Application Note

78K/0 Series

8-bit Single-chip Microcontrollers

Basics (II)

μPD78044F Subseries
μPD78044H Subseries
μPD780208 Subseries
μPD780228 Subseries

Document No. U10121EJ3V0AN00 (3rd edition)
Date Published August 1997 J

SUMMARY OF CONTENTS

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.

M7 96.5

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: 800-366-9782
Fax: 800-729-9288

NEC Electronics (Germany) GmbH
Duesseldorf, Germany
Tel: 0211-65 03 02
Fax: 0211-65 03 490

NEC Electronics (UK) Ltd.
Milton Keynes, UK
Tel: 01908-691-133
Fax: 01908-670-290

NEC Electronics Italiana s.r.1.
Milano, Italy
Tel: 02-66 75 41
Fax: 02-66 75 42 99

NEC Electronics (Germany) GmbH
Benelux Office
Eindhoven, The Netherlands
Tel: 040-2445845
Fax: 040-2444580

NEC Electronics (France) S.A.
Velizy-Villacoublay, France
Tel: 01-30-67 58 00
Fax: 01-30-67 58 99

NEC Electronics (France) S.A.
Spain Office
Madrid, Spain
Tel: 01-504-2787
Fax: 01-504-2860

NEC Electronics (Germany) GmbH
Scandinavia Office
Taeby, Sweden
Tel: 08-63 80 820
Fax: 08-63 80 388

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 Singapore Pte. Ltd.
United Square, Singapore 1130
Tel: 253-8311
Fax: 250-3583

NEC Electronics Taiwan Ltd.
Taipei, Taiwan
Tel: 02-719-2377
Fax: 02-719-5951

NEC do Brasil S.A.
Sao Paulo-SP, Brasil
Tel: 011-889-1680
Fax: 011-889-1689

J96. 8

Major Changes

Page	Description
Throughout	The following products have been added as applicable products: μPD78044F, μPD78044H, and μPD780228 subseries, μPD780206, and μPD780208
	The following subseries have been dropped as applicable products: μPD78024 and μPD78044A subseries
P.37, 38 P.39, 40 P.53, 54 P.128, 129 P.216, 217 P.258, 260	The following register formats and tables are described separately according to the products: **Tables 3-1** and **3-2**, Figures **3-1**, **3-2**, **4-2**, **4-3**, **8-1**, **8-2**, **9-1**, **9-2**, **10-7**, and **10-8**
P.1	The following subseries have been added in **Section 1.1**. μPD78075B, μPD78075BY, μPD780018, μPD780018Y, μPD780058, μPD780058Y, μPD78058F, μPD78058FY, μPD780034, μPD780034Y, μPD780024, μPD780024Y, μPD78014H, μPD780964, μPD780924, μPD780228, μPD78044H, μPD78044F, μPD780308, μPD780308Y, μPD78064B, μPD78098B, μPD780973, μPD780805 subseries, and μPD78P0914
P.40	**Table 3-3** has been added.
P.53	**Note 2** and **Caution 2** have been added to **Figure 4-2**.
P.55	**Figure 4-4** has been added.
P.63	A **Caution** has been added to **Figure 5-5**.
P.127	**Table 8-2** has been added.
P.130	**Note 4** and a **Caution** have been added to **Figure 8-3**.
P.139	A **Caution** has been added to **Figure 8-9**.
P.141	**Section 8.1** The μPD6252 has been defined as a product for maintenance purposes only.
P.219	**Figure 9-4** has been added.

The mark ★ shows major revised points.

[MEMO]

PREFACE

Target users This application note is for engineers who wish to understand 78K/0 Series devices and design application programs using these devices.

* • **Target products in each subseries**

μPD78044F subseries :	μPD78042F, μPD78043F, μPD78044F, μPD78045F, μPD78P048A
μPD78044H subseries :	μPD78044H, μPD78045H, μPD78046H, μPD78P048B[Note]
μPD780208 subseries :	μPD780204, μPD780205, μPD780206, μPD780208, μPD78P0208
μPD780228 subseries :	μPD780226[Note], μPD780228[Note], μPD78F0228[Note]

Note Under development

Objective The purpose of this application note is to use program examples to help users to understand the basic functions of 78K/0 Series devices.

The program and hardware structures published here are illustrative examples and are not designed for mass production.

Organization This application note is broadly divided into the following areas.

- Overview
- Software
- Hardware

The following application notes are supported.

Document name	Document No.		Applicable subseries	Description
	Japanese	English		
78K/0 Series Application Note, Basics (I)	IEA-715	IEA-1288	µPD78002, 78002Y µPD78014, 78014Y µPD78018F, 78018FY	Describes basic functions of 78K/0 Series products, using program examples.
78K/0 Series Application Note, Basics (II)	U10121J	This manual	µPD78044F µPD78044H µPD780208 µPD780228	
78K/0 Series Application Note, Basics (III)	IEA-767	U10182E	µPD78054, 78054Y µPD78064, 78064Y µPD78078, 78078Y µPD78083 µPD78098	
78K/0 Series Application Note, Floating-Point Operation Program	IEA-718	IEA-1289	All subseries of 78K/0 Series [Except for µPD78002 and µPD78002Y subseries]	Describes the floating-point operation application programs of 78K/0 Series products.
µPD78014 Series Application Note, Electronic Notes	IEA-744	IEA-1301	µPD78014 [Only the µPD78014 and µPD78P014 are applicable.]	Describes the functions and configuration of electronic notes, using µPD78014 subseries products as examples.

Caution In this application note, the application examples and program listings are written for the main system clock operating at 4.19 MHz. They are not for the main system clock operating at 5.0 MHz.

Reading this note This application note is for 78K/0 Series products, but each subseries has different functions. Each subseries is described in the chapters listed in the following table. Sample applications for each subseries are given in those chapters indicated by circles.

Chapter \ Subseries	µPD78044F	µPD78044H	µPD780208	µPD780228
Chapter 1 Overview	o	o	o	o
Chapter 2 Software Basics	o	o	o	o
Chapter 3 System Clock Switching Application	o	o	o	o
Chapter 4 Watchdog Timer Application	o	o	o	o
Chapter 5 16-bit Timer/Event Counter Application	o	o	o	-
Chapter 6 8-bit Timer/Event Counter Application	o	o	o	-
Chapter 7 Watch Timer Application	o	o	o	-
Chapter 8 Serial Interface Application	o	o	o	-
Chapter 9 A/D Converter Application	o	o	o	o
Chapter 10 Applications of FIP Controller/Driver	o	o	o	o
Chapter 11 Applications of 6-bit Up/Down Counter	o	-	-	-

Legend

Significance of the data description : The left side is high-order data and the right side is low-order data.
Active-low description : $\overline{\text{xxx}}$ (line above pin and signal names)
Note : Explanation of the note attached to the text.
Caution : Contents that should be read carefully
Remark : Supplemental explanation of the text
Number descriptions : Binary numbers .... xxxx or xxxxB
Decimal numbers xxxx
Hexadecimal numbers .. xxxxH

Application area

- Consumer product field

Related Documents

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

- **Common documents**

Document name	Document number	
	Japanese	English
78K/0 Series Application Note, Basics (II)	U10121J	This manual
78K/0 Series User's Manual, Instruction	U12326J	IEU-1372
78K/0 Series Instruction Set	U10904J	-
78K/0 Series Instruction Table	U10903J	-

- **Documents for μPD78044F subseries**

Document name	Document number	
	Japanese	English
μPD78042F, 78043F, 78044F, 78045F Data Sheet	U10700J	U10700E
μPD78P048A Data Sheet	U10611J	U10611E
μPD78044F Subseries User's Manual	U10908J	U10908E
μPD78044A, 78044F Subseries Special Function Register Table	U10701J	-

*• **Documents for μPD78044H subseries**

Document name	Document number	
	Japanese	English
μPD78044H, 78045H, 78046H Data Sheet	U10865J	U10865E
μPD78P048B Data Sheet	To be created	To be created
μPD78044H Subseries User's Manual	U11756J	U11756E

- **Documents for μPD780208 subseries**

Document name	Document number	
	Japanese	English
μPD780204, 780205, 780206, 780208 Data Sheet	U10436J	U10436E
μPD78P0208 Data Sheet	U11295J	U11295E
μPD780208 Subseries User's Manual	U11302J	U11302E
μPD780208 Subseries Special Function Register Table	U10997J	-

* • **Documents for μPD780228 subseries**

Document name	Document number	
	Japanese	English
μPD780226, 780228 Data Sheet	U11797J	U11797E
μPD78F0228 Preliminary Product Information	U11971J	U11971E
μPD780228 Subseries User's Manual	U12012J	U12012E

The above documents may be revised without notice. Use the latest versions when you design an application system.

[MEMO]

CONTENTS

LIST OF FIGURES (1/4)

LIST OF FIGURES (2/4)

LIST OF FIGURES (3/4)

LIST OF FIGURES (4/4)

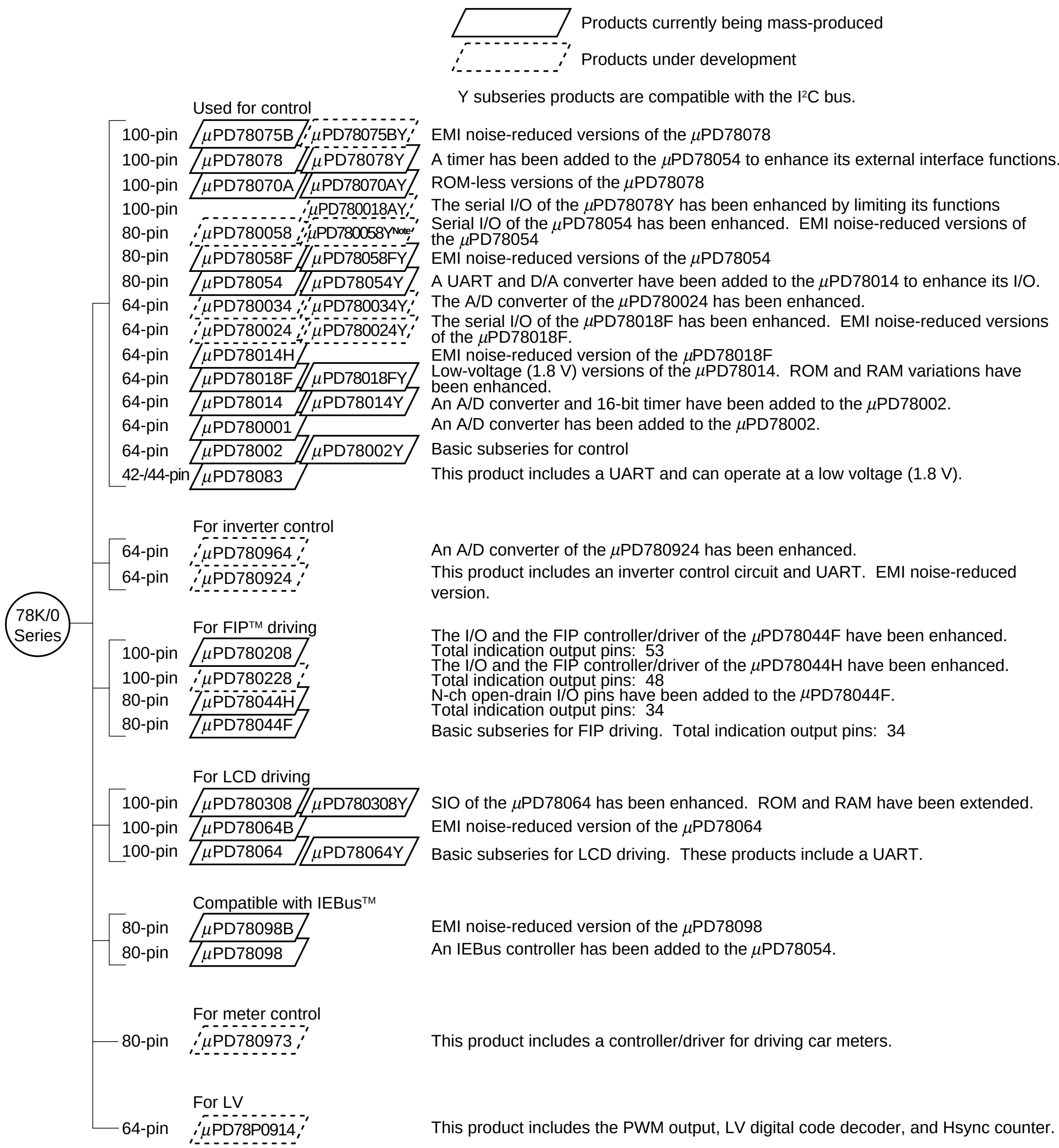
Figure No.	Title	Page

LIST OF TABLES

CHAPTER 1 OVERVIEW

* 1.1 78K/0 SERIES PRODUCT DEVELOPMENT

The 78K/0 series products were developed as shown below. The subseries names are indicated in frames.



Note Being planned

The table below shows the main differences between subseries.

Subseries name		ROM capacity	Timer				8-bit A/D	10-bit A/D	8-bit D/A	Serial interface	I/O	Minimum V_{DD}	External expansion
			8-bit	16-bit	Watch	WDT							
For control	µPD78075B	32K-40 K	4 ch	1 ch	1 ch	1 ch	8 ch	-	2 ch	3 ch (UART: 1 ch)	88 pins	1.8 V	o
	µPD78078	48K-60K											
	µPD78070A	-									61 pins	2.7 V	
	µPD780058	24K-60K	2 ch						2 ch	3ch (time-multiplexing UART: 1ch)	68 pins	1.8 V	
	µPD78058F	48K-60K								3 ch (UART: 1 ch)	69 pins	2.7 V	
	µPD78054	16K-60K										2.0 V	
	µPD780034	8K-32K					-	8 ch	-	3 ch (UART: 1 ch, time-multiplexing 3-wire: 1ch)	51 pins	1.8 V	
	µPD780024						8 ch	-					
	µPD78014H									2 ch	53 pins		
	µPD78018F	8K-60K											
	µPD78014	8K-32K										2.7 V	
	µPD780001	8K		-	-					1 ch	39 pins		-
	µPD78002	8K-16K			1 ch		-				53 pins		o
	µPD78083				-		8 ch			1 ch (UART: 1 ch)	33 pins	1.8 V	-
For inverter control	µPD780964	8K-32K	3 ch	**Note**	-	1 ch	-	8 ch	-	2 ch (UART: 2 ch)	47 pins	2.7 V	o
	µPD780924						8 ch	-					
For FIP driving	µPD780208	32K-60K	2 ch	1 ch	1 ch	1 ch	8 ch	-	-	2 ch	74 pins	2.7 V	-
	µPD780228	48K-60K	3 ch	-	-					1 ch	72 pins	4.5 V	
	µPD78044H	32K-48K	2 ch	1 ch	1 ch						68 pins	2.7 V	
	µPD78044F	16K-40K								2 ch			
For LCD driving	µPD780308	48K-60K	2 ch	1 ch	1 ch	1 ch	8 ch	-	-	3ch (time-multiplexing UART: 1ch)	57 pins	2.0 V	-
	µPD78064B	32K								2 ch (UART: 1 ch)			
	µPD78064	16K-32K											
Compatible with IEBus	µPD78098B	40K-60K	2 ch	1 ch	1 ch	1 ch	8 ch	-	2 ch	3 ch (UART: 1 ch)	69 pins	2.7 V	o
	µPD78098	32K-60K											
For meter control	µPD780973	24K-32K	3 ch	1 ch	1 ch	1 ch	5 ch	-	-	2 ch (UART: 1 ch)	56 pins	4.5 V	-
For LV	µPD78P0914	32K	6 ch	-	-	1 ch	8 ch	-	-	2 ch	54 pins	4.5 V	o

Note 10-bit timer: 1 channel

1.2 78K/0 SERIES FEATURES

The 78K/0 Series devices are 8-bit single-chip microcontrollers ideally suited for applications in the consumer field.

* The μPD78044F subseries are devices that implement high-speed, high-performance CPUs and have on-chip peripheral hardware, such as ROM, RAM, I/O ports, timers, serial interfaces, A/D converter, FIP controller/driver, 6-bit up/down counter, and interrupt controllers.

* The μPD78044H subseries of devices has been implemented by adding N-ch open-drain I/O pins to the μPD78044F subseries.

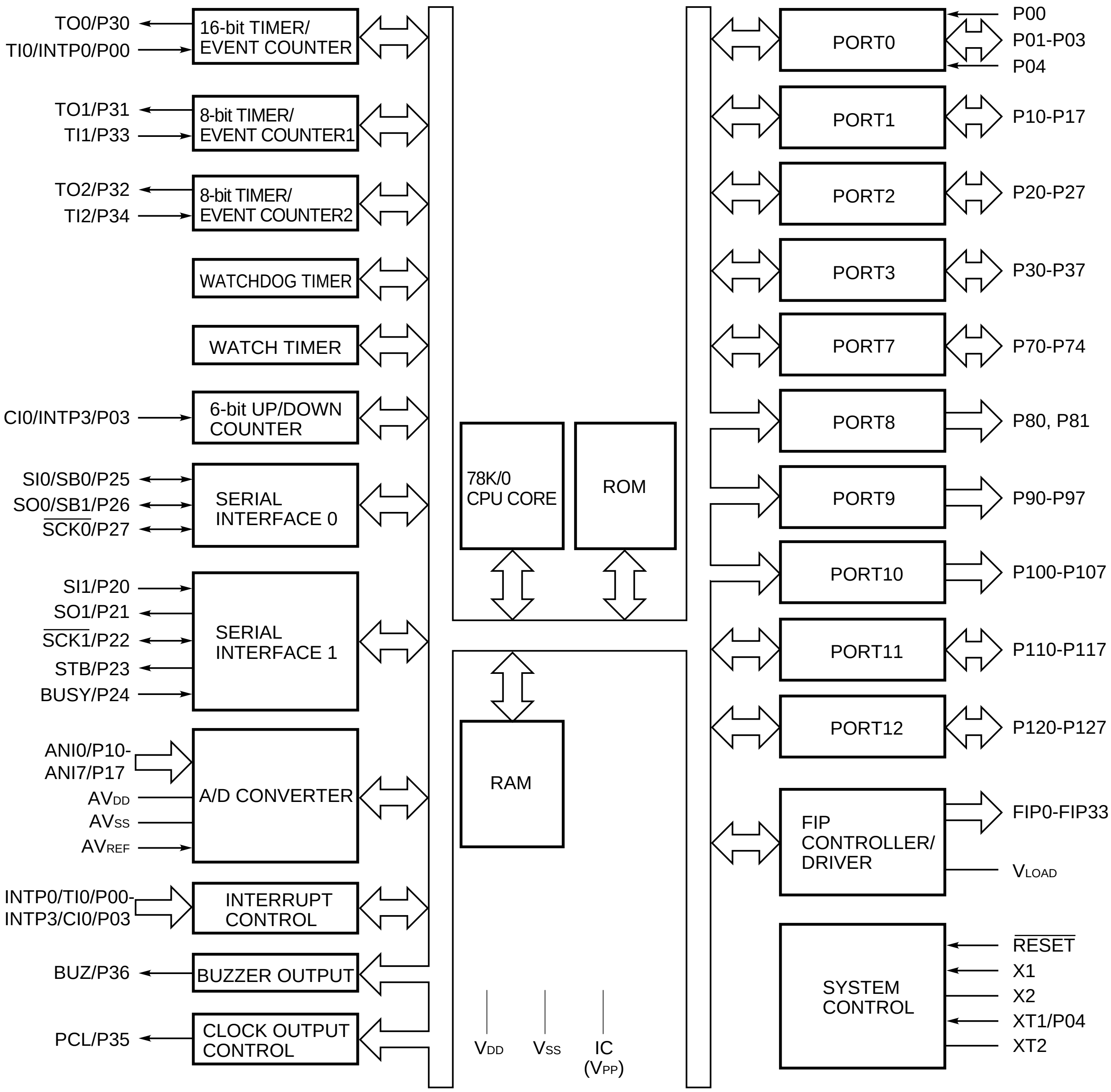
The μPD780208 subseries has an enhanced version of the FIP controller/driver of the μPD78044F subseries.

The μPD780228 subseries has an enhanced version of the FIP controller/driver of the μPD78044H subseries.

* The one-time PROM or EPROM versions or flash memory version, that can operate at the same low voltage as mask ROM versions, such as the μPD78P048A, μPD78P048B, μPD78P0208, and μPD78F0228 are also provided. These products are well suited for fast shift to production of application systems and small-lot production.

A block diagram and an overview of the functions of each subseries are shown on the following pages.

Figure 1-1. Block Diagram of the μPD78044F Subseries



Remarks 1. The capacities of the internal ROM and RAM differ depending on the product.

2. The value enclosed in parentheses is applied to the μPD78P048A.

*

Table 1-1. Function Overview of the µPD78044F Subseries (1/2)

Item		µPD78042F	µPD78043F	µPD78044F	µPD78045F	µPD78P048A
Internal memory	ROM	Masked ROM				One-time PROM/EPROM
		16K bytes	24K bytes	32K bytes	40K bytes	60K bytes[Note 1]
	High-speed RAM	512 bytes		1024 bytes		1024 bytes[Note 2]
	Extended RAM	-				1024 bytes
	Buffer RAM	64 bytes				
	FIP display RAM	48 bytes				
General-purpose registers		8 bits x 8 x 4 banks				
Minimum instruction execution time	For main system clock	0.4 µs/0.8 µs/1.6 µs/3.2 µs/6.4 µs (at 5.0 MHz)				
	For subsystem clock	122 µs (at 32.768 kHz)				
Instruction set		• 16-bit operations • Multiplication/division (8 bits x 8 bits, 16 bits/8 bits) • Bit (set, reset, test, Boolean operations) • BCD conversion, etc.				
I/O ports (including those multiplexed with FIP pins)		• Total : 68 pins • CMOS input : 2 pins • CMOS I/O : 27 pins • N-ch open-drain I/O : 5 pins • P-ch open-drain I/O : 16 pins • P-ch open-drain output : 18 pins				
FIP controller/driver		• Total : 34 pins • Segments : 9 to 24 pins • Digits : 2 to 16 pins				
A/D converter		• 8-bit resolution x 8 channels • Power supply voltage: AV_{DD} = 4.0 to 6.0 V				
Serial interface		• 3-wire serial I/O, SBI, or 2-wire serial I/O mode selectable : 1 channel • 3-wire serial I/O mode (with automatic transmission/ reception function of up to 64 bytes) : 1 channel				
Timer		• 16-bit timer/event counter : 1 channel • 8-bit timer/event counter : 2 channels • Watch timer : 1 channel • Watchdog timer : 1 channel • 6-bit up/down counter : 1 channel				
Timer outputs		3 (one for 14-bit PWM output)				

Notes 1. The memory size switching register (IMS) can be used to select 16K, 24K, 32K, 40K, or 60K bytes.

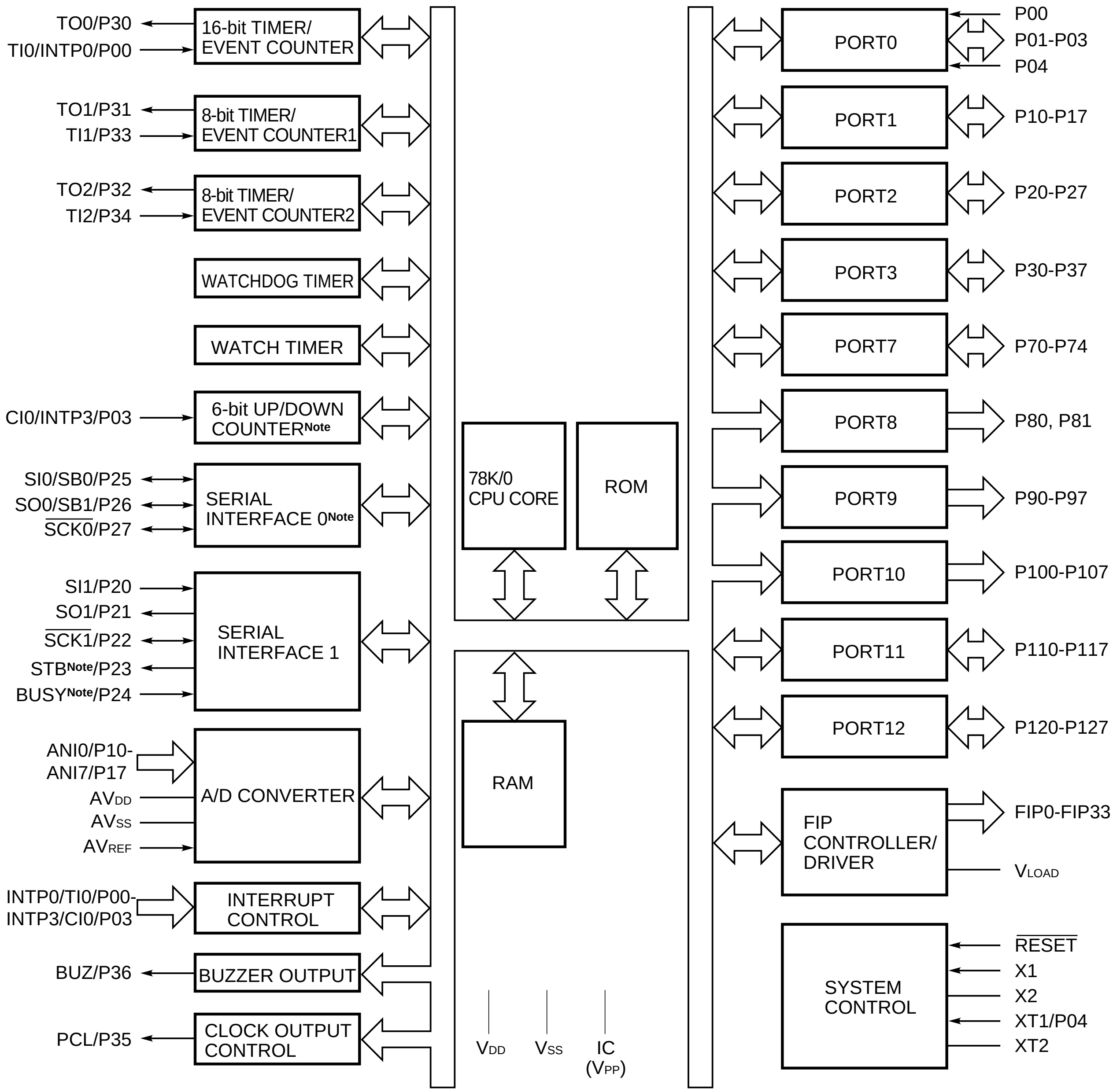
2. The IMS can be used to select 512K or 1024K bytes.

Table 1-1. Function Overview of the μPD78044F Subseries (2/2)

Item \ Product name		μPD78042F	μPD78043F	μPD78044F	μPD78045F	μPD78P048A
Clock output		19.5 kHz, 39.1 kHz, 78.1 kHz, 156 kHz, 313 kHz, 625 kHz (at main system clock of 5.0 MHz) 32.768 kHz (at subsystem clock of 32.768 kHz)				
Buzzer output		1.2 kHz, 2.4 kHz, 4.9 kHz (at 5.0 MHz: main system clock)				
Vectored interrupt factors	Maskable	Internal: 10, external: 4				
	Non-maskable	Internal: 1				
	Software	1				
Test input		Internal: 1				
Power supply voltage		V_{DD} = 2.7 to 6.0 V				
Package		• 80-pin plastic QFP (14 x 20 mm) • 80-pin ceramic WQFN: Only for the μPD78P048A				

*

Figure 1-2. Block Diagram of the µPD78044H Subseries



Note Only for the µPD78P048B

Remarks 1. The capacities of the internal ROM and RAM differ depending on the product.

2. The value enclosed in parentheses is applied to the µPD78P048B.

*

Table 1-2. Function Overview of the μPD78044H Subseries (1/2)

Item		μPD78044H	μPD78045H	μPD78046H	μPD78P048B[Note 1]
Internal memory	ROM	Masked ROM			One-time PROM/EPROM
		32K bytes	40K bytes	48K bytes	60K bytes[Note 2]
	High-speed RAM	1024 bytes			
	Extended RAM	-			1024 bytes[Note 3]
	Buffer RAM	-			64 bytes
	FIP display RAM	48 bytes			
General-purpose register		8 bits x 8 x 4 banks			
Minimum instruction execution time	For main system clock	0.4 μs/0.8 μs/1.6 μs/3.2 μs/6.4 μs (at 5.0 MHz)			
	For subsystem clock	122 μs (at 32.768 kHz)			
Instruction set		• 16-bit operations • Multiplication/division (8 bits x 8 bits, 16 bits/8 bits) • Bit manipulations (set, reset, test, Boolean operations) • BCD conversion, etc.			
I/O (including those multiplexed with FIP pins)		• Total : 68 lines ports • CMOS input : 2 lines • CMOS I/O : 19 lines • N-ch open-drain I/O : 13 lines • P-ch open-drain I/O : 16 lines • P-ch open-drain output : 18 lines			
FIP controller/driver		• Total : 34 lines • Segments: 9 to 24 lines • Digits : 2 to 16 lines			
A/D converter		• 8-bit resolution x 8 channels • Power supply voltage: AV_{DD} = 4.0 to 5.5 V			• 8-bit resolution x 8 channels • Power supply voltage: AV_{DD} = 4.0 to 6.0 V
Serial interface		• 3-wire serial I/O mode: 1 channel			• 3-wire serial I/O, SBI, or 2-wire serial I/O mode: 1 channel • 3-wire serial I/O mode with automatic transmission/reception function: 1 channel

Notes 1. Under development

2. The memory size switching register (IMS) can be used to select 32K, 40K, 48K, or 60K bytes.

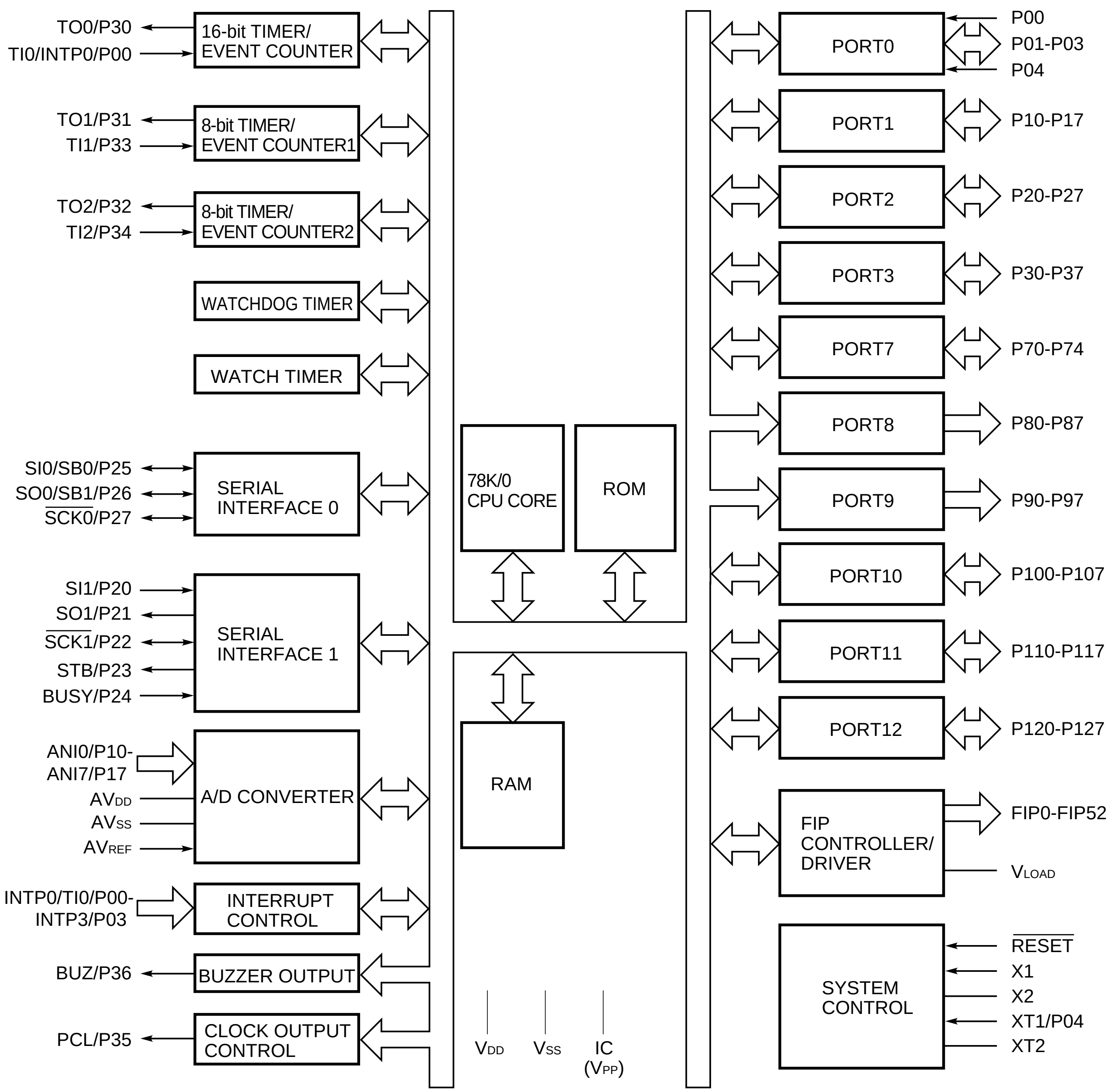
3. The internal extended RAM size switching register (IXS) can be used to select 0 or 1024 bytes.

Table 1-2. Function Overview of the μPD78044H Subseries (2/2)

Item		μPD78044H	μPD78045H	μPD78046H	μPD78P048B[Note]
Timer		• 16-bit timer/event counter : 1 channel • 8-bit timer/event counter : 2 channels • Watch timer : 1 channel • Watchdog timer : 1 channel			• 16-bit timer/event counter: 1 channel • 8-bit timer/event counter: 2 channels • Watch timer: 1 channel • Watchdog timer: 1 channel • 6-bit up/down counter: 1 channel
Timer outputs		3 lines (one for 14-bit PWM output)			
Clock output		19.5 kHz, 39.1 kHz, 78.1 kHz, 156 kHz, 313 kHz, 625 kHz (at main system clock of 5.0 MHz) 32.768 kHz (at subsystem clock of 32.768 kHz)			
Buzzer output		1.2 kHz, 2.4 kHz, 4.9 kHz (at main system clock of 5.0 MHz)			
Vectored interrupt factors	Maskable	Internal: 8, External: 4			Internal: 10, External: 4
	Non-maskable	Internal: 1			
	Software	1			
Test input		Internal: 1			
Power supply voltage		V_{DD} = 2.7 to 5.5 V			V_{DD} = 2.7 to 6.0 V
Package		• 80-pin plastic QFP (14 x 20 mm)			• 80-pin plastic QFP (14 x 20 mm) • 80-pin ceramic WQFN

Note Under development

Figure 1-3. Block Diagram of the μPD780208 Subseries



Remark
1. The capacities of the internal ROM and RAM differ depending on the product.
2. The value enclosed in parentheses is applied to the μPD78P0208.

Table 1-3. Function Overview of the μPD780208 Subseries (1/2)

*

Item		μPD780204	μPD780205	μPD780206	μPD780208	μPD78P0208
Internal memory	ROM	Masked ROM				One-time PROM/EPROM
		32K bytes	40K bytes	48K bytes	60K bytes	60K bytes Note 1
	High-speed RAM	1024 bytes				
	Extended RAM	-		1024 bytes		1024 bytes Note 2
	Buffer RAM	64 bytes				
	FIP display RAM	80 bytes				
General-purpose registers		8 bits x 8 x 4 banks				
Minimum instruction execution time	For main system clock	0.4 μs/0.8 μs/1.6 μs/3.2 μs/6.4 μs (at 5.0 MHz)				
	For subsystem clock	122 μs (at 32.768 kHz)				
Instruction set		• 16-bit operations • Multiplication/division (8 bits x 8 bits, 16 bits/8 bits) • Bit (set, reset, test, Boolean operations) • BCD conversion, etc.				
I/O ports (including those multiplexed with FIP pins)		• Total : 74 pins • CMOS input : 2 pins • CMOS I/O : 27 pins • N-ch open-drain I/O : 5 pins • P-ch open-drain I/O : 24 pins • P-ch open-drain output : 16 pins				
FIP controller/driver		• Total : 53 pins • Segments : 9 to 40 pins • Digits : 2 to 16 pins				
A/D converter		• 8-bit resolution x 8 channels • Power supply voltage: AV_{DD} = 4.0 to 5.5 V				
Serial interface		• 3-wire serial I/O, SBI, or 2-wire serial I/O mode selectable : 1 channel • 3-wire mode (with automatic transmission/ reception function of up to 64 bytes) : 1 channel				
Timer		• 16-bit timer/event counter : 1 channel • 8-bit timer/event counter : 2 channels • Watch timer : 1 channel • Watchdog timer : 1 channel				
Timer outputs		3 (one for 14-bit PWM output)				

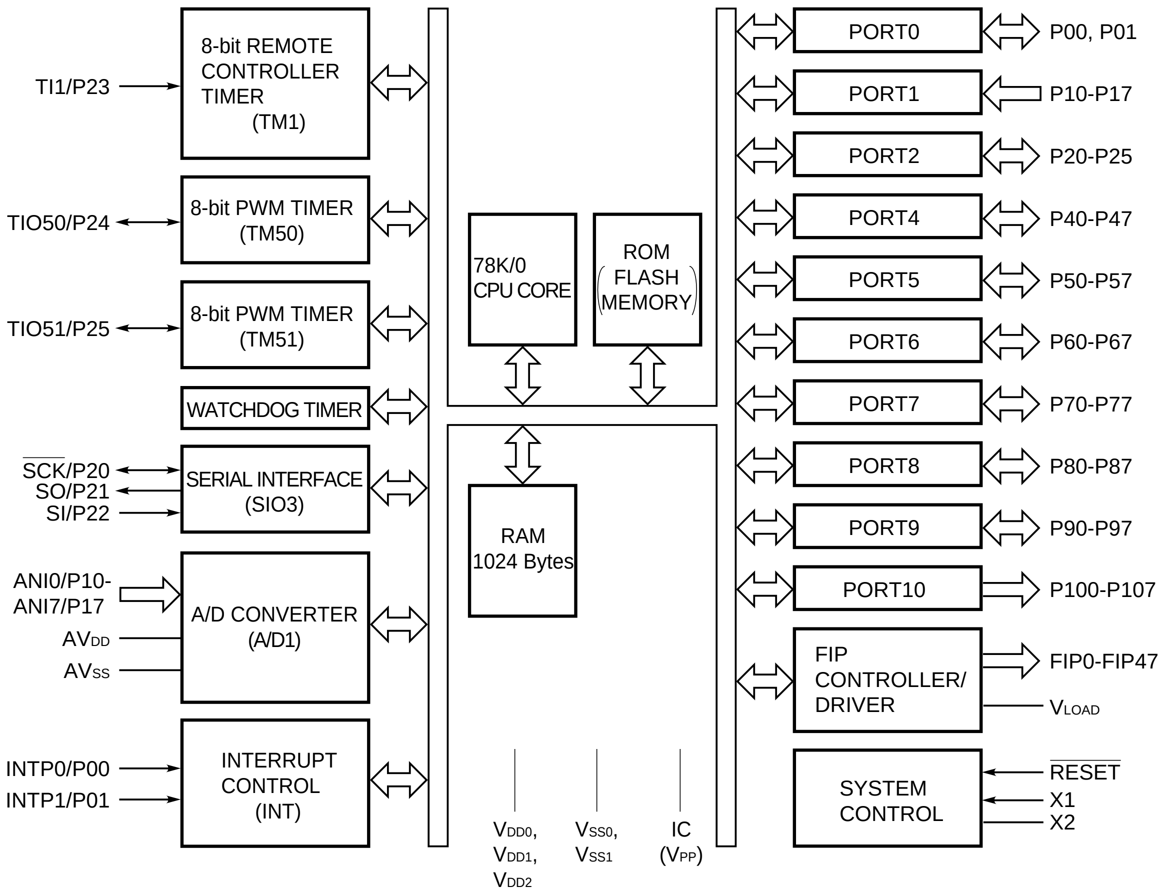
Notes 1. The memory size switching register (IMS) can be used to select 32K, 40K, 48K, or 60K bytes.

2. The internal extended RAM size switching register (IXS) can be used to select either 0 or 1024 bytes.

Table 1-3. Function Overview of the μPD780208 Subseries (2/2)

Item \ Product name		μPD780204	μPD780205	μPD780206	μPD780208	μPD78P0208
Clock output		19.5 kHz, 39.1 kHz, 78.1 kHz, 156 kHz, 313 kHz, 625 kHz (at main system clock of 5.0 MHz) 32.768 kHz (at subsystem clock of 32.768 kHz)				
Buzzer output		1.2 kHz, 2.4 kHz, 4.9 kHz (at 5.0 MHz: main system clock)				
Vectored interrupt factors	Maskable	Internal: 9, external: 4				
	Non-maskable	Internal: 1				
	Software	1				
Text input		Internal: 1				
Power supply voltage		V_{DD} = 2.7 to 5.5 V				
Package		• 100-pin plastic QFP (14 x 20 mm) • 100-pin ceramic WQFN: Only for the μPD78P0208				

* **Figure 1-4. Block Diagram of the μPD780228 Subseries**



Remarks 1. The internal ROM capacity differs depending on the product.

2. The value in parentheses applies to the μPD78F0228 only.

*

Table 1-4. Function Overview of the μPD780228 Subseries

Item \ Product name		μPD780226	μPD780228	μPD78F0228
Internal memory	ROM	Masked ROM		Flash memory
		48K bytes	60K bytes	60K bytes[Note]
	High-speed RAM	1024 bytes		
	Extended RAM	512 bytes		
	FIP display RAM	96 bytes		
General-purpose registers		8 bits x 8 x 4 banks		
Minimum instruction execution time		0.4 μs/0.8 μs/1.6 μs/3.2 μs/6.4 μs (at main system clock of 5.0 MHz)		
Instruction set		• 16-bit operations • Multiplication/division (8 bits x 8 bits, 16 bits/8 bits) • Bit (set, reset, test, Boolean operations) • BCD conversion, etc.		
I/O ports (including those multiplexed with FIP pins)		• Total : 72 pins • CMOS input : 8 pins • CMOS I/O : 16 pins • N-ch open-drain I/O : 16 pins • P-ch open-drain I/O : 24 pins • P-ch open-drain output : 8 pins		
FIP controller/driver		• Total : 48 pins • 10-mA display current : 16 pins • 3-mA display current : 32 pins		
A/D converter		• 8-bit resolution x 8 channels • Power supply voltage: AV_{DD} = 4.5 to 5.5 V		
Serial interface		• 3-wire serial I/O mode: 1 channel		
Timer		• 8-bit remote controller timer : 1 channel • 8-bit PWM timer : 2 channels • Watchdog timer : 1 channel		
Timer outputs		2 (8-bit PWM output enabled)		
Vectored interrupt factors	Maskable	Internal: 6, external: 4		
	Non-maskable	Internal: 1		
	Software	1		
Power supply voltage		V_{DD} = 4.5 to 5.5 V		
Package		100-pin plastic QFP (14 x 20 mm)		

Note The memory size switching register (IMS) can be used to select 48K or 60K bytes.

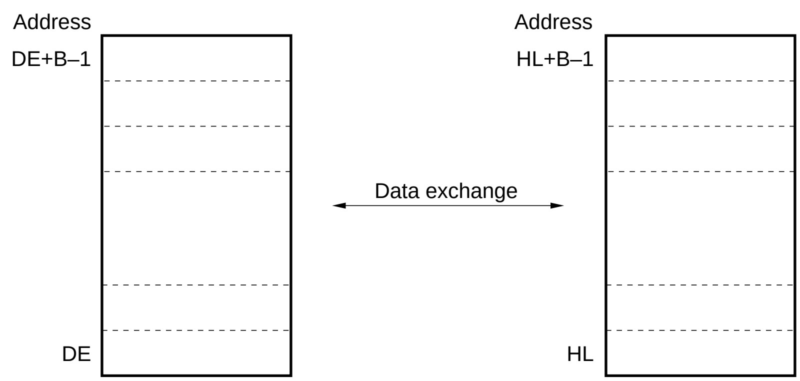
Caution The μPD780228 subseries is under development.

CHAPTER 2 SOFTWARE BASICS

2.1 DATA TRANSFER

The addresses set in the DE and HL registers are the first addresses used in data exchange. The number of bytes in the data exchange is specified in the B register.

Figure 2-1. Data Exchange



(1) Registers used

A, B, DE, HL

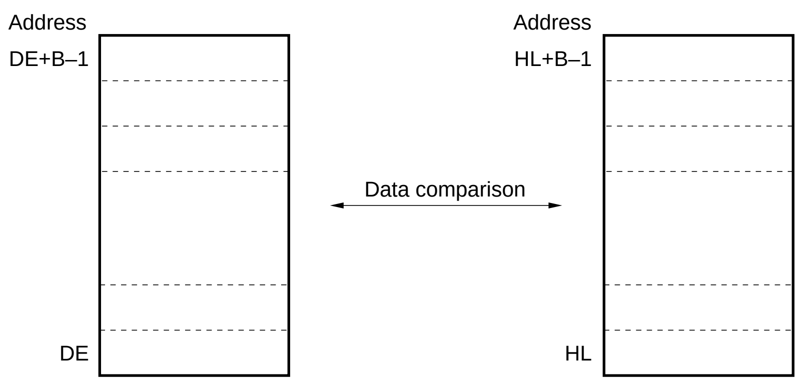
(2) Program listing

```
EXCH:
      MOV    A,[DE]
      XCH    A,[HL]
      XCH    A,[DE]
      INCW   DE
      INCW   HL
      DBNZ   B,$EXCH
      RET
```

2.2 DATA COMPARISON

The addresses set in the DE and HL registers are the first addresses used in data comparison. The number of bytes in the data comparison is specified in the B register. When the comparison result is equal, the CY flag is set to 0. When the result is not equal, CY is set to 1. After the flag setting, processing is returned to the main program.

Figure 2-2. Data Comparison



(1) Registers used

A, B, DE, HL

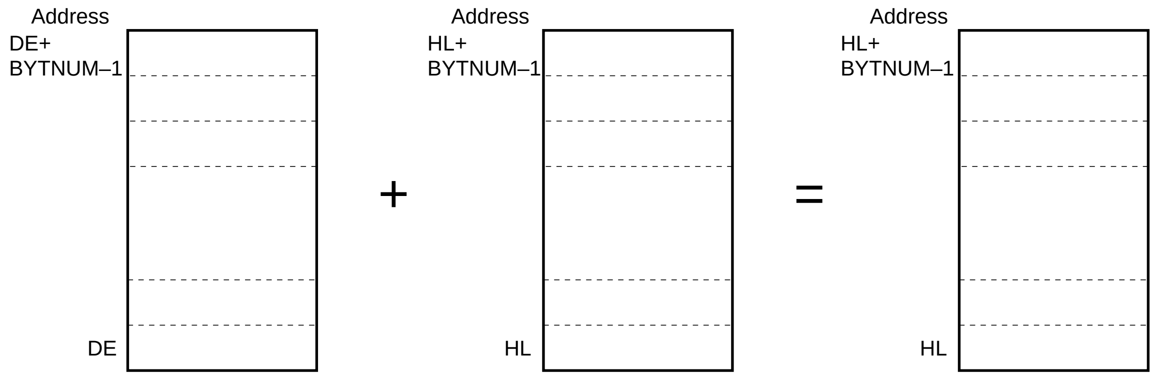
(2) Program listing

```
COMP:
        MOV     A,[DE]
        CMP     A,[HL]
        BNZ     $ERROR
        INCW    DE
        INCW    HL
        DBNZ    B,$COMP
        CLR1    CY
        BR      RTN
ERROR:
        SET1    CY
RTN:
        RET
```

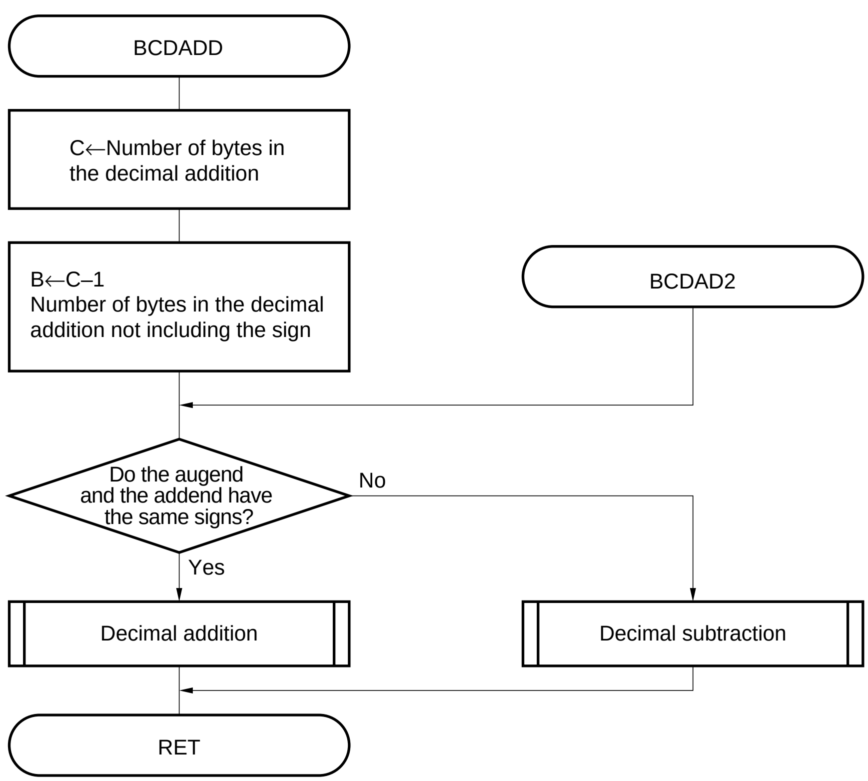
2.3 DECIMAL ADDITION

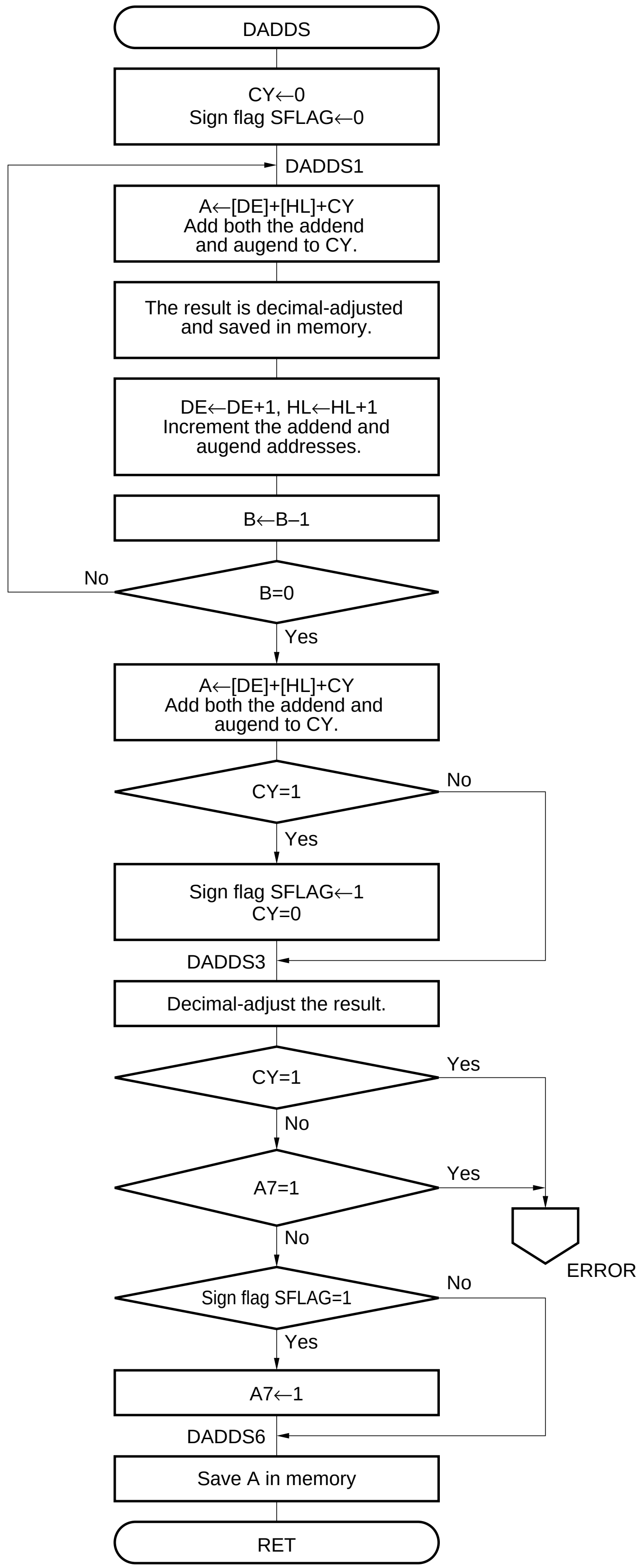
The lowest addresses for decimal addition are specified in the DE and HL registers. The number of digits specified in BYTNUM are added. The addition result is saved in the area pointed to by the HL register. When the addition result is an overflow or an underflow, the processing branches to error processing. Have the branch address defined as 'ERROR' in main program and make it a public declaration.

