NEC

Preliminary User's Manual

µPD1615A(A), µPD1615B(A), µPD1615F(A), µPD1616F(A), µPD16F15A

8-bit Single-Chip Microcontroller

Hardware

MS-DOS and MS-Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
PC/AT and PC DOS are trademarks of IBM Corp.

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

The export of this product from Japan is regulated by the Japanese government. To export this product may be prohibited without governmental license, the need for which must be judged by the customer. The export or re-export of this product from a country other than Japan may also be prohibited without a license from that country. Please call an NEC sales representative.

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.l.
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

Introduction

Readers

   This manual has been prepared for user engineers who want to understand the functions of the µPD1615A subseries and design and develop its application systems and programs.

   µPD1615A Subseries: µPD1615A, µPD1615B, µPD1615F, µPD16F15A, µPD1616F.

Purpose

   This manual is intended for users to understand the functions described in the Organization below.

Organization

   The µPD1615A subseries manual is separated into two parts: this manual and the instruction edition (common to the 78K/0 series).

µPD1615A **subseries** **This Manual**	**78K/0 series** **User's Manual** **Instruction**

   • Pin functions
   • Internal block functions
   • Interrupt
   • Other on-chip peripheral functions

   • CPU functions
   • Instruction set
   • Explanation of each instruction

How to Read This Manual

   Before reading this manual, you should have general knowledge of electric and logic circuits and microcontrollers.

   • When you want to understand the function in general:
     → Read this manual in the order of the contents.
   • How to interpret the register format:
     → For the bit number enclosed in square, the bit name is defined as a reserved word in the assembler and the compiler.
   • To make sure the details of the registers when you know the register name.
     → Refer to **Appendix C.**

Chapter Organization

   This manual devides the descriptions for the subseries into different chapters as shown below. Read only the chapters related to the device you use.

	Chapter	µPD1615A	µPD1615B	µPD1615F	µPD16F15A	µPD1616F
Chapter 1	Outline	O	O	O	O	O
Chapter 2	Pin Function	O	O	O	O	O
Chapter 3	CPU Architecture	O	O	O	O	O
Chapter 4	Port Functions	O	O	O	O	O
Chapter 5	Clock Generator	O	O	O	O	O
Chapter 6	16-Bit Timer/Counter	O	O	O	O	O
Chapter 7	8-Bit Timer/Event Counters 50, 51	O	O	O	O	O
Chapter 8	Watch Timer	O	O	O	O	O
Chapter 9	Watchdog Timer	O	O	O	O	O
Chapter 10	Clock Output Control Circuit	O	O	O	O	O
Chapter 11	A/D-Converter	O	O	O	O	O
Chapter 12	Serial Interface Outline	O	O	O	O	O
Chapter 13	Serial Interface Channel 3	O	O	O	O	O
Chapter 14	Serial Interface UART	O	O	O	O	O
Chapter 15	VAN Controller	O	O	O	O	O
Chapter 16	LCD Controller/Driver	O	O	O	O	—
Chapter 17	Sound Generator	O	O	O	O	O
Chapter 18	Interrupt Functions	O	O	O	O	O
Chapter 19	Standby Function	O	O	O	O	O
Chapter 20	Reset Function	O	O	O	O	O
Chapter 21	µPD16F15A	O	O	O	O	O
Chapter 22	Instruction Set	O	O	O	O	O
Appendix A	Development Tools	O	O	O	O	O
Appendix B	Embedded Software	O	O	O	O	O
Appendix C	Register	O	O	O	O	O
Appendix D	Revision History	O	O	O	O	O

Related Documents

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

- **Related documents for μPD1615 subseries**

Document name	Document No.	
	Japanese	English
μPD1615A Preliminary Product Information	-	U13723E
μPD16F15A Preliminary Product Information	-	U13606E
μPD1615A Subseries User s Manual	-	This manual
78K/0 Series User s Manual-Instruction	IEU-849	IEU-1372
78K/0 Series Instruction Table	U10903J	-
78K/0 Series Instruction Set	U10904J	U12326E
μPD1615A Subseries Special Function Register Table	-	-

- **Related documents for development tool (User's Manuals)**

Document name		Document No.	
		Japanese	English
RA78K Series Assembler Package	Operation	EEU-809	EEU-1399
	Language	EEU-815	EEU-1404
RA78K Series Structured Assembler Preprocessor		EEU-817	EEU-1402
CC78K Series C Compiler	Operation	EEU-656	EEU-1280
	Language	EEU-655	EEU-1284
CC78K/0 C Compiler	Operation	U11517J	-
	Language	U11518J	-
CC78K/0 C Compiler Application Note	Programming Note	EEA-618	EEA-1208
CC78K Series Library Source File		EEU-777	-
IE-78K0-NS-A		U10057J	U10057E
IE-78K0-NS-P04		-	U13359E
IE-1615-NS-EM4			
NP-80GC-TQ		-	-
SM78K0 System Simulator Windows Base	Reference	U10181J	U10092E
SM78K0 Series System Simulator	External part user open Interface	U10092J	-
IBM PC/AT (DC DOS) Base		-	U14379E

• **Related documents for embedded software (User's Manual)**

Document name		Document No.	
		Japanese	English
78K/0 Series Real-Time OS	Basics	U11537J	-
	Installation	U11536J	-
	Technicals	U11538J	-
78K/0 Series OS MX78K0	Basics	EEU-5010	-
Fuzzy Knowledge Data Creation Tool		EEU-829	EEU1438
78K/0, 78IK/II, 87AD Series Fuzzy Inference Development Suppport System-Translator		EEU-862	EEU-1444
78K/0 Series Fuzzy Inference Development Suppport System- Fuzzy Inference Module		EEU-858	EEU-1441
78K/0 Series Fuzzy Inference Development Suppport System- Fuzzy Inference Debugger		EEU-921	EEU-1458

• **Other Documents**

Document name	Document No.	
	Japanese	English
IC Package Manual	C10943X	
Semiconductor Device Mounting Technology Manual	C10535J	C10535E
Quality Grade on NEC Semiconductor Devices	C11531J	C11531E
Reliability Quality Control on NEC Semiconductor Devices	C10983J	C10983E
Electric Static Discharge (ESD) Test	MEM-539	-
Semiconductor Devices Quality Assurance Guide	MEI-603	MEI-1202
Microcontroller Related Product Guide - Third Party Manufacturers	U11416J	-

Caution: **The above documents are subject to change without prior notice. Be sure to use the latest version document when starting design.**

Table of Contents

List of Figures

List of Tables

Table 16-1: Maximum Number of Display Pixels 270
Table 16-2: LCD Controller/Driver Configuration 271
Table 16-3: Frame Frequencies (Hz) 274
Table 16-4:   COM Signals 276
Table 16-5: LCD Drive Voltages 277
Table 16-6: LCD Drive Voltages (with On-Chip Split Resistor)connected externally 280
Table 16-7: Selection and Non-Selection Voltages (COM0) 283
Table 16-8: Selection and Non-Selection Voltages (COM0, COM1) 286
Table 16-9: Selection and Non-Selection Voltages (COM0 to COM2) 289
Table 16-10: Selection and Non-Selection Voltages (COM0 to COM3) 293

Table 17-1: Sound Generator Configuration 298

Table 18-1: Interrupt Source List 307
Table 18-2: Various Flags Corresponding to Interrupt Request Sources 310
Table 18-3: Times from Maskable Interrupt Request Generation to Interrupt Service 319
Table 18-4: Interrupt Request Enabled for Multiple Interrupt during Interrupt Servicing 322

Table 19-1: HALT Mode Operating Status 329
Table 19-2: Operation after HALT Mode Release 331
Table 19-3: STOP Mode Operating Status 332
Table 19-4: Operation after STOP Mode Release 334

Table 20-1: Hardware Status after Reset (1/2) 338
Table 20-1: Hardware Status after Reset (2/2) 339

Table 21-1: Differences among µPD16F15A and Mask ROM Versions 341
Table 21-2: Values of the Memory Size Switching Register for the Different Devices 342
Table 21-3: Examples of internal Extension RAM Size Switching Register Settings 343
Table 21-4: Transmission Method List 344
Table 21-5: Main Functions of Flash Memory Programming 345

Table 22-1: Operand Identifiers and Description Methods 349

[MEMO]

Chapter 1 Outline (µPD1615A Subseries)

1.1 Features

• Internal memory

Table 1-1: Internal high capacity ROM and RAM

Item Part Number	Program Memory (ROM)	Data Memory				Package
		Internal High-Speed RAM	LCD Display RAM	Internal Expansion RAM	VAN	
µPD1615A(A)	60 K bytes	1024 bytes	40 bytes	1024 bytes	256 bytes	80-pin plastic QFP (fine pitch)
µPD1615B(A)	48 K bytes	1024 bytes	40 bytes	512 bytes	256 bytes	80-pin plastic QFP (fine pitch)
µPD1615F(A)	32 K bytes	768 bytes	40 bytes	512 bytes	256 bytes	80-pin plastic QFP (fine pitch)
µPD16F15A	60 K bytes	1024 bytes	40 bytes	1024 bytes	256 bytes	80-pin plastic QFP (fine pitch)
µPD1616F(A)	32 K bytes	768 bytes	-	512 bytes	256 bytes	80-pin plastic QFP (fine pitch)

• Instruction execution time can be changed from high speed (0.25 µs) to ultra low speed
• I/O ports: 57
• 8-bit resolution A/D converter : 4 channels
• Sound generator
• LCD-controller/driver

• VAN-Interface
• Serial interface : 2 channels
  • 3-wire mode : 1 channel
  • UART mode : 1 channel
• Timer   : 5 channels
• Supply voltage : V_{DD} = 4.0 to 5.5 V

1.2 Application

Multifunction display, steering controller, climate controller etc.

1.3 Ordering Information

Part Number   Package

µPD1615AGC(A) - xxx - 8BT 80-pin plastic QFP (14 x 14 mm, resin thickness 1.4 mm)
µPD1615BGC(A) - xxx - 8BT 80-pin plastic QFP (14 x 14 mm, resin thickness 1.4 mm)
µPD1615FGC(A) - xxx - 8BT 80-pin plastic QFP (14 x 14 mm, resin thickness 1.4 mm)
µPD1616FGC(A) - xxx - 8BT 80-pin plastic QFP (14 x 14 mm, resin thickness 1.4 mm)
µPD16F15AGC - 8BT   80-pin plastic QFP (14 x 14 mm, resin thickness 1.4 mm)

1.4 Pin Configuration (Top View)

80-pin plastic QFP (14 x 14 mm)

µPD1615AGC(A) - xxx - 8BT
µPD1615BGC(A) - xxx - 8BT
µPD1615FGC(A) - xxx - 8BT
µPD1616FGC(A) - xxx - 8BT
µPD16F15AGC - 8BT

Figure 1-1: Pin Configuration µPD1615A, µPD1615B, µPD1615F, µPD16F15A

Top pins (80–61):
P13/ANI3 (80), AVss (79), P127/SI3/S0 (78), P126/SO3/S1 (77), P125/SCK3/S2 (76), P124/TxD0/S3 (75), P123/RxD0/S4 (74), P122/TIO1/S5 (73), P121/TI00/TO0/S6 (72), P120/PCL/S7 (71), P117/S8 (70), P116/S9 (69), P115/S10 (68), P114/S11 (67), P113/S12 (66), P112/S13 (65), P111/S14 (64), P110/S15 (63), P107/S16 (62), P106/S17 (61)

Left pins (1–20):
P12/ANI2 (1), P11/ANI1 (2), P10/ANI0 (3), AVDD/AVREF (4), RESET (5), X1 (6), X2 (7), VPP/IC (8), CL1 (9), CL2 (10), VSS0 (11), VDD0 (12), P00/INTP0 (13), P01/INTP1 (14), P02/INTP2 (15), P06/TI50/TO50 (16), P07/TI51/TO51 (17), Rx1VAN (18), Rx2VAN (19), Rx0VAN (20)

Right pins (60–41):
P105/S18 (60), P104/S19 (59), P103/S20 (58), P102/S21 (57), P101/S22 (56), P100/S23 (55), P97/S24 (54), P96/S25 (53), P95/S26 (52), P94/S27 (51), P93/S28 (50), P92/S29 (49), P91/S30 (48), P90/S31 (47), P87/S32 (46), P86/S33 (45), P85/S34 (44), P84/S35 (43), P83/S36 (42), P82/S37 (41)

Bottom pins (21–40):
TxVAN (21), P47/SGO/SGOF (22), P46/SGOA (23), P45 (24), P44 (25), P43 (26), P42 (27), P41 (28), P40 (29), VSS1 (30), VDD1 (31), VLC0 (32), VLC1 (33), VLC2 (34), COM0 (35), COM1 (36), COM2 (37), COM3 (38), P80/S39 (39), P81/S38 (40)

Cautions: 1. Connect IC (internally connected) pin directly to Vss.

2. AVDD pin should be connected to VDD.

3. AVss pin should be connected to Vss.

27

Figure 1-2: Pin Configuration µPD1616F
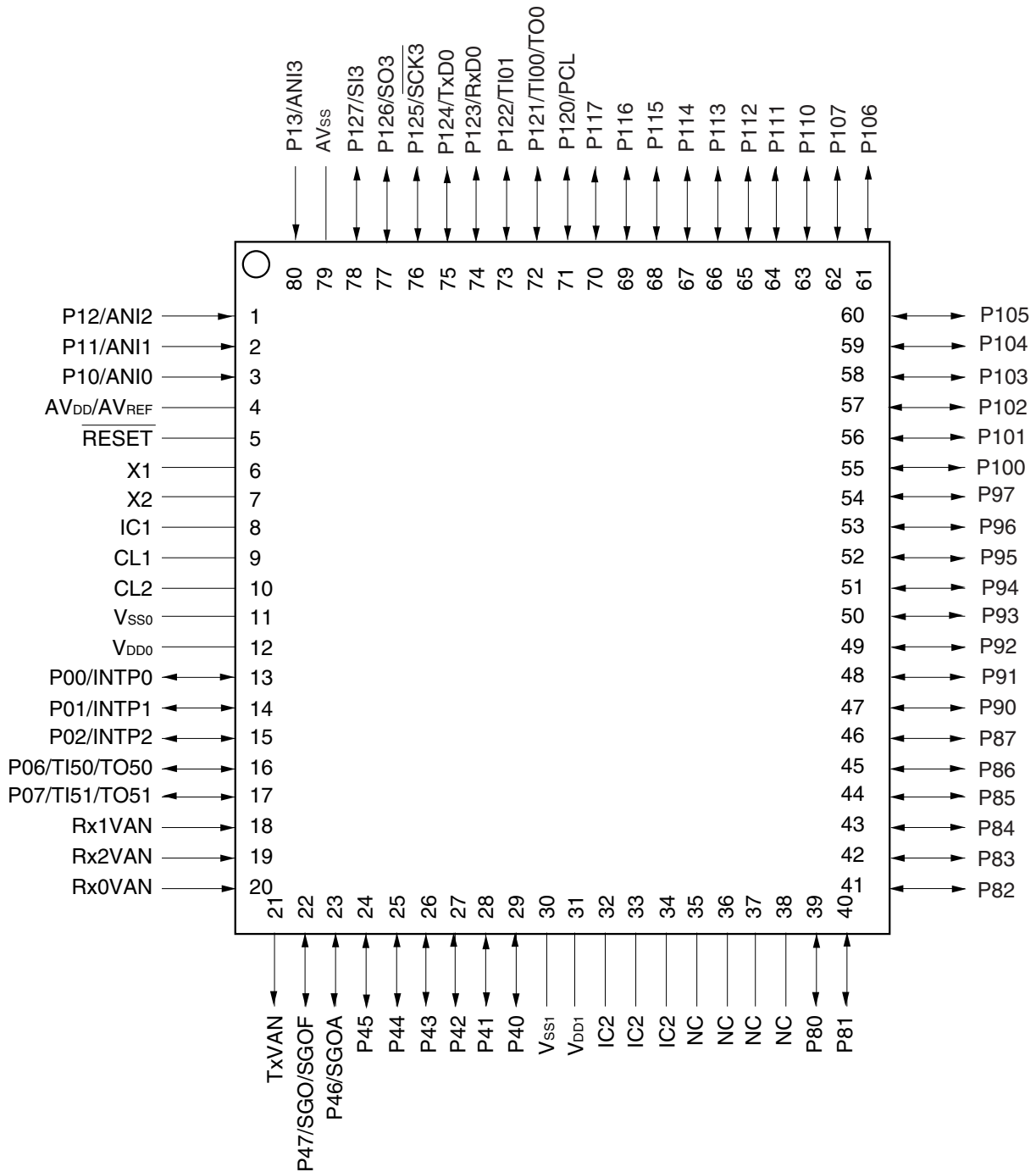
Cautions: 1. Connect IC1 (internally connected) pin directly to VSS.
2. Connect IC2 (internally connected) pin directly to VDD.
3. AVDD pin should be connected to VDD.
4. AVSS pin should be connected to VSS.
5. NC pins are not connected.

Pin Identifications

P00 to P02, P06, P07	: Port0	RxD0	: Receive Data
P10 to P13	: Port1	TxD0	: Transmit Data
P40 to P47	: Port4	SGO	: Sound Generator Output
P80 to P87	: Port8	SGOA	: Sound Generator Amplitude
P90 to P97	: Port9	SGOF	: Sound Generator Frequency
P100 to P107	: Port10	PCL	: Programmable Clock Output
P110 to P117	: Port11	S0 to S39	: Segment Output
P120 to P127	: Port12	COM0 to COM3	: Common Output
INTP0 to INTP2	: Interrupt External	X1, X2	: Crystal (Main System Clock)
TI00, TI01, TI50, TI51	: Timer Input	CL1, CL2	: RC (Subsystem Clock)
TO0 , TO51, TO52	: Timer Output	RESET	: Reset
Rx0VAN	: VAN Receive Data	ANI0 to ANI3	: Analog Input
Rx1VAN	: VAN Receive Data	AV_{SS}	: Analog Ground
Rx2VAN	: VAN Receive Data	AV_{DD}/AV_{REF}	: Analog Power Supply and
TxVAN	: VAN Transmit Data		Reference Voltage
SI3	: Serial Input	V_{PP}	: Programming Power supply
SO3	: Serial Output	V_{SS}	: Ground
SCK3	: Serial Clock	IC, IC1, IC2	: Internally Connected
		NC	: Not Connected

1.5 78K/0 Series Development

These products are a further development in the 78K/0 Series. The designations appearing inside the boxes are subseries names.

Products in mass production

Products under development

Y subseries products are compatible with I²C bus.

For control

100-pin	μPD78078	μPD78078Y	Timer added to the μPD78054, external interface functions enhanced
100-pin	μPD78070A	μPD78070AY	ROM-less product for the μPD78078
80-pin	μPD78054	μPD78054Y	UART and D/A added to the μPD78014, I/O enhanced
64-pin	μPD78018F	μPD78018FY	Low-voltage (1.8 V) operation version of the μPD78014, ROM and RAM variations enhanced
64-pin	μPD78014	μPD78014Y	A/D and 16-bit timer added to the μPD78002
64-pin	μPD780001		A/D added to the μPD78002
64-pin	μPD78002	μPD78002Y	Basic subseries for control
42/44-pin	μPD78083		Internal UART, low-voltage (1.8 V) operation possible

For FIP driving

100-pin	μPD780208	I/O, FIP C/D of the μPD78044A enhanecd, display output total: 53
80-pin	μPD78044A	6-bit U/D counter added to the μPD78024, display output total: 34
64-pin	μPD78024	Basic subseries for FIP driving, display output total: 26

For LCD driving

100-pin	μPD78064	μPD78064Y	Subseries for LCD driving, internal UART
	μPD780308	μPD780308Y	

For IEBus™

80-pin	μPD78098	IEBus controller added to the μPD78054

100-pin	μPD780948	
80-pin	μPD780828B	CANBus controller
64-pin	μPD780816	

78K/0 Series

1.6 Block Diagram

Figure 1-3: Block Diagram µPD1615A, µPD1615B, µPD1615F, µPD16F15A
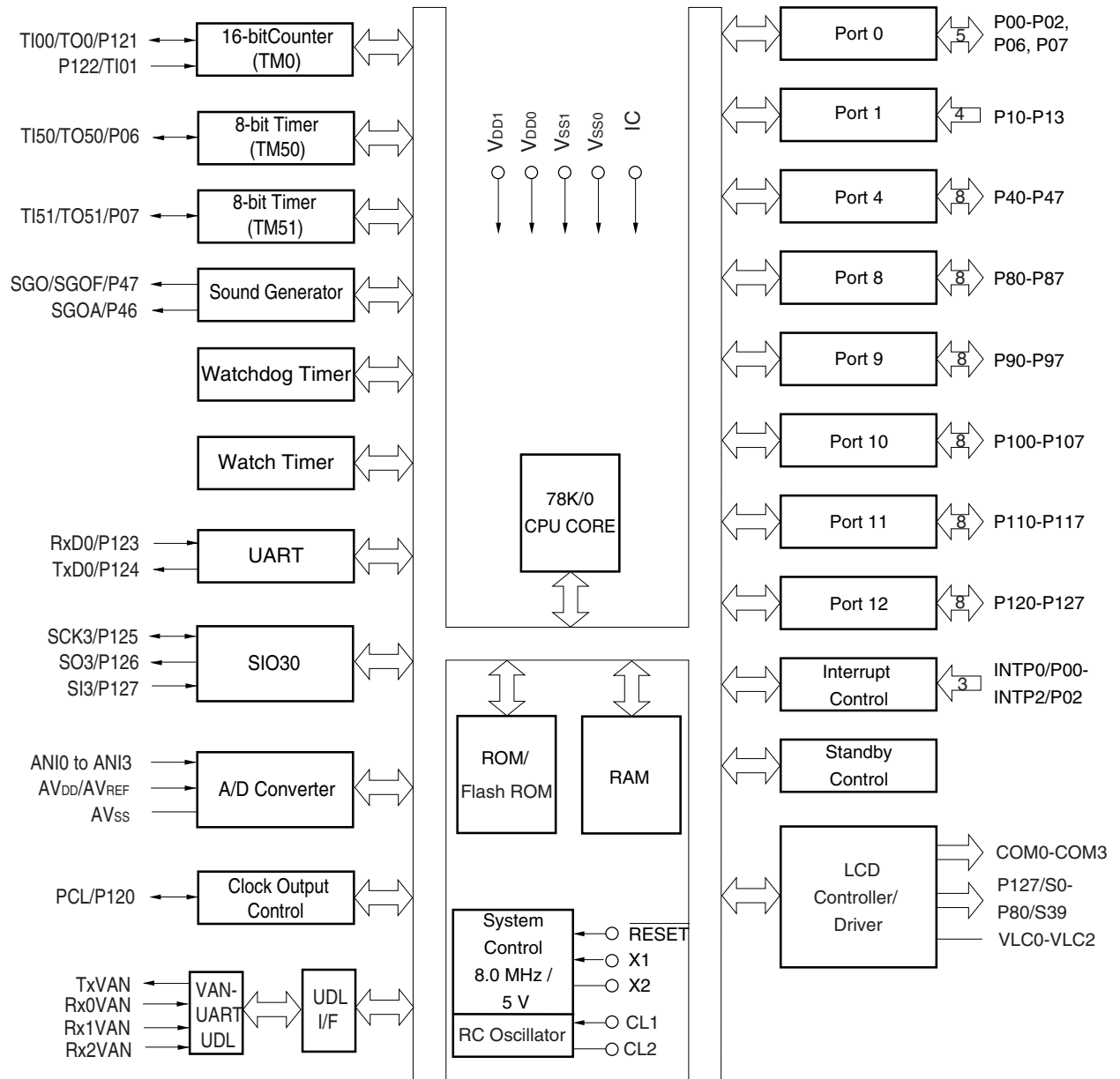
Remark: The internal ROM and RAM capacity depends on the product.

Figure 1-4: Block Diagram µPD1616F
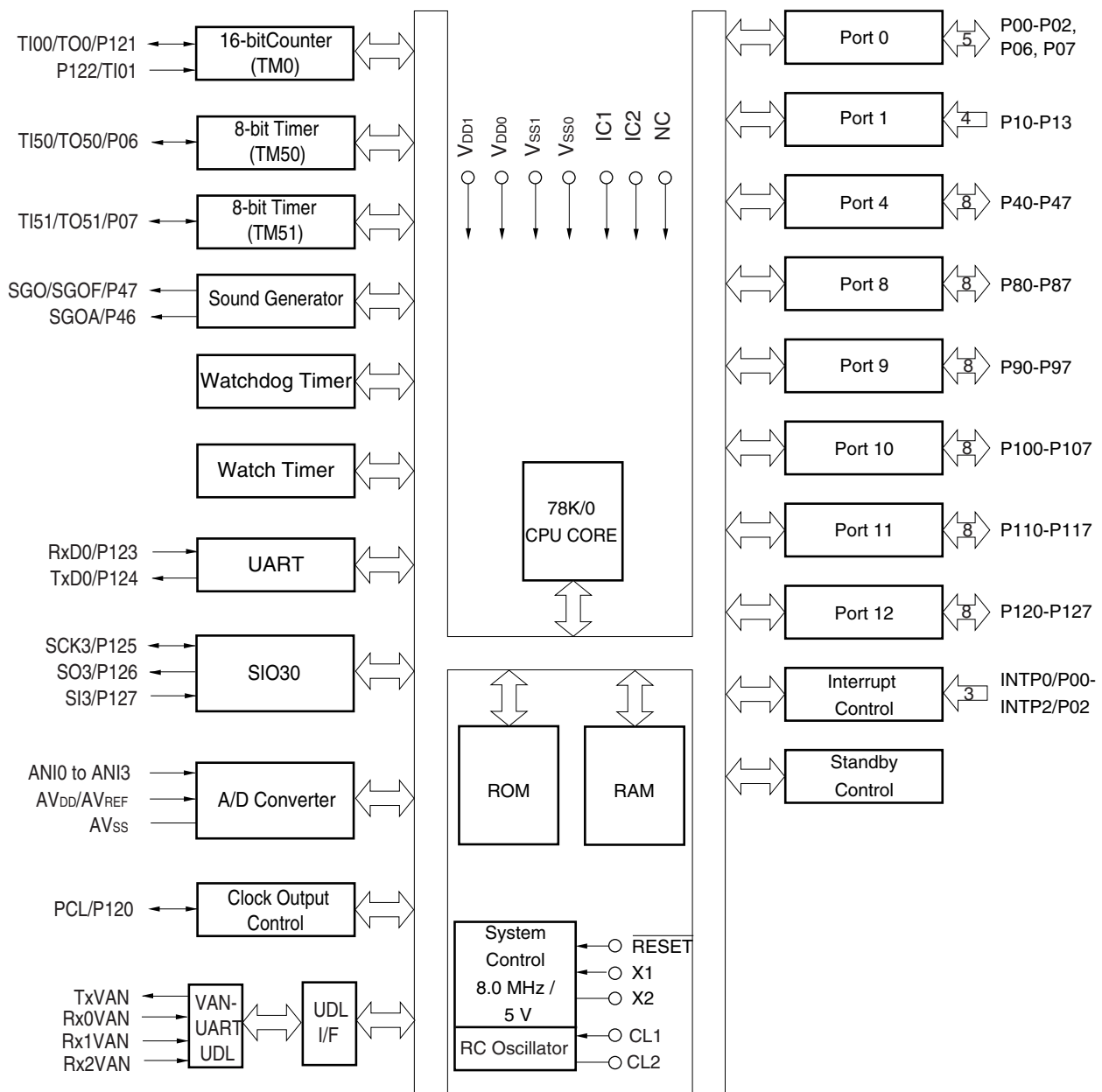
1.7 Overview of Functions

Item	Part Number	µPD1615A(A)	µPD1615B(A)	µPD1615F(A)	µPD1616F(A)	µPD16F15A
Internal memory	ROM	60 Kbytes	48 Kbytes	32 Kbytes	32 Kbytes	60 Kbytes
	Internal high-speed RAM	1024 bytes	1024 bytes	768 bytes	768 bytes	1024 bytes
	LCD Display RAM	40 bytes	40 bytes	40 bytes	-	40 bytes
	Internal Expansion RAM	1024 bytes	512 bytes	512 bytes	512 bytes	1024 bytes
Memory space		64 Kbytes				
General registers		8 bits x 32 registers (8 bits x 8 registers x 4 banks)				
Instruction cycle		On-chip instruction execution time selective function				
	When main system clock selected	0.25 µs/0.5 µs/1 µs/2 µs/4 µs (at 8 MHz)				
	When subsystem clock selected	122 µs (at 32.768 KHz)				
Instruction set		• 16-bit operation • Multiplication/division (8 bits x 8 bits, 16 bits/8 bits) • Bit manipulation (set, reset, test, boolean operation) • BCD adjustment, etc.				
I/O ports		Total : 57 • CMOS input : 4 • CMOS I/O : 53				
A/D converter		• 8 bit resolution x 4 channels				
Serial Interface		• 3-wire mode : 1 channel • UART mode : 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 output		2 (8-bit PWM output x 2)				
Clock output		62.5 KHz, 125 KHz, 250 KHz, 500 KHz, 1 MHz, 2 MHz, 4 MHz, 8 MHz (at main system clock of 8.0 MHz)				
Sound Generator		1 channel (as separate or composed output)				
LCD Controller/Driver		40 seg x 4 COM				
VAN		1 channel				
Vectored interrupts	Maskable interrupts	Internal : 15 External : 3				
	Non-maskable interrupts	Internal : 1				
	Software interrupts	Internal : 1				
Supply voltage		VDD = 4.0 V to 5.5 V				
Package		80-pin plastic QFP (14 mm x 14 mm)				

1.8 Mask Options

There are no mask options provided.

1.9 Differences between Flash and Mask ROM version

The differences between the two versions are shown in the table below. Differences of the electrical specification are given in the data sheet.

Table 1-2: Differences between Flash and Mask ROM version

	Flash Version	Mask ROM Version
ROM	Flash EEPROM	Mask ROM
V_PP Pin	Yes	None (IC pin)

[Memo]

Chapter 2 Pin Function (μPD1615A(A) Subseries)

2.1 Pin Function List

Normal Operating Mode Pins / Pin Input/Output Types

Table 2-1-1: Pin Input/Output Types μPD1615A(A), μPD1615B(A), μPD1615F(A), μPD16F15A

Input / Output	Pin Name	Function	Alternate Function	After Reset
Input / Output	P00	Port 0 5 bit input / output port Input / output mode can be specified bit-wise	INTP0	Input
	P01		INTP1	Input
	P02		INTP2	Input
	P06		TI50/TO50	Input
	P07		TI51/TO51	Input
Input	P10-P13	Port 1 4 bit input port Input mode can be specified bit-wise	ANI0-ANI3	Input
Input / Output	P40	Port 4 8 bit input/output port Input / output mode can be specified bit-wise	-	Input
	P41		-	Input
	P42		-	Input
	P43		-	Input
	P44		-	Input
	P45		-	Input
	P46		SG0A	Input
	P47		SG0/SG0F	Input
Input/ Output	P80-P87	Port 8 8 bit input / output port Input / output mode can be specified bit-wise This port can be used as segment signal output port or an I/O port in 1-bit units by setting port function register	S39 - S32	Input
Input/ Output	P90-P97	Port 9 8 bit input / output port Input / output mode can be specified bit-wise This port can be used as segment signal output port or an I/O port in 1-bit units by setting port function register	S31 - S24	Input
Input/ Output	P100-P107	Port 10 8 bit input / output port Input / output mode can be specified bit-wise This port can be used as segment signal output port or an I/O port in 1-bit units by setting port function register	S23 - S16	Input
Input/ Output	P110-P117	Port 11 8 bit input / output port Input / output mode can be specified bit-wise This port can be used as segment signal output port or an I/O port in 1-bit units by setting port function register	S15 - S8	Input
Input/ Output	P120	Port 12 8 bit input / output port Input / output mode can be specified bit-wise This port can be used as segment signal output port or an I/O port in 1-bit units by setting port function register	PCL/S7	Input
	P121		TI00/TO0/S6	
	P122		TI01/S5	
	P123		RxD0/S4	
	P124		TxD0/S3	
	P125		SCK3/S2	
	P126		SO3/S1	
	P127		SI3/S0	

Table 2-1-2: Pin Input/Output Types µPD1616F(A)

Input /Output	Pin Name	Function	Alternate Function	After Reset
Input /Output	P00	Port 0 5 bit input / output port Input / output mode can be specified bit-wise	INTP0	Input
	P01		INTP1	Input
	P02		INTP2	Input
	P06		TI50/TO50	Input
	P07		TI51/TO51	Input
Input	P10-P13	Port 1 4 bit input port Input mode can be specified bit-wise	ANI0-ANI3	Input
Input /Output	P40	Port 4 8 bit input/output port Input / output mode can be specified bit-wise	-	Input
	P41		-	Input
	P42		-	Input
	P43		-	Input
	P44		-	Input
	P45		-	Input
	P46		SG0A	Input
	P47		SG0/SG0F	Input
Input/Output	P80-P87	Port 8 8 bit input / output port Input / output mode can be specified bit-wise This port can be used as segment signal output port or an I/O port in 1-bit units by setting port function register	-	Input
Input/Output	P90-P97	Port 9 8 bit input / output port Input / output mode can be specified bit-wise This port can be used as segment signal output port or an I/O port in 1-bit units by setting port function register	-	Input
Input/Output	P100-P107	Port 10 8 bit input / output port Input / output mode can be specified bit-wise This port can be used as segment signal output port or an I/O port in 1-bit units by setting port function register	-	Input
Input/Output	P110-P117	Port 11 8 bit input / output port Input / output mode can be specified bit-wise This port can be used as segment signal output port or an I/O port in 1-bit units by setting port function register	-	Input
Input/Output	P120	Port 12 8 bit input / output port Input / output mode can be specified bit-wise This port can be used as segment signal output port or an I/O port in 1-bit units by setting port function register	PCL	Input
	P121		TI00/TO0	
	P122		TI01	
	P123		RxD0	
	P124		TxD0	
	P125		SCK3	
	P126		SO3	
	P127		SI3	

2.2 Non-Port Pins

Table 2-2-1: Non-Port Pins µPD1615A(A), µPD1615B(A), µPD1615F(A), µPD16F15A

Pin Name	I/O	Function	After Reset	Alternate Function Pin
INTP0	Input	External interrupts with specifiable valid edges (rising edge, falling edge, both rising and falling edges)	Input	P00
INTP1				P01
INTP2				P02
SI3	Input	Serial interface serial data input	Input	P127/S0
SO3	Output	Serial interface serial data output	Input	P126/S1
SCK3	Input/ Output	Serial interface serial clock input / output	Input	P125/S2
RxD0	Input	Asynchronous serial interface data input	Input	P123/S4
TxD0	Output	Asynchronous serial interface data output	Input	P124/S3
Rx0VAN, Rx1VAN, Rx2VAN	Input	VAN serial data input	Input	-
TxVAN	Output	VAN serial data output	Output	-
TI00	Input	External count clock input to 16-bit timer (TM0)	Input	P121/TO0/S6
TI01				P122/S5
TI50		External count clock input to 8-bit timer (TM50)		P06/TO50
TI51		External count clock input to 8-bit timer (TM51)		P07/TO51
TO0	Output	16-bit timer output	Input	P121/TI00/S6
TO50		8-bit timer output (also used for PWM output)		P06/TI50
TO51		8-bit timer output (also used for PWM output)		P07/TI51
PCL	Output	Clock output	Input	P120/S7
S0 to S7	Output	Segment signal output of LCD controller / driver	Input	P127 to P120
S8 to S15				P117 to P110
S16 to S23				P107 to P100
S24 to S31				P97 to P90
S32 to S39				P87 to P80
COM0-COM3	Output	Common signal output of LCD controller/driver	Output	-
V_{LC0} to V_{LC2}	-	LCD drive voltage	-	-
SGO	Output	Sound generator output	Input	P47/SGOF
SGOA	Output	Sound generator amplitude output	Input	P46
SGOF	Output	Sound generator frequency output	Input	P47/SGO
ANI0 to ANI3	Input	A/D Converter analog input	Input	P10 P13
AV_{DD}/ AV_{REF}	-	A/D Converter reference voltage input and power supply	-	-
AV_{SS}	-	A/D Converter ground potential. Connect to V_{SS}.	-	-
RESET	Input	System reset input	-	-
X1	-	Connection for main system clock	-	-
X2	-	Connection for main system clock	-	-
CL1	Input	RC connection for subsystem clock	-	-
CL2	-	RC connection for subsystem clock	-	-
$V_{DD}1$, $V_{DD}2$	-	Positive power supply	-	-
$V_{SS}1$, $V_{SS}2$	-	Ground potential	-	-
IC	-	Internal connection. Connect directly to V_{SS}	-	-
V_{PP}	-	Programming voltage. Connect directly to V_{SS} except flash programming.	-	-

Table 2-2-2: Non-Port Pins µPD1616F(A)

Pin Name	I/O	Function	After Reset	Alternate Function Pin
INTP0	Input	External interrupts with specifiable valid edges (rising edge, falling edge, both rising and falling edges)	Input	P00
INTP1				P01
INTP2				P02
SI3	Input	Serial interface serial data input	Input	P127
SO3	Output	Serial interface serial data output	Input	P126
SCK3	Input/ Output	Serial interface serial clock input / output	Input	P125
RxD0	Input	Asynchronous serial interface data input	Input	P123
TxD0	Output	Asynchronous serial interface data output	Input	P124
Rx0VAN, Rx1VAN, Rx2VAN	Input	VAN serial data input	Input	-
TxVAN	Output	VAN serial data output	Output	-
TI00	Input	External count clock input to 16-bit timer (TM0)	Input	P121/TO0
TI01				P122
TI50		External count clock input to 8-bit timer (TM50)		P06/TO50
TI51		External count clock input to 8-bit timer (TM51)		P07/TO51
TO0	Output	16-bit timer output	Input	P121/TI00
TO50		8-bit timer output (also used for PWM output)		P06/TI50
TO51		8-bit timer output (also used for PWM output)		P07/TI51
PCL	Output	Clock output	Input	P120
SGO	Output	Sound generator output	Input	P47/SGOF
SGOA	Output	Sound generator amplitude output	Input	P46
SGOF	Output	Sound generator frequency output	Input	P47/SGO
ANI0 to ANI3	Input	A/D Converter analog input	Input	P10 P13
AV_{DD}/ AV_{REF}	-	A/D Converter reference voltage input and power supply	-	-
AV_{SS}	-	A/D Converter ground potential. Connect to V_{SS}.	-	-
RESET	Input	System reset input	-	-
X1	-	Connection for main system clock	-	-
X2	-	Connection for main system clock	-	-
CL1	Input	RC connection for subsystem clock	-	-
CL2	-	RC connection for subsystem clock	-	-
$V_{DD}1, V_{DD}2$	-	Positive power supply	-	-
$V_{SS}1, V_{SS}2$	-	Ground potential	-	-
IC1	-	Internal connection. Connect directly to V_{SS}	-	-
IC2	-	Internal connection. Connect directly to V_{DD}	-	-
NC	-	Not connected	-	-

2.3 Description of Pin Functions

2.3.1 P00 to P02, P06 and P07 (Port 0)

This is a 5-bit input/output port. Beside serving as input/output port, it supports functions as an external interrupt input, an external count clock input to the timer and a timer signal output.
The following operating modes can be specified bit-wise.

(1) Port mode

P00 to P02, P06 and P07 function as input/output ports. P00 to P02, P06 and P07 can be specified for input or output ports bitwise with a port mode register 0.

(2) Control mode

In this mode, this port supports the function like external interrupt input, an external count clock input to the timer and a timer signal output.

(a) INTP0 to INTP2

INTP0 to INTP2 are external interrupt input pins which can specify valid edges (rising edge, falling edge, and both rising and falling edges).

(b) TI50

Pin for external count clock input to 8-bit timer/event counter.

(c) TI51

Pin for external count clock input to 8-bit timer/event counter.

(d) TO50

Pin for output of the 8-bit timer/event counter.

(e) TO51

Pin for output of the 8-bit timer/event counter.

2.3.2 P10 to P13 (Port 1)

This is a 4-bit input port. Beside serving as input port, it functions as an A/D converter analog input.
The following operating modes can be specified bit-wise.

(1) Port mode

Thisport functions as 4-bit input ports.

(2) Control mode

This port functions as A/D converter analog input pins (ANI0 to ANI3).

2.3.3 P40 to P47 (Port 4)

This is an 8-bit input/output port. Beside serving as input/output port, this port functions as sound generator output.
The following operating modes can be specified bit-wise.

(1) Port mode

This port functions as an 8-bit input/output port. It can be specified bit-wise as input or output ports with the port mode register 4.

(2) Control mode

This port functions as timer input, clock output, and sound generator output.

(a) SGO, SGOA and SGOF

Pins for separate or composed signal ouput of the sound generator.

2.3.4 P80 to P87 (Port 8)

This is an 8-bit input/output port. Beside serving as input/output port, this port supports an LCD controller/driver.

The following operating modes can be specified bit-wise.

(1) Port mode

This port functions as an 8-bit input/output port. It can be specified bit-wise as input/ output ports with the port mode register 8.

(2) Control mode

In this mode it functions as segment signal output pins (S32 to S39) of the LCD controller/driver.

2.3.5 P90 to P97 (Port 9)

This is an 8-bit input/output port. In addition to its use as an input/output port, it supports also segment signal output function of the LCD controller/driver.

The following operating modes can be specified bit-wise.

(1) Port mode

Port 9 functions as an 8-bit input/output port. Bit-wise specification as an input port or output port is possible by meaning of port mode register 9.

(2) Control mode

Port 9 supports the segment signal output pins (S24 to S31) of the LCD controller/driver.

2.3.6 P100 to P107 (Port 10)

This is an 8-bit input/output port. In addition to its use as an input/output port, it supports also segment signal output functions of the LCD controller/driver.
The following operating modes can be specified bit-wise.

(1) Port mode

Port 10 functions as an 8-bit input/output port. Bit-wise specification as an input port or output port is possible by meaning of port mode register 10.

(2) Control mode

Port 10 supports the segment signal output pins (S16 to S23) of the LCD controller/driver.

2.3.7 P110 to P117 (Port 11)

This is an 8-bit input/output port. In addition to its use as an input/output port, it supports also segment signal output functions of the LCD controller/driver.
The following operating modes can be specified bit-wise.

(1) Port mode

Port 11 functions as an 8-bit input/output port. Bit-wise specification as an input port or output port is possible by meaning of port mode register 11.

(2) Control mode

Port 11 supports the segment signal output pins (S15 to S8) of the LCD controller/driver.

2.3.8 P120 to P127 (Port 12)

These are 8-bit input/output ports. Besides serving as input/output ports, they function as data input/output to/from the serial interface, serial interface clock input/output, as segment signal output pins of LCD controller/driver and as processor clock output.

The following operating modes can be specified bit-wise.

(1) Port mode

These ports function as 8-bit input/output ports. They can be specified bit-wise as input or output ports with port mode register 12.

(2) Control mode

These ports function as serial interface data input/output, clock input/output.

(a) SI3, SO3

Serial interface serial data input/output pins

(b) SCK3

Serial interface serial clock input/output pins

(c) RxD0, TxD0

Asynchronous serial interface data input/output pins

(d) PCL

Clock output pin.

(e) LCD controller/driver

These ports function as segment output signal pins (S0 to S7) of LCD controller/driver.

Caution: When this port is used as a serial interface, the I/O and output latches must be set according to the function the user requires.

2.3.9 COM0 to COM3

These are LCD controller/driver common signal output pins. They output common signals under the following condition:
- static mode
- 1/2 duty cycle is performed in 1/2 bias mode
- 1/3 duty cycle is performed in 1/2 bias mode
- 1/3 duty cycle is performed in 1/3 bias mode
- 1/4 duty cycle is performed in 1/3 bias mode

2.3.10 VLC0 to VLC2

These are LCD drive voltage pins. In the Flash EEPROM and the MaskROM product an external split resistors are necessary.

2.3.11 AV$_{DD}$/AV$_{REF}$

A/D converter reference voltage input pin and the power supply for the A/D-converter. When A/D converter is not used, connect this pin to V$_{DD}$.

2.3.12 AV$_{SS}$

This is a ground voltage pin of A/D converter. Always use the same voltage as that of the VSS pin even when A/D converter is not used.

2.3.13 RESET

This is a low-level active system reset input pin.

2.3.14 X1 and X2

Crystal resonator connect pins for main system clock oscillation. For external clock supply, input it to X1.

2.3.15 CL1 and CL2

Crystal resonator connect pins for subsystem clock oscillation. For external clock supply, input it to CL1 and let CL2 open.

2.3.16 V$_{DD0}$/V$_{DD1}$

Positive power supply pins.

2.3.17 V$_{SS0}$/V$_{SS1}$

Ground potential pins.

2.3.18 V$_{PP}$ (µPD16F15A only)

High-voltage apply pin for FLASH programming mode setting. Connect it directly to VSS with the shortest possible wire in the normal operating mode. When a voltage difference is produced between the IC pin and V$_{SS}$ pin because the wiring between those two pins is too long or an external noise is input to the IC pin, the user's program may not run normally.

Figure 2-1: Connection of IC Pins

• **Connect IC pins to Vss pins directly**.

Vss   IC

As short as possible
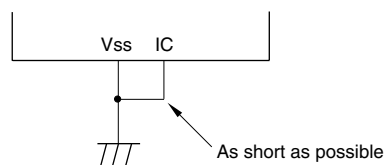
2.4 Pin I/O Circuits and Recommended Connection of Unused Pins

The input/output circuit type of each pin and recommended connection of unused pins are shown in the following table.

For the input/output circuit configuration of each type, see table.

Table 2-3-1: Types of Pin Input/Output Circuits μPD1615A(A), μPD1615B(A), μPD1615F(A), μPD16F15A (1/2)

Pin Name	Input/Output Circuit Type	I/O	Recommended Connection for Unused Pins
P00/INTP0	8	I/O	Connect to V$_{SS}$ via a resistor individually
P01/INTP1			
P02/INTP2			
P06/TI50/TO50			
P07/TI51/TO51			
P10/ANI0	9	I	Connect directly to V$_{DD}$ or V$_{SS}$
P11/ANI1			
P12/ANI2			
P13/ANI3			
P40	5	I/O	Connect to V$_{DD}$ or V$_{SS}$ via a resistor individually
P41			
P42			
P43			
P44			
P45			
P46/SGOA			
P47/SGO/SGOF			
P80/S39	17	I/O	Connect to V$_{DD}$ or V$_{SS}$ via a resistor individually
P81/S38			
P82/S37			
P83/S36			
P84/S35			
P85/S34			
P86/S33			
P87/S32			
P90/S31	17	I/O	Connect to V$_{DD}$ or V$_{SS}$ via a resistor individually
P91/S30			
P92/S29			
P93/S28			
P94/S27			
P95/S26			
P96/S25			
P97/S24			

Table 2-3-1: Types of Pin Input/Output Circuits µPD1615A(A), µPD1615B(A), µPD1615F(A),
µPD16F15A (2/2)

Pin Name	Input/Output Circuit Type	I/O	Recommended Connection for Unused Pins
P100/S23	17	I/O	Connect to V_{DD} or V_{SS} via a resistor individually
P101/S22			
P102/S21			
P103/S20			
P104/S19			
P105/S18			
P106/S17			
P107/S16			
P110/S15	17	I/O	Connect to V_{DD} or V_{SS} via a resistor individually
P111/S14			
P112/S13			
P113/S12			
P114/S11			
P115/S10			
P116/S9			
P117/S8			
P120/S7/PCL	17	I/O	Connect to V_{DD} or V_{SS} via a resistor individually
P121/S6/TI00/TO0	17-C		
P122/S5/TI01	17-C		
P123/S4/RxD0	17-C		
P124/S3/TxD0	17		
P125/S2/SCK3	17-C		
P126/S1/SO3	17		
P127/S0/SI3	17-C		
COM0 to COM3	18	O	Leave open
V_{LC0} to V_{LC2}	-	-	Connect to V_{DD}
Rx0VAN, Rx1VAN, Rx2VAN	2	I	-
TxVAN	19	O	-
CL1	-	I	Connect to V_{DD} or V_{SS}
CL2	-	-	Leave open
$\overline{RESET}$	2	I	-
AV$_{DD}$/AV$_{REF}$	-	I	Connect to V_{DD}
AV$_{SS}$	-	-	Connect to V_{SS}
IC	-	-	Connect directly to V_{SS}
V$_{PP}$	1	-	Connect directly to V_{SS} (except for flash programming)

Table 2-3-2: Types of Pin Input/Output Circuits μPD1616F(A) (1/2)

Pin Name	Input/Output Circuit Type	I/O	Recommended Connection for Unused Pins
P00/INTP0	8	I/O	Connect to V$_{DD}$ or V$_{SS}$ via a resistor individually
P01/INTP1			
P02/INTP2			
P06/TI50/TO50			
P07/TI51/TO51			
P10/ANI0	9	I	Connect directly to V$_{DD}$ or V$_{SS}$
P11/ANI1			
P12/ANI2			
P13/ANI3			
P40	5	I/O	Connect to V$_{DD}$ or V$_{SS}$ via a resistor individually
P41			
P42			
P43			
P44			
P45			
P46/SGOA			
P47/SGO/SGOF			
P80	5	I/O	Connect to V$_{DD}$ or V$_{SS}$ via a resistor individually
P81			
P82			
P83			
P84			
P85			
P86			
P87			
P90	5	I/O	Connect to V$_{DD}$ or V$_{SS}$ via a resistor individually
P91			
P92			
P93			
P94			
P95			
P96			
P97			

Table 2-3-2: Types of Pin Input/Output Circuits µPD1616F(A) (2/2)

Pin Name	Input/Output Circuit Type	I/O	Recommended Connection for Unused Pins
P100	8	I/O	Connect to V$_{DD}$ or V$_{SS}$ via a resistor individually
P101			
P102			
P103			
P104			
P105			
P106			
P107			
P110	5	I/O	Connect to V$_{DD}$ or V$_{SS}$ via a resistor individually
P111			
P112			
P113			
P114			
P115			
P116			
P117			
P120/ PCL	5	I/O	Connect to V$_{DD}$ or V$_{SS}$ via a resistor individually
P121/TI00/TO0	8		
P122/TI01	8		
P123/RxD0	8		
P124/ TxD0	5		
P125/ SCK3	8		
P126/SO3	5		
P127/SI3	8		
Rx0VAN, Rx1VAN, Rx2VAN	2	I	-
TxVAN	19	O	-
CL1	-	I	Connect to V$_{DD}$ or V$_{SS}$
CL2	-	-	Leave open
RESET	2	I	-
AV$_{DD}$/AV$_{REF}$	-	I	Connect to V$_{DD}$
AV$_{SS}$	-	-	Connect to V$_{SS}$
IC1	-	-	Connect directly to V$_{SS}$
IC2	-	-	Connect directly to V$_{DD}$
NC	-	-	Leave open

Figure 2-2: Pin Input/Output Circuits (1/2)
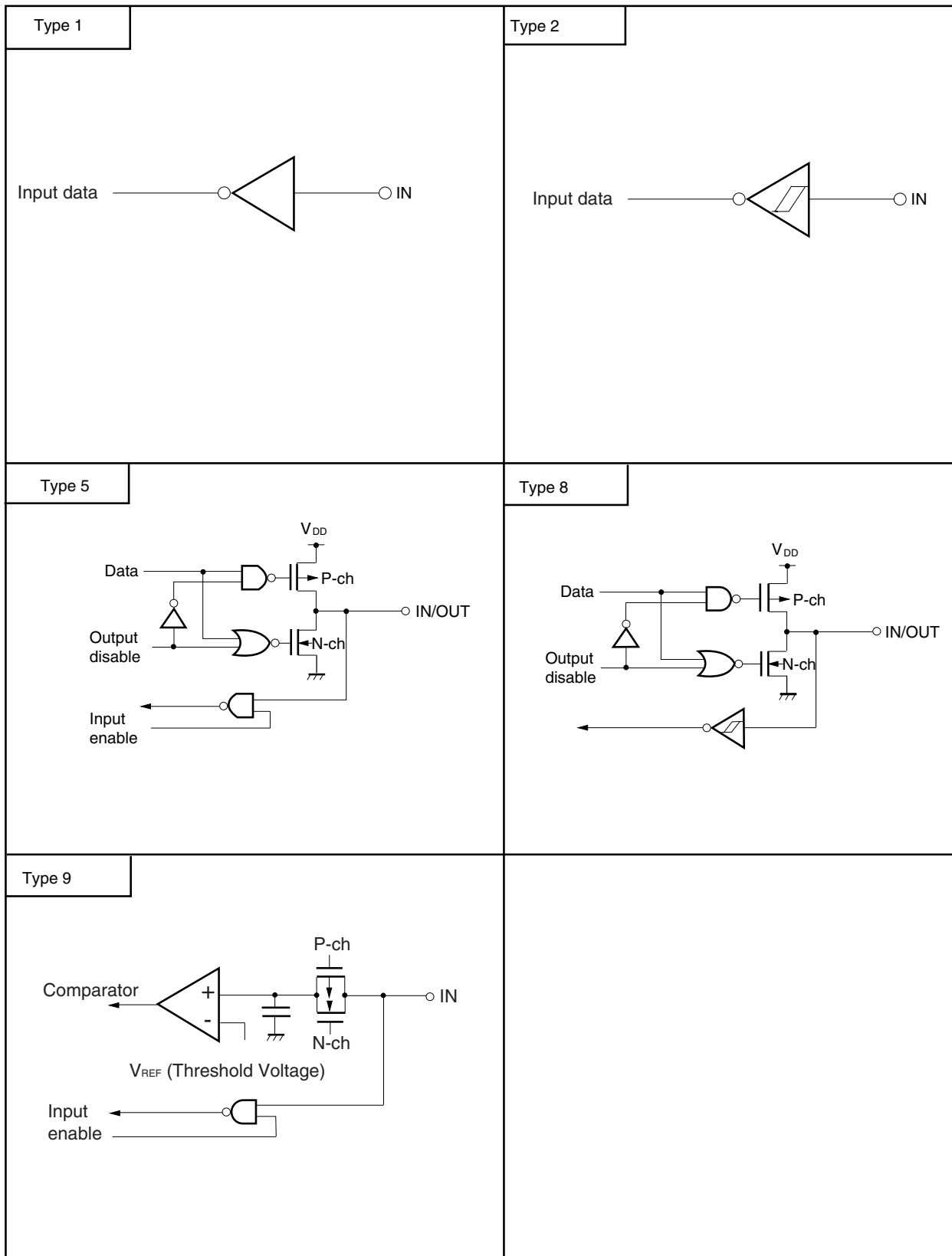
Figure 2-2: Pin Input/Output Circuits (2/2)
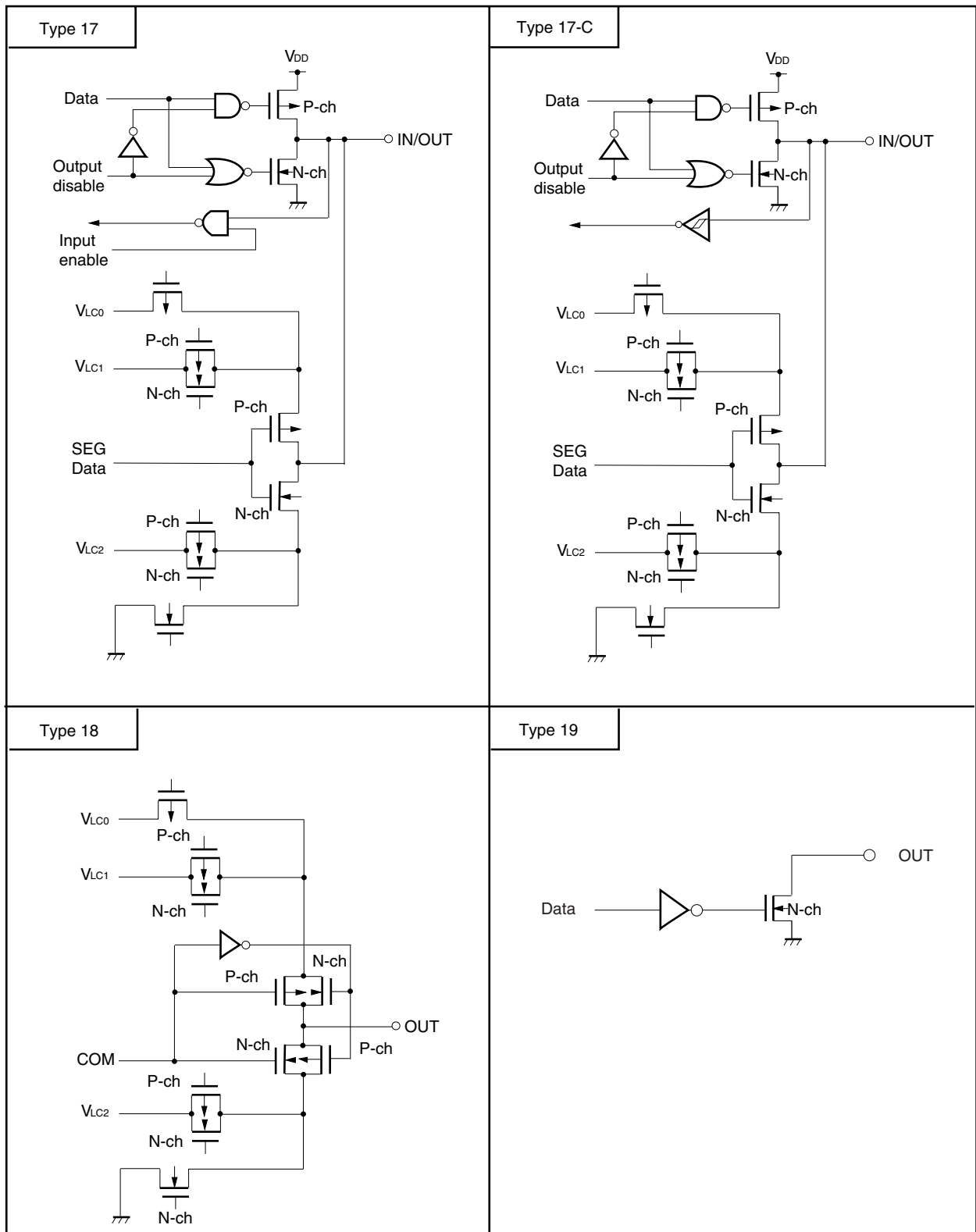
[Memo]

Chapter 3 CPU Architecture

3.1 Memory Space

The memory map of the µPD1615A(A) is shown in Figure 3-1.

Figure 3-1: Memory Map µPD1615A(A)
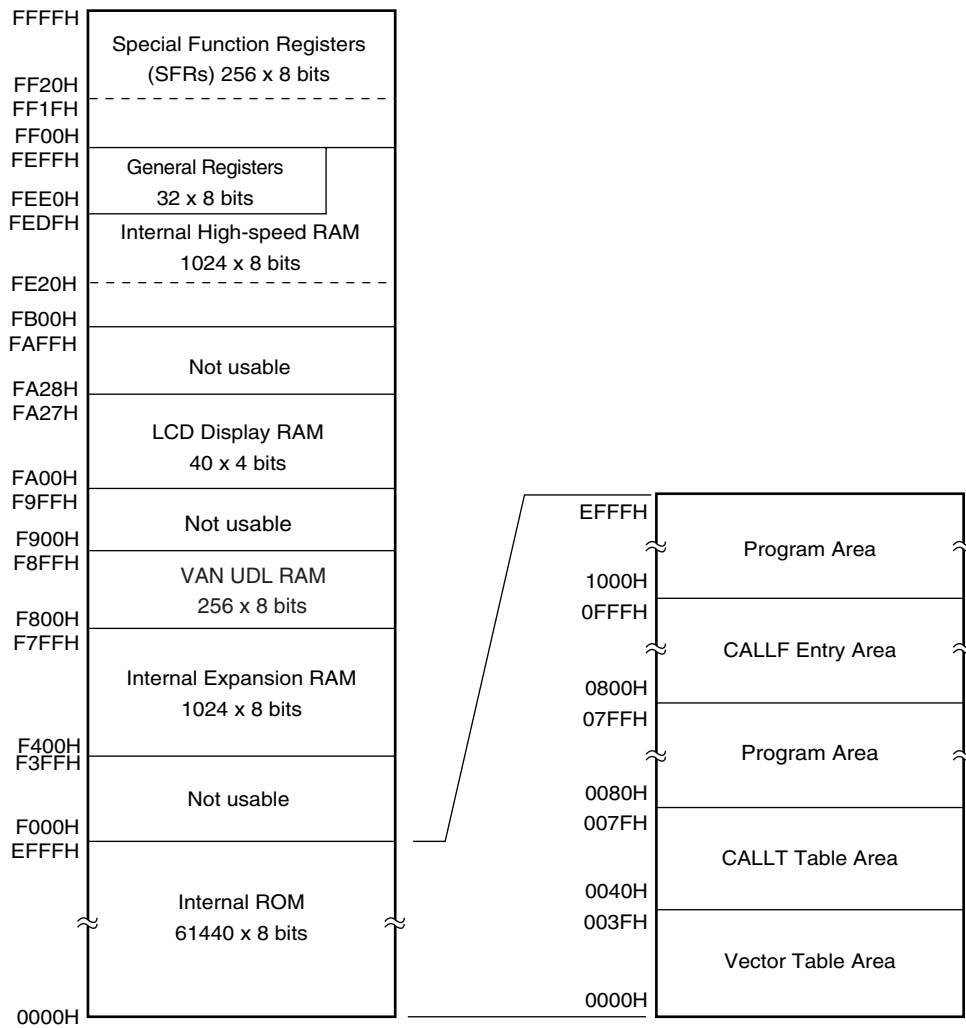
The memory map of the µPD1615B(A) is shown in Figure 3-2.

Figure 3-2: Memory Map µPD1615B(A)
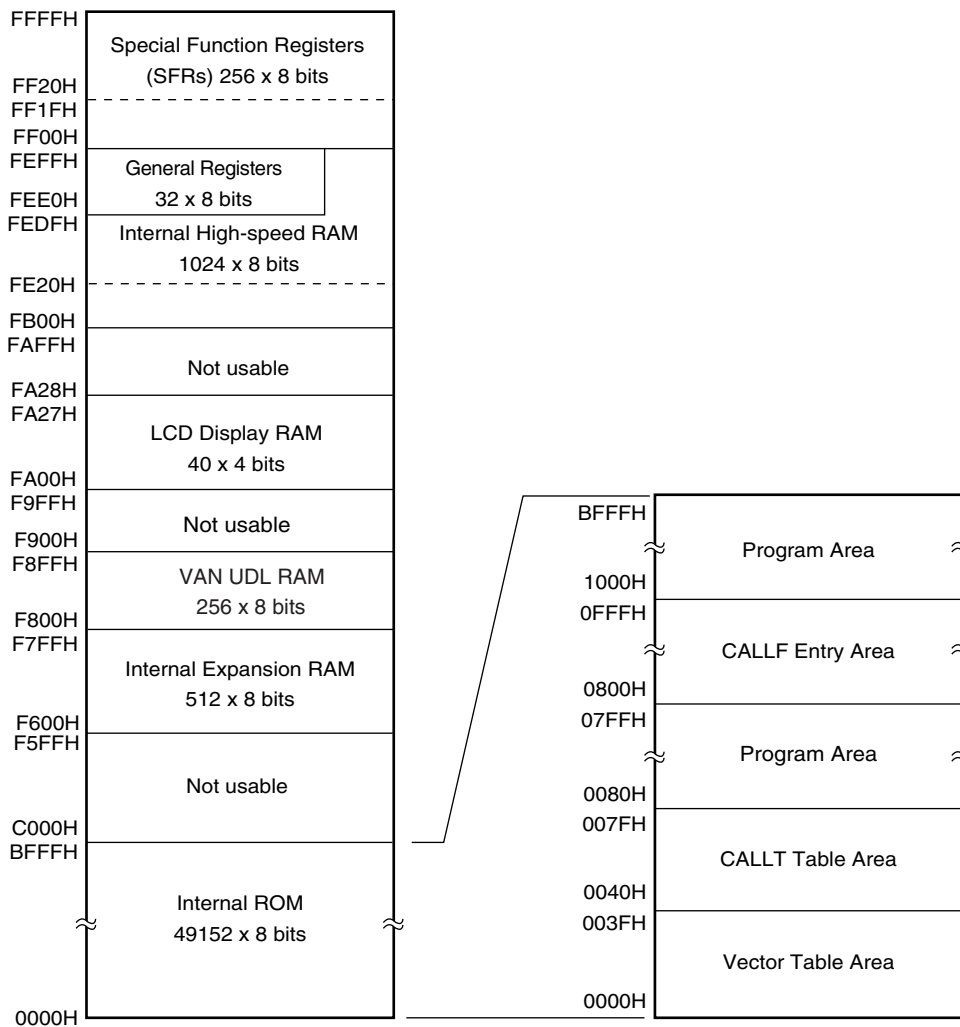
The memory map of the µPD1615F(A) is shown in Figure 3-3.

Figure 3-3: Memory Map µPD1615F(A)
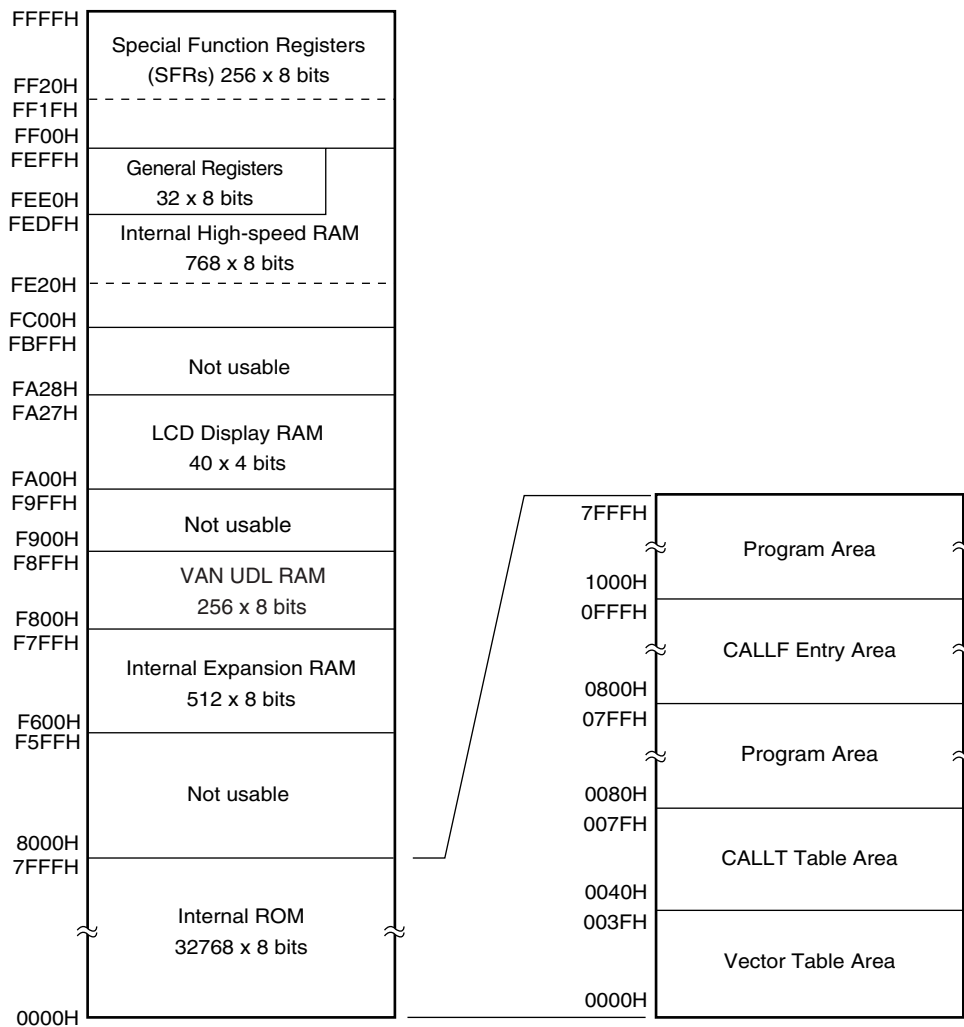
The memory map of the µPD1616F(A) is shown in Figure 3-4.

Figure 3-4: Memory Map µPD1616F(A)

Address	Region
FFFFH	Special Function Registers (SFRs) 256 x 8 bits
FF20H	
FF1FH – FF00H	
FEFFH – FEE0H	General Registers 32 x 8 bits
FEDFH	Internal High-speed RAM 768 x 8 bits
FE20H	
FC00H	
FBFFH – F900H	Not usable
F8FFH – F800H	VAN UDL RAM 256 x 8 bits
F7FFH – F600H	Internal Expansion RAM 512 x 8 bits
F5FFH – 8000H	Not usable
7FFFH – 0000H	Internal ROM 32768 x 8 bits

Address	Region
7FFFH – 1000H	Program Area
0FFFH – 0800H	CALLF Entry Area
07FFH – 0080H	Program Area
007FH – 0040H	CALLT Table Area
003FH – 0000H	Vector Table Area

The memory map of the µPD16F15A is shown in Figure 3-5.

Figure 3-5: Memory Map µPD16F15A

FFFFH	Special Function Registers (SFRs) 256 x 8 bits
FF20H	
FF1FH	
FF00H	
FEFFH	General Registers 32 x 8 bits
FEE0H	
FEDFH	Internal High-speed RAM 1024 x 8 bits
FE20H	
FB00H	
FAFFH	Not usable
FA28H	
FA27H	LCD Display RAM 40 x 4 bits
FA00H	
F9FFH	Not usable
F900H	
F8FFH	VAN UDL RAM 256 x 8 bits
F800H	
F7FFH	Internal Expansion RAM 1024 x 8 bits
F400H	
F3FFH	Not usable
F000H	
EFFFH	Internal ROM 61440 x 8 bits
0000H	

EFFFH	Program Area
1000H	
0FFFH	CALLF Entry Area
0800H	
07FFH	Program Area
0080H	
007FH	CALLT Table Area
0040H	
003FH	Vector Table Area
0000H	

3.1.1 Internal program memory space

The internal program memory space stores programs and table data. This is generally accessed by the program counter (PC). The µPD1615A subseries have various size of internal ROMs or Flash EPROM as shown below.

Table 3-1: Internal ROM Capacities

Part Number	Internal ROM	
	Type	Capacity
µPD1615A(A)	Mask ROM	61440 x 8-bits
µPD1615B(A)	Mask ROM	49152 x 8-bits
µPD1615F(A)	Mask ROM	32768 x 8-bits
µPD1616F(A)	Mask ROM	32768 x 8-bits
µPD16F15A	Flash ROM	61440 x 8-bits

The internal program memory is divided into three areas: vector table area, CALLT instruction table area, and CALLF instruction table area. These areas are described on the next page.

(1) Vector table area

The 64-byte area 0000H to 003FH is reserved as a vector table area. The $\overline{\text{RESET}}$ input and program start addresses for branch upon generation of each interrupt request are stored in the vector table area.

Of the 16-bit address, low-order 8 bits are stored at even addresses and high-order 8 bits are stored at odd addresses.

Table 3-2: Vectored Interrupts

Vector Table Address	Interrupt Request
0004H	INWDT
0006H	INTVE
0008H	INTVT
000AH	INTTVR
000CH	INTP0
000EH	INTP1
0010H	INTP2
0012H	INTTM00
0014H	INTTM01
0016H	INTTM50
0018H	INTTM51
001AH	INTWTI
001CH	INTWT
001EH	INTCSI3
0020H	INTSER
0022H	INTSR
0024H	INTST
0026H	INTAD

(2) CALLT instruction table area

The 64-byte area 0040H to 007FH can store the subroutine entry address of a 1-byte call instruction (CALLT).

(3) CALLF instruction entry area

The area 0800H to 0FFFH can perform a direct subroutine call with a 2-byte call instruction (CALLF).

3.1.2 Internal data memory space

The µPD1615A subseries units incorporate the following RAMs.

(1) Internal high-speed RAM

This is a 1024 x 8-bit configuration in the area FB00H to FEFFH or a 768 x 8 bits configuration in the area FC00H to FEFFH. The 4 banks of general registers, each bank consisting of eight 8-bit registers, are allocated in the 32-byte area FEE0H to FEFFH.

The internal high-speed RAM can also be used as a stack memory.

(2) LCD-Display RAM

Buffer RAM is allocated to the 40 x 4 bits area from FA00H to FA27H. LCD-Display RAM can also be used as normal RAM. The LCD Display RAM is not available in the µPD1616F(A).

(3) Internal expansion RAM

Internal expansion RAM is allocated to the 1024-byte area from F400H to F7FFH for the µPD1615A(A) and the µPD16F15A. For the µPD1615B(A), µPD1615F(A), and µPD1616F(A) is the 512-byte area located between F600H and F7FFH.

(4) VAN UDL RAM

The VAN UDL RAM is located in a 256-byte area from F800H to F8FFH.

3.1.3 Special function register (SFR) area

An on-chip peripheral hardware special function register (SFR) is allocated in the area FF00H to FFFFH. (Refer to **Table 3-3**).

Caution: Do not access addresses where the SFR is not assigned.

3.1.4 Data memory addressing

The µPD1615A subseries is provided with a varity of addressing modes which take account of memory manipulability, etc. Special addressing methods are possible to meet the functions of the special function registers (SFRs) and general registers. The data memory space is the entire 64K-byte space (0000H to FFFFH). Figures 3-6 to 3-10 show the data memory addressing modes.

For details of addressing, refer to **3.4 Operand Address Addressing**.

Figure 3-6: Data Memory Addressing µPD1615A(A)
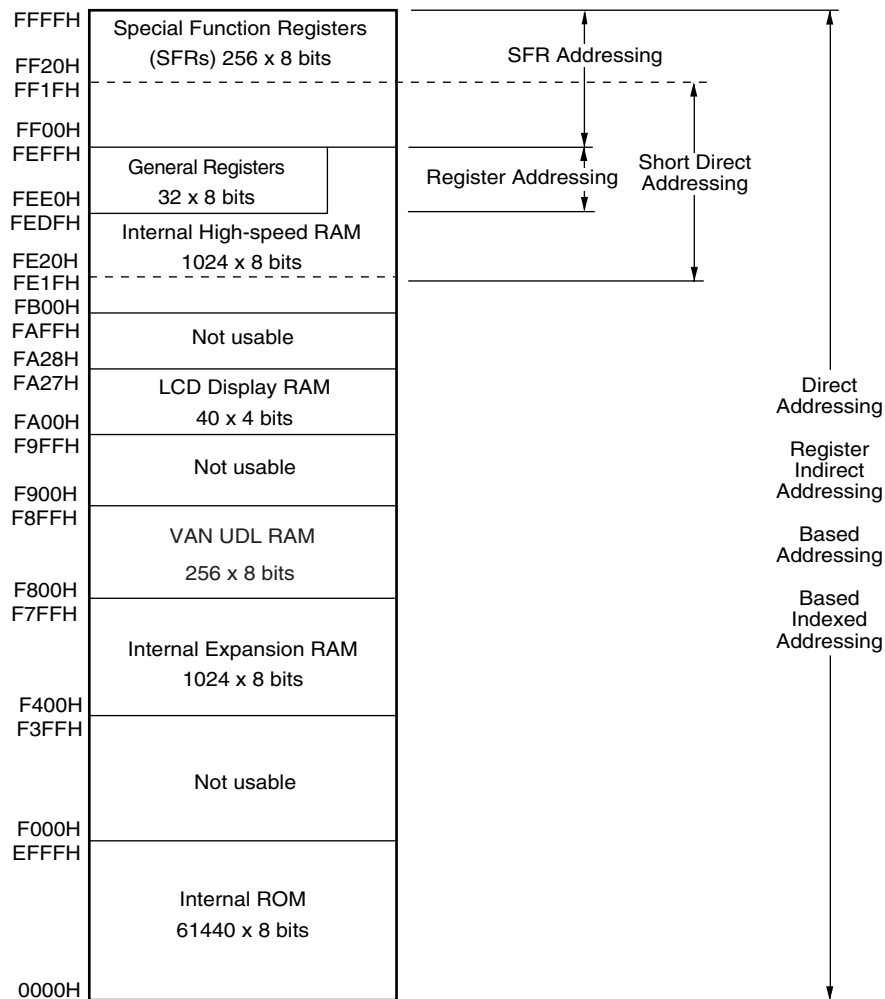
Figure 3-7: Data Memory Addressing µPD1615B(A)

FFFFH	Special Function Registers
FF20H	(SFRs) 256 x 8 bits
FF1FH	
FF00H	
FEFFH	
	General Registers
FEE0H	32 x 8 bits
FEDFH	Internal High-speed RAM
FE20H	1024 x 8 bits
FE1FH	
FB00H	
FAFFH	Not usable
FA28H	
FA27H	LCD Display RAM
FA00H	40 x 4 bits
F9FFH	Not usable
F900H	
F8FFH	VAN UDL RAM
	256 x 8 bits
F800H	
F7FFH	Internal Expansion RAM
	512 x 8 bits
F600H	
F5FFH	
	Not usable
C000H	
BFFFH	Internal ROM
	49152 x 8 bits
0000H	

SFR Addressing

Register Addressing

Short Direct Addressing

Direct Addressing

Register Indirect Addressing

Based Addressing

Based Indexed Addressing

Figure 3-8: Data Memory Addressing μPD1615F(A)

FFFFH	Special Function Registers (SFRs) 256 x 8 bits
FF20H	
FF1FH	
FF00H	
FEFFH	General Registers 32 x 8 bits
FEE0H	
FEDFH	Internal High-speed RAM 768 x 8 bits
FE20H	
FE1FH	
FC00H	
FBFFH	Not usable
FA28H	
FA27H	LCD Display RAM 40 x 4 bits
FA00H	
F9FFH	Not usable
F900H	
F8FFH	VAN UDL RAM 256 x 8 bits
F800H	
F7FFH	Internal Expansion RAM 512 x 8 bits
F600H	
F5FFH	Not usable
8000H	
7FFFH	Internal ROM 32768 x 8 bits
0000H	

SFR Addressing

Register Addressing

Short Direct Addressing

Direct Addressing

Register Indirect Addressing

Based Addressing

Based Indexed Addressing

Figure 3-9: Data Memory Addressing µPD1616F(A)
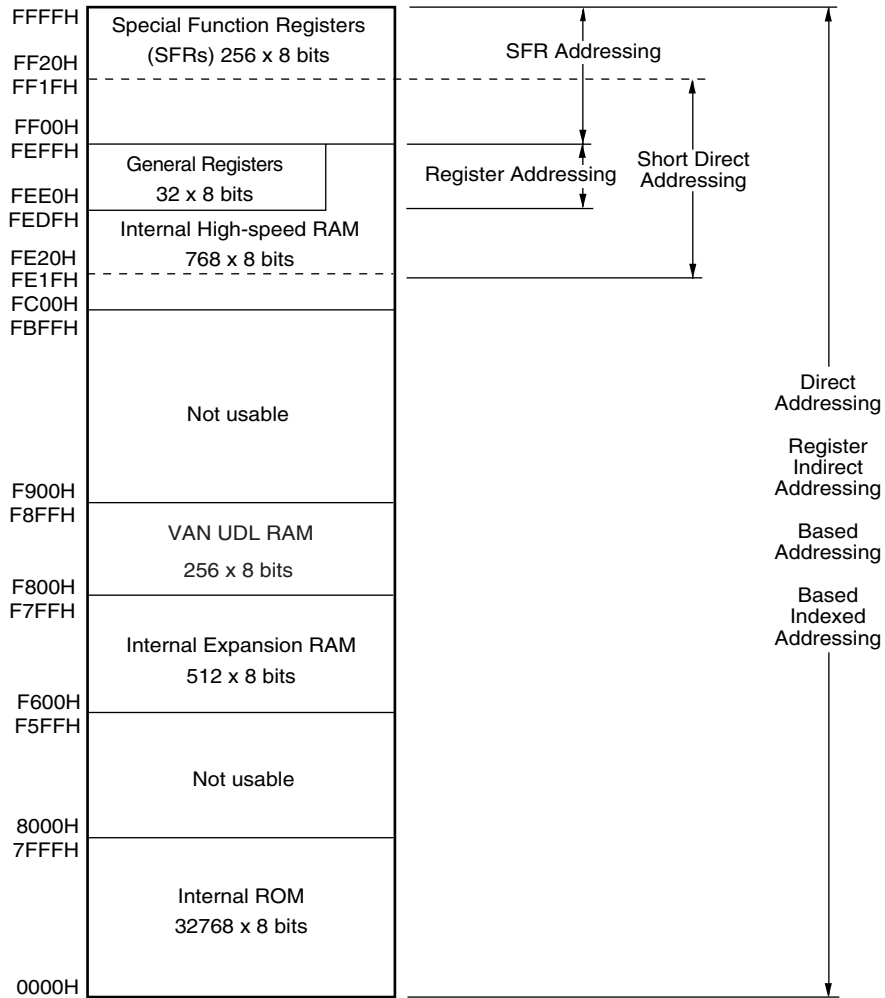
Figure 3-10: Data Memory Addressing µPD16F15A
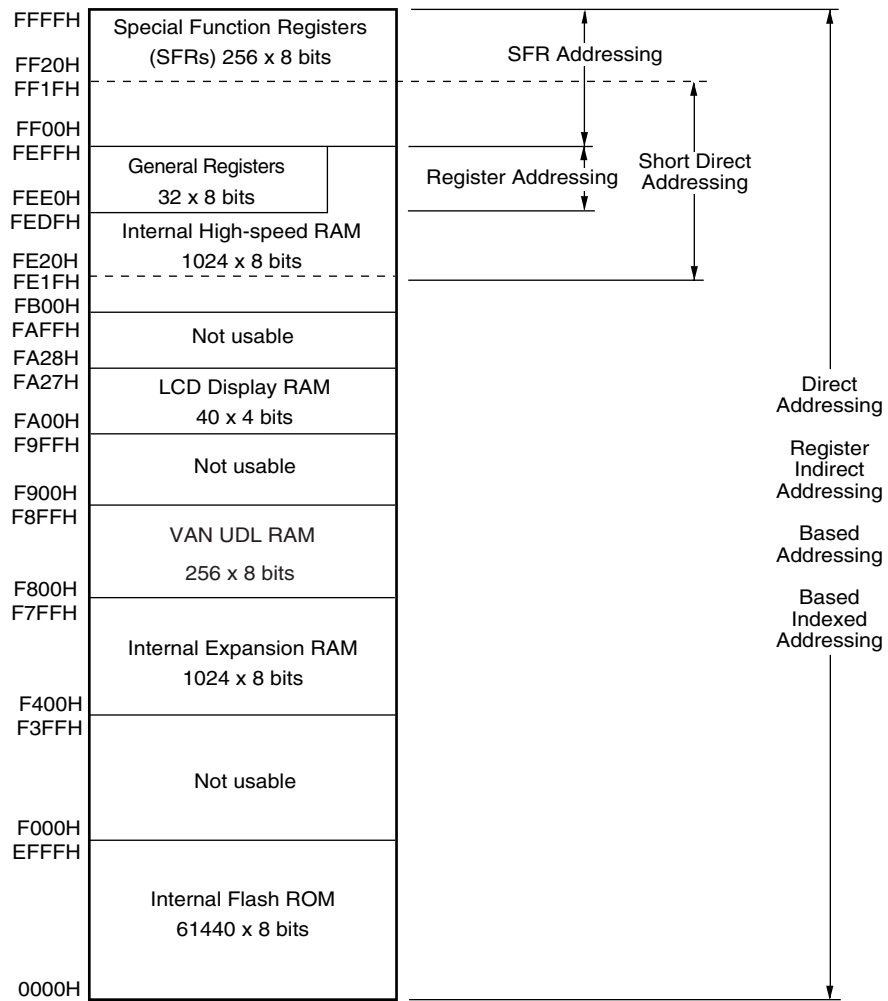
3.2 Processor Registers

The μPD1615A subseries units incorporate the following processor registers.

3.2.1 Control registers

The control registers control the program sequence, statuses, and stack memory. The control registers consist of a program counter, a program status word and a stack pointer.

(1) Program counter (PC)

The program counter is a 16-bit register which holds the address information of the next program to be executed.

In normal operation, the PC is automatically incremented according to the number of bytes of the instruction to be fetched. When a branch instruction is executed, immediate data and register contents are set.

$\overline{\text{RESET}}$ input sets the reset vector table values at addresses 0000H and 0001H to the program counter.

Figure 3-11: Program Counter Configuration

15	0
PC	
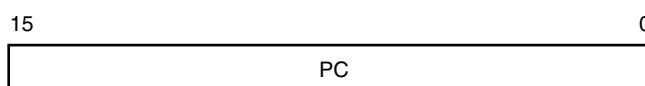
(2) Program status word (PSW)

The program status word is an 8-bit register consisting of various flags to be set/reset by instruction execution.

Program status word contents are automatically stacked upon interrupt request generation or PUSH PSW instruction execution and are automatically reset upon execution of the RETB, RETI and POP PSW instructions.

$\overline{\text{RESET}}$ input sets the PSW to 02H.

Figure 3-12: Program Status Word Configuration

7							0
IE	Z	RBS1	AC	RBS0	0	ISP	CY
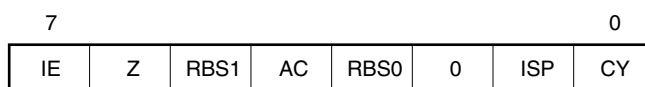
(a) Interrupt enable flag (IE)

This flag controls the interrupt request acknowledge operations of the CPU.

When 0, the IE is set to interrupt disabled (DI) status. All interrupts except non-maskable interrupt are disabled.

When 1, the IE is set to interrupt enabled (EI) status and interrupt request acknowledge is controlled with an in-service priority flag (ISP), an interrupt mask flag for various interrupt sources, and a priority specification flag.

The IE is reset to (0) upon DI instruction execution or interrupt request acknowledgement and is set to (1) upon EI instruction execution.

(b) Zero flag (Z)

When the operation result is zero, this flag is set (1). It is reset (0) in all other cases.

(c) Register bank select flags (RBS0 and RBS1)

These are 2-bit flags to select one of the four register banks.
In these flags, the 2-bit information which indicates the register bank selected by SEL RBn instruction execution is stored.

(d) Auxiliary carry flag (AC)

If the operation result has a carry from bit 3 or a borrow at bit 3, this flag is set (1). It is reset (0) in all other cases.

(e) In-service priority flag (ISP)

This flag manages the priority of acknowledgeable maskable vectored interrupts. When 0, acknowledgment of the vectored interrupt request specified to low-order priority with the priority specify flag registers (PR0L, PR0H, and PR1L) is disabled. Whether an actual interrupt request is acknowledged or not is controlled with the interrupt enable flag (IE).

(f) Carry flag (CY)

This flag stores overflow and underflow upon add/subtract instruction execution. It stores the shift-out value upon rotate instruction execution and functions as a bit accumulator during bit manipulation instruction execution.

(3) Stack pointer (SP)

This is a 16-bit register to hold the start address of the memory stack area. Only the internal high-speed RAM area can be set as the stack area.

Figure 3-13: Stack Pointer Configuration

```
15                                                          0
┌──────────────────────────────────────────────────────────┐
│                           SP                               │
└──────────────────────────────────────────────────────────┘
```
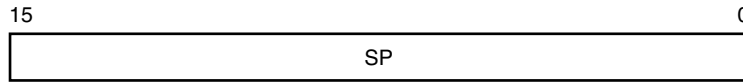
The SP is decremented ahead of write (save) to the stack memory and is incremented after read (reset) from the stack memory.
Each stack operation saves/resets data as shown in Figures 3-8 and 3-9.

