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

V850E/MA1™

32ビット・シングルチップ・マイクロコンピュータ

ハードウエア編

μPD703103A
μPD703105A
μPD703106A
μPD703106A(A)
μPD703107A
μPD703107A(A)
μPD70F3107A
μPD70F3107A(A)

資料番号 U15179JJ2V0AN00（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。

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　：μPD703103A, 70F3107A, 70F3107A(A)

ユーザ判定品：μPD703105A, 703106A, 703106A(A), 703107A, 703107A(A)

- **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	・次の製品を削除 μPD703103, 703105, 703106, 703107, 70F3107 ・次の製品を追加 μPD703103A, 703105A, 703106A, 703106A(A), 703107A, 703107A(A), 70F3107A, 70F3107A(A)
p.20	1. 4　**オーダ情報**　記述変更
p.21	1. 5　**端子接続図**（Top View）　を追加
p.24	1. 6　**内部ブロック図**　を追加
p.56, 58, 59	2. 6　SDRAM**との接続**　V850E/MA1**のアドレス端子を変更**
p.101	**図**4－5　**システム・ウエイト・コントロール・レジスタ**（VSWC）**の設定**　**を変更**
p.102	4. 2. 3　**シングルチップ・モード**0**での動作例**（STEP1）　**【プログラム例】**　VSWC**レジスタの設定値を変更**
p.181	**図**4－35　TU-V850E/MA1　SDRAM**と接続する**V850E/MA1**のアドレス端子を変更**

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，V850E/MA1の機能を理解し，それらを使用した応用システムを設計するユーザを対象とします。
対象製品は次のようになります。

- 標準品：μPD703103A, 703105A, 703106A, 703107A, 70F3107A
- 特別品：μPD703106A(A)[注], 703107A(A)[注], 70F3107A(A)

注　開発中

目　　的　このアプリケーション・ノートではV850E/MA1を用いたシステムの例として「V850E/MA1トレーニング・ユニット（TU-V850E/MA1）」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。

- ●V850E/MA1の概要
- ●バス・インタフェース接続回路例（1）（直結可能メモリの接続例）
- ●バス・インタフェース接続回路例（2）（付加回路を使用しての接続例）
- ●アプリケーション例

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

注意1. **このマニュアルの使用例は，一般電子機器用の『標準』品質水準品用に作成してあります。『特別』品質水準を要求する用途にこのマニュアル中の使用例を使用する場合は，実際に使用する各部品および回路について，その品質水準についてご検討のうえご使用ください。**

2. **特別品のマニュアルとして使用する場合には，次のように読み替えてください。**

μPD703106A　→　μPD703106A(A)
μPD703107A　→　μPD703107A(A)
μPD70F3107A　→　μPD70F3107A(A)

V850E/MA1の電気的特性を知りたいとき
→別冊のデータ・シートを参照してください。

V850E/MA1のハードウエア機能を知りたいとき
→別冊のV850E/MA1　**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。

V850E/MA1の命令機能を知りたいとき
→別冊のV850E1　**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**）**または**

／xxx（**信号名称の前に“／”記号**）

メモリ･マップのアドレス：**上部－上位，下部－下位**

注　：**本文中に付けた注の説明**

注意　：**気を付けて読んでいただきたい内容**

備考　：**本文の補足説明**

数の表記　：2**進数**　…　xxxx**または**xxxxB

10**進数**　…　xxxx

16**進数**　…　xxxxH

2**のべき数を示す接頭語（アドレス空間，メモリ容量）**　：K（**キロ**）…2^{10} = 1024

M（**メガ**）…2^{20} = 1024^2

G（**ギガ**）…2^{30} = 1024^3

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

V850E/MA1**に関する資料**

資　料　名	資料番号
V850E1　**ユーザーズ・マニュアル　アーキテクチャ編**	U14559J
V850E/MA1　**ユーザーズ・マニュアル　ハードウエア編**	U14359J
μPD70F3107A, 70F3107A(A)　**データ・シート**	U15846J
μPD703103A, 703105A, 703106A, 703106A(A), 703107A, 703107A(A)　**データ・シート**	U15578J
V850E/MA1　**アプリケーション・ノート　ハードウエア編**	**このマニュアル**
V850**シリーズ**^{TM}　**ユーザーズ・マニュアル　フラッシュ・メモリ・セルフ・プログラミング**	U15673J

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-V850E-MC, IE-V850E-MC-A（インサーキット・エミュレータ）		U14487J
IE-703107-MC-EM1（インサーキット・エミュレータ・オプション・ボード）		U14481J
CA850（Ver.2.30以上） （Cコンパイラ・パッケージ）	操作編	U14568J
	C言語編	U14566J
	プロジェクト・マネージャ編	U14569J
	アセンブリ言語編	U14567J
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
RX-NET（TCP/IPライブラリ）		U15083J
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-FP3（フラッシュ・メモリ・プログラマ）		U13502J
PG-FP4（フラッシュ・メモリ・プログラマ）		U15260J

目　　次

図の目次（1/3）

図の目次（2/3）

図の目次（3/3）

表の目次

第1章　V850E/MA1の概要

V850E/MA1は，NECのシングルチップ・マイクロコンピュータ「V850シリーズ」の1製品です。この章では，V850E/MA1の概要を簡単に説明します。

1.1 概　　説

V850E/MA1は，システム・オン・チップ時代のシステムLSIの核となるCPUコアとして新たに開発したASIC用32ビットRISC型CPUコア「V850E1 CPU」を搭載した32ビット・シングルチップ・マイクロコンピュータです。ROM，RAM，および，各種メモリ・コントローラ，DMAコントローラ，リアルタイム・パルス・ユニット，シリアル・インタフェース，A/Dコンバータなどの周辺機能を内蔵し，大容量データ処理と高度なリアルタイム制御を実現します。

(1)「V850E1 CPU」搭載

「V850E1 CPU」は，V850シリーズ搭載のCPUコア「V850 CPU」に対し，外部バス・インタフェースの性能を強化し，C言語のswitch文処理，テーブル・ルックアップの分岐，スタック・フレームの生成／削除，データ変換など，主に高級言語に対応した命令を追加することにより，制御系だけではなく，データ処理系にも対応したCPUコアです。

なお，命令コードは，V850 CPUに対して，オブジェクト・コード・レベルでの上位互換性を持たせているため，V850 CPU搭載システムのソフトウエア資産をそのまま使用できます。

(2) 外部メモリ・インタフェース機能

外部メモリ・インタフェースとして，セパレート構成のアドレス・バス（26ビット），データ・バス（16ビット）とSDRAM，ROM用インタフェースのほかEDO DRAM，ページROMなどに直結できる各種メモリ・コントローラを内蔵しているので，システム性能を上げるとともにアプリケーション・システムの部品点数を削減できます。

また，DMAコントローラにより，外部メモリ間の転送と並行してCPU内部の演算やデータ転送を行えるため，画像データや音声データなどの大容量データ処理が可能となるうえ，内蔵のROMとRAMを使用した高速な命令実行により，モータ制御，通信制御などのリアルタイム制御も同時に実現できます。

★ **(3) フラッシュ・メモリ内蔵（μPD70F3107A）**

フラッシュ・メモリ内蔵品（μPD70F3107A）は，高速アクセス可能なフラッシュ・メモリを内蔵しており，アプリケーション・システム上にV850E/MA1を実装したままプログラムの書き換えが行えるため，システム開発期間の短縮が実現できます。また，システム出荷後のメインテナンス性を飛躍的に向上させることができます。

(4) 充実したミドルウエア，開発環境製品群

V850E/MA1はJPEG，JBIG，MH/MR/MMRなどのミドルウエアを高速実行できます。また，音声認識，音声合成などの処理を実現するミドルウエアも用意されているので，これらのミドルウエアと組み合わせることにより，マルチメディア・システムを容易に実現できます。

また，最適化Cコンパイラ，ディバッガ，インサーキット・エミュレータ，シミュレータ，システム・パフォーマンス・アナライザなどの統合された開発環境も用意しています。

1.2 特　　徴

○命令数　　83

○最小命令実行時間　　20 ns（内部50 MHz動作時）

○汎用レジスタ　　32ビット×32本

○命令セット　　V850E1 CPU
符号付き乗算（16ビット×16ビット→32ビット，または32ビット×32ビット→64ビット）：1-2クロック
飽和演算命令（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
符号付きロード命令

○メモリ空間　　256 Mバイト・リニア・アドレス空間（プログラム／データ共有）
チップ・セレクト出力機能：8空間
メモリ・ブロック分割機能：2 M, 4 M, 8 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／データ分離型バス）
16/8ビット・バス・サイジング機能
バス・ホールド機能
外部ウエイト機能
アドレス・セットアップ・ウエイト機能
エンディアン制御機能

★ ○内蔵メモリ

製品名	内蔵ROM	内蔵RAM
μPD703103A	なし	4 Kバイト
μPD703105A	128 Kバイト（マスクROM）	4 Kバイト
μPD703106A	128 Kバイト（マスクROM）	10 Kバイト
μPD703107A	256 Kバイト（マスクROM）	10 Kバイト
μPD70F3107A	256 Kバイト（フラッシュ・メモリ）	10 Kバイト

○割り込み／例外　　外部割り込み　：25本（NMI含む）
内部割り込み　：33要因
例外　：1要因
8レベルの優先順位指定可能

○メモリ・アクセス制御

DRAMコントローラ（EDO DRAM, SDRAMに対応）

ページROMコントローラ

○DMAコントローラ

4チャネル構成

転送単位　　：8ビット／16ビット

最大転送回数：65536（2^{16}）回

転送タイプ　：フライバイ（1サイクル）転送／2サイクル転送

転送モード　：シングル転送／シングルステップ転送／ブロック転送

転送対象　　：メモリ←→メモリ，メモリ←→I/O

転送要求　　：外部要求／内蔵周辺I/O／ソフトウエア

DMA転送終了（ターミナル・カウント）出力信号

ネクスト・アドレス設定機能

○I/Oライン　　入力ポート　：9

入出力ポート：106

○リアルタイム・パルス・ユニット

16ビット・タイマ／イベント・カウンタ：4ch

16ビット・タイマ：4本

16ビット・キャプチャ／コンペア・レジスタ：8本

16ビット・インターバル・タイマ：4ch

○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）

CSI/UART　：2ch

UART　　：1ch

CSI　　：1ch

○A/Dコンバータ　　10ビット分解能A/Dコンバータ：8ch

○PWM（Pulse Width Modulation）

8/9/10/12ビット分解能PWM：2ch

○クロック・ジェネレータ

PLLクロック・シンセサイザによる10逓倍機能

外部クロック入力による2分周機能

○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　　144ピン・プラスチックLQFP（ファインピッチ）（20×20）
　　161ピン・プラスチックFBGA（13×13）

○CMOS構造　　完全スタティック回路

1.3　応用分野

インクジェット・プリンタ，ファクシミリ，ディジタル・スチル・カメラ，DVDプレーヤ，ビデオ・プリンタ，PPC，情報家電など

★ 1.4　オーダ情報

品　名	パッケージ	品質水準
μPD703103AGJ-UEN	144ピン・プラスチックLQFP（ファインピッチ）（20×20）	標準（一般電子機器用）
μPD703105AGJ-xxx-UEN	〃	〃
μPD703106AGJ-xxx-UEN	〃	〃
μPD703107AGJ-xxx-UEN	〃	〃
μPD70F3107AGJ-UEN	〃	〃
μPD703106AF1-xxx-EN4	161ピン・プラスチックFBGA（13×13）	〃
μPD703107AF1-xxx-EN4	〃	〃
μPD70F3107AF1-EN4	〃	〃
μPD703106AGJ(A)-xxx-UEN[注]	144ピン・プラスチックLQFP（ファインピッチ）（20×20）	特別（高信頼度電子機器用）
μPD703107AGJ(A)-xxx-UEN[注]	〃	〃
μPD70F3107AGJ(A)-UEN	〃	〃

注　開発中

備考　xxxはROMコード番号です。

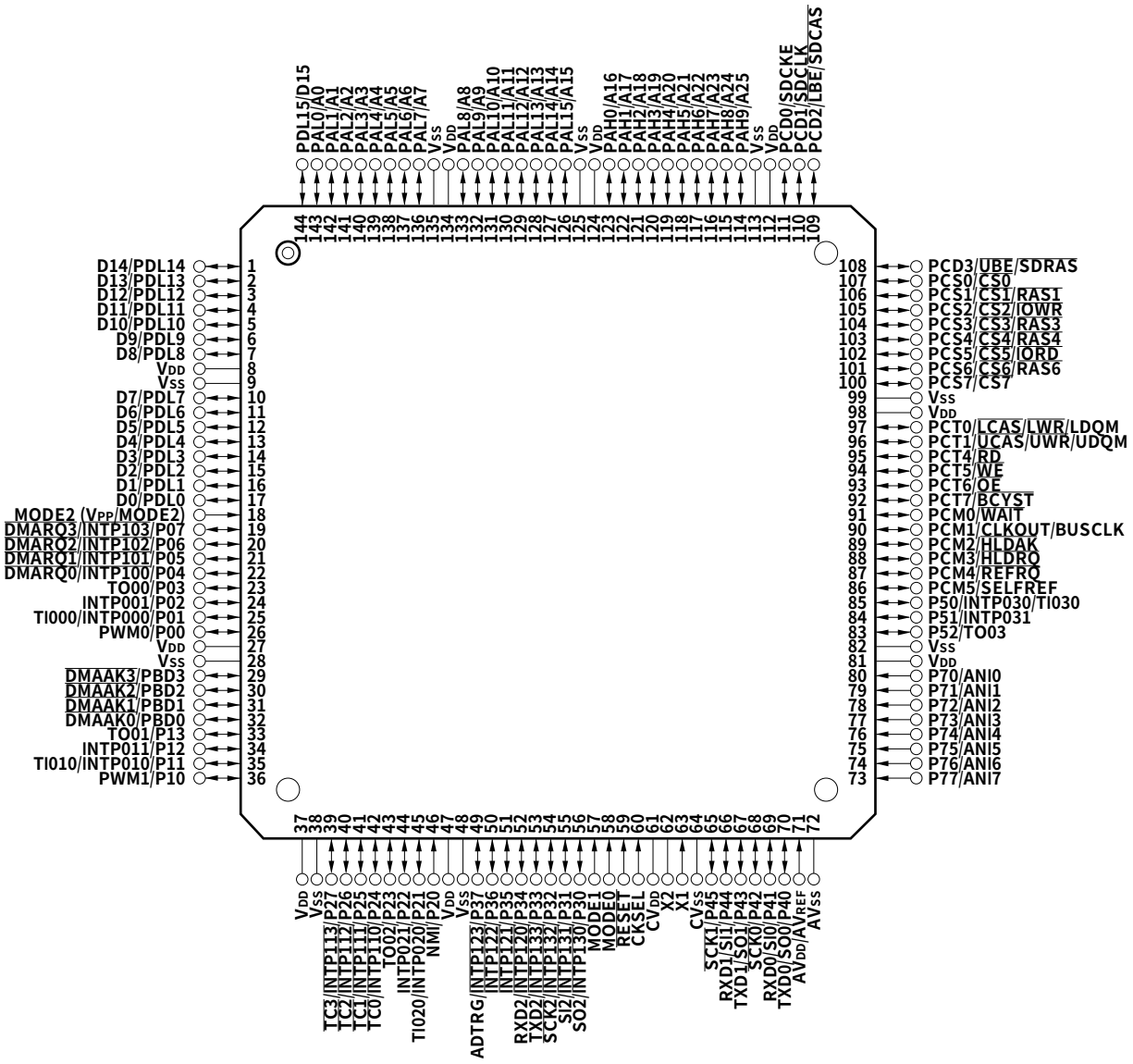
μPD703106A, 703107A, 70F3107Aとμ PD703106A(A), 703107A(A), 70F3107A(A)では，品質水準以外の相違はありません。

品質水準とその応用分野の詳細については当社発行の資料「NEC半導体デバイスの品質水準」（資料番号　C11531J）をご覧ください。

★ 1. 5　端子接続図（Top View）

・144ピン・プラスチックLQFP（ファインピッチ）（20×20）

μPD703103AGJ-UEN
μPD703105AGJ-xxx-UEN
μPD703106AGJ-xxx-UEN
μPD703107AGJ-xxx-UEN
μPD70F3107AGJ-UEN
μPD703106AGJ(A)-xxx-UEN
μPD703107AGJ(A)-xxx-UEN
μPD70F3107AGJ(A)-UEN



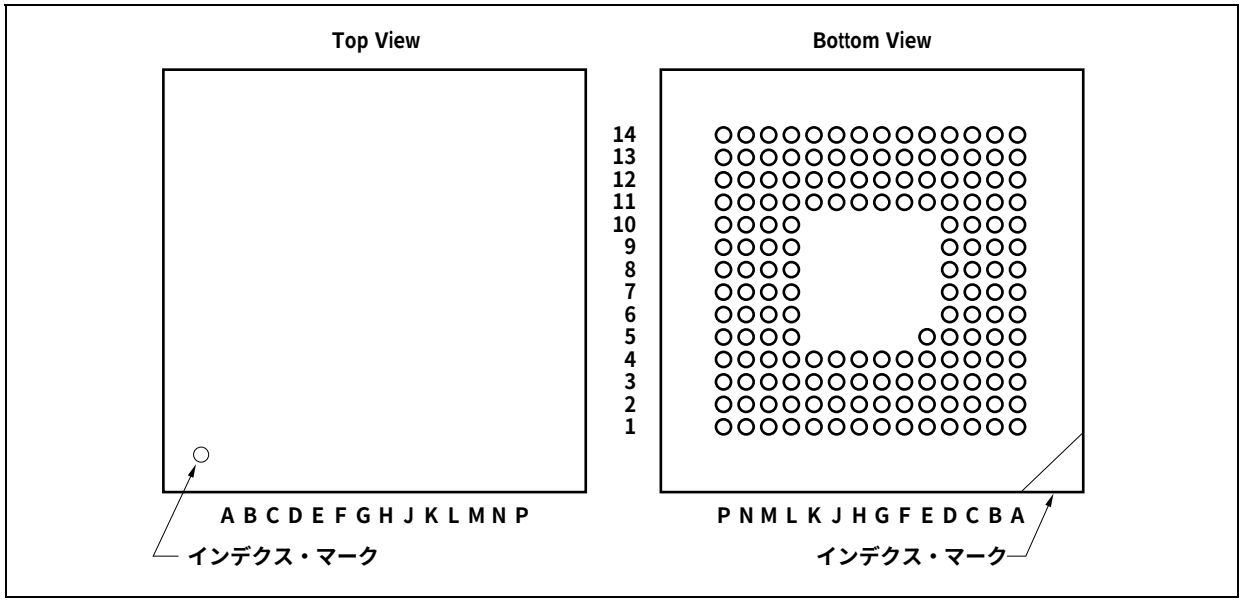
備考　（　）内はμPD70F3107Aでの端子名です。

• 161ピン・プラスチックFBGA（13×13）

μPD703106AF1-xxx-EN4

μPD703107AF1-xxx-EN4

μPD70F3107AF1-EN4



(1/2)

ピン番号	名　称	ピン番号	名　称	ピン番号	名　称
A1	−	B10	A21/PAH5	D5	A6/PAL6
A2	D15/PDL15	B11	A25/PAH9	D6	A10/PAL10
A3	A2/PAL2	B12	SDCLK/PCD1	D7	A14/PAL14
A4	A5/PAL5	B13	$\overline{CS1}$/$\overline{RAS1}$/PCS1	D8	A16/PAH0
A5	−	B14	−	D9	A20/PAH4
A6	A9/PAL9	C1	−	D10	A23/PAH7
A7	A12/PAL12	C2	D9/PDL9	D11	SDCKE/PCD0
A8	A15/PAL15	C3	D13/PDL13	D12	$\overline{CS0}$/PCS0
A9	A17/PAH1	C4	A1/PAL1	D13	$\overline{CS5}$/$\overline{IORD}$/PCS5
A10	−	C5	A7/PAL7	D14	−
A11	A24/PAH8	C6	V_{DD}	E1	D5/PDL5
A12	V_{DD}	C7	A11/PAL11	E2	D7/PDL7
A13	$\overline{LBE}$/$\overline{SDCAS}$/PCD2	C8	V_{DD}	E3	D8/PDL8
A14	$\overline{UBE}$/$\overline{SDRAS}$/PCD3	C9	A19/PAH3	E4	D11/PDL11
B1	−	C10	A22/PAH6	E5	−
B2	D12/PDL12	C11	V_{SS}	E11	$\overline{CS6}$/$\overline{RAS6}$/PCS6
B3	A0/PAL0	C12	$\overline{CS3}$/$\overline{RAS3}$/PCS3	E12	$\overline{CS4}$/$\overline{RAS4}$/PCS4
B4	A4/PAL4	C13	$\overline{CS2}$/$\overline{IOWR}$/PCS2	E13	$\overline{CS7}$/PCS7
B5	V_{SS}	C14	−	E14	V_{SS}
B6	A8/PAL8	D1	V_{SS}	F1	D2/PDL2
B7	A13/PAL13	D2	D10/PDL10	F2	D3/PDL3
B8	V_{SS}	D3	D14/PDL14	F3	D4/PDL4
B9	A18/PAH2	D4	A3/PAL3	F4	V_{DD}

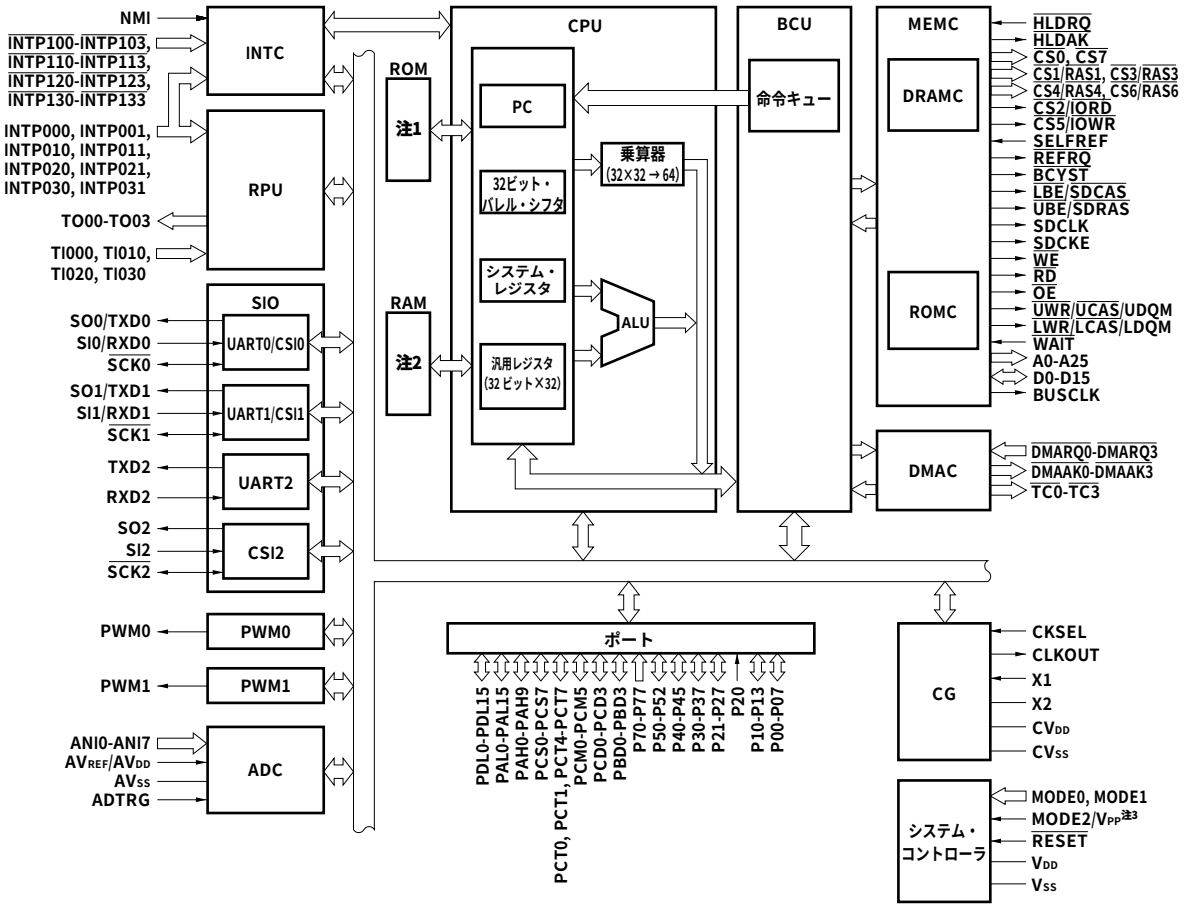
(2/2)

ピン番号	名　称	ピン番号	名　称	ピン番号	名　称
F11	$\overline{RD}$/PCT4	L6	V_{DD}	P5	−
F12	V_{DD}	L7	$\overline{INTP122}$/P36	P6	$\overline{INTP121}$/P35
F13	$\overline{LCAS}$/$\overline{LWR}$/LDQM/PCT0	L8	SI2/$\overline{INTP131}$/P31	P7	$\overline{SCK2}$/$\overline{INTP132}$/P32
F14	$\overline{UCAS}$/$\overline{UWR}$/UDQM/PCT1	L9	$\overline{RESET}$	P8	MODE1
G1	MODE2（MODE2/V_{PP}）	L10	TXD1/SO1/P43	P9	CV_{DD}
G2	$\overline{DMARQ3}$/$\overline{INTP103}$/P07	L11	ANI7/P77	P10	X1
G3	D0/PDL0	L12	ANI4/P74	P11	−
G4	D6/PDL6	L13	ANI3/P73	P12	RXD1/SI1/P44
G11	$\overline{WAIT}$/PCM0	L14	ANI2/P72	P13	RXD0/SI0/P41
G12	$\overline{WE}$/PCT5	M1	−	P14	−
G13	$\overline{BCYST}$/PCT7	M2	INTP011/P12		
G14	$\overline{OE}$/PCT6	M3	TO01/P13		
H1	$\overline{DMARQ2}$/$\overline{INTP102}$/P06	M4	$\overline{TC2}$/$\overline{INTP112}$/P26		
H2	$\overline{DMARQ1}$/$\overline{INTP101}$/P05	M5	TI020/INTP020/P21		
H3	$\overline{DMARQ0}$/$\overline{INTP100}$/P04	M6	V_{SS}		
H4	D1/PDL1	M7	RXD2/$\overline{INTP120}$/P34		
H11	$\overline{REFRQ}$/PCM4	M8	MODE0		
H12	$\overline{HLDRQ}$/PCM3	M9	CKSEL		
H13	$\overline{HLDAK}$/PCM2	M10	$\overline{SCK1}$/P45		
H14	CLKOUT/BUSCLK/PCM1	M11	TXD0/SO0/P40		
J1	TO00/P03	M12	ANI6/P76		
J2	TI000/INTP000/P01	M13	ANI5/P75		
J3	V_{DD}	M14	−		
J4	INTP001/P02	N1	−		
J11	TO03/P52	N2	PWM1/P10		
J12	TI030/INTP030/P50	N3	$\overline{TC3}$/$\overline{INTP113}$/P27		
J13	SELFREF/PCM5	N4	$\overline{TC0}$/$\overline{INTP110}$/P24		
J14	INTP031/P51	N5	NMI/P20		
K1	PWM0/P00	N6	ADTRG/$\overline{INTP123}$/P37		
K2	V_{SS}	N7	TXD2/$\overline{INTP133}$/P33		
K3	$\overline{DMAAK1}$/PBD1	N8	SO2/$\overline{INTP130}$/P30		
K4	$\overline{DMAAK3}$/PBD3	N9	X2		
K11	ANI1/P71	N10	CV_{SS}		
K12	ANI0/P70	N11	$\overline{SCK0}$/P42		
K13	V_{SS}	N12	AV_{DD}/AV_{REF}		
K14	V_{DD}	N13	AV_{SS}		
L1	−	N14	−		
L2	$\overline{DMAAK2}$/PBD2	P1	V_{DD}		
L3	TI010/INTP010/P11	P2	V_{SS}		
L4	$\overline{DMAAK0}$/PBD0	P3	$\overline{TC1}$/$\overline{INTP111}$/P25		
L5	TO02/P23	P4	INTP021/P22		

備考1. A1, A5, A10, B1, B14, C1, C14, D14, E5, L1, M1, M14, N1, N14, P5, P11, P14**の端子は，オープンにしてください。**

2. **（　）は**μPD70F3107A**での端子名です。**

★ 1. 6　内部ブロック図



注1.　μPD703103A　　：ROM**レス**

μPD703105A, 703106A　：128 K**バイト（マスク**ROM）

μPD703107A　　：256 K**バイト（マスク**ROM）

μPD70F3107A　　：256 K**バイト（フラッシュ・メモリ）**

2.　μPD703103A, 703105A　　：4 K**バイト**

μPD703106A, 703107A, 70F3107A　：10 K**バイト**

3.　μPD70F3107A**のみ有効**

第2章　バス・インタフェース接続回路例（1）

V850E/MA1に直結できるメモリについて説明します。回路例と関連レジスタの設定は次に示す考え方で構成しています。

(1) V850E/MA1の内部システム・クロックおよびバス・サイクルは50 MHz

備考　表2－2　SRAM接続時のウエイト／アイドル設定とバス・クロック一覧で50 MHz以外のウエイト設定，アイドル設定を示します。

(2) ウエイト制御はプログラマブル・ウエイトを使用（$\overline{\text{WAIT}}$端子：未使用（プルアップ））

(3) 外部バス・マスタは接続しない（$\overline{\text{HLDRQ}}$端子：未使用（プルアップ））

(4) BCPレジスタは8xHを設定，BECレジスタは任意

- バス・サイクル周期：通常
- $\overline{\text{IORD}}$, $\overline{\text{IOWR}}$の動作：必要に応じて決定
- エンディアンの設定：システム（使用するソフトウエア）によって決定

(5) 接続するメモリは2. 1　SRAMとの接続を除いて3 Vデバイスを例にあげていますが，5 Vデバイスの場合もV850E/MA1と接続デバイスのDC特性（TTLレベル）を満たせば直結可能

次に5V SRAM（μPD434008A：V850E/MA1に直結可能）とV850E/MA1のDC特性を比較します。

表2－1　5 V SRAMとV850E/MA1のDC特性

V850E/MA1の動作	V850E/MA1 バス関連端子の入出力特性	SRAM（μPD434008A） 入出力特性
リード時	V_{IH}　：2.0 V（MIN.） 5.5 V（MAX.） V_{IL}　：0.2 V_{DD}（MAX.）	V_{OH}　：2.4 V（MIN.） V_{OL}　：0.4 V（MAX.）
ライト時	V_{OH}　：0.8 V_{DD}（MIN.） V_{OL}　：0.45 V（MAX.）	V_{IH}　：2.2 V（MIN.） V_{IL}　：0.8 V（MAX.）

また，プリント基板配線上の遅延時間は含まれていないので，実際の回路では必要に応じて数nsのマージンを考慮してください。

2. 1　SRAMとの接続

2. 1. 1　μPD434008Aとの接続（8ビット・バス幅例）

SRAM（μPD434008ALE-12：512 K×8ビット，5 V）を1つ（8ビット・バス幅，512 Kバイトの外部メモリ空間）接続する例を示します。

【回路構成】

- 内部システム・クロック：50 MHz
- 接続デバイス：μPD434008ALE-12×1つ
- 使用$\overline{\text{CS}}$信号：$\overline{\text{CS0}}$

  外部メモリ空間の0100000H-017FFFFH（ブロック0）に配列することとします。

  0180000H-01FFFFFHはイメージとなります。

【接続の考え方と注意点】

8ビット・バス幅で構成するためSRAMのデータ・バスはV850E/MA1のD0-D7に，アドレス・バスはV850E/MA1のA0-A18に接続します。また$\overline{\text{CS}}$端子，$\overline{\text{OE}}$端子，$\overline{\text{WE}}$端子はV850E/MA1の$\overline{\text{CS0}}$端子，$\overline{\text{RD}}$端子，$\overline{\text{LWR}}$端子にそれぞれ接続します。

【レジスタの設定 】

- $\overline{\text{CS0}}$の使用ブロックとデバイス・タイプ：ブロック0，SRAM，外部I/O
- ウエイト設定：1ウエイト
- アドレス・セットアップ・ウエイト：0ウエイト
- アイドル・ステート：1ステート

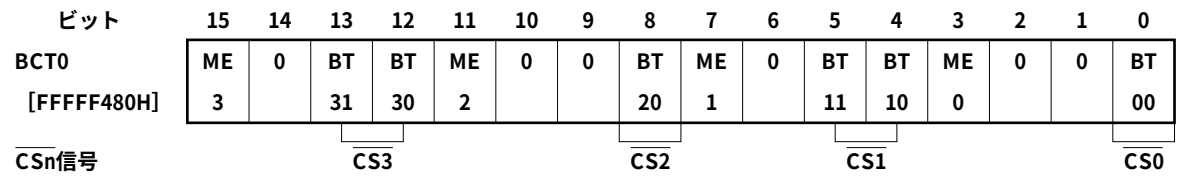
図2－1　チップ・エリア選択コントロール・レジスタ0（CSC0）の設定

ビット	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSC0	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS
[FFFFF060H]	33	32	31	30	23	22	21	20	13	12	11	10	03	02	01	00

ブロック0アクセス時$\overline{\text{CS0}}$出力

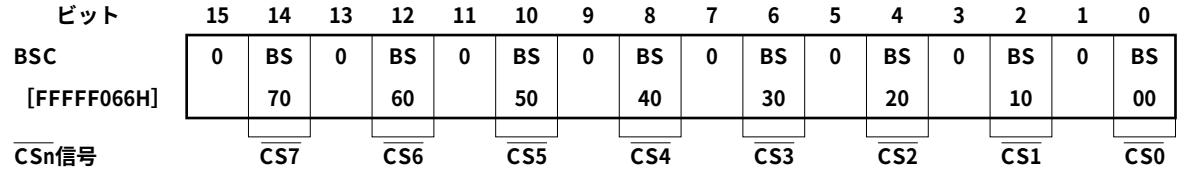
CSC0の設定値：xxxxxxxxxxxx0001B

図2－2　バス・サイクル・タイプ・コンフィギュレーション・レジスタ0（BCT0）の設定



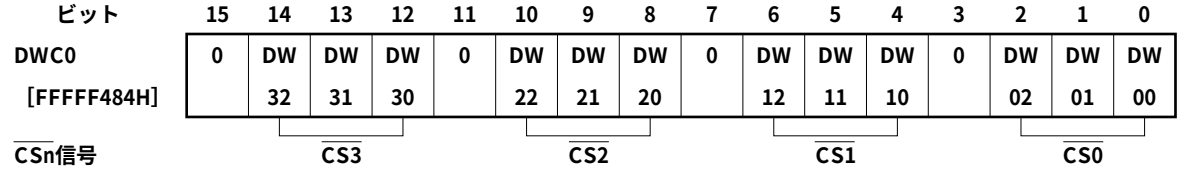
$\overline{CS0}$空間をメモリ・コントローラ動作許可，SRAM，外部I/Oに指定
BCT0の設定値：x0xxx00xx0xx1000B

図2－3　バス・サイズ・コンフィギュレーション・レジスタ（BSC）の設定



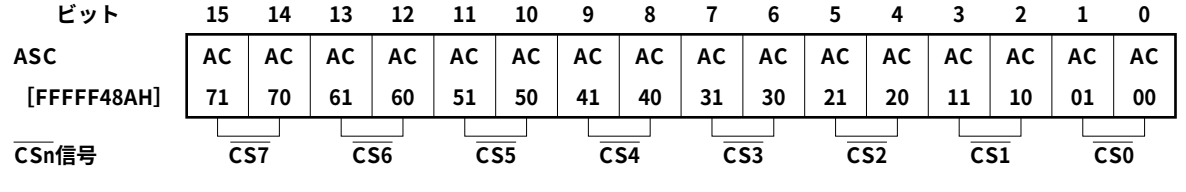
$\overline{CS0}$空間をデータ・バス幅8ビットに指定
BSCの設定値：0x0x0x0x0x0x0x00B

図2－4　データ・ウエイト・コントロール・レジスタ0（DWC0）の設定



$\overline{CS0}$空間のウエイト：1
DWC0の設定値：0xxx0xxx0xxx0001B

図2－5　アドレス・セットアップ・ウエイト・コントロール・レジスタ（ASC）の設定



$\overline{CS0}$空間のアドレス・セットアップ・ウエイト：0
ASCの設定値：xxxxxxxxxxxxxx00B

図2－6　バス・サイクル・コントロール・レジスタ（BCC）の設定

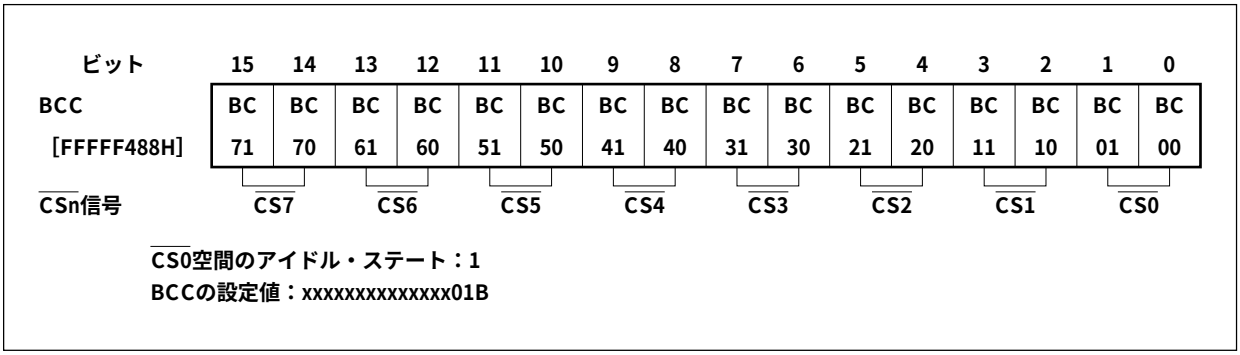


図2－7　μPD434008ALE-12接続回路例

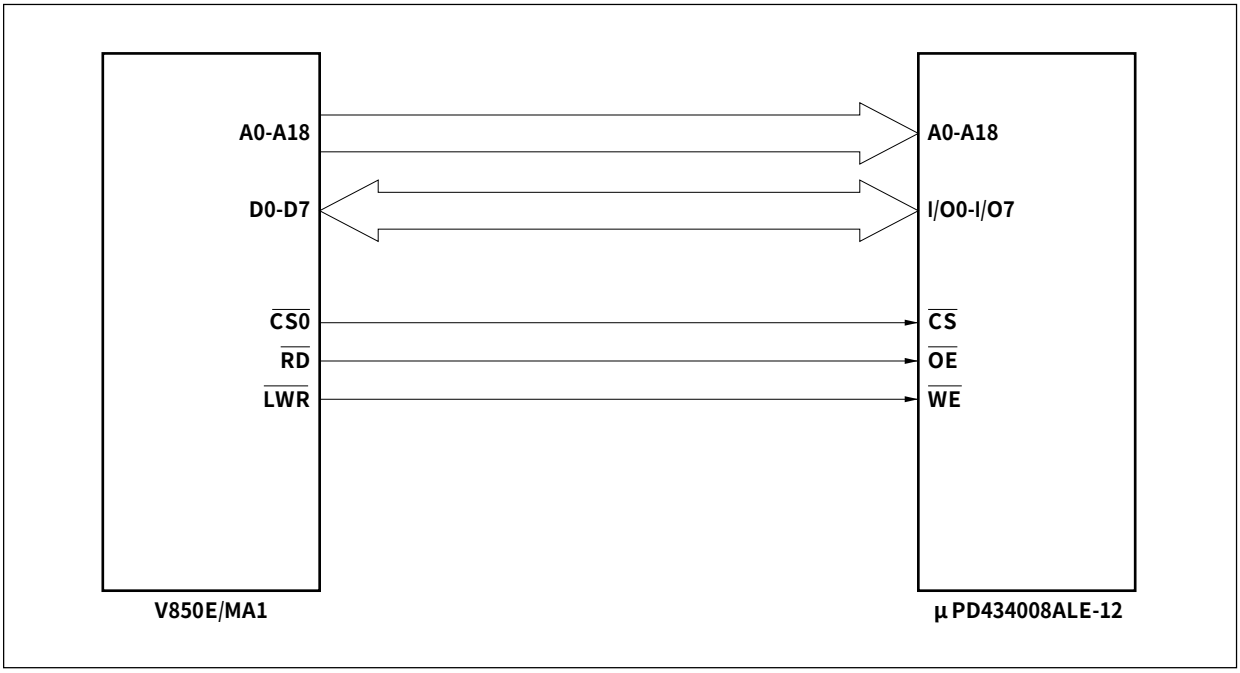
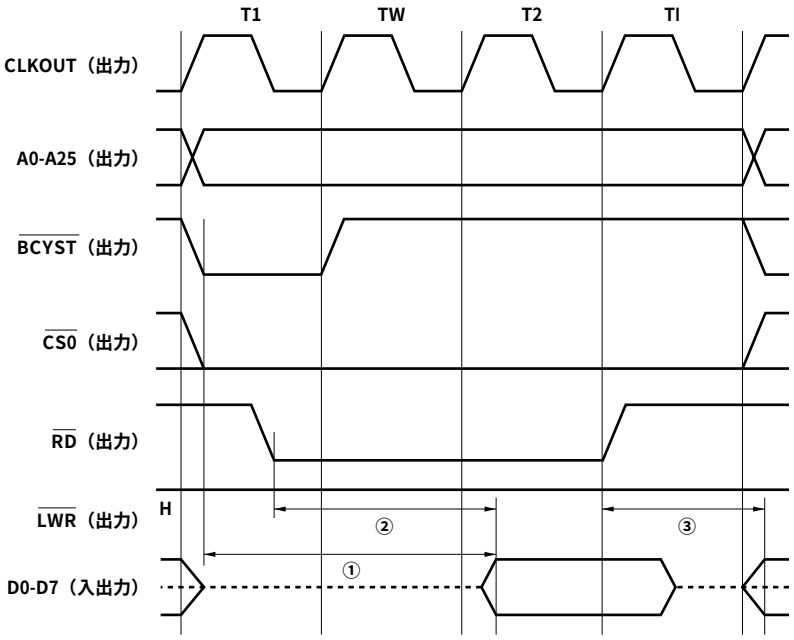


図2－8　μPD434008ALE-12のリード動作



① μPD434008ALE-12のアドレス，CSアクティブからの出力遅延時間：12 ns（MAX.）
V850E/MA1の電気的特性より，データ入力設定時間（対アドレス）の最大値

t_{SAID}（ns）　=（2＋w＋w_D＋w_{AS}）T－21
= 3×20－21　；w = 0, w_D = 1, w_{AS} = 0, T = 20 ns
= 39 ns（＞12 ns）

② μPD434008ALE-12のOEアクティブからの出力遅延時間：6 ns（MAX.）
V850E/MA1の電気的特性より，データ入力設定時間（対RD）の最大値

t_{SRDID}（ns）　=（1.5＋w＋w_D）T－21
= 2.5×20－21；w = 0, w_D = 1, T = 20 ns
= 29 ns（＞6 ns）

③ μPD434008ALE-12のOEインアクティブからの出力フローティング遅延時間：6 ns（MAX.）
V850E/MA1の電気的特性より，データ出力遅延時間（対RD↑）の最小値

t_{DRDOD}（ns）=（0.5＋i）T－10
= 1.5×20－10；i = 1, T = 20 ns
= 20 ns（＞6 ns）

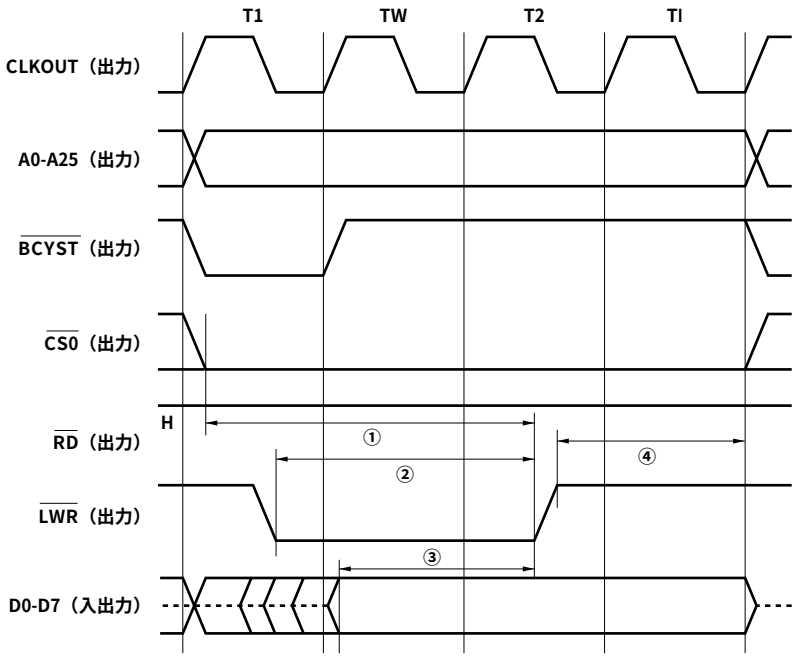
またRDハイ・レベル幅の最小値

t_{WRDH}（ns）　=（0.5＋w_{AS}＋i）T－10
= 1.5×20－10；w_{AS} = 0, i = 1, T = 20 ns
= 20 ns（＞6 ns）

備考1.　破線はハイ・インピーダンスを示しています。

2.　T：t_{CYK}（CLKOUT出力周期）
w：WAITによるウエイト数
w_D：DWC1, DWC2によるウエイト数
w_{AS}：ASCによるアドレス・セットアップ・ウエイト数
i：アイドル・ステート数

図2－9　μPD434008ALE-12のライト動作



① μPD434008ALE-12のアドレス，CSアクティブからライト終了までの時間：8 ns（MIN.）
V850E/MA1の電気的特性より，アドレス設定時間（対LWR↑）の最小値
t_{SAWR}（ns）＝（1.5＋w＋w_D＋w_{AS}）T－10
＝2.5×20－10　；w = 0, w_D = 1, w_{AS} = 0, T = 20 ns
＝40 ns（＞8 ns）

② μPD434008ALE-12のWEアクティブ・パルス幅：8 ns（MIN.）
V850E/MA1の電気的特性より，LWRロウ・レベル幅の最小値
t_{WWRL}（ns）＝（1＋w＋w_D）T－10
＝2×20－10；w = 0, w_D = 1, T = 20 ns
＝30 ns（＞8 ns）

③ μPD434008ALE-12のデータ・バリッドからライト終了までの時間：6 ns（MIN.）
V850E/MA1の電気的特性より，データ出力設定時間（対LWR↑）の最小値
t_{SODWR}（ns）＝（0.5＋w_{AS}＋w＋w_D）T－10
＝1.5×20－10；w_{AS} = 0, w = 0, w_D = 1, T = 20 ns
＝20 ns（＞6 ns）

④ μPD431008ALE-12のライト終了からのデータ保持時間：0 ns（MIN.）
V850E/MA1の電気的特性より，データ出力保持時間（対LWR↑）の最小値
t_{HWROD}（ns）＝（0.5＋i）T－10
＝1.5×20－10；i = 1, T = 20 ns
＝20 ns（＞0 ns）

備考1.　破線はハイ・インピーダンスを示しています。
2.　T：t_{CYK}（CLKOUT出力周期）
w：WAITによるウエイト数
w_D：DWC1, DWC2によるウエイト数
w_{AS}：ASCによるアドレス・セットアップ・ウエイト数
i：アイドル・ステート数

2. 1. 2　μPD434008Aとの接続（16ビット・バス幅例）

SRAM（μPD434008ALE-12：512 K×8ビット，5 V）を2つ（16ビット・バス幅，1 Mバイトの外部メモリ空間）接続する例を示します。

【回路構成】

- 内部システム・クロック：50 MHz
- 接続デバイス：μPD434008ALE-12×2つ
- 使用$\overline{CS}$信号：$\overline{CS0}$

  外部メモリ空間の0100000H-01FFFFFH（ブロック0）に配列することとします。

【接続の考え方と注意点】

16ビット・バス幅で構成するため，V850E/MA1のアドレス・バス（A1-A19）をSRAMのA0-A18に接続します。SRAMの$\overline{WE}$端子は，V850E/MA1のD0-D7に接続したSRAMをV850E/MA1の$\overline{LWR}$端子に，V850E/MA1のD8-D15に接続したSRAMを$\overline{UWR}$端子にそれぞれ接続します。SRAMの$\overline{CS}$端子，$\overline{OE}$端子はV850E/MA1の$\overline{CS0}$端子，$\overline{RD}$端子にそれぞれ接続します。

【レジスタの設定】

BSC（$\overline{CSn}$空間のバス幅指定）の設定を除いて2. 1. 1　μPD434008Aとの接続（8ビット・バス幅例）と同等です。

図2－10　バス・サイズ・コンフィギュレーション・レジスタ（BSC）の設定

ビット	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BSC	0	BS	0	BS	0	BS	0	BS	0	BS	0	BS	0	BS	0	BS
[FFFFF066H]		70		60		50		40		30		20		10		00
$\overline{CSn}$信号		$\overline{CS7}$		$\overline{CS6}$		$\overline{CS5}$		$\overline{CS4}$		$\overline{CS3}$		$\overline{CS2}$		$\overline{CS1}$		$\overline{CS0}$

$\overline{CS0}$空間をデータ・バス幅16ビットに指定
BSCの設定値：0x0x0x0x0x0x0x01B

図2－11　μPD434008ALE-12接続回路例

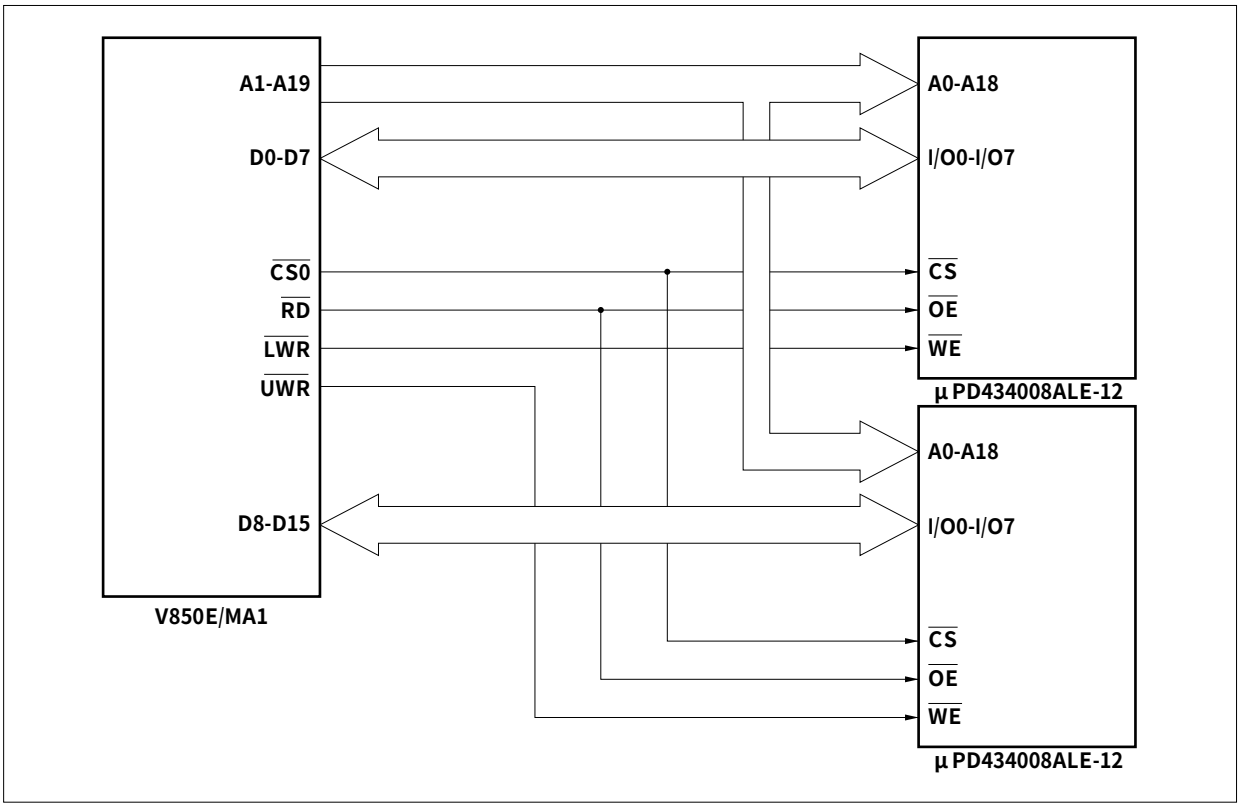
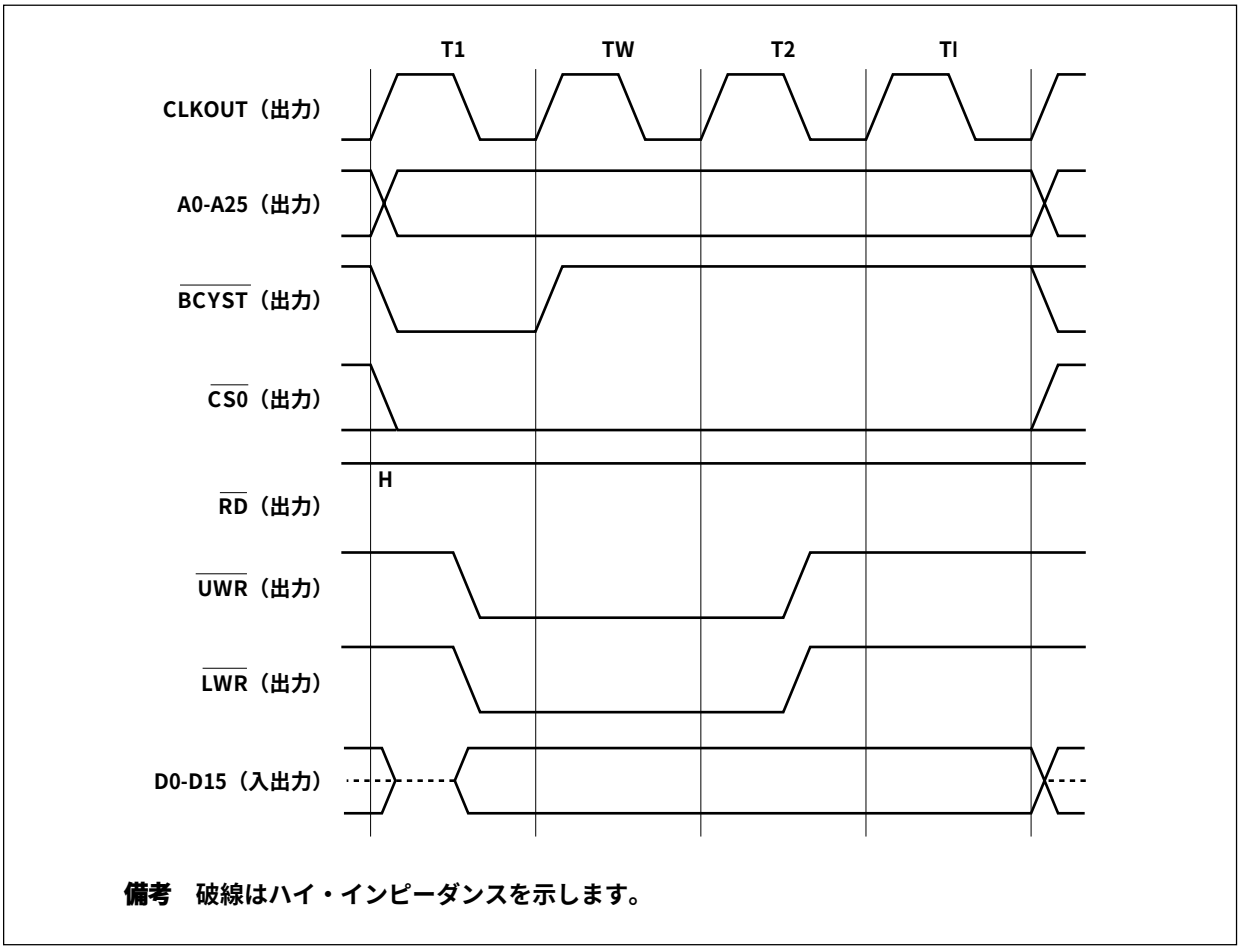
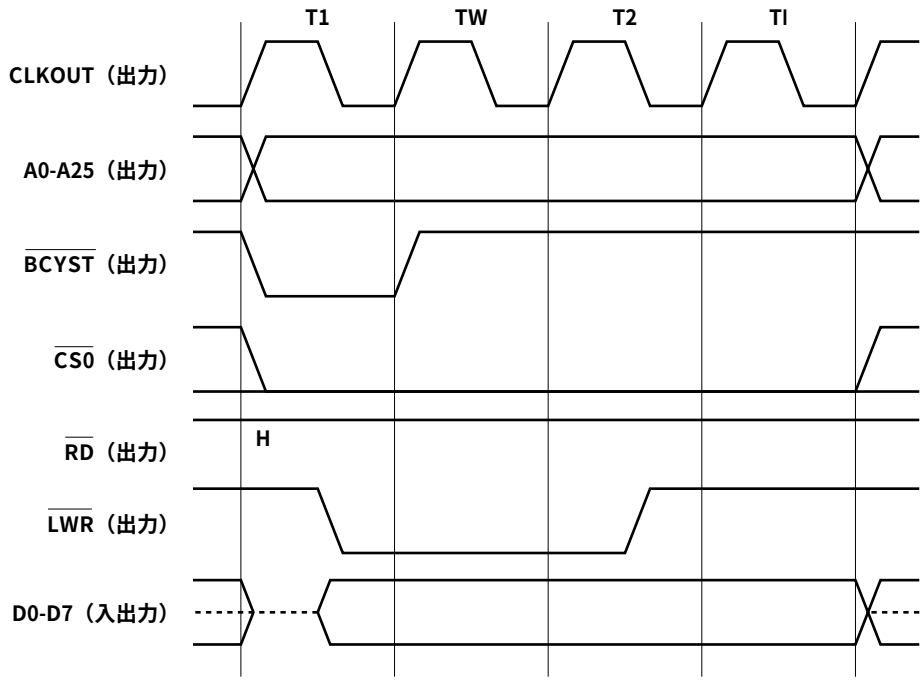