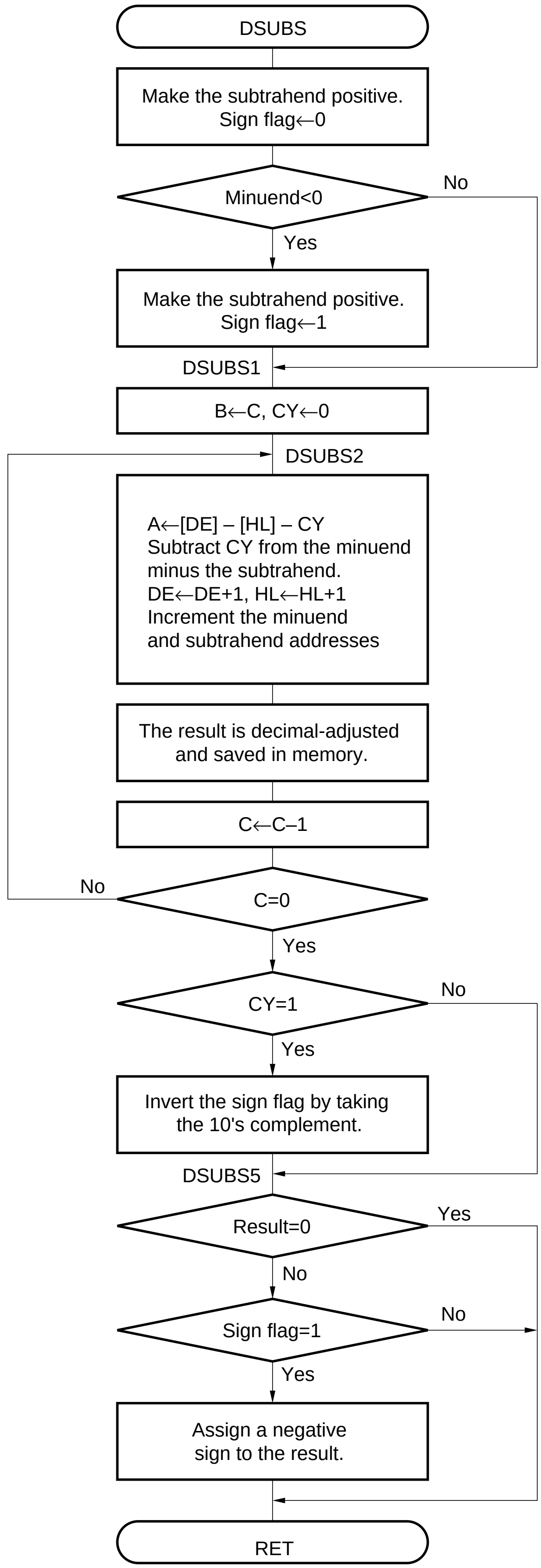
Figure 2-3. Decimal Addition



(1) Flowchart




DADDS
CY←0
Sign flag SFLAG←0
DADDS1
A←[DE]+[HL]+CY
Add both the addend and augend to CY.
The result is decimal-adjusted and saved in memory.
DE←DE+1, HL←HL+1
Increment the addend and augend addresses.
B←B–1
B=0
No
Yes
A←[DE]+[HL]+CY
Add both the addend and augend to CY.
CY=1
No
Yes
Sign flag SFLAG←1
CY=0
DADDS3
Decimal-adjust the result.
CY=1
Yes
No
A7=1
Yes
No
ERROR
Sign flag SFLAG=1
No
Yes
A7←1
DADDS6
Save A in memory
RET


DSUBS
Make the subtrahend positive.
Sign flag←0
Minuend<0
No
Yes
Make the subtrahend positive.
Sign flag←1
DSUBS1
B←C, CY←0
DSUBS2
A←[DE] – [HL] – CY
Subtract CY from the minuend
minus the subtrahend.
DE←DE+1, HL←HL+1
Increment the minuend
and subtrahend addresses
The result is decimal-adjusted
and saved in memory.
C←C–1
No
C=0
Yes
CY=1
No
Yes
Invert the sign flag by taking
the 10's complement.
DSUBS5
Result=0
Yes
No
Sign flag=1
No
Yes
Assign a negative
sign to the result.
RET

(2) Registers used

AX, BC, DE, HL

(3) Program listing

```
;*********************************************************
;                                                        *
;   Input parameters                                     *
;      HL register: start address of the addend          *
;      DE register: start address of the augend          *
;   Output parameters                                    *
;      HL register: start address of the operation result   *
;                                                        *
;*********************************************************

        PUBLIC  BCDADD,BCDAD1,BCDAD2
        PUBLIC  DADDS
        PUBLIC  DSUBS
        EXTRN   ERROR               ; Branch address for error processing
        EXTBIT  SFLAG               ; Sign flag
;
BYTNUM  EQU     4                   ; Set the number of operand digits
;
        CSEG
BCDADD:
        MOV     C,#BYTNUM           ; Set the number of operand digits in the C register.
BCDAD1:
        MOV     A,C
        MOV     B,A
        DEC     B
BCDAD2:
        MOV     A,[HL+BYTNUM-1]     ; Read in the most significant bit (sign data) of the augend
        XCHW    AX,DE
        XCHW    AX,HL
        XCHW    AX,DE
        XOR     A,[HL+BYTNUM-1]     ; Read in the most significant bit (sign data) of the augend
        XCHW    AX,HL
        XCHW    AX,DE
        XCHW    AX,HL

        BT      A.7,$BCDAD3         ; Do the signs agree?  ELSE subtraction processing
        CALL    !DADDS              ; THEN addition processing
        RET
BCDAD3:
        CALL    !DSUBS
        RET
```

```
;=============================================================
;               ***** Decimal Addition *****
;=============================================================

DADDS:
        CLR1  CY
        CLR1  SFLAG
DADDS1:
        MOV   A,[DE]              ; Start addition from the least significant digit
        ADDC  A,[HL]
        ADJBA
        MOV   [HL],A
        INCW  HL
        INCW  DE
        DBNZ  B,$DADDS1           ; End addition of (number-of-operand-digits – 1)

        MOV   A,[DE]
        ADDC  A,[HL]
DADDS2:
        BNC   $DADDS3             ; Negative addition
        SET1  SFLAG               ; THEN set in the negative state
        CLR1  CY
DADDS3:
        ADJBA
        BNC   $DADDS4
        BR    ERROR
DADDS4:
        BF    A.7,$DADDS5
        BR    ERROR
DADDS5:
        BF    SFLAG,$DADDS6       ; Set sign
        SET1  A.7
DADDS6:
        MOV   [HL],A
        RET
```

```
;==================================================================
;                  ***** Decimal Subtraction *****
;==================================================================

DSUBS:
        PUSH    HL
        CLR1    SFLAG
        MOV     A,[HL+BYTNUM-1]  ; Set the subtrahend to positive value.
        CLR1    A.7
        MOV     [HL+BYTNUM-1],A
        XCHW    AX,DE
        XCHW    AX,HL
        XCHW    AX,DE
        MOV     A,[HL+BYTNUM-1]
        BF      A.7,$DSUBS1      ; The minuend is negative.
        CLR1    A.7              ; THEN set the minuend to a positive value.
        MOV     [HL+BYTNUM-1],A
        SET1    SFLAG            ; Set the sign to negative.
DSUBS1:
        XCHW    AX,HL
        XCHW    AX,DE
        XCHW    AX,HL
        MOV     A,C
        MOV     B,A
        CLR1    CY
DSUBS2:
        MOV     A,[DE]
        SUBC    A,[HL]
        ADJBS
        MOV     [HL],A
        INCW    HL
        INCW    DE
        DBNZ    C,$DSUBS2        ; End of the subtraction of the number of operand digits.

        BNC     $DSUBS5          ; THEN subtrahend > minuend
        POP     HL
        PUSH    HL
        MOV     A,B
        MOV     C,A
DSUBS3:
        MOV     A,#99H           ; Complement operation on the subtraction result
        SUB     A,[HL]           ; (subtraction-result – 99H)
        ADJBS
        MOV     [HL],A
        INCW    HL
        DBNZ    C,$DSUBS3


        POP     HL
        PUSH    HL
        SET1    CY

        MOV     A,B
        MOV     C,A
```

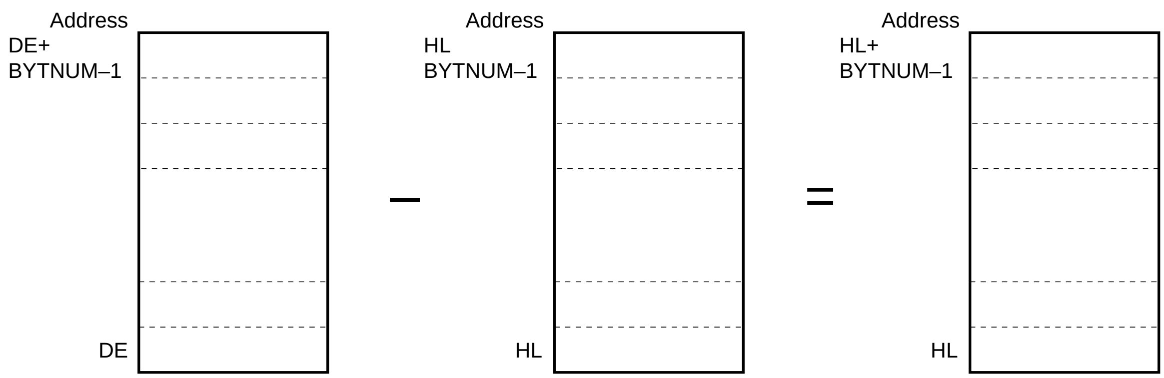
```
DSUBS4:
        MOV     A,#0                ; Add 1 to the complement operation result.
        ADDC    A,[HL]
        ADJBA

        MOV     [HL],A
        INCW    HL
        DBNZ    C,$DSUBS4
        MOV1    CY,SFLAG
        NOT1    CY
        MOV1    SFLAG,CY
;=====================================================
;       ***** 0 Check of Operation Result *****
;=====================================================

DSUBS5:
        MOV     A,B
        MOV     C,A
        POP     HL
        PUSH    HL
        MOV     A,#0
DSUBS6:
        CMP     A,[HL]              ; 0 check from the low-order digit
        INCW    HL
        BNZ     $DSUBS7
        DBNZ    C,$DSUBS6           ; End of checking all digits for 0
        POP     HL                  ; THEN subtraction result = 0
        RET
DSUBS7:
        BF      SFLAG,$DSUBS8       ; Subtraction result is negative.
        POP     HL                  ; THEN set sign
        PUSH    HL
        MOV     A,[HL+BYTNUM-1]
        SET1    A.7
        MOV     [HL+BYTNUM-1],A
DSUBS8:
        POP     HL
        RET
```

2.4 DECIMAL SUBTRACTION

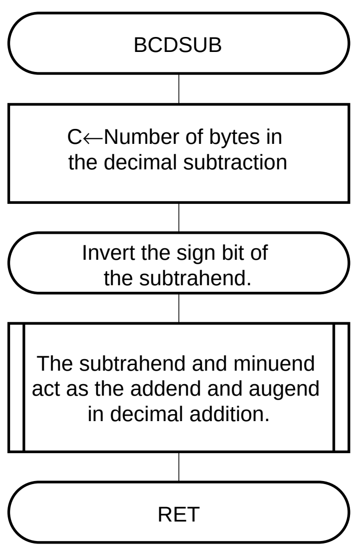
The lowest addresses for decimal subtraction are set in the DE and HL registers. Subtraction is performed on the number of digits specified in BYTNUM. The subtraction result is saved in the area specified in the HL register. Additionally, when the subtraction result is an overflow or an underflow, the processing branches to error processing. Have the branch address defined as 'ERROR' in main program and make it a public declaration.

This program replaces the augend and addend with the minuend and subtrahend respectively, and calls the decimal addition program.

Figure 2-4. Decimal Subtraction



(1) Flowchart



(2) Registers used

AX, BC, DE, HL

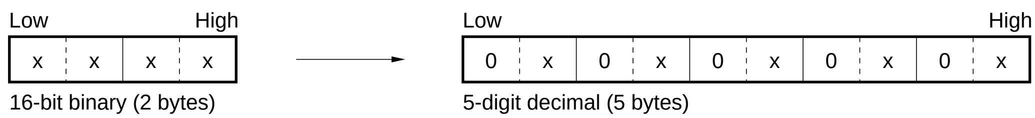
(3) Program listing

```
;************************************************************
;   Input parameters                                        *
;      HL register: start address of the subtrahend         *
;      DE register: start address of the minuend            *
;   Output parameters                                       *
;      HL register: start address of the operation result   *
;                                                           *
;************************************************************

       PUBLIC BYTNUM
       PUBLIC BCDSUB
       EXTRN  BCDADD,BCDAD2
;
BYTNUM EQU    4                   ; Set the number of operand digits
;
       CSEG
BCDSUB:
       MOV    C,#BYTNUM           ; Set the number of operand digits in the C register.
BCDSU1:
       MOV    A,C
       MOV    B,A
       DEC    B

       MOV    A,[HL+BYTNUM-1]     ; Set the most significant bit (sign data) of the subtrahend for use in
addition.
       MOV1   CY,A.7              ; Invert the sign data.
       NOT1   CY
       MOV1   A.7,CY
       MOV    [HL+BYTNUM-1].A
       CALL   !BCDAD2             ; Call decimal addition processing.
       RET
```

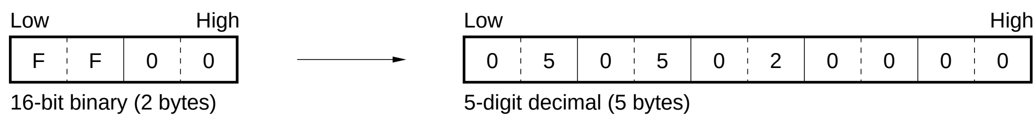
2.5 BINARY-TO-DECIMAL CONVERSION

16-bit binary data in the data memory is converted into 5-digit decimal data and saved in the data memory. The 16-bit binary data are divided by the decimal number 10 (4 times) and the conversion is based on the values of the results and remainders of these operations.

Figure 2-5. Binary-to-Decimal Conversion



Example FFH is converted into decimal.



(1) Registers used

AX, BC, HL

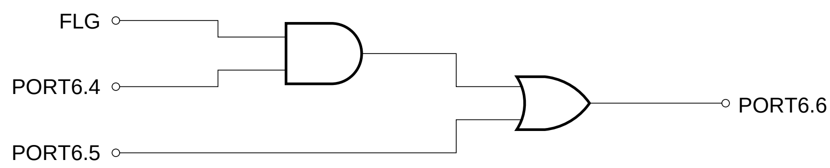
(2) Program listing

```
       PUBLIC B_DCONV
       DATDEC EQU     10

       DSEG   SADDRP
REGA:  DS     2            ; Save 16-bit binary data.
REGB:  DS     5            ; Save 5-digit decimal data.

       COLUMN EQU     4

B_DCONV:
       MOVW   AX,REGA
       MOV    B,#COLNUM
       MOVW   HL,#REGB
B_D1:
       MOV    C,#DATDEC
       DIVUW  C
       XCH    A,C
       MOV    [HL],A
       INCW   HL
       XCH    A,C
       DBNZ   B,$B_D1
       MOV    A,X
       MOV    [HL],A
       RET
```

2.6 BIT OPERATION MANIPULATION INSTRUCTION

The logical product (AND) of the 1-bit flag in data memory and bit 4 in port 6 is taken. The logical sum (OR) of the result and bit 5 of port 6 is output to bit 6 of port 6.

Figure 2-6. Bit Operation



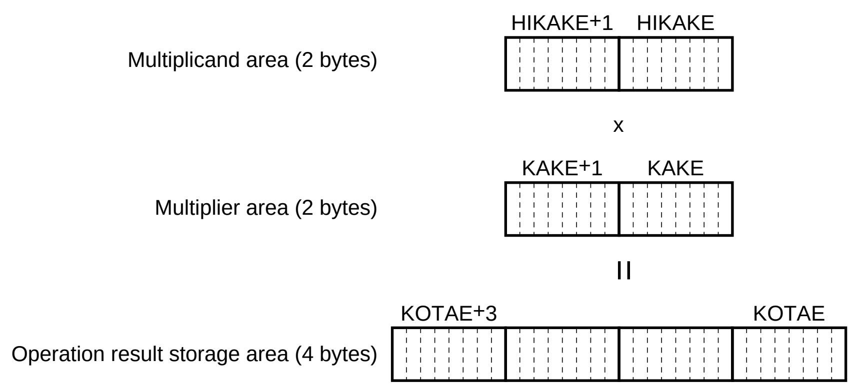
(1) Program listing

```
        PUBLIC  BIT_OP,FLG

        BSEG
FLG     DBIT

BIT_OP:
        MOV1    CY,FLG
        AND1    CY,P6.4
        OR1     CY,P6.5
        MOV1    P6.6,CY
        RET
```

2.7 BINARY MULTIPLICATION (16 BITS x 16 BITS)

The data in the multiplicand area (HIKAKE; 16 bits) and the multiplier area (KAKE; 16 bits) are multiplied. The result is saved in the operation result storage area (KOTAE).

Figure 2-7. Binary Multiplication



<Processing>

Multiplication is implemented by adding the multiplicand only the number of “1” bits in the multiplier.

<Use>

Set the data in the multiplicand area (HIKAKE) and the multiplier area (KAKE), and then call the subroutine S_KAKERU.

```
        EXTRN S_KAKERU
        EXTRN HIKAKE,KAKE,KOTAE
MAIN:                             ; Multiplier
                .
                .
        HIKAKE=WORKA (A)          ; Multiplicand data save in the multiplicand area
        HIKAKE+1=WORKA+1 (A)      ;
        KAKE=WORKB (A)            ; Multiplier data save in the multiplier area
        KAKE+1=WORKB+1 (A)        ;
        CALL  !S_KAKERU           ; Multiplication routine call
        HL=#KOTAE                 ; HL <- RAM address of the operation result storage area
                .                 ; Stores the result by the indirect address transfer
                .
                .
```

Caution Manipulate data memory in 8-bit units.

(1) I/O conditions

- Input parameters

  HIKAKE : Save the multiplicand data.

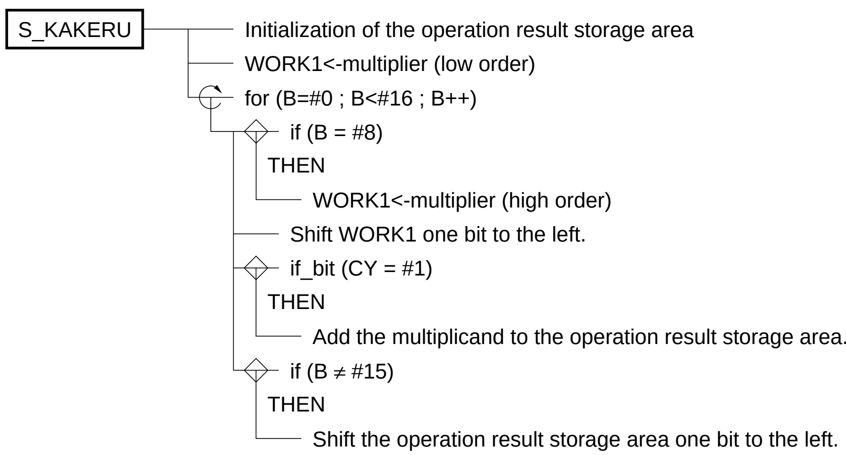
  KAKE : Save the multiplier data.

- Output parameter

  KOTAE : Saves the operation result.

(2) SPD chart

[Multiplication subroutine]



(3) Registers used

A, B

(4) Program listing

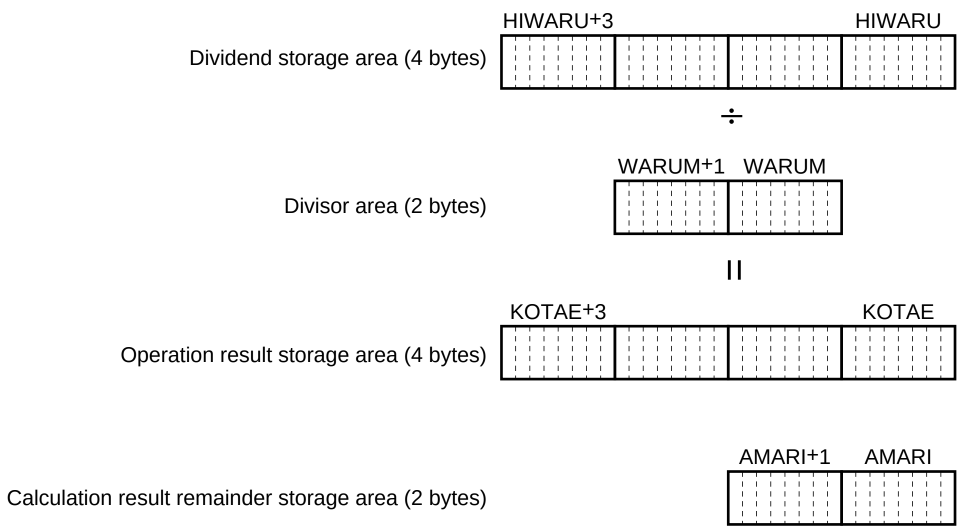
```
$PC(044A)
;
PUBLIC HIKAKE,S_KAKERU,KAKE,KOTAE
;
;*****************************************
;     RAM definition
;*****************************************
            DSEG    SADDR
HIKAKE:     DS      2                         ; Multiplicand area
KAKE:       DS      2                         ; Multiplier area
WORK1:      DS      1                         ; Work area
KOTAE:      DS      4                         ; Operation result storage area
;
;*****************************************
;     Multiplication
;*****************************************
        CSEG                                  ;
S_KAKERU:                                     ;
        WORK1=KAKE+1 (A)                      ; Save multiplier (low order) in the work area.
        KOTAE=#0                              ; Initialize the operation result storage area.
        KOTAE+1=#0                            ;
        KOTAE+2=#0                            ;
        KOTAE+3=#0                            ;
        for(B=#0;B<#16;B++) (A)               ; If at the end of the low-order multiplier,
            if(B == #8) (A)                   ; save the high-order multiplier in the work area.
                WORK1=KAKE (A)                ;
                endif                         ;
                A=WORK1                       ; Shift the multiplier one bit to the left.
                CLR1   CY                     ;
                ROLC   A,1                    ;
                WORK1=A                       ;
                if_bit (CY)                   ; If carry,
                   KOTAE+=HIKAKE (A)          ; add the multiplicand to the operation result
                   (KOTAE+1)+=HIKAKE+1,CY (A) ; storage area.
                   (KOTAE+2)+=#0,CY (A)       ;
                   (KOTAE+3)+=#0,CY (A)       ;
                endif                         ;
                if(B != #15) (A)              ;
                   KOTAE+=KOTAE (A)           ; Shift the operation result storage area one bit to
                   KOTAE+1+=KOTAE+1,CY (A)    ; the left.
                   KOTAE+2+=KOTAE+2,CY (A)    ;
                   KOTAE+3+=KOTAE+3,CY (A)    ;
                endif                         ;
            next                              ;
            RET                               ;
            END                               ;
```

2.8 BINARY DIVISION (32 BITS/16 BITS)

The dividend area (HIWARU; 32 bits) is divided by the divisor area (WARUM; 16 bits) and the result is saved in the operation result storage area (KOTAE). If there is a remainder, it is saved in the calculation result remainder storage area (AMARI).

When the divisor is 0, an error results.

Figure 2-8. Binary Division



<Processing>

The dividend is shifted left starting from the high-order digit into the work area. If the contents of the work area is greater than the divisor, the divisor is subtracted from the work area, and 1 is set in the least significant bit of the dividend. In the above method, division is implemented by operating only on the number of bits in the dividend.

When the divisor is 0, the error flag (F_ERR) is set.

<Use>

Set data in the dividend area (HIWARU) and divisor area (WARUM), and then call the S_WARU subroutine.

```
        EXTRN S_WARU
        EXTRN HIWARU,WARUM,KOTAE
        EXBIT F_ERR

MAIN:
              .                   ;
              .                   ;
        HIWARU=WORKA (A)          ; Save the dividend data in the dividend area
        HIWARU+1=WORKA+1 (A)      ;
        WARUM=WORKB (A)           ; Save divisor data in the divisor area
        WARUM+1=WORKB+1 (A)       ;
        CALL  !S_WARU             ; Division routine call
        HL=#KOTAE                 ; HL <- Save the RAM address of the operation result
              .                   ; storage area
              .                   ;
        if_bit(F_ERR)             ;
           Calculation error processing ;
        endif                     ;
              .
              .
              .
```

Caution Manipulate data memory in 8-bit units.

(1) I/O conditions

- Input parameters

  HIWARU : Save the dividend data.

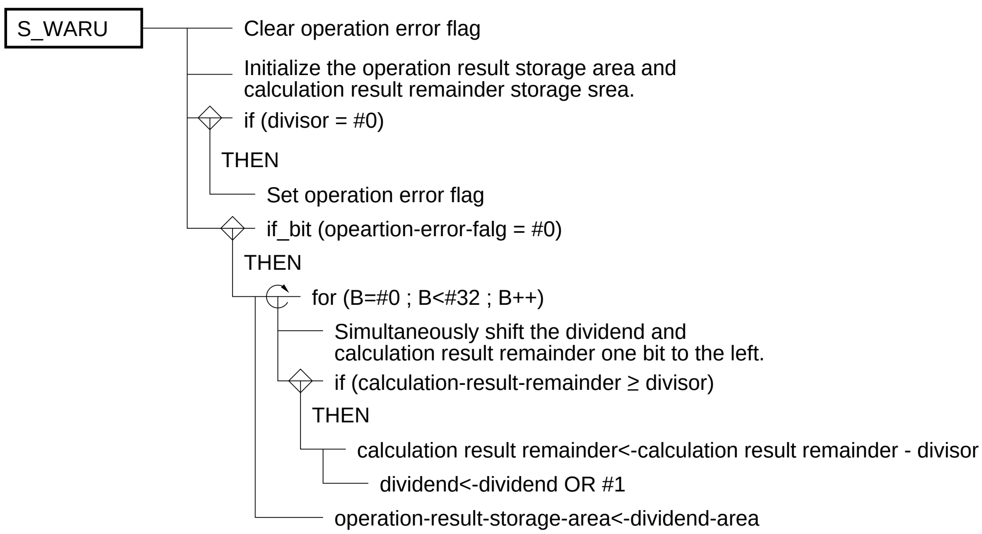
  WARUM : Save the divisor data.

- Output parameters

  KOTAE : Save the calculation result.

(2) SPD chart

[Division subroutine]



(3) Registers used

A, B

(4) Program listing

```
$PC(044A)
;
PUBLIC    S_WARU,HIWARU,WARUM,F_ERR
EXTRN     KOTAE
;
;******************************************
;                RAM definition
;******************************************
        DSEG    SADDR
HIWARU: DS      4                          ; Dividend area
WARUM:  DS      2                          ; Divisor area
AMARI:  DS      2                          ; Calculation result remainder storage area
        BSEG
F_ERR   DBIT                               ; Operation error flag
;******************************************
;                 Division
;******************************************
      CSEG                                 ;
S_WARU:                                    ;
      CLR1 F_ERR                           ; Clear operation error flag
      AMARI=#0                             ; Clear the calculation result remainder storage area
      AMARI+1=#0                           ; to zero
      KOTAE=#0                             ; Clear the operation result storage area to zero
      KOTAE+1=#0                           ;
      KOTAE+2=#0                           ;
      KOTAE+3=#0                           ;
      if(WARUM == #0)                      ; Divisor = 0?
        if(WARUM+1 == #0)                  ;
          SET1   F_ERR                     ; If the divisor is 0, set the operation error flag.
        endif                              ;
      endif                                ;
      if_bit(!F_ERR)                       ; Operation error?
        for(B=#0;B < #32;B++) (A)          ; Start the 32-bit division.
          HIWARU+=HIWARU (A)               ; Shift the dividend and the remainder one bit to the left.
          HIWARU+1+=HIWARU+1,CY (A)        ;
          HIWARU+2+=HIWARU+2,CY (A)        ;
          HIWARU+3+=HIWARU+3,CY (A)        ;
          AMARI+=AMARI,CY (A)              ;
          AMARI+1+=AMARI+1,CY (A)          ;
                                           ;
          if(AMARI+1 > WARUM+1) (A)        ; Remainder ≥ divisor?
            AMARI-=WARUM (A)               ; Remainder = remainder – divisor
            AMARI+1-=WARUM+1,CY (A)        ;
            HIWARU |= #1                   ; Save 1 in the first bit of the dividend area.
          elseif_bit(Z)                    ;
            if(AMARI >= WARUM) (A)         ;
              AMARI-=WARUM(A)              ;
              AMARI+1-=WARUM+1,CY (A);
              HIWARU |= #1                 ;
            endif                          ;
          endif                            ;
        next                               ;
        KOTAE=HIWARU (A)                   ; Save the operation result.
        KOTAE+1=HIWARU+1 (A)               ;
        KOTAE+2=HIWARU+2 (A)               ;
        KOTAE+3=HIWARU+3 (A)               ;
      endif                                ;
      RET                                  ;
      END
```

CHAPTER 3 SYSTEM CLOCK SWITCHING APPLICATION

The 78K/0 Series can control the selection of the CPU clock and oscillator operation by rewriting the processor clock control register (PCC).

The display mode registers 0 and 1 (DSPM0, DSPM1) can be used to set mode of the noise eliminator for the subsystem clock and enable or disable display operation (except for the μPD780228 subseries).

When the CPU clock is changed, it takes the time shown in Tables 3-1 and 3-2 from when a rewrite instruction is used to the PCC until the CPU clock is actually changed. For a while after an instruction to rewrite the PCC is issued, therefore, it cannot be determined which clock, old or new, is used by the CPU. When a main system clock is to be stopped or a STOP instruction is to be executed, a wait enough to assure instructions listed in Tables 3-1 and 3-2 have been executed is needed.

Table 3-1. Maximum Time Required to Change the CPU Clock (μPD78044F, μPD78044H, and μPD780208 Subseries)

Setting before switching				Setting after switching					
CSS	PCC2	PCC1	PCC0	CSS PCC2 PCC1 PCC0 0 0 0 0	CSS PCC2 PCC1 PCC0 0 0 0 1	CSS PCC2 PCC1 PCC0 0 0 1 0	CSS PCC2 PCC1 PCC0 0 0 1 1	CSS PCC2 PCC1 PCC0 0 1 0 0	CSS PCC2 PCC1 PCC0 1 x x x
0	0	0	0		16 instructions	16 instructions	16 instructions	16 instructions	$f_X/2f_{XT}$ instructions (64)
	0	0	1	8 instructions		8 instructions	8 instructions	8 instructions	$f_X/4f_{XT}$ instructions (32)
	0	1	0	4 instructions	4 instructions		4 instructions	4 instructions	$f_X/8f_{XT}$ instructions (16)
	0	1	1	2 instructions	2 instructions	2 instructions		2 instructions	$f_X/16f_{XT}$ instructions (8)
	1	0	0	1 instruction	1 instruction	1 instruction	1 instruction		$f_X/32f_{XT}$ instructions (4)
1	x	x	x	1 instruction	1 instruction	1 instruction	1 instruction	1 instruction	

Caution Selecting of the frequency division of the CPU clock (PCC0-PCC2) and switching from main system clock to subsystem clock (CSS: 0 -> 1) must not be performed simultaneously.
However, selecting of the frequency division of the CPU clock (PCC0-PCC2) and switching from subsystem clock to main system clock (CSS: 1 -> 0) can be performed simultaneously.

Remarks 1. The execution time of one instruction is the minimum instruction execution time of the CPU clock before switching.

2. Time enclosed in parentheses is required when f_X = 5.0 MHz and f_{XT} = 32.768 kHz.

* **Table 3-2. Maximum Time Required to Change the CPU Clock (μPD780228 Subseries)**

Setting before switching			Setting after switching														
PCC2	PCC1	PCC0	PCC2	PCC1	PCC0	PCC2	PCC1	PCC0	PCC2	PCC1	PCC0	PCC2	PCC1	PCC0	PCC2	PCC1	PCC0
			0	0	0	0	0	1	0	1	0	0	1	1	1	0	0
0	0	0				16 instructions			16 instructions			16 instructions			16 instructions		
0	0	1	8 instructions						8 instructions			8 instructions			8 instructions		
0	1	0	4 instructions			4 instructions						4 instructions			4 instructions		
0	1	1	2 instructions			2 instructions			2 instructions						2 instructions		
1	0	0	1 instruction			1 instruction			1 instruction			1 instruction					

Remark The execution time of one instruction is the minimum instruction execution time of the CPU clock before switching.

Figure 3-1. Format of the Processor Clock Control Register (μPD78044F, μPD78044H, and μPD780208 Subseries)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
PCC	MCC	FRC	CLS	CSS	0	PCC2	PCC1	PCC0	FFFBH	04H	R/W[Note 1]

R/W

CSS	PCC2	PCC1	PCC0	CPU clock (f_{CPU}) selection[Note 2]
0	0	0	0	f_X
	0	0	1	$f_X/2$
	0	1	0	$f_X/2^2$
	0	1	1	$f_X/2^3$
	1	0	0	$f_X/2^4$
1	0	0	0	$f_{XT}/2$
	0	0	1	
	0	1	0	
	0	1	1	
	1	0	0	
Other than the above				Setting prohibited

R

CLS	CPU clock status
0	Main system clock
1	Subsystem clock

R/W

FRC	Selection of the feedback resistor of the subsystem clock
0	Use on-chip feedback resistor.
1	Do not use on-chip feedback resistor.

R/W

MCC	Control of the main system clock's oscillation[Note 3]
0	Oscillation possible
1	Oscillation stop

Notes 1. Bit 5 is read-only.

2. In the μPD78044F and μPD78044H subseries, FIP display is possible only when CSS is 0 and PCC2-PCC0 is 000 or 001.

3. When the CPU is operating under the subsystem clock, use MCC to stop the oscillation of the main system clock. Do not use the STOP instruction.

Caution Always set 0 in bit 3.

Remarks 1. f_X : Oscillation frequency of the main system clock

2. f_{XT}: Oscillation frequency of the subsystem clock

* **Figure 3-2. Format of the Processor Clock Control Register (μPD780228 Subseries)**

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
PCC	0	0	0	0	0	PCC2	PCC1	PCC0	FFFBH	04H	R/W

PCC2	PCC1	PCC0	CPU clock (f_{CPU}) selection
0	0	0	f_X
0	0	1	$f_X/2$
0	1	0	$f_X/2^2$
0	1	1	$f_X/2^3$
1	0	0	$f_X/2^4$
Other than the above			Setting prohibited

Caution Always set 0 in bits 3 to 7.

Remark f_X: Oscillation frequency of the main system clock

Of the instructions for the μPD78044F, μPD78044H, μPD780208, and μPD780228 subseries, the fastest requires two CPU clocks. Thus, the relationship between the CPU clock (f_{CPU}) and minimum instruction execution time is as shown in Table 3-3.

* **Table 3-3. Relationship between the CPU Clock and Minimum Instruction Execution Time**

CPU clock (f_{CPU})	Minimum instruction execution time: $2/f_{CPU}$
f_X	0.4 μs
$f_X/2$	0.8 μs
$f_X/2^2$	1.6 μs
$f_X/2^3$	3.2 μs
$f_X/2^4$	6.4 μs
f_{XT} **Note**	122 μs

Note Only for the μPD78044F, μPD78044H, and μPD780208 subseries

Remark f_X = 5.0 MHz, f_{XT} = 32.768 kHz
f_X: Oscillation frequency of the main system clock
f_{XT}: Oscillation frequency of the subsystem clock

Figure 3-3. Format of the Display Mode Register 0 (μPD78044F and μPD78044H Subseries)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
DSPM0	KSF	DSPM06	0	0	SEGS3	SEGS2	SEGS1	SEGS0	FFA0H	00H	R/W Note 1

SEGS3	SEGS2	SEGS1	SEGS0	Number of display segments
0	0	0	0	9
0	0	0	1	10
0	0	1	0	11
0	0	1	1	12
0	1	0	0	13
0	1	0	1	14
0	1	1	0	15
0	1	1	1	16
1	0	0	0	17
1	0	0	1	18
1	0	1	0	19
1	0	1	1	20
1	1	0	0	21
1	1	0	1	22
1	1	1	0	23
1	1	1	1	24

DSPM06	Mode setting for the noise eliminator of the subsystem clock Note 2
0	2.5 MHz < f_X ≤ 5.0 MHz
1	1.25 MHz < f_X ≤ 2.5 MHz

KSF	Timing status
0	Display timing
1	Key scan timing

Notes 1. Bit 7 (KSF) is read-only.

2. Specify a value in accordance with the oscillation frequency of the main system clock (f_X). The noise eliminator can be used during FIP display operation.

Remark f_X: Oscillation frequency of the main system clock

Figure 3-4. Format of the Display Mode Register 0 (μPD780208 Subseries) (1/2)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
DSPM0	KSF	DSPM06	DSPM05	SEGS4	SEGS3	SEGS2	SEGS1	SEGS0	FFA0H	00H	R/W

R/W

SEGS4	SEGS3	SEGS2	SEGS1	SEGS0	Number of display segments (display mode 1)	Number of display outputs (display mode 2)
0	0	0	0	0	9	9
0	0	0	0	1	10	10
0	0	0	1	0	11	11
0	0	0	1	1	12	12
0	0	1	0	0	13	13
0	0	1	0	1	14	14
0	0	1	1	0	15	15
0	0	1	1	1	16	16
0	1	0	0	0	17	17
0	1	0	0	1	18	18
0	1	0	1	0	19	19
0	1	0	1	1	20	20
0	1	1	0	0	21	21
0	1	1	0	1	22	22
0	1	1	1	0	23	23
0	1	1	1	1	24	24
1	0	0	0	0	25	25
1	0	0	0	1	26	26
1	0	0	1	0	27	27
1	0	0	1	1	28	28
1	0	1	0	0	29	29
1	0	1	0	1	30	30
1	0	1	1	0	31	31
1	0	1	1	1	32	32
1	1	0	0	0	33	33
1	1	0	0	1	34	34
1	1	0	1	0	35	35
1	1	0	1	1	36	36
1	1	1	0	0	37	37
1	1	1	0	1	38[Note]	38
1	1	1	1	0	39[Note]	39
1	1	1	1	1	40[Note]	40

Note If the total number of digits and segments exceeds 53, digits have precedence over segments.

Figure 3-4. Format of the Display Mode Register 0 (μPD780208 Subseries) (2/2)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
DSPM0	KSF	DSPM06	DSPM05	SEGS4	SEGS3	SEGS2	SEGS1	SEGS0	FFA0H	00H	R/W Note 1

R/W

DSPM05	Setting of display mode
0	Display mode 1 (segment/character type)
1	Display mode 2 (type that a segment extends two or more grids)

R/W

DSPM06	Mode setting for the noise eliminator of the subsystem clock Note 2
0	2.5 MHz < f_X ≤ 5.0 MHz
1	1.25 MHz < f_X ≤ 2.5 MHz Note 3

R

KSF	Timing status
0	Display timing
1	Key scan timing

Notes 1. Bit 7 (KSF) is read-only.

2. Specify a value in accordance with the oscillation frequency of the main system clock (f_X). The noise eliminator can be used during FIP display operation.

3. When f_X is used from above 1.25 MHz to 2.5 MHz, set 1 in DSPM06 before FIP display.

Remark f_X: Oscillation frequency of the main system clock

Figure 3-5. Format of the Display Mode Register 1 (μPD78044F and μPD78044H Subseries)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
DSPM1	DIGS3	DIGS2	DIGS1	DIGS0	DIMS3	DIMS2	DIMS1	DIMS0	FFA1H	00H	R/W

DIMS0	Display cycle selection
0	1024/f_X as 1 display cycle (One display cycle is 204.8 μs at 5.0 MHz.)
1	2048/f_X as 1 display cycle (One display cycle is 409.6 μs at 5.0 MHz.)

DIMS3	DIMS2	DIMS1	Cut width of the digit signal
0	0	0	1/16
0	0	1	2/16
0	1	0	4/16
0	1	1	6/16
1	0	0	8/16
1	0	1	10/16
1	1	0	12/16
1	1	1	14/16

DIGS3	DIGS2	DIGS1	DIGS0	Number of display digits
0	0	0	0	Display stop (static display)Note
0	0	0	1	2
0	0	1	0	3
0	0	1	1	4
0	1	0	0	5
0	1	0	1	6
0	1	1	0	7
0	1	1	1	8
1	0	0	0	9
1	0	0	1	10
1	0	1	0	11
1	0	1	1	12
1	1	0	0	13
1	1	0	1	14
1	1	1	0	15
1	1	1	1	16

Note When display is disabled, a port output latch can be operated to enable static display.

Remark f_X: Oscillation frequency of the main system clock

Figure 3-6. Format of the Display Mode Register 1 (μPD780208 Subseries)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
DSPM1	DIGS3	DIGS2	DIGS1	DIGS0	DIMS3	DIMS2	DIMS1	DIMS0	FFA1H	00H	R/W

DIMS0	Setting of display mode cycle
0	1024/f_X as 1 display cycle (One display cycle is 204.8 μs at 5.0 MHz.)
1	2048/f_X as 1 display cycle (One display cycle is 409.6 μs at 5.0 MHz.)

DIMS3	DIMS2	DIMS1	Cut width of the FIP output signal
0	0	0	1/16
0	0	1	2/16
0	1	0	4/16
0	1	1	6/16
1	0	0	8/16
1	0	1	10/16
1	1	0	12/16
1	1	1	14/16

DIGS3	DIGS2	DIGS1	DIGS0	Number of display digits (display mode 1) DSPM05 = 0	Number of display patterns (display mode 2) DSPM05 = 1
0	0	0	0	Disabled display (static display)Note	Disabled display (static display)Note
0	0	0	1	2	2
0	0	1	0	3	3
0	0	1	1	4	4
0	1	0	0	5	5
0	1	0	1	6	6
0	1	1	0	7	7
0	1	1	1	8	8
1	0	0	0	9	9
1	0	0	1	10	10
1	0	1	0	11	11
1	0	1	1	12	12
1	1	0	0	13	13
1	1	0	1	14	14
1	1	1	0	15	15
1	1	1	1	16	16

Note When display is disabled, a port output latch can be operated to enable static display.

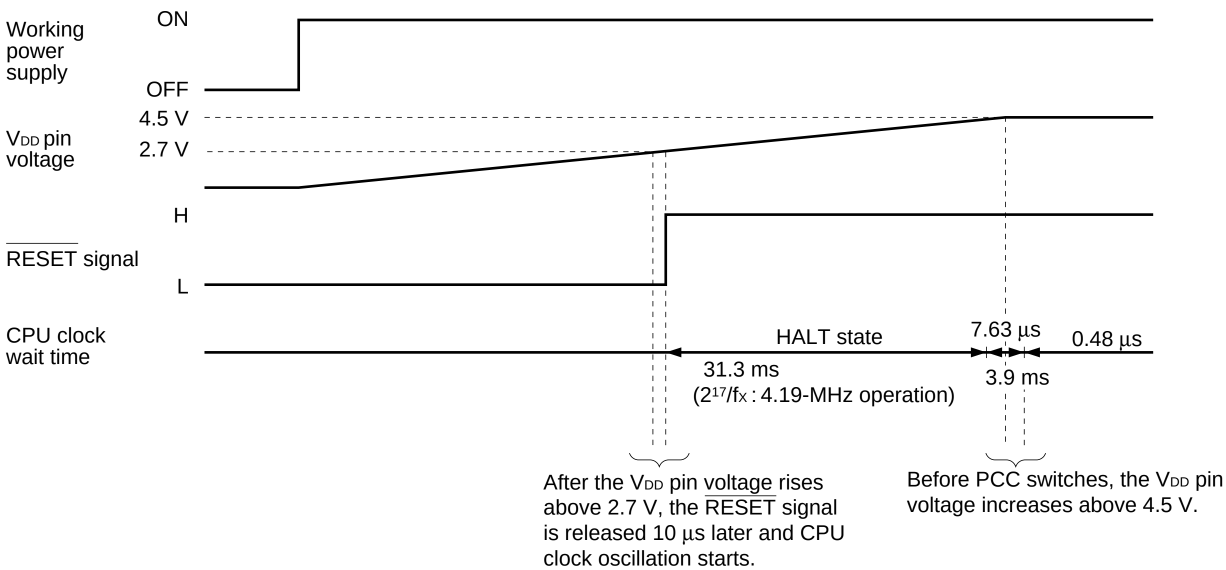
Remark f_X : Oscillation frequency of the main system clock
DSPM05 : Bit 5 of display mode register 0

3.1 SWITCHING PCC AFTER $\overline{\text{RESET}}$

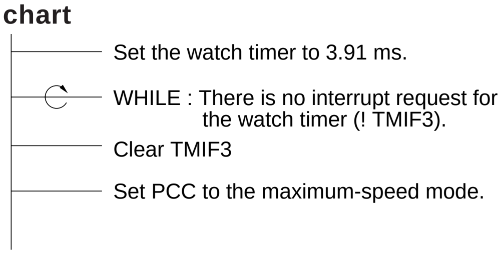
By issuing the $\overline{\text{RESET}}$ signal, the slowest mode (processor clock control register(PCC) = 04H) of main system clock is selected for the CPU clock. As a result, when running at the maximum speed, PCC is rewritten and the CPU clock is set to the maximum speed (PCC = 00H). However, in order to operate at the maximum speed mode, the V_{DD} pin voltage must be increased to the range where high-speed operation is possible and be stable.

In this example, the time until the voltage increase is awaited by the watch timer (3.91-ms interval period selected). After the wait, the CPU clock switches to the maximum speed.

Figure 3-7. CPU Clock Switching after $\overline{\text{RESET}}$ (μPD78044F Subseries)



(1) SPD chart



(2) Program listing

```
;********************************
;*       Wait setting
;********************************
      TCL2=#00010000B
      TMC2=#00110110B          ; Set the watch timer to 3.91 ms.
      while_bit(!TMIF3)        ; 3.91 ms?
      endw
      CLR1    WTIF
      PCC=#00000000B           ; Set the CPU clock to the maximum speed.
```

3.2 SWITCHING DURING POWER ON/OFF

The 78K/0 Series can select the subsystem clock based on the processor clock control register(PCC) setting and can operate with an ultralow power consumption. Consequently, by adding a backup power, such as a NiCd battery or super capacitor, to the system, operation can continue even when power fails.

In this example, by detecting whether the power is on or off in INTP1 (select detection edge by detecting both the rising and falling edges), the on or off decision is made based on this port level and PCC switches. Figure 3-8 shows an example circuit. Figure 3-9 shows the switching timing of the system clock.

Figure 3-8. Example of the System Clock Switching Circuit

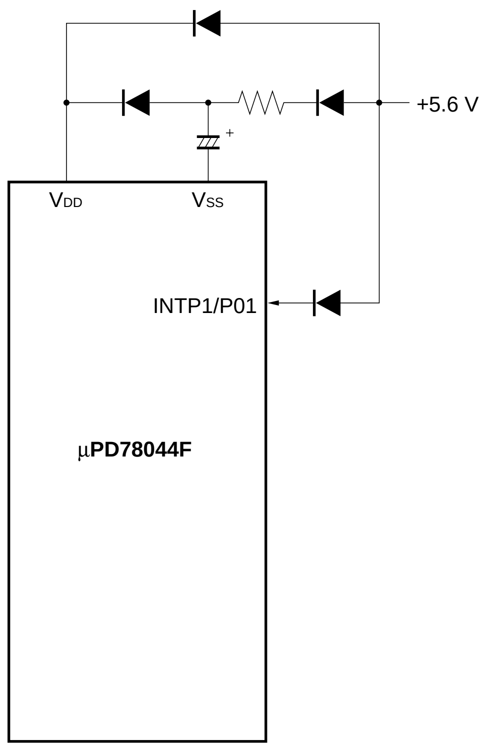
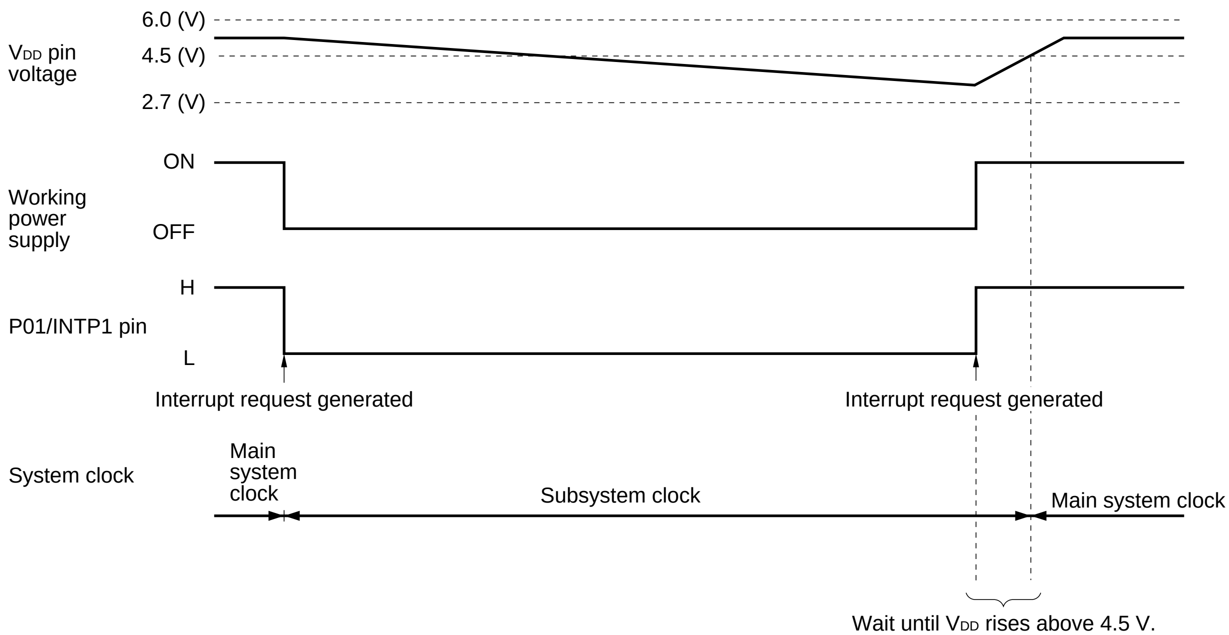
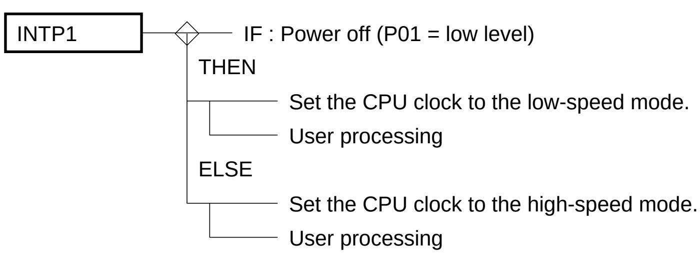


Figure 3-9. System Clock Switching during Power On and Off (μPD78044F Subseries)



(1) SPD chart



(2) Program listing

```
VEP0  CSEG  AT 08H
      DW    INTP1                    ; INTP1 vector address setting

      MOV   INTM0,#00110000B  ; Both edge detection mode
      CLR1  PMK1
      EI
;****************************************
;*   Low-speed/high-speed mode setting
;****************************************
INTP1:
       if_bit(!P0.1)
;          On-chip hardware setting (low speed)
;          User processing

          PCC=#10010000B             ; Set to low-speed mode.

       else
;          On-chip hardware setting (high speed)
;          User processing

          PCC=#00000000B             ; Set to high-speed mode.
   endif
   RETI
```

[MEMO]

CHAPTER 4 WATCHDOG TIMER APPLICATION

The watchdog timer in the 78K/0 Series has the two functions of a watchdog timer mode to detect runaway operation of the microcontroller and an interval timer mode.

The watchdog timer is set by timer clock selection register 2 (TCL2), watchdog timer mode register (WDTM), and watchdog timer clock selection register (WDCS).

* **Cautions 1. WDCS is incorporated into the μPD780228 subseries only.**

* **2. The format of the registers incorporated into the μPD780228 subseries differs from that of the registers incorporated into the μPD78044F, μPD78044H, and μPD780208 subseries. When using any of the sample programs described in this chapter with the μPD780228 subseries, replace the register settings with those for the μPD780228 subseries.**

Figure 4-1. Format of Timer Clock Selection Register 2 (μPD78044F, μPD78044H, and μPD780208 Subseries)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
TCL2	TCL27	TCL26	TCL25	TCL24	0	TCL22	TCL21	TCL20	FF42H	00H	R/W

TCL22	TCL21	TCL20	Count clock selection	
			Watchdog timer mode	Interval timer mode
0	0	0	$f_X/2^3$ (625 kHz)	$f_X/2^4$ (313 kHz)
0	0	1	$f_X/2^4$ (313 kHz)	$f_X/2^5$ (156 kHz)
0	1	0	$f_X/2^5$ (156 kHz)	$f_X/2^6$ (78.1 kHz)
0	1	1	$f_X/2^6$ (78.1 kHz)	$f_X/2^7$ (39.1 kHz)
1	0	0	$f_X/2^7$ (39.1 kHz)	$f_X/2^8$ (19.5 kHz)
1	0	1	$f_X/2^8$ (19.5 kHz)	$f_X/2^9$ (9.8 kHz)
1	1	0	$f_X/2^9$ (9.8 kHz)	$f_X/2^{10}$ (4.9 kHz)
1	1	1	$f_X/2^{11}$ (2.4 kHz)	$f_X/2^{12}$ (1.2 kHz)

TCL24	Count clock selection for the watch timer[Note]
0	$f_X/2^8$ (19.5 kHz)
1	f_{XT} (32.768 kHz)

TCL27	TCL26	TCL25	Selection of the buzzer output frequency
0	x	x	Buzzer output prohibited
1	0	0	$f_X/2^{10}$ (4.9 kHz)
1	0	1	$f_X/2^{11}$ (2.4 kHz)
1	1	0	$f_X/2^{12}$ (1.2 kHz)
1	1	1	Setting prohibited

Note When a main system clock at 1.25 MHz or lower and an FIP controller/driver are used simultaneously, select $f_X/2^8$ as the count clock for the watch timer.

Caution When TCL2 will be rewritten with data other than identical data, rewrite after temporarily stopping timer operation.

Remarks 1. f_X : Main system clock oscillation frequency

2. f_{XT} : Subsystem clock oscillation frequency

3. x : Don't care

4. The values in parentheses apply to operation with f_X = 5.0 MHz or f_{XT} = 32.768 kHz.

*

Figure 4-2. Format of the Watchdog Timer Mode Register (μPD78044F, μPD78044H, and μPD780208 Subseries)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
WDTM	RUN	0	0	WDTM4	WDTM3	0	0	0	FFF9H	00H	R/W

WDTM4	WDTM3	Operating mode selection for the watchdog timer[Note 1]
0	x	Interval timer mode[Note 2] (During overflow, a maskable interrupt request is issued.)
1	0	Watchdog timer mode 1 (During overflow, a non-maskable interrupt request is issued.)
1	1	Watchdog timer mode 2 (During overflow, reset operation starts.)

RUN	Selection of watchdog timer operation[Note 3]
0	Stop count
1	After clearing the counter, start the count.

Notes 1. When WDTM3 and WDTM4 are set to 1 once, they cannot be cleared to 0 by software.

2. In this mode, the watchdog timer start operating as an interval timer immediately after RUN is set to 1.

3. When RUN is set once to 1, it cannot be cleared to 0 by software. As a result, when the count starts, stopping by means other than $\overline{\text{RESET}}$ input is not possible.

Cautions 1. When RUN is set to 1 and the watchdog timer work was cleared, the period of an actual overflow becomes a maximum of 0.5% shorter than the time set in timer clock selection register 2.

2. When watchdog timer mode 1 or 2 is being used, check that the interrupt request flag (TMIF4) is set to 0 and set WDTM4 to 1.
If WDTM4 is set to 1 while TMIF4 is set to 1, a non-maskable interrupt request occurs regardless of the contents of WDTM3.

Remark x: Don't care

* **Figure 4-3. Format of the Watchdog Timer Mode Register (μPD780228 Subseries)**

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
WDTM	RUN	0	0	WDTM4	WDTM3	0	0	0	FFF9H	00H	R/W

WDTM4	WDTM3	Operating mode selection for the watchdog timer[Note 1]
0	x	Interval timer mode (During overflow, a maskable interrupt request is issued.)
1	0	Watchdog timer mode 1 (During overflow, a non-maskable interrupt request is issued.)
1	1	Watchdog timer mode 2 (During overflow, reset operation starts.)

RUN	Selection of watchdog timer operation[Note 2]
0	Stop count
1	After clearing the counter, start the count.

Notes 1. When WDTM3 and WDTM4 are set to 1 once, they cannot be cleared to 0 by software.

2. When RUN is set once to 1, it cannot be cleared to 0 by software. As a result, when the count starts, stopping by means other than $\overline{\text{RESET}}$ input is not possible.

Caution When RUN is set to 1 and the watchdog timer work was cleared, the period of an actual overflow becomes a maximum of 0.5% shorter than the set time.

Remark x: Don’t care

*

Figure 4-4. Format of the Watchdog Timer Clock Selection Register (Only for the μPD780228 Subseries)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
WDCS	0	0	0	0	0	WDCS2	WDCS1	WDCS0	FF42H	00H	R/W

WDCS2	WDCS1	WDCS0	Overflow time of the watchdog/interval timer
0	0	0	$2^{12}/f_X$ (819 μs)
0	0	1	$2^{13}/f_X$ (1.64 ms)
0	1	0	$2^{14}/f_X$ (3.28 ms)
0	1	1	$2^{15}/f_X$ (6.55 ms)
1	0	0	$2^{16}/f_X$ (13.1 ms)
1	0	1	$2^{17}/f_X$ (26.2 ms)
1	1	0	$2^{18}/f_X$ (52.4 ms)
1	1	1	$2^{20}/f_X$ (210 ms)

Remarks 1. f_X: Oscillation frequency of the main system clock

2. The values in parentheses apply to operation with f_X = 5.0 MHz.

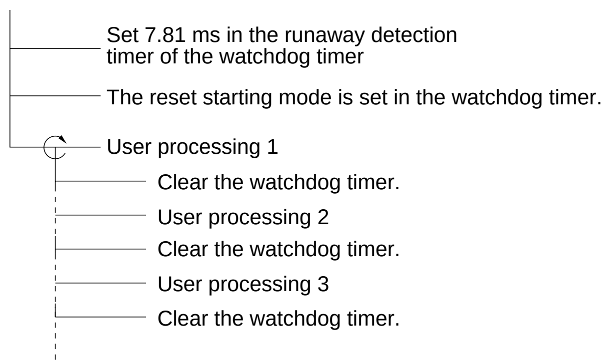
4.1 SETTING THE WATCHDOG TIMER MODE

In processing operation of the watchdog timer after detecting the runaway, there is reset processing or non-maskable interrupt servicing. Either one can be selected by the watchdog timer mode register (WDTM). When the watchdog timer mode is used, the timer must be cleared in a time interval shorter than the set runaway detection time. When the timer is not cleared, an overflow occurs and reset or interrupt servicing is executed.

The runaway detection time for the watchdog timer is set in timer clock selection register 2(TCL2).

In this example, 7.81 ms is selected in the runaway detection time and reset processing operation is selected when an overflow occurs.

(1) SPD chart



(2) Program listing

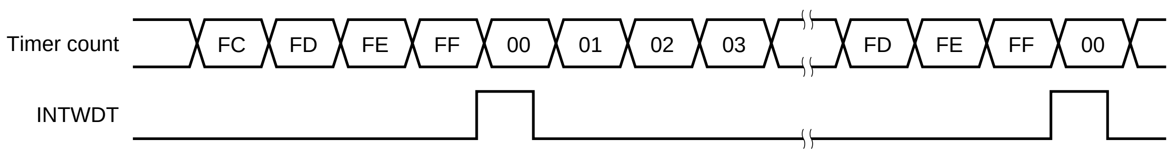
```
;*********************************
;*    Watchdog timer setting
;*********************************

      TCL2=#00000100B          ; Set the watchdog timer to 7.81 ms.
      WDTM=#10011000B          ; Set the reset start mode.
            |
;     User processing 1
            |
      SET1    RUN              ; Timer clear
            |
;     User processing 2
            |
      SET1    RUN              ; Timer clear
            |
;     User processing 3
            |
      SET1    RUN              ; Timer clear
            |
```

4.2 INTERVAL TIMER MODE SETTING

When the interval timer mode is used, the interval time is set in timer clock selection register 2(TCL2) (interval time = 977 µs to 250 ms at f_X = 4.19 MHz). This interval timer sets the interrupt request flag (TMIF4) when the timer overflows.

In this example, setting the three times of 977 µs, 7.82 ms, and 250 ms is illustrated.

Figure 4-5. Count Timing of the Watchdog Timer



(1) Program listing

<1> Setting 977 µs

```
TCL2 = #00000000B    ; Set to 977 µs.
WDTM = #10001000B    ; Select the interval timer mode.
```

<2> Setting 7.82 ms

```
TCL2 = #00000011B    ; Set to 7.82 ms.
WDTM = #10001000B    ; Select the interval timer mode.
```

<3> Setting 250 ms

```
TCL2 = #00000111B    ; Set to 250 ms.
WDTM = #10001000B    ; Select the interval timer mode.
```

CHAPTER 5 16-BIT TIMER/EVENT COUNTER APPLICATION

The 16-bit timer/event counter in the 78K/0 Series supports the following functions:

- Interval timer
- PWM output
- Pulse width measurement
- External event counter
- Square wave output

The 16-bit timer/event counter requires the setting of the following six registers:

- Timer clock selection register 0 (TCL0)
- 16-bit timer mode control register (TMC0)
- 16-bit timer output control register (TOC0)
- Port mode register 3 (PM3)
- External interrupt mode register (INTM0)
- Sampling clock selection register (SCS)

Figure 5-1. Format of Timer Clock Selection Register 0

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
TCL0	CLOE	TCL06	TCL05	TCL04	TCL03	TCL02	TCL01	TCL00	FF40H	00H	R/W

TCL03	TCL02	TCL01	TCL00	PCL output clock selection
0	0	0	0	f_{XT} (32.768 kHz)
0	1	1	1	$f_X/2^3$ (625 kHz)
1	0	0	0	$f_X/2^4$ (313 kHz)
1	0	0	1	$f_X/2^5$ (156 kHz)
1	0	1	0	$f_X/2^6$ (78.1 kHz)
1	0	1	1	$f_X/2^7$ (39.1 kHz)
1	1	0	0	$f_X/2^8$ (19.5 kHz)
Other than the above				Setting prohibited

TCL06	TCL05	TCL04	Selection of the count clock of the 16-bit timer register
0	0	0	TI0 (valid edge settable)
0	0	1	f_X (5.0 MHz)
0	1	0	$f_X/2$ (2.5 MHz)
0	1	1	$f_X/2^2$ (1.25 MHz)
1	0	0	$f_X/2^3$ (625 kHz)
Other than the above			Setting prohibited

CLOE	PCL output control
0	Output prohibited
1	Output enabled

Cautions 1. Setting the valid edge for the TI0/INTP0 pin is performed by the external interrupt mode register (INTM0). In addition, selecting the frequency of the sampling clock is performed by the sampling clock selection register (SCS).

2. After setting TCL00 to TCL03 when PCL output is enabled, set 1 in CLOE by using a 1-bit memory manipulation instruction.

