To our customers,

## Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

# Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

RENESAS

# Application Note

# V850E/MA1™

## 32-Bit Single-Chip Microcontrollers

## Hardware

---

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

Document No. U15179EJ2V0AN00 (2nd edition)
Date Published May 2002 N CP(K)

**[MEMO]**

## NOTES FOR CMOS DEVICES

### ① PRECAUTION AGAINST ESD FOR SEMICONDUCTORS

Note:

Strong electric field, when exposed to a MOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop generation of static electricity as much as possible, and quickly dissipate it once, when it has occurred. Environmental control must be adequate. When it is dry, humidifier should be used. It is recommended to avoid using insulators that easily build static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work bench and floor should be grounded. The operator should be grounded using wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions need to be taken for PW boards with semiconductor devices on it.

### ② HANDLING OF UNUSED INPUT PINS FOR CMOS

Note:

No connection for CMOS device inputs can be cause of malfunction. If no connection is provided to the input pins, it is possible that an internal input level may be generated due to noise, etc., hence causing malfunction. CMOS devices behave differently than Bipolar or NMOS devices. Input levels of CMOS devices must be fixed high or low by using a pull-up or pull-down circuitry. Each unused pin should be connected to $V_{DD}$ or GND with a resistor, if it is considered to have a possibility of being an output pin. All handling related to the unused pins must be judged device by device and related specifications governing the devices.

### ③ STATUS BEFORE INITIALIZATION OF MOS DEVICES

Note:

Power-on does not necessarily define initial status of MOS device. Production process of MOS does not define the initial operation status of the device. Immediately after the power source is turned ON, the devices with reset function have not yet been initialized. Hence, power-on does not guarantee out-pin levels, I/O settings or contents of registers. Device is not initialized until the reset signal is received. Reset operation must be executed immediately after power-on for devices having reset function.

The export of these products from Japan is regulated by the Japanese government. The export of some or all of these products may be prohibited without governmental license. To export or re-export some or all of these products from a country other than Japan may also be prohibited without a license from that country. Please call an NEC sales representative.

| | |
|---|---|
| License not needed: | μPD703103A, 70F3107A, 70F3107A(A) |
| The customer must judge the need for license: | μPD703105A, 703106A, 703106A(A), 703107A, 703107A(A) |

# Regional Information

Some information contained in this document may vary from country to country. Before using any NEC product in your application, please contact the NEC office in your country to obtain a list of authorized representatives and distributors. They will verify:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, registered trademarks, export restrictions, and other legal issues may also vary from country to country.


**NEC Electronics Inc. (U.S.)**
Santa Clara, California
Tel: 408-588-6000
800-366-9782
Fax: 408-588-6130
800-729-9288

**NEC do Brasil S.A.**
Electron Devices Division
Guarulhos-SP, Brasil
Tel: 11-6462-6810
Fax: 11-6462-6829

**NEC Electronics (Europe) GmbH**
Duesseldorf, Germany
Tel: 0211-65 03 01
Fax: 0211-65 03 327

- **Sucursal en España**
  Madrid, Spain
  Tel: 091-504 27 87
  Fax: 091-504 28 60

- **Succursale Française**
  Vélizy-Villacoublay, France
  Tel: 01-30-67 58 00
  Fax: 01-30-67 58 99

- **Filiale Italiana**
  Milano, Italy
  Tel: 02-66 75 41
  Fax: 02-66 75 42 99

- **Branch The Netherlands**
  Eindhoven, The Netherlands
  Tel: 040-244 58 45
  Fax: 040-244 45 80

- **Branch Sweden**
  Taeby, Sweden
  Tel: 08-63 80 820
  Fax: 08-63 80 388

- **United Kingdom Branch**
  Milton Keynes, UK
  Tel: 01908-691-133
  Fax: 01908-670-290

**NEC Electronics Hong Kong Ltd.**
Hong Kong
Tel: 2886-9318
Fax: 2886-9022/9044

**NEC Electronics Hong Kong Ltd.**
Seoul Branch
Seoul, Korea
Tel: 02-528-0303
Fax: 02-528-4411

**NEC Electronics Shanghai, Ltd.**
Shanghai, P.R. China
Tel: 021-6841-1138
Fax: 021-6841-1137

**NEC Electronics Taiwan Ltd.**
Taipei, Taiwan
Tel: 02-2719-2377
Fax: 02-2719-5951

**NEC Electronics Singapore Pte. Ltd.**
Novena Square, Singapore
Tel: 253-8311
Fax: 250-3583


J02.4

**Major Revisions in This Edition**

| Page | Description |
|---|---|
| Throughout | • Deletion of the following product names<br>μPD703103, 703105, 703106, 703107, and 70F3107<br>• Addition of the following product names:<br>μPD703103A, 703105A, 703106A, 703106A(A), 703107A, 703107A(A), 70F3107A, and 70F3107A(A) |
| p. 19 | Change of description in **1.4 Ordering Information** |
| p. 20 | Addition of **1.5 Pin Configuration (Top View)** |
| p. 23 | Addition of **1.6 Internal Block Diagram** |
| p. 55 | Change of V850E/MA1 address pin in **2.6 Connection with SDRAM** |
| p. 100 | Change of **Figure 4-5. Settings in System Wait Control Register (VSWC)** |
| p. 101 | Change of setting value in VSWC register in **4.2.3 Operaiton example in single-chip mode 0 (STEP 1) [program example]** |
| pp. 91, 179, 181, and 185 | Change of **Figure 4-35. TU-V850E/MA1** |

**The mark ★ shows major revised points.**

# INTRODUCTION

**Readers**

This application note is intended for users who wish to understand the functions of the V850E/MA1 to design application systems using the V850E/MA1.

The target products are as follows:

- Standard models: μPD703103A, 703105A, 703106A, 703107A, and 70F3107A
- Special models: μPD703106A(A)[Note], 703107A(A)[Note], and 70F3107A(A)

**Note** Under development

**Purpose**

The purpose of this application note is to help the user understand the composition of a system that uses the V850E/MA1, using the V850E/MA1 training unit (TU-V850E/MA1) as an example.

**Organization**

This application note is broadly divided into the following sections.

- V850E/MA1 introduction
- Bus interface connection circuit example (1) (connection example of directly connectable memory)
- Bus interface connection circuit example (2) (connection example using additional circuits)
- Application examples

**How to Read This Manual**

It is assumed that the readers of this application note have general knowledge in the fields of electrical engineering, logic circuits, and microcontrollers.

**Cautions 1. Application examples in this manual are intended for the "standard" quality models for general-purpose electronic systems. When using an example in this manual for an application that requires the "special" quality grade, evaluate each component and circuit to be actually used to see if they satisfy the required quality standard.**

**2. To use this manual for the products of special grade, take it as follows:**

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

- For details of the electrical specifications of the V850E/MA1
  →Refer to the data sheet (separately provided).

- For details of a hardware function of the V850E/MA1
  →Refer to the **V850E/MA1 User's Manual Hardware** (separately provided).

- For details of an instruction function of the V850E/MA1
  →Refer to the **V850E1 User's Manual Architecture** (separately provided).

**Conventions**

| | |
|---|---|
| Data significance: | Higher digits on the left and lower digits on the right |
| Active low representation: | $\overline{xxx}$ (overscore over pin or signal name) or /xxx ("/" before signal name) |
| Memory map address: | Top: higher, bottom: lower |
| **Note:** | Footnote for item marked with **Note** in the text |
| **Caution:** | Information requiring particular attention |
| **Remark:** | Supplementary information |
| Numeric representation: | Binary ... xxxx or xxxxB<br>Decimal ... xxxx<br>Hexadecimal ... xxxxH |
| Prefix indicating power of 2 (address space, memory capacity): | K (kilo) ... $2^{10}$ = 1,024<br>M (mega) ... $2^{20}$ = $1,024^2$<br>G (giga) ... $2^{30}$ = $1,024^3$ |

**Related documents**

The related documents indicated in this publication may include preliminary versions. However, preliminary versions are not marked as such.

**Document related to V850E/MA1**

| Document Name | Document No. |
|---|---|
| V850E1 User's Manual Architecture | U14559E |
| V850E/MA1 User's Manual Hardware | U14359E |
| $\mu$PD70F3107A, 70F3107A(A) Data Sheet | U15846E |
| $\mu$PD703103A, 703105A, 703106A, 703106A(A), 703107A, 703107A(A) Data Sheet | U15578E |
| V850E/MA1 Application Note Hardware | This manual |
| V850 Series™ User's Manual Flash Memory Self Programming Library | U15673E |

**Document related to development tools (user's manuals)**

| Document Name | | Document No. |
|---|---|---|
| IE-V850E-MC, IE-V850E-MC-A (In-circuit emulator) | | U14487E |
| IE-703107-MC-EM1 (In-circuit emulator option board) | | U14481E |
| CA850 (Ver.2.30 or later) (C compiler package) | Operation | U14568E |
| | C Language | U14566E |
| | Project Manager | U14569E |
| | Assembly Language | U14567E |
| CA850 (Ver.2.40 or later) (C compiler package) | Operation | U15024E |
| | C Language | U15025E |
| | Project Manager | U15026E |
| | Assembly Language | U15027E |
| ID850 (Ver.2.40) (Integrated debugger) | Operation Windows™ Based | U15181E |
| SM850 (Ver.2.40) (System simulator) | Operation Windows Based | U15182E |
| SM850 (Ver.2.00 or later) (System simulator) | External Part User Open Interface Specification | U14873E |
| RX850 (Ver.3.13 or later) (Real-time OS) | Basics | U13430E |
| | Installation | U13410E |
| | Technical | U13431E |
| RX850 Pro (Ver.3.13) (Real-time OS) | Basics | U13773E |
| | Installation | U13774E |
| | Technical | U13772E |
| RD850 (Ver.3.01) (Task debugger) | | U13737E |
| RD850 Pro (Ver.3.01) (Task debugger) | | U13916E |
| AZ850 (Ver.3.0) (System performance analyzer) | | U14410E |
| PG-FP3 (Flash memory programmer) | | U13502E |
| PG-FP4 (Flash memory programmer) | | U15260E |

**CONTENTS**

## LIST OF FIGURES (1/3)

## LIST OF FIGURES (2/3)

## LIST OF FIGURES (3/3)

## LIST OF TABLES

# CHAPTER 1 V850E/MA1 INTRODUCTION

The V850E/MA1 is a product of NEC's single-chip microcontroller "V850 Series". This chapter gives a simple outline of the V850E/MA1.

## 1.1 Outline

The V850E/MA1 is a 32-bit single-chip microcontroller that integrates the V850E1 CPU, which is a 32-bit RISC-type CPU core for ASIC, newly developed as the CPU core central to system LSI for the current age of system-on-chip. This device incorporates ROM, RAM, and various peripheral functions such as memory controllers, a DMA controller, real-time pulse unit, serial interfaces, and an A/D converter for realizing high-capacity data processing and sophisticated real-time control.

**(1) V850E1 CPU**

The V850E1 CPU is a CPU core that enhances the external bus interface performance of the V850 CPU, which is the CPU core integrated in the V850 Series, and has added instructions supporting high-level languages, such as C-language switch statement processing, table lookup branching, stack frame creation/deletion, and data conversion. This enhances the performance of both data processing and control.

It is possible to use the software resources of the V850 CPU integrated system since the instruction codes of the V850E1 are upwardly compatible at the object code level with those of the V850 CPU.

**(2) External memory interface function**

The V850E/MA1 features various on-chip external memory interfaces including separately configured address (26 bits) and data (16 bits) buses, and SDRAM and ROM interfaces, as well as on-chip memory controllers that can be directly linked to EDO DRAM, page ROM, etc., thereby raising system performance and reducing the number of parts needed for application systems.

Also, through the DMA controller, CPU internal calculations and data transfers can be performed simultaneously with transfers to and from the external memory, so it is possible to process large volumes of image data or voice data, etc., and through high-speed execution of instructions using internal ROM and RAM, motor control, communications control and other real time control tasks can be realized simultaneously.

★ **(3) On-chip flash memory (μPD70F3107A)**

The on-chip flash memory version (μPD70F3107A) has on-chip flash memory, which is capable of high-speed access, and since it is possible to rewrite a program with the V850E/MA1 mounted as is in the application system, system development time can be reduced and system maintainability after shipping can be markedly improved.

**(4) A full range of middleware and development environment products**

The V850E/MA1 can execute middleware such as JPEG, JBIG, and MH/MR/MMR at high speed. Also, middleware that enables voice recognition, voice synthesis, and other such processing is available, and by including these middleware programs, a multimedia system can be easily realized.

A development environment system that includes an optimized C compiler, debugger, in-circuit emulator, simulator, system performance analyzer, and other elements is also available.

## 1.2 Features

○ Number of instructions: 83

○ Minimum instruction execution time: 20 ns (at internal 50 MHz operation)

○ General registers: 32 bits × 32

○ Instruction set: V850E1 CPU
Signed multiplication (16 bits × 16 bits → 32 bits or 32 bits × 32 bits → 64 bits): 1 to 2 clocks
Saturated operation instructions (with overflow/underflow detection function)
32-bit shift instructions: 1 clock
Bit manipulation instructions
Load/store instructions with long/short format
Signed load instructions

○ Memory space: 256 MB linear address space (common program/data use)
Chip select output function: 8 spaces
Memory block division function: 2, 4, 8 MB/block
Programmable wait function
Idle state insertion function

○ External bus interface: 16-bit data bus (address/data separated)
16-/8-bit bus sizing function
Bus hold function
External wait function
Address setup wait function
Endian control function

★ ○ Internal memory

| Part Number | Internal ROM | Internal RAM |
|---|---|---|
| μPD703103A | None | 4 KB |
| μPD703105A | 128 KB (Mask ROM) | 4 KB |
| μPD703106A | 128 KB (Mask ROM) | 10 KB |
| μPD703107A | 256 KB (Mask ROM) | 10 KB |
| μPD70F3107A | 256 KB (Flash memory) | 10 KB |

○ Interrupts/exceptions: External interrupts: 25 (including NMI)
Internal interrupts: 33 sources
Exceptions: 1 source
Eight levels of priorities can be set.

○ Memory access controller
- ○ DRAM controller (Compatible with EDO DRAM and SDRAM)
- ○ Page-ROM controller

- DMA controller:
  - 4 channels
  - Transfer units: 8 bits/16 bits
  - Maximum transfer count: 65,536 ($2^{16}$)
  - Transfer type: Flyby (1-cycle)/2-cycle
  - Transfer mode: Single/Single step/Block
  - Transfer target: Memory ↔ memory, memory ↔ I/O
  - Transfer request: External request/On-chip peripheral I/O/ Software
  - DMA transfer terminate (terminal count) output signal
  - Next address setting function
- I/O lines:
  - Input ports: 9
  - I/O ports: 106
- Real-time pulse unit:
  - 16-bit timer/event counter: 4 channels
    - 16-bit timers: 4
    - 16-bit capture/compare registers: 8
  - 16-bit interval timer: 4 channels
- Serial interfaces (SIO):
  - Asynchronous serial interface (UART)
  - Clocked serial interface (CSI)
    - CSI/UART: 2 channels
    - UART: 1 channel
    - CSI: 1 channel
- A/D converter:
  - 10-bit resolution A/D converter: 8 channels
- PWM (Pulse Width Modulation):
  - 8-/9-/10-/12-bit resolution PWM: 2 channels
- Clock generator:
  - A ×10 multiplication function through a PLL clock synthesizer
  - Divide-by-two function through an external clock input
- Power save function:
  - HALT/IDLE/software STOP mode
- Package:
  - 144-pin plastic LQFP (fine pitch) (20 × 20)
  - 161-pin plastic FBGA (13 × 13)
- CMOS technology:
  - All static circuits

## 1.3 Applications

Ink-jet printer, facsimile, digital still camera, DVD player, video printer, PPC, information equipment, etc.

## ★ 1.4 Ordering Information

| Part Number | Package | Quality Grade |
|---|---|---|
| μPD703103AGJ-UEN | 144-pin plastic LQFP (fine pitch) (20 × 20) | Standard (for general-purpose electronic systems) |
| μPD703105AGJ-×××-UEN | 144-pin plastic LQFP (fine pitch) (20 × 20) | Standard (for general-purpose electronic systems) |
| μPD703106AGJ-×××-UEN | 144-pin plastic LQFP (fine pitch) (20 × 20) | Standard (for general-purpose electronic systems) |
| μPD703107AGJ-×××-UEN | 144-pin plastic LQFP (fine pitch) (20 × 20) | Standard (for general-purpose electronic systems) |
| μPD70F3107AGJ-UEN | 144-pin plastic LQFP (fine pitch) (20 × 20) | Standard (for general-purpose electronic systems) |
| μPD703106AF1-×××-EN4 | 161-pin plastic FBGA (13 × 13) | Standard (for general-purpose electronic systems) |
| μPD703107AF1-×××-EN4 | 161-pin plastic FBGA (13 × 13) | Standard (for general-purpose electronic systems) |
| μPD70F3107AF1-EN4 | 161-pin plastic FBGA (13 × 13) | Standard (for general-purpose electronic systems) |
| μPD703106AGJ(A)-×××-UEN[Note] | 144-pin plastic LQFP (fine pitch) (20 × 20) | Special (for high-reliability electronic systems) |
| μPD703107AGJ(A)-×××-UEN[Note] | 144-pin plastic LQFP (fine pitch) (20 × 20) | Special (for high-reliability electronic systems) |
| μPD70F3107AGJ(A)-UEN | 144-pin plastic LQFP (fine pitch) (20 × 20) | Special (for high-reliability electronic systems) |

**Note** Under development

**Remark** ××× indicates ROM code suffix.

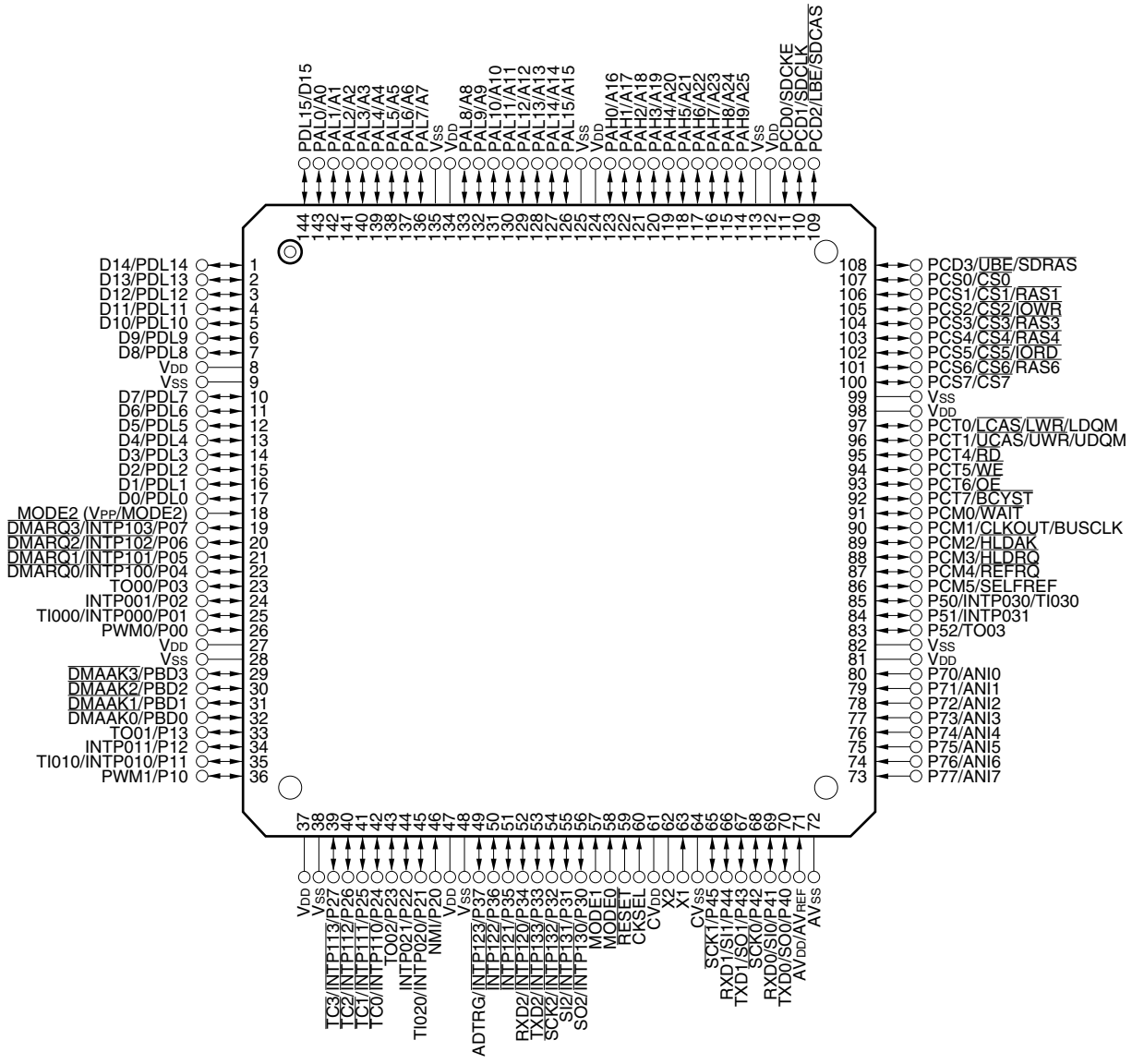
The μPD703106A, 703107A, and 70F3107A do not differ from the μPD703106A(A), 703107A(A), and 70F3107A(A) except the quality grade.

Please refer to **Quality Grades on NEC Semiconductor Devices** (Document No. C11531E) published by NEC Corporation to know the specification of quality grade on the devices and its recommended applications.

## ⋆ 1.5 Pin Configuration (Top View)

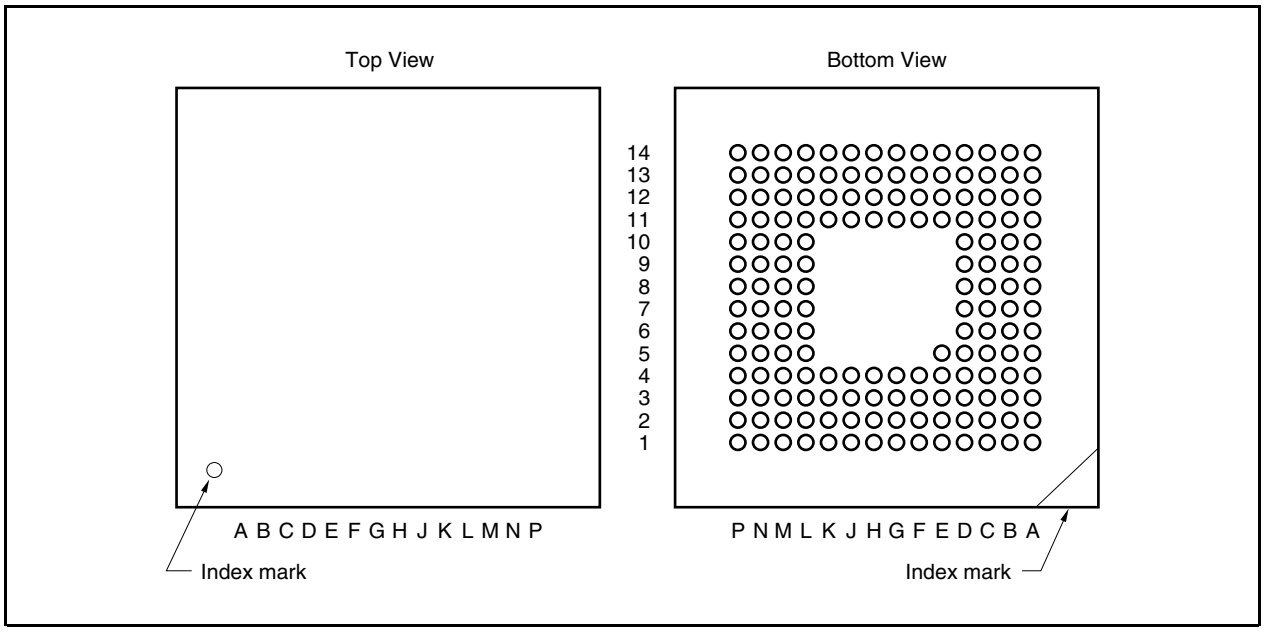
- 144-pin plastic LQFP (fine pitch) (20 × 20)

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




**Remark** Items in parentheses are pin names in the μPD70F3107A.

- 161-pin plastic FBGA (13 × 13)
  μPD703106AF1-×××-EN4
  μPD703107AF1-×××-EN4
  μPD70F3107AF1-EN4




(1/2)

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

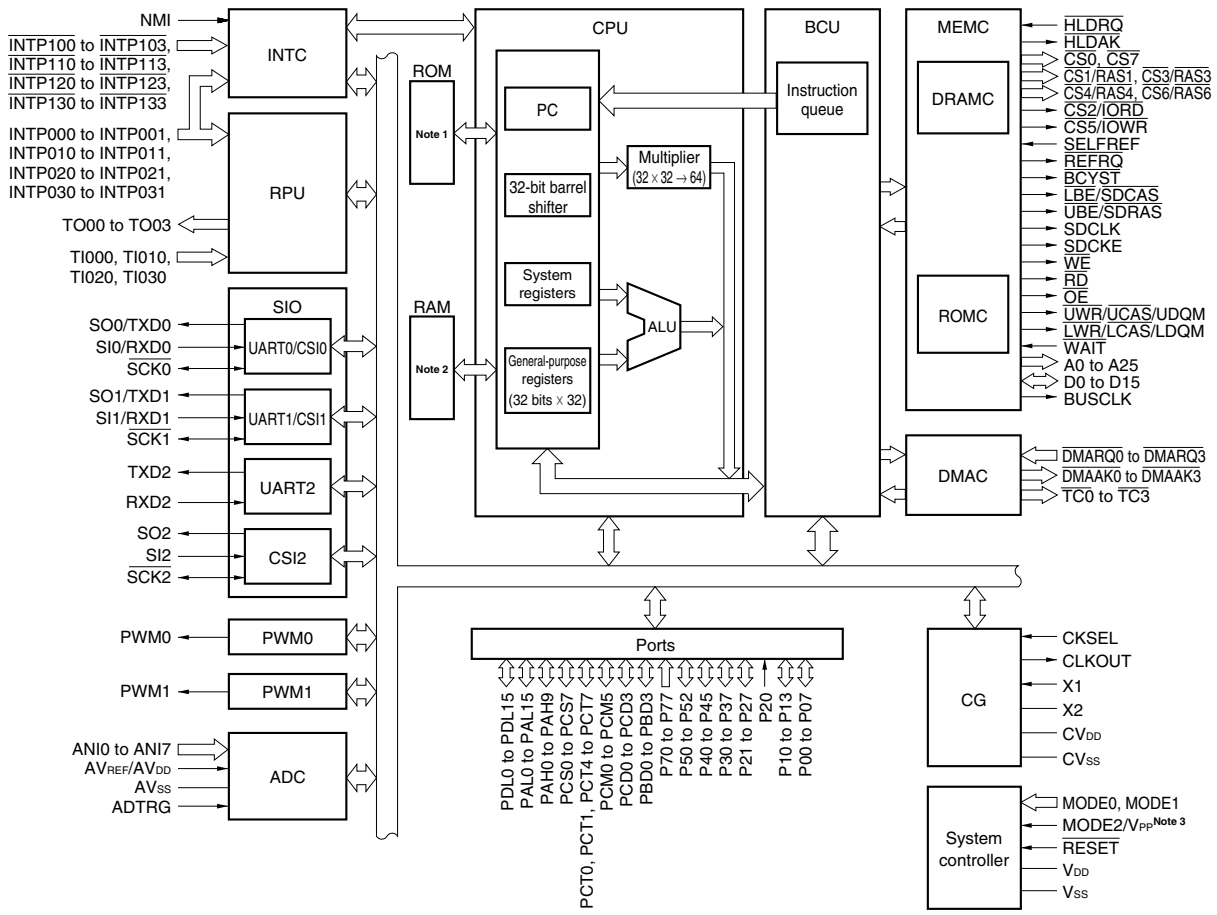
(2/2)

| Pin Number | Pin Name | Pin Number | Pin Name | Pin Number | Pin Name |
|---|---|---|---|---|---|
| E5 | – | J12 | TI030/INTP030/P50 | M10 | $\overline{SCK1}$/P45 |
| E11 | $\overline{CS6}$/$\overline{RAS6}$/PCS6 | J13 | SELFREF/PCM5 | M11 | TXD0/SO0/P40 |
| E12 | $\overline{CS4}$/$\overline{RAS4}$/PCS4 | J14 | INTP031/P51 | M12 | ANI6/P76 |
| E13 | $\overline{CS7}$/PCS7 | K1 | PWM0/P00 | M13 | ANI5/P75 |
| E14 | $V_{SS}$ | K2 | $V_{SS}$ | M14 | – |
| F1 | D2/PDL2 | K3 | $\overline{DMAAK1}$/PBD1 | N1 | – |
| F2 | D3/PDL3 | K4 | $\overline{DMAAK3}$/PBD3 | N2 | PWM1/P10 |
| F3 | D4/PDL4 | K11 | ANI1/P71 | N3 | $\overline{TC3}$/$\overline{INTP113}$/P27 |
| F4 | $V_{DD}$ | K12 | ANI0/P70 | N4 | $\overline{TC0}$/$\overline{INTP110}$/P24 |
| F11 | $\overline{RD}$/PCT4 | K13 | $V_{SS}$ | N5 | NMI/P20 |
| F12 | $V_{DD}$ | K14 | $V_{DD}$ | N6 | ADTRG/$\overline{INTP123}$/P37 |
| F13 | $\overline{LCAS}$/$\overline{LWR}$/LDQM/PCT0 | L1 | – | N7 | TXD2/$\overline{INTP133}$/P33 |
| F14 | $\overline{UCAS}$/$\overline{UWR}$/UDQM/PCT1 | L2 | $\overline{DMAAK2}$/PBD2 | N8 | SO2/$\overline{INTP130}$/P30 |
| G1 | MODE2 (MODE2/$V_{PP}$) | L3 | TI010/INTP010/P11 | N9 | X2 |
| G2 | $\overline{DMARQ3}$/$\overline{INTP103}$/P07 | L4 | $\overline{DMAAK0}$/PBD0 | N10 | $CV_{SS}$ |
| G3 | D0/PDL0 | L5 | TO02/P23 | N11 | $\overline{SCK0}$/P42 |
| G4 | D6/PDL6 | L6 | $V_{DD}$ | N12 | $AV_{DD}$/$AV_{REF}$ |
| G11 | $\overline{WAIT}$/PCM0 | L7 | $\overline{INTP122}$/P36 | N13 | $AV_{SS}$ |
| G12 | $\overline{WE}$/PCT5 | L8 | SI2/$\overline{INTP131}$/P31 | N14 | – |
| G13 | $\overline{BCYST}$/PCT7 | L9 | $\overline{RESET}$ | P1 | $V_{DD}$ |
| G14 | $\overline{OE}$/PCT6 | L10 | TXD1/SO1/P43 | P2 | $V_{SS}$ |
| H1 | $\overline{DMARQ2}$/$\overline{INTP102}$/P06 | L11 | ANI7/P77 | P3 | $\overline{TC1}$/$\overline{INTP111}$/P25 |
| H2 | $\overline{DMARQ1}$/$\overline{INTP101}$/P05 | L12 | ANI4/P74 | P4 | INTP021/P22 |
| H3 | $\overline{DMARQ0}$/$\overline{INTP100}$/P04 | L13 | ANI3/P73 | P5 | – |
| H4 | D1/PDL1 | L14 | ANI2/P72 | P6 | $\overline{INTP121}$/P35 |
| H11 | $\overline{REFRQ}$/PCM4 | M1 | – | P7 | $\overline{SCK2}$/$\overline{INTP132}$/P32 |
| H12 | $\overline{HLDRQ}$/PCM3 | M2 | INTP011/P12 | P8 | MODE1 |
| H13 | $\overline{HLDAK}$/PCM2 | M3 | TO01/P13 | P9 | $CV_{DD}$ |
| H14 | CLKOUT/BUSCLK/PCM1 | M4 | $\overline{TC2}$/$\overline{INTP112}$/P26 | P10 | X1 |
| J1 | TO00/P03 | M5 | TI020/INTP020/P21 | P11 | – |
| J2 | TI000/INTP000/P01 | M6 | $V_{SS}$ | P12 | RXD1/SI1/P44 |
| J3 | $V_{DD}$ | M7 | RXD2/$\overline{INTP120}$/P34 | P13 | RXD0/SI0/P41 |
| J4 | INTP001/P02 | M8 | MODE0 | P14 | – |
| J11 | TO03/P52 | M9 | CKSEL | | |

**Remarks 1.** Leave the A1, A5, A10, B1, B14, C1, C14, D14, E5, L1, M1, M14, N1, N14, P5, P11, and P14 pins open.

**2.** Items in parentheses are pin names in the μPD70F3107A.

## ★ 1.6 Internal Block Diagram




**Notes 1.** μPD703103A: ROMless
μPD703105A, 703106A: 128 KB (mask ROM)
μPD703107A: 256 KB (mask ROM)
μPD70F3107A: 256 KB (flash memory)

**2.** μPD703103A, 703105A: 4 KB
μPD703106A, 703107A, 70F3107A: 10 KB

**3.** Valid for μPD70F3107A only

# CHAPTER 2 BUS INTERFACE CONNECTION CIRCUIT EXAMPLES (1)

This chapter describes the memory directly connected to the V850E/MA1. The following approach was used to configure the circuit examples and related register settings as shown below.

(1) V850E/MA1's internal system clock and bus cycle frequency is 50 MHz.

**Remark** Wait settings and idle settings other than 50 MHz are described in **Table 2-2 List of Wait/Idle Settings and Bus Clocks for SRAM Connection**.

(2) Programmable wait is used for wait control ($\overline{\text{WAIT}}$ pin: not used (pulled up)

(3) External bus master is not connected ($\overline{\text{HLDRQ}}$ pin: not used (pulled up)

(4) Set 8xH to the BCP register and any value to the BEC register.
- Bus cycle period: Normal
- Operation of $\overline{\text{IORD}}$ and $\overline{\text{IOWR}}$: Determine as necessary
- Endian setting: Determine according to system (target software)

(5) Examples of memory to be connected use 3 V devices except in **2.1 Connection with SRAM**. However, a 5 V device can be connected as long it meets the DC characteristics (TTL level) required when connecting to the V850E/MA1.

Table 2-1 below compares the DC characteristics of the V850E/MA1 and a 5 V SRAM (such as the $\mu$PD434008A, which can be directly connected to the V850E/MA1).

**Table 2-1. DC Characteristics of 5 V SRAM and V850E/MA1**

| Operation of V850E/MA1 | V850E/MA1 Bus-Related Pin I/O Characteristics | I/O Characteristics of SRAM ($\mu$PD434008A) |
|---|---|---|
| During read | $V_{IH}$: 2.0 V (MIN.)<br>5.5 V (MAX.)<br>$V_{IL}$: 0.2 $V_{DD}$ (MAX.) | $V_{OH}$: 2.4 V (MIN.)<br><br>$V_{OL}$: 0.4 V (MAX.) |
| During write | $V_{OH}$: 0.8 $V_{DD}$ (MIN.)<br>$V_{OL}$: 0.45 V (MAX.) | $V_{IH}$: 2.2 V (MIN.)<br>$V_{IL}$: 0.8 V (MAX.) |

Note that it may be necessary to add a margin of several ns to actual circuits since the delay time from the PCB wiring has not been included.

## 2.1 Connection with SRAM

### 2.1.1 Connection with μPD434008A (example of 8-bit bus width)

The following is an example in which one SRAM (μPD434008ALE-12: 512 Kbits × 8 bits, 5 V) is connected as a 512 KB external memory space with 8-bit bus width).

**[Circuit configuration]**

- Internal system clock: 50 MHz
- Connected devices: μPD434008ALE-12 (1 device)
- $\overline{CS}$ signal used: $\overline{CS0}$
  Assigned to address range 0100000H to 017FFFFH (block 0) in external memory space. 0180000H to 01FFFFFH is used for the image.

**[Connection approach and caution points]**

Due to its 8-bit bus width, the SRAM data bus is connected to D0 to D7 in the V850E/MA1 and its address bus is connected to A0 to A18 in the V850E/MA1. Also, the SRAM's $\overline{CS}$ pin, $\overline{OE}$ pin, and $\overline{WE}$ pin are respectively connected to the V850E/MA1's $\overline{CS0}$ pin, $\overline{RD}$ pin, and $\overline{LWR}$ pin.

**[Register settings]**

- Block and device type used with $\overline{CS0}$: block 0, SRAM, external I/O
- Wait setting: 1 wait
- Address setup wait: 0 waits
- Idle states: 1 state

**Figure 2-1. Settings in Chip Area Select Control Register 0 (CSC0)**

| Bit | 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 |

$\overline{CS0}$ output when block 0 is accessed
Setting value in CSC0: xxxxxxxxxxxx0001B

**Figure 2-2. Settings in Bus Cycle Type Configuration Register 0 (BCT0)**

| Bit | 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 signal | | | $\overline{CS3}$ | | | | | $\overline{CS2}$ | | | $\overline{CS1}$ | | | | | $\overline{CS0}$ |

Memory controller enabled, SRAM, external I/O are specified for $\overline{CS0}$ space
Setting value in BCT0: x0xxx00xx0xx1000B

**Figure 2-3. Settings in Bus Size Configuration Register (BSC)**

| Bit | 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 signal | | $\overline{CS7}$ | | $\overline{CS6}$ | | $\overline{CS5}$ | | $\overline{CS4}$ | | $\overline{CS3}$ | | $\overline{CS2}$ | | $\overline{CS1}$ | | $\overline{CS0}$ |

8-bit data bus width is set for $\overline{CS0}$ space
Setting value in BSC: 0x0x0x0x0x0x0x00B

**Figure 2-4. Settings in Data Wait Control Register 0 (DWC0)**

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DWC0 | 0 | DW | DW | DW | 0 | DW | DW | DW | 0 | DW | DW | DW | 0 | DW | DW | DW |
| [FFFFF484H] | | 32 | 31 | 30 | | 22 | 21 | 20 | | 12 | 11 | 10 | | 02 | 01 | 00 |
| $\overline{CS}$n signal | | | $\overline{CS3}$ | | | | $\overline{CS2}$ | | | | $\overline{CS1}$ | | | | $\overline{CS0}$ | |

Number of waits set in $\overline{CS0}$ space: 1
Setting value in DWC0: 0xxx0xxx0xxx0001B

**Figure 2-5. Settings in Address Setup Wait Control Register (ASC)**

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ASC | AC | AC | AC | AC | AC | AC | AC | AC | AC | AC | AC | AC | AC | AC | AC | AC |
| [FFFFF48AH] | 71 | 70 | 61 | 60 | 51 | 50 | 41 | 40 | 31 | 30 | 21 | 20 | 11 | 10 | 01 | 00 |
| $\overline{CS}$n signal | $\overline{CS7}$ | | $\overline{CS6}$ | | $\overline{CS5}$ | | $\overline{CS4}$ | | $\overline{CS3}$ | | $\overline{CS2}$ | | $\overline{CS1}$ | | $\overline{CS0}$ | |

Number of address setup waits set in $\overline{CS0}$ space: 0
Setting value in ASC: xxxxxxxxxxxxxx00B

**Figure 2-6. Settings in Bus Cycle Control Register (BCC)**

| Bit | 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 |
| $\overline{\text{CS}}$n signal | | $\overline{\text{CS7}}$ | | $\overline{\text{CS6}}$ | | $\overline{\text{CS5}}$ | | $\overline{\text{CS4}}$ | | $\overline{\text{CS3}}$ | | $\overline{\text{CS2}}$ | | $\overline{\text{CS1}}$ | | $\overline{\text{CS0}}$ |

Number of idle states set in $\overline{\text{CS0}}$ space: 1
Setting value in BCC: xxxxxxxxxxxxxx01B

**Figure 2-7. Example of μPD434008ALE-12 Connection Circuit**

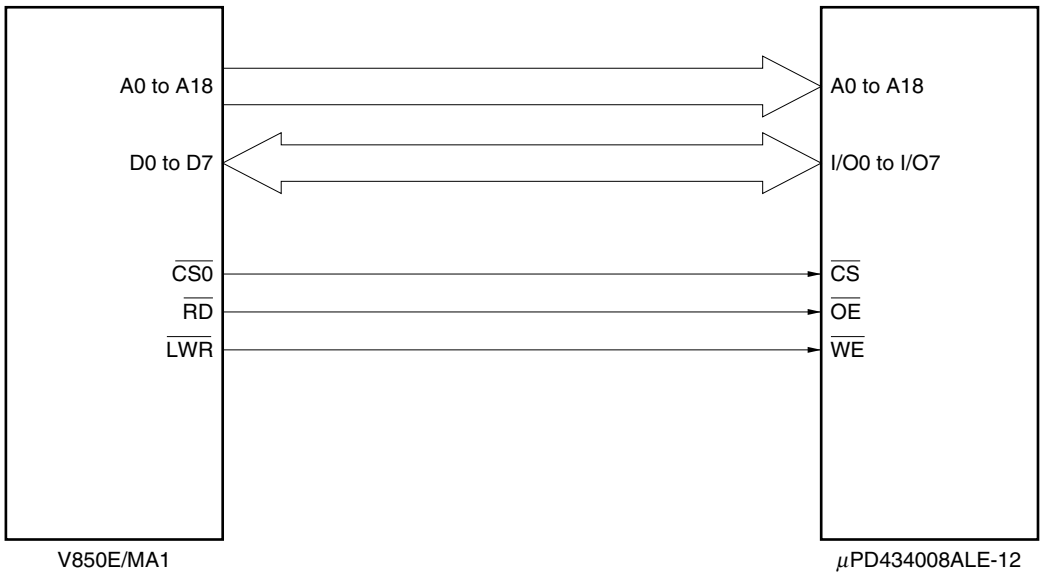
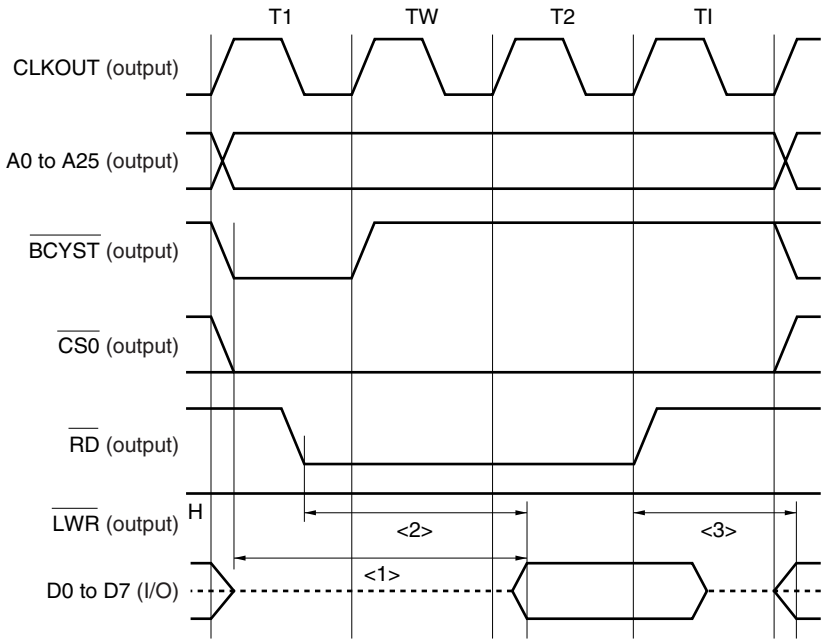


**Figure 2-8. Read Operation of μPD434008ALE-12**




<1> Output delay time from active state transition of address and $\overline{CS}$ signals in μPD434008ALE-12: 12 ns (Max.)

Maximum value for data input setup time (to address), based on the V850E/MA1's electrical specifications:

$t_{SAID}$ (ns) $= (2 + w + w_D + w_{AS})T - 21$

$= 3 \times 20 - 21$; w = 0, $w_D$ = 1, $w_{AS}$ = 0, T = 20 ns

= 39 ns (> 12 ns)

<2> Output delay time from active state transition of $\overline{OE}$ signal in μPD434008ALE-12: 6 ns (Max.)

Maximum value for data input setup time (to $\overline{RD}$), based on the V850E/MA1's electrical specifications;

$t_{SRDID}$ (ns) $= (1.5 + w + w_D)T - 21$

$= 2.5 \times 20 - 21$; w = 0, $w_D$ = 1, T = 20 ns

= 29 ns (> 6 ns)

<3> Output floating delay time from inactive state transition of $\overline{OE}$ signal in μPD434008ALE-12: 6 ns (Max.)

Minimum value for data output delay time (from $\overline{RD}\uparrow$), based on the V850E/MA1's electrical specifications:

$t_{DRDOD}$ (ns) $= (0.5 + i)T - 10$

$= 1.5 \times 20 - 10$; i = 1, T = 20 ns

= 20 ns (> 6 ns)

Minimum value for $\overline{RD}$ high level width:

$t_{WRDH}$ (ns) $= (0.5 + w_{AS} + i)T - 10$

$= 1.5 \times 20 - 10$; $w_{AS}$ = 0, i = 1, T = 20 ns

= 20 ns (> 6 ns)

**Remarks 1.** Broken lines indicate high impedance.

**2.** T: $t_{CYK}$ (CLKOUT output period)

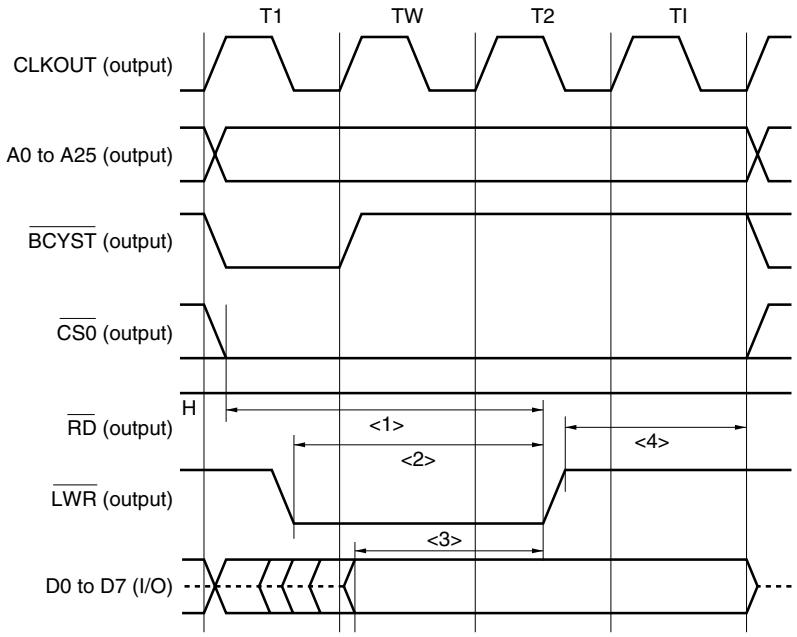
w: Number of waits from $\overline{WAIT}$

$w_D$: Number of waits from DWC1, DWC2

$w_{AS}$: Number of address setup waits from ASC

i: Number of idle states

**Figure 2-9. Write Operation of μPD434008ALE-12**




<1> Time from active state transition of address and $\overline{CS}$ signals in μPD434008ALE-12 to end of write operation: 8 ns (Min.)

Minimum value for address setup time (to $\overline{LWR}\uparrow$), based on the V850E/MA1's electrical specifications:

$t_{SAWR}$ (ns) = $(1.5 + w + w_D + w_{AS})T - 10$

= $2.5 \times 20 - 10$; w = 0, $w_D$ = 1, $w_{AS}$ = 0, T = 20 ns

= 40 ns (> 8 ns)

<2> $\overline{WE}$ active pulse width of μPD434008ALE-12: 8 ns (Min.)

Minimum value for $\overline{LWR}$ low level pulse width, based on the V850E/MA1's electrical specifications:

$t_{WWRL}$ (ns) = $(1 + w + w_D)T - 10$

= $2 \times 20 - 10$; w = 0, $w_D$ = 1, T = 20 ns

= 30 ns (> 8 ns)

<3> Time from data valid state in μPD434008ALE-12 to end of write operation: 6 ns (Min.)

Minimum value for data output setup time (to $\overline{LWR}\uparrow$), based on the V850E/MA1's electrical specifications:

$t_{SODWR}$ (ns) = $(0.5 + w_{AS} + w + w_D)T - 10$

= $1.5 \times 20 - 10$; $w_{AS}$ = 0, w = 0, $w_D$ = 1, T = 20 ns

= 20 ns (> 6 ns)

<4> Data hold time from end of μPD434008ALE-12's write operation: 0 ns (Min.)

Minimum value for data output hold time (from $\overline{LWR}\uparrow$), based on the V850E/MA1's electrical specifications:

$t_{HWROD}$ (ns) = $(0.5 + i)T - 10$

= $1.5 \times 20 - 10$; i = 1, T = 20 ns

= 20 ns (> 0 ns)

**Remarks 1.** Broken lines indicate high impedance.

**2.** T: $t_{CYK}$ (CLKOUT output period)

w: Number of waits from $\overline{WAIT}$

$w_D$: Number of waits from DWC1, DWC2

$w_{AS}$: Number of address setup waits from ASC

i: Number of idle states

### 2.1.2 Connection with μPD434008A (example of 16-bit bus width)

The following is an example in which two SRAMs (μPD434008ALE-12: 512 Kbits × 8 bits, 5 V) are connected as a 1 MB external memory space with 16-bit bus width).

**[Circuit configuration]**

- Internal system clock: 50 MHz
- Connected devices: μPD434008ALE-12 (2 devices)
- $\overline{CS}$ signal used: $\overline{CS0}$
  Assigned to address range 0100000H to 01FFFFFH (block 0) in external memory space.

**[Connection approach and caution points]**

Due to its 16-bit bus width, A0 to A19 in the V850E/MA1's address bus is connected to A0 to A18 in the SRAM. The SRAM's $\overline{WE}$ pin is used to connect the SRAM that is connected to D0 to D7 in the V850E/MA1 to the V850E/MA1's $\overline{LWR}$ pin and is also used to connect the SRAM that is connected to D8 to D15 in the V850E/MA1 to the $\overline{UWR}$ pin. The SRAM's $\overline{CS}$ and $\overline{OE}$ pins are respectively connected to the V850E/MA1's $\overline{CS0}$ and $\overline{RD}$ pins.