図2－12　μPD434008ALE-12のライト動作（16ビット・アクセス）



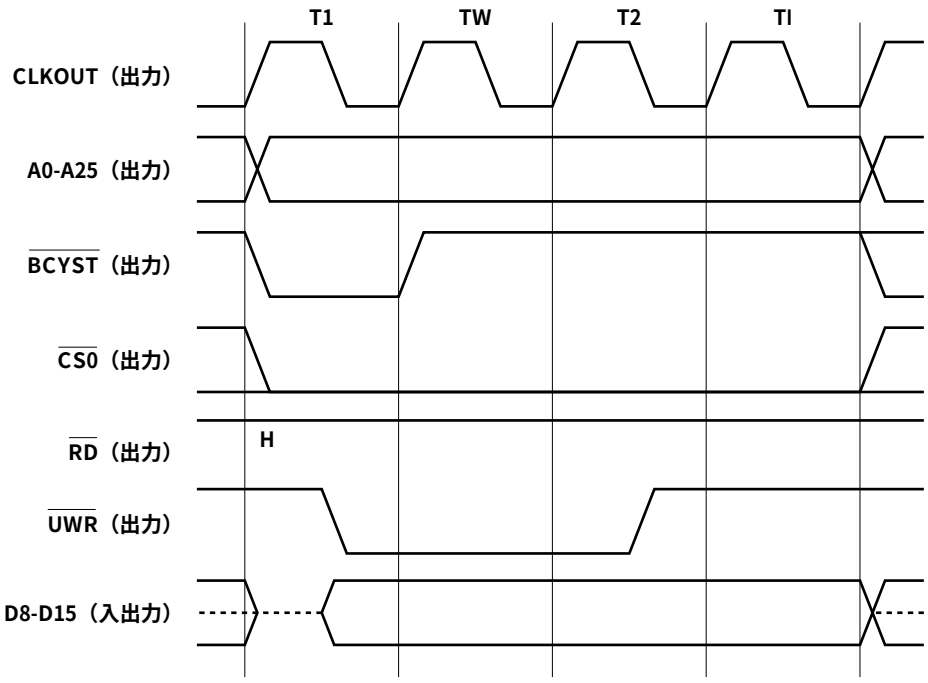
備考　破線はハイ・インピーダンスを示します。

図2－13　μPD434008ALE-20のライト動作（D0-D7に対する8ビット・アクセス）



備考　破線はハイ・インピーダンスを示します。

図2－14　μPD434008ALE-20のライト動作（D8-D15に対する8ビット・アクセス）



備考　破線は，ハイ・インピーダンスを示します。

表2－2　SRAM接続時のウエイト／アイドル設定とバス・クロック一覧

（a）μPD434008ALE-12との接続

fxx（CLKOUT）	データ・ウエイト	アドレス・セットアップ・ウエイト	アイドル・ステート	制限されるタイミング
fxx≦31 MHz	0	0	0	－
31 MHz＜fxx（≦50 MHz）	1	0	1	t_{DRDOD}, t_{SODWR}

（b）μPD434008ALE-15/μPD434008ALLE-15との接続

fxx（CLKOUT）	データ・ウエイト	アドレス・セットアップ・ウエイト	アイドル・ステート	制限されるタイミング
fxx≦29 MHz	0	0	0	－
29 MHz＜fxx（≦50 MHz）	1	0	1	t_{DRDOD}, t_{SODWR}

（c）μPD434008ALE-20/μPD434008ALLE-20との接続

fxx（CLKOUT）	データ・ウエイト	アドレス・セットアップ・ウエイト	アイドル・ステート	制限されるタイミング
fxx≦26 MHz	0	0	0	－
26 MHz＜fxx（≦50 MHz）	1	0	1	t_{DRDOD}, t_{SODWR}

（d）μPD431000A-A10との接続（V_{CC}≧3.0 V, $\overline{CE1}$を$\overline{CSn}$に接続，CE2をハイ・レベル固定で使用した場合）

fxx（CLKOUT）	データ・ウエイト	アドレス・セットアップ・ウエイト	アイドル・ステート	制限されるタイミング
fxx≦7 MHz	0	0	0	－
7 MHz＜fxx≦11 MHz	1	0	0	t_{SODWR}
11 MHz＜fxx≦21 MHz	1	0	1	t_{DRDOD}
21 MHz＜fxx≦33 MHz	2	0	1	t_{SODWR}
33 MHz＜fxx≦41 MHz	3	0	2	t_{DRDOD}, t_{SAID}
41 MHz＜fxx≦49 MHz	3	1	2	t_{SAID}
49 MHz＜fxx（≦50 MHz）	3	2	2	

2. 2　PROMとの接続

PROM（HN27C4096H：256 K×16ビット）を1つ（512 Kバイトの外部ROM空間）接続する例を示します。

【回路構成】

- 内部システム・クロック：50 MHz
- 接続デバイス：HN27C4096H-85×1つ
- 使用$\overline{\text{CS}}$信号：$\overline{\text{CS0}}$

  外部メモリ空間の0100000H-017FFFFH（ブロック0）に配列することとします。

  0180000H-01FFFFFHはイメージとなります。

【接続の考え方と注意点】

- V850E/MA1のアドレス信号（A1-A18）をHN27C4096H-85のアドレス信号（A0-A17）に接続します。
- PROMへのアクセスはSRAMのリード動作と同等です。したがって関連レジスタの設定値は2. 1. 2 μPD434008Aとの接続（16ビット・バス幅例）と同等になります（各種ウエイト設定のみ変更）。

【レジスタの設定】

- $\overline{\text{CS0}}$の使用ブロックとデバイス・タイプ：ブロック0，SRAM，外部I/O
- ウエイト設定：4ウエイト（2ウエイト[注]）
- アドレス・セットアップ・ウエイト：0ウエイト（2ウエイト[注]）
- アイドル・ステート：2ステート

注　ウエイト設定 = 2，アドレス・セットアップ・ウエイト = 2としても同等です。

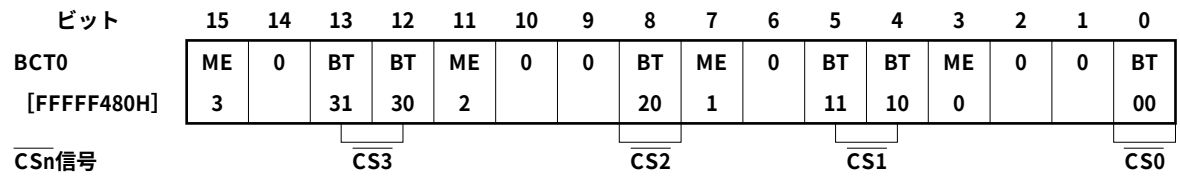
図2－15　チップ・エリア選択コントロール・レジスタ0（CSC0）の設定

ビット	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSC0	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS
[FFFFF060H]	33	32	31	30	23	22	21	20	13	12	11	10	03	02	01	00

ブロック0アクセス時$\overline{\text{CS0}}$出力

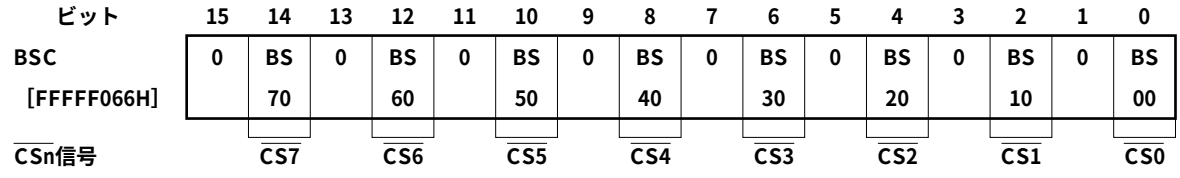
CSC0の設定値：xxxxxxxxxxxx0001B

図2－16　バス・サイクル・タイプ・コンフィギュレーション・レジスタ0（BCT0）の設定



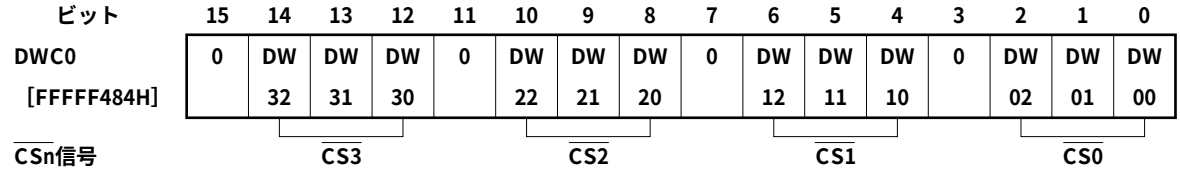
$\overline{CS0}$空間をメモリ・コントローラ動作許可，SRAM，外部I/Oに指定
BCT0の設定値：x0xxx00xx0xx1000B

図2－17　バス・サイズ・コンフィギュレーション・レジスタ（BSC）の設定



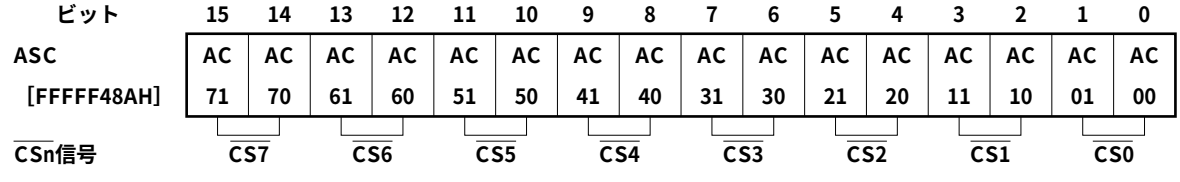
$\overline{CS0}$空間をデータ・バス幅16ビットに指定
BSCの設定値：0x0x0x0x0x0x0x01B

図2－18　データ・ウエイト・コントロール・レジスタ0（DWC0）の設定



$\overline{CS0}$空間のウエイト：4
DWC0の設定値：0xxx0xxx0xxx0100B

図2－19　アドレス・セットアップ・ウエイト・コントロール・レジスタ（ASC）の設定



$\overline{CS0}$空間のアドレス・セットアップ・ウエイト：0
ASCの設定値：xxxxxxxxxxxxxx00B

図2－20　バス・サイクル・コントロール・レジスタ（BCC）の設定

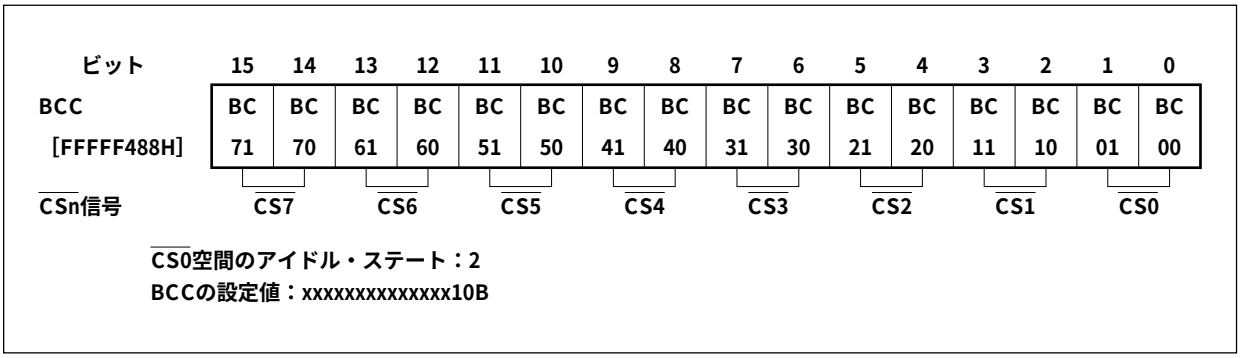


図2－21　HN27C4096H-85接続回路例

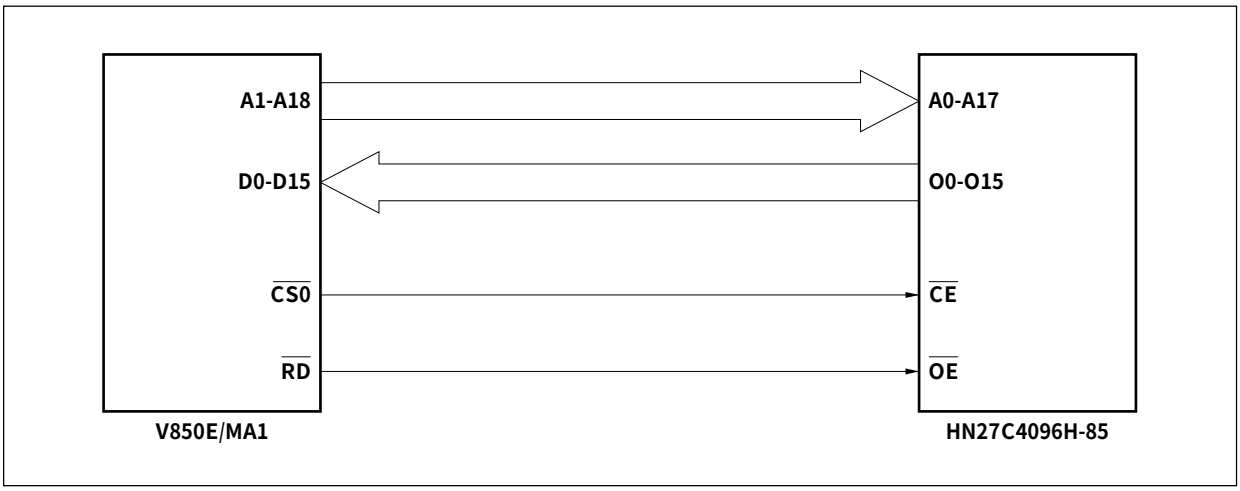
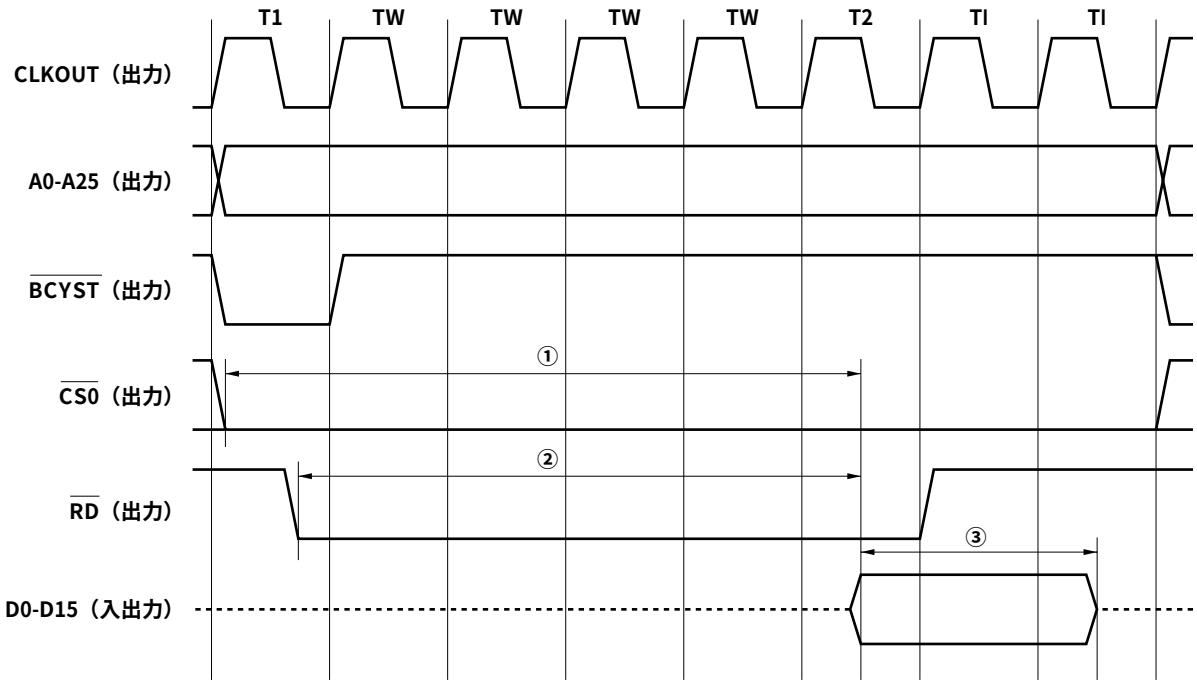


図2－22　HN27C4096H-85のリード動作



① HN27C4096H-85の$\overline{CE}$信号アクティブからのデータ出力遅延時間：85 ns（MAX.）

② HN27C4096H-85の$\overline{OE}$信号アクティブからのデータ出力遅延時間：45 ns（MAX.）

③ HN27C4096H-85の$\overline{OE}$信号インアクティブからのデータ出力フローティング遅延時間：30 ns（MAX.）

備考　破線はハイ・インピーダンスを示します。

2. 3　ページROMとの接続

ページROM（HN27C4000G：256 K×16ビット/512 K×8ビット）を1つ（16ビット・バス幅，512 Kバイトの外部ROM空間）接続する例を示します。

【回路構成】

- 内部システム・クロック：50 MHz
- 接続デバイス：HN27C4000G-10×1つ
- 使用$\overline{CS}$信号：$\overline{CS1}$

  外部メモリ空間の0200000H-027FFFFH（ブロック1）に配列することとします。

注意　この例ではブロック1を$\overline{CS1}$で使用する例を示しています。この例の場合，ブロック1と0800000H-3FFFFFFHのアクセスで$\overline{CS1}$が出力されます（0280000H-03FFFFFH，0800000H-3FFFFFFHがイメージ）。

【接続の考え方と注意点】

- HN27C4000G-10の$\overline{BYTE}$/V_{PP}端子は16ビットで使用するためプルアップする。
- CSC0レジスタの設定はブロック１を$\overline{CS1}$で使用するためブロック１アクセス時$\overline{CS0}$，$\overline{CS2}$とも「出力しない」に設定する。
- HN27C4000G-10のページ・サイズは8バイトのためPRCレジスタの設定は4ワード×16ビットとする。

【レジスタの設定】

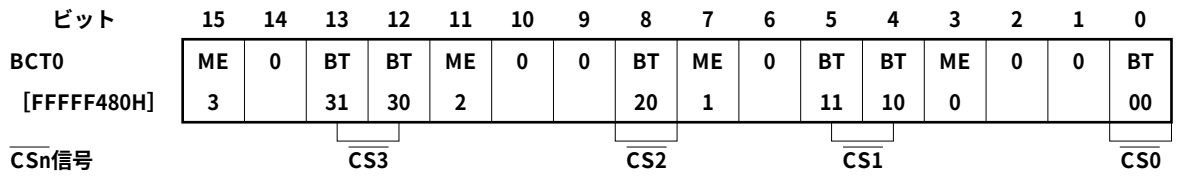
- $\overline{CS1}$の使用ブロックとデバイス・タイプ：ブロック1，ページROM
- オフページ時のウエイト設定：5ウエイト
- オンページ時のウエイト設定：2ウエイト
- アドレス・セットアップ・ウエイト：0ウエイト
- アイドル・ステート：2ステート

図2－23　チップ・エリア選択コントロール・レジスタ0（CSC0）の設定

ビット	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSC0	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS
[FFFFF060H]	33	32	31	30	23	22	21	20	13	12	11	10	03	02	01	00

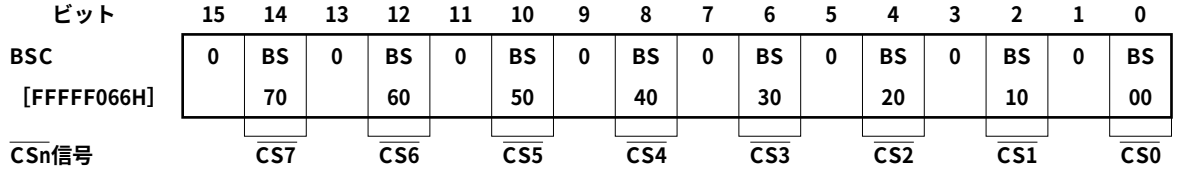
ブロック1アクセス時$\overline{CS1}$出力
CSC0の設定値：xxxxxx0xxxxxxx0xB

図2－24　バス・サイクル・タイプ・コンフィギュレーション・レジスタ0（BCT0）の設定



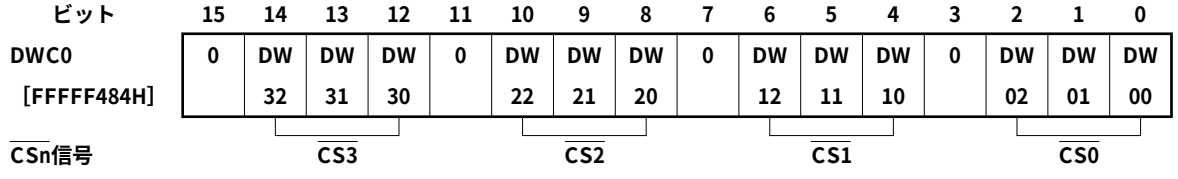
$\overline{CS1}$空間をメモリ・コントローラ動作許可，ページROMに指定
BCT0の設定値：x0xxx00x1001x00xB

図2－25　バス・サイズ・コンフィギュレーション・レジスタ（BSC）の設定



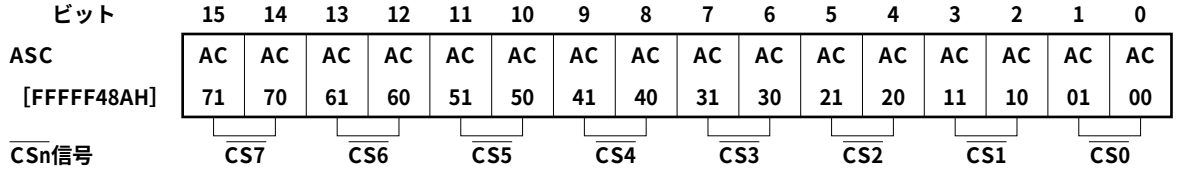
$\overline{CS1}$空間をデータ・バス幅16ビットに指定
BSCの設定値：0x0x0x0x0x0x010xB

図2－26　データ・ウエイト・コントロール・レジスタ0（DWC0）の設定



$\overline{CS1}$空間のウエイト：5（オフ・ページ時）
DWC0の設定値：0xxx0xxx01010xxxB

図2－27　アドレス・セットアップ・ウエイト・コントロール・レジスタ（ASC）の設定



$\overline{CS1}$空間のアドレス・セットアップ・ウエイト：0
ASCの設定値：xxxxxxxxxxxx00xxB

図2－28　バス・サイクル・コントロール・レジスタ（BCC）の設定

ビット	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BCC	BC	BC	BC	BC	BC	BC	BC	BC	BC	BC	BC	BC	BC	BC	BC	BC
[FFFFF488H]	71	70	61	60	51	50	41	40	31	30	21	20	11	10	01	00
$\overline{CS}$n信号		$\overline{CS7}$		$\overline{CS6}$		$\overline{CS5}$		$\overline{CS4}$		$\overline{CS3}$		$\overline{CS2}$		$\overline{CS1}$		$\overline{CS0}$

$\overline{CS1}$空間のアイドル・ステート：2
BCCの設定値：xxxxxxxxxxxx10xxB

図2－29　ページROMコンフィギュレーション・レジスタ（PRC）の設定

ビット	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRC	0	PRW	PRW	PRW	0	0	0	0	0	0	0	0	MA6	MA5	MA4	MA3
[FFFFF49AH]		2	1	0												

オンページ時のウエイト数：2
ページ・サイズ：4ワード×16ビット
PRCの設定値：0010000000000000B

図2－30　HN27C4000G-10接続回路例

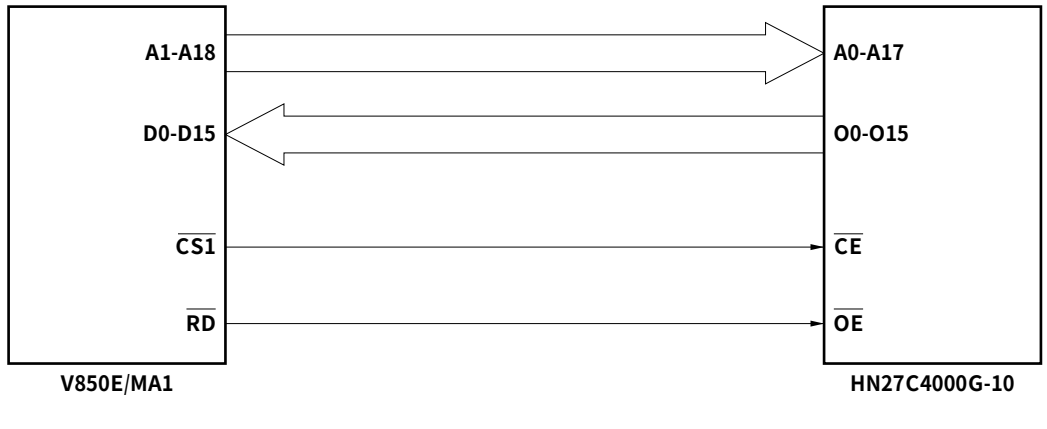
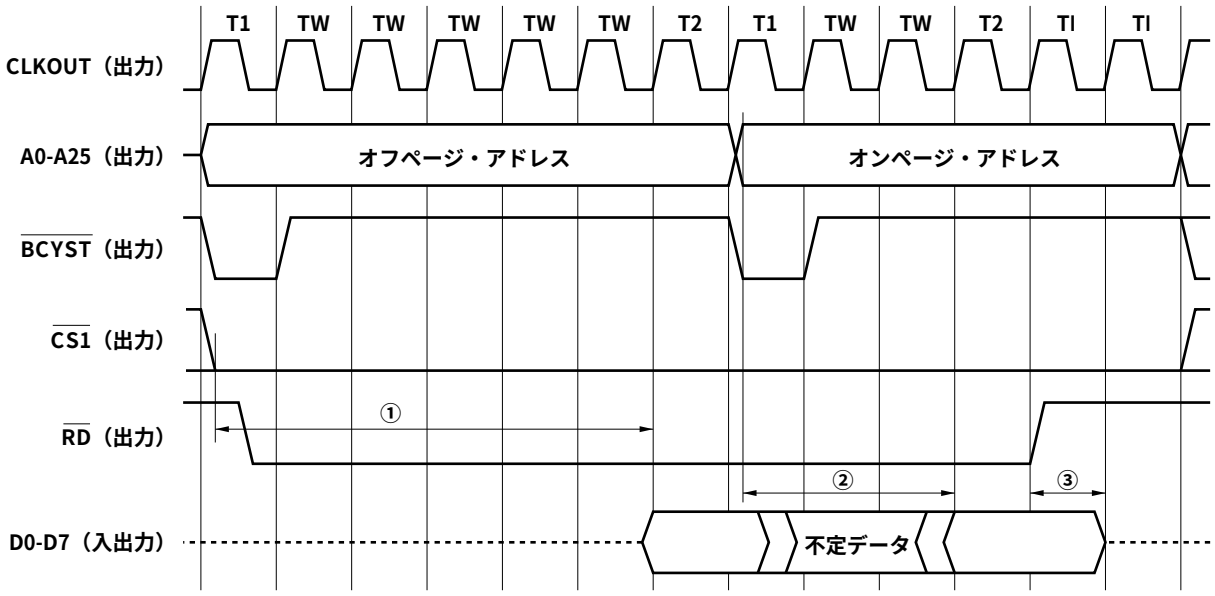


図2－31　HN27C4000G-10のリード動作



① HN27C4000G-10の$\overline{CE}$信号アクティブからのデータ出力遅延時間：100 ns（MAX.）

② オンページ時アドレス変化からのデータ出力遅延時間：50 ns（MAX.）

③ データ出力フローティング遅延時間：35 ns（MAX.）

注意　8ビット・バス幅では，ページ・ヒット時最大4バス・サイクル分（16ビット・バス幅では最大2サイクル）$\overline{RD}$信号をロウ・レベルに保って連続アクセスします。その間のアイドルは挿入されません。

備考　破線はハイ・インピーダンスを示します。

2. 4　フラッシュ・メモリ（SST29LE010）との接続

フラッシュ・メモリ（SST29LE010：128 K×8ビット）を1つ使用し，8ビット・バス幅，128 Kバイトの外部メモリ空間を接続する例を示します。

【回路構成】

- 内部システム・クロック：50 MHz
- 接続デバイス：SST29LE010-150×1つ
- 使用$\overline{CS}$信号：$\overline{CS2}$

  外部メモリ空間の0400000H-041FFFFH（ブロック2）に配列することとします。

  0420000H-05FFFFFHはイメージになります。

【接続の考え方と注意点】

- 8ビット・バス幅で構成するためSST29LE010のデータ・バスはV850E/MA1のD0-D7に，アドレス・バスはV850E/MA1のA0-A16に接続します。また$\overline{CE}$端子，$\overline{OE}$端子，$\overline{WE}$端子はV850E/MA1の$\overline{CS2}$端子，$\overline{RD}$端子，$\overline{LWR}$端子にそれぞれ接続します。

  備考1. SST29LE010は3V単一でリード／ライト可能なフラッシュ・メモリです。RY/$\overline{BY}$端子も配列されていませんのでSRAMとの接続と同様になります。

  2. RY/$\overline{BY}$端子を配列しているフラッシュ・メモリを使用する場合はRY/$\overline{BY}$端子をV850E/MA1のポート入力などに接続することによりライト時／イレース時の完了状態を監視することができます。

- $\overline{CS2}$空間のウエイトはアドレス・セットアップ・ウエイト = 1，データ・ウエイト = 6，アイドル・ステート = 2を設定します。

  備考　SST29LE010の$\overline{WE}$信号ハイ・レベル・パルス幅の最小値（50 ns）を満足するためにi = 2, w_{AS} = 1を設定します。$\overline{WE}$信号ハイ・レベル・パルス幅の最小値はi = 3, w_{AS} = 0でも満足しますが，その場合リード時の$\overline{CE}$信号アクティブからのデータ設定時間の最小値（150 ns）を満足するためにw_D = 7が必要となって1クロック遅くなります。

- SST29LE010のデータ・ライト動作とイレース動作の一例を示します。

  チップ・イレース：

  ① SST29LE010のアドレス5555H[注1]にデータAAHをライト
  ② SST29LE010のアドレス2AAAH[注2]にデータ55Hをライト
  ③ SST29LE010のアドレス5555H[注1]にデータ80Hをライト
  ④ SST29LE010のアドレス5555H[注1]にデータAAHをライト
  ⑤ SST29LE010のアドレス2AAAH[注2]にデータ55Hをライト
  ⑥ SST29LE010のアドレス5555H[注1]にデータ10Hをライト
  ⑦ 所定の時間（20.2 ms）経過後イレース完了[注3]

データ・ライト：

① SST29LE010のアドレス5555H[注1]にデータAAHをライト

② SST29LE010のアドレス2AAAH[注2]にデータ55Hをライト

③ SST29LE010のアドレス5555H[注1]にデータA0Hをライト

④ SST29LE010の該当アドレスに該当データをライト[注4]

⑤ 所定の時間（10.2 ms）経過後ライト完了[注3]

注1. SST29LE010のA15, A16は任意です。この例ではV850E/MA1の外部メモリ空間のアドレス0405555Hに対応します。

2. SST29LE010のA15, A16は任意です。この例ではV850E/MA1の外部メモリ空間の0402AAAHに対応します。

3. SST29LE010のD7ビット（ポーリング・ビット）またはD6ビット（トグル・ビット）を監視する方法もあります。

・データ・ポーリング・ビット（D7）：

該当アドレスをリードすることにより動作完了を確認できます。

ビジィ中は反転データ（イレースの場合0，ライトの場合ライト・データの反転）が出力されます。

ライト時は最後にライトしたアドレス，オール・イレース時は最終アドレス（この例ではV850E/MA1の041FFFFH）をリードして確認します。

・トグル・ビット（D6）：

該当アドレスを2回連続してリードすることにより動作完了を確認できます。

ビジィ中は2回の連続したリード動作に対してトグルします。連続して同一データがリードできたら完了です。ライト時は最後にライトしたアドレス，オール・イレース時は最終アドレスをリードして確認します。

4. SST29LE010では最大1ページ（128バイト）の連続ライトが可能です。

【レジスタの設定】

- $\overline{CS2}$の使用ブロックとデバイス・タイプ：ブロック2，SRAM，外部I/O
- ウエイト設定：6ウエイト
- アドレス・セットアップ・ウエイト：1ウエイト
- アイドル・ステート：2ステート

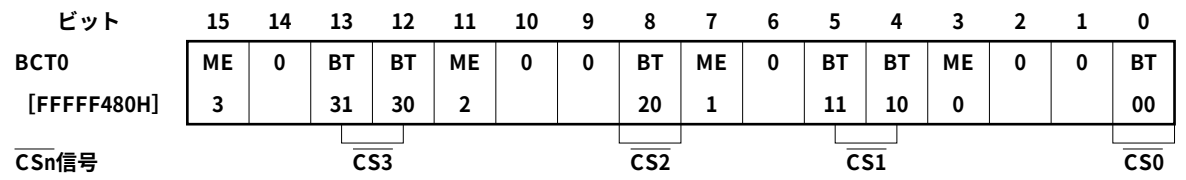
図2－32　チップ・エリア選択コントロール・レジスタ0（CSC0）の設定

ビット	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSC0	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS
[FFFFF060H]	33	32	31	30	23	22	21	20	13	12	11	10	03	02	01	00

ブロック2アクセス時$\overline{CS2}$出力

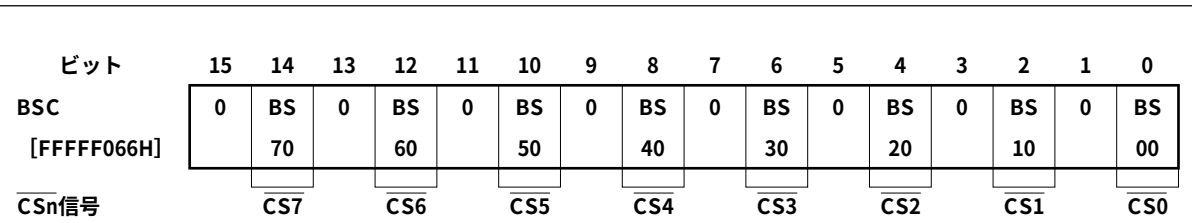
CSC0の設定値：xxxx0100xxxxx0xxB

図2－33　バス・サイクル・タイプ・コンフィギュレーション・レジスタ0（BCT0）の設定



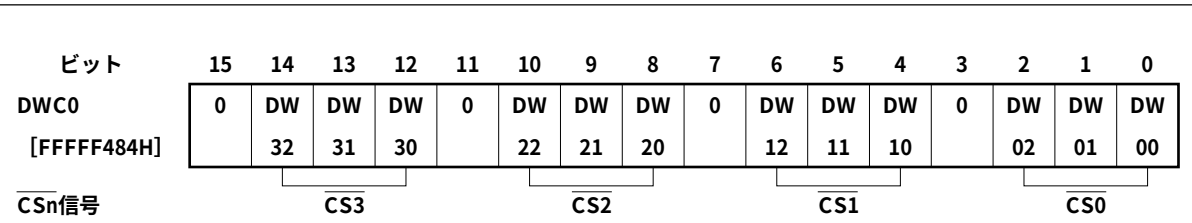
$\overline{CS2}$空間をメモリ・コントローラ動作許可，SRAM，外部I/Oに指定
BCT0の設定値：x0xx1000x0xxx00xB

図2－34　バス・サイズ・コンフィギュレーション・レジスタ（BSC）の設定



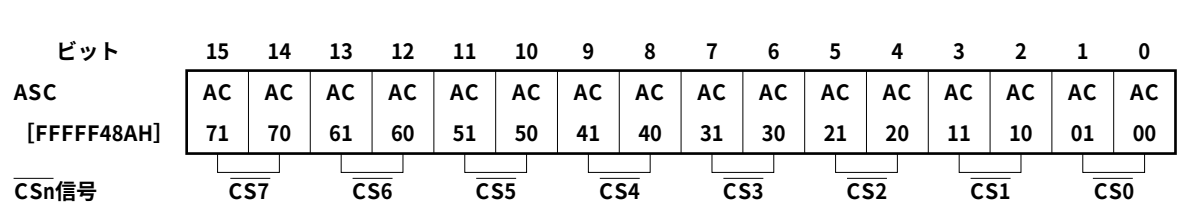
$\overline{CS2}$空間をデータ・バス幅8ビットに指定
BSCの設定値：0x0x0x0x0x000x0xB

図2－35　データ・ウエイト・コントロール・レジスタ0（DWC0）の設定



$\overline{CS2}$空間のウエイト：6
DWC0の設定値：0xxx01100xxx0xxxB

図2－36　アドレス・セットアップ・ウエイト・コントロール・レジスタ（ASC）の設定



$\overline{CS2}$空間のアドレス・セットアップ・ウエイト：1
ASCの設定値：xxxxxxxxxx01xxxxB

図2－37　バス・サイクル・コントロール・レジスタ（BCC）の設定

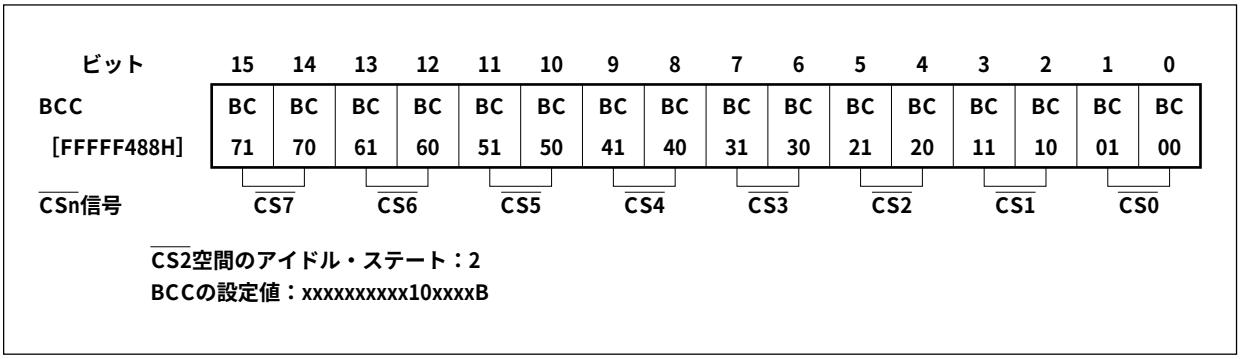


図2－38　SST29LE010-150接続回路例

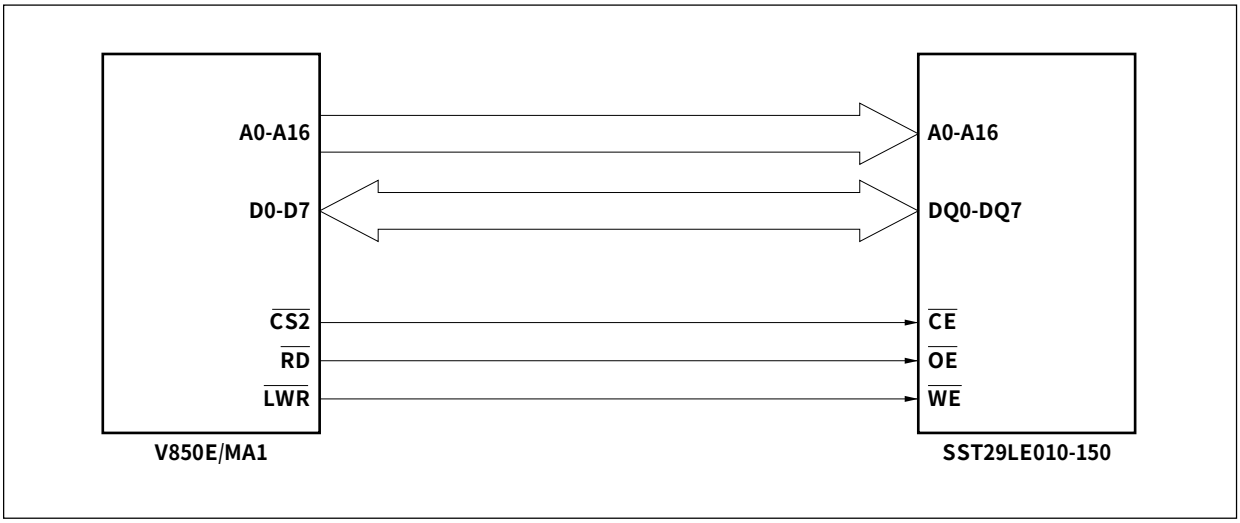
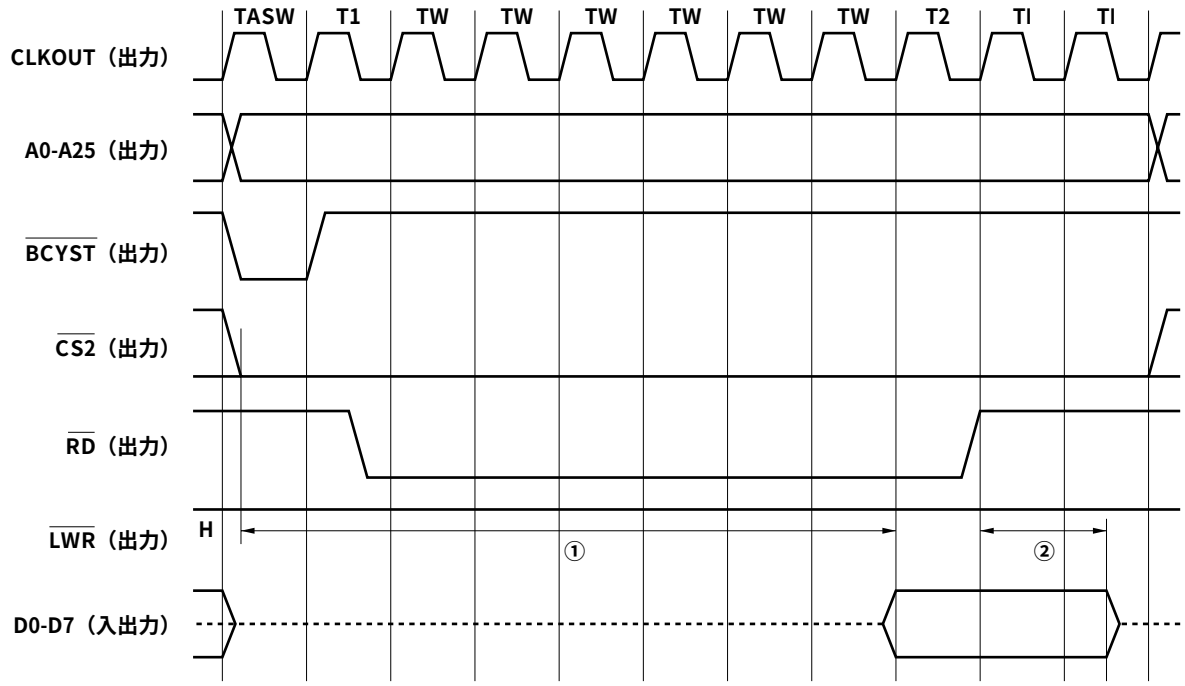


図2－39　SST29LE010-150のリード動作

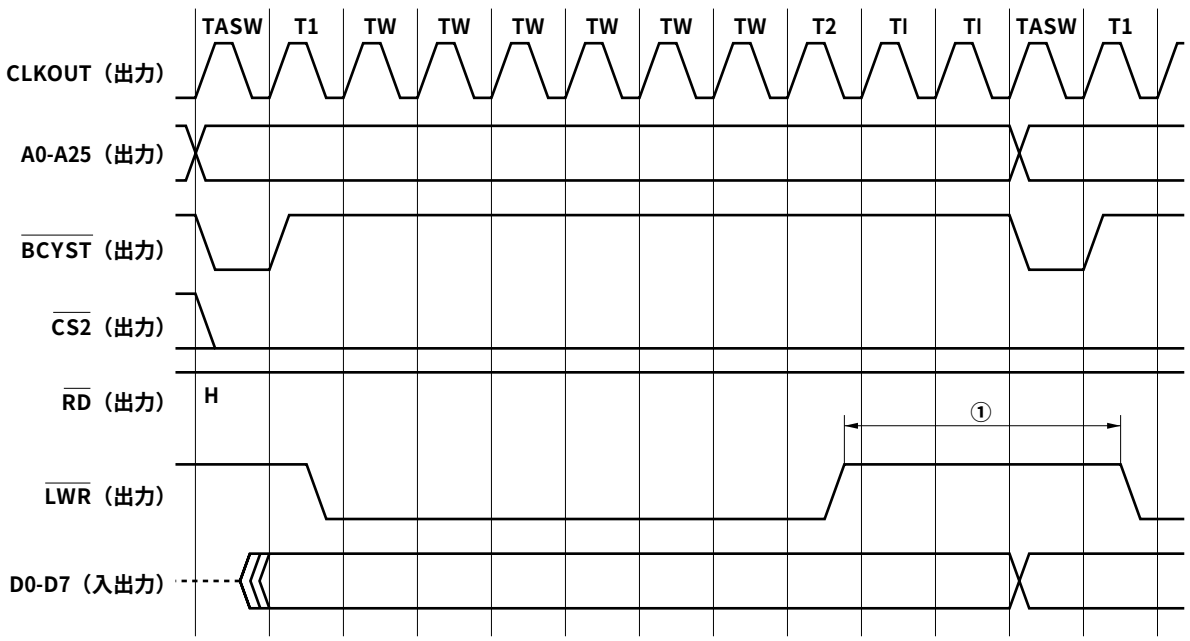


① SST29LE010のCE信号アクティブからのデータ出力遅延時間：150 ns（MAX.）

② SST29LE010のOE信号インアクティブからのデータ出力フローティング遅延時間：30 ns（MAX.）

備考　破線はハイ・インピーダンスを示しています。

図2ー40　SST29LE010-150のライト動作



① SST29LE010-150の$\overline{WE}$信号ハイ・レベル・パルス幅：50 ns（MIN.）

備考　破線はハイ・インピーダンスを示します。

2. 5　EDO DRAMとの接続

EDO DRAM（HM5164165FL：4 M×16ビット）を1つ（16ビット・バス幅，8 Mバイトの外部メモリ空間）接続する例を示します。

【回路構成】

- 内部システム・クロック：50 MHz
- 接続デバイス：HM5164165FL-5×1つ
- 使用$\overline{\text{CS}}$信号：$\overline{\text{CS1}}$（$\overline{\text{RAS1}}$）
  外部メモリ空間の0800000H-0FFFFFFHに配列することとします。
  1000000H-3FFBFFFHはイメージとなります。

【接続の考え方と注意点】

- V850E/MA1の$\overline{\text{CS1}}$（$\overline{\text{RAS1}}$）端子をEDO DRAMの$\overline{\text{RAS}}$端子に接続しますのでアクセス・タイミングはDRC1に設定します。
- 16ビット・バス幅で接続しますのでV850E/MA1のアドレス端子（A1-A13）をHM5164165FLのA0-A12に接続します。
- HM5164165FLのアドレスはロウ・アドレスが13ビット，カラム・アドレスが9ビットなのでロウ・アドレス出力時にV850E/MA1のA1-A13が次の状態になるようにアドレス・マルチプレクス幅（= 9ビット）を設定します。

ロウ・アドレス	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1
	a22	－	－	－	－	－	－	－	－	－	－	a11	a10

- CBRリフレッシュ・サイクルは4096サイクル/64 msなのでリフレッシュ・インターバルが15.6 μs以下になるように設定します。
- この例では$\overline{\text{RAS}}$アクセス時間の設定でt_{RH} = 2, t_{DA} = 1と設定してもt_{RH} = 1, t_{DA} = 2と設定しても仕様を満足するのでt_{RH} = 2, t_{DA} = 1とします。このように，t_{RH}やt_{DA}など，どちらか一方にウエイト数を挿入すれば十分な場合は，発生頻度の少ない方に挿入します。

表2－3に，この回路例で必要なウエイト設定においてのV850E/MA1とHM5164165FL-5とのAC特性を比較します。

表2－3　V850E/MA1とHM5164165FL-5とのAC特性

項　　目	V850E/MA1	HM5164165FL-5
$\overline{RAS}$パルス幅（MIN.）	t_{RASP} ：（2＋w_{RH}＋w_{DA}）T－10 = 70 ns	（>）50 ns
$\overline{RAS}$プリチャージ時間（MIN.）	t_{RP} ：w_{RP}×T－10 = 30 ns	(=) 30 ns
ロウ・アドレス保持時間（MIN.）	t_{RAH} ：（0.5＋w_{RH}）T－10 = 20 ns	（>）8 ns
$\overline{RAS}$アクセス時間（MAX.）	t_{RAC} ：（2＋w_{RH}＋w_{DA}）T－21 = 59 ns	（>）50 ns
$\overline{CAS}$パルス幅（MIN.）	t_{HCAS} ：（0.5＋w_{DA}）T－10 = 20 ns	（>）8 ns
$\overline{CAS}$プリチャージ時間（MIN.）	t_{CP} ：（0.5＋w_{CP}）T－10 = 20 ns	（>）8 ns
カラム・アドレス保持時間（MIN.）	t_{CAH} ：（0.5＋w_{DA}）T－10 = 20 ns	（>）8 ns
$\overline{CAS}$アクセス時間（MAX.）	t_{CAC} ：（1＋w_{DA}）T－21 = 19 ns	（>）13 ns
$\overline{OE}$↑データ出力遅延時間（MIN.）	t_{DRDOD} ：（1＋i）T－10 = 30 ns	（>）13 ns
データ保持時間（対$\overline{CAS}$↓）（MIN.）	t_{DH} ：（0.5＋w_{DA}）T－10 = 20 ns	（>）8 ns

備考1.　主要なAC特性のみを比較したものです。50 MHz時，T = 20 nsとなります。

2.　T = t_{CYK}（CLKOUT出力周期）

w_{RP}（ロウ・アドレス・プリチャージ・ウエイト）= 2

w_{RH}（ロウ・アドレス・ホールド・ウエイト）= 1

w_{DA}（データ・アクセス・ウエイト）= 1

w_{CP}（$\overline{CAS}$プリチャージ・ウエイト）= 1

i（アイドル・ステート）= 1

【レジスタの設定】

- $\overline{CS1}$のデバイス・タイプ：EDO DRAM
- $\overline{CS1}$空間のバス幅：16ビット
- アイドル・ステート：1ステート
- オンページ・アクセス：許可
- $\overline{RAS}$ホールド・モード：許可

備考　EDO DRAMアクセスではDWC0レジスタの設定値は意味を持ちません。

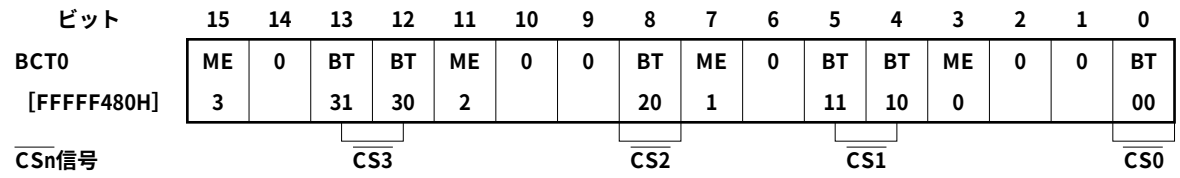
図2－41　チップ・エリア選択コントロール・レジスタ0（CSC0）の設定

ビット	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSC0	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS
[FFFFF060H]	33	32	31	30	23	22	21	20	13	12	11	10	03	02	01	00

