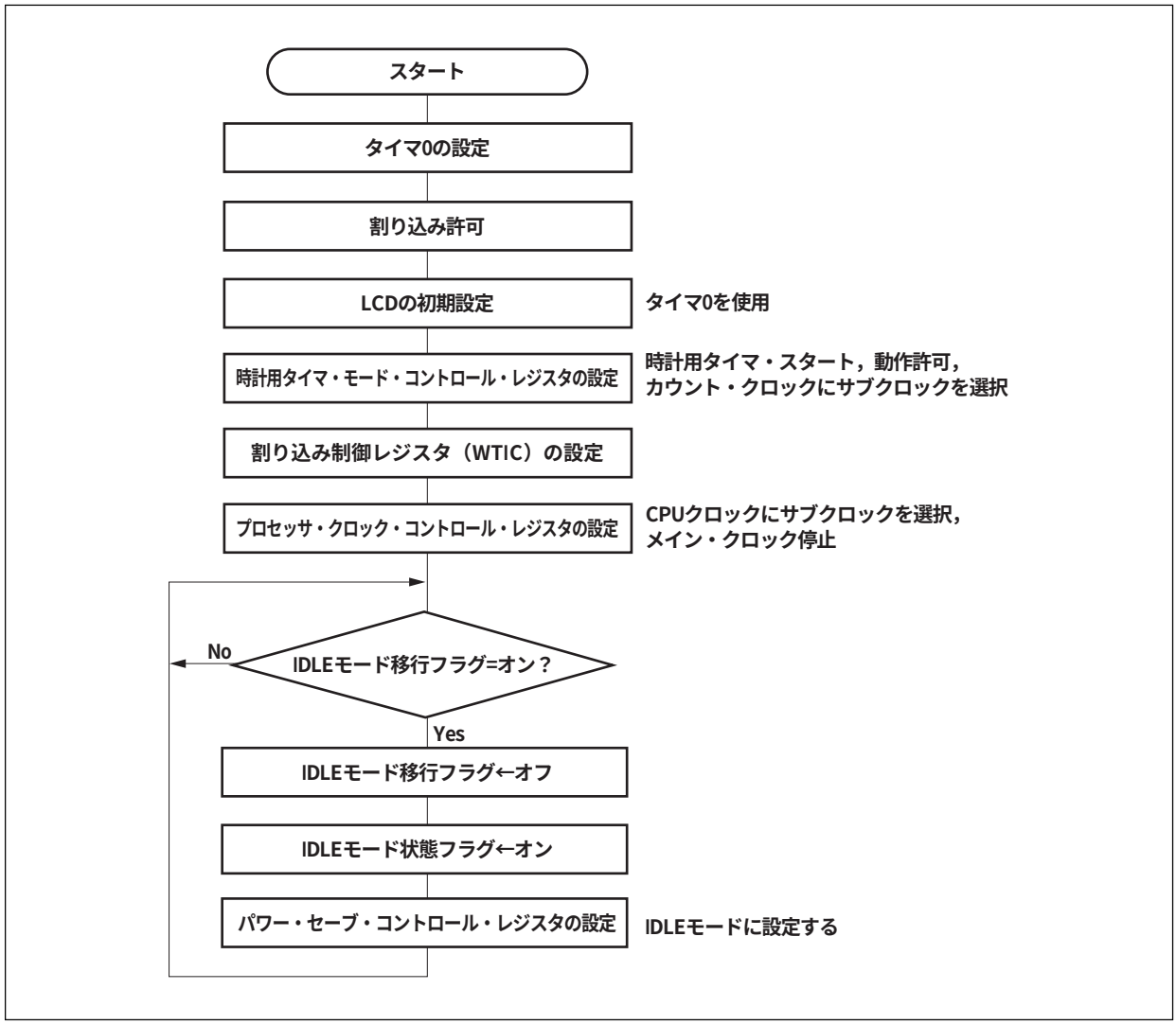
```
    /* 時計用タイマ・モード・コントロール・レジスタの設定 */
    WTM = 0x83 ;                                /* 時計用タイマ・スタート,動作許可 */
                                                /* カウント・クロックをサブクロックに選択 */
    /* 時計割り込みのレベルはタイマ0割り込みレベルより,高いものに設定する */
    WTIC = 0x06 ;                               /* 時計用タイマ割り込み許可(レベル6) */
    /* PCCレジスタは特定レジスタなので,特定のシーケンスにより変更する */
    __asm("movea    0x44, r0, r13");
    __asm("stsr     5,        r11");            /* プログラム・ステータス・ワードのロード */
    __asm("mov      r11,      r12");            /* r12=r11 */
    __asm("or       0x80,     r11");            /* PSWのNPビット=1 */
    __asm("ldsr     r11,        5");            /* NPビット=1にしたPSWを書き込む */
    __asm("st.b     r0, PRCMD[r0]");            /* PRCMDへの書き込み */
    /* CPUクロックをサブクロックに選択 */
    __asm("st.b     r13,  PCC[r0]");            /* プロセッサ・クロック・コントロール・レジスタへの設定 */
    __asm("ldsr     r12,        5");            /* PSWを元に戻す */
    __asm("nop");                               /* ダミー命令(5命令) */
    __asm("nop");
    __asm("nop");
    __asm("nop");
    __asm("nop");
    /* 永久ループ */
    for ( ; ; ) {
    }
}

/* ------------------------------------------------------
 *  時計用タイマ割り込み処理
 *------------------------------------------------------
 */
#pragma interrupt INTWT _int_wt
void _int_wt(void)
{
    if ( flg_wt == 0 ){                         /* 2回目の時計割り込み? */
        flg_wt = 1 ;                            /* 1秒経過フラグのオン */
    }else{
        flg_wt = 0 ;                            /* 1秒経過フラグのオフ */
        wt_num++ ;                              /* 時間データのインクリメント */
        /* 時間データをフォーマットする */
        tm_buf[0] = 0x30 + (char)(wt_num/(60*60*10)) ;
        tm_buf[1] = 0x30 + (char)((wt_num/(60*60))%10) ;
        tm_buf[2] = ':' ;
        tm_buf[3] = 0x30 + (char)(((wt_num%(60*60))/60)/10) ;
        tm_buf[4] = 0x30 + (char)(((wt_num%(60*60))/60)%10) ;
        tm_buf[5] = ':' ;
        tm_buf[6] = 0x30 + (char)((wt_num%60)/10) ;
        tm_buf[7] = 0x30 + (char)((wt_num%60)%10) ;
        disp_flg = 1 ;                          /* 表示フラグのオン */
    }
}

/* ------------------------------------------------------
 *  タイマ0割り込み処理
 *------------------------------------------------------
 */
#pragma interrupt INTTM00 _int_tm00
void _int_tm00(void)
{
    tm0_num++ ;                                 /* タイマ0カウンタのインクリメント */
    if ( disp_flg == 1 ) {                      /* 表示フラグがオン? */
        if ( col_ctr != 8 ) {                   /* 表示カラム位置が8以下? */
        *( lcd_addr + 2 ) = tm_buf[col_ctr] ; /* 時間データ1文字表示 */
        col_ctr++ ;                   /* 表示カラム位置インクリメント */
    }else{
        *lcd_addr = 0x02 ;            /* LCDホーム・ポジション */
        disp_flg = 0 ;                /* 表示フラグのオフ */
        col_ctr = 0 ;                 /* 表示カラム位置を0に戻す */
    }
  }
}
```

4.4.9　IDLEモード時の時計動作とNMIスイッチによる復帰

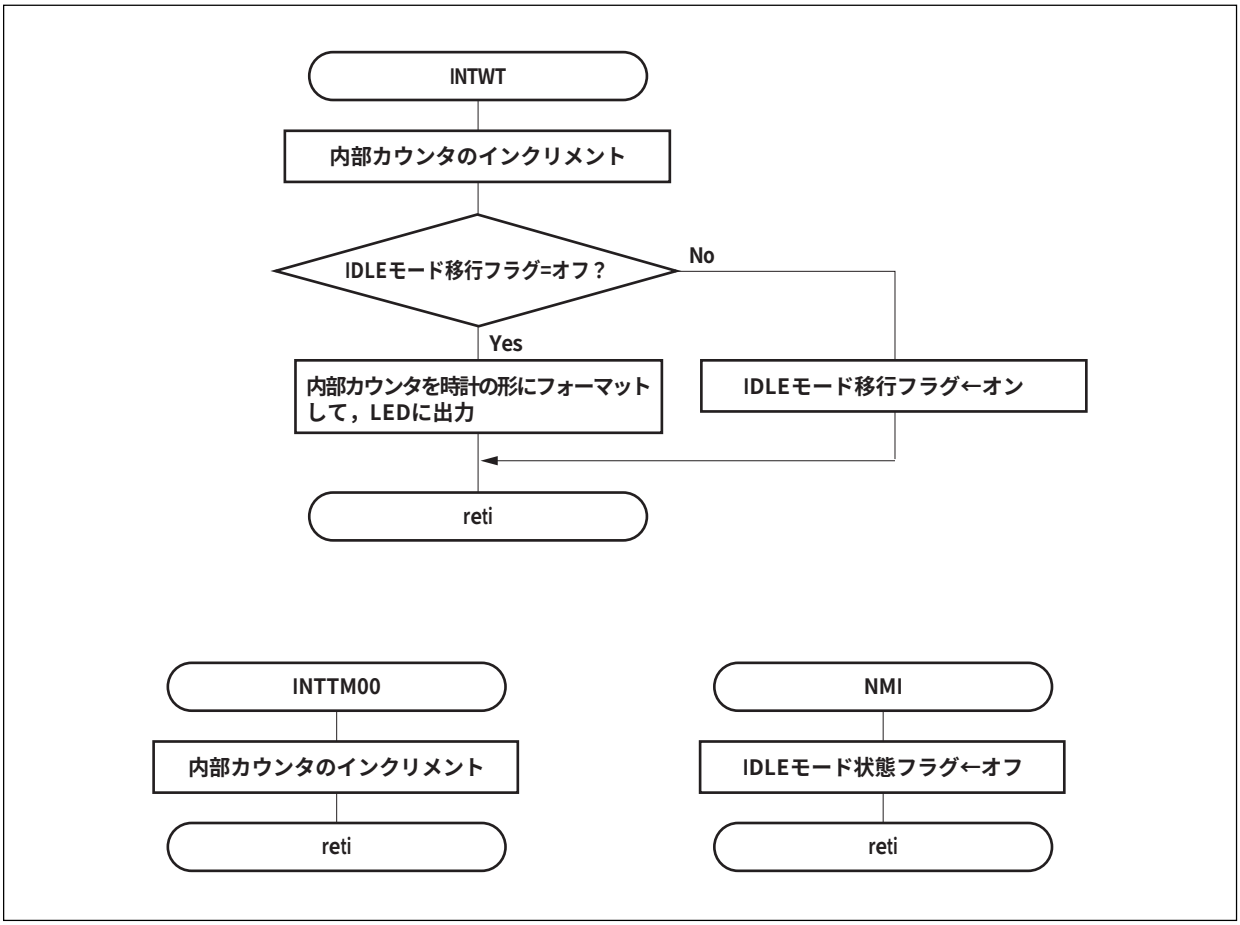
通常モードでは，時計を表示します。

INTスイッチを押すとIDLEモードに移行させます。サブクロックがある場合，時計用タイマはカウント・クロックにfxxを選択しているので，IDLEモード時も動作しています。IDLEモード時にはINTWTにより通常のモードに復帰しますが，ここでは内部カウンタを更新するときだけで，表示しないでIDLEモードに再度移行させます。NMIにより通常モードに復帰したとき，内部フラグをクリアして，時計を表示します。