3. When the TM0 count clock is TI0 and the count value is read, read from TM0 and not from the capture register (CR01).

4. When data other than identical data will be rewritten in TCL0, rewrite after temporarily stopping timer operation.

Remarks 1. f_X : Main system clock oscillation frequency

2. f_{XT} : Subsystem clock oscillation frequency

3. TI0 : Input pin of the 16-bit timer/event counter

4. TM0: 16-bit timer register

5. The values in parentheses apply to operation with f_X = 5.0 MHz or f_{XT} = 32.768 kHz.

Figure 5-2. Format of the 16-Bit Timer Mode Control Register

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
TMC0	0	0	0	0	TMC03	TMC02	TMC01	OVF0	FF48H	00H	R/W

OVF0	Overflow detection of the 16-bit timer register
0	No overflow
1	Overflow

TMC03	TMC02	TMC01	Selection of operating mode and clear mode	Selection of TO0 output timing	Interrupt request generation
0	0	0	Stop operation (Clear TM0 to 0.)	No change	Not generated
0	0	1	PWM mode (free running)	PWM pulse output	Generated when TM0 and CR00 match
0	1	0	Free running mode	TM0 and CR00 match.	
0	1	1		TM0 and CR00 match or valid edge occurs at TI0.	
1	0	0	When there is a valid edge at TI0, clear and start.	TM0 and CR00 match.	
1	0	1		TM0 and CR00 match or valid edge occurs at TI0.	
1	1	0	When TM0 and CR00 match, clear and start.	TM0 and CR00 match.	
1	1	1		TM0 and CR00 match or valid edge occurs at TI0.	

Cautions 1. Perform switching of the clear mode and TO0 output timing after timer operation is stopped (Set 000 in TMC01-TMC03.)

2. Setting the valid edge of the TI0/INTP0 pin is performed by the external interrupt mode register (INTM0). In addition, the sampling clock frequency is specified in the sampling clock selection register (SCS).

3. When PWM mode is used, after setting the PWM mode, set the data in CR00.

4. When TM0 and CR00 matched and the mode to clear and start was selected, the CR00 setting is FFFFH. When the value in TM0 changes from FFFFH to 0000H, the OVF0 flag is set to 1.

5. The 16-bit timer register begins operating when a value other than 000 (operation stop mode) is set in TMC01-TMC03. To stop the operation, set 000 in TMC01-TMC03.

Remarks 1. TO0 : Output pin of the 16-bit timer/event counter
2. TI0 : Input pin of the 16-bit timer/event counter
3. TM0 : 16-bit timer register
4. CR00 : Compare register 00

Figure 5-3. Format of the 16-Bit Timer Output Control Register

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
TOC0	0	0	0	0	LVS0	LVR0	TOC01	TOE0	FF4EH	00H	R/W

TOE0	Output control of the 16-bit timer/event counter
0	Output prohibited (port mode)
1	Output enabled

TOC01	PWM mode	Mode other than PWM mode
	Selection of active level	Control of timer output flip-flop
0	Active high	Inverse operation prohibited
1	Active low	Inverse operation enabled

LVS0	LVR0	Setting the state of the timer output flip-flop of the 16-bit timer/event counter
0	0	No change
0	1	Reset the timer output flip-flop (0)
1	0	Set the timer output flip-flop (1)
1	1	Setting prohibited

Cautions 1. Always set TOC0 after timer operation has stopped.
2. 0 is read from LVS0 and LVR0 when read after setting data.

Figure 5-4. Format of the Port Mode Register 3

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
PM3	PM37	PM36	PM35	PM34	PM33	PM32	PM31	PM30	FF23H	FFH	R/W

PM3n	I/O mode selection of pin P3n (n = 0 to 7)
0	Output mode (output buffer on)
1	Input mode (output buffer off)

Caution When the P30/TO0 pin is used for timer output, set 0 in the output latches of PM30 and P30.

Figure 5-5. Format of the External Interrupt Mode Register

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
INTM0	ES31	ES30	ES21	ES20	ES11	ES10	0	0	FFECH	00H	R/W

ES11	ES10	Valid edge selection for INTP0
0	0	Falling edge
0	1	Rising edge
1	0	Setting prohibited
1	1	Both rising and falling edges

ES21	ES20	Valid edge selection for INTP1
0	0	Falling edge
0	1	Rising edge
1	0	Setting prohibited
1	1	Both rising and falling edges

ES31	ES30	Valid edge selection for INTP2
0	0	Falling edge
0	1	Rising edge
1	0	Setting prohibited
1	1	Both rising and falling edges

* **Caution Set the valid edge of the INTP0/TI0/P00 pin after timer operation is stopped by setting 0 in bits 1 to 3 (TMC01 to TMC03) of the 16-bit timer mode control register (TMC0).**

Remarks 1. The INTP0 pin also acts as the TI0/P00 pin.

2. The INTP3 pin use the falling edge only.

Figure 5-6. Format of the Sampling Clock Selection Register

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
SCS	0	0	0	0	0	0	SCS1	SCS0	FF47H	00H	R/W

SCS1	SCS0	Selection of the INTP0 sampling clock
0	0	$f_X/2^{N+1}$
0	1	Setting prohibited
1	0	$f_X/2^6$ (78.1 kHz)
1	1	$f_X/2^7$ (39.1 kHz)

Caution **$f_X/2^{N+1}$ is the clock supplied to the CPU. $f_X/2^6$ and $f_X/2^7$ are the clocks supplied to peripheral hardware. $f_X/2^{N+1}$ stops in the HALT mode.**

Remarks 1. N: Value (N = 0 to 4) set in bits 0 to 2 (PCC0 to PCC2) in the processor clock control register (PCC)

2. f_X: Main system clock oscillation frequency

3. The values in parentheses apply to operation with f_X = 5.0 MHz.

5.1 INTERVAL TIMER SETTING

When the interval timer is used, first the timer clock selection register (TCL0) and 16-bit timer mode control register (TMC0) are set. The clear mode of the 16-bit timer is set in TMC0. The interval time is set in TCL0.

Then, the setting time and the compare register (CR00) from the count clock are set. The setting time is set by the following procedure.

Setting-time = (compare-register-value + 1) x count-clock-period

This example illustrates how to set the setting time of interval timer to 10 ms and 50 ms.

(a) For a 10 ms interval

<1> TMC0 setting

Select clear and start when TM0 and CR00 match.

<2> TCL0 setting

A setting greater than 10 ms is possible and the f_X mode with the highest resolution is selected.

<3> CR00 setting

$$10 \text{ ms} = (N + 1) \times \frac{1}{4.19 \text{ MHz}}$$

N = 10 ms x 4.19 MHz – 1 = 41899

(1) Program listing

```
CR00=#41899
TCL0=#00010000B ; Select the count clock fX.
TMC0=#00001100B ; The 16-bit timer/event counter is set to clear and start when TM0 and CR00
                  match.
```

(b) For a 50-ms interval

<1> TMC0 setting

Select clear and start when TM0 and CR00 match.

<2> TCL0 setting

A setting greater than 50 ms is possible and the $f_X/2^2$ mode with the highest resolution is selected.

<3> CR00 setting

$$50 \text{ ms} = (N + 1) \times \frac{1}{4.19 \text{ MHz}/2^2}$$

$$N = 50 \text{ ms} \times 4.19 \text{ MHz}/2^2 - 1 \fallingdotseq 52374$$

(1) Program listing

```
CR00=#52374
TCL0=#00110000B ; Select the count clock fX/2^2.
TMC0=#00001100B ; The 16-bit timer/event counter is set to clear and start when TM0 and CR00
                  match.
```

5.2 PWM OUTPUT

When the PWM output is used, set the PWM mode in the 16-bit timer mode control register (TMC0) and the 16-bit timer/event counter in the output enabled state in the 16-bit timer output control register (TOC0).

The PWM pulse width (active level) is determined by the value set in CR00. However, because PWM in the 78K/0 Series has 14-bit resolution, bits 2 to 15 become valid in the compare register (CR00). (Set bits 0 and 1 in CR00 to 0.)

In this example, the basic period of the PWM mode is set to 61.0 µs ($2^8/f_X$) and the active level is set to active-low. Also, the pulse width setting program rewrites the high-order 4 bits based on a parameter (00H to 0FH). Consequently, this application example can have a PWM output in 16 steps (CR00 = 0FFCH to FFFCH).

(1) Package description

<Symbols declared as public>

PWM : PWM output subroutine name
PWMOUT : Input parameter of PWM active level

<Registers used>

AX

<RAM used>

Name	Use	Attribute	Byte
PWMOUT	PWM active-level setting	SADDR	1

<Nesting>

1 level, 2 bytes

<Hardware used>

- 16-bit timer/event counter
- P30/TO0

<Initial settings>

- 16-bit timer/event counter setting
  - PWM output mode TMC0=#00000010B
  - Basic PWM period of 61.0 µs TCL0=#00010000B
  - Active-low output TOC0=#00000011B
- P30 output mode PM30=0
- P30 output latch P30=0

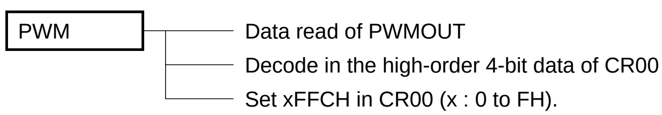
<Startup procedure>

After setting data in PWMOUT of the RAM, call the subroutine PWM.

(2) Use example

```
EXTRN  PWM,PWMOUT
   :
TOC0=#00000011B     ; Setting PWM output and active-low
TCL0=#00010000B     ; Select the count clock fX.
TMC0=#00000010B     ; PWM mode setting
   :
PWMOUT=A            ; Input parameter setting of active level
CALL   !PWM
```

(3) SPD chart



(4) Program listing

```
        PUBLIC PWM,PWMOUT
PWM_DAT DSEG   SADDR
PWMOUT: DS     1                    ; PWM output data area (0 to 15)
;************************************
;*     PWM output (16 levels)
;************************************
PO_SEG CSEG
PWM:
        A=PWMOUT                    ; Read high-order data of PWMOUT
        A<<=1
        A<<=1
        A<<=1
        A<<=1
        A!=#0FH                     ; Set low-order 12 bits in 0FFCH.
        X=#0FCH
        CR00=AX
        RET
```

5.3 REMOTE CONTROL RECEPTION

Two examples of programs are introduced for remote control reception using the 16-bit timer/event counter.

- The counter is cleared when a valid edge is detected by the remote control. The pulse width from the timer count value (capture register CR01) is measured until the next valid edge is detected.
- The timer is allowed to run freely and the pulse width is measured from the difference in the counter between valid edges. In addition, this is synchronized to the PWM output.

The remote control signal is received by a PIN light receiving diode, introduced to the µPC1490 receiving preamplifier for remote control and input at pin P00/INTP0. An example remote control circuit is shown in Figure 5-7. The format of the remote control signal is shown in Figure 5-8.

Figure 5-7. Example of the Remote Control Receiving Circuit

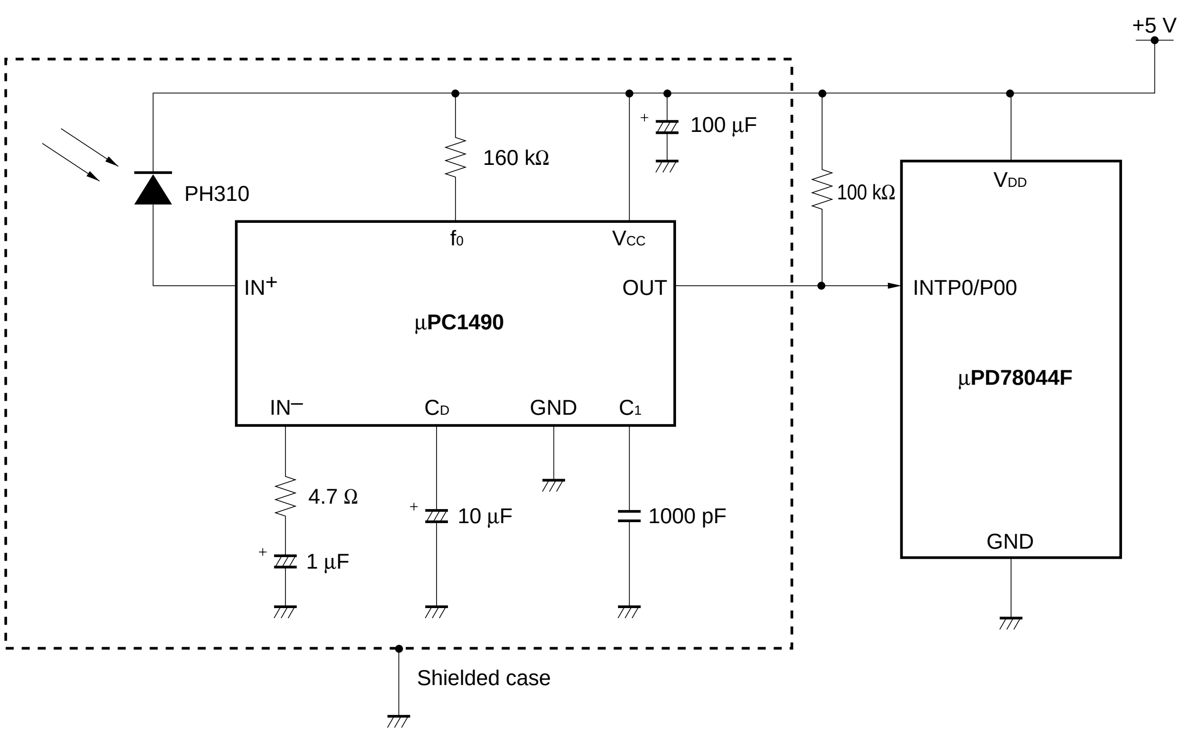
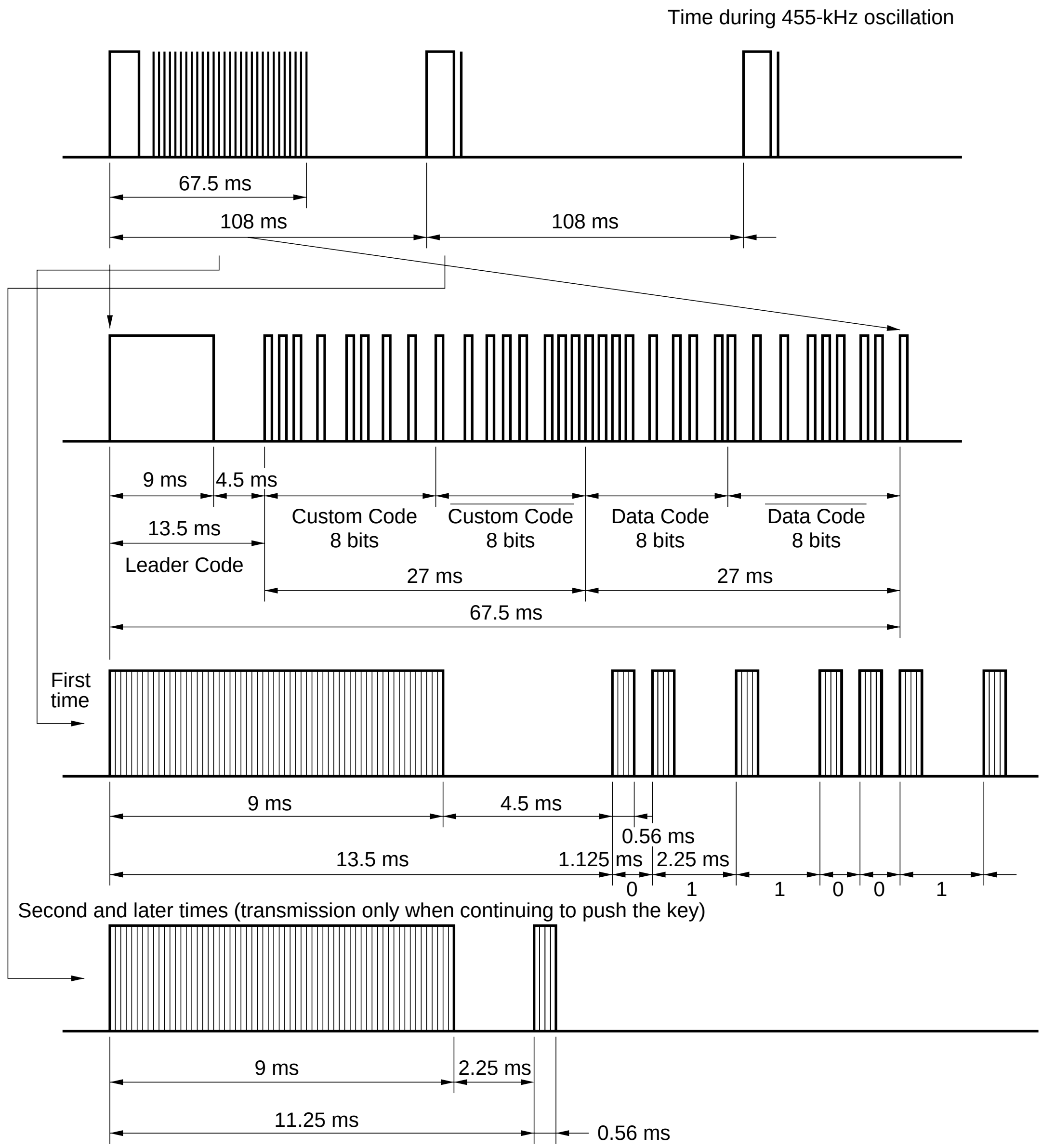
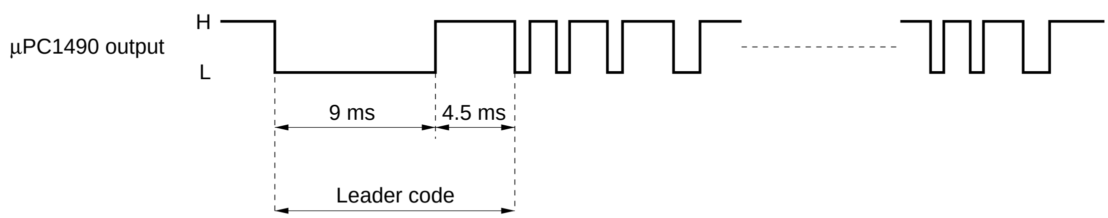


Figure 5-8. IC Output Signal for Remote Control Transmission



Because the μPC1490 preamplifier for remote control reception used in this circuit example is active-low, the level inputs to the μPD78044F subseries become inverted data of the data transmitted by the remote control.

Figure 5-9. Output Signal of the Receiving Preamplifier



5.3.1 Remote Control Reception by a Counter Clear

In this program, the valid pulse width when receiving a remote control signal is shown in Table 5-1 and the processing for each signal is described in **<1>** to **<6>**. The repeat signal of the remote control signal is valid for only the 250 ms following a valid input. Also, when a signal is input within 3 ms after a normal read, data is also invalid.

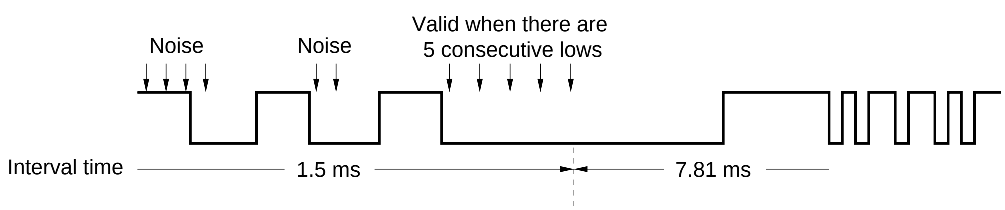
Table 5-1. Valid Time for Input Signal

Signal name		Output time	Valid time
Leader code (low)		9 ms	6.8 ms-11.8 ms
Leader code (high)	Normal	4.5 ms	3 ms-5 ms
	Repeat	2.25 ms	1.8 ms-3 ms
Custom code/data code	0	1.125 ms	0.5 ms-1.8 ms
	1	2.25 ms	1.8 ms-2.5 ms

<1> Leader code (low)

The interval of the 16-bit timer/event counter is set to 1.5 ms and port level sampling is performed by interrupt servicing. When the low-level input is detected five consecutive times, a leader code is judged to be present and the interval changes to 7.81 ms. Then, by having an interrupt request at the rising edge of INTP0, the low-level pulse width of the leader code is measured.

Figure 5-10. Sampling the Remote Control Signal



<2> Leader code (high)

Based on an interrupt request at the falling edge at INTP0, the high-level pulse width of the leader code is measured by the timer counter.

<3> Custom/data code

Based on an interrupt request at the falling edge at INTP0, the pulse width is measured at every bit (1 period). After the 32nd bit of data is read in, a match of the inverted data and custom code is tested. Furthermore, the absence of data at the 33rd bit is verified.

<4> Repeat code detection

When the high level of the leader code is less than 3 ms, the pulse width is measured until a rising edge occurs at INTP0 after the leader code is output.

<5> Valid period of the repeat code

After valid data is input, there is sampling by interrupt servicing of the 16-bit timer/event counter (1.5 ms interval) and the valid period of 250 ms for the repeat code is measured.

<6> Time out during pulse width measurement

When an interrupt request (7.81 ms) of the 16-bit timer/event counter occurred during pulse width measurement, a time out occurs and the data become invalid.

(1) Package description

<Symbols declared as public>

RMDATA : Saves remote control reception data
RPT : Decision flag for the repeat valid interval
IPDTFG : Decision flag indicating the presence of valid data
RMDTOK : Decision flag indicating the presence of a valid input signal
RMDTSET : Decision flag indicating the presence of an input signal

<Registers used>

Bank 0: AX, BC, HL

<RAM used>

Name	Use	Attribute	Byte
RPTCT	Repeat code valid time counter	SADDR	1
RMENDCT	No input time counter after data input		
SELMOD	Mode selection		
LD_CT	Leader signal detection counter		
RMDATA	Valid data storage area		
WORKP	Input signal storage area	SADDRP	4

<Flags used>

Name	Use
IPDTFG	Presence of valid data
RMDTOK	Presence of a valid input signal
RMDTSET	Presence of an input signal
RPT	Decision on whether the repeat valid interval has elapsed

<Nesting>

5 levels, 12 bytes

<Hardware used>

- 16-bit timer/event counter
- P00/INTP0

<Initial settings>

- 16-bit timer/event counter setting
  - Time clear mode when TM0 and CR00 match — TMC0 = #00001100B
  - Count/clock f_X — TCL0 = #00010000B
  - Compare register 00 — CR00 = #6290
- INTP0 sampling clock $f_X/2^7$ — SCS = #00000011B
- INTP0 high-priority interrupt request — PPR0 = 0
- 16-bit timer/event counter interrupt enabled — TMMK0 = 0
- Define custom code in CSTM. This is a public declaration.
- RAM clear

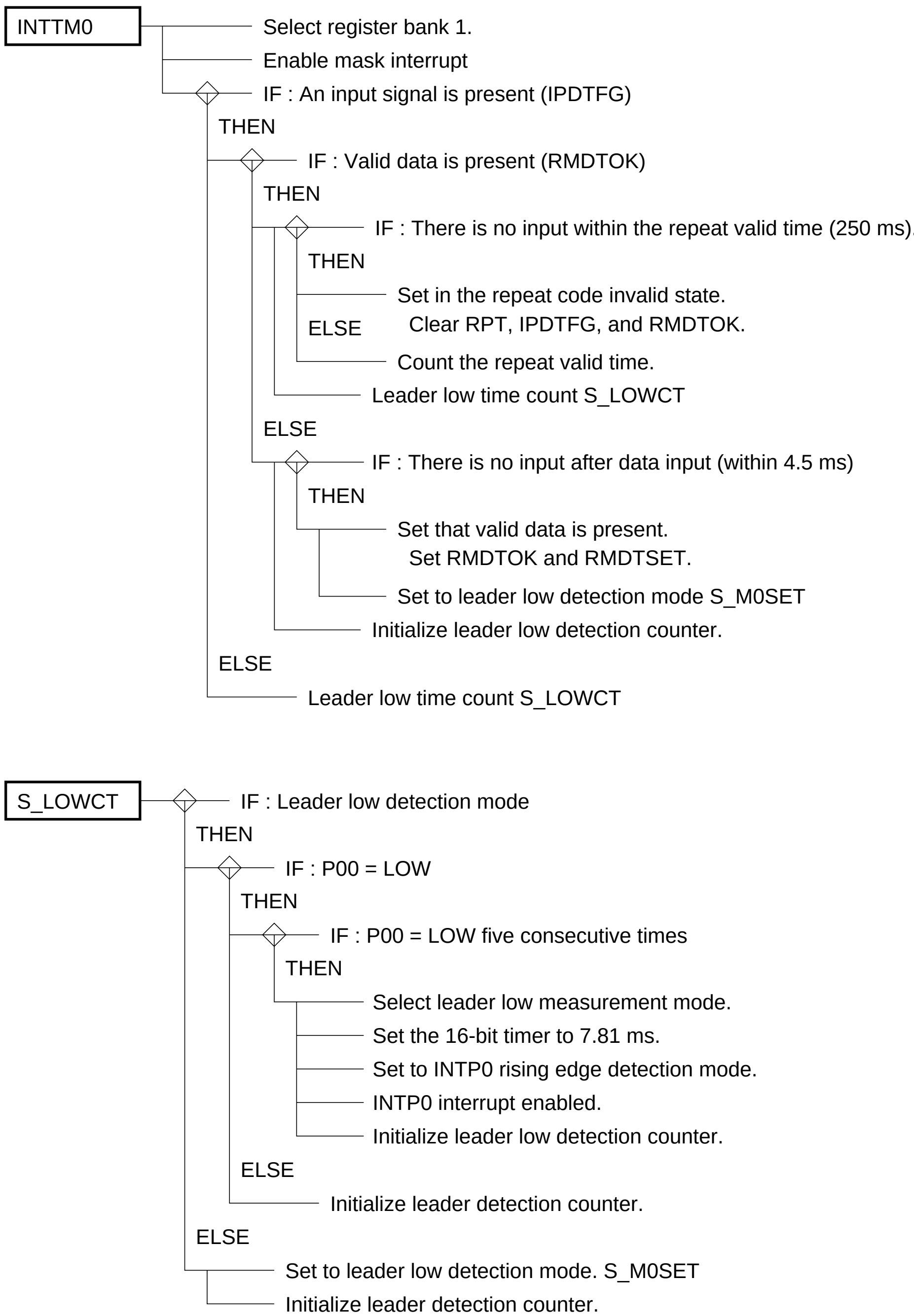
<Startup procedure>

Start using the INTP0 and INTTM0 interrupt requests.

(2) Example use

```
      PUBLIC CSTM
      EXTRN  RMDATA,RPTCT
      EXTBIT RPT,RMDTSET,IPDTFG

CSTM  EQU    9DH                  ; Remote control custom code

      CR00=#6290
      TCL0=#00010000B             ; Set to 1.5 ms.
      TMC0=#00001100B
      SCS=#00000011B              ; INTP0 sampling clock is fX/128.

      CLR1   PPR0                 ; High priority INTP0
      CLR1   RPT                  ; Clear flag
      CLR1   IPDTFG
      CLR1   RMDTSET

      CLR1   TMMK0                ; Enable timer interrupt
      EI

DT_TEST:
      if_bit(RMDTSET)
          CLR1   RMDTSET
          if_bit(RPT)
;
;             Repeat processing
;
          else
;
;             Input present processing
;
          endif
      else
          if_bit(!RPT)
;
;             No input present processing
;
          endif
      endif
```

(3) SPD chart


INTTM0
Select register bank 1.
Enable mask interrupt
IF : An input signal is present (IPDTFG)
THEN
IF : Valid data is present (RMDTOK)
THEN
IF : There is no input within the repeat valid time (250 ms).
THEN
Set in the repeat code invalid state.
Clear RPT, IPDTFG, and RMDTOK.
ELSE
Count the repeat valid time.
Leader low time count S_LOWCT
ELSE
IF : There is no input after data input (within 4.5 ms)
THEN
Set that valid data is present.
Set RMDTOK and RMDTSET.
Set to leader low detection mode S_M0SET
Initialize leader low detection counter.
ELSE
Leader low time count S_LOWCT
S_LOWCT
IF : Leader low detection mode
THEN
IF : P00 = LOW
THEN
IF : P00 = LOW five consecutive times
THEN
Select leader low measurement mode.
Set the 16-bit timer to 7.81 ms.
Set to INTP0 rising edge detection mode.
INTP0 interrupt enabled.
Initialize leader low detection counter.
ELSE
Initialize leader detection counter.
ELSE
Set to leader low detection mode. S_M0SET
Initialize leader detection counter.

```
INTPO
  Select register bank 0.
  100-µs wait WAIT
  CASE : SELMOD
    OF : 1
      Leader low measurement mode LEAD_L
    OF : 2
      Leader high measurement mode LEAD_H
    OF : 3
      Custom code/data read mode CDCODE
    OF : 4
      Repeat code detection mode REPCD
    OF : 5
      Error data detection mode ENDCHK
```

```
LEAD_L
  IF : P00 = HIGH
  THEN
    100-µs wait WAIT
    IF : P00 = HIGH
    THEN
      Timer read CR_READ
      IF : 6.8 ms ≤ leader low ≤ 11.8 ms
      THEN
        Select leader high detection mode.
        Set to INTP0 falling edge detection mode.
      ELSE
        Set to leader low detection mode. S_M0SET
```

```
LEAD_H
  IF : P00 = LOW
  THEN
    100-µs wait WAIT
    IF : P00 = LOW
    THEN
      Timer read CR_READ
      IF : 2 ms ≤ leader high ≤ 5 ms
      THEN
        IF : leader high ≥ 3 ms
        THEN
          Select the custom code/data read mode.
          Initialize the data storage area.
        ELSE
          Select the repeat detection mode.
          Set the INTP0 rising edge detection mode.
      ELSE
        Set to leader low detection mode. S_M0SET
```

CDCODE

- IF : P00 = LOW
  - THEN
    - 100-µs wait WAIT
    - IF : P00 = LOW
      - THEN
        - Timer read CR_READ
        - IF : 0.5 ms < input data ≤ 2.5 ms
          - THEN
            - IF : Input data ≥ 1.8 ms
              - THEN
                - Set CY.
              - ELSE
                - Clear CY.
            - Save CY in the data storage area.
            - IF : End of 32-bit data input
              - THEN
                - IF : Custom code match
                  - THEN
                    - IF : Match of inversed data of custom/data code
                      - THEN
                        - Save data code.
                        - Set in the input data present state.
                          Set IPDTFG and clear RMDTSET, RPT, and RMDTOK.
                        - Set to error data detection mode. S_M5SET
                      - ELSE
                        - Set to leader low detection mode. S_M0SET
                  - ELSE
                    - Set to leader low detection mode. S_M0SET
          - ELSE
            - Set to leader low detection mode. S_M0SET

REPCD

- IF : P00 = HIGH
  - THEN
    - 100-µs wait WAIT
    - IF : P00 = HIGH
      - THEN
        - IF : Valid data is present.
          - THEN
            - Timer read CR_READ
            - IF : Repeat code ≤ 1 ms
              - THEN
                - Set in the repeat code valid state.
                  Set RPT.
                - Set in the end of data input state.
                - Set to error data detection mode. S_M5SET
              - ELSE
                - Set to leader low detection mode. S_M0SET
          - ELSE
            - Set to error data detection mode. S_M5SET

ENDCHK
- IF : P00 = LOW
  THEN
  - 100-µs wait WAIT
  - IF : P00 = LOW
    THEN
    - Set in the invalid input signal state.
      Clear IPDTFG and RPT.
    - Set to leader low detection mode. S_M0SET

CR_READ
- Read capture register.
- Stop operation of 16-bit timer.
- Start timer.

S_M0SET
- Select the leader low detection mode.
- INTP0 interrupt prohibited
- Set the 16-bit timer to 1.5 ms

S_M5SET
- Select the error data detection mode.
- Set the counter for the repeat valid time.
- Set the 16-bit timer to 1.5 ms.

(4) Program listing

```
        PUBLIC  RPT,IPDTFG,RMDTOK,RMDTSET
        PUBLIC  RMENDCT,RPTCT,SELMOD,LD_CT,RMDATA
        EXTRN   CSTM
RM_DAT  DSEG    SADDR
RPTCT:  DS      1                       ; Repeat code valid time counter
RMENDCT:DS      1                       ; No input time counter after data input
SELMOD: DS      1                       ; Mode selection
LD_CT:  DS      1                       ; Leader signal detection counter
RMDATA: DS      1                       ; Valid data storage area

RM_DATP DSEG    SADDRP
WORKP:  DS      4                       ; Input signal storage area

        BSEG
IPDTFG  DBIT                            ; Valid data is present.
RMDTOK  DBIT                            ; Input signal is valid.
RMDTSET DBIT                            ; Input signal is present.
RPT     DBIT                            ; Repeat code valid period

VEP0    CSEG    AT 06H
        DW      INTP0                   ; INTP0 vector address setting

VETM0   CSEG    AT 14H
        DW      INTTM0                  ; 16-bit timer vector address setting

;**********************************************
;    Remote control signal timer processing
;**********************************************
TM0_SEG    CSEG
INTTM0:

        SEL RB1
        EI                              ; Interrupt enabled (INTP0)
        if_bit(IPDTFG)                  ; Is the input signal present?
            if_bit(RMDTOK)              ; Is the data valid?
            RPTCT--
                if(RPTCT==#0)           ; Repeat invalid time
                    CLR1  RPT           ; Repeat code invalid state
                    CLR1  IPDTFG
                    CLR1  RMDTOK
                endif
                CALL   !S_LOWCT
            else
                RMENDCT--
                if(RMENDCT==#0)
                    SET1  RMDTOK        ; Set to valid data is present.
                    SET1  RMDTSET
                    CALL  !S_M0SET      ; Set to leader (low) detection mode.
                endif
                LD_CT=#5
            endif
        else
            CALL          !S_LOWCT
        endif
        RET1
```

```
S_LOWCT:
        if(SELMOD==#0)              ; Leader (low) detection mode?
           if_bit(!P0.0)
              LD_CT--
              if(LD_CT==#0)
                 SELMOD=#1          ; Leader (low) measurement mode
                 TMC0=#00000000B
                 CR00=#32767        ; 7.81-ms timer
                 TMC0=#00001100B
                 INTM0=#00000100B
                 CLR1     PIF0
                 CLR1     PMK0      ; INTP0 interrupt enabled
                 LD_CT=#5
              endif
           else
              LD_CT=#5
           endif
        else
           CALL  !S_M0SET           ; Set to leader (low) detection mode.
           LD_CT=#5
        endif
        RET
$EJECT
;************************************************************
;*   Remote control signal edge detection processing
;************************************************************
P0_SEG  CSEG
INTP0;

        SEL   RB0
        CALL  !WAIT                 ; 100-µs wait
        switch(SELMOD)
        case 1:
           CALL  !LEAD_L            ; Leader low detection processing
           break
        case 2:
           CALL  !LEAD_H            ; Leader high detection processing
           break
        case 3:
           CALL  !CDCODE            ; Custom/data code read processing
           break
        case 4:
           CALL  !REPCD             ; Repeat code detection processing
           break
        case 5:
           CALL  !ENDCHK            ; Error data detection processing
        ends
        RETI
```

```
;*******************************
;*   Leader low detection
;*******************************
LEAD_L:

     if_bit(P0.0)                   ; Level check P0.0 = 0:noise
        CALL  !WAIT                 ; 100-µs wait
        if_bit(P0.0)
           CALL  !CR_READ           ; Timer value read
           if(AX>=#3354)            ; 6.8 ms – (1.5 ms x 4)
              if(AX<#18035)         ; 11.8 ms – (1.5 ms x 5)
                 SELMOD=#2          ; Leader high detection mode
                 INTM0=#00000000B   ; INTP0 falling edge
              else
                 CALL    !S_M0SET   ; Set to leader (low) detection mode.
              endif
           else
              CALL    !S_M0SET      ; Set to leader (low) detection mode.
           endif
        endif
     endif
     RET
$EJECT
;******************************
;*   Leader high detection
;******************************
LEAD_H:
     if_bit(!P0.0)                  ; Level check P0.0 = 1:noise
        CALL  !WAIT                 ; 100-µs wait
        if_bit(!P0.0)
           CALL  !CR_READ           ; Timer value read
           if(AX>=#5710-160/2)      ; 1.8 ms – 100 µs x 2 – 160 clocks (edge detection -> timer start)
              if(AX<#20132-160/2)   ; 5 ms – 100 µs x 2 – 160 clocks (edge detection ->  timer start)
                 if(AX>#11743-160/2) ; Custom/data code (3 ms – 100 µs x 2)?
                    SELMOD=#3        ; Data read mode
                    WORKP=#0000H     ; Initialize work area.
                    (WORKP)+2=#8000H ; Set most significant bit to 1 (for verifying the end of data).
                 else
                    SELMOD=#4        ; Repeat detection mode
                    INTM0=#00000100B ; INTP0 rising
                 endif
              else
                 CALL   !S_M0SET    ; Set to leader (low) detection mode.
              endif
           else
              CALL    !S_M0SET      ; Set to leader (low) detection mode.
           endif
        endif
     endif
     RET
$EJECT
```

```
;*******************************
;*   Custom/data code read
;*******************************
CDCODE:
      if_bit(!P0.0)                  ; Level check P0.0 = 1:noise
         CALL  !WAIT                 ; 100-µs wait
         if_bit(!P0.0)
            CALL  !CR_READ           ; Timer value read
            if(AX>=#1257-190/2)      ; 0.5 ms - 100 µs x 2 – 190 clocks (edge detection -> timer start)
               if(AX<#9646-190/2)    ; 2.5 ms - 100 µs x 2 – 190 clocks (edge detection -> timer start)
                  if(AX>=#6710-190/2); 1.8 ms - 100 µs x 2 – 190 clocks (edge detection -> timer start)
                     SET1  CY
                  else
                     CLR1  CY
                  endif
                  HL=#WORKP+3        ; Set work area address.
                  C=#4               ; Set number of digits in work area.
               WKSHFT:
                  A=[HL]             ; 1-bit data save
                  RORC  A,1          ; 1-bit shift
                  [HL]=A
                  HL--
                  DBNZ  C,$WKSHFT    ; Completed the shift of all digits.
                  if_bit(CY)         ; Is 32-bit input finished?
                     if(WORKP+0==#CSTM) (A)
                                     ; Custom code check
                        A^=WORKP+1
                        if(A==#0FFH) ; Custom code inverted data check
                           A=WORKP+2
                           A^=WORKP+3 ; Data code inverted data check
                           if(A==#0FFH)
                                     ; Save input data.
                              RMDATA=WORKP+2 (A)
                                     ; Set in the input data present state.
                              SET1    IPDTFG
                              CLR1    RMDTSET
                              CLR1    RPT
                              CLR1    RMDTOK
                              CALL    !S_M5SET
                           else
                                     ; Set to leader (low) detection mode.
                              CALL    !S_M0SET
                           endif
                        else
                                     ; Set to leader (low) detection mode.
                           CALL   !S_M0SET
                        endif
                     else
                        CALL  !S_M0SET
```

```
                    endif
                 endif
              else
                 CALL  !S_M0SET     ; Set to leader (low) detection mode.
              endif
           else
              CALL  !S_M0SET        ; Set to leader (low) detection mode.
           endif
        endif
     endif
     RET
$EJECT
;*******************************
;*   Repeat code detection
;*******************************
REPCD:
     if_bit(P0.0)                   ; Level check P0.0 = 0:noise
        CALL  !WAIT                 ; 100-µs wait
        if_bit(P0.0)
           if_bit(RMDTOK)           ; Is valid data present?
              CALL  !CR_READ        ; Timer value read
              if(AX<=#3354-190/2)   ; 1 ms – 100 µs x 2 – 190 clocks (edge detection -> timer start)
                 SET1   RPT
                 CLR1   RMDTOK      ; Input signal check after the end of data
                 CLR1   RMDTSET
                 CALL   !S_M5SET
              else
                 CALL   !S_M0SET    ; Set to leader (low) detection mode.
              endif
           else
              CALL    !S_M0SET      ; Set to leader (low) detection mode.
           endif
        endif
     endif
     RET
$EJECT
```

```
;*****************************************
;*        Error data detection
;*****************************************
ENDCHK:
       if_bit(!P0.0)                    ; Level check P0.0 = 1:noise
          CALL    !WAIT                 ; 100-µs wait
          if_bit(!P0.0)
             CLR1    !PDTFG             ; Error data input
             CLR1    RPT                ; Input signal invalid
             CALL    !S_M0SET           ; Set to leader (low) detection mode.
          endif
       endif
       RET

;*****************************************
;*           100-µs wait
;*****************************************
WAIT:
       B=#(838-14-12-8)/12              ; CALL(14), RET(12), MOV(8)
WAITCT:                                 ; 100-µs setting
       DBNZ  B,$WAITCT                  ; 1 instruction, 12 clocks
       RET

;*****************************************
;*   Leader (low) detection mode setting
;*****************************************
S_M0SET:
       TMC0=#00000000B
       CR00=#6290
       TCL0=#00010000B                  ; Set timer to 1.5 ms.
       TMC0=#00001100B
       SELMOD=#0                        ; Leader (low) detection mode
       SET1  PMK0
       RET

;*****************************************
;*   Error data detection mode setting
;*****************************************
S_M5SET:
       RPTCT=#173                       ; Counter for 250-ms measurement
       SELMOD=#5                        ; End of data input mode
       RMENDCT=#3                       ; Counter for no input verification
       TMC0=#00000000B                  ; Operation stopped
       CR00=#6290                       ; Set to 1.5 ms.
       TMC0=#00001100B
       RET
;*****************************************
;*        Read timer count value
;*****************************************
CR_READ:
       AX=CR01
       TMC0=#00000000B                  ; Stop operation
       TMC0=#00001100B                  ; Timer start
       RET
```

5.3.2 Remote Control Reception by PWM Output and Free Running

In this program, the valid pulse widths when the remote control signal is the received signal are shown in Table 5-2. The processing methods for each signal are explained in **<1>** to **<6>**.

Table 5-2. Valid Time of the Input Signal

Signal name		Output time	Valid time
Leader code (low)		9 ms	3 ms-10 ms
Leader code (high)	Normal	4.5 ms	3 ms-5 ms
	Repeat	2.25 ms	1.8 ms-3 ms
Custom code/data code	0	1.125 ms	0.5 ms-1.8 ms
	1	2.25 ms	1.8 ms-2.5 ms

<1> Leader code (low)

An interrupt request during the detection of the falling edge of INTP0 causes the 16-bit capture register (CR01) value to be saved in memory. When the rising edge occurs, the pulse width is measured from the difference with the 16-bit compare register (CR00).

<2> Leader code (high)

Based on an interrupt request due to the falling edge of INTP0, the pulse width during the high level of the leader code is measured by the timer count.

<3> Custom/data code

Based on an interrupt request due to the falling edge of INTP0, the pulse width is measured for each bit (1 period). After the 32nd bit of data is read in, test for a match of the inverse data and custom code. Furthermore, the absence of a 33rd bit of data is verified.

<4> Repeat code detection

When the high level of the leader code is less than 3 ms, the pulse width is measured until the rising edge of INTP0 after the leader code output.

<5> Valid time for repeat code

After valid data input, the overflow flag (OVF0) of the 16-bit timer/event counter is tested in the main program. A valid time of 250 ms for the repeat code is measured.

<6> Time out during pulse width measurement

The OVF0 of the 16-bit timer/event counter during pulse width measurement is tested in the main program. When detected twice, a time out occurs and data becomes invalid.

Because the 16-bit timer/event counter in this example is operated in the PWM output mode, by linking the program shown in **Section 5.2**, remote control reception and PWM output can be simultaneously executed.

(1) Package description

<Symbols declared as public>

TIM_PRO : Name of subroutine for timer overflow processing
RMDATA : Saves remote control reception data.
RPT : Decision flag for the repeat valid interval
IPDTFG : Decision flag indicating the presence of valid data
RMDTOK : Decision flag indicating the presence of a valid input signal
RMDTSET: Decision flag indicating the presence of an input signal
OVSENS : Timer overflow detection flag in INTP0 processing

<Register used>

Bank 0: AX, BC, HL

<RAM used>

Name	Use	Attribute	Byte
RPTCT	Time counter for valid repeat code	SADDR	1
RMENDCT	No input time counter after data input		
SELMOD	Mode selection		
LD_CT	Leader signal detection counter		
RMDATA	Valid data storage area		
TO_CNT	Timer overflow detection counter		
CR01_NP	Newest timer counter value storage area	SADDRP	2
CR01_OP	Previous timer counter value storage area		
WORKP	Input signal storage area		4

<Flags used>

Name	Use
IPDTFG	Presence of valid data
RMDTOK	Presence of valid input signal
RMDTSET	Presence of input signal
RPT	Decision on whether the repeat valid interval has elapsed
TO_FLG	Timer overflow present
OVSENS	Timer overflow detection in INTP0 processing

<Nesting>

5 levels, 11 bytes

<Hardware used>

- 16-bit timer/event counter
- P00/INTP0
- P30/TO0

<Initial settings>

- 16-bit timer/event counter setting

PWM output mode	TMC0 = #00000010B
Basic PWM period of 61.0 μs	TCL0 = #00010000B
Active-low output	TOC0 = #00000011B

- P30 output mode PM30 = 0
- INTP0 sampling clock $f_X/2^7$ SCS = #00000011B
- INTP0 high-priority interrupt request PPR0 = 0
- INTP0 interrupt enabled PMK0 = 0
- Define custom code in CSTM. This is a public declaration.
- Clear RAM

<Startup procedure>

- Test OVF0 of the 16-bit timer/event counter. When OVF0 is set, call the TIM_PRO subroutine.
- Start using an interrupt request based on the edge detection of the remote control signal.

(2) Example use

```
        PUBLIC  CSTM
        EXTRN   RMDATA,RPTCT,PWM,PWMOUT,TIM_PR0
        EXTBIT  RPT,RMDTSET,IPDTFG,TO_FLG,OVSENS
CSTM    EQU     9DH                     ; Custom code

        TOC0=#00000011B                 ; Setting of PWM output and active low
        TCL0=#00010000B                 ; Select fX count clock.
        TMC0=#00000010B                 ; Overflow present in PWM mode.
        INTM0=#00000000B                ; INTP0 falling edge
        SCS=#00000011B                  ; INTP0 sampling clock of fX/128

        CLR1    PPR0                    ; INTP0 high priority
        CLR1    RPT                     ; Clear flag
        CLR1    IPDTFG
        CLR1    RMDTSET

        CLR1    PMK0                    ; INTP0 interrupt enabled
        EI
DT_TEST:
        if_bit(OVSENS)                  ; Timer overflow detection in INTP0 processing
            CLR1    OVSENS
            CALL    !TIM_PR0
        elseif_bit(OVF0)                ; Timer overflow is present.
            CLR1    OVF0
            SET1    TO_FLG
            CALL    !TIM_PR0
        endif
        if_bit(RMDTSET)
            CLR1    RMDTSET
            if_bit(RPT)
;
;                  Repeat processing
;
            else
;
;                  Input present processing
;
            endif
        else
            if_bit(!RPT)
;
;                  No input present processing
;
            endif
        endif
        MOV     PWMOUT.A
        CALL    !PWM
```

(3) SPD chart

```
TIM_PRO ─◇─ IF : An input signal is present.
          THEN
            ◇─ IF : There is valid data.
               THEN
                 ◇─ IF : Repeat code invalid period.
                    THEN
                      Set in the repeat code invalid state.
                        Clear RPT, IPDTFG, and RMDTOK.
                 Timer overflow check TO_CHK
               ELSE
                 ◇─ IF : There is no input after the end of data input (within 61.0 µs x 2)
                    THEN
                      Set to valid data is present.
                        Set RMDTOK and RMDTSET.
                      Set to leader low detection mode. S_M0SET
          ELSE
            Timer overflow check TO_CHK
```

```
TO_CHK ─◇─ IF : Leader low detection mode.
         THEN
           Set to no timer overflow
         ELSE
           Timer overflow count
           ◇─ IF : Timer overflows twice
              THEN
                Set to leader low detection mode. S_M0SET
```

```
INTP0 ─── Select register bank 0.
      ─── 100-µs wait WAIT
      ─◇─ CASE : SELMOD
         OF : 0
           Leader low detection mode RM_STA
         OF : 1
           Leader low measurement mode LEAD_L
         OF : 2
           Leader high measurement mode LEAD_H
         OF : 3
           Custom code/data read mode CDCODE
         OF : 4
           Repeat code detection mode REPCD
         OF : 5
           Error data detection mode ENDCHK
```

RM_STA

- IF : P00 = LOW
  - THEN
    - 100-µs wait WAIT
    - IF : P00 = LOW
      - THEN
        - Save data in capture register into memory.
        - Select leader low measurement mode 1.
        - Set to the INTP0 rising edge detection mode.

LEAD_L

- IF : P00 = HIGH
  - THEN
    - 100-µs wait WAIT
    - IF : P00 = LOW
      - THEN
        - Timer read PW_CT
        - IF : 3 ms ≤ leader low ≤ 10ms
          - THEN
            - Select the leader high detection mode.
            - Set to INTP0 falling edge detection mode.
          - ELSE
            - Set to leader low detection mode. S_M0SET

LEAD_H

- IF : P00 = LOW
  - THEN
    - 100-µs wait WAIT
    - IF : P00 = LOW
      - THEN
        - Timer read PW_CT
        - IF : 2 ms ≤ leader high ≤ 5 ms
          - THEN
            - IF : Leader high ≥ 3 ms
              - THEN
                - Select custom code/data read mode.
                - Initialize the data storage area.
              - ELSE
                - Select the repeat detection mode.
                - Set to INTP0 rising edge detection mode.
          - ELSE
            - Set to leader low detection mode. S_M0SET

CDCODE

- IF : P00 = LOW
  - THEN
    - 100-µs wait WAIT
    - IF : P00 = LOW
      - THEN
        - Timer read CR_READ
        - IF : 0.5 ms < input data ≤ 2.5 ms
          - THEN
            - IF : input data ≥ 1.8 ms
              - THEN
                - Set CY.
              - ELSE
                - Clear CY.
            - Save CY in data storage area.
            - IF : End of 32-bit data input.
              - THEN
                - IF : Custom code match
                  - THEN
                    - IF : Inverse data of custom/data code match
                      - THEN
                        - Save data code.
                        - Set in the input data present state.
                          Set IPDTFG and clear RMDTSET, RPT, and RMDTOK.
                        - Set to error data detection mode. S_M5SET
                      - ELSE
                        - Set to leader low detection mode S_M0SET
                  - ELSE
                    - Set to leader low detection mode S_M0SET
          - ELSE
            - Set to leader low detection mode S_M0SET

REPCD

- IF : P00 = HIGH
  - THEN
    - 100-µs wait WAIT
    - IF : P00 = HIGH
      - THEN
        - IF : Valid data is present.
          - THEN
            - Timer read PW_CT
            - IF : Repeat code ≤ 1 ms
              - THEN
                - Set in the repeat code valid state.
                  Set RPT.
                - Set in the end of data input state.
                - Set to error data detection mode. S_M5SET
              - ELSE
                - Set to leader low detection mode. S_M0SET
          - ELSE
            - Set to error data detection mode. S_M5SET

ENDCHK

- IF : P00 = LOW
  - THEN
    - 100-µs wait WAIT
    - IF : P00 = LOW
      - THEN
        - Set the input signal in the invalid state.
          Clear IPDTFG and RPT.
        - Set to leader low detection mode. S_M0SET

PW_CT

- IF : OVF occurs after edge detection.
  - THEN
    - IF : OVF generated < interrupt reception processing time or more (65 clocks)
      - THEN
        - Set to timer overflow present.
- Read capture register.
- Subtract the capture register value from the previous value.
- IF : Borrow generated in the subtraction result (CY= 1)
  - THEN
    - IF : There is timer overflow (TO_FLG = 1)
      - THEN
        - Clear CY flag.
  - ELSE
    - IF : There is timer overflow (TO_FLG = 1)
      - THEN
        - Set CY flag.
- Save the value in the capture register into memory.

S_M0SET

- Select the leader low detection mode.
  Clear TO_FLG.
- Set to INTP0 falling edge detection mode.

S_M5SET

- Select the error data detection mode.
- Set the counter for repeat valid time.

(4) Program listing

```
        PUBLIC  TIM_PRO,RPT,IPDTFG,RMDTOK,RMDTSET
        PUBLIC  RMENDCT,RPTCT,SELMOD,LD_CT,RMDATA
        PUBLIC  TO_FLG,OVSENS
        EXTRN   CSTM

RM_DAT  DSEG    SADDR
RPTCT:  DS      1                       ; Counter for repeat code valid time
RMENDCT:DS      1                       ; Counter for no input time after data input
SELMOD: DS      1                       ; Mode selection
LD_CT   DS      1                       ; Leader signal detection counter
RMDATA: DS      1                       ; Valid data storage area
TO_CNT: DS      1                       ; Timer overflow counter

RM_DATP DSEG    SADDRP
CR01_NP:DS      2                       ; Newest timer counter value storage area
CR01_OP:DS      2                       ; Previous timer counter value storage area
WORKP:  DS      4                       ; Input signal storage area

        BSEG
!PDTFG  DBIT                            ; Valid data is present.
RMDTOK  DBIT                            ; Input signal is valid.
RMDTSET DBIT                            ; Input signal is present.
RPT     DBIT                            ; Repeat code valid period
TO_FLG  DBIT                            ; Timer overflow is present.
OVSENS  DBIT                            ; Timer overflow detection in INTP0 processing

VEP0    CSEG    AT 06H
        DW      INTP0                   ; Setting of the INTP0 vector address

$EJECT
;***********************************************
;*   Remote control signal timer processing
;***********************************************
TM0_SEG CSEG
TIM_PRO:
        if_bit(IPDTFG)                  ; Is an input signal present?
            if_bit(RMDTOK)              ; Is the data valid?
                RPTCT--
                if(RPTCT==#0)           ; Repeat invalid time
                    CLR1    RPT         ; Repeat code invalid state
                    CLR1    !PDTFG
                    CLR1    RMDTOK
                endif
            else
                RMENDCT--
                if(RMENDCT==#0)
                    SET1    RMDTOK      ; Set to valid data is present.
                    SET1    RMDTSET
                    CALL    !S_M0SET    ; Set to leader (low) detection mode.
                endif
            endif
        else
            CALL    !TO_CHK             ; Timer overflow check
        endif
        RET
```

```
TO_CHK:
      if(SELMOD==#0)
         CLR1   TO_FLG
      else
         TO_CNT++
         if(TO_CNT==#2)
            CALL     !S_M0SET                  ; Set to starting edge detection mode.
         endif
      endif
      RET
   $EJECT
;**********************************************
;*   Remote control signal edge detection
;**********************************************
P0_SEG  CSEG
INTP0:
        SEL RB0

        CALL     !WAIT                         ; 100-µs wait

        switch(SELMOD)

        case 0:
           CALL  !RM_STA                       ; Starting edge detection
           break
        case 1:
           CALL  !LEAD_L                       ; Leader low detection
           break
        case 2:
           CALL  !LEAD_H                       ; Leader high detection
           break
        case 3:
           CALL  !CDCODE                       ; Custom/data code read
           break
        case 4:
           CALL  !REPCD                        ; Repeat code detection
           break
        case 5:
           CALL  !ENDCHK                       ; Error data detection
        ends
        RETI

;**********************************************
;*          Starting edge detection
;**********************************************
RM_STA:
        CLR1     TO_FLG                        ; Timer counter starts
        if_bit(!P0.0)                          ; Level check P0.0 = 1:noise
           CALL     !WAIT                      ; 100-µs wait
           if_bit(!P0.0)
              CR01_OP=CR01 (AX)                ; Save capture register.
              SELMOD=#1                        ; Leader low detection mode
              INTM0=#00000100B                 ; INTP0 rising edge
              TO_CNT=#0
           endif
        endif
        RET
```

```
;******************************
;*   Leader low detection
;******************************
LEAD_L:
     if_bit(P0.0)                        ; Level check P0.0 = 1:noise
        CALL   !WAIT                     ; 100-µs wait
        if_bit(P0.0)
           CALL   !PW_CT                 ; Timer value read
           if_bit(!CY)
              TO_CNT=#0
              if(AX>=#12582)             ; 3 ms
                 if(AX<#41942)           ; 10 ms
                    SELMOD=#2            ; Leader high detection mode
                    INTM0=#00000000B     ; INTP0 falling edge
                 else
                    CALL    !S_M0SET     ; Set to starting edge detection mode.
                 endif
              else
                 CALL   !S_M0SET         ; Set to starting edge detection mode.
              endif
           else
              CALL   !S_M0SET            ; Set to starting edge detection mode.
           endif
        endif
     endif
     RET
  $EJECT
;******************************
;*   Leader high detection
;******************************
LEAD_H:
     if_bit(!P0.0)                       ; Level check P0.0 = 0:noise
        CALL   !WAIT                     ; 100-µs wait
        if_bit(!P0.0)
           CALL   !PW_CT                 ; Timer value read
           if_bit(!CY)
              TO_CNT=#0
              if(AX>=#7549)              ; 1.8 ms
                 if(AX<#20971)           ; 5 ms
                    if(AX>#12582)        ; Custom/data code (3 ms)?
                       SELMOD=#3         ; Data read mode
                       WORKP=#0000H      ; Initialize work area.
                       (WORKP)+2=#8000H  ; Set the most significant bit to 1 (to verify the end of data).
                    else
                       SELMOD=#4         ; Repeat detection mode
                       INTM0=#00000100B  ; INTP0 rising
                    endif
                 else
                    CALL    !S_M0SET     ; Set to starting edge detection mode.
                 endif
              else
                 CALL   !S_M0SET         ; Set to starting edge detection mode.
              endif
           else
              CALL !S_M0SET              ; Set to starting edge detection mode.
           endif
        endif
     endif
     RET
  $EJECT
```

```
;******************************
;*   Custom/data code read
;******************************
CDCODE:
     if_bit(!P0.0)                      ; Level check P0.0 = 1: noise
        CALL  !WAIT                     ; 100-µs wait
        if_bit(!P0.0)
           CALL  !PW_CT                 ; Timer value read
           if_bit(!CY)
              TO_CNT=#0
              if(AX>=#2096)             ; 0.5 ms
                 if(AX<#10485)          ; 2.5 ms
                    if(AX>=#7549)       ; 1.8 ms
                       SET1     CY
                    else
                       CLR1     CY
                    endif

                    HL=#WORKP+3         ; Set work area address.
                    C=#4                ; Set the number of digits in the work area.
                 WKSHFT:
                    A=[HL]              ; 1-bit data save
                    RORC  A,1           ; 1-bit shift
                    [HL]=A
                    HL--
                    DBNZ  C,$WKSHFT     ; End of shifting all digits

                    if_bit(CY)          ; End of 32-bit input?
                                        ; Custom code check
                       if(WORKP+0==#CSTM) (A)
                          A^=WORKP+1
                          if(A==#0FFH)  ; Custom code inverse data check
                             A=WORKP+2
                                        ; Data code inverse data check
                             A^=WORKP+3
                             if(A==#0FFH)
                                        ; Save input data.
                                RMDATA=WORKP+2 (A)
                                        ; Set in the input data present state.
                                SET1    IPDTFG
                                CLR1    RMDTSET
                                CLR1    RPT
                                CLR1    RMDTOK
                                CALL    !S_M5SET
                             else
                                        ; Set to starting edge detection mode.
                                CALL    !S_M0SET
                             endif
                          else
                                        ; Set to starting edge detection mode.
                             CALL    !S_M0SET
                          endif
                       else
                          CALL  !S_M0SET
                       endif
                    endif
                 else
                    CALL  !S_M0SET      ; Set to starting edge detection mode.
                 endif
              else
```

```
                CALL  !S_M0SET         ; Set to starting edge detection mode.
             endif
          else
             CALL  !S_M0SET            ; Set to starting edge detection mode.
          endif
       endif
    endif
    RET
$EJECT

;******************************
;*   Repeat code detection
;******************************
REPCD:
    if_bit(P0.0)                       ; Level check P0.0 = 1: noise
       CALL   !WAIT                    ; 100-µs wait
       if_bit(P0.0)
          if_bit(RMDTOK)               ; Is the data valid?
             CALL    !PW_CT            ; Timer value read
             if_bit(!CY)
                TO_CNT=#0
                if(AX<=#4193)          ; 1 ms
                   SET1    RPT
                   CLR1    RMDTOK      ; Input signal check at the end of data
                   CLR1    RMDTSET
                   CALL    !S_M5SET
                else
                   CALL    !S_M0SET    ; Set to starting edge detection mode.
                endif
             else
                CALL    !S_M0SET       ; Set to starting edge detection mode.
             endif
          else
             CALL   !S_M0SET           ; Set to starting edge detection mode.
          endif
       endif
    endif
    RET
  $EJECT
```

```
;********************************************
;*           Error data detection
;********************************************
ENDCHK:
     if_bit(!P0.0)                ; Level check P0.0 = 1:noise
        CALL  !WAIT               ; 100-µs wait
        if_bit(!P0.0)
           CLR1  IPDTFG           ; Error data input
           CLR1  RPT              ; Input signal invalid
           CALL  !S_M0SET         ; Set to starting edge detection mode.
        endif
     endif
     RET

;********************************************
;*   Calculation of capture register value
;********************************************
PW_CT:
     if_bit(OVF0)                 ; OVF0 after edge detection?
        if(CR01<#10000-33) (AX)   ; Interrupt reception processing time = 65 clocks (MAX)
           CLR1  OVF0
           SET1  OVSENS
           SET1  TO_FLG
        endif
     endif

     CR01_NP=CR01 (AX)            ; Capture register value read

     A=CR01_NP+0                  ; AX=CR01_NP-CR01_OP
     A-=CR01_OP
     X=A
     A=CR01_NP+1
     SUBC  A,CR01_OP+1

     BC=AX                        ; Calculation result save
     if_bit(CY)                   ; CR01_NP>CR01_OP
        if_bit(TO_FLG)            ; Timer overflow present (flag test).
           CLR1  CY               ; Normal data
        endif
     else
        if_bit(TO_FLG)            ; Timer overflow
           SET1  CY               ; Error occurred.
        endif
     endif

     CR01_OP=CR01_NP (AX)
     AX=BC                        ; Calculation result restored.
     CLR1  TO_FLG
     RET
```

```
;*******************************************
;*              100-µs wait
;*******************************************
WAIT:
     R=#(838-14-12-8)/12      ; CALL(14), RET(12), MOV(8)
WAITCT:                       ; Set 100-µs.
     DBNZ    B,$WAITCT        ; 1 instruction, 12 clocks
     RET

;*******************************************
;*   Starting edge detection mode setting
;*******************************************
S_M0SET:
     TO_CNT=#0
     SELMOD=#0                ; Starting edge detection mode
     INTM0=#00000000B         ; INTP0 falling edge
     RET

;*******************************************
;*   Error data detection mode setting
;*******************************************
S_M5SET:
     RPTCT=#16                ; Counter for 250-ms measurement
     SELMOD=#5                ; End of data input mode
     RMENDCT=#2               ; Counter to verify no input
     RET
```

CHAPTER 6 8-BIT TIMER/EVENT COUNTER APPLICATION

The 8-bit timer/event counter in the 78K/0 Series has the three functions of interval timer, external event counter, and square-wave output. In addition, the 8-bit timer/event counter has two on-chip channels. Moreover, they can be used as a 16-bit timer/event counter by connecting them in cascade.