CSC0の設定値：xxxxxxxxxxxxxxxxB

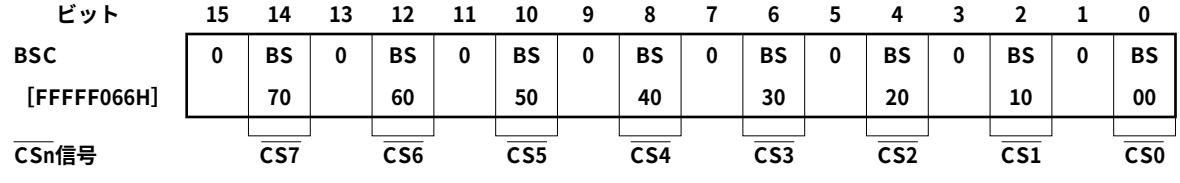
注意　ブロック0-3を使用する場合はアクセス時に$\overline{CS1}$が出力されない値を設定してください。

図2－42　バス・サイクル・タイプ・コンフィギュレーション・レジスタ0（BCT0）の設定



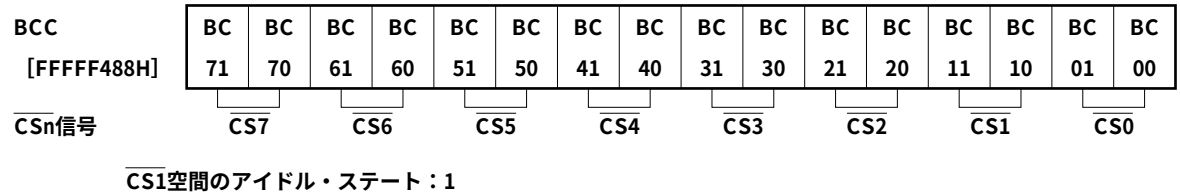
$\overline{CS1}$空間をメモリ・コントローラ動作許可，EDO DRAMに指定
BCT0の設定値：x0xxx00x1010x00xB

図2－43　バス・サイズ・コンフィギュレーション・レジスタ（BSC）の設定



$\overline{CS1}$空間をデータ・バス幅16ビットに指定
BSCの設定値：0x0x0x0x0x0x010xB

図2－44　バス・サイクル・コントロール・レジスタ（BCC）の設定



$\overline{CS1}$空間のアイドル・ステート：1
BCCの設定値：xxxxxxxxxxxx01xxB

図2－45　DRAMコンフィギュレーション・レジスタ1（SCR1）の設定

ビット	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCR1	PAE	0	RPC	RPC	RHC	RHC	DAC	DAC	CPC	CPC	0	RHD	ASO	ASO	DAW	DAW
[FFFFF4A4H]	11		11	01	11	01	11	01	11	01		1	11	01	11	01

PAE11　　：1（オンページ・アクセス許可）
RPC11, RPC01　：1, 0（ロウ・アドレスのプリチャージ・ウエイト数：W_{RP} = 2）
RHC11, RHC01　：0, 1（ロウ・アドレスのホールド・ウエイト数：W_{RH} = 1）
DAC11, DAC01　：0, 1（データ・ウエイト数：W_{DA} = 1）
CPC11, CPC01　：0, 1（カラム・アドレスのプリチャージ・ウエイト数：W_{CP} = 1）
RHD1　　：0（$\overline{RAS}$ホールド・モードを許可）
ASO11, ASO01　：0, 1（データ・バス幅：16ビット）
DAW11, DAW01　：0, 1（アドレス・マルチプレクス幅：9ビット）
SCR1の設定値　：A545H

図2－46　リフレッシュ・コントロール・レジスタ1（RFS1）の設定

ビット	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RFS1	REN	0	0	0	0	0	RCC	RCC	0	0	RIN	RIN	RIN	RIN	RIN	RIN
[FFFFF4A6H]	1						11	01			15	14	13	12	11	10

REN1　　：1（リフレッシュ許可）
RCC11, RCC01　：0, 0（リフレッシュ・カウント・クロック：32/f_{XX}）
RIN15-RIN10　：0, 1, 0, 1, 1, 1（インターバル・ファクタ：24）
RFS1の設定値　：8017H
f_{XX} = 50 MHz（0.02 μs）なので上記設定でのリフレッシュ間隔は次のようになります。
　リフレッシュ間隔　= 24×0.02×32
　　= 15.4 μs（＜15.6 μs）

図2－47　リフレッシュ・ウエイト・コントロール・レジスタ（RWC）の設定

ビット	7	6	5	4	3	2	1	0
RWC [FFFFF49EH]	RRW1	RRW0	RCW2	RCW1	RCW0	SRW2	SRW1	SRW0

RRW1, RRW0：1, 0（リフレッシュ$\overline{\text{RAS}}$ウエイト数：RRW = 2）

RCW2-RCW0：0, 0, 0（リフレッシュ・サイクル・ウエイト数：RCW = 1（必ず1ウエイト挿入されます））

SRW2-SRW0：0, 0, 1（セルフ・リフレッシュ解除ウエイト数：SRW = 1）

RWCの設定値：41H

注意1. HM5164165FL-5のセルフ・リフレッシュ解除時の$\overline{\text{RAS}}$プリチャージ時間：90 ns（MIN.）

V850E/MA1の電気的特性より

セルフ・リフレッシュ解除時の$\overline{\text{RAS}}$プリチャージ時間の最小値

t_{RPS}（ns）=（2＋2w_{SRW}＋w_{RRW}）T－10

= 6×20－10　　；w_{SRW} = 1, w_{RRW} = 2, T = 20

= 110 ns（＞90 ns）

2. ほかに接続するEDO DRAMがある場合，RRW, RCWの設定は遅いメモリに合わせます。

備考　T：t_{CYK}（CLKOUT出力周期）

w_{RRW}：RWCレジスタのRRW0, RRW1ビットによるウエイト数

w_{SRW}：RWCレジスタのSRW0-SRW2ビットによるウエイト数

図2－48　HM5164165FL-5接続回路例

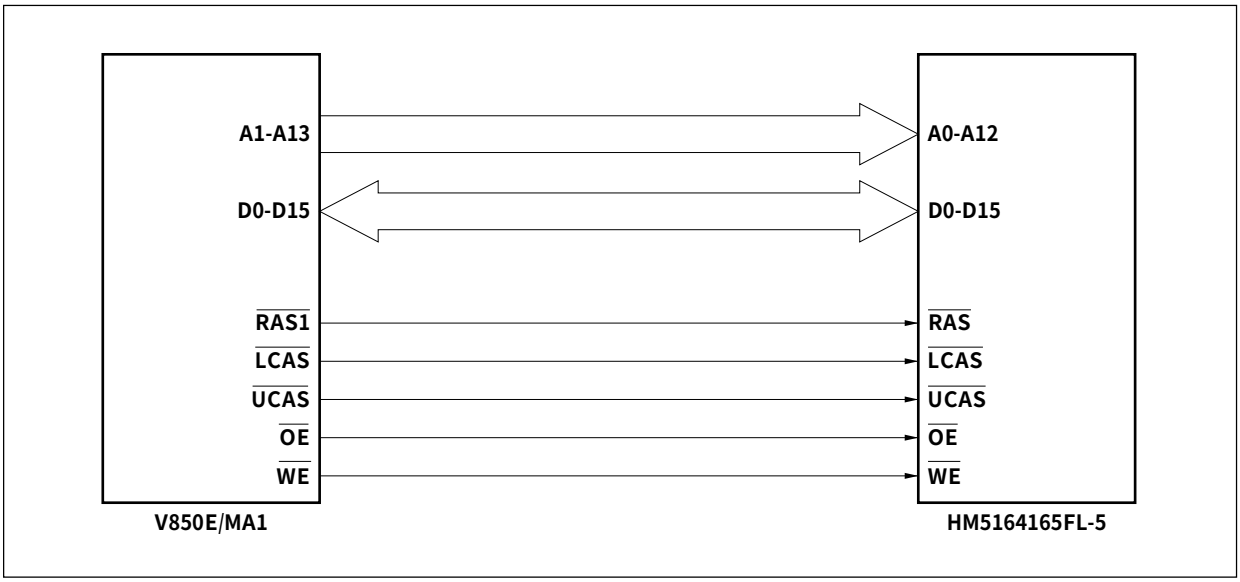
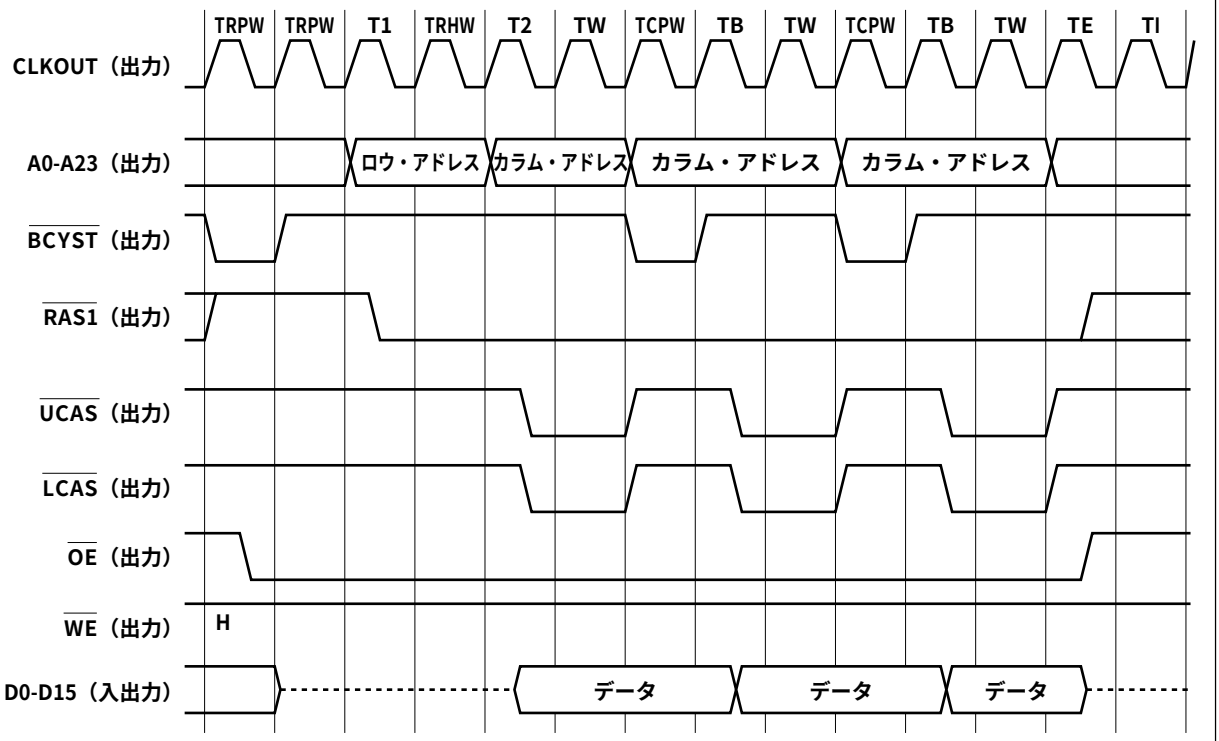
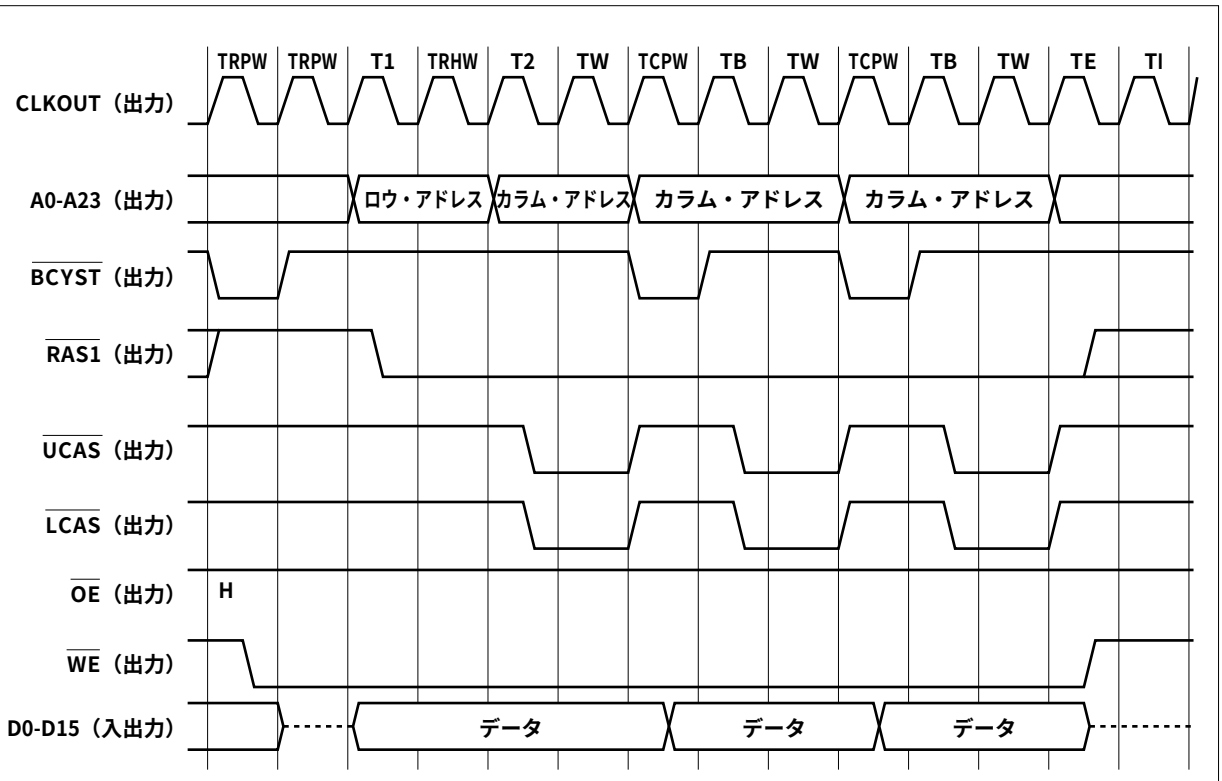


図2－49　HM5164165FL-5のリード動作



備考　破線はハイ・インピーダンスを示します。

図2－50　HM5164165FL-5のライト動作



備考　破線はハイ・インピーダンスを示します。

2. 6　SDRAMとの接続

SDRAM（μPD4564163G5：1 M×16ビット×4バンク）を1つ（8 Mバイトの外部メモリ空間）接続する例を示します。

【回路構成】

- 内部システム・クロック：50 MHz
- 接続デバイス：μPD4564163G5-A10×1つ
- 使用$\overline{CS}$信号：$\overline{CS3}$

  外部メモリ空間の4000000H-47FFFFFHに配列することとします（4800000H-7FFFFFFHは8 Mバイト単位のイメージとなります）。

注意　この空間にプログラムは配列できません。プログラム・エリアとして使用する場合は下位64 Mバイト空間に配列します。

【接続の考え方と注意点】

- V850E/MA1の$\overline{CS3}$端子をSDRAMの$\overline{CS}$端子に接続しますのでアクセス・タイミングはSCR3レジスタに設定します。
- ★ 16ビット・バス幅で接続しますのでV850E/MA1のアドレス端子A1-A12, A21, A22をμPD4564163G5のA0-A13に接続します。
- μPD4564163G5のアドレスはロウ・アドレスが12ビット，カラム・アドレスが8ビットなのでロウ・アドレス出力時にV850E/MA1のA1-A12が次の状態になるようにアドレス・マルチプレクス幅（8ビット）を設定します。

ロウ・アドレス	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1
	a20	– –									a10	a9

★ **注意**　V850E/MA1のA21, A22はμPD4564163G5のバンク・セレクト・アドレスになります。

- μPD4564163G5-A10のリフレッシュ・サイクルは4096サイクル/64 msなので，リフレッシュ・インターバルが15.6 μs以下になるように設定します。
- μPD4564163G5-A10のAC特性は次のとおりです。したがって，SCR3レジスタは$\overline{CAS}$レーテンシ = 2，BCW = 1で使用し，アイドル・ステートは0を設定します。
- $\overline{CAS}$レーテンシ = 2での最大クロック：77 MHz
- リフレッシュ（またはアクティブ）コマンドから次のリフレッシュ（またはアクティブ）コマンドまでの最小時間：70 ns
- 同一バンクでのアクティブ・コマンドからリード・ライト・コマンドまでの最小時間：20 ns
- クロック立ち上がりからのデータ・フロート遅延時間[注]（MAX.）：7 ns

注　V850E/MA1のAC特性（t_{DSDOD}）=（1＋i）T－5

= 15 ns（＞7 ns）　i = 0, T = 20 ns

【レジスタの設定】

- $\overline{CS3}$**のデバイス・タイプ：**SDRAM
- $\overline{CS3}$**空間のバス幅：**16**ビット**
- **アイドル・ステート：**0**ステート**

備考　SDRAM**アクセスでは**DWC0**レジスタの設定値は意味を持ちません。また，**$\overline{CS3}$**空間の領域は固定されているため**CSC0**の設定値は意味を持ちません。

図2－51　**バス・サイクル・タイプ・コンフィギュレーション・レジスタ**0（BCT0）**の設定**

ビット	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BCT0	ME	0	BT	BT	ME	0	0	BT	ME	0	BT	BT	ME	0	0	BT
[FFFFF480H]	3		31	30	2			20	1		11	10	0			00
$\overline{CS}$n信号			$\overline{CS3}$					$\overline{CS2}$			$\overline{CS1}$					$\overline{CS0}$

$\overline{CS3}$**空間をメモリ・コントローラ動作許可，SDRAMに指定**
BCT0の設定値：1011x00xx0xxx00xB

図2－52　**バス・サイズ・コンフィギュレーション・レジスタ（**BSC**）の設定**

ビット	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BSC	0	BS	0	BS	0	BS	0	BS	0	BS	0	BS	0	BS	0	BS
[FFFFF066H]		70		60		50		40		30		20		10		00
$\overline{CS}$n信号		$\overline{CS7}$		$\overline{CS6}$		$\overline{CS5}$		$\overline{CS4}$		$\overline{CS3}$		$\overline{CS2}$		$\overline{CS1}$		$\overline{CS0}$

$\overline{CS3}$**空間をデータ・バス幅16ビットに指定**
BSCの設定値：0x0x0x0x010x0x0xB

図2－53　**バス・サイクル・コントロール・レジスタ（**BCC**）の設定**

ビット	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BCC	BC	BC	BC	BC	BC	BC	BC	BC	BC	BC	BC	BC	BC	BC	BC	BC
[FFFFF488H]	71	70	61	60	51	50	41	40	31	30	21	20	11	10	01	00
$\overline{CS}$n信号		$\overline{CS7}$		$\overline{CS6}$		$\overline{CS5}$		$\overline{CS4}$		$\overline{CS3}$		$\overline{CS2}$		$\overline{CS1}$		$\overline{CS0}$

$\overline{CS3}$**空間のアイドル・ステート：0**
BCCの設定値：xxxxxxxx00xxxxxxB

図2－54　SDRAMコンフィギュレーション・レジスタ3（SCR3）の設定

ビット	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCR3	0	LTM	LTM	LTM	0	0	0	0	BCW	BCW	SSO	SSO	RAW	RAW	SAW	SAW
[FFFFF4ACH]		23	13	03					13	03	13	03	13	03	13	03

LTM23-LTM03　：0, 1, 0（CASレーテンシ = 2）
BCW13, BCW03 ：0, 1（コマンド間ウエイト数：BCW = 1）
SSO13, SSO03　：0, 1（データ・バス幅 = 16ビット）
RAW13, RAW03 ：0, 1（ロウ・アドレス幅 = 12）
SAW13, SAW03 ：0, 0（アドレス・マルチプレクス幅 = 8ビット）
SCR3の設定値　：2054H

図2－55　リフレッシュ・コントロール・レジスタ3（RFS3）の設定

ビット	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RFS3	REN	0	0	0	0	0	RCC	RCC	0	0	RIN	RIN	RIN	RIN	RIN	RIN
[FFFFF4AEH]	1						11	01			15	14	13	12	11	10

REN1　：1（リフレッシュ許可）
RCC11, RCC01　：0, 0（リフレッシュ・カウント・クロック：32/fxx）
RIN15-RIN10　：0, 1, 0, 1, 1, 1（インターバル・ファクタ：24）
RFS3の設定値　：8017H
fxx = 50 MHz（0.02 μs）なので上記設定でのリフレッシュ間隔は次のようになります。
リフレッシュ間隔 = 24×0.02×32
= 15.4 μs（＞15.6 μs）

図2－56　μPD4564163G5-A10接続回路例

★

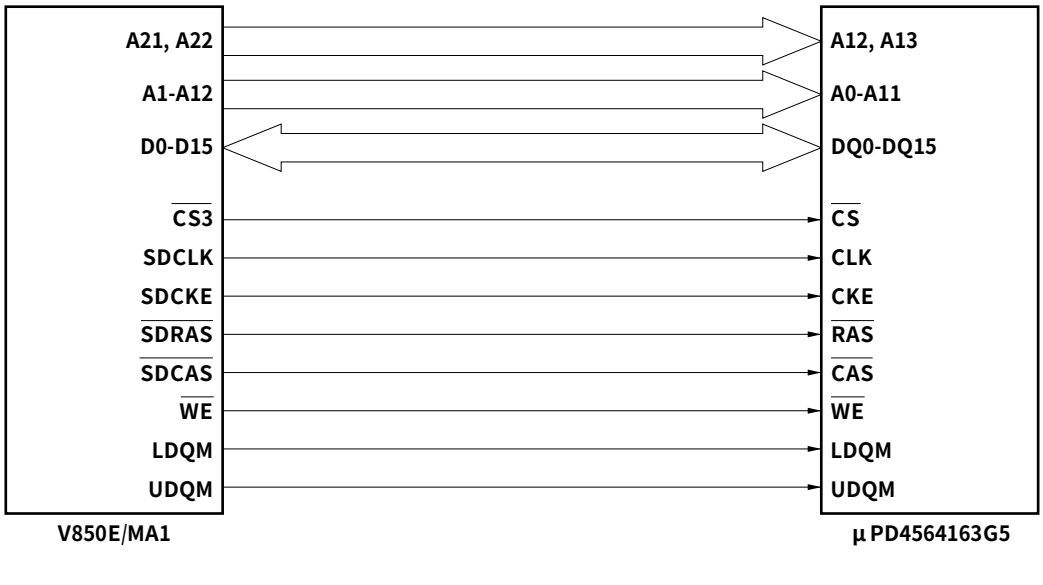
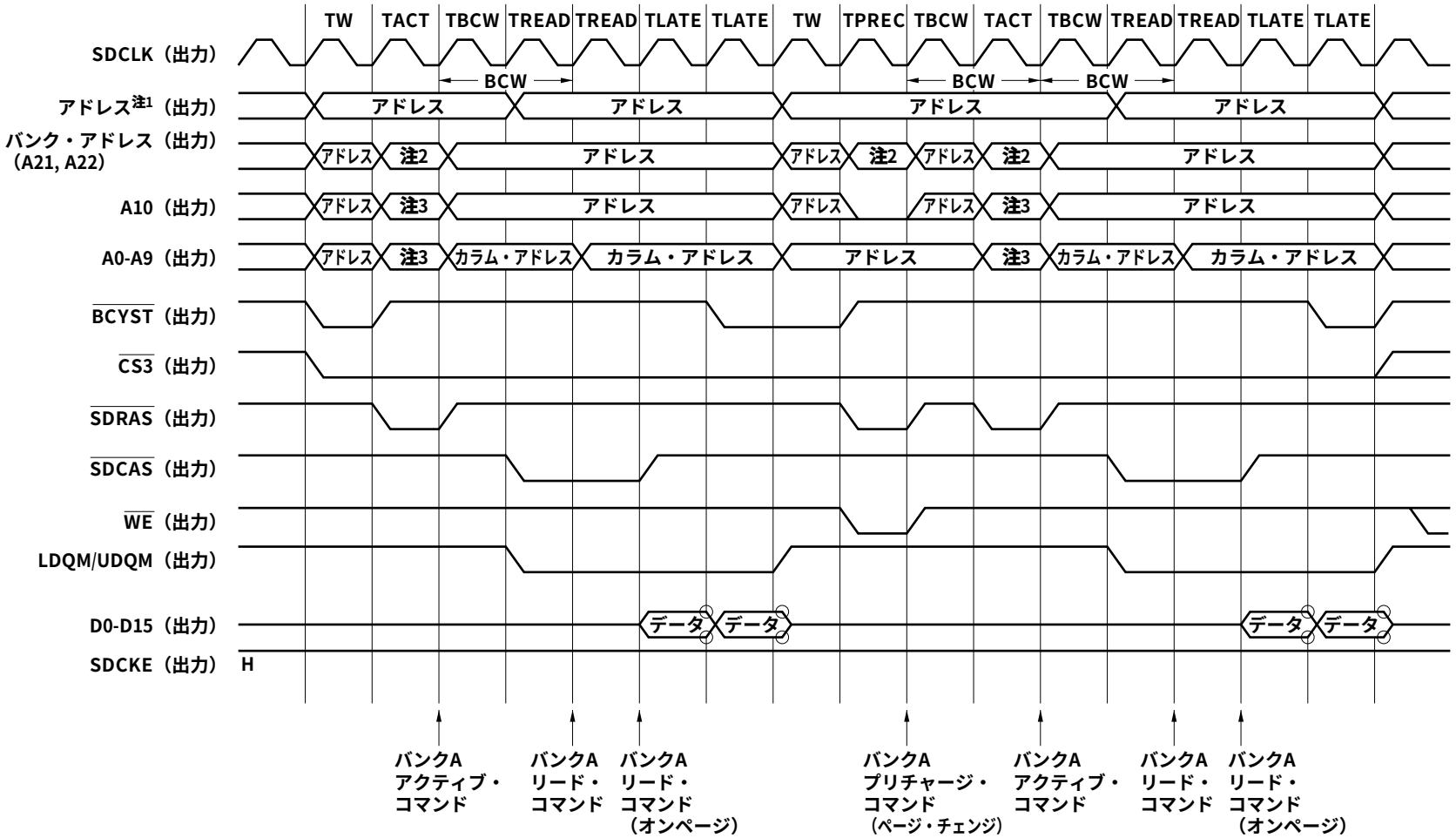


図2－57　μPD4564163G5-A10のリード動作★



注1.　A0-A9, A10，**バンク・アドレス以外**

2.　**バンク・アドレス**

3.　**ロウ・アドレス**

第3章　バス・インタフェース接続回路例（2）

V850E/MA1のバス・インタフェースに直結できないメモリを，付加回路を使用して接続する例を説明します。回路例は次に示す考え方で構成しています。

- V850E/MA1の内部システム・クロックおよびバス・サイクルは50 MHz
- 特に断らないかぎり$\overline{\text{WAIT}}$端子は未使用
- BCPレジスタは8xHを設定，BECレジスタは任意
- 3. 5　$\overline{\text{HLDRQ}}$信号を使用した外部リフレッシュ・ユニットとの接続を除いて外部バス・マスタは接続しない（3. 5以外$\overline{\text{HLDRQ}}$信号は使用しない）

また付加回路による信号伝達遅延時間を2 ns（MIN.），7 ns（MAX.）として計算しています。実際の回路では使用するデバイスにあわせて遅延時間を考慮してください。

この章では【レジスタの設定】を第2章に比べて簡略化しています。

3. 1　16ビットSRAMとの接続

SRAM（μPD434016ALE：256 K×16ビット）を1つ使用し，512 Kバイトの空間を接続する例を示します。

【回路構成】

- 内部システム・クロック：50 MHz
- 接続デバイス：μPD434016ALE-12×1つ
- 使用$\overline{\text{CS}}$信号：$\overline{\text{CS5}}$

  外部メモリ空間のFA00000H-FA7FFFFH（ブロック5）に配列することとします。

【接続の考え方と注意点】

μPD434016ALE-12のアドレス・バスはV850E/MA1のA1-A18に接続します。$\overline{\text{CS}}$端子，$\overline{\text{OE}}$端子，$\overline{\text{LB}}$端子，$\overline{\text{UB}}$端子はV850E/MA1の$\overline{\text{CS5}}$端子，$\overline{\text{RD}}$端子，$\overline{\text{LBE}}$端子，$\overline{\text{UBE}}$端子にそれぞれ接続します。また，V850E/MA1にはSRAM用の$\overline{\text{WE}}$信号が用意されていませんので，μPD434016ALE-12の$\overline{\text{WE}}$端子は，$\overline{\text{LWR}}$端子，$\overline{\text{UWR}}$端子により作成して接続します。

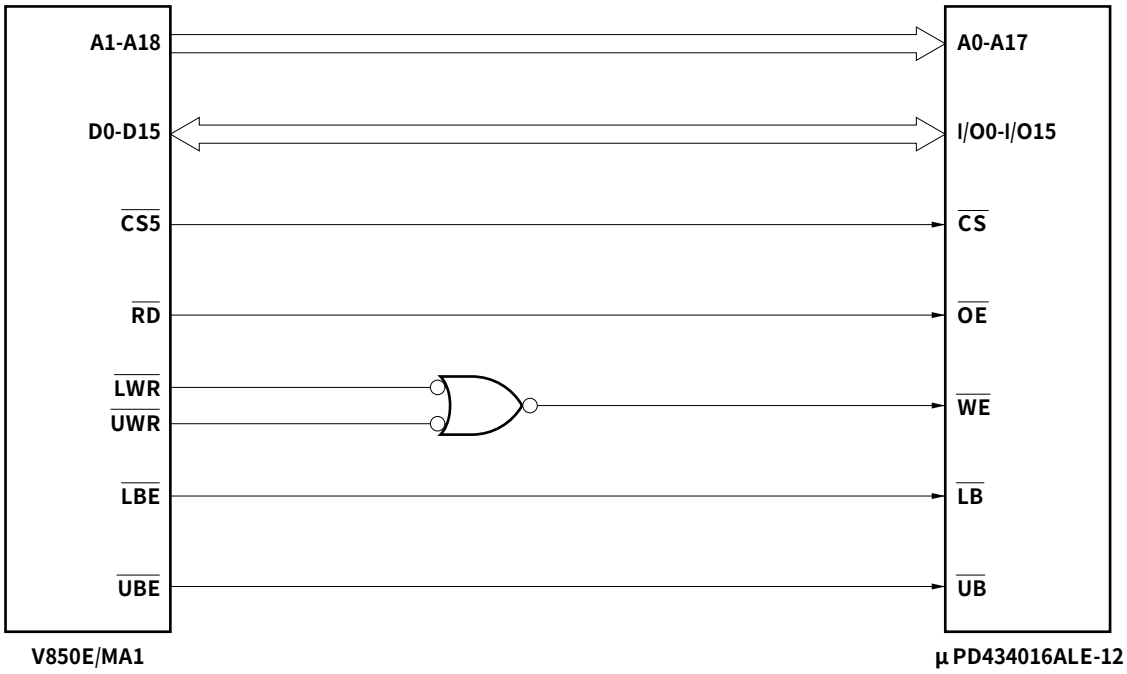
注意　$\overline{\text{LBE}}$信号は$\overline{\text{SDRAS}}$信号と，$\overline{\text{UBE}}$信号は$\overline{\text{SDCAS}}$信号と機能がマルチプレクスしています。$\overline{\text{UBE}}$端子，$\overline{\text{LBE}}$端子として機能させるにはPFCCDレジスタのビット2，3の設定で$\overline{\text{LBE}}$信号，$\overline{\text{UBE}}$信号の機能を選択する必要があります。V850E/MA1にSDRAMと16ビットSRAMの両方を接続するには$\overline{\text{SDRAS}}$信号，$\overline{\text{SDCAS}}$信号を選択しますので$\overline{\text{UBE}}$信号，$\overline{\text{LBE}}$信号はV850E/MA1の$\overline{\text{RD}}$端子，$\overline{\text{LWR}}$端子，$\overline{\text{UWR}}$端子で作成します（図3－1を参照）。

【レジスタの設定】

表3ー1　16ビットSRAMとの接続

レジスタ名称	設定値	機　　能
CSC1	xxxx0100xxxxx0xxB	ブロック5：$\overline{CS5}$出力
BCT1	x00xx0xx1000x0xxB	$\overline{CS5}$：メモリ・コントローラ許可，SRAM，外部I/O
BSC	0x0x010x0x0x0x0xB	$\overline{CS5}$：16ビット
DWC1	0xxx0xxx00010xxxB	$\overline{CS5}$：1ウエイト
ASC	xxxx00xxxxxxxxxxB	$\overline{CS5}$：アドレス・セットアップ・ウエイト0
BCC	xxxx01xxxxxxxxxxB	$\overline{CS5}$：アイドル・ステート1

図3ー1　μPD434016ALE-12接続回路例



（参考回路）SDRAMとの競合でUBE信号，LBE信号が使用できない場合の接続回路例

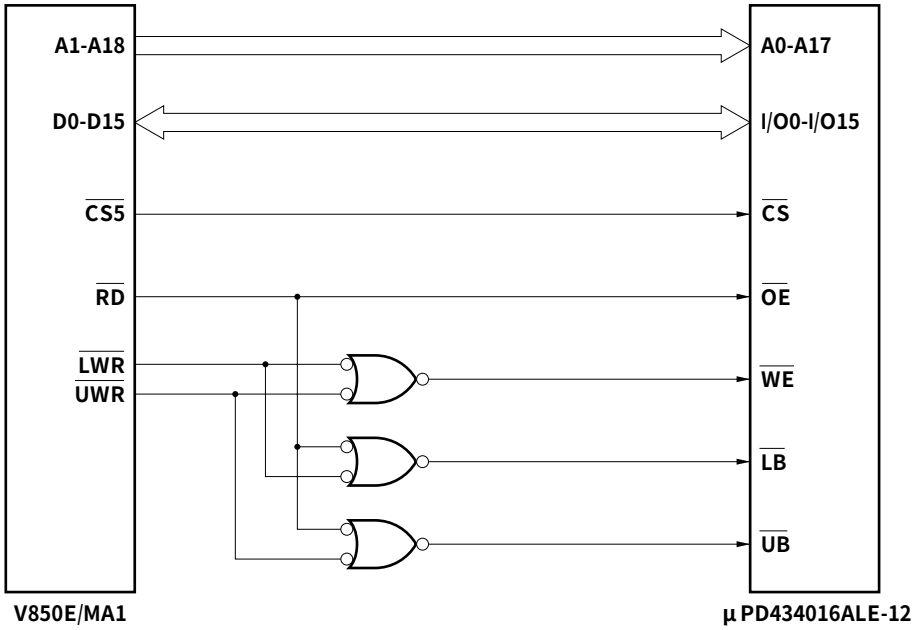


図3－2　μPD434016ALE-12のリード動作（16ビット・アクセス）（1/2）

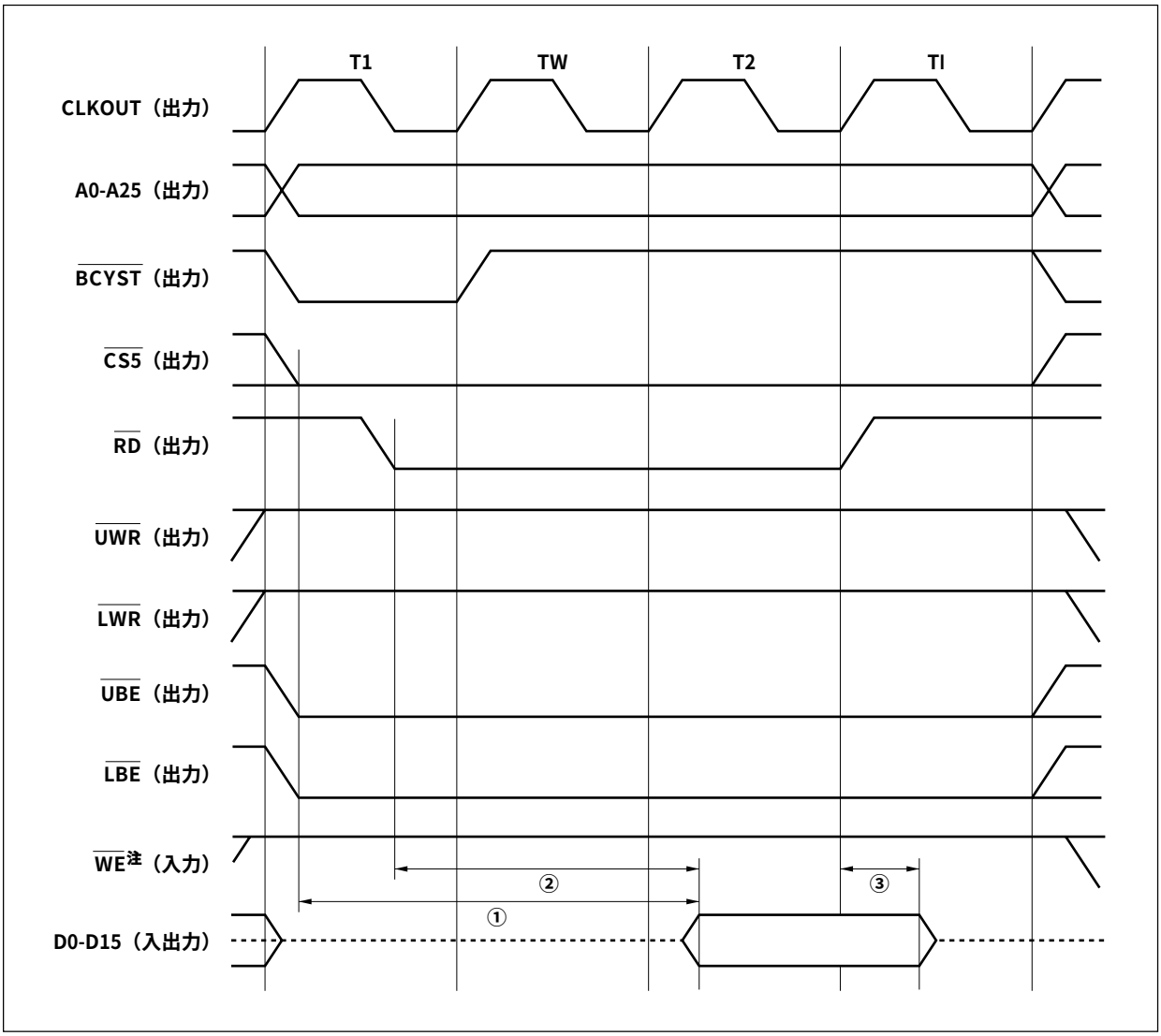


図3－2　μPD434016ALE-12のリード動作（16ビット・アクセス）（2/2）

① μPD434016ALE-12のアドレス，$\overline{CS}$アクティブからの出力遅延時間：12 ns（MAX.）
V850E/MA1の電気的特性よりデータ入力設定時間（対アドレス）の最大値

t_{SAID}（ns）　=（2＋w＋w_D＋w_{AS}）T－21
= 3×20－21；w = 0, w_D = 1, w_{AS} = 0, T = 20 ns
= 39 ns（＞12 ns）

② μPD434016ALE-12の$\overline{OE}$アクティブからの出力遅延時間：6 ns（MAX.）
V850E/MA1の電気的特性よりデータ入力設定時間（対$\overline{RD}$↓）の最大値

t_{SRDID}（ns）　=（1.5＋w＋w_D）T－21
= 2.5×20－21；w = 0, w_D = 1, T = 20 ns
= 29 ns（＞6 ns）

③ μPD434016ALE-12の$\overline{OE}$インアクティブからの出力フローティング遅延時間：6 ns（MAX.）
V850E/MA1の電気的特性よりデータ出力遅延時間（対$\overline{RD}$↑）の最小値

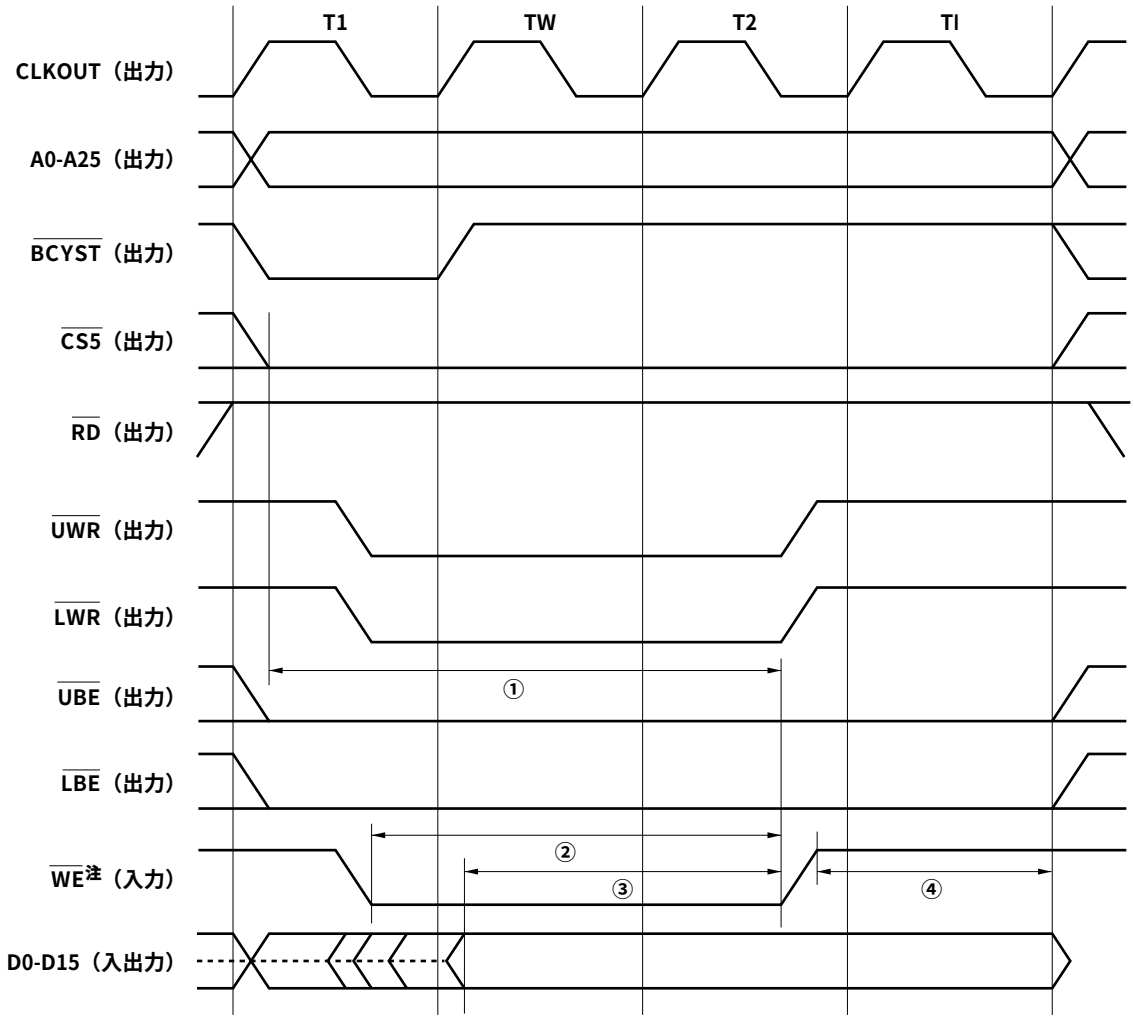
t_{DRDOD}（ns）　=（0.5＋i）T－10
= 1.5×20－10；i = 1, T = 20 ns
= 20 ns（＞6 ns）

注　$\overline{WE}$はμPD434016ALE-12の端子です。

備考1.　破線はハイ・インピーダンスを示します。

2.　T：t_{CYK}（CLKOUT出力周期）
w：$\overline{WAIT}$によるウエイト数
w_D：DWC1, DWC2によるウエイト数
w_{AS}：ASCによるアドレス・セットアップ・ウエイト数
i：アイドル・ステート数

図3－3　μPD434016ALE-12のライト動作（16ビット・アクセス）（1/2）



① μPD434016ALE-12のアドレス，CSアクティブからライト終了までの時間：8 ns（MIN.）
V850E/MA1の電気的特性より，アドレス設定時間（対UWR, LWR↑）の最小値

t_{SAWR}（ns）＝（1.5＋w＋w_D＋w_{AS}）T－10
＝ 2.5×20－10；w = 0, w_D = 1, w_{AS} = 0, T = 20 ns
＝ 40 ns（＞8 ns）

② μPD434016ALE-12のWEアクティブ・パルス幅：8 ns（MIN.）
V850E/MA1の電気的特性よりUWR, LWRロウ・レベル幅の最小値

t_{WWRL}（ns）＝（1＋w＋w_D）T－10
＝ 2×20－10；w_D = 1, w = 0, T = 20 ns
＝ 30 ns（＞8 ns）

注　WEはμPD434016ALE-12の端子です。

備考1.　破線はハイ・インピーダンスを示します。

2.　T：t_{CYK}（CLKOUT出力周期）
w：WAITによるウエイト数
w_D：DWC1, DWC2によるウエイト数
w_{AS}：ASCによるアドレス・セットアップ・ウエイト数

図3－3　μPD434016ALE-12のライト動作（16ビット・アクセス）（2/2）

③　μPD434016ALE-12のデータ・バリッドからライト終了までの時間：6 ns（MIN.）

V850E/MA1の電気的特性より，データ出力設定時間（対$\overline{\text{UWR}}$, $\overline{\text{LWR}}$↑）の最小値

t_{SODWR}（ns）＝（0.5＋w_{AS}+w＋w_D）T－10

＝ 1.5×20－10；w_D = 1, w = 0, T = 20 ns

＝ 20 ns（＞6 ns）

④　μPD434016ALE-12のライト終了からのデータ保持時間：0 ns（MIN.）

V850E/MA1の電気的特性よりデータ出力保持時間（対$\overline{\text{UWR}}$, $\overline{\text{LWR}}$↑）の最小値

t_{HWROD}（ns）＝（0.5＋i）T－10

＝ 1.5×20－10；i = 1, T = 20 ns

＝ 20 ns

$\overline{\text{WE}}$の付加回路による最大遅延時間を考慮すると

t_{HWROD}－付加回路の遅延時間　＝ 20－7

＝ 13 ns（＞0 ns）

備考　T：t_{CYK}（CLKOUT出力周期）

w：$\overline{\text{WAIT}}$によるウエイト数

w_D：DWC1, DWC2によるウエイト数

w_{AS}：ASCによるアドレス・セットアップ・ウエイト数

i：アイドル・ステート数

3. 2　低速デバイス（μPD4991A）との接続

カレンダ／時計機能IC（μPD4991A）を外部データ・バスに接続する例を示します。

【回路構成】

- 内部システム・クロック：50 MHz
- 接続デバイス：μPD4991A×1つ
- 使用$\overline{\text{CS}}$信号：$\overline{\text{CS7}}$

  外部メモリ空間のFE00000H-FE0000FH（ブロック7）に配列することとします。

  FE00010H-FFFBFFFHはイメージとなります。

【回路設計上の考え方と注意点】

- μPD4991Aを通常動作（V_{DD} = 3.3 V）で使用することとします。
- V_{DD} = 3.3 V時，μPD4991Aのアクセス・タイム（リード時$\overline{\text{CS}}$信号アクティブからデータ確定までの時間）は210 ns（MIN.）となります。V850E/MA1のアドレス・セットアップ・ウエイトとデータ・ウエイトを使用することによりアクセス・タイムを満足できます。
- V_{DD} = 3.3 V時，μPD4991Aのデータ出力フローティング遅延時間が70 ns（MAX.）と大きく，50 MHzで動作した場合，V850E/MA1のアイドル・ステート挿入機能では不足するためデータ・バスはバス・バッファを経由して接続します。
- V850E/MA1のA0-A3をμPD4991AのA0-A3に接続します。

  備考　μPD4991AのA0-A3をV850E/MA1のA0-A3に接続した場合は8ビット・バス幅を指定します。16ビット・バス幅空間に接続するときはV850E/MA1のA1-A4に接続します。いずれの場合もバイト・アクセスのみが有効（16ビット・バス幅でリトル・エンディアン動作時は偶数アドレスへのアクセス，ビッグ・エンディアン動作時は奇数アドレスへのアクセス）でデータは下位4ビットのみが有効となります。

- V850E/MA1の$\overline{\text{CS7}}$端子，$\overline{\text{RD}}$端子[注]，$\overline{\text{LWR}}$端子[注]をμPD4991Aの$\overline{\text{CS1}}$端子，$\overline{\text{OE}}$端子，$\overline{\text{WE}}$端子にそれぞれ接続します。

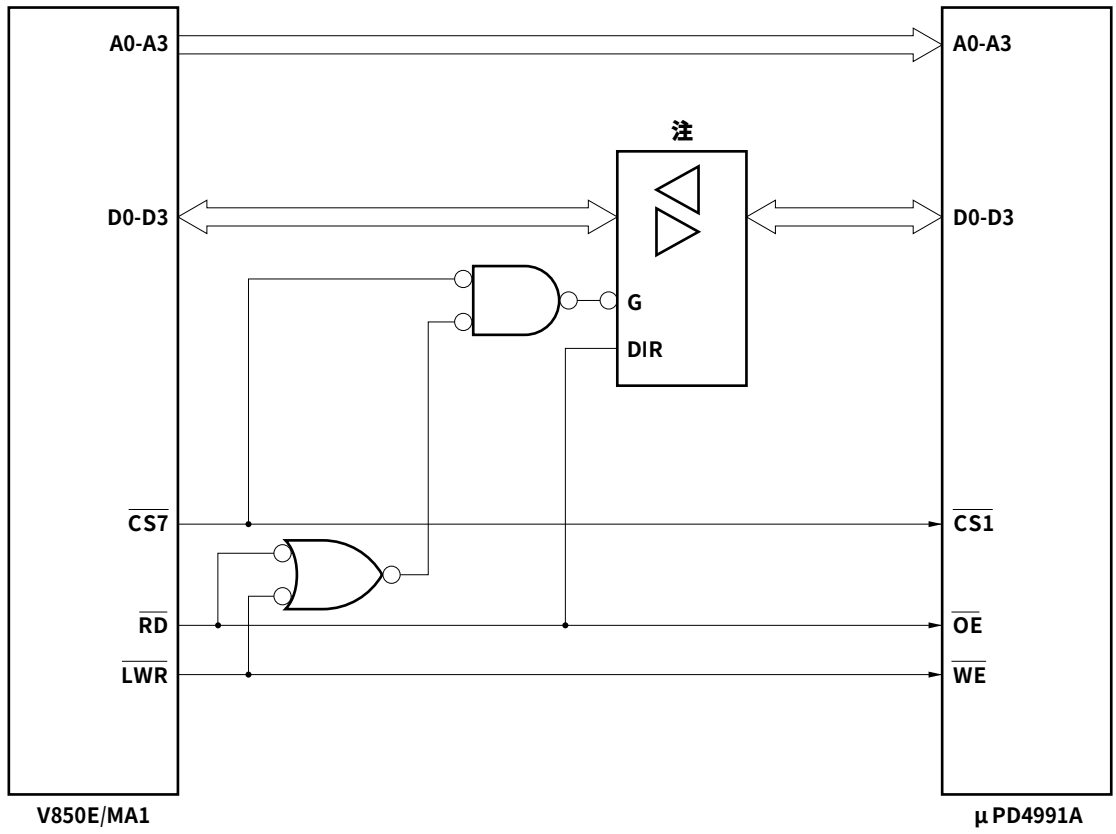
  注　BCPレジスタのIOENビットを1に設定し，$\overline{\text{IORD}}$端子，$\overline{\text{IOWR}}$端子を接続しても同等になります。$\overline{\text{RD}}$端子と$\overline{\text{IORD}}$端子および$\overline{\text{LWR}}$端子（または$\overline{\text{UWR}}$端子）と$\overline{\text{IOWR}}$端子の動作はDMAフライバイ転送時を除いて同等となります。

【レジスタの設定】

表3－2　低速デバイスとの接続

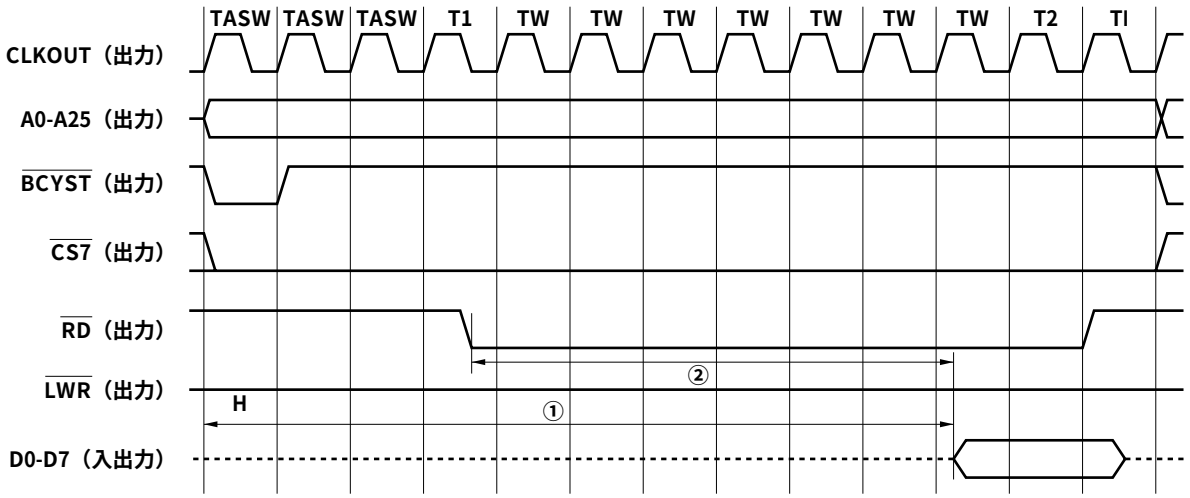
レジスタ名称	設定値	機　　能
CSC1	xxxxxxxxxxxxxxx1B	ブロック7：$\overline{CS7}$
BCT1	1000x0xxx00xx0xxB	$\overline{CS7}$：メモリ・コントローラ許可，SRAM，外部I/O
BSC	000x0x0x0x0x0x0xB	$\overline{CS7}$：8ビット
DWC1	01110xxx0xxx0xxxB	$\overline{CS7}$：データ・ウエイト7
ASC	11xxxxxxxxxxxxxxB	$\overline{CS7}$：アドレス・セットアップ・ウエイト3
BCC	01xxxxxxxxxxxxxxB	$\overline{CS7}$：アイドル・ステート1