(1) フロー・チャート (1/2)



（1）フロー・チャート（2/2）


INTWT
内部カウンタのインクリメント
IDLEモード移行フラグ=オフ？
No
Yes
内部カウンタを時計の形にフォーマットして，LEDに出力
IDLEモード移行フラグ←オン
reti
INTTM00
内部カウンタのインクリメント
reti
NMI
IDLEモード状態フラグ←オフ
reti

(2) アセンブラ・プログラム

```
.globl  start

.globl  _int_nmi
.globl  _int_p0
.globl  _int_wt
.globl  _int_tm00

.set    tm_buf,      0x120000                   -- 時計表示データ(xx:xx:xx)
.set    lcd_addr,    0x140000                   -- LCDアドレス

## 割り込みハンドラのセット
.section "NMI"                                  -- NMI割り込みハンドラ
        jr      _int_nmi                        -- 割り込み処理へ
.section "INTP0"                                -- INTP割り込みハンドラ
        jr      _int_p0                         -- 割り込み処理へ
.section "INTWT"                                -- INTWT割り込みハンドラ
        jr      _int_wt                         -- 割り込み処理へ
.section "INTTM00"                              -- INTTM00割り込みハンドラ
        jr      _int_tm00                       -- 割り込み処理へ

.text
        .align 4
start:
        mov     r0,          r26                -- r26(1秒経過フラグ)のオフ
        mov     r0,          r27                -- 時間データの初期化
        mov     r0,          r28                -- IDLEモードを示すフラグ:オフ

init_lcd:
        mov     0x01,        r11                -- r11:=1
        -- fxx/16(fxx=16MHz):=カウント･クロックを1MHzに指定
        st.b    r11,         PRM0[r0]           -- プリスケーラ･モード･レジスタ0の設定
        st.b    r0,          TOC0[r0]           -- タイマ出力コントロール･レジスタ0の設定
        movea   0x64,  r0,   r11                -- r11:=0x64
        -- 割り込み間隔を0.1msに設定
        st.h    r11,         CR00[r0]
        -- CR00をコンペア･レジスタとして動作
        st.b    r0,          CRC0[r0]           -- キャプチャ/コンペア･コントロール･レジスタ0の設定
        mov     0x0c,        r11
        -- TM0とCR00の一致でクリア&スタート
        st.b    r11,         TMC0[r0]           -- タイマ･モード･コントロール･レジスタ0の設定
        mov     0x07,        r11
        st.b    r11,         TMIC00[r0]         -- タイマ割り込み許可
        mov     r0,          r25                -- r25:=タイマ･カウンタとして使用
        ei

        movea   lcd_addr & 0xffff, r0, r20
        movhi   lcd_addr >> 16, r20, r20        -- LCD アドレス
        movea   0x30,  r0,   r10
        st.b    r10,         0[r20]             -- ファンクション・セット 8ビット 1行 5*7
wait_1:
        cmp     41,           r25
        jn      wait_1                          -- 4.1ms以上待つ

        movea   0x30,  r0,   r10
        st.b    r10,         0[r20]             -- ファンクション・セット 8ビット 1行 5*7
        mov     r0,          r25
wait_2:
        cmp     2,           r25
        jn      wait_2                          -- 100us以上待つ
        movea   0x30,  r0,   r10
        jarl    bf_chk,      r29                -- LCDビジィ・チェック
        st.b    r10,         0[r20]             -- ファンクション・セット 8ビット 1行 5*7
        movea   0x38,  r0,   r10
        jarl    bf_chk,      r29                -- LCDビジィ・チェック
        st.b    r10,         0[r20]             -- ファンクション・セット 8ビット 2行 5*7
        mov     0x01,        r10
```

```
        jarl    bf_chk,       r29               -- LCDビジィ・チェック
        st.b    r10,          0[r20]            -- 表示クリア,カーソルをホーム位置へ
        mov     0x06,         r10
        jarl    bf_chk,       r29               -- LCDビジィ・チェック
        st.b    r10,          0[r20]            -- カーソルはインクリメント，表示シフトしない
        mov     0x0C,         r10
        jarl    bf_chk,       r29               -- LCDビジィ・チェック
        st.b    r10,          0[r20]            -- 表示をON，カーソルをOFF
        movea   0x47,  r0,    r11
        st.b    r11,          TMIC00[r0]        -- タイマ割り込み不許可

init_wt:
        ## 時計用タイマ･モード･コントロール･レジスタの設定
        movea   0x83,  r0,    r10
        st.b    r10,          WTM[r0]           -- 時計用タイマ・スタート,動作許可
        -- カウント･クロックをサブクロックに選択
        mov     0x07,         r10
        st.b    r10,          WTIC[r0]          -- 時計用タイマ割り込み許可
init_pic0:
        mov     0x07,         r10
        st.b    r10,          PIC0[r0]          -- INTスイッチ割り込み許可
set_sub_clock:
        ## PSCレジスタは特定レジスタなので，特定のシーケンスにより変更する
        stsr    5,            r21               -- プログラム・ステータス・ワードのロード
        mov     r21,          r22               -- r22=r21
        or      0x80,         r21               -- PSWのNPビット=1
        movea   0xc0,   r0,   r10
        ldsr    r21,          5
        st.b    r0,           PRCMD[r0]         -- PRCMDへの書き込み
        ## CLKOUT出力停止する
        st.b    r10,          PSC[r0]           -- パワー・セーブ・コントロール・レジスタの設定
        ldsr    r22,          5
        nop                                     -- ダミー命令(2命令)
        nop
        ## PCCレジスタは特定レジスタなので，特定のシーケンスにより変更する
        movea   0x64,   r0,   r13
        stsr    5,            r11               -- プログラム・ステータス・ワードのロード
        mov     r11,          r12               -- r12=r11
        or      0x80,         r11               -- PSWのNPビット=1
        ldsr    r11,          5                 -- NPビット=1にしたPSWを書き込む
        st.b    r0,           PRCMD[r0]         -- PRCMDへの書き込み
        ## サブクロック選択,メイン・クロック停止
        st.b    r13,          PCC[r0]           -- プロセッサ・クロック・コントロール・レジスタへの設定
        ldsr    r12,          5                 -- PSWを元に戻す
        nop                                     -- ダミー命令(2命令)
        nop
fv_loop:
        cmp     r0,           r24               -- r24:=INTスイッチ押下フラグ・オン
        bz      fv_loop
        mov     r0,           r24               -- r24:=INTスイッチ押下フラグ・オフ
        mov     1,            r28               -- IDLEモードを示すフラグ:オン
        ## 以降，特定レジスタを設定する手順
        stsr    5,            r21
        mov     r21,          r22
        or      0x80,         r21
        mov     0x04,         r10
        ldsr    r21,          5
        st.b    r0,           PRCMD[r0]         -- PRCMDへの書き込み
        ## CLKOUT出力許可，IDLEモードに設定する
        st.b    r10,          PSC[r0]           -- パワー・セーブ・コントロール・レジスタの設定
        ldsr    r22,          5
        nop                                     -- ダミー命令(2命令)
        nop
        br      fv_loop                         -- 永久ループ

bf_chk:
        movhi   hi1(lcd_addr), r0, r7           -- r7:=LCD アドレス
        movea   lo(lcd_addr), r7, r7
        movea   0x80,   r0,   r8                -- r8:=0x80
```

```
bf_chk0:
        ld.b    0[r7],        r9
        andi    0x80,   r9,   r9
        cmp     r8,           r9
        bz      bf_chk0
        jmp     [r29]

_int_wt:                                        -- 時計用タイマ割り込み処理
        cmp     r0,           r26               -- r26(1秒経過フラグ)のチェック
        bnz     r11_0_no
        mov     1,            r26               -- r26(1秒経過フラグ)のオフ
        add     1,            r27               -- 時間データのインクリメント
        cmp     1,            r28               -- IDLEモード?

        bz      disp_skip                       --  if yes ,then jump
        ## r10 を時計の形にフォーマットしてLCDに表示
        movhi   hi1(tm_buf), r0, r20
        movea   lo(tm_buf), r20, r20
        movea   0x03a,  r0,   r21               -- ':'キャラクタのセット
        st.b    r21,          2[r20]            -- 時間表示文字列の2文字目
        st.b    r21,          5[r20]            -- 時間表示文字列の5文字目
        mov     r27,          r7                -- r7=r27
        movea   60*60*10, r0, r6
        andi    0xffff,   r6, r6                -- r6=60*60*10
        jarl    ___div,       lp                -- r6=r7/(60*60*10)
        add     0x30,         r6                -- r6=0x30+r6
        st.b    r6,           0[r20]            -- x*10時間
        mov     r27,          r15               -- r15=r27
        movea   60*60,   r0,  r6                -- r6=60*60
        divh    r6            r15               -- r15=r15/(60*60)
        mov     r15,          r16               -- r16=r15
        mov     10,           r7                -- r7=10
        divh    r7,           r16               -- r16=r16/10
        mulh    10,           r16               -- r16=r16*10
        sub     r16,          r15               -- r15=r15-r16
        add     0x30,         r15               -- r15=0x30+r15
        st.b    r15,          1[r20]            -- x時間
        mov     r27,          r15               -- r15=r27
        divh    r6,           r15               -- r15=r15/(60*60)
        mulh    60*60,        r15               -- r15=r15*(60*60)
        mov     r27,          r16               -- r16=r27
        sub     r15,          r16               -- r16=r16-r15
        movea   60,           r0, r6            -- r6=60
        divh    r6,           r16               -- r16=r16/60
        mov     r16,          r15               -- r15=r16
        divh    r7,           r15               -- r15=r15/10
        add     0x30,         r15               -- r15=0x30+r15
        st.b    r15,          3[r20]            -- x*10分
        mov     r16,          r15               -- r15=r16
        divh    r7,           r16               -- r16=r16/10
        mulh    10,           r16               -- r16=r16*10
        sub     r16,          r15               -- r15=r15-r16
        add     0x30,         r15               -- r15=0x30+r15
        st.b    r15,          4[r20]            -- x分
        mov     r27,          r15               -- r15=r27
        divh    r6,           r15               -- r15=r15/60
        mulh    60,           r15               -- r15=r15*60
        mov     r27,          r16               -- r16=r27
        sub     r15,          r16               -- r16=r16-r15
        mov     r16,          r15               -- r15=r16
        divh    r7,           r15               -- r15=r15/10
        add     0x30,         r15               -- r15=0x30+r15
        st.b    r15,          6[r20]            -- x*10秒
        mov     r16,          r15               -- r15=r16
        divh    r7,           r16               -- r16=r16/10
        mulh    10,           r16               -- r16=r16*10
        sub     r16,          r15               -- r15=r15-r16
        add     0x30,         r15               -- r15=0x30+r15
        st.b    r15,          7[r20]            -- x秒
```

```
        movea   lcd_addr & 0xffff, r0, r21
        movhi   lcd_addr >> 16, r21, r21        -- r21:=LCDアドレス
        mov     0x02,           r15
        jarl    bf_chk,         r29             -- LCDビジィ・チェック
        st.b    r15,            0[r21]          -- LCDホーム・ポジション
        ld.b    0[r20],         r15             -- 時間データ1文字ロード
        jarl    bf_chk,         r29             -- LCDビジィ・チェック
        st.b    r15,            2[r21]          -- LCDに時間データ1文字表示
        ld.b    1[r20],         r15             -- 時間データ1文字ロード
        jarl    bf_chk,         r29             -- LCDビジィ・チェック
        st.b    r15,            2[r21]          -- LCDに時間データ1文字表示
        ld.b    2[r20],         r15             -- 時間データ1文字ロード
        jarl    bf_chk,         r29             -- LCDビジィ・チェック
        st.b    r15,            2[r21]          -- LCDに時間データ1文字表示
        ld.b    3[r20],         r15             -- 時間データ1文字ロード
        jarl    bf_chk,         r29             -- LCDビジィ・チェック
        st.b    r15,            2[r21]          -- LCDに時間データ1文字表示
        ld.b    4[r20],         r15             -- 時間データ1文字ロード
        jarl    bf_chk,         r29             -- LCDビジィ・チェック
        st.b    r15,            2[r21]          -- LCDに時間データ1文字表示
        ld.b    5[r20],         r15             -- 時間データ1文字ロード
        jarl    bf_chk,         r29             -- LCDビジィ・チェック
        st.b    r15,            2[r21]          -- LCDに時間データ1文字表示
        ld.b    6[r20],         r15             -- 時間データ1文字ロード
        jarl    bf_chk,         r29             -- LCDビジィ・チェック
        st.b    r15,            2[r21]          -- LCDに時間データ1文字表示
        ld.b    7[r20],         r15             -- 時間データ1文字ロード
        jarl    bf_chk,         r29             -- LCDビジィ・チェック
        st.b    r15,            2[r21]          -- LCDに時間データ1文字表示
        br      disp_skip
r11_0_no:
        mov     r0,             r26             -- r26(1秒経過フラグ)のオン
disp_skip:
        cmp     1,              r28             -- IDLEモード?
        bnz     int_wt_exit                     -- if no ,then jump
        mov     1,              r24             -- r24:=INTスイッチ押下フラグ･オン
int_wt_exit:
        reti                                    -- 割り込み処理から復帰


### INTスイッチ押下によりIDLEモードにする
_int_p0:                                        -- INTスイッチ割り込み処理
        mov     1,              r24             -- r24:=INTスイッチ押下フラグ･オン
        reti

_int_nmi:                                       -- NMI割り込み処理
        mov     0,              r28             -- IDLEモードを示すフラグ:オフ
        reti

_int_tm00:                                      -- タイマ0割り込み処理
        add     1,              r25             -- r25をインクリメント
        reti
```