Caution: **Since RESET input makes SP contents indeterminate, be sure to initialize the SP before instruction execution.**

Figure 3-14: Data to be Saved to Stack Memory
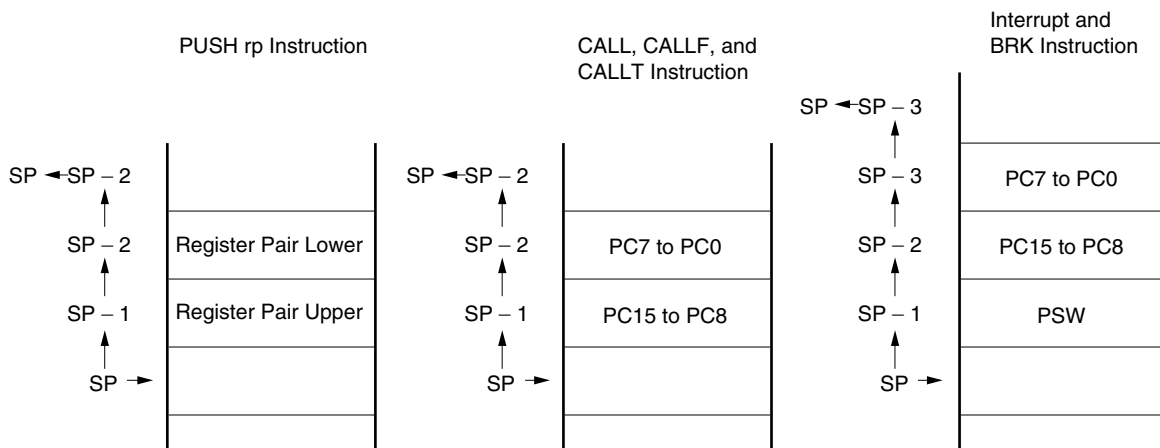
Figure 3-15: Data to be Reset to Stack Memory
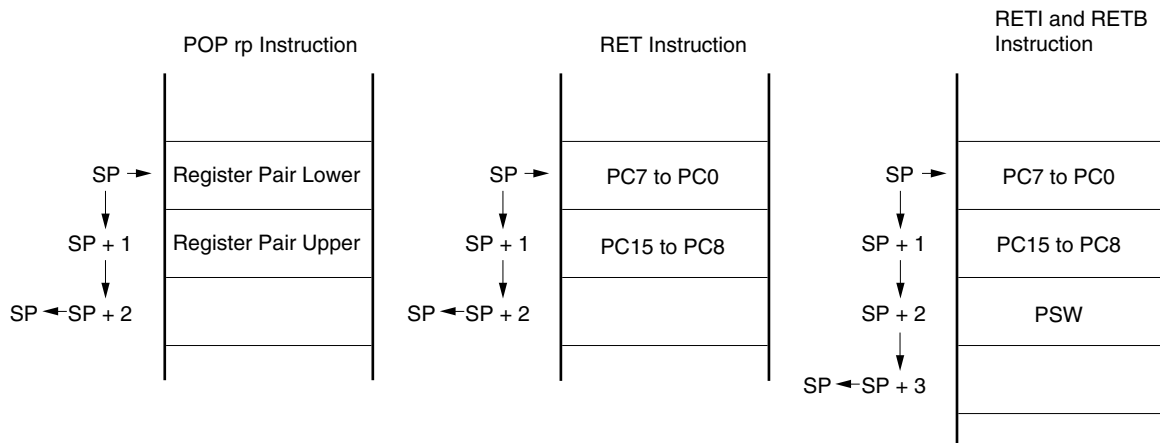
3.2.2 General registers

A general register is mapped at particular addresses (FEE0H to FEFFH) of the data memory. It consists of 4 banks, each bank consisting of eight 8-bit registers (X, A, C, B, E, D, L, and H).

Each register can also be used as an 8-bit register. Two 8-bit registers can be used in pairs as a 16-bit register (AX, BC, DE, and HL).

They can be described in terms of function names (X, A, C, B, E, D, L, H, AX, BC, DE, and HL) and absolute names (R0 to R7 and RP0 to RP3).

Register banks to be used for instruction execution are set with the CPU control instruction (SEL RBn). Because of the 4-register bank configuration, an efficient program can be created by switching between a register for normal processing and a register for interruption for each bank.

Figure 3-16: General Register Configuration

(a) Absolute Name
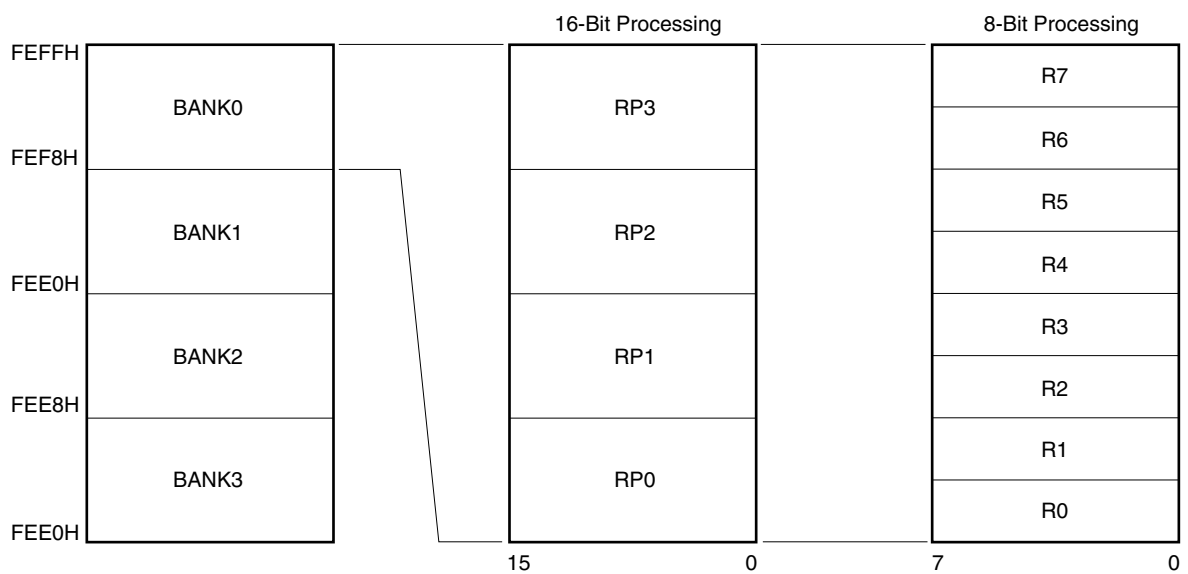
(b) Function Name
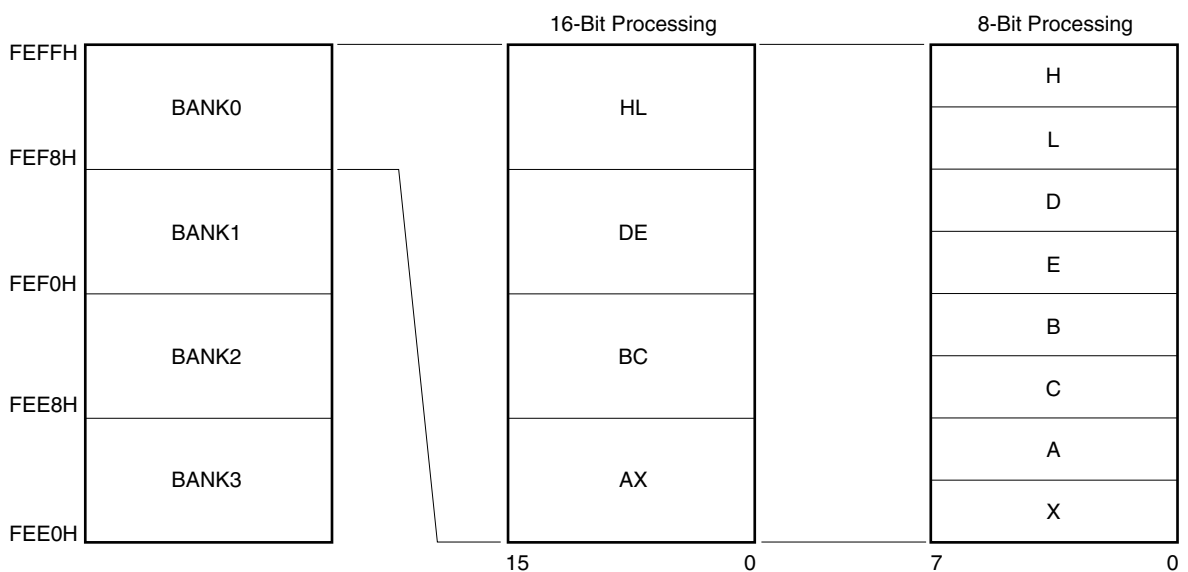
3.2.3 Special function register (SFR)

Unlike a general register, each special function register has special functions.

It is allocated in the FF00H to FFFFH area.

The special function registers can be manipulated in a similar way as the general registers, by using operation, transfer, or bit-manipulate instructions. The special function registers are read from and written to in specified manipulation bit units (1, 8, and/or 16) depending on the register type.

Each manipulation bit unit can be specified as follows.

- 1-bit manipulation
  Describe the symbol reserved with assembler for the 1-bit manipulation instruction operand (sfr.bit).
  This manipulation can also be specified with an address.
- 8-bit manipulation
  Describe the symbol reserved with assembler for the 8-bit manipulation instruction operand (sfr).
  This manipulation can also be specified with an address.
- 16-bit manipulation
  Describe the symbol reserved with assembler for the 16-bit manipulation instruction operand (sfrp).
  When addressing an address, describe an even address.

Table 3-3 gives a list of special function registers. The meaning of items in the table is as follows.

- Symbol
  The assembler software translates these symbols into corresponding addresses where the special function registers are allocated. These symbols should be used as instruction operands in the case of programming.
- R/W
  This column shows whether the corresponding special function register can be read or written.
  R/W   : Both reading and writing are enabled.
  R   : The value in the register can read out. A write to this register is ignored.
  W   : A value can be written to the register. Reading values from the register is impossible.
- Manipulation
  The register can be manipulated in bit units.
- After reset
  The register is set to the value immediately after the $\overline{\text{RESET}}$ signal is input.

Table 3-3: Special Function Register List (1/2)

Address	SFR Name	Symbol	R/W	Manipulatable Bit Unit			After
				1 bit	8 bits	16 bits	Reset
FF00H	Port 0	P0	R/W	○	○	-	00H
FF01H	Port 1	P1	R	○	○	-	00H
FF04H	Port 4	P4	R/W	○	○	-	00H
FF08H	Port 8	P8	R/W	○	○	-	00H
FF09H	Port 9	P9	R/W	○	○	-	00H
FF0AH	Port 10	P10	R/W	○	○	-	00H
FF0BH	Port 11	P11	R/W	○	○	-	00H
FF0CH	Port 12	P12	R/W	○	○	-	00H
FF10H	8-bit compare register 50	CR50	R/W	-	○	-	00H
FF11H	8-bit compare register 51	CR51	R/W	-	○	-	00H
FF12H	8-bit timer/counter 50	TM50	R	-	○	-	00H
FF13H	8-bit timer/counter 51	TM51	R	-	○	-	00H
FF14H	16-bit capture/compare register 00	CR00	R/W	-	-	○	0000H
FF15H							
FF16H	16-bit capture/compare register 01	CR01	R/W	-	-	○	0000H
FF17H							
FF18H	Serial shift register	SIO3	R/W	-	○	-	00H
FF1AH	Transmission shift register	TXS0	W	-	○	-	FFH
	Reception shift register	RXB0	R	-	○	-	FFH
FF1BH	A/D conversion result register	ADCR1	R	-	○	-	00H
FF20H	Port mode register 0	PM0	R/W	○	○	-	FFH
FF24H	Port mode register 4	PM4	R/W	○	○	-	FFH
FF28H	Port mode register 8	PM8	R/W	○	○	-	FFH
FF29H	Port mode register 9	PM9	R/W	○	○	-	FFH
FF2AH	Port mode register 10	PM10	R/W	○	○	-	FFH
FF2BH	Port mode register 11	PM11	R/W	○	○	-	FFH
FF2CH	Port mode register 12	PM12	R/W	○	○	-	FFH
FF40H	Clock output select register	CKS	R/W	○	○	-	00H
FF41H	Watch timer operation mode register	WTM	R/W	○	○	-	00H
FF42H	Watchdog timer clock select register	WDCS	R/W	-	○	-	00H
FF48H	External interrupt rising edge enable register	EGP	R/W	○	○	-	00H
FF49H	External interrupt falling edge enable register	EGN	R/W	○	○	-	00H
FF58H	Port function register 8	PF8	R/W	○	○	-	00H
FF59H	Port function register 9	PF9	R/W	○	○	-	00H
FF5AH	Port function register 10	PF10	R/W	○	○	-	00H
FF5BH	Port function register 11	PF11	R/W	○	○	-	00H
FF5CH	Port function register 12	PF12	R/W	○	○	-	00H

Table 3-3: Special Function Register List (2/2)

Address	SFR Name	Symbol		R/W	Manipulatable Bit Unit			After Reset
					1 bit	8 bits	16 bits	
FF60H	16-bit timer mode control register 0	TMC0		R/W	O	O	-	00H
FF61H	Prescaler mode register 0	PRM0		R/W	-	O	-	00H
FF62H	Capture compare control register 0	CRC0		R/W	O	O	-	00H
FF63H	Timer output control register 0	TOC0		R/W	O	O	-	00H
FF64H	16-bit timer/counter 0	TM0		R	-	-	O	00H
FF65H								
FF66H	Sound generator control register	SGCR		R/W	O	O	-	00H
FF67H	Sound generator 7-bit amplitude register	SGAM		R/W	O	O	-	00H
FF68H	Sound generator buzzer control register	SGBR		R/W	O	O	-	00H
FF6FH	Serial I/F mode register	CSIM3		R/W	O	O	-	00H
FF70H	8-bit timer mode control register 50	TMC50		R/W	O	O	-	04H
FF71H	Timer clock select register 50	TCL50		R/W	-	O	-	00H
FF74H	8-bit timer mode control register 51	TMC51		R/W	O	O	-	04H
FF75H	Timer clock select register 51	TCL51		R/W	-	O	-	00H
FF78H	VAN-UDL clock control register	UDLCCL		R/W	O	-	-	00H
FF80H	A/D converter mode register 1	ADM1		R/W	O	O	-	00H
FF81H	Analog input channel specification register 1	ADS1		R/W	-	O	-	00H
FF82H	Power fail detector value comparison mode register	PFM		R/W	-	O	-	00H
FF83H	Power fail detector threshold value setting register	PFT		R/W	-	O	-	00H
FF84H	On Emulator for power-fail detection	DAM0		R/W	O	O	-	00H
FFA0H	Asynchronous serial interface mode register	ASIM0		R/W	O	O	-	00H
FFA1H	Asynchronous serial interface status register	ASIS0		R/W	-	O	-	00H
FFA2H	Baud rate generator control register	BRGC0		R/W	-	O	-	00H
FFB0H	LCD display mode register	LCDM		R/W	O	O	-	00H
FFB2H	LCD clock control register	LCDC		R/W	O	O	-	00H
FFE0H	Interrupt request flag register	IF0	IF0L	R/W	O	O	O	00H
FFE1H	Interrupt request flag register		IF0H	R/W	O	O		00H
FFE2H	Interrupt request flag register	IF1L		R/W	O	O	O	00H
FFE4H	Interrupt mask flag register	MK0	MK0L	R/W	O	O	O	FFH
FFE5H	Interrupt mask flag register		MK0H	R/W	O	O		FFH
FFE6H	Interrupt mask flag register	MK1L		R/W	O	O	-	FFH
FFE8H	Priority flag specification register	PR0	PR0L	R/W	O	O	O	FFH
FFE9H	Priority flag specification register		PR0H	R/W	O	O		FFH
FFEAH	Priority flag specification register	PR1L		R/W	O	O	-	00H
FFF0H	Internal memory size switching register	IMS		R/W	-	O	-	CFH
FFF4H	Internal extended RAM size switching register	IXS		R/W	-	O	-	0CH
FFF9H	Watchdog timer mode register	WDTM		R/W	O	O	-	00H
FFFAH	Oscillation stabilisation time select register	OSTS		R/W	-	O	-	04H
FFFBH	Processor clock control register	PCC		R/W	O	O	-	04H

3.3 Instruction Address Addressing

An instruction address is determined by program counter (PC) contents. The PC contents are normally incremented (+1 for each byte) automatically according to the number of bytes of an instruction to be fetched each time another instruction is executed. However, when a branch instruction is executed, the branch destination information is set to the PC and branched by the following addressing. (For details of instructions, refer to **78K/0 User's Manual - Instructions (U12326E)**.

3.3.1 Relative addressing

The value obtained by adding 8-bit immediate data (displacement value: jdisp8) of an instruction code to the start address of the following instruction is transferred to the program counter (PC) and branched.

The displacement value is treated as signed two's complement data (-128 to +127) and bit 7 becomes a sign bit.

In other words, the range of branch in relative addressing is between -128 and +127 of the start address of the following instruction. This function is carried out when the BR $addr16 instruction or a conditional branch instruction is executed.

Figure 3-17: Relative Addressing
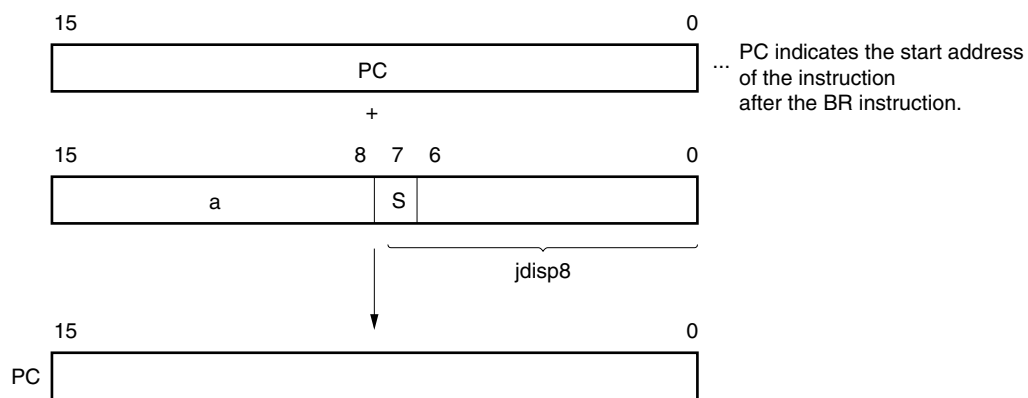
When S = 0, all bits of a are 0.
When S = 1, all bits of a are 1.

3.3.2 Immediate addressing

Immediate data in the instruction word is transferred to the program counter (PC) and branched.

This function is carried out when the CALL !addr16 or BR !addr16 or CALLF !addr11 instruction is executed.

CALL !addr16 and BR !addr16 instructions can branch to all the memory space.

CALLF !addr11 instruction branches to the area from 0800H to 0FFFH.

Figure 3-18: Immediate Addressing
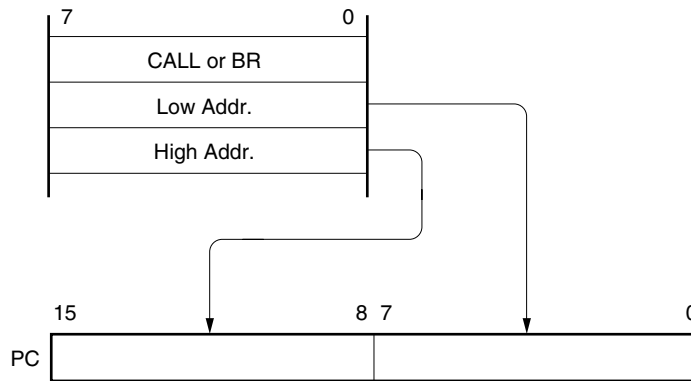
In the case of CALL !addr16 and BR !addr16 instructions
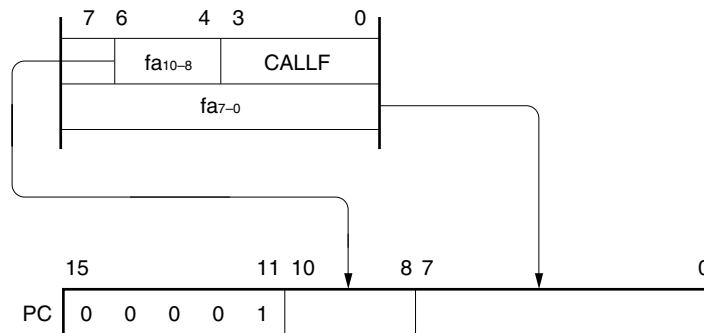
In the case of CALLF !addr11 instruction

3.3.3 Table indirect addressing

Table contents (branch destination address) of the particular location to be addressed by bits 1 to 5 of the immediate data of an operation code are transferred to the program counter (PC) and branched.

Table indirect addressing is carried out when the CALLT [addr5] instruction is executed. This instruction can refer to the address stored in the memory table 40H to 7FH and branch to all the memory space.

Figure 3-19: Table Indirect Addressing
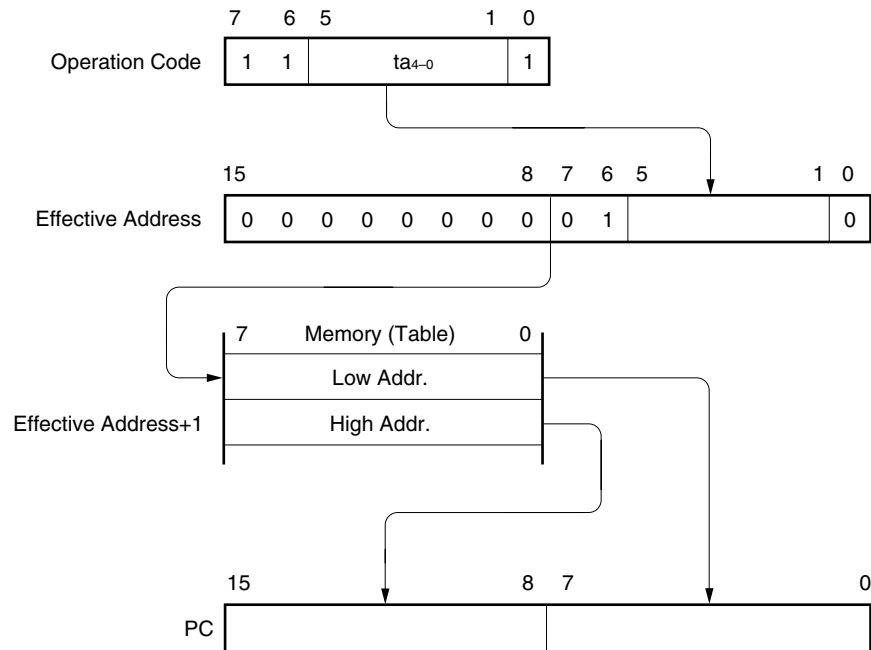
3.3.4 Register addressing

Register pair (AX) contents to be specified with an instruction word are transferred to the program counter (PC) and branched.

This function is carried out when the BR AX instruction is executed.

Figure 3-20: Register Addressing
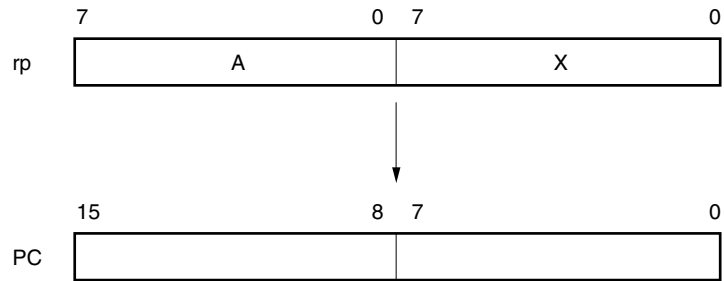
3.4 Operand Address Addressing

The following methods are available to specify the register and memory (addressing) which undergo manipulation during instruction execution.

3.4.1 Implied addressing

The register which functions as an accumulator (A and AX) in the general register is automatically (implicitly) addressed.

Table 3-4: Implied Addressing

Instruction	Register to be Specified by Implied Addressing
MULU	A register for multiplicant and AX register for product storage
DIVUW	AX register for dividend and quotient storage
ADJBA/ADJBS	A register for storage of numeric values which become decimal correction targets
ROR4/ROL4	A register for storage of digit data which undergoes digit rotation

Operand format

Because implied addressing can be automatically employed with an instruction, no particular operand format is necessary.

Description example

In the case of MULU X

With an 8-bit x 8-bit multiply instruction, the product of A register and X register is stored in AX. In this example, the A and AX registers are specified by implied addressing.

3.4.2 Register addressing

The general register is accessed as an operand. The general register to be accessed is specified with register bank select flags (RBS0 and RBS1) and register specify code (Rn, RPn) in the instruction code.

Register addressing is carried out when an instruction with the following operand format is executed.

When an 8-bit register is specified, one of the eight registers is specified with 3 bits in the operation code.

Table 3-5: Register Addressing

Operand format

Identifier	Description
r	X, A, C, B, E, D, L, H
rp	AX, BC, DE, HL

'r' and 'rp' can be described with function names (X, A, C, B, E, D, L, H, AX, BC, DE and HL) as well as absolute names (R0 to R7 and RP0 to RP3).

Description example

Figure 3-21: Register Addressing

```
Operation code    | 0  1  1  0  0  0  1  0 |
                                  └──┴──┘
                                       └──────── Register specify code
```
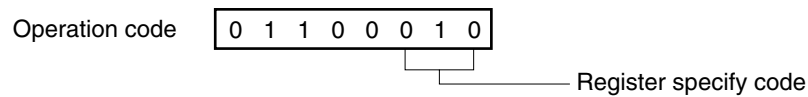
MOV A, C;   when selecting C register as r

```
Operation code    | 1  0  0  0  0  1  0  0 |
                                └──┴──┘
                                     └──────── Register specify code
```
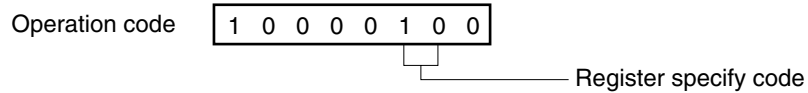
INCW DE;   when selecting DE register pair as rp

3.4.3 Direct addressing

The memory indicated by immediate data in an instruction word is directly addressed.

Operand format

Table 3-6: Direct Addressing

Identifier	Description
addr16	Label or 16-bit immediate data

Description example

MOV A, !0FE00H; when setting !addr16 to FE00H

Operation code | 1 0 0 0 1 1 1 0 | OP code

| 0 0 0 0 0 0 0 0 | 00H

| 1 1 1 1 1 1 1 0 | FEH

3.4.4 Short direct addressing

The memory to be manipulated in the fixed space is directly addressed with 8-bit data in an instruction word.

The fixed space to which this addressing is applied to is the 256-byte space, from FE20H to FF1FH. An internal high-speed RAM and a special function register (SFR) are mapped at FE20H to FEFFH and FF00H to FF1FH, respectively.

The SFR area where short direct addressing is applied (FF00H to FF1FH) is a part of the SFR area. In this area, ports which are frequently accessed in a program, a compare register of the timer/event counter, and a capture register of the timer/event counter are mapped and these SFRs can be manipulated with a small number of bytes and clocks.

When 8-bit immediate data is at 20H to FFH, bit 8 of an effective address is set to 0. When it is at 00H to 1FH, bit 8 is set to 1. Refer to Figure 3-16 below.

Operand format

Table 3-7: Short Direct Addressing

Identifier	Description
saddr	Label of FE20H to FF1FH immediate data
saddrp	Label of FE20H to FF1FH immediate data (even address only)

Description example

MOV 0FE30H, #50H; when setting saddr to FE30H and immediate data to 50H.

Operation code   `0 0 0 1 0 0 0 1`   OP code

`0 0 1 1 0 0 0 0`   30H (saddr-offset)

`0 1 0 1 0 0 0 0`   50H (immediate data)
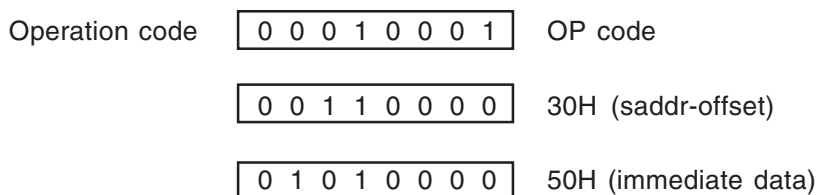
Illustration

Figure 3-22: Short Direct Addressing
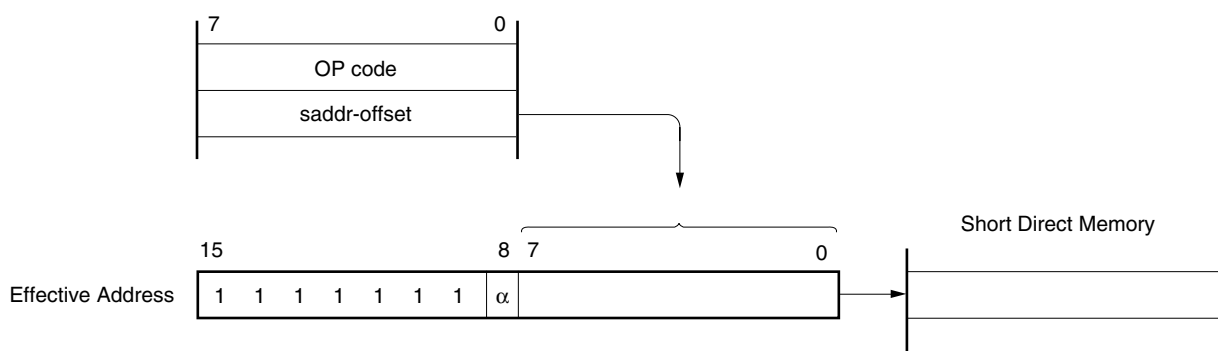
When 8-bit immediate data is 20H to FFH, α = 0
When 8-bit immediate data is 00H to 1FH, α = 1

3.4.5 Special function register (SFR) addressing

The memory-mapped special function register (SFR) is addressed with 8-bit immediate data in an instruction word.

This addressing is applied to the 240-byte spaces FF00H to FFCFH and FFE0H to FFFFH. However, the SFR mapped at FF00H to FF1FH can be accessed with short direct addressing.

Operand format

Table 3-8: Special-Function Register (SFR) Addressing

Identifier	Description
sfr	Special-function register name
sfrp	16-bit manipulatable special-function register name (even address only)

Description example

MOV PM0, A; when selecting PM0 (FE20H) as sfr

Operation code | 1 1 1 1 0 1 1 0 | OP code

| 0 0 1 0 0 0 0 0 | 20H (sfr-offset)
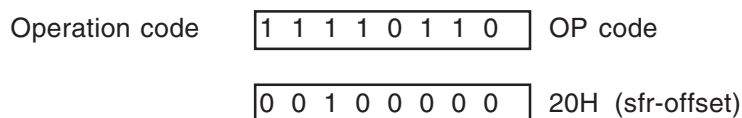
Illustration

Figure 3-23: Special-Function Register (SFR) Addressing
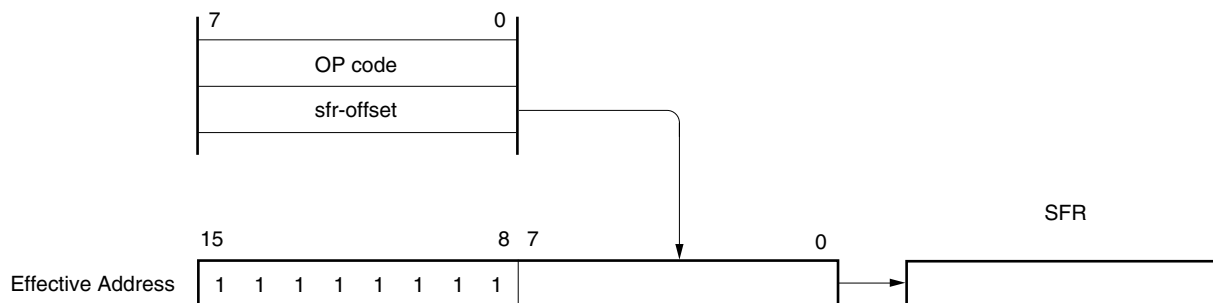
3.4.6 Register indirect addressing

The memory is addressed with the contents of the register pair specified as an operand. The register pair to be accessed is specified with the register bank select flag (RBS0 and RBS1) and the register pair specify code in the instruction code. This addressing can be carried out for all the memory spaces.

Operand format

Table 3-9: Register Indirect Addressing

Identifier	Description
-	[DE], [HL]

Description example

MOV A, [DE]; when selecting [DE] as register pair

Operation code | 1 0 0 0 0 1 0 1 |

Illustration

Figure 3-24: Special-Function Register (SFR) Addressing
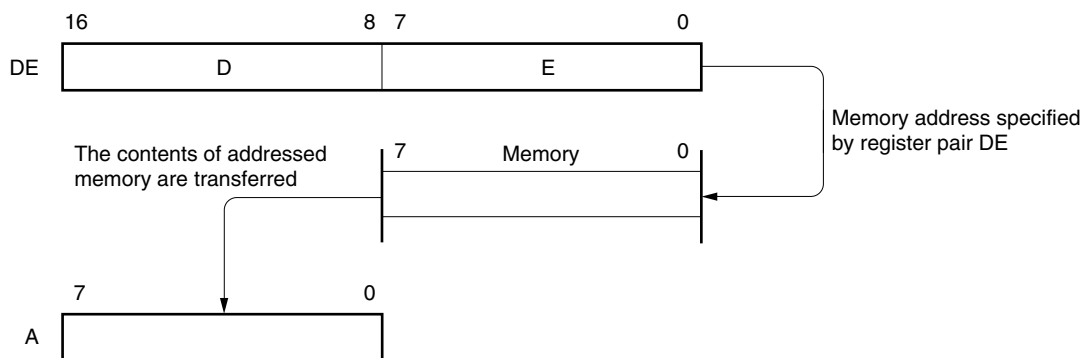
3.4.7 Based addressing

8-bit immediate data is added to the contents of the base register, that is, the HL register pair, and the sum is used to address the memory. The HL register pair to be accessed is in the register bank specified with the register bank select flags (RBS0 and RBS1). Addition is performed by expanding the offset data as a positive number to 16 bits. A carry from the 16th bit is ignored. This addressing can be carried out for all the memory spaces.

Operand format

Table 3-10: Based Addressing

Identifier	Description
-	[HL + byte]

Description example

MOV A, [HL + 10H]; when setting byte to 10H

Operation code | 1 0 1 0 1 1 1 0 |

| 0 0 0 1 0 0 0 0 |

3.4.8 Based indexed addressing

The B or C register contents specified in an instruction are added to the contents of the base register, that is, the HL register pair, and the sum is used to address the memory. The HL, B, and C registers to be accessed are registers in the register bank specified with the register bank select flag (RBS0 and RBS1).

Addition is performed by expanding the contents of the B or C register as a positive number to 16 bits. A carry from the 16th bit is ignored. This addressing can be carried out for all the memory spaces.

Operand format

Table 3-11: Based Indexed Addressing

Identifier	Description
-	[HL + B], [HL + C]

Description example

In the case of MOV A, [HL + B]

Operation code   1 0 1 0 1 0 1 1

3.4.9 Stack addressing

The stack area is indirectly addressed with the stack pointer (SP) contents.

This addressing method is automatically employed when the PUSH, POP, subroutine call and RETURN instructions are executed or the register is saved/reset upon generation of an interrupt request.

Stack addressing enables to address the internal high-speed RAM area only.

Description example

In the case of PUSH DE

Operation code   1 0 1 1 0 1 0 1
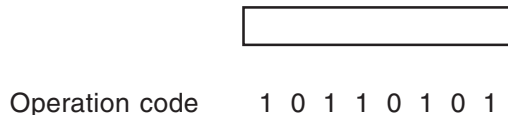
[Memo]

Chapter 4 Port Functions

4.1 Port Functions

The µPD1615A subseries units incorporate four input ports and fifty-three input/output ports. Figure 4-1 shows the port configuration. Every port is capable of 1-bit and 8-bit manipulations and can carry out considerably varied control operations. Besides port functions, the ports can also serve as on-chip hardware input/output pins.

Figure 4-1: Port Types
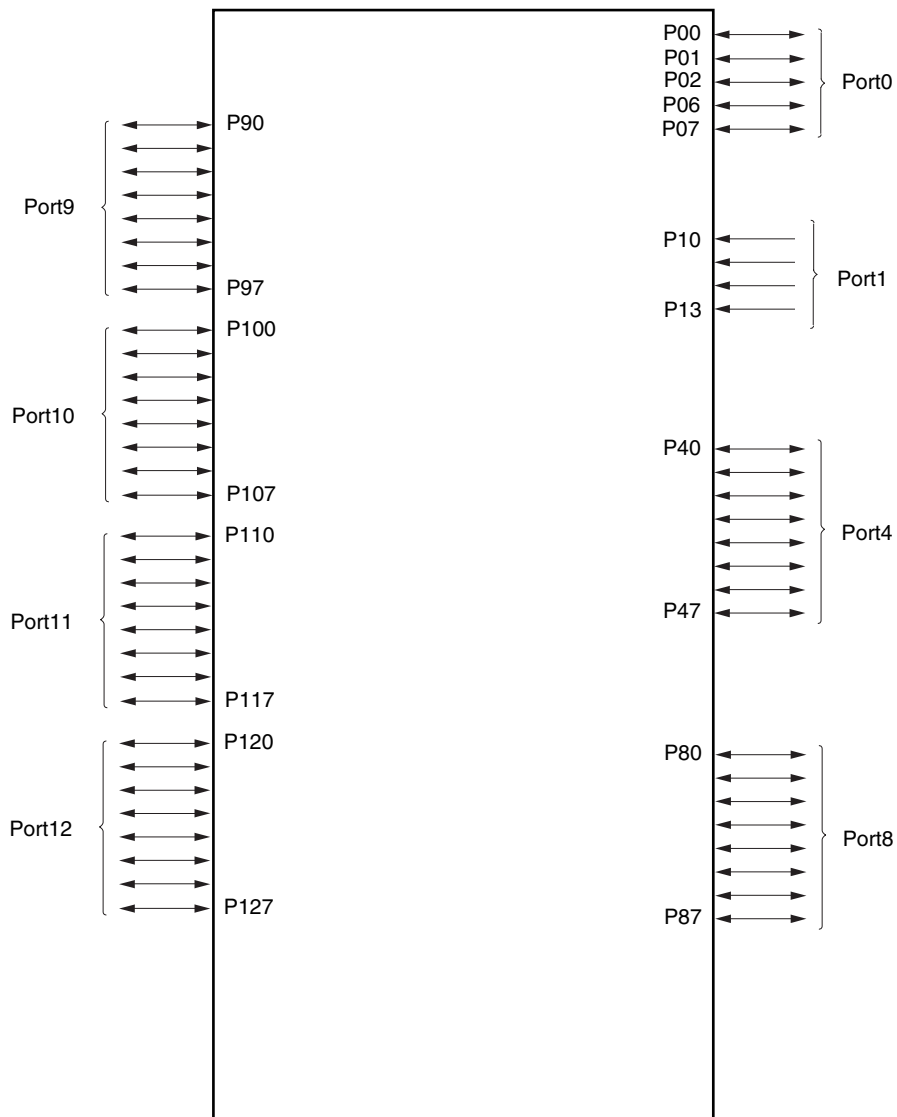
Table 4-1: Pin Input/Output Types µPD1615A(A), µPD1615B(A), µPD1615F(A), µPD16F15A

Input / Output	Pin Name	Function	Alternate Function	After Reset
Input / Output	P00	Port 0 5 bit input / output port Input / output mode can be specified bit-wise	INTP0	Input
	P01		INTP1	Input
	P02		INTP2	Input
	P06		TI50/TO50	Input
	P07		TI51/TO51	Input
Input	P10-P13	Port 1 4 bit input port Input mode can be specified bit-wise	ANI0-ANI3	Input
Input / Output	P40	Port 4 8 bit input/output port Input / output mode can be specified bit-wise	-	Input
	P41		-	Input
	P42		-	Input
	P43		-	Input
	P44		-	Input
	P45		-	Input
	P46		SG0A	Input
	P47		SG0/SG0F	Input
Input/ Output	P80-P87	Port 8 8 bit input / output port Input / output mode can be specified bit-wise This port can be used as segment signal output port or an I/O port in 1-bit units by setting port function register	S39 - S32	Input
Input/ Output	P90-P97	Port 9 8 bit input / output port Input / output mode can be specified bit-wise This port can be used as segment signal output port or an I/O port in 1-bit units by setting port function register	S31 - S24	Input
Input/ Output	P100-P107	Port 10 8 bit input / output port Input / output mode can be specified bit-wise This port can be used as segment signal output port or an I/O port in 1-bit units by setting port function register	S23 - S16	Input
Input/ Output	P110-P117	Port 11 8 bit input / output port Input / output mode can be specified bit-wise This port can be used as segment signal output port or an I/O port in 1-bit units by setting port function register	S15 - S8	Input
Input/ Output	P120	Port 12 8 bit input / output port Input / output mode can be specified bit-wise This port can be used as segment signal output port or an I/O port in 1-bit units by setting port function register	PCL/S7	Input
	P121		TI00/TO0/S6	
	P122		TI01/S5	
	P123		RxD0/S4	
	P124		TxD0/S3	
	P125		SCK3/S2	
	P126		SO3/S1	
	P127		SI3/S0	

Table 4-2: Pin Input/Output Types μPD1616F(A)

Input / Output	Pin Name	Function	Alternate Function	After Reset
Input / Output	P00	Port 0 5 bit input / output port Input / output mode can be specified bit-wise	INTP0	Input
	P01		INTP1	Input
	P02		INTP2	Input
	P06		TI50/TO50	Input
	P07		TI51/TO51	Input
Input	P10-P13	Port 1 4 bit input port Input mode can be specified bit-wise	ANI0-ANI3	Input
Input / Output	P40	Port 4 8 bit input/output port Input / output mode can be specified bit-wise	-	Input
	P41		-	Input
	P42		-	Input
	P43		-	Input
	P44		-	Input
	P45		-	Input
	P46		SG0A	Input
	P47		SG0/SG0F	Input
Input/ Output	P80-P87	Port 8 8 bit input / output port Input / output mode can be specified bit-wise This port can be used as segment signal output port or an I/O port in 1-bit units by setting port function register	-	Input
Input/ Output	P90-P97	Port 9 8 bit input / output port Input / output mode can be specified bit-wise This port can be used as segment signal output port or an I/O port in 1-bit units by setting port function register	-	Input
Input/ Output	P100-P107	Port 10 8 bit input / output port Input / output mode can be specified bit-wise This port can be used as segment signal output port or an I/O port in 1-bit units by setting port function register	-	Input
Input/ Output	P110-P117	Port 11 8 bit input / output port Input / output mode can be specified bit-wise This port can be used as segment signal output port or an I/O port in 1-bit units by setting port function register	-	Input
Input/ Output	P120	Port 12 8 bit input / output port Input / output mode can be specified bit-wise This port can be used as segment signal output port or an I/O port in 1-bit units by setting port function register	PCL	Input
	P121		TI00/TO0	
	P122		TI01	
	P123		RxD0	
	P124		TxD0	
	P125		SCK3	
	P126		SO3	
	P127		SI3	

4.2 Port Configuration

A port consists of the following hardware:

Table 4-3: Port Configuration

Item	Configuration
Control register	Port mode register (PMm: m = 0, 4, 8 to 12) Port function register (PFm: m = 8 to 12)
Port	Total: 57 ports

4.2.1 Port 0

Port 0 is an 5-bit input/output port with output latch. P00 to P02 and P06, P07 pins can be specified as input mode/output mode in 1-bit units with the port mode register 0 (PM0).

Dual-functions include external interrupt request input.

RESET input sets port 0 to input mode.

Figure 4-2 shows block diagram of port 0.

Caution: Because port 0 also supports the external interrupt request input, when the port function output mode is specified and the output level is changed, the interrupt request flag is set. Thus, when the output mode is used, set the interrupt mask flag to 1.

Figure 4-2: P00 to P02 and P06, P07 Configurations
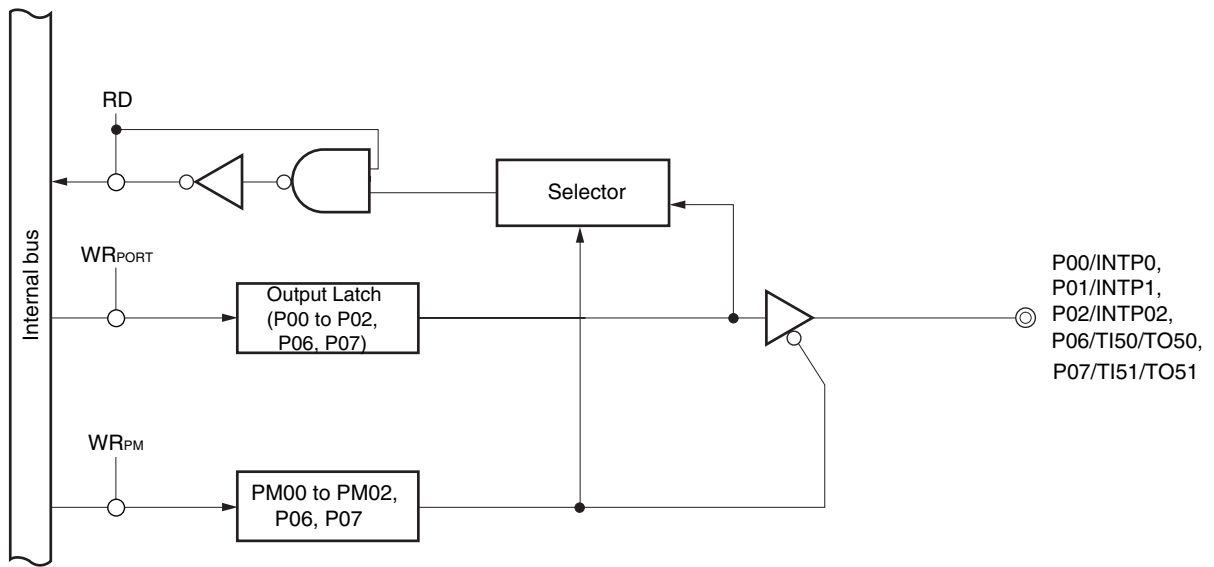
PM : Port mode register

RD : Port 0 read signal

WR : Port 0 write signal

4.2.2 Port 1

Port 1 is a 4-bit input only port.

Dual-functions include an A/D converter analog input.

Figure 4-3 shows a block diagram of port 1.

Figure 4-3: P10 to P13 Configurations
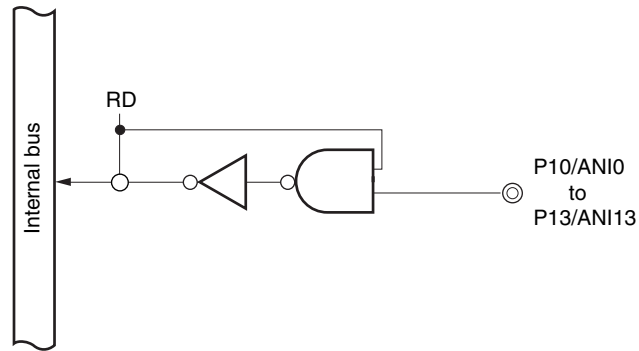
RD : Port 1 read signal

4.2.3 Port 4

Port 4 is an 8-bit input/output port with output latch. P40 to P47 pins can specify the input mode/output mode in 1-bit units.

Dual-function includes the sound generator output.

RESET input sets port 4 to input mode.

Figure 4- 4 shows a block diagram of port 4.

Figure 4-4: P40 to P47 Configurations
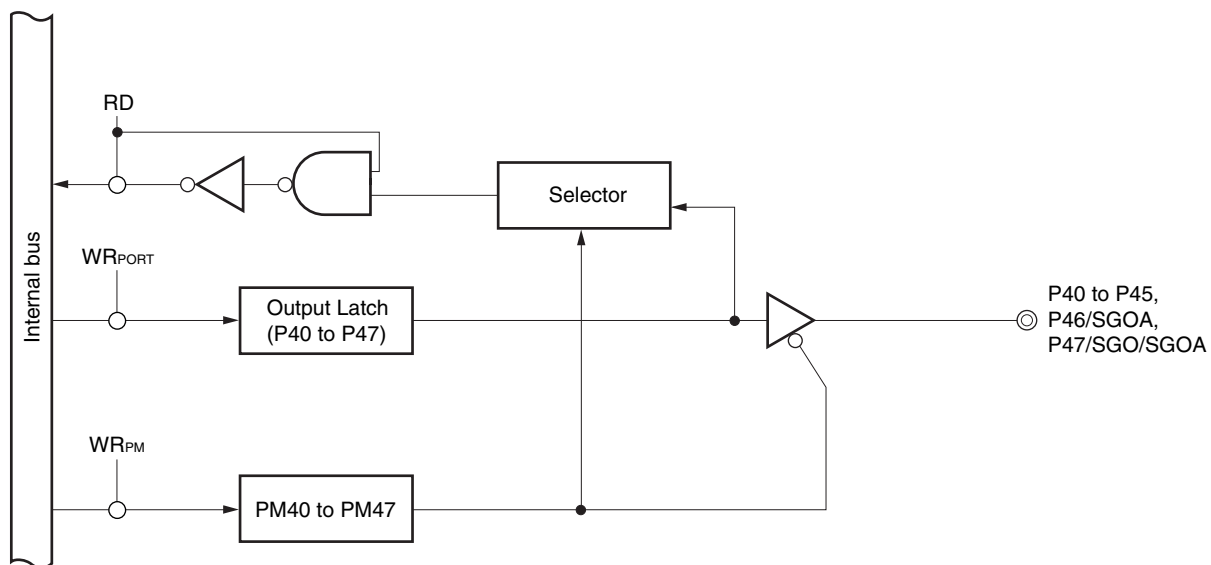
PM : Port mode register

RD : Port 4 read signal

WR : Port 4 write signal

4.2.4 Port 8

Port 8 is an 8-bit input/output port with output latch. P80 to P87 pins can be specified as input mode/output mode in 1-bit units with the port mode register 8 (PM8).

Dual-function includes the segment signal outputs of LCD controller driver. The dual-function can be selected with the port function register 8 (PF8).

RESET input sets port 8 to input mode.

Figure 4-5 shows a block diagram of port 8.

Figure 4-5: P80 to P87 Configurations
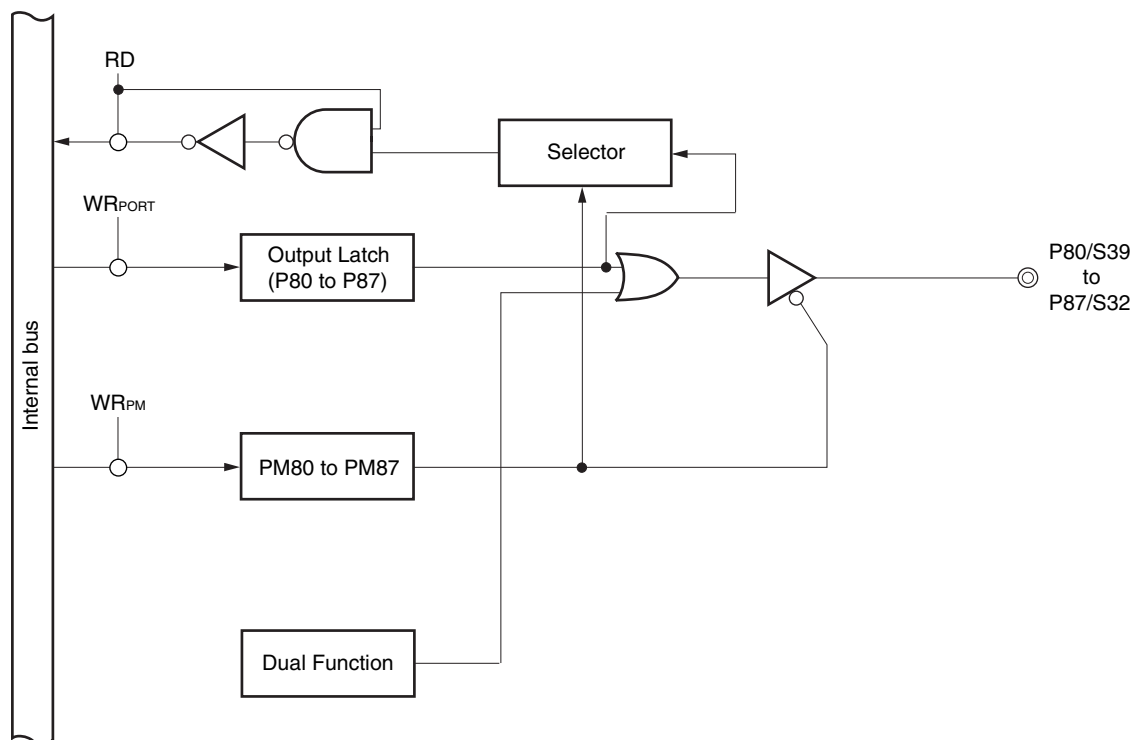
PM : Port mode register
RD   : Port 8 read signal
WR : Port 8 write signal

Note: The LCD controller/driver segment signal output is only valid on the μPD1615 and the μPD16F15.

4.2.5 Port 9

This is an 8-bit input/output port with output latches. Input mode/output mode can be specified in 1-bit units with a port mode register 9.

Dual-function includes the segment signal outputs of LCD controller driver. The dual-function can be specified with the port function register 9 (PF9).

RESET input sets port 9 to input mode.

Figure 4-6 shows a block diagram of port 9.

Caution:　**When used as segment lines, set the port function PF9 according to its functions.**

Figure 4-6: P90 to P97 Configurations
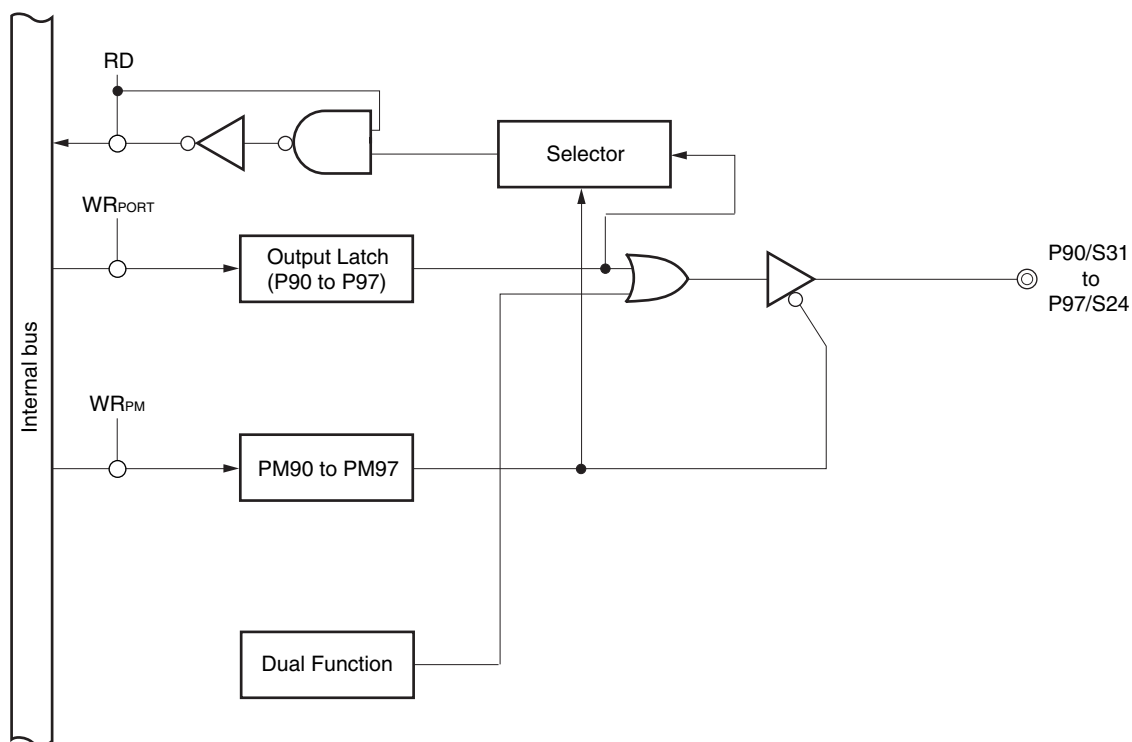
PM　:　Port mode register
RD　:　Port 9 read signal
WR　:　Port 9 write signal

Note:　**The LCD controller/driver segment signal output is only valid on the µPD1615A(A), µPD1615B(A), µPD1615F(A), and the µPD16F15A.**

4.2.6 Port 10

This is an 8-bit input/output port with output latches. Input mode/output mode can be specified in 1-bit units with a port mode register 10.

These pins are dual function pins and serve as segment signal output of LCD controller driver. The dual-function can be specified with the port function register 10 (PF10).

RESET input sets port 10 to input mode.

Figure 4-7 shows a block diagram of port 10.

Caution: When used as segment lines, set the port function PF9 according to its functions.

Figure 4-7: P100 to P107 Configurations
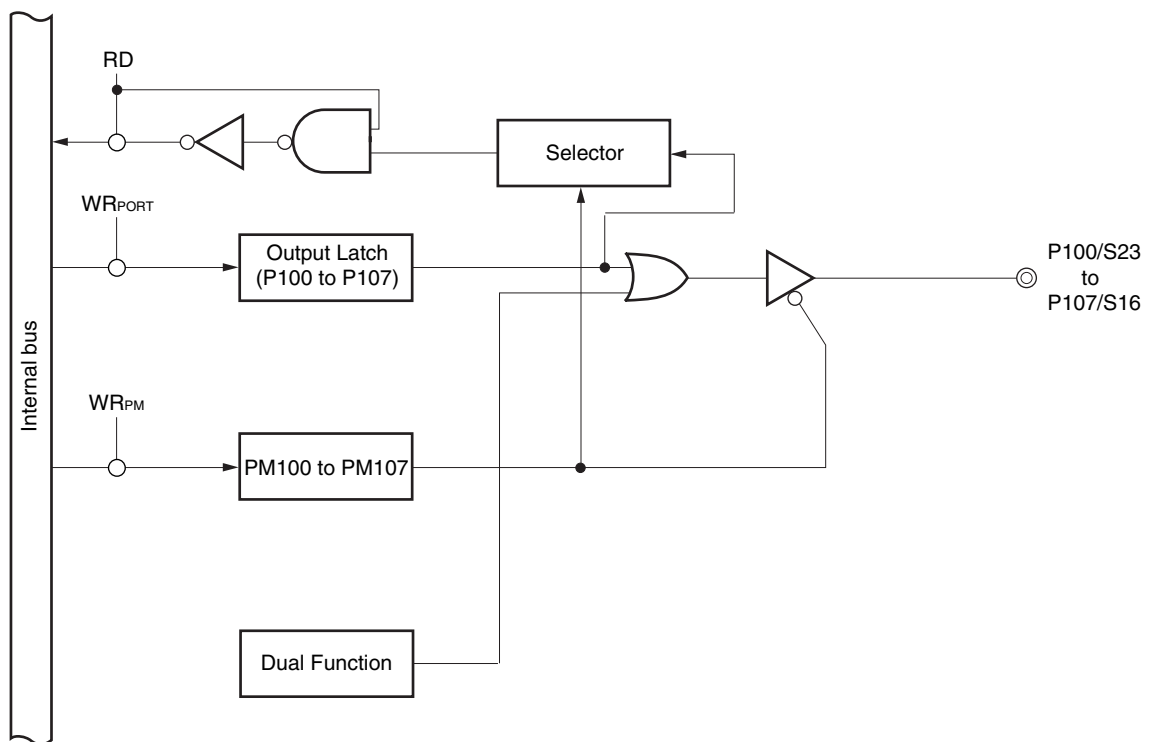
PM : Port mode register
RD   : Port 10 read signal
WR : Port 10 write signal

Note: The LCD controller/driver segment signal output is only valid on the µPD1615A(A), µPD1615B(A), µPD1615F(A), and the µPD16F15A.

4.2.7 Port 11

This is an 8-bit input/output port with output latches. Input mode/output mode can be specified in 1-bit units with a port mode register 11.

These pins are dual function pins and serve as segment signal output of LCD controller driver. The dual-function can be specified with the port function register 11 (PF11).

$\overline{\text{RESET}}$ input sets port 11 to input mode.

Figure 4-8 shows a block diagram of port 11.

Figure 4-8: P110 to P117 Configurations
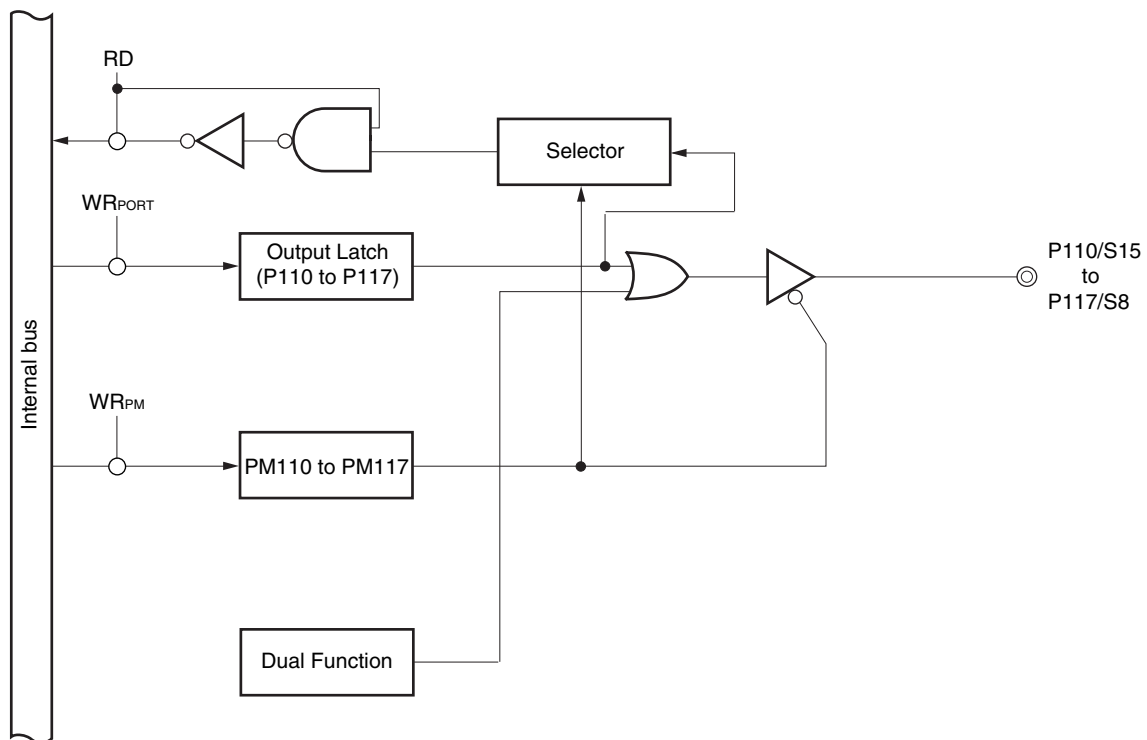

```
PM  :  Port mode register
RD  :  Port 11 read signal
WR  :  Port 11 write signal
```

Note: **The LCD controller/driver segment signal output is only valid on the μPD1615A(A), μPD1615B(A), μPD1615F(A), and the μPD16F15A.**

4.2.8 Port 12

This is an 8-bit input/output port with output latches. Input mode/output mode can be specified in 1-bit units with a port mode register 12.

These pins are dual function pins and serve as segment signal output of LCD controller driver. The dual-function can be specified with the port function register 12 (PF12).

RESET input sets port 12 to input mode.

Figure 4-9 shows a block diagram of port 12.

Figure 4-9: P120 to P127 Configurations
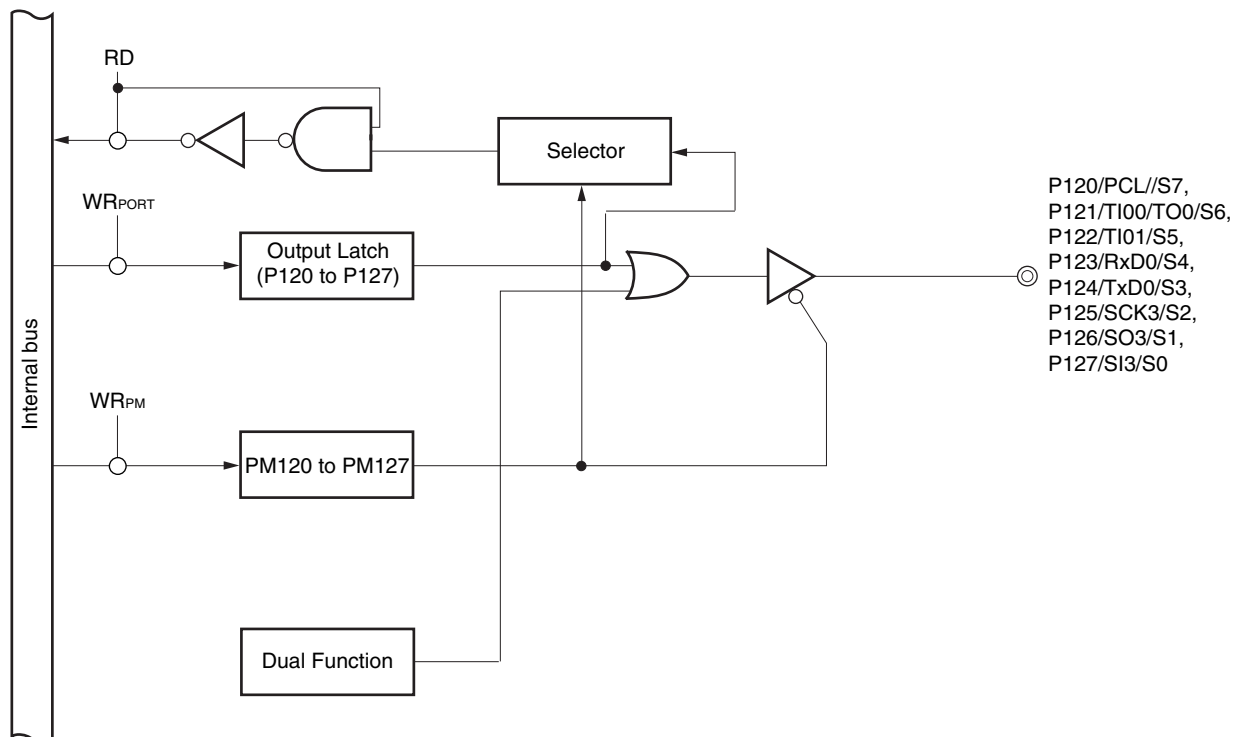
PM : Port mode register
RD : Port 12 read signal
WR : Port 12 write signal

Note: **The LCD controller/driver segment signal output is only valid on the µPD1615A(A), µPD1615B(A), µPD1615F(A), and the µPD16F15A.**

4.3 Port Function Control Registers

The following four types of registers control the ports.
- Port mode registers (PM0, PM4, PM8 to PM12)
- Port function registers (PFm : m = 8 to 12)

(1) Port mode registers (PM0, PM4, PM8 to PM12)
These registers are used to set port input/output in 1-bit units.
PM0, PM4, PM7, PM10 and PM12 are independently set with a 1-bit or 8-bit memory manipulation instruction.
RESET input sets registers to FFH.

When port pins are used as alternate-function pins, set the port mode register and output latch according to the function.

Cautions: 1. Pins P10 to P13 are input-only pins.
2. As port 0 has an alternate function as external interrupt request input, when the port function output mode is specified and the output level is changed, the interrupt request flag is set. When the output mode is used, therefore, the interrupt mask flag should be set to 1 beforehand.

Figure 4-10: Port Mode Register Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
PM0	PM07	PM06	1	1	1	PM02	PM01	PM00	FF20H	FFH	R/W
PM4	PM47	PM46	PM45	PM44	PM43	PM42	PM41	PM40	FF24H	FFH	R/W
PM8	PM87	PM86	PM85	PM84	PM83	PM82	PM81	PM80	FF28H	FFH	R/W
PM9	PM97	PM96	PM95	PM94	PM93	PM92	PM91	PM90	FF29H	FFH	R/W
PM10	PM107	PM106	PM105	PM104	PM103	PM102	PM101	PM100	FF2AH	FFH	R/W
PM11	PM117	PM116	PM115	PM114	PM113	PM112	PM111	PM110	FF2BH	FFH	R/W
PM12	PM127	PM126	PM125	PM124	PM123	PM122	PM121	PM120	FF2CH	FFH	R/W

PMmn	PMmn Pin Input/Output Mode Selection (m = 0 - 4, 8, 12; n = 0 - 7)
0	Output mode (output buffer ON)
1	Input mode (output buffer OFF)

3) Port function register (PF8 to PF12)

This register is used to set LCD segment function of ports 8 to 12.

PF8 to PF12 are set with an 1-bit or 8-bit manipulation instruction.

$\overline{RESET}$ input set this registors to 00H.

Figure 4-11: Port Function Register (PF8 to PF12) Format

PF8	PF87	PF86	PF85	PF84	PF83	PF82	PF81	PF80	FF58H	00H	R/W
PF9	PF97	PF96	PF95	PF94	PF93	PF92	PF91	PF90	FF59H	00H	R/W
PF10	PF107	PF106	PF105	PF104	PF103	PF102	PF101	PF100	FF5AH	00H	R/W
PF11	PF117	PF116	PF115	PF114	PF113	PF112	PF111	PF110	FF5BH	00H	R/W
PF12	PF127	PF126	PF125	PF124	PF123	PF122	PF121	PF120	FF5CH	00H	R/W

PMmn	PFmn Port Function Selection (m = 8 to 12; n = 0 to 7)
0	Port function
1	LCD segment function

Caution: For μPD1616F(A) it is only allowed to set 00h to the port function register.

4.4 Port Function Operations

Port operations differ depending on whether the input or output mode is set, as shown below.

4.4.1 Writing to input/output port
(1) Output mode
A value is written to the output latch by a transfer instruction, and the output latch contents are output from the pin.
Once data is written to the output latch, it is retained until data is written to the output latch again.

(2) Input mode
A value is written to the output latch by a transfer instruction, but since the output buffer is OFF, the pin status does not change.
Once data is written to the output latch, it is retained until data is written to the output latch again.

Caution: **In the case of 1-bit memory manipulation instruction, although a single bit is manipulated the port is accessed as an 8-bit unit. Therefore, on a port with a mixture of input and output pins, the output latch contents for pins specified as input are undefined except for the manipulated bit.**

4.4.2 Reading from input/output port
(1) Output mode
The output latch contents are read by a transfer instruction. The output latch contents do not change.

(2) Input mode
T he pin status is read by a transfer instruction. The output latch contents do not change.

4.4.3 Operations on input/output port
(1) Output mode
An operation is performed on the output latch contents, and the result is written to the output latch. The output latch contents are output from the pins.
Once data is written to the output latch, it is retained until data is written to the output latch again.

(2) Input mode
The output latch contents are undefined, but since the output buffer is OFF, the pin status does not change.

Caution: **In the case of 1-bit memory manipulation instruction, although a single bit is manipulated the port is accessed as an 8-bit unit. Therefore, on a port with a mixture of input and output pins, the output latch contents for pins specified as input are undefined, even for bits other than the manipulated bit.**

[Memo]

Chapter 5 Clock Generator

5.1 Clock Generator Functions

The clock generator generates the clock to be supplied to the CPU and peripheral hardware. The following two types of system clock oscillators are available.

(1) Main system clock oscillator

This circuit oscillates at frequencies of 3.9 to 8.38 MHz. Oscillation can be stopped by executing the STOP instruction or setting the processor clock control register.

(2) Subsystem clock oscillator

The circuit oscillates at a typical frequency of 40 KHz. Oscillation cannot be stopped.

5.2 Clock Generator Configuration

The clock generator consists of the following hardware.

Table 5-1: Clock Generator Configuration

Item	Configuration
Control register	Processor clock control register (PCC)
Oscillator	Main system clock oscillator Subsystem clock oscillator

Figure 5-1: Block Diagram of Clock Generator
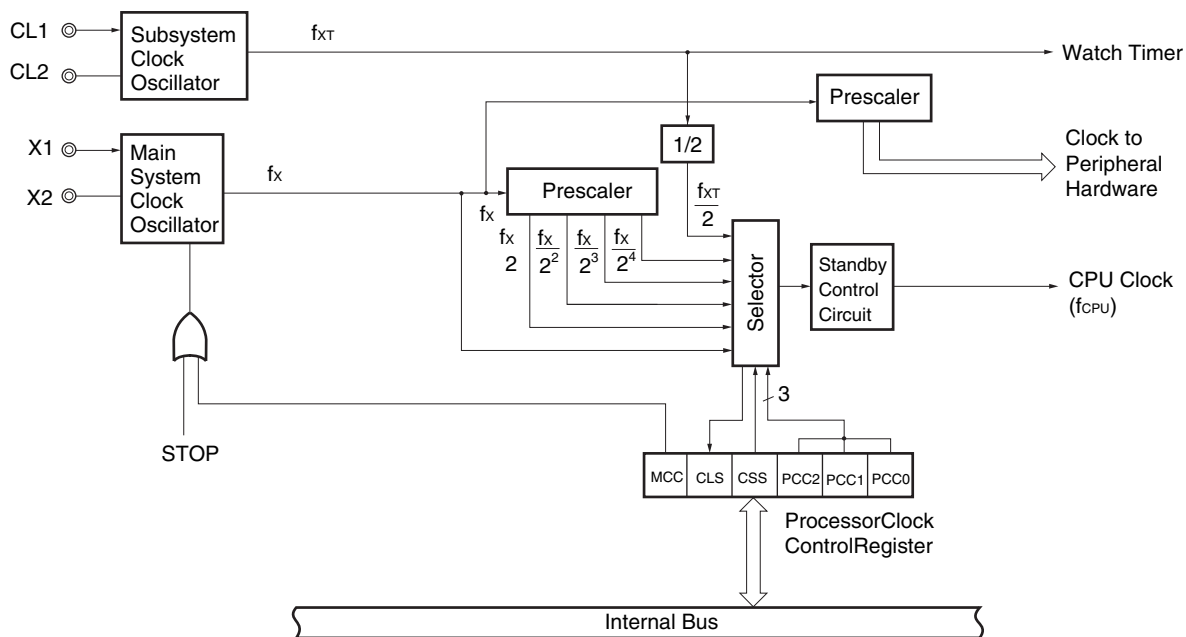
5.3 Clock Generator Control Register

The clock generator is controlled by the processor clock control register (PCC).

(1) Processor clock control register (PCC)

The PCC selects a CPU clock and the division ratio, determines whether to make the main system clock oscillator operate or stop.

The PCC is set with a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets the PCC to 04H.

Figure 5-2: Processor Clock Control Register Format

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

R/W	CSS	PCC2	PCC1	PCC0	CPU Clock Selection (fCPU)
		0	0	0	fx (0.25 μs)
		0	0	1	fx/2 (0.5 μs)
	0	0	1	0	fx/2^2 (1 μs)
		0	1	1	fx/2^3 (2 μs)
		1	0	0	fx/2^4 (4 μs)
		0	0	0	
		0	0	1	
	1	0	1	0	fxT/2 (122 μs)
		0	1	1	
		1	0	0	
	Other than above				Setting prohibited

R	CLS	CPU Clock Status
	0	Main system clock
	1	Subsystem clock

R/W	MCC	Main System Clock Oscillation Control
	0	Oscillation enable
	1	Oscillation stopped

Notes: 1. Bit 5 is a read-only bit.
2. When the CPU is operating on the subsystem clock, MCC should be used to stop the main system clock oscillation. A STOP instruction should not be used.

Cautions: 1. Bit 3 must be set to 0.
2. When external clock input is used MCC should not be set, because the X2 pin is connected to V_DD via a resistor.

Remarks: 1. fx   : Main system clock oscillation frequency
2. fxT   : Subsystem clock oscillation frequency
3. Figures in parentheses indicate minimum instruction execution time: 2fCPU when operating at fx = 8.0 MHz or fxT = 32.768 kHz.

5.4 System Clock Oscillator

5.4.1 Main system clock oscillator
The main system clock oscillator oscillates with a crystal resonator or a ceramic resonator (standard: 8.0 MHz) connected to the X1 and X2 pins.
External clocks can be input to the main system clock oscillator. In this case, the clock signal to the X1 pin and an inversed phase clock signal to the X2 pin.
Figure 5-3 shows an external circuit of the main system clock oscillator.

Figure 5-3: External Circuit of Main System Clock Oscillator

(a) Crystal and ceramic oscillation   **(b) External clock**
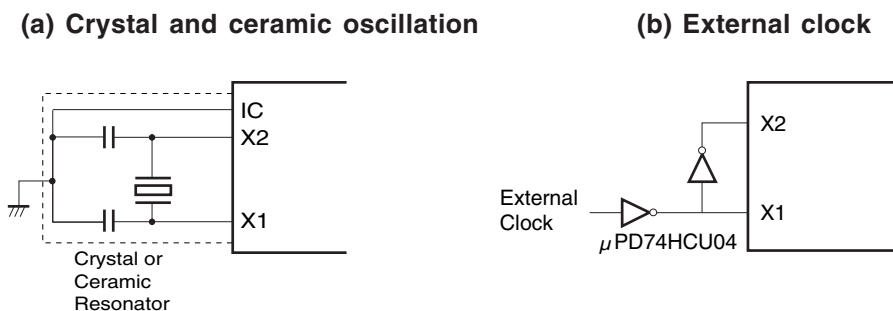
Caution: **Do not execute the STOP instruction and do not set MCC [bit 7 of processor clock control register (PCC)] to 1 if an external clock is input. This is because when the STOP instruction or MCC is set to 1, the main system clock operation stops and the X2 pin is connected to V$_{DD1}$ via a pull-up resistor.**

5.4.2 Subsystem clock oscillator

The subsystem clock oscillator oscillates with a RC-resonator (standard: 40kHz) connected to the CL1 and CL2 pins.

External clocks can be input to the subsystem clock oscillator. In this case, input a clock signal to the CL1 pin and open the CL2 pin.

Figure 5-4 shows an external circuit of the subsystem clock oscillator.

Figure 5-4: External Circuit of Subsystem Clock Oscillator
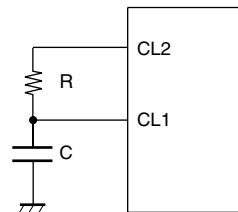
(a) RC oscillation   **(b) External clock**
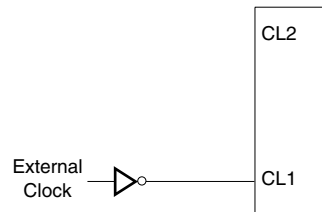
Caution: **When using a main system clock oscillator and a subsystem clock oscillator, carry out wiring in the broken-line area in Figures 6-3 and 6-4 as follows to prevent any effects from wiring capacities.**

- **Minimize the wiring length.**
- **Do not allow wiring to intersect with other signal conductors. Do not allow wiring to come near abruptly changing high current.**
- **Set the potential of the grounding position of the oscillator capacitor to that of Vss. Do not ground to any ground pattern where high current is present.**
- **Do not fetch signals from the oscillator.**

Take special note of the fact that the subsystem clock oscillator is a circuit with low-level amplification so that current consumption is maintained at low levels.

Figure 5-5 shows examples of oscillator having bad connection.

Figure 5-5: Examples of Oscillator with Bad Connection (1/3)

(a) Wiring of connection circuits is too long

(b) A signal line crosses over oscillation circuit lines
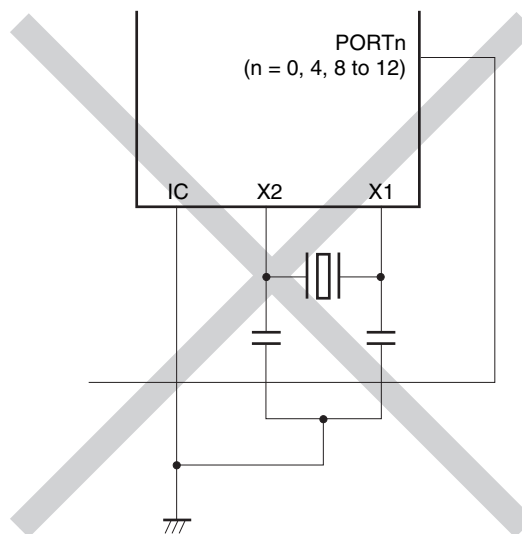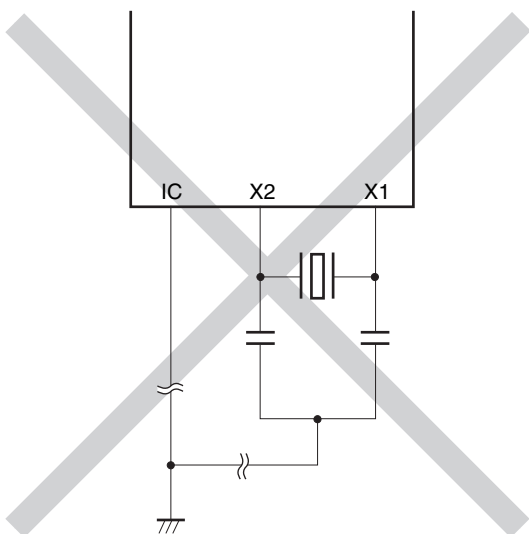
Figure 5-5: Examples of Oscillator with Bad Connection (2/3)

(c) Changing high current is too near a signal conductor

(d) Current flows through the grounding line of the oscillator (potential at points A, B, and C fluctuate)
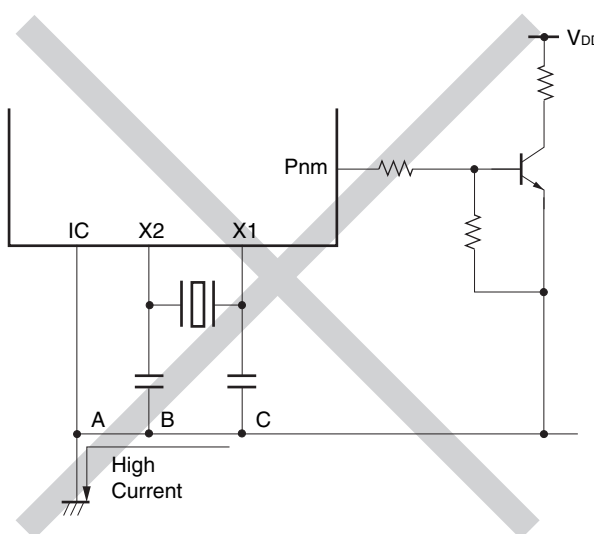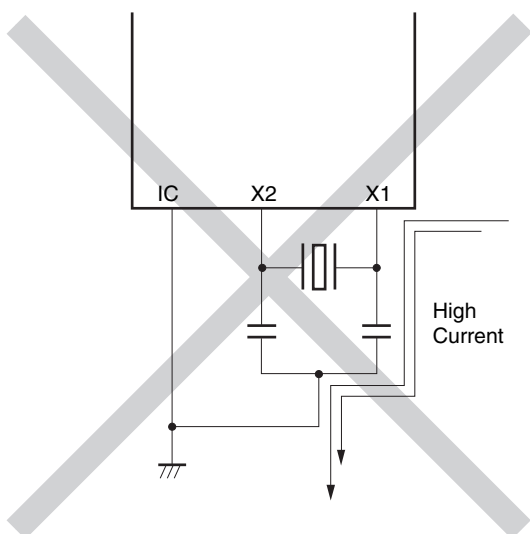
Figure 5-5: Examples of Oscillator with Bad Connection (3/3)

(e) Signals are fetched
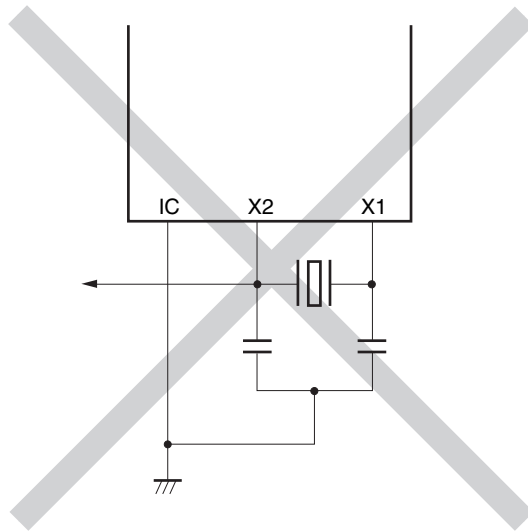
(f) Signal conductors of the main and sub-system clock are parallel and near each other

CL1 and CL2 are wiring in parallel
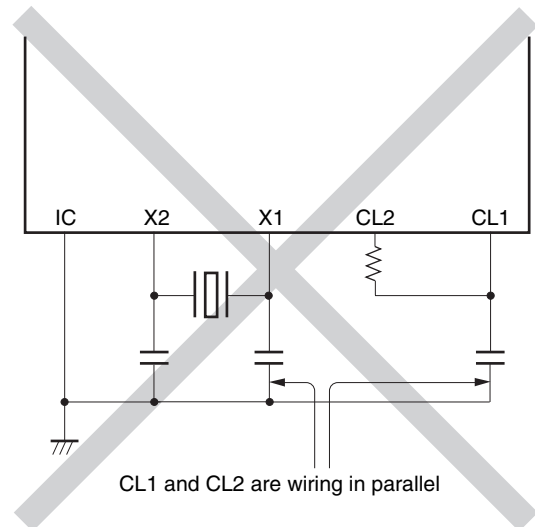
Caution: In Figure 5-5 (f), CL1 and X1 are wired in parallel. Thus, the cross-talk noise of X1 may increase with CL1, resulting in malfunctioning. To prevent that from occurring, it is recommended to wire CL1 and X1 so that they are not in parallel, and to connect the IC pin between CL1 and X1 directly to Vss.

5.4.3 When no subsystem clocks are used

If it is not necessary to use subsystem clocks for low power consumption operations and clock operations, connect the CL1 and CL2 pins as follows.

CL1: Connect to VDD or GND
CL2: Open

5.5 Clock Generator Operations

The clock generator generates the following various types of clocks and controls the CPU operating mode including the standby mode.

- Main system clock f_X
- Subsystem clock f_{XT}
- CPU clock f_{CPU}
- Clock to peripheral hardware

The following clock generator functions and operations are determined with the processor clock control register (PCC).

(a) Upon generation of $\overline{\text{RESET}}$ signal, the lowest speed mode of the main system clock (4 μs when operated at 8.0 MHz) is selected (PCC = 04H). Main system clock oscillation stops while low level is applied to $\overline{\text{RESET}}$ pin.

(b) With the main system clock selected, one of the five CPU clock stages (f_X, $f_X/2$, $f_X/2^2$, $f_X/2^3$ or $f_X/2^4$) can be selected by setting the PCC.

(c) With the main system clock selected, two standby modes, the STOP and HALT modes, are available.

(d) The PCC can be used to select the subsystem clock and to operate the system with low current consumption (122 μs when operated at 32.768 kHz).

(e) With the subsystem clock selected, main system clock oscillation can be stopped with the PCC. The HALT mode can be used. However, the STOP mode cannot be used. (Subsystem clock oscillation cannot be stopped.)

5.5.1 Main system clock operations

When operated with the main system clock (with bit 5 (CLS) of the processor clock control register (PCC) set to 0), the following operations are carried out by PCC setting.

(a) Because the operation guarantee instruction execution speed depends on the power supply voltage, the instruction execution time can be changed by bits 0 to 2 (PCC0 to PCC2) of the PCC.

(b) If bit 7 (MCC) of the PCC is set to 1 when operated with the main system clock, the main system clock oscillation does not stop. When bit 4 (CSS) of the PCC is set to 1 and the operation is switched to subsystem clock operation (CLS = 1) after that, the main system clock oscillation stops (see Figure 5-6).