The 8-bit timer/event counter requires the setting of the following five registers:

- Timer clock selection register 1 (TCL1)
- 8-bit timer mode control register (TMC1)
- 8-bit timer output control register (TOC1)
- Port mode register 3 (PM3)
- Port 3 (P3)

Figure 6-1. Format of Timer Clock Selection Register 1

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
TCL1	TCL17	TCL16	TCL15	TCL14	TCL13	TCL12	TCL11	TCL10	FF41H	00H	R/W

TCL13	TCL12	TCL11	TCL10	Count clock selection of 8-bit timer register 1
0	0	0	0	Falling edge at TI1
0	0	0	1	Rising edge at TI1
0	1	0	1	$f_X/2$ (2.5 MHz)
0	1	1	0	$f_X/2^2$ (1.25 MHz)
0	1	1	1	$f_X/2^3$ (625 kHz)
1	0	0	0	$f_X/2^4$ (313 kHz)
1	0	0	1	$f_X/2^5$ (156 kHz)
1	0	1	0	$f_X/2^6$ (78.1 kHz)
1	0	1	1	$f_X/2^7$ (39.1 kHz)
1	1	0	0	$f_X/2^8$ (19.5 kHz)
1	1	0	1	$f_X/2^9$ (9.8 kHz)
1	1	1	0	$f_X/2^{10}$ (4.9 kHz)
1	1	1	1	$f_X/2^{12}$ (1.2 kHz)
Other than the above				Setting prohibited

TCL17	TCL16	TCL15	TCL14	Count clock selection of 8-bit timer register 2
0	0	0	0	Falling edge at TI2
0	0	0	1	Rising edge at TI2
0	1	0	1	$f_X/2$ (2.5 MHz)
0	1	1	0	$f_X/2^2$ (1.25 MHz)
0	1	1	1	$f_X/2^3$ (625 kHz)
1	0	0	0	$f_X/2^4$ (313 kHz)
1	0	0	1	$f_X/2^5$ (156 kHz)
1	0	1	0	$f_X/2^6$ (78.1 kHz)
1	0	1	1	$f_X/2^7$ (39.1 kHz)
1	1	0	0	$f_X/2^8$ (19.5 kHz)
1	1	0	1	$f_X/2^9$ (9.8 kHz)
1	1	1	0	$f_X/2^{10}$ (4.9 kHz)
1	1	1	1	$f_X/2^{12}$ (1.2 kHz)
Other than the above				Setting prohibited

Caution When TCL1 will be rewritten with data other than identical data, rewrite after temporarily stopping the timer.

Remarks 1. f_X : Main system clock oscillation frequency

2. TI1: Input pin of 8-bit timer register 1

3. TI2: Input pin of 8-bit timer register 2

4. The values in parentheses apply to operation with f_X = 5.0 MHz.

Figure 6-2. Format of the 8-Bit Timer Mode Control Register

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
TMC1	0	0	0	0	0	TMC12	TCE2	TCE1	FF49H	00H	R/W

TCE1	Controlling the operation of 8-bit timer register 1
0	Stop operation (TM1 is cleared to 0)
1	Operation enabled

TCE2	Controlling the operation of 8-bit timer register 2
0	Stop operation (TM2 is cleared to 0)
1	Operation enabled

TMC12	Selection of the operating mode
0	8-bit timer register x 2-channel mode (TM1, TM2)
1	16-bit timer register x 1-channel mode (TMS)

Cautions 1. Switch the operating mode after stopping timer operation.

2. When used as a 16-bit timer register, use TCE1 to enable or stop operation.

Figure 6-3. Format of the 8-Bit Timer Output Control Register

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
TOC1	LVS2	LVR2	TOC15	TOE2	LVS1	LVR1	TOC11	TOE1	FF4FH	00H	R/W

TOE1	Output control of 8-bit timer/event counter 1
0	Output prohibited (port mode)
1	Output enabled

TOC11	Control of timer output flip-flop of 8-bit timer/event counter 1
0	Inversion disabled
1	Inversion enabled

LVS1	LVR1	Setting the state of the timer output flip-flop of 8-bit timer/event counter 1
0	0	Do not change.
0	1	Reset the timer output flip-flop (0)
1	0	Set the timer output flip-flop (1)
1	1	Setting prohibited

TOE2	Output control of 8-bit timer/event counter 2
0	Output prohibited (port mode)
1	Output enabled

TOC15	Control of timer output flip-flop of 8-bit timer/event counter 2
0	Inversion disabled
1	Inversion enabled

LVS2	LVR2	Setting the state of the timer output flip-flop of 8-bit timer/event counter 2
0	0	Do not change
0	1	Reset the timer output flip-flop (0)
1	0	Set the timer output flip-flop (1)
1	1	Setting prohibited

Cautions 1. Always set TOC1 after stopping timer operation.

2. When LVS1, LVS2, LVR1, and LVR2 are read out after data were set, 0's are read out.

Figure 6-4. Format of Port Mode Register 3

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
PM3	PM37	PM36	PM35	PM34	PM33	PM32	PM31	PM30	FF23H	FFH	R/W

PM3n	Selection of the I/O mode of pin P3n (n=0 to 7)
0	Output mode (output buffer on)
1	Input mode (output buffer off)

Caution When the P31/TO1 and P32/TO2 pins are used for timer output, do not only set the output latches of PM31 and PM32 to 0, also set the output latches of P31 and P32 to 0.

6.1 SETTING THE INTERVAL TIMER

When the interval timer is used, the operating mode of the 8-bit timer is set by the 8-bit timer mode control register (TMC1) and the interval time is set by timer clock selection register 1 (TCL1).

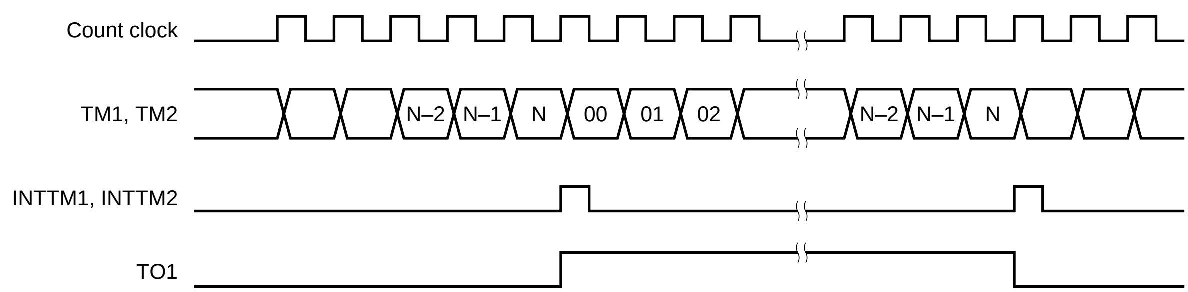
The values of the compare registers (CR10, CR20) are set based on the interval time and count clock. The setting time is determined in the form shown below.

Setting-time = (value-in-compare-register + 1) x count-clock-period

The setting time can be determined in a similar manner even when used as an 8-bit timer or as a 16-bit timer. However, when used as a 16-bit timer, the count clock becomes the value selected in bits 0 to 3 (TCL10 to TCL13) in TCL1.

Next, examples of each mode of the 8-bit timer and 16-bit timer are illustrated.

Figure 6-5. Count Timing of an 8-Bit Timer



6.1.1 Setting an 8-Bit Timer

This example describes using 8-bit timer 2 and setting the interval times of 500 μs and 100 ms.

(a) For the 500-μs interval

<1> TMC1 setting

Select the “8-bit timer register x 2-channel” mode and enable “8-bit timer 2” operation.

<2> TCL1 setting

A setting above 500 μs is possible. Select the highest possible resolution of $f_X/2^4$.

<3> CR20 setting

$$500\ \mu s = (N + 1) \times \frac{1}{4.19\ MHz/2^4}$$

$$N = 500\ \mu s \times 4.19\ MHz/2^4 - 1 \fallingdotseq 130$$

(1) Program listing

```
TCL1 = #10001000B    ; Select fX/2^4 for the count clock.
CR20 = #130
TMC1 = #00000010B
```

(b) For the 100-ms interval

<1> TMC1 setting

Select the “8-bit timer register x 2-channel” mode and enable “8-bit timer 2” operation.

<2> TCL1 setting

A setting above 100 ms is possible. Select the highest possible resolution of $f_X/2^{12}$.

<3> CR20 setting

$$100 \text{ ms} = (N + 1) \times \frac{1}{4.19 \text{ MHz}/2^{12}}$$

$$N = 100 \text{ ms} \times 4.19 \text{ MHz}/2^{12} - 1 \fallingdotseq 101$$

(1) Program listing

```
TCL1 = #11111111B     ; Select count clock fX/2^12.
CR20 = #101
TMC1 = #00000010B
```

6.1.2 Setting the 16-Bit Timer

This example describes connecting 8-bit timer 1 and 8-bit timer 2 in a cascade and setting the interval times of 500 ms and 10 s.

(a) For the 500-ms interval

<1> TMC1 setting

In the “16-bit timer register x 1-channel” mode, enable the operations of 8-bit timers 1 and 2.

<2> TCL1 setting

A setting above 500 ms is possible. Select the highest possible resolution of $f_X/2^5$.

<3> CR10 and CR20 settings

$$500 \text{ ms} = \frac{N + 1}{4.19 \text{ MHz}/2^5}$$

$N = 500 \text{ ms} \times 4.19 \text{ MHz}/2^5 - 1 \fallingdotseq 65468 = \text{FF6CH}$

CR10 = 6CH, CR20 = FFH

(1) Program listing

```
TCL1 = #00001001B
CR10 = #06CH        ; Set 65468 in CR10 and CR20.
CR20 = #0FFH        ; CR10 = 6CH, CR20 = FFH
TMC1 = #00000111B
```

(b) For the 10-s interval

<1> TMC1 setting

In the “16-bit timer register x 1-channel” mode, enable the operations of 8-bit timers 1 and 2.

<2> TCL1 setting

A setting above 10 s is possible. Select the highest possible resolution of $f_X/2^{10}$.

<3> CR10 and CR20 settings

$$10 \text{ s} = \frac{N + 1}{4.19 \text{ MHz}/2^{10}}$$

$N = 10 \text{ s} \times 4.19 \text{ MHz}/2^{10} - 1 \fallingdotseq 40959 = 9FFFH$

CR10 = FFH, CR20 = 9FH

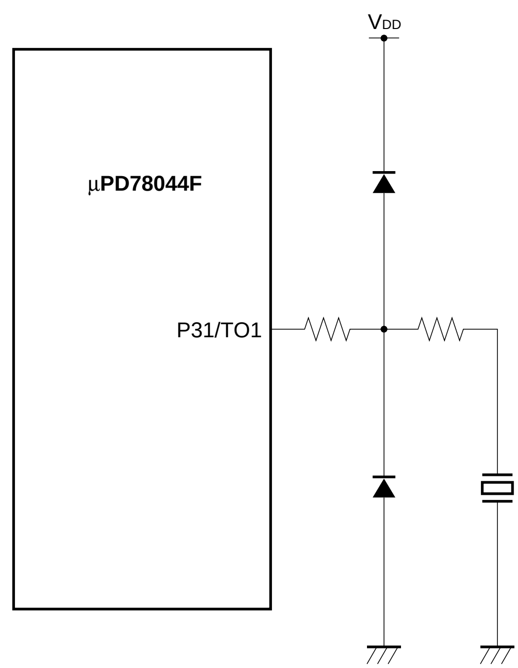
(1) Program listing

```
TCL1 = #00001110B
CR10 = #0FFH        ; Set 40959 in CR10 and CR20.
CR20 = #9FH         ; CR10 = FFH, CR20 = 9FH
TMC1 = #00000111B
```

6.2 MUSICAL SCALE GENERATION

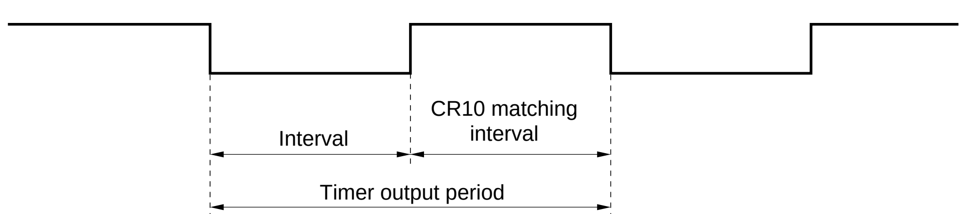
In this example, the square-wave output (P31/TO1) function of 8-bit timer/event counter 1 is used and a program is illustrated that supplies pulses to an externally attached buzzer to generate a musical scale.

Figure 6-6. Musical Scale Generation Circuit



The output frequency from pin P31/TO1 is set in the count clock and the compare register. In this example, because the center of the frequencies of the musical scale is set in the range of 523 Hz and 1046 Hz, $f_X/2^5$ is selected for the count clock. Table 6-1 lists the settings of the musical scale, compare registers, and frequencies of the pulses to be output. However, because the timer output matches the compare register twice and is created in one period, the interval setting is in one-half of a period.

Figure 6-7. Timer Output and Interval



In the temporal length of the sound, the interval time is set in the 8-bit timer/event counter 2. The number of interrupts are counted and the output time is determined. In this example, 8-bit timer/event counter 2 is set to 20 ms.

Table 6-1. Musical Scale and Frequencies

Musical scale	Musical scale frequencies (Hz)	Compare register value	Output frequency (Hz)
C	523.25	124	524.3
D	587.33	111	585.1
E	659.25	98	662.0
F	698.46	93	697.2
G	783.98	83	780.2
A	880.00	73	885.6
B	987.77	65	993.0
C	1046.5	62	1040

The format of the data table in this program is shown below.

```
TABLE:
        DB musical scale data 1, sound length data 1
        DB musical scale data 2, sound length data 2
                   :                      :
        DB musical scale data n, sound length data n
        DB 0,                    0
```

When there is a rest, musical scale data is set to 0. At the end of data, the length of the sound data is set to 0.

Example Count of the 8-bit timer/event counter 2 when the sound is output for one second
Count = 1 s/20 ms = 50 (Data for the count is set to 50.)

Data in this program illustrates an example where C, D, E, ..., C are each output in order for one second.

(1) Package description

<Symbols declared as public>

MLDY: Subroutine name of the musical scale generation program

<Registers used>

Bank 0: A, B, HL

<RAM used>

Name	Use	Attribute	Byte
POINT	Save the pointer of table data.	SADDR	1
LNG	Count the length of the sound data.		

<Nesting>

1 level, 3 bytes

<Hardware used>

- 8-bit timer/event counters 1 and 2
- P31/TO1

<Initial settings>

- Subroutine MLDY is set.
- Interrupts enabled

<Startup procedure>

- Call subroutine MLDY.

(2) Use example

```
EXTRN   MLDY
   :
CALL    !MLDY
EI
```

(3) SPD chart

MLDY
- Set P31/TO1 to the output mode.
- Set the pointer (POINT) to the reference table to 0.
- Set the length of the sound data (LNG) to the initial data of 1.
- 8-bit timer/event counter 1 is set to the output mode.
- 8-bit timer/event counter 2 is set to 20 ms.
- 8-bit timer 2 interrupt enabled.

INTTM2
- Select register bank 0
- Decrement the length of the sound data (LNG)
- IF : At the end of the output time
  - THEN
    - Look up sound data indicated by the pointer
    - IF : Sound data ≠ No-sound data
      - THEN
        - Set sound data in the compare register of timer 1
      - ELSE
        - Disable TO1 output of timer 1
    - Look up the length data of the sound
    - IF : Length of sound data ≠ End of musical scale generation data
      - THEN
        - Set the length of sound data
      - ELSE
        - Timer 2 interrupt disabled
          - Timer 2 operation stopped

(4) Program listing

```
        PUBLIC   MLDY

VETM2   CSEG     AT 18H
        DW       INTTM2             ;  Setting the vector address of 8-bit timer/event counter 2
ML_DAT  DSEG     SADDR
POINT:  DS       1                  ; Pointer to table data
LNG:    DS       1                  ; Length data of sound

;************************************************
;*   Musical scale generation initialization
;************************************************
ML_SEG  CSEG
MLDY:
        CLR1     PM3.1              ; Set bit 1 of port 3 in output mode.
        POINT=#0                    ; Initial setting of the pointer
        LNG=#1
        TOC1=#00000011B             ; Set to the TO1 output mode.
        TCL1=#11011001B
        CR20=#163                   ; Set timer 2 to 20 ms.
        TMC1=#00000010B             ; Timer 2 operation enabled
        CLR1     TMMK2              ; Timer 2 interrupt enabled
        RET
$EJECT
```

```
;*****************************************
;*   Setting musical scale generation
;*****************************************
TM2_SEG CSEG
INTTM2:
        SEL RB0
        LNG--
        if(LNG==#0)
            B=POINT (A)
            HL=#TABLE                    ; Setting the start address of the table
            A=[HL+B]
            if(A!=#0)
                CLR1    TCE1             ; Sound data setting
                CR10=A
                SET1    TOE1
                SET1    TCE1
            else
                CLR1    TOE1
            endif

            B++                          ; Increment pointer.
            A=[HL+B]                     ; Read the length data of the sound
            if(A!=#0)                    ; Is the sound being output?
                LNG=A                    ; Sound length data setting
                B++
                POINT=B (A)
            else
                SET1    TMMK2            ; Timer 2 interrupt disabled
                CLR1    TCE2             ; Timer 2 operation stopped
            endif
        endif
        RETI
;*****************************************
;*        Musical scale data table
;*****************************************
TABLE:
        DB      124,50                   ; C
        DB      111,50                   ; D
        DB      98,50                    ; E
        DB      93,50                    ; F
        DB      83,50                    ; G
        DB      73,50                    ; A
        DB      65,50                    ; B
        DB      62,50                    ; C
        DB      00,00                    ; End
```

CHAPTER 7 WATCH TIMER APPLICATION

The 78K/0 Series watch timer has a watch timer function that uses as the source signal the main system clock or the subsystem clock and overflows every 0.5 seconds, and an interval timer function capable of setting six types of basic times. These two functions can be used at the same time.

The watch timer is set by timer clock selection register 2 (TCL2) and watch timer mode control register (TMC2).

Figure 7-1. Format of Timer Clock Selection Register 2

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
TCL2	TCL27	TCL26	TCL25	TCL24	0	TCL22	TCL21	TCL20	FF42H	00H	R/W

TCL22	TCL21	TCL20	Count clock selection	
			Watchdog timer mode	Interval timer mode
0	0	0	$f_X/2^3$ (625 kHz)	$f_X/2^4$ (313 kHz)
0	0	1	$f_X/2^4$ (313 kHz)	$f_X/2^5$ (156 kHz)
0	1	0	$f_X/2^5$ (156 kHz)	$f_X/2^6$ (78.1 kHz)
0	1	1	$f_X/2^6$ (78.1 kHz)	$f_X/2^7$ (39.1 kHz)
1	0	0	$f_X/2^7$ (39.1 kHz)	$f_X/2^8$ (19.5 kHz)
1	0	1	$f_X/2^8$ (19.5 kHz)	$f_X/2^9$ (9.8 kHz)
1	1	0	$f_X/2^9$ (9.8 kHz)	$f_X/2^{10}$ (4.9 kHz)
1	1	1	$f_X/2^{11}$ (2.4 kHz)	$f_X/2^{12}$ (1.2 kHz)

TCL24	Count clock selection for the watch timer[Note]
0	$f_X/2^8$ (19.5 kHz)
1	f_{XT} (32.768 kHz)

TCL27	TCL26	TCL25	Selection of the buzzer output frequency
0	x	x	Buzzer output prohibited
1	0	0	$f_X/2^{10}$ (4.9 kHz)
1	0	1	$f_X/2^{11}$ (2.4 kHz)
1	1	0	$f_X/2^{12}$ (1.2 kHz)
1	1	1	Setting prohibited

Note When a main system clock at 1.25 MHz or lower and an FIP controller/driver are used simultaneously, select $f_X/2^8$ as the count clock for the watch timer.

Caution When TCL2 will be rewritten with data other than identical data, rewrite after temporarily stopping timer operation.

Remarks 1. f_X : Main system clock oscillation frequency

2. f_{XT} : Subsystem clock oscillation frequency

3. x : Don't care

4. The values in parentheses apply to operation with f_X = 5.0 MHz or f_{XT} = 32.768 kHz.

Figure 7-2. Format of the Watch Timer Mode Control Register

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
TMC2	0	TMC26	TMC25	TMC24	TMC23	TMC22	TMC21	TMC20	FF4AH	00H	R/W

TMC23	TMC20	Selection of the set time for the watch flag
0	0	$2^{14}/f_W$ (0.5 s)
1		$2^{13}/f_W$ (0.25 s)
0	1	$2^{5}/f_W$ (977 µs)
1		$2^{4}/f_W$ (488 µs)

TMC21	Control of prescaler operation[Note]
0	Clear after stopping operation
1	Operation enabled

TMC22	Control operation of a 5-bit counter
0	Clear after stopping operation
1	Operation enabled

TMC26	TMC25	TMC24	Selection of the prescaler interval time
0	0	0	$2^{4}/f_W$ (488 µs)
0	0	1	$2^{5}/f_W$ (977 µs)
0	1	0	$2^{6}/f_W$ (1.95 ms)
0	1	1	$2^{7}/f_W$ (3.91 ms)
1	0	0	$2^{8}/f_W$ (7.81 ms)
1	0	1	$2^{9}/f_W$ (15.6 ms)
Other than the above			Setting prohibited

Caution When a watch timer is used, do not frequently clear the prescaler.

Remarks 1. f_W: Clock frequency of the watch timer ($f_X/2^8$ or f_{XT})

2. The values in parentheses apply to operation with f_W = 32.768 kHz.

7.1 WATCH AND LED DISPLAY PROGRAM

An example using the watch timer is illustrated for a time count using the 0.5-second overflow and LED dynamic display using the 1.95-ms interval.

When the time count tests the overflow flag each time a subroutine is called. When it is set, count up processing of seconds is performed. Because overflow is generated at 0.5 s, when there are 120 counts, 1 minute is counted. The overflow test is performed at 1.95-ms intervals in order not to lose data. The watch display of this program is a 24-hour display. Minute data and hour data are separately stored in memory as the high-order and low-order digits.

Figure 7-3. Schematic of Watch Data

Seconds data	Minutes data		Hours data	
0-120	Low order 0-9	High order 0-5	Low order 0-9	High order 0-2

An LED dynamic display is a four-digit display that switches the display digits in each 1.95-ms interval. In this example, the high-order four bits of P3 in the digit signal selects P12 where the LEDs in the segment signal can be driven directly.

The LED display displays the digits shown in the display digit area (DIGCT) of the LED display area (LEDDP). Also, when the digit signal switches, switching is performed after the segment signal is turned off in order not to shift neighboring digit displays.

Figure 7-4. LED Display Timing

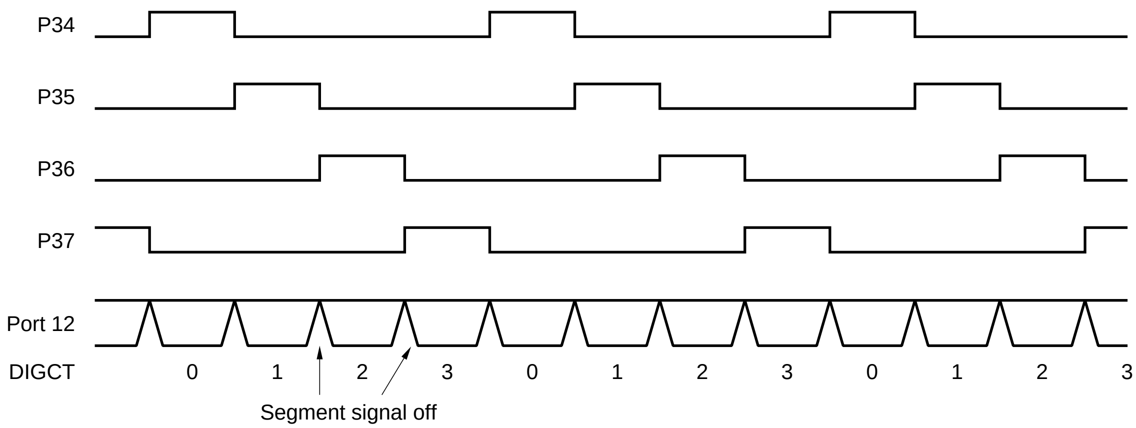
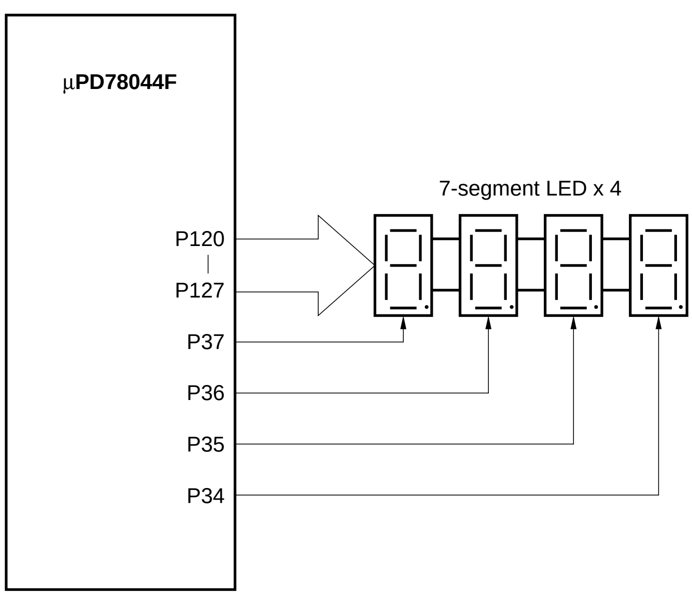


Figure 7-5. Example Circuit of the Watch Timer



(1) Package description

<Symbols declared as public>

SECD : Area storing seconds data
MINDP : Area storing minutes data
HOURDP : Area storing hours data
LEDDP : LED display area

<Registers used>

Bank 0: AX, B, HL

<RAM used>

Name	Use	Attribute	Byte
MINDP	Minutes data storage	SADDRP	2
HOURDP	Hours data storage		
SECD	Seconds data storage		1
DIGCT	LED display digit data storage		
LEDDP	LED display data		4

<Hardware used>

- Watch timer
- P34-37
- P12

<Initial settings>

- Watch operation of 0.5 s, interval of 1.95 ms TMC2 = #00100110B
- Watch timer interrupt enabled TMMK3 = 0

<Startup procedure>

Startup is by the interval timer interrupt request of the watch timer.

(2) Use example

```
EXTRN  MINDP,HOURDP,SECD,LEDDP

TMC2 = #00100110B                 ; 0.5-s watch operation, 1.95-ms interval
CLR1   TMMK3                      ; Watch timer interrupt enabled
EI
```

(3) SPD chart

INTTM3
- Select register bank 0
- Time count TIME
- LEDs display LEDDSP

LEDDSP
- Turn segment signal off
- IF : Digit counter (DIGCT) = 0
  - THEN
    - Initial setting of the digit signal
  - ELSE
    - Shift the digit signal to the one bit in the high-order direction
- The segment signal that displays the digit counter is output
- Increment the digit counter

TIME
- IF : The watch timer interrupt request flag is set
  - THEN
    - Increment seconds counter
    - IF : Seconds counter=120
      - THEN
        - Set the seconds counter to 0
        - Minutes (low order) counter increment
        - IF : Minutes (low order) counter=10
          - THEN
            - Set minutes (low order) counter to 0
            - Increment minutes (high order) counter
            - IF : Minutes (high order) counter=6
              - THEN
                - Set the minutes (high order) counter to 0
                - Increment hours (high order) counter
                - IF : Time data ≠ 0204H
                  - THEN
                    - IF : Hours (low order) counter=10
                      - THEN
                        - Set the hours (low order) counter to 0
                        - Increment the hours (high order) counter
                  - ELSE
                    - Set the hours counter to 0

(4) Program listing

```
        PUBLIC  HOURDP,MINDP,SECD,LEDDP

WT_DATP DSEG    SADDRP
MINDP:  DS      2                       ; Area storing minutes data
HOURDP: DS      2                       ; Area storing hours data
SECD:   DS      1                       ; Area storing seconds data
DIGCT:  DS      1                       ; LEDs display digit area
LEDDP:  DS      4                       ; LEDs display area

VETM3   CSEG    AT 12H
        DW      INTTM3                  ; Setting the vector address of the watch timer

;***************************************
;*   Interval interrupt processing
;***************************************
TM3_SEG CSEG
INTTM3:
        SEL RB0
        CALL    !TIME
        CALL    !LEDDPSP
        RETI
```

```
;********************
;*    LED display
;********************
LEDDPSP:
        P12=#0FFH                       ; Segment output off
        DIGCT&=#00000011B               ; Adjustment of digit counter (0 to 3)
        if(DIGCT==#0)
            A=P3
            A&=#00001111B               ; Initial setting of digit signal (high-order 4 bits)
            A!=#00010000B
            P3=A
        else
            A=P3
            A&=#11110000B               ; Shift high-order 4 bits.
            X=A
            A=P3
            A+=X
            P3=A
        endif

        B=DIGCT (A)                     ; Address setting of display data
        HL=#LEDDP                       ; Start address of the display area
        B=[HL+B] (A)                    ; Display data setting
        HL=#SEGDT                       ; Change to segment data.
        P12=[HL+B] (A)                  ; Segment signal output

        DIGCT++
        RET
SEGDT:
        DB      11000000B               ; 0
        DB      11111001B               ; 1
        DB      10100100B               ; 2
        DB      10110000B               ; 3
        DB      10011001B               ; 4
        DB      10010010B               ; 5
        DB      10000010B               ; 6
        DB      11111000B               ; 7
        DB      10000000B               ; 8
        DB      10010000B               ; 9
        DB      10001000B               ; A
        DB      10000011B               ; B
        DB      11000110B               ; C
        DB      10100001B               ; D
        DB      10000110B               ; E
        DB      10001110B               ; F
$EJECT
```

```
;***********************
;*   Watch count up
;***********************
TIME:
      if_bit(WTIF)                                  ; 0.5-s test
          CLR1    WTIF
          SECD++                                    ; 120 = 60 s/0.5
          if(SECD==#120)
              SECD=#0
              (MINDP+0)++                           ; Increment low-order part of minutes.
              if((MINDP+0)==#10)                    ; Digit carry
                  (MINDP+0)=#0
                  (MINDP+1)++                       ; Increment high-order part of minutes.
                  if(MINDP+1==#6)                   ; Digit carry
                      (MINDP+1)=#0
                      (HOURDP+0)++
                      if(HOURDP!=#0204H) (AX)       ; Is hour data 24?
                          if((HOURDP+0)==#10)       ; Digit carry
                             (HOURDP+0)=#0
                             (HOURDP+1)++
                          endif
                      else
                          HOURDP=#0000H
                      endif
                  endif
              endif
          endif
      endif
      RET
```

[MEMO]

CHAPTER 8 SERIAL INTERFACE APPLICATION

Table 8-1 lists the serial interfaces of the 78K/0 Series.

*

Table 8-1. Available Serial Interface Channels in Each Subseries

Serial interface configuration / Subseries	Channel 0			Channel 1		Channel 3
	3-wire	2-wire	SBI	3-wire	3-wire mode with automatic transmission/ reception function	3-wire
μPD78044F	o	o	o	o	o	x
μPD78044H	x	x	x	o	x	x
μPD780208	o	o	o	o	o	x
μPD780228	x	x	x	x	x	o

Remark o: Function available x: Function not available

The serial interface requires the setting of the following registers.

*

Table 8-2. Serial Interface Registers

Serial interface	Registers to be used
Channel 0	• Timer clock selection register (TCL3) • Serial operating mode register 0 (CSIM0) • Serial bus interface control register (SBIC) • Interrupt timing specification register (SINT)
Channel 1	• Timer clock selection register (TCL3) • Serial operating mode register 1 (CSIM1) • Automatic data transmit/receive control register (ADTC) • Automatic data transmit/receive interval setting register (ADTI)

Remark This chapter describes only the register formats and sample applications for serial interface channels 0 and 1. For details of the register format for channel 3, refer to the μ**PD780228 Subseries User's Manual** (U12012E).

Figure 8-1. Format of Timer Clock Selection Register 3 (μPD78044F and μPD780208 Subseries)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
TCL3	TCL37	TCL36	TCL35	TCL34	TCL33	TCL32	TCL31	TCL30	FF43H	88H	R/W

TCL33	TCL32	TCL31	TCL30	Selection of serial clock for serial interface channel 0
0	1	1	0	$f_X/2^2$ (1.25 MHz)
0	1	1	1	$f_X/2^3$ (625 kHz)
1	0	0	0	$f_X/2^4$ (313 kHz)
1	0	0	1	$f_X/2^5$ (156 kHz)
1	0	1	0	$f_X/2^6$ (78.1 kHz)
1	0	1	1	$f_X/2^7$ (39.1 kHz)
1	1	0	0	$f_X/2^8$ (19.5 kHz)
1	1	0	1	$f_X/2^9$ (9.8 kHz)
Other than the above				Setting prohibited

TCL37	TCL36	TCL35	TCL34	Selection of serial clock for serial interface channel 1
0	1	1	0	$f_X/2^2$ (1.25 MHz)
0	1	1	1	$f_X/2^3$ (625 kHz)
1	0	0	0	$f_X/2^4$ (313 kHz)
1	0	0	1	$f_X/2^5$ (156 kHz)
1	0	1	0	$f_X/2^6$ (78.1 kHz)
1	0	1	1	$f_X/2^7$ (39.1 kHz)
1	1	0	0	$f_X/2^8$ (19.5 kHz)
1	1	0	1	$f_X/2^9$ (9.8 kHz)
Other than the above				Setting prohibited

Caution When data other than the same data is rewritten into TCL3, rewrite after temporarily stopping timer operation.

Remarks 1. f_X: Main system clock oscillation frequency

2. The values in parentheses apply to operation with f_X = 5.0 MHz.

*

Figure 8-2. Format of Timer Clock Selection Register 3 (μPD78044H Subseries)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
TCL3	TCL37	TCL36	TCL35	TCL34	0	0	0	0	FF43H	88H	R/W[Note]

TCL37	TCL36	TCL35	TCL34	Selection of serial clock for serial interface channel 1
0	1	1	0	$f_X/2^2$ (1.25 MHz)
0	1	1	1	$f_X/2^3$ (625 kHz)
1	0	0	0	$f_X/2^4$ (313 kHz)
1	0	0	1	$f_X/2^5$ (156 kHz)
1	0	1	0	$f_X/2^6$ (78.1 kHz)
1	0	1	1	$f_X/2^7$ (39.1 kHz)
1	1	0	0	$f_X/2^8$ (19.5 kHz)
1	1	0	1	$f_X/2^9$ (9.8 kHz)
Other than the above				Setting prohibited

Note Bits 0 to 3 are read-only.
When bits 0 to 3 are read, the operation will become unpredictable.

Caution When data other than the same data is rewritten into TCL3, rewrite after temporarily stopping timer operation.

Remarks 1. f_X: Main system clock oscillation frequency

2. The values in parentheses apply to operation with f_X = 5.0 MHz.

Figure 8-3. Format of Serial Operating Mode Register 0 (Only for the μPD78044F and μPD780208 Subseries) (1/2)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
CSIM0	CSIE0	COI	WUP	CSIM04	CSIM03	CSIM02	CSIM01	CSIM00	FF60H	00H	R/W[Note 1]

R/W

CSIM01	CSIM00	Clock selection for serial interface channel 0
0	x	Input clock from the outside to pin $\overline{SCK0}$
1	0	Output of the 8-bit timer register 2 (TM2)
1	1	Clock specified by bits 0 to 3 of the timer clock selection register 3 (TCL3)

R/W

CSIM04	CSIM03	CSIM02	PM25	P25	PM26	P26	PM27	P27	Operating mode	First bit	SI0/SB0/P25 pin function	SO0/SB1/P26 pin function	$\overline{SCK0}$/P27 pin function
0	x	0	1	x	0	0	0	1	3-wire serial I/O mode	MSB	SI0[Note 2] (input)	SO0 (CMOS output)	$\overline{SCK0}$ (CMOS I/O)
		1								LSB			
1	0	0	Note 3 x	Note 3 x	0	0	0	1	SBI mode	MSB	P25 (CMOS I/O)	SB1 (N-channel open drain I/O)	$\overline{SCK0}$ (CMOS I/O)
		1	0	0	Note 3 x	Note 3 x	0	1			SB0 (N-channel open drain I/O)	P26 (CMOS I/O)	
1	1	0	Note 3 x	Note 3 x	0	0	0	1	2-wire serial I/O mode	MSB	P25 (CMOS I/O)	SB1 (N-channel open drain I/O)	$\overline{SCK0}$ (N-channel open drain I/O)
		1	0	0	Note 3 x	Note 3 x	0	1			SB0 (N-channel open drain I/O)	P26 (CMOS I/O)	

R/W

WUP	Wakeup function control[Note 4]
0	An interrupt request signal is issued at each serial transfer in all of the modes.
1	When the address received after the bus release in the SBI mode (when CMDD = RELD = 1) matches the data in the slave address register, an interrupt request signal is issued.

Notes 1. Bit 6 (COI) is read-only.

2. When only transmission is performed, this pin can be used as P25 (CMOS input).

3. This pin can be used for a port function.

* **4.** When the wakeup function is used (WUP = 1), set bit 5 (SIC) of the interrupt timing specification register (SINT) to 0.

* **Caution The operating mode (3-wire serial I/O, 2-wire serial I/O, or SBI mode) must not be changed while serial interface channel 0 is enabled. To change the operating mode, temporarily stop serial operation beforehand.**

Remark x : Don't care
PMxx : Port mode register
Pxx : Port output latch

Figure 8-3. Format of Serial Operating Mode Register 0 (Only for the μPD78044F and μPD780208 Subseries) (2/2)

R	COI	Slave address comparison result flag[Note]
	0	Data in the slave address register and data in serial I/O shift register 0 do not match.
	1	Data in the slave address register and data in serial I/O shift register 0 match.

R/W	CSIE0	Control of serial interface channel 0 operation
	0	Stop operation
	1	Enable operation

Note When CSIE0 = 0, COI becomes 0.

* **Caution The operating mode (3-wire serial I/O, 2-wire serial I/O, or SBI mode) must not be changed while serial interface channel 0 is enabled. To change the operating mode, temporarily stop serial operation beforehand.**

Figure 8-4. Format of the Serial Operating Mode Register 1 (μPD78044F and μPD780208 Subseries)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
CSIM1	CSIE1	DIR	ATE	0	0	0	CSIM11	CSIM10	FF68H	00H	R/W

CSIM11	CSIM10	Clock selection for serial interface channel 1
0	x	External clock input[Note 1] to $\overline{SCK1}$ pin
1	0	Output of 8-bit timer register 2 (TM2)
1	1	Clock set by bits 4 to 7 of timer clock selection register 3 (TCL3)

ATE	Selection of operating mode of serial interface channel 1
0	3-wire serial I/O mode
1	3-wire serial I/O mode with automatic transmit/receive function

DIR	First bit	SI1 pin function	SO1 pin function
0	MSB	SI1/P20 (input)	SO1 (CMOS output)
1	LSB		

CSIE1	CSIM11	PM20	P20	PM21	P21	PM22	P22	Shift register 1 operation	Control of serial clock counter operation	SI1/P20 pin function	SO1/P21 pin function	$\overline{SCK1}$/P22 pin function
0	x	x[Note 2]	x[Note 2]	x[Note 2]	x[Note 2]	x[Note 2]	x[Note 2]	Stop operation	Clear	P20 (CMOS I/O)	P21 (CMOS I/O)	P22 (CMOS I/O)
1	0	1[Note 3]	x[Note 3]	0	0	1	x	Enable operation	Count operation	SI1[Note 3] (input)	SO1 (CMOS output)	$\overline{SCK1}$ (input)
	1					0	1					$\overline{SCK1}$ (CMOS output)

Notes 1. When an external clock input is selected and CSIM11 is 0, set bits 2 and 1(STRB and BUSY1) of the automatic data transmit/receive control register (ADTC) to 0.

2. This can be used for a port function.

* **3.** When only transmission is performed, this can be used as P20. Set bit 7 (RE) of ADTC to 0.

Remark x : Don't care
PMxx : Port mode register
Pxx : Port output latch

* **Figure 8-5. Format of the Serial Operating Mode Register 1 (μPD78044H Subseries)**

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
CSIM1	CSIE1	DIR	0[Note 1]	0	0	0	CSIM11	CSIM10	FF68H	00H	R/W

CSIM11	CSIM10	Clock selection for serial interface channel 1
0	x	External clock input to $\overline{SCK1}$ pin
1	0	Output of 8-bit timer register 2 (TM2)
1	1	Clock set by bits 4 to 7 of timer clock selection register 3 (TCL3)

DIR	First bit	SI1 pin function	SO1 pin function
0	MSB	SI1/P20 (input)	SO1 (CMOS output)
1	LSB		

CSIE1	CSIM11	PM20	P20	PM21	P21	PM22	P22	Shift register 1 operation	Control of serial clock counter operation	SI1/P20 pin function	SO1/P21 pin function	$\overline{SCK1}$/P22 pin function
0	x	Note 2 x	Note 2 x	Note 2 x	Note 2 x	Note 2 x	Note 2 x	Stop operation	Clear	P20 (CMOS I/O)	P21 (CMOS I/O)	P22 (CMOS I/O)
1	0	Note 3 1	Note 3 x	0	0	1	x	Enable operation	Count operation	SI1[Note 3] (input)	SO1 (CMOS output)	$\overline{SCK1}$ (input)
	1					0	1					$\overline{SCK1}$ (CMOS output)

Notes 1. Always set to 0.

2. This can be used for a port function.

3. When only transmission is performed, this can be used as P20 (CMOS I/O).

Remark x : Don't care
PMxx : Port mode register
Pxx : Port output latch

Figure 8-6. Format of the Interrupt Timing Setting Register (Only for the µPD78044F and µD780208 Subseries)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
SINT	0	CLD	SIC	SVAM	0	0	0	0	FF63H	00H	R/W[Note 1]

R/W

SVAM	SVA bits used as the slave address
0	Bits 0 to 7
1	Bits 1 to 7

R/W

SIC	Selection of the cause of the INTCSI0 interrupt[Note 2]
0	Set CSIIF0 at the end of a transfer in serial interface channel 0.
1	Set CSIIF0 at the end of a transfer in serial interface channel 0 or when a bus release is detected.

R

CLD	Level of $\overline{\text{SCK0}}$/P27 pin[Note 3]
0	Low level
1	High level

Notes 1. Bit 6 (CLD) is read-only.

2. When the wakeup function is used, set SIC to 0.

3. When CSIE0 = 0, CLD becomes 0.

Caution Always set bits 0 to 3 to 0.

Remark SVA : Slave address register

CSIIF0 : Interrupt request flag which corresponds to INTCSI0

CSIE0 : Bit 7 of serial operating mode register 0 (CSIM0)

Figure 8-7. Format of the Serial Bus Interface Control Register (Only for the μPD78044F and μPD780208 Subseries) (1/2)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
SBIC	BSYE	ACKD	ACKE	ACKT	CMDD	RELD	CMDT	RELT	FF61H	00H	R/W[Note]

R/W	RELT	This is used to output the bus release signal. The SO latch is set (1) by RELT = 1. After setting the SO latch, this bit is automatically cleared (0). In addition, it is cleared (0) when CSIE0 = 0.

R/W	CMDT	This is used for command signal output. The SO latch is cleared (0) by CMDT = 1. After clearing the SO latch, this bit is automatically cleared (0). In addition, it is cleared (0) when CSIE0 = 0.

R — RELD: Bus release detection

Clearing conditions (RELD = 0)	Setting conditions (RELD = 1)
• When a start transfer instruction is executed • When the values in SIO0 and SVA do not match while receiving an address • When CSIE0 = 0 • When $\overline{\text{RESET}}$ is input	• When a bus release signal (REL) is detected

R — CMDD: Command detection

Clearing conditions (CMDD = 0)	Setting conditions (CMDD = 1)
• When a start transfer instruction is executed • When a bus release signal (REL) is detected • When CSIE0 = 0 • When $\overline{\text{RESET}}$ is input	• When a command signal (CMD) is detected

R/W	ACKT	The acknowledge signal is output synchronized to the falling edge of the $\overline{\text{SCK0}}$ clock immediately after the execution of the instruction that is set (1). After the output, this bit is automatically cleared (0). In addition, when starting the transfer in the serial interface and CSIE0 = 0, this bit is also cleared (0).

Note Bits 2, 3, and 6 (RELD, CMDD, ACKD) are read-only.

Remark CSIE0: Bit 7 of serial operating mode register 0 (CSIM0)

Figure 8-7. Format of the Serial Bus Interface Control Register (Only for the μPD78044F and μPD780208 Subseries) (2/2)

R/W	ACKE	Acknowledge signal output control	
	0	Automatic output of the acknowledge signal is disabled. (Output by ACKT is possible.)	
	1	Before the end of transfer	The acknowledge signal is output synchronized to the falling edge of the ninth $\overline{\text{SCK0}}$ clock (automatically output by ACKE = 1).
		After the transfer ends	The acknowledge signal is output synchronized to the falling edge of the $\overline{\text{SCK0}}$ clock immediately after the execution of the instruction that is set (1) (automatically output by ACKE = 1). However, after the acknowledge signal is output, this is not automatically cleared (0).

R	ACKD	Acknowledge detection
	Clearing conditions (ACKD = 0)	Setting conditions (ACKD = 1)
	• When a falling edge of the $\overline{\text{SCK0}}$ clock occurs immediately after the busy mode was released after executing a start transfer instruction • When CSIE0 = 0 • When $\overline{\text{RESET}}$ is input	• When the acknowledge signal ($\overline{\text{ACK}}$) is detected at the rising edge of the $\overline{\text{SCK0}}$ clock after the transfer ends

R/W	BSYE[Note]	Control of synchronous busy signal output
	0	The output of the busy signal is disabled synchronous to the falling edge of the $\overline{\text{SCK0}}$ clock immediately after executing the instruction that is cleared (0).
	1	The busy signal is output starting at the falling edge of the $\overline{\text{SCK0}}$ clock following the acknowledge signal.

Note The busy mode can be released at the start of transfer in the serial interface. However, the BSYE flag is not cleared to 0.

Remark CSIE0 : Bit 7 of serial operating mode register 0 (CSIM0)

Figure 8-8. Format of the Automatic Data Transmit/Receive Control Register (Only for the μPD78044F and μPD780208 Subseries)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
ADTC	RE	ARLD	ERCE	ERR	TRF	STRB	BUSY1	BUSY0	FF69H	00H	R/W[Note 1]

R/W

BUSY1	BUSY0	Busy input control
0	x	Busy input is not used.
1	0	Busy input enabled (active high)
1	1	Busy input enabled (active low)

R/W

STRB	Strobe output control
0	Strobe output disabled
1	Strobe output enabled

R

TRF	Status of the automatic transmit/receive function[Note 2]
0	Detect the end of an automatic transmit/receive. (When interrupting automatic transmit/receive or when ARLD = 0, this bit becomes 0.)
1	During automatic transmit/receive (By writing to SIO1, this becomes 1.)

R

ERR	Error detection of the automatic transmit/receive function
0	No error during automatic transmit/receive (By writing to SIO1, this bit becomes 0.)
1	Error present during automatic transmit/receive

R/W

ERCE	Control of the error checking of the automatic transmit/receive function
0	Error checking disabled during automatic transmit/receive
1	Error checking enabled during automatic transmit/receive (only when BUSY1 = 1)

R/W

ARLD	Selection of the operating mode of the automatic transmit/receive function
0	Single-shot mode
1	Repeat mode

R/W

RE	Receive control of the automatic transmit/receive function
0	Reception disabled
1	Reception enabled

Notes 1. Bits 3 and 4 (TRF, ERR) are read-only.

2. Make the decision on the end of automatic transmit/receive based on TRF and not on CSIIF1 (interrupt request flag).

(Continued on the next page)

Caution **When bit 1 (CSIM11) of serial operating mode register 1 (CSIM1) is set to 0 and the external clock input is selected, set STRB and BUSY1 in ADTC to 0.**

Remark x: Don't care

Figure 8-9. Format of the Automatic Data Transmit/Receive Interval Setting Register (Only for the µPD78044F and µPD780208 Subseries) (1/2)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
ADTI	ADTI7	0	0	ADTI4	ADTI3	ADTI2	ADTI1	ADTI0	FF6BH	00H	R/W

ADTI7	Control the interval time for data transfer
0	No control of the interval time by ADTI[Note 1]
1	Control of the interval time by ADTI (ADTI0 to ADTI4)

ADTI4	ADTI3	ADTI2	ADTI1	ADTI0	Setting the interval time for data transfer (during f_X = 5.0 MHz operation)	
					Minimum value[Note 2]	Maximum value[Note 2]
0	0	0	0	0	36.8 µs + 0.5/f_{SCK}	40.0 µs + 1.5/f_{SCK}
0	0	0	0	1	62.4 µs + 0.5/f_{SCK}	65.6 µs + 1.5/f_{SCK}
0	0	0	1	0	88.0 µs + 0.5/f_{SCK}	91.2 µs + 1.5/f_{SCK}
0	0	0	1	1	113.6 µs + 0.5/f_{SCK}	116.8 µs + 1.5/f_{SCK}
0	0	1	0	0	139.2 µs + 0.5/f_{SCK}	142.4 µs + 1.5/f_{SCK}
0	0	1	0	1	164.8 µs + 0.5/f_{SCK}	168.0 µs + 1.5/f_{SCK}
0	0	1	1	0	190.4 µs + 0.5/f_{SCK}	193.6 µs + 1.5/f_{SCK}
0	0	1	1	1	216.0 µs + 0.5/f_{SCK}	219.2 µs + 1.5/f_{SCK}
0	1	0	0	0	241.6 µs + 0.5/f_{SCK}	244.8 µs + 1.5/f_{SCK}
0	1	0	0	1	267.2 µs + 0.5/f_{SCK}	270.4 µs + 1.5/f_{SCK}
0	1	0	1	0	292.8 µs + 0.5/f_{SCK}	296.0 µs + 1.5/f_{SCK}
0	1	0	1	1	318.4 µs + 0.5/f_{SCK}	321.6 µs + 1.5/f_{SCK}
0	1	1	0	0	344.0 µs + 0.5/f_{SCK}	347.2 µs + 1.5/f_{SCK}
0	1	1	0	1	369.6 µs + 0.5/f_{SCK}	372.8 µs + 1.5/f_{SCK}
0	1	1	1	0	395.2 µs + 0.5/f_{SCK}	398.4 µs + 1.5/f_{SCK}
0	1	1	1	1	420.8 µs + 0.5/f_{SCK}	424.0 µs + 1.5/f_{SCK}

(Continued on the next page)

Notes 1. The interval time depends only on CPU processing.

2. Errors are contained in the interval time for data transfer. The minimum and maximum values of the interval time for each data transfer are determined from the following equations (n: values set in ADTI0 to ADTI4). However, when the minimum value calculated from the following equation is less than $2/f_{SCK}$, the minimum value of the interval time becomes $2/f_{SCK}$.

$$\text{minimum value} = (n + 1) \times \frac{2^7}{f_X} + \frac{56}{f_X} + \frac{0.5}{f_{SCK}}$$

$$\text{maximum value} = (n + 1) \times \frac{2^7}{f_X} + \frac{72}{f_X} + \frac{1.5}{f_{SCK}}$$

Cautions 1. Do not write to ADTI during operation of the automatic transmit-receive function.

2. Always set bits 5 and 6 to 0.

* **3. When ADTI is being used to control the interval time for data transfer performed using the automatic transmission/reception function, busy control is disabled.**

Remarks 1. f_X : Main system clock oscillation frequency

2. f_{SCK} : Serial clock frequency

Figure 8-9. Format of the Automatic Data Transmit/Receive Interval Setting Register (Only for the μPD78044F and μPD780208 Subseries) (2/2)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
ADTI	ADTI7	0	0	ADTI4	ADTI3	ADTI2	ADTI1	ADTI0	FF6BH	00H	R/W

ADTI4	ADTI3	ADTI2	ADTI1	ADTI0	Setting the interval time for data transfer (during f_X = 5.0 MHz operation)	
					Minimum value^Note	Maximum value^Note
1	0	0	0	0	446.4 μs + $0.5/f_{SCK}$	449.6 μs + $1.5/f_{SCK}$
1	0	0	0	1	472.0 μs + $0.5/f_{SCK}$	475.2 μs + $1.5/f_{SCK}$
1	0	0	1	0	497.6 μs + $0.5/f_{SCK}$	500.8 μs + $1.5/f_{SCK}$
1	0	0	1	1	523.2 μs + $0.5/f_{SCK}$	526.4 μs + $1.5/f_{SCK}$
1	0	1	0	0	548.8 μs + $0.5/f_{SCK}$	552.0 μs + $1.5/f_{SCK}$
1	0	1	0	1	574.4 μs + $0.5/f_{SCK}$	577.6 μs + $1.5/f_{SCK}$
1	0	1	1	0	600.0 μs + $0.5/f_{SCK}$	603.2 μs + $1.5/f_{SCK}$
1	0	1	1	1	625.6 μs + $0.5/f_{SCK}$	628.8 μs + $1.5/f_{SCK}$
1	1	0	0	0	651.2 μs + $0.5/f_{SCK}$	654.4 μs + $1.5/f_{SCK}$
1	1	0	0	1	676.8 μs + $0.5/f_{SCK}$	680.0 μs + $1.5/f_{SCK}$
1	1	0	1	0	702.4 μs + $0.5/f_{SCK}$	705.6 μs + $1.5/f_{SCK}$
1	1	0	1	1	728.0 μs + $0.5/f_{SCK}$	731.2 μs + $1.5/f_{SCK}$
1	1	1	0	0	753.6 μs + $0.5/f_{SCK}$	756.8 μs + $1.5/f_{SCK}$
1	1	1	0	1	779.2 μs + $0.5/f_{SCK}$	782.4 μs + $1.5/f_{SCK}$
1	1	1	1	0	804.8 μs + $0.5/f_{SCK}$	808.0 μs + $1.5/f_{SCK}$
1	1	1	1	1	830.4 μs + $0.5/f_{SCK}$	833.6 μs + $1.5/f_{SCK}$

Note Errors are contained in the interval time for data transfer. The minimum and maximum values of the interval time for each data transfer are determined from the following equations (n: values set in ADTI0 to ADTI4). However, when the minimum value calculated from the following equation is less than $2/f_{SCK}$, the minimum value of the interval time becomes $2/f_{SCK}$.

$$\text{minimum value} = (n + 1) \times \frac{2^7}{f_X} + \frac{56}{f_X} + \frac{0.5}{f_{SCK}}$$

$$\text{maximum value} = (n + 1) \times \frac{2^7}{f_X} + \frac{72}{f_X} + \frac{1.5}{f_{SCK}}$$

Cautions 1. Do not write to ADTI during operation of the automatic transmit/receive function.