**[Register settings]**

The register settings are the same as in **2.1.1 Connection with μPD434008A (example of 8-bit bus width)**, except for the BSC register (that specifies the $\overline{CSn}$ space's bus width), which is explained below.

**Figure 2-10. Settings in Bus Size Configuration Register (BSC)**

| Bit | 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 |
| $\overline{CSn}$ signal | | $\overline{CS7}$ | | $\overline{CS6}$ | | $\overline{CS5}$ | | $\overline{CS4}$ | | $\overline{CS3}$ | | $\overline{CS2}$ | | $\overline{CS1}$ | | $\overline{CS0}$ |

16-bit data bus width is set for $\overline{CS0}$ space
Setting value in BSC: 0x0x0x0x0x0x01B

**Figure 2-11. μPD434008ALE-12 Connection Circuit Example**

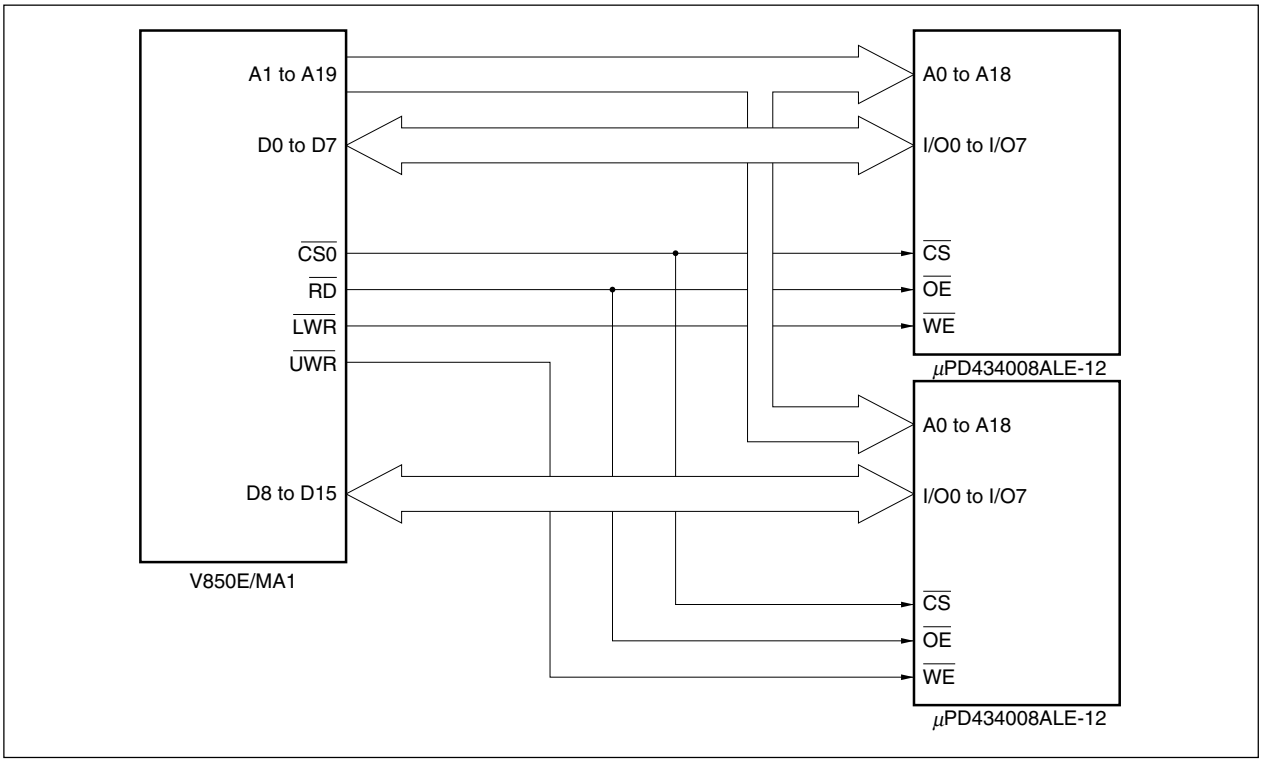
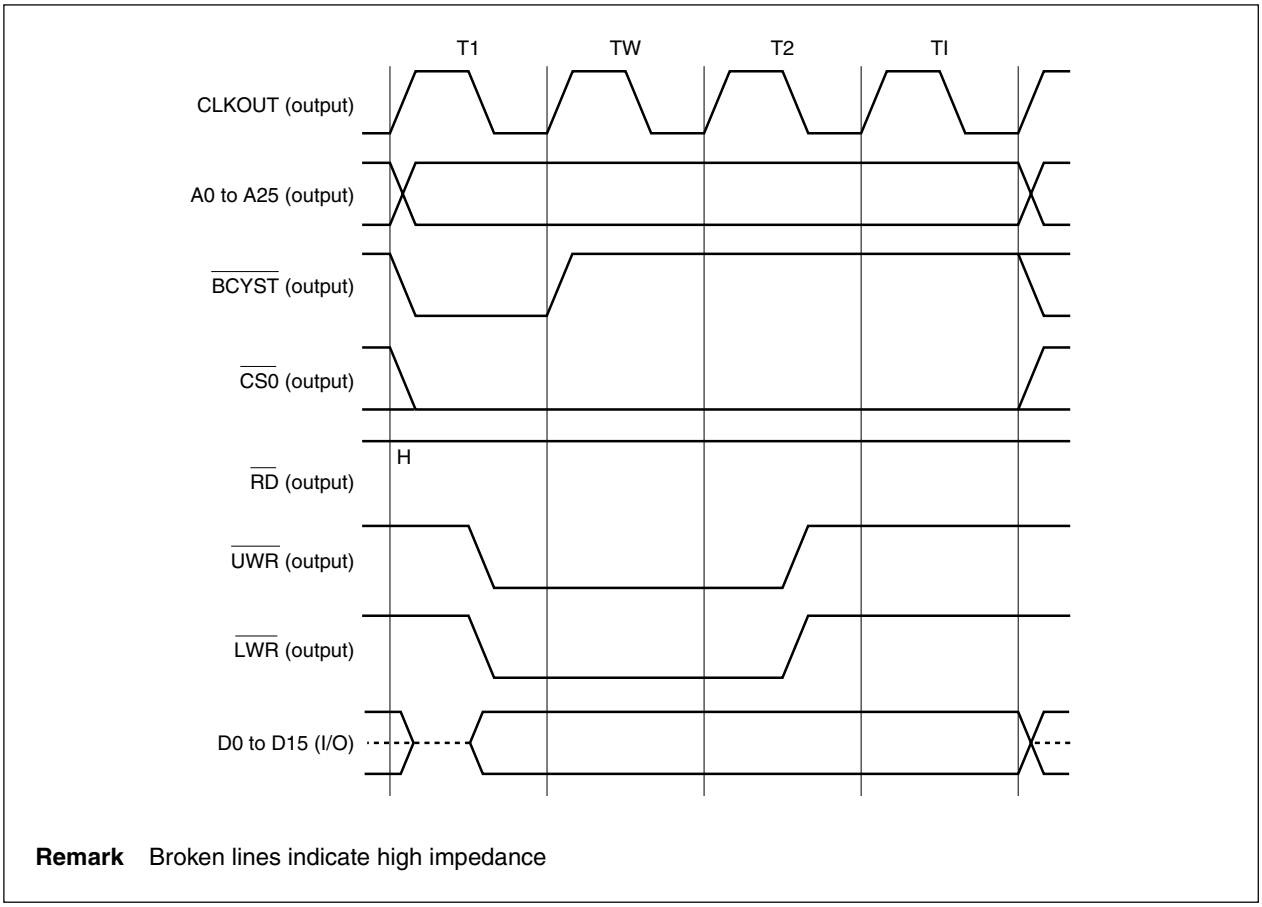
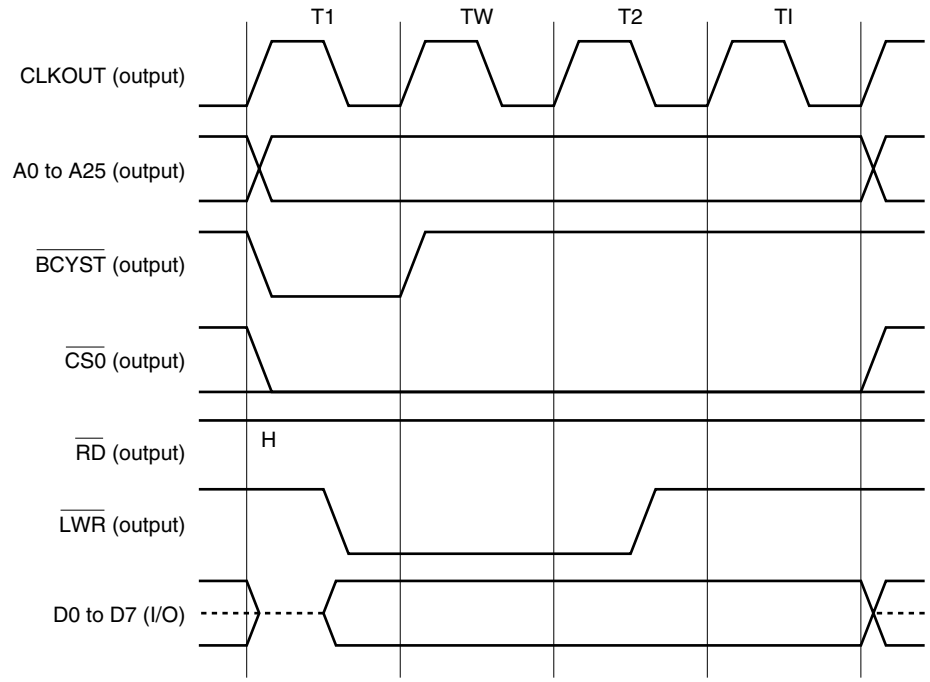



**Figure 2-12. Write Operation of μPD434008ALE-12 (16-Bit Access)**



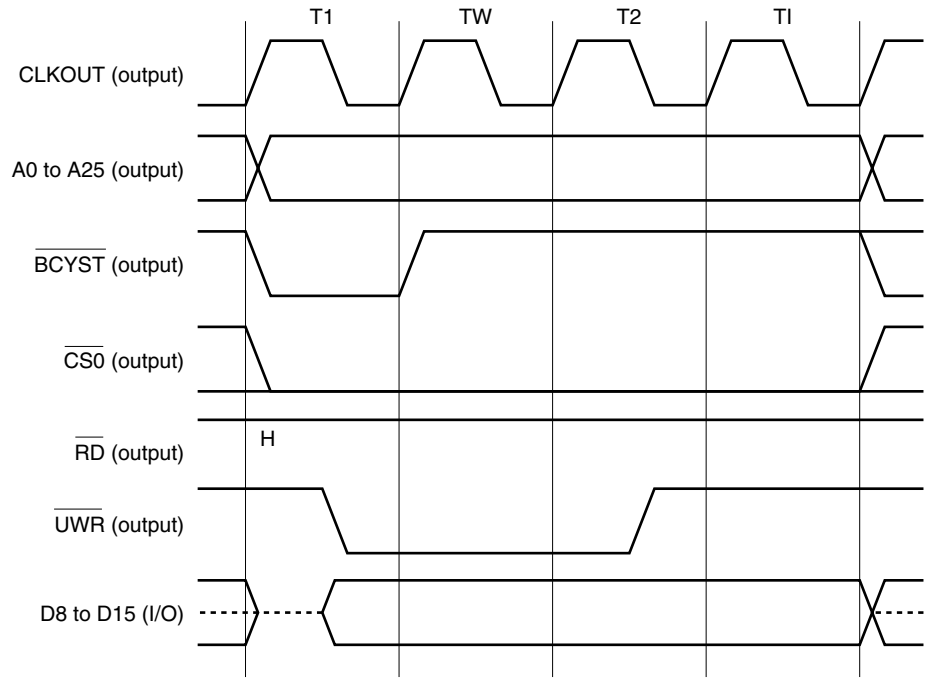

**Remark** Broken lines indicate high impedance

**Figure 2-13. Write Operation of μPD434008ALE-20 (8-Bit Access to D0 to D7)**




**Remark** Broken lines indicate high impedance

**Figure 2-14. Write Operation of μPD434008ALE-20 (8-Bit Access to D8 to D15)**




**Remark** Broken lines indicate high impedance

**Table 2-2. List of Wait/Idle Settings and Bus Clocks for SRAM Connection**

**(a) Connection with μPD434008ALE-12**

| fxx (CLKOUT) | Data Waits | Address Setup Waits | Idle States | Restricted Timing |
|---|---|---|---|---|
| fxx ≤ 31 MHz | 0 | 0 | 0 | – |
| 31 MHz < fxx (≤ 50 MHz) | 1 | 0 | 1 | $t_{DRDOD}$, $t_{SODWR}$ |

**(b) Connection with μPD434008ALE-15/μPD434008ALLE-15**

| fxx (CLKOUT) | Data Waits | Address Setup Waits | Idle States | Restricted Timing |
|---|---|---|---|---|
| fxx ≤ 29 MHz | 0 | 0 | 0 | – |
| 29 MHz < fxx (≤ 50 MHz) | 1 | 0 | 1 | $t_{DRDOD}$, $t_{SODWR}$ |

**(c) Connection with μPD434008ALE-20/μPD434008ALLE-20**

| fxx (CLKOUT) | Data Waits | Address Setup Waits | Idle States | Restricted Timing |
|---|---|---|---|---|
| fxx ≤ 26 MHz | 0 | 0 | 0 | – |
| 26 MHz < fxx (≤ 50 MHz) | 1 | 0 | 1 | $t_{DRDOD}$, $t_{SODWR}$ |

**(d) Connection with μPD431000A-A10**
**(when $V_{CC}$ ≥ 3.0 V, $\overline{CE1}$ is connected to $\overline{CSn}$, and CE2 is fixed at high level)**

| fxx (CLKOUT) | Data Waits | Address Setup Waits | Idle States | Restricted Timing |
|---|---|---|---|---|
| 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 Connection with PROM

The following is an example in which one PROM (HN27C4096H: 256 Kbits × 16 bits) is connected (as a 512 KB external ROM space).

**[Circuit configuration]**

- Internal system clock: 50 MHz
- Connected device: HN27C4096H-85 × 1
- $\overline{CS}$ signal used: $\overline{CS0}$
  Assigned to address range 0100000H to 017FFFFH (block 0) in external memory space. 0180000H to 01FFFFFH is used for the image.

**[Connection approach and caution points]**

- Address signals (A1 to A18) in the V850E/MA1 are connected to address signals (A0 to A17) in the HN27C4096H-85.
- PROM access is similar to the SRAM read operation. Accordingly, the setting values in the relevant registers are the same as described in **2.1.2 Connection with μPD434008A (example of 16-bit bus width)** (only the wait settings are different).

**[Register settings]**

- Block and device type used with $\overline{CS0}$: Block 0, SRAM, external I/O
- Wait setting: 4 waits (2 waits[Note])
- Address setup waits: 0 waits (2 waits[Note])
- Idle states: 2 states

**Note** This is the same as when the wait setting = 2 and address setup waits = 2.

**Figure 2-15. Settings in Chip Area Select Control Register 0 (CSC0)**

| Bit | 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 |

$\overline{CS0}$ output when accessing block 0
Setting value in CSC0: xxxxxxxxxxxx0001B

**Figure 2-16. Settings in Bus Cycle Type Configuration Register 0 (BCT0)**

| Bit | 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 signal | | | $\overline{CS3}$ | | | | | $\overline{CS2}$ | | | $\overline{CS1}$ | | | | | $\overline{CS0}$ |

Memory controller enabled, SRAM, external I/O are specified for $\overline{CS0}$ space
Setting value in BCT: x0xxx00xx0xx1000B

**Figure 2-17. Settings in Bus Size Configuration Register (BSC)**

| Bit | 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 signal | | $\overline{CS7}$ | | $\overline{CS6}$ | | $\overline{CS5}$ | | $\overline{CS4}$ | | $\overline{CS3}$ | | $\overline{CS2}$ | | $\overline{CS1}$ | | $\overline{CS0}$ |

16-bit data bus width is set for $\overline{CS0}$ space
Setting value in BSC: 0x0x0x0x0x0x0x01B

**Figure 2-18. Settings in Data Wait Control Register 0 (DWC0)**

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DWC0 | 0 | DW | DW | DW | 0 | DW | DW | DW | 0 | DW | DW | DW | 0 | DW | DW | DW |
| [FFFFF484H] | | 32 | 31 | 30 | | 22 | 21 | 20 | | 12 | 11 | 10 | | 02 | 01 | 00 |
| $\overline{CS}$n signal | | | $\overline{CS3}$ | | | | $\overline{CS2}$ | | | | $\overline{CS1}$ | | | | $\overline{CS0}$ | |

Number of waits set in $\overline{CS0}$ space: 4
Setting value in DWC0: 0xxx0xxx0xxx0100B

**Figure 2-19. Settings in Address Setup Wait Control Register (ASC)**

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ASC | AC | AC | AC | AC | AC | AC | AC | AC | AC | AC | AC | AC | AC | AC | AC | AC |
| [FFFFF48AH] | 71 | 70 | 61 | 60 | 51 | 50 | 41 | 40 | 31 | 30 | 21 | 20 | 11 | 10 | 01 | 00 |
| $\overline{CS}$n signal | $\overline{CS7}$ | | $\overline{CS6}$ | | $\overline{CS5}$ | | $\overline{CS4}$ | | $\overline{CS3}$ | | $\overline{CS2}$ | | $\overline{CS1}$ | | $\overline{CS0}$ | |

Number of address setup waits set in $\overline{CS0}$ space: 0
Setting value in ASC: xxxxxxxxxxxxxx00B

**Figure 2-20. Settings in Bus Cycle Control Register (BCC)**

| Bit | 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 signal | $\overline{CS7}$ | | $\overline{CS6}$ | | $\overline{CS5}$ | | $\overline{CS4}$ | | $\overline{CS3}$ | | $\overline{CS2}$ | | $\overline{CS1}$ | | $\overline{CS0}$ | |

Number of idle states set in $\overline{CS0}$ space: 2

Setting value in BCC: xxxxxxxxxxxxxx10B

**Figure 2-21. Example of HN27C4096H-85 Connection Circuit**

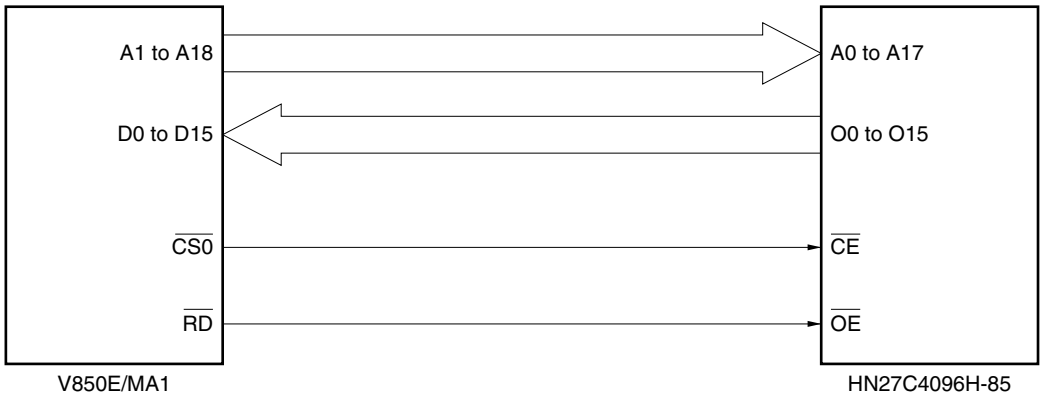
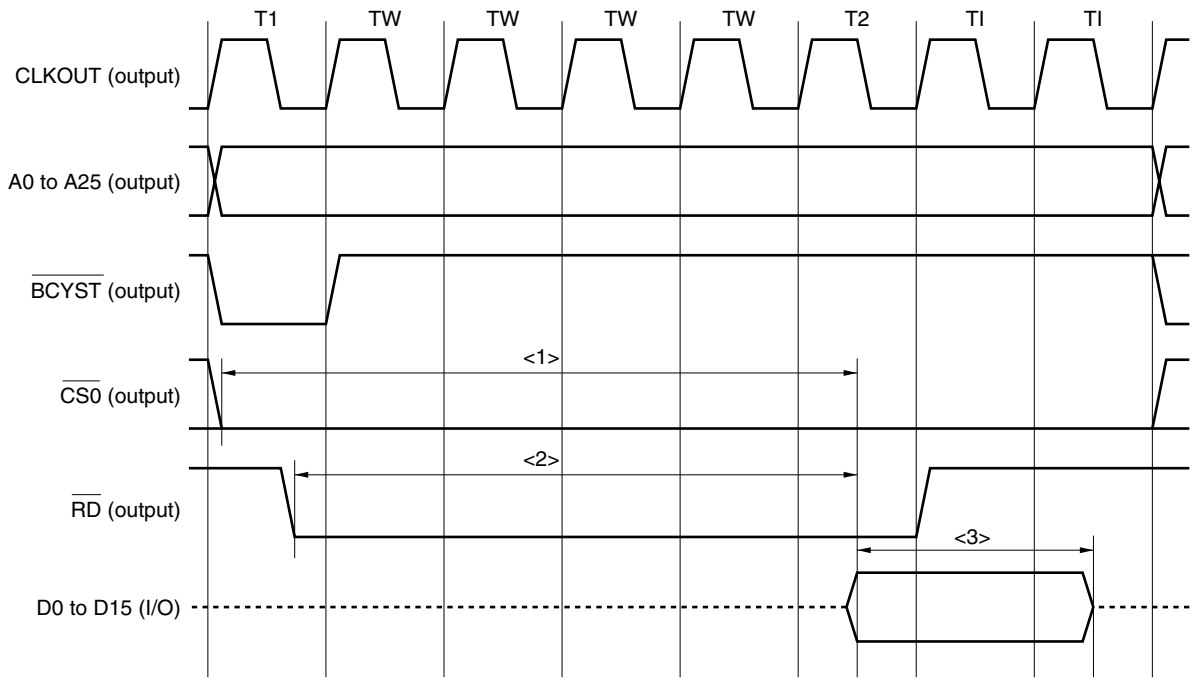


**Figure 2-22. Read Operation of HN27C4096H-85**




<1> Data output delay time from active state transition of HN27C4096H-85's $\overline{CE}$ signal: 85 ns (Max.)

<2> Data output delay time from active state transition of HN27C4096H-85's $\overline{OE}$ signal: 45 ns (Max.)

<3> Data output floating delay time from inactive state transition of HN27C4096H-85's OE signal: 30 ns (Max.)

**Remark** Broken lines indicate high impedance.

## 2.3 Connection with Page ROM

The following is an example in which one page ROM (HN27C4000G: 256 Kbits × 16 bits or 512 Kbits × 8 bits) is connected (as a 512 KB external ROM space with 16-bit bus width).

**[Circuit configuration]**

- Internal system clock: 50 MHz
- Connected device: HN27C4000G-10 × 1
- $\overline{CS}$ signal used: $\overline{CS1}$
  Assigned to address range 0200000H to 027FFFFH (block 1) in external memory space.

**Caution In this example, block 1 is used for $\overline{CS1}$. $\overline{CS1}$ is output when block 1 and address range 0800000H to 3FFFFFFH are accessed (0280000H to 03FFFFFH, 0800000H to 3FFFFFFH are used for the image).**

**[Connection approach and caution points]**

- The HN27C4000G-10's $\overline{BYTE}$/$V_{PP}$ pin is used in 16-bit mode and is therefore pulled up.
- Since block 1 is used for $\overline{CS1}$, the CSC0 register is set so that both $\overline{CS0}$ and $\overline{CS2}$ are not output while block 1 is being accessed.
- Since the HN27C4000G-10's page size is 8 bytes, the PRC register is set for 4 words × 16 bits.

**[Register settings]**

- Block and device types used with $\overline{CS1}$: Block 1, Page ROM
- Wait setting (off-page): 5 waits
- Wait setting (on-page): 2 waits
- Address setup waits: 0 waits
- Idle states: 2 states

**Figure 2-23. Settings in Chip Area Select Control Register 0 (CSC0)**

| Bit | 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 |

$\overline{CS1}$ output when accessing block 1
Setting value in CSC0: xxxxxx0xxxxxxx0xB

**Figure 2-24. Settings in Bus Cycle Type Configuration Register 0 (BCT0)**

| Bit | 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 signal | | | $\overline{CS3}$ | | | | | $\overline{CS2}$ | | | $\overline{CS1}$ | | | | | $\overline{CS0}$ |

Memory controller enabled, page ROM is specified for $\overline{CS1}$
Setting value in BCT: x0xxx00x1001x00xB

**Figure 2-25. Settings in Bus Size Configuration Register (BSC)**

| Bit | 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 signal | | $\overline{CS7}$ | | $\overline{CS6}$ | | $\overline{CS5}$ | | $\overline{CS4}$ | | $\overline{CS3}$ | | $\overline{CS2}$ | | $\overline{CS1}$ | | $\overline{CS0}$ |

16-bit data bus width is set for $\overline{CS1}$ space
Setting value in BSC: 0x0x0x0x0x0x010xB

**Figure 2-26. Settings in Data Wait Control Register 0 (DWC0)**

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DWC0 | 0 | DW | DW | DW | 0 | DW | DW | DW | 0 | DW | DW | DW | 0 | DW | DW | DW |
| [FFFFF484H] | | 32 | 31 | 30 | | 22 | 21 | 20 | | 12 | 11 | 10 | | 02 | 01 | 00 |
| $\overline{CS}$n signal | | | $\overline{CS3}$ | | | | $\overline{CS2}$ | | | | $\overline{CS1}$ | | | | $\overline{CS0}$ | |

Number of waits set in $\overline{CS1}$ space: 5 (off-page)
Setting value in DWC0: 0xxx0xxx01010xxxB

**Figure 2-27. Settings in Address Setup Wait Control Register (ASC)**

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ASC | AC | AC | AC | AC | AC | AC | AC | AC | AC | AC | AC | AC | AC | AC | AC | AC |
| [FFFFF48AH] | 71 | 70 | 61 | 60 | 51 | 50 | 41 | 40 | 31 | 30 | 21 | 20 | 11 | 10 | 01 | 00 |
| $\overline{CS}$n signal | $\overline{CS7}$ | | $\overline{CS6}$ | | $\overline{CS5}$ | | $\overline{CS4}$ | | $\overline{CS3}$ | | $\overline{CS2}$ | | $\overline{CS1}$ | | $\overline{CS0}$ | |

Number of address setup waits set in $\overline{CS1}$ space: 0
Setting value in ASC: xxxxxxxxxxxx00xxB

**Figure 2-28. Settings in Bus Cycle Control Register (BCC)**

| Bit | 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 signal | | $\overline{CS7}$ | | $\overline{CS6}$ | | $\overline{CS5}$ | | $\overline{CS4}$ | | $\overline{CS3}$ | | $\overline{CS2}$ | | $\overline{CS1}$ | | $\overline{CS0}$ |

Number of idle states set in $\overline{CS1}$ space: 2
Setting value in BCC: xxxxxxxxxxxx10xxB

**Figure 2-29. Settings in Page ROM Configuration Register (PRC)**

| Bit | 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 | | | | | | | | | | | | |

Number of waits for on-page: 2
Page size: 4 words × 16 bits
Setting value in PRC: 0010000000000000B

**Figure 2-30. Example of HN27C4000G-10 Connection Circuit**

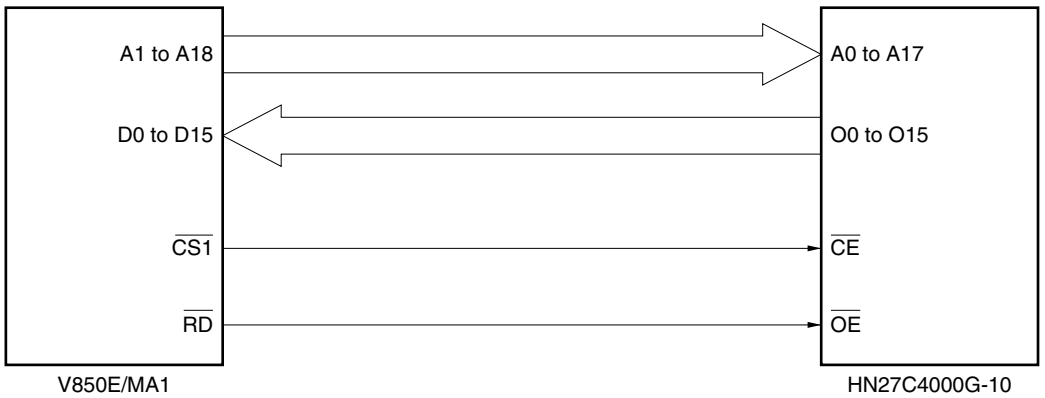
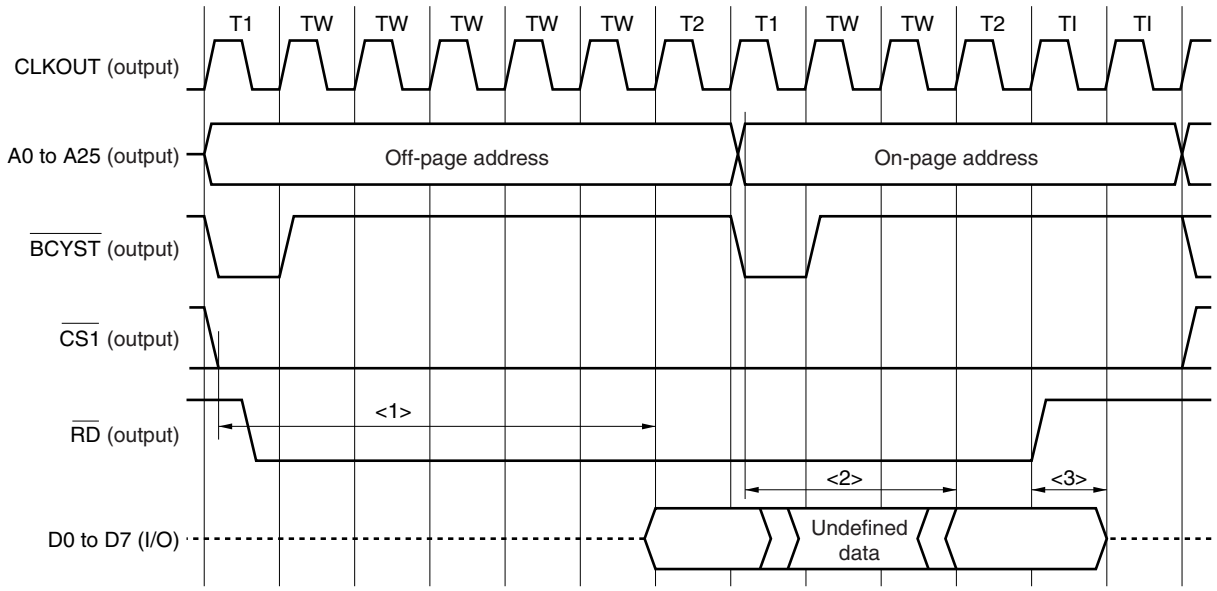


**Figure 2-31. Read Operation of HN27C4000G-10**




<1> Data output delay time from active state transition of HN27C4000G-10's $\overline{CE}$ signal: 100 ns (Max.)

<2> Data output delay time from address modification during an on-page: 50 ns (Max.)

<3> Data output floating delay time: 35 ns (Max.)

**Caution The $\overline{RD}$ signal is held at low level for up to four bus cycles at page hit when using 8-bit bus width (or up to two cycles when using 16-bit bus width) to access consecutively. No idles are inserted between accesses.**

**Remark** Broken lines indicate high impedance.

## 2.4 Connection with Flash Memory (SST29LE010)

The following is an example in which one flash memory (SST29LE010, 128 Kbits × 8 bits) is connected as a 128 KB external memory space with 8-bit bus width.

**[Circuit configuration]**

- Internal system clock: 50 MHz
- Connected device: SST29LE010-150 × 1
- $\overline{CS}$ signal used: $\overline{CS2}$
  Assigned to address range 0400000H to 041FFFFH (block 2) in external memory space. The range from 0420000H to 05FFFFFH is used for the image.

**[Connection approach and caution points]**

- Due to its 8-bit bus width, the SST29LE010's data bus is connected to D0 to D7 in the V850E/MA1 and its address bus is connected to A0 to A16 in the V850E/MA1. Also, the SST29LEO10's $\overline{CE}$ pin, $\overline{OE}$ pin, and $\overline{WE}$ pin are respectively connected to the V850E/MA1's $\overline{CS2}$ pin, $\overline{RD}$ pin, and $\overline{LWR}$ pin.

  **Remarks 1.** The SST29LE010 is a read/write accessible flash memory that operates on a 3 V single power supply. Its pin array does not include a RY/$\overline{BY}$ pin, so it connects in the same as an SRAM.

  **2.** If using a flash memory whose pin array does include a RY/$\overline{BY}$ pin, completion status of write/erase operations can be monitored by connecting the RY/$\overline{BY}$ pin to the V850E/MA1, such as via a port input.

- The $\overline{CS2}$ space's wait settings are address setup waits = 1, data waits = 6, and idle states = 2.

  **Remark** To satisfy the minimum value (50 ns) for the SST29LE010's $\overline{WE}$ signal high level pulse width, set i = 2 and $w_{AS}$ = 1. Although settings of i = 3 and $w_{AS}$ = 0 also satisfy the minimum value for the $\overline{WE}$ signal's high level pulse width, in this case $w_D$ = 7 must be set to satisfy the minimum value (150 ns) for data setup time after the $\overline{CE}$ signal becomes active during a read operation, which results in a delay of one clock cycle.

- The following describes the SST29LE010's data write operation and erase operation.
  Chip erase:
  <1> Write data AAH to address 5555H[Note 1] in SST29LE010
  <2> Write data 55H to address 2AAAH[Note 2] in SST29LE010
  <3> Write data 80H to address 5555H[Note 1] in SST29LE010
  <4> Write data AAH to address 5555H[Note 1] in SST29LE010
  <5> Write data 55H to address 2AAAH[Note 2] in SST29LE010
  <6> Write data 10H to address 5555H[Note 1] in SST29LE010
  <7> Erase operation is completed after a specified time (20.2 ms)[Note 3]

Data write:

<1> Write data AAH to address 5555H[Note 1] in SST29LE010

<2> Write data 55H to address 2AAAH[Note 2] in SST29LE010

<3> Write data A0H to address 5555H[Note 1] in SST29LE010

<4> Write appropriate data to appropriate address in SST29LE010[Note 4]

<5> Write operation is completed after a specified time (10.2 ms)[Note 3]

**Notes 1.** Any values can be set for A15 and A16 in the SST29LE010. In this example, this corresponds to address 0405555H in the V850E/MA1's external memory space.

**2.** Any values can be set for A15 and A16 in the SST29LE010. In this example, this corresponds to address 0402AAAH in the V850E/MA1's external memory space.

**3.** The following are some ways to monitor the SST29LE010's D7 bit (polling bit) and D6 bit (toggle bit).

- Data polling bit (D7):
  Completion of an operation can be confirmed by reading the corresponding address.
  When busy, the inverted data (inverted data of 0 if an erase operation or of the write data if a write operation) is output.
  For a write operation, the last address written is read and confirmed and, for an "erase all" operation, the last address (in this case, 041FFFFH) is read and confirmed.
- Toggle bit (D6):
  Completion of an operation can be confirmed by reading the corresponding address twice consecutively.
  When busy, a toggle occurs for two consecutive read operations. The operation is completed when the same data is read twice consecutively.
  For a write operation, the last address written is read and confirmed, and for an "erase all" operation, the last address is read and confirmed.

**4.** In the SST29LE010, consecutive write operations can continue for up to one page (128 bytes).

**[Register settings]**

- Block and device types used with $\overline{CS2}$: Block 2, SRAM, External I/O
- Wait setting: 6 waits
- Address setup waits: 1 wait
- Idle states: 2 states

**Figure 2-32. Settings in Chip Area Select Control Register 0 (CSC0)**

| Bit | 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 |

$\overline{CS2}$ output when accessing block 2
Setting value in CSC0: xxxx0100xxxxx0xxB

**Figure 2-33. Settings in Bus Cycle Type Configuration Register 0 (BCT0)**

| Bit | 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 signal | | | $\overline{CS3}$ | | | | | $\overline{CS2}$ | | | $\overline{CS1}$ | | | | | $\overline{CS0}$ |

Memory controller enabled, SRAM, external I/O are specified for $\overline{CS2}$ space
Setting value in BCT0: x0xx1000x0xxx00xB

**Figure 2-34. Settings in Bus Size Configuration Register (BSC)**

| Bit | 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 signal | | $\overline{CS7}$ | | $\overline{CS6}$ | | $\overline{CS5}$ | | $\overline{CS4}$ | | $\overline{CS3}$ | | $\overline{CS2}$ | | $\overline{CS1}$ | | $\overline{CS0}$ |

8-bit data bus width is set for $\overline{CS2}$ space
Setting value in BSC: 0x0x0x0x0x000x0xB

**Figure 2-35. Settings in Data Wait Control Register 0 (DWC0)**

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DWC0 | 0 | DW | DW | DW | 0 | DW | DW | DW | 0 | DW | DW | DW | 0 | DW | DW | DW |
| [FFFFF484H] | | 32 | 31 | 30 | | 22 | 21 | 20 | | 12 | 11 | 10 | | 02 | 01 | 00 |
| $\overline{CS}$n signal | | | $\overline{CS3}$ | | | | $\overline{CS2}$ | | | | $\overline{CS1}$ | | | | $\overline{CS0}$ | |

Number of waits set in $\overline{CS2}$ space: 6
Setting value in DWC0: 0xxx01100xxx0xxxB

**Figure 2-36. Settings in Address Setup Wait Control Register (ASC)**

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ASC | AC | AC | AC | AC | AC | AC | AC | AC | AC | AC | AC | AC | AC | AC | AC | AC |
| [FFFFF48AH] | 71 | 70 | 61 | 60 | 51 | 50 | 41 | 40 | 31 | 30 | 21 | 20 | 11 | 10 | 01 | 00 |
| $\overline{CS}$n signal | $\overline{CS7}$ | | $\overline{CS6}$ | | $\overline{CS5}$ | | $\overline{CS4}$ | | $\overline{CS3}$ | | $\overline{CS2}$ | | $\overline{CS1}$ | | $\overline{CS0}$ | |

Number of address setup waits set in $\overline{CS2}$ space: 1
Setting value in ASC: xxxxxxxxxx01xxxxB

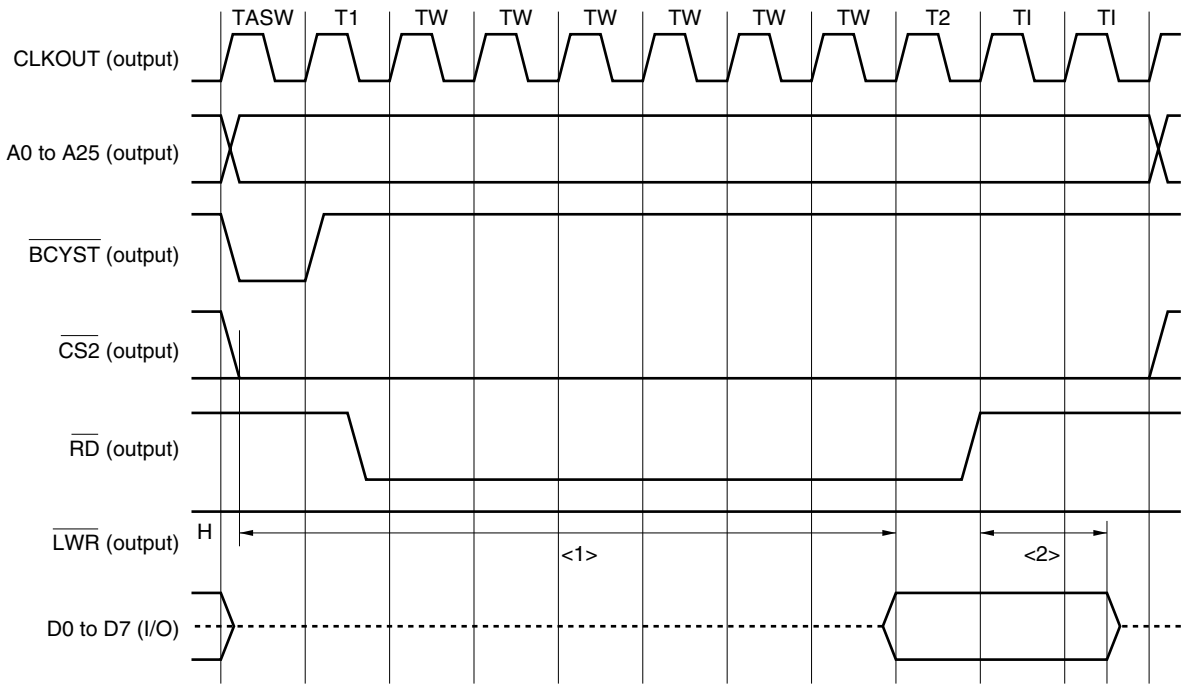
**Figure 2-37. Settings in Bus Cycle Control Register (BCC)**

| Bit | 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 |
| $\overline{\text{CS}}$n signal | | $\overline{\text{CS7}}$ | | $\overline{\text{CS6}}$ | | $\overline{\text{CS5}}$ | | $\overline{\text{CS4}}$ | | $\overline{\text{CS3}}$ | | $\overline{\text{CS2}}$ | | $\overline{\text{CS1}}$ | | $\overline{\text{CS0}}$ |

Number of idle states set in $\overline{\text{CS2}}$ space: 2
Setting value in BCC: xxxxxxxxxx10xxxxB

**Figure 2-38. Example of SST29LE010-150 Connection Circuit**

**Figure 2-39. Read Operation of SST29LE010-150**

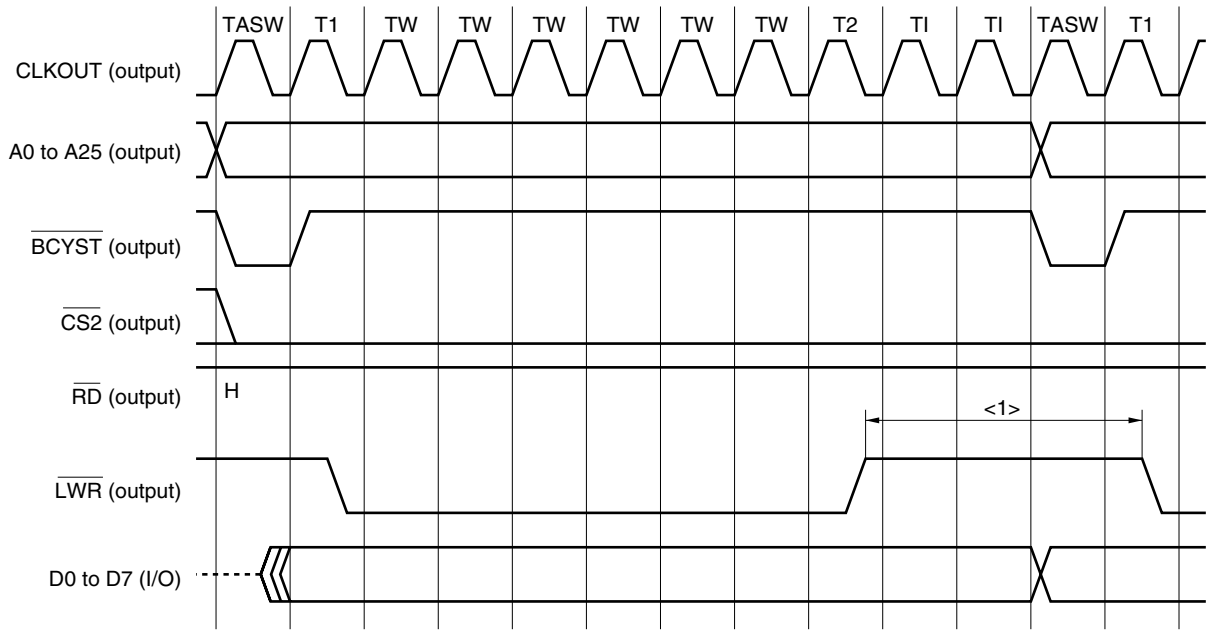



<1> Data output delay time from active state transition of SST29LE010-150's $\overline{CE}$ signal: 150 ns (Max.)

<2> Data output floating delay time from inactive state transition of SST29LE010-150's $\overline{OE}$ signal: 30 ns (Max.)

**Remark** Broken lines indicate high impedance.

**Figure 2-40. Write Operation of SST29LE010-150**




<1> High level pulse width of SST29LE010-150's $\overline{WE}$ signal: 50 ns (Min.)

**Remark** Broken lines indicate high impedance

## 2.5 Connection with EDO DRAM

The following is an example in which one EDO DRAM (HM5164165FL: 4 Mbits × 16 bits) is connected (as an 8 MB external memory space with 16-bit bus width).

**[Circuit configuration]**

- Internal system clock: 50 MHz
- Connected device: HM5164165FL-5 × 1
- $\overline{CS}$ signal used: $\overline{CS1}$ ($\overline{RAS1}$)
  Assigned to address range 0800000H to 0FFFFFFH in external memory space.
  The address range 1000000H to 3FFBFFFH is used for the image).

**[Connection approach and caution points]**

- Since the V850E/MA1's $\overline{CS1}$ ($\overline{RAS1}$) pin connects to the EDO DRAM's $\overline{RAS}$ pin, the access timing is set in DRC1.
- Since the connection uses a 16-bit bus width, the V850E/MA1's address pins (A1 to A13) connect to the HM5164165FL's address pins A0 to A12.
- Since the HM5164165FL's addresses consist of a 13-bit row address and a 9-bit column address, when outputting the row address the address multiplex width (= 9 bits) is set as shown below to make the following settings for address pins A1 to A13 in the V850E/MA1.

| Row address | A13 | A12 | A11 | A10 | A9 | A8 | A7 | A6 | A5 | A4 | A3 | A2 | A1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | a22 | - - - - - - - - - - - - - - - - - - - - - - - - - - - | | | | | | | | | | a11 | a10 |

- Since the CBR refresh cycle is 4096 cycles/64 ms, the refresh interval should be set at 15.6 $\mu$s or less.
- In this case, the specifications can be satisfied by setting the $\overline{RAS}$ access time as either "$w_{RH}$ = 2, $w_{DA}$ = 1" or "$w_{RH}$ = 1, $w_{DA}$ = 2", the setting "$w_{RH}$ = 2, $w_{DA}$ = 1" is assumed. If extra waits can be inserted for either setting, insert them for the less frequent setting.

Table 2-3 compares the AC characteristics of the V850E/MA1 and HM5164165FL-5 based on the wait settings required in this circuit example.