Figure 5-6: Main System Clock Stop Function (1/2)

(a) Operation when MCC is set after setting CSS with main system clock operation
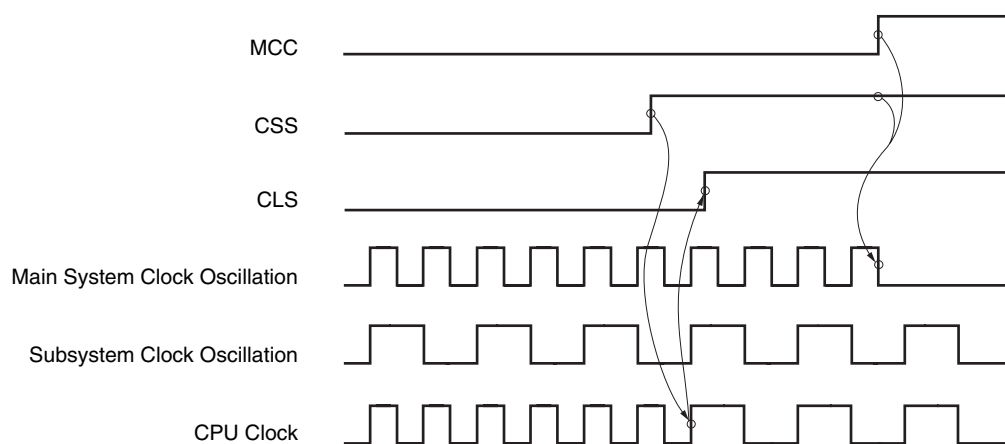
(b) Operation when MCC is set in case of main system clock operation
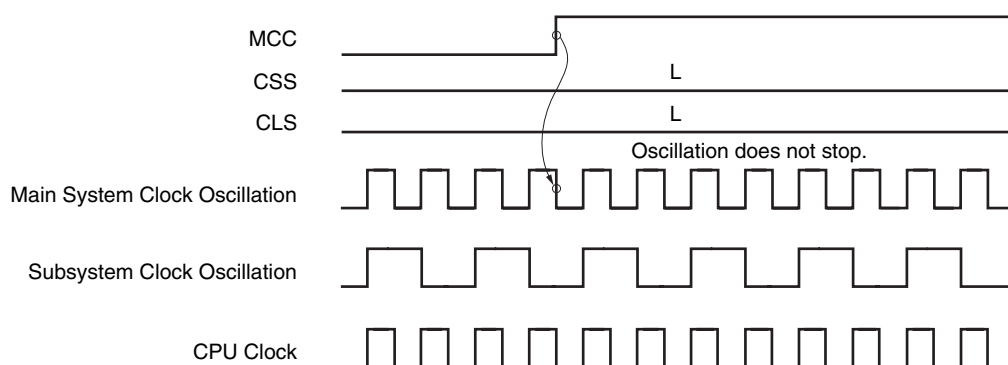
Figure 5-6: Main System Clock Stop Function (2/2)

(c) Operation when CSS is set after setting MCC with main system clock operation
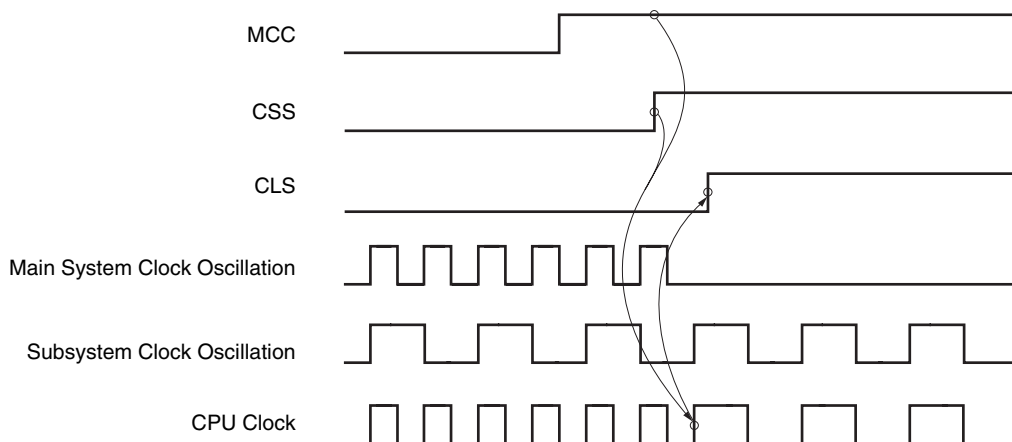
5.5.2 Subsystem clock operations

When operated with the subsystem clock (with bit 5 (CLS) of the processor clock control register (PCC) set to 1), the following operations are carried out.

(a) The instruction execution time remains constant (122 μs when operated at 32.768 kHz) irrespective of bits 0 to 2 (PCC0 to PCC2) of the PCC.

(b) Watchdog timer counting stops.

Caution: Do not execute the STOP instruction while the subsystem clock is in operation.

5.6 Changing System Clock and CPU Clock Settings

5.6.1 Time required for switchover between system clock and CPU clock

 The system clock and CPU clock can be switched over by means of bit 0 to bit 2 (PCC0 to PCC2) and bit 4 (CSS) of the processor clock control register (PCC).
 The actual switchover operation is not performed directly after writing to the PCC, but operation continues on the pre-switchover clock for several instructions (see Table 5-2).
 Determination as to whether the system is operating on the main system clock or the subsystem clock is performed by bit 5 (CLS) of the PCC register.

Table 5-2: Maximum Time Required for CPU Clock Switchover

Set Values after Switchover					Set Values before Switchover																						
MCS	CSS	PCC2	PCC1	PCC0	CSS	PCC2	PCC1	PCC0	CSS	PCC2	PCC1	PCC0	CSS	PCC2	PCC1	PCC0	CSS	PCC2	PCC1	PCC0	CSS	PCC2	PCC1	PCC0	CSS	PCC2	PCC1	PCC0
					0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0	1	X	X	X
X	0	0	0	0					8 instructions				4 instructions				2 instructions				1 instruction				1 instruction			
		0	0	1	16 instructions								4 instructions				2 instructions				1 instruction				1 instruction			
		0	1	0	16 instructions				8 instructions								2 instructions				1 instruction				1 instruction			
		0	1	1	16 instructions				8 instructions				4 instructions								1 instruction				1 instruction			
		1	0	0	16 instructions				8 instructions				4 instructions				2 instructions								1 instruction			
1	1	X	X	X	$f_X/2f_{XT}$ instruction (77 instructions)				$f_X/4f_{XT}$ instruction (39 instructions)				$f_X/8f_{XT}$ instruction (20 instructions)				$f_X/16f_{XT}$ instruction (10 instructions)				$f_X/32f_{XT}$ instruction (5 instructions)							
0					$f_X/4f_{XT}$ instruction (39 instructions)				$f_X/8f_{XT}$ instruction (20 instructions)				$f_X/16f_{XT}$ instruction (10 instructions)				$f_X/32f_{XT}$ instruction (5 instructions)				$f_X/64f_{XT}$ instruction (3 instructions)							

Caution: **Selection of the CPU clock cycle scaling factor (PCC0 to PCC2) and switchover from the main system clock to the subsystem clock (changing CSS from 0 to 1) should not be performed simultaneously. Simultaneous setting is possible, however, for selection of the CPU clock cycle scaling factor (PCC0 to PCC2) and switchover from the subsystem clock to the main system clock (changing CSS from 1 to 0).**

Remarks: **1.** One instruction is the minimum instruction execution time with the pre-switchover CPU clock.

5.6.2 System clock and CPU clock switching procedure

This section describes switching procedure between system clock and CPU clock.

Figure 5-7: System Clock and CPU Clock Switching
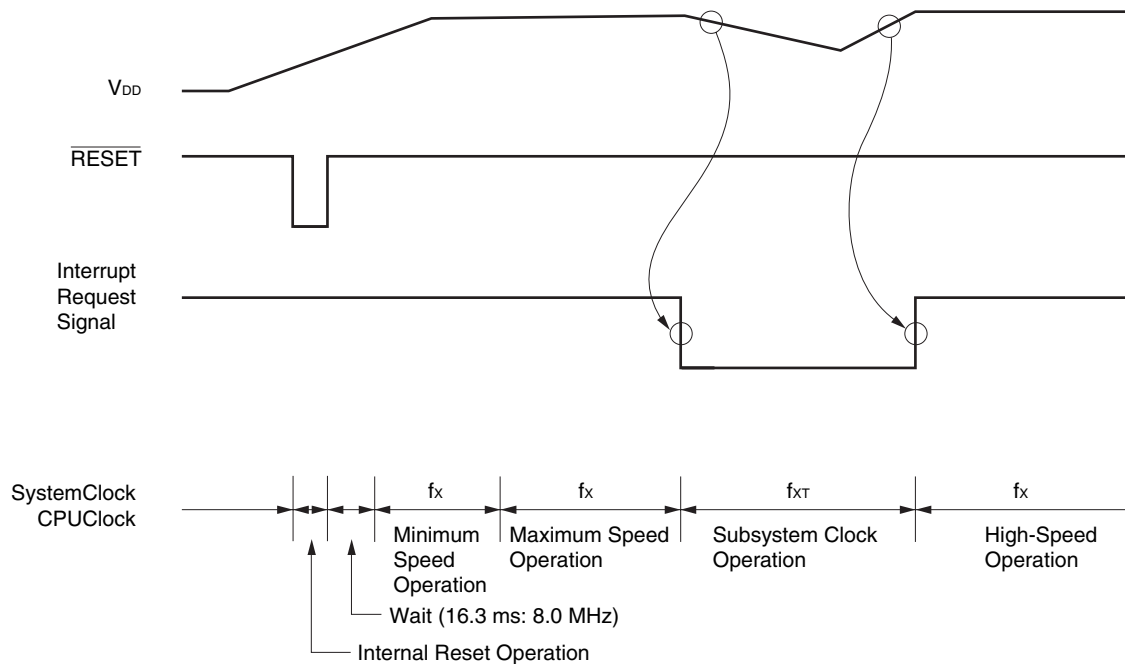
(1) The CPU is reset by setting the $\overline{\text{RESET}}$ signal to low level after power-on. After that, when reset is released by setting the $\overline{\text{RESET}}$ signal to high level, main system clock starts oscillation. At this time, oscillation stabilization time ($2^{17}/f_X$) is secured automatically.
After that, the CPU starts executing the instruction at the minimum speed of the main system clock (4 μs when operated at 8.0 MHz).

(2) After the lapse of a sufficient time for the V_{DD} voltage to increase to enable operation at maximum speeds, the processor clock control register (PCC) is rewritten and the maximum-speed operation is carried out.

(3) Upon detection of a decrease of the V_{DD} voltage due to an interrupt request signal, the main system clock is switched to the subsystem clock (which must be in an oscillation stable state).

(4) Upon detection of V_{DD} voltage reset due to an interrupt request signal, 0 is set to bit 7 (MCC) of PCC and oscillation of the main system clock is started. After the lapse of time required for stabilization of oscillation, the PCC is rewritten and the maximum-speed operation is resumed.

Caution: **When subsystem clock is being operated while main system clock was stopped, if switching to the main system clock is made again, be sure to switch after securing oscillation stable time by software.**

[Memo]

Chapter 6 16-Bit Timer/ Event Counter

6.1 16-bit Timer/Event Counter Function

16-bit timer/event counter (TM0) has the following functions:

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

(1) Interval timer
When 16-bit timer/event counter is used as an interval timer, it generates an interrupt request at predetermined time intervals.

(2) PPG output
16-bit timer/event counter can output a square wave whose frequency and output pulse width can be freely set.

(3) Pulse width measurement
16-bit timer/event counter can be used to measure the pulse width of a signal input from an external source.

(4) External event counter
16-bit timer/event counter can be used to measure the number of pulses of a signal input from an external source.

(5) Square wave output
16-bit timer/event counter can output a square wave any frequency.

6.2 16-bit Timer/Event Counter Configuration

16-bit timer/event counter (TM0) consists of the following hardware:

Table 6-1: Configuration of 16-bit Timer/Event Counter (TM0)

Item	Configuration
Timer register	16 bits x 1 (TM0)
Register	Capture/compare register: 16 bits x 2 (CR00, CR01)
Timer output	1 (TO0)
Control register	16-bit timer mode control register (TMC0) Capture/compare register 0 (CRC0) 16-bit timer output control register (TOC0) Prescaler mode register 0 (PRM0) Port mode register 12 (PM12)

Figure 6-1: Block Diagram of 16-Bit Timer/Event Counter (TM0)

115
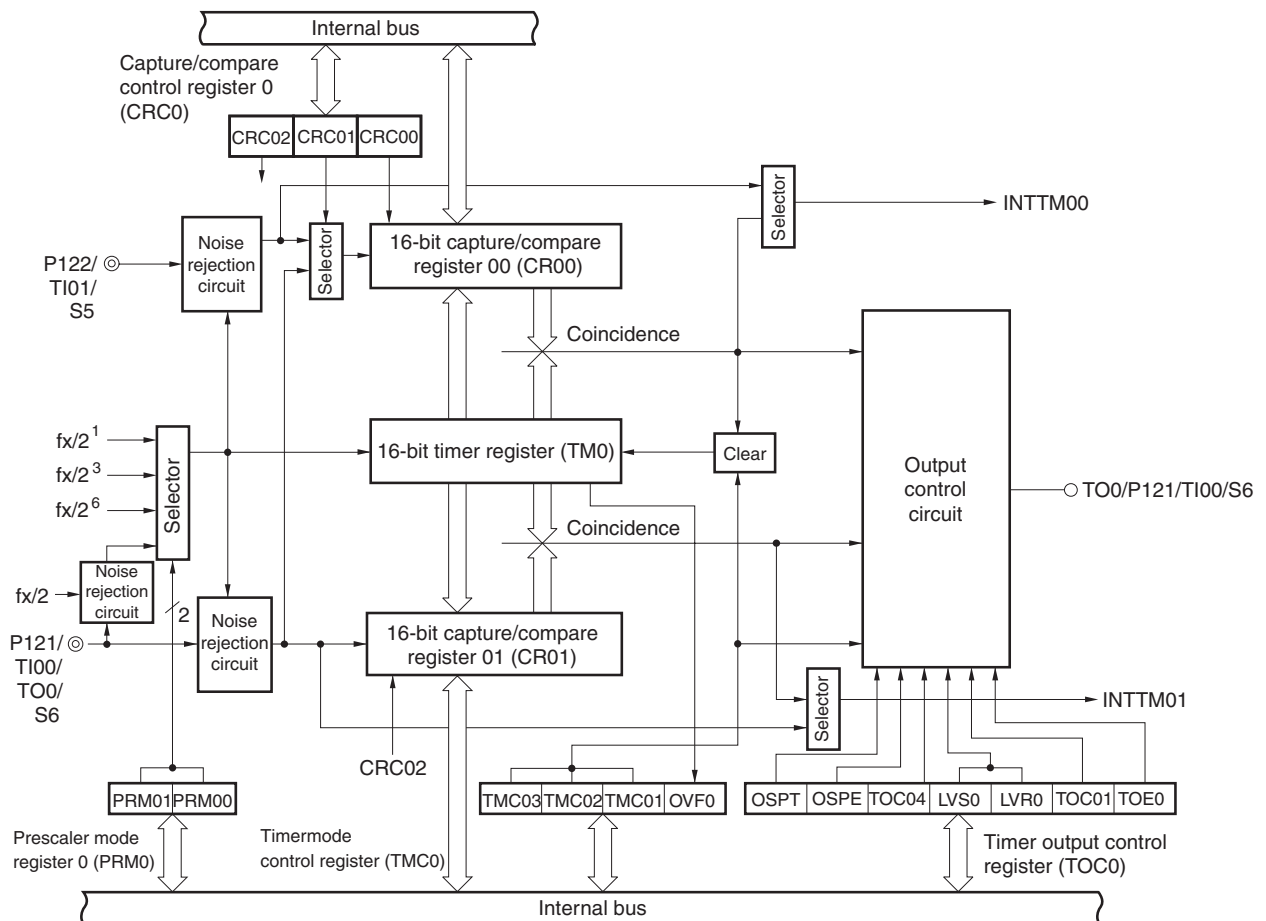
1) 16-bit timer register (TM0)

TM0 is a 16-bit read-only register that counts count pulses.

The counter is incremented in synchronization with the rising edge of an input clock. If the count value is read during operation, input of the count clock is temporarily stopped, and the count value at that point is read. The count value is reset to 0000H in the following cases:

<1> $\overline{\text{RESET}}$ is input.
<2> TMC03 and TMC02 are cleared.
<3> Valid edge of TI00 is input in the clear & start mode by inputting valid edge of TI00.
<4> TM0 and CR00 coincide with each other in the clear & start mode on coincidence between TM0 and CR00.

2) Capture/compare register 00 (CR00)

CR00 is a 16-bit register that functions as a capture register and as a compare register. Whether this register functions as a capture or compare register is specified by using bit 0 (CRC00) of the capture/compare control register 0.

- **When using CR00 as compare register**

  The value set to CR00 is always compared with the count value of the 16-bit timer register (TM0). When the values of the two coincide, an interrupt request (INTTM00) is generated. When TM00 is used as an interval timer, CR00 can also be used as a register that includes the interval time and as register which sets the pulse width in the PPD operation mode.

- **When using CR00 as capture register**

  The valid edge of the TI00 or TI01 pin can be selected as a capture trigger. The valid edge of TI00 and TI01 is performed via the prescaler mode register 0 (PRM0).

  Tables 6-2 and 6-3 show the conditions that apply when the capture trigger is specified as the valid edge of the TI00 pin and the valid edge of the TI01 pin respectively.

Table 6-2: Valid Edge of TI00 Pin and Valid Edge of Capture Trigger of Capture/Compare Register

ES01	ES00	Valid Edge of TI00 Pin	Capture Trigger of CR00	Capture Trigger of CR01
0	0	Falling edge	Rising edge	Falling edge
0	1	Rising edge	Falling edge	Rising edge
1	0	Setting prohibited	Setting prohibited	Setting prohibited
1	1	Both rising and falling edges	No capture operation	Both rising and falling edges

Table 6-3: Valid Edge of TI01 Pin and Valid Edge of Capture Trigger of Capture/Compare Register

ES01	ES00	Valid Edge of TI01 Pin	Capture Trigger of CR00
0	0	Falling edge	Rising edge
0	1	Rising edge	Falling edge
1	0	Setting prohibited	Setting prohibited
1	1	Both rising and falling edges	Both rising and falling edges

CR00 is set by a 16-bit memory manipulation instruction.
After RESET input, the value of CR00 is undefined.

Cautions: 1. Set a value other than 0000H in CR00. This means 1-pulse count operation cannot be performed when CR00 is used as an event counter. However, in the free-running mode and in the clear mode using the valid edge of TI00, if 0000H is set to CR00, an interrupt request (INTTM00) is generated following overflow (FFFFH).

2. If the new value of CR00 is less than the value of 16-bit timer counter 0 (TM0), TM0 continues counting, overflows, and than starts counting from 0 again. If the new value CR00 is less than the old value, therefore, the timer must be restarted after the value of CR00 is changed.

3) 16-bit capture/compare register 01 (CR01)

CR01 is a 16-bit register which has the functions of both a capture register and a compare register. Whether it is used as a capture register or a compare register is set bit 2 (CRC02) of capture/compare control register 0 (CRC0).

- **When CR01 is used as a compare register**

    The value set in the CR01 is constantly compared with the 16-bit timer counter 0 (TM0) count value, and an interrupt request (INTTM01) is generated if they match.

- **When CR01 is used as a capture register**

    It is possible to select the valid edge of the TI00 pin as the capture trigger. The TI00 valid edge is set by means ofthe prescaler mode register 0 (PRM0).

CR01 is set by a 16-bit memory manipulation instruction.
The value of this register is undefined when $\overline{\text{RESET}}$ is input.

Caution: Set other than 0000H to CR01. This means, that an 1-pulse count operation cannot be performed when CR01 is used as an event counter. However, in the free-running mode and in the clear mode using the valid edge of TI00, if 0000H is set to CR01, an interrupt request (INTTM01) is generated following overflow (FFFFH).

6.3 16-Bit Timer/Event Counter Control Register

The following four types of registers control 16-bit timer/event counter (TM0).

- 16-bit timer mode control register (TMC0)
- Capture/compare control register (CRC0)
- 16-bit timer output control register (TOC0)
- Prescaler mode register 0 (PRM0)
- Port mode register 12 (PM12)

(1) 16-bit timer mode control register (TMC0)

This register specifies the operation mode of the 16-bit timer and the clear mode, output timing, and overflow detection of the 16-bit timer register.

TMC0 is set by a 1-bit or 8-bit memory manipulation instruction.

RESET input sets TMC0 to 00H.

Caution: The 16-bit timer register starts operating when a value other than 0, 0 (operation stop mode) is set to TMC02 and TMC03. To stop the operation, set 0, 0 to TMC02 and TMC03.

Figure 6-2: Format of 16-Bit Timer Mode Control Register (TMC0)

Address: FF60H   After Reset: 00H   R/W

Symbol	7	6	5	4	3	2	1	⓪
TMC0	0	0	0	0	TMC03	TMC02	TMC01	OVF0

TMC03	TMC02	TMC01	Operating Mode Clear mode and clear mode	Selection of TO0 output timing	Generation of interrupt
0	0	0	Operation stop (TM0 is cleared to 0).	Not affected	Does not generate.
0	0	1			
0	1	0	Free running mode	Coincidence between TM0 and CR00 or coincidence between TM0 and CR01	Generates on coincidence between TM0 and CR00 and coincidence between TM0 and CR01.
0	1	1		Coincidence between TM0 and CR00, coincidence between TM0 and CR01, or valid edge of TI00	
1	0	0	Clears and starts at valid edge of TI00.	-	
1	0	1		-	
1	1	0	Clears and starts on coincidence between TM0 and CR00.	Coincidence between TM0 and CR00 or coincidence between TM0 and CR01	
1	1	1		Coincidence between TM0 and CR00, coincidence between TM0 and CR01, or valid edge of TI00	

0VF0	Detection of overflow of 16-bit timer register
0	Overflows.
1	Does not overflow.

Cautions: 1. **Before changing the clear mode and TO0 output timing, be sure to stop the timer operation (reset TMC02 and TMC03 to 0, 0).**

2. **The valid edge of the TI00 pin is selected by using the prescaler mode register 0 (PRM0).**

3. **When a mode in which the timer is cleared and started on coincidence between TM0 and CR00, the OVF0 flag is set to 1 when the count value of TM0 changes from FFFFH to 0000H with CR00 set to FFFFH.**

Remark: T00 : output pin of 16-bit timer/counter (TM0)
TI00 : input pin of 16-bit timer/counter (TM0)
TM0 : 16-bit timer register
CR00: compare register 00
CR01: compare register 01

(2) Capture/compare control register 0 (CRC0)

This register controls the operation of the capture/compare registers (CR00 and CR01).

CRC0 is set by a 1-bit or 8-bit memory manipulation instruction.

RESET input sets CRC0 to 00H.

Figure 6-3: Format of Capture/Compare Control Register 0 (CRC0)

Address: FF62H   After Reset: 00H   R/W

Symbol	7	6	5	4	3	2	1	0
CRC0	0	0	0	0	0	CRC02	CRC01	CRC00

CRC02	Selection of operation mode of CR01
0	Operates as compare register
1	Operates as capture register

CRC01	Selection of capture trigger of CR00
0	Captured at valid edge of TI01
1	Captured in reverse phase of valid edge of TI00

CRC00	Selection of operation mode of CR00
0	Operates as compare register
1	Operates as capture register

Cautions: **1. Before setting CRC0, be sure to stop the timer operation.**
2. When the mode in which the timer is cleared and started on coincidence between TM0 and CR00 is selected by the 16-bit timer mode control register (TMC0), do not specify CR00 as a capture register.
3. If valid edge of TI00 is both falling and rising, the capture operation is not available when CRC01 = 1.
4. To surely perform the capture operation, the capture trigger requires a pulse two times longer than the count clock selected by the prescaler mode register 0 (PRM0).

(3) 16-bit timer output control register (TOC0)

This register controls the operation of the 16-bit timer/event counter (TM0) output control circuit by setting or resetting the R-S flip-flop, enabling or disabling reverse output, enabling or disabling output of 16-bit timer/counter (TM0).

TOC0 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets TOC0 to 00H.

Figure 6-4 shows the format of TOC0.

Figure 6-4: Format of 16-Bit Timer Output Control Register (TOC0)

Address: FF63H   After Reset: 00H   R/W

Symbol	7	6	5	4	③	②	1	⓪
TOC0	0	0	0	TOC04	LVS0	LVR0	TOC01	TOE0

TOC04	Timer output F/F control on coincidence between CR01 and TM0
0	Disables inversion timer output
1	Enables inversion timer output

LVS0	LVR0	Set status of timer output F/F of 16-bit timer/counter (TM0)
0	0	Not affected
0	1	Resets timer output F/F (0)
1	0	Sets timer output F/F (1)
1	1	Setting prohibited

TOC01	Timer output F/F control on coincidence between CR00 and TM0
0	Disables inversion timer output F/F
1	Enables inversion timer output F/F

TOE0	Output control of 16-bit timer/counter (TM0)
0	Disables output (port mode)
1	Enables output

Cautions: 1. Before setting TOC0, be sure to stop the timer operation.
2. LVS0 and LVR0 are 0 when read after data have been set to them.
3. Be sure to set bits 5 to 7 to 0.

(4) Prescaler mode register 0 (PRM0)

This register selects a count clock of the 16-bit timer/event counter (TM0) and the valid edge of TI00, TI01 input. PRM0 is set by an 8-bit memory manipulation instruction.

RESET input sets PRM0 to 00H.

Figure 6-5: Format of Prescaler Mode Register 0 (PRM0)

Address: FF61H   After Reset : 00H   R/W

Symbol	7	6	5	4	3	2	1	0
PRM0	ES11	ES10	ES01	ES00	0	0	PRM01	PRM00

ES11	ES10	Selection of valid edge of TI01
0	0	Falling edge
0	1	Rising edge
1	0	Setting prohibited
1	1	Both falling and rising edges

ES01	ES00	Selection of valid edge of TI00
0	0	Falling edge
0	1	Rising edge
1	0	Setting prohibited
1	1	Both falling and rising edges

PRM01	PRM00	Selection of count clock
0	0	$f_x/2^1$ (4.00 MHz)
0	1	$f_x/2^3$ (1.00 MHz)
1	0	$f_x/2^6$ (125 KHz)
1	1	Valid edge of TI00[Note]

Note: The external clock requires a pulse two times longer than internal count clock ($f_x/2^1$).

Cautions: 1. If the valid edge of TI00 is to be set to the count clock, do not set the clear/start mode and the capture trigger at the valid edge of TI00.
2. Be sure to stop timer operation before setting PRM0.
3. If the TI00 or TI01 in is high level immediately after system reset, the rising edge is immediately detected after the rising edge or both the rising and falling edges are set as the valid edge(s) of the TI00 pin or TI01 pin to enable the operation of the 16-bit timer/counter 0 (TM0). Please be careful when pulling up the TI00 pin or the TI01 pin. However, when re-enabling operation after the operation has been stopped once, the rising edge is not detected.

Remarks: 1. fx: Main system clock operation frequency
2. TI00,TI01: 16-bit timer/event counter 0 input pin
3. Figures in parentheses are for operation with fx = 8.0 MHz.

(5) Port mode register 12 (PM12)

This register sets port 12 input/output in 1-bit units.

When using the P121/TO0/TI00/S6 pin for timer output, set PM121 and the output latch of P121 to 0.

PM12 is set with an 1-bit or 8-bit memory manipulation instruction.

RESET input sets PM12 value to FFH.

Figure 6-6: Port Mode Register 12 (PM12) Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
PM12	PM127	PM126	PM125	PM124	PM123	PM122	PM121	PM120	FF2CH	FFH	R/W

PM12n	P12n pin input/output mode selection (n = 0 to 7)
0	Output mode (output buffer ON)
1	Input mode (output buffer OFF)

(6) Port function register 12 (PM12)

This register sets the port function of port 12 in 1-bit units.

When using the timer for timer output or timer input, the register PF12 has to be set to port function.

PM12 is set with an 1-bit or 8-bit memory manipulation instruction.

RESET input sets PM12 value to 00H.

Figure 6-7: Port Function Register 12 (PM12) Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
PF12	PF127	PF126	PF125	PF124	PF123	PF122	PF121	PF120	FF5CH	00H	R/W

PF12n	P12n function selection (n = 0 to 7)
0	Port mode
1	LCD mode

Note: For the μPD1616 set always 00H to PF12.

6.4 16-Bit Timer/Event Counter Operations

6.4.1 Operation as interval timer (16 bits)

The 16-bit timer/event counter operates as an interval timer when the 16-bit timer mode control register (TMC0) and capture/compare control register 0 (CRC0) are set as shown in Figure 6-8.

In this case, 16-bit timer/event counter repeatedly generates an interrupt at the time interval specified by the count value set in advance to the 16-bit capture/compare register 00 (CR00).

When the count value of the 16-bit timer register (TM0) coincides with the set value of CR00, the value of TM0 is cleared to 0, and the timer continues counting. At the same time, an interrupt request signal (INTTM00) is generated.

The count clock of the 16-bit timer/event counter can be selected by bits 0 and 1 (PRM00 and PRM01) of the prescaler mode register 0 (PRM0).

Figure 6-8: Control Register Settings When Timer 0 Operates as Interval Timer
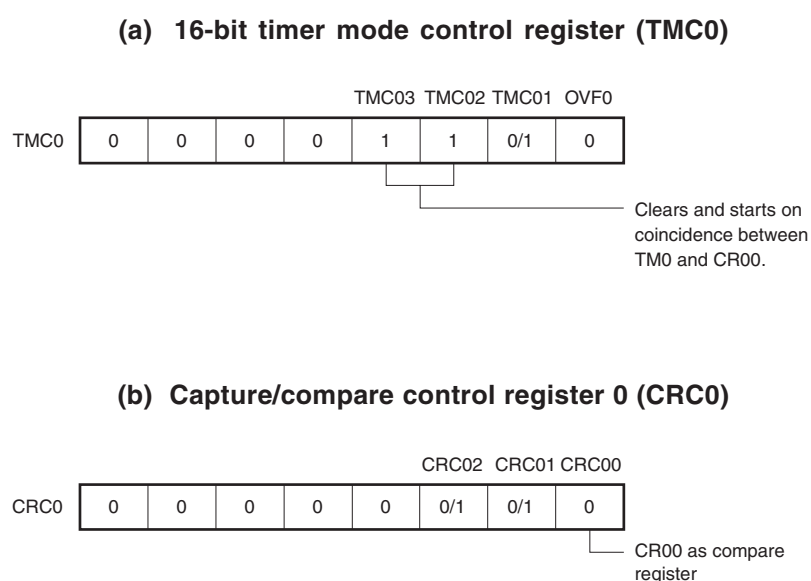
(a) 16-bit timer mode control register (TMC0)

				TMC03	TMC02	TMC01	OVF0
TMC0	0	0	0	0	1	1	0/1	0

Clears and starts on coincidence between TM0 and CR00.

(b) Capture/compare control register 0 (CRC0)

					CRC02	CRC01	CRC00
CRC0	0	0	0	0	0	0/1	0/1	0

CR00 as compare register

Remark: 0/1: When these bits are reset to 0 or set to 1, the other functions can be used along with the interval timer function. For details, refer to Figures 6-2 and 6-3.

Figure 6-9: Configuration of Interval Timer
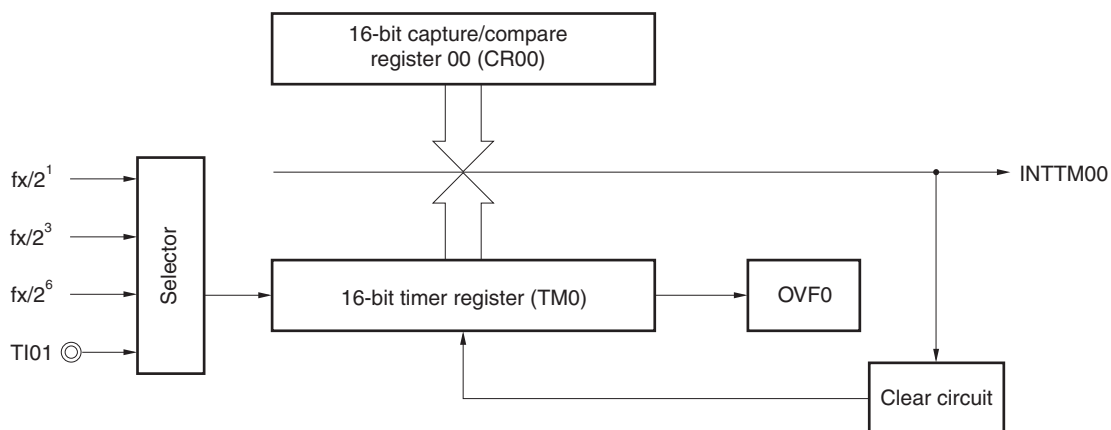
Figure 6-10: Timing of Interval Timer Operation
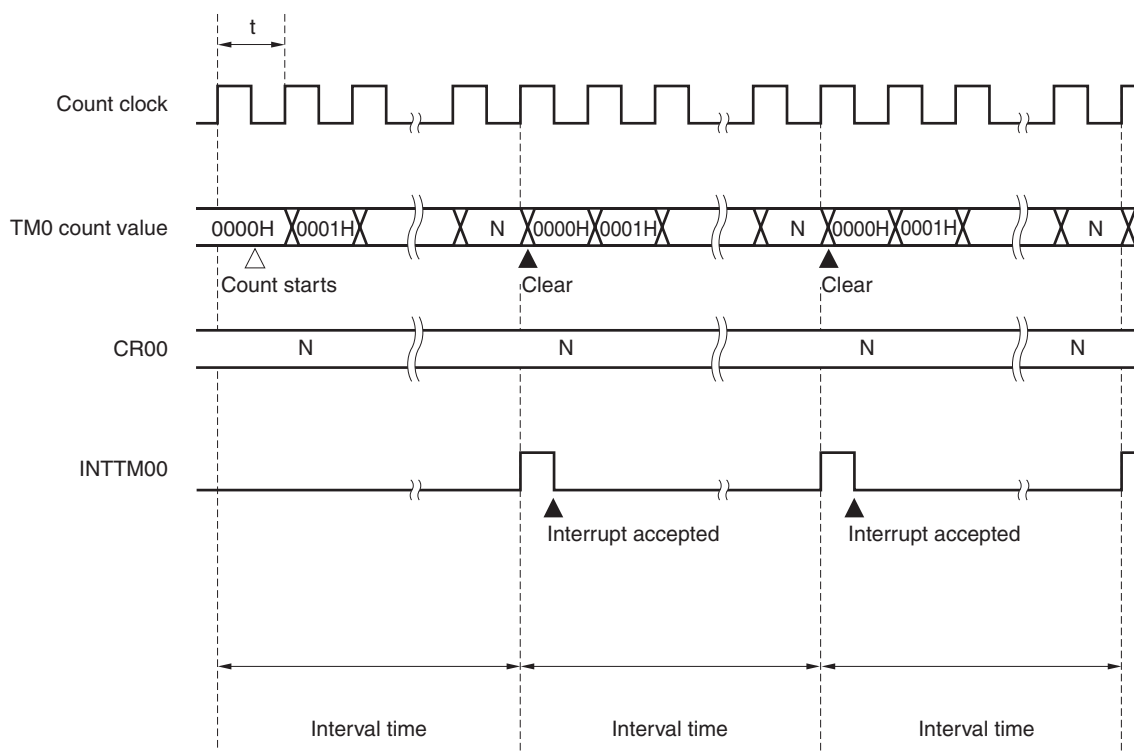
Remark: Interval time = (N+1) x t: N = 0000H to FFFFH

6.4.2 PPG output operation

The 16-bit timer/counter can be used for PPG (Programmable Pulse Generator) output by setting the 16-bit timer mode control register (TMC0) and capture/compare control register 0 (CRC0) as shown in Figure 6-11.

The PPG output function outputs a rectangular wave with a cycle specified by the count value set in advance to the 16-bit capture/compare register 00 (CR00) and a pulse width specified by the count value set in advance to the 16-bit capture/compare register 01 (CR01).

Figure 6-11: Control Register Settings in PPG Output Operation

(a) 16-bit timer mode control register (TMC0)

				TMC03	TMC02	TMC01	OVF0
TMC0	0	0	0	0	1	1	0	0

Clears and starts on coincidence between TM0 and CR00.

(b) Capture/compare control register 0 (CRC0)

						CRC02	CRC01	CRC00
CRC0	0	0	0	0	0	0	X	0

CR00 as compare register

CR01 as compare register

(c) 16-bit timer output control register (TOC0)

	OSPT	OSPE	TOC04	LVS0	LVR0	TOC01	TOE0
TOC0	0	0	0	1	0/1	0/1	1	1

Enables TO0 output

Reverses output on coincidence between TM0 and CR00

Specifies initial value of TO0 output F/F

Reverses output on coincidence between TM0 and CR01

Disables one-shot pulse output

Remark: x   :   don't care

on :   can be used for other functions

Caution: Make sure that 0000H ≤ CR01 < CR00 ≤ FFFFH is set to CR00 and CR01.

6.4.3 Pulse width measurement

The 16-bit timer register (TM0) can be used to measure the pulse widths of the signals input to the TI00 and TI01 pins.

Measurement can be carried out with TM0 used as a free running counter or by restarting the timer in synchronization with the edge of the signal input to the TI00 pin.

(1) Pulse width measurement with free running counter and one capture register

If the edge specified by the prescaler mode register 0 (PRM0) is input to the TI00 pin when the 16-bit timer register (TM0) is used as a free running counter (refer to Figure 6-12), the value of TM0 is loaded to the 16-bit capture/compare register 01 (CR01), and an external interrupt request signal (INTTM01) is set.

The edge is specified by using bits 6 and 7 (ES10 and ES11) of the prescaler mode register 0 (PRM0).

The rising edge, falling edge, or both the rising and falling edges can be selected.

The valid edge is detected through sampling at a count clock cycle selected by the prescaler mode register 0n (PRM0), and the capture operation is not performed until the valid level is detected two times. Therefore, noise with a short pulse width can be rejected.

Figure 6-12: Control Register Settings for Pulse Width Measurement with Free Running Counter and One Capture Register

(a) 16-bit timer mode control register (TMC0)

(b) Capture/compare control register 0 (CRC0)

Remark: 0/1: When these bits are reset to 0 or set to 1, the other functions can be used along with the pulse width measurement function. For details, refer to Figures 6-2 and 6-3.

NEC µPD1615A(A), µPD1615B(A), µPD1615F(A), µPD1616F(A), µPD16F15A
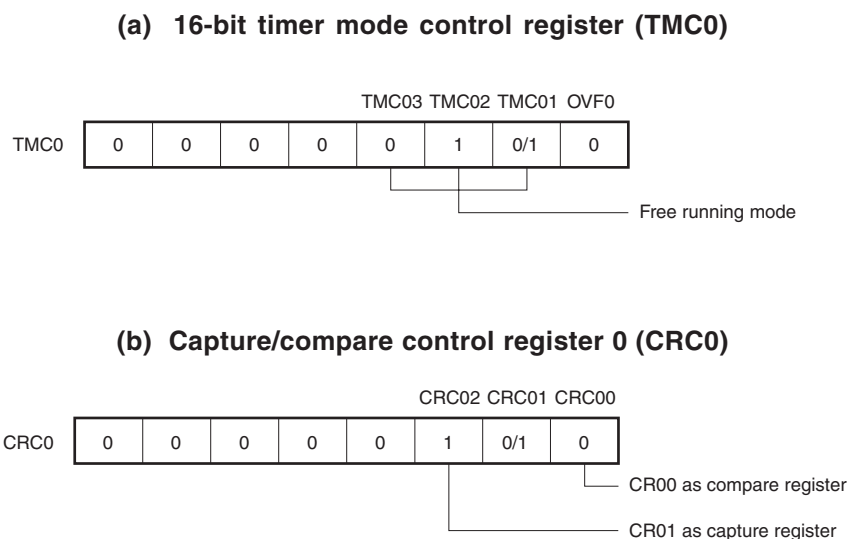
Figure 6-13: Configuration for Pulse Width Measurement with Free Running Counter
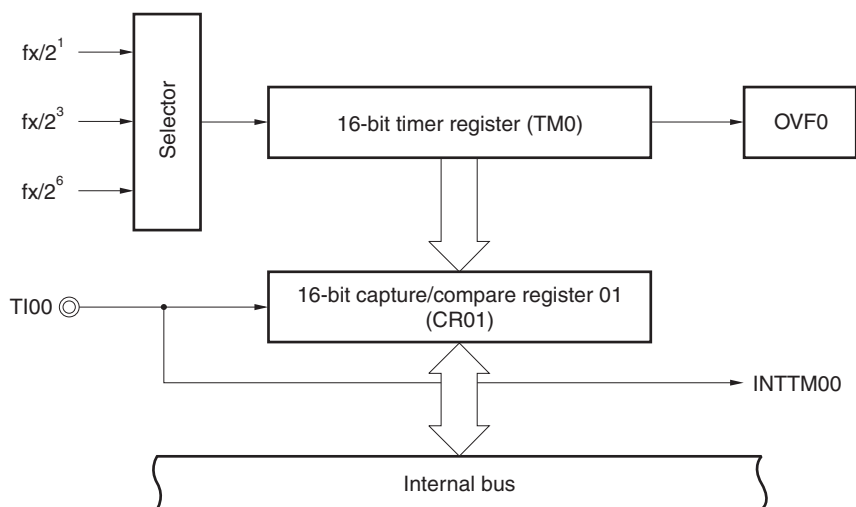
Figure 6-14: Timing of Pulse Width Measurement with Free Running Counter and One Capture Register (with both edges specified)

129
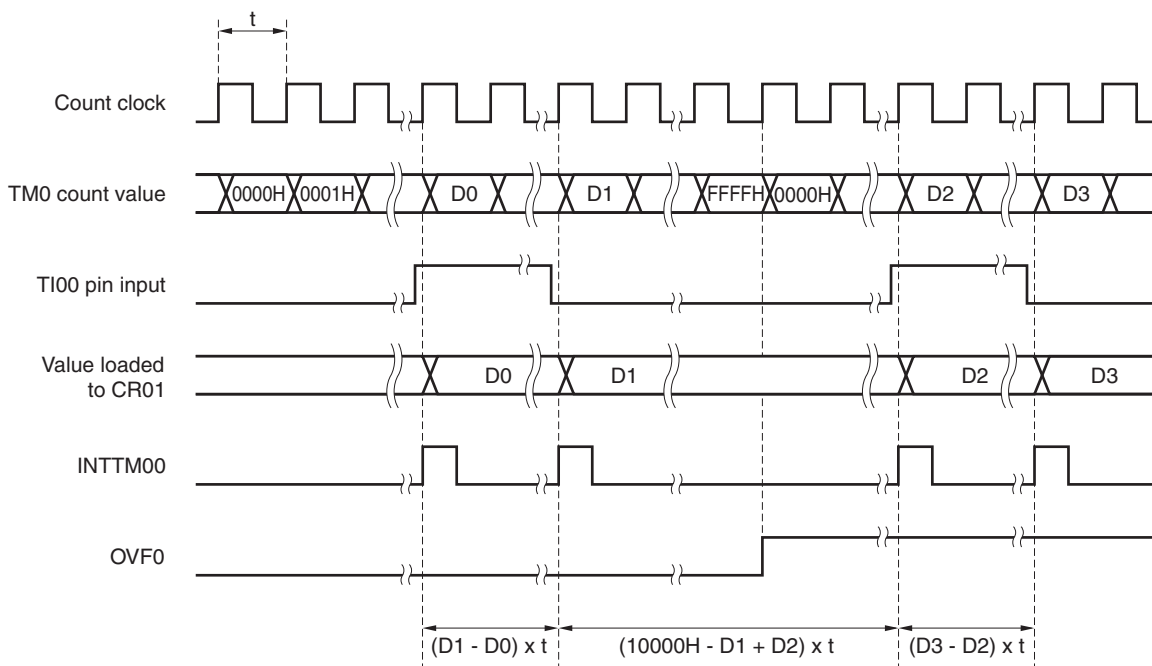
(2) Measurement of two pulse widths with free running counter

The pulse widths of the two signals respectively input to the TI00 and TI01 pins can be measured when the 16-bit timer register (TM0) is used as a free running counter (refer to Figure 6-14).

When the edge specified by bits 4 and 5 (ES00 and ES01) of the prescaler mode register 0 (PRM0) is input to the TI00 pin, the value of the TM0 is loaded to the 16-bit capture/compare register 01 (CR01) and an external interrupt request signal (INTTM01) is set.

When the edge specified by bits 6 and 7 (ES10 and ES11) of the prescaler mode register 0 (PRM0) is input to the TI01 pin, the value of TM0 is loaded to the 16-bit capture/compare register 00 (CR00), and an external interrupt request signal (INTTM00) is set.

The edges of the TI00 and TI01 pins are specified by bits 4 and 5 (ES00 and ES01) and bits 6 and 7 (ES10 and ES11) of PRM0, respectively. The rising, falling, or both rising and falling edges can be specified.

The valid edge of TI00 pin and TI01 pin is detected through sampling at a count clock cycle selected by the prescaler mode register 0 (PRM0), and the capture operation is not performed until the valid level is detected two times. Therefore, noise with a short pulse width can be rejected.

Figure 6-15: Control Register Settings for Measurement of Two Pulse Widths with Free Running Counter
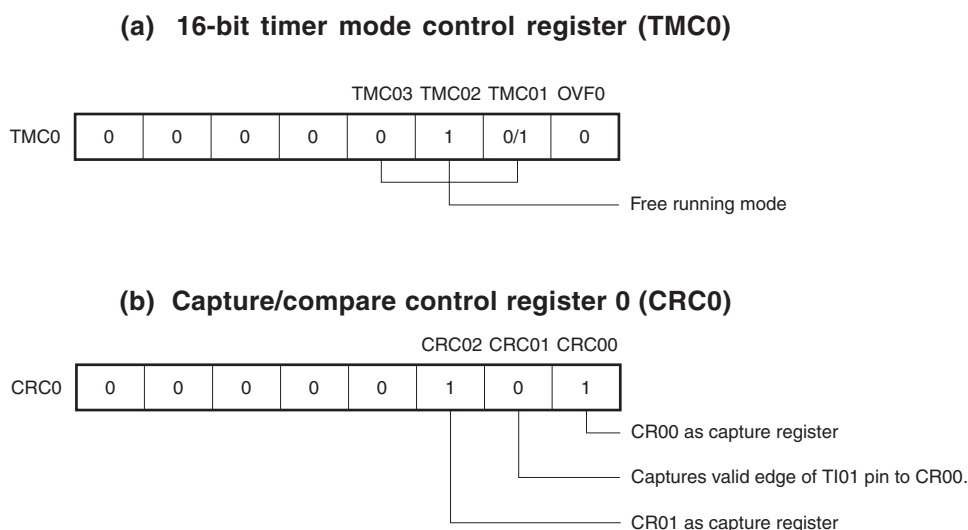
(a) 16-bit timer mode control register (TMC0)

(b) Capture/compare control register 0 (CRC0)

Remark: 0/1: When these bits are reset to 0 or set to 1, the other functions can be used along with the pulse width measurement function. For details, refer to Figures 6-2 and 6-3.

- **Capture operation (free running mode)**

The following figure illustrates the operation of the capture register when the capture trigger is input.

Figure 6-16: CR01 Capture Operation with Rising Edge Specified
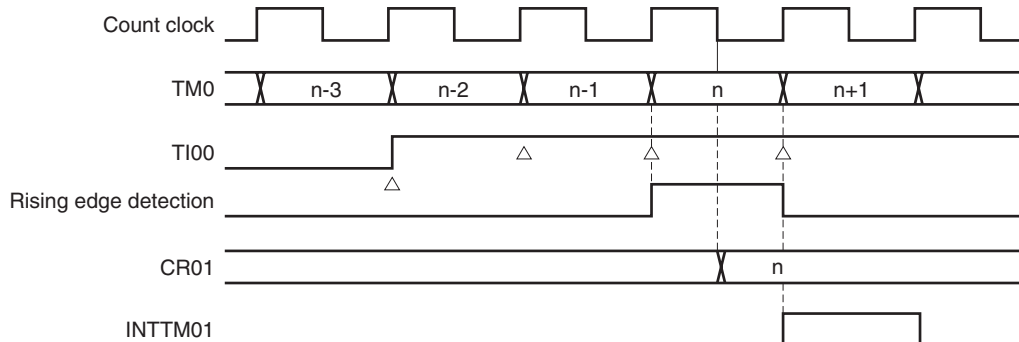
Figure 6-17: Timing of Pulse Width Measurement with Free Running Counter (with both edges specified)
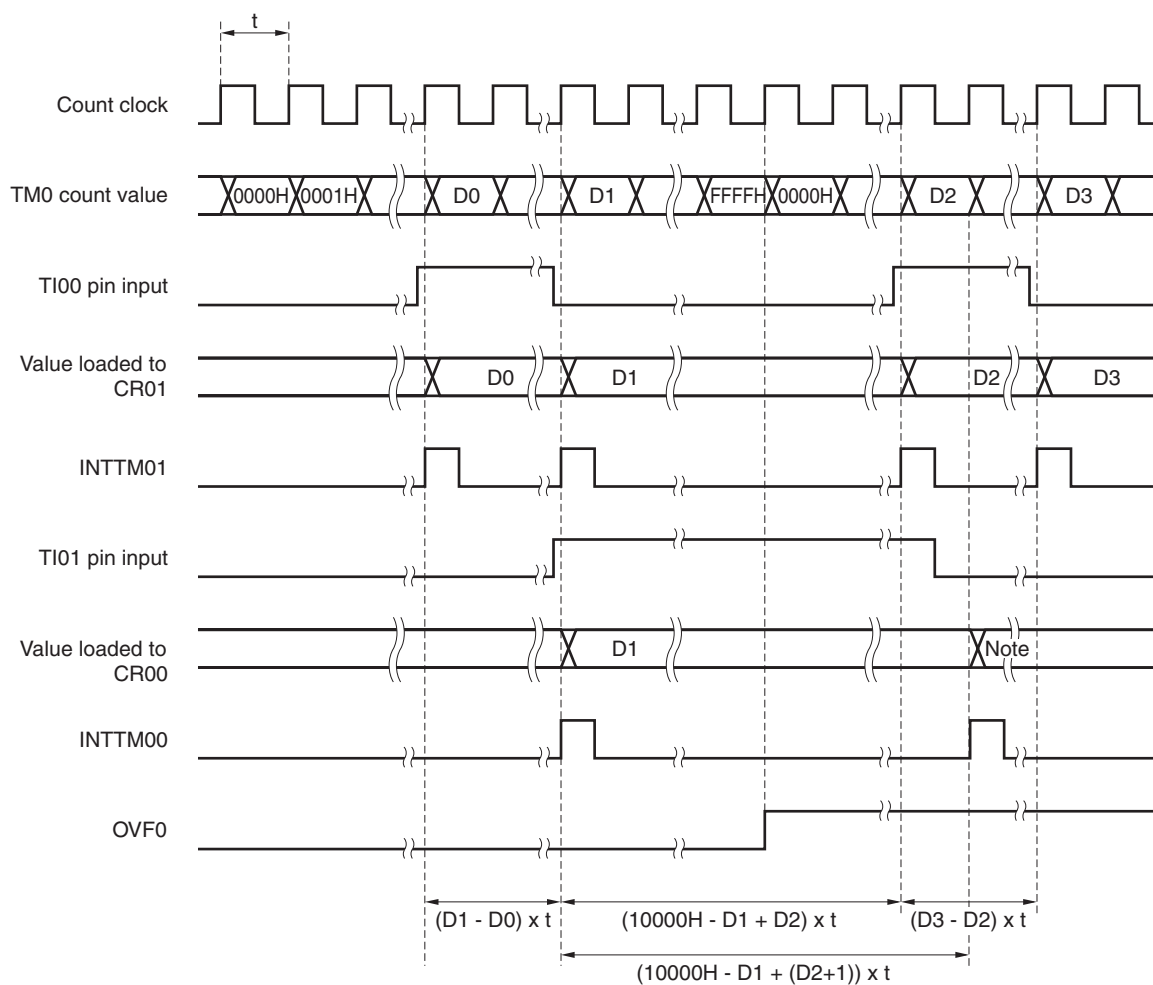
Note: $D2 + 1$

(3) Pulse width measurement with free running counter and two capture registers

When the 16-bit timer register (TM0) is used as a free running counter (refer to Figure 6-17), the pulse width of the signal input to the TI00 pin can be measured.

When the edge specified by bits 4 and 5 (ES00 and ES01) of the prescaler mode register 0 (PRM0) is input to the TI00 pin, the value of TM0 is loaded to the 16-bit capture/compare register 01 (CR01), and an external interrupt request signal (INTTM01) is set.

The value of TM0 is also loaded to the 16-bit capture/compare register 00 (CR00) when an edge reverse to the one that triggers capturing to CR01 is input.

The edge of the TI00 pin is specified by bits 4 and 5 (ES00 and ES01) of the prescaler mode register 0 (PRM0). The rising or falling edge can be specified.

The valid edge of TI00 pin and TI01 pin is detected through sampling at a count clock cycle selected by the prescaler mode register 0 (PRM0), and the capture operation is not performed until the valid level is detected two times. Therefore, noise with a short pulse width can be rejected.

Caution: **If the valid edge of the TI00 pin is specified to be both the rising and falling edges, the capture/compare register 00 (CR00) cannot perform its capture operation.**

Figure 6-18: Control Register Settings for Pulse Width Measurement with Free Running Counter and Two Capture Registers

(a) 16-bit timer mode control register (TMC0)

(b) Capture/compare control register 0 (CRC0)
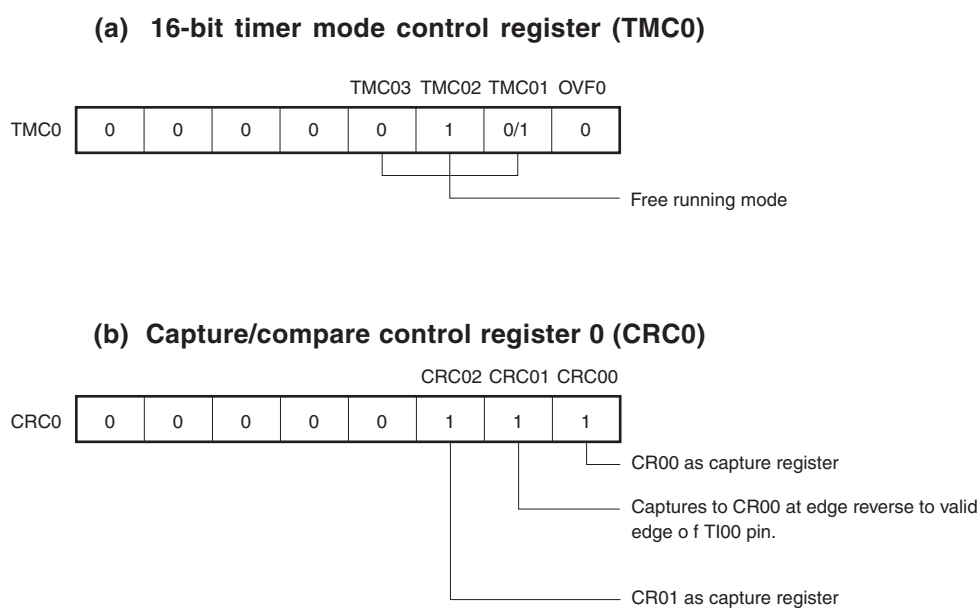
Remark: 0/1: When these bits are reset to 0 or set to 1, the other functions can be used along with the pulse width measurement function. For details, refer to Figures 6-2 and 6-3.

Figure 6-19: Timing of Pulse Width Measurement with Free Running Counter and Two Capture Registers (with rising edge specified)

<figure>

Count clock

TM0 count value: 0000H 0001H, D0, D1, FFFFH 0000H, D2, D3

TI00 pin input

Value loaded to CR01: D0, D2

Value loaded to CR00: D1, D3

INTTM01

OVF0

(D1 - D0) x t   (10000H - D1 + D2) x t   (D3 - D2) x t

</figure>
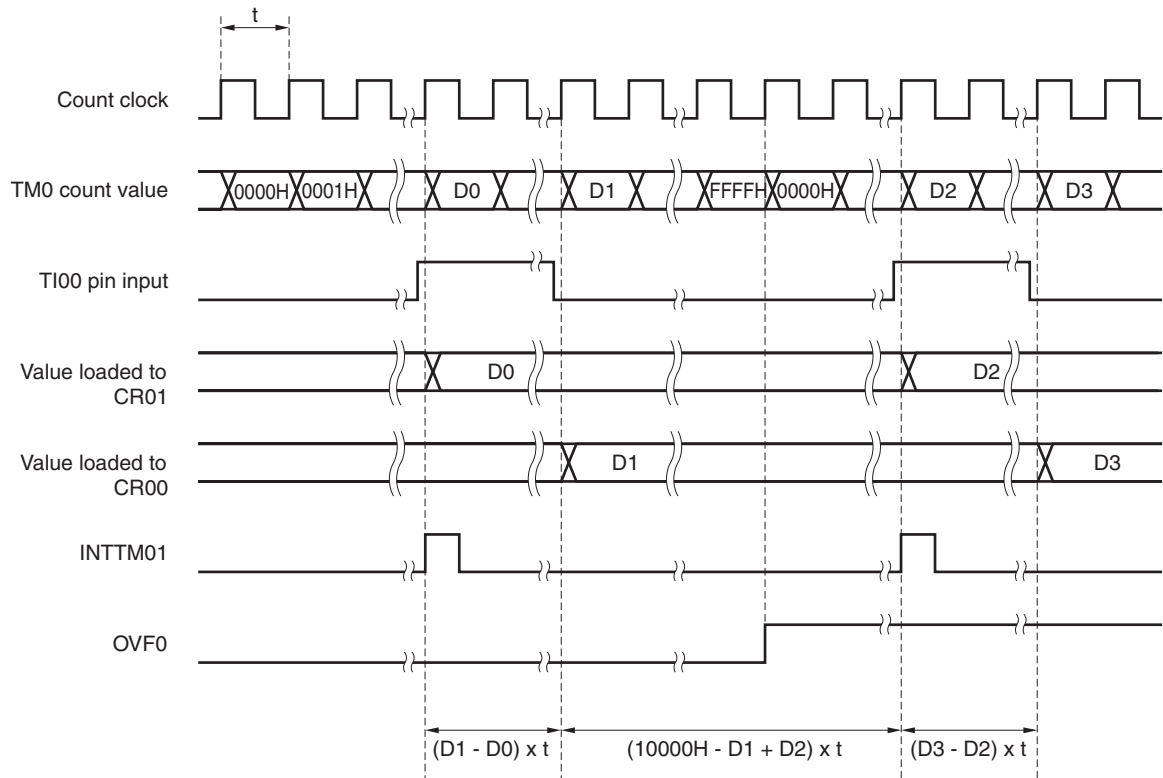
(4) Pulse width measurement by restarting

When the valid edge of the TI00 pin is detected, the pulse width of the signal input to the TI00n pin can be measured by clearing the 16-bit timer register (TM0) once and then resuming counting after loading the count value of TM0 to the 16-bit capture/compare register 01 (CR01).

The edge of the TI00 pin is specified by bits 4 and 5 (ES00 and ES01) of PRM0. The rising or falling edge can be specified.

The valid edge is detected through sampling at a count clock cycle selected by the prescaler mode register 0 (PRM0), and the capture operation is not performed until the valid level is detected two times. Therefore, noise with a short pulse width can be rejected.

Caution: **If the valid edge of the TI00 pin is specified to be both the rising and falling edges, the capture/compare register 00 (CR00) cannot perform its capture operation.**

Figure 6-20: Control Register Settings for Pulse Width Measurement by Restarting
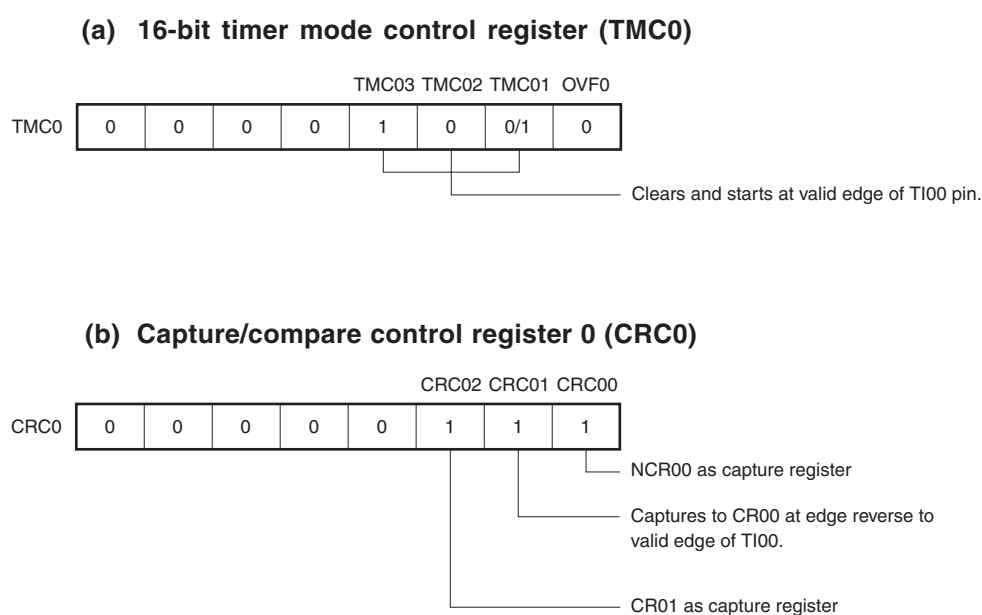
(a) 16-bit timer mode control register (TMC0)

(b) Capture/compare control register 0 (CRC0)

Remark: 0/1: When these bits are reset to 0 or set to 1, the other functions can be used along with the pulse width measurement function. For details, refer to Figures 6-2 and 6-3.

Figure 6-21: Timing of Pulse Width Measurement by Restarting (with rising edge specified)
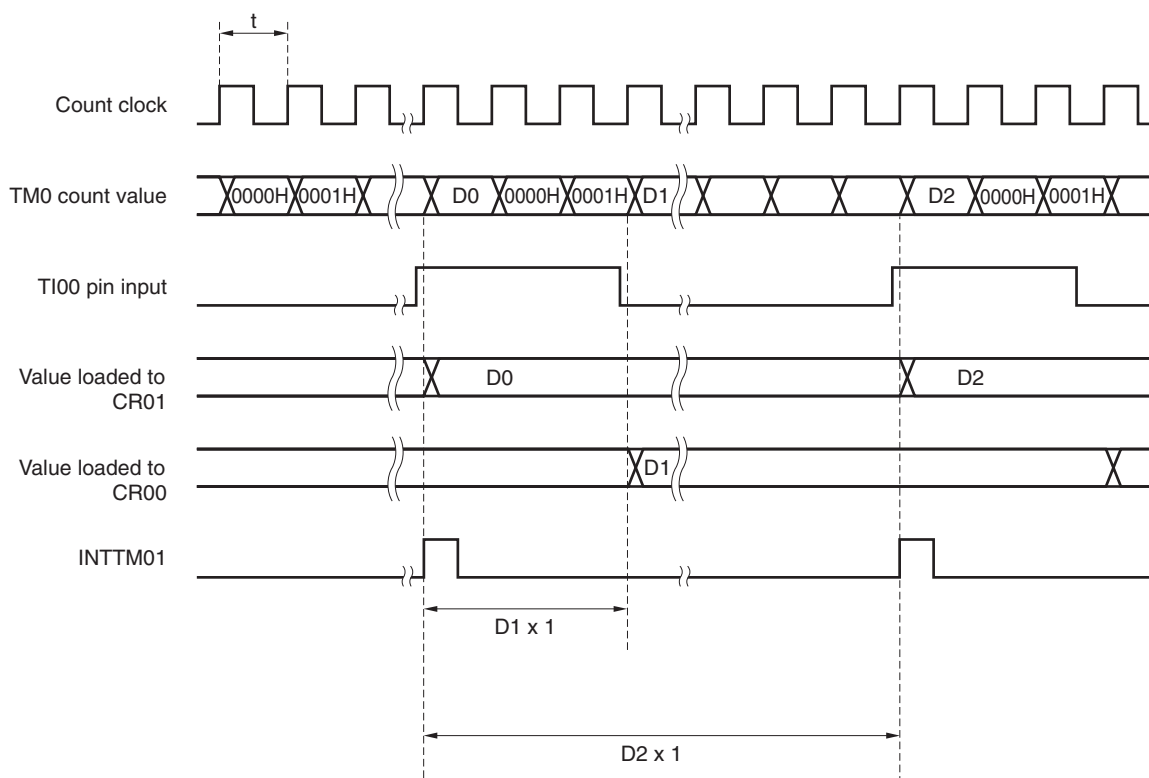
6.4.4 Operation as external event counter

16-bit timer/event counter can be used as an external event counter which counts the number of clock pulses input to the TI00 pin from an external source by using the 16-bit timer register (TM0).

Each time the valid edge specified by the prescaler mode register 0 (PRM0) has been input to the TI00 pin, TM0 is incremented.

When the count value of TM0 coincides with the value of the 16-bit capture/compare register 00 (CR00), TM0 is cleared to 0, and an interrupt request signal (INTTM00) is generated.

The edge of the TI00 pin is specified by bits 4 and 5 (ES00 and ES01) of the prescaler mode register 0 (PRM0). The rising, falling, or both the rising and falling edges can be specified.

The valid edge is detected through sampling at a count clock cycle, selected by the prescaler mode register 0 (PRM0) and performed until the valid level is detected two times. Therefore, noise with a short pulse width can be rejected.

Figure 6-22: Control Register Settings in External Event Counter Mode

(a) 16-bit timer mode control register (TMC0)
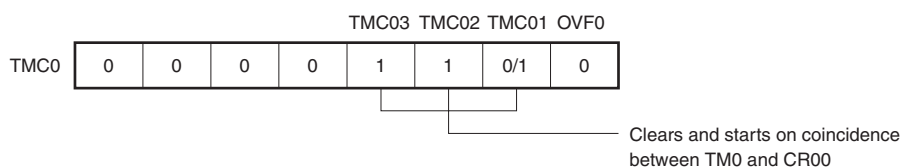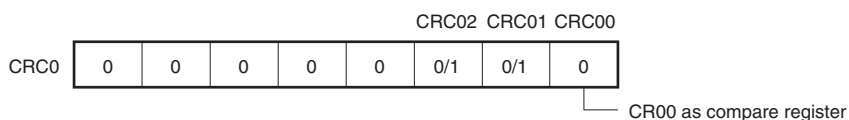
						TMC03	TMC02	TMC01	OVF0
TMC0	0	0	0	0	1	1	0/1	0

Clears and starts on coincidence
between TM0 and CR00

(b) Capture/compare control register 0 (CRC0)

						CRC02	CRC01	CRC00
CRC0	0	0	0	0	0	0/1	0/1	0

CR00 as compare register

Remark: 0/1: When these bits are reset to 0 or set to 1, the other functions can be used along with the external event counter function. For details, refer to Figures 6-2 and 6-3.

Figure 6-23: Configuration of External Event Counter
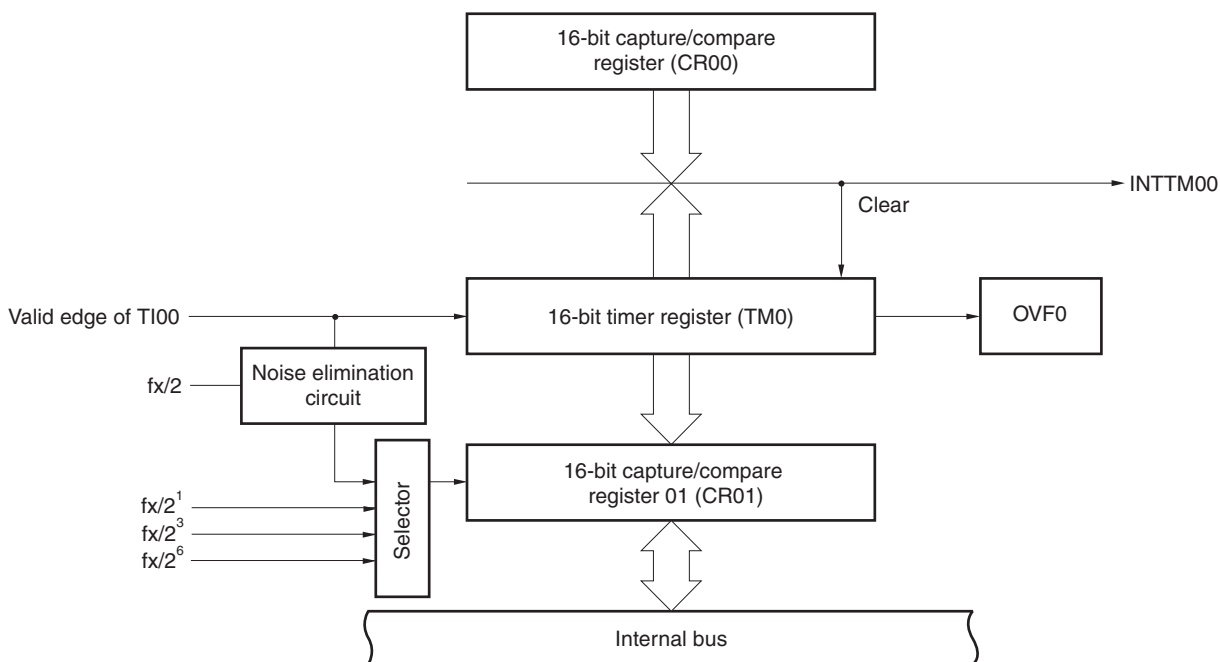
Figure 6-24: Timing of External Event Counter Operation (with rising edge specified)
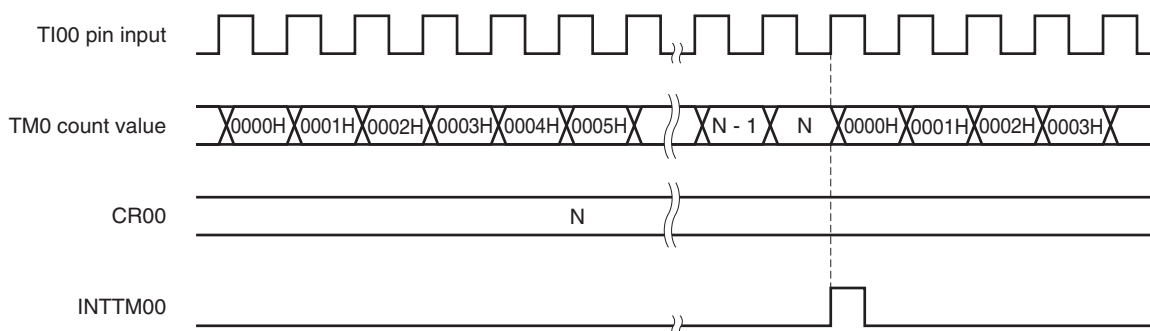
Caution: **Read TM0 when reading the count value of the external event counter.**

6.4.5 Operation to output square wave

The 16-bit timer/event counter can be used to output a square wave with any frequency at an interval specified by the count value set in advance to the 16-bit capture/compare register 00 (CR00).

By setting bits 0 (TOE0) and 1 (TOC01) of the 16-bit timer output control register to 1, the output status of the TO0 pin is reversed at an interval specified by the count value set in advance to CR00. In this way, a square wave of any frequency can be output.

Figure 6-25:　Set Contents of Control Registers in Square Wave Output Mode

(a)　16-bit timer mode control register (TMC0)

	TMC03	TMC02	TMC01	OVF0
TMC0	0	0	0	0	1	1	0/1	0

Clears and starts on coincidence between TM0 and CR00.

(b)　Capture/compare control register 0 (CRC0)

	CRC02	CRC01	CRC00
CRC0	0	0	0	0	0	0/1	0/1	0

CR00 as compare register

(c)　16-bit timer output control register (TOC0)

OSPT	OSPE	TOC04	LVS0	LVR0	TOC01	TOE0
TOC0	0	0	0	0	0/1	0/1	1	1

Enables TO0 output

Reverses output on coincidence between TM0 and CR00

Specifies initial value of TO0 output F/F

Does not reverse output on coincidence between TM0 and CR01

Disables one-shot pulse output

Remark:　0/1:　When these bits are reset to 0 or set to 1, the other functions can be used along with the square wave output function.　For details, refer to Figures 6-2, 6-3, and 6-4.

Figure 6-26:　Timing of Square Wave Output Operation
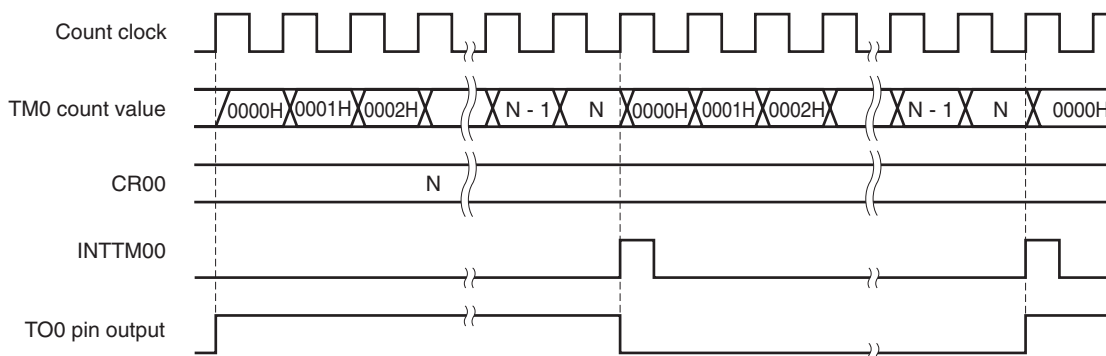
6.5 16-Bit Timer/Event Counter Operating Precautions

(1) Error on starting timer
   An error of up to 1 clock occurs before the coincidence signal is generated after the timer has been started.
   This is because the 16-bit timer register (TM0) is started asynchronously in respect to the count pulse.

Figure 6-27: Start Timing of 16-Bit Timer Register
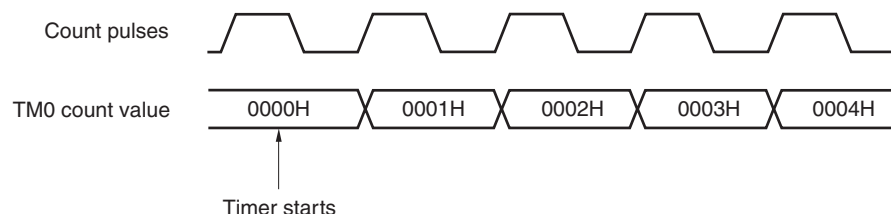
(2) 16-bit compare register setting
   Set another value than 0000H to the 16-bit captured compare register CR00, CR01. This means, that a 1-pulse count operation cannot be performed, when it is used as event counter.

(3) Setting compare register during timer count operation
   If the value to which the current value of the 16-bit capture/compare register 00 (CR00) has been changed is less than the value of the 16-bit timer register (TM0), TM0 continues counting, overflows, and starts counting again from 0.   If the new value of CR00 (M) is less than the old value (N), the timer must be restarted after the value of CR00 has been changed.

Figure 6-28: Timing after Changing Compare Register during Timer Count Operation
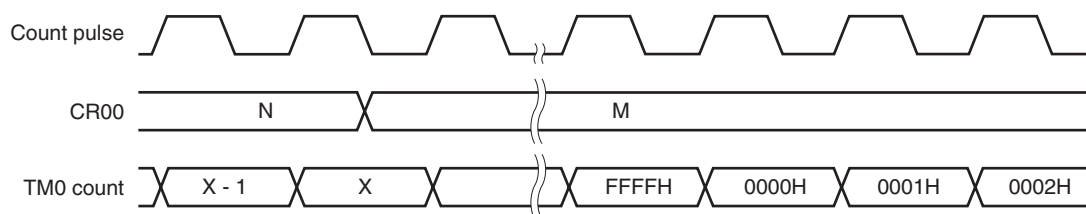
Remark: N > X > M

(4) Data hold timing of capture register

If the valid edge is input to the TI00 pin while the 16-bit capture/compare register 01 (CR01) is read, CR01 performs the capture operation, but this capture value is not guaranteed. However, the interrupt request flag (INTTM01) is set as a result of detection of the valid edge.

Figure 6-29: Data Hold Timing of Capture Register
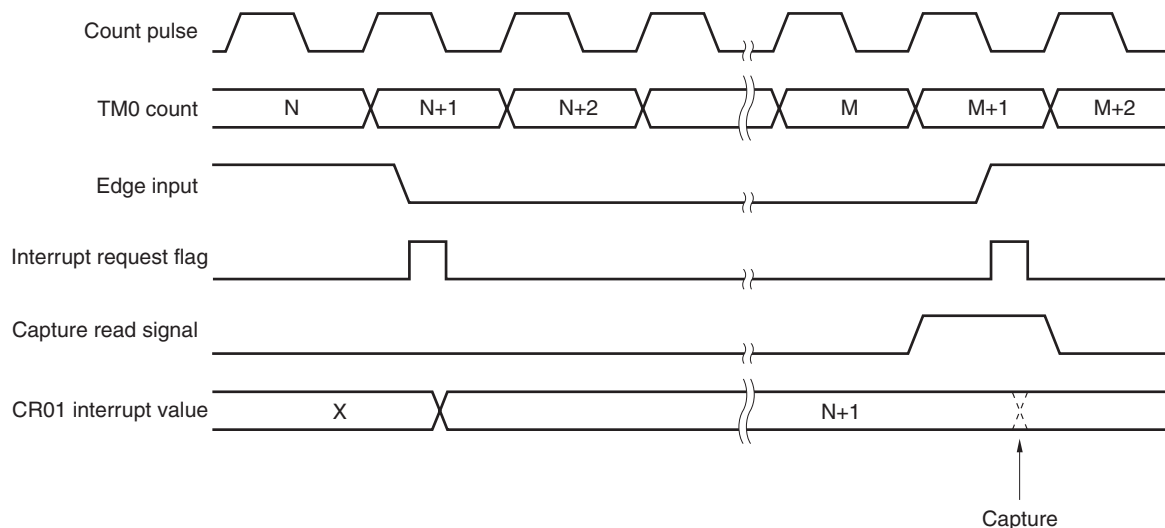
(5) Setting valid edge

Before setting the valid edge of the TI00 pin, stop the timer operation by resetting bits 2 and 3 (TMC02 and TMC03) of the 16-bit timer mode control register to 0, 0. Set the valid edge by using bits 4 and 5 (ES00 and ES01) of the prescaler mode register 0 (PRM0).

(6) Operation of OVF0 flag

The OVF0 flag is set to 1 in the following case:

Select mode in which 16-bit timer/counter is cleared and started on coincidence between TM0 and CR00.

↓

Set CR00 to FFFFH

↓

When TM0 counts up from FFFFH to 0000H
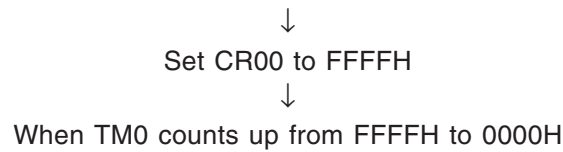
Figure 6-30: Operation Timing of OVF0 Flag
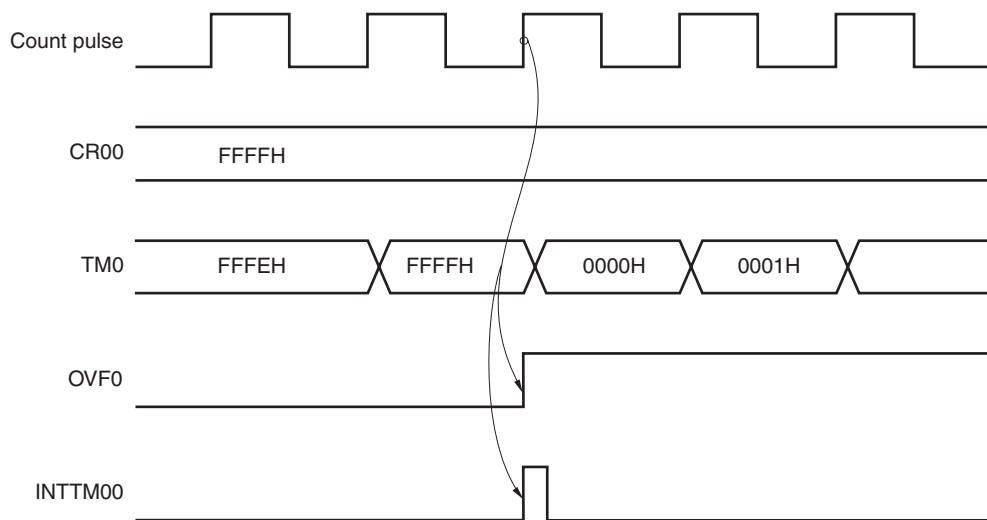
(7) Contending operations

(a) The contending operation between the read time of 16-bit capture/compare register (CR00/CR01) and capture trigger input (CR00/CR01 used as capture register)
Capture/trigger input is prior to the other. The data read from CR00/CR01 is not defined.

(b) The coincidence timing of contending operation between the write period of 16-bit capture/compare register (CR00/CR01) and 16-bit timer register (TM0) (CR00/CR01 used as a compare register)
The coincidence discriminant is not performed normally. Do not write any data to CR00/CR01 near the coincidence timing.

(8) Timer operation

(a) Even if the 16-bit timer/counter 0 (TM0) is read, the value is not captured by 16-bit timer capture/ compare register 01 (CR01).

(b) Regardless of the CPU's operation mode, when the timer stops, the input signals to pins TI00/ TI01 are not aknowledged.

(9) Capture operation

(a) If TI00 is specified as the valid edge of the count clock, capture operation by the capture register specified as the trigger for TI00 is not possible.

(b) If both the rising and the falling edges are selected as the valid edges of TI00, capture is not performed.

(c) To ensure the reliability of the capture operation, the capture trigger requires a pulse two times longer than the count clock selected by prescaler mode register 0 (PRM0).

(d) The capture operation is performed at the fall of the count clock. An interrupt request input (INTTM0n), however, is generated at the rise of the next count clock.

(10) Compare operation

(a) The INTTM0 may not be generated if the set value of 16-bit timer capture registers 00, 01 (CR00, CR01) and the count value of 16-bit timer/counter 0 (TM0) match and CR00 and CR01 are overwritten at the timing of INTTM0 generation. Therefore, do not overwrite CR00 and CR01 frequently even if overwriting the same value.

(b) Capture operation may not be performed for CR00/CR01 set in compare mode even if a capture trigger has been input.

(11) Edge detection

(a) If the TI00 pin or the TI01 pin is high level immediately after system reset and rising edge or both the rising and falling edges are specified as the valid edge for the TI00 pin or TI01 pin to enable the 16-bit timer/counter 0 (TM0) operation, a rising edge is detected imediately after. Be careful when pulling up the TI00 pin or the TI01 pin. However, the rising edge is not detected at restart after the operation has been stopped once.

(b) The sampling clock used to remove noise differs when a TI00 pin valid edge is used as a count clock and when it is used as a capture trigger. In the former case, the count clock is $fx/2^1$, and in the latter case the count clock is selected by prescaler mode register 0 (PRM0). When a valid level of the TI00 pin is detected twice by sampling with the above-mentioned sampling clock, the capture operation is started, therefore noise with short pulse can be removed.

[Memo]

Chapter 7 8-Bit Timer/Event Counters 50 and 51

7.1 8-Bit Timer/Event Counters 50 and 51 Functions

The 8-bit timer event counters 50 and 51 (TM50, TM51) have the following functions.

- Interval timer
- External event counter
- Square-wave output
- PWM output

(1) 8-bit interval timer

Interrupts are generated at the preset time intervals.

Table 7-1: 8-Bit Timer/Event Counter 50 Interval Times

Minimum Interval Width	Maximum Interval Width	Resolution
2^1 x 1/fx (250 ns)	2^9 x 1/fx (64 µs)	2^1 x 1/fx (250 ns)
2^3 x 1/fx (1 µs)	2^{11} x 1/fx (256 µs)	2^3 x 1/fx (1 µs)
2^5 x 1/fx (4 µs)	2^{13} x 1/fx (1 ms)	2^5 x 1/fx (4 µs)
2^7 x 1/fx (16 µs)	2^{15} x 1/fx (4 ms)	2^7 x 1/fx (16 µs)
2^8 x 1/fx (32 µs)	2^{16} x 1/fx (8 ms)	2^8 x 1/fx (32 µs)
2^{11} x 1/fx (256 µs)	2^{19} x 1/fx (65 ms)	2^{11} x 1/fx (256 µs)

Table 7-2: 8-Bit Timer/Event Counter 51 Interval Times

Minimum Interval Width	Maximum Interval Width	Resolution
1/fx (125 ns)	2^8 x 1/fx (32 µs)	1/fx (125 ns)
2^4 x 1/fx (2 µs)	2^{12} x 1/fx (512 µs)	2^4 x 1/fx (2 µs)
2^6 x 1/fx (8 µs)	2^{14} x 1/fx (2 ms)	2^6 x 1/fx (8 µs)
2^7 x 1/fx (16 µs)	2^{15} x 1/fx (4 ms)	2^7 x 1/fx (16 µs)
2^8 x 1/fx (32 µs)	2^{16} x 1/fx (8 ms)	2^8 x 1/fx (32 µs)
2^{10} x 1/fx (128 µs)	2^{18} x 1/fx (32 ms)	2^{10} x 1/fx (128 µs)

Remarks: **1.** fx: Main system clock oscillation frequency
2. Values in parentheses when operated at fx = 8.0 MHz.

(2) External event counter

The number of pulses of an externally input signal can be measured.

(3) Square-wave output

A square wave with any selected frequency can be output.

Table 7-3: 8-Bit Timer/Event Counter 50 Square-Wave Output Ranges

Minimum Pulse Width	Maximum Pulse Width	Resolution
2^1 x 1/fx (250 ns)	2^9 x 1/fx (64 µs)	2^1 x 1/fx (250 ns)
2^3 x 1/fx (1 µs)	2^{11} x 1/fx (256 µs)	2^3 x 1/fx (1 µs)
2^5 x 1/fx (4 µs)	2^{13} x 1/fx (1 ms)	2^5 x 1/fx (4 µs)
2^7 x 1/fx (16 µs)	2^{15} x 1/fx (4 ms)	2^7 x 1/fx (16 µs)
2^8 x 1/fx (32 µs)	2^{16} x 1/fx (8 ms)	2^8 x 1/fx (32 µs)
2^{11} x 1/fx (256 µs)	2^{19} x 1/fx (65 ms)	2^{11} x 1/fx (256 µs)

Table 7-4: 8-Bit Timer/Event Counter 50 Square-Wave Output Ranges

Minimum Pulse Width	Maximum Pulse Width	Resolution
1/fx (125 ns)	2^8 x 1/fx (32 µs)	1/fx (125 ns)
2^4 x 1/fx (2 µs)	2^9 x 1/fx (512 µs)	2^1 x 1/fx (2 µs)
2^6 x 1/fx (8 µs)	2^{11} x 1/fx (2 ms)	2^3 x 1/fx (8 µs)
2^7 x 1/fx (16 µs)	2^{13} x 1/fx (4 ms)	2^5 x 1/fx (16 µs)
2^8 x 1/fx (32 µs)	2^{15} x 1/fx (8 ms)	2^7 x 1/fx (32 µs)
2^{10} x 1/fx (128 µs)	2^{20} x 1/fx (32 ms)	2^{12} x 1/fx (128 µs)

Remarks: **1.** fx: Main system clock oscillation frequency
2. Values in parentheses when operated at fx = 8.0 MHz.

(4) PWM output

TM50 and TM51 can generate an 8-bit resolution PWM output.

7.2 8-Bit Timer/Event Counters 50 and 51 Configurations

The 8-bit timer/event counters 50 and 51 consist of the following hardware.