2. Always set bits 5 and 6 to 0.

* **3. When ADTI is being used to control the interval time for data transfer performed using the automatic transmission/reception function, busy control is disabled.**

Remarks 1. f_X : Main system clock oscillation frequency

2. f_{SCK} : Serial clock frequency

8.1 INTERFACING WITH EEPROM™ (μPD6252)

The μPD6252[Note] is a 2048-bit electrically programmable and erasable ROM (EEPROM). Writing to and reading from the μPD6252 is performed through a 3-wire serial interface.

* **Note** The μPD6252 is provided for maintenance purposes only.

Figure 8-10. μPD6252 Pin Configuration

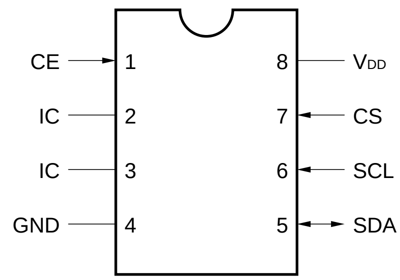
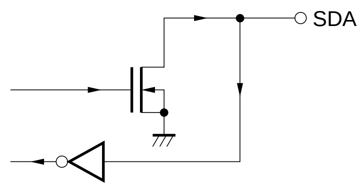


Table 8-3. Description of µPD6252 Pins

Pin number	Pin name	I/O	Function
1	CE	CMOS input	Set high during data transfer. **Caution Do not switch to this pin from high to low during data transfer.** When this pin is switched from high to low, operate in the state where the CS pin (pin 7) is set low. When pins CE and CS are both set to low levels, the standby state is entered. In the standby state, low power consumption results.
2 3	IC	-	Set the IC pin to a high or low level individually using an external resistor.
4	GND	-	Ground
5	SDA	CMOS input/ N-channel open-drain output	This pin is for data I/O. Attach a pull-up resistor externally for the N-channel open-drain I/O. 
6	SCL	CMOS input	This is the clock input pin for data transfer.
7	CS	CMOS input	This is the chip select pin. The µPD6252 can be operated by a high input. The read and write operations of a memory cell are not possible when at the low level. In the state where the SCL pin is high, this pin changes from low to high and the signal for starting operation of the serial bus interface results. In addition, when this pin changes from high to low, the signal of the end of operation of the serial bus interface results.
8	V_{DD}	-	Positive voltage (+5 V ±10%)

8.1.1 Communication in the 2-Wire Serial I/O Mode

The 3-wire system in the μPD6252[Note] indicates the three wires of the serial clock (SCL), data (SDA), and chip select (CS). Consequently, except for handshakes, because the required wires in an interface become the two clock and data wires, when the 78K/0 Series is used to establish an interface, the 2-wire serial I/O mode is selected. An example using the μPD78044F subseries is explained here.

* **Note** The μPD6252 is provided for maintenance purposes only.

Figure 8-11. μPD6252 Connection Example

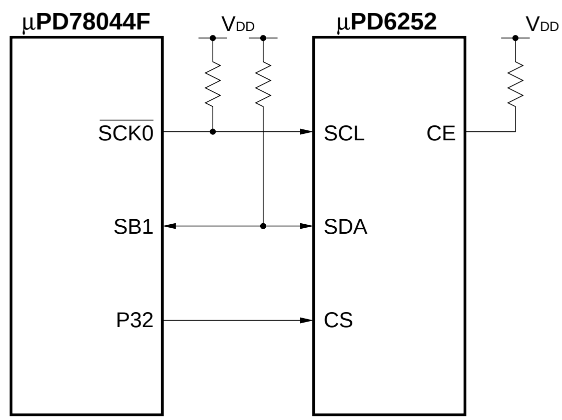


Table 8-4 and Figure 8-12 show the commands and communication formats when the μPD6252 is read and written.

Table 8-4. μPD6252 Command List

Command name	Command	Operation description
RANDOM WRITE	00000000B [00H] MSB C_7-C_0	After setting the word address (WA) (8 bits), write data is transferred. The write data are consecutive and a maximum of three bytes can be set. Correspondence of the word addresses WA : first data byte WA+1: second data byte WA+2: third data byte The write operation is executed during an internal write cycle after timing in which the CS pin falls from high to low.
CURRENT READ	10000000B [80H] MSB C_7-C_0	The memory contents, that are specified in the word address (WA) (current address) when the command was set, are sent to the read data buffer. When data is read from the SDA pin, the word address (WA) is incremented for every 8 bits read out and the corresponding memory contents are sent to the read data buffer.
RANDOM READ	11000000B [C0H] MSB C_7-C_0	After setting the word address (WA), a data read is executed with the set word address (WA) as the first address. The difference from CURRENT READ is the word address (WA) is set after the command is executed. After setting the word address (WA), the operation is identical to CURRENT READ.

Figure 8-12. µPD6252 Communication Format (1/2)

(1) RANDOM WRITE

(2) CURRENT READ

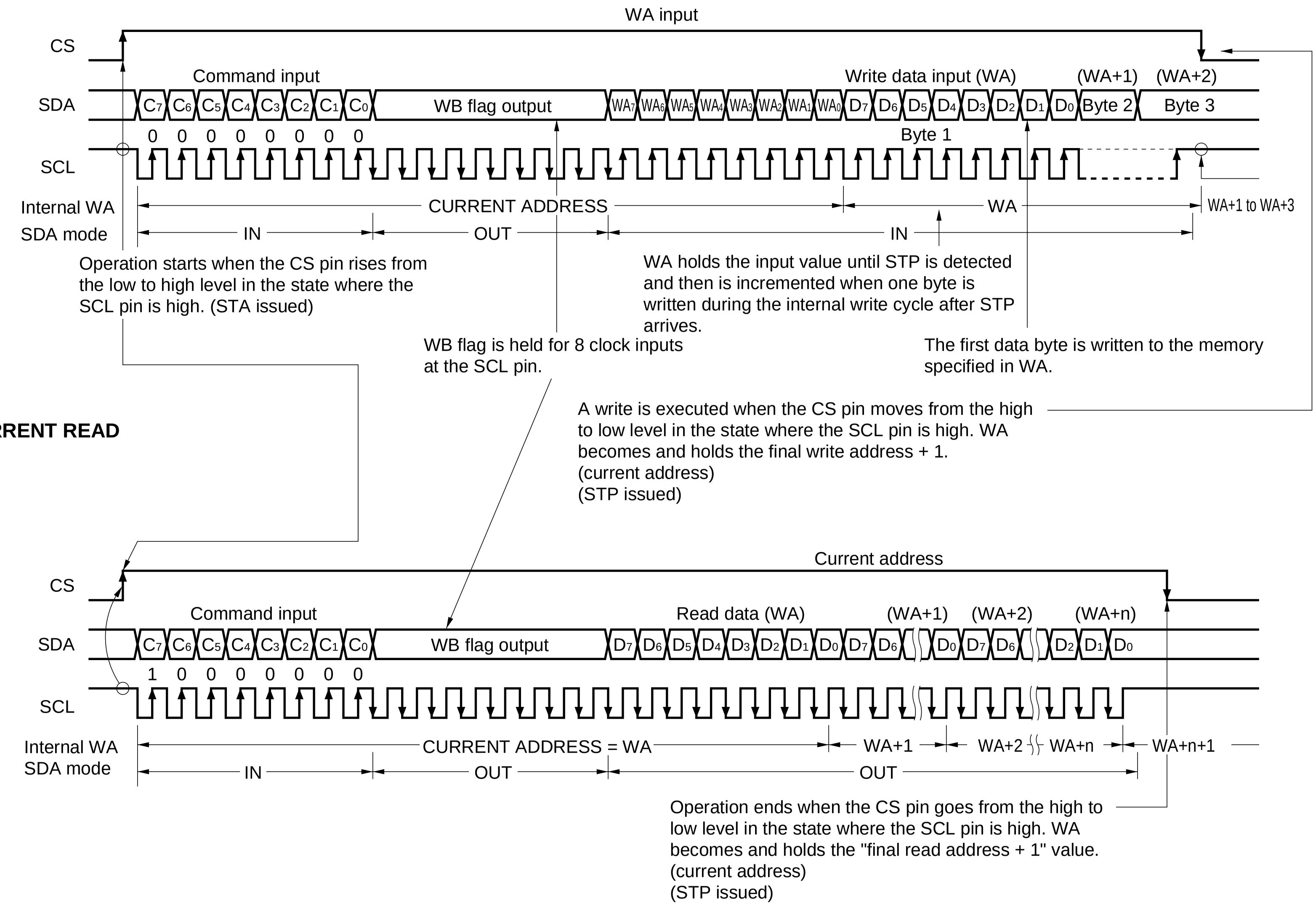
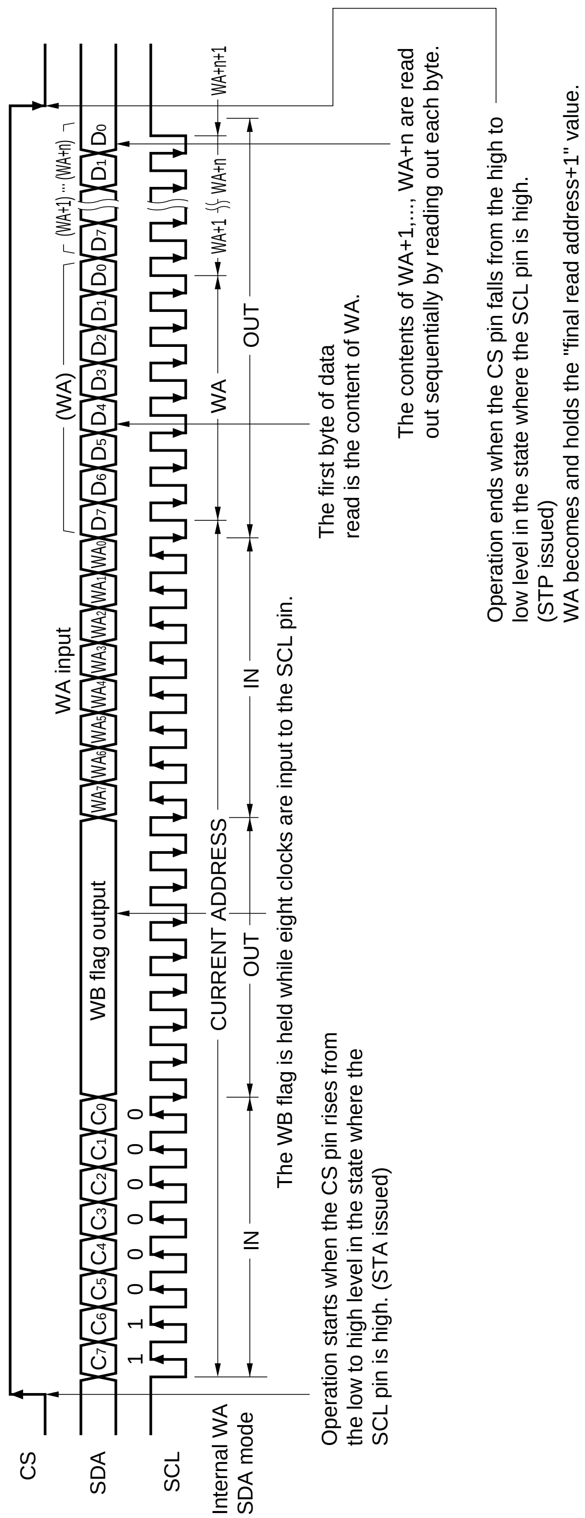


Figure 8-12. μPD6252 Communication Format (2/2)

(3) RANDOM READ



A program for the μPD6252 is illustrated in **<1>** to **<5>**. In this example, the number of data bytes in one write or read in the interface is fixed at one byte. In addition, when the μPD6252 is write busy (WB) while interfacing, the busy flag is set.

<1> The CS pin (P32) is set to the high level to initiate the interface.
<2> The write and read commands are transmitted.
<3> WRITE BUSY data is received. If in the state where interfacing with the μPD6252 is possible, 00H is received. When data other than 00H is received, the WRITE BUSY state is judged and processing to stop communication is performed.
<4> Data for the command is transferred.
<5> The CS pin (P32) is set low to end communication.

(1) Package description

<Symbols declared as public>

T3_6252 : Name of μPD6252 transfer subroutine
RWRITE : RANDOM WRITE command value
RREAD : RANDOM READ command value
CREAD : CURRENT READ command value
WADAT : Word address storage area
TRNDAT : Transmission data storage area
RCVDAT : Receive data storage area
CMDDAT : Command data storage area
BUSYFG : Busy state test flag
CS6252 : CS pin (P32) of μPD6252

<Registers used>

A

<RAM used>

Name	Use	Attributes	Bytes
WAADR	Stores the word address (before the transfer begins)	SADDR	1
TRNDAT	Stores the transmission data (before the transfer begins)		
RCVDAT	Stores the receive data (after the transfer ends)		
CMDDAT	Stores the command data (before the transfer begins)		

<Flag used>

Name	Use
BUSYFG	WRITE BUSY state setting

<Nesting>

1 level, 3 bytes

<Hardware used>

- Serial interface channel 0
- P32

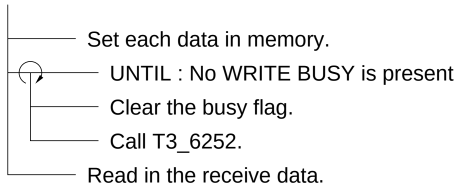
<Initial settings>

- Serial interface channel 0 settings
  2-wire serial I/O mode, SB1 pin selection CSIM0=#10011011B
- Serial clock $f_X/2^4$ TCL3=#xxxx1000B
- SB1 latch is the high level RELT=1

<Starting procedure>

Set the required data corresponding to command and T3_6252 is called. After returning from a subroutine, the busy flag (BUSYFG) is tested. When the busy flag is set, the transfer must be repeated because no transfer was performed. When in the receiving mode, after returning from a subroutine, the receive data is stored in RCVDAT.

(2) Use example




```
EXTRN    RWRITE,RREAD,CREAD
EXTRN    WADAT,TRNDAT,RCVDAT,CMDDAT,T3_6252
EXTBIT   BUSYFG,CS6252

    CSIM0=#10011011B              ; 2-wire serial I/O mode and SB1 pin settings
    TCL3=#10011000B               ; Set SCK0 = 262 kHz.
    CLR1     SB0
    CLR1     CS6252               ; Set the CS pin on the µPD6252 to the low level.
    CLR1     PM3.2

    CMDDAT=A
      :
      :
    WADAT=A
      :
      :
    TRNDAT=A
      :
      :
    repeat
             CLR1   BUSYFG
             CALL   !T3_6252
    until_bit(!BUSYFG)
      :
      :
    A=RCVDAT
```

(3) SPD chart

```
T3_6252
  Clear the busy flag.
  Issue the start bit.
  Transmit command.
  WHILE : Waiting for the end of transfer (CSIIF0 = 0)
  Busy signal received
  WHILE : Waiting for the end of transfer (CSIIF0 = 0)
  IF : Not in the WB state (SIO0 = 00H)
  THEN
      CASE : CMDDAT
      OF : RWRITE
          Transfer the word address.
          WHILE : Waiting for the end of transfer
          Transmit data.
          WHILE : Waiting for the end of transfer
          BREAK
      OF : RREAD
          Transfer the word address.
          WHILE : Waiting for the end of transfer
      OF : CREAD
          Receive data.
          WHILE : Waiting for the end of transfer
          Save the receive data in memory.
  ELSE
      Set in busy state.
        Set BUSYFG.
  Issue stop bit.
```

(4) Program listing

```
        PUBLIC  RWRITE,RREAD,CREAD
        PUBLIC  WADAT,TRNDAT,RCVDAT,CMDDAT,T3_6252
        PUBLIC  BUSYFG,CS6252
CSI_DAT DSEG    SADDR
WADAT:  DS      1                        ; Word address storage area
TRNDAT: DS      1                        ; Transmission data storage area
RCVDAT: DS      1                        ; Receive data storage area
CMDDAT: DS      1                        ; Command data storage area

CSI_FLG BSEG
BUSYFG  DBIT                             ; Busy state setting

RWRITE  EQU     00H                      ; RANDOM WRITE mode
RREAD   EQU     0C0H                     ; RANDOM READ mode
CREAD   EQU     080H                     ; CURRENT READ mode
CS6252  EQU     0FF03H.2                 ; 0FF03H=PORT3

CSI_SEG CSEG
;***************************************
;*   µPD6252 (3-wire) communication
;***************************************
T3_6252:
        CLR1    BUSYFG
        SET1    CS6252                   ; Issue the start bit.
        SIO0=CMDDAT (A)                  ; Transfer the command.
        while_bit(!CSIIF0)               ; Wait for the end of transfer.
        endw
        CLR1    CSIIF0
        SIO0=#0FFH                       ; Start reception of the busy signal.
        while_bit(!CSIIF0)               ; Wait for the end of transfer.
        endw
        CLR1    CSIIF0
        if(SIO0==#00H)                   ; Busy check
            switch (CMDDAT)
                case RWRITE:
                    SIO0=WADAT (A)       ; Transfer the word address.
                    while_bit(!CSIIF0)   ; Wait for the end of transfer.
                    endw
                    CLR1    CSIIF0
                    SIO0=TRNDAT (A)      ; Start the data transfer.
                    while_bit(!CSIIF0)   ; Wait for the end of transfer.
                    endw
                    CLR1    CSIIF0
                break

                case RREAD:
                    SIO0=WADAT (A)       ; Transfer the word address.
                    while_bit(!CSIIF0)   ; Wait for the end of transfer.
                    endw
                    CLR1    CSIIF0
```

```
        case CREAD:
            SIO0=#0FFH              ; Start data reception.
            while_bit(!CSIIF0)      ; Wait for the end of transfer.
            endw
            CLR1    CSIIF0
            RCVDAT=SIO0 (A)         ; Store the receive data.
    ends
else
    SET1    BUSYFG                  ; Set in the busy state
endif
CLR1    CS6252
RET
```

8.2 INTERFACING WITH THE OSD LSI (μPD6451A)

The μPD6451A, an OSD (On-Screen Display) LSI, displays VCR programming information or TV channels on a display by using it in conjunction with a microcontroller. To interface with the μPD6451A, the four lines of DATA, CLK, STB, and BUSY are used. An example using the μPD78044F subseries is described here.

Figure 8-13. Connection Example with μPD6451A

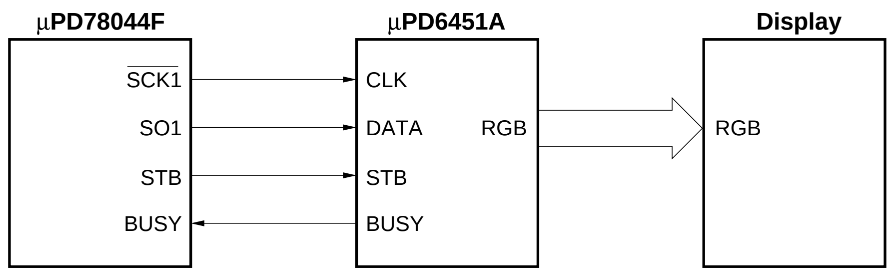
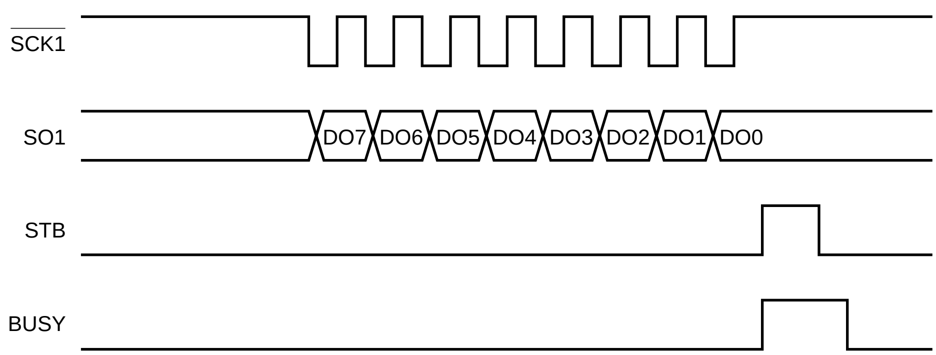


Figure 8-14. μPD6451A Communication Format



The output of the strobe signal (STB) and testing the busy signal (BUSY) used in handshaking for interfacing to the μPD6451A are automatically performed in serial interface channel 1 of the 78K/0 Series. To match the μPD6451A's communication format, the strobe signal output enable and busy signal input enable (active high) mode is selected. Data (maximum of 32 bytes) to be transmitted to the buffer RAM area (FAC0H-FADFH) are automatically transmitted when the number of data bytes to be transmitted is set at the automatic data transmit/receive address pointer (ADTP) and multiple bytes of data are consecutive.

(1) Package description

<Symbols declared as public>

TR6451 : Name of µPD6451A transfer subroutine
DTVAL : Area for setting the number of transmission data bytes

<Register used>

A

<RAM used>

Name	Use	Attributes	Bytes
DTVAL	Stores the number of bytes of transmission data	SADDR	1

<Nesting>

1 level, 2 bytes

<Hardware used>

- Serial interface channel 1

<Initial settings>

- Serial interface channel 1 settings
  Automatic transmit/receive operation enabled, MSB first — CSIM1=#10100011B
  Busy input enabled (active high), strobe output enabled,
  single shot mode — ADTC=#00000110B
- Interval time for data transfer — ADTI=#00000000B
- Serial clock $f_X/2^4$ — TCL3=#1000xxxxB
- Set the P22 output latch to the high level.
- P21, P22, P23 set in output mode, P24 in input mode — PM2=#xxx1000xB

<Startup procedure>

When data will be transmitted to the buffer RAM (transmission from a high order address), the number of data bytes to be transmitted is set in DTVAL and TR6451 is called. When the data transfer ends, bit 3 (TRF) of the automatic data transmit/receive control register (ADTC) can be tested for verification.

(2) Use example

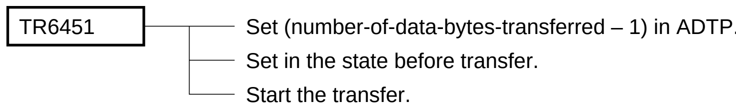
Set data in the buffer RAM.
Set the number of transmission data bytes in DTVAL.
Call TR6451.
WHILE : Waiting for the transfer to end

```
      EXTRN   TR6451, DTVAL

SCK1  EQU     P2.2
          :
          :
      P2=#00000100B
      PM2=#11110001B
      CSIM1=#10100011B             ; Set to the automatic transmit/receive function.
      TCL3=#10001000B              ; SCK1 = 262 kHz
      ADTC=#00000110B              ; The strobe and busy signals are present.
      ADTI=#00000000B
          :
          :
      DE=#TABLE1                   ; Table reference address setting for transmission data
      HL=#0FAC0H                   ; Start address setting of the buffer RAM
      B=32                         ; Number of transmission data bytes setting

      while(B>#0)                  ; Transfer transmission data to the buffer RAM.
              B--
              [HL+B]=[DE] (A)
              DE++
      endw

      DATVAL=#32                   ; Number of transmission data bytes setting
      CALL    !TR6451
      while_bit(TRF)               ; Wait for the transfer to end.
      endw
```

```
TABLE1:
        DB      11111111B           ; Power-on-reset command 1
        DB      01000000B           ; Vertical address 0
        DB      11000000B           ; Horizontal address 0
        DB      10000000B           ; Character size
        DB      11111100B           ; Command 0
        DB      11101001B           ; LC send ON, blinking OFF, display ON

        DB      10001100B           ; Blinking ON, character: RED

        DB      11011011B           ; Color setting, background color: CYAN
        DB      10010101B           ; Display line 5
        DB      10100000B           ; Display digit 0

        DB      07H                 ; 7
        DB      08H                 ; 8
        DB      1BH                 ; K
        DB      6DH                 ; /
        DB      00H                 ; 0
        DB      10H
        DB      11H                 ; A
        DB      20H                 ; P
        DB      20H                 ; P
        DB      1CH                 ; L
        DB      19H                 ; I
        DB      13H                 ; C
        DB      11H                 ; A
        DB      24H                 ; T
        DB      19H                 ; I
        DB      00H                 ; O
        DB      1EH                 ; N
        DB      10H
        DB      1EH                 ; N
        DB      00H                 ; O
        DB      24H                 ; T
        DB      15H                 ; E
```

Remark For information on the commands and data in the output table data, refer to the μ**PD6451A Data Sheet** (Document No. IC-2337A).

(3) SPD chart



(4) Program listing

```
        PUBLIC  TR6451,DTVAL

CSI_DAT DSEG    SADDR
DTVAL:  DS      1                   ; Number of data bytes setting area

CSI_SEG CSEG
;*******************************
;*   µPD6451A communication
;*******************************
TR6451:
        A=DTVAL                     ; Number of data bytes setting
        A--
        ADTP=A
        SIO1=#0FFH                  ; Start the transfer.
        RET
```

8.3 SBI MODE INTERFACE

The 78K/0 Series has the SBI mode which conforms to the NEC serial bus format. The SBI mode allows one master CPU to communicate with multiple slave CPUs via the two wires of clock and data. An example using the μPD78044F subseries is explained here.

Figure 8-15 shows a connection example and Figure 8-16 shows the communication format when using the SBI mode.

Figure 8-15. Connection Example of the SBI Mode

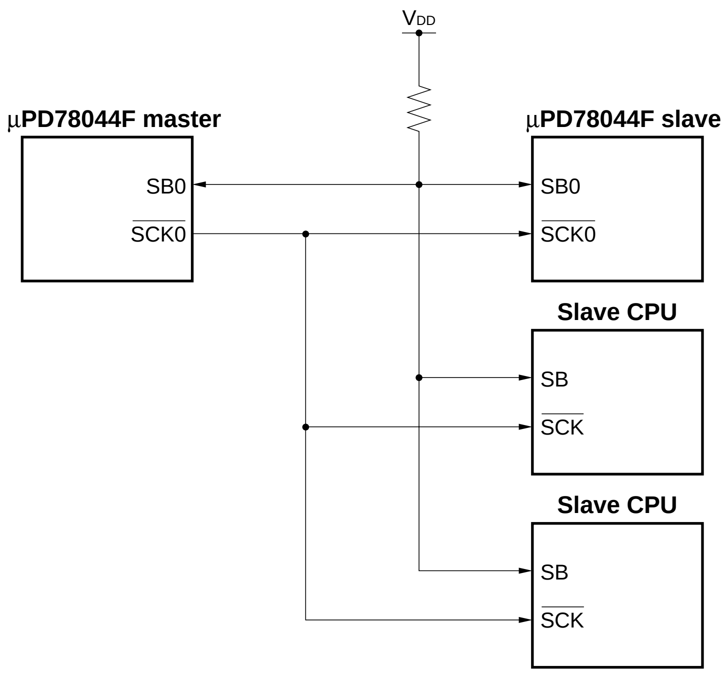
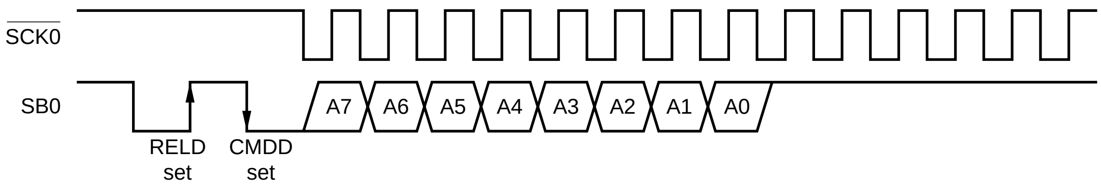
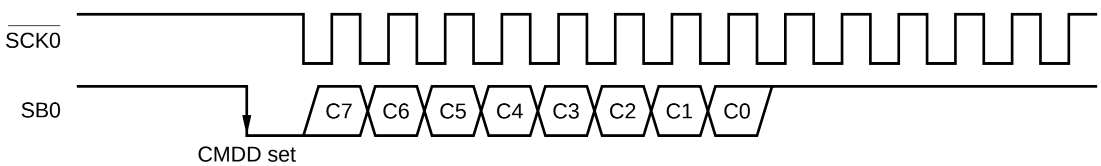


Figure 8-16. SBI Mode Communication Format

(a) Address transmission



(b) Command transmission



(c) Data transmission and reception

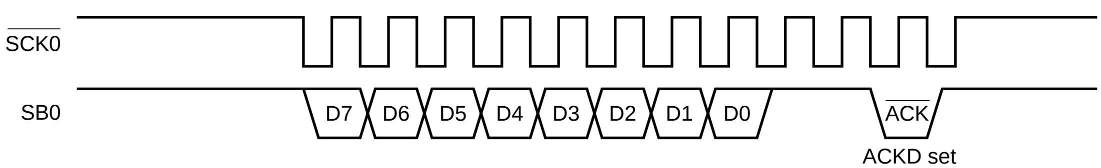


Table 8-5. SBI Mode Signal List

Signal name	Output side	Meaning
Address	Master	Slave device selection
Command	Master	Instruction to a slave device
Data	Master/slave	Data processed by a slave or master
Clock	Master	Transmit/receive synchronization signal for serial data
$\overline{ACK}$	Receiving side[Note]	Reception response signal
$\overline{BUSY}$	Slave	State where communication is not possible

Note During normal operation, the receiving side outputs this signal, but when an error occurs that results in time out processing, the master CPU outputs this signal.

8.3.1 Application as a Master CPU

The processing in (a) to (d) is performed for the slave CPU.

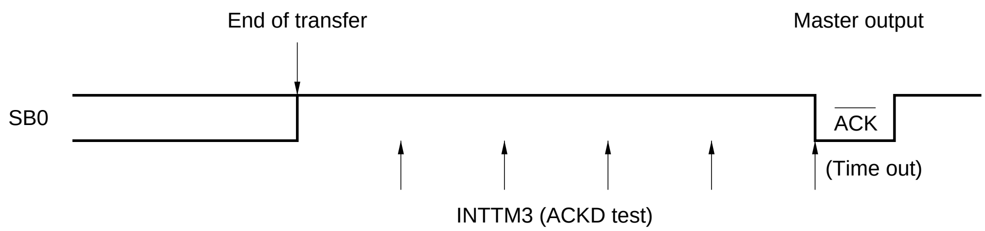
(a) Address transmission
(b) Command transmission
(c) Data transmission
(d) Data reception

Error checks **<1>** and **<2>** are performed in the communication in (a) to (d).

<1> Time out processing

During a master CPU transmission, when the $\overline{\text{ACK}}$ signal is not returned within a constant time (here, within the time it takes for the watch timer to generate five interrupt requests), an error is judged. The master CPU outputs the $\overline{\text{ACK}}$ signal and processing ends.

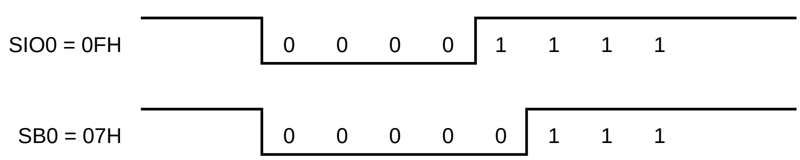
Figure 8-17. Timed Out $\overline{\text{ACK}}$ Signal



<2> Bus line test

The master CPU tests whether the data was correctly output to the bus line by setting the transmission data in serial I/O shift register 0 (SIO0) and slave address register (SVA). Because bus line data is received by SIO0, the normal output of data is verified by testing bit 6 (COI) of serial operating mode register 0 (CSIM0) (set when SIO0 and SVA match) at the end of transfer.

Figure 8-18. Bus Line Test



In Figure 8-18, because the values at the end of transfer do not match (SIO0 = 07H, SVA = 0FH), COI = 0 results and an error is generated in the bus line.

(1) Package description

<Symbols declared as public>

M_TRANS : Name of master SBI transfer subroutine
TR_MODE : Storage area of the selection of the transfer mode
TRNDAT : Transmission data storage area
RCVDAT : Receive data storage area
TRADR : Selection of the address transmission mode
TRCMD : Selection of command transmission mode
TRDAT : Selection of data transmission mode
RCDAT : Selection of data reception mode
ERRORF : Error state test flag

<Registers used>

Subroutine A

<RAM used>

Name	Use	Attributes	Bytes
TR_MODE	Stores the selection of the transfer mode	SADDR	1
ACKCT	Time out counter		
TRNDAT	Stores the transmission data		
RCVDAT	Stores the receive data		

<Flags used>

Name	Use
RCVFLG	Reception mode setting
BUSYFG	Busy state setting
ERRORF	Error state setting
ACKWFG	$\overline{\text{ACK}}$ signal wait state setting

<Nesting>

2 levels, 5 bytes

<Hardware used>

- Serial interface channel 0
- Watch timer

<Initial settings>

• Serial interface channel 0 settings	
SBI mode, SB1 pin selection	CSIM0=#10010011B
• Serial clock $f_X/2^4$	TCL3=#xxxx1000B
• Set SO0 latch high.	RELT = 1
• Set the P27 output latch to the high level.	P27=1
• 1.95-ms interval for the watch timer	TMC2=#00100110B
• Watch timer interrupt enabled	

<Startup procedure>

The data required for the transfer mode is set and M_TRANS is called. After returning from the subroutine, by testing the error flag (ERRORF), the presence of a transfer error can be determined. In addition, during the receiving mode, after returning from the subroutine, the reception data is saved in RCVDAT.

(2) Use example

```
 ├── Transfer mode setting
 ├── Transmission data setting
 ├── Call M_TRANS.
 ├─┬ IF : Error occurs.
 │ └── Error processing
```

```
EXTRN    M_TRANS,TR_MODE,TRADR,TRCMD,TRDAT,RCDAT
EXTRN    TRNDAT,RCVDAT
EXTBIT   ERRORF

SCK0     EQU     P2.7
SB1      EQU     P2.5
      :
      :
SET1     SB1
CSIM0=#10010111B                ; Operate in the SBI mode.
TCL3=#10001000B                 ; SCK0 = 262 kHz
TMC2=#00100110B                 ; Set a 1.95-ms interval for the watch timer.
CLR1     BSYE                   ; Disable the busy signal output.
SET1     RELT                   ; Set the output latch.
SET1     SCK0
CLR1     SB1
CLR1     CSIMK0                 ; Enable serial interface channel 0 interrupt.
CLR1     TMMK3                  ; Enable watch timer interrupt.
EI                              ; Enable master interrupt.
      :
      :
TR_MODE=#TRADR
TRNDAT=#5AH
CALL     !M_TRANS
if_bit(ERRORF)
    Error processing
endif
```

(3) SPD chart

```
M_TRANS
 ◇ CASE : TR_MODE
   OF : TRADR
     WHILE : SB0 = LOW
     WHILE : SCK0 = LOW
     Output command signal.
     Output bus release signal.
   OF : TRCMD
     WHILE : SB0 = LOW
     WHILE : SCK0 = LOW
     Output command signal.
   OF : TRDAT
     Set in the transmission mode.
       Clear RCVFLG.
     BREAK
   OF : RCDAT
     Set in the reception mode.
       Set RCVFLG.
     Set the output off data (FFH) of the bus line.
     BREAK
 Set to the transfer state.
   Set BUSYFG.
 Set transmission data in SIO0 and SVA.
 WHILE : Busy transferring (set BUSYFG)
 Save SIO0 data in RCVDAT.
 ◇ IF : transmission mode
   THEN
   ◇ IF : Error generated in the bus line
     THEN
       Set the error state.
         Set ERRORF.
```

INTCSI0

- Select register bank 0
- IF : Transmission mode
  - THEN
    - IF : No $\overline{ACK}$ signal reception
      - THEN
        - Set the $\overline{ACK}$ waiting state.
          Set ACKWFG
      - ELSE
        - Release the busy state.
          Clear BUSYFG
        - Release the error state.
          Clear ERRORF
  - ELSE
    - Output $\overline{ACK}$ signal
      Clear BUSYFG, ERRORF

INTTM3

- Select register bank 0
- IF : $\overline{ACK}$ waiting state
  - THEN
    - IF : $\overline{ACK}$ signal has been received.
      - THEN
        - Release the $\overline{ACK}$ waiting state.
          Clear ACKWFG
        - Release the busy state.
          Clear BUSYFG
      - ELSE
        - IF : Time out
          - THEN
            - Process the time out error
            - Release the $\overline{ACK}$ waiting state
              Clear ACKWFG
            - Release the busy state
              Clear BUSYFG

(4) Program listing

```
        PUBLIC  M_TRANS,TR_MODE,TRADR,TRCMD,TRDAT,RCDAT
        PUBLIC  TRANDAT,RCVDAT,ERRORF

VECSI0  CSEG    AT 0EH
        DW      INTCSI0       ; Vector address setting of serial interface channel 0
VETM3   CSEG    AT 12H
        DW      INTTM3        ; Vector address setting of the watch timer

SBI_DAT DSEG    SADDR
TRNDAT: DS      1             ; Transmission data
RCVDAT: DS      1             ; Reception data
TR_MODE:DS      1             ; Transfer mode setting
ACKCT:  DS      1             ; ACK time out count

SBI_FLG BSEG
RCVFLG  DBIT                  ; Reception mode setting
BUSYFG  DBIT                  ; Busy transferring state
ERRORF  DBIT                  ; Error display
ACKWFG  DBIT                  ; ACK wait state

SB0     EQU     P2.5
SCK0    EQU     P2.7

TRADR   EQU     1             ; Address transmission mode selection
TRCMD   EQU     2             ; Command transmission mode selection
TRDAT   EQU     3             ; Data transmission mode selection
RCDAT   EQU     4             ; Data reception mode selection
```

```
;************************************
;*   SBI data transfer processing
;************************************
SBI_SEG CSEG
M_TRANS:
    switch(TR_MODE)
    case TRADR:
        SET1    PM2.5
        while_bit(!SB0)                ; SB0 = HIGH?
        CLR1    PM2.5
        endw
        while_bit(!SCK0)               ; SCK = HIGH?
        endw
        SET1    CMDT                   ; Command signal output
        NOP                            ; Wait
        SET1    RELT                   ; Bus release signal output
        A=#TRCMD
    case TRCMD:
        SET1    PM2.5
        while_bit(!SB0)                ; SB0 = HIGH?
        CLR1    PM2.5
        endw
        while_bit(!SCK0)               ; SCK = HIGH?
        endw
        SET1    CMDT                   ; Command signal output
        A=#TRDAT
    case TRDAT:
        CLR1    RCVFLG                 ; Set in the transmission mode.
        A=TRNDAT                       ; Transmission data setting
        break
    case RCDAT:
        SET1    RCVFLG                 ; Set in the reception mode.
        MOV     A,#0FFH                ; Reception buffer off
        break
    ends

        SET1    BUSYFG                 ; Set in the busy transferring state.
        SVA=A                          ; For use in bus line testing
        SIO0=A                         ; Start transfer.

        while_bit(BUSYFG)              ; Busy transferring
        endw
        RCVDAT=SIO0 (A)                ; Reception data storage
        if_bit(!RCVFLG)                ; Reception mode
            if_bit(!COI)               ; Bus line output is no good.
                SET1    ERRORF         ; Set in the error state.
            endif
        endif
        RET
```

```
;**************************************
;*   INTCSI0 interrupt processing
;**************************************
CSI_SEG CSEG
INTCSI0:
        SEL RB0
        if_bit(!RCVFLG)                  ; Transmission mode
            if_bit(!ACKD)                ; No acknowledge signal received
                ACKCT=#5                 ; Setting of the acknowledge signal wait state
                SET1    ACKWFG
            else
                CLR1    BUSYFG           ; Release the busy state.
                CLR1    ERRORF           ; Release the error state.
            endif
        else
            SET1    ACKT                 ; Output the acknowledge signal.
            CLR1    BUSYFG               ; Release the busy state.
            CLR1    ERRORF               ; Release the error state.
        endif
        RETI

;**************************************
;*        Time out processing
;**************************************
TM3_SEG CSEG
INTTM3:
        SEL RB0
        if_bit(ACKWFG)                   ; In the acknowledge signal wait state?
            if_bit(ACKD)                 ; Has the acknowledge signal been received?
                CLR1    ACKWFG           ; Release the acknowledge signal wait state.
                CLR1    BUSYFG           ; Release the busy state.
            else
                ACKCT--
                if(ACKCT==#0)            ; Time out?
                    SET1    ACKT         ; Time out error processing
                    SET1    ERRORF
                    CLR1    ACKWFG       ; Release the acknowledge signal wait state
                    CLR1    BUSYFG       ; Release the busy state.
                endif
            endif
        endif
```

8.3.2 Application as a Slave CPU

Addresses, commands, and data are received from the master CPU and data are transmitted to the master CPU.

In this example, the wakeup function is used and an address is received. A wakeup function is a function that generates an interrupt request signal only when the address transmitted by the master CPU matches the value set in the slave address register (SVA) while in the SBI mode. Consequently, INTCSI0 is generated only in the slave CPU selected by the master CPU. The slave CPUs that are not selected can be operated without generating a spurious interrupt request.

When selected, a slave CPU releases the wakeup function (generates an interrupt request signal at the end of the transfer) and interfaces with the master CPU. In addition, discriminating addresses, commands, and data is done by testing bits 2 and 3 (RELD and CMDD) of the serial bus interface control register (SBIC).

Because there is no automatic return to a state where no slave CPU is selected, a program is required that returns to the unselected state by, for example, command processing between the master and slaves.

(1) Package description

<Symbols declared as public>

RCVDAT: Reception data storage area

<Registers used>

Bank 0: A

<RAM used>

Name	Use	Attributes	Bytes
RCVDAT	Stores the reception data	SADDR	1

<Flags used>

Name	Use
RCVFLG	Reception mode setting

<Nesting>

1 level, 3 bytes

<Hardware used>

- Serial interface channel 0

<Initial settings>

- Serial interface channel 0 settings
  SBI mode, SB1 pin, wakeup mode
  Serial clock is the external clock input　　CSIM0=#10010011B
- Synchronous busy signal output　　BYSE=1
- Set the SO0 latch to the high level.　　RELT=1
- Slave address　　SVA=#SLVADR
- Enable serial interface channel 0 interrupt

<Startup procedure>

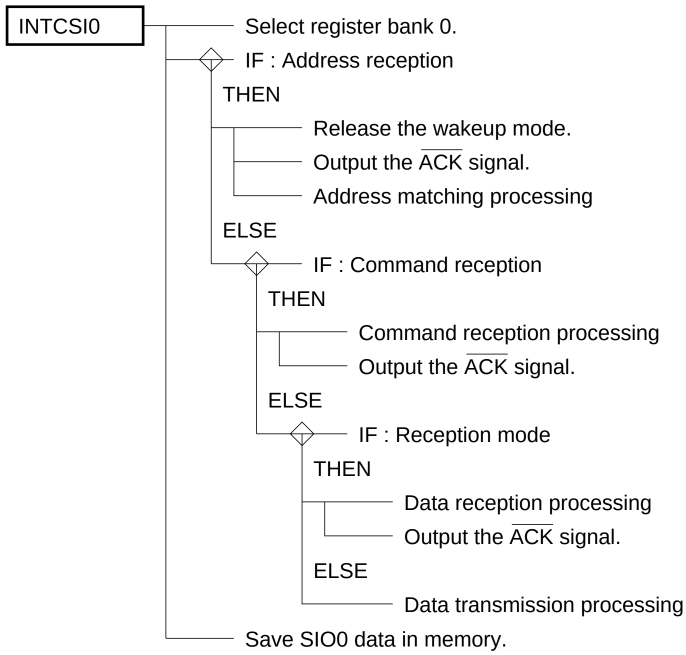
Generating INTCSI0 starts interrupt servicing. The following processing occurs in the interrupt servicing.

- Address, command, and data discrimination
- $\overline{\text{ACK}}$ signal output
- Storing the receive data in RCVDAT.

(2) Use example

```
EXTRN   RCVDAT
EXTBIT  RCVFLG

SLVADR  EQU     5AH
SB1     EQU     P2.5
     :
     :
SET1    SB1
CSIM0=#10110100B            ; Select the external clock input, SB1 pin, wakeup mode
SET1    RELT                ; Set the output latch to the high level.
SET1    BSYE                ; Set in the busy automatic output mode.
SVA=#SLVADR                 ; Slave address setting
SIO0=#0FFH                  ; Start serial transfer instruction
CLR1    SB1
CLR1    CSIMK0              ; Enable the serial interface channel 0 interrupt.
EI                          ; Enable the master interrupt.
```

(3) SPD chart


INTCSI0
Select register bank 0.
IF : Address reception
THEN
Release the wakeup mode.
Output the $\overline{ACK}$ signal.
Address matching processing
ELSE
IF : Command reception
THEN
Command reception processing
Output the $\overline{ACK}$ signal.
ELSE
IF : Reception mode
THEN
Data reception processing
Output the $\overline{ACK}$ signal.
ELSE
Data transmission processing
Save SIO0 data in memory.

(4) Program listing

```
VECSI0  CSEG    AT 0EH
        DW      INTCSI0                 ; Vector address setting of serial interface channel 0
CSI_DAT DSEG    SADDR
RCVDAT: DS      1                       ; Receive data storage area

CSI_FLG BSEG
RCVFLG  DBIT                            ; Reception mode setting

CSI_SEG CSEG
;***********************************
;*   INTCSI0 interrupt processing
;***********************************
INTCSI0:
        SEL RB0
        if_bit(RELD)                    ; Go to address reception.
            CLR1    WUP                 ; Release the wakeup mode.
            SET1    ACKT                ; Output the acknowledge signal.
;   User processing (address reception)

;***********************************

        elseif_bit(CMDD)                ; Go to command reception.
;           User processing (command reception)

            SET1    ACKT                ; Output the acknowledge signal.
        else

            if_bit(RCVFLG)
;               User processing (data reception processing)
                SET1    ACKT            ; Output the acknowledge signal.
            else
;               User processing (data transmission processing)
            endif
;***********************************
        endif
        RCVDAT=SIO0 (A)

        RETI
```

8.4 3-WIRE SERIAL I/O MODE INTERFACE

The function of the 3-wire serial I/O mode (serial clock, data input, data output) of serial channel 0 of the 78K/0 Series is used in communication between the master and slave CPUs. In this example, synchronized master-slave communication is demonstrated by adding one busy signal line as the handshake signal. The busy signal is active-low and is output by the slaves. In addition, data is 8 bits long and the MSB is transmitted first. An example using the μPD78044F subseries is described.

Figure 8-19. Connection Example of the 3-Wire Serial I/O Mode

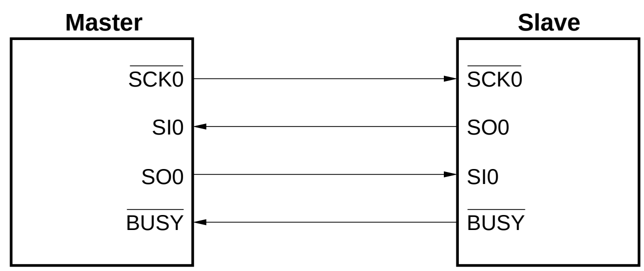
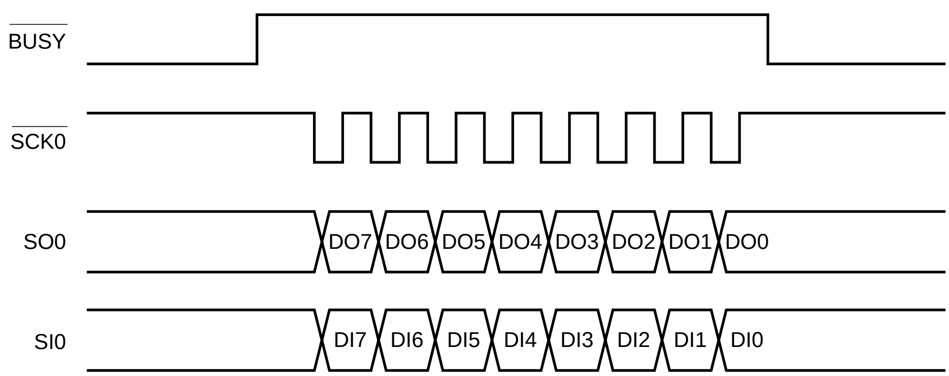


Figure 8-20. Communication Format of the 3-Wire Serial I/O Mode



8.4.1 Application as a Master CPU

The serial clock is set to $f_X/2^4$. Communication with the slave CPU is performed synchronized to this serial clock.

After setting the transmission data, the master CPU begins the transfer. However, when the slave CPU is in the busy state (low busy signal), there is no transfer and the busy flag (BUSYFG) is set.

(1) Package description

<Symbols declared as public>

TRANS : Name of the master 3-wire transfer subroutine
TDATA : Transmission data storage area
RDATA : Receive data storage area
BUSY : Busy signal input port
TREND : End of transfer test flag
BUSYFG : Busy state test flag

<Registers used>

Interrupt bank 0 A
Subroutine A

<RAM used>

Name	Use	Attributes	Bytes
TDATA	Stores the transmission data	SADDR	1
RDATA	Stores the receive data		

<Flags used>

Name	Use
TREND	End of transfer state setting
BUSYFG	Busy state setting

<Nesting>

2 levels, 5 bytes

<Hardware used>

- Serial interface channel 0
- P33

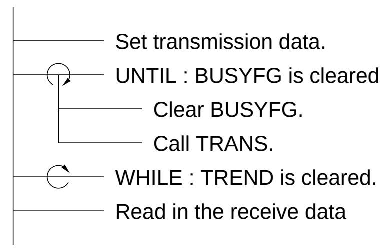
<Initial settings>

- Serial interface channel 0 settings
  3-wire serial I/O mode, MSB first — CSIM0=#10000011B
- Serial clock $f_X/2^4$ — TCL3=#xxxx1000B
- Set the P27 output latch to the high level. — P27=1
- P33 input mode
- Enable the serial interface channel 0 interrupt.

<Startup procedure>

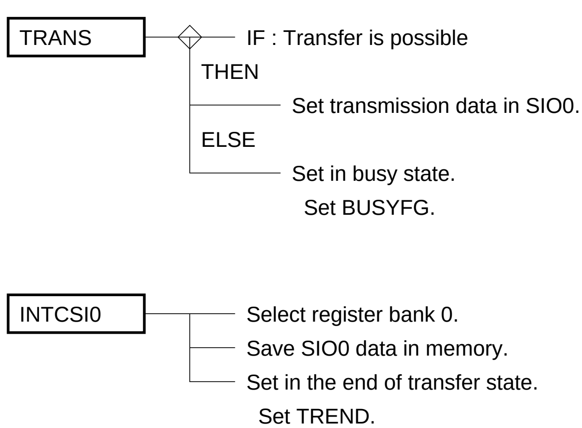
The transmission data is set in TDATA and TRANS is called. After returning from the subroutine, the busy flag (BUSYFG) is tested. When the busy flag is set, the transfer must be repeated because no transfer was performed. In addition, when the busy flag is cleared, receive data is saved in RDATA because the transfer has ended.

(2) Use example




```
EXTRN   TDATA,RDATA,TRANS
EXTBIT  TREND,BUSYFG,BUSY

SCK0    EQU     P2.7
        :
        :
    CSIM0=#10000011B                ; Set to 3-wire serial I/O mode and MSB first.
    TCL3=#10001000B                 ; Set to SCK0 = 262 kHz.
    SET1    SCK0
    SET1    PM3.3                   ; Bit 3 of port 3 set in input mode
    CLR1    CSIMK0                  ; Enable the serial interface channel 0 interrupt.
    EI
        :
        :
    TDATA=A                         ; Transmission data setting
    repeat
            CLR1    BUSYFG          ; Busy test
            CALL    !TRANS
    until_bit(!BUSYFG)
    while_bit(!TREND)               ; End of transfer
    endw
    A=RDATA                         ; Read in the received data.
```

(3) SPD chart



(4) Program listing

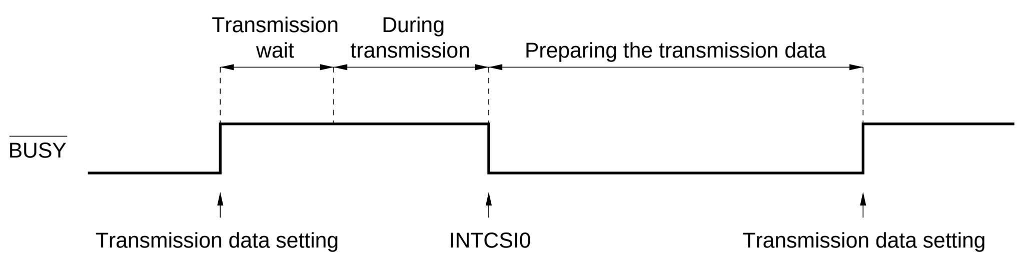
```
        PUBLIC  TRANS,RDATA,TDATA,BUSY,TREND,BUSYFG
VECSI0  CSEG    AT 0EH
        DW      INTCSI0             ; Vector address setting of serial interface channel 0
BUSY    EQU     0FF03H.3            ; 0FF03H = PORT 3

CSI_DAT DSEG    SADDR
RDATA:  DS      1                   ; Receive data storage area
TDATA:  DS      1                   ; Transmission data storage area


CSI_FLG BSEG
TREND   DBIT                        ; End of transfer state setting
BUSYFG  DBIT                        ; Busy state setting

CSI_SEG CSEG

;************************************
;*   INTCSI0 interrupt servicing
;************************************
INTCSI0:
        SEL RB0
        RDATA=SIO0 (A)              ; Save receive data.
        SET1    TREND               ; Set in the end of transfer state.
        RETI

;************************************
;*         3-wire (master)
;************************************
TRANS:
        if_bit(BUSY)                ; Transfer possible state
            SIO0=TDATA (A)          ; Set the transmission data.
        else
            SET1    BUSYFG          ; Set in busy state.
        endif
        RET
```

8.4.2 Application as a Slave CPU

Synchronous transmission/reception of 8-bit data is performed while synchronized to the serial clock from the master CPU. The busy signal from the slave CPU is output at a low level (busy state) while the transmission data is being prepared. The output timing of this busy signal releases the busy signal (high level) by setting the transmission data (CALL !TRANS). A busy signal (low level) is output as a result of interrupt servicing for INTCSI0 at the end of the transfer.

Consequently, the busy state begins at the end of the transfer and lasts until the data is set.

Figure 8-21. Busy Signal Output



(1) Package description

<Symbols declared as public>

TRANS: Name of the slave 3-wire transfer subroutine
TDATA: Transmission data storage area
RDATA: Receive data storage area
BUSY : Busy signal output port
TREND: End of transfer test flag

<Registers used>

Interrupt bank 0 A
Subroutine A

<RAM used>

Name	Use	Attributes	Bytes
TDATA	Store the transmission data.	SADDR	1
RDATA	Store the receive data.		

<Flags used>

Name	Use
TREND	End of transfer state setting

<Nesting>

2 levels, 5 bytes

<Hardware used>

- Serial interface channel 0
- P33

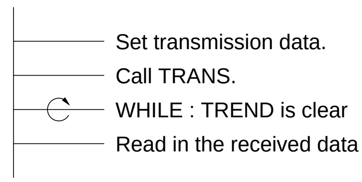
<Initial settings>

- Serial interface channel 0 settings
  3-wire serial I/O mode, MSB first, external clock input   CSIM0=#10000000B
- P33 set in output mode   P33=0
- Busy state setting
- Enable the serial interface channel 0 interrupt.

<Startup procedure>

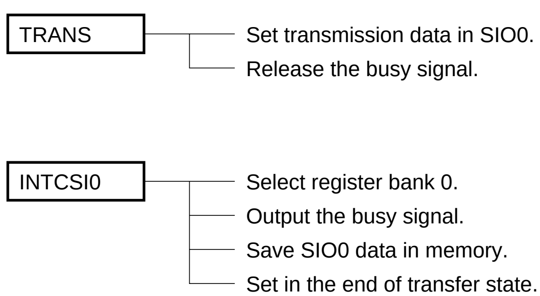
The transmission data is set in TDATA and TRANS is called. Because the busy signal is released in TRANS processing, the state to wait for communication with the master CPU is entered. After communication ends, interrupt service is started by generating INTCSI0. The end of the transfer can be verified by testing TREND. After TREND is set, the received data is saved in RDATA.

(2) Use example




```
EXTRN    TDATA,RDATA,TRANS
EXTBIT   TREND,BUSY
      :
      :
CSIM0=#10000000B             ; Set to the 3-wire serial I/O mode and MSB first.
CLR1    BUSY                 ; Busy state
CLR1    PM3.3                ; Bit 3 of port 3 set in output mode
CLR1    CSIMK0               ; Enable the serial interface channel 0 interrupt.
EI
      :
      :
TDATA=A                      ; Transmission data setting
CALL    !TRANS
while_bit(!TREND)            ; End of transfer
endw
A=RDATA                      ; Read in the receive data
```

(3) SPD chart



(4) Program listing

```
        PUBLIC  RDATA,TDATA,BUSY,TREND,BUSYFG
        PUBLIC  TRANS
VECSI0  CSEG    AT 0EH
        DW      INTCSI0             ; Vector address setting of serial interface channel 0

CSI_DAT DSEG    SADDR
RDATA:  DS      1                   ; Receive data storage area
TDATA:  DS      1                   ; Transmission data storage area

CSI_FLG BSEG
TREND   DBIT                        ; End of transfer state setting
BUSYFG  DBIT                        ; Busy state setting

BUSY    EQU     0FF03H.3            ; 0FF03H=PORT3

CSI_SEG CSEG
;**************************************
;*   INTCSI0 interrupt servicing
;**************************************
INTCSI0:
        SEL RB0
        CLR1    BUSY                ; Set in the busy state.
        RDATA=SIO0 (A)              ; Save the receive data.
        SET1    TREND               ; Set in the end of transfer state.
        RETI

;**************************************
;*          3-wire (slave)
;**************************************
TRANS:
        SIO0=TDATA (A)              ; Transmission data setting
        SET1    BUSY                ; Release the busy state.
        RET
```

8.5 HALF-DUPLEX ASYNCHRONOUS COMMUNICATION

The clocked serial interface channel 0 is used to perform half-duplex asynchronous communication. Two application examples are presented using the 3-wire mode and the SBI mode. The communication protocol is as follows.