(3) Cプログラム

```
#pragma ioreg                                   /* 周辺I/0レジスタ名を使用可能にする */

/* ------------------------------------------------------
 *  グローバル変数
 *------------------------------------------------------
 */
int flg_wt  ;                                   /* 1秒経過フラグ */
int wt_num ;                                    /* 時間データ */
char tm_buf[8] ;                                /* フォーマットされた時間データ */
int tm0_num ;                                   /* タイマ0カウンタ */
int intsw_flg ;                                 /* IDLEモード移行フラグ */
int idle_mode ;                                 /* IDLEモード状態フラグ */
char *lcd_addr ;                                /* LCDアドレス */

/* ------------------------------------------------------
 *  LCDビジィ・チェック・ルーチン
 *------------------------------------------------------
 */
void bf_chk(void)
{
    while ( ((*lcd_addr) & 0x80) == 0x80 ){
    }
}

/* ------------------------------------------------------
 *  メイン処理
 *------------------------------------------------------
 */
void main(void)
{
    /* グローバル変数初期化 */
    flg_wt = 0 ;                                /* 1秒経過フラグ */
    wt_num = 0 ;                                /* 時間データ */
    tm0_num = 0 ;                               /* タイマ0カウンタ */
    intsw_flg = 0 ;                             /* IDLEモード移行フラグ */
    idle_mode = 0 ;                             /* IDLEモード状態フラグ */
    lcd_addr = (char *)0x140000;                /* LCDアドレス */

    /* プリスケーラ･モード･レジスタの設定 */
    PRM0 = 0x01 ;                               /* fxx/16(fxx=16MHz):=カウント･クロックを1MHzに指定 */
    TOC0 = 0 ;                                  /* タイマ出力コントロール･レジスタ0の設定 */
    CR00 = 100 ;                                /* 割り込み間隔を0.1msに設定 */
    /* キャプチャ･コンペア･コントロール･レジスタの設定 */
    CRC0 = 0 ;                                  /* CR00をコンペア･レジスタとして動作 */
    /* タイマ･モード･コントロール･レジスタの設定 */
    TMC0 = 0x0c ;                               /* TM0とCR00の一致でクリア&スタート */
    /* タイマ割り込み許可(レベル7) */
    TMIC00 = 0x07 ;                             /* タイマ割り込み許可(レベル7) */
    PIC0 = 0x07 ;                               /* INTスイッチ割り込み許可 */
    __EI();                                     /* 割り込み許可 */

    *lcd_addr = 0x30 ;                          /* ファンクション・セット 8ビット 1行 5*7 */
    for ( tm0_num = 0 ; tm0_num < 41 ; ) {      /* 4.1ms以上待つ */
    }
    *lcd_addr = 0x30 ;                          /* ファンクション・セット 8ビット 1行 5*7 */
    for ( tm0_num = 0 ; tm0_num < 2 ; ) {       /* 100us以上待つ */
    }
    *lcd_addr = 0x30 ;                          /* ファンクション・セット 8ビット 1行 5*7 */
    bf_chk();
    *lcd_addr = 0x38 ;                          /* ファンクション・セット 8ビット 2行 5*7 */
    bf_chk();
    *lcd_addr = 0x01 ;                          /* 表示クリア,カーソルをホーム位置へ */
    bf_chk();
    *lcd_addr = 0x06 ;                          /* カーソルはインクリメント，表示シフトしない */
    bf_chk();
    *lcd_addr = 0x0C ;                          /* 表示をON，カーソルをOFF */
```