**Table 2-3. Comparison of AC Characteristics of V850E/MA1 and HM5164165FL-5**

| Item | V850E/MA1 | HM5164165FL-5 |
|---|---|---|
| $\overline{RAS}$ pulse width (Min.) | $t_{RASP}$: $(2 + w_{RH} + w_{DA})T - 10 = 70$ ns | (>) 50 ns |
| $\overline{RAS}$ precharge time (Min.) | $t_{RP}$: $w_{RP} \times T - 10 = 30$ ns | (=) 30 ns |
| Row address hold time (Min.) | $t_{RAH}$: $(0.5 + w_{RH})T - 10 = 20$ ns | (>) 8 ns |
| $\overline{RAS}$ access time (Max.) | $t_{RAC}$: $(2 + w_{RH} + w_{DA})T - 21 = 59$ ns | (>) 50 ns |
| $\overline{CAS}$ pulse width (Min.) | $t_{HCAS}$: $(0.5 + w_{DA})T - 10 = 20$ ns | (>) 8 ns |
| $\overline{CAS}$ precharge time (Min.) | $t_{CP}$: $(0.5 + w_{CP})T - 10 = 20$ ns | (>) 8 ns |
| Column address hold time (Min.) | $t_{CAH}$: $(0.5 + w_{DA})T - 10 = 20$ ns | (>) 8 ns |
| $\overline{CAS}$ access time (Max.) | $t_{CAC}$: $(1 + w_{DA})T - 21 = 19$ ns | (>) 13 ns |
| $\overline{OE}\uparrow$ data output delay time (Min.) | $t_{DRDOD}$: $(1 + i)T - 10 = 30$ ns | (>) 13 ns |
| Data hold time (from $\overline{CAS}\downarrow$) (Min.) | $t_{DH}$: $(0.5 + w_{DA})T - 10 = 20$ ns | (>) 8 ns |

**Remarks 1.** Only principal AC characteristics are compared. T = 20 ns during 50 MHz operation.

**2.** T = $t_{CYK}$ (CLKOUT output period)

$w_{RP}$ (row address precharge waits) = 2

$w_{RH}$ (row address hold waits) = 1

$w_{DA}$ (data access waits) = 1

$w_{CP}$ ($\overline{CAS}$ precharge waits) = 1

i (idle states) = 1

**[Register settings]**

- Device type used with $\overline{CS1}$: EDO DRAM
- Bus width of $\overline{CS1}$ space: 16 bits
- Idle states: 1 states
- On-page access: Enabled
- $\overline{RAS}$ hold mode: Enabled

**Remark** The setting values in register DWC0 have no significance during EDO DRAM access.

**Figure 2-41. Settings in Chip Area Select Control Register 0 (CSC0)**

| Bit | 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 |

Setting value in CSC0: xxxxxxxxxxxxxxxxB

**Caution If using blocks 0 to 3, be sure to set a value that disables output of $\overline{CS1}$ during access.**

**Figure 2-42. Settings in Bus Cycle Type Configuration Register 0 (BCT0)**

| Bit | 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 signal | | | | $\overline{CS3}$ | | | | $\overline{CS2}$ | | | | $\overline{CS1}$ | | | | $\overline{CS0}$ |

Memory controller enabled and EDO DRAM setting are specified for $\overline{CS1}$ space
Setting value in BCT0: x0xxx00x1010x00xB

**Figure 2-43. Settings in Bus Size Configuration Register (BSC)**

| Bit | 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 signal | | $\overline{CS7}$ | | $\overline{CS6}$ | | $\overline{CS5}$ | | $\overline{CS4}$ | | $\overline{CS3}$ | | $\overline{CS2}$ | | $\overline{CS1}$ | | $\overline{CS0}$ |

16-bit data bus width is set for $\overline{CS1}$ space
Setting value in BSC: 0x0x0x0x0x0x010xB

**Figure 2-44. Settings in Bus Cycle Control Register (BCC)**

| Bit | 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 signal | | $\overline{CS7}$ | | $\overline{CS6}$ | | $\overline{CS5}$ | | $\overline{CS4}$ | | $\overline{CS3}$ | | $\overline{CS2}$ | | $\overline{CS1}$ | | $\overline{CS0}$ |

Number of idle states set in $\overline{CS1}$ space: 1
Setting value in BCC: xxxxxxxxxxxx01xxB

**Figure 2-45. Settings in DRAM Configuration Register 1 (SCR1)**

| Bit | 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 (On-page access is enabled)
RPC11, RPC01: 1, 0 (Row address's precharge waits: $w_{RP}$ = 2)
RHC11, RHC01: 0, 1 (Row address's hold waits: $w_{RH}$ = 1)
DAC11, DAC01: 0, 1 (Data waits: $w_{DA}$ = 1)
CPC11, CPC01: 0, 1 (Column address precharge waits: $w_{CP}$ = 1)
RHD1: 0 ($\overline{RAS}$ hold mode is enabled)
ASO11, ASO01: 0, 1 (Data bus width: 16 bits)
DAW11, DAW01: 0, 1 (Address multiplex width: 9 bits)
Setting value in SCR1: A545H

**Figure 2-46. Settings in Refresh Control Register 1 (RFS1)**

| Bit | 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 (Refresh is enabled)
RCC11, RCC01: 0, 0 (Refresh count clock: $32/f_{XX}$)
RIN15 to RIN10: 0, 1, 0, 1, 1, 1 (interval factor: 24)
RSF1 setting: 8017H
Given that $f_{XX}$ = 50 MHz (0.02 $\mu$s), the refresh interval for the above settings is as follows.
Refresh interval = $24 \times 0.02 \times 32$
= 15.4 $\mu$s (< 15.6 $\mu$s)

**Figure 2-47. Settings in Refresh Wait Control Register (RWC)**

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

RRW1, RRW0: 1, 0 (Refresh $\overline{RAS}$ waits: RRW = 2)
RCW2 to RCW0: 0, 0, 0 (Refresh cycle waits: RCW = 1) (Be sure to insert one wait.)
SRW2 to SRW0: 0, 0, 1 (Self refresh cancel waits: SRW = 1)
RWC setting: 41H

**Cautions 1. $\overline{RAS}$ precharge time during HM5164165FL-5's self refresh cancel operation: 90 ns (Min.)**
**Based on the V850E/MA1's electrical specifications, the minimum $\overline{RAS}$ precharge time during the self refresh cancel operation is:**
**$t_{RPS}$ (ns) = (2 + 2$w_{SRW}$ + $w_{RRW}$)T – 10**
**= 6 × 20 – 10; $w_{SRW}$ = 1, $w_{RRW}$ = 2, T = 20**
**= 110 ns (> 90 ns)**

**2. If there are any other EDO DRAM devices to be connected, the RRW and RCW settings must be made for lower-speed memory.**

**Remark** T: $t_{CYK}$ (CLKOUT output period)
$w_{RRW}$: Number of waits from RRW0, RRW1 bits of RWC register
$w_{SRW}$: Number of waits from SRW0 to SRW2 bits of RWC register

**Figure 2-48. Example of HM5164165FL-5 Connection Circuit**

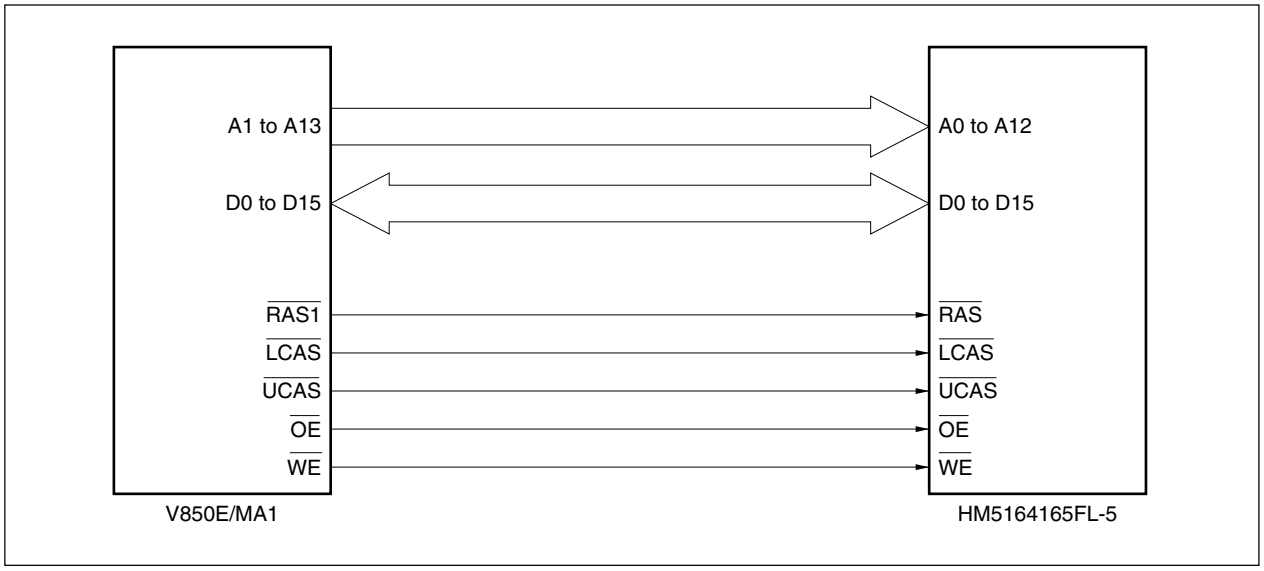
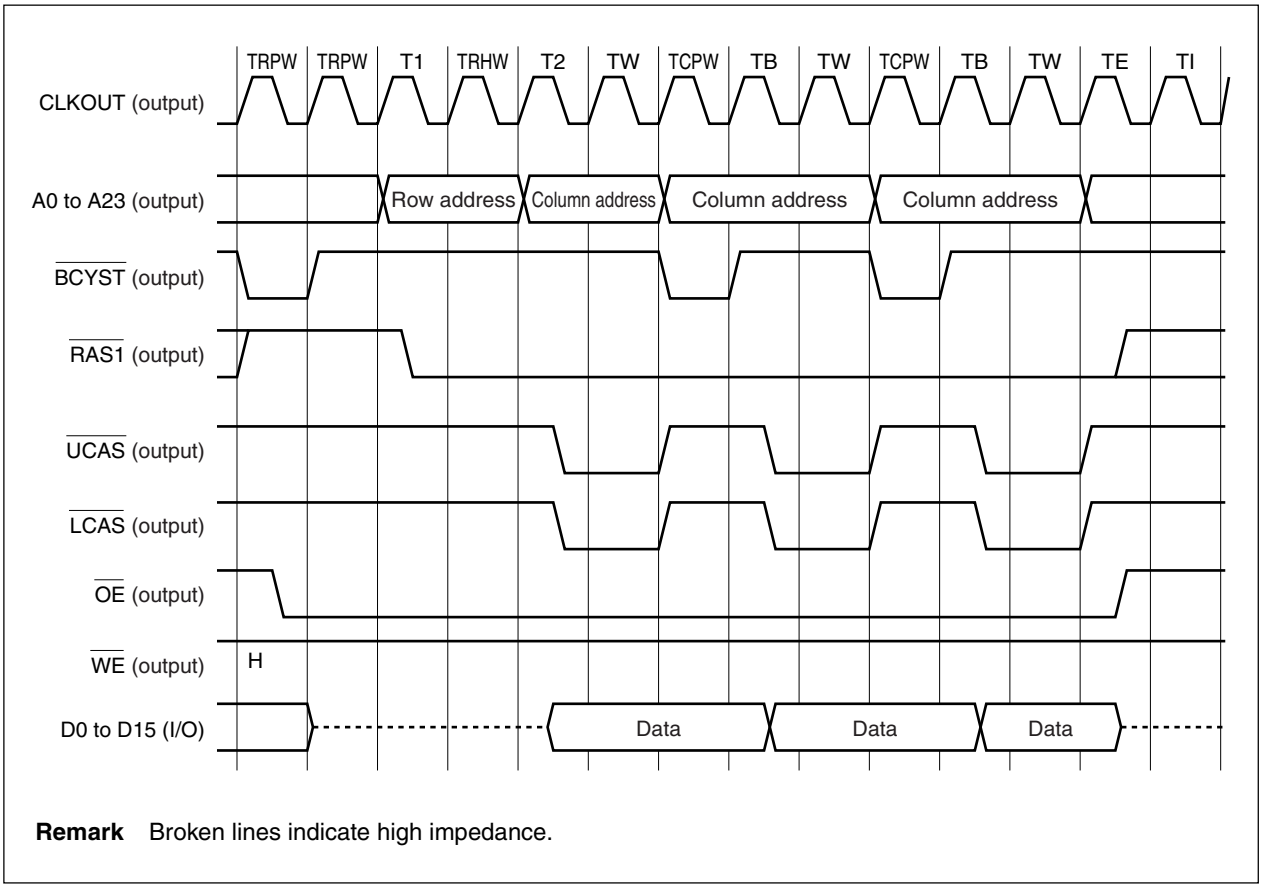
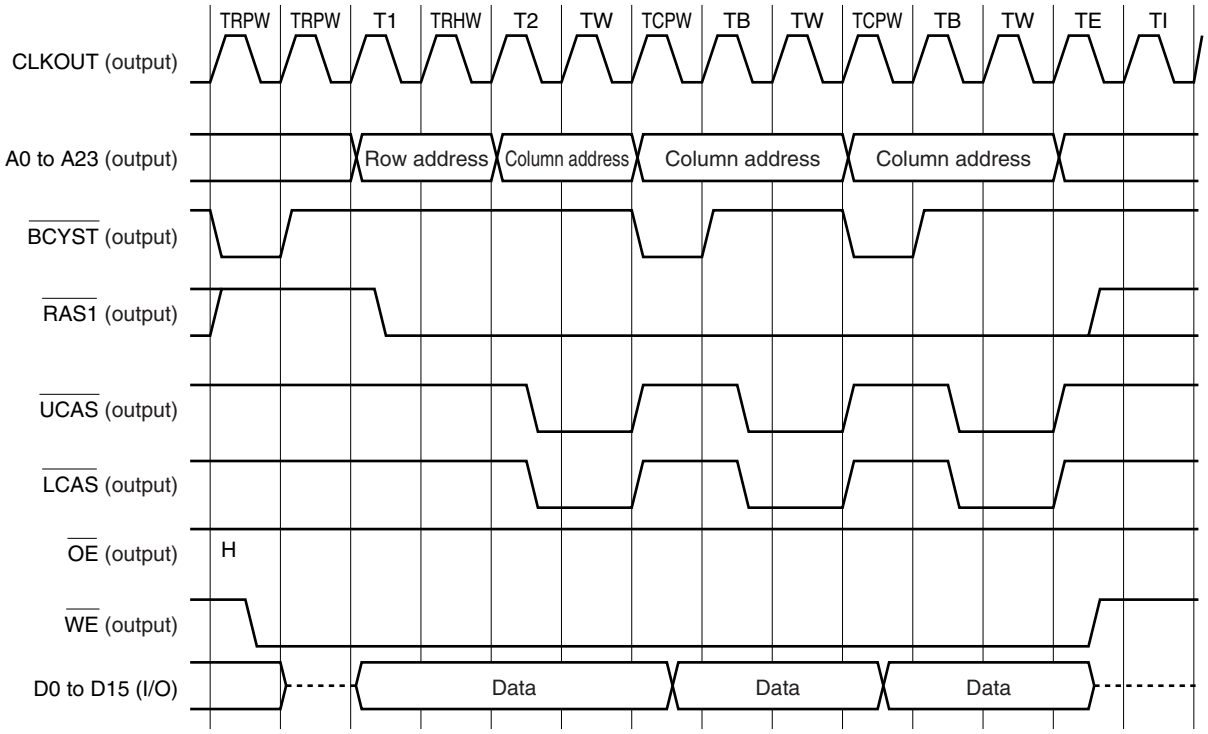



**Figure 2-49. Read Operation of HM5164165FL-5**




**Remark** Broken lines indicate high impedance.

**Figure 2-50. Write Operation of HM5164165FL-5**




**Remark** Broken lines indicate high impedance.

## 2.6 Connection with SDRAM

The following is an example in which one SDRAM (μPD4564163G5: 1 Mbit × 16 bits × 4 banks) is connected (as an 8 MB external memory space).

**[Circuit configuration]**

- Internal system clock: 50 MHz
- Connected device: μPD4564163G5-A10 × 1
- $\overline{CS}$ signal used: $\overline{CS3}$
  Assigned to address range 4000000H to 47FFFFFH in external memory space (the address range from 4800000H to 7FFFFFFH is used in 8 MB units for the image).

**Caution Programs cannot be assigned to this space. If a space to be used as a program area is needed, assign it to a lower 64 MB space.)**

**[Connection approach and caution points]**

- Since the V850E/MA1's $\overline{CS3}$ pin connects to the SDRAM's $\overline{CS}$ pin, the access timing is set in register SCR3.
- ★ Since the connection uses a 16-bit bus width, the V850E/MA1's address pins (A1 to A12, A21, A22) connect to the μPD4564163G5's address pins A0 to A13.
- Since the μPD4564163G5's addresses consist of a 12-bit row address and an 8-bit column address, when outputting the row address the address multiplex width (8-bit) is set as shown below to make the following settings for address pins A1 to A12 in the V850E/MA1.

| Row address | A12 | A11 | A10 | A9 | A8 | A7 | A6 | A5 | A4 | A3 | A2 | A1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | a20 | – – – – – – – – – – – – – – – – – – – – – – – – | | | | | | | | | a10 | a9 |

★ **Caution A21 and A22 in the V850E/MA1 are the μPD4564163G5's bank select addresses.**

- Since the μPD4564163G5-A10's refresh cycle is 4096 cycles/64 ms, the refresh interval should be set at 15.6 μs or less.
- The μPD4564163G5-A10's AC characteristics are described below. Based on these characteristics, the SCR3 register should be used with "$\overline{CAS}$ latency = 2" and "BCW = 1" and the idle state setting should be "0".
- The maximum clock frequency when $\overline{CAS}$ latency = 2 is 77 MHz.
- The minimum time between a refresh (or active) command and the next refresh (or active) command is 70 ns.
- The minimum time between an active command and a read/write command for the same bank is 20 ns.
- The maximum data float delay time[Note] from the clock rising edge is 7 ns.

**Note** V850E/MA1's AC characteristics ($t_{DSDOD}$) = (1 + i)T − 5
= 15 ns (> 7 ns) i = 0, T = 20 ns

**[Register settings]**

- Device type used with $\overline{CS3}$: SDRAM
- Bus width of $\overline{CS3}$ space: 16 bits
- Idle states: 0 states

**Remark** The settings in register DWC0 have no significance during SDRAM access. Also, the settings in register CSC0 have no significance because the $\overline{CS3}$ space is a fixed area.

**Figure 2-51. Settings in Bus Cycle Type Configuration Register 0 (BCT0)**

| Bit | 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 signal | | | $\overline{CS3}$ | | | | | $\overline{CS2}$ | | | $\overline{CS1}$ | | | | | $\overline{CS0}$ |

Memory controller enabled and SDRAM setting are specified for $\overline{CS3}$ space
Setting value in BCT0: 1011x00xx0xxx00xB

**Figure 2-52. Settings in Bus Size Configuration Register (BSC)**

| Bit | 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 signal | | $\overline{CS7}$ | | $\overline{CS6}$ | | $\overline{CS5}$ | | $\overline{CS4}$ | | $\overline{CS3}$ | | $\overline{CS2}$ | | $\overline{CS1}$ | | $\overline{CS0}$ |

16-bit data bus width is set for $\overline{CS3}$ space
Setting value in BSC: 0x0x0x0x010x0x0xB

**Figure 2-53. Settings in Bus Cycle Control Register (BCC)**

| Bit | 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 signal | $\overline{CS7}$ | | $\overline{CS6}$ | | $\overline{CS5}$ | | $\overline{CS4}$ | | $\overline{CS3}$ | | $\overline{CS2}$ | | $\overline{CS1}$ | | $\overline{CS0}$ | |

Number of idle states set in $\overline{CS3}$ space: 0
Setting value in BCC: xxxxxxxx00xxxxxxB

**Figure 2-54. Settings in SDRAM Configuration Register 3 (SCR3)**

| Bit | 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 to LTM03: 0, 1, 0 (CAS latency = 2)
BCW13, BCW03: 0, 1 (Waits between commands: BCW = 1)
SSO13, SSO03: 0, 1 (Data bus width = 16 bits)
RAW13, RAW03: 0, 1 (Row address width: 12)
SAW13, SAW03: 0, 0 (Address multiplex width = 8 bits)
Setting value in SCR3: 2054H

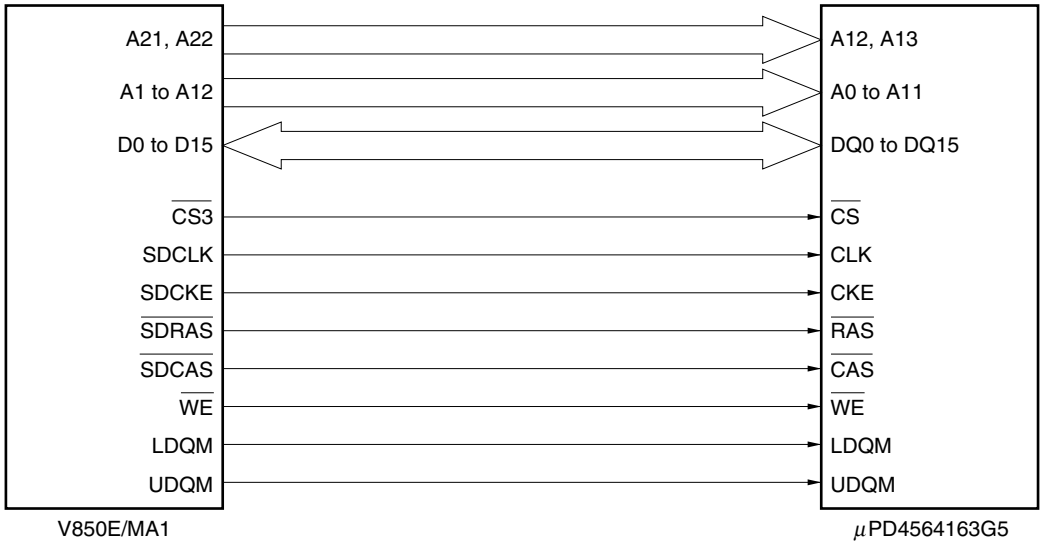
**Figure 2-55. Settings in Refresh Control Register 3 (RFS3)**

| Bit | 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 |

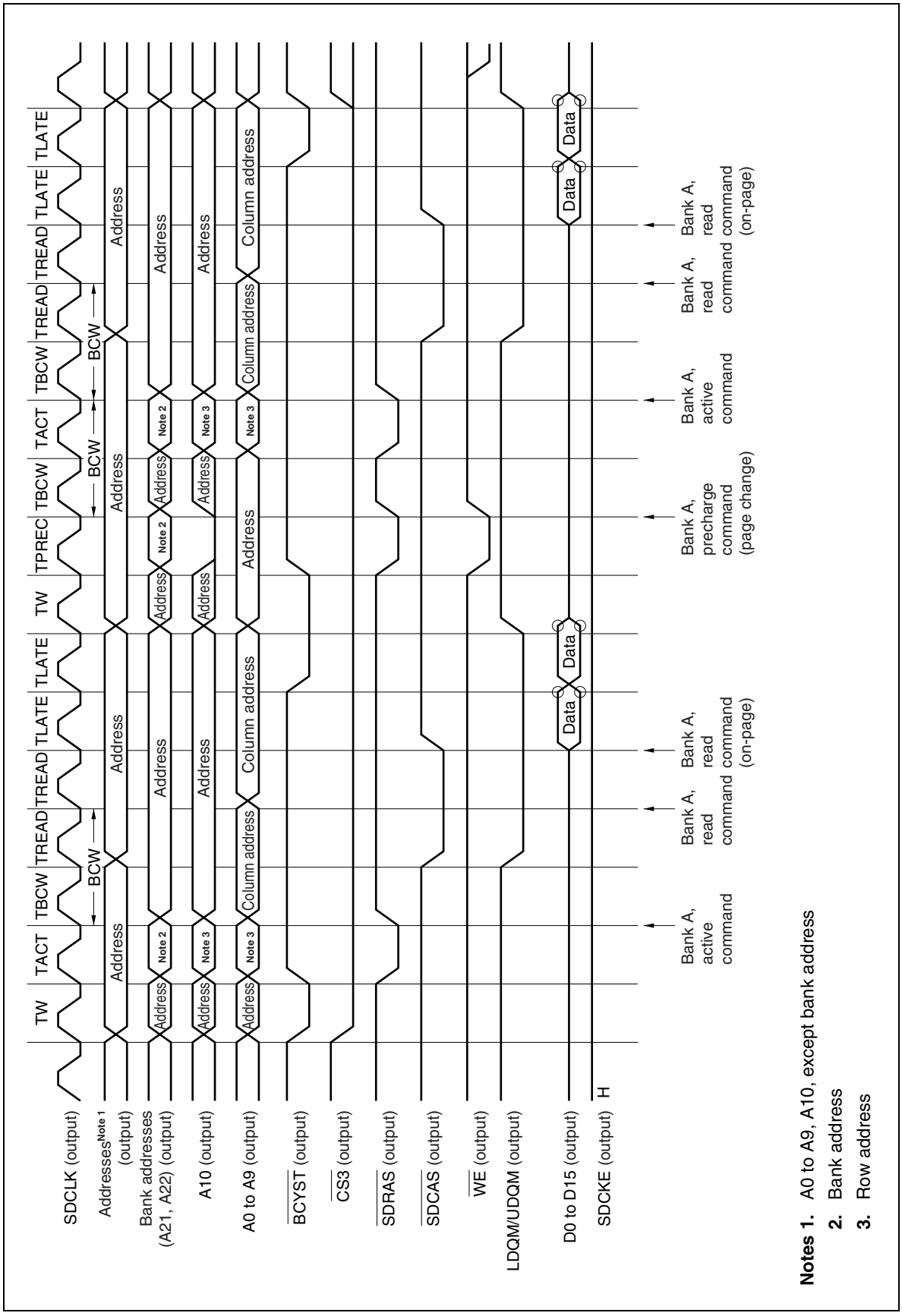
REN1: 1 (Refresh is enabled)
RCC11, RCC01: 0, 0 (Refresh count clock: $32/f_{XX}$)
RIN15 to RIN10: 0, 1, 0, 1, 1, 1 (interval factor: 24)
RFS3 setting value: 8017H
Given that $f_{XX}$ = 50 MHz (0.02 $\mu$s), the refresh interval for the above settings is as follows.
Refresh interval = $24 \times 0.02 \times 32$
= 15.4 $\mu$s (< 15.6 $\mu$s)

**Figure 2-56. Example of $\mu$PD4564163G5-A10 Connection Circuit**

★



★ **Figure 2-57. Read Operation of μPD4564163G5-A10**




**Notes 1.** A0 to A9, A10, except bank address

**2.** Bank address

**3.** Row address

# CHAPTER 3 BUS INTERFACE CONNECTION CIRCUIT EXAMPLES (2)

This chapter describes memory that cannot be directly connected to the V850E/MA1's bus interface and therefore uses an added circuit. The following approach was used to configure the circuit examples shown below.

- V850E/MA1's internal system clock and bus cycle frequency is 50 MHz
- Unless otherwise specified, the $\overline{\text{WAIT}}$ pin is not used
- 8xH is set in the BCP register and any value is set in the BEC register
- An external bus master is not connected, except as described in **3.5 Connection of External Refresh Unit Using $\overline{\text{HLDRQ}}$ Signal** ($\overline{\text{HLDRQ}}$ signal is not used except as described in section 3.5).

The signal propagation delay time using an attached circuit is calculated as between 2 ns (Min.) and 7 ns (Max.). In an actual circuit, make sure that the delay time takes into account the actually used devices.

This chapter provides an abbreviated version of the "Register settings" in CHAPTER 2.

## 3.1 Connection with 16-Bit SRAM

The following is an example of a circuit that uses one SRAM ($\mu$PD434016ALE: 256 Kbits $\times$ 16 bits) to connect a 512 KB space.

**[Circuit configuration]**

- Internal system clock: 50 MHz
- Connected device: $\mu$PD434016ALE-12 $\times$ 1
- $\overline{\text{CS}}$ signal used: $\overline{\text{CS5}}$
  Assigned to address range FA00000H to FA7FFFFH in external memory space (block 5).

**[Connection approach and caution points]**

The $\mu$PD434016ALE-12's address bus connects to pins A1 to A18 on the V850E/MA1. The $\overline{\text{CS}}$, $\overline{\text{OE}}$, $\overline{\text{LB}}$, and $\overline{\text{UB}}$ pins connect respectively to the V850E/MA1's $\overline{\text{CS5}}$, $\overline{\text{RD}}$, $\overline{\text{LBE}}$, and $\overline{\text{UBE}}$ pins. Since the V850E/MA1 is not equipped with a $\overline{\text{WE}}$ signal for SRAM, the $\mu$PD434016ALE-12's $\overline{\text{WE}}$ pin is created from the $\overline{\text{LWR}}$ and $\overline{\text{UWR}}$ pins.

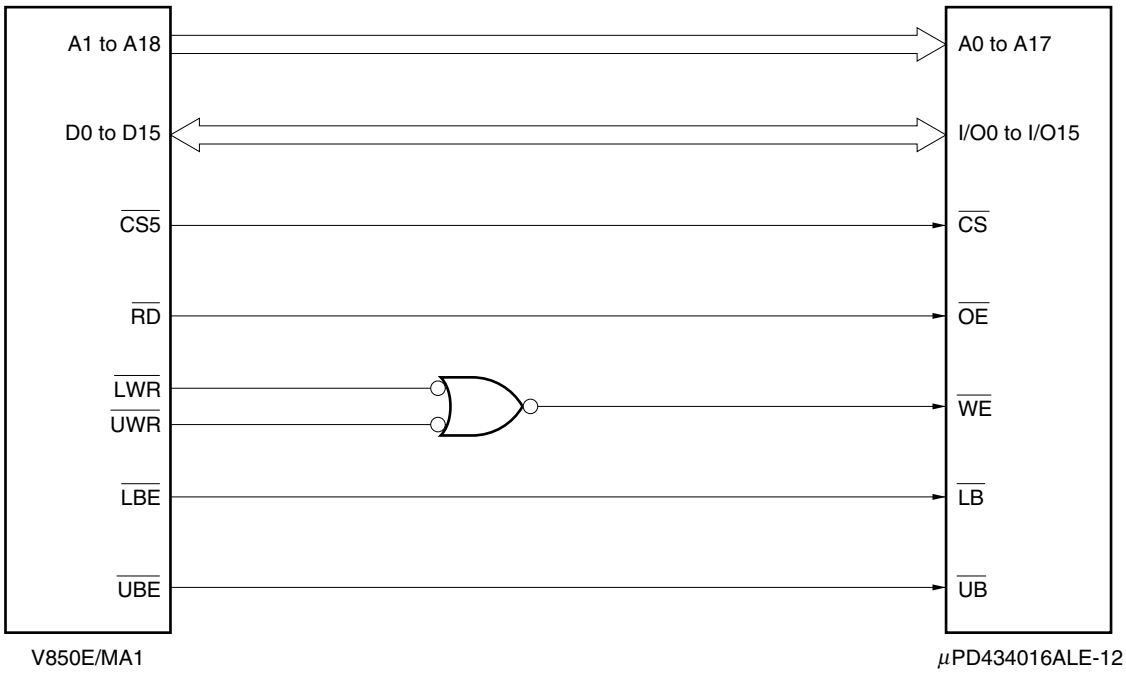
**Caution The $\overline{\text{LBE}}$ signal is multiplexed with the function of the SDRAS signal and the $\overline{\text{UBE}}$ signal is multiplexed with the function of the $\overline{\text{SDCAS}}$ signal.**
**To set the $\overline{\text{UBE}}$ pin to function as the $\overline{\text{LBE}}$ pin, bits 2 and 3 in the PFCCD register must be set to select the $\overline{\text{LBE}}$ signal and $\overline{\text{UBE}}$ signal functions. When connecting both an SDRAM and a 16-bit SRAM to the V850E/MA1, the $\overline{\text{SDRAS}}$ signal and $\overline{\text{SDCAS}}$ signal are selected, and the $\overline{\text{UBE}}$ signal and $\overline{\text{LBE}}$ signal are created from the V850E/MA1's $\overline{\text{RD}}$, $\overline{\text{LWR}}$, and $\overline{\text{UWR}}$ pins (see Figure 3-1 $\mu$PD434016ALE-12 Connection Circuit Example).**

**[Register settings]**

**Table 3-1. Connection with 16-Bit SRAM**

| Register Name | Setting value | Function |
|---|---|---|
| CSC1 | xxxx0100xxxxx0xxB | Block 5: $\overline{CS5}$ output |
| BCT1 | x00xx0xx1000x0xxB | $\overline{CS5}$: Memory controller enabled, SRAM, external I/O |
| BSC | 0x0x010x0x0x0x0xB | $\overline{CS5}$: 16 bits |
| DWC1 | 0xxx0xxx00010xxxB | $\overline{CS5}$: 1 wait |
| ASC | xxxx00xxxxxxxxxxB | $\overline{CS5}$: Address setup waits = 0 |
| BCC | xxxx01xxxxxxxxxxB | $\overline{CS5}$: Idle state 1 |

**Figure 3-1. μPD434016ALE-12 Connection Circuit Example**




(Reference circuit) Example of connection circuit when $\overline{\text{UBE}}$ and $\overline{\text{LBE}}$ signals cannot be used due to conflict with SDRAM)

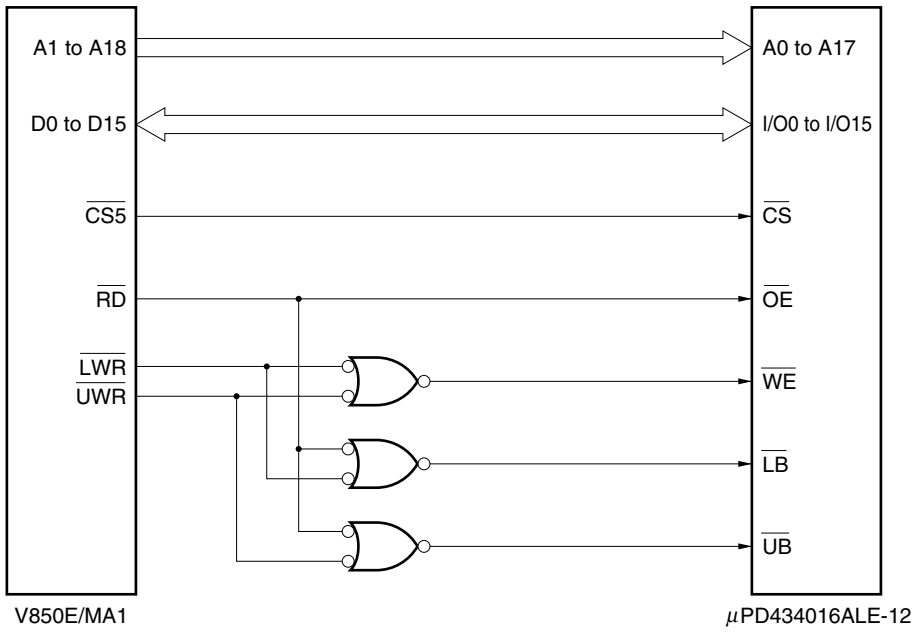


**Figure 3-2. Read Operation of μPD434016ALE-12 (16-Bit Access) (1/2)**

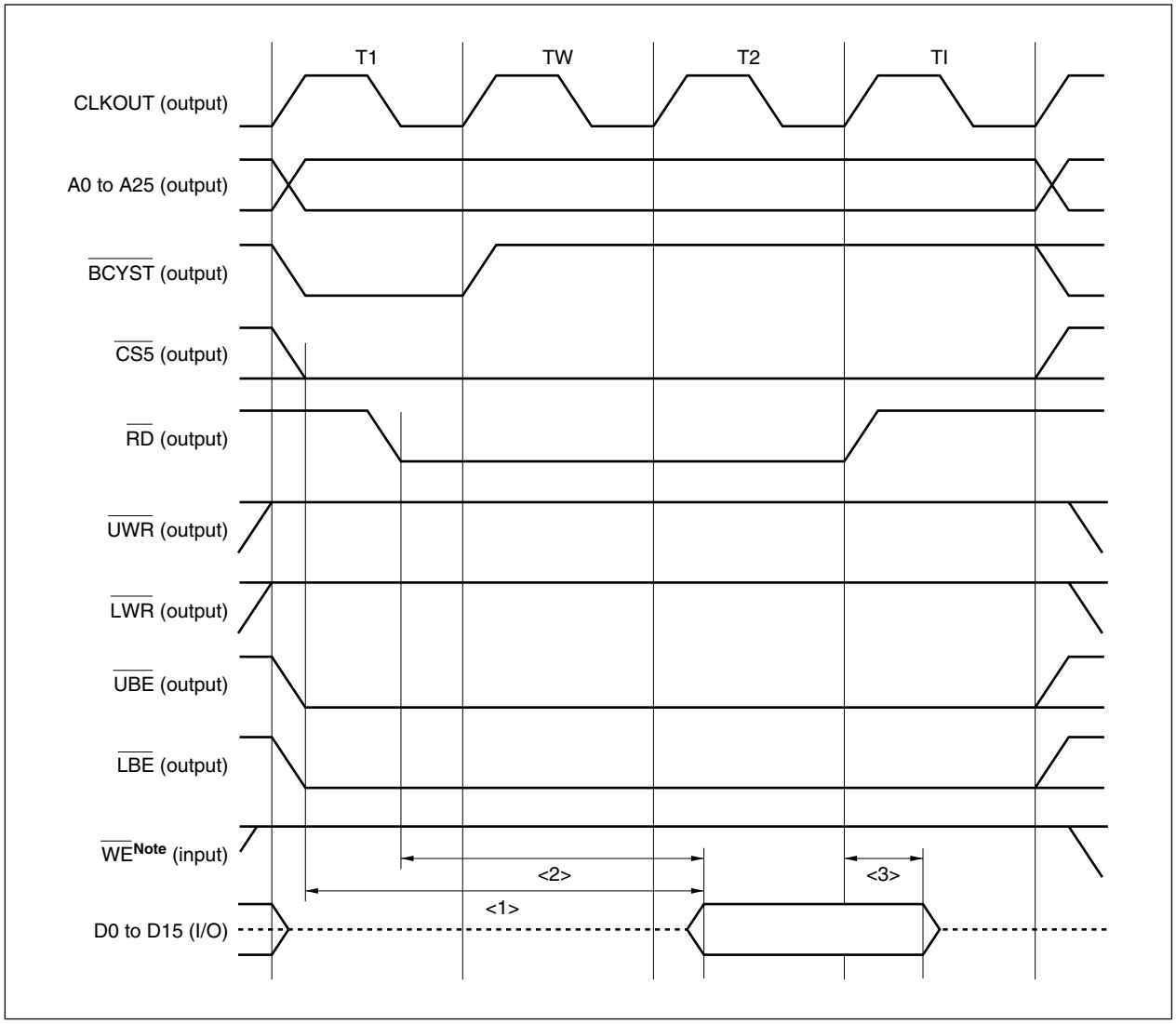


**Figure 3-2. Read Operation of μPD434016ALE-12 (16-Bit Access) (2/2)**

<1> Output delay time from active state transition of address and $\overline{CS}$ in μPD434016ALE-12: 12 ns (Max.)
Based on the V850E/MA1's electrical specifications, the maximum data input setup time (to address) is as follows.

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

<2> Output delay time from active state transition of $\overline{OE}$ in μPD434016ALE-12: 6 ns (Max.)
Based on the V850E/MA1's electrical specifications, the maximum data input setup time (to $\overline{RD}\downarrow$) is as follows.

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

<3> Output floating delay time from inactive state transition of $\overline{OE}$ in μPD434016ALE-12: 6 ns (Max.)
Based on the V850E/MA1's electrical specifications, the minimum data output delay time (from $\overline{RD}\uparrow$) is as follows.

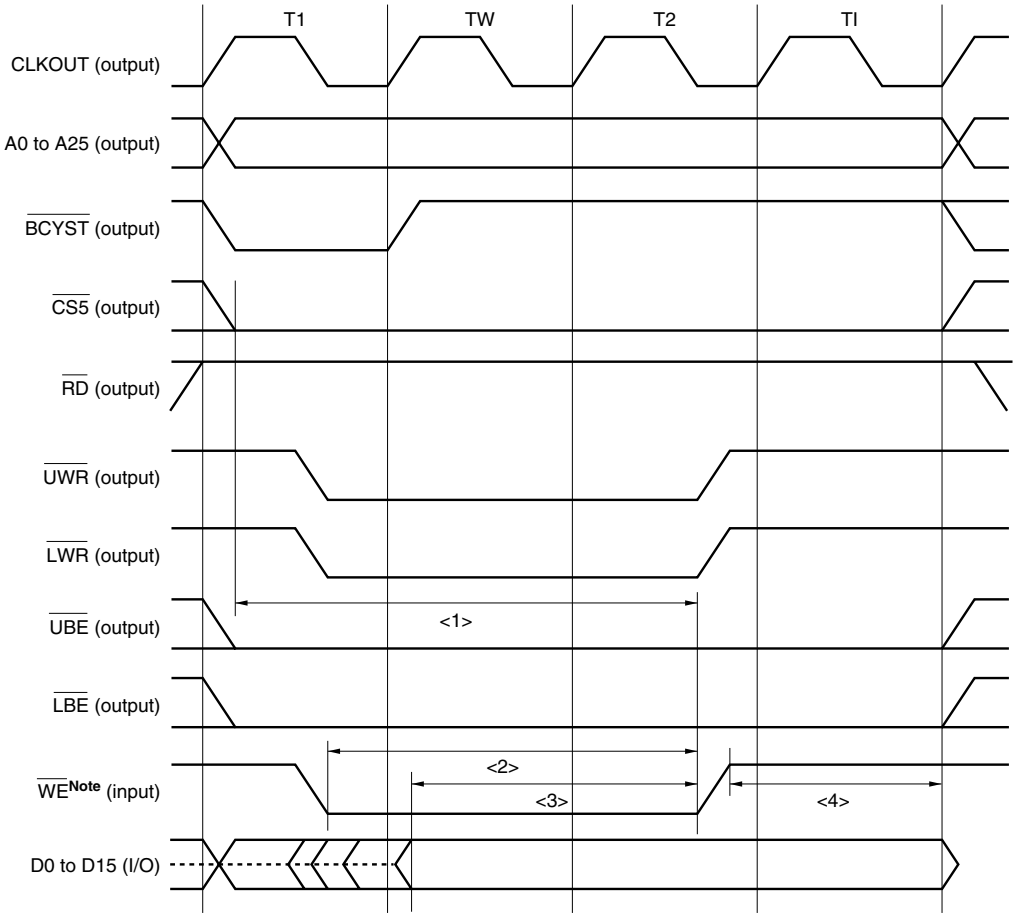
$t_{DRDOD}$ (ns) $= (0.5 + i)T - 10$
$= 1.5 \times 20 - 10$; i = 1, T = 20 ns
= 20 ns (> 6 ns)

**Note** $\overline{WE}$ is a pin on the μPD434016ALE-12.

**Remarks 1.** Broken lines indicate high impedance.

**2.** T: $t_{CYK}$ (CLKOUT output period)
w: Number of waits from $\overline{WAIT}$
$w_D$: Number of waits from DWC1, DWC2
$w_{AS}$: Number of address setup waits from ASC
i: Number of idle states

**Figure 3-3. Write Operation of μPD434016ALE-12 (16-Bit Access) (1/2)**




<1> Time until end of write operation after the μPD434016ALE-12's address and $\overline{CS}$ become active: 8 ns (Min.)

Based on the V850E/MA1's electrical specifications, the minimum address setup time (to $\overline{UWR}$ and $\overline{LWR}\uparrow$) is as follows.

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

<2> μPD434016ALE-12's $\overline{WE}$ active pulse width: 8 ns (Min.)

Based on the V850E/MA1's electrical specifications, the minimum low level width of $\overline{UWR}$ and $\overline{LWR}$ is as follows.

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

**Note** $\overline{WE}$ is a pin on the μPD434016ALE-12.

**Remarks** 1. Broken lines indicate high impedance.

2. T: $t_{CYK}$ (CLKOUT output period)
w: Number of waits from $\overline{WAIT}$
$w_D$: Number of waits from DWC1, DWC2
$w_{AS}$: Number of address setup waits from ASC

**Figure 3-3. Write Operation of μPD434016ALE-12 (16-Bit Access) (2/2)**

<3> Time until end of write operation after the μPD434016ALE-12's data becomes valid: 6 ns (Min.)

Based on the V850E/MA1's electrical specifications, the minimum data output setup time (to $\overline{UWR}$ and $\overline{LWR}\uparrow$) is as follows.

$t_{SODWR}$ (ns) $= (0.5 + w_{AS} + w + w_D)T - 10$

$= 1.5 \times 20 - 10$; $w_D = 1$, $w = 0$, $T = 20$ ns

$= 20$ ns (> 6 ns)

<4> Data hold time from end of μPD434016ALE-12's write operation: 0 ns (Min.)

Based on the V850E/MA1's electrical specifications, the minimum data output hold time (from $\overline{UWR}$ and $\overline{LWR}\uparrow$) is as follows.

$t_{HWROD}$ (ns) $= (0.5 + i)T - 10$

$= 1.5 \times 20 - 10$; $i = 1$, $T = 20$ ns

$= 20$ ns

When the maximum delay time due to the added circuit for $\overline{WE}$ is taken into account:

$t_{HWROD}$ – delay time of added circuit $= 20 - 7$

$= 13$ ns (> 0 ns)

**Remark** T: $t_{CYK}$ (CLKOUT output period)

w: Number of waits from $\overline{WAIT}$

$w_D$: Number of waits from DWC1, DWC2

$w_{AS}$: Number of address setup waits from ASC

i: Number of idle states

## 3.2 Connection with Low-Speed Device (μPD4991A)

This example includes a calendar/timer function IC (μPD4991A) that is connected to an external data bus.

**[Circuit configuration]**

- Internal system clock: 50 MHz
- Connected device: μPD4991A × 1
- $\overline{CS}$ signal used: $\overline{CS7}$
  Assigned to address range FE00000H to FE0000FH (block 7) in external memory space. The address range from FE00010H to FFFBFFFH is used for the image.

**[Connection approach and caution points]**

- The μPD4991A is used in normal operation ($V_{DD}$ = 3.3 V).
  When $V_{DD}$ = 3.3 V, the μPD4991A's access time (from when the $\overline{CS}$ signal becomes active during a read operation until the data has been confirmed) is 210 ns (Min.). This access time can be met by using the V850E/MA1's address setup wait and data wait functions.
- When $V_{DD}$ = 3.3 V, the μPD4991A's data output floating delay time increases to 70 ns (Max.). Since the V850E/MA1's idle state insertion function is inadequate during 50 MHz operation, a bus buffer must be used to connect the data bus.
- Pins A0 to A3 on the V850E/MA1 connect to pins A0 to A3 on the μPD4991A.

  **Remark** When A0 to A3 on the μPD4991A is connected to A0 to A3 on the V850E/MA1, specify an 8-bit bus width. When connecting via a 16-bit bus width space, connect to A1 to A4 on the V850E/MA1. In either case, only byte access is valid (with 16-bit bus width, access occurs via even-numbered addresses during little endian operation and occurs via odd-numbered addresses during big endian operation), and only the lower four bits of data are valid.

- The V850E/MA1's $\overline{CS7}$ pin, $\overline{RD}$ pin[Note], and $\overline{LWR}$ pin[Note] connect respectively to the μPD4991A's $\overline{CS1}$ pin, $\overline{OE}$ pin, and $\overline{WE}$ pin.

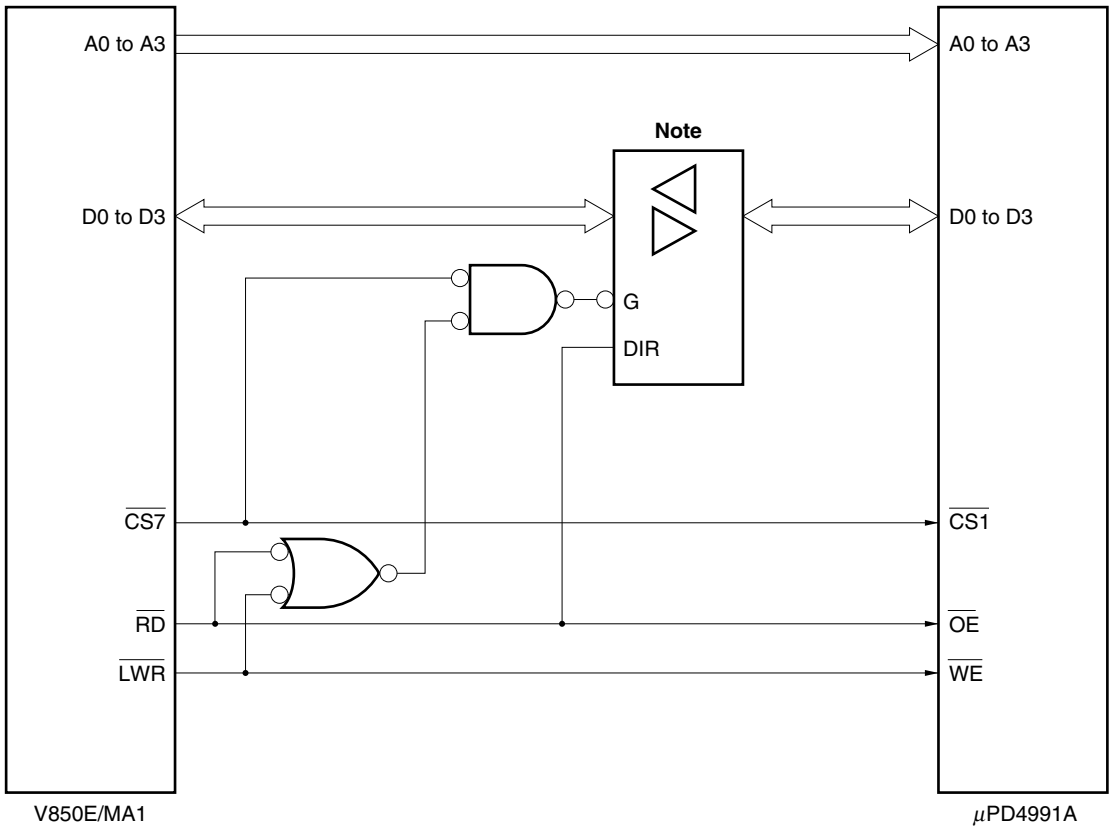
  **Note** This is similar to connecting the $\overline{IORD}$ and $\overline{IOWR}$ pins when "1" has been set to the IOEN bit in the BCP register. The $\overline{RD}$ pin, $\overline{IORD}$ pin, $\overline{LWR}$ pin (or $\overline{UWR}$ pin), and $\overline{IOWR}$ pin all operate in the same way except during DMA flyby transfers.

**[Register settings]**

**Table 3-2. Connection with Low-Speed Devices**

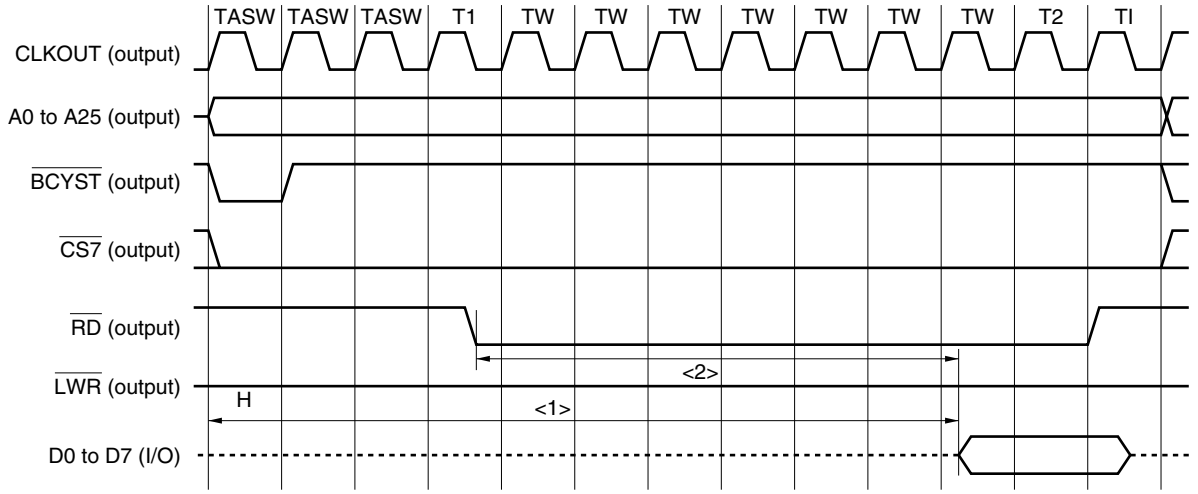
| Register Name | Setting value | Function |
|---|---|---|
| CSC1 | xxxxxxxxxxxxxxx1B | Block 7: $\overline{CS7}$ |
| BCT1 | 1000x0xxx00xx0xxB | $\overline{CS7}$: Memory controller enabled, SRAM, external I/O |
| BSC | 000x0x0x0x0x0x0xB | $\overline{CS7}$: 8 bits |
| DWC1 | 01110xxx0xxx0xxxB | $\overline{CS7}$: Data wait 7 |
| ASC | 11xxxxxxxxxxxxxxB | $\overline{CS7}$: Address setup wait 3 |
| BCC | 01xxxxxxxxxxxxxxB | $\overline{CS7}$: Idle state 1 |

**Figure 3-4. Example of μPD4991A Connection Circuit**




**Note** This is assumed to be a circuit equivalent to the 74xx245.

**Figure 3-5. Read Operation of μPD4991A**

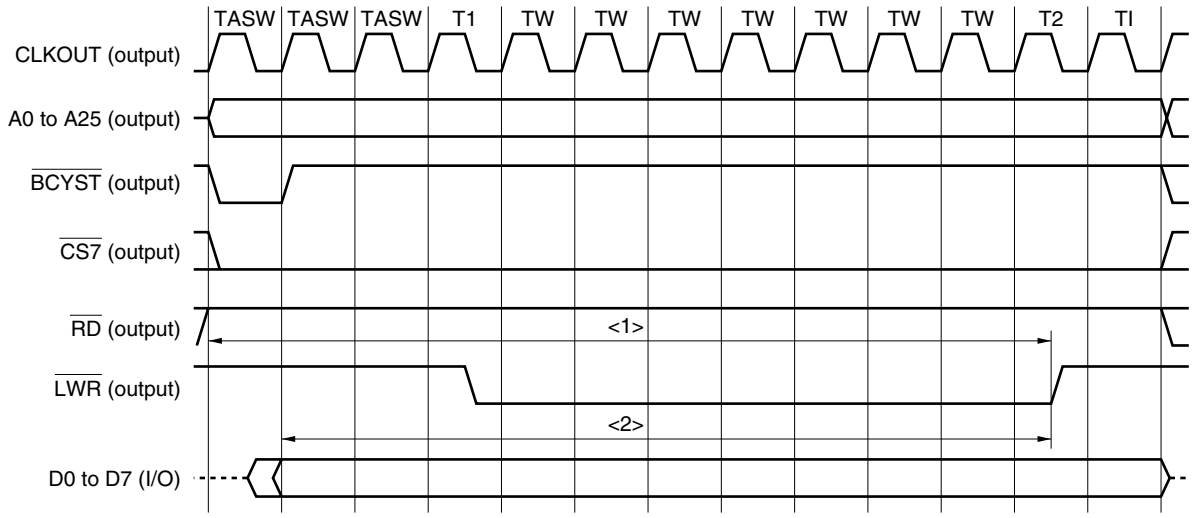



<1> Output delay time from active state transition of $\overline{CS1}$ signal in μPD4991A ($V_{DD}$ = 3.3 V): 210 ns (Max.)