図3－4　μPD4991A接続回路例



注　74xx245相当の回路を想定しています。

図3ー5　μPD4991Aのリード動作

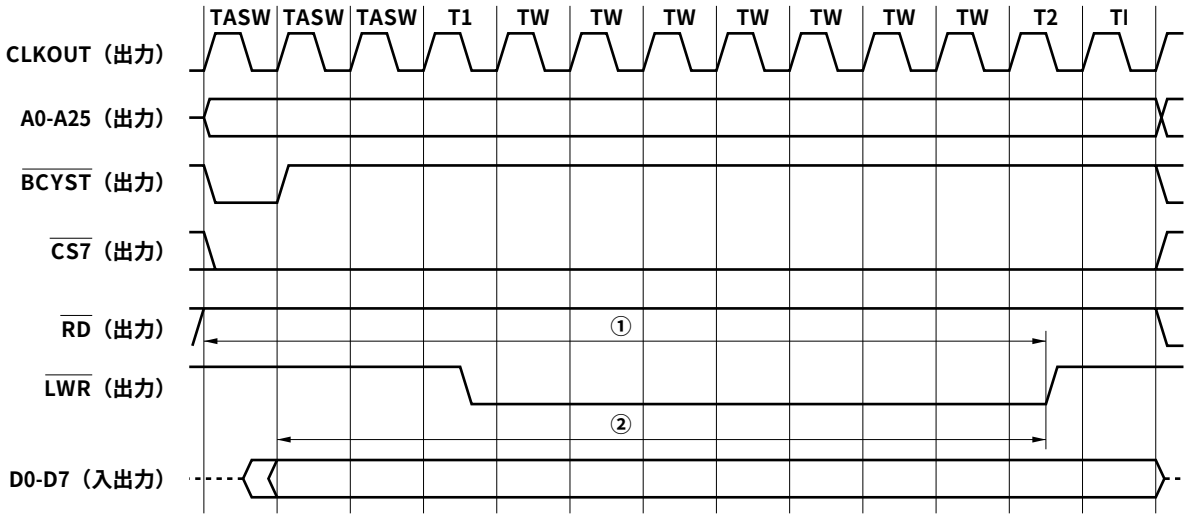


① μPD4991A（V_{DD} = 3.3 V）CS1信号アクティブからのデータ出力遅延時間：210 ns（MAX.）

② μPD4991A（V_{DD} = 3.3 V）OE信号アクティブからのデータ出力遅延時間：110 ns（MAX.）

備考　破線はハイ・インピーダンスを示します。

図3ー6　μPD4991Aのライト動作



① μPD4991A（V_{DD} = 3.3 V）CS1信号アクティブからライト終了までの時間：170 ns（MIN.）

② μPD4991A（V_{DD} = 3.3 V）データ・バリッドからライト終了までの時間：100 ns（MIN.）

備考　破線はハイ・インピーダンスを示します。

3. 3　$\overline{\text{WAIT}}$端子の制御

3. 3. 1　$\overline{\text{WAIT}}$端子の制御例（1）（μPD63310との接続）

ステレオ・サウンド・コーデック（μPD63310）を外部データ・バスに接続する例を示します。

μPD63310リード時の$\overline{\text{RB}}$信号（V850E/MA1の$\overline{\text{RD}}$端子に接続）に200 ns必要としますので$\overline{\text{WAIT}}$端子を使用した外部ウエイト制御によりV850E/MA1に接続します。

【回路構成】

- 内部システム・クロック：50 MHz
- 接続デバイス：μPD63310GK×1つ
- μPD63310GKをV_{DD} = 5 Vで使用
- 使用$\overline{\text{CS}}$信号：$\overline{\text{CS7}}$

  外部メモリ空間のFE00000H-FE00001H（ブロック7）に配列することとします。

【回路設計上の考え方と注意点】

- レベル変換のためのIC[注1]を経由し，V850E/MA1の$\overline{\text{CS7}}$端子，$\overline{\text{RD}}$端子[注2]，$\overline{\text{LWR}}$端子[注2]をμPD63310GKの$\overline{\text{CSB}}$端子，$\overline{\text{RB}}$端子，$\overline{\text{WB}}$端子にそれぞれ接続します。

  注1. μPD63310GKをV_{DD} = 5 Vで使用する場合，V_{IH} = 4.5 V（MIN.）ですのでV_{OH} = 4.5 V（MIN.）を満足できるデバイス（74ACシリーズなど）を経由して接続します。

  2. $\overline{\text{IORD}}$端子，$\overline{\text{IOWR}}$端子を接続しても同等です（3. 2　低速デバイス（μPD4991A）との接続参照）。

- リード時μPD63310GKの$\overline{\text{RB}}$信号は200 ns（MIN.），ライト時の$\overline{\text{WB}}$信号は120 ns（MIN.）のため，V850E/MA1の$\overline{\text{WAIT}}$端子に外部付加回路を接続します。
- 外部付加回路による$\overline{\text{WAIT}}$端子の制御はリード時のT1ステートの立ち上がりでアクティブにし，必要なウエイト数をカウントしたのちCLKOUTの立ち上がりでインアクティブになるように設計します。ライト時は$\overline{\text{WAIT}}$端子をアクティブにせず，ソフトウエアによるウエイト制御のみを使用します。したがって$\overline{\text{CS7}}$空間のソフトウエア・ウエイトはライト動作を満足する値（この例では6）を設定します。

  備考　ウエイト・サイクルは，$\overline{\text{WAIT}}$端子制御によるウエイト・サイクルとプログラマブル・ウエイトの設定によるサイクルの論理和として挿入され，どちらか多い方のウエイト・サイクル数だけ挿入されます。

  この例の方式ではT1ステート後に$\overline{\text{WAIT}}$端子がアクティブになりますので，リード時，ライト時の両方とも$\overline{\text{WAIT}}$端子によるウエイト制御を行う場合は，少なくとも最初の1ウエイト分はプログラマブル・ウエイトを挿入します。

- レベル変換のためICを経由して，V850E/MA1のD0-D5とA0をμPD63310GKのD0-D5, SELR（レジスタの指定）にそれぞれ接続します。

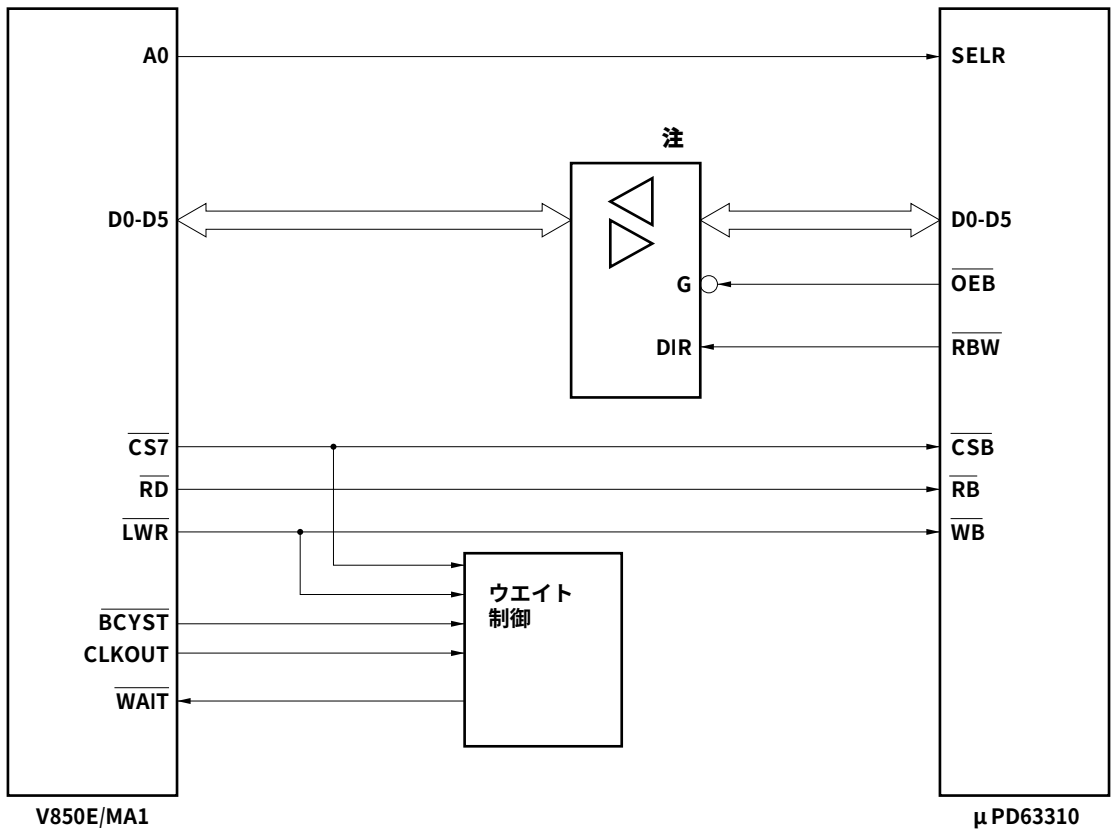
備考　μPD63310GKのSELR端子をV850E/MA1のA0に接続した場合は8ビット・バス幅を指定します。16ビット・バス幅空間に接続するときはV850E/MA1のA1に接続します。いずれの場合もバイト・アクセスのみが有効（16ビット・バス幅でリトル・エンディアン動作時は偶数アドレスへのアクセス，ビッグ・エンディアン動作時は奇数アドレスへのアクセス）で，データは下位5ビットのみが有効となります。

【レジスタの設定】

表3－3　μPD63310との接続

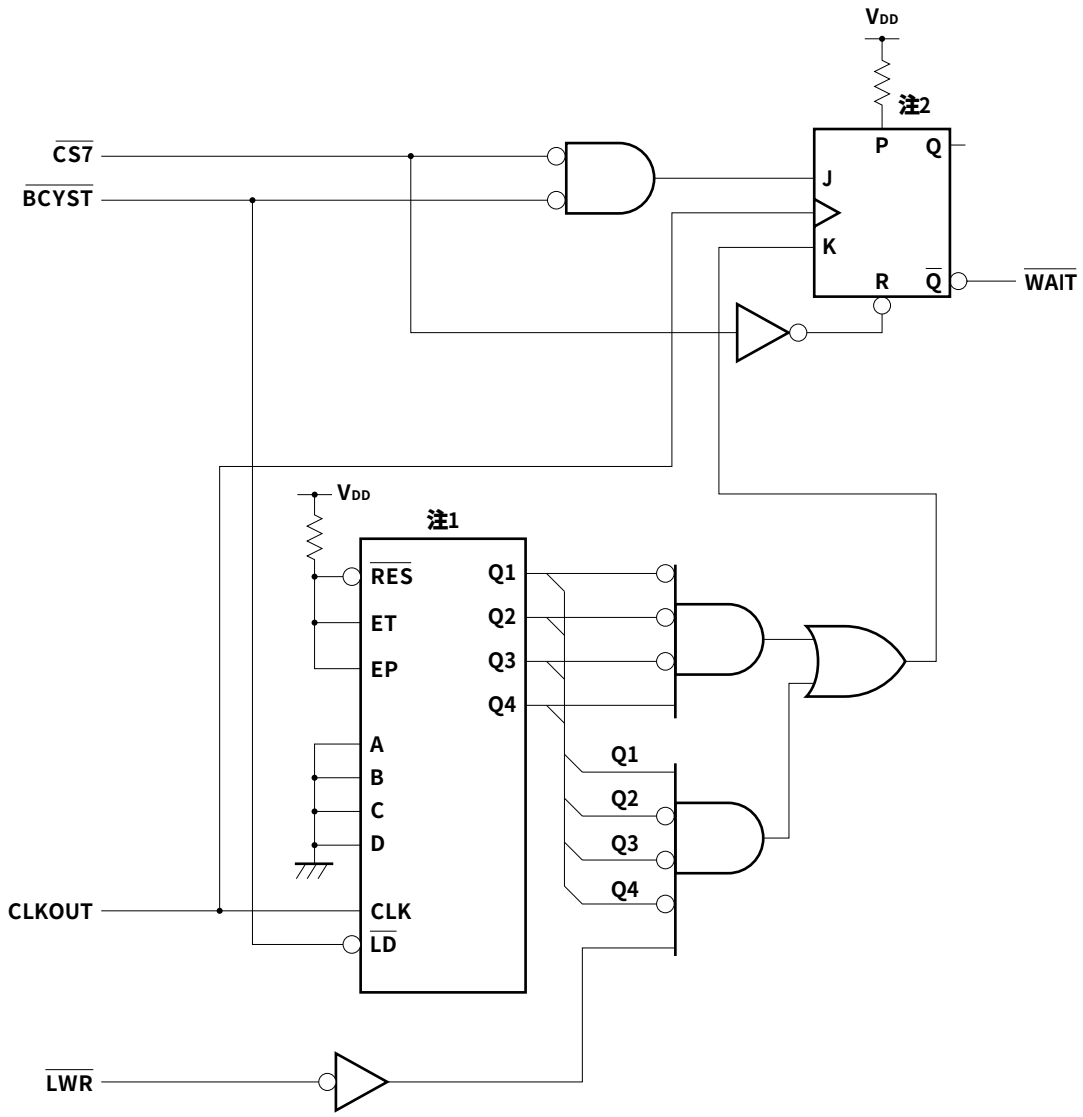
レジスタ名称	設定値	機　　能
CSC1	xxxxxxxxxxxxxxx1B	ブロック7：$\overline{CS7}$
BCT1	1000x0xxx00xx0xxB	$\overline{CS7}$：メモリ・コントローラ許可，SRAM，外部I/O
BSC	000x0x0x0x0x0x0xB	$\overline{CS7}$：8ビット
DWC1	01100xxx0xxx0xxxB	$\overline{CS7}$：データ・ウエイト6
ASC	00xxxxxxxxxxxxxxB	$\overline{CS7}$：アドレス・セットアップ・ウエイト0
BCC	10xxxxxxxxxxxxxxB	$\overline{CS7}$：アイドル・ステート2

図3－7　μPD63310接続回路例



注　74xx245相当の回路を想定しています。

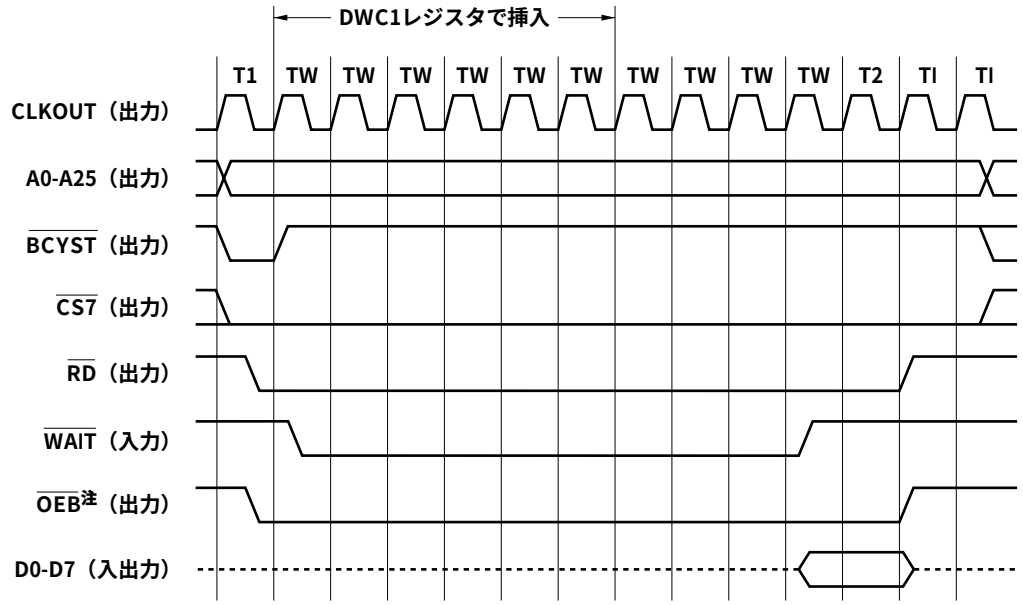
図3－8　ウエイト制御部の回路構成



注1.　74161**相当の回路を想定しています。**

2.　7476**相当の回路を想定しています。**

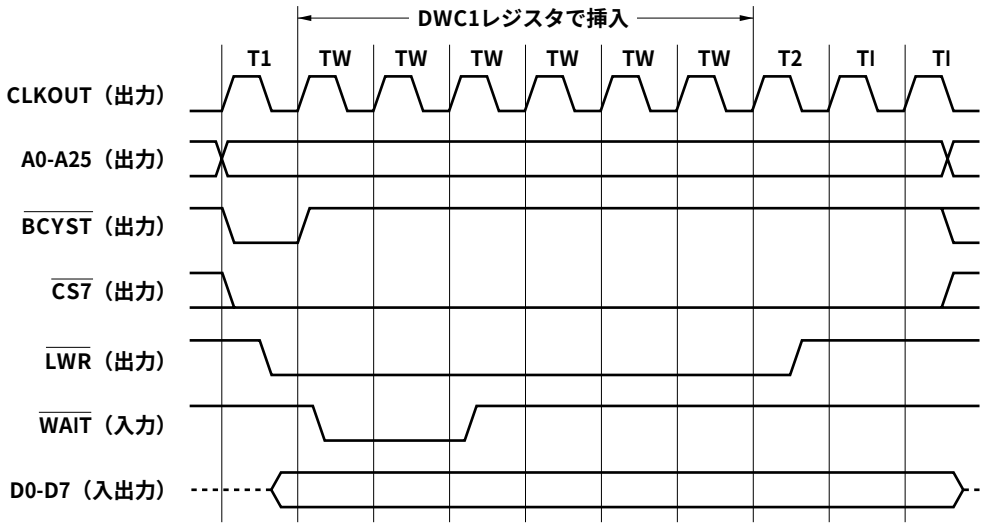
図3－9　μPD63310のリード動作



注　μPD63310の端子です。

備考　破線はハイ・インピーダンスを示します。

図3－10　μPD63310のライト動作



備考　破線はハイ・インピーダンスを示します。

3. 3. 2　$\overline{\text{WAIT}}$端子の制御例（2）（デュアル・ポートRAMとの接続）

デュアル・ポートRAM（IDT7007S20：32 K×8ビット）を1つ使用して32 Kバイトの外部メモリ空間を接続する例を示します。

【回路構成】

内部システム・クロック：50 MHz

接続デバイス：IDT7007S20×1つ

使用$\overline{\text{CS}}$信号：$\overline{\text{CS5}}$

外部メモリ空間のFA00000H-FA07FFFH（ブロック5）に配列することとします。

【接続の考え方と注意点】

- IDT7007S20の左側をV850E/MA1に接続する（V850E/MA1のA0-A14，D0-D7，$\overline{\text{CS5}}$端子，$\overline{\text{RD}}$端子，$\overline{\text{LWR}}$端子をIDT7007S20のA_{0L}-A_{14L}，I/O_{0L}-I/O_{7L}，$\overline{CE_L}$，$\overline{OE_L}$，$R/\overline{W}_L$にそれぞれ接続する）。

  備考　ライト側は他のプロセッサから制御することとします。

- IDT7007S20のM/$\overline{S}$端子はプルアップ，$\overline{SEM_L}$端子と$\overline{INT_L}$端子は使用しない。

  備考　M/$\overline{S}$　：IDT7007S20を複数使用する場合にマスタかスレーブかを設定します。ここでは1つだけ使用するためマスタを設定します。
  $\overline{SEM_L}$　：セマフォ機能を使用する場合のセレクト端子です。ここでは使用しないのでプルアップします。
  $\overline{INT_L}$　：ライト側プロセッサが特定エリアをライトしたときアクティブになる割り込み端子です。ここでは使用しないため，オープンにします。

- IDT7007S20の$\overline{BUSY_L}$端子がアクティブになった場合，データの確定は$\overline{BUSY_L}$端子の立ち上がりから20 ns（MAX.）必要であるため，ハードウエアで1クロック遅延させた信号と$\overline{BUSY_L}$端子の論理和をV850E/MA1の$\overline{\text{WAIT}}$端子に接続します。
- IDT7007S20の$\overline{BUSY_L}$端子はIDT7007S20の$\overline{CE_L}$信号のアクティブから，確定（$\overline{BUSY_L}$端子がアクティブになるか否かも含めて）まで20 ns（MAX.）なのでDWC1の設定[注]は1ウエイトとします。

  注　アドレス・セットアップ・ウエイト = 1でも満足しますが，その他のAC特性（V850E/MA1のt_{SRDID}など）との関係でデータ・ウエイト = 1を設定します。

- 8ビット・バス幅で接続するためBSCレジスタでの$\overline{\text{CS1}}$空間の設定は8ビットとします。

【レジスタの設定】

表3ー4　IDT7007S20との接続

レジスタ名称	設定値	機　　能
CSC1	xxxx0100xxxxx0xxB	ブロック5：$\overline{CS5}$
BCT1	x00xx0xx1000x0xxB	$\overline{CS5}$：メモリ・コントローラ許可，SRAM，外部I/O
BSC	0x0x000x0x0x0x0xB	$\overline{CS5}$：8ビット
DWC1	0xxx0xxx00010xxxB	$\overline{CS5}$：ウエイト1
ASC	xxxxxxxxxxxx00xxB	$\overline{CS5}$：アドレス・セットアップ・ウエイト0
BCC	xxxxxxxxxxxx01xxB	$\overline{CS5}$：アイドル・ステート1

図3ー11　IDT7007S20接続回路例

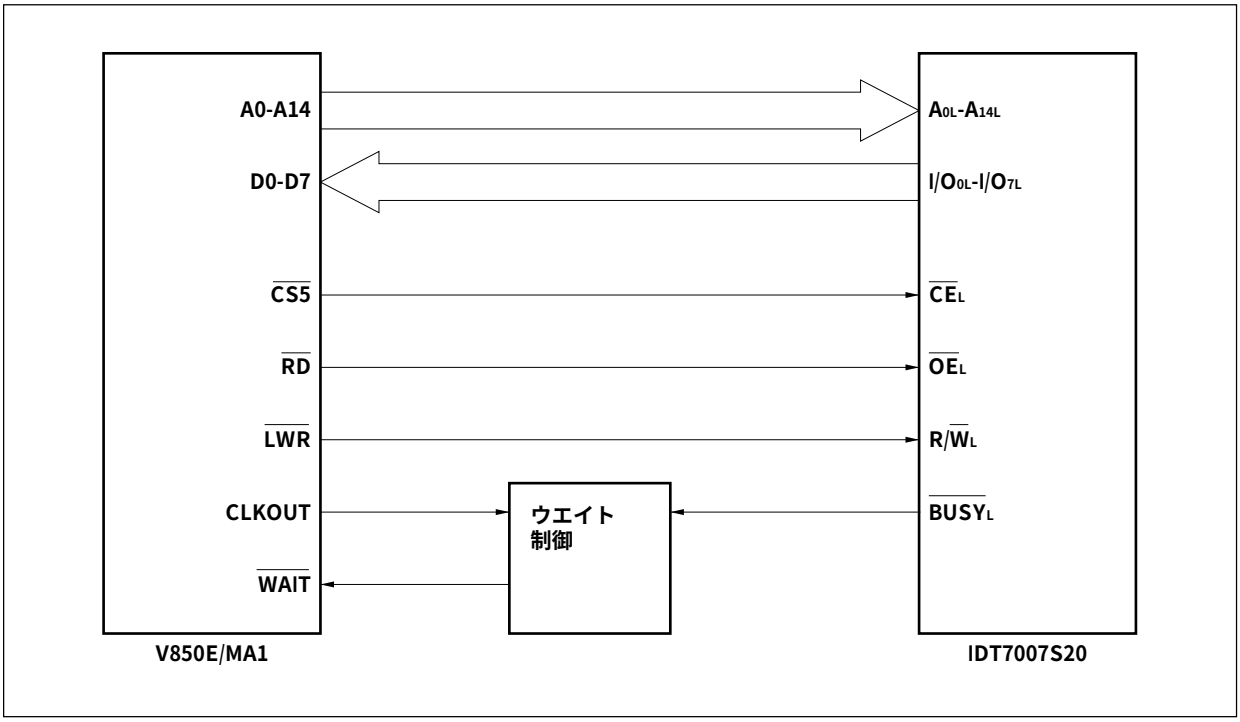


図3－12　ウエイト制御部回路構成

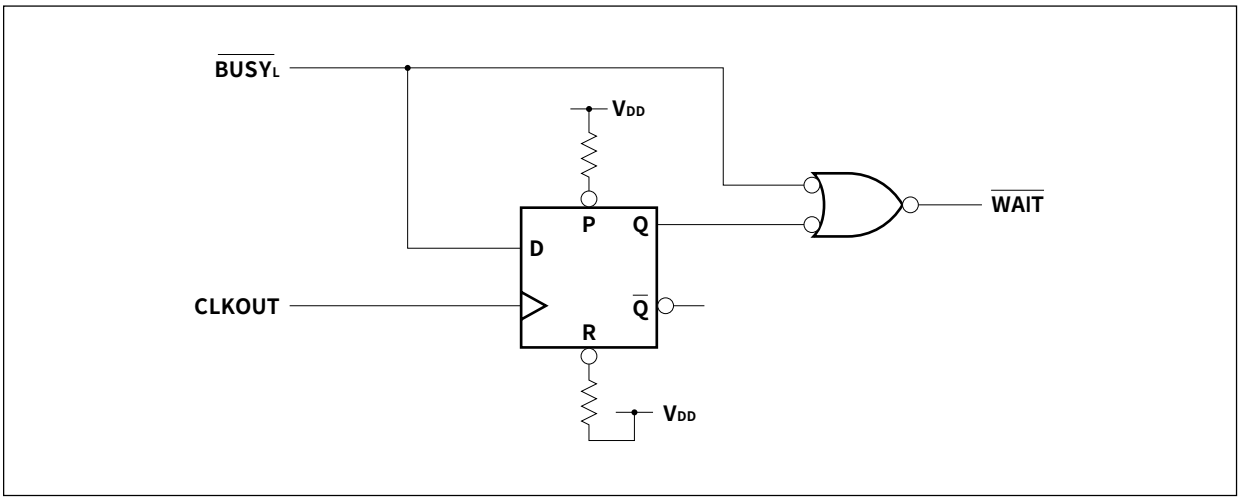
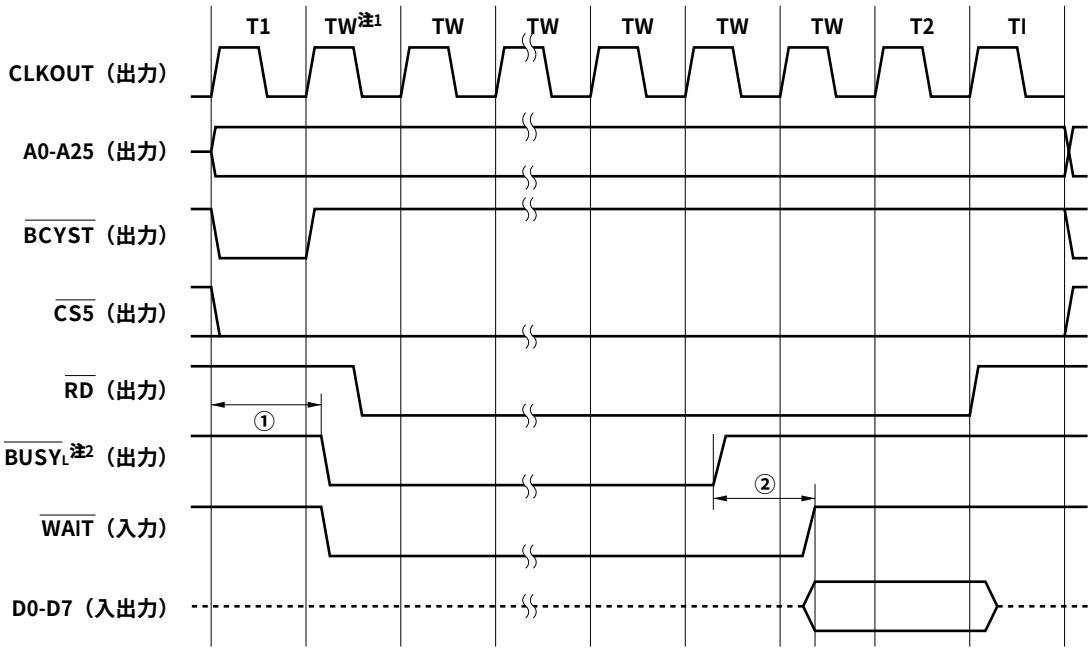


図3－13　IDT7007S20のリード動作（BUSYL端子アクティブ時）



① IDT7007S20のCELアクティブからBUSYL確定までの時間：20 ns（MAX.）

② IDT7007S20のBUSYL解除からデータ確定までの時間：20 ns（MAX.）

注1. DWCレジスタの設定により必ず1ウエイト挿入されます。

2. BUSYLはIDT7007S20の端子です。

備考　破線部はハイ・インピータンスを示します。

3. 4　複数のEDO DRAM（HM5164165FL-5）との接続

EDO DRAM（HM5164165FL：4 M×16ビット）を4つ（16ビット・バス幅，32 Mバイトの外部メモリ空間）接続する例を示します。

【回路構成】

- 内部システム・クロック：50 MHz
- 接続デバイス：HM5164165FL-5×4つ
- 使用$\overline{CS}$信号：$\overline{CS4}$（$\overline{RAS4}$）（エリア2）
  外部メモリ空間の8000000H-9FFFFFFHに配列することとします。
  A000000H-BFFFFFFHはイメージとなります。

【接続の考え方と注意点】

- V850E/MA1のA1-A13, D0-D15, $\overline{RAS4}$端子，$\overline{OE}$端子，$\overline{WE}$端子を4つのHM5164165FL-5のA0-A12, IO1-IO16, $\overline{RAS}$端子，$\overline{OE}$端子，$\overline{WE}$端子に直結します。
- V850E/MA1の$\overline{CAS}$信号（$\overline{LCAS}$, $\overline{UCAS}$）とV850E/MA1の上位アドレス（A23, A24）をデコードして4つのHM5164165FL-5に接続します。また接続する$\overline{CAS}$信号はリフレッシュ時にもアクティブになるように回路を構成します。

  備考　$\overline{RAS}$信号を上位アドレスでデコードする方法もありますが，V850E/MA1が他の$\overline{CS}$空間をアクセスしたときに上位アドレスが変化しますので，$\overline{RAS}$ホールド・モードが使用できなくなります。

- EDO DRAMのアクセス・タイミングはSCR4レジスタに設定します。
- SCR4レジスタの設定は付加回路による$\overline{CAS}$信号の遅延を考慮する以外は2. 5　EDO DRAMとの接続と同等になります。

  注意　この回路例では付加回路による$\overline{CAS}$信号の遅延7 ns（MAX.）を考慮した場合$\overline{CAS}$アクセス時間（t_{DH}）がw_{DA} = 1では満足できなくなりますので，w_{DA} = 2を設定します。それ以外は2. 5　EDO DRAMとの接続と同等です。

【レジスタの設定】

表3－5　複数のEDO DRAMとの接続

レジスタ名称	設定値	機　　能
BCT1	x00xx0xxx00x1010B	$\overline{CS4}$：メモリ・コントローラ許可，EDO DRAM
BSC	0x0x0x010x0x0x0xB	$\overline{CS4}$：16ビット
BCC	xxxxxx01xxxxxxxxB	$\overline{CS4}$：アイドル1
SCR4	A645H	オンページ・アクセス許可，W_{RP} = 2, W_{RH} = 1, W_{DA} = 2, W_{CP} = 1，16ビット幅，マルチプレクス幅 = 9ビット
RFS4	8017H	リフレッシュ許可，リフレッシュ・カウント・クロック = 32/f_{XX}，インターバル・ファクタ = 24
RWC	40H	RRW = 1, RCW = 0, SRW = 0

注意1. エリア2は$\overline{CS4}$（$\overline{RAS4}$）に固定されていますのでCSC0, CSC1レジスタの設定は意味を持ちません。

2. EDO DRAMアクセスではDWC0, DWC1レジスタ，ASCレジスタの設定は意味を持ちません。

備考　W_{RP}：ロウ・アドレス・プリチャージ・ウエイト数

W_{RH}：ロウ・アドレス・ホールド・ウエイト数

W_{DA}：データ・ウエイト数

W_{CP}：カラム・アドレス・プリチャージ・ウエイト数

RRW：リフレッシュ$\overline{RAS}$ウエイト数

RCW：リフレッシュ・サイクル・ウエイト数

SRW：セルフ・リフレッシュ解除ウエイト数

図3－14　HM5164165FL-5（×4つ）接続回路例

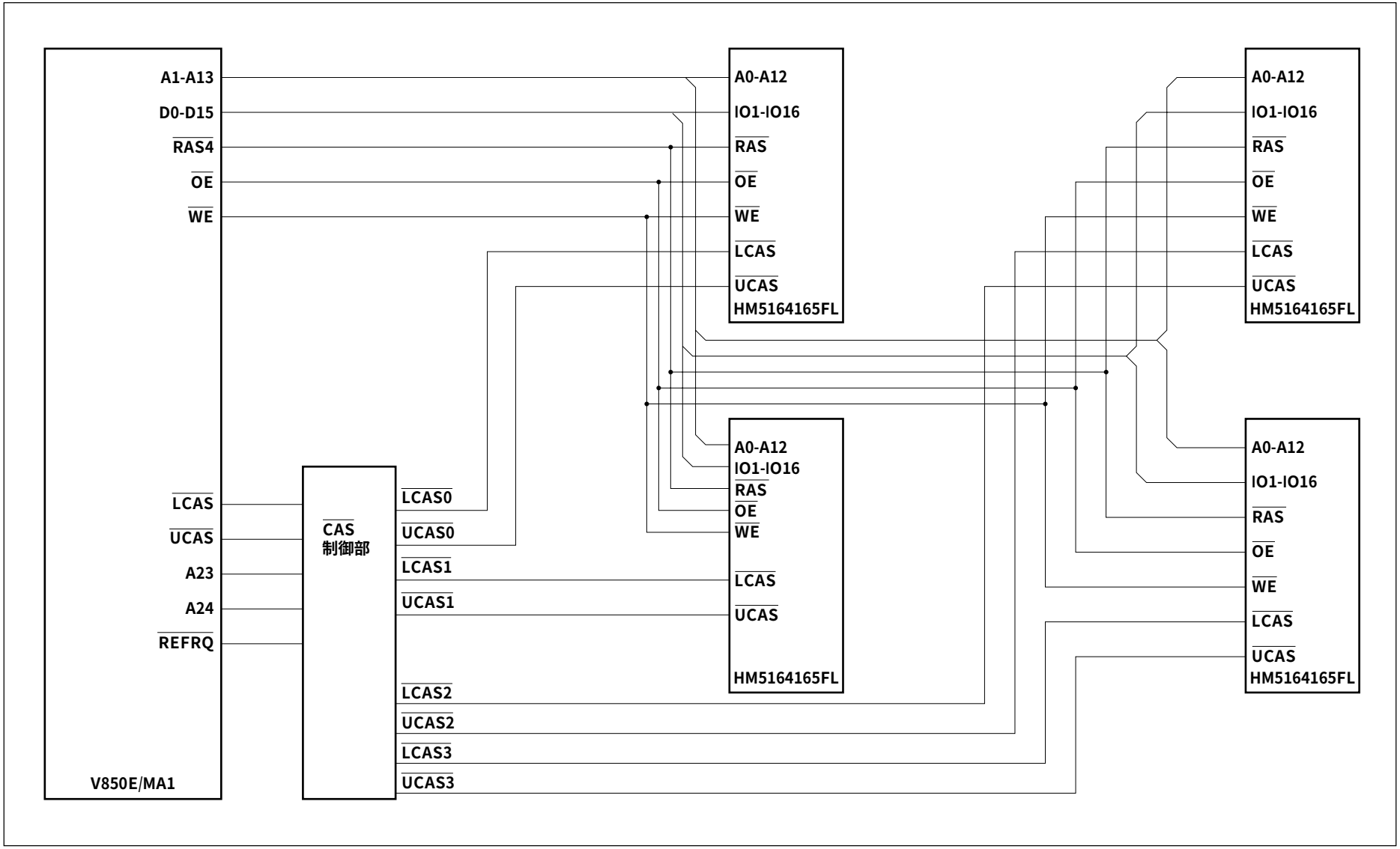
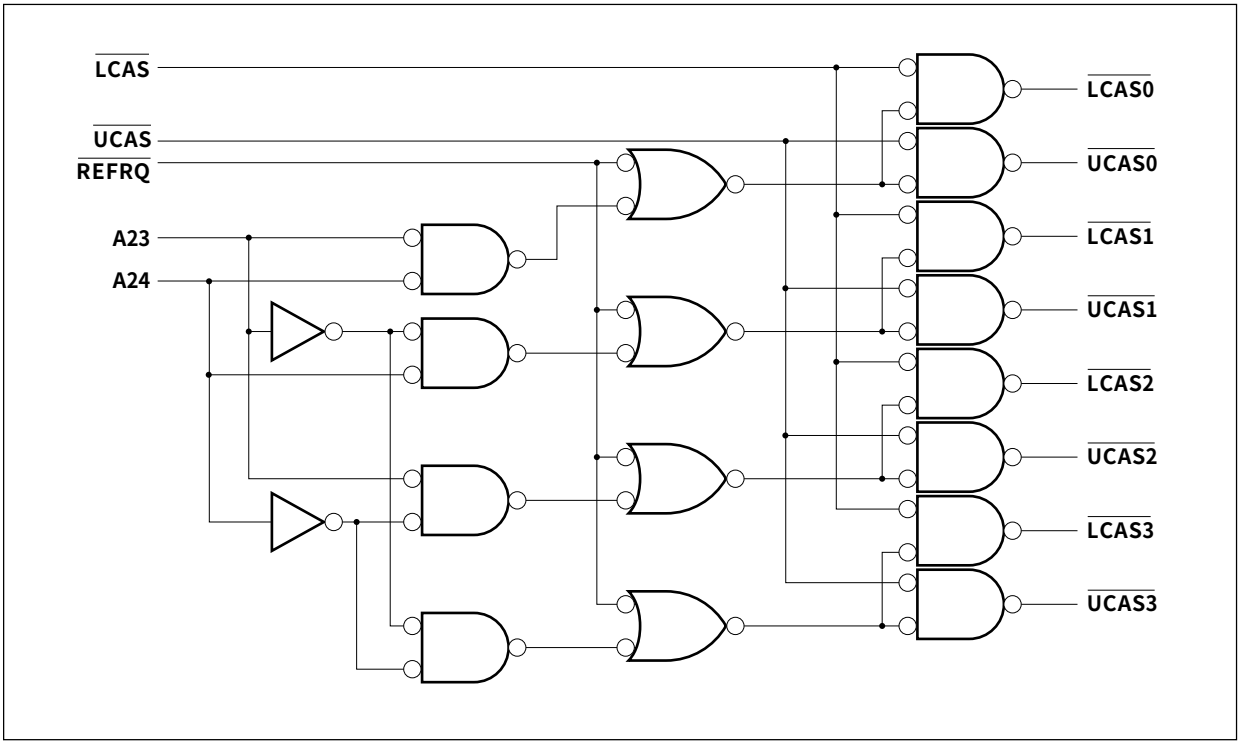


図3－15　$\overline{\text{CAS}}$制御部の回路構成



3. 5　$\overline{\text{HLDRQ}}$信号を使用した外部リフレッシュ・ユニットとの接続

外部接続したEDO DRAMのリフレッシュを外部ユニットで制御する例を示します。

【回路構成】

- 内部システム・クロック：50 MHz
- 接続デバイス：HM5164165FL-5×1つ
  リフレッシュ制御を除いて2. 5　EDO DRAMとの接続と同等の回路構成（使用$\overline{\text{CS}}$信号は$\overline{\text{RAS4}}$）とします。
- 外部リフレッシュ・ユニット：
  一定時間ごとに$\overline{\text{HLDRQ}}$端子をアクティブにし，V850E/MA1がホールド状態に遷移したらCBRリフレッシュを2回行い，$\overline{\text{HLDRQ}}$端子をインアクティブにします。

【接続の考え方と注意点】

- 30 μs周期のクロックでトリガを発生させ，V850E/MA1の$\overline{\text{HLDRQ}}$端子をアクティブにし，$\overline{\text{HLDAK}}$端子がアクティブ・レベルになったらCBRリフレッシュ制御を2回行い，リフレッシュ動作完了後，$\overline{\text{HLDRQ}}$端子をインアクティブにする回路を作成します。
- EDO DRAMの$\overline{\text{RAS}}$端子，$\overline{\text{LCAS}}$端子，$\overline{\text{UCAS}}$端子はV850E/MA1の$\overline{\text{RAS4}}$端子，$\overline{\text{LCAS}}$端子，$\overline{\text{UCAS}}$端子と外部リフレッシュ制御回路をデコードして接続します。
- SCR4レジスタの設定はリフレッシュ禁止にします。

注意　DRC4レジスタ，RWCレジスタ以外のレジスタ設定とリフレッシュ以外のタイミングは3. 4　複数のEDO DRAM（HM5164165FL-5）との接続と同等となります。

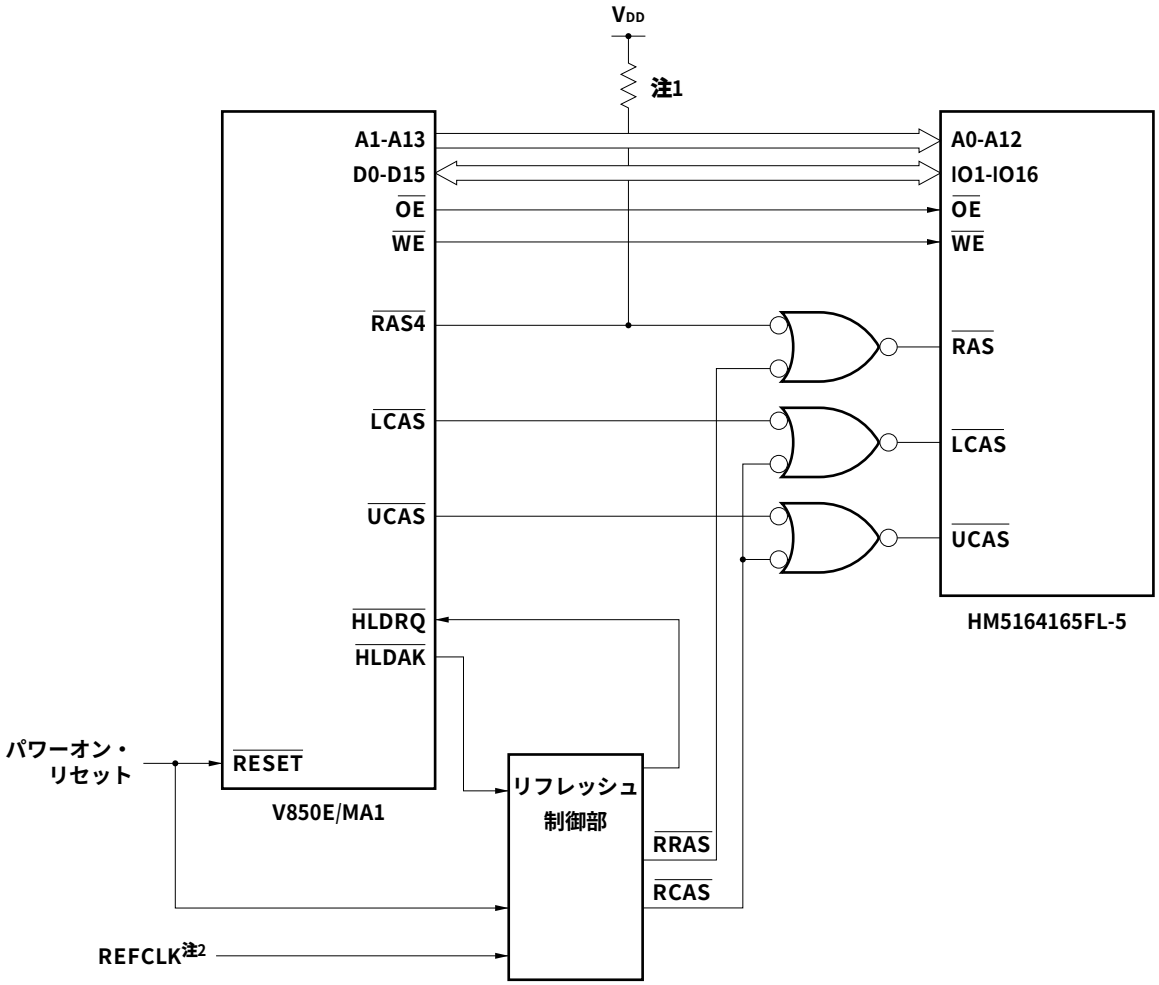
【レジスタの設定】

表3－6　$\overline{\text{HLDRQ}}$信号を使用した外部リフレッシュ・ユニットとの接続

レジスタ名称	設定値	機　　能
BCT1	x00xx0xxx00x1010B	$\overline{\text{CS4}}$：メモリ・コントローラ許可，EDO DRAM
BSC	0x0x0x010x0x0x0xB	$\overline{\text{CS4}}$：16ビット
BCC	xxxxxx01xxxxxxxxB	$\overline{\text{CS4}}$：アイドル・ステート1
SCR4	A645H	オンページ・アクセス許可，W_{RP} = 2, W_{RH} = 1, W_{DA} = 2, W_{CP} = 1，16ビット幅，マルチプレクス幅 = 9ビット
RFS4	0xxxxxxxxxxxxxxxB	リフレッシュ禁止
RWC	xxxxx001B	セルフ・リフレッシュ解除のウエイト数：1

備考　W_{RP}：ロウ・アドレス・プリチャージ・ウエイト数
W_{RH}：ロウ・アドレス・ホールド・ウエイト数
W_{DA}：データ・ウエイト数
W_{CP}：カラム・アドレス・プリチャージ・ウエイト数

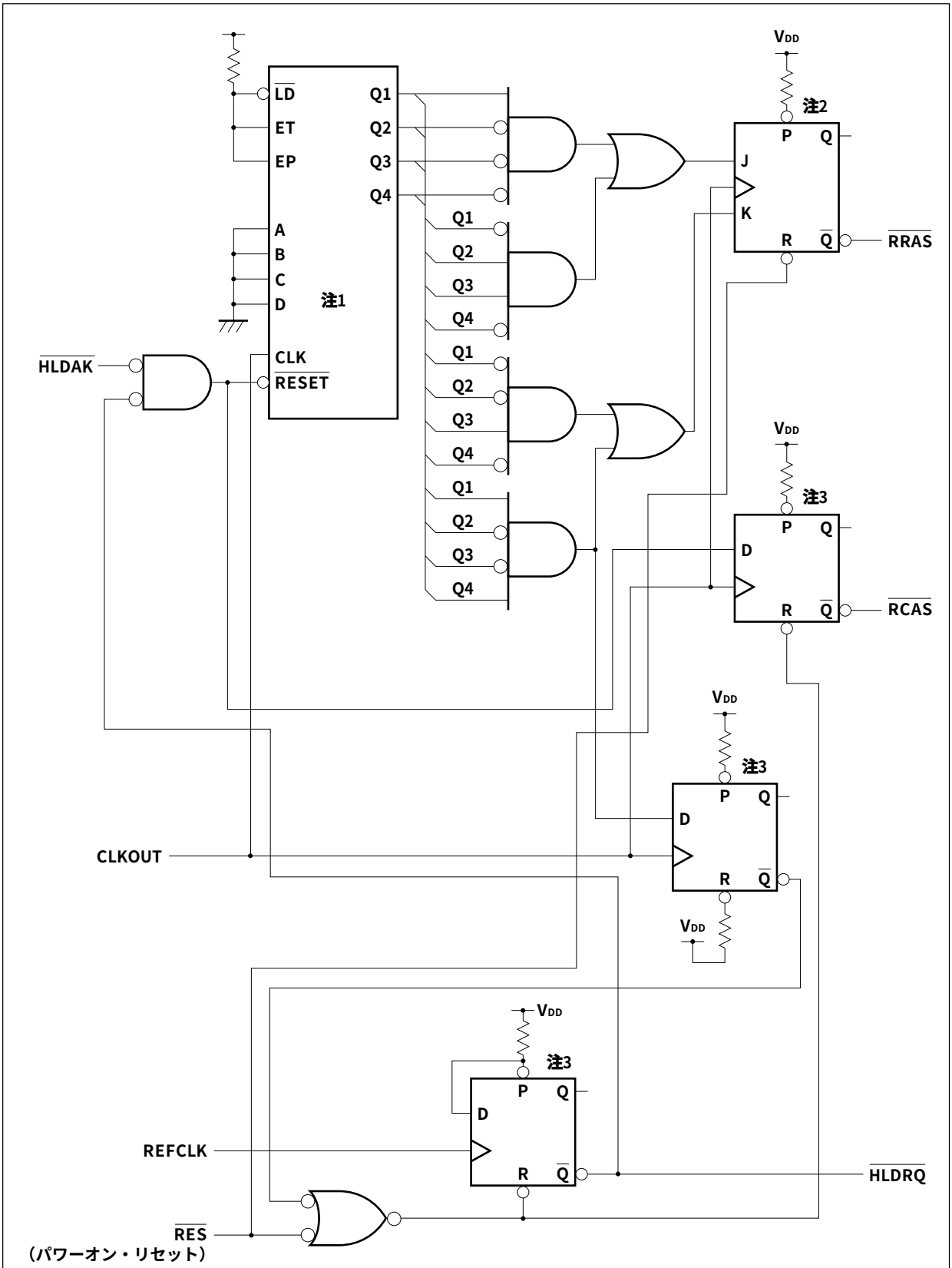
図3－16　$\overline{\text{HLDRQ}}$信号を使用した外部リフレッシュ・ユニット



注1. $\overline{\text{RAS}}$**ホールド・モード使用時に**$\overline{\text{RASn}}$**信号がロウ・レベルでホールド状態に遷移すると，**EDO DRAM**の**$\overline{\text{RAS}}$**信号にスパイク・ノイズが発生する危険があります。この例では，プルアップ抵抗の接続でスパイク・ノイズの発生を防いでいます。**

2. 30 μs**周期のクロックが入力されることとします。実際の回路では，**V850E/MA1**内蔵の**RPU**の出力信号（**TO0n**）などで作成してください。**

図3－17　リフレッシュ制御部の回路構成

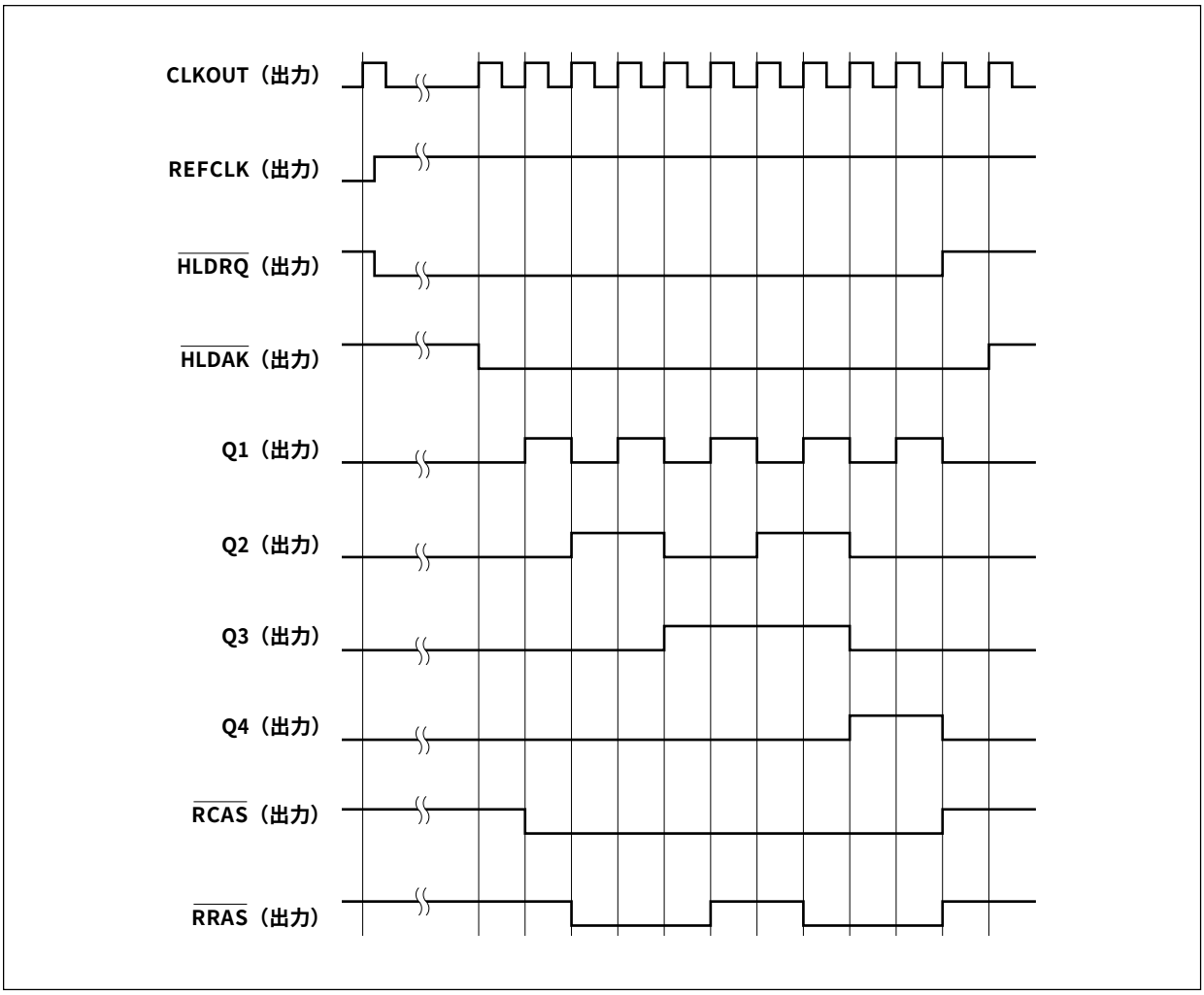


注1.　74161**相当の回路を想定しています。**

2.　7476**相当の回路を想定しています。**

3.　7474**相当の回路を想定しています。**

図3－18　リフレッシュ制御タイミング



第4章　アプリケーション例

V850E/MA1を搭載したCPUボード（TU-V850E/MA1）の機能と回路およびプログラム例を紹介します。

4. 1　TU-V850E/MA1の機能

4. 1. 1　概　　要

TU-V850E/MA1は，32ビット・シングルチップ・マイクロコンピュータV850シリーズの製品であるV850E/MA1の学習を目的として開発されたトレーニング・ユニットです。