Table 7-5: 8-Bit Timer/Event Counters 50 and 51 Configurations

Item	Configuration
Timer register	8 bits x 2 (TM50, TM51)
Register	Compare register 8 bits x 2 (CR50, CR51)
Timer output	2 (TO50, TO51)
Control register	Timer clock select register 50 and 51 (TCL50, TCL51)
	8-bit timer mode control registers 5 and 6 (TMC50, TMC51)
	Port mode registers 0 (PM0)

Figure 7-1: 8-Bit Timer/Event Counter 50 Block Diagram
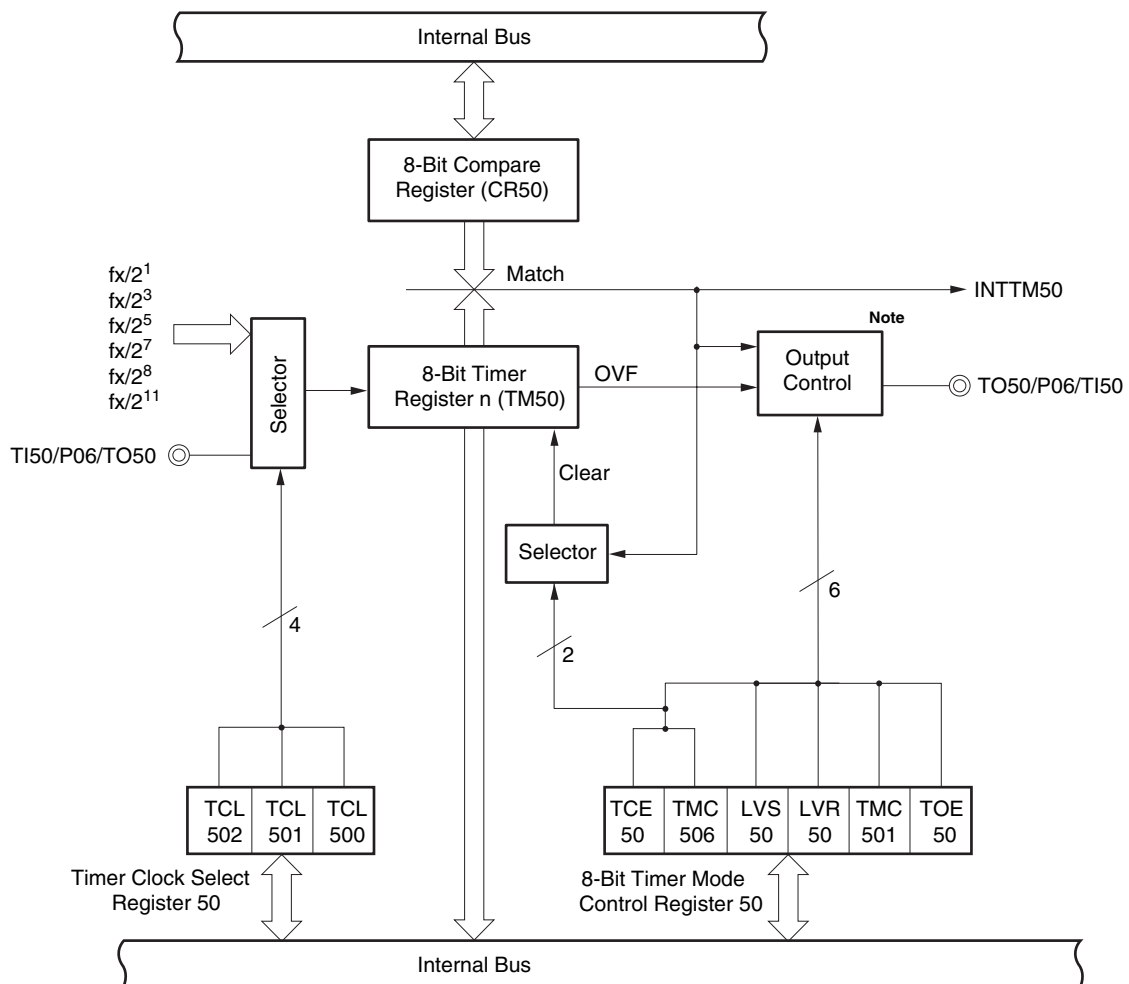
Note: Refer to Figure 7-2 for details of configurations of 8-bit timer/event counters 50 and 51 output control circuits.

Figure 7-2: 8-Bit Timer/Event Counter 51 Block Diagram
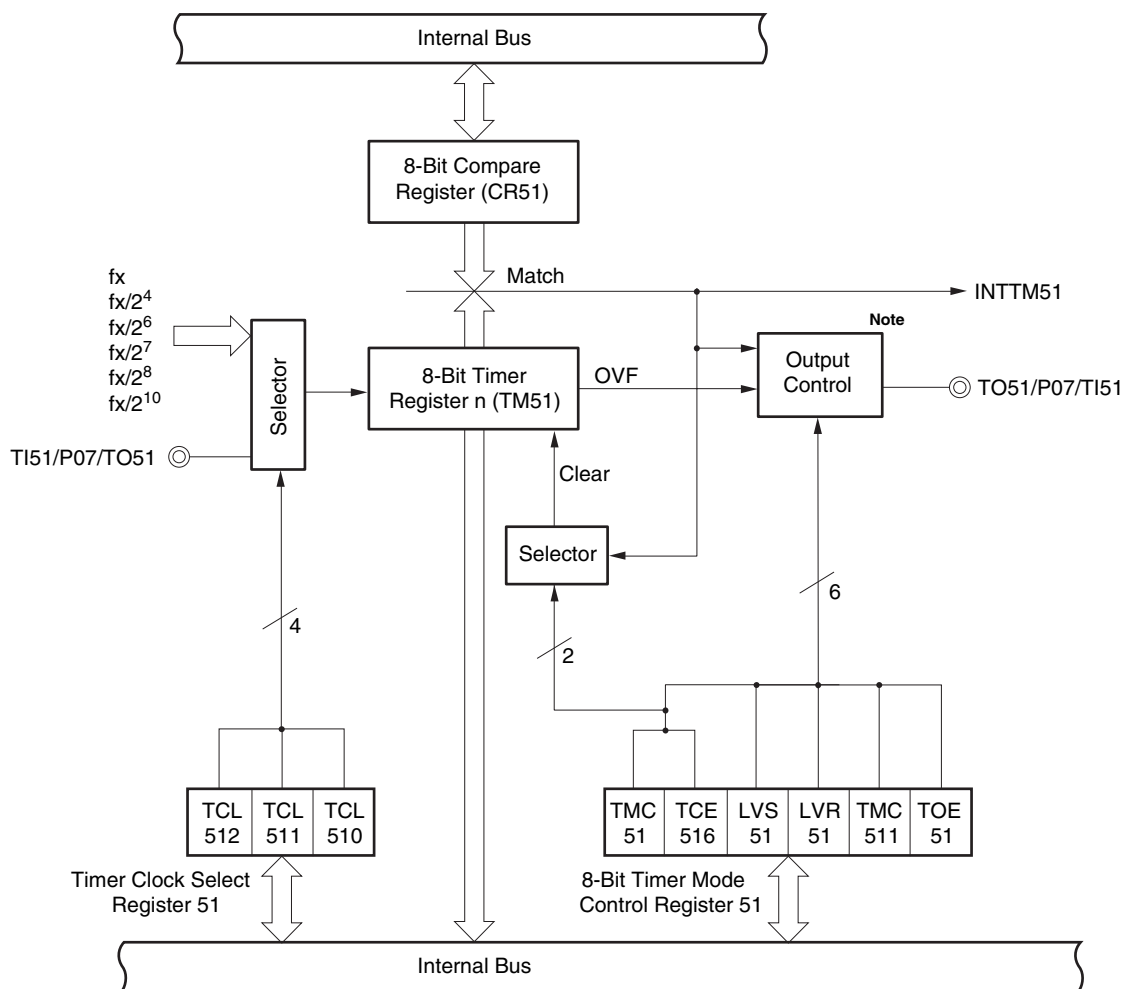
Note: Refer to Figure 7-3 for details of configurations of 8-bit timer/event counters 50 and 51 output control circuits.

Figure 7-3: Block Diagram of 8-Bit Timer/Event Counters 50 and 51 Output Control Circuit
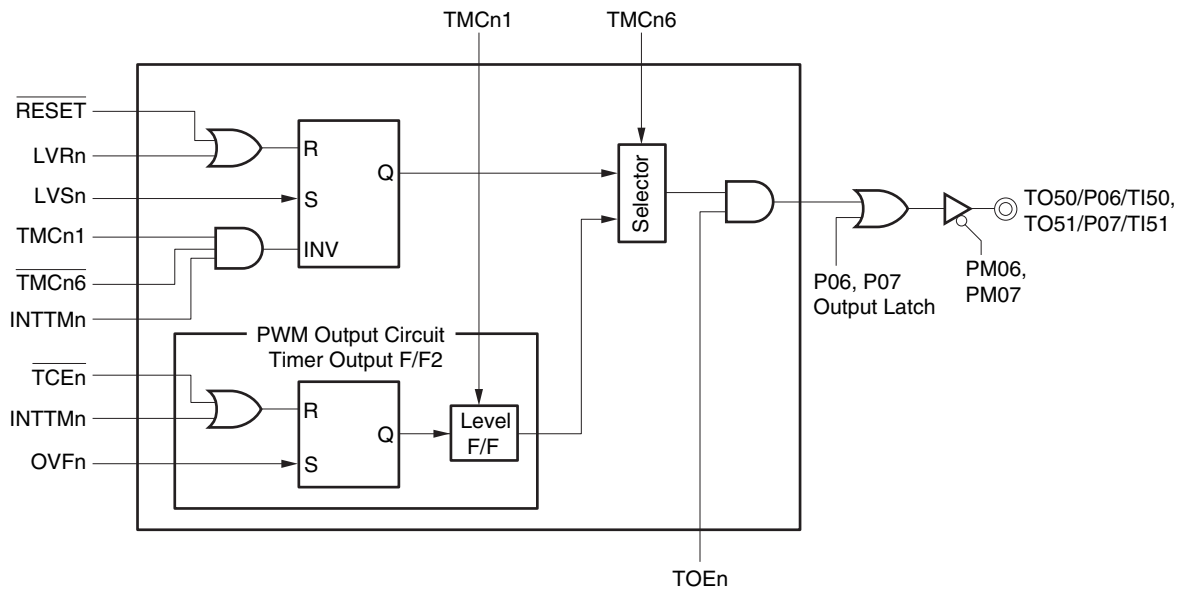
Remarks: 1. The section in the line is an output control circuit.
   2. n = 50, 51

(1) Compare register 50 and 51 (CR50, 51)
These 8-bit registers compare the value set to CR50 to 8-bit timer register 5 (TM50) count value, and the value set to CR51 to the 8-bit timer register 51 (TM51) count value, and, if they match, generate interrupts request (INTTM50 and INTTM51, respectively).
CR50 and CR51 are set with an 8-bit memory manipulation instruction. They cannot be set with a 16-bit memory manipulation instruction. The 00H to FFH values can be set.
RESET input sets CR50 and CR51 values to 00H.

Caution: To use PWM mode, set CRn value before setting TMCn (n = 50, 51) to PWM mode.

(2) 8-bit timer registers 50 and 51 (TM50, TM51)
These 8-bit registers count count pulses.
TM50 and TM51 are read with an 8-bit memory manipulation instruction.
RESET input sets TM50 and TM51 to 00H.

149

7.3 8-Bit Timer/Event Counters 50 and 51 Control Registers

The following three types of registers are used to control the 8-bit timer/event counters 50 and 51.

- Timer clock select register 50 and 51 (TCL50, TCL51)
- 8-bit timer mode control registers 50 and 51 (TMC50, TMC51)
- Port mode register 0 (PM0)

(1) Timer clock select register 50 (TCL50)
This register sets count clocks of 8-bit timer register 50.

TCL50 is set with an 8-bit memory manipulation instruction.

RESET input sets TCL50 to 00H.

Figure 7-4: Timer Clock Select Register 50 Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
TCL50	0	0	0	0	0	TCL502	TCL501	TCL500	FF71H	00H	R/W

TCL502	TCL501	TCL500	8-bit Timer Register 50 Count Clock Selection
0	0	0	TI50 falling edge [Note]
0	0	1	TI50 rising edge [Note]
0	1	0	$fx/2^1$ (4.0 MHz)
0	1	1	$fx/2^3$ (1.0 MHz)
1	0	0	$fx/2^5$ (250 kHz)
1	0	1	$fx/2^7$ (62.5 kHz)
1	1	0	$fx/2^8$ (31.25 kHz)
1	1	1	$fx/2^{11}$ (3.9 kHz)
Other than above			Setting prohibited

Note: When clock is input from the external, timer output (PWM output) cannot be used.

Caution: **When rewriting TCL50 to other data, stop the timer operation beforehand.**

Remarks: **1.** fx: Main system clock oscillation frequency
             2. TI50: 8-bit timer register 50 input pin
             3. Values in parentheses apply to operation with fx = 8.0 MHz

(2) Timer clock select register 51 (TCL51)

This register sets count clocks of 8-bit timer register 51.

TCL51 is set with an 8-bit memory manipulation instruction.

RESET input sets TCL51 to 00H.

Figure 7-5: Timer Clock Select Register 51 Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
TCL51	0	0	0	0	0	TCL512	TCL511	TCL510	FF75H	00H	R/W

TCL512	TCL511	TCL510	8-bit Timer Register 51 Count Clock Selection
0	0	0	TI51 falling edge [Note]
0	0	1	TI51 rising edge [Note]
0	1	0	fx (8.0 MHz)
0	1	1	$fx/2^4$ (500 kHz)
1	0	0	$fx/2^6$ (125 kHz)
1	0	1	$fx/2^7$ (62.5 kHz)
1	1	0	$fx/2^8$ (31.25 kHz)
1	1	1	$fx/2^{10}$ (7.8 kHz)
Other than above			Setting prohibited

Note: When clock is input from the external, timer output (PWM output) cannot be used.

Caution: When rewriting TCL51 to other data, stop the timer operation beforehand.

Remarks: 1. fx: Main system clock oscillation frequency
            2. TI51: 8-bit timer register 51 input pin
            3. Values in parentheses apply to operation with fx = 8.0 MHz

(3) 8-bit timer mode control register 50 (TMC50)

This register enables/stops operation of 8-bit timer register 50, sets the operating mode of 8-bit timer register 50 and controls operation of 8-bit timer/event counter 50 output control circuit.

It selects the R-S flip-flop (timer output F/F 1,2) setting/resetting, the active level in PWM mode, inversion enabling/disabling in modes other than PWM mode and 8-bit timer/event counter 5 timer output enabling/disabling.

TMC50 is set with a 1-bit or 8-bit memory manipulation instruction.

RESET input sets TMC50 to 04H.

Figure 7-6: 8-Bit Timer Output Control Register 50 Format

Symbol	<7>	6	5	4	<3>	<2>	1	<0>	Address	After Reset	R/W
TMC50	TCE50	TMC506	0	0	LVS50	LVR50	TMC501	TOE50	FF70H	04H	R/W

TOE50	8-Bit Timer/Event Counter 50 Output Control
0	Output disabled (Port mode)
1	Output enabled

TMC501	In PWM Mode	In Other Mode
	Active level selection	Timer output F/F1 control
0	Active high	Inversion operation disabled
1	Active low	Inversion operation enabled

LVS50	LVR50	8-Bit Timer/Event Counter 50 Timer Output F/F1 Status Setting
0	0	No change
0	1	Timer output F/F1 reset (0)
1	0	Timer output F/F1 set (1)
1	1	Setting prohibited

TMC506	8-Bit Timer/Event Counter 50 Operating Mode Selection
0	Clear & start mode on match of TM50 and CR50
1	PWM mode (free-running)

TCE50	8-Bit Timer Register 50 Operation Control
0	Operation Stop (TM50 clear to 0)
1	Operation Enable

Cautions: **1. Timer operation must be stopped before setting TMC50.**
2. If LVS50 and LVR50 are read after data are set, they will be 0.
3. Be sure to set bit 4 and bit 5 to 0.

Note: **If TM50 is used as clock generation for SIO3, no clock will be supplied to SIO3 unless TOE50 is set to 1. In this case a square wave signal is output from the TO50 pin.**

(4) 8-bit timer mode control register 51 (TMC51)

This register enables/stops operation of 8-bit timer register 51, sets the operating mode of 8-bit timer register 51 and controls operation of 8-bit timer/event counter 51 output control circuit.

It selects the R-S flip-flop (timer output F/F 1,2) setting/resetting, active level in PWM mode, inversion enabling/disabling in modes other than PWM mode and 8-bit timer/event counter 51 timer output enabling/disabling.

TMC51 is set with a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets TMC51 to 04H.

Figure 7-7: 8-Bit Timer Output Control Register 51 Format

Symbol	<7>	6	5	4	<3>	<2>	1	<0>
TMC51	TCE51	TMC516	0	0	LVS51	LVR6	TMC511	TOE51

Address   After Reset   R/W
FF74H   04H   R/W

TOE51	8-Bit Timer/Event Counter 51 Output Control
0	Output disabled (Port mode)
6	Output enabled

TMC511	In PWM Mode	In Other Mode
	Active level selection	Timer output F/F1 control
0	Active high	Inversion operation disabled
1	Active low	Inversion operation enabled

LVS51	LVR51	8-Bit Timer/Event Counter 51 Timer Output F/F1 Status Setting
0	0	No change
0	1	Timer output F/F1 reset (0)
1	0	Timer output F/F1 set (1)
1	1	Setting prohibited

TMC516	8-Bit Timer/Event Counter 51 Operating Mode Selection
0	Clear & start mode on match of TM51 and CR51
1	PWM mode (free-running)

TCE51	8-Bit Timer Register 51 Operation Control
0	Operation Stop (TM51 clear to 0)
1	Operation Enable

Cautions   1. Timer operation must be stopped before setting TMC51.

   2. If LVS51 and LVR51 are read after data are set, they will be 0.

   3. Be sure to set bit 4 and bit 5 to 0.

(5) Port mode register 0 (PM0)

This register sets port 0 input/output in 1-bit units.

When using the P06/TI50/TO50 and P07/TI51/TO51 pins for timer output, set PM06, PM07 and output latches of P06 and P07 to 0.

PM0 is set with a 1-bit or 8-bit memory manipulation instruction.

RESET input sets PM0 to FFH.

Figure 7-8: Port Mode Register 0 Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
PM0	PM07	PM06	1	1	1	PM02	PM01	PM00	FF20H	FFH	R/W

PM0n	P0n Pin Input/Output Mode Selection (n=0 to 7)
0	Output mode (output buffer ON)
1	Input mode (output buffer OFF)
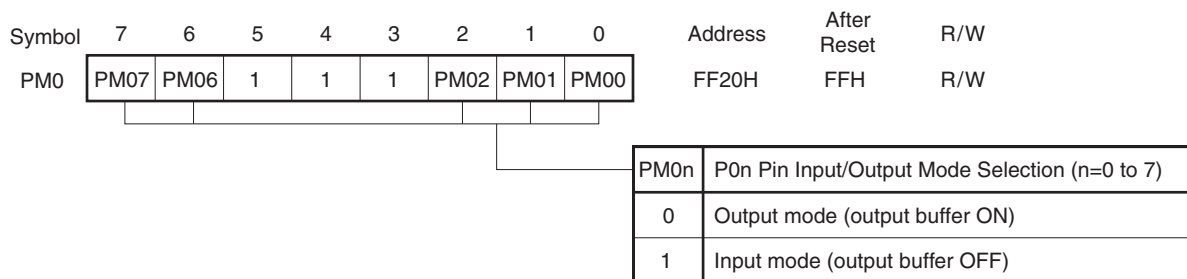
7.4 8-Bit Timer/Event Counters 50 and 51 Operations

7.4.1 Interval timer operations

Setting the 8-bit timer mode control registers (TMC50 and TMC51) as shown in Figure 7-9 allows operation as an interval timer. Interrupts are generated repeatedly using the count value preset in 8-bit compare registers (CR50 and CR51) as the interval.

When the count value of the 8-bit timer register 50 or 51 (TM50, TM51) matches the value set to CR50 or CR51, counting continues with the TM50 or TM51 value cleared to 0 and the interrupt request signal (INTTM50, INTTM51) is generated.

Count clock of the 8-bit timer register 50 (TM50) can be selected with the timer clock select register 50 (TCL50) and count clock of the 8 bit timer register 51 (TM51) can be selected with the timer clock select register 51 (TCL51).

Figure 7-9: 8-Bit Timer Mode Control Register Settings for Interval Timer Operation
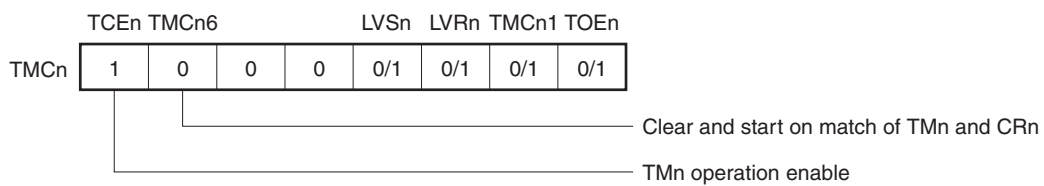
Remarks: **1.** 0/1: Setting 0 or 1 allows another function to be used simultaneously with the interval timer. See **9.3 (3), (4)** for details.
2. n = 50, 51

Figure 7-10: Interval Timer Operation Timings (1/3)
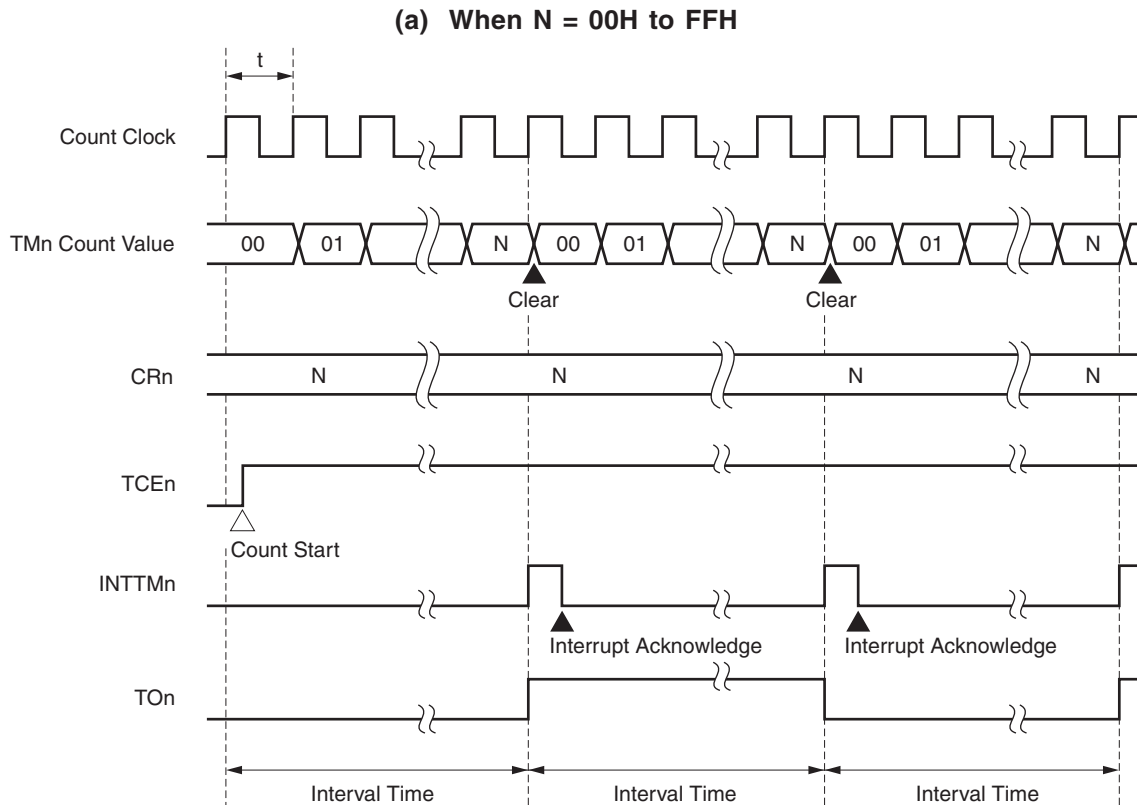
(a) When N = 00H to FFH

Remarks: **1.** Interval time = (N + 1) x t: N = 00H to FFH
2. n = 50, 51
3. Signal output at TO50, when defined as square wave output.

Figure 7-10: Interval Timer Operation Timings (2/3)

(b) When CRn = 00H

Count clock

TMn

CRn

TCEn

INTTMn

TIOn

Interval time

(c) When CRn = FFH

Count clock

TMn

CRn

TCEn

INTTMn

Interrupt received

TIOn

Interrupt received
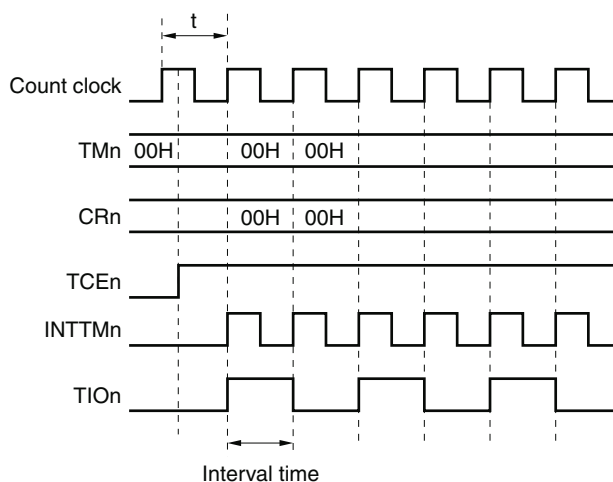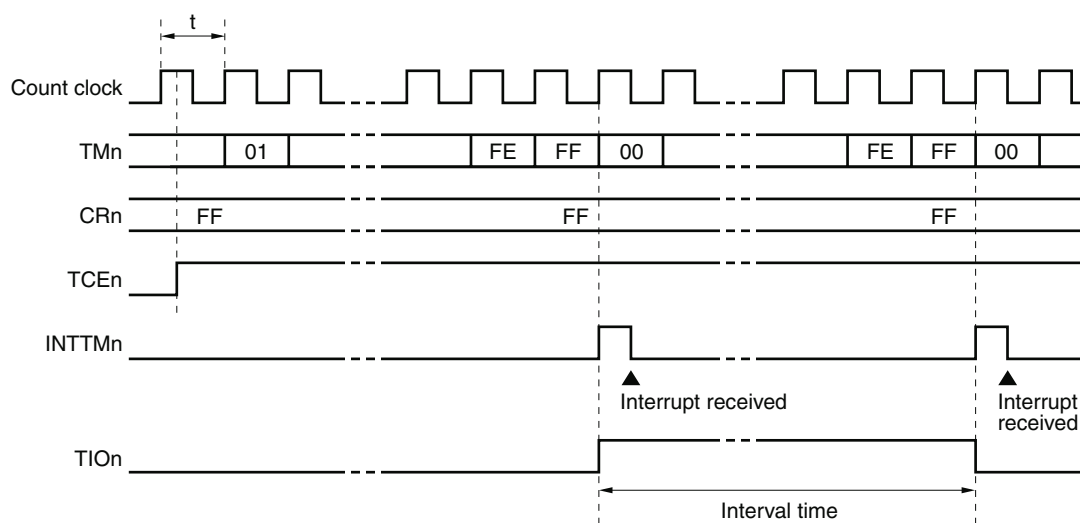
Interval time

Remark: n = 50, 51

Figure 7-10: Interval Timer Operation Timings (3/3)

(d)　Operated by CR5n transition (M < N)

CRn transition　　TMn overflows since M < N
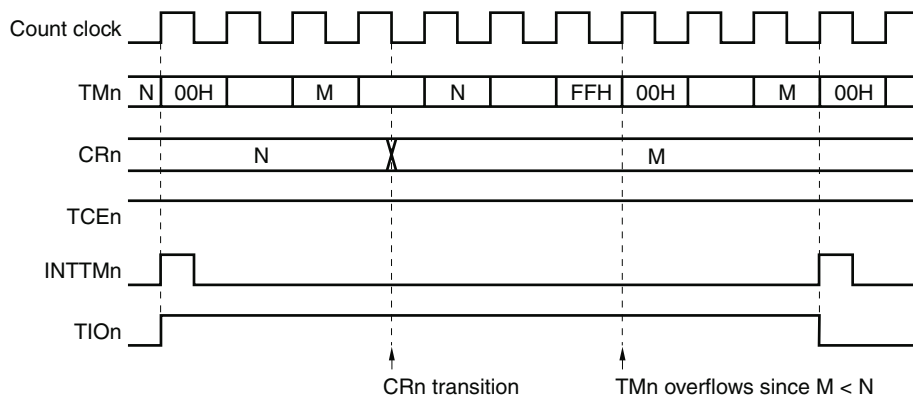
(e)　Operated by CR5n transition (M > N)

CRn transition
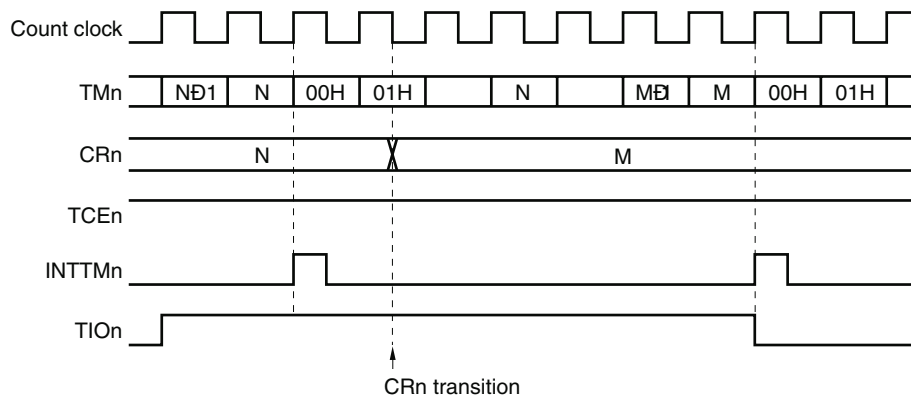
Remark:　n = 50, 51

Table 7-6: 8-Bit Timer/Event Counters 50 Interval Times

TCLn2	TCLn1	TCLn0	Minimum Interval Time	Maximum Interval Time	Resolution
0	0	0	Tin input cycle	2^8 x Tin input cycle	Tin input edge input cycle
0	0	1	Tin input cycle	2^8 x Tin input cycle	Tin input edge input cycle
0	1	0	2^1 x 1/fx (250 ns)	2^9 x 1/fx (64 μs)	2^1 x 1/fx (250 ns)
0	1	1	2^3 x 1/fx (1 μs)	2^{11} x 1/fx (256 μs)	2^3 x 1/fx (1 μs)
1	0	0	2^5 x 1/fx (4 μs)	2^{13} x 1/fx (1 ms)	2^5 x 1/fx (4 μs)
1	0	1	2^7 x 1/fx (16 μs)	2^{15} x 1/fx (4 ms)	2^7 x 1/fx (16 μs)
1	1	0	2^8 x 1/fx (32 μs)	2^{16} x 1/fx (8 ms)	2^8 x 1/fx (32 μs)
1	1	1	2^{11} x 1/fx (256 μs)	2^{19} x 1/fx (65 ms)	2^{11} x 1/fx (256 μs)
Other than above			Setting prohibited		

Table 7-7: 8-Bit Timer/Event Counters 51 Interval Times

TCLn2	TCLn1	TCLn0	Minimum Interval Time	Maximum Interval Time	Resolution
0	0	0	Tin input cycle	2^8 x Tin input cycle	Tin input edge input cycle
0	0	1	Tin input cycle	2^8 x Tin input cycle	Tin input edge input cycle
0	1	0	1/fx (125 ns)	2^8 x 1/fx (32 μs)	1/fx (125 ns)
0	1	1	2^4 x 1/fx (2 μs)	2^{12} x 1/fx (512 μs)	2^4 x 1/fx (2 μs)
1	0	0	2^6 x 1/fx (8 μs)	2^{14} x 1/fx (2 ms)	2^6 x 1/fx (8 μs)
1	0	1	2^7 x 1/fx (16 μs)	2^{15} x 1/fx (4 ms)	2^7 x 1/fx (16 μs)
1	1	0	2^8 x 1/fx (32 μs)	2^{16} x 1/fx (8 ms)	2^8 x 1/fx (32 μs)
1	1	1	2^{10} x 1/fx (128 μs)	2^{18} x 1/fx (32 ms)	2^{10} x 1/fx (128 μs)
Other than above			Setting prohibited		

Remarks: **1.** fx: Main system clock oscillation frequency

2. Values in parentheses apply to operation with fx = 8.0 MHz.

3. n = 50, 51

7.4.2 External event counter operation

The external event counter counts the number of external clock pulses to be input to the TI50/P06/TO50 and TI51/P07/TO51 pins with 8-bit timer registers 50 and 51 (TM50 and TM51).

TM50 and TM51 are incremented each time the valid edge specified with timer clock select registers 50 and 51 (TCL50 and TCL51) is input. Either rising or falling edge can be selected.

When the TM50 and TM51 counted values match the values of 8-bit compare registers (CR50 and CR51), TM50 and TM51 are cleared to 0 and the interrupt request signals (INTTM50 and INTTM51) are generated.

Figure 7-11: 8-Bit Timer Mode Control Register Setting for External Event Counter Operation
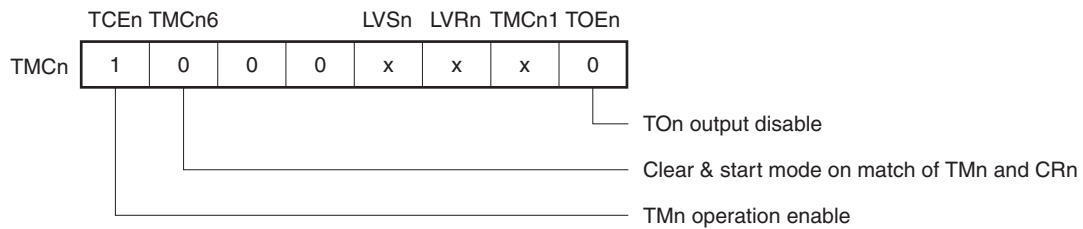
Remarks: **1.** n = 50, 51
   2. x: don't care

Figure 7-12: External Event Counter Operation Timings (with Rising Edge Specified)
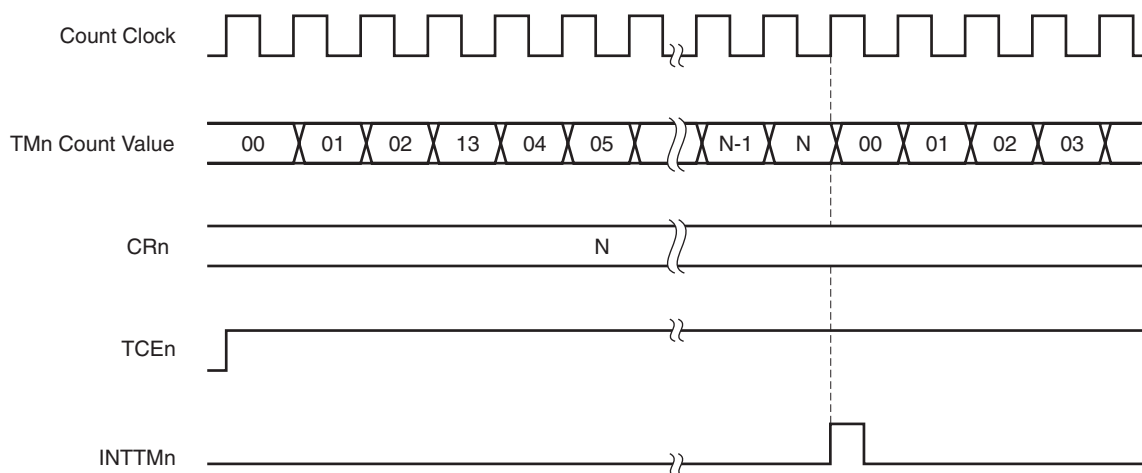
Remarks: **1.** N = 00H to FFH
   2. n = 50, 51

159

7.4.3 Square-wave output

A square wave with any selected frequency is output at intervals of the value preset to 8-bit compare registers (CR50 and CR51).

The TO50/P06/TI50 or TO51/P07/TI51 pin output status is reversed at intervals of the count value preset to CR50 or CR51 by setting bit 1 (TMC501) and bit 0 (TOE50) of the 8-bit timer output control register 5 (TMC50), or bit 1 (TMC511) and bit 0 (TOE51) of the 8-bit timer mode control register 6 (TMC51) to 1.

This enables a square wave of any selected frequency to be output.

Figure 7-13: 8-Bit Timer Mode Control Register Settings for Square-Wave Output Operation
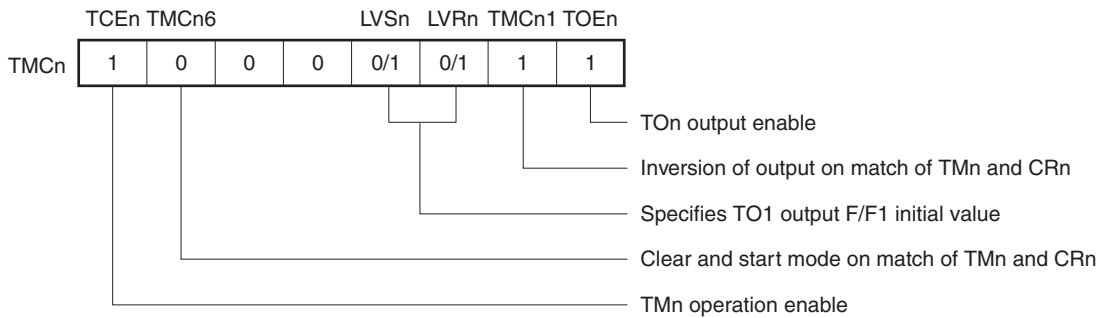
Caution: **When TI50/P06/TO50 or TI51/P07/TO51 pin is used as the timer output, set port mode register (PM00 or PM07) and output latch to 0.**

Remark: n = 50, 51

Figure 7-14: Square-wave Output Operation Timing
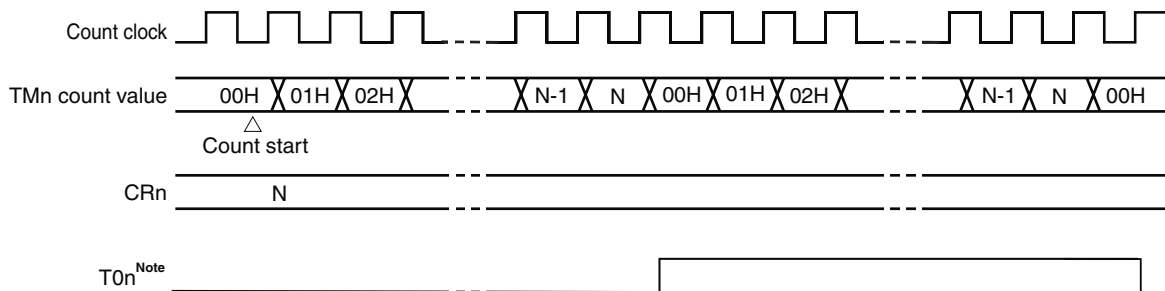
Note: TOn output initial value can be set by bits 2 and 3 (LVRn, LVSn) of the 8-bit timer mode control register TCMn.

Remark: n = 50, 51

Table 7-8: 8-Bit Timer/Event Counters 50 Square-Wave Output Ranges

Minimum Pulse Time	Maximum Pulse Time	Resolution
2^1 x 1/fx (250 ns)	2^9 x 1/fx (64 µs)	2^1 x 1/fx (250 ns)
2^3 x 1/fx (1 µs)	2^{11} x 1/fx (256 µs)	2^3 x 1/fx (1 µs)
2^5 x 1/fx (4 µs)	2^{13} x 1/fx (1 ms)	2^5 x 1/fx (4 µs)
2^7 x 1/fx (16 µs)	2^{15} x 1/fx (4 ms)	2^7 x 1/fx (16 µs)
2^8 x 1/fx (32 µs)	2^{16} x 1/fx (8 ms)	2^8 x 1/fx (32 µs)
2^{11} x 1/fx (256 µs)	2^{19} x 1/fx (65 ms)	2^{11} x 1/fx (256 µs)

Table 7-9: 8-Bit Timer/Event Counters 51 Square-Wave Output Ranges

Minimum Pulse Time	Maximum Pulse Time	Resolution
1/fx (125 ns)	2^8 x 1/fx (32 µs)	1/fx (125 ns)
2^4 x 1/fx (2 µs)	2^{12} x 1/fx (512 µs)	2^4 x 1/fx (2 µs)
2^6 x 1/fx (8 µs)	2^{14} x 1/fx (2 ms)	2^6 x 1/fx (8 µs)
2^7 x 1/fx (16 µs)	2^{15} x 1/fx (4 ms)	2^7 x 1/fx (16 µs)
2^8 x 1/fx (32 µs)	2^{16} x 1/fx (8 ms)	2^8 x 1/fx (32 µs)
2^{10} x 1/fx (128 µs)	2^{18} x 1/fx (32 ms)	2^{10} x 1/fx (128 µs)

Remarks: **1.** f: Main system clock oscillation frequency
2. Values in parentheses when operated at fx = 8.0 MHz.
3. n = 50, 51

7.4.4 PWM output operations

Setting the 8-bit timer mode control registers (TMC50 and TMC51) as shown in Figure 7-15 allows operation as PWM output. Pulses with the duty rate determined by the values preset in 8-bit compare registers (CR50 and CR51) output from the TO50/P06/TI50 or TO51/P07/TI51 pin.

Select the active level of PWM pulse with bit 1 of the 8-bit timer mode control register 50 (TMC50) or bit 1 of the 8-bit timer mode control register 51 (TMC51).

This PWM pulse has an 8-bit resolution. The pulse can be converted into an analog voltage by integrating it with an external low-pass filter (LPF). Count clock of the 8-bit timer register 50 (TM50) can be selected with the timer clock select register 50 (TCL50) and count clock of the 8-bit timer register 51 (TM51) can be selected with the timer clock select register 51 (TCL51).

PWM output enable/disable can be selected with bit 0 (TOE50) of TMC50 or bit 0 (TOE51) of TMC51.

Figure 7-15: 8-Bit Timer Control Register Settings for PWM Output Operation
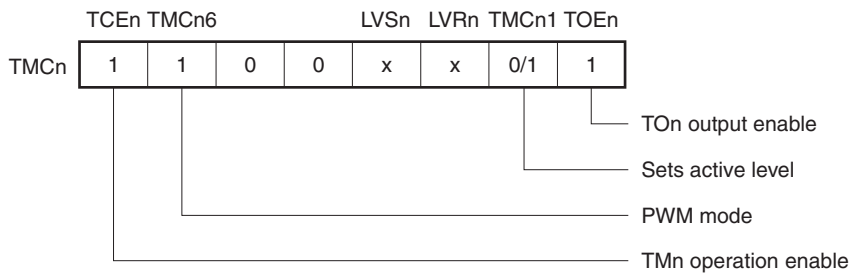
Remarks: 1. n = 50, 51
           2. x: don't care

Figure 7-16: PWM Output Operation Timing (Active high setting)
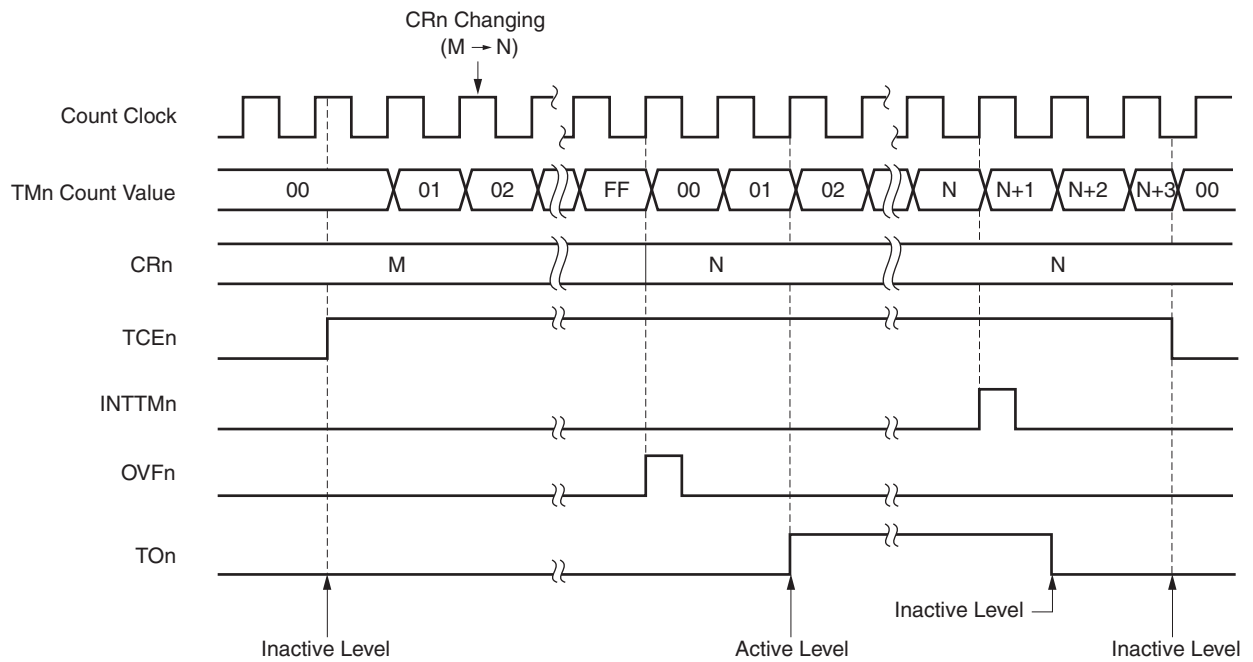
Remark: n = 50, 51

Figure 7-17: PWM Output Operation Timings (CRn0 = 00H, active high setting)
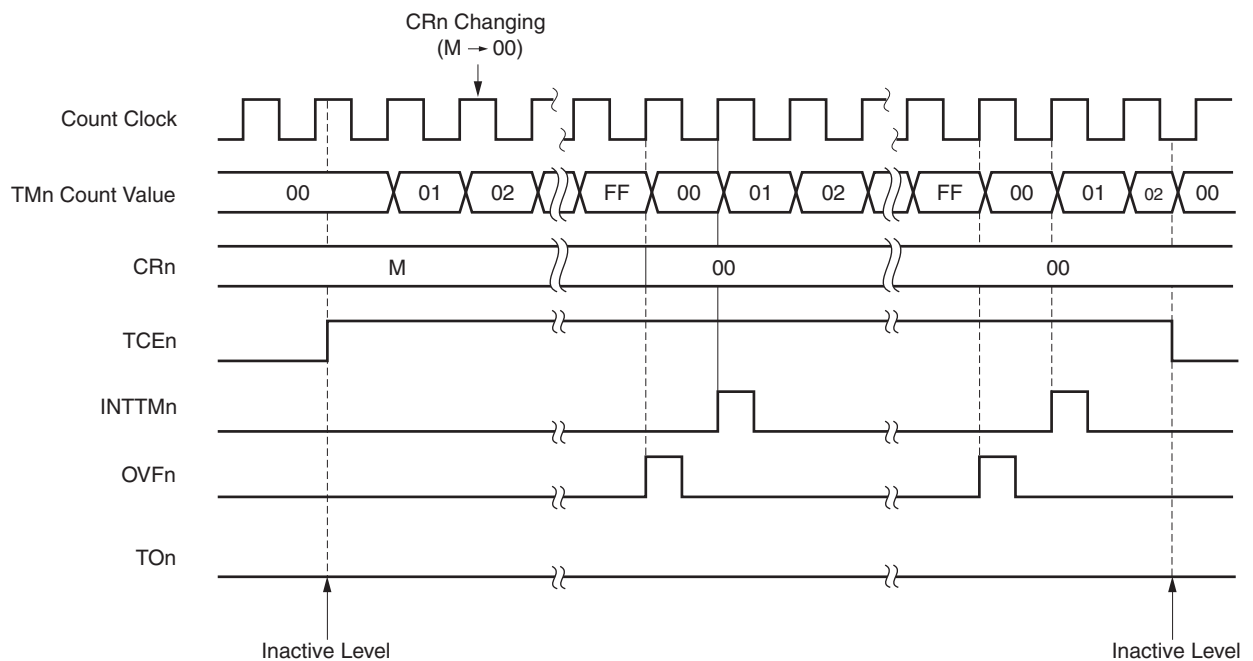
Remark: n = 50, 51

Figure 7-18: PWM Output Operation Timings (CRn = FFH, active high setting)
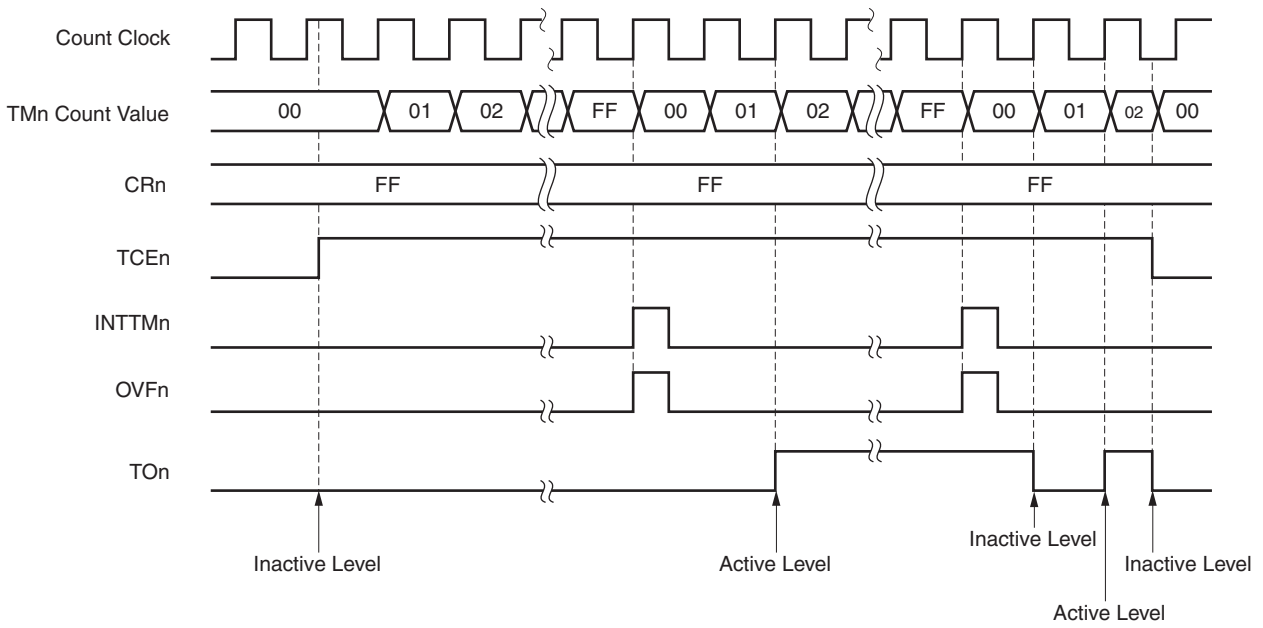
Remark: n = 50, 51

Figure 7-19: PWM Output Operation Timings (CRn changing, active high setting)
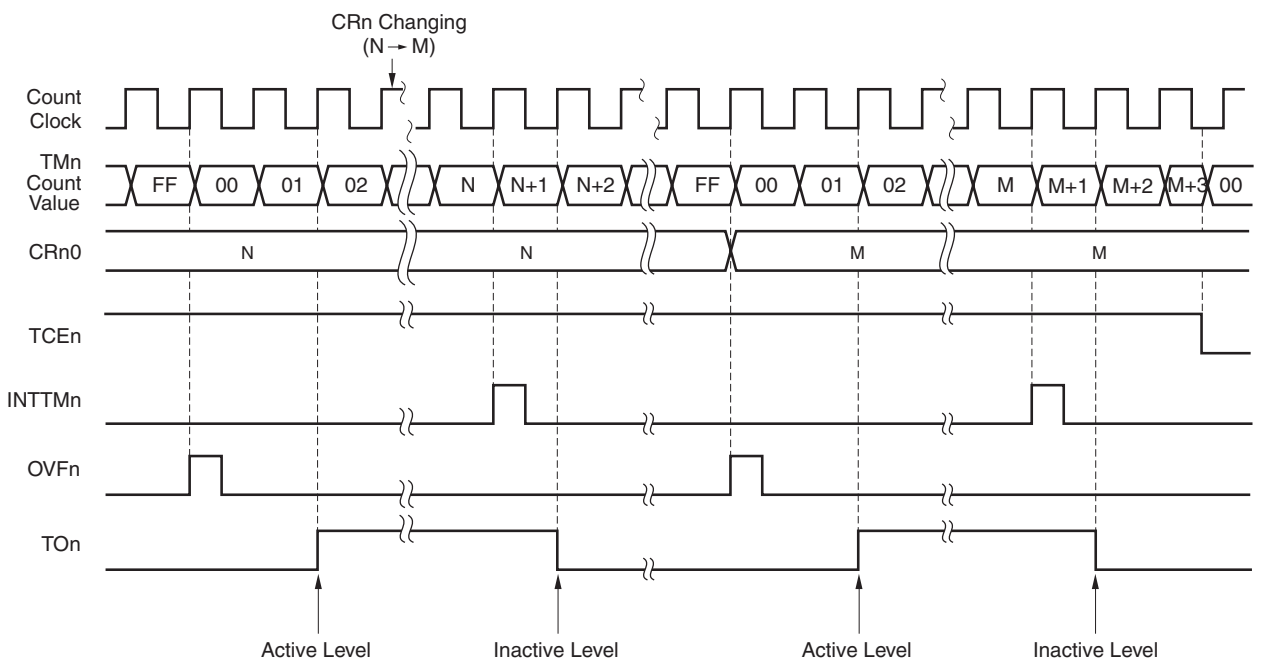
Remark: n = 50, 51

Caution: **If CRn is changed during TMn operation, the value changed is not reflected until TMn overflows.**

7.5 Cautions on 8-Bit Timer/Event Counters 50 and 51

(1) Timer start errors
An error with a maximum of one clock might occur concerning the time required for a match signal to be generated after the timer starts. This is because 8-bit timer registers 50 and 51 are started asynchronously with the count pulse.

Figure 7-20: 8-bit Timer Registers 50 and 51 Start Timings
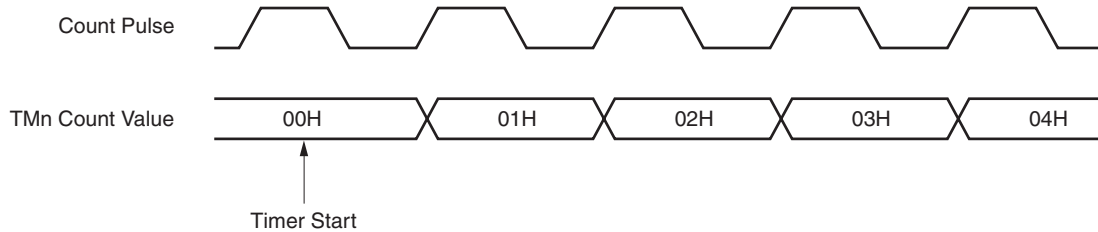
Remark: n = 50, 51

(2) Compare registers 50 and 51 sets
The 8-bit compare registers (CR50 and CR51) can be set to 00H.
Thus, when an 8-bit compare register is used as an event counter, one-pulse count operation can be carried out.

Figure 7-21: External Event Counter Operation Timings
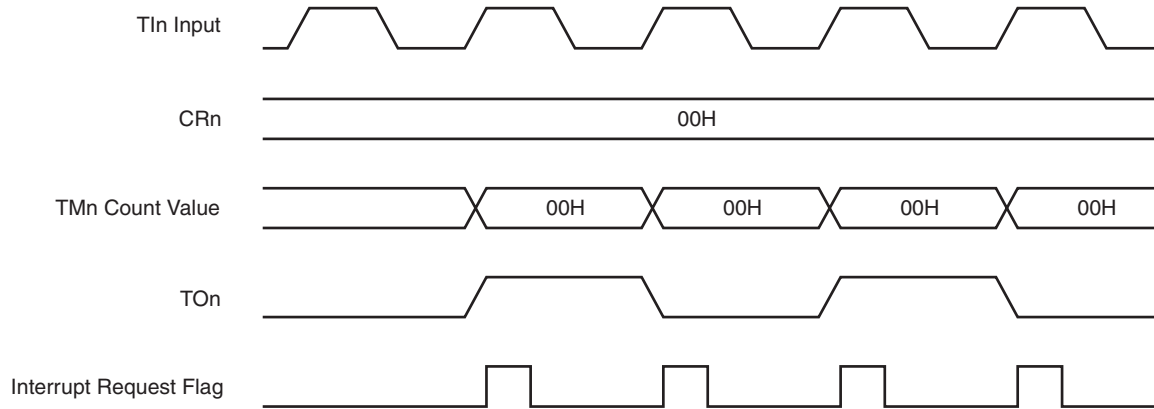
Remark: n = 50, 51

(3) Operation after compare register change during timer count operation
If the values after the 8-bit compare registers (CR50 and CR51) are changed are smaller than those of 8-bit timer registers (TM50 and TM51), TM50 and TM51 continue counting, overflow and then restarts counting from 0. Thus, if the value (M) after CR50 and CR51 change is smaller than that (N) before change it is necessary to restart the timer after changing CR50 and CR51.

Figure 7-22: Timings after Compare Register Change during Timer Count Operation
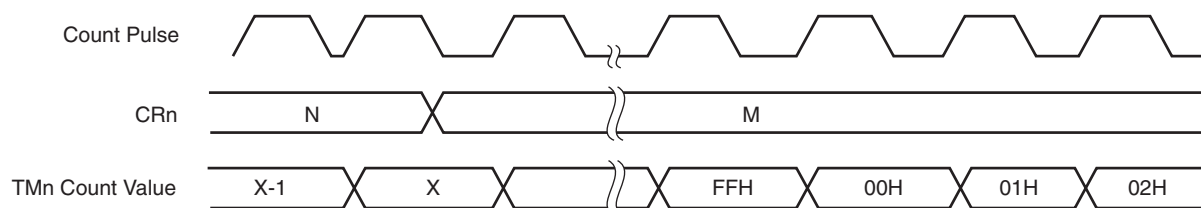
Remark: n = 50, 51

[Memo]

Chapter 8 Watch Timer

8.1 Watch Timer Functions

The watch timer has the following functions:

• Watch timer
• Interval timer

The watch timer and the interval timer can be used simultaneously.
The figure 8-1 shows Watch Timer Block Diagram.

Figure 8-1: Block Diagram of Watch Timer
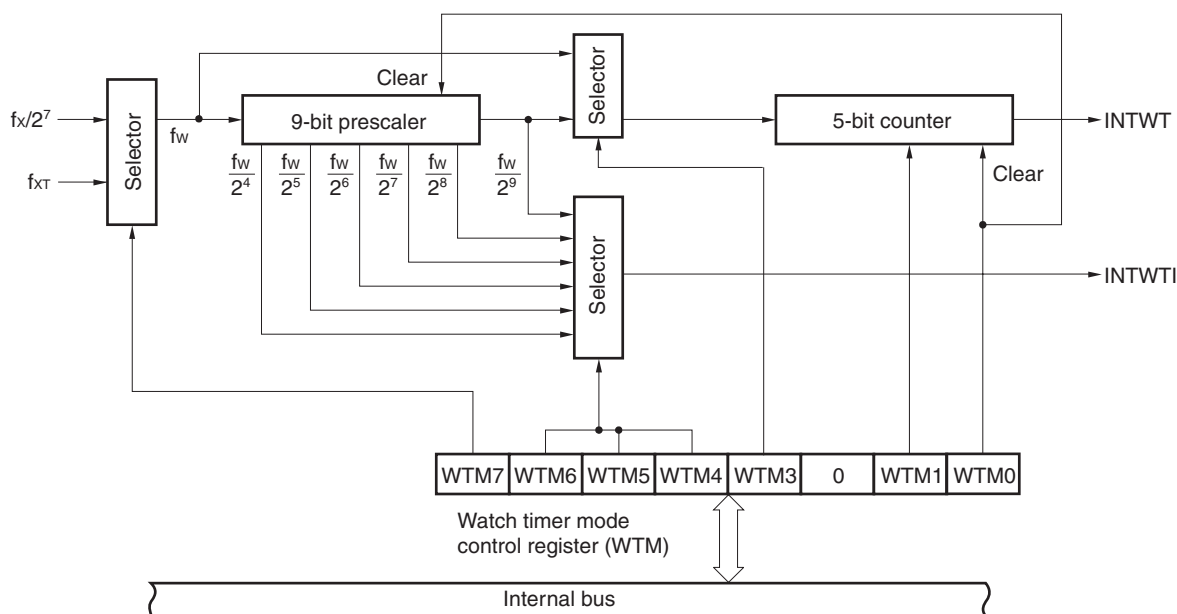
(1) Watch timer

When the main system clock or subsystem clock is used, interrupt requests (INTWT) are generated at 0.5 second intervals.

(2) Interval timer

Interrupt requests (INTWTI) are generated at the preset time interval.

Table 8-1: Interval Time Selection

Interval Time	When operated at fx=8.00 MHz	When operated at fxT=32.768 KHz
2^4/fw	256 μs	488 μs
2^5/fw	512 μs	977 μs
2^6/fw	1 ms	1,95 ms
2^7/fw	2 ms	3,91 ms
2^8/fw	4 ms	7,81 ms
2^9/fw	8,19 ms	15,6 ms

Remark: fx: Main system clock oscillation frequency
fxT: Subsystem clock oscillation frequency

8.2 Watch Timer Configuration

The watch timer consists of the following hardware.

Table 8-2: Watch Timer Configuration

Item	Configuration
Counter	5 bits x 1
Prescaler	9 bits x 1
Control register	Watch timer mode control register (WTM)

8.3 Watch Timer Mode Register (WTM)

This register sets the watch timer count clock, the watch timer operating mode, and prescaler interval time and enables/disables prescaler and 5-bit counter operations. WTM is set with a 1-bit or 8-bit memory manipulation instruction.
$\overline{\text{RESET}}$ input sets WTM to 00H.

Figure 8-2: Watch Timer Mode Control Register (WTM) Format

Symbol	7	6	5	4	3	2	1	0	Address	AfterReset	R/W
WTM	WTM7	WTM6	WTM5	WTM4	WTM3	0	WTM1	WTM0	FF41H	00H	R/W

WTM7	Watch Timer Count Clock Selection
0	Input clock set to $fx/2^7$
1	Input clock set to fx_T

WTM6	WTM5	WTM4	Prescaler Interval Time Selection	
			fx = 8.00 MHz Operation	fx$_T$ = 32.768 kHz Operation
0	0	0	2^4/fw (256 µs)	2^4/fw (488 µs)
0	0	1	2^5/fw (512 s)	2^5/fw (977 µs)
0	1	0	2^6/fw (1 ms)	2^6/fw (1.95 ms)
0	1	1	2^7/fw (2 ms)	2^7/fw (3.91 ms)
1	0	0	2^8/fw (4 ms)	2^8/fw (7.81 ms)
1	0	1	2^9/fw (8.19 ms)	2^9/fw (15.6 ms)
Other than above			Setting prohibited	

WTM3	Watch Operating Mode Selections
0	Normal operating mode (interrupt generation at 2^{14}/fw)
1	Fast feed operating mode (interrupt generation at 2^5/fw)

WTM1	5-Bit Counter Operation Control
0	Clear after operation stop
1	Operation enable

WTM0	Prescaler Operation Control
0	Clear after operation stop
1	Operation enable

Caution: **When the watch timer is used, the prescaler should not be cleared frequently. When rewriting WTM4 to WTM6 to other data, stop the timer operation beforehand.**

Remarks: **1.** fw: Watch timer clock frequency ($fx/2^7$ or fxt)
　　　　　　2. fx:　 Main system clock oscillation frequency
　　　　　　3. fxt: Subsystem clock oscillation frequency

8.4 Watch Timer Operations

8.4.1 Watch timer operation

When the 32.768-KHz subsystem clock is used, the timer operates as a watch timer with a 0.5-second interval.

The watch timer is generated interrupt request at the constant time interval.

When bit 0 (WTM0) and bit 1 (WTM1) of the watch timer mode control register (WTM) are set to 1, the count operation starts. When set to 0, the 5-bit counter is cleared and the count operation stops.

For simultaneous operation of the interval timer, zero-second start can be set only for the watch timer by setting WTM1 to 0. However, since the 9-bit prescaler is not cleared the first overflow of the watch timer (INTWT) after zero-second start may include an error of up to 2^9 x 1/fw.

8.4.2 Interval timer operation

The watch timer operates as interval timer which generates interrupt request repeatedly at an interval of the preset count value.

The interval time can be selected with bits 4 to 6 (WTM4 to WTM6) of the watch timer mode control register (WTM).

Table 8-3: Interval Timer Operation

WTM6	WTM5	WTM4	Interval Time	fx=8.00 MHz Operation	fxT=32.768 MHz Operation
0	0	0	2^4 x 1/fw	256 µs	488 µs
0	0	1	2^4 x 1/fw	512 µs	977 µs
0	1	0	2^4 x 1/fw	1 ms	1.95 ms
0	1	1	2^4 x 1/fw	2 ms	3.91 ms
1	0	0	2^4 x 1/fw	4 ms	7.81 ms
1	0	1	2^4 x 1/fw	8.19 ms	15.6 ms
Other than above			Setting prohibited		

Remark: fx: Main system clock oscillation frequency

fxT: Subsystem clock oscillation frequency

fw: Watch timer clock frequency

Figure 8-3: Operation Timing of Watch Timer/Interval Timer
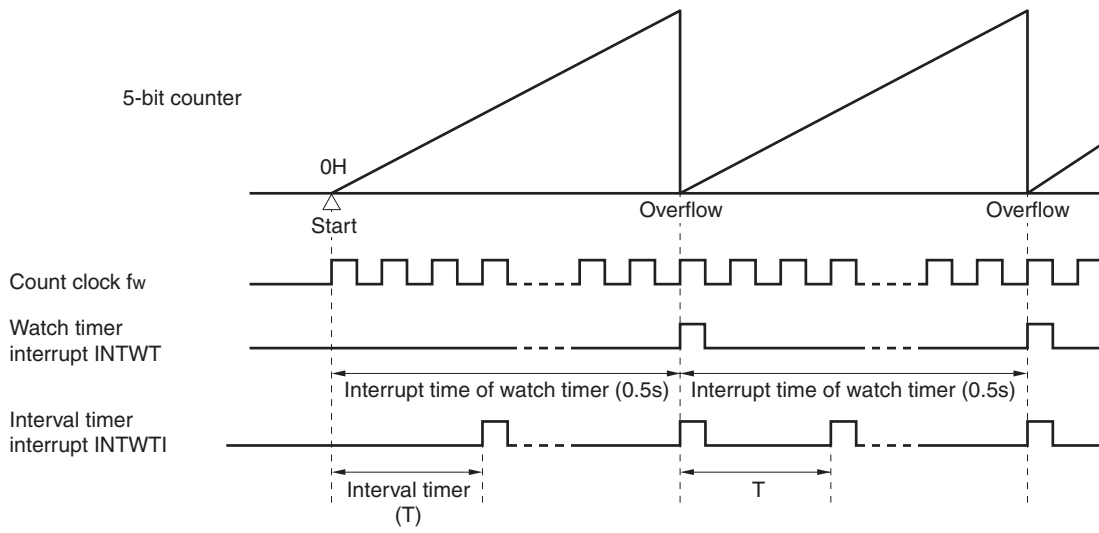
Remark: fw: Watch timer clock frequency

[Memo]

Chapter 9 Watchdog Timer

9.1 Watchdog Timer Functions

The watchdog timer has the following functions:

- Watchdog timer
- Interval timer

Caution: Select the watchdog timer mode or the interval timer mode with the watchdog timer mode register (WDTM).

(1) Watchdog timer mode

An inadvertent program loop is detected. Upon detection of the inadvertent program loop, a non-maskable interrupt request or $\overline{\text{RESET}}$ can be generated.

Table 9-1: Watchdog Timer Inadvertent Program Overrun Detection Times

Runaway Detection Time	
2^{12} x 1/fx	2^{12} x 1/fx (512 μs)
2^{13} x 1/fx	2^{13} x 1/fx (1 ms)
2^{14} x 1/fx	2^{14} x 1/fx (2 ms)
2^{15} x 1/fx	2^{15} x 1/fx (4 ms)
2^{16} x 1/fx	2^{16} x 1/fx (8.19 ms)
2^{17} x 1/fx	2^{17} x 1/fx (16.38 ms)
2^{18} x 1/fx	2^{18} x 1/fx (32.76 ms)
2^{20} x 1/fx	2^{20} x 1/fx (131 ms)

Remark: Figures in parentheses apply to operation with fx = 8.0 MHz.

(2) Interval timer mode

Interrupts are generated at the preset time intervals.

Table 9-2: Interval Times

Interval Time	
2^{12} x 1/fx	2^{12} x 1/fx (512 μs)
2^{13} x 1/fx	2^{13} x 1/fx (1 ms)
2^{14} x 1/fx	2^{14} x 1/fx (2 ms)
2^{15} x 1/fx	2^{15} x 1/fx (4 ms)
2^{16} x 1/fx	2^{16} x 1/fx (8.19 ms)
2^{17} x 1/fx	2^{17} x 1/fx (16.38 ms)
2^{18} x 1/fx	2^{18} x 1/fx (32.76 ms)
2^{20} x 1/fx	2^{20} x 1/fx (131 ms)

Remark: Figures in parentheses apply to operation with fx = 8.0 MHz.

9.2 Watchdog Timer Configuration

The watchdog timer consists of the following hardware.

Table 9-3: Watchdog Timer Configuration

Item	Configuration
Control register	Timer clock select register (WDCS) Watchdog timer mode register (WDTM)

Figure 9-1: Watchdog Timer Block Diagram
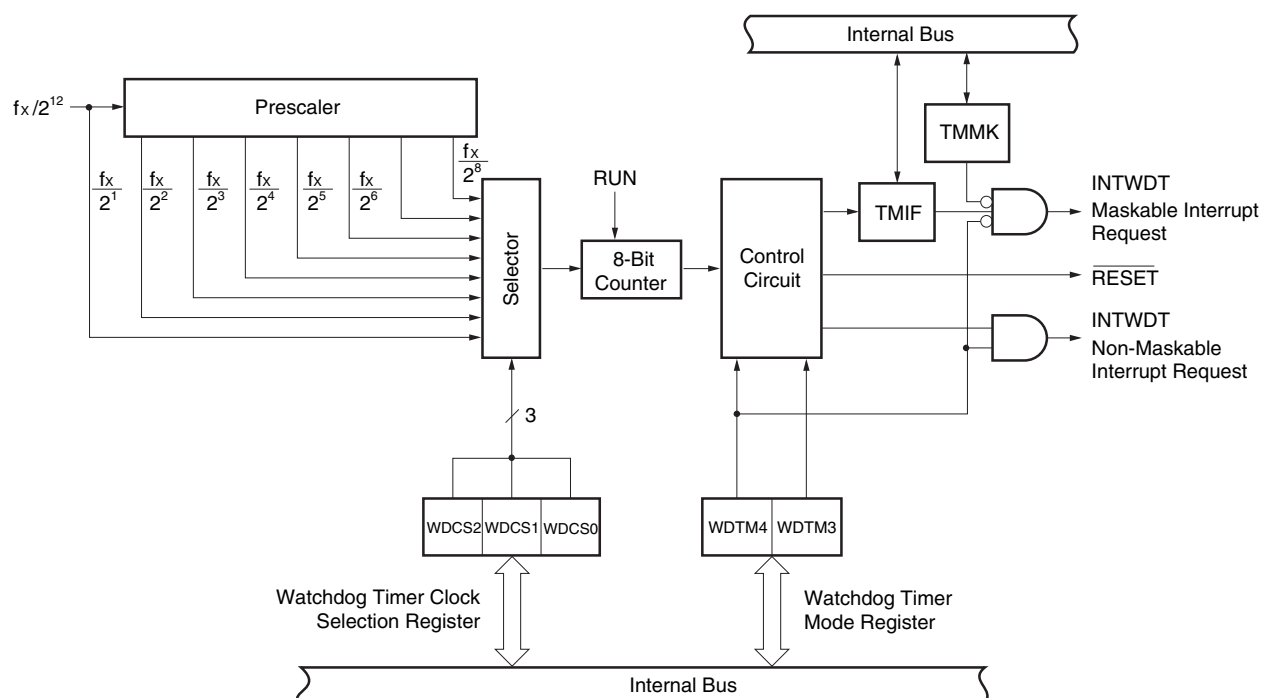
9.3 Watchdog Timer Control Registers

The following two types of registers are used to control the watchdog timer.

- Watchdog timer clock select register (WDCS)
- Watchdog timer mode register (WDTM)

(1) Watchdog timer clock select register (WDCS)
This register sets the watchdog timer count clock.
WDCS is set with an 8-bit memory manipulation instruction.
$\overline{\text{RESET}}$ input sets WDCS to 00H.

Figure 9-2: Watchdog Timer Clock Select Register Format

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

WDCS2	WDCS1	WDCS0	Overflow time of watchdog 1 interval timer
0	0	0	$fx/2^{12}$ (512 µs)
0	0	1	$fx/2^{13}$ (1 ms)
0	1	0	$fx/2^{14}$ (2 ms)
0	1	1	$fx/2^{15}$ (4 ms)
1	0	0	$fx/2^{16}$ (8.19 ms)
1	0	1	$fx/2^{17}$ (16.38 ms)
1	1	0	$fx/2^{18}$ (32.76 ms)
1	1	1	$fx/2^{20}$ (131 ms)

Caution: When rewriting WDCS to other data, stop the timer operation beforehand.

Remarks: 1. fx: Main system clock oscillation frequency
2. Figures in parentheses apply to operation with fx = 8.0 MHz.

(2) Watchdog timer mode register (WDTM)

This register sets the watchdog timer operating mode and enables/disables counting.

WDTM is set with a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets WDTM to 00H.

Figure 9-3: Watchdog Timer Mode Register Format

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

WDTM4	WDTM3	Watchdog Timer Operation Mode Selection [Note 1]
0	X	Interval timer mode (Maskable interrupt occurs upon generation of an overflow)
1	0	Watchdog timer mode 1 (Non-maskable interrupt occurs upon generation of an overflow)
1	1	Watchdog timer mode 2 (Reset operation is activated upon generation of an overflow)

RUN	Watchdog Timer Operation Mode Selection [Note 2]
0	Count stop
1	Counter is cleared and counting starts

Notes: 1. Once set to 1, WDTM3 and WDTM4 cannot be cleared to 0 by software.
2. Once set to 1, RUN cannot be cleared to 0 by software.
Thus, once counting starts, it can only be stopped by $\overline{\text{RESET}}$ input.

Caution: **When 1 is set in RUN so that the watchdog timer is cleared, the actual overflow time is up to 0.5 % shorter than the time set by watchdog timer clock select register.**

Remark: x = don't care.

9.4 Watchdog Timer Operations

9.4.1 Watchdog timer operation
When bit 4 (WDTM4) of the watchdog timer mode register (WDTM) is set to 1, the watchdog timer is operated to detect any inadvertent program loop.

The watchdog timer count clock (inadvertent program loop detection time interval) can be selected with bits 0 to 2 (WDCS0 to WDCS2) of the timer clock select register (WDCS).

Watchdog timer starts by setting bit 7 (RUN) of WDTM to 1. After the watchdog timer is started, set RUN to 1 within the set overrun detection time interval. The watchdog timer can be cleared and counting is started by setting RUN to 1. If RUN is not set to 1 and the inadvertent program loop detection time is past, system reset or a non-maskable interrupt request is generated according to the WDTM bit 3 (WDTM3) value.

The watchdog timer can be cleared when RUN is set to 1.

The watchdog timer continues operating in the HALT mode but it stops in the STOP mode. Thus, set RUN to 1 before the STOP mode is set, clear the watchdog timer and then execute the STOP instruction.

Cautions: **1. The actual overrun detection time may be shorter than the set time by a maximum of 0.5 %.**
2. When the subsystem clock is selected for CPU clock, watchdog timer count operation is stopped.

Table 9-4: Watchdog Timer Overrun Detection Time

WDCS2	WDCS1	WDCS0	Runaway Detection Time
0	0	0	$fx/2^{12}$ (512 μs)
0	0	1	$fx/2^{13}$ (1 ms)
0	1	0	$fx/2^{14}$ (2 ms)
0	1	1	$fx/2^{15}$ (4 ms)
1	0	0	$fx/2^{16}$ (8.19 ms)
1	0	1	$fx/2^{17}$ (16.38 ms)
1	1	0	$fx/2^{18}$ (32.76 ms)
1	1	1	$fx/2^{20}$ (131 ms)

Remarks: **1.** fx: Main system clock oscillation frequency
2. Figures in parentheses apply to operation with fx = 8.0 MHz.

9.4.2 Interval timer operation

The watchdog timer operates as an interval timer which generates interrupts repeatedly at an interval of the preset count value when bit 3 (WDTM3) of the watchdog timer mode register (WDTM) is set to 0, respectively.

When the watchdog timer operates as interval timer, the interrupt mask flag (TMMK4) and priority specify flag (TMPR4) are validated and the maskable interrupt request (INTWDT) can be generated. Among maskable interrupts, the INTWDT default has the highest priority.

The interval timer continues operating in the HALT mode but it stops in STOP mode. Thus, set bit 7 (RUN) of WDTM to 1 before the STOP mode is set, clear the interval timer and then execute the STOP instruction.

Cautions: **1. Once bit 4 (WDTM4) of WDTM is set to 1 (with the watchdog timer mode selected), the interval timer mode is not set unless RESET input is applied.**
2. The interval time just after setting with WDTM may be shorter than the set time by a maximum of 0.5 %.
3. When the subsystem clock is selected for CPU clock, watchdog timer count operation is stopped.

Table 9-5: Interval Timer Interval Time

WDCS2	WDCS1	WDCS0	Interval Time
0	0	0	$fx/2^{12}$ (512 μs)
0	0	1	$fx/2^{13}$ (1 ms)
0	1	0	$fx/2^{14}$ (2 ms)
0	1	1	$fx/2^{15}$ (4 ms)
1	0	0	$fx/2^{16}$ (8.19 ms)
1	0	1	$fx/2^{17}$ (16.38 ms)
1	1	0	$fx/2^{18}$ (32.76 ms)
1	1	1	$fx/2^{20}$ (131 ms)

Remarks: **1.** fx: Main system clock oscillation frequency
2. Figures in parentheses apply to operation with fx = 8.0 MHz.

[Memo]

Chapter 10 Clock Output Control Circuit

10.1 Clock Output Control Circuit Functions

The clock output control circuit is intended for carrier output during remote controlled transmission and clock output for supply to peripheral LSI. Clocks selected with the clock output selection register (CKS) are output from the PCL/P120/S7 pin.
Follow the procedure below to output clock pulses.

(1) Select the clock pulse output frequency (with clock pulse output disabled) with bits 0 to 3 (CCS0 to CCS2) of CKS.
(2) Set the P120 output latch to 0.
(3) Set bit 0 (PM120) of port mode register 120 to 0 (set to output mode).
(4) Set bit 4 (CLOE) of clock output selection register to 1.

Caution: **Clock output cannot be used when setting the output latch to 1.**

Remark: When clock output enable/disable is switched, the clock output control circuit does not output pulses with small widths (See the portions marked with ***** in Figure 12-1).

Figure 10-1: Remote Controlled Output Application Example
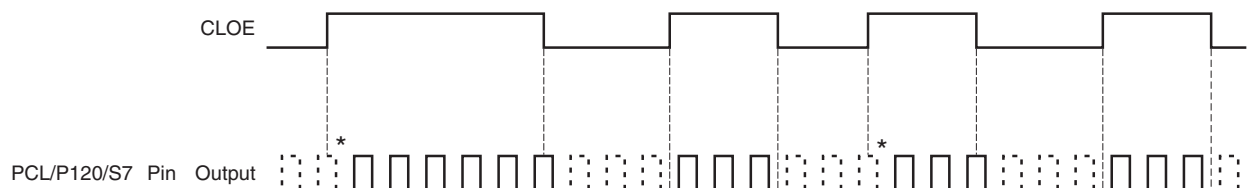
10.2 Clock Output Control Circuit Configuration

The clock output control circuit consists of the following hardware.

Table 10-1: Clock Output Control Circuit Configuration

Item	Configuration
Control register	Clock output selection register (CKS) Port mode register 3 (PM3)

Figure 10-2: Clock Output Control Circuit Block Diagram
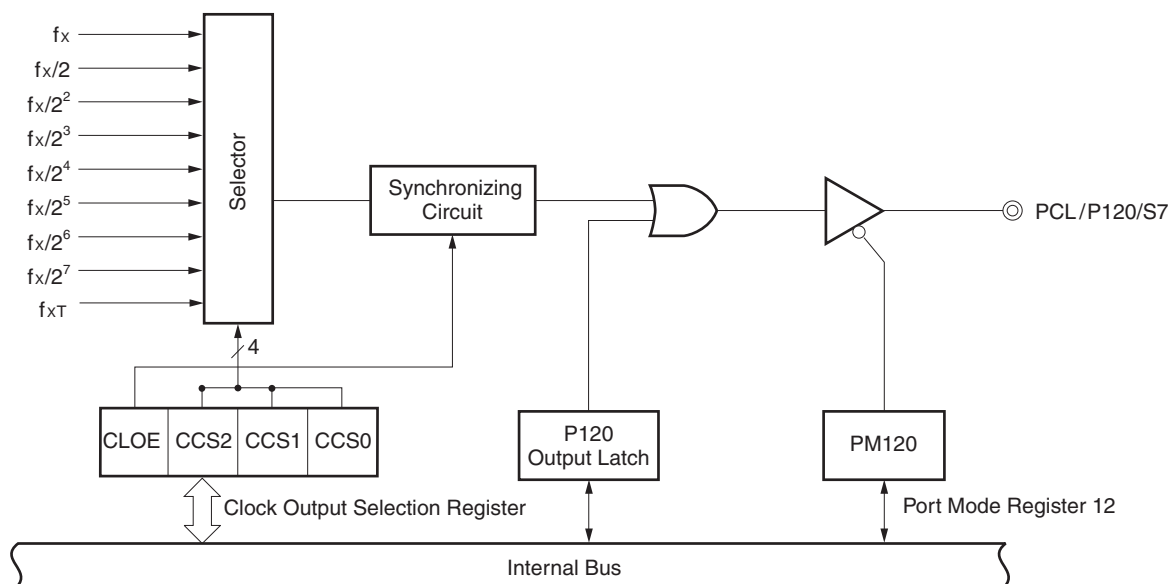
10.3 Clock Output Function Control Registers

The following two types of registers are used to control the clock output function.

- Clock output selection register (CKS)
- Port mode register 12 (PM12)

(1) Clock Output Selection Register (CKS)

This register sets PCL output clock.

CKS is set with a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets CKS to 00H.

Caution: **When enabling PCL output, set CCS50 to CCS52, then set 1 in CLOE with a 1-bit memory manipulation instruction.**

Figure 10-3: Clock Output Selection Register Format

Symbol	⑦	6	5	4	3	2	1	0	Address	After Reset	R/W
CKS	0	0	0	CLOE	CCS3	CCS2	CCS1	CCS0	FF40H	00H	R/W

CCS3	CCS2	CCS1	CCS0	PCL Output Clock Selection
0	0	0	0	f_X (8 MHz)
0	0	0	1	$f_X/2^1$ (4 MHz)
0	0	1	0	$f_X/2^2$ (2 MHz)
0	0	1	1	$f_X/2^3$ (1 MHz)
0	1	0	0	$f_X/2^4$ (500 KHz)
0	1	0	1	$f_X/2^5$ (250 KHz)
0	1	1	0	$f_X/2^6$ (125 KHz)
0	1	1	1	$f_X/2^7$ (62.5 KHz)
1	0	0	0	f_{XT} (32.7 KHz)
Other than above				Setting prohibited

CLOE	PCL Output Control
0	Output disable
1	Output enable

Remarks: **1.** f_X: Main system clock oscillation frequency

2. f_{XT}: subsystem clock oscillation frequency.

3. Figures in parentheses apply to operation with f_X = 8.0 MHz and f_{XT} = 32.718 kHz.

(2) Port Mode Register 12 (PM12)

This register sets port 12 input/output in 1-bit units.

When using the P120/PCL/S7 pin for clock output function, set PM120 and output latch of P120 to 0. PM12 is set with a 1-bit or 8-bit memory manipulation instruction.

RESET input sets PM12 to FFH.

Figure 10-4: Port Mode Register 12 Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
PM12	PM127	PM126	PM125	PM124	PM123	PM122	PM121	PM120	FF2CH	FFH	R/W

PM12n	P12n pin input/output mode selection (n = 0 to 7)
0	Output mode (output buffer ON)
1	Input mode (output buffer OFF)
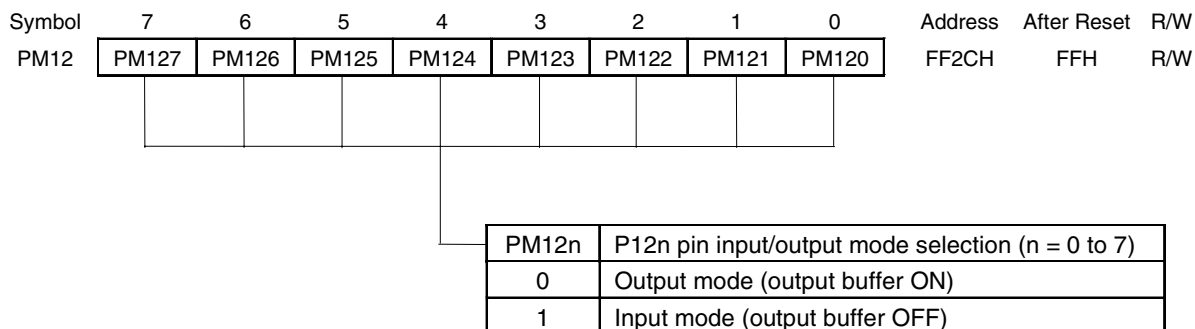
(3) Port Function Register 12 (PF12)

This register sets the port function of port 12 in 1-bit units.

When using the PCL output, the register PF12 has to be set to port function.

PF12 is with an 1-bit or an 8-bit memory manipulation instruction.

RESET input sets PM12 to 00H.

Figure 10-5: Port Function Register 12 (PF12) Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
PF12	PF127	PF126	PF125	PF124	PF123	PF122	PF121	PF120	FF5CH	00H	R/W

PF12n	P12n port function selection (n = 0 to 7)
0	Port mode
1	LCD mode
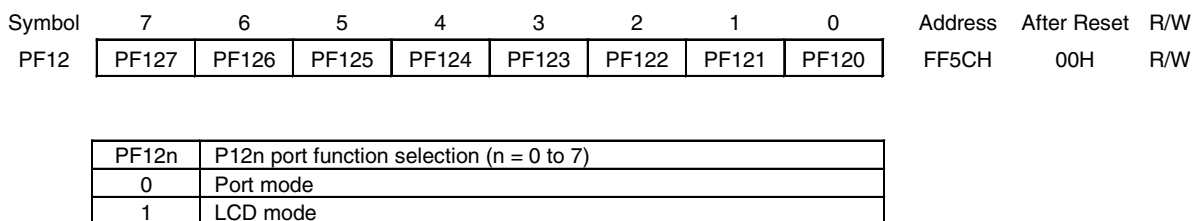
Note: **For the µPD1616 set always 00H to PF12.**

[Memo]

Chapter 11 A/D Converter

11.1 A/D Converter Functions

The A/D converter is an 8-bit resolution converter that converts analog inputs into digital values. It can control up to 4 analog input channels (ANI0 to ANI3).
This A/D converter has the following functions:

(1) A/D conversion with 8-bit resolution
One channel of analog input is selected from ANI0 to ANI3, and A/D conversion is repeatedly executed with a resolution of 8 bits. Each time the conversion has been completed, an interrupt request (INTAD) is generated.

(2) Power-fail detection function
This function is to detect for example a voltage drop in the battery of an automobile. The result of A/D conversion (value of the ADCR1 register) and the value of PFT register (PFT: power-fail compare threshold value register) are compared. If the condition for comparison is satisfied, the INTAD is generated.