<2> Output delay time from active state transition of $\overline{OE}$ signal in μPD4991A ($V_{DD}$ = 3.3 V): 110 ns (Max.)

**Remark** Broken lines indicate high impedance.

**Figure 3-6. Write Operation of μPD4991A**




<1> Time until end of write operation after the μPD4991A's ($V_{DD}$ = 3.3 V) $\overline{CS1}$ signal become active: 170 ns (Min.)

<2> Time until end of write operation after the μPD4991A's ($V_{DD}$ = 3.3 V) data becomes valid: 100 ns (Min.)

**Remark** Broken lines indicate high impedance.

## 3.3 $\overline{\text{WAIT}}$ Pin Control

### 3.3.1 $\overline{\text{WAIT}}$ pin control example (1) (connection with μPD63310)

In this example, a stereo sound codec (the μPD63310) is connected to an external data bus.

During the μPD63310's read operation, 200 ns are required for the $\overline{\text{RB}}$ signal (connected to the V850E/MA1's $\overline{\text{RD}}$ pin), so the $\overline{\text{WAIT}}$ pin is used to connect to the V850E/MA1 with an external wait control function.

**[Circuit configuration]**

- Internal system clock: 50 MHz
- Connected device: μPD63310GK × 1
- The μPD63310GK is used at $V_{DD}$ = 5 V.
- $\overline{\text{CS}}$ signal used: $\overline{\text{CS7}}$
  Assigned to address range FE00000H to FE00001H (block 7) in external memory space.

**[Connection approach and caution points]**

- The V850E/MA1's $\overline{\text{CS7}}$ pin, $\overline{\text{RD}}$ pin[Note 2], and $\overline{\text{LWR}}$ pin[Note 2] are respectively connected via a level-converting IC[Note 1] to the μPD63310GK's $\overline{\text{CSB}}$ pin, $\overline{\text{RB}}$ pin, and $\overline{\text{WB}}$ pin.

  **Notes 1.** When using the μPD63310GK at $V_{DD}$ = 5 V, $V_{IH}$ = 4.5 V (Min.). Therefore, the connection must be made via a device (such as a 74AC Series device) that meets the requirement of $V_{OH}$ = 4.5 V (Min.).
  **2.** This is the same as when the $\overline{\text{IORD}}$ and $\overline{\text{IOWR}}$ pins are connected (see **3.2 Connection with Low-Speed Device (μPD4991A)**).

- During a read operation, 200 ns (Min.) are required for the μPD63310$\overline{\text{GK}}$'s RB signal. During a write operation, 120 ns (Min.) are required for the WB signal. Consequently, the $\overline{\text{WAIT}}$ pin is used to connect to the V850E/MA1 with an externally added circuit.
- $\overline{\text{WAIT}}$ pin control using an externally added circuit is designed so that active transition of the $\overline{\text{WAIT}}$ pin occurs at the rising edge of the T1 state during a read operation, followed by an inactive transition that occurs at the rising edge of CLKOUT after the required number of waits have been counted. During a write operation, the WAIT pin does not become active and only software-based wait control is used. Consequently, a value that satisfies the requirements of the write operation (in this case, the value is 6) is set for software-based wait control in the $\overline{\text{CS7}}$ space.

  **Remark** The inserted waits are the greater of either the ORed result of the waits due to $\overline{\text{WAIT}}$ pin control or the waits due to the programmable wait setting.
  In this example, the $\overline{\text{WAIT}}$ pin becomes active after the T1 state, so when using $\overline{\text{WAIT}}$ pin control for both read and write operations, a programmable wait must be inserted for at least first wait.

- The D0 to D5 and A0 pins on the V850E/MA1 are connected respectively via a level conversion IC to the D0 to D5 and SELR pins on the μPD63310GK.

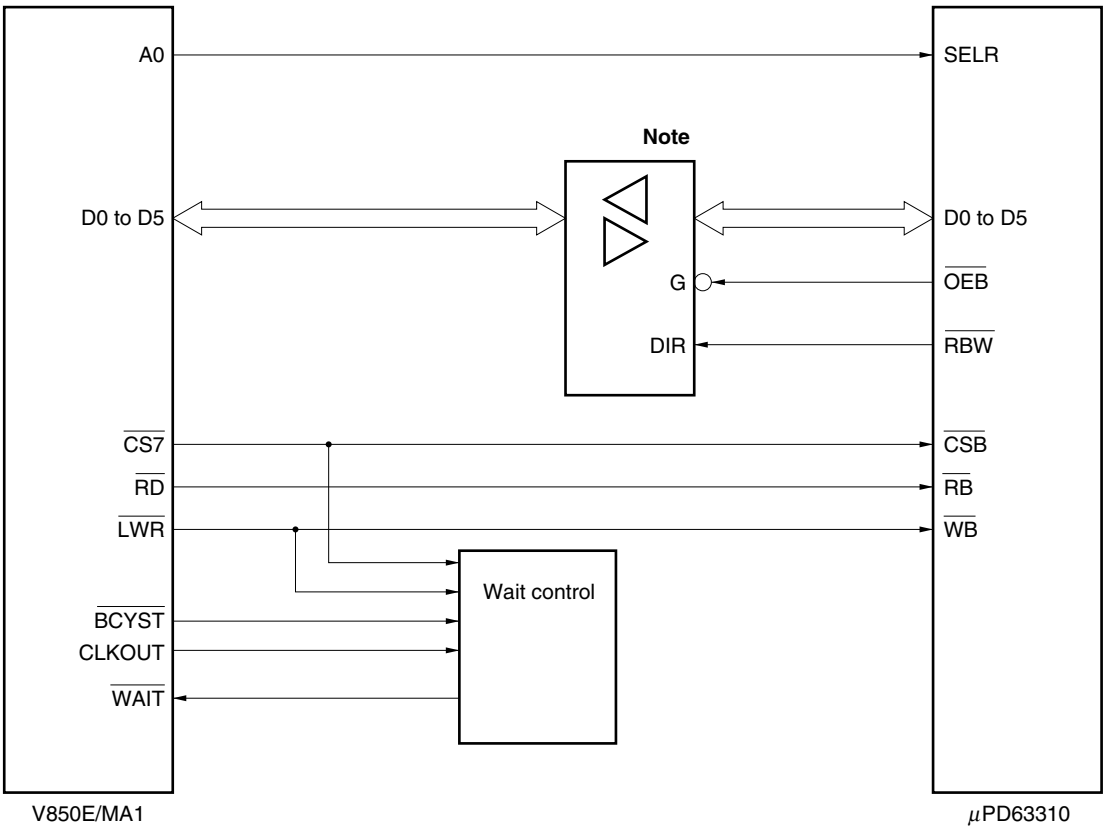
  **Remark** When the μPD63310GK's SELR pin is connected to the V850E/MA1's A0 pin, specify an 8-bit bus width. When connecting via a 16-bit bus width space, connect the μPD63310's SELR pin to the V850E/MA1's A1 pin. In either case, only byte access is valid (with 16-bit bus width, access performed to even-numbered addresses during little endian operation and performed to odd-numbered addresses during big endian operation), and only the lower five bits of data are valid.

**[Register settings]**

**Table 3-3. Connection with μPD63310**

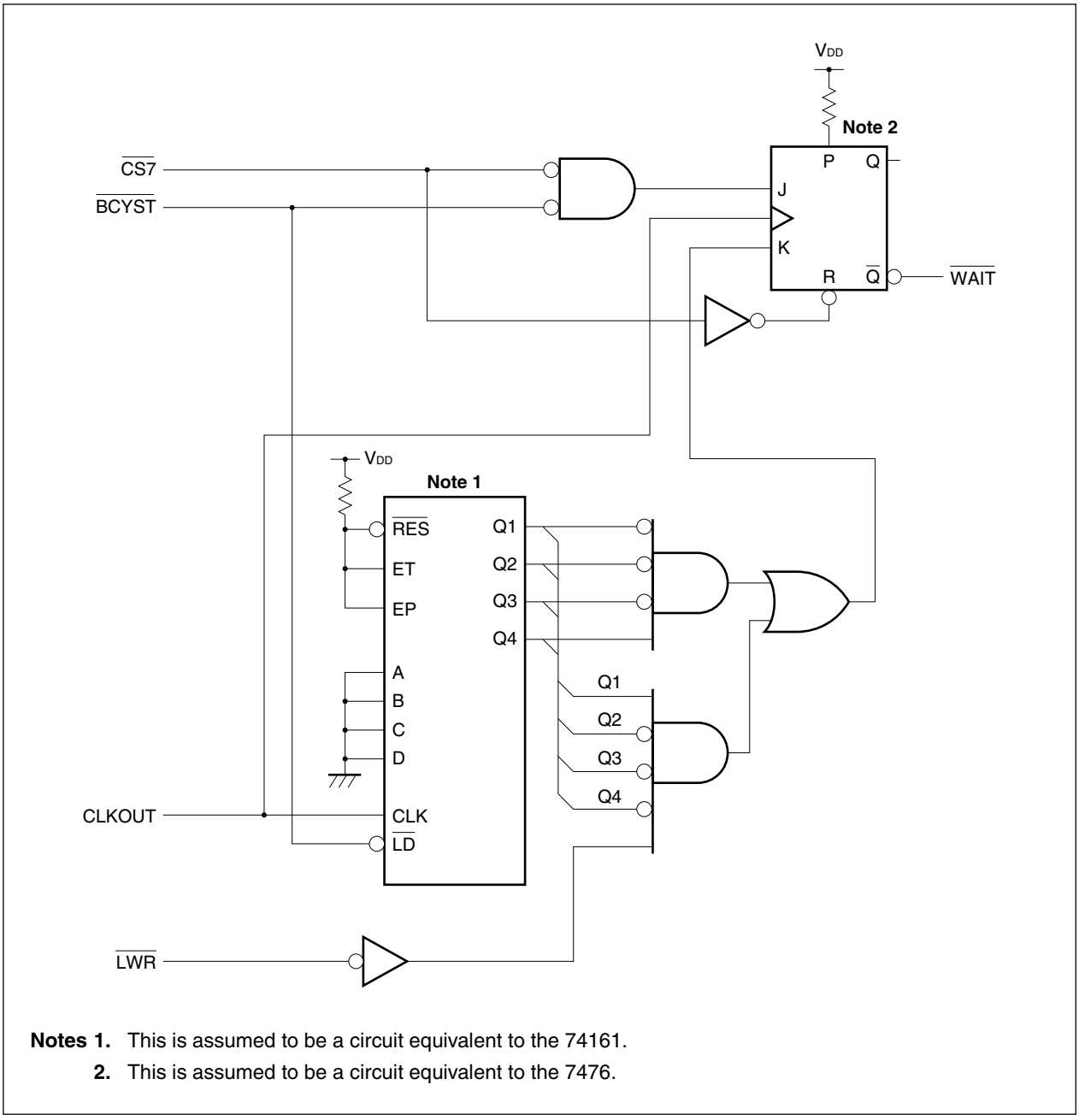
| Register Name | Setting value | Function |
|---|---|---|
| CSC1 | xxxxxxxxxxxxxxx1B | Block 7: $\overline{CS7}$ |
| BCT1 | 1000x0xxx00xx0xxB | $\overline{CS7}$: Memory controller enabled, SRAM, external I/O |
| BSC | 000x0x0x0x0x0x0xB | $\overline{CS7}$: 8 bits |
| DWC1 | 01100xxx0xxx0xxxB | $\overline{CS7}$: Data wait 6 |
| ASC | 00xxxxxxxxxxxxxxB | $\overline{CS7}$: Address setup wait 0 |
| BCC | 10xxxxxxxxxxxxxxB | $\overline{CS7}$: Idle state 2 |

**Figure 3-7. Example of μPD63310 Connection Circuit**




**Note** This is assumed to be a circuit equivalent to the 74xx245.

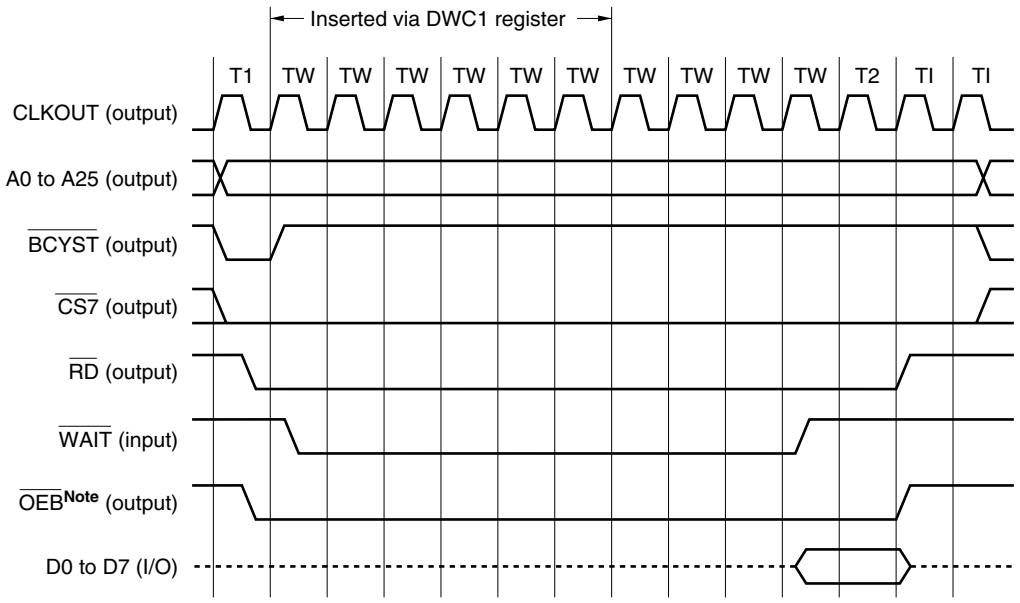
**Figure 3-8. Circuit Configuration of Wait Control Block**




**Notes 1.** This is assumed to be a circuit equivalent to the 74161.

**2.** This is assumed to be a circuit equivalent to the 7476.

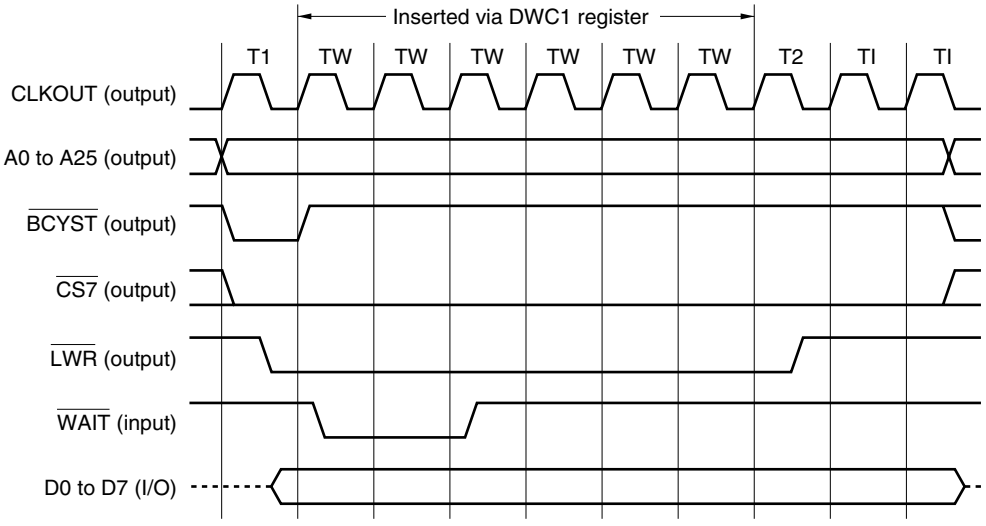
**Figure 3-9. Read Operation of μPD63310**




**Note** This pin is on the μPD63310.

**Remark** Broken lines indicate high impedance.

**Figure 3-10. Write Operation of μPD63310**




**Remark** Broken lines indicate high impedance.

### 3.3.2 $\overline{\text{WAIT}}$ pin control example (2) (connection with dual port RAM)

In this example, a dual port RAM (the IDT7007S20: 32 Kbits × 8 bits) is used to externally connect a 32 KB external memory space.

**[Circuit configuration]**

- Internal system clock: 50 MHz
- Connected device: IDT7007S20 × 1
- $\overline{\text{CS}}$ signal used: $\overline{\text{CS5}}$
  Assigned to address range FA00000H to FA07FFFH (block 5) in external memory space.

**[Connection approach and caution points]**

- The left side of the IDT7007S20 is connected to the V850E/MA1 (the V850E/MA1's A0 to A14 pins, D0 to D7 pins, $\overline{\text{CS5}}$ pin, $\overline{\text{RD}}$ pin, and $\overline{\text{LWR}}$ pin are respectively connected to the IDT7007S20's $A_{0L}$ to $A_{14L}$ pins, $I/O_{0L}$ to $I/O_{7L}$ pins, $\overline{CE}_L$ pin, $\overline{OE}_L$ pin, and $R/\overline{W}_L$ pins).

  **Remark** The right side of the IDT7007S20 is left to be controlled by another processor.

- Pull up the IDT7007S20's $M/\overline{S}$ pin and do not use the $\overline{\text{SEM}}_L$ and $\overline{\text{INT}}_L$ pins.

  **Remark** $M/\overline{S}$: When using more than one IDT7007S20 device, use this bit to set them as master or slave devices. In this case, there is only one device, which is set to master mode.
  $\overline{\text{SEM}}_L$: This is a select pin for the semaphore function. This pin is not used in this example, and therefore should be pulled up.
  $\overline{\text{INT}}_L$: This is an interrupt pin that becomes active when the processor on the right side writes to a specified area. This pin is not used in this example, and therefore should be left open.

- Once the IDT7007S20's $\overline{\text{BUSY}}_L$ pin has become active, at least 20 ns is required from the rising edge of the $\overline{\text{BUSY}}_L$ pin before data can be confirmed. Therefore, a signal delayed by one clock cycle is ORed with the $\overline{\text{BUSY}}_L$ pin and the result is connected to the V850E/MA1's $\overline{\text{WAIT}}$ pin.
- Once the IDT7007S20's $\overline{CE}_L$ signal has become active, its $\overline{\text{BUSY}}_L$ pin requires up to 20 ns to confirm (including whether or not the $\overline{\text{BUSY}}_L$ pin becomes active), so 1 wait is set to DWC1[Note].

  **Note** This condition can be met by setting "address setup waits = 1", but "data waits = 1" is set in view of other AC characteristics (such as the V850E/MA1's $t_{SRDID}$).

- Since these connections are made using an 8-bit bus width, the setting value in the BSC register's $\overline{\text{CS1}}$ space is 8 bits.

**[Register settings]**

**Table 3-4. Connection with IDT7007S20**

| Register Name | Setting value | Function |
|---|---|---|
| CSC1 | xxxx0100xxxxx0xxB | Block 5: $\overline{CS5}$ |
| BCT1 | x00xx0xx1000x0xxB | $\overline{CS5}$: Memory controller enabled, SRAM, external I/O |
| BSC | 0x0x000x0x0x0x0xB | $\overline{CS5}$: 8 bits |
| DWC1 | 0xxx0xxx00010xxxB | $\overline{CS5}$: Data wait 1 |
| ASC | xxxxxxxxxxxx00xxB | $\overline{CS5}$: Address setup wait 0 |
| BCC | xxxxxxxxxxxx01xxB | $\overline{CS5}$: Idle state 1 |

**Figure 3-11. Example of IDT7007S20 Connection Circuit**

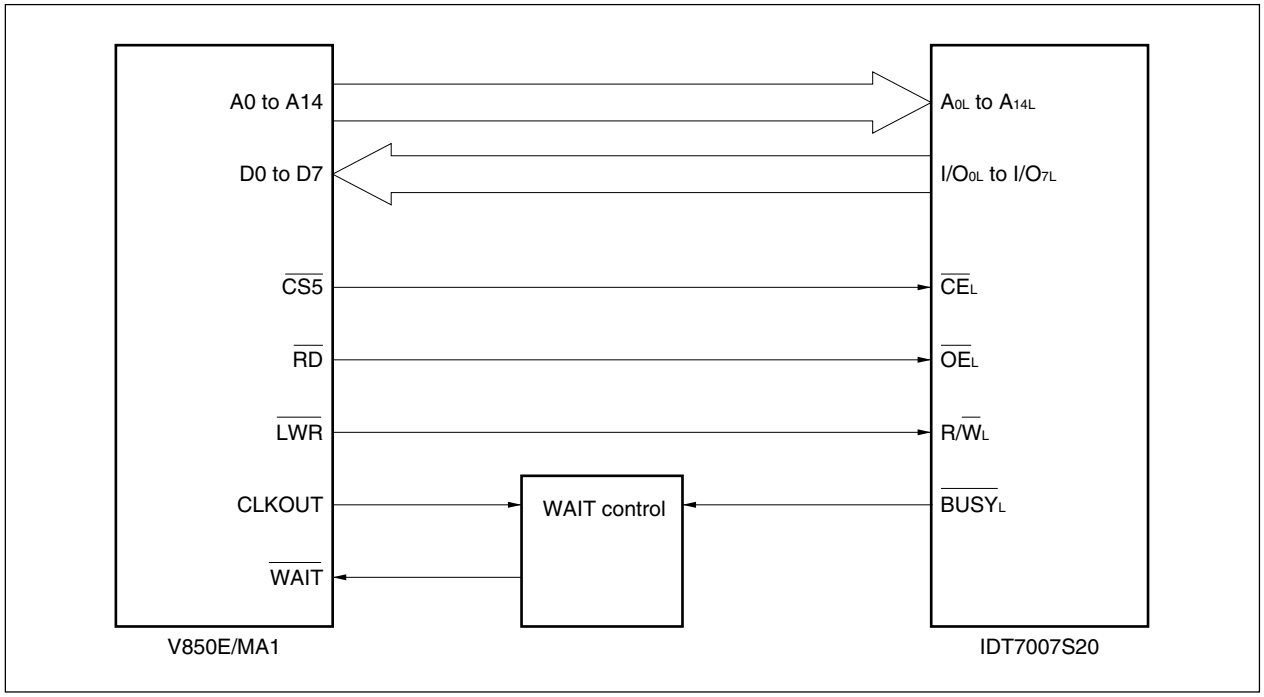


**Figure 3-12. Circuit Configuration of $\overline{\text{WAIT}}$ Control Block**

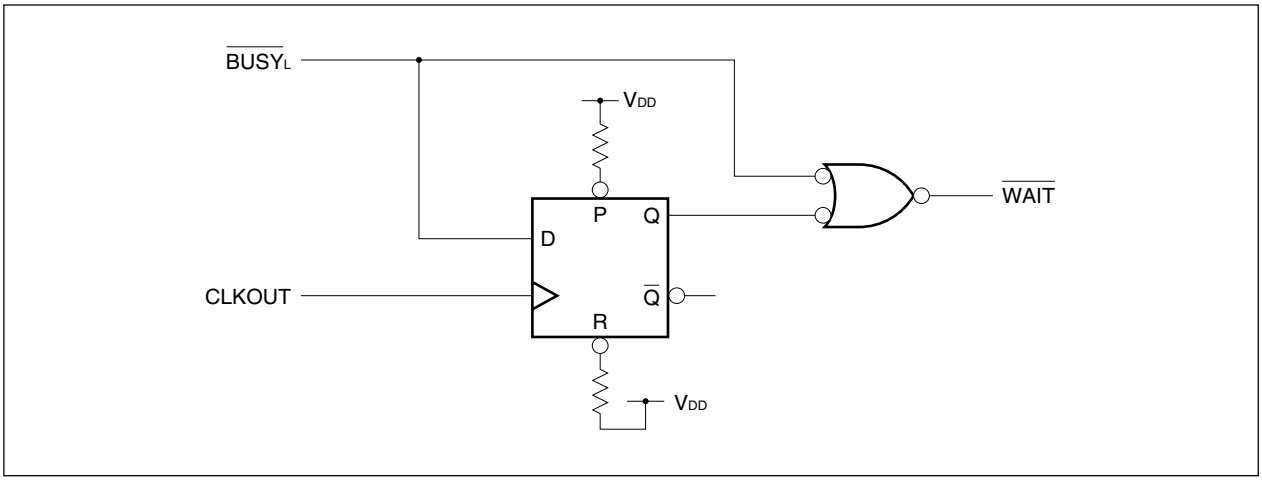
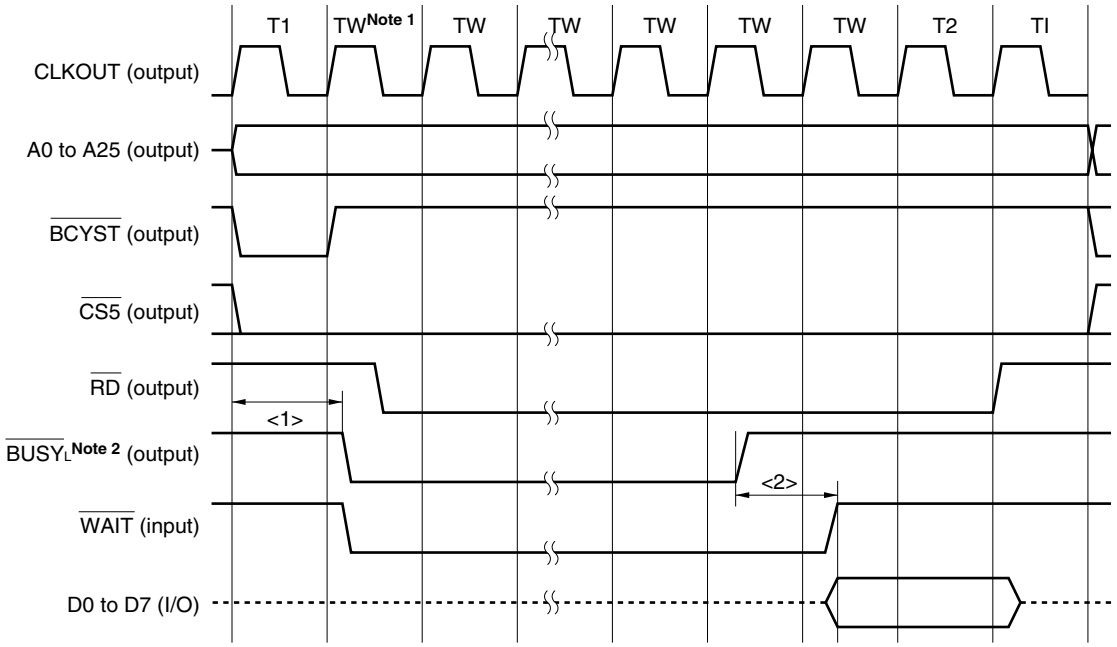



**Figure 3-13. Read Operation of IDT7007S20 (When $\overline{\text{BUSY}}_L$ Pin Is Active)**




<1> Time from active state transition of $\overline{\text{CE}}_L$ to $\overline{\text{BUSY}}_L$ in IDT7007S20 is confirmed: 20 ns (Max.)

<2> Time from when $\overline{\text{BUSY}}_L$ is canceled to when the data is confirmed in IDT7007S20: 20 ns (Max.)

**Notes 1.** One wait is inserted due to the setting values to the DWC register.

**2.** $\overline{\text{BUSY}}_L$ pin is on the IDT7007S20.

**Remark** Broken lines indicate high impedance.

## 3.4 Connection with Multiple EDO DRAMs (HM5164165FL-5)

The following is an example in which four EDO DRAMs (HM5164165FL: 4 Mbits × 16 bits) are connected (as a 32 MB external memory space with 16-bit bus width).

**[Circuit configuration]**

- Internal system clock: 50 MHz
- Connected device: HM5164165FL-5 × 4
- $\overline{CS}$ signal used: $\overline{CS4}$ ($\overline{RAS4}$) (Area 2)
  Assigned to address range 8000000H to 9FFFFFFH in external memory space.
  The address range from A000000H to BFFFFFFH is used for the image).

**[Connection approach and caution points]**

- The V850E/MA1's A1 to A13 pins, D0 to D15 pins, $\overline{RAS4}$ pin, $\overline{OE}$ pin and $\overline{WE}$ pin connect directly to the A0 to A12 pins, IO1 to IO16 pins, $\overline{RAS}$ pin, $\overline{OE}$ pin, and $\overline{WE}$ pin on the four HM5164165FL-5 devices.
- The V850E/MA1's $\overline{CAS}$ signals ($\overline{LCAS}$ and $\overline{UCAS}$) and the V850E/MA1's higher addresses (A23 and A24) are decoded and then connected to the four HM5164165FL-5 devices. The circuit is configured so that the connected $\overline{CAS}$ signal becomes active even after a refresh operation.

  **Remark** There is also a method that decodes the $\overline{RAS}$ signal using higher addresses, but this higher address changes when the V850E/MA1 accesses the $\overline{CS}$ space, so $\overline{RAS}$ hold mode should not be used.

- The EDO DRAM's access timing is set in the SCR4 register.
- The settings in the SCR4 register are the same as were described in **2.5 Connection with EDO DRAM** except in case where delay of the $\overline{CAS}$ signal due to an added circuit must be considered.

  **Caution In this circuit example, "$w_{DA}$ = 2" is set since "$w_{DA}$ = 1" does not satisfy the $\overline{CAS}$ access time ($t_{DH}$) when the 7 ns delay (Max.) of the $\overline{CAS}$ signal due to an added circuit is taken into account. In other case, connection settings are the same as described in 2.5 Connection with EDO DRAM.**

**[Register settings]**

**Table 3-5. Connection with Multiple EDO DRAMs**

| Register Name | Setting value | Function |
|---|---|---|
| BCT1 | x00xx0xxx00x1010B | $\overline{CS4}$: Memory controller enabled, EDO DRAM |
| BSC | 0x0x0x010x0x0x0xB | $\overline{CS4}$: 16 bits |
| BCC | xxxxxx01xxxxxxxxB | $\overline{CS4}$: Idle state 1 |
| SCR4 | A645H | On-page access enabled, $w_{RP}$ = 2, $w_{RH}$ = 1, $w_{DA}$ = 2, $w_{CP}$ = 1, 16-bit width, multiplex width = 9 bits |
| RFS4 | 8017H | Refresh enabled, refresh count clock = 32/$f_{XX}$, interval factor = 24 |
| RWC | 40H | RRW = 1, RCW = 0, SRW = 0 |

**Cautions 1. Since area 2 is fixed to $\overline{CS4}$ ($\overline{RAS4}$), settings in the CSC0 and CSC1 registers have no significance.**

**2. During EDO DRAM access, settings in DWC0, DWC1 registers and ASC register have no significance.**

**Remark** $w_{RP}$: Number of row address precharge waits
$w_{RH}$: Number of row address hold waits
$w_{DA}$: Number of data waits
$w_{CP}$: Number of column address precharge waits
RRW: Number of refresh $\overline{RAS}$ waits
RCW: Number of refresh cycle waits
SRW: Number of self refresh cancellation waits

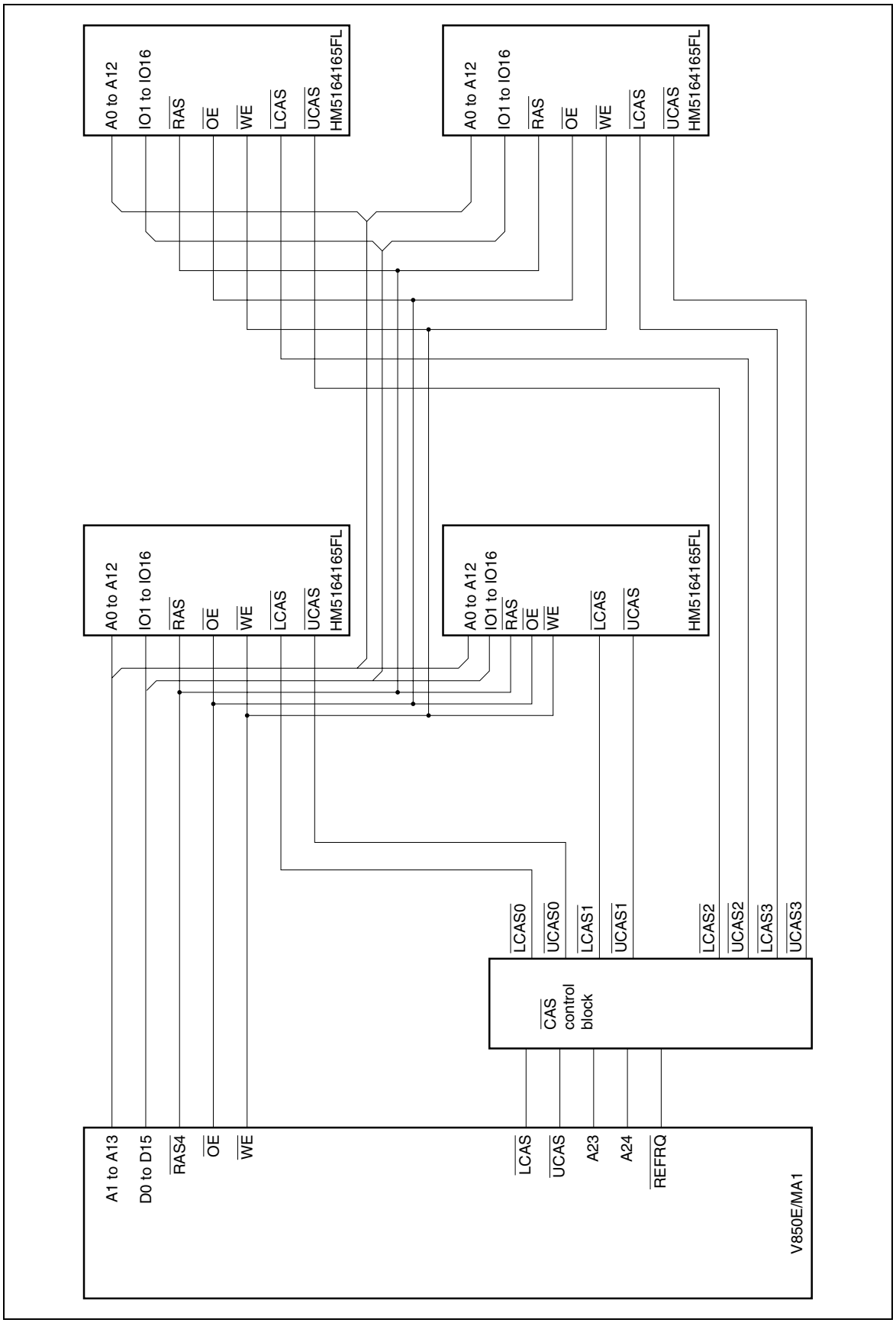
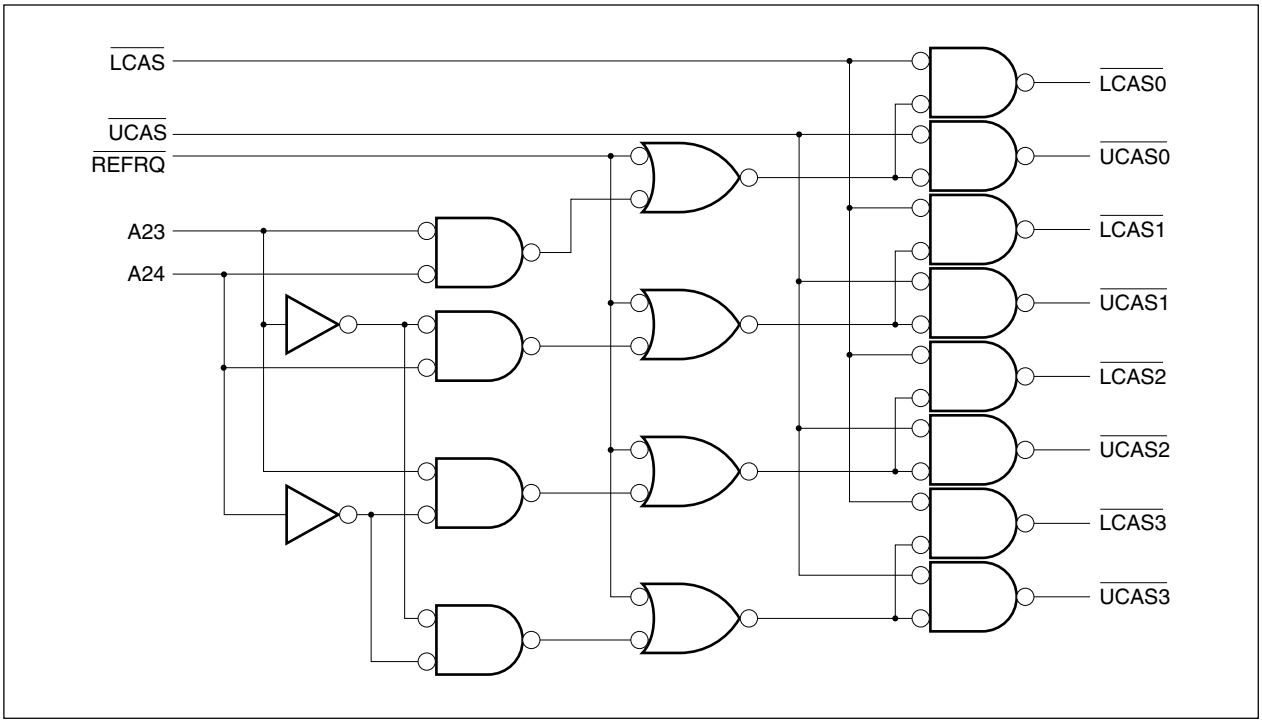



**Figure 3-14. Example of HM5164165FL-5 (×4) Connection Circuit**

**Figure 3-15. Detailed View of $\overline{\text{CAS}}$ Control Block**



## 3.5 Connection with External Refresh Unit Using $\overline{\text{HLDRQ}}$ Signal

In the following example, an externally connected EDO DRAM's refresh function is controlled by an external unit.

**[Circuit configuration]**

- Internal system clock: 50 MHz
- Connected device: HM5164165FL-5 × 1
  Except for the refresh control function, the circuit configuration (using $\overline{\text{RAS4}}$ as the $\overline{\text{CS}}$ signal) is the same as that described in **2.5 Connection with EDO DRAM**.
- External refresh unit:
  The $\overline{\text{HLDRQ}}$ pin is set as active after each specified period and, once the V850E/MA1 has been switched to hold mode, CBR refresh is performed twice before the $\overline{\text{HLDRQ}}$ pin is set as inactive.

**[Connection approach and caution points]**

- The V850E/MA1's $\overline{\text{HLDRQ}}$ pin is set as active when triggered by a 30 $\mu$s clock cycle. Once the $\overline{\text{HLDAK}}$ pin is set to active level, CBR refresh control is performed twice. Design the circuit so that, after the refresh operations are completed, the $\overline{\text{HLDRQ}}$ pin is set as inactive.
- The EDO DRAM's $\overline{\text{RAS}}$ pin, $\overline{\text{LCAS}}$ pin, and $\overline{\text{UCAS}}$ pin are connected to the V850E/MA1's $\overline{\text{RAS4}}$ pin, $\overline{\text{LCAS}}$ pin, $\overline{\text{UCAS}}$ pin by decoding an external refresh control circuit.
- Set the SCR4 register so that the refresh function is prohibited.

**Caution Register settings other than those in the DRC4 and RWC registers and timing other than for refresh operations are the same as were described in "other than those in the DRC4 and RWC registers and timing other than for refresh operations are the same as were described in 3.4 Connection with Multiple EDO DRAMs (HM5164165FL-5).**

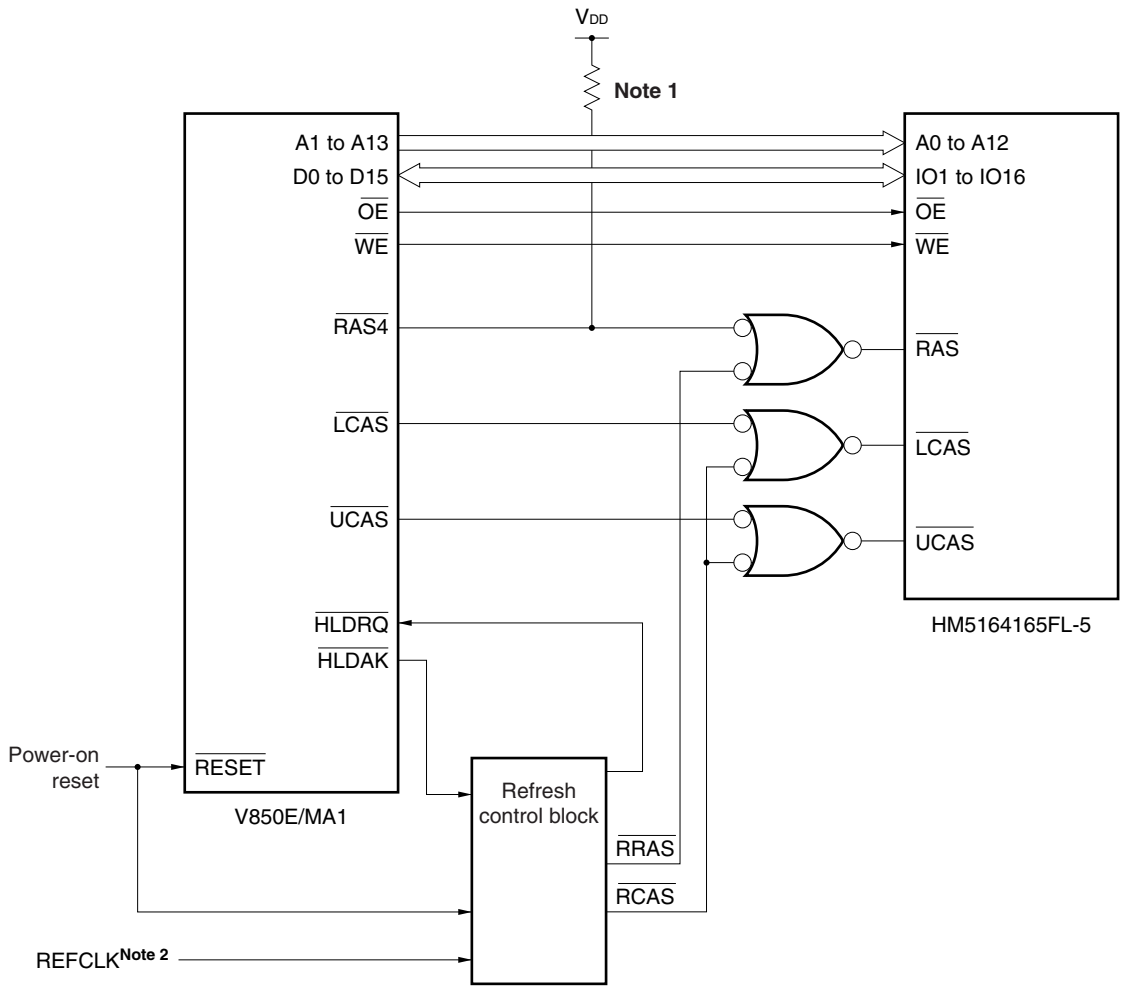
**[Register settings]**

**Table 3-6. Connection with External Refresh Unit Using $\overline{\text{HLDRQ}}$ Signal**

| Register Name | Setting value | Function |
|---|---|---|
| BCT1 | x00xx0xxx00x1010B | $\overline{\text{CS4}}$: Memory controller enabled, EDO DRAM |
| BSC | 0x0x0x010x0x0x0xB | $\overline{\text{CS4}}$: 16 bits |
| BCC | xxxxxx01xxxxxxxxB | $\overline{\text{CS4}}$: Idle state 1 |
| SCR4 | A645H | On-page access enabled, $w_{RP}$ = 2, $w_{RH}$ = 1, $w_{DA}$ = 2, $w_{CP}$ = 1, 16-bit width, multiplex width = 9 bits |
| RFS4 | 0xxxxxxxxxxxxxxxB | Refresh prohibited |
| RWC | xxxxx001B | Waits for canceling self refresh: 1 |

**Remark** $w_{RP}$: Number of row address precharge waits
$w_{RH}$: Number of row address hold waits
$w_{DA}$: Number of data waits
$w_{CP}$: Number of column address precharge waits

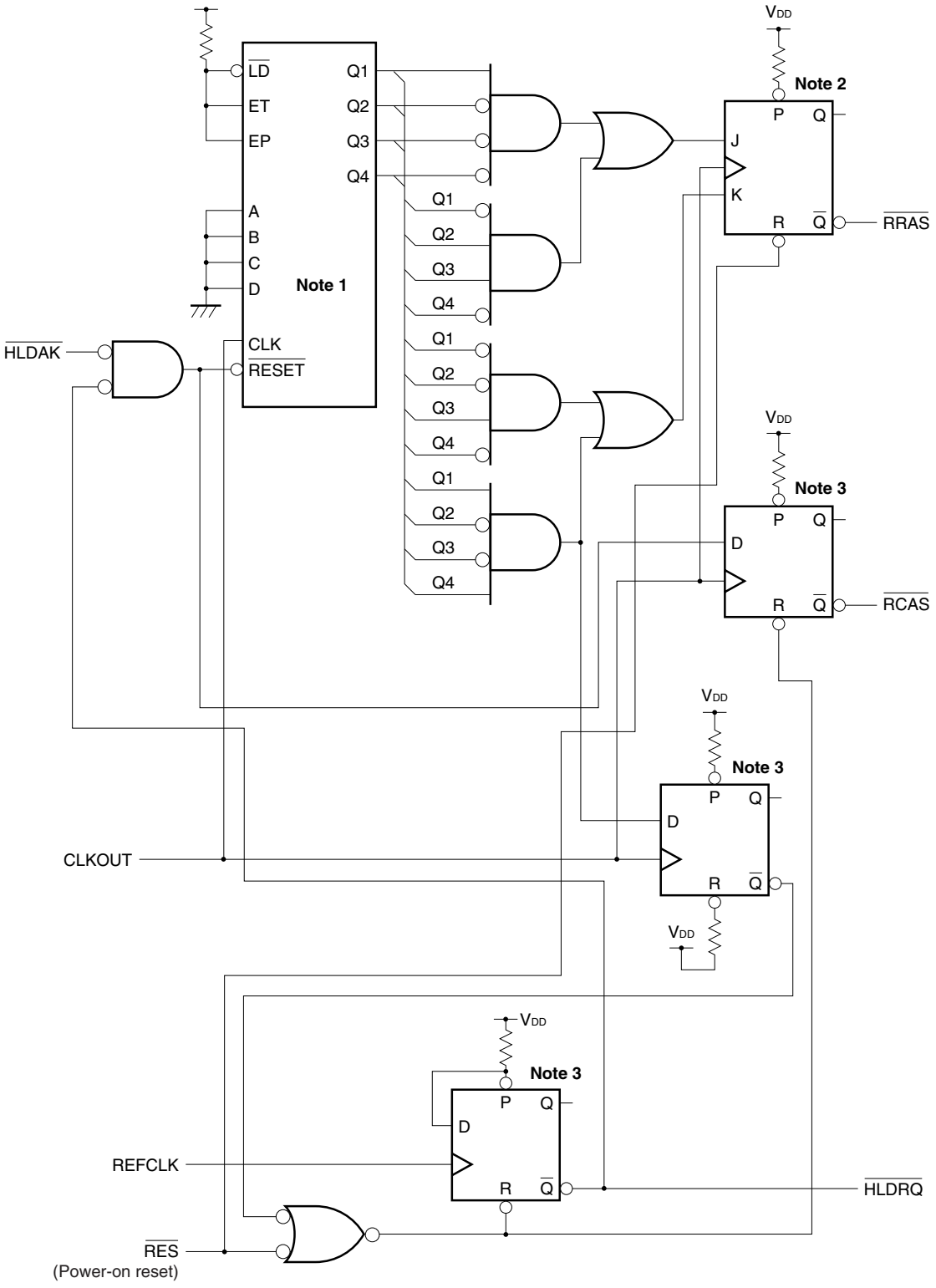
**Figure 3-16. Connection with External Refresh Unit Using $\overline{\text{HLDRQ}}$ Signal**




**Notes 1.** When using $\overline{\text{RAS}}$ hold mode, when the $\overline{\text{RASn}}$ signal enters hold mode at low level, spike noise may occur for the EDO DRAM's $\overline{\text{RAS}}$ signal. In this example, a pull-up resistor is connected to prevent such spike noise.