TU-V850E/MA1の特徴は次のとおりです。

（1）V850E/MA1用インサーキット・エミュレータを搭載

（2）バスの動作をロジック・アナライザにて観測可能

バス・インタフェース関連のすべての信号をロジック・アナライザ用コネクタに配列

（3）動作速度は50 MHz（MAX.）

OSCはOSC1とOSC2の2種類を用意し，ジャンパ・スイッチで供給クロックを選択

OSC1は標準供給クロック，OSC2はソケット上に実装されるためクロック供給の変更が可能

（4）各種メモリを搭載

- EPROM
- SRAM
- EDO DRAM
- シンクロナスDRAM
- フラッシュ・メモリ
- ページROM
- シリアルEEPROM™（V850E/MA1内蔵CSI2に接続）

（5）シリアル・インタフェース

RS-232-C×1チャネル（V850E/MA1内蔵UART1に接続）

（6）内部フラッシュ・メモリ用フラッシュ・ライタ・インタフェースを搭載

V850E/MA1内蔵CSI0に接続

(7) アナログ入力

正弦波発振器の出力を接続

0-3 V**アナログ出力を接続**

光センサの出力を接続

(8) 汎用スイッチ入力8点，汎用LED**出力8点**

V850E/MA1**ポート機能に接続**

(9) 7**セグメント**LED**×**5**桁**

ソフトウエアによる制御とハードウエアによる制御を，スイッチにより選択

ハードウエア制御時は正弦波発振器またはRPU（TO02**出力）の周波数を表示**

(10) 回転エンコーダ出力付きDC**モータ搭載**

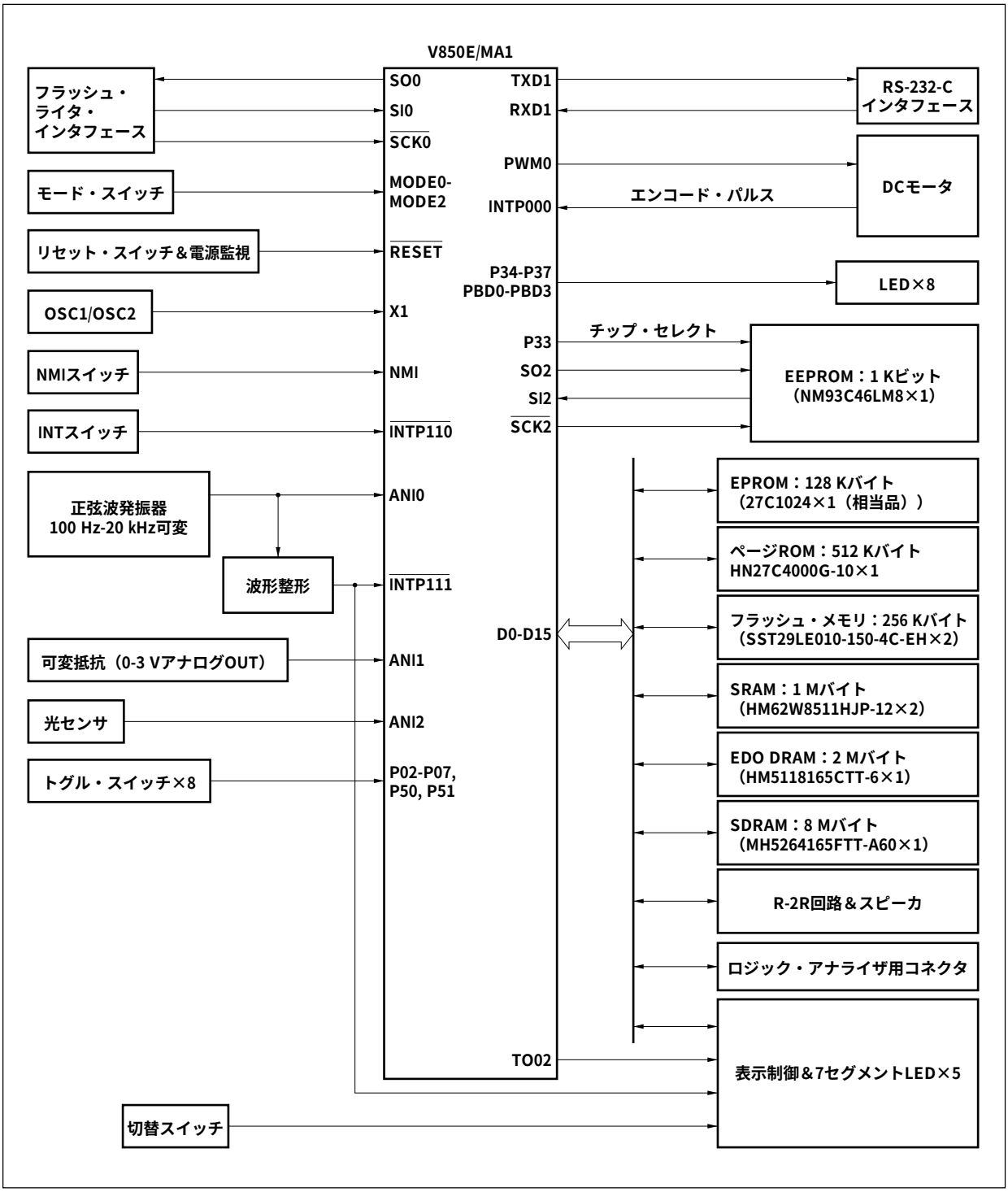
モータの速度制御はV850E/MA1**内蔵**PWM0**を使用**

(11) スピーカ搭載

バス・インタフェースに接続されたR-2R**回路で制御**

(12) 使用電源はAC100/200 V

図4－1　ボード構成図


V850E/MA1
フラッシュ・ライタ・インタフェース
SO0
SI0
SCK0
TXD1
RXD1
RS-232-C インタフェース
モード・スイッチ
MODE0-MODE2
PWM0
DCモータ
エンコード・パルス
INTP000
リセット・スイッチ＆電源監視
RESET
P34-P37
PBD0-PBD3
LED×8
OSC1/OSC2
X1
チップ・セレクト
P33
SO2
SI2
SCK2
EEPROM：1 Kビット（NM93C46LM8×1）
NMIスイッチ
NMI
INTスイッチ
INTP110
正弦波発振器 100 Hz-20 kHz可変
ANI0
EPROM：128 Kバイト（27C1024×1（相当品））
ページROM：512 Kバイト HN27C4000G-10×1
波形整形
INTP111
D0-D15
フラッシュ・メモリ：256 Kバイト（SST29LE010-150-4C-EH×2）
可変抵抗（0-3 Vアナログ OUT）
ANI1
SRAM：1 Mバイト（HM62W8511HJP-12×2）
光センサ
ANI2
EDO DRAM：2 Mバイト（HM5118165CTT-6×1）
トグル・スイッチ×8
P02-P07, P50, P51
SDRAM：8 Mバイト（MH5264165FTT-A60×1）
R-2R回路＆スピーカ
ロジック・アナライザ用コネクタ
TO02
表示制御＆7セグメントLED×5
切替スイッチ

4.1.2　メモリ・マップ

モード設定スイッチをシングルチップ・モード0に指定した場合のメモリ・マップを示します。

図4－2　メモリ・マップ

4.1.3　周辺機能接続ユニット

(1) RPUの接続

TO02出力が外部表示制御部に接続されます。スイッチの設定によりTO02出力の周波数を7セグメントLEDに表示します。

(2) UARTの接続

UART1を調歩同期式のRS-232-Cインタフェースとして使用します。信号はTXD, RXDの2線式です。モデム制御信号のRS信号，ER信号はアクティブ・レベルに固定され，CS信号，DR信号は接続されません。ジャンパ・スイッチの設定でTXD信号をRXD信号に折り返すことが可能です。

通信方式　：非同期
通信速度　：最大19200 bps（内部ボーレート・ジェネレータで設定）
出力信号　：TXD
入力信号　：RXD
使用コネクタ：DSUB9ピン

(3) CSIの接続

CSI0, CSI2がそれぞれ内部フラッシュ・ライタ・インタフェース・コネクタとEEPROMに接続されます。専用フラッシュ・ライタを使用するときはMODE設定スイッチをフラッシュ・メモリ・プログラミング・モードに設定します。

EEPROMのDI端子，DO端子，CLK端子がそれぞれSO2, SI2, $\overline{\text{SCK2}}$に接続されます。また，EEPROMのCS端子（デバイス・セレクト端子）はP33に接続されます。

(4) PWMの接続

PWM0出力をDCモータの速度制御に使用します。PWM0端子がハイ・レベルで100 %の回転速度，ロウ・レベルで停止します。また，DCモータの回転エンコード・パルスはINTP000端子に接続されます。

(5) A/Dコンバータの接続

ANI0に正弦波発振器（100 Hz-20 kHz）の出力を接続，ANI1に0-3 Vの可変抵抗を接続，ANI2に光センサを接続します。

(6) 割り込み信号の入力

NMIスイッチ，DCモータ・エンコード・パルス，INTスイッチ，正弦波発振器の出力を方形波に変換したパルス信号がそれぞれNMI, INTP000, $\overline{\text{INTP110}}$, $\overline{\text{INTP111}}$に接続されます。

(7) 入出力ポートの接続

P02-P07, P50, P51に8個の汎用トグル・スイッチが接続され，P34-P37, PBD0-PBD3に8個の汎用LEDが接続されます。またP33はEEPROMのCS端子に接続されます。

4.1.4　バス・インタフェース接続ユニット

EPROM，フラッシュ・メモリ，SRAM, EDO DRAM，シンクロナスDRAM，ページROM，5個の7セグメントLEDおよびR-2Rアナログ出力回路を経由してのスピーカがV850E/MA1のバス・インタフェースに接続されます。

バス・インタフェースに接続されるすべてのユニットのウエイト制御はV850E/MA1により制御されます（$\overline{\text{WAIT}}$端子はプルアップされます）。

バス・インタフェースに接続されるユニットは，7セグメントLED, R-2Rアナログ出力が8ビット・バス幅，その他が16ビット・バス幅で構成されます。

（1）EPROM

外部メモリ空間のブロック0に1 Mビット EPROM（64 K×16ビット：27C1024）相当品が実装可能な40ピン・ソケットが1個実装されています。$\overline{\text{CS0}}$信号のアクティブでセレクトされます。

（2）SRAM

外部メモリ空間のブロック1に4 Mビット SRAM（512 K×8ビット：HM62W8511HJP-12）が2つ実装されています。$\overline{\text{CS1}}$信号のアクティブでセレクトされます。

（3）ページROM

外部メモリ空間のブロック2にバースト・アクセス可能な4 Mビット EPROM（256 K×16ビット：HN27C4000G-10）が40ピンICソケットに実装されています。$\overline{\text{CS2}}$信号のアクティブで選択されます。

（4）EDO DRAM

外部メモリ空間のエリア1に16 Mビット EDO DRAM（1 M×16ビット：HM5118165CTT-6）が1つ実装されています。$\overline{\text{RAS}}$信号は$\overline{\text{RAS3}}$端子を使用します。リフレッシュ・サイクルはV850E/MA1がサポートします。

（5）シンクロナスDRAM

外部メモリ空間のエリア2（$\overline{\text{CS2}}$空間）に64 Mビット・シンクロナスDRAM（1 M×16ビット×4バンク：HM5264165FTT-A60）が1つ実装されます。リフレッシュ・サイクルはV850E/MA1がサポートします。

（6）フラッシュ・メモリ

外部メモリ空間のブロック0に1 Mビット・フラッシュ・メモリ（128 K×8ビット：SST29LE010-150-4C-EH）が2個実装されています。$\overline{\text{CS0}}$信号のアクティブでセレクトされます。

（7）EEPROM

シリアル・タイプの1 Kビット EEPROM（NM93C46LM8）が1つ実装されます。V850E/MA1のCSI2で制御されます。また，NM93C46LM8のCS端子はV850E/MA1のP33端子に接続されます。

4.1.5　設定スイッチ

(1) 動作モード指定スイッチ（DSW1）

V850E/MA1の動作モードを指定する4ビットのDIPスイッチです。V850E/MA1のMODE0，MODE1端子に接続されています。スイッチはオンで「0」，オフで「1」に設定されます。

MODE2端子はフラッシュ・ライタにより設定されます。通常時MODE2端子はGNDレベルに固定されます。

スイッチ名称	意　味
DSW1-1	MODE0端子の設定
DSW1-2	MODE1端子の設定
DSW1-3, DSW1-4	未使用

(2) フラッシュ・ライタ用スライド・スイッチ（SSW1）

専用フラッシュ・ライタでV850E/MA1内蔵フラッシュ・メモリの内容をリード／ライトするときに使用するスライド・スイッチです。ライト側に設定した場合，V850E/MA1の$\overline{\text{RESET}}$端子は専用フラッシュ・ライタにより制御されます。

スイッチ名称	意　味
NORMAL	通常動作
WRITE	内蔵フラッシュ･メモリ：リード／ライト

(3) 供給クロック設定スイッチ（JP1）

V850E/MA1のX1端子に供給するクロックを設定するジャンパ・スイッチです。

スイッチ名称	意　味
1-2短絡	OSC1を選択
2-3短絡	OSC2を選択

(4) RS-232-Cインタフェース折り返しスイッチ（JP2）

RS-232-Cインタフェースの送信データと受信データを接続するスイッチです。

スイッチ名称	意　味
1-2短絡	通常動作
2-3短絡	送信データを受信データに折り返す

(5) 7セグメントLED切り替えスイッチ（SW6）

7セグメントLEDの表示をソフトウエア制御にするか，ハードウエア制御にするかの選択を行うオルタネート・トグル・スイッチです。

スイッチ名称	意　味
SOFT	ソフトウエア制御
HARD	ハードウエア制御

(6) 周波数表示選択スイッチ（SW7）

7セグメントLEDの周波数表示を正弦波発振器のモニタにするか，TO02出力のモニタにするかを設定するオルタネート・トグル・スイッチです。このスイッチは7セグメントLED切替スイッチがハードウエア制御になっているときのみ有効です。

スイッチ名称	意　味
TO02	TO02出力選択
$\overline{\text{INTP111}}$	正弦波発振器出力選択

(7) DCモータ起動スイッチ

DCモータの制御を有効にするオルタネート・トグル・スイッチです。DCモータ起動スイッチがオフされているときは，モータが停止します。

スイッチ名称	意　味
オン	DCモータ起動可
オフ	DCモータ強制停止

(8) 汎用トグル・スイッチ（TSW1-TSW8）

V850E/MA1の入力ポートに接続される8個のオルタネート・トグル・スイッチです。P02-P07がTSW3-TSW8に，P50, P51がTSW1, TSW2にそれぞれ対応します。各スイッチはオンで“0”，オフで“1”になります。

(9) INTスイッチ（INT）

V850E/MA1にマスカブル割り込みを発生させる押しボタン・スイッチです。$\overline{\text{INTP110}}$端子に接続されています。

(10) NMIスイッチ（NMI）

V850E/MA1にノンマスカブル割り込みを発生させる押しボタン・スイッチです。

(11) リセット・スイッチ（RESET）

このユニットをリセットする押しボタン・スイッチです。パワーオン・リセット信号とORされます。

リセット・スイッチはフラッシュ・ライタ用スライド・スイッチがライト側に設定されている場合は無効です。

4.1.6　周辺I/Oの仕様

(1) ポート機能

8本の入力ポートと9本の出力ポートを使用しています。次に一覧を示します。

ポート名称	入出力	機　　能
P50	入力	汎用トグル・スイッチ1（オン = 0，オフ = 1）
P51	入力	汎用トグル・スイッチ2（オン = 0，オフ = 1）
P02	入力	汎用トグル・スイッチ3（オン = 0，オフ = 1）
P03	入力	汎用トグル・スイッチ4（オン = 0，オフ = 1）
P04	入力	汎用トグル・スイッチ5（オン = 0，オフ = 1）
P05	入力	汎用トグル・スイッチ6（オン = 0，オフ = 1）
P06	入力	汎用トグル・スイッチ7（オン = 0，オフ = 1）
P07	入力	汎用トグル・スイッチ8（オン = 0，オフ = 1）
PBD0	出力	汎用LED1（点灯 = 1，消灯 = 0）
PBD1	出力	汎用LED2（点灯 = 1，消灯 = 0）
PBD2	出力	汎用LED3（点灯 = 1，消灯 = 0）
PBD3	出力	汎用LED4（点灯 = 1，消灯 = 0）
P34	出力	汎用LED5（点灯 = 1，消灯 = 0）
P35	出力	汎用LED6（点灯 = 1，消灯 = 0）
P36	出力	汎用LED7（点灯 = 1，消灯 = 0）
P37	出力	汎用LED8（点灯 = 1，消灯 = 0）
P33	出力	シリアルEEPROMのチップ・セレクト（セレクト = 1）

(2) 外部割り込み端子

3本のマスカブル割り込みと1本のNMI割り込みが接続されます。次に一覧を示します。

割り込み名称	エッジの指定	接続信号
NMI	立ち下がり	NMIスイッチ
INTP000	立ち上がり	DCモータ・エンコード・パルス（216パルス／1回転）
$\overline{\text{INTP110}}$	立ち下がり	INTスイッチ
$\overline{\text{INTP111}}$	立ち上がり	正弦波発振器（1パルス／1サイクル）

(3) シリアル・インタフェース

CSI0, CSI2, UART1を使用しています。次に一覧を示します。

ユニット名称	機　　能
CSI0	V850E/MA1内部フラッシュ・メモリ・インタフェースとして使用します。 専用フラッシュ・ライタに適合する構成になっています。
CSI2	シリアルEEPROMのインタフェースとして使用します。 EEPROMのアクセスは$\overline{\text{CS}}$端子（P33）が‘1’のとき有効です。
UART1	RS-232-Cインタフェースとして使用します。モデム制御信号はサポートされません（RS, ERは常時アクティブ，CS, DR, CDはオープン）。プロトコルにあわせてプログラミングしてください。

(4) PWM**ユニット**

PWM0**を**DC**モータの制御に使用します。**PWM0**の状態がハイ・レベルでモータが動作，ロウ・レベルで停止します。**100 **%の出力（**PWM1**の出力がハイ・レベルを保持）で**250**回転／分となります。回転速度は**INTP000**でモニタ可能です。**

(5) A/D**コンバータ**

0-3 V**のアナログ入力が**ANI0-ANI2**の**3**チャネルに入力されます。**

次に一覧を示します。

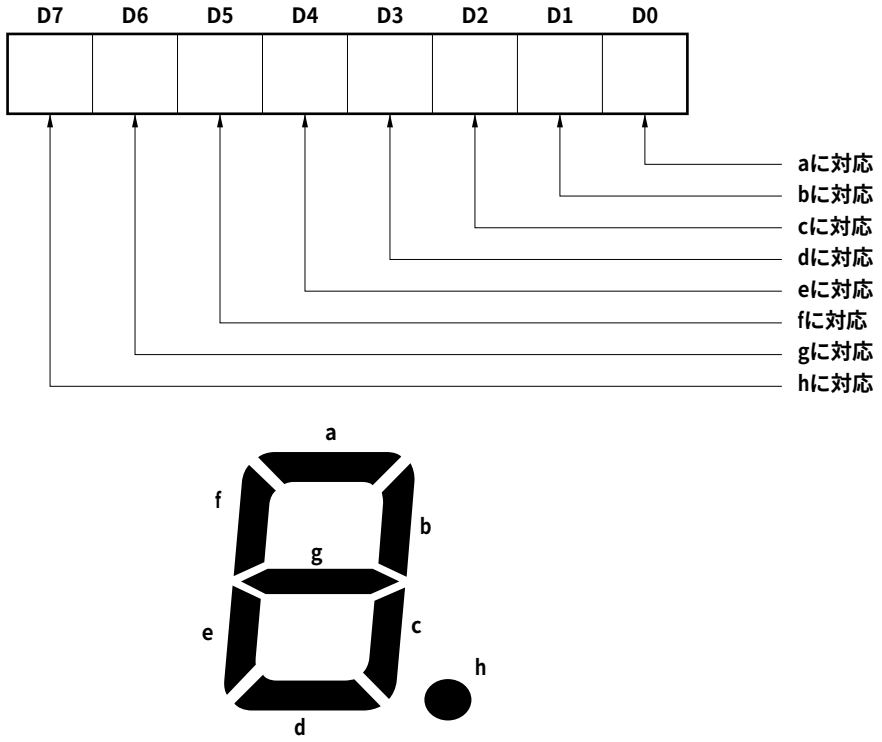
端子名称	機　　能
ANI0	**正弦波入力（**100 Hz-20 kHz**）**
ANI1	**可変抵抗による**0-3 V**入力**
ANI2	**光センサによる**0-3 V**（暗で**0 V**，明で**3 V**）入力**

(6) 7**セグメント**LED**表示レジスタ**

7**セグメント**LED**を点灯させるためのリード／ライト可能なレジスタです。**7**セグメント表示切り替えスイッチがソフトウエア制御側になっているとき**7**セグメント**LED**表示レジスタの設定が**LED**に反映します。**

アクセスは7**セグメント表示切替スイッチがソフトウエア制御になっていなくても有効です。**

図4－3　7**セグメント**LED**表示レジスタ**



備考　各LED**は‘**1**’で点灯，‘**0**’で消灯します。**

(7) R-2Rアナログ出力レジスタ

8ビットのリード／ライト可能なアナログ出力ポートです。FFHで最大電圧，00Hで最小電圧が設定されます。R-2Rアナログ出力レジスタの設定はスピーカに反映されます。

表4－1　I/Oレジスター覧

リトル・エンディアンでのアドレス	ビック・エンディアンでのアドレス	機　　能
XXXX XXX0H	XXXX XXX3H	7セグメントLED1桁目のレジスタ
XXXX XXX2H	XXXX XXX1H	7セグメントLED2桁目のレジスタ
XXXX XXX4H	XXXX XXX7H	7セグメントLED3桁目のレジスタ
XXXX XXX6H	XXXX XXX5H	7セグメントLED4桁目のレジスタ
XXXX XXX8H	XXXX XXXBH	7セグメントLED5桁目のレジスタ
XXXX XXXAH	XXXX XXX9H	R-2Rアナログ出力レジスタ
XXXX XXXBH-XXXX XXXFH	XXXX XXXCH-XXXX XXXFH	未使用

備考　アドレスは$\overline{\text{CS7}}$空間内のアドレスです。

4.1.7　接続コネクタ

(1) RS-232-Cインタフェース・コネクタ（CN1）

使用コネクタ：DSUB9ピン（DE-9S-T-N：JAE）

ピン番号	信号名称	信号の意味
1	NC	未接続
2	RD	受信データ
3	SD	送信データ
4	ER	端末レディ
5	GND	信号グランド
6	NC	未接続
7	RS	送信要求
8	NC	未接続
9	NC	未接続

備考　RS, ERはアクティブ・レベルに固定されます。

(2) フラッシュ・メモリ・プログラミング・インタフェース・コネクタ（CN2）

使用コネクタ：10ピン・ヘッダ（PS-10PE-D4T1-B1：JAE）

ピン番号	信号名称	信号の意味
1	GND	グランド
2	SO0	シリアル・データ出力
3	SI0	シリアル・データ入力
4	$\overline{\text{SCK0}}$	シリアル・クロック入力
5	NC	未接続
6	$\overline{\text{RESET}}$	リセット信号入力
7	V_{DD}	ボード電源
8	V_{PP}	V_{PP}電圧入力
9	NC	未接続
10	NC	未接続

(3) ロジック・アナライザ接続コネクタ（CN3-CN6）

CN3

ピン番号	信号名	ピン番号	信号名
1	NC	2	NC
3	NC	4	D15
5	D14	6	D13
7	D12	8	D11
9	D10	10	D9
11	D8	12	D7
13	D6	14	D5
15	D4	16	D3
17	D2	18	D1
19	D0	20	GND

CN4

ピン番号	信号名	ピン番号	信号名
1	NC	2	NC
3	NC	4	A15
5	A14	6	A13
7	A12	8	A11
9	A10	10	A9
11	A8	12	A7
13	A6	14	A5
15	A4	16	A3
17	A2	18	A1
19	A0	20	GND

CN5

ピン番号	信号名	ピン番号	信号名
1	NC	2	NC
3	NC	4	$\overline{CS7}$
5	$\overline{CS6}$	6	$\overline{CS5}$
7	$\overline{CS4}$	8	$\overline{CS3}$
9	$\overline{CS2}$	10	$\overline{CS1}$
11	$\overline{CS0}$	12	A23
13	A22	14	A21
15	A20	16	A19
17	A18	18	A17
19	A16	20	GND

CN6

ピン番号	信号名	ピン番号	信号名
1	NC	2	NC
3	NC	4	SELFREF
5	$\overline{REFRQ}$	6	CLKOUT
7	$\overline{WAIT}$	8	$\overline{BCYST}$
9	$\overline{OE}$	10	$\overline{WE}$
11	$\overline{RD}$	12	$\overline{UCAS}$/$\overline{UWR}$/UDQM
13	$\overline{LCAS}$/$\overline{LWR}$/LDQM	14	$\overline{UBE}$/$\overline{SDRAS}$
15	$\overline{LBE}$/$\overline{SDCAS}$	16	SDCLK
17	SDCKE	18	A25
19	A24	20	GND

4. 2　アプリケーション・プログラム例

4. 2. 1　開発ツール

掲載してあるプログラム例は，すべてNEC製の開発環境を使用して作成しております。

(1) Cコンパイラ・パッケージ（CA850）

Cコンパイラ・パッケージに関するユーザーズ・マニュアルは次の4つです。

- CA850　Cコンパイラ・パッケージ　操作編（U15024J）
- CA850　Cコンパイラ・パッケージ　C言語編（U15025J）
- CA850　Cコンパイラ・パッケージ　アセンブリ言語編（U15027J）
- CA850　Cコンパイラ・パッケージ　プロジェクト・マネージャ編（Windowsベース）（U15026J）

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

(2) V850シリーズ用Cソース・ディバッガ（ID850）

V850シリーズ用Cソース・ディバッガに関するユーザーズ・マニュアルです。

- ID850　統合ディバッガ　操作編（Windowsベース）（U15181J）

(3) IE-V850E-MC-A

V850E1用インサーキット・エミュレータ（3.3 V用）に関するユーザーズ・マニュアルです。

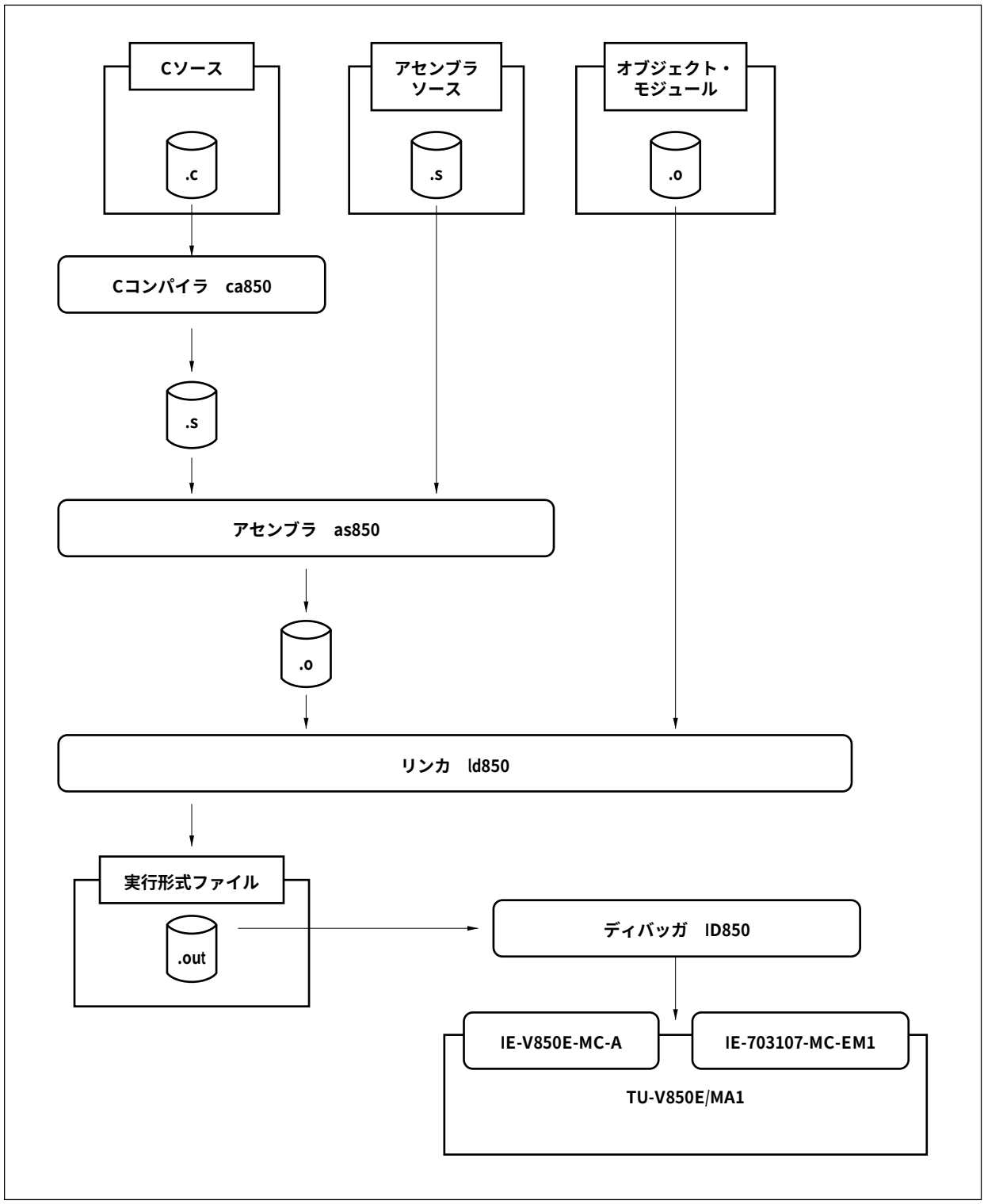
- IE-V850E-MC, IE-V850E-MC-A　インサーキット・エミュレータ　（U14487J）

(4) IE-703107-MC-EM1

V850E/MA1用インサーキット・エミュレータ・オプション・ボードに関するユーザーズ・マニュアルです。

- IE-703107-MC-EM1　インサーキット・エミュレータ・オプション・ボード　（U14481J）

図4－4　開発ツールの構成



はじめてNEC**製開発ツールを使う方は導入として次のマニュアルが便利です。**

V800**シリーズ**™**開発ツール（**32**ビット対応）チュートリアル・ガイド（**Windows**ベース）（**U15196J**）**

4. 2. 2　プログラムの構成

アプリケーション・プログラム例は次のような構成になっています。①→②→③の順に，CPUにとって使用可能な資源が広がっています。

① 4. 2. 3　シングルチップ・モード0での動作例（STEP1）
シングルチップ・モード0でV850E/MA1を動作させます。
CPUの動作のみ確認できます。

② 4. 2. 4　シングルチップ・モード0＋拡張モードでの動作例（STEP2）
シングルチップ・モード0かつ外部拡張モードでV850E/MA1を動作させます。
CPUと外部バス（メモリなど）を接続した動作が確認できます。

③ 4. 2. 5　各種周辺機能を使った動作例（STEP3）
①＋②をベースに内蔵周辺I/Oや外部デバイスなどの動作を確認します。
③は，次の2つの初期化処理を行ってからアプリケーション・プログラムが動作するようになっています。
・アプリケーション・プログラム例に出てくる初期化ファイル
〔cpuinit.s〕：CPU基本機能初期化部分
〔businit.s〕：外部メモリ空間を使う場合のCPUバス制御機能初期化部分

4. 2. 3　シングルチップ・モード0での動作例（STEP1）

CPU基本機能の初期化のみ行い，外部空間は使用せずにV850E/MA1を動作させます。

（1）CPU基本機能初期化のポイント

V850E/MA1において周辺I/Oレジスタの初期化では，必ず最初に次のレジスタを設定してください。

・システム・ウエイト・コントロール・レジスタ（VSWC）
・クロック・コントロール・レジスタ（CKC）

そのあと，必要に応じてほかの各レジスタを設定してください。

★ **図4－5　システム・ウエイト・コントロール・レジスタ（VSWC）の設定**

アドレス：FFFFF06EH

	7	6	5	4	3	2	1	0
初期値	0	1	1	1	0	1	1	1
ビット名	－	－	－	－	－	－	－	－
設定値	0	0	0	1	0	0	1	0

動作周波数（f_{XX}）	VSWCの設定値
4 MHz≦f_{XX}＜33 MHz	11H
33 MHz≦f_{XX}≦50 MHz	12H（推奨）または13H
初期値	77H

ここでは，40 MHzで動作させるため，VSWCレジスタは，12Hを設定します。

(2) CPUクロック初期化のポイント

クロック・コントロール・レジスタへのデータ設定は特定シーケンスで行ってください。

図4－6　クロック・コントロール・レジスタ（CKC）の設定

アドレス：FFFFF822H

	7	6	5	4	3	2	1	0
初期値	0	0	0	0	0	0	0	0
ビット名	0	0	TBCS	CESEL	0	CKDIV2	CKDIV1	CKDIV0
設定値	0	0	0	1	0	1	1	1

ビット位置	ビット名	意　味
5	TBCS	タイム・ベース・カウンタのクロック選択 0：$f_X/2^8$
4	CESEL	X1, X2端子機能の選択 1：X1端子に外部クロックを接続
2-0	CKDIV2- CKDIV0	内部システム・クロック周波数（f_{XX}）の決定 111：10×f_X

【プログラム例】

```
#### CPU基本機能初期化 ####
_cpu_initial:
★    mov    0x12,  r6          -- 動作周波数（fxx）33MHz≦fxx≦50MHzを選択
     st.b   r6,    VSWC[r0]    -- システム・ウエイト・コントロール・レジスタを設定

####  特定シーケンスによるアクセス   ####
     mov    0x00a0,r6          -- ①NMI割り込み禁止
     ldsr   r6,    5           --   PSWのNP bitを1に設定
     mov    0x17,  r6          -- ②汎用レジスタにCKCに設定するデータを用意
                               --   X1端子に外部クロック接続，fxx=10×fx
     st.b   r6,    PHCMD[r0]   -- ③ペリフェラル・コマンド・レジスタにダミー・データを書き込み
     st.b   r6,    CKC[r0]     -- ④クロック・コントロール・レジスタを設定
     nop                       -- ⑤NOP命令挿入（5命令発行）
     nop                       -- ⑥NOP命令挿入
     nop                       -- ⑦NOP命令挿入
     nop                       -- ⑧NOP命令挿入
     nop                       -- ⑨NOP命令挿入
     mov    0x0020,r6          -- ⑩NMI割り込み禁止を解除
     ldsr   r6,    5           --   PSWのNP bitを0に戻す

# 内蔵RAM領域（xfffc000番地）に固定データを書き込む。
# ＜使用レジスタ＞
#     r10：内蔵RAMアドレス
#     r11：設定データ（55H）

#外部参照宣言
.extern  _cpu_init

#リセットおよび割り込み
.section "RESET",text          -- リセット・ハンドラ宣言
     jr     _start             -- サンプル・プログラム・コード部の先頭へジャンプ

#定数定義
.set internal_ram, 0xfffc000   -- 内蔵RAM先頭アドレス
.set write_data,   0x55        -- 書き込みデータ

#プログラム本体
.text
.align 4                       -- 4byteで整列
.globl _start                  -- 外部宣言
_start:
     jarl   _cpu_init,    r31  -- CPU基本機能初期化部
     mov    internal_ram, r10  -- 書き込みアドレス設定
     mov    write_data,   r11  -- 書き込みデータ設定
     st.b   r11,   0x0[r10]    -- バイトで書き込み
_forever:
     br     _forever           -- 永久ループ
```

4. 2. 4　シングルチップ・モード0＋拡張モードでの動作例（STEP2）

4. 2. 3　**シングルチップ・モード0での動作例（STEP1）のCPU基本機能初期化に加えて，CPUバス制御機能初期化を行っており，外部空間のアクセスが可能となります。**

(1) 端子機能初期化のポイント

シングルチップ・モード0を選択し，コントロール・モード時に出力／入出力動作する端子をポート・モードからコントロール・モードに切り替える場合には，必ず次の手順で設定してください。

① **コントロール・モードで出力する信号のインアクティブ・レベルをポートn（Pn）の該当するビットに設定します。**

② **ポートnモード・コントロール・レジスタ（PMCn）により，コントロール・モードに切り替えます。**（n = 0-5, AL, AH, DL, CS, CT, CM, CD, BD）

備考　**①を行わない場合は，ポート・モードからコントロール・モードに切り替える際にポートnの内容が一瞬出力されることがあります。**

メモリ・アクセスに関わる端子の初期値は，動作モード（MODE0-MODE2**端子の状態で決定）によって異なります。**

	右記以外のモード	シングルチップ・モード0 （内蔵ROM領域を00000000H番地から配置）
PMCAL	アドレス・バスの下位16ビット	ポート
PMCAH	アドレス・バスの上位10ビット	ポート
PMCDL	データ・バスの16ビット	ポート
PMCCS	チップ・セレクト信号	ポート
PMCCT	バス・サイクル関連信号	ポート
PMCCM	DRAM関連信号	ポート
PMCCD	SDRAM関連信号	ポート

備考　**シングルチップ・モード0では，バス制御機能端子ではなくポート端子として起動しています。**

	右記以外のモード	ROMレス・モード1（8ビット・データ・バス空間）
BSC	16ビット・データ・バス幅	8ビット・データ・バス幅

	右記以外のモード	フラッシュ・メモリ・プログラミング
PMC4	ポート	CSI0端子（$\overline{SCK0}$, SI0, SO0）

【プログラム例】

```
[file:bus_init.s]

#########################################################
#    バス制御端子を指定
#########################################################
     st.h    r0,     PAL[r0]        -- A0-A15出力信号のインアクティブ・レベル Loを設定
     mov     0xffff,r6              -- A0-A15端子選択
     st.h    r6,     PMCAL[r0]

     st.h    r0,     PAL[r0]        -- A16-A25出力信号のインアクティブ・レベル Loを設定
     mov     0x03ff,r6              -- A16-A25端子選択
     st.h    r6,     PMCAH[r0]

     st.h    r0,     PDL[r0]        -- D0-D15出力信号のインアクティブ・レベル Loを設定
     mov     0xffff,r6              -- D0-D15端子選択
     st.h    r6,     PMCDL[r0]

     mov     0xff,   r6             -- CS0-CS7出力信号のインアクティブ・レベル Hiを設定
     st.b    r6,     PCS[r0]
     mov     0xff,   r6             -- CS0-CS7端子選択
     st.b    r6,     PMCCS[r0]

     mov     0xf3,   r6             -- BCYST,OE,WE,RD,UCAS/UWR/UDQM,LCAS/LWR/LDQM出力信号の
     st.b    r6,     PCT[r0]        -- インアクティブ・レベル Hiを設定
     mov     0xf3,   r6             -- BCYST,OE,WE,RD,UCAS/UWR/UDQM,LCAS/LWR/LDQM端子選択
     st.b    r6,     PMCCT[r0]

     mov     0x14,   r6             -- REFREQ,HLDAK出力信号のインアクティブ・レベル Hiを設定
     st.b    r6,     PCM[r0]        -- CLKOUT出力信号のインアクティブ・レベル Hiを設定
     mov     0x32,   r6             -- SELFREF,REFRQ,CLKOUT端子選択
     st.b    r6,     PMCCM[r0]
     mov     0x0c,   r6             -- UBE/SDRAS,LBE/SDCAS出力信号のインアクティブ・レベルHiを設定
     st.b    r6,     PCD[r0]        -- SDCKE,SDCLK出力信号のインアクティブ・レベル Loを設定
     mov     0x02,   r6             -- 注意：初めにSDCLK端子を選択
     mov     0x0d,   r7             -- UBE/SDRAS,LBE/SDCAS,SDCKE端子選択
     st.b    r6,     PMCCD[r0]
     ld.b    PMCCD[r0],r6
     or      r7,     r6
     st.b    r6,     PMCCD[r0]      -- UBE/SDRAS,LBE/SDCAS,SDCLK,SDCKE端子選択

     mov     0x0c,   r6             -- SDRAS,SDCAS端子選択
     st.b    r6,     PFCCD[r0]
```

```
#######################################################
#    バス制御端子以外の機能を指定
#######################################################
     st.b    r0,     P0[r0]          -- PWM0出力信号のインアクティブ・レベル Loを設定
     mov     0x03,   r6              -- PWM0,INTP000端子選択
     st.b    r6,     PMC0[r0]

     st.b    r0,     P2[r0]          -- TO02出力信号のインアクティブ・レベル Loを設定
     mov     0x39,   r6              -- NMI,TO02,INTP110,INTP111端子選択
     st.b    r6,     PMC2[r0]

     mov     0x04,   r6              --P37-P34（LED）のインアクティブ・レベル Lo（消灯）を設定
                                     --P33（EPROMへのCS）のインアクティブ・レベル Loを設定
                                     --SCK2入出力信号のインアクティブ・レベル Hiを設定
                                     --SO2出力信号のインアクティブ・レベル Loを設定
     st.b    r6,     P3[r0]
     mov     0x07,   r6
     st.b    r6,     PM3[r0]         -- P37-P33（LED）,P33（EEPROMへのCS）出力モード
     st.b    r6,     PMC3[r0]        -- SO2,SI2,SCK2端子選択

     st.b    r0,     PBD[r0]         -- PBD3-PBD0（LED）のインアクティブ・レベル Lo（消灯）を設定
     mov     0xf0,   r6              -- PBD0-PBD3出力モード（LED）
     st.b    r6,     PMBD[r0]


     mov     0x08,   r6              -- TXD1（PORT43）出力信号High Level
     st.b    r6,     P4[r0]
     mov     0x18,   r6              -- RXD1/SI1,TXD1/SO1端子選択
     st.b    r6,     PMC4[r0]
     mov     0x18,   r6              -- RXD1入力,TXD1出力モード選択
     st.b    r6,     PFC4[r0]
```

(2) メモリ・ブロック空間分割初期化のポイント

256 Mバイトのメモリ空間が64 Mバイトずつ4つの領域（エリア0-エリア3）に分割されています。

エリア0：　64 Mバイト（00000000H-03FFFFFFH）主に$\overline{CS1}$信号

エリア1：　64 Mバイト（04000000H-07FFFFFFH）$\overline{CS3}$信号固定

エリア2：　64 Mバイト（08000000H-0BFFFFFFH）$\overline{CS4}$信号固定

エリア3：　64 Mバイト（0C000000H-0FFFFFFFH）主に$\overline{CS6}$信号

さらに，次のように2 Mバイト単位でメモリ・ブロックに分割し，チップ・セレクト信号を細かく指定できます。

エリア0：下位8 Mバイト（00000000H-007FFFFFH）$\overline{CS0}$, $\overline{CS1}$, $\overline{CS2}$信号選択可能

エリア3：上位8 Mバイト（0F800000H-0FFFFFFFH）$\overline{CS7}$, $\overline{CS6}$, $\overline{CS5}$信号選択可能

図4－7　チップ・エリア選択コントロール・レジスタ0（CSC0）の設定

アドレス：FFFFF060H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
初期値	0	0	1	0	1	1	0	0	0	0	0	1	0	0	0	1
ビット名	CS33	CS32	CS31	CS30	CS23	CS22	CS21	CS20	CS13	CS12	CS11	CS10	CS03	CS02	CS01	CS00
設定値	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1

設定に意味を持ちません。　　設定に意味を持ちません。

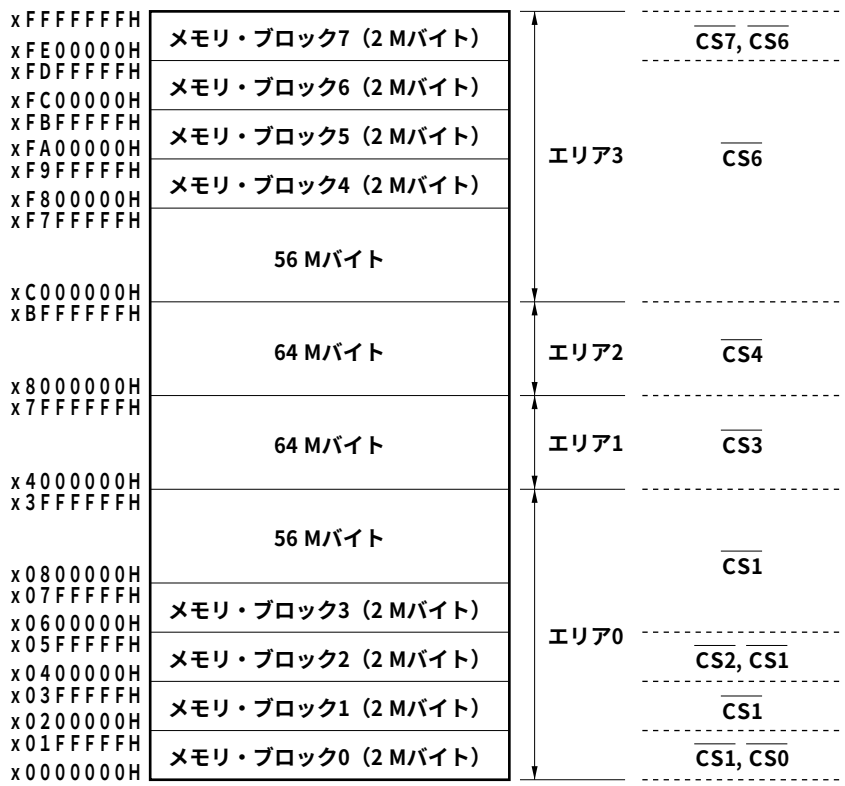
図4－8　チップ・エリア選択コントロール・レジスタ1（CSC1）の設定

アドレス：FFFFF062H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
初期値	0	0	1	0	1	1	0	0	0	0	0	1	0	0	0	1
ビット名	CS43	CS42	CS41	CS40	CS53	CS52	CS51	CS50	CS63	CS62	CS61	CS60	CS73	CS72	CS71	CS70
設定値	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

設定に意味を持ちません。　　設定に意味を持ちません。

図4－9　各$\overline{\text{CSn}}$信号領域



備考1.　**このプログラム例では**$\overline{\text{CS5}}$**は使用しません。**

2.　**エリア0の優先順位：**$\overline{\text{CS0}}$＞$\overline{\text{CS2}}$＞$\overline{\text{CS1}}$

エリア3の優先順位：$\overline{\text{CS7}}$＞$\overline{\text{CS6}}$

3.　n = 0-7

(3) メモリ接続デバイス初期化のポイント

BCT0, BCT1レジスタへの書き込みは，リセット後に行ってください。

書き込み後は値を変更しないでください。

図4－10　バス・サイクル・タイプ・コンフィギュレーション・レジスタ0, 1（BCT0, BCT1）の設定

BCT0　　アドレス：FFFFF480H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
初期値	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0
ビット名	ME3	0	BT31	BT30	ME2	0	0	BT20	ME1	0	BT11	BT10	ME0	0	0	BT00
設定値	1	0	1	0	1	0	0	1	1	0	0	0	1	0	0	0
			$\overline{CS3}$					$\overline{CS2}$			$\overline{CS1}$					$\overline{CS0}$

BCT1　　アドレス：FFFFF482H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
初期値	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0
ビット名	ME7	0	0	BT70	ME6	0	BT61	BT60	ME5	0	0	BT50	ME4	0	BT41	BT40
設定値	1	0	0	0	1	0	0	0	0	0	0	0	1	0	1	1
				$\overline{CS7}$			$\overline{CS6}$					$\overline{CS5}$			$\overline{CS4}$	

ビット位置	ビット名	意　味
15, 11, 7, 3（BCT0），15, 11, 7, 3（BCT1）	MEn（n = 0-7）	メモリ・コントローラの動作許可をチップ・セレクト信号nごとに選択 0：動作禁止（ME5） 1：動作許可（ME0-ME4, ME6, ME7）
8, 0（BCT0），12, 4（BCT1）	BTn0（n = 0, 2, 5, 7）	直接接続するデバイスをチップ・セレクト信号nごとに指定 0：SRAMまたは外部I/O（BT00, BT50, BT70） 1：ページROM（BT20）
13, 12, 5, 4（BCT0）9, 8, 1, 0（BCT1）	BTn1, BTn0（n = 1, 3, 4, 6）	直接接続するデバイスをチップ・セレクト信号nごとに指定 00：SRAMまたは外部I/O（BT11/BT10, BT61/BT60） 10：EDO DRAM（BT31/BT30） 11：SDRAM（BT41/BT40）

【プログラム例】

```
[file:bus_init.s]

####  メモリ・ブロック機能初期化  ####
     mov    0x0401,r6           -- エリア0（64MB）
                                -- 00000000H-001fffffH:CS0（ 2MB）
                                -- 00200000H-003fffffH:CS1（ 2MB）
                                -- 00400000H-005fffffH:CS2（ 2MB）
                                -- 00600000H-03ffffffH:CS1（58MB）
                                -- エリア1（64MB）
                                -- 04000000H-07ffffffH:CS3（64MB）
                                -- エリア2（64MB）
                                -- 08000000H-0bffffffH:CS4（64MB）
     mov    0x0001,r7           -- エリア3（64MB）
                                -- 0c000000H-0fdfffffH:CS6（62MB）
                                -- 0fe00000H-0fffffffH:CS7（ 2MB）
     st.h   r6,    CSC0[r0]     -- チップ・エリア選択コントロール・レジスタ0
     st.h   r7,    CSC1[r0]     -- チップ・エリア選択コントロール・レジスタ1

     mov    0xa988,r6           -- CS0:接続デバイスなし
                                -- CS1:SRAM          00200000H-002fffffH（1MB）
                                -- CS2:PAGE ROM      00400000H-0047ffffH（512KB）
                                -- CS3:EDO DRAM      04000000H-041fffffH（2MB）

     mov    0x880b,r7
                                -- CS4:SDRAM         08000000H-087fffffH（8MB）
                                -- CS5:空間なし
                                -- CS6:Flash ROM     c0000000H-c003ffffH（256KB）
                                -- CS7:I/O            fe000000H-fe0000fH（16Byte）
     st.h   r6,    BCT0[r0]     -- バス・サイクル・タイプ・コンフィギュレーション・レジスタ0
     st.h   r7,    BCT1[r0]     -- バス・サイクル・タイプ・コンフィギュレーション・レジスタ1
```

(4) バス・サイズ初期化のポイント

BSCレジスタへの書き込みは，リセット後に行ってください。

書き込み後は値を変更しないでください。

図4－11　バス・サイズ・コンフィギュレーション・レジスタ（BSC）の設定

アドレス：FFFFF066H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
初期値	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
ビット名	0	BS70	0	BS60	0	BS50	0	BS40	0	BS30	0	BS20	0	BS10	0	BS00
設定値	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1
		$\overline{CS7}$		$\overline{CS6}$		$\overline{CS5}$		$\overline{CS4}$		$\overline{CS3}$		$\overline{CS2}$		$\overline{CS1}$		$\overline{CS0}$

ビット位置	ビット名	意　味
14, 12, 10, 8, 6, 4, 2, 0	BSn0 (n = 0-7)	メモリ・コントローラの動作許可をチップ・セレクト信号nごとに選択 0：8ビット（BS70） 1：16ビット（BS00-BS60）

備考　ROMレス・モード1のときの初期値は，全空間8ビット・データ・バス幅の0000Hです。

(5) エンディアン形式初期化のポイント

NEC製純正開発ツール（ディバッガやコンパイラ）においては，ビッグ・エンディアン形式の場合，種々の使用制限事項がありますので注意してください。

制限事項に関しては，V850E/MA1　**ユーザーズ・マニュアル　ハードウエア編　の**4. 5. 4　NEC**製開発ツールにおけるビッグ・エンディアン形式の使用制限**を参照してください。

図4－12　**エンディアン・コンフィギュレーション・レジスタ（**BEC**）の設定**

アドレス：FFFFF068H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
初期値	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
ビット名	0	BE70	0	BE60	0	BE50	0	BE40	0	BE30	0	BE20	0	BE10	0	BE00
設定値	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		$\overline{CS7}$		$\overline{CS6}$		$\overline{CS5}$		$\overline{CS4}$		$\overline{CS3}$		$\overline{CS2}$		$\overline{CS1}$		$\overline{CS0}$

ビット位置	ビット名	意　味
14, 12, 10, 8, 6, 4, 2, 0	BEn0 (n = 0-7)	**メモリ・コントローラの動作許可をチップ・セレクト信号nごとに選択** 0：**リトル・エンディアン**（BE70-BE00）

【プログラム例】

```
[file:bus_init.s]

####  バス・アクセス初期化  ####
     mov    0x1555,r6           -- CS0-CS6（Memory）空間 16bit bus
                                -- CS7    （I/O）   空間  8bit bus
     st.h   r6,    BSC[r0]      -- バス・サイズ・コンフィギュレーション・レジスタ
     mov    0x0000,r6           -- CS0-CS7全空間 リトル・エンディアン
     st.h   r6,    BEC[r0]      --エンディアン・コンフィギュレーション・レジスタ
```

(6) プログラマブル・ウエイト初期化のポイント

- 内蔵ROM領域，内蔵RAM領域，内蔵周辺I/O領域はプログラマブル・ウエイトの対象外です。
- ページROMオンページ・アクセス，EDO DRAMアクセス，SDRAMアクセスのウエイト制御は各メモリ・コントローラごとに行います。
- DWC0, DWC1レジスタへの書き込みは，リセット後に行ってください。書き込み後は値を変更しないでください。

図4－13　データ・ウエイト・コントロール・レジスタ0, 1（DWC0, DWC1）の設定

DWC0　　アドレス：FFFFF484H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
初期値	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1
ビット名	0	DW32	DW31	DW30	0	DW22	DW21	DW20	0	DW12	DW11	DW10	0	DW02	DW01	DW00
設定値	0	0	0	0	0	1	0	1	0	0	0	1	0	1	0	1
		$\overline{CS3}$				$\overline{CS2}$				$\overline{CS1}$				$\overline{CS0}$		

DWC1　　アドレス：FFFFF486H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
初期値	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1
ビット名	0	DW72	DW71	DW70	0	DW62	DW61	DW60	0	DW52	DW51	DW50	0	DW42	DW41	DW40
設定値	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0
		$\overline{CS7}$				$\overline{CS6}$				$\overline{CS5}$				$\overline{CS4}$		