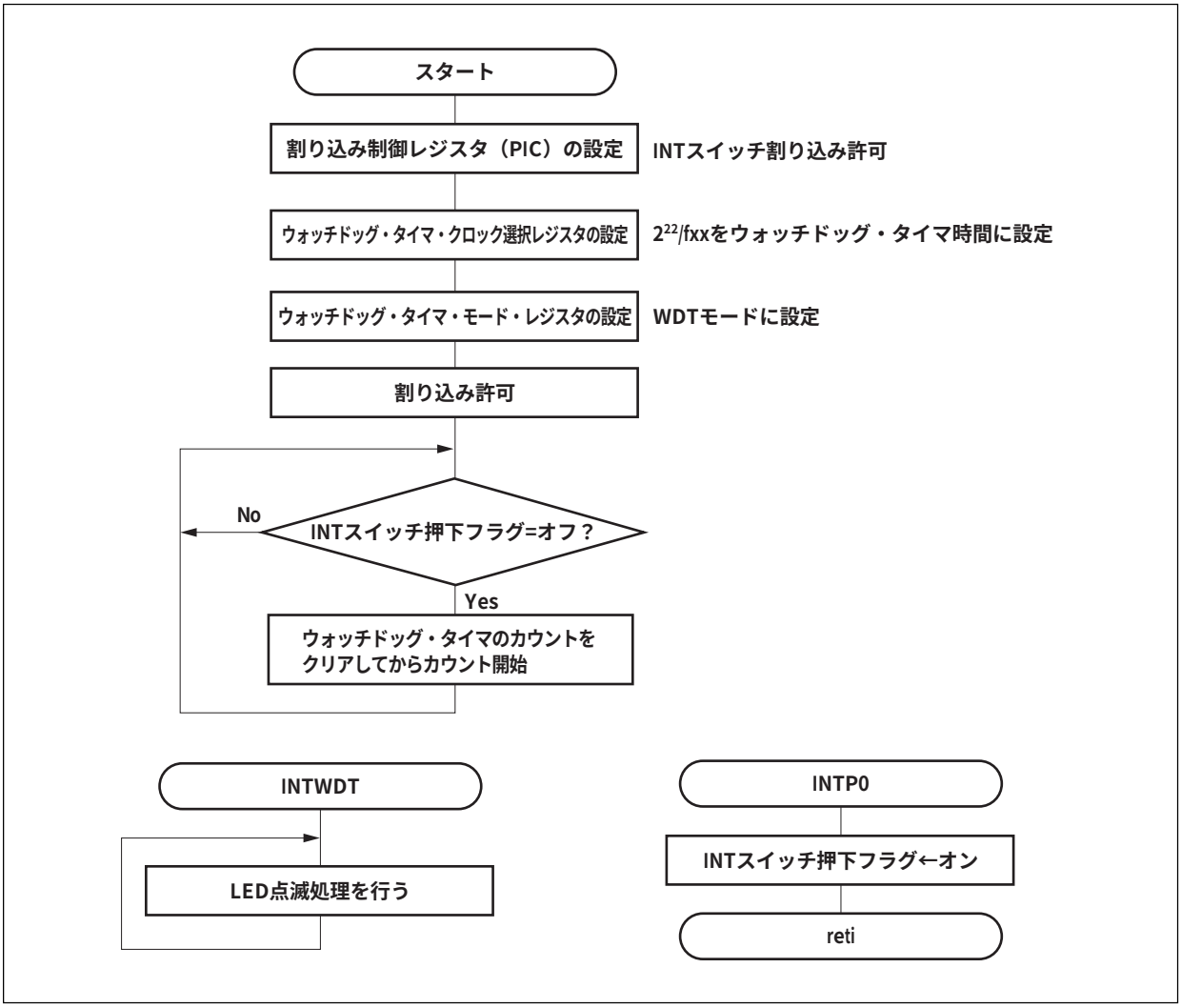
```
    /* 時計用タイマ･モード･コントロール･レジスタの設定 */
    WTM = 0x83 ;                                /* 時計用タイマ・スタート,動作許可 */
                                                /* カウント・クロックをサブクロックに選択 */
    /* 時計割り込みのレベルはタイマ0割り込みレベルより,高いものに設定する */
    WTIC = 0x06 ;                               /* 時計用タイマ割り込み許可(レベル6) */

/* PSCレジスタは特定レジスタなので，特定のシーケンスにより変更する */
    __asm("stsr     5,      r21") ;
    __asm("mov      r21,    r22") ;
    __asm("or       0x80,   r21") ;
    __asm("movea    0xC0,   r0, r10");
    __asm("ldsr     r21,    5") ;       /* PSWのNPビットをオンにしてライト */
    __asm("st.b     r0,     PRCMD[r0]");/* PRCMDへの書き込み */
    /* CLKOUT出力禁止 */
    __asm("st.b     r10,    PSC[r0]") ; /* CLKOUT出力禁止に設定する */
    __asm("ldsr     r22,    5") ;       /* PSWを元に戻す */
    __asm("nop") ;                      /* ダミー命令(2命令) */
    __asm("nop") ;
/* PCCレジスタは特定レジスタなので，特定のシーケンスにより変更する */
    __asm("movea    0x64,   r0, r13");
    __asm("stsr     5,      r11");      /* プログラム・ステータス・ワードのロード */
    __asm("mov      r11,    r12");      /* r12=r11 */
    __asm("or       0x80,   r11");      /* PSWのNPビット=1 */
    __asm("ldsr     r11,    5");        /* PSWのNPビットをオンにしてライト */
    __asm("st.b     r0,     PRCMD[r0]");/* PRCMDへの書き込み */
    /* CPUクロックをサブクロックに選択，メイン・クロック停止 */
    __asm("st.b     r13,    PCC[r0]");/* プロセッサ・クロック・コントロール・レジスタへの設定 */
    __asm("ldsr     r12,    5");      /* PSWを元に戻す */
    __asm("nop");                       /* ダミー命令(2命令) */
    __asm("nop");
    /* 永久ループ */
    for ( ; ; ) {
        if ( intsw_flg == 1 ) {                 /* IDLEモード移行フラグがオン ? */
            intsw_flg = 0 ;                     /* IDLEモード移行フラグのオフ */
            idle_mode = 1 ;                     /* IDLEモード状態フラグのオン */
            /* PSCレジスタは特定レジスタなので，特定のシーケンスにより変更する */
            __asm("stsr 5,      r21") ;
            __asm("mov  r21,    r22") ;
            __asm("or   0x80,   r21") ;
            __asm("mov  0x04,   r10") ;
            __asm("ldsr r21,    5") ;           /* PSWのNPビットをオンにしてライト */
            __asm("st.b r0,     PRCMD[r0]") ;   /* PRCMDへの書き込み */
            /* パワー・セーブ・コントロール・レジスタの設定 */
            __asm("st.b r10,    PSC[r0]") ;     /* CLKOUT出力許可，IDLEモードに設定する */
            __asm("ldsr r22,    5") ;           /* PSWを元に戻す */
            __asm("nop") ;                      /* ダミー命令(2命令) */
            __asm("nop") ;
        }
    }
}

/* ------------------------------------------------------
 *  時計用タイマ割り込み処理
 *------------------------------------------------------
 */
#pragma interrupt INTWT _int_wt
void _int_wt(void)
{
    if ( flg_wt == 0 ){                         /* 2回目の時計割り込み? */
        flg_wt = 1 ;                            /* 1秒経過フラグのオン */
    }else{
        flg_wt = 0 ;                            /* 1秒経過フラグのオフ */
        wt_num++ ;                              /* 時間データのインクリメント */
        if ( idle_mode == 0 ) {
            /* 時間データをフォーマットする */
            tm_buf[0] = 0x30 + (char)(wt_num/(60*60*10)) ;
            tm_buf[1] = 0x30 + (char)((wt_num/(60*60))%10) ;
            tm_buf[2] = ':' ;
```

```
            tm_buf[3] = 0x30 + (char)(((wt_num%(60*60))/60)/10) ;
            tm_buf[4] = 0x30 + (char)(((wt_num%(60*60))/60)%10) ;
            tm_buf[5] = ':' ;
            tm_buf[6] = 0x30 + (char)((wt_num%60)/10) ;
            tm_buf[7] = 0x30 + (char)((wt_num%60)%10) ;
            /* LCDに出力 */
            bf_chk();                           /* LCDビジィ・チェック */
            *( lcd_addr + 2 ) = tm_buf[0] ;     /* 時間データ1文字表示 */
            bf_chk();                           /* LCDビジィ・チェック */
            *( lcd_addr + 2 ) = tm_buf[1] ;     /* 時間データ1文字表示 */
            bf_chk();                           /* LCDビジィ・チェック */
            *( lcd_addr + 2 ) = tm_buf[2] ;     /* 時間データ1文字表示 */
            bf_chk();                           /* LCDビジィ・チェック */
            *( lcd_addr + 2 ) = tm_buf[3] ;     /* 時間データ1文字表示 */
            bf_chk();                           /* LCDビジィ・チェック */
            *( lcd_addr + 2 ) = tm_buf[4] ;     /* 時間データ1文字表示 */
            bf_chk();                           /* LCDビジィ・チェック */