Figure 11-1: A/D Converter Block Diagram
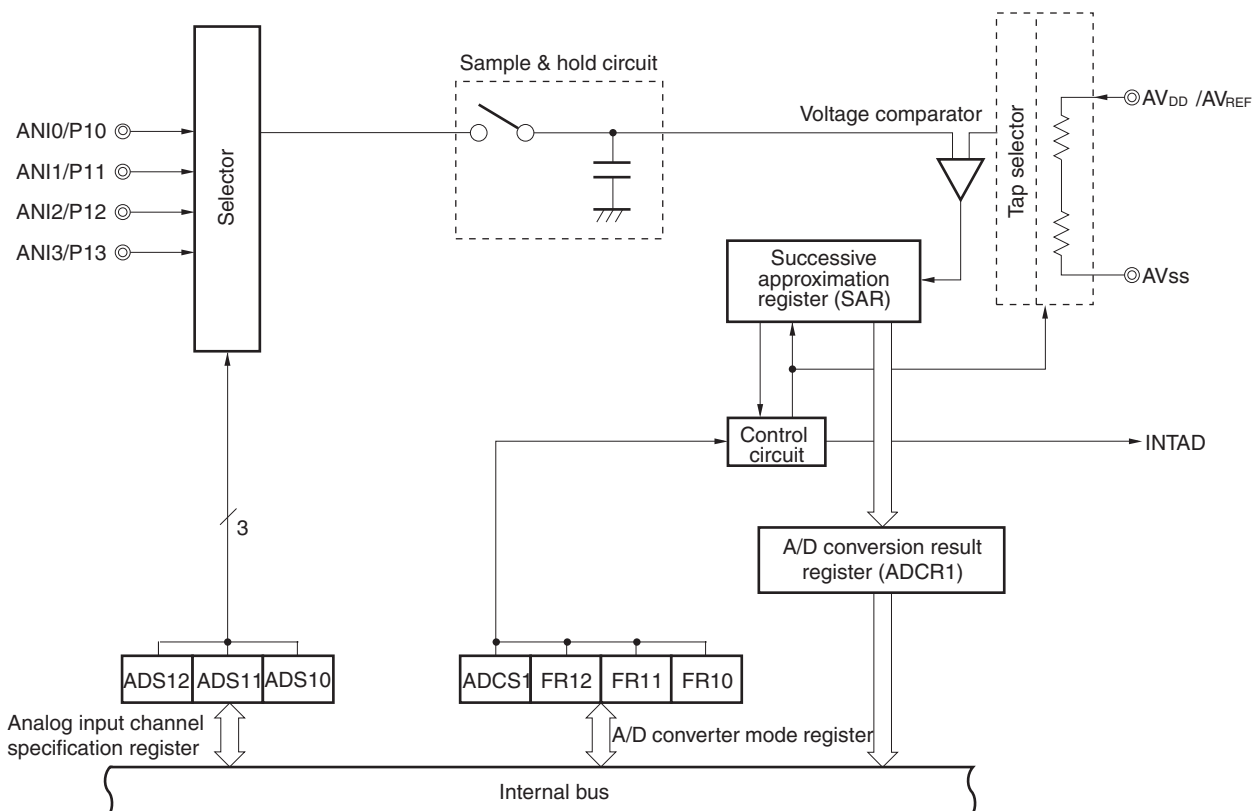
Figure 11-2: Power-Fail Detection Function Block Diagram
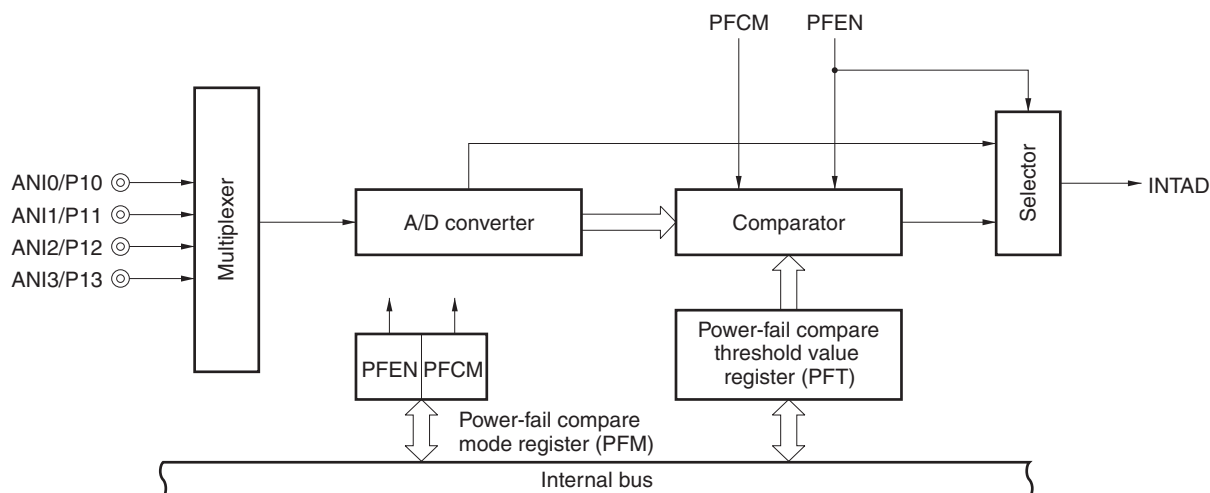
11.2 A/D Converter Configuration

A/D converter consists of the following hardware.

Table 11-1: A/D Converter Configuration

Item	Configuration
Analog input	8 channels (ANI0 to ANI7)
Register	Successive approximation register (SAR) A/D conversion result register (ADCR1)
Control register	A/D converter mode register (ADM1) Analog input channel specification register (ADS1) Power-fail compare mode register (PFM) Power-fail compare threshold value register (PFT)

(1) Successive approximation register (SAR)
This register compares the analog input voltage value to the voltage tap (compare voltage) value applied from the series resistor string, and holds the result from the most significant bit (MSB). When up to the least significant bit (LSB) is set (end of A/D conversion), the SAR contents are transferred to the A/D conversion result register.

(2) A/D conversion result register (ADCR1)
This register holds the A/D conversion result. Each time when the A/D conversion ends, the conversion result is loaded from the successive approximation register.
ADCR1 is read with an 8-bit memory manipulation instruction.
RESET input clears ADCR1 to 00H.

Caution: **If a write operation is executed to the A/D converter mode register (ADM1) and the analog input channel specification register (ADS1) the contents of ADCR1 are undefined. Read the conversion result before a write operation is executed to ADM1 and ADS1. If a timing other than the above is used, the correct conversion result may not be read.**

(3) Sample & hold circuit

The sample & hold circuit samples each analog input sequentially applied from the input circuit, and sends it to the voltage comparator. This circuit holds the sampled analog input voltage value during A/D conversion.

(4) Voltage comparator

The voltage comparator compares the analog input to the series resistor string output voltage.

(5) Series resistor string

The series resistor string is in AV_{DD} to AV_{SS}, and generates a voltage to be compared to the analog input.

(6) ANI0 to ANI3 pins

These are four analog input pins to input analog signals to the A/D converter. ANI0 to ANI3 are alternate-function pins that can also be used for digital input.

Caution: **Use ANI0 to ANI3 input voltages within the specification range. If a voltage higher than AV_{DD} or lower than AV_{SS} is applied (even if within the absolute maximum rating range), the conversion value of that channel will be undefined and the conversion values of other channels may also be affected.**

(7) AV_{DD}/AV_{REF} pin

This pin inputs the A/D converter reference voltage and is used as the AD-converter power supply pin.

It converts signals input to ANI0 to ANI3 into digital signals according to the voltage applied between AV_{DD}/AV_{REF} and AV_{SS}.

Keep the AV_{DD}/AV_{REF} pin always at the same potential on the V_{DD} pin, even when the AD-converter is no used.

(8) AV_{SS} pin

This is the GND potential pin of the A/D converter. Always keep it at the same potential as the V_{SS} pin even when not using the A/D converter.

11.3 A/D Converter Control Registers

The following 4 types of registers are used to control A/D converter.

- A/D converter mode register (ADM1)
- Analog input channel specification register (ADS1)
- Power-fail compare mode register (PFM)
- Power-fail compare threshold value register (PFT)

(1) A/D converter mode register (ADM1)

This register sets the conversion time for analog input to be A/D converted, conversion start/stop and external trigger. ADM1 is set with 1-bit or 8-bit memory manipulation instruction.
$\overline{\text{RESET}}$ input clears ADM1 to 00H.

Figure 11-3: A/D Converter Mode Register (ADM1) Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
ADM1	ADCS1	0	FR12	FR11	FR10	0	0	0	FF80H	00H	R/W

ADCS1	A/D Conversion Operation Control
0	Stop conversion operation
1	Enable conversion operation

FR12	FR11	FR10	Conversion Time Selection [Note]
0	0	0	144/fx
0	0	1	120/fx
0	1	0	96/fx
1	0	0	288/fx
1	0	1	240/fx
1	1	0	192/fx
Other than above			Setting prohibited

Note: Set FR10 to FR12 that the A/D conversion time is 15 µs or more.

Caution: Bits 0 to 2 and bit 6 must be set to 0.

Remark: fx: Main system clock oscillation frequency

(2) Analog input channel specification register (ADS1)

This register specifies the analog voltage input port for A/D conversion.

ADS1 is set with an 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input clears ADS1 to 00H.

Figure 11-4: Analog Input Channel Specification Register (ADS1) Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
ADS1	0	0	0	0	0	0	ADS11	ADS10	FF81H	00H	R/W

ADS11	ADS10	Analog Input Channel Specification
0	0	ANI0
0	1	ANI1
1	0	ANI2
1	1	ANI3

Caution:　Bits 2 to 7 must be set to 0.

(3) Power-fail compare mode register (PFM)

The power-fail compare mode register (PFM) controls a comparison operation. PFM is set with an 8-bit manipulation instruction.

$\overline{\text{RESET}}$ input clears PFM to 00H.

Figure 11-5: Power-Fail Compare Mode Register (PFM) Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
PFM	PFEN	PFCM	0	0	0	0	0	0	FF82H	00H	R/W

PFEN	Enables Power-Fail Comparison
0	Disables power-fail comparison (used as normal A/D converter)
1	Enables power-fail comparison (used to detect power failure)

PFCM		Power-Fail Compare Mode Selection
0	ADCR1 ≥ PFT	Generates interrupt request signal INTAD
0	ADCR1 < PFT	Does not generate interrupt request signal INTAD
1	ADCR1 ≥ PFT	Does not generate interrupt request signal INTAD
1	ADCR1 < PFT	Generates interrupt request signal INTAD

Caution: Bits 0 to 5 must be set to 0.

(4) Power-fail compare threshold value register (PFT)

The power-fail compare threshold value register (PFT) sets a threshold value against which the result of A/D conversion is to be compared.

PFT is set with an 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input clears PFT to 00H.

Figure 11-6: Power-fail compare threshold value register (PFT)

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
PFT	PFT7	PFT6	PFT5	PFT4	PFT3	PFT2	PFT1	PFT0	FF83H	00H	R/W

11.4 A/D Converter Operations

11.4.1 Basic operations of A/D converter

<1> Select one channel for A/D conversion with the analog input channel specification register (ADS1).

<2> The voltage input to the selected analog input channel is sampled by the sample & hold circuit.

<3> When sampling has been done for a certain time, the sample & hold circuit is placed in the hold state and the input analog voltage is held until the A/D conversion operation is ended.

<4> Bit 7 of the successive approximation register (SAR) is set internally so that the tap selector starts with a series resistor string voltage tap of (1/2) AV_{DD}.

<5> The voltage difference between the series resistor string voltage tap and analog input is compared with the voltage comparator. If the analog input is greater than (1/2) AV_{DD}, the MSB of SAR remains set. If the analog input is smaller than (1/2) AV_{DD}, the MSB is reset.

<6> Next, bit 6 of SAR is automatically set, and the operation proceeds to the next comparison. The series resistor string voltage tap is selected according to the preset value of bit 7, as described below.
 • Bit 7 = 1: (3/4) AV_{DD}
 • Bit 7 = 0: (1/4) AV_{DD}
The voltage tap and analog input voltage are compared and bit 6 of SAR is manipulated as follows.
 • Analog input voltage ≥ Voltage tap: Bit 6 = 1
 • Analog input voltage < Voltage tap: Bit 6 = 0

<7> Comparison is continued in this way up to bit 0 of SAR.

<8> Upon completion of the comparison of 8 bits, an effective digital result value remains in SAR, and the result value is transferred to and latched in the A/D conversion result register (ADCR1). At the same time, the A/D conversion end interrupt request (INTAD) can also be generated.

Caution: **The first A/D conversion value just after A/D conversion is undefined.**

Figure 11-7: Basic Operation of 8-Bit A/D Converter
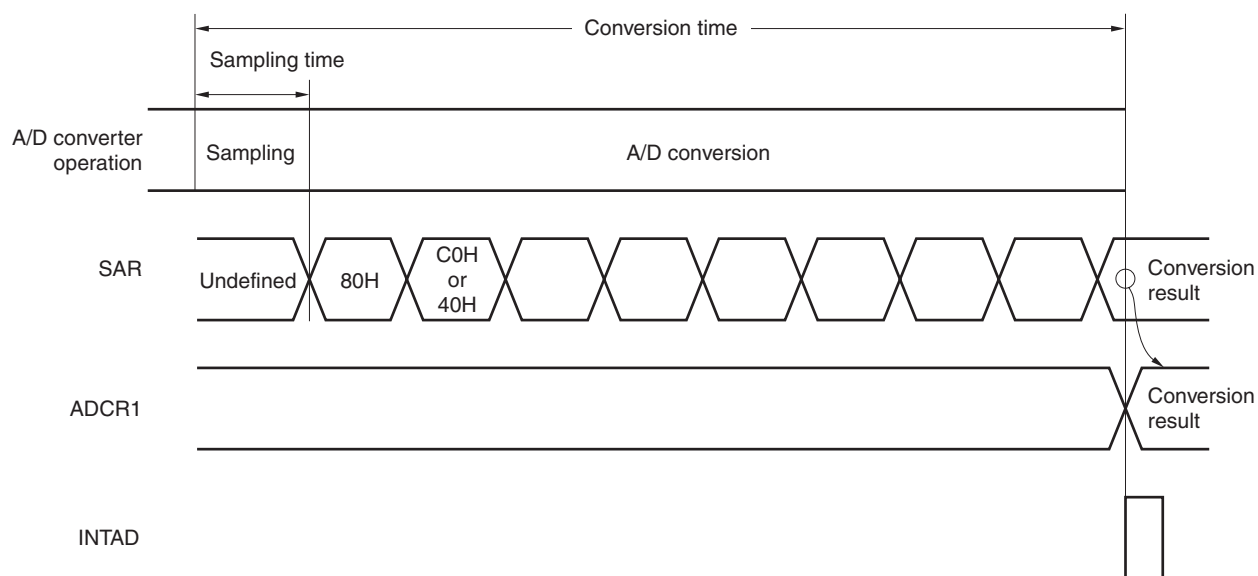
A/D conversion operations are performed continuously until bit 7 (ADCS1) of the A/D converter mode register (ADM1) is reset (to 0) by software.

If a write operation to the ADM1 and analog input channel specification register (ADS1) is performed during an A/D conversion operation, the conversion operation is initialized, and if the ADCS1 bit is set (to 1), conversion starts again from the beginning.

RESET input sets the A/D conversion result register (ADCR1) to 00H.

11.4.2 Input voltage and conversion results

The relation between the analog input voltage input to the analog input pins (ANI0 to ANI3) and the A/D conversion result (stored in the A/D conversion result register (ADCR1)) is shown by the following expression.

$$ADCR1 = INT \left(\frac{VIN}{AV_{DD}} \times 256 + 0.5 \right)$$

or

$$(ADCR1 - 0.5) \times \frac{AV_{DD}}{256} \leq V_{IN} < (ADCR1 + 0.5) \times \frac{AV_{DD}}{256}$$

where, INT( ) : Function which returns integer part of value in parentheses
        V_{IN} : Analog input voltage
        AV_{DD} : AV_{DD} pin voltage
        ADCR1 : A/D conversion result register (ADCR1) value

Figure 11-8 shows the relation between the analog input voltage and the A/D conversion result.

Figure 11-8: Relation between Analog Input Voltage and A/D Conversion Result
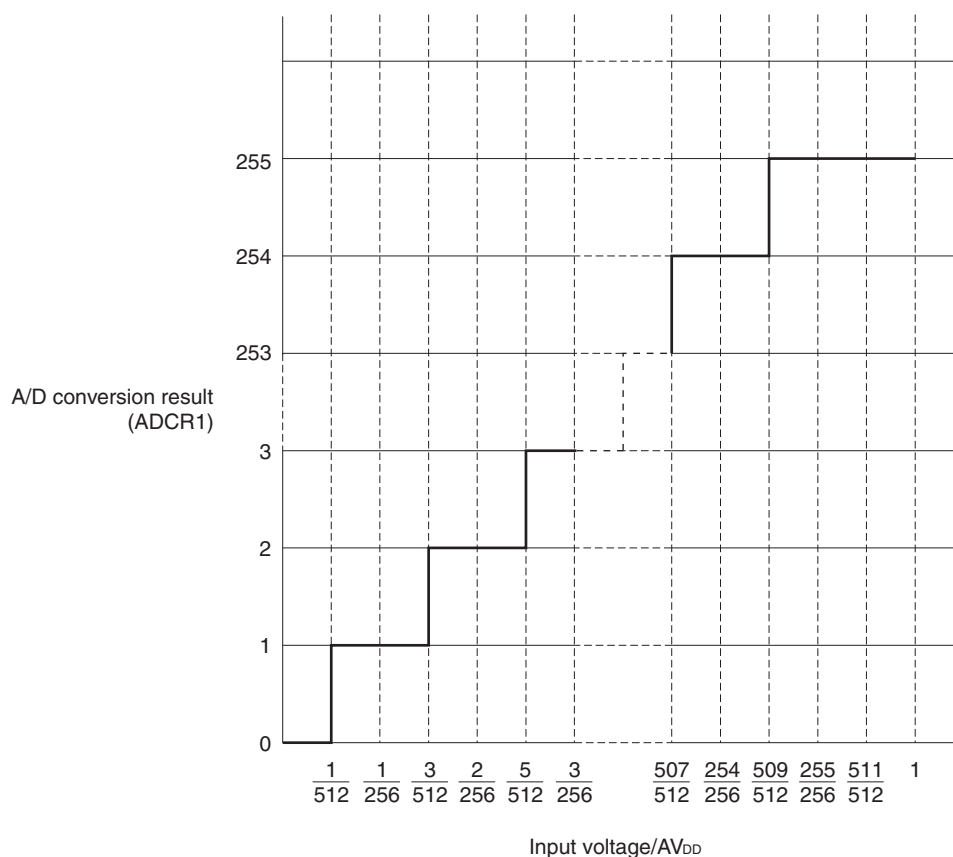
11.4.3 A/D converter operation mode

The operation mode of the A/D converter is the select mode. One analog input channel is selected from among ANI0 to ANI3 with the analog input channel specification register (ADS1) and A/D conversion is performed.

The following two types of functions can be selected by setting the PFEN flag of the PFM register.

(1) Normal 8-bit A/D converter (PFEN = 0)
(2) Power-fail detection function (PFEN = 1)

(1) A/D conversion (when PFEN = 0)

When bit 7 (ADCS1) of the A/D converter mode register (ADM1) is set to 1 and bit 7 of the power-fail compare mode register (PFM) is set to 0, A/D conversion of the voltage applied to the analog input pin specified with the analog input channel specification register (ADS1) starts.

Upon the end of the A/D conversion, the conversion result is stored in the A/D conversion result register (ADCR1), and the interrupt request signal (INTAD) is generated. After one A/D conversion operation is started and ended, the next conversion operation is immediately started. A/D conversion operations are repeated until new data is written to ADS1.

If ADS1 is rewritten during A/D conversion operation, the A/D conversion operation under execution is stopped, and A/D conversion of a newly selected analog input channel is started.

If data with ADCS1 set to 0 is written to ADM1 during A/D conversion operation, the A/D conversion operation stops immediately.

(2) Power-fail detection function (when PFEN = 1)

When bit 7 (ADCS1) of the A/D converter mode register (ADM1) and bit 7 (PFEN) of the power-fail compare mode register (PFM) are set to 1, A/D conversion of the voltage applied to the analog input pin specified with the analog input channel specification register (ADS1) starts.

Upon the end of the A/D conversion, the conversion result is stored in the A/D conversion result register (ADCR1), compared with the value of the power-fail compare threshold value register (PFT), the INTAD is generated under the condition specified by the PFCM flag of the PFM register.

Caution: **When executing power-fail comparison, the interrupt request signal (INTAD) is not generated on completion of the first conversion after ADCS1 has been set to 1. INTAD is valid from completion of the second conversion.**

Figure 11-9: A/D Conversion
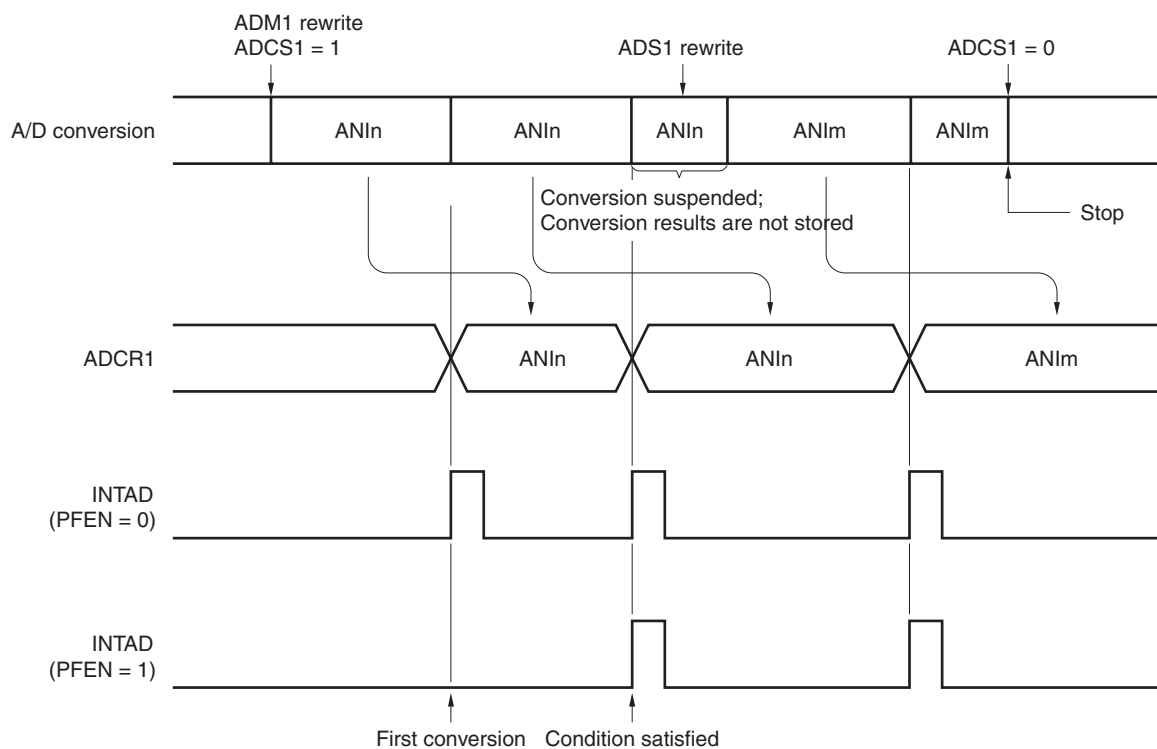
Remarks: **1.** n = 0, 1, ..., 7
             2. m = 0, 1, ..., 7

11.5 A/D Converter Precautions

(1) Current consumption in standby mode

A/D converter stops operating in the standby mode. At this time, current consumption can be reduced by setting bit 7 (ADCS1) of the A/D converter mode register (ADM1) to 0 to stop conversion.

Figure 11-10 shows how to reduce the current consumption in the standby mode.

Figure 11-10: Example Method of Reducing Current Consumption in Standby Mode
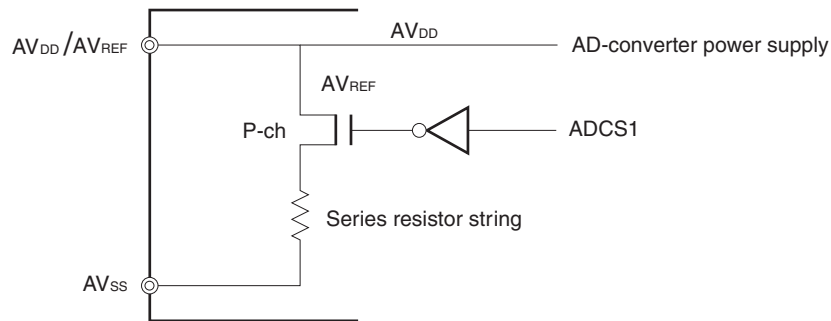
(2) Input range of ANI0 to ANI3

The input voltages of ANI0 to ANI3 should be within the specification range. In particular, if a voltage higher than AV_{DD}/AV_{REF} or lower than AV_{SS} is input (even if within the absolute maximum rating range), the conversion value of that channel will be undefined and the conversion values of other channels may also be affected.

(3) Contending operations

<1> Contention between A/D conversion result register (ADCR1) write and ADCR1 read by instruction upon the end of conversion

ADCR1 read is given priority. After the read operation, the new conversion result is written to ADCR1.

<2> Contention between ADCR1 write and A/D converter mode register (ADM1) write or analog input channel specification register (ADS1) write upon the end of conversion

ADM1 or ADS1 write is given priority. ADCR1 write is not performed, nor is the conversion end interrupt request signal (INTAD) generated.

(4) Noise countermeasures

To maintain 8-bit resolution, attention must be paid to noise input to pin AV_{DD}/AV_{REF} and pins ANI0 to ANI3. Because the effect increases in proportion to the output impedance of the analog input source, it is recommended that a capacitor be connected externally as shown in the Figure 11-11 to reduce noise.

Figure 11-11: Analog Input Pin Handling

If there is a possibility that noise equal to or higher than AV_{DD}/AV_{REF} or equal to or lower than AV_{SS} may enter, clamp with a diode with a small V_F value (0.3 V or lower).

Analog power supply and reference voltage input

AV_{DD}/AV_{REF}

ANI0 to ANI3

C = 100 to 1000 pF

AV_{SS}

V_{SS}
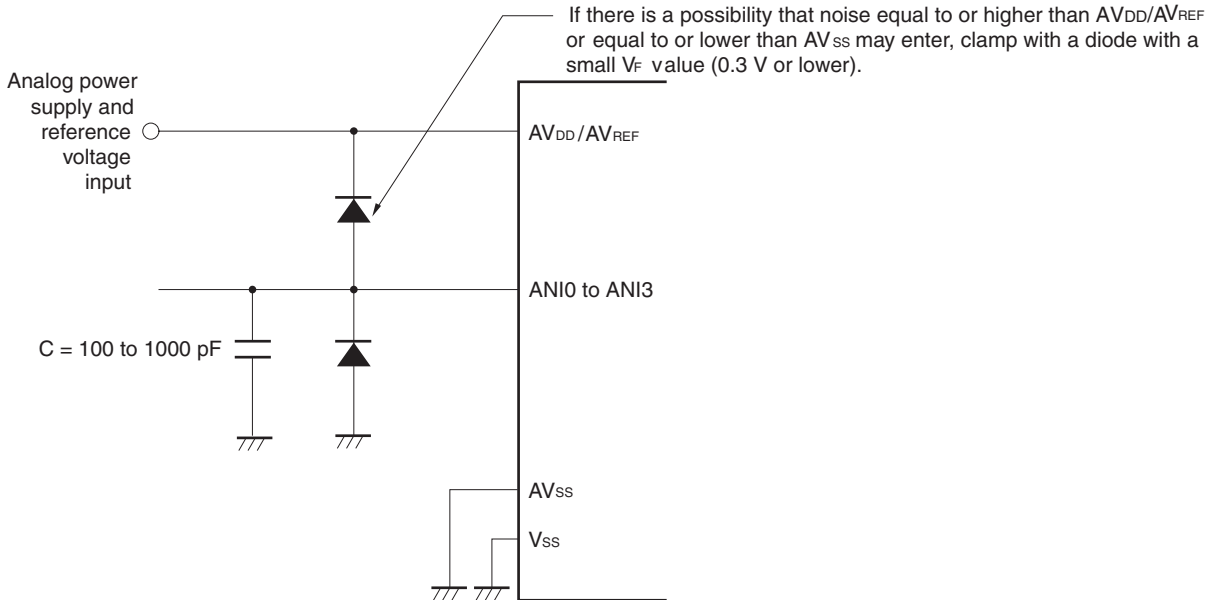
(5) ANI0 to ANI3

The analog input pins (ANI0 to ANI3) also function as input port pins (P10 to P13).

When A/D conversion is performed with any of pins ANI0 to ANI3 selected, do not execute a port input instruction while conversion is in progress, as this may reduce the conversion resolution. Also, if digital pulses are applied to a pin adjacent to the pin in the process of A/D conversion, the expected A/D conversion value may not be obtainable due to coupling noise. Therefore, avoid applying pulses to pins adjacent to the pin undergoing A/D conversion.

(6) AV_{DD}/AV_{REF} pin input impedance

A series resistor string of approximately 21 kΩ is connected between the AV_{DD}/AV_{REF} pin and the AV_{SS} pin.

Therefore, if the output impedance of the reference voltage is high, this will result in parallel connection to the series resistor string between the AV_{DD} pin and the AV_{SS} pin, and there will be a large reference voltage error.

(7) Interrupt request flag (ADIF)

The interrupt request flag (ADIF) is not cleared even if the analog input channel specification register (ADS1) is changed.

Caution is therefore required if a change of analog input pin is performed during A/D conversion. The A/D conversion result and conversion end interrupt request flag for the pre-change analog input may be set just before the ADS1 rewrite, if the ADIF is read immediately after the ADS1 rewrite, the ADIF may be set despite to the fact that the A/D conversion for the post-change analog input has not ended.

When the A/D conversion is stopped and then resumed, clear ADIF before the A/D conversion operation is resumed.

Figure 11-12: A/D Conversion End Interrupt Request Generation Timing
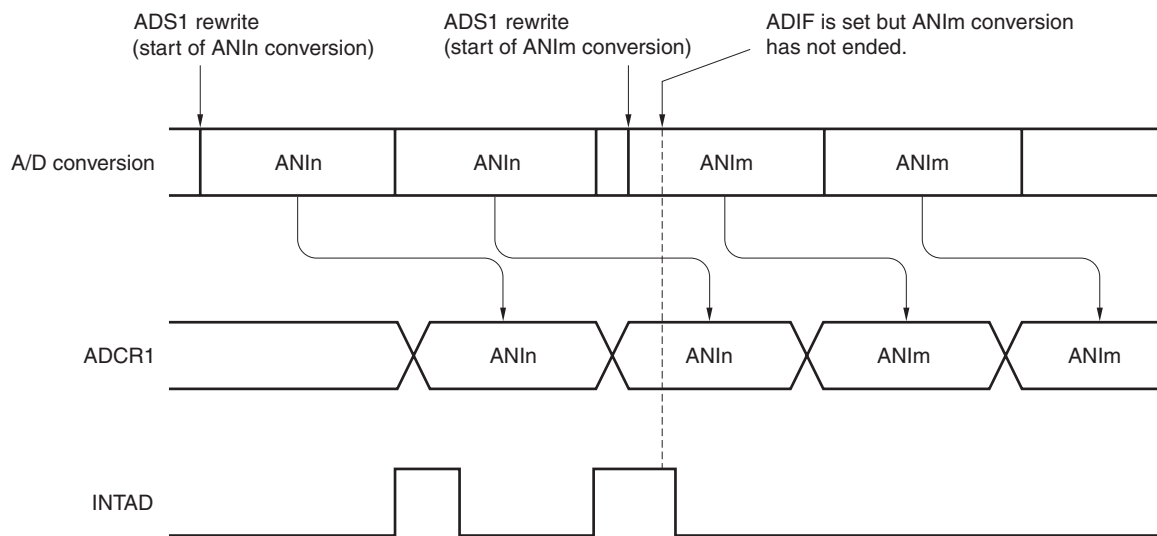
Remarks: **1.** n = 0, 1, ..., 7
2. m = 0, 1, ..., 7

(8) Read of A/D conversion result register (ADCR1)

When a write operation is executed to A/D converter mode register (ADM1) and analog input channel specification register (ADS1), the contents of ADCR1 are undefined. Read the conversion result before write operation is executed to ADM1, ADS1. If a timing other than the above is used, the correct conversion result may not be read.

11.6 Cautions on Emulation

To perform debugging with an in-circuit emulator, the D/A converter mode register (DAM0) must be set. DAM0 is a register used to set the I/O board.

11.6.1 D/A converter mode register (DAM0)

DAM0 is necessary if the power-fail detection function is used. Unless DAM0 is set, the power-fail detection function cannot be used. DAM0 is a write-only register.

Because the I/O board uses an external analog comparator and a D/A converter to implement part of the power-fail detection function, the reference voltage must be controlled. Therefore, set bit 0 (DACE) of DAM0 to 1 when using the power-fail detection function.

Figure 11-13: D/A Converter Mode Register (DAM0) Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
DAM0	0	0	0	0	0	0	0	DACE	FF84H	00H	W

DACE	Reference Voltage Control
0	Disabled
1	Enabled (when power-fail detection function is used)

Cautions: 1. DAM0 is a special register that must be set when debugging is performed with an in-circuit emulator. Even if this register is used, the operation of the μPD1615A Subseries is not affected. However, delete the instruction that manipulates this register from the program at the final stage of debugging.
2. Bits 7 to 1 must be set to 0.

[Memo]

Chapter 12 Serial Interface Outline

12.1 Serial Interface Outline

The μPD1615A subseries incorporates two channels of serial interfaces.

Table 12-1: Differences between the Serial Interface Channels

Serial Transfer Mode	μPD1615A(A)	μPD1615B(A)	μPD1615F(A)	μPD1616F(A)	μPD16F15A
SIO 3 (3-wire serial I/O)	○	○	○	○	○
UART	○	○	○	○	○

Remark: O : Provided
             — : Not provided

[Memo]

Chapter 13 Serial Interface SIO3

13.1 Serial Interface Channel 3 Functions

The SIO3 has the following two modes.

- Operation stop mode
- 3-wire serial I/O mode

(1) Operation stop mode
This mode is used if serial transfer is not performed. For details, see **15.5.1 Operation Stop Mode**.

(2) 3-wire serial I/O mode (fixed as MSB first)
This is an 8-bit data transfer mode using three lines: a serial clock line ($\overline{SCK3}$), serial output line (SO3), and serial input line (SI3).

Since simultaneous transmit and receive operations are enabled in 3-wire serial I/O mode, the processing time for data transfers is reduced.

The first bit in the 8-bit data in serial transfers is fixed as the MSB.

3-wire serial I/O mode is useful for connection to a peripheral I/O device that includes a clock-synchronous serial interface, like a display controller, etc. For details see **13.5.2 Three-Wire Serial I/O Mode**.

Figure 13-1 shows a block diagram of the SIO3.

Figure 13-1: Block Diagram of SIO3
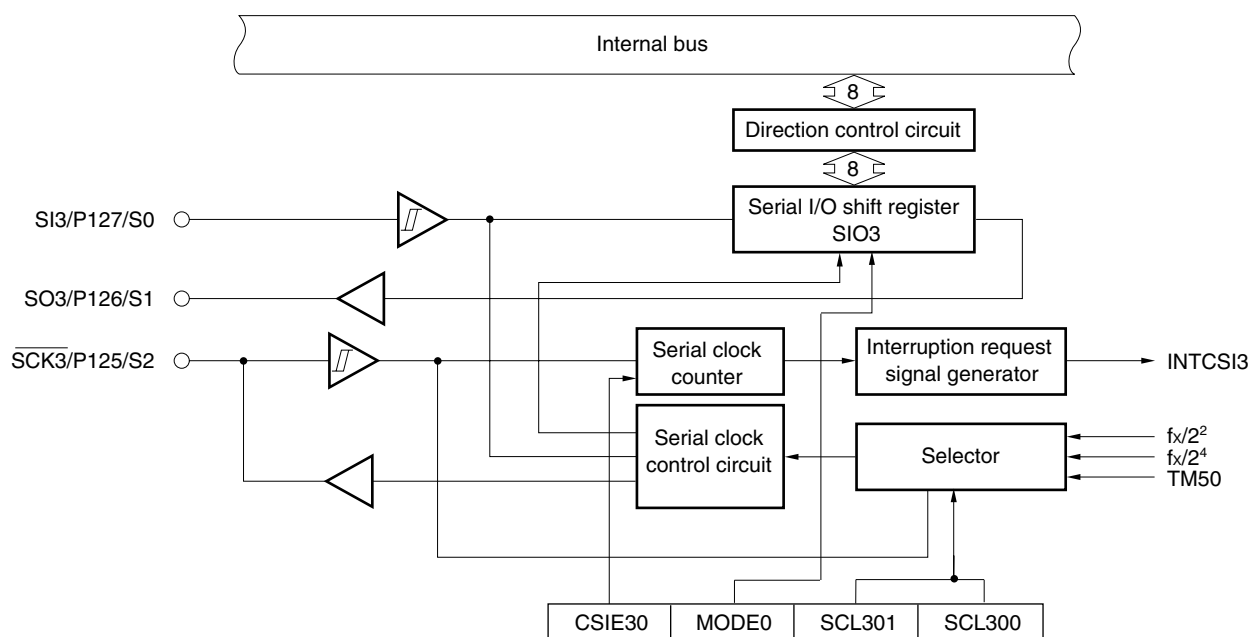
13.2 Serial Interface Channel 3 Configuration

The SIO3 includes the following hardware.

Table 13-1: Composition of SIO3

Item	Configuration
Registers	Serial I/O shift register 3 (SIO3)
Control registers	Serial operation mode register 3 (CSIM3)

(1) Serial I/O shift register 3 (SIO3)

This is an 8-bit register that performs parallel-serial conversion and serial transmit/receive (shift operations) synchronized with the serial clock.

SIO3 is set by an 8-bit memory manipulation instruction.

When "1" is set to bit 7 (CSIE30) of the serial operation mode register 3 (CSIM3), a serial operation can be started by writing data to or reading data from SIO3.

When transmitting, data written to SIO3 is output via the serial output (SO3).

When receiving, data is read from the serial input (SI3) and written to SIO3.

The RESET signal resets the register value to 00H.

Caution: **Do not access SIO3 during a transmit operation unless the access is triggered by a transfer start. (Read is disabled when MODE = 0 and write is disabled when MODE = 1.)**

13.3 List of SFRs (Special Function Registers)

Table 13-2: List of SFRs (Special Function Registers)

SFR name	Symbol	R/W	Units available for bit manipulation			Value when reset
			1 bit	8 bits	16 bits	
Serial operation mode register 3	CSIM3	R/W	○	○	—	00H
Serial I/O shift register 3	SIO3		—	○	—	

13.4   Serial Interface Control Registers

The SIO3 uses the following type of register for control functions.

• Serial operation mode register 3 (CSIM3)

(1) Serial operation mode register 3 (CSIM3)
This register is used to enable or disable SIO3's serial clock, operation modes, and specific operations.
CSIM3 can be set via a 1-bit or 8-bit memory manipulation instruction.
The $\overline{\text{RESET}}$ input sets the value to 00H.

Figure 13-2: Format of Serial Operation Mode Register 3 (CSIM3)

Address: FF6FH   When reset: 00H   R/W

Symbol	7	6	5	4	3	2	1	0
CSIM3	CSIE30	0	0	0	0	MODE0	SCL301	SCL300

CSIE30	Enable/disable specification for SIO3		
	Shift register operation	Serial counter	Port **Note 1**
0	Operation stop	Clear	Port function
1	Operation enable	Count operation enable	Serial operation + port function

MODE0	Transfer operation modes and flags		
	Operation mode	Transfer start trigger	P126/SO3/SA
0	Transmit/receive mode	Write to SIO3	SO3 output
1	Receive-only mode **Note 2**	Read from SIO3	Port function

SCL301	SCL300	Clock selection
0	0	External clock input
0	1	$f_x/2^2$
1	0	$f_x/2^4$
1	1	TM50 output

Notes: **1.** When CSIE30 = 0 (SIO3 operation stop status), the pins connected to SI3 and SO3 can be used for port functions.

2. When MODE0 = 1 (Receive mode), pin P126/SO3/S1 can be used for port function.

Caution: **If TM50 is used as clock generation for SIO3, no clock will be supplied to SIO3 unless TOE50 is set to 1. In this case a square wave output signal is output from the TO50 pin.**

13.5   Serial Interface Operations

This section explains on two modes of SIO3.

13.5.1 Operation stop mode
This mode is used if the serial transfers are not performed to reduce power consumption.
During the operation stop mode, the pins can be used as normal I/O ports as well.

(1) Register settings
The operation stop mode can be set via the serial operation mode register 3 (CSIM3).
CSIM3 can be set via 1-bit or 8-bit memory manipulation instructions.
The $\overline{\text{RESET}}$ input sets the value to 00H.

Figure 13-3: Format of Serial Operation Mode Register 3 (CSIM3)

Address: FF6FH When reset: 00H   R/W

Symbol	7	6	5	4	3	2	1	0
CSIM3	CSIE30	0	0	0	0	MODE0	SCL301	SCL300

CSIE30	SIO3 operation enable/disable specification		
	Shift register operation	Serial counter	Port **Note**
0	Operation stop	Clear	Port function
1	Operation enable	Count operation enable	Serial operation + port function

Note: When CSIE30 = 0 (SIO3 operation stop status), the pins connected to SI3 and SO3 can be used for port functions.

13.5.2 Three-wire serial I/O mode

The three-wire serial I/O mode is useful when connecting a peripheral I/O device that includes a clock-synchronous serial interface, a display controller, etc.

This mode executes the data transfer via three lines: a serial clock line ($\overline{\text{SCK3}}$), serial output line (SO3), and serial input line (SI3).

(1) Register settings

The 3-wire serial I/O mode is set via serial operation mode register 3 (CSIM3).

CSIM3 can be set via 1-bit or 8-bit memory manipulation instructions.

The $\overline{\text{RESET}}$ input set the value to 00H .

Figure 13-4: Format of Serial Operation Mode Register 3 (CSIM3)

Address: FF6FH   When reset: 00H   R/W

Symbol	7	6	5	4	3	2	1	0
CSIM30	CSIE30	0	0	0	0	MODE0	SCL301	SCL300

CSIE30	Enable/disable specification for SIO3		
	Shift register operation	Serial counter	Port Note 1
0	Operation stop	Clear	Port function
1	Operation enable	Count operation enable	Serial operation + port function

MODE0	Transfer operation modes and flags		
	Operation mode	Transfer start trigger	P126/SO3/S1
0	Transmit/receive mode	Write to SIO3	SO3 output
1	Receive-only mode Note 2	Read from SIO3	Port function

SCL301	SCL300	Clock selection (f_X = 8.00 MHz)
0	0	External clock input
0	1	$f_X/2^2$
1	0	$f_X/2^4$
1	1	TM50 output

Note: 1. When CSIE30 = 0 (SIO3 operation stop status), the pins connected to SI3 and SO3 can be used for port functions.
2. When M0DE0 = 1 (Receive mode), pin P126/SO3/S1 can be used for port function.

Caution: If TM50 is used as clock generation for SIO3, no clock will be supplied to SIO3 unless TOE50 is set to 1. In this case a square wave output signal is output from the TO50 pin.

(2) Communication Operations

In the three-wire serial I/O mode, data is transmitted and received in 8-bit units. Each bit of data is sent or received synchronized with the serial clock.

The serial I/O shift register 3 (SIO3) is shifted synchronized with the falling edge of the serial clock. The transmission data is held in the SO3 latch and is output from the SO3 pin. The data is received via the SI3 pin synchronized with the rising edge of the serial clock is latched to SIO3.

The completion of an 8-bit transfer automatically stops operation of SIO3 and sets a serial transfer completion flag.

Figure 13-5: Timing of Three-wire Serial I/O Mode

Transfer starts in synchronized with the serial clock's falling edge
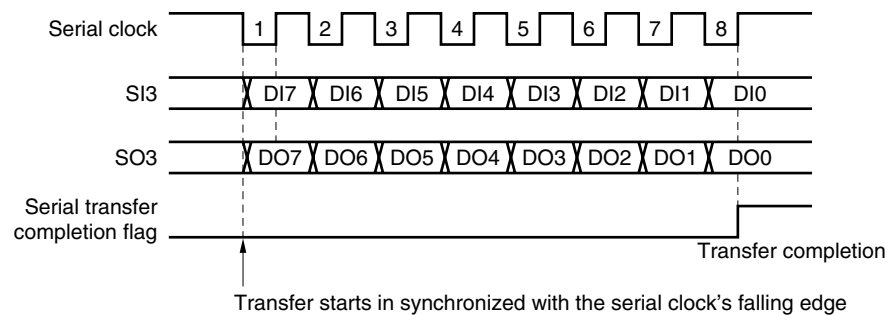
(3) Transfer start

A serial transfer starts when the following two conditions have been satisfied and transfer data has been set to serial I/O shift register 3 (SIO3).

- The SIO3 operation control bit (CSIE30) = 1
- After an 8-bit serial transfer, the internal serial clock is either stopped or is set to high level.
- Transmit/receive mode
  When CSIE30 = 1 and MODE0 = 0, transfer starts when writing to SIO3.
- Receive-only mode
  When CSIE30 = 1 and MODE0 = 1, transfer starts when reading from SIO3.

Cautions: 1. After the data has been written to SIO30, the transfer will not start even if the CSIE30 bit value is set to "1".1.

2. For a continuous data reception in the transmit/receive mode you schould restart the transfer trigger (write to SIO3) after the received data has been read out.

The completion of an 8-bit transfer automatically stops the serial transfer operation and sets a serial transfer completion flag.

[Memo]

Chapter 14 Serial Interface UART

14.1   Serial Interface UART Functions

The serial interface UART has the following two modes.

(1) Operation stop mode
This mode is used if the serial transfer is performed to reduce power consumption.
For details, see **14.5.1 Operation Stop Mode**.

(2) Asynchronous serial interface (UART) mode
This mode enables the full-duplex operation where one byte of data is transmitted and received after the start bit.
The on-chip dedicated UART baud rate generator enables communications using a wide range of selectable baud rates.
For details, see **14.5.2 Asynchronous Serial Interface (UART) Mode**.

Figure 14-1 shows a block diagram of the UART macro.

Figure 14-1: Block Diagram of UART
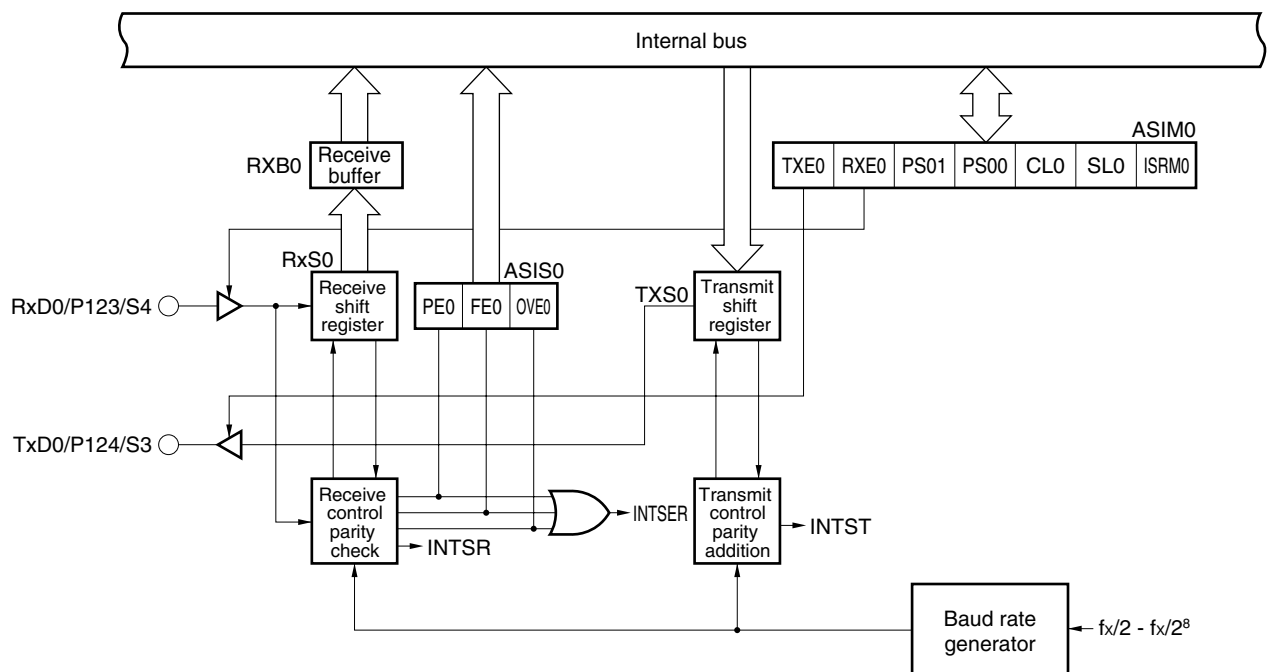
14.2 Serial Interface UART Configuration

The UART includes the following hardware.

Table 14-1: Configuration of UART

Item	Configuration
Registers	Transmit shift register 1 (TXS0) Receive shift register 1 (RXS0) Receive buffer register (RXB0)
Control registers	Asynchronous serial interface mode register (ASIM0) Asynchronous serial interface status register (ASIS0) Baud rate generator control register (BRGC0)

(1) Transmit shift register 1 (TXS0)

This register is for setting the transmit data. The data is written to TXS0 for transmission as serial data. When the data length is set as 7 bits, bits 0 to 6 of the data written to TXS0 are transmitted as serial data. Writing data to TXS0 starts the transmit operation.

TXS0 can be written via 8-bit memory manipulation instructions. It cannot be read.

When $\overline{\text{RESET}}$ is input, its value is FFH.

Caution: **Do not write to TXS0 during a transmit operation.**
The same address is assigned to TXS0 and the receive buffer register (RXB0). A read operation reads values from RXB0.

(2) Receive shift register 1 (RXS0)

This register converts serial data input via the RxD pin to parallel data. When one byte of the data is received at this register, the receive data is transferred to the receive buffer register (RXB0).

RXS0 cannot be manipulated directly by a program.

(3) Receive buffer register (RXB0)

This register is used to hold receive data. When one byte of data is received, one byte of new receive data is transferred from the receive shift register (RXS0).

When the data length is set as 7 bits, receive data is sent to bits 0 to 6 of RXB0. The MSB must be set to "0" in RXB0.

RXB0 can be read to via 8-bit memory manipulation instructions. It cannot be written to.

When $\overline{\text{RESET}}$ is input, its value is FFH.

Caution: **The same address is assigned to RXB0 and the transmit shift register (TXS0). During a write operation, values are written to TXS0.**

(4) Transmission control circuit

The transmission control circuit controls transmit operations, such as adding a start bit, parity bit, and stop bit to data that is written to the transmit shift register (TXS0), based on the values set to the asynchronous serial interface mode register (ASIM0).

(5) Reception control circuit

The reception control circuit controls the receive operations based on the values set to the asynchronous serial interface mode register (ASIM0). During a receive operation, it performs error checking, such as parity errors, and sets various values to the asynchronous serial interface status register (ASIS0) according to the type of error that is detected.

14.3 List of SFRS (Special Function Registers)

Table 14-2: List of SFRs (Special Function Registers)

SFR name	Symbol	R/W	Units available for bit manipulation			Value when reset
			1 bit	8 bits	16 bits	
Transmit shift register	TXS0	W	-	O	-	FFH
Receive buffer register	RXB0	R				
Asynchronous serial interface mode register	ASIM0	R/W	O	O	-	00H
Asynchronous serial interface status register	ASIS0	W	-	O	-	
Baud rate generator control register	BRGC0	R/W	-	O	-	

14.4 Serial Interface Control Registers

The UART uses the following three types of registers for control functions.

- Asynchronous serial interface mode register (ASIM0)
- Asynchronous serial interface status register (ASIS0)
- Baud rate generator control register (BRGC0)

(1) Asynchronous serial interface mode register (ASIM0)
This is an 8-bit register that controls the UART serial transfer operation.
ASIM0 can be set by 1-bit or 8-bit memory manipulation instructions.
RESET input sets the value to 00H.
Figure 14-2 shows the format of ASIM0.

Figure 14-2: Format of Asynchronous Serial Interface Mode Register (ASIM0)

Address: FFA0H When reset: 00H   R/W

Symbol	7	6	5	4	3	2	1	0
ASIM0	TXE0	RXE0	PS01	PS00	CL0	SL0	ISRM0	0

TXE0	RXE0	Operation mode	RxD0/P123/S4pin function	TxD0/P124/S3pin function
0	0	Operation stop	Port function	Port function
0	1	UART0 mode (receive only)	Serial operation	Port function
1	0	UART0 mode (transmit only)	Port function	Serial operation
1	1	UART0 mode (transmit and receive)	Serial operation	Serial operation

PS01	PS00	Parity bit specification
0	0	No parity
0	1	Zero parity always added during transmittion No parity detection during reception (parity errors do not occur)
1	0	Odd aprity
1	1	Even parity

CL0	Character length specification
0	7 bits
1	8 bits

SL0	Stop bit length specification for transmit data
0	1 bit
1	2 bits

ISRM0	Receive completion interrupt control when error occurs
0	Receive completion interrupt is issued when an error occurs
1	Receive completion interrupt is not issued when an error occurs

Caution: Do not switch the operation mode until after the current serial transmit/receive operation has stopped.

(2) Asynchronous serial interface status register (ASIS0)

When a receive error occurs during UART mode, this register indicates the type of error.

ASIS0 can be read using an 8-bit memory manipulation instruction.

When $\overline{\text{RESET}}$ is input, its value is 00H.

Figure 14-3: Format of Asynchronous Serial Interface Status Register (ASIS0)

Address: FFA1H When reset: 00H   R

Symbol	7	6	5	4	3	2	1	0
ASIS0	0	0	0	0	0	PE0	FE0	OVE0

PE0	Parity error flag
0	No parity error
0	Parity error (Incorrect parity bit detected)

FE0	Framing error flag
0	No framing error
1	Framing error[Note 1] (Stop bit not detected)

OVE0	Overrun error flag
0	No overrun error
1	Overrun error[Note 2] (Next receive operation was completed before data was read from receive buffer register)

Notes: 1. Even if a stop bit length of two bits has been set to bit 2 (SL0) in the asynchronous serial interface mode register (ASIM0), the stop bit detection during a receive operation only applies to a stop bit length of 1 bit.

2. Be sure to read the contents of the receive buffer register (RXB0) when an overrun error has occurred.

Until the contents of RXB0 are read, further overrun errors will occur when receiving data.

(3) Baud rate generator control register (BRGC0)

This register sets the serial clock for UART.

BRGC can be set via an 8-bit memory manipulation instruction.

When $\overline{\text{RESET}}$ is input, its value is 00H.

Figure 14-4 shows the format of BRGC0.

Figure 14-4: Format of Baud Rate Generator Control Register (BRGC0)

Address: FFA2H   When reset: 00H   R/W

Symbol	7	6	5	4	3	2	1	0
BRGC0	0	TPS02	TPS01	TPS00	MDL03	MDL02	MDL01	MDL00

(f_x = 8.00 MHz)

TPS02	TPS01	TPS00	Source clock selection for 5-bit counter	n
0	0	0	$f_x/2^1$	1
0	0	1	$f_x/2^2$	2
0	1	0	$f_x/2^3$	3
0	1	1	$f_x/2^4$	4
1	0	0	$f_x/2^5$	5
1	0	1	$f_x/2^6$	6
1	1	0	$f_x/2^7$	7
1	1	1	$f_x/2^8$	8

MDL03	MDL02	MDL01	MDL00	Input clock selection for baud rate generator	k
0	0	0	0	$f_{SCK}/16$	0
0	0	0	1	$f_{SCK}/17$	1
0	0	1	0	$f_{SCK}/18$	2
0	0	1	1	$f_{SCK}/19$	3
0	1	0	0	$f_{SCK}/20$	4
0	1	0	1	$f_{SCK}/21$	5
0	1	1	0	$f_{SCK}/22$	6
0	1	1	1	$f_{SCK}/23$	7
1	0	0	0	$f_{SCK}/24$	8
1	0	0	1	$f_{SCK}/25$	9
1	0	1	0	$f_{SCK}/26$	10
1	0	1	1	$f_{SCK}/27$	11
1	1	0	0	$f_{SCK}/28$	12
1	1	0	1	$f_{SCK}/29$	13
1	1	1	0	$f_{SCK}/30$	14
1	1	1	1	Setting prohibit	—

Caution: **Writing to BRGC0 during a communication operation may cause abnormal output from the baud rate generator and disable further communication operations. Therefore, do not write to BRGC0 during a communication operation.**

Remarks: **1.** f_{SCK}: Source clock for 5-bit counter
2. n: Value set via TPS00 to TPS02 ($1 \le n \le 8$)
3. k: Value set via MDL00 to MDL03 ($0 \le k \le 14$)

14.5   Serial Interface Operations

This section explains the three modes of the UART.

14.5.1 Operation stop mode
This mode is used when serial transfers are not performed to reduce power consumption.
In the operation stop mode, pins can be used as ordinary ports.

(1) Register settings
Operation stop mode settings are made via the asynchronous serial interface mode register (ASIM0).
ASIM0 can be set via 1-bit or 8-bit memory manipulation instructions.
When $\overline{\text{RESET}}$ is input, its value is 00H.

Figure 14-5: Register Settings

Address: FFA0H   When reset: 00H   R/W

Symbol	7	6	5	4	3	2	1	0
ASIM0	TXE0	RXE0	PS01	PS00	CL0	SL0	ISRM0	0

TXE0	RXE0	Operation mode	RxD0/P123/S4pin function	TxD0/P124/S3pin function
0	0	Operation stop	Port function	Port function
0	1	UART0 mode (receive only)	Serial operation	Port function
1	0	UART0 mode (transmit only)	Port function	Serial operation
1	1	UART0 mode (transmit and receive)	Serial operation	Serial operation

Caution: Do not switch the operation mode until after the current serial transmit/receive operation has stopped.

14.5.2 Asynchronous serial interface (UART) mode
This mode enables full-duplex operation where one byte of the data is transmitted or received after the start bit.
The on-chip dedicated UART baud rate generator enables communications by using a wide range of selectable baud rates.

(1) Register settings
The UART mode settings are made via the asynchronous serial interface mode register (ASIM0), asynchronous serial interface status register (ASIS0), and the baud rate generator control register (BRGC0).

(a) Asynchronous serial interface mode register (ASIM0)

ASIM0 can be set by 1-bit or 8-bit memory manipulation instructions.

When $\overline{\text{RESET}}$ is input, its value is 00H.

Figure 14-6: Asynchronous serial interface mode register (ASIM0)

Address: FFA0H When reset: 00H   R/W

Symbol	7	6	5	4	3	2	1	0
ASIM0	TXE0	RXE0	PS01	PS00	CL0	SL0	ISRM0	0

TXE0	PEX0	Operation mode	RxD0/P123/S4pin function	TxD0/P124/S3pin function
0	0	Operation stop	Port function	Port function
0	1	UART0 mode (receive only)	Serial operation	Port function
1	0	UART0 mode (transmit only)	Port function	Serial operation
1	1	UART0 mode (transmit and receive)	Serial operation	Serial operation

PS01	PS00	Parity bit specification
0	0	No parity
0	1	Zero parity always added during transmittion No parity detection during reception (parity errors do not occur)
1	0	Odd aprity
1	1	Even parity

CL0	Character length specification
0	7 bits
0	8 bits

SL0	Stop bit length specification for transmit data
0	1 bit
1	2 bits

ISRM0	Receive completion interrupt control when error occurs
0	Receive completion interrupt is issued when an error occurs
1	Receive completion interrupt is not issued when an error occurs

Caution: **Do not switch the operation mode until after the current serial transmit/receive operation has stopped.**

(b) Asynchronous serial interface status register (ASIS0)

ASIS0 can be read using an 8-bit memory manipulation instruction.

When $\overline{\text{RESET}}$ is input, its value is 00H.

Figure 14-7: Asynchronous serial interface status register (ASIS0)

Address: FFA1H When reset: 00H   R

Symbol	7	6	5	4	3	2	1	0
ASIS0	0	0	0	0	0	PE0	FE0	OVE0

PE0	Parity error flag
0	No parity error
1	Parity error (Incorrect parity bit detected)

FE0	Framing error flag
0	No framing error
1	Framing error **Note 1** (Stop bit not detected)

OVE0	Overrun error flag
0	No overrun error
1	Overrun error **Note 2** (Next receive operation was completed before data was read from receive buffer register)

Notes: 1. Even if a stop bit length of two bits has been set to bit 2 (SL0) in the asynchronous serial interface mode register (ASIM0), stop bit detection during a receive operation only applies to a stop bit length of 1 bit.

2. Be sure to read the contents of the receive buffer register (RXB0) when an overrun error has occurred.

Until the contents of RXB0 are read, further overrun errors will occur when receiving data.

(c) Baud rate generator control register (BRGC0)

BRGC0 can be set by an 8-bit memory manipulation instruction.

When $\overline{\text{RESET}}$ is input, its value is 00H.

Figure 14-8: Baud rate generator control register (BRGC0)

Address: FFA2H   When reset: 00H   R/W

Symbol	7	6	5	4	3	2	1	0
BRGC0	0	TPS02	TPS01	TPS00	MDL03	MDL02	MDL01	MDL00

(fx = 8.00 MHz)

TPS02	TPS01	TPS00	Source clock selection for 5-bit counter	n
0	0	0	$f_x/2^1$	1
0	0	1	$f_x/2^2$	2
0	1	0	$f_x/2^3$	3
0	1	1	$f_x/2^4$	4
1	0	0	$f_x/2^5$	5
1	0	1	$f_x/2^6$	6
1	1	0	$f_x/2^7$	7
1	1	1	$f_x/2^8$	8

MDL03	MDL02	MDL01	MDL00	Input clock selection for baud rate generator	k
0	0	0	0	$f_{SCK}/16$	0
0	0	0	1	$f_{SCK}/17$	1
0	0	1	0	$f_{SCK}/18$	2
0	0	1	1	$f_{SCK}/19$	3
0	1	0	0	$f_{SCK}/20$	4
0	1	0	1	$f_{SCK}/21$	5
0	1	1	0	$f_{SCK}/22$	6
0	1	1	1	$f_{SCK}/23$	7
1	0	0	0	$f_{SCK}/24$	8
1	0	0	1	$f_{SCK}/25$	9
1	0	1	0	$f_{SCK}/26$	10
1	0	1	1	$f_{SCK}/27$	11
1	1	0	0	$f_{SCK}/28$	12
1	1	0	1	$f_{SCK}/29$	13
1	1	1	0	$f_{SCK}/30$	14
1	1	1	1	Setting prohibit	—

Caution: **Writing to BRGC0 during a communication operation may cause abnormal output from the baud rate generator and disable further communication operations. Therefore, do not write to BRGC0 during a communication operation.**

Remarks: **1.** f_{SCK}: Source clock for 5-bit counter

2. n: Value set via TPS00 to TPS02 ($1 \leq n \leq 8$)

3. k: Value set via MDL00 to MDL03 ($0 \leq k \leq 14$)

The transmit/receive clock that is used to generate the baud rate is obtained by dividing the main system clock.

- Use of main system clock to generate a transmit/receive clock for baud rate
  The main system clock is divided to generate the transmit/receive clock. The baud rate generated by the main system clock is determined according to the following formula.

$$[\text{Baud rate}] = \frac{fx}{2^{n+1}(k + 16)} \quad [\text{bps}]$$

   fx : Oscillation frequency of main system clock (in Hz)

   n : Value set via TPS00 to TPS02 ($1 \leq n \leq 8$)
      For details, see Table 17-3.

   k : Value set via MDL00 to MDL02 ($0 \leq k \leq 14$)

Table 17-3 shows the relation between the 5-bit counter's source clock assigned to bits 4 to 6 (TPS00 to TPS02) of BRGC0 and the "n" value in the above formula.

Table 14-3: Relation between 5-bit Counter's Source Clock and "n" Value

TPS02	TPS01	TPS00	5-bit counter's source clock selected	n
0	0	0	$fx/2^1$	1
0	0	1	$fx/2^2$	2
0	1	0	$fx/2^3$	3
0	1	1	$fx/2^4$	4
1	0	0	$fx/2^5$	5
1	0	1	$fx/2^6$	6
1	1	0	$fx/2^7$	7
1	1	1	$fx/2^8$	8

Remark: fx: Oscillation frequency of main system clock.

- **Error tolerance range for baud rates**

  The tolerance range for baud rates depends on the number of bits per frame and the counter's division rate $[1/(16 + k)]$.

  Table 14-4 describes the relation between the main system clock and the baud rate and Figure 14-9 shows an example of a baud rate error tolerance range.

Table 14-4: Relation between Main System Clock and Baud Rate

Baud rate (bps)	fx = 8.00 MHz		fx = 4.00 MHz	
	BRGC0	ERR(%)	BRGC0	ERR(%)
600	7AH	0.16	6AH	0.16
1200	6AH	0.16	5AH	0.16
2400	5AH	0.16	4AH	0.16
4800	4AH	0.16	3AH	0.16
9600	3AH	0.16	2AH	0.16
19200	2AH	0.00	1AH	0.00
38400	1AH	0.16	0AH	0.16
76800	0AH	0.16	-	-
115200	02H	0.16	-	-

Remarks: **1.** fx: Oscillation frequency of main system clock
2. n: Value set via TPS00 to TPS02 ($1 \leq n \leq 8$)
3. k: Value set via MDL00 to MDL03 ($0 \leq k \leq 14$)

Figure 14-9: Error Tolerance (when k = 0), including Sampling Errors
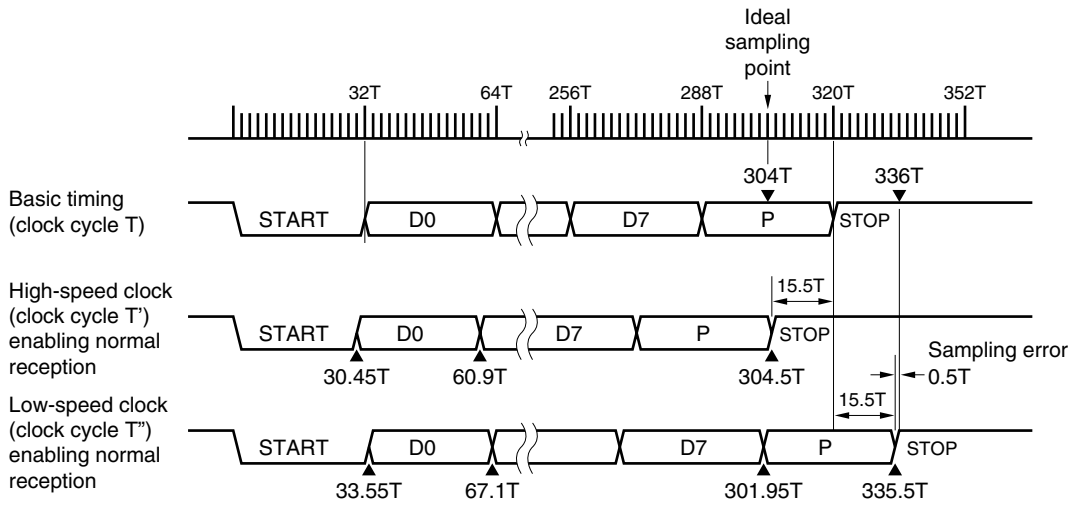
Remark:　T: 5-bit counter's source clock cycle

Baud rate error tolerance (when k = 0) $= \dfrac{\pm 15.5}{320} \times 100 = 4.8438\ (\%)$

(2) Communication operations

(a) Data format

As shown in Figure 14-10, the format of the transmit/receive data consists of a start bit, character bits, a parity bit, and one or more stop bits.

The asynchronous serial interface mode register (ASIM0) is used to set the character bit length, parity selection, and stop bit length within each data frame.

Figure 14-10: Format of Transmit/Receive Data in Asynchronous Serial Interface

	Start bit	D0	D1	D2	D3	D4	D5	D6	D7	Parity bit	Stop bit	

1 data frame
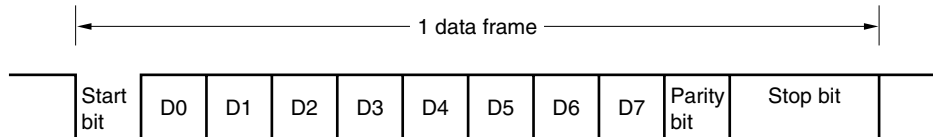
- Start bit .... 1 bit
- Character bits ... 7 bits or 8 bits
- Parity bit .... Even parity, odd parity, zero parity, or no parity
- Stop bit(s) .... 1 bit or 2 bits

When "7 bits" is selected as the number of character bits, only the low-order 7 bits (bits 0 to 6) are valid, so that during a transmission the highest bit (bit 7) is ignored and during reception the highest bit (bit 7) must be set to "0".

The asynchronous serial interface mode register (ASIM0) and the baud rate generator control register (BRGC0) are used to set the serial transfer rate.

If a receive error occurs, information about the receive error can be recognized by reading the asynchronous serial interface status register (ASIS0).

(b) Parity types and operations

The parity bit is used to detect bit errors in transfer data. Usually, the same type of parity bit is used by the transmitting and receiving sides. When odd parity or even parity is set, errors in the parity bit (the odd-number bit) can be detected. When zero parity or no parity is set, errors are not detected.

(1) Even parity

- During transmission
  The number of bits in transmit data that includes a parity bit is controlled so that there are an even number of "1" bits. The value of the parity bit is as follows.

  If the transmit data contains an odd number of "1" bits : the parity bit value is "1"
  If the transmit data contains an even number of "1" bits: the parity bit value is "0"

- During reception
  The number of "1" bits is counted among the transfer data that include a parity bit, and a parity error occurs when the result is an odd number.

(2) Odd parity

- During transmission
  The number of bits in transmit data that includes a parity bit is controlled so that there is an odd number of "1" bits. The value of the parity bit is as follows.

  If the transmit data contains an odd number of "1" bits : the parity bit value is "0"
  If the transmit data contains an even number of "1" bits: the parity bit value is "1"

- During reception
  The number of "1" bits is counted among the transfer data that include a parity bit, and a parity error occurs when the result is an even number.

(3) Zero parity
  During transmission, the parity bit is set to "0" regardless of the transmit data.
  During reception, the parity bit is not checked. Therefore, no parity errors will occur regardless of whether the parity bit is a "0" or a "1".

(4) No parity
  No parity bit is added to the transmit data.
  During reception, receive data is regarded as having no parity bit. Since there is no parity bit, no parity errors will occur.

(c) Transmission

The transmit operation is started when transmit data is written to the transmit shift register (TXS0). A start bit, parity bit, and stop bit(s) are automatically added to the data.

Starting the transmit operation shifts out the data in TXS0, thereby emptying TXS0, after which a transmit completion interrupt (INTST) is issued.

The timing of the transmit completion interrupt is shown in Figure 14-11.

Figure 14-11: Timing of Asynchronous Serial Interface Transmit Completion Interrupt

(a) Stop bit length: 1 bit

(b) Stop bit length: 2 bits
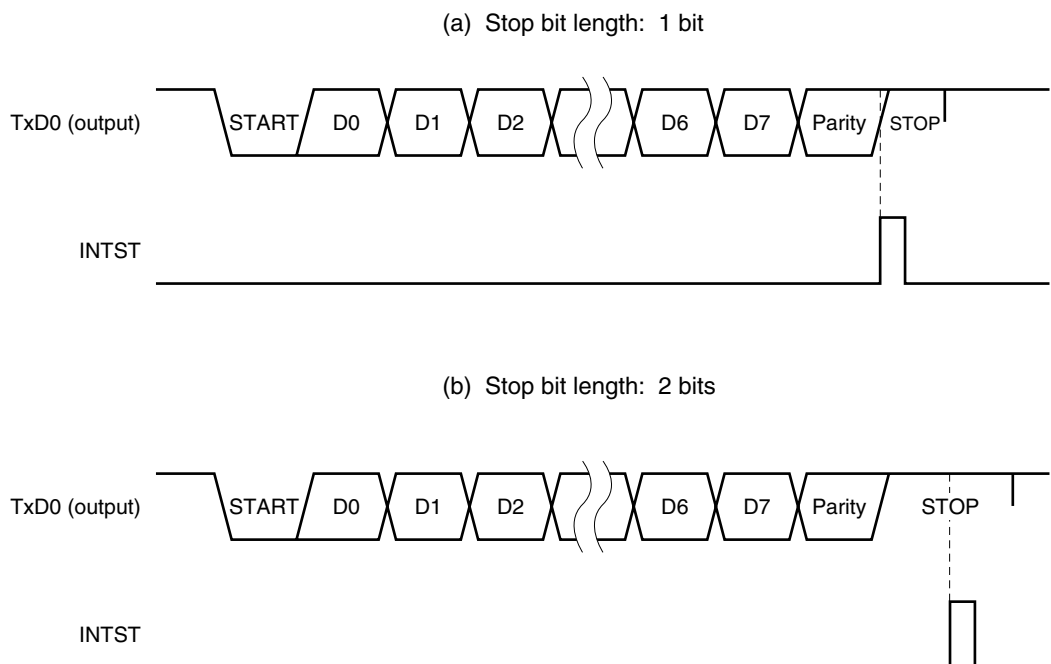
Caution: **Do not write to the asynchronous serial interface mode register (ASIM0) during a transmit operation. Writing to ASIM0 during a transmit operation may disable further transmit operations (in such cases, enter a RESET to restore normal operation). Whether or not a transmit operation is in progress can be determined via software using the transmit completion interrupt (INTST) or the interrupt request flag (STIF) that is set by INTST.**

(d) Reception

The receive operation is enabled when "1" is set to bit 6 (RXE0) of the asynchronous serial interface mode register (ASIM0), and input data via RxD pin is sampled.

The serial clock specified by ASIM0 is used when sampling the RxD0 pin.

When the RxD0 pin goes low, the 5-bit counter begins counting and the start timing signal for data sampling is output if half of the specified baud rate time has elapsed. If the sampling of the RxD0 pin input of this start timing signal yields a low-level result, a start bit is recognized, after which the 5-bit counter is initialized and starts counting and data sampling begins. After the start bit is recognized, the character data, parity bit, and one-bit stop bit are detected, at which point reception of one data frame is completed.

Once the reception of one data frame is completed, the receive data in the shift register is transferred to the receive buffer register (RXB0) and a receive completion interrupt (INTSR) occurs.

Even if an error has occurred, the receive data in which the error occurred is still transferred to RXB0 and INTSR occurs (see Figure 14-9).

If the RXE0 bit is reset (to "0") during a receive operation, the receive operation is stopped immediately.

At this time, neither the contents of RXB0 and ASIS0 do not change, nor does INTSR or INTSER occur.

Figure 14-12 shows the timing of the asynchronous serial interface receive completion interrupt.

Figure 14-12: Timing of Asynchronous Serial Interface Receive Completion Interrupt
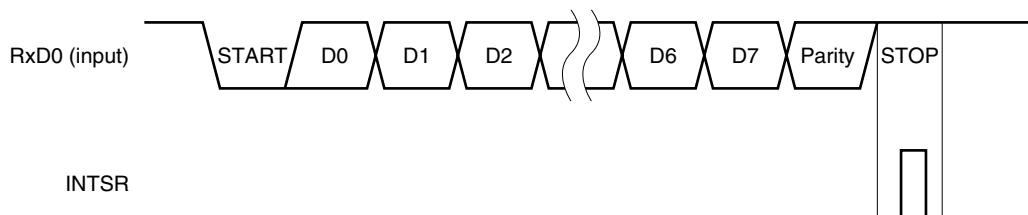
Caution: Be sure to read the contents of the receive buffer register (RXB0) even when a receive error has occurred. Overrun errors will occur during the next data receive operations and the receive error status will remain until the contents of RXB0 are read.

(e) Receive errors

Three types of errors can occur during a receive operation: parity error, framing error, or overrun error. If, as the result of the data reception, an error flag is set to the asynchronous serial interface status register (ASIS0), a receive error interrupt (INTSER) will occur. Receive error interrupts are generated before receive interrupts (INTSR). Table 17-5 lists the causes behind receive errors.

As part of receive error interrupt (INTSER) servicing, the contents of ASIS0 can be read to determine which type of error occurred during the receive operation (see Table 14-5 and Figure 14-13).

The content of ASIS0 is reset (to "0") if the receive buffer register (RXB0) is read or when the next data is received (if the next data contains an error, another error flag will be set).

Table 14-5: Causes of Receive Errors

Receive error	Cause	ASIS0 value
Parity error	Parity specified during transmission does not match parity of receive data	04H
Framing error	Stop bit was not detected	02H
Overrun error	Reception of the next data was completed before data was read from the receive buffer register	01H

Figure 14-13: Receive Error Timing
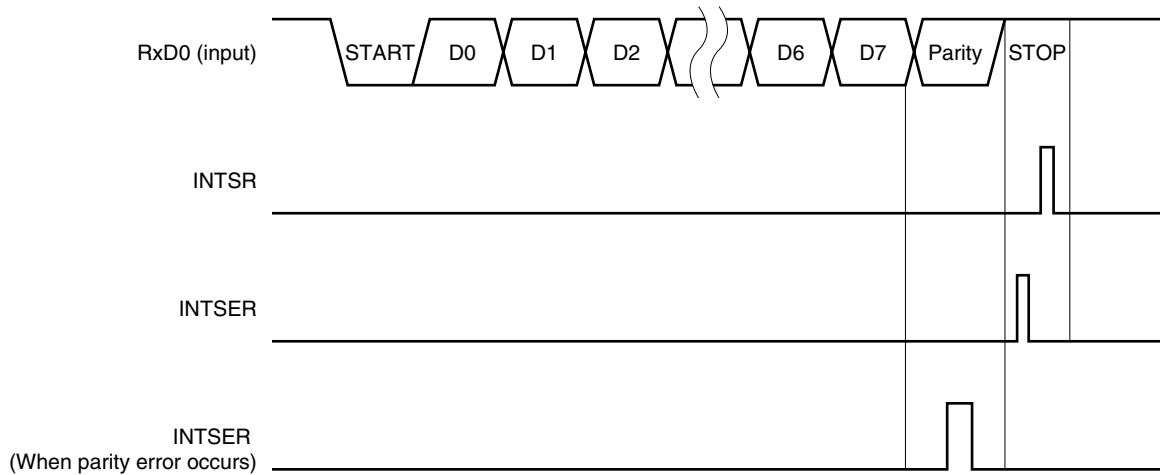
Cautions: 1. The contents of ASIS0 are reset (to "0") when the receive buffer register (RXB0) is read or when the next data is received. To obtain information about the error, be sure to read the contents of ASIS0 before reading RXB0.

2. Be sure to read the contents of the receive buffer register (RXB0) even when a receive error has occurred. Overrun errors will occur during the next data receive operations and the receive error status will remain until the contents of RXB0 are read.

14.6   Standby Function

Serial transfer operations can be performed during HALT mode.

During STOP mode, serial transfer operations are stopped and the values in the asynchronous serial interface mode register (ASIM0), transmit shift register (TXS0), receive shift register (RxS0), and receive buffer register (RXB0) remain as they were just before the clock was stopped.

Output from the TxD0 pin retains the immediately previous data if the clock is stopped (if the system enters STOP mode) during a transmit operation. If the clock is stopped during a receive operation, the data received before the clock was stopped is retained and all subsequent operations are stopped. The receive operation can be restarted once the clock is restarted.

[Memo]

Chapter 15 VAN Controller

15.1 Features

- The VAN UART is compatible with the ISO 11519 VAN standard, Part 3, revision 4.00.

- The VAN UART executes all the VAN frame types:
* Programmed in autonomous mode (RANK bit = 0), it performs the transmission and reception of data frames (transmits from the SOF field or from the IDEN field) and read frames as well as the in frame response.
* Programmed in synchronous mode (RANK bit = 1), it performs the transmission (transmits from the IDEN field only) and reception of data and read frames as well as the in frame response.

- The transmission and reception of these frames can be done up to 500 kTS/s for an 8 MHz quartz clock.

- The VAN frame is encoded in Enhanced Manchester.

- In autonomous mode the choice of the bus speed is programmable via a 4 bit prescaler (DIAG_CTRL_REG register). A bit of this prescaler performing a division by 1,2,3 or 5 permits the use of "non binary" quartz clocks having a frequency of 3, 5 or 6 MHz.

- The VAN UART carries out the collision detection and goes into receive mode if lost arbitration before the end of the current Time Slot (TS). The circuit generates an interrupt if required by the user. The collision is not considered as an error.

- The VAN UART re-synchronises the transmission and reception clocks at each edge detected on the bus line.

- The VAN UART incorporates a cell calculating the CRC in transmission and in reception.

- The VAN UART integrates the line diagnosis function, which consists of:
* The digital filtering of the outputs of the three comparators RXD0, RXD1 and RXD2.
* Asynchronous diagnosis.
* Synchronous diagnosis.
* Transmission diagnosis (with enable bit).
* Protocol error (8 consecutive dominant TS).
* Possibility to force one of the three comparators.

- The VAN UART signals the errors that occurred on the VAN bus and generate an interrupt connected to each error if required by the user.
3 bits implanted in the status register STAT_REG differentiate the errors in transmission or in reception.

15.2   Overview of the VAN Bus

15.2.1 VAN UART Description
The VAN UART cell integrated in this microcontroller is comform to the VAN Standard (ISO 11519, Part 3, Rev 4.00).

15.2.2 VAN UART Interface

Figure 15-1: VAN UART Interface
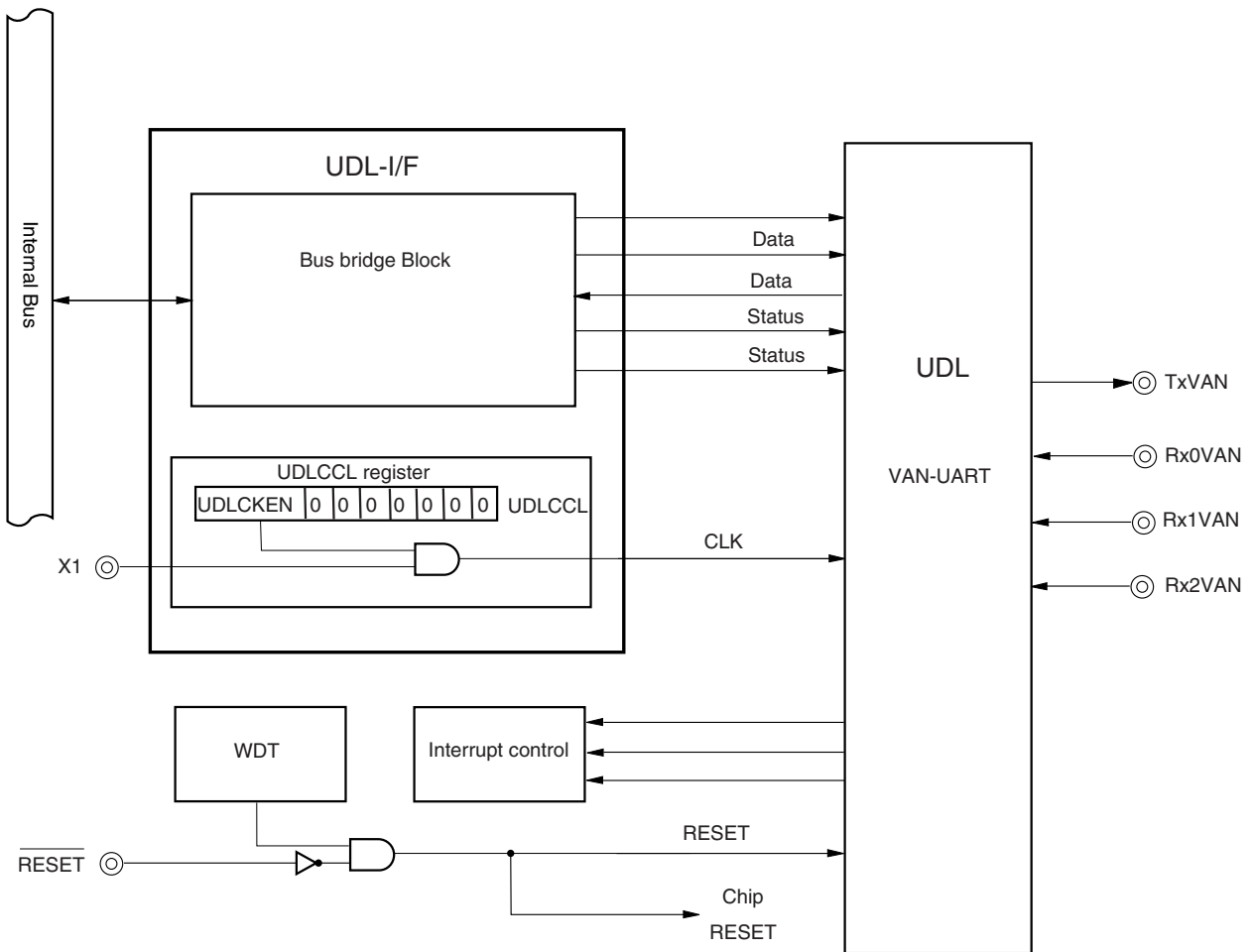
The VAN UART is realised with one transmit register and one receive register. The application software may check the status registers in order to get information of the bus state and the received or transmitted messages. The device has the capability to generate an interrupt as soon as one byte is transmitted or received. Care has to be taken when transmitting or receiving in order not to miss the TBE (INT1) or RDA (INT2) interrupts occuring on every byte (TBE means transmit buffer empty and RDA means received data available). At each of these interrupts, the application software has to perform a data exchange between the application and the TX/RX register.

Figure 15-2: VAN UART Block Diagram
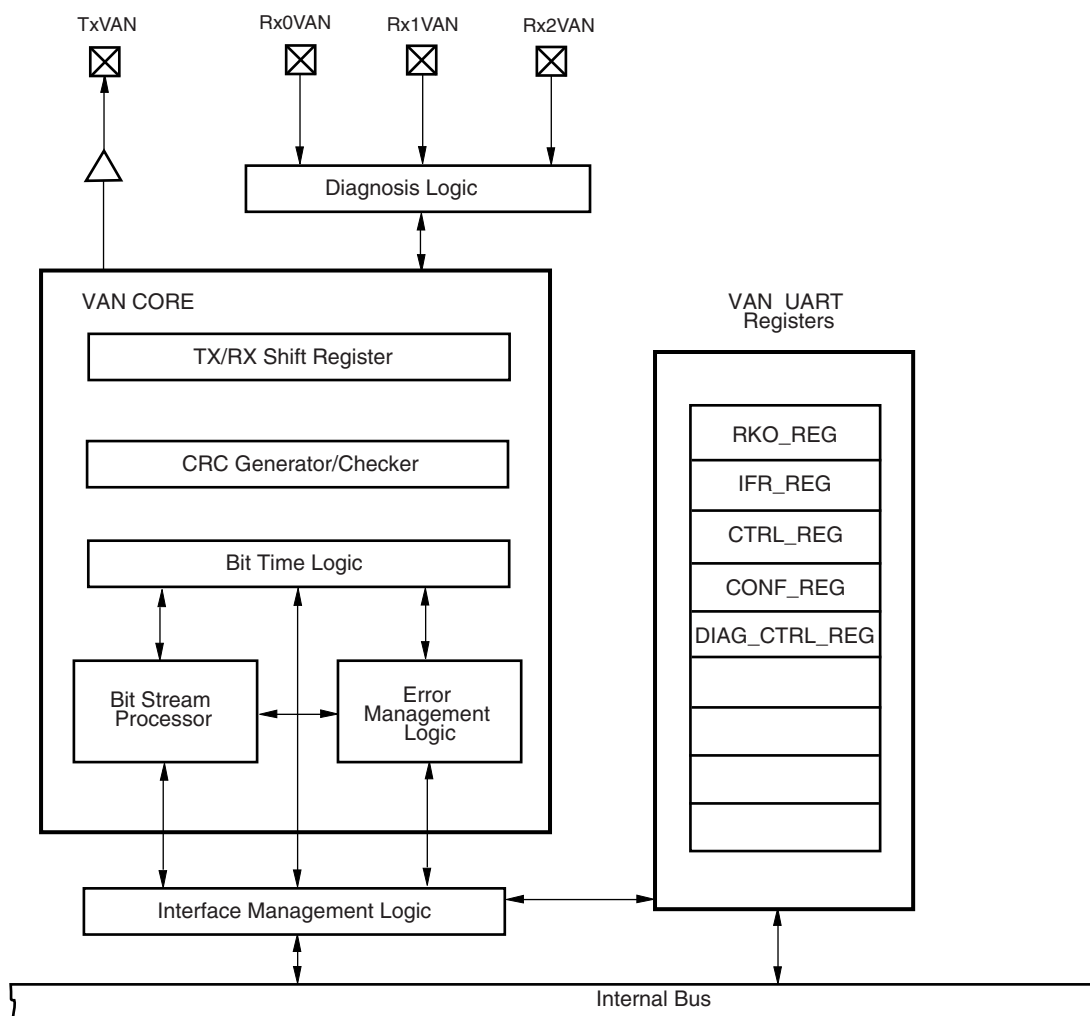
Interface Management Logic (IML) :

The IML executes the CPU's transmission and reception commands and controls the data transfer between CPU, Rx/Tx and VAN registers. It provides the VAN UART interface with Rx/Tx data from the memory mapped Register Block. It sets and resets the VAN status informations and generates interrupts to the CPU. It also generates the bit clock according the divider chosen by application software.