**2.** It is assumed that a 30 $\mu$s clock cycle is being input. In an actual circuit, use output signals (TO0n, etc.) from the V850E/MA1's internal RPU to generate the clock.

**Figure 3-17. Circuit Configuration of Refresh Control Block**

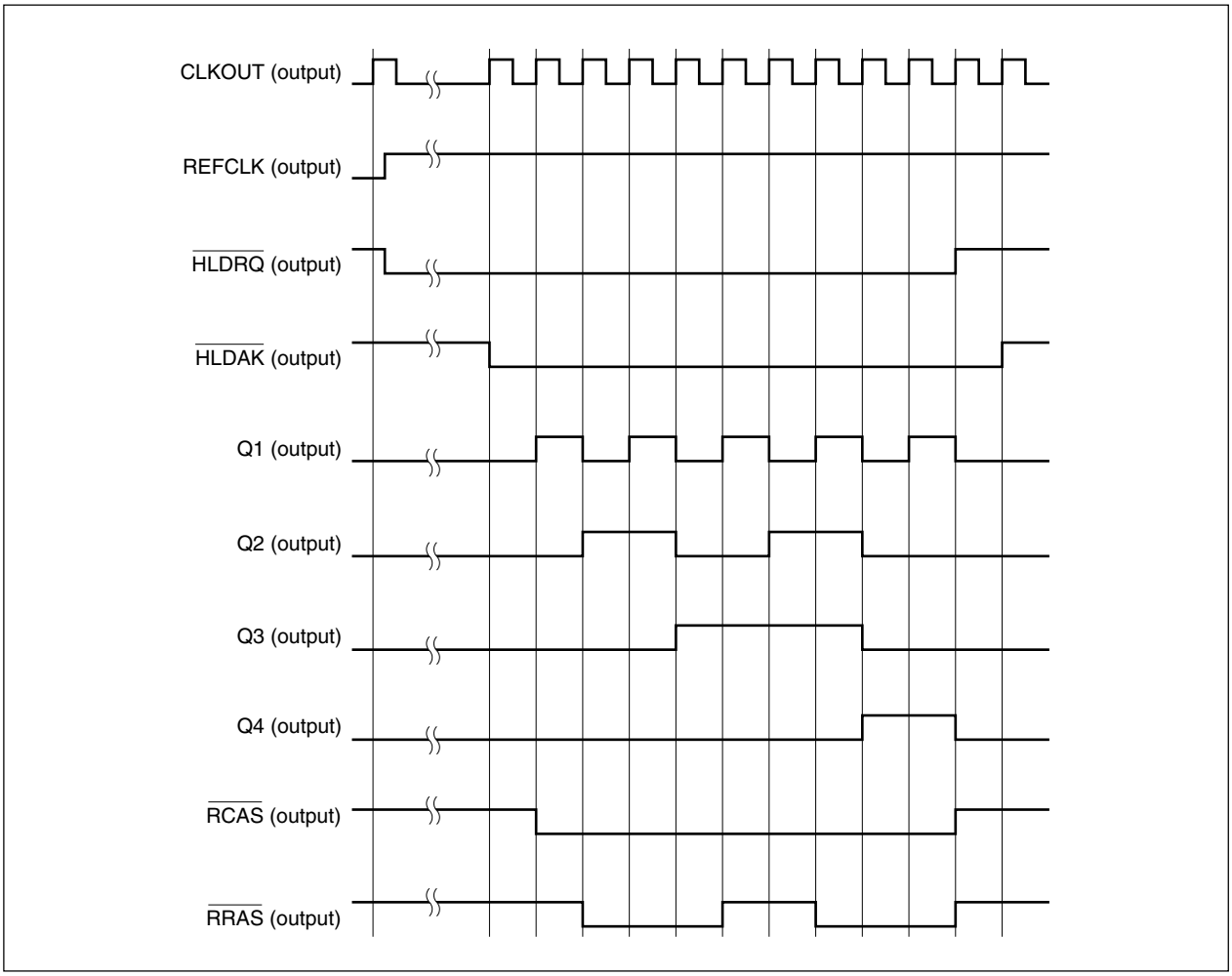



**Notes 1.** This is assumed to be a circuit equivalent to the 74161.

**2.** This is assumed to be a circuit equivalent to the 7476.

**3.** This is assumed to be a circuit equivalent to the 7474.

**Figure 3-18. Refresh Control Timing**



# CHAPTER 4 APPLICATION EXAMPLES

This chapter describes functions, circuits, and program examples related to the TU-V850E/MA1 CPU board that includes the V850E/MA1.

## 4.1 Functions of TU-V850E/MA1

### 4.1.1 Outline

The TU-V850E/MA1 is a training unit that has been developed as a tool for studying the V850E/MA1, which is a product in the V850 Series of 32-bit single-chip microcontrollers.

The TU-V850E/MA1's features are described below.

**(1) Includes an in-circuit emulator for the V850E/MA1**

**(2) Bus operations can be observed using a logic analyzer**

All signals related to the bus interface can be assigned to logic analyzer connectors.

**(3) Supports 50 MHz (Max.) operation**

Two types of oscillators (OSC1 and OSC2) are available and the supplied clock can be selected via jumper switch settings. OSC1 is supplied as the reference supplied clock, and OSC2's supplied clock can be changed since its is mounted on a socket.

**(4) Equipped with various types of memory**

- EPROM
- SRAM
- EDO DRAM
- Synchronous DRAM
- Flash memory
- Page ROM
- Serial EEPROM™ (when connected to CSI2 in the V850E/MA1)

**(5) Serial interface**

RS-232-C × 1 channel (Connects to UART1 in V850E/MA1)

**(6) Includes flash programmer interface for on-chip flash memory**

Connects to CSIO in V850E/MA1

**(7) Analog input**

Connects to output from sine-wave oscillator
Connects to analog output (0 V to 3 V)
Connects to output of optical sensor

**(8) General-purpose switch inputs (8) and LED outputs (8)**

Connects to V850E/MA1's port function

**(9) 7-segment LEDs (5 digits)**

A switch is used to select between software control and hardware control for the LED display. The display shows the frequency of the sine-wave oscillator or RPU (TO02 output) when hardware control has been selected.

**(10) Includes DC motor with rotary encoder output**

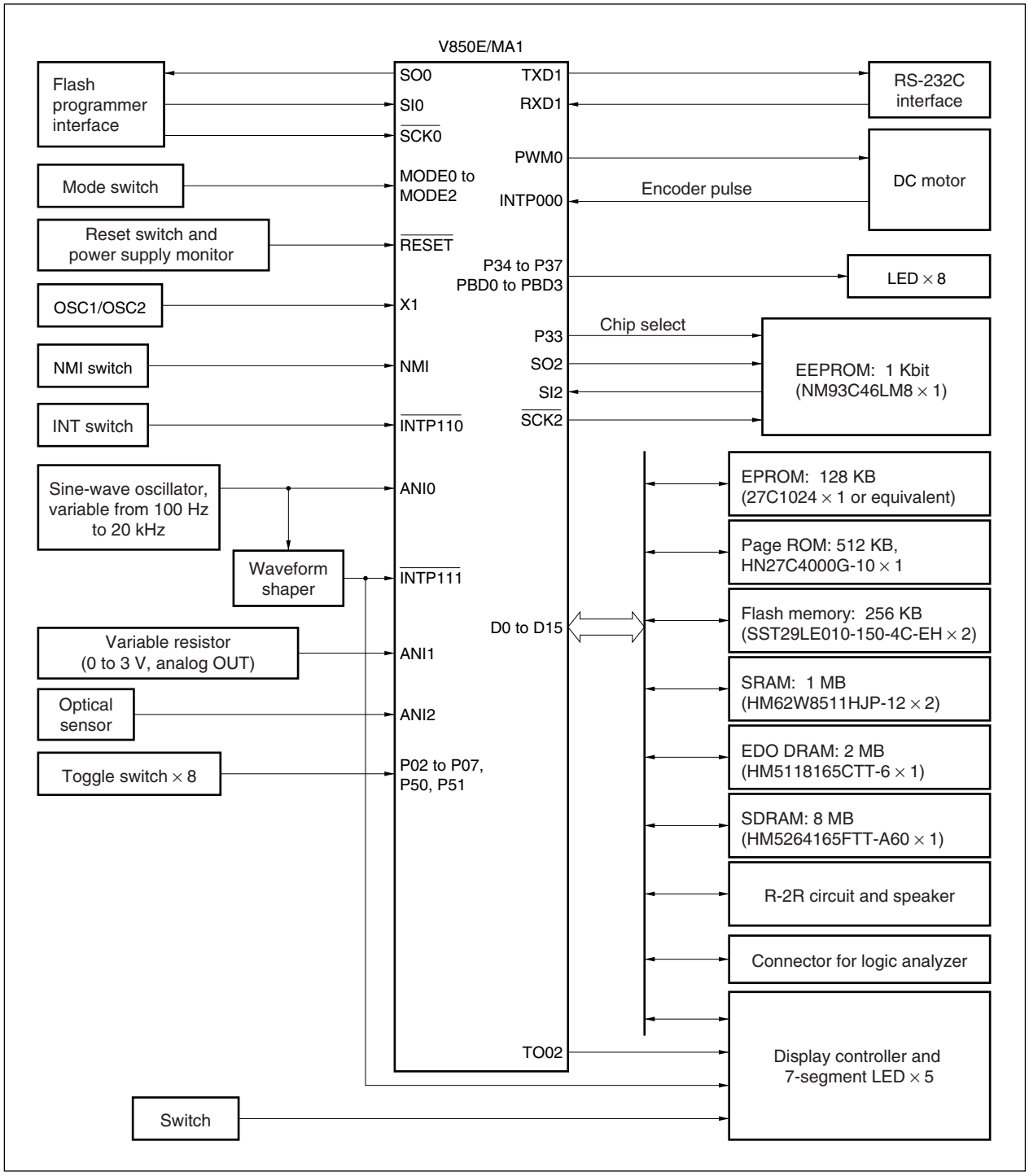
PWM0 in the V850E/MA1 is used to control the motor speed.

**(11) Includes a speaker**

Control is via an R-2R circuit connected to the bus interface.

**(12) Uses AC 100 V/200 V power supply**

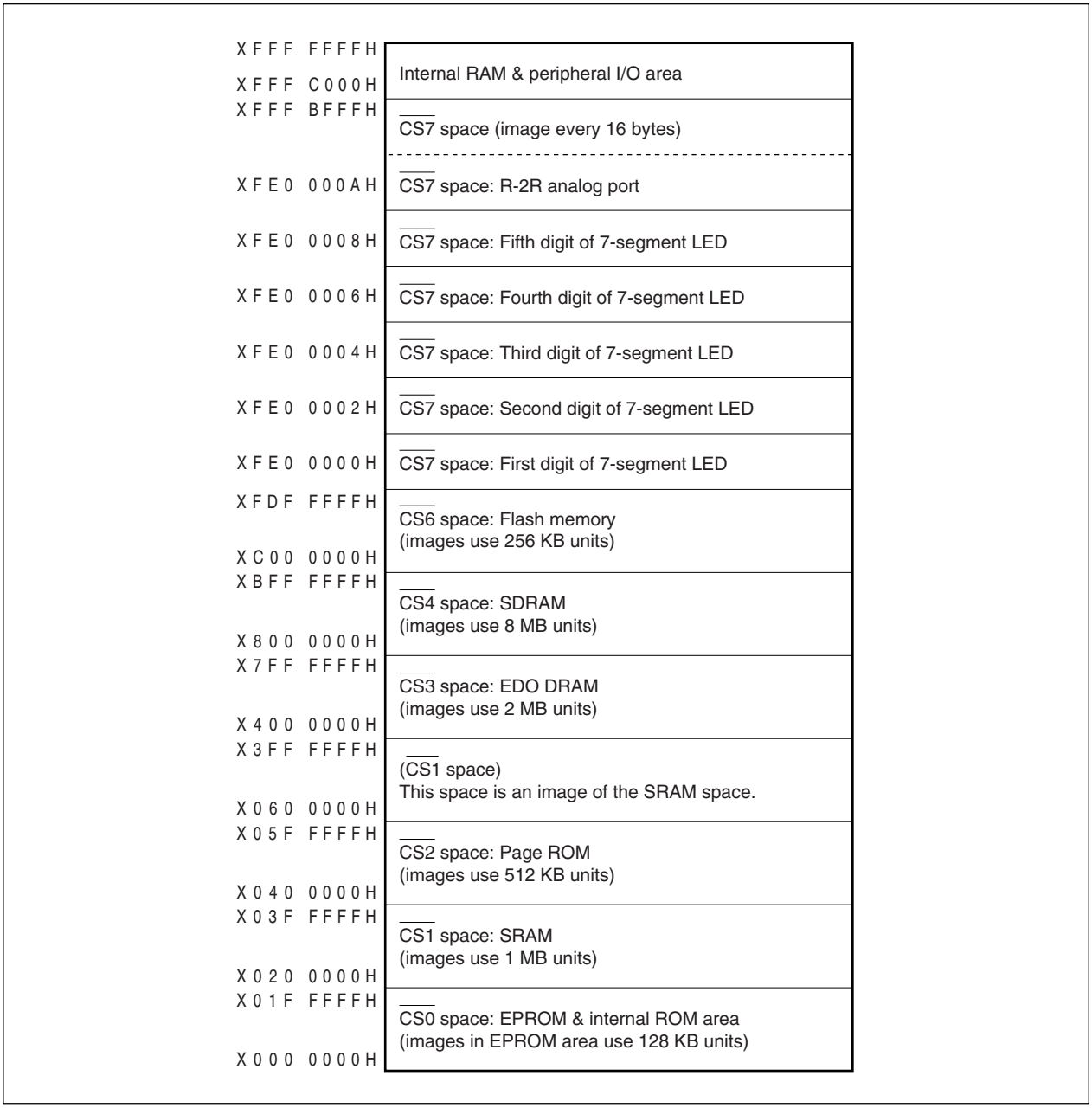
**Figure 4-1. Board Configuration Diagram**



### 4.1.2 Memory map

The following memory map is used when single-chip mode 0 has been selected via the mode select switch.

**Figure 4-2. Memory Map**



### 4.1.3 Peripheral function connection units

**(1) RPU connection**

Output from TO02 is connected to the external display control block. The frequency of the TO02 output is set via switch settings and is displayed on the 7-segment LED.

**(2) UART connection**

UART1 is used for start-stop synchronous RS-232C interfaces. The signals used two lines: TXD and RXD. The modem control signals (RS and ER signals) are fixed at active level and the CS and DR signals are not connected. The TXD signal can be looped back to the RXD signal via jumper switch settings.

| | |
|---|---|
| Communication method: | Asynchronous |
| Communication speed: | Up to 19,200 bps (set via on-chip baud rate generator) |
| Output signal: | TXD |
| Input signal: | RXD |
| Connector: | DSUB9 pin |

**(3) CSI connection**

CSI0 and CSI2 are respectively connected to the internal flash programmer interface connector and the EEPROM. When using a dedicated flash programmer, set the MODE switch to flash memory programming mode.

The EEPROM's DI pin, DO pin, and CLK pin are respectively connected to SO2, SI2, and $\overline{\text{SCK2}}$. In addition, the EEPROM's CS pin (device select pin) is connected to P33.

**(4) PWM connection**

PWM0 output is used to control the speed of the DC motor. When PWM0 pin is at high level, the DC motor operates at full speed and when it is at low level, the DC motor is stopped. The DC motor's rotary encoder pulse is connected to the INTP000 pin.

**(5) A/D converter connection**

The sine-wave oscillator's output (100 Hz to 20 kHz) is connected to ANI0, the 0 V to 3 V variable resistor is connected to ANI1, and the optical sensor is connected to ANI2.

**(6) Interrupt signal input**

The NMI switch, DC motor encoder pulse, INT switch, and the pulse signal converted to a wave form from the sine-wave oscillator's output are respectively connected to NMI, INTP000, $\overline{\text{INTP110}}$, and $\overline{\text{INTP111}}$.

**(7) I/O port connection**

Eight general-purpose toggle switches are connected to P02 to P07, P50, and P51 and eight general-purpose LEDs are connected to P34 to P37 and PBD0 to PBD3. Also, P33 is connected to the EEPROM's CS pin.

### 4.1.4 Bus interface connection unit

The V850E/MA1's bus interface connects with an EPROM, flash memory, SRAM, EDO DRAM, synchronous DRAM, page ROM, five 7-segment LEDs, and to a speaker via an R-2R analog output circuit.

Wait control for all units connected to the bus interface is performed by the V850E/MA1 (the $\overline{\text{WAIT}}$ pin is pulled up).

Among the various units connected via the bus interface, only the 7-segment LED and the R-2R analog output have an 8-bit bus width; all other units have a 16-bit bus width.

**(1) EPROM**

The V850E/MA1 includes one 40-pin socket that can be used to install a 1 Mbit EPROM (64 Kbits × 16 bits: 27C1024) or equivalent product in block 0 of the external memory space. It is selected when the $\overline{\text{CS0}}$ signal is active.

**(2) SRAM**

Two 4 Mbit SRAMs (512 Kbits × 8 bits: HM62W8511HJP-12) are mounted in block 1 of the external memory space. It is selected when the $\overline{\text{CS1}}$ signal is active.

**(3) Page ROM:**

A burst access-enabled 4 Mbit EPROM (256 Kbits × 16 bits: HN27C4000G-10) is mounted in a 40-pin IC socket in block 2 of the external memory space. It is selected when the $\overline{\text{CS2}}$ signal is active.

**(4) EDO DRAM:**

A 16 Mbit EDO DRAM (1 Mbit × 16 bits: HM5118165CTT-6) is mounted in area 1 of the external memory space. The $\overline{\text{RAS3}}$ pin is used for the $\overline{\text{RAS}}$ signal. Refresh cycles are supported by the V850E/MA1.

**(5) Synchronous DRAM**

A 64 Mbit synchronous DRAM (1 Mbit × 16 bits × 4 banks: HM5264165FTT-A60) is mounted in area 2 (the $\overline{\text{CS2}}$ space) in the external memory space. Refresh cycles are supported by the V850E/MA1.

**(6) Flash memory**

Two 1 Mbit flash memories (128 Kbits × 8 bits: SST29LE010-150-4C-EH) are mounted in block 0 of the external memory space. They are selected when the $\overline{\text{CS0}}$ signal is active.

**(7) EEPROM**

A serial type 1 Kbit EEPROM (NM93C46LM8) is mounted. It is controlled by the V850E/MA1's $\overline{\text{CSI2}}$. The NM93C46LM8's CS pin is connected to the V850E/MA1's P33 pin.

### 4.1.5 Setting switches

**(1) Operation mode specification switch (DSW1)**

These are 4-bit DIP switches that specify the V850E/MA1's operation mode. They are connected to the V850E/MA1's MODE0 and MODE1 pins. These switch settings are "0" for ON and "1" for OFF. The MODE2 pin is set via the flash programmer. In normal operation, the MODE2 pin is fixed at GND level.

| Switch Name | Meaning |
|---|---|
| DSW1-1 | Sets MODE0 pin |
| DSW1-2 | Sets MODE1 pin |
| DSW1-3, DSW1-4 | Not used |

**(2) Slide switch for flash programmer (SSW1)**

This slide switch is used when using the dedicated flash programmer to perform read or write operations involving the contents of the V850E/MA1's on-chip flash memory. When set to the write side, the V850E/MA1's $\overline{\text{RESET}}$ pin is controlled by the dedicated flash programmer.

| Switch Name | Meaning |
|---|---|
| NORMAL | Normal operation |
| WRITE | On-chip flash memory: read/write |

**(3) Supplied clock setting switch (JP1)**

This jumper switch specifies the clock that is supplied to the V850E/MA1's X1 pin.

| Switch Name | Meaning |
|---|---|
| 1-2 short | Selects OSC1 |
| 2-3 short | Selects OSC2 |

**(4) RS-232C interface loopback switch (JP2)**

This switch connects the send data and receive data in the RS-232C interface.

| Switch Name | Meaning |
|---|---|
| 1-2 short | Normal operation |
| 2-3 short | Transmit data is looped back to receive data |

**(5) 7-segment LED switch (SW6)**

This is an alternate toggle switch that selects between software control or hardware control of the 7-segment LED display.

| Switch Name | Meaning |
|---|---|
| SOFT | Software control |
| HARD | Hardware control |

**(6) Frequency display select switch (SW7)**

This is an alternate toggle switch that select between two monitor modes for the 7-segment LED frequency display: monitoring of the sine-wave oscillator or monitoring of the TO02 output. This switch is valid only when the 7-segment LED switch is set for hardware control.

| Switch Name | Meaning |
|---|---|
| TO02 | Selects TO02 output |
| $\overline{\text{INTP111}}$ | Selects sine-wave oscillator output |

**(7) DC motor start switch**

This is an alternate toggle switch that sets DC motor control as valid. The motor is stopped when this switch is set to OFF.

| Switch Name | Meaning |
|---|---|
| ON | Enables startup of DC motor |
| OFF | Forcibly stops operation of DC motor |

**(8) General-purpose toggle switches (TSW1 to TSW8)**

This is a set of eight alternate toggle switches that are connected to the V850E/MA1's input ports. P02 to P07 correspond to TSW3 to TSW8 and P50 and P51 correspond to TSW1 and TSW2. "0" is read when each switch is ON and "1" is read when each is OFF.

**(9) INT switch (INT)**

This is a push-button switch that generates a maskable interrupt to the V850E/MA1. It is connected to the $\overline{\text{INTP110}}$ pin.

**(10) NMI switch (NMI)**

This is a push-button switch that generates a non-maskable interrupt to the V850E/MA1.

**(11) Reset switch (RESET)**

This is a push-button switch that resets this unit. This switch takes the ORed result of the power-on reset signal.

The reset switch is ignored when the slide switch for the flash programmer has been set to the WRITE side.

### 4.1.6 Peripheral I/O specifications

#### (1) Port functions

Eight input ports and nine output ports are used. They are listed below.

| Port Name | I/O | Function |
|---|---|---|
| P50 | Input | General-purpose toggle switch 1 (ON = 0, OFF = 1) |
| P51 | Input | General-purpose toggle switch 2 (ON = 0, OFF = 1) |
| P02 | Input | General-purpose toggle switch 3 (ON = 0, OFF = 1) |
| P03 | Input | General-purpose toggle switch 4 (ON = 0, OFF = 1) |
| P04 | Input | General-purpose toggle switch 5 (ON = 0, OFF = 1) |
| P05 | Input | General-purpose toggle switch 6 (ON = 0, OFF = 1) |
| P06 | Input | General-purpose toggle switch 7 (ON = 0, OFF = 1) |
| P07 | Input | General-purpose toggle switch 8 (ON = 0, OFF = 1) |
| PBD0 | Output | General-purpose LED1 (ON = 1, OFF = 0) |
| PBD1 | Output | General-purpose LED2 (ON = 1, OFF = 0) |
| PBD2 | Output | General-purpose LED3 (ON = 1, OFF = 0) |
| PBD3 | Output | General-purpose LED4 (ON = 1, OFF = 0) |
| P34 | Output | General-purpose LED5 (ON = 1, OFF = 0) |
| P35 | Output | General-purpose LED6 (ON = 1, OFF = 0) |
| P36 | Output | General-purpose LED7 (ON = 1, OFF = 0) |
| P37 | Output | General-purpose LED8 (ON = 1, OFF = 0) |
| P33 | Output | Serial EEPROM chip select (select = 1) |

#### (2) External interrupt pins

One NMI pin and three maskable interrupt pins are connected. They are listed below.

| Interrupt Name | Edge Specification | Connected Signal |
|---|---|---|
| NMI | Falling edge | NMI switch |
| INTP000 | Rising edge | DC motor encoder pulse (216 pulses per rotation) |
| $\overline{\text{INTP110}}$ | Falling edge | INT switch |
| $\overline{\text{INTP111}}$ | Rising edge | Sine-wave oscillator (1 pulse per cycle) |

#### (3) Serial interface

CSI0, CSI2, and UART1 are used. They are listed below.

| Unit Name | Function |
|---|---|
| CSI0 | This is used as the V850E/MA1's on-chip flash memory interface. It is optimally configured for use with the dedicated flash programmer. |
| CSI2 | This is used as the serial EEPROM's interface. EEPROM access is valid only when the value of the $\overline{\text{CS}}$ pin (P33) is "1". |
| UART1 | This is used as the RS-232C interface. Modem control signals are not supported (RS and ER are always active and CS, DR, and CD are left open). Program these as required for the target protocol. |

**(4) PWM unit**

PWM0 is used to control the DC motor. The motor operates when PWM0 is at high level and is stopped when PWM0 is at low level. When at 100% output (when PWM1 output is held at high level), the motor speed is 250 rpm (rotations per minute). The rotation speed (rpm) can be monitored via INTP000.

**(5) A/D converter**

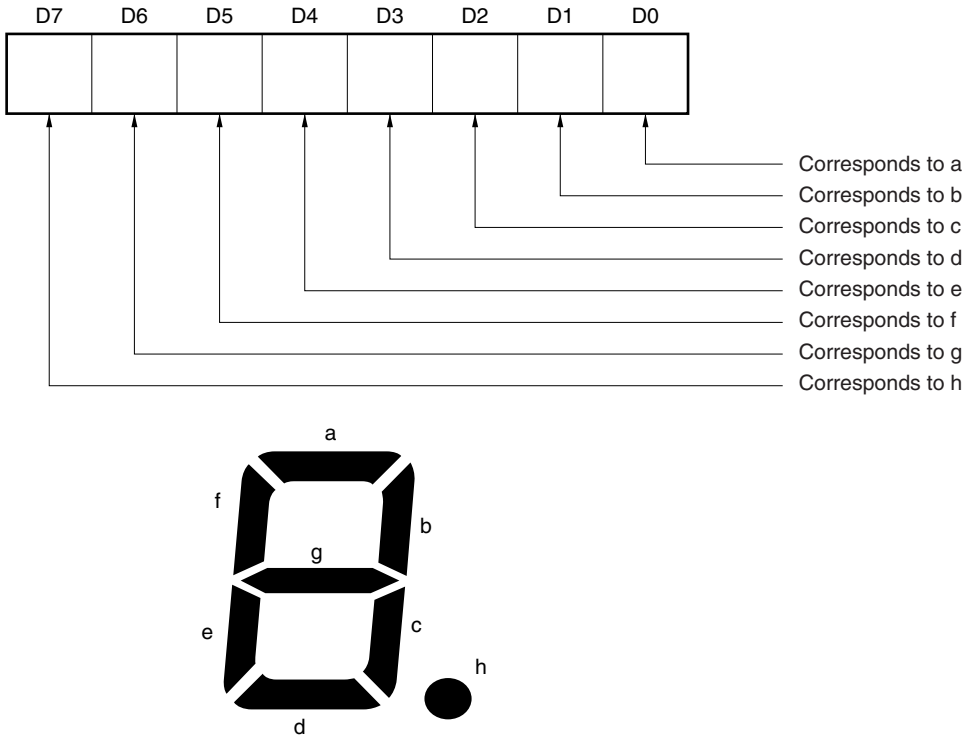
A 0 V to 3 V analog input is connected via three channels (ANI0 to ANI2 pins). These are listed below.

| Pin Name | Function |
|---|---|
| ANI0 | Sine-wave input (100 Hz to 20 kHz) |
| ANI1 | 0 V to 3 V input from variable resistor |
| ANI2 | 0 V to 3 V (0 V = dark, 3 V = bright) input from optical sensor |

**(6) 7-segment LED display register**

This is a read/write register that is used to light the LED display. When the 7-segment LED switch is set to the “software control” side, the settings in this 7-segment LED display register are reflected on the LED. This register is accessible even when the 7-segment LED switch is not set for “software control”.

**Figure 4-3. 7-Segment LED Display Register**




**Remark** Each LED is ON (lit) when the corresponding bit value is “1” and is OFF (not lit) when “0”.

**(7) R-2R analog output register**

This is an analog output port that is read/write accessible in 8-bit units. FFH sets the maximum voltage and 00H sets the minimum voltage. The settings in the R-2R analog output register are reflected in the output from the speaker.

**Table 4-1. List of I/O Registers**

| Address in Little Endian Order | Address in Big Endian Order | Function |
|---|---|---|
| XXXX XXX0H | XXXX XXX3H | Register for first digit of 7-segment LED |
| XXXX XXX2H | XXXX XXX1H | Register for second digit of 7-segment LED |
| XXXX XXX4H | XXXX XXX7H | Register for third digit of 7-segment LED |
| XXXX XXX6H | XXXX XXX5H | Register for fourth digit of 7-segment LED |
| XXXX XXX8H | XXXX XXXBH | Register for fifth digit of 7-segment LED |
| XXXX XXXAH | XXXX XXX9H | R-2R analog output register |
| XXXX XXXBH to XXXX XXXFH | XXXX XXXCH to XXXX XXXFH | Not used |

**Remark** These are addresses within the $\overline{CS7}$ space.

### 4.1.7 Connectors

**(1) RS-232C interface connector (CN1)**

**Connector type: DSUB9 pin (DE-9S-T-N: JAE)**

| Pin No. | Signal Name | Meaning of Signal |
|---|---|---|
| 1 | NC | Not connected |
| 2 | RD | Receive data |
| 3 | SD | Transmit data |
| 4 | ER | Terminal ready |
| 5 | GND | Signal ground |
| 6 | NC | Not connected |
| 7 | RS | Transmit request |
| 8 | NC | Not connected |
| 9 | NC | Not connected |

**Remark** RS and ER are fixed at active level.

### (2) Flash memory programming interface connector (CN2)

**Connector type: 10-pin header (PS-10PE-D4T1-B1: JAE)**

| Pin No. | Signal Name | Meaning of Signal |
|---|---|---|
| 1 | GND | Ground |
| 2 | SO0 | Serial data output |
| 3 | SI0 | Serial data input |
| 4 | $\overline{\text{SCK0}}$ | Serial clock input |
| 5 | NC | Not connected |
| 6 | $\overline{\text{RESET}}$ | Reset signal input |
| 7 | $V_{DD}$ | Board power supply |
| 8 | $V_{PP}$ | $V_{PP}$ voltage input |
| 9 | NC | Not connected |
| 10 | NC | Not connected |

### (3) Logic analyzer connectors (CN3 to CN6)

**CN3**

| Pin No. | Signal Name | Pin No. | Signal Name |
|---|---|---|---|
| 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**

| Pin No. | Signal Name | Pin No. | Signal Name |
|---|---|---|---|
| 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**

| Pin No. | Signal Name | Pin No. | Signal Name |
|---|---|---|---|
| 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**

| Pin No. | Signal Name | Pin No. | Signal Name |
|---|---|---|---|
| 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 Examples of Application Programs

### 4.2.1 Development tools

The program examples shown here were all created using NEC's development environment.

**(1) C compiler package (CA850)**

The following four user's manuals are related to the C compiler package.

- CA850 C Compiler Package Operation (U15024E)
- CA850 C Compiler Package C Language (U15025E)
- CA850 C Compiler Package Assembly Language (U15027E)
- CA850 C Compiler Package Project Manager (Windows Based) (U15026E)

**Remark** #pragma extension codes in C language and on-chip peripheral I/O register name abbreviations or quasi directives in assembly language are inherent to the C compiler package (CA850).

**(2) C source debugger (ID850) for V850 Series**

The following user's manual is related to the C source debugger for V850 Series.

- ID850 Integrated Debugger Operation (Windows Based) (U15181E)

**(3) IE-V850E-MC-A**

The following user's manual is related to the V850E1 in-circuit emulator (3.3 V).

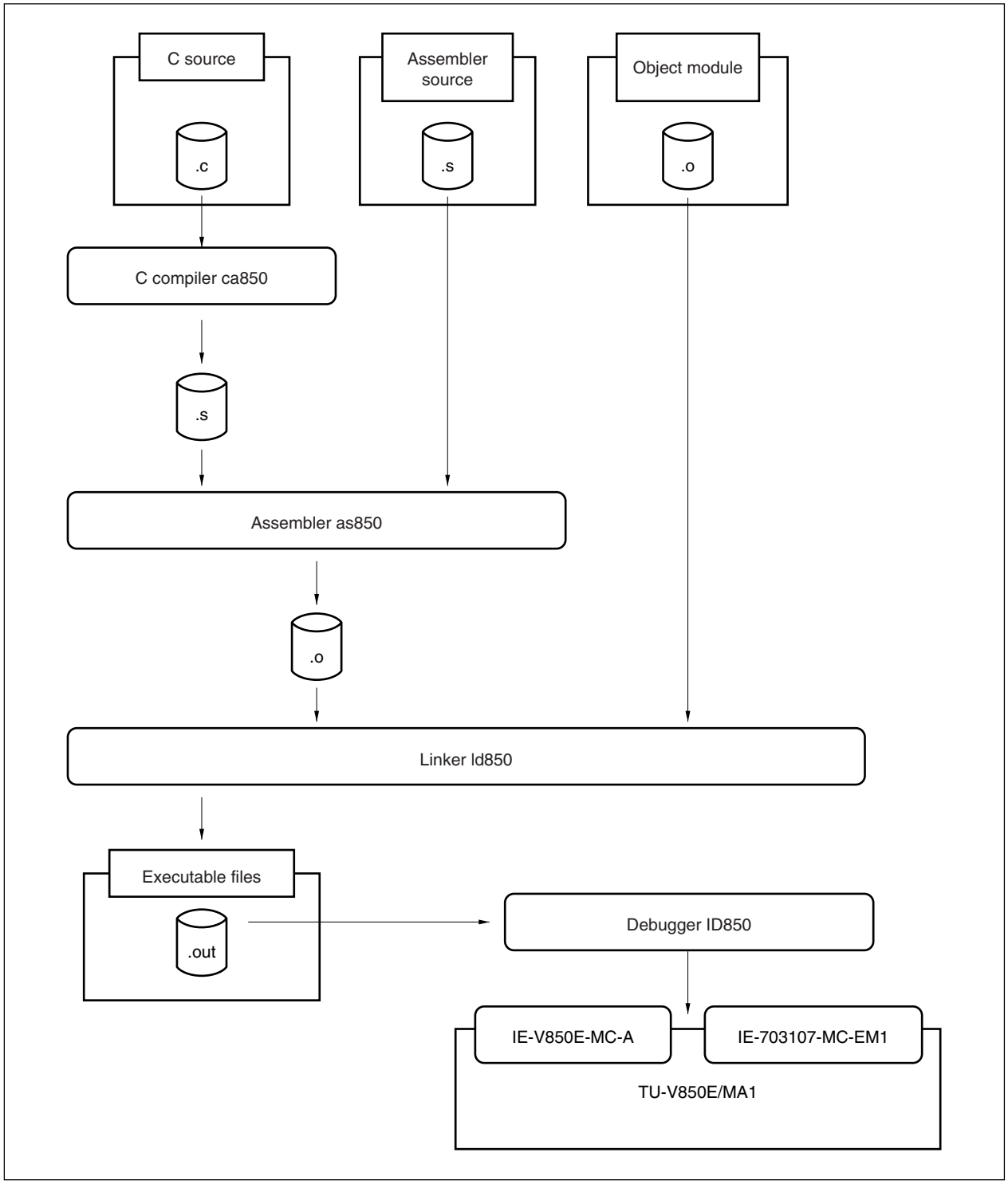
- IE-V850E-MC, IE-V850E-MC-A In-Circuit Emulator (U14487E)

**(4) IE-703107-MC-EM1**

The following User's Manual is related to the in-circuit emulator option board for the V850E/MA1.

- IE-703107-MC-EM1 In-Circuit Emulator Option Board (U14481E)

**Figure 4-4. Configuration of Development Tools**




The following manual is recommended to anyone who is using NEC development tools for the first time.
**V800 Series™ Development Tools (32-bit supported) Tutorial Guide Windows Based (U14218E)**

### 4.2.2 Program configuration

The configuration of application programs is described below. The range of resources available to the CPU increases in the order given (<1> → <2> → <3>).

**<1> 4.2.3 Operation example in single-chip mode 0 (STEP1)**

The V850E/MA1 operates in single-chip mode 0. Only CPU operations can be recognized.

**<2> 4.2.4 Operation example in single-chip mode 0 + extended mode (STEP2)**

The V850E/MA1 operates in single-chip mode 0 and in externally extended mode. CPU operations and operations connected via an external bus (such as memory) can be recognized.

**<3> 4.2.5 Operation example using various peripheral functions (STEP3)**

Operations such as on-chip peripheral I/O operations and external device operations that are based on <1> and <2> above are recognized. For <3>, the application programs operate once the following two initialization operations are completed.

- Initialization files that appear in application program examples
  [cpuinit.s]: Initializes basic functions of CPU
  [businit.s]: Initializes CPU bus control function when using an external memory space

### 4.2.3 Operation example in single-chip mode 0 (STEP1)

This initializes the CPU's basic functions. The V850E/MA1 operates without using any external space.

**(1) Points concerning initialization of CPU's basic functions**

Be sure to set following registers before initializing any peripheral I/O register in the V850E/MA1.

- System wait control register (VSWC)
- Clock control register (CKC)

Afterward, set other registers as needed.

★ **Figure 4-5. Settings in System Wait Control Register (VSWC)**

Address: FFFFF06EH

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Initial value | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| Bit name | – | – | – | – | – | – | – | – |
| Setting value | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |

| Operating frequency ($f_{XX}$) | Setting value in VSWC |
|---|---|
| 4 MHz ≤ $f_{XX}$ < 33 MHz | 11H |
| 33 MHz ≤ $f_{XX}$ ≤ 50 MHz | 12H (recommended) or 13H |
| Initial value | 77H |

In this case, “12H” is set in the VSWC register since operation is at 40 MHz.

**(2) Points concerning initialization of CPU clock**

Be sure to use the specified sequence when setting data to the clock control register.

**Figure 4-6. Settings in Clock Control Register (CKC)**

Address: FFFFF822H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit name | 0 | 0 | TBCS | CESEL | 0 | CKDIV2 | CKDIV1 | CKDIV0 |
| Setting value | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |

| Bit position | Bit name | Description |
|---|---|---|
| 5 | TBCS | Selects time-based counter clock<br>0: $f_X/2^8$ |
| 4 | CESEL | Selects function of X1 and X2 pins<br>1: External clock is connected to X1 pin |
| 2 to 0 | CKDIV2 to CKDIV0 | Sets internal system clock frequency ($f_{XX}$)<br>111: 10 × $f_X$ |

**[Program example]**

```
#### Initialization of CPU's basic functions ####
_cpu_initial:
     mov    0x12,  r6          -- Selects operating frequency (fxx)as 33 MHz ≤ fxx ≤ 50
                                  MHz
     st.b   r6,    VSWC[r0]    -- Sets system wait control register

#### Access via specified sequence ####
     mov    0x00a0,r6          -- <1> NMI interrupt prohibited
     ldsr   r6,    5           --     Sets "1" to PSW's NP bit
     mov    0x17,  r6          -- <2> Prepares data to be written to general-purpose
                                      register CKC
                               --     External clock connection to X1 pin, fxx = 10 x fx
     st.b   r6,    PHCMD[r0]   -- <3> Writes dummy data to peripheral command register
     st.b   r6,    CKC[r0]     -- <4> Sets clock control register
     nop                       -- <5> Inserts NOP instruction (5 instructions are
                                      issued)
     nop                       -- <6> Inserts NOP instruction
     nop                       -- <7> Inserts NOP instruction
     nop                       -- <8> Inserts NOP instruction
     nop                       -- <9> Inserts NOP instruction
     mov    0x0020,r6          -- <10> Clears NMI interrupt prohibition
     ldsr   r6,    5           --      Returns "0" to PSW's NP bit

# Writes fixed data to internal RAM area (address xfffc000)
# <Registers used>
#    r10:  Internal RAM address
#    r11:  Setting data (55H)

#External reference declaration
.extern  _cpu_init

# Reset and interrupt
.section "RESET",text          -- Reset handler declaration
     jr     _start             -- Jumps to start of sample program code block

#Constant definitions
.set internal_ram, 0xfffc000   -- Internal RAM start address
.set write_data,   0x55        -- Write data

#Main program
.text
.align 4                       -- 4-byte alignment
.globl _start                  -- External declaration
_start:
     jarl   _cpu_init,    r31 -- Initialization of CPU's basic functions
     mov    internal_ram, r10 -- Sets write address
     mov    write_data,   r11 -- Sets write data
     st.b   r11,   0x0[r10]   -- Writes bytewise
_forever:
     br     _forever           -- Endless loop
```

### 4.2.4 Operation example in single-chip mode 0 + extended mode (STEP2)

In addition to the initialization of the CPU's basic functions, as described in **4.2.3 Operation example in single-chip mode 0 (STEP1)** above, CPU bus control functions are initialized to enable access to an external space.

**(1) Points concerning pin function initialization**

Select single-chip mode 0, and be sure to make settings in the following procedure when switching the pin for output or I/O operations during control mode from port mode to control mode (n = 0 to 5, AL, AH, DL, CS, CT, CM, CD, or BD).

<1> Set the inactive level of the signal output during control mode to the pin that corresponds to Port n (Pn).
<2> Use the Port n mode control register (PMCn) to switch the control mode.

**Remark** Unless <1> above is performed, the contents of Port n will be immediately output when switching from port mode to control mode.

The initial values of pins related to memory access (determined by the state of the MODE0 to MODE2 pins) varies according to the operating mode.

| | Modes Other Than Single-Chip Mode 0 | Single-Chip Mode 0<br>(Internal ROM area is assigned to the addresses from 00000000H) |
|---|---|---|
| PMCAL | Lower 16 bits of address bus | Port |
| PMCAH | Higher 10 bits of address bus | Port |
| PMCDL | Data bus's 16 bits | Port |
| PMCCS | Chip select signal | Port |
| PMCCT | Bus cycle-related signals | Port |
| PMCCM | DRAM-related signals | Port |
| PMCCD | SDRAM-related signals | Port |

**Remark** When in single-chip mode 0, these pins operate as port pins rather than bus control function pins.

| | Modes Other Than ROMless Mode 1 | ROMless Mode 1 (8-Bit Data Bus Space) |
|---|---|---|
| BSC | 16-bit data bus width | 8-bit data bus width |

| | Modes Other Than Flash Memory Programming Mode | Flash Memory Programming Mode |
|---|---|---|
| PMC4 | Port | CSI0 pins ($\overline{SCK0}$, SI0, SO0) |

**[Program example]**

```
[file:bus_init.s]

#########################################################
#   Specification of bus control pins
#########################################################
     st.h   r0,    PAL[r0]     -- Sets inactive level (Lo) of A0 to A15 output signals
     mov    0xffff,r6          -- Selects A0 to A15 pins
     st.h   r6,    PMCAL[r0]

     st.h   r0,    PAL[r0]     -- Sets inactive level (Lo) of A16 to A25 output signals
     mov    0x03ff,r6          -- Selects A16 to A25 pins
     st.h   r6,    PMCAH[r0]

     st.h   r0,    PDL[r0]     -- Sets inactive level (Lo) of D0 to D15 output signals
     mov    0xffff,r6          -- Selects D0 to D15 pins
     st.h   r6,    PMCDL[r0]

     mov    0xff,  r6          -- Sets inactive level (Hi) of CS0 to CS7 output signals
     st.b   r6,    PCS[r0]
     mov    0xff,  r6          -- Selects CS0 to CS7 pins
     st.b   r6,    PMCCS[r0]

     mov    0xf3,  r6          -- Sets inactive level (Hi) of BCYST, OE, WE, RD,
                                  UCAS/UWR/UDQM,
                                  and LCAS/LWR/LDQM output signals
     st.b   r6,    PCT[r0]     -- Sets inactive level (Hi)
     mov    0xf3,  r6          -- Selects BCYST, OE, WE, RD, UCAS/UWR/UDQM, and
                                  LCAS/LWR/LDQM pins
     st.b   r6,    PMCCT[r0]

     mov    0x14,  r6          -- Sets inactive level (Hi) of REFREQ and HLDAK output
                                  signals
     st.b   r6,    PCM[r0]     -- Sets inactive level (Hi) of CLKOUT output signal
     mov    0x32,  r6          -- Selects SELFREF, REFRQ, and CLKOUT pins
     st.b   r6,    PMCCM[r0]
     mov    0x0c,  r6          -- Sets inactive level (Hi) of UBE/SDRAS and LBE/SDCAS
                                  output signals
     st.b   r6,    PCD[r0]     -- Sets inactive level (Lo) of SDCKE and SDCLK output
                                  signals
     mov    0x02,  r6          -- Caution:  Select the SDCLK pin first.
     mov    0x0d,  r7          -- Selects UBE/SDRAS, LBE/SDCAS, and SDCKE pins
     st.b   r6,    PMCCD[r0]
     ld.b   PMCCD[r0],r6
     or     r7,    r6
     st.b   r6,    PMCCD[r0]   -- Selects UBE/SDRAS, LBE/SDCAS, SDCLK, and SDCKE pins

     mov    0x0c,  r6          -- Selects SDRAS and SDCAS pins
     st.b   r6,    PFCCD[r0]
```

```
#####################################################################
#   Specification of functions other than bus control pin functions
#####################################################################
     st.b   r0,    P0[r0]      -- Sets inactive level (Lo) of PWM0 output signal
     mov    0x03,  r6          -- Selects PWM0 and INTP000 pins
     st.b   r6,    PMC0[r0]

     st.b   r0,    P2[r0]      -- Sets inactive level (Lo) of TO02 output signal
     mov    0x39,  r6          -- Selects NMI, TO02, INTP110, and INTP111 pins
     st.b   r6,    PMC2[r0]

     mov    0x04,  r6          --Sets inactive level (Lo) (not lit) of P37 to P34 (LED)
                               --Sets inactive level (Lo) of P33 (CS to EPROM)
                               --Sets inactive level (Hi) of SCK2 I/O signal
                               --Sets inactive level (Lo) of SO2 output signal
     st.b   r6,    P3[r0]
     mov    0x07,  r6
     st.b   r6,    PM3[r0]     -- P37 to P33 (LED), P33 (CS to EPROM) output mode
     st.b   r6,    PMC3[r0]    -- Selects SO2, SI2, and SCK2 pins

     st.b   r0,    PBD[r0]     -- Sets inactive level (Lo) (not lit) of PBD3 to PBD0
                                  (LED)
     mov    0xf0,  r6          -- PBD3 to PBD0 output mode (LED)
     st.b   r6,    PMBD[r0]


     mov    0x08,  r6          -- TXD1 (PORT43) output signal  High level
     st.b   r6,    P4[r0]
     mov    0x18,  r6          -- Selects RXD1/SI1 and TXD1/SO1 pins
     st.b   r6,    PMC4[r0]
     mov    0x18,  r6          -- Selects RXD1 input and TXD1 output modes
     st.b   r6,    PFC4[r0]
```

**(2) Points concerning initialization of memory block space distribution**

A 256 MB memory space is divided into four 64 MB areas (area 0 to area 3).

Area 0: 64 MB (00000000H to 03FFFFFFH), $\overline{CS1}$ signal (mainly)

Area 1: 64 MB (04000000H to 07FFFFFFH), $\overline{CS3}$ signal (fixed)

Area 2: 64 MB (08000000H to 0BFFFFFFH), $\overline{CS4}$ signal (fixed)

Area 3: 56 MB (0C000000H to 0FFFFFFFH), $\overline{CS6}$ signal (mainly)

Moreover, above 64 MB memory spaces can be divided into memory blocks in 2 MB units and chip select signals can be specified in detail.

Area 0: Lower 8 MB (00000000H to 007FFFFFH), $\overline{CS0}$, $\overline{CS1}$, and $\overline{CS2}$ signals can be selected

Area 1: Higher 8 MB (0F800000H to 0FFFFFFFH), $\overline{CS7}$, $\overline{CS6}$, and $\overline{CS5}$ signals can be selected

**Figure 4-7. Settings in Chip Area Select Control Register 0 (CSC0)**

Address: FFFFF060H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| Bit name | CS33 | CS32 | CS31 | CS30 | CS23 | CS22 | CS21 | CS20 | CS13 | CS12 | CS11 | CS10 | CS03 | CS02 | CS01 | CS00 |
| Setting value | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

These settings are not significant. These settings are not significant.

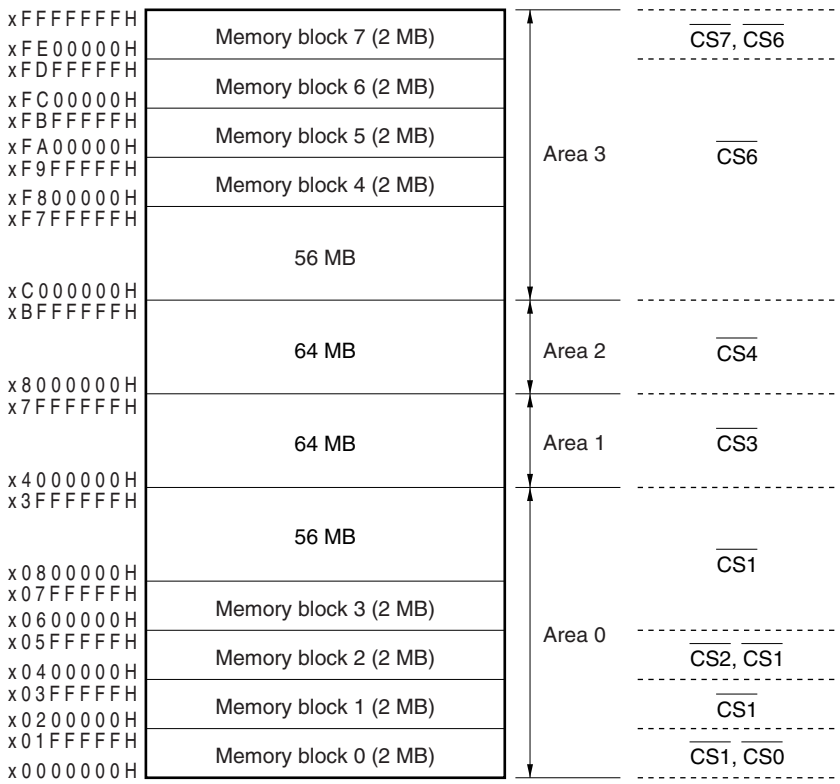
**Figure 4-8. Settings in Chip Area Select Control Register 1 (CSC1)**

Address: FFFFF062H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| Bit name | CS43 | CS42 | CS41 | CS40 | CS53 | CS52 | CS51 | CS50 | CS63 | CS62 | CS61 | CS60 | CS73 | CS72 | CS71 | CS70 |
| Setting value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

These settings are not significant. These settings are not significant.