            *( lcd_addr + 2 ) = tm_buf[5] ;     /* 時間データ1文字表示 */
            bf_chk();                           /* LCDビジィ・チェック */
            *( lcd_addr + 2 ) = tm_buf[6] ;     /* 時間データ1文字表示 */
            bf_chk();                           /* LCDビジィ・チェック */
            *( lcd_addr + 2 ) = tm_buf[7] ;     /* 時間データ1文字表示 */
            bf_chk();                           /* LCDビジィ・チェック */
            *lcd_addr = 0x02 ;                  /* LCDホーム・ポジション */
        }else{
            /* メイン処理で再度IDLEモードにするためのフラグをオン */
            intsw_flg = 1 ;
        }
    }
}

/* ------------------------------------------------------
 *  タイマ0割り込み処理
 *------------------------------------------------------
 */
#pragma interrupt INTTM00 _int_tm00
void _int_tm00(void)
{
    tm0_num++ ;                                 /* タイマ0カウンタのインクリメント */
}

/* ------------------------------------------------------
 *  INTスイッチ割り込み処理
 *------------------------------------------------------
 */
#pragma interrupt INTP0 _int_p0
void _int_p0(void)
{
    intsw_flg = 1 ;                             /* IDLEモード移行フラグ・オン */
    idle_mode = 1 ;                             /* IDLEモード状態フラグ・オン */
}

/* ------------------------------------------------------
 *  NMI割り込み処理
 *------------------------------------------------------
 */
#pragma interrupt NMI _int_nmi
void _int_nmi(void)
{
idle_mode = 0 ;                         /* IDLEモード状態フラグ・オフ */
}
```

4. 4. 10　ウォッチドッグ・タイマによる暴走検出プログラム

INTスイッチを押すとウォッチドッグ･タイマ・モード・レジスタのRUNビットをセットしないループになります。このとき．ウォッチドッグ・タイマが動作してINTWDTが発生します。ここでの処理はLEDを点滅させます。

(1) フロー・チャート



(2) アセンブラ・プログラム

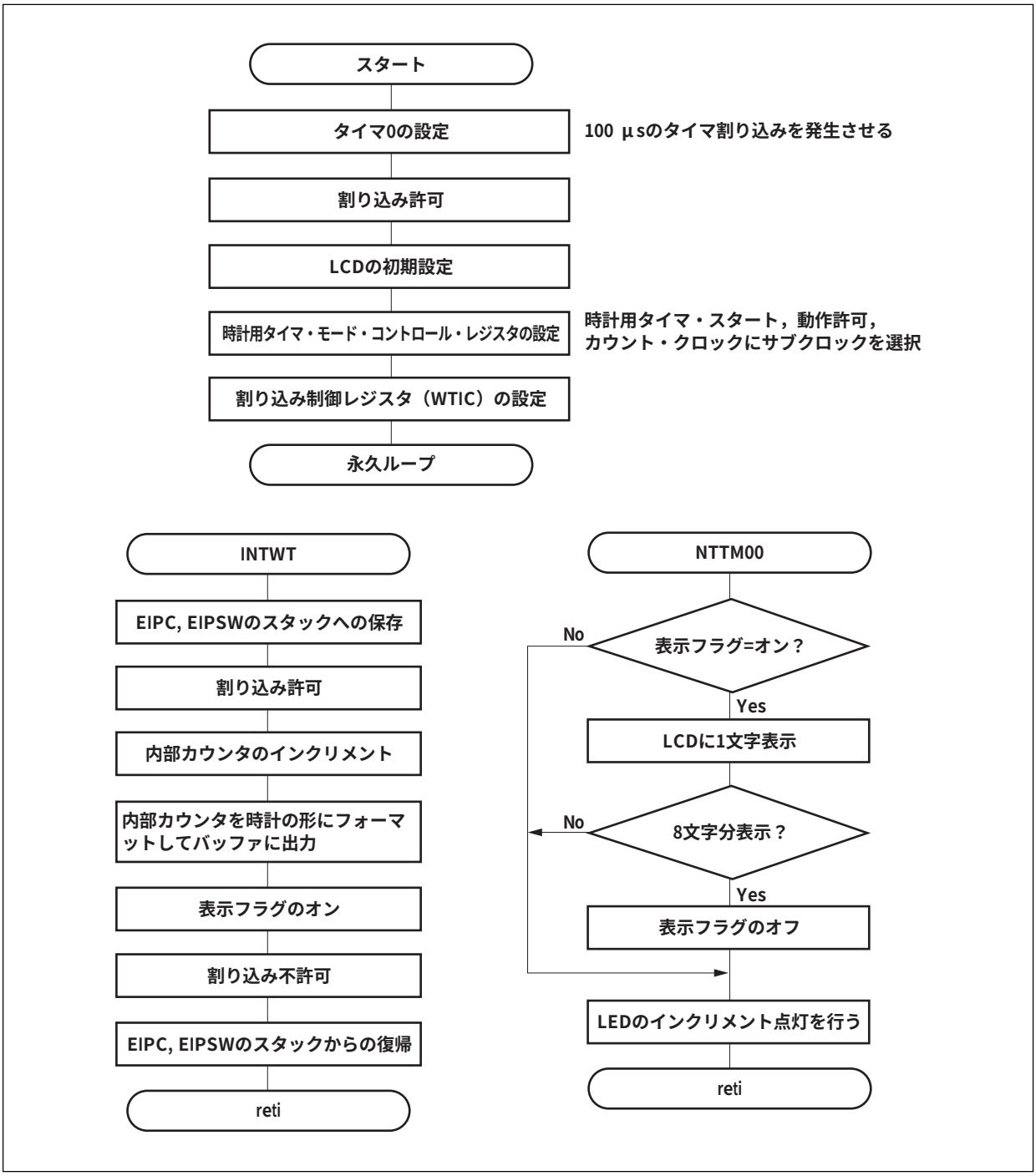
```
.globl  start

## 割り込みハンドラのセット
.section "INTPO"                                -- INTPO割り込みハンドラ
        jr      _int_p0                         -- 割り込み処理へ
.section "INTWDT"                               -- INTWDT割り込みハンドラ
        jr      _int_wdt                        -- 割り込み処理へ

.text
        .align 4
start:

        mov     0x07,    r10
        st.b    r10,     PIC0[r0]               -- INTスイッチ割り込み許可

        mov     0x07,    r10
        ## (2の22乗)/fxxをウォッチドッグ･タイマ時間に設定
        st.b    r10,     WDCS[r0]               -- ウォッチドッグ･タイマ･クロック選択レジスタの設定

        movea   0x10,    r0, r10
        ## ウォッチドッグ･タイマ･モード1に設定
        st.b    r10,     WDTM[r0]               -- ウォッチドッグ･タイマ･モード･レジスタの設定
        mov     r0,      r20                    -- r20(INTSW押下フラグ)のオフ
        ei                                      -- 割り込み許可
LABEL1:
        nop
        nop
        nop
        nop
        nop
        -- r20が0だったら(すなわちINTスイッチが押されていなければ)
        cmp     r0,      r20
        bnz     LABEL1                          -- ジャンプせず
        set1    7,       WDTM[r0]               -- ウォッチドッグ・タイマのカウンタをクリアし,カウントを開始する
        br      LABEL1

## ウォッチドッグ･タイマが暴走検出するとここにジャンプする
led_start:                                      -- LED点滅処理
        mov     r0,      r16                    -- r16:=点灯か消灯かのフラグ
led_on_off:
        jarl    wait,    r29
        cmp     r0,      r16
        bz      led_on                          -- r16=0でLED点灯処理へ
        br      led_off                         -- r16=1でLED消灯処理へ
led_on:
        movea   0x10,    r0, r17
        st.b    r17,     P3[r0]                 -- LED 点灯
        mov     1,       r16                    -- r16=1にしLED消灯処理へ
        br      led_on_off
led_off:
        mov     r0,      r17
        st.b    r17,     P3[r0]                 -- LED 消灯
        mov     r0,      r16                    -- r16=0にしLED点灯処理へ
        br      led_on_off

wait:                                           -- ソフトウエア・ウエイト処理
        or1     1000000 & 0xffff, r0, r15
        movhi   1000000 >> 16, r15, r15
wait_1:
        add     -1,      r15                    -- 1000000回ループする
        bnz     wait_1
        jmp     [r29]

_int_p0:
        mov     r1,      r6                     -- r1の退避
        mov     1,       r20                    -- r20のオン
```

```
        mov     r6,     r1                      -- r1の復帰
        reti

_int_wdt:                                       -- ウォッチドッグ･タイマ割り込み処理
        jr      led_start                       -- LED点滅処理へ
```