This divider divides the input clock by the value defined in the VAN Prescaler. The following picture shows the generation of the VAN clock :

Figure 15-3: Generation of the VAN Clock
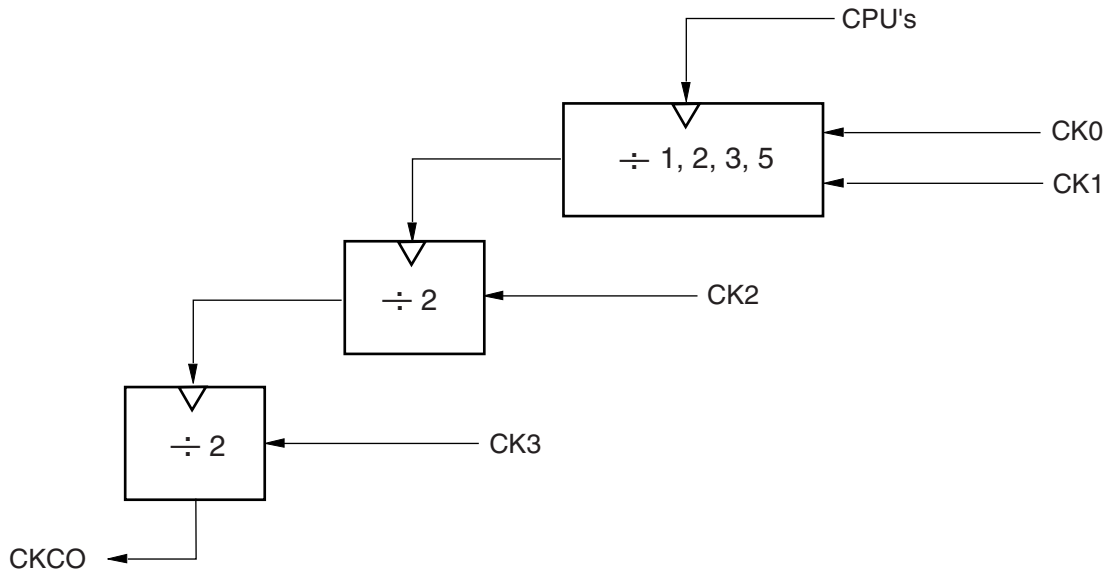
The prescaler (CK0-CK3) is chosen in the DIAG_CTRL_REG register.

VAN Core :

The VAN Core incorporates two main state machines (transmission and reception) and controls the output driver TxVAN, the CRC logic and the Tx/Rx shift register. It also controls the synchronization to the VAN bus (according to VAN specifications) by the Bit Time Logic (BTL). It also detects all the symbols included in a VAN frame like the Start Of Frame (SOF), the End Of Data (EOD), the Acknowledge (ACK), the End Of Frame (EOF) or the Inter Frame Separation (IFS). It codes and decodes any VAN data according to the Enhanced-Manchester code.

Bit Stream Processor (BSP) :

The BSP is a sequencer that controls the data stream between the IML (parallel data) and the VAN bus line (serial data). It controls the BTL with regard to transmission, reception, arbitration and generates error signals according to the VAN bus specifications.

Error Management Logic (EML) :

The EML is responsible for the fault confinement of the VAN protocol. It also sets and resets the error flag bits and interrupts and changes the error status bits in the Status register.
Any error on the VAN bus line generates an interrupt if enabled by the application software (INT0 interrupt).

Cyclic Redundancy Check (CRC) generator and checker :

The CRC generator consists of a 15-bit shift register and the logic required to generate the checksum of the bit-stream. It informs the EML about the result of a receiver checksum.

The checksum is generated by the polynomial :

$$g(x) = x^{15} + x^{11} + x^{10} + x^9 + x^8 + x^7 + x^4 + x^3 + x^2 + 1$$

This logic performs the calculation of the CRC in transmission and in reception.

Receive/Transmit (RX/TX) register :

The Rx/Tx register is a 8-bit shift register controlled by the VAN Core. It is loaded or read by the IML which holds the data to be transmitted or the data that was received.

Bit Time Logic (BTL) :

The BTL is responsible for counting the bits and the bytes. It also resynchronise the bits according to VAN specifications.

Diagnosis Logic and Output Driver:

The Diagnosis Logic is responsible to hold the communication whenever one of the two wires of the VAN bus line (DATA and /DATA) is short-circuited to ground or battery or is opened-circuit. It decides on which line Rx0VAN, Rx1VAN or Rx2VAN, the VAN UART will continue to communicate. Operating on the RXD0 line is named «nominal or differential mode» because there is no default neither on the DATA line nor on the /DATA one.

Operating on the Rx1VAN or Rx2VAN line is named «degraded mode» since there is a default on DATA or /DATA and it is no longer a differential communication.

Assuming the Diagnosis Logic decides to put the device in the «degraded mode», it can also put it back to the «differential mode» when the problem on the DATA or /DATA has disapeared.

VAN UART Registers :

The register block consists of 21 registers which are described in more details in the following paragraphs.

15.3   Functional description

15.3.1 Overview of the VAN UART Registers

Figure 15-4: Overview of the VAN UART Registers

Register	Addr								
RK0_REG	F800H		TX6	TX5	TX4	TX3	TX2	TX1	TX0
IFR_REG	F801H	IFR7	IFR6	IFR5	IFR4	IFR3	IFR2	IFR1	IFR0
CTRK_REG	F802H	0	0	0	0	STOP-TR	ACK-REQ	LAST-BYTE	SOFT-RESET
CONFIG_REG	F803H	0	0	0	IT12	RANK	IFR	MSK1	MSK0
DIAG_CTRL_REG	F804H	CK3	CK2	CK1	CK0	DIAG-TOP	ENAB_EMECB	DIA1	DIA0
MSK1_MSG_REG	F805H								
MSK1_LSG_REG	F806H						0	0	0
AC1_MSG_REG	F807H								
AC1_LSG_REG	F808H						0	0	0
MSK2_MSG_REG	F809H								
MSK2_LSG_REG	F80AH						0	0	0
AC2_MSG_REG	F80BH								
AC2_LSG_REG	F80CH						0	0	0
AC3_MSG_REG	F80DH								
AC3_LSG_REG	F80EH						0	0	0
AC4_MSG_REG	F80FH								
AC4_LSG_REG	F810H						0	0	0
STAT_REG	F811H	0	LA_RESP	EOM	LA	ACK	ERR2	ERR1	ERR0
REC_REG	F812H	RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0
DIAG_STAT_REG	F813H	0	0	0	0	0	SC	SB	SA
INT_ENABLE_REG	F820H	GIE	RDAE	TBEE	FTE	FRE	LAE	EOME	0

15.3.2   Autonomous mode functions

15.3.2.1 Autonomous mode features
The user sets the VAN UART in autonomous mode by setting the RANK bit to 0. The transmission clock is the quartz clock divided by the prescaler chosen by the user in the DIAG_CTRL_REG register.
For example:
To be able to detect the frames, whose speed is 250 kTS/s, the minimum frequency of this quartz clock must be 4 MHz.
The component executes all VAN frame types:
* The transmission of data transmit (write) or data request (read) frames (SOF included or rank 0) at any speed up to 500 kTS/s (with an 8 MHz quartz) depending on the division ratio chosen in the DIAG_CTRL_REG register.
* The reception of data frames at the same speeds depending on the programming of the prescaler.
* The transmission of data transmit or data request frames from the address field (synchronisation on the start bit or rank 1) at any speed depending on the programming of the prescaler.
* The transmission of in frame responses (or rank 16) at any speed depending on the programming of the prescaler.

15.3.2.2 Programming of the prescaler in Rank 0 transmission (SOF included)
Programming of the prescaler permits Rank 0 frames to be transmitted at different speeds without changing the quartz clock.
For example:
W hen an 8 MHz quartz clock supplies the UART, it is capable of sweeping the range 62,5 kTS/s to 500 kTS/s in rank 0 transmission.
The prescaler is chosen using the DIAG_CTRL_REG register with the 4 bits CK3, CK2, CK1 and CK0. The 2 least significant bits CK1 and CK0 are used to program a divider by 1, 2, 3 or 5 whilst the 2 other bits are used to program a divider to a power of 2.

Figure 15-5: Prescaler in Rank 0 transmission
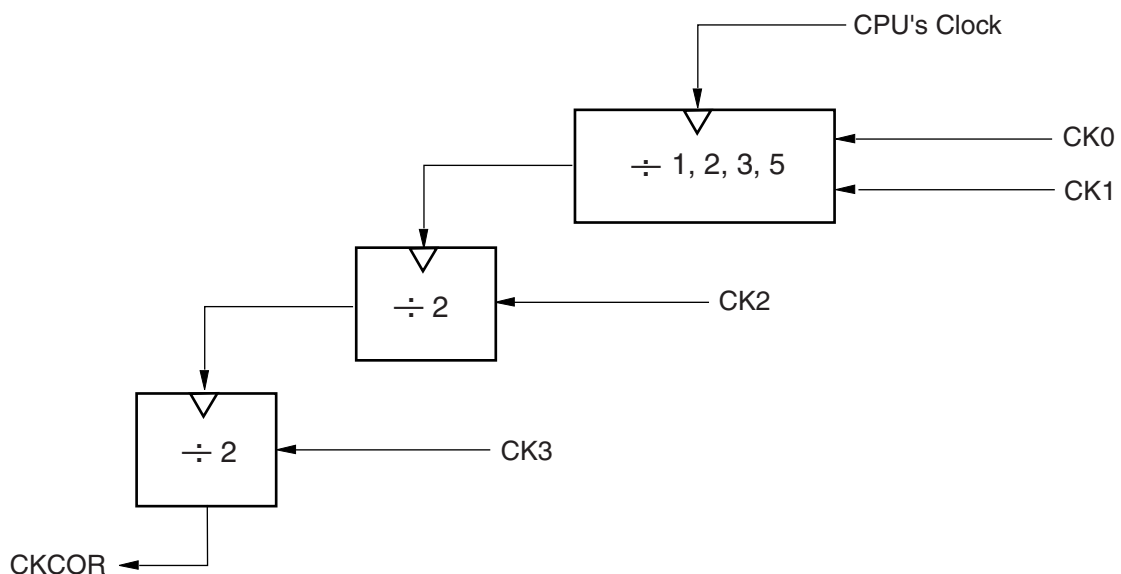
Table 15-1: Network Speeds as a Function of the Quartz Clock and the Chosen Division Ratio

Quartz (MHz)		1	2	3	4	5	6	8
Div	Ratio	Network speed (KTS)						
0000	1	62.5	125		250			500
0001	2	31.25	62.5		125			250
0010	3			62.5			125	
0011	5					62.5		
0100	2	31.25	62.5		125			250
0101	4	15.625	31.25		62.5			125
0110	6			31.25			62.5	
0111	10					31.25		
1000	2	31.25	62.5		125			250
1001	4	15.625	31.25		62.5			125
1010	6			31.25			62.5	
1011	10					31.25		
1100	4	15.625	31.25		62.5			125
1101	8	7.81	15.625		31.25			62.5
1110	12			15.625			31.25	
1111	20					15.625		

15.3.2.3 Transmission features in autonomous mode

A transmit request is triggered by writing in the rank0 transmit register RK0_REG when the component is in receive or in idle (typically after an EOM interrupt).

A rank 0 transmission start by the transmission of the SOF symbol following the detection of the EOF symbol (8 recessive TS) followed by the IFS symbol (4 recessive TS).

If these 12 recessive TS could not be detected on the network, the component then synchronises itself on the start bit seen on the bus. The transmission request is satisfied but transformed into rank 1 transmission.

In the autonomous mode, the component performs also the in frame response (IFR). To do this, the bit IFR must be set to 1 in the CTRL_REG register. In addition, the component must be in reception on the R/W bit of the command field of the VAN frame (please note that this receive state can be due to a lost of arbitration during the first or the second identifier byte).

The VAN UART compares the received identifier with one or more identifiers located in the MSK (mask) and AC (Acceptance Code) registers and generates or not a received byte interrupt. Then, the microcontroller accepts or refuses to respond in the frame (whether this identifier corresponds or not to an in frame response).

Writing of the first byte of the response in the IFR transmit register IFR_REG shows an acceptation.

Not writing shows a refusal.

15.3.3 Synchronous mode functions

15.3.3.1 Synchronous mode features

The user sets the VAN UART in synchronous mode by setting the RANK bit to 1 in the control register CTRL_REG. The transmission clock is the quartz clock divided by the prescaler chosen by the user in the DIAG_CTRL_REG register.

For example:

To be able to detect the frames, whose speed is 250 kTS/s, the minimum frequency of this quartz clock must be 4 MHz.

The component can no longer transmit rank 0 frames. However, it can receive data frames. It can transmit rank 1 frames (data frames and read frames synchronised on the start bit) and in frame responses. The range of speeds depends on the frequency of the quartz clock; at 8MHz, the range spreads from 62,5 kTS/s to 500 kTS/s.

15.3.3.2 Transmission features in synchronous mode

For rank 1 transmission, the transmit request is still triggered by writing in the Rank0 transmit register RK0_REG when the component is in receive or in idle (typically after an EOM interrupt). The transmission is triggered after the detection of a start bit. The transmission characteristic of an in frame response is identical to that mentioned in autonomous mode.

15.3.4 Handling of a collision

The UART automatically goes into reception during a lost arbitration after collision detection. This lost arbitration may be signalled either by interrupt, if it is enabled by the user (LAE bit of the INT_ENABLE_REG register), or by reading the LA bit in the status register STAT_REG.

15.3.5 Executing the CRC

15.3.5.1 CRC transmission

The transmission of the CRC is possible thanks to a CRC module integrated in the UART.

It is performed by the following way:

The LAST-BYTE bit in the CTRL_REG register is set when there are no more bytes to transmit. The UART then automatically completes the frame by the two CRC bytes followed by the EOD symbol. In the case of a read frame, the LAST-BYTE bit should be set after the second identifier byte because if the requested node does not send its data, the UART will complete the frame by sending immediately the 2 CRC bytes.

Therefore, such a frame does not contain any data. This case is described in detail further.

15.3.5.2 Reception of the CRC

For high-speed applications, the UART incorporates a CRC module, which compares the received CRC with the calculated CRC. This comparison is carried out in transmission and in reception, giving place, in the latter case, to the transmission of a possible acknowledge.

15.3.6 Control of the acknowledge bit

In reception, if the EOD symbol has been detected and if the CRC is correct, then if the ACK-REQ bit is set to 1 in the CTRL_REG register before the end of the EOD field, a positive acknowledge is transmitted. Otherwise, the UART stays in reception, which is equivalent to a negative acknowledge.

The acknowledge bit is decoded in transmission as in reception and its value is indicated in the STAT_REG register by the ACK bit. The microcontroller compares the value of the ACK bit with the RAK bit received (and memorised) in the command field of the VAN frame.

15.3.7 Error control and Interrupt control

15.3.7.1 Error control

3 bits ERR2, ERR1 and ERR0 encode any error in transmission or in reception in the status register STAT_REG.

Table 15-2: Error Table

ERR2	ERR1	ERR0	Type of error
0	0	0	no error : initialisation
0	0	1	Physical violation
0	1	0	Not used
0	1	1	Code Violation in reception
1	0	0	Not used
1	0	1	CRC error in reception
1	1	0	Format error (ACK)
1	1	1	Transmission or Reception lock up

Information on the error table:

When the code violation received is 00 on the TS 8 and 9 of a byte, the error signalled is a CRC error in reception as it is not possible to distinguish this violation from the EOD symbol. Any other code violation received is signalled by a code violation in reception.

15.3.7.2 Interrupt control

An error is signalled by an interrupt if the user defines it. Any interrupt that would have been generated after the detection of an error is deleted.

The interrupt sources are listed below:

LA_RESP   :   Lost arbitration in the RTR bit (Response)
EOM   :   End of message
LA   :   Lost arbitration
FT   :   Failed transmit (refer to ERR0, ERR1, ERR2 for status)
FR   :   Failed receive (refer to ERR0, ERR1, ERR2 for status)
These sources generate the INT0 interrupt.

TBE   :   Transmit buffer empty
                  This source generates the INT1 interrupt.
RDA   :   Received data available
                  This source generates the INT2 interrupt.

- EOM interrupt

The EOM interrupt appears at the end of the acknowledgement field if no error has occurred in the frame. Otherwise, it appears as soon as an error is detected. This permits, in particular to detect errors that could occur in the identification field and to synchronise on it.
This interrupt is generated on INT0.
It can be disabled in the INT_ENABLE_REG register by the EOME bit.
It can also be masked by VEMK bit in MKOL register.

- LA interrupt

The LA interrupt appears at the end of the byte where the collision occurred even if the UART has automatically switched to the reception mode in the current Time Slot.
This interrupt is also generated on INT0.
This interrupt is signalled in the REG-STAT register by the LA bit.
It can be disabled in the INT_ENABLE_REG register by the LAE bit.

- LA_RESP interrupt

The LA_RESP interrupt appears when the UART performs a read frame and when the collision occurred on the RTR bit. That means that response is in progress. The UART has automatically switched to the reception mode to receive that response.
This interrupt is also generated on the INT0 pin.
This interrupt is signalled in the REG-STAT register by the LA_RESP bit.
It can be also disabled in the INT_ENABLE_REG register by the LAE bit.

- TBE interrupt

The TBE interrupt appears at the start of the 9th TS of a new byte before the old RK0_REG or IFR_REG register has been loaded in the transmit/receive shift register.
This interrupt is generated on INT1.
It can be disabled in the INT_ENABLE_REG register by the TBEE bit.
It can also be masked by VTMK bit in MKOL register.

It signifies that a byte must be loaded into the RK0_REG or IFR_REG register, but can be ignored if the microcontroller has no more bytes to transmit. In this case, it sets the LAST-BYTE bit in the control register CTRL_REG for the transmission of the CRC.

- RDA interrupt
The RDA interrupt appears at the start of the 9th TS of the current byte before the RECEP_REG register has been loaded by the transmit/receive shift register.
This interrupt is generated on INT2.
It can be disabled in the INT_ENABLE_REG register by the bit RDAE.
It can also be masked by VRMK bit in MKOL register.
It signifies that the byte contained in the RECEP_REG register must be read.

- Case of an in frame response:
The user can choose to perform or not the in frame response using the IFR bit in the control register CTRL_REG.
If IFR = 0, the component cannot perform the in frame response.
If IFR = 1, the component is able to respond in the frame under conditions (see the transmission characteristics of rank 16). The interrupts are generated following two manners:
If IT12 = 0, the interrupts are generated byte after byte. The comparison of the identifier field is made by UART.
If IT12 = 1, the interrupts are generated byte after byte except during the second byte of the identifier where one RDA interrupt appears at the end of the 12th bit of the VAN identi fier field. This allows the microcontroller to make the comparison itself. In this case, the UART supplies the byte for the address comparison and helps the microcontroller to search for the byte to be transmitted in the in frame response…

Table 15-3: Frame Responce

0 0 0 0	IDEN2
4 bits	4 bits

…after the RDA interrupt at the 12th bit, so as to be able to add to an address to point on the table of bytes to be transmitted without needing to mask the 4 most significant bits of this byte.

- FT interrupt
The FT interrupt appears after a physical violation, a format error (acknowledge error in transmission) or a transmission lock-up (when there is no write access to the transmission register or to the control register between the transmission of two consecutive bytes). In case of a transmission lock-up, the UART does not complete the frame with the two bytes of CRC and stops just after the last byte loaded.

The bits ERR2, ERR1 and ERR0 signal the error in the status register STAT_REG.
It is generated on INT0.

It can be disabled in the INT_ENABLE_REG register by the FTE bit.

- FR interrupt

The FR interrupt appears after a code violation, a CRC error or a format error (acknowledgement error in reception) or a reception lock-up (when there is no read access to the reception register between the reception of two consecutive bytes). In case of a reception lock-up, the UART does not receive the rest of the frame and stops just after the last byte.

The bits ERR2, ERR1 and ERR0 signal the error in the status register STAT_REG.

It is generated on INT0.

It can be disabled in the INT_ENABLE_REG register by the FRE bit.

15.4 VAN UART Registers

The VAN UART consists of the following registers.

Table 15-4: VAN UART Registers

Address	Register NAME	SYMBOL	After Reset	R/W	Manipulatable bit unit		
					1bit	8bit	16bit
F800H	Rank 0 Register	RK0_REG	FFH	R/W	O	O	X
F801H	In Frame Transmit Register	IFR_REG	FFH	R/W	O	O	X
F802H	Control Register	CTRL_REG	00H	R/W	O	O	X
F803H	Configuration Register	CONF_REG	08H	R/W	O	O	X
F804H	Diagnosis Control Register	DIAG_CTRL_REG	17H	R/W	O	O	X
F805H	Mask1 Register	MSK1_MSB_REG	00H	R/W	O	O	X
F806H	Mask1 Register	MSK1_LSB_REG	00H	R/W	O	O	X
F807H	Acceptance Code 1	AC1_MSB_REG	00H	R/W	O	O	X
F808H	Acceptance Code 1	AC1_LSB_REG	00H	R/W	O	O	X
F809H	Mask2 Register	MSK2_MSB_REG	00H	R/W	O	O	X
F80AH	Mask2 Register	MSK2_LSB_REG	00H	R/W	O	O	X
F80BH	Acceptance Code	AC2_MSB_REG	00H	R/W	O	O	X
F80CH	Acceptance Code	AC2_LSB_REG	00H	R/W	O	O	X
F80DH	Acceptance Code	AC3_MSB_REG	00H	R/W	O	O	X
F80EH	Acceptance Code	AC3_LSB_REG	00H	R/W	O	O	X
F80FH	Acceptance Code	AC4_MSB_REG	00H	R/W	O	O	X
F810H	Acceptance Code	AC4_LSB_REG	00H	R/W	O	O	X
F811H	Status Register	STAT_REG	08H	R	O	O	X
F812H	Receive Register	REC_REG	FFH	R	O	O	X
F813H	Diagnosis Status Register	DIAG_STAT_REG	00H	R	O	O	X
F820H	Interrupt Enable Register	INT_ENABLE_REG	00H	R/W	O	O	X

15.4.1 Rank0 Transmission Register (RK0_REG)

The rank0 transmission register is loaded by the microcontroller to trigger a transmit request. RK0_REG is set with a 1-bit or 8-bit manipulation instruction.

$\overline{\text{RESET}}$ input set this register to FFH.

Figure 15-6:   Rank0 Transmission Register Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
RK0_REG	TX7	TX6	TX5	TX4	TX3	TX2	TX1	TX0	F800H	FFH	R/W

It is also loaded each time the INT1 interrupt is generated, except if the microcontroller has no more bytes to transmit. In this case, it sets, instead, the LAST-BYTE bit in the control register CTRL_REG.

For a standard transmission (rank0 or rank1), the microcontroller has up to one byte duration to load this register.

For the in frame response, it has up to one byte duration if the IT12 bit is set to 0 or up to only 4 TS if the IT12 bit is set to 1.

The loading limit is 14/16 of the last TS of the byte. If this limit is no met, the component will detect a lock up error and will signal it.

The transmission is done MSB first (TX7 is transmitted first).

15.4.2 In Frame Response Register (IFR_REG)

The IFR Transmit Register is written when the user wish to transmit an In Frame Response (IFR). IFR_REG is set with a 1-bit or 8-bit manipulation instruction.

$\overline{\text{RESET}}$ input set this register to FFH.

Figure 15-7:   Frame Responce Register Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
IFR_REG	IFR7	IFR6	IFR5	IFR4	IFR3	IFR2	IFR1	IFR0	F801H	FFH	R/W

The VAN UART will receive the identification field (12 bits), compares it with the Acceptance Codes and start the transmission by the 16th bit of the frame. This kind of transmission is named «rank16 frame».
No IFR is transmitted if this register is not written.
The application software should only write data bytes in this register (28 maximum) and not address bytes. These datas correspond to the datas to be answered in the IFR. This software must specify the last byte of data in the CTRL_REG register.
The device will transmit an In Frame Response only if the identification field that was received on the VAN bus matches with one of the Acceptance Codes.
Every byte transmitted generates a INT1 interrupt corresponding to the TBE status (Transmit Buffer Empty) meaning that the IFR_REG register was loaded in the shift register. A writing in this register resets the internal TBE flag.

- Case of lost arbitration during the identification field of a rank0 frame
The following picture shows an arbitration during the identification field of a rank0 frame. That means the VAN UART has first tried to transmit a rank0 frame. Nevertherless , at the same moment, another VAN node is also communicating with a higher priority identification field. The VAN UART looses the arbitration and goes into the receive mode.
It can happen that this frame was also a request frame for the VAN UART.
In order to handle these cases, the application software has to write in both registers (RK0 and IFR) to prevent from this kind of arbitration. The VAN UART will then select automatically the right register. If a lost arbitration has occured, the IFR_REG is selected otherwise the RK0_REG is chosen.

Figure 15-8:   Frame Responce Register Function
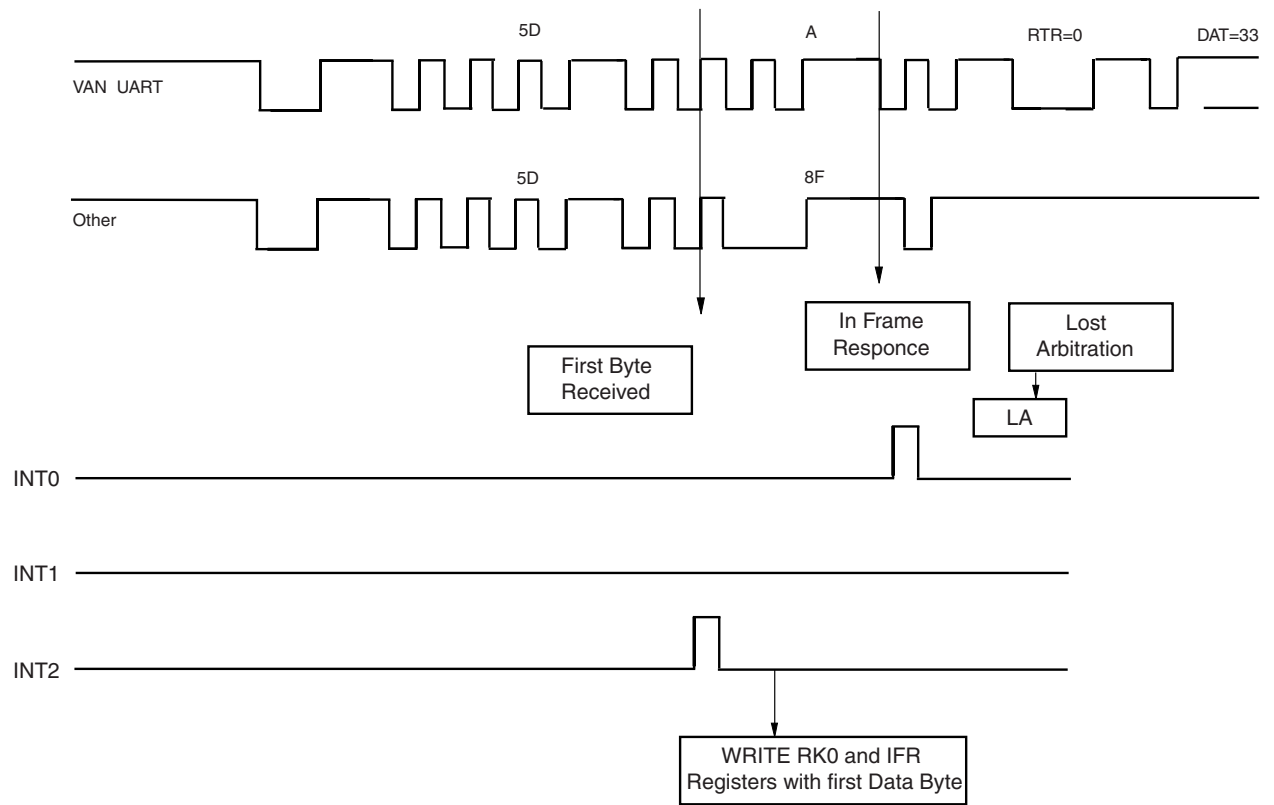
15.4.3 Control Register (CTRL_REG)

The Control Register is used to control the VAN UART during the transmisision or to initiate a RESET.
CTRL_REG is set with a 1-bit or 8-bit manipulation instruction.

RESET input set this register to 00H.

Figure 15-9:   Control Register Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
CTRL_REG	0	0	0	0	STOP-TR	ACK-REQ	LAST-BYTE	SOFT-RESET	F802H	00H	R/W

Note:
The bits of this register are "SET ONLY type " bits. They are set by the application software and resetted automatically by the VAN UART. Writing 0 in these bits will have no effect.

STOP-TR: Stop Transmit

Table 15-5:   Stop Transmit

STOP-TR	Stop Transmit
0	No influence
1	Stop the transmission in progress

It can be used in any type of transmission.

ACK-REQ: Acknowledge Request

Table 15-6:   Acknowledge Request

ACK-REQ	Acknowledge Request
0	No influence
1	Transmit request of an acknowledge bit

The microcontroller decodes the value of the RAK bit (bit 2 of the 2nd byte of the frame). According to this value, it will choose to set the ACK-REQ bit in the control register CTRL-REG or not. Note that ACK-REQ occupies the same position as RAK in the byte.

Figure 15-10:   Control Register Block Diagram

Byte 2 of the Frame memorised in the microcontroller	IDEN3	IDEN2	IDEN1	IDEN0	EXT	RAK	R/W	RTR
CTRL_REG of the UART					STOP TR	ACK REQ	LAST BYTE	SOFT RESET
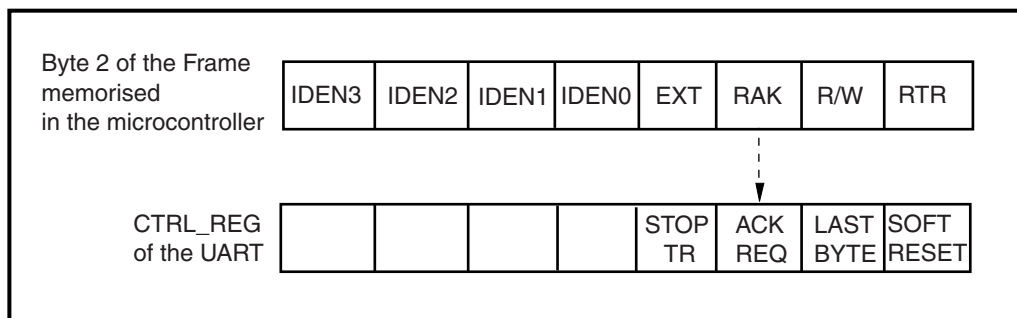
Therefore, a mask with 04h of the 2nd byte needs to be made and written in the control register.

Figure 15-11:   Control Register Function
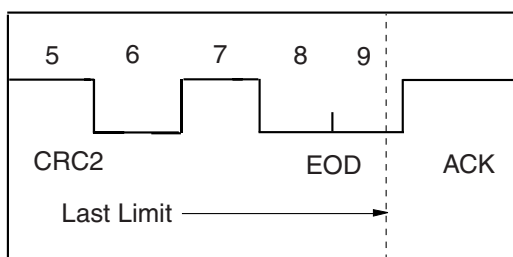
The last limit for setting the ACK-REQ is 13/16 of the 2nd TS of the EOD symbol.
Following the results of the frame (identifier recognised and correct CRC), the acknowledge bit may be transmitted.

LAST-BYTE:

Table 15-7:   Last-Byte

LAST-BYTE	Last transmission Byte
0	No influence
1	Sign to the VAN UART that the current byte is the last one

Figure 15-12:   Last-Byte
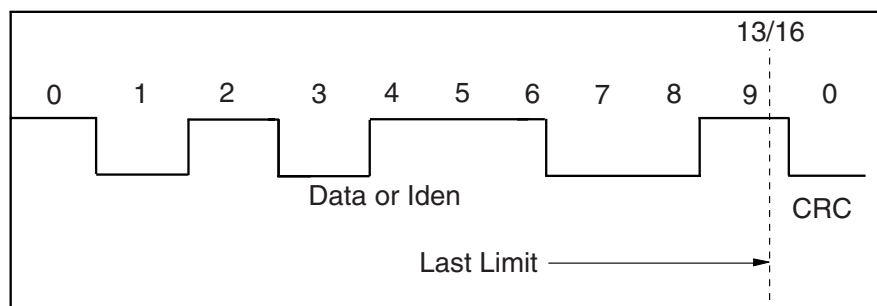
The UART places the 2 CRC bytes after it. This may occur during a write frame or a read frame: in the case of a write frame, the last byte of data is signalled after the last data is transmitted. In the case of a read frame, the last byte is signalled after loading the 2nd identifier. The LAST-BYTE bit must be activated, as, if the response is missing, the CRC will be automatically set by the UART following the identifier n°2.

SOFT-RESET: Software reset

Table 15-8:   Software Reset

SOFT-RESET	Soft Reset
0	No influence
1	Software reset with the initialisation of the VAN UART

This bit should be used if a major problem is detected during the operation of the VAN UART, or if it is incorrectly used. The result is the same as a hardware reset. The VAN UART must be re-configured.

15.4.4 Configuration Register (CONF_REG)

The Configuration Register is used to configure the interrupt generation, the UART mode and response and the mask function.
CONF_REG is set with a 1-bit or 8-bit manipulation instruction.

$\overline{\text{RESET}}$ input set this register to 08H.

Figure 15-13:   Configuration Register (CONF_REG) Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
CONF_REG	0	0	0	IT12	RANK	IFR	MSK1	MSK0	F803H	08H	R/W

IT12: Enable / Disable interrupt on the 12th bit of the identifier field.

Table 15-9:   Enable / Disable interrupt on the 12th bit of the identifier field

IT12	Interrupt on the 12th bit of the identifier field
0	Disables the interrupt on the 12th bit of the identifier field. The UART only supplies «byte» interrupts during a frame.
1	Enables the interrupt on the 12th bit of the identifier field. This allows the microcontroler to receive the whole identifier and to compare it if necessary.

Case where IT12 = 0

Figure 15-14:   Case where IT12 = 0

251
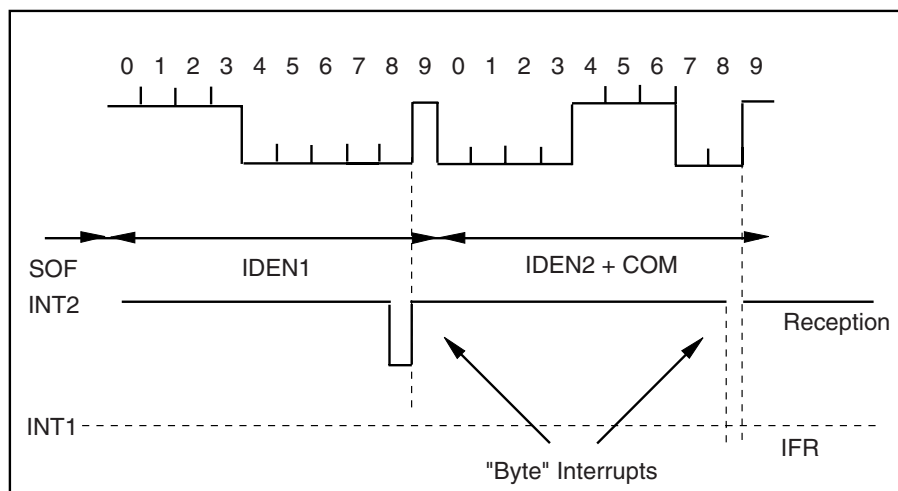
Case where IT12 = 1

Figure 15-15:   Case where IT12 = 1
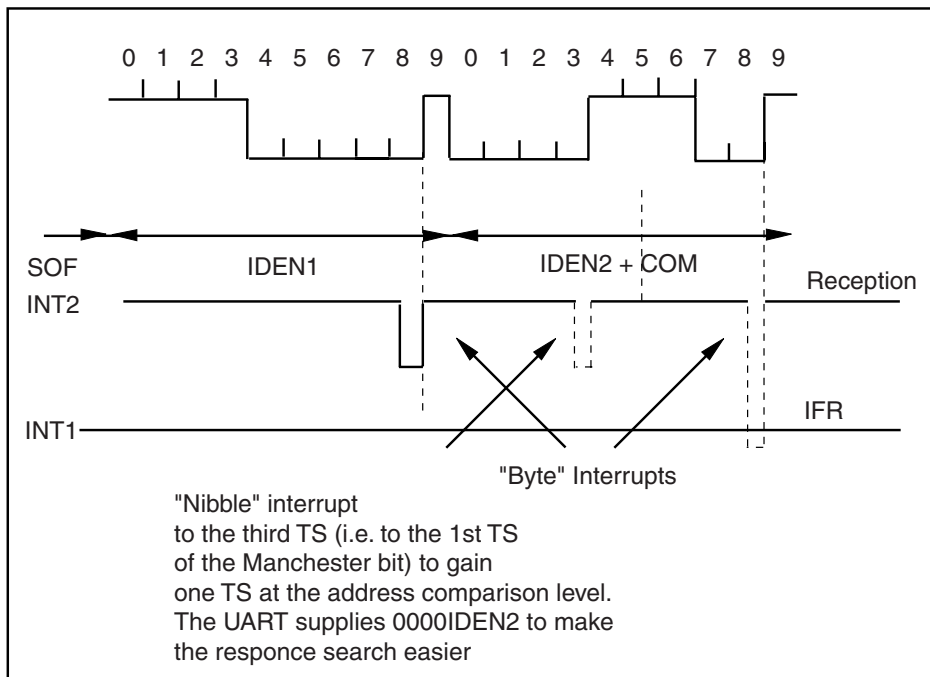
RANK: Rank 0 / Rank 1 mode

Table 15-10:   Rank 0 / Rank 1 mode

RANK	VAN UART Mode Selection
0	VAN UART in autonomous mode
1	VAN UART in synchronous mode

In autonomous mode, a quartz clock is compulsory for the generation of the SOF symbol. The precision needed is +/-1%.

Remark: On initialisation, the UART is set in synchronous mode and disables the in frame response.

IFR: Enable / Disable In Frame Response

Table 15-11: Enable / Disable In Frame Response

IFR	In Frame Response
0	Disables the in frame response
1	Enables the in frame response

MSK1, MSK0: Mask Enable / Disable

Table 15-12: Mask Enable / Disable

MSK1	MSK0	Function
0	0	Masks 1 and 2 activated (all identifiers filtered)
0	1	Mask1 inhibited
1	0	Mask2 inhibited
1	1	Masks 1 and 2 inhibited (all identifiers accepted)

MSK1 and MSK0 combinations allow enabling or disabling all or part of the mask mechanism applied on the identification field described further on.

15.4.5 Diagnosis Control Register (DIAG_CTRL_REG)

The Diagnosis Control Register allows to configure the bus speed, the communication mode and diagnostic functions.
DIAG_CTRL_REG is set with a 1-bit or 8-bit manipulation instruction.

$\overline{\text{RESET}}$ input set this register to 17H.

Figure 15-16: Diagnosis Control Register (DIAG_CTRL_REG) Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
DIAG_CT RL_REG	CK3	CK2	CK1	CK0	DIAG-TOP	ENAB_E MECB	DIA1	DIA0	F804H	17H	R/W

CK3, CK2, CK1 and CK0: Prescaler
The prescaler is used to fix the division ratio between the quartz clock and the speed of the bus. This prescaler is defined in 4 bits.
The least significant bits CK3 and CK2 are used to pre-divide by a ratio of 1,2,3 or 5. So, the UART can operate with quartz frequencies other than to the powers of 2.
This pre-divider by 3 or 5 permits an operation at "round" speeds in terms of Kbits/s or KTS/s with "non binary" frequencies such as 3, 5, 6 MHz.

Figure 15-17: Prescaler Block Diagram
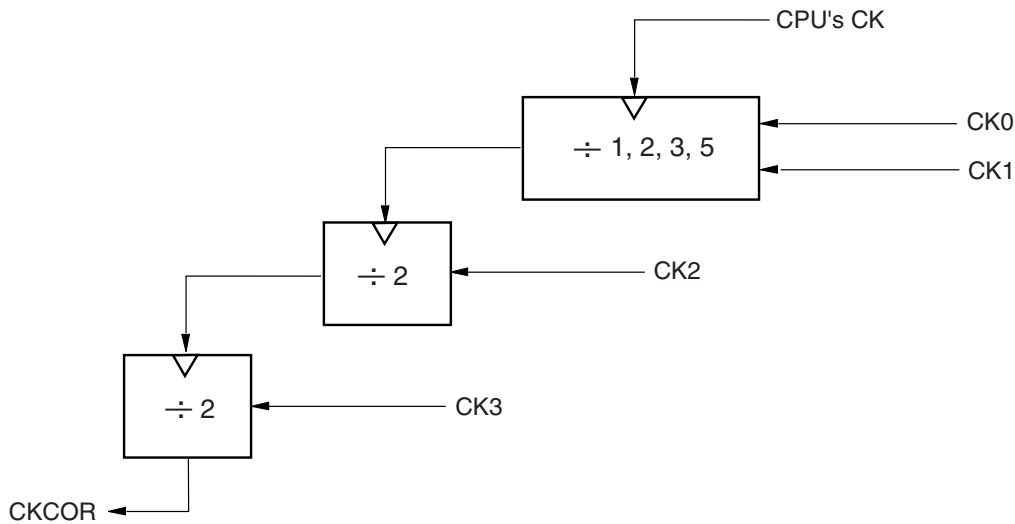
Table 15-13: Prescaler - Network Speeds as a Function of the Quartz Clock and the Chosen Division Ratio

Quartz (MHz)		1	2	3	4	5	6	8
Div	Ratio	Network speed (KTS)						
0000	1	62.5	125		250			500
0001	2	31.25	62.5		125			250
0010	3			62.5			125	
0011	5					62.5		
0100	2	31.25	62.5		125			250
0101	4	15.625	31.25		62.5			125
0110	6			31.25			62.5	
0111	10					31.25		
1000	2	31.25	62.5		125			250
1001	4	15.625	31.25		62.5			125
1010	6			31.25			62.5	
1011	10					31.25		
1100	4	15.625	31.25		62.5			125
1101	8	7.81	15.625		31.25			62.5
1110	12			15.625			31.25	
1111	20					15.625		

DIAG-TOP: Synchronous diagnosis clock

Table 15-14: Synchronous Diagnosis Clock

DIAG-TOP	Synchronous diagnosis clock selection
0	No pulse on the internal DIAG-CLOCK signal
1	Pulse on the internal DIAG-CLOCK signal

The pulse on the internal DIAG-CLOCK signal is used for the synchronous diagnosis clock (see Information on the characteristics of the clock DIAG-CLOCK in the paragraph describing the diagnosis function).

EN-EMECB: Enable the transmit diagnosis

Table 15-15:   Enable the Transmit Diagnosis

EN-EMECB	Transmit diagnostic
0	Enables the transmission diagnosis
1	Disables the transmission diagnosis

Due to the diagnosis set-up problems in transmission, this bit permits this part of the diagnosis to be disabled or enabled.

DIA1, DIA0: Choice of communication mode

Table 15-16:   Choice of Communication Mode

DIA1	DIA0	Communication mode
0	0	Forced operation on RXD0
0	1	Forced operation on RXD1
1	0	Forced operation on RXD2
1	1	Automatic operation

The 2 least significant bits DIA1 and DIA0 allow the user to choose the communication mode.

15.4.6 Mask1 registers (MSK1_MSB_REG, MSK1_LSB_REG)

These 2 registers allow to compare the 12 bits of the VAN identification field plus the EXT bit.
M̄S̄K̄1_MSB_REG, M̄S̄K̄1_LSB_REG is set with a 1-bit or 8-bit manipulation instruction.
R̄ĒS̄ĒT̄ input sets these registers to 00H.

Figure 15-18-1: Mask1 register MSK1_MSB_REG Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
MSK_MSB_REG	B11	B10	B9	B8	B7	B6	B5	B4	F805H	00H	R/W

Figure 15-18-2: Mask1 register MSK1_LSB_REG Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
MSK_LSB_REG	B3	B2	B1	B0	Ext	0	0	0	F806H	00H	R/W

Writing «0» enables the comparison of the corresponding bit.
Writing «1» disables the comparison of the corresponding bit that becomes a «don't care bit».

15.4.7 Acceptance Code 1 registers (AC1_MSB_REG, AC1_LSB_REG)

These 2 registers allow to choose the code acceptance which is the value of the identification field that the user wish to match with. They work together with the MSK1 registers.
AC1_MSB_REG, AC1_LSB_REG is set with a 1-bit or 8-bit manipulation instruction.
$\overline{\text{RESET}}$ input sets these registers to 00H.

Figure 15-19-1:   Acceptance Code 1 register AC1_MSB_REG

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
AC1_MSB_REG									F807H	00H	R/W

Figure 15-19-2:   Acceptance Code 1 register AC1_LSB_REG

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
AC1_LSB_REG									F808H	00H	R/W

The behaviour of the receive interrupt (INT2) according this comparison is described in the paragraph «Receive Interrupt Behaviour».

15.4.8 Mask2 registers (MSK2_MSB_REG, MSK2_LSB_REG)

These 2 registers allow to compare the 12 bits of the VAN identification field plus the EXT bit.
MSK1_MSB_REG, MSK1_LSB_REG is set with a 1-bit or 8-bit manipulation instruction.
RESET input sets these registers to 00H.

Figure 15-20-1:   Mask2 register MSK2_MSB_REG Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
MSK2_MSB _REG	B11	B10	B9	B8	B7	B6	B5	B4	F809H	00H	R/W

Figure 15-20-2:   Mask2 register MSK2_LSB_REG Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
MSK2_LSB _REG	B3	B2	B1	B0	Ext	0	0	0	F80AH	00H	R/W

Writing «0» enables the comparison of the corresponding bit.
Writing «1» disables the comparison of the corresponding bit that becomes a «don't care bit».

15.4.9 Acceptance Code 2, 3 and 4 Registers (AC2_MSB_REG, AC2_LSB_REG, AC3_MSB_REG, AC3_LSB_REG, AC4_MSB_REG, AC4_LSB_REG)

These 6 registers allow to choose the code acceptance which is the value of the identification field that the user wish to match with. They work together with the MSK2 registers.
AC2_MSB_REG, AC2_LSB_REG, AC3_MSB_REG, AC3_LSB_REG, AC4_MSB_REG, AC4_LSB_REG are set with a 1-bit or 8-bit manipulation instruction.
RESET input sets these registers to 00H.

Figure 15-21:   Acceptance Code 2, 3 and 4 Registers Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
AC2_MSB_REG	B11	B10	B9	B8	B7	B6	B5	B4	F80BH	00H	R/W

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
AC2_LSB_REG	B3	B2	B1	B0	Ext	0	0	0	F80CH	00H	R/W

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
AC3_MSB_REG	B11	B10	B9	B8	B7	B6	B5	B4	F80DH	00H	R/W

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
AC3_LSB_REG	B3	B2	B1	B0	Ext	0	0	0	F80EH	00H	R/W

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
AC4_MSB_REG	B11	B10	B9	B8	B7	B6	B5	B4	F80FH	00H	R/W

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
AC4_LSB_REG	B3	B2	B1	B0	Ext	0	0	0	F810H	00H	R/W

The behaviour of the receive interrupt (INT2) according this comparison is described in the paragraph «Receive Interrupt Behaviour».

15.4.10 Status Register (STAT_REG)

This register allows to control a lost arbitration, the end of message, the acknowledge and the error type during a transmission or a reception.
STAT_REG can be read with a 1-bit or an 8-bit manipulation instruction.
$\overline{RESET}$ input sets this register to 08H.

Figure 15-22:   Status Register (STAT_REG) Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
STAT_REG	0	LA_RESP	EOM	LA	ACK	ERR 2	ERR 1	ERR 0	F811H	08H	R

LA_RESP, LA: Lost arbitration information

Table 15-17:   LA_RESP, LA

LA_RESP	Lost Arbitration information
0	Arbitration is not lost during RTR bit
1	Arbitration lost during the RTR bit of the command field. It is considered as a lost arbitration due to a response.

LA	Lost Arbitration information
0	Arbitration is not lost
1	Arbitration lost not in the RTR bit of the command field

The UART automatically goes into reception after loosing arbitration during a collision.

These 2 kinds of collision may be signalled either by interrupt (INT0), if enabled by the user (LAE bit of the INT_ENABLE_REG register), or by reading these 2 bits in the status register STAT_REG. It is worthwhile noting that reading the status register causes all the bits to be reset to 0 (except ACK, which is set to 1).

EOM: End of message

Table 15-18:   EOM

EOM	End of Message
0	End of Message as not given under a.) or b.).
1	a.) If the frame is correct, the EOM flag is set after the EOD symbol and the ERR2, ERR1, ERR0 bits show 000. b.) If the frame is not correct, the EOM flag is also set when the error is detected and the ERR2, ERR1, ERR0 bits show this error.

261

The EOM flag is set when a VAN frame is transmitted or received correctly or incorrectly.
These 2 kinds of EOM may be signalled either by interrupt (INT0), if enabled by the user (EOME bit of the INT_ENABLE_REG register), or by reading the EOM bit in the status register STAT_REG.

During an EOM interrupt (INT0), the microcontroller can read:
- LA: Signals a possible collision with lost arbitration in the current frame. The application software should memorise this information to retry the transmission of this frame.
- LA_RESP: Indicates a lost arbitration during the RTR bit. This lost arbitration is due to a response.
- ACK: Indicates the value of the acknowledge bit:
            0: positive
            1: no acknowledge
             The ACK bit is described in the paragraph "Control of the acknowledge bit ".
- Err: Signals the type of transmit or receive error.

             The ERRx bits are described in the paragraph "Error control" where the bit combination are given.

15.4.11 Receive register (REC_REG)

This register is used as receive register of a reception.
STAT_REG can be read with a 1-bit or an 8-bit manipulation instruction.
RESET input sets this register to FFH.

Figure 15-23:   Receive register (REC_REG) Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
REC_REG	RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0	F812H	FFH	R

The receive register is read by the microcontroller each time the RDA interrupt (INT2) is generated by the UART indicating that a new byte is received.
The reading limit of the receive register is 13/16 of the last TS of the byte or 13/16 of the third TS of the second byte of identifier in case of IT12 is set. If this limit is not met, the component will detect an overrun and will signal a lock up error.
The reception is done MSB first (RX7 is received first).

Receive interrupt behaviour :
The RDA receive interrupt (INT2) is generated only if the received VAN identifier matches with one of the identifiers written in the ACx registers. The AC1 registers work with the MSK1 mask registers and the AC2, AC3 and AC4 registers work with the MSK2 mask registers.

Since the VAN identifier is built with 12 bits, it is received over 2 bytes. Three cases can occur :
* The received identifier does not match at all. The VAN UART does not produce any interrupt.
* The first byte matches but not the second one. The VAN UART generates the first receive interrupt (INT2) but since the second identifier byte does not match, the UART will wait for the end of the current frame to generate the EOM interrupt (INT0).
* The whole received identifier matches. The VAN UART generates all the receive interrups and the EOM interrupt.

15.4.12 Diagnosis Status Register (DIAG_STAT_REG)

This register is used for the diagnose of the receive lines.
DIAG_STAT_REG can be read with a 1-bit or an 8-bit manipulation instruction.
RESET input sets this register to 00H.

Figure 15-24: Diagnosis Status Register (DIAG_STAT_REG) Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
DIAG_STAT_REG	0	0	0	0	0	SC	SB	SA	F813H	00H	R

The bits SA and SB indicate the line chosen by the diagnosis circuit.

Table 15-19:   The bits SA and SB

SB	SA	Line chosen
0	0	Differential mode (Rx0VAN)-- No fault
0	1	DATAB mode (Rx2VAN) -- Fault on DATA
1	0	DATA mode (Rx1VAN) -- Fault on DATAB
1	1	Major Error

To perform this diagnosis, the circuit needs the synchronous diagnosis clock (SDC).
The synchronous diagnosis circuit is necessary to go back to the nominal mode, which is the differential mode.
If no fault is detected between two edges of this clock, the circuit goes back to the nominal mode (line Rx0VAN). This delay of one synchronous diagnosis clock period, is used to solve bad contact problems (on connectors for example). Thus, it is equal to a few milliseconds or even a few dozen milliseconds. Anyway, this is very large comparing to the TS clock (duration of TS).
To generate it, the user must set DIAG-TOP to 1 in the diagnosis control register DIAG_CTRL_REG.

Table 15-20:   The bit SC

SC	VAN UART comparator comparison
1	Discrepancy between the 3 comparator Rx0VAN, Rx1VAN and Rx2VAN during the reception.
0	No discrepancy between the 3 comparator Rx0VAN, Rx1VAN and Rx2VAN during the reception.

In normal operation, the SC bit equals 0, the 3 comparators give an identical result.

15.4.13 Interrupt enable register (INT_ENABLE_REG)

This register allows to enable/disable the interrupt sources of the VAN UART.
INT_ENABLE_REG is set with a 1-bit or an 8-bit manipulation instruction.
RESET input sets this register to 00H.

Figure 15-25: Interrupt enable register (INT_ENABLE_REG) Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
INT_ENA BLE_REG	GIE	RDA E	TBE E	FTE	FRE	LAE	EOM E	0	F820H	00H	R/W

Table 15-21:   Interrupt enable register (INT_ENABLE_REG) (1/2)

GIE: Global Interrupt Enable

GIE	Global Interrupt enable
0	Disables all the interrupt sources
1	Enables interrupt sources which can be disabled one by one with the following bits

RDAE: RDA Enable

RDAE	Receive interrupt
0	Receive interrupt disabled
1	Receive interrupt enabled

TBEE: TBE Enable

TBEE	Transmit interrupt
0	Transmit interrupt disabled
1	Transmit interrupt enabled

FTE: FT Enable

FTE	Fail transmit interrupt
0	Failed transmit interrupt disabled
1	Failed transmit interrupt enabled

Table 15-21:   Interrupt enable register (INT_ENABLE_REG) (2/2)

This interrupt will coincide with an EOM interrupt, as an error will cause a premature end of message.

FRE: FR Enable

FRE	Fail receive interrupt
0	Failed receive interrupt disabled
1	Failed receive interrupt enabled

This interrupt will coincide with an EOM interrupt, as an error will cause a premature end of message.

LAE: LA Enable

LAE	Lost arbitration interrupt
0	Lost arbitration interrupt disabled
1	Lost arbitration interrupt enabled

EOME: EOM Enable

EOME	End of Message interrupt
0	End of Message interrupt disabled
1	End of Message interrupt enabled

This interrupt occurs in the case of an end of message, i.e. after the acknowledge field or during an error (premature end of message).

15.4.14 VAN clock selection register (UDLCCL)

This SFR register enables the clock suppply to the VAN UART. UDLCCL is set with a 1-bit or an 8-bit manipulation instruction.
RESET input sets this register to 00H.

Figure 15-26: VAN clock selection register (UDLCCL) Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
UDLCCL	UDLCKEN	0	0	0	0	0	0	0	FF78H	00H	R/W

Table 15-22:   VAN clock selection register (UDLCCL)

UDLCKEN	VAN UDL clock control
0	Disable VAN clock supply
1	Enable VAN clock supply

Caution :   The VAN UART clock is disable at RESET. Application software must enable it in order to handle VAN communication.

15.5   VAN UART initialisation

1) Enable the clock via UDLCCL SFR register.

2) Configure the component:
   a) Choose the UART mode of operation owing to the RANK bit in the configuration register
   b) Enable or disable the In Frame Response using the IFR bit in the same register
   c) Enable or disable the generation of the IT12 interrupt using the IT12 bit in the same register
   d) Enable or disable the identifier filtering mechanism using the MSK1 and MSK0 bits in the same register
   e) Program the MSKx and ACx registers if filtering is enabled.

3) Program the prescaler to choose the network communication speed.

4) Enable the interrupts for the micro and the VAN UART.

[Memo]

Chapter 16 LCD Controller/Driver

16.1 LCD Controller/Driver Functions

The functions of the LCD controller/driver incorporated in the µPD1615A subseries are shown below.

(1) Automatic output of segment signals and common signals is possible by automatic writing of the display data memory.
(2) Any of five display modes can be selected.
   • Static
   • 1/2 duty (1/2 bias)
   • 1/3 duty (1/2 bias)
   • 1/3 duty (1/3 bias)
   • 1/4 duty (1/3 bias)
(3) Any of four frame frequencies can be selected in each display mode.
(4) Maximum of 40 segment signal outputs (S0 to S39); 4 common signal outputs (COM0 to COM3). The prt function register (PF) has to be set to LCD mode to allow the segment signal output. This LCD mode can be set bit-wise.

The maximum number of displayable pixels in each display mode is shown in Table 16-1.

Table 16-1: Maximum Number of Display Pixels

Bias Method	Time division	Common Signals Used	Maximum Number of Pixels
-	Static	COM0 (COM1, 2, 3)	40 (40 segments x 1 common)
1/2	2	COM0, COM1	80 (40 segments x 2 commons)
	3	COM0 - COM2	120 (40 segments x 3 commons)
1/3	3		
	4	COM0 - COM3	160 (40 segments x 4 commons)

16.2 LCD Controller/Driver Configuration

The LCD controller/driver is composed of the following hardware.

Table 16-2: LCD Controller/Driver Configuration

Item	Configuration
Display outputs	Segment signals : 40 Segment signal input/output port dual function : 40 Common signals : 4 (COM0 to COM3)
Control registers	LCD display mode register (LCDM) LCD display control register (LCDC)

Figure 16-1: LCD Controller/Driver Block Diagram
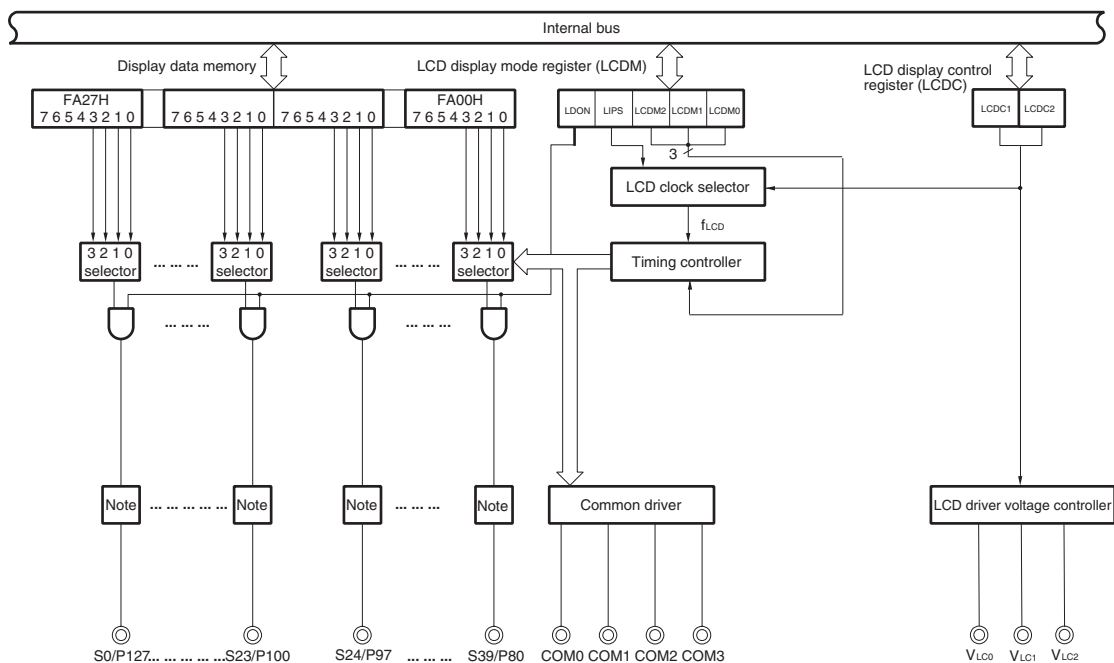
Note: Segment driver

Figure 16-2: LCD Clock Select Circuit Block Diagram
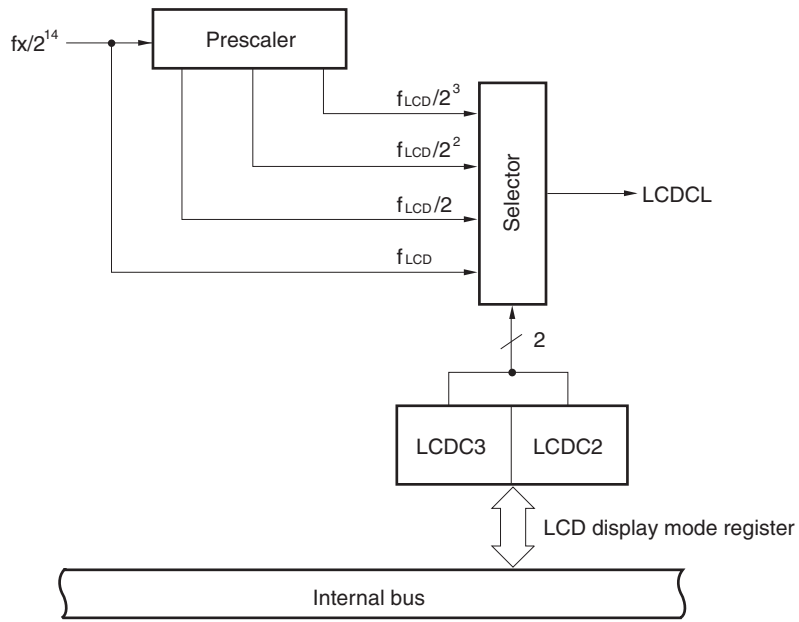
Remarks: 1. LCDCL : LCD clock
         2. f$_{LDC}$: LCD clock frequency

16.3 LCD Controller/Driver Control Registers

The LCD controller/driver is controlled by the following two registers.
• LCD display mode register (LCDM)
• LCD display control register (LCDC)

(1) LCD display mode register (LCDM)

This register sets display operation enabling/ disabling, the LCD driving power and the LCD display mode.
LCDM is set with a 1-bit or 8-bit memory manipulation instruction.
RESET input sets LCDM to 00H.

Figure 16-3: LCD Display Mode Register Format

Symbol	7	6	5	4	3	2	1	0	Address	AfterReset	R/W
LCDM	LCDON	0	0	LIPS	0	LCDM2	LCDM1	LCDM0	FFB0H	00H	R/W

LCDON	LCD Display Enable/Disable
0	Display off
1	Display on

LIPS	LCD driving power supply selection
0	Does not supply power to LCD
1	Supplies power to LCD from V$_{DD}$ pin

LCDM2	LCDM1	LCDM0	Selects display mode of LCD controller/driver	
			Time division	Bias mode
0	0	0	4	1/3
0	0	1	3	1/3
0	1	0	2	1/2
0	1	1	3	1/2
1	0	0	Static display mode	
Other than above			Setting prohibited	

Table 16-3: Frame Frequencies (Hz)

LCDC3	LCDC2	Frame frequency (Hz)							
		fx=4.0 MHz				fx=8.0 MHz			
		Static	1/2	1/3	1/4	Static	1/2	1/3	1/4
0	0	244	122	81.4	61	488	244	162.8	122
0	1	122	61	40.7	30.5	244	122	81.4	61
1	0	61	30.5	20.3	15.3	122	61	40.7	30.5
1	1	30.5	15.3	10.2	7.6	61	30.5	20.3	15.3

Remark: 1.Figures in parentheses apply to operation with fx = 4.0 MHz or fx = 8.0 MHz.

(2) LCD display clock control register (LCDC)

This register sets the LCD clock.
LCDC is set with a 1-bit or 8-bit memory manipulation instruction.
$\overline{\text{RESET}}$ input sets LCDC to 00H.

Figure 16-4: LCD Display Clock Control Register Format

Symbol	7	6	5	4	3	2	1	0	Address	AfterReset	R/W
LCDC	0	0	0	0	LCDC3	LCDC2	0	0	FFB2H	00H	R/W

LCDC3	LCDC2	Selection of LCD clock
0	0	$fx/2^{17}$
0	1	$fx/2^{16}$
1	0	$fx/2^{15}$
1	0	$fx/2^{14}$

16.4   LCD Controller/Driver Settings

   LCD controller/driver settings should be performed as shown below. When the LCD controller/driver is used, the watch timer should be set to the operational state beforehand.

<1> Set the initial value in the display data memory (FA00H to FA27H).

<2> Set the pins to be used as segment outputs in the port function registers (PF8 to PF12).

<3> Set the display mode, operating mode in the LCD display mode register (LCDM), and the LCD clock in the LCD clock control register (LCDC).

Next, set data in the display data memory according to the display contents.

16.5   LCD Display Data Memory

The LCD display data memory is mapped onto addresses FA00H to FA27H. The data stored in the LCD display data memory can be displayed on an LCD panel by the LCD controller/driver.
Figure 16-5 shows the relationship between the LCD display data memory contents and the segment outputs/common outputs.
Any area not used for display can be used as normal RAM.

*Figure 16-5:   Relationship between LCD Display Data Memory Contents
and Segment/Common Outputs*
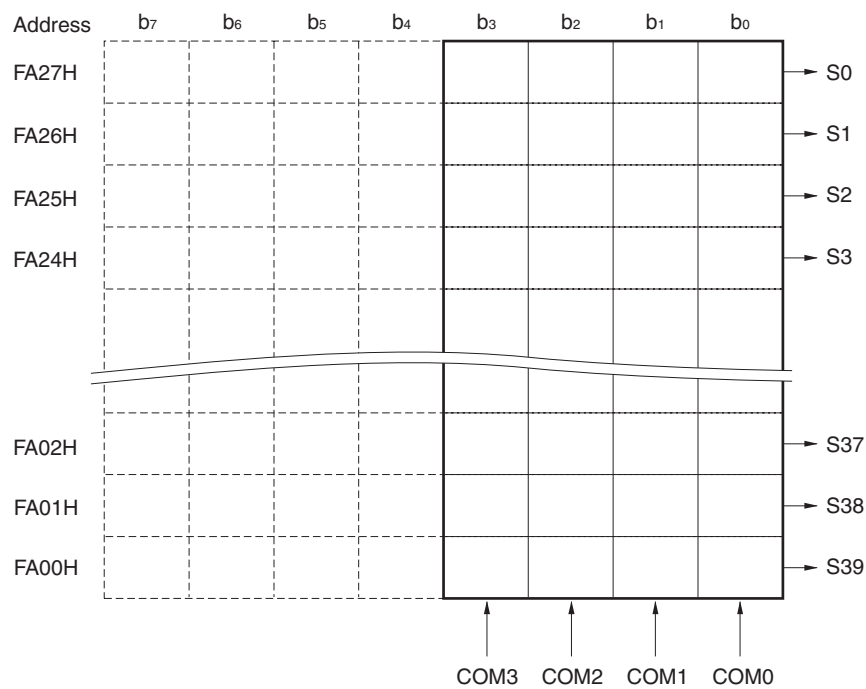
Caution: The higher 4 bits of the LCD display data memory do not incorporate memory. Be sure to set them to 0.

16.6   Common Signals and Segment Signals

An individual pixel on an LCD panel lights when the potential difference of the corresponding common signal and segment signal reaches or exceeds a given voltage (the LCD drive voltage V_{LCD}).
As an LCD panel deteriorates if a DC voltage is applied in the common signals and segment signals, it is driven by AC voltage.

(1) Common signals

For common signals, the selection timing order is as shown in Table 16-4 according to the number of time divisions set, and operations are repeated with these as the cycle. In the static display mode, the same signal is output to COM0 through COM3.
With 2-time-division operation, pins COM2 and COM3 are left open, and with 3-time-division operation, the COM3 pin is left open.

Table 16-4:   COM Signals

COM signal / Time division	COM0	COM1	COM2	COM3
Static	→	→	→	→
2-time division	→		Open	Open
3-time division	→			Open
4-time division	→			

(2) Segment signals

Segment signals correspond to a 40-byte LCD display data memory. Each display data memory bit 0, bit 1, bit 2, and bit 3 is read in synchronization with the COM0, COM1, COM2 and COM3 timings respectively, and if the value of the bit is 1, it is converted to the selection voltage. If the value of the bit is 0, it is converted to the non-selection voltage and output to a segment pin (S0 to S39).
Consequently, it is necessary to check what combination of front surface electrodes (corresponding to the segment signals) and rear surface electrodes (corresponding to the common signals) of the LCD display to be used form the display pattern, and then write bit data corresponding on a one-to-one basis with the pattern to be displayed.
In addition, because LCD display data memory bits 1 and 2 are not used with the static display mode, bits 2 and 3 are not used with the 2-time-division method, and bit 3 is not used with the 3-time-division method, these can be used for other than display purposes.
Bits 4 to 7 are fixed at 0.

(3) Common signal and segment signal output waveforms

The voltages shown in Table 16-5 are output in the common signals and segment signals. The ±V_{LCD} ON voltage is only produced when the common signal and segment signal are both at the selection voltage; other combinations produce the OFF voltage.

Table 16-5: LCD Drive Voltages

a) Static display mode

Common \ Segment		Select	Non-select
		V_{SS1}, V_{LC0}	V_{LC0}, V_{SS1}
V_{LC0}, V_{SS1}		-V_{LCD}, +V_{LCD}	0 V, 0 V

(b) 1/2 bias method

Common \ Segment		Select	Non-select
		V_{SS1}, V_{LC0}	V_{LC0}, V_{SS1}
Select level	V_{LC0}, V_{SS1}	-V_{LCD}, +V_{LCD}	0 V, 0 V
Non-select level	V_{LC1} = V_{LC2}	-1/2 V_{LCD}, +1/2 V_{LCD}	+1/2 V_{LCD}, -1/2 V_{LCD}

(c) 1/3 bias method

Common \ Segment		Select	Non-select
		V_{SS1}, V_{LC0}	V_{LC1}, V_{LC2}
Select level	V_{LC0}, V_{SS1}	-V_{LCD}, +V_{LCD}	-1/3 V_{LCD}, +1/3 V_{LCD}
Non-select level	V_{LC2}, V_{LC1}	-1/3 V_{LCD}, +1/3 V_{LCD}	-1/3 V_{LCD}, +1/3 V_{LCD}

Figure 16-6 shows the common signal waveform, and Figure 16-7 shows the common signal and segment signal voltages and phases.

Figure 16-6: Common Signal Waveform

(a) Static display mode
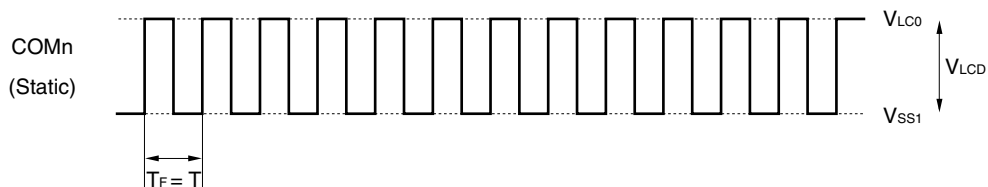
Remarks: 1.T: One LCDCL cycle
2.TF: Frame frequency

(b) 1/2 bias method
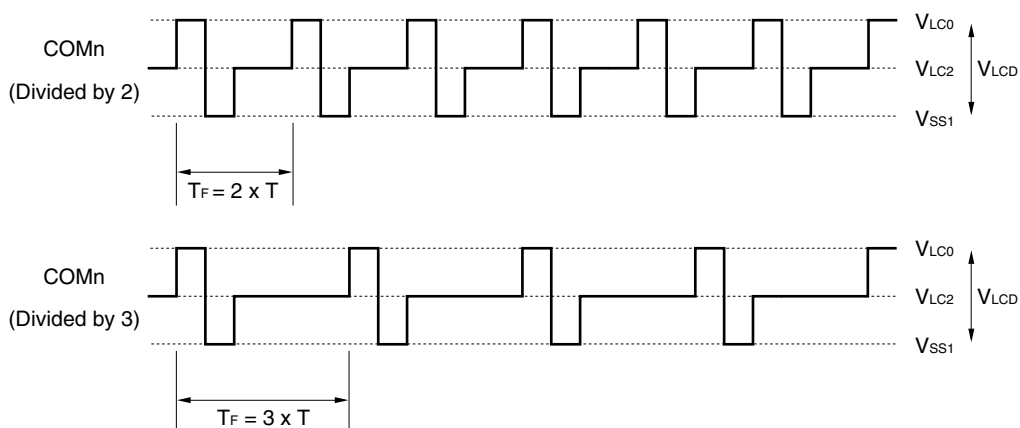
Remarks: 1.T: One LCDCL cycle
2.TF: Frame frequency

(c) 1/3 bias method
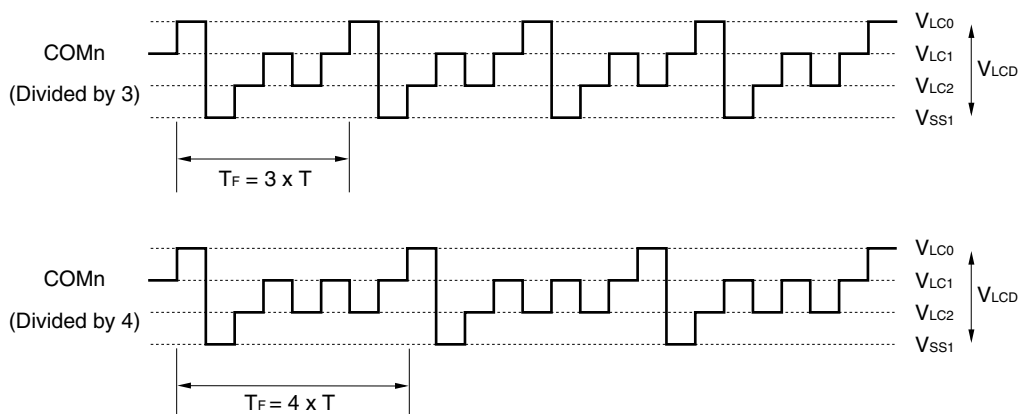
Remarks: 1.T: One LCDCL cycle
2.TF: Frame frequency

Figure 16-7: Common Signal and Static Signal Voltages and Phases

(a) Static display mode
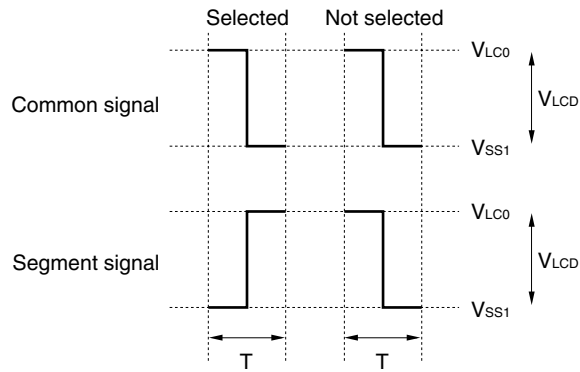
Remark: T : One LCDCL cycle

(b) 1/2 bias method
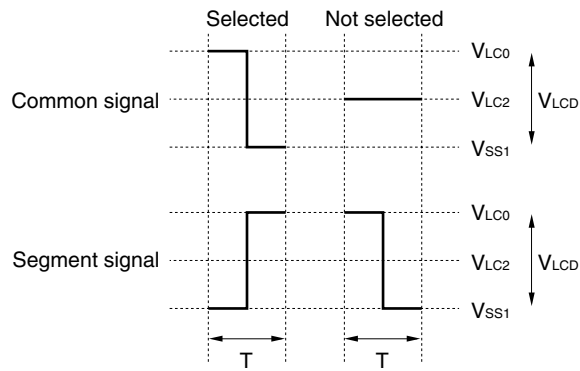
Remark: T : One LCDCL cycle

(c) 1/3 bias method
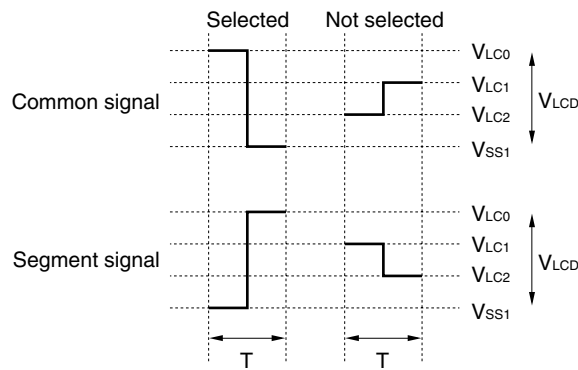
Remark: T : One LCDCL cycle

16.7   Supply of LCD Drive Voltages V_{LC0}, V_{LC1}, V_{LC2}

The split resistors makes it possible to produce LCD drive voltages appropriate to the various bias methods shown in Table 16-6 without using external split resistors.

Table 16-6: LCD Drive Voltages (with On-Chip Split Resistor)connected externally

Bias Method / LCD Drive Voltage	No bias (static mode)	1/2 bias	1/3 bias
V_{LC0}	V_{LCD}	V_{LCD}	V_{LCD}
V_{LC1}	2/3 V_{LCD}	1/2 V_{LCD}	2/3 V_{LCD}
V_{LC2}	1/3 V_{LCD}		1/3 V_{LCD}

An example of supply of the LCD drive voltage from off-chip is shown in Figure 16-9. Stepless LCD drive voltages can be supplied by means of variable resistor r.

Note: The 1615A Subseries has no split resistors inside. The split resistors have to be set externally for the different LCD voltages.

Figure 16-8: LCD Drive Power Supply Connection Examples (with External Split Resistor)

(a)   Static display mode [Note]
      (Example with V_{DD1} = 5 V, V_{LCD} = 5 V)
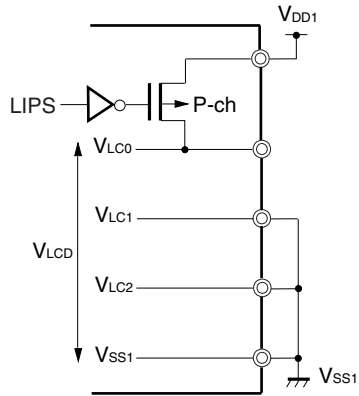
Note: LIPS should always be set to 1 (including in standby mode).

(b) 1/2 bias method
      (Example with VDD1 = 5 V, VLCD = 5 V)
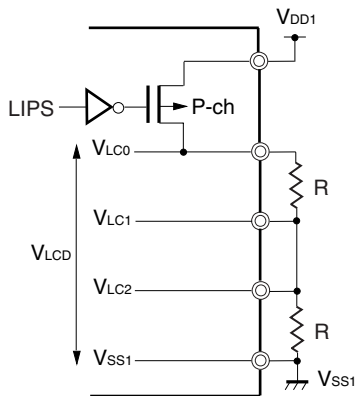
(c) 1/3 bias method
      (Example with VDD1 = 5 V, VLCD = 5 V)
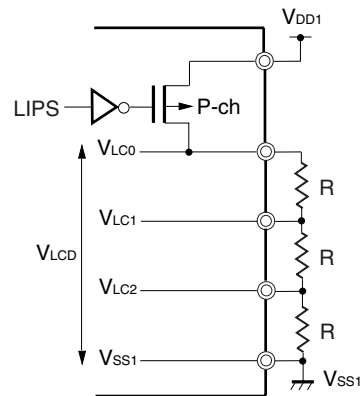
Caution: The LCD split resistors have to be set externally.

Figure 16-9: Example of LCD Drive Voltage Supply from Off-Chip

$$V_{LCD} = \frac{3R}{3R + r} V_{DD1}$$
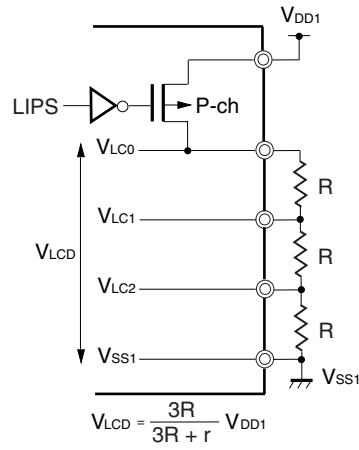
Caution: The LCD split resistors have to be set externally.

16.8 Display Modes

16.8.1 Static display example

Figure 16-11 shows the connection of a static type 5-digit LCD panel with the display pattern shown in Figure 16-10 with segment (S0 to S39) and common (COM0) signals. The display example is "123.45," and the display data memory contents (addresses FA68H to FA27H) correspond to this. An explanation is given here taking the example of the third digit "3." ($\exists$.). In accordance with the display pattern in Figure 16-10, selection and non-selection voltages must be output to pins S16 through S23 as shown in Table 16-7 at the COM0 common signal timing.

Table 16-7: Selection and Non-Selection Voltages (COM0)

Segment / Common	S16	S17	S18	S19	S20	S21	S22	S23
COM0	S	S	S	S	NS	S	NS	S

S: Selection, NS: Non-selection

From this, it can be seen that 10101111 must be prepared in the BIT0 bits of the display data memory corresponding to S16 to S23.
The LCD drive waveforms for S19, S20, and COM0 are shown in Figure 16-12. When S19 is at the selection voltage at the timing for selection with COM0, it can be seen that the $+V_{LCD}/-V_{LCD}$ AC square wave, which is the LCD illumination (ON) level, is generated.
Shorting the COM0 through COM3 lines increases the current drive capability because the same waveform as COM0 is output to COM1 through COM3.

Figure 16-10: Static LCD Display Pattern and Electrode Connections
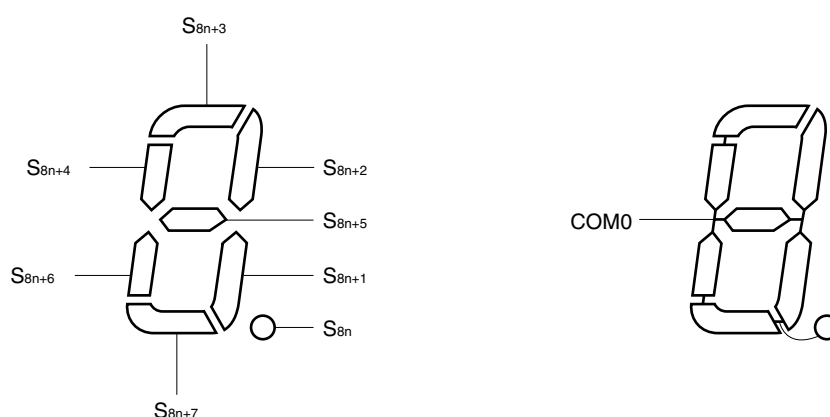
n = 0 to 4

Figure 16-11:   *Static LCD Panel Connection Example*
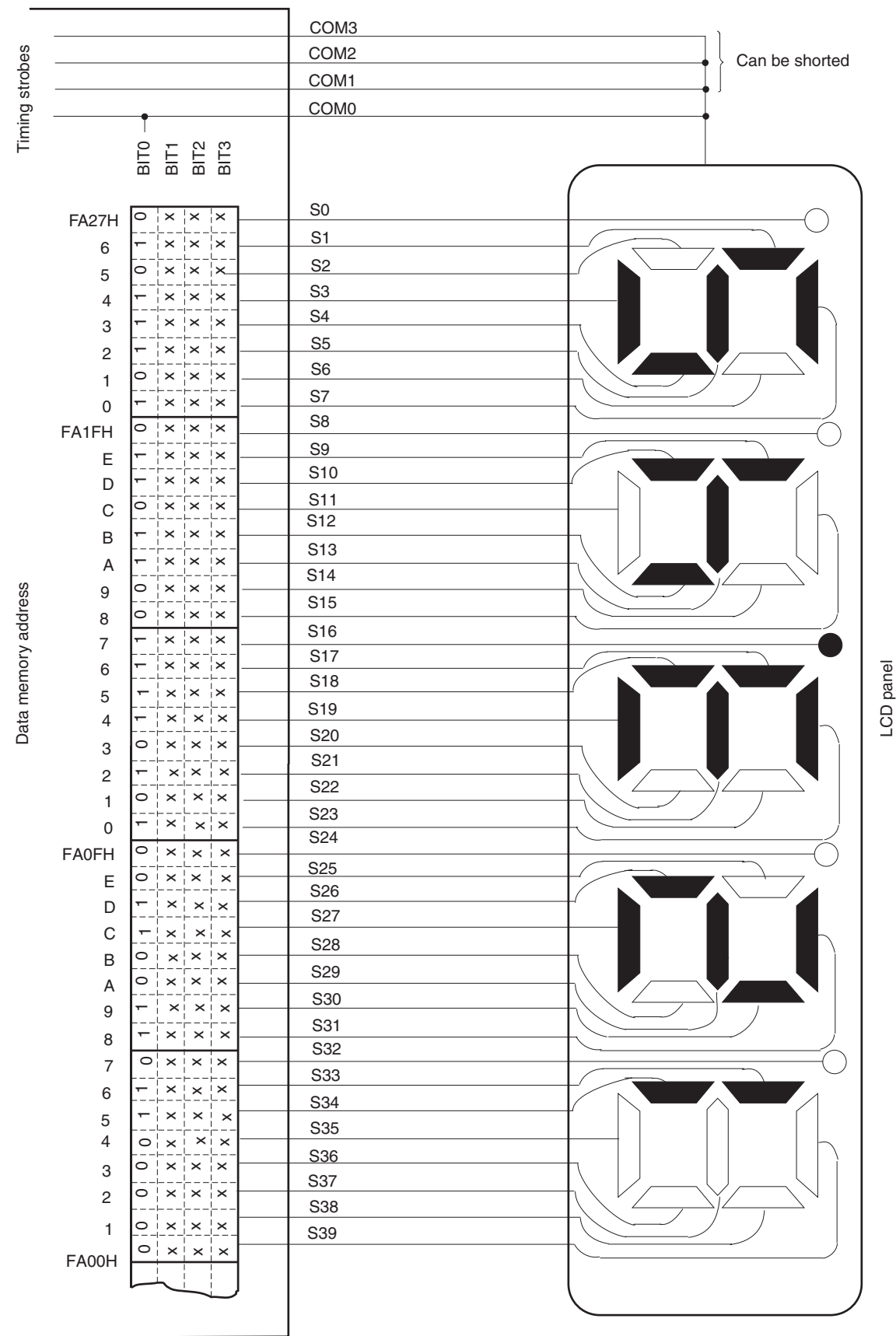
Figure 16-12: Static LCD Drive Waveform Examples
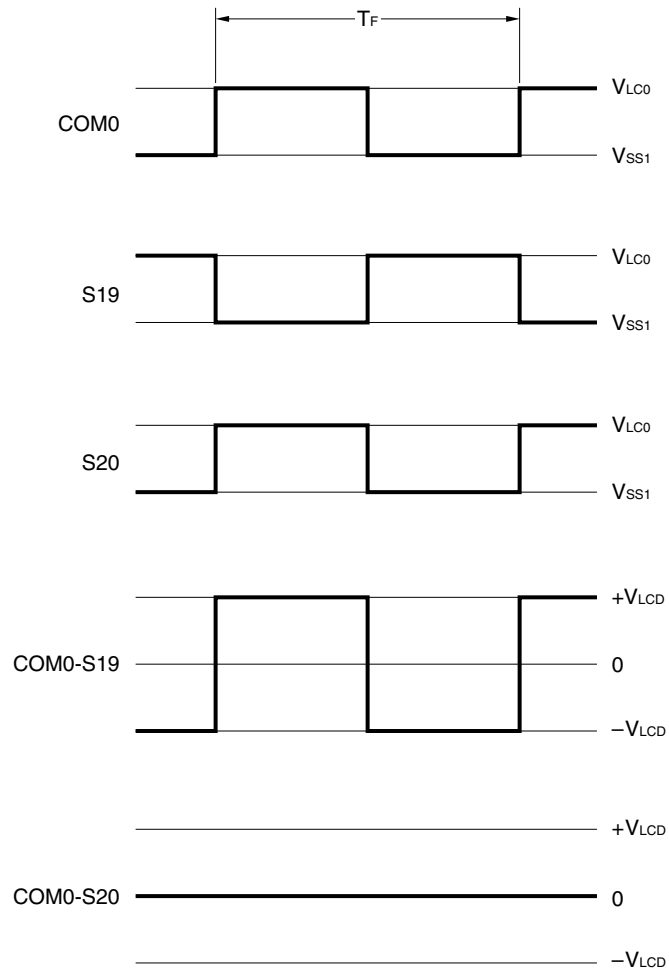
16.8.2   2-time-division display example

Figure 16-14 shows the connection of a 2-time-division type 10-digit LCD panel with the display pattern shown in Figure 16-13 with segment signals (S0 to S39) and common signals (COM0, COM1). The display example is "123456.7890," and the display data memory contents correspond to this.
An explanation is given here taking the example of the eighth digit "3" (ͺ). In accordance with the display pattern in Figure 16-13, selection and non-selection voltages must be output to pins S28 through S31 as shown in Table 16-8 at the COM0 and COM1 common signal timings.