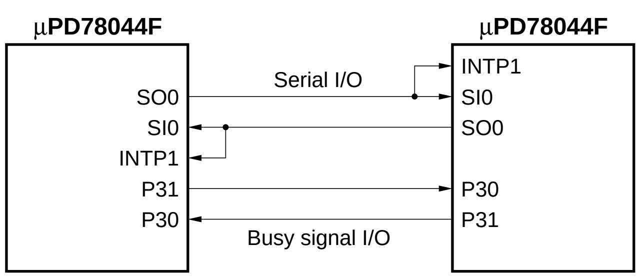
Transmission speed	: 9600 bps
Start bit	: 1 bit
Character length	: 8 bits (LSB first)
Parity bit	: 1 bit (even/odd parity can be selected)
Stop bit	: 2 bits

Because the transmission speed is set to 9600 bps, 8-bit timer/event counter 2 is used to generate the serial clock.

8.5.1 Half-Duplex Asynchronous Communication of the 3-Wire Mode

Figure 8-22 illustrates the system structure. Serial input and output is performed via the SI0 and SO0 pins, respectively. Bits 0 and 1 of port 3 are used as I/O for the BUSY signal. When the BUSY signal is ‘L,’ serial communication is possible.

Figure 8-22. System Structure (3-Wire Mode)



(1) Transmission in the 3-wire mode

Data transmission processing is explained below.

<1> Start bit -> Transmission time wait based on the output latch operation of the serial interface and 8-bit timer/event counter 2

Caution To prevent a timing delay in data reception due to the loss of the start bit, assign high priority to the INTP1 interrupt request.

<2> Data -> Transmission by the serial buffer

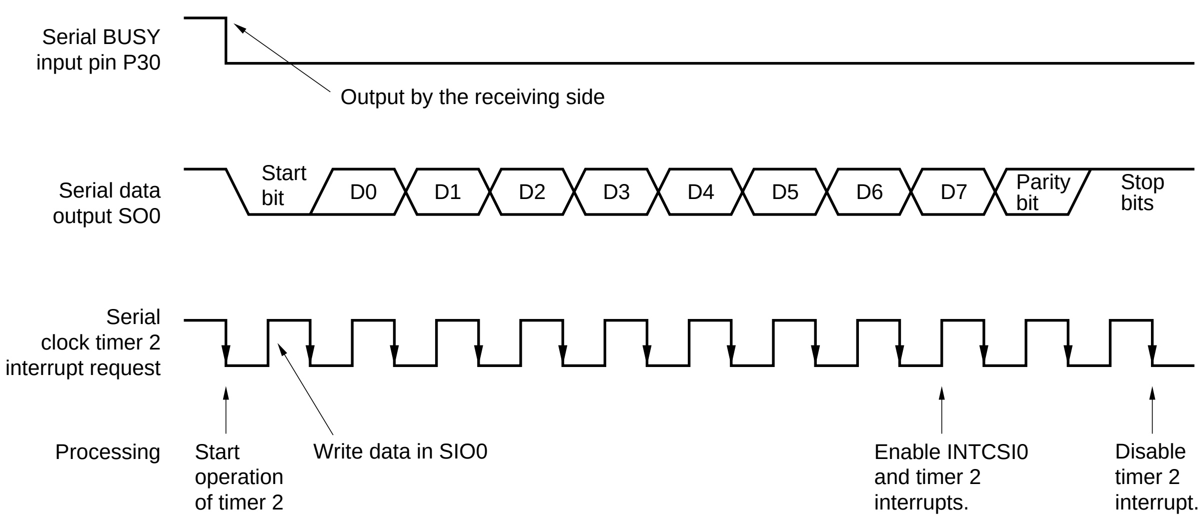
<3> Parity bit -> The output latch of the serial interface is manipulated in the interrupt servicing of 8-bit timer/event counter 2 and the parity bit is output.

Caution To prevent a delay in the transmission timing, assign high priority to interrupt requests from 8-bit timer/event counter 2.

<4> Stop bit -> The output latch of the serial interface in the interrupt servicing in 8-bit timer/event counter 2 is set and the stop bit is output.

Caution To prevent a delay in the transmission timing, assign high priority to interrupt requests from 8-bit timer/event counter 2.

Figure 8-23. 3-Wire Mode Transmission Format



(2) Reception in the 3-wire mode

The following example illustrates data reception processing.

<1> Start bit -> Reception is started by a port test and the detection of a falling edge at the INTP1 pin.

Caution To prevent a timing delay in data reception due to the loss of the start bit, assign high priority to the INTP1 interrupt request.

<2> Data -> Reception by the serial buffer

<3> Parity bit -> The port is tested in the interrupt servicing of 8-bit timer/event counter 2 and the parity bit is output.

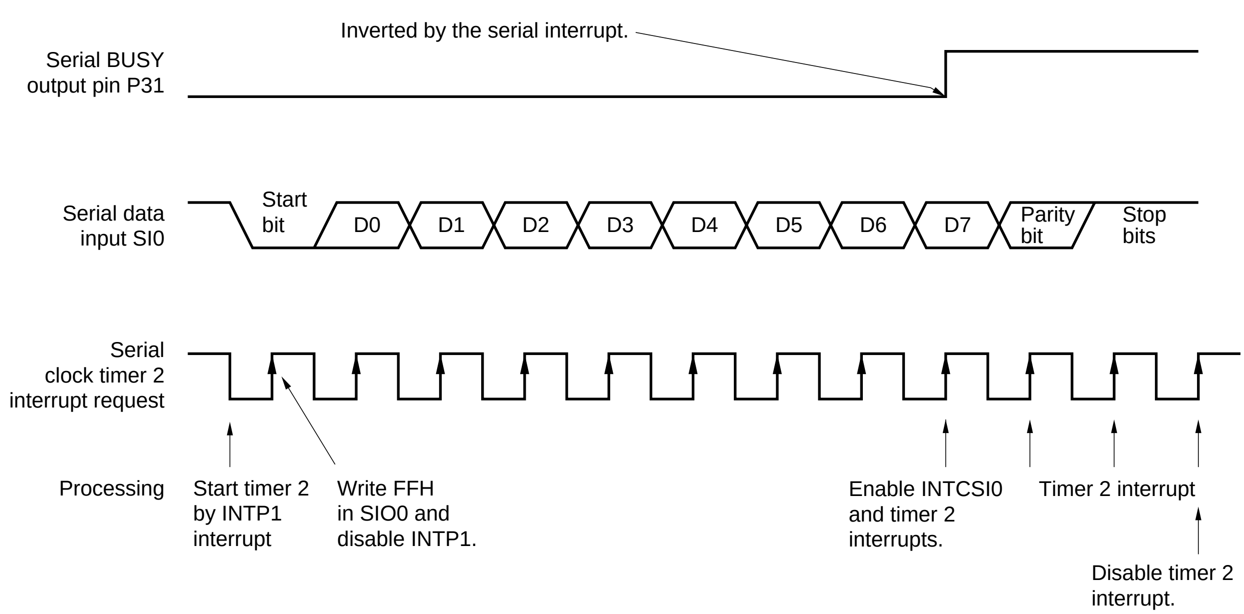
Caution To prevent a delay in reception timing, assign high priority to interrupt requests from 8-bit timer/event counter 2.

<4> Stop bit -> The port is tested in the interrupt servicing for 8-bit timer/event counter 2 and the stop bit is output.

Caution To prevent a delay in reception timing, assign high priority to interrupt requests from 8-bit timer/event counter 2.

When a parity error or an overrun error is generated, the flag is set.

Figure 8-24. 3-Wire Mode Reception Format



(3) Package description

<Symbols declared as public>

- Subroutine names
  S_SOSHIN : Name of transmission subroutine
  S_JUSHIN : Name of reception subroutine
- Input parameters
  SODATA : Stores transmission data
  F_PARITY : Indicates an even or odd parity selection state
  F_TUSHIN : Indicates a receiving or transmitting state
- Output parameters
  JUDATA : Stores the receive data
  F_DATA : This is set after reception ends.
  F_ERRP : Indicates a parity error
  F_ERRE : Indicates an end bit error
- I/O parameter
  F_PADATA : Stores the communication parity bit

<Registers used>

Bank 0 A
Bank 1 A
Bank 2 A

<RAM used>

Name	Use	Attributes	Bytes
SODATA	Transmission data storage area	SADDR	1
JUDATA	Receive data storage area	SADDR	1
C_WORK	State storage counter	SADDR	1
i	Work counter for loop operation	SADDR	1
j	Work counter for loop operation	SADDR	1

<Flags used>

Name		Use
F_PARITY	Parity selection flag	Set when odd parity is selected.
F_PADATA	Parity bit storage flag	Stores the parity.
F_TUSHIN	Communication flag	Set during communication.
F_ERRP	Parity error flag	Set when a parity error occurs.
F_ERRE	End bit error flag	Set when an end bit error occurs.
F_DATA	End of reception flag	Set at the end of reception.
F_WORK	Work flag	For work

<Nesting>

1 level, 3 bytes

<Hardware used>

- Serial interface channel 0 (3-wire mode)
- 8-bit timer/event counter 2
- External interrupt edge detection (INTP1 pin)

<Initial settings>

• Set in the S_SOSHIN and S_JUSHIN subroutines.	
• Port 2: bit 5 input port; bit 6 output port settings	PM2=#x01xxxxxB
• Port 3: bit 0 input port; bit 1 output port settings	PM3=#xxxxxx01B
• Serial interface channel 0 settings	
3-wire mode, serial clock = 8-bit timer/event counter 2 selection	CSIM0=#10000110B
• 8-bit timer/event counter 2 setting	
9600-bps baud rate setting	CR20=#54
8-bit timer register x 2-channel mode	TCL1=#01100000B
8-bit timer/event counter 2 operation disabled	TOC1=#00000000B
	TMC1=#00000000B
• INTP1 setting INTP1 falling edge	INTM0=#00000000B
• High priority 8-bit timer/event counter 2 interrupt	CLR1 TMPR2
• High priority INTP1 interrupt	CLR1 PPR1
• Serial interface interrupt enabled	CLR1 CSIMK0

<Startup procedure>

- Set in the following order when starting data transmission or reception.

  - Starting data transmission
    <1> Store the transmission data in the SODATA area.
    <2> Set the transmission flag.
    <3> Call the S_SOSHIN subroutine.
  - Starting data reception
    <1> Clear the communication flag (F_TUSHIN). (Set to 0.)
    <2> Invert the BUSY signal.
    <3> Call the S_JUSHIN subroutine.

- When interrupt requests other than those in the 78K/0 Series package are used, to enable high priority interrupts, set the ISP flag to 0 at the beginning of interrupt processing and enable interrupts.

(4) Use example

This example illustrates selecting an even or odd parity bit and selecting transmission or reception by using key input.

```
EXTRN   SODATA
EXTRN   JUDATA,S_SOSHIN,S_JUSHIN
EXTBIT  F_PARITY,F_DATA,F_PADATA,F_TUSHIN
EXTBIT  F_ERRE,F_ERRP
;
BUSY_O  EQU P3.1
BUSY_I  EQU P3.0
PARIKEY EQU 22                          ; Decoded parity key value
JYUSHIN EQU 21                          ; Decoded reception key value
TUSHIN  EQU 20                          ; Decoded transmission key value
;**************************************
;              Initialize
;**************************************
VERES   CSEG    AT 00H
    DW  RES_STA
M3          CSEG                        ;
RES_STA:                                ;
    MOV P2,#0BFH                        ; P2.5=H,P2.6=L
    MOV P3,#0FFH                        ;
    MOV PM2,#00100000B                  ; P2.5 = input port, P2.6 = output port
    MOV PM3,#00000001B                  ; P3.0 = input port, P3.1 = output port
;***8-bit timer register settings***
    CR20=#54                            ;
    TCL1=#01100000B                     ; 1.05-MHz count clock
    TOC1=#00000000B                     ;
    TMC1=#00000000B                     ; 8-bit timer register selection, timer 2 operation disabled
;***Serial interface 0 settings***
    CSIM0=#10000110B                    ; 3-wire mode, serial clock selection, 8-bit timer 2
    SET1    RELT                        ;
;***INTP1 settings***
    INTM0=#00000000B                    ; INTP1 falling edge
    CLR1    TMPR2                       ; High priority timer 2 interrupt
    CLR1    PPR1                        ; High priority INTP1 interrupt
    CLR1    PIF1                        ; Clear the INTP1 request flag.
    CLR1    TMIF2                       ; Clear the timer 2 request flag.
    CLR1    CSIIF0                      ; Clear the serial interface request flag.
    CLR1    CSIMK0                      ; Enable the serial interface interrupt.
    while(forever)                      ;
            .                           ;
            .                           ;
```

```
if_bit(F_KEYON)                       ; Is the key on flag 1?
    switch(M_KEYON)                   ;
    case PARIKEY:                     ; The pressed key was the parity key.
        SET1    CY                    ; Invert the even/odd parity decision
        CY ^=F_PARITY                 ;
        F_PARITY=CY                   ;
        break                         ;
    case TUSHIN:                      ; The pressed key was the communication key.
        SET1    F_TUSHIN              ; Set the communication flag (during transmission).
        CLR1    F_SOEND               ;
        break                         ;
    case JYUSHIN:                     ; The pressed key was the reception key.
        CLR1    F_TUSHIN              ; Clear the communication flag (during reception)
        CY=BUSY_0                     ; Inverted BUSY signal data is output.
        NOT1    CY                    ;
        BUSY_0=CY                     ;
        if_bit(CY)                    ;
            SET1    PMK1              ; INTP1 interrupt is disabled.
        else                          ;
            CLR1    F_ERRP            ;
            CLR1    F_ERRE            ;
            CALL    !S_JUSHIN         ;
        endif                         ;
        break                         ;
    ends                              ;
endif                                 ;
      .
      .
      .
if_bit(!F_SOEND)                      ;
    if_bit(F_TUSHIN)                  ; Is the communication flag set?
        CY=BUSY_I                     ; Is the BUSY signal inactive?
        if_bit(!CY)                   ;
            SODATA=#0                 ;
            SET1    F_SOEND           ;
            SODATA=WORK               ; Transmission data storage area <- transmission data
            CALL    !S_SOSHIN         ;
        endif                         ;
    endif                             ;
endif
```

(5) SPD chart

[Reception subroutine]

S_JUSHIN
- Clear INTP1 request flag.
- Enable INTP1 interrupt.

[Transmission subroutine]

S_SOSHIN
- if (odd parity is selected)
  THEN
  - Set the parity data flag.
- for (i = #0 ; i < #8 ; i+ +)
  - CY <- Least significant bit of the transmission data
  - The exclusive-OR is taken of CY and the parity data flag.
  - Transfer the result to the parity data flag.
- The timer output flip-flop of 8-bit timer/event counter 2 is reset and the inversion operation is enabled.
- Clear request flag of 8-bit timer/event counter 2.
- Disable interrupts.
- Enable 8-bit timer/event counter 2 operation.
- Transmit the start bit.
- Wait the time it takes to transmit the start bit.
- SIO0 <- transmission data
- Enable interrupts.

[Parity end bit communication processing (8-bit timer/event counter 2 interrupt)]

TAIMA2

- Switch to bank 1.
- if (transmitting data)
  - THEN
    - switch (What data is being transmitted?)
      - [case : 1]
        - Parity data transmission
      - [case : 2]
        - First end bit transmission
      - [case : 3]
        - Second end bit transmission
        - Disable 8-bit timer/event counter 2 interrupt.
        - Disable operation of 8-bit timer/event counter 2.
    - switch (What data is being received?)
      - [case : 1]
        - Parity data read
      - [case : 2]
        - if (Is first end bit error present?)
          - THEN
            - Set end bit error flag.
      - [case : 3]
        - if (Is second end bit error present?)
          - THEN
            - Set end bit error flag.
        - Disable 8-bit timer/event counter 2 interrupt.
        - Disable operation of 8-bit timer/event counter 2.
        - if (Does parity data match?)
          - THEN
            - if (Is end bit present?)
              - THEN
                - Set end of reception flag.
          - ELSE
            - Set parity error flag.

[Data transmission/reception completion processing]

- INTSI0
  - Switch to bank 2.
  - Clear 8-bit timer/event counter 2 request flag.
  - Enable 8-bit timer/event counter 2 interrupt.
  - Output 'H' to the BUSY0 signal.
  - if (receiving)
    THEN
    - Read in receive data.

[Startup processing for data reception (INTP1 interrupt processing)]

- INTP1
  - Switch to bank 1.
  - Clear the 8-bit timer/event counter 2 request flag.
  - Clear the 8-bit timer 2 counter.
  - Enable the operation of 8-bit timer/event counter 2.
  - Wait the time to read the start bit.
  - if (Is the INTP1 pin asserted?)
    THEN
    - Disable INTP1 interrupt.
    - Read preparation for receive data

(6) Program listing

```
PUBLIC   F_PADATA,F_PARITY
PUBLIC   F_DATA,F_TUSHIN
PUBLIC   JUDATA,SODATA,S_JUSHIN,S_SOSHIN
PUBLIC   F_ERRP,F_ERRE
;
VEINTP1      CSEG    AT 08H
             DW      INTP              ; INTP1 vector address setting
VEINTSI0     CSEG    AT 0EH
             DW      INTSI0            ; Vector address setting of serial interface channel 0
VETIM2       CSEG    AT 18H
             DW      TAIMA2            ; Vector address setting of 8-bit timer 2
;
SI0          EQU     P2.5
BUSY_0       EQU     P3.1
BUSY_1       EQU     P3.0
;
MORAM        DSEG    SADDR
SODATA:      DS      1                 ; Transmission data storage area
C_WORK:      DS      1                 ; Work counter
JUDATA:      DS      1                 ; Received data storage area
i:           DS      1                 ; Work counter
k:           DS      1                 ; Work counter
;
MOFLG        BSEG
F_PARITY     DBIT                      ; Parity selection flag
F_ERRP       DBIT                      ; Parity error flag
F_ERRE       DBIT                      ; End bit error flag
F_DATA       DBIT                      ; End of reception flag
F_PADATA     DBIT                      ; Parity data flag
F_WORK       DBIT                      ; Work flag
F_TUSHIN     DBIT                      ; Communication flag
;**********************************
;         Reception routine
;**********************************
JUSHIN   CSEG                          ;
S_JUSHIN:                              ;
    CLR1    PIF1                       ; Clear INTP1 request flag.
    CLR1    PMK1                       ; Enable the INTP1 interrupt.
    RET                                ;
```

```
;**********************************
;       Transmission routine
;**********************************
SOSHIN  CSEG
S_SOSHIN:                           ;
    CLR1    F_PADATA                ; Clear parity data.
    if_bit(F_PARITY)                ; Is odd parity selected?
        SET1    F_PADATA            ; Set parity data.
    endif                           ;
    A=SODATA                        ;
    for(i=#0;i<#8;i++)              ; Determine parity data.
        RORC    A,1                 ;
        CY ^=F_PADATA               ;
        F_PADATA = CY               ;
    next                            ;
    TOC1=#01100000B (A)             ;
    CLR1    TMIF2                   ; Clear timer 2 request flag.
    DI                              ;
    SET1    TCE2                    ; Enable 8-bit timer operation.
    SET1    CMDT                    ; Transmit the start bit.
    while_bit(!TMIF2)               ; Wait for the start bit to be transmitted.
    endw                            ;
    CLR1    TMIF2                   ;
    SIO0=SODATA (A)                 ; Start data transmission.
    EI                              ;
    RET                             ;
;**********************************
;   Timer 2 interrupt servicing
;**********************************
TIM2        CSEG                    ;
TAIMA2:                             ;
    SEL RB1                         ; Set to bank 1.
    if_bit(F_TUSHIN)                ; Is the communication flag set?
        if(C_WORK <= #4)            ; Work counter contents
            switch(C_WORK)          ; 0: parity data transmission
            case 0:                 ;
                if_bit(F_PADATA)    ;
                    SET1    RELT    ;
                else                ;
                    SET1    CMDT    ;
                endif               ;
                break               ;
            case 2:                 ; 2: end bit transmission
                SET1    RELT        ; Transmit 'H.'
                break               ;
            case 4:                 ; 4: end bit transmission
                SET1    RELT        ; Transmit 'H.'
                SET1    TMMK2       ; Disable timer 2 interrupt.
                CLR1    TCE2        ; Disable 8-bit timer operation.
                C_WORK=#0           ;
                break               ;
            ends                    ;
            C_WORK++                ;
        else                        ;
            C_WORK=#0               ;
        endif
```

```
else                                ;
    if(C_WORK <= #6)                ; Receiving?
        switch(C_WORK)              ; Work counter contents
        case 1:                     ; 1: read in parity data
            CY=SI0                  ;
            F_PADATA=CY             ;
            break                   ;
        case 3:                     ; 3: check the end bit
            if_bit(!SI0)            ; If an error is present, set the end bit error flag.
                SET1    F_ERRE      ;
            endif                   ;
            break                   ;
        case 5:                     ; 5: check the end bit
            if_bit(!SI0)            ; If an error is present, set the end bit error flag.
                SET1    F_ERRE      ;
            endif                   ;
            C_WORK=#0               ;
            SET1    TMMK2           ; Disable timer 2 interrupt.
            CLR1    TCE2            ; Disable 8-bit timer operation.
            CLR1    F_WORK          ;
            if_bit(F_PARITY)        ;
                SET1    F_WORK      ;
            endif                   ;
            A=JUDATA                ;
            for(i=#0;i<#8;i++)      ; Store in the receive data.
                RORC  A,1           ;
                CY ^= F_WORK        ;
                F_WORK = CY         ;
            next                    ;
            CLR1    F_ERRP          ;
            CLR1    F_DATA          ;
            F_WORK ^= F_PADATA (CY) ;
            if_bit(!F_WORK)         ; Check parity data.
               if_bit(!F_ERRE)      ; Check end bit data.
                  SET1    F_DATA    ;
               endif                ;
            else                    ; If parity data matches, set F_DATA.
                SET1    F_ERRP      ; If the parity data does not match, set the parity error flag.
            endif                   ;
            break                   ;
        ends                        ;
        C_WORK++                    ;
    else                            ;
        C_WORK=#0                   ;
    endif                           ;
endif                               ;
RETI                                ;
```

```
;*********************************************
;    INTSI0 interrupt servicing (reception)
;*********************************************
S_SI0   CSEG                                ;
INTSI0:                                     ;
    SEL RB2                                 ; Set to bank 2.
    CLR1    TMIF2                           ; Clear timer 2 request flag.
    CLR1    TMMK2                           ; Enable timer 2 interrupt.
    SET1    BUSY_0                          ; Output high BUSY signal.
    if_bit(!F_TUSHIN)                       ;
        JUDATA=SIO0 (A)                     ;
    endif                                   ;
    C_WORK=#0                               ; Clear the work counter to zero.
    RETI                                    ;
;*********************************************
;    INTP1 interrupt servicing (reception)
;*********************************************
S_P1    CSEG                                ;
INTP1:                                      ;
    SEL RB1                                 ;
    CLR1    TMIF2                           ; Clear timer 2 request flag.
    CLR1    TCE2                            ; Clear timer 2 counter.
    SET1    TCE2                            ; Enable timer operation.
    while_bit(!TMIF2)                       ;
    endw                                    ;
    CLR1    TMIF2                           ;
    if_bit(!SI0)                            ; INTP1 chattering processing
        TOC1=#10100000B                     ;
        SET1    PMK1                        ; Disable INTP1 interrupt.
        SIO0=#0FFH                          ;
    endif                                   ;
    RETI                                    ;
    END
```

8.5.2 Half-Duplex Asynchronous Communication in the SBI Mode

Figure 8-25 shows the system structure. Serial input and output are performed via pin SB0. Bits 0 and 1 of port 3 are used for input and output of the BUSY signal. When the BUSY signal is low, serial communication is possible.

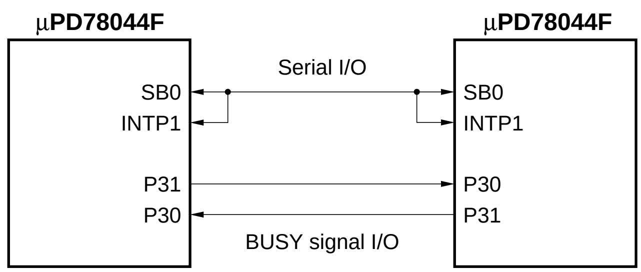
Cautions concerning the use of the SBI mode are given below.

<1> Set bit 5 of port 2 (SB0) in the output mode when reset starts. However, when the SB0 port is tested, set SB0 in the input mode. At the end of port testing, set in the output mode again.

<2> After the last stop bit is transmitted and detected in serial communication, enable serial operation again after it has been disabled.

Essentially, the end of SBI communication is determined by checking the ready signal after detecting the acknowledge signal. However, because the acknowledge signal is used in transmitting and receiving the parity bit, when a ‘1’ parity bit is transmitted and received, the condition for the end of SBI communication does not hold. When this is not considered to be the end of serial communication, sometimes the next communication does not operate normally.

Figure 8-25. System Structure (SBI Mode)



(1) Transmission in the SBI mode

Data transmission processing is shown below.

<1> Start bit -> Transmission time wait based on the output latch operation of the serial interface and 8-bit timer/event counter 2

Caution To prevent a timing delay in data reception due to the loss of the start bit, assign high priority to the INTP1 interrupt request.

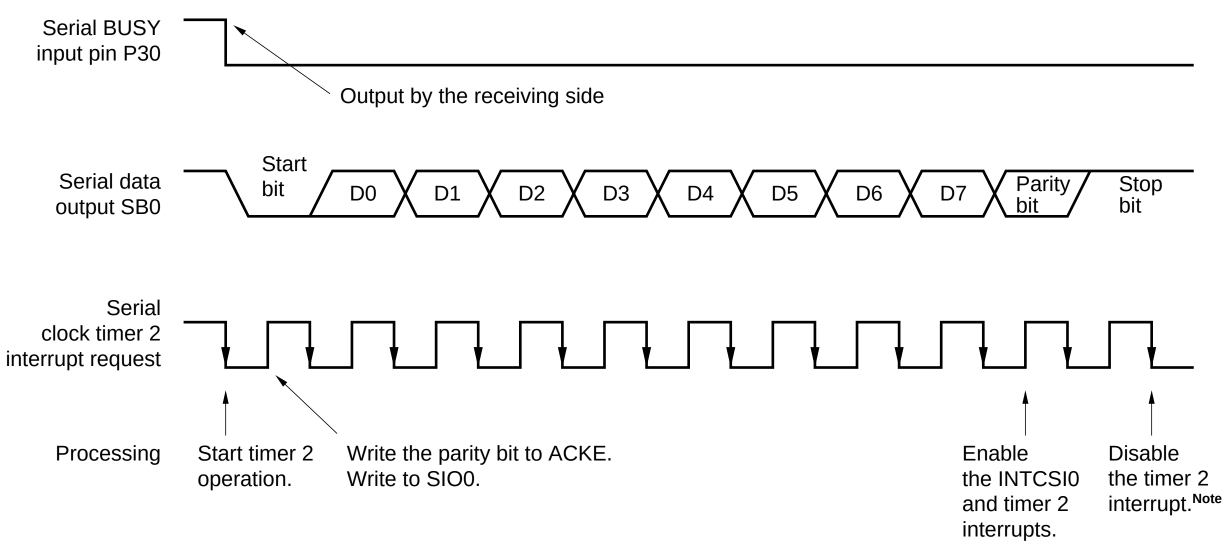
<2> Data and parity bits -> 9-bit transmission by the serial buffer and the acknowledge signal

<3> Stop bit -> The output latch of the serial interface is set in the interrupt servicing of 8-bit timer/event counter 2 and the stop bit is output.

Cautions 1. To prevent delays in the transmission timing, assign high priority to an interrupt request from 8-bit timer/event counter 2.

2. If the second stop bit has been transmitted, enable operation again after serial operation for verifying the end of transmission is disabled once.

Figure 8-26. SBI Mode Transmission Format



Note After serial operation is disabled once, set again to enable.

(2) Reception in the SBI mode

Data reception processing is shown below.

<1> Start bit -> Start reception by detecting a falling edge at pin INTP1 and testing the port

Cautions 1. When testing the port, set in the following order.
<1> Set bit 5 (SI0) of port 2 to an input port.
<2> Test the port and write to SIO0.
<3> Reset bit 5 of port 2 in the output mode.
2. To prevent delayed timing in data reception due to the loss of the start bit, assign high priority to the INTP1 interrupt request.

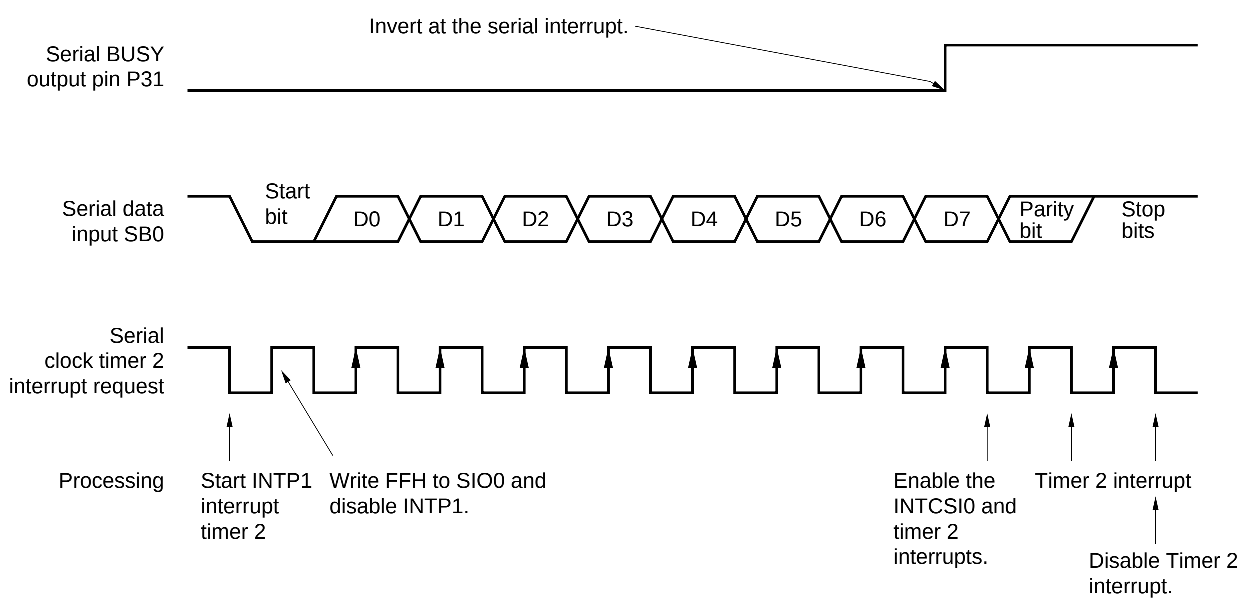
<2> Data and parity bits -> Reception by the serial buffer and acknowledge detection

<3> Stop bit -> Test the port in interrupt servicing for 8-bit timer/event counter 2 and output the parity bit.

Cautions 1. To prevent delays in the transmission timing, assign high priority to an interrupt request from 8-bit timer/event counter 2.
2. If the second stop bit has been transmitted, enable operation again after serial operation for verifying the end of transmission is disabled once.

When a parity or an overrun error occurs, set the flag.

Figure 8-27. SBI Mode Reception Format



(3) Package description

<Symbols declared as public>

- Subroutine names
  S_SOSHIN : Name of transmission subroutine
  S_JUSHIN : Name of reception subroutine
- Input parameters
  SODATA : Stores transmission data
  F_PARITY : Indicates even and odd parity selection state
  F_TUSHIN : Indicates the busy receiving or transmitting state
- Output parameters
  JUDATA : Stores receive data
  F_DATA : If reception is over, this is set.
  F_ERRP : Indicates a parity error
  F_ERRE : Indicates an end bit error
- I/O parameter
  F_PADATA : Stores the parity bit for communication

<Registers used>

Bank 0 A
Bank 1 A
Bank 2 A

<RAM used>

Name	Use	Attributes	Bytes
SODATA	Transmission data storage area	SADDR	1
JUDATA	Receive data storage area	SADDR	1
C_WORK	State storage counter	SADDR	1
i	Work counter for loop operation	SADDR	1
j	Work counter for loop operation	SADDR	1

<Flags used>

Name		Use
F_PARITY	Parity selection flag	Set when odd parity is selected.
F_PADATA	Parity bit storage flag	Stores the parity.
F_TUSHIN	Communication flag	Set during communication.
F_ERRP	Parity error flag	Set when a parity error occurs.
F_ERRE	End bit error flag	Set when an end bit error occurs.
F_DATA	End of reception flag	Set at the end of reception.
F_WORK	Work flag	For work

<Nesting>

1 level, 3 bytes

<Hardware used>

- Serial interface channel 0 (SBI mode)
- 8-bit timer/event counter 2
- External interrupt edge detection (INTP1 pin)

<Initial settings>

- After a reset start at the pin (P25) for I/O data, set the following before the serial transmission of the first byte.
  <1> Set the output latch of P25 to 1.
  <2> Set bit 0 (RELT) of the serial bus control register (SBIC) to 1.
  <3> This time, set the output latch of the P25 set to 1 to 0.
- Set in the S_SOSHIN and S_JUSHIN subroutines.
- Port 2: bit 5 input port, bit 6 output port settings   PM2=#x01xxxxxB
- Port 3: bit 0 input port, bit 1 output port settings   PM3=#xxxxxx01B
- Serial interface channel 0 setting
  SBI mode, serial clock = 8-bit timer 2 selection   CSIM0=#10010110B
- 8-bit timer/event counter 2 settings
  9600-bps baud rate setting   CR20=#54
  8-bit timer register x 2-channel mode   TCL1=#01100000B
  8-bit timer/event counter 2 operation disabled   TOC1=#00000000B
  TMC1=#00000000B
- INTP1 setting INTP1 falling edge   INTM0=#00000000B
- High-priority 8-bit timer/event counter 2 interrupt   CLR1 TMPR2
- High-priority INTP1 interrupt   CLR1 PPR1
- Enable serial interface interrupt   CLR1 CSIMK0

<Startup procedure>

- Set the following order when starting data transmission and reception.

  - Starting data transmission
    <1> Store transmission data in the SODATA area.
    <2> Set transmission flag.
    <3> Call the S_SOSHIN subroutine.
  - Starting data reception
    <1> Clear the communication flag (F_TUSHIN). (Set to 0.)
    <2> Invert the busy signal.
    <3> Call the S_JUSHIN subroutine.

- When interrupt requests other than those in the 78K/0 Series package are used, to enable high priority interrupts, set the ISP flag to 0 at the beginning of interrupt processing and enable interrupts.

(4) Use example

This example illustrates selecting an even or odd parity bit and selecting transmission or reception by using key input.

```
EXTRN    SODATA
EXTRN    JUDATA,S_SOSHIN,S_JUSHIN
EXTBIT   F_PADATA,F_PARITY,F_DATA,F_TUSHIN
EXTBIT   F_ERRP,F_ERRE
;
TUSHIN   EQU 20
JYUSHIN  EQU 21
PARIKEY  EQU 22
BUSY_O   EQU P3.1
BUSY_I   EQU P3.0
SB0      EQU P2.5
;***************************************
;            Initialize
;***************************************
M3S          CSEG                       ;
RES_STA:                                ;
    MOV P2,#9FH                         ; P2.5=L, P2.6=L
    MOV P3,#0FFH                        ;
    MOV PM2,#00000000B                  ; P2.5 = output mode
    MOV PM3,#00000001B                  ; P3.0 = input port, P3.1 = output port
;***8-bit timer register setting***
    CR20=#54                            ;
    TCL1=#01100000B                     ; 1.05-MHz count clock
    TOC1=#00000000B                     ;
    TMC1=#00000000B                     ; 8-bit timer register selection and timer 2 operation disable
;***Serial interface 0 settings***
    SET1    SB0                         ;
    CSIM0=#10000110B                    ; SBI mode, serial clock selection, 8-bit timer 2
    SET1    RELT                        ;
    CLR1    SB0                         ;
;***INTP1 settings***
    CLR1    TMPR2                       ; High priority timer 2 interrupt
    CLR1    PPR1                        ; High priority INTP1 interrupt
    INTM0=#00000000B                    ; INTP1 falling edge
    CLR1    PIF1                        ; Clear the INTP1 request flag.
    CLR1    TMIF2                       ; Clear the timer 2 request flag.
    CLR1    CSIIF0                      ; Clear the serial interface request flag.
    CLR1    KSIF                        ; Clear the interrupt request flag.
    CLR1    CSIMK0                      ; Enable serial interface interrupt.
    CLR1    KSMK                        ; Enable INTKS interrupt.
```

```
while(forever)                      ;
      .                             ;
      .                             ;
   if_bit(F_KEYON)                  ; Is the key on flag 1?
       switch(M_KEYON)              ;
       case PARIKEY:                ; The pressed key was the parity key.
           SET1    CY               ; Invert odd or even parity decision
           CY ^= F_PARITY           ;
           F_PARITY=CY              ;
           break                    ;
       case TUSHIN:                 ; The pressed key was the communication key.
           SET1    F_TUSHIN         ; Set the communication flag (while transmitting).
           CLR1    F_SOEND          ;
           break                    ;
       case JYUSHIN:                ; The pressed key was the reception key.
           CLR1    F_TUSHIN         ; Clear the communication flag (while receiving).
           CY=BUSY_0                ; Output the inverted BUSY signal data.
           NOT1    CY               ;
           BUSY_0=CY                ;
           it_bit(CY)               ;
               SET1    PMK1         ; Disable INTP1 interrupt.
           else                     ;
               CLR1    F_ERRP       ;
               CLR1    F_ERRE       ;
               CALL    !S_JUSHIN    ;
           endif                    ;
           break                    ;
       ends                         ;
   endif
         .
         .
   if_bit(!F_SOEND)                 ; Is the communication flag set?
       if_bit(F_TUSHIN)             ; Is the BUSY signal inactive?
           CY=BUSY_I                ;
           if_bit(!CY)              ;
               SET1    F_SOEND      ;
               SODATA=#0            ;
               SODATA=WORK (A)      ; Transmission data storage area <- transmission data
               CALL    !S_SOSHIN    ; Call the transmission routine.
           endif                    ;
       endif                        ;
   endif                            ;
```

(5) SPD chart

[Reception subroutine]

S_JUSHIN
- Clear INTP1 request flag.
- Enable INTP1 interrupt.

[Transmission subroutine]

S_SOSHIN
- Reverse the direction of the transmission data.
- if (odd parity is selected)
  THEN
  - Set parity data flag.
- for (i = #0 ; i < #8 ; i++)
  - CY <- Least significant bit of the transmission data
  - The exclusive-OR is taken of CY and the parity data flag.
  - Transfer the result to the parity data flag.
- The timer output flip-flop of 8-bit timer/event counter 2 is reset and the inverse operation is enabled.
- Clear request flag of 8-bit timer 2.
- Disable interrupts.
- Enable 8-bit timer 2 operation.
- Transmit start bit.
- Wait the time the start bit is transmitted.
- ACKE <- $\overline{\text{parity bit data}}$
- SIO0 <- transmission data
- Enable interrupts.

[Stop bit transmission/reception processing (8-bit timer/event counter 2 interrupt servicing)]

```
TAIMA2
  Switch to bank 1.
  if (data is being transmitted)
  THEN
    switch (What data is being transmitted?)
      [case : 1]
        First end bit transmission
      [case : 2]
        Second end bit transmission
        Disable 8-bit timer counter 2 interrupt.
        Disable operation of 8-bit timer 2.
    switch (What data is being received?)
      [case : 1]
        if (Is the first end bit error present?)
        THEN
          Set end bit error flag.
      [case : 2]
        if (Is the second end bit error present?)
        THEN
          Set end bit error flag.
        Disable 8-bit timer counter 2 interrupt.
        Disable operation of 8-bit timer 2.
        if (Does the parity data match?)
        THEN
          Set end of reception flag.
        ELSE
          Set parity error flag.
```

[Data transmission/reception completion processing (INTSI0 interrupt servicing)]

- INTSI0
  - Switch to bank 2.
  - Clear 8-bit timer 2 request flag.
  - Enable 8-bit timer 2 interrupt.
  - Output the high level to the BUSY0 signal.
  - if (receiving)
    THEN
    - Read in received data.
    - Reverse the received data that was read in.
    - Read in parity data.

[Starting processing for data reception (INTP1 interrupt servicing)]

- INTP1
  - Switch to bank 1.
  - Clear 8-bit timer 2 counter.
  - Enable operation of 8-bit timer 2.
  - Wait the time for the start bit to be read in.
  - if (Is the INTP1 pin asserted?)
    THEN
    - Clear the 8-bit timer 2 request flag.
    - Disable INTP1 interrupt.
    - Received data read preparation

(6) Program listing

```
PUBLIC   JUDATA
PUBLIC   SODATA,F_PARITY,S_SOSHIN
PUBLIC   F_DATA,S_JUSHIN,F_PADATA,F_TUSHIN
PUBLIC   F_ERRE,F_ERRP
;
VEINTP1      CSEG    AT 08H
             DW      INTP1
VEINTSI0     CSEG    AT 0EH
             DW      INTSI0
VETIM2       CSEG    AT 18H
             DW      TAIMA2
;
SB0          EQU     P2.5
BUSY_O       EQU     P3.1
BUSY_I       EQU     P3.0
PORT25       EQU     PM2.5
;
MOSRAM       DSEG    SADDR
SODATA:      DS      1              ; Transmission data storage area
C_WORK:      DS      1              ; Work counter
JUDATA:      DS      1              ; Receive data storage area
i:           DS      1              ; Work counter
k:           DS      1              ; Work counter
;
MOSFLG       BSEG
F_ERRP       DBIT                   ; Parity error flag
F_ERRE       DBIT                   ; End bit error flag
F_DATA       DBIT                   ; End of reception flag
F_PADATA     DBIT                   ; Parity data flag
F_PARITY     DBIT                   ; Parity selection flag
F_WORK       DBIT                   ; Flag work area
F_TUSHIN     DBIT                   ; Communication flag
;
;*********************************
;        Reception routine
;*********************************
JUSHIN   CSEG                       ;
S_JUSHIN:                           ;
    CLR1    PIF1                    ; Clear the request flag.
    CLR1    PMK1                    ; Enable INTP1 interrupt.
    RET                             ;
```

```
;
;*************************************
;    Transmission routine
;*************************************
SOSHIN  CSEG
S_SOSHIN:                             ;
    A=SODATA                          ; Reverse the direction of the transmission data.
    SODATA=#0                         ;
    if_bit(A.7)                       ;
        SET1    SODATA.0              ;
    endif                             ;
    if_bit(A.6)                       ;
        SET1    SODATA.1              ;
    endif                             ;
    if_bit(A.5)                       ;
        SET1    SODATA.2              ;
    endif                             ;
    if_bit(A.4)                       ;
        SET1    SODATA.3              ;
    endif                             ;
    if_bit(A.3)                       ;
        SET1    SODATA.4              ;
    endif                             ;
    if_bit(A.2)                       ;
        SET1    SODATA.5              ;
    endif                             ;
    if_bit(A.1)                       ;
        SET1    SODATA.6              ;
    endif                             ;
    if_bit(A.0)                       ;
        SET1    SODATA.7              ;
    endif                             ;
    CLR1    F_PADATA                  ; Clear the parity data flag.
    if_bit(F_PARITY)                  ; Is odd parity currently selected?
        SET1    F_PADATA              ; Set the parity data.
    endif                             ;
    A=SODATA                          ;
    for(k=#0;k<#8;k++)                ; Parity data setting.
        RORC    A,1                   ;
        CY ^= F_PADATA                ;
        F_PADATA = CY                 ;
    next                              ;
    TOC1=#01100000B (A)               ;
    CLR     TMIF2                     ; Clear the timer 2 request flag.
    DI                                ;
    SET1    TCE2                      ; Enable 8-bit timer operation.
    SET1    CMDT                      ; Start bit transmission.
    while_bit(!TMIF2)                 ; Wait the time for the start bit to be transmitted.
    endw                              ;
    CLR1    TMIF2                     ;
    SET1    ACKE                      ; Clear acknowledge.
    if_bit(F_PADATA)                  ; Clear acknowledge when parity data is 1.
        CLR1    ACKE                  ;
    endif                             ;
    SIO0=SODATA (A)                   ; Start data transmission
    EI                                ;
    RET                               ;
```

```
;
;*********************************
;   Timer 2 interrupt servicing
;*********************************
TIM2    CSEG                        ;
TAIMA2:                             ;
    SEL RB1                         ; Set to bank 1.
    if_bit(F_TUSHIN)                ; Busy communicating?
        if(C_WORK < #3)             ; Work mode contents
            switch(C_WORK)          ; 0: end bit transmission
            case 0:                 ;
                SET1    RELT        ;
                break               ;
            case 2:                 ; 2: end bit transmission
                SET1    RELT        ; Disable 8-bit timer 2 interrupt.
                SET1    TMMK2       ;
                CLR1    TCE2        ; Disable 8-bit timer 2 operation.
                SET1    SB0         ; Set bit 5 of port 2 to an input port.
                CLR1    CSIE0       ; Disable serial operation.
                SET1    CSIE0       ; Enable serial operation.
                SET1    RELT        ;
                CLR1    SB0         ; Set bit 5 of port 2 to an output mode.
                C_WORK=#0           ;
                break               ;
            ends                    ;
            C_WORK++                ;
        else                        ;
            C_WORK=#0               ;
        endif                       ;
```

```
else                                        ;
    if(C_WORK < #4)                         ; Busy receiving?
        SET1    PORT25                      ; Set bit 5 of port 2 to an input port.
        switch(C_WORK)                      ; Work mode contents
        case 1:                             ; 1: If the end bit is high, set the end bit error flag.
            if_bit(!SB0)                    ;
                SET1    F_ERRE              ;
            endif                           ;
            break                           ;
        case 3:                             ; 3: If the end bit is high, set the end bit error flag.
            if_bit(!SB0)                    ;
                SET1    F_ERRE              ;
            endif                           ;
            SET1    SB0                     ; Bit 5 of port 2 = High
            CLR1    CSIE0                   ; Disable serial operation.
            SET1    CSIE0                   ; Enable serial operation.
            SET1    RELT                    ;
            CLR1    SB0                     ; Bit 5 of port 2 = Low
            C_WORK=#0                       ;
            SET1    TMMK2                   ; Disable 8-bit timer 2 interrupt.
            CLR1    TCE2                    ; Disable 8-bit timer operation.
            CLR1    F_WORK                  ;
            if_bit(F_PARITY)                ;
                SET1    F_WORK              ;
            endif                           ;
            A=JUDATA                        ;
            for(i=#0;i<#8;i++)              ; Store in the receive data.
                RORC   A,1                  ;
                CY ^= F_WORK                ;
                F_WORK = CY                 ;
            next                            ;
            CLR1    F_ERRP                  ;
            CLR1    F_DATA                  ;
            F_WORK ^= F_PADATA (CY);
            if_bit(!F_WORK)                 ; Check parity data.
                if_bit(!F_ERRE)             ;
                    SET1     F_DATA         ; If a normal reception, set the F_DATA flag.
                endif                       ;
            else                            ;
                SET1    F_ERRP              ; If a parity error occurs, set the F_ERRP flag.
            endif                           ;
            CLR1    F_WORK                  ;
            break                           ;
        ends                                ;
        CLR1    PORT25                      ; Set bit 5 of port 2 to an output port.
        C_WORK++                            ;
    else                                    ;
        C_WORK=#0                           ;
    endif                                   ;
endif                                       ;
RETI
```

```
;
;*********************************************
;  INTSI0 interrupt servicing (reception)
;*********************************************
S_SI0       CSEG                         ;
INTSI0:                                  ;
    SEL RB2                              ;
    CLR     TMIF2                        ; Clear timer 2 request flag.
    CLR1    TMMK2                        ; Enable timer 2 interrupt.
    SET1    BUSY_O                       ;
    if_bit(!F_TUSHIN)                    ;
        A=SIO0                           ;
        JUDATA=#0                        ;
        if_bit(A.7)                      ; Reread the receive data in reverse.
            SET1    JUDATA.0             ;
        endif                            ;
        if_bit(A.6)                      ;
            SET1    JUDATA.1             ;
        endif                            ;
        if_bit(A.5)                      ;
            SET1    JUDATA.2             ;
        endif                            ;
        if_bit(A.4)                      ;
            SET1    JUDATA.3             ;
        endif                            ;
        if_bit(A.3)                      ;
            SET1    JUDATA.4             ;
        endif                            ;
        if_bit(A.2)                      ;
            SET1    JUDATA.5             ;
        endif                            ;
        if_bit(A.1)                      ;
            SET1    JUDATA.6             ;
        endif                            ;
        if_bit(A.0)                      ;
            SET1    JUDATA.7             ;
        endif                            ;
        CLR1    F_PADATA                 ; Read in the parity data.
        CY=ACKD                          ;
        NOT1    CY                       ;
        F_PADATA=CY                      ;
    endif                                ;
    C_WORK=#0                            ;
    RETI                                 ;
```

```
;********************************************
;    INTP1 interrupt servicing (reception)
;********************************************
S_P1            CSEG                    ;
INTP1:                                  ;
    SEL RB1                             ;
    CLR1    TMIF2                       ; Clear timer 2 request flag.
    CLR1    TCE2                        ; Clear timer 2 counter.
    SET1    TCE2                        ; Enable timer operation.
    while_bit(!TMIF2)                   ;
    endw                                ;
    CLR1    TMIF2                       ;
    SET1    PORT25                      ; Set port to an input port.
    if_bit(!SB0)                        ; Chattering processing of INTP1
        CLR1    ACKE                    ;
        TOC1=#10100000B                 ;
        SET1    PMK1                    ; Disable INTP1 interrupt.
        SIO0=#0FFH                      ;
    endif                               ;
    CLR1    PORT25                      ; Set bit 5 of port 2 to an output port.
    RETI                                ;
    END
```

[MEMO]

CHAPTER 9 A/D CONVERTER APPLICATION

The A/D converter in the 78K/0 Series has 8-bit resolution and eight channels, and is a successive approximation type. Its only operating mode is the select mode, but the start of conversion can also be specified by using an external trigger. In addition, when there is no external trigger, the selected channel is repeated and A/D conversion is performed.

The A/D converter is set by the A/D converter mode registers (ADM and ADM0), A/D converter input selection register (ADIS), and analog input channel specification register (ADS0).

* **Cautions 1. ADM0 and ADS0 are incorporated into the μPD780228 subseries only.**

* **2. The format of the registers incorporated into the μPD780228 subseries differs from that of the μPD78044F, μPD78044H, and μPD780208 subseries. When using any of the sample programs described in this chapter with the μPD780228 subseries, replace the register settings with those for the μPD780228 subseries.**

Figure 9-1. Format of the A/D Converter Mode Register (μPD78044F, μPD78044H, and μPD780208 Subseries)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
ADM	CS	TRG	FR1	FR0	ADM3	ADM2	ADM1	1	FF80H	01H	R/W

ADM3	ADM2	ADM1	Analog input channel selection
0	0	0	ANI0
0	0	1	ANI1
0	1	0	ANI2
0	1	1	ANI3
1	0	0	ANI4
1	0	1	ANI5
1	1	0	ANI6
1	1	1	ANI7

FR1	FR0	A/D conversion time selection[Note 1]	
		With f_X = 5.0 MHz	With f_X = 4.19 MHz
0	0	160/f_X (32.0 μs)	160/f_X (38.1 μs)
0	1	80/f_X (Setting prohibited[Note 2])	80/f_X (19.1 μs)
1	0	200/f_X (40.0 μs)	200/f_X (47.7 μs)
1	1	Setting prohibited	

TRG	External trigger selection
0	No external trigger (software start mode)
1	Conversion started by an external trigger (hardware start mode)

CS	A/D converter operation control
0	Stop operation
1	Start operation

Notes 1. Set the A/D conversion time to at least 19.1 μs.

2. Setting is prohibited because the A/D conversion time is less than 19.1 μs.

Cautions 1. Set bit 0 to 1.

2. When executing the HALT or STOP instruction, clear bit 7 (CS) of the ADM register to stop A/D conversion operations prior to the instruction execution. This reduces the total power consumption of the device in the standby mode, because the A/D converter consumes much power when operating.

3. To restart A/D conversion operation, clear the interrupt request flag (ADIF) to 0.

Remark f_X: Main system clock oscillation frequency

*

Figure 9-2. Format of the A/D Converter Mode Register (μPD780228 Subseries)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
ADM0	CS0	0	FR02	FR01	FR00	0	0	0	FF80H	00H	R/W

FR02	FR01	FR00	A/D conversion time selection[Note 1]	
			With f_X = 5.0 MHz	With f_X = 4.19 MHz
0	0	0	144/f_X (28.8 μs)	144/f_X (34.4 μs)
0	0	1	120/f_X (24 μs)	120/f_X (28.6 μs)
0	1	0	96/f_X (19.2 μs)	96/f_X (22.9 μs)
1	0	0	72/f_X (14.4 μs)	72/f_X (17.2 μs)
1	0	1	60/f_X (Setting prohibited[Note 2])	60/f_X (14.3 μs)
1	1	0	48/f_X (Setting prohibited[Note 2])	48/f_X (Setting prohibited[Note 2])
Other than the above			Setting prohibited	

CS0	A/D converter operation control
0	Stop converter operation
1	Enable converter operation

Notes 1. Set the A/D conversion time to at least 14 μs.

2. Setting is prohibited because the A/D conversion time is less than 14 μs.

Caution The results of conversion obtained immediately after setting bit 7 (CS0) to 1 will be unpredictable.

Remark f_X: Oscillation frequency of the main system clock

Figure 9-3. Format of the A/D Converter Input Selection Register (μPD78044F, μPD78044H, and μPD780208 Subseries)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
ADIS	0	0	0	0	ADIS3	ADIS2	ADIS1	ADIS0	FF84H	00H	R/W

ADIS3	ADIS2	ADIS1	ADIS0	Selection of the number of analog input channels
0	0	0	0	No analog input channels (P10-P17)
0	0	0	1	1 channel (ANI0, P11-P17)
0	0	1	0	2 channels (ANI0, ANI1, P12-P17)
0	0	1	1	3 channels (ANI0-ANI2, P13-P17)
0	1	0	0	4 channels (ANI0-ANI3, P14-P17)
0	1	0	1	5 channels (ANI0-ANI4, P15-P17)
0	1	1	0	6 channels (ANI0-ANI5, P16, P17)
0	1	1	1	7 channels (ANI0-ANI6, P17)
1	0	0	0	8 channels (ANI0-ANI7)
Other than the above				Setting prohibited

Cautions 1. Set the analog input channel in the following order.

<1> Set the number of analog input channels in ADIS.

<2> For channels set for analog input in ADIS, the channel for A/D conversion selects one channel in the A/D converter mode register (ADM).

2. Regardless of the value of bit 1 (PUO1) in the pull-up resistor option register (PUO), the channel selected for analog input in ADIS does not use the on-chip pull-up resistor.

★ **Figure 9-4. Format of the Analog Input Channel Specification Register (Only for the μPD780228 Subseries)**

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
ADS0	0	0	0	0	ADS03	ADS02	ADS01	ADS00	FF81H	00H	R/W

ADS03	ADS02	ADS01	ADS00	Analog input channel selection
0	0	0	0	ANI0
0	0	0	1	ANI1
0	0	1	0	ANI2
0	0	1	1	ANI3
0	1	0	0	ANI4
0	1	0	1	ANI5
0	1	1	0	ANI6
0	1	1	1	ANI7
Other than the above				Setting prohibited

9.1 LEVEL METER

The analog voltage input to the A/D converter is displayed by 16 LEDs. The LED display is arranged in a 4 x 4 matrix. An example using the μPD78044F subseries is described here.

Because the objective in this example is a level meter, this LED display digitally shows the current decibel level of the analog ANIn pin input. Figure 9-5 shows the level meter circuit. Figure 9-6 shows the relationship between the A/D conversion result and the number of display digits.

Figure 9-5. Level Meter Circuit Example

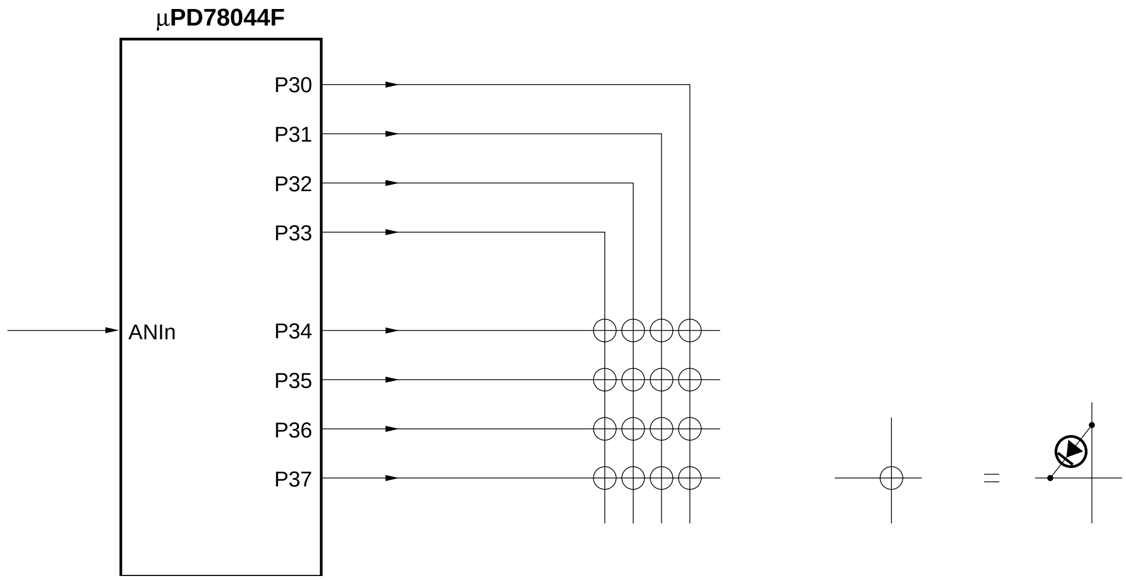
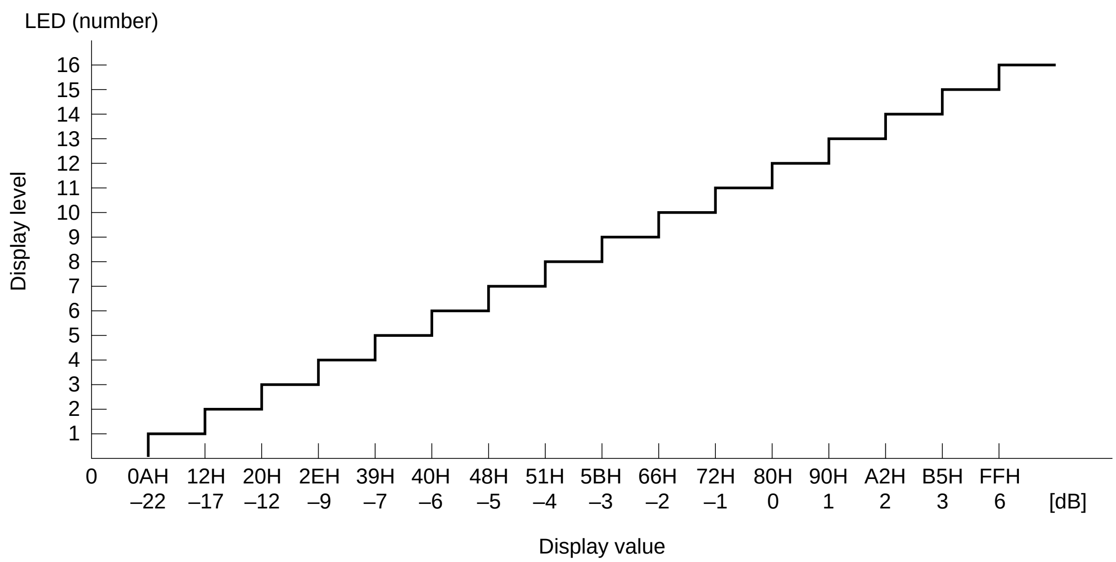


Figure 9-6. A/D Conversion Result and LED Display



The level meter in this example operates in the manner described in **<1>** to **<3>**.