4. 4. 11　多重割り込み

時計用タイマ割り込み処理中も，タイマ0割り込みを処理します。時計用タイマ割り込みでは内部時計カウンタをインクリメントして，タイマ0割り込みでは時計データの表示とLEDのインクリメント点灯を行います。

(1) フロー・チャート



(2) アセンブラ・プログラム

```
# ＜使用レジスタ＞
#   r3                  ：スタック・アドレス
#   r6                  ：r1退避用レジスタ
#   r11,r12,r13,r14,r15：メイン処理作業用
#   r16,r17,r18,r19,r20：タイマ0割り込み処理作業用
#   r21,r22,r23,r24,r25：時計用タイマ割り込み処理作業用
#   r26,r27,r28,r29     ：bf_chk ルーチン・ワーク・レジスタ
.globl  start

.set    tm_buf,     0x120000                -- 時計表示データ(xx:xx:xx)(8バイト)
.set    col_ctr,    0x120008                -- 時計表示データの表示カラム数(1バイト)
.set    flg_wt,     0x120009                -- 1秒経過フラグ(1バイト)
.set    wt_num,     0x120010                -- 時計データ(4バイト)
.set    disp_flg,   0x120014                -- 時計LCD表示状態フラグ(1で表示)
.set    tm0_num,    0x120018                -- タイマ0カウンタ(4バイト)
.set    led_ivl,    0x12001c                -- LED 表示インターバル(4バイト)
.set    mem_last,   0x120020                -- ワーク変数領域最終アドレス
.set    lcd_addr,   0x140000                -- LCDアドレス
.set    stk_addr,   0xffefff                -- スタック･アドレス

## 割り込みハンドラのセット
.section "INTWT"                        -- INTWT割り込みハンドラ
        jr      _int_wt                         -- 割り込み処理へ
.section "INTTM00"                      -- INTTM00割り込みハンドラ
        jr      _int_tm00                       -- 割り込み処理へ

.text
        .align 4
start:
        mem_init:
        movhi   hi1(tm_buf), r0, r11            -- r11:=ワーク・メモリの先頭アドレス
        movea   lo(tm_buf), r11, r11
        movhi   hi1(mem_last), r0, r12          -- r12:=ワーク変数領域最終アドレス
        movea   lo(mem_last), r12, r12
mem_init_lp:
        st.b    r0,           0[r11]            -- ワーク変数の0セット
        addi    1,    r11,    r11               -- 0クリアするアドレスのインクリメント
        cmp     r11,          r12               -- 0クリアするアドレスが最終アドレスでなければ
        bnz     mem_init_lp                     -- ループする
ins_3a:
        movhi   hi1(tm_buf), r0, r11            -- r11:=ワーク・メモリの先頭アドレス
        movea   lo(tm_buf), r11, r11
        movea   0x3a, r0,     r13               -- 時間データに
        st.b    r13,          2[r11]            -- ':'=0x3Aをセットする
        st.b    r13,          5[r11]            -- ':'=0x3Aをセットする
set_stack_adr:
        movhi   hi1(stk_addr), r0, r3           -- スタック・アドレスの設定
        movea   lo(stk_addr), r3, r3

init_tm0:
        movea   0x01, r0,  r11                  -- r11:=1
        -- fxx/16(fxx=16MHz):=カウント･クロックを1MHzに指定
        st.b    r11,          PRM0[r0]          -- プリスケーラ･モード･レジスタ0の設定
        st.b    r0,           TOC0[r0]          -- タイマ出力コントロール･レジスタ0の設定
        movea   0x64, r0,     r11               -- r11:=100=0x64
        -- 割り込み間隔を0.1msに設定
        st.h    r11,          CR00[r0]
        -- CR00をコンペア･レジスタとして動作
        st.b    r0,           CRC0[r0]          -- キャプチャ/コンペア･コントロール･レジスタ0の設定
        mov     0x0c,         r11
        -- TM0とCR00の一致でクリア&スタート
        st.b    r11,          TMC0[r0]          -- タイマ･モード･コントロール･レジスタ0の設定
        mov     0x06,         r11
        st.b    r11,          TMIC00[r0]        -- タイマ割り込み許可(レベル6)
        ei
init_lcd:
```

```
        movhi   hi1(lcd_addr), r0, r11          -- r11:=LCDアドレス
        movea   lo(lcd_addr), r11, r11
        movhi   hi1(tm0_num), r0, r12           -- r12:=タイマ0カウンタ･アドレス
        movea   lo(tm0_num), r12, r12
        movea   0x30,   r0,   r13               -- r13:=0x30
        st.b    r13,          0[r11]            -- ファンクション・セット 8ビット 1行 5*7
        st.w    r0,           0[r12]            -- タイマ0カウンタ=0
        movea   0x29,   r0,   r13               -- r13:=0x29=41
wait_1:
        ld.w    0[r12],       r14               -- r14:=タイマ0カウンタ
        cmp     r14,          r13               -- タイマ0カウンタが41に達したかのチェック
        bp      wait_1                          -- 4.1ms以上待つ
        st.b    r13,          0[r11]            -- ファンクション・セット 8ビット 1行 5*7
        st.w    r0,           0[r12]            -- タイマ0カウンタ=0
        mov     0x2,          r13               -- r13:=0x2
wait_2:
        ld.w    0[r12],       r14               -- r14:=タイマ0カウンタ
        cmp     r14,          r13               -- タイマ0カウンタが2に達したかのチェック
        bp      wait_2                          -- 100us以上待つ
        st.b    r13,          0[r11]            -- ファンクション・セット 8ビット 1行 5*7
        jarl    bf_chk,       r29               -- LCDビジィ・チェック
        movea   0x38,   r0,   r13
        st.b    r13,          0[r11]            -- ファンクション・セット 8ビット 2行 5*7
        jarl    bf_chk,       r29               -- LCDビジィ・チェック
        mov     0x01,         r13
        st.b    r13,          0[r11]            -- 表示クリア,カーソルをホーム位置へ
        jarl    bf_chk,       r29               -- LCDビジィ・チェック
        mov     0x06,         r13
        st.b    r13,          0[r11]            -- カーソルはインクリメント，表示シフトしない
        jarl    bf_chk,       r29               -- LCDビジィ・チェック
        mov     0x0C,         r13
        st.b    r13,          0[r11]            -- 表示をON，カーソルをOFF

init_wt:
        ## 時計用タイマ･モード･コントロール･レジスタの設定
        movea   0x83,   r0,   r11
        st.b    r11,          WTM[r0]           -- 時計用タイマ・スタート,動作許可
                                                -- カウント・クロックをサブクロックに選択
        mov     0x07,         r11               -- r10:=0x07
        st.b    r11,          WTIC[r0]          -- 時計用タイマ割り込み許可(レベル7)
fv_loop:
        br      fv_loop

bf_chk:
        movhi   hi1(lcd_addr), r0, r27          -- r27:=LCDアドレス
        movea   lo(lcd_addr), r27, r27
        movea   0x80,   r0,   r26               -- r26:=0x80
bf_chk0:
        ld.b    0[r27],       r28
        andi    0x80,   r28,  r28
        cmp     r26,          r28
        bz      bf_chk0
        jmp     [r29]

_int_wt:
        ## EIPC,EIPSWのスタックへの保存
        add     -12,          r3
        st.w    r1,           8[r3]             -- r1の退避
        stsr    0,            r21               -- EIPC の退避
        st.w    r21,          4[r3]
        stsr    1,            r21
        st.w    r21,          0[r3]             -- EIPSWの退避
        ei                                      -- 多重割り込みを可能にする

wt_chk:
        movhi   hi1(flg_wt), r0, r21            -- r21:=1秒経過フラグ
        movea   lo(flg_wt), r21, r21
        not1    0,            0[r21]            -- flg_wtを反転
        tst1    0,            0[r21]            -- flg_wtが1だったら
```

```
        bnz     wt_exit                         -- なにもせずスキップする

#       mov     0x800000,     r21
        movhi   0x200000 >> 16, r0, r21
#       mov     0x100000,     r21
#       mov     0x80000,      r21
lp1:
        add     -1,           r21
        bnz     lp1


wt_data_set:
        movhi   hi1(wt_num), r0, r21            -- r21:=時計データ
        movea   lo(wt_num), r21, r21
        ld.w    0[r21],       r22               -- wt_numレジスタにロード
        addi    1,    r22,    r22               -- wt_numインクリメント
        st.w    r22,          0[r21]            -- wt_numメモリにストア
        ## r22 を時計の形にフォーマットしてメモリに格納
        movhi   hi1(tm_buf), r0, r21            -- r21:=時計表示データ
        movea   lo(tm_buf), r21, r21
        mov     r22,          r7                -- r7=r22
        movea   60*60*10, r0, r6
        andi    0xffff,   r6, r6                -- r6=60*60*10
        jarl    ___div      , lp                -- r6=r7/(60*60*10)
        add     0x30,         r6                -- r6=0x30+r6
        st.b    r6,           0[r21]            -- x*10時間
        mov     r22,          r23               -- r23=r22
        movea   60*60,    r0, r6                -- r6=60*60
        divh    r6,           r23               -- r23=r23/(60*60)
        mov     r23,          r24               -- r24=r23
        mov     10,           r7                -- r7=10
        divh    r7,           r24               -- r24=r24/10
        mulh    10,           r24               -- r24=r24*10
        sub     r24,          r23               -- r23=r23-r24
        add     0x30,         r23               -- r23=0x30+r23
        st.b    r23,          1[r21]            -- x時間
        mov     r22,          r23               -- r23=r22
        divh    r6,           r23               -- r23=r23/(60*60)
        mulh    60*60,        r23               -- r23=r23*(60*60)
        mov     r22,          r24               -- r24=r22
        sub     r23,          r24               -- r24=r24-r23
        movea   60,       r0, r6                -- r6=60
        divh    r6,           r24               -- r24=r24/60
        mov     r24,          r23               -- r23=r24
        divh    r7,           r23               -- r23=r23/10
        add     0x30,         r23               -- r23=0x30+r23
        st.b    r23,          3[r21]            -- x*10分
        mov     r24,          r23               -- r23=r24
        divh    r7,           r24               -- r24=r24/10
        mulh    10,           r24               -- r24=r24*10
        sub     r24,          r23               -- r23=r23-r24
        add     0x30,         r23               -- r23=0x30+r23
        st.b    r23,          4[r21]            -- x分
        mov     r22,          r23               -- r23=r22
        divh    r6,           r23               -- r23=r23/60
        mulh    60,           r23               -- r23=r23*60
        mov     r22,          r24               -- r24=r22
        sub     r23,          r24               -- r24=r24-r23
        mov     r24,          r23               -- r23=r24
        divh    r7,           r23               -- r23=r23/10
        add     0x30,         r23               -- r23=0x30+r23
        st.b    r23,          6[r21]            -- x*10秒
        mov     r24,          r23               -- r23=r24
        divh    r7,           r24               -- r24=r24/10
        mulh    10,           r24               -- r24=r24*10
        sub     r24,          r23               -- r23=r23-r24
        add     0x30,         r23               -- r23=0x30+r23
        st.b    r23,          7[r21]            -- x秒
        movhi   hi1(disp_flg), r0, r21          -- r21:=時計LCD表示状態フラグ
```

```
        movea   lo(disp_flg), r21, r21
        set1    0,          0[r21]          -- disp_flgのオン
wt_exit:
        ## EIPC,EIPSWのスタックへからの復帰
        di                                  -- 割り込み禁止
        ld.w    0[r3],      r21
        ldsr    r21,        1               -- EIPSWの復帰
        ld.w    4[r3],      r21
        ldsr    r21,        0               -- EIPCの復帰
        ld.w    8[r3],      r1              -- r1の復帰
        add     12,         r3
        reti

_int_tm00:
        mov     r1,         r6              -- r1の退避
tm_countup:
        movhi   hi1(tm0_num), r0, r16       -- r16:=タイマ0カウンタ
        movea   lo(tm0_num), r16, r16
        ld.w    0[r16],     r17             -- tm0_numをレジスタにロード
        addi    1,    r17,  r17             -- tm0_numインクリメント
        st.w    r17,        0[r16]          -- tm0_numをメモリにストア
tm_chk_flg:
        movhi   hi1(disp_flg), r0, r16      -- r16:=時計LCD表示状態フラグ
        movea   lo(disp_flg), r16, r16
        tst1    0,          0[r16]          -- disp_flgのチェック
        bz      tm_led                      -- disp_flgが0だったらジャンプ
tm_disp_lcd:
        movhi   hi1(col_ctr), r0, r16       -- r16:=時計表示データの表示カラム数
        movea   lo(col_ctr), r16, r16
        ld.b    0[r16],     r17             -- col_ctrをレジスタにロード
        mov     0x07,       r18             -- r18:=7
        cmp     r17,        r18             -- col_ctrが7以上だったら
        bn      tm_cursol_home              -- ジャンプ

        movhi   hi1(tm_buf), r0, r18        -- r17:=時計表示データ
        movea   lo(tm_buf), r18, r18
        add     r17,        r18
        ld.b    0[r18],     r20             -- r20:=表示キャラクタ

        movhi   hi1(lcd_addr), r0, r19      -- r19:=LCD アドレス
        movea   lo(lcd_addr), r19, r19
        st.b    r20,        2[r19]
        addi    1,    r17,  r17             -- col_ctrインクリメント
        st.b    r17,        0[r16]          -- col_ctrのメモリへのストア
        br      tm_led                      -- LED 表示処理へ
tm_cursol_home:
        movhi   hi1(lcd_addr), r0, r19      -- r19:=LCD アドレス
        movea   lo(lcd_addr), r19, r19
        mov     0x02,       r20
        st.b    r20,        0[r19]          -- LCDホーム・ポジション
        movhi   hi1(disp_flg), r0, r16      -- r16:=時計LCD表示状態フラグ
        movea   lo(disp_flg), r16, r16
        clr1    0,          0[r16]          -- disp_flgのオフ
        movhi   hi1(col_ctr), r0, r16       -- r16:=時計表示データの表示カラム数
        movea   lo(col_ctr), r16, r16
        st.b    r0,         0[r16]          -- col_ctr = 0
        br      tm_led                      -- LED 表示処理へ
tm_led:
        ## LED は25.6msのインターバルでインクリメント点灯を行う
        movhi   hi1(led_ivl), r0, r16       -- r16:=LED 表示間隔アドレス
        movea   lo(led_ivl), r16, r16
        ld.w    0[r16],     r17             -- r17:=LED 表示間隔
        addi    0x01,  r17, r17             -- LED 表示間隔インクリメント
        st.w    r17,        0[r16]          -- インクリメントしたデータをメモリにストア
        mov     r17,        r16             -- r16:=r17
        movea   0xff,  r0,  r18             -- r18:=0xff
        and     r18,        r17             -- r17の下位8ビットが0でなければ
        bnz     tm_exit                     -- ジャンプする
led_disp:
```

```
        shr     8,          r16             -- r16の8ビット目から15ビット目を
        and     0xff,       r16             -- 算出する
        mov     r16,        r17             -- r17:=r16
        and     0x0f,       r16             -- 表示データの下位4ビットを
        shl     4,          r16             -- ポート3の上位4ビットに
        st.b    r16,        P3[r0]          -- 出力する
        and     0xf0,       r17             -- 表示データの上位4ビットを
        shr     4,          r17             -- ポート11の下位4ビットに
        st.b    r17,        P11[r0]         -- 出力する
tm_exit:
        mov     r6,         r1              -- r1の復帰
        reti
```