**Figure 4-9. $\overline{CSn}$ Signal Areas**




**Remarks 1.** $\overline{CS5}$ is not used in this program example.

**2.** Priority in area 0: $\overline{CS0}$ > $\overline{CS2}$ > $\overline{CS1}$
Priority in area 3: $\overline{CS7}$ > $\overline{CS6}$

**3.** n = 0 to 7

**(3) Points concerning initialization of memory connection devices**

Be sure to reset before writing to the BCT0 and BCT1 registers.

Do not change register values after writing.

**Figure 4-10. Settings in Bus Cycle Type Configuration Registers 0 and 1 (BCT0 and BCT1)**

BCT0 Address: FFFFF480H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Bit name | ME3 | 0 | BT31 | BT30 | ME2 | 0 | 0 | BT20 | ME1 | 0 | BT11 | BT10 | ME0 | 0 | 0 | BT00 |
| Setting value | 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 Address: FFFFF482H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Bit name | ME7 | 0 | 0 | BT70 | ME6 | 0 | BT61 | BT60 | ME5 | 0 | 0 | BT50 | ME4 | 0 | BT41 | BT40 |
| Setting value | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| | | | | $\overline{CS7}$ | | | $\overline{CS6}$ | | | | | $\overline{CS5}$ | | | $\overline{CS4}$ | |

| Bit position | Bit name | Description |
|---|---|---|
| 15, 11, 7, 3 (BCT0), 15, 11, 7, 3 (BCT1) | Men (n = 0 to 7) | Selects memory controller operation enable setting for each chip select signal n<br>0: Operation disabled (ME5)<br>1: Operation enabled (ME0 to ME4, ME6, ME7) |
| 8, 0 (BCT0), 12, 4 (BCT1) | BTn0 (n = 0, 2, 5, 7) | Selects directly connected device for each chip select signal n<br>0: SRAM or external I/O (BT00, BT50, BT70)<br>1: Page ROM (BT20) |
| 13, 12, 5, 4 (BCT0) 9, 8, 1, 0 (BCT1) | BTn1, BTn0 (n = 1, 3, 4, 6) | Selects directly connected device for each chip select signal n<br>00: SRAM or external I/O (BT11/BT10, BT61/BT60)<br>10: EDO DRAM (BT31/BT30)<br>11: SDRAM (BT41/BT40) |

**[Program example]**

```
[file:bus_init.s]

#### Initialization of memory block function ####
     mov    0x0401,r6            -- Area 0 (64 MB)
                                 -- 00000000H to 001fffffH:  CS0 (2 MB)
                                 -- 00200000H to 003fffffH:  CS1 (2 MB)
                                 -- 00400000H to 005fffffH:  CS2 (2 MB)
                                 -- 00600000H to 03ffffffH:  CS1 (58 MB)
                                 -- Area 1 (64 MB)
                                 -- 04000000H to 07ffffffH:  CS3 (64 MB)
                                 -- Area 2 (64 MB)
                                 -- 08000000H to 0bffffffH:  CS4 (64 MB)
     mov    0x0001,r7            -- Area 3 (64 MB)
                                 -- 0c000000H to 0fdfffffH:  CS6 (62 MB)
                                 -- 0fe00000H to 0fffffffH:  CS7 (2 MB)
     st.h   r6,    CSC0[r0]      -- Chip area select control register 0
     st.h   r7,    CSC1[r0]      -- Chip area select control register 1

     mov    0xa988,r6            -- CS0:  No connected device
                                 -- CS1:  SRAM        00200000H to 002fffffH (1 MB)
                                 -- CS2:  PAGE ROM    00400000H to 0047ffffH (512 KB)
                                 -- CS3:  EDO DRAM    04000000H to 041fffffH (2 MB)

     mov    0x880b,r7
                                 -- CS4:  SDRAM       08000000H to 087fffffH (8 MB)
                                 -- CS5:  No space
                                 -- CS6:  Flash ROM   c0000000H to c003ffffH (256 KB)
                                 -- CS7:  I/O         fe000000H to fe0000fH (16 Byte)
     st.h   r6,    BCT0[r0]      -- Bus cycle type configuration register 0
     st.h   r7,    BCT1[r0]      -- Bus cycle type configuration register 1
```

**(4) Points concerning initialization of bus size**

Be sure to reset before writing to the BSC register.

Do not change register values after writing.

**Figure 4-11. Settings in Bus Size Configuration Register (BSC)**

Address: FFFFF066H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| Bit name | 0 | BS70 | 0 | BS60 | 0 | BS50 | 0 | BS40 | 0 | BS30 | 0 | BS20 | 0 | BS10 | 0 | BS00 |
| Setting value | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| | | $\overline{CS7}$ | | $\overline{CS6}$ | | $\overline{CS5}$ | | $\overline{CS4}$ | | $\overline{CS3}$ | | $\overline{CS2}$ | | $\overline{CS1}$ | | $\overline{CS0}$ |

| Bit position | Bit name | Description |
|---|---|---|
| 14, 12, 10, 8, 6, 4, 2, 0 | BSn0 (n = 0 to 7) | Selects memory controller operation enable setting for individual chip select signal n<br>0: 8 bits (BS70)<br>1: 16 bits (BS00 to BS60) |

**Remark** The initial value in ROMless mode 1 is 0000H (8-bit data bus width for entire space).

**(5) Points concerning initialization of endian format**

Note with caution that various use constraints exist for the big endian format when using NEC's original development tools (debuggers and compilers).

For details of these constraints, see **4.5.4 Big endian method usage restrictions in NEC development tools** in the **V850E/MA1 User's Manual Hardware**.

**Figure 4-12. Settings in Endian Configuration Register (BEC)**

Address: FFFFF068H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| Bit name | 0 | BE70 | 0 | BE60 | 0 | BE50 | 0 | BE40 | 0 | BE30 | 0 | BE20 | 0 | BE10 | 0 | BE00 |
| Setting value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | $\overline{CS7}$ | | $\overline{CS6}$ | | $\overline{CS5}$ | | $\overline{CS4}$ | | $\overline{CS3}$ | | $\overline{CS2}$ | | $\overline{CS1}$ | | $\overline{CS0}$ |

| Bit position | Bit name | Description |
|---|---|---|
| 14, 12, 10, 8, 6, 4, 2, 0 | BEn0 (n = 0 to 7) | Selects memory controller operation enable setting for each chip select signal n<br>0: Little endian (BE70 to BE00) |

**[Program example]**

```
[file:bus_init.s]

#### Bus access initialization ####
     mov    0x1555,r6            -- CS0 to CS6 (Memory) space, 16-bit bus
                                 -- CS7 (I/O) space, 8-bit bus
     st.h   r6,    BSC[r0]       -- Bus size configuration register
     mov    0x0000,r6            -- Little endian in entire space CS0 to CS7
     st.h   r6,    BEC[r0]       --Endian configuration register
```

**(6) Points concerning initialization of programmable wait**

- Programmable waits cannot be used in internal ROM areas, internal RAM areas, or on-chip peripheral I/O areas.
- Wait control of page ROM on-page access, EDO DRAM access, and SDRAM access is performed by the each memory controller.
- Be sure to reset before writing to the DWC0, DWC1 registers. Do not change register values after writing.

**Figure 4-13. Settings in Data Wait Control Registers 0 and 1 (DWC0 and DWC1)**

DWC0 Address: FFFFF484H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| Bit name | 0 | DW32 | DW31 | DW30 | 0 | DW22 | DW21 | DW20 | 0 | DW12 | DW11 | DW10 | 0 | DW02 | DW01 | DW00 |
| Setting value | 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 Address: FFFFF486H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| Bit name | 0 | DW72 | DW71 | DW70 | 0 | DW62 | DW61 | DW60 | 0 | DW52 | DW51 | DW50 | 0 | DW42 | DW41 | DW40 |
| Setting value | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | $\overline{CS7}$ | | | | $\overline{CS6}$ | | | | $\overline{CS5}$ | | | | $\overline{CS4}$ | | |

| Bit position | Bit name | Description |
|---|---|---|
| 14 to 12,<br>10 to 8,<br>6 to 4,<br>2 to 0 | DWn2 to DWn0<br>(n = 0 to 7) | Specifies the number of wait states to be inserted in the CSn space<br>• $\overline{CS0}$ (EPROM space)<br>101: Insert 5 waits (DW02 to DW00)<br>• $\overline{CS1}$ (SRAM space)<br>001: Inserts 1 wait (DW12 to DW10)<br>• $\overline{CS2}$ (page ROM space)<br>101: Inserts 5 waits (DW22 to DW20)<br>• $\overline{CS3}$ to $\overline{CS5}$<br>000: None inserted (DWn2 to DWn0: n = 3 to 5)<br>• $\overline{CS6}$ (flash memory space)<br>111: Inserts 7 waits (DW62 to DW60)<br>• $\overline{CS7}$ (I/O space)<br>001: Inserts 1 wait (DW72 to DW70) |

**(7) Points concerning initialization of address setup wait**

- Address setup waits cannot be used in EDO DRAM areas or SDRAM areas.
- Address setup waits cannot be inserted using the $\overline{\text{WAIT}}$ pin's external wait function.

**Figure 4-14. Settings in Address Setup Wait Control Register (ASC)**

Address: FFFFF48AH

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Bit name | AC71 | AC70 | AC61 | AC60 | AC51 | AC50 | AC41 | AC40 | AC31 | AC30 | AC21 | AC20 | AC11 | AC10 | AC01 | AC00 |
| Setting value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | $\overline{\text{CS7}}$ | | $\overline{\text{CS6}}$ | | $\overline{\text{CS5}}$ | | $\overline{\text{CS4}}$ | | $\overline{\text{CS3}}$ | | $\overline{\text{CS2}}$ | | $\overline{\text{CS1}}$ | | $\overline{\text{CS0}}$ | |

| Bit position | Bit name | Description |
|---|---|---|
| 15 to 0 | ACn1, ACn0<br>(n = 0 to 7) | Specifies the number of wait states to be inserted in CSn space<br>00: None inserted (ACn1, ACn0) (n = 0 to 7) |

**Figure 4-15. Settings in Bus Cycle Period Control Register (BCP)**

Address: FFFFF48CH

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit name | BCP | 0 | 0 | 0 | IOEN | 0 | 0 | 0 |
| Setting value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit position | Bit name | Description |
|---|---|---|
| 7 | BCP | Selects bus cycle period<br>0: Normal |
| 3 | IOEN | Enables/disables operation of $\overline{\text{IORD}}$ and $\overline{\text{IOWR}}$ in SRAM, external ROM, and external I/O cycles<br>0: Disable |

**(8) Points concerning initialization of bus cycle period**

- During flyby DMA transfers that target the SRAM, external ROM, or external I/O, $\overline{\text{IORD}}$ and $\overline{\text{IOWR}}$ signals are output regardless of the IOEN bit's setting.
- To set a bus cycle period that is one-half of the internal system clock, use the BUSCLK pin. After setting "1" to the IOEN bit in the BCP register, be sure to also set the PMCCM and PFCCM registers.
- Be sure to reset before writing to the BCP register. Do not change register values after writing.

**[Program example]**

```
[file:businit.s]

#### Initialization of wait function ####
     mov    0x0515,r6          -- CS0:  Insert 5 waits (TW) in EPROM space
                               -- CS1:  Insert 1 wait (TW) in SRAM space
                               -- CS2:  Insert 5 waits (TW) in PAGE ROM space
                               -- CS3:  0 waits (TW) in EDO DRAM space
     mov    0x1700,r7          -- CS4:  0 waits (TW) in SDRAM space
                               -- CS6:  Insert 7 waits (TW) in Flash ROM space
                               -- CS7:  Insert 1 wait (TW) in I/O space
     st.h   r6,    DWC0[r0]    -- Data wait control register 0
     st.h   r7,    DWC1[r0]    -- Data wait control register 1

     mov    0x0000,r6          -- CS0 to CS7:  0 waits (TASW) in entire space
     st.h   r6,    ASC[r0]     -- Address setup control register

     mov    0x00,r6            -- Normal cycle
                               -- Disables operation of IORD and IOWR in SRAM, external
                                  ROM, or external I/O cycles
     st.b   r6,    BCP[r0]     -- Bus cycle period control register
```

**(9) Points concerning initialization of idle state**

- Idle states cannot be used in internal ROM areas, internal RAM areas, or on-chip peripheral I/O areas.
- Be sure to reset before writing to the BCC register. Do not change register values after writing.

**Figure 4-16. Settings in Bus Cycle Control Register (BCC)**

Address: FFFFF488H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Initial value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Bit name | BC71 | BC70 | BC61 | BC60 | BC51 | BC50 | BC41 | BC40 | BC31 | BC30 | BC21 | BC20 | BC11 | BC10 | BC01 | BC00 |
| Setting value | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| | $\overline{CS7}$ | | $\overline{CS6}$ | | $\overline{CS5}$ | | $\overline{CS4}$ | | $\overline{CS3}$ | | $\overline{CS2}$ | | $\overline{CS1}$ | | $\overline{CS0}$ | |

| Bit position | Bit name | Description |
|---|---|---|
| 15 to 0 | BCn1, BCn0<br>(n = 0 to 7) | Specifies the number of wait states to be inserted in $\overline{CSn}$ space<br>• $\overline{CS0}$ (EPROM space)<br>11: Insert 3 idle states (BC01 and BC00)<br>• $\overline{CS1}$ (SRAM space)<br>00: None inserted (BC11 and BC00)<br>• $\overline{CS2}$ (page ROM space)<br>10: Inserts 2 waits (BC21 and BC20)<br>• $\overline{CS3}$ to $\overline{CS5}$<br>00: None inserted (BCn1 and BCn0: n = 3 to 5)<br>• $\overline{CS6}$ (flash memory space)<br>10: Inserts 2 waits (BC61 and BC60)<br>• $\overline{CS7}$ (I/O space)<br>00: None inserted (BC71, BC70) |

**[Program example]**

```
[file:businit.s]
#### Initialization of idle function ####
mov  0x2023,r6            -- CS0:  Insert 3 idle states (TI) in EPROM space
                          -- CS1:  0 idle states (TI) in SRAM space
                          -- CS2:  Insert 2 idle states (TI) in PAGE ROM space
                          -- CS3:  0 idle states (TI) in EDO DRAM space
                          -- CS4:  0 idle states (TI) in SDRAM space
                          -- CS6:  Insert 2 idle states (TI) in Flash ROM space
                          -- CS7:  0 idle state (TI) in I/O space
st.h r6,    BCC[r0]       -- Bus cycle control register
```

### (10) Points concerning initialization of SDRAM

<1> When using an SDRAM, be sure to follow the initialization sequence described below.

- Set PMCn (port n mode control register) (n = 0 to 5, AL, AH, DL, CS, CT, CM, or CD).
- Set CSCn (chip area select control register n), and determine a chip select space (n = 0 or 1).
- Set BCTn (bus cycle type configuration register n), and determine the type of memory device in each chip select space. (n = 0 or 1).
- Set RFSn (SDRAM refresh control register n) (n = 1, 3, 4, or 6).
- Set SCRn (SDRAM configuration register n) (n = 1, 3, 4, or 6).

<2> SDRAM configuration register n (SCRn)

- When accessing the SDRAM area, wait 20 clock cycles after setting registers.
- When initializing the SDRAM area more than once, be sure to enter another instruction between the two register write instructions.
- Be sure to reset before writing to this register. Do not change register values after writing.

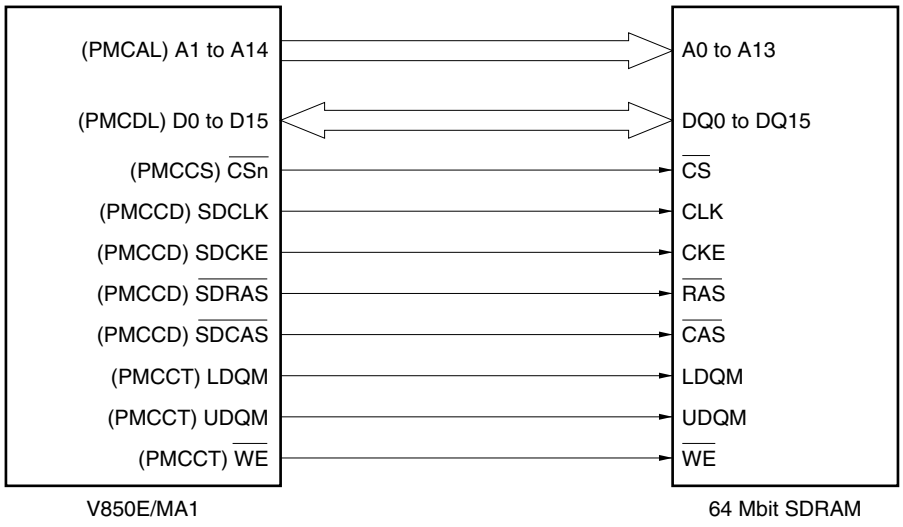
<3> SDRAM refresh control register n (RFSn)

Be sure to reset before writing to this register. Do not change register values after writing.

<4> Refresh wait control register (RWC)

Be sure to reset before writing to this register. Do not change register values after writing.

**Figure 4-17. Image of SDRAM Initialization Sequence**




<1> Set the PMCn (port mode n control register) to the external bus mode used with the SDRAM (external memory space) (n = 0 to 5, AL, AH, DL, CS, CT, CM, or CD). When setting the PMCCD register, do not set SDCLK output mode and SDCKE output mode at the same time. Always set SDCLK output mode first, then set SDCKE output mode.

<2> Set the CSCn chip area select control register n, then determine the chip select space (n = 0 or 1). See **4.2.4 (2) Points concerning initialization of memory block space distribution** ($\overline{CS4}$ 64 MB space for SDRAM)

<3> Set BCTn (bus cycle type configuration register n), and select SDRAM as the type of memory for the $\overline{CS4}$ space.

<4> RFSn: SDRAM refresh control register n (n = 1, 3, 4, or 6)

**Note**

SCRn: SDRAM configuration register n (n = 1, 3, 4, or 6)

**Note** Be sure to set these in this order.

**(11) Points concerning page ROM initialization**

- Be sure to reset before writing to PRC (page ROM configuration register). Do not change register values after writing.
- For on-page access, wait is controlled based on PRC settings.
- For off-page access, wait is controlled based on DWCn settings.

**[Program example]**

```
#### Initialization of memory access control function ####
[file:businit.s]
#########################################################
#    EDO DRAM settings
#########################################################
     mov    0x98,  r6          -- Inserts 2 waits (TRRW) for the RAS signal's high
                                  level width
                               -- Inserts 3 waits (TRCW) for the RAS signal's low level
                                  width
                               -- Inserts 0 waits (TSRW) to cancel self refresh
     st.b   r6,    RWC[r0]     -- Refresh wait control register

     mov    0xb546, r6         -- Enables on-page access
                               -- Inserts 3 waits (TRPW) in row address PRE time
                               -- Inserts 1 wait (TRHW) in row address HOLD time
                               -- Inserts 1 wait (TW) in data access time
                               -- Inserts 1 wait (TCPW) in column address PRE time
                               -- Enables RAS hold mode
                               -- Address shift width:  16 bits (on-page)
                               -- Address multiplex width:  10 bits
     st.h   r6,DRC3[r0]         -- DRAM configuration register 3

     mov    0x8106, r6         -- Enables refresh
                               -- Refresh count clock = 128/fxx
                               -- Refresh interval factor = 7 (17.92 us during 50 MHz
                                  operation)
     st.h   r6,RFC3[r0]         -- Refresh control register 3
```

```
#################################################################
#    SDRAM settings
#################################################################
     mov    0x8106, r6          -- Enables refresh
                                -- Refresh count clock = 128/fxx
                                -- Refresh interval factor = 7 (17.92 μs during 50 MHz
                                   operation)
     st.h   r6,RFC4[r0]         -- SDRAM refresh control register 4

     mov    0x2014, r6          -- CAS latency during read:  2 latency (TLATE)
                                -- ACT-RD or PRE-ACT:  1 wait (TBCW)
                                -- Address shift width:  16 bits (on-page)
                                -- Row address width:  12 bits
                                -- Address multiplex width:  8 bits
     st.h   r6,DRC4[r0]         -- DRAM configuration register 4

#################################################################
#    Wait specifications, etc., for page ROM on-page access
#################################################################
     mov    0x2000, r6          -- 4 words x 16 bits (8 words x 8 bits)
                                -- Inserts 2 waits (TW) for on-page access
     st.h   r6,PRC[r0]          -- Page ROM configuration register
```

### 4.2.5 Operation example using various peripheral functions (STEP3)

In addition to the initialization of the CPU's basic functions, CPU bus control functions are initialized. Also, various on-chip peripheral I/O functions can be initialized to enable operations using a variety of resources.

#### (1) I/O using port functions

##### (a) Overview of functions

The statuses of switches connected to input ports are read and are then displayed via LEDs connected to an output port.

Eight switch inputs are distributed and allocated among two ports, P0 and P5, and eight LEDs are distributed and allocated among two other ports, P3 and PBD. Consequently, the read data is edited before being output.

##### (b) Program list

```
# Set up PORT0 and PORT5 input ports connected to toggle switches so that output goes
# to PORT3
# and PBD output ports connected to LEDs.
# <Registers used>
#    r10:  PORT0 input buffer
#    r11:  PORT5 input buffer
#    r12:  PBD output buffer

# External reference declaration
.extern  _cpu_init
.extern  _bus_init

# Reset and interrupt
.section "RESET",text     -- Reset handler declarations
     jr    _start         -- Jumps to start of sample program code
# Main program
.text
.align 4                  -- 4-byte alignment
.globl _start             -- External declaration

_start:
     jarl  _cpu_init,r31  -- Initialization block of CPU basic functions
      jarl _bus_init,r31  -- Initialization block of CPU bus control functions
_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) Display processing based on external interrupt processing

#### (a) Overview of function

Each time that the INT switch (connected to the $\overline{\text{INTP110}}$ pin) is pressed, the output port settings are toggled between “all lit” and “all not lit”. This program does not retain register settings, but register settings must be saved or restored in an actual interrupt processing program.

#### (b) Program list

```
# An external interrupt is occurred by INTP110 which is connected to the INT switch.
# Each time that the INT switch is pressed, the settings for PORT3 and PBD
# (output ports connected to LEDs) are toggled between "all lit" and "all not lit".
# <Registers used>
#    r10:  Buffer fetching LED's higher 4 bits (temporary)
#    r11:  Buffer fetching LED's lower 4 bits
#    r12:  LED flag

#External reference declarations
.extern  _cpu_init
.extern  _bus_init

# Reset and interrupt
.section "RESET",text      -- Reset handler declaration
     jr     _start         -- Jumps to start of sample program code
.section "INTP110",text    -- INTP110 interrupt handler declaration
     jr     _int_p110      -- Jumps to start of INTSW interrupt processing

#Main program
.text
.align 4                   -- 4-byte alignment
.globl _start              -- External declaration

_start:
     jarl   _cpu_init,r31 -- Initialization block of CPU basic functions
     jarl   _bus_init,r31 -- Initialization block of CPU bus control functions

     st.b   r0,    INTM2  -- Falling edge specification for INTP110 interrupt
     mov    0x07,  r10    -- Clears INTP110 mask, level 7
     st.b   r10,   P11IC0
     mov    r0,    r12    -- Initialization of LED flag
     ei                   -- Enables interrupts
```

```
_forever:
     nop
     jr     _forever      -- Endless loop

#    INTP110 (INT switch) interrupt processing
_int_p110:
     not    r12,   r12           -- invert LED flag
     mov    r12,   r10           -- Copy to temporary
     mov    r10,   r11           -- Copy to temporary
     andi   0xf0,  r10,  r10     -- Fetch higher 4 bits
     st.b   r10,   P3            -- Output to P3
     andi   0x0f,  r11,  r11     -- Fetch lower 4 bits
     st.b   r11,   PBD           -- Output to PBD
     reti                        -- Restore from interrupt processing
```

### (3) Display updates using timer function

#### (a) Overview of function

Internal interrupt request (INTM000) is generated and LED settings are incremented and displayed via interrupt processing using timer C0.

Values set to the timer assume a 40 MHz operating clock, and the setting values are those for which match interrupts occur every 10 ms in a compare register.

#### (b) Points concerning initialization of timer function

The following is an example of timer function initialization using timer mode control register 10 (TMCC10).

- The TMCCAE1 bit cannot be set at the same time as another bit. Therefore, be sure to set the TMCCAE1 bit before setting other bits and before making settings in other registers in the TMC1 unit.
- When TMCCE1 = 1, any repeated setting of TMCCE1 = 1 is ignored. To restart to timer, set TMCCE1 = 0, then set TMCCE1 = 1.

| Bit Position | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit Name | OVF1 | CS12 | CS11 | CS10 | 0 | 0 | TMCCE1 | TMCCAE1 |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Setting value <1> | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Setting value <2> | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Setting value <3> | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| Setting value <4> | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |

Be sure to make these settings in the following order.

Setting value <1>: Reset (initial setting after power-on)

Setting value <2>: Start TMCCAE1 (clock supply)

Setting value <3>: Select count clock

Setting value <4>: Enable count operation (TMCCAE1: 0 → 1)

**Caution Setting <2> and <3> above at the same time invalidates the count clock selection.**

**(c) Program example of timer function initialization**

```
#### Initialization of timer function ####
   set1   0x0,   TMCC10      -- Starts clock supply to TMC1 unit (TMCCAE1 = 1)
   ld.b   TMCC10,   r10
   ori    0x50,  r10,   r10  -- Selects TMC1 internal count clock,
                                fxx/128 = 3.2 us (40 MHz)          } First setting
   st.b   r10,   TMCC10
   <Abridged>
   set1   0x1,   TMCC10      -- Enables TMC1 count
   <Timer count processing 1>
   ld.h   TMC1,  r15         -- Reads TMC1 count

   clr1   0x1,   TMCC10      -- Resets TMC1 count (TMCCE1 = 0)   ↓ Note  } Second and
   set1   0x1,   TMCC10      -- Enables TMC1 count (TMCCE1 = 1)          subsequent settings
   <Timer count processing 2>
   ld.h  TMC1, r15           -- Reads TMC1 count
```

**Note** Restarts when setting is changed from 0 to 1.

**(d) Pattern of faulty initial setting**

```
movea    0x51,r0,r10     -- Starts clock supply to TMC1 unit (TMCCAE1 = 1)
st.b     r10, TMCC10     -- Selects TMC1 internal count clock, fxx/128 = 3.2 us
                            (40 MHz)
set1     0x1, TMCC00     -- Enables TMC1 count
```

The TMCCAE1 bit cannot be set at the same time as another bit. Therefore, when 51H is set to the TMCC10 register, the TMCCAE1 bit's value becomes "1" but the initial value (000) is retained in bits CS12 to CS10 of the TMCC10 register and timer operation is based on fxx/4.

**(e) Pattern of faulty restart**

```
set1      0x0,     TMCC10     -- Starts clock supply to TMC1 unit (CAE1 = 1) Note 1
ld.b      TMCC10,r10
ori       0x50,    r10,r10    -- Selects TMC1 internal count clock, fxx/128 = 3.2 us
                                 (40 MHz)
st.b      r10,     TMCC10
ld.h      TMC1,    r15        -- Reads TMC1 count
set1      0x1,     TMCC10     -- Enables TMC1 count
      <Time count processing 1>
ld.h      TMC1,    r15        -- Reads TMC1 count
set1      0x1,     TMCC10     -- Enables TMC1 count (TMCCE1 = 1) Note 2
      <Time count processing 2>
ld.h      TMC1,    r15        -- Reads TMC1 count
```

**Notes 1.** Setting is 0 → 1 since 0 is initial setting.

**2.** Setting is 1 → 1 since it is the second setting.

When set in this way, the count value from <Time count processing 2> is a continuation of the count from <Time count processing 1>.

**(f) Program list**

```
# Timer C0 is used to increment and display the LED data in every 10 ms.
# <Registers used>
#    r10:  Temporary
#    r11:  Temporary
#    r12:  Counter

#External reference declaration
.extern     _cpu_init
.extern     _bus_init

#Reset and interrupt
.section "RESET",text            -- Reset handler declaration
     jr     _start               -- Jumps to start of sample program code
.section "INTM000",text          -- INTM000 interrupt handler declaration
     jr     _int_tm000           -- Jumps to start of timer C0 compare match interrupt
processing

#Main program
.text
.align 4                         -- 4-byte alignment
.globl _start                    -- External declaration
```

```
_start:
     jarl   _cpu_init,r31        -- Initialization block of CPU basic functions
     jarl   _bus_init,r31        -- Initialization block of CPU bus control functions

#Initialization of timer C0
     set1   0x0, TMCC00          -- TMCCAE0 = 1:  Starts clock supply to TMC0 (entire
                                    unit)
     ld.b   TMCC00,r10           --
     ori    0x50,  r10,    r10   -- CS02, CS01, CS00 = 1, 0, 1
     st.b   r10,   TMCC00        -- Selects TMC0 internal count clock, fXX/128 = 3.2 us
                                    (40 MHz)

     mov    0x09,  r10           -- Auto clear and start (CCLR0)
     st.b   r10,   TMCC01        -- Select compare mode (CMS00)

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

     mov    0x07,  r10           -- Clears compare match interrupt masking, level 7
     st.b   r10,   P00IC0
     set1   0x1,   TMCC00        -- Enables TMC0 count (TMCCE0 = 1)

     st.b   r0,    P3            -- Sets higher 4 bits of LED not to lit
     st.b   r0,    PBD           -- Sets lower 4 bits of LED not to lit
     mov    r0,    r12           -- Initializes counter

     ei                          -- Enables interrupts
_forever:
     nop
     jr     _forever             -- Endless loop

#    INTM000 (10 ms) interrupt processing
_int_tm000:
     addi   1,     r12,    r12   -- Counts up
     mov    r12,   r10           -- Copies counter to temporary
     mov    r10,   r11           -- Copies counter to temporary
     andi   0xf0,  r10,    r10   -- Fetches higher 4 bits of counter value
     st.b   r10,   P3            -- Outputs to P3
     andi   0x0f,  r11,    r11   -- Fetches lower 4 bits of counter value
     st.b   r11,   PBD           -- Outputs to PBD
     reti                        -- Restore from interrupt processing
```

### (4) Control of counter display based on multiple interrupts

#### (a) Overview of function

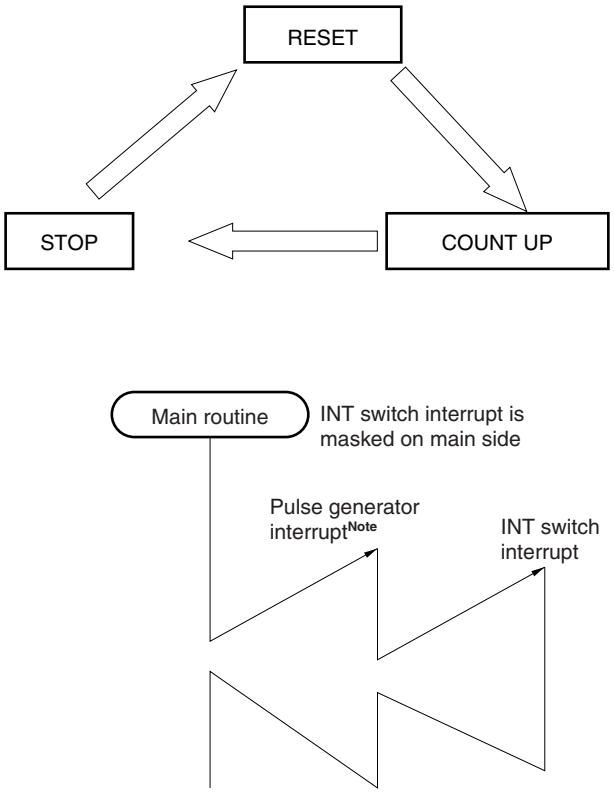
Signals from the pulse generator that are input to the $\overline{\text{INTP111}}$ pin are counted (counter values range from 0 to 9999) and are displayed via the 7-segment LED.

When the INT switch is pressed, the counter operation mode is switched among three modes. An interrupt request triggered by the INT switch is enabled (mask cleared) only when included in a pulse generator interrupt to ensure that multiple interrupts do not occur.

When the debugger is started and executed, the 7-segment LED counter display is stopped at “0000” (RESET mode). If the INT switch is pressed while in RESET mode, an $\overline{\text{INTP110}}$ signal is generated to start the count operation (COUNT UP mode). The next time the INT switch is pressed, the count is stopped and the count value is retained (STOP mode). When the INT switch is pressed again, the counter value returns to “0000” (RESET mode).

When the INT switch is pressed, an $\overline{\text{INTP110}}$ signal is generated and processing continues as indicated by the arrows in the figure below.

**Figure 4-18. Control of Counter Display Based on Multiple Interrupts**




**Note** Masking of INT switch interrupts can be cleared only within pulse generator interrupt processing routines.

**Remark** Priority: Pulse generator interrupt < INT switch interrupt

### (b) Program list

```
# Pulse generator interrupt is used to increment value displayed via 7-segment
# LED in range from 0 to 9999.
# Each time the INT switch is pressed, the mode is switched among three modes.
# INT switch interrupts have higher priority than pulse generator interrupts,
# but INT switch interrupts are allowed only within pulse generator interrupts.
#
# MODE 0:  Reset counter
# MODE 1:  Continue counter
# MODE 2:  Stop counter
# <Registers used>
#    r10:  Temporary
#    r11:  Counter mode
#    r12:  Display request flag
#    r13:  Counter
#    r14:  Dummy counter

#External reference declaration
.extern  _cpu_init
.extern  _bus_init
.extern  _disp_num

#Reset and interrupt
.section "RESET",text            -- Reset handler declaration
     jr     _start               -- Jumps to start of sample program code
.section "INTP110",text          -- INTP110 interrupt handler declaration
     jr     _int_p110            -- Jumps to start of INTSW interrupt processing block
.section "INTP111",text          -- INTP111 interrupt handler declaration
     jr     _int_p111            -- Jumps to start of pulse generator interrupt
                                    processing block

#Main program
.text
.align 4                         -- 4-byte alignment
.globl _start                    -- External declaration

_start:
     jarl   _cpu_init,r31        -- Initialization of CPU's basic functions
     jarl   _bus_init,r31        -- Initialization of CPU's bus control functions
     mov    0xfffc200, r3        -- Sets stack pointer (Start of internal RAM + 0x200)
     st.b   r0,    P3
     st.b   r0,    PBD
```

```
     mov    0x04,  r10          -- Falling edge of INTP110, rising edge of INTP111
     st.b   r10,   INTM2
     mov    0x46,  r10          -- INT switch interrupt (INTP110)
     st.b   r10,   P11IC0       -- Masks interrupt, priority level 6
     mov    0x07,  r10          -- Pulse generator interrupt (INTP111)
     st.b   r10,   P11IC1       -- Clears interrupt mask, priority level 7
     mov    r0,    r11          -- Initializes counter mode
     mov    r0,    r12          -- No display request
     mov    r0,    r13          -- Initializes counter
     ei
_loop:
     cmp    r0,    r12          -- Display request ON?
     be     _loop               -- No processing if OFF
     mov    r0,    r12          -- If ON, sets display request to "OFF"
     mov    r13,   r6           -- Passes counter value to argument
     jarl   _num_disp,r31       -- Calls display processing block
     br     _loop

##########################################
#    INTP110 (INT-SW) interrupt processing
##########################################
_int_p110:
     -- Saves register
     add    -4,    r3
     .option nowarning
     st.w   r1,    0[r3]
     .option warning

     cmp    2,     r11          -- Counter mode 2?
     bz     _int_p110_2
     -- Not counter mode 2
     add    1,     r11          -- Counter mode + 1
     br     _int_p110_ext
_int_p110_2:
     -- Counter mode 2
     mov    r0,    r11          -- Initializes to counter mode 0
_int_p110_ext:
     -- Restores register
     .option nowarning
     ld.w   0[r3],  r1
     .option warning
     add    4,     r3
     reti
```

```
##############################################
#  INTP111 (Pulse-Gen) interrupt processing
##############################################
_int_p111:
     -- Saves EIPC and EIPSW registers
     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        -- Clears INTP110 (INTSW) interrupt masking
     ei                          -- Enables multiple interrupts
#### INTSW interrupts are enabled after this point. ####
     cmp    r0,    r11           -- Counter mode 0?
     bne    _int_p111_1

     -- Counter mode 0 (counter reset display "0000")
     mov    r0,    r13           -- Initializes counter
     movea  0xff,  r0,   r12     -- Display request ON
     br     _int_ret

     -- Counter mode 1 (count-up display in range from 0 to 99999)
_int_p111_1:
     cmp    1,     r11           -- Counter mode 1?
     bne    _int_ret
     add    1,     r13           -- Counter + 1
     cmp    99999, r13
     bnh    _int_p111_nc         -- Over count upper limit?
     mov    r0,    r13           -- Initializes counter
_int_p111_nc:
     movea  0xff,  r0,   r12     -- Display request ON

     -- Counter mode 2 (stop counter)
_int_ret:
     di
     set1   6,     P11IC0        -- Masks INTP110 (INTSW) interrupt
#### INTSW interrupts are enabled up to this point ####
```

```
     -- Restores EIPC and EIPSW registers
     .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
# Conversion of binary data to decimal and display via 7-segment LED
# r6:   Binary data to be converted (argument)
# r20:  Divide by 10 quotient
# r21:  Divide by 10 remainder
# r22:  Display buffer address
# r23:  Start address of numerical pattern table
# r24:  Work
# r25:  Value returned from div10 processing is stored in destination address
#   (start address of 7-segment LED)
#
# Displays lower five digits of conversion result
#
.set        DISPBUFSIZE, 10
.bss
.lcomm             _disp_buf, DISPBUFSIZE, 4

.text
.globl             _num_disp
_num_disp:
# Saves registers (r1 and r31)
     addi   -8,    r3,    r3
     .option nowarning
     st.w   r1,    0[r3]
     .option warning
     st.w   r31,   4[r3]
```

```
# Clears display buffer to 0
_bindec:
     mov    #_disp_buf,   r22    -- Sets start address of display buffer
     mov    10,    r24           -- Sets zero-clear loop counter
_bindec1:
     st.b   r0,    0[r22]        -- Clears to 0
     add    1,     r22           -- Increments address
     add    -1,    r24           -- Decrements counter
     bnz    _bindec1             -- 10 times completed?

# Binary-decimal conversion
     mov    #_disp_buf+9, r22    -- Sets end address of display buffer
_bindec2:
     and    r6,    r6            -- Is dividend 0?
     bz     _bindec3             -- If dividend is 0, stops division processing.
     jarl   _div10,r31           -- Calls divide by 10 processing
     st.b   r21,   0[r22]        -- Stores quotient to display buffer
     add    -1,    r22           -- Decrements display buffer
     mov    r20,   r6            -- Resets remainder to dividend
     br     _bindec2

# 5-digit display output to 7-segment LED
_bindec3:
     movhi  0x0fe0,r0,    r25    -- Sets start address of 7-segment LED register
     mov    #_num_data,   r23    -- Sets start address of numerical pattern table
     mov    #_disp_buf,   r22    -- Sets start address of display buffer

     ld.b   5[r22],r20           -- Reads decimal number for fifth digit (10,000+)
     add    r23,   r20           -- Adds table address offset
     ld.b   0[r20],r24           -- Reads numerical pattern
     st.b   r24,   8[r25]        -- Outputs fifth digit to 7-segment LED

     ld.b   6[r22],r20           -- Reads decimal number for fourth digit (1,000+)
     add    r23,   r20           -- Adds table address offset
     ld.b   0[r20],r24           -- Reads numerical pattern
     st.b   r24,   6[r25]        -- Outputs fourth digit to 7-segment LED
     ld.b   7[r22],r20           -- Reads decimal number for third digit (100+)
     add    r23,   r20           -- Adds table address offset
     ld.b   0[r20],r24           -- Reads numerical pattern
     st.b   r24,   4[r25]        -- Outputs third digit to 7-segment LED

     ld.b   8[r22],r20           -- Reads decimal number for second digit (10+)
     add    r23,   r20           -- Adds table address offset
```

```
ld.b 0[r20],r24           -- Reads numerical pattern
     st.b   r24,   2[r25]        -- Outputs second digit to 7-segment LED

     ld.b   9[r22],r20           -- Reads decimal number for first digit (1+)
     add    r23,   r20           -- Adds table address offset
     ld.b   0[r20],r24           -- Reads numerical pattern
     st.b   r24,   0[r25]        -- Outputs first digit to 7-segment LED

# Restores registers (r1 and r31)
     ld.w   4[r3], r31
     .option nowarning
     ld.w   0[r3], r1
     .option warning
     addi   8,     r3,    r3
     jmp    [r31]

#
# Divide by 10 processing
#
#  r6:  Dividend (div10)
# r10:  Divide by 10 quotient (div10)
# r11:  Divide by 10 remainder (div10)

_div10:
     mov    r0,    r20          -- Initializes quotient to 0
_div10_1:
     cmp    10,    r6           -- Is dividend less than 10?
     bn     _div10_2            -- End subtraction if less than 10
     add    -10,   r6           -- Subtracts 10 from dividend
     add    1,     r20          -- Counts up number of subtractions (quotient)
     br     _div10_1
_div10_2:
     mov    r6,    r21          -- Remaining value after last subtraction (remainder)
     jmp    [r31]               -- Returns to caller

# Numerical pattern table
.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) Using DMA function to transfer data between memories

#### (a) Overview of function

Using DMA channel 0, two cycle transfers are performed to transfer data from SRAM to EDO DRAM in block transfer mode. This 512-byte DMA transfer sends half-word data 256 times from the SRAM's start address (0x200000) to the EDO DRAM's start address (0x4000000). Eight LEDS are all lit when the DMA transfer completion interrupt occurs.

#### (b) Program list

```
# DMA's block 2 cycle transfer is used to transfer data (16 bits x 256 times
# (= 512 bytes))
# in the SRAM area to the EDO DRAM area.
# <Registers used>
#    r10:  Temporary
#    r11:  Temporary
#    r12:  Temporary

#External reference declaration
.extern     _cpu_init
.extern     _bus_init

#Constant definitions
.set EDO DRAM,     0x04000000    -- EDO DRAM address definition
.set SRAM,         0x00200000    -- SRAM address definition

#Reset and interrupt
.section "RESET",text            -- Reset handler declaration
     jr     _start               -- Jumps to start of sample program code
.section "INTDMA0",text          -- INTDMA0 interrupt handler declaration
     jr     _int_dma0            -- Jumps to start of DMA0 transfer completion
                                    interrupt processing block

#Main program
.text
.align 4                         -- 4-byte alignment
.globl _start                    -- External declaration

_start:
     jarl   _cpu_init,r31        -- Initialization block of CPU basic functions
     jarl   _bus_init,r31        -- Initialization block of CPU bus control functions

     st.b   r0,    P3            -- Higher 4 bits of LED not lit
     st.b   r0,    PBD           -- Higher 4 bits of LED not lit
```

```
#DMA0 initialization
     mov    SRAM,  r10          -- Sets transfer source address
     mov    r10,   r11          -- Copies transfer source address
     shr    16,    r10          -- Higher 12-bit side of DMA0 transfer address
     st.h   r10,   DSA0H
     andi   0xffff,r11,   r11   -- Lower 16-bit side of DMA0 transfer source address
     st.h   r11,   DSA0L

     mov    EDO DRAM,r10        -- Sets transfer destination address
     mov    r10,   r11          -- Copies transfer destination address
     shr    16,    r10          -- Higher 12-bit side of DMA0 transfer destination
                                    source address
     st.h   r10,   DDA0H
     andi   0xffff,r11,   r11   -- Lower 16-bit side of DMA0 transfer destination
                                    address
     st.h   r11,   DDA0L

     mov    0xff,  r10          -- Sets DMA transfer count (256 times)
     st.h   r10,   DBC0

     mov    0x400c,       r10    -- Sets DMA0 transfer mode
                                 -- 16-bit data, transfer source++/transfer
                                    destination++,
     st.h   r10,   DADC0        -- Block transfer mode, 2-cycle transfer

     mov    0x07,  r10          -- Clears masking, level 7
     st.b   r10,   DMAIC0       -- Settings in DMA0 transfer completion interrupt
                                    control register

     ei                         -- Enables interrupts

     st.b   r0,    DTFR0        -- Prohibits DMA requests from on-chip peripheral I/O

#    ld.b   DCHC0, r10          -- Reads and discards TC0 value (required during
                                    multiple starts)

     mov    0x03,  r10          -- Enables (E00 = 1) and starts (STG0 = 1) DMA
                                    transfers
     st.b   r10,   DCHC0
_forever:
     nop
     jr     _forever            -- Endless loop

#    INTDMA0 interrupt processing
_int_dma0:
     mov    0xf0,  r12
     st.b   r12,   P3           -- Higher 4 bits of LED lit
     mov    0x0f,  r12
     st.b   r12,   PBD          -- Lower 4 bits of LED lit
reti
```

### (6) Use of timer function to measure pulse frequency

#### (a) Overview of function

This repeats the processing that counts the number of pulses entered during one second and display the count value. Values set to the timer are based on a 50 MHz operating clock and the setting values are those for which match interrupts occur every 10 ms in a compare register. When timer interrupts have occurred 100 times, the system prepares to display the number of pulses counted since the last timer interrupt. The main program displays the count value.

#### (b) Program list

```
/**********************************/
/*   Display pulse frequency       */
/**********************************/

#include    "common.h"

void int_tm000 (void);           /* 10 ms interval interrupts */
void int_p111 (void);            /* Pulse generator interrupt */

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

main ()
{
     init_pmode ();
     TMCC00.0 =     1;           /* TMC0 reset */
     TMCC00 |=      0x50;        /* TMC0 fxx/128 (t = 2.56 us) during 50 MHz operation
                                 */
     TMCC01 =       0x09;        /* auto clear&start , CCC00 is compare */
     CCC00 =        3906;        /* for 10 ms */
     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) interrupt processing */
/*****************************************/
#pragma interrupt INTP111 int_p111
_ _interrupt
void int_p111 (void)
{
     if (counter<99999)
            counter++;
}
/*************************************/
/* INTM000 (10 ms) interrupt processing */
/*************************************/
#pragma interrupt INTP000 int_tm000
_ _interrupt
void int_tm000 (void)
{
     times++;
     if (times==100)
     {
            times=0;
            freq = counter;
            counter = 0;
            disp_req = ON;
     }
}
```

### (7) Use of A/D and D/A conversion functions for sine wave I/O

#### (a) Overview of function

Sine waves that are input to analog input pins are sampled at an interval of 125 $\mu$s, and A/D conversion is then started. When interrupt processing occurs at the completion of A/D conversion, the conversion results are output via a speaker connected to an R-2R (D/A conversion) circuit.

#### (b) Program list

```
/*************************************/
/* A/D conversion results of sine waves */
/* entered every 125 us are output via  */
/* an R-2R (D/A) speaker.               */
/*************************************/

#include    "common.h"

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

main ()
{

     TMCC00.0 = 1;        /* TMC0 reset */
     TMCC00 |= 0x10;      /* TMC0 fxx/8 (t = 0.16 us) during 50 MHz operation */
     TMCC01 = 0x09;       /* auto clear&start , CCC00 is compare */
     CCC00 = 781;         /* for 125 us */
     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 (125 us) interrupt processing */
/*****************************************/
#pragma interrupt INTM000 int_tm000
_ _interrupt
void int_tm000 (void)
{
     ADM0 |= 0x80;  /* Converter enable */
}

/*********************************/
/*   INTAD interrupt processing   */
/*********************************/
#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) Use of PWM for DC motor speed control

#### (a) Overview of function

Values set via switches are used for speed control based on PWM duty factors.

This program changes the settings made to the PWM each time the switch states have changed, and displays the values via LEDs.

#### (b) Program list

```
/*************************************/
/* Switch to PWM0 & LED              */
/* Monitor switches and reflect switch */
/* setting changes in PWM            */
/*************************************/

#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;         /* Switch input */
     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;  /* Switch input */
            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) Use of CSI for EEPROM write/read processing

#### (a) Overview of function

The CSI function is used to perform read and write operations to and from an EEPROM that is connected to CSI2. After a 16-word segment of data has been written, the data is read and displayed via the 7-segment LED.

#### (b) Program list

```
/**********************************/
/*  Serial EEPROM interface using  */
/*  CSI function                   */
/**********************************/

#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 = 0.64 us) during 50 MHz operation
                                  */
     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 */

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

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

/* Display read data */
     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-byte transmit/receive processing  */
/*****************************************/
```

```
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;
}

/***********************************/
/*   Interval (ms) processing      */
/***********************************/
void wait_ms ( unsigned short work )
{
     s_timer =  work;
     while ( s_timer )
     {
          ;                          /* dummy statement */
     }
}

/***********************************/
/*   INTCSI2 interrupt processing  */
/***********************************/
#pragma interrupt INTCSI2 int_csi2
_ _interrupt
void int_csi2 (void)
{
     flag_csi = OFF;
}
/***************************************/
/*  INTM000 (1 ms) interrupt processing */
/***************************************/
#pragma interrupt INTM000 int_tm000
_ _interrupt
void int_tm000 (void)
{
     CCC00 += 1562;                 /* for 1ms interval */
     if (s_timer !=0 )
     {
          s_timer--;
     }
}
```

### (10) Detection of changes in optical sensor input

#### (a) Overview of function

Optical sensor input at a 100 ms interval undergoes A/D conversion and the result is displayed via the 7-segment LED. An alarm is sounded if the latest result is a value that is at least 3 greater or less than the previous result value.