<1> Measurement method

A/D conversion is performed every 20 ms. The data of the last four conversions are averaged and used in the LED display data.

<2> Display method

The LED is updated every 20 ms. The LED display is a 4 x 4 = 16 dynamic display. 8-bit timer/ event counter 1 (interval time: 2 ms) is used in the dynamic display.

<3> Peak hold

The maximum display level hold during a constant period (1 second) is called the peak hold. Even when the display level drops during the constant period, only the maximum display level of the LEDs is held. As a result, the hold period of the hold level ranges from 20 ms to 1 s.

Figure 9-7. Conceptual Diagram of the Peak Hold

	Constant period (1 s)																
Hold level	6	6	6	6	7	8	9	9	9	9	9	9	4	4	4	5	6	6
Display level	6	5	4	5	7	8	9	8	7	6	5	5	4	3	3	5	6	2

(1) Package description

<Symbols declared as public>

LEVEL : Name of LED display subroutine
DSPLEV : Display level storage area
HLDLEV : Hold level storage area
CT20MS : 20-ms measurement counter
CT1S : 1-s measurement counter

<Registers used>

AX, HL, BC (subroutine servicing)
Bank 0: A, HL, B (interrupt servicing)

<RAM used>

Name	Use	Attributes	Bytes
ADDAT	A/D conversion value storage	SADDR	4
DSPLEV	Display level storage		1
HLDLEV	Hold level storage		
CT20MS	20-ms measurement counter		
CT1S	1-s measurement counter		
DIGCNT	Display digit counter		
DSPDAT	Display data storage		4
WORKCT	Work counter for loop operation		1

<Flags used>

Name	Use
T20MSF	Set every 20 ms.
T1SF	Set every 1 s.

<Nesting>

2 levels, 5 bytes

<Hardware used>

- A/D converter
- 8-bit timer/event counter 1
- P3

<Initial settings>

- Channel selection and operation start of the A/D converter ADM=#1000xxx1B
- 2-ms interval for the 8-bit timer/event counter 1 TCL1=#10101010B
TMC1=#00000001B
CR10=130
- P3 output mode
- Set the P3 output latch to the low level.
- INTTM1 interrupt enabled

<Startup procedure>

This program is divided into the two parts of A/D conversion processing (subroutine) and LED display processing (interrupts).

- A/D conversion processing
  Call LEVEL at least once every 20 ms from the main processing. In LEVEL processing, A/D conversion is performed when 20 ms have elapsed.
- LED display
  The 4 x 4 matrix LED display performs a dynamic display by using interrupt servicing by 8-bit timer/event counter 1 (interval: 2 ms). In addition, in interrupt servicing by 8-bit timer/event counter 1, the flags of T20MSF (read in A/D conversion value) and T1SF (end of the hold period) used in A/D conversion are set using the interval (2 ms).

(2) Use example

```
EXTRN   LEVEL,CT20MS,CT1S

MOV     CT20MS,#10
MOV     CT1S,#50
MOV     TMC2,#00100110B
CLR1    TMMK3

P3=#00H                     ; Turn off LED display
PM3=#00000000B
ADM=#10000001B              ; ANI0 pin, start operation
TCL1=#10101010B             ; Set 8-bit timer/event counter 1 to 2 ms.

CR10=#130
TMC1=#00000001B
CLR1    TMMK1               ; Enable 8-bit timer/event counter 1 interrupt.
EI
```

(3) SPD chart

```
LEVEL
  IF : 20 ms has elapsed (T20MSF = 1)
  THEN
    Clear T20MSF
    Store the A/D conversion value in memory
    Average the A/D conversion values of the last four conversions
    (FOR : WORKCT = #0 ; WORKCT < #16 ; WORKCT++)
      IF : conversion result > comparison data for the display level
      THEN
        Update the comparison data
      ELSE
        BREAK
    Save the display data in memory
    IF : less than 1 second (T1SF = 0)
    THEN
      IF : hold level < display level
      THEN
        Set the display level to the hold level
    ELSE
      Clear T1SF
      Set the display level in the hold level
    Convert the display level and hold level into the segment signal
    Combine the digit signal and segment signal and store in memory

INTTM1
  Select register bank 0
  Output the OFF signal to digit and segment
  Output the memory contents indicating the digit counter
  Increment the digit counter
  Decrement the 20-ms measurement counter
  IF : 20 ms has elapsed (CT20MS = 0)
  THEN
    Set the 20-ms counter to 10
    Set to the 20-ms elapsed state
    Set T20MSF
    Decrement the 1-s measurement counter
    IF : 1 second has elapsed (CT1S = 0)
    THEN
      Set 50 in the 1-s counter
      Set to the 1-s elapsed state
      Set T1SF
```

(4) Program listing

```
        PUBLIC  LEVEL,HLDLEV,DSPLEV,CT20MS,CT1S

AD_DAT  DSEG    SADDR
ADDAT:  DS      4                       ; A/D conversion result storage area
DSPLEV: DS      1                       ; Display level value
HLDLEV: DS      1                       ; Hold level value
CT20MS: DS      1                       ; 20-ms measurement counter
CT1S:   DS      1                       ; 1-s measurement counter
DIGCNT: DS      1                       ; Display digit counter
DSPDAT: DS      4                       ; Display data
WORKCT: DS      1

AD_FLG  BSEG
T20MSF  DBIT                            ; 20-ms measurement
T1SF    DBIT                            ; 1-s measurement

VETM1   CSEG    AT 16H
        DW      INTTM1                  ; Set vector address of 8-bit timer/event counter 1

AD_SEG  CSEG
;**********************************
;*   Level meter data setting
;**********************************
LEVEL:
        IF_BIT(T20MSF)                  ; 20-ms check
            CLR1    T20MSF
            A=ADCR                      ; A/D conversion input
            A<->ADDAT                   ; Save A/D conversion value.
            A<->ADDAT+1
            A<->ADDAT+2
            A<->ADDAT+3
                                        ; Average the last four A/D conversion values.
            AX=#0H
            HL=#ADDAT                   ; Data storage address
            for(WORKCT=#0;WORKCT<#4;WORKCT++)
                A+=[HL]
                HL++
                if_bit(CY)              ; Carry
                    X++                 ; High-order digit
                endif
            next

            A<->X
            C=#4                        ; Average four conversions
            AX/=C                       ; AX/C=AX (quotient)...C (remainder)
            if(C>=#2) (A)               ; Remainder processing (carry ≥ 2)
                X++                     ; Carry processing
            endif

            HL=#LEVTBL
            B=#0                        ; Conversion result storage register
            for(WORKCT=#0;WORKCT<#16;WORKCT++)
                if(X>=[HL+B]) (A)       ; Data comparison
                    B++
                else
                    break
                endif
            next
```

```
        DSPLEV=B (A)                    ; Display data decision

        if_bit(!T1SF)                   ; 1 s (hold update)
            X=HLDLEV (A)                ; Comparison of hold and display levels
            if(X<DSPLEV) (A)
                HLDLEV=DSPLEV (A)
            endif
        else
            CLR1    T1SF
            HLDLEV=DSPLEV (A)
        endif

        HL=#DSPTBL
        A=DSPLEV                        ; Create display level.
        A+=A
        B=A
        A=HLDLEV
        A+=A
        C=A
        X=[HL+B] (A)
        B++
        A=[HL+B]
        HL=#HLDTBL                      ; Create hold level.
        A<->X
        A|=[HL+C]
        A<->X
        C++
        A|=[HL+C]
        BC=AX

        HL=#DSPDAT                      ; First digit segment signal setting
        A=C
        A&=#0FH
        A|=#00010000B                   ; Digit signal setting
        [HL]=A
        HL++
        A=C                             ; Second digit segment signal setting
        A>>=1
        A>>=1
        A>>=1
        A>>=1
        A&=#0FH
        A|=#00100000B                   ; Digit signal setting
        [HL]=A
        HL++
        A=B                             ; Third digit segment signal setting
        A&=#0FH
        A|=#01000000B                   ; Digit signal setting
        [HL]=A
        HL++
        A=B                             ; Fourth digit segment signal setting
        A>>=1
        A>>=1
        A>>=1
        A>>=1
        A&=#0FH
        A|=#10000000B                   ; Digit signal setting
        [HL]=A
    endif
```

```
        RET

LEVTBL:
        DB    0AH
        DB    12H
        DB    20H
        DB    2EH
        DB    39H
        DB    40H
        DB    48H
        DB    51H
        DB    5BH
        DB    66H
        DB    72H
        DB    80H
        DB    90H
        DB    0A2H
        DB    0B5H
        DB    0FFH

DSPTBL:
        DW    0000000000000000B
        DW    0000000000000001B
        DW    0000000000000011B
        DW    0000000000000111B
        DW    0000000000001111B
        DW    0000000000011111B
        DW    0000000000111111B
        DW    0000000001111111B
        DW    0000000011111111B
        DW    0000000111111111B
        DW    0000001111111111B
        DW    0000011111111111B
        DW    0000111111111111B
        DW    0001111111111111B
        DW    0011111111111111B
        DW    0111111111111111B
        DW    1111111111111111B

HLDTBL:
        DW    0000000000000000B
        DW    0000000000000001B
        DW    0000000000000010B
        DW    0000000000000100B
        DW    0000000000001000B
        DW    0000000000010000B
        DW    0000000000100000B
        DW    0000000001000000B
        DW    0000000010000000B
        DW    0000000100000000B
        DW    0000001000000000B
        DW    0000010000000000B
        DW    0000100000000000B
        DW    0001000000000000B
        DW    0010000000000000B
        DW    0100000000000000B
        DW    1000000000000000B
$EJECT
```

```
;***********************************
;*      Level meter output
;***********************************
TM1_SEG CSEG
INTTM1:
        SEL RB0
        P3=#00000000B           ; Turn off digit and segment signals.
        HL=#DSPDAT
        B=DIGCNT (A)
        P3=[HL+B] (A)
        DIGCNT++
        DIGCNT&=#00000011B
        CT20MS--                ; 20 ms ?
        if(CT20MS==#0)
            CT20MS=#10          ; Initial counter setting
            SET1    T20MSF
            CT1S--              ; 1 s ?
            if(CT1S==#0)
                CT1S=#50        ; Initial counter setting
                SET1    T1SF
            endif
        endif
        RETI
```

9.2 THERMOMETER

In this example, a thermistor (6 kΩ/0 °C) is used in a temperature sensor and measures temperatures between –20 °C and +50 °C. Changes in resistance corresponding to the temperature of the thermistor can be represented in the following way.

$$R = R_0 \exp \{B\ (1/T - 1/T_0)\}$$

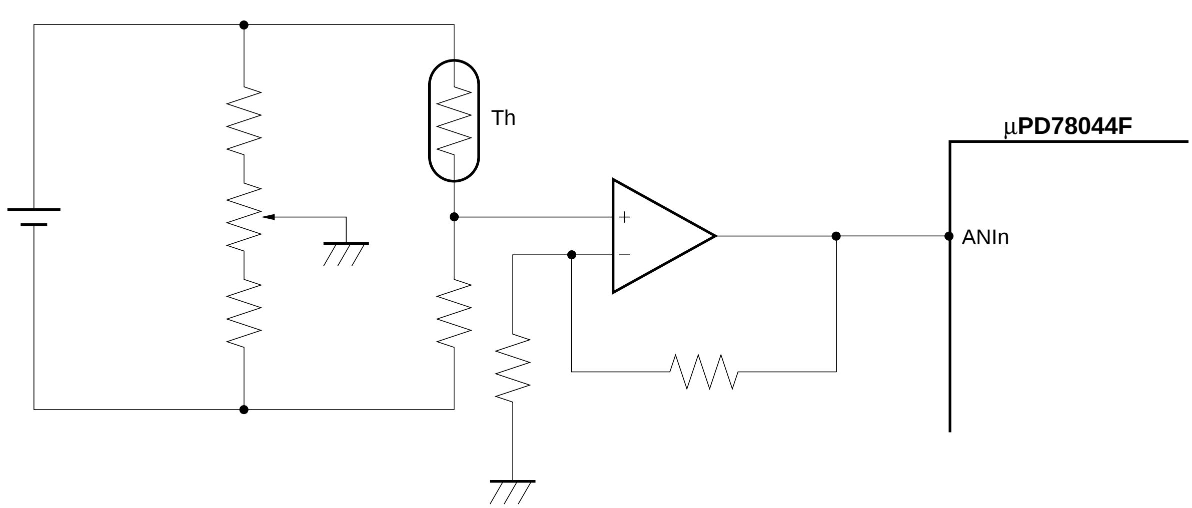
R : Resistance at some temperature T [°K]
T : Any temperature [°K]
R_0 : Resistance at the reference temperature T_0 [°K]
T_0 : Reference temperature [°K]
B : Constant determined from the reference temperature T_0 [°K] and T_0 [°K]

However, the B constant is not constant and is changed by the temperature. The B constant is transformed in the above equation and can be determined by the following equation.

$$B = \frac{1}{(1/T - 1/T_0)} \ln \frac{R}{R_0}$$

An example circuit is shown in Figure 9-8. This circuit is set so that 0 V is input at –20 °C and 5 V are input at +50 °C.

Figure 9-8. Thermometer Circuit Example



In this example circuit, because the thermistor characteristics are not linear, the input analog voltage is changed into a temperature from –20 °C to +50 °C by comparing the voltage with table data and not by a calculation. This conversion result is saved in the RAM (DSPDAT) as two BCD digits. Figure 9-9 shows the thermistor characteristics. Table 9-1 shows the relationship between the temperature and the A/D conversion value.

Also, the measurement method changes the average of the four conversion results into a temperature. Therefore, the conversion result is stored in the display area. Consequently, one datum in four is updated. For example, when measurement processing is performed every 250 ms, the display update period becomes one second.

Figure 9-9. Temperature and Output Characteristics

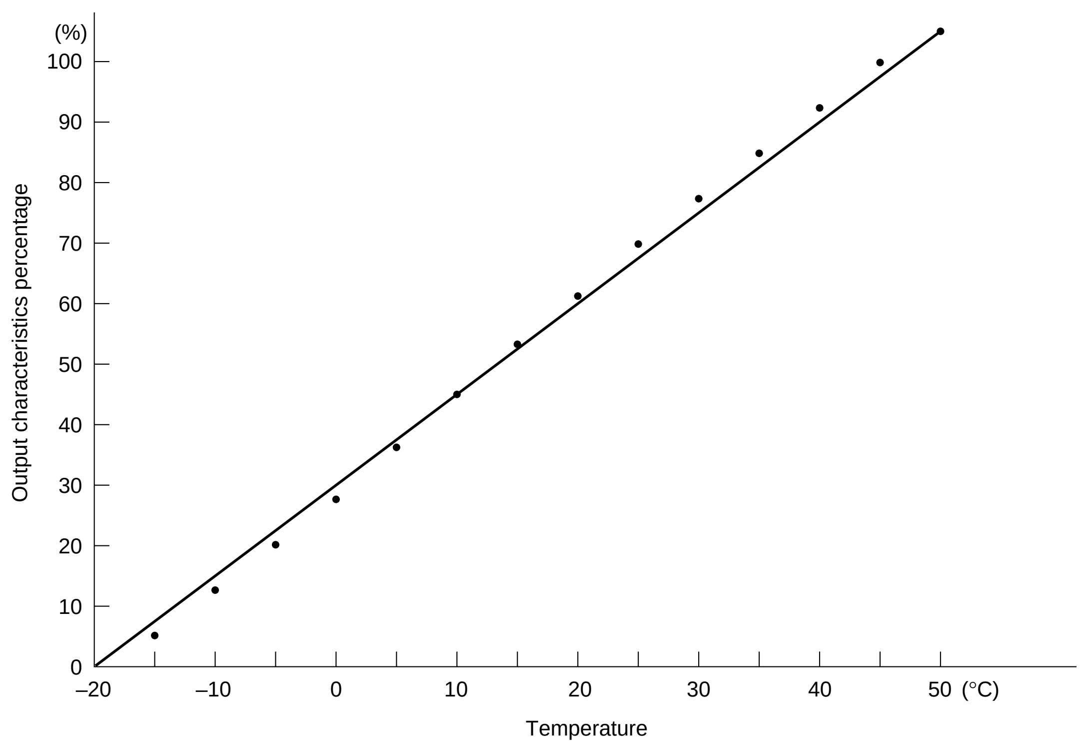


Table 9-1. A/D Conversion Values and Temperatures

Conversion value	Temperature [°C]	Conversion value	Temperature [°C]	Conversion value	Temperature [°C]	Conversion value	Temperature [°C]
00	–20.0	38	–2.5	82	15.5	CB	33.5
01	–19.5	3C	–1.5	86	16.5	CE	34.5
04	–18.5	40	–0.5	8B	17.5	D2	35.5
07	–17.5	44	0.5	8F	18.5	D6	36.5
0A	–16.5	48	1.5	93	19.5	D9	37.5
0C	–15.5	4C	2.5	97	20.5	DC	38.5
0F	–14.5	50	3.5	9B	21.5	E0	39.5
12	–13.5	54	4.5	9F	22.5	E3	40.5
16	–12.5	58	5.5	A3	23.5	E7	41.5
19	–11.5	5C	6.5	A8	24.5	EA	42.5
1C	–10.5	60	7.5	AC	25.5	ED	43.5
1F	–9.5	64	8.5	B0	26.5	F0	44.5
23	–8.5	69	9.5	B4	27.5	F3	45.5
26	–7.5	6D	10.5	B7	28.5	F6	46.5
2A	–6.5	71	11.5	BB	29.5	F9	47.5
2D	–5.5	75	12.5	BF	30.5	FC	48.5
31	–4.5	7A	13.5	C3	31.5	FE	49.5
35	–3.5	7E	14.5	C7	32.5	FF	50.0

(1) Package description

<Symbols declared as public>

THMETER : Name of the thermometer subroutine
DSPDAT : Display data storage area
CNTPRO : Number of inputs test counter
MINUSF : Minus temperature display flag
T250MSF : Flag for setting 250 ms

<Registers used>

AX, BC, HL

<RAM used>

Name	Use	Attributes	Bytes
ADDAT	A/D conversion value storage	SADDR	4
DSPDAT	Display data storage		2
CNTPRO	Number of inputs test counter		1
WORKCT	Work counter for loop operation		

<Flag used>

Name	Use
T250MSF	When set, measurement processing is executed.
MINUSF	Set when the temperature is minus.

<Nesting>

1 level, 2 bytes

<Hardware used>

A/D converter

<Initial settings>

Channel selection and operation start for A/D converter ADM=#1000xxx1B

<Startup procedure>

In timer processing, set the T250MSF flag in each measurement period. Then, call THMETER at least once during the measurement period.

(2) Use example

```
        EXTRN   THMETER,DSPDAT,CNTPRO
        EXTBIT  MINUSF,T250MSF

AD_DAT  DSEG    SADDR
CT250MS:DS      1                       ; 250-ms measurement counter
LEDD:   DS      4                       ; LED display area
DIGCT:  DS      1                       ; LED display digit counter

VETM3   CSEG    AT 12H
        DW      INTTM3                  ; Vector address setting of the watch timer

        MOV     TMC2,#00100110B         ; 1.95-ms setting for the watch timer
        CLR1    TMMK3
          :
          :
        CT250MS=#128
        CNTPRO=#4
        ADM=#10000011B                  ; ANI1 pin, operation start
          :
          :

;***************************************
;   Watch timer interrupt servicing
;          1.95-ms interval
;***************************************
INTTM3:                                 ; 1.95-ms interrupt servicing
          :
          :
        DBNZ    CT250MS,$RTNTM3
        MOV     CT250MS,#128            ; 250-ms had elapsed
        SET1    T250MSF
RTNTM3:
          :
          :
        RETI
```

(3) SPD chart

```
THMETER
  IF : 250 ms has elapsed (T250MS = 1)
  THEN
      Clear T250MS
      Save the A/D conversion value in memory
      IF : 4 conversions are saved in memory
      THEN
          Average four A/D conversion values
          (FOR : WORKCT = #0 ; WORKCT < #70 ; WORKCT++)
              IF : conversion result > comparison data for temperature conversion
              THEN
                  Update comparision data
              ELSE
                  BREAK
          IF : Temperature data is a negative value
          THEN
              Set in the minus state
                Set MINUSF
          Convert the temperature data into a decimal number and save in memory
```

(4) Program listing

```
        PUBLIC  THMETER,DSPDAT,CNTPRO,T250MSF,MINUSF

AD_DAT  DSEG    SADDR
ADDAT:  DS      4                      ; A/D conversion result storage area
DSPDAT: DS      2                      ; Display data
CNTPRO: DS      1                      ; Test the number of inputs.
WORKCT: DS      1

AD_FLG  BSEG
T250MSF DBIT                           ; 250-ms setting
MINUSF  DBIT                           ; Negative data setting

TH_SEG  CSEG
;********************************
;*   Temperature data setting
;********************************
THMETER:
        if_bit(T250MSF)                ; 250 ms
            CLR1    T250MSF
            A=ADCR
            A<->ADDAT
            A<->ADDAT+1
            A<->ADDAT+2
            A<->ADDAT+3

            CNTPRO--
            if(CNTPRO==#0)
                CNTPRO=#4
                AX=#0H
                HL=#ADDAT              ; Data storage address
                for(WORKCT=#0;WORKCT<#4;WORKCT++)
                    A+=[HL]
                    HL++
                    if_bit(CY)         ; Carry present.
                        X++            ; Carry
                    endif
                next

                A<->X
                C=#4
                AX/=C                  ; AX/C=AX (quotient)...C (remainder)
                if(C>=#2) (A)          ; Remainder processing (carry ≥ 2)
                    X++                ; Carry processing
                endif

                A=X                    ; Convert to temperature data.
                B=#0
                HL=#THRTBL
                if(A==#0FFH)
                    B=#70
                else
                    for(WORKCT=#0;WORKCT<#70;WORKCT++)
                        if(X>=[HL+B]) (A)
                            B++
                        else
                            break
                        endif
                    next
```

```
            endif

            CLR1    MINUSF
            A=#20                       ; Temperature data 20
            B-=A
            if_bit(CY)                  ; To decimal conversion
                SET1    MINUSF
                A=#0
                A-=B                    ; Take the absolute value of data
                A<->B
            endif
            X=#0                        ; Decimal conversion
            A=B
            A<->X
            C=#10
            AX/=C                       ; Temperature data/10
            DSPDAT=C (A)                ; Update display data.
            (DSPDAT+1)=X (A)
        endif
    endif
    RET
```

```
THRTBL:
;
        DB      1               ; –19.5
        DB      4               ; –18.5
        DB      7               ; –17.5
        DB      0AH             ; –16.5
        DB      0CH             ; –15.5
        DB      0FH             ; –14.5
        DB      12H             ; –13.5
        DB      16H             ; –12.5
        DB      19H             ; –11.5
        DB      1CH             ; –10.5
        DB      1FH             ; –9.5
        DB      23H             ; –8.5
        DB      26H             ; –7.5
        DB      2AH             ; –6.5
        DB      2DH             ; –5.5
        DB      31H             ; –4.5
        DB      35H             ; –3.5
        DB      38H             ; –2.5
        DB      3CH             ; –1.5
        DB      40H             ; –0.5
        DB      44H             ; +0.5
        DB      48H             ; 1.5
        DB      4CH             ; 2.5
        DB      50H             ; 3.5
        DB      54H             ; 4.5
        DB      58H             ; 5.5
        DB      5CH             ; 6.5
        DB      60H             ; 7.5
        DB      64H             ; 8.5
        DB      69H             ; 9.5
        DB      6DH             ; 10.5
        DB      71H             ; 11.5
        DB      75H             ; 12.5
        DB      7AH             ; 13.5
        DB      7EH             ; 14.5
        DB      82H             ; 15.5
        DB      86H             ; 16.5
        DB      8BH             ; 17.5
        DB      8FH             ; 18.5
        DB      93H             ; 19.5
        DB      97H             ; 20.5
        DB      9BH             ; 21.5
        DB      9FH             ; 22.5
        DB      0A3H            ; 23.5
        DB      0A8H            ; 24.5
        DB      0ACH            ; 25.5
        DB      0B0H            ; 26.5
        DB      0B4H            ; 27.5
        DB      0B7H            ; 28.5
        DB      0BBH            ; 29.5
        DB      0BFH            ; 30.5
        DB      0C3H            ; 31.5
        DB      0C7H            ; 32.5
        DB      0CBH            ; 33.5
        DB      0CEH            ; 34.5
        DB      0D2H            ; 35.5
        DB      0D6H            ; 36.5
```

```
DB      0D9H            ; 37.5
DB      0DCH            ; 38.5
DB      0E0H            ; 39.5
DB      0E3H            ; 40.5
DB      0E7H            ; 41.5
DB      0EAH            ; 42.5
DB      0EDH            ; 43.5
DB      0F0H            ; 44.5
DB      0F3H            ; 45.5
DB      0F6H            ; 46.5
DB      0F9H            ; 47.5
DB      0FCH            ; 48.5
DB      0FEH            ; 49.5
```

9.3 ANALOG KEY INPUT

The A/D converter is used to read in 16 keys. In order to perform key input, the circuit is configured so that when a key is pressed, a voltage unique to that key is input into the A/D converter.

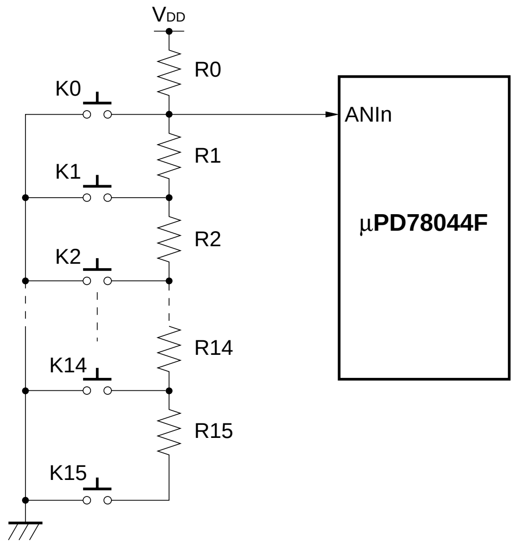
In this example, because 16 different keys are read in, the V_{DD} voltage is divided into 16 levels. This voltage is converted into a key code. Table 9-2 shows the relationship between the input voltage and key code (00H-0FH). When there is no key input, the key code is 10H.

Table 9-2. Input Voltages and Key Codes

Input voltage (V)	A/D conversion value	Key code
GND	00-07H	00H
$1/16V_{DD}$	08-17H	01H
$2/16V_{DD}$	18-27H	02H
$3/16V_{DD}$	28-37H	03H
$4/16V_{DD}$	38-47H	04H
$5/16V_{DD}$	48-57H	05H
$6/16V_{DD}$	58-67H	06H
$7/16V_{DD}$	68-77H	07H
$8/16V_{DD}$	78-87H	08H
$9/16V_{DD}$	88-97H	09H
$10/16V_{DD}$	98-A7H	0AH
$11/16V_{DD}$	A8-B7H	0BH
$12/16V_{DD}$	B8-C7H	0CH
$13/16V_{DD}$	C8-D7H	0DH
$14/16V_{DD}$	D8-E7H	0EH
$15/16V_{DD}$	E8-F7H	0FH
V_{DD}	F8-FFH	10H

Figure 9-10 shows an example circuit implementing the relationship between the input voltage and the key code. However, when two or more keys are pressed in this circuit, the key having the smaller code is given priority and read in.

Figure 9-10. Analog Key Input Circuit Example



Resistors R0 to R15 used in the circuit shown in Figure 9-10 can be determined from the following equation.

$$\sum_{K=1}^{n} R_K = \frac{n \times R0}{16 - n}$$

Table 9-3 shows the resistances of R1 to R15 based on this equation when R0 was 1 kΩ. (Because the resistances are based on the color-coded display on commercial resistors, the calculation results may differ.)

Table 9-3. Resistances of R1 to R15

Resistor number	Resistance (Ω)	Resistor number	Resistance (Ω)	Resistor number	Resistance (Ω)
R1	68	R6	150	R11	560
R2	75	R7	180	R12	750
R3	82	R8	220	R13	1.3 k
R4	100	R9	270	R14	2.7 k
R5	120	R10	390	R15	8.2 k

In this program, the analog voltage that was input is converted into a key code listed in Table 9-2. After chattering is absorbed, the code is saved in RAM. Chattering absorption uses a technique where a key becomes valid when the key code matches five consecutive times. For example, when sampling is performed every 5 ms, chattering lasting 20 ms to 25 ms is absorbed. When the key input changed, the key change flag (KEYCHG) is set.

(1) Package description

<Symbols declared as public>

AKEYIN : Name of analog key input subroutine
KEYDAT : Key code storage area
PASTDT : Key code storage area for chattering absorption
CHATCT : Chattering absorption counter
KEYCHG : Key change test flag
CHTENDF : End of chattering absorption test flag
KEYOFF : Key code when there is no key input

<Registers used>

A

<RAM used>

Name	Use	Attributes	Bytes
PASTDAT	Key code storage for chattering absorption	SADDR	1
KEYDAT	Key code storage		
CHATCNT	Chattering counter		

<Flags used>

Name	Use
KEYCHG	Set when the key changes
CHTENDF	Set at the end of chattering absorption

<Nesting>

1 level, 2 bytes

<Hardware used>

A/D converter

<Initial settings>

Channel selection and operation start of A/D converter ADM=#1000xxx1B

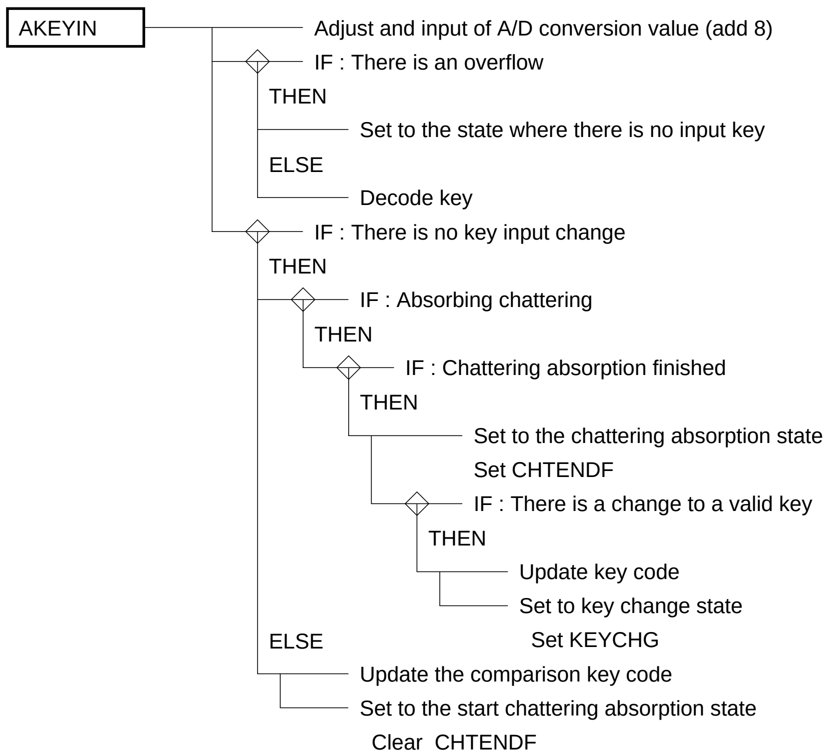
<Startup procedure>

- Call AKEYIN in each constant interval.
- Read in the key code after testing the key change flag. Also, because the key change flag is not cleared in the subroutine, clear after testing the flag.

(2) Use example

```
        EXTRN   AKEYIN,KEYDAT,PASTDT,CHATCT
        EXTRN   KEYOFF

        EXTBIT  KEYCHG,CHTENDF

VETM3   CSEG    AT 12H
        DW      INTTM3                     ; Vector address setting of the watch timer

MAINDAT DSEG    SADDR
CT5MS:  DS      1

        TMC2=#00100110B
        CLR1    TMMK3
        CT5MS=#3

        KEYDAT=#KEYOFF                     ; Set the OFF data in the key data.
        PASTDT=#KEYOFF
        CHATCT=#CHAVAL                     ; Set the chattering count to 5 times.
        CLR1    CHTENDF
        CLR1    KEYCHG
        ADM=#10000101B                     ; ANI2 pin, operation start
        EI
          :
          :
        if_bit(KEYCHG)                     ; Did the key change?
                CLR1    KEYCHG
                ; Key input processing
        endif
          :
          :

;***************************************
;     Watch timer interrupt servicing
;             1.95-ms interval
;***************************************
INTTM3:                                    ; 1.95-ms interrupt servicing
          :
          :
        DBNZ    CT5MS,$RTNTM3
        MOV     CT5MS,#3                   ; 1.95 ms x 3 elapsed
        CALL    !AKEYIN
RTNTM3:
          :
          :
        RETI
```

(3) SPD chart


AKEYIN
Adjust and input of A/D conversion value (add 8)
IF : There is an overflow
THEN
Set to the state where there is no input key
ELSE
Decode key
IF : There is no key input change
THEN
IF : Absorbing chattering
THEN
IF : Chattering absorption finished
THEN
Set to the chattering absorption state
Set CHTENDF
IF : There is a change to a valid key
THEN
Update key code
Set to key change state
Set KEYCHG
ELSE
Update the comparison key code
Set to the start chattering absorption state
Clear CHTENDF

(4) Program listing

```
        PUBLIC  AKEYIN,KEYDAT,PASTDT
        PUBLIC  CHATCT,KEYOFF
        PUBLIC  KEYCHG,CHTENDF
AK_DAT  DSEG    SADDR
KEYDAT: DS      1                       ; Key data storage area
PASTDT: DS      1                       ; Chattering key data
CHATCT: DS      1                       ; Chattering counter

AK_FLG  BSEG
KEYCHG  DBIT                            ; Key change.
CHTENDF DBIT                            ; End of chattering absorption state

KEYOFF  EQU     10H                     ; OFF key data
CHAVAL  EQU     5                       ; Chattering absorption count

AK_SEG  CSEG
;**************************
;*   Analog key input
;**************************
AKEYIN:
        A=ADCR                          ; A/D conversion input
        A+=#8                           ; Data adjustment
        if_bit(CY)
            A=#KEYOFF                   ; Set to the no input key state.
        else
            A>>=1                       ; Decode key
            A>>=1
            A>>=1
            A>>=1
            A&=#0FH
        endif
        if(A==PASTDT)                   ; No key change
            if_bit(!CHTENDF)            ; Absorbing chattering
                CHATCT--                ; End of chattering absorption
                if(CHATCT==#0)
                    SET1    CHTENDF     ; Set to the end of chattering absorption state
                    A=PASTDT
                    if(A!=KEYDAT)       ; There is a valid key change.
                        KEYDAT=A        ; Update key data.
                        SET1    KEYCHG  ; Set to the key change state.
                    endif
                endif
            endif
        else
            PASTDT=A                    ; Update previous key data.
            CHATCT=#CHAVAL-1            ; Start chattering absorption.
            CLR1    CHTENDF
        endif
        RET
```

9.4 4-CHANNEL INPUT A/D CONVERSION

This section describes an A/D conversion method where four channels are scanned. A/D conversion is started by a software start.

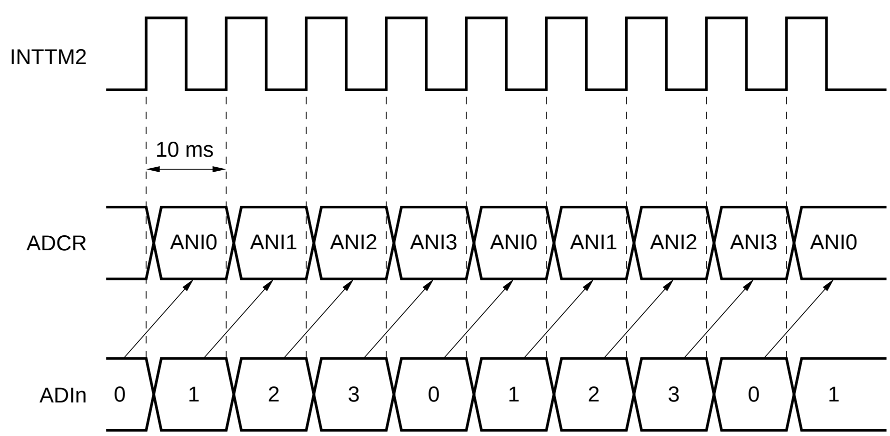
Analog voltages input to the four selected channels undergo A/D conversion. The conversion result of each channel is saved in RAM.

An interrupt request is generated by 8-bit timer/event counter 1 and the conversion result is read into processing for the interrupt request and channel conversion is performed. Because the time set for 8-bit timer/event counter 1 is 10 ms, measuring the waiting time for A/D conversion is not necessary.

Caution When the interrupt time changes, set the following.

- **The timer is set to a value longer than { A/D-conversion-completion-time + interrupt-return-time + interrupt-servicing-time }**
- **Flags are tested at the end of conversion.**

Figure 9-11. Timing Chart in the 4-Channel Scanning Mode



(1) Package description

<Symbols declared as public>

- Output parameters

  M_CH0: Stores the conversion result of channel 0
  M_CH1: Stores the conversion result of channel 1
  M_CH2: Stores the conversion result of channel 2
  M_CH3: Stores the conversion result of channel 3

<Registers used>

A

<RAM used>

Name	Use	Attributes	Bytes
M_CH0	Storage area for channel 0 conversion result	SADDR	1
M_CH1	Storage area for channel 1 conversion result	SADDR	1
M_CH2	Storage area for channel 2 conversion result	SADDR	1
M_CH3	Storage area for channel 3 conversion result	SADDR	1
M_MODE	Mode storage area	SADDR	1

<Nesting>

1 level, 3 bytes

<Hardware used>

- A/D converter
- 8-bit timer/event counter 1
- Port 1 (P10-P13)

<Initial settings>

• Channel selection and operation start of the A/D converter	ADM=#1000xxxxB
• Channel number selection of A/D converter	ADIS=#00000100B
• 10-ms interval for 8-bit timer/event counter 1	TCL1=#00001101B
	TMC1=#00000001B
	CR10=#81
• TMMK1 interrupt enabled	

(2) Use example

```
EXTRN    M_CH0,M_CH1,M_CH2,M_CH3,M_MODE
;**************************************
;              Initialize
;**************************************
M4         CSEG                 ;
RES_STA:
    SEL  RB0                    ;
    DI                          ;
           .
           .
    ADM=#10000001B              ; A/D operation start, no external trigger, channel 0 selected
    ADIS=#00000100B             ; Analog input, 4 channels selected
    CR10=#81                    ; Modulo register 81 setting
    TCL1=#00001101B             ; Count/clock 8.2 kHz
    TMC1=#00000001B             ; Enable 8 bit/timer/register 1 operation
    CLR1   TMIF1                ; Clear timer 1 interrupt request flag.
    CLR1   TMMK1                ; Enable timer 1 interrupt.
    EI                          ;
    M_MODE=#0                   ; Set the initial value (0 channels) in the mode area
           .
           .
    while(forever)              ;
           .
           .
        A=M_CH0                 ; A <- channel 0 data
           .
           .
        A=M_CH1                 ; A <- channel 1 data
           .
           .
        A=M_CH2                 ; A <- channel 2 data
           .
           .
        A=M_CH3                 ; A <- channel 3 data
           .
           .
```

(3) SPD chart

[A/D conversion processing]

```
KASAN ──┬── Read in the conversion result of the channel in the previous A/D conversion
        ├── Change the channel
        └── ADM<-Changed channel selection
```

(4) Program listing

```
;
;*************************************
;             A/D conversion
;*************************************
;
$PC(044A)                                   ;
;
PUBLIC   M_CH0,M_CH1,M_CH2,M_CH3,M_MODE    ;
;
VEINTM1  CSEG     AT 16H
         DW   KASAN
;*************************************
;              RAM definition
;*************************************
             DSEG    SADDR
M_CH0:       DS      1                      ; RAM area for channel 0 addition
M_CH1:       DS      1                      ; RAM area for channel 1 addition
M_CH2:       DS      1                      ; RAM area for channel 2 addition
M_CH3:       DS      1                      ; RAM area for channel 3 addition
M_MODE:      DS      1                      ; Mode storage area
;
         CSEG                               ;
KASAN:
         SEL RB2                            ; Switch to bank 2.
         switch(M_MODE)                     ; Which channel is currently selected?
         case 0:                            ; Channel 0:
             M_CH0=ADCR (A)                 ; Transfer conversion result to RAM
             M_MODE++                       ;
             ADM=#10000011B                 ; Change channel selection to 1.
             break                          ;
         case 1:                            ; Channel 1:
             M_CH1=ADCR (A)                 ; Transfer conversion result to RAM
             M_MODE++                       ;
             ADM=#10000101B                 ; Change channel selection to 2.
             break                          ;
         case 2:                            ; Channel 2:
             M_CH2=ADCR (A)                 ; Transfer conversion result to RAM
             M_MODE++                       ;
             ADM=#10000111B                 ; Change channel selection to 3.
             break                          ;
         case 3:                            ; Channel 3:
             M_CH3=ADCR (A)                 ; Transfer conversion result to RAM
             M_MODE=#0                      ;
             ADM=#10000001B                 ; Change channel selection to 0.
             break                          ;
         ends                               ;
         RETI                               ;
         END
```

CHAPTER 10 APPLICATIONS OF FIP CONTROLLER/DRIVER

The functions of the FIP controller/driver are listed below. The differences between the μPD78044F, μPD78044H, μPD780208, and μPD780228 subseries are listed in Table 10-1.

(1) Segment signal output (DMA operation) by automatically reading display data and automatic output of digit signals
(2) Display mode register controlling FIP (fluorescent indicator panel) (See Table 10-1.)
(3) Those pins not used for FIP display can be used either as output port or I/O port pins (however, pins FIP0 through FIP12 of the μPD780208 subseries and pins FIP0 through FIP15 of the μPD780228 subseries are dedicated to display output). *
(4) Brightness can be set to one of eight steps by using display mode register 1 (DSPM1).
(5) Hardware for key scan application
- Generates an interrupt request signal (INTKS) indicating the key scan timing
- Key scan signals are output from segment output pins if data for key scanning is set to port (see Table 10-1).
- Key scan data output timing can be detected by key scan flag (KSF).
- * Whether the key scan timing is inserted can be selected (only for the μPD780228 subseries).

(6) High-voltage output buffer directly driving FIP
(7) Pull-down resistor can be connected by mask option to display output pins.
(8) Any digit signal output timing can be set by selecting display mode 2 with display mode register 0 (DSPM0) (μPD780208 subseries only).

* **Caution The format of the registers incorporated into the μPD780228 subseries differs from that of the registers incorporated into the μPD78044F, μPD78044H, and μPD780208 subseries. When using any of the sample programs described in this chapter with the μPD780228 subseries, replace the register settings with those for the μPD780228 subseries.**

★ Table 10-1. Differences between μPD78044F, μPD78044H, μPD780208, and μPD780228 Subseries

<table>
<tr><th>Subseries / Item</th><th>μPD78044F subseries</th><th>μPD78044H subseries</th><th>μPD780208 subseries</th><th>μPD780228 subseries</th></tr>
<tr><td>Number of segments</td><td colspan="2">9-24</td><td>9-40</td><td rowspan="2">Up to 48 for total number of segments and digits</td></tr>
<tr><td>Number of digits</td><td colspan="3">2-16</td></tr>
<tr><td>Display mode</td><td colspan="2">• Segment type</td><td colspan="2">• Segment type
• Character type
• Type that a segment extends two or more grids</td></tr>
<tr><td>Multiplexed key scan port</td><td colspan="2">Ports 11 and 12</td><td>Ports 8-12</td><td>Ports 7-10</td></tr>
<tr><td>Controlling register</td><td colspan="2">Display mode registers 0 and 1 (DSPM0 and DSPM1)</td><td colspan="2">Display mode registers 0-2 (DSPM0-DSPM2)</td></tr>
</table>

Figure 10-1. Format of Display Mode Register 0 (μPD78044F and μPD78044H Subseries)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
DSPM0	KSF	DSPM06	0	0	SEGS3	SEGS2	SEGS1	SEGS0	FFA0H	00H	R/W[Note 1]

SEGS3	SEGS2	SEGS1	SEGS0	Number of display segments
0	0	0	0	9
0	0	0	1	10
0	0	1	0	11
0	0	1	1	12
0	1	0	0	13
0	1	0	1	14
0	1	1	0	15
0	1	1	1	16
1	0	0	0	17
1	0	0	1	18
1	0	1	0	19
1	0	1	1	20
1	1	0	0	21
1	1	0	1	22
1	1	1	0	23
1	1	1	1	24

DSPM06	Mode setting for the noise eliminator of the subsystem clock[Note 2]
0	2.5 MHz < f_X ≤ 5.0 MHz
1	1.25 MHz < f_X ≤ 2.5 MHz

KSF	Timing status
0	Display timing
1	Key scan timing

Notes 1. Bit 7 (KSF) is read-only.

2. Specify a value in accordance with the oscillation frequency of the main system clock (f_X). The noise eliminator can be used during FIP display operation.

Caution When using the FIP controller/driver with a main system clock of 1.25 MHz, use the main system clock (TCL24 (bit 4 of timer clock selection register 2 (TCL2)) = 0) for the watch timer.

Remark f_X: Oscillation frequency of the main system clock

Figure 10-2. Format of Display Mode Register 0 (μPD780208 Subseries) (1/2)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
DSPM0	KSF	DSPM06	DSPM05	SEGS4	SEGS3	SEGS2	SEGS1	SEGS0	FFA0H	00H	R/W

R/W

SEGS4	SEGS3	SEGS2	SEGS1	SEGS0	Number of display segments (display mode 1)	Number of display outputs (display mode 2)
0	0	0	0	0	9	9
0	0	0	0	1	10	10
0	0	0	1	0	11	11
0	0	0	1	1	12	12
0	0	1	0	0	13	13
0	0	1	0	1	14	14
0	0	1	1	0	15	15
0	0	1	1	1	16	16
0	1	0	0	0	17	17
0	1	0	0	1	18	18
0	1	0	1	0	19	19
0	1	0	1	1	20	20
0	1	1	0	0	21	21
0	1	1	0	1	22	22
0	1	1	1	0	23	23
0	1	1	1	1	24	24
1	0	0	0	0	25	25
1	0	0	0	1	26	26
1	0	0	1	0	27	27
1	0	0	1	1	28	28
1	0	1	0	0	29	29
1	0	1	0	1	30	30
1	0	1	1	0	31	31
1	0	1	1	1	32	32
1	1	0	0	0	33	33
1	1	0	0	1	34	34
1	1	0	1	0	35	35
1	1	0	1	1	36	36
1	1	1	0	0	37	37
1	1	1	0	1	38[Note]	38
1	1	1	1	0	39[Note]	39
1	1	1	1	1	40[Note]	40

Note If the total number of digits and segments exceeds 53, digits have precedence over segments.

Figure 10-2. Format of Display Mode Register 0 (μPD780208 Subseries) (2/2)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
DSPM0	KSF	DSPM06	DSPM05	SEGS4	SEGS3	SEGS2	SEGS1	SEGS0	FFA0H	00H	R/W Note 1

R/W

DSPM05	Setting of display mode
0	Display mode 1 (segment/character type)
1	Display mode 2 (type that a segment extends two or more grids)

R/W

DSPM06	Mode setting for the noise eliminator of the subsystem clock Note 2
0	2.5 MHz < f_X ≤ 5.0 MHz
1	1.25 MHz < f_X ≤ 2.5 MHz Note 3

R

KSF	Timing status
0	Display timing
1	Key scan timing

Notes 1. Bit 7 (KSF) is read-only.

2. Specify a value in accordance with the oscillation frequency of the main system clock (f_X). The noise eliminator can be used during FIP display operation.

3. When f_X is used from above 1.25 MHz to 2.5 MHz, set 1 in DSPM06 before FIP display.

Caution When using the FIP controller/driver with a main system clock of 1.25 MHz, use the main system clock (TCL24 (bit 4 of timer clock selection register 2 (TCL2)) = 0) for the watch timer.

Remark f_X: Oscillation frequency of the main system clock

*

Figure 10-3. Format of Display Mode Register 0 (μPD780228 Subseries)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
DSPM0	DSPEN	0	FOUT5	FOUT4	FOUT3	FOUT2	FOUT1	FOUT0	FF90H	10H	R/W

FOUT5	FOUT4	FOUT3	FOUT2	FOUT1	FOUT0	Number of FIP output pins
0	1	0	1	1	1	17-24
0	1	1	1	1	1	25-32
1	0	0	1	1	1	33-40
1	0	1	1	1	1	41-48
Other than the above						Setting prohibited

DSPEN	Enabling or disabling FIP display
0	Enable FIP display
1	Disable FIP display

Cautions 1. Always set bit 6 to 0.

2. When bit 7 (DSPEN) is 1, do not write data into bits other than DSPEN.

3. The output latch of the port multiplexed with the pins used for FIP output must be set to 0.

Figure 10-4. Format of Display Mode Register 1 (μPD78044F and μPD78044H Subseries)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
DSPM1	DIGS3	DIGS2	DIGS1	DIGS0	DIMS3	DIMS2	DIMS1	DIMS0	FFA1H	00H	R/W

DIMS0	Display cycle selection
0	1024/f_X as 1 display cycle (One display cycle is 204.8 μs at 5.0 MHz.)
1	2048/f_X as 1 display cycle (One display cycle is 409.6 μs at 5.0 MHz.)

DIMS3	DIMS2	DIMS1	Cut width of the digit signal
0	0	0	1/16
0	0	1	2/16
0	1	0	4/16
0	1	1	6/16
1	0	0	8/16
1	0	1	10/16
1	1	0	12/16
1	1	1	14/16

DIGS3	DIGS2	DIGS1	DIGS0	Number of display digits
0	0	0	0	Disabled display (static display)Note
0	0	0	1	2
0	0	1	0	3
0	0	1	1	4
0	1	0	0	5
0	1	0	1	6
0	1	1	0	7
0	1	1	1	8
1	0	0	0	9
1	0	0	1	10
1	0	1	0	11
1	0	1	1	12
1	1	0	0	13
1	1	0	1	14
1	1	1	0	15
1	1	1	1	16

Note When display is disabled, a port output latch can be operated to enable static display.

Remark f_X: Oscillation frequency of the main system clock

Figure 10-5. Format of Display Mode Register 1 (μPD780208 Subseries)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
DSPM1	DIGS3	DIGS2	DIGS1	DIGS0	DIMS3	DIMS2	DIMS1	DIMS0	FFA1H	00H	R/W

DIMS0	Setting of display mode cycle
0	1024/f_X as 1 display cycle (One display cycle is 204.8 μs at 5.0 MHz.)
1	2048/f_X as 1 display cycle (One display cycle is 409.6 μs at 5.0 MHz.)

DIMS3	DIMS2	DIMS1	Cut width of the FIP output signal
0	0	0	1/16
0	0	1	2/16
0	1	0	4/16
0	1	1	6/16
1	0	0	8/16
1	0	1	10/16
1	1	0	12/16
1	1	1	14/16

DIGS3	DIGS2	DIGS1	DIGS0	Number of display digits (display mode 1) DSPM05 = 0	Number of display patterns (display mode 2) DSPM05 = 1
0	0	0	0	Disabled display (static display)Note	Disabled display (static display)Note
0	0	0	1	2	2
0	0	1	0	3	3
0	0	1	1	4	4
0	1	0	0	5	5
0	1	0	1	6	6
0	1	1	0	7	7
0	1	1	1	8	8
1	0	0	0	9	9
1	0	0	1	10	10
1	0	1	0	11	11
1	0	1	1	12	12
1	1	0	0	13	13
1	1	0	1	14	14
1	1	1	0	15	15
1	1	1	1	16	16

Note When display is disabled, a port output latch can be operated to enable static display.

Remark f_X : Oscillation frequency of the main system clock
DSPM05 : Bit 5 of display mode register 0

*

Figure 10-6. Format of Display Mode Register 1 (μPD780228 Subseries)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
DSPM1	FBLK2	FBLK1	FBLK0	FPAT4	FPAT3	FPAT2	FPAT1	FPAT0	FF91H	01H	R/W

FPAT4	FPAT3	FPAT2	FPAT1	FPAT0	Number of display patterns
0	0	0	0	1	2
0	0	0	1	0	3
0	0	0	1	1	4
0	0	1	0	0	5
0	0	1	0	1	6
0	0	1	1	0	7
0	0	1	1	1	8
0	1	0	0	0	9
0	1	0	0	1	10
0	1	0	1	0	11
0	1	0	1	1	12
0	1	1	0	0	13
0	1	1	0	1	14
0	1	1	1	0	15
0	1	1	1	1	16
Other than the above					Setting prohibited

FBLK2	FBLK1	FBLK0	Blanking width for the FIP output signal
0	0	0	1/16
0	0	1	2/16
0	1	0	4/16
0	1	1	6/16
1	0	0	8/16
1	0	1	10/16
1	1	0	12/16
1	1	1	14/16

Caution When bit 7 (DSPEN) of display mode register 0 (DSPM0) is 1, do not write data into display mode register 1 (DSPM1).

Figure 10-7. Format of Display Mode Register 2 (μPD780208 Subseries) (1/2)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
DSPM2	0	0	USEG5	USEG4	USEG3	USEG2	USEG1	USEG0	FFA1H	00H	R/W

USEG5	USEG4	USEG3	USEG2	USEG1	USEG0	Number of write mask bits
0	0	0	0	0	0	None
0	0	0	0	0	1	1
0	0	0	0	1	0	2
0	0	0	0	1	1	3
0	0	0	1	0	0	4
0	0	0	1	0	1	5
0	0	0	1	1	0	6
0	0	0	1	1	1	7
0	0	1	0	0	0	8
0	0	1	0	0	1	9
0	0	1	0	1	0	10
0	0	1	0	1	1	11
0	0	1	1	0	0	12
0	0	1	1	0	1	13
0	0	1	1	1	0	14
0	0	1	1	1	1	15
0	1	0	0	0	0	16
0	1	0	0	0	1	17
0	1	0	0	1	0	18
0	1	0	0	1	1	19
0	1	0	1	0	0	20
0	1	0	1	0	1	21
0	1	0	1	1	0	22
0	1	0	1	1	1	23
0	1	1	0	0	0	24
0	1	1	0	0	1	25
0	1	1	0	1	0	26
0	1	1	0	1	1	27
0	1	1	1	0	0	28
0	1	1	1	0	1	29
0	1	1	1	1	0	30
0	1	1	1	1	1	31

Figure 10-7. Format of Display Mode Register 2 (μPD780208 Subseries) (2/2)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
DSPM2	0	0	USEG5	USEG4	USEG3	USEG2	USEG1	USEG0	FFA1H	00H	R/W

USEG5	USEG4	USEG3	USEG2	USEG1	USEG0	Number of write mask bits
1	0	0	0	0	0	32
1	0	0	0	0	1	33
1	0	0	0	1	0	34
1	0	0	0	1	1	35
1	0	0	1	0	0	36
1	0	0	1	0	1	37
1	0	0	1	1	0	38
1	0	0	1	1	1	39
Other than the above						Setting prohibited

*

Figure 10-8. Format of Display Mode Register 2 (μPD780228 Subseries)

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
DSPM2	KSF	KSM	0	0	0	0	FCYC1	FCYC0	FF92H	00H	R/W

FCYC1	FCYC0	Display cycle
0	0	$2^{12}f_X$ (819.2 μs)
0	1	$2^{11}/f_X$ (409.6 μs)
1	0	$2^{10}/f_X$ (204.8 μs)
1	1	Setting prohibited

KSM	Selection of key scan cycle insertion
0	Insert a key scan cycle
1	Insert no key scan cycle

KSF	Status of the key scan cycle
0	During a cycle other than the key scan cycle
1	During the key scan cycle

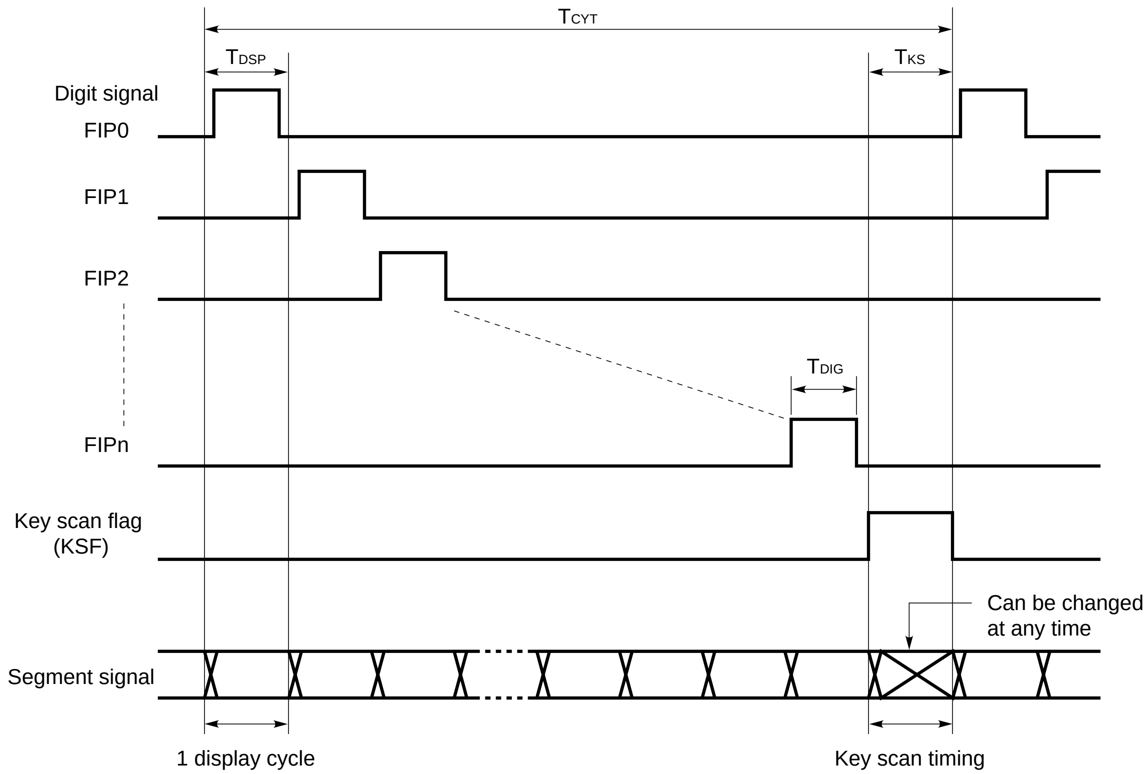
Cautions 1. Always set 0 in bits 2 to 5.