ビット位置	ビット名	意　味
14-12, 10-8, 6-4, 2-0	DWn2-DWn0 (n = 0-7)	CSn空間に挿入するウエイトのステート数を指定 ・$\overline{CS0}$（EPROM空間） 101：5ウエイト挿入（DW02-DW00） ・$\overline{CS1}$（SRAM空間） 001：1ウエイト挿入（DW12-DW10） ・$\overline{CS2}$（ページROM空間） 101：5ウエイト挿入（DW22-DW20） ・$\overline{CS3}$-$\overline{CS5}$ 000：挿入しない（DWn2-DWn0：n = 3-5） ・$\overline{CS6}$（フラッシュ・メモリ空間） 111：7ウエイト挿入（DW62-DW60） ・$\overline{CS7}$（I/O空間） 001：1ウエイト挿入（DW72-DW70）

(7) アドレス・セットアップ・ウエイト初期化のポイント

- EDO DRAM, SDRAM領域はアドレス・セットアップ・ウエイトの対象外です。
- $\overline{\text{WAIT}}$端子による外部ウエイト機能でアドレス・セットアップ・ウエイトを挿入することはできません。

図4－14　アドレス・セットアップ・ウエイト・コントロール・レジスタ（ASC）の設定

アドレス：FFFFF48AH

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
初期値	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ビット名	AC71	AC70	AC61	AC60	AC51	AC50	AC41	AC40	AC31	AC30	AC21	AC20	AC11	AC10	AC01	AC00
設定値	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	$\overline{\text{CS7}}$		$\overline{\text{CS6}}$		$\overline{\text{CS5}}$		$\overline{\text{CS4}}$		$\overline{\text{CS3}}$		$\overline{\text{CS2}}$		$\overline{\text{CS1}}$		$\overline{\text{CS0}}$	

それぞれのSRAM，ページROM空間に合った適切なウエイトを選択します。

ビット位置	ビット名	意　味
15-0	ACn1, ACn0 (n = 0-7)	CSn空間に挿入するウエイトのステート数を指定 00：挿入しない（ACn1, ACn0（n = 0-7））

図4－15　バス・サイクル・ピリオド・コントロール・レジスタ（BCP）の設定

アドレス：FFFFF48CH

	7	6	5	4	3	2	1	0
初期値	0	0	0	0	0	0	0	0
ビット名	BCP	0	0	0	IOEN	0	0	0
設定値	0	0	0	0	0	0	0	0

ビット位置	ビット名	意　味
7	BCP	バス・サイクル周期の選択 0：通常
3	IOEN	SRAM，外部ROM，外部I/Oサイクルにおける$\overline{\text{IORD}}$, $\overline{\text{IOWR}}$動作の許可／禁止 0：禁止

(8) バス・サイクル周期初期化のポイント

- SRAM，外部ROM，外部I/Oを対象としたフライバイのDMA転送時は，IOENビットの設定にかかわらず $\overline{\text{IORD}}$, $\overline{\text{IOWR}}$信号が出力されます。
- バス・サイクル周期を内部システム・クロックの1/2にしたい場合は，BUSCLK端子を使用します。BCPレジスタのIOENビットをセット（1）したあとに，PMCCMとPFCCMレジスタを必ず設定してください。
- BCPレジスタへの書き込みは，リセット後に行ってください。書き込み後は値を変更しないでください。

【プログラム例】

```
[file:businit.s]

#### ウエイト機能初期化 ####
     mov    0x0515,r6          -- CS0:EPROM          空間 5 wait（TW）挿入
                               -- CS1:SRAM           空間 1 wait（TW）挿入
                               -- CS2:PAGE ROM       空間 5 wait（TW）挿入
                               -- CS3:EDO DRAM       空間 0 wait（TW）
     mov    0x1700,r7          -- CS4:SDRAM          空間 0 wait（TW）
                               -- CS6:FlashROM       空間 7 wait（TW）挿入
                               -- CS7:I/O            空間 1 wait（TW）挿入
     st.h   r6,    DWC0[r0]    -- データ・ウエイト・コントロール・レジスタ0
     st.h   r7,    DWC1[r0]    -- データ・ウエイト・コントロール・レジスタ1

     mov    0x0000,r6          -- CS0-CS7       全空間 0 wait（TASW）
     st.h   r6,    ASC[r0]     -- アドレス・セットアップ・コントロール・レジスタ

     mov    0x00,  r6          -- 通常周期
                               -- SRAM,外部ROM,外部I/Oサイクル時のIORD,IOWR動作禁止
     st.b   r6,    BCP[r0]     -- バス・サイクル・ピリオド・コントロール・レジスタ
```

(9) アイドル・ステート初期化のポイント

・内蔵ROM領域，内蔵RAM領域，内蔵周辺I/O領域はアイドル・ステート挿入の対象外です。

・BCCレジスタへの書き込みは，リセット後に行ってください。書き込み後は値を変更しないでください。

図4－16　バス・サイクル・コントロール・レジスタ（BCC）の設定

アドレス：FFFFF488H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
初期値	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
ビット名	BC71	BC70	BC61	BC60	BC51	BC50	BC41	BC40	BC31	BC30	BC21	BC20	BC11	BC10	BC01	BC00
設定値	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	1
	$\overline{CS7}$		$\overline{CS6}$		$\overline{CS5}$		$\overline{CS4}$		$\overline{CS3}$		$\overline{CS2}$		$\overline{CS1}$		$\overline{CS0}$	

それぞれのメモリ空間に合った適切なアイドルを選択します。

ビット位置	ビット名	意　味
15-0	BCn1, BCn0 (n = 0-7)	$\overline{CSn}$空間に挿入するウエイトのステート数を指定 ・$\overline{CS0}$（EPROM空間） 11：3アイドル挿入（BC01, BC00） ・$\overline{CS1}$（SRAM空間） 00：挿入しない（BC11, BC00） ・$\overline{CS2}$（ページROM空間） 10：2ウエイト挿入（BC21, BC20） ・$\overline{CS3}$-$\overline{CS5}$ 00：挿入しない（BCn1, BCn0：n = 3-5） ・$\overline{CS6}$（フラッシュ・メモリ空間） 10：2ウエイト挿入（BC61, BC60） ・$\overline{CS7}$（I/O空間） 00：挿入しない（BC71, BC70）

【プログラム例】

```
[file:businit.s]
#### アイドル機能初期化 ####
mov  0x2023,r6            -- CS0: EPROM              空間 3 idle（TI）挿入
                          -- CS1: SRAM               空間 0 idle（TI）
                          -- CS2: PAGEROM            空間 2 idle（TI）挿入
                          -- CS3: EDOROM             空間 0 idle（TI）
                          -- CS4: SDRAM              空間 0 idle（TI）
                          -- CS6: FlashROM           空間 2 idle（TI）挿入
                          -- CS7: I/O                空間 0 idle（TI）
st.h r6,    BCC[r0]       -- バス・サイクル・コントロール・レジスタ
```

（10）SDRAM初期化のポイント

① SDRAMを使用する場合は，必ず次の初期化シーケンスに従ってください。
- PMCn（ポートnモード・コントロール・レジスタ）を設定する（n = 0-5, AL, AH, DL, CS, CT, CM, CD）。
- CSCn（チップ・エリア選択コントロール・レジスタn）を設定し，チップ・セレクト空間を確定する（n = 0, 1）。
- BCTn（バス・サイクル・タイプ・コンフィギュレーション・レジスタn）の設定により，各チップ・セレクト空間のメモリの種類を確定する（n = 0, 1）。
- RFSn（SDRAM用リフレッシュ・コントロール・レジスタn）を設定する（n = 1, 3, 4, 6）。
- SCRn（SDRAMコンフィギュレーション・レジスタn）を設定する（n = 1, 3, 4, 6）。

② SDRAMコンフィギュレーション・レジスタn（SCRn）
- SDRAM領域をアクセスする場合，レジスタ設定直後は20クロック待ってください。
- SDRAM領域を複数初期化する場合，レジスタに書き込む命令と命令の間には必ずほかの命令を入れてください。
- レジスタへの書き込みはリセット後に行ってください。書き込み後は値を変更しないでください。

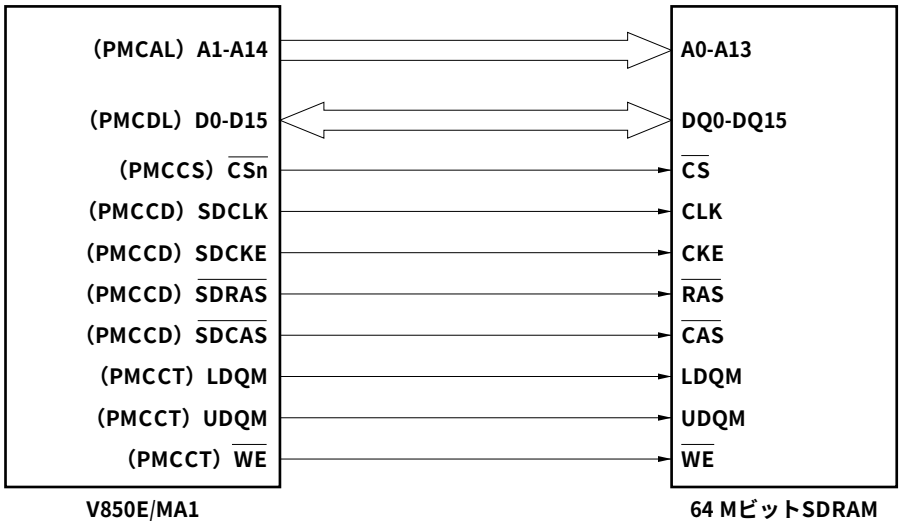
③ SDRAM用リフレッシュ・コントロール・レジスタn（RFSn）

レジスタへの書き込みはリセット後に行ってください。書き込み後は値を変更しないでください。

④ リフレッシュ・ウエイト・コントロール・レジスタ（RWC）

レジスタへの書き込みはリセット後に行ってください。書き込み後は値を変更しないでください。

図4－17　SDRAM初期化シーケンスのイメージ



① PMCn（ポートnモード・コントロール・レジスタ）をSDRAM（外部メモリ空間）を使用する外部バス状態に設定します（n = 0-5, AL, AH, DL, CS, CT, CM, CD）。PMCCDレジスタを設定する場合，SDCLK出力モードとSDCKE出力モードを同時に設定しないでください。必ずSDCLK出力モードの設定を行ったあとに，SDCKE出力モードを設定してください。

② CSCn（チップ・エリア選択コントロール・レジスタn）を設定し，チップ・セレクト空間を確定します（n = 0, 1）。4. 2. 4（2）メモリ・ブロック空間分割初期化のポイントを参照してください（SDRAM用に$\overline{\text{CS4}}$空間64 Mバイト）。

③ BCTn（バス・サイクル・タイプ・コンフィギュレーション・レジスタn）の設定により，$\overline{\text{CS4}}$空間のメモリ種類をSDRAMとします（n = 0, 1）。

④ RFSn（SDRAM用リフレッシュ・コントロール・レジスタn）（n = 1, 3, 4, 6）

注

SCRn（SDRAMコンフィギュレーション・レジスタn）（n = 1, 3, 4, 6）

注　この順序で設定してください。

（11）ページROM初期化のポイント

・PRC（ページROMコンフィギュレーション・レジスタ）への書き込みはリセット後に行ってください。
　書き込み後は値を変更しないでください。
・オンページ・アクセスの場合，PRCの設定によってウエイト制御されます。
・オフページ・アクセスの場合，DWCnの設定によってウエイト制御されます。

【プログラム例】

```
####  メモリ・アクセス制御機能初期化  ####
[file:businit.s]
#########################################################
#     EDO  DRAMの設定
#########################################################
     mov    0x98,  r6            -- RAS信号ハイ・レベル幅に 2 wait（TRRW）挿入
                                 -- RAS信号ロウ・レベル幅に 3 wait（TRCW）挿入
                                 -- セルフ・リフレッシュ解除に  0 wait（TSRW）
     st.b   r6,    RWC[r0]       -- リフレッシュ・ウエイト・コントロール・レジスタ

     mov    0xb546, r6           -- オンページ・アクセス許可
                                 -- ロウ・アドレス PRE時間  3 wait（TRPW）挿入
                                 -- ロウ・アドレスHOLD時間  1 wait（TRHW）挿入
                                 -- データ・アクセス時間に  1 wait（TW）挿入
                                 -- カラム・アドレス PRE時間 1 wait（TCPW）挿入
                                 -- RASホールド・モード許可
                                 -- アドレス・シフト幅      16 bit（On-page）
                                 -- アドレス・マルチプレクス幅 10 bit
     st.h   r6,DRC3[r0]          -- DRAM コンフィギュレーション・レジスタ3

     mov    0x8106, r6           -- リフレッシュ許可
                                 -- リフレッシュ・カウント・クロック=128/fxx
                                 -- リフレッシュ・インターバル・ファクタ=7（50MHz時17.92us）
     st.h   r6,RFC3[r0]          -- リフレッシュ・コントロール・レジスタ3
```

```
#########################################################
#     SDRAMの設定
#########################################################
     mov    0x8106, r6          -- リフレッシュ許可
                                -- リフレッシュ・カウント・クロック=128/fxx
                                -- リフレッシュ・インターバル・ファクタ=7（50MHz時17.92us）
     st.h   r6,RFC4[r0]         -- SDRAM用リフレッシュ・コントロール・レジスタ4

     mov    0x2014, r6          -- リード時CAS Latency 2 Latency（TLATE）
                                -- ACT-RD,PRE-ACT時に1 wait（TBCW）
                                -- アドレス・シフト幅      16 bit（On-page）
                                -- ロウ・アドレス幅        12 bit
                                -- アドレス・マルチプレクス幅  8 bit
     st.h   r6,DRC4[r0]         -- DRAM コンフィギュレーション・レジスタ4

#########################################################
#     Page ROMオンページ・アクセス時のウエイトの指定など
#########################################################
     mov    0x2000, r6          -- 4WORD×16bit（8WORD×8bit）
                                -- オンページ・アクセス時に 2 wait（TW）挿入
     st.h   r6,PRC[r0]          -- ページROMコンフィギュレーション・レジスタ
```

4.2.5　各種周辺機能を使った動作例（STEP3）

CPU基本機能初期化に加えてCPUバス制御機能初期化を行っております。さらに，各種内蔵周辺I/Oの初期化を行うことで，さまざまな資源を使った動作が可能となります。

(1) ポート機能による入出力

(a) 機能概要

入力ポートに接続されたスイッチの状態を読み取り，出力ポートに接続されたLEDへ表示します。

8個のスイッチ入力はP0とP5の2つのポートに分散して割り付けられており，8個のLEDもP3とPBDの2つのポートに分散して割り付けられています。このため読み取ったデータを編集した上で出力しています。

(b) プログラム・リスト

```
# トグル・スイッチに接続されている入力ポートであるPORT0とPORT5
# を加工し，LEDに接続されている出力ポートのPORT3とPBDへ出力する。
# ＜使用レジスタ＞
#     r10：PORT0入力用バッファ
#     r11：PORT5入力用バッファ
#     r12：PBD出力用バッファ

#外部参照宣言
.extern  _cpu_init
.extern  _bus_init

#リセットおよび割り込み
.section "RESET",text      -- リセット・ハンドラ宣言
     jr    _start          -- サンプル・プログラム・コード部の先頭へジャンプ
#プログラム本体
.text
.align 4                   -- 4byteで整列
.globl _start              -- 外部宣言

_start:
     jarl  _cpu_init,r31   -- CPU基本機能初期化部
      jarl _bus_init,r31   -- CPUバス制御機能初期化部
_io_loop:
     ld.b  P0[r0],r10
     andi  0xfc,  r10,     r10
     ld.b  P5[r0],r11
     andi  0x03,  r11,     r11
     or    r11,   r10
     andi  0xf0,  r10,     r12
     st.b  r12,   P3[r0]
     andi  0x0f,  r10,     r12
     st.b  r12,   PBD[r0]
     jr    _io_loop
```

(2) 外部割り込み処理による表示処理

(a) 機能概要

$\overline{\text{INTP110}}$端子に接続されたINTスイッチが押されるたびに，出力ポートに全点灯，全消灯を繰り返します。このプログラムでは，レジスタの保存をしていませんが，実際の割り込み処理プログラムでは，レジスタの退避，復帰が必要になります。

(b) プログラム・リスト

```
# INTスイッチに接続されているINTP110によって外部割り込みを発生させ
# INTスイッチ押すごとに，LEDに接続されている出力ポートPORT3とPBDに
# 全点灯と全消灯を繰り返し行う。
# ＜使用レジスタ＞
#     r10：LED上位4bit取り出しバッファ（テンポラリ）
#     r11：LED下位4bit取り指しバッファ
#     r12：LEDフラグ

#外部参照宣言
.extern  _cpu_init
.extern  _bus_init

#リセットおよび割り込み
.section "RESET",text      -- リセット・ハンドラ宣言
     jr      _start        -- サンプル・プログラム・コード部の先頭へジャンプ
.section "INTP110",text    -- INTP110割り込みハンドラ宣言
     jr      _int_p110     -- INTSW割り込み処理部の先頭へジャンプ

#プログラム本体
.text
.align 4                   -- 4byteで整列
.globl _start              -- 外部宣言

_start:
     jarl    _cpu_init,r31 -- CPU基本機能初期化部
     jarl    _bus_init,r31 -- CPUバス制御機能初期化部

     st.b    r0,     INTM2 -- INTP110割り込み立ち下りエッジ指定
     mov     0x07,   r10   -- INTP110割り込みマスク解除，レベル7
     st.b    r10,    P11IC0
     mov     r0,     r12   -- LEDフラグ初期化
     ei                    -- 割り込み許可
```

```
_forever:
     nop
     jr     _forever       -- 永久ループ

#    INTP110（INTスイッチ）割り込み処理
_int_p110:
     not    r12,   r12           -- LEDフラグ反転
     mov    r12,   r10           -- テンポラリへcopy
     mov    r10,   r11           -- テンポラリへcopy
     andi   0xf0,  r10,  r10     -- 上位4bit取り出し
     st.b   r10,   P3            -- P3へ出力
     andi   0x0f,  r11,  r11     -- 下位4bit取り出し
     st.b   r11,   PBD           -- PBDへ出力
     reti                        -- 割り込み処理から復帰
```

(3) タイマ機能を使用した表示の更新

(a) 機能概要

タイマC0を使って，内部割り込み要求（INTM000）を発生させ，割り込み処理でLEDのインクリメント表示を行います。

タイマへの設定値は，動作クロック40 MHzを想定した値で，コンペア・レジスタには10 msごとに一致割り込み要求が発生する値をセットしています。

(b) タイマ機能初期化のポイント

タイマ・モード・コントロール・レジスタ10（TMCC10）を例にタイマ機能初期化の例を示します。

- TMCCAE1ビットとその他のビットは同時にはセットできません。必ずTMCCAE1ビットをセットしたあとに，他のビット，およびその他のTMC1ユニットのレジスタを設定してください。
- TMCCE1 = 1のとき，再度，TMCCE1 = 1としても，設定は無効です。タイマを再スタートする場合は，TMCCE1 = 0にしたあと，TMCCE1 = 1としてください。

ビット位置	7	6	5	4	3	2	1	0
ビット名	OVF1	CS12	CS11	CS10	0	0	TMCCE1	TMCCAE1
初期値	0	0	0	0	0	0	0	0
設定値①	0	0	0	0	0	0	0	0
設定値②	0	0	0	0	0	0	0	1
設定値③	0	1	0	1	0	0	0	1
設定値④	0	1	0	1	0	0	1	1

必ず，次の順序で行ってください。

設定値①：リセット（電源立ち上げ初期値）

設定値②：TMCCAE1（クロック供給）開始

設定値③：カウント・クロック選択

設定値④：カウント動作許可（TMCCAE1：0→1）

注意　設定値②と設定値③の同時設定はカウント・クロック選択が無効になります。

(c) タイマ機能初期化のプログラム例

```
#### タイマ機能初期化　####
    set1   0x0,    TMCC10       -- TMC1 ユニットへクロック供給開始（TMCCAE1=1）
    ld.b   TMCC10,    r10
    ori    0x50,   r10,    r10   -- TMC1 内部カウント・クロック fxx/128=3.2us（40MHz）選択
    st.b   r10,    TMCC10                                                    } 初回設定
    <中　略>
    set1   0x1,    TMCC10       -- TMC1 カウント許可
    <タイマ計測処理１>
    ld.h   TMC1,   r15          -- TMC1 カウント読み出し

    clr1   0x1,    TMCC10       -- TMC1 カウント・リセット（TMCCE1=0）   ↓ 注   } 2回目以降設定
    set1   0x1,    TMCC10       -- TMC1 カウント許可（TMCCE1=1）
    <タイマ計測処理２>
    ld.h   TMC1,   r15          -- TMC1 カウント読み出し
```

注　0→1への変化で再起動となります。

(d) 初回設定のN.G.パターン

```
movea     0x51,r0,r10      -- TMC1 ユニットへクロック供給開始（TMCCAE1=1）
st.b      r10, TMCC10      -- TMC1 内部カウント・クロック fxx/128=3.2us（40MHz）選択
set1      0x1, TMCC00      -- TMC1 カウント許可
```

TMCCAE1ビットとその他のビットは同時にセットできません。このため，TMCC10レジスタに51Hを設定すると，TMCCAE1ビットは1になるのですが，TMCC10レジスタのCS12-CS10ビットは初期値（000）のままで，タイマの動作はfxx/4で動作してしまいます。

(e) 再起動のN.G.パターン

```
set1      0x0,     TMCC10     -- TMC1 ユニットへクロック供給開始(CAE1=1)注1
ld.b      TMCC10,r10
ori       0x50,    r10,r10    -- TMC1 内部カウント・クロック fxx/128=3.2us(40MHz)選択
st.b      r10,     TMCC10
ld.h      TMC1,    r15        -- TMC1カウント読み出し
set1      0x1,     TMCC10     -- TMC1 カウント許可
        <時間計測処理1>
ld.h      TMC1,    r15        -- TMC1 カウント読み出し
set1      0x1,     TMCC10     -- TMC1 カウント許可(TMCCE1=1) 注2
        <時間計測処理2>
ld.h      TMC1,    r15        -- TMC1 カウント読み出し
```

注1. 初期値0からの設定なので0→1

2. 2回目の設定なので1→1

このように設定すると，＜時間計測処理2＞のカウント値は，＜時間計測処理1＞からの続行カウントとなってしまいます。

(f) プログラム・リスト

```
# タイマC0を使って10msごとにLEDをインクリメント表示する。
# ＜使用レジスタ＞
#     r10：テンポラリ
#     r11：テンポラリ
#     r12：カウンタ

#外部参照宣言
.extern     _cpu_init
.extern     _bus_init

#リセットおよび割り込み
.section "RESET",text           -- リセット・ハンドラ宣言
     jr     _start              -- サンプル・プログラム・コード部の先頭へジャンプ
.section "INTM000",text         -- INTM000割り込みハンドラ宣言
     jr     _int_tm000          -- タイマC0コンペア一致割り込み処理部の先頭へジャンプ

#プログラム本体
.text
.align 4                        -- 4byteで整列
.globl _start                   -- 外部宣言
```

```
_start:
     jarl   _cpu_init,r31          -- CPU基本機能初期化部
     jarl   _bus_init,r31          -- CPUバス制御機能初期化部

#タイマC0初期化
     set1   0x0, TMCC00            -- TMCCAE0=1,TMC0 ユニット全体へクロック供給開始
     ld.b   TMCC00,r10             --
     ori    0x50,  r10,    r10     -- CS02,CS01,CS00=1,0,1
     st.b   r10,   TMCC00          -- TMC0 内部カウント・クロック fxx/128=3.2us（40MHz）選択

     mov    0x09,  r10             -- 自動クリア&スタート（CCLR0）
     st.b   r10,   TMCC01          -- コンペア・モード選択（CMS00）

     mov    3906,  r10             -- 10ms=3.2us*3125
     st.h   r10,   CCC00

     mov    0x07,  r10             -- コンペア一致割り込みマスク解除,レベル7
     st.b   r10,   P00IC0
     set1   0x1,   TMCC00          -- TMC0 カウント許可（TMCCE0=1）

     st.b   r0,    P3              -- LED上位4bit消灯
     st.b   r0,    PBD             -- LED下位4bit消灯
     mov    r0,    r12             -- カウンタ初期化

     ei                            -- 割り込み許可
_forever:
     nop
     jr     _forever               -- 永久ループ

#    INTM000（10ms） 割り込み処理
_int_tm000:
     addi   1,     r12,    r12     -- カウントup
     mov    r12,   r10             -- カウンタをテンポラリへcopy
     mov    r10,   r11             -- カウンタをテンポラリへcopy
     andi   0xf0,  r10,    r10     -- カウンタ値上位4bit取り出し
     st.b   r10,   P3              -- P3へ出力
     andi   0x0f,  r11,    r11     -- カウンタ値下位4bit取り出し
     st.b   r11,   PBD             -- PBDへ出力
     reti                          -- 割り込み処理から復帰
```

(4) 多重割り込みによるカウンタ表示の制御

(a) 機能概要

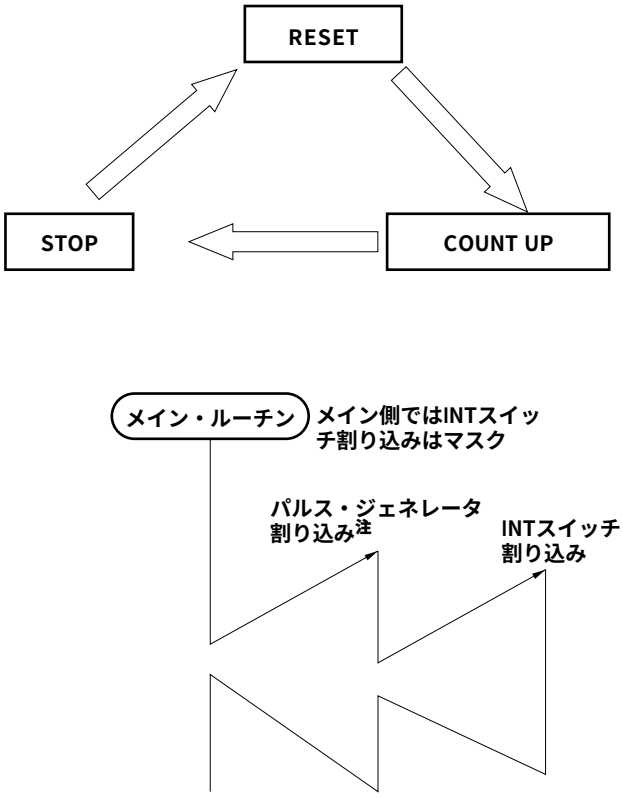
$\overline{\text{INTP111}}$端子に入力されるパルス・ジェネレータからの信号をカウント（カウンタの値は，0-9999の範囲）し，7セグメント LEDに表示します。

INTスイッチを押すとカウント動作モードが3段階に切り替わります。INTスイッチによる割り込み要求は，多重割り込みを確実に発生させる都合上，パルス・ジェネレータ割り込みの中でのみ許可（マスク解除）しています。

ディバッカを起動して実行させると，7セグメントLEDカウンタが“ 0000 ”の表示で停止しています（RESET状態）。RESET状態で$\overline{\text{INTP110}}$信号が発生し，カウントを開始します（COUNT UP状態）。次にINTスイッチが押されると，カウントが停止しカウント値を保持した状態になります（STOP状態）。さらにINTスイッチが押されるとカウント値が“ 0000 ”になります（RESET状態）。

INTスイッチが押され，$\overline{\text{INTP110}}$が発生するたびに，矢印の方向に遷移します。

図4－18　多重割り込みによるカウンタ表示の制御



注　パルス・ジェネレータ割り込み処理ルーチン内でのみ，INTスイッチ割り込みのマスクが解除できます。

備考　優先順位：パルス・ジェネレータ割り込み＜INTスイッチ割り込み

(b) プログラム・リスト

```
# パルス・ジェネレータ割り込みを使い，7SEG LEDに0-9999までの数値をインクリメント
# 表示する。INTスイッチを押すとモードが3段階に切り替わる。
# INTスイッチはパルス・ジェネレータ割り込みより優先順位が高く，
# パルス・ジェネレータ割り込み内でのみ，INTスイッチ割り込みを許可している。
#
# MODE 0:カウンタリセット
# MODE 1:カウンタ続行
# MODE 2:カウンタ停止
# ＜使用レジスタ＞
#     r10:テンポラリ
#     r11:カウンタ・モード
#     r12:表示要求フラグ
#     r13:カウンタ
#     r14:ダミー・カウンタ

#外部参照宣言
.extern  _cpu_init
.extern  _bus_init
.extern  _disp_num

#リセットおよび割り込み
.section "RESET",text           -- リセット・ハンドラ宣言
     jr      _start             -- サンプル・プログラム・コード部の先頭へジャンプ
.section "INTP110",text         -- INTP110割り込みハンドラ宣言
     jr      _int_p110          -- INTSW割り込み処理部の先頭へジャンプ
.section "INTP111",text         -- INTP111割り込みハンドラ宣言
     jr      _int_p111          -- パルス・ジェネレータ割り込み処理部の先頭へジャンプ

#プログラム本体
.text
.align 4                        -- 4byteで整列
.globl _start                   -- 外部宣言

_start:
     jarl    _cpu_init,r31      -- CPU基本機能初期化部
     jarl    _bus_init,r31      -- CPUバス制御機能初期化部
     mov     0xfffc200, r3      -- Stack Pointer設定（内蔵RAM先頭+0x200）
     st.b    r0,     P3
     st.b    r0,     PBD
```

```
    mov    0x04,  r10          -- INTP110立ち下りエッジ,INTP111立ち上がりエッジ
    st.b   r10,   INTM2
    mov    0x46,  r10          -- INTスイッチ割り込み（INTP110）
    st.b   r10,   P11IC0       -- 割り込みマスク,優先順位6
    mov    0x07,  r10          -- パルス・ジェネレータ割り込み（INTP111）
    st.b   r10,   P11IC1       -- 割り込みマスク解除,優先順位7
    mov    r0,    r11          -- カウンタ・モード初期化
    mov    r0,    r12          -- 表示要求なし
    mov    r0,    r13          -- カウンタ初期化
    ei
_loop:
    cmp    r0,    r12          -- 表示要求ON？
    be     _loop               -- OFFなら何もしない
    mov    r0,    r12          -- ONなら表示要求OFFに設定
    mov    r13,   r6           -- カウンタ値を引数に渡す
    jarl   _num_disp,r31       -- 表示処理部を呼ぶ
    br     _loop

#################################
#    INTP110（INT-SW）割り込み処理
#################################
_int_p110:
    -- レジスタ退避
    add    -4,    r3
    .option nowarning
    st.w   r1,    0[r3]
    .option warning

    cmp    2,     r11          -- カウンタ・モード2？
    bz     _int_p110_2
    -- カウンタ・モード2以外
    add    1,     r11          -- カウンタ・モード+1
    br     _int_p110_ext
_int_p110_2:
    -- カウンタ・モード2
    mov    r0,    r11          -- カウンタ・モード0に初期化
_int_p110_ext:
    -- レジスタ復帰
    .option nowarning
    ld.w   0[r3],  r1
    .option warning
    add    4,     r3
    reti
```

```
#################################
#  INTP111（Pulse-Gen）割り込み処理
#################################
_int_p111:
     -- EIPC,EIPSWレジスタ退避
     add    -4,    r3
     .option nowarning
     st.w   r1,    0[r3]
     .optionwarning
     add    -8,    r3
     .optionnowarning
     stsr   0,     r1
     st.w   r1,    4[r3]
     stsr   1,     r1
     st.w   r1,    0[r3]
     .optionwarning

     clr1   6,     P11IC0        -- INTP110（INTSW）割り込みマスク解除
     ei                          -- 多重割り込み許可
#### ここからINTSW割り込み可能 ####
     cmp    r0,    r11           -- カウンタ・モード0 ?
     bne    _int_p111_1

     -- カウンタ・モード0（カウンタ"0000"リセット表示）
     mov    r0,    r13           -- カウンタ初期化
     movea  0xff,  r0,    r12    -- 表示要求ON
     br     _int_ret

     -- カウンタ・モード1（0-99999までカウント・アップ表示）
_int_p111_1:
     cmp    1,     r11           -- カウンタ・モード1 ?
     bne    _int_ret
     add    1,     r13           -- カウンタ+1
     cmp    99999, r13
     bnh    _int_p111_nc        -- カウンタ上限値越えた？
     mov    r0,    r13           -- カウンタ初期化
_int_p111_nc:
     movea  0xff,  r0,    r12    -- 表示要求ON

     -- カウンタ・モード2（カウンタ停止）
_int_ret:
     di
     set1   6,     P11IC0        -- INTP110（INTSW）割り込みマスク
#### ここまでINTSW割り込み可能 ####
```

```
     -- EIPC,EIPSWレジスタ復帰
     .optionnowarning
     ld.w   0[r3], r1
     ldsr   r1,    1
     ld.w   4[r3], r1
     ldsr   r1,    0
     .optionwarning
     add    8,     r3
     .optionnowarning
     ld.w   0[r3], r1
     .optionwarning
     add    4,     r3
     reti

# numdisp.s
#  バイナリ・データを10進変換し7セグメントLEDに表示
#   r6: 被変換2進数（引数）
#  r20: 10で割った商（div10）
#  r21: 10で割った余り（div10）
#  r22: 表示バッファ・アドレス
#  r23: 数字パターン・テーブル先頭アドレス
#  r24: ワーク
#  r25: div10処理呼び出し時の戻り先アドレス格納
#    （7seg LED先頭アドレス）
#
#  変換結果の下位5桁を表示します
#
.set        DISPBUFSIZE, 10
.bss
.lcomm              _disp_buf, DISPBUFSIZE, 4

.text
.globl              _num_disp
_num_disp:
# レジスタ退避（r1,r31）
     addi   -8,    r3,    r3
     .option nowarning
     st.w   r1,    0[r3]
     .option warning
     st.w   r31,   4[r3]
```

```
# 表示バッファ0クリア
_bindec:
     mov    #_disp_buf,   r22    -- 表示バッファ先頭アドレス設定
     mov    10,    r24           -- 0クリア・ループ回数設定
_bindec1:
     st.b   r0,    0[r22]        -- 0クリア
     add    1,     r22           -- アドレス・インクリメント
     add    -1,    r24           -- カウンタ・デクリメント
     bnz    _bindec1             -- 10回終了?

# 2進10進変換
     mov    #_disp_buf+9, r22    -- 表示バッファ末尾アドレス設定
_bindec2:
     and    r6,    r6            -- 割られる数は0 ?
     bz     _bindec3             -- 割られる数0ならば除算処理中止
     jarl   _div10,r31           -- 10で割る処理を呼び出す
     st.b   r21,   0[r22]        -- 商を表示バッファに格納
     add    -1,    r22           -- 表示バッファ・デクリメント
     mov    r20,   r6            -- 余りを割られる数に再設定
     br     _bindec2

# 7セグメントLEDへ5桁の表示
_bindec3:
     movhi  0x0fe0,r0,    r25    -- 7seg LED register先頭アドレス設定
     mov    #_num_data,   r23    -- 数字パターン・テーブル先頭アドレス設定
     mov    #_disp_buf,   r22    -- 表示バッファ先頭アドレス設定

     ld.b   5[r22],r20           -- 1万の位10進数読み出し
     add    r23,   r20           -- テーブル・アドレス・オフセット加算
     ld.b   0[r20],r24           -- 数字パターン読み出し
     st.b   r24,   8[r25]        -- 7seg LED 5桁目へ出力

     ld.b   6[r22],r20           -- 千の位10進数読み出し
     add    r23,   r20           -- テーブル・アドレス・オフセット加算
     ld.b   0[r20],r24           -- 数字パターン読み出し
     st.b   r24,   6[r25]        -- 7seg LED 4桁目へ出力
     ld.b   7[r22],r20           -- 百の位10進数読み出し
     add    r23,   r20           -- テーブル・アドレス・オフセット加算
     ld.b   0[r20],r24           -- 数字パターン読み出し
     st.b   r24,   4[r25]        -- 7seg LED 3桁目へ出力

     ld.b   8[r22],r20           -- 十の位10進数読み出し
     add    r23,   r20           -- テーブル・アドレス・オフセット加算
```

```
      ld.b    0[r20],r24          -- 数字パターン読み出し
      st.b    r24,    2[r25]      -- 7seg LED 2桁目へ出力

      ld.b    9[r22],r20          -- 一の位10進数読み出し
      add     r23,    r20         -- テーブル・アドレス・オフセット加算
      ld.b    0[r20],r24          -- 数字パターン読み出し
      st.b    r24,    0[r25]      -- 7seg LED 1桁目へ出力

# 使用レジスタ復帰（r1,r31）
      ld.w    4[r3], r31
      .option nowarning
      ld.w    0[r3], r1
      .option warning
      addi    8,      r3,     r3
      jmp     [r31]

#
# 10で割る処理
#
#  r6: 被除数（div10）
# r10: 10で割った商（div10）
# r11: 10で割った余り（div10）

_div10:
      mov     r0,     r20         -- 商を0に初期化
_div10_1:
      cmp     10,     r6          -- 割られる数は10未満 ?
      bn      _div10_2            -- 10未満なら減算を終了
      add     -10,    r6          -- 割られる数から10を減算
      add     1,      r20         -- 減算回数をカウント・アップ（商）
      br      _div10_1
_div10_2:
      mov     r6,     r21         -- 減算で最後に残った数（余り）
      jmp     [r31]               -- 呼び出し元へ戻る

# 数字パターン・テーブル
.data
_num_data:
#           "0", "1", "2", "3", "4"
      .byte   0x3f,0x06,0x5b,0x4f,0x66
#           "5", "6", "7", "8", "9"
      .byte   0x6d,0x7d,0x27,0x7f,0x67
```

(5) DMA機能によるメモリ間のデータ転送

(a) 機能概要

DMAチャネル0を使い，SRAMからEDO DRAMへブロック転送モードによる2サイクル転送を行います。SRAMの先頭アドレス（0x200000番地）からEDO DRAMの先頭アドレス（0x4000000番地）へハーフワード×256回，512バイト分のDMA転送を行います。DMA転送終了割り込みで，8個のLEDを全点灯しています。

(b) プログラム・リスト

```
# DMAのブロック2サイクル転送を使って，SRAM領域のデータを
# EDO DRAM領域へ16bit×256回（512byte）転送する。
# ＜使用レジスタ＞
#     r10：テンポラリ
#     r11：テンポラリ
#     r12：テンポラリ

#外部参照宣言
.extern     _cpu_init
.extern     _bus_init

#定数定義
.set EDO DRAM,      0x04000000    -- EDO DRAMアドレス定義
.set SRAM,          0x00200000    -- SRAM アドレス定義

#リセットおよび割り込み
.section "RESET",text             -- リセット・ハンドラ宣言
     jr     _start                -- サンプル・プログラム・コード部の先頭へジャンプ
.section "INTDMA0",text           -- INTDMA0割り込みハンドラ宣言
     jr     _int_dma0             -- DMA0転送終了割り込み処理部の先頭へジャンプ

#プログラム本体
.text
.align 4                          -- 4byteで整列
.globl _start                     -- 外部宣言

_start:
     jarl   _cpu_init,r31         -- CPU基本機能初期化部
     jarl   _bus_init,r31         -- CPUバス制御機能初期化部

     st.b   r0,     P3            -- LED上位4bit消灯
     st.b   r0,     PBD           -- LED下位4bit消灯
```

```
#DMA0初期化
     mov    SRAM,  r10          -- 転送元アドレス設定
     mov    r10,   r11          -- 転送元アドレスcopy
     shr    16,    r10          -- DMA0 転送元アドレス上位12bit側
     st.h   r10,   DSA0H
     andi   0xffff,r11,   r11   -- DMA0 転送元アドレス下位16bit側
     st.h   r11,   DSA0L

     mov    EDO DRAM,r10        -- 転送先アドレス設定
     mov    r10,   r11          -- 転送先アドレスcopy
     shr    16,    r10          -- DMA0 転送先アドレス上位12bit側
     st.h   r10,   DDA0H
     andi   0xffff,r11,   r11   -- DMA0 転送先アドレス下位16bit側
     st.h   r11,   DDA0L

     mov    0xff,  r10          -- DMA転送回数設定（256回）
     st.h   r10,   DBC0

     mov    0x400c,r10          -- DMA0 転送モード設定
                                -- 16bit data,転送元++／転送先++,
     st.h   r10,   DADC0        -- ブロック転送モード，2サイクル転送

     mov    0x07,  r10          -- マスク解除，レベル7
     st.b   r10,   DMAIC0       -- DMA0転送終了割り込み制御レジスタ設定

     ei                         -- 割り込み許可

     st.b   r0,    DTFR0        -- 内蔵周辺I/OからのDMA要求禁止

#    ld.b   DCHC0, r10          -- TC0 読み捨て（複数回起動する時に必要）

     mov    0x03,  r10          -- DMA 転送許可（E00=1）と開始（STG0=1）
     st.b   r10,   DCHC0
_forever:
     nop
     jr     _forever            -- 永久ループ

#    INTDMA0 割り込み処理
_int_dma0:
     mov    0xf0,  r12
     st.b   r12,   P3           -- LED上位4bit点灯
     mov    0x0f,  r12
     st.b   r12,   PBD          -- LED下位4bit点灯
reti
```

(6) タイマ機能を使用したパルス周波数の測定

(a) 機能概要

1秒間に入力されるパルスの数をカウントし，表示する処理を繰り返します。タイマへの設定値は，動作クロック50 MHzを想定した値で，コンペア・レジスタには，10 msごとに一致割り込み要求が発生する値をセットしています。このタイマ割り込みが100回発生したところで，前回のタイマ割り込み以降にカウントしたパルス数を表示する準備をします。カウントの表示は，メイン・プログラムで行っています。

(b) プログラム・リスト

```
/**********************************/
/*    パルスの周波数を表示する          */
/**********************************/

#include    "common.h"

void int_tm000（void）;            /* 10ms interval 割り込み */
void int_p111（void）;             /* Pulse generator 割り込み */

unsigned    int    counter,freq ;
unsigned    char   times,disp_req ;

main ()
{
     init_pmode () ;
     TMCC00.0 =     1;             /* TMC0 reset */
     TMCC00 |=      0x50;          /* TMC0 fxx/128（t=2.56us） 50MHz時 */
     TMCC01 =       0x09;          /* auto clear&start , CCC00 is compare */
     CCC00 =        3906;          /* for 10ms */
     TMCC00.1 =     1;             /* TMC0 count enable */

     P3 = 0x00;                    /* LED turn off */
     PBD = 0x00;

     INTM2=0x04;                   /* INTP111（Pos.Edg） */
     P00IC0=0x07;                  /* mask off, priority 7 */
     P11IC1=0x07;                  /* mask off, priority 7 */
     times = 0;
     disp_req = OFF;
     counter = 0;
     __EI () ;                     /* enable interrupt */
```

```
    while (1)
    {
        if (disp_req == ON)
        {
            disp_req = OFF;
            disp_num (freq) ;
        }
    }
}
/***********************************/
/* INTP111 (Pulse-Gen) 割り込み処理   */
/***********************************/
#pragma interrupt INTP111 int_p111
__interrupt
void int_p111 (void)
{
    if (counter<99999)
        counter++;
}
/***********************************/
/*  INTM000 (10ms)  割り込み処理      */
/***********************************/
#pragma interrupt INTP000 int_tm000
__interrupt
void int_tm000 (void)
{
    times++;
    if (times==100)
    {
        times=0;
        freq = counter;
        counter = 0;
        disp_req = ON;
    }
}
```

（7）A/D, D/A変換機能を使用した正弦波の入出力

（a）機能概要

125 μs間隔で，アナログ入力端子に入力される正弦波をサンプリングし，A/D変換を起動します。A/D変換終了の割り込み処理で，変換結果をR-2R（D/A変換）回路に接続されたスピーカより出力します。

（b）プログラム・リスト

```
/**********************************/
/* 125usインターバルで正弦波入力        */
/* をA/D変換し結果をR-2R（D/A）         */
/* スピーカより出力する                 */
/**********************************/

#include    "common.h"

void int_tm000（void）;
void int_ad（void）;

main（）
{

     TMCC00.0 = 1;         /* TMC0 reset */
     TMCC00 |= 0x10;       /* TMC0 fxx/8（t=0.16us） 50MHz時 */
     TMCC01 = 0x09;        /* auto clear&start , CCC00 is compare */
     CCC00 = 781;          /* for 125us */
     TMCC00 = 0x13;        /* TMC0 count enable */
     TMCC00.1 = 1;         /* TMC0 count enable */
     P00IC0 = 0x07;        /* INTTM000 mask off */

     ADM2 = 0x00;          /* ADC reset */
     ADM2 |= 0x01;         /* ADC enable */
     ADM0 = 0x10;          /* ANI0 sel.1buf */
     ADM1 = 0x03;          /* A/D trg. */
     ADIC = 0x07;          /* INTAD mask off */
     __EI（）;             /* enable interrupt */
     while（1）
     {
            ;              /* dummy statement */
     }
}
```

```
/**********************************/
/*  INTM000（125us） 割り込み処理      */
/**********************************/
#pragma interrupt INTM000 int_tm000
__interrupt
void int_tm000（void）
{
     ADM0 |= 0x80;  /* Converter enable */
}

/**********************************/
/*   INTAD 割り込み処理                */
/**********************************/
#pragma interrupt INTAD int_ad
__interrupt
void int_ad（void）
{
     unsigned char result;
     result = ADCR0H;
     R_2R_DA = result;
     P3  = result & 0xf0;  /* LED7-4 */
     PBD = result & 0x0f;  /* LED3-0 */
}
```

(8) PWMによるDCモータの速度制御

(a) 機能概要

スイッチで設定された値をPWMのデューティ比とした速度制御を行います。

このプログラムでは，スイッチの状態が変化したとき，PWMへの設定を変更するとともに，LEDにその値を表示しています。

(b) プログラム・リスト

```
/**********************************/
/* Switch to PWM0 & LED           */
/* スイッチを監視し，変化したとき       */
/* スイッチの状態をＰＷＭに反映する     */
/**********************************/

#include    "common.h"

void init_pmode（void）;

main（）
{
     unsigned char sw_state;
     unsigned char data_1,data_2,data_3;

     PWMC0 = 0x40;
     PWMC0 |= 0x80;
     data_1 = P0 & 0xfc;          /* スイッチ入力 */
     data_2 = P5 & 0x03;
     sw_state = data_1 | data_2;
     PWMB0 = （unsigned short）sw_state;
     P3 = sw_state & 0xf0;
     PBD = sw_state & 0x0f;
     while （1）
     {
            data_1 = P0 & 0xfc;  /* スイッチ入力 */
            data_2 = P5 & 0x03;
            data_3 = data_1 | data_2;
            if （ data_3 != sw_state ）
            {
                   sw_state = data_3;
                   PWMB0 = （unsigned short）sw_state;
                   P3 = sw_state & 0xf0;
                   PBD = sw_state & 0x0f;
            }
     }
}
```

(9) CSIによるEEPROM書き込み／読み出し処理

(a) 機能概要

CSI2に接続されたEEPROMに対し，CSI機能を使った書き込みと読み出しを行います。16ワード分の書き込みのあと，そのデータの読み出しを行い，読み出したデータを7セグメントLEDに表示します。

(b) プログラム・リスト

```
/**********************************/
/*  CSI機能を使った                    */
/*  シリアルEEPROMインタフェース       */
/**********************************/

#include    "common.h"

#define     EWEN    0x98 /* erase/write enable  */
#define     ERAL    0x90 /* erase all           */
#define     EWDS    0x80 /* erase/write disable */

void                int_csi2 (void) ;
void                int_tm000 (void) ;
void                init_pmode (void) ;
void                wait_ms ( unsigned short ) ;
void                e2_write  ( unsigned char, unsigned short ) ;
unsigned   short    e2_read  ( unsigned char ) ;
void                e2_com  ( unsigned char ) ;
unsigned   char     e2_interact  ( unsigned char ) ;

unsigned   char     flag_csi, int_ctr;
unsigned   short    s_timer;

main ()
{
     unsigned   char     i;
     unsigned   short    r_data[16];
     unsigned   short    w_data[16]=
            {       0x0123,0x1234,0x2345,0x3456,0x4567,0x5678,0x6789,0x789a,
                    0x89ab,0x9abc,0xabcd,0xbcde,0xcdef,0xdef0,0xef01,0xf012  };
```

```
    TMCC00.0 = 1;               /* TMC0 reset */
    TMCC00 |= 0x30;             /* TMC0 fxx/32 (t=0.64us)  50MHz時 */
    TMCC01 = 0x01;              /* compare clear disable, CCC00 is compare */
    CCC00 = 1562;               /* for 1ms interval */
    TMCC00.0 = 1;               /* TMC0 count enable */
    CSIM2 = 0xc0;
    CSIC2 = 0x05;
    CSIIC2 = 0x07;
    P00IC0 = 0x07;
    __EI();                     /* enable interrupt */

    e2_com ( EWEN );            /* erase/write enable */

/* 書き込み */
    for ( i=0; i<0x10; i++ )
    {
        e2_write ( i+0x10, w_data[i] );
        wait_ms ( 10 );
    }

/* 読み出し */
    for ( i=0; i<0x10; i++ )
    {
        r_data[i] = e2_read ( i+0x10 );
    }

/* 読み出しデータ表示 */
    while(1)
    {
        for (i=0; i<0x10; i++)
        {
            disp_hex ((unsigned int) r_data[i]) ;
            wait_ms ( 1000 ) ;
        }
    }
}
/*******************************/
/*   EEPROM command output     */
/*******************************/
void e2_com ( unsigned char command )
{
    P3 |= 0x08;                         /* chip select ON */
    wait_ms ( 2 ) ;
```

```
    e2_interact ( command ) ;
    e2_interact ( 0x00 ) ;
    wait_ms ( 2 ) ;
    P3 &= 0xf7;                          /* chip select OFF */
}

/*********************************/
/*   EEPROM write                */
/*********************************/
void e2_write  ( unsigned char adrs, unsigned short data )
{
    P3 |= 0x08;                                /* chip select ON */
    wait_ms ( 2 ) ;
    e2_interact ( (adrs>>1) | 0xa0 ) ;
    e2_interact ( adrs<<7 ) ;
    e2_interact ( (unsigned char) (data>>8) ) ;
    e2_interact ( (unsigned char) data ) ;
    wait_ms ( 2 ) ;
    P3 &= 0xf7;                                /* chip select OFF */
}

/*********************************/
/*   EEPROM read                 */
/*********************************/
unsigned short e2_read  ( unsigned char adrs )
{
    unsigned short data, d1, d2, d3;

    P3 |= 0x08;                                /* chip select ON */
    wait_ms ( 2 ) ;
    e2_interact ( (unsigned char) ( (adrs>>1) | 0xc0) ) ;
    d1 = e2_interact ( (unsigned char) (adrs<<7) ) ;
    d2 = e2_interact ( 0x00 ) ;
    d3 = e2_interact ( 0x00 ) ;
    data = ( d1<<10) | (d2<<2) | (d3>>6 ) ;
    wait_ms ( 2 ) ;
    P3 &= 0xf7;                                /* chip select OFF */
    return data;
}

/*********************************/
/*   １バイト送受信処理          */
/*********************************/
```

```
unsigned char e2_interact （ unsigned char com ）
{
     while （ CSIM2 & 0x01 ） ;           /* wait end of transimission */
     flag_csi = ON;
     SOTB2 = com;
     while （ flag_csi ） ;               /* wait end of transimission */
     return SIO2;
}

/************************************/
/*   インターバル（ms）処理              */
/************************************/
void wait_ms ( unsigned short work )
{
     s_timer =  work;
     while （ s_timer ）
     {
          ;                      /* dummy statement */
     }
}

/************************************/
/*   INTCSI2 割り込み処理               */
/************************************/
#pragma interrupt INTCSI2 int_csi2
__interrupt
void int_csi2（void）
{
     flag_csi = OFF;
}
/************************************/
/*  INTM000（1msec）  割り込み処理     */
/************************************/
#pragma interrupt INTM000 int_tm000
__interrupt
void int_tm000（void）
{
     CCC00 += 1562;              /* for 1ms interval */
     if （s_timer !=0 ）
     {
          s_timer--;
     }
}
```

(10) 光センサの入力変化検出

(a) 機能概要

100 ms間隔で，光センサ入力をA/D変換し，結果を7セグメントLEDに表示するとともに，前回の結果と値が3以上離れていれば警報を鳴らします。

(b) プログラム・リスト

```
/**********************************/
/*  光センサのテスト                */
/* 100msec間隔で光センサ入力を      */
/* A/D変換し                       */
/* ①結果を7セグLEDに表示する       */
/* ②前回と2以上離れていたら警報     */
/**********************************/

#include    "common.h"

void        int_tm000(void);
void        int_ad(void);
unsigned char int_ctr, prev;
unsigned short alarm=0;
unsigned char result;

main()
{

     PWMC0 = 0x40;
     PWMC0 |= 0x80;

     TMCC00.0 = 1;        /* TMC0 reset */
     TMCC00 |= 0x30;      /* TMC0 fxx/32(t=0.64us) 50MHz時 */
     TMCC01 = 0x09;       /* auto clear&start , CCC00 is compare */
     CCC00 = 1562;        /* for 1msec */
     TMCC00.1 = 1;        /* TMC0 count enable */
     P00IC0 = 0x07;       /* INTM000 mask off */

     ADM2 = 0x00;         /* ADC reset */
     ADM2 |= 0x01;        /* ADC enable */
     ADM0 = 0x12;         /* ANI2 sel.1buf */
     ADM1 = 0x03;         /* A/D trg. */
     ADIC = 0x07;         /* INTAD mask off */
```

```
    int_ctr=0;
    __EI() ;            /* enable interrupt */

    while (1)
    {
        ;               /* dummy statement */
    }
}

/***********************************/
/*  INTM000（1msec）　割り込み処理     */
/***********************************/
#pragma interrupt INTM000 int_tm000
__interrupt
void    int_tm000 (void)
{
        if (alarm !=0)
        {
                alarm--;
                if (alarm & 0x0001)
                {
                        R_2R_DA=0xC0;
                }else
                {
                        R_2R_DA=0x40;
                }
        }
        int_ctr++;
        if (int_ctr==100)
        {
                int_ctr = 0;
                ADM0 |= 0x80; /* Converter enable */
        }
}
```

```
/**********************************/
/*   INTAD 割り込み処理                */
/**********************************/
#pragma interrupt INTAD int_ad
__interrupt
void int_ad (void)
{
      result = ADCR2H;
      P3  = result & 0xf0;       /* LED7-4 */
      PBD = result & 0x0f;       /* LED3-0 */
      disp_num ( (unsigned int) result) ;
      if  (result>prev+2 || result<prev-2)
      {
             alarm=250;
      }
      prev=result;
      PWMB0 =  (unsigned short) result;
}
```