#### (b) Program list

```
/************************************************/
/* Optical sensor test                          */
/* Optical sensor input at a 100 ms interval    */
/* undergoes A/D conversion                     */
/* <1> Result is displayed via 7-segment LED    */
/* <2> Alarm sounds if latest result is at least */
/* 2greater or less than the previous result    */
/************************************************/

#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 = 0.64 us) during 50 MHz operation */
     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 (1 ms) interrupt processing  */
/*************************************/
#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 interrupt processing    */
/**********************************/
#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) Multi-channel music output

#### (a) Overview of function

This program performs three-channel parallel processing to successively output music via a speaker while the music data is being read. The program controls output so that the sum of the three channels' output is output via a speaker that is connected to a D/A converter. Although each channel is set up to operate independently, this program delays the start of identical music data output to implement reproduction with echo effects.

#### (b) Processing in program

Each musical note consists of a "tone" and a "duration," and musical notes are arranged in a music table in the order in which they are played.

There are 36 (0 to 35) types of tone codes and tones can be expressed across three octaves. Tone code 255 indicates a musical rest and code 254 indicates the end of the music table.

As for the duration codes, 8 indicates a quarter note, and multiples of 8 are set to indicate longer notes.

A table format is set up to indicate the output inversion intervals used to generate a frequency for each tone and to set the inversion count (unit count) needed to output 32nd notes.

The output inversion intervals corresponding to tones are used to output tones while a compare register is being updated, and the data output to the D/A converter is inverted whenever a match interrupt occurs. A single note is played by repeating the count calculated based on duration code $\times$ unit count.

To ensure clear breaks between notes, this program inserts a musical rest (silent period) between the notes. Once output of one note is completed, ch_ma [n] is set to ON to generate a rest. Once output of the rest is completed, processing resumes for the next note.

Starting from the first note, the notes are played by being read and output individually, and the playing of the score ends when tone code 254 (end) is read.

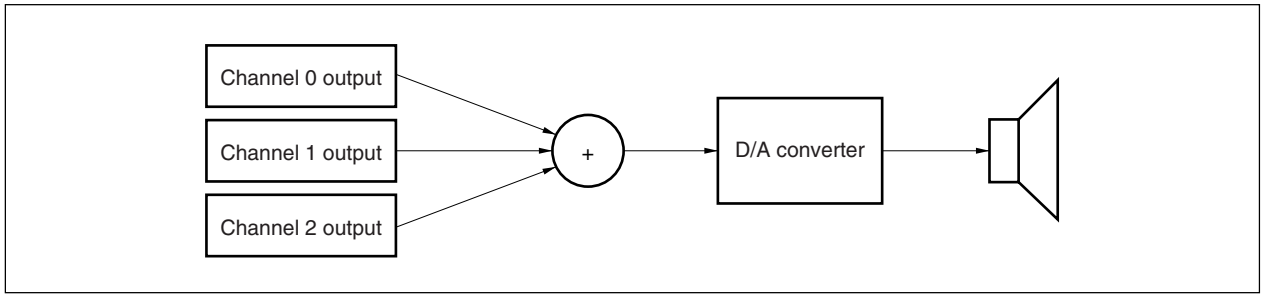
#### (c) Interrupts used

- INTM000: Interval timer interrupt that occurs to generate a tone via channel 0
- INTM001: Interval timer interrupt that occurs to generate a tone via channel 1
- INTM010: Interval timer interrupt that occurs to generate a tone via channel 2
- INTM011: 1 ms interval timer interrupt

**(d) Combination of multi-channel output**

The following method is used to combine multi-channel output for output via a single speaker.

**Figure 4-19. Combination of Multi-Channel Output**




The D/A converter has 256 levels (00H to 0FFH) of resolution and the amplitude of combined 3-channel output is confined to this range as well.

Output from each channel to the D/A converter oscillates between the positive and negative directions centered on 80H, as is shown in Figure 4-20 below. In the figure, output is based on amplitude of ±30H for channel 0, ±20H for channel 1, and ±10H for channel 2. Consequently, output operates in a range from 20H to E0H.

**Figure 4-20. Output to D/A Converter (1/2)**

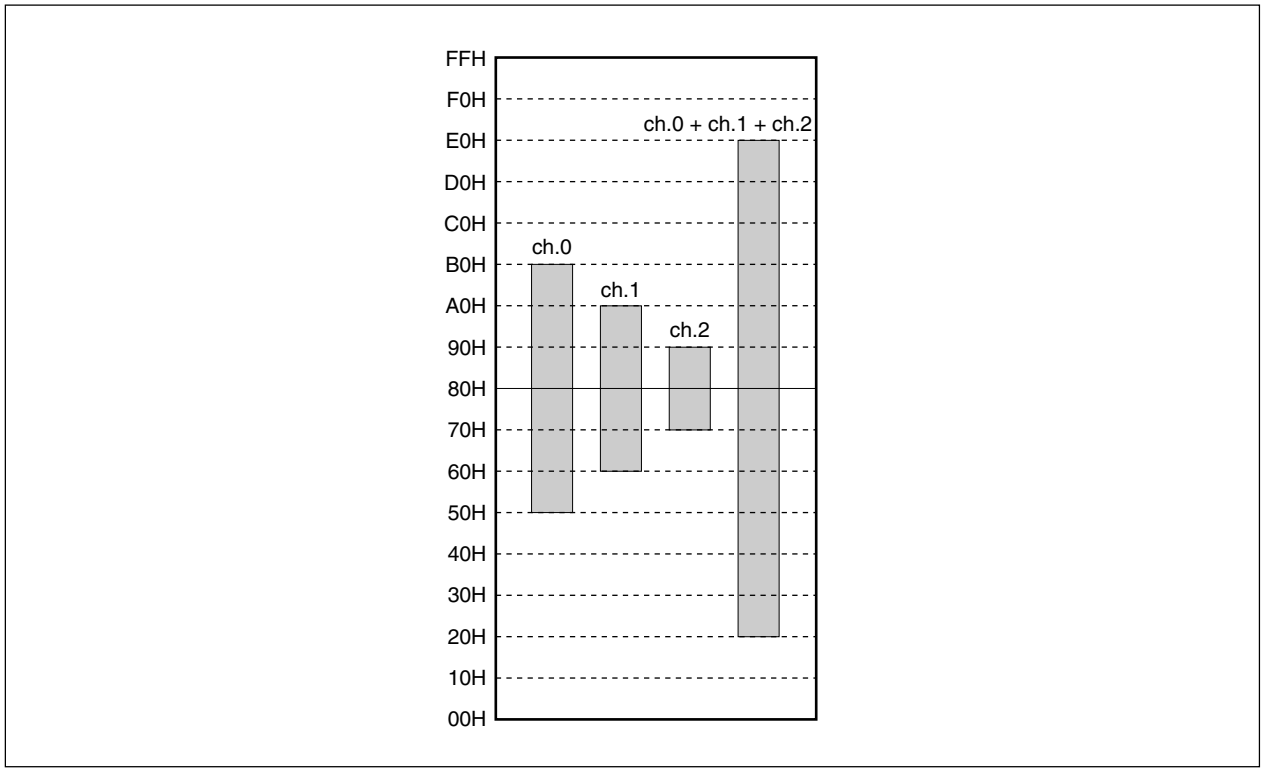
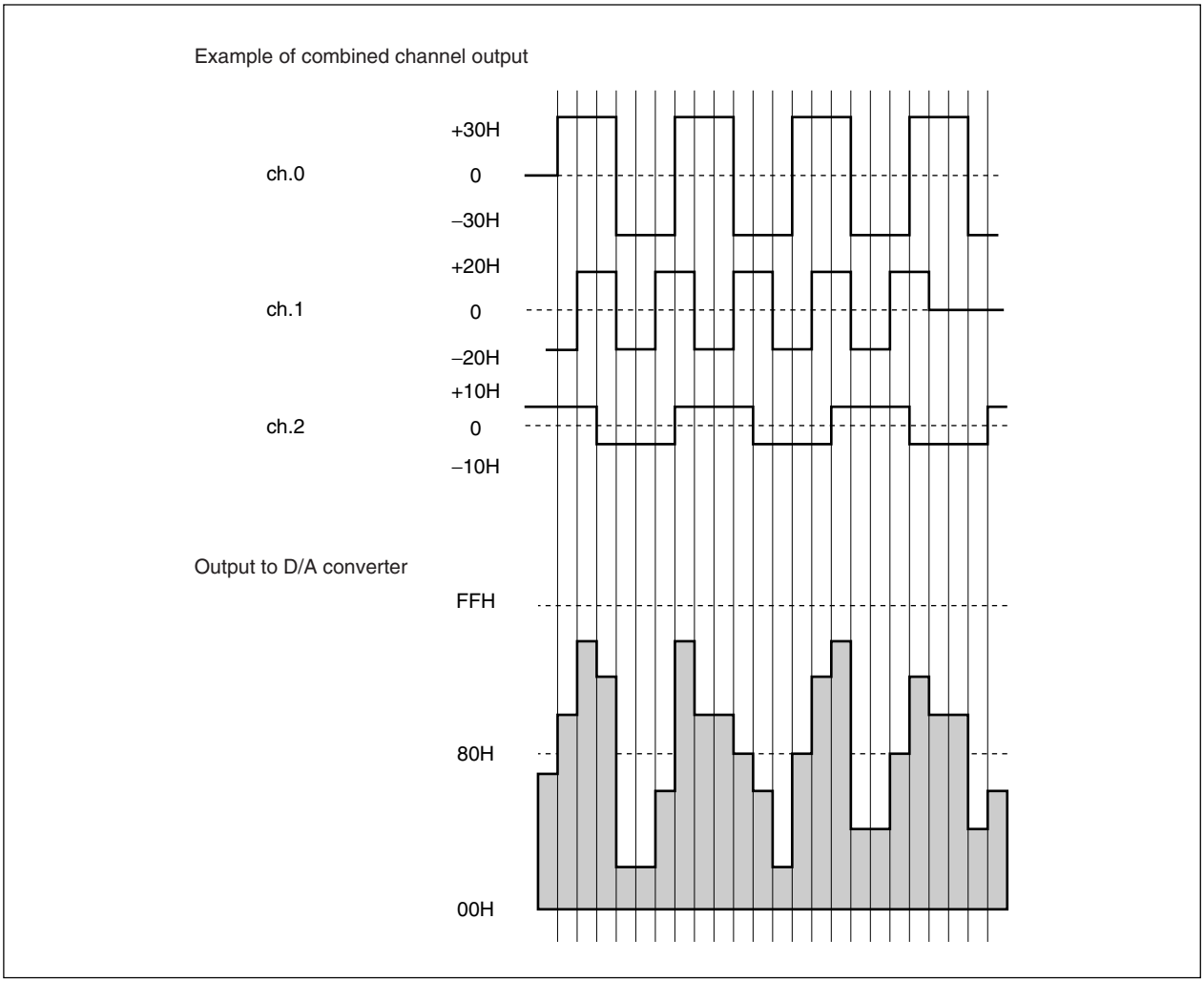


**Figure 4-20. Output to D/A Converter (2/2)**




**(e) music_start function**

This function assigns channel numbers and music data start addresses, then starts playing the music.

When music output is started by the music_start function, the interval timer interrupt used to generate tones for to the specified channels is set to enabled mode, and is set back to disabled mode when music output is finished. This mode can be checked to determine whether or not music is being played.

**(f) mix_da function**

Signed values that have been stored to the ch_val[0], ch_val[1], and ch_val[2] areas are combined and output to the D/A converter. The range of output values is 00H to FFH, centered on 80H.

**(g) Flowchart**

**Figure 4-21. Multi-Channel Music Output (Main Function)**

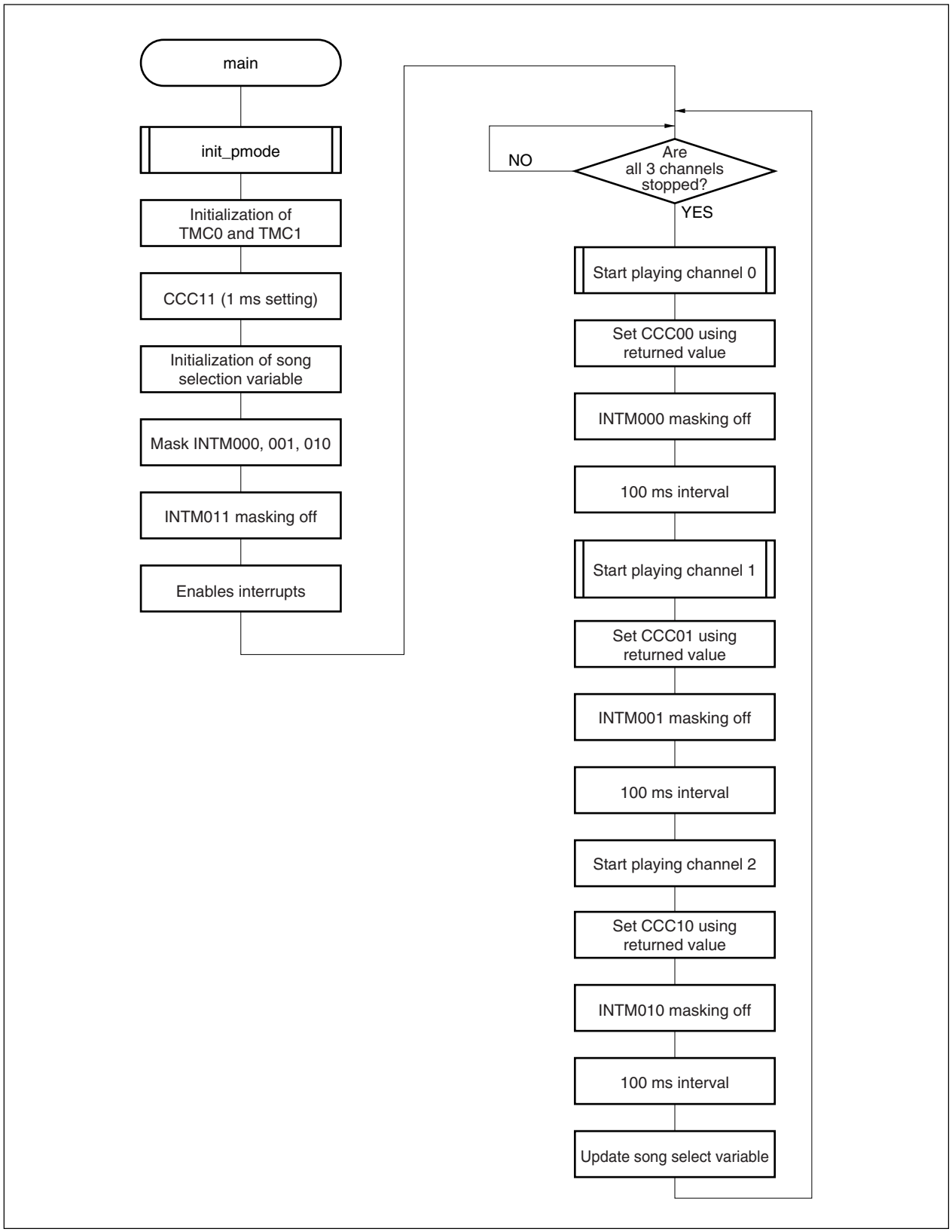
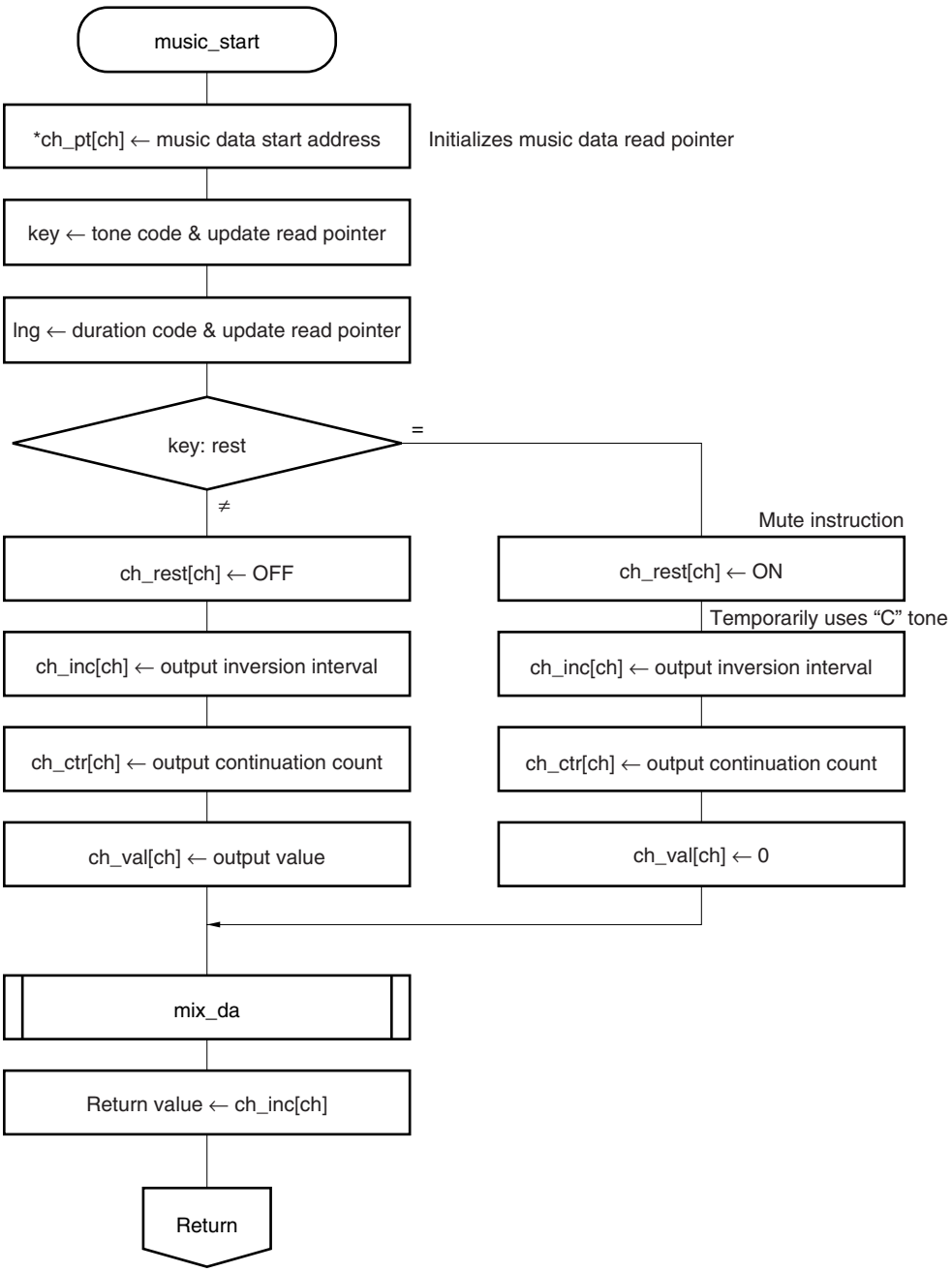


**Figure 4-22. Multi-Channel Music Output (Initialization of Music Data Read Pointer)**




**Remarks 1.** The output inversion interval and output continuation count are values corresponding to read tone codes that are fetched from freq_tabl[][] and determined.

**2.** Interrupts are used for the following.

- INTM000: Channel 0 tone interval
- INTM001: Channel 1 tone interval
- INTM010: Channel 2 tone interval
- INTM011: 1 ms interval

**Figure 4-23. Multi-Channel Music Output (Output to D/A Converter)**

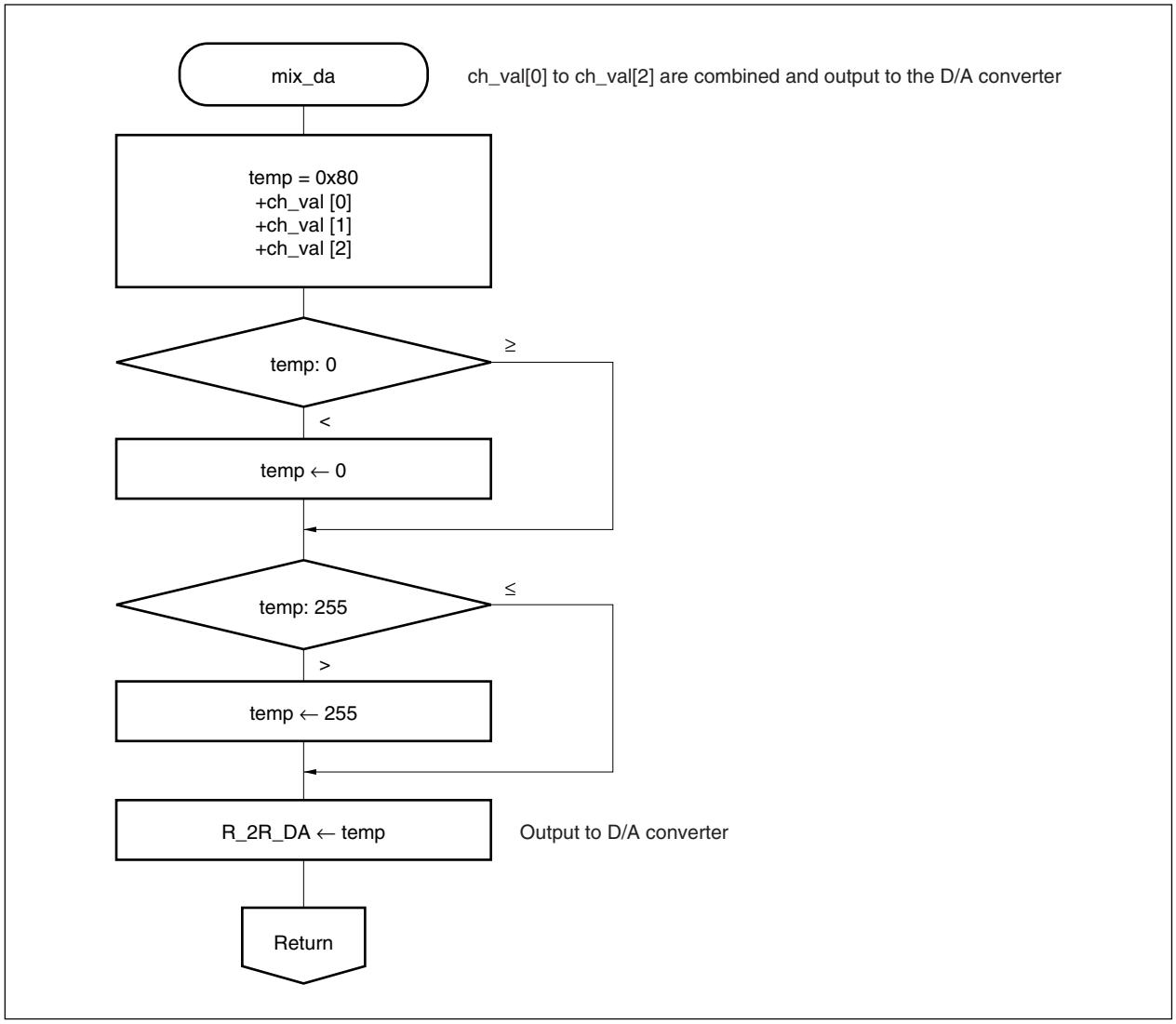


**Figure 4-24. Multi-Channel Music Output (Channel 0 Tone Interval Interrupt Processing)**

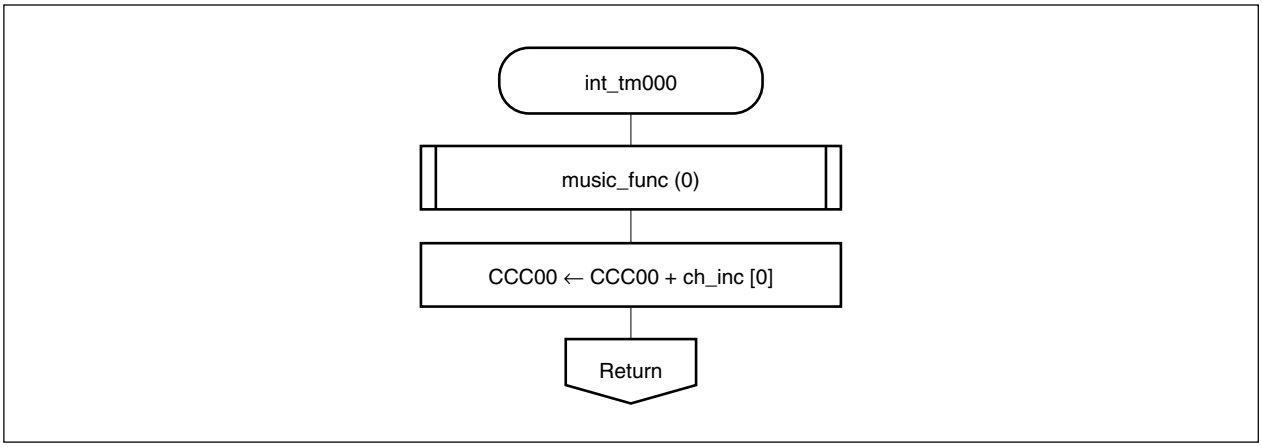



**Figure 4-25. Output of Multi-Channel Music (Channel 1 Tone Interval Interrupt Processing)**

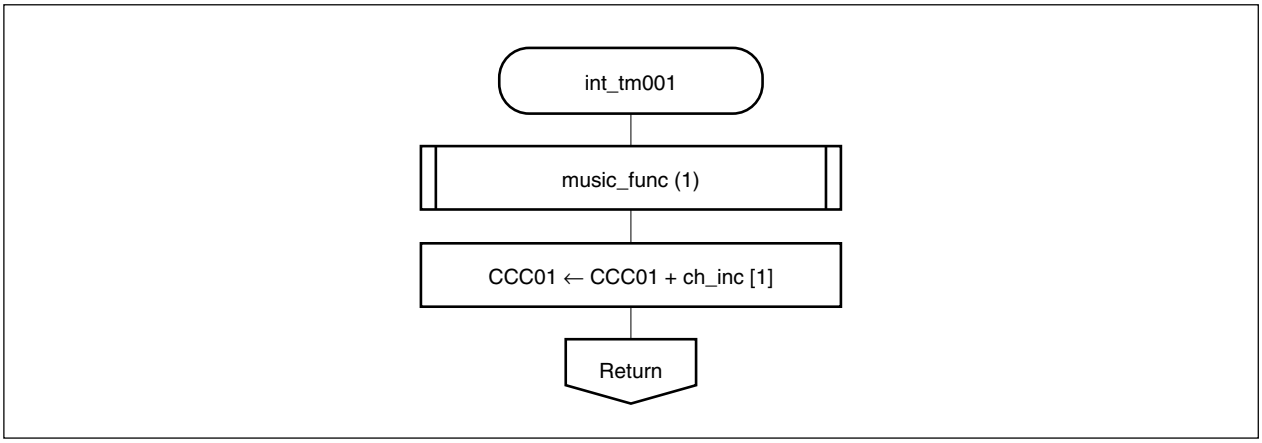



**Figure 4-26. Multi-Channel Music Output (Channel 2 Tone Interval Interrupt Processing)**

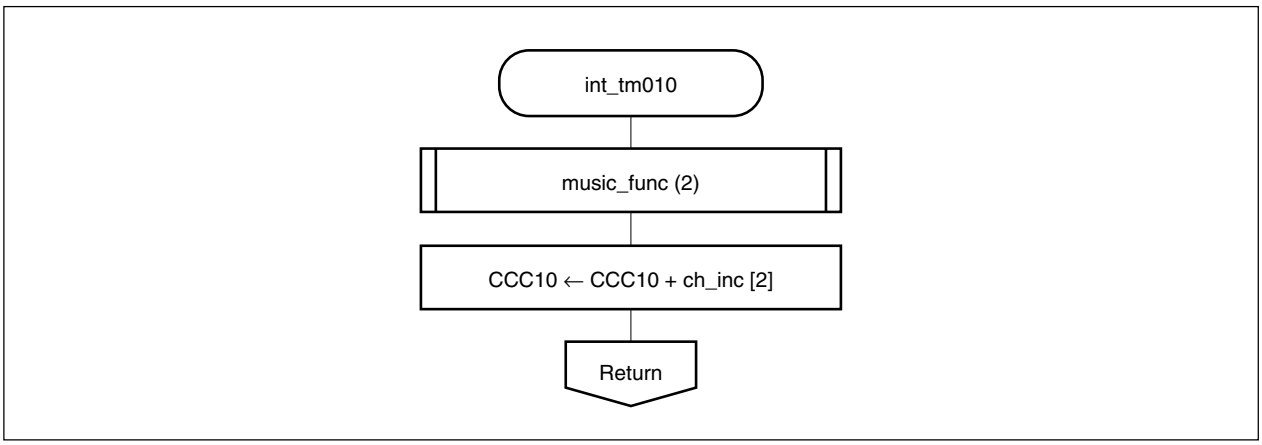


**Figure 4-27. Multi-Channel Music Output (1 ms Interval Interrupt Processing)**

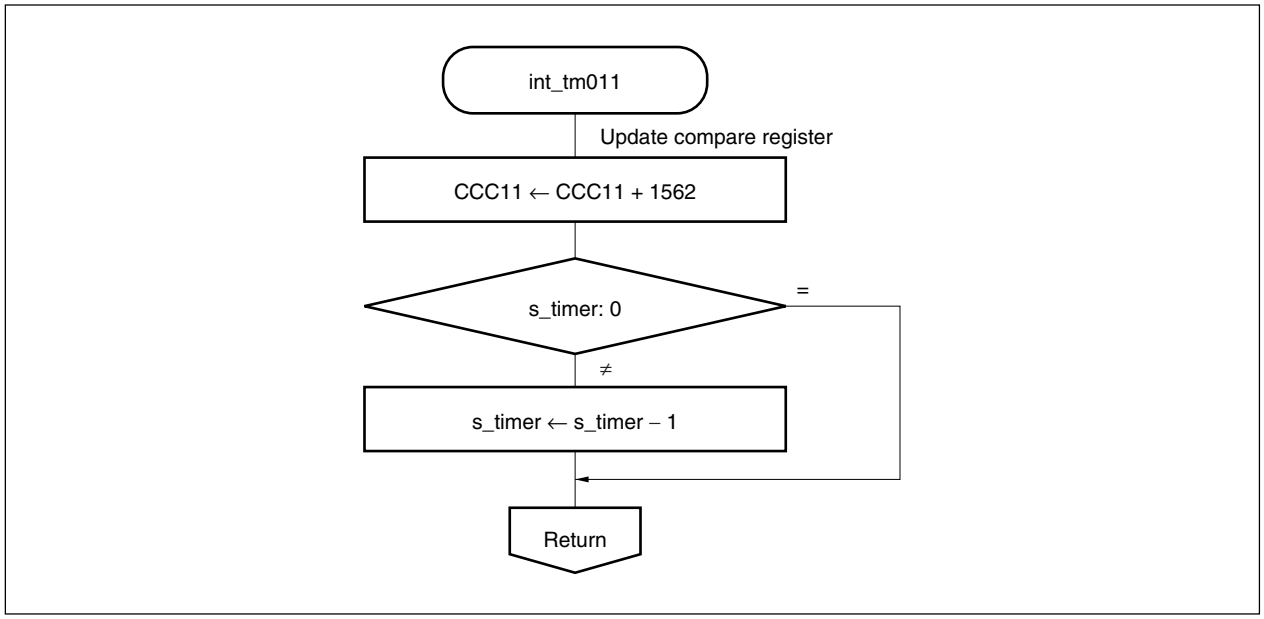


**Figure 4-28. Multi-Channel Music Output (Tone Interval Interrupt Common Processing) (1/2)**

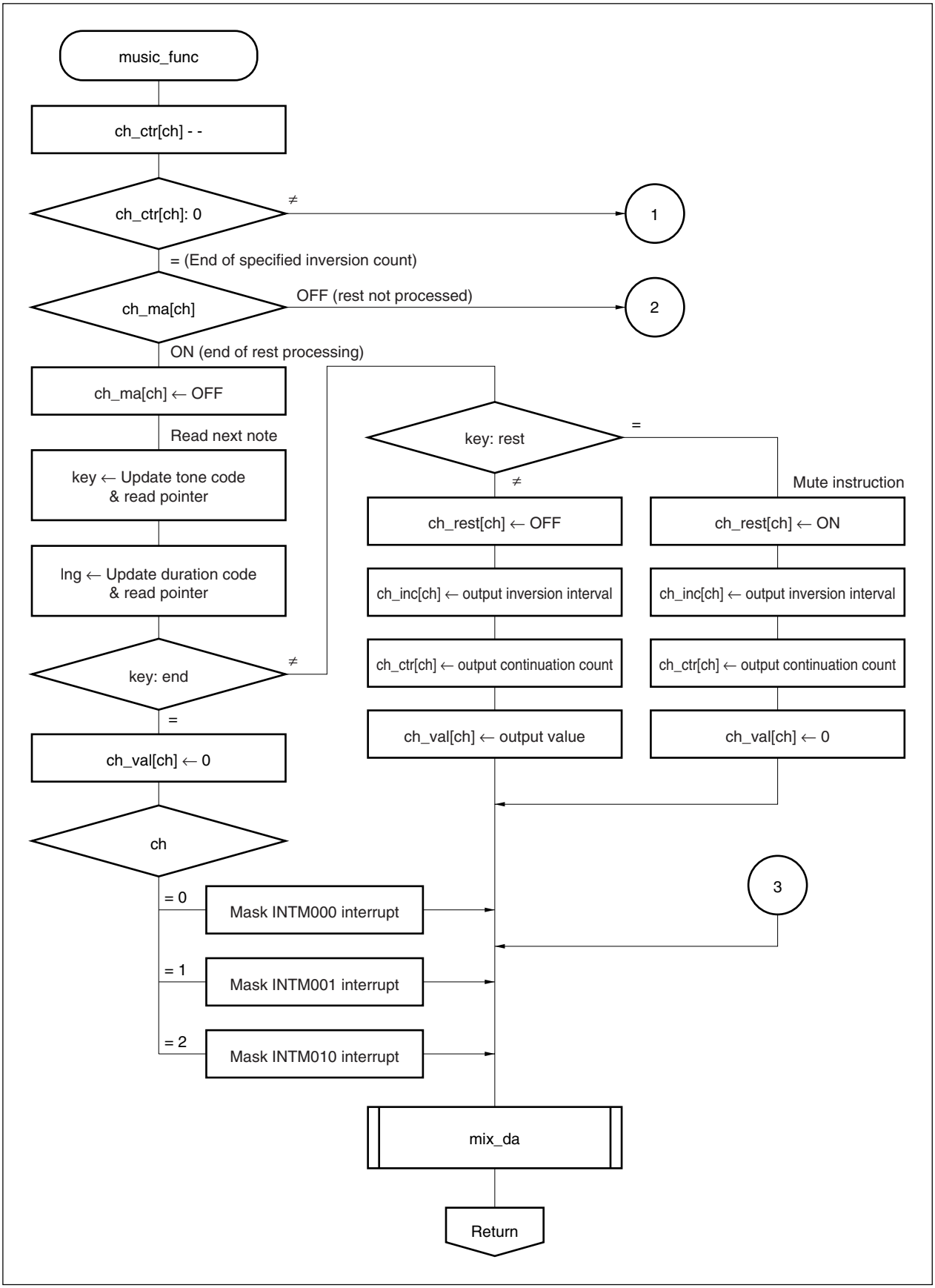
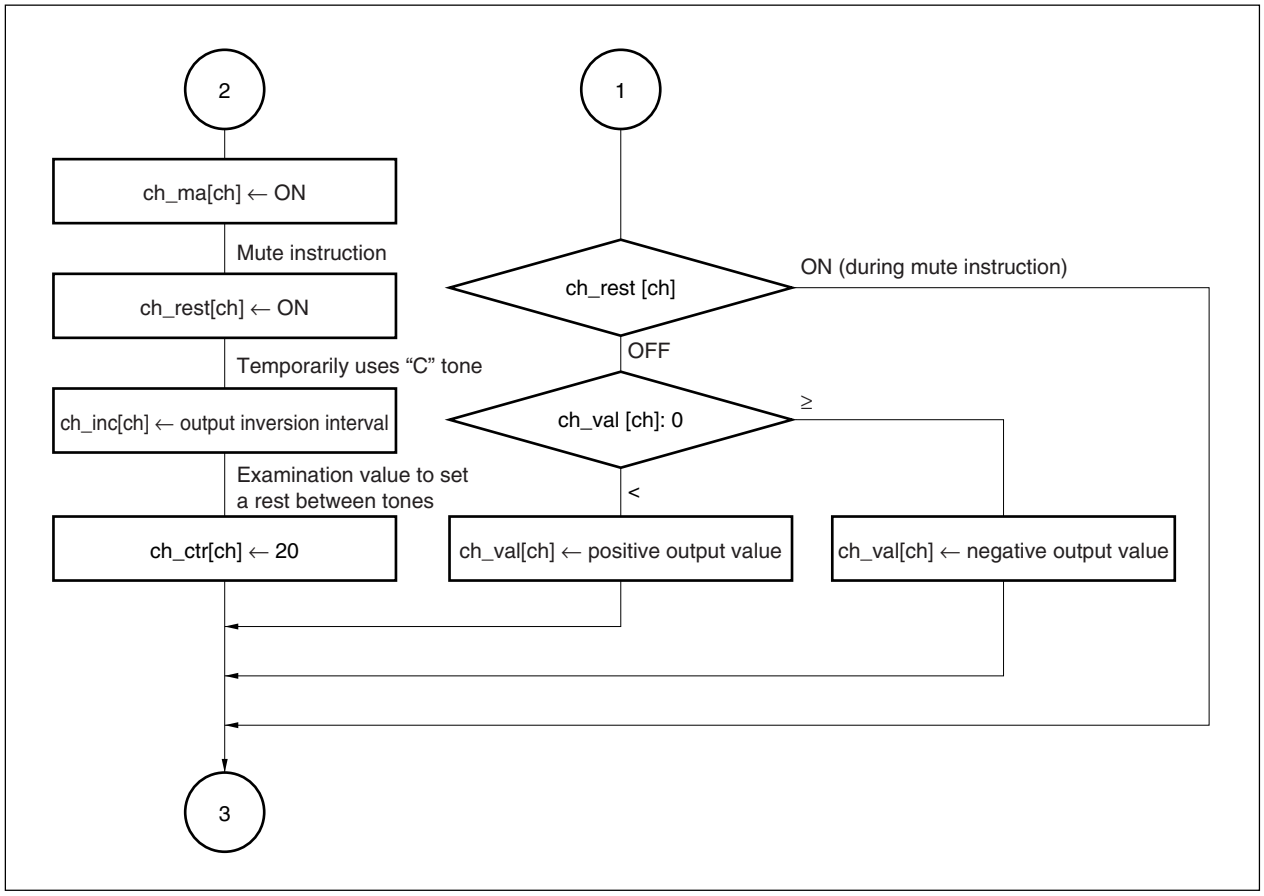


**Figure 4-28. Multi-Channel Music Output (Tone Interval Interrupt Common Processing) (2/2)**



### (h) Program list

```
/*******************************************/
/*                                         */
/*  Automatic playing of multi-channel music  */
/*                                         */
/*******************************************/

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

extern unsigned short      freq_tabl[36][2];     /* Generation tone frequency table */
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 = 0.64 us) during 50 MHz operation */
          TMCC01 = 0x03;         /* compare clear disable, CCC00,01 are compare */
          TMCC00.1 = 1;          /* TMC0 count enable */

          TMCC10.0 = 1;          /* TMC1 reset */
          TMCC10 |= 0x30;        /* TMC0 fxx/32 (t = 0.64 us) during 50 MHz operation */
          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;
}

/*************************************************************/
/*   Timer interrupt processing (for multi-channel music)    */
/*************************************************************/
#pragma interrupt INTM000 int_tm000
_ _interrupt
void int_tm000 (void)
{
     music_func (0);
     CCC00 += ch_inc[0]/2;        /* Updates compare register */
}

/* for music ch.1 */
#pragma interrupt INTP001 int_tm001
_ _interrupt
void int_tm001 (void)
{
     music_func (1);
     CCC01 += ch_inc[1]/2;        /* Updates compare register */
}

/* for music ch.2 */
#pragma interrupt INTM010 int_tm010
_ _interrupt
void int_tm010 (void)
{
     music_func (2);
     CCC10 += ch_inc[2]/2;        /* Updates compare register */
}

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

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

/************************************/
/* Timer interrupt common processing */
/************************************/

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 ();
}

/*************************************************/
/*   Tone data header file                       */
/*************************************************/
#define       SILENCE       0xff    /* Mute                */
#define       C_L           0       /* Low C               */
#define       Db_L          1       /*     C#/Db           */
#define       D_L           2       /*     D               */
#define       Eb_L          3       /*     D#/Eb           */
#define       E_L           4       /*     E               */
#define       F_L           5       /*     F               */
#define       Gb_L          6       /*     F#/Gb           */
#define       G_L           7       /*     G               */
#define       Ab_L          8       /*     G#/Ab           */
#define       A_L           9       /*     A               */
#define       Bb_L          10      /*     A#/Bb           */
#define       B_L           11      /*     B               */
#define       C_M           12      /* Middle C            */
#define       Db_M          13      /*     C#/Db           */
#define       D_M           14      /*     D               */
#define       Eb_M          15      /*     D#/Eb           */
#define       E_M           16      /*     E               */
```

```
#define       F_M           17     /*     F              */
#define       Gb_M          18     /*     F#/Gb          */
#define       G_M           19     /*     G              */
#define       Ab_M          20     /*     G#/Ab          */
#define       A_M           21     /*     A              */
#define       Bb_M          22     /*     A#/Bb          */
#define       B_M           23     /*     B              */
#define       C_H           24     /* High C             */
#define       Db_H          25     /*     C#/Db          */
#define       D_H           26     /*     D              */
#define       Eb_H          27     /*     D#/Eb          */
#define       E_H           28     /*     E              */
#define       F_H           29     /*     F              */
#define       Gb_H          30     /*     F#/Gb          */
#define       G_h           31     /*     G              */
#define       Ab_H          32     /*     G#/Ab          */
#define       A_H           33     /*     A              */
#define       Bb_H          34     /*     A#/Bb          */
#define       B_H           35     /*     B              */
#define       EndS          0xfe   /* End of Score       */

/***********************************************/
/*   Notes                                     */
/***********************************************/
#define       T32           1      /*  32nd note      */
#define       T16           2      /*  16th note      */
#define       T8            4      /*  8th note       */
#define       T4            8      /*  Quarter note   */
#define       T2            16     /*  Half note      */
#define       T1            32     /*  Whole note     */

/***********************************************/
/*   Rests                                     */
/***********************************************/
#define       R32           1      /*  32nd rest      */
#define       R16           2      /*  16th rest      */
#define       R8            4      /*  16th rest      */
#define       R4            8      /*  Quarter rest   */
```

```
#define       R2            16     /*  Half rest       */
#define       R1            32     /*  Whole rest      */
#define _FF 255
#define _PP 80

/**** Constant definition file ****/
#include "music_def.h"

/*************************************/
/*                                   */
/*   Generation tone frequency table    */
/*                                   */
/*************************************/
unsigned short                    freq_tabl[36][2]=
{
      2986, 49,                   /*  0:  Low C (invalid)  */
      2819, 52,                   /*  1:  C#/Db            */
      2660, 55,                   /*  2:  D                */
      2519, 58,                   /*  3:  D#/Eb            */
      2370, 62,                   /*  4:  E                */
      2237, 65,                   /*  5:  F                */
      2111, 69,                   /*  6:  F#/Gb            */
      1993, 74,                   /*  7:  G                */
      1881, 78,                   /*  8:  G#/Ab            */
      1776, 83,                   /*  9:  A                */
      1676, 87,                   /* 10:  A#/Bb            */
      1582, 93,                   /* 11:  B                */

      1493, 98,                   /* 12:  Middle C         */
      1409, 104                   /* 13:  C#/Db            */
      1330, 110,                  /* 14:  D                */
      1260, 116,                  /* 15:  D#/Eb            */
      1185, 124,                  /* 16:  E                */
      1119, 131,                  /* 17:  F                */
      1056, 139,                  /* 18:  F#/Gb            */
      996,  147,                  /* 19:  G                */
      941,  156,                  /* 20:  G#/Ab            */
      888,  165,                  /* 21:  A                */
      838,  175,                  /* 22:  A#/Bb            */
      791,  185,                  /* 23:  B                */

      747,  196,                  /* 24:  High C           */
      705,  208,                  /* 25:  C#/Db            */
```

```
      665,  220,                  /* 26:  D            */
      630,  233,                  /* 27:  D#/Eb        */
      593,  247,                  /* 28:  E            */
      559,  262,                  /* 29:  F            */
      528,  278,                  /* 30:  F#/Gb        */
      498,  294,                  /* 31:  G            */
      470,  311,                  /* 32:  G#/Ab        */
      444,  330,                  /* 33:  A            */
      419,  350,                  /* 34:  A#/Bb        */
      395,  370                   /* 35:  B            */
};

/*************************************/
/*                                   */
/*   Music (0):  Sakura              */
/*                                   */
/*************************************/
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 }
     };

/*****************************************/
/*                                       */
/*   Data for other seven songs is omitted. */
/*                                       */
/*****************************************/
```

```
/*************************************/
/*                                   */
/*   Song table                      */
/*                                   */
/*************************************/
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]    } ;

/**** Common files (including initialization block)****/
#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

/*************************************/
/*   Numeric display function (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];
}
```

```
/*****************************************/
/*   Hexadecimal display function (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) Proportional/integral control of DC motor speed

#### (a) Overview of function

This function controls the speed of the DC motor that is connected to a PWM unit.
The rotation speed (rpm) is controlled according to the duty factor of pulses output from the PWM unit.
For each rotation, 12 encoder pulses are output from the encoder attached to the DC motor. The current rotation speed can be checked by monitoring these encoder pulses.
The input voltage, for which a volume adjustment is available, is read by the A/D converter and the target rotation speed is determined using the read value.
The current rotation speed and target rotation speed are both monitored, and PWM control is performed to track the target rotation speed.
Tracking control is achieved through a combination of proportional and integral control methods.

#### (b) Program configuration

- Main function
  Initialization of peripheral I/O (timer unit, PWM unit, A/D converter)
  Initialization of global variables and starting of A/D converter
  An endless loop can occur due to the following processing.
    When A/D conversion is completed and conversion result is used to confirm target speed
    When A/D conversion is restarted

- INTM000 interrupt handler (interrupts at 10 ms interval)
  PWM output is based on proportional/integral control using current speed and target speed.
  Current speed and target speed are displayed via 7-segment LEDs.
  Differential between current speed and target speed is displayed via LEDs.

- INTP000 interrupt handler (encoder pulse interrupts)
  Adds “1” to the encoder pulse count.

- INTAD interrupt handler
  Stores conversion results to global variables.
  Sets A/D conversion completion flag.