2. When bit 7 (DSPEN) of display mode register 0 (DSPM0) is 1, do not write data into display mode register 2 (DSPM2).

Remarks 1. f_X: Oscillation frequency of the main system clock

2. The values in parentheses apply to operation with f_X = 5.0 MHz.

Figure 10-9. FIP Controller Operation Timing



n : Number of display digits - 1
(2 to 16 digits selectable by display mode register 1 (DSPM1)

T_{DSP} : One display cycle
(1024/f_X (244 µs at 4.19 MHz) or 2048/f_X (488 µs at 4.19 MHz))

T_{KS} : Key scan timing
($T_{KS} = T_{DSP}$)

T_{CYT} : Display cycle
($T_{CYT} = T_{DSP}$ x (number of digits + 1))

T_{DIG} : Digit signal pulse width
(eight types, selectable with display mode register 1 (DSPM1))

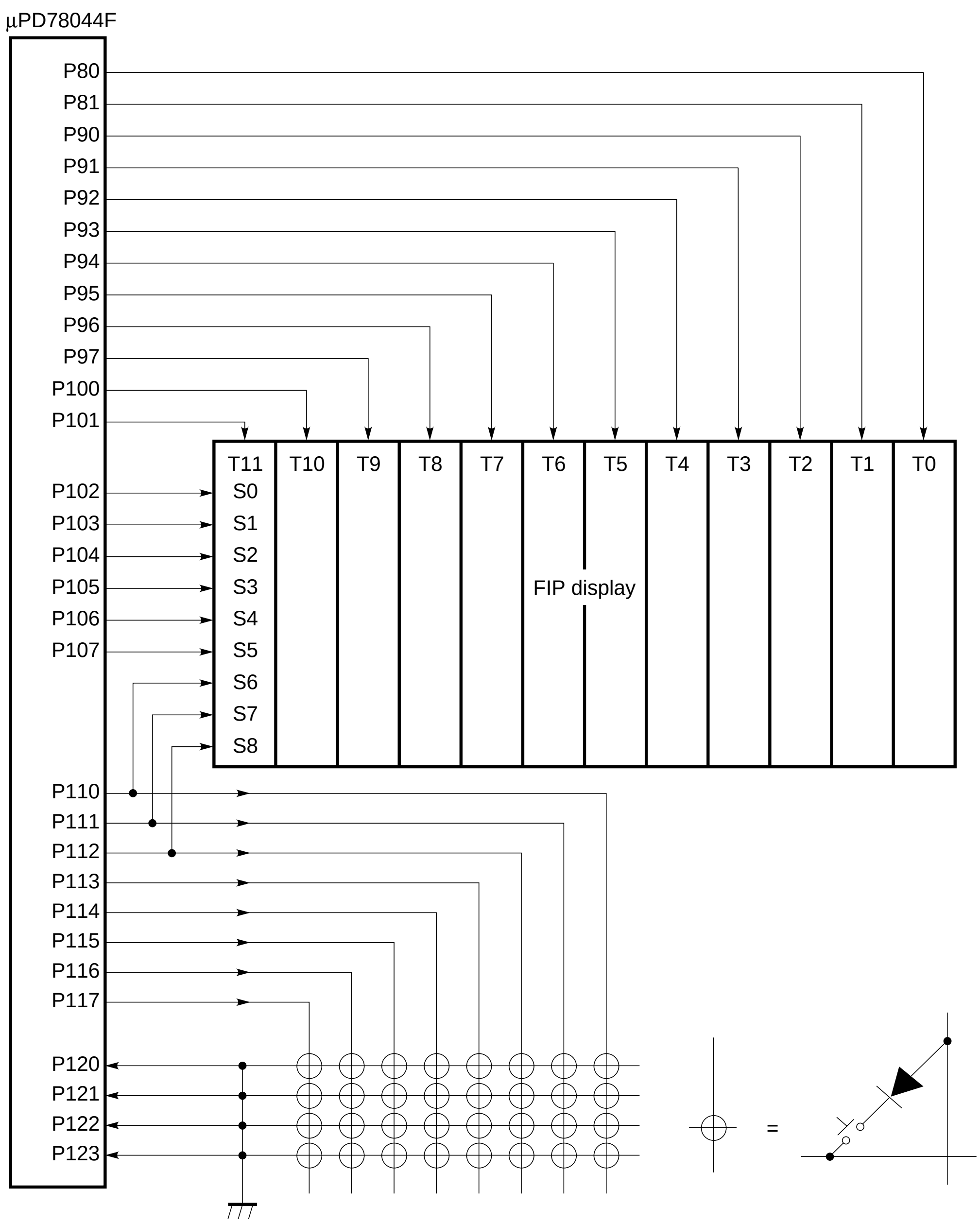
10.1 12-DIGIT DISPLAY FOR FIP AND KEY INPUT

This section shows an example of processing an FIP having 12 digits by 9 segments and 8 x 4 key inputs by using the FIP controller/driver of the µPD78044F subseries.

In this example, a key of the 8 x 4 key matrix that has been pressed is displayed on the first digit of the FIP (TO in Figure 10-10), and the data that has already been displayed is shifted one column to the left.

Figure 10-10 shows the configuration.

Figure 10-10. Configuration of 12-Digit FIP Display and Key Input



10.1.1 12-Digit FIP Display

(1) Setting the number of segments and number of digits

With the circuit shown in Figure 10-10, twelve digits are displayed using eight key scan signals.

The 9 segment x 12 digit FIP display mode is set. Nine segments is the minimum value for the selected number.

Figure 10-11 shows the pin layout according to the number of display digits for which nine segments are displayed.

Figure 10-11. Pin Layout for 9-Segment Display

Pin Name	Selected Number of Display Digits								
	Display stops	2	....	9	10	11	12	....	16
FIP0/P80	P80	T0		T0	T0	T0	T0		T0
FIP1/P81	P81	T1		T1	T1	T1	T1		T1
FIP2/P90	P90	P90		T2	T2	T2	T2		T2
FIP3/P91	P91	P91		T3	T3	T3	T3		T3
FIP4/P92	P92	P92		T4	T4	T4	T4		T4
FIP5/P93	P93	P93		T5	T5	T5	T5		T5
FIP6/P94	P94	P94		T6	T6	T6	T6		T6
FIP7/P95	P95	P95		T7	T7	T7	T7		T7
FIP8/P96	P96	P96		T8	T8	T8	T8		T8
FIP9/P97	P97	P97		P97	T9	T9	T9		T9
FIP10/P100	P100	S0		S0	S0	T10	T10		T10
FIP11/P101	P101	S1		S1	S1	S0	T11		T11
FIP12/P102	P102	S2		S2	S2	S1	S0		T12
FIP13/P103	P103	S3		S3	S3	S2	S1		T13
FIP14/P104	P104	S4		S4	S4	S3	S2		T14
FIP15/P105	P105	S5		S5	S5	S4	S3		T15
FIP16/P106	P106	S6	....	S6	S6	S5	S4	....	S0
FIP17/P107	P107	S7		S7	S7	S6	S5		S1
FIP18/P110	P110	S8∨P110		S8∨P110	S8∨P110	S7∨P110	S6∨P110		S2∨P110
FIP19/P111	P111	P111		P111	P111	S8∨P111	S7∨P111		S3∨P111
FIP20/P112	P112	P112		P112	P112	P112	S8∨P112		S4∨P112
FIP21/P113	P113	P113		P113	P113	P113	P113		S5∨P113
FIP22/P114	P114	P114		P114	P114	P114	P114		S6∨P114
FIP23/P115	P115	P115		P115	P115	P115	P115		S7∨P115
FIP24/P116	P116	P116		P116	P116	P116	P116		S8∨P116
FIP25/P117	P117	P117		P117	P117	P117	P117		P117
FIP26/P120	P120	P120		P120	P120	P120	P120		P120
FIP27/P121	P121	P121		P121	P121	P121	P121		P121
FIP28/P122	P122	P122		P122	P122	P122	P122		P122
FIP29/P123	P123	P123		P123	P123	P123	P123		P123
FIP30/P124	P124	P124		P124	P124	P124	P124		P124
FIP31/P125	P125	P125		P125	P125	P125	P125		P125
FIP32/P126	P126	P126		P126	P126	P126	P126		P126
FIP33/P127	P127	P127		P127	P127	P127	P127		P127

∨ : Logical add (OR)

(shaded box) : Area used by this program

(2) Display data memory

The display data memory is an area that stores the segment data to be displayed on an FIP.

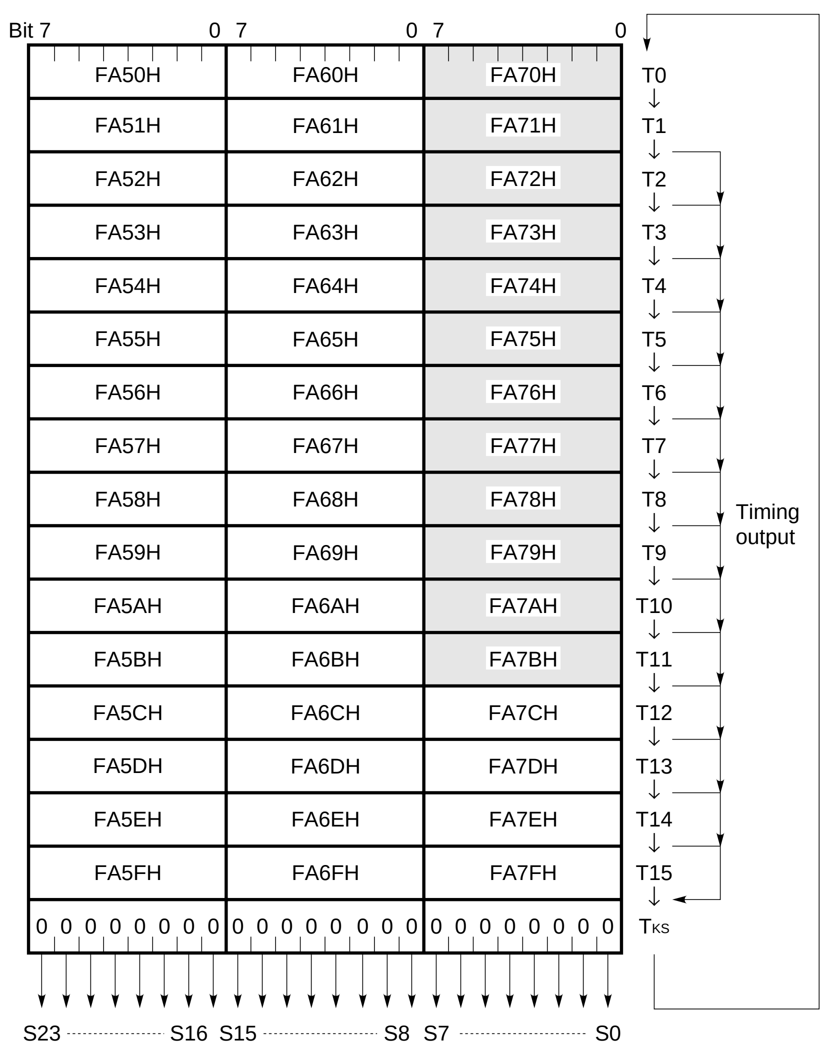
This area is mapped to addresses FA50H through FA7FH. The FIP controller reads data from this area independently of instruction operation to enable the display of the FIP and outputs a segment signal synchronized with digit signals (DMA operation).

Any unused portion of this area can be used as an ordinary RAM area.

When a key is scanned, all digit signals are cleared to 0, and the data of the output latches of ports 11 and 12 are output to the FIP18/P110 through pins FIP33/P127.

The shaded portion in Figure 10-12 indicates the area used by this program.

Figure 10-12. Relationship between Contents of Display Data Memory and Segment Output



(3) Display

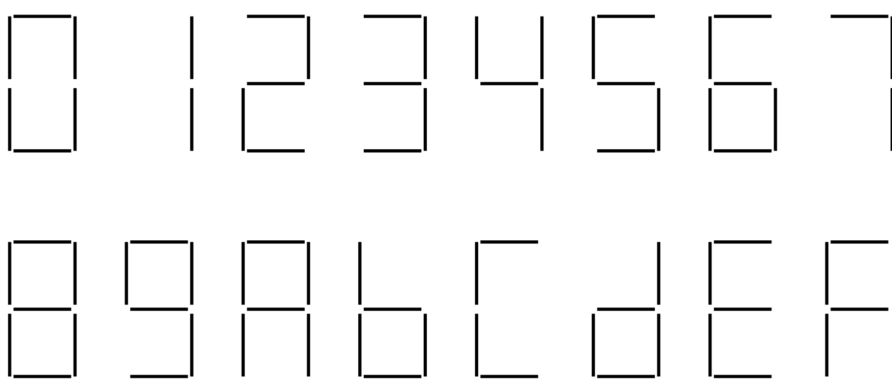
To display an FIP, output the data set for a display digit as the digit signals and write the data to be displayed as segment signals (i.e., to the display data memory).

As an example, the setting of display mode registers 0 and 1 when 9 segments and 12 digits are to be displayed is shown below.

Figure 10-13 shows an example display based on this setting.

- Setting of DSPM0
  DSPM0 = #00000000B ; Selects 9 segments
- Setting of DSPM1
  DSPM1 = #10110011B ; Selects 12 digits, a digit signal cut width of 2/16, and a display cycle of 488 μs (at 4.19 MHz)

Figure 10-13. Display Example



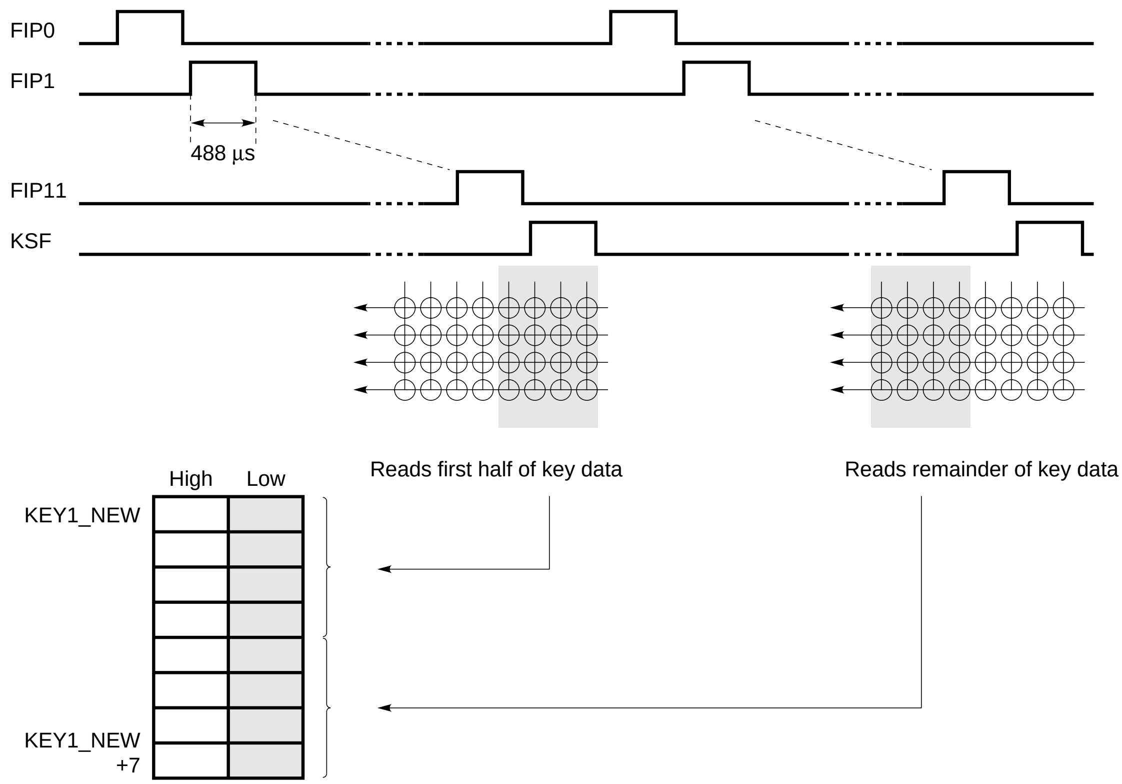
10.1.2 Key Input

An example of a program that receives input from an 8 x 4 key matrix is shown.

The circuit used for this program uses port 11 (P110 through P117) for the key scan signals and the lower 4 bits (P120 through P123) of port 12 for the key return signals (see Figure 10-10).

The key scan flag (KSF) is set to 1 while keys are scanned and is cleared to 0 during display. When this flag is set to 1, an interrupt request occurs and keys are input by this interrupt. Because not all of the 8 x 4 keys can be input during the time made available by one interrupt request (488 μs), the interrupt request must be issued twice to enable input of all the keys. The timing chart shown in Figure 10-14 illustrates how all the keys are input.

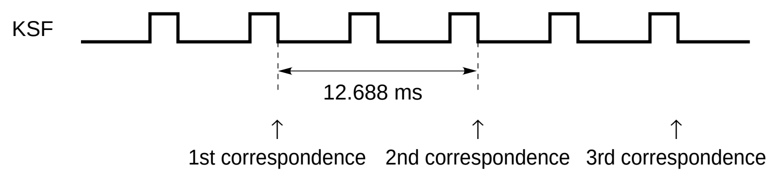
Figure 10-14. Key Interrupt Timing Chart



One input key corresponds to 1 bit and is stored in RAM. The RAM data is set according to the pressed key. When the key is released, the data is cleared. By sequentially testing each bit of RAM data starting from the first bit, therefore, the statuses of the keys can be checked. Chattering is compensated for by validating the key only if the key data coincides with the corresponding RAM bit three times in a row.

Because 12 digits are displayed and the keys are scanned every 12.688 ms (= 13 x 488 µs (display cycle selectable) x two times (number of interrupts necessary for inputting all the keys)) in this example, chattering of about 25 ms to 38 ms can be eliminated. If a key input is changed, the key change flag (F_KHENKA) is set. Figure 10-15 illustrates how chattering is eliminated.

Figure 10-15. Compensating for Chattering



To prevent unwanted data from being displayed during display, the timing is checked by the key scan flag (KSF) at the beginning and end of key scan processing.

10.1.3 Description of Package

(1) FIP display

An FIP display program is not included in the package. Refer to the explanation of initial setting and display data conversion processing in Section 10.1.4.

(2) Key input

Because key input processing is performed as interrupt processing, the key input processing performed by this package is executed when the INTKS interrupt request is enabled.

<Symbols declared as public>

- Output parameters

KEY1_OLD : Stores key bit after eliminating chattering
KEY1_NEW : Stores key bit while eliminating chattering
SCAN : Stores scanned key data
NEWKEYP : Stores RAM address used to store next key bit while eliminating chattering
F_KHENKA : Set if current key is found to be different from previous key after eliminating chattering

<Registers used>

Bank 2, AX, HL, DE, B

<RAM used>

Name	Use	Attributes	Bytes
C_CAHTA	Chattering counter	SADDR	1
KEY1_OLD	Previous key bit input storage area	SADDR	8
KEY1_NEW	Current key bit input storage area	SADDR	8
SCAN	Key scan data storage area	SADDR	1
NEWKEYP	Next key bit input storage area	SADDRP	2
WORK	Key data transfer area	SADDR	1
i	Loop processing work counter	SADDR	1
B_FIP1	Stores display data	SADDR	12

<Flags used>

Name	Use
F_KHENKA	Set upon change in key input.
F_KEYEND	Set when four keys are scanned.

<Nesting>

1 level, 3 bytes

<Hardware used>

- FIP controller/driver
- Port 11
- Port 12 (P120 through P123)

<Initial settings>

- Setting of DSPM0
  DSPM0 = #00000000B ; Selects 9 segments
- Setting of DSPM1
  DSPM1 = #10110011B ; 12 display digits, digit signal cut width of 2/16, and display cycle of 488 μs
- Port 11 output mode
  PM11 = #00000000B
- INTKS interrupt enable
  CLR1 KSMK

<Processing>

The input key data is stored into the KEY1_NEW area after the processing of the INTKS interrupt. All keys are completely input after the INTKS interrupt request has occurred two times. The determined key is stored into the KEY1_OLD area.

<Usage>

- Set RAM as follows after reset and start:
  NEWKEYP = #KEY1_NEW ; key bit storage RAM address
  SCAN = #00000001B ; key scan data initial value
- Input the key data after testing the key change flag.
  Because the key change flag is not cleared to 0 by interrupt processing, clear this flag after flag test.

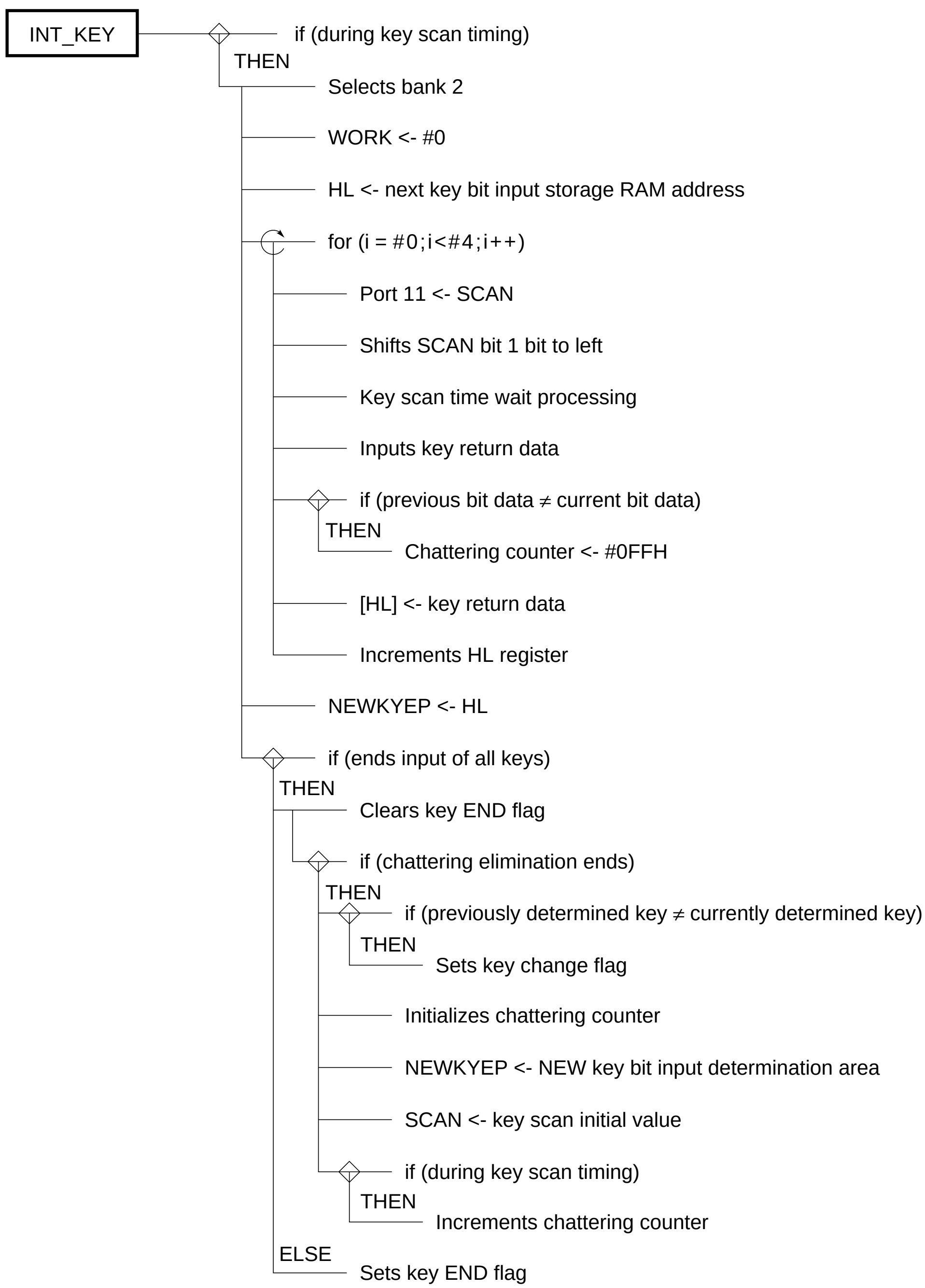
10.1.4 Example of Use

In the program example shown below, the initial setting of the key scan work area and display data conversion processing are performed for FIP display.

```
EXTRN    KEY1_OLD,KEY1_NEW,SCAN,NEWKEYP
EXTBIT   F_KHENKA
;
FIP1     EQU      0FA70H
;
B_FIP1:      DS       12                        ; FIP display 1st digit output BUF

M1           CSEG                               ;
RES_STA:

    DI                                          ;
    DSPM0=#00000000B                            ; Selects 9 segments
    DSPM1=#10110011B                            ; 12 display digits, cut width of 2/16, display cycle of 488 µs
    PM11=#00000000B                             ; Port 11 output mode
    CLR1    KSIF                                ; Clears the interrupt request flag
    CLR1    KSMK                                ; Enables INTKS interrupt
;
    SCAN=#00000001B                             ; Key scan data initial value
    NEWKEYP=#KEY1_NEW                           ;
    EI                                          ; INTKS interrupt (INT_KEY) started by enabled interrupt
;
    while(forever)                              ;
        IF_BIT(F_KHENKA)                        ; Key change flag set?
         CLR1    F_KHENKA                       ; Clears key change flag
            Decode processing

    for(B=#0;B<#12.B++)                         ; Converts 12 FIP display digits into output data and stores that
        HL=#B_FIP1                              ; data into the output BUF
        X=[HL+B]                                ;
        A=#0                                    ;
        AX+=#DISPLAY                            ;
        HL=AX                                   ;
        A=[HL]                                  ;
        HL=#FIP1                                ;
        [HL+B]=A                                ;
    next                                        ;
;
FIPDAT   CSEG                                   ;
DISPLAY:
        DB  11111100B                           ; 0
        DB  01100000B                           ; 1
        DB  11011010B                           ; 2
        DB  11110010B                           ; 3
        DB  01100110B                           ; 4
        DB  10110110B                           ; 5
        DB  10111110B                           ; 6
        DB  11100000B                           ; 7
        DB  11111110B                           ; 8
        DB  11110110B                           ; 9
        DB  11101110B                           ; A
        DB  00111110B                           ; B
        DB  10011100B                           ; C
        DB  01111010B                           ; D
        DB  10011110B                           ; E
        DB  10001110B                           ; F
    END
```

10.1.5 SPD Chart

[Key input processing (INTKS interrupt processing)]



10.1.6 Program Listing

```
;***********************************************
;    Key input processing (INTKS interrupt)
;***********************************************
;
$PC(044A)
PUBLIC   KEY1_OLD,KEY1_NEW,SCAN,NEWKEYP
PUBLIC   F-KHENKA
;
VEINTKS  CSEG    AT 1CH
         DW      INT_KEY
;
CHATDAT  EQU     02H                            ; Number of times chattering is eliminated
SCANDAT  EQU     00000001B                      ; First key scan data
;
;***********************************************
;                RAM definition
;***********************************************
;
KEYRAM      DSEG    SADDR
KEY1_OLD:   DS      8                           ; Previous key bit input determination data area
KEY1_NEW:   DS      8                           ; Current key bit input determination data area
C_CHATA:    DS      1                           ; Chattering counter
WORK:       DS      1                           ; Work area
SCAN:       DS      1                           ; Key scan data storage area
i:          DS      1                           ; Work counter area
            DSEG    SADDRP
NEWKEYP:    DS      2                           ; Next key bit input determination RAM address storage area
;
KEYFLG      BSEG
F_KHENKA    DBIT                                ; Key change flag
F_KEYEND    DBIT                                ; Key END flag
;
KEY     CSEG                                    ;
INT_KEY:
    IF_BIT(KSF)                                 ; Checks flag of INTKS
        SEL RB2                                 ; Selects bank 2
        WORK=#0                                 ;
        HL=NEWKEYP (AX)                         ; Stores next key storage RAM address into HL register
        for(i=#0;i<=#4;i++)                     ;
            P11=SCAN (A)                        ; Outputs key scan signal
            A=SCAN                              ; Shifts scan signal 1 bit to left
            ROL A,1                             ;
            SCAN=A                              ;
            for(B=#0;B<#6;B++) (A)              ; Scan time wait processing
            next                                ;
            A=P12                               ; Key return input
            A &= #0FH                           ;
            WORK=A                              ; Stores key return to WORK area
```

```
        if(A!=[HL])                         ;
            C_CHATA=#0FFH                   ; Clears chattering counter unless the same as the previous
        endif                               ; value
        [HL]=WORK (A)                       ;
        HL++                                ;
    next                                    ;
    NEWKEYP=HL (AX)                         ;
    if_bit(F_KEYEND)                        ; All keys input?

        CLR1     F_KEYEND                   ;
        if(C_CHATA>#CHATDAT)                ; End of chattering elimination?
            DE=#KEY1_OLD                    ;
            HL=#KEY1_NEW                    ; Previously determined key ≠ currently determined key?
            for(i=#0;i<#8;i++)              ;
                if([DE]!=[HL]) (A)          ; Sets key change flag
                    SET1    F_KHENKA        ;
                endif                       ;
                [DE]=[HL] (A)               ;
                DE++                        ;
                HL++                        ;
            next                            ;
            C_CHATA=#0                      ; Clears chattering counter
            NEWKEYP=#KEY1_NEW               ; Initializes next key bit input determination RAM address
            SCAN=#SCANDAT                   ; Initializes key scan data
        else                                ;
            if_bit(KSF)                     ; Checks INTKS flag
                C_CHATA++                   ; Increments chattering counter if OK
            endif                           ;
        endif                               ;
    else                                    ;
        SET1    F_KEYEND                    ;
    endif                                   ;
ENDIF                                       ;
RETI                                        ;
END
```

CHAPTER 11 APPLICATIONS OF 6-BIT UP/DOWN COUNTER

The 6-bit up/down counter is incremented or decremented at the valid edge of the CI0/P03/INTP3 pin.

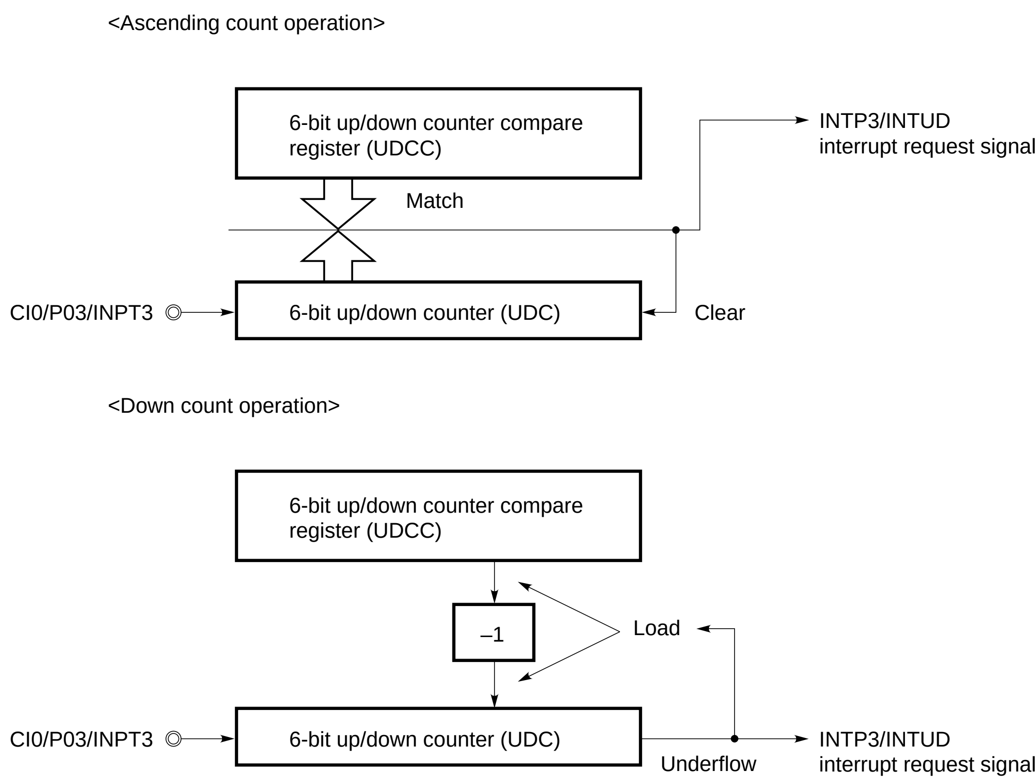
This counter uses a 6-bit up/down register (UDC) to count the number of count pulses input to the CI0/P03/INTP3 pin (see Figure 11-1).

If the value of the UDC coincides with the value of a 6-bit up/down counter compare register (UDCC) in ascending count mode, an interrupt request flag (PIF3) is set, and the UDC is cleared to 0.

If the UDC underflows in the descending count mode, the interrupt request flag (PIF3) is set, and a value of UDCC minus 1 is loaded into the UDC.

The 6-bit up/down counter is controlled by a 6-bit up/down counter control register (UDM).

Figure 11-1. Block Diagram of 6-Bit Up/Down Counter



Caution When using the 6-bit up/down counter, set the CI0/P03/INTP3 pin to input mode (by setting bit 3 (PM03) of port mode register 0 to 1).

Figure 11-2. Format of 6-Bit Up/Down Counter Control Register

Symbol	7	6	5	4	3	2	1	0	Address	At reset	R/W
UDM	0	0	0	UDM4	UDM3	UDM2	UDM1	UDM0	FFA8H	00H	R/W

UDM0	Valid edge selection
0	Falling edge of CI0
1	Rising edge of CI0

UDM1	Operation mode selection
0	Descending count operation
1	Ascending count operation

UDM2	Count operation control
0	Stops count operation (count value retained)
1	Enables count operation

UDM3	Counter clearing
0	Count operation
1	Clears counter

UDM4	Selection of setting signal for INTP3 flag
0	Set by INTP3 interrupt signal
1	Set by INTUD interrupt signal

Cautions 1. Do not set UDM0, UDM1, and UDM3 at the same time as the input of the valid edge of the CI0/P03/INTP3 pin.

2. When 1 is written into UDM3, the UDC is cleared to 0.
When the UDC is cleared, UDM3 is automatically reset to 0.

3. The UDC cannot be read or written until data is set in it after $\overline{\text{RESET}}$.

11.1 1-SECOND COUNTER

This section provides an example in which the 6-bit up/down counter generates an interrupt request every 1 second when an external frequency of 60 Hz is input to CI0. The interrupt processing increments or decrements a RAM counter (C_COUNT) by using a count direction flag (F_HOUKOU).

(1) Description of package

<Symbols declared as public>

- Subroutine name
  S_UPDOWN: Subroutine incrementing/decrementing counter
- Input parameter
  F_HOUKOU : Up/down count status
  DATAU : Data stored to compare register (frequency: 60 Hz)
- Output parameter
  C_COUNT : Stores counter value

<Register used>

None

<RAM used>

Name	Use	Attributes	Bytes
C_COUNT	RAM counter	SADDR	1

<Flags used>

Name	Use
F_HOUKOU	Count direction flag (counter counts down when this flag is set)

<Nesting>

2 levels, 5 bytes

<Hardware used>

- 6-bit up/down counter

<Initial settings>

- Set by subroutine S_UPDOWN
- INTUD interrupt enabled

<Usage>

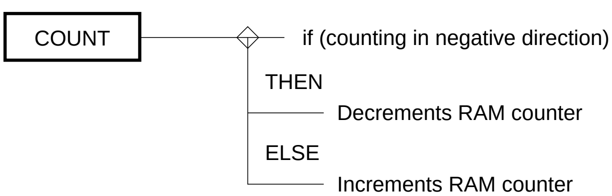
- Counting is started when the INTUD interrupt request is enabled.
- Call subroutine S_UPDOWN to change the count direction (between up and down).

(2) Example of use

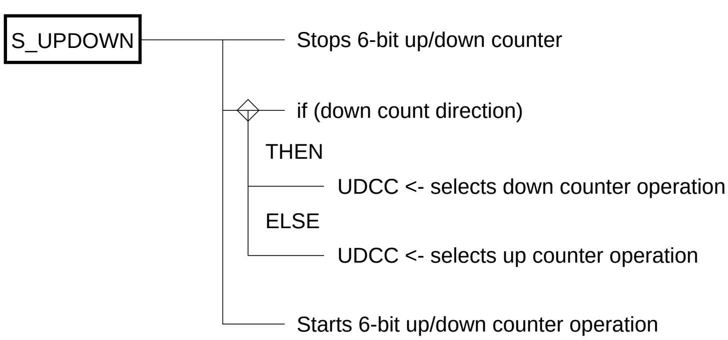
```
EXTRN   S_UPDOWN.DATAU
M2          CSEG                        ;
RES_STA:
    DI                                  ;
    UDC=#0                              ; Clears 6-bit up/down counter
    UDCC=#DATAU                         ; Sets value to compare register up
    UDM=#00011110B                      ; Set by INTUD interrupt signal.  Ascending count operation.
    CLR1    PIF3                        ; Clears INTP3 (INTUD) interrupt request flag
    CLR1    KSIF                        ; Clears interrupt request flag
    CLR1    PMK3                        ; Enables INTUD interrupt
    CLR1    KSMK                        ; Enables INTKS interrupt
    EI                                  ;

    if(up/down change)
        CALL        !S_UPDOWN
    endif
```

(3) SPD chart

[Count processing (INTUD interrupt processing)]



[Count direction change routine]



(4) Program list

```
;
;***********************************************
;        6-bit up/down counter (INTUD)
;***********************************************
$PC(044A)                                       ;
                                                ;
PUBLIC  C_COUNT,F_HOUKOU                        ;
PUBLIC  S_UPDOWNU                               ;
PUBLIC  DATAU                                   ;

                                                ;
VEINTUD CSEG    AT 0CH
        DW   COUNT
DATAU   EQU  60                                 ; 60 Hz cycle
;***********************************************
;               RAM definition
;***********************************************
             DSEG   SADDR                       ;
C_COUNT:     DS     1                           ; RAM counter
             BSEG                               ;
F_HOUKOU     DBIT                               ; Count direction flag
                                                ;
        CSEG                                    ;
COUNT:
        SEL  RB2                                ;
        if_bit(F_HOUKOU)                        ; Count direction flag = 1?
             C_COUNT--                          ; yes -> decrements RAM counter
        else                                    ;
             C_COUNT++                          ; no -> increments RAM counter
        endif                                   ;
        RETI                                    ;
;
;***********************************************
;        RAM counter up/down subroutine
;***********************************************
S_UPDOWN:
        CLR1    UDM.2                           ; Stops count operation
        if_bit(F_HOUKOU)                        ;
             CLR1   UDM.1                       ; Down counter operation
        else                                    ;
             SET1   UDM.1                       ; Up counter operation
        endif                                   ;
        SET1    UDM.2                           ; Starts count operation
        RET                                     ;
    END
```

[MEMO]

APPENDIX A SPD CHART DESCRIPTION

SPD is an acronym derived from Structured Programming Diagrams.

“Structured” means logical design and organization using basic logical structures, and involves structuring the logical processes of a program.

All programs can be created by only combining basic logical structures (sequencing, selection, repetition). (This is called the structured theorem.) Thus, the program flow is clarified by its structure and reliability improves. There are a variety of ways to represent program structure; however, a graphical technique called SPD is used at NEC.

Below, the SPD symbols used in the SPD technique are described and compared to flowchart symbols.

Table A-1. Comparison of SPD Symbols and Flowcharts (1/2)

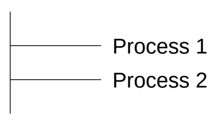
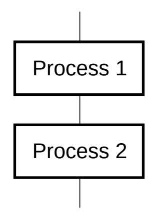
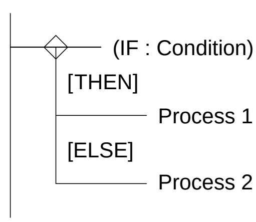
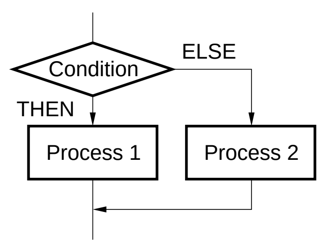
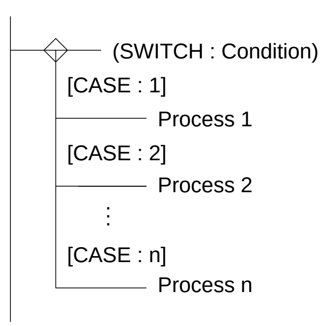
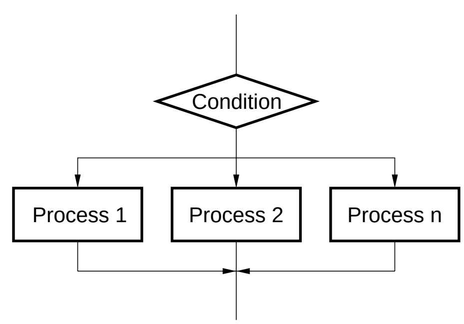
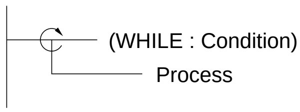
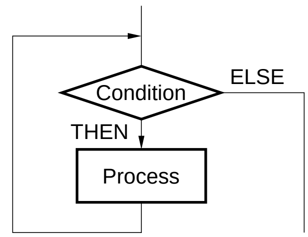
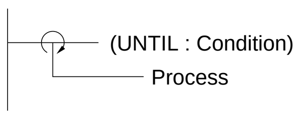
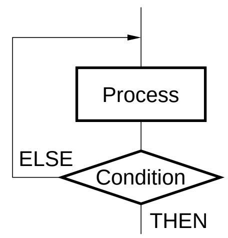
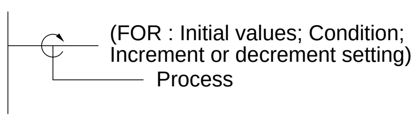
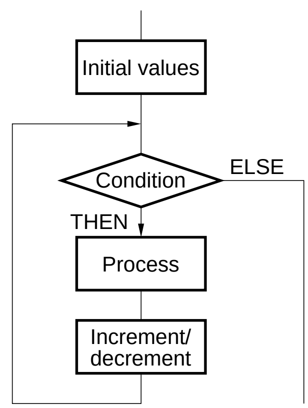
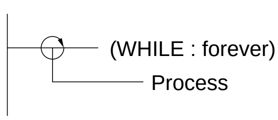
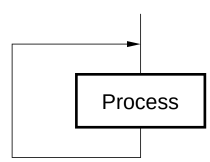
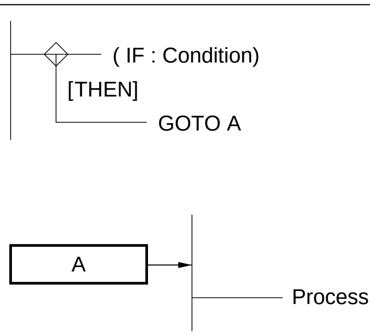
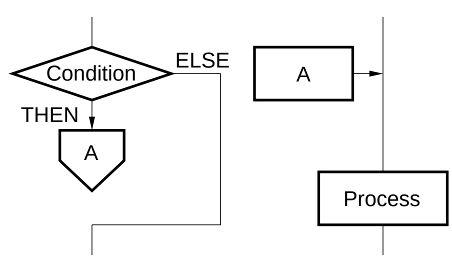
Process name	SPD symbol	Flowchart symbol
Sequential processing		
Conditional branch (IF)		
Conditional branch (SWITCH)		

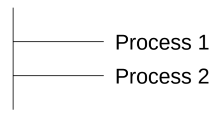
Table A-1. Comparison of SPD Symbols and Flowcharts (2/2)

Process name	SPD symbol	Flowchart symbol
Conditional loop (WHILE)		
Conditional loop (UNTIL)		
Conditional loop (FOR)		
Infinite loop		
Connector		

1. Sequential Processing

Sequential processing is executed in the output order from the top to the bottom.

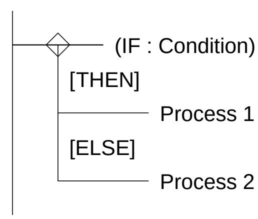
- **SPD chart**



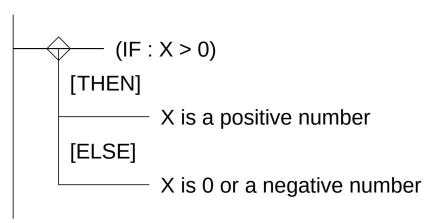
2. Conditional Branch: 2 Branches (IF)

The processing content is selected based on whether the condition specified in IF is true or false (THEN/ELSE).

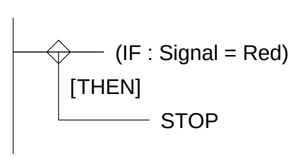
- **SPD chart**



Examples1. Determine whether X is positive or negative.



2. If the signal is red, stop.

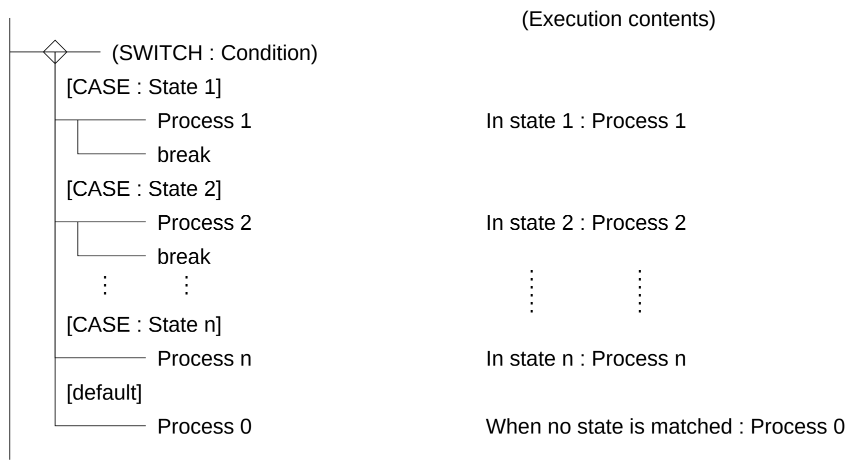


3. Conditional Branch: Multiple Branches (SWITCH)

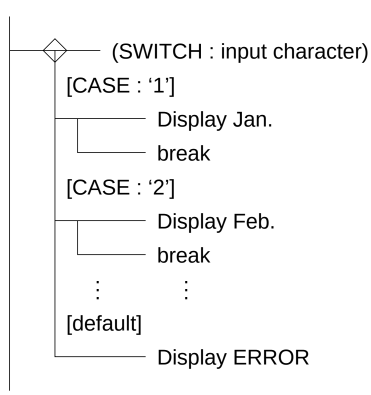
The condition specified by SWITCH is compared to the states indicated by CASE and the processing is selected. The two types of processing for a SWITCH statement are the case where only processing in the matched state is executed and the case where processing starts at the matched state and continues on below it. (When processing is not continued, ‘break’ is written.) Also, when no condition is matched, ‘default’ processing is executed (specifying ‘default’ is optional).

(1) For only the matched state

- **SPD chart**

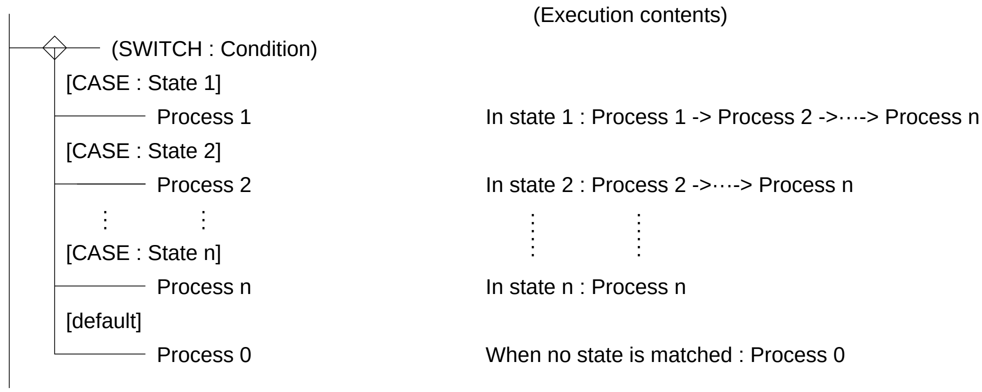


Example Display the name of a month by entering a character.

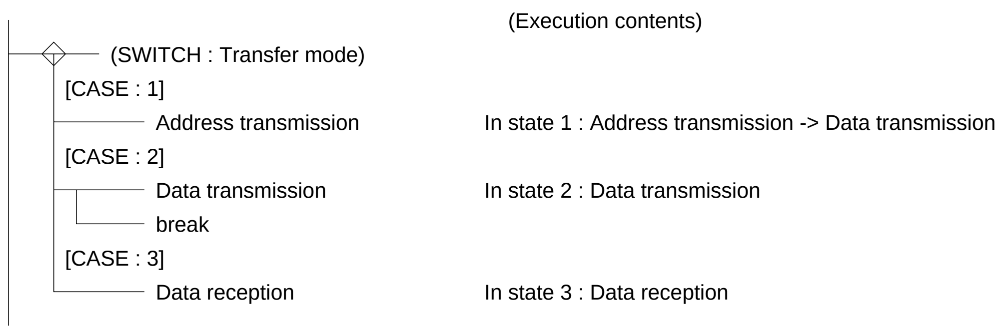


(2) For processing beginning at the matched state

- **SPD chart**



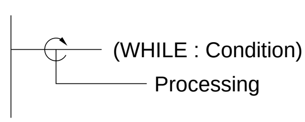
Example Communication through a serial interface



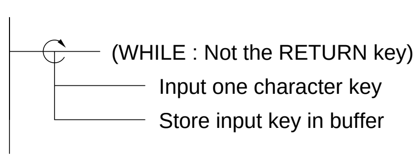
4. Conditional Loop (WHILE)

The condition specified in WHILE is evaluated. The processing is repeatedly executed as long as the condition holds. (When the condition does not hold from the beginning, nothing is executed.)

- **SPD chart**



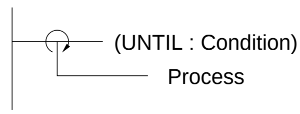
Example The keys are buffered until the RETURN key is input.



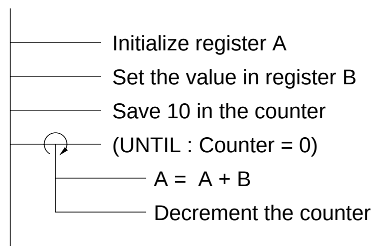
5. Conditional Loop (UNTIL)

The condition specified in UNTIL is evaluated after the process. The process is repeatedly executed until the condition holds. (Even when the condition does not hold at the beginning, the process is executed once.)

- **SPD chart**



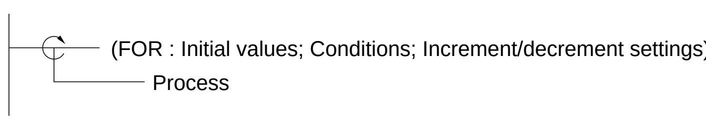
Example The value in register B is multiplied by 10 and saved in register A.



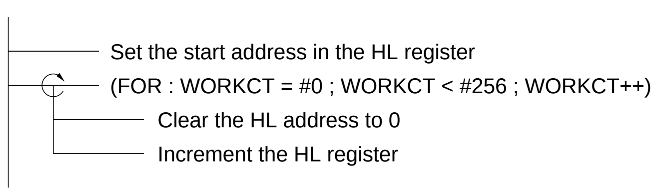
6. Conditional Loop (FOR)

The process is repeatedly executed until the parameter conditions specified in FOR hold.

- **SPD chart**



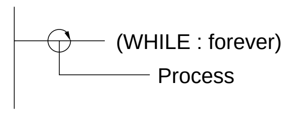
Example Beginning at the HL address, clear 256 bytes to 0.



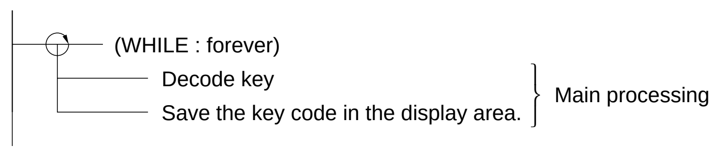
7. Infinite Loop

By specifying ‘forever’ as the WHILE condition, the process is repeatedly executed forever.

- **SPD chart**



Example Repeatedly execute the main processing.

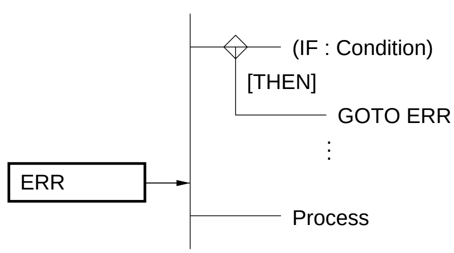


8. Connector (GOTO)

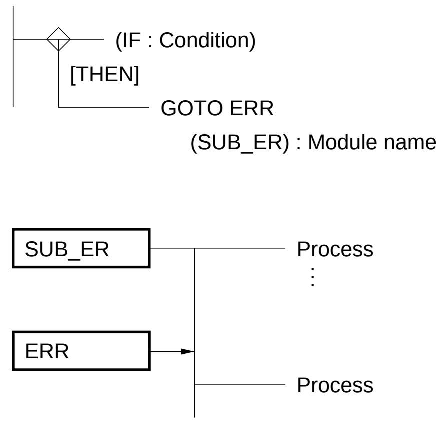
The specified address is unconditionally branched to.

- **SPD chart**

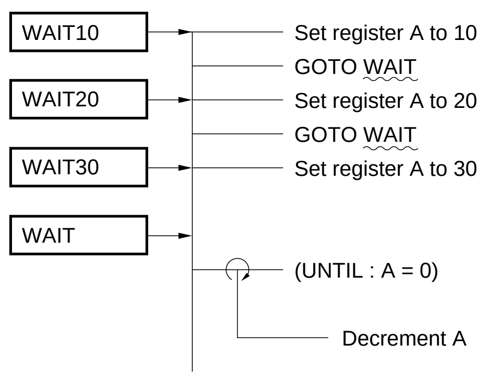
(1) Branch to the same module



(2) Branch to different modules



Example At the starting address of the subroutine, the parameter is selected and a wait is set.



9. Connector (Continue)

When one SPD module lasts multiple pages, the processing flow is shown below.

- **SPD chart**

APPENDIX B REVISION HISTORY

The complete revision history is shown below. The applicable places are shown for the chapter in each edition.

Edition no.	Major revisions from the previous edition	Applicable chapters
Version 2	The following chapters and sections have been added: **Sections 2.1** to **2.6**, **Chapters 3** to **7**, **Sections 8.1** to **8.4** and **9.1** to **9.3**	Throughout
	The following subseries have been added as applicable products: µPD78024, µPD78044A, and µPD780208	
	The following subseries are no longer applicable products: µPD78002, µPD78002Y, µPD78014, and µPD78014Y subseries (These subseries are described in Basics (I).) µPD78044 Subseries µPD78054 and µPD78064 subseries (These subseries are described in Basics (III).)	
	"Configuration of 12-Digit FIP Display and Key Input" has been changed.	Chapter 10
Version 3	The following products have been added as applicable products: µPD78044F, µPD78044H, and µPD780228 subseries, µPD780206, and µPD780208	Throughout
	The following subseries have been dropped as applicable products: µPD78024 and µPD78044A subseries	
	The following subseries have been added in Section 1.1. µPD78075B, µPD78075BY, µPD780018, µPD780018Y, µPD780058, µPD780058Y, µPD78058F, µPD78058FY, µPD780034, µPD780034Y, µPD780024, µPD780024Y, µPD78014H, µPD780964, µPD780924, µPD780228, µPD78044H, µPD78044F, µPD780308, µPD780308Y, µPD78064B, µPD78098B, µPD780973, and µPD780805 subseries, and µPD78P0914	Chapter 1
	Table 3-3 has been added.	Chapter 3
	Note 2 and Caution 2 have been added to Figure 4-2.	Chapter 4
	Figure 4-4 has been added.	
	A Caution has been added to Figure 5-5.	Chapter 5
	Table 8-2 has been added.	Chapter 8
	Note 4 and a Caution have been added to Figure 8-3.	
	A Caution has been added to Figure 8-9.	
	Section 8.1 The µPD6252 has been defined as a product provided for maintenance purposes only.	
	Figure 9-4 has been added.	Chapter 9

[MEMO]

NEC

Facsimile Message

From:

Name

Company

Tel. FAX

Address

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.

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
Technical Documentation Dept.
Fax: +49-211-6503-274

South America
NEC do Brasil S.A.
Fax: +55-11-889-1689

Hong Kong, Philippines, Oceania
NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

Korea
NEC Electronics Hong Kong Ltd.
Seoul Branch
Fax: 02-528-4411

Taiwan
NEC Electronics Taiwan Ltd.
Fax: 02-719-5951

Asian Nations except Philippines
NEC Electronics Singapore Pte. Ltd.
Fax: +65-250-3583

Japan
NEC Corporation
Semiconductor Solution Engineering Division
Technical Information Support Dept.
Fax: 044-548-7900

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	❏	❏	❏	❏