4. 5　TB-V850/SA1の回路図

TB-V850/SA1の回路図を次に示します。

(1) CPUとトグル・スイッチ

V850/SA1，発振器，振動子，電源バックアップ回路，シリアルEEPROM，トグル・スイッチ×8などで構成されています。

・NM24C02LM：2 Kビット・シリアルEEPROM（I^2Cバス・インタフェース）

(2) デバイス制御回路

FMP7128SLC84-10，高速ページDRAMなどで構成されています。

・FMP7128SLC84-10：FPGA（Field Programmable Gate Array）
CPUのバス・インタフェース関連の信号を入力して，メモリの制御信号を作成しています。FPGAのデータは，コネクタ（CN1）を経由して書き込まれます。

(3) メモリ

ROM, SRAM, LCD用コネクタなどで構成されています。

・74FCT16245 ：16ビット双方向バッファ
・74FCT16373 ：16ビット・トランスペアレント・ラッチ
・74FCT244　 ：8ビット・バッファ

注意　LCD（L1652）がCN2コネクタを経由して接続されています。

(4) I/Oインタフェース

電源監視回路，フラッシュ・ライタ・インタフェース，RS-232-Cインタフェース，LED×8などで構成されています。

・MAX232：RS-232-Cドライバ／レシーバ
・TL7700 ：電源監視デバイス

注意　V850/SA1専用フラッシュ・ライタは，CN7コネクタに接続されています。

(5) アナログ入出力

スピーカ・インタフェース，可変抵抗，DCモータ・インタフェースなどで構成されています。

・BLA8-103J：ラダー抵抗（R-2R回路）

注意1．スピーカは，CN5コネクタに接続されています。
2．DCモータは，CN6コネクタに接続されています。

図4－4　CPUとトグル・スイッチ

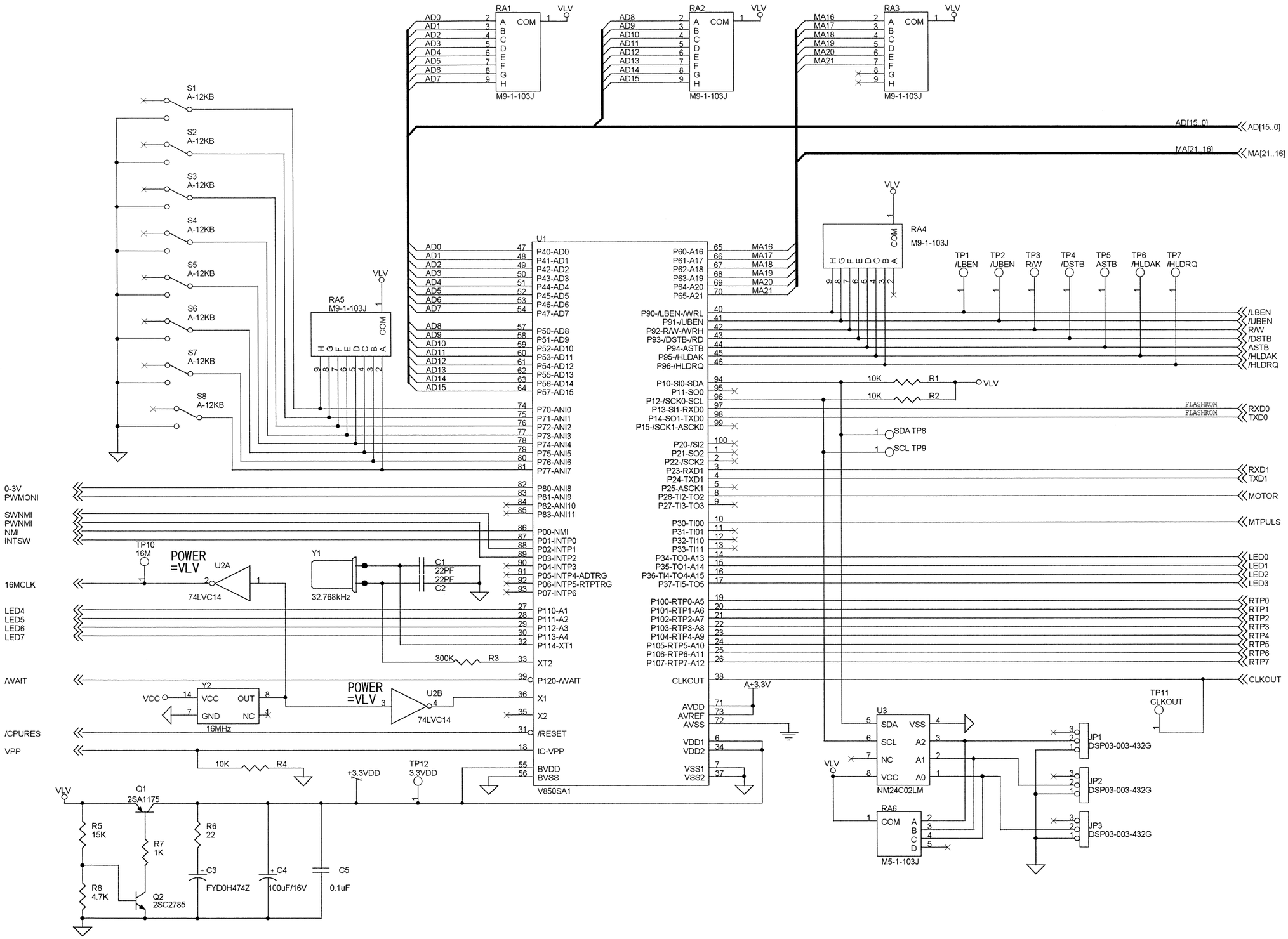


図4－5　デバイス制御回路

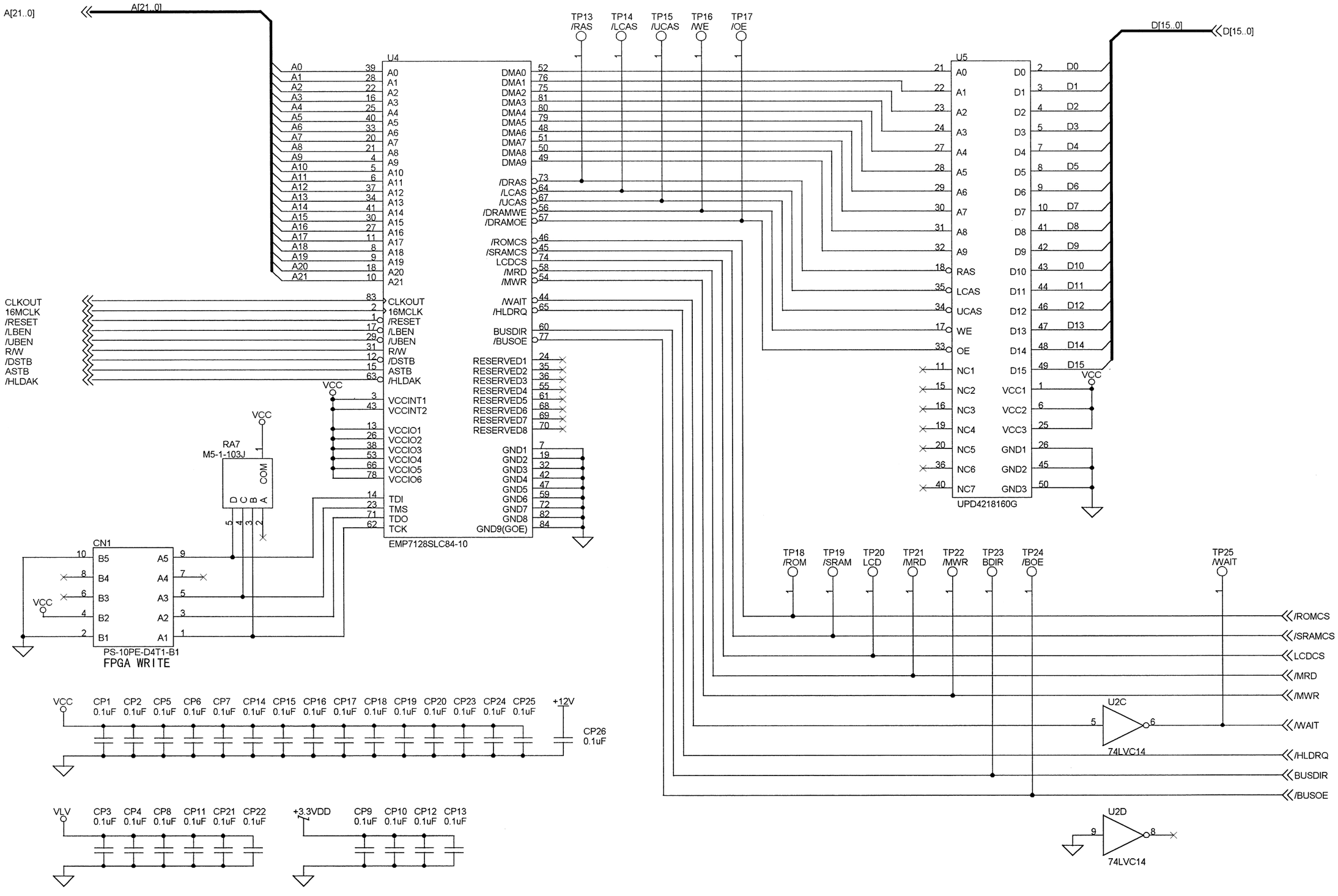


図4−6　メモリ

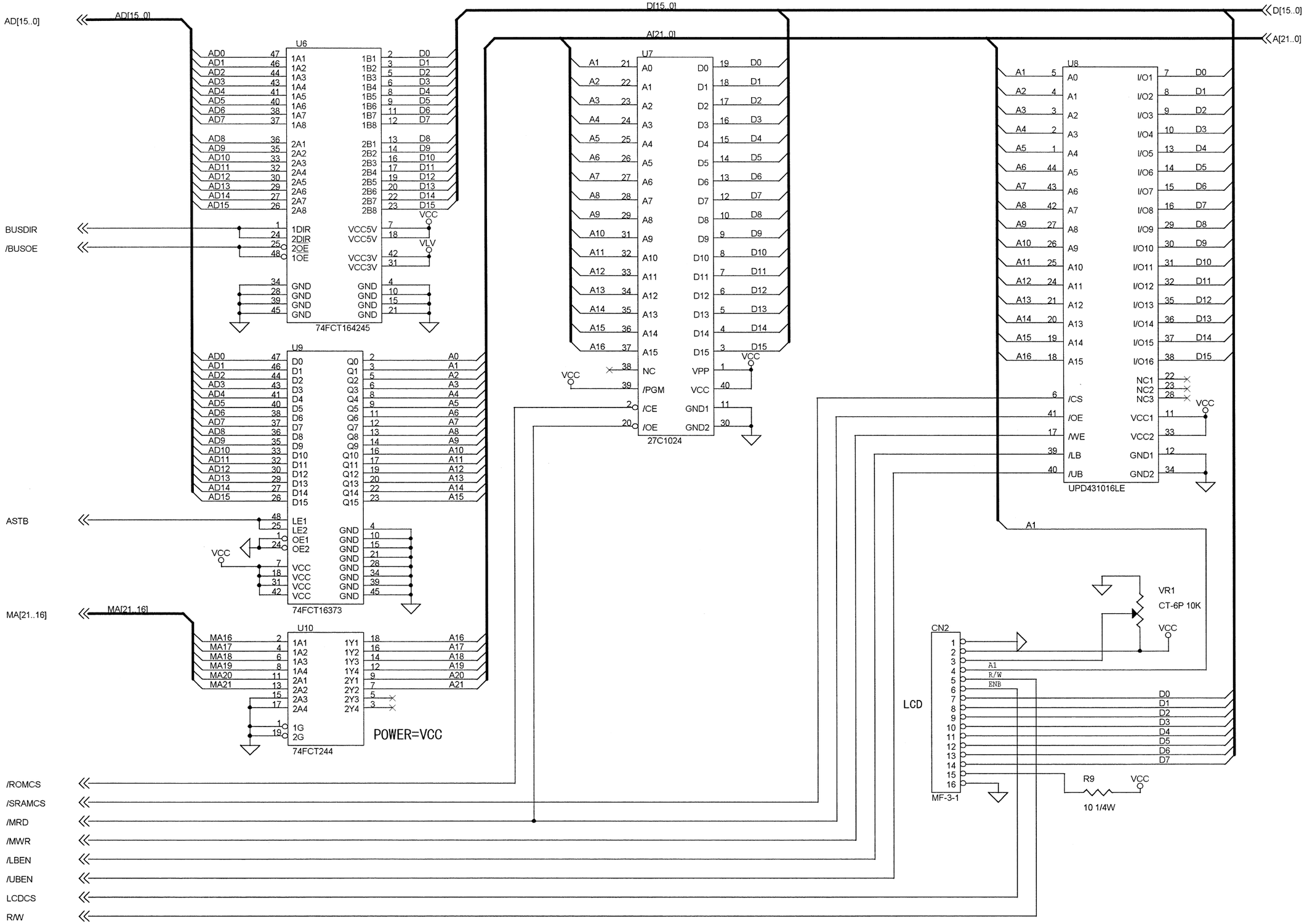

AD[15..0]
D[15..0]
A[21..0]
U6
74FCT164245
BUSDIR
/BUSOE
U7
27C1024
U8
UPD431016LE
U9
74FCT16373
ASTB
MA[21..16]
U10
74FCT244
POWER=VCC
VR1
CT-6P 10K
CN2
MF-3-1
LCD
R9
10 1/4W
/ROMCS
/SRAMCS
/MRD
/MWR
/LBEN
/UBEN
LCDCS
R/W