**(c) Flowchart**

**Figure 4-29. Proportional/Integral Control of DC Motor Speed (Main Function)**

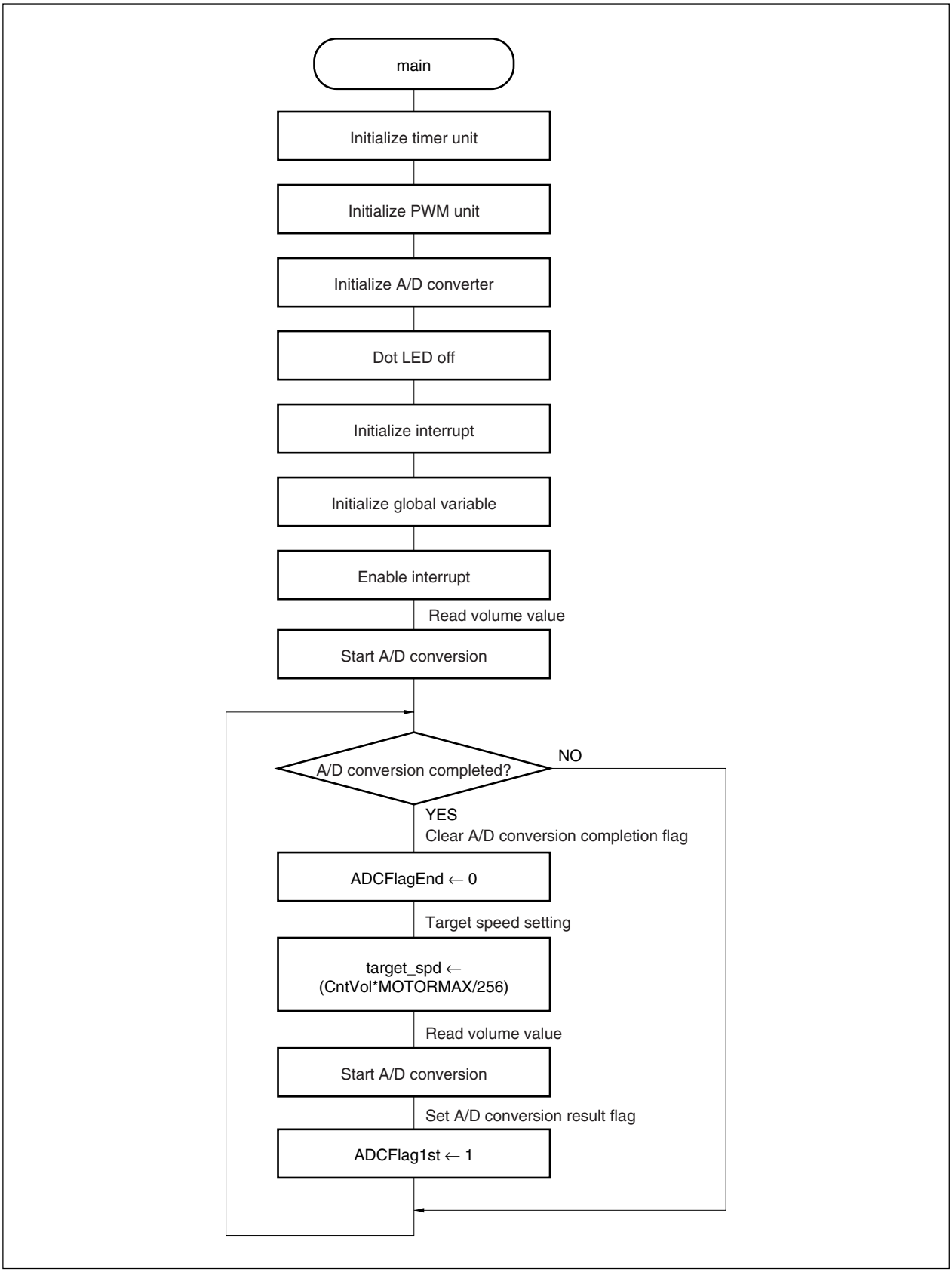


**Figure 4-30. Proportional/Integral Control of DC Motor Speed (INTP000 Interrupt Handler)**

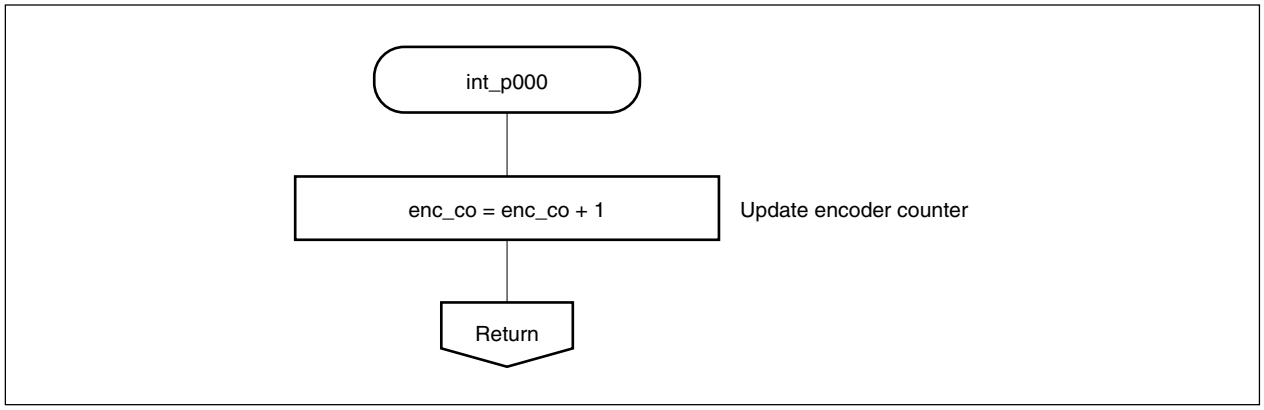



**Figure 4-31. Proportional/Integral Control of DC Motor Speed (INTAD Interrupt Handler)**

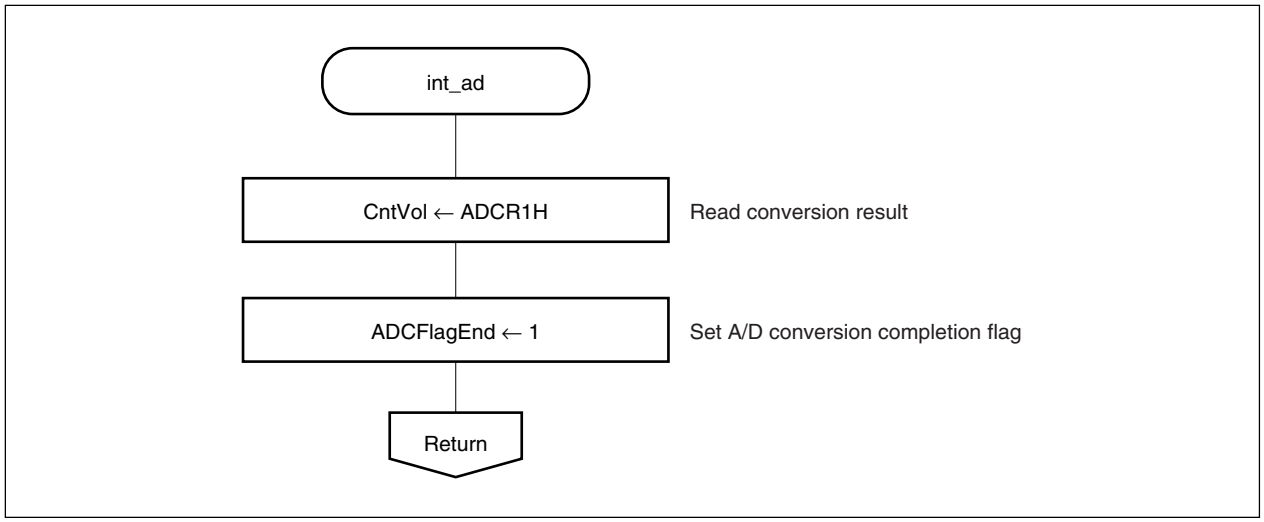


**Figure 4-32. Proportional/Integral Control of DC Motor Speed (INTM000 Interrupt Handler)**

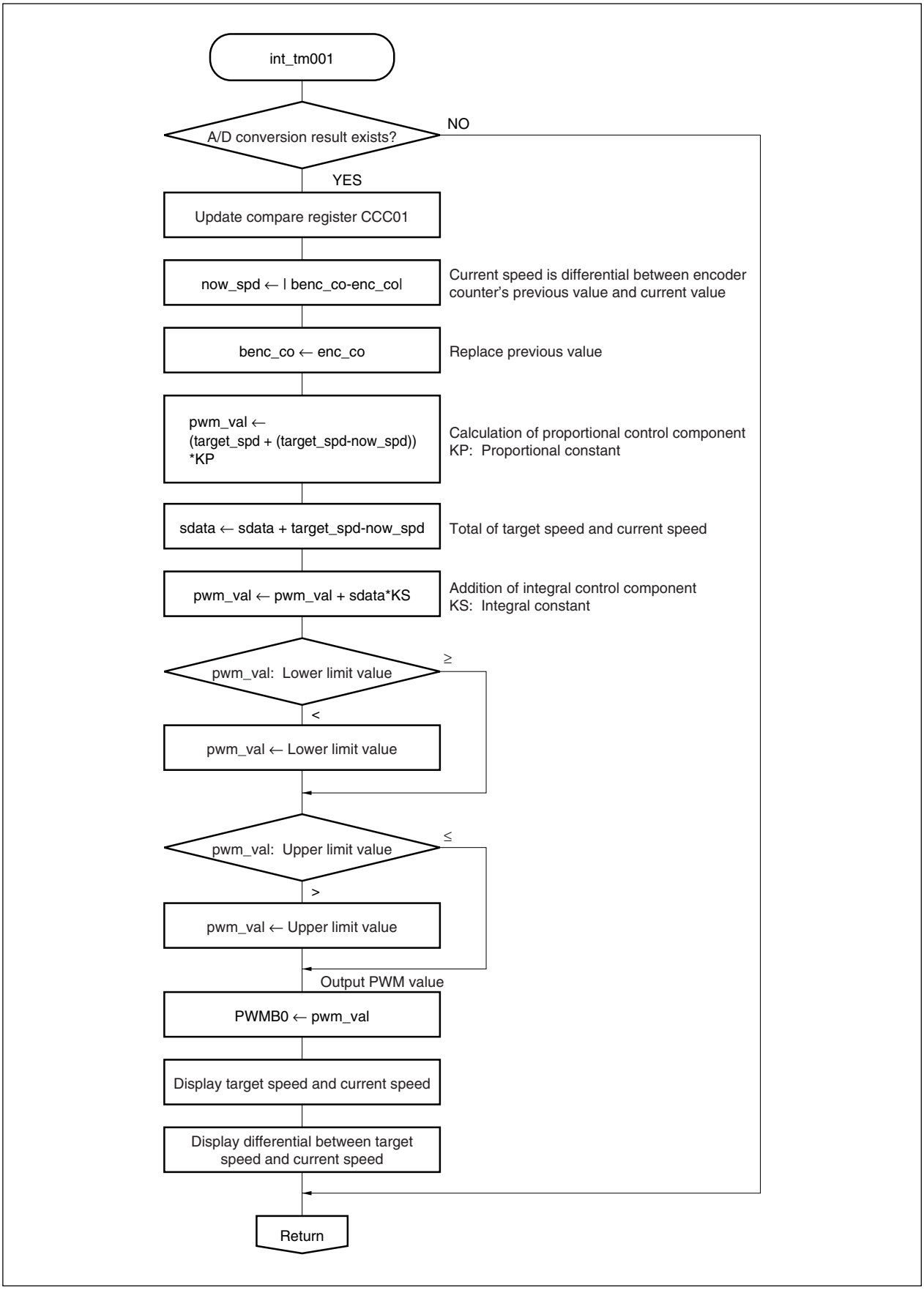
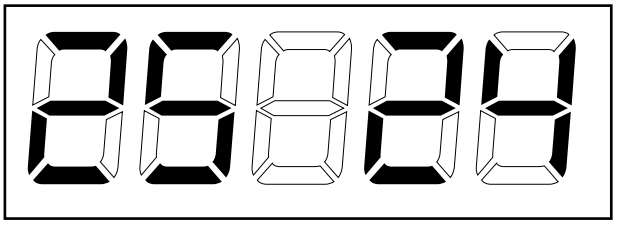


**Figure 4-33. 7-Segment LED**



Target speed Current speed

**Figure 4-34. Dot LED (Speed Differential Display)**

| Display | Current speed vs. target speed |
|---|---|
|  | –4 or greater (slow) |
|  | –3 |
|  | –2 |
|  | –1 |
|  | Target speed = current speed |
|  | +1 |
|  | +2 |
|  | +3 or more (fast) |

### (d) Program list

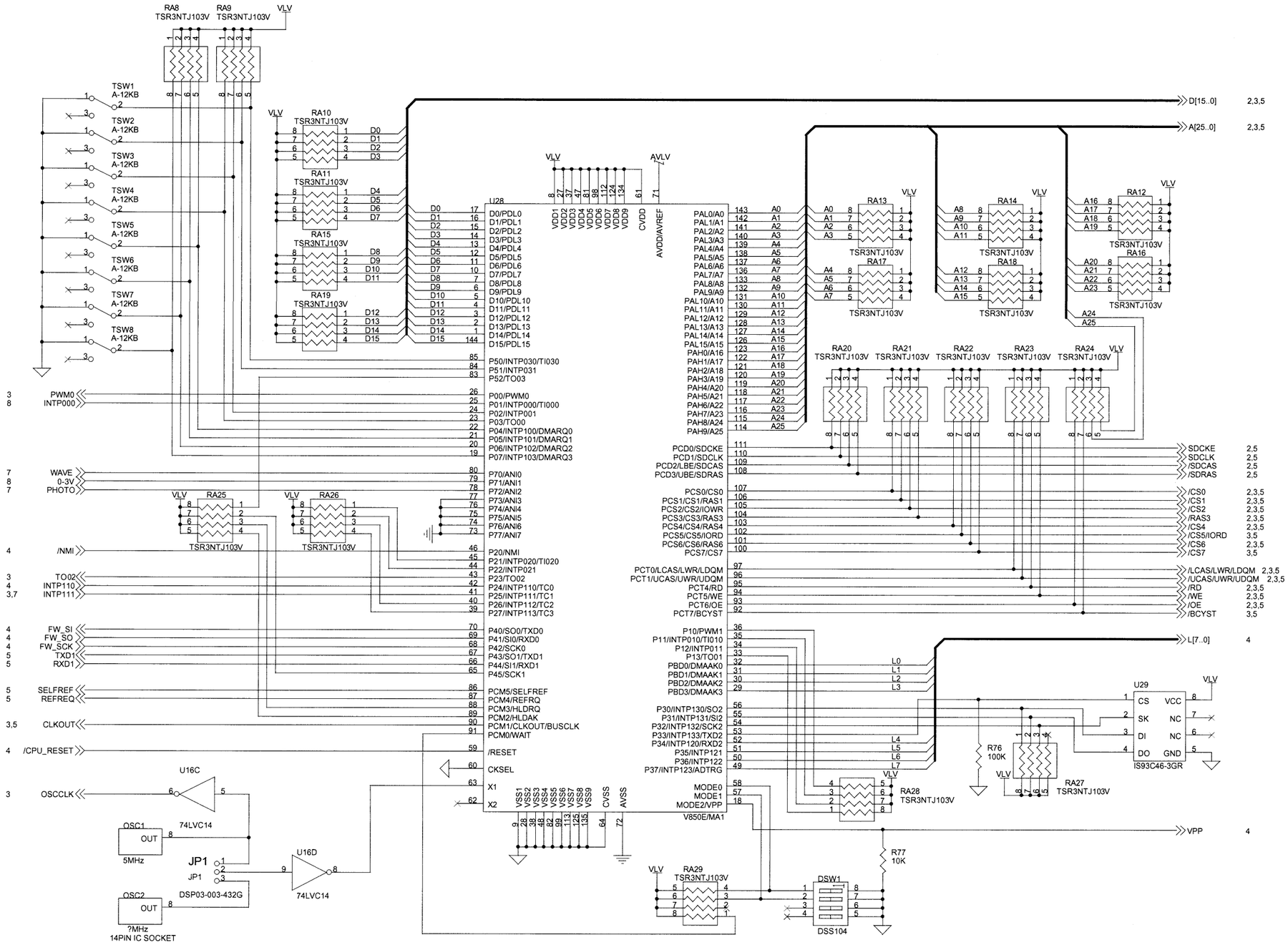
```
/*****************************************************************
      Proportional/integral control of DC motor
When the volume control is turned, the DC motor's target speed is
displayed in the fifth and fourth digits of the 7-segment LED.
Encoder pulses are sampled at an interval of 10 ms and proportional
and integral feedback control is used to maintain the target speed.
The current speed is displayed in the second and first digits of the
7-segment LED.
The differential between the current speed and target speed is
displayed via the Dot LED indicator.

The DC motor is controlled by the V850E/MA1's on-chip PWM output.
The volume control is connected to the V850E/MA1's on-chip A/D
converter.
Encoder pulses are output from the DC motor and these pulses trigger
INTP000 interrupts.
12 encoder pulses are output per rotation.
*****************************************************************/
#include      "common.h"

/* Constant definitions */
#defineMOTORMAX      73     /* Motor's maximum rated values    */
#defineLOWERLIMIT    0      /* Lower limit value of PWM output */
#defineUPPERLIMIT    255    /* Upper limit value of PWM output */
#defineKP            2.7    /* Proportional constant           */
#defineKS            0.5    /* Integral constant               */

/* Function prototype declaration */
void          int_p000(void);
void          int_tm001(void);
void          int_ad(void);

/* Global variable definition */
short         CntVol;       /* Volume value for speed specification */
int           ADCFlagEnd;   /* A/D conversion completion flag       */
int           ADCFlag1st;   /* A/D conversion initial flag          */
unsigned int  enc_co;       /* Encoder counter                      */
unsigned int  benc_co;      /* Previous encoder counter             */
short         target_spd;   /* Target speed                         */
int           sdata;        /* Integral value                       */
```

```
/*********************************/
/*            main               */
/*********************************/
void main(void)
{
      /* Initialization of timer C0 unit */
      TMCC00.0 = 1;         /* Operation enabled:  TMCCAE0 = 1                 */
      TMCC00 |= 0x50;       /* Operating clock = fxx/128 (3.2 us), during 40 MHz */
                            /* operation                                       */
      TMCC01.1 = 1;         /* CCC01 compare mode:  CMS01 = 1                  */
      CCC01 = 39060;        /* Compare value setting 10 ms = 3.2 us * 39060    */
      TMCC00.1 = 1;         /* Count start:  TMCCE0 = 1                        */

      /* Initialization of PWM unit */
      PWMB0 = 0;            /* Output low                                      */
                            /* Active high, counter = 8-bit length             */
      PWMC0 = 0x43;         /* Operating clock = fxx/16 (0.4 us), during 40 MHz  */
                            /* operation                                       */
      PWMC0 |= 0x80;        /* Operation enabled                               */

      /* Initialization of A/D converter unit */
      ADM2  = 0x00;         /* Reset                                           */
      ADM2 |= 0x01;         /* Operation enabled                               */
      ADM0  = 0x11;         /* ANI1 pin specification, select, 1 buffer        */
      ADM1  = 0x03;         /* A/D trigger, conversion time = 6.0 us, 40 MHz   */
                            /* operation                                       */

      P3 = 0x00;            /* Higher 4 bits of LED off                        */
      PBD = 0x00;           /* Lower 4 bits of LED off                         */

      /* Initialization of interrupts   */
      SESC0=0x01;           /* Valid edge of INTP000 is rising edge                */
      P00MK0=0;             /* INTP000:  Clears masking of encoder pulse interrupts */
      P00MK1=0;             /* INTM001:  Clears masking of timer C0 interrupts     */
      ADMK=0;               /* INTAD:  Clears masking of A/D conversion completion  */
                            /* interrupts                                          */

      /* Initialization of global variables */
      CntVol          = 0;  /* Clears speed specification volume value             */
      enc_co          = 0;  /* Initializes encoder accumulative counter            */
      benc_co         = 0;  /* Clears previous encoder counter                     */
      target_spd      = 0;  /* Clears target speed                                 */
      sdata           = 0;  /* Clears integral value                               */
      ADCFlagEnd      = 0;  /* Clears A/D conversion completion flag               */
      ADCFlag1st      = 0;  /* Clears A/D conversion initial flag                  */

      _ _EI();              /* Interrupts enabled                                  */
```

```
      ADM0 |= 0x80;         /* Start A/D conversion:  CE = 1                    */

       while(1) {
           if ( ADCFlagEnd ) {                          /* End of A/D conversion    */
                                                        /* value?                   */
                 ADCFlagEnd = 0;                        /* Clears A/D conversion    */
                                                        /* completion flag          */
                 target_spd = ( CntVol * MOTORMAX / 256 );/* Sets target speed      */
                 ADM0 |= 0x80;                          /* Starts A/D conversion    */
                 ADCFlag1st = 1;                        /* Sets A/D conversion      */
                                                        /* initial flag             */
           }
       }
}

/****************************************/
/* INTTM001 (10 ms) interrupt processing */
/****************************************/
#pragma interrupt INTM001 int_tm001
_ _interrupt
void    int_tm001(void)
{
      int now_spd;           /* Current speed                          */
      int pwm_val;           /* PWM output value                       */
      int diff;              /* Speed differential                     */
      unsigned char led;     /* Dot LED display buffer                 */

      if( ADCFlag1st )       /* A/D conversion started more than once? */
      {
            /* Proportional/integral control of DC motor */
            CCC01 += 39060;                     /* Resets 10 ms timer compare value */
            now_spd = abs( benc_co - enc_co );  /* Calculates current speed         */
            benc_co = enc_co;                   /* Updates previous encoder value   */
            pwm_val = ( target_spd + ( target_spd - now_spd ) ) * KP;/* Calculates  */
                                                /* proportional component           */
            sdata += ( target_spd - now_spd );  /* Total speed                      */
            pwm_val += ( sdata * KS );          /* Calculates integral component    */
            if( pwm_val < LOWERLIMIT )pwm_val = LOWERLIMIT;/* Lower limit PWM value */
            if( pwm_val > UPPERLIMIT )pwm_val = UPPERLIMIT;/* Upper limit PWM value */
            PWMB0 = pwm_val;                    /* Outputs PWM value                */

            /* Numerical display of target speed and current speed via 7-segment LED */
            disp_num(((unsigned int)target_spd * 1000) + (unsigned char)now_spd);
            LED_DGT3 = 0;    /* Sets empty space to third digit of 7-segment LED     */
```

```
            /* Displays speed control mode via dot LED */
            diff = now_spd - target_spd; /* Calculates differential to target speed */
            if(diff>=0)
            { /* When faster than target speed */
            switch(diff)
            {/* Down to target speed */
                  case 0  :  led =0x10; break;/* Match with target speed        */
                  case 1  :  led =0x30; break;/* Faster by one encoder pulse    */
                  case 2  :  led =0x50; break;/* Faster by two encoder pulses   */
                  default :  led =0x90; break;/* Faster by three or more        */
                                              /* encoder pulses                 */
            }
      }else
      { /* When slower than target speed */
            switch(abs(diff))
            { /* Up to target speed */
                  case 1  :  led =0x18; break;/* Slower by one encoder pulse     */
                  case 2  :  led =0x14; break;/* Slower by two encoder pulses    */
                  case 3  :  led =0x12; break;/* Slower by three or more encoder */
                                              /* pulses                          */
                  default :  led =0x11; break;/* Slower by four or more encoder  */
                                              /* pulses                          */
            }
      }
      P3 = led & 0xf0;        /* Higher 4 bits of LED lit */
      PBD= led & 0x0f;        /* Lower 4 bits of LED lit  */
      }
}
/*******************************/
/*  INP000 interrupt processing */
/*******************************/
#pragma interrupt INTP000 int_p000
_ _interrupt
void    int_p000(void)
{
       enc_co++;              /* Counts encoder total */
}
/*******************************/
/*  INTAD interrupt processing  */
/*******************************/
#pragma interrupt INTAD int_ad
_ _interrupt
void    int_ad(void)
{
       CntVol = ADCR1H;       /* Acquires A/D conversion value       */
       ADCFlagEnd = 1;        /* Sets A/D conversion completion flag */
}
```

**Figure 4-35. TU-V850E/MA1 (1/8)**



★

**Figure 4-35. TU-V850E/MA1 (2/8)**

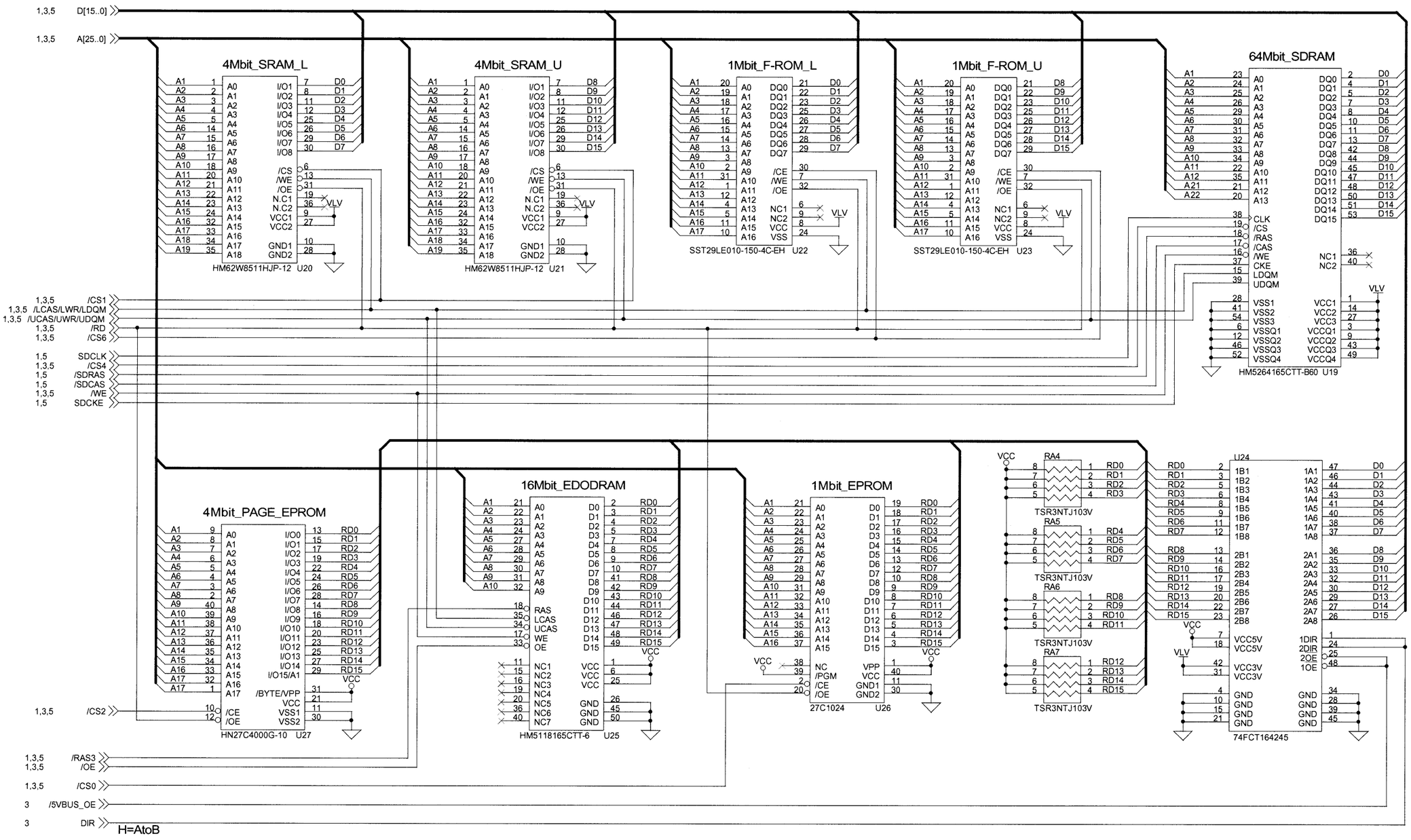


**Figure 4-35. TU-V850E/MA1 (3/8)** ★

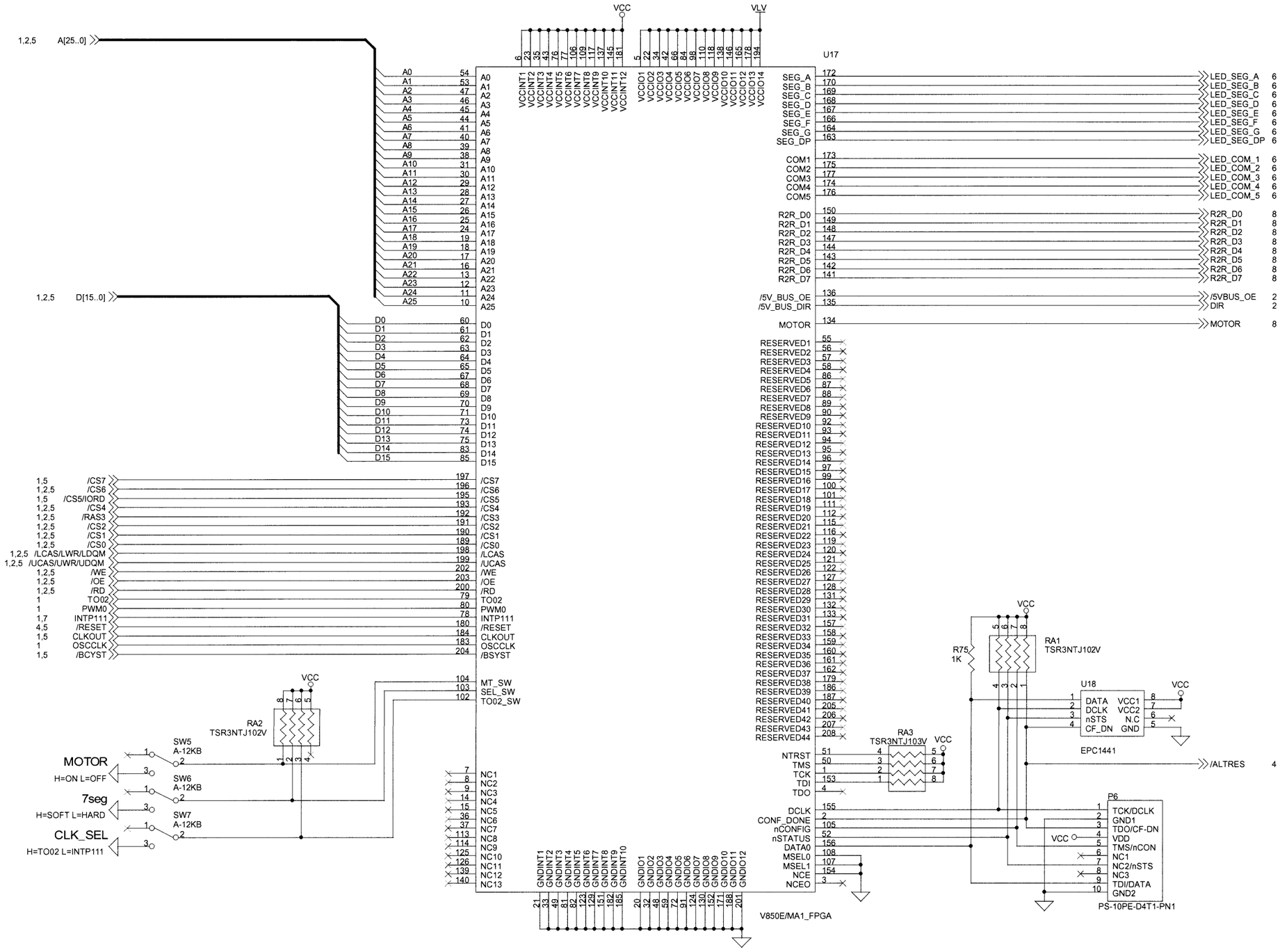
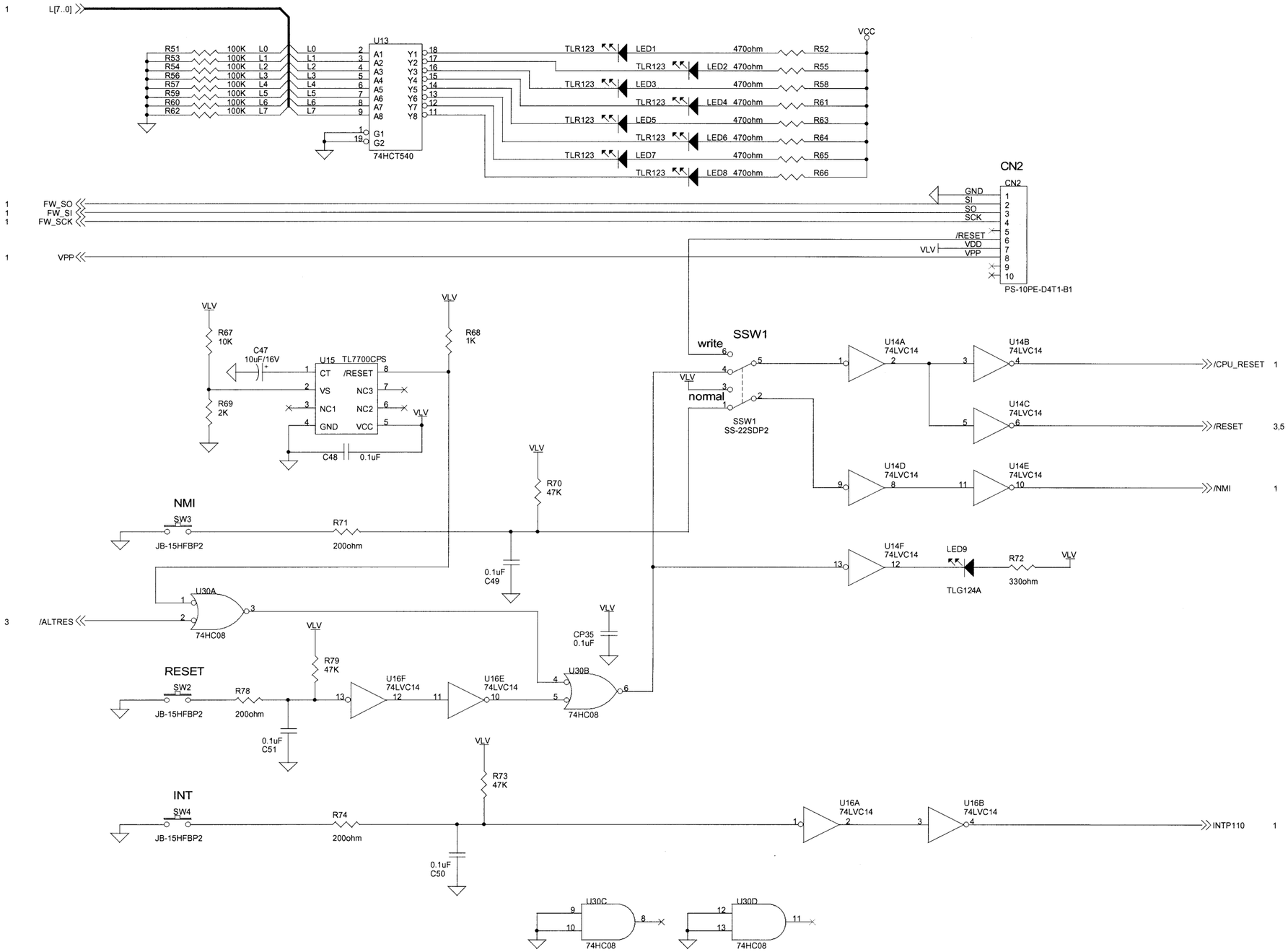


**Figure 4-35. TU-V850E/MA1 (4/8)**



★
**Figure 4-35. TU-V850E/MA1 (5/8)**

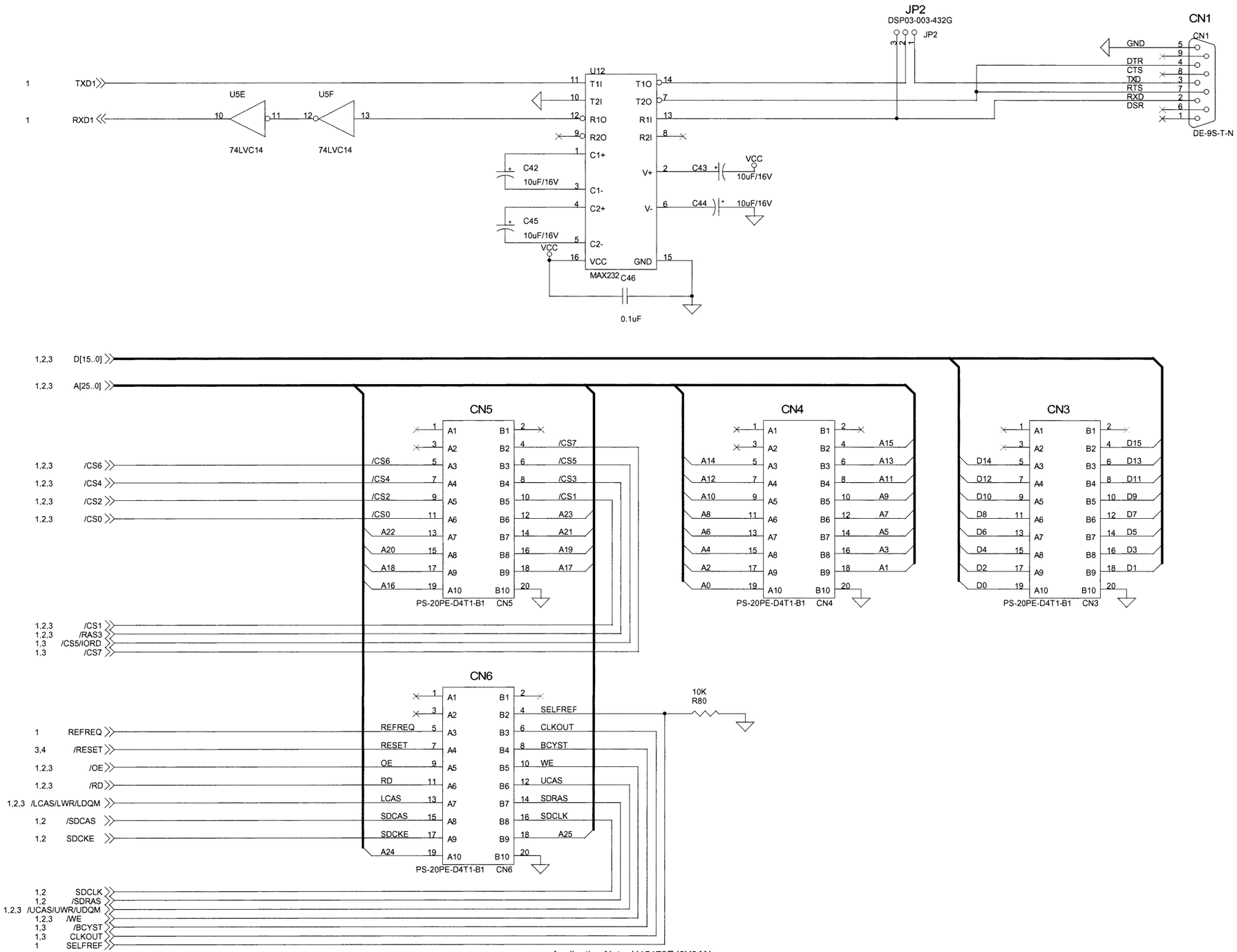


**Figure 4-35. TU-V850E/MA1 (6/8)**

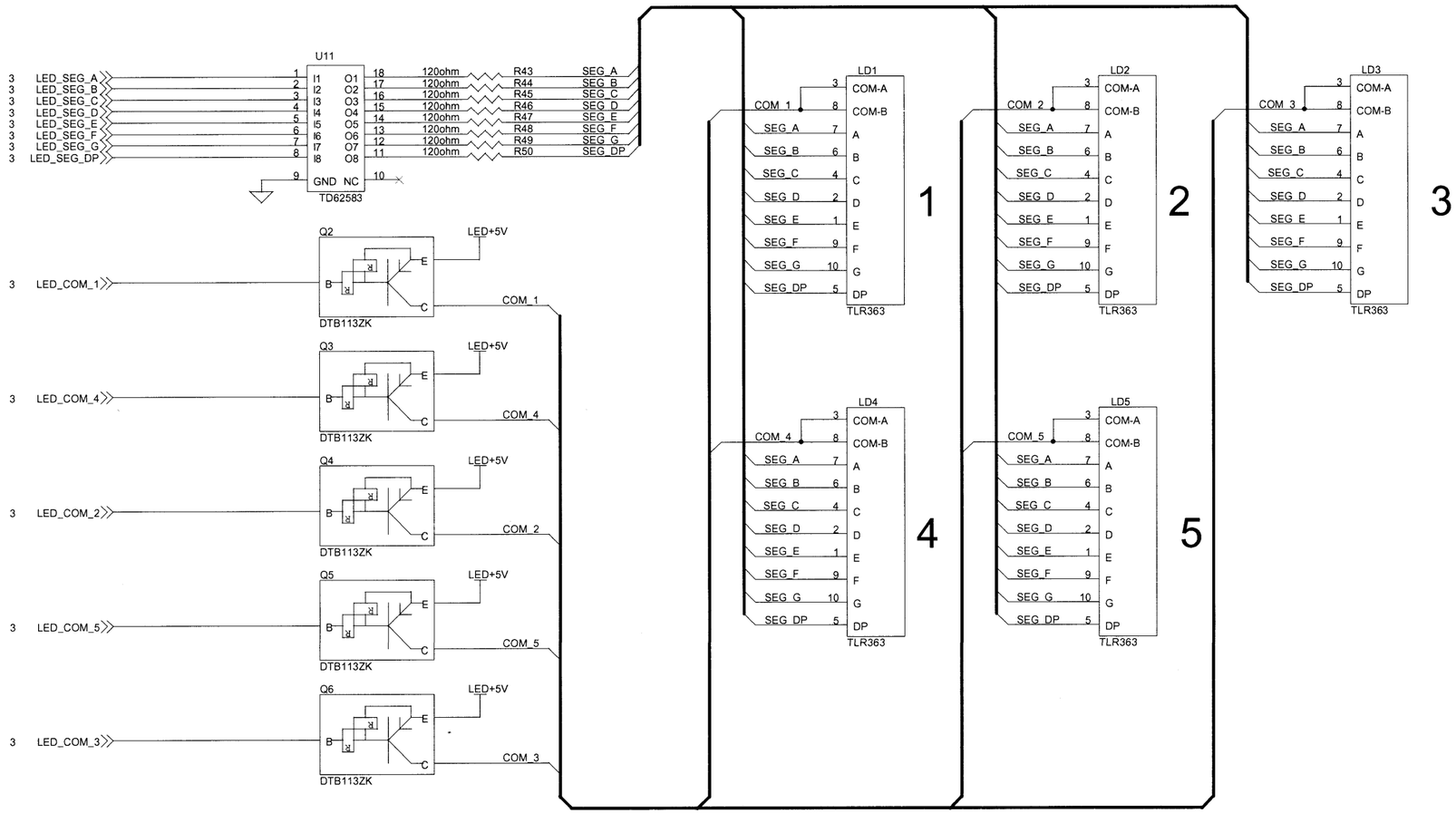


**Figure 4-35. TU-V850E/MA1 (7/8)**

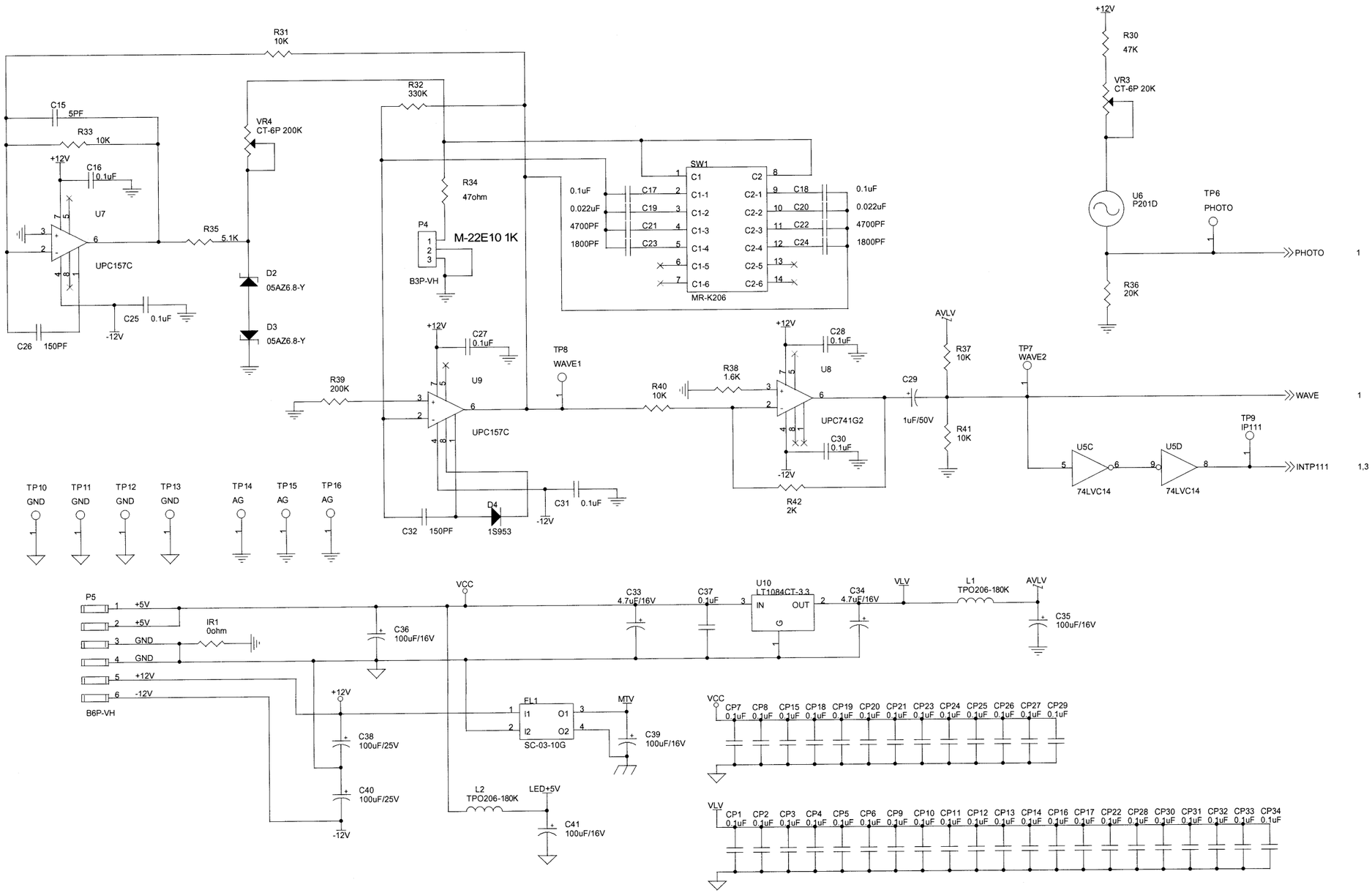
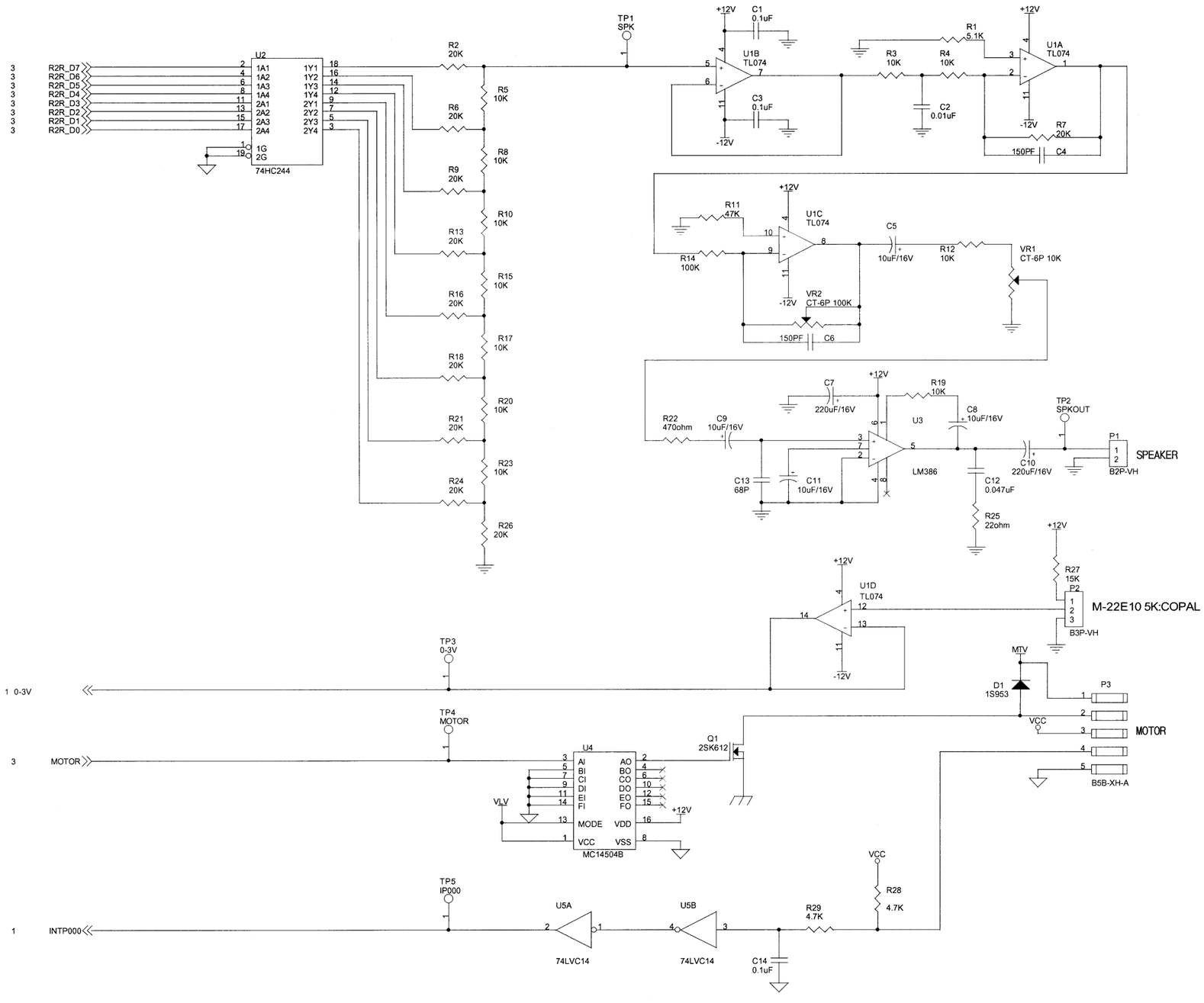


**Figure 4-35. TU-V850E/MA1 (8/8)**



# APPENDIX A INDEX

★

# APPENDIX B REVISION HISTORY

Major Revisions by edition and revised chapters are shown below.

| Edition | Major Revisions from Previous Version | Revised Chapters |
|---|---|---|
| 2nd | • Deletion of the following product names<br>μPD703103, 703105, 703106, 703107, and 70F3107<br>• Addition of the following product names:<br>μPD703103A, 703105A, 703106A, 703106A(A), 703107A, 703107A(A), 70F3107A, and 70F3107A(A) | Throughout |
| | Change of description in **1.4 Ordering Information** | **CHAPTER 1 V850E/MA1 INTRODUCTION** |
| | Addition of **1.5 Pin Configuration (Top View)** | |
| | Addition of **1.6 Internal Block Diagram** | |
| | Change of V850E/MA1 address pin in **2.6 Connection with SDRAM** | **CHAPTER 2 BUS INTERFACE CONNECTION CIRCUIT EXAMPLES (1)** |
| | Change of **Figure 4-5. Settings in System Wait Control Register (VSWC)** | **CHAPTER 4 APPLICATION EXAMPLES** |
| | Change of setting value in VSWC register in **4.2.3 Operaiton example in single-chip mode 0 (STEP 1) [program example]** | |
| | Change of **Figure 4-35. TU-V850E/MA1** | |

**[MEMO]**

NEC

# Facsimile Message

Although NEC has taken all possible steps to ensure that the documentation supplied to our customers is complete, bug free and up-to-date, we readily accept that errors may occur. Despite all the care and precautions we've taken, you may encounter problems in the documentation. Please complete this form whenever you'd like to report errors or suggest improvements to us.

From:

Name

Company

Tel. FAX

Address

*Thank you for your kind support.*


**North America**
NEC Electronics Inc.
Corporate Communications Dept.
Fax: +1-800-729-9288
+1-408-588-6130

**Europe**
NEC Electronics (Europe) GmbH
Market Communication Dept.
Fax: +49-211-6503-274

**South America**
NEC do Brasil S.A.
Fax: +55-11-6462-6829

**Hong Kong, Philippines, Oceania**
NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

**Korea**
NEC Electronics Hong Kong Ltd.
Seoul Branch
Fax: +82-2-528-4411

**P.R. China**
NEC Electronics Shanghai, Ltd.
Fax: +86-21-6841-1137

**Taiwan**
NEC Electronics Taiwan Ltd.
Fax: +886-2-2719-5951

**Asian Nations except Philippines**
NEC Electronics Singapore Pte. Ltd.
Fax: +65-250-3583

**Japan**
NEC Semiconductor Technical Hotline
Fax: +81- 44-435-9608


I would like to report the following error/make the following suggestion:

Document title: _____

Document number: _____ Page number: _____

If possible, please fax the referenced page or drawing.

| Document Rating | Excellent | Good | Acceptable | Poor |
|---|---|---|---|---|
| Clarity | ❑ | ❑ | ❑ | ❑ |
| Technical Accuracy | ❑ | ❑ | ❑ | ❑ |
| Organization | ❑ | ❑ | ❑ | ❑ |