(11) 複数チャネルの音楽演奏

(a) 機能概要

楽譜データを読み出しながら，スピーカから順次音階を出力する処理を，3チャネル分並行して行うプログラムです。D/A変換回路に接続されたスピーカには，各チャネル出力の和が出力されるようにプログラム制御しています。各チャネルは独立して動作するように作られていますが，このプログラムでは，同じ楽譜の演奏を，ディレイを持たせてスタートすることにより，エコーを効かせた演奏を実現しています。

(b) プログラム処理

ひとつの音符は「音階」と「長さ」で構成し，楽譜テーブルに演奏順に並べておきます。

音階コードは0-35の36種類で，3オクターブ分の音階を表現します。音階コード255は休符を，254は楽譜テーブルの終端を示します。

長さコードは四分音符の長さを8で表し，それぞれの音符に見合った値を設定します。

音階ごとに，該当周波数を発生させるための出力反転間隔と，その音階を32分音符分の長さを出力させるのに必要な反転回数（単位カウント）がテーブル形式で用意されています。

音階の出力は，その音階に対応する出力反転間隔を用いてコンペア・レジスタを更新しながら，一致割り込みが発生するたびにD/A変換回路へ出力するデータを反転します。ひとつの音符の演奏は，この操作を，長さコード×単位カウントで算出される回数分繰り返すことにより実現します。

それぞれの音をはっきり刻むために，このプログラムでは音符と音符の間に音を出さない「間（ま）」を挿入しています。ひとつの音符の出力が終わったあと，ch_ma［n］をオンにして，この「間」を作り出しています。「間」の出力が終了したあと，次の音符処理に移ります。

楽譜の先頭から順に，ひとつずつ音符を読み出して演奏を続け，音階コードが254（終端）の音符を読み出したところで演奏を終了します。

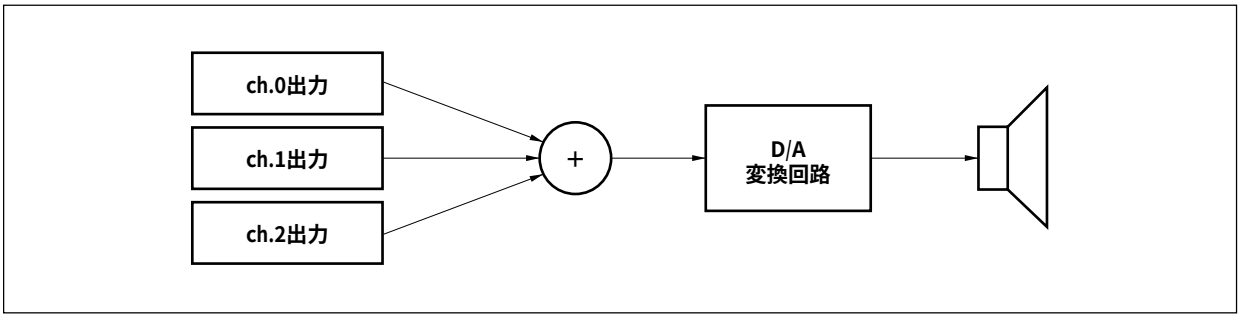
(c) 使用する割り込み

- INTM000：ch.0の音階を生成するためのインターバル・タイマ割り込み
- INTM001：ch.1の音階を生成するためのインターバル・タイマ割り込み
- INTM010：ch.2の音階を生成するためのインターバル・タイマ割り込み
- INTM011：1 msインターバル・タイマ割り込み

(d) 複数チャネル出力の合成

複数チャネルの出力を，ひとつのスピーカから出力するため，次の方法で合成します。

図4－19　複数チャネル出力の合成



D/A変換回路の分解能は0-FFHの256段階であり，3チャネルの出力を合成したときの振幅も，この範囲におさまるようにします。

各チャネルのD/A変換回路への出力は80Hを中心に図4－20のようにプラス方向／マイナス方向へ交互に振動するものとし，チャネル0は±30H，チャネル1は±20H，チャネル2は±10Hの振幅で合成されるものとします。したがって出力は20H-E0Hの範囲で動作することになります。

図4－20　D/A変換回路への出力（1/2）

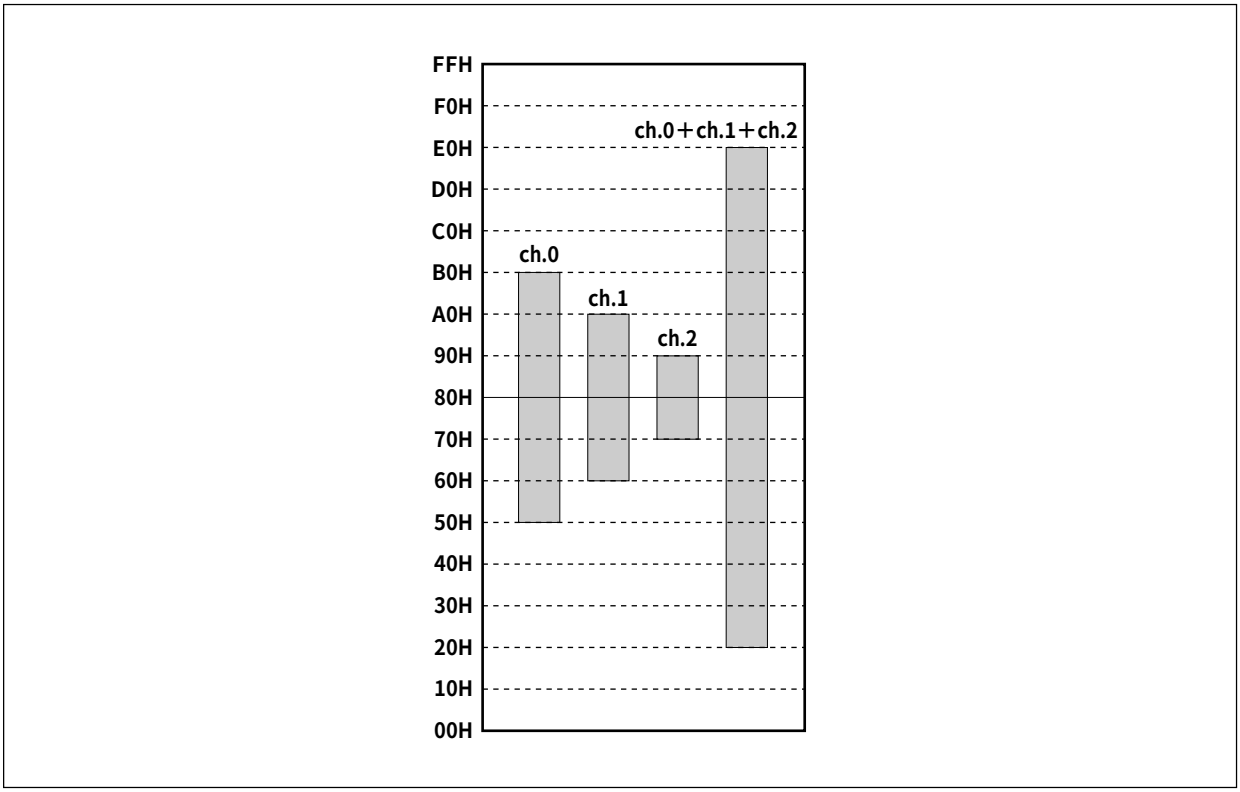
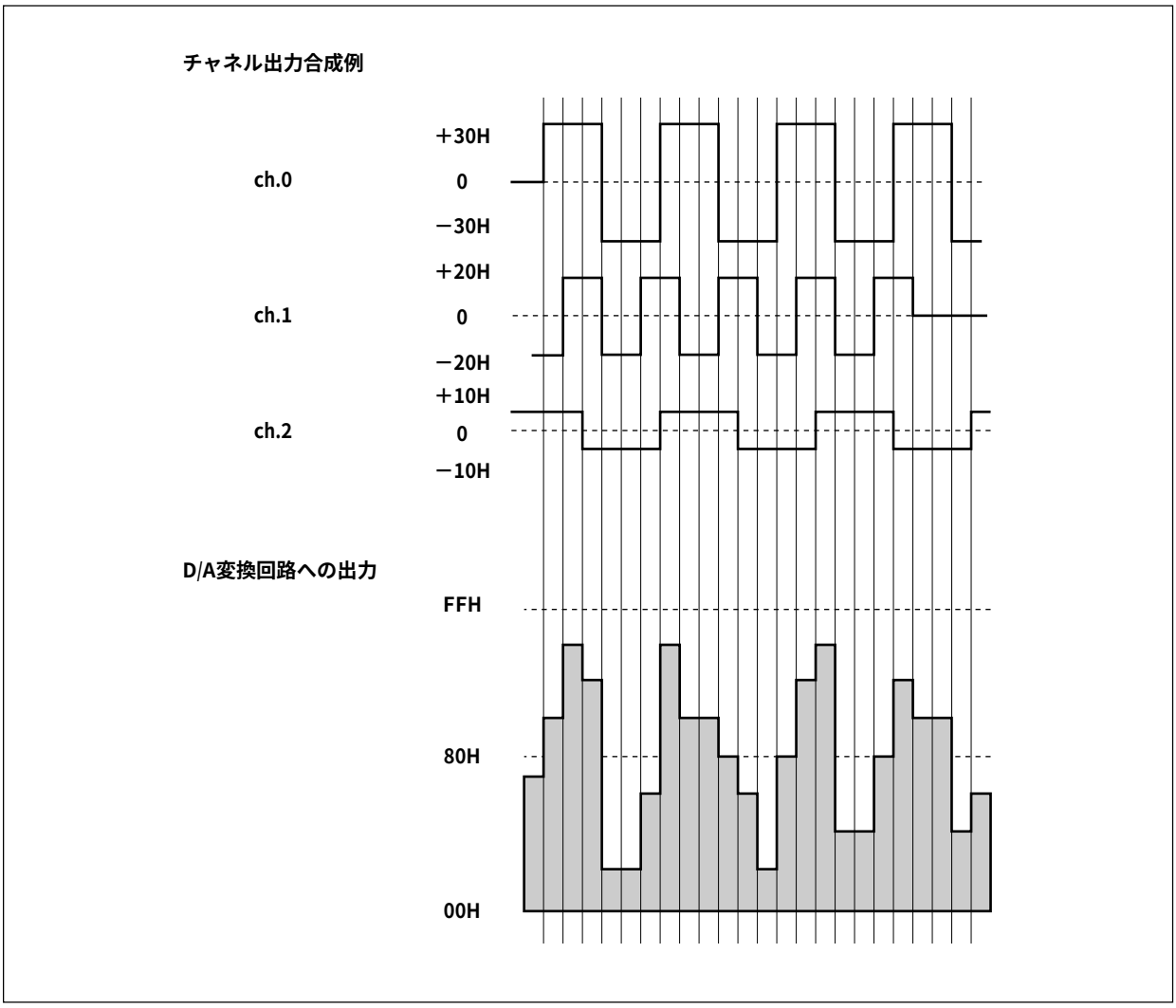


図4－20　D/A変換回路への出力（2/2）



（e）music_start関数

チャネル番号と，楽譜データの先頭アドレスを与え，演奏を開始させます。

music_start関数で演奏をスタートさせると，指定したチャネルに対応する音階生成用のインターバル・タイマ割り込みが許可状態になり，演奏が終了すると禁止状態になります。この状態をチェックすることで演奏中か否かの識別が可能です。

（f）mix_da関数

ch_val[0]，ch_val[1]，ch_val[2]の各領域に格納されている符号付きの値を合成して，D/A変換回路へ出力します。出力する値は80Hを中心にした00H-FFHの値となります。

（G）フロー・チャート

図4－21　複数チャネルの音楽演奏（メイン関数）

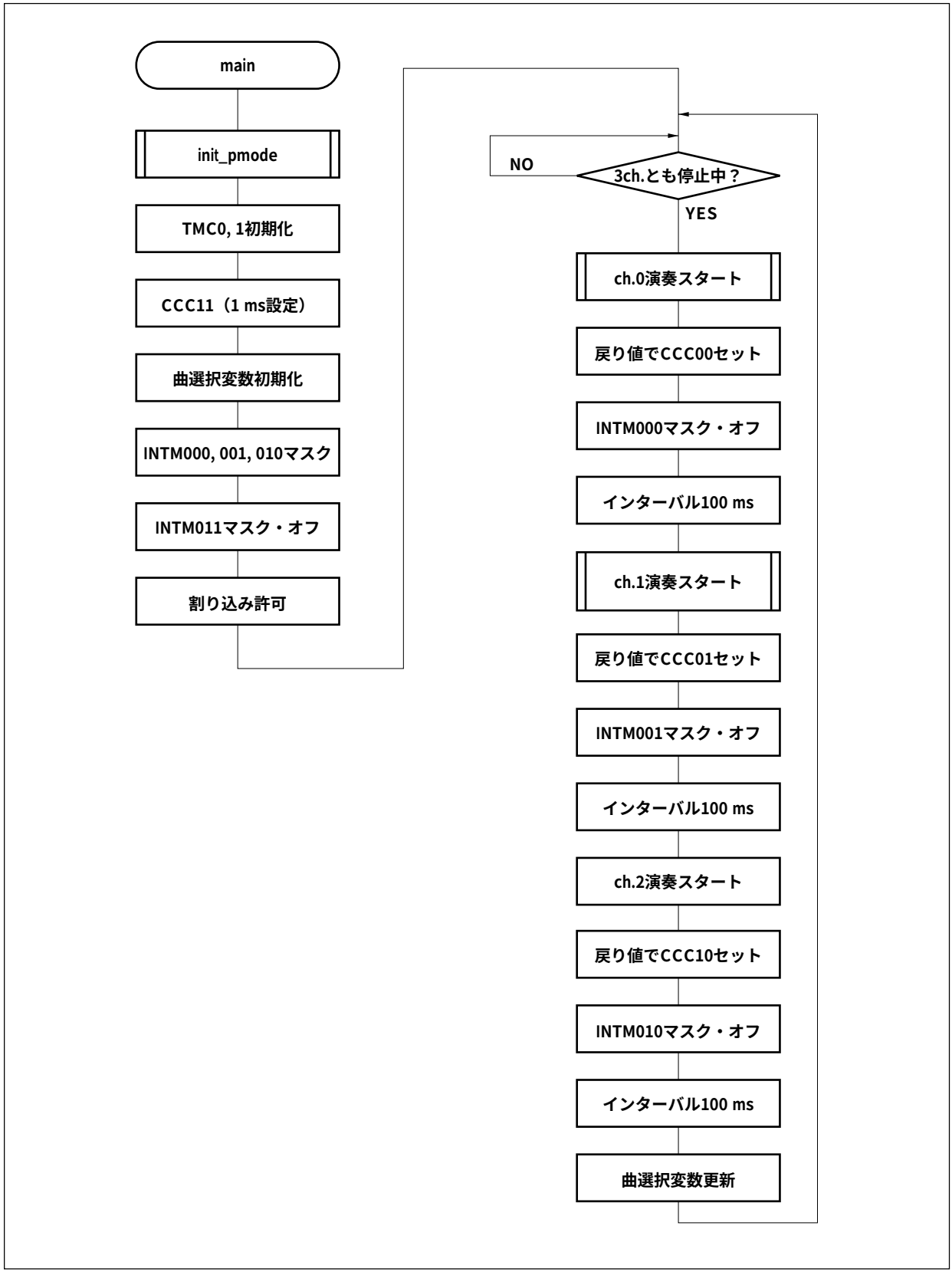
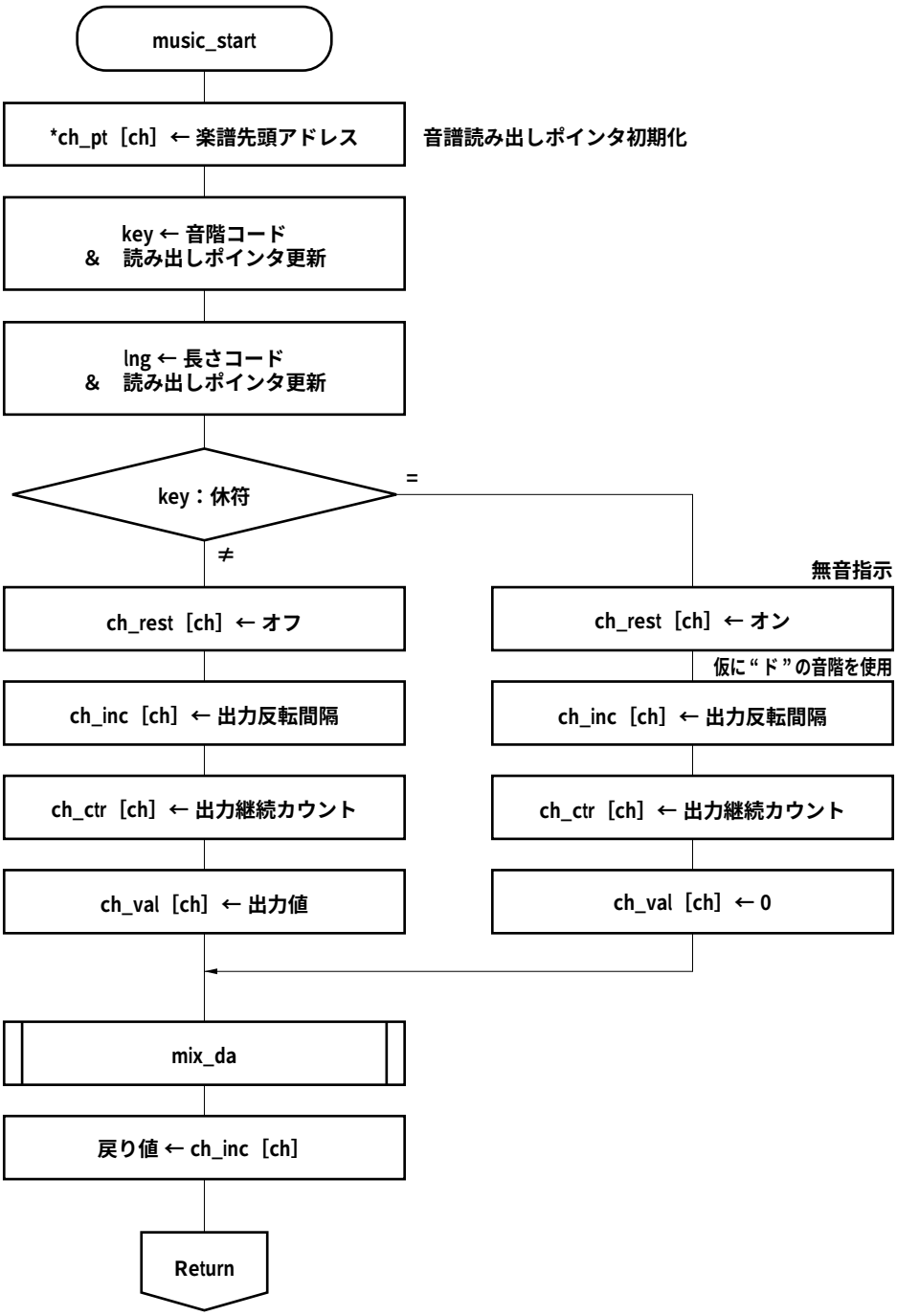


図4－22　複数チャネルの音楽演奏（楽譜読み出しポインタ初期化）



備考1.　出力反転間隔，出力継続カウントは，読み出した音階コードに応じた値をfreq_tabl[ ][ ]より取り出し確定します。

2.　割り込み用途は次のとおりです。

- INTM000：ch.0音階インターバル
- INTM001：ch.1音階インターバル
- INTM010：ch.2音階インターバル
- INTM011：1 msインターバル

図4－23　複数チャネルの音楽演奏（D/A変換回路への出力）

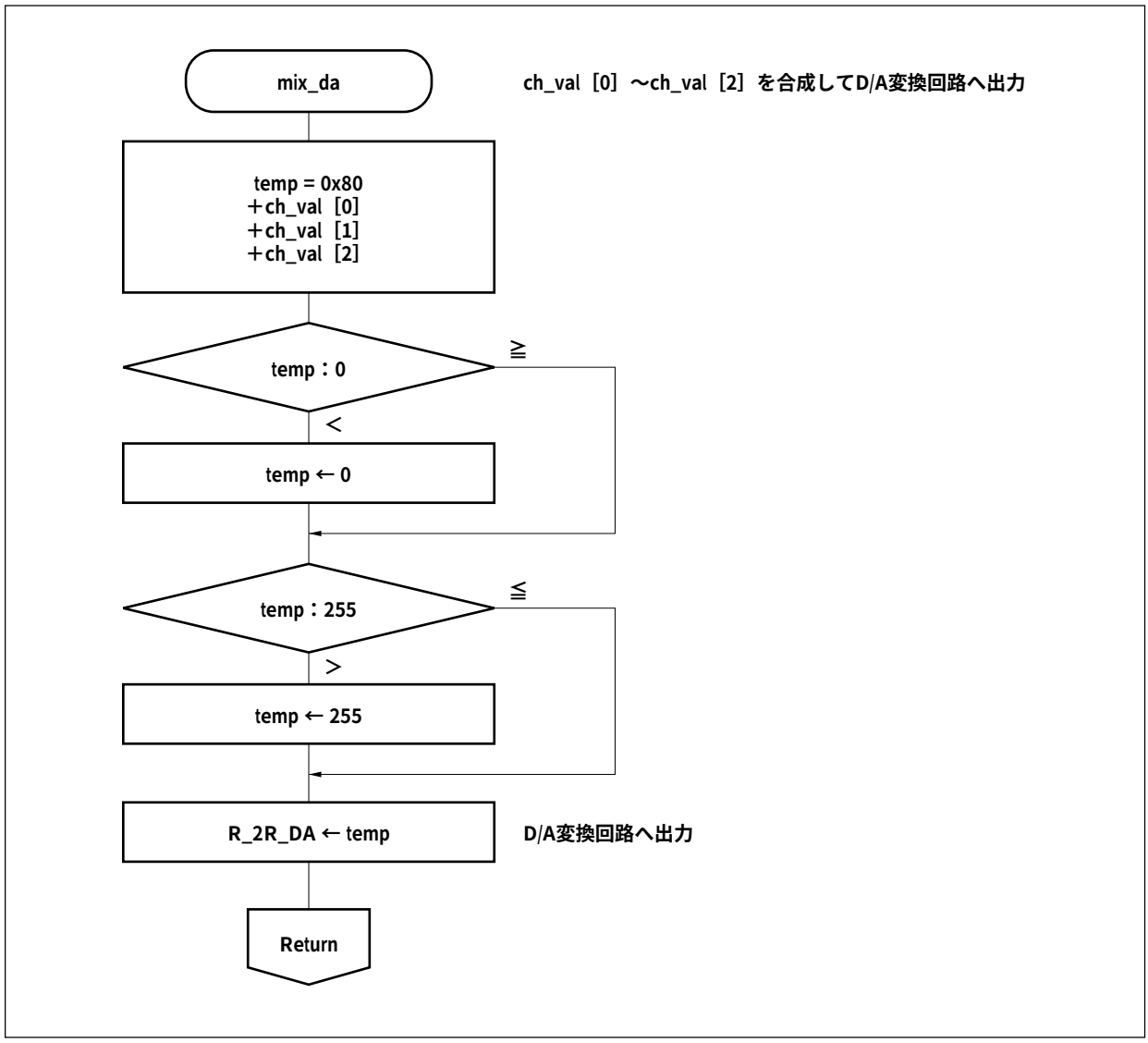


図4－24　複数チャネルの音楽演奏（ch.0音階インターバル割り込み処理）

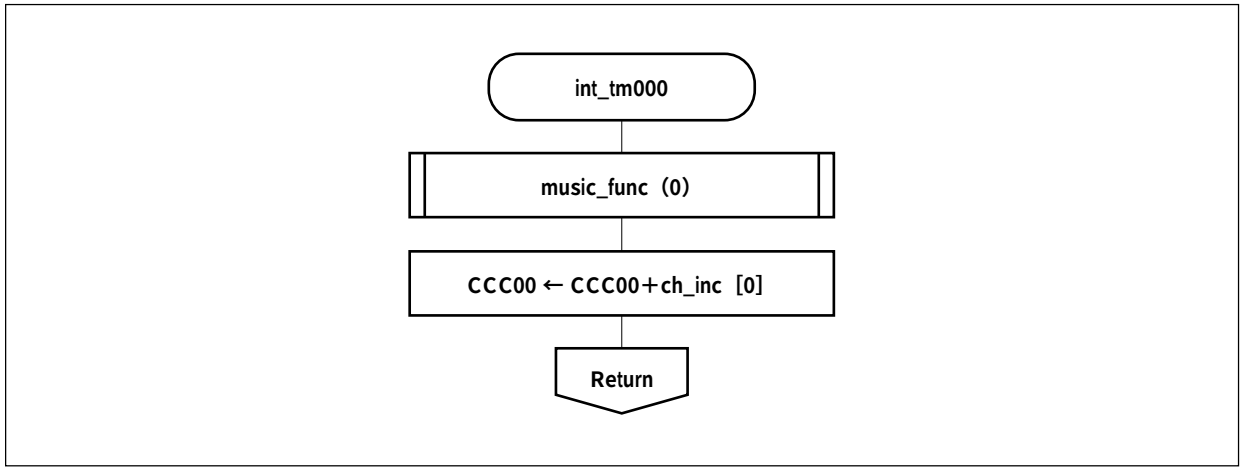


図4－25　複数チャネルの音楽演奏（ch.1音階インターバル割り込み処理）

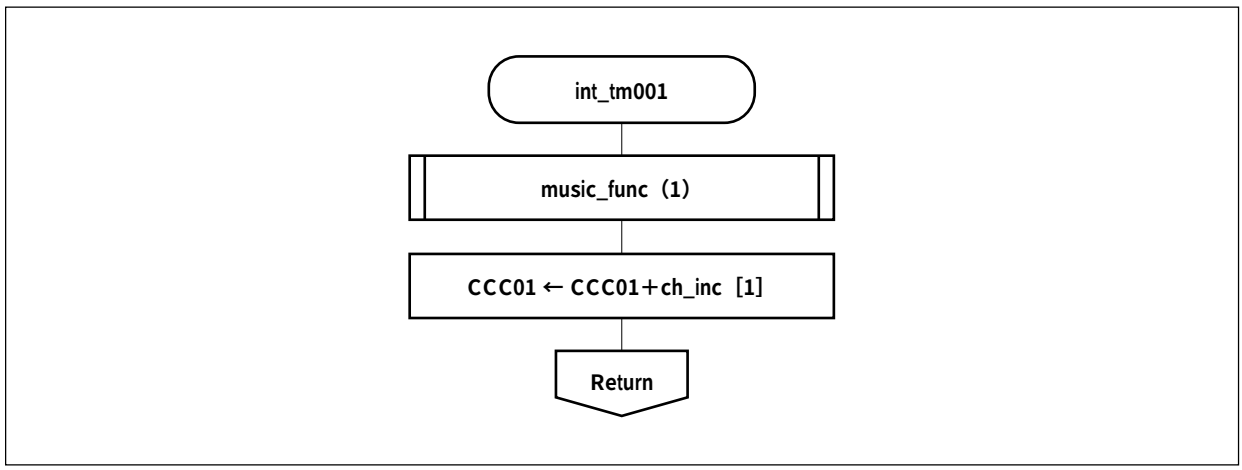


図4－26　複数チャネルの音楽演奏（ch.2音階インターバル割り込み処理）

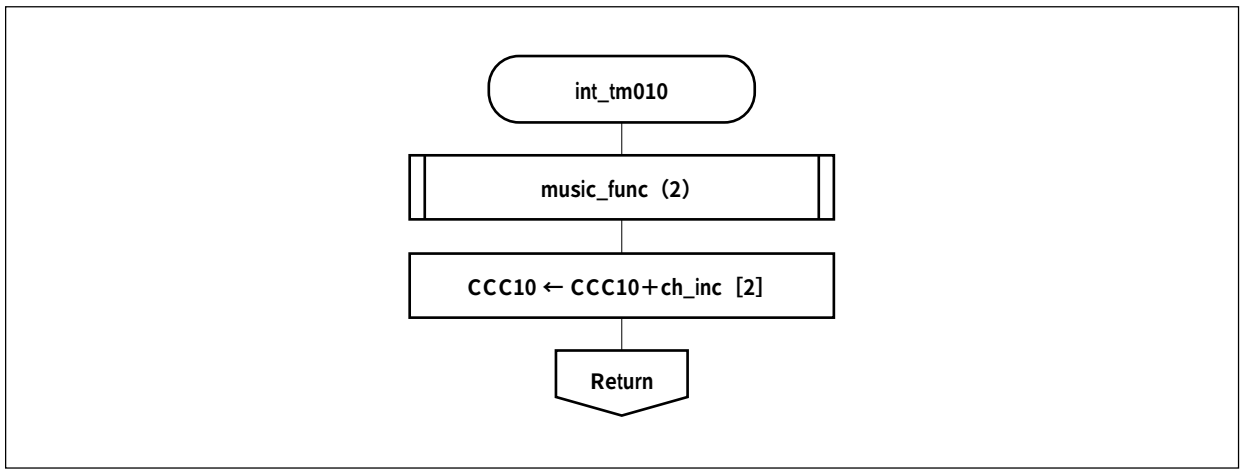


図4－27　複数チャネルの音楽演奏（1 msインターバル割り込み処理）

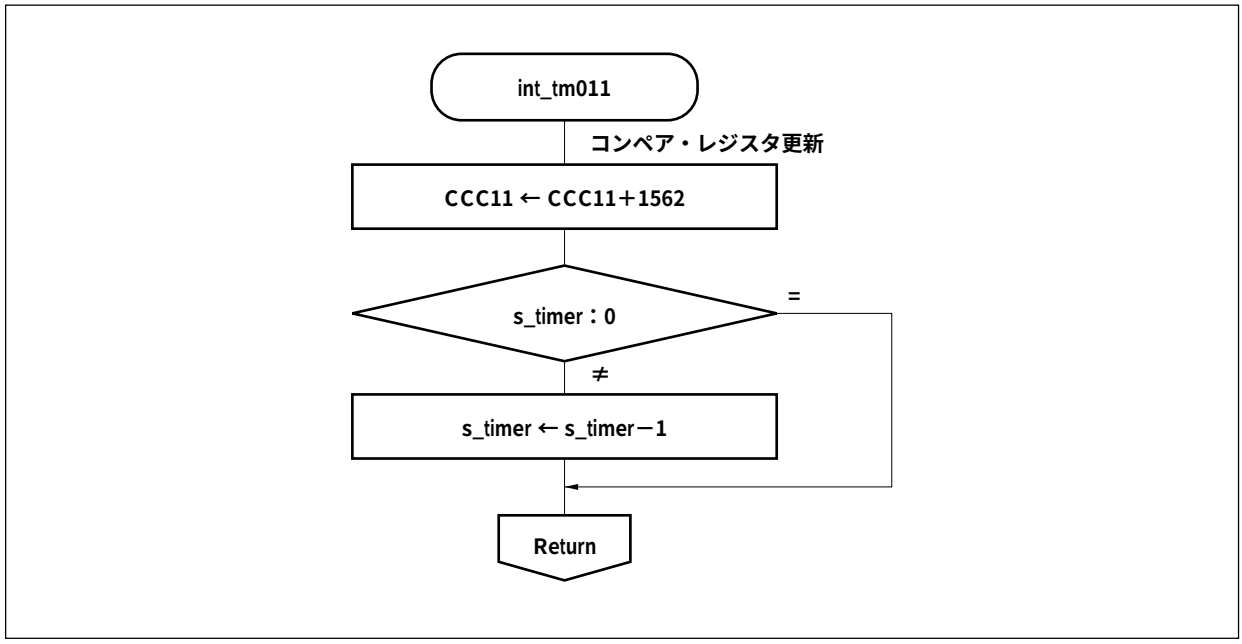


図4－28　複数チャネルの音楽演奏（音階インターバル割り込み共通処理）（1/2）

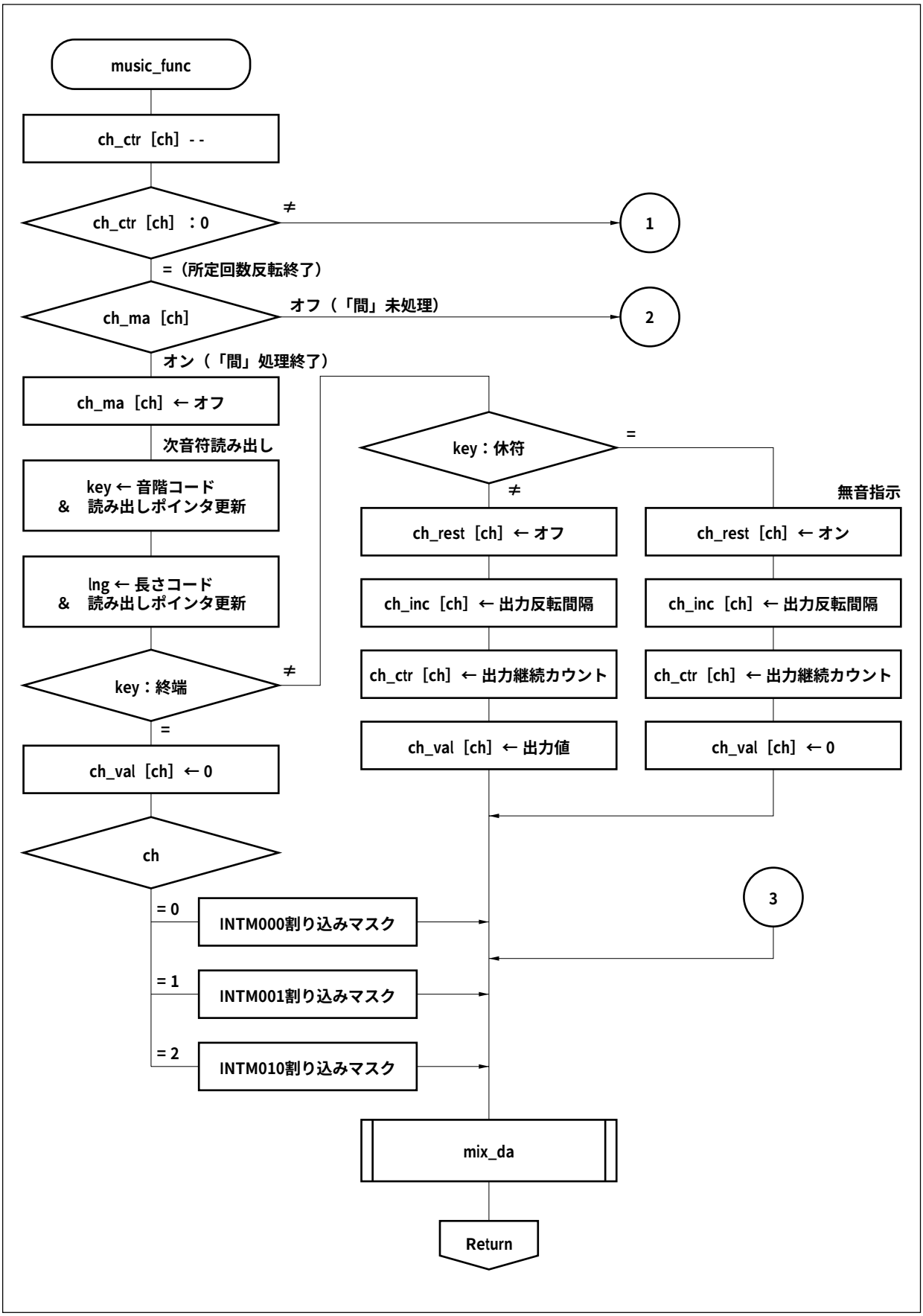
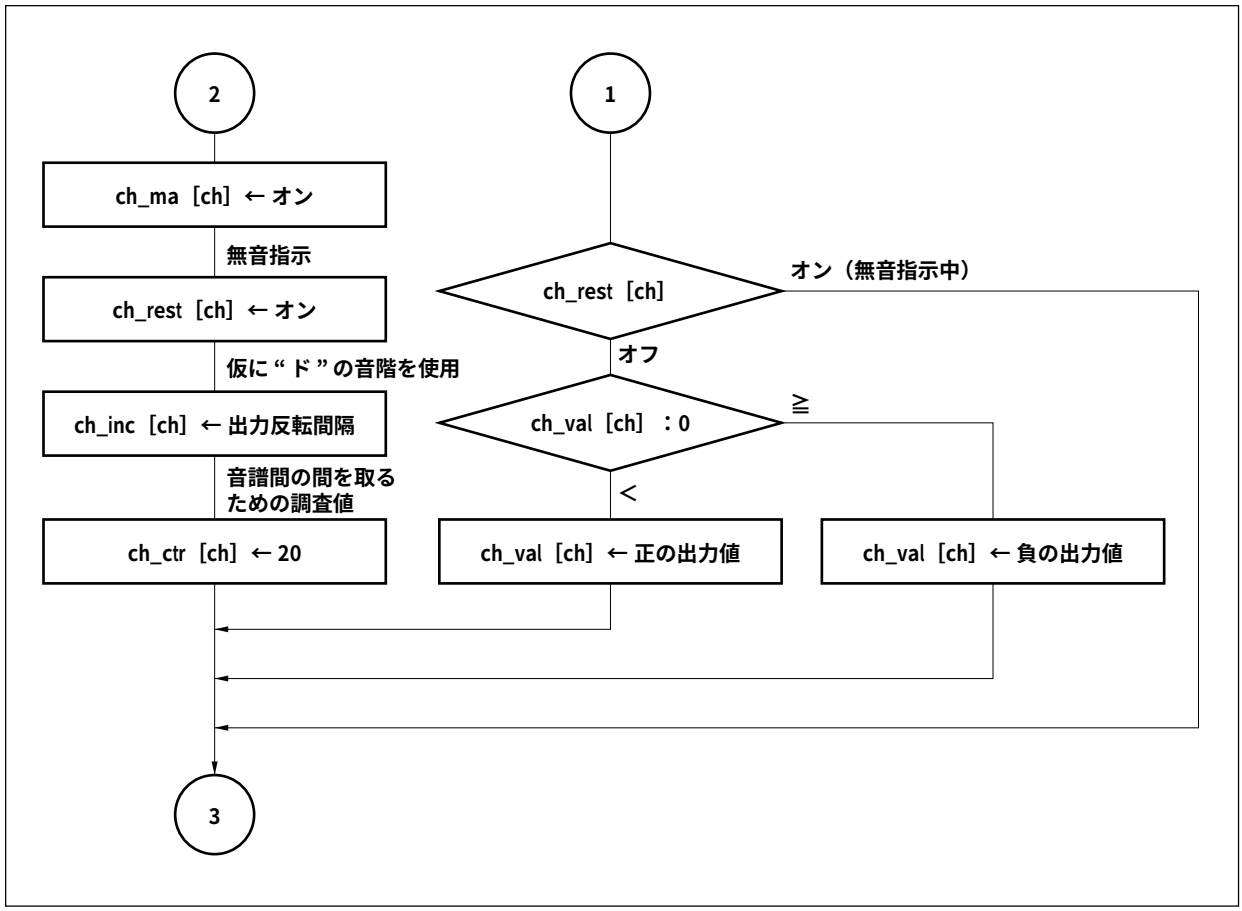


図4－28　複数チャネルの音楽演奏（音階インターバル割り込み共通処理）（2/2）



(h) プログラム・リスト

```
/**********************************/
/*                                */
/*   複数チャネル自動演奏         */
/*                                */
/**********************************/

#include    "common.h"
#include    "music_def.h"

extern unsigned short      freq_tabl[36][2];     /* 生成音階周波数テーブル */
extern unsigned char       sakura_tbl[60][2],hana_tbl[70][2],natsu_tbl[104][2];
extern unsigned char       daremo_tbl[79][2],jinglebell_tbl[56][2],doremi_tbl[68][2];
extern unsigned char       shiroi_tbl[156][2],yesterday_tbl[132][2] ;
extern unsigned char       *sou_po[8];

void int_tm000 (void) ;
void int_tm001 (void) ;
void int_tm010 (void) ;
void int_tm011 (void) ;

unsigned char music_start (unsigned char, unsigned char*) ;
void music_func (unsigned char) ;
void mix_da (void) ;

unsigned char       *ch_pt[3],ch_rest[3],ch_ma[3], sel;
int  ch_val[3];
unsigned short      ch_ctr[3],ch_inc[3];
unsigned int s_timer;
unsigned char led=0;

void debug (void)
{
            if (P00IC0 & 0x40) led |= 0x20; else led &=0xdf;
            if (P00IC1 & 0x40) led |= 0x40; else led &=0xbf;
            if (P01IC0 & 0x40) led |= 0x80; else led &=0x7f;
            P3=led;
            PBD=led;
}
```

```
main ()
{
        TMCC00.0 = 1;           /* TMC0 reset */
        TMCC00 |= 0x30;         /* TMC0 fxx/32 (t=0.64us)  50MHz時 */
        TMCC01 = 0x03;          /* compare clear disable, CCC00,01 are compare */
        TMCC00.1 = 1;           /* TMC0 count enable */

        TMCC10.0 = 1;           /* TMC1 reset */
        TMCC10 |= 0x30;         /* TMC1 fxx/32 (t=0.64us)  50MHz時 */
        TMCC11 = 0x03;          /* compare clear disable, CCC10,11 are compare */
        TMCC10.1 = 1;           /* TMC1 count enable */

        CCC11      = 1562;      /* for 1ms interval */
        sel = 1;
        P00IC0 = P00IC1 = P01IC0 = 0x47;
        P01IC1 = 0x07;
        __EI () ;

    while (1)
    {
        debug () ;
        if  ( ( P00IC0 & P00IC1 & P01IC0 & 0x40)  == 0x40)
        {
                led     &= 0xf8;
                led |=  (sel & 0x07) ;

                CCC00   = TMC0 + music_start (0,sou_po[sel]) ;
                P00IC0  &= 0x3f;

                debug () ;
                s_timer = 100;
                while (s_timer)  ;

                CCC01   = TMC0 + music_start (1,sou_po[sel]) ;
                P00IC1  &= 0x3f;

                debug () ;
                s_timer = 100;
                while (s_timer)  ;

                CCC10   = TMC1 + music_start (2,sou_po[sel]) ;
                P01IC0  &= 0x3f;
```

```
                    debug () ;
                    s_timer = 100;
                    while (s_timer) ;

                    sel++;
                    sel &= 0x07;
                    if (sel==0) sel++;
          }
     }
}

/***********************************/
/*          music start             */
/***********************************/
unsigned char music_start (unsigned char ch, unsigned char *score_p)
{
     unsigned char key,lng;

     ch_pt[ch] = score_p;
     key = *ch_pt[ch]++;
     lng = *ch_pt[ch]++;
     if (key != SILENCE)
     {
            ch_rest[ch] = OFF;
            ch_inc[ch] = freq_tabl[key][0];
            ch_ctr[ch] = freq_tabl[key][1]*lng;
            ch_val[ch] = (3-ch) *0x10;
     }else
     {
            ch_rest[ch] = ON;
            ch_inc[ch] = freq_tabl[12][0];
            ch_ctr[ch] = freq_tabl[12][1]*lng;
            ch_val[ch] = 0x00;
     }
     mix_da () ;
     return ch_inc[ch];
}
```

```
/***********************************/
/*   mix D/A output                */
/***********************************/
void mix_da (void)
{
       int temp;
     temp = 0x80 + ch_val[0] + ch_val[1] + ch_val[2];
     if  (temp < 0)      temp = 0;
     if  (temp > 255)    temp = 255;
     R_2R_DA =  (unsigned char) temp;
}

/*************************************************/
/*   タイマ割込み処理（for multi-channel music）    */
/*************************************************/
#pragma interrupt INTM000 int_tm000
__interrupt
void int_tm000 (void)
{
     music_func (0) ;
     CCC00 += ch_inc[0]/2;         /* コンペア・レジスタ更新 */
}

/* for music ch.1 */
#pragma interrupt INTP001 int_tm001
__interrupt
void int_tm001 (void)
{
     music_func (1) ;
     CCC01 += ch_inc[1]/2;         /* コンペア・レジスタ更新 */
}

/* for music ch.2 */
#pragma interrupt INTM010 int_tm010
__interrupt
void int_tm010 (void)
{
     music_func (2) ;
     CCC10 += ch_inc[2]/2;         /* コンペア・レジスタ更新 */
}

/* for 1ms interval */
#pragma interrupt INTM011 int_tm011
```

```
__interrupt
void int_tm011 (void)
{
     CCC11   += 1562;              /* for 1ms interval */
     if  (s_timer !=0 )
     {
            s_timer--;
     }
}

/*************************/
/* タイマ割り込み共通処理      */
/*************************/

void music_func (unsigned char ch)
{
     unsigned char key, lng;

     ch_ctr[ch]--;
     if  (ch_ctr[ch] == 0)
     {
            switch  (ch_ma[ch])
            {
                   case OFF:
                          ch_ma[ch]      = ON;
                          ch_rest[ch]    = ON;
                          ch_inc[ch]     = freq_tabl[12][0];
                          ch_ctr[ch]     = 20;
                          ch_val[ch]     = 0x00;
                          break;
                   case ON:
                          ch_ma[ch]      = OFF;
                          key    = *ch_pt[ch]++;
                          lng    = *ch_pt[ch]++;
                          if  (key == EndS)
                          {
                                 ch_val[ch]     = 0x0;
                                 switch  (ch)
                                 {
                                        case 0:P00IC0 |= 0x40;
                                                      break;
                                        case 1:P00IC1 |= 0x40;
                                                      break;
```

```
                                        case 2:P01IC0 |= 0x40;
                                                        break;
                                }
                        }else if（key == SILENCE）
                        {
                                ch_rest[ch]    = ON;
                                ch_inc[ch]     = freq_tabl[12][0];
                                ch_ctr[ch]     = freq_tabl[12][1]*lng;
                                ch_val[ch]     = 0x00;
                        }else
                        {
                                ch_rest[ch]    = OFF;
                                ch_inc[ch]     = freq_tabl[key][0];
                                ch_ctr[ch]     = freq_tabl[key][1]*lng;
                                ch_val[ch]     = （3-ch）*0x10;
                        }
                        break;
                }

        }else if （ch_rest[ch] == OFF）
        {
                if（ch_val[ch] < 0） ch_val[ch] = （3-ch）*0x10;
                else ch_val[ch] = -（3-ch）*0x10;
        }
        mix_da（）;
}

/*************************************************/
/*    音階データ・ヘッダ・ファイル                */
/*************************************************/
#define       SILENCE       0xff    /* 無効音              */
#define       C_L           0       /* 低音部ド            */
#define       Db_L          1       /*       ＃ド，♭レ    */
#define       D_L           2       /*       レ            */
#define       Eb_L          3       /*       ＃レ，♭ミ    */
#define       E_L           4       /*       ミ            */
#define       F_L           5       /*       ファ          */
#define       Gb_L          6       /*       ＃ファ，♭ソ  */
#define       G_L           7       /*       ソ            */
#define       Ab_L          8       /*       ＃ソ，♭ラ    */
#define       A_L           9       /*       ラ            */
#define       Bb_L          10      /*       ＃ラ，♭シ    */
#define       B_L           11      /*       シ            */
```

```
#define       C_M           12      /* 中音部ド              */
#define       Db_M          13      /*       ＃ド，♭レ      */
#define       D_M           14      /*       レ              */
#define       Eb_M          15      /*       ＃レ，♭ミ      */
#define       E_M           16      /*       ミ              */
#define       F_M           17      /*       ファ            */
#define       Gb_M          18      /*       ＃ファ，♭ソ    */
#define       G_M           19      /*       ソ              */
#define       Ab_M          20      /*       ＃ソ，♭ラ      */
#define       A_M           21      /*       ラ              */
#define       Bb_M          22      /*       ＃ラ，♭シ      */
#define       B_M           23      /*       シ              */
#define       C_H           24      /* 高音部ド              */
#define       Db_H          25      /*       ＃ド，♭レ      */
#define       D_H           26      /*       レ              */
#define       Eb_H          27      /*       ＃レ，♭ミ      */
#define       E_H           28      /*       ミ              */
#define       F_H           29      /*       ファ            */
#define       Gb_H          30      /*       ＃ファ，♭ソ    */
#define       G_h           31      /*       ソ              */
#define       Ab_H          32      /*       ＃ソ，♭ラ      */
#define       A_H           33      /*       ラ              */
#define       Bb_H          34      /*       ＃ラ，♭シ      */
#define       B_H           35      /*       シ              */
#define       EndS          0xfe    /* End of Score          */

/***********************************************/
/*    音符                                     */
/***********************************************/
#define       T32           1       /*  32分音符     */
#define       T16           2       /*  16分音符     */
#define       T8            4       /*  8分音符      */
#define       T4            8       /*  4分音符      */
#define       T2            16      /*  2分音符      */
#define       T1            32      /*  全音符       */

/***********************************************/
/*    休符                                     */
/***********************************************/
#define       R32           1       /*  32分休符     */
#define       R16           2       /*  16分休符     */
#define       R8            4       /*  8分休符      */
#define       R4            8       /*  4分休符      */
```

```
#define       R2            16      /*  2分休符       */
#define       R1            32      /*  全休符        */
#define _FF 255
#define _PP 80

/**** 定数定義ファイル ****/
#include "music_def.h"

/*****************************************/
/*                                       */
/*    生成音周波数データ・テーブル       */
/*                                       */
/*****************************************/
unsigned short      freq_tabl[36][2]=
{
      2986, 49,                   /*  0：低音部ド（無効）  */
      2819, 52,                   /*  1：  ＃ド，♭レ      */
      2660, 55,                   /*  2：  レ              */
      2519, 58,                   /*  3：  ＃レ，♭ミ      */
      2370, 62,                   /*  4：  ミ              */
      2237, 65,                   /*  5：  ファ            */
      2111, 69,                   /*  6：  ＃ファ，♭ソ    */
      1993, 74,                   /*  7：  ソ              */
      1881, 78,                   /*  8：  ＃ソ，♭ラ      */
      1776, 83,                   /*  9：  ラ              */
      1676, 87,                   /* 10：  ＃ラ，♭シ      */
      1582, 93,                   /* 11：  シ              */

      1493, 98,                   /* 12：中音部ド          */
      1409, 104                   /* 13：  ＃ド，♭レ      */
      1330, 110,                  /* 14：  レ              */
      1260, 116,                  /* 15：  ＃レ，♭ミ      */
      1185, 124,                  /* 16：  ミ              */
      1119, 131,                  /* 17：  ファ            */
      1056, 139,                  /* 18：  ＃ファ，♭ソ    */
      996,  147,                  /* 19：  ソ              */
      941,  156,                  /* 20：  ＃ソ，♭ラ      */
      888,  165,                  /* 21：  ラ              */
      838,  175,                  /* 22：  ラ，♭シ        */
      791,  185,                  /* 23：  シ              */

      747,  196,                  /* 24：高音部ド          */
      705,  208,                  /* 25：  ＃ド，♭レ      */
```

```
      665,  220,                    /* 26：  レ          */
      630,  233,                    /* 27：  #レ，♭ミ    */
      593,  247,                    /* 28：  ミ          */
      559,  262,                    /* 29：  ファ        */
      528,  278,                    /* 30：  #ファ，♭ソ  */
      498,  294,                    /* 31：  ソ          */
      470,  311,                    /* 32：  #ソ，♭ラ    */
      444,  330,                    /* 33：  ラ          */
      419,  350,                    /* 34：  #ラ，♭シ    */
      395,  370                     /* 35：  シ          */
};

/****************************************/
/*                                      */
/*   楽譜（0）：     さくら             */
/*                                      */
/****************************************/
unsigned char       sakura_tbl[60][2] =
     { { SILENCE, T4 }, { SILENCE, T4 }, { SILENCE, T4 }, { SILENCE, T4 },
       { D_H, T4 }, { D_H, T4 }, { E_H, T2 }, { D_H, T4 }, { D_H, T4 },
       { E_H, T2 }, { D_H, T4 }, { E_H, T4 }, { F_H, T4 }, { E_H, T4 },
       { D_H, T4 }, { E_H, T8 }, { D_H, T8 }, { Bb_M, T2 },
       { A_M, T4 }, { F_M, T4 }, { A_M, T4 }, { Bb_M, T4 },
       { A_M, T4 }, { A_M, T8 }, { F_M, T8 }, { E_M, T2 },

       { D_H, T4 }, { E_H, T4 }, { F_H, T4 }, { E_H, T4 },
       { D_H, T4 }, { E_H, T8 }, { D_H, T8 }, { Bb_M, T2 },
       { A_M, T4 }, { F_M, T4 }, { A_M, T4 }, { Bb_M, T4 },
       { A_M, T4 }, { A_M, T8 }, { F_M, T8 }, { E_M, T2 },

       { D_H, T4 }, { D_H, T4 }, { E_H, T2 }, { D_H, T4 }, { D_H, T4 },
       { E_H, T2 },
       { SILENCE, T8 }, { A_M, T8 }, { Bb_M, T2 }, { E_H, T8 },
       { D_H, T8 }, { Bb_M, T1 },{ A_M, T1 },
       { SILENCE, T4 }, { SILENCE, T4 }, { SILENCE, T4 }, { SILENCE, T4 },
       { EndS,T1 }
     };

/****************************************/
/*                                      */
/*   その他の7曲のデータは省略します。  */
/*                                      */
/****************************************/
```

```
/*************************************/
/*                                   */
/*    楽曲テーブル                   */
/*                                   */
/*************************************/
unsigned char   *sou_po[8] =
      {     &sakura_tbl[0][0],&hana_tbl[0][0],&natsu_tbl[0][0],
            &daremo_tbl[0][0],&jinglebell_tbl[0][0],&doremi_tbl[0][0],
            &shiroi_tbl[0][0],&yesterday_tbl[0][0]      } ;

/**** コモン・ファイル（初期化部含む）****/
#pragma             ioreg
#define     ON  0xff
#define     OFF 0x00

#define     LED_DGT1        * (unsigned char *) 0xfe00000
#define     LED_DGT2        * (unsigned char *) 0xfe00002
#define     LED_DGT3        * (unsigned char *) 0xfe00004
#define     LED_DGT4        * (unsigned char *) 0xfe00006
#define     LED_DGT5        * (unsigned char *) 0xfe00008
#define     R_2R_DA         * (unsigned char *) 0xfe0000a

/*********************************/
/*    数値表示関数（disp_num）          */
/*********************************/
void disp_num ( unsigned int num )
{
     unsigned char num_data[10] =
     {0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x27,0x7f,0x67};

     unsigned int  d_sample;

     d_sample = num;
     LED_DGT5 = num_data[d_sample / 10000];
     d_sample = d_sample % 10000;
     LED_DGT4 = num_data[d_sample / 1000];
     d_sample = d_sample % 1000;
     LED_DGT3 = num_data[d_sample / 100];
     d_sample = d_sample % 100;
     LED_DGT2 = num_data[d_sample / 10];
     LED_DGT1 = num_data[d_sample % 10];
}
```

```
/**********************************/
/*   16進表示関数（disp_hex）         */
/**********************************/
void disp_hex ( unsigned int num )
{
     unsigned char hex_data[16]=
     {  0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x27,
        0x7f,0x67,0x77,0x7c,0x58,0x5e,0x79,0x71 };

     LED_DGT5 = 0x00;
     LED_DGT4 = hex_data[ (num>>12)  & 0x000f];
     LED_DGT3 = hex_data[ (num>>8)  & 0x000f];
     LED_DGT2 = hex_data[ (num>>4)  & 0x000f];
     LED_DGT1 = hex_data[num & 0x000f];
}
```