図4－7　I/Oインタフェース

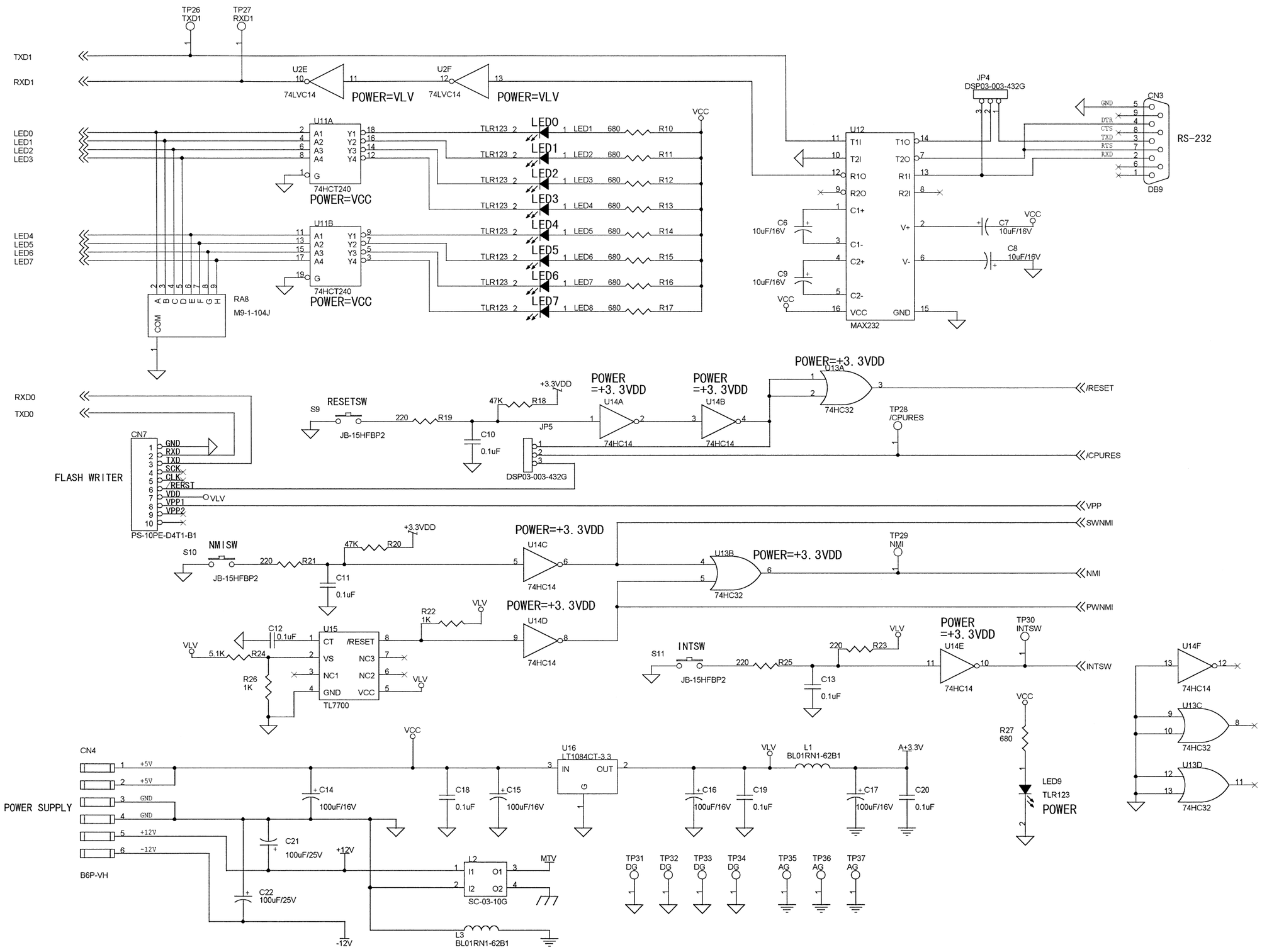
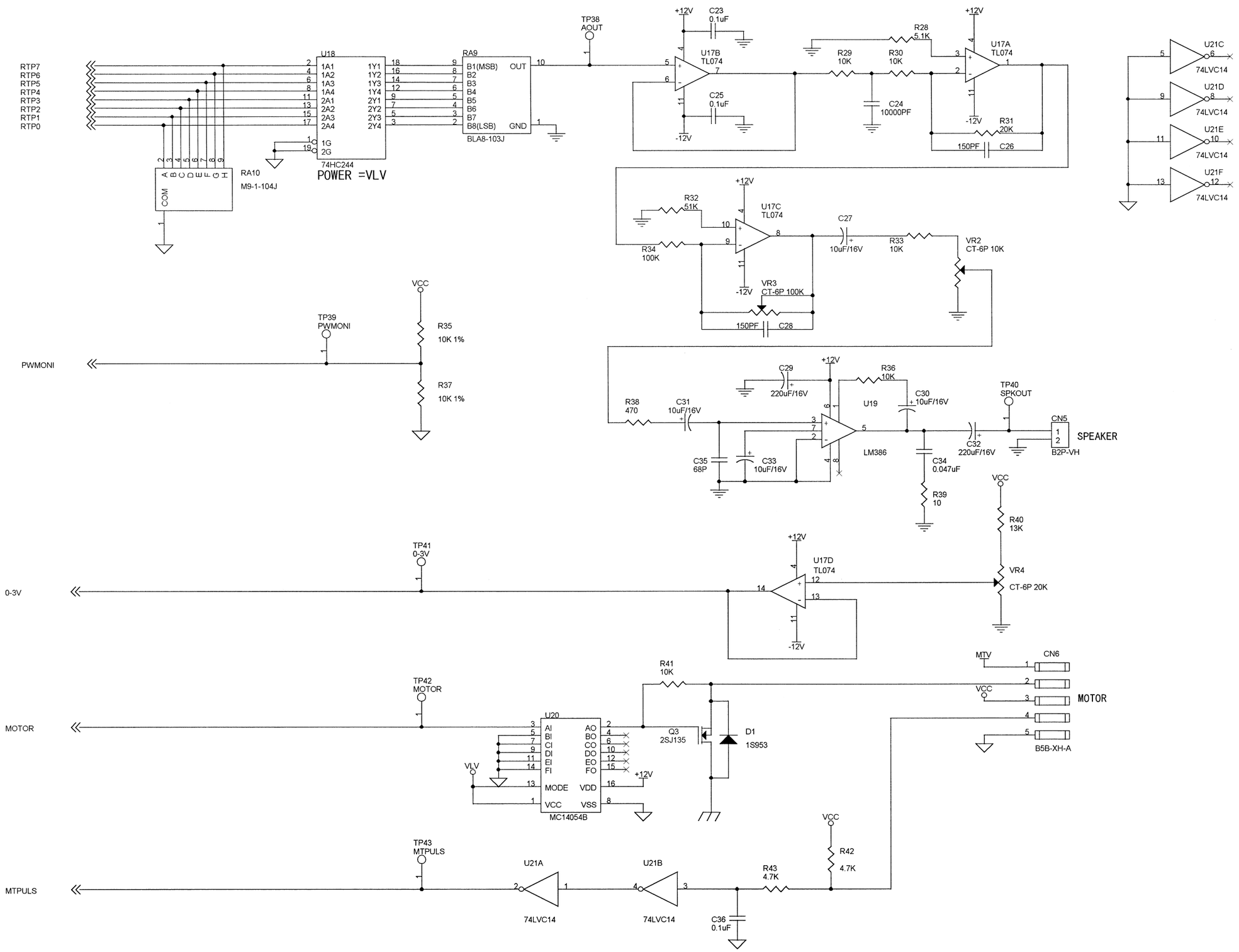


図4－8　アナログ入出力


RTP7
RTP6
RTP5
RTP4
RTP3
RTP2
RTP1
RTP0
U18
74HC244
RA10
MS-1-104J
COM
RA9
B1(MSB)
B2
B3
B4
B5
B6
B7
B8(LSB)
OUT
GND
BLA8-103J
POWER =VLV
TP38
AOUT
U17B
TL074
U17A
TL074
U17C
TL074
U17D
TL074
R28
5.1K
R29
10K
R30
10K
R31
20K
R32
51K
R33
10K
R34
100K
R35
10K 1%
R36
10K
R37
10K 1%
R38
470
R39
10
R40
13K
R41
10K
R42
4.7K
R43
4.7K
C23
0.1uF
C24
10000PF
C25
0.1uF
C26
150PF
C27
10uF/16V
C28
150PF
C29
220uF/16V
C30
10uF/16V
C31
10uF/16V
C32
220uF/16V
C33
10uF/16V
C34
0.047uF
C35
68P
C36
0.1uF
VR2
CT-6P 10K
VR3
CT-6P 100K
VR4
CT-6P 20K
+12V
-12V
VCC
U21C
U21D
U21E
U21F
74LVC14
TP39
PWMONI
PWMONI
TP40
SPKOUT
U19
LM386
CN5
SPEAKER
B2P-VH
TP41
0-3V
0-3V
TP42
MOTOR
MOTOR
U20
MC14054B
VLV
Q3
2SJ135
D1
1S953
MTV
CN6
B5B-XH-A
TP43
MTPULS
MTPULS
U21A
U21B
74LVC14

付録A　総合索引

A. 1　50音で始まる語句の索引

A. 2　数字，アルファベットで始まる語句の索引

★

付録B　改版履歴

これまでの改版履歴を次に示します。なお，適用箇所は各版での章を示します。

版数	内　　容	適用箇所
第2版	μPD703014B, 703014BY, 703015B, 703015BY, 70F3015B, 70F3015BYを追加	全般
	表1－1　V850/SA1の製品一覧を追加	第1章　V850/SA1の概要
	1. 2　特　徴　最小命令実行時間に記述を追加	
	1. 6. 2（2）バス・コントロール・ユニット（BCU）記述を削除	
	1. 8　CPUレジスタ・セット　r2の用途変更	
	1. 8. 1（1）汎用レジスタ　r2の用途および動作変更	
	1. 13　周辺I/Oレジスタ　フラッシュ・メモリ・プログラミング・モード・コントロール・レジスタ（FLPMC）を追加	
	1. 13　周辺I/Oレジスタ　注を追加	
	2. 1. 1　BV_{DD}端子の接続　を削除	第2章　バス・インタフェース接続回路例

〔メ　モ〕

お問い合わせ先

【技術的なお問い合わせ先】

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

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

【営業関係お問い合わせ先】

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

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

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

【資料の請求先】

上記営業関係お問い合わせ先またはNEC特約店へお申しつけください。

【NECエレクトロンデバイス　ホームページ】

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

アンケート記入のお願い

お手数ですが，このドキュメントに対するご意見をお寄せください。今後のドキュメント作成の参考にさせていただきます。

［ドキュメント名］V850/SA1　アプリケーション・ノート

（U13851JJ2V0AN00（第2版））

［お名前など］（さしつかえのない範囲で）

御社名（学校名，その他）　（　　）
ご住所　（　　）
お電話番号　（　　）
お仕事の内容　（　　）
お名前　（　　）

1．ご評価（各欄に○をご記入ください）

項　　目	大変良い	良　い	普　通	悪　い	大変悪い
全体の構成					
説明内容					
用語解説					
調べやすさ					
デザイン，字の大きさなど					
その他（　　）					
（　　）					

2．わかりやすい所（第　章，第　章，第　章，第　章，その他　　）

理由［　　］

3．わかりにくい所（第　章，第　章，第　章，第　章，その他　　）

理由［　　］

4．ご意見，ご要望

5．このドキュメントをお届けしたのは

NEC販売員，特約店販売員，その他（　　）

ご協力ありがとうございました。
下記あてにFAXで送信いただくか，最寄りの販売員にコピーをお渡しください。

日本電気(株) NEC エレクトロンデバイス
半導体テクニカルホットライン
FAX：（044）435-9608

2000.6

キリトリ