Table 16-8: Selection and Non-Selection Voltages (COM0, COM1)

Segment Common	S28	S29	S30	S31
COM0	S	S	NS	NS
COM1	NS	S	S	S

S: Selection, NS: Non-selection

From this, it can be seen that, for example, xx10 must be prepared in the display data memory corresponding to S31.
Examples of the LCD drive waveforms between S31 and the common signals are shown in
Figure 16-15. When S31 is at the selection voltage at the COM1 selection timing, it can be seen that the +VLCD/–VLCD AC square wave, which is the LCD illumination (ON) level, is generated.

Figure 16-13:   2-Time-Division LCD Display Pattern and Electrode Connections
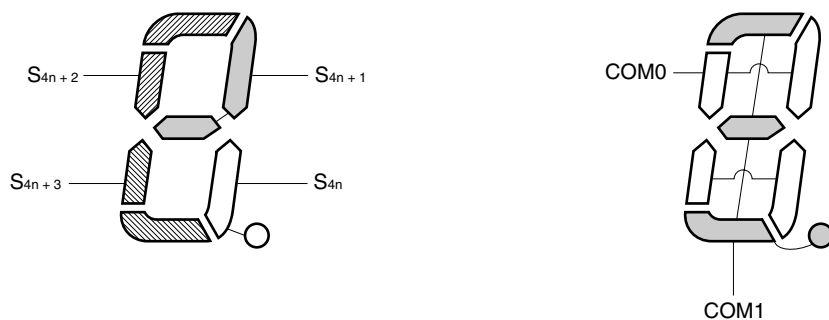
n = 0 to 9

Figure 16-14: 2-Time-Division LCD Panel Connection Example
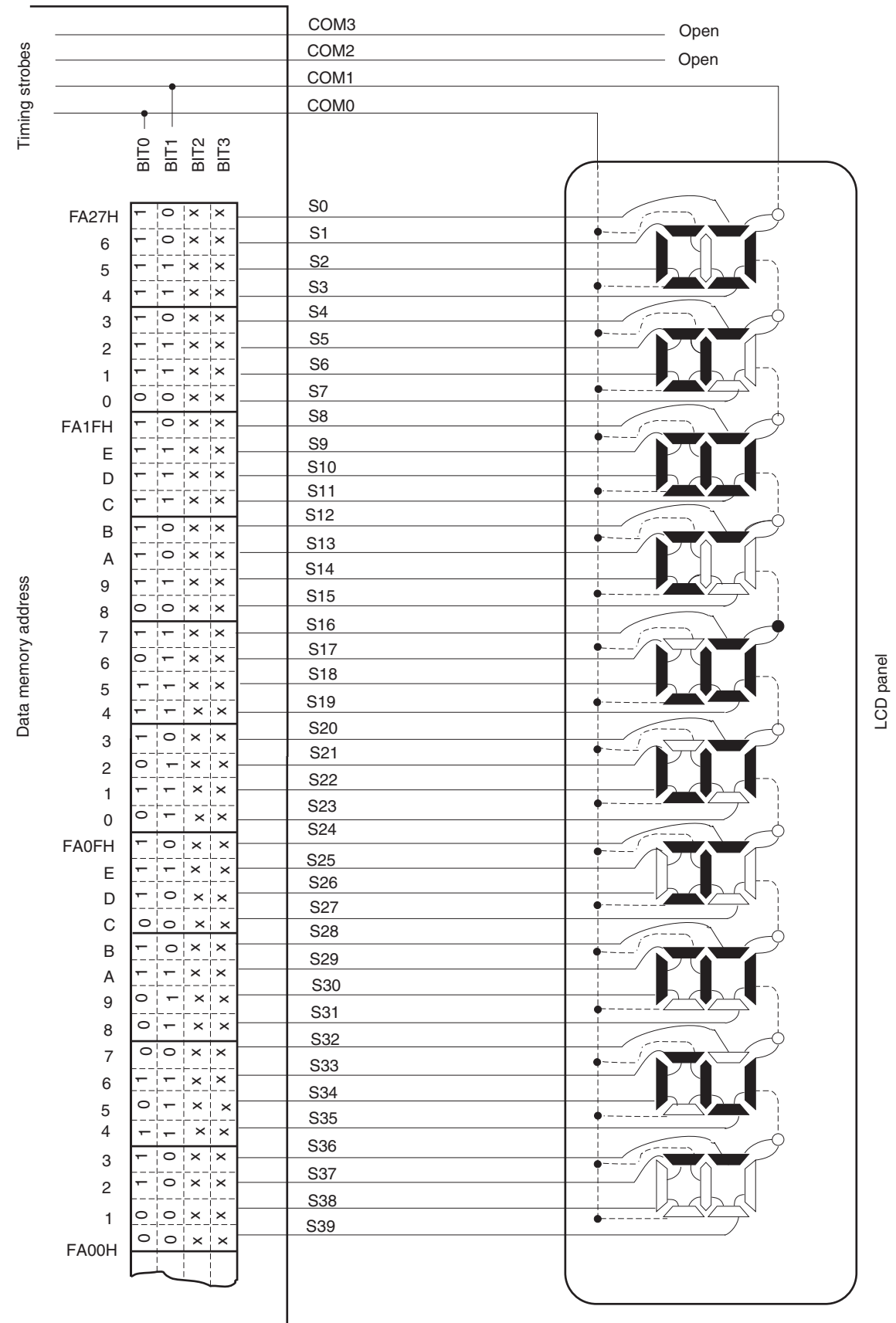
Remark: In bits marked X, any data can be stored because this is a 2-time-division display.

Figure 16-15: 2-Time-Division LCD Drive Waveform Examples (1/2 Bias Method)
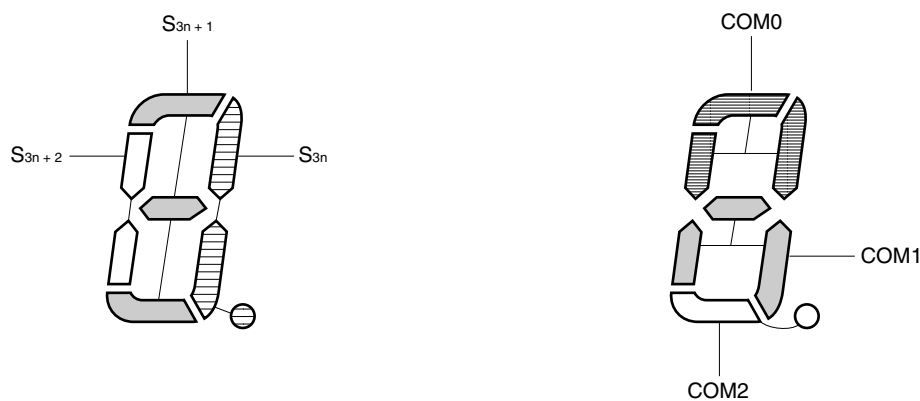
Figure 16-17: 3-Time-Division LCD Panel Connection Example

Timing strobes

COM3 — Open
COM2
COM1
COM0

BIT0 BIT1 BIT2 BIT3

Data memory address

Address	BIT0	BIT1	BIT2	BIT3	Segment
F A27H	1	1	0	x	S0
6	1	1	1	x	S1
5	0	0	x	x	S2
4	1	0	0	x	S3
3	0	0	0	x	S4
2	0	0	x	x	S5
1	1	1	0	x	S6
0	0	0	0	x	S7
FA1FH	0	0	x	x	S8
E	1	1	0	x	S9
D	1	0	1	x	S10
C	1	1	x	x	S11
B	1	1	0	x	S12
A	1	1	0	x	S13
9	1	0	x	x	S14
8	1	1	0	x	S15
7	1	1	1	x	S16
6	1	1	x	x	S17
5	1	1	0	x	S18
4	1	0	0	x	S19
3	1	0	x	x	S20
2	0	1	1	x	S21
1	1	1	1	x	S22
0	1	1	x	x	S23
FA0FH	0	1	0	x	S24
E	1	1	1	x	S25
D	1	0	x	x	S26
C	1	1	0	x	S27
B	0	1	0	x	S28
A	1	0	x	x	S29
9	1	1	0	x	S30
8	1	1	1	x	S31
7	0	0	x	x	S32
6	1	0	0	x	S33
5	1	1	1	x	S34
4	0	1	x	x	S35
3	1	1	0	x	S36
2	0	0	0	x	S37
1	0	0	x	x	S38
FA00H					

LCD panel

Remarks: 1. x' : Irrelevant bits because they have no corresponding segment in the LCD panel
         2. x : Irrelevant bits because this is a 3-time-division display

Figure 16-18: 3-Time-Division LCD Drive Waveform Examples (1/2 Bias Method)
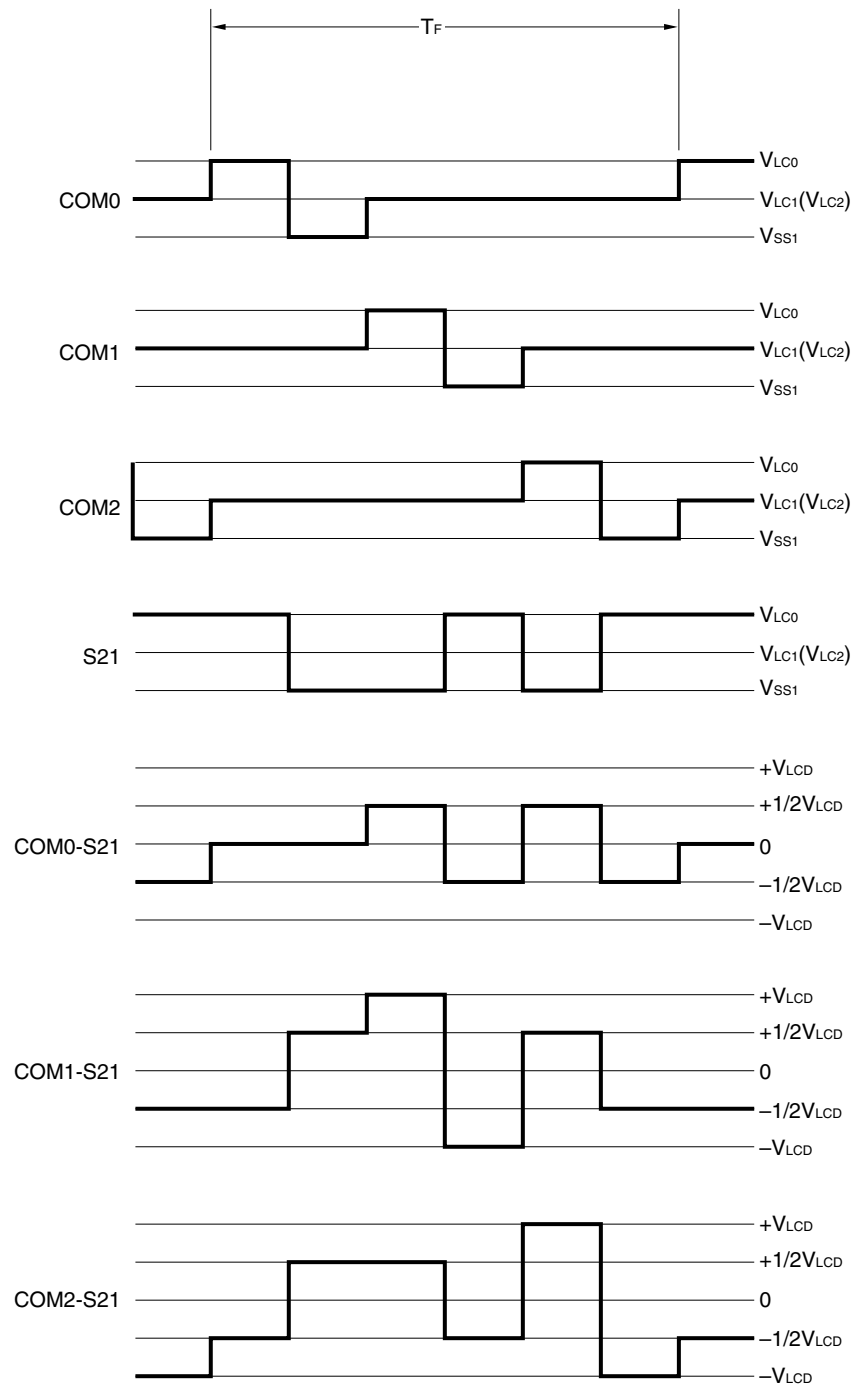
Figure 16-19: 3-Time-Division LCD Drive Waveform Examples (1/3 Bias Method)
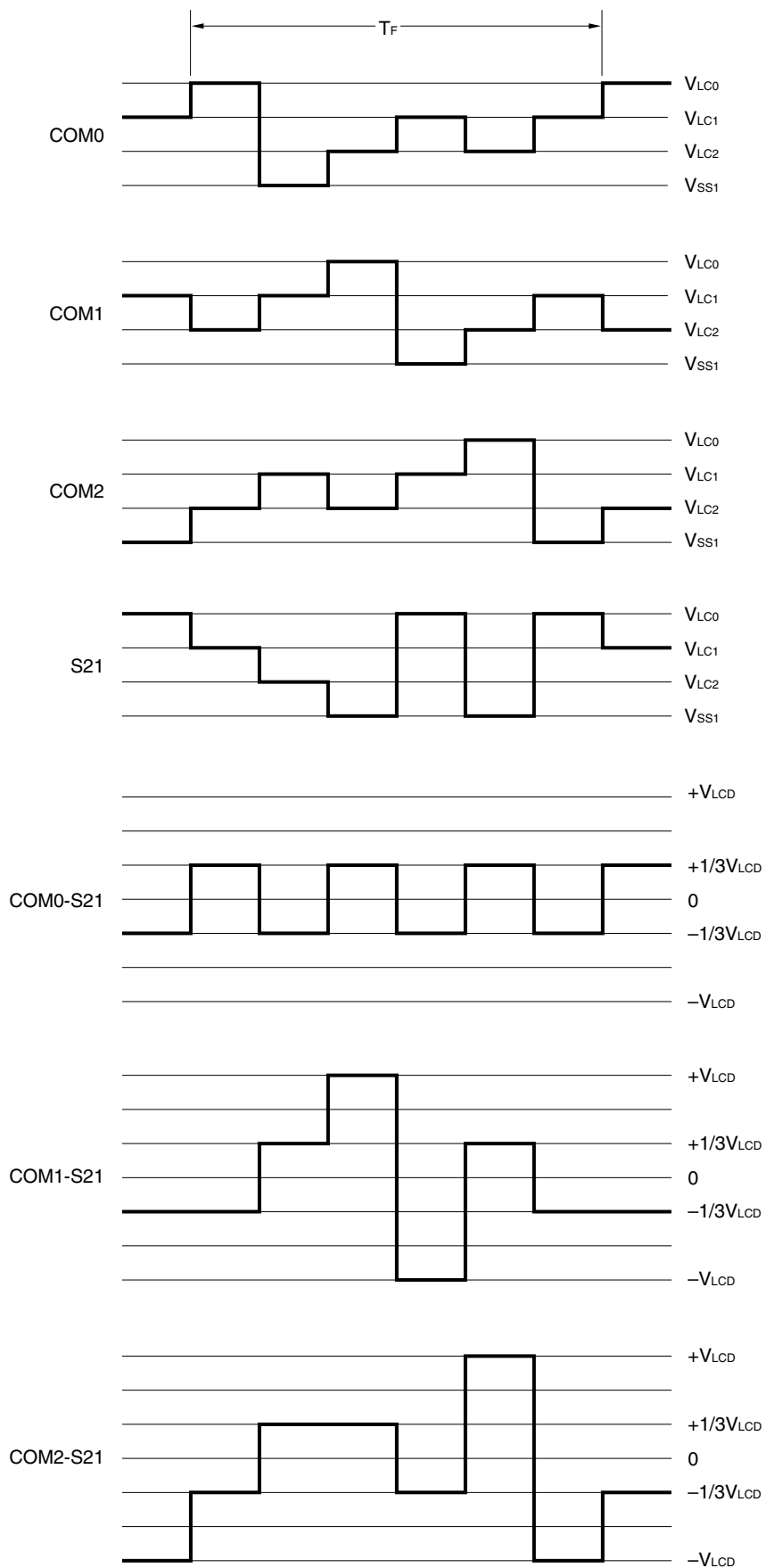
16.8.4 4-time-division display example

Figure 16-21 shows the connection of a 4-time-division type 20-digit LCD panel with the display pattern shown in Figure 16-20 with segment signals (S0 to S39) and common signals (COM0 to COM3). The display example is "123456.78901234567890," and the display data memory contents correspond to this.

An explanation is given here taking the example of the 15th digit "6." (⑥.). In accordance with the display pattern in Figure 16-20, selection and non-selection voltages must be output to pins S28 and S29 as shown in Table 16-10 at the COM0 to COM3 common signal timings.

Table 16-10: Selection and Non-Selection Voltages (COM0 to COM3)

| Common \ Segment	S28	S29
COM0	S	S
COM1	NS	S
COM2	S	S
COM3	S	S

S: Selection, NS: Non-selection

From this, it can be seen that 1101 must be prepared in the display data memory (address FA0BH) corresponding to S28.

Examples of the LCD drive waveforms between S28 and the COM0 and COM1 signals are shown in Figure 16-22 (for the sake of simplicity, waveforms for COM2 and COM3 have been omitted). When S28 is at the selection voltage at the COM0 selection timing, it can be seen that the $+V_{LCD}/-V_{LCD}$ AC square wave, which is the LCD illumination (ON) level, is generated.

Figure 16-20: 4-Time-Division LCD Display Pattern and Electrode Connections
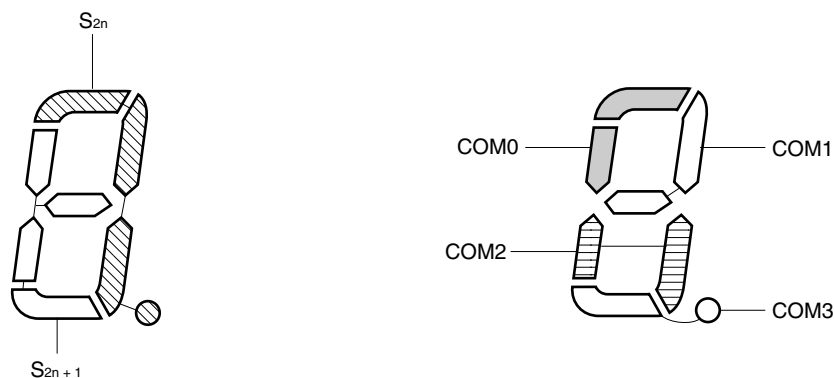
n = 0 to 18

Figure 16-21: 4-Time-Division LCD Panel Connection Example
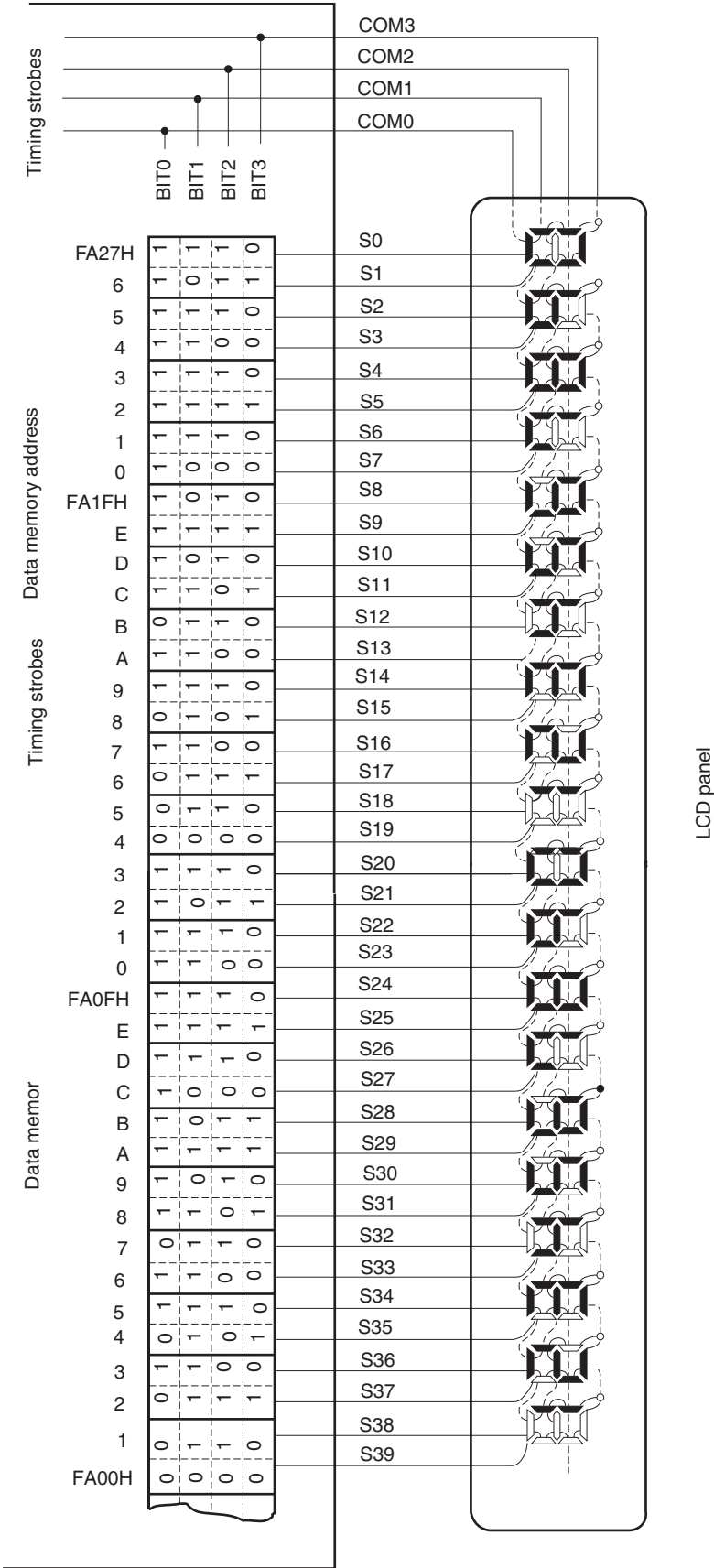
Figure 16-22: 4-Time-Division LCD Drive Waveform Examples (1/3 Bias Method)
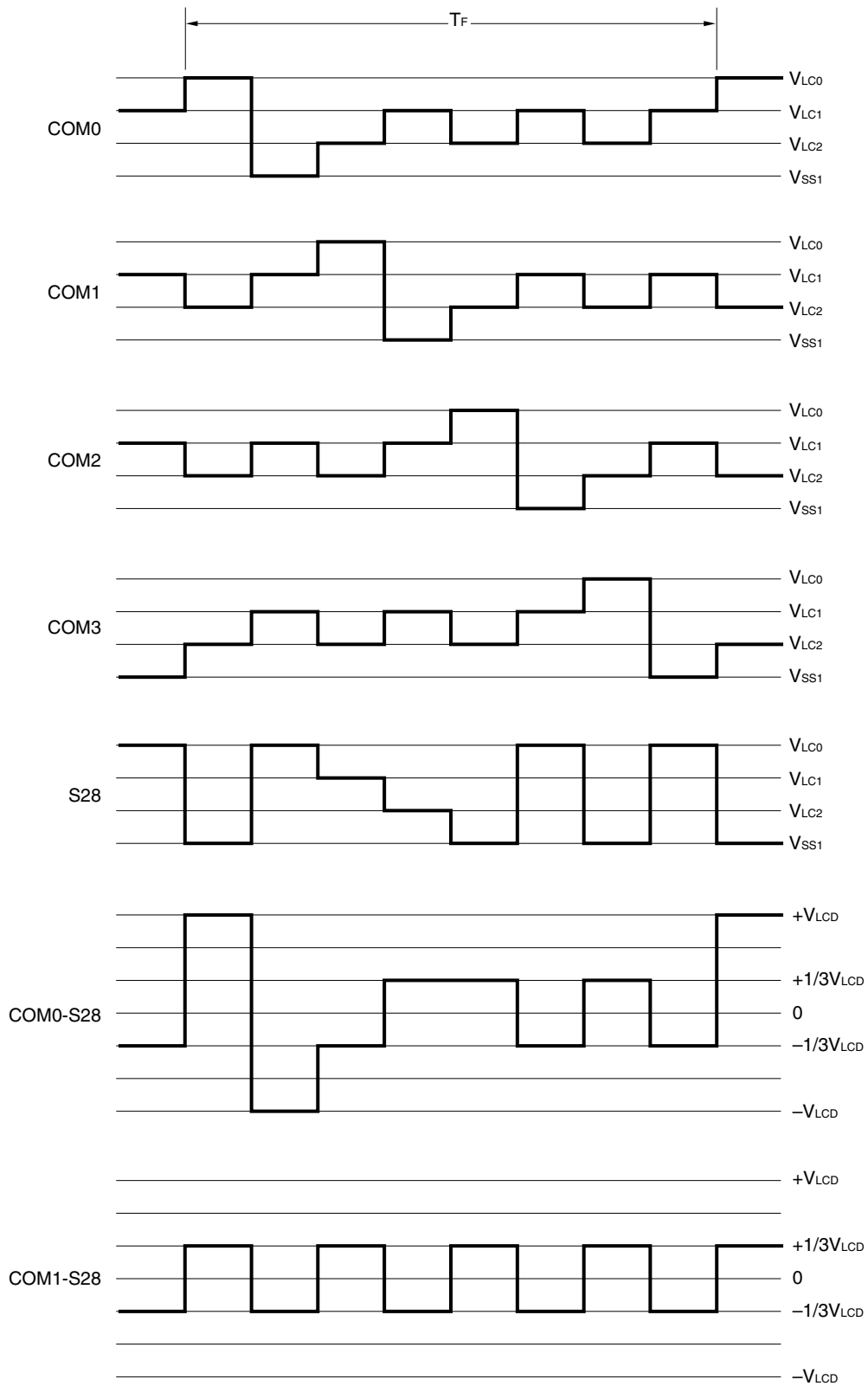
[Memo]

Chapter 17 Sound Generator

17.1 Sound Generator Function

The sound generator has the function to sound the buzzer from an external speaker, and the following two signals are output.

(1) Basic cycle output signal (with/without amplitude)

A buzzer signal with a variable frequency in a range of 0.25 to 7.3 kHz (at fx = 8.00 MHz) can be output. The amplitude of the basic cycle output signal can be varied by ANDing the basic cycle output signal with the 7-bit-resolution PWM signal, to enable control of the buzzer sound volume.

(2) Amplitude output signal

A PWM signal with a 7-bit resolution for variable amplitude can be independently output.

Figure 17-1 shows the sound generator block diagram and Figure17-2 shows the concept of each signal.

Figure 17-1: Sound Generator Block Diagram
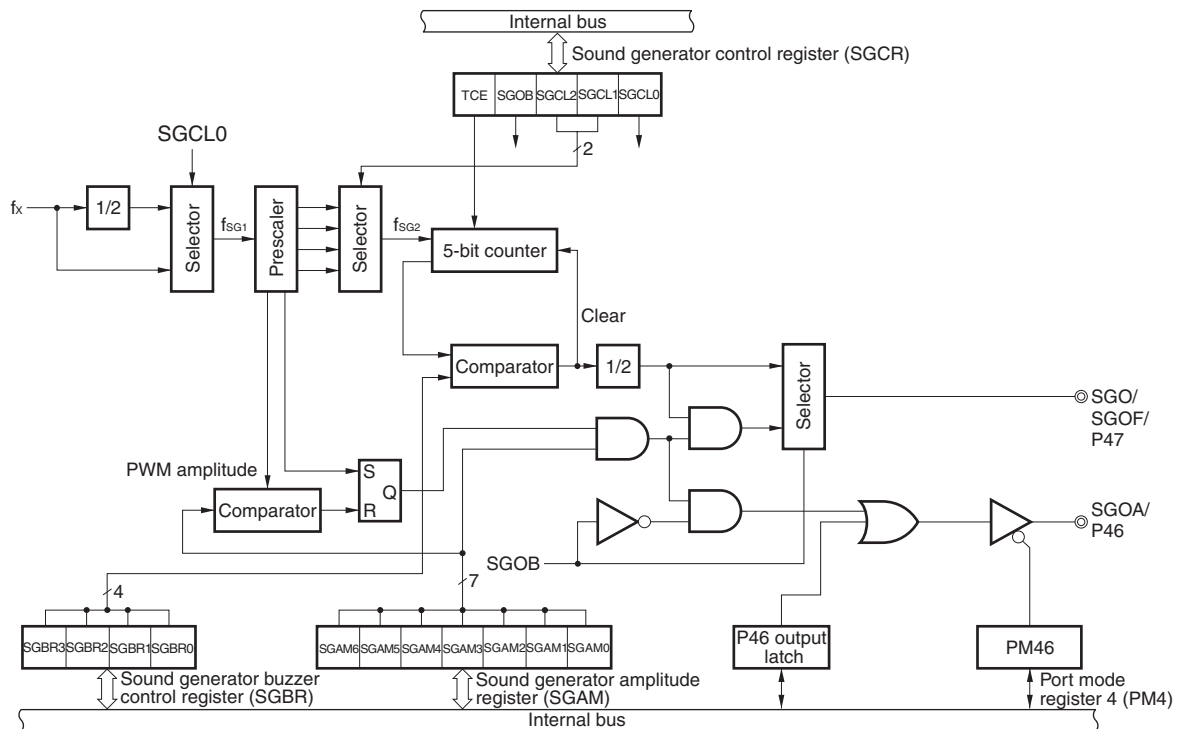
Figure 17-2: Concept of Each Signal
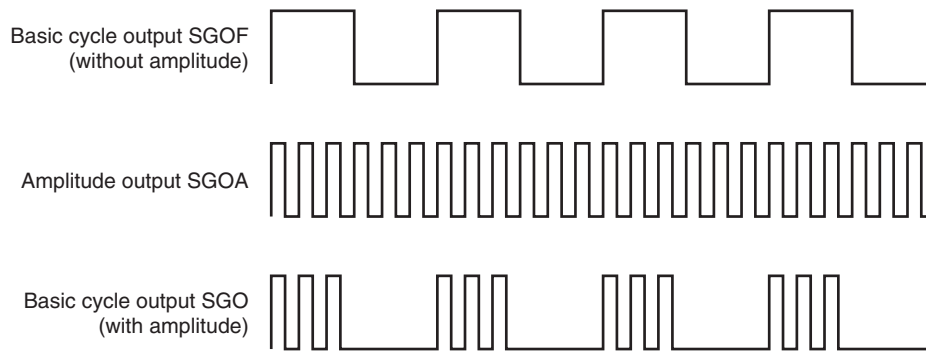
17.2 **Sound Generator Configuration**

The sound generator consists of the following hardware.

Table 17-1: Sound Generator Configuration

Item	Configuration
Counter	8 bits x 1, 5 bits x 1
SG output	SGO/SGOF (with/without append bit of basic cycle output) SGOA (amplitude output)
Control register	Sound generator control register (SGCR) Sound generator buzzer control register (SGBR) Sound generator amplitude register (SGAM)

17.3 Sound Generator Control Registers

The following three types of registers are used to control the sound generator.

- Sound generator control register (SGCR)
- Sound generator buzzer control register (SGBR)
- Sound generator amplitude control register (SGAM)

(1)Sound generator control register (SGCR)
SGCR is a register which sets up the following four types.

- Controls sound generator output
- Selects output of sound generator
- Selects sound generator input frequency f_{SG1}
- Selects 5-bit counter input frequency f_{SG2}

SGCR is set with a 1-bit or 8-bit memory manipulation instruction.
$\overline{\text{RESET}}$ input clears SGCR to 00H.
Figure 17-3 shows the SGCR format.

Figure 17-3: Sound Generator Control Register (SGCR) Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
SGCR	TCE	0	0	0	SGOB	SGCL2	SGCL1	SGCL0	FF66H	00H	R/W

TCE	Sound Generator Output Selection
0	Timer operation stopped SGOF/SGO and SGOA for low-level output
1	Sound generator operation SGOF/SGO and SGOA for output

Caution: Before setting the TCE bit, set all the other bits.

Remark: SGOF: Basic cycle signal (without amplitude)
SGO: Basic cycle signal (with amplitude)
SGOA: Amplitude signal

SGOB	Sound Generator Output Selection
0	Selects SGOF and SGOA outputs
1	Selects SGO and PCL outputs

SGCL2	SGCL1	5-Bit Counter Input Frequency f_{SG2} Selection
0	0	$f_{SG2} = f_{SG1}/2^5$
0	1	$f_{SG2} = f_{SG1}/2^6$
1	0	$f_{SG2} = f_{SG1}/2^7$
1	1	$f_{SG2} = f_{SG1}/2^8$

SGCL0	Sound Generator Input Frequency Selection
0	$f_{SG1} = f_x/2^7$
1	$f_{SG1} = f_x/2^8$

Cautions: 1. When rewriting SGCR to other data, stop the timer operation (TCE = 0) beforehand.
2. Bits 4 to 6 must be set to 0.

The sound generator output frequency f$_{SG}$ can be calculated by the following expression.

$$f_{SG} = 2^{(SGCL0 - SGCL1 - 2 \times SGCL2 - 7)} \times \{fx/(SGBR + 17)\}$$

Substitute set 0 or 1 to SGCL0 to SGCL2 in the above expression. Substitute a decimal value to SGBR. Where fx = 8 MHz, SGCL0 to SGCL2 is (1, 0, 0), and SGBR0 to SGBR3 is (1, 1, 1, 1), SGBR = 15. Therefore,

$$f_{SG} = 2^{(1 - 0 - 2 \times 0 - 7)} \times \{fx/(15 + 17)\}$$
$$= 3.906 \text{ kHz}$$

(2) Sound generator buzzer control register (SGBR)

SGBR is a register that sets the basic frequency of the sound generator output signal.
SGBR is set with a 1-bit or 8-bit memory manipulation instruction.
$\overline{\text{RESET}}$ input clears SGBR to 00H.
Figure 17-4 shows the SGBR format.

Figure 17-4: Sound Generator Buzzer Control Register (SGBR) Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
SGBR	0	0	0	0	SGBR3	SGBR2	SGBR1	SGBR0	FF68H	00H	R/W

Cautions: 1. **When rewriting SGBR to other data, stop the timer operation (TCE = 0) beforehand.**
2. **Bits 4 to 7 must be set to 0.**

Figure 17-5: Sound Generator Frequency Selection

SGCL0	SGBR				SGCL2,1 (Hz)			
	4-bit comparator				00	01	10	11
0	0	0	0	0	7352.9	3676.5	1838.2	919.1
	0	0	0	1	6944.4	3472.2	1736.1	868.1
	0	0	1	0	6578.9	3289.5	1644.7	822.4
	0	0	1	1	6250.0	3125.0	1562.5	781.3
	0	1	0	0	5952.4	2976.2	1488.1	744.0
	0	1	0	1	5681.8	2840.9	1420.5	710.2
	0	1	1	0	5434.8	2717.4	1358.7	679.3
	0	1	1	1	5208.3	2604.2	1302.1	651.0
	1	0	0	0	5000.0	2500.0	1250.0	625.0
	1	0	0	1	4807.7	2403.8	1201.9	601.0
	1	0	1	0	4629.6	2314.8	1157.4	578.7
	1	0	1	1	4464.3	2232.1	1116.1	558.0
	1	1	0	0	4310.3	2155.2	1077.6	538.8
	1	1	0	1	4166.7	2083.3	1041.7	520.8
	1	1	1	0	4032.3	2016.1	1008.1	504.0
	1	1	1	1	3906.3	1953.1	976.6	488.3
1	0	0	0	0	3676.5	1838.2	919.1	459.6
	0	0	0	1	3472.2	1736.1	868.1	434.0
	0	0	1	0	3289.5	1644.7	822.4	411.2
	0	0	1	1	3125.0	1562.5	781.3	390.6
	0	1	0	0	2976.2	1488.1	744.0	372.0
	0	1	0	1	2840.9	1420.5	710.2	355.1
	0	1	1	0	2717.4	1358.7	679.3	339.7
	0	1	1	1	2604.2	1302.1	651.0	325.5
	1	0	0	0	2500.0	1250.0	625.0	312.5
	1	0	0	1	2403.8	1201.9	601.0	300.5
	1	0	1	0	2314.8	1157.4	578.7	289.4
	1	0	1	1	2232.1	1116.1	558.0	279.0
	1	1	0	0	2155.2	1077.6	538.8	269.4
	1	1	0	1	2083.3	1041.7	520.8	260.4
	1	1	1	0	2016.1	1008.1	504.0	252.0
	1	1	1	1	1953.1	976.6	488.3	244.1

(3) Sound generator amplitude register (SGAM)

SGAM is a register that sets the amplitude of the sound generator output signal.

SGAM is set with a 1-bit or 8-bit memory manipulation instruction.

RESET input clears SGAM to 00H.

Figure 17-6 shows the SGAM format.

Figure 17-6: Sound Generator Amplitude Register (SGAM) Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
SGAM	0	SGAM6	SGAM5	SGAM4	SGAM3	SGAM2	SGAM1	SGAM0	FF67H	00H	R/W

SGAM6	SGAM5	SGAM4	SGAM3	SGAM2	SGAM1	SGAM0	Amplitude
0	0	0	0	0	0	0	0/128
0	0	0	0	0	0	1	2/128
0	0	0	0	0	1	0	3/128
0	0	0	0	0	1	1	4/128
0	0	0	0	1	0	0	5/128
0	0	0	0	1	0	1	6/128
0	0	0	0	1	1	0	7/128
0	0	0	0	1	1	1	8/128
0	0	0	1	0	0	0	9/128
0	0	0	1	0	0	1	10/128
0	0	0	1	0	1	0	11/128
0	0	0	1	0	1	1	12/128
0	0	0	1	1	0	0	13/128
0	0	0	1	1	0	1	14/128
0	0	0	1	1	1	0	15/128
0	0	0	1	1	1	1	16/128
0	0	1	0	0	0	0	17/128
0	0	1	0	0	0	1	18/128
0	0	1	0	0	1	0	19/128
0	0	1	0	0	1	1	20/128
0	0	1	0	1	0	0	21/128
0	0	1	0	1	0	1	22/128
0	0	1	0	1	1	0	23/128
0	0	1	0	1	1	1	24/128
0	0	1	1	0	0	0	25/128
0	0	1	1	0	0	1	26/128
0	0	1	1	0	1	0	27/128
0	0	1	1	0	1	1	28/128
0	0	1	1	1	0	0	29/128
0	0	1	1	1	0	1	30/128
0	0	1	1	1	1	0	31/128
			f				*f*
0	1	1	1	1	1	1	128/128

Cautions: 1. **When rewriting the contents of SGAM, the timer operation does not need to be stopped. However, note that a high level may be output for one period due to rewrite timing.**

2. **Bit 7 must be set to 0.**

303

17.4 Sound Generator Operations

17.4.1 To output basic cycle signal SGOF (without amplitude)

Select SGOF output by setting bit 3 (SGOB) of the sound generator control register (SGCR) to "0".
The basic cycle signal with a frequency specified by the SGCL0 to SGCL2 and SGBR0 to SGBR3 is output.
At the same time, the amplitude signal with an amplitude specified by the SGAM0 to SGAM6 is output from the SGOA pin.

Figure 17-7: Sound Generator Output Operation Timing without Amplitude
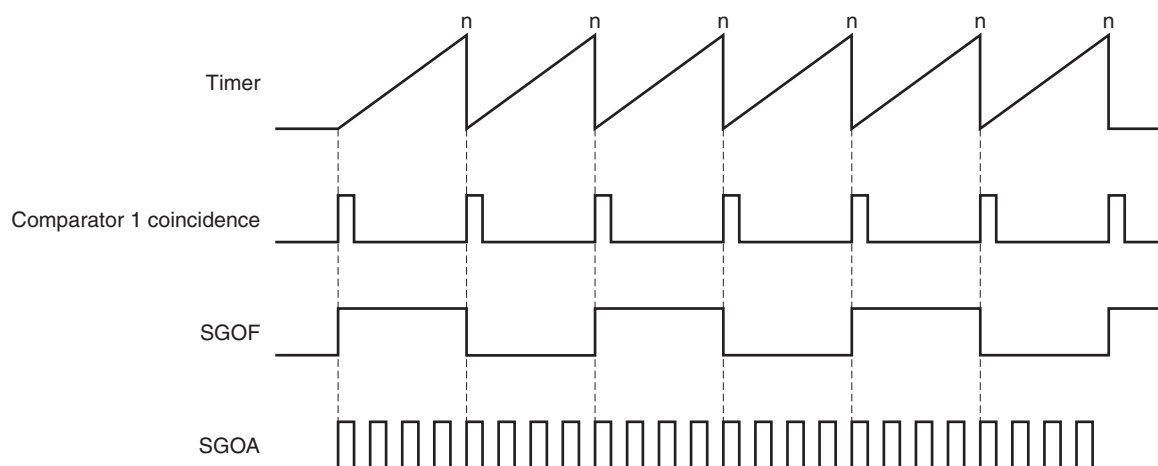
17.4.2 To output basic cycle signal SGO (with amplitude)

Select SGO output by setting bit 3 (SGOB) of the sound generator control register (SGCR) to "1".
The basic cycle signal with a frequency specified by the SGCL0 to SGCL2 and SGBR0 to SGBR3 is output.
When SGO output is selected, the SGOA pin can be used as a PCL output (clock output) or I/O port pin.

Figure 17-8: Sound Generator Output Operation Timing with Amplitude
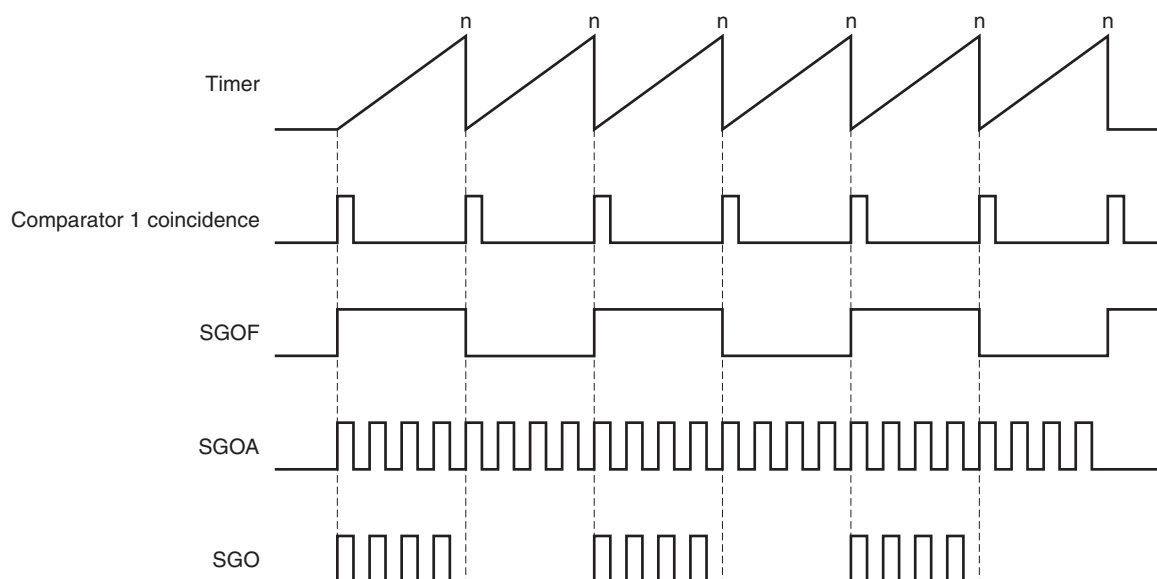
[Memo]

Chapter 18 Interrupt Functions

18.1 Interrupt Function Types

The following three types of interrupt functions are used.

(1) Non-maskable interrupt

This interrupt is acknowledged unconditionally even in a disabled state. It does not undergo interrupt priority control and is given top priority over all other interrupt requests.

It generates a standby release signal.

The non-maskable interrupt has one source of interrupt request from the watchdog timer.

(2) Maskable interrupts

These interrupts undergo mask control. Maskable interrupts can be divided into a high interrupt priority group and a low interrupt priority group by setting the priority specify flag register (PR0L, PR0H, and PR1L).

Multiple high priority interrupts can be applied to low priority interrupts. If two or more interrupts with the same priority are simultaneously generated, each interrupts has a predetermined priority (see Table 18-1).

A standby release signal is generated.

The maskable interrupt has seven sources of external interrupt requests and fifteen sources of internal interrupt requests.

(3) Software interrupt

This is a vectored interrupt to be generated by executing the BRK instruction. It is acknowledged even in a disabled state. The software interrupt does not undergo interrupt priority control.

18.2 Interrupt Sources and Configuration

There are total of 24 non-maskable, maskable, and software interrupts in the interrupt sources.

Table 18-1: Interrupt Source List

Interrupt type	Priority (default)	Interrupt request source		Vector code address	Basic structure type
Resetting	-	RESET	Reset input	0000H	
Non-maskable	-	INTWDT	Watchdog timer overflow (when non-maskable interrupt is selected)	0004H	(A)
Maskable	0	INTWDT	Watchdog timer overflow (when interval timer is selected)		(B)
	1	INTVE	INTVE → VAN-End of Message	0006H	
	2	INTVT	INTVT → VAN-Emission	0008H	
	3	INTVR	INTVR → VAN-Reception	000AH	
	4	INTP0	External interrupt pin input edge detection	000CH	(C)
	5	INTP1		000EH	
	6	INTP2		0010H	
	7	INTTM00	Agreement between TM00 and CR00 (when compare register is specified) TI01 valid edge detection (when capture register is specified)	0012H	(B)
	8	INTTM01	Agreement between TM00 and CR01 (when compare register is specified) TI00 valid edge detection (when capture register is specified)	0014H	
	9	INTTM50	Agreement between TM50 and CR50	0016H	
	10	INTTM51	Agreement between TM51 and CR51	0018H	
	11	INTWTI	Watch timer interval interrupt	001AH	
	12	INTWT	Watch interrupt	001CH	
	13	INTCSI3	SIO30 transfer completion	001EH	
	14	INTSER	UART0 reception error occurrence	0020H	
	15	INTSR	UART0 reception completion	0022H	
	16	INTST	UART0 transmission completion	0024H	
	17	INTAD	A/D conversion end	0026H	
Software	-	BRK	Execution of BRK instruction	003EH	(D)

Notes: 1. Default priorities are intended for two or more simultaneously generated maskable interrupt requests. 0 is the highest priority and 26 is the lowest priority.
2. Basic configuration types (A) to (D) correspond to (A) to (D) of Figure 18-1.

307

Figure 18-1: Basic Configuration of Interrupt Function (1/2)

(A) Internal non-maskable interrupt
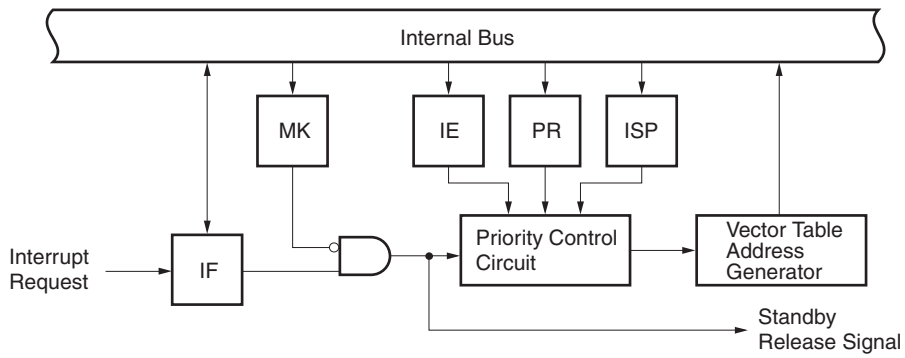
```
                                              Internal Bus
                                                   ↑
                                                   │
                          ┌──────────────┐   ┌──────────────┐
Interrupt                 │Priority      │   │Vector Table  │
Request   ────────●──────▶│Control       │──▶│Address       │
                  │       │Circuit       │   │Generator     │
                  │       └──────────────┘   └──────────────┘
                  │
                  │                              Standby
                  └────────────────────────────▶Release Signal
```

(B) Internal maskable interrupt

```
                         Internal Bus
      ↑↓        ↓        ↓      ↓      ↓                    ↑
      │      ┌─────┐  ┌─────┐┌─────┐┌─────┐                │
      │      │ MK  │  │ IE  ││ PR  ││ ISP │                │
      │      └─────┘  └─────┘└─────┘└─────┘                │
      │         │        │      │      │                   │
   ┌─────┐      │        └──────┴──────┘                   │
   │     │      │     ┌──────────────┐   ┌──────────────┐  │
Interrupt│ IF  │──○───│Priority      │──▶│Vector Table  │──┘
Request──▶│    │  )───│Control       │   │Address       │
   └─────┘      │     │Circuit       │   │Generator     │
                ●─────└──────────────┘   └──────────────┘
                │                           Standby
                └──────────────────────────▶Release Signal
```

        IF   :   Interrupt request flag
        IE   :   Interrupt enable flag
        ISP :   Inservice priority flag
        MK :   Interrupt mask flag
        PR   :   Priority specify flag

Figure 18-1: Basic Configuration of Interrupt Function (2/2)

(C) External maskable interrupt (except INTP0)
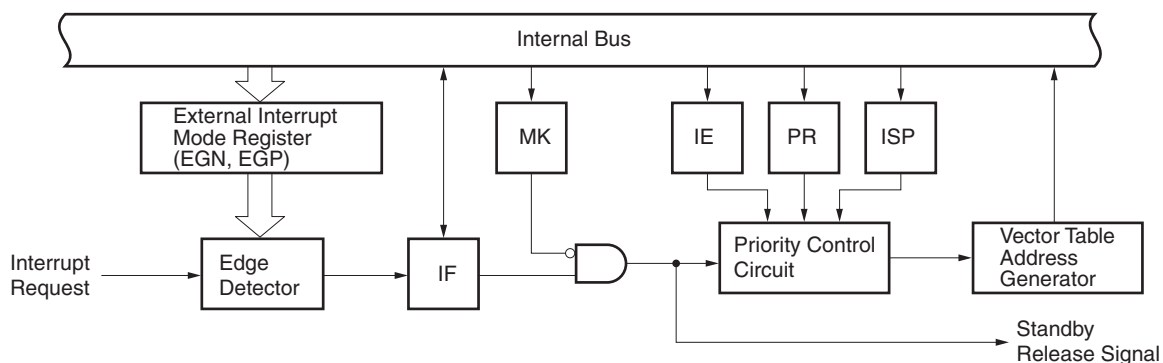
(D) Software interrupt
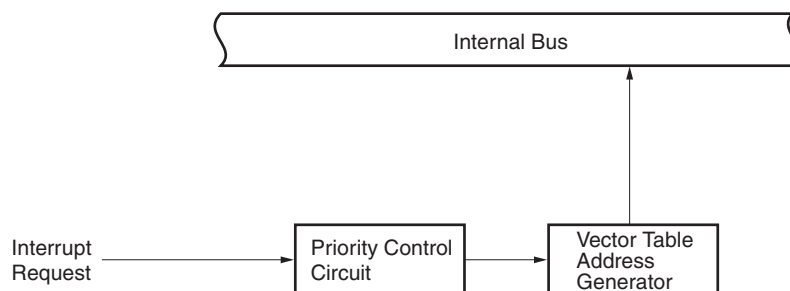
IF : Interrupt request flag
IE : Interrupt enable flag
ISP : Inservice priority flag
MK : Interrupt mask flag
PR : Priority specify flag

18.3 Interrupt Function Control Registers

The following six types of registers are used to control the interrupt functions.

- Interrupt request flag register (IF0L, IF0H, IF1L)
- Interrupt mask flag register (MK0L, MK0H, MK1L)
- Priority specify flag register (PR0L, PR0H, PR1L)
- External interrupt mode register (EGP, EGN)
- Program status word (PSW)

Table 18-2 gives a listing of interrupt request flags, interrupt mask flags, and priority specify flags corresponding to interrupt request sources.

Table 18-2: Various Flags Corresponding to Interrupt Request Sources

Interrupt Request Signal Name	Interrupt Request Flag	Interrupt Mask Flag	Priority Specify Flag
INTWDT	WDTIF	WDTMK	WDTPR
INTVE	VEIF	VEMK	VEPR
INTVT	VTIF	VTMK	VTPR
INTVR	VRIF	VRMK	VRPR
INTP0	PIF0	PMK0	PPR0
INTP1	PIF1	PMK1	PPR1
INTP2	PIF2	PMK2	PPR2
INTTM00	TMIF00	TMMK00	TMPR00
INTTM01	TMIF01	TMMK01	TMPR01
INTM50	TMIF50	TMMK50	TMPR50
INTM51	TMIF51	TMMK51	TMPR51
INTWTI	WTIIF	WTIMK	WTIPR
INTWT	WTIF	WTMK	WTPR
INTCSI3	CSIIF3	CSIMK3	CSIPR3
INTSER	SERIF	SERMK	SERPR
INTSR	SRIF	SRMK	SRPR
INTST	STIF	STMK	STPR
INTAD	ADIF	ADMK	ADPR

(1) Interrupt request flag registers (IF0L, IF0H, IF1L)

The interrupt request flag is set to 1 when the corresponding interrupt request is generated or an instruction is executed. It is cleared to 0 when an instruction is executed upon acknowledgment of an interrupt request or upon application of $\overline{\text{RESET}}$ input.

IF0L, IF0H and IF1L are set with a 1-bit or 8-bit memory manipulation instruction. If IF0L and IF0H are used as a 16-bit register IF0, use a 16-bit memory manipulation instruction for the setting.

$\overline{\text{RESET}}$ input sets these registers to 00H.

Figure 18-2: Interrupt Request Flag Register Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
IF0L	TMIF00	PIF2	PIF1	PIF0	VRIF	VTIF	VEIF	WDTIF	FFE0H	00H	R/W
IF0H	SRIF	SERIF	CSIIF3	WTIF	WTIIF	TMIF51	TMIF50	TMIF01	FFE1H	00H	R/W
IF1L	0	0	0	0	0	0	ADIF	STIF	FFE2H	00H	R/W

xxIFx	Interrupt Request Flag
0	No interrupt request signal
1	Interrupt request signal is generated; Interrupt request state
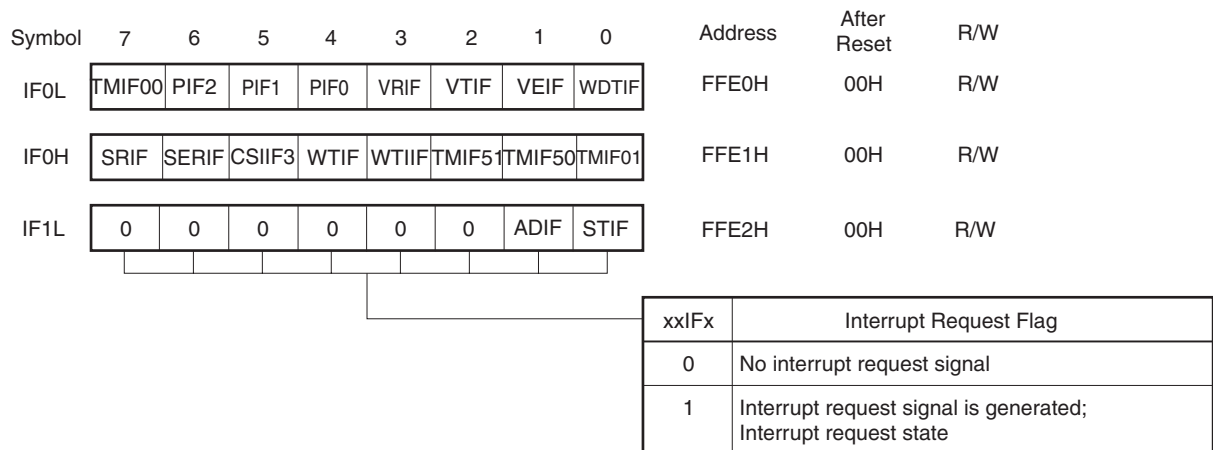
Cautions: 1. WDTIF flag is R/W enabled only when a watchdog timer is used as an interval timer. If used in the watchdog timer mode 1, set WDTIF flag to 0.

2. Set always 0 in IF1L bit 2 to bit 7.

(2) Interrupt mask flag registers (MK0L, MK0H, MK1L)

The interrupt mask flag is used to enable/disable the corresponding maskable interrupt service and to set standby clear enable/disable.

MK0L, MK0H and MK1L are set with a 1-bit or 8-bit memory manipulation instruction. If IF0L and IF0H are used as a 16-bit register MK0, use a 16-bit memory manipulation instruction for the setting.

$\overline{\text{RESET}}$ input sets these registers to FFH.

Figure 18-3: Interrupt Mask Flag Register Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
MK0L	TMMK00	PMK2	PMK1	PMK0	VRMK	VTMK	VEMK	WDTMK	FFE4H	FFH	R/W
MK0H	SRMK	SERMK	CSIMK3	WTMK	WTIMK	TMMK51	TMMK50	TMMK01	FFE5H	FFH	R/W
MK1L	1	1	1	1	1	1	ADMK	STMK	FFE6H	FFH	R/W

xxMKx	Interrupt Servicing Control
0	Interrupt servicing enabled
1	Interrupt servicing disabled
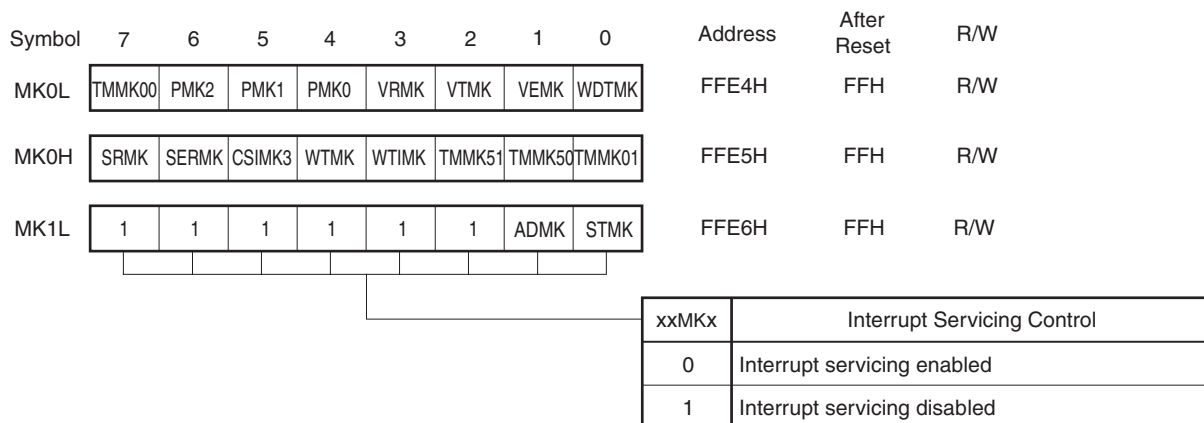
Cautions: 1. If WDTMK flag is read when a watchdog timer is used as a non-maskable interrupt, MK0 value becomes undefined.
2. Set always 1 in MK1L bit 2 to bit 7.

(3) Priority specify flag registers (PR0L, PR0H, PR1L)

The priority specify flag is used to set the corresponding maskable interrupt priority orders.

PR0L, PR0H and PR1L are set with a 1-bit or 8-bit memory manipulation instruction. If IF0L and IF0H are used as a 16-bit register PR0, use a 16-bit memory manipulation instruction for the setting. RESET input sets these registers to FFH.

Figure 18-4: Priority Specify Flag Register Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
PR0L	TMPR00	PPR2	PPR1	PPR0	VRPR	VTPR	VEPR	WDTPR	FFE8H	FFH	R/W
PR0H	SRPR	SERPR	CSIPR3	WTPR	WTIPR	TMPR51	TMPR50	TMPR01	FFE9H	FFH	R/W
PR1L	0	0	0	0	0	0	ADPR	STPR	FFEAH	FFH	R/W

xxPRx	Priority Level Selection
0	High priority level
1	Low priority level
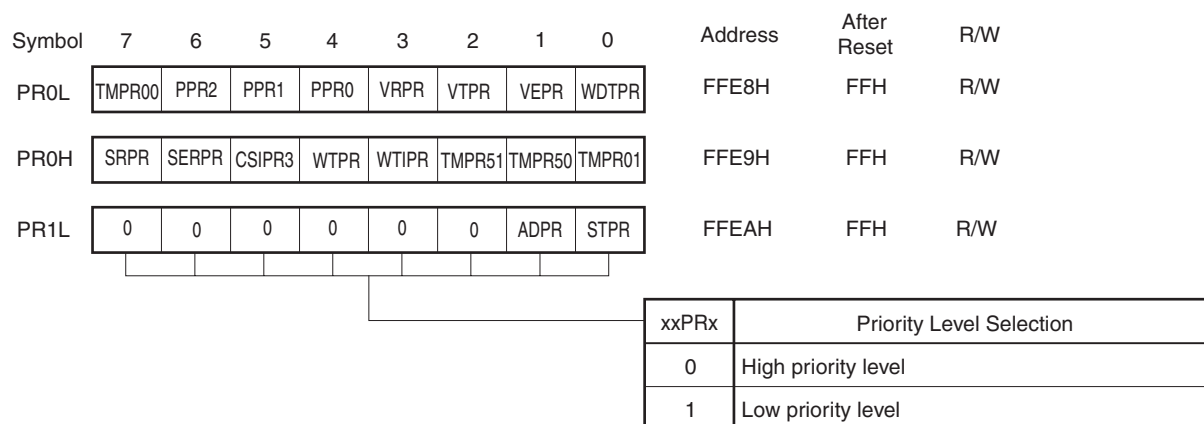
Cautions: 1. When a watchdog timer is used as a non-maskable interrupt, set 1 in WDTPR flag.
2. Set always 1 in PR1L bit 2 to bit 7.

(4) External interrupt rising edge enable register (EGP), external interrupt falling edge enable register (EGN)

EGP and EGN specify the valid edge to be detected on pins P00 to P02.

EGP and EGN can be read or written to with a 1-bit or 8-bit memory manipulation instruction.

These registers are set to 00H when the $\overline{\text{RESET}}$ signal is output.

Figure 18-5: Formats of External Interrupt Rising Edge Enable Register and External Interrupt Falling Edge Enable Register

Symbol	7	6	5	4	3	2	1	0	Address	On Reset	R/W
EGP	0	0	0	0	0	EGP2	EGP1	EGP0	FF48H	00H	R/W

Symbol	7	6	5	4	3	2	1	0	Address	On Reset	R/W
EGN	0	0	0	0	0	EGN2	EGN1	EGN0	FF49H	00H	R/W

EGPn	EGNn	Valid edge of INTPn pin (n = 0 4)
0	0	Interrupt disable
0	1	Falling edge
1	0	Rising edge
1	1	Both rising and falling edges

(5) Program status word (PSW)

The program status word is a register to hold the instruction execution result and the current status for interrupt request. The IE flag to set maskable interrupt enable/disable and the ISP flag to control multiple interrupt servicing are mapped.

Besides 8-bit unit read/write, this register can carry out operations with a bit manipulation instruction and dedicated instructions (EI and DI). When a vectored interrupt request is acknowledged, and when the BRK instruction is executed, the contents of PSW automatically is saved into a stack and the IE flag is reset to 0. If a maskable interrupt request is acknowledged contents of the priority specify flag of the acknowledged interrupt are transferred to the ISP flag. The acknowledged contents of PSW is also saved into the stack with the PUSH PSW instruction. It is reset from the stack with the RETI, RETB, and POP PSW instructions.

RESET input sets PSW to 02H.

Figure 18-6: Program Status Word Format

	7	6	5	4	3	2	<1>	<0>	After Reset
PSW	IE	Z	RBS1	AC	RBS0	0	ISP	CY	02H

► Used when Normal Instruction is Executed

ISP	Priority of Interrupt Currently Being Received
0	High-priority interrupt servicing (low-priority interrupt disable)
1	Interrupt request not acknowledged or low-priority interrupt servicing (all-maskable interrupts enable)

IE	Interrupt Request Acknowledge Enable/Disable
0	Disable
1	Enable

315

18.4 Interrupt Servicing Operations

18.4.1 Non-maskable interrupt request acknowledge operation

A non-maskable interrupt request is unconditionally acknowledged even if in an interrupt request acknowledge disable state. It does not undergo interrupt priority control and has highest priority over all other interrupts.

If a non-maskable interrupt request is acknowledged, the acknowledged interrupt is saved in the stacks, PSW and PC, in that order, the IE and ISP flags are reset to 0, and the vector table contents are loaded into PC and branched.

A new non-maskable interrupt request generated during execution of a non-maskable interrupt servicing program is acknowledged after the current execution of the non-maskable interrupt servicing program is terminated (following RETI instruction execution) and one main routine instruction is executed. If a new non-maskable interrupt request is generated twice or more during non-maskable interrupt service program execution, only one non-maskable interrupt request is acknowledged after termination of the non-maskable interrupt service program execution.

Figure 18-7: Flowchart from Non-Maskable Interrupt Generation to Acknowledge
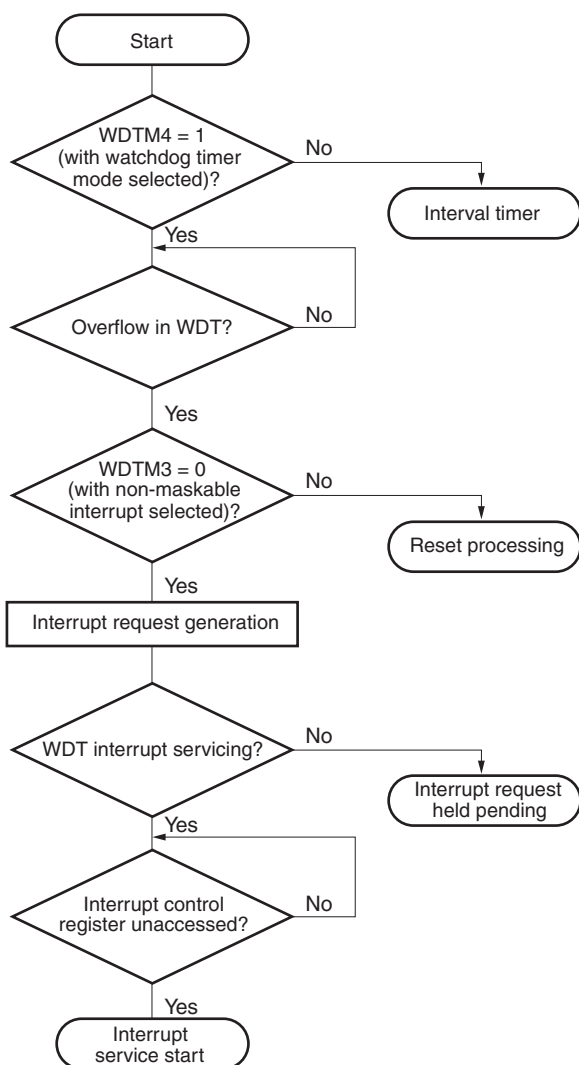
WDTM: Watchdog timer mode register
WDT: Watchdog timer

Figure 18-8: Non-Maskable Interrupt Request Acknowledge Timing
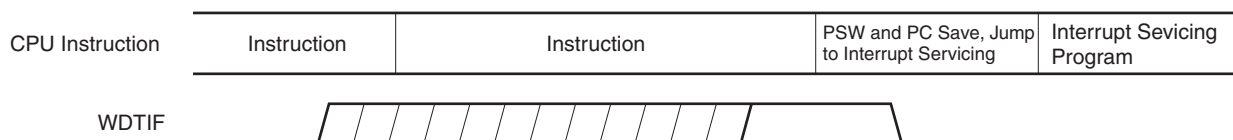
WDTIF: Watchdog timer interrupt request flag

317

Figure 18-9: Non-Maskable Interrupt Request Acknowledge Operation

(a) If a new non-maskable interrupt request is generated during non-maskable interrupt servicing program execution

Main Routine

NMI Request

1 Instruction Execution

NMI Request → NMI Request Reserve

Reserved NMI Request Processing
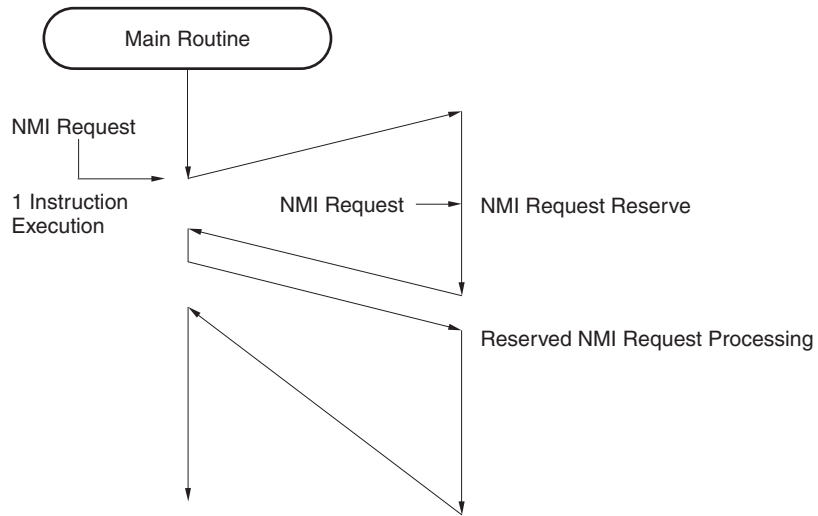
(b) If two non-maskable interrupt requests are generated during non-maskable interrupt servicing program execution

Main Routine

NMI Request

1 Instruction Execution

NMI Request → Reserved

NMI Request → Reserved

Although two or more NMI requests have been generated, only one request has been acknowledged.
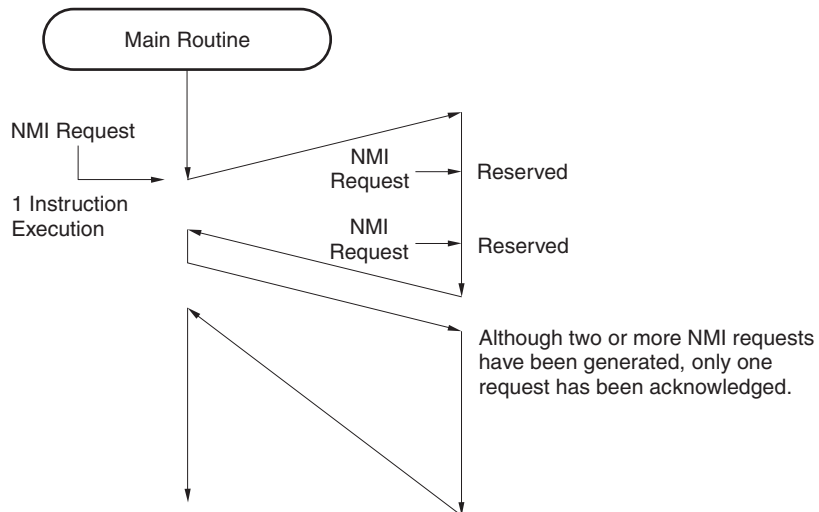
18.4.2 Maskable interrupt request acknowledge operation

A maskable interrupt request becomes acknowledgeable when an interrupt request flag is set to 1 and the interrupt mask (MK) flag is cleared to 0. A vectored interrupt request is acknowledged in an interrupt enable state (with IE flag set to 1). However, a low-priority interrupt request is not acknowledged during high-priority interrupt service (with ISP flag reset to 0).

Wait times maskable interrupt request generation to interrupt servicing are as follows.

Table 18-3: Times from Maskable Interrupt Request Generation to Interrupt Service

	Minimum Time	Maximum Time[Note]
When xxPRx = 0	7 clocks	32 clocks
When xxPRx = 1	8 clocks	33 clocks

Note: If an interrupt request is generated just before a divide instruction, the wait time is maximized.

Remark: 1 clock: $\dfrac{1}{f_{CPU}}$ (f_{CPU}: CPU clock)

If two or more maskable interrupt requests are generated simultaneously, the request specified for higher priority with the priority specify flag is acknowledged first. If two or more requests are specified for the same priority with the priority specify flag, the interrupt request with the higher default priority is acknowledged first.

Any reserved interrupt requests are acknowledged when they become acknowledgeable.

Figure 18-10 shows interrupt request acknowledge algorithms.

When a maskable interrupt request is acknowledged, the contents of program status word (PSW) and program counter (PC) are saved to stacks, in this order. Then, the IE flag is reset (to 0), and the value of the acknowledged interrupt priority specify flag is transferred to the ISP flag. Further, the vector table data determined for each interrupt request is loaded into PC and branched.

Return from the interrupt is possible with the RETI instruction.

Figure 18-10: Interrupt Request Acknowledge Processing Algorithm
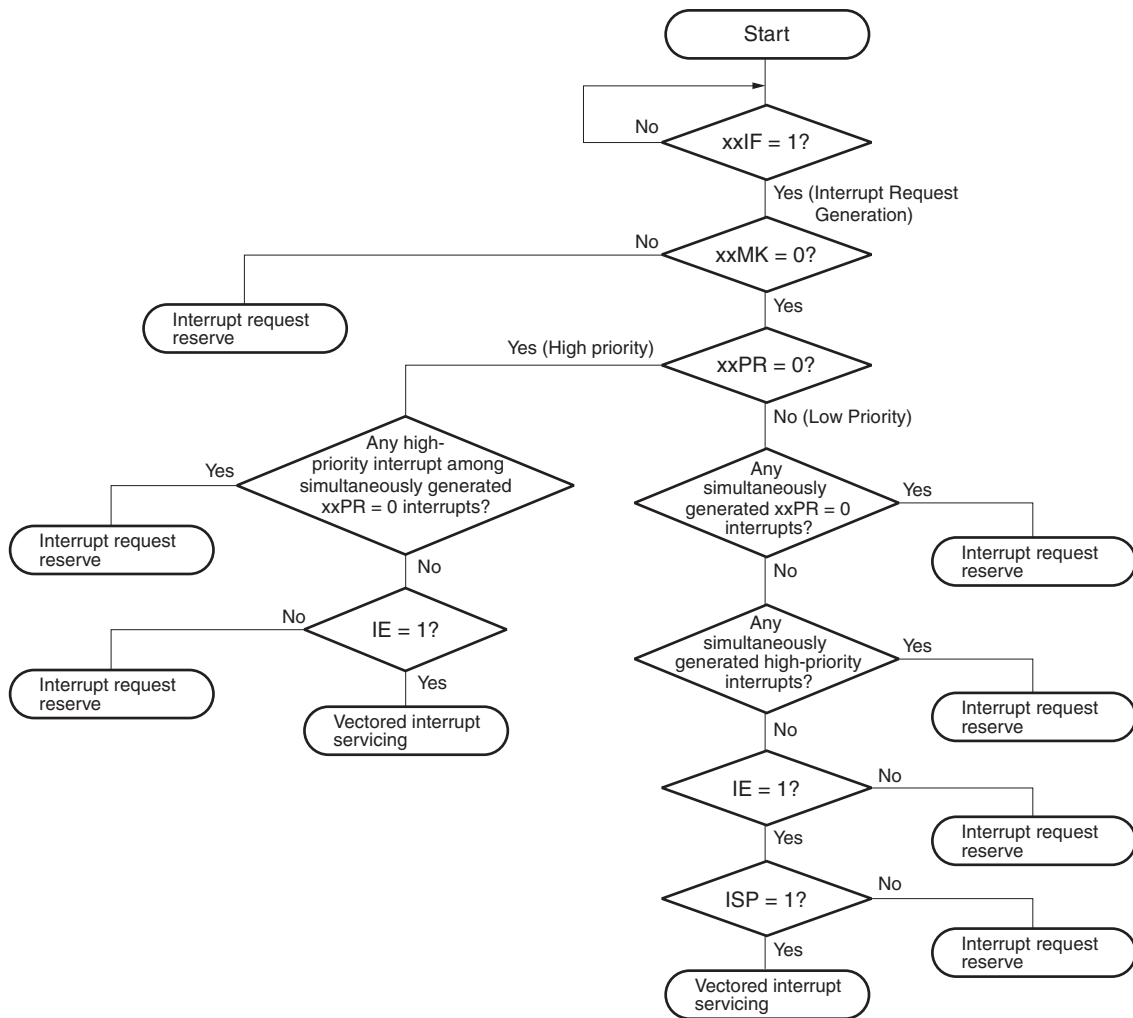
xxIF   : Interrupt request flag
xxMK : Interrupt mask flag
xxPR : Priority specify flag
IE   : Flag to control maskable interrupt request acknowledge
ISP   : Flag to indicate the priority of interrupt being serviced (0 = an interrupt with higher priority is being serviced, 1 = interrupt request is not acknowledged or an interrupt with lower priority is being serviced)

Figure 18-11: Interrupt Request Acknowledge Timing (Minimum Time)
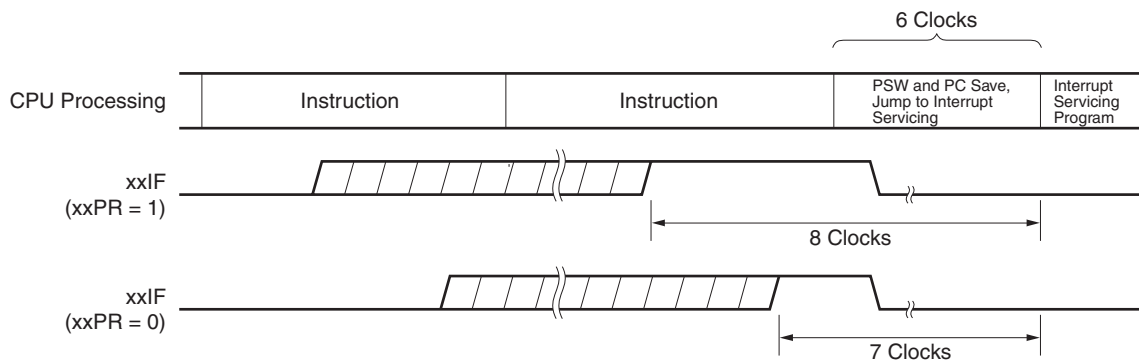
Remark: 1 clock: $\dfrac{1}{f_{CPU}}$ (f_{CPU}: CPU clock)

Figure 18-12: Interrupt Request Acknowledge Timing (Maximum Time)
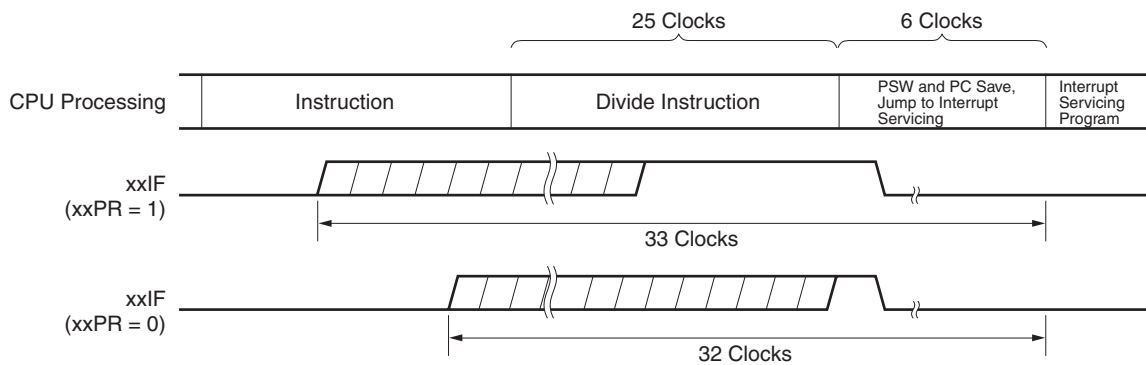
Remark: 1 clock: $\dfrac{1}{f_{CPU}}$ (f_{CPU}: CPU clock)

18.4.3 Software interrupt request acknowledge operation

A software interrupt request is acknowledged by BRK instruction execution. Software interrupt cannot be disabled.

If a software interrupt is acknowledged, the contents of program status word (PSW) and program counter (PC) are saved to stacks, in this order. Then the IE flag is reset (to 0), and the contents of the vector tables (003EH and 003FH) are loaded into PC and branched.

Return from the software interrupt is possible with the RETB instruction.

Caution: Do not use the RETI instruction for returning from the software interrupt.

18.4.4 Multiple interrupt servicing

A multiple interrupt consists in acknowledging another interrupt during the execution of the interrupt. A multiple interrupt is generated only in the interrupt request acknowledge enable state (IE = 1) (except non-maskable interrupt). As soon as an interrupt request is acknowledged, it enters the acknowledge disable state (IE = 0). Therefore, in order to enable a multiple interrupt, it is necessary to set the interrupt enable state by setting the IE flag (1) with the EI instruction during interrupt servicing.

Even in an interrupt enabled state, a multiple interrupt may not be enabled. However, it is controlled according to the interrupt priority. There are two priorities, the default priority and the programmable priority. The multiple interrupt is controlled by the programmable priority control.

If an interrupt request with the same or higher priority than that of the interrupt being serviced is generated, it is acknowledged as a multiple interrupt. In the case of an interrupt with a priority lower than that of the interrupt being processed, it is not acknowledged as a multiple interrupt.

Interrupt request not acknowledged as a multiple interrupt due to interrupt disable or a low priority is reserved and acknowledged following one instruction execution of the main processing after the completion of the interrupt being serviced.

During non-maskable interrupt servicing, multiple interrupts are not enabled.

Table 18-4 shows an interrupt request enabled for multiple interrupt during interrupt servicing, and Figure 18-13 shows multiple interrupt examples.

Table 18-4: Interrupt Request Enabled for Multiple Interrupt during Interrupt Servicing

Multiple Interrupt Request / Interrupt being Serviced		Non-maskable Interrupt Request	Maskable Interrupt Request			
			xxPR = 0		xxPR = 1	
			IE = 1	IE = 0	IE = 1	IE = 0
Non-maskable interrupt		D	D	D	D	D
Maskable interrupt	ISP = 0	E	E	D	D	D
	ISP = 1	E	E	D	E	D
Software interrupt		E	E	D	E	D

Remarks: **1.** E: Multiple interrupt enable
    2. D: Multiple interrupt disable
    3. ISP and IE are the flags contained in PSW
       ISP = 0: An interrupt with higher priority is being serviced
       ISP = 1: An interrupt request is not accepted or an interrupt with lower priority is being serviced
       IE = 0: Interrupt request acknowledge is disabled
       IE = 1: Interrupt request acknowledge is enabled
    4. xxPR is a flag contained in PR0L, PR0H, and PRIL
       xxPR = 0: Higher priority level
       xxPR = 1: Lower priority level

Figure 18-13: Multiple Interrupt Example (1/2)

Example 1. Two multiple interrupts generated
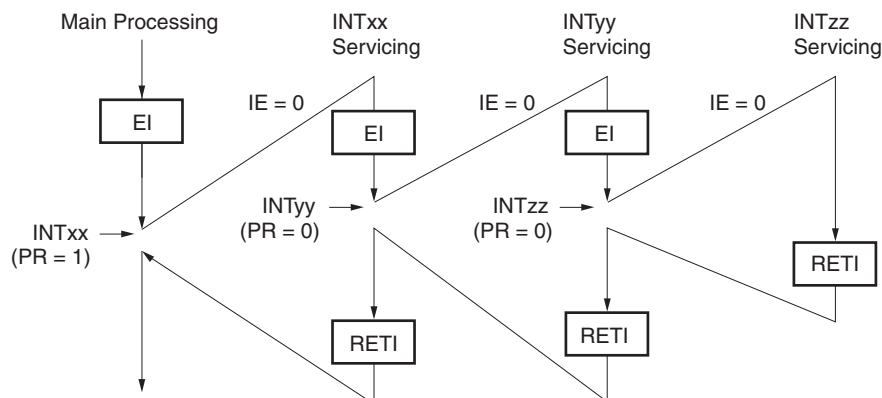
During interrupt INTxx servicing, two interrupt requests, INTyy and INTzz are acknowledged, and a multiple interrupt is generated. An EI instruction is issued before each interrupt request acknowledge, and the interrupt request acknowledge enable state is set.

Example 2. Multiple interrupt is not generated by priority control
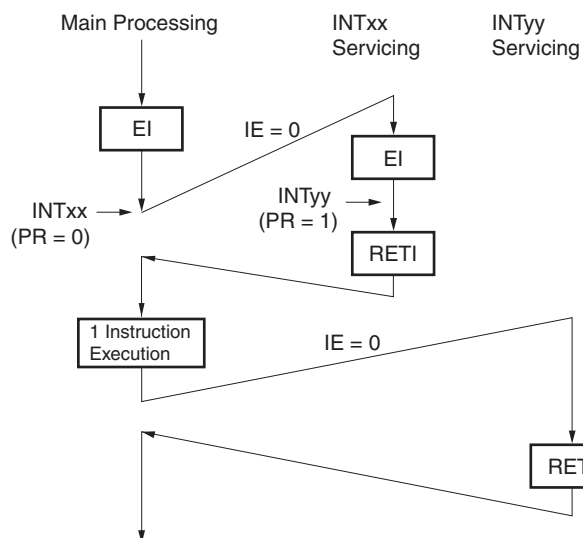
The interrupt request INTyy generated during interrupt INTxx servicing is not acknowledged because the interrupt priority is lower than that of INTxx, and a multiple interrupt is not generated. INTyy request is retained and acknowledged after execution of 1 instruction execution of the main processing.

PR = 0 : Higher priority level
PR = 1 : Lower priority level
IE = 0   : Interrupt request acknowledge disable

Figure 18-13: Multiple Interrupt Example (2/2)

Example 3. A multiple interrupt is not generated because interrupts are not enabled
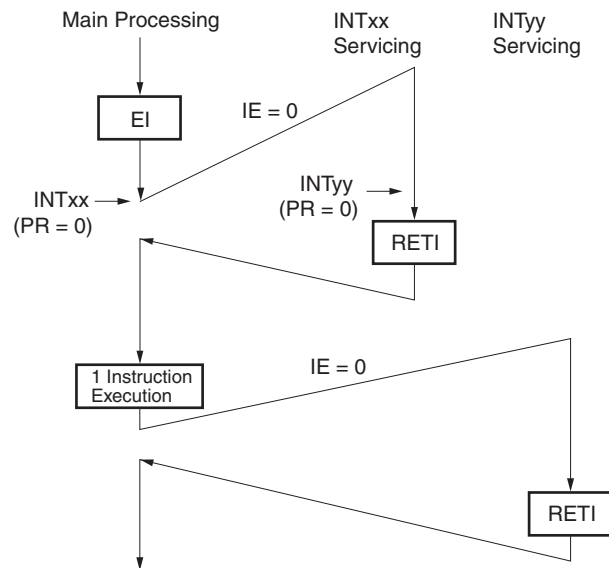
Because interrupts are not enabled in interrupt INTxx servicing (an EI instruction is not issued), interrupt request INTyy is not acknowledged, and a multiple interrupt is not generated. The INTyy request is reserved and acknowledged after 1 instruction execution of the main processing.

PR = 0 : Higher priority level
IE = 0   : Interrupt request acknowledge disable

18.4.5 Interrupt request reserve

Some instructions may reserve the acknowledge of an instruction request until the completion of the execution of the next instruction even if the interrupt request is generated during the execution. The following shows such instructions (interrupt request reserve instruction).

- MOV PSW, #byte
- MOV A, PSW
- MOV PSW, A
- MOV1 PSW.bit, CY
- MOV1 CY, PSW.bit
- AND1 CY, PSW.bit
- OR1   CY, PSW.bit
- XOR1 CY, PSW.bit
- SET1/CLR1 PSW.bit
- RETB
- RETI
- PUSH PSW
- POP PSW
- BT   PSW.bit, $addr16
- BF   PSW.bit, $addr16
- BTCLRPSW.bit, $addr16
- EI
- DI
- Manipulate instructions for IF0L, IF0H, IF1L, MK0L, MK0H, MK1L, PR0L, PR0H, PR1L, INTM0, INTM1 registers

Caution: **BRK instruction is not an interrupt request reserve instruction described above. However, in a software interrupt started by the execution of BRK instruction, the IE flag is cleared to 0. Therefore, interrupt requests are not acknowledged even when a maskable interrupt request is issued during the execution of the BRK instruction. However, non-maskable interrupt requests are acknowledged.**

Figure 18-14 shows the interrupt request hold timing.