（12）比例／積分制御によるDCモータの速度制御

（a）機能概要

PWMユニットに接続された直流モータの速度制御を行います。

回転速度は，PWMユニットから出力するパルスのデューティでコントロールします。

直流モータに付属するエンコーダから，1回転あたり12個のエンコーダ・パルスが出力されます。

このエンコーダ・パルスを監視することにより現在の回転速度を知ることができます。

ボリュームにより調整できる入力電圧を，A/Dコンバータより読み取り，その値により目標回転速度を確定します。

現在の回転速度と目標回転速度を監視し，目標回転速度に追随させるPWM制御を行います。

比例制御と積分制御を組み合わせた方式で追随制御を実現します。

（b）プログラム構成

・メイン関数

周辺I/O（タイマ・ユニット，PWMユニット，A/Dコンバータ）初期化

グローバル変数初期化，A/Dコンバータ起動

次の処理による永久ループ

A/D変換が終了していたら，変換結果により目標速度確定

A/D変換再起動

・INTM000割り込みハンドラ（10 msインターバル割り込み）

現在速度と目標速度を使用し，比例／積分制御によりPWM出力

目標速度，現在速度を7セグメントLEDに表示

目標速度，現在速度の差をLEDに表示

・INTP000割り込みハンドラ（エンコーダ・パルス割り込み）

エンコーダ・パルス・カウント変数にプラス1

・INTAD割り込みハンドラ

変換結果をグローバル変数に格納

A/D変換終了フラグ・セット

(c) フロー・チャート

図4－29　比例／積分制御によるDCモータの速度制御（メイン関数）

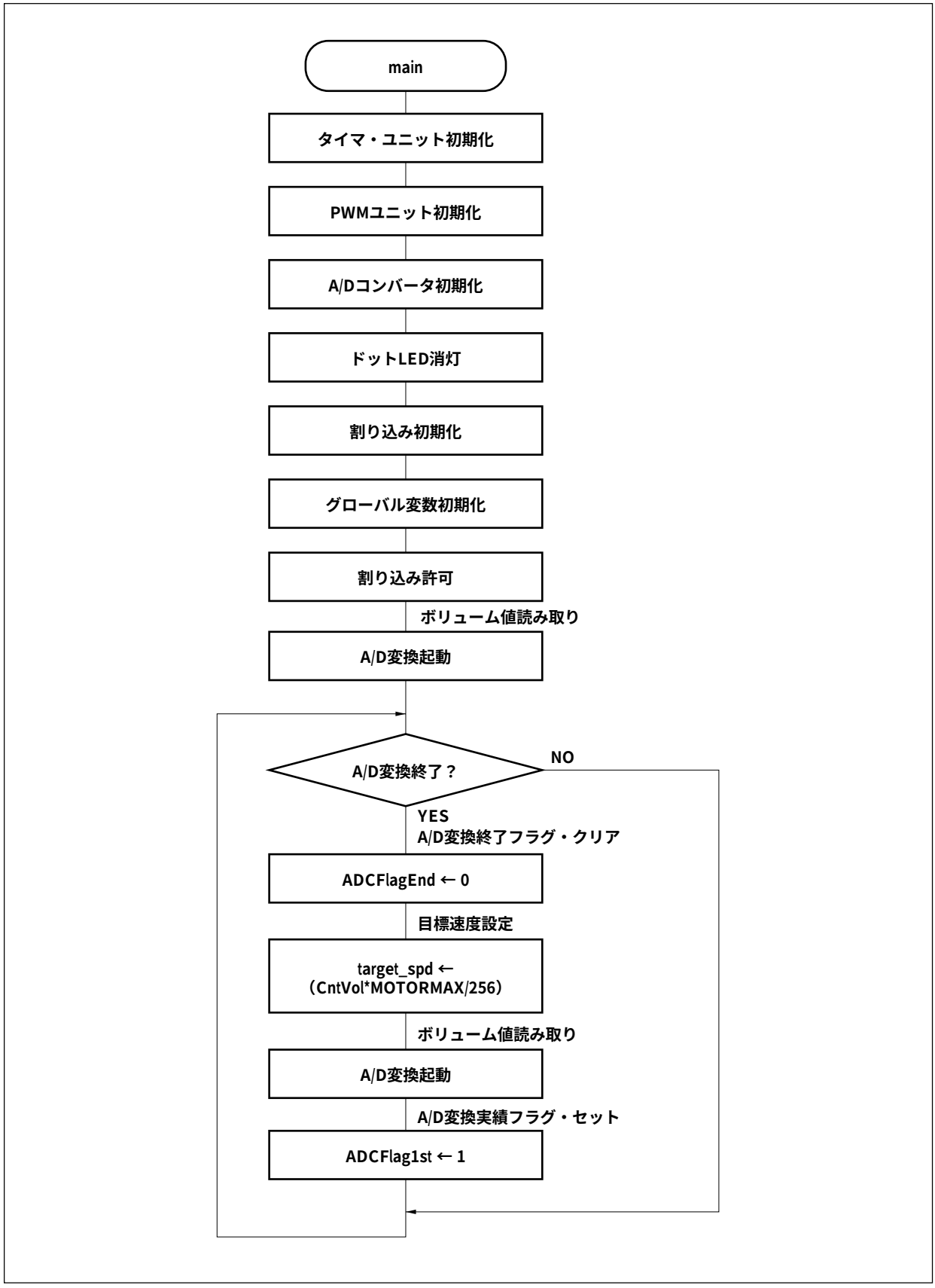


図4－30　比例／積分制御によるDCモータの速度制御（INTP000割り込みハンドラ）

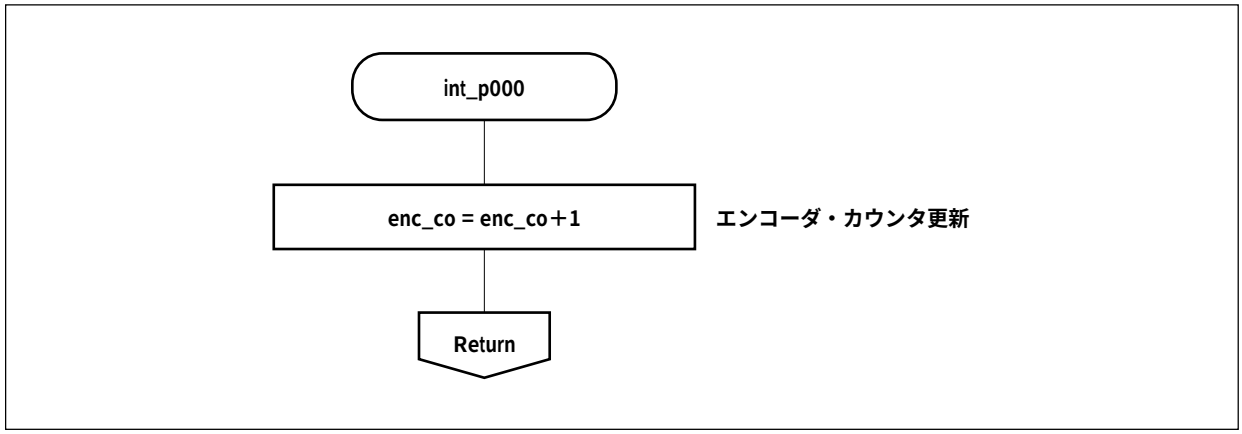


図4－31　比例／積分制御によるDCモータの速度制御（INTAD割り込みハンドラ）

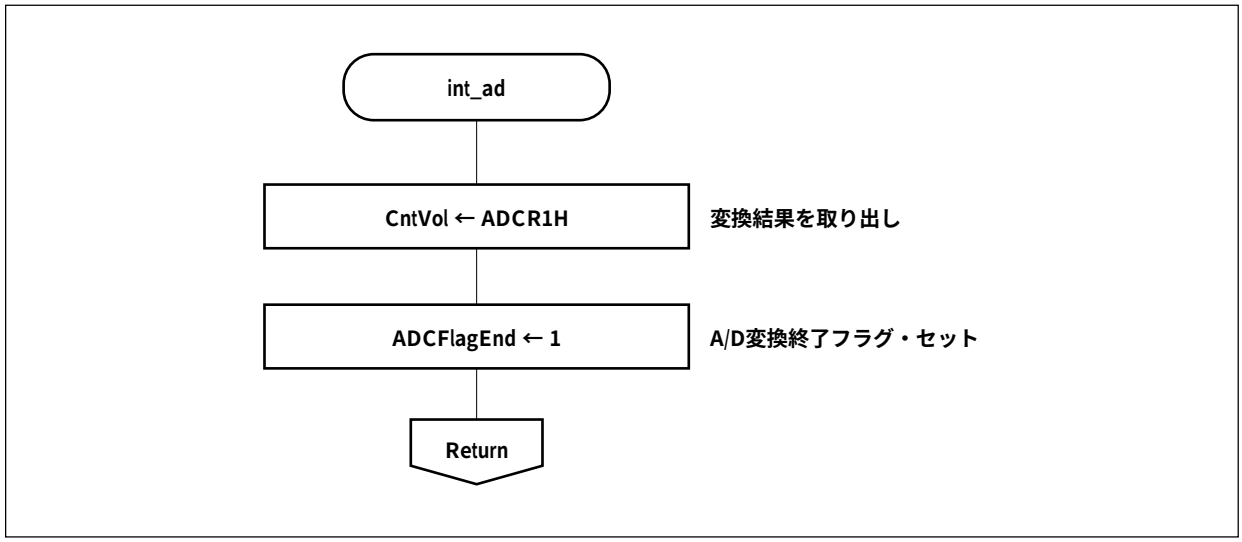


図4－32　比例／積分制御によるDCモータの速度制御（INTM000割り込みハンドラ）

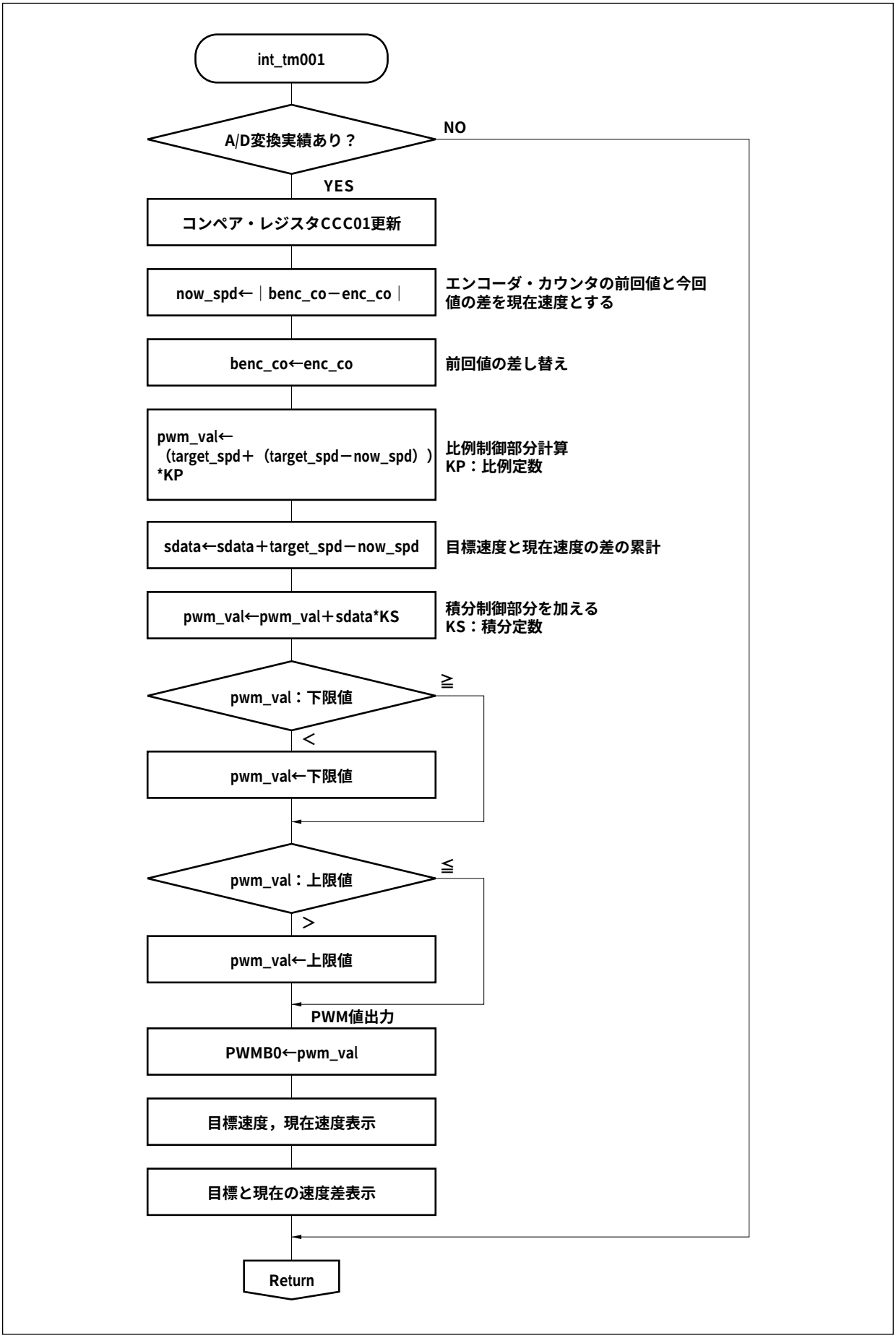
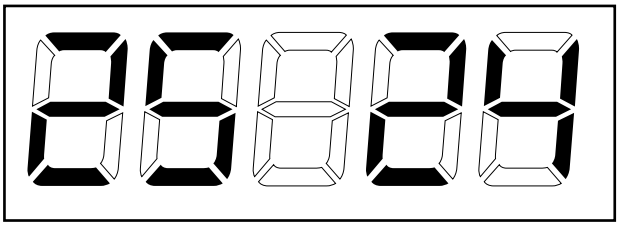


図4－33　7セグメントLED



目標速度　　現在速度

図4－34　ドットLED（速度差表示）

表　　示	目標に対する現在速度
	－4以上（遅い）
	－3
	－2
	－1
	目標速度 = 現在速度
	＋1
	＋2
	＋3以上（速い）

(d) プログラム・リスト

```
/*****************************************************************
        比例と積分によるDCモータ制御
ボリュームを回すことで、DCモータの目標速度を7segLEDの5,4桁目に表示する。
10msec毎の割り込みでエンコーダ・パルスをサンプリングし,目標速度を維持するように
比例と積分によりフィードバック制御を行う。
現在速度は7segLEDの2,1桁目に表示する。
現在速度と目標速度との差分をDotLEDにインジケータ表示する。

DCモータはV850E/MA1内蔵のPWM出力により制御されている。
ボリュームはV850E/MA1内蔵のA/Dコンバータに接続されている。
DCモータからエンコーダ・パルスが出力され、そのパルスはINTP000に割り込む。
エンコーダ・パルスは1回転に12パルス出力される。
*****************************************************************/
#include      "common.h"

/* 定数値定義 */
#defineMOTORMAX       73      /* モータ最大定格値       */
#defineLOWERLIMIT     0       /* PWM出力下限値          */
#defineUPPERLIMIT     255     /* PWM出力上限値          */
#defineKP             2.7     /* 比例定数               */
#defineKS             0.5     /* 積分定数               */

/* 関数プロトタイプ宣言 */
void          int_p000(void);
void          int_tm001(void);
void          int_ad(void);

/* グローバル変数定義 */
short         CntVol;         /* 速度指定ボリューム値          */
int           ADCFlagEnd;     /* A/D変換終了フラグ             */
int           ADCFlag1st;     /* A/D変換初回フラグ             */
unsigned int  enc_co;         /* エンコーダ・カウンタ          */
unsigned int  benc_co;        /* 前回エンコーダ・カウンタ      */
short         target_spd;     /* 目標速度                      */
int           sdata;          /* 積分値                        */
```

```
/*******************************/
/*            main関数          */
/*******************************/
void main(void)
{
      /* タイマC0ユニット初期化 */
      TMCC00.0 = 1;          /* 動作許可:TMCCAE0=1                */
      TMCC00 |= 0x50;        /* 動作Clock=fxx/128(3.2us) 40MHz時  */
      TMCC01.1 = 1;          /* CCC01 コンペア・モード:CMS01=1     */
      CCC01 = 39060;         /* コンペア値設定10ms=3.2us*39060     */
      TMCC00.1 = 1;          /* カウント開始:TMCCE0=1              */

      /* PWMユニット初期化     */
      PWMB0 = 0;             /* 出力Low                           */
                             /* アクティブHigh,カウンタ8bit長      */
      PWMC0 = 0x43;          /* 動作Clock fxx/16(0.4us) 40MHz時   */
      PWMC0 |= 0x80;         /* 動作許可                          */

      /* A/Dコンバータユニット初期化 */
      ADM2  = 0x00;          /* リセット                          */
      ADM2 |= 0x01;          /* 動作許可                           */
      ADM0  = 0x11;          /* ANI1端子指定，セレクト,1バッファ    */
      ADM1  = 0x03;          /* A/Dトリガ,変換時間=6.0us 40MHz時   */

      P3 = 0x00;             /* LED 上位4bit消灯                  */
      PBD = 0x00;            /* LED 下位4bit消灯                  */

      /* 割り込み初期化        */
      SESC0=0x01;            /* INTP000の有効エッジは立ち上がり                */
      P00MK0=0;              /* INTP000:エンコーダ・パルス割り込みマスク解除    */
      P00MK1=0;              /* INTM001:タイマC0割り込みマスク解除              */
      ADMK=0;                /* INTAD:A/D変換終了割り込みマスク解除             */

      /* グローバル変数初期    */
      CntVol          = 0;   /* 速度指定ボリューム値クリア                     */
      enc_co          = 0;   /* エンコーダ累計カウンタ初期化                   */
      benc_co         = 0;   /* 前回エンコーダ・カウンタ・クリア               */
      target_spd      = 0;   /* 目標速度クリア                                 */
      sdata           = 0;   /* 積分値クリア                                   */
      ADCFlagEnd      = 0;   /* A/D変換終了フラグ・クリア                      */
      ADCFlag1st      = 0;   /* A/D変換初回フラグ・クリア                      */

      __EI();                /* 割り込み許可                                   */
```

```
        ADM0 |= 0x80;          /* A/D変換開始:CE=1                          */

         while(1) {
             if ( ADCFlagEnd ) {                          /* A/D変換値終了?          */
                   ADCFlagEnd = 0;                        /* A/D変換終了フラグ・クリア */
                   target_spd = ( CntVol * MOTORMAX / 256 );/* 目標速度設定          */
                   ADM0 |= 0x80;                          /* A/D変換開始             */
                   ADCFlag1st = 1;                        /* A/D変換初回フラグ・セット */
             }
        }
}

/*******************************/
/*  INTTM001(10msec) 割り込み処理  */
/*******************************/
#pragma interrupt INTM001 int_tm001
__interrupt
void    int_tm001(void)
{
        int now_spd;            /* 現在速度                  */
        int pwm_val;            /* PWM出力値                 */
        int diff;               /* 速度差                    */
        unsigned char led;      /* Dot LED表示バッファ       */

        if( ADCFlag1st )        /* A/D変換一回以上起動済み?   */
        {
              /* DCモータ比例／積分制御                  */
              CCC01 += 39060;                       /* 10msタイマ・コンペア値再設定      */
              now_spd = abs( benc_co - enc_co );    /* 現在速度算出                      */
              benc_co = enc_co;                     /* 前回エンコーダ値更新              */
              pwm_val = ( target_spd + ( target_spd - now_spd ) ) * KP;/* 比例部分計算 */
              sdata += ( target_spd - now_spd );    /* 速度の累計                        */
              pwm_val += ( sdata * KS );            /* 積分部分計算                      */
              if( pwm_val < LOWERLIMIT )pwm_val = LOWERLIMIT;/* PWM値下限            */
              if( pwm_val > UPPERLIMIT )pwm_val = UPPERLIMIT;/* PWM値上限            */
              PWMB0 = pwm_val;                      /* PWM値出力                         */

              /* 目標速度と現在速度を7segLEDに数値表示    */
              disp_num(((unsigned int)target_spd * 1000) + (unsigned char)now_spd);
              LED_DGT3 = 0;     /* 7segLEDの3桁目に空白 */
```

```
            /* 速度制御状態をDot LEDに表示   */
            diff = now_spd - target_spd; /* 目標速度までの差分を算出 */
            if(diff>=0)
            { /* 目標速度より早い場合 */
            switch(diff)
            {   /* 目標速度まで */
                  case 0  :   led =0x10; break;/* 目標速度と一致           */
                  case 1  :   led =0x30; break;/* エンコーダ1パルス分速い   */
                  case 2  :   led =0x50; break;/* エンコーダ2パルス分速い   */
                  default :   led =0x90; break;/* エンコーダ3パルス以上速い */
            }
      }else
      { /* 目標速度より遅い場合 */
            switch(abs(diff))
            { /* 目標速度まで */
                  case 1  :   led =0x18; break;/* エンコーダ1パルス分遅い   */
                  case 2  :   led =0x14; break;/* エンコーダ2パルス分遅い   */
                  case 3  :   led =0x12; break;/* エンコーダ3パルス分遅い   */
                  default :   led =0x11; break;/* エンコーダ4パルス以上遅い */
            }
      }
      P3 = led & 0xf0;         /* LED 上位4bit点灯 */
      PBD= led & 0x0f;         /* LED 下位4bit点灯 */
      }
}
/*******************************/
/*  INP000        割り込み処理      */
/*******************************/
#pragma interrupt INTP000 int_p000
__interrupt
void    int_p000(void)
{
       enc_co++;               /* エンコーダ累計カウント */
}
/*******************************/
/*  INTAD         割り込み処理      */
/*******************************/
#pragma interrupt INTAD int_ad
__interrupt
void    int_ad(void)
{
       CntVol = ADCR1H;        /* A/D変換値取得               */
       ADCFlagEnd = 1;         /* A/D変換終了フラグ・セット    */
}
```

〔メ　モ〕

図4－35　TU-V850E/MA1（1/8）

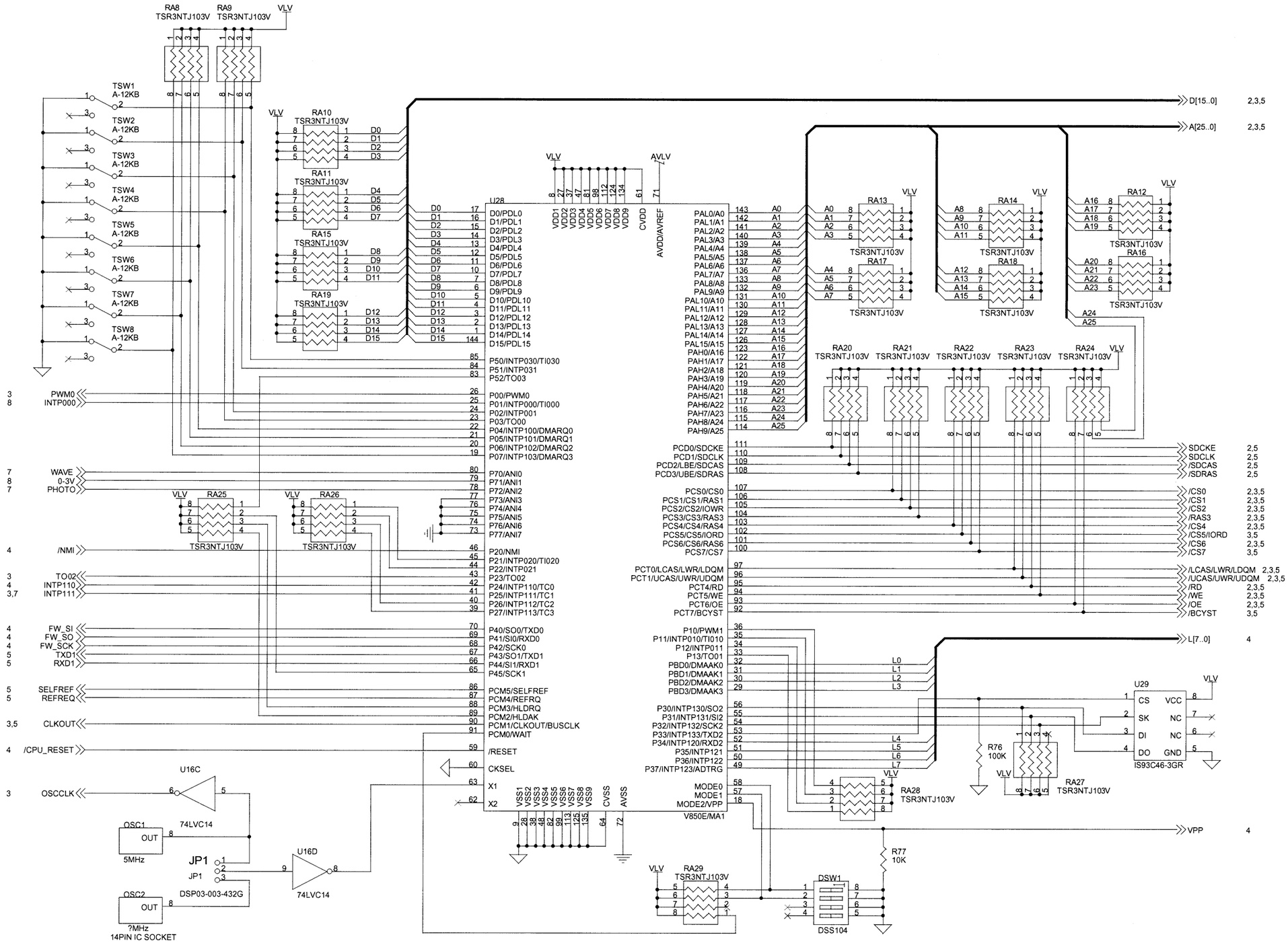


図4－35　TU-V850E/MA1（2/8）

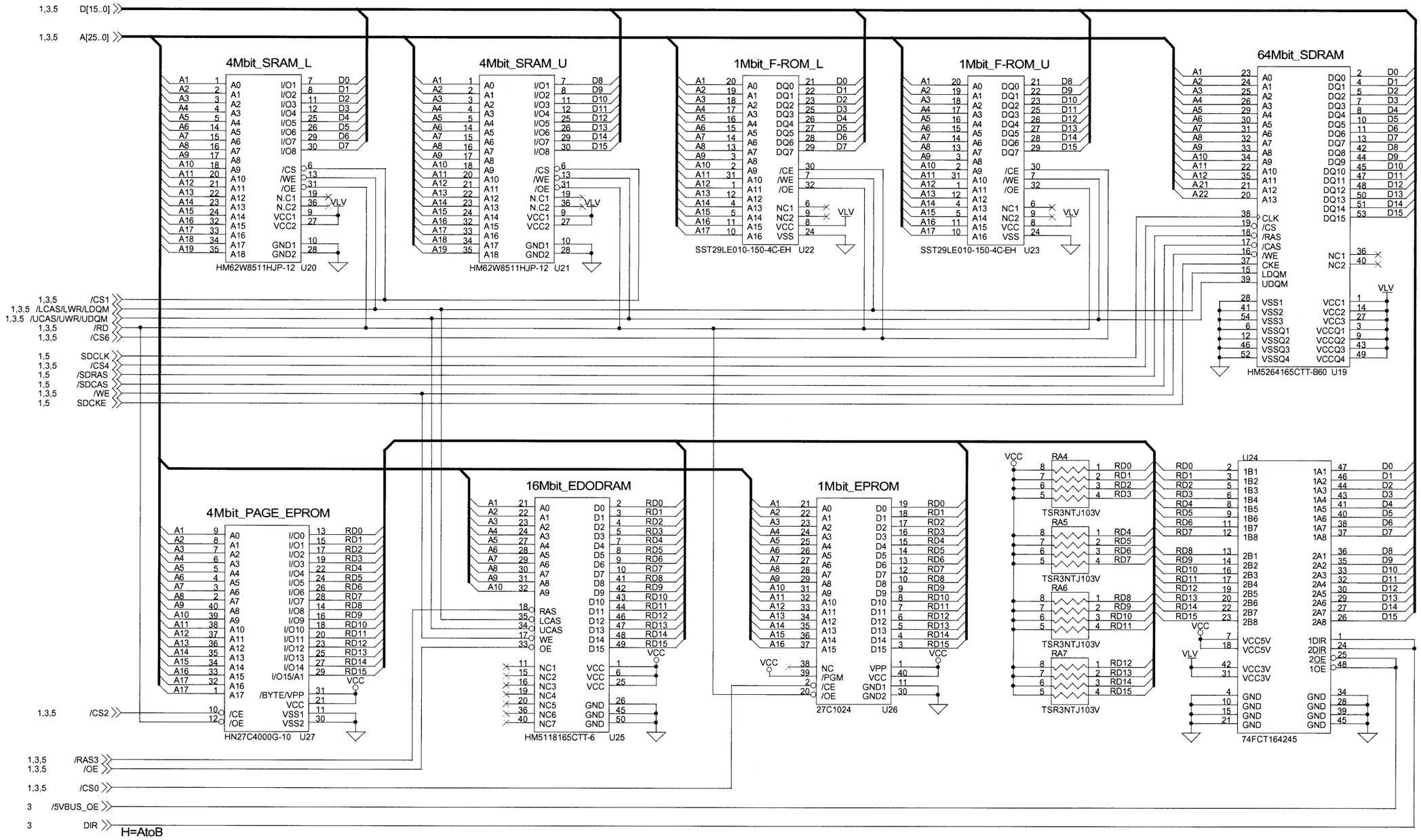


図4－35　TU-V850E/MA1（3/8）

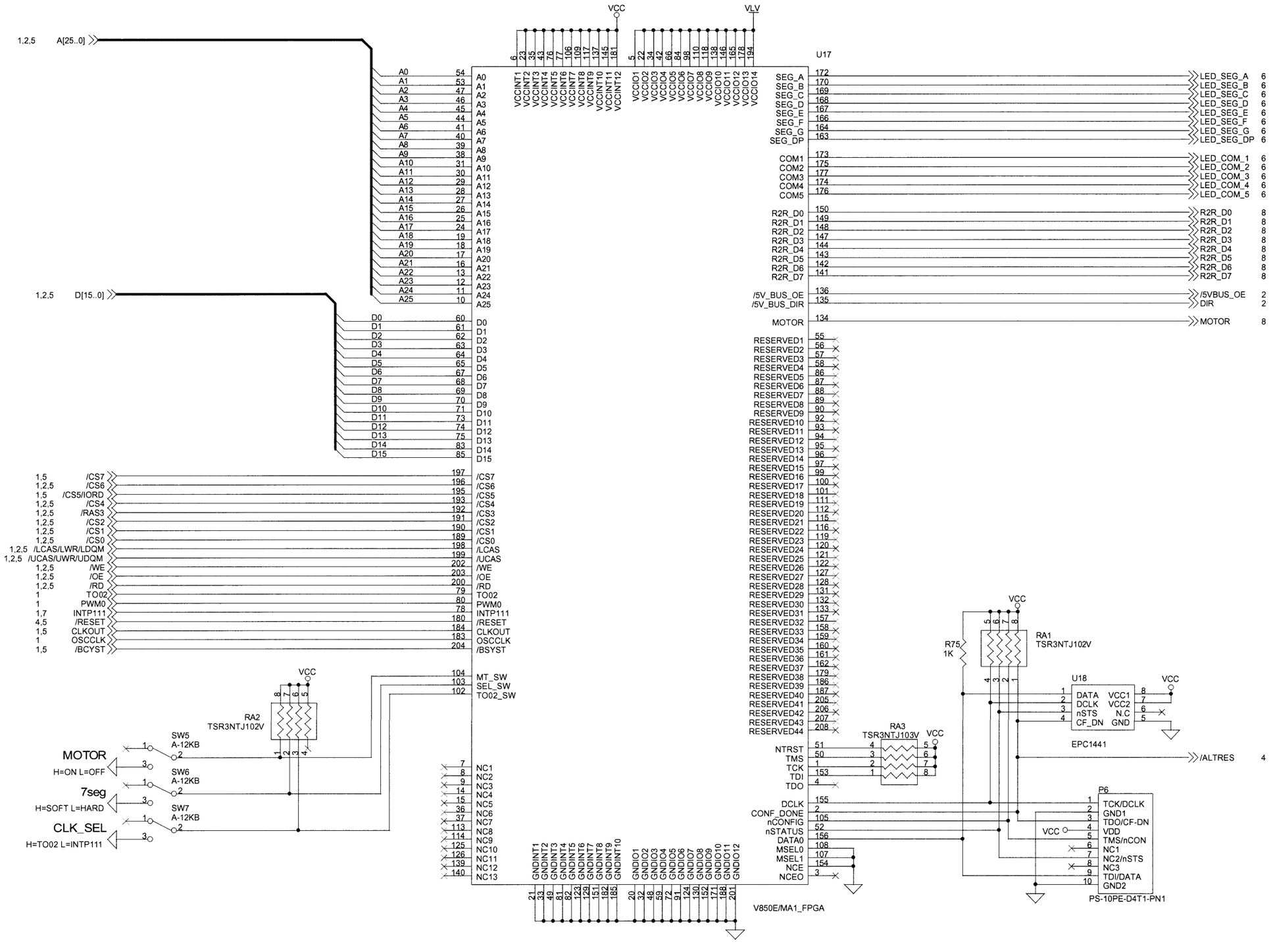


図4－35　TU-V850E/MA1（4/8）

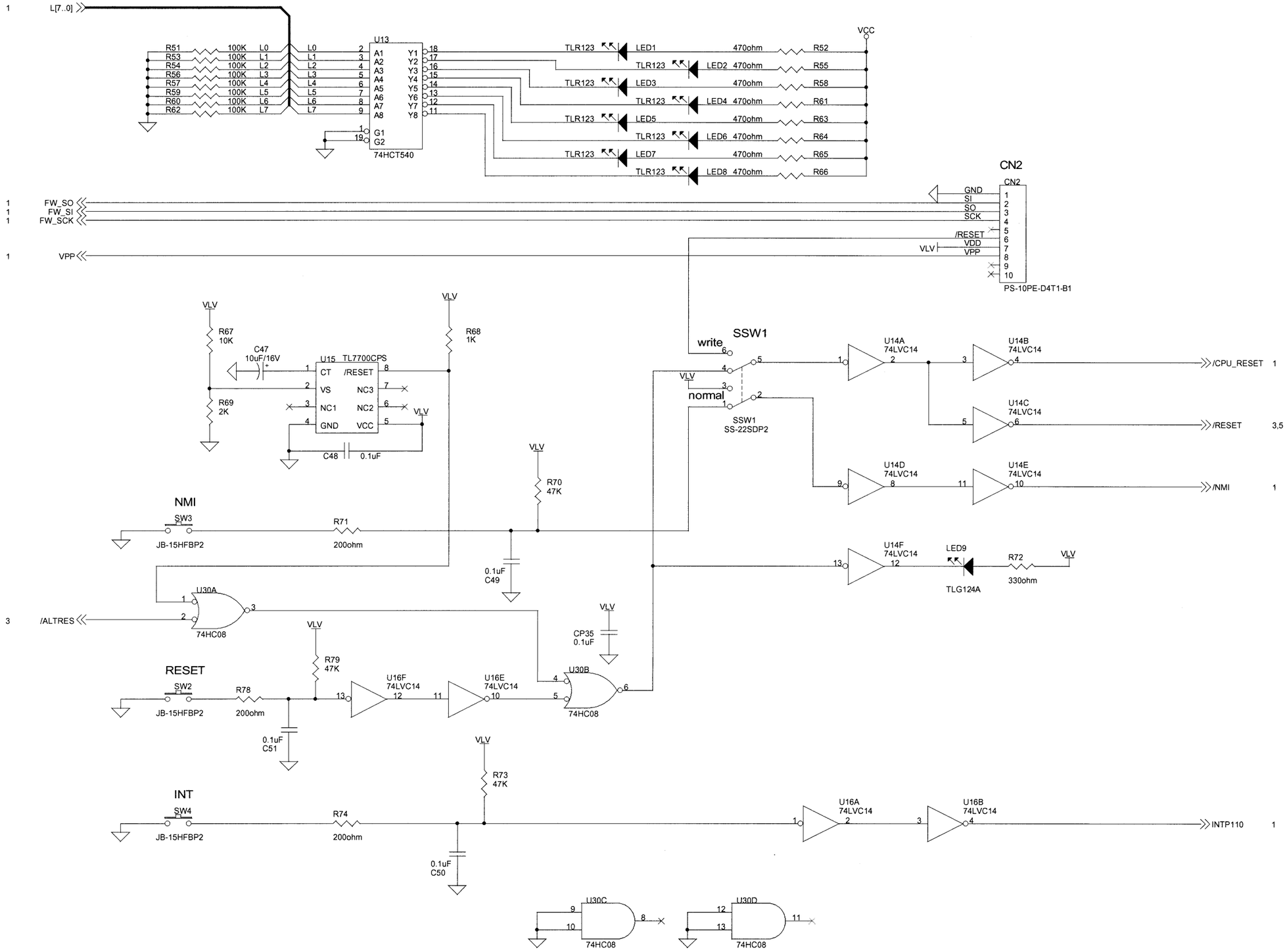


図4－35　TU-V850E/MA1（5/8）

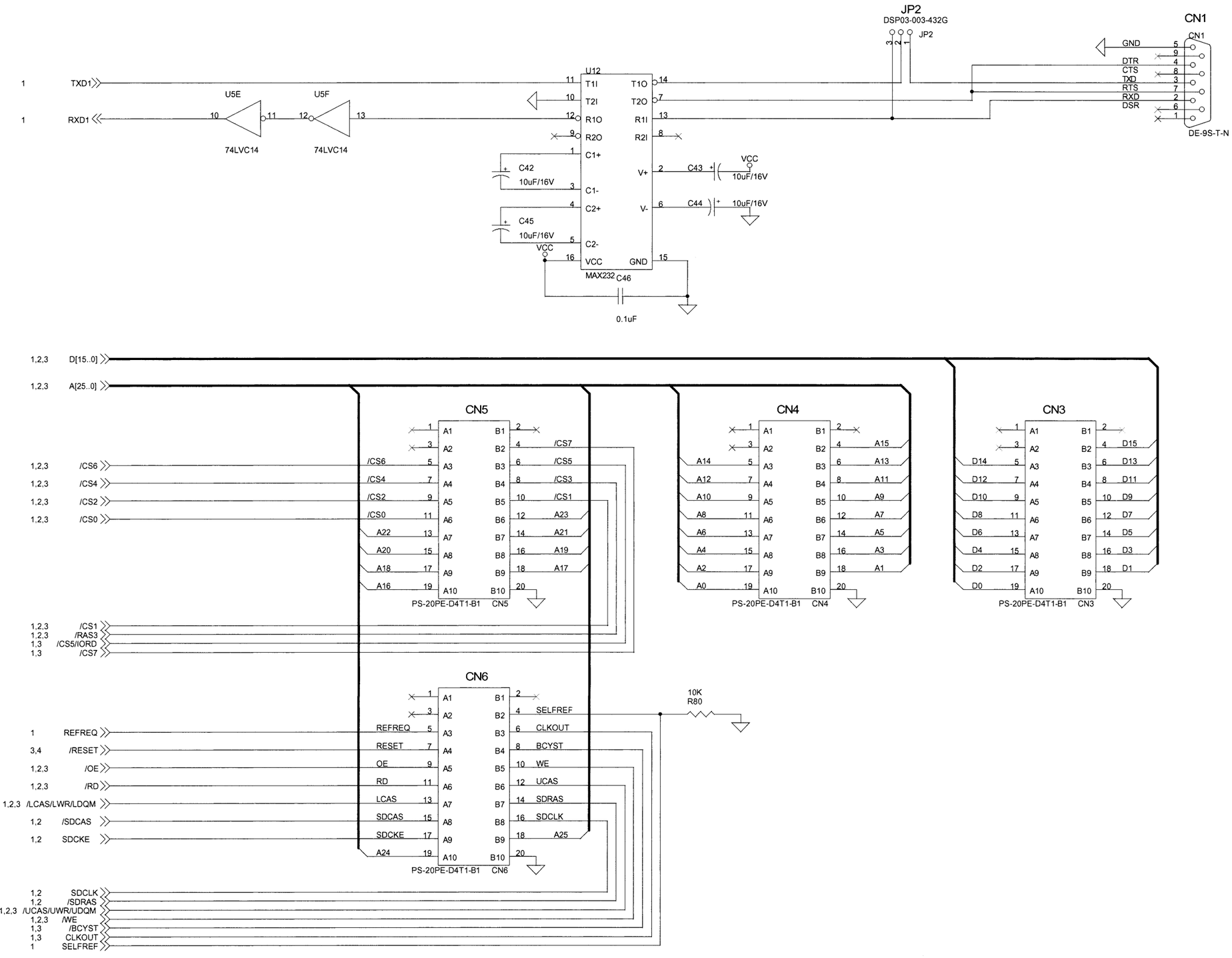


図4－35　TU-V850E/MA1（6/8）

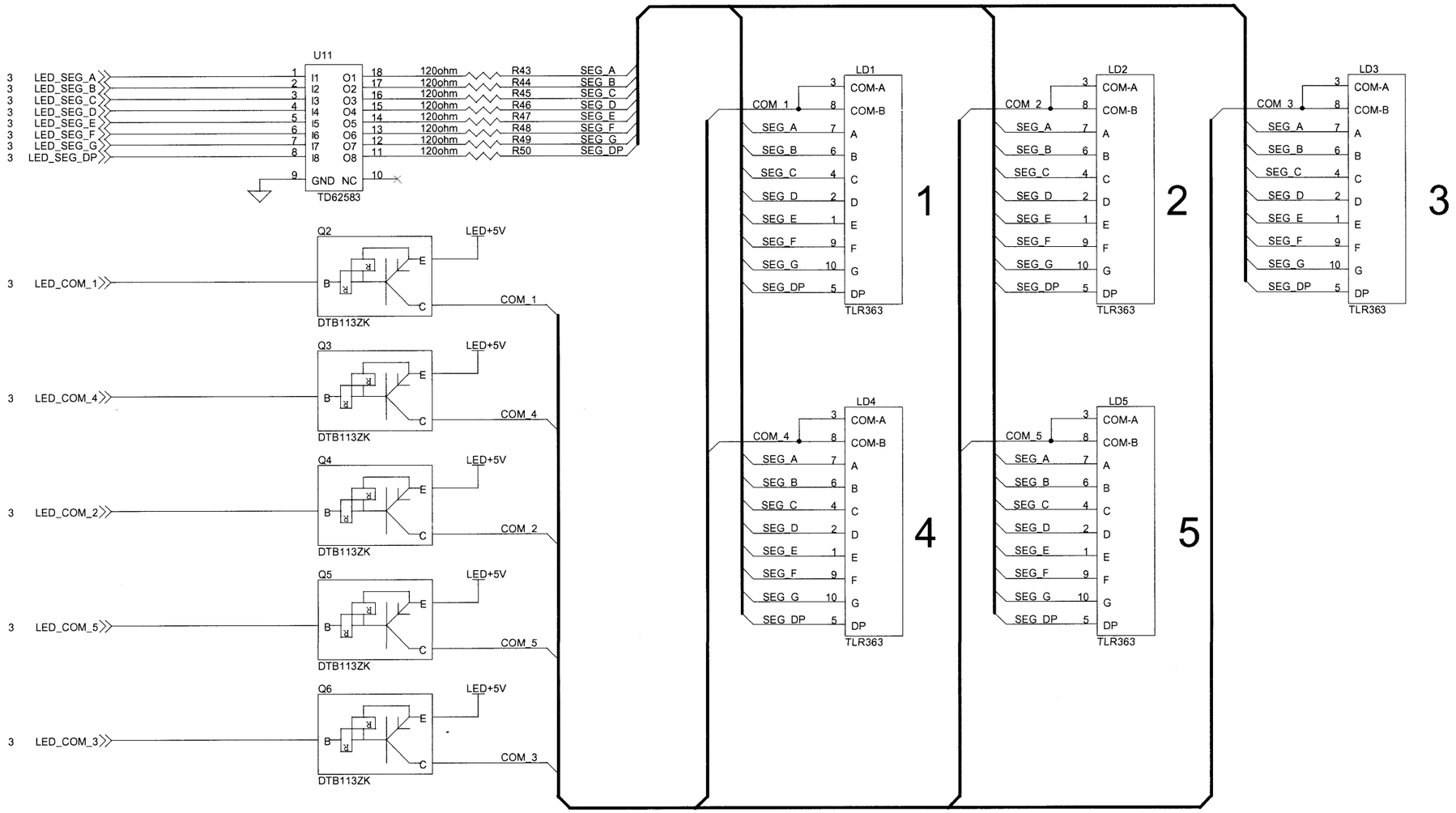


図4－35　TU-V850E/MA1（7/8）

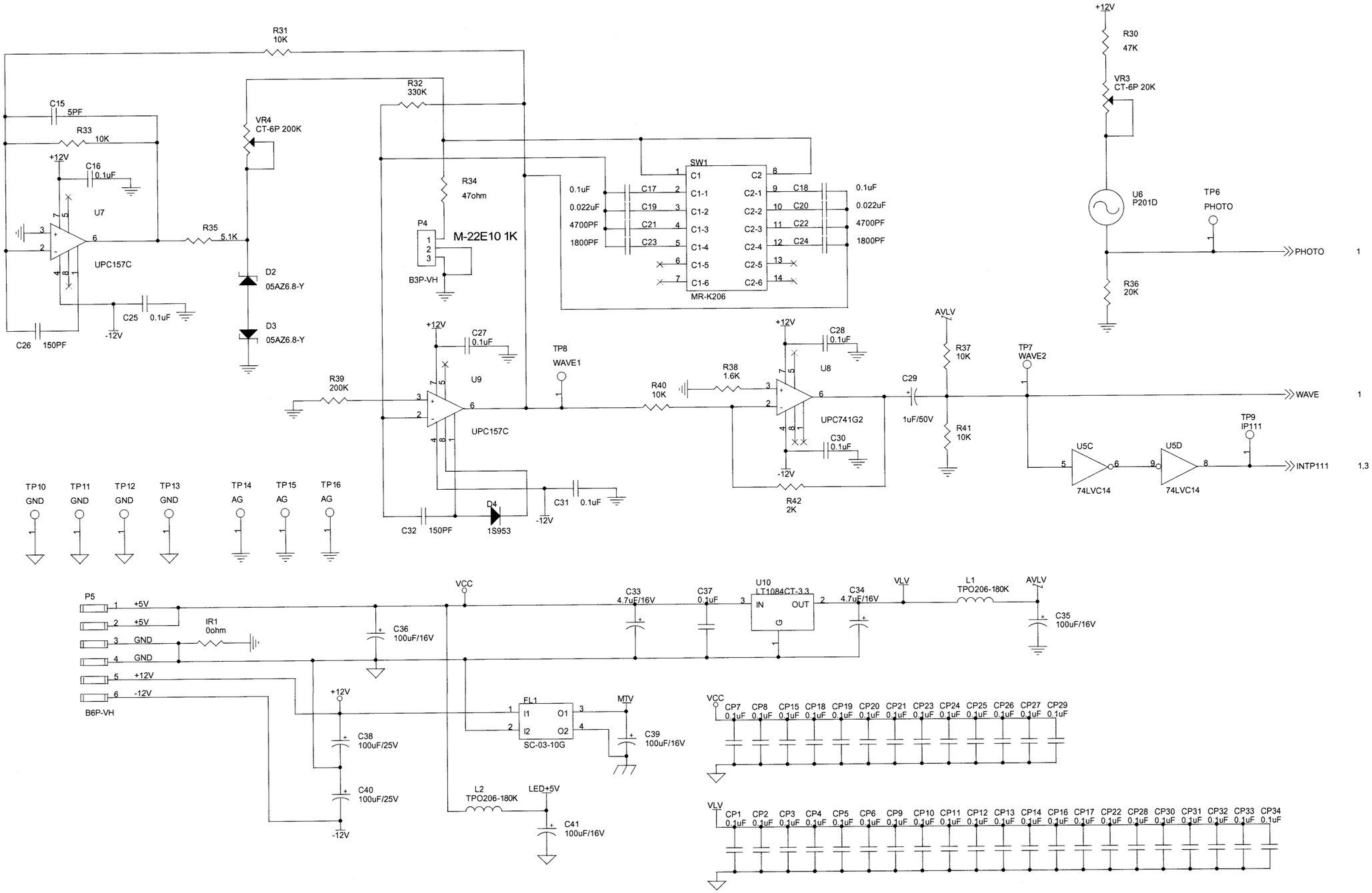
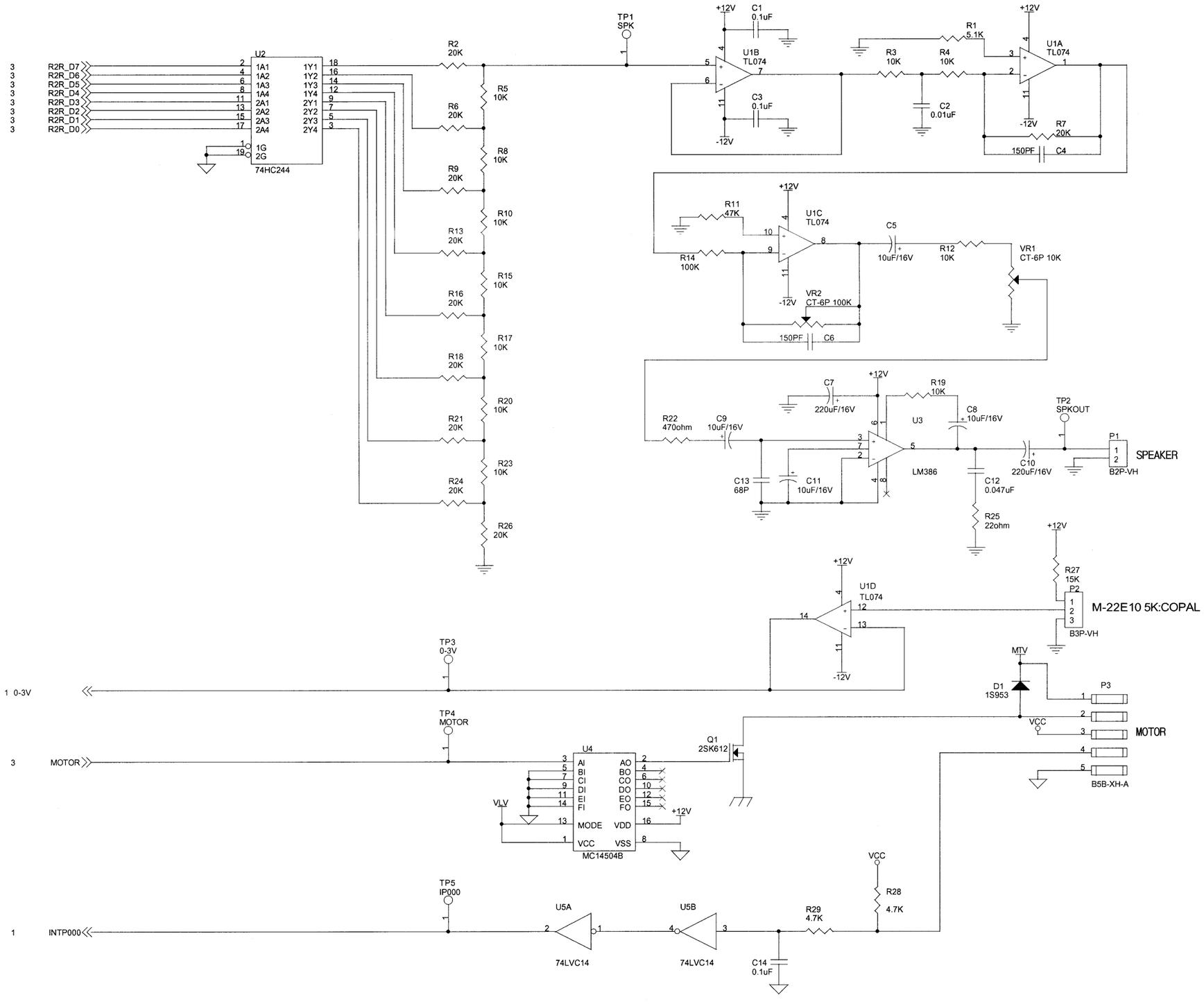


図4－35　TU-V850E/MA1（8/8）



付録A　総合索引

A. 1　50音で始まる語句の索引

A. 2　数字または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	・次の製品を削除 μPD703103, 703105, 703106, 703107, 70F3107 ・次の製品を追加 μPD703103A, 703105A, 703106A, 703106A(A), 703107A, 703107A(A), 70F3107A, 70F3107A(A)	全般
	1. 4　**オーダ情報**　記述変更	第1章 V850E/MA1の概要
	1. 5　**端子接続図**（Top View）　を追加	
	1. 6　**内部ブロック図**　を追加	
	2. 6　SDRAM**との接続**　V850E/MA1**のアドレス端子を変更**	第2章 バス・インタフェース接続回路例（1）
	図4－5　**システム・ウエイト・コントロール・レジスタ**（VSWC）**の設定**　を変更	第4章 アプリケーション例
	4. 2. 3　**シングルチップ・モード**0**での動作例**（STEP1）　**【プログラム例】**　VSWC レジスタの設定値を変更	
	図4－35　TU-V850E/MA1　SDRAM**と接続する**V850E/MA1**のアドレス端子を変更**	

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］V850E/MA1　アプリケーション・ノート　ハードウエア編
（U15179JJ2V0AN00（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（　　）					
（　　）					

キリトリ

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