Figure 18-14: Interrupt Request Hold

CPU processing	Instruction N	Instruction M	Save PSW and PC, Jump to interrupt service	Interrupt service program
xxIF				
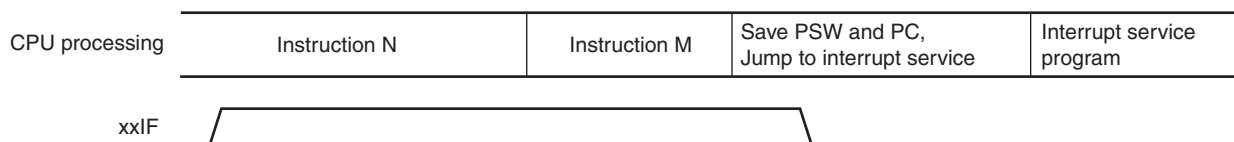
Remarks: **1.** Instruction N: Instruction that holds interrupts requests
2. Instruction M: Instructions other than interrupt request pending instruction
3. The xxPR (priority level) values do not affect the operation of xxIF (interrupt request).

[Memo]

Chapter 19 Standby Function

19.1 Standby Function and Configuration

19.1.1 Standby function
The standby function is designed to decrease power consumption of the system. The following two modes are available.

(1) HALT mode
HALT instruction execution sets the HALT mode. The HALT mode is intended to stop the CPU operation clock. System clock oscillator continues oscillation. In this mode, current consumption cannot be decreased as in the STOP mode. The HALT mode is valid to restart immediately upon interrupt request and to carry out intermittent operations such as watch applications.

(2) STOP mode
STOP instruction execution sets the STOP mode. In the STOP mode, the main system clock oscillator stops and the whole system stops. CPU current consumption can be considerably decreased.
Data memory low-voltage hold (down to V_{DD} = 2.0 V) is possible. Thus, the STOP mode is effective to hold data memory contents with ultra-low current consumption. Because this mode can be cleared upon interrupt request, it enables intermittent operations to be carried out.
However, because a wait time is necessary to secure an oscillation stabilization time after the STOP mode is cleared, select the HALT mode if it is necessary to start processing immediately upon interrupt request.

In any mode, all the contents of the register, flag, and data memory just before standby mode setting are held. The input/output port output latch and output buffer statuses are also held.

Cautions: 1. The STOP mode can be used only when the system operates with the main system clock (subsystem clock oscillation cannot be stopped). The HALT mode can be used with either the main system clock or the subsystem clock.
2. When proceeding to the STOP mode, be sure to stop the peripheral hardware operation and execute the STOP instruction.
3. The following sequence is recommended for power consumption reduction of the A/D converter when the standby function is used: first clear bit 7 (CS) to 0 to stop the A/D conversion operation, and then execute the HALT or STOP instruction.

19.1.2 Standby function control register

A wait time after the STOP mode is cleared upon interrupt request till the oscillation stabilizes is controlled with the oscillation stabilization time select register (OSTS).

OSTS is set with an 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets OSTS to 04H. However, it takes $2^{17}/fx$ until the STOP mode is cleared by $\overline{\text{RESET}}$ input.

Figure 19-1: Oscillation Stabilization Time Select Register Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
OSTS	0	0	0	0	0	OSTS2	OSTS1	OSTS0	FFFAH	04H	R/W

OSTS2	OSTS1	OSTS0	Selection of Oscillation Stabilization Time when STOP Mode is Released
0	0	0	$2^{12}/fx$ (512 µs)
0	0	1	$2^{14}/fx$ (2 ms)
0	1	0	$2^{15}/fx$ (4.1 ms)
0	1	1	$2^{16}/fx$ (8.19 ms)
1	0	0	$2^{17}/fx$ (16.38 ms)
Other than above			Setting prohibited
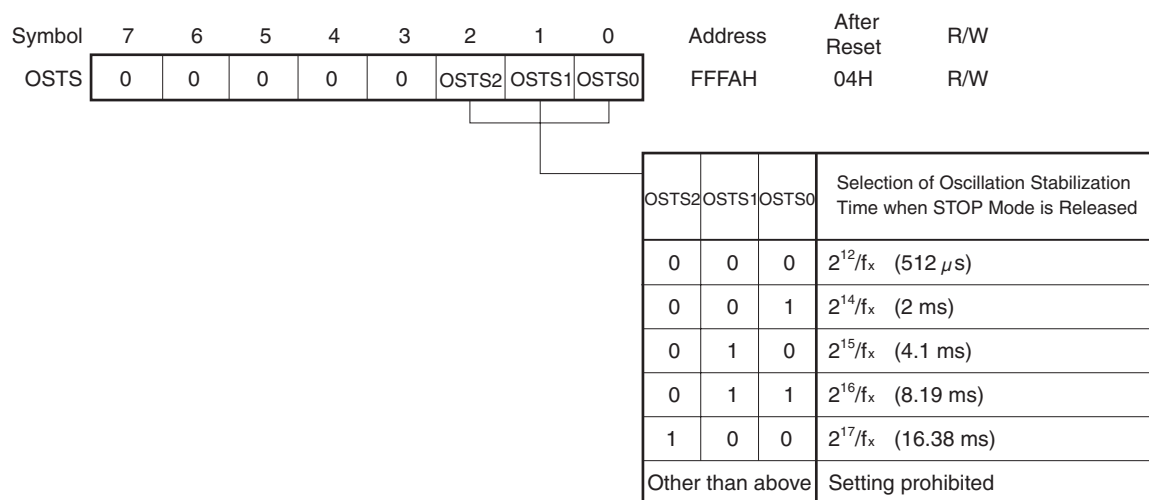
Caution: **The wait time after STOP mode clear does not include the time (see "a" in the illustration below) from STOP mode clear to clock oscillation start, regardless of clearance by $\overline{\text{RESET}}$ input or by interrupt generation.**
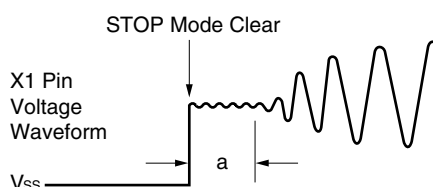
Remarks: **1.** fx: Main system clock oscillation frequency
2. Values in parentheses apply to operating at fx = 8.00 MHz

19.2 Standby Function Operations

19.2.1 HALT mode

(1) HALT mode set and operating status

The HALT mode is set by executing the HALT instruction. It can be set with the main system clock or the subsystem clock. The operating status in the HALT mode is described below.

Table 19-1: HALT Mode Operating Status

HALT mode setting / Item	HALT execution during main system clock operation	HALT execution during subsystem clock operation (Main system clock stops)
Clock generator	Both main and subsystem clocks can be oscillated / Clock supply to the CPU stops	
CPU	Operation stops	
Port (output latch)	Status before HALT mode setting is held	
16-bit timer /event counter (TM0)	Operable	Operation stops
8-bit timer event counter (TM50/TM51)	Operable	Operable when TI50 or TI51 is selected as count clock
Watch timer	Operable	Operable when fxt is selected as count clock
Watchdog timer	Operable	Operation stops
A/D converter	Operation stops	
Serial I/F - SIO3	Operable	Operable at external SCK
Serial I/F - UART	Operable	Operation stops
VAN	Operable	Operation stops
Sound generator	Operable	Operation stops
External interrupt (INTP0 to INTP2)	Operable	
LCD	Operable	Operation stops

(2) HALT mode clear

The HALT mode can be cleared with the following four types of sources.

(a) Clear upon unmasked interrupt request

An unmasked interrupt request is used to clear the HALT mode. If interrupt acknowledge is enabled, vectored interrupt service is carried out. If disabled, the next address instruction is executed.

Figure 19-2: HALT Mode Clear upon Interrupt Generation
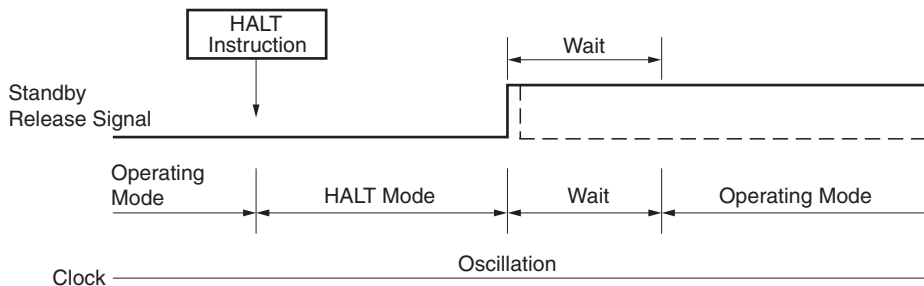
Remarks: **1.** The broken line indicates the case when the interrupt request which has cleared the standby status is acknowledged.

2. Wait time will be as follows:
- When vectored interrupt service is carried out: 8 to 9 clocks
- When vectored interrupt service is not carried out: 2 to 3 clocks

(b) Clear upon non-maskable interrupt request

The HALT mode is cleared and vectored interrupt service is carried out whether interrupt acknowledge is enabled or disabled.

(c) Clear upon $\overline{\text{RESET}}$ input

As is the case with normal reset operation, a program is executed after branch to the reset vector address.

Figure 19-3: HALT Mode Release by $\overline{\text{RESET}}$ Input
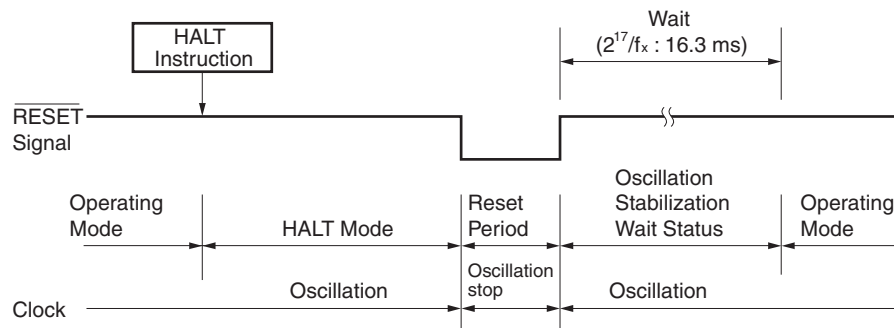
Remarks: **1.** fx: Main system clock oscillation frequency
2. Values in parentheses apply to operation at fx = 8.0 MHz

Table 19-2: Operation after HALT Mode Release

Release Source	MKxx	PRxx	IE	ISP	Operation
Maskable interrupt request	0	0	0	x	Next address instruction execution
	0	0	1	x	Interrupt service execution
	0	1	0	1	Next address instruction execution
	0	1	x	0	
	0	1	1	1	Interrupt service execution
	1	x	x	x	HALT mode hold
Non-maskable interrupt request	-	-	x	x	Interrupt service execution
$\overline{\text{RESET}}$ input	-	-	x	x	Reset processing

x: Don't care.

19.2.2 STOP mode

(1) STOP mode set and operating status
The STOP mode is set by executing the STOP instruction. It can be set only with the main system clock.

Cautions: 1. When the STOP mode is set, the X2 pin is internally connected to V$_{DD}$ via a pull-up resistor to minimize leakage current at the crystal oscillator. Thus, do not use the STOP mode in a system where an external clock is used for the main system clock.
2. Because the interrupt request signal is used to clear the standby mode, if there is an interrupt source with the interrupt request flag set and the interrupt mask flag reset, the standby mode is immediately cleared if set. Thus, the STOP mode is reset to the HALT mode immediately after execution of the STOP instruction. After the wait set using the oscillation stabilization time select register (OSTS), the operating mode is set.

The operating status in the STOP mode is described below.

Table 19-3: STOP Mode Operating Status

STOP mode setting / Item	With subsystem clock	Without subsystem clock
Clock generator	Only main system clock stops oscillation	
CPU	Operation stops	
Port (output latch)	Status before STOP mode setting is held	
16-bit timer /event counter (TM0)	Operation stops	
8-bit timer event counter 5 and 6	Operable when TI50 or TI51 are selected as count clock	
Watch timer	Operable when fxt is selected as count clock	Operation stops
Watchdog timer	Operation stops	
A/D converter	Operation stops	
Serial I/F - SIO3	Operable at external SCK	
Serial I/F - UART	Operation stops	
VAN	Operation stops	
Sound generator	Operation stops	
External interrupt (INTP0 to INTP2)	Operable	
LCD	Operation stops	

(2) STOP mode release

The STOP mode can be cleared with the following three types of sources.

(a) Release by unmasked interrupt request

An unmasked interrupt request is used to release the STOP mode. If interrupt acknowledge is enabled after the lapse of oscillation stabilization time, vectored interrupt service is carried out. If interrupt acknowledge is disabled, the next address instruction is executed.

Figure 19-4: STOP Mode Release by Interrupt Generation
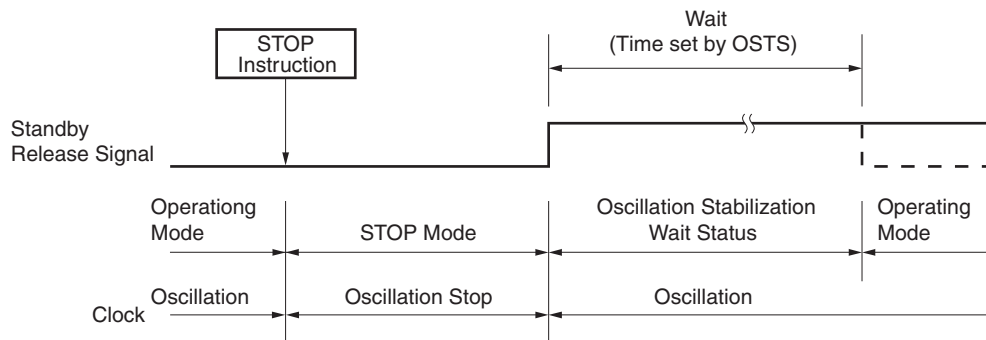
Remark: The broken line indicates the case when the interrupt request which has cleared the standby status is acknowledged.

(b) Release by R̄E̅S̅E̅T̅ input

The STOP mode is cleared and after the lapse of oscillation stabilization time, reset operation is carried out.

Figure 19-5: Release by STOP Mode R̄E̅S̅E̅T̅ Input
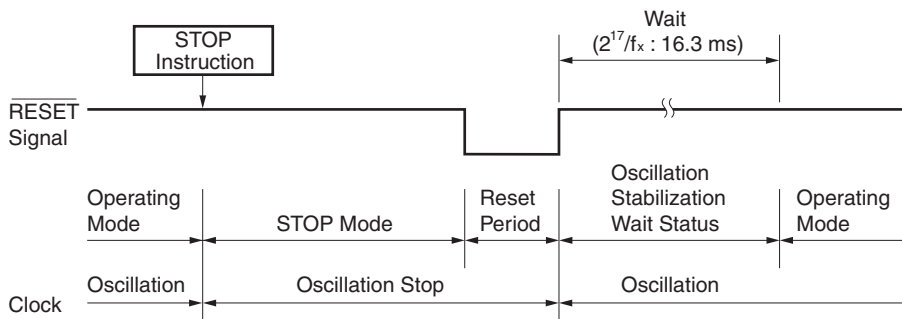
Remarks   **1.** fx: Main system clock oscillation frequency
              2. Values in parentheses apply to operation at fx = 5.0 MHz

Table 19-4: Operation after STOP Mode Release

Release Source	MKxx	PRxx	IE	ISP	Operation
Maskable interrupt request	0	0	0	x	Next address instruction execution
	0	0	1	x	Interrupt service execution
	0	1	0	1	Next address instruction execution
	0	1	x	0	
	0	1	1	1	Interrupt service execution
	1	x	x	x	STOP mode hold
Non-maskable interrupt request	-	-	x	x	Interrupt service execution
R̄E̅S̅E̅T̅ input	-	-	x	x	Reset processing

x: Don't care.

[Memo]

Chapter 20 Reset Function

20.1 Reset Function

The following two operations are available to generate the reset signal.

(1) External reset input with $\overline{\text{RESET}}$ pin
(2) Internal reset by watchdog timer overrun time detection

External reset and internal reset have no functional differences. In both cases, program execution starts at the address at 0000H and 0001H by $\overline{\text{RESET}}$ input.
When a low level is input to the $\overline{\text{RESET}}$ pin or the watchdog timer overflows, a reset is applied and each hardware is set to the status as shown in Table 20-1. Each pin has high impedance during reset input or during oscillation stabilization time just after reset clear.
When a high level is input to the $\overline{\text{RESET}}$ input, the reset is cleared and program execution starts after the lapse of oscillation stabilization time (2^{17}/fx). The reset applied by watchdog timer overflow is automatically cleared after a reset and program execution starts after the lapse of oscillation stabilization time (2^{17}/fx) (see Figure 20-2 to 20-4).

Cautions: **1. For an external reset, input a low level for 10 μs or more to the $\overline{\text{RESET}}$ pin.**
2. During reset input, main system clock oscillation remains stopped but subsystem clock oscillation continues.
3. When the STOP mode is cleared by reset, the STOP mode contents are held during reset input. However, the port pin becomes high-impedance.

Figure 20-1: Block Diagram of Reset Function
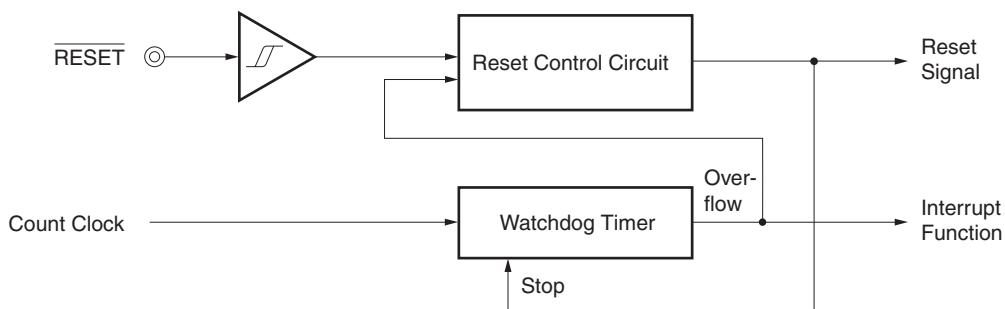
Figure 20-2: Timing of Reset Input by $\overline{RESET}$ Input
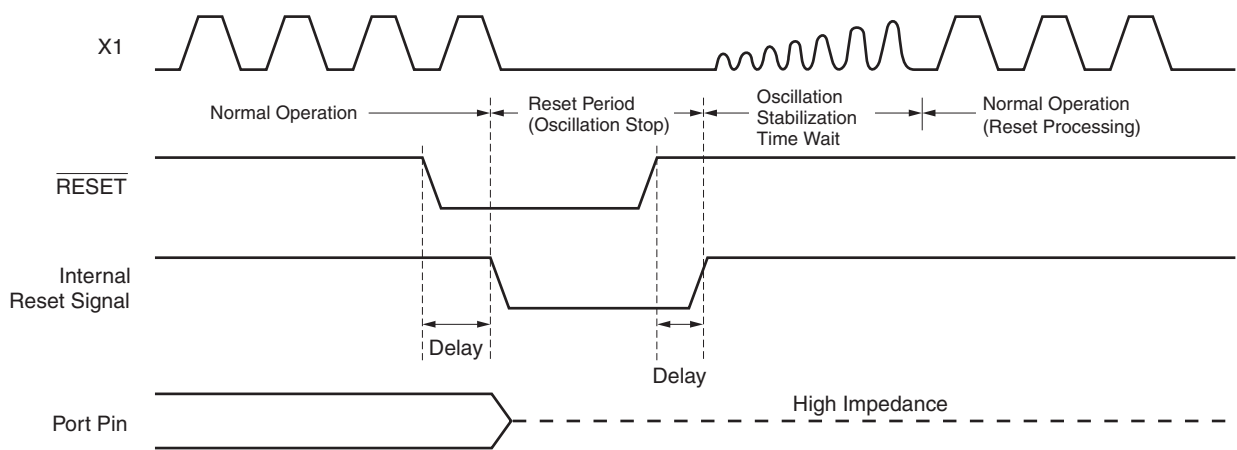
Figure 20-3: Timing of Reset due to Watchdog Timer Overflow
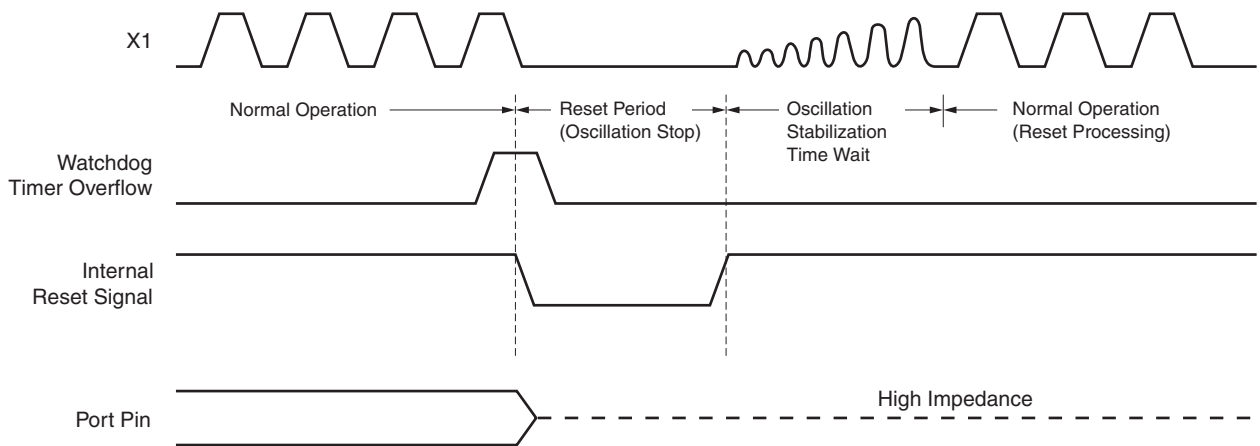
Figure 20-4: Timing of Reset Input in STOP Mode by $\overline{RESET}$ Input

337
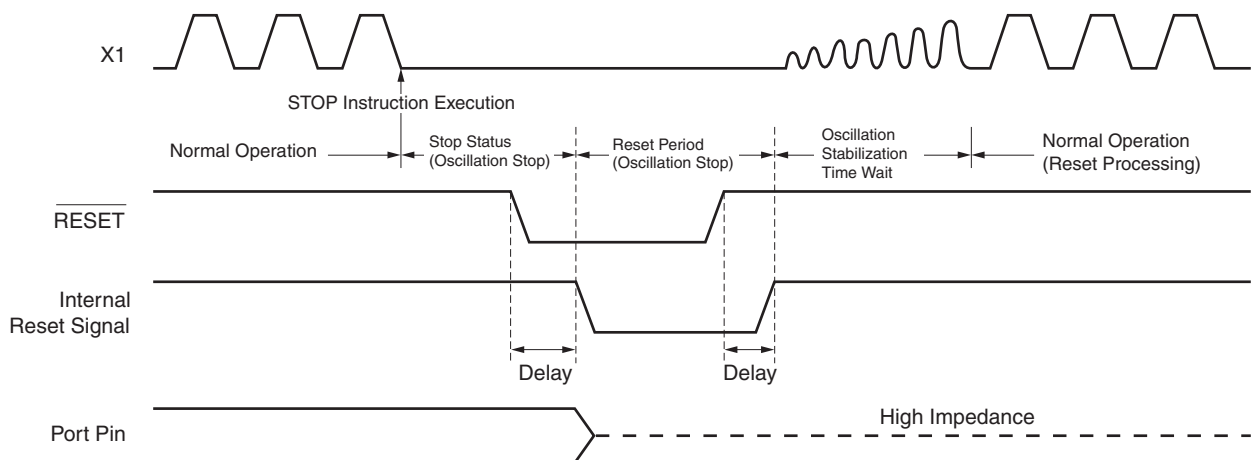
Table 20-1: Hardware Status after Reset (1/2)

Hardware		Status after Reset
Program counter (PC) Note 1		The contents of reset vector tables (0000H and 0001H) are set
Stack pointer (SP)		Undefined
Program status word (PSW)		02H
RAM	Data memory	Undefined) Note 2
	General register	Undefined) Note 2
Port (Output latch)	Ports 0, 4, 8 to 12 (P0, P4, P8 to P12)	00H
Port mode register (PM0, PM4, PM8 to PM12)		FFH
Port function register (PF8 to PF12)		00H
Processor clock control register (PCC)		04H
Memory size switching register (IMS)		CFH
Internal expansion RAM size switching register (IXS)		0CH
Oscillation stabilization time select register (OSTS)		04H
16-bit timer/event counter 0	Timer register (TM0)	0000H
	Capture/compare register (CR00, CR01)	0000H
	Prescaler selection register (PRM0)	00H
	Mode control register (TMC0)	00H
	Capture/compare control register 0 (CRC0)	00H
	Output control register (TOC0)	00H
8-bit timer/event counters 50 and 51	Timer register (TM50, TM51)	00H
	Compare register (CR50, CR51)	00H
	Clock select register (TLC50, TLC51)	00H
	Mode control register (TMC50, TMC51)	04H
Watch timer	Mode register (WTM)	00H
Watchdog timer	Clock selection register (WDCS)	00H
	Mode register (WDTM)	00H
PCL clock output	Clock output selection register (CKS)	00H
Sound generator	Control register (SGCR)	00H
	Amplitude control register (SGAM)	00H
	Buzzer control register (SGBC)	00H

Notes: **1.** During reset input or oscillation stabilization time wait, only the PC contents among the hardware statuses become undefined. All other hardware statuses remains unchanged after reset.

2. The post-reset status is held in the standby mode.

Table 20-1: Hardware Status after Reset (2/2)

Hardware		Status after Reset
Serial interface	Operating mode register 0 (CSIM30)	00H
	Shift register 0 (CSIO30)	00H
	Operating mode register 1 (CSIM31)	00H
	Shift register 1 (CSIO31)	00H
	Asynchronous mode register (ASIM0)	00H
	Asynchronous status register (ASIS0)	00H
	Baudrate generator control register (BRGC0)	00H
	Transmit shift register (TXS0)	FFH
	Receive buffer register (RXB0)	
A/D converter	Mode register (ADM1)	00H
	Conversion result register (ADCR1)	00H
	Input select register (ADS1)	00H
	Power fail comparator mode (PFM)	00H
	Power fail threshold register (PFT)	00H
LCD controller/driver	Mode register (LCDM)	00H
	Control register (LCDC)	00H
Interrupt	Request flag register (IF0L, IF0H, IF1L)	00H
	Mask flag register (MK0L, MK0H, MK1L)	FFH
	Priority specify flag register (PR0L, PR0H, PR1L)	FFH
	External interrupt rising edge register (EGP)	00H
	External interrupt falling edge register (EGN)	00H
VAN	UDL clock control register (UDLCCL)	00H

[Memo]

Chapter 21   µPD16F15A

The µPD16F15A replaces the internal mask ROM of the µPD1615A(A) series with flash memory to which a program can be written, deleted and overwritten while mounted on the substrate.
Table 21-1 lists the differences among the µPD16F15A and the mask ROM versions.

Table 21-1: Differences among µPD16F15A and Mask ROM Versions

Item	µPD16F15A	Mask ROM Versions
IC pin	None	Available
VPP pin	Available	None
Electrical characteristics	See data sheet of each product	

Caution: **Flash memory versions and mask ROM versions differ in their noise tolerance and noise emission. If replacing flash memory versions with mask ROM versions when changing from test production to mass production, be sure to perform sufficient evaluation with CS versions (not ES versions) of mask ROM versions.**

21.1 Memory Size Switching Register (IMS)

This register specifies the internal memory size by using the memory size switching register (IMS), so that the same memory map as on the mask ROM version can be achieved.
IMS is set with an 8-bit memory manipulation instruction.
RESET input sets this register to CFH.

Figure 21-1: Memory Size Switching Register Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
IMS	RAM2	RAM1	RAM0	0	ROM3	ROM2	ROM1	ROM0	FFF0H	Note	R/W

ROM3	ROM2	ROM1	ROM0	Internal ROM size selection
1	0	0	0	32 Kbytes
1	1	0	0	48 Kbytes
1	1	1	1	60 Kbytes
Other than above				Setting prohibited

RAM2	RAM1	RAM0	Internal high-speed RAM size selection
0	0	0	768 bytes
1	1	0	1024 bytes
Other than above			Setting prohibited
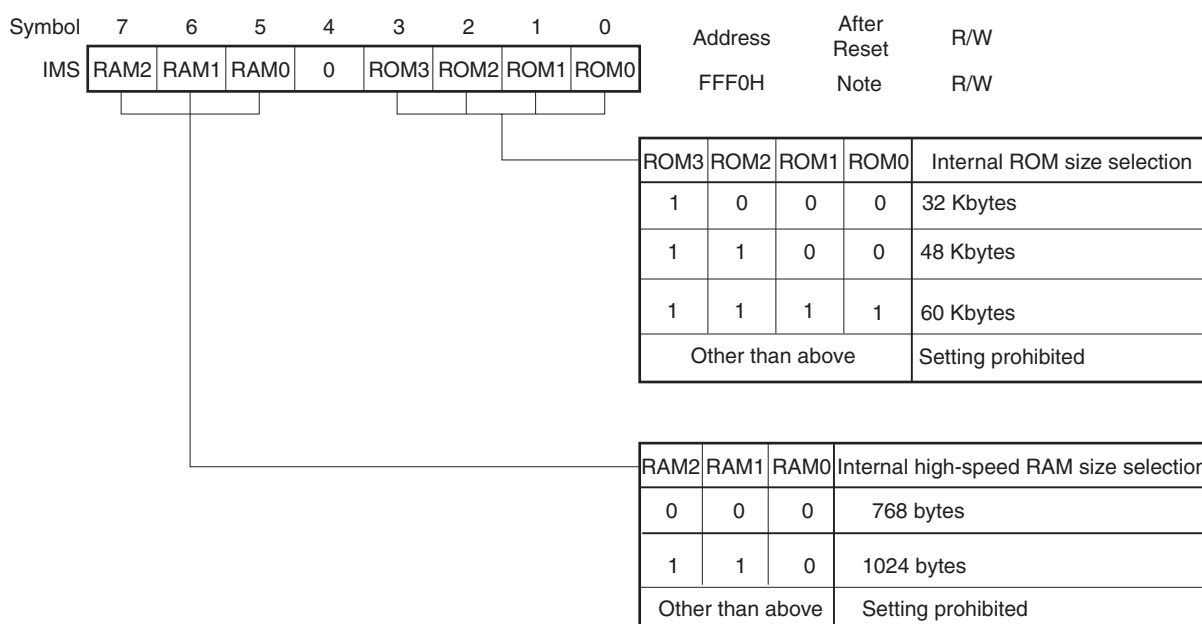
Note: The values after reset depend on the product (See Table 21-2).

Table 21-2: Values of the Memory Size Switching Register for the Different Devices

Part Number	Value
µPD1615A(A)	CFH
µPD1615B(A)	CCH
µPD1615F(A)	08H
µPD1616F(A)	08H
µPD16F15A	CFH

Caution: **When the µPD1615A(A), µPD1615B(A), µPD1615F(A),µPD1616F(A), and the µPD16F15A are used, be sure to set the value of the IMS register as given in the Table 21-2.**

21.2 Internal Extension RAM Size Switching Register

The µPD16F15A allow users to define its internal expansion RAM size by using the internal expansion RAM size switching register (IXS), so that the same memory mapping as that of a mask ROM version with a different internal extension RAM is possible.
The IXS is set by an 8-bit memory manipulation instruction.
RESET signal input sets IXS to 0CH.

Caution: **When the µPD1615A(A), µPD1615B(A), µPD1615F(A),µPD1616F(A), and the µPD16F15A are used, be sure to set the value specified in the Table 21-3 to IXS. Other settings are prohibited.**

Figure 21-2: Internal Extension RAM Size Switching Register Format

Symbol	7	6	5	4	3	2	1	0	Address	After Reset	R/W
IXS	0	0	0	0	IXRAM3	IXRAM2	IXRAM1	IXRAM0	FFF4H	0CH	W

IXRAM3	IXRAM2	IXRAM1	IXRAM0	Internal extension RAM capacity selection
1	0	1	1	512 bytes
1	0	1	0	1024 bytes
Other than above				Setting prohibited
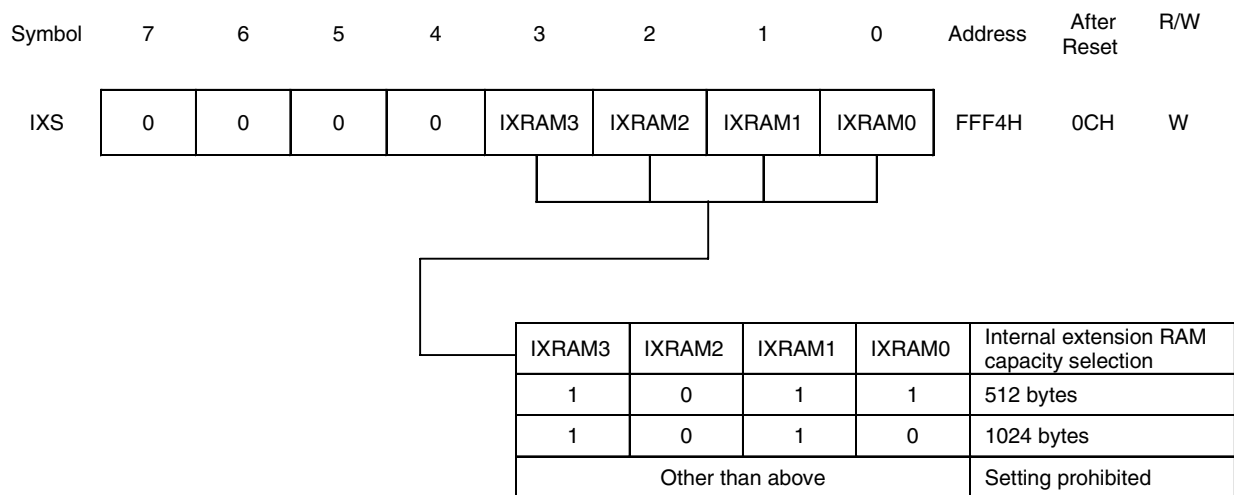
The value whitch is set in the IXS that has the identical memory map to the mask ROM versions is given in the Table 21-3.

Table 21-3: Examples of internal Extension RAM Size Switching Register Settings

Relevant Mask ROM Version	IXS Setting
µPD1615A(A)	0AH
µPD1615B(A)	0BH
µPD1615F(A)	0BH
µPD1616F(A)	0BH
µPD16F15A	0AH

Caution: **When the µPD1615A(A), µPD1615B(A), µPD1615F(A),µPD1616F(A), and the µPD16F15A are used, be sure to set the value of the IXS register as given in the Table 21-3.**

21.3 Flash memory programming

On-board writing of flash memory (with device mounted on target system) is supported.
On-board writing is done after connecting a dedicated flash writer to the host machine and target system.
Moreover, writing to flash memory can also be performed using a flash memory writing adapter connected to the Flash Programmer.

21.3.1 Selection of transmission method

Writing to flash memory is performed using Flashpro and serial communication. Select the transmission method for writing from Table 21-4. For the selection of the transmission method, a format like the one shown in Figure 21-3 is used. The transmission methods are selected with the V_{PP} pulse numbers shown in Table 21-4.

Table 21-4: Transmission Method List

Transmission Method	Number of Channels	Pin Used	Number of V_{PP} Pulses
3-wire serial I/O	1	SI3/P127 SO3/P126 SCK3/P125	0
Pseudo 3-wire serial I/O	1	P40 (Serial clock input) P41(Serial data input) P42(Serial data input)	12
UART	1	RxD0/P123 TxD0/P124	8

Cautions: **1. Be sure to select the number of V_{PP} pulses shown in Table 25-3 for the transmission method.**
2. If performing write operations to flash memory with the UART transmission method, set the main system clock oscillation frequency to 4 MHz or higher.

Figure 21-3: Transmission Method Selection Format
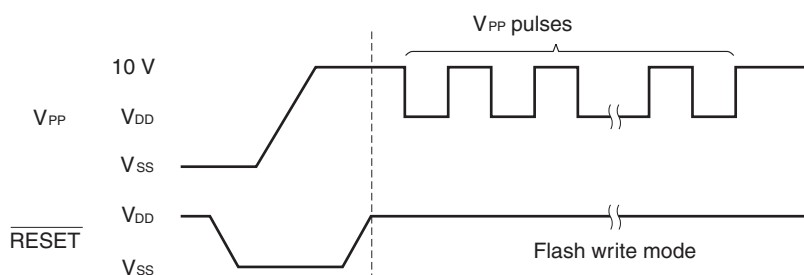
21.3.2 Initialization of the programming mode

When V_{PP} reaches up to 10 V with $\overline{RESET}$ terminal activated, on-board programming mode becomes available.
After release of $\overline{RESET}$, the programming mode is selected by the number of V_{PP} pulses.

21.3.3 Flash memory programming function

Flash memory writing is performed through command and data transmit/receive operations using the selected transmission method. The main functions are listed in Table 21-5.

Table 21-5: Main Functions of Flash Memory Programming

Function	Description
Reset	Detects write stop and transmission synchronization.
Batch verify	Compares entire memory contents and input data.
Batch delete	Deletes the entire memory contents.
Batch blank check	Checks the deletion status of the entire memory.
High-speed write	Performs writing to flash memory according to write start address and number of write data (bytes).
Continuous write	Performs successive write operations using the data input with high-speed write operation.
Status	Checks the current operation mode and operation end.
Oscillation frequency setting	Inputs the resonator oscillation frequency information.
Delete time setting	Inputs the memory delete time.
Baud rate setting	Sets the transmission rate when the UART method is used.
Silicon signature read	Outputs the device name, memory capacity, and device block information.

21.3.4 Flash programmer connection

Connection of Flash programmer and µPD16F15A differs depending on communication method (3-wire serial I/O, UART). Each case of connection shows in Figures 21-4, 21-5 and 21-6.

Figure 21-4: Connection of Flash Programmer Using 3-Wire Serial I/O Method
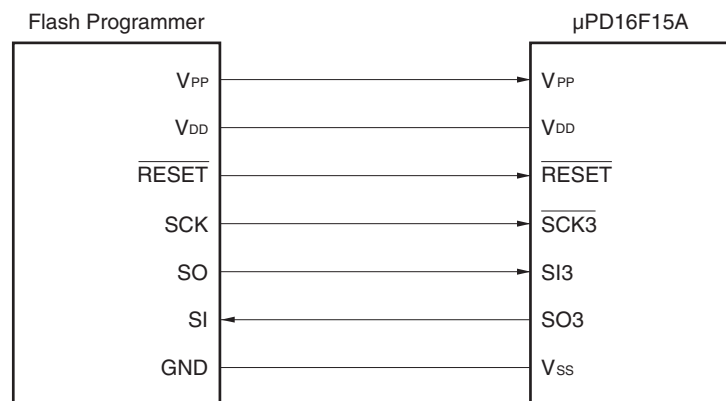
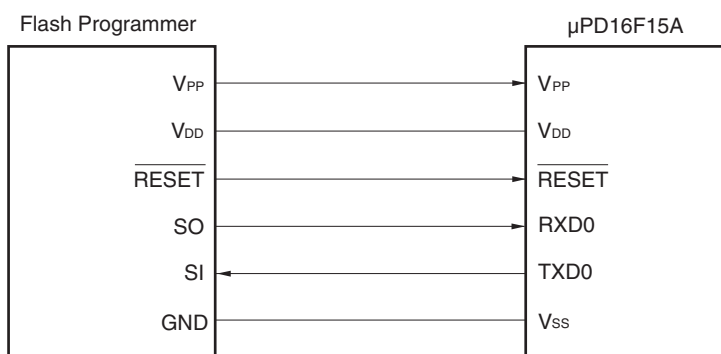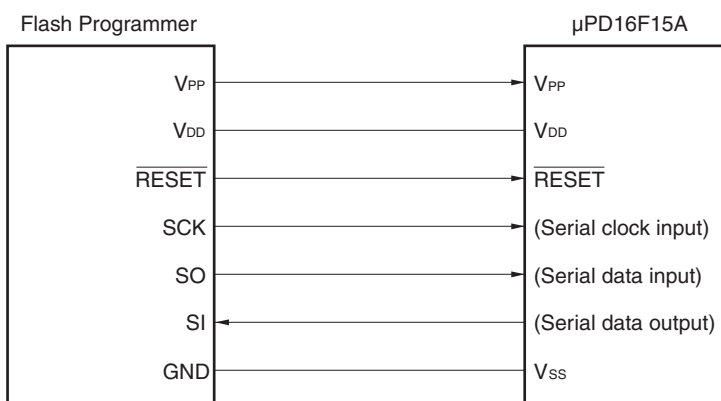*Figure 21-5: Flash Programmer Connection Using UART Method*

Flash Programmer   μPD16F15A

V$_{PP}$	→ V$_{PP}$
V$_{DD}$	→ V$_{DD}$
$\overline{RESET}$	→ $\overline{RESET}$
SO	→ RXD0
SI	← TXD0
GND	V$_{SS}$

Figure 21-6: Flash Programmer Connection Using Pseudo 3-wire Serial I/O

Flash Programmer   μPD16F15A

V$_{PP}$	→ V$_{PP}$
V$_{DD}$	→ V$_{DD}$
$\overline{RESET}$	→ $\overline{RESET}$
SCK	→ (Serial clock input)
SO	→ (Serial data input)
SI	← (Serial data output)
GND	V$_{SS}$

V$_{PP}$:   10.3 V applied from the on-board programming tool.

$\overline{RESET}$:   A $\overline{RESET}$ is generated and the device is set to the on-board programming mode.

System clock: The CPU clock for the device may be supplied by the on-board program tool. Alternatively the crystal or ceramic oscillator on the target H/W can be used in the on-board programming mode. The external system clock has to be connected with the X1 pin on the device.

V$_{DD}$:   The power supply for the device may be supplied by the on-board program tool. Alternatively the power supply on the target H/W can be used in the on-board programming mode.

GND: Ground level V$_{SS}$.

SCK: Serial clock generated by the on-board programming tool.

SI: Serial data sent by the on-board programming tool.

SO: Serial data sent by the device.

R$_X$D0:   Serial data sent by the on-board programming tool.

T$_X$D0:   Serial data sent by the device.

21.3.5 Flash programming precautions

• Please make sure that the signals used by the on-board programming tool do not conflict with other devices on the target H/W.

• A read functionality is not supported because of software protection. Only a verify operation of the whole Flash EPROM is supported. In verify mode data from start address to final address (EFFFH) has to be supplied by the programming tool. The device compares each data with on-chip flash content and replies with a signal for O.K. or not O.K.

[Memo]

Chapter 22 Instruction Set

This chapter describes each instruction set of the µPD1615A subseries as list table. For details of its operation and operation code, refer to the separate document **"78K/0 series USER'S MANUAL - Instruction (U12326E)**."

22.1 Legends Used in Operation List

22.1.1 Operand identifiers and description methods

Operands are described in "Operand" column of each instruction in accordance with the description method of the instruction operand identifier (refer to the assembler specifications for detail). When there are two or more description methods, select one of them. Alphabetic letters in capitals and symbols, #, !, $ and [ ] are key words and must be described as they are. Each symbol has the following meaning.

- # : Immediate data specification
- ! : Absolute address specification
- $: Relative address specification
- [ ] : Indirect address specification

In the case of immediate data, describe an appropriate numeric value or a label. When using a label, be sure to describe the #, !, $, and [ ] symbols.
For operand register identifiers, r and rp, either function names (X, A, C, etc.) or absolute names (names in parentheses in the table below, R0, R1, R2, etc.) can be used for description.

Table 22-1: Operand Identifiers and Description Methods

Identifier	Description Method
r rp sfr sfrp	X (R0), A (R1), C (R2), B (R3), E (R4), D (R5), L (R6), H (R7), AX (RP0), BC (RP1), DE (RP2), HL (RP3) Special-function register symbol[Note] Special-function register symbol (16-bit manipulatable register even addresses only)[Note]
saddr saddrp	FE20H-FF1FH Immediate data or labels FE20H-FF1FH Immediate data or labels (even address only)
addr16 addr11 addr5	0000H-FFFFH Immediate data or labels (Only even addresses for 16-bit data transfer instructions) 0800H-0FFFH Immediate data or labels 0040H-007FH Immediate data or labels (even address only)
word byte bit	16-bit immediate data or label 8-bit immediate data or label 3-bit immediate data or label
RBn	RB0 to RB3

Note: Addresses from FFD0H to FFDFH cannot be accessed with these operands.

Remark: For special-function register symbols, refer to "Table 3-3: Special-Function Register List".

22.1.2 Description of "operation" column

A	:	A register; 8-bit accumulator
X	:	X register
B	:	B register
C	:	C register
D	:	D register
E	:	E register
H	:	H register
L	:	L register
AX	:	AX register pair; 16-bit accumulator
BC	:	BC register pair
DE	:	DE register pair
HL	:	HL register pair
PC	:	Program counter
SP	:	Stack pointer
PSW	:	Program status word
CY	:	Carry flag
AC	:	Auxiliary carry flag
Z	:	Zero flag
RBS	:	Register bank select flag
IE	:	Interrupt request enable flag
NMIS	:	Non-maskable interrupt servicing flag
( )	:	Memory contents indicated by address or register contents in parentheses
X_H, X_L	:	Higher 8 bits and lower 8 bits of 16-bit register
$\wedge$	:	Logical product (AND)
$\vee$	:	Logical sum (OR)
$\veebar$	:	Exclusive logical sum (exclusive OR)
——	:	Inverted data
addr16	:	16-bit immediate data or label
jdisp8	:	Signed 8-bit data (displacement value)

22.1.3 Description of "flag operation" column

(Blank)	:	Not affected
0	:	Cleared to 0
1	:	Set to 1
X	:	Set/cleared according to the result
R	:	Previously saved value is restored

22.2 Operation List

Instruction Group	Mnemonic	Operands	Byte	Clock Note 1	Clock Note 2	Operation	Flag Z	Flag AC	Flag CY
8-bit data transfer	MOV	r, #byte	2	4	–	r ← byte			
		saddr, #byte	3	6	7	(saddr) ← byte			
		sfr, #byte	3	–	7	sfr ← byte			
		A, r Note 3	1	2	–	A ← r			
		r, A Note 3	1	2	–	r ← A			
		A, saddr	2	4	5	A ← (saddr)			
		saddr, A	2	4	5	(saddr) ← A			
		A, sfr	2	–	5	A ← sfr			
		sfr, A	2	–	5	sfr ← A			
		A, !addr16	3	8	9 + n	A ← (addr16)			
		!addr16, A	3	8	9 + m	(addr16) ← A			
		PSW, #byte	3	–	7	PSW ← byte	x	x	x
		A, PSW	2	–	5	A ← PSW			
		PSW, A	2	–	5	PSW ← A	x	x	x
		A, [DE]	1	4	5 + n	A ← (DE)			
		[DE], A	1	4	5 + m	(DE) ← A			
		A, [HL]	1	4	5 + n	A ← (HL)			
		[HL], A	1	4	5 + m	(HL) ← A			
		A, [HL + byte]	2	8	9 + n	A ← (HL + byte)			
		[HL + byte], A	2	8	9 + m	(HL + byte) ← A			
		A, [HL + B]	1	6	7 + n	A ← (HL + B)			
		[HL + B], A	1	6	7 + m	(HL + B) ← A			
		A, [HL + C]	1	6	7 + n	A ← (HL + C)			
		[HL + C], A	1	6	7 + m	(HL + C) ← A			
	XCH	A, r Note 3	1	2	–	A ↔ r			
		A, saddr	2	4	6	A ↔ (saddr)			
		A, sfr	2	–	6	A ↔ (sfr)			
		A, !addr16	3	8	10 + n + m	A ↔ (addr16)			
		A, [DE]	1	4	6 + n + m	A ↔ (DE)			
		A, [HL]	1	4	6 + n + m	A ↔ (HL)			
		A, [HL + byte]	2	8	10 + n + m	A ↔ (HL + byte)			
		A, [HL + B]	2	8	10 + n + m	A ↔ (HL + B)			
		A, [HL + C]	2	8	10 + n + m	A ↔ (HL + C)			

Notes: 1. When the internal high-speed RAM area is accessed or instruction with no data access
2. When an area except the internal high-speed RAM area is accessed.
3. Except "r = A"

Remarks: 1. One instruction clock cycle is one cycle of the CPU clock (fCPU) selected by the PCC register.
2. This clock cycle applies to internal ROM program.
3. n is the number of waits when external memory expansion area is read from.
4. m is the number of waits when external memory expansion area is written to.

Instruction Group	Mnemonic	Operands	Byte	Clock Note 1	Clock Note 2	Operation	Flag Z	AC	CY
16-bit data transfer	MOVW	rp, #word	3	6	–	rp ← word			
		saddrp, #word	4	8	10	(saddrp) ← word			
		sfrp, #word	4	–	10	sfrp ← word			
		AX, saddrp	2	6	8	AX ← (saddrp)			
		saddrp, AX	2	6	8	(saddrp) ← AX			
		AX, sfrp	2	–	8	AX ← sfrp			
		sfrp, AX	2	–	8	sfrp ← AX			
		AX, rp Note 3	1	4	–	AX ← rp			
		rp, AX Note 3	1	4	–	rp ← AX			
		AX, !addr16	3	10	12 + 2n	AX ← (addr16)			
		!addr16, AX	3	10	12 + 2m	(addr16) ← AX			
	XCHW	AX, rp Note 3	1	4	–	AX × rp			
8-bit operation	ADD	A, #byte	2	4	–	A, CY ← A + byte	x	x	x
		saddr, #byte	3	6	8	(saddr), CY ← (saddr) + byte	x	x	x
		A, r Note 4	2	4	–	A, CY ← A + r	x	x	x
		r, A	2	4	–	r, CY ← r + A	x	x	x
		A, saddr	2	4	5	A, CY ← A + (saddr)	x	x	x
		A, !addr16	3	8	9 + n	A, CY ← A + (addr16)	x	x	x
		A, [HL]	1	4	5 + n	A, CY ← A + (HL)	x	x	x
		A, [HL + byte]	2	8	9 + n	A, CY ← A + (HL + byte)	x	x	x
		A, [HL + B]	2	8	9 + n	A, CY ← A + (HL + B)	x	x	x
		A, [HL + C]	2	8	9 + n	A, CY ← A + (HL + C)	x	x	x
	ADDC	A, #byte	2	4	–	A, CY ← A + byte + CY	x	x	x
		saddr, #byte	3	6	8	(saddr), CY ← (saddr) + byte + CY	x	x	x
		A, r Note 4	2	4	–	A, CY ← A + r + CY	x	x	x
		r, A	2	4	–	r, CY ← r + A + CY	x	x	x
		A, saddr	2	4	5	A, CY ← A + (saddr) + CY	x	x	x
		A, !addr16	3	8	9 + n	A, CY ← A + (addr16) + CY	x	x	x
		A, [HL]	1	4	5 + n	A, CY ← A + (HL) + CY	x	x	x
		A, [HL + byte]	2	8	9 + n	A, CY ← A + (HL + byte) + CY	x	x	x
		A, [HL + B]	2	8	9 + n	A, CY ← A + (HL + B) + CY	x	x	x
		A, [HL + C]	2	8	9 + n	A, CY ← A + (HL + C) + CY	x	x	x

Notes: 1. When the internal high-speed RAM area is accessed or instruction with no data access
2. When an area except the internal high-speed RAM area is accessed
3. Only when rp = BC, DE or HL
4. Except "r = A"

Remarks: 1. One instruction clock cycle is one cycle of the CPU clock (f_{CPU}) selected by the PCC register.
2. This clock cycle applies to internal ROM program.
3. n is the number of waits when external memory expansion area is read from.
4. m is the number of waits when external memory expansion area is written to.

Instruction Group	Mnemonic	Operands	Byte	Clock Note 1	Clock Note 2	Operation	Flag Z	Flag AC	Flag CY
8-bit operation	SUB	A, #byte	2	4	–	A, CY ← A – byte	x	x	x
		saddr, #byte	3	6	8	(saddr), CY ← (saddr) – byte	x	x	x
		A, r ^Note 3	2	4	–	A, CY ← A – r	x	x	x
		r, A	2	4	–	r, CY ← r – A	x	x	x
		A, saddr	2	4	5	A, CY ← A – (saddr)	x	x	x
		A, !addr16	3	8	9 + n	A, CY ← A – (addr16)	x	x	x
		A, [HL]	1	4	5 + n	A, CY ← A – (HL)	x	x	x
		A, [HL + byte]	2	8	9 + n	A, CY ← A – (HL + byte)	x	x	x
		A, [HL + B]	2	8	9 + n	A, CY ← A – (HL + B)	x	x	x
		A, [HL + C]	2	8	9 + n	A, CY ← A – (HL + C)	x	x	x
	SUBC	A, #byte	2	4	–	A, CY ← A – byte – CY	x	x	x
		saddr, #byte	3	6	8	(saddr), CY ← (saddr) – byte – CY	x	x	x
		A, r ^Note 3	2	4	–	A, CY ← A – r – CY	x	x	x
		r, A	2	4	–	r, CY ← r – A – CY	x	x	x
		A, saddr	2	4	5	A, CY ← A – (saddr) – CY	x	x	x
		A, !addr16	3	8	9 + n	A, CY ← A – (addr16) – CY	x	x	x
		A, [HL]	1	4	5 + n	A, CY ← A – (HL) – CY	x	x	x
		A, [HL + byte]	2	8	9 + n	A, CY ← A – (HL + byte) – CY	x	x	x
		A, [HL + B]	2	8	9 + n	A, CY ← A – (HL + B) – CY	x	x	x
		A, [HL + C]	2	8	9 + n	A, CY ← A – (HL + C) – CY	x	x	x
	AND	A, #byte	2	4	–	A ← A ∧ byte	x		
		saddr, #byte	3	6	8	(saddr) ← (saddr) ∧ byte	x		
		A, r ^Note 3	2	4	–	A ← A ∧ r	x		
		r, A	2	4	–	r ← r ∧ A	x		
		A, saddr	2	4	5	A ← A ∧ (saddr)	x		
		A, !addr16	3	8	9 + n	A ← A ∧ (addr16)	x		
		A, [HL]	1	4	5 + n	A ← A ∧ [HL]	x		
		A, [HL + byte]	2	8	9 + n	A ← A ∧ [HL + byte]	x		
		A, [HL + B]	2	8	9 + n	A ← A ∧ [HL + B]	x		
		A, [HL + C]	2	8	9 + n	A ← A ∧ [HL + C]	x		

Notes: **1.** When the internal high-speed RAM area is accessed or instruction with no data access
2. When an area except the internal high-speed RAM area is accessed
3. Except "r = A"

Remarks: **1.** One instruction clock cycle is one cycle of the CPU clock (fCPU) selected by the PCC register.
2. This clock cycle applies to internal ROM program.
3. n is the number of waits when external memory expansion area is read from.

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
8-bit operation	OR	A, #byte	2	4	–	A ← A∨byte	x		
		saddr, #byte	3	6	8	(saddr) ← (saddr)∨byte	x		
		A, r Note 3	2	4	–	A ← A∨r	x		
		r, A	2	4	–	r ← r∨A	x		
		A, saddr	2	4	5	A ← A∨(saddr)	x		
		A, !addr16	3	8	9 + n	A ← A∨(addr16)	x		
		A, [HL]	1	4	5 + n	A ← A∨(HL)	x		
		A, [HL + byte]	2	8	9 + n	A ← A∨(HL + byte)	x		
		A, [HL + B]	2	8	9 + n	A ← A∨(HL + B)	x		
		A, [HL + C]	2	8	9 + n	A ← A∨(HL + C)	x		
	XOR	A, #byte	2	4	–	A ← A∀byte	x		
		saddr, #byte	3	6	8	(saddr) ← (saddr)∀byte	x		
		A, r Note 3	2	4	–	A ← A∀r	x		
		r, A	2	4	–	r ← r∀A	x		
		A, saddr	2	4	5	A ← A∀(saddr)	x		
		A, !addr16	3	8	9 + n	A ← A∀(addr16)	x		
		A, [HL]	1	4	5 + n	A ← A∀(HL)	x		
		A, [HL + byte]	2	8	9 + n	A ← A∀(HL + byte)	x		
		A, [HL + B]	2	8	9 + n	A ← A∀(HL + B)	x		
		A, [HL + C]	2	8	9 + n	A ← A∀(HL + C)	x		
	CMP	A, #byte	2	4	–	A – byte	x	x	x
		saddr, #byte	3	6	8	(saddr) – byte	x	x	x
		A, r Note 3	2	4	–	A – r	x	x	x
		r, A	2	4	–	r – A	x	x	x
		A, saddr	2	4	5	A – (saddr)	x	x	x
		A, !addr16	3	8	9 + n	A – (addr16)	x	x	x
		A, [HL]	1	4	5 + n	A – (HL)	x	x	x
		A, [HL + byte]	2	8	9 + n	A – (HL + byte)	x	x	x
		A, [HL + B]	2	8	9 + n	A – (HL + B)	x	x	x
		A, [HL + C]	2	8	9 + n	A – (HL + C)	x	x	x

Notes: **1.** When the internal high-speed RAM area is accessed or instruction with no data access
2. When an area except the internal high-speed RAM area is accessed
3. Except "r = A"

Remarks: **1.** One instruction clock cycle is one cycle of the CPU clock (fCPU) selected by the PCC register.
2. This clock cycle applies to internal ROM program.
3. n is the number of waits when external memory expansion area is read from.

Instruction Group	Mnemonic	Operands	Byte			Operation			
16-bit operation	ADDW	AX, #word	3	6	–	AX, CY ← AX + word	x	x	x
	SUBW	AX, #word	3	6	–	AX, CY ← AX – word	x	x	x
	CMPW	AX, #word	3	6	–	AX – word	x	x	x
Multiply/ divide	MULU	X	2	16	–	AX ← A x X			
	DIVUW	C	2	25	–	AX (Quotient), C (Remainder) ← AX ÷ C			
Increment/ decrement	INC	r	1	2	–	r ← r + 1	x	x	
		saddr	2	4	6	(saddr) ← (saddr) + 1	x	x	
	DEC	r	1	2	–	r ← r – 1	x	x	
		saddr	2	4	6	(saddr) ← (saddr) – 1	x	x	
	INCW	rp	1	4	–	rp ← rp + 1			
	DECW	rp	1	4	–	rp ← rp – 1			
Rotate	ROR	A, 1	1	2	–	(CY, $A_7 \leftarrow A_0$, $A_{m-1} \leftarrow A_m$) x 1 time			x
	ROL	A, 1	1	2	–	(CY, $A_0 \leftarrow A_7$, $A_{m+1} \leftarrow A_m$) x 1 time			x
	RORC	A, 1	1	2	–	(CY ← A_0, A_7 ← CY, $A_{m-1} \leftarrow A_m$) x 1 time			x
	ROLC	A, 1	1	2	–	(CY ← A_7, A_0 ← CY, $A_{m+1} \leftarrow A_m$) x 1 time			x
	ROR4	[HL]	2	10	12 + n + m	$A_{3-0} \leftarrow (HL)_{3-0}$, $(HL)_{7-4} \leftarrow A_{3-0}$, $(HL)_{3-0} \leftarrow (HL)_{7-4}$			
	ROL4	[HL]	2	10	12 + n + m	$A_{3-0} \leftarrow (HL)_{7-4}$, $(HL)_{3-0} \leftarrow A_{3-0}$, $(HL)_{7-4} \leftarrow (HL)_{3-0}$			
BCD adjust	ADJBA		2	4	–	Decimal Adjust Accumulator after Addition	x	x	x
	ADJBS		2	4	–	Decimal Adjust Accumulator after Subtract	x	x	x
Bit manipulate	MOV1	CY, saddr.bit	3	6	7	CY ← (saddr.bit)			x
		CY, sfr.bit	3	–	7	CY ← sfr.bit			x
		CY, A.bit	2	4	–	CY ← A.bit			x
		CY, PSW.bit	3	–	7	CY ← PSW.bit			x
		CY, [HL].bit	2	6	7 + n	CY ← (HL).bit			x
		saddr.bit, CY	3	6	8	(saddr.bit) ← CY			
		sfr.bit, CY	3	–	8	sfr.bit ← CY			
		A.bit, CY	2	4	–	A.bit ← CY			
		PSW.bit, CY	3	–	8	PSW.bit ← CY	x	x	
		[HL].bit, CY	2	6	8 + n + m	(HL).bit ← CY			

Notes: 1. When the internal high-speed RAM area is accessed or instruction with no data access
2. When an area except the internal high-speed RAM area is accessed

Remarks: 1. One instruction clock cycle is one cycle of the CPU clock (f_{CPU}) selected by the PCC register.
2. This clock cycle applies to internal ROM program.
3. n is the number of waits when external memory expansion area is read from.
4. m is the number of waits when external memory expansion area is written to.

Instruction Group	Mnemonic	Operands	Byte	Clock		Operation	Flag		
				Note 1	Note 2		Z	AC	CY
Bit manipu-late	AND1	CY, saddr.bit	3	6	7	CY ← CY ∧ (saddr.bit)			x
		CY, sfr.bit	3	–	7	CY ← CY ∧ sfr.bit			x
		CY, A.bit	2	4	–	CY ← CY ∧ A.bit			x
		CY, PSW.bit	3	–	7	CY ← CY ∧ PSW.bit			x
		CY, [HL].bit	2	6	7 + n	CY ← CY ∧ (HL).bit			x
	OR1	CY, saddr.bit	3	6	7	CY ← CY ∨ (saddr.bit)			x
		CY, sfr.bit	3	–	7	CY ← CY ∨ sfr.bit			x
		CY, A.bit	2	4	–	CY ← CY ∨ A.bit			x
		CY, PSW.bit	3	–	7	CY ← CY ∨ PSW.bit			x
		CY, [HL].bit	2	6	7 + n	CY ← CY ∨ (HL).bit			x
	XOR1	CY, saddr.bit	3	6	7	CY ← CY ⊻ (saddr.bit)			x
		CY, sfr.bit	3	–	7	CY ← CY ⊻ sfr.bit			x
		CY, A.bit	2	4	–	CY ← CY ⊻ A.bit			x
		CY, PSW. bit	3	–	7	CY ← CY ⊻ PSW.bit			x
		CY, [HL].bit	2	6	7 + n	CY ← CY ⊻ (HL).bit			x
	SET1	saddr.bit	2	4	6	(saddr.bit) ← 1			
		sfr.bit	3	–	8	sfr.bit ← 1			
		A.bit	2	4	–	A.bit ← 1			
		PSW.bit	2	–	6	PSW.bit ← 1	x	x	x
		[HL].bit	2	6	8 + n + m	(HL).bit ← 1			
	CLR1	saddr.bit	2	4	6	(saddr.bit) ← 0			
		sfr.bit	3	–	8	sfr.bit ← 0			
		A.bit	2	4	–	A.bit ← 0			
		PSW.bit	2	–	6	PSW.bit ← 0	x	x	x
		[HL].bit	2	6	8 + n + m	(HL).bit ← 0			
	SET1	CY	1	2	–	CY ← 1			1
	CLR1	CY	1	2	–	CY ← 0			0
	NOT1	CY	1	2	–	CY ← $\overline{CY}$			x

Notes: **1.** When the internal high-speed RAM area is accessed or instruction with no data access
          2. When an area except the internal high-speed RAM area is accessed

Remarks: **1.** One instruction clock cycle is one cycle of the CPU clock (f$_{CPU}$) selected by the PCC register.
            2. This clock cycle applies to internal ROM program.
            3. n is the number of waits when external memory expansion area is read from.
            4. m is the number of waits when external memory expansion area is written to.

Instruction Group	Mnemonic	Operands	Byte	Clock Note 1	Clock Note 2	Operation	Flag Z	Flag AC	Flag CY
Call/return	CALL	!addr16	3	7	–	$(SP-1) \leftarrow (PC+3)_H, (SP-2) \leftarrow (PC+3)_L$, $PC \leftarrow addr16, SP \leftarrow SP - 2$			
	CALLF	!addr11	2	5	–	$(SP - 1) \leftarrow (PC + 2)_H, (SP - 2) \leftarrow (PC + 2)_L$, $PC_{15-11} \leftarrow 00001, PC_{10-0} \leftarrow addr11$, $SP \leftarrow SP - 2$			
	CALLT	[addr5]	1	6	–	$(SP - 1) \leftarrow (PC + 1)_H, (SP - 2) \leftarrow (PC + 1)_L$, $PC_H \leftarrow (00000000, addr5 + 1)$, $PC_L \leftarrow (00000000, addr5)$, $SP \leftarrow SP - 2$			
	BRK		1	6	–	$(SP - 1) \leftarrow PSW, (SP - 2) \leftarrow (PC + 1)_H$, $(SP - 3) \leftarrow (PC + 1)_L, PC_H \leftarrow (003FH)$, $PC_L \leftarrow (003EH), SP \leftarrow SP - 3, IE \leftarrow 0$			
	RET		1	6	–	$PC_H \leftarrow (SP + 1), PC_L \leftarrow (SP)$, $SP \leftarrow SP + 2$			
	RETI		1	6	–	$PC_H \leftarrow (SP + 1), PC_L \leftarrow (SP)$, $PSW \leftarrow (SP + 2), SP \leftarrow SP + 3$, $NMIS \leftarrow 0$	R	R	R
	RETB		1	6	–	$PC_H \leftarrow (SP + 1), PC_L \leftarrow (SP)$, $PSW \leftarrow (SP + 2), SP \leftarrow SP + 3$	R	R	R
Stack manipulate	PUSH	PSW	1	2	–	$(SP - 1) \leftarrow PSW, SP \leftarrow SP - 1$			
		rp	1	4	–	$(SP - 1) \leftarrow rp_H, (SP - 2) \leftarrow rp_L$, $SP \leftarrow SP - 2$			
	POP	PSW	1	2	–	$PSW \leftarrow (SP), SP \leftarrow SP + 1$	R	R	R
		rp	1	4	–	$rp_H \leftarrow (SP + 1), rp_L \leftarrow (SP)$, $SP \leftarrow SP + 2$			
	MOVW	SP, #word	4	–	10	$SP \leftarrow word$			
		SP, AX	2	–	8	$SP \leftarrow AX$			
		AX, SP	2	–	8	$AX \leftarrow SP$			
Unconditional branch	BR	!addr16	3	6	–	$PC \leftarrow addr16$			
		$addr16	2	6	–	$PC \leftarrow PC + 2 + jdisp8$			
		AX	2	8	–	$PC_H \leftarrow A, PC_L \leftarrow X$			
Conditional branch	BC	$addr16	2	6	–	$PC \leftarrow PC + 2 + jdisp8$ if $CY = 1$			
	BNC	$addr16	2	6	–	$PC \leftarrow PC + 2 + jdisp8$ if $CY = 0$			
	BZ	$addr16	2	6	–	$PC \leftarrow PC + 2 + jdisp8$ if $Z = 1$			
	BNZ	$addr16	2	6	–	$PC \leftarrow PC + 2 + jdisp8$ if $Z = 0$			

Notes: 1. When the internal high-speed RAM area is accessed or instruction with no data access
   2. When an area except the internal high-speed RAM area is accessed

Remarks: 1. One instruction clock cycle is one cycle of the CPU clock (f_{CPU}) selected by the PCC register.
   2. This clock cycle applies to internal ROM program.

Instruction Group	Mnemonic	Operands	Byte			Operation	
Conditional branch	BT	saddr.bit, $addr16	3	8	9	PC ← PC + 3 + jdisp8 if(saddr.bit) = 1	
		sfr.bit, $addr16	4	–	11	PC ← PC + 4 + jdisp8 if sfr.bit = 1	
		A.bit, $addr16	3	8	–	PC ← PC + 3 + jdisp8 if A.bit = 1	
		PSW.bit, $addr16	3	–	9	PC ← PC + 3 + jdisp8 if PSW.bit = 1	
		[HL].bit, $addr16	3	10	11 + n	PC ← PC + 3 + jdisp8 if (HL).bit = 1	
	BF	saddr.bit, $addr16	4	10	11	PC ← PC + 4 + jdisp8 if(saddr.bit) = 0	
		sfr.bit, $addr16	4	–	11	PC ← PC + 4 + jdisp8 if sfr.bit = 0	
		A.bit, $addr16	3	8	–	PC ← PC + 3 + jdisp8 if A.bit = 0	
		PSW.bit, $addr16	4	–	11	PC ← PC + 4 + jdisp8 if PSW. bit = 0	
		[HL].bit, $addr16	3	10	11 + n	PC ← PC + 3 + jdisp8 if (HL).bit = 0	
	BTCLR	saddr.bit, $addr16	4	10	12	PC ← PC + 4 + jdisp8 if(saddr.bit) = 1 then reset(saddr.bit)	
		sfr.bit, $addr16	4	–	12	PC ← PC + 4 + jdisp8 if sfr.bit = 1 then reset sfr.bit	
		A.bit, $addr16	3	8	–	PC ← PC + 3 + jdisp8 if A.bit = 1 then reset A.bit	
		PSW.bit, $addr16	4	–	12	PC ← PC + 4 + jdisp8 if PSW.bit = 1 then reset PSW.bit	x x x
		[HL].bit, $addr16	3	10	12 + n + m	PC ← PC + 3 + jdisp8 if (HL).bit = 1 then reset (HL).bit	
	DBNZ	B, $addr16	2	6	–	B ← B – 1, then PC ← PC + 2 + jdisp8 if B ≠ 0	
		C, $addr16	2	6	–	C ← C –1, then PC ← PC + 2 + jdisp8 if C ≠ 0	
		saddr. $addr16	3	8	10	(saddr) ← (saddr) – 1, then PC ← PC + 3 + jdisp8 if(saddr) ≠ 0	
CPU control	SEL	RBn	2	4	–	RBS1, 0 ← n	
	NOP		1	2	–	No Operation	
	EI		2	–	6	IE ← 1(Enable Interrupt)	
	DI		2	–	6	IE ← 0(Disable Interrupt)	
	HALT		2	6	–	Set HALT Mode	
	STOP		2	6	–	Set STOP Mode	

Notes: **1.** When the internal high-speed RAM area is accessed or instruction with no data access
2. When an area except the internal high-speed RAM area is accessed

Remarks: **1.** One instruction clock cycle is one cycle of the CPU clock (fCPU) selected by the PCC register.
2. This clock cycle applies to internal ROM program.
3. n is the number of waits when external memory expansion area is read from.
4. m is the number of waits when external memory expansion area is written to.

22.3 Instructions Listed by Addressing Type

(1) 8-bit instructions

MOV, XCH, ADD, ADDC, SUB, SUBC, AND, OR, XOR, CMP, MULU, DIVUW, INC, DEC, ROR, ROL, RORC, ROLC, ROR4, ROL4, PUSH, POP, DBNZ

Second Operand / First Operand	#byte	A	r^Note	sfr	saddr	!addr16	PSW	[DE]	[HL]	[HL + byte] [HL + B] [HL + C]	$addr16	1	None
A	ADD ADDC SUB SUBC AND OR XOR CMP		MOV XCH ADD ADDC SUB SUBC AND OR XOR CMP	MOV XCH	MOV XCH ADD ADDC SUB SUBC AND OR XOR CMP	MOV XCH ADD ADDC SUB SUBC AND OR XOR CMP	MOV	MOV XCH	MOV XCH ADD ADDC SUB SUBC AND OR XOR CMP	MOV XCH ADD ADDC SUB SUBC AND OR XOR CMP		ROR ROL RORC ROLC	
r	MOV	MOV ADD ADDC SUB SUBC AND OR XOR CMP											INC DEC
B, C											DBNZ		
sfr	MOV	MOV											
saddr	MOV ADD ADDC SUB SUBC AND OR XOR CMP	MOV									DBNZ		INC DEC
!addr16		MOV											
PSW	MOV	MOV											PUSH POP
[DE]		MOV											
[HL]		MOV											ROR4 ROL4
[HL + byte] [HL + B] [HL + C]		MOV											
X													MULU
C													DIVUW

Note: Except r = A

(2) 16-bit instructions

MOVW, XCHW, ADDW, SUBW, CMPW, PUSH, POP, INCW, DECW

Second Operand / 1st Operand	#word	AX	rp**Note**	sfrp	saddrp	!addr16	SP	None
AX	ADDW SUBW CMPW		MOVW XCHW	MOVW	MOVW	MOVW	MOVW	
rp	MOVW	MOVW**Note**						INCW DECW PUSH POP
sfrp	MOVW	MOVW						
saddrp	MOVW	MOVW						
!addr16		MOVW						
SP	MOVW	MOVW						

Note: Only when rp = BC, DE, HL

(3) Bit manipulation instructions

MOV1, AND1, OR1, XOR1, SET1, CLR1, NOT1, BT, BF, BTCLR

Second Operand / First Operand	A.bit	sfr.bit	saddr.bit	PSW.bit	[HL].bit	CY	$addr16	None
A.bit						MOV1	BT BF BTCLR	SET1 CLR1
sfr.bit						MOV1	BT BF BTCLR	SET1 CLR1
saddr.bit						MOV1	BT BF BTCLR	SET1 CLR1
PSW.bit						MOV1	BT BF BTCLR	SET1 CLR1
[HL].bit						MOV1	BT BF BTCLR	SET1 CLR1
CY	MOV1 AND1 OR1 XOR1	MOV1 AND1 OR1 XOR1	MOV1 AND1 OR1 XOR1	MOV1 AND1 OR1 XOR1	MOV1 AND1 OR1 XOR1			SET1 CLR1 NOT1

(4) Call/instructions/branch instructions

CALL, CALLF, CALLT, BR, BC, BNC, BZ, BNZ, BT, BF, BTCLR, DBNZ

Second Operand First Operand	AX	!addr16	!addr11	[addr5]	$addr16
Basic instruction	BR	CALL BR	CALLF	CALLT	BR BC BNC BZ BNZ
Compound instruction					BT BF BTCLR DBNZ

(5) Other instructions

ADJBA, ADJBS, BRK, RET, RETI, RETB, SEL, NOP, EI, DI, HALT, STOP

[Memo]

[Memo]

Appendix A Development Tools

The following development tools are available for the development of systems that employ the µPD1615A subseries.

- **Support for PC98-NX series**
  Unless otherwise specified, products compatible with IBM PC/ATTM computers are compatible with PC98-NX series computers. When using PC98-NX series computers, refer to the explanation for IBM PC/AT computers.

- **Windows**
  Unless otherwise specified, "Windows" means the following OSs.
   - Windows 95
   - Windows NT^{TM} Ver 4.0
   - Windows 2000

Figure A-1 shows the development tool configuration.

Figure A-1: Development Tool Configuration

When using the in-circuit emulator IE-78K0-NS-A
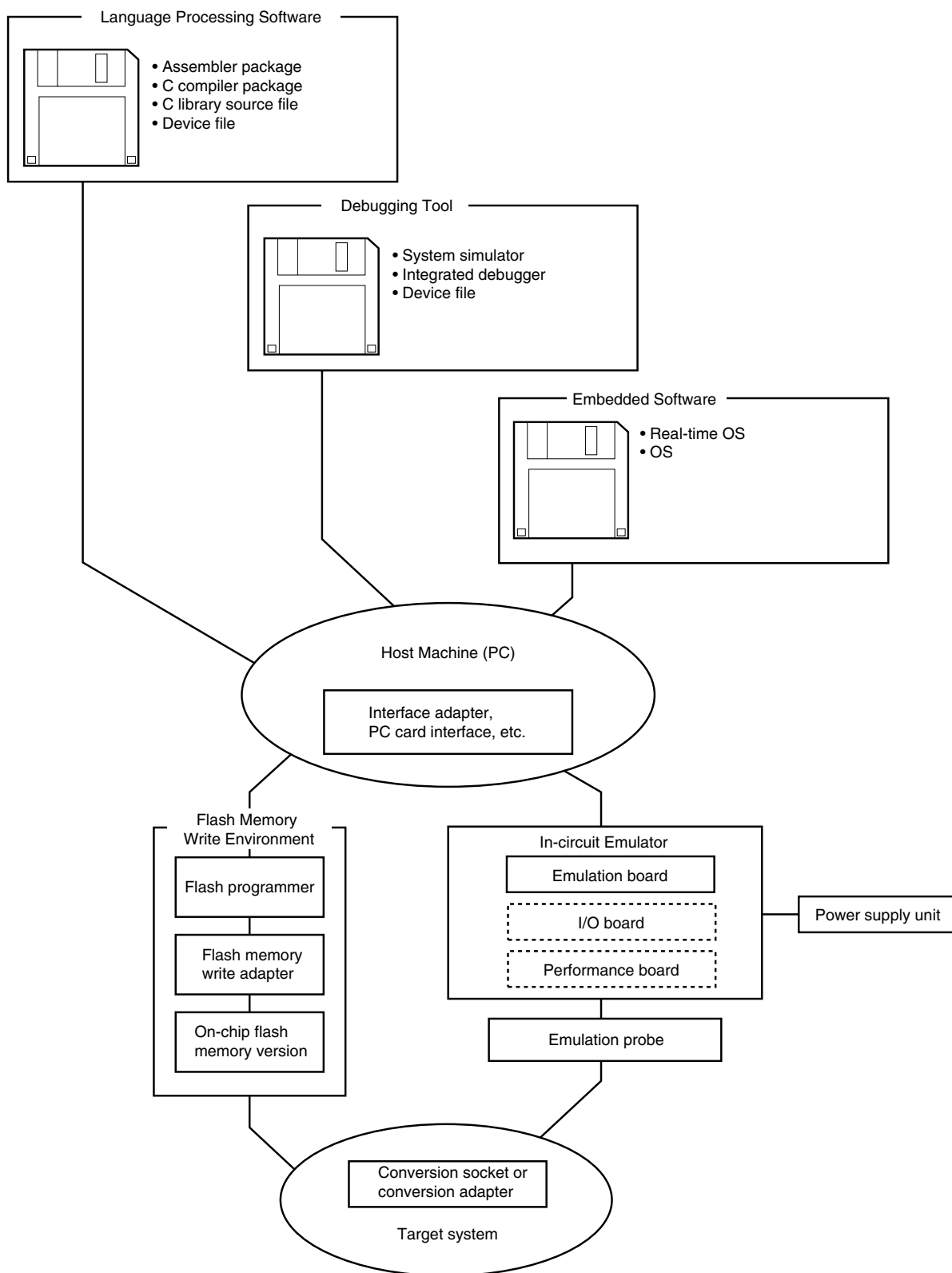
Remark: Items in broken line boxes differ according to the development environment. See B.3.1 Hardware.

A.1 Language Processing Software

RA78K/0 Assembler Package	This assembler converts programs written in mnemonics into an object code executable with a microcomputer. Further, this assembler is provided with functions capable of automatically creating symbol tables and branch instruction optimization. This assembler is used in combination with an optional device file. **<Precaution when using RA78K/0 in PC environment>** This assembler package is a DOS-based application. It can also be used in Windows, however, by using the Project Manager (included in assembler package) on Windows.
CC78K/0 C Compiler Package	This compiler converts programs written in C language into object code executable with a microcomputer. This compiler is used in combination with an optional assembler package and device file. <Precaution when using CC78K/0 in PC environment> This C compiler package is a DOS-based application. It can also be used in Windows, however, by using the Project Manager (included in assembler package) on Windows.
Device File	This file contains information peculiar to the device. This device file should be used in combination with an optional tool (RA78K/0, CC78K/0, SM78K0, ID78K0-NS, and ID78K0). Corresponding OS and host machine differ depending on the tool to be used with.
CC78K/0-L C Library Source File	This is a source file of functions configuring the object library included in the C compiler package (CC78K/0). It is required to match the object library included in C compiler package to the customer's specifications.

IAR Software

A78000	Assembler package used for the 78K0 series
ICC78000	C compiler package used for the 78K0 series
XLINK	Linker package used for the 78K0 series

A.2 Flash Memory Writing Tools

FlashMASTER Flashpro III (part number: FL-PR3, PG-FP3) Flash Programmer	Flash programmer dedicated to microcontrollers with on-chip flash memory.
FA-80GC Flash Memory Writing Adapter	Flash memory writing adapter used connected to the Flash Programmer • FA-80GC : 64-pin plastic QFP (GC-8BT type)

Note: Under development

Remark: FL-PR2, FL-PR3, FA-64CW, FA-64GC, and FA-64GK are products of Naito Densei Machida Mfg. Co., Ltd.
Phone: (044) 822-3813 Naito Densei Machida Mfg. Co., Ltd.

A.3 Debugging Tools

A.3.1 Hardware

(1) When using the in-circuit emulator IE-78K0-NS-A

IE-780K0-NS-A In-Circuit Emulator	In-circuit emulator serves to debug hardware and software when developing application systems using the 78K/0 Series product. It corresponds to integrated debugger ID78K0-NS. This emulator is used in combination with power supply unit, emulation probe, and interface adapter which is required to connect this emulator to the host machine.
IE-70000-MC-PS-B Power Supply Unit	This adapter is used for supplying power from a receptacle of 100-V to 240-V AC.
IE-70000-98-IF-C Interface Adapter	This adapter is required when using the PC-9800 series computer (except notebook type) as the IE-78K0-NS-A host machine (C bus compatible).
IE-70000-CD-IF-A PC Card Interface	This is PC card and interface cable required when using notebook type computer as the IE-78K0-NS-A host machine (PCMCIA socket compatible).
IE-70000-PC-IF-C Interface Adapter	This adapter is required when using the IBM PC compatible computers as the IE-78K0-NS-A host machine (ISA bus compatible).
IE-70000-PCI-IF-A Interface Adapter	This adapter is required when using a computer with PCI bus as the IE-78K0-NS-A host machine.
IE-78K0-NS-P04 Emulation Board	This board emulates the operations of the peripheral hardware peculiar to a device. It should be used in combination with an in-circuit emulator.
IE-1615-NS-EM4 Probe Board	This probe is used to connect the in-circuit emulator to a target system.
NP-80GC-TQ Emulation Probe	This probe is used to connect the in-circuit emulator to a target system and is designed for use with 80-pin plastic QFP (GC-8BT type).
NQPACK080SB HQPACK080SB YQPACK080SB YQSOCKET080SBF	This conversion socket connects the NP-80GC-TQ to a target system board designed for a 80-pin plastic QFP (GC-8BT type).

(2) Socket Details

NQPACK080SB	Socket for soldering on the target
YQPACK080SB	Adapter socket for connecting the probe to the NQPACK080SB
HQPACK080SB	Lid socket for connecting the device to the NQPACK080SB
YQSOCKET080SBF	Height adapter between the YQPACK080SB and the probe

A.3.2 Software

SM78K0 System Simulator	This system simulator is used to perform debugging at C source level or assembler level while simulating the operation of the target system on a host machine. This simulator runs on Windows. Use of the SM78K0 allows the execution of application logical testing and performance testing on an independent basis from hardware development without having to use an in-circuit emulator, thereby providing higher development efficiency and software quality. The SM78K0 should be used in combination with the optional device file .
ID78K0-NS Integrated Debugger (supporting in-circuit emulator IE-78K0- NS-A)	This debugger is a control program to debug 78K/0 Series microcontrollers. It adopts a graphical user interface, which is equivalent visually and operationally to Windows or OSF/Motif™. It also has an enhanced debugging function for C language programs, and thus trace results can be displayed on screen in C-language level by using the windows integration function which links a trace result with its source program, disassembled display, and memory display. In addition, by incorporating function modules such as task debugger and system performance analyzer, the efficiency of debugging programs, which run on real-time OSs can be improved. It should be used in combination with the optional device file.

[Memo]

Appendix B Embedded Software

For efficient development and maintenance of the µPD1615A Subseries, the following embedded software products are available.

B.1 Real-Time OS

RX78K/0 Real-time OS	RX78K/0 is a real-time OS conforming to the ITRON specifications. Tool (configurator) for generating nucleus of RX78K/0 and plural information tables is supplied. Used in combination with an optional assembler package (RA78K/0) and device file. **\<Precaution when using RX78K/0 in PC environment>** The real-time OS is a DOS-based application. It should be used in the DOS Prompt when using in Windows.

Caution: When purchasing the RX78K/0, fill in the purchase application form in advance and sign the user agreement.

MX78K0 OS	MX78K0 is an OS for ITRON specification subsets. A nucleus for the MX78K0 is also included as a companion product. This manages tasks, events, and time. In the task management, determining the task execution order and switching from task to the next task are performed. **\<Precaution when using MX78K0 in PC environment>** The MX78K0 is a DOS-based application. It should be used in the DOS Prompt when using in Windows.

[Memo]

Appendix C Register Index

C.1 Register Index (In Alphabetical Order with Respect to Register Names)

C.2 Register Index (In Alphabetical Order with Respect to Register Symbol)

ADCR1   : A/D conversion result register 1
ADM1   : A/D converter mode register
ADS1   : Analog input channel specification register
ASIM0   : Asynchronous serial interface mode register
ASIS0   : Asynchronous serial interface status register
BRGC0 : Baud rate generator control register
CKS   : Clock output selection register
CR00   : Capture/compare register 00
CR01   : Capture/compare register 01
CR50   : 8-bit compare register 50
CR51   : 8-bit compare register 51
CRC0   : Capture/compare control register
CSIM30 : Serial operation mode register 0
DAM0   : D/A converter mode register
EGN   : External interrupt falling edge enable register
EGP   : External interrupt rising edge enable register
IF0H   : Interrupt request flag register 0H
IF0L   : Interrupt request flag register 0L
IF1L   : Interrupt request flag register 1L
IMS   : Memory size switching register
IXS   : Internal extension RAM size switching register
LCDC   : LCD display control register
LCDM   : LCD display mode register
MK0H   : Interrupt mask flag register 0H
MK0L   : Interrupt mask flag register 0L
MK1L   : Interrupt mask flag register 1L
OSTS   : Oscillation stabilization time selection register

P0   : Port 0
P1   : Port 1
P4   : Port 4
P8   : Port 8
P9   : Port 9
P10   : Port 10
P11   : Port 11
P12   : Port 12
PCC   : Processor clock contrtol register
PF8   : Port function register 8
PF9   : Port function register 9
PF10   : Port function register 10
PF11   : Port function register 11
PF12   : Port function register 12
PFM   : Power-fail compare mode register
PFT   : Power-fail compare threshold value register
PM0   : Port mode register 0
PM4   : Port mode register 4
PM8   : Port mode register 8
PM9   : Port mode register 9
PM10   : Port mode register 10
PM11   : Port mode register 11
PM12   : Port mode register 12
PR0H   : Priority specify flag register 0H
PR0L   : Priority specify flag register 0L
PR1L   : Priority specify flag register 1L
PRM0   : Prescaler mode register 0
PSW   : Program status word
RXB0   : Receive buffer register
RXS0   : Receive shift register

SAR : Successive approximation register

SGAM : Sound generator amplitude register

SGBC : Sound generator buzzer control register

SGCR : Sound generator control register

SIO30 : Serial I/O shift register 30

TCL50 : Timer clock selection register 50

TCL51 : Timer clock selection register 51

TM0 : 16-bit timer register 0

TM50 : 8-bit counter 50

TM51 : 8-bit counter 51

TMC0 : 16-bit timer mode control register 0

TMC50 : 8-bit timer mode control register 50

TMC51 : 8-bit timer mode control register 51

TOC0 : 16-bit timer output control register

TXS0 : Transmit shift register

UDLCCL: UDL clock control register

WDCS : Watchdog timer clock selection register

WDTM : Watchdog timer mode register

WTM : Watch timer mode control register

[Memo]

Appendix D Revision History

The following shows the revision history up to present. Application portions signifies the chapter of each edition.

Edition No.	Major items revised	Revised Sections

Appendix D Revision History

Edition No.	Major items revised	Revised Sections

[Memo]

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	**Hong Kong, Philippines, Oceania** NEC Electronics Hong Kong Ltd. Fax: +852-2886-9022/9044	**Asian Nations except Philippines** NEC Electronics Singapore Pte. Ltd. Fax: +65-250-3583
Europe NEC Electronics (Europe) GmbH Technical Documentation Dept. Fax: +49-211-6503-274	**Korea** NEC Electronics Hong Kong Ltd. Seoul Branch Fax: 02-528-4411	**Japan** NEC Semiconductor Technical Hotline Fax: 044-548-7900
South America NEC do Brasil S.A. Fax: +55-11-6465-6829	**Taiwan** NEC Electronics Taiwan Ltd. Fax: 02-2719-5951	

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	☐	☐	☐	☐

CS